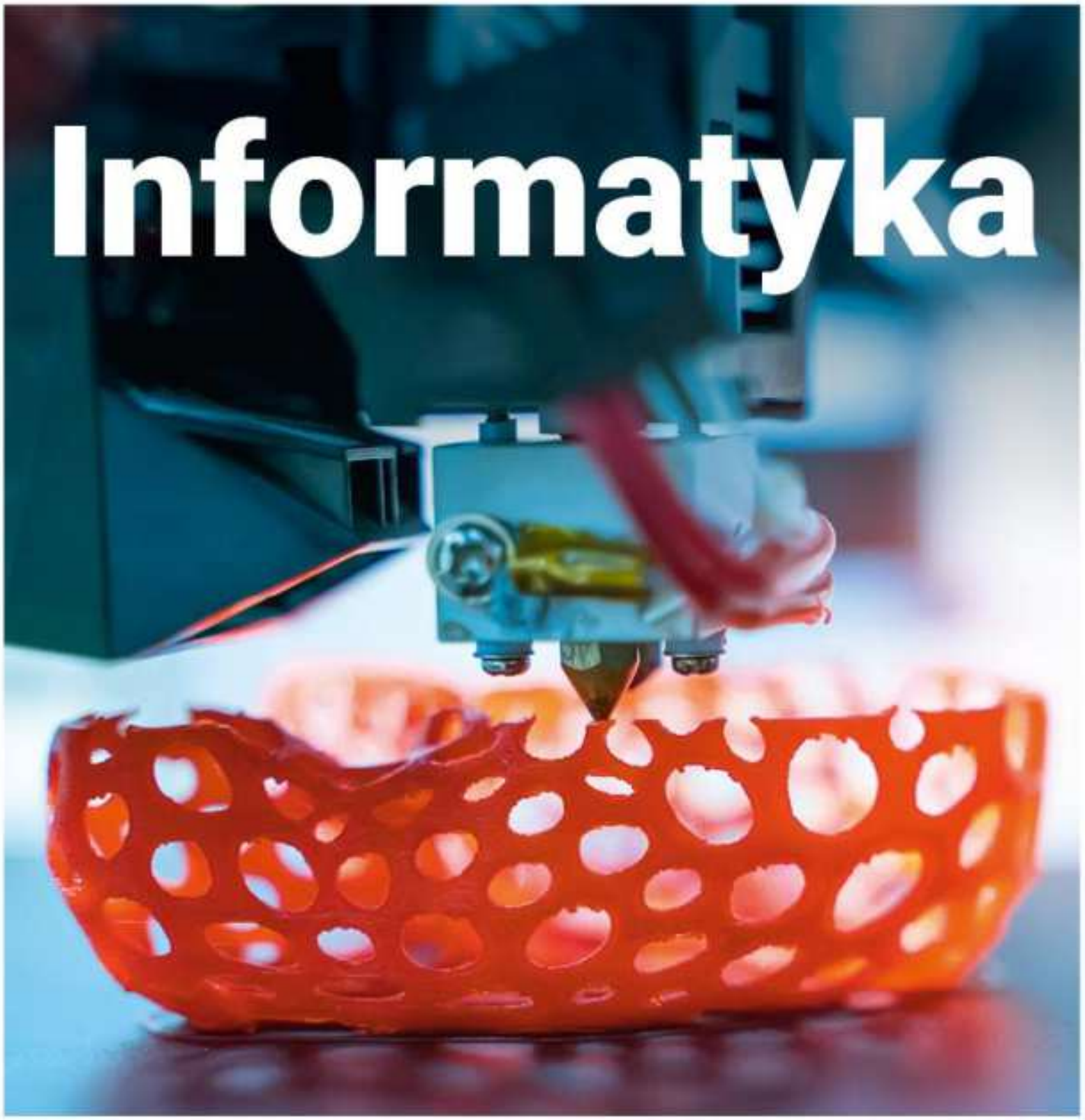
OPERON
Edukacja jest podróżą

ZAKRES ROZSZERZONY

# Informatyka



Zmiana MEN 2024

Oznaczono usunięte i zmienione treści

1

Podręcznik dla **szkół ponadpodstawowych**

Wojciech Hermanowski

Sławomir Sidor

# Informatyka

Podręcznik dla **szkół ponadpodstawowych**

1

ZAKRES ROZSZERZONY

OPERON
Edukacja jest podróżą

2024

Projekt okładki: Paweł Kwacz
Redaktor prowadzący: Sebastian Przybyszewski
Redakcja językowa: zespół
Redakcja graficzna i skład: zespół
Korekta: Katarzyna Kupracz
Fotoedycja: Marta Jasińska
Zdjęcia: Shutterstock

OPERON Sp. z o.o. oświadcza, że z należytą starannością podjął wszelkie działania zmierzające do powiadomienia podmiotów majątkowych praw autorskich do utworów słownych i fotograficznych wykorzystanych w podręczniku o zamiarze ich publikacji celem uzyskania od twórców lub ich następców prawnych wymaganej zgody za przysługującym wynagrodzeniem w stosownej wysokości. Osoby, których prawa w sposób niezawiniony przez wydawnictwo zostały naruszone, prosimy o bezpośredni kontakt z wydawcą.

Zastrzeżonych nazw firm użyto w tym podręczniku wyłącznie w celu identyfikacji.

Podręcznik dopuszczony do użytku szkolnego przez ministra właściwego do spraw oświaty i wychowania i wpisany do wykazu podręczników przeznaczonych do kształcenia ogólnego do nauczania informatyki w zakresie rozszerzonym na poziomie szkoły ponadpodstawowej, na podstawie opinii rzeczoznawców: dr. Tomasza Huka, dr. Przemysława Ogonowskiego, mgr Teresy Kosyry-Cieślak.

Etap edukacyjny: III
Typ szkoły: szkoła ponadpodstawowa
Rok dopuszczenia: 2019
**Numer dopuszczenia: 1048/1/2019**

© Copyright by OPERON Sp. z o.o. & Wojciech Hermanowski, Sławomir Sidor
Gdynia 2019
Wszelkie prawa zastrzeżone. Kopiowanie w całości lub we fragmentach bez zgody wydawcy zabronione.
N6845

Wydawca:
OPERON Sp. z o.o.
81-541 Gdynia, al. Zwycięstwa 96/98
tel. centrali 58 679 00 00
e-mail: info@operon.pl
www.operon.pl

ISBN 978-83-7879-967-2

Zgodnie z decyzją MEN od września 2024 r. obowiązuje zmieniona, uszczuplona o ok. 20% podstawa programowa. W związku z tym **w podręczniku znalazły się stosowne oznaczenia**.

Takim symbolem oznaczono treści usunięte z podstawy programowej.

Takim symbolem oznaczono treści, które zostały zmienione. Nauczyciel przedstawi uczniowi zmiany i wyjaśni, czego powinien się nauczyć.

# Spis treści


## I. Przestrzeganie prawa i zasad bezpieczeństwa podczas pracy przy komputerze

1. Bądź uczciwy, czyli przestrzeganie prawa w świecie informatyki ..... 10
2. 8, 16, 32, 64, czyli jak rozwój technologii wpływa na rozwój społeczeństw ..... 18
3. Chrońmy informacje, czyli techniki uwierzytelniania, kryptografia i podpis elektroniczny ..... 24
4. Praktycznie i teoretycznie, czyli trendy historyczne w rozwoju technologii ..... 30
5. Kim jestem, czyli jak bezpiecznie budować wizerunek w sieci ..... 38

Podsumowanie przed sprawdzianem ..... 44

## II. Podstawy programowania i środowisko programistyczne

6. Przypomnij sobie, czyli podstawy tworzenia algorytmów ..... 46
7. Algorytmy. Utrwalamy swoją wiedzę ..... 52
8. Pozycyjne reprezentacje liczb, czyli tablice w algorytmice ..... 57
9. Narzędzia, czyli jak dobrać i skonfigurować środowisko programistyczne ..... 60
10. C z plusami, czyli uruchamiamy pierwsze programy w C++ ..... 71
11. C z plusem, czyli podstawy programowania ..... 78
12. C z plusem, czyli podstaw programowania ciąg dalszy ..... 86

Podsumowanie przed sprawdzianem ..... 92

## III. Programowanie i algorytmy

13. Wspólna miara, czyli podstawy złożoności obliczeniowej ..... 94
14. Przez 1 i siebie, czyli jak zbadać, czy dana liczba jest liczbą pierwszą ..... 98
15. Która pierwsza, czyli jak program bada liczby ..... 103
16. Więcej pierwszych, czyli generujemy liczby ..... 109
17. Największy i najmniejszy, czyli jak znaleźć NWD i NWW ..... 113
18. Euklides i rekurencja, czyli szukamy NWD ..... 119
19. Metoda iteracji, czyli algorytm Euklidesa w innej postaci ..... 124
20. Dodawanie ułamków, czyli jak wykorzystać NWW i NWD w programie komputerowym ..... 131
21. Skracamy i wyłączamy, czyli dodawania ułamków ciąg dalszy ..... 136
22. Szyfr Cezara, czyli poznajemy szyfr przesuwający ..... 141
23. Konspiracja, czyli tworzymy własne szyfry ..... 146

24. Z kluczem publicznym, czyli bezpieczniejsze szyfrowanie ..... 154
25. Szyfrowanie zachłanne, czyli kody Huffmana ..... 158
26. Tworzenie szyfrów przestawieniowych, czyli kodowanie znaków o zmiennej długości ..... 161
27. Automat wydaje, czyli jak dynamicznie wydawać resztę ..... 166
28. Wydajemy resztę, czyli jak wykorzystać metodę zachłanną ..... 176
29. Co z tym groszem, czyli błąd zaokrąglenia i błąd przybliżenia ..... 179
30. Sprawdzam, czyli porównywanie tekstów i wyszukiwanie wzorca w tekście metodą naiwną ..... 185
31. Od ostatniego, czyli szukamy wzorca tekstu metodą Boyera–Moore'a ..... 188
32. Z lewej i z prawej, czyli szukamy palindromów ..... 192
33. Przestawianie liter, czyli jak stworzyć anagram ..... 195
34. Od A do Z, czyli alfabetyczne sortowanie wyrazów ..... 198
35. Odwrotna notacja polska, czyli zapisujemy wyrażenia arytmetyczne ..... 202
36. Naiwny jak algorytm, czyli szukamy najmniejszego i największego elementu metodą naiwną ..... 205
37. Dwie dodatkowe tablice, czyli stosujemy algorytm optymalny ..... 207
38. Dziel i zwyciężaj, czyli sortowanie metodą scalania ..... 209
39. Raz krócej, raz dłużej, czyli sortowanie kubełkowe ..... 213
Podsumowanie przed sprawdzianem ..... 215

## IV. Grafika i druk

40. Modele w przestrzeni, czyli podstawy druku 3D ..... 220
41. Modelujemy, czyli jak projektować obiekty 3D ..... 227
42. Wizualizacja pomysłów, czyli projektujemy dom w edytorze 3D ..... 236
43. Urządzamy mieszkanie, czyli wizualizacje 3D ..... 245
44. Wypukłości nie tylko 3D, czyli tworzymy elementy graficzne publikacji ..... 250
45. Dynamicznie, czyli jak tworzyć ruchome wizualizacje ..... 259
46. Z wydruku do komputera, czyli skanujemy i odczytujemy dokumenty ..... 262
47. Dokumentujemy wydarzenia, czyli aktywna praca z aparatem fotograficznym ..... 270
48. Szturmowiec w chmurze, czyli poprawiamy zdjęcia w edytorze grafiki rastrowej ..... 278
49. Tego tu nie było, czyli poprawiamy rzeczywistość na fotografiach ..... 286
50. Własny film, czyli jak twórczo wykorzystać kamerę ..... 291
51. Ze stratą lub bez, czyli jak kompresować pliki ..... 300
52. Klatki do komputera, czyli jak transmitować strumieniowo ..... 304
Podsumowanie przed sprawdzianem ..... 311

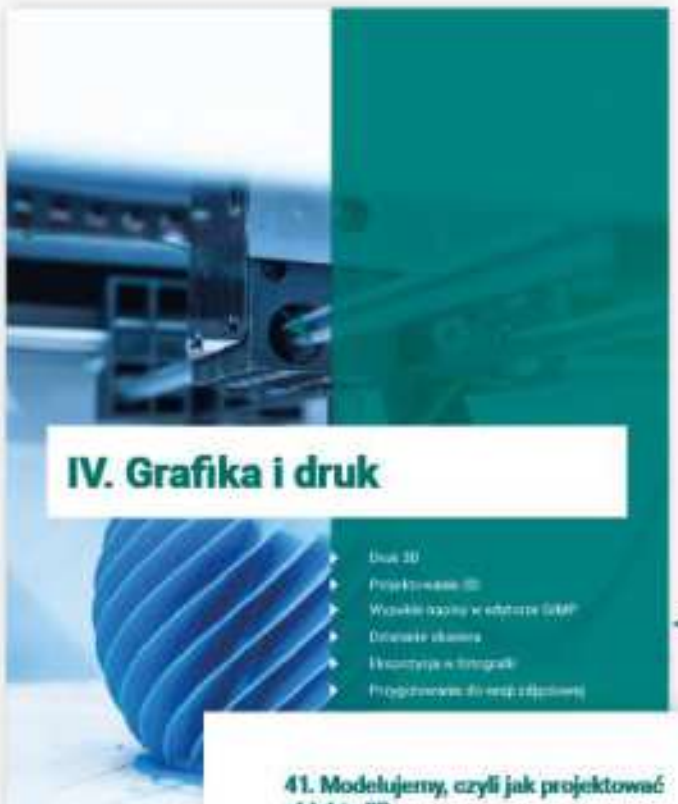
## V. Komputer w sieci


53. Nie wszystko jest takie oczywiste, czyli jak działa internet .....314
54. Zabezpieczenia firewall.....323
55. Sieć to nie tylko internet, czyli poznajemy topologię sieci komputerowych .....325
56. Warstwowy model sieci komputerowej .....330
57. Funkcje urządzeń i protokoły .....333
58. Kto tam, czyli identyfikujemy komputery w sieci .....336
59. Kupujemy świadomie, czyli poznajemy parametry urządzeń peryferyjnych.....341
60. Projektuję rozbudowę i zakup nowego zestawu.....349
Podsumowanie przed sprawdzianem.....352

# Wstęp

## Jak jest zbudowany nasz podręcznik

Zanim zaczniecie korzystać z podręcznika, zapoznajcie się z jego budową i przyjrzyjcie się wszystkim jego elementom. Zostały one skonstruowane tak, by ułatwić wam naukę i pomóc w błyskawicznym znalezieniu potrzebnych informacji.




Każdy dział podręcznika otwiera spis tematów z danego obszaru.



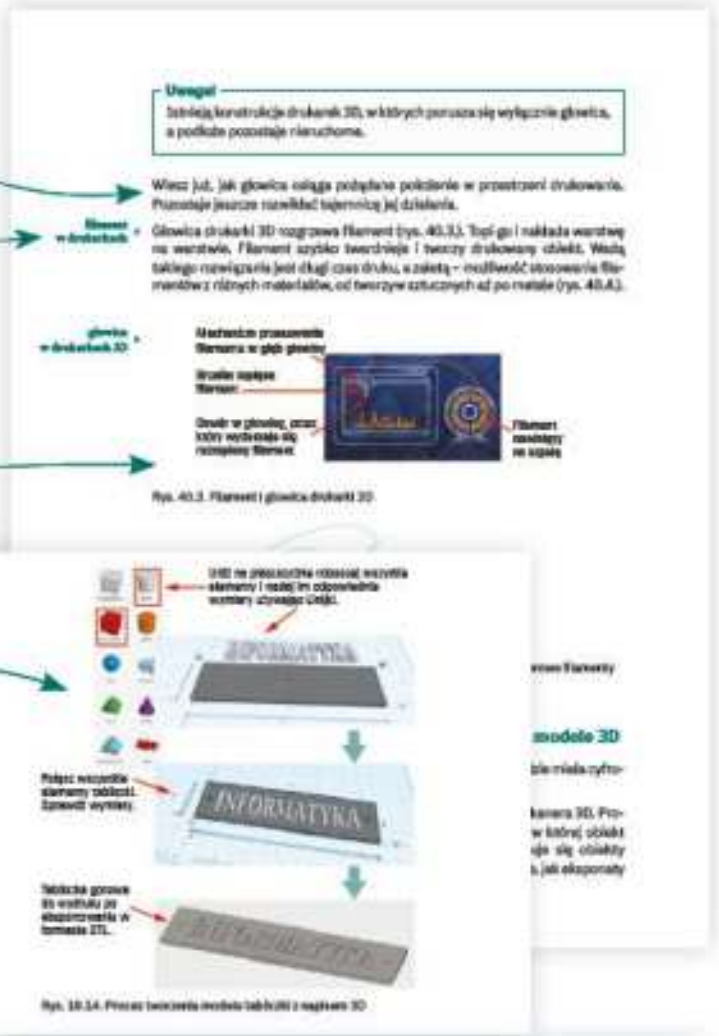

Bezpośrednio pod tytułem rozdziału znajdziecie moduł „Na tej lekcji", w którym wypunktowane są cele danego tematu.

W tekście pojawiają się wytłuszczenia. Dzięki nim łatwiej odnajdziecie istotne zagadnienia, pojęcia i nazwy.

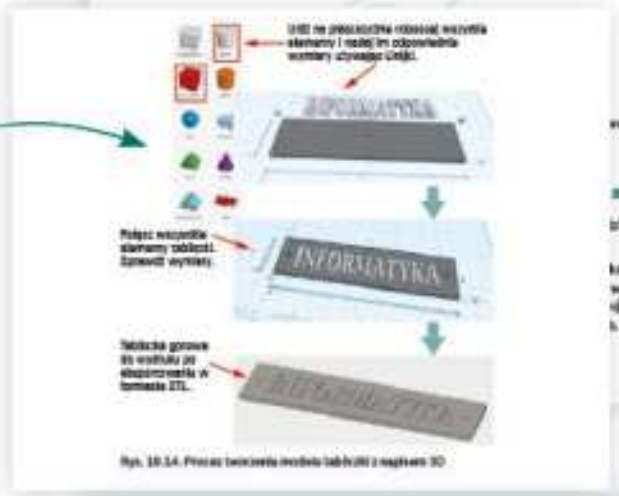
**Przystępny język publikacji** ułatwi wam zrozumienie treści i powtarzanie materiału.

**Hasła na marginesach** to treści obowiązkowe do zapamiętania z lekcji. Zostaną one powtórzone w podsumowaniu lekcji.

**Schematy** i rysunki pomogą wam usystematyzować wiedzę.




**Zrzuty ekranu** pełnią funkcję tutoriali (samouczków), dzięki którym będziecie mogli szybko zastosować wskazówki w praktyce.

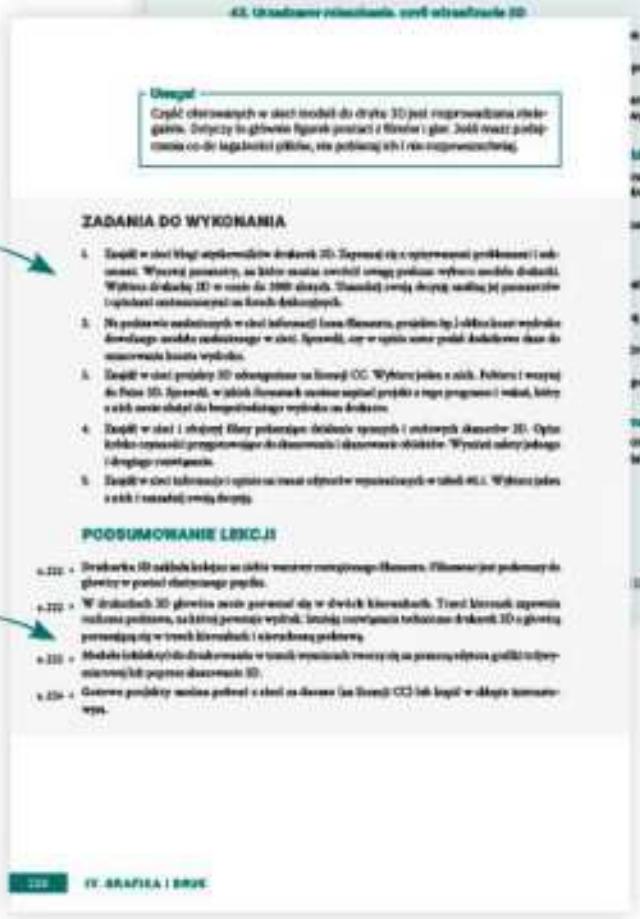



Podsumowanie przed sprawdzaniem zawiera najważniejsze treści do zapamiętania z działu. Jeśli ich nie pamiętacie, wróćcie na stronę wskazaną obok i poszukajcie hasła na marginesie.




Zadania do wykonania przećwiczycie w toku lekcji lub wykorzystacie do nauki w domu albo przed kartkówką czy sprawdzianem. Gwiazdką oznaczono trudniejsze zadania.

**W podręczniku nie należy umieszczać pisemnych odpowiedzi do zadań.**




Podsumowanie lekcji to esencja najważniejszych treści do zapamiętania Jeśli ich nie pamiętacie, wróćcie na stronę wskazaną obok. Znajdziecie tam na marginesie odpowiednie hasło, a w treści lekcji – jego omówienie.

# I. Przestrzeganie prawa i zasad bezpieczeństwa podczas pracy przy komputerze

- Prawo w informatyce
- Ustawa o prawie autorskim
- Postęp technologiczny a wydajność komputera
- E-learning
- Wizerunek w sieci
- Prywatność w sieci

# 1. Bądź uczciwy, czyli przestrzeganie prawa w świecie informatyki

**NA TEJ LEKCJI:**

- dowiesz się, jak legalnie korzystać z cudzych publikacji;
- poznasz ustawę o prawie autorskim i prawach pokrewnych.

Korzystasz z internetu od wielu lat – czytasz informacje, oglądasz filmy, słuchasz muzyki. Czy wiesz, skąd biorą się te materiały? Kto je tworzy? Kto publikuje? Czy robi to za darmo, czy może dostaje wynagrodzenie, a może utrzymuje się z tantiem? To bardzo ważne pytania w kontekście uczciwości i przestrzegania prawa w sieci. Co więc jest legalne, a co nie? Co jest dopuszczalne, a czego nie należy tolerować?

## 1.1. Prawa majątkowe

Nie trzeba być hakerem lub oszustem internetowym, by złamać prawo. Często niestety robi się to nieświadomie, ale – jak wiadomo – nieznajomość prawa nie zwalnia z odpowiedzialności. Być może zastanawiasz się teraz, czy ty również popełniasz przestępstwa lub wykroczenia, nie wiedząc o tym. Spróbuj samodzielnie odpowiedzieć na to pytanie, rozważając poniższe przykłady.

**Przykład 1.**

Złodziej wchodzi do sklepu z płytami z muzyką lub filmami. Podchodzi do półki, zdejmuje z niej płytę i chowa pod kurtkę. Wychodzi ze sklepu bez zapłacenia za towar, przez co popełnia przestępstwo. Czy pobierając utwory z tej płyty w postaci plików bez wnoszenia opłaty, dokonujesz takiego samego czynu, jak ów złodziej?

pobieranie treści z internetu

Odpowiedź na to pytanie jest złożona. Z jednej strony bezpłatne pobranie piosenek niczym nie różni się od wyniesienia ze sklepu płyty i również jest moralnie naganne. Jednak polskie prawo pozwala na pobieranie utworów opublikowanych w sieci na własny użytek. W takim przypadku mówi się o dozwolonym użytku. Powinniśmy być przy tym świadomi, że w ten sposób pozbawiamy twórcę i wydawcę należnego zysku. Jeśli pliki pobieramy na przykład z sieci Torrent, to rozpoczęcie pobierania jest równoznaczne z udostępnianiem pliku (a nawet fragmentu pliku) innym użytkownikom tej sieci. Ponieważ udostępnianie treści chronionych prawem autorskim jest przestępstwem, nieświadomie można stać się przestępcą. Należy liczyć się z konsekwencjami złamania prawa.

**Przykład 2.**

Student musi wykonać pracę zaliczeniową. Do jej wykonania potrzebuje programu komputerowego, na który go nie stać. Pobiera więc piracką wersję tego programu, instaluje na swoim komputerze i tworzy swoją pracę. Czy jest złodziejem?

Każde pobranie, zainstalowanie lub użycie pirackiej wersji programu komputerowego jest przestępstwem.

**Przykład 3.**

Ukazała się najnowsza wersja pewnej popularnej gry komputerowej. Wiadomo, że tego typu wydawnictwa są kosztowne. Pewien gracz pobiera piracką wersję, instaluje na komputerze i gra. Czy jest złodziejem?

W świetle prawa każde ściągnięcie z internetu programu komputerowego lub gry bez zgody właściciela praw (czyli np. pobranie z pirackiego serwera) jest złamaniem prawa. To taka sama kradzież jak wyniesienie pudełka z grą ze sklepu bez zapłaty. Nie ma tu wyjątków. Współczesne gry aktywnie korzystają z internetu, w związku z tym komputer z pirackim programem jest łatwy do namierzenia.

**Przykład 4.**

Pewien uczeń przygotowuje wypracowanie. Znajduje w sieci pracę na podobny temat. Pobiera ją, przepisuje do zeszytu i oddaje nauczycielowi do sprawdzenia. Czy jest złodziejem?

Z pewnością dokonał kradzieży czyjejś pracy. Nie szanuje też praw autorskich, ponieważ podpisał ją własnym nazwiskiem. Popełnił w ten sposób plagiat. Ustawa o prawach autorskich i prawach pokrewnych w takim przypadku mówi, że:

**Art. 115.**

1. Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3.
2. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nagranie.

Plagiat jest więc przestępstwem przeciwko prawu. Jeśli taki czyn popełni student lub naukowiec, poniesie jeszcze kary przewidziane przez uczelnię i oczywiście utraci autorytet i wiarygodność. W przypadku ucznia oczywistą reakcją jest ocena niedostateczna i obniżenie oceny z zachowania. Nawet jeśli oszustwo się uda, to pozostaje kwestia sensowności takiego działania. Kopiując pracę, nie uczysz się, a więc jej cel nie zostaje spełniony.

**Przykład 5.**

W kabriolecie na siedzeniu leży telefon. Przechodzień dostrzega go i zabiera. Czy to, że urządzenie było dostępne w łatwy sposób, usprawiedliwia kradzież? Czy to, że komercyjne programy, gry i aplikacje są dostępne w sieci, oznacza, że można je pobierać bez opłaty?

Odpowiedź na to pytanie jest oczywista.




Rys. 1.1. Przykłady kradzieży za pośrednictwem internetu

A teraz zastanów się i odpowiedz na jeszcze jedno pytanie: czy kiedyś dopuściłeś/dopuściłaś się kradzieży. Wyciągnij wnioski.

## 1.2. Prawa autorskie, czyli jak legalnie korzystać z cudzych publikacji

W przepisach używa się pojęć, które musisz zrozumieć, aby znać prawo. Wiesz, czym jest utwór? Warto to wiedzieć, ponieważ właśnie utwory są chronione prawem. Pojęcie to zdefiniowano w rozdziale 1 Ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (DzU 1994, nr 24, poz. 83, z późn. zm.).

prawa autorskie

**Rozdział 1.**
**Przedmiot prawa autorskiego**

**Art. 1.**

1. Przedmiotem prawa autorskiego jest każdy przejaw działalności twórczej o indywidualnym charakterze, ustalony w jakiejkolwiek postaci, niezależnie od wartości, przeznaczenia i sposobu wyrażenia (utwór).
2. W szczególności przedmiotem prawa autorskiego są utwory:
   1) wyrażone słowem, symbolami matematycznymi, znakami graficznymi (literackie, publicystyczne, naukowe, kartograficzne oraz programy komputerowe);
   2) plastyczne;
   3) fotograficzne;
   4) lutnicze;
   5) wzornictwa przemysłowego;
   6) architektoniczne, architektoniczno-urbanistyczne i urbanistyczne;
   7) muzyczne i słowno-muzyczne;
   8) sceniczne, sceniczno-muzyczne, choreograficzne i pantomimiczne;
   9) audiowizualne (w tym filmowe).

Zatem programy komputerowe, a więc aplikacje, gry, programy sieciowe itp., są dziełami chronionymi. Teraz warto zadać sobie pytanie, w jaki sposób zabezpieczyć swoje prawa na przykład jako programista. Oczywiście to także wyjaśnia ustawa.

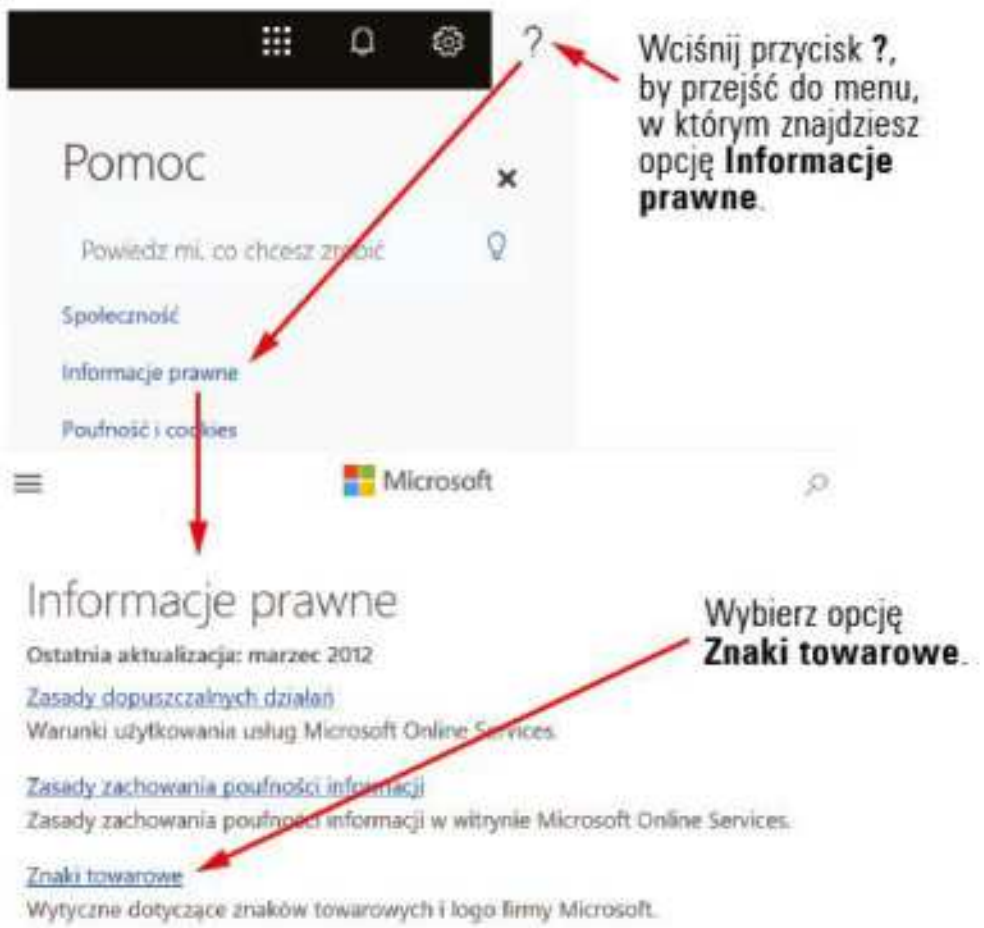
**Art. 1.**

$2^1$. Ochroną objęty może być wyłącznie sposób wyrażenia; nie są objęte ochroną odkrycia, idee, procedury, metody i zasady działania oraz koncepcje matematyczne.
3. Utwór jest przedmiotem prawa autorskiego od chwili ustalenia, chociażby miał postać nieukończoną.
4. Ochrona przysługuje twórcy niezależnie od spełnienia jakichkolwiek formalności.

Warto zastanowić się, na czym polega ochrona wytworów ludzkiego umysłu wymienionych w punkcie 3. Aby chronić je prawem, należy je opatentować w Urzędzie Patentowym. Może to zrobić każdy po złożeniu odpowiedniego zgłoszenia i dokumentacji opisującej przedmiot patentu.

Oczywiście przepisy szczegółowe są skomplikowane i często nie zdajemy sobie sprawy, że niektóre elementy urządzeń komputerowych (także innych), części kodu programów komputerowych, technologie i rozwiązania stosowane w układach scalonych itp. są chronione prawem patentowym lub autorskim. Zazwyczaj mowa o tym w informacji dołączonej do urządzenia czy programu zawierającej słowa: „Wszelkie prawa zastrzeżone". Prawo patentowe chroni także znaki towarowe, w tym nazwy własne programów, na przykład Office 365, ikony systemu operacyjnego, programów komputerowych, a nawet kroje liter używane w grach lub filmach.

Uruchom dowolny program, na przykład MS Word lub chmurę Office 365, i znajdź zapisy informujące o ochronie praw. Jak myślisz, w której opcji menu je znajdziesz? Zazwyczaj są one zapisane w **Pomocy** lub informacjach **O programie**. Nie będziesz mieć kłopotów z ich odnalezieniem (rys. 1.2.).




Rys. 1.2. Opcja wyświetlania wszystkich znaków towarowych chronionych prawem będących własnością Microsoft i wykorzystywanych w Office 365

**Uwaga!**

W przypadku Office 365 znaków towarowych jest tak wiele, że posegregowano je alfabetycznie i utworzono menu wyszukiwania. Aby je odczytać, należy wybrać odpowiednią literę.

Należy odróżnić prawa osobiste od praw majątkowych do utworów. Pierwsze są niezbywalne i nie wygasają, co oznacza, że pozostają zawsze przy jego autorze – nawet gdy sprzeda on lub odstąpi prawa majątkowe do dysponowania utworem i zarabiania na jego publikowaniu. Natomiast prawa majątkowe są zbywalne, a po ich nabyciu można czerpać z nich zyski. Należy jednak oznaczyć utwór nazwiskiem jego autora, tak jak w przypadku książek (chyba że w umowie zapisano inne warunki). Prawa majątkowe wygasają po 70 latach od śmierci twórcy lub pierwszego upowszechnienia dzieła.

Z doświadczenia wiesz, że wiele utworów, artykułów czy opracowań zawiera fragmenty innych dzieł. Czy to jest legalne? Tak – ale po spełnieniu pewnych warunków. Określono je w artykule 29 ustawy o prawie autorskim i prawach pokrewnych:

ustawa o prawie autorskim i prawach pokrewnych

> **Art. 29.** Wolno przytaczać w utworach stanowiących samoistną całość urywki rozpowszechnionych utworów oraz rozpowszechnione utwory plastyczne, utwory fotograficzne lub drobne utwory w całości, w zakresie uzasadnionym celami cytatu, takimi jak wyjaśnianie, polemika, analiza krytyczna lub naukowa, nauczanie lub prawami gatunku twórczości.

plagiat

Cytat musi więc spełniać funkcję posiłkową, na przykład stanowić dowód na poparcie tezy, ilustrować omawiane zagadnienia, być przedmiotem analizy, recenzji lub polemiki. Cytat należy wyraźnie oznaczyć. Trzeba pamiętać, że nie wolno dokonywać w nim jakichkolwiek zmian. Najważniejsze jest jednak wskazanie autora. Przykład tekstu, w którym można zacytować utwór, to recenzja. Jej autor, omawiając dzieło, często posługuje się jego fragmentami. Twórcy korzystają również z innych form zapożyczeń. W porozumieniu z autorem przetwarzają dzieło za jego zgodą. Tak dzieje się z piosenkami, które wykonują i interpretują różni wykonawcy.

licencja CC

Co zrobić, gdy tworzymy projekt i potrzebujemy na przykład ilustracji na dany temat lub podkładu muzycznego do filmu? Najbezpieczniejszym rozwiązaniem jest skorzystanie z treści udostępnionych na licencjach CC (Creative Commons, czytaj: kriejtiw komons). Wszystkie treści na licencji CC są darmowe i nie wymagają pozwolenia na użycie. Jeśli zdecydujesz się skorzystać z takiego dzieła, sprawdź dokładnie zasady, na których będzie to możliwe. Istnieją bowiem różne odmiany tego typu licencji. Na przykład jeśli ktoś chce wykorzystać utwór muzyczny na licencji CC, w napisach końcowych lub w innym miejscu musi wymienić autora muzyki.

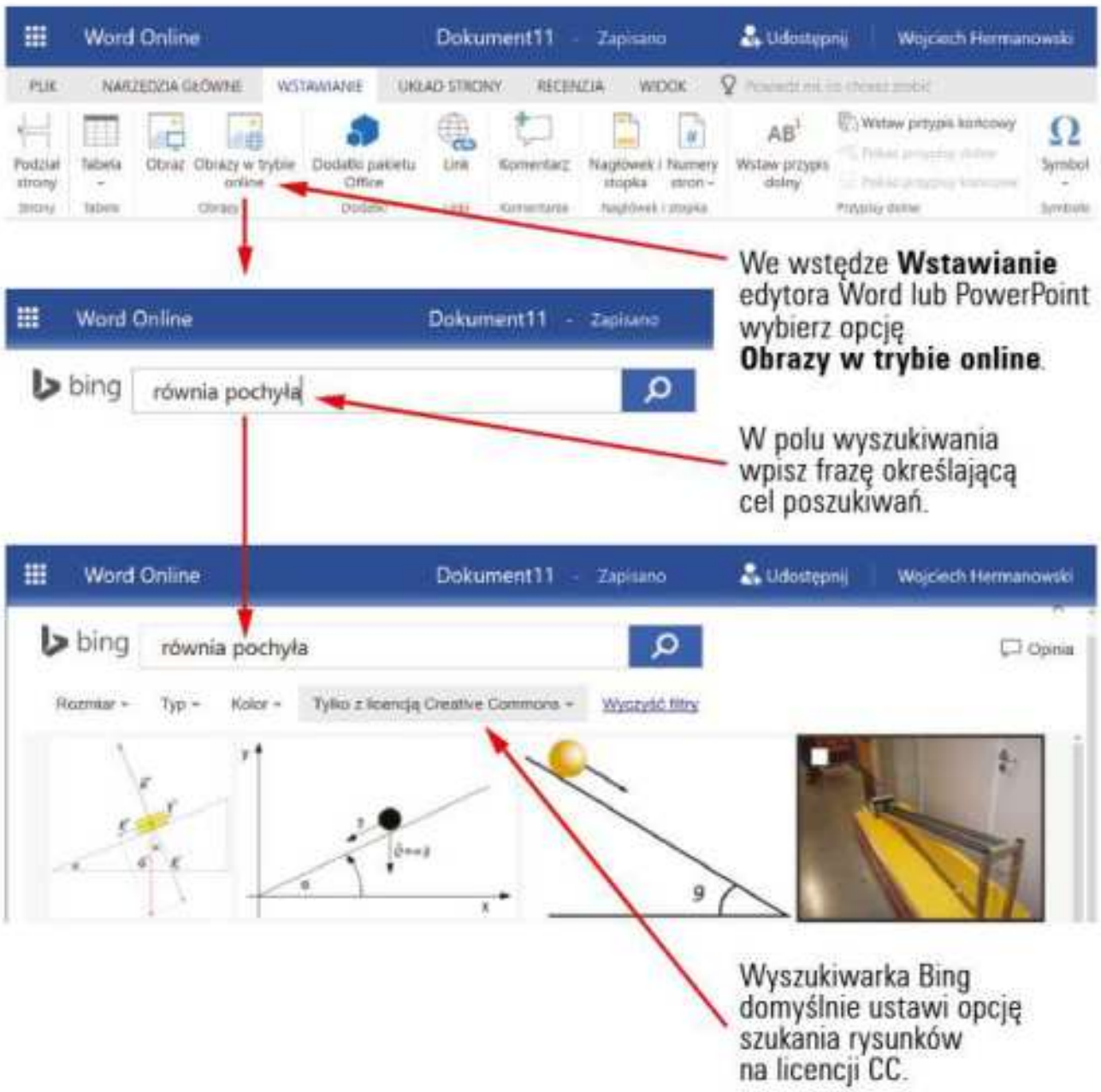
W trakcie poszukiwań darmowych utworów możesz spotkać się z pojęciem „domena publiczna". To zbiór utworów, z których można korzystać bez ograniczeń, na przykład z powodu wygaśnięcia praw majątkowych. W Polsce w tym rozumieniu stosuje się pojęcie „własność publiczna państwa". Jeśli chcesz wykonać i nagrać utwór powstały w XIX wieku, możesz to zrobić bez opłat.

Gdzie szukać treści na licencji CC? Oczywiście możesz użyć dowolnej wyszukiwarki i dodać do szukanych haseł litery CC lub skorzystać z zaawansowanych opcji wyszukiwania określających prawo do użytkowania. W sieci można znaleźć także wyszukiwarki darmowych treści. Przykładem może być: *https://search.creativecommons.org*.

**Wykonaj ćwiczenie:**

Jeśli korzystasz z pakietu MS Office, Office 365 lub darmowej chmury OneDrive, masz ułatwione zadanie, ponieważ edytory w tej chmurze mają możliwość wyszukiwania treści z domyślnie ustawioną opcją licencji CC.

Uruchom edytor tekstu lub prezentacji. Wyszukaj taką ilustrację na temat równi pochyłej, której możesz legalnie użyć w swoim dokumencie (rys. 1.3.).




Rys. 1.3. Wyszukiwanie ilustracji na licencji CC z zastosowaniem opcji edytora Word

**Uwaga!**

Zastosuj zasadę ograniczonego zaufania i sprawdzaj, czy wyszukane przez Bing (rys. 1.3.) ilustracje rzeczywiście udostępniono na licencji CC. Zanotuj także autora ilustracji.

## ZADANIA DO WYKONANIA

1. Użyj wyszukiwarki i znajdź strony internetowe o prawie autorskim. Oceń ich wiarygodność i wybierz godne zaufania, na przykład portale prawnicze. Odrzuć fora dyskusyjne, ponieważ każdy uczestnik dyskusji może wpisywać w nich swoje opinie, a nie wszystkie są wiarygodne. Zapoznaj się z treścią artykułów ustawy mówiących o prawie do cytatu i dozwolonym użytku. Określ, czy twoje dotychczasowe działania w sieci są zgodne z ustawą.
2. Za pomocą wyszukiwarki znajdź w sieci ilustracje związane z ostatnio omawianym tematem na lekcjach fizyki. Upewnij się, że wybrana ilustracja jest udostępniona na licencji CC. Wklej ilustrację do dokumentu tekstowego lub prezentacji. Podpisz ją nazwiskiem i imieniem autora.
3. Napisz (w zeszycie lub edytorze tekstu) krótką pracę o konsekwencjach pobierania i używania pirackich kopii programów lub gier komputerowych. W pracy użyj dozwolonych prawem cytatów z wiarygodnych źródeł. Zwróć uwagę na to, by posłużyć się cytatami zgodnie z prawem.
4. Wyszukaj w sieci artykuły prasowe dotyczące plagiatów z różnych dziedzin. Zwróć uwagę na przykład na prace naukowe, filmy, muzykę. Dowiedz się, jakie konsekwencje ponieśli oszuści.
5. Sprawdź, które znaki towarowe zastrzegł producent twojej ulubionej gry komputerowej. Postaraj się odnaleźć je w grze.

## PODSUMOWANIE LEKCJI

- Prawo do dozwolonego użytku stanowi, że można legalnie **pobierać treści z internetu,** np. filmy, muzykę, rysunki, na własny użytek. Jeśli jednak robisz to za pośrednictwem sieci takich jak Torrent, pobierając plik, jednocześnie udostępniasz go innym użytkownikom, a to jest już przestępstwem. Każde pobranie, zainstalowanie lub **użycie pirackiej wersji** programu komputerowego, w tym gry, **jest przestępstwem**. s. 10
- **Przestrzeganie prawa autorskiego** jest podstawowym obowiązkiem każdego korzystającego z zasobów sieciowych. Nieznajomość prawa nie chroni przed konsekwencjami jego naruszenia. s. 13
- Podstawowym dokumentem określającym prawa do utworów jest **ustawa o prawie autorskim i prawach pokrewnych**. Możesz używać w swoich pracach fragmentów lub całości niewielkich utworów na zasadzie cytatu. s. 15
- Możesz używać w swoich projektach utworów rozpowszechnianych na **licencji CC** (Creative Commons). Najczęściej spotykanym warunkiem skorzystania z takiego utworu jest umieszczenie w dziele imienia i nazwiska autora. s. 15
- **Plagiat** jest przestępstwem. Nie wolno kopiować fragmentów lub całości prac innych autorów i publikować ich jako swoich. s. 15

# 2. 8, 16, 32, 64, czyli jak rozwój technologii wpływa na rozwój społeczeństw

**NA TEJ LEKCJI:**

- dowiesz się, jak postęp technologiczny wpływa na wydajność komputera;
- dowiesz się, do czego można dziś wykorzystywać komputery osobiste.

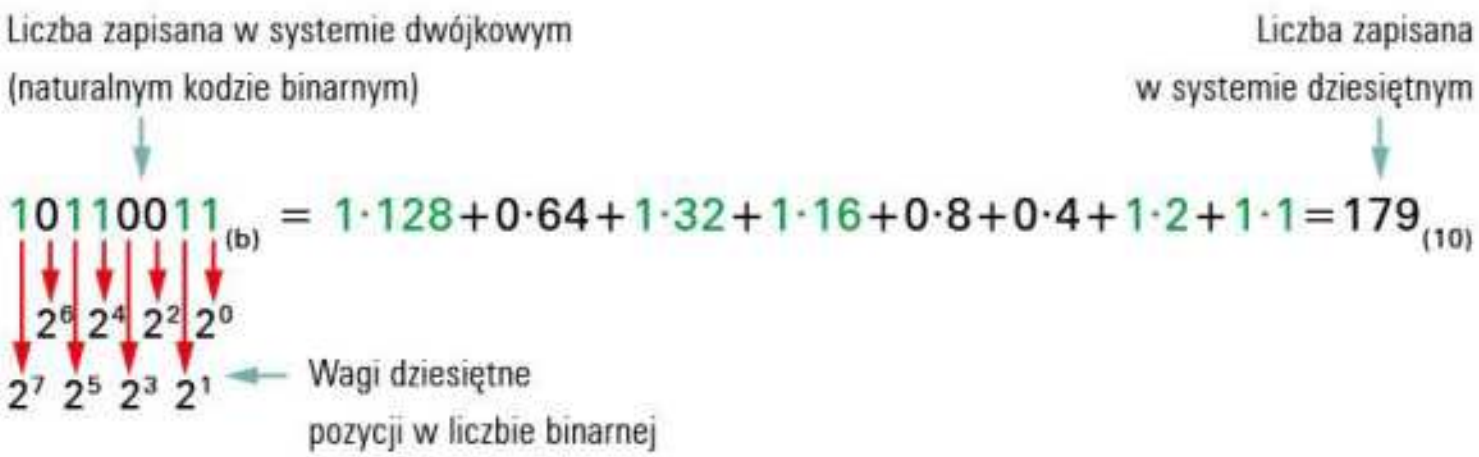
Technologie zmieniają świat. Zgodzisz się z tym zdaniem? Maszyna parowa wpłynęła na rozwój transportu i produkcji masowej, silnik spalinowy dał możliwość szybkiego przemieszczania się po drogach, ale największym osiągnięciem ludzkości było wykorzystanie prądu elektrycznego. Wyobrażasz sobie życie bez niego? Większość osiągnięć technologicznych współczesnego świata przetwarza energię elektryczną. Na co przetwarza ją komputer? Jak zmienił świat?

## 2.1. Postęp technologiczny, czyli jak technologie wpływają na wydajność komputera

postęp technologiczny a wydajność komputera

Czas na wyjaśnienie liczb z tytułu rozdziału: 8, 16, 32 i 64 to liczby ściśle związane z rozwojem technologii mikroprocesorowej. Określają one liczbę bitów liczby binarnej, która może być przechowywana w pamięci, przetwarzana i przesyłana magistralą w mikrokomputerze. Załóżmy, że masz dwa komputery. Jeden z nich operuje na liczbach 8-bitowych (tyle, co pierwsze komputery osobiste i konsolki do gier telewizyjnych), a drugi na 64-bitowych (współczesny komputer osobisty). Umiesz wyznaczyć zakres liczb przetwarzanych w obu maszynach?

Przyda ci się powtórka ze szkoły podstawowej, czyli wiedza o dwójkowym zapisie liczb (w informatyce: NKB, czyli naturalnym kodzie binarnym). Czy pamiętasz, że to kod wagowy, a wartość kolejnych bitów określa kolejna potęga dwójki? To oznacza, że wagi bitów w 8-bitowej liczbie zapisanej w NKB to: 128, 64, 32, 16, 8, 4, 2, 1. Dlaczego w kolejności malejącej? A w jakiej zapisujesz pozycje liczby dziesiętnej? Waga 1 to pierwsza cyfra z prawej, a 10 – to druga z prawej. Tu jest tak samo. Czym jest waga pozycji cyfry w liczbie zarówno w systemie dwójkowym, jak i dziesiętnym? Przypomni ci to rysunek 2.1.




Rys. 2.1. Wagi dziesiętne pozycji liczby binarnej

Wróćmy do naszego problemu. Na rysunku 2.1. przedstawiono liczbę ośmiobitową. Wiadomo, że osiągnęłaby ona największą wartość dla ośmiu jedynek. Jak obliczyć wartość dziesiętną takiej liczby? Oczywiście sumując wagi dziesiętne wszystkich pozycji. Da to liczbę $255_{(10)}$. Z tego wniosek, że procesor 8-bitowy potrafił przetwarzać liczby z zakresu od 0 do $255_{(10)}$. Jak obliczyć liczbę liczb możliwych do zapisania w postaci 64-bitowej? Sumowanie wag byłoby czasochłonne. Spróbuj opracować prosty sposób na obliczanie maksymalnej wartości dziesiętnej liczby zapisanej z użyciem dowolnej liczby bitów. Pomoże ci w tym rysunek 2.2.

$$1111\ 1111_{(b)} = 1\cdot128+1\cdot64+1\cdot32+1\cdot16+1\cdot8+1\cdot4+1\cdot2+1\cdot1=255_{(10)}$$

$$1111\ 1111_{(b)} + 1_{(b)} = 1\ 0000\ 0000_{(b)} = 1\cdot256_{(10)} = 2^8{}_{(10)}$$

Wniosek:
$2^7+2^6+2^5+2^4+2^3+2^2+2^1=2^8-1$

Rys. 2.2. Podpowiedź do rozwiązania problemu obliczania maksymalnej wartości dziesiętnej liczby binarnej

Jeśli spróbowałeś/spróbowałaś rozwiązać problem, to dostrzegasz, że dodanie 1 do maksymalnej liczby (wszystkie pozycje mają wartość 1) zapisanej za pomocą *n* bitów spowoduje zwiększenie liczby bitów do *n*+1. Pozycja ta przybierze wartość 1, a pozostałe 0. Szukany wzór powinien więc mieć postać:

$$L_{max} = 2^{n+1} - 1$$

Maksymalna wartość dziesiętna liczby binarnej zapisana za pomocą *n* bitów

Zgodnie ze wzorem maksymalna liczba dziesiętna zapisana za pomocą 64 bitów to:

$$2^{64} - 1 = 18446744073709551616 - 1 = 18446744073709551615$$

Zwiększenie liczby bitów z 8 do 64 (8-krotne) w słowie przetwarzanym przez procesor lub przesyłanym magistralami zwiększyło aż 72 340 172 838 076 673-krotnie liczbę możliwych do przetwarzania liczb. Dotyczy to także liczby przesyłanych informacji za pośrednictwem magistral.

**Czy teraz doceniasz postęp, jaki nastąpił w technologii informatycznej? Jak bardzo wzrosła wydajność komputerów?**

Postęp w budowie mikroprocesorów i innych układów cyfrowych polega nie tylko na zwiększaniu liczby przetwarzanych bitów, lecz także na unowocześnianiu technologii ich budowy.

Poniżej wymieniono kilka cech postępu technologicznego mających wpływ na funkcjonalność komputerów.

Tab. 2.1. Cechy postępu technologicznego w dziedzinie informatyki

| Cecha | Znaczenie |
|---|---|
| Częstotliwość pracy | W pierwszych komputerach domowych procesory taktowano z częstotliwością od 1 MHz (np. Commodore 64). Współcześnie może ona przekraczać 4 GHz, czyli procesor pracuje około 4000 razy szybciej. Znacząco wpływa to na wydajność komputerów i możliwość współpracy z zaawansowanymi systemami operacyjnymi. |
| Pobór energii i zasilanie | Zmiana technologii wytwarzania układów scalonych znacznie zmniejszyła zapotrzebowanie na energię. Obecnie komputery przenośne mogą pracować nawet kilkanaście godzin zasilane wyłącznie z akumulatora. Świadczy to również o dużym postępie w dziedzinie źródeł zasilania. |
| Stopień scalenia | Pierwszy procesor firmy Intel potrzebował kilkudziesięciu układów towarzyszących. Obecnie nawet cały komputer można zbudować w jednym układzie scalonym. Dzięki miniaturyzacji i pozostałym cechom można np. budować coraz to wydajniejsze telefony komórkowe. |
| Architektura układu | Inżynierowie pracują także nad nowymi rozwiązaniami technicznymi. Poszczególne bloki procesorów i innych układów różnią się od poprzedników nie tylko wydajnością, lecz także budową i zastosowanymi rozwiązaniami technicznymi. Dzięki temu można na przykład budować układy, które przy tej samej częstotliwości taktowania co ich poprzednicy pracują znacznie wydajniej. |
| Pojemność pamięci RAM | Współczesne systemy operacyjne, zarówno w PC, jak i telefonach, wymagają coraz to większej pojemności pamięci operacyjnej. Pierwszym domowym komputerom starczało 48 kB. Obecnie standardem jest kilka lub kilkanaście GB. Pozwala to także na instalowanie nowych wersji programów i aplikacji, które przy takiej ilości pamięci pracują płynnie i pewnie. |

## 2.2. Zaczęło się od rozrywki, czyli do czego wykorzystujemy komputery osobiste

W filmie *Piraci z Krzemowej Doliny* umieszczono scenę, w której młody Steve Wozniak, bliski przyjaciel i wspólnik Steve'a Jobsa, przedstawił prototyp komputera Apple specjaliście z firmy Hewlett-Packard (obecnie HP). Ten po chwili zastanowienia zadał pytanie: „A do czego zwykłemu człowiekowi komputer?". Jobs miał podpisany kontrakt i wszystko, co wymyślił, miało należeć do firmy. Jak wiemy, HP nie widziało zastosowania dla komputera osobistego. Ale pytanie zostaje otwarte. Nikt wtedy nie był w stanie na nie odpowiedzieć. Obecnie możemy jedynie powiedzieć, czym jest komputer, ale nikt nie zdoła określić, do czego będzie wykorzystywany w przyszłości.

**Czy umiesz wymienić choćby kilka zastosowań współczesnych komputerów w kontekście rozwoju naszej cywilizacji?** Można dzielić je na wiele sposobów. Z jednej strony postęp informatyki znacznie przyspiesza rozwój wszelkich dziedzin nauki, pomaga w życiu codziennym itp., a z drugiej pozwala na wykorzystywanie technologii do przestępstw czy budowania coraz to groźniejszej broni.

Komputery przyczyniły się ponadto do zmian cywilizacyjnych i rozwoju społeczeństw. W historii świata nie było chyba nigdy wcześniej wynalazku, który nie tylko zmienił sposoby produkcji czy komunikacji, lecz także znacząco wpłynął na zmiany zachowań społecznych.

Obecnie każdy posiadacz telefonu komórkowego może mieć dostęp do sieci przez całą dobę. Jeśli do tego korzysta z komunikatorów i portali społecznościowych, to może nie tylko utrzymywać ciągły kontakt ze znajomymi, lecz także na bieżąco śledzić aktualności oraz wypowiedzi i opinie artystów, polityków itp. Z jednej strony pozwala to na większą aktywność i świadomość obywatelską, a z drugiej – uzależnia od informacji. Bez względu na wiek i zaawansowanie w posługiwaniu się środkami IT (technologia informacyjna, ang. information technology, czytaj: informejszyn teknolodży) ludzie uzależniają się od ciągłego uczestniczenia w życiu wirtualnym. Nie ma chyba użytkownika sieci, który kilka razy dziennie nie czytałby bieżących informacji, wypowiedzi na portalach społecznościowych czy korespondencji.

Powszechny dostęp do internetu zmienił także oblicze polityki. Dawniej, jeśli polityk chciał przekazać coś swoim wyborcom, musiał zgłosić się do mediów. Obecnie większość polityków używa na przykład Twittera. Publikowane tam wypowiedzi docierają do wielu milionów ludzi w tym samym czasie. Polityk obecnie może pominąć oficjalne media, chociażby stacje TV, i nie musi już zabiegać o ich przychylność. Rola mediów ogranicza się więc do komentowania wypowiedzi. Zanim jednak to się stanie, swoje opinie zdążą już wypowiedzieć użytkownicy danej usługi. Wszystko dzięki temu, że większość użytkowników telefonów jest on-line przez całą dobę.

Oczywiście od sieci nie da się uciec. Współczesny świat rozwija się głównie dzięki możliwości szybkiej i nieograniczonej wymiany informacji. Problem w tym, by robić to świadomie i rozsądnie.

| | |
|---|---|
| Telefon – dawniej stacjonarny, a dziś mobilny globalnie. | Smartfon – gotowość do komunikacji on-line na wiele sposobów przez całą dobę. Nie trzeba już łącza i komputera PC. |
| Wymiana dokumentów natychmiastowa bez względu na miejsce nadania i odbioru. | Szybkie usługi i zakupy przez sieć. W tym także zakup biletów komunikacji, na koncerty itp. |
| Komunikatory wykorzystujące transmisje strumieniowe dźwięku i obrazu. Komunikacja bezpośrednia bez ograniczeń. | Bankowość i płatność on-line, w tym także za pomocą telefonu. |
| E-mail, bez ograniczeń czasowych. | Informacje i wiadomości dostępne natychmiast po opublikowaniu. W tym także fałszywe, które powinniśmy umieć ocenić. |
| Wymiana nieograniczonej ilości danych. | E-urzędy. Obywatel może kontaktować się, składać i otrzymywać dokumenty itp. z instytucjami państwa, administracji, samorządu lokalnego i innych instytucji za pośrednictwem sieci. |
| Profesjonalne i amatorskie transmisje stumieniowe na żywo. Dawniej dostępne jedynie dla stacji telewizyjnych. | E-learning, czyli nauka na odległość za pomocą środków elektronicznych. Część zajęć na uczelniach i w szkołach odbywa się z wykorzystaniem platformy e-learningowej. Kursy doskonalące oraz inne szkolenia także są dostępne w sieci. |

e-urzędy i e-learning

ewolucja komputerów a zmiana zachowań społecznych

Rys. 2.3. Niektóre z zastosowań IT mających wpływ na zmiany cywilizacyjne

## ZADANIA DO WYKONANIA

1. Zapisuj lub archiwizuj w dowolny sposób czas używania swojego telefonu. Nie rozróżniaj przy tym wykonywanych czynności ani ich okoliczności. Załóż, że twój aktywny dzień ma 12 godzin. Oblicz, ile procent tego czasu patrzyłeś/patrzyłaś na ekran telefonu. To dużo czy mało? Oceń wynik doświadczenia i uzasadnij swoją ocenę.
2. Od 12 godzin odejmij czas spędzony na lekcjach w szkole (nie licząc przerw) i popraw obliczenia z zadania 1. Dokonaj ponownej analizy i oceń, który z przypadków daje więcej informacji na temat twojej aktywności w sieci. Wyciągnij wnioski. Znasz siebie i wiesz, na co wykorzystałeś/wykorzystałaś ten czas. Jaka część czasu spędzonego z telefonem miała niewielkie znaczenie dla twojego życia i rozwoju? Jaką część tego czasu mogłeś/mogłaś poświęcić na odpoczynek, relaks na świeżym powietrzu, naukę itp.? Co wartościowego można by w tym czasie zrobić?
3. Na podstawie wyników z zadania 2. określ procentowo ilość czasu spędzonego przed ekranem telefonu, która była niezbędna (np. poświęcona na rozmowy z bliskimi, robienie zdjęć, dzielenie się ważnymi informacjami itp.).
4. Wyświetl stronę internetową urzędu miasta, gminy, powiatu, sołectwa z terenu, na którym mieszkasz. Sprawdź, czy oferuje ona usługi e-urzędu. Jeśli tak, dowiedz się, do czego służą i jak ułatwiają życie mieszkańcom.
5. Za pomocą kalkulatora oblicz, ile różnych liczb można przesłać magistralą liczącą 128 bitów. Porównaj wynik z magistralą 64-bitową. Znajdź w sieci przykłady urządzeń komputerowych, które przetwarzają słowa 128-bitowe. Sprawdź, czy dostępne są 256-bitowe.

## PODSUMOWANIE LEKCJI

- **Rozwój komputerów** osobistych rozpoczął nowy okres w rozwoju cywilizacji. 8-krotne zwiększenie liczby bitów (z 8 do 64) przesyłanych lub przetwarzanych jednocześnie przez procesor znacznie zwiększyło liczbę jednocześnie przesyłanych lub przetwarzanych informacji – liczb (72 340 172 838 076 673-krotnie). Ma to wielki wpływ na **wydajność komputerów**. s. 18
- **E-learning** to nauczanie lub uczenie się na odległość za pośrednictwem środków elektroniczno-informatycznych. Platformy e-learningowe wykorzystuje się na wielu płaszczyznach – od wspomagania nauczania w szkołach, przez kursy doskonalące i nauczanie akademickie. Również sprawy urzędowe można załatwiać **za pośrednictwem usług elektronicznych**. s. 22
- Ewolucja komputerów osobistych zmieniła także zachowania społeczne. Każdy może publicznie wypowiadać swoje poglądy i ogłaszać informacje z pominięciem tradycyjnych mediów, np. **za pośrednictwem Twittera.** s. 22

# 3. Chrońmy informacje, czyli techniki uwierzytelniania, kryptografia i podpis elektroniczny

**NA TEJ LEKCJI:**

- dowiesz się, czym jest uwierzytelnienie i po co się je stosuje;
- poznasz, na czym polega kodowanie i szyfrowanie.

Coraz częściej, aby dostać się do jakichś zasobów sieci, musimy uwierzytelnić swoją tożsamość. Dzięki temu możemy korzystać z bezpiecznej poczty elektronicznej, bankowości internetowej czy e-urzędów.

## 3.1. Uwierzytelnianie

uwierzytelnianie ▸ Czym jest samo uwierzytelnianie? Według słownika języka polskiego ten termin znaczy „stwierdzenie autentyczności dokumentu lub podpisu". W informatyce od niedawna zamiast słowa „uwierzytelnienie" zaczęto również używać określeń „autentykacja" lub „autentyfikacja", które są niepoprawne.

Samo uwierzytelnienie oznacza potwierdzenie zadeklarowanej tożsamości. Ma na celu uzyskanie pewności, że jesteśmy tą osobą, instytucją lub firmą, za którą się podajemy.

Za każdym razem, kiedy korzystamy z zasobów chronionych, niezależnie, czy to jest sieć WWW, czy na przykład bank, musimy przejść trzy podstawowe kroki: identyfikację, uwierzytelnienie i autoryzację. Dopiero po pozytywnym ich pokonaniu dostaniemy się do zasobów chronionych.

### Proces uwierzytelniania

Pozytywne przejście uwierzytelnienia wymaga przejścia swego rodzaju procesu, który składa się z trzech kroków.

**Krok 1. Identyfikacja**

Podczas identyfikacji jest sprawdzana nasza tożsamość, na przykład:

- aby zalogować się na pocztę elektroniczną czy do konta w mediach społecznościowych, podajemy login, który następnie jest weryfikowany w bazie danych, by potwierdzić, czy użytkownik o takiej nazwie w ogóle istnieje;
- jeżeli chcemy uzyskać informacje dotyczące naszego konta na infolinii banku, zostaniemy poproszeni o podanie naszego imienia i nazwiska;

- podczas połączenia przeglądarki z serwerem SSL, na którym znajduje się szukana przez nas strona internetowa, serwer przedstawia certyfikat bezpieczeństwa: podaje swoją nazwę komputera (host) oraz element adresu strony internetowej (domenę).

**Krok 2. Uwierzytelnianie**

To ten moment, kiedy strona ufająca za pomocą dostępnych sobie narzędzi i metod weryfikuje, czy dany użytkownik (osoba) jest tym, za którego się podaje, na przykład:

- serwer, na którym mamy swoją skrzynkę pocztową, prosi nas o wpisanie hasła (wczytanie pliku z kluczem), które podaliśmy podczas rejestracji, i weryfikuje jego poprawność;
- obsługa banku zadaje kolejne pytania, które pomogą w weryfikacji naszej tożsamości, najczęściej prosi o podanie daty urodzenia, nazwiska panieńskiego matki, produktów, które mamy wybrane w danym banku;
- przeglądarka weryfikuje podpis cyfrowy złożony pod certyfikatem serwera, któremu ma zaufać.



Rys. 3.1. Uwierzytelnianie klienta przez bank

**Krok 3. Autoryzacja**

To ostatni element uwierzytelniania. W tym kroku otrzymamy prawo lub odmowę dostępu do danych. Może zdarzyć się jeszcze sytuacja, gdy nasza tożsamość czy autentyczność zostanie potwierdzona, ale nie będziemy mieli wszystkich uprawnień dostępu do danych.

### Techniki uwierzytelniania

techniki uwierzytelniania

W zależności od kanału komunikacyjnego stosuje się różne techniki uwierzytelniania:

- w przypadku dokumentów pisanych może być wymagany na przykład: podpis osoby uprawnionej, pieczęć instytucji, parafowanie dokumentu przez osobę uprawnioną, znak wodny, potwierdzenie notarialne na dokumencie, które zaświadcza, że dokument jest oryginalny;
- w przypadku osób czy istot żywych techniką uwierzytelnienia jest zabezpieczenie biometryczne (rozpoznanie tęczówki oka, linie papilarne, geometria dłoni, biometria układu naczyń krwionośnych dłoni), dokumenty tożsamości, karta elektroniczna, biochip, token;
- w przypadku dokumentów i wiadomości elektronicznych technikami uwierzytelnienia są podpis cyfrowy, kod uwierzytelnienia wiadomości;
- w przypadku komunikacji elektronicznej stosuje się techniki oparte na:
  - udowodnieniu posiadania hasła (kryptografia symetryczna) lub klucza prywatnego (kryptografia asymetryczna),
  - dowodzie z wiedzą zerową – procedura, w której jedna ze stron potrafi udowodnić drugiej, że dysponuje pewną informacją bez jej ujawniania,
  - haśle jednorazowym.

### Uwierzytelnienia dwuskładnikowe

uwierzytelnienie dwuskładnikowe

Uwierzytelnienia dwuskładnikowe jako technikę autoryzacji stosuje się przy dostępie do danych lub systemów o szczególnej wartości. W takich sytuacjach w celu uwierzytelnienia stosuje się łącznie dwie metody z opisanych powyżej. Przykładem takiego działania jest dwuetapowa autoryzacja podczas logowania się do poczty elektronicznej. Oprócz podania nazwy użytkownika i hasła może być wymagane również wpisanie jednorazowego kodu otrzymanego SMS-em lub potwierdzenie naszej tożsamości na urządzeniu, które podamy podczas ustawiania dwuskładnikowych zabezpieczeń.

## 3.2. Kryptografia

kodowanie i szyfrowanie

Jednym ze sposobów zabezpieczania wiadomości, znanym od zarania dziejów, było ich kodowanie w taki sposób, aby tylko nadawca i odbiorca poznali ich treści. Terminem kodowania określa się również szyfrowanie, chociaż zastosowania techniczne wskazują, że kodowanie i szyfrowanie to dwie różne metody. W kodowaniu używa się słów kodowych, czyli wyrażeń bądź pojedynczych słów zastępujących właściwą treść wiadomości. Standardowo znajdują się one w tzw. książce kodowania. Szyfrowanie natomiast odbywa się na poziomie pojedynczych znaków lub ich grup. W nowoczesnych szyfrach, stosowanych na przykład do kodowania poczty elektronicznej, odbywa to się nawet na poziomie pojedynczych bitów.

### Kody jedno- i dwuczęściowe

Początkowo książki kodowe tworzono z uszeregowanym tekstem jawnym (pierwotną, niezakodowaną wiadomością). W myśl takiego uporządkowania tekstu na przykład w kodzie numerycznym wyrazom zaczynającym się na literę „a" były przyporządkowane liczby mniejsze niż te, które przyporządkowano wyrazom zaczynającym się na „z". Niestety taki sposób kodowania niósł za sobą duże ryzyko szybkiego odkodowania i przewidywalność ułatwiającą złamanie kodu. W kodowaniu jednoczęściowym stosowano tę samą księgę kodów zarówno do zakodowania wiadomości, jak i do jej odkodowania.

Aby utrudnić złamanie kodu i pozbyć się błędów podczas kodowania, zaczęto stosować kodowanie dwuskładnikowe. Do zakodowania i rozkodowania wiadomości potrzeba dwóch książek kodowych. Taki proces można porównać do tłumaczenia tekstu z języka obcego, do czego należy użyć dwóch słowników, na przykład polsko-niemieckiego i niemiecko-polskiego.

Oczywiście tego typu kodowanie wymagało więcej pracy od osób zajmujących się zarówno samym tworzeniem wiadomości, jak i jej odkodowywaniem. Trzeba było zachować większą ostrożność przy przekazywaniu wiadomości oraz ksiąg kodujących i dekodujących. System ten jednak zapewniał większe bezpieczeństwo niż kodowanie jednoczęściowe i sprawiał trudności przy jego łamaniu.

### Kody jednorazowe

Kody jednorazowe to ustalone słowa, wyrażenia lub symbole stosowane raz w ogólnodostępnym przekazie. Standardowo są one tworzone w taki sposób, aby tylko adresat wiadomości mógł się zorientować, że użycie konkretnego wyrazu lub zdania oznacza konkretne polecenie lub przekaz o wykonanym zadaniu. Oczywiście muszą one być w taki sposób skonstruowane, aby nie budziły podejrzenia, na przykład „Agata zaczęła rodzić" może informować o podłożonym ładunku wybuchowym i uruchomieniu czasowego detonatora.

### Superszyfrowanie

Dzisiaj stosuje się coraz bardziej złożone metody kryptografii. Standardowo oprócz kodowania wiadomości w celach bezpieczeństwa jeszcze się ją szyfruje. W przypadku kodów numerycznych do poszczególnych grup liczb dodaje się szereg cyfr, które stanowią kod klucza. Po co oprócz szyfrowania wiadomości używa się kodu? Poza samym bezpieczeństwem przekazu takiej wiadomości kod kompresuje (zmienia sposób zapisu przez zmniejszenie nadmiaru informacji) wiadomość i koryguje błędy.

## 3.3. Podpis elektroniczny

Podpis elektroniczny to jedna z metod uwierzytelnienia dokumentów elektronicznych zapewniająca im autentyczność i skutki prawne.

◂ podpis elektroniczny

## Dyrektywa o Wspólnej Infrastrukturze Podpisów Elektronicznych

Pojęcie podpisu elektronicznego w krajach Unii Europejskiej wprowadzono w 1999 roku Dyrektywą o Wspólnej Infrastrukturze Podpisów Elektronicznych (1999/93/WE). Wprowadzenie tych przepisów i zastosowanie podpisu elektronicznego w dużej mierze ułatwia funkcjonowanie w dzisiejszym świecie. Przykładowo: możemy bez wychodzenia z domu rozliczyć podatek dochodowy czy złożyć wniosek o wydanie nowego dowodu osobistego.

W dyrektywie wyróżniono trzy rodzaje podpisu elektronicznego:

- podpis elektroniczny – deklaracja tożsamości autora, złożona w postaci elektronicznej pod dokumentem;
- zaawansowany podpis elektroniczny – podpis, który dzięki zastosowaniu odpowiednich środków technicznych jest jednoznacznie związany z dokumentem oraz autorem i trudny do sfałszowania;
- kwalifikowany podpis elektroniczny – forma zaawansowanego podpisu elektronicznego spełniająca szczególne wymagania techniczne i organizacyjne – osoby korzystające z tej formy podpisu elektronicznego posiadają certyfikat kwalifikowany wystawiony przez kwalifikowany urząd certyfikacji oraz klucz prywatny przechowywany na karcie kryptograficznej.

## Podpis cyfrowy

Podpis cyfrowy jest matematycznym narzędziem wykorzystywanym w podpisie elektronicznym umożliwiającym identyfikację tożsamości podmiotów przesyłających dokumenty elektroniczne. Służy on zapewnieniu funkcji bezpieczeństwa, takich jak: autentyczność pochodzenia dokumentu, niezaprzeczalność, która utrudnia wyparcie się autorstwa lub znajomości treści dokumentu, oraz integralność pozwalającą na wykrycie nieautoryzowanych modyfikacji dokumentu.

Aby wymienione wyżej funkcje bezpieczeństwa zadziałały, należy zastosować następujące środki:

- instrumenty techniczne – algorytm, protokoły i formaty kryptograficzne, które będą realizowały funkcje bezpieczeństwa;
- instrumenty prawne – przepisy potwierdzające zgodność z prawem wymienionych instrumentów technicznych;
- instrumenty organizacyjne – jak urzędy certyfikacyjne, których zadaniem jest poświadczenie związku klucza publicznego z konkretną osobą.

Stosowanie podpisu cyfrowego wiąże się z zastosowaniem kryptografii asymetrycznej, w której klucz prywatny służy do składania podpisu, a klucz publiczny – do jego weryfikacji.

Często w życiu codziennym posługujemy się zamiennie pojęciami podpisu cyfrowego i podpisu elektronicznego. Jednak w prawie unijnym zostały one rozróżnione. Podpis cyfrowy zgodnie z normą ISO 7498-2: 1998 jest traktowany jako „zbiór danych dołączonych do innych danych lub ich przekształcenie kryptograficzne, które pozwala odbiorcy danych udowodnić pochodzenie danych i zabezpieczyć je przed fałszerstwem". Natomiast podpis elektroniczny zgodnie z Dyrektywą 1999/93/WE jest operacją podpisywania konkretnych danych (dokumentu elektronicznego) przez osobę fizyczną.

## ZADANIA DO WYKONANIA

1. Wyszukaj informacje na temat działania maszyny szyfrującej Enigma, której działanie rozpracowali trzej polscy naukowcy: Marian Rejewski, Henryk Zygalski i Jerzy Różycki.
2. Dowiedz się, do czego wykorzystywano superszyfrowanie podczas II wojny światowej. Opisz w zeszycie jeden z przykładów.

## PODSUMOWANIE LEKCJI

- **Uwierzytelnianie** to stwierdzenie autentyczności, które sprawdza się poprzez identyfikację, uwierzytelnienie i autoryzację. s. 24
- **Techniki uwierzytelnienia** różnią się w zależności od tego, co uwierzytelniamy – może to być na przykład podpis, zabezpieczenie biometryczne czy podpis elektroniczny. s. 26
- **Uwierzytelnienie dwuskładnikowe** stosuje się przy dostępie do danych o szczególnej ważności. Polega na łączeniu dwóch różnych technik uwierzytelniania. s. 26
- **Kodowanie i szyfrowanie** to dwie odmienne techniki kryptografii. Samo kodowanie lub szyfrowanie wykorzystuje się w kodach jednoczęściowych. Ze względów bezpieczeństwa warto stosować kody dwuczęściowe. s. 26
- **Podpis elektroniczny** jest zestawem procedur prawnych posługujących się technologią podpisu cyfrowego. Ma on ułatwić wymianę dokumentów elektronicznych. s. 27

# 4. Praktycznie i teoretycznie, czyli trendy historyczne w rozwoju technologii

**NA TEJ LEKCJI:**

- dowiesz się, jak rozwój informatyki wpłynął na życie zawodowe człowieka;
- zrozumiesz, że rozwój technologii informacyjnej przyczynił się do powstania nowych usług.

To ciekawość świata towarzysząca nam od początku istnienia rodzaju ludzkiego, a zarazem chęć podporządkowania sobie przyrody i przetrwania na Ziemi, spowodowała, że człowiek zaczął wytwarzać rzeczy i narzędzia, które miały mu w tym pomóc. W historii ludzkości dokonano kilku spektakularnych wynalazków, jak opanowanie ognia, wynalezienie koła, maszyny parowej, telefonu czy też komputera – dzięki nim odmieniło się życie ludzi.

## 4.1. Wpływ technologii komputerowej na najważniejsze obszary życia osobistego

Od dnia, kiedy uruchomiono pierwszy komputer i uznano go za rzecz niezbędną w życiu człowieka, minęły dziesiątki lat. Obecnie komputery towarzyszą nam w każdej dziedzinie życia. Używa się ich w nauce, handlu, medycynie, komunikacji, przemyśle. Trudno jest nam sobie dzisiaj wyobrazić życie bez tych maszyn.

Nowe technologie, których ogromny rozwój trwa od drugiej połowy XX wieku, znacznie zmieniły życie codzienne człowieka. Wiele czynności, takich jak komunikacja interpersonalna, zainteresowania, czynności zawodowe, prowadzenie działalności gospodarczej, stało się o wiele łatwiejszych.

Rozwój technologii poprawił jakość przemieszczania się – jeszcze 100 lat temu dostępnymi środkami komunikacji, służącymi do przemieszczania się na duże odległości, były kolej parowa czy statki, a przemysł motoryzacyjny dopiero się rozwijał. Obecnie podróżujemy do dowolnego miejsca na Ziemi samolotami, poruszamy się z prędkością kilkuset kilometrów na godzinę nowoczesnymi pociągami, a niektóre samochody mają napęd hybrydowy lub elektryczny.

Do komunikacji interpersonalnej wykorzystujemy przede wszystkim komunikatory głosowe, wideo i tekstowe. Na początku XX wieku najczęściej informacje przekazywano listownie lub osobiście. Dzisiaj mamy dostęp do telefonii komórkowej, a funkcja telefonu w smartfonach jest jedną z wielu, gdyż te urządzenia to właściwie małe przenośne komputery. Na początku ubiegłego wieku dostęp do telefonu był bardzo ograniczony, a rozmowy – bardzo drogie.



Rys. 4.1. Koleje dużych prędkości

Przekazywanie wiadomości odbywało się przede wszystkim w formie pisanej i drukowanej oraz radiowej. Dzisiaj internet i telewizja gwarantują nam stały dostęp do informacji. Dzięki rozwojowi technologii komputerowej możemy na bieżąco komentować w mediach społecznościowych każdą wiadomość.

Nabywanie produktów w dużej mierze odbywa się przez internet, bez wychodzenia z domu. Sto lat temu człowiek, aby cokolwiek kupić, musiał udać się do sklepu, a dostęp do informacji o produktach był bardzo ograniczony. Obecnie poprzez internet zapoznajemy się z opiniami producentów oraz nabywców towarów i usług, co pomaga podjąć decyzję o zakupie wybranych produktów.

Finansami zarządzamy dzisiaj dzięki bankowości internetowej. Coraz częściej korzystamy z transakcji bezgotówkowych, za pomocą kart płatniczych, płatności internetowych czy poprzez aplikacje na smartfonach i smartwatchach. Wcześniej wykorzystywaliśmy do takich rzeczy tylko gotówkę lub czeki.

Obecnie internet objął właściwie wszystkie sfery życia człowieka. Jeszcze niedawno, chcąc poznać informacje o jakimś produkcie, korzystaliśmy z komputera stacjonarnego, a potrzebne dane na temat produktu drukowaliśmy, bo dostęp do internetu był bardzo drogi. Obecnie dostęp do interesujących treści mamy na bieżąco w każdym miejscu.

Zwróć uwagę na zagrożenia płynące z tak szybkiego rozwoju nowoczesnych technologii i wykorzystywania ich w życiu codziennym:

- Bezpieczeństwo danych – stały dostęp do internetu, korzystanie z ogólnodostępnych sieci Wi-Fi powodują, że jesteśmy o wiele bardziej narażeni na ataki bądź kradzież w sieci. Przejmowanie kont e-mail, okradanie kont w bankowości internetowej bądź kradzież tożsamości (podszywanie się pod kogoś innego) są częste.
- Przyzwyczajenie do ułatwień skutkuje obniżoną aktywnością fizyczną i interpersonalną. Przedkładamy znajomości wirtualne oraz komunikację elektroniczną nad znajomości w życiu realnym i kontakt osobisty. Towarzyszyć temu mogą ograniczenia intelektualne, na przykład trudności w bezbłędnym pisaniu, gdyż w komunikacji wirtualnej pomagają nam w poprawianiu błędów słowniki, a także obniżenie zdolności obliczeń matematycznych, które wykonujemy z pomocą dostępnych w sieci kalkulatorów. Nadmiernie ufamy urządzeniom korzystającym z nadajników GPS, przez co tracimy umiejętność poruszania się w terenie bez urządzeń nawigacyjnych.

Rozwój technologii komputerowej może doprowadzić do tego, że zatracimy zdolność normalnego funkcjonowania bez urządzeń i technologii mobilnej.

## 4.2. Rozwój i wpływ informatyki na życie zawodowe człowieka

rozwój komputeryzacji a życie zawodowe człowieka

Rewolucja przemysłowa w XVIII wieku i jej druga odsłona w wieku XIX wpłynęły na zmianę ekonomii rolniczo-rzemieślniczej w gospodarkę opartą na masowej produkcji dóbr. Obecne przemiany przemysłowe i technologiczne wymagają przede wszystkim pracowników umysłowych, a znacznie mniej pracowników fizycznych, przy czym jeszcze więcej produkuje się dóbr osobistych niż wcześniej. Żyjemy w czasach charakteryzujących się bardzo dużym tempem przemian i postępu technologicznego. Rozwój informatyki powoduje zmiany w strukturach zatrudnienia, coraz częściej maszyny zaczynają przejmować obowiązki pracowników. Pewne umiejętności stają się zbędne na rynku pracy, ale pojawiają się nowe dziedziny przemysłu, które jeszcze kilka lat temu nie istniały. Współcześnie pracujący muszą ciągle się doszkalać i przekwalifikowywać. Groźba „technologicznego bezrobocia" staje się impulsem do dalszego osobistego rozwoju.

Ciągły rozwój technologii powoduje zmiany na rynku pracy. Popyt na niewykwalifikowanych pracowników maleje, a zwiększa się zapotrzebowanie na specjalistów. Najlepszym przykładem tego jest branża informatyczna. Nieustanny rozwój sprzętu i technologii informatycznej, coraz większa ilość gromadzonych i przetwarzanych danych sprawiają, że z roku na rok rośnie zapotrzebowanie na specjalistów z branży IT.



Rys. 4.2. Automatyzacja pracy

Od kilku dekad diametralnie zmienił się również sektor branży finansowej. Rozwój komputeryzacji tej branży spowodował zapotrzebowanie na nowych specjalistów – doświadczonych menadżerów finansowych, analityków, specjalistów od inwestycji.

Komputeryzacja również przyczyniła się do innowacyjnego rozwoju branży chemicznej, biotechnologicznej czy medycznej. Nowych specjalistów poszukują koncerny farmaceutyczne, instytuty badawcze i różnego rodzaju firmy produkcyjne.

W przytoczonych przykładach umiejętności posługiwania się wiedzą z technologii informatycznej czy wykorzystywaniem urządzeń komputerowych są niezbędne.

W dzisiejszych czasach łatwiej znajdą pracę osoby, które mają przygotowanie interdyscyplinarne, potrafiące łączyć zmysł artystyczny z nowoczesną technologią.

## 4.3. E-usługi

Postęp w dziedzinie technologii informacyjnej i sprzętu komputerowego dał możliwości rozwoju całkiem nowym gałęziom usług, o których jeszcze pod koniec XX wieku tylko się mówiło. Należą do nich e-usługi. Obejmują wszelkiego rodzaju działalność usługową prowadzoną drogą elektroniczną poprzez

e-usługi

sieć telekomunikacyjną lub internet, w pełni zautomatyzowaną, z minimalnym udziałem człowieka. Do e-usług według unijnej dyrektywy, zalicza się:

- produkty w formie cyfrowej łącznie z oprogramowaniem, jego modyfikacjami i nowszymi wersjami oprogramowania;
- usługi umożliwiające lub wspomagające obecność przedsiębiorstw lub osób w sieci elektronicznej, takie jak witryna lub strona internetowa;
- usługi generowane automatycznie przez komputer i przesyłane przez internet lub sieć elektroniczną w odpowiedzi na określone dane wprowadzane przez usługobiorcę;
- rynek on-line (np. aukcje internetowe);
- pakiety usług internetowych oferujących dostęp do informacji, np. strony umożliwiające dostęp do: aktualnych wiadomości, informacji meteorologicznych, turystycznych, gier, hosting witryn internetowych, do grup dyskusyjnych.

Najbardziej popularnymi e-usługami, z których obecnie korzystamy, są:

- strony WWW,
- e-maile,
- e-rozmowy (komunikatory, czaty itp.),
- e-nauka (kursy językowe, zawodowe, profesjonalne kursy dla pracowników firm, studia licencjackie, magisterskie i uzupełniające, korepetycje bez udziału nauczyciela),
- e-booki,
- e-handel (e-commerce; wym. i komers)
- e-administracja,
- e-bankowość,
- e-ubezpieczenia,
- e-kultura,
- e-turystyka.

Wszystkie wyżej przedstawione przykłady to tylko część tego, co można zakwalifikować do e-usług. Ta część gospodarki obecnie najszybciej się rozwija.

## 4.4. Rozwój komunikacji społecznej

Powstanie i rozwój cywilizacji nie byłyby możliwe bez opanowania technik komunikowania się. Bez umiejętności porozumiewania się człowiek nie byłby w stanie zainicjować postępu technicznego oraz tworzyć społeczności.

Czym w ogóle jest komunikacja? Według słownika języka polskiego komunikacja to przekazywanie i odbieranie informacji w bezpośrednim kontakcie z drugą osobą bądź przepływ informacji między urządzeniami, na przykład telefonami lub komputerami.

Oczywiście u zarania dziejów ludzkości porozumiewano się w bezpośrednim kontakcie z drugą osobą. Wraz z upływem czasu możliwości sposobów komunikacji wzrastały i stawały się coraz bardziej skomplikowane. Rozwój metod komunikowania się spowodował, że ludzie zaczynają komunikować się ze sobą szybciej i na większe odległości.

Dzisiaj używamy technik komunikowania się, o których nasi przodkowie nie śnili. Pokolenia, które przyjdą po nas, będą się komunikowały za pomocą technologii dzisiaj występujących w literaturze *science fiction*. Nowe technologie komunikowania sprawiają, że społeczeństwa szybciej i sprawniej wymieniają się informacjami. Dzisiaj jesteśmy w stanie brać udział w wydarzeniach dziejących się po drugiej stronie globu, na przykład dzięki transmisji z otwarcia igrzysk olimpijskich.



Rys. 4.3. Komunikacja interpersonalna

Urządzenia współcześnie używane do komunikacji są bardzo zaawansowane. Jeszcze na początku XXI wieku mało kto wyobrażał sobie komunikatory wideo pozwalające na rozmowę w czasie rzeczywistym. Urządzenia takie jak palmtop stanowiły zapowiedź tego, czym są dzisiejsze smartfony.

▸ komputery osobiste




Rys. 4.4. Socialmedia

Człowiek wykorzystując dzisiejszą technologię, w dowolnym miejscu i o dowolnej porze przesyła wiadomości tekstowe, dźwięk czy obraz statyczny lub wideo. W tym samym czasie cała ludzkość może oglądać premierowy odcinek nowej serii ulubionego serialu. Jeszcze 10 lat temu wydawało się to nieprawdopodobne.

Dzisiaj w internecie znajduje się większość zasobów, do których wygasły prawa autorskie. Do reszty informacji mamy dostęp po wykupieniu licencji, abonamentu lub jednorazowego dostępu.

nowoczesne technologie

XIX i XX stulecie to czas komunikacji interpersonalnej – rozwój telegrafu, telefonu, radia czy telewizji były oddzielone od siebie zaledwie dekadami. W wieku XXI nowe technologie komunikacyjne pojawiają się co kilka lat. E-mail, komunikatory tekstowe czy wideo zdobywają coraz większą popularność. Nowe wersje Skypa czy Messengera pojawiają się co kilkadziesiąt dni. Stale są rozbudowane o nowe możliwości.

Naukowcy prognozują, że postęp technologiczny, a przede wszystkim możliwości komunikowania się, zdecydowanie odmienią oblicze naszej cywilizacji.

## ZADANIA DO WYKONANIA

1. Wyszukaj informacje na temat usługi e-PUAP. Omów je i uzasadnij, które z nich – twoim zdaniem – będą najczęściej wykorzystywane przez użytkowników.
2. Podaj, jakie zagrożenia niesie za sobą tak szybki rozwój technologii informacyjnej związanej z komunikacją interpersonalną.
3. Znajdź informacje na temat komunikatora Gadu-Gadu. Wskaż przyczyny braku zainteresowania tą usługą.

## PODSUMOWANIE LEKCJI

- **Rozwój komputeryzacji** przemysłu powoduje, że coraz częściej i szybciej zmienia się struktura zawodowa społeczeństwa. Najbardziej poszukiwanymi pracownikami są osoby z umiejętnościami i wykształceniem technicznym, które są w stanie współpracować z wykorzystaniem technologii komputerowej. s. 32
- **E-usługi** są najszybciej rozwijającą się obecnie dziedziną gospodarki. Pozwalają za pomocą sieci telekomunikacyjnej lub internetu rozwijać działalność usługową. s. 33
- **Komputery osobiste** są powszechne na co dzień i mają ogromny wpływ na nasze życie osobiste i zawodowe. s. 35
- Rozwój możliwości komunikowania się jeszcze nigdy nie był tak szybki. Nowoczesne technologie umożliwiające przesyłanie informacji w czasie rzeczywistym zrewolucjonizowały naszą cywilizację. s. 36

# 5. Kim jestem, czyli jak bezpiecznie budować wizerunek w sieci

**NA TEJ LEKCJI:**

- dowiesz się, kto i co chroni twoje dane;
- dowiesz się, jak dbać o swój wizerunek w sieci.

Internet jest źródłem olbrzymiej ilości informacji. Dostęp do nich jest praktycznie nieograniczony. Czy jednak masowy dostęp do wszelkich informacji nie zagraża twojej prywatności? Czy zdjęcia z dyskoteki, na których widać, jak się bawisz, umieszczone na portalach społecznościowych bez twojej wiedzy są takim naruszeniem? Czy zastanawiasz się przed umieszczeniem w internecie różnych fotografii, filmów i tekstów, że mogą one komuś sprawić przykrość, spowodować różne kłopoty? Gdzie leży granica prywatności? Czy można dysponować czyimś wizerunkiem bez jego wiedzy? Czy instytucje mogą żądać od ciebie danych osobowych? Jakie skutki może mieć fakt, że informacjami o tobie dysponują firmy, banki, urzędy i osoby prywatne?

## 5.1. Kto i co chroni twoje dane, czyli co mówi prawo

Problem prywatności i bezpieczeństwa danych osobowych istnieje tak długo, jak istnieją środki przekazu informacji. Dlatego 19 grudnia 1966 roku ONZ podała do ratyfikacji Międzynarodowy Pakt Praw Obywatelskich i Politycznych. Polska ratyfikowała go w 1977 roku. Jego artykuł 17 brzmi:

1. **Nikt nie może być narażony na samowolną lub bezprawną ingerencję w jego życie prywatne, rodzinne, dom czy korespondencję ani też na bezprawne zamachy na jego cześć i imię.**
2. **Każdy ma prawo do ochrony prawnej przed tego rodzaju ingerencjami i zamachami.**

naruszenie prywatności w sieci

Masz więc prawo do ochrony swojej prywatności i domagania się usunięcia informacji ją naruszających. Do niedawna proces dochodzenia swoich praw był długotrwały i mało skuteczny z powodu braku odpowiednich przepisów prawa. Parlament Europejski opracował rozporządzenie w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (2016/679 z 27 kwietnia 2016 r.). Rozporządzenie to wprowadzono 25 maja 2018 roku we wszystkich krajach członkowskich UE. Obywatele Unii są więc chronieni tym prawem na jej terenie. Rozporządzenie w skrócie określa się jako RODO.

Czym właściwie są dane osobowe? Domyślasz się, że są nimi imię i nazwisko, PESEL, prywatny lub firmowy numer telefonu, adres poczty elektronicznej, także dane identyfikujące twój komputer, na przykład adres IP lub MAC (unikatowy adres karty sieciowej) oraz inne identyfikatory elektroniczne.

Bez wiedzy osoby, której dotyczą dane, nie wolno ich przetwarzać ani automatycznie, ani przez działalność człowieka. A czym jest przetwarzanie? To każda operacja na tych danych, przykładowo: zbieranie, udostępnianie, utrwalanie, porządkowanie, adaptowanie, modyfikacja, przeglądanie, łączenie, usuwanie, niszczenie, używanie, między innymi do wysyłki korespondencji czy towarów.

**Uwaga!**

Każdy, kto nie stosuje przepisów RODO, podlega wysokiej karze. Każdy obywatel Unii Europejskiej ma prawo do przeglądania danych, które zostały do tej pory zgromadzone, np. w sklepie wysyłkowym. Jeśli uzna, że nie są one już potrzebne danemu podmiotowi lub osobie, może kategorycznie żądać ich usunięcia, a operator danych musi je usunąć.

Państwo obszernie informuje o zasadach i przepisach RODO pod adresem: *https://www.gov.pl/cyfryzacja/rodo-informator*. Znajdziesz tam szczegółowe informacje, a także ulotki i filmy przybliżające obywatelowi (rys. 5.1.) i przedsiębiorcy jego prawa i obowiązki w związku z wprowadzeniem RODO. Wczytaj tę stronę i zapoznaj się z tymi materiałami. To jedno z najważniejszych dla każdego obywatela praw. Twoim obowiązkiem jest zapoznanie się z nim, by uniknąć kłopotów w związku z obrotem danymi osobowymi.




Rys. 5.1. Ulotka informacyjna Ministerstwa Cyfryzacji dotycząca wprowadzenia RODO

RODO a usuwanie informacji z bazy danych

## 5.2. Dbaj o swój wizerunek, czyli nic ponad konieczność

dbanie o wizerunek w sieci

Przypadki umieszczania w sieci fotografii lub filmów bez zgody osób na nich uwiecznionych są częste. Osoby publikujące bezprawnie takie treści, wśród których znaczną grupę stanowią ludzie młodzi, robią to dla żartu lub z chęci ośmieszenia kogoś. Nie zdają sobie sprawy z krzywdy, jaką mogą wyrządzić danej osobie i jej bliskim. Zdarza się, że ofiara takiego działania wykazała się zbyt wielkim zaufaniem, a nawet naiwnością, w stosunku do sprawcy, na przykład sama przesłała mu swoje zdjęcia. Upublicznienie materiałów nierzadko ma tragiczne skutki – może prowadzić do depresji, a nawet samobójstw.

Udostępnianie zdjęć lub nagrań bez zgody osób na nich uwiecznionych nie tylko jest niezgodne z prawem, RODO i paktem ONZ, lecz także jest sprzeczne z normami moralnymi. To przestępstwo, za które sprawca może odpowiadać przed sądem.

Jednak każdy człowiek powinien zdawać sobie sprawę, że zdjęcia i filmy, na których został uwieczniony, mogą zostać opublikowane. Pamiętaj o tym i dbaj o swój wizerunek. Nie pozwalaj się fotografować nieznanym osobom. Jeśli już to nastąpi, wyraźnie zastrzeż swoje prawa lub zażądaj skasowania materiału.

Twoje prawa do zdjęć i filmów są także chronione przez ustawę o prawie autorskim i prawach pokrewnych. Artykuł 81 mówi:

> 1. Rozpowszechnianie wizerunku wymaga zezwolenia osoby na nim przedstawionej. W braku wyraźnego zastrzeżenia zezwolenie nie jest wymagane, jeżeli osoba ta otrzymała umówioną zapłatę za pozowanie.

Jednocześnie ta sama ustawa stwierdza, że:

> **Art. 83.** Do roszczeń w przypadku rozpowszechniania wizerunku osoby na nim przedstawionej oraz rozpowszechniania korespondencji bez wymaganego zezwolenia osoby, do której została skierowana, stosuje się odpowiednio przepis art. 78 ust. 1; roszczeń tych nie można dochodzić po upływie dwudziestu lat od śmierci tych osób.

Osoba, która umieściła zdjęcie lub film, nie jest ścigana przez prawo automatycznie, a jedynie po zgłoszeniu tego faktu (możliwości popełnienia przestępstwa) przez osobę poszkodowaną.

Jakie zdjęcia i filmy zawierające wizerunek obcej osoby można pokazywać bez konsekwencji? Określono to w artykule 81 powyższej ustawy:

2. Zezwolenia nie wymaga rozpowszechnianie wizerunku:
   1) osoby powszechnie znanej, jeżeli wizerunek wykonano w związku z pełnieniem przez nią funkcji publicznych, w szczególności politycznych, społecznych, zawodowych,
   2) osoby stanowiącej jedynie szczegół całości takiej jak zgromadzenie, krajobraz, publiczna impreza.

Artykuł 83 powyższej ustawy dotyczy również ochrony korespondencji elektronicznej. Nie wolno czytać lub rozpowszechniać korespondencji bez zgody autora lub adresata. Dotyczy to także przypadku, w którym nie następuje włamanie do skrzynki pocztowej. Mimo tak sformułowanego prawa należy dbać o bezpieczeństwo haseł do skrzynek i innych kont internetowych.

Większość ludzi korzystających z internetu nie podejrzewa, że może paść ofiarą przestępców. Często są oni łatwowierni i opowiadają nieznajomym o sobie lub upowszechniają w portalach społecznościowych informacje o tym, gdzie mieszkają, jaki mają samochód, ile zarabiają itp. Zdarza się, że takie dane ich rozmówca skrzętnie gromadzi. Jeśli na przykład umieścisz w portalu zdjęcie swojego nowego domu i podasz adres lub inne dane, które umożliwią jego ustalenie, oraz napiszesz, że właśnie wybierasz się na 2 tygodnie za granicę, to jest to doskonała wiadomość dla złodzieja. Pamiętaj, że nie tylko twoi znajomi mają dostęp do informacji, które opublikowałeś/opublikowałaś.

W portalach społecznościowych nie ma mechanizmu sprawdzania tożsamości. Każdy może w nich podszyć się pod ciebie. Takie zachowanie to kradzież tożsamości. Szkodzi ono zarówno tobie, jak i twojej rodzinie oraz znajomym.

internetowe formularze a dane osobiste

Wiele usług internetowych wymaga rejestracji. Jeśli wymaga to podania danych osobistych, na przykład twojego imienia i nazwiska z adresem i numerem telefonu, to poważnie zastanów się, czy koniecznie musisz korzystać z tych usług. Jeśli wymagany jest tylko mail, warto utworzyć skrzynkę pocztową na jednym z darmowych serwerów i używać jej do takich rejestracji. Zapisane podczas rejestracji dane trafiają do bazy. Często ignoruje się mały znacznik stawiany domyślnie przy formule, że zgadzasz się na przesyłanie między innymi informacji handlowych, a w przypadku użycia roboczej skrzynki pocztowej będzie to miało mniejsze znaczenie.

Zapoznaj się ze schematami dotyczącymi prywatności w sieci (rys. 5.2. i 5.3. na następnej stronie).




Rys. 5.2. Działania zwiększające bezpieczeństwo wizerunku w sieci




Rys. 5.3. Jak nie naruszać prawa do prywatności

## ZADANIA DO WYKONANIA

1. Przejrzyj dokładnie swoje konta na portalach społecznościowych i usuń z nich wszystkie informacje, które mogą być niewłaściwie wykorzystane przez innych użytkowników, np. datę urodzenia, adresy, nieudane zdjęcia, zdjęcia znajomych (jeśli nie masz zgody na publikację).
2. Przejrzyj zawartość portali internetowych, w szczególności społecznościowych, oraz strony internetowe, aby sprawdzić, czy ktoś narusza twoje prawa do wizerunku i prywatności. Skorzystaj przy tym z wyszukiwarek. Zareaguj odpowiednio, jeśli stwierdzisz nieprawidłowości.
3. Wykonaj prezentację, w której przedstawisz przykłady łamania prawa do prywatności. Dane zaczerpnij z portali informacyjnych. Omów, jak można było uniknąć tych nadużyć lub zminimalizować ich skutki.
4. Wyszukaj w sieci dokumenty prawa mówiące o ochronie praw człowieka do prywatności oraz komentarze prawników do nich. Znajdź także artykuły i opracowania na temat interpretacji tych praw. Przeczytaj je i wyciągnij wnioski dotyczące zasad zachowania się w sieci.
5. Odwiedź stronę *https://www.gov.pl/cyfryzacja/rodo-informator* i przeczytaj informacje na niej zawarte. Zapoznaj się także z dokumentami, które można pobrać z tej strony. Na ich podstawie przeanalizuj twoje nowe prawa i zrób listę portali i instytucji, z których powinieneś/powinnaś wycofać swoje dane.

## PODSUMOWANIE LEKCJI

- W przypadku odkrycia rozmyślnego, złośliwego lub drastycznego **naruszenia twojej prywatności** natychmiast reaguj – powiadom bliskie osoby i policję. Nie daj się szantażować. s. 38
- Na podstawie RODO **możesz żądać usunięcia swoich danych z każdej bazy danych**. Wyjątek stanowią te, które są niezbędne do realizacji wcześniej podpisanej umowy, np. na usługi bankowe czy teleinformatyczne. Jednak i te instytucje muszą ujawnić, jakie dane na twój temat zgromadziły. s. 39
- Dbaj o swój wizerunek poprzez **rozsądne umieszczanie materiałów** na własny temat oraz danych potrzebnych w uruchamianiu niektórych usług. Dzięki temu znacznie zwiększysz bezpieczeństwo twoje i twoich danych. s. 40
- **Nie wypełniaj internetowych formularzy danymi osobistymi,** jeśli nie jest to absolutnie konieczne. Nie udostępniaj swoich danych osobom oferującym pracę w internecie. Prawo do takich danych mają jedynie pracodawcy, z którymi podpisujesz umowę o pracę. Dotyczy to także pracy sezonowej, np. w wakacje. s. 41

## PODSUMOWANIE PRZED SPRAWDZIANEM

### 1. Bądź uczciwy, czyli przestrzeganie prawa w świecie informatyki

s. 13 • **Prawa majątkowe** to nie **prawa autorskie**. Naruszenie obu jest przestępstwem przeciw własności intelektualnej.

s. 13 • **Ochroną może być objęty wyłącznie** sposób wyrażenia; **nie są objęte ochroną** odkrycia, idee, procedury, metody i zasady działania oraz koncepcje matematyczne.

s. 13 • **Utwór jest przedmiotem prawa autorskiego** od chwili ustalenia, chociażby miał pozostać nieukończony.

s. 13 • **Ochrona przysługuje twórcy** niezależnie od spełnienia jakichkolwiek formalności.

s. 15 • **Licencja CC** (Creative Commons) pozwala na używanie i rozpowszechnianie utworu bez zezwolenia. Należy jednak podać autora utworu, np. w napisach końcowych filmu, w którym użyto muzyki udostępnionej na licencji CC.

### 2. 8, 16, 32, 64, czyli jak rozwój technologii wpływa na rozwój społeczeństw

s. 18 • **Binarny system liczbowy** jest podstawą konstrukcji i działania urządzeń cyfrowych, w tym także komputerów.

s. 25 • **Powszechny dostęp** do kanałów informacyjnych zmienił sposób postrzegania świata i życia społeczeństwa.

s. 25 • Dzięki rozwojowi systemów komputerowych **każdy ma możliwość głoszenia swoich poglądów.**

s. 25 • Poprzez **ogólnodostępną** telefonię komórkową i internet całkowicie zmieniły się sposoby komunikowania się i pracy, a także rozrywki społeczeństwa. Rozwój cywilizacji przyspieszył.

### 3. Chronimy informacje, czyli techniki uwierzytelniania, kryptografia i podpis elektroniczny

s. 24 • **Uwierzytelnianie** to stwierdzenie autentyczności, które jest sprawdzane poprzez identyfikację, uwierzytelnienie i autoryzację.

s. 26 • **Techniki uwierzytelnienia** różnią się w zależności od tego, co uwierzytelniamy – może to być np. podpis, zabezpieczenie biometryczne czy podpis elektroniczny.

s. 26 • **Uwierzytelnienie dwuskładnikowe** stosuje się w dostępie do danych o szczególnej ważności, wymaga użycia łącznie dwóch różnych technik uwierzytelniania.

s. 26 • **Kodowanie i szyfrowanie** to dwie odmienne techniki kryptografii. Samo kodowanie lub szyfrowanie stosuje się w kodach jednoczęściowych. Ze względów bezpieczeństwa warto stosować kody dwuczęściowe.

s. 27 • **Podpis elektroniczny** jest zestawem procedur prawnych wykorzystujących jako technologię podpis cyfrowy. Ułatwia wymianę dokumentów elektronicznych.

### 4. Praktycznie i teoretycznie, czyli trendy historyczne w rozwoju technologii

s. 32 • **Rozwój komputeryzacji** przemysłu powoduje, że coraz częściej i szybciej zmienia się struktura zawodowa. Na rynku pracy najbardziej poszukiwanymi pracownikami są osoby z umiejętnościami i wykształceniem technicznym, które są w stanie współpracować za pomocą technologii komputerowej.

s. 33 • **E-usługi** są najszybciej rozwijającą się obecnie dziedziną gospodarki. Pozwalają rozwinąć działalność usługową za pomocą sieci telekomunikacyjnej lub internetu.

s. 35 • **Komputery osobiste** towarzyszą nam już od co najmniej dwóch dekad. Całkowicie zdominowały nasze codzienne życie i mają ogromny wpływ na wszystkie strefy naszego życia osobistego od aspektów zawodowych po relacje międzyludzkie.

s. 36 • Rozwój możliwości komunikowania się jeszcze nigdy nie był tak szybki. **Nowoczesne technologie** umożliwiające przesyłanie informacji w czasie rzeczywistym zrewolucjonizowały naszą cywilizację.

### 5. Kim jestem, czyli jak bezpiecznie budować wizerunek w sieci

s. 38 • **Masz prawo do ochrony swojej prywatności** i domagania się usunięcia informacji ją naruszających.

s. 39 • **Dane osobowe obywateli Unii Europejskiej są chronione** prawem na jej terenie. Rozporządzenie **RODO** opracowały Parlament Europejski i Rada Europy. Zgodnie z RODO możesz żądać usunięcia z baz danych informacji o sobie.

s. 40 • **Rozpowszechnianie wizerunku** wymaga zezwolenia osoby na nim przedstawionej. Dbanie o własny wizerunek **jest obowiązkiem** każdego obywatela.

s. 41 • **Nigdy nie** udostępniaj lekkomyślnie swoich danych osobowych.

# II. Podstawy programowania i środowisko programistyczne



- Algorytmy
- Sposoby zapisu algorytmów
- Cechy algorytmów
- Języki programowania
- Eclipse
- Język C++
- Funkcje
- Strumieniowanie danych

# 6. Przypomnij sobie, czyli podstawy tworzenia algorytmów

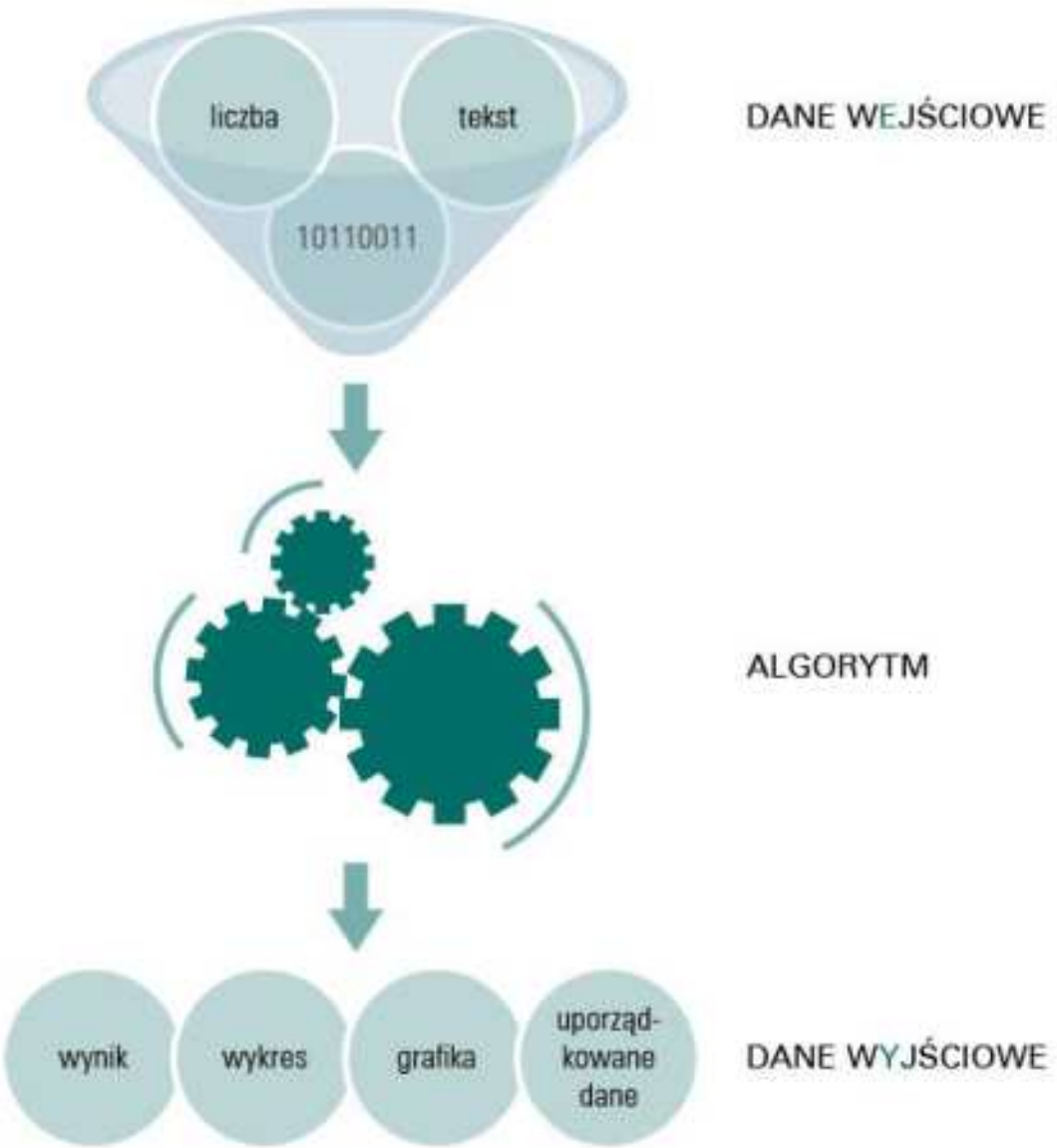
**NA TEJ LEKCJI:**

- poznasz sposoby zapisu algorytmów;
- nauczysz się specyfikacji algorytmu.

W trakcie nauki informatyki niejednokrotnie spotkałeś/spotkałaś się z algorytmami. Sposoby ich zapisu, interpretacji i edycji nie powinny być ci obce. Jest to na tyle ważne, że warto powtórzyć i przypomnieć sobie najważniejsze zagadnienia z tej dziedziny.

## 6.1. Prozą i graficznie, czyli różne sposoby zapisu algorytmów

Czy pamiętasz z poprzednich lat nauki definicję algorytmu? Jeśli nie, poniższy rysunek ci ją przypomni.




Rys. 6.1. Mechanizm przetwarzania danych za pomocą algorytmu

**Algorytm** to skończony i uporządkowany opis precyzyjnie sformułowanych czynności prowadzący do przetworzenia danych wejściowych na wyjściowe.

algorytm – przetwarzanie danych

**Dane wejściowe** nie muszą się jednoznacznie kojarzyć z danymi komputerowymi. Algorytmy mogą opisywać różne procesy. Jeśli nie wiesz, jak wykonać jakąś czynność, na przykład za pomocą edytora grafiki, sięgasz po instrukcje lub tutoriale. Znajdziesz ich wiele, chociażby w serwisie YouTube. Te instrukcje także można nazwać algorytmami. Większość doprowadzi cię do celu, jednak nie wszystkie są czytelne i zrozumiałe. Ze sposobami zapisu algorytmów jest podobnie – ich dobór zależy od specyfiki opisywanego problemu.

Najprostszym sposobem opisu rozwiązania problemu jest **zapis słowny**. Autor po prostu zdanie po zdaniu opisuje kolejne czynności.

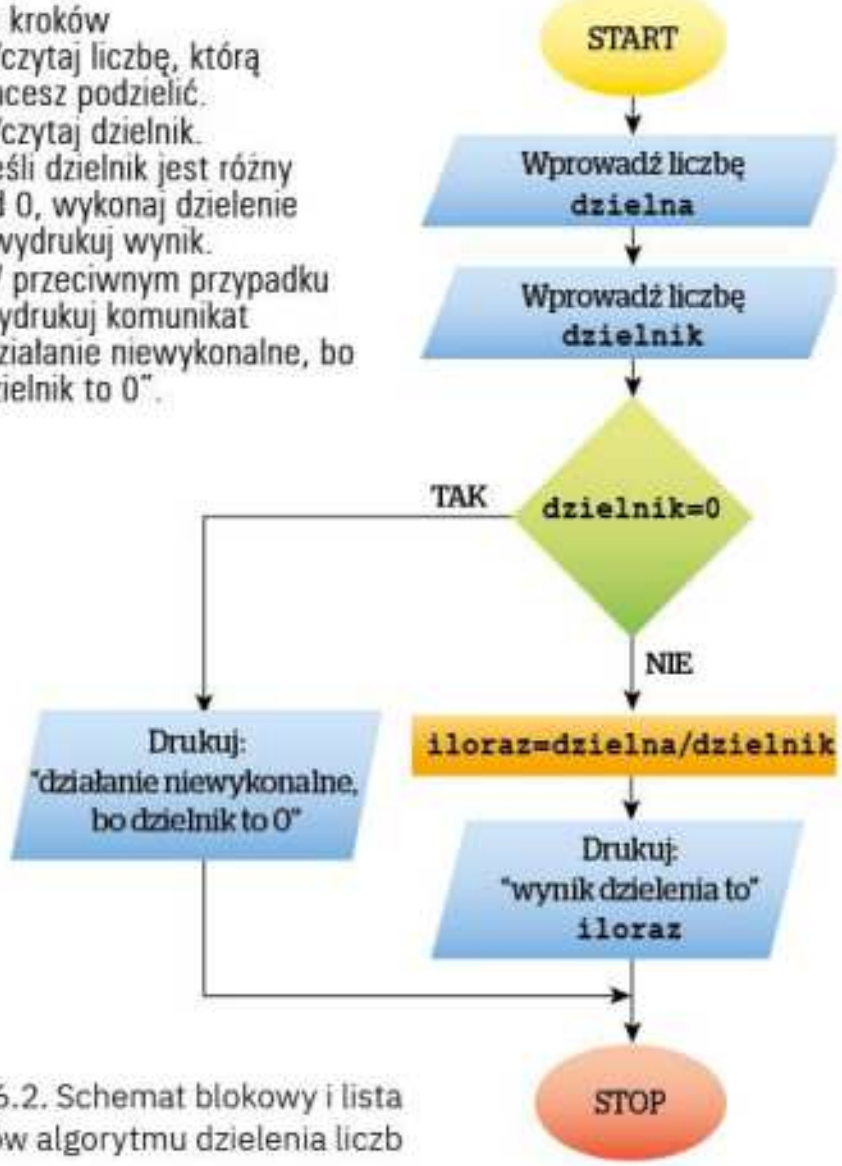
Precyzyjny opis powinien zawierać tak zwaną **listę kroków**. Ten sposób sprawdza się w nieskomplikowanych przypadkach. Ma jednak kilka wad. Po pierwsze lista sporządzona w języku polskim raczej nie będzie czytelna dla programisty z Chin. Po drugie opis komplikuje się, gdy w algorytmie występują pętle i warunki. Wtedy odczytujący go programista może łatwo się pogubić.

lista kroków

Wiesz już, że innym, bardzo popularnym sposobem zapisu algorytmu jest **schemat blokowy**. Przedstawia on w sposób graficzny kolejne czynności połączone strzałkami, co pozwala na tworzenie dowolnie skomplikowanych schematów

schemat blokowy

Lista kroków
1. Wczytaj liczbę, którą chcesz podzielić.
2. Wczytaj dzielnik.
3. Jeśli dzielnik jest różny od 0, wykonaj dzielenie i wydrukuj wynik. W przeciwnym przypadku wydrukuj komunikat „Działanie niewykonalne, bo dzielnik to 0".




Rys. 6.2. Schemat blokowy i lista kroków algorytmu dzielenia liczb

i łatwą ich weryfikację oraz testowanie. Jeśli w poszczególnych blokach znajdą się ogólnie stosowane znaki, jak wzory i symbole matematyczne, algorytm będzie czytelny i zrozumiały. Przykład prostego algorytmu dzielenia zapisanego w postaci schematu blokowego i listy kroków znajdziesz na rysunku 6.2. Analizując zapis graficzny, przypomnij sobie znaczenie poszczególnych bloków – **wejścia–wyjścia**, **decyzyjnego** i **operacyjnego**.

pseudokod ▸ Istnieje jeszcze jeden sposób zapisu algorytmu. Używają go głównie programiści, ponieważ zachowuje on strukturę podobną do języka programowania. Przy czym nie zachowuje on reguł składniowych konkretnego języka. Dzięki temu na jego podstawie łatwiej jest tworzyć programy komputerowe w różnych językach. Są jednak w nim zawarte konkretne sugestie rozwiązania problemu. Programista może więc skupić się jedynie na tworzeniu kodu w konkretnym języku programowania.

Poniższy przykład dotyczy działania hotelowego zamka do drzwi z czytnikiem kart zbliżeniowych.

```
Jeżeli wykryłeś kartę to
    jeżeli karta prawidłowa to
        otwórz drzwi;
    w przeciwnym przypadku
        zapal czerwoną kontrolkę;
```

Tak zapisany algorytm można w prosty sposób zaimplementować (czyli przystosować do pracy) w dowolnym języku programowania. Jeśli polskie określenia zastąpimy angielskimi, znanymi z różnych języków programowania (if...else), algorytm będzie zrozumiały na całym świecie.

cechy dobrego algorytmu ▸ Dobry algorytm powinien mieć następujące cechy:

- **Precyzyjne zdefiniowanie** – algorytm nie może pozostawiać dowolności w interpretacji zapisu. Każdy, kto z niego skorzysta, musi go rozumieć w taki sam sposób.
- **Skończoność** – algorytm musi po pewnej liczbie kroków dać wynik, czyli wygenerować dane wyjściowe. Liczba operacji musi być skończona.
- **Wykonywalność** – wszystkie operacje opisane w algorytmie muszą być możliwe do wykonania.
- **Efektywność** – algorytm powinien być tak ułożony, by wynik jego działania był dostępny po jak najmniejszym zaangażowaniu zasobów komputera i w jak najkrótszym czasie. Zależy to jednak nie tylko od algorytmu, lecz także od właściwości komputera, języka użytego do ułożenia programu, systemu operacyjnego i wielu innych parametrów.

## 6.2. Zanim zaczniesz, czyli tworzenie specyfikacji algorytmu

Bez względu na to, jakiego sposobu zapisu użyjesz przy tworzeniu algorytmu, musi on spełniać założenia projektu. Zapisuje się je w postaci tak zwanej **specyfikacji**. Powstaje ona podczas wstępnych prac nad algorytmem. Mogą to być na przykład uzgodnienia z zamawiającym lub rozwinięcie własnego pomysłu na program. Specyfikacja zazwyczaj składa się z trzech części. Najważniejszą, uzgadnianą z autorem pomysłu lub zamawiającym program, jest szczegółowy opis problemu. W specyfikacji można podać także warunki, które powinny być spełnione przez dane i wyniki. Dobrze, gdy w opisie znajdą się także zależności zachodzące między danymi wejściowymi a wynikami, na przykład wzory. Specyfikację można zapisać w dowolny sposób. Ważne, by uwzględniała wszystkie aspekty problemu. Tak przygotowana specyfikacja staje się podstawą do tworzenia algorytmu. Przeanalizujmy to na prostym przykładzie.

Tab. 6.1. Przykład specyfikacji problemu

| Problem | Obliczanie pierwiastków równania kwadratowego |
|---|---|
| Opis problemu | Dane jest równanie kwadratowe w postaci:<br>$y = ax^2 + bx + c$<br>Znając współczynniki *a*, *b*, *c*, wyznacz pierwiastki równania w dziedzinie liczb rzeczywistych. Uwzględnij wszystkie przypadki. |
| Dane wejściowe | współczynniki *a*, *b*, *c* |
| Dane wyjściowe | *x*1 i *x*2 dla delty większej od 0<br>*x* dla delty równej 0<br>komunikat o braku rozwiązania w dziedzinie liczb rzeczywistych dla delty mniejszej od 0. |

Tak przygotowana specyfikacja może stać się podstawą do utworzenia algorytmu (rys. 6.3).

```
start
wprowadź liczby a, b, c;
delta=b²-4ac;
jeżeli delta<0 drukuj „Brak rozwiązań w dziedzinie
liczb rzeczywistych”;
jeżeli delta=0 {x=-b/2a; drukuj x;};
jeżeli delta>0 {x1=(-b+pierwiastek(delta))/2a);
x2=(-b-pierwiastek(delta))/2a); drukuj x1, x2;}
stop.
```

Rys. 6.3. Przykładowy algorytm w postaci pseudokodu

W specyfikacji, a następnie algorytmie opisuje się dane wejściowe i wyjściowe. Przypomnij sobie, czym one są. **Dane wejściowe** w wielu przypadkach to nie tylko liczby lub tekst wprowadzany za pośrednictwem klawiatury. Wiele z nich dostaje się do komputera za pośrednictwem urządzeń peryferyjnych, takich jak mikrofon, skaner, kamera, mierniki temperatury, ciśnienia, zawartości pyłów zawieszonych i innych wielkości fizycznych. Wszystkie muszą być przetworzone przez algorytm na dane wyjściowe.




Rys. 6.4. Przykłady danych wejściowych algorytmu

**Dane wyjściowe** powstają na skutek przetwarzania danych wejściowych zgodnie z algorytmem. Tu także mamy do czynienia z rozmaitymi typami danych. Dane wyjściowe mogą być jednocześnie wejściowymi dla innych algorytmów. Przykładem może być klawiatura elektroniczna, która w zasadzie jest samodzielnym komputerem odczytującym stan klawiszy i wysyłającym odpowiednie kody – dane do komputera nadrzędnego, na przykład komputera osobistego, centralki domofonu, centralki zamka elektronicznego. Innym przykładem jest drukarka, dla której program – sterownik tworzy dane na podstawie dokumentów przeznaczonych do wydrukowania.

Takich przykładów można znaleźć jeszcze bardzo dużo. Domyślasz się już, że rodzaj danych wyjściowych może być taki sam jak danych wejściowych albo różnić się od nich w znaczny sposób. Algorytm może wygenerować liczby, teksty, ale także obraz, dźwięk, dane dla innych algorytmów.

Przed tobą ćwiczenia z układania algorytmów. Powodzenia.

## ZADANIA DO WYKONANIA

1. Algorytm z rysunku 6.2. zapisz w postaci pseudokodu. Porównaj wszystkie trzy sposoby zapisu rozwiązania tego problemu. Który z nich jest dla ciebie bardziej czytelny?
2. Na rysunku 6.4. przedstawiono niektóre z możliwych rodzajów danych wejściowych. Scharakteryzuj każdy z przykładów pod kątem pochodzenia lub urządzenia, które je wytworzyło.
3. Poznałeś/poznałaś kilka rodzajów danych wyjściowych. Zastanów się, przejrzyj zasoby sieci i wymień kilka innych. Omów je krótko w zeszycie.
4. Pewien algorytm przetwarza dane z kamery HD na obraz z nałożonymi efektami graficznymi. Został wykorzystany w pewnym programie. Jakie może być jego przeznaczenie?

5*. Jakiego rodzaju dane wejściowe będą wykorzystywane przez algorytm, który opisuje obliczanie prędkości obrotowej koła? Omów konstrukcję potrzebnego urządzenia.

## PODSUMOWANIE LEKCJI

- Każdy algorytm **przetwarza dane wejściowe na wyjściowe** w ściśle określony sposób. **Dane wyjściowe algorytmu** mogą być danymi wejściowymi innego algorytmu. ◂ s. 47
- **Lista kroków** to zapis algorytmu za pomocą słów. ◂ s. 47
- **Schemat blokowy** to graficzny sposób zapisu algorytmu. ◂ s. 47
- Do **cech dobrego algorytmu** należą: **precyzyjne zdefiniowanie procesów, skończoność, wykonywalność, efektywność**. ◂ s. 48
- **Pseudokod** jest sposobem zapisu algorytmu zbliżonym do języka programowania. Nie stosuje się w nim składni konkretnego języka. ◂ s. 48

# 7. Algorytmy. Utrwalamy swoją wiedzę

**NA TEJ LEKCJI:**

- nauczysz się pisać algorytmy za pomocą pseudokodu, listy kroków oraz schematów blokowych.

Wiesz już, że algorytmy składają się z szeregu kroków, które należy wykonać tak, aby z danych wejściowych otrzymać pożądane rozwiązanie. Oczywiście poszczególne kroki trzeba wykonywać w odpowiedniej kolejności.

## 7.1. Sprawdźmy, czy trójkąt jest prostokątny

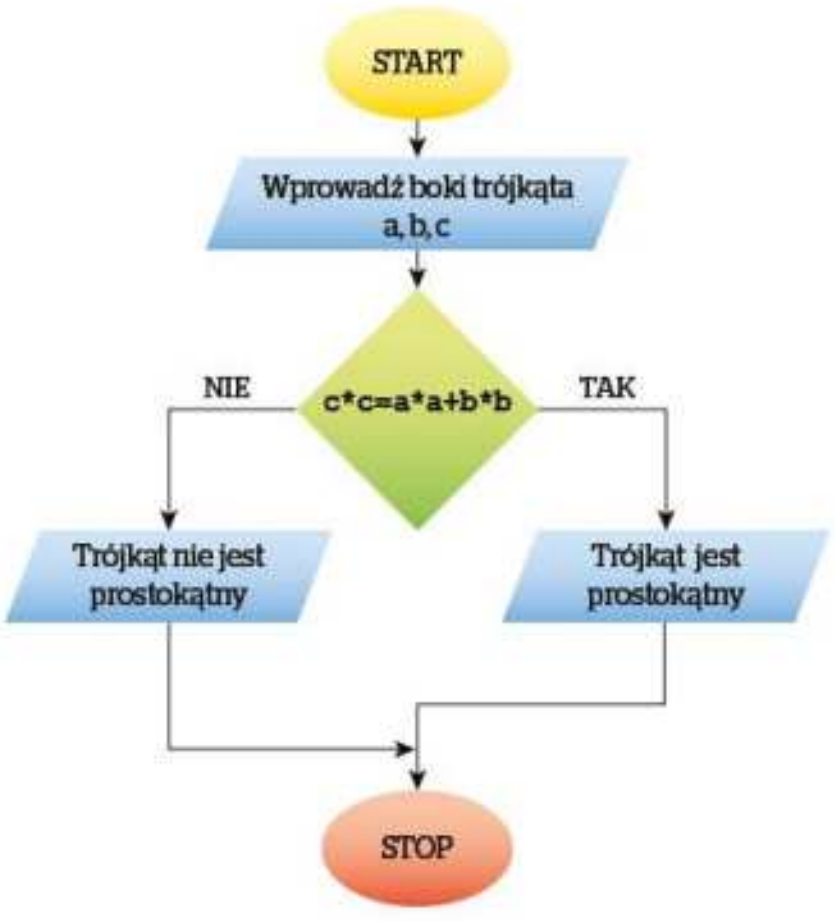
algorytm ▸ Zacznijmy od małej rozgrzewki. Na początek sprawdzimy, czy dany trójkąt jest prostokątny. Z pewnością nie raz rozwiązywaliście takie zadanie na matematyce.

Z twierdzenia Pitagorasa wiemy, że boki trójkąta prostokątnego muszą spełnić zależność: $c^2=a^2+b^2$.

Jeżeli chcemy sprawdzić, czy trójkąt jest prostokątny, musimy sprawdzić, czy jego boki spełniają zależność podaną w twierdzeniu Pitagorasa.

Poznaliście wcześniej dwie metody tworzenia algorytmów – listę kroków i schemat blokowy. Wykorzystamy obie do tego zadania.

### Lista kroków i schemat blokowy




Rys. 7.1. Schemat blokowy sprawdzania, czy trójkąt jest prostokątny

**Lista kroków**

zapisywanie algorytmów

1. Wczytaj boki trójkąta *a*, *b*, *c*.
2. Sprawdź, czy jest spełniona zależność: *c*c=a*a+b*b*.
3. Jeżeli zależność jest spełniona, wypisz: Trójkąt jest prostokątny. W przeciwnym razie wypisz: Trójkąt nie jest prostokątny.

## 7.2. Sprawdźmy, czy podana liczba jest liczbą parzystą

Problem, czy dana liczba jest liczbą parzystą, często podlega analizie. Z lekcji matematyki wiemy, że liczba jest parzysta, jeśli po podzieleniu jej przez dwa wynik dzielenia wyjdzie bez reszty. Pora zapisać to w postaci algorytmu.

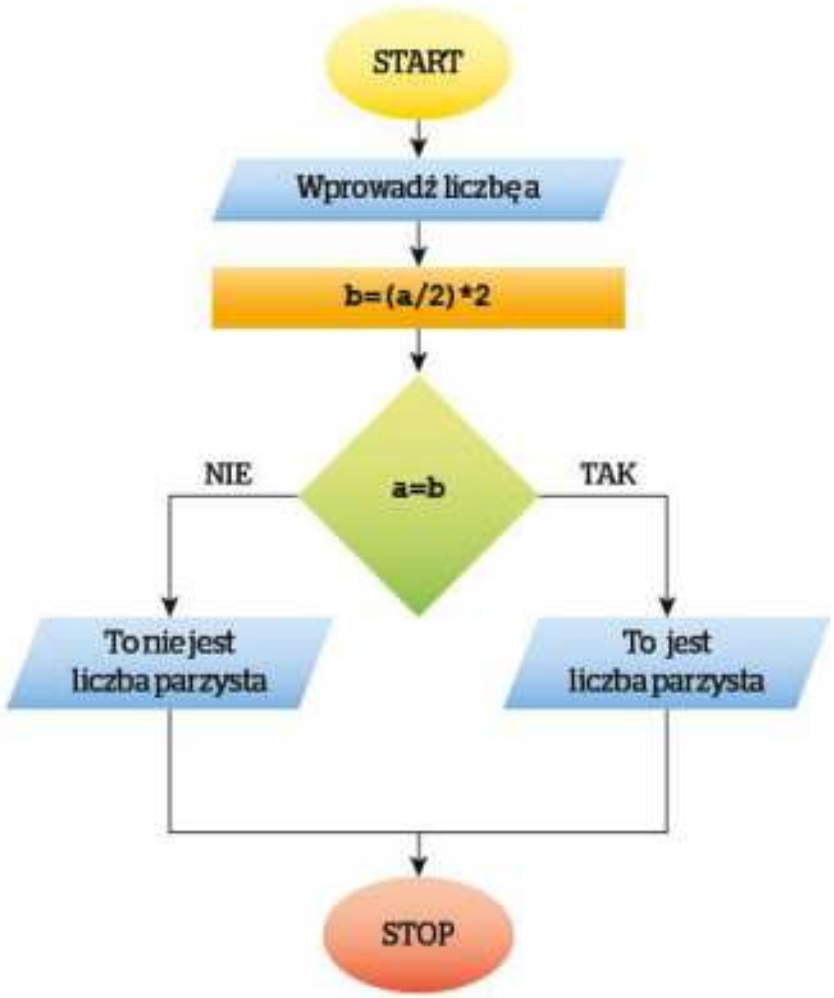
Aby dowiedzieć się, czy liczba jest parzysta, trzeba sprawdzić, czy reszta z dzielenia równa się 0, czy też zostanie 1 reszty. Najłatwiej będzie podzielić daną liczbę przez 2 i odrzucić resztę, a następnie ponownie pomnożyć ją przez 2 – jeżeli w wyniku otrzymamy ponownie tę samą liczbę, to nasza liczba jest parzysta.

**Lista kroków i schemat blokowy**

schemat blokowy

**Lista kroków**

1. Wczytaj liczbę *a*.
2. Podziel liczbę *a* przez 2, odrzuć resztę i pomnóż wynik przez 2.
3. Sprawdź, czy otrzymany wynik jest równy liczbie *a*.
4. Jeżeli otrzymany wynik jest równy liczbie *a*, oznacza to, że wprowadzona liczba jest liczbą parzystą, jeżeli nie jest równy, wprowadzona liczba jest liczbą nieparzystą.




Rys. 7.2. Schemat blokowy sprawdzania, czy dana liczba jest parzysta

## 7.3. Sprawdźmy, czy suma jest większa od różnicy

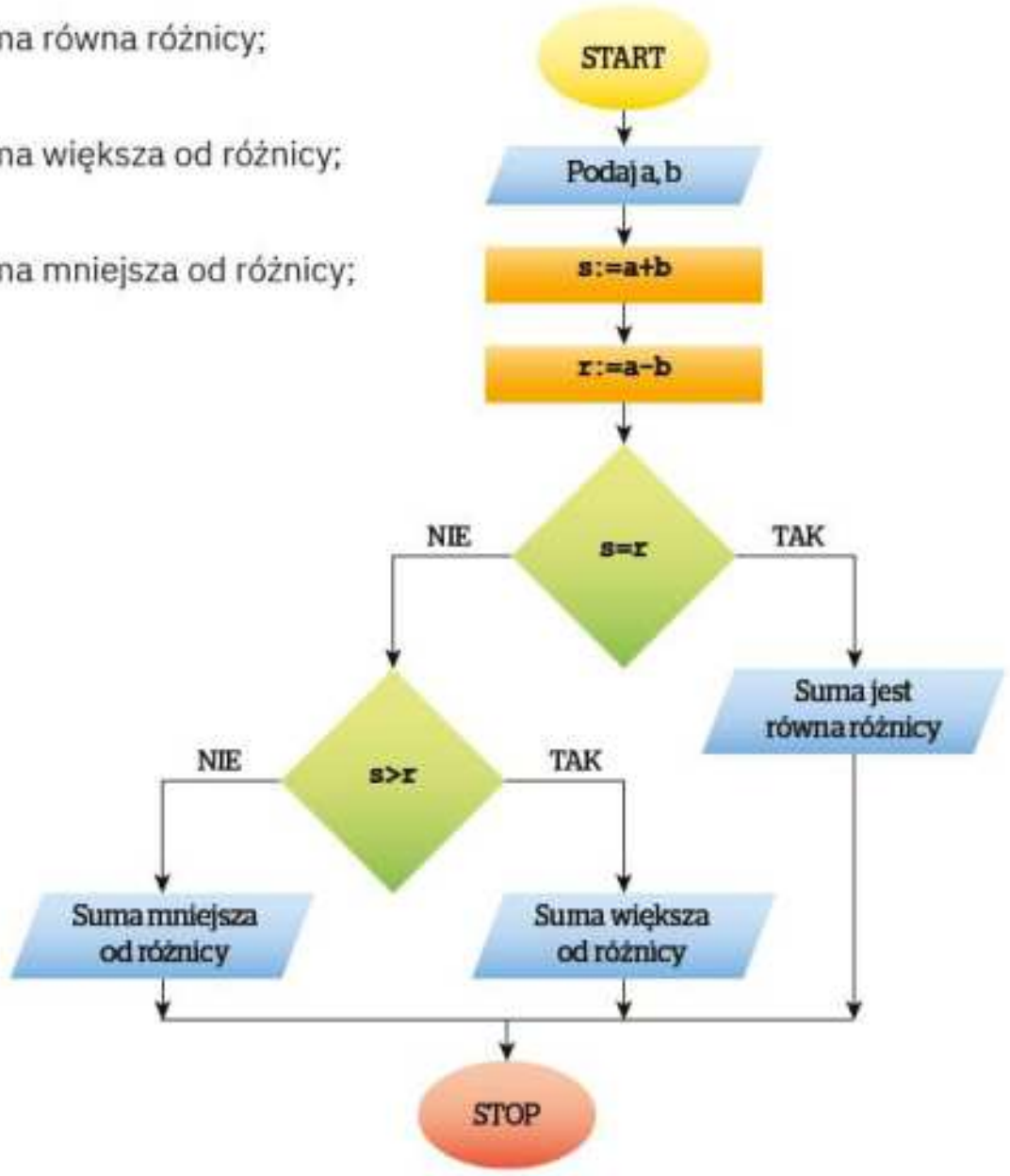
Kolejny algorytm, który stworzymy, sprawdzi, czy suma wczytanych dwóch liczb całkowitych jest większa od różnicy.

Tym razem stworzymy algorytm jako pseudokod i jako schemat blokowy. Zaczniemy od pseudokodu.

**Pseudokod sprawdzający, czy suma dwóch liczb całkowitych jest większa od ich różnicy:**

```
PROGRAM Suma_roznica;
ZMIENNE a, b, s, r: całkowite;
ZACZNIJ
    WCZYTAJ (a,b);
    s:= a+b;
    r:= a–b;
    JEŻELI s=r;
    WYPISZ: Suma równa różnicy;
    JEŻELI s>r;
    WYPISZ: Suma większa od różnicy;
    JEŻELI s<r;
    WYPISZ: Suma mniejsza od różnicy;
ZAKOŃCZ
```

**Schemat blokowy**




Rys. 7.3. Schemat blokowy sprawdzający, czy suma liczb całkowitych jest większa od różnicy tych liczb

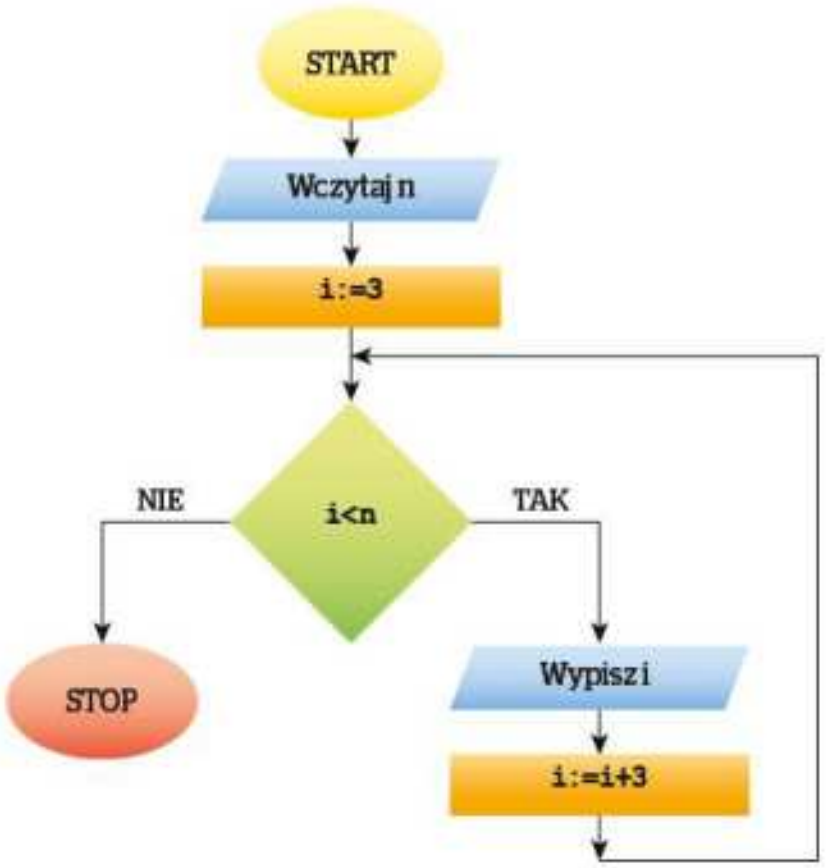
## 7.4. Liczby podzielne przez 3

Tym razem stwórzmy algorytm, który będzie wypisywał liczby z przedziału od 0 do *n* podzielne przez 3. Tak jak poprzednio zapiszemy ten algorytm w formie pseudokodu i schematu blokowego.

**Pseudokod wypisujący liczby od 0 do *n* podzielne przez 3**

```
PROGRAM Podzielne_przez_3;
ZMIENNE n, i: całkowite;
ZACZNIJ
    WCZYTAJ n;
    i:=3;
    JEŻELI i<n
    WYPISZ i
    i:=i+3
    JEŻELI i<n
ZAKOŃCZ
```

**Schemat blokowy**




Rys. 7.4. Schemat blokowy wypisujący liczby z zakresu od 0 do *n* podzielne przez 3

## ZADANIA DO WYKONANIA

1. Napisz dowolną metodą algorytm obliczający wysokość w trójkącie prostokątnym, jeśli dane są wartości boków *a, b, c*.
2. Utwórz algorytm w postaci schematu blokowego obliczającego pole trapezu.
3. Korzystając z treści zawartych w rozdziale 7.2., przekształć przedstawiony tam schemat blokowy w taki, który sprawdza, czy dana liczba jest liczbą całkowitą nieparzystą mieszczącą się w przedziale od 10 do 100.
4. Napisz algorytm w postaci pseudokodu, który sprawdzi, czy różnica dwóch liczb całkowitych jest liczbą większą od 0.
5. Napisz algorytm w postaci pseudokodu, który poda wynik równania kwadratowego.
6. Napisz algorytm w postaci schematu blokowego, który poda wynik równania $ax+b=0$.
7. Napisz algorytm za pomocą pseudokodu i schematów blokowych. Algorytm ma obliczać średnią arytmetyczną trzech liczb, których wartości są wczytywane podczas działania algorytmu. Wynik wyświetl na ekranie monitora.
8. Napisz algorytm za pomocą pseudojęzyka i schematów blokowych. Algorytm ma obliczać objętość ($V$) oraz sumę długości wszystkich krawędzi prostopadłościanu ($D$). Wartości zmiennych *a, b, c* są podawane podczas działania algorytmu.

## PODSUMOWANIE LEKCJI

s. 52 ▸ • **Algorytm** składa się z szeregu kroków, które mają nas doprowadzić do rozwiązania.

s. 53 ▸ • Algorytmy można zapisać w różnych postaciach, np. w formie **listy kroków, schematu blokowego lub pseudokodu**.

s. 53 ▸ • Najbardziej popularnym zapisem jest **schemat blokowy**, który ułatwia w późniejszym etapie kodowania napisanie właściwego programu.

# 8. Pozycyjne reprezentacje liczb, czyli tablice w algorytmice

**NA TEJ LEKCJI:**

- dowiesz się, czym są liczby doskonałe;
- dowiesz się, czym są tablice i schemat Hornera.

Czym jest doskonałość? To stan lub efekt w swojej istocie bezbłędny, dopracowany, bezproblemowy, nienaganny. Doskonałość może odnosić się do cech charakteru człowieka, do przedmiotu czy dzieła, ale również to pojęcie spotykamy w matematyce.

Aby zamienić jakąkolwiek liczbę w dowolnym systemie liczbowym na system dziesiętny, wykorzystujemy do tego celu schemat Hornera wartości wielomianu, a w algorytmie korzystamy z tablic.

## 8.1. Liczby doskonałe

liczby doskonałe

Pojęcie doskonałości w matematyce dotyczy liczb doskonałych.

dzielnik właściwy

Liczba doskonała jest sumą wszystkich swoich dzielników właściwych – to znaczy od niej mniejszych. Najmniejszą liczbą doskonałą jest 6, ponieważ $6 = 3 + 2 + 1$. Kolejna taka liczba to 28, gdyż $28 = 14 + 7 + 4 + 2 + 1$.

Spróbujmy zbudować algorytm w postaci listy kroków, który sprawdzi, czy dana liczba jest liczbą doskonałą.

1. Wczytaj liczbę *x*, która będzie sprawdzana, czy jest liczbą doskonałą.
2. Ustaw sumę na 0.
3. Dla każdej liczby *i* od 1 do zaokrąglonego *x*/2 sprawdź, czy jest ona dzielnikiem liczby *x*. Jeżeli jest, to dodaj *i*.
4. Jeśli suma jest równa wprowadzonej liczbie *x*, to ta liczba jest liczbą doskonałą.

## 8.2. Liczby doskonałe, parzyste i pierwsze

Leonard Euler i liczby parzyste

Szwajcarski fizyk i matematyk Leonard Euler, wykorzystując sposób znajdowania liczb doskonałych parzystych opisanych przez Euklidesa, udowodnił, że liczby doskonałe parzyste można opisać wzorem $(2^p - 1)*2^{p-1}$, gdzie $p$ jest liczbą pierwszą.

## 8.3. Pozycyjne reprezentacje liczb

W tym podrozdziale zajmiemy się zamianą systemu dwójkowego na system dziesiętny. Systemu dwójkowego – inaczej nazywanego systemem binarnym – używa się w komputerach przede wszystkim dlatego, że łatwiej fizycznie zaprojektować urządzenie, które będzie operowało tylko na dwóch stanach – prawdzie lub fałszu, sygnale lub jego braku.

Urządzenia, takie jak pamięć operacyjna, przechowują bity, wykorzystując prąd. Jeśli dana jednostka pamięci nie ma w danej chwili napięcia, to mamy do czynienia z cyfrą 0, jeżeli natomiast to napięcie się pojawi, to z cyfrą 1. Podobnie dyski twarde wykorzystujące nośniki magnetyczne wykorzystują namagnesowanie – jeżeli dwie komórki mają ten sam ładunek, to mamy 0, w przeciwnym razie 1.

Każdą liczbę zapisaną w systemie dwójkowym możemy przedstawić za pomocą systemu dziesiętnego i odwrotnie. Przeanalizujmy poniższy przykład, a później spróbujmy zapisać go w formie algorytmu.

Liczbę $(10111)_2$ zapisaną w systemie binarnym zamieniamy na system dziesiętny. Zaczynamy zawsze od prawej strony i kolejne cyfry mnożymy przez kolejne potęgi liczby 2, począwszy od $2^0=1$.

$$(10111)_2=1*2^4+0*2^3+1*2^2+1*2^1+1*2^0=$$
$$=1*16+0*8+1*4+1*2+1*1=16+0+4+2+1=23$$

Zapiszmy teraz te obliczenia w formie algorytmu. Oczywiście algorytm będzie uniwersalny i nie będzie dotyczył tylko tej konkretnej liczby podanej w przykładzie. Zanim jednak się za to zabierzemy, przypomnijmy, co to są tablice.

**tablica** ▸ Tablica to kontener uporządkowanych danych tego samego typu, w którym poszczególne elementy są dostępne za pomocą kluczy (indeksów). Rozmiar tablicy jest albo ustalony z góry (tablice statyczne), albo można go zmieniać w trakcie wykonywania programu (tablice dynamiczne). Tablice jednowymiarowe mogą przechowywać inne tablice, dzięki czemu uzyskuje się tablice wielowymiarowe.

**schemat Hornera** ▸ Przejdźmy do algorytmu. Zamianę liczby zapisanej w kodzie dwójkowym na liczbę dziesiętną zrealizujemy schematem Hornera wartości wielomianu. Pamiętajmy, że liczba naturalna $a$ zapisana w systemie dziesiętnym ma następującą postać w systemie binarnym dla pewnego $n$:

$$a=b_{n-1}*2_{n-1}+b_{n-2}*2_{n-2}+...+b_1*2^1+b_0*2^0$$

Wykorzystując ten algorytm, można zbudować taki algorytm, który zamieni liczbę z dowolnego systemu liczbowego na system dziesiętny. Oczywiście musimy zmienić powyższy zapis schematu Hornera wartości wielomianu, zamieniając liczbę 2 na $s$, które będzie symbolizowało dowolny system liczbowy:

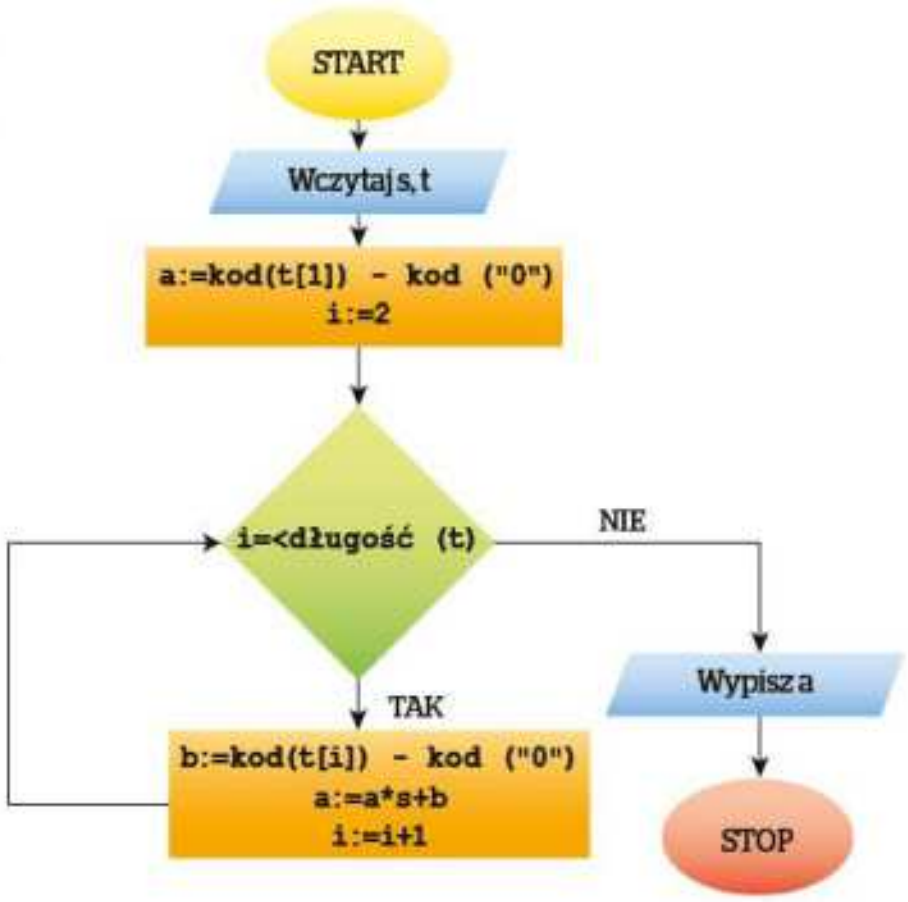
$$a=b_{n-1}*s_{n-1}+b_{n-2}*s_{n-2}+...+b_1*s^1+b_0*s^0$$

Jeżeli chcemy zamienić liczbę z systemu, który ma więcej niż 10 znaków, na przykład z systemu szesnastkowego na system dziesiętny, to musimy od 11 znaku używać liter.

Dane wejściowe:

$s$ – podstawa systemu pozycyjnego zapisu liczby $p \in N, p \in \{2, 3, \ldots, 10\}$,

$t$ – tekst zawierający ciąg znaków ASCII, przedstawiających poprawny zapis liczb.




Rys. 8.1. Schemat blokowy zamiany liczby binarnej na dziesiętną

## ZADANIA DO WYKONANIA

1. Zapisz algorytm w postaci schematu blokowego do punktu pierwszego rozdziału.
2. Zapisz algorytm w postaci pseudokodu do punktu pierwszego rozdziału.
3. Zapisz algorytm w postaci pseudokodu, który sprawdzi, czy dana liczba jest liczbą doskonałą, podzielną przez 4 i składa się z 6 dzielników.
4. Zapisz algorytm w postaci pseudokodu, który zamieni liczbę zapisaną w systemie szesnastkowym na liczbę w systemie dziesiętnym.
5. Zapisz algorytm w postaci schematu blokowego, który zamieni liczbę w dowolnym systemie liczbowym na liczbę w systemie dziesiętnym.

6*. Zapisz algorytm w postaci listy kroków i schematu blokowego, który zamieni liczbę z systemu dwójkowego na liczbę w systemie ósemkowym.

## PODSUMOWANIE LEKCJI

- **Liczba doskonała** to liczba, która jest sumą wszystkich swoich dzielników właściwych. s. 57
- **Dzielnik właściwy** jest zawsze mniejszy od dzielnej. s. 57
- **Euklides** jako pierwszy wskazał, że liczby doskonałe to liczby parzyste. **Leonard Euler** znalazł sposób na wyszukanie liczby doskonałej dzięki liczbom pierwszym. s. 57
- Do zamiany liczb z dowolnego systemu liczbowego na system dziesiętny wykorzystuje się **schemat Hornera** wartości wielomianu. s. 58
- **Tablice** są potrzebne przy tworzeniu algorytmów do zamiany systemów liczbowych na system dziesiętny. Mogą mieć rozmiar z góry ustalony (**tablice statyczne**) lub można ten rozmiar zmieniać w trakcie wykonywania programu (**tablice dynamiczne**). s. 58

# 9. Narzędzia, czyli jak dobrać i skonfigurować środowisko programistyczne

**NA TEJ LEKCJI:**

- poznasz nazwy popularnych języków programowania;
- poznasz środowisko IDE;
- dowiesz się, jak zainstalować polskie menu w Eclipse.

Który z języków programowania jest najlepszy? Wiesz już, że nie można odpowiedzieć jednoznacznie na to pytanie. Można jednak ocenić przydatność do danych zastosowań i sprawdzić, który z nich cieszy się największym powodzeniem wśród programistów. Wybierzemy dwa języki: C++ oraz Java. Wraz z nimi poznasz także środowiska programistyczne.

## 9.1. Wieża Babel, czyli jak wybierać język programowania

Wśród języków, którymi posługujemy się w życiu, dominuje angielski. Samoistnie stał się językiem międzynarodowym. Dlaczego? To zagadnienie dla socjologów, a zjawisko zapewne jest skomplikowane i szczegółowo badane. A co z językami programowania? Ich także jest wiele.

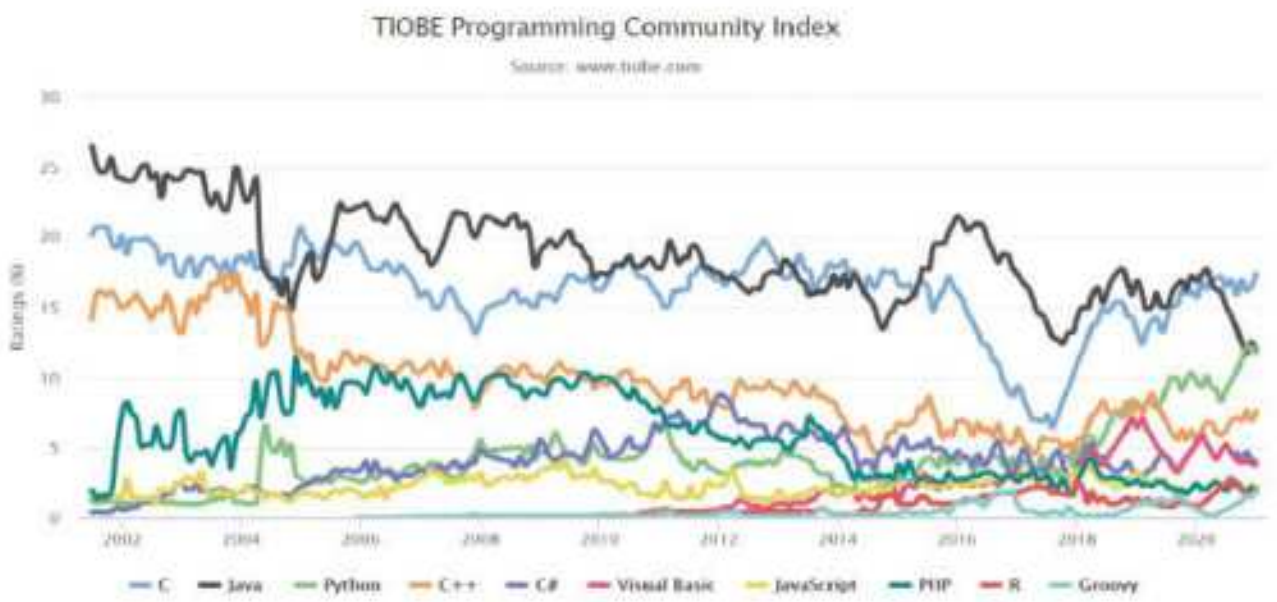
lista popularnych języków programowania

Wśród nich znajdują się takie, których powszechność stosowania można porównać z językiem angielskim. Sprawdzisz to na stronie *https://www.tiobe.com/tiobe-index/* (rys. 9.1.).

| Jul 2018 | Jul 2017 | Change | Programming Language | Ratings | Change |
|---|---|---|---|---|---|
| 1 | 1 | | Java | 16.139% | +2.37% |
| 2 | 2 | | C | 14.662% | +7.34% |
| 3 | 3 | | C++ | 7.615% | +2.04% |
| 4 | 4 | | Python | 6.361% | +2.82% |
| 5 | 7 | ^ | Visual Basic .NET | 4.247% | +1.20% |
| 6 | 5 | v | C# | 3.795% | +0.28% |
| 7 | 6 | v | PHP | 2.832% | -0.26% |
| 8 | 8 | | JavaScript | 2.831% | +0.22% |
| 9 | - | ^^ | SQL | 2.334% | +2.33% |
| 10 | 18 | ^^ | Objective-C | 1.453% | -0.44% |

Rys. 9.1. Klasyfikacja popularności języków programowania wg TIOBE na podstawie ocen użytkowników

Popularność poszczególnych języków programowania zmienia się i zależy od wielu czynników – im szerszy jest zakres zastosowania, tym większy udział (rys. 9.2.). Dane ze strony TIOBE wynikają z badań opinii użytkowników.




Rys. 9.2. Rozkład popularności języków programowania na podstawie ocen użytkowników na przestrzeni lat

Analizując wyniki badań, można łatwo zauważyć dominację Java, C i C++. W związku z tym w naszych dalszych rozważaniach i nauce programowania na poziomie podstawowym będziemy wykorzystywać język C++.

**Uwaga!**

Porównując różne języki programowania, należy brać pod uwagę nie tylko popularność, lecz także specyfikę danego projektu. Nie można jednoznacznie określić, który z języków jest ogólnie najlepszy.

## 9.2. Krótko o filiżance, czyli dlaczego Java jest tak popularna

Jeśli porównamy program ułożony w C++ i Java, możemy dojść do wniosku, że w Javie jest on dłuższy (tab. 9.1.).

◂ języki programowania

Tab. 9.1. Fragmenty programów w Java i C++ wykonujących to samo zadanie

| Java | C++ |
|---|---|
| `int liczba=input.nextInt();` | `Scanf("%d", &liczba);` |

Czy to wada, czy zaleta? Z jednej strony używa się większej liczby znaków, ale z drugiej – polecenia są znacznie bardziej zrozumiałe. Przekonasz się o tym, wykonując ćwiczenia z naszego podręcznika i samodzielnie układając programy w obu językach.

Niedogodność wynikająca z używania dłuższych nazw rozkazów i poleceń nie utrudni pracy, jeśli zostanie dobrany edytor, który automatycznie uzupełnia tekst programu.

Wielu programistów wśród zalet Java wymienia jej niezależność od platformy, na której będzie uruchamiany program. Znając ten język, można tworzyć programy dla komputerów PC, smartfonów, serwerów stron itp.

Dla Java napisano wiele standardowych bibliotek (funkcji wykonujących różne działania). Dzięki nim nie ma potrzeby pisania ich od podstaw. Użycie bibliotek jest bardzo proste, o czym przekonasz się w dalszej części nauki programowania. Podobne biblioteki istnieją także dla innych języków. Ich liczba ciągle się zwiększa.

## 9.3. C++, czyli język nie tylko dla PC

język C++ ▸ C++ jest językiem uniwersalnym. Dzięki swoim właściwościom, na przykład możliwości precyzyjnego wykorzystania sprzętu, pozwala na efektywne gospodarowanie zasobami komputera, chociażby pamięcią, i aktywne oszczędzanie energii w urządzeniach mobilnych. Dużą zaletą jest szybkość działania, co oczywiście dla danego programu także zależy od umiejętności programisty.

Czy wiesz, w jakim języku pisze się na przykład systemy operacyjne, pakiety biurowe, skomplikowane gry? W większości tych zastosowań język C++ nie ma obecnie sobie równych. Oczywiście także i w tej dziedzinie powstają nowe języki i ich odmiany. Być może, gdy będziesz już zawodowym programistą, dzisiejszy układ zmieni się całkowicie. Jednak umiejętność programowania w Java i C++ pozwoli na szybkie opanowanie nowych narzędzi programistycznych i języków.

Języka C++ używa się także bardzo często do programowania mikrosterowników, robotów i innych urządzeń sterowanych komputerowo. Także amatorzy elektroniki i informatyki mogą programować w C++ niewielkie systemy mikroprocesorowe z rodziny ARDUINO lub bardziej skomplikowane Raspberry Pi.

## 9.4. IDE dla Java, czyli poznajemy środowisko programistyczne ECLIPSE

IDE ▸ Czym jest **IDE** (ang. Integral Development Environment, czytaj: integral dewelopment enwajorment)? Jeśli programowałeś/programowałaś już w szkole podstawowej, zapewne pamiętasz, że to program lub zespół programów służących do tworzenia, testowania, modyfikowania i konserwacji programów komputerowych w danym języku. Często takie środowiska pozwalają programować w różnych językach, jak Java i C++. Najprostsze, graficzne zapewne znasz z programowania w **Scratch** lub **mBlock**.

Eclipse ▸ Do programowania w Java i C++ możesz użyć **Eclipse**. Jego zalety poznasz w trakcie nauki programowania.

Jak każdy program komputerowy IDE także wymaga uruchomienia i konfiguracji. Zatem – do dzieła.

Przygotowanie środowiska programistycznego do pracy wymaga kilku wcześniej zdobytych przez ciebie umiejętności. Możesz więc skorzystać z krótkiego algorytmu (rys. 9.3.) lub – jeśli napotkasz kłopoty – sięgnąć do szczegółowego opisu umieszczonego poniżej.




Rys. 9.3. Instalacja IDE Eclipse w komputerze PC

IDE nie zawiera niezbędnego do poprawnej pracy oprogramowania Java. W twoim systemie prawdopodobnie jest ono zainstalowane. Jeśli nie, Eclipse się nie uruchomi. Nie zrażaj się tym i odwiedź stronę *www.oracle.com/technetwork/java/javase/downloads* (rys. 9.4.) – postępuj według opisów lub rysunku 9.4. Wybierz JDK (Java Development Kit), ponieważ to darmowe oprogramowanie udostępniające środowisko niezbędne do programowania w języku Java. Takiego właśnie będziesz potrzebować do prawidłowej pracy Eclipse. Instalacja powinna przebiegać automatycznie. Czytaj uważnie komunikaty instalatora. Jeśli wykryje on starszą wersję oprogramowania i zaproponuje ci jej usunięcie – wyraź zgodę. Zwróć uwagę na wybór odpowiedniej wersji dla twojego systemu operacyjnego.

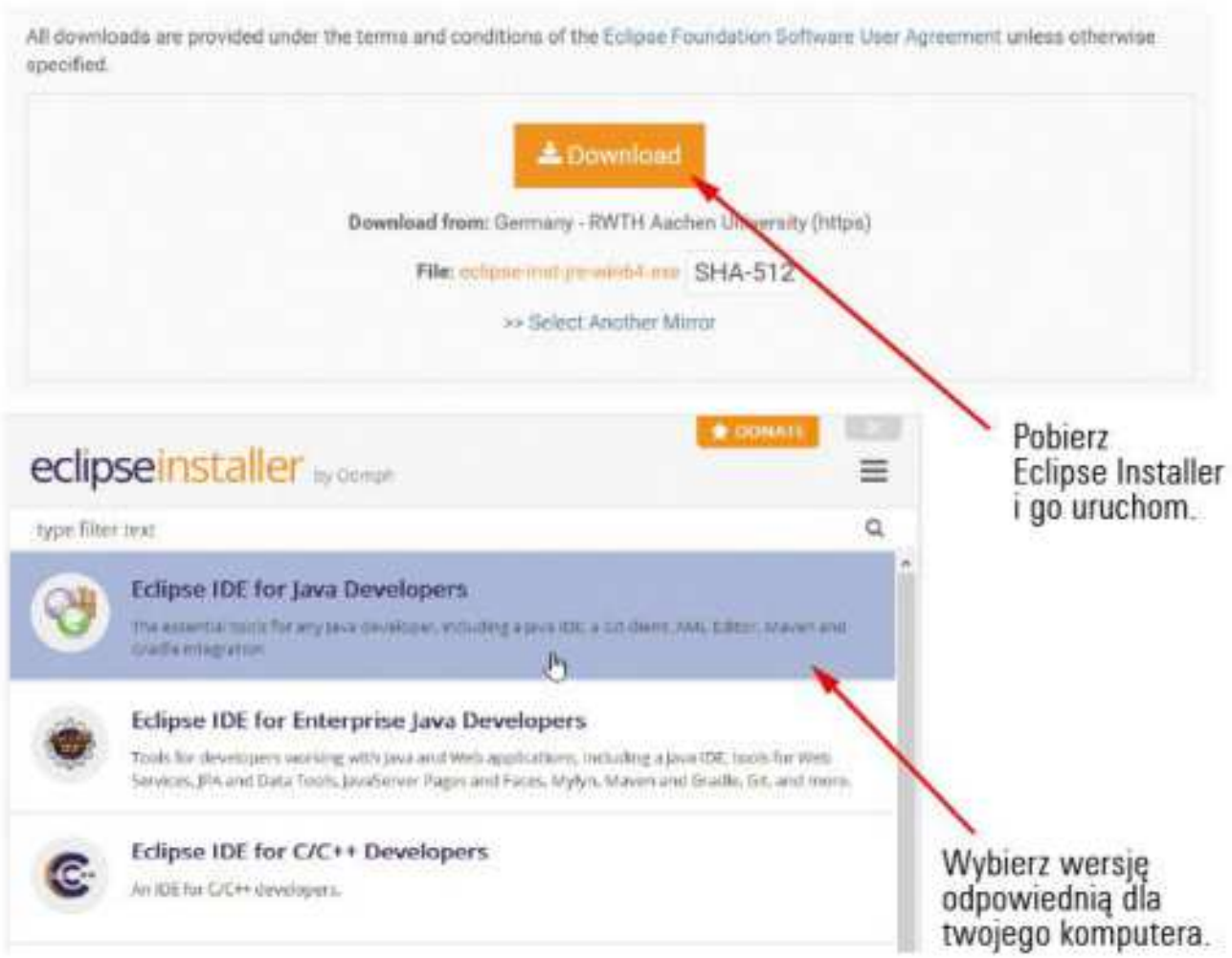



Rys. 9.4. Instalacja oprogramowania Java Development Kit

Wszystko, czego teraz potrzebujesz do zainstalowania IDE dla Java, znajdziesz na stronie *http://www.eclipse.org*.

Pobierz Eclipse Installer i go uruchom.

IDE dla Javy i C++ różnią się, dlatego należy pobrać odpowiednie pliki – **Eclipse IDE for Java Developers**. Rozpocznijmy od środowiska do programowania w Java (rys. 9.5.). Przygotowaniem IDE dla C++ zajmiemy się w następnym rozdziale.



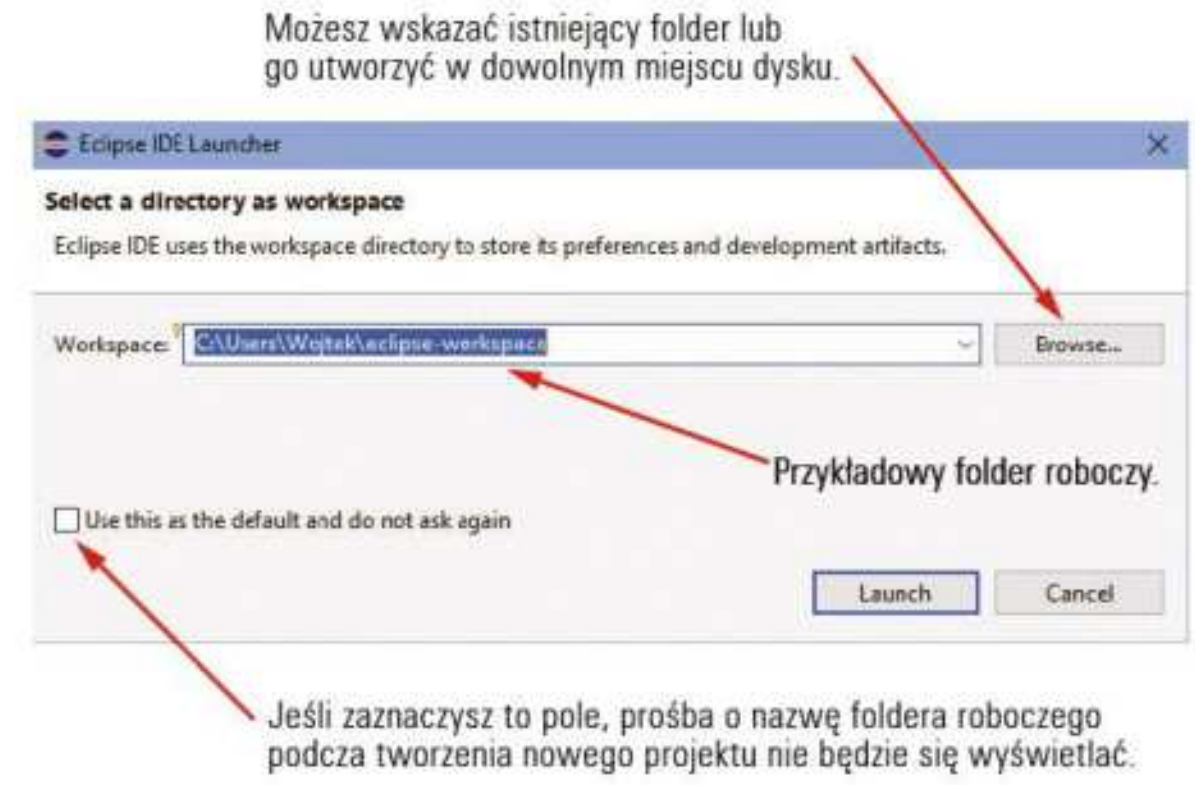

Rys. 9.5. Wybór IDE ze strony www.eclipse.org/downloads/packages/

Pobrany plik musisz rozpakować. Został on poddany kompresji ZIP, dlatego twój system poradzi sobie z tym bez instalowania dodatkowego oprogramowania. Wszystkie pliki i foldery Eclipsa wypakuj do jednego foldera. Może być dowolny, lecz dla porządku umieść je w folderze utworzonym w *C:\Program Files*, na przykład *C:\Program Files\Eclipse Java*. Aby zyskać szybszy dostęp do Eclipse, utwórz skrót do pliku *eclipse.exe* i umieść go na pulpicie. Nadaj mu nazwę Eclipse dla Java, by odróżnić go od skrótu do środowiska dla C++.

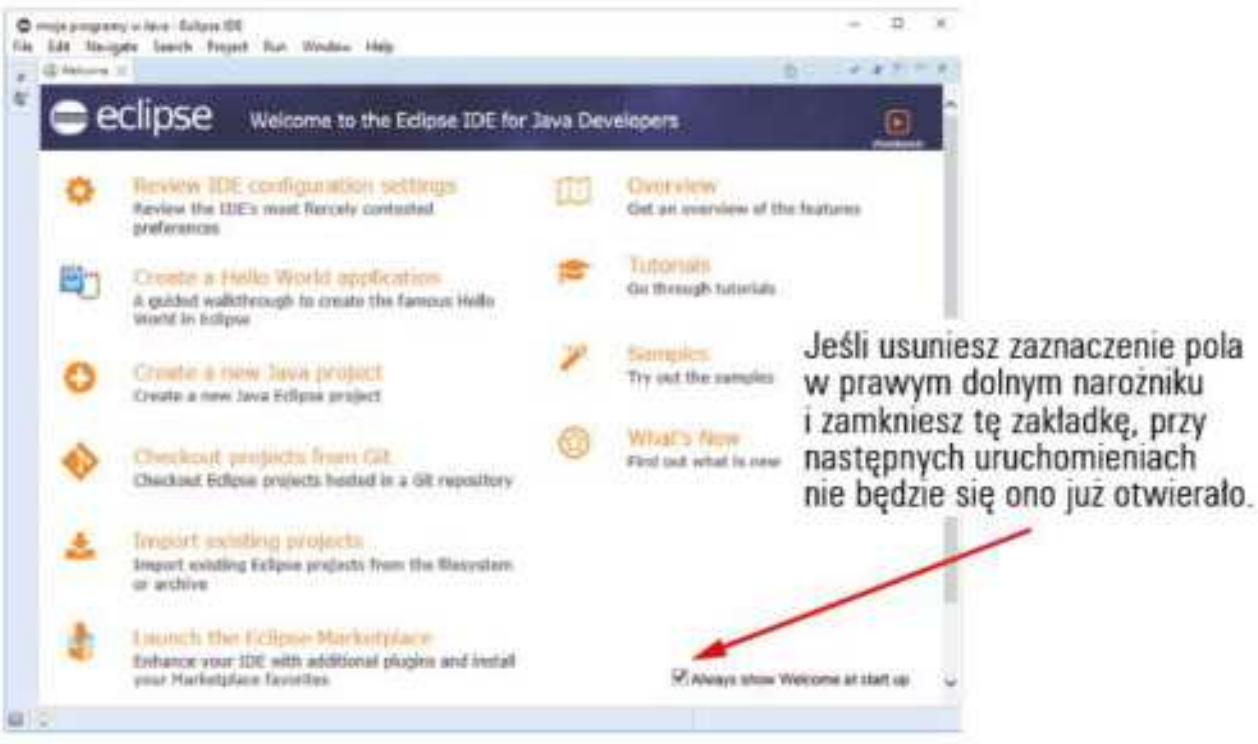
Podczas pierwszego uruchomienia IDE poprosi o umiejscowienie **foldera roboczego** (rys. 9.6.). Jeśli nie zdecydujesz inaczej, to w nim będą zapisywane wszystkie projekty tworzone przez ciebie.

IDE uruchomi się z oknem startowym (rys. 9.7.). W nim możesz wybrać na przykład dostęp do tutoriali, które mogą pomóc w użytkowaniu środowiska. Przejrzyj je, zanim rozpoczniesz używanie IDE. Jeśli usuniesz zaznaczenie pola

w prawym dolnym narożniku i zamkniesz tę zakładkę, przy następnych uruchomieniach nie będzie się ono już otwierało.

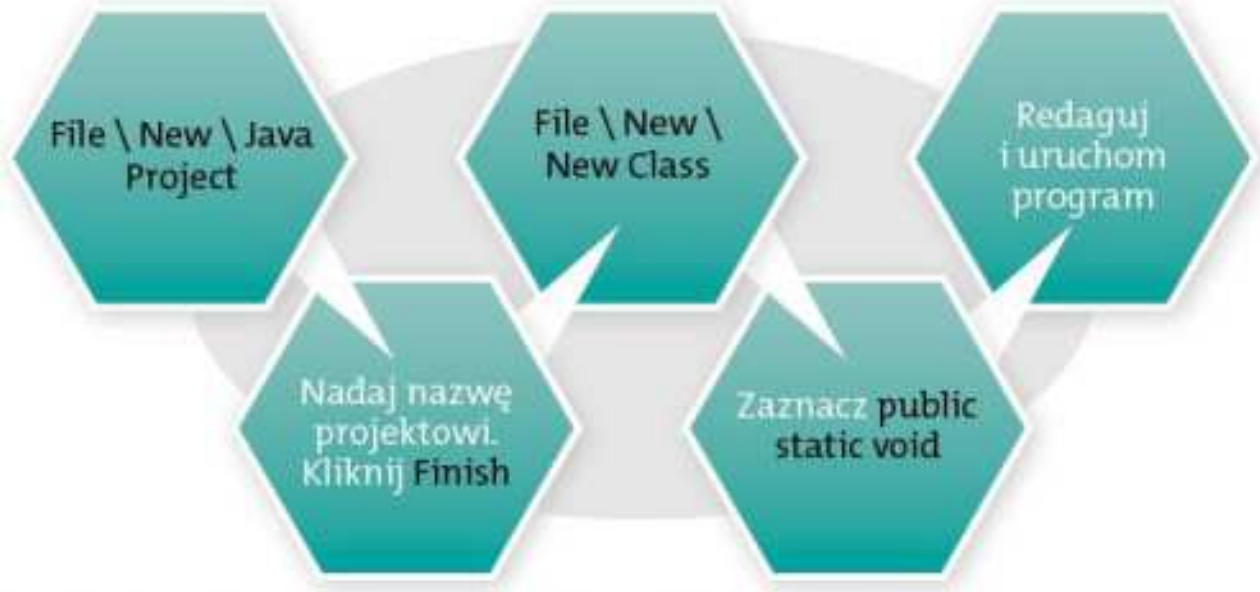



Rys. 9.6. Ustalenie foldera roboczego dla programów w Eclipse

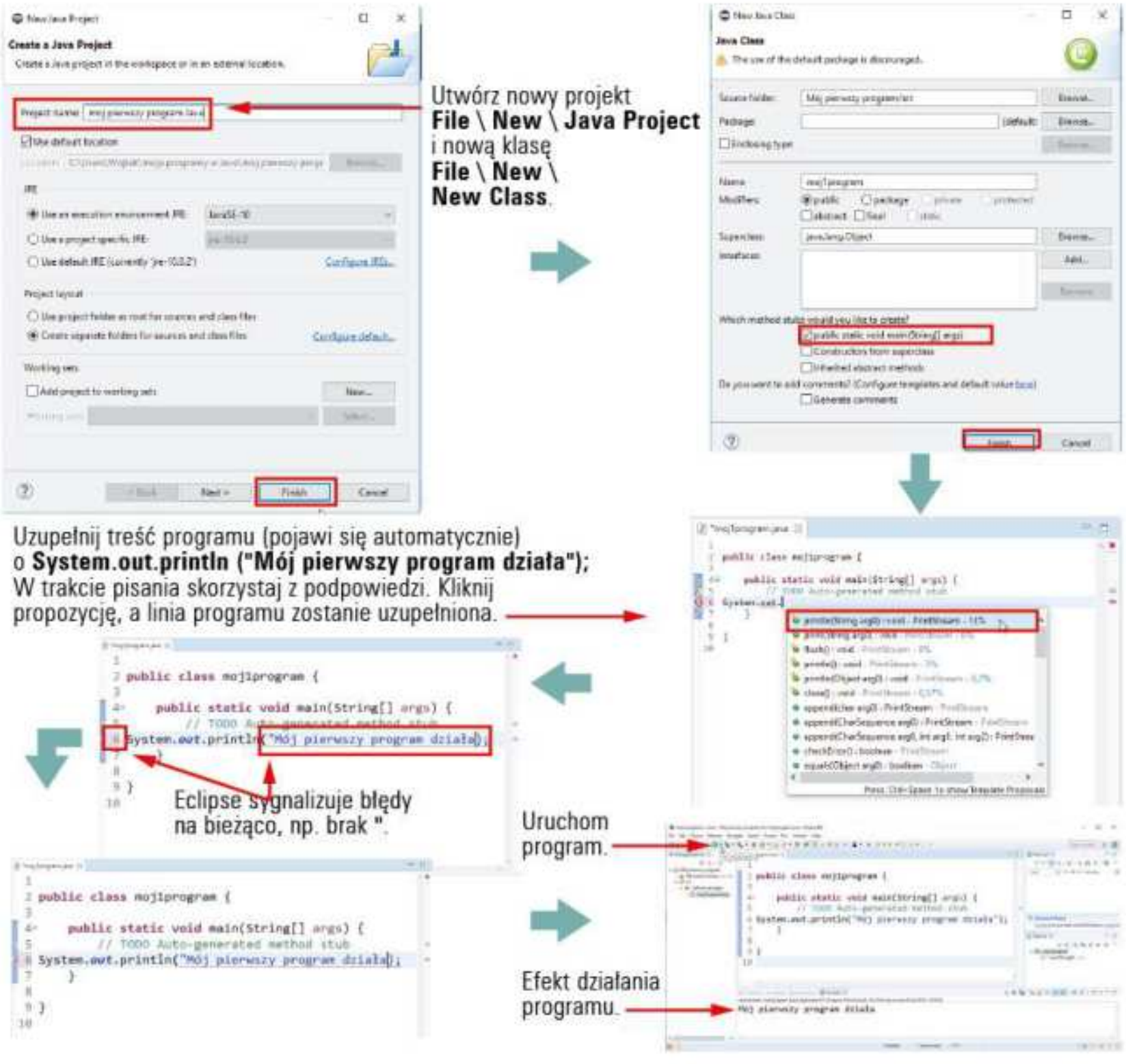



Rys. 9.7. Okno startowe Eclipse dla Java

IDE Eclipse jest już praktycznie gotowe do pracy. Czas sprawdzić, jak działa. Uruchom krótki program, który wyświetli na ekranie napis „Mój pierwszy program działa". Działaj według algorytmu (rys. 9.8.). Jeśli potrzebujesz więcej szczegółów, posłuż się opisem z rysunku 9.9.




Rys. 9.8. Proces tworzenia nowego projektu w Eclipse dla Java




Rys. 9.9. Proces tworzenia i uruchamiania programu ułożonego w Java w IDE Eclipse

Efekt działania programu jest prawidłowy. Poddajmy analizie ten prosty kod.

```
public class mojlprogram
{
  public static void main(String[] args)
    {
      System.out.println(„Mój pierwszy program działa”);
    }
}
```

Pierwsza linia kodu określa metodę rozruchową programu, od której zaczyna się wykonywanie wszystkich instrukcji.

```
public class mojlprogram
```

Publiczna klasa o nazwie `mojlprogram`.

`public` – specyfikator dostępu mówi o zasięgu klasy.

`class` – słowo kluczowe występujące przed nazwą danej klasy.

```
public static void main(String[] args)
```

Taki zapis oznacza metodę rozruchową programu. W tym przypadku to metoda statyczna, co oznacza, że wykorzystując tę metodę, nie odwołujesz się do metod niezadeklarowanych jako statyczne. Jeśli chcesz poznać cechy metod i dokładne znaczenie słów kluczowych `public` i `static`, to sięgnij do źródeł opisujących język Java.

`void` – funkcja ta nie zwraca żadnej wartości.

Zasadnicza część programu:

```
System.out.println(„Mój pierwszy program działa”);
```

drukuje ciąg znaków podany w cudzysłowie.

## 9.5. Plik zamiast File, czyli Eclipse po polsku

instalowanie polskiego menu w Eclipse

Wszystkie menu ułożono w języku angielskim. Jeśli chcesz zmienić język na polski, postępuj jak pokazano na rysunkach 9.10. i 9.11. Dokładną instrukcję znajdziesz pod adresem *www.eclipse.org/babel/downloads.php*.

**Uwaga!**

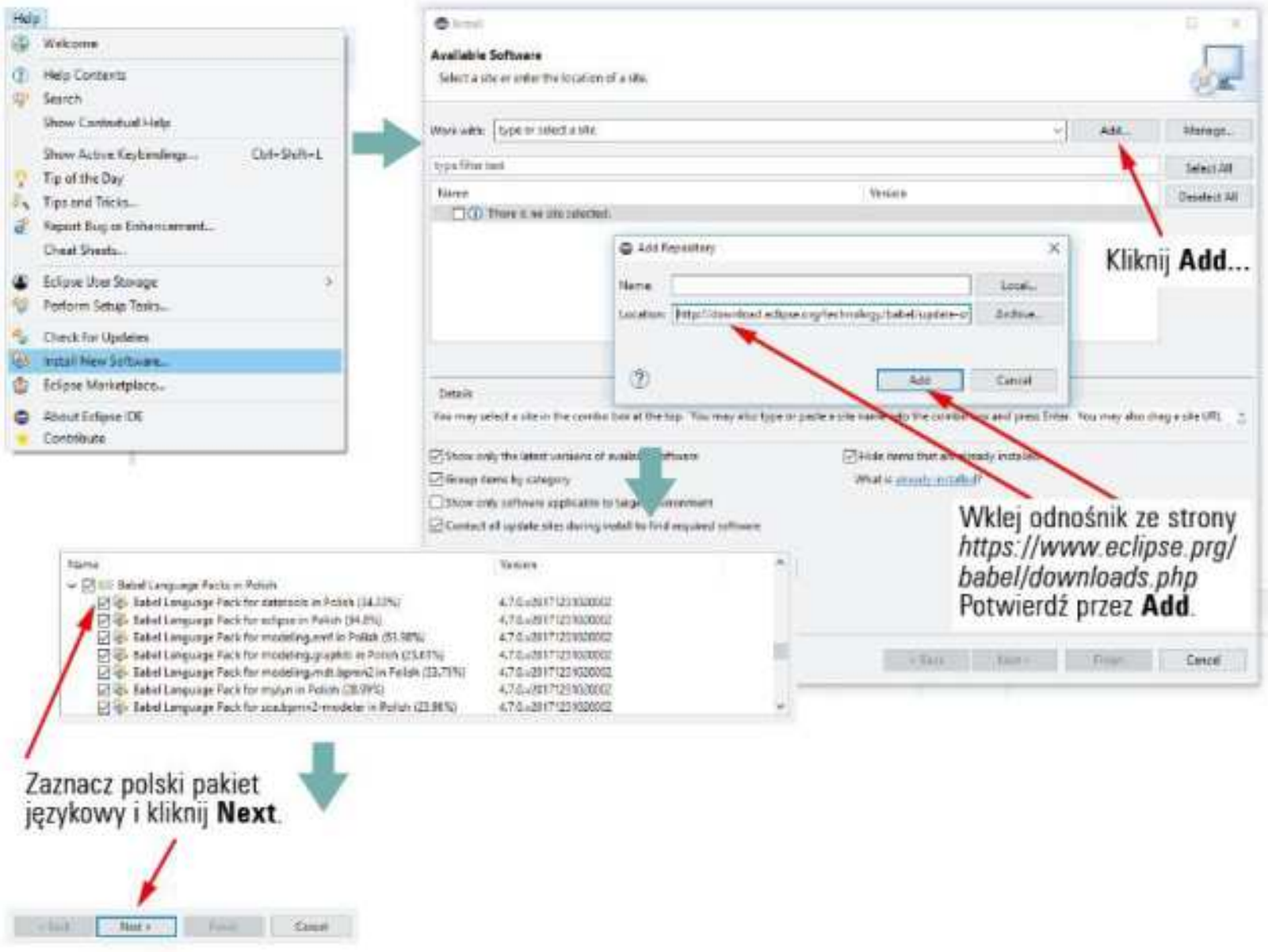
Język polski menu i okien dialogowych pomaga zrozumieć przeznaczenie poszczególnych opcji, jednak utrudnia późniejszą pracę i naukę programowania. Większość samouczków, instrukcji i przykładów opisuje Eclipse w wersji angielskiej, a jeśli zostaniesz programistą, także będziesz korzystać z wersji po angielsku.

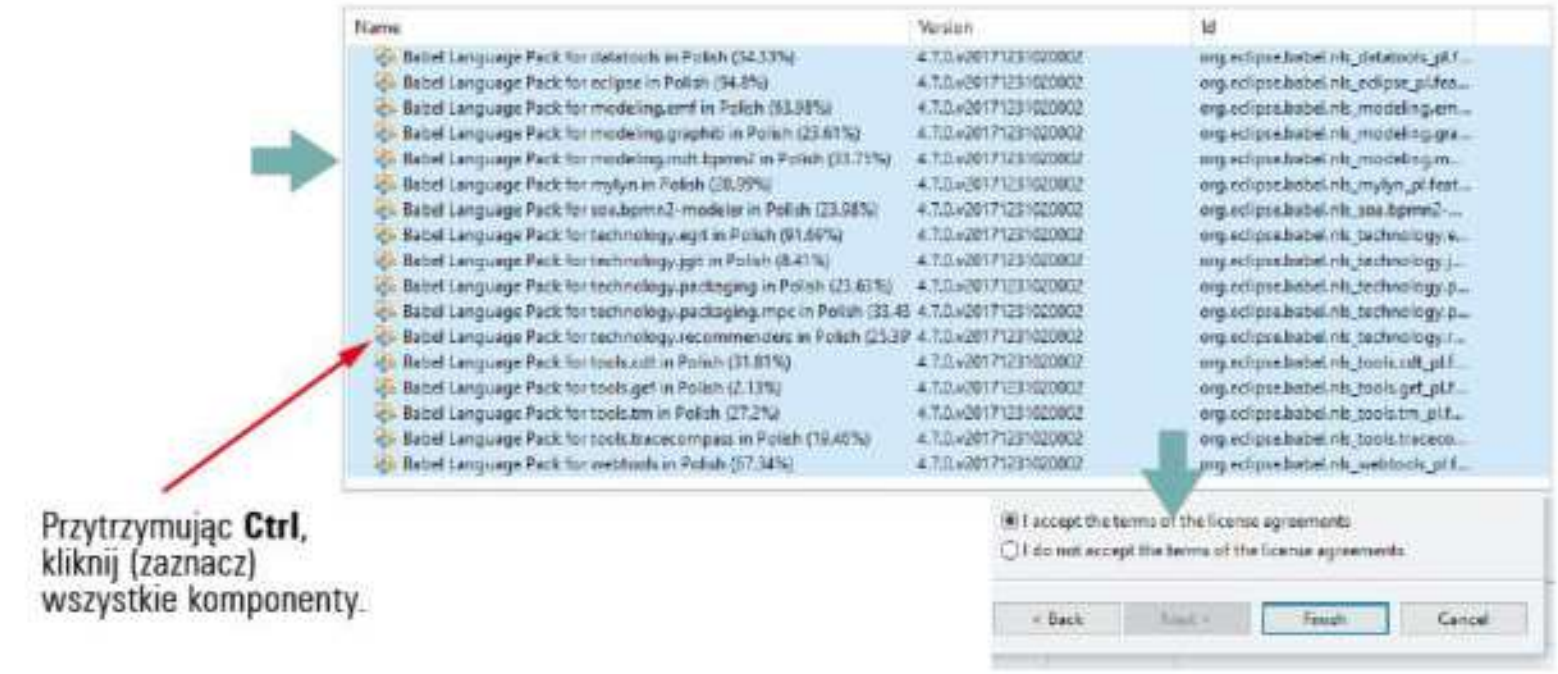



Rys. 9.10. Sekwencja czynności instalacji polskiego pakietu językowego dla Eclipse.

**Uwaga!**

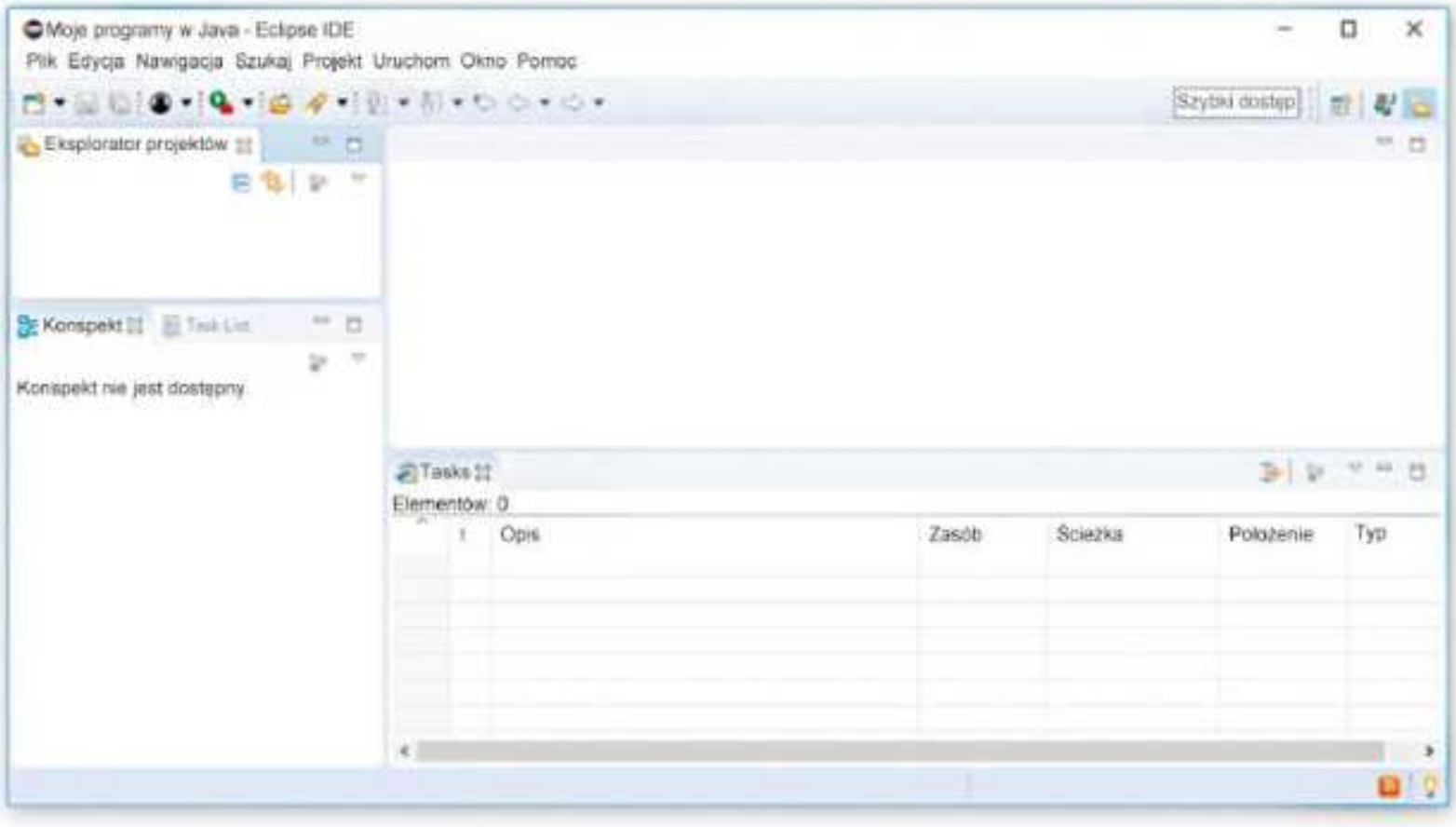
Jeśli nie zaznaczysz wszystkich komponentów polskiego pakietu, instalacja się nie powiedzie.






Rys. 9.11. Instalowanie polskiego pakietu językowego do Eclipse

Po wciśnięciu przycisku **Finish** nastąpi zamknięcie i ponowne uruchomienie Eclipse. Menu będą wyświetlane w języku polskim (rys. 9.12.).




Rys. 9.12. Okno IDE Eclipse z menu w języku polskim

**Uwaga!**

Jeśli chcesz korzystać z polskiego menu w obu instalacjach Eclipse – C++ i Java, w obu musisz wykonać te same czynności.

Jeśli Eclipse działa poprawnie, możesz rozpocząć naukę programowania w Java. Zanim jednak to nastąpi, prześledź następny rozdział opisujący proces konfigurowania i uruchamiania Eclipse dla C++.

## ZADANIA DO WYKONANIA

1. Przeczytaj jeszcze raz fragment rozdziału o instalacji polskiego pakietu językowego. W nowym folderze umieść zawartość pliku (pobranego w trakcie ćwiczenia z rozdziału) z IDE Eclipse dla Java. Stamtąd uruchom java.exe, a następnie zainstaluj polski pakiet językowy. Porównaj polskie menu z menu w języku angielskim. Otwórz dowolny tutorial i sprawdź, na ile język polski menu utrudnia jego zrozumienie.
2. Sprawdź na stronie TIOBE informacje na temat popularności i oceny języków programowania. Oprócz danych wydrukowanych w rozdziale znajdziesz tam wiele innych. Przeanalizuj je i wyciągnij wnioski, np. na temat trendów dotyczących języków programowania. Jeśli nie znasz dostatecznie języka angielskiego, skorzystaj z translatora, np. Google.
3. Znajdź w internecie fora dla początkujących programistów i wątki dotyczące porównania środowisk programistycznych. Poczytaj opinie na temat Eclipse i porównaj z opiniami o innych IDE. Przedstaw wnioski.

4*. Zapoznaj się z możliwościami użycia Notepad++ do edycji programów w Java. Dowiedz się, co jeszcze oprócz edytora będzie potrzebne do uruchomienia ułożonych w nim programów.

5*. Uruchom Eclipse dla Java. Z menu **Pomoc** wybierz opcję **Ściągawki**. Odnajdź temat „Utwórz aplikację Hello World". Korzystając z tej ściągawki, ułóż i uruchom program Hello World.

## PODSUMOWANIE LEKCJI

s. 60 ▸ • Aktualną listę najchętniej stosowanych i popularnych języków programowania znajdziesz na stronie *www.tiobe.com/tiobe-index/*.

s. 61 ▸ • Nie można jednoznacznie stwierdzić, który z **języków programowania** jest najlepszy. Przydatność danego języka zależy od rodzaju programu, jaki ma być w nim ułożony i w jakim urządzeniu lub systemie ma być uruchomiony.

s. 62 ▸ • **C++** jest językiem uniwersalnym. Dzięki swoim właściwościom, np. możliwości precyzyjnego wykorzystania sprzętu, pozwala na efektywne gospodarowanie zasobami komputera.

s. 62 ▸ • **IDE** (ang. Integral Development Environment) to program lub zespół programów służących do tworzenia, testowania, modyfikowania i konserwacji programów komputerowych w danym języku.

s. 62 ▸ • **Eclipse to IDE** zaprojektowane dla programistów posługujących się różnymi językami programowania w tym Java i C++. Eclipse zawiera system uzupełniania oraz podpowiadania poprawnej składni poleceń i rozkazów.

s. 67 ▸ • Jeśli chcesz w Eclipse zainstalować menu w języku polskim, użyj opcji **Install New Software**... z menu Help. Będzie także potrzebny odnośnik do pakietu językowego ze strony *https://www.eclipse.org/babel/downloads.php*.

# 10. C z plusami, czyli uruchamiamy pierwsze programy w C++

**NA TEJ LEKCJI:**

- dowiesz się, jak poprawnie skonfigurować Eclipse dla C++;
- dowiesz się, jak postawić pierwsze kroki w programowaniu C++.

Zainstalowanie i uruchomienie IDE jest tylko pierwszym krokiem do nauki programowania. Język to tylko narzędzie. Dokonaliśmy już wyboru obu elementów, czas na tworzenie programów. Nauka programowania trwa długo, a sukces w dużym stopniu zależy od twojego zaangażowania. Są jednak pewne zagadnienia, które powinien poznać każdy, zanim zacznie układać swoje pierwsze programy. Przypomnimy więc najważniejsze z nich. Przedtem przygotujemy środowisko IDE Eclipse dla języka C++.

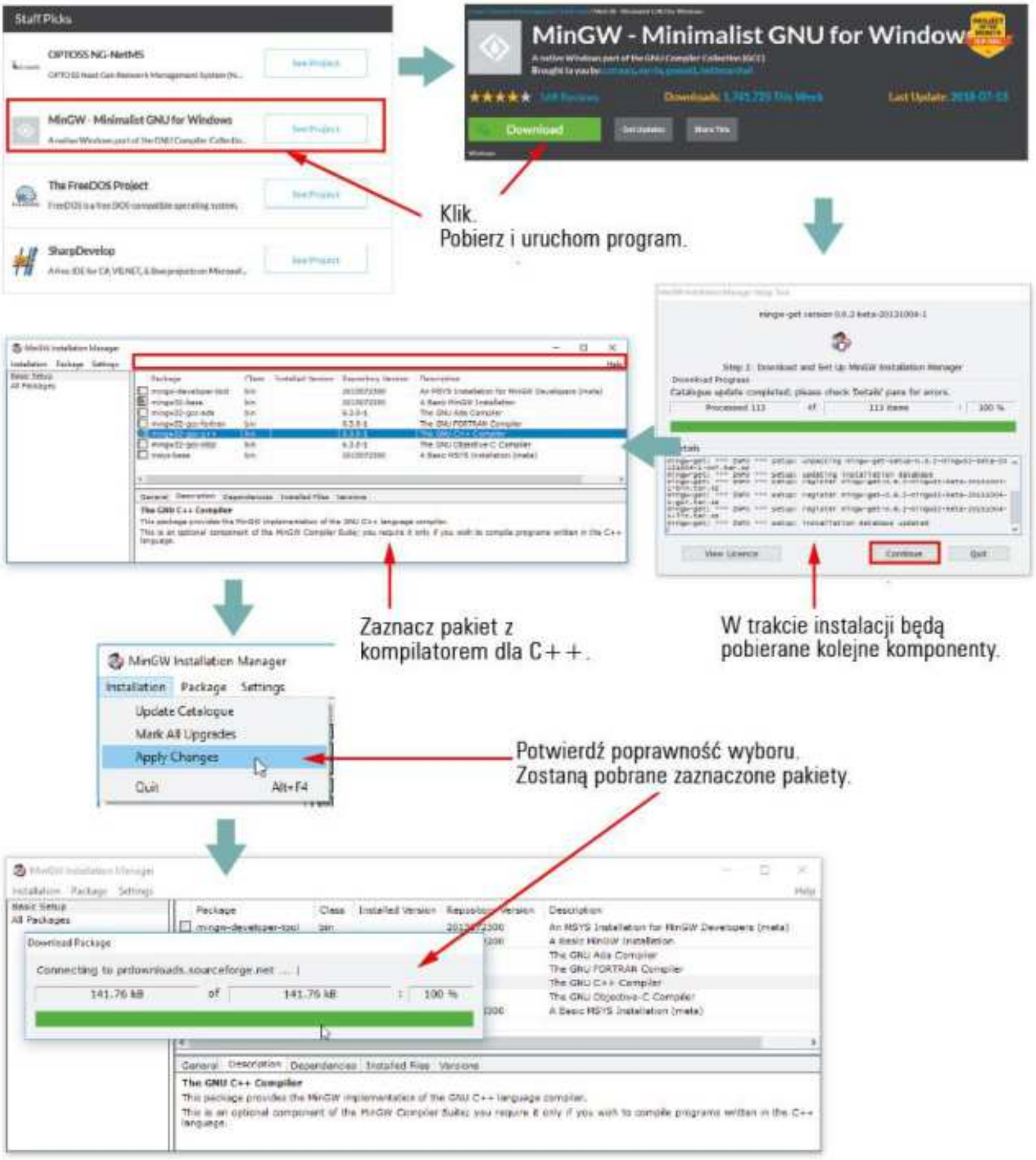
## 10.1. IDE dla C++, czyli czego jeszcze brakuje w Eclipse

◂ Eclipse a język C++

Eclipse dla Java i C++ z pozoru się nie różnią. Te same okna i bardzo podobne menu. Jednak C++ wymaga zainstalowania dodatkowego oprogramowania, które zawiera narzędzia pozwalające na kompilację (o której więcej na s. 78) programów (kodów źródłowych) przeznaczonych dla Windows. Powszechnie używanym dla C++ jest **MinGW** (Minimalist GNU for Windows). Szczegółowe informacje na temat tego darmowego środowiska znajdziesz w internecie.

◂ MinGW

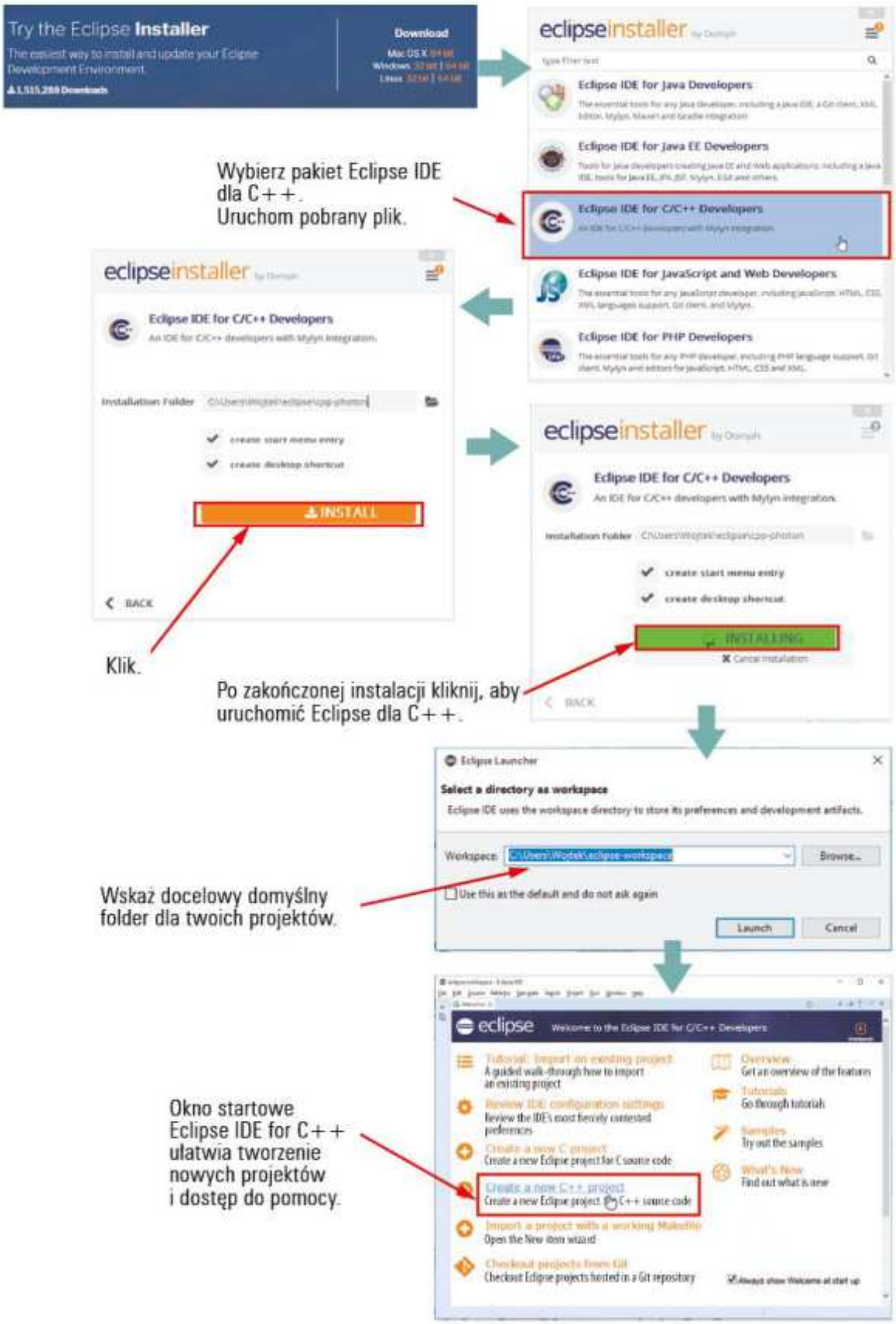
Zanim pobierzesz i uruchomisz Eclipse, przeprowadź instalację MinGW (rys. 10.1.). Odszukaj w sieci instalator tego środowiska. Możesz skorzystać ze strony *sourceforge.net*, na której znajdziesz wiele narzędzi typu Open-Source dla programistów.




Rys. 10.1. Instalacja pakietu MinGW

Po zainstalowaniu MinGW możesz już pobrać i uruchomić Eclipse dla C++. Proces wygląda podobnie jak w przypadku wersji dla Java.

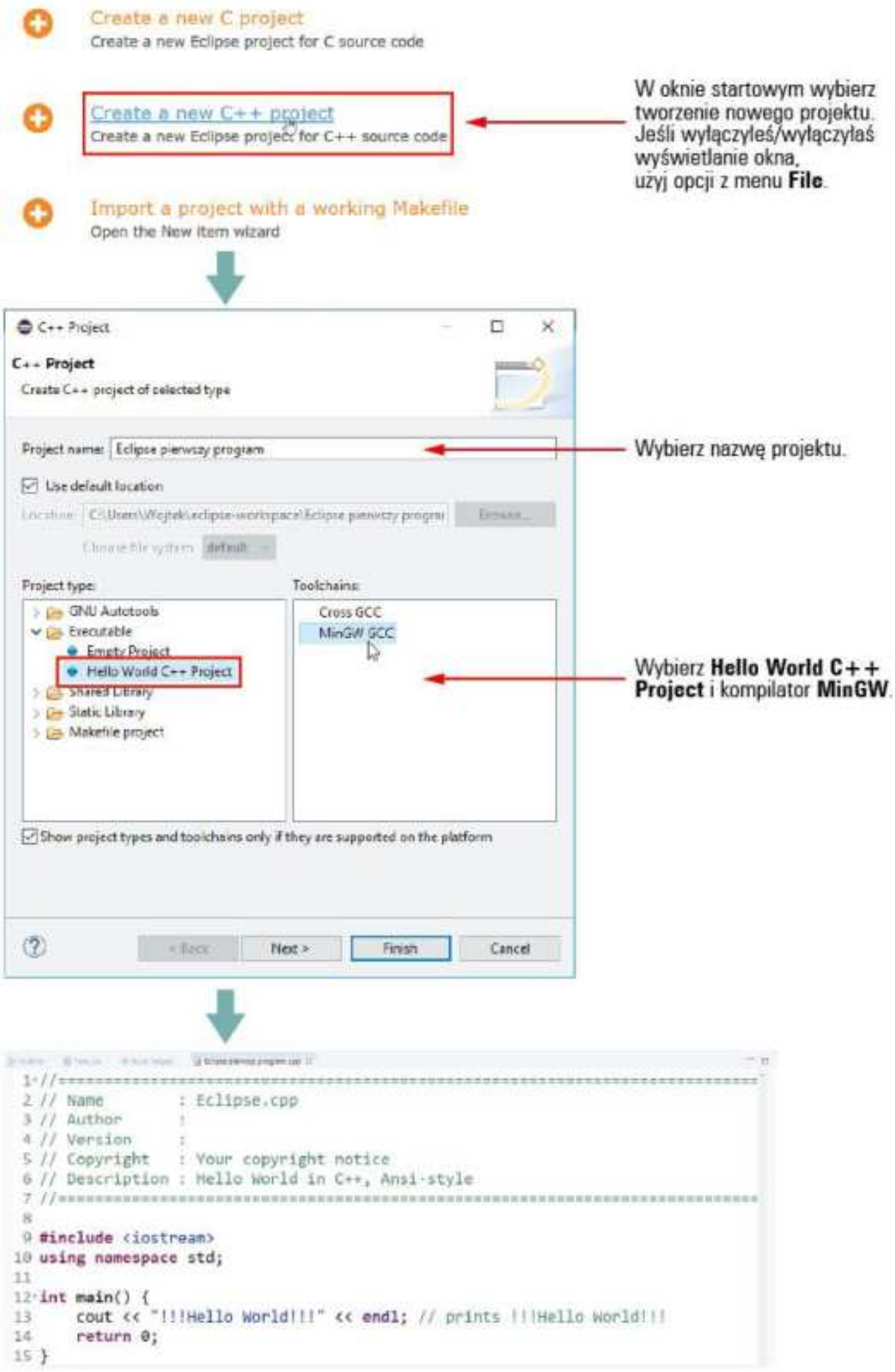
Rozpocznij od pobrania pliku instalacyjnego ze strony *http://www.eclipse.org/downloads/packages/installer* (rys. 10.2.).




Rys. 10.2. Pobieranie i instalacja Eclipse dla C++

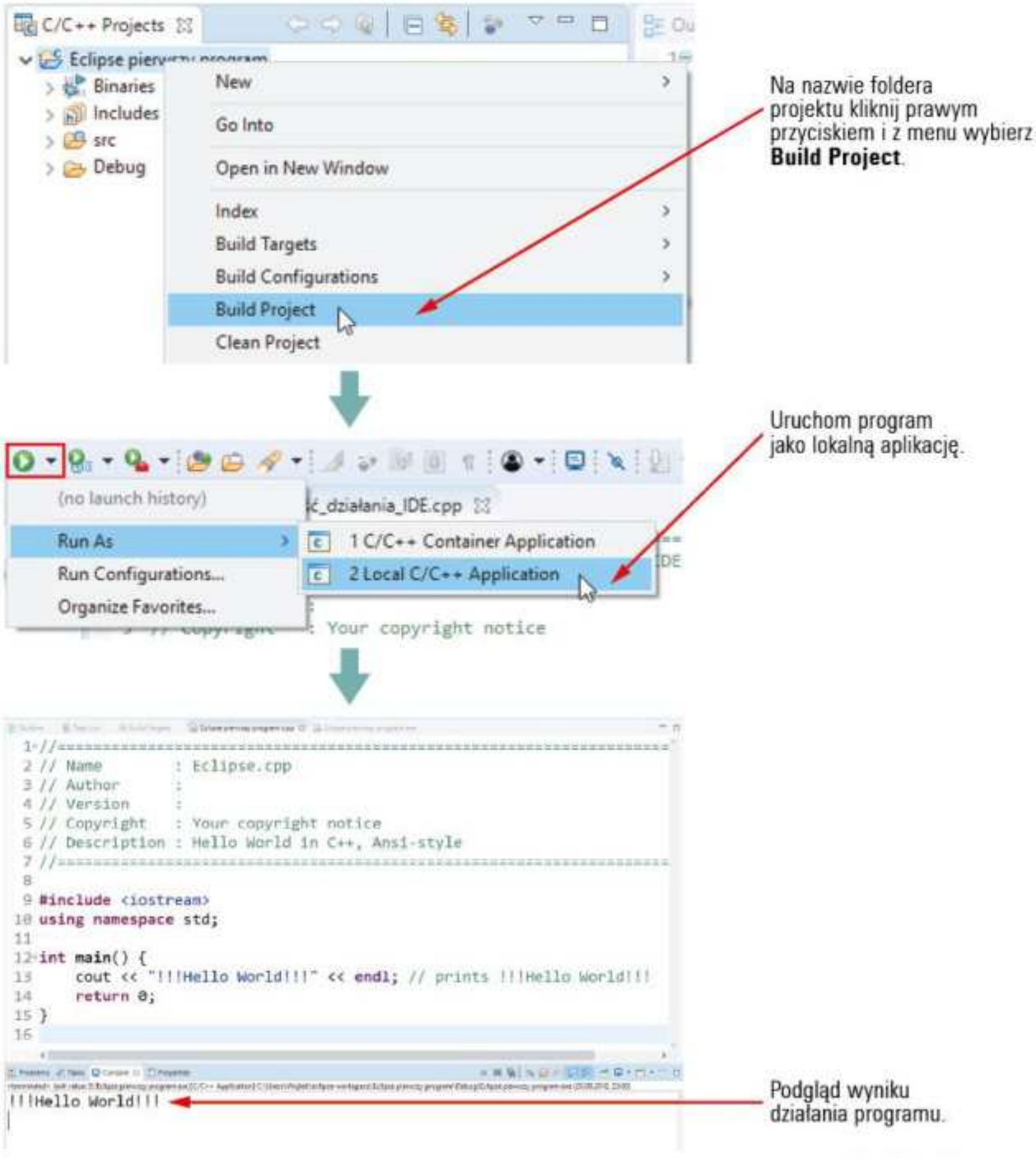
## 10.2. Zaczynamy, czyli pierwszy program w C++

Eclipse ułatwia stawianie pierwszych kroków w programowaniu w C++. Sprawdź to samodzielnie, wykonując czynności z rysunku 10.3.




Rys. 10.3. Tworzenie nowego projektu w Eclipse C++ z programem testowym

Twój projekt jest gotowy do uruchomienia. Napis !!!Hello World!!! możesz zmienić na dowolny. Aby sprawdzić działanie programu i przetestować IDE, uruchom ten program (rys. 10.4.).




Rys. 10.4. Uruchomienie przykładowego programu w Eclipse C++

◂ Eclipse C++

Wiesz już, że program działa. Świadczy to o tym, że prawidłowo zainstalowano IDE Eclipse. Warto jednak bliżej przyjrzeć się programowi (rys. 10.5.).

```
1 //============================================================================
2 // Name        : Eclipse.cpp
3 // Author      :
4 // Version     :
5 // Copyright   : Your copyright notice
6 // Description : Hello World in C++, Ansi-style
7 //============================================================================
8
9 #include <iostream>
10 using namespace std;
11
12 int main() {
13     cout << "!!!Hello World!!!" << endl; // prints !!!Hello World
14     return 0;
15 }
```

Informacje o autorze, właścicielu praw itp. bez znaczenia dla jego działania. Tekst po // jest komentarzem.

Rys. 10.5. Przykładowy program w Eclipse C++

```
#include <iostream>
```

znak # ▸ Znak # przed instrukcją oznacza, że jest ona dyrektywą preprocesora (programu, którego zadaniem jest przetworzenie kodu źródłowego). Wykonanie takiego polecenia następuje przed kompilacją. Pozwala to na dołączenie do programu plików nagłówkowych, czyli umożliwienie mu korzystania z funkcji danej biblioteki. Po dołączeniu `iostream` będzie możliwe wykorzystanie konsoli do wyświetlania wyniku działania programu, czyli skierowanie strumienia danych do konsoli systemu (na ekran).

```
using namespace std;
```

Oznacza informację dla kompilatora, że w programie zostanie użyta standardowa przestrzeń `std` (czyli miejsce, w którym trzymane są nazwy zmiennych, funkcji itp.) zawierająca deklaracje dostępu do całej biblioteki C++. Przy układaniu dużych programów mogą nastąpić konflikty nazw, czyli różne obiekty mogą mieć takie same nazwy. Można je umieszczać w różnych przestrzeniach i dzięki temu nie będą ze sobą kolidowały. Można to wytłumaczyć na przykładzie uczniów o tym samym nazwisku i imieniu. Jeśli należeliby do różnych klas, to ich przestrzenie są oddzielne i te osoby nie zostaną pomylone. W przestrzeni szkoły będą kolidowały i mogą prowadzić do pomyłek.

```
int main()
```

Nagłówek oznacza początek głównej funkcji programu. Słowo kluczowe `int` oznacza, iż system operacyjny będzie informowany o wyniku działania programu. Najczęściej nie jest to konieczne i w takim przypadku można użyć słowa kluczowego `void`. Na końcu funkcji z nagłówkiem **int** należy dopisać `return 0;`. Będzie to informacja dla systemu o zakończeniu operacji (przekazanie wartości 0).

```
cout << „!!!Hello World!!!" << endl;
```

To zasadnicza część programu, która powoduje wyświetlenie napisu. Ciąg znaków komunikatu należy umieścić w cudzysłowie.

**Słowo kluczowe** – słowo, które w danym języku programowania ma już przypisane znaczenie.

**Uwaga!**

Długie komentarze umieszcza się pomiędzy `/*` a `*/`.

Skoro uruchomiłeś/uruchomiłaś IDE, możesz rozpocząć programowanie. Przypomnij sobie podstawy z poprzednich lat nauki. Następny rozdział pomoże ci uporządkować tę wiedzę.

## ZADANIA DO WYKONANIA

1. Zmodyfikuj program opisany w rozdziale tak, by wyświetlał na ekranie inny tekst. Ustal doświadczalnie, jak wprowadzić poprawki, by tekst zajmował kilka linii. Przedstaw swoje rozwiązanie.
2. Po zainstalowaniu Eclipse dla C++ zdecyduj, czy będziesz korzystał z jego polskiej wersji językowej. Jeśli tak, zmień ją, opierając się na opisie wersji dla Java. Porównaj jakość spolszczenia obu wersji. Przedyskutuj, w jaki sposób można przywrócić wersję z językiem angielskim.
3. Ułóż algorytm opisujący czynności, które należy wykonać od momentu rozpoczęcia nowego projektu, aż do uruchomienia dowolnego programu w Eclipse dla C++.

4*. Uzupełnij algorytm z zadania 3. opisem prawidłowej struktury programu ułożonego w C++.

5*. Zmodyfikuj opisany wyżej program w taki sposób, by pierwsze linie oznaczone jako komentarz zawierały twoje dane. Jeśli uważasz, że inne informacje, np. nazwa szkoły, klasy, powinny się tam znaleźć, dodaj linie. Znajdź sposób na to, by nowe projekty automatycznie rozpoczynały się od twoich komentarzy.

## PODSUMOWANIE LEKCJI

- Środowisko programistyczne Eclipse może być wykorzystane do programowania w **języku C++**. Należy jednak pobrać i zainstalować odpowiednią wersję. Sposób posługiwania się podstawowymi opcjami Eclipse C++ jest taki sam, jak w przypadku środowiska **Eclipse dla Java**. ◂ s. 71
- Jednym z powszechnie stosowanych programów jest **MinGW**. Należy go zainstalować przed instalacją Eclipse dla C++. ◂ s. 71
- Do kompilacji **C++ w Eclipse** są potrzebne narzędzia pozwalające na kompilację programów w środowisku Windows. ◂ s. 75
- **Dyrektywy oznaczone #** są wykonywane przed kompilacją i powodują dołączenie odpowiednich bibliotek (zbiorów funkcji). ◂ s. 76

# 11. C z plusem, czyli podstawy programowania

**NA TEJ LEKCJI:**

- poznasz podstawową strukturę programu C++;
- dowiesz się, czym są operatory i instrukcje sterujące.

Twoje środowisko programistyczne jest już gotowe do pracy. Możesz zacząć programowanie. W tym rozdziale znajdziesz informacje o podstawowych zadaniach, jakie pełnią poszczególne polecenia i konstrukcje języka C++. W kolejnych rozdziałach poznasz ich działanie w praktyce. Tajniki C++ opanujesz w zakresie niezbędnym do ułożenia mało skomplikowanych programów. Poznanie wszystkich zagadnień związanych z programowaniem w tym języku zajmuje dużo więcej czasu i ćwiczeń niż jest to przewidziane na szkolne lekcje.

## 11.1. Co i gdzie, czyli podstawowa struktura programu w języku C++

Układając program w języku C++, programista tworzy **kod źródłowy** składający się z wyrażeń zrozumiałych dla człowieka. Kod ten po sprawdzeniu pod kątem

kompilacja ▸ poprawności składni jest **kompilowany**, czyli budowany jest program wykonywalny, w którego skład wchodzi tłumaczenie na kod maszynowy kodu programu wraz z użytymi bibliotekami oraz nagłówek powodujący, że program będzie mógł być uruchamiany w systemie operacyjnym bez udziału środowiska programistycznego. Te czynności wykonuje kompilator. Sprawdzaniem poprawno-

debugger ▸ ści składni zajmuje się program określany jako **debugger** (rys. 11.1.). Nazwano go tak, ponieważ błędy w programach nazywa się potocznie robakami – **bugami** (z j. ang.). Jak już wiesz, edytor, debbuger i kompilator składają się na środowisko programistyczne.



Projektant oprogramowania
– projektowanie sposobu rozwiązania problemu
– tworzenie algorytmów




Praca programisty w środowisku programistycznym
– edycja kodu źródłowego
– debuggowanie – sprawdzanie poprawności kodu źródłowego
– kompilacja – tłumaczenie na kod maszynowy



Programista we współpracy z projektantem
– testowanie – sprawdzanie poprawności działania programu
– optymalizacja – poszukiwanie lepszych rozwiązań wewnątrz programu
– ponowne testowanie

Rys. 11.1. Kolejne etapy tworzenia programu komputerowego

Język, którym mówimy, tworzą wyrazy połączone w zdania z zachowaniem zasad gramatyki. Podobnie jest z językiem programowania. Składa się on z poleceń, instrukcji (wyboru i warunkowych), nazw zmiennych, operatorów oraz bibliotek i funkcji.

Każdy program w C++ powinien mieć załączoną przynajmniej jedną bibliotekę, wyznaczoną przestrzeń nazw oraz przynajmniej jedną funkcję. Jeśli jest jedna, to powinna to być:

```
main();
```

Przykładowy program spełniający te warunki przedstawiono na rysunku 11.2.

Wszystkie wiersze zaczynające się od # to dyrektywy. Nie służą one do realizacji algorytmu. Informują jednak debugger i kompilator o dołączeniu do programu pliku zawierającego na przykład bibliotekę lub fragment kodu.

```
1 //============================================================================
2 // Name        : program_w_cpp.cpp
3 // Author      : Wojtek
4 // Version     :1.1
5 // Copyright   : Wojtek
6 // Description : struktura programu w C++
7 //============================================================================
8
9 #include <iostream>
10 using namespace std;
11
12 int main() {
13     cout << "Jak się masz?" << endl;
14     return 0;
15 }
```

Linia programu zaczynająca się // jest traktowana jako komentarz.

**#include** dołącza bibliotekę o nazwie podanej pomiędzy znakami **<** oraz **>**. **using namespace** ustala przestrzeń nazw.

Funkcja **main()** tworzy blok główny programu. Wszystko, co znajduje się pomiędzy **{}**, będzie wykonywane aż do napotkania instrukcji **return**.

Rys. 11.2. Przykładowa struktura programu w języku C++

`using namespace` informuje kompilator, że będzie używana dana przestrzeń nazw. Dla standardowych bibliotek C++ to `std`. Dzięki temu możesz napisać:

`cout <<"jak się masz" << endl;` zamiast `std::cout <<"jak się masz" << endl;`

W przypadku jednej linii nie ma to większego znaczenia, jednak w programie takich linii jest bardzo dużo. Domyślasz się więc, że definiowanie przestrzeni nazw może znacznie uprościć układanie programu.

## 11.2. Operatory, czyli działania i relacje

Czym są operatory? Dzięki nim jest możliwe zapisanie działań matematycznych (tab. 11.1.) i relacji (tab. 11.2.). ◂ operatory

Tab. 11.1. Operatory arytmetyczne w języku C++

| Działanie | Operator | Przykład |
|---|---|---|
| dodawanie | + | `y = x + 1` |
| odejmowanie | - | `y = x - 4` |
| mnożenie | * | `y = x * z` |
| dzielenie | / | `y = ( x + 25 ) /z` |
| reszta z dzielenia (modulo) | % | `y = z % 2` |

Tab. 11.2. Operatory relacyjne w języku C++

| Opis | Operator |
|---|---|
| mniejszy | < |
| większy | > |
| równy | == |
| różny | != |
| mniejszy lub równy | <= |
| większy lub równy | >= |

Przykładowo: jeśli napiszesz $y=(x-4)\ \%7$, będzie to oznaczało, że $y$ będzie równe reszcie z dzielenia $(x-4)$ przez 7. Wynik takiego działania jest zawsze liczbą całkowitą. Przykładowo: dla $x=20$ $y=(20-4)\ \%7=2$.

## 11.3. Zmienne, czyli przechowywanie danych

zmienne ▸ Definiując zmienną, nadajemy jej nazwę. Jeśli na przykład zdefiniujesz zmienną `komunikat = „wprowadź pierwszą liczbę";`, będzie to oznaczać, że pod nazwą `komunikat` kryje się tekst ujęty w cudzysłów. Jeśli danego tekstu będziesz używać wielokrotnie, możesz zapisać go w zmiennej. Będzie on także mógł być modyfikowany. W wielu programach i urządzeniach ten sam komunikat wyświetla się w różnych sytuacjach, na przykład „Dziękujemy i zapraszamy

string ▸ ponownie". Jeśli zadeklarujesz zmienną (dla tekstu – `string`) i nadasz jej taką wartość, to w kodzie programu nie musisz używać całego komunikatu w poleceniach wyświetlania. Na przykład:

```
//deklaracja zmiennej przechowującej tekst
string komunikat;
//nadanie wartości zmiennej
komunikat = „ Dziękujemy i zapraszamy ponownie „;
//wyświetlenie tekstu zawartego w zmiennej
cout << komunikat;
```

Jeżeli wyświetlany komunikat będzie długi, to oceń, jaki jest zysk dla programisty. Łatwo też na przykład zmienić język komunikatów. Wystarczy jedynie zmienić wartość zmiennej.

Można zmodyfikować program z rysunku 11.2. w taki sposób, by strumień (więcej o strumieniach na s. 87) z napisem zawierał wartość zmiennej `komunikat`.

```
1 //========================================
2 // Name        : string.cpp
3 // Author      : Wojtek
4 // Version     :1.0
5 // Copyright   : Wojtek
6 // Description : struktura programu w C++
7 //========================================
8
9 #include <iostream>
10 using namespace std;
11 string komunikat="jak się masz?";
12 int main() {
13
14     cout << komunikat << endl;
15     return 0;
16 }
```

Zadeklarowano zmienną typu **string**, nadając jej wartość.

Ta linia nie wydrukuje słowa „komunikat", lecz to, co kryje się pod zmienną **komunikat**.

Rys. 11.3. Przykład zastosowania zmiennej typu string

Tab. 11.3. Typy danych liczbowych lub logicznych, które można przechowywać w zmiennych

| Nazwa typu | Liczba bajtów | Zakres wartości |
|---|---|---|
| `bool` | 1 | false lub true (fałsz lub prawda; 0 lub 1) |
| `char` | 1 | od –128 do 127 |
| `unsigned char` | 1 | od 0 do 255 |
| `wchar_t` | 2 | od 0 do 65 535 |
| `short` | 2 | od –32 768 do 32 767 |
| `unsigned short` | 2 | od 0 do 65'535 |
| `int` | 4 | od –2 147 483 648 do 2 147 483 647 |
| `unsigned int` | 4 | od 0 do 4 294 967 295 |
| `long` | 4 | od –2 147 483 648 do 2 147 483 647 |
| `unsigned long` | 4 | od 0 do 4 294 967 295 |
| `long long` | 8 | od –9 223 372 036 854 775 808 do 9 223 372 036 854 775 807 |
| `unsigned long long` | 8 | od 0 do 18 446 744 073 709 551 615 |
| `float` | 4 | 3.4E +/– 38 (7 cyfr) |
| `double` | 8 | 1.7E +/– 308 (15 cyfr) |
| `long double` | 8 | 1.7E +/– 308 (15 cyfr) |

Pamiętasz zapewne, że zmienne mogą przechowywać różne typy danych. Dlaczego jest ich wiele? Otóż zdefiniowanie zmiennej rezerwuje dla niej miejsce w pamięci. Jeśli więc wiemy, że nasza zmienna przyjmuje wartości od 0 do 255, wystarczy zarezerwować dla niej jedynie bajt. Jeśli jednak zmienna przyjmuje wartość całkowitą dodatnią w bardzo dużym zakresie w C++, można zarezerwować jej aż 8 bajtów i największą liczbą, jaką można wpisać pod zmienną, będzie 18 446 744 073 709 551 615.

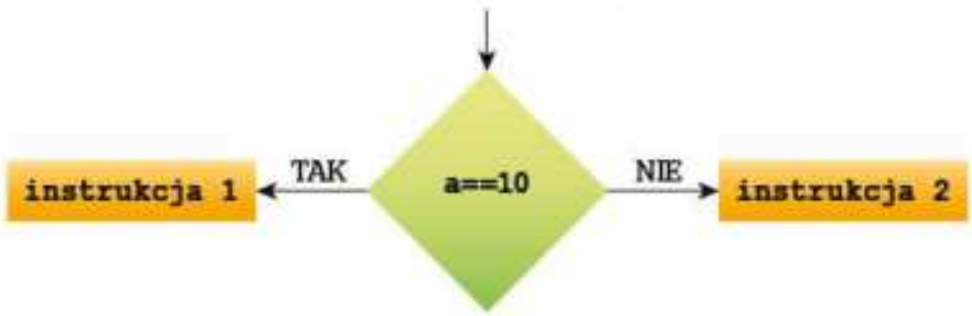
Wykaz wszystkich typów danych liczbowych, dla których można zadeklarować zmienną, znajdziesz w tabeli 11.3.

Jak myślisz, ile bajtów zajmie liczba 25 wpisana do zmiennej zadeklarowanej jako `long`? Oczywiście tego typu zmienna zawsze zajmie w pamięci 4 bajty, bez względu na to, jaką wartość z jej zakresu przyjmie.

## 11.4. Instrukcje sterujące, czyli realizacja bloków warunkowych algorytmu

W każdym algorytmie, który wymaga rozgałęzień, stosuje się bloki decyzyjne (rys. 11.4.). Jeśli w przedstawionym przykładzie warunek zostanie spełniony, to jest realizowana `instrukcja 1`, jeśli nie – `instrukcja 2`.

Do zapisania warunków w języku C++ używa się operatorów logicznych i symboli porównawczych (tab. 11.4.).




Rys. 11.4. Przykład bloku decyzyjnego

Tab. 11.4. Operatory logiczne i symbole porównawcze w języku C++

| Operator | Symbol porównawczy | Działanie |
|---|---|---|
| && | | koniunkcja (I) |
| \|\| | | alternatywa (LUB) |
| ! | | negacja (NIE) |
| | == | równy |
| | != | różny |
| | > | większy |
| | < | mniejszy |
| | >= | większy lub równy |
| | <= | mniejszy lub równy |

Dzięki blokowi decyzyjnemu można rozgałęziać algorytmy i, w zależności od spełnienia lub niespełnienia warunku, podejmować odpowiednie działania. W językach programowania takie bloki zapisuje się na przykład za pomocą instrukcji warunkowej. W wielu językach programowania wygląda to bardzo podobnie. W C++ ma ona postać jak na rysunku 11.5.

pętle lub instrukcje warunkowe

```
//Postać pierwsza instrukcji warunkowej

if (warunek)
{ instrukcja 1; // jeśli warunek będzie spełniony,
  instrukcja 2; // zostaną wykonane instrukcje w { },
}               // a jeśli nie, zostaną one pominięte

//Postać druga instrukcji warunkowej

if (warunek)
{ instrukcja 1; //Jeśli warunek będzie spełniony,
  instrukcja 2; //zostaną wykonane instrukcje 1 i 2,
}               //a pominięte instrukcje 3 i 4
else
{instrukcja 3;  //W przeciwnym przypadku zostaną
 instrukcja 4;  //pominięte instrukcje 1 i 2, a wykonane
}               //3 i 4
```

Rys. 11.5. Instrukcje warunkowe w C++

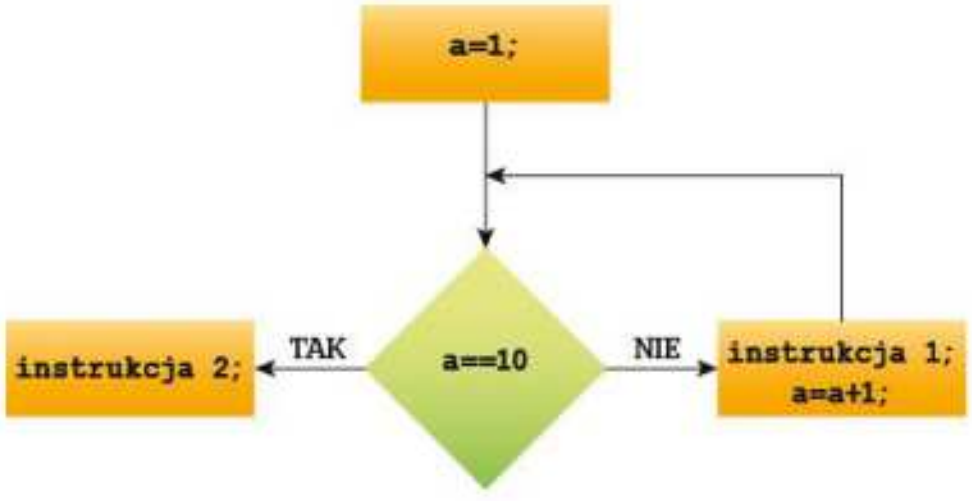
Jeśli uczyłeś/uczyłaś się programowania w Scratch, to zapewne pamiętasz badanie warunków (rys. 11.6.).

Użycie odpowiedniej postaci instrukcji warunkowej zależy od algorytmu, który należy zrealizować.

Jeśli algorytm zawiera zapętlenie (rys. 11.7.), realizowane dopóki spełniony jest warunek, można użyć **instrukcji pętli**.




Rys. 11.6. Instrukcje warunkowe Scratch




Rys. 11.7. Przykład pętli w algorytmie

Poznaj dwie postaci pętli stosowane w języku C++. Pierwsza z nich (rys. 11.8.) jest wykonywana do momentu, gdy warunek zostanie spełniony, a w drugiej (rys. 11.9.) warunek zostaje sprawdzony po wykonaniu wszystkich instrukcji. Oznacza to, że w drugim przypadku wszystkie polecenia i instrukcje pętli zostaną wykonane przynajmniej raz i powtarzane tak długo, aż warunek zostanie spełniony.

```
while (a==10)
{
instrukcja 1;   //Instrukcja 1 i 2 wykonywana dopóki
instrukcja 2    //jest spełniony warunek
}               //W przypadku jednej instrukcji {} nie są
                //wymagane
```

Rys. 11.8. Przykład pętli while w języku C++

```
do
{
instrukcja 1;   //Instrukcje 1 i 2 wykonywane
instrukcja 2;   //przynajmniej raz i powtarzane
}               //
while (a==10); //dopóki warunek jest spełniony
//W przypadku jednej  instrukcji {} nie są wymagane
```

Rys. 11.9. Przykład pętli do ... while w języku C++

Fragment algorytmu z rysunku 11.7. można także zrealizować za pomocą pętli `for` (rys. 11.10.).

```
for (i=1; i<10; i++) //Począwszy od i=1, dopóki
{                    // i<10,wykonuj instrukcje w { }
instrukcja 1;        // za każdym razem zwiększając
instrukcja 2;        // i o 1
}
```

Rys. 11.10. Instrukcja sterująca for w języku C++

Zauważ, że instrukcja z algorytmu `i = i + 1;` jest realizowana jako parametr instrukcji sterującej for (`i++`). Czym więc różni się w tym przypadku konstrukcja `while` od `for`? For jest wykonywana pewną ustaloną liczbę razy. W przykładzie dla `a` od 1 do 10 pętla `for` zostanie wykonana 9 razy. Przy `a` równym 10 nastąpi wyjście z pętli (rys. 11.7.). W przypadku pętli `while` lub `do while` liczba wykonywanych pętli zależy od spełnienia warunku. W przypadku pętli `do while` warunek jest sprawdzany na końcu, po wykonaniu instrukcji. Dlatego ta pętla zawsze będzie wykonana chociaż raz.

Pętle mogą się zagnieżdżać, czyli występować wewnątrz innych pętli.

## ZADANIA DO WYKONANIA

1. Podaj na przykładzie różnicę pomiędzy instrukcją warunkową `if` a `if else`.
2. Określ, ile razy zostanie zrealizowana pętla for, jeśli będzie miała postać:

```
for (a=0; a<=10; a++)
{
cout <<"pętla nr "<<a <<endl;
}
```

   Porównaj tę pętlę z użytą w przykładzie (rys. 11.10.).

```
for (a=1; a<10; a++)
```

   Czy ich działanie w programie będzie takie samo, czy różne?
3. Na dowolnych przykładach wskaż różnice pomiędzy pętlą `while` a `do while`.
4*. Narysuj w zeszycie algorytm dla fragmentu programu ułożonego w języku C++. Wzoruj się na przykładach z rozdziału.

```
{
  a=0;
  do
  {
  cout << "informatyka" << endl;
  a=a+1;
  }
  while (a!=10);
return 0;
}
```

5*. Ile razy w konsoli z zadania 4. zostałby wydrukowany napis „informatyka"? Uzasadnij swoją odpowiedź.

## PODSUMOWANIE LEKCJI

- Języki programowania mają ściśle ustaloną składnię sprawdzaną w trakcie tworzenia kodu przez **debugger**. ◂ s. 78
- **Kompilacja** polega na zamianie kodu źródłowego programu na język zrozumiały przez komputer. Nazywamy go kodem maszynowym. ◂ s. 78
- **Działania arytmetyczne i logiczne** są zapisywane w języku C++ za pomocą operatorów. ◂ s. 79
- **Zmienne** mogą być deklarowane dla różnych typów danych. Dla różnych typów są rezerwowane w pamięci różne ilości bajtów. ◂ s. 80
- Tekst zapisuje się w zmiennych typu **string**. ◂ s. 80
- Do realizacji bloków decyzyjnych w algorytmach w zależności od tego, co realizuje algorytm, można użyć **pętli lub instrukcji warunkowych**. ◂ s. 83

# 12. C z plusem, czyli podstaw programowania ciąg dalszy

**NA TEJ LEKCJI:**

- poznasz zastosowanie i znaczenie funkcji oraz tablic i strumieni.

Znasz już zasady tworzenia algorytmów i niektóre elementy języka programowania. Przed tobą zadania programistyczne, do których niezbędne będzie poszerzenie tej wiedzy i zdobycie nowych umiejętności. Poznasz zastosowanie oraz znaczenie funkcji, tablic i strumieni. Po zdobyciu tej wiedzy przystąpisz do tworzenia własnych programów.

## 12.1. Tablice bez kredy, czyli jedna nazwa, wiele zmiennych

tablice ▸ Wiesz, że przed użyciem zmiennej musisz ją zadeklarować, podając typ wartości, które będzie przechowywać, i oryginalną nazwę. Istnieją jednak dane związane ze sobą, na przykład kolejne wyniki losowania lub cogodzinne pomiary temperatury. Deklarowanie oddzielnej zmiennej dla każdej z nich nie byłoby wygodne i znacznie utrudniałoby układanie programów.

**Przykład 1.**

Lista uczniów klasy zawiera 30 nazwisk. Każde ma w dzienniku przydzielony swój numer. Jeśli zmienną nazwalibyśmy przykładowo: `klasala`, to zmienna `klasala[5]` zawierałaby nazwisko spod numeru `5` z dziennika.

Zmiennym przewidzianym dla wielu danych można więc nadać jedną nazwę. Sposób deklaracji jest podobny do deklarowania pozostałych zmiennych.

**Przykład 2.**

Zmienne dla pojedynczej wartości:

```
float temperatura;
```

Jeśli pomiar temperatury ma się odbywać co godzinę w ciągu jednej doby, zadeklarujemy zmienną tablicową (macierz) jednowymiarową.

```
float temperatura[23];
```

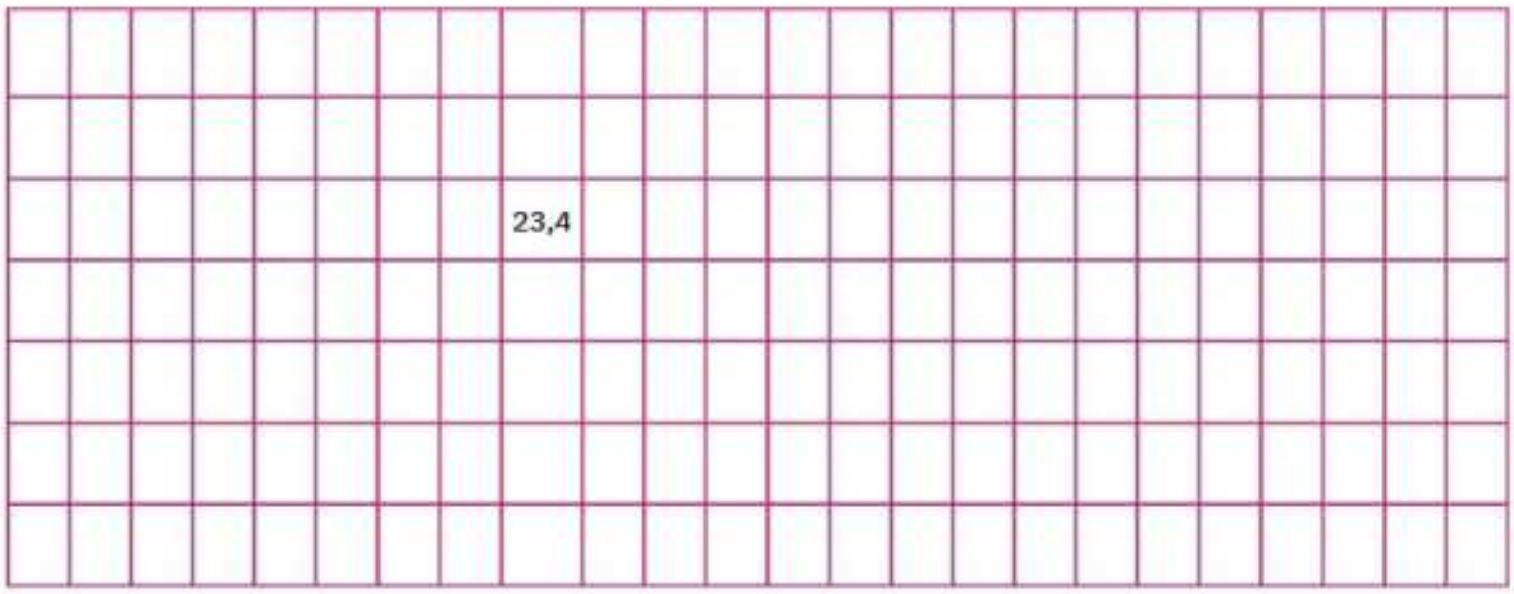
Zapewne zastanawiasz się, dlaczego dla 24 pomiarów w deklaracji tablicy użyto liczby 23. Nie ma tu błędu, ponieważ indeksacja (numeracja) poszczególnych elementów tablicy zaczyna się od 0. Zatem pierwszy pomiar zostanie zapisany pod `temperatura[0]`, a na przykład 6 pod `temperatura[5]`.

A co, jeśli takie pomiary będzie trzeba wykonać przez 7 dni? Wtedy można użyć macierzy dwuwymiarowej, którą można porównać do rozmieszczenia krzeseł w sali kinowej. Jeśli znasz rząd i numer krzesła, trafisz na swoje miejsce.

**Przykład 3.**

Tablica przechowująca wyniki pomiarów temperatury co godzinę przez 7 dni:

```
float temperatura[6][23];
```




Rys. 12.1. Przykład danych wpisanych do zmiennej tablicowej

Domyślasz się, że możliwe jest deklarowanie tablic o wielu wymiarach. Jeśli będą ci potrzebne, z pewnością poradzisz sobie z ich deklaracją i używaniem.

◂ tablice wielowymiarowe

## 12.2. Strumienie, czyli wczytujemy i wyprowadzamy informacje

W programach zdarza się, że dane muszą być wprowadzone przez użytkownika, a wynik, na przykład obliczeń, ma zostać wyświetlony na ekranie. Jak można te czynności zrealizować w języku C++? W tym celu wykorzystuje się strumieniowanie, czyli kierowanie strumienia danych w odpowiednim kierunku. Do ich określenia wykorzystuje się słowa z języka angielskiego `out` wskazujące na zewnątrz oraz `in` – do środka. Jeśli dodasz do nich na początku `c` kojarzące się z nazwą języka C++, otrzymasz nazwy strumieni: wejściowego `cin` i wyjściowego `cout`. Dodatkowo kierunki przepływu strumienia wskazują operatory `<<` dla strumienia na zewnątrz (wyprowadzanie danych) oraz `>>` dla strumienia do wewnątrz (wczytywanie danych). Wczytywanie i wyprowadzanie danych może dotyczyć zmiennych różnego typu.

◂ strumienie

**Przykład 4.**

Układamy program do mnożenia trzech liczb całkowitych. Przeanalizuj fragment kodu wprowadzania danych i strumieniowania wyniku (rys. 12.2.). Znaki, które mają pojawić się na ekranie, należy umieścić w cudzysłowie (na rysunku

niebieskie). Wartości zmiennych są reprezentowane przez ich nazwy (na rysunku czarne). Słowo `endl` oznacza koniec linii, a więc następny strumień danych będzie wyświetlany w nowej linii. Zwróć uwagę, że poszczególne strumieniowane dane są poprzedzone operatorem kierunku.

```
cout << "wprowadź liczbę a " << endl;
cin >> a;
cout << "wprowadź liczbę b " << endl;
cin >> b;
cout << "wprowadź liczbę c " << endl;
cin >> c;
```

Rys. 12.2. Fragment programu strumieniujący dane

Zastanów się i odpowiedz, co będzie efektem działania tego fragmentu kodu programu. Czy twoje przypuszczenia są zgodne z rysunkiem 12.3.?

```
wprowadź liczbę a
12
wprowadź liczbę b
13
wprowadź liczbę c
15
```

Rys. 12.3. Efekt działania fragmentu kodu programu wprowadzającego wartości do trzech zmiennych

## 12.3. Nie tylko zmienne mają nazwy, czyli budowa i rola funkcji w programie

funkcja ▸ Jeśli takie same działania są wykonywane w wielu miejscach programu, można utworzyć funkcję i do niej się odwoływać. Po pierwsze nie będzie trzeba wklejać tego samego kodu w każdym miejscu, w którym ma być wykonany. Po drugie kod staje się krótszy. Po trzecie wszelkie zmiany i poprawki są wykonywane jedynie w kodzie funkcji. Funkcja może przetwarzać argumenty – dane przekazane przez program. Niektóre funkcje nie potrzebują takich argumentów, na przykład funkcja drukująca stałe komunikaty. Dzięki funkcjom można więc wydzielić w kodzie bloki odpowiedzialne za poszczególne zadania.

Cechy funkcji:

- ma nazwę, przez którą jest wywoływana;
- może przetwarzać argumenty wejściowe, jeśli tak przewidziano w programie;
- może zwracać przetworzone dane (wyniki działania funkcji).

**Przykład 5.**

W pewnym programie wielokrotnie używa się obliczania średniej z trzech liczb *a*, *b* oraz *c*. Opracowano więc funkcję, która dokonuje obliczeń. Przed nazwą funkcji podano typ wartości, jaką funkcja ma zwrócić. W tym przypadku `float`,

ponieważ średnia może nie być całkowita. Tworzenie funkcji rozpoczyna się podobnie jak deklarowanie zmiennej tablicowej jednowymiarowej. W miejscu `[]` pojawił się jednak `()`. Kod funkcji zwany **ciałem funkcji** zaczyna się i kończy nawiasem klamrowym `{}`.

Zmienne utworzone w funkcji to zmienne tymczasowe. Oznacza to, że po zakończeniu działania funkcji nie pamiętają już swojego stanu; za każdym wywołaniem funkcji wartości są im nadawane na nowo. Przypomnij sobie, na czym polega wykorzystanie kalkulatora. Jeśli mnożysz liczby, to kolejno je wprowadzasz i po wciśnięciu „=" otrzymujesz wynik.

Podobnie przekazywane do funkcji są parametry. Przeanalizuj poniższy przykład (rys. 12.4.).

```
1  #include <iostream>
2  using namespace std;
3  int a,b,c;
4
5  int iloczyn (int d, int e, int f)
6  {
7      return (d*e*f);
8  }
9
10 int main()
11 {
12     cout << "wprowadź liczbę a " << endl;
13     cin >> a;
14     cout << "wprowadź liczbę b " << endl;
15     cin >> b;
16     cout << "wprowadź liczbę c " << endl;
17     cin >> c;
18         cout <<endl<<a<<"*"<<b<<"*"<<c<<"="<<iloczyn(a,b,c)<<endl
19 return 0;
20 }
```

Definicja zmiennych globalnych

Ciało funkcji z określeniem parametrów funkcji

Główna funkcja programu

Rys. 12.4. Przykład zastosowania funkcji ze zmiennymi lokalnymi

W ciele funkcji zadeklarowano parametry lokalne `d`, `e` oraz `f` i określono, że dane w nich przechowywane będą typu `int` (liczby całkowite). Funkcja zwróci iloczyn `d*e*f`. To oznacza, że nazwa funkcji użyta w programie przyjmie taką wartość. Należy jednak przekazać jej trzy argumenty. Mogą to być zmienne lub konkretne liczby. Program (rys. 12.4.) prosi o wprowadzenie trzech liczb i zapisuje je w zmiennych `a`, `b` oraz `c`. Następnie drukuje ich iloczyn. Oblicza go, wywołując funkcję i przekazując jej wprowadzone dane. Wewnątrz funkcji parametr `d` przyjmuje aktualną wartość zmiennej `a`, i analogicznie: `e` przyjmuje wartość `b` oraz `f` wartość `c`.

◂ zmienne

Zauważ, że ten rozdział nie wyczerpuje zagadnień związanych z budową i wykorzystywaniem funkcji. Są one opisane w literaturze dla programistów. Jednak zdobyta wiedza wystarczy ci do układania prostych programów z użyciem funkcji.

## 12.4. Zagnieżdżanie instrukcji, czyli pętla w pętli

instrukcje warunkowe

Znasz już instrukcje warunkowe i pętle. W niektórych programach istnieje konieczność zagnieżdżenia pętli w pętli.

**Przykład 6.**

instrukcja pętli for

Twój program ma utworzyć tablicę dwuwymiarową `tablica[4][3]` wypełnioną zerami. Potrzebujesz dwóch zmiennych całkowitych do sterowania pętlami. Należy wykorzystać instrukcję pętli for zagnieżdżoną w pętli for.

```
1  #include <iostream>
2  using namespace std;
3
4  int i,j;
5  int tablica[4][3];
6
7  int main()
8  {
9  for(i=0;i<5;i++)
10 for(j=0;j<4;j++)
11 {
12 tablica[i][j]=0;
13 cout <<"tab ["<<i<<"]["<<j<<"] = "<< tablica[i][j]<<endl;
14 }
15 }
```

Pętla zagnieżdżona

Rys. 12.5. Przykładowy program wypełniający tablicę wartością 0

```
tab [0][0] = 0
tab [0][1] = 0
tab [0][2] = 0
tab [0][3] = 0
tab [1][0] = 0
tab [1][1] = 0
tab [1][2] = 0
tab [1][3] = 0
tab [2][0] = 0
tab [2][1] = 0
tab [2][2] = 0
tab [2][3] = 0
tab [3][0] = 0
tab [3][1] = 0
tab [3][2] = 0
tab [3][3] = 0
tab [4][0] = 0
tab [4][1] = 0
tab [4][2] = 0
tab [4][3] = 0
```

Wartości zmiennych *i* oraz *j* są indeksami zmiennej tablicowej. Zmieniają się zgodnie zasadami działania pętli for, dlatego przyjmą wszystkie możliwe wartości indeksów zmiennej.

Na tej samej zasadzie można zagnieżdżać instrukcje warunkowe.

Rys. 12.6. Efekt działania programu wypełniającego tablicę

## ZADANIA DO WYKONANIA

1. Naszkicuj w zeszycie, jak będą wyglądały tablice trójwymiarowa i czterowymiarowa. Udowodnij, że twoje rysunki rzeczywiście są modelami takich tablic.
2. Zmodyfikuj program z rysunku 12.4. w taki sposób, by obliczał sumę kwadratów wprowadzonych liczb. Zweryfikuj wynik działania programu.
3. Zmodyfikuj program wpisujący zera do tablicy dwuwymiarowej w taki sposób, by zamiast zer wpisywał kolejne numery kolumn tablicy. Uzasadnij swoje rozwiązanie.
4. Co będzie wynikiem działania programu z rysunku 12.5., gdy zamieni się miejscami w pętlach zmienne pomocnicze *i* oraz *j*. Opracuj wnioski i udowodnij swoją tezę.

5*. Na podstawie programu z rysunku 12.4. ułóż program, który na wprowadzonych liczbach a, b oraz c, będzie wykonywał mnożenie, dzielenie lub dodawanie – w zależności od decyzji użytkownika. Uzasadnij użycie funkcji w tym programie.

## PODSUMOWANIE LEKCJI

- **Tablice** to zmienne przechowujące więcej niż jedną wartość. Każda z nich jest opisana indeksem. ◂ s. 86
- Istnieją **tablice wielowymiarowe**. ◂ s. 87
- Strumieniowanie danych **cin i cout** umożliwia wprowadzanie i wyprowadzanie wartości zmiennych (**cout** umożliwia także wyprowadzanie znaków, np. komunikatów). ◂ s. 87
- Funkcje są fragmentami kodu programu mającymi nazwę. **Wynik działania funkcji** jest zwracany pod jej nazwą. ◂ s. 88
- **Zmienne** deklarowane w funkcjach są zmiennymi lokalnymi i nie zachowują wartości po zakończeniu działania funkcji. ◂ s. 89
- Do funkcji można przekazać dane (**argumenty**). ◂ s. 89
- **Pętle for** mogą być zagnieżdżone. ◂ s. 90
- Zagnieżdżać można również **instrukcje warunkowe**. ◂ s. 90

## PODSUMOWANIE PRZED SPRAWDZIANEM

### 6. Przypomnij sobie, czyli podstawy tworzenia algorytmów

s. 47
- **Algorytm** to skończony i uporządkowany opis precyzyjnie sformułowanych czynności prowadzący do przetworzenia danych wejściowych na wyjściowe.

s. 47
- **Najczęściej używane sposoby zapisu algorytmu** to: lista kroków, schemat blokowy, pseudokod.

s. 47
- Specyfikacja algorytmu składa się najczęściej z trzech części: **opisu problemu, opisu danych wejściowych i opisu danych wyjściowych.**

s. 48
- Do **cech dobrego algorytmu** należą: **precyzyjne zdefiniowanie, skończoność, efektywność.**

### 7. Algorytmy. Utrwalamy swoją wiedzę

s. 52
- **Algorytm** składa się z szeregu kroków, które mają nas doprowadzić do rozwiązania.

s. 53
- **Algorytmy możemy zapisać** w postaci np. listy kroków, schematu blokowego lub pseudokodu.

s. 53
- Najbardziej popularnym zapisem jest **schemat blokowy**, który ułatwia w późniejszym etapie kodowania napisanie właściwego programu.

### 8. Pozycyjne reprezentacje liczb, czyli tablice w algorytmice

s. 57
- **Liczby doskonałe** to takie liczby, które są sumą wszystkich swoich dzielników właściwych.

s. 57
- **Dzielnik właściwy** jest zawsze mniejszy od dzielnej.

s. 57
- **Euklides** jako pierwszy wskazał, że liczby doskonałe to liczby parzyste.

s. 57
- **Leonard Euler** znalazł sposób na wyszukanie liczby doskonałej dzięki liczbom pierwszym.

s. 58
- Do zamiany liczb z dowolnego systemu liczbowego na system dziesiętny wykorzystujemy **schemat Hornera** wartości wielomianu.

s. 58
- **Tablice** są wykorzystywane przy tworzeniu algorytmów do zamiany systemów liczbowych na system dziesiętny.

s. 58
- Tablice mogą mieć rozmiar z góry ustalony (**tablice statyczne**) lub można ten rozmiar zmieniać w trakcie wykonywania programu (**tablice dynamiczne**).

### 9. Narzędzia, czyli jak dobrać i skonfigurować środowisko programistyczne

s. 60
- **Dane o popularności** języków programowania można znaleźć na stronie TIOBE.

s. 62
- **C++** jest językiem uniwersalnym. Dzięki swoim właściwościom pozwala na efektywne gospodarowanie zasobami komputera.

s. 62
- **Eclipse to IDE** zaprojektowane dla programistów posługujących się różnymi językami programowania.

### 10. C z plusami, czyli uruchamiamy pierwsze programy w C++

s. 71
- **MinGW** (Minimalist GNU for Windows) jest niezbędne do kompilowania programów w C++ w Eclipse. Pobieranie i instalacja wymaga precyzji działania.

s. 75
- Tworzenie nowego projektu w Eclipse C++ rozpocznij od wybrania z menu **File** opcji **New**, a następnie **Create a new C++ project**. Pamiętaj o zaznaczeniu kompilatora MinGW.

s. 76
- **Dyrektywy** oznaczone # są wykonywane przed kompilacją i powodują dołączenie odpowiednich bibliotek (zbiorów funkcji). Bez **bibliotek** nie ma programów w C++.

### 11. C z plusem, czyli podstawy programowania

s. 78
- **Kompilacja** polega na zamianie kodu źródłowego programu na język zrozumiały przez komputer.

s. 79
- **Operatory** arytmetyczne, relacyjne i logiczne pozwalają na zapisywanie działań matematycznych i logicznych.

s. 80
- **Zmienne** mogą być deklarowane dla różnych typów danych.

s. 83
- **Instrukcje sterujące** pozwalają na realizację rozgałęzień (bloków warunkowych) algorytmów.

### 12. C z plusem, czyli podstaw programowania ciąg dalszy

s. 86
- **Zmienne tablicowe** pozwalają przechowywać uporządkowane dane.

s. 88
- **Funkcje** są fragmentami kodu programu mającymi nazwę. Mogą być wywoływane w programie dzięki użyciu ich nazw. Można do nich przekazywać wartości zmiennych.

s. 90
- **Pętle i instrukcje warunkowe** można zagnieżdżać.

# III. Programowanie i algorytmy

- Sito Eratostenesa
- Liczby pierwsze
- Program na podstawie algorytmu
- Algorytmy Euklidesa
- NWW i NWD w programie komputerowym
- Dodawanie ułamków
- Szyfrowanie
- Tablica kodów ASCII

# 13. Wspólna miara, czyli podstawy złożoności obliczeniowej

**NA TEJ LEKCJI:**

- dowiesz się, na czym polega czasowa złożoność obliczeniowa;
- dowiesz się, czym jest notacja omikron oraz operacja dominująca.

Złożoność obliczeniowa, a dokładniej teoria złożoności obliczeniowej jest działem teorii obliczeń. Dzieli się on na teorię obliczalności, czyli odpowiada na pytanie, które problemy dają się rozwiązać za pomocą komputera, i złożoność obliczeniową, czyli mówi, jak szybko da to się zrobić. Rozważanymi zasobami są standardowo dwie wielkości – czas i pamięć, ale może również pojawić się liczba procesorów potrzebna do obliczeń.

## 13.1. Czasowa złożoność obliczeniowa (złożoność czasowa)

czasowa złożoność obliczeniowa

Przez czasową złożoność obliczeniową będziemy rozumieli ilość czasu potrzebnego do rozwiązania problemu w zależności od tego, ile mamy danych wejściowych. Możemy zatem opisać ją jako pewną funkcję liczby danych wejściowych:

$T(n) = f(n)$

Na przykład: $T(n) = 8n^2 - 12 + 3$

operacja dominująca

Czasową złożoność obliczeniową wyrażamy najczęściej w liczbie operacji dominujących, które należy wykonać dla $n$ danych, aby otrzymać rozwiązanie. Operacja dominująca jest czynnością bezpośrednio wpływającą na czas wykonywania całego algorytmu. Najczęściej złożoność czasową wyrażamy w liczbie operacji dominujących. Jeżeli chodzi o podawanie złożoności czasowej w jednostkach czasu, jest ono niewygodne, ponieważ wyniki mogą się od siebie znacząco różnić. Wynik może w tym przypadku zależeć od szybkości danego komputera, na którym dokonano pomiarów. Trudno w takim wypadku porównać te dane z wynikami z innych komputerów z powodu różnic. Podając wyniki w liczbie operacji dominujących, uniezależniamy je od szybkości komputerów.

## 13.2. Złożoność pamięciowa

złożoność pamięciowa

Złożoność pamięciowa ma za zadanie określić liczbę komórek pamięci, która będzie zajęta przez dane i wyniki pośrednie tworzone w trakcie pracy algorytmu.

Ponieważ często zużycie zasobów w algorytmie zależy od postaci przetwarzanych, obie złożoności – czasowa i pamięciowa – mogą występować w trzech odmianach:

1. $T_O(n)$ – optymistycznej
2. $T_A(n)$ – średniej
3. $T_W(n)$ – pesymistycznej

Aby wyjaśnić powyższe terminy, przeanalizujemy algorytm wyszukiwania robaczywej śliwki w koszyku.

*Dopóki w koszyku są śliwki, wyjmij jedną, obejrzyj ją, jeśli jest robaczywa, to zakończ. W przeciwnym razie odłóż ją do drugiego koszyka i wróć do początku.*

Rozważmy, ile operacji dominujących (ocena jakości śliwki) wykona ten algorytm dla $n$ śliwek.

1. Przypadek optymistyczny – wyciągnięcie robaczywej śliwki za pierwszym razem – $T_o(n) = 1$ (złożoność optymistyczna).
2. Przypadek najgorszy – w koszyku nie ma robaczywych śliwek lub robaczywa śliwka zostanie wyciągnięta jako ostatnia – $T_w(n) = n$ (złożoność pesymistyczna).
3. W tym przypadku robaczywą śliwkę wyciągniemy z koszyka po przejrzeniu połowy owoców – może być ona wyciągnięta raz wcześniej, raz później, ale średnio wyjdzie nam, że przejrzymy połowę $n/2$ – $T_A(n) = n/2$ (złożoność średnia).

Złożoność optymistyczna określa zużycie zasobów dla najkorzystniejszego zestawu danych, średnia – dla typowych danych, a złożoność pesymistyczna – dla zasobów najbardziej niekorzystnych.

## 13.3. Określenie czasowej złożoności obliczeniowej algorytmu

Wykonajmy algorytm obliczania sumy kolejnych liczb naturalnych, który obliczy złożoność czasową. Dla każdej operacji zdefiniujmy czasy wykonania, które oznaczymy literą $t$ z odpowiednim indeksem. Aby otrzymać wzór na czasową złożoność obliczeniową tego algorytmu, po jego wykonaniu będziemy musieli zsumować czasy, które przypisaliśmy każdej operacji.

| Krok | Operacja | Czas wykonania |
|---|---|---|
| Krok 1 | Wczytaj $n$ | $1*t_1$ |
| Krok 2 | suma = 0 | $1*t_2$ |
| Krok 3 | $i = 1$ | $1*t_3$ |
| Krok 4 | Jeśli $i > n$, to idź do Kroku 8 | $(n+1)*t_{4a}$ dla jeśli..., $1*t_{4b}$ dla idź do... |
| Krok 5 | suma = suma + $i$ | $n * t_5$ |
| Krok 6 | $i = i + 1$ | $n * t_6$ |
| Krok 7 | Idź do Kroku 4 | $n * t_{4b}$ |
| Krok 8 | Wypisz sumę | $1 * t_8$ |
| Krok 9 | Zakończ | $1 * t_9$ |

Sumujemy czasy wykonania poszczególnych operacji i otrzymujemy w ten sposób złożoność czasową. W Kroku 4 są dwa czasy – jeden dotyczy operacji *jeśli*, która jest wykonywana zawsze o jeden raz więcej niż wynosi $n$, a drugi czas dotyczy operacji *idź do*, która jest wykonywana tylko jeden raz przy zakończeniu pętli.

$$T(n) = t_1 + 2 * t_2 + (n + 1) * t_{4a} + t_{4b} + n * t_5 + n * t_6 + n * t_{4b} + t_8 + t_9$$

$$T(n) = t_1 + 2 * t_2 + (n + 1) * t_{4a} + (n + 1) * t_{4b} + n * (t_5 + t_6) + t_8 + t_9$$

$$T(n) = (n + 1) * (t_{4a} + t_{4b}) + n * (t_5 + t_6) + t_1 + 2 * t_2 + t_8 + t_9$$

Określmy teraz następujące stałe, które odnoszą się do czasów wykonania odpowiednich grup.

$$t_a = t_{4a} + t_{4b}$$

$$t_b = t_5 + t_6$$

$$t_c = t_1 + 2 * t_2 + t_8 + t_9$$

Otrzymamy w ten sposób wzór na czasową złożoność tego algorytmu:

$$T(n) = t_a * (n + 1) + t_b * n + t_c$$

$$T(n) = t_a * n + t_a + t_b * n + t_c$$

$$T(n) = (t_a + t_b) * n + t_a + t_c$$

Wzór oczywiście możemy uprościć, wprowadzając kolejne stałe:

$$t_{ab} = t_a + t_b$$

$$t_{ac} = t_a + t_c$$

Po wprowadzeniu tych dodatkowych stałych otrzymamy następujący wzór:

$$T(n) = t_{ab} * n + t_{ac}$$

Z otrzymanego wzoru widzimy, że czas wykonania algorytmu sumującego liczby naturalne jest liniowo zależny od ilości sumowanych liczb. Jeśli $n$ wzrośnie dwa razy, to czas wykonania algorytmu też powinien w przybliżeniu wzrosnąć dwukrotnie. Tak przedstawiona złożoność obliczeniowa ma wymiar jednostek czasu, które zostały użyte do pomiaru czasu wykonania poszczególnych kroków algorytmu na konkretnym komputerze.

## 13.4. Klasy złożoności

notacja omikron

Przy analizie algorytmów korzysta się z tzw. klas złożoności obliczeniowej, które określają rząd funkcji $T(n)$. Jednym ze sposobów określenia rzędu tej funkcji jest popularna notacja omikron – zwana także notacją dużego O.

Mówimy, że $T(n) = O(f(n))$ (funkcja złożoności obliczeniowej $T(n)$ jest rzędu funkcji $f(n)$), jeśli potrafimy znaleźć takie $n_0 \in N$ oraz takie $c \in R$, że dla każdego $n \geq n_0$ prawdziwa jest nierówność:

$$T(n) \leq cf(n)$$

Przeanalizujmy przykład funkcji złożoności obliczeniowej, która wyraża się wzorem $T(n) = 5 * n - 4$.

Udowodnimy, że $T(n) = O(n)$. W tym celu musimy wskazać takie $n_0$ i takie $c$, aby dla każdego $n$ większego od $n_0$ była spełniona nierówność:

$$5 * n - 4 \leq c*n$$

Wystarczy przyjąć:

$n^0 = 1$
$c = 5$

Wtedy $5 * n - 4 < 5 * n$, co jest spełnione dla wszystkich $n \geq 1$. Udowodniliśmy w ten sposób, że $T(n) = O(n)$.

## ZADANIA DO WYKONANIA

1. Wykonaj w postaci schematu blokowego algorytm obliczania sumy kolejnych liczb naturalnych, który dodatkowo obliczy złożoność czasową.
2. Przedstaw jako zapis programu w języku C++ algorytm obliczania sumy kolejnych liczb naturalnych, który dodatkowo obliczy złożoność czasową.
3. Wyznacz czasową złożoność obliczeniową oraz klasę czasowej złożoności obliczeniowej dla następującego algorytmu:

   **Wejście:**
   $n$ – ilość liczb w tablicy
   $T[]$ – tablica zawierająca $n$ liczb
   **Wyjście:**
   $s$ – wynik pracy algorytmu
   Dane pomocnicze:
   $i, j$ – indeksy elementów
   Krok 1: $s = 0$
   Krok 2: $i = 0$
   Krok 3: Jeśli $i = n - 0$, to zakończ
   Krok 4: $j = 0$
   Krok 5: Jeśli $j = 10$, to idź do Kroku 9
   Krok 6: $s = s + T[i + j]$
   Krok 7: $j = j + 1$
   Krok 8: Idź do Kroku 5
   Krok 9: $i = i + 1$
   krok 10: Idź do Kroku 3
4. Algorytm zapisany w zadaniu 3. przedstaw jako kod programu C++.

## PODSUMOWANIE LEKCJI

- **Czasowa złożoność obliczeniowa** to ilość czasu potrzebnego do rozwiązania problemu w zależności od tego, ile jest danych wejściowych. ◂ s. 94
- Czasową złożoność obliczeniową wyraża się w jednostkach czasu albo w **liczbie operacji dominujących**, które należy wykonać dla $n$ danych, aby otrzymać rozwiązanie. ◂ s. 94
- **Złożoność pamięciowa** ma za zadanie określić liczbę komórek pamięci, która będzie zajęta przez dane i wyniki pośrednie tworzone w trakcie pracy algorytmu. ◂ s. 94
- Klasy złożoności obliczeniowej określają rząd funkcji $T(n)$. Jednym ze sposobów określenia rzędu tej funkcji jest popularna **notacja omikron**, zwana także notacją dużego O. ◂ s. 96

# 14. Przez 1 i siebie, czyli jak zbadać, czy dana liczba jest liczbą pierwszą

**NA TEJ LEKCJI:**

- dowiesz się, czym jest sito Eratostenesa;
- nauczysz się sprawdzać, czy liczba jest pierwsza.

Matematycy badający liczby odkryli, że niektóre są podzielne jedynie przez 1 lub samą siebie. Nazwali je pierwszymi, a pozostałe złożonymi. Dlaczego? Otóż liczby pierwsze mają zadziwiająca cechę. Każdą liczbę naturalną (oprócz 1) można przedstawić za pomocą iloczynu liczb pierwszych. Ten proces nazwano rozkładem na czynniki pierwsze. Jak sprawdzić, czy dana liczba jest liczbą pierwszą? Okazuje się, że nie jest to takie proste.

## 14.1. Starożytni znaleźli sposób, czyli sito Eratostenesa wyszukuje liczby pierwsze z przedziału

Podobnie jak nie można wskazać największej z liczb, również nie ma możliwości wskazania największej liczby pierwszej. Uczeni starają się wyznaczyć coraz to większe, ale okazuje się, że im dalej na osi liczbowej, tym występują one rzadziej. Obecnie największą odnalezioną jest 2 do potęgi 74207281 pomniejszona o 1. Zapisanie jej za pomocą cyfr zajęłoby resztę objętości podręcznika, ponieważ zawiera ona 22 338 618 cyfr. Sprawdzenie, czy liczba tego rzędu wielkości jest liczbą pierwszą, stanowi wielkie wyzwanie nawet dla najszybszych komputerów na świecie.

Znane algorytmy radzą sobie sprawnie do pewnego zakresu. Jedna z pierwszych metod została opracowana ponad 2000 lat temu przez Greka Eratostenesa. Nazwano ją sitem Eratostenesa. Oto algorytm w postaci listy kroków.

sito Eratostenesa

**Sito Eratostenesa**

1. Określ liczbę naturalną $n$ z przedziału od 2 do $n$, czyli zakres badanych liczb.
2. Usuń ze zbioru wszystkie liczby większe od 2 i podzielne przez 2.
3. Wybierz najmniejszą z niewykreślonych liczb i usuń ze zbioru wszystkie liczby przez nią podzielne, oprócz niej samej.
4. Wykonuj punkt 3. do wyczerpania zasobu liczb z założonego przedziału od 2 do $n$.

Tab. 14.1. Przykład zastosowania sita Eratostenesa dla $n$=16

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| **2** | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| **2** | **3** | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| **2** | **3** | 4 | **5** | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| **2** | **3** | 4 | **5** | 6 | **7** | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| **2** | **3** | 4 | **5** | 6 | **7** | 8 | 9 | 10 | **11** | 12 | 13 | 14 | 15 | 16 |
| **2** | **3** | 4 | **5** | 6 | **7** | 8 | 9 | 10 | **11** | 12 | **13** | 14 | 15 | 16 |

*n* niewykreślone liczby z zakresu od 2 do *n*, gdzie *n*=16

*n* liczby wykreślone

*n* liczby pierwsze

Powyższy algorytm jest skuteczny w odnajdywaniu kolejnych liczb pierwszych. Zrealizowany za pomocą programu komputerowego wymagałby zarezerwowania znacznej ilości pamięci do ich przechowywania. Ilość ta nie byłaby stała, lecz zależna od zakresu badanych liczb.

## 14.2. Pierwsza czy złożona, czyli algorytm badania liczby

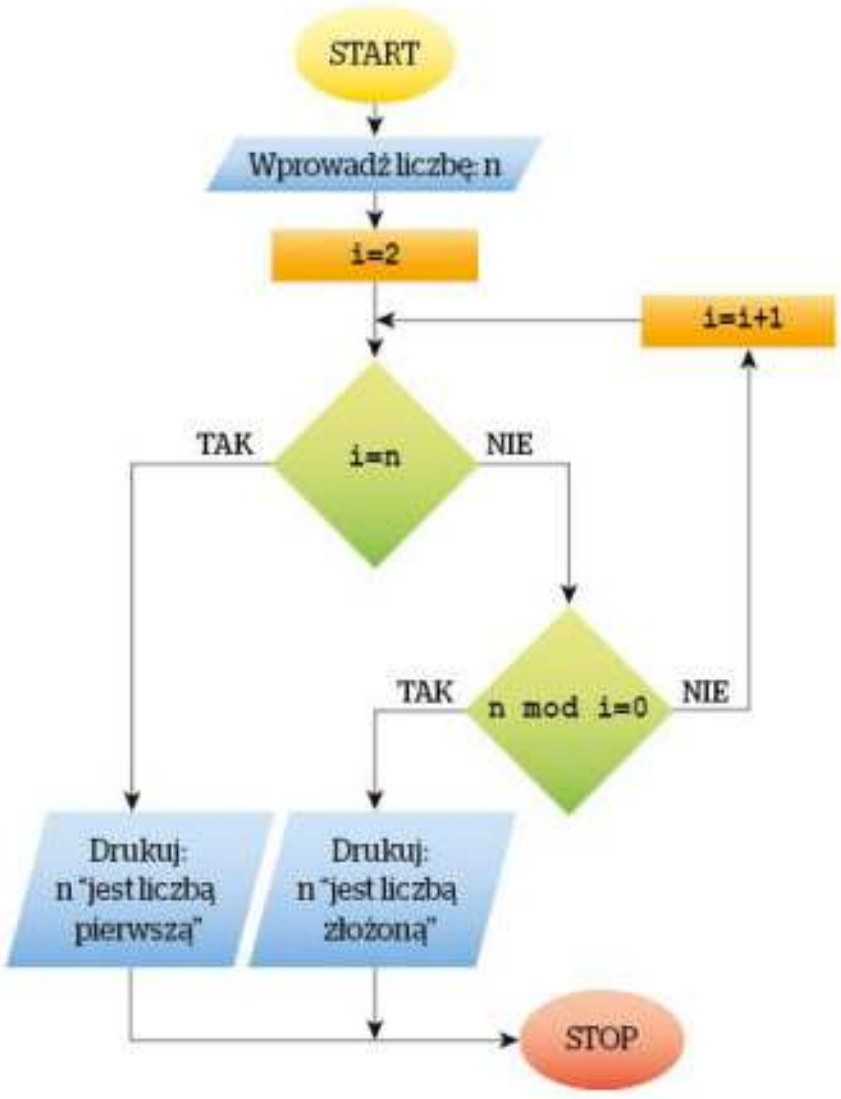
Liczby pierwsze odgrywają bardzo ważną rolę w informatyce.

wielkie liczby pierwsze

Bardzo duże liczby pierwsze są wykorzystywane w niektórych trudnych do złamania szyfrach z kluczem publicznym, jak RSA (algorytm Rivesta-Shamira-Adlemana) wykorzystywanym między innymi do szyfrowania operacji bankowych. O kluczach publicznych przeczytasz w rozdziale o szyfrowaniu.

liczby pierwsze w informatyce

Jedna z metod badania liczby naturalnej wykorzystuje właściwość, że liczba pierwsza nie ma dzielników (liczb naturalnych) w przedziale od 2 do pierwiastka kwadratowego z siebie <2,[$\sqrt{n}$]>. Jeśli znajdzie się dzielnik, dla którego reszta z dzielenia będzie miała wartość 0, to liczba ta jest **złożona**. Na tym obliczenia można zakończyć. Inna metoda polega na badaniu reszty z dzielenia liczby $n$ przez kolejne liczby naturalne z zakresu od 2 do $n$ (rys. 14.1.).




Rys. 14.1. Schemat blokowy algorytmu wykrywającego liczby pierwsze

**Uwaga!**

Operator mod użyty w schemacie blokowym oblicza resztę z dzielenia. W języku C++ zostanie użyty operator %.

Prześledźmy algorytm z rysunku 14.1. i sprawdźmy jego działanie na przykładzie liczb 6 (tab. 14.2.) i 5 (tab. 14.3.).

Tab. 14.2. Test algorytmu wykrywającego liczby pierwsze dla liczby 6

| Krok | Działanie | Wynik | |
|---|---|---|---|
| 1. | Wczytaj `n` | $n = 6$ | |
| 2. | Zmienna `i` przyjmuje wartość 2 | `i = 2` | |
| 3. | Badaj warunek `i = n` | $2 \neq 6$ | NIE |
| 4. | Zbadaj resztę z dzielenia $n$ przez `i` | 6/2=3 reszta 0 | TAK |
| 5. | Drukuj komunikat | 6 jest liczbą złożoną | |
| 6. | STOP | Koniec algorytmu | |

Tab. 14.3. Test algorytmu wykrywającego liczby pierwsze dla liczby 5

| Krok | Działanie | Wynik | |
|---|---|---|---|
| 1. | Wczytaj `n` | $n = 5$ | |
| 2. | Zmienna `i` przyjmuje wartość 2 | `i = 2` | |
| 3. | Badaj warunek `i = n` | $2 \neq 5$ | NIE |
| 4. | Czy reszta z dzielenia $n$ przez `i = 0` | 5/2=2 reszta 1 | NIE |
| 5. | Zwiększ wartość zmiennej `i` o 1 | `i = 3` | |
| 6. | Badaj warunek `i = n` | $3 \neq 5$ | NIE |
| 7. | Czy reszta z dzielenia $n$ przez `i = 0` | 5/3=1 reszta 2 | NIE |
| 8. | Zwiększ wartość zmiennej `i` o 1 | `i = 4` | |
| 9. | Badaj warunek `i = n` | $4 \neq 5$ | NIE |
| 10. | Czy reszta z dzielenia $n$ przez `i = 0` | 5/4=1 reszta 1 | NIE |
| 11. | Zwiększ wartość zmiennej `i` o 1 | `i = 5` | |
| 12. | Badaj warunek `i = n` | $5 = 5$ | TAK |
| 13. | Drukuj komunikat | 5 jest liczbą pierwszą | |
| 14. | STOP | Koniec algorytmu | |

Ten sam algorytm można zapisać w postaci listy kroków.

**Specyfikacja i lista kroków**

Tab. 14.4. Algorytm wykrywający liczby pierwsze w postaci listy kroków

| Specyfikacja problemu | Zbadać, czy dana liczba naturalna większa od 1 jest liczbą pierwszą, i wydrukować odpowiedni komunikat. |
|---|---|
| Dane wejściowe | `n`<br>zmienna zawierająca badaną liczbę |
| Dane wyjściowe | komunikat o tym, czy<br>`n`<br>jest liczbą pierwszą czy złożoną |
| Zmienne | `n`<br>zmienna zawierająca badaną liczbę<br>`i`<br>zmienna pomocnicza |
| Lista kroków | 1. Wczytanie wartości zmiennej<br>`n`<br>2. Ustalenie wartości zmiennej<br>`i=2`<br>3. Jeśli<br>`i=n`<br>drukuj komunikat<br>*n* `jest liczbą pierwszą`<br>zakończ algorytm<br>4. Jeśli<br>`i≠n`<br>sprawdź resztę z dzielenia<br>*n* `mod i  (n%i w języku c++)`<br>dla reszty 0 drukuj komunikat<br>*n* `jest liczbą złożoną`<br>w przeciwnym przypadku zwiększ zmienną pomocniczą o 1<br>`i:=i+1`<br>5. Ponownie badaj, czy<br>`i=n`<br>Jeśli NIE, powtarzaj punkt 4, w przeciwnym przypadku drukuj komunikat<br>*n* `jest liczbą pierwszą`<br>zakończ algorytm |

## ZADANIA DO WYKONANIA

1. Przedstaw właściwości liczb pierwszych i ich zastosowanie. Przejrzyj wiarygodne źródła, w tym także strony internetowe, aby znaleźć informacje o badaniach nad liczbami pierwszymi i ich zastosowaniach, np. nad opracowaniem wzorów na ich odnajdywanie.
2. Sprawdź działanie schematu blokowego algorytmu dla liczb 3, 7 i 11. Policz, ile razy dla każdej z nich zostanie wykonana pętla, w której badana jest reszta z dzielenia. Postaraj się ułożyć wzór, który dla liczby pierwszej poda prawidłowo liczbę przejścia przez tę pętlę.
3. Przetestuj algorytm podany w postaci listy kroków dla dwóch wybranych liczb pierwszych i dwóch złożonych. Podaj liczbę kroków dla każdej z nich.

4*. Wyszukaj w sieci informacje na temat modyfikacji poznanego algorytmu ograniczających liczbę kroków prowadzących do celu. Zapoznaj się z nimi i przedstaw, na czym one polegają. Uzasadnij ich skuteczność.

5*. Jedna z metod badania liczby naturalnej wykorzystuje właściwość, że liczba pierwsza nie ma dzielników w przedziale od 2 do pierwiastka kwadratowego z siebie. Zmodyfikuj algorytm omówiony w rozdziale tak, by wykorzystywał tę właściwość.

## PODSUMOWANIE LEKCJI

s. 99 • **Wielkie liczby pierwsze** odgrywają dużą rolę w kryptologii i szyfrowaniu danych komputerowych. Dzięki nim można tworzyć bardzo trudne do przełamania szyfry z kluczem publicznym.

s. 98 • **Metoda sito Eratostenesa** powstała w starożytności; pozwala znaleźć liczby pierwsze spośród liczb z zakresu od 2 do $n$.

s. 99 • **Liczby 0 i 1** nie są zaliczane do liczb pierwszych. Zbiór liczb pierwszych jest **nieograniczony.** Największe odkryte liczby pierwsze można zapisać z użyciem ponad 22 milionów cyfr.

# 15. Która pierwsza, czyli jak program bada liczby

**NA TEJ LEKCJI:**

- nauczysz się układać program na podstawie algorytmu;
- dowiesz się, jak zmodyfikować algorytm.

Znasz już jeden z algorytmów, dzięki któremu można zbadać, czy liczba jest liczbą pierwszą. Wiesz, jak działa, i sprawdziłeś/sprawdziłaś go na przykładach. Czas na ułożenie na jego podstawie programu w języku programowania. Przypomnij sobie zagadnienia z działu II. Będą ci potrzebne opisane tam narzędzia.

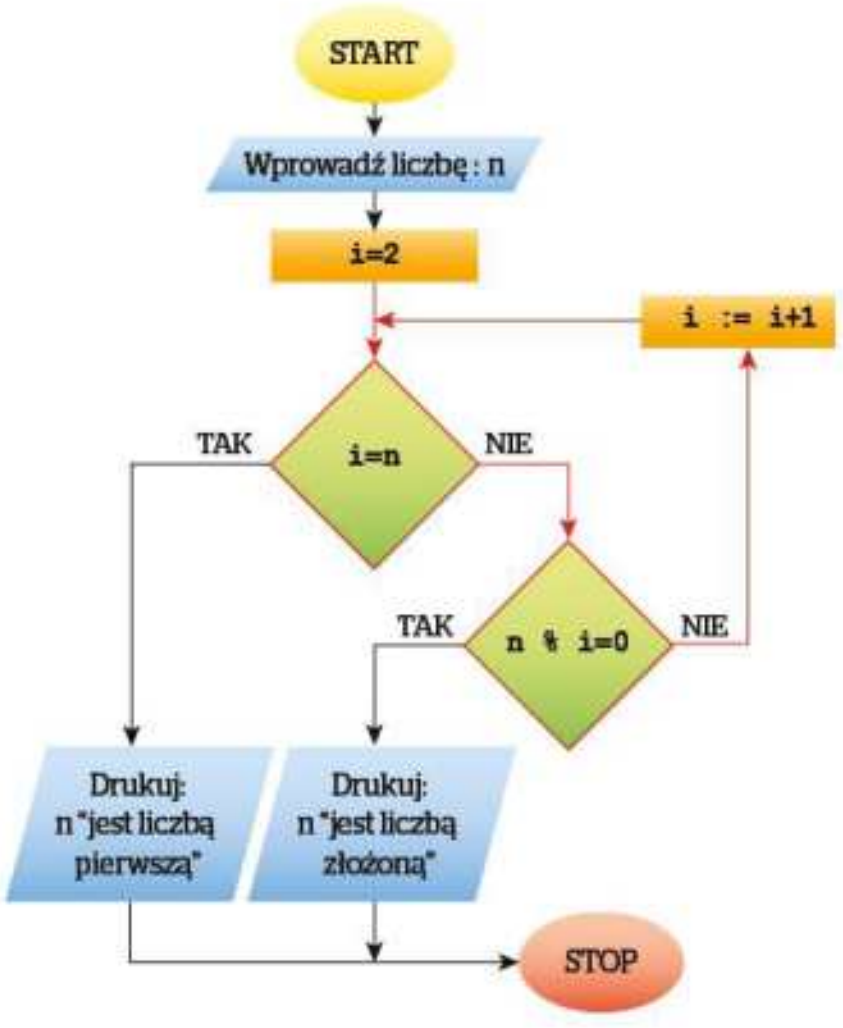
## 15.1. Ile razy powtórzyć, czyli pętla w programie badania liczb

program na podstawie algorytmu

Przeanalizuj algorytm, według którego będziesz układać program. Przyjrzyj się szczególnie pętli zawierającej oba warunki w przypadku ich wystąpienia (kierunki NIE; rys. 15.1.).

Jeśli badaną liczbą będzie na przykład 7, to ta pętla będzie „obchodzona" 5 razy, kolejno dla wartości zmiennej *i* od 2 do 6. Dla większych liczb będzie tych „obejść" odpowiednio więcej ($n$–2). W programie musi więc być polecenie, które zrealizuje tę pętlę. Spróbujmy zapisać słowami, co dzieje się, gdy warunek `i=n` nie jest spełniony. Można to zrobić na kilka sposobów, na przykład tak:

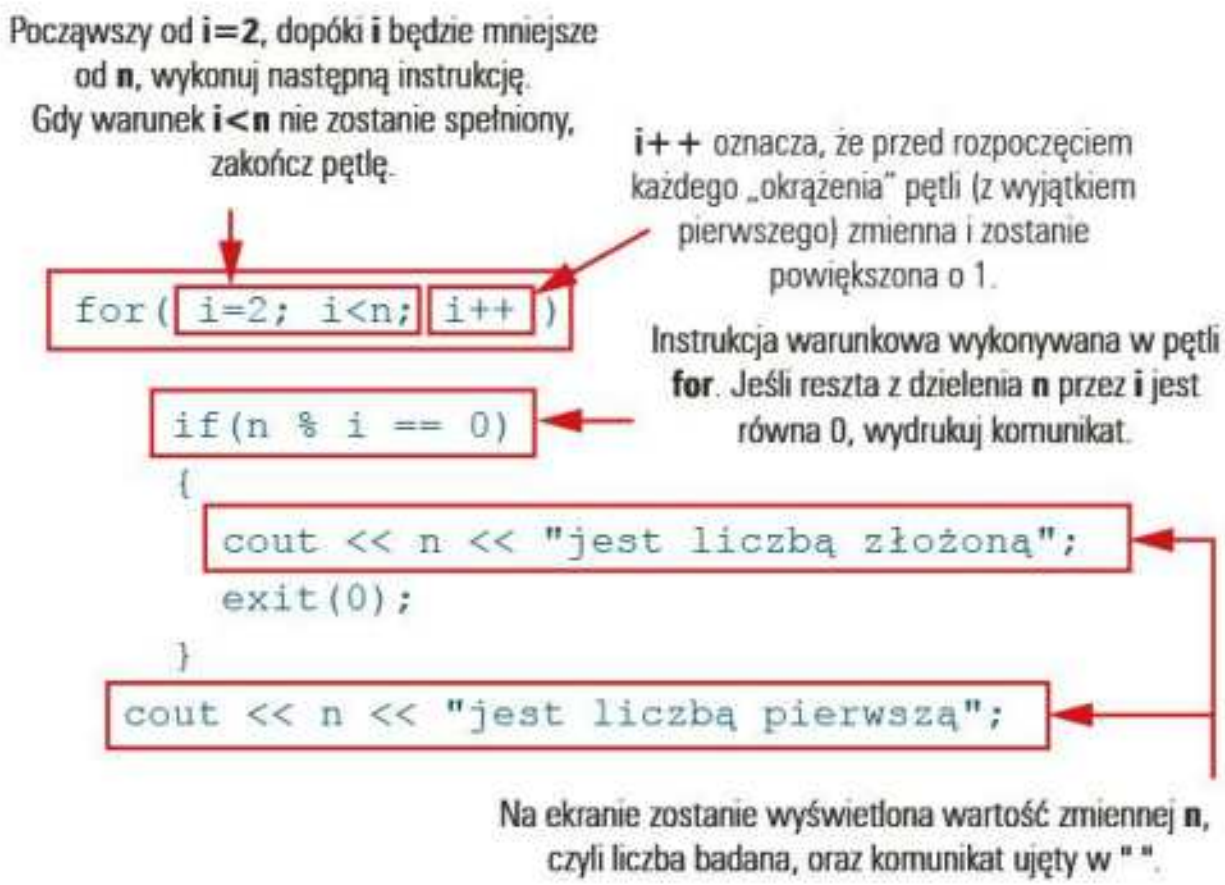
Dla każdej kolejnej wartości naturalnej `i`, począwszy od 2 (to wynika z pełnego algorytmu), aż do liczby o 1 mniejszej od badanej, sprawdzaj, czy reszta z dzielenia *n* przez `i` jest zerem. Jeśli tak, opuść pętlę.




Rys. 15.1. Algorytm z zaznaczoną pętlą obliczania reszty z dzielenia

kierunek strumienia danych

Czy podobnie można to zapisać w języku C++? Nie identycznie, ponieważ C++, jak każdy język, ma swoją składnię. Jednak podobieństwa będą widoczne (rys. 15.2.).




Rys. 15.2. Fragment programu w języku C++ realizujący pętlę

Ponieważ instrukcja warunkowa `if` jest jedyną instrukcją w pętli `for`, będzie wykonywana dopóki parametry pętli na to pozwolą. Czyli:

```
i<n
```

Instrukcja zakończenia programu:

```
exit(0);
```

nie tylko kończy program, lecz także uniemożliwia wyświetlenie dwóch sprzecznych komunikatów w przypadku, gdy liczba okaże się złożona. Po prostu instrukcja:

```
cout << n << " jest liczbą pierwszą";
```

nie zostanie już wykonana.

Polecenia z rysunku 15.2. nie stanowią jeszcze programu. Są jedynie zapisem fragmentu algorytmu z użyciem poleceń i składni języka C++. W rzeczywistości działający program musi zawierać także inne polecenia (rys. 15.3.). Jak widać, omawiany wcześniej fragment stał się częścią programu ułożonego w języku C++.

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5 int n, p;
6 cout << "Podaj liczbę naturalną: ";
7 cin >> n;
8 for (p=2; p<n; p++)
9 if(n % p == 0)
10 {
11 cout << n << " jest liczbą złożoną";
12 exit(0);
13 }
14 cout << n <<" jest liczbą pierwszą";
15 exit(0);
16 }
17
```

Rys. 15.3. Program sprawdzający, czy liczba *n* jest liczbą pierwszą

**1 linia** dołącza do programu podstawową bibliotekę obsługującą operacje wejścia wyjścia. Dzięki niej można między innymi wprowadzać dane na przykład z klawiatury i wyświetlać dane na ekranie, jak w przykładzie.

**Linia 2** to instrukcja, która ustala tak zwaną przestrzeń nazw. Dzięki temu nie trzeba pisać nazwy przestrzeni (`std`) przed każdym identyfikatorem (nazwą) z biblioteki iostream. Na przykład bez tej instrukcji linia 7 musiałaby mieć postać using `std::cin >> n;` zamiast `cin >> n`.

**Linia 3** to początek głównej funkcji programu kończącej się w **linii 16**.

**Linia 5** zawiera deklaracje zmiennych. W tym przypadku informujemy kompilator, że będziemy używać dwóch zmiennych o nazwach `n` oraz `p`, które będą liczbami całkowitymi. Świadczy o tym słowo `int` (typ zmiennej całkowitej – integer).

**Linia 6** to polecenie wyświetlenia na ekranie napisu (wysłania strumienia na konsolę). Znaki `<<` symbolizują kierunek wysyłania tych danych – w tym przypadku napis ujęty w " " na wyjście (na ekran – konsolę). Każdy prawidłowo sformułowany komunikat ułatwi użytkownikowi korzystanie z programu.

**Linia 7** zawiera instrukcję, która zapisze pod nazwą zmiennej wartość wprowadzoną z klawiatury. Ponieważ *n* było deklarowane jako `int`, użytkownik powinien wprowadzić liczbę całkowitą. Zatwierdzenie wpisu następuje po wciśnięciu **Enter**. Ponieważ badamy jedynie liczby naturalne >1, informujemy o tym w komunikacie (wyświetlanym w poleceniu z linii 6). Można także wprowadzić do programu badanie tego warunku.

**Linie od 8 do 15** zajmuje właściwe badanie liczby i wyświetlanie jego wyniku, które zostało opisane powyżej.

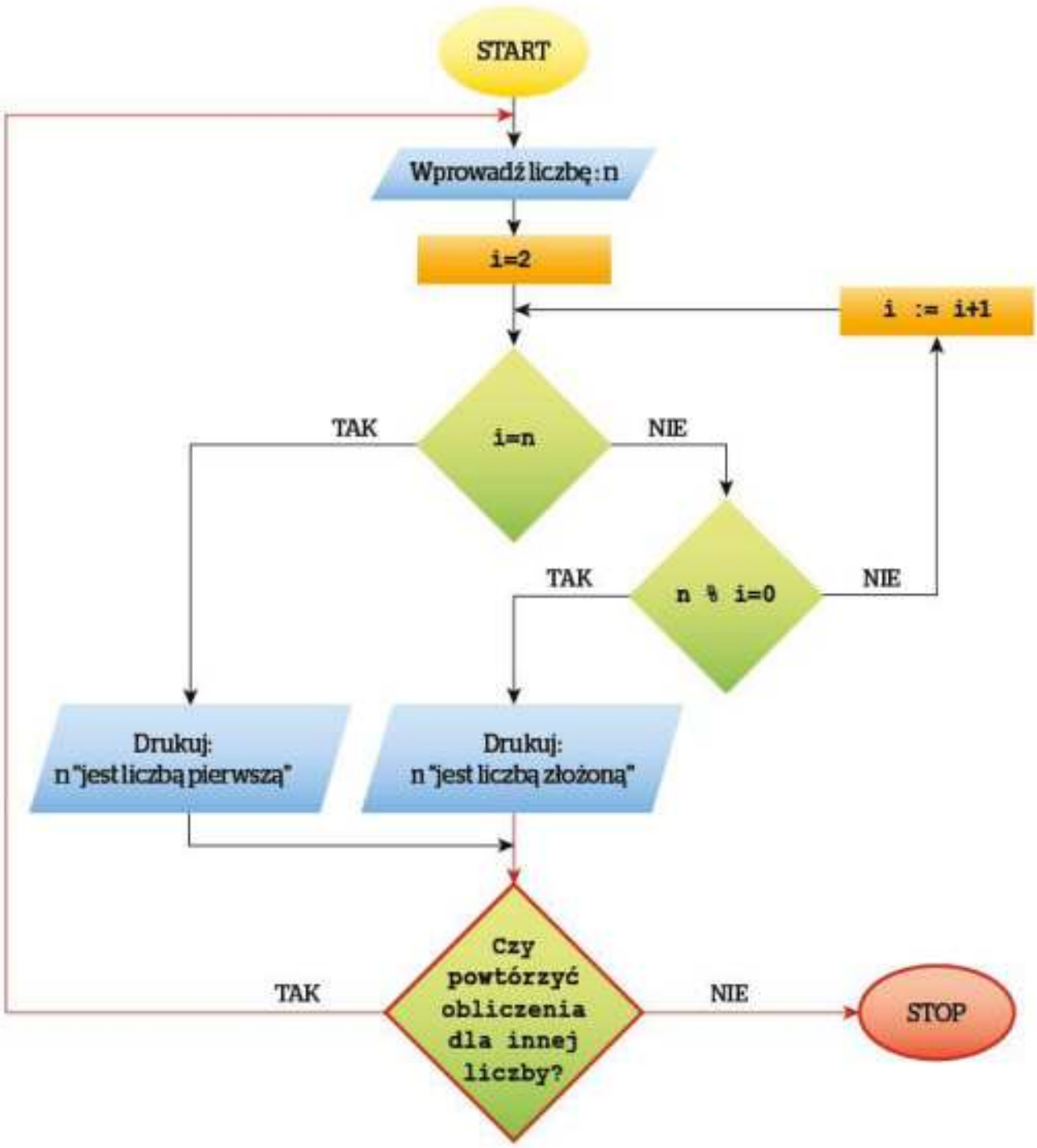
Polecenie `exit (0)` oznacza zakończenie programu i usunięcie z pamięci jego danych.

**Linia 16** zawiera nawias `}` kończący główną funkcję `main()` programu.

## 15.2. Badamy następną, czyli modyfikujemy program

modyfikowanie algorytmu i programu

Ułożony program, mimo że działa prawidłowo, ma pewne wady – na przykład testuje tylko jedną liczbę i kończy pracę. Aby przetestować następną, należy ponownie go uruchomić. Zmodyfikujmy algorytm w taki sposób, by możliwe było testowanie następnych liczb lub zakończenie działania programu (rys. 15.4.).




Rys. 15.4. Algorytm umożliwiający sprawdzanie dowolnej ilości liczb

Modyfikacja nie jest skomplikowana i polega na dodaniu badania jednego warunku. Czy opracowanie na jego podstawie programu będzie równie łatwe? Po pierwsze należy użyć nowej zmiennej, na przykład `k`, która będzie badana w warunku. Po drugie należy znaleźć odpowiednie miejsce na umieszczenie w programie instrukcji warunkowej. Zastosujemy także nieco inną realizację algorytmu badania liczb. Użyjemy pętli `while` i `do while` (rys. 15.5.). Do dzieła.

```cpp
#include <iostream>
using namespace std;
int main()
{
int i=2, n, k;

do
    {
    cout <<endl<<"podaj liczbe naturalna wieksza od 1  ";
    cin >> n;

    while (n%i!=0 && i<=n)

        i++;
        if (i*i<=n)
            cout <<n<< " jest liczbą złożoną"<<endl;
        else
            cout <<n<< "  jest liczba pierwsza"<<endl;

    cout << "Wprowadzasz nową liczbe? 1-TAK, dowolny klawisz-NIE ";
    cin >> k;
    i=2;
    }
while (k==1);

    cout <<endl<<"koniec programu";
    exit (0);

}
```

Rys. 15.5. Przykładowy program do badania wielu liczb

W miejscu pętli for z pierwszej realizacji algorytmu, pojawiła się pętla `while (n%i!=0 && i<=n)`, która będzie realizowana, dopóki reszta z dzielenia *n* przez `i` jest różna od 0 (`!=`) oraz `i` jest mniejsze od badanej liczby lub jej równe.

◂ pętle

A co z wprowadzaniem następnych liczb? Organizuje to pętla `do while (k==1);`, która działa nieprzerwanie, gdy `k` jest równe 1 (wartość zmiennej wprowadzona z klawiatury). Oznacza to, że podstawienie pod `k` każdej innej wartości spowoduje opuszczenie pętli.

Efekt działania programu można obserwować w konsoli (rys. 15.6.).

◂ efekt działania programu

```
podaj liczbę naturalną większą od 1  23
23 jest liczbą pierwszą
Wprowadzasz nową liczbę? 1-TAK, dowolny klawisz-NIE 1

podaj liczbę naturalną większą od 1  32
32 jest liczbą złożoną
Wprowadzasz nową liczbę? 1-TAK, dowolny klawisz-NIE 1

podaj liczbę naturalną większą od 1  445
445 jest liczbą złożoną
Wprowadzasz nową liczbę? 1-TAK, dowolny klawisz-NIE 3

koniec programu
```

Rys. 15.6. Efekt działania programu dla przykładowych danych

Algorytmy sprawdzania, czy liczba jest pierwszą, a więc i programy ułożone na ich podstawie mogą się różnić. Zapewne zauważyłeś/zauważyłaś, że program z rysunku 15.6. stosuje inny sposób – zamiast badania warunku `i<n` bada `i*i<=n`. To właśnie przykład zastosowania innej metody sprawdzania liczby.

## ZADANIA DO WYKONANIA

1. Zbadaj, czy program z rysunku 15.5. można zmodyfikować tak, by był badany warunek `i<=n` zamiast `i*i<=n` i nie zmieniło to poprawności działania programu, a przy tym efekt działania programu pozostał bez zmian. Przetestuj rozwiązanie na kilku przykładach. Porównaj wyniki.
2. Zmień komunikaty wyświetlane przez program z zachowaniem poprawności treści, które niosą. Zmodyfikuj sposób wyświetlania, posługując się poleceniem `endl;`.
3. Zmodyfikuj algorytm, dodając badanie, czy wprowadzona liczba *n* jest większa od 1. Jeśli tak nie będzie, program powinien podać odpowiedni komunikat i poprosić o ponowne wpisanie liczby.
4. Ułóż program na podstawie algorytmu z zadania 3. Sformułuj odpowiednie komunikaty i zorganizuj odpowiednie ich wyświetlanie w konsoli.

5*. Zmodyfikuj algorytm i ułóż na jego podstawie program, w którym zostaną wyeliminowane wszelkie błędy wprowadzania danych, czyli liczby, które nie są naturalne oraz mniejsze od 1, a także litery i inne znaki.

## PODSUMOWANIE LEKCJI

s. 103 • Na podstawie jednego algorytmu można ułożyć wiele różniących się programów, używając do tego różnych poleceń.

s. 104 • O kierunku **strumienia danych** świadczą znaki `<<` i `>>`.

s. 106 • **Algorytm i program można modyfikować** oraz dodawać im więcej funkcjonalności.

s. 107 • **Pętle** występujące w algorytmie można realizować w programie za pomocą różnych instrukcji.

s. 107 • Efekty działania programu można obserwować w konsoli **w środowisku Eclipse,** w którym układa się i uruchamia program.

# 16. Więcej pierwszych, czyli generujemy liczby

**NA TEJ LEKCJI:**

- dowiesz się, do czego w informatyce stosuje się liczby pierwsze;
- poznasz sposoby generowania liczb pierwszych.

Liczba naturalna jest liczbą pierwszą wtedy, gdy ma dokładnie dwa różne dzielniki: 1 i samą siebie. W informatyce liczby pierwsze są powszechnie wykorzystywane, na nich opierają się zaawansowane systemy szyfrowania używane w bankowości czy przez służby państwowe i wojsko. Dlatego znajomość sposobów generacji liczb pierwszych jest obowiązkowa dla osób, które chcą w przyszłości zająć się zawodowo informatyką.

## 16.1. Znajdowanie *n* kolejnych liczb pierwszych – podejście pierwsze

W tej części zajmiemy się najprostszym sposobem generacji liczb pierwszych. Weźmiemy kolejne liczby naturalne – zaczniemy od liczby 2, ponieważ liczba 1 dzieli się tylko przez siebie i brakuje nam drugiego dzielnika.

Wybraną liczbę naturalną $p$ próbujemy dzielić przez liczby od 2 do $p - 1$. Jeżeli żadna z tych liczb nie jest dzielnikiem naszej liczby, tylko ona sama, to oznacza, że to liczba pierwsza. Wyprowadzamy ją w specjalnym liczniku, odnotowujemy ten fakt i sprawdzamy w ten sam sposób kolejną liczbę. Gdy sprawdzimy wszystkie liczby $n$, które wcześniej sobie założyliśmy, to działanie naszego algorytmu się skończy.

◀ liczba naturalna

Zapiszmy teraz ten algorytm jako listę kroków.

**Wejście:**

$n$ – liczba określająca, ile liczb pierwszych należy wygenerować, $n \in N$

**Wyjście:**

$n$ kolejnych liczb pierwszych

Zmienne pomocnicze:

$lp$ – zlicza kolejno wygenerowane liczby pierwsze – $lp \in N$

$p$ – kolejno testowane liczby naturalne – $p \in N$

$d$ – kolejne dzielniki – $d \in N$

**Lista kroków:**

Krok 1: $lp = 0$

Krok 2: $p = 2$

Krok 3: Dopóki $lp < n$, wykonuj kroki od Kroku 4 do Kroku 8

Krok 4: Dla $d = 2, 3, ..., p - 1$ wykonuj Krok 5

Krok 5: Jeśli $p$ mod $d = 0$, idź do Kroku 8

Krok 6: Wypisz $p$

Krok 7: $lp = lp + 1$

Krok 8: $p = p + 1$

Krok 9: Zakończ

Zapiszmy teraz powyższą listę kroków w formie programu języka C++.

```
#include <iostream>
using namespace std;
int main ()
{
 unsigned int n,lp,p,d;
 bool t;

 cin >> n;
 lp = 0;
 p = 2;
 while (lp < n)
  {
   t = true;
   for(d = 2; d < p; d++)
     if(p % d == 0)
      {
        t = false;
        break;
      }
   if (t)
   {
      cout << p << " ";
      lp++;
   }
   p++;
  }
  cout << endl;
  return 0;
}
```

## 16.2. Znajdowanie *n* kolejnych liczb pierwszych – podejście drugie

liczby pierwsze w informatyce

Algorytm, który stworzyliśmy powyżej, sprawdza się tylko w sytuacjach, gdy trzeba wygenerować niezbyt dużą liczbę liczb pierwszych. Przy małym $n$ będzie on działał bardzo szybko, ale dla dużego $n$ – przeciwnie. Jest to typowa cecha algorytmów o klasie złożoności obliczeniowej $O(n^2)$. Aby się przekonać, że liczba $p$ jest liczbą pierwszą, algorytm musi wykonać $p - 2$ testy.

Zastanówmy się zatem, czy algorytm powinien sprawdzać podzielność liczby $p$ w całym przedziale $<2, p - 1>$.

Jeśli liczba $p$ jest złożona, to rozkłada się na iloczyn czynników pierwszych:

$$p = d_1 * d_2 * ... * d_k$$

Czynników tych musi być przynajmniej 2 i muszą być one różne od 1 i $p$. W przedziale od pierwiastka z $p$ do $p - 1$ może leżeć tylko jeden czynnik – gdyby było ich więcej, to ich iloczyn przekroczyłby wartość liczby $p$. Skoro drugi czynnik nie może być większy od pierwiastka z $p$, to musi być od niego mniejszy lub mu równy. Z kolei przy teście na pierwszość liczby $p$ (algorytm określający, czy dana liczba jest pierwsza, czy złożona) wystarczy nam, że znajdziemy pierwszą podzielność, aby wyeliminować $p$. Wynika z tego prosty wniosek – podzielność liczby $p$ wystarczy sprawdzić dla dzielników z przedziałów od 2 do pierwiastka całkowitego z $p$. Jeśli żaden z dzielników z tego przedziału nie dzieli $p$, to $p$ jest pierwsze, ponieważ w pozostałej części przedziału $<2, p - 1>$ nie mogą już wystąpić czynniki $p$.

Drugie ulepszenie będzie polegało na eliminacji niektórych wartości $p$. Na przykład jedyną liczbą pierwszą parzystą jest 2. Wszystkie inne są już nieparzyste. Idąc dalej tym tropem, dokonujemy dalszych eliminacji. Dwoma początkowymi liczbami pierwszymi są 2 i 3. Zatem z ciągu dalszych kandydatów na liczby pierwsze możemy wyeliminować wszystkie wielokrotności liczb 2 i 3, takie jak: 6, 8, 9, 10, 12, 14, 15... Liczby te mają postać $6k$, $6k \pm 2$ i $6k \pm 3$, dla $k = 1, 2,...$ Pozostają jedynie do sprawdzenia liczby postaci $6k \pm 1$, a tych jest już zdecydowanie mniej.

Trzecie ulepszenie polega na tym, że sprawdzamy podzielność nie przez kolejne liczby z przedziału od 2 do pierwiastka z $p$, lecz przez liczby pierwsze wpadające w ten przedział. Algorytm w miarę wyszukiwania kolejnych liczb pierwszych umieszcza je w osobnej tablicy i wykorzystuje do testowania podzielności nowych kandydatów na liczbę pierwszą. Wymaga to tablicy $n$-elementowej, ale opłaci się szybkością eliminacji liczb złożonych. Zwykle tak jest, iż szybkość pracy algorytmu zwiększa się kosztem większego zapotrzebowania na pamięć.

Zapiszmy teraz algorytm wyznaczający liczby pierwsze przez sprawdzenie w nieco ulepszonej wersji.

**Wejście:**

$n$ – liczba określająca, ile liczb pierwszych należy wygenerować, $n \in N$

**Wyjście:**

$n$ kolejnych liczb pierwszych; wygenerowane liczby pierwsze znajdują się również w kolejnych $n$ elementach tablicy $tlp$ – indeksy rozpoczynają się od 0

Zmienne pomocnicze:

$lp$ – zlicza kolejno wygenerowane liczby pierwsze, $lp \in N$
$p$ – kolejno testowane liczby naturalne, $p \in N$
$g$ – zawiera pierwiastek całkowity z $p$, $g \in N$
$k$ – używane do generacji liczb $p$, $k \in N$
$d$ – wraz z $k$ używane do generacji liczb $p$, $d \in Z$
$tlp$ – tablica liczb pierwszych, $tlp[i] \in N$, dla $i = 0, 1, ..., n-1$
$i$ – wykorzystywane przy numeracji dzielników z $tlp$, $i \in N$

**Lista kroków:**

Krok 1: $lp = 0$
Krok 2: $k = 1$; $d = 1$; $p = 1$
Krok 3: Dopóki $lp < n$, wykonuj kroki od Kroku 6 do Kroku 16
Krok 4: Jeśli $lp < 3$, to $p = p + lp$, idź do Kroku 14
Krok 5: $p = 6 * k + d$
Krok 6: Jeśli $d = 1$, to idź do Kroku 8
Krok 7: $d = 1$ i idź do Kroku 9
Krok 8: $d = -1$; $k = k + 1$
Krok 9: $g = [\sqrt{p}]$
Krok 10: $i = 2$
Krok 11: Dopóki $tlp[i] \leq g$, wykonuj kroki od Krok 12 do Krok 13
Krok 12: Jeśli $p$ mod $tlp[i] = 0$, to następny obieg pętli od Kroku 3
Krok 13: $i = i + 1$
Krok 14: $tlp[lp] = p$
Krok 15: Wypisz $p$
Krok 16: $lp = lp + 1$
Krok 17: Zakończ

## ZADANIA DO WYKONANIA

1. Napisz program z wykorzystaniem języka Java, posługując się algorytmem z rozdziału 15.1.
2. Napisz program z wykorzystaniem języka C++, posługując się algorytmem z rozdziału 15.2.
3. Napisz program z wykorzystaniem języka Java, posługując się algorytmem z rozdziału 15.2.

## PODSUMOWANIE LEKCJI

s. 109 ▸ • **Liczba naturalna** jest liczbą pierwszą wtedy, gdy ma dokładnie dwa różne dzielniki: 1 i siebie samą. Wybraną liczbę naturalną próbujemy dzielić przez liczby od 2 do $p - 1$. Jeżeli żadna z tych liczb nie jest dzielnikiem naszej liczby, tylko ona sama, to oznacza, że jest to liczba pierwsza.

s. 111 ▸ • W informatyce **liczby pierwsze** mają ogromne zastosowanie, na nich opierają się zaawansowane systemy szyfrowania używane w bankowości czy przez służby państwowe i wojsko.

# 17. Największy i najmniejszy, czyli jak znaleźć NWD i NWW

**NA TEJ LEKCJI:**

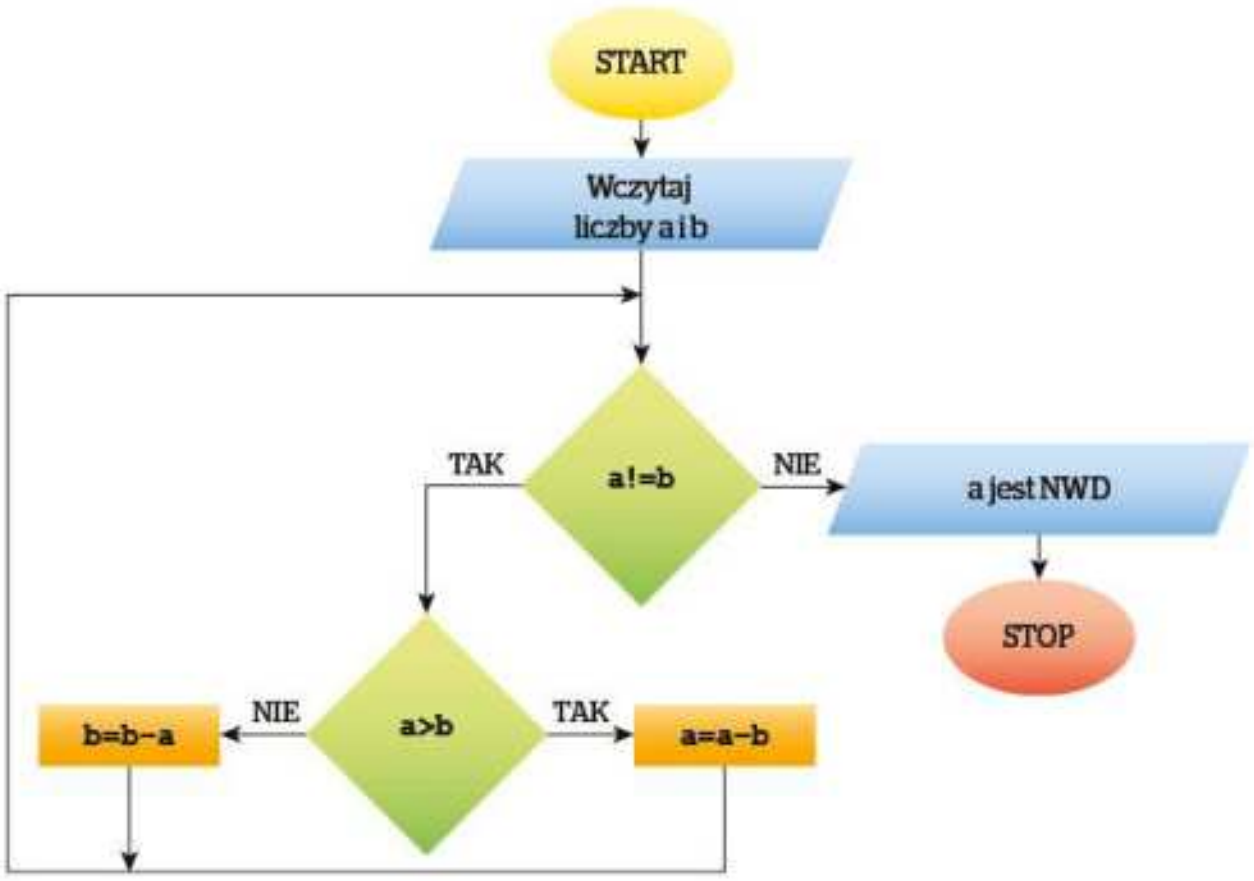
- poznasz algorytm optymalny i nieoptymalny znajdowania NWD;
- dowiesz się, jak ułożyć program, który oblicza NWD i NWW.

Pamiętasz naukę dodawania i odejmowania ułamków? Należało znaleźć największą wspólną wielokrotność mianownika. Czas nauczyć tego twój komputer. A gdy już zmusisz go do obliczenia największego wspólnego dzielnika (NWD), zmodyfikujesz algorytm tak, by znajdował najmniejszą wspólną wielokrotność (NWW).

## 17.1. Nieoptymalnie i optymalnie, czyli algorytmy Euklidesa

metoda znajdowania NWD

Metodę znajdowania NWD przypisuje się greckiemu uczonemu Euklidesowi z Aten. Jest ona skuteczna, ale przy pewnym układzie liczb może być mało wydajna. Oznacza to, że liczba kroków potrzebnych algorytmowi do zakończenia obliczeń może być znaczna i bliska większej z liczb. Dlatego tę metodę nazywamy nieoptymalnym algorytmem Euklidesa lub inaczej metodą odejmowania (rys. 17.1.).




Rys. 17.1. Algorytm nieoptymalnej metody Euklidesa

algorytm optymalny i nieoptymalny

Prześledźmy działanie algorytmu dla przykładowych liczb. Sprawdź, czy umiesz odczytywać i testować algorytmy. Zakryj kartką wiersze tabeli (tab. 17.1.). Śledź algorytm dla liczb $a = 18$ i $b = 48$. Pierwszy wiersz tabeli jest prosty do odgadnięcia. Wykonuj kolejne kroki algorytmu i sprawdzaj, czy prawidłowo odczytujesz algorytm, odkrywając kolejne wiersze poniższej tabeli.

Tab. 17.1. Realizacja nieoptymalnego algorytmu Euklidesa dla liczb 18 i 48

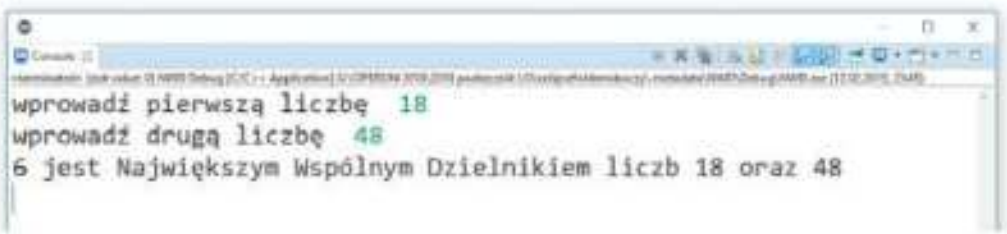
| Krok | Działanie | Wynik | |
|---|---|---|---|
| 1. | Wprowadzenie danych do algorytmu | $a = 18, b = 48$ | |
| 2. | Sprawdzenie warunku końca algorytmu ($a \neq b$) | $18 \neq 48$ | TAK |
| 3. | Porównanie liczb $a$ i $b$ ($a > b$) | $18 > 48$ | NIE |
| 4. | Wykonanie odejmowania $b-a$, wynik podstaw pod b | $b=48-18=30$ | |
| 5. | Sprawdzenie warunku końca algorytmu ($a \neq b$) | $18 \neq 30$ | TAK |
| 6. | Porównanie liczb $a$ i $b$ ($a > b$) | $18 > 30$ | NIE |
| 7. | Wykonanie odejmowania $b-a$, wynik podstaw pod $b$ | $b=30-18=12$ | |
| 8. | Sprawdzenie warunku końca algorytmu ($a \neq b$) | $18 \neq 12$ | TAK |
| 9. | Porównanie liczb $a$ i $b$ ($a > b$) | $18 > 12$ | TAK |
| 10. | Wykonanie odejmowania $a-b$, wynik podstaw pod $a$ | $a=18-12=6$ | |
| 11. | Sprawdzenie warunku końca algorytmu ($a \neq b$) | $6 \neq 12$ | TAK |
| 12. | Porównanie liczb $a$ i $b$ ($a > b$) | $6 > 12$ | NIE |
| 13. | Wykonanie odejmowania $b-a$, wynik podstaw pod $b$ | $b = 12 - 6 = 6$ | |
| 14. | Sprawdzenie warunku końca algorytmu ($a \neq b$) | $6 \neq 6$ | NIE |
| 15. | Koniec działania algorytmu | NWD = 6 | |

Algorytm obliczył NWD. Czas, by zrobił to twój komputer. Podobnie jak w poprzednim rozdziale użyjemy środowiska Eclipse i języka C++.

```
#include <iostream>
using namespace std;
int a,b,c,d;
int main()
{
cout <<"wprowadź pierwszą liczbę  ";
cin >> a;
c=a;
cout <<"wprowadź drugą liczbę  ";
cin >> b;
d=b;
while(a != b)
    if(a > b) a = a - b;
    else b = b - a;
cout << a <<" jest Największym Wspólnym Dzielnikiem liczb "<<c<<" oraz "<<d <<en
return 0;
}
```

Rys. 17.2. Przykładowy program realizujący nieoptymalny algorytm Euklidesa

Jeśli uważnie przeczytałeś/przeczytałaś rozdział 10., linie od 1 do 11 powinny być dla ciebie zrozumiałe. W linii 12 zaczyna się realizacja zasadniczej części algorytmu. Zgodnie z nim obliczenia będą prowadzone do stanu, w którym $a = b$. Dlatego w linii 12 została użyta pętla `while (a!=b)`, czyli realizowana tak długo, gdy $a \neq b$. Wewnątrz pętli (od linii 12 do 16) zgodnie z algorytmem jest badany warunek, czy $a > b$, i w zależności od wyniku następują odpowiednie działania arytmetyczne (odejmowanie). Jeśli $a$ i $b$ będą sobie równe, warunek pętli nie zostanie spełniony i wykona się polecenie z linii 15, czyli zostanie wysłany strumień danych do konsoli i wyświetli się wynik obliczeń (rys. 17.3.). Zapewne dostrzegasz, że oprócz zmiennych `a` i `b`, do których wpisuje się na początku dane liczby, zadeklarowano także zmienne całkowite `c` i `d`. Nie ma to nic wspólnego z realizacją algorytmu, ale pozwala na wyświetlenie pełnego komunikatu końcowego. Dlaczego? Przeanalizuj jeszcze raz program. Zauważ, że wartości `a` i `b` zmieniają się w trakcie jego wykonywania, a to oznacza, że gubimy dane wejściowe i ich wyświetlenie w komunikacie jest niemożliwe. Stąd zapamiętanie ich tuż po wprowadzeniu pod inną nazwą zmiennej.



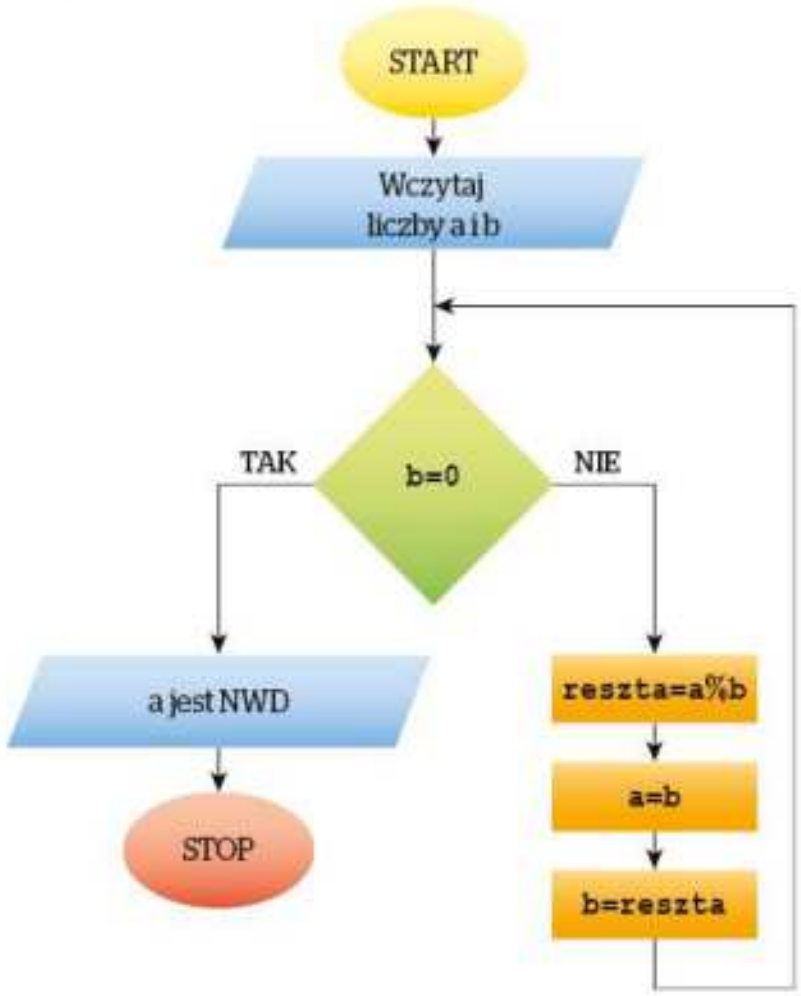

Rys. 17.3. Wynik działania programu dla liczb 18 i 48

Skoro istnieje algorytm nieoptymalny, musi istnieć optymalny, czyli wykonujący obliczenia w mniejszej liczbie kroków. Wykorzystano w nim obliczanie reszty z dzielenia, a więc to samo działanie, które wykorzystaliśmy do badania liczb pierwszych. Metodę optymalną algorytmu Euklidesa poznasz, analizując algorytm (rys. 17.4.).

**Przypomnienie**

Operator % użyty w przykładzie w języku C++ oblicza resztę z dzielenia.

Prześledź działanie algorytmu dla liczb $a = 48$ i $b = 18$ (tab. 17.2.).




Rys. 17.4. Przykładowy algorytm optymalnej metody Euklidesa

Tab. 17.2. Realizacja optymalnego algorytmu Euklidesa dla liczb 18 i 48

| Krok | Działanie | Wynik | |
|---|---|---|---|
| 1. | Wprowadzenie danych do algorytmu | $a = 48, b = 18$ | |
| 2. | Sprawdzenie, czy $b = 0$ | $b = 18$ | NIE |
| 3. | Obliczanie reszty z dzielenia $a$ % $b$ | reszta = 48 % 18 = 12 | |
| 4. | Podstawienie $a = b$ | $a = 18$ | |
| 5. | Podstawienie $b$ = reszta | $b = 12$ | |
| 6. | Sprawdzenie, czy $b = 0$ | $b = 12$ | NIE |
| 7. | Obliczanie reszty z dzielenia $a$ % $b$ | reszta = 18 % 12 = 6 | |
| 8. | Podstawienie $a = b$ | $a = 12$ | |
| 9. | Podstawienie $b$ = reszta | $b = 6$ | |
| 10. | Sprawdzenie, czy $b = 0$ | $b = 6$ | NIE |
| 11. | Obliczanie reszty z dzielenia $a$ % $b$ | reszta = 12 % 6 = 0 | |
| 12. | Podstawienie $a = b$ | $a = 6$ | |
| 13. | Podstawienie $b$ = reszta | $b = 0$ | |
| 14. | Sprawdzenie, czy $b = 0$ | $b = 0$ | TAK |
| 15. | Komunikat końcowy | $a = 6$ jest NWD | |

Porównując tabele dla obu wersji algorytmu Euklidesa, szybko dojdziesz do wniosku, dlaczego pierwszy nazwano nieoptymalnym. Dla tych samych liczb w wersji nieoptymalnej warunki są badane 9 razy. W drugim przypadku – 4. Algorytm zoptymalizowanej metody zdecydowanie zmniejsza liczbę wykonywania pętli w wypadku, gdy mamy do czynienia z parą liczb pierwszych lub kiedy ich NWD jest mały. Czas na napisanie programu w języku C++.

pętla while ▸ W pętli głównej użyjemy instrukcji `while`.

```
#include <iostream>
using namespace std;
int a,b,c,d,reszta;
int main()
{
    cout <<"wprowadź pierwszą liczbę  ";
    cin >> a;
    c=a;
    cout <<"wprowadź drugą liczbę  ";
    cin >> b;
    d=b;
while (b!=0)
 {
   reszta=a % b;
   a=b;
   b=reszta;
  }
cout << a <<" jest Największym Wspólnym Dzielnikiem liczb "<<c<<" oraz "<<d <<end
return 0;
}
```

Rys. 17.5. Przykładowa realizacja algorytmu optymalnej metody Euklidesa

Jeśli przeanalizowałeś/przeanalizowałaś algorytm (rys. 17.4.), program przedstawiony na rysunku 17.5. nie wymaga omówienia. Podobnie jak w poprzednim i tu użyto dodatkowych zmiennych przechowujących dane wejściowe, by było możliwe wyświetlenie pełnego komunikatu (rys. 17.6.).

```
wprowadź pierwszą liczbę  18
wprowadź drugą liczbę  48
6 jest Największym Wspólnym Dzielnikiem liczb 18 oraz 48
```

Rys. 17.6. Komunikaty wyświetlane przez przykładowy program obliczający NWD metodą optymalną Euklidesa

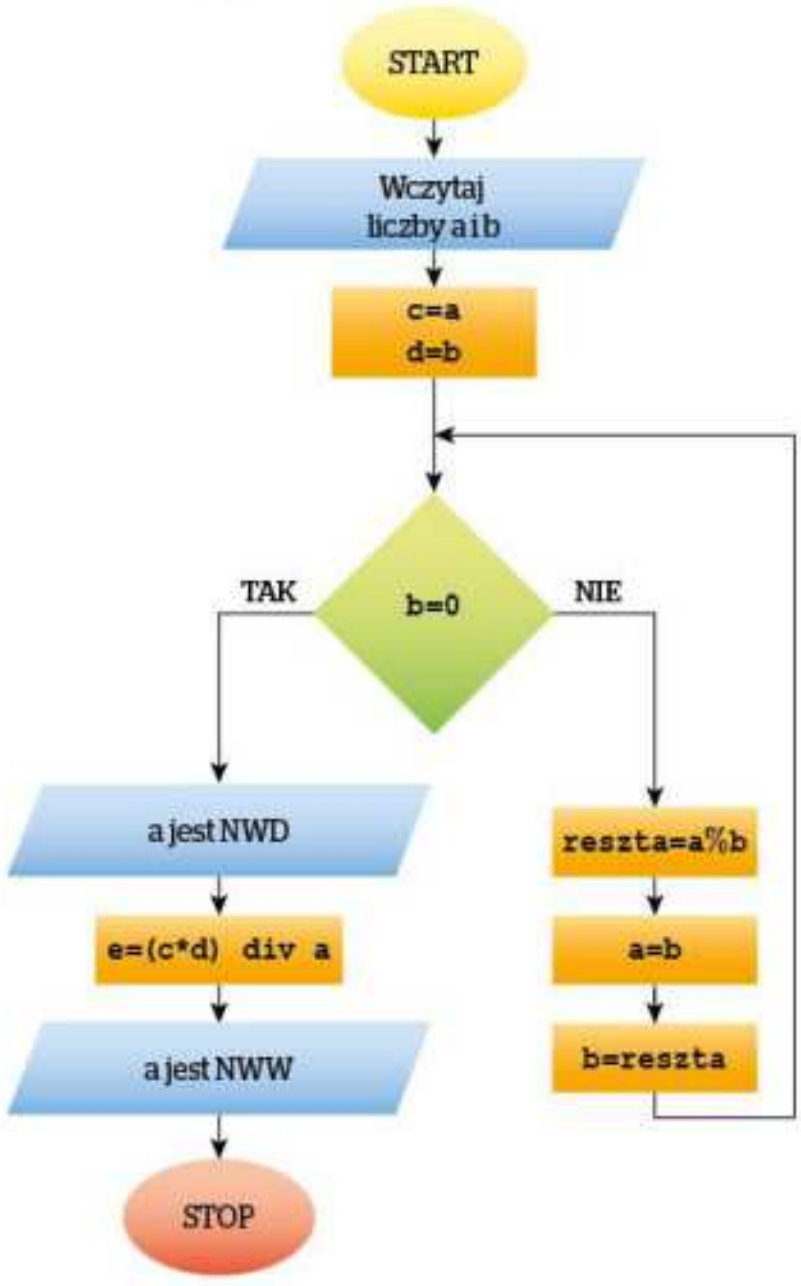
Pozostaje nam już tylko ułożenie algorytmu obliczającego najmniejszą wspólną wielokrotność NWW dwóch liczb. Potrzebny będzie tylko jeden wzór:

NWW (*a*, *b*) = (*a* * *b*) div NWD (*a*, *b*),

czyli podzielenie iloczynu obu liczb przez ich NWD. Spotkałeś/spotkałaś się już z tą metodą przy skracaniu ułamków. Zmodyfikujmy więc algorytm NWD w taki sposób, by w wyniku podawał NWW (rys. 17.7.).

Program po modyfikacji może obliczać zarówno NWD, jak i NWW (rys. 17.8.).

Zauważ, że NWW nie jest obliczane w osobnej linii programu, lecz bezpośrednio w poleceniu strumieniowania do konsoli `cout`. Dzięki temu nie ma potrzeby używania dodatkowej zmiennej. Byłaby potrzebna, gdyby NWW była używana w dalszej części programu.




Rys. 17.7. Algorytm obliczania NWW

```
#include <iostream>
using namespace std;
int a,b,c,d,reszta;

int main()
{
    cout <<"wprowadź pierwszą liczbę  ";
    cin >> a;
    c=a;
    cout <<"wprowadź drugą liczbę  ";
    cin >> b;
    d=b;
while (b!=0)
 {
   reszta=a % b;
   a=b;
   b=reszta;
 }
cout << a <<" jest Największym Wspólnym Dzielnikiem liczb "<<c<<" oraz "<<d <<endl;
cout << c*d/a <<" jest Najmniejszą Wspólną Wielokrotnością liczb "<<c<<" oraz "<<d <<endl;
return 0;
}
```

Rys. 17.8. Program obliczający NWW

Efektem działania tego programu jest wyświetlenie zarówno NWD, jak i NWW dla tej samej pary liczb.




Rys. 17.9. Efekt działania programu obliczającego NWD i NWW dla tych samych liczb

## ZADANIA DO WYKONANIA

1. Wyszukaj w internecie lub literaturze artykuły na temat metody Euklidesa. Porównaj proponowane rozwiązania z podanymi w rozdziale. Wskaż, na czym polegają różnice.
2. Zmodyfikuj program NWW tak, by umożliwiał obliczanie NWD dla dowolnej ilości par liczb. Inspiracji szukaj w poprzednim rozdziale. Dobierz odpowiednie komunikaty dla użytkownika.
3. Ułóż program obliczający NWW z zastosowaniem algorytmu nieoptymalnego Euklidesa znajdującego NWD.

4*. Ułóż program wskazujący taką parę liczb spośród 5 wprowadzonych, która będzie miała największy NWD. Opracuj odpowiednie komunikaty i zadbaj o wygodne wprowadzanie danych do programu. Skorzystaj z doświadczenia tworzenia programu według optymalnej metody Euklidesa.

## PODSUMOWANIE LEKCJI

s. 113 • Autorstwo metod obliczania **NWD jest przypisywane Euklidesowi**, a algorytmy zostały nazwane jego imieniem.

s. 113 • Istnieje **algorytm nieoptymalny i optymalny**. Różnica w ilości powtórzeń pętli jest tym większa, im mniejszy względem najmniejszej z badanych liczb jest NWD.

s. 116 • Do realizacji pętli algorytmów NWD i NWW najwygodniej jest użyć **pętli while.**

# 18. Euklides i rekurencja, czyli szukamy NWD

**NA TEJ LEKCJI:**

- przypomnisz sobie, czym jest NWD i do czego jest wykorzystywany;
- dowiesz się, czym jest rekurencja.

Największym wspólnym dzielnikiem (NWD) dla dwóch (lub więcej) liczb naturalnych jest największa liczba naturalna dzieląca każdą z nich. NWD znajdujemy za pomocą algorytmu Euklidesa. Uczony napisał go około 300 r. p.n.e. Algorytm Euklidesa dotyczył dwóch odcinków. Miał posłużyć do wskazania wspólnej miary, za pomocą której można by było zmierzyć każdy z odcinków, to znaczy dało się ją odłożyć na obu odcinkach całkowitą liczbę razy.

## 18.1. Szukamy NWD

Największy wspólny dzielnik liczb $a$ i $b$ zapisuje się zwykle nwd($a$, $b$) lub NWD($a$, $b$). Pojęcie to najczęściej wykorzystuje się podczas redukcji ułamków do postaci nieskracalnej, czyli takiej, w której licznik i mianownik są względnie pierwsze. ◂ NWD

Dwie liczby nazwiemy względnie pierwszymi, jeżeli ich największym wspólnym dzielnikiem jest 1, na przykład liczbami względnie pierwszymi są 10 i 49.

Przykładowo największym wspólnym dzielnikiem dla liczb 42 i 56 jest liczba 14, stąd:

$$\frac{42}{56} = \frac{3 \cdot 14}{4 \cdot 14} = \frac{3}{4}$$

Jak wyszukać największy wspólny dzielnik? Najprościej metodą wykorzystującą dzielenie z resztą. Aby znaleźć wspólny dzielnik dla 70 i 28, należy te liczby podzielić przez siebie – wynikiem jest 2 i 14 reszty. Następnie dzielimy 28 przez resztę, czyli przez 14, otrzymujemy 2 i 0 reszty, co oznacza, że 14 jest największym wspólnym dzielnikiem.

70 : 28 = 2 i 14 reszty

28 : 14 = 2 i 0 reszty

NWD(70;28) = 14

Poszukajmy NWD dla liczb 550 i 325:

550 : 325 = 1 i 225 reszty

325 : 225 = 1 i 100 reszty

225 : 100 = 2 i 25 reszty

100 : 25 = 4 i 0 reszty

NWD(550;325) = 25

Inną metodą wyszukania NWD, jest odejmowanie od większej liczby mniejszej (dotyczy to również wyniku odejmowania).

Weźmy dla przykładu liczby 54 i 36:

54 > 36, więc 54 – 36 = 18

36 > 18, więc 36– 18 = 18

ponieważ 18 = 18, NWD(54;36) = 18

## 18.2. Rekurencja

rekurencja ▸ Rekurencja, nazywana również rekursją, to nic innego jak odwoływanie się na przykład funkcji lub definicji do samej siebie.

Główne zastosowanie, oprócz informatyki, rekurencja ma w matematyce i logice. W tej ostatniej opiera się ona na założeniu istnienia stanu początkowego oraz zdania lub kilku zdań stanowiących podstawę wnioskowania.

Przykładem rekurencji w logice będzie następujący algorytm:

**Reguła:**
Każdy ojciec jest starszy od swojego syna; każdy ojciec jest czyimś synem

**Stan początkowy:**
Jestem 22-letnim mężczyzną

**Teza:**
Ojciec mojego ojca jest starszy ode mnie

**Dowód:**
1. mój ojciec jest starszy ode mnie
2. mój ojciec jest czyimś synem
3. ojciec mojego ojca jest starszy od mojego ojca
4. ojciec mojego ojca jest czyimś synem
5. itd

ciąg Fibonacciego ▸ W matematyce przykładem rekurencji będzie ciąg Fibonacciego, który definiuje funkcję, odwołując się w definicji do niej samej.

Pokrótce opisać można go w następujący sposób – pierwszy wyraz jest równy 0, drugi jest równy 1, każdy następny będzie sumą dwóch poprzednich.

W programowaniu rekurencja to wywołanie przez funkcję samej siebie. Można porównać rekurencję do pętli, ponieważ jest ona również wywoływana określoną liczbę razy.

Najprostszym przykładem obrazującym rekurencję w programowaniu będzie program obliczający silnię. Silnia *n*! to iloczyn kolejnych liczb naturalnych mniejszych od liczby *n* lub jej równych.

$$n! = \begin{cases} 1, n = 0 \\ n \cdot (n-1)!, \ n \geq 1 \end{cases}$$

Rys. 18.1. Wzór na silnię *n*!

Wykorzystując powyższy wzór, obliczymy silnię z liczby 4:

4! = 4 * 3! = 4 * 3 * 2! = 4 * 3 * 2 * 1! = 4 * 3 * 2 * 1 * 0!

4! = 4 * 3 * 2 * 1 * 1

4! = 24

W obliczeniach silni cofamy się od *n* = 4 do *n* = 0, gdy osiągniemy 0, to rekurencja ustaje.

Stwórzmy program do obliczania silni w języku C++.

```
#include <iostream>
using namespace std;
unsigned silnia(unsigned n)
{
    if(n == 0) return 1;
    else       return n * silnia(n - 1);
}

int main()
{
    unsigned n;
    cin >> n;
    cout << n << "! = " << silnia(n) << endl << endl;
    return 0;
}
```

Ten sam program stwórzmy teraz za pomocą języka Java.

```
class Silnia {
  public int silnia(int i)
  {
    if (i == 0)
      return 1;
    else
      return i * silnia(i - 1);
  }
  static public void main(String argv[])
  {
    Silnia s = new Silnia();
    System.out.println(s.silnia(10));
  }
}
```

## 18.3. Algorytm Euklidesa, czyli rekurencyjny największy wspólny dzielnik (NWD)

algorytm Euklidesa

- Algorytm Euklidesa możemy również zapisać w sposób rekurencyjny:

$$\mathrm{NWD}(a, b) = \begin{cases} a, b = 0 \\ \mathrm{NWD}(b, a \bmod b), b \neq 0 \end{cases}$$

Rys. 18.2. Algorytm Euklidesa

Może się to wydawać dziwne, lecz tak właśnie działa omawiany wcześniej algorytm Euklidesa z dzieleniem. Traktuje on liczbę *b* jako resztę z dzielenia *a* przez *b*. Jeśli reszta z dzielenia jest równa 0, to kończy, zwracając *a*. Jeśli nie, to *a* zastępuje przez *b*, czyli obecną resztą z dzielenia, a *b* zastępuje nową resztą z dzielenia i znów wykonuje sam siebie, aż do pożądanego skutku.

```
#include <iostream>
using namespace std;
unsigned NWD(unsigned a, unsigned b)
{
    if(b == 0) return a;
    else       return NWD(b, a % b) ;
}
```

```
int main()
{
    unsigned a, b;
    cin >> a >> b;
    cout << "NWD(" << a << "," << b
                   << ") = " << NWD(a, b) << endl
                   << endl;
                   return 0;
               }
```

## ZADANIA DO WYKONANIA

1. Napisz algorytm w postaci listy kroków, który wylicza NWD z wykorzystaniem algorytmu Euklidesa i rekurencji.
2. Napisz algorytm w postaci pseudokodu, który wylicza NWD z wykorzystaniem algorytmu Euklidesa i rekurencji.
3. Napisz algorytm w postaci schematu blokowego, który wylicza NWD z wykorzystaniem algorytmu Euklidesa i rekurencji.
4. Napisz program z wykorzystaniem języka Java, który wylicza NWD z wykorzystaniem algorytmu Euklidesa i rekurencji.

## PODSUMOWANIE LEKCJI

- **Największym wspólnym dzielnikiem** (NWD) dla dwóch (lub więcej) liczb naturalnych jest największa liczba naturalna dzieląca każdą z nich. Umiejętność wskazywania NWD najczęściej wykorzystuje się podczas redukcji ułamków do postaci nieskracalnej, czyli takiej, w której licznik i mianownik są względnie pierwsze. ◂ s. 119
- **Rekurencja**, nazywana również rekursją, to odwoływanie się na przykład funkcji lub definicji do samej siebie. ◂ s. 120
- W matematyce przykładem rekurencji będzie **ciąg Fibonacciego**, w którym pierwszy wyraz jest równy 0, drugi jest równy 1, a każdy następny będzie sumą dwóch poprzednich. ◂ s. 120
- NWD znajdujemy za pomocą **algorytmu Euklidesa**, który nie dotyczył liczb naturalnych, ale dwóch odcinków. ◂ s. 122

# 19. Metoda iteracji, czyli algorytm Euklidesa w innej postaci

**NA TEJ LEKCJI:**

- dowiesz się, czym jest iteracja;
- nauczysz się pisać algorytm iteracyjny.

W poprzednim rozdziale poznaliśmy metodę rekurencyjną zastosowania algorytmu Euklidesa, a w tym zajmiemy się metodą iteracyjną. Iteracja to metoda w analizie matematycznej i programowaniu, która polega na wielokrotnym stosowaniu tego samego przekształcenia lub tej samej procedury, a także to kolejne przekształcanie lub procedura.

## 19.1. Pętla równa się iteracja

iteracja

Skoro iteracja polega na wielokrotnym powtarzaniu tej samej operacji lub ciągu operacji, to tworzymy ją, stosując tzw. pętlę. Mamy z nią do czynienia, gdy w pewnym kroku algorytmu wracamy do jednego z wcześniejszych kroków.




Rys. 19.1. Schemat blokowy obrazujący działanie pętli

instrukcje pętli w języku C++

pętla for each w języku Java

Aby napisać program realizujący algorytm iteracyjny, korzystamy z instrukcji iteracyjnych, czyli z tzw. instrukcji pętli. W języku C++ instrukcjami iteracyjnymi są instrukcje: `for`, `while` i `do while`, natomiast w języku Java: `for`, `while`, `do ... while` i `for each`.

Pętla `for` w języku C++ ma postać:

**for** (wyrażenie_początkowe; warunek_powtórzenia; wyrażenie pętli)
instrukcja;

**– wyrażenie_początkowe** – wykonywane jest jednokrotnie na początku, służy do ustawienia wartości tzw. zmiennej sterującej pętlą;

**– warunek_powtórzenia** – jeżeli jest spełniony, to pętla powtórzy swoje działanie;

**wyrażenie_pętli** – jest wykonywane po każdym powtórzeniu pętli; to wyrażenie, które zmienia wartość zmiennej sterującej.

**Przykład:**

Wypisanie liczb od 1 do 10

```
for (x=1; x<=10; x++) cout <<x<<endl
```

Wypisanie liczb od 10 do 1

```
for (x=10; x>=1; x--) cout<<x<<endl
```

Możemy również zagnieżdżać pętle. Z zagnieżdżoną pętlą `for` mamy do czynienia wtedy, gdy instrukcją powtarzaną w pętli jest kolejna instrukcja `for`.

**Przykład:**

```
for (x=1; x<=10; x++)
{
  for (y=1; y<=10; y++)
    cout<<"*";
  cout<<endl;
}
```

Pętla `while` w C++ ma postać:

**while** (warunek_kończący_pętlę)
{
  powtarzany_blok_instrukcji
}

Dopóki warunek jest spełniony (czyli wartość logiczna wynosi true), dopóty pętla będzie powtarzała instrukcję (lub blok instrukcji) występującą bezpośrednio po słowie kluczowym `while`.

**Przykład:**

Program ma wypisać liczby od 10 do 0. Jak wypisze liczbę 0, ma pojawić się napis Booom!.

```
#include <iostream>
void odliczanie( int iStart )
{
    while( iStart >= 0 )
    {
        std::cout << iStart << ", ";
        --iStart;
    } //while
    std::cout << "Booom!" << std::endl;
}

int main()
{
    odliczanie( 10 );
    return 0;
}
```

Pętla `do ... while` w C++ ma postać:

```
do
{
   instrukcje
} while (warunek_konieczny);
```

**Przykład:**

```
#include <iostream>
int main()
{
    int ile = 4;
    do
    {
        std::cout << "Napis" << std::endl;
        ile--;
    } while( ile > 0 );

    return 0;
}
```

Pętla `for` w języku Java ma postać:

```
for (wyrażenie_początkowe; warunek; modyfikator_licznika) {
   instrukcja do wykonania
}
```

**Przykład:**

```
public class Odczyt{
  public static void main(String[] args){
    for(int i=0; i<10; i++){
      System.out.println("To jest petla");
    }
    System.out.println("Koniec petli");
  }
}
```

Pętla `while` w języku Java ma postać:

```
while (warunek) {
   instrukcje_do_wykonania
}
```

**Przykład:**

```
public class Odczyt{
  public static void main(String[] args){
    int licznik = 0;

    while(licznik<10){
      System.out.println("To jest petla");
      licznik++;
    }
          System.out.println("Koniec petli");
  }
}
```

Pętla `do ... while` w języku Java ma postać:

```
do {
   instrukcje_do_wykonania
}
while (warunek);
```

**Przykład:**

```
public class Odczyt{
  public static void main(String[] args){
    int licznik = 0;

    do{
      System.out.println("To jest petla");
      licznik++;
    }
    while(licznik<10);
    System.out.println("Koniec petli");
  }
}
```

Pętla `for each` w języku Java to konstrukcja, która pozwala na sekwencyjne przeglądanie różnych zbiorów danych – mogą to być na przykład tablice. Ma ona następującą postać:

**for** (Typ_Obiektu nazwa_obiektu: nazwa_tablicy) {

...

}

**Przykład:**

```
public class Test {
  public static void main(String args[]){
    int[] tablica = new int[10];
    //wypełnienie tablicy
    for(int i=0; i<10; i++)
      tablica[i] = i+1;

    //wyświetlenie przy użyciu pętli for each
    for(int x : tablica)
      System.out.println(x);
  }
}
```

Zapoznaliśmy się z działaniem pętli, czyli iteracją w językach C++ i Java. Spróbujmy teraz napisać algorytm z użyciem pętli, który obliczy NWD z wykorzystaniem algorytmu Euklidesa.

**Wejście:**

$a$, $b$ – liczby naturalne, których NWD poszukujemy

**Wyjście:**

$t$ – tymczasowo przechowuje dzielnik i należy do zbioru liczb naturalnych

**Lista kroków:**

Krok 1: Dopóki $b \neq 0$, wykonuj kroki od Kroku 2 do kroku 4

Krok 2: $t = b$

Krok 3: $b = a \bmod b$

Krok 4: $a = t$

Krok 5: Wypisz $a$

Krok 6: Zakończ

W języku C++ algorytm Euklidesa z wykorzystaniem iteracji prezentuje się następująco:

```
#include <iostream>
using namespace std;

int main()
{
  unsigned int a,b,t;

  cin >> a >> b;
  while(b)
  {
    t = b;
    b = a % b;
    a = t;
  }
  cout << a << endl;
  return 0;
}
```

## ZADANIA DO WYKONANIA

1. Napisz program w języku C++ wyświetlający twoje imię w 5 linijkach.
2. Napisz program w języku Java wyświetlający 10 ponumerowanych linii z twoim imieniem.
3. Napisz program w języku C++, który wyprowadzi na ekran monitora prostokąt narysowany za pomocą znaków „*" o wielkości *x*, *y* podanej przez użytkownika.
4. Zmodyfikuj poniższy program napisany w języku C++ tak, aby używal pętli while zamiast pętli for.

```
#include <iostream>
void zakres( int x, int y )
{
    for( int i = x; i <= y; ++i )
        std::cout << i << ", ";

    std::cout << "koniec" << std::endl;
}

int main()
{
    zakres( 1, 5 );
    zakres( 8, - 1 );
    zakres( 10, 10 );
    return 0;
}
```

5. Napisz prosty kalkulator w języku Java, który będzie dodawać, odejmować, mnożyć i dzielić. Zabezpiecz kalkulator przed podawaniem niepoprawnych liczb i operacji.
6. Napisz program w języku Java, który pobierze od użytkownika całkowitą liczbę dodatnią. Następnie wyświetl na ekranie odliczanie z tekstem „Rakieta startuje za...", gdzie zamiast wielokropka pojawią się liczby od podanej przez użytkownika do 0. Napisz program w trzech wersjach przy użyciu różnych pętli.
7. Napisz program w języku Java, który wyszuka NWD liczb parzystych, wykorzystując algorytm Euklidesa w formie iteracyjnej.

## PODSUMOWANIE LEKCJI

s. 124 ▸ • **Iteracja** – metoda w analizie matematycznej i programowaniu polegająca na wielokrotnym stosowaniu tego samego przekształcenia lub tej samej procedury, a także kolejne przekształcanie lub procedura.

s. 124 ▸ • W programowaniu iteracja polega na wykorzystaniu pętli. W języku C++ mamy do dyspozycji **trzy rodzaje pętli:** for, while i do ... while.

s. 124 ▸ • W języku Java dochodzi dodatkowa **pętla for each**, dzięki której można wykonywać operacje na tablicach.

# 20. Dodawanie ułamków, czyli jak wykorzystać NWW i NWD w programie komputerowym

**NA TEJ LEKCJI:**

- dowiesz się, jak ułożyć algorytm dodawania ułamków;
- poznasz funkcje.

NWW i NWD stosujesz od pierwszych lat nauki matematyki. Bez nich niemożliwe byłoby na przykład dodawanie ułamków. Umiesz to robić, ale czy wiesz, jak ułożyć algorytm i program wykonujący to działanie? Czy wiesz, jak wykorzystać w nich ułożone wcześniej programy obliczające NWW? Zmierz się z tym zadaniem.

## 20.1. Łączymy algorytmy, czyli nie wyważamy otwartych drzwi

algorytm dodawania ułamków

Budowanie algorytmu zacznijmy od opisu ogólnie znanej metody dodawania ułamków na prostym przykładzie.

$$\frac{3}{16}+\frac{5}{12}=\frac{3\cdot 3}{48}+\frac{4\cdot 5}{48}=\frac{9}{48}+\frac{20}{48}=\frac{29}{48}$$

Jak widać, pierwszym krokiem było znalezienie NWW mianowników ułamków. Dla 16 i 12 to 48. Następnie należy podzielić NWW przez mianownik danego ułamka i wymnożyć wynik przez jego licznik. Mając dwa ułamki o tych samych mianownikach, wystarczy już tylko dodać do siebie liczniki i sprawdzić, czy wynik da się uprościć. Na tym właściwie można skończyć układanie algorytmu. Taka jego postać (lista kroków) jest przecież dopuszczalna. Spróbuj jednak utworzyć algorytm w postaci schematu blokowego (rys. 20.1.).

Jeśli przeanalizujesz algorytm z rysunku 20.1., dojdziesz do wniosku, że otrzymasz wynik w postaci nieuproszczonego ułamka. Rozwiązaniem tego problemu zajmiesz się w następnym rozdziale. Teraz skup się na ułożeniu programu w języku C++. Z pozoru nie powinno to sprawić ci trudności, ponieważ w algorytmie nie występują bloki warunkowe. Czy na pewno? Obliczenie NWW nie obędzie się bez warunków, ale ten problem rozwiązaliśmy wspólnie w poprzednim rozdziale.




Rys. 20.1. Przykładowy schemat blokowy algorytmu dodawania ułamków

## 20.2. Kodujemy dodawanie, czyli układamy program dodający ułamki

Do ułożenia programu potrzeba kodu programu znajdującego NWW dwóch liczb. Z algorytmu wynika, że w programie wystąpią cztery zmienne wejściowe – licznik i mianownik każdej z liczb oraz dwie zmienne – licznik i mianownik wyniku.

```
1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      int nww, a, b, i, l1, l2, m1, m2, licznik, mianownik;
7
8      cout<<"Podaj licznik pierwszego ułamka  ";
9      cin >> l1;
10     cout << "Podaj mianownik pierwszego ułamka  ";
11     cin >> m1;
12
13     cout<<"Podaj licznik drugiego ułamka  ";
14     cin >> l2;
15     cout << "Podaj mianownik drugiego ułamka  ";
16     cin >> m2;
17
18     a=m2;
19     b=m1;
20
21     while(b!=0)              //obliczanie NWD
22         {
23          i = b;
24          b = a%b;
25          a = i;
26         }                    // a = NWD
27
28     nww=m1*m2/a;
29
30     mianownik = nww;
31     licznik = mianownik/m1*l1 + mianownik/m2*l2;
32
33     cout<<l1<<"/"<<m1<<" + "<<l2<<"/"<<m2<<" = "<<licznik<<"/"<<mianownik;
34
35     return 0;
36 }
```

Rys. 20.2. Przykładowy program w języku C++ obliczający sumę dwóch ułamków

Główne obliczenia są wykonane w liniach 30 i 31. Korzystają one jednak z wcześniej obliczonego NWW. Największa wspólna wielokrotność zostaje z kolei obliczona na podstawie NWD. Tak więc te fragmenty kodu można było skopiować z programów z poprzedniego rozdziału. W programie z rysunku 20.2. nie jest konieczne używanie zmiennej `nww`. Użyto jej dla ułatwienia analizy programu. Zamiast linii 28 i 30 można by napisać: `mianownik = m1*m2/a;`.

Zwróć uwagę, jak program organizuje wprowadzenie danych. Dzięki odpowiednim komunikatom użytkownik programu wie, które wartości ma wprowadzić. Linia 33 to wydruk wyniku. Dzięki temu, że zmienne `l1`, `l2`, `m1` oraz `m2` nie zmieniają swojej wartości od momentu ich wpisania, można było podać wynik wraz z danymi wejściowymi (rys. 20.3.).

```
Podaj licznik pierwszego ułamka  3
Podaj mianownik pierwszego ułamka  16
Podaj licznik drugiego ułamka  5
Podaj mianownik drugiego ułamka  12
3/16 + 5/12 = 29/48
```

Rys. 20.3. Wynik działania przykładowego algorytmu dodawania ułamków

## 20.3. NWD i NWW jako funkcje, czyli modyfikujemy program

Ułożony program działa prawidłowo. Gdyby nie komentarze, trudno byłoby niedoświadczonemu programiście domyślić się, jakie zadanie realizuje. Jednak gdyby tak utworzyć oddzielne funkcje obliczania NWW i NWD? Wtedy, gdy w funkcji głównej należałoby użyć tych wartości, nie trzeba by ich obliczać. Obliczenia wykonałyby funkcje. W mało skomplikowanym programie zysk nie jest tak widoczny. Jeśli układa się program, w którym obliczanie pewnej wartości dla różnych danych występuje wielokrotnie, takie rozwiązanie jest wskazane. Kod staje się bardziej zrozumiały i uporządkowany, a także nie powtarzają się w nim w wielu miejscach te same fragmenty.

Przypomnij sobie, czym jest funkcja. To fragment programu, któremu nadano nazwę. Dzięki temu może być wykonywany zawsze, gdy zostanie ona użyta w kodzie. Funkcja może przetwarzać argumenty, które są zmiennymi wejściowymi. Istnieją funkcje bez argumentów.

◀ funkcja

Zapiszmy funkcję obliczającą NWD (rys. 20.4.).

```
int NWD(int a, int b)    //Funkcja obliczająca NWD
{
     int i;
     while(b!=0)
     {
      i = b;
      b = a%b;
      a = i;
     }
     return a;
}
```

Rys. 20.4. Przykładowa funkcja obliczająca NWD dwóch liczb

Łatwo zauważyć, że kod funkcji zawiera te same instrukcje, co fragment programu z rys. 20.2. To oczywiste, gdyż sposób obliczania NWD się nie zmienił. Pierwsza linia `int NWD(int a, int b)` nadaje funkcji, której kod znajduje się poniżej, nazwę NWD oraz deklaruje jej argumenty. W tym przypadku to dwie zmienne o typie danych `int`, czyli liczb całkowitych.

Jeśli w pewnym miejscu kodu głównego programu (w funkcji `main()`) należy obliczyć NWW, a do tego potrzebne jest NWD dwóch liczb, można użyć funkcji (rys. 20.4.), podając w nawiasie obie liczby oddzielone przecinkiem. Będą one argumentami funkcji.

Przykładowe użycie funkcji NWD w celu obliczenia NWW dla argumentów `a` i `b`:

```
NWW=a/NWD(a,b)*b;
```

Czas na użycie funkcji w programie dodającym dwa ułamki (rys. 20.5.).

```
1  #include<iostream>
2  using namespace std;
3
4  int NWD(int a, int b)        //Funkcja obliczająca NWD liczb całkowitych a i b
5  {
6      int i;
7      while(b!=0)
8      {
9       i = b;
10      b = a%b;
11      a = i;
12     }
13     return a;
14 }
15
16
17 int NWW(int a, int b)        //Funkcja obliczająca NWW liczb a i b
18 {                           //wykorzystująca funkcje NWD
19     return a/NWD(a,b)*b;
20 }
21
22 int main()                   //Główna funkcja programu
23 {
24     int l1, l2, m1, m2, licznik, mianownik;
25
26
27     cout<<"Podaj licznik pierwszego ułamka  ";
28     cin >> l1;
29     cout << "Podaj mianownik pierwszego ułamka  ";
30     cin >> m1;
31
32     cout<<"Podaj licznik drugiego ułamka  ";
33     cin >> l2;
34     cout << "Podaj mianownik drugiego ułamka  ";
35     cin >> m2;
36
37     mianownik = NWW(m1, m2);     //mianownik wyniku jest równy NWW mianowników dodawanych ułamków
38     licznik = (mianownik/m1)*l1 + (mianownik/m2)*l2; //Obliczanie sumy rozszerzeń liczników
39
40     cout<<l1<<"/"<<m1<<" + "<<l2<<"/"<<m2<<" = "<<licznik<<"/"<<mianownik;
41
42     return 0;
43 }
```

Rys. 20.5. Przykładowy program z zastosowaniem funkcji

Po dołączeniu standardowej biblioteki (linia 1) i określeniu przestrzeni zmiennych (linia 2) następuje zdefiniowanie funkcji obliczającej NWD, a następnie funkcji obliczającej NWW. Jak widać w linii 19, wewnątrz funkcji została użyta funkcja NWD zdefiniowana wcześniej.

W linii 37 i 38 następuje obliczenie sumy ułamków. Do otrzymania wyniku jest używana funkcja NWW.

Porównaj kody obu programów. W drugim został wprowadzony ład. Nie ma już problemu z analizą programu. Przykładowo w pierwszym kodzie to zmienna a przyjmuje wartość NWD i dalej jest pod tą nazwą używana. Mało doświadczonemu programiście może być trudno odgadnąć jej znaczenie poza pętlą obliczania NWD. W drugim kodzie NWD jest obliczane przez funkcję wywoływaną w tym samym miejscu, w którym była zmienna a. Ponieważ jej przeznaczenie jest znane, analiza programu staje się łatwiejsza.

Zastosowanie funkcji ułatwia modyfikację lub poprawianie kodu. Dzięki użyciu funkcji ewentualne zmiany w jej kodzie nie pociągają za sobą konieczności zmian w części kodu, w którym ma ona być wykonana.

## ZADANIA DO WYKONANIA

1. Zmodyfikuj algorytm dodawania ułamków w taki sposób, by obliczał sumę trzech ułamków. Ułóż go tak, by ułatwić programiście użycie funkcji obliczających NWW i NWD.
2. Zaproponuj inny sposób na wprowadzenie zmiennych **m1**, **m2**, **l1** i **l2** do programu dodającego ułamki. Może ona polegać na użyciu innych komunikatów, zmodyfikowanym wyświetlaniu wprowadzonych danych itp.
3. Sprawdź działanie programu dodającego ułamki ujemne. Dokonaj obliczeń samodzielnie, a następnie porównaj je z wynikami działania programu. Wyjaśnij, dlaczego program działa także dla liczb ujemnych. Czy można powiedzieć, że ten program odejmuje ułamki?

4*. Ułóż algorytm dodający dowolną, ustaloną liczbę ułamków.

5*. Ułóż program dodający dowolną liczbę ułamków. Zdefiniuj w tym celu odpowiednią funkcję.

## PODSUMOWANIE LEKCJI

- **Algorytm dodawania ułamków** wykorzystuje algorytm znajdowania NWW i NWD. Jeśli znana jest NWW, dodawanie ułamków nieuwzględniające wyłączania części całkowitej nie wymaga badania warunków. ◂ s. 131
- **Funkcja** to fragment programu, któremu nadano nazwę. Dzięki temu można ją wielokrotnie wykorzystać w kodzie programu, bez konieczności każdorazowego wpisywania odpowiadającego jej kodu. **Użycie funkcji** w programie ułatwia jego analizę, późniejszą modyfikację i ogranicza wielkość kodu. ◂ s. 133

# 21. Skracamy i wyłączamy, czyli dodawania ułamków ciąg dalszy

**NA TEJ LEKCJI:**

- ułożysz program upraszczający ułamki;
- ułożysz program wyłączający z ułamka całości.

Twój program dodający ułamki już działa. Niestety wyniki podaje w postaci nieuproszczonej. Niektóre przekraczają także 1, czyli da się z nich wyłączyć całości. Nie zawsze jest to wskazane w dalszych obliczeniach, jednak wynik wyświetlany z wyłączeniem całości i w postaci uproszczonego ułamka będzie dla użytkownika programu atrakcyjniejszy. Zmodyfikujmy więc program.

## 21.1. Skracanie ułamków, czyli zmiana postaci wyniku działania programu dodawania

Program dodający dwa ułamki oblicza licznik i mianownik sumy. Przy czym mianownik jest najmniejszą wspólną wielokrotnością mianowników liczb dodawanych. Po wykonaniu obliczeń może się okazać, że wynik (ułamek) można uprościć. Umiesz już to robić. Jak zmodyfikować program, by dawał wynik w postaci uproszczonego ułamka?

Aby skrócić postać ułamka, należy znaleźć największy wspólny dzielnik licznika i mianownika. Kolejnym ruchem jest wykonanie dzielenia licznika i mianownika przez ich NWD.

modyfikacja programu a funkcje

W drugiej wersji programu dodającego ułamki pojawiły się funkcje. Tym razem funkcja obliczająca NWD będzie wykorzystana wielokrotnie – do wykonania podstawowego zadania, czyli dodania ułamków, oraz uproszczenia postaci wyniku.

```
1 #include<iostream>
2 using namespace std;
3
4 int NWD(int a, int b)        //Funkcja obliczająca NWD
5 {
6     int i;
7     while(b!=0)
8     {
9      i = b;
10     b = a%b;
11     a = i;
12    }
13    return (a);
14 }
15
16
17 int NWW(int a, int b)        //Funkcja obliczająca NWW
18 {
19    return a/NWD(a,b)*b;
20 }
21
22 int main()                   //Główna funkcja programu
23 {
24    int i, l1, l2, m1, m2, licznik, mianownik;
25
26
27    cout<<"Podaj licznik pierwszego ułamka  ";
28    cin >> l1;
29    cout << "Podaj mianownik pierwszego ułamka  ";
30    cin >> m1;
31
32    cout<<"Podaj licznik drugiego ułamka  ";
33    cin >> l2;
34    cout << "Podaj mianownik drugiego ułamka  ";
35    cin >> m2;
36
37    mianownik = NWW(m1, m2);
38    i = licznik = mianownik/m1*l1 + mianownik/m2*l2;
39    licznik = licznik / NWD(i,mianownik);
40    mianownik = mianownik / NWD(i,mianownik);
41
42    cout<<l1<<"/"<<m1<<" + "<<l2<<"/"<<m2<<" = "<<licznik<<"/"<<mianownik;
43
44    return 0;
45 }
```

Rys. 21.1. Przykładowy program dodający ułamki i podający uproszczony wynik

Do testu poprawności działania użyjmy tych samych liczb.

```
Podaj licznik pierwszego ułamka  12
Podaj mianownik pierwszego ułamka  18
Podaj licznik drugiego ułamka  7
Podaj mianownik drugiego ułamka  12
12/18 + 7/12 = 45/36
```

```
Podaj licznik pierwszego ułamka  12
Podaj mianownik pierwszego ułamka  18
Podaj licznik drugiego ułamka  7
Podaj mianownik drugiego ułamka  12
12/18 + 7/12 = 5/4
```

Rys. 21.2. Porównanie wyników działania programów dodających ułamki

Do programu dodającego ułamki dodano dwie linie (39 i 40). Są one zgodne z założeniami poczynionymi powyżej, jednak niedosłownie. Dlaczego?

W działaniach użyto zmiennej `i`. Przeanalizuj jej rolę w programie. Zauważ, że przechowuje ona licznik pierwotnego wyniku. Co by się stało, gdyby nie została zastosowana (rys. 21.2.)?

```
37      mianownik = NWW(m1, m2);
38      licznik = mianownik/m1*l1 + mianownik/m2*l2;
39      licznik = licznik / NWD(licznik,mianownik);
40      mianownik = mianownik / NWD(licznik,mianownik);
41
42      cout<<l1<<"/"<<m1<<" + "<<l2<<"/"<<m2<<" = "<<licznik<<"/"<<mianownik;
```

```
Podaj licznik pierwszego ułamka  12
Podaj mianownik pierwszego ułamka  18
Podaj licznik drugiego ułamka  7
Podaj mianownik drugiego ułamka  12
12/18 + 7/12 = 5/36
```

zmienna pomocnicza

Rys. 21.3. Nieprawidłowy wynik działania programu bez zmiennej pomocniczej *i*

### Analiza

W liniach 37 i 38 (rys. 21.3.) następuje dodanie danych ułamków. Linia 39 daje uproszczenie licznika. Od tej chwili zmienna `licznik` ma nową wartość. Obliczanie na jej podstawie NWD dla uproszczenia mianownika jest więc błędem.

Czy można inaczej ułożyć fragment programu odpowiedzialny za uproszczenie postaci wyniku? W programie z rysunku 21.1. funkcja NWD we fragmencie upraszczającym wynik jest wywoływana dwukrotnie. Można użyć jej tylko jeden raz, podstawiając wynik jej działania pod zmienną `i` (rys. 21.4.).

```
37      mianownik = NWW(m1, m2);
38      licznik = mianownik/m1*l1 + mianownik/m2*l2;
39      i=NWD(licznik, mianownik);
40      licznik = licznik / i;
41      mianownik = mianownik / i;
```

Rys. 21.4. Fragment programu z jednokrotnym wywołaniem funkcji NWD

Po takim działaniu linie 40 i 41 nie zawierają już wywołania funkcji, a jedynie używają zmiennej `i`. Z punktu widzenia czasu wykonywania programu jest to rozwiązanie korzystniejsze. Tracimy jednak na czytelności kodu.

## 21.2. Wyłączamy całości, czyli wynik w ostatecznej postaci

Wynik w postaci ułamka może być przydatny, gdy używamy go do dalszych obliczeń. Jeśli jest on ostatecznym wynikiem wyświetlanym użytkownikowi, ciekawiej byłoby wyłączyć z niego całości. Jak to zrobić bez użycia komputera?

1. Podziel licznik przez mianownik.
2. Zbadaj, czy część całkowita wyniku jest różna od 0. Jeśli tak, zapisz ją jako całkowitą część badanego ułamka.
3. Licznik ułamka zmniejsz o iloczyn części całkowitej i mianownika.

Zrealizujmy ten algorytm, dołączając go do kodu programu dodającego ułamki (rys. 21.5.).

```
int main()                  //Główna funkcja programu
{
    int c, i, l1, l2, m1, m2, licznik, mianownik;

    cout<<"Podaj licznik pierwszego ułamka  ";
    cin >> l1;
    cout << "Podaj mianownik pierwszego ułamka  ";
    cin >> m1;

    cout<<"Podaj licznik drugiego ułamka  ";
    cin >> l2;
    cout << "Podaj mianownik drugiego ułamka  ";
    cin >> m2;

    mianownik = NWW(m1, m2);
    licznik = mianownik/m1*l1 + mianownik/m2*l2;
    i=NWD(licznik, mianownik);
    licznik = licznik / i;
    mianownik = mianownik / i;

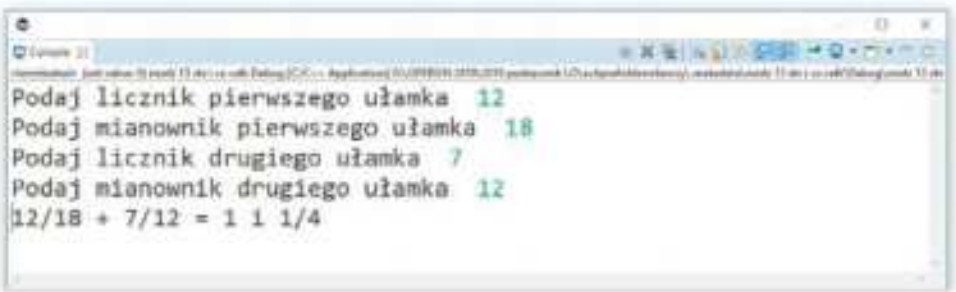
    if (licznik/mianownik != 0)
    {
        c=licznik/mianownik;
        licznik=licznik-c*mianownik;
        cout<<l1<<"/"<<m1<<" + "<<l2<<"/"<<m2<<" = " <<c <<" i "<<licznik<<"/"<<mianownik
    }
    else
        cout<<l1<<"/"<<m1<<" + "<<l2<<"/"<<m2<<" = "<<licznik<<"/"<<mianownik;

    return 0;
}
```

Rys. 21.5. Funkcja główna programu zawierającego wyłączenie z wyniku dodawania ułamków części całkowitej

zmienna zadeklarowana jako całkowita

Instrukcja `if` została użyta jedynie w celu zapewnienia właściwego wydruku wyniku. Bez badania tego warunku program w miejscu części całkowitej mógłby wydrukować 0 dla ułamków mniejszych od 1. Nie ma potrzeby szukania części całkowitej z dzielenia licznika przez mianownik, ponieważ zmienna `c` jest zadeklarowana jako całkowita, więc wynik dzielenia nie będzie zawierał części ułamkowej. Dla wyników mniejszych od 1 `c=0`. Sprawdźmy poprawność działania dla naszych danych testowych (rys. 21.6.).




Rys. 21.6. Wynik działania programu z wyłączeniem części całkowitej z wyniku dodawania ułamków

## ZADANIA DO WYKONANIA

1. Na podstawie analizy programu z podręcznika wyjaśnij znaczenie i celowość tworzenia funkcji.
2. Zmodyfikuj program dodawania ułamków w taki sposób, by podawał wynik w postaci równości, w której po lewej stronie zostanie wyświetlony ułamek przed uproszczeniem, a po prawej – skrócony.
3. Po wykonaniu zadania 2. dodaj do wydruku wynik z wyłączoną częścią całkowitą.
4. Zbadaj działanie programu z rozdziału w wersji ostatecznej dla różnych ułamków. Sprawdź, jak jest wyświetlany ujemny wynik. Zmodyfikuj wyświetlanie w taki sposób, by znak „–" został wyświetlony przed ostatecznym wynikiem, np. –1 i 3/4.

5*. Zmodyfikuj program z rysunku 21.6. w taki sposób, by można było dodawać trzy ułamki. Przedstaw sposób takiej modyfikacji dla dowolnej liczby ułamków.

## PODSUMOWANIE LEKCJI

s. 136 ▸ • Łatwa **modyfikacja programu** jest możliwa dzięki użyciu funkcji.

s. 138 ▸ • **Zmienna pomocnicza** pozwoliła uniknąć błędnego obliczenia wyniku.

s. 139 ▸ • Wynik dzielenia zapisany pod **zmienną** zadeklarowaną jako **całkowita** zawiera jedynie część całkowitą tego działania.

# 22. Szyfr Cezara, czyli poznajemy szyfr przesuwający

**NA TEJ LEKCJI:**

- poznasz prosty szyfr przesuwający, czyli szyfr Cezara;
- poznasz szyfr wieloalfabetowy,
- dowiesz się, jak ważne jest szyfrowanie informacji.

Nie wiemy, kto pierwszy zaszyfrował informacje. Wiemy za to, w jakim celu to zrobił – nie chciał, by zostały one odczytane przez osoby do tego nieuprawnione. Z różnymi sposobami szyfrowania można się spotkać w powieściach lub filmach. Każdy może wymyślić własny szyfr. Może on być bardzo skomplikowany, gdy użyjesz nietypowych, choć prostych rozwiązań. Ale zanim zaczniesz szyfrować własnym szyfrem, poznaj szyfr, z którego według niektórych źródeł jako pierwszy skorzystał Juliusz Cezar. Stąd nazwa – szyfr Cezara.

## 22.1. Prosty szyfr przesuwający, czyli Cezar szyfruje informacje

Wyobraź sobie, że przechwytujesz tajną informację, na przykłada taką:

◂ szyfr Cezara

```
v c b i u f h c d u d
```

Zdajesz sobie sprawę, że do jej rozszyfrowania potrzebujesz klucza. Klucz może być wskazówką, wzorem lub opisem metody rozszyfrowania tego tekstu. Komunikat składa się z liter alfabetu łacińskiego, ale jego znaczenie jest zagadką. A może rozwiązanie tego problemu wcale nie jest tak trudne, na jakie wygląda? Wcielmy się w rolę kryptologa, przed którym stanęło zadanie rozszyfrowania informacji. Zacznijmy od wypisania do tabeli pierwszych kilku liter szyfru (tab. 22.1.). W sąsiednich kolumnach wpiszmy litery występujące w alfabecie w sąsiedztwie znaków szyfru. Dla ułatwienia wypisz obok wszystkie litery alfabetu łacińskiego:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

Co zrobić, gdy w jednym lub drugim kierunku zostanie litera? Zapętl alfabet, zakładając, że po literze `z` wystąpi `a` oraz przed `a` wystąpi `z`.

Tab. 22.1. Tabela pomocna w odszyfrowaniu szyfru przesuwającego

| 3 litery w lewo | 2 litery w lewo | 1 litera w lewo | Zaszyfrowana informacja | 1 litera w prawo | 2 litery w prawo | 3 litery w prawo |
|---|---|---|---|---|---|---|
| s | t | u | **v** | w | x | y |
| z | a | b | **c** | d | e | f |
| y | z | a | **b** | c | d | e |
| f | g | h | **i** | j | k | l |
| r | s | t | **u** | v | w | x |
| c | d | e | **f** | g | h | i |
| e | f | g | **h** | i | j | k |
| z | a | b | **c** | d | e | f |
| a | b | c | **d** | e | f | g |
| r | s | t | **u** | v | w | x |
| a | b | c | **d** | e | f | g |

Teraz pozostaje analiza otrzymanych wyników. Szybko okaże się, że sensowne wyrazy otrzymasz już w pierwszej kolumnie, w której każdą z liter kodu zastąpiono literą poprzedzającą ją o **trzy pozycje** w alfabecie. Tak właśnie postąpiono w czasach Juliusza Cezara.

Jak opiszesz klucz potrzebny do **zaszyfrowania informacji**? Pomoże w tym diagram (rys. 22.1.).

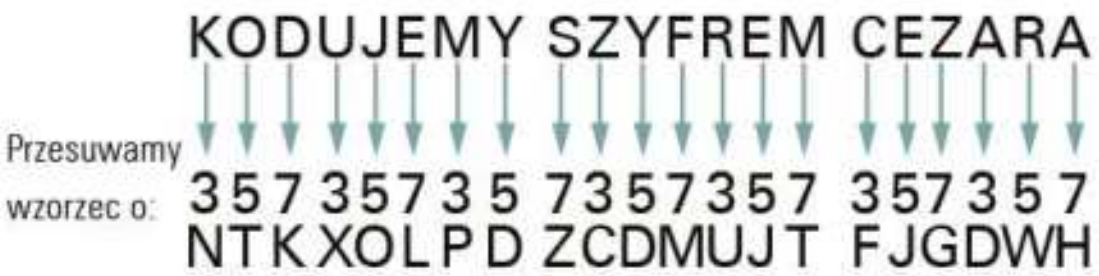



Rys. 22.1. Diagram ilustrujący metodę szyfrowania szyfrem Cezara

Wniosek: każdą z liter oryginalnej wiadomości należy zastąpić trzecią literą następującą po niej w alfabecie. Proces deszyfracji przebiega odwrotnie. Jeśli każdą literę zastąpimy literą, której pozycja w alfabecie jest oddalona o pewną stałą liczbę (np. o 3), to tę liczbę można nazwać kluczem szyfrowania.

Czy istnieją jeszcze inne szyfry oparte na metodzie przesuwania znaków? Oczywiście, że tak. Ty także możesz taką opracować, zmieniając liczbę lub kierunek podstawień. Ponieważ ta metoda jest dość znana, mogłaby być nieskuteczna. Dlatego można ją udoskonalić, tworząc **szyfr przesuwający wieloalfabetowy**. W jaki sposób? Czy liczba podstawień musi zawsze być taka sama? Oczywiście, że nie. Mogą to być na przykład liczby pierwsze 3, 5 i 7. W tym przypadku odnalezienie 1 litery odkodowanej wiadomości przebiegałoby jak w szyfrze Cezara. Następną znaleźć można nie na trzeciej, a na piątej pozycji, kolejną na siódmej, a czwartą – ponownie na trzeciej (rys. 22.2.).

szyfr wieloalfabetowy




Zmiana szyfru następuje wraz ze zmianą pozycji znaku:

– pierwszy znak szyfrujemy, przesuwając wzorzec o **3 znaki** w prawo;

– drugi znak szyfrujemy, przesuwając wzorzec o **5 znaków** w prawo;

– trzeci znak szyfrujemy, przesuwając wzorzec o **7 znaków** w prawo;

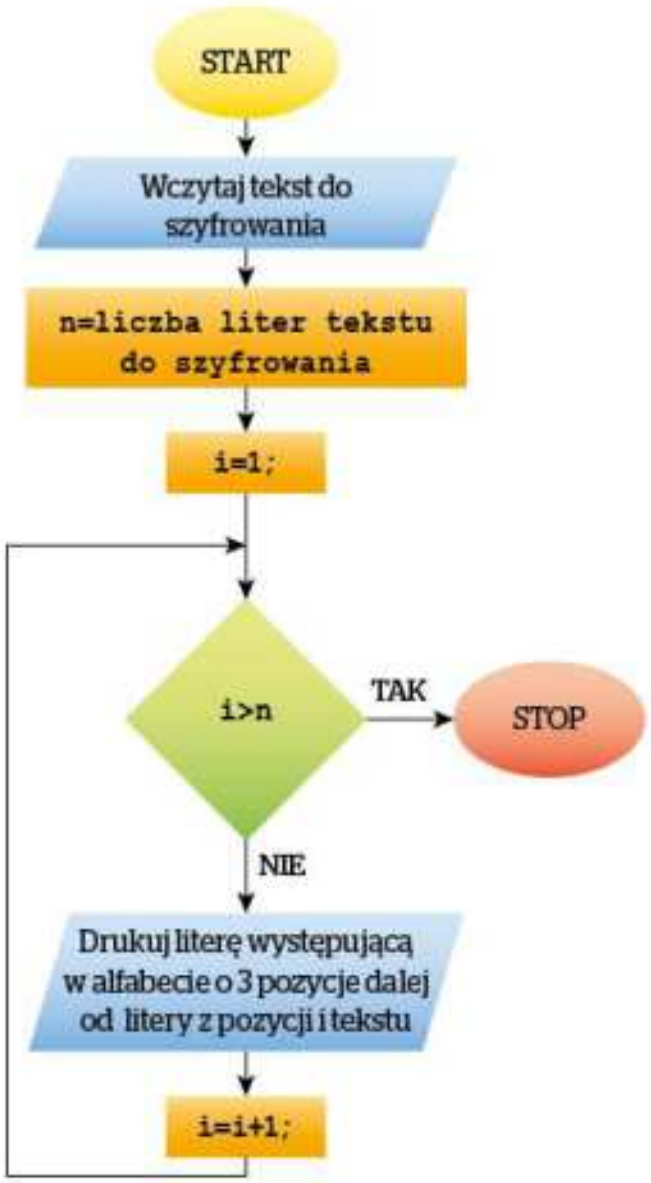
– czwarty znak szyfrujemy, przesuwając wzorzec o **3 znaki** w prawo.

Rys. 22.2. Szyfrowanie przykładowym szyfrem wieloalfabetowym

W powyższych przykładach nie stosuje się przerw między wyrazami, ponieważ to ułatwiłoby deszyfrację. Nie korzysta się także z polskich znaków, a ponadto nie rozróżnia się dużych i małych liter.

## 22.2. Szyfr przesuwający na schemacie, czyli tworzymy algorytm

Algorytm szczegółowy tej metody ułożony z myślą o realizacji w języku programowania musiałby opisywać sposób zapisywania liter w zmiennej tablicowej o rozmiarze zgodnym z długością tekstu. Musisz nabyć więcej umiejętności, by sprostać temu zadaniu. Nic nie stoi na przeszkodzie, byś poznał istotę szyfru Cezara (rys. 22.3.).




Rys. 22.3. Algorytm opisujący metodę szyfrowania szyfrem Cezara

**szyfrowanie informacji**

Szyfrowanie informacji ma wielkie znaczenie dla użytkowników sieci. Oczywiście nie korzysta się z tak prymitywnych metod jak szyfr Cezara. Szyfrowanie, choć ważne w zastosowaniach indywidualnych, ma ogromne znaczenie dla instytucji odpowiedzialnych za bezpieczeństwo danych. Szyfrowane są nie tylko informacje, lecz także transmisje sieciowe. Masz z tym do czynienia za każdym razem, gdy logujesz się do swojej poczty lub na konto dowolnej usługi. Skąd o tym wiadomo? Zwróć uwagę na symbole, które pokazują się obok adresu takiej usługi (rys. 22.4.).

**HTTPS**




Rys. 22.4. Oznaczenie połączenia szyfrowanego dla strony internetowej

## ZADANIA DO WYKONANIA

1. Opracuj własny szyfr przesuwający oparty na kluczu wybierającym kolejne litery. Zaszyfruj nim tekst „Szyfrszefa". Narysuj w zeszycie schemat blokowy opisujący twoją metodę szyfrowania.
2. Zastosuj dwukrotnie szyfr Cezara do zaszyfrowania tekstu „czytaniejestsuper". Oznacza to, że będziesz musiał/musiała po jednokrotnym szyfrowaniu powtórzyć czynności dla już zakodowanej informacji. Sprawdź doświadczalnie, czy da to taki sam efekt, jak szyfrowanie ze skokiem 6 zamiast 3, jak w szyfrze Cezara. Wyciągnij wnioski i je uzasadnij. Postaraj się uogólnić opis tego przypadku na inne, podobne.
3. Czy algorytm z rysunku 22.3. może służyć do opisania dowolnej metody opartej na stosowanej w szyfrze Cezara? Jeśli nie, to co należałoby zmienić?
4. W szyfrze Cezara zastosowano skok o 3 litery. Dla jakiej liczby skoków szyfrowanie nie miałoby sensu? Do jakiej liczby skoków szyfrowanie tą metodą ma sens? Jakie liczby skoków dałyby takie same efekty szyfrowania?

5**. Ułóż algorytm do napisania programu szyfrującego metodą szyfru Cezara. Dla ułatwienia zastosuj stałą długość kodowanego tekstu.

## PODSUMOWANIE LEKCJI

- Jedną z metod podstawieniowych szyfrowania informacji jest **szyfr Cezara**, w którym każdą literę w informacji zastępuje się literą znajdującą się w alfabecie łacińskim o trzy pozycje dalej. Przykładowo: a ->d. ◂ s. 141
- Istnieje nieskończona ilość tworzenia metod szyfrowania **metodą przesuwającą,** ponieważ można korzystać z różnych sposobów wybierania kolejnej litery, np. co czwartą lub w odległościach opisanych kluczem, np. 3, 5, 7. Mamy wtedy do czynienia z **szyframi wieloalfabetowymi.** ◂ s. 143
- **Szyfrowanie skomplikowanymi metodami** ma wielkie znaczenie dla bezpieczeństwa danych. ◂ s. 144
- **https://** przed adresem oznacza szyfrowanie transmisji dla tej strony, np. podczas logowania się do konta pocztowego lub konta usługi. ◂ s. 144

# 23. Konspiracja, czyli tworzymy własne szyfry

**NA TEJ LEKCJI:**

- zrozumiesz, jaką rolę odgrywają klucze w szyfrowaniu informacji,
- dowiesz się, czym są kryptografia symetryczna i asymetryczna;
- ułożysz własny szyfr z kluczem generowanym losowo.

Kryptografia to jedna z ciekawszych dziedzin nauki. Zajmują się nią naukowcy na najsłynniejszych uniwersytetach świata. Skuteczne szyfrowanie informacji stało się jednym z najważniejszych zagadnień współczesnej informatyki. Domowy użytkownik sieci korzysta z szyfrowanych połączeń na przykład z bankiem. Rzadziej szyfruje swoje listy. Jednak szyfrowanie korespondencji i przesyłania danych pomiędzy firmami, bankami, politykami itp. jest już normą – im skuteczniejsze sposoby, tym bezpieczniejsze dane. Czy zatem uczeń może stworzyć własny, niepowtarzalny kod ze zmiennym kluczem? Zmierzmy się z tym problemem.

## 23.1. Zamknij – otwórz, czyli jaką rolę odgrywają klucze w szyfrowaniu informacji

Do otwarcia każdego zamka potrzebny jest klucz. Nie ma znaczenia, czy to zamek mechaniczny czy elektroniczny. Klucz jest niezbędny. Dopóki znają go wyłącznie upoważnione osoby, informacja jest bezpieczna.

klucz w szyfrowaniu

**Klucz** w szyfrowaniu stanowi informację, dzięki której można zaszyfrować lub odszyfrować informację właściwą lub dane.

kryptografia symetryczna

Algorytmy symetryczne do szyfrowania i odszyfrowywania wykorzystują klucze znane nadawcy i odbiorcy. Mówimy wtedy, że mamy do czynienia z **kryptografią symetryczną** (rys. 23.1.). Zazwyczaj dla zwiększenia bezpieczeństwa dla każdego połączenia generuje się nowy klucz. Należy więc znaleźć sposób na przekazanie go odbiorcy.

kryptografia asymetryczna

Do tego można użyć metody **asymetrycznej**, która jest bezpieczniejsza, ale wymaga skomplikowanych obliczeń. Dlatego stosuje się ją najczęściej do szyfrowania krótkich informacji.




Rys. 23.1. Schemat szyfrowania symetrycznego

W szyfrowaniu asymetrycznym używa się **kluczy publicznych i prywatnych**. Klucz prywatny pełni funkcję swoistego podpisu nadawcy i powinien być znany tylko jemu. Klucz prywatny i publiczny stanowią unikatową parę (rys. 23.2.).

szyfrowanie asymetryczne

Funkcje kluczy mogą być łączone, na przykład informacja najpierw jest podpisywana kluczem prywatnym nadawcy, a potem szyfrowanym kluczem publicznym odbiorcy. Po jej odebraniu program deszyfrujący odwraca kolejność działań. Odszyfrowuje informację kluczem prywatnym odbiorcy, a następnie sprawdza zgodność sygnatury z kluczem publicznym autora. Klucze prywatne i publiczne są tworzone na różne sposoby – im bardziej wyrafinowany, tym klucz jest trudniejszy do odgadnięcia Jeśli zainteresował cię temat kryptografii, możesz dowiedzieć się więcej z dostępnych publikacji.




Rys. 23.2. Schemat szyfrowania asymetrycznego

Czy zatem każdy jest w stanie utworzyć własny sposób szyfrowania informacji? Zawsze można opracować sposób oparty na kryptografii symetrycznej z jednym kluczem do szyfrowania i deszyfrowania, lecz zmieniającym się na przykład losowo. Taki klucz będzie musiał znać także odbiorca. Do dzieła.

## 23.2. Klucz w tablicy, czyli szyfrujemy skuteczniej od Cezara

W przypadku szyfru Cezara kluczem była liczba wyznaczająca znak do podstawienia za oryginalną literę. Wykorzystajmy ten pomysł, jednak zamiast stałego skoku użyjmy zmiennego.

**tablica kodów ASCII**

Ponadto zamiast liter z tekstu wykorzystamy wszystkie znaki z 256-znakowej tablicy kodów ASCII (informacje o kodach ASCII przyporządkowanych poszczególnym znakom i literom powinieneś/powinnaś znać ze szkoły podstawowej; jeśli ich nie pamiętasz, skorzystaj z wiadomości w internecie). Oznacza to, że litery w tablicy zawierającej tekst do szyfrowania będą traktowane jako odpowiadające im kody, czyli liczby. Do nich zostanie dodany klucz, który z powrotem będzie wyświetlony jako tekst (tab. 23.1.). Taka operacja prowadzi do zaszyfrowania tekstu. Deszyfracja polega na odejmowaniu wartości klucza (tab. 23.2.).

Zaszyfrujmy tym sposobem wyraz „Dom".

Tab. 23.1. Przykład szyfrowania

| Litera tekstu | ASCII | Klucz | ASCII po dodaniu klucza | Zaszyfrowany wyraz |
|---|---|---|---|---|
| D | 68 | 12 | 80 | P |
| o | 111 | 13 | 124 | \| |
| m | 109 | 5 | 114 | r |

Tab. 23.2. Przykład deszyfracji

| Litera tekstu | ASCII | Klucz | ASCII po dodaniu klucza | Zaszyfrowany wyraz |
|---|---|---|---|---|
| P | 80 | 12 | 68 | D |
| \| | 124 | 13 | 111 | o |
| r | 114 | 5 | 109 | m |

Do zaszyfrowania i odszyfrowania danej litery użyjemy tego samego klucza. Dla każdego kolejnego znaku będzie on inny i zgodny z tablicą kluczy.

W pierwszym projekcie użyjemy stałego zestawu kluczy w postaci 20-elementowej tablicy z liczbami (rys. 23.3.).

```
1  #include <iostream>
2  #include <ctime>
3  using namespace std;
4
5  void szyfrowanie(char *tekst, int przesuniecie[])
6  {
7    int skok = 0;
8    for (int i=0; tekst[i] != '\0'; i++)
9    {
10     tekst[i] = ( tekst[i] + przesuniecie[skok] );
11     skok ++;
12     if(skok == 19) skok = 0;
13   }
14 }
15
```

Funkcja szyfrująca

```
16 void odszyfrowanie(char *tekst, int przesuniecie[])
17 {
18   int skok = 0;
19   for (int i=0; tekst[i] != '\0'; i++)
20   {
21    tekst[i] = ( tekst[i] - przesuniecie[skok] );
22    skok ++;
23    if(skok == 19) skok = 0;
24   }
25 }
26
27 int main()
28 {
29 int klucz[]={12,13,5,3,4,6,12,19,11,10,9,8,3,7,1,14,12,17,6,2};
30
31     cout<<endl;
32     char tesktdoszyfrowania[100];
33     cin.get ( tesktdoszyfrowania, 100);
34     cout << tesktdoszyfrowania <<endl;
35     szyfrowanie(tesktdoszyfrowania, klucz);
36     cout << tesktdoszyfrowania << endl;
37     odszyfrowanie(tesktdoszyfrowania, klucz);
38     cout << tesktdoszyfrowania;
39
40 return 0;
41 }
```

Funkcja deszyfrująca

Rys. 23.3. Przykładowy program szyfrujący i deszyfrujący ze stałym kluczem w postaci tablicy liczb

W przykładowym programie zbudowano dwie funkcje – jedna szyfruje, a druga odszyfrowuje informacje tekstowe. Ta konstrukcja pozwala na prześledzenie procesu kodowania i sprawdzenie poprawności działania. Oczywiście, jeśli ta metoda miałaby być wykorzystana, musiałyby istnieć dwa oddzielne programy wykorzystujące ten sam klucz. Założono także, że ciąg znaków, jaki tworzy informację, nie będzie dłuższy od 100. Oczywiście łatwo to zmienić w **liniach 32 i 33**.

Główną funkcję programu zaczyna utworzenie 20-elementowej tablicy z wpisanym kluczem. Są to przypadkowo ułożone liczby zakresu od 0 do 19. Zauważ, że niektóre się powtarzają. Nie ma to znaczenia dla skuteczności szyfrowania.

**Linia 32** to deklaracja tablicy dla zmiennej wejściowej zawierającej tekst do zaszyfrowania. Kolejna linia to strumieniowanie znaków z klawiatury, czyli wprowadzanie tekstu, który ma być szyfrowany. Funkcja `cin.get` służy do wprowadzania znaków do zmiennej tablicowej. Wymaga podania nazwy zmiennej i maksymalnej liczby znaków. Użycie `cin` spowodowałoby wstawienie w miejscu spacji wartości `0`, co zakończyłoby pętlę szyfrowania.

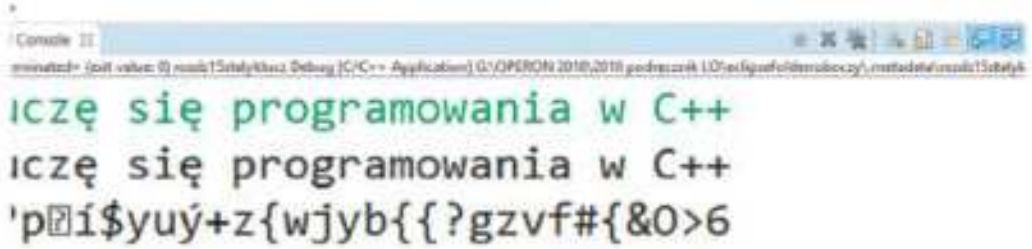
**Linia 34** to kontrolne wyświetlanie wprowadzonego tekstu. Oczywiście nie ma wpływu na szyfrowanie, ale w naszym przykładzie posłuży do porównania z tekstem po odszyfrowaniu.

**Linia 35** to wywołanie funkcji szyfrującej z przekazaniem wartości zmiennych. Pierwszą jest tablica z tekstem do szyfrowania, drugą – klucz szyfrujący.

**Linia 36** wysyła strumień z zakodowanym tekstem. W tym momencie zmienna tablicowa uprzednio zawierająca tekst do szyfrowania została zmieniona przez funkcję szyfrującą z zakodowaną informacją.

**Linia 37** wywołuje funkcję deszyfrującą. Program przekazuje jej tablicę z zaszyfrowanym tekstem i tablicę z kluczem.

**Linia 38** strumieniuje odszyfrowany tekst (zmienioną przez funkcję deszyfrującą zmienną tablicową) na ekran konsoli. Teraz można porównać tekst przed szyfrowaniem i po nim (rys. 23.4.).




Rys. 23.4. Wynik działania przykładowego programu szyfrującego i deszyfrującego ze stałym kluczem

zmienna tablicowa ▸ A jak działa sam mechanizm szyfrowania? Wewnątrz funkcji zmienna `tekst[]` jest wypełniona wartościami kodów ASCII znaków z szyfrowanego tekstu, a tablica `przesunięcie[]` przyjmuje wartości zmiennej tablicowej `klucz[]`. Pętla for wykonuje się do momentu, gdy zostanie wykryty koniec tekstu:

```
for (int i=0; tekst[i] != '\0'; i++)
```

Kluczowe jest tu użycie warunku `tekst[i] != '\0'`, czyli badanie, czy kolejny element tablicy nie jest pusty. A co w pętli? Do kolejnych kodów liter tekstu szyfrowanego zostaną dodane kolejne liczby z tablicy kluczy. Użyty warunek pozwala na wielokrotne, cykliczne użycie tablicy z kluczami w przypadku kodowania tekstu dłuższego niż 20 znaków. Funkcja zmienia elementy tablicy, w której wcześniej był tekst do szyfrowania. Teraz jej zawartość stanowi zaszyfrowany tekst.

Funkcja deszyfrująca jest bardzo podobna do szyfrującej. Ma przywrócić pierwotne wartości tablicy z tekstem, dlatego musi od aktualnych kodów ASCII odjąć te same klucze (liczby), które służyły do ich zakodowania.

## 23.3. Generujemy losowo, czyli dla każdego szyfrowania inny klucz

Skomplikujmy nieco życie złodziejom informacji i zamiast stałej, zaszytej w kodzie programu tablicy kluczy użyjmy takiej, która przy każdorazowym uruchomieniu szyfrowania będzie zawierała inne klucze.

Czy to możliwe? Jeśli użyjemy generatora liczb pseudolosowych, nie będzie to trudne. Zmodyfikujmy więc nasz program, dodając kod wypełniający przypadkowymi liczbami tablicę kluczy (rys. 23.5.).

```
 1 #include <iostream>
 2 #include <ctime>
 3 using namespace std;
 4
 5 void szyfrowanie(char *tekst, int przesuniecie[])
 6 {
 7   int skok = 0;
 8   for (int i=0; tekst[i] != '\0'; i++)
 9   {
10       tekst[i] = ( tekst[i] + przesuniecie[skok] );
11       skok ++;
12       if(skok == 19) skok = 0;
13   }
14 }
15 void odszyfrowanie(char *tekst, int przesuniecie[])
16 {
17   int skok = 0;
18   for (int i=0; tekst[i] != '\0'; i++)
19   {
20     tekst[i] = ( tekst[i] - przesuniecie[skok] );
21     skok ++;
22     if(skok == 19) skok = 0;
23   }
24
25 }
26 int main()
27
28 {
29 int klucz[20], i;
30 srand( time( NULL ) );
31 for (i=0; i<20; i++)
32 {
33 klucz[i]=((rand() % 19 ) + 0 );
34 cout << klucz[i] << " ";
35 }
36
37 cout<<endl;
38     char tekstdoszyfrowania[100];
39     cin.get ( tekstdoszyfrowania, 100);
40     cout << tekstdoszyfrowania <<endl;
41     szyfrowanie(tekstdoszyfrowania, klucz);
42     cout << tekstdoszyfrowania << endl;
43     odszyfrowanie(tekstdoszyfrowania, klucz);
44     cout << tekstdoszyfrowania;
45 return 0;
46
47 }
```

Rys. 23.5. Przykładowy program z losowo wypełnianą tablicą kluczy

Do generowania liczb losowych przydatnych w naszym zadaniu będzie potrzebna funkcja `time()`. Wymaga to dołączenia biblioteki `ctime`.

W miejscu deklaracji wypełnionej kluczami tablicy `klucz[]`, został dodany kod wypełniający tablicę o tej samej nazwie (linie od 28 do 35) losowymi liczbami. Dzięki zachowaniu nazw zmiennych nie trzeba modyfikować reszty kodu programu.

Jak działa wypełnianie tablicy losowo wybranymi liczbami?

```
int klucz[20], i;
srand( time( NULL ) );
for (i=0; i<20; i++)
{
 klucz[i]=((rand() % 19 ) + 0 );
 cout << klucz[i] << " ";
}
```

Wielkość tablicy kluczy pozostaje bez zmian, dlatego została zadeklarowana jako `klucz[20]`. Zmienna `i` jest potrzebna do sterowania pętlą, która wykonuje się 20 razy, za każdym razem wpisując do kolejnej pozycji tablicy (zależnej od `i`) losową wartość z przedziału 0 do 19. Skąd jednak liczby losowe?

**funkcja rand()** ▸ Liczby całkowite generuje funkcja `rand()`.

Aby uzyskać każdorazowo inną wartość, należy powiązać funkcję z czasem rzeczywistym wskazywanym przez zegar komputera. Stąd konieczność dołączenia biblioteki `ctime`. Dzięki niej będzie można użyć funkcji `time`, która zwraca czas zegara komputera jako liczbę. Użyta w funkcji `srand` ustawiającej wartość początkową liczby zapewni różne wartości dla każdej kompilacji programu Zakres liczb (`od 0 do 19`)zapewnia `%19 + 0`.

W przykładowym programie dodano także strumieniowanie do konsoli wygenerowanej tablicy kluczy. Pozwoli to zaobserwować losowość jej wypełniania i weryfikację poprawności szyfrowania i deszyfrowania.

```
9 4 2 4 12 0 13 13 4 1 14 12 1 0 5 9 11 12 11 17
uczę się programowania w C++
uczę się programowania w C++
~g|i,sv÷$q€{hrfvz?lwmc$? P8/
uczę się programowania w C++

13 13 5 5 0 2 9 4 18 10 15 2 0 10 16 8 14 1 9 13
uczę się programowania w C++
uczę się programowania w C++
,p▯d uri2z?qg|qu}xj{vf%w"L/=
uczę się programowania w C++

6 16 0 16 9 3 16 12 0 4 16 18 1 7 6 16 3 15 8 5
uczę się programowania w C++
uczę się programowania w C++
{szú)vyö t,?hyg}r†itya0€#S7+
uczę się programowania w C++
```

Rys. 23.6. Przykłady szyfrowania tekstu programem z zastosowaniem losowych kluczy

Dzięki losowości generowania tablicy kluczy odszyfrowanie tekstu bez jej znajomości jest bardzo trudne.

Obie metody zazwyczaj dają różne wartości dla tych samych znaków szyfrowanych. Wyjątek stanowi rzadki przypadek kodowania tej samej liczby tym samym kluczem.

Przykładowe programy, po szyfrowaniu, nie zmieniają liczby znaków.

## ZADANIA DO WYKONANIA

1. Sprawdź dla dowolnego tekstu poprawność metody szyfrowania użytej w przykładach. Wyniki przedstaw w tabelach wzorowanych na tabelach 23.1. i 23.2. Użyj tablicy kodów ASCII.
2. Zmodyfikuj program ze stałymi kluczami w taki sposób, by wyświetlał w konsoli kody ASCII zaszyfrowanych tekstów. Na podstawie tabeli kodów sprawdź, czy funkcja szyfrująca i deszyfrująca działają prawidłowo. Sprawdzenia dokonaj na trzech dowolnych znakach tekstu.
3. Zmodyfikuj program ze stałymi kluczami w taki sposób, by ograniczyć liczbę kluczy w tablicy do 5. Obserwuj wyniki szyfrowania i porównaj z wynikami dla tablicy 20-elementowej. Czy szyfrowanie stało się mniej bezpieczne?
4. W programie z losowo wypełnianą tablicą kluczy zmień sposób wyświetlania wyników w konsoli w taki sposób, by informacja była pełna. Na ich podstawie użytkownik powinien bez wątpliwości wskazywać i nazywać wyświetlane dane.
5. Zmodyfikuj program z losowo generowanymi kluczami w taki sposób, by szyfrowanie i deszyfrowanie odbywało się dwukrotnie przy użyciu tych samych kluczy. Czy zwiększy to bezpieczeństwo szyfrowanych danych?

## PODSUMOWANIE LEKCJI

- **Klucz w szyfrowaniu** jest informacją, dzięki której można zaszyfrować lub odszyfrować informację. ◂ s. 146
- **Kryptografia symetryczna** używa tego samego klucza do szyfrowania i deszyfrowania. Muszą go mieć zarówno nadawca, jak i odbiorca. ◂ s. 146
- **Kryptografia asymetryczna** używa kluczy publicznych i prywatnych. Klucz prywatny zna jedynie odbiorca. ◂ s. 146
- **Szyfrowanie asymetryczne** często jest stosowane do szyfrowania kluczy kryptografii symetrycznej. ◂ s. 147
- Teksty są przechowywane w postaci **kodów ASCII**. Teksty można szyfrować, dodając klucz w postaci liczby całkowitej do kodów ASCII poszczególnych znaków lub go odejmując od tych kodów. ◂ s. 148
- Program może przekazać do funkcji także **zmienną tablicową**. ◂ s. 150
- **Funkcja rand()** generuje całkowite liczby losowe. ◂ s. 152

# 24. Z kluczem publicznym, czyli bezpieczniejsze szyfrowanie

**NA TEJ LEKCJI:**

- poznasz zasadę działania najpopularniejszego algorytmu asymetrycznego, czyli RSA;
- dowiesz się, jakie są etapy działania algorytmu RSA.

Szyfrowanie ma istotne znaczenie w zapewnieniu bezpieczeństwie danych, ich przechowywaniu czy przesyłaniu. W dzisiejszych czasach, w których większość transakcji bankowych i komunikacja społeczna odbywa się on-line, spotykamy się z nim na każdym kroku. Logowanie do banku czy poczty musi być bezpieczne. W tym rozdziale zajmiemy się najpopularniejszym algorytmem kryptograficznym – RSA.

RSA ▸

## 24.1. RSA – najpopularniejszy algorytm asymetryczny z kluczem publicznym

W dzisiejszych czasach możemy wyróżnić trzy rodzaje szyfrów: asymetryczny, symetryczny blokowy i symetryczny strumieniowy.

Szyfr asymetryczny stosuje się głównie w szyfrowaniu poczty elektronicznej. Zasada jego działania jest dość prosta. W jego skład wchodzą dwa klucze – publiczny i prywatny. Klucz publiczny służy do zakodowania wiadomości, natomiast klucz prywatny – do jej deszyfracji. Klucza prywatnego nie udostępnia się osobom trzecim, więc możliwość jego złamania jest prawie niemożliwa. Głównym założeniem szyfru asymetrycznego są operacje bardzo trudne do odwrócenia, dzięki czemu jego skuteczność jest ogromna.

Przeciwieństwem klucza asymetrycznego jest klucz symetryczny, w którym klucz deszyfrujący jest wyznaczany z klucza szyfrującego.

My zajmiemy się szyfrem asymetrycznym, a dokładniej – algorytmem zaprojektowanym w 1977 r. przez Rona Rivesta (czytaj: riwesta), Adiego Shamira (czytaj: szamira) oraz Leonarda Adelmana. Od nazwisk tych twórców pochodzi nazwa algorytmu – algorytm RSA. To pierwszy, a zarazem najpopularniejszy i stosowany do dzisiaj algorytm do szyfrowania oraz do podpisów cyfrowych.

Budowa klucza publicznego w algorytmie RSA opiera się na trudności faktoryzacji, czyli rozkładu na czynniki pierwsze dużych liczb złożonych (składających się z ponad 100 cyfr). Gwarantuje to bezpieczeństwo szyfru, gdyż rozkład dużej liczby na czynniki pierwsze zajmuje wiele czasu.

## 24.2. Tworzymy szyfr za pomocą algorytmu RSA

trzy etapy działania algorytmu RSA

Aby stworzyć szyfr za pomocą algorytmu RSA, należy przejść przez trzy etapy prac:

1. Tworzenie klucza RSA, czyli wygenerowanie klucza publicznego i tajnego – ten pierwszy jest przekazywany wszystkim zainteresowanym i umożliwi zaszyfrowanie danych. Klucz tajny pozwala na deszyfrację danych zaszyfrowanych kluczem publicznym i jest trzymany w ścisłej tajemnicy.

Zabierzmy się zatem do pracy i stwórzmy klucze.

Krok 1: Znajdźmy dwie duże liczby pierwsze – najlepiej mające co najmniej 128 bitów. Oznaczmy je jako $p$ i $r$.

Krok 2: Obliczmy $F = (p - 1) * (q - 1)$ oraz $n = p * q$. $F$ to tzw. funkcja Eulera (funkcja przypisująca każdej liczbie naturalnej liczbę liczb względnie pierwszych z nią i nie większych od niej).

Krok 3: Wykorzystując odpowiednio algorytm Euklidesa, znajdźmy liczbę $e$, która jest względnie pierwszą z wyliczoną wartością funkcji Eulera $F$ (NWD($e$, $F$) = 1). Liczba ta powinna również spełniać nierówność $1 < e < n$. Nie musi być ona pierwsza, ale musi być nieparzysta.

Krok 4: Obliczmy liczbę odwrotną modulo $F$ do liczby $e$, czyli spełniającą równanie $d * e \bmod F = 1$. Znajdujemy liczbę $d$, której różnica z odwrotnością modularną liczby $e$ jest podzielna przez $n$. Inaczej mówiąc – reszta z dzielenia $d$ przez $m$ i $e$ przez $m$ musi być taka sama, na przykład: $5 \equiv 3 \pmod 2$, bo $5 \bmod 2 = 1$ i $3 \bmod 2 = 1$.

Krok 5: Klucz publiczny jest parą liczb ($e$, $n$); $e$ nazywamy publicznym wykładnikiem. Można go przekazywać wszystkim zainteresowanym.

Krok 6: Klucz tajny to para liczb ($d$, $n$); $d$ nazywamy prywatnym wykładnikiem. Klucz ten należy przechowywać pod ścisłym nadzorem.

**Przykład:**

Wybieramy dwie liczby:
$p = 13, q = 11$

Obliczmy $F$:
$F = (p - 1) * (q - 1)$, a zatem $F = (13 - 1) * (11 - 1) = 12 * 10 = 120$
$F = 120$

Obliczmy moduł $n$:
$n = p * q = 13 * 11 = 143$

Wyznaczamy wykładnik publiczny $e$, ma być on względnie pierwszy z $F$, czyli z liczbą 120 – warunek ten spełnia liczba 7.

Wyznaczamy wykładnik prywatny, który ma być odwrotnością modulo $F$ liczby $e$, czyli:

$d * 7 \bmod 120 = 1$

Liczbą spełniającą ten warunek jest liczba 103.

Klucz publiczny $(e, n) – (7, 143)$
Klucz tajny $(d, n) – (103, 143)$

**2.** Szyfrowanie kluczem publicznym RSA – użytkownik po otrzymaniu klucza publicznego koduje za jego pomocą swoje dane i przesyła w postaci szyfru RSA do adresata, który posiada klucz prywatny umożliwiający deszyfrację.

Krok 7: Otrzymanie od adresata klucza publicznego w postaci pary liczb $(e, n)$.

Krok 8: Wiadomość do zaszyfrowania zamieniamy na liczby naturalne $t$, które muszą spełniać nierówność $0 < t < n$.

Krok 9: Na tak otrzymanych liczbach wykonujemy operację szyfrowania i otrzymujemy liczby $c = t^e \bmod n$.

Krok 10: Liczby $c$ są zaszyfrowaną postacią liczb $t$ i przekazuje się je adresatowi wiadomości. Klucz $(e, n)$ umożliwił ich zaszyfrowanie, lecz nie pozwala na ich deszyfrację.

**Przykład:**

Otrzymaliśmy klucz publiczny (7, 143). Za jego pomocą możemy zakodować liczby od 0 do 142 – liczby 0 i 1 nie zostaną zaszyfrowane.

Załóżmy, że chcemy przesłać adresatowi zaszyfrowaną liczbę 123. W tym przypadku musimy obliczyć wartość wyrażenia:

$c = 123^7 \bmod 143 = 425927596977747 \bmod 143 = 7$

$c = 7$ – jest to zaszyfrowana liczba 123, którą przesyłamy do adresata.

**3.** Rozszyfrowywanie kluczem prywatnym RSA – adresat po otrzymaniu wiadomości deszyfruje ją za pomocą klucza prywatnego.

Krok 11: Otrzymaliśmy zakodowaną wiadomość w postaci liczb naturalnych $c$, które muszą spełniać warunek $0 < c < n$, wraz z wcześniej wygenerowanym kluczem szyfrującym $(e, n)$.

Krok 12: Liczbę $c$ przekształcamy na pierwotną wartość $t$, stosując wzór $t = c^d \bmod n$.

Krok 13: Z otrzymanej liczby $t$ odtwarzamy według ustalonego systemu znaki tekstu. Teraz możemy odczytać wiadomość.

**Przykład:**
Otrzymaliśmy zakodowaną wiadomość o wartości $c = 7$. Posiadamy klucz prywatny (103, 143), który posłuży do deszyfracji wiadomości.

Wykonujemy następujące działania:

$t = 7^{103} \bmod 143$

Potęga jest zbyt duża, aby można ją było w normalny sposób obliczyć (języki programowania mają zwykle ograniczenia co do wielkości liczb całkowitych). Jednakże nie interesuje nas wartość liczbowa potęgi, a jedynie reszta z dzielenia jej przez 143. Możemy więc rozłożyć potęgę na iloczyn składników o wykładnikach równych kolejnym potęgom liczby dwa:

$7^{103} \bmod 143 = (7^{64} \bmod 143) * (7^{32} \bmod 143) * (7^{4} \bmod 143) * (7^{2} \bmod 143) * * 7 \bmod 143$

$7^{1} \bmod 143 = 7$

$7^{2} \bmod 143 = (7^{1} \bmod 143)^2 \bmod 143 = 49 \bmod 143 = 49$

$7^{4} \bmod 143 = (7^{2} \bmod 143)^2 \bmod 143 = (49)^2 \bmod 143 = 113$

$7^{8} \bmod 143 = (7^{4} \bmod 143)^2 \bmod 143 = (113)^2 \bmod 143 = 42$

$7^{16} \bmod 143 = (7^{8} \bmod 143)^2 \bmod 143 = (42)^2 \bmod 143 = 48$

$7^{32} \bmod 143 = (7^{16} \bmod 143)^2 \bmod 143 = (48)^2 \bmod 143 = 16$

$7^{64} \bmod 143 = (7^{32} \bmod 143)^2 \bmod 143 = (16)^2 \bmod 143 = 113$

Do wyliczenia potęgi bierzemy tylko te reszty, które występują w sumie potęg 2 (jeśli byłoby ich bardzo dużo, to każde mnożenie można wykonać z operacją modulo, dzięki czemu wynik nigdy nie wyjdzie poza wartość modułu):

$t = 7^{103} \bmod 143 = 113 * 16 * 113 * 49 * 7 \bmod 143 = 123$

## ZADANIA DO WYKONANIA

1. Przeprowadź proces szyfrowania i deszyfracji dla liczb $p = 7$ i $q = 19$.

2*. Napisz program w języku Java umożliwiający wykonanie szyfrowania i deszyfracji za pomocą algorytmu RSA.

## PODSUMOWANIE LEKCJI

- **RSA** jest jednym z najpopularniejszych algorytmów szyfrowania z kluczem publicznym. Został zaprojektowany przez Rona Rivesta, Adiego Shamira i Leonarda Adelmana w 1977 r. ◂ s. 154
- **Algorytm RSA działa w trzech etapach**: tworzenia kluczy, szyfrowania kluczem publicznym i deszyfracji kluczem prywatnym. ◂ s. 155

# 25. Szyfrowanie zachłanne, czyli kody Huffmana

**NA TEJ LEKCJI:**

- dowiesz się, na czym polega metoda zachłanna i w jakim celu się ją stosuje;
- poznasz kody Huffmana oraz zdobędziesz informacje, jak je wykorzystywać.

Kodowanie Huffmana to jedna z najprostszych w implementacji (wdrożeniu) metod kompresji bezstratnej (czyli zmianie zapisu informacji zmniejszającej liczbę bitów, pod warunkiem, że metoda ta gwarantuje możliwość odtworzenia informacji do identycznej postaci pierwotnej). Została ona opracowana przez Amerykanina Davida Huffmana (czytaj: dejwida hafmana) w 1952 r.

## 25.1. Metoda zachłanna

metoda zachłanna ▸ Metoda zachłanna polega na dojściu do rozwiązania postawionego problemu w taki sposób, aby rozpatrywać dane w kolejności uporządkowanej, na przykład posortowane. W danym kroku wybiera się te dane, które w danym momencie są najodpowiedniejsze.

metoda optymalna ▸ Metoda ta prowadzi na ogół do rozwiązania przybliżonego, choć istnieją problemy, dla których właśnie ona jest metodą optymalną.

**Przykład:**

Znajdźmy w tablicy T taki podzbiór, który spełni następujące warunki:
– w każdej kolumnie znajduje się co najmniej jeden wybrany element,
– suma wybranych elementów jest maksymalna.

Optymalny algorytm do tego zadania będzie oparty na metodzie zachłannej – wybieraj kolejne elementy o maksymalnych wartościach, dla których jest spełniony warunek 1.

„Zachłanność" tego algorytmu polega na wybieraniu największego elementu, czyli elementu najbardziej odpowiedniego w danej chwili.

## 25.2. Kody Huffmana

kody Huffmana ▸ Kody Huffmana to jedna z powszechnie dzisiaj stosowanych metod kompresji danych. Opiera się ona na strategii zachłannej.

Kompresja ta polega na zastępowaniu znaków kompresowanego pliku krótszymi kodami, dzięki czemu wynikowy ciąg bitów jest znacznie krótszy od wejściowego.

Jak wiadomo, znaki koduje się na 8 bitach, dzięki czemu można zakodować aż do 256 znaków. Bardzo rzadko się zdarza, aby w pliku występowały wszystkie znaki ASCII, wobec tego można wykorzystać mniej bitów i przez to zaoszczędzić dużo miejsca.

kodowanie znaków ASCII

Znaki można kodować na dwa sposoby:

1. za pomocą stałej długości,
2. za pomocą kodów o zmiennej długości.

W tym rozdziale zajmiemy się pierwszym sposobem (drugi sposób zostanie omówiony w następnym rozdziale). Załóżmy, że mamy pewien ciąg znaków zbudowany z dużych liter od A do Z: ABHSSSKSKKKKKWETTY. Czy można go zapisać w krótszy sposób? Zwróćmy uwagę, że w naszym ciągu część znaków się powtarza – można je zapisać w bardziej efektywny sposób – ciąg powtarzających się znaków zastąpmy liczbą i znakiem, który się powtarza, a liczba będzie oznaczała ilość powtórzeń: ABHSSSKSKKKKKWETTY = = ABH3SKS5KWE2TY.

Tekst jest wyraźnie krótszy i możemy go bez problemu przetworzyć na tekst pierwotny. Jednak kompresja taka była możliwa dzięki wprowadzeniu dodatkowych znaków – cyfr.

Załóżmy, że chcemy kodować cztery znaki: A, B, C i D. Przypiszmy tym znakom kody trzybitowe:

A – 000
B – 001
C – 010
D – 100

Wykorzystujemy tylko cztery znaki, zatem nie potrzebujemy ośmiu bitów – wystarczą nam trzy bity, gdyż $2^3 = 8$. Kod ASCII litery A to 01000001. Zastąpiliśmy go kodem 000 i zyskaliśmy zmniejszenie wielkości danych o 62,5%.

Spróbujmy teraz zakodować następujący tekst:

ABACCADADDB

Każdą literę zastępujemy odpowiadającą jej grupą trzech bitów

ABACCADADDB – 00000100001001000010000010010000 1

Proces odwrotny jest równie prosty. Ciąg bitów dzielimy na trzybitowe grupy i każdą z tych grup zamieniamy na litery:

01000000010001000100 - 010 000 000 010 001 000 100 – CAACBAD

Tak działa kodowanie liter w standardzie ASCII, gdzie każdy znak to 8 bitów.

## ZADANIA DO WYKONANIA

1. Napisz algorytm w formie schematu blokowego skracający zapis ciągu znaków. W miejscu, gdzie pojawiają się po sobie te same znaki, zastąp je liczbą i znakiem, który się powtarza, a liczba będzie oznaczała ilość powtórzeń.
2. Napisz program z wykorzystaniem języka C++ skracający zapis ciągu znaków. W miejscu, gdzie pojawiają się po sobie te same znaki, zastąp je liczbą i znakiem, który się powtarza, a liczba będzie oznaczała ilość powtórzeń.
3. Napisz algorytm w formie schematu blokowego umożliwiający wykonanie szyfrowania i deszyfracji ciągu składającego się ze znaków, które przyjmują następujące wartości: A = 00, B = 01, C = 10, D = 11.
4. Napisz program w języku Java umożliwiający wykonanie szyfrowania i deszyfracji ciągu składającego się ze znaków, które przyjmują następujące wartości: A = 00, B = 01, C = 10, D = 11.

## PODSUMOWANIE LEKCJI

s. 158 • **Metoda zachłanna** w celu wyznaczenia najlepszego rozwiązania dokonuje w danym momencie wyboru najlepszego rozwiązania częściowego.

s. 158 • Typowe zadanie rozwiązywane metodą zachlanną ma **charakter optymalizacji.**

s. 158 • **Kody Huffmana** to jedna z powszechnie dzisiaj stosowanych metod kompresji danych. Została opracowana w 1952 r. przez Amerykanina Davida Huffmana.

s. 159 • Kodami Huffmana **znaki ASCII można kodować na dwa sposoby**: za pomocą stałej długości oraz za pomocą kodów o zmiennej długości.

# 26. Tworzenie szyfrów przestawieniowych, czyli kodowanie znaków o zmiennej długości

**NA TEJ LEKCJI:**

- poznasz szyfr przestawieniowy;
- dowiesz się, jak zaszyfrować tekst jawny za pomocą szyfru przestawieniowego z tablicą.

Szyfr przestawieniowy to jedna z klas metod szyfrowania. Szyfry te charakteryzują się tym, że w zaszyfrowanym tekście występują wszystkie znaki z tekstu jawnego, ale w innej kolejności.

kodowanie znaków o zmiennej długości

Kodowanie znaków o zmiennej długości jest natomiast drugim sposobem kodowania znaków ASCII za pomocą kodów Huffmana. Sposób ten wykazuje lepsze własności dla znaków częściej występujących.

## 26.1. Szyfr przestawieniowy

szyfr przestawieniowy

Szyfr ten polega na zmianie położenia znaków tworzących całość tekstu jawnego, przez co staje się nieczytelny dla osoby niewtajemniczonej. Oczywiście, aby tak zaszyfrowaną wiadomość odczytać, potrzeba narzędzia do deszyfracji.

Być może w dzieciństwie bawiliście się w szyfrowanie wiadomości albo tworzyliście własny język, na przykład poprzez dodawanie litery A do każdej sylaby? Jednym z przykładów szyfrowania przestawnego może być bardzo podobny do niego szyfr – stworzymy bardzo prosty, ale utrudniający osobie niewtajemniczonej odczytanie zaszyfrowanej wiadomości. Zamiast wstawiać literę A, przestawimy litery w parach znaków.

**Przykład:**

tekst: ALA MA KOTA
pary: AL A MA KO TA
zamiana: LA A AM OK AT
szyfr: LAA AM OKAT

Jeżeli wyraz w naszym tekście jawnym ma nieparzystą liczbę znaków, to przestawiamy tylko znaki parzyste – ostatnią literę zostawiamy bez zmian, na przykład: ALA dzielimy na AL A, zamieniamy tylko parę AL na LA i samo A pozostaje, po zaszyfrowaniu otrzymujemy wyraz LAA. Zwróćcie uwagę, że nasz szyfr jest symetryczny do tekstu jawnego. Gdy zaszyfrujemy naszą metodą otrzymany szyfr, wynikiem będzie nasz tekst jawny.

**Przykład:**

szyfr: LAA AM OKAT
pary: LA A AM OK AT
zamiana: AL A MA KO TA
szyfr (który staje się kodem jawnym): ALA MA KOTA

Algorytm szyfrowania przestawieniowego ze zmianą liter w parach:

**Wejście:**
Łańcuch tekstowy $a$, który zawiera tekst poddawany szyfrowaniu lub deszyfracji

**Wyjście:**
Zaszyfrowany tekst $a$

**Elementy pomocnicze:**
$i$ – indeks, gdzie $i$ należy do zbioru liczb naturalnych

**Lista kroków:**
Krok 1: $i = 0$
Krok 2: Dopóki $i + 1 < |a|$ wykonuj koki od Kroku 3 do Kroku 4
Krok 3: $a[i] \leftrightarrow a[i+1]$
Krok 4: $i = i + 2$
Krok 5: Zakończ

## 26.2. Szyfr przestawieniowy z tablicą

szyfr przestawieniowy z tablicą

Kolejną możliwością zaszyfrowania tekstu jawnego jest użycie szyfru przestawieniowego z tablicą – dla danego łańcucha $a$ wyznaczamy liczbę neutralną $n$, aby $n^2 \leq |a|$.

W tym przypadku tekst, który wprowadzimy, zostanie podzielony na podane przez nas grupy $n$-znakowe. Ostatnia grupa, jeśli będzie niepełna, zostanie dopełniona znakami zdefiniowanymi przez nas – cyframi, znakami przestankowymi lub literami z zakresu od A do Z. Z grupy powstaje tablica. W algorytmie, który wykorzystamy do tworzenia szyfru na wyjście, przesyłamy kolejne kolumny tej tablicy.

**Przykład:**

tekst: CZYTAĆ TO BARDZIEJ ŻYĆ – 22 znaki (razem ze spacjami), $n = 5$
grupy: CZYTA Ć TO BARDZ IEJ Ż YĆAAA
tablica: CZYTA
Ć TO
BARDZ
IEJ Ż
YĆAAA
kolumny: CĆBIY Z AEĆ YTRJA TOD A A ZŻA

W rzeczywistości nie musimy tworzyć tablicy. Kolumny do odczytu uzyskamy po odczytaniu znaków z odstępem $n$, na przykład znaki z kolumny pierwszej na pozycjach $1 + 6(i - 1)$, w drugiej kolumnie na pozycjach $2 + 5(i - 1)$ itd.

Deszyfracja tego kodu polega, tak jak we wcześniejszym przykładzie, na ponownym wykonaniu tego samego algorytmu nad szyfrem – zatem szyfr jest symetryczny.

Algorytm szyfrowania przestawieniowego z pomocą tablicy:

**Wyjście**
Łańcuch tekstowy $a$, który zawiera tekst jawny

**Wyjście:**
Zaszyfrowany łańcuch $t$

**Elementy pomocnicze:**
$n$ – liczba wierszy lub kolumn w tablicy, gdzie $n$ należy do zbioru liczb naturalnych
$i, j$ – indeksy należące do zbioru liczb naturalnych

Liczba kroków:
Krok 1: $n = 1$
Krok 2: Dopóki $n^2 < |a|$ wykonuj $n = n + 1$
Krok 3: Dopóki $|a| < n^2$ wykonuj $a = a +$ "A"
Krok 4: $t =$ " "
Krok 5: Dla $j = 1, 2, \ldots, n$ wykonuj Krok 6
Krok 6: Dla $i = 0, 1, \ldots, n - 1$ wykonuj $t = t + a[j + n * i]$
Krok 7: Zakończ

## 26.3. Kody bezprzystankowe, czyli kody o zmiennej długości

kody bezprzystankowe

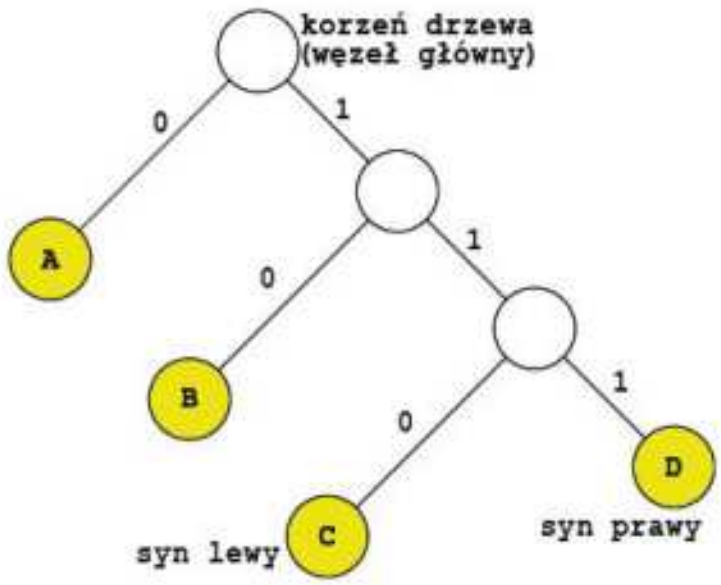
Alternatywą dla opisanych we wcześniejszym rozdziale kodów o stałej długości są kody o długości zmiennej. Wykazują one lepsze właściwości dla znaków częściej występujących. Zauważmy, że lepiej jest przypisać mniej bitów znakom częściej występującym i więcej bitów znakom rzadziej występującym.

Jeśli chcemy stosować kody o różnej liczbie bitów, to muszą one spełniać jeden, bardzo ważny warunek – żaden kod nie może być przedrostkiem innego kodu, na przykład A = 0, B = 01. Mamy kod 010100. Jaki znak jest na początku?

Żaden kod dłuższy nie może zaczynać się tak samo jak kod krótszy, ponieważ nie będziemy w stanie ich rozróżnić.

Kody bezprzystankowe reprezentuje się drzewami binarnymi, które mają ułatwić nam odczyt – gałąź lewa zawsze w takim drzewie będzie miała wartość bitową równą 0, a gałąź prawa – równą 1. Idąc po ścieżce, zbieramy po kolei bity:

A = 0, B = 10, C = 110, D = 111




Rys. 26.1. Drzewo prezentujące zapis kodu bezprzystankowgo

Zasada tworzenia drzewa kodu bezprzystankowego jest następująca:

Korzeń drzewa powinien zawsze zawierać węzeł startowy.

Zapamiętujemy kodowany znak i odczytujemy kolejne bity jego kodu w pętli, ustawiając węzeł bieżący na korzeń drzewa. Jeśli nie ma dalszych bitów, to zapamiętany znak umieszczamy w bieżącym węźle. Inaczej odczytujemy bit kodu. Jeśli ma on wartość 0 i nie istnieje lewy syn bieżącego węzła, to tworzymy lewego syna. Jeśli ma wartość 1 i prawy syn nie istnieje, to tworzymy go. Przechodzimy do lewego syna (dla bitu 0) lub do prawego syna (dla bitu 1) i kontynuujemy pętlę odczytu bitów.

Gdy pętla się zakończy, przechodzimy do definicji kolejnego znaku i całość procedury powtarzamy. Jeśli brak dalszych definicji, kończymy. Drzewo jest gotowe.

## ZADANIA DO WYKONANIA

1. Zaszyfruj podaną informację i dokonaj jej deszyfracji za pomocą szyfru podanego w przykładzie prezentowanym w podrozdziale 26.1.:

   KODY HUFFMANA SĄ ZWIĄZANE Z KOMPRESJĄ PLIKÓW
2. Napisz program z wykorzystaniem języka C++ szyfrujący ciąg znaków, wykorzystując algorytm zapisany w podrozdziale 26.1.
3. Napisz program z wykorzystaniem języka Java szyfrujący ciąg znaków, wykorzystując algorytm zapisany w podrozdziale 26.1.
4. Zaszyfruj podaną informację i dokonaj jej deszyfracji za pomocą szyfru podanego w przykładzie prezentowanym w podrozdziale 26.2.:

   KTO CZYTA KSIĄŻKI ŻYJE PODWÓJNIE

5. Napisz program z wykorzystaniem języka C++ szyfrujący ciąg znaków, wykorzystując algorytm zapisany w podrozdziale 26.2. dla dowolnego *n*.
6. Napisz program z wykorzystaniem języka Java szyfrujący ciąg znaków, wykorzystując algorytm zapisany w podrozdziale 26.2. dla dowolnego *n*.
7. Napisz algorytm – listę kroków – tworzenia drzewa kodu bezprzystankowego.
8. Napisz program w języku C++, który wczytuje definicję kodu bezprzystankowego, tworzy na jej podstawie odpowiednie drzewo kodu, po czym wczytuje ciąg zer i jedynek, w których jest zapisana wiadomość. Następnie odtwarza wiadomość, korzystając z utworzonego drzewa kodu. Pierwsza liczba określa ilość słów kodowych *n*. Następnie występuje *n* wierszy typu znak, spacja, kod binarny. W ostatnim wierszu jest łańcuch z zakodowaną tym kodem wiadomością.

## PODSUMOWANIE LEKCJI

- **Szyfr przestawieniowy** to jedna z klas metod szyfrowania. Szyfry te charakteryzują się tym, że w zaszyfrowanym tekście występują wszystkie znaki z tekstu jawnego, ale w innej kolejności. ◂ s. 161
- **Kodowanie znaków o zmiennej długości** jest drugim sposobem kodowania znaków ASCII za pomocą kodów Huffmana. Kody o długości zmiennej wskazują lepsze właściwości dla znaków częściej występujących, ponieważ lepiej jest przypisać mniej bitów znakom częściej występującym i więcej bitów znakom rzadziej występującym. ◂ s. 161
- Kolejną możliwością zaszyfrowania tekstu jawnego jest **użycie szyfru przestawieniowego z tablicą** – dla danego łańcucha wyznaczamy liczbę neutralną *n*, aby $n^2 \leq |a|$. ◂ s. 162
- **Kody bezprzystankowe** reprezentuje się drzewami binarnymi, które mają ułatwić nam odczyt. ◂ s. 163

# 27. Automat wydaje, czyli jak dynamicznie wydawać resztę

**NA TEJ LEKCJI:**

- dowiesz się, na czym polega metoda zachłanna;
- użyjesz w programie zmiennych tablicowych.

Wrzucając pieniądze do automatu z napojami, nie musisz odliczać dokładnej kwoty. Urządzenie samo sumuje nominały, wydaje towar i resztę. Jak komputer w automacie oblicza resztę? Czy algorytm jest skomplikowany? Opracujmy go wspólnie. Nie będzie to bardzo trudne.

## 27.1. Zachłannie, czyli wydajemy resztę z użyciem najmniejszej liczby nominałów

Przypomnij sobie, jak sprzedawca w sklepie wydaje resztę. Oczywiście może to zrobić, używając dowolnych nominałów. Robi to na tyle racjonalnie, by w kasie nie zabrakło żadnego z nich. Automat musi kierować się innymi kryteriami. Nie tylko kwota musi się zgadzać, lecz także wydawanie musi być optymalne, czyli z użyciem jak najmniejszej liczby wydawanych monet. Załóżmy, że ma zostać wydana kwota 13 zł i 60 groszy z użyciem najmniejszej liczby monet (rys. 27.1.).




Rys. 27.1. Optymalne wydanie reszty w kwocie 13,6 zł

Zastanów się teraz nad kwotami, które można wydawać za pomocą zestawu monet z rysunku 27.1. Jeśli jest ich dostatecznie dużo, możesz wydać dowolną kwotę pod warunkiem, że nie będzie ona zawierała pojedynczych groszy. Z tego wniosek, że dzięki odpowiednio dobranym w systemie monetarnym nominałom, można złożyć dowolną kwotę.

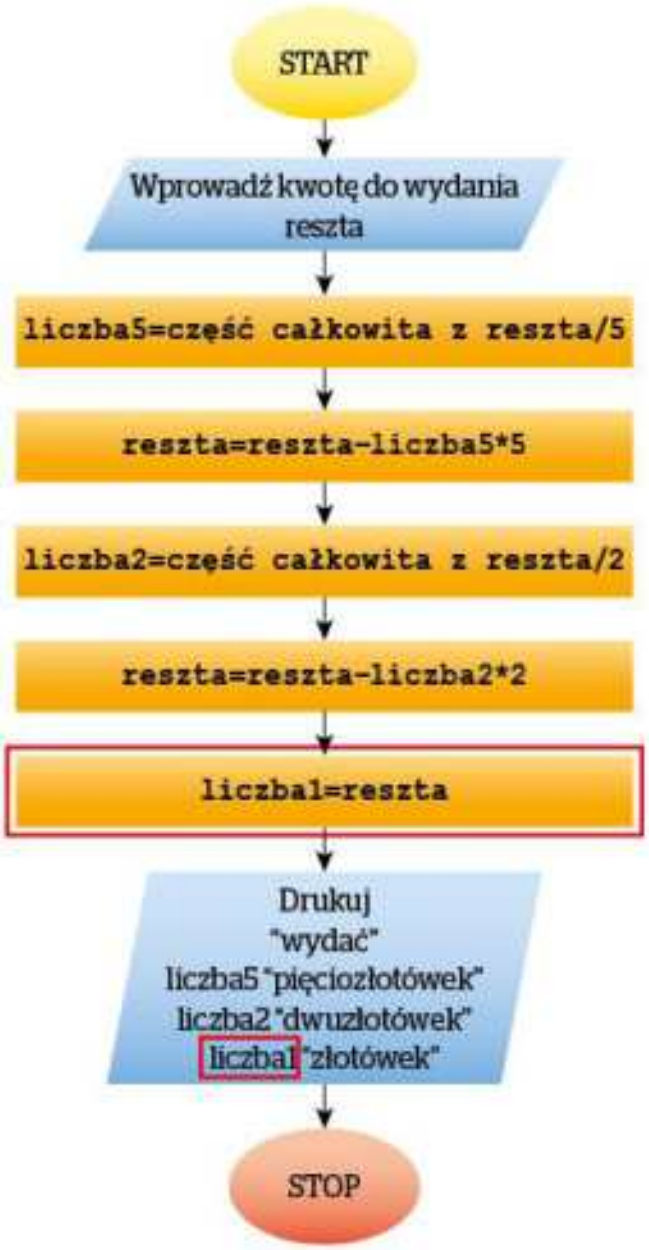
**metoda zachłanna** ▸ Wydawanie reszty w opisany sposób, czyli zaczynając od największej możliwej liczby najwyższych nominałów, nazywamy metodą zachłanną. Jeśli przeanalizujesz tę metodę, zapewne dojdziesz do wniosku, że nie sprawdzi się ona dla nietypowego zestawu nominałów. Gdybyśmy chcieli kwotę 6 złotych wydać za pomocą monet dwu- i pięciozłotowych, algorytm zachłanny wydałby najpierw

monetę pięciozłotową, bo jej wartość mieści się w 6. Kasjerka w sklepie wydałaby trzy monety dwuzłotowe. Ale ona nie stosuje algorytmu zachłannego.

Przystępując do opracowania algorytmu wydawania reszty metodą zachłanną, musimy założyć, że do dyspozycji są wszystkie dostępne nominały w odpowiedniej ilości. Dla uproszczenia w przykładzie wykorzystamy jedynie nominały 1, 2 i 5 złotych oraz założymy, że wydajemy resztę niezawierającą groszy (rys. 27.2.). W przykładowym algorytmie użyto następujących zmiennych:

`reszta` – zmienna zawierająca kwotę do wydania. W trakcie procesu wydawania zmienia wartość na równą kwocie, która pozostała do wydania.

`liczba5`, `liczba2`, `liczba1` to zmienne, w których ostatecznie zostanie zapisana liczba poszczególnych nominałów do wydania.




Rys. 27.2. Przykład algorytmu wydającego resztę metodą zachłanną dla reszty wyrażonej liczbą całkowitą

Algorytm nie wymaga omówienia. Zastanów się jednak, czy blok zaznaczony czerwoną ramką jest konieczny do otrzymania prawidłowego wyniku. Od tego miejsca wartość zmiennej `reszta` nie zmienia się do końca działania algorytmu. Może więc być wyświetlona jako liczba wydawanych złotówek, czyli

zmienna `liczba1` staje się niepotrzebna. Właśnie dokonaliśmy weryfikacji optymalności kodu programu.

indeks 0 ▸ Modyfikacja wiąże się także ze zmianą w bloku wyjściowym – jako liczbę wydawanych złotówek należy wpisać wartość zmiennej `reszta`. Może ona mieć wartość 0 dla parzystych kwot do wydania lub 1 dla nieparzystych. Na tym przykładzie widać, że każdy algorytm powinien być weryfikowany nie tylko pod kątem funkcjonalności (daje prawidłowe wyniki czy nie), lecz także pod kątem jego budowy i wykorzystania zasobów komputera. W tym przypadku dodatkowa zmienna spowoduje zarezerwowanie większego obszaru pamięci.

Program realizujący algorytm z rysunku 27.2. wykorzystuje fakt, że zmienna zadeklarowana jako `int` przechowująca wynik dzielenia będzie zawierała jedynie jego część całkowitą. Nie ma więc potrzeby korzystania z odpowiedniej funkcji (`div`) podającej wartość całkowitą liczby (rys. 27.3.).

```
1 #include <iostream>
2 #include <stdlib.h>
3
4 using namespace std;
5
6 int liczba5=0, liczba2=0, liczba1=0, reszta;
7 int main ()
8 {
9 cout << "Podaj kwotę do wydania: ";
10 cin >> reszta;
11
12 liczba5=reszta/5;
13 reszta=reszta-liczba5*5;
14 liczba2=reszta/2;
15 reszta=reszta-liczba2*2;
16
17 cout << "do wydania" << endl;
18 cout << liczba5 <<" szt. pięciozłotówek" << endl;
19 cout << liczba2 <<" szt. dwuzłotówek" << endl;
20 cout << reszta <<" szt. złotówek" << endl;
21 return 0;
22 }
```

Rys. 27.3. Program wydający resztę według przykładowego algorytmu zachłannego

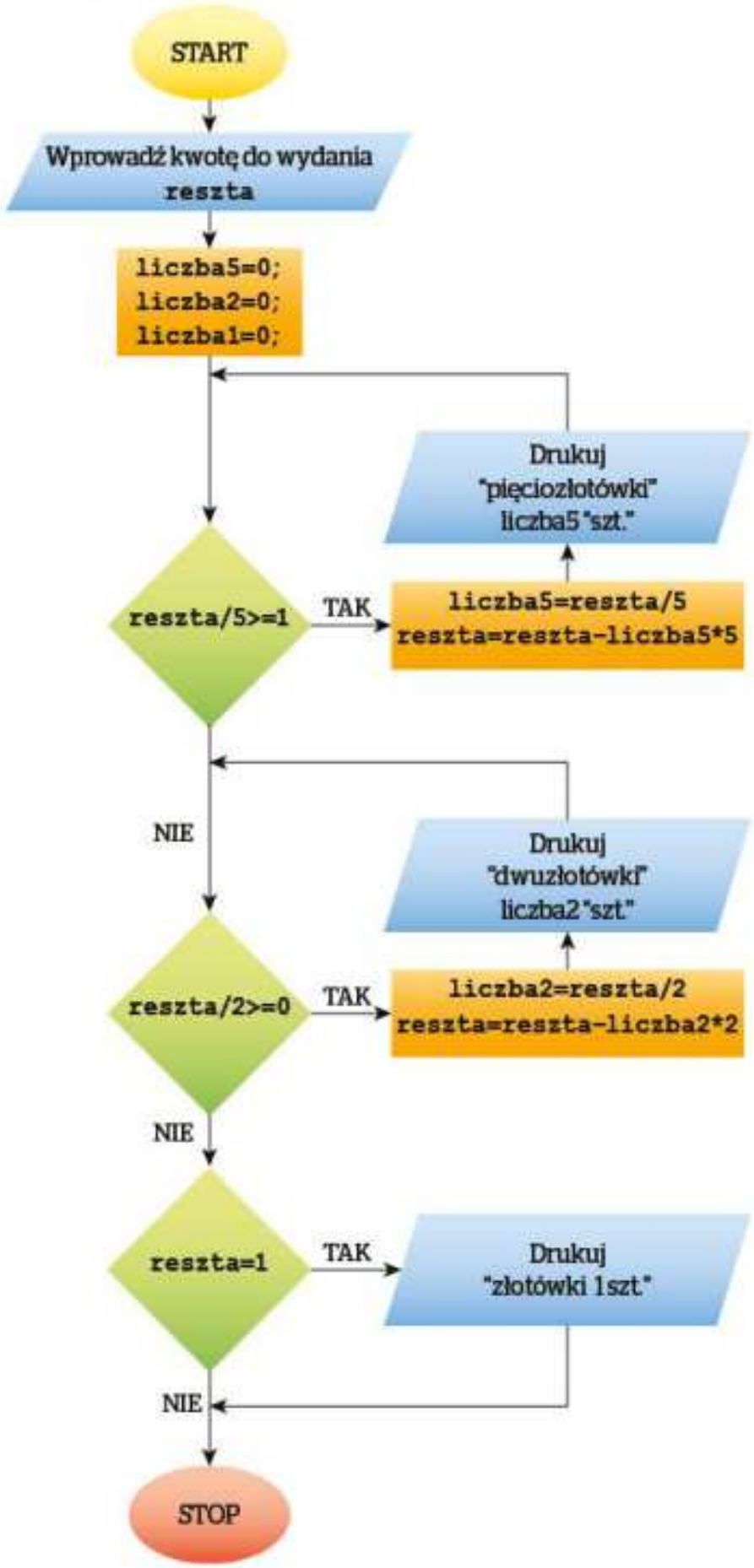
```
Podaj kwotę do wydania: 23
do wydania
4 szt. pięciozłotówek
1 szt. dwuzłotówek
1 szt. złotówek
```

```
Podaj kwotę do wydania: 10
do wydania
2 szt. pięciozłotówek
0 szt. dwuzłotówek
0 szt. złotówek
```

Rys. 27.4. Wyniki działania programu wydającego resztę metodą zachłanną dla kwoty 23 złote i 10 złotych

Program (rys. 27.3.) będzie zawsze wyświetlał liczbę wszystkich nominałów, nawet gdy nie będą używane do wydania reszty (rys. 27.4.). Zmodyfikujmy algorytm i program.

Dzięki zastosowaniu badania warunków można opuścić drukowanie i obliczenia dla nominałów większych od pozostałej do wydania kwoty (rys. 27.5.). W programie ułożonym według tego algorytmu zastosowano badanie warunków za pomocą instrukcji `if` (rys. 27.6.).




Rys. 27.5. Inne rozwiązanie algorytmu dla metody zachłannej wydawania reszty

```cpp
#include <iostream>
#include <stdlib.h>

using namespace std;

int liczba5=0, liczba2=0, liczba1=0, reszta;
int main ()
{
cout << "Wprowadź kwotę do wydania: ";
cin >> reszta;
cout << endl;
if (reszta/5>=1)
{
liczba5=reszta/5;
reszta=reszta-liczba5*5;
cout << "pięciozłotówki - " << liczba5 << " szt."<< endl;
}

if (reszta/2>=1)
{
liczba2=reszta/2;
reszta=reszta-liczba2*2;
cout << "dwuzłotówki - " << liczba2 << " szt." << endl;
}

if (reszta==1)
cout << "złotówki - 1 szt.";

return 0;
}
```

Rys. 27.6. Przykładowy program realizujący algorytm z rysunku 27.5.

```
Wprowadź kwotę do wydania: 11

pięciozłotówki - 2 szt.
złotówki - 1 szt.
```

Wydanie 11 złotych metodą zachłanną nie przewiduje dwuzłotówek.

Rys. 27.7. Wynik działania programu z rysunku 27.6.

## 27.2. Nominały w tablicach, czyli jedna zmienna, wiele wartości

W obiegu znajdują się zarówno monety, jak i banknoty. Program komputerowy może podpowiedzieć sprzedawcy, ile i jakich nominałów ma użyć do wydania reszty. W Polsce w obiegu są następujące nominały banknotów: 500 zł, 200 zł, 100 zł, 50 zł, 20 zł, 10 zł oraz monet: 5 zł, 2 zł, 1 zł, 50 gr, 20 gr, 10 gr, 5 gr, 2 gr i 1 gr. Jedynie banknotem pięćsetzłotowym nie da się wydawać reszty. Gdyby dla wszystkich pozostałych 14 nominałów budować algorytm podobny do tego z rysunku 27.3., należałoby rozbudować go o kolejnych 11 dzieleń lub

warunków. Można jednak posłużyć się tablicami, w których zostaną zapisane wszystkie nominały i użyć bloku warunkowego (rys. 27.8.). Spowoduje to, że algorytm po odpowiednich zmianach w tablicy `nominaly` będzie wydawał także resztę dla nominałów używanych na przykład w Stanach Zjednoczonych, w których używa się monety 25 centów, a nie ma monet o nominałach 2 i 20 centów.

**Przypomnienie**

◂ zmienne tablicowe

Odśwież swoją wiedzę na temat zmiennych tablicowych. Tablica jest zmienną, która może przechowywać wiele wartości. Każda pozycja tablicy jest indeksowana (od 0 do *n*), to oznacza, że ma swój numer. W związku z tym podczas deklaracji należy podać jej rozmiar. W C++, jeśli podczas deklaracji tablicy zostanie ona wypełniona danymi, podawanie rozmiaru nie jest konieczne. Podobnie jak inne zmienne tablice muszą mieć nazwę i określony typ danych, jakie będą przechowywały.

**Przykład:**

```
int tablica[]={20000,10000,5000,2000,1000,
500,200,100,50,20,10,5,2,1};
```

lub inaczej

```
int tablica[13];
```

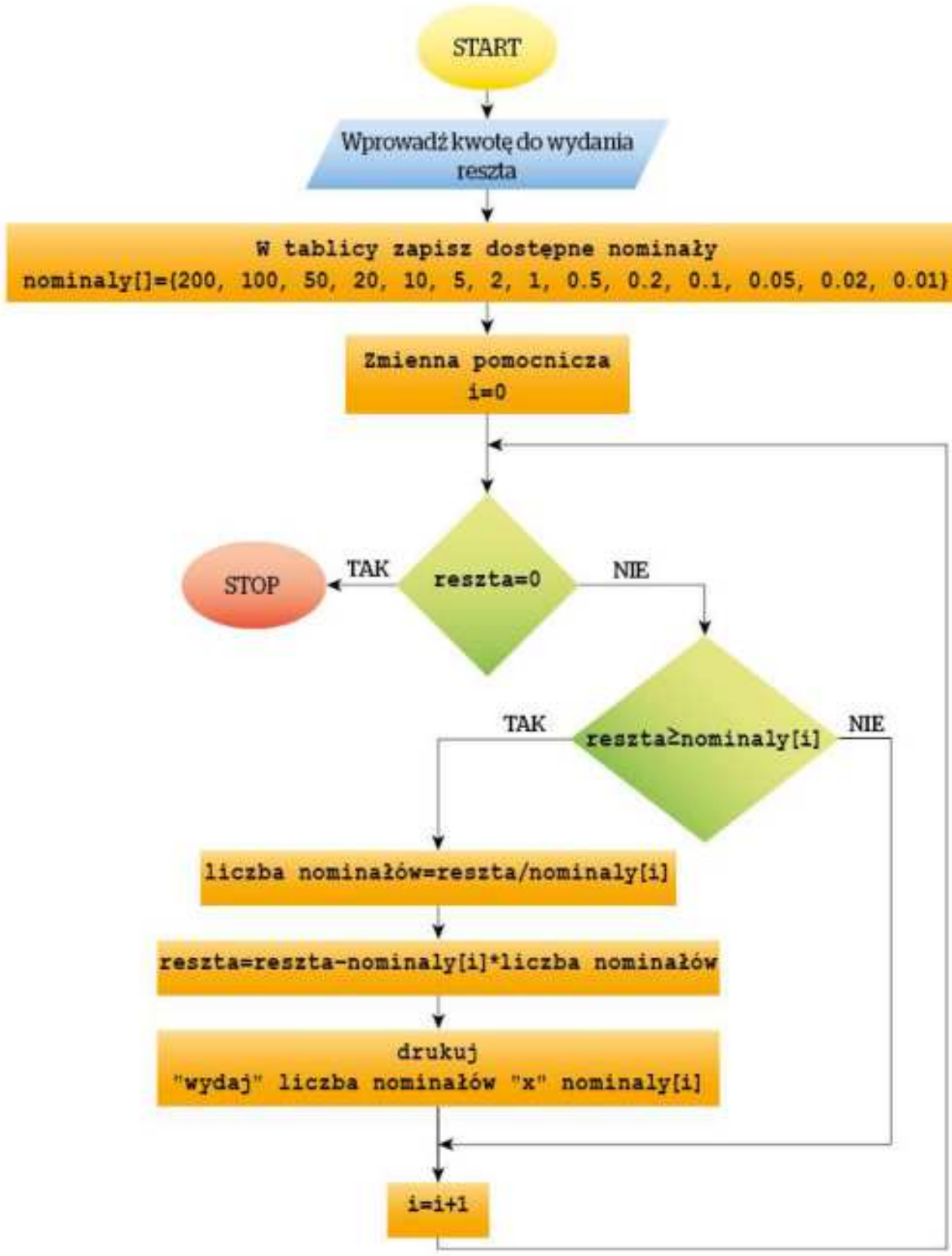
W pierwszym przypadku tablica została wypełniona 14 danymi. W drugim – ta sama tablica musi być wypełniana danymi w czasie realizacji programu. Oczywiście w obu przypadkach program może wpisywać do nich nowe wartości. W przykładzie `tablica[0]` będzie miała wartość `20000`, `tablica[4]` `1000`, a `tablica[8]` to `50`. Jak widać, indeks wskazuje bezpośrednio pozycję w tablicy.

Można zadeklarować tablice o większej liczbie wymiarów.

Zmienne tablicowe są nazywane także macierzami.

Wróćmy do naszego zadania. Algorytm (rys. 27.8.) w zasadniczej części nadal realizuje metodę zachłanną wydawania reszty. Tym razem jednak używa do tego wszystkich możliwych nominałów. W przykładowym algorytmie obliczona liczba poszczególnych nominałów jest drukowana tuż po obliczeniu i nie jest przechowywana.

Prześledźmy działanie algorytmu na przykładzie. Załóżmy, że kwota do wydania to 40,10 zł. Taką też wartość zapiszmy pod zmienną `reszta`.




Rys. 27.8. Przykładowy algorytm wydawania reszty metodą zachłanną używający 14 nominałów

Tab. 27.1. Analiza działania algorytmu wydawania reszty metodą zachłanną dla kwoty 40,10 zł

| Krok | Działanie | Wartości |
|---|---|---|
| 1. | Wprowadzenie kwoty do wydania | *reszta* = 40.10 |
| 2. | Wypełnienie tablicy nominałami | *nominały*[13]={200, 100, 50, 20, 10, 5, 2, 1, 0.5, 0.2, 0.1, 0.05, 0.02, 0.01} |
| 3. | Indeks tablicy – zmienna *i* – ustawiony na pierwszej danej tablicy | *i* = 0 |
| 4. | Czy reszta jest już wydana | *reszta*=40.1 ≠ 0<br>kierunek – NIE |
| 5. | Czy kwota do wydania zawiera w sobie przynajmniej jedną wartość nominału 200 | *nominały*[0]=200<br>40.1<*nominały*[0]<br>Kierunek – NIE |
| 6. | Indeks tablicy ustawia się na następny nominał | *i*=0+1=1 |
| 7. | Czy reszta jest już wydana | *reszta*=40.1 ≠ 0<br>kierunek – NIE |
| Kroki 4, 5, 6, 7 są wykonywane do momentu, gdy *i* = 3 | | |
| 8. | Reszta do wydania jest większa od 20 | *reszta*>=*nominały*[3]=20<br>kierunek TAK |
| 9. | Obliczanie liczby dwudziestozłotówek | *liczbanominałow*=40.1/20=2<br>Nie ma części ułamkowej, ponieważ zmienna jest typu *int* |
| 10. | Obliczanie pozostałej kwoty po wydaniu 20 zł | *reszta*=40,1–(20*2)=0.1 zł |
| 11. | Wydruk liczby dwudziestozłotówek | Wydaj 2 x 20 |
| 12. | Indeks następnego nominału | *i*=3+1=4 |
| 13. | Reszta jeszcze nie wydana | *reszta*=0.1 ≠ 0<br>kierunek – NIE |
| Dla nominałów od 4 do 9 wartość warunku *reszta*>=*nominały*[*i*] – NIE | | |
| 14. | *i*=10 | 0.1>=*nominały*[10]=0.1<br>kierunek TAK |
| 15. | Obliczanie liczby dziesięciogroszówek | *liczbanominałow*=0.1/0.1=1 |
| 16. | Obliczanie pozostałej kwoty po wydaniu 2x20 zł i 10 gr | *reszta*=0.1-0.1.*1=0 |
| 17. | Wydruk liczby dziesięciogroszówek | Wydaj 1 x 0.1 |
| 18. | Cała reszta wydana | Warunek *reszta*=0 spełniony<br>Koniec obliczeń |

Realizacja algorytmu (rys. 27.8.) w języku C++ napotyka na pewne trudności. Otóż część nominałów jest mniejsza od 1. Z tej przyczyny tablicę powinno się zadeklarować jako zmienną typu `float`. Pozostała kwota do wydania (zmienna `reszta`) także musi być liczbą rzeczywistą. Gdy do wydania zostaje 0,01 złotego, wartość zmiennej `reszta` powinna wynosić 0,01. Niestety komputer oblicza wartość zmiennej `reszta` z pewnym przybliżeniem, co spowoduje błędny wynik dla kwot do wydania o nieparzystej liczbie groszy. W programie zastosujemy więc zaokrąglenie do setnych części złotego. W programie (rys. 27.9.) została użyta funkcja `round` powodująca zaokrąglenie wyniku do części całkowitej w sposób, który znasz z matematyki, na przykład dla liczby 0,6 da wynik 1, a dla 0,4 da 0. Jej użycie jest możliwe po dołączeniu biblioteki `math.h`. Zaokrąglenie wyniku obliczania reszty pozostałej do wydania do dwóch miejsc po przecinku realizuje linia 20.

```
reszta=round((reszta-(nominaly[i]*liczbanominalow))*100)/100;
```

Tablica nominałów będzie miała postać:

```
float nominaly[]={200,100,50,20,10,5,2,1,0.5,0.2,0.1,0.05,0.02,0.01}
```

Nie ma potrzeby podawania rozmiaru tablicy w [ ], skoro podczas deklaracji wpisujemy do niej konkretne dane.

```
1 #include <iostream>
2 #include <stdlib.h>
3 #include <math.h>
4 using namespace std;
5
6 int liczbanominalow, i=0;
7 float nominaly[]={200,100,50,20,10,5,2,1,0.5,0.2,0.1,0.05,0.02,0.01}, reszta;
8
9 int main()
10 {
11 cout << "Podaj kwotę do wypłacenia: ";
12 cin >> reszta;
13
14 cout << endl << "wydaj następujące nominały:" << endl << endl;
15 while (reszta!=0)
16 {
17 if (reszta >= nominaly[i])
18  {
19   liczbanominalow=reszta/nominaly[i];
20   reszta=round((reszta-(nominaly[i]*liczbanominalow))*100)/100;
21
22   cout << nominaly[i] << " zł - " << liczbanominalow << " szt." << endl;
23  }
24 i++;
25 }
26 return 0;
27 }
```

Rys. 27.9. Przykładowy program wydający resztę metodą zachłanną korzystający z wszystkich nominałów

```
Podaj kwotę do wypłacenia: 444.31

wydaj następujące nominały:

200 zł - 2 szt.
20 zł - 2 szt.
2 zł - 2 szt.
0.2 zł - 1 szt.
0.1 zł - 1 szt.
0.01 zł - 1 szt.
```

Rys. 27.10. Wynik działania programu wydającego resztę metodą zachłanną wykorzystującego wszystkie nominały

## ZADANIA DO WYKONANIA

1. Jaką wartość może przybierać liczba wydawanych najniższych nominałów, np. 1 zł 1 gr? Uzasadnij swoją odpowiedź i podaj kilka przykładów.
2. Zmodyfikuj program z rysunku 27.3. w taki sposób, by wydawanie reszty realizowała funkcja. Zdefiniuj ją. Efekt działania programu (wyświetlanie komunikatów) powinien pozostać bez zmian.
3. Automaty, w których można dokonać zakupów, operują najczęściej nominałami od 5 zł do 10 groszy. Zmodyfikuj algorytm i program wydawania reszty w taki sposób, by mógł być zastosowany w takich urządzeniach.
4. Zmodyfikuj program z rysunku 27.9. w taki sposób, by można było wprowadzać kwotę reszty do wydania w złotówkach, np. 34,51 zł. Sprawdź, jak dokładność wykonywania obliczeń wpływa na działanie programu dla różnych kwot, w tym 34,51 zł.
5. Zmodyfikuj program wydawania reszty z wykorzystaniem wszystkich dostępnych nominałów w USA.

## PODSUMOWANIE LEKCJI

- **Metoda zachłanna** zakłada wydawanie reszty nominałami największymi z możliwych. Można opracować wiele algorytmów wykorzystujących metodę zachłanną, w tym także uwzględniających różne zakresy dostępnych nominałów. ◂ s. 166
- Dane zapisane w zmiennej tablicowej są indeksowane numerem miejsca w tablicy. Dla tablicy jednowymiarowej pierwsza ma **indeks 0**. ◂ s. 168
- **Zmienne tablicowe** są zmiennymi, w których można przechowywać wiele wartości. Wszystkie są tego samego typu, deklarowanego podczas deklaracji tablicy. Jeśli podczas deklaracji zmiennej tablicowej jednowymiarowej wypełnimy ją danymi, nie ma potrzeby podawania jej rozmiaru. ◂ s. 171

# 28. Wydajemy resztę, czyli jak wykorzystać metodę zachłanną

**NA TEJ LEKCJI:**

- dowiesz się, w jaki sposób automat wydaje resztę;
- nauczysz się wydawać resztę za pomocą algorytmu zachłannego.

Często robimy zakupy w różnego rodzaju automatach. Ale czy kiedyś zastanawialiśmy się, w jaki sposób automaty te wydają nam resztę, jeżeli wrzucimy kwotę większą niż wymagana? Spróbujemy rozwiązać ten problem za pomocą algorytmu zachłannego, czyli takiego, który w każdym kroku podejmuje najkorzystniejszą decyzję.

## 28.1. Ile reszty?

wydawanie reszty ▸ Problem wydawania reszty dotyczy tego, w jakich nominałach wydać resztę, aby użyć jak najmniejszej liczby monet.

Przyjmijmy, że mamy ciąg nominałów $A = (c_1, c_2, ..., c_n)$ oraz kwotę do wydania $r$. Monety są posortowane nominałami rosnąco ($c_1 < c_2 < ... < c_n$). Należy wyznaczyć taki nieujemny współczynnik $k_1, k_2, ..., k_n$, że $k_1c_1 + k_2c_2 + ... + k_nc_n = r$, a suma $k_1 + k_2 + ... + k_n$ jest jak najmniejsza.

Aby sobie uprościć zadanie, przyjmijmy, że wszystkie nominały oraz reszta $r$ są podzielne przez najmniejszy nominał $c_1$ (zazwyczaj najmniejszym nominałem jest 1, a pozostałe kwoty do wydania to liczby naturalne). Takie założenie zapobiegnie sytuacji, w której nie będzie można wydać kwoty reszty.

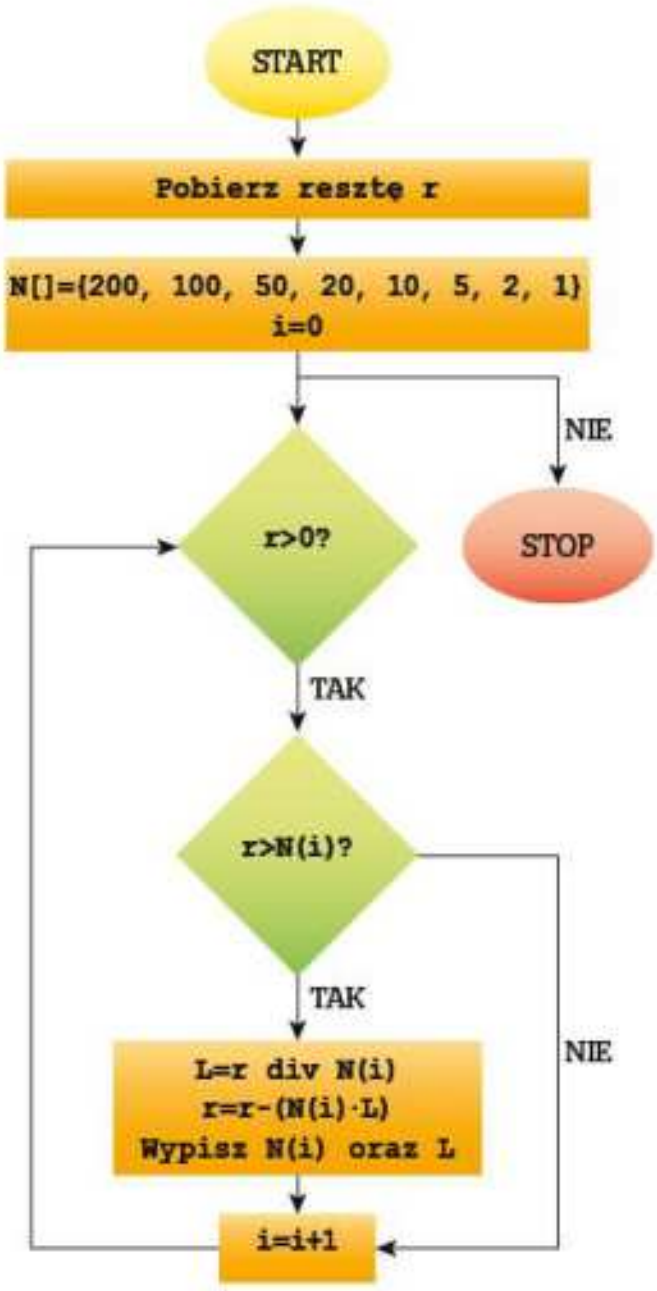
Rozwiązanie problemu możemy rozpatrywać w dwóch wariantach – ograniczonym i nieograniczonym. Wariant ograniczony polega na tym, że dysponujemy jedynie określoną liczbą monet z każdego nominału. Rozpatrzymy wariant nieograniczony, który daje możliwość skorzystania z nieograniczonej liczby monet każdego nominału.

Na początku ustalamy nominały, jakimi będziemy wydawali resztę – niech to będzie tablica $N$ o wartościach $N[200, 100, 50, 20, 10, 5, 2, 1]$.

algorytm zachłanny ▸ Do rozwiązania problemu wykorzystamy **algorytm zachłanny**, czyli do wypłacenia reszty zawsze będziemy używali najwyższych dostępnych nominałów. Tablicę posortujemy w sposób malejący. Na początku szukamy nominałów równych wypłacanej kwocie lub od niej mniejszych. Gdy je znajdziemy, wypłacamy największą możliwą ich ilość. Liczbą tą będzie wynik dzielenia, oczywiście bez reszty, wypłacanej kwoty przez wartość odnalezionego nominału.

Resztę pomniejszamy o kwotę wypłaconą za pomocą użytego nominału, następnie czynność powtarzamy tak długo, aż wypłacimy całą resztę.

Działanie tego algorytmu zilustrowano na poniższym schemacie blokowym.




Rys. 28.1. Schemat blokowy algorytmu – wydajemy resztę metodą zachłanną

Przeanalizujmy powyższy algorytm na konkretnym przykładzie – załóżmy, że mamy do zapłaty rachunek 168 zł i włożyliśmy do automatu banknot o nominale 200 zł. Do wydania mamy 32 zł.

Największym nominałem nie większym niż 30 jest 20.

32 div 20 = 1 – wypłacamy banknot 20 zł (div jest funkcją zwracającą część całkowitą z dzielenia liczb).

32 – (20 * 1) = 12 – zostało nam jeszcze do wydania 12 zł.

Kolejną wartością nie większą niż aktualna reszta jest 10.

12 div 10 = 1 – wypłacamy banknot 10 zł.

12 – (10 * 1) = 2 – zostało nam jeszcze do wydania 2 zł.

Największym nominałem nie większym niż 2 jest 2.

2 div 2 = 1 – wypłacamy 2 zł.

2 – (2 *1) = 0

Reszta została wydana za pomocą nominałów:

20 * 1 + 10 * 1 + 2 * 1 = 32

## ZADANIA DO WYKONANIA

1. Korzystając z przykładu przedstawionego w rozdziale 28.1., przeprowadź analizę wydawania reszty. Masz do dyspozycji następujące nominały banknotów i monet: 500 zł, 200 zł, 100 zł, 50 zł, 20 zł, 10 zł, 5 zł, 2 zł, 1 zł, 50 gr, 20 gr, 10 gr, 5 gr, 2 gr, 1 gr. Za zakupy w automacie zapłacono 335,78 zł banknotem o nominale 500 zł. Ile automat wypłaci reszty i w jakich nominałach?
2. Napisz program z wykorzystaniem języka C++ rozwiązujący zadanie przykładowe w podrozdziale 28.1.
3. Napisz program z wykorzystaniem języka Java rozwiązujący zadanie przykładowe w podrozdziale 28.1.
4. Napisz program z wykorzystaniem języka C++ rozwiązujący zadanie 1.
5. Napisz program z wykorzystaniem języka Java rozwiązujący zadanie 1.

## PODSUMOWANIE LEKCJI

s. 176 ▸ • Rozwiązanie problemu **wydawania reszty** polega na ustaleniu, w jakich nominałach wydać resztę, aby użyć jak najmniejszej liczby monet.

s. 176 ▸ • Za pomocą **algorytmu zachłannego** można wydać najkorzystniejszą resztę z kwoty większej niż jest wymagana.

# 29. Co z tym groszem, czyli błąd zaokrąglenia i błąd przybliżenia

**NA TEJ LEKCJI:**

- zrozumiesz, do czego jest nam potrzebne zaokrąglanie liczb;
- dowiesz się, czym jest błąd zaokrąglenia, bezwzględny i względny.

Na początku przypomnijmy sobie wiadomości z lekcji matematyki w szkole podstawowej. Potrzebne będą informacje o zaokrąglaniu liczb i podawaniu ich w przybliżeniu.

zaokrąglanie liczb

Zaokrąglanie liczb polega na ustaleniu dokładności zaokrąglenia. Należy wskazać cyfrę, względem której jest określane zaokrąglenie. Przyjęło się, że jeśli dana liczba ma cyfrę jedności 5, 6, 7, 8 lub 9, to do pełnych dziesiątek zaokrąglamy ją „do góry". Natomiast jeśli liczba ma cyfrę jedności 0, 1, 2, 3 lub 4, to zaokrąglamy ją „w dół". W związku z tym:

$203 \approx 200$

$271 \approx 270$

$308 \approx 310$

Powyższy przykład dotyczy również zaokrąglania do setek, tysięcy itd.

## 29.1. Do czego jest potrzebne zaokrąglanie liczb?

Zacznijmy od czegoś, co pozornie nie jest związane z tematem tego rozdziału. W różnych opracowaniach możesz znaleźć informację, że odległość Ziemi od Słońca to 149 000 000 km. Czy tę wartość podano dokładnie, czy w przybliżeniu? Czy długość drogi pokonanej autostradą A1 z Bydgoszczy do Gdańska wynosi 167 km? Dokładnie tyle i ani centymetra więcej? Czy w 2017 r. liczba ludności w Polsce wynosiła 38 milionów, a może trochę mniej? A jaka jest powierzchnia oceanów na Ziemi? Większość źródeł podaje, że oceany zajmują 71% powierzchni naszej planety. Niestety, wszystkie podane przykłady to liczby podane w przybliżeniu, w zaokrągleniu „w górę" lub „w dół".

Wartość podana jako „mniej więcej" w matematyce oznacza zaokrąglenie. Należy ustalić jego dokładność, na przykład do 1 miliona, 1 miliarda czy 1 cm, i podać liczbę dość bliską liczbie rzeczywistej, którą łatwiej zapamiętać.

Wszyscy na co dzień robimy zakupy. Jeśli kilogram bananów kosztuje 3,99 zł, to łatwiej zapamiętamy tę cenę, czy jej zaokrąglenie do 4 zł? Jeśli na rachunek składają się pozycje, które kosztowały: 3,99 zł, 15,79 zł, 20,18 zł to czy nie lepiej podać je jako: 4 zł, 16 zł, 20 zł? Rzeczywisty koszt zakupu tych trzech produktów

to 39,96 zł, a ten sam koszt policzony po zaokrąglonych cenach to 40 zł – więc nie jest on dokładny. Różnica wynosi 4 grosze, ale czy jest ona ważna? Często nie jest, a stosując zaokrąglenia, ułatwiamy sobie życie.

## Zaokrąglanie liczb naturalnych i ułamków dziesiętnych – zaokrąglanie liczb do rzędu dziesiątek

zaokrąglanie liczb do rzędu dziesiątek

Zaokrąglanie liczb do rzędu dziesiątek polega na znalezieniu jak najbliższej tej liczbie wielokrotności liczby 10. Przykładowo: liczbę 37 chcemy zaokrąglić do rzędu dziesiątek, więc szukamy wielokrotności liczb 10: 0, 10, 20, 30, 40, 50... – zauważamy, że 37 znajduje się między 30 i 40 oraz że najbliższą tej liczbie wielokrotnością liczby 10 jest 40 (mniejsza różnica), więc przybliżeniem liczby 37 będzie liczba 40. Jak tą metodą przybliżyć liczbę, której wartością jedności jest 5, na przykład 45? Różnica między 40 a 45 jest taka sama, jak między 45 a 50. Jeśli zaokrąglana liczba znajduje się w połowie, to jako wynik zaokrąglenia przyjmujemy tę większą liczbę. W przypadku liczby 45 będzie to 50. Reguła ta nie dotyczy liczb mniejszych od 0.

Zajmijmy się teraz ułamkami. Mamy liczbę 33,76 i chcemy ją zaokrąglić do pełnej wielokrotności liczby 10. Sprawdzamy, do której wielokrotności jej bliżej. Liczba 33,76 znajduje się między 30 i 40, a bliżej jej do 30, więc zaokrągleniem do pełnej wielokrotności liczby 33,76 będzie liczba 30.

Ta sama reguła obowiązuje przy zaokrąglaniu do pełnych setek, tysięcy itd.

## Zaokrąglanie liczb naturalnych i ułamków dziesiętnych – zaokrąglanie liczb do rzędu jedności

Zaokrąglanie liczb do rzędu jedności niczym się nie różni od wyżej omówionych sposobów. Jeśli jakąś liczbę chcemy zaokrąglić do rzędu jedności, to z wielokrotności liczby 1, tj.: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 itd., wybieramy tę, do której danej liczbie jest najbliżej. Jeśli ta liczba znajduje się w połowie między wymienionymi wielokrotnościami, to trzeba wybrać większą wielokrotność. Przykładowo: liczba 7,5 znajduje się w połowie między 7 i 8, więc jej zaokrągleniem do jedności to 8, a nie 7.

## Zaokrąglenie liczby do rzędu dziesiątek

zaokrąglanie liczb do rzędu części dziesiątych

Zajmijmy się teraz zaokrąglaniem ułamka dziesiętnego do jednej liczby po przecinku, czyli do rzędu części dziesiątych. W procesie tym chodzi o to, aby wynikiem takiego zaokrąglania była liczba z jedną cyfrą po przecinku, nawet jeśli tą liczbą będzie 0.

Liczbę 53,59 chcemy zaokrąglić do jednej liczby po przecinku. Ponieważ wynik ma zawierać zawsze jedną liczbę po przecinku, robimy pewnego rodzaju „cięcie" danej liczby, aby kończyła się ona na części dziesiętnej ułamka. Jeżeli cyfra, która zapisana jako setna część ułamka jest większa od 5 lub równa 5, to cyfrę zapisaną jako część dziesiątą podwyższamy o 1, na przykład: $53{,}59 \approx 53{,}6$, w przeciwnym przypadku cyfrę zapisaną jako część dziesiątą zostawiamy bez zmian, na przykład $21{,}13 \approx 21{,}1$.

Zasada ta również obowiązuje, jeżeli będzie zaokrąglać nasze ułamki do 2, 3 i dalszych miejsc za przecinkiem, na przykład: 3,236 ≈ 3,24, 3,5622 ≈ 3,562.

## 29.2. Zaokrąglanie liczb z nadmiarem i niedomiarem

Zaokrąglania, o których była mowa w poprzednich rozdziałach, to jeszcze nie wszystko. Rozważmy przykład: rodzice zaplanowali remont łazienki, chcą na podłodze wymienić terakotę. Sklep sprzedaje ją w paczkach po 1 m². Nie można więc kupić dokładnie tyle płytek, by pokryły powierzchnię przykładowo: 4,3 m², 7,5 m² czy 4,25 m². Można kupić: 1 m², 2 m², 3 m² itd. Pole powierzchni podłogi w łazience rodziców wynosi 5,25 m². Gdyby tę liczbę zaokrąglić do jedności, czyli do pełnych metrów kwadratowych, to wyszłoby, że muszą kupić 5 m². Jednak to nie wystarczy na całą podłogę, więc trzeba dokupić jeszcze jedną paczkę, czyli 1 m². Zatem by całą podłogę w łazience pokryć tą terakotą, rodzice kupią 6 m², a nie 5 m². Matematycznie rzecz ujmując, mieliśmy do czynienia z zaokrąglaniem liczby 5,25 do jedności, ale w przypadku tego zadania, po uwzględnieniu wielokrotności liczby 1, trzeba było wybrać liczbę 6, a nie 5, mimo że liczbie 5,25 bliżej do 5 niż do 6.

Zaokrąglanie liczb do większej liczby spośród dwóch danych wielokrotności nazywamy zaokrąglaniem z nadmiarem. Na przykład:

Zaokrąglimy liczbę 5136,3336 z nadmiarem do rzędu setek – 5136,3336 ≈ 5200,0000.

Teraz wyobraź sobie, że razem z sześciorgiem znajomych z klasy chcesz kupić pizzę i coś do picia. Złożyliście się po 5 zł. Za zebraną kwotę robisz zakupy, po których zostało ci 2,50 zł. Chcesz każdemu oddać odpowiednią sumę. Dzielisz więc 2,50 zł przez liczbę osób, czyli przez 7, i otrzymujesz, że: 2,50 : 7 = 0,35(714285) zł. Nie wyszła równa liczba. Ile zatem musisz oddać znajomym? Przyjmij, że z każdym chcesz się rozliczyć z dokładnością do 1 grosza, czyli do 0,01 zł. Zaokrąglasz więc otrzymaną liczbę do 2 miejsc po przecinku (jak omówiono w poprzednim rozdziale) i każdemu zwracasz po 0,36 zł (bo 3 cyfra za przecinkiem jest większa od 4). Masz jednak nowy problem, gdyż 7 razy 0,36 zł daje 2,52 zł, a zostało ci 2,50 zł. Musisz więc wybrać: albo sobie zwracasz o 2 grosze mniej (czyli 0,34 zł, a pozostałym po 0,36 zł), albo pozostałym osobom dajesz po 0,35 zł i sobie bierzesz 0,40 zł. Druga wersja z matematycznego punktu widzenia to zaokrąglenie z niedomiarem, ponieważ została zignorowana, w tym przypadku, 3 cyfra za przecinkiem.

Zaokrąglimy liczbę 5166,3336 z niedomiarem do rzędu setek – 5166,3336 ≈ 5100,0000.

## 29.3. Błąd zaokrąglenia

Błąd zaokrąglenia to różnica pomiędzy dokładną wartością oraz liczbą użytą w obliczeniach numerycznych. Błąd przybliżenia pojawia się, gdy wartości pomiarowe wykorzystane w obliczeniach są obarczone błędem lub ze względów obliczeniowych dokładne wyliczenie jakiejś wartości nie jest możliwe.

◂ błąd zaokrąglenia

Najczęściej pojawiające się błędy z powodu zaokrągleń to błędy pomiarowe i błędy obliczeniowe. Pierwszych nie da się uniknąć w przypadku liczb dotyczących rzeczywistych odległości czy rozmiarów różnych obiektów na świecie. Jednak zastosowanie bardzo kosztowych technik metrologicznych może je zmniejszyć do wartości pomijalnych w obliczeniach.

Błędy obliczeniowe powstają na skutek niedoskonałości algorytmu obliczeniowego. Liczby rzeczywiste w komputerach przedstawia się zawsze ze skończoną precyzją (błędy zaokrągleń). W pewnych szczególnych przypadkach błędy numeryczne ulegają sumowaniu, co wprowadza znaczne niedokładności stosowanych przybliżeń.

Niektóre algorytmy obliczeniowe są bardzo czułe na błędy przybliżeń. Szczególnie jest to widoczne w układach nieliniowych, jak np. pogoda na Ziemi (bardzo trudno jest podać prawidłową długoterminową prognozę pogody). Dokładne modelowanie zjawisk tego typu jest prawie niemożliwe, nawet przy ogromnej mocy obliczeniowej.

## Błąd bezwzględny

błąd bezwzględny ▸ Błąd bezwzględny, oznaczany grecką literą ε (epsilon), to maksymalna niezgodność pomiędzy wartością przybliżoną i pozbawioną błędów. Inaczej mówiąc, to różnica między otrzymanym wynikiem a zaokrąglaną liczbą. Błąd ten może być liczbą zarówno dodatnią, jak i ujemną.

Niech $a$ oznacza liczbę, którą będziemy zaokrąglać, $b$ – wartość zaokrąglenia, a $\Delta c$ – poszukiwany błąd bezwzględny:

$a = 207$
$b = 210$
$\varepsilon = b - a = 210 - 201 = 3$

Błąd bezwzględny $\varepsilon = 3$

Rozważmy inny przykład: $513 \approx 510$

$a = 513$
$b = 510$
$\varepsilon = b - a = 510 - 513 = -3$

W rozpatrywanym przez nas odczycie nie interesuje nas ani wartość ujemna, ani dodatnia błędu bezwzględnego (błąd losowy). To na otrzymany wynik narzucamy moduł (wartość bezwzględną) i wówczas nasz wynik będzie zawsze dodatni. Przykładowo dla powyższego zaokrąglenia otrzymamy:

$|\varepsilon| = |b - a| = |510 - 513| = |-3| = 3$

O tym, czy na ten błąd należy narzucać moduł, czy nie, decydujemy my, gdyż wynika on tylko z naszych potrzeb. Na ogół przyjmuje się, że jeśli w zadaniu nie jest napisane, czy należy na niego narzucać moduł, czy nie, to domyślnie się to robi.

## Błąd względny

błąd względny

Aby obliczyć błąd względny (oznaczany jako δ) należy podzielić błąd bezwzględny Δc przez liczbę, która była zaokrąglana (czyli przez wartość dokładną *a*), i otrzymany wynik zamienić na procenty (pomnożyć go przez 100 i dopisać symbol %). Zatem:

$\delta = \varepsilon / a * 100\%$

lub w razie potrzeby:

$|\delta| = |\varepsilon| / |a| * 100\%$

Rozpatrzmy następujący przykład:

$a = 1600$
$b = 1682$
$\varepsilon = 82$

$\delta = \varepsilon / a * 100\%$
$\delta = 82 / 1600 * 100\% \approx 5{,}125\%$

## ZADANIA DO WYKONANIA

1. Napisz algorytm w postaci schematu blokowego, który będzie zaokrąglał podaną liczbę z nadmiarem do rzędu:
   a) tysięcy,
   b) setek,
   c) dziesiątek,
   d) jedności.
2. Napisz program komputerowy, który zaokrągli podaną liczbę z nadmiarem do rzędu tysięcy. Użyj algorytmu napisanego w zadaniu 1. i wykorzystaj język C++. Jeżeli wprowadzona liczba będzie mniejsza niż 1000, na ekranie powinna pojawić się informacja: „Wprowadź liczbę większą od 1000". Jeżeli użytkownik wprowadzi ciąg znaków, który nie jest liczbą, na ekranie powinna pojawić się informacja: „Podany ciąg znaków nie jest liczbą".
3. Napisz algorytm w postaci pseudokodu, który będzie zaokrąglał podaną liczbę z niedomiarem do rzędu:
   a) tysięcy,
   b) setek,
   c) dziesiątek,
   d) jedności.
4. Napisz program komputerowy, który zaokrągli podaną liczbę z nadmiarem do rzędu setek. Użyj algorytmu napisanego w zadaniu 3. i wykorzystaj język Java. Jeżeli wprowadzona liczba będzie mniejsza niż 100, na ekranie powinna pojawić się informacja: „Wprowadź liczbę większą od 100". Jeżeli użytkownik wprowadzi ciąg znaków, który nie jest liczbą, na ekranie powinna pojawić się informacja: „Podany ciąg znaków nie jest liczbą".
5. Napisz algorytm w postaci schematu blokowego, który będzie obliczał błąd bezwzględny zaokrąglenia liczby *N* do rzędu setek.

6. Napisz program komputerowy, który oblicza błąd bezwzględny zaokrąglenia liczby *N* do rzędu setek. Użyj algorytmu napisanego w zadaniu 5. i wykorzystaj język C++. Jeżeli wprowadzona liczba będzie mniejsza niż 100, na ekranie powinna pojawić się informacja: „Wprowadź liczbę większą od 100". Jeżeli użytkownik wprowadzi ciąg znaków, który nie jest liczbą, na ekranie powinna pojawić się informacja: „Podany ciąg znaków nie jest liczbą".
7. Napisz algorytm w postaci schematu blokowego, który oblicza błąd względny zaokrąglenia liczby *N* do rzędu tysięcy.
8. Napisz program komputerowy, który oblicza błąd względny zaokrąglenia liczby *N* do rzędu tysięcy. Użyj algorytmu napisanego w zadaniu 7. i wykorzystaj język Java. Jeżeli wprowadzona liczba będzie mniejsza niż 1000, na ekranie powinna pojawić się informacja: „Wprowadź liczbę większą od 1000". Jeżeli użytkownik wprowadzi ciąg znaków, który nie jest liczbą, na ekranie powinna pojawić się informacja: „Podany ciąg znaków nie jest liczbą".

## PODSUMOWANIE LEKCJI

s. 179 • **Zaokrąglanie liczb** polega na ustaleniu dokładności zaokrąglenia, czyli na wskazaniu cyfry, względem której jest określane zaokrąglenie, zastąpienie zerami wszystkich cyfr na prawo od wskazanej cyfry oraz zwiększenie wskazanej cyfry o jeden.

s. 180 • **Zaokrąglanie liczb do rzędu dziesiątek** polega na znalezieniu jak najbliższej tej liczbie wielokrotności liczby 10.

s. 180 • **Zaokrąglenie liczby do rzędu części dziesiątych** polega na znalezieniu najbliższej tej liczbie wielokrotności liczby 0,1.

s. 181 • **Błąd zaokrąglenia** to różnica pomiędzy dokładną wartością oraz liczbą użytą w obliczeniach numerycznych.

s. 182 • **Błąd bezwzględny** jest obliczany jako maksymalna niezgodność pomiędzy wartością przybliżoną i pozbawioną błędów.

s. 183 • **Błąd względny** pokazuje, jaką częścią danej liczby jest wartość, o którą obniżyliśmy lub powiększyliśmy liczbę.

# 30. Sprawdzam, czyli porównywanie tekstów i wyszukiwanie wzorca w tekście metodą naiwną

**NA TEJ LEKCJI:**

- dowiesz się, czym jest wzorzec;
- przeanalizujesz, na czym polega implementacja (wdrożenie) podczas szukania wzorca.

Problem Wyszukiwania Wzorca – WW (ang. pattern matching) to jeden z podstawowych problemów tekstowych, który intensywnie badali wybitni informatycy. Rozwiązaniem jest wskazanie w ciągu $z$ wszystkich pozycji $i$ takich, że zachodzi równość:

$z[i: i + |q|] = q$

Oznacza to, iż wzorzec $q$ jest fragmentem łańcucha $z$ występującym na pozycji $i$-tej.

## 30.1. Znajdź mnie, czyli wyszukiwanie wzorca w tekście

Czym jest wzorzec? To spójny ciąg znaków, a raczej podciąg (podtekst), który występuje w danym ciągu znaków.

◂ wzorzec

Przypuśćmy, że mamy ciąg znaków $z$ o określonej długości $n$ znaków oraz wzorzec $q$ o długości $m$ znaków. Dla każdej $i$-tej litery musimy sprawdzić, czy $m$ kolejnych znaków jest zgodnych ze znakami we *wzorcu*. Porównanie wykonywane jest tak długo, dopóki istnieje kolejne $m$ liter począwszy od $i$-tego indeksu. Złożoność tego algorytmu dla przypadku, w którym w każdym sprawdzanym zakresie nie zgadza się tylko ostatni znak, wyniesie $Q(n*m)$. Algorytm ten można porównać z liniowym $Q(n)$, ponieważ zazwyczaj nie zgadza się już pierwszy znak w ciągu.

Algorytm naiwny wyszukiwania wzorca w łańcuchu tekstowym:

**Wejście:**
$z$ – łańcuch znaków
$q$ – wzorzec

**Wyjście:**
Wszystkie pozycje wyszukiwanego wzorca $q$ w łańcuchu $z$

**Elementy pomocnicze:**
$i$ – pozycja okna $i \in N$
$n$ – długość łańcucha $z$, $n \in N$
$m$ – długość wzorca $q$, $m \in N$

**Lista kroków:**
Krok 1: $n = |s|$
Krok 2: $m = |q|$
Krok 3: Dla $i = 0, 1, ..., n - m$ wykonaj Krok 4
Krok 4: Jeśli $q = s[i: i + m]$, to wypisz $i$
Krok 5: Zakończenie

## Implementacja podczas szukania wzorca

wyszukiwanie wzorca

Załóżmy, że nasza funkcja *A*() będzie przyjmowała dwa argumenty z tablicy znaków: *ciag_znakow* i *wzorzec*. Wynikiem działania naszej funkcji będzie wartość logiczna zmiennej *ciag_znakow* występuje *wzorzec*, problem jest rozwiązany w poniższym kodzie zapisanym w języku C++.

```
bool A(char* ciag_znakow, char* wzorzec) {
  for (int i = 0; ciag_znakow[i]; i++){
    int j = 0;
    while (wzorzec[j] != '\0' && ciag_znakow[i + j]
== wzorzec[j])
      j++;
    if (j > 0 && wzorzec[j] == '\0')
      return true;
  }
  return false;
}
```

Przeprowadźmy analizę powyższego programu: dla każdego znaku w naszym łańcuchu (ciag_znakow) musimy sprawdzić, czy kolejne znaki tekstu od *i*-tej pozycji są identyczne jak w zadanym przez nas wzorcu. Po zakończeniu pętli mamy dwa przypadki: wzorzec pasuje i wtedy indeks *j* będzie większy od zera i będzie on wskazywał we wzorcu znak końca danych. W takim przypadku zostanie nam zwrócona prawda, ponieważ tekst występuje. W przeciwnym razie program będzie kontynuował pętle for. Jeśli po sprawdzeniu wszystkich znaków w naszym tekście wzorzec nie został odnaleziony, to zostanie zwrócony fałsz.

Powyższy algorytm można zoptymalizować, nie sprawdzając ostatnich $m - 1$ indeksów w sprawdzanym przez nas łańcuchu (ciag_znakow), ponieważ dalszy fragment jest krótszy od długości wzorca *m*. Należy również pamiętać o tym, że ten algorytm oszczędza czas, nie szukając długości *ciag_znakow* i *wzorzec*.

Innym rozwiązaniem problemu może być wykorzystanie idei wartowników. Na końcu naszego łańcucha oraz wzorca umieścimy dwa różne znaki. Rozwiązanie to zapobiegnie wyjściu poza obszar łańcucha przy testowaniu znaków w oknie i we wzorcu. Dzięki takiemu rozwiązaniu odpada nam sprawdzenie zakresu indeksów w oknie – algorytm wykona mniej operacji.

Algorytm naiwny wyszukiwania wzorca z wartownikami:

**Wejście:**
*z* – łańcuch znaków
*q* – wzorzec

**Wyjście:**
Wszystkie pozycje wyszukiwanego wzorca $q$ w łańcuchu $z$

**Elementy pomocnicze:**
$i$ – pozycja okna $i \in N$
$j$ – pozycja wewnątrz okna, $j \in N$
$n$ – długość łańcucha $z$, $n \in N$
$m$ – długość wzorca $q$, $m \in N$

**Lista kroków:**
Krok 1: $n = |z|$
Krok 2: $m = |p|$
Krok 3: $z = z + 'A'$
Krok 4: $q = q + 'B'$
Krok 5: Dla $i = 0, 1, ..., n - m$ wykonaj kroki od Krok 6 do Krok 8
Krok 6: $j = 0$
Krok 7: Dopóki $z[i + j] = q[j]$, wykonaj $j = j + 1$
Krok 8: Jeśli $j = m$, to wypisz $i$
Krok 9: Usuń ostatni znak z $z$
Krok 10: Usuń ostatni znak z $q$
Krok 11: Zakończ

## ZADANIA DO WYKONANIA

1. W celu przetestowania funkcji omawianej w rozdziale 30.1.napisz dodatkową funkcję testującą, która wyświetli dla podanych parametrów zrozumiałe dla człowieka dane: Czy w tekście (*ciag_znakow*) występuje wzorzec (*wzorzec*)?
2. Napisz funkcję, która będzie przyjmowała tablicę znaków łańcuch (*z*) i wzorzec (*q*), następnie zwróci liczbę całkowitą (*C*), oznaczającą ilość wystąpień wzorca (*q*) w łańcuchu. Przetestuj działanie dla łańcucha znaków „rybak" i wzorca „ba". Program napisz z wykorzystaniem języka C++.
3. Napisz funkcję przeanalizowaną w rozdziale 30.1. z wykorzystaniem języka Java.
4. Napisz funkcję, która będzie przyjmowała tablicę znaków (*z*) i wzorzec (*q*), następnie zwróci liczbę całkowitą (*C*), oznaczającą ilość wystąpień wzorca (*q*) w tablicy (*z*). Przetestuj działanie dla łańcucha znaków „edukacja" i wzorca „pl". Program napisz z wykorzystaniem języka Java.

## PODSUMOWANIE LEKCJI

- **Wzorzec** jest spójnym ciągiem znaków, a raczej podciągiem (podtekstem), który występuje w danym ciągu znaków. Dla każdej *i*-tej litery musimy sprawdzić *m* kolejnych znaków, czy są one zgodne ze znakami we wzorcu. Porównanie jest wykonywane tak długo, dopóki istnieje kolejne *m* liter począwszy od *i*-tego indeksu. ◂ s. 185
- **Wyszukiwanie wzorca** jest jednym z głównych problemów tekstowych badanych przez naukowców, wykorzystywanych m.in. w edytorach tekstów do wyszukiwania informacji w tekście. ◂ s. 186

# 31. Od ostatniego, czyli szukamy wzorca tekstu metodą Boyera–Moore'a

**NA TEJ LEKCJI:**

- poznasz algorytm BM;
- będziesz znał zastosowanie algorytmu BM.

W 1975 r. amerykańscy naukowcy Robert Boyer (czytaj: bojer) i J Strother Moore (czytaj: dżej strater mur) wynaleźli algorytm, który został uznany za najszybszy, a zarazem najpraktyczniejszy algorytm wyszukiwania wzorca, a co najważniejsze – o wiele skuteczniejszy od algorytmu Knutha–Morrisa–Pratta (czytaj: knuta morisa prata) stosowanego do tej pory. Algorytm Boyera–Moore'a obecnie jest wykorzystywany w prawie każdym edytorze tekstu przy wyszukiwaniu tekstów.

## 31.1. Od ostatniego, czyli wyznaczmy wszystkie wystąpienia wzorca w łańcuchu znaków

algorytm Roberta Boyera i J Strothera Moore'a

Algorytm stworzony przez Boyera i Moore'a nazywamy algorytmem BM. Rozpoczyna on porównywanie łańcucha od ostatniego znaku wzorca, czyli odwrotnie niż opisany w rozdziale 30. algorytm. Co uzyskamy w ten sposób? Załóżmy, że ostatni znak wzorca nie będzie się zgadzał ze znakiem szukanym w tekście oraz że znak z przeszukiwanego tekstu nie występuje dalej we wzorcu, to okno wzorca będziemy mogli od razu przesunąć o tyle pozycji, ile znaków zawiera sam wzorzec. W przeciwnym wypadku wzorzec pozycjonujemy w taki sposób, aby ustawić pozycje znaku jednocześnie występującego we wzorcu i w przeszukiwanym tekście. Dzięki takiej konstrukcji algorytm Boyera–Moore'a o wiele szybciej znajduje rozwiązanie.

Wyszukajmy wzorzec ciągu znaków AABCCB w tekście ABBCBDBAABCCBD.

| LP | | Opis |
|---|---|---|
| 1 | ABBCBDBAABCCBD<br>AABCCB | Okno wzorca zostało umieszczone na początku przeszukiwanego ciągu znaków. |
| 2 | ABBCBDBAABCCBD<br>AABCCB | Rozpoczynamy porównanie od ostatniego znaku wzorca. Znaki różnią się od siebie, a dodatkowo znak D nie występuje we wzorcu. |
| 3 | ABBCBDBAABCCBD<br>AABCCB | Przesuwamy nasz wzorzec o 6 kolejnych znaków, ponieważ z tylu znaków składa się nasz wzorzec. |
| 4 | ABBCBDBAABCCBD<br>AABCCB | Ponownie porównujemy ostatni znak w naszym wzorcu z łańcuchem. Znaki się nie zgadzają, ale tym razem litera C występuje w naszym wzorcu. |

| LP | | Opis |
|---|---|---|
| 5 | A B B C B D B A A B C C B D<br>A A B C C B | Okno wzorca przesuwamy w taki sposób, aby litery C, które występują zarówno w sprawdzanym ciągu znaków, jak i we wzorcu, pokrywały się. |
| 6 | A B B C B D B A A B C C B D<br>A A B C C B | Porównujemy kolejne znaki – wszystkie znaki się pokrywają. |

Rys. 31.1. Wyszukiwanie w tekście z wykorzystaniem wzorca i algorytmu BM

## 31.2. Algorytm Boyera–Moore'a dla algorytmu wyszukiwania w tekście z tablicą Ost[]

algorytm Boyera–Moore'a

Tablica *Ost* jest dodatkowym elementem w strukturze danych, którą wykorzystujemy w algorytmie Boyera–Moore'a. Przechowywane są w niej pozycje ostatnich wystąpień we wzorcu *q* wszystkich możliwych znaków alfabetu (w naszym przypadku są tylko cztery znaki). Jeśli danego znaku nie będzie we wzorcu, to pozycja ustawiana jest na –1.

**Przykład:**
Przyjmijmy, że nasza tablica *Ost* będzie zawierała cztery elementy, na przykład znaki ABCD. Dla wzorca ACCBC nasza tablica ma postać:

*Ost*[A:(0), B:(3), C:(4), D:(–1)]

Niech *i* oznacza pozycję początku okna wzorca *q* w przeszukiwanym tekście *z*, a *j* pozycję we wzorcu, w którym występuje niezgodność ze znakiem tekstu.

| LP | Możliwe przypadki | Opis |
|---|---|---|
| 1 | i : 0 1 2 3 4 5 6 7 8 .<br>z : ? B D A C ? ? ? ? ?<br>q : A C C B C<br>j : 0 1 2 3 4<br>A C C B C C | Litera C występuje we wzorcu *q* na pozycji 4. Okna wzorca nie przesuwamy, ponieważ $i = i + j - Ost[C] = 0 + 4 - 4 = 0$. |
| 2 | i : 0 1 2 3 4 5 6 7 8 .<br>z : ? B D A C ? ? ? ? ?<br>q : A C C B C<br>j : 0 1 2 3 4<br>A C C B C C | Litera D nie występuje we wzorcu *q*. Okna wzorca przesuwamy na pozycję 3, ponieważ $i = i + j - Ost[D] = 0 + 2 - (-1) = 3$. |
| 3 | i : 0 1 2 3 4 5 6 7 8 .<br>z : ? B D A C ? ? ? ? ?<br>q : A C C B C<br>j : 0 1 2 3 4<br>A C C B C C | W tym przykładzie zastosowanie naszej reguły daje nam ujemne przesunięcie okna wzorca – wzorzec przesuwamy w lewo: $i = i + j - Ost[B] = 0 + 1 - 3 = -2$.<br><br>Rozwiązaniem będzie zastosowanie takiej reguły, że w przypadku, kiedy wzorzec okna jest niezgodny, przesuwamy go zawsze o większą z dwóch liczb:<br>$i = i + \max(1, j - Ost[B])$<br>$i = i + \max(1, 1, -3) = 0 + \max(1, -2) = 0 + 1 = 1$ |

Rys. 31.2. Wykorzystanie algorytmu BM do wyszukiwania w tekście z użyciem tablicy

## Algorytm Boyera–Moore'a dla algorytmu wyszukiwania w tekście z tablicą Ost[]

**Wejście:**
$q$ – wzorzec
$k_1$ – kod pierwszego znaku
$k_n$ – kod ostatniego znaku alfabetu

**Wyjście:**
*Ost* – tablica o indeksach od 0 do $k_n - k_1$.
$Ost[k_n - k_i] \in C$ i określają ostatnie położenie znaku o kodzie $k_i$ w naszym wzorcu. Takie sformułowanie problemu jest konieczne dla języka C++, który wymaga, aby indeksy tablic rozpoczynały się od 0.

Element pomocniczy:
$i$ – zmienna licznikowa pętli, $i \in N$

**Lista kroków:**
Krok 1: Dla $i = 0, 1, ..., k_n - k_1$, wykonaj $Ost[i] = -1$
Krok 2: Dla $i = 0, 1, ..., |q| - 1$, wykonaj $Ost[\text{kod}(q[i] - k_1] = i$
Krok 3: Zakończ

## Wyszukiwanie wzorca za pomocą uproszczonego algorytmu Boyera–Moore'a

**Wejście:**
$z$ – łańcuch znaków do przeszukania
$q$ – wzorzec
$k_1$ – kod pierwszego znaku
$k_n$ – kod ostatniego znaku alfabetu

**Wyjście:**
Kolejne pozycje wystąpień wzorca w łańcuchu znaków. Wartość –1 nie wskaże nam żadnej pozycji wzorca i będzie oznaczała, że wzorzec nie pojawia się w sprawdzanym łańcuchu.

**Elementy pomocnicze:**
*Ost* – tablica ostatnich pozycji wszystkich znaków we wzorcu $q$, $Ost \in C$
$i$ – pozycja okna wzorca w łańcuchu $z$, $i \in N$
$j$ – pozycja znaku we wzorcu $q$, który będziemy porównywać ze znakiem łańcucha $z$, $j - N$
max($a$, $b$) – funkcja zwracająca większą z liczb $a$ i $b$
$pq$ – zawiera pozycję wzorca $q$ w łańcuchu $z$, $pq - N$
$d$ – długość wzorca $q$, $d - N$
$e$ – długość łańcucha $z$, $e - N$

**Lista kroków:**
Krok 1: Dla $k_1$ i $k_n$ oblicz tablicę *Ost*
Krok 2: $d = |q|$
Krok 3: $e = |z|$

Krok 4: $pq = -1$
Krok 5: $i = 0$
Krok 6: Dopóki $i \leq e - d$, wykonuj kroki od Krok 7 do Krok 13
Krok 7: $j = d$
Krok 8: Dopóki $(j > -1)$ && $(q[j] = z[i + j])$, wykonuj $j = j - 1$
Krok 9: Jeśli $j > -1$, to idź do Krok 12
Krok 10: $pq = i$
Krok 11: Wypisz $pq$
Krok 12: $i = i + \max(1, j - Ost[z(i + j)])$
Krok 13: Kontynuuj pętlę od Kroku 6
Krok 14: Jeśli $pq = -1$, to wypisz $-1$
Krok 15: Zakończ

## ZADANIA DO WYKONANIA

1. Użyj listy kroków z podrozdziału *Algorytm Boyera–Moore'a dla algorytmu wyszukiwania w tekście z tablicą Ost[]* i stwórz algorytm wyszukujący wzorzec z wykorzystaniem algorytmu BM za pomocą schematu blokowego.
2. Użyj listy kroków i schematu blokowego stworzonego w zadaniu 1. i napisz program komputerowy wyszukujący wzorzec z wykorzystaniem algorytmu BM za pomocą języka C++.
3. Użyj listy kroków z podrozdziału *Wyszukiwanie wzorca za pomocą uproszczonego algorytmu Boyera–Moore'a* i stwórz algorytm wyszukujący wzorzec z wykorzystaniem uproszczonego algorytmu BM za pomocą schematu blokowego.
4. Użyj listy kroków i schematu blokowego stworzonego w zadaniu 3. i napisz program komputerowy wyszukujący wzorzec z wykorzystaniem uproszczonego algorytmu BM za pomocą języka Java.

## PODSUMOWANIE LEKCJI

- **Algorytm Roberta Boyera i J Strothera Moore'a** jest uznawany za najszybszy, a zarazem najpraktyczniejszy algorytm wyszukiwania wzorca. ◂ s. 188
- Algorytm BM rozpoczyna porównywanie łańcucha od ostatniego znaku wzorca, dzięki czemu o wiele szybciej znajduje rozwiązanie. ◂ s. 188
- **Algorytm Boyera i Moore'a** (algorytm BM) jest wykorzystywany obecnie w prawie każdym edytorze tekstu przy wyszukiwaniu tekstów. ◂ s. 189

# 32. Z lewej i z prawej, czyli szukamy palindromów

**NA TEJ LEKCJI:**

- dowiesz się, czym jest palindrom i jakie ma zastosowanie w informatyce;
- odkryjesz, czym jest algorytm naiwny.

palindrom ▸ Palindrom (gr. *palindromeo* – biec z powrotem) to wyrażenie brzmiące tak samo czytane od lewej do prawej i od prawej do lewej. Przykładem takiego wyrazu jest „kajak".

Dzieje palindromu sięgają starożytności. Być może powstał jako zabawa słowna, a może od razu niektóre z palindromów miały być czymś w rodzaju szyfru, może zaklęcia. Współcześnie palindrom to przede wszystkim rozrywka umysłowa i żart słowny, który bawi głównie kontrastem między idealną dokładnością w postaci symetrii a chaosem często bezsensownych zwrotów i zdań powstających po jego skonstruowaniu.

## 32.1. Od strony lewej do prawej i na odwrót

Załóżmy, że chcemy sprawdzić, czy podany przez nas łańcuch *z* jest palindromem oraz czy zawiera palindromy o długości większej niż 1. Mamy dwa łańcuchy znaków: palindrom KAJAK i AJKAJ, który palindromem nie jest.

Wprowadźmy symbol łańcucha znaków *zP*, który ma odwróconą kolejność znaków w stosunku do łańcucha *z*.

**Przykład:**
*z* = SNOP
*zP* = PONS

Łańcuch znaków będzie palindromem, jeśli da się rozłożyć na dwa podłańcuchy *y* i *yP* według poniższego schematu:

*z* = *yyP* – palindrom parzysty
*z* = *yXyp* – palindrom nieparzysty, gdzie *X* jest dowolną literą alfabetu

**Przykład:**
SNOPPONS – palindrom parzysty, ponieważ: SNOP + PONS
SNOP*Y*PONS – palindrom nieparzysty, ponieważ: SNOP + *Y* + PONS

algorytm naiwny ◂ Do rozwiązania problemu zastosujmy algorytm naiwny. Algorytm ten nazywamy naiwnym ze względu na fakt, że odzwierciedla on ludzki sposób myślenia, który twórcy przenoszą na jego działanie. Nie stosuje się w nim zaawansowanych

i skomplikowanych obliczeń matematycznych, co ma przełożenie na jego niską wydajność. Algorytm naiwny jest najprostszą metodą na znalezienie wzorca w tekście. Jego zasada działania opiera się na porównaniu odpowiednich liter tekstu i wzorca. Algorytmy naiwne są algorytmami o małej wydajności. Aby rozwiązać problem, muszą one wykonać większą liczbę powtórzeń.

Algorytm naiwny rozłoży nam nasz łańcuch *z* na wszystkie możliwe podłańcuchy (zestawy identycznych znaków występujących w badanym łańcuchu) *y* o długości nie mniejszej niż 2 znaki – zgodnie z założeniem naszego zadania. Następnie sprawdzi, czy powstałe podłańcuchy dadzą się przedstawić w postaci *yyP* lub *yXyP*. Sprawdzenie tego będzie polegać na porównaniu znaków od początku do końca powstałego podłańcucha, przy wykorzystaniu dwóch indeksów. Pierwszy ustawia się na pierwszym znaku podłańcucha *r*, a drugi na ostatnim. Następnie porównamy wskazane przez te indeksy znaki podłańcucha *r*. Jeśli znaki będą różne, to nasz podłańcuch nie jest palindromem. W przeciwnym wypadku – jeśli znaki są równe – to indeksy przesuwamy – lewy w prawo, a prawy w lewo. Jeśli indeksy się miną, to oznacza, że zachodzi jedna z dwóch równości:

$r = yyP$ lub $r = yXyP$

Jeżeli jedna z powyższych równości będzie zachodziła, to nasz podłańcuch jest palindromem.

Algorytm naiwny wyszukiwania palindromów:

**Wejście:**
*z* – łańcuch znaków

◂ łańcuch znaków

**Wyjście:**
Wszystkie palindromy zawarte w łańcuchu *z*

**Elementy pomocnicze:**
*i*, *j* – indeksy znaków w łańcuchu $z$ – $i, j \in N$
*n* – długość łańcucha *z*, $d \in N$
*iP* – indeks prawy, $iP \in N$
*iL* – indeks lewy, $iL \in N$

**Lista kroków:**
Krok 1: $n = |z|$
Krok 2: Dla $i = 0, 1, \ldots n-2$, wykonaj kroki od Krok 3 do Krok 10
Krok 3: Dla $j = i + 2, i + 3, \ldots, n-1$, wykonaj kroki od Krok 4 do Krok 10
Krok 4: $iL = i$
Krok 5: $iP = j - 1$
Krok 6: Dopóki $iL < iP$ wykonaj kroki od Krok 7 do Krok 9
Krok 7: Jeśli $z[iL] \neq z[iP]$, to wróć do Krok 3
Krok 8: $iL = iL + 1$
Krok 9: $iP = iP - 1$
Krok 10: Wypisz $z[i, j]$
Krok 11: Zakończ

Palindromy to nie tylko wyrazy, które czyta się tak samo od lewej do prawej i na odwrót. W każdym języku występują również całe zdania, które w ten sposób można przeczytać, na przykład: „Ikar łapał raki".

## ZADANIA DO WYKONANIA

1. Napisz program komputerowy w języku C++, który stworzy losowy łańcuch znaków składający się ze znaków: A, B, C, D. Łańcuch ma mieć długość 30 znaków. Następnie wyszukaj w nim wszystkie palindromy i je wypisz.
2. Stwórz program w języku C++, który obliczy, ile podłańcuchów będących palindromami występuje we wprowadzonym przez ciebie ciągu znaków.
3. Stwórz program komputerowy w języku Java, który sprawdzi, czy wprowadzony przez ciebie ciąg znaków jest palindromem, oraz wypisze informację, czy jest to palindrom parzysty czy nieparzysty.
4. Napisz program w języku Java, który wczytuje slowo ze standardowego wejścia i bada, czy da się je rozłożyć na palindromy parzyste. Jeśli nie, to zapisuje w standardowym wyjściu tylko jeden wyraz „NIE", a jeśli da się rozłożyć, to zapisuje jego rozkłady na minimalną i maksymalną liczbę palindromów parzystych. Wiadomości potrzebnych do rozwiązania tego zadania poszukaj w internecie.

## PODSUMOWANIE LEKCJI

s. 192 ▸ • **Palindrom** to wyrażenie brzmiące tak samo czytane od lewej do prawej i od prawej do lewej. Przykładem palindromu jest wyraz „kajak".

s. 192 ▸ • Do wyszukiwania palindromów wykorzystuje się **algorytm naiwny.**

s. 193 ▸ • **Łańcuch znaków** jest palindromem, jeśli da się rozłożyć na dwa podłańcuchy *y* i *yP* według schematu:

$z = yyP$ – palindrom parzysty

$z$ = yXyp – palindrom nieparzysty, gdzie $X$ jest dowolną literą alfabetu

# 33. Przestawianie liter, czyli jak stworzyć anagram

**NA TEJ LEKCJI:**

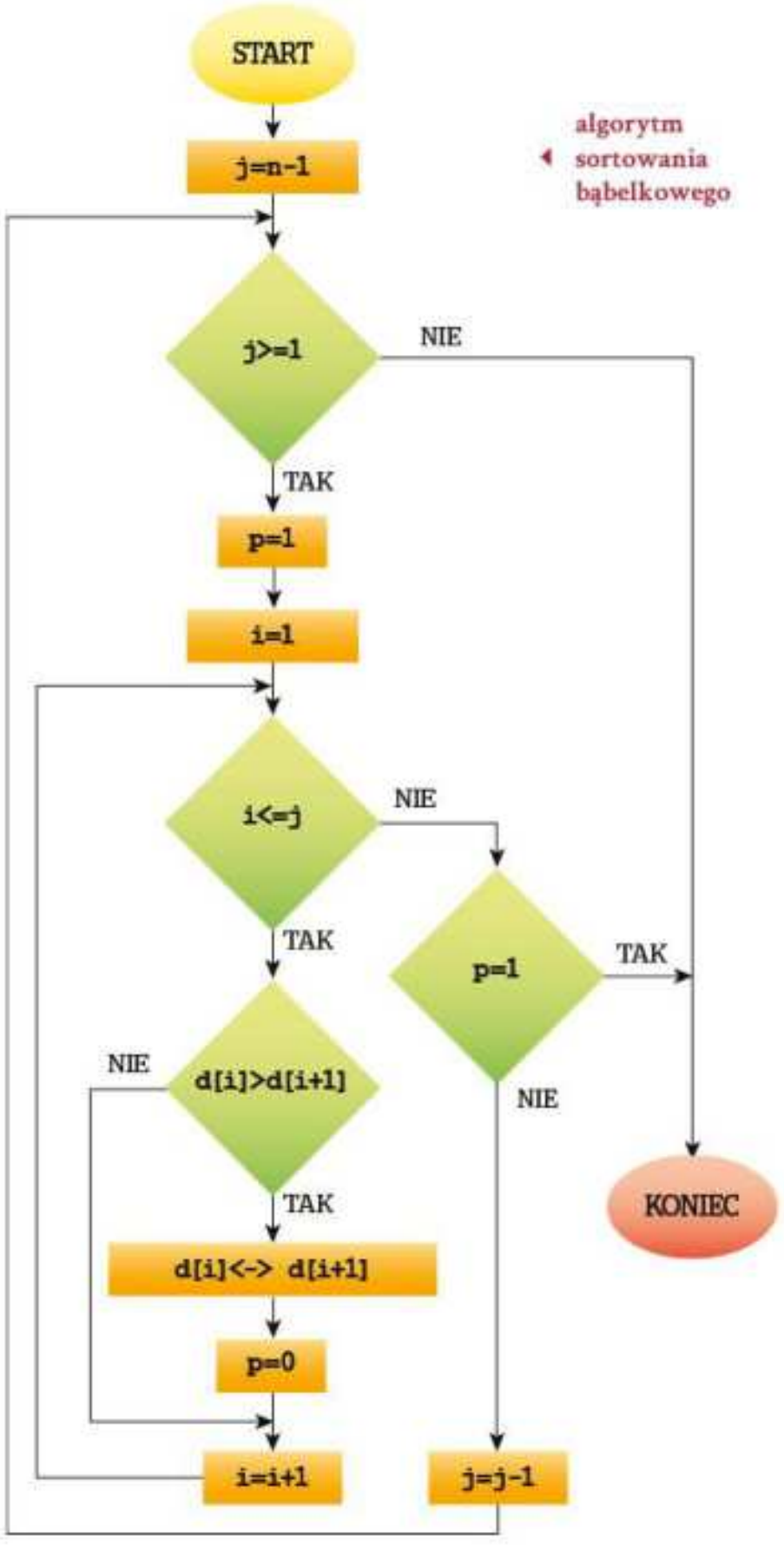
- dowiesz się, czym jest anagram i jakie ma zastosowanie w informatyce;
- przypomnisz sobie, czym jest algorytm sortowania bąbelkowego.

Nazwa anagram wywodzi się od greckich słów *ana*, czyli „nad" i *gramma*, czyli „litera". Anagram to wyraz, wyrażenie lub całe zdanie, które powstaje przez przestawienie liter lub sylab innego wyrazu lub zdania – warunkiem jest wykorzystanie wszystkich liter lub sylab z wyrazu (zdania) wyjściowego.

## 33.1. Wróćmy do sortowania bąbelkowego

Aby zająć się problemem anagramów, musimy przypomnieć sobie algorytm sortowania bąbelkowego. Jest on jednym z najstarszych algorytmów sortujących. Jego działanie opiera się na cyklicznym porównaniu par sąsiadujących ze sobą elementów i zmianie ich kolejności w przypadku, kiedy nie jest spełnione kryterium porządkowania zbioru. Całą operację powtarza się tak długo, aż cały zbiór zostanie posortowany.

Schemat blokowy sortowania bąbelkowego przedstawiono na rysunku 33.1.

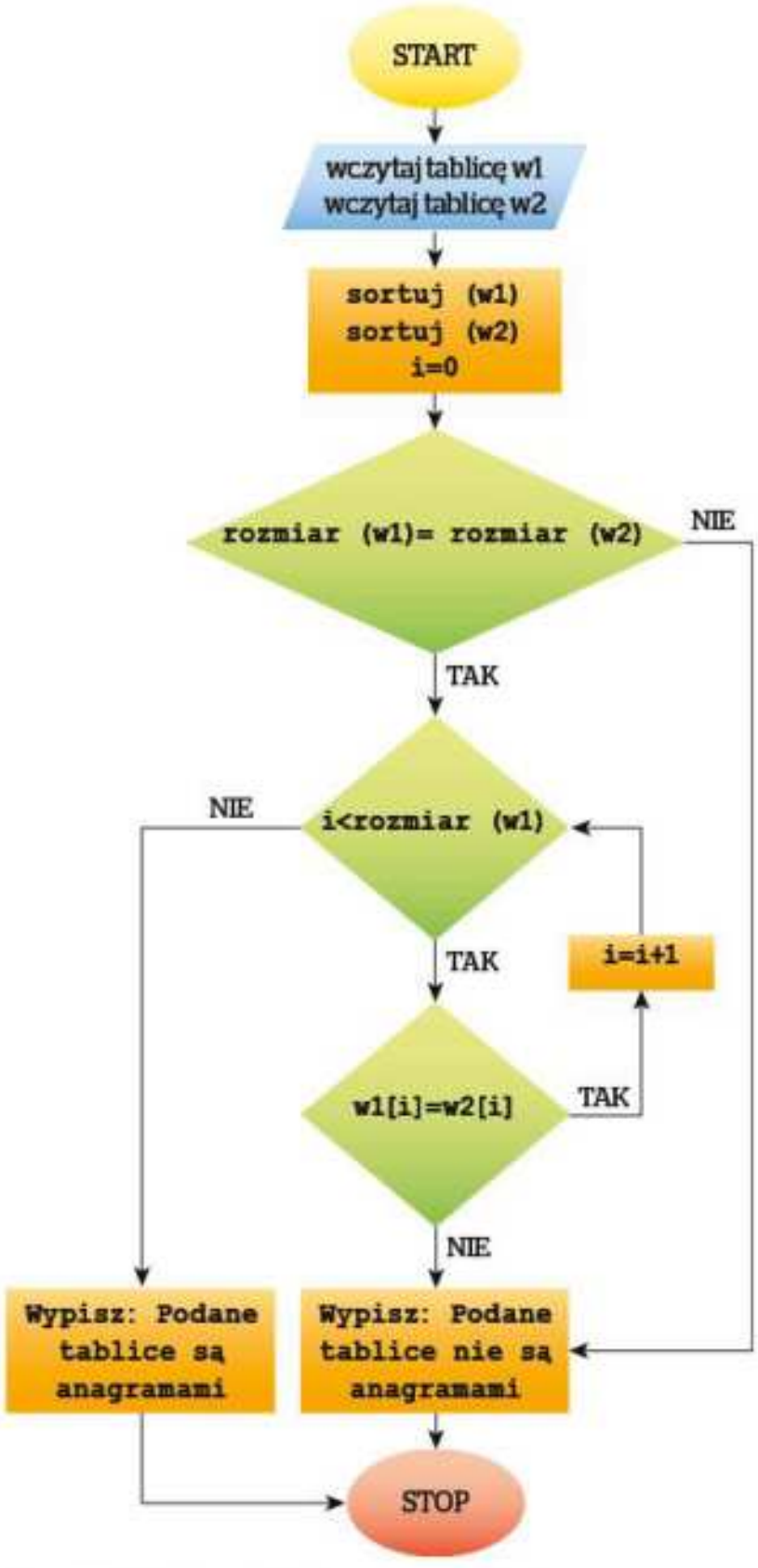



algorytm sortowania bąbelkowego

Rys. 33.1. Schemat blokowy sortowania bąbelkowego

## 33.2. Anagram

anagram ▸ Anagram to wyraz bądź wyrażenie powstałe z innego wyrazu bądź wyrażenia, które zawiera wszystkie litery z wyrazu „matki". Przykładami anagramów są: „matura – trauma", „Karol – rolka", „Adam – dama", „arbuz – burza".




Rys. 33.2. Schemat blokowy sprawdzający za pomocą sortowania bąbelkowego, czy tablica jest anagramem

Jeżeli chcemy się dowiedzieć, czy dany wyraz jest anagramem innego, na początku musimy sprawdzić, czy długość obu wyrazów jest identyczna. Jeżeli ich długości się różnią, to nie mamy do czynienia z anagramem. Kolejnym etapem będzie sortowanie znaków w wyrazie lub ciągu liczb, gdyż one też mogą być używane jako anagramy. Możemy je posortować alfabetycznie lub w przypadku liczb od najmniejszej do największej. Jeżeli utworzą się identyczne łańcuchy znaków, mamy do czynienia z anagramami.

**Przykład:**
Mamy wyraz „arbuz" utworzony z niego anagram to „burza". Gdy posortujemy litery alfabetycznie, otrzymamy dwa identyczne ciągi – „abruz".

Schemat blokowy takiego sortowania przedstawiono na rysunku 33.2.

## ZADANIA DO WYKONANIA

1. Napisz program komputerowy w języku C++, który sprawdzi, czy wczytane dwa łańcuchy znaków są anagramami.
2. Napisz program w języku C++, który sprawdzi, czy słowa są anagramami, wykreślając ze słowa *a* po kolei litery znajdujące się w słowie *b*.
3. Napisz program komputerowy w języku Java, który sprawdzi, czy wczytane dwa łańcuchy liczb są anagramami.
4. Napisz program w języku Java, który sprawdzi, czy słowa są anagramami, wykreślając ze słowa *a* po kolei litery znajdujące się w słowie *b*.

## PODSUMOWANIE LEKCJI

- Aby sprawdzić, czy wyraz jest anagramem, musimy użyć **sortowania bąbelkowego**. Jego działanie opiera się na cyklicznym porównaniu par sąsiadujących ze sobą elementów i zmianie ich kolejności w przypadku, gdy nie jest spełnione kryterium porządkowego zbioru. ◂ s. 195
- **Anagram** to wyraz, wyrażenie lub zdanie, które powstaje przez przestawienie liter lub sylab innego wyrazu, wyrażenia lub zdania – warunkiem jest wykorzystanie wszystkich liter lub sylab z wyrazu (zdania) wejściowego. ◂ s. 196

# 34. Od A do Z, czyli alfabetyczne sortowanie wyrazów

**NA TEJ LEKCJI:**

- dowiesz się, czym jest sortowanie alfabetyczne; jednopoziomowe oraz przez wybór.

sortowanie alfabetyczne

Sortowanie alfabetyczne to po prostu alfabetyczne porządkowanie terminów i haseł według określonych reguł. Zagadnienie samego porządkowania możemy rozpatrywać na kilku płaszczyznach. Wykorzystywane jest jako porządkowanie wyrazów w słownikach bądź w skorowidzach.

sortowanie jednopoziomowe

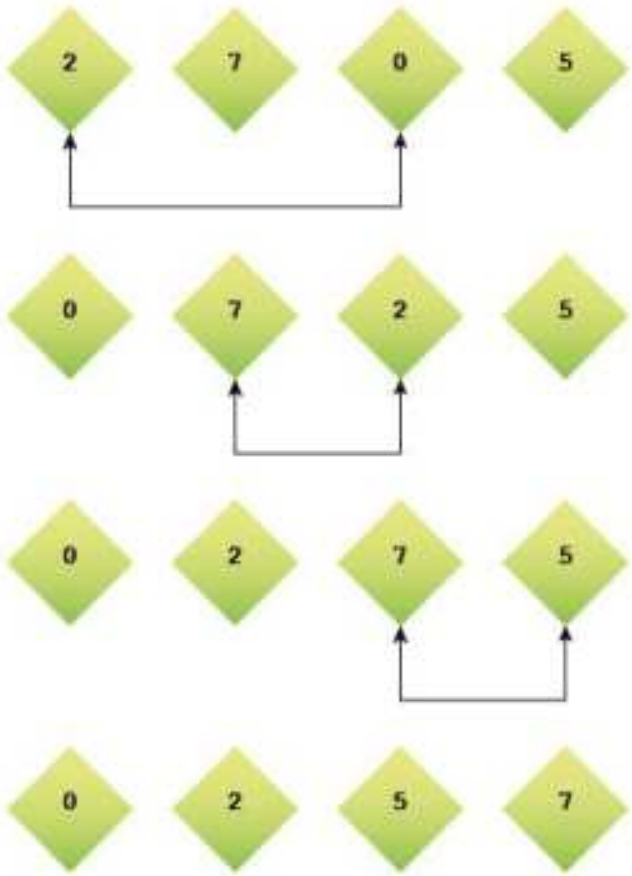
W informatyce najczęściej spotykamy się z sortowaniem w arkuszach kalkulacyjnych lub edytorach tekstu, w których stosuje się sortowanie jednopoziomowe, czyli liczby od najmniejszej do największej, lub sortowanie alfabetyczne od A do Z.

## 34.1. Sortowanie przez wybór

sortowanie przez wybór

W metodzie sortowania przez wybór, zwanej też sortowaniem przez wymianę, szukamy na początku elementu najmniejszego. Po jego znalezieniu podmieniamy go z pierwszym elementem w tablicy. Kolejnym krokiem będzie znowu szukanie najmniejszego elementu w naszym zbiorze, z pominięciem elementu pierwszego i wstawienie go na pozycję drugą. Czynność tę powtarzamy do otrzymania jednoelementowego podzbioru.

Przedstawiony obok rysunek pokazuje, jak wygląda sortowanie zbioru rosnącego. Jeżeli chcemy nasz zbiór posortować malejąco, to zamiast najmniejszego elementu, szukamy elementu największego. Pozostała część procedury się nie zmienia.



Rys. 34.1. Sortowanie przez wybór

Algorytm sortowania przez wybór – lista kroków:

**Wejście:**
$n$ – liczba elementów w zbiorze, który będziemy sortować, $n \in N$
$t[]$ – tablica $n$-elementowa, która będzie sortowana

**Wyjście:**
$t[]$ – posortowana tablica $n$-elementowa

**Elementy pomocnicze:**
$i, j$ – zmienne sterujące sortowaniem, $i, j \in N$
$min$ – pozycja elementu minimalnego w tablicy $d[]$, $min \in N$

**Lista kroków:**
Krok 1: Dla $j = 1,2, ..., n-1$ wykonaj kroki od Krok 2 do Krok 4
Krok 2: $min = j$
Krok 3: Dla $i = j + 1, j + 2, ..., n$ jeżeli $t[i] < t[min]$, to $min = i$
Krok 4: $t[j] \leftrightarrow t[min]$
Krok 5: Koniec

Pętla zewnętrzna, która steruje zmienną *j*, wyznacza kolejne elementy naszego zbioru o indeksach od 1 do $n-1$, w których będą umieszczone najmniejsze elementy. Na początku tej pętli zakładamy, że elementem minimalnym jest element $t[j]$ i zapamiętujemy indeks zmiennej *min*.

W drugiej pętli, sterowanej zmienną *i*, porównujemy pozostałe elementy zbioru z elementem, który już wybraliśmy – $t[min]$. Jeśli element zbioru $t[i]$ będzie mniejszy od elementu wybranego ($t[min]$), to znaleźliśmy następny element minimalny. Zapamiętujemy jego pozycję w *min* i kontynuujemy drugą pętlę.

Po zakończeniu drugiej pętli – wewnętrznej – *min* zawiera indeks elementu minimalnego. Zamieniamy miejscami element $t[j]$ z elementem $t[min]$. Po tej operacji element minimalny znajduje się na swoim docelowym miejscu. Zwiększamy *j*, przechodzimy do kolejnego elementu naszego zbioru i kontynuujemy pętlę zewnętrzną.

Sortowanie przez wybór – program w języku C++:

```
#include<iostream>
using namespace std;
int main()
{
   int ilosc_liczb, max, i,j, pomoc;
   cout << "Podaj ilosc liczb do posortowania: ";
   cin >> ilosc_liczb;
   int *tablica = new int [ilosc_liczb];
   for (i = 0; i < ilosc_liczb; i++)
   {
      cout << "Podaj liczba: ";
      cin >> tablica[i];
   }
   for (i = 0; i < ilosc_liczb; i++)
   {
```

```
            max = i; // indeks najmniejszego znalezionego elementu
            for (j = i+1; j < ilosc_liczb; j++)
            {
               if (tablica[j] < tablica[max])
                  max = j;
            }
            pomoc = tablica[max];
            tablica[max] = tablica[i];
            tablica[i] = pomoc;
      }
      for (i = 0; i < ilosc_liczb; i++)
      cout << "tablica[" << i << "] = " << tablica[i] << endl;
      delete [] tablica;
      return 0;
}
```

Sortowanie przez wybór – program w języku Java:

```
public class Select {
   private static int t[];
   private static int n;
   private static void selectionsort(int t[], int n) {
      int min,i,j,temp;
      for (i=0;i<n-1;i++) {
         min=i;
         for (j=i+1;j<n;j++)
            if (t[j]<t[min])
               min=j;
      }
      temp=t[min];
      t[min]=t[i];
      t[i]=temp;
   }

   public static void main(String[] args) {
      int i,liczba;
      System.out.println("Ile liczb chcesz posortowac?");
      n = Console.readInt("?");
      t = new int[n];
      for(i=0; i<n; i++) {
         t [i] = Console.readInt("Wprowadz liczbe #" + (i+1));
      }
```

```
        System.out.println("Tablica przed posortowaniem:");
        for(i=0; i<n; i++)
           System.out.println(t [i]);
        selectionsort(t,n);
        System.out.println("Tablica po posortowaniu:");
        for(i=0; i<n; i++)
           System.out.println(t[i]);
        }
    }
}
```

## ZADANIA DO WYKONANIA

1. Na podstawie sortowania przez wybór uzasadnij, że czas sortowania zależy od rozkładu elementów w porządkowanym zbiorze.
2. Napisz program przy wykorzystaniu języka C++, który wyświetli wprowadzony łańcuch znaków przed sortowaniem i po nim.
3. Napisz program w języku C++ tworzący w pamięci komputera dziesięcioelementową tablicę liczb typu double (w której będą przechowywane liczby zmiennoprzecinkowe) i zapełnij ją (z użyciem pętli for) liczbami losowymi z przedziału [0, 1]. Następnie (z użyciem drugiej pętli for) wypisz zawartość tablicy. Na koniec uzupełnij kod programu tak, aby program znajdował i wypisywał indeks najmniejszego elementu.
4. Napisz w języku Java program, który posortuje w porządku malejącym tablicę 10 liczb podanych z klawiatury przez użytkownika.

## PODSUMOWANIE LEKCJI

- **Sortowanie alfabetyczne** to porządkowanie alfabetyczne terminów i haseł według określonych reguł. ◂ s. 198
- W informatyce najczęściej jest stosowane sortowanie w arkuszach kalkulacyjnych lub edytorach tekstu, w których jest wykorzystywane **sortowanie jednopoziomowe**. ◂ s. 198
- W programowaniu w celu sortowania alfabetycznego wykorzystujemy **sortowanie przez wybór**, zwane też sortowaniem przez wymianę. ◂ s. 198

# 35. Odwrotna notacja polska, czyli zapisujemy wyrażenia arytmetyczne

**NA TEJ LEKCJI:**

- dowiesz się, czym jest i do czego służy zapis postfiksowy;
- poznasz wyrażenie ONP.

zapis postfiksowy

Odwrotna notacja polska jest sposobem zapisu wyrażeń arytmetycznych, w których znak wykonywanej operacji umieszcza się po operandach (liczbach lub danych, na których wykonuje się operacje) – to tzw. zapis postfiksowy – a nie między nimi, jak jest robione standardowo. Odwrotna notacja polska stanowi przeciwieństwo notacji polskiej, w której znaki wykonywanej operacji zapisuje się przed operandami.

Zapis w odwrotnej notacji polskiej (ONP lub RPN od angielskiego określenia reverse Polish notation, czytaj: rewers polisz notejszyn) pozwala na całkowitą rezygnację z użycia nawiasów w wyrażeniach, jako że kolejność wykonywanych działań jest jednoznacznie określona.

## 35.1. Odwrotna notacja polska

Odwrotna notacja polska to ogromne ułatwienie wykonywania obliczeń przez komputer, ponieważ nie stosuje się w niej nawiasów i zachowuje się kolejność wykonywanych działań. Algorytmy obliczania wartości wyrażenia w ONP są bardzo proste i wykorzystują stos, czyli dane są dokładane na wierzch stosu i pobierane z wierzchołka.

Odwrotna notacja polska została opracowana przez australijskiego naukowca Charlesa Hamblina (czytaj: czarlsa hamblina) jako odwrócenie beznawiasowe notacji polskiej Jana Łukasiewicza.

Zapis konwencjonalnego wyrażenia arytmetycznego (3 + 6) * 5 w ONP wygląda następująco: 3 6 + 5 *, natomiast ((3 + 9) / 3 + (10 – 5) * 4) / 2 w ONP to: 3 9 + 3 / 10 5 – 4 * + 2 /.

W zapisie konwencjonalnym operatory matematyczne (znaki dzielenia, mnożenia, dodawania i odejmowania) są umieszczane między argumentami. Operatory mają również swoje priorytety, czyli ważność. Mogą być również łączone w prawo lub w lewo. Jeśli w wyrażeniach arytmetycznych mamy operatory o różnych priorytetach, to najpierw będą wykonywane te o wyższym priorytecie, na przykład:

3 + 6 * 5 = 3 + 30 = 33

Jeśli chcemy zmienić kolejność wykonywania działań, użyjemy nawiasów:

(3 + 6) * 5 = 9 * 5 = 45

Podczas stosowania odwrotnej notacji polskiej nie mamy tego problemu, nawiasy stają się zbędne:

3 + 6 * 5 = 3 6 5 * + = 3 30 + = 33
(3 + 6) * 5 = 3 6 + 2 * = 9 5 * = 45

Jeżeli chcemy użyć ONP do obliczenia wartości wyrażenia, będziemy potrzebowali stosu. Zasada jego zastosowania jest prosta. Wyrażenie ONP zaczynamy przeglądać od lewej strony do prawej. Jeżeli napotkamy liczbę, to umieszczamy ją na samym szczycie stosu, czyli liniowej strukturze danych (w których każdy element, poza pierwszym i ostatnim, posiada swojego poprzednika i następcę). Jeśli napotkamy operator, to ze stosu pobieramy dwie ostatnio położone na stosie liczby i wykonujemy na nich działanie arytmetyczne zgodne z napotkanym operatorem. Wynik działania umieszczamy z powrotem na stosie. Oczywiście na jego szczycie. Gdy nasze wyrażenie przejrzymy do końca, to na szczycie stosu będziemy mieli jego wynik.

| Wyrażenie ONP | | | | | Element | Operacja | Stos |
|---|---|---|---|---|---|---|---|
| 3 | 6 | 5 | * | + | | | ——— |
| 3 | 6 | 5 | * | + | 3 | na stos | 3<br>——— |
| 3 | 6 | 5 | * | + | 6 | na stos | 6<br>3<br>——— |
| 3 | 6 | 5 | * | + | 5 | na stos | 5<br>6<br>3<br>——— |
| 3 | 6 | 5 | * | + | * | pobierz 5 i 6<br>pomnóż 5 * 6<br>wynik na stos | 30<br>3<br>——— |
| 3 | 6 | 5 | * | + | + | pobierz 30 i 3<br>dodaj 30 + 3<br>wynik na stos | 33<br>——— |

Rys. 35.1. Działanie arytmetyczne z zastosowaniem ONP

W omawianym w tym rozdziale algorytmie musimy przyjąć pewne założenia. Będziemy sprawdzać algorytm na prostych czterech operatorach: dodawaniu (+), odejmowaniu (–), mnożeniu (*) i dzieleniu (/). Wyrażenie, które będzie poddane analizie, musi być poprawne – algorytm nie będzie sprawdzał jego poprawności. Podczas korzystania z algorytmu ONP należy pamiętać, że liczby muszą mieć postać akceptowalną przez dany język programowania, w którym będziemy tworzyć nasz program. Każdy wprowadzany element zostanie umiejscowiony w osobnym wierszu – w ten sposób nie będziemy borykać się z problemem analizy naszego tekstu pod kątem zawartości w nim liczb i operatorów. W naszym algorytmie też będziemy musieli rozpoznać, czy wprowadzony element jest liczbą, znakiem „=" czy operatorem.

W języku C++ do tego celu najczęściej wykorzystuje się własności strumienia (ciągu bajtów o nieokreślonej długości), który przy złej konwersji (zmianie formy danych) da nam wartość 0.

wyrażenie ONP ▸ Algorytm obliczenia wartości wyrażenia ONP:

**Wejście:**

stos ▸ *S* – stos liczb zmiennoprzecinkowych (liczby rzeczywiste zapisane w postaci wykładniczej 1=1*10^0 lub notacji naukowej 1 = 1E0)

Kolejne elementy naszego wyrażenia będą odczytywane ze standardowego wejścia.

**Wyjście:**
Wartość wyrażenia ONP na szczycie stosu *S*

**Elementy pomocnicze:**
*c* – przechowuje odczytaną informację na wejściu jako łańcuch tekstowy
*a*1, *a*2 – elementy przechowujące argumenty operacji

**Lista kroków:**
Krok 1: Wczytaj *c*
Krok 2: Jeśli *c* = „=", to zakończ
Krok 3: Jeśli *c* jest liczbą, to idź do Krok 9
Krok 4: *a*2 = *S*.top()
*S*.pop()
Krok 5: *a*1 = *S*.top()
*S*.pop()
Krok 6: Wykonaj operację na *a*1 i *a*2 zgodnie z zawartością *c*. Wynik umieść w *a*1
Krok 7: *S*.push(*a*1)
Krok 8: Wróć do Krok 1
Krok 9: Przekształć *c* na liczbę w *a*1
Krok 10: Idź do Krok 7

## ZADANIA DO WYKONANIA

1. Napisz program w języku C++, który wczytuje ze standardowego wejścia kolejne wyrażenia ONP i oblicza ich wartość. Wynik jest zapisywany po wprowadzeniu znaku „=". Program powinien wtedy kończyć działanie.
2. Zbuduj wyrażenie w postaci ONP dla następującego wyrażenia:
((3 + 4) * 5 – (3 + 1))/2
3. Zbuduj wyrażenie w postaci infiksowej (z nawiasami) dla wyrażenia w postaci ONP:
3 3 3 + * 6 9 3 2 + / – +
4. Rozpisz w formie algorytmu poszczególne kroki dla wyrażenia:
6 3 / 2 5 + *. Do rozpisania wyrażenia użyj tabelki jak w rozdziale 35.1.

## PODSUMOWANIE LEKCJI

s. 202 ▸ • Odwrotna notacja polska jest sposobem zapisu wyrażeń arytmetycznych, w których znak wykonywanej operacji umieszcza się po operandach – to tzw. **zapis postfiksowy** – a nie między nimi, jak jest robione standardowo.

s. 204 ▸ • **ONP** jest ułatwieniem wykonywania obliczeń przez komputer, ponieważ nie stosuje się w niej nawiasów i zachowuje się kolejność wykonywanych działań.

s. 204 ▸ • Do obliczeń wyrażenia ONP jest wykorzystywany **stos**.

# 36. Naiwny jak algorytm, czyli szukamy najmniejszego i największego elementu metodą naiwną

**NA TEJ LEKCJI:**

- dowiesz się, czym jest i do czego służy sortowanie naiwne;
- poznasz wersje rekurencyjną i iteracyjną sortowania naiwnego.

Sortowanie naiwne jest jednym z mniej efektywnych algorytmów, lecz dającym zawsze poprawne wyniki. Ten algorytm działa według bardzo prostej zasady – jest wyznaczany najmniejszy bądź największy element zbioru i przenoszony na jego początek (koniec).

## 36.1. Sortowanie naiwne

sortowanie naiwne

Działanie algorytmu sortowania naiwnego polega na przeglądaniu kolejnych par sąsiednich elementów sortowanego zbioru. Jeżeli para aktualnie badanych elementów jest w złej kolejności, zamieniamy je miejscami i przeglądamy ponownie od początku zbioru. Przeglądanie trwa tak długo, aż wszystkie pary zostaną przebadane, a zbiór posortowany. Naiwność tego algorytmu polega na tym, że po znalezieniu nieposortowanych elementów algorytm zamienia je miejscami i rozpoczyna całą procedurę od początku zbioru. Sortowanie odbywa się w miejscu (oznacza to, że nie wymaga wykorzystania dodatkowej pamięci).

wersja rekurencyjna i iteracyjna

Sortowanie naiwne występuje w wersjach rekurencyjnej (odwoływanie się np. funkcji lub definicji do samej siebie) i iteracyjnej (powtarzanie tej samej operacji w pętli określoną liczbę razy lub aż do spełnienia określonego warunku). Wersja iteracyjna jest lepsza, ponieważ nie zajmuje dodatkowo pamięci na kolejne poziomy wywołań rekurencyjnych.

Lista kroków sortowania naiwnego:

**Wejście:**
$n$ – liczba elementów w sortowanym zbiorze, $n \in N$
$t[]$ – zbiór $n$-elementowy, elementy zbioru mają indeksy od 1 do $n$

**Wyjście:**
$t[]$ – posortowany zbiór $n$-elementowy, elementy zbioru mają indeksy od 1 do $n$

**Elementy pomocnicze:**
$i$ – zmienna sterująca pętli, $i \in$ N

**Lista kroków:**
Krok 1: Dla $i = 1, 2, \ldots n - 1$ wykonaj Krok 2
Krok 2: Jeśli $t[i] > t[i + 1]$, to $t[i] \leftrightarrow t[i + 1]$ i idź do Krok 1
Krok 3: Koniec

Działanie algorytmu zaczynamy przejrzeniem zbioru od pierwszego elementu – indeks *i* przyjmuje wartość 1. W pętli sprawdzamy kolejność elementu *t*[*i*] z kolejnym elementem *t*[*i* + 1]. Przyjęliśmy porządek rosnący, więc w tym przypadku *t*[*i*] > *t*[*i* + 1] elementy są w złej kolejności. Jeśli przyjmiemy, że będziemy sortować elementy w porządku malejącym, to należy zmienić relację większości na relację mniejszości.

Zamieniamy teraz elementy miejscami, indeks *i* ustawiamy ponownie na 1 i wracamy na początek pętli.

Jeżeli porównywane przez nas elementy będą w dobrej kolejności, to zwiększamy indeks *i* o 1. Sprawdzamy, czy indeks osiągnął już wartość końcową. Jeśli nie, to wracamy na początek pętli. W przeciwnym razie nasz zbiór jest już posortowany i kończymy działanie algorytmu.

## ZADANIA DO WYKONANIA

1. Napisz w języku C++ program, który będzie realizował algorytm sortowania naiwnego.
2. W celach badawczych napisz program algorytmu sortowania naiwnego, który będzie testował czas wykonania algorytmu. Ze względu na zbyt długi czas sortowania ogranicz zbiór do 8000 elementów.

## PODSUMOWANIE LEKCJI

s. 205 ▸ • **Sortowanie naiwne** wyznacza największy i najmniejszy element zbioru, przenosząc na początek lub na koniec zbioru badany element.

s. 205 ▸ • Sortowanie naiwne występuje w wersjach rekurencyjnej i iteracyjnej. **Wersja iteracyjna** jest lepsza, ponieważ nie zajmuje dodatkowo pamięci na kolejne poziomy wywołań rekurencyjnych.

# 37. Dwie dodatkowe tablice, czyli stosujemy algorytm optymalny

**NA TEJ LEKCJI:**

- dowiesz się, czym jest metoda „dziel i rządź";
- poznasz zasadę działania algorytmu optymalnego.

metoda „dziel i rządź"

Jedną z głównych metod projektowania algorytmów w informatyce jest metoda „dziel i rządź", która prowadzi do bardzo efektywnych rozwiązań. W tej strategii problem dzieli się rekurencyjnie na dwa lub więcej mniejszych problemów do rozwiązania – te podproblemy są zazwyczaj tego samego lub podobnego typu. Praca nad mniejszymi problemami zazwyczaj trwa do momentu, gdy staną się one stosunkowo łatwe do rozwiązania. Wtedy scala się je i uzyskuje rozwiązanie całego problemu.

## 37.1. Optymalne wyszukiwanie najmniejszego i największego elementu

algorytm optymalny

Zasada działania tego algorytmu jest bardzo prosta. Badamy jednorazowo dwie liczby. Porównujemy je ze sobą i w zależności od wyników większą z nich przydzielamy do podzbioru maksimum, a mniejszą lub równą – do podzbioru minimum. Otrzymujemy w ten sposób dwa podzbiory. W pierwszym, tym z największymi liczbami, znajduje się nasze maksimum, a w drugim podzbiorze – liczba najmniejsza. Takie działanie będzie skuteczne i bardzo szybkie, jeżeli mamy parzystą liczbę elementów do porównania. Jeżeli natomiast wyszukujemy maksimum i minimum z nieparzystej liczby elementów, to w ostatnim porównaniu dublujemy element i dostaniemy parzystą ich liczbę.

Aby znaleźć oba elementy, zastosujemy najprostsze przeszukiwanie liniowe dla tych podzbiorów.

Algorytm jednoczesnego wyszukiwania największego i najmniejszego elementu:

**Wejście:**
$n$ – liczba elementów w tablicy $M$, $n \in N$
$M$ – tablica zawierająca elementy do przeszukania. Indeksy elementów rozpoczynają się od 0, a kończą na $n$–1. Tablica musi umożliwić dodanie elementu, jeżeli ich liczba jest nieparzysta.

**Wyjście:**
Element maksymalny i minimalny z tablicy $M$

**Elementy pomocnicze:**
$i$ – przebiega przez kolejne indeksy elementów tablicy $M$, $i \in N$
$maxZ$ – tymczasowy element maksymalny
$maxP$ – pozycja $maxZ$

**Lista kroków:**
Krok 1: Jeśli $n$ mod 2 = 1, $M[n] = M[n - 1]$
Krok 2: $minM$ = największa liczba całkowita
Krok 3: $maxM$ = najmniejsza liczba całkowita
Krok 4: $i = 0$
Krok 5: Dopóki $i < n$ wykonuj kroki od Krok 6 do Krok 12
Krok 6: Jeśli $M[i] > M[i + 1]$, to idź do Krok 10
Krok 7: Jeśli $M[i] < minM$, to $minM = M[i]$
Krok 8: Jeśli $M[i + 1] > maxM$, to $maxM = M[i + 1]$
Krok 9: Idź do Krok 12
Krok 10: Jeśli $M[i] > maxM$, to $maxM = M[i]$
Krok 11: Jeśli $M[i + 1] < minM$, to $minM = M[i + 1]$
Krok 12: $i = i + 2$
Krok 13: Wypisz $minM$ i $maxM$
Krok 14: Zakończ

## ZADANIA DO WYKONANIA

1. Napisz w języku C++ program, który będzie generował 14 liczb otrzymanych za pomocą generatora liczb losowych (pseudolosowych) z zakresu od 0 do 99. Program ma zapisywać wygenerowane liczby do tablicy Z, a następnie wyszukać najmniejszą i największą z nich. Jako wynik jest wypisywana zawartość całej tablicy w 15 kolejnych wierszach, a w wierszu 17 zostaną podane elementy minimalny i maksymalny.
2. Napisz w języku C++ program, który będzie generował 7 liczb pseudolosowych z zakresu od 0 do 9. Program wygenerowane liczby ma zapisać do tablicy $Z$, a następnie wyszukać najmniejszą i największą z nich. Jako wynik jest wypisywana zawartość całej tablicy w 7 kolejnych wierszach, a w wierszu 9 zostaną podane elementy minimalny i maksymalny.

## PODSUMOWANIE LEKCJI

s. 207 ▸ • **Metoda „dziel i rządź"** jest najchętniej wykorzystywaną metodą podczas projektowania algorytmów informatycznych. Prowadzi ona do bardzo efektownych rozwiązań.

s. 207 ▸ • **Algorytmem optymalnym** wyszukamy największy i najmniejszy element tablicy. Rozwiązanie problemu polega na podziale na dwa mniejsze podproblemy, które łatwiej rozwiązać.

# 38. Dziel i zwyciężaj, czyli sortowanie metodą scalania

**NA TEJ LEKCJI:**

- poznasz algorytmy sortowania poprzez scalanie oraz algorytm rekurencyjny.

W tym rozdziale zajmiemy się jednym z tzw. szybkich algorytmów.

W informatyce zostało przyjęte, że algorytmy proste mają dużą złożoność obliczeniową, natomiast algorytmy zaawansowane – małą. Wiele algorytmów sortujących, a takim jest również algorytm metodą scalania, korzysta z rekurencji.

algorytmy proste

## 38.1. Sortowanie metodą scalania

W 1945 r. John von Neumann (czytaj: dżon fon nojman) wynalazł algorytm sortowania poprzez scalanie.

sortowanie metodą scalania

To algorytm rekurencyjny, czyli taki, który do rozwiązania problemu wykorzystuje samego siebie ze zmienionym zestawem danych. Algorytm sortowania poprzez scalanie wykorzystuje zasadę „dziel i zwyciężaj" polegającą na dzieleniu zadania głównego na mniejsze podzadania, aż rozwiązanie stanie się oczywiste. Algorytm sortowania metodą scalania dzieli porządkowany zbiór na kolejne połowy tak długo, jak długo taki podział jest możliwy.

rekurencja

Następnie w ten sposób uzyskane części zbioru są sortowane rekurencyjnie tym samym algorytmem. Posortowane części na koniec są scalane.

sortowanie rekurencyjne

## 38.2. Scalanie zbiorów uporządkowanych

Główną operacją wykonywaną przez algorytm jest scalenie uporządkowanych zbiorów w jeden, również uporządkowany. Całą operację realizujemy poprzez wykorzystanie pomocniczego zbioru, w którym tymczasowo odkładamy scalone elementy dwóch pozostałych zbiorów.

Połączmy dla przykładu dwa uporządkowane już podzbiory:
(1, 5, 7)
(2, 3, 6)

| Scalone zbiory | | | Zbiór tymczasowy | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 7 | | | | | | |
| 2 | 3 | 6 | | | | | | |
| | 5 | 7 | 1 | | | | | |
| 2 | 3 | 6 | | | | | | |
| | 5 | 7 | 1 | | | | | |
| 2 | 3 | 6 | | | | | | |
| | 5 | 7 | 1 | 2 | | | | |
| | 3 | 6 | | | | | | |
| | 5 | 7 | 1 | 2 | | | | |
| | 3 | 6 | | | | | | |
| | 5 | 7 | 1 | 2 | 3 | | | |
| | | 6 | | | | | | |
| | 5 | 7 | 1 | 2 | 3 | | | |
| | | 6 | | | | | | |
| | | 7 | 1 | 2 | 3 | 5 | | |
| | | 6 | | | | | | |
| | | 7 | 1 | 2 | 3 | 5 | | |
| | | 6 | | | | | | |
| | | 7 | 1 | 2 | 3 | 5 | 6 | |
| | | | | | | | | |
| | | 7 | 1 | 2 | 3 | 5 | 6 | |
| | | | | | | | | |
| | | | 1 | 2 | 3 | 5 | 6 | 7 |
| | | | | | | | | |

Rys. 38.1. Scalanie zbiorów uporządkowanych

Zanim zajmiemy się napisaniem algorytmu scalania dwóch zbiorów, musimy zdefiniować dane. Załóżmy, że elementy zbioru będą przechowywane w jednej tablicy *z*[], a każdy element w tablicy ma swój indywidualny numer, czyli indeks z zakresu od 1 do *n*. Ponieważ w naszym algorytmie podzbiory są przyległymi do siebie częściami innego zbioru, musimy jeszcze zdefiniować indeksy wybranych elementów tych podzbiorów:

*ipm* – indeks pierwszego elementu w młodszym podzbiorze
*ips* – indeks pierwszego elementu w starszym podzbiorze
*ios* – indeks ostatniego elementu w starszym podzbiorze

Przez podzbiór młodszy rozumiemy zbiór zawierający elementy o indeksach mniejszych niż indeksy elementów w podzbiorze starszym.

Wyliczenie końcowego elementu młodszego podzbioru nie powinno sprawić większych problemów – będzie on o 1 mniejszy od indeksu pierwszego elementu podzbioru starszego.

Specyfikacja algorytmu scalania:
Scalaj(*ipm*, *ips*, *ios*)

**Wejście:**
*z*[] – zbiór scalany
*ipm* – indeks pierwszego elementu w młodszym podzbiorze, $ipm \in N$
*ips* – indeks pierwszego elementu w starszym podzbiorze, $ips \in N$
*ios* – indeks ostatniego elementu w starszym podzbiorze, $ios \in N$

**Wyjście:**
*z*[] – scalany zbiór

**Elementy pomocnicze:**
*p*[] – zbiór pomocniczy, który zawiera tyle samo elementów, co zbiór *z*[]
*i*1 – indeks elementów w młodszej połówce zbioru *z*[], $i1 \in N$
*i*2 – indeks elementów w starszej połówce zbioru *z*[], $i2 \in N$
*i* – indeks elementów w zbiorze pomocniczym *p*[], $i \in N$

**Lista kroków:**
Krok 1: $i1 = ipm$; $i2 = ips$; $i = ipm$
Krok 2: Dla $i = ipm, ipm + 1, \ldots ios$: wykonaj
    jeśli $(i1 = ips) - (i2 \leq ios$ i $z[i1] > z[i2]$, to
    $p[i] = z[i2]$; $i2 = i2 + 1$
    inaczej
    $p[i] = z[i1]$; $i1 = i1 + 1$
Krok 3: Dla $i = ipm, ipm + 1, \ldots, ios$: $z[i] = p[i]$
Krok 4: Zakończ

Specyfikacja algorytmu sortującego:
Sortuj_scalajac(*ipm*, *ios*)

**Wejście:**
*z*[] – sortowany zbiór
*ipm* – indeks pierwszego elementu w młodszym podzbiorze, $ipm \in N$
*ios* – indeks ostatniego elementu w starszym podzbiorze, $ios \in N$

**Wyjście:**
*z*[] – scalany zbiór

**Elementy pomocnicze:**
*ips* – indeks pierwszego elementu w starszym podzbiorze, $ips \in N$

**Lista kroków:**
Krok 1: *ips* = (*ipm* + *ios* + 1) div 2
Krok 2: Jeśli *ips* – *ipm* > 1, to Sortuj_scalajac(*ipm*, *ips* – 1)
Krok 3: Jeśli *ios* – *ips* > 0, to Sortuj_scalajac(*ips*, *ios*)
Krok 4: Scalaj(*ipm*, *ips*, *ios*)
Krok 5: Zakończ

## ZADANIA DO WYKONANIA

1. Narysuj algorytm scalania w formie schematu blokowego.
2. Narysuj algorytm sortowania w formie schematu blokowego.
3. Napisz program sortowania przez scalanie w języku C++ dla zbioru parzystego, który ma 10 elementów.
4. Napisz program sortowania przez scalanie w języku C++ dla zbioru parzystego, którego liczebność będzie podana przez użytkownika.
5. Narysuj algorytm scalania w formie schematu blokowego dla zbioru nieparzystego.
6. Narysuj algorytm sortowania w formie schematu blokowego dla zbioru nieparzystego.

## PODSUMOWANIE LEKCJI

s. 209 • **Algorytmy proste** mają dużą złożoność obliczeniową, natomiast algorytmy zaawansowane – małą.

s. 209 • Wiele algorytmów sortujących, a takim jest również algorytm metodą scalania, korzysta z **rekurencji**.

s. 209 • W 1945 r. John von Neumann wynalazl **algorytm sortowania poprzez scalanie**. To algorytm rekurencyjny, czyli taki, który do rozwiązania problemu wykorzystuje samego siebie ze zmienionym zestawem danych.

s. 209 • Algorytm sortowania metodą scalania dzieli porządkowany zbiór na kolejne połowy tak długo, dopóki taki podział jest możliwy. Następnie w ten sposób uzyskane części zbioru są **sortowane rekurencyjnie** tym samym algorytmem. Posortowane części na koniec są scalane.

# 39. Raz krócej, raz dłużej, czyli sortowanie kubełkowe

**NA TEJ LEKCJI:**

- poznasz algorytm sortowania kubełkowego oraz dowiesz się, czym jest algorytm sortowania metodą scalania.

sortowanie kubełkowe

Sortowanie kubełkowe to kolejny z algorytmów sortowania. Najczęściej jest stosowany, gdy liczby w zadanym przedziale są rozłożone jednostajnie.

cecha algorytmu sortowania

Cechą charakterystyczną tego algorytmu jest to, że jest najefektywniejszy do sortowania dużych zbiorów liczb całkowitych o małym zakresie wartości.

## 39.1. Sortowanie metodą kubełkową

opracowanie algorytmu sortowania kubełkowego

Algorytm sortowania kubełkowego został opracowany w 1956 r. przez E.J. Isaaca i R.C. Singletona. Algorytm wrzuca elementy sortowanej tablicy do *n* kubełków. Każdy z kubełków otrzymuje liczby z określonego przedziału, a następnie zostaje posortowany.

Algorytm na sam koniec tworzy tablicę, pobierając kolejne dane z kubełków. Najlepszą złożoność ma on dla danych, które są równo rozłożone. Złożoność czasowa wynosi wtedy tyle samo, co w przypadku algorytmu szybkiego. W przypadku, gdy w każdym kubełku będzie się znajdował tylko jeden element, algorytm zachowuje się jak przy sortowaniu przez zliczenie. Przyjęło się, że dane sortowane przez algorytm powinny zawierać się w przedziale [0, 1]. Jeśli elementy zbioru są spoza tego zakresu, wystarczy rozpatrywaną liczbę podzielić przez maksymalny element z listy i na tej podstawie przyporządkować go do kubełka.

**Przykład:**
Mamy do posortowania tablicę *T*, która ma następujące elementy: *T*[13, 27, 29, 33, 2, 21, 10, 18]. Na samym początku ustalamy zakres liczb, a następnie decydujemy, na ile kubełków elementy zostaną podzielone. W naszym przypadku zakres liczb to: [2, 33], a dane zostaną podzielone na $n = 4$ kubełki. Kolejnym krokiem będzie przyporządkowanie danych do kubełków:

| Kubełek | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Zakres | [2, 10) | [10, 18) | [18, 27) | [27, 33] |
| Elementy | {2} | {10, 13} | {18, 21} | {27, 29, 33} |

Na tym etapie każdy kubełek sortujemy dowolnym algorytmem:

| Kubełek | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Elementy | {2, 10} | {13, 18} | {21, 27} | {29, 33} |

Ostatnim krokiem będzie stworzenie nowej tablicy, która będzie zawierać wszystkie elementy z naszych kubełków, ale już posortowane. Posortowane elementy naszej tablicy *T* to: *Tp*[2, 10, 13, 18, 21, 27, 29, 33].

Jeżeli zakres liczb wynosi [0, 1], to nie będziemy mieli problemu z ustaleniem zakresu w kubełkach. Jednak naszym zakresem jest [2, 33], więc dzielimy każdą liczbę przez maksymalną liczbę z zakresu. W naszym przypadku będzie to 33. Otrzymujemy następujące wartości:

| Elementy | 13 | 27 | 29 | 33 | 2 | 21 | 10 | 18 |
|---|---|---|---|---|---|---|---|---|
| Elementy | ~0.40 | ~0.82 | ~0.88 | 1 | ~0.06 | ~0.64 | ~0.30 | ~0.55 |

Przyporządkowanie do kubełków wygląda następująco:

| Kubełek | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Zakres | [0, 0.25) | [0.25, 0.5) | [0.5, 0.75) | [0.75, 1] |
| Elementy | {2} | {10, 13} | {18, 21} | {27, 29, 33} |

## ZADANIA DO WYKONANIA

1. Napisz rekurencyjną wersję algorytmu w języku C++.
2. Napisz w języku C++ program sortujący listę, na której znajdują się elementy zarówno dodatnie, jak i ujemne.
3. Napisz w języku Java program sortujący listę, na której znajdują się elementy zarówno dodatnie, jak i ujemne.

## PODSUMOWANIE LEKCJI

s. 213 ▸ • **Sortowanie kubełkowe** jest najczęściej wykorzystywanym algorytmem przez informatyków, jeśli posortowane mają być liczby rozłożone jednostajnie w danym przedziale.

s. 213 ▸ • **Cechą charakterystyczną algorytmu** jest to, że pozwala sortować duże zbiory liczb całkowitych o małym zakresie. Duży zakres również uda się nim posortować.

s. 213 ▸ • Algorytm sortowania kubełkowego został stworzony w **1956 r.** przez E.J. Isaaca i R.C. Singletona.

## PODSUMOWANIE PRZED SPRAWDZIANEM

### 13. Wspólna miara, czyli podstawy złożoności obliczeniowej

- **Czasowa złożoność obliczeniowa** to ilość czasu potrzebnego do rozwiązania problemu w zależności od tego, ile mamy danych wejściowych. ▸ s. 94
- Czasową złożoność obliczeniową wyrażamy albo w jednostkach czasu, albo w liczbie **operacji dominujących**, które należy wykonać dla *n* danych, aby otrzymać rozwiązanie. ▸ s. 94
- **Złożoność pamięciowa** określa liczbę komórek pamięci, która będzie zajęta przez dane i wyniki pośrednie tworzone w trakcie pracy algorytmu. ▸ s. 94
- Klasy złożoności obliczeniowej określają rząd funkcji $T(n)$. Jednym ze sposobów określenia rzędu tej funkcji jest popularna **notacja omikron** – zwana także notacją dużego O. ▸ s. 96

### 14. Przez 1 i siebie, czyli jak zbadać, czy dana liczba jest liczbą pierwszą

- **Metoda sita Eratostenesa** powstała w starożytności i wyszukuje liczby pierwsze spośród liczb z zakresu od 2 do *n*. ▸ s. 98
- By zbadać, czy dana **liczba całkowita jest pierwsza**, można użyć metody dzielenia jej przez liczby całkowite i sprawdzania reszty. ▸ s. 99
- **Wielkie liczby pierwsze** odgrywają dużą rolę w kryptologii i szyfrowaniu danych komputerowych. Są pomocne przy tworzeniu szyfrów z kluczem publicznym. ▸ s. 99

### 15. Która pierwsza, czyli jak program bada liczby

- **Modyfikacje** programów mogą zwiększyć szybkość ich działania. ▸ s. 106
- Program dający odpowiedź na pytanie, **czy liczba jest pierwsza,** zawiera **pętle**. ▸ s. 107
- **Pętle for** i **while** mogą być stosowane w różnych realizacjach tego samego algorytmu. ▸ s. 107

### 16. Więcej pierwszych, czyli generujemy liczby

- Liczba naturalna jest **liczbą pierwszą** wtedy, gdy ma dokładnie dwa różne dzielniki: 1 i siebie samą. ▸ s. 111
- W informatyce liczby pierwsze mają ogromne zastosowanie, na nich opierają się zaawansowane systemy szyfrowania, używane w bankowości czy przez służby państwowe i wojsko. ▸ s. 111
- Wybraną liczbę naturalną *p* próbujemy dzielić przez liczby od 2 do $p - 1$. Jeżeli żadna z tych liczb nie jest dzielnikiem naszej liczby, tylko ona sama, to oznacza, że jest to liczba pierwsza. ▸ s. 111

### 17. Największy i najmniejszy, czyli jak znaleźć NWD i NWW

- Autorstwo metody obliczania **NWD** jest przypisywane **Euklidesowi**, a **algorytm** służący do obliczenia NWD został nazwany jego imieniem. ▸ s. 113
- Istnieje algorytm **nieoptymalny** i **optymalny**. Różnica w ilości powtórzeń pętli jest tym większa, im mniejszy względem najmniejszej z badanych liczb jest NWD. ▸ s. 113
- W programie szukającym **NWD** i **NWW** można użyć pętli **while**. ▸ s. 116

### 18. Euklides i rekurencja, czyli szukamy NWD

- Największym wspólnym dzielnikiem (**NWD**) dla dwóch (lub więcej) liczb naturalnych jest największa liczba naturalna dzieląca każdą z nich. ▸ s. 119
- Umiejętność wskazywania NWD najczęściej wykorzystuje się podczas redukcji ułamków do postaci nieskracalnej, czyli takiej, w której licznik i mianownik są względnie pierwsze. ▸ s. 119
- **Rekurencja**, nazywana również rekursją, to odwoływanie się np. funkcji lub definicji do samej siebie. ▸ s. 120
- W matematyce przykładem rekurencji będzie **ciąg Fibonacciego**, w którym pierwszy wyraz jest równy 0, drugi jest równy 1, a każdy następny to suma dwóch poprzednich. ▸ s. 120
- W programowaniu rekurencja to wywołanie przez funkcję samej siebie. ▸ s. 120
- NWD znajdujemy za pomocą **algorytmu Euklidesa**, który nie dotyczył liczb naturalnych, a dwóch odcinków. ▸ s. 122

### 19. Metoda iteracji, czyli algorytm Euklidesa w innej postaci

- **Iteracja** to metoda w analizie matematycznej i programowaniu polegająca na wielokrotnym stosowaniu tego samego przekształcenia lub procedury, jak również kolejne przekształcanie lub procedura. ▸ s. 124

s. 124 • W programowaniu korzystając z iteracji, stosujemy **pętle**.
s. 124 • W języku C++ mamy do dyspozycji trzy rodzaje pętli: **for, while** i **do...while**.
s. 124 • W języku Java dochodzi nam dodatkowa pętla **for each**, dzięki której możemy robić operacje na tablicach.

### 20. Dodawanie ułamków, czyli jak wykorzystać NWW i NWD w programie komputerowym

s. 131 • **Algorytm dodawania ułamków** wykorzystuje algorytmy znajdowania NWW i NWD.
s. 133 • **Kod znajdowania NWW i NWD** można zapisać **w postaci funkcji**. Dzięki temu możliwe będzie wielokrotne ich wykorzystanie bez konieczności umieszczania kodu obliczeń w każdym wymagającym obliczeń NWW i NWD miejscu programu.

### 21. Skracamy i wyłączamy, czyli dodawania ułamków ciąg dalszy

s. 136 • Funkcja obliczająca NWD w programie skracającym postać ułamka będzie wykorzystana wielokrotnie.
s. 138 • Wartości mianowników są **przekazywane do funkcji** obliczających NWW i NWD.
s. 139 • W języku C++ wynik dzielenia zapisany pod zmienną zadeklarowaną jako **int** zawiera jedynie **część całkowitą tego działania.**

### 22. Szyfr Cezara, czyli poznajemy szyfr przesuwający

s. 141 • **Szyfr Cezara** jest szyfrem przesuwającym, wykorzystującym alfabet łaciński.
s. 144 • **https://** przed adresem internetowym oznacza szyfrowanie transmisji danych.

### 23. Konspiracja, czyli tworzymy własne szyfry

s. 146 • Z **kryptografią symetryczną** mamy do czynienia wtedy, gdy wykorzystuje się **klucz** znany nadawcy i odbiorcy.
s. 146 • **Metoda asymetryczna** polega na stosowaniu **kluczy publicznych** i **prywatnych**.
s. 147 • Do przesyłania tablicy kluczy można użyć **szyfrowania asymetrycznnego**.
s. 150 • Użycie **tablicy kluczy** do szyfru przesuwającego utrudnia deszyfrację, ponieważ dla każdej pozycji szyfrowanego tekstu może występować inny klucz.
s. 152 • Tablice kluczy mogą być tworzone **generatorem liczb losowych.**

### 24. Z kluczem publicznym, czyli bezpieczniejsze szyfrowanie

s. 154 • **RSA** jest jednym z najpopularniejszych algorytmów szyfrowania z kluczem publicznym.
s. 155 • Algorytm RSA działa w **trzech etapach**: tworzenia kluczy, szyfrowania kluczem publicznym i deszyfracji kluczem prywatnym.

### 25. Szyfrowanie zachłanne, czyli kody Huffmana

s. 158 • **Metoda zachłanna** w celu wyznaczenia najlepszego rozwiązania dokonuje w danym momencie wyboru najlepszego rozwiązania częściowego.
s. 158 • Typowe zadanie rozwiązywane metodą zachłanną ma charakter **optymalizacji**.
s. 158 • **Kody Huffmana** to jedna z powszechnie dzisiaj stosowanych metod kompresji danych.
s. 159 • Znaki **ASCII** można kodować kodami Huffmana na dwa sposoby: za pomocą kodów o stałej długości oraz za pomocą kodów o zmiennej długości.

### 26. Tworzenie szyfrów przestawieniowych, czyli kodowanie znaków o zmiennej długości

s. 161 • **Szyfr przestawieniowy** to jedna z klas metod szyfrowania. Charakteryzuje się tym, że w zaszyfrowanym tekście występują wszystkie znaki z tekstu jawnego, ale w innej kolejności.
s. 161 • **Kodowanie znaków o zmiennej długości** jest drugim sposobem kodowania znaków ASCII za pomocą kodów Huffmana.
s. 161 • **Kody o długości zmiennej** wskazują lepsze właściwości dla znaków częściej występujących, ponieważ lepiej jest przypisać mniej bitów znakom częściej występującym i więcej bitów znakom rzadziej występującym.
s. 162 • Tekst jawny można zaszyfrować również poprzez użycie **szyfru przestawieniowego z tablicą** – dla danego łańcucha $\alpha$ wyznaczamy liczbę neutralną $n$, aby $n2 \leq |\alpha|$.
s. 163 • **Kody bezprzystankowe** reprezentuje się drzewami binarnymi, które mają ułatwić odczyt.

## 27. Automat wydaje, czyli jak dynamicznie wydawać resztę

- Wydawanie reszty w opisany sposób, czyli zaczynając od największej możliwej liczby najwyższych nominałów, nazywamy **metodą zachłanną**. ▸ s. 166
- Do wydawania reszty wszystkimi nominałami można zastosować **tablicę.** ▸ s. 171

## 28. Wydajemy resztę, czyli jak wykorzystać metodę zachłanną

- Rozwiązanie **problemu wydawania reszty** polega na ustaleniu, w jakich nominałach wydać resztę, aby użyć jak najmniejszej liczby monet. ▸ s. 176
- Problem można rozwiązać w wariancie ograniczonym polegającym na tym, że dysponujemy jedynie określoną liczbą monet z każdego nominału, oraz w wariancie nieograniczonym, który daje możliwość skorzystania z nieograniczonej liczby monet każdego nominału. ▸ s. 176
- **Algorytm zachłanny** pozwala na wykorzystanie w rozwiązaniu problemu najwyższych dostępnych nominałów. ▸ s. 176

## 29. Co z tym groszem, czyli błąd zaokrąglenia i błąd przybliżenia

- **Zaokrąglanie liczb** polega na ustaleniu dokładności zaokrąglenia, tj. na wskazaniu cyfry, względem której jest określane zaokrąglenie, zastąpienie zerami wszystkich cyfr na prawo od wskazanej cyfry oraz zwiększenie wskazanej cyfry o jeden. ▸ s. 179
- **Zaokrąglanie liczb do rzędu dziesiątek** polega na znalezieniu jak najbliższej tej liczbie wielokrotności liczby 10. ▸ s. 180
- **Zaokrąglenie liczby do rzędu części dziesiątych** polega na znalezieniu najbliższej tej liczbie wielokrotności liczby 0,1. ▸ s. 180
- **Błąd zaokrąglenia** to różnica pomiędzy dokładną wartością oraz liczbą użytą w obliczeniach numerycznych. ▸ s. 181
- **Błąd bezwzględny** oblicza się jako maksymalną niezgodność pomiędzy wartością przybliżoną i pozbawioną błędów. ▸ s. 182
- **Błąd względny** pokazuje, jaką częścią danej liczby jest wartość, o jaką obniżyliśmy lub powiększyliśmy liczbę. ▸ s. 183

## 30. Sprawdzam, czyli porównywanie tekstów i wyszukiwanie wzorca w tekście metodą naiwną

- Wyszukiwanie **wzorca** jest jednym z głównych problemów tekstowych badanych przez naukowców, wykorzystywanych m.in. w edytorach tekstów do wyszukiwania informacji w tekście. ▸ s. 185
- Wzorzec jest spójnym ciągiem znaków, a raczej podciągiem (podtekstem), który występuje w danym ciągu znaków. ▸ s. 185
- Dla każdej $i$-tej litery musimy sprawdzić $m$ kolejnych znaków, czy są one zgodne ze znakami we wzorcu. Porównanie jest wykonywane tak długo, dopóki istnieje kolejne $m$ liter począwszy od $i$-tego indeksu. ▸ s. 186

## 31. Od ostatniego, czyli szukamy wzorca tekstu metodą Boyera–Moore'a

- **Algorytm Boyera–Moore'a** (**algorytm BM**) jest uznawany za najszybszy, a zarazem najbardziej praktyczny algorytm wyszukiwania wzorca. ▸ s. 188
- Algorytm BM jest wykorzystywany obecnie w prawie każdym edytorze tekstu przy wyszukiwaniu tekstów. ▸ s. 188
- Działanie algorytmu BM rozpoczyna się porównywaniem łańcucha od ostatniego znaku wzorca, dzięki czemu o wiele szybciej znajduje on rozwiązanie. ▸ s. 188

## 32. Z lewej i z prawej, czyli szukamy palindromów

- **Palindrom** to wyrażenie brzmiące tak samo czytane od lewej do prawej i od prawej do lewej. Przykładem palindromu jest słowo „kajak". ▸ s. 192
- Do wyszukiwania palindromów wykorzystuje się **algorytm naiwny.** ▸ s. 192
- **Łańcuch znaków** jest palindromem, jeśli da się rozłożyć na dwa podłańcuchy $y$ i $yP$ według poniższego schematu: ▸ s. 193
  - $z = yyP$ – palindrom parzysty
  - $z = yXyp$ – palindrom nieparzysty, gdzie $X$ jest dowolną literą alfabetu.

## 33. Przestawianie liter, czyli jak stworzyć anagram

s. 195 ▸ • Aby sprawdzić, czy wyraz jest anagramem, musimy użyć **sortowania bąbelkowego**. Jego działanie opiera się na cyklicznym porównaniu par sąsiadujących ze sobą elementów i zmianie ich kolejności, gdy nie jest spełnione kryterium porządkowego zbioru.

s. 196 ▸ • **Anagram** to wyraz, wyrażenie lub całe zdanie, które powstaje przez przestawienie liter lub sylab innego wyrazu, wyrażenia lub zdania – warunkiem jest wykorzystanie wszystkich liter lub sylab z wyrazu (wyrażenia, zdania) wyjściowego.

## 34. Od A do Z, czyli alfabetyczne sortowanie wyrazów

s. 198 ▸ • **Sortowanie alfabetyczne** to porządek alfabetyczny, na który składa się zbiór reguł używanych do porządkowania terminów i haseł.

s. 198 ▸ • W informatyce najczęściej sortowanie jest wykorzystywane w arkuszach kalkulacyjnych lub edytorach tekstu, w których używa się **sortowania jednopoziomowego**.

s. 198 ▸ • W programowaniu w celu **sortowania alfabetycznego** wykorzystuje się sortowanie przez wybór, zwane też sortowaniem przez wymianę.

## 35. Odwrotna notacja polska, czyli zapisujemy wyrażenia arytmetyczne

s. 202 ▸ • Odwrotna notacja polska jest sposobem zapisu wyrażeń arytmetycznych, w których znak wykonywanej operacji jest umieszczony po operandach – to tzw. **zapis postfiksowy** – a nie między nimi, jak to jest robione standardowo.

s. 204 ▸ • **ONP** ułatwia wykonywanie obliczeń przez komputer, ponieważ nie stosuje się w niej nawiasów i zachowuje się kolejność wykonywanych działań.

s. 204 ▸ • Do obliczeń wyrażenia ONP jest wykorzystywany **stos**.

## 36. Naiwny jak algorytm, czyli szukamy najmniejszego i największego elementu metodą naiwną

s. 205 ▸ • **Sortowanie naiwne** wyznacza największy i najmniejszy element zbioru, przenosząc badany element na początek lub na koniec zbioru.

s. 205 ▸ • Sortowanie naiwne występuje w **wersjach rekurencyjnej i iteracyjnej**. Wersja iteracyjna jest lepsza, ponieważ nie zajmuje dodatkowo pamięci na kolejne poziomy wywołań rekurencyjnych.

## 37. Dwie dodatkowe tablice, czyli stosujemy algorytm optymalny

s. 207 ▸ • **Metoda „dziel i rządź"** jest najchętniej wykorzystywaną metodą podczas projektowania algorytmów informatycznych. Prowadzi ona do bardzo efektownych rozwiązań.

s. 207 ▸ • **Algorytmem optymalnym** wyszukamy największy i najmniejszy element tablicy. Rozwiązanie problemu polega na podziale na dwa mniejsze podproblemy, które łatwiej rozwiązać.

## 38. Dziel i zwyciężaj, czyli sortowanie metodą scalania

s. 209 ▸ • **Algorytmy proste** mają dużą złożoność obliczeniową, natomiast algorytmy zaawansowane – małą.

s. 209 ▸ • Wiele algorytmów sortujących, a takim jest również algorytm metodą scalania, korzysta z **rekurencji**.

s. 209 ▸ • Algorytm **sortowania poprzez scalanie** to algorytm rekurencyjny, czyli taki, który do rozwiązania problemu wykorzystuje samego siebie ze zmienionym zestawem danych.

s. 209 ▸ • Algorytm sortowania metodą scalania dzieli porządkowany zbiór na kolejne połowy, tak długo, dopóki to możliwe. Następnie w ten sposób uzyskane części zbioru są **sortowane rekurencyjnie** tym samym algorytmem. Posortowane części na koniec są scalane.

## 39. Raz krócej raz dłużej, czyli sortowanie kubełkowe

s. 213 ▸ • **Sortowanie kubełkowe** jest najczęściej wykorzystywanym algorytmem przez informatyków, gdy posortowane mają być liczby rozłożone jednostajnie w danym przedziale.

s. 213 ▸ • **Cechą charakterystyczną** tego algorytmu jest to, że pozwala sortować duże zbiory liczb całkowitych o małym zakresie. Te z dużego zakresu również można nim posortować.



# IV. Grafika i druk



- Druk 3D
- Projektowanie 3D
- Wypukłe napisy w edytorze GIMP
- Działanie skanera
- Ekspozycja w fotografii
- Przygotowanie do sesji zdjęciowej
- Kadrowanie
- Edytowanie zdjęcia w warstwach
- Filmowanie
- Strumieniowanie danych

# 40. Modele w przestrzeni, czyli podstawy druku 3D

**NA TEJ LEKCJI:**

- poznasz zasadę działania drukarki 3D;
- dowiesz się, po co potrzebne są modele 3D.

Drukarki 3D przestały być wyłącznie wyposażeniem laboratoriów i biur konstrukcyjnych. Znalazły zastosowanie w wielu dziedzinach życia, a coraz częściej można je spotkać także w domach. Są w stanie wydrukować praktycznie każdy przedmiot od małej figurki do domu mieszkalnego lub mostu. Planuje się wykorzystać tę technologię w budowie baz na Księżycu i planetach. O wiele łatwiej dowieźć tam materiały budowlane niż całe, gotowe moduły. To jednak odległa przyszłość. Na szczęście nawet amator drukowania 3D bez problemu znajdzie odpowiedni program i nauczy się nim posługiwać. Wielu z nich można używać za darmo. Jakie powinny spełniać wymagania i skąd brać inspiracje do projektów? Jak działają drukarki 3D? Tego dowiesz się w rozdziale.

## 40.1. Warstwa na warstwie, czyli jak działa drukarka 3D

Wyobraź sobie, że zbudowałeś/zbudowałaś z klocków jakiś obiekt, na przykład pałac (rys. 40.1.). A teraz pomyśl, jak rozwiązać problem zbudowania takiego samego obiektu przez twoją koleżankę lub kolegę w tym samym czasie, lecz w oddalonym miejscu – przykładowo w ich mieszkaniu. Czy umiesz ułożyć algorytm opisujący kolejność czynności i gwarantujący zbudowanie identycznych pałaców?



Rys. 40.1. Pałac z klocków z kolorowymi warstwami elementów

Przyjmijmy, że budowniczy będą dysponować takim samym zestawem klocków, a algorytm zapiszemy w postaci listy kroków.

Założenia algorytmu:

1. Kolejne czynności opisz jako układanie warstw klocków.
2. Warstwy klocków będą układane od lewej strony do prawej.
3. Przesunięcie rozpoczęcia kolejnej warstwy względem niższej liczone jest liczbą kołków konstrukcyjnych.

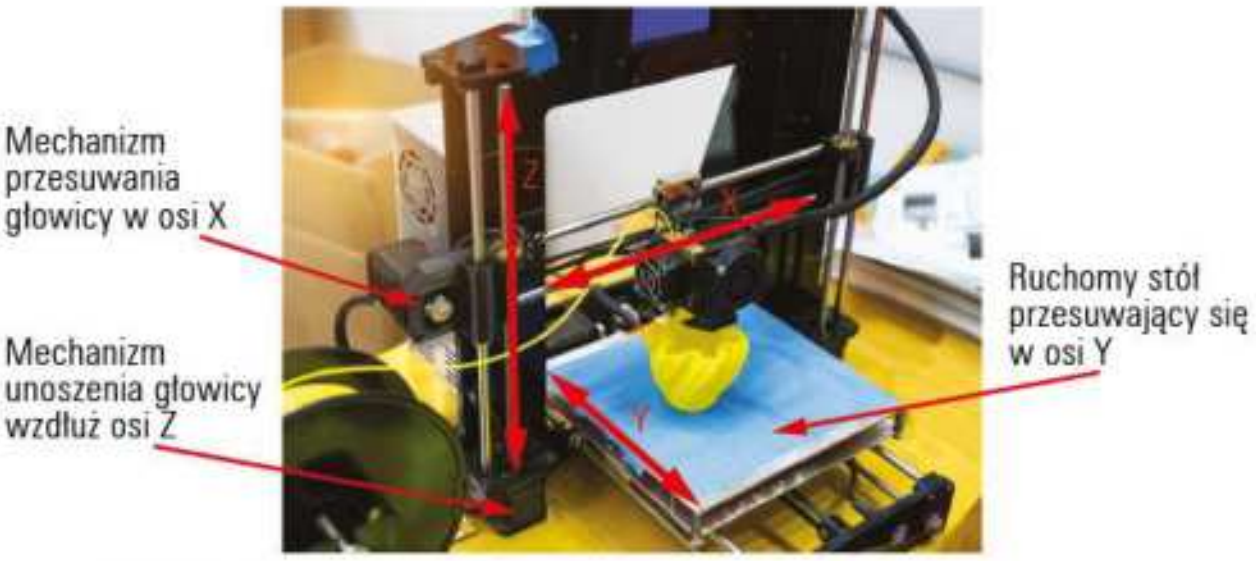
Ułóż samodzielnie taki algorytm, uwzględniając założenia, i porównaj z algorytmem na stronie obok.

Algorytm budowania pałacu z rysunku 40.1.

1. Z sześciu czerwonych klocków utwórz pierwszą warstwę (od lewej do prawej), układając je jeden obok drugiego.
2. Z żółtych klocków ułóż drugą warstwę, zaczynając z przesunięciem o jeden rządek kołków konstrukcyjnych w prawo i kończąc o jeden przed końcem pierwszej warstwy.
3. Z zielonych klocków ułóż trzecią warstwę, zaczynając z przesunięciem o jeden rządek kołków konstrukcyjnych w prawo i kończąc o jeden przed końcem drugiej warstwy.
4. Z czerwonych klocków ułóż czwartą warstwę, zaczynając z przesunięciem o dwa rządki kołków konstrukcyjnych w prawo i kończąc o dwa przed końcem trzeciej warstwy.
5. Na środku czwartej warstwy ułóż zielony klocek tworzący piątą warstwę.
6. Ułóż klocek żółty na zielonym (szósta warstwa).
7. Siódmą warstwę utwórz z klocka niebieskiego zakończonego kopułką.

Wspólną cechą wszystkich punktów listy kroków jest układanie warstw. Tę własność wykorzystano także podczas tworzenia technologii druku 3D. Tam również układa się warstwę na warstwie według określonego algorytmu. Ponieważ druk ma być przestrzenny, warstwy nakłada się wzdłuż osi XYZ (rys. 40.2.).

Program komputerowy zamienia projekt (model) tworzony w edytorze grafiki 3D na zbiór warstw, które muszą być nałożone na siebie, by powstał obiekt. Podobnie wygląda wydruk na kartce, w którym obraz składa się z kolejnych linii na płaszczyźnie.




Rys. 40.2. Mechanizmy drukarki 3D

Dzięki złożeniu ruchów głowicy i podłoża można umieścić głowicę w dowolnym punkcie przestrzeni drukowania z uwzględnieniem najmniejszego możliwego skoku mechanizmów.

**Uwaga!**

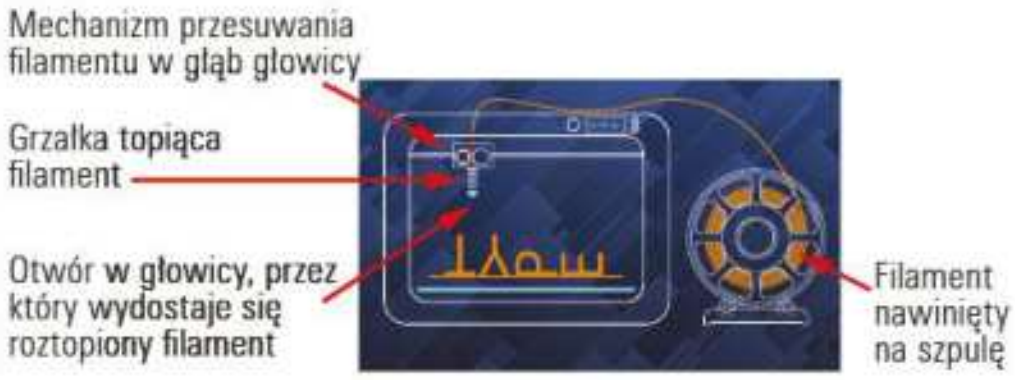
Istnieją konstrukcje drukarek 3D, w których porusza się wyłącznie głowica, a podłoże pozostaje nieruchome.

Wiesz już, jak głowica osiąga pożądane położenie w przestrzeni drukowania. Pozostaje jeszcze rozwikłać tajemnicę jej działania.

**filament w drukarkach**

Głowica drukarki 3D rozgrzewa filament (rys. 40.3.). Topi go i nakłada warstwę na warstwie. Filament szybko twardnieje i tworzy drukowany obiekt. Wadą takiego rozwiązania jest długi czas druku, a zaletą – możliwość stosowania filamentów z różnych materiałów, od tworzyw sztucznych aż po metale (rys. 40.4.).

**głowica w drukarkach 3D**




Rys. 40.3. Filament i głowica drukarki 3D



Rys. 40.4. Różnokolorowe filamenty do druku 3D

## 40.2. Edytory i skanowanie, czyli skąd się biorą modele 3D

Drukarka, choćby najlepsza, niczego nie wydrukuje, jeśli nie będzie miała cyfrowego modelu wytworzonego za pomocą komputera.

**model 3D**

Ten może powstać w edytorze grafiki 3D lub z zastosowaniem skanera 3D. Profesjonalne występują w wersji stołowej (rys. 40.5.) i ręcznej, w której obiekt pozostaje nieruchomy. Za pomocą ręcznych skanerów skanuje się obiekty o dużych gabarytach oraz takie, które nie mogą być przenoszone, jak eksponaty muzealne.

Najtańszym rozwiązaniem dla amatorów i pasjonatów modelowania 3D są aplikacje na smartfony. Jakość powstałych modeli nie jest wysoka, ale po ich edycji i korekcie można uzyskać zadowalające efekty. Informacje o tych aplikacjach bez trudu znajdziesz w sieci.




Rys. 40.5. Skaner i edytor grafiki 3D

Skanery 3D działają nie tylko w paśmie promieniowania widzialnego. Istnieją takie, które potrafią wiernie skopiować kształt organów ludzkiego ciała. Modele są następnie drukowane i wykorzystywane przykładowo do ćwiczeń przed skomplikowaną operacją.

Skany 3D można także bez trudu zamówić w wyspecjalizowanych firmach.

A co, jeśli nie chcemy drukować modeli rzeczywistych obiektów? Nie wszystkie edytory grafiki 3D nadają się do tworzenia modeli od podstaw. Jakie powinny spełniać warunki? Domyślasz się, że oprócz edycji w trzech wymiarach powinny pozwolić zapisywać pliki wynikowe zgodne z obsługiwanymi przez drukarki. Jest ich wiele. Im więcej obsługuje program, tym lepiej, jednak najlepiej sprawdzić, z jakimi współpracuje posiadana drukarka. Listę znajdziesz w instrukcji.

fomat plików dla drukarek 3D

**STL** – w tym formacie geometria obiektu jest zbudowana z trójkątów. Mimo wad to najpowszechniej stosowany format pliku dla drukarek 3D.

**OBJ** – format przeznaczony dla programów do animacji 3D. Geometria obiektu jest przedstawiana za pomocą wierzchołków wielokątów, przy czym każdy z punktów w zapisie występuje tylko raz. OBJ stał się uniwersalnym sposobem zapisu plików z obiektami 3D. Powszechnie wykorzystuje się go do druku przestrzennego.

**VRML** – zaawansowany format uwzględniający kolory, fakturę, przezroczystość i inne parametry powierzchni. Ze względu na swoje zalety i darmową licencję również cieszy się popularnością wśród użytkowników drukarek 3D.

**3MF** – format opracowany przez Microsoft wspierany przez wiele wiodących firm zajmujących się techniką projektowania i drukowania. Charakteryzuje się niewielkim rozmiarem.

Który program do edycji modeli 3D z nich wybrać? Wszystko zależy od potrzeb. Początkujący użytkownicy zapewne będą szukać darmowych i łatwych w obsłudze programów, z niewielkimi wymaganiami sprzętowymi. Przykłady zostały zebrane w poniższej tabeli.

Tab. 40.1. Przykłady programów do edycji obiektów 3D

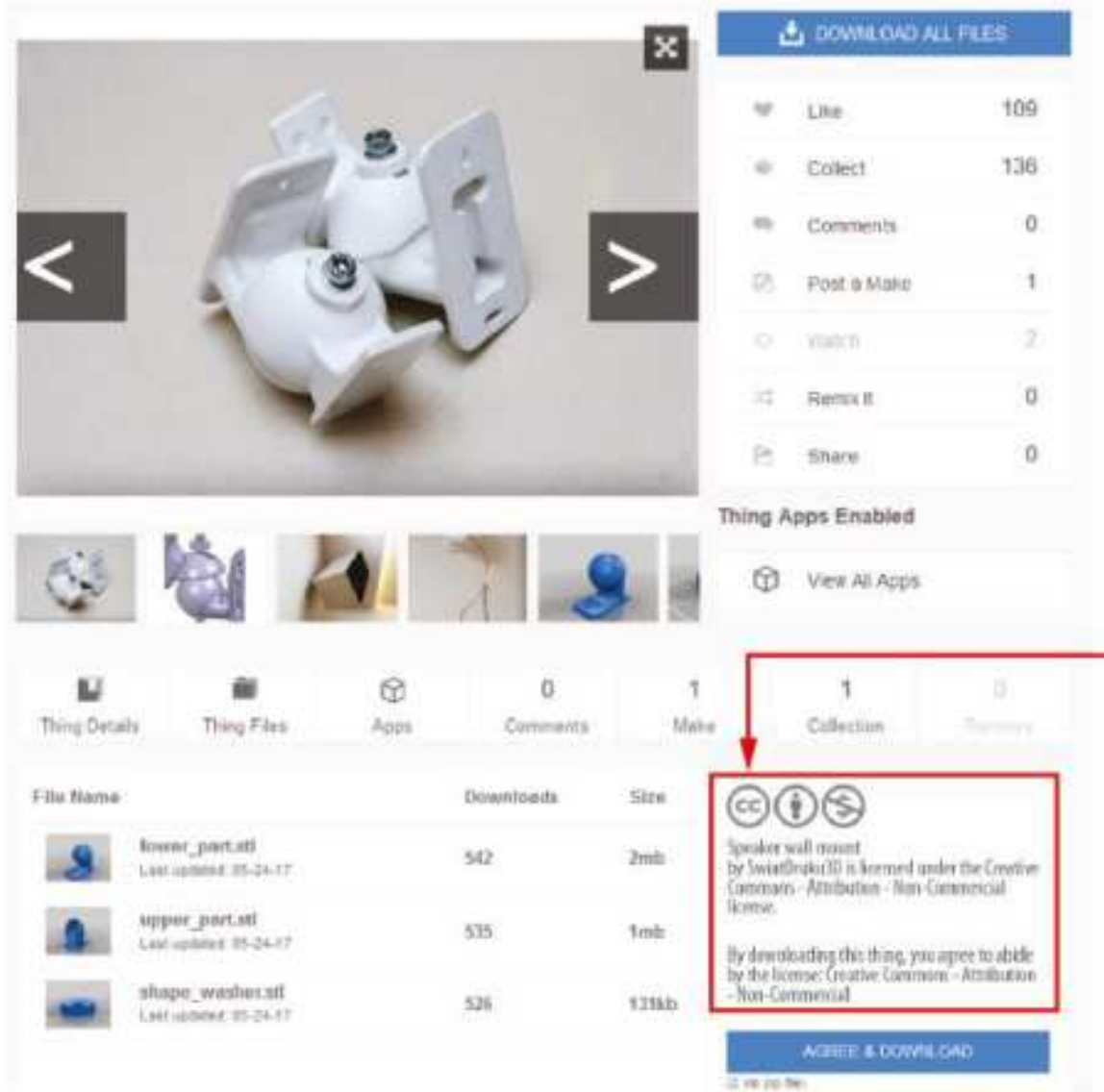
| Nazwa programu | Opis | System operacyjny |
|---|---|---|
| **Tinkercad** | To działający w chmurze, łatwy w obsłudze program dla początkujących. Choć ma niewielką funkcjonalność, wystarcza do stworzenia prostych projektów. Wymaga założenia konta, które daje dostęp także do innych programów on-line producenta. Program eksportuje do formatów OBJ i STL. Przygotowano edycję w języku polskim. | on-line |
| **DesignSpark** | Program udostępniono w wersji całkowicie darmowej. Eksportuje pliki w popularnych formatach. Wymaga darmowej rejestracji. Menu przygotowano w języku polskim. Przygotowano także wersję komercyjną tego programu. | Windows |
| **BRL-CAD** | To zaawansowany darmowy program do projektowania CAD (ang. Computer Aided Design). Jego możliwości są zbliżone do oferty profesjonalnych edytorów. Eksportuje pliki w popularnych formatach. | Windows, OS X, Linux |
| **FreeCAD** | Kolejny program klasy CAD o stosunkowo łatwej obsłudze. Jest darmowy. Nie udostępniono polskiej wersji. | Windows, OS X, Linux |

## 40.3. Udostępniane w sieci, czyli nie wszystko musisz robić od podstaw

projekty dla drukarek

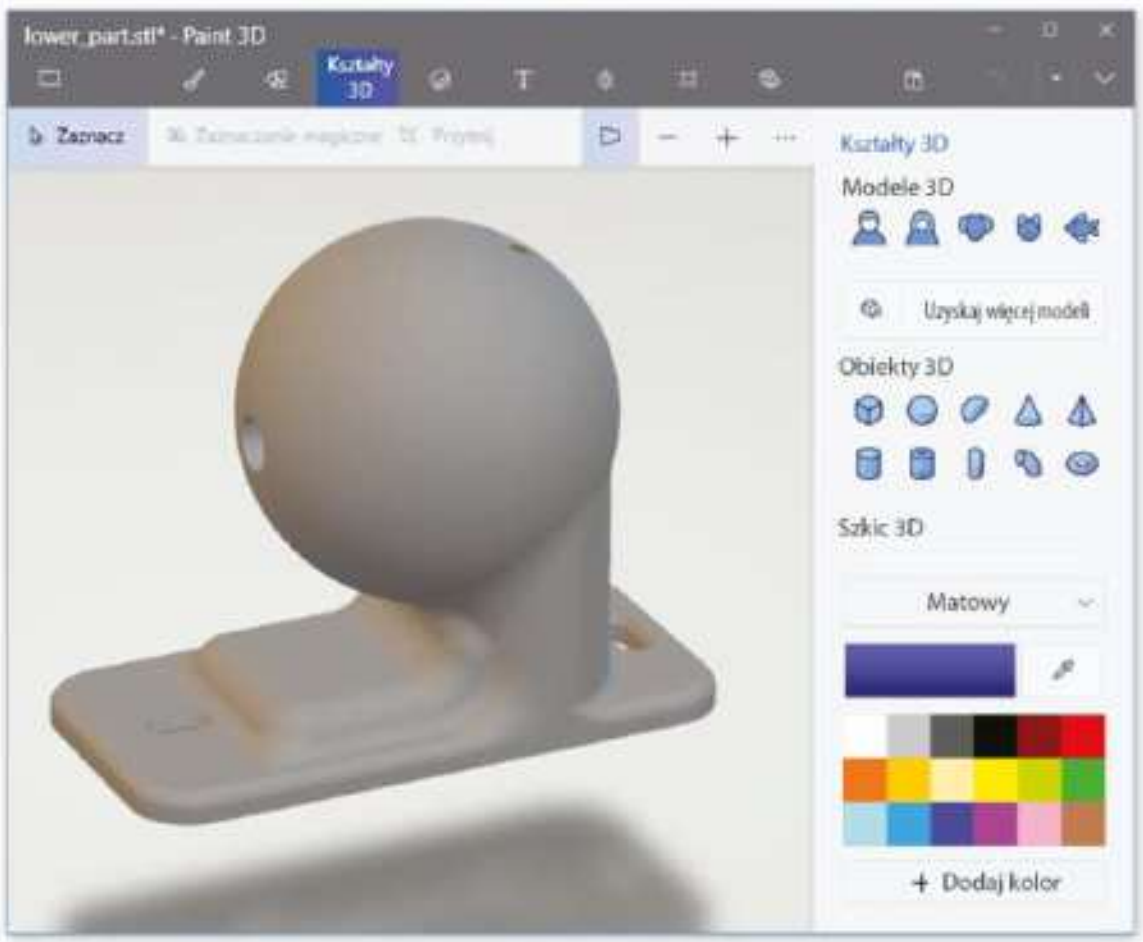
Projekty dla drukarek 3D są zapisywane w plikach o standardowych parametrach. Nietrudno się domyślić, że są one dostępne także w sieci. Niektóre są oferowane darmo, a inne za opłatą. Jeśli istnieje potrzeba wykonania jakiegoś nietypowego elementu, można go zamówić w specjalistycznej firmie. Dla początkujących najciekawszą ofertę stanowią oczywiście projekty gotowe do natychmiastowego wykorzystania. Możesz wykorzystać projekty darmowe lub komercyjne sprzedawane w sklepach internetowych. Przed ich pobraniem należy zwrócić uwagę na to, czy nie naruszają czyichś praw autorskich i praw majątkowych (rys. 40.6.).

Projekty różnią się od ilustracji na stronie, z której zostały pobrane. Jak przeglądać pobrane pliki? Można to zrobić, wczytując projekt do edytora 3D, na przykład użytkownicy Windows skorzystają z nowej wersji programu Paint oznaczonego jako 3D (rys. 40.7.).




CC oznacza licencję Creative Commons. Taki projekt można wykorzystać bez opłat. Pozostałe ikony dotyczą zakazu wykorzystywania projektu do celów komercyjnych.

Rys. 40.6. Przykładowy projekt dla drukarki 3D dostępny w sieci




Rys. 40.7. Przeglądanie pobranego pliku STL w programie Paint 3D

**Uwaga!**

Część oferowanych w sieci modeli do druku 3D jest rozprowadzana nielegalnie. Dotyczy to głównie figurek postaci z filmów i gier. Jeśli masz podejrzenia co do legalności plików, nie pobieraj ich i nie rozpowszechniaj.

## ZADANIA DO WYKONANIA

1. Znajdź w sieci blogi użytkowników drukarek 3D. Zapoznaj się z opisywanymi problemami i sukcesami. Wynotuj parametry, na które musisz zwrócić uwagę podczas wyboru modelu drukarki. Wybierz drukarkę 3D w cenie do 3000 złotych. Uzasadnij swoją decyzję analizą jej parametrów i opiniami umieszczonymi na forach dyskusyjnych.
2. Na podstawie znalezionych w sieci informacji (cena filamentu, projektu itp.) oblicz koszt wydruku dowolnego modelu znalezionego w sieci. Sprawdź, czy w opisie autor podał dodatkowe dane do oszacowania kosztu wydruku.
3. Znajdź w sieci projekty 3D udostępniane na licencji CC. Wybierz jeden z nich. Pobierz i wczytaj do Paint 3D. Sprawdź, w jakich formatach możesz zapisać projekt z tego programu i wskaż, który z nich może służyć do bezpośredniego wydruku na drukarce.
4. Znajdź w sieci i obejrzyj filmy pokazujące działanie ręcznych i stołowych skanerów 3D. Opisz krótko czynności przygotowujące do skanowania i skanowanie obiektów. Wymień zalety jednego i drugiego rozwiązania.
5. Znajdź w sieci informacje i opinie na temat edytorów wymienionych w tabeli 40.1. Wybierz jeden z nich i uzasadnij swoją decyzję.

## PODSUMOWANIE LEKCJI

s. 222 ▸ • **Drukarka 3D** nakłada kolejno na siebie warstwy roztopionego filamentu. **Filament** jest podawany do głowicy w postaci elastycznego pręcika.

s. 222 ▸ • W drukarkach 3D **głowica może poruszać się w dwóch kierunkach**. Trzeci kierunek zapewnia ruchoma podstawa, na której powstaje wydruk. Istnieją rozwiązania techniczne drukarek 3D z głowicą poruszającą się w trzech kierunkach i nieruchomą podstawą.

s. 222 ▸ • **Modele (obiekty) do drukowania** w trzech wymiarach tworzy się za pomocą edytora grafiki trójwymiarowej lub poprzez skanowanie 3D.

s. 224 ▸ • **Gotowe projekty można pobrać z sieci za darmo** (na licencji CC) lub kupić w sklepie internetowym.

# 41. Modelujemy, czyli jak projektować obiekty 3D

**NA TEJ LEKCJI:**

- poznasz chmurę i aplikację Tinkercad;
- stworzysz swój pierwszy projekt dla drukarki 3D.

Wiesz już, jak działają drukarki 3D i które programy służą do edycji modeli. Umiesz także szukać gotowych modeli w odpowiednim formacie i importować je. Czas na samodzielną edycję i stworzenie prostego modelu, który będzie można wydrukować na drukarce 3D. Skorzystasz z dwóch darmowych edytorów z polskim menu i możliwością eksportu pliku w najpopularniejszych formatach.

## 41.1. Bez instalacji, czyli edytor 3D w chmurze

◂ Tinkercad

Z pewnością znasz wiele programów on-line. Skoro istnieją edytory rozmaitych dokumentów, to warto poszukać także tych do modelowania 3D. Wśród nich znajdziesz **Tinkercad** ze strony *www.tinkercad.com* (rys. 41.1.). Na stronie startowej trzeba założyć konto niezbędne do korzystania z edytora. Zanim jednak to zrobisz, możesz zapoznać się z właściwościami programu i tutorialami pokazującymi najważniejsze narzędzia edytora. Możliwe jest też przejrzenie galerii gotowych projektów dostępnych dla każdego z użytkowników.




Rys. 41.1. Okno startowe edytora grafiki 3D on-line

Korzystając z Tinkercad, poznasz niektóre mechanizmy tworzenia modeli 3D i zdobędziesz umiejętności, które w przyszłości pomogą ci w pracy z bardziej rozbudowanymi edytorami. Edycja w trzech wymiarach znacznie różni się od klasycznej na płaszczyźnie. Zacznijmy od najprostszych czynności w przykładzie modyfikowania gotowego projektu.

Zakładanie konta nie sprawi ci trudności. Jeśli nie chcesz zakładać konta w chmurze, możesz zalogować się poprzez jedną z usług społecznościowych (rys. 41.2.).



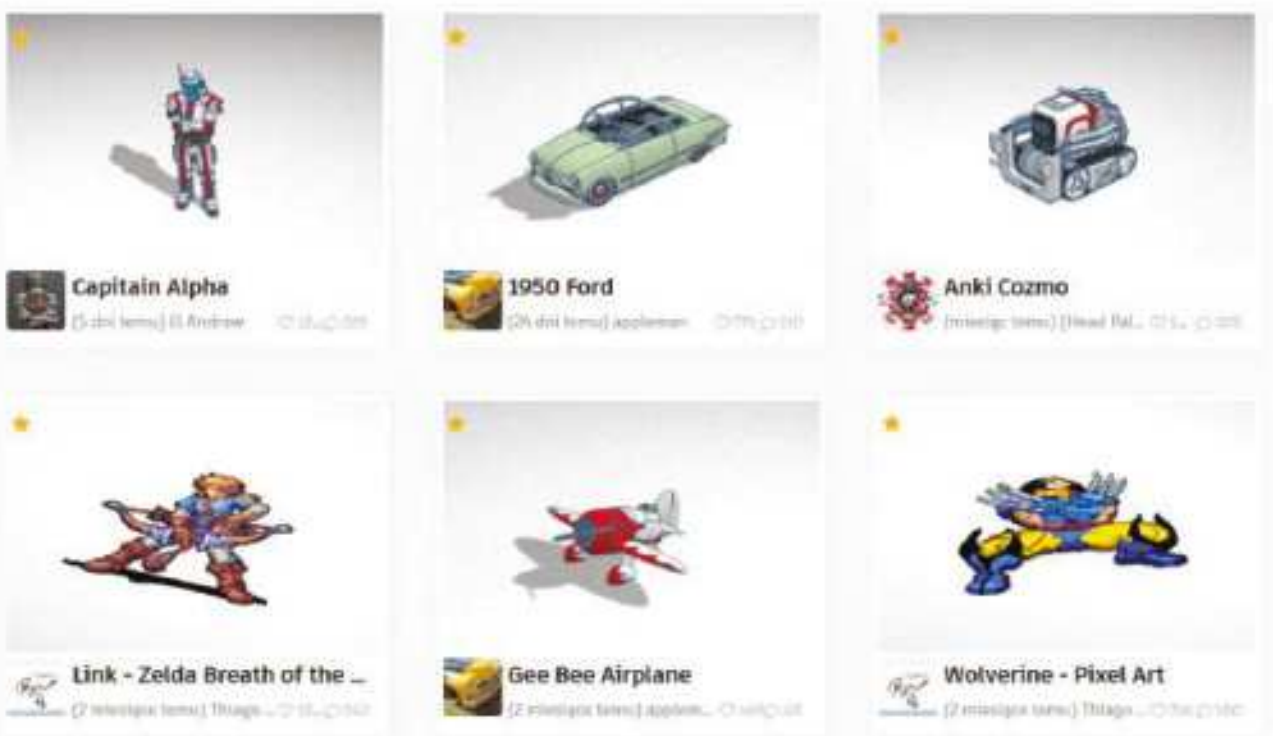

Rys. 41.2. Portale społecznościowe, które mogą służyć do logowania do Tinkercad

Przed przystąpieniem do pracy warto skorzystać z pomocy oferowanej przez chmurę. Dzięki temu szybko opanujesz podstawowe czynności i posługiwanie się narzędziami (rys. 41.3.).




Rys. 41.3. Podstawowe opcje pomocy w chmurze Tinkercad

Po wstępnym zapoznaniu się ze środowiskiem programu czas na ćwiczenie. Zacznij od przejrzenia projektów w opcji **Galeria** (rys. 41.4.). Po naprowadzeniu kursora na pole projektu możesz obracać obiekt w płaszczyźnie poziomej.



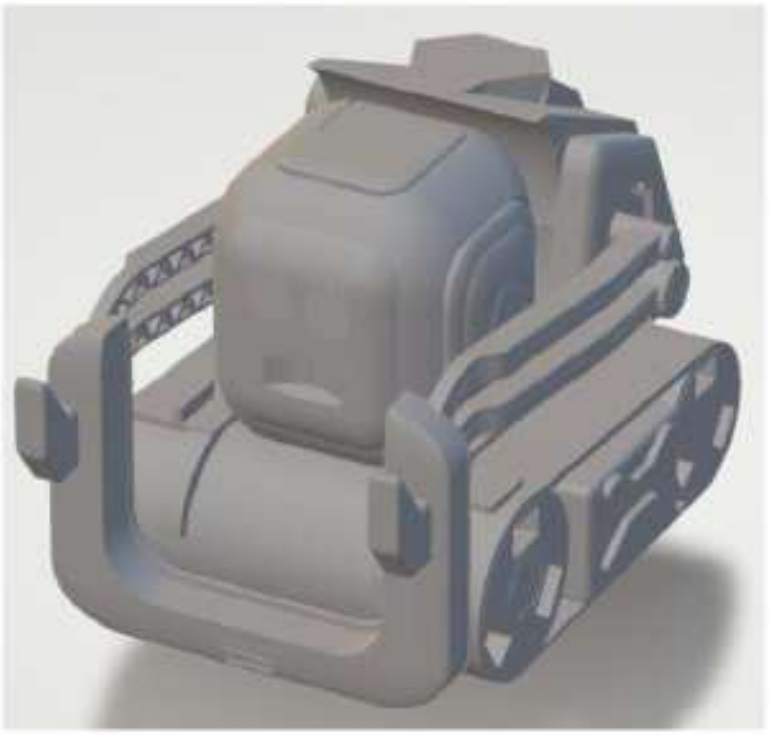

Rys. 41.4. Przykłady projektów dostępnych w chmurze Tinkercad

Do ćwiczenia wybraliśmy projekt buldożera.




Rys. 41.5. Proces pobierania projektu z galerii chmury Tinkercad

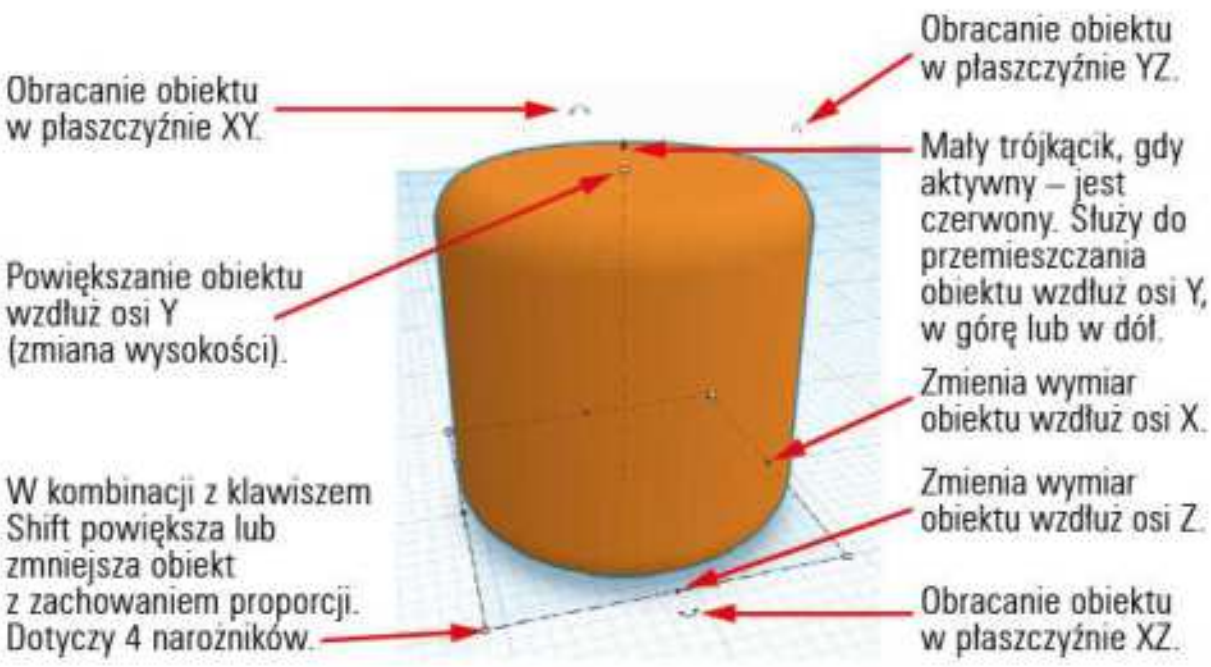
edytor Paint 3D ▸ Pobrany projekt możesz obejrzeć w programie Paint 3D (rys. 41.6.). Będzie pozbawiony kolorów, ale dokładnie odwzoruje bryłę do wydruku.



Rys. 41.6. Projekt wyświetlony za pomocą Paint 3D

## 41.2. Komin i napis, czyli modyfikujemy projekt

Dorysujmy do projektu komin – rurę wydechową silnika i napis na przedniej części karoserii. W trakcie ćwiczenia wykorzystaj umiejętności zdobyte na lekcji na temat chmury oraz podpowiedzi z rysunków 41.7. i 41.8., na których przedstawiono funkcje punktów kontrolnych i przeznaczenie przycisków myszy.



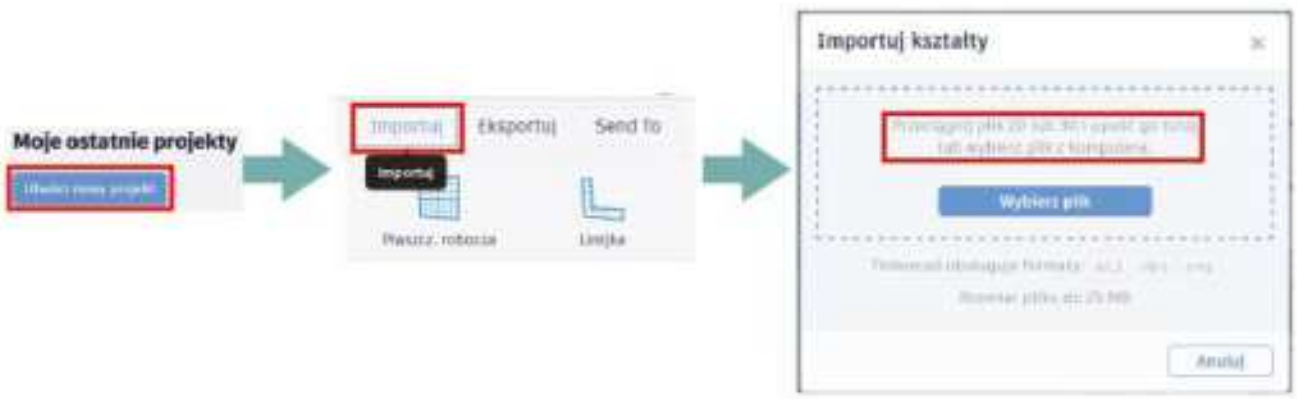

Rys. 41.7. Funkcje punktów kontrolnych w edytorze grafiki 3D




Rys. 41.8. Funkcje przycisków myszy w edycji 3D

Po zalogowaniu się do Tinkercad utwórz nowy projekt i wczytaj pobrany wcześniej plik STL (rys. 41.9.). Poćwicz sposoby podglądu projektu zgodnie z rysunkiem 41.8.

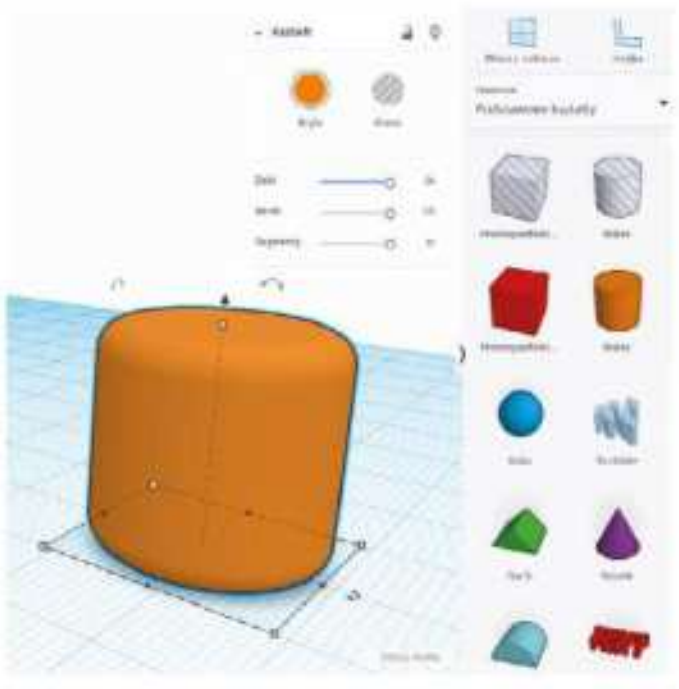
◂ plik STL



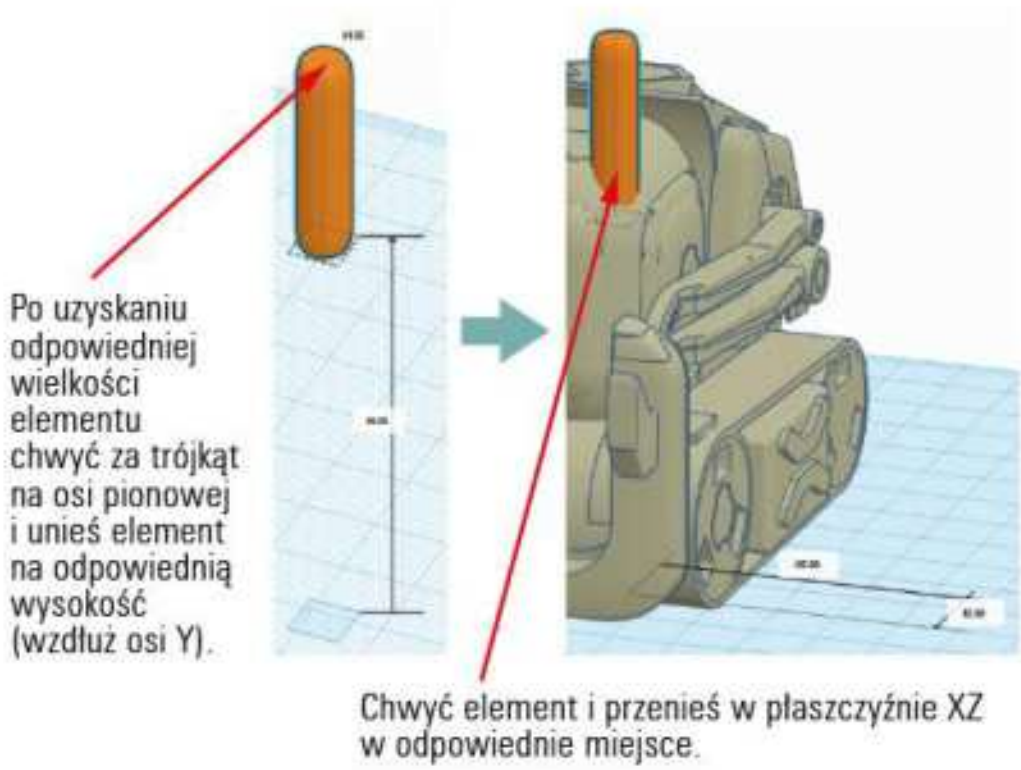

Rys. 41.9. Proces wczytywania projektu z dysku do edytora w chmurze **Tinkercad**

Skoro importowany model jest już gotowy, wprowadź nasze modyfikacje.

Zapewne wstawiony walec będzie miał niepasujące proporcje i wymiary. Wiesz już, jakie znaczenie mają punkty kontrolne edytora. Ustal więc wysokość walca, a następnie umieść go w odpowiednim miejscu (rys. 41.11.). W trakcie pracy poznasz różnice pomiędzy edycją na płaszczyźnie a pracą z użyciem edytora w trzech wymiarach. Szybko jednak przyzwyczaisz się do techniki operowania obszarem roboczym jako przestrzenią. Takie ćwiczenie przyda się także w poznawaniu zagadnień geometrii przestrzennej na lekcjach matematyki.



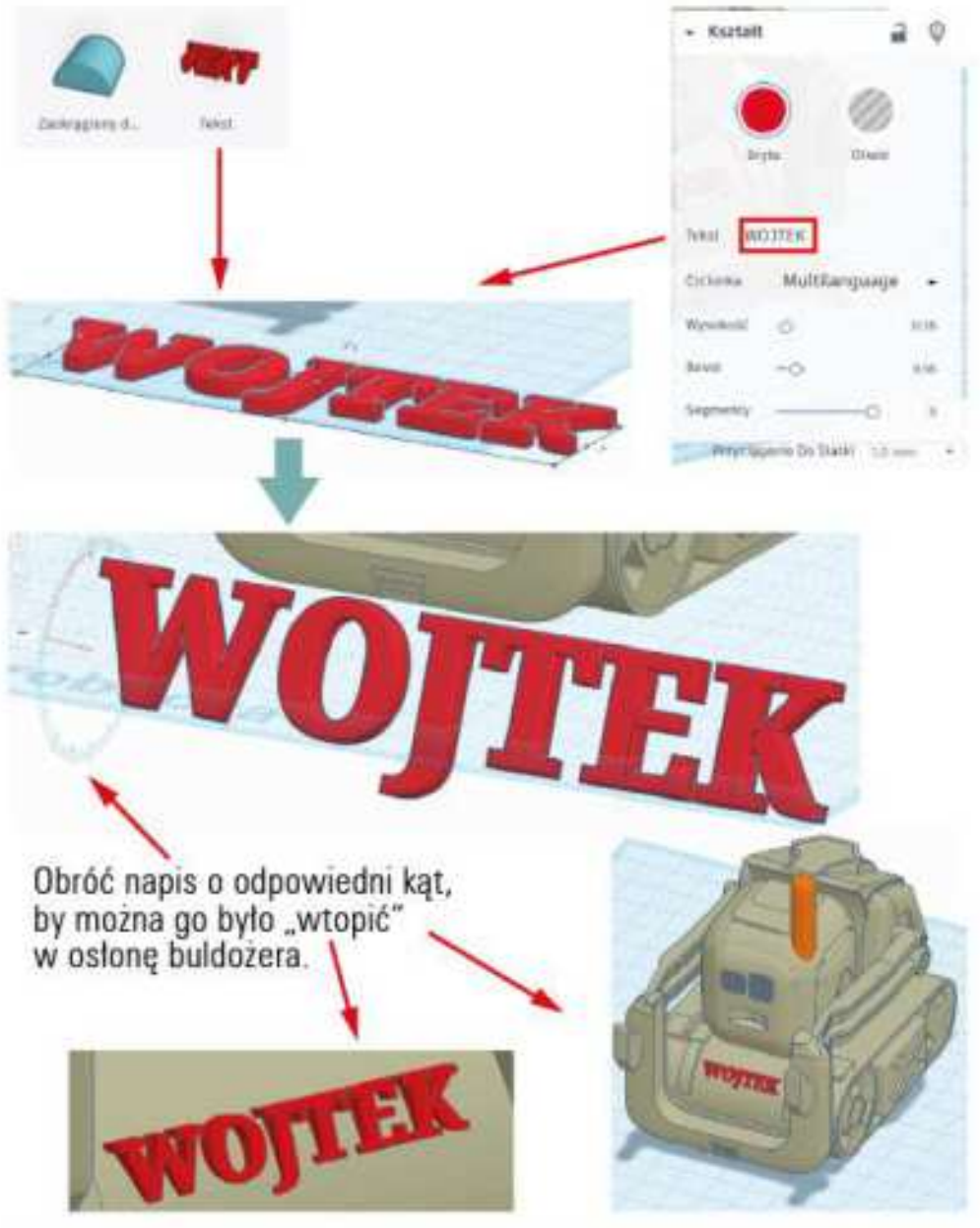
Rys. 41.10. Wstawianie elementu modelu 3D




Rys. 41.11. Przemieszczanie obiektu wzdłuż osi XYZ

Czas na umieszczenie na modelu napisu. Z menu wybierz opcję **Napis** i postępuj według opisu. Zwróć uwagę na to, by grubość liter była wystarczająca do wtopienia ich w karoserię pojazdu. W przeciwnym przypadku będą odstawały od powierzchni buldożera i mogą wystąpić kłopoty z wydrukiem 3D. Użyjesz także narzędzia do obracania wokół osi X. W trakcie pracy sprawdzaj widok modelu z wielu stron, by napis był rzeczywiście dobrze wpasowany w bryłę pojazdu.

**format STL** ▸ Po zakończonej pracy wyeksportuj plik do formatu STL i obejrzyj go w edytorze Paint 3D.




Rys. 41.12. Wstawianie napisu do projektu 3D




Rys. 41.13. Model gotowy do wydruku

**Uwaga!**

Tak jak w innych chmurach w Tinkercad pliki projektów zapisują się automatycznie.

## 41.3. Tabliczka, czyli tworzymy pierwszy projekt dla drukarki 3D

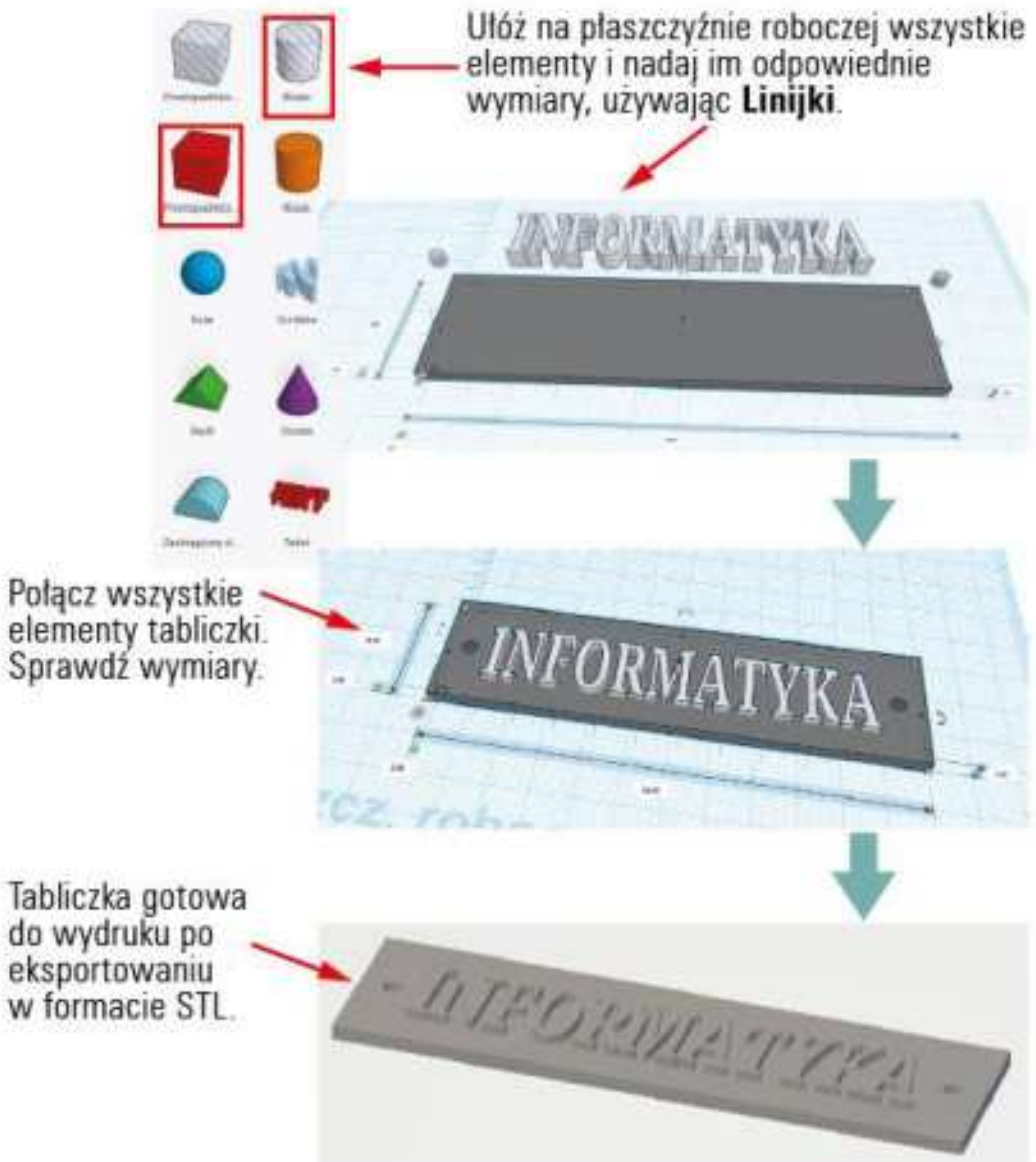
Zaprojektujemy i wykonamy tabliczkę, którą będzie można przymocować do drzwi pracowni informatycznej. Oczywiście pierwszym krokiem jest utworzenie nowego projektu. Postaraj się nadać mu oryginalną nazwę. Ustaw płaszczyznę roboczą w taki sposób, by wygodnie pracować z obiektami.

W tym projekcie **ważne są wymiary** i wzajemne proporcje. Napis musi wtopić się w tabliczkę, a otwory muszą przechodzić na wylot. Osiągniesz to, odpowiednio manipulując punktami kontrolnymi poznanymi wcześniej.

Aby dokładnie dopasować wymiary modelu, wybierz narzędzie **Linijka**. Przyłóż ją w narożniku elementu. W dolnej części okna roboczego możesz zmieniać wymiar jednej małej kratki płaszczyzny roboczej. Domyślnie ustawiono 0,1 mm, co oznacza, że duża kratka podziałki ma boki o długości 1 cm.

◂ narzędzie Linijka

Wysokość walców tworzących otwory powinna być równa grubości tabliczki, a liter – odpowiednio większa, ponieważ muszą wystawać ponad płaszczyznę tabliczki.




Rys. 41.14. Proces tworzenia modelu tabliczki z napisem 3D

Jeśli zainteresowało cię tworzenie modeli trójwymiarowych dla drukarek, skorzystaj z tutoriali udostępnionych w chmurze (rys. 41.15.). Znajdziesz tam wiele ciekawych projektów z dokładnym opisem ich kreślenia. Jeśli język angielski w opisie sprawia ci problem, skopiuj podpowiedzi i użyj tłumacza, na przykład Google.




Rys. 41.15. Przykładowe projekty dostępne w Tinkercad w postaci opisów ich tworzenia

## ZADANIA DO WYKONANIA

1. Korzystając z programu Tinkercad, wykonaj tabliczkę na drzwi z wklęsłym napisem. Zachowaj odpowiednie wymiary i proporcje projektu. Wyeksportuj pracę do formatu STL. Obejrzyj model za pomocą Paint 3D.
2. Znajdź ciekawy model na licencji CC wśród dostępnych w Tinkercad. Edytuj go w tym programie i postaraj się ustalić, z jakich elementów jest zbudowany.
3. Wykonaj projekt breloka do kluczy do sal lekcyjnych w postaci gruszki z wklęsłymi numerami i elementami ozdobnymi. Zadbaj o odpowiedni rozmiar. Zastanów się, jak sprawić, by wnętrze gruszki było puste. Oszczędzi to bardzo dużo filamentu.
4. Wykonaj projekt zabawki – labiryntu. Załóż, że ścianki mają grubość 2 mm, wysokość 1 cm. Na końcu umieść otwór, do którego może wpaść metalowa kulka przemieszczająca się po labiryncie w momencie przechylania zabawki. Wybierz kształt zewnętrzny labiryntu, na przykład prostokąt, kwadrat, koło.
5. Zaprojektuj podstawkę pod telefon komórkowy umożliwiającą ustawienie aparatu pod kątem 60 stopni. Może ona służyć do oglądania filmów lub czytania e-booków bez konieczności trzymania smartfona.

## PODSUMOWANIE LEKCJI

- Jednym z edytorów 3D przeznaczonych dla początkujących grafików i amatorów druku 3D jest program **Tinkercad** pracujący w chmurze. Aby z niego skorzystać, można założyć własne konto lub zalogować się za pomocą jednego z portali społecznościowych lub Google. Tinkercad oferuje wiele samouczków: od podstawowych do opisów tworzenia skomplikowanych projektów. ◂ s. 227
- Gotowy projekt można wyeksportować do formatu **OBJ** lub **STL**. ◂ s. 232
- Pliki dla drukarki 3D w formacie STL można przeglądać w **Paint 3D**. ◂ s. 230
- Wymiary modeli można sprawdzać za pomocą **narzędzia Linijka**. Jest ona także przydatna przy zmianach wielkości elementów lub całych modeli. ◂ s. 233

# 42. Wizualizacja pomysłów, czyli projektujemy dom w edytorze 3D

**NA TEJ LEKCJI:**

- dowiesz się, czym jest i do czego służy SketchUp;
- stworzysz projekt 3D domu i poznasz podstawowe narzędzia edycyjne w SketchUp.

Projektowanie 3D to nie tylko tworzenie modeli do wydruku dla specjalistycznych obrabiarek przemysłowych i innych maszyn. Do tego typu projektów używa się programów CAD. To już wiesz. Zdajesz sobie jednak sprawę, że technologia pozwalająca na uzyskiwanie przestrzennych obrazów służy także wielu innym celom. Większość gier komputerowych powstaje dzięki zaawansowanym edytorom 3D. Projekty architektoniczne, wizualizacje budynków i inne również są wykonywane za pomocą takich programów. Obok wyspecjalizowanych pakietów dla inżynierów istnieją edytory mniej zaawansowane, pozbawione modułów ściśle związanych z technologiami budownictwa, ale dające możliwość pracy twórczej nad wizualizacją projektu. Może właśnie dziś wpadniesz na pomysł budowy swojego wymarzonego domu?

## 42.1. Nowe i stare, czyli społeczność SketchUp prezentuje projekty

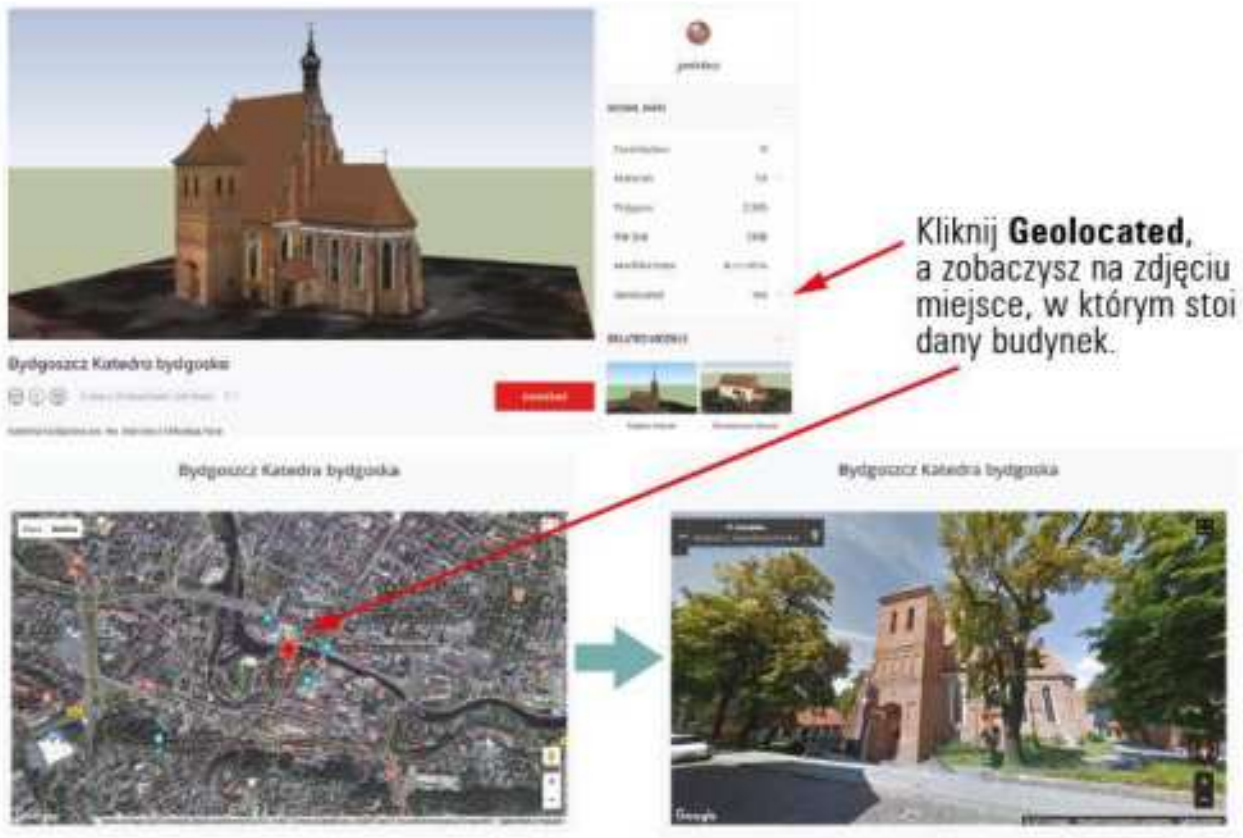
SketchUp ▸

Wśród programów do edycji grafiki 3D znajdują się takie, które dzięki intuicyjnie zaprojektowanym narzędziom i darmowej licencji stały się bardzo popularne wśród miłośników architektury. Wpływ na to ma także udostępnienie darmowej wersji on-line i w postaci aplikacji dla każdego powszechnie używanego systemu operacyjnego. Jednym z najpopularniejszych jest **SketchUp**. Obecnie jego wersja komercyjna jest programem bardzo zaawansowanym i wykorzystywanym przez wielu projektantów. My posłużymy się darmową wersją on-line.

konto na SketchUp ▸

Podobnie jak w przypadku Tinkercad możesz zalogować się do tej chmury kontem Google lub założyć własne konto (twoje projekty będą zapisywane w chmurze). Najlepiej ocenić edytor po dziełach w nim tworzonych. Łatwo je znajdziesz. Ciekawym doświadczeniem będzie szukanie wizualizacji różnych budowli, na przykład z twojej okolicy. Okazuje się bowiem, że użytkownicy odwzorowali w SketchUp całe ulice i dzielnice oraz większość polskich zabytków. Przy okazji możesz zdobyć wiele informacji na temat tych obiektów i zwiedzić okolicę za pomocą aplikacji Google Maps i Google Earth. Takie umiejętności przydadzą się na lekcjach historii i podczas przygotowywania tras turystycznych na wycieczkę klasową (rys. 42.1.).

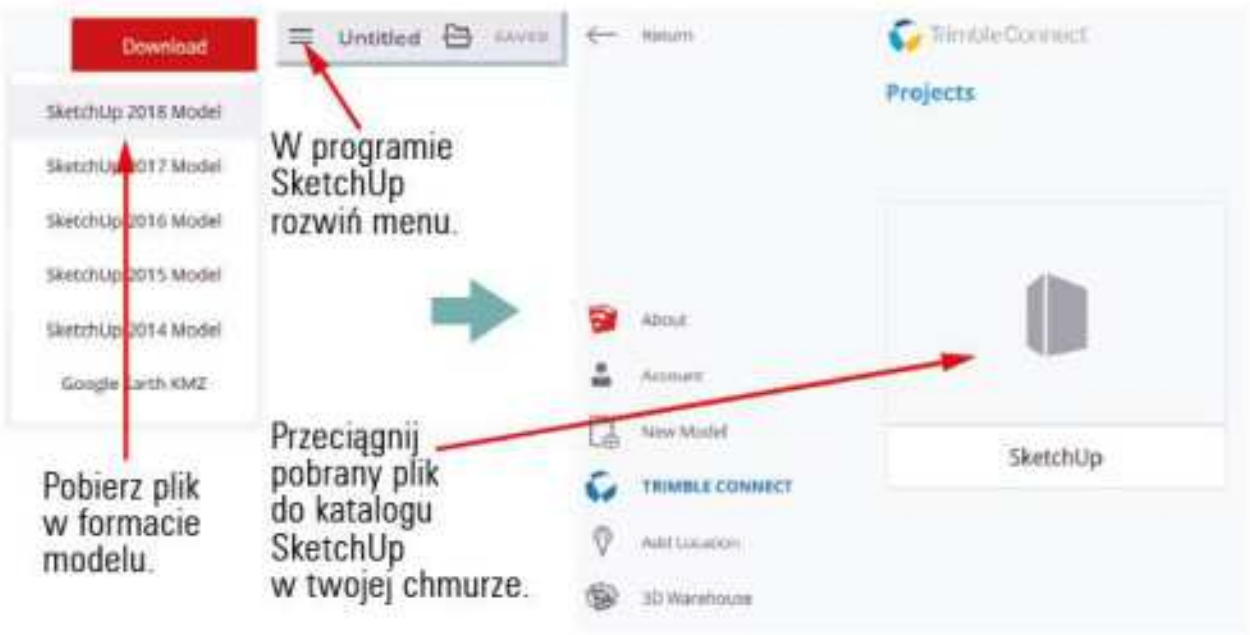
Przy okazji możesz sprawdzić, jak w tych usługach wygląda okolica, w której mieszkasz. Wpisz w wyszukiwarkę frazę: SketchUp Bydgoszcz Fara.




Rys. 42.1. Cyfrowy model budowli i powiązania z mapami Google

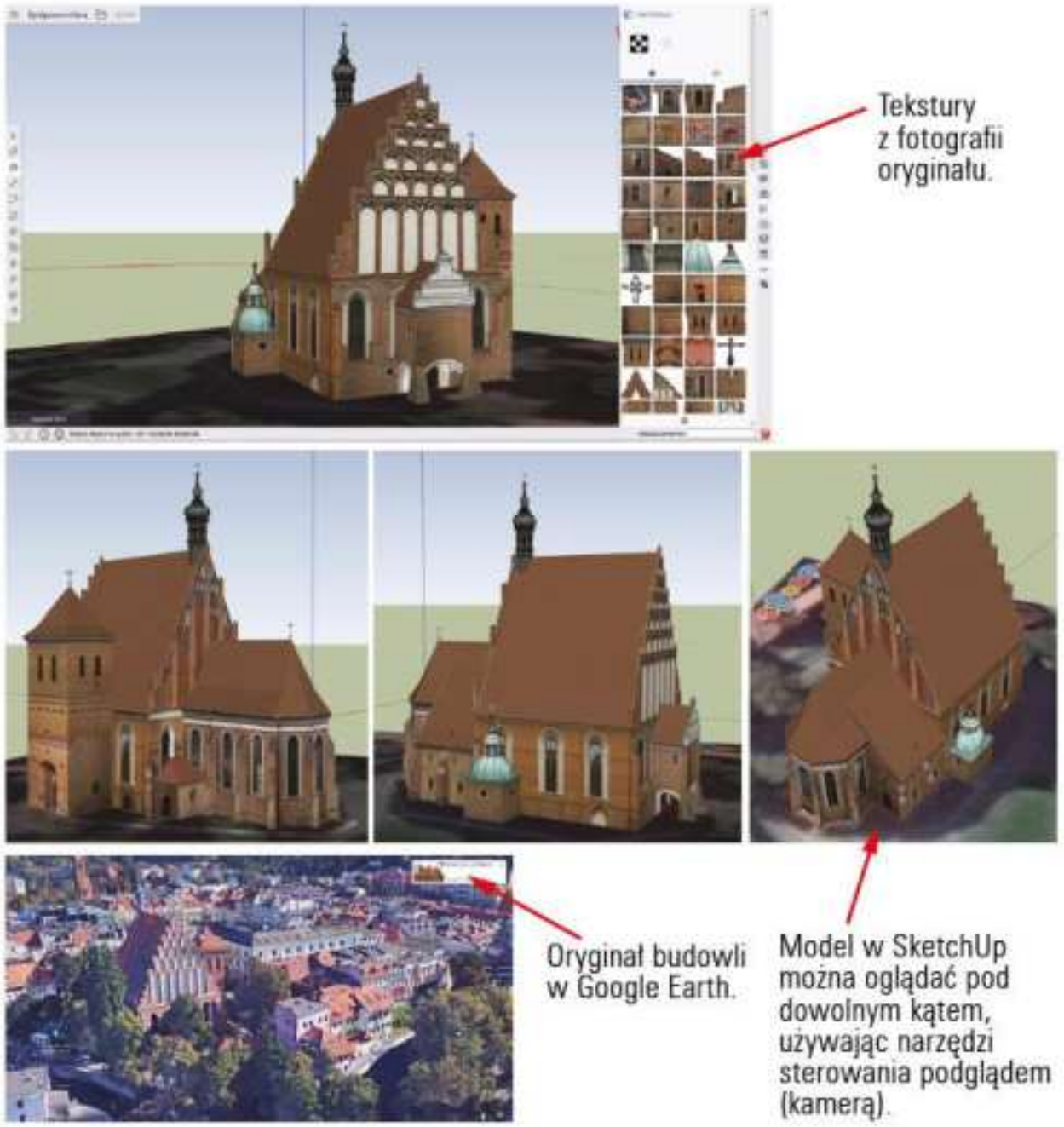
Gdy odbędziesz już wirtualną wycieczkę, możesz pobrać (rys. 42.2.) i obejrzeć dany model ze wszystkich stron. Zauważ, że przykładowy budynek jest pokryty w części teksturami wyciętymi z fotografii oryginału (rys. 42.3.).

◂ makiety budynków




Rys. 42.2. Proces kopiowania modelu do chmury programu SketchUp

Klikając dwukrotnie ikonę wczytanego do chmury projektu, uruchomisz jego edycję (rys. 42.3.).




Rys. 42.3. Pobrany model w edycji w programie SketchUp

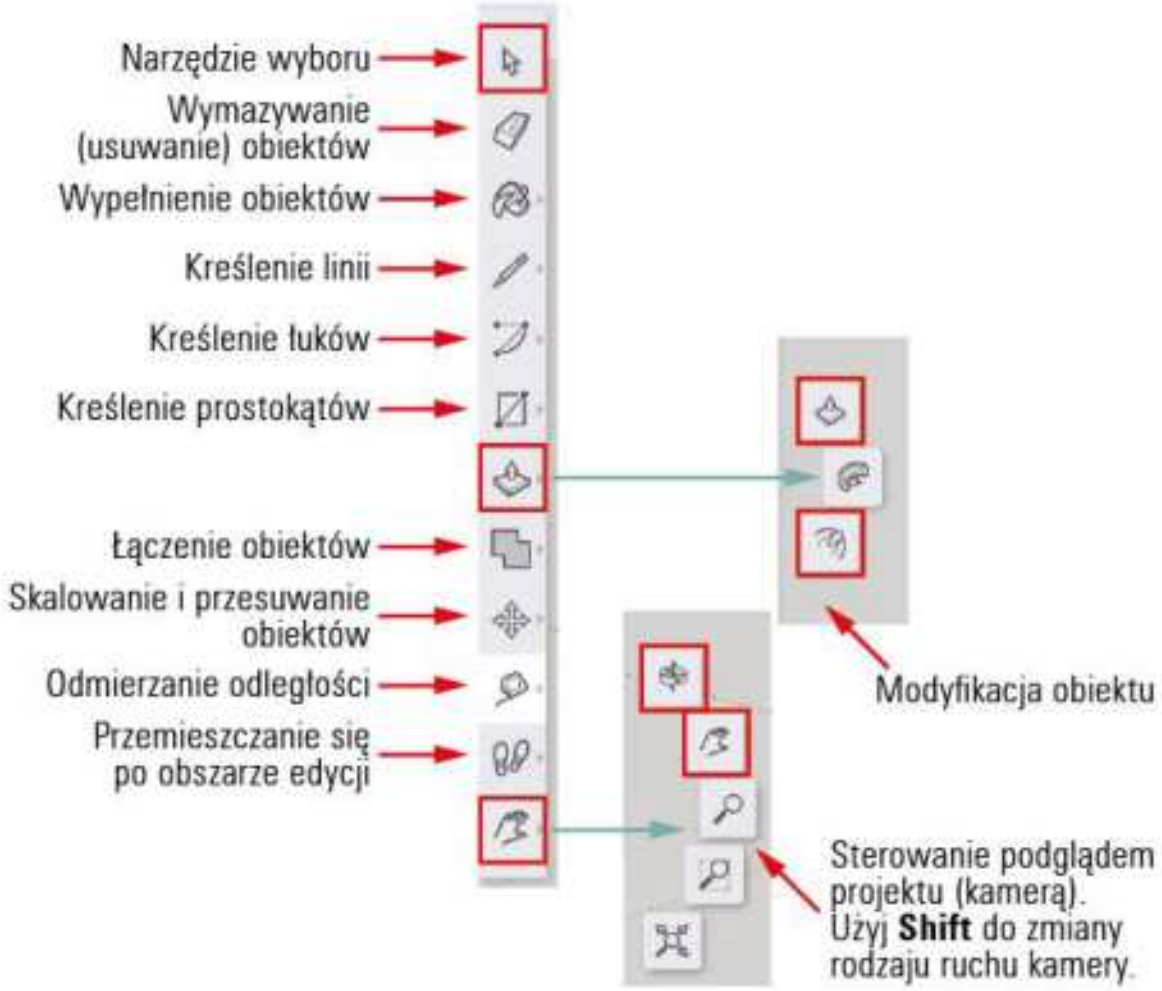
## 42.2. Mój wymarzony dom, czyli tworzymy model budynku w edytorze 3D

Czas na uruchomienie wyobraźni i zaprojektowanie nowego obiektu. Zanim zabierzesz się do pracy, obejrzyj tutoriale, które dokładnie opisują funkcje narzędzi i metody tworzenia projektów budynków. Oryginalne lekcje znajdziesz pod adresem *sketchup.com/learn/videos/*. Obejrzenie wspomnianych filmów jest podstawą do sprawnego wykonania ćwiczenia i szybkiego opanowania metody pracy z edytorem SketchUp. Nie wszystko da się opisać, czasami lepiej zobaczyć to na filmie.

Po obejrzeniu lekcji przygotowanych przez autorów edytora uruchom program w chmurze i rozpocznij projektowanie od utworzenia nowego modelu (opcja **New Model**). Wybierz **Prosty szablon** (**Simple template**) w systemie **metrycznym**. Odległości będą liczone w metrach. Na pustym polu edycyjnym pojawi się postać człowieka. Kliknij ją jeden raz i usuń klawiszem **Del**. Teraz twój edytor jest gotowy do pracy. Przykładowy projekt stworzono, by pokazać użycie podstawowych narzędzi. Twój projekt może wyglądać zupełnie inaczej. Zanim przystąpisz do pracy, uruchom wyobraźnię.

◂ edytor 3D

Jeśli obejrzenie instrukcji producenta programu jest niewystarczające, skorzystaj z rysunku 42.4. Czerwoną ramką zaznaczono szczególnie ważne narzędzia.



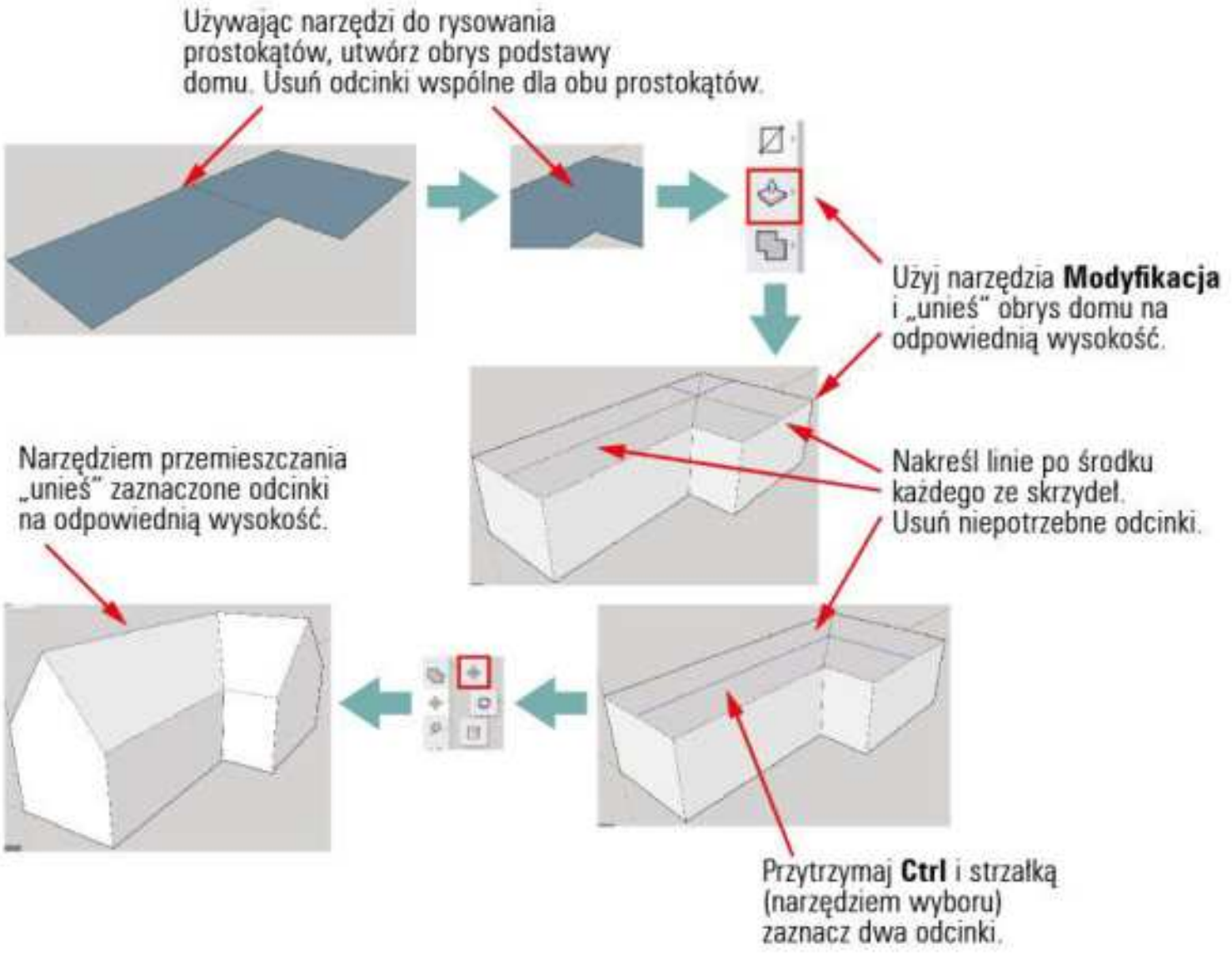

Rys. 42.4. Podstawowe narzędzia edycyjne w **SketchUp**

Pracę rozpoczniemy od narysowania bryły domu (rys. 42.5.). Aby sprawnie wykonać wszystkie czynności, przypomnij sobie metody pracy z narzędziami z obejrzanych filmów producenta.

1. Do usuwania linii i płaszczyzn użyj narzędzia wymazywania (ikona gumki).
2. Linia kończy się i zaczyna w punktach przecięć, a nie w miejscach, w których zaczęto i skończono kreślić. Jest przyciągana do węzłów.
3. Zaznaczanie obiektów, na przykład linii, uzyskasz dzięki Narzędziu wyboru (strzałka).

4. W trakcie kreślenia obiektów pokazują się punkty kontrolne pomagające umiejscowić element w odpowiednim miejscu przestrzeni. Ich kolory są zgodne z osią. Punkt pokaże się także pośrodku odcinka, którego ma dotykać kreślony obiekt.

5. Przesuwanie itp. dotyczy wskazanego obiektu albo zaznaczonych obiektów.




Rys. 42.5. Tworzenie bryły modelu domu

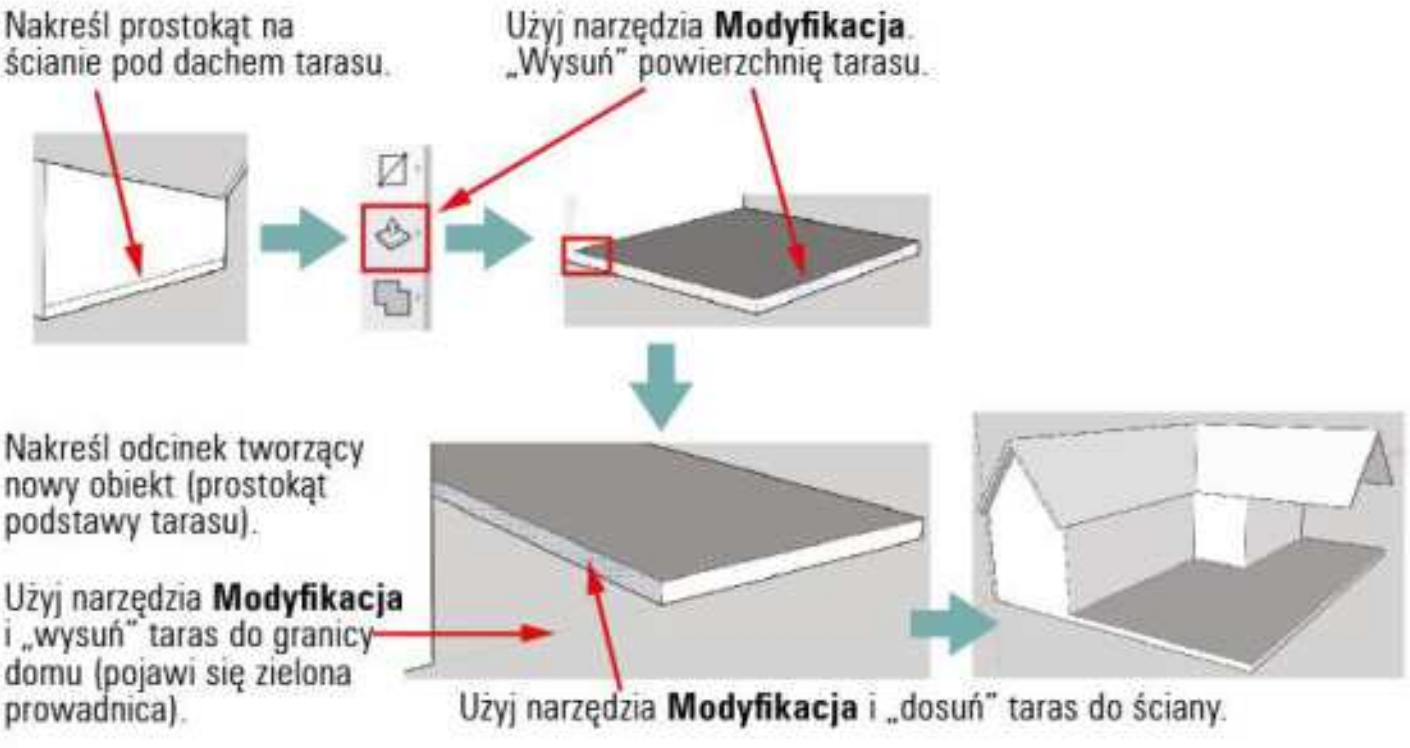
Twój dom nabiera kształtów. Bryłę zmodyfikuj tak, by była zbliżona do rzeczywistych domów. Wydłuż dach, dodaj okna i taras. Użyj do tego znanych już narzędzi – **Kreślenie linii, prostokąta** i **Modyfikacja**.

Pamiętaj, że w kreśleniu bardzo pomocne są punkty kontrolne i sygnalizacja kolorami położenia względem osi.



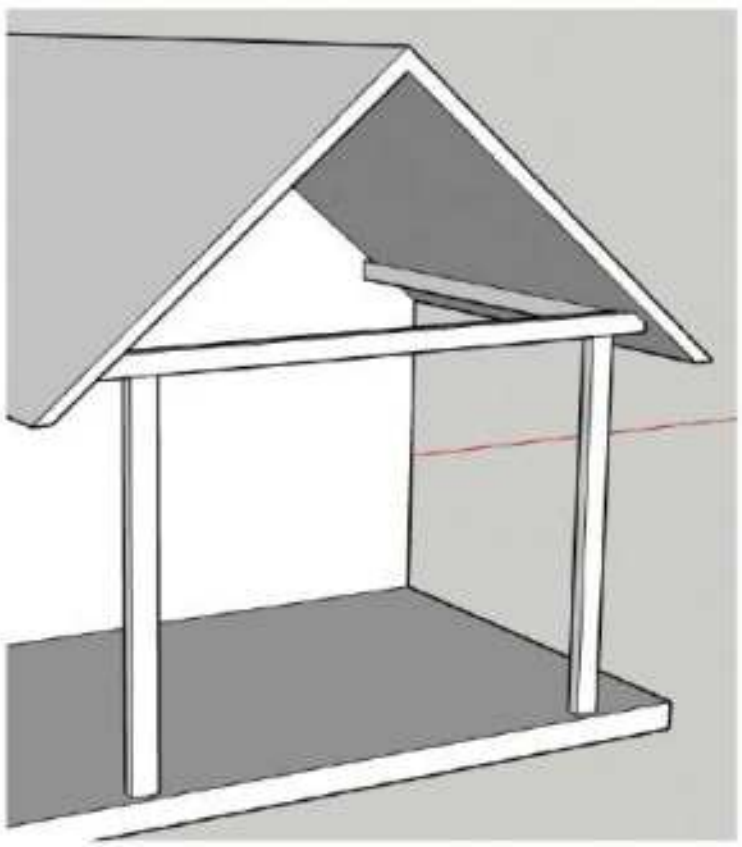

Rys. 42.6. Modyfikacja bryły projektu domu

Domyślasz się, co będzie następnym krokiem w tworzeniu projektu domu? Wykonaj taras, podpory dachu tarasu i okna. Potrzebne do tego będą narzędzia używane do tworzenia i modyfikacji bryły. Pamiętaj o korzystaniu z mechanizmu wyrównywania (rys. 42.7.).



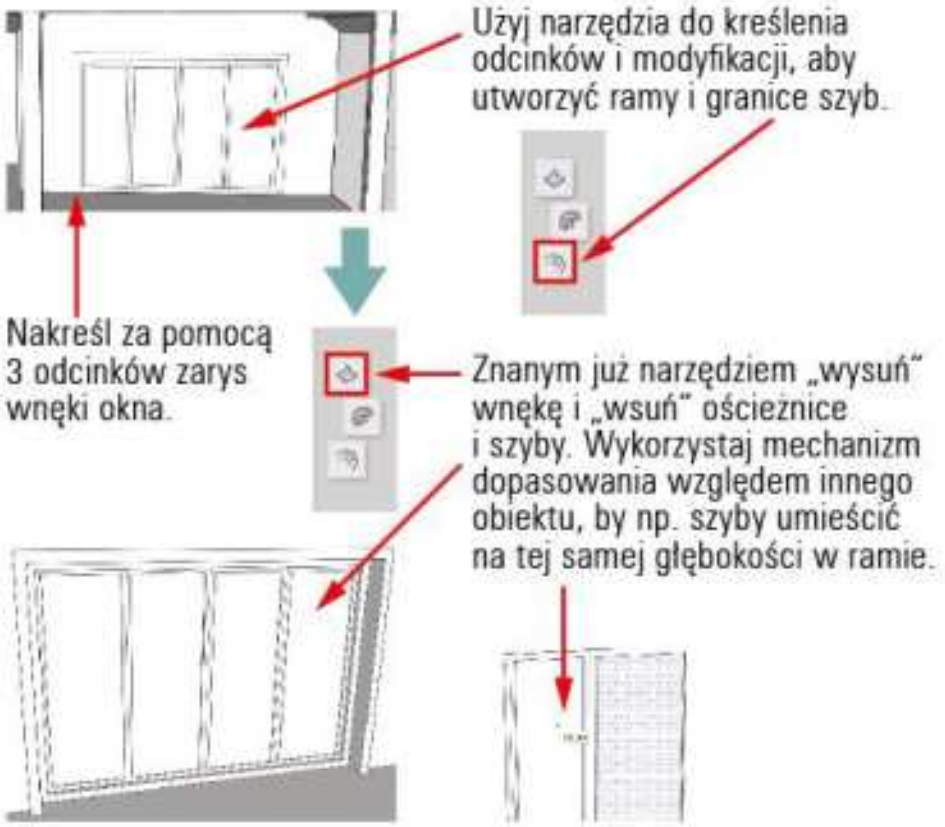

Rys. 42.7. Tworzenie tarasu za pomocą narzędzia Modyfikacja

Jeśli taras jest już gotowy, czas podeprzeć dach. Belki możesz kreślić z odcinków lub prostokątów z użyciem **Modyfikacji** (rys. 42.8.).



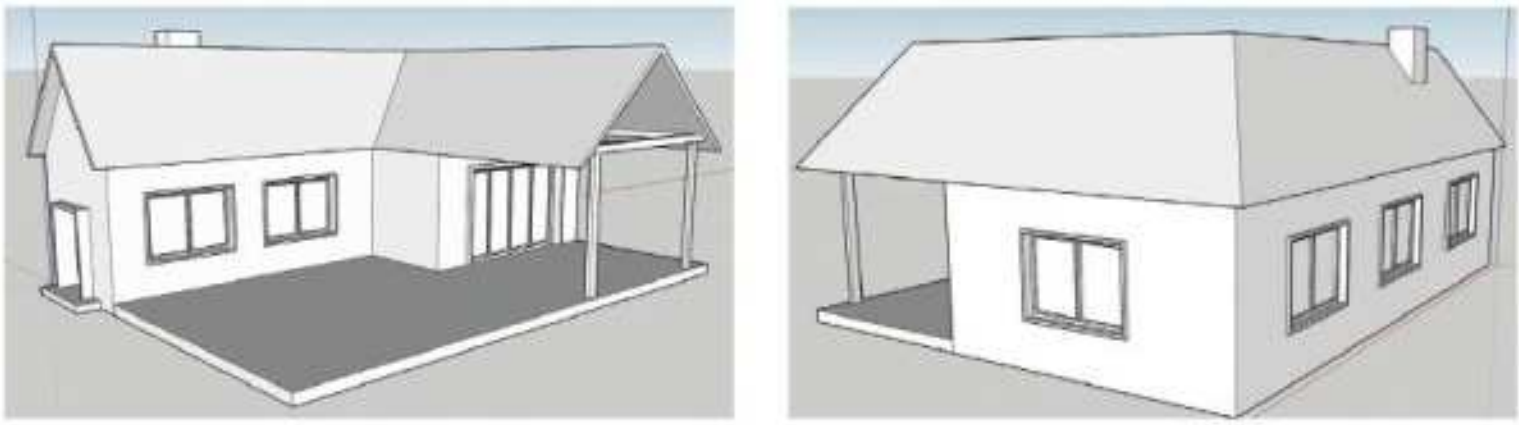
Rysunek 42.8. Belki podtrzymujące dach tarasu

W twoim domu brakuje okien. Postępuj według rysunku 42.9.



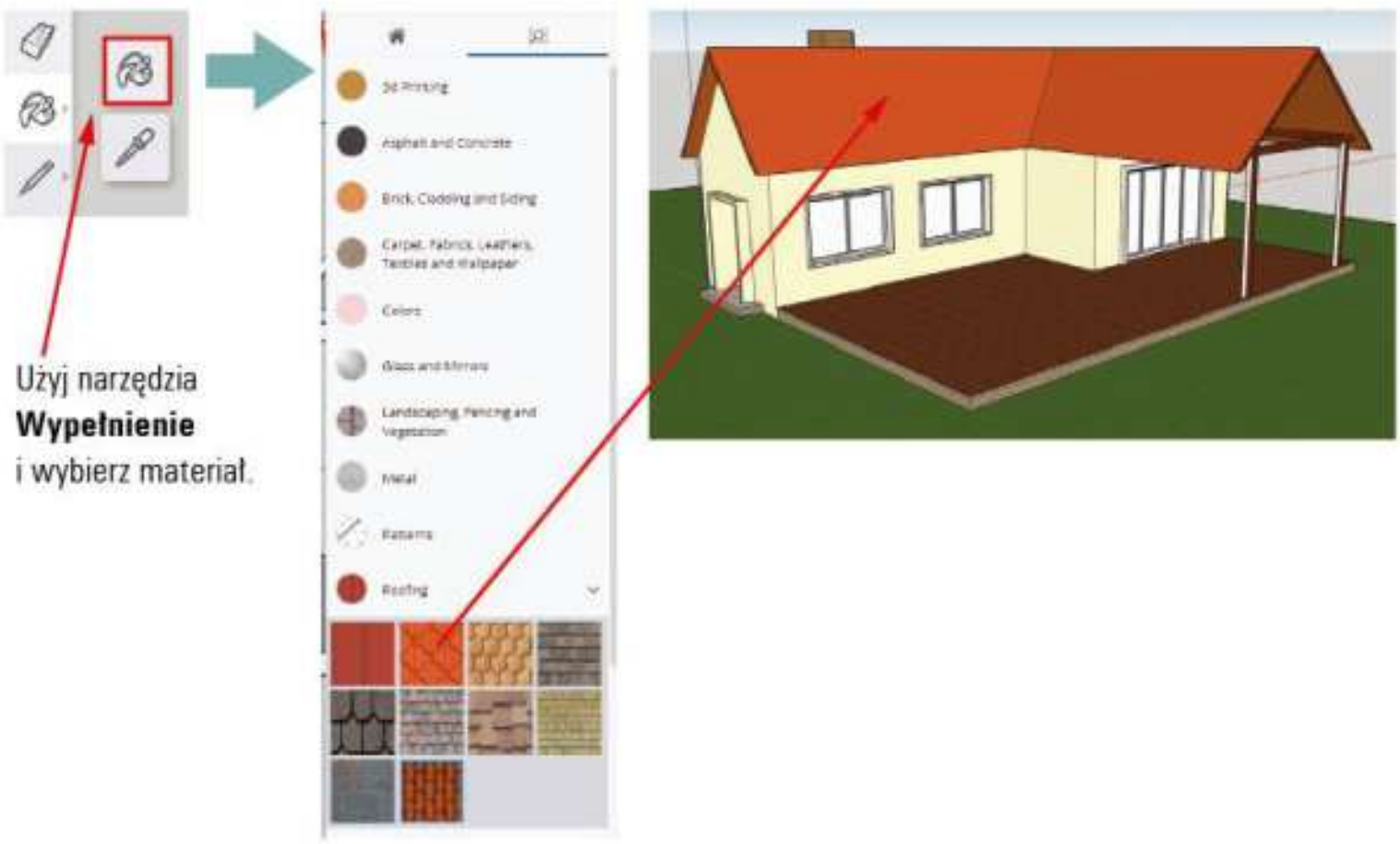

Rys. 42.9. Proces tworzenia okien w projekcie domu

Jeśli wstawiłeś/wstawiłaś już wszystkie okna i drzwi, to projekt domu można uznać za gotowy (rys. 42.10.). Znasz już działanie narzędzi, więc możesz dorysować jeszcze rynny i dopracować szczegóły.



Rys. 42.10. Gotowy projekt bryły domu

Ostatnim aktem tworzenia modelu jest jego kolorowanie i pokrywanie teksturą (rys. 42.11.).




Rys. 42.11. Pokrywanie elementów projektu kolorem i teksturami

## ZADANIA DO WYKONANIA

1. Na stronie *3dwarehouse.sketchup.com* znajdź modele wykonane w SketchUp, które przedstawiają budowle z twojej okolicy. Porównaj je z oryginałami. Wskaż, co można poprawić, aby były bardziej realistyczne lub przedstawiały więcej szczegółów. Sprawdź, jak tę budowlę i twoją okolicę przedstawia się w usługach Google. Wykorzystaj do tego opcję Geolocated.
2. Do modelu stworzonego w trakcie ćwiczenia dodaj detale, np. rynny, teksturę drewna na belkach podtrzymujących daszek, teksturę na drzwiach wejściowych. Oceń, na ile to zmieniło wygląd pracy i czy warto było podjąć ten trud.
3. Do modelu z ćwiczenia dodaj okna dachowe. Przemyśl ich kształt – zdecyduj, czy mają być wpuszczone w dach, zamocowane na nim czy może w postaci lukarny (znajdź znaczenie tego słowa).
4. Zaplanuj wycieczkę klasową. Wykonaj prezentację zawierającą modele budowli wykonane w SketchUp wraz z mapami i widokiem z Google Street View miejsc, które chcesz odwiedzić.

5*. Za pomocą SketchUp wykonaj model 3D budynku twojej szkoły. Postaraj się uwzględnić jak najwięcej szczegółów. Zorganizuj wystawę prac w postaci wydruków.

## PODSUMOWANIE LEKCJI

s. 236 ▸ • Obok zaawansowanych edytorów grafiki 3D, służących do tworzenia wysokiej jakości i rozdzielczości obrazów, np. do gier, istnieją także narzędzia do tworzenia projektów przestrzennych. Jednym z nich jest **SketchUp**. Oprócz wersji darmowej istnieją także bardziej zaawansowane wersje komercyjne.

s. 236 ▸ • W SketchUp on-line możesz **założyć własne konto** z miejscem na swoje projekty na dysku serwera lub logować się za pomocą Google.

s. 237 ▸ • Pasjonaci architektury używają SketchUp do tworzenia **makiet rzeczywistych budynków**, np. zabytków. Można je znaleźć na stronie *3dwarehouse.sketchup.com* i przeglądać lub edytować. Dzięki powiązaniu z mapami i usługami Google można dowiedzieć się, w jakim miejscu zostały zbudowane i obejrzeć okolicę.

s. 239 ▸ • **Edytory 3D** w tym SketchUp są wyposażone w mechanizmy ułatwiające kreślenie i dopasowujące położenie obiektów w przestrzeni do innych obiektów lub do osi, np. podczas przesuwania lub kreślenia. Na **płaszczyzny budynków** można nakładać tekstury i kolory. W edytorze 3D działają wszystkie podstawowe mechanizmy znane z innych programów, **np. zaznaczanie, kopiowanie i wklejanie**.

# 43. Urządzamy mieszkanie, czyli wizualizacje 3D

**NA TEJ LEKCJI:**

- uzyskasz informacje o grafice komputerowej 3D;
- nauczysz się projektować za pomocą Sweet Home 3D.

Początki grafiki komputerowej sięgają lat 60. XX w. Pierwszym znanym grafikiem komputerowym był koreański artysta Nam June Paik (czytaj: nam dżun pejk). Jeden z jego projektów przedstawiał szereg zestawionych ze sobą monitorów. Projekt ten wsparła firma Xerox, która opracowała specjalny graficzny interfejs. W latach 80. XX w. nastąpił rozwój oprogramowania graficznego. Właśnie wtedy dzięki firmie Adobe pojawiły się na rynku takie programy, jak Photoshop czy Illustrator.

Od lat 90. ubiegłego wieku zaczyna się intensywny rozwój grafiki 3D, która jest używana na szeroką skalę do dnia dzisiejszego w technice i kinematografii.

◂ grafika 3D

Razem z grafiką 3D zaczęło rozwijać się również komputerowe wspomaganie projektowania w grafice 2D i 3D. Dzisiaj tzw. programy CAD (czytaj: kad, skrót pochodzi od angielskiej nazwy computer aided design, czytaj: kompjuter ejded dizajn, oznaczającej projektowanie wspomagane komputerowo) są nieodzownym elementem projektowania i wizualizacji w architekturze, inżynierii, a także podczas rozpraw sądowych.

◂ projektowanie w grafice 2D i 3D

W tym rozdziale zajmiemy się wizualizacją 3D mieszkania za pomocą programu Sweet Home 3D.

## 43.1. Sweet Home 3D – mój pierwszy projekt

Program Sweet Home 3D, który z łatwością znajdziesz w internecie, jest darmowym programem służącym do projektowania wnętrz budynków. Pozwala on na projektowanie planów budynku, rozmieszczaniu mebli w gotowym już projekcie na planie 2D z podglądem 3D. A co najważniejsze – jest dostępny w języku polskim.

◂ Sweet Home 3D

Jakie możliwości daje ten program? Dzięki niemu możemy:

- precyzyjnie narysować ściany naszego mieszkania;
- wstawić stolarkę okienną;
- wstawić meble dostępne w dość bogatej bazie danych programu;
- obejrzeć projekt w 3D;
- tworzyć realistyczny obraz komputerowy oraz animacje z uwzględnieniem światła słonecznego;
- korzystać z bogatej bazy modeli 3D;
- wydrukować i wyeksportować nasz projekt.

## Instalacja oprogramowania

Po pobraniu programu (*www.sweethome3d.com/pl*) przechodzimy do instalacji. Tak jak z każdym innym programem klikamy dwa razy prawym przyciskiem myszy i uruchamiamy instalator.

**Krok 1:** Wybieramy język instalacji.



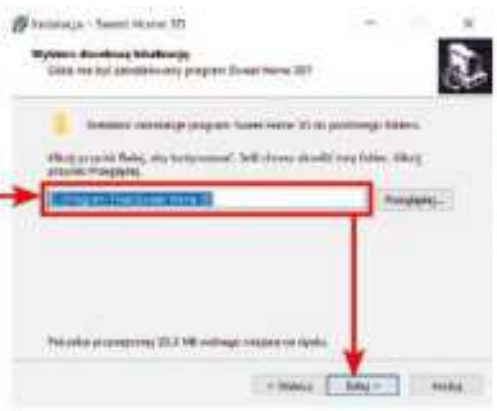

**Krok 2:** Uruchamiamy kreator instalacji. Klikamy przycisk „Dalej".




**Krok 3:** Akceptujemy warunki umowy i klikamy przycisk „Dalej".



**Krok 4:** Wybór docelowego miejsca instalacji. Po wyborze klikamy przycisk „Dalej".



**Krok 5:** Wybieramy folder Menu Start, gdzie program będzie tworzył skróty, klikamy przycisk „Dalej".



**Krok 6:** Wybieramy opcję utworzenia skrótu na Pulpicie. Po zaznaczeniu lub odznaczeniu pola wyboru klikamy przycisk „Dalej".



**Krok 7:** Rozpoczynamy instalację programu. Klikamy przycisk „Instaluj".



**Krok 8:** Kończymy instalację programu. Jeżeli zostawimy zaznaczone pole wyboru „Uruchom program Sweet Home 3D", to po kliknięciu przycisku „Zakończ" program się uruchomi.

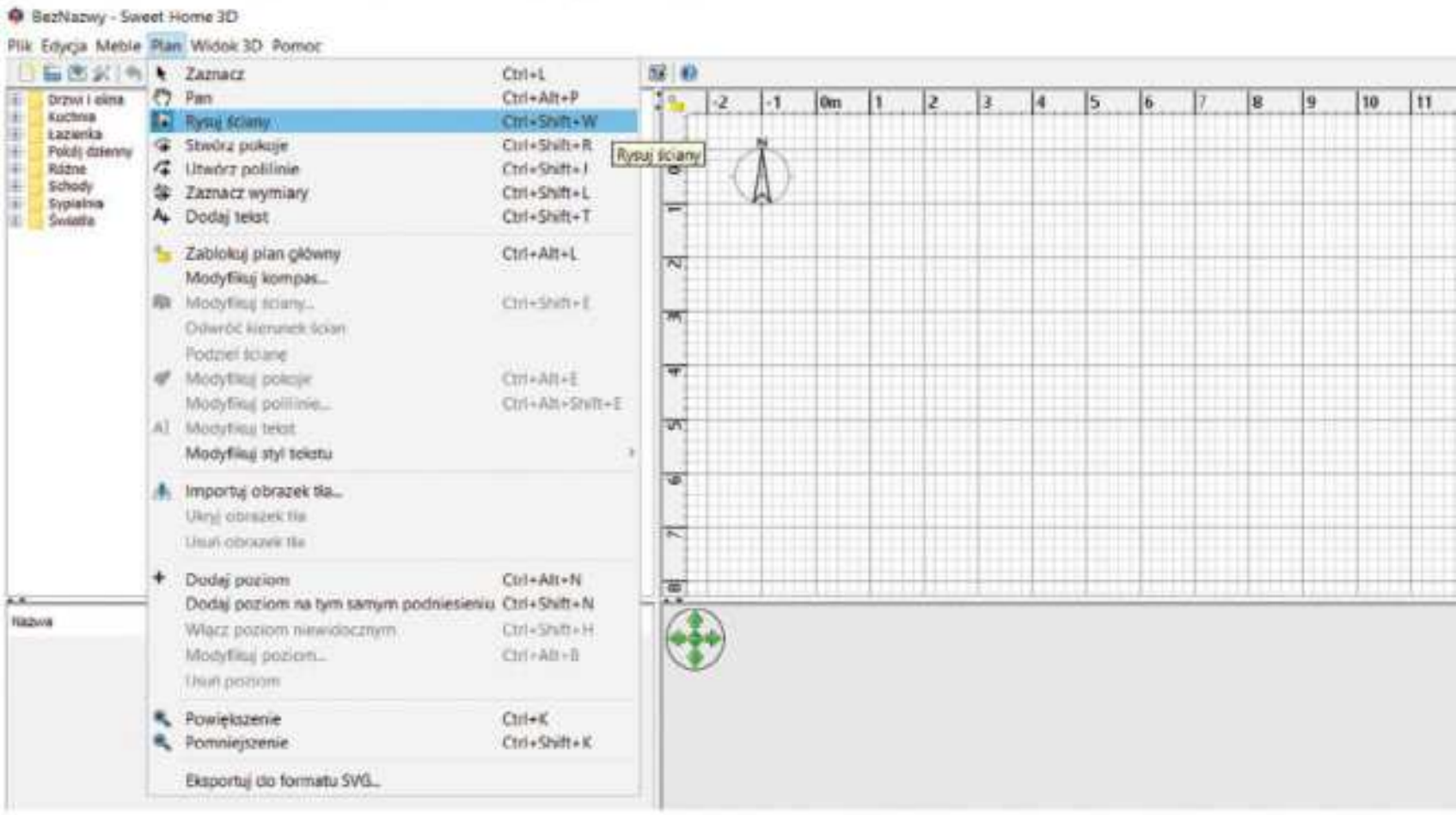



Rys. 43.1. Instalacja programu Sweet Home 3D

Po uruchomieniu programu Sweet Home 3D mamy otwarty od razu nowy projekt i możemy przystąpić do projektowania naszego mieszkania.

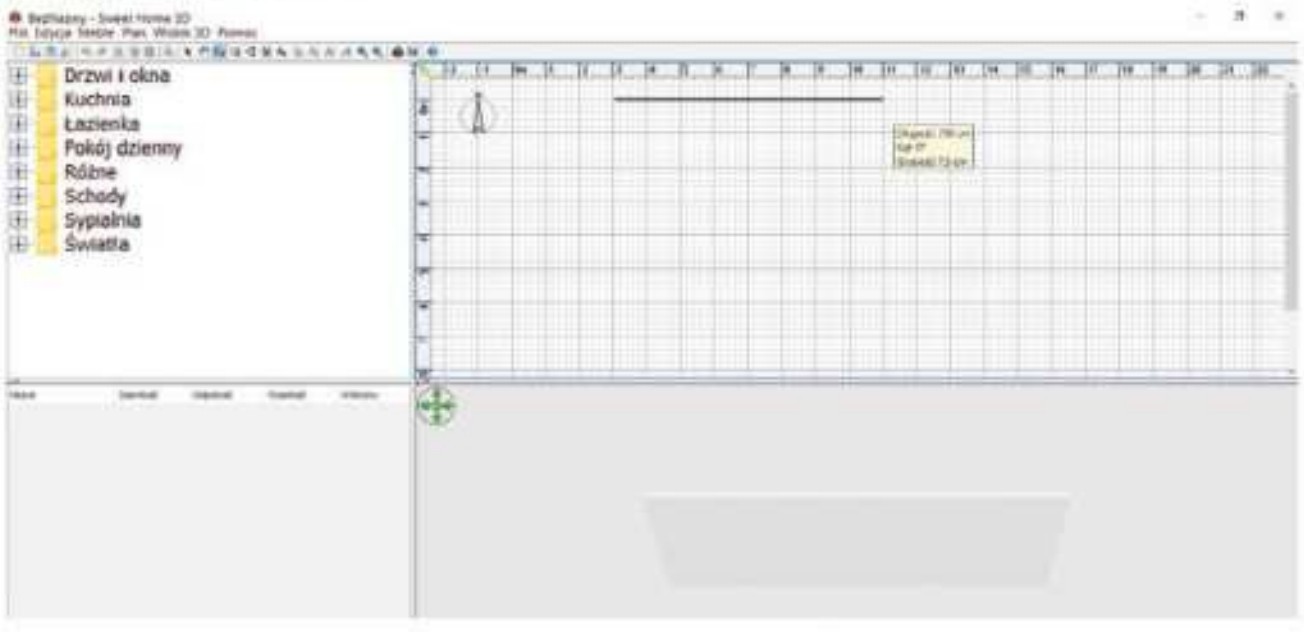
W tym rozdziale zaprojektujemy jeden pokój, umeblujemy go i stworzymy wizualizację 3D naszego projektu. Cały projekt przygotujemy w formie grafiki 2D. Zaczynamy!

**Krok 1:** Rozpoczynamy od narysowania ścian naszego pokoju. W tym celu z górnego menu wybieramy **Plan** → **Rysuj ściany**.




Rys. 43.2. Wybór możliwości narysowania ścian

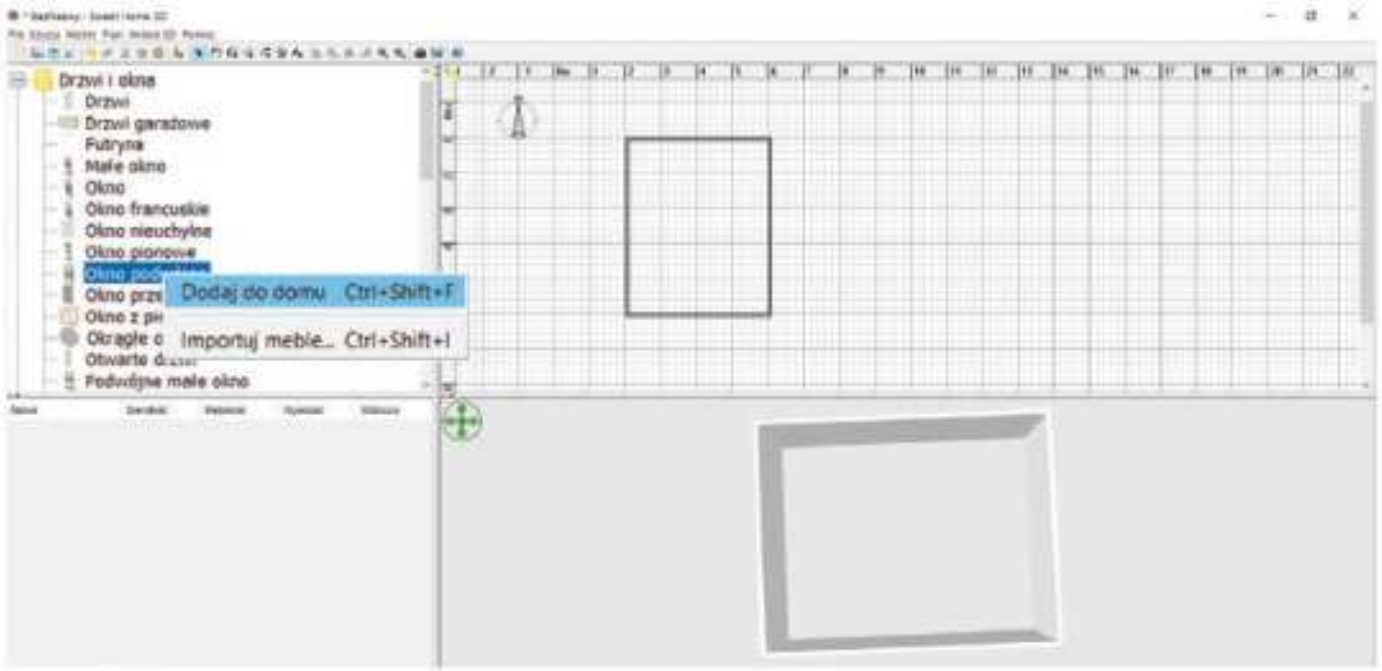
Rysując ścianę, zwróćcie uwagę, że pojawiają się informacje o długości i grubości ściany oraz kącie, pod jakim się ona znajduje. W dolnej części okna mamy podgląd 3D projektu.



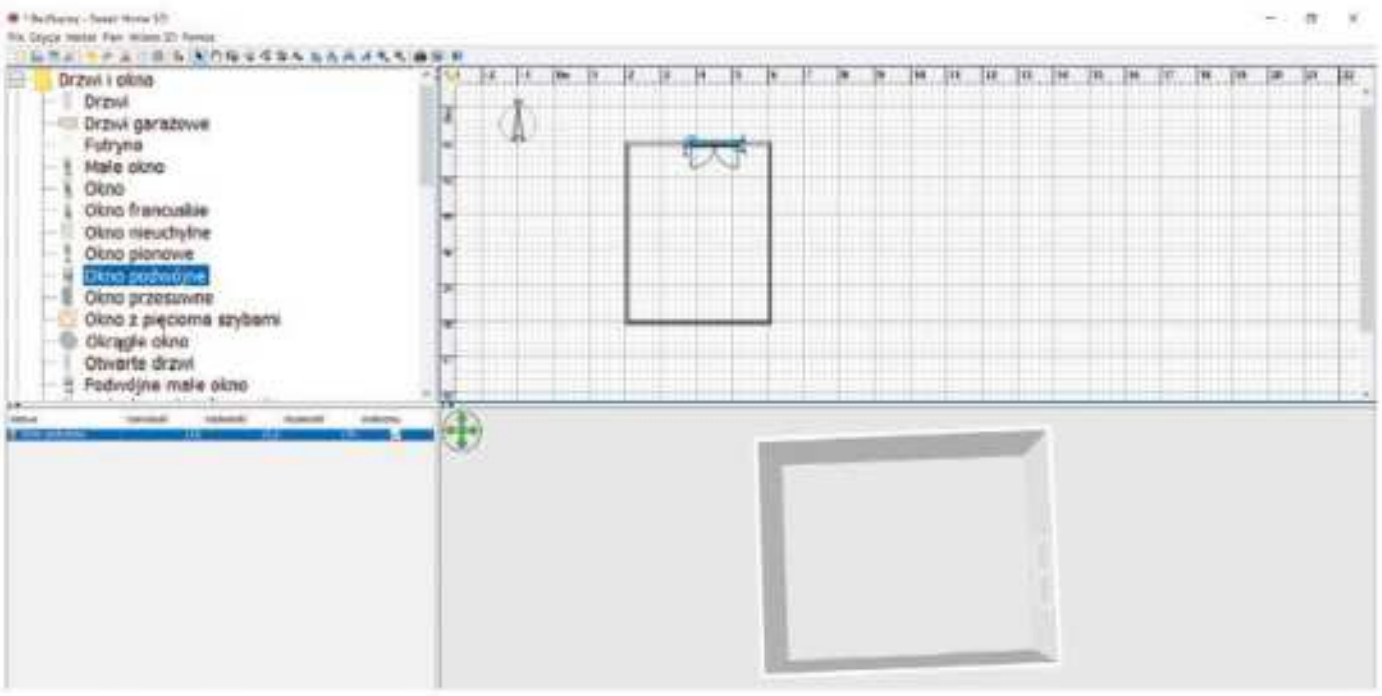

Rys. 43.3. Rysowanie ścian w programie Sweet Home 3D

**Krok 2:** Przyjmijmy, że pokój ma wymiary 400 cm x 500 cm. Na jednej z krótszych ścian umieścimy okna, a na przeciwległej – drzwi wejściowe do pokoju.

Dostęp do projektów okien i drzwi mamy w lewej górnej części okna. Po rozwinięciu drzewa **Drzwi i okna** możemy wybrać interesujące nas elementy i umieścić je na naszym projekcie. Aby dodać okno do naszego projektu, klikamy prawym przyciskiem myszy wybrany element. Wówczas wyświetli się menu rozwijane, z którego wybieramy opcję **Dodaj do domu**. Po wybraniu tego elementu pojawi się on na planie po prawej stronie. Możemy umieścić go w interesującym nas miejscu, łapiąc go i przesuwając na ścianę.



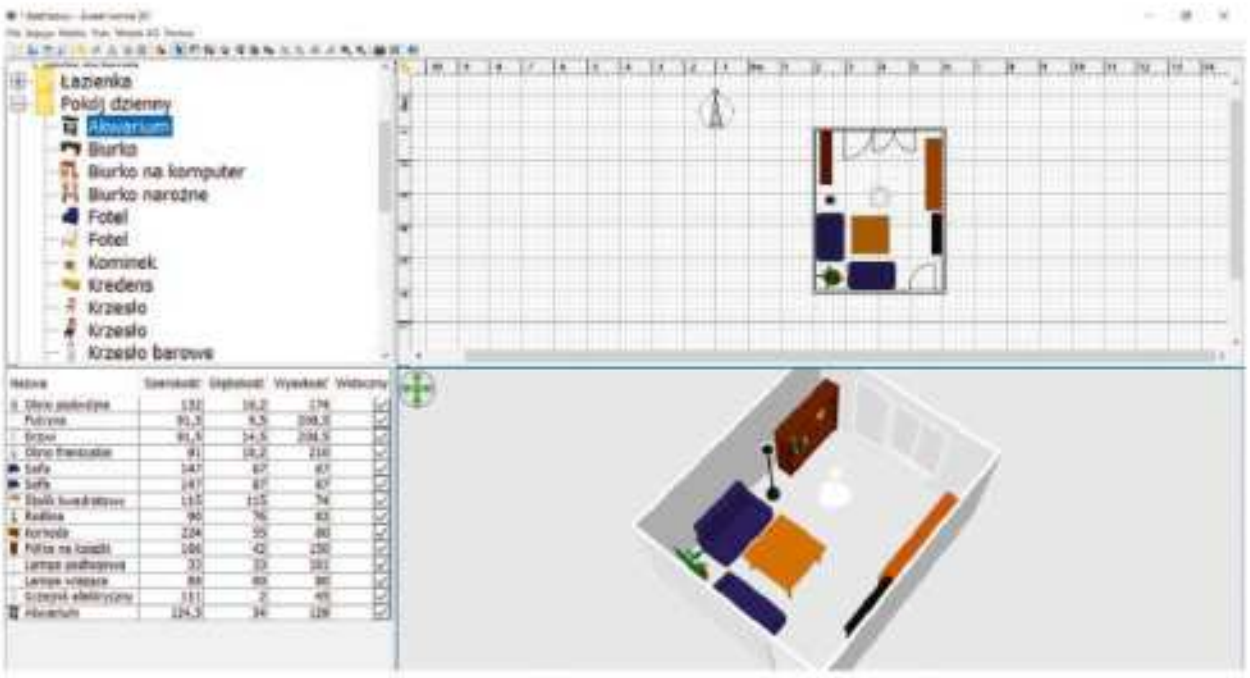

Rys. 43.4. Wybór projektowanego elementu – okna




Rys. 43.5. Umieszczenie wybranego elementu w projekcie

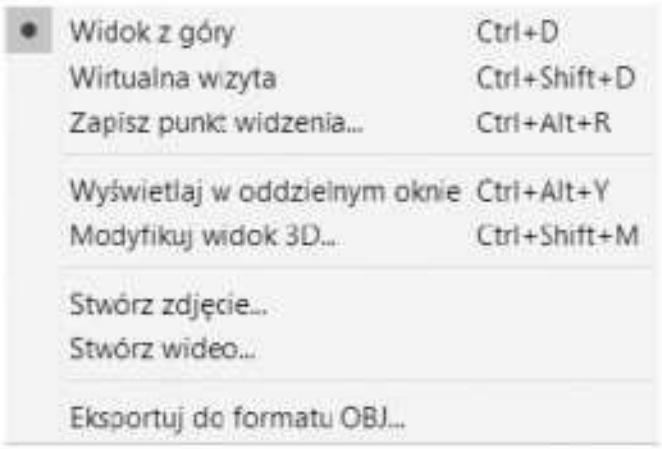
**Krok 3:** Tak kolejno postępujemy z pozostałymi elementami naszego projektu. Wstawiamy drzwi, meble (które wybieramy z opcji **Pokój dzienny** na głównym drzewie). Z opcji **Różne** na drzewie umieszczonym w górnym lewym rogu możemy wstawić inne elementy do naszego projektu, na przykład kaloryfer. Postępujemy jak w przypadku okna.

**Krok 4:** Z elementu drzewa **Światła** pobieramy oświetlenie.




Rys. 43.6. Projekt pokoju wykonany w programie Sweet Home 3D

Projektując pokój, możemy podglądać wyniki naszej pracy w dolnej części okna programu – znajduje się tam wizualizacja 3D. Możemy w tym oknie obracać ją o 360°. Jeśli to pole klikniemy prawym przyciskiem myszy, rozwinie się nam menu kontekstowe, które daje możliwości wyboru różnych opcji.




Rys. 43.7. Menu kontekstowe programu Sweet Home 3D wybrane w oknie wizualizacji 3D

## ZADANIA DO WYKONANIA

1. Przedstaw plan techniczny i wizualizację 3D swojego pokoju w programie Sweet Home 3D.
2. Za pomocą programu Sweet Home 3D zaprojektuj mieszkanie dwupokojowe z aneksem kuchennym, balkonem, łazienką i osobną toaletą.
3. Przedstaw plan techniczny i wizualizację 3D pracowni komputerowej w twojej szkole w programie Sweet Home 3D.

## PODSUMOWANIE LEKCJI

- Od lat 90. XX w. intensywnie rozwija się **grafika 3D**, która jest używana na szeroką skalę w technice i kinematografii. ◂ s. 245
- Razem z grafiką 3D zaczęło rozwijać się również komputerowe wspomaganie **projektowania w grafice 2D i 3D**. ◂ s. 245
- Program **Sweet Home 3D** jest darmowym programem służącym do projektowania wnętrz budynków. Pozwala on na projektowanie planów budynku, ustawianie mebli w gotowym już projekcie na planie 2D z podglądem 3D. ◂ s. 245

# 44. Wypukłości nie tylko 3D, czyli tworzymy elementy graficzne publikacji

**NA TEJ LEKCJI:**

- nauczysz się tworzyć wypukłe napisy w edytorze GIMP;
- dowiesz się, jak przygotować ciekawe nagłówki stron internetowych itp.

Edytory grafiki rastrowej służą nie tylko do edycji fotografii, lecz także do tworzenia ciekawych rysunków. Część ozdobnych napisów spotykanych na stronach internetowych i w innych publikacjach wygląda atrakcyjniej po dodaniu wypukłości. Oczywiście można skorzystać z gotowych rozwiązań, ale więcej satysfakcji da przygotowanie ich samodzielnie. Ponadto umiejętność tworzenia takich napisów przyda się także do przekształcania innych obiektów, na przykład przycisków, strzałek i innych elementów publikacji, takich jak strony internetowe i prezentacje.

## 44.1. 3D w GIMP, czyli tworzymy ozdobny napis

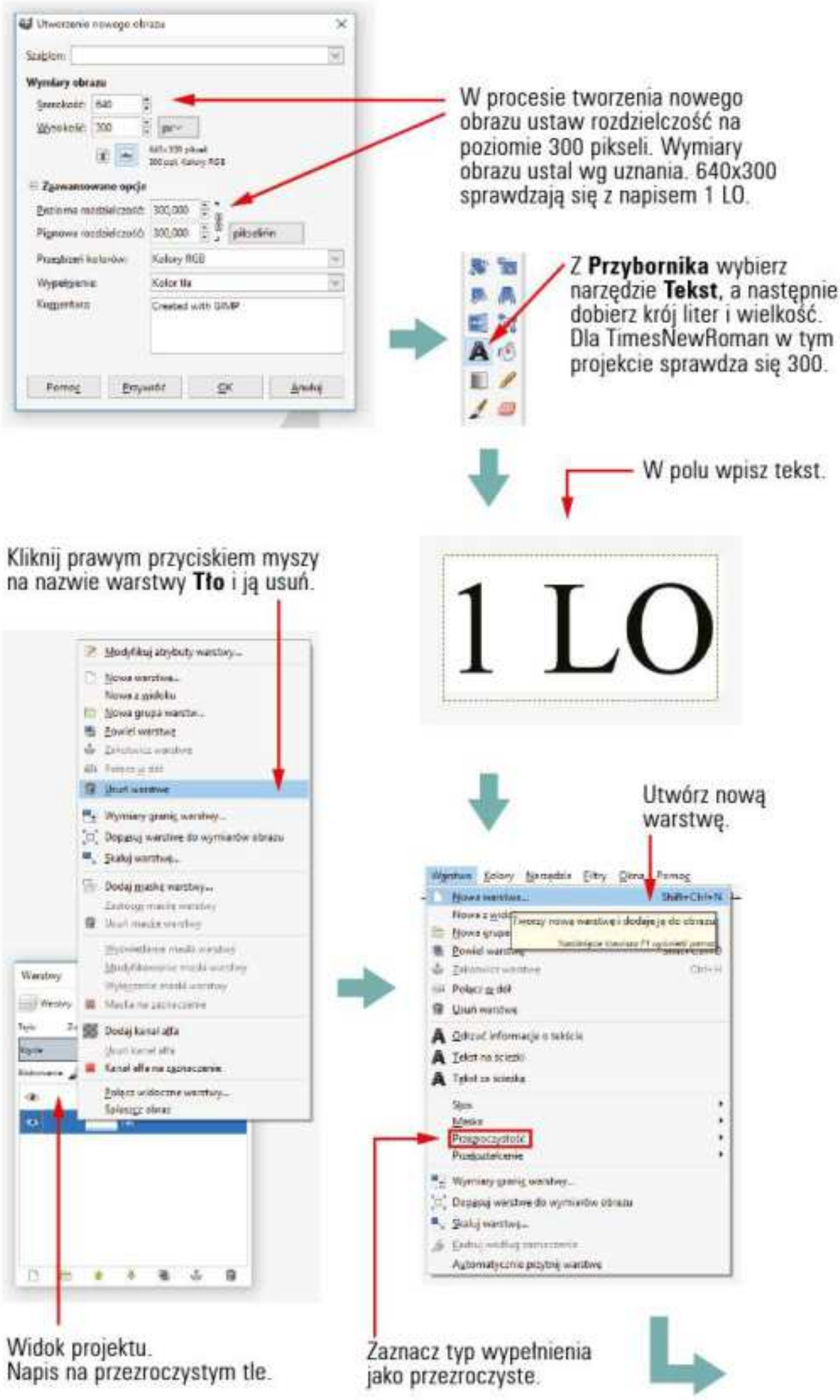
Do utworzenia wypukłego napisu użyjemy darmowego edytora grafiki rastrowej GIMP. To narzędzie możesz zainstalować w Windows, MAC OS i Linux. W niektórych dystrybucjach Linux jest ono dołączone i nie wymaga instalacji, na przykład w Ubuntu. W przykładzie zaprezentowano napis 1 LO (rys. 44.1.), ale w swoich ćwiczeniach możesz utworzyć dowolny obiekt, przykładowo logo swojej szkoły. Efekt końcowy powinien przypominać litery znane z wyszukiwarki Google. Ćwiczenie z rysunku 44.1. można zastosować do wszystkich obiektów i wykorzystać na przykład do kolorowania fotografii.

**warstwy w edytorze graficznym** ▸

Zanim przystąpisz do realizacji projektu, przypomnij sobie niektóre zasady posługiwania się edytorami grafiki wyposażonymi w system warstw (rys. 44.1.). Jeśli pamiętasz je i biegle posługujesz się GIMP-em, możesz od razu przystąpić do tworzenia napisu. Użyj po prostu opcji zmiany koloru liter.

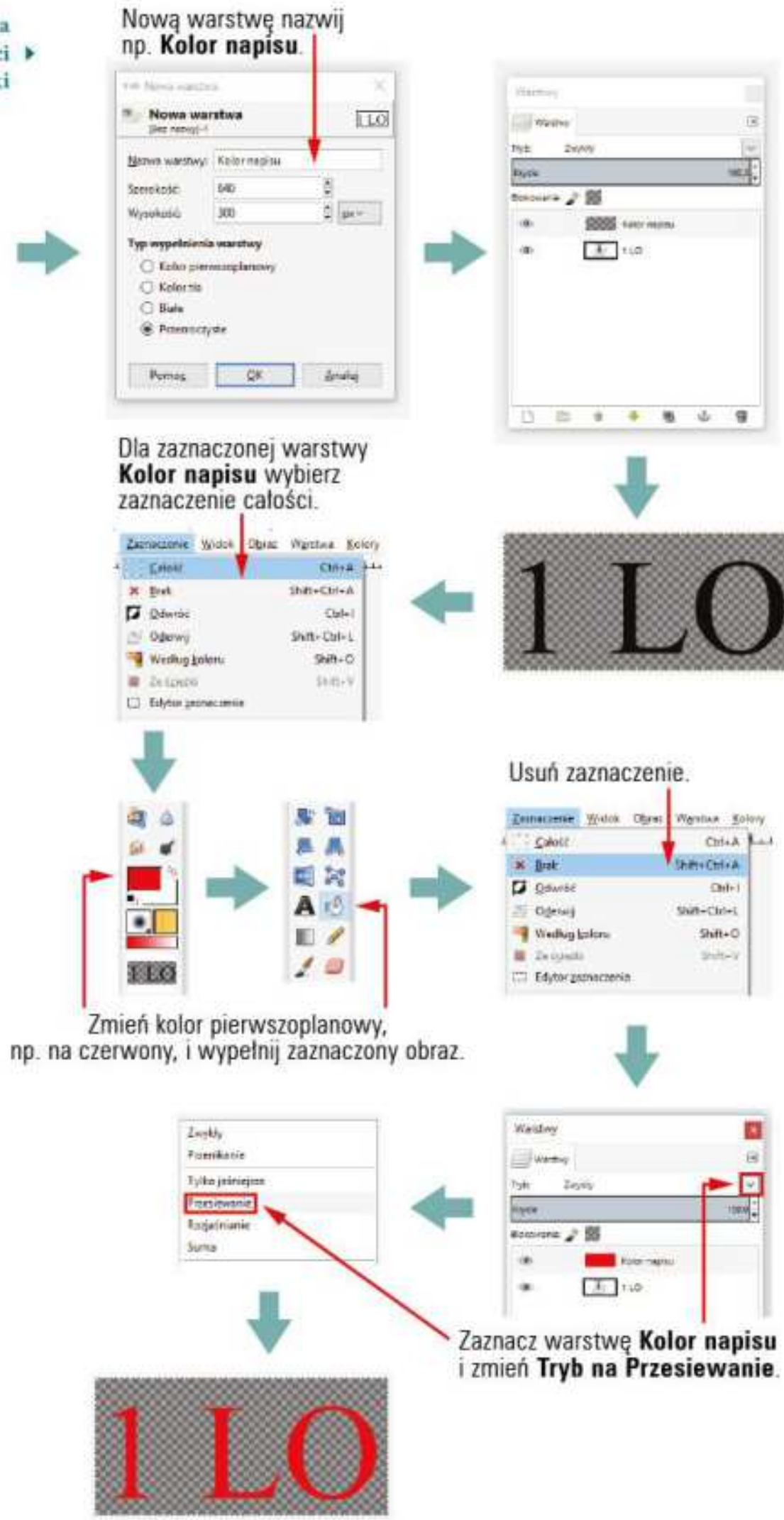
Warto zapamiętać sposób opisany na rysunku 44.1. Dzięki niemu zmienisz barwę dowolnego obiektu, na przykład ustawiając pewien poziom przezroczystości, pokolorujesz czarno-białe zdjęcia. Użyto przy tym warstw, bez których w żadnym z edytorów graficznych nie można wiele zrobić. Warstwy przypominają przezroczyste folie, na których rysuje się elementy całej kompozycji. Dzięki temu można je względem siebie przesuwać, skalować, kolorować i nakładać w różnej kolejności. Poznane w trakcie ćwiczenia zasady operowania warstwami można zastosować także w innych edytorach, na przykład w profesjonalnym PhotoShop. Teraz czas przystąpić do pracy i z płaskich liter utworzyć wypukłe.

ustawienia
rozdzielczości
i wielkości grafiki

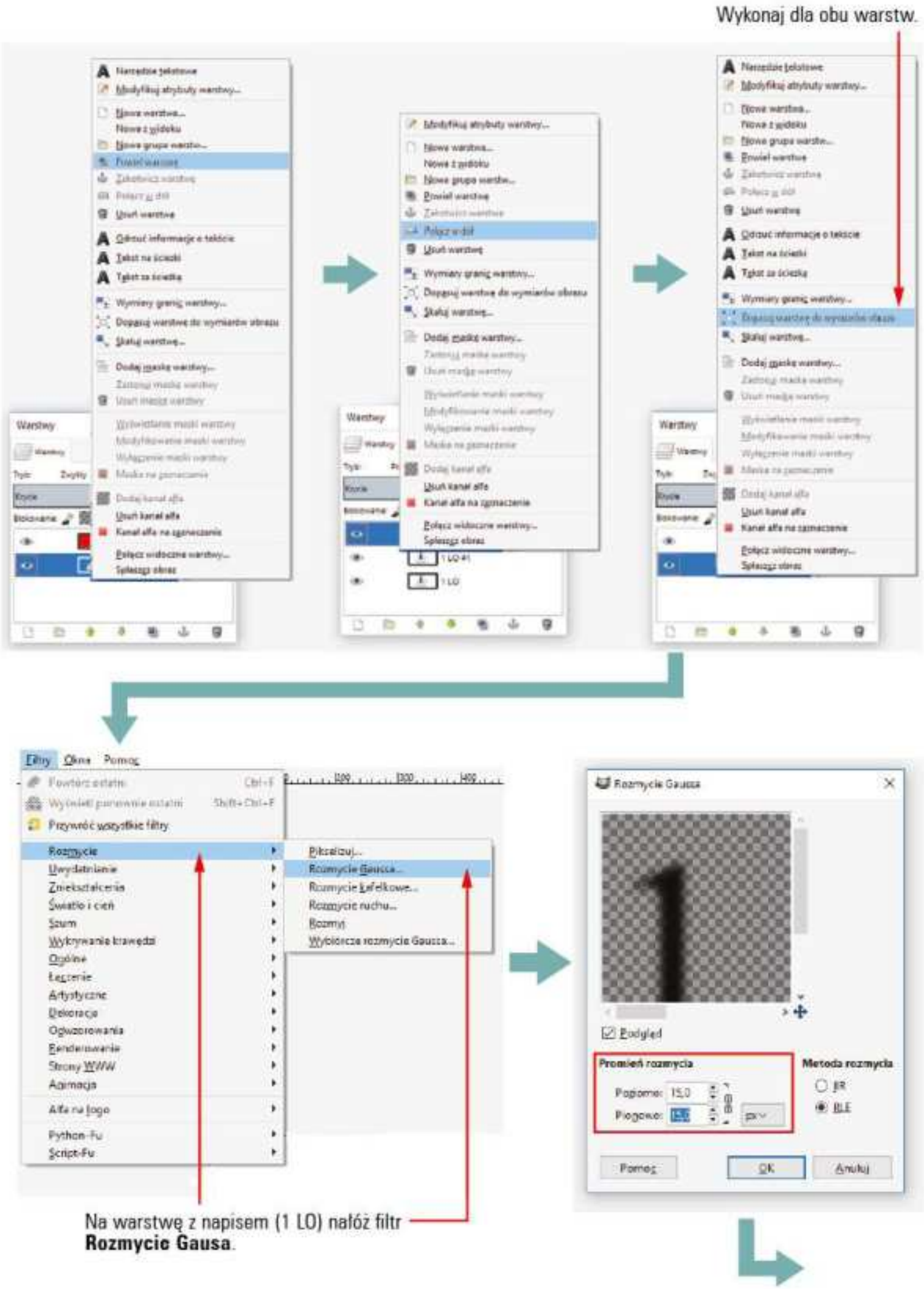



Rys. 44.1A. Proces zmiany koloru obiektu graficznego w GIMP

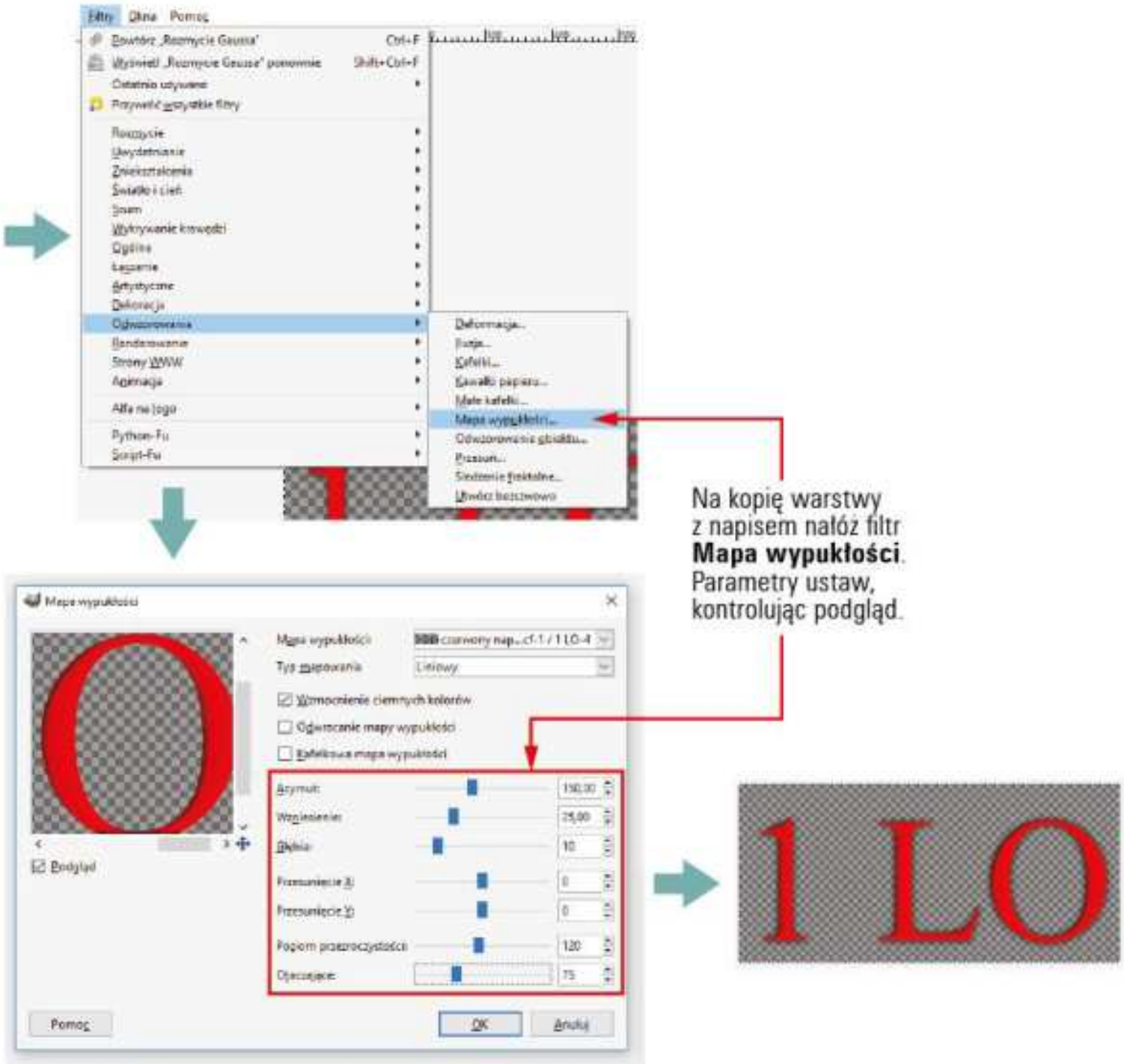
ustawienia rozdzielczości i wielkości grafiki




Rys. 44.1B. Proces zmiany koloru obiektu graficznego w GIMP




Rys. 44.2A. Proces tworzenia wypukłego napisu programem GIMP




Rys. 44.2B. Proces tworzenia wypukłego napisu programem GIMP

Ponieważ nasz napis w wyniku włączenia kanału alfa znajduje się na przezroczystym tle, należy go wyeksportować do pliku PNG lub innego typu zachowującego przezroczystość. Dlaczego? Zapisanie go w postaci JPG spowoduje, że napis znajdzie się na białym tle, co nie jest wskazane, jeśli docelowo ma on być umieszczony na danym tle (rys. 44.3.).




Rys. 44.3. Efekt wklejenia na tło plików JPG i PNG

## 44.2. Różne warstwy, czyli tworzymy nagłówek

tworzenie innych elementów

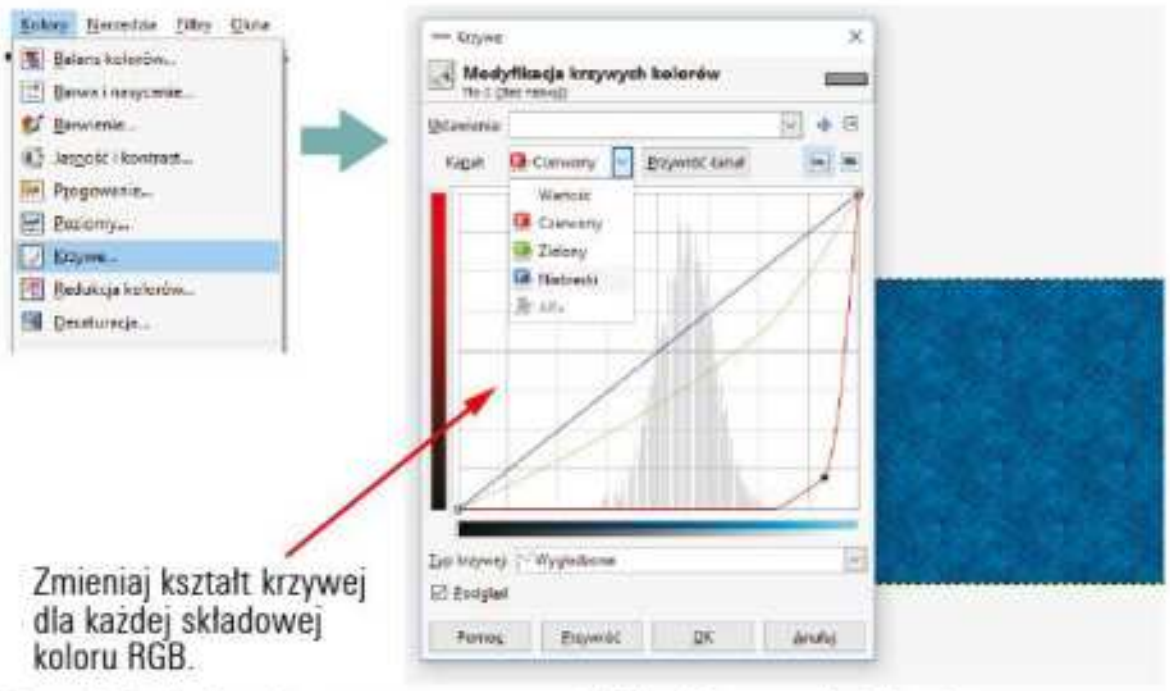
Użycie podobnych narzędzi i warstw możesz wykorzystać do utworzenia ciekawego nagłówka strony internetowej, czasopisma itp.

użycie filtrów

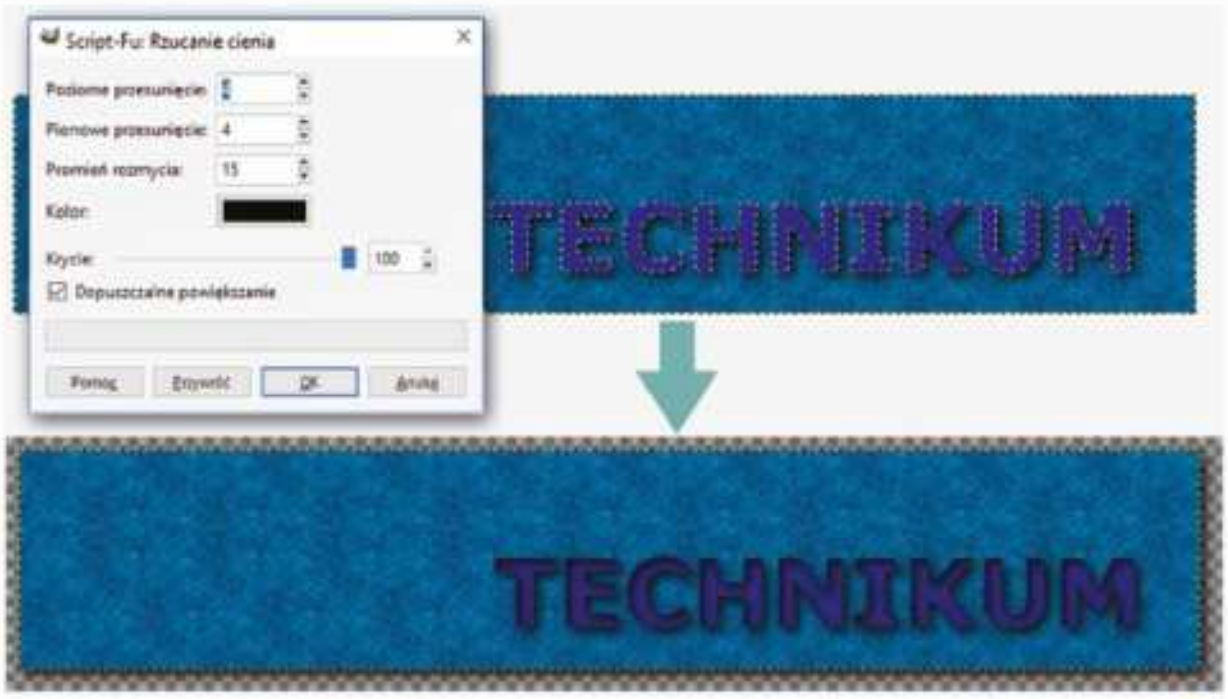
W opisie (rys. 44.4.) użyto jednej z metod zmiany kolorów rysunku polegającej na użyciu krzywych kolorów. W trakcie wykonywania ćwiczenia możesz sprawdzić, jak ich ułożenie i kształt wpływa na kolor. Nagłówek z ćwiczenia jest niebieski. Możesz uzyskać ten kolor, ustawiając poziomo na poziomie zerowym krzywe kolorów czerwonego i zielonego. Znajomość tej metody przyda ci się do korekcji barw innych rysunków i fotografii. Użyjesz także filtru **Rzucania cienia**. Jego parametry można regulować i obserwować, jaki mają wpływ na wygląd obrazu.




Rys. 44.4. Algorytm tworzenia nagłówka z napisem




Rys. 44.5. Zmiana barw za pomocą modyfikacji krzywych kolorów




Rys. 44.6. Nakładanie filtra „Rzucanie cienia" na zaznaczone litery




Rys. 44.7. Efekt włączania i wyłączania warstw w jednym projekcie

Pierwszy z przykładów z rysunku 44.7. może zostać użyty jako znak wodny (rys. 44.8.). Należy go wyeksportować do pliku z zachowaniem kanału alfa.




Rys. 44.8. Znak wodny wykonany w programie GIMP

## ZADANIA DO WYKONANIA

1. Przeanalizuj jeszcze raz ćwiczenie z rozdziału i wytlumacz, dlaczego wyłączanie odpowiednich warstw powoduje dane zmiany.
2. a) Wzorując się na ćwiczeniu z rozdziału, wykonaj napis „Tęcza". Zastosuj w warstwie **Kolor napisu** wszystkie barwy tęczy ułożone według kolejności. W ten sposób twój wypukły napis będzie wielokolorowy.

   b) Wykonaj ćwiczenie 2.a, stosując płynne przejścia pomiędzy kolorami tęczy. Możesz zastosować zaznaczanie ze zmiękczaniem i zaokrąglaniem krawędzi.
3. Sprawdź, jak będzie wyglądał twój napis, gdy ćwiczenie wykonasz z pominięciem warstwy **Kolor napisu**, a barwę liter uzyskasz bezpośrednio poprzez wybór koloru napisu. Opisz różnice i przyczyny ich powstania.
4. Do napisu z ćwiczenia opisanego w rozdziale dodaj cień. Użyj odpowiedniego filtra. Gotowy napis wraz z cieniem wyeksportuj do odpowiedniego formatu zachowującego przezroczystość.

5*. Do napisu z ćwiczenia opisanego w rozdziale dodaj cień. Użyj odpowiedniego filtra. Przekształć warstwę z cieniem tak, by całość sprawiała wrażenie stojącego, oświetlonego od przodu napisu. Gotowy rysunek wraz z cieniem wyeksportuj do odpowiedniego formatu zachowującego przezroczystość.

6. Wykorzystaj doświadczenia z ćwiczenia z wklęsłym tekstem i wykonaj znak wodny do trwałego oznaczania twoich zdjęć. Może on zawierać twoje nazwisko lub inicjały.

## PODSUMOWANIE LEKCJI

s. 250 ▸ • W trakcie wykonywania ćwiczeń przypomniałeś/przypomniałaś sobie, jak wykorzystać **warstwy w edytorze graficznym** do tworzenia ciekawych efektów graficznych. Umiejętność operowania na warstwach pozwoli ci sprawnie używać innych edytorów grafiki i fotografii.

s. 255 ▸ • Zdobyte i utrwalone umiejętności możesz wykorzystać **do tworzenia elementów prezentacji, stron internetowych i dokumentów tekstowych**. Dzięki możliwości samodzielnego ustawienia **rozdzielczości i wielkości grafiki** możesz także tworzyć projekty drukowane na dużych formatach, np. ściankach reklamowych.

s. 255 ▸ • **Użycie filtrów** wymagało zmiany ich parametrów w oknach dialogowych. Mogłeś/mogłaś zbadać, jak wpływają one na efekt graficzny. Warto eksperymentować z filtrami i warstwami programów graficznych. Można uzyskać zaskakujące efekty. Ten sam projekt bez niektórych warstw może wyglądać zupełnie inaczej.

# 45. Dynamicznie, czyli jak tworzyć ruchome wizualizacje

**NA TEJ LEKCJI:**

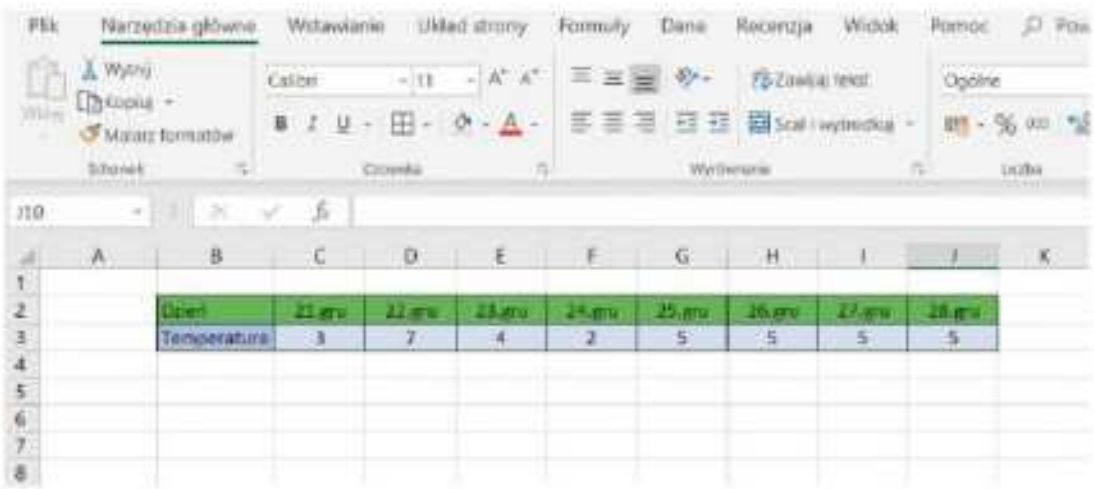
- dowiesz się, czym jest wizualizacja i jaki jest jej cel;
- dowiesz się, czym jest animacja i gdzie jest wykorzystywana.

Za pomocą środków wizualnych obecne społeczeństwo wymienia się zarówno ideami abstrakcyjnymi, jak i komunikatami o wartości rzeczywistej. Można śmiało powiedzieć, że bez wizualizacji badania naukowe byłyby bardzo ograniczone. Wizualizacja jest wykorzystywana w dyscyplinach technicznych czy medycynie. W szkole na większości zajęć lekcyjnych nauczyciele korzystają ze środków wizualnych, aby przybliżyć uczniom jakiś problem czy zagadnienie. Wizualizacja stała się również jednym z ważniejszych środków wyrazu artystycznego.

## 45.1. Wizualizacja danych

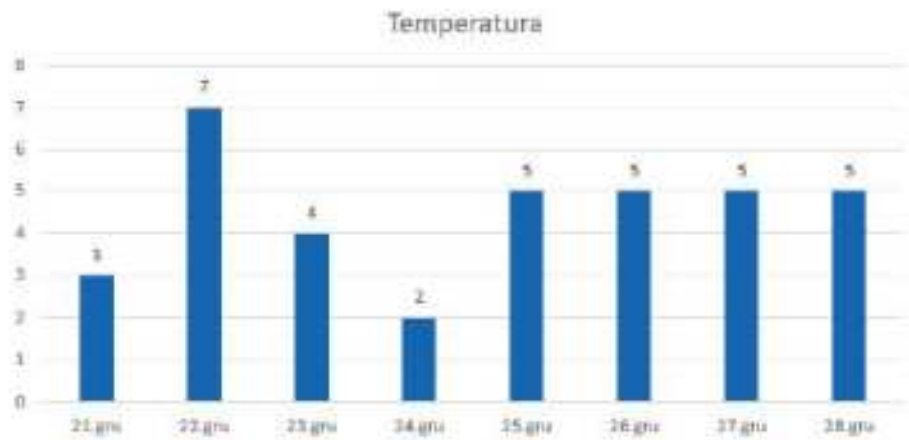
Przeanalizujmy prognozę pogody na kilka następnych dni, otrzymaną z pewnego serwisu pogodowego.

◂ wizualizacja danych




Rys. 45.1. Dane o przewidywanej temperaturze w Bydgoszczy

Na podstawie tych danych sporządzimy wizualizację tego, jak może zmienić się temperatura w Bydgoszczy w ciągu kolejnych 7 dni. Potrzebny będzie do tego arkusz kalkulacyjny. Dane zawarte w tabeli zostaną przedstawione za pomocą wykresu kolumnowego.




Rys. 45.2. Wizualizacja temperatury przewidywanej w ciągu 7 dni w Bydgoszczy

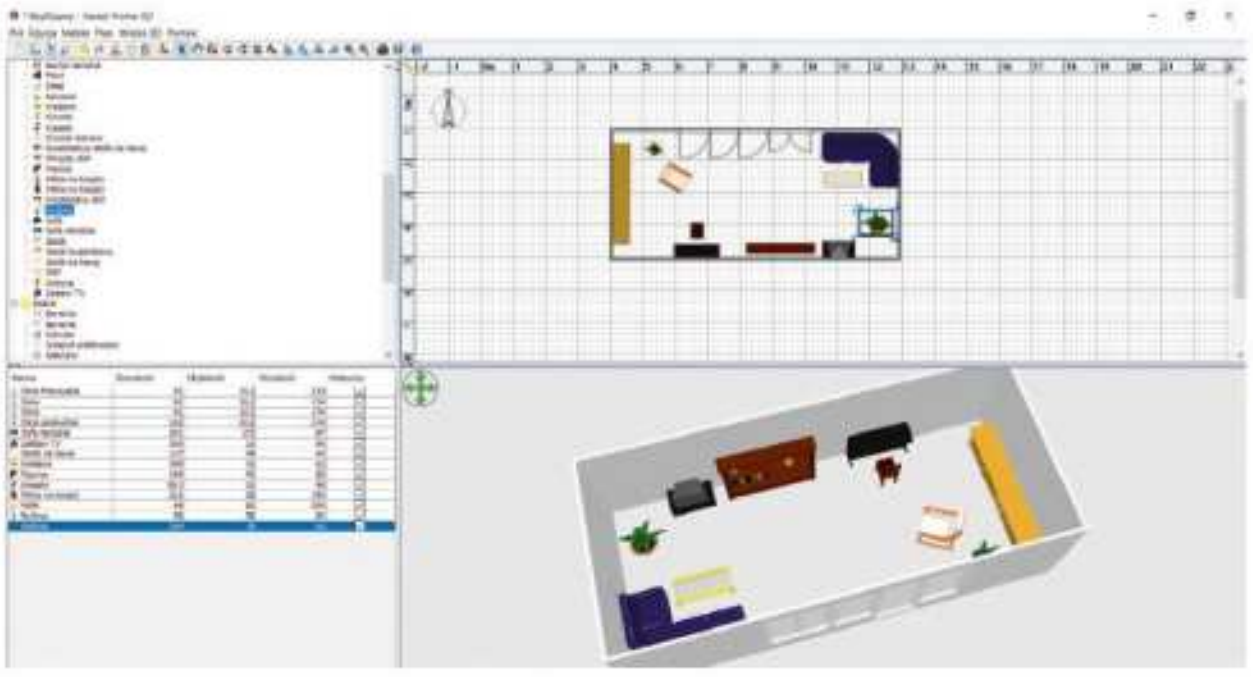
Wizualizacja danych to graficzne przedstawienie informacji, które zostały zestawione w pewnej schematycznej formie. Głównym celem wizualizacji jest skuteczny i zrozumiały przekaz zawartych w niej treści. Przedstawienie danych w odpowiedni sposób pozwala na poprawne i szybkie zrozumienie zależności opisanych przez prezentowane dane.

## 45.2. Rysunek techniczny

**rysunek techniczny** ▸ Rysunek techniczny jest wizualizacją konwencjonalnie przedstawiającą plany, konstrukcje lub zasady działania przedmiotów mechanicznych itp. Osoby, które tworzą tego typu wizualizacje, są zobowiązane do przestrzegania szeregu norm regulujących stosowne oznaczenia, rodzaje perspektywy czy sposoby pomiarów.

W przeszłości rysunki techniczne były kreślone ręcznie, obecnie wykorzystuje się oprogramowanie do komputerowego wspomagania projektowania, tzw. programy CAD. Oprócz grafiki dwuwymiarowej programy te mogą również prezentować wizualizacje 3D.

Przykładem takiego programu jest Sweet Home 3D, z którego korzystaliśmy już wcześniej.



Rys. 45.3. Wizualizacja 2D i 3D wykonana w programie Sweet Home 3D

## 45.3. Animacja komputerowa

**animacja komputerowa** ▸ Animacja to technika, która wykorzystuje ruchome obrazy w celu przedstawienia pewnego procesu w określonym czasie. Wykorzystując trójwymiarowość w animacji, jednocześnie można przedstawić wiele aspektów danego zagadnienia w sposób dynamiczny. Najczęściej animacja pojawia się w produkcjach filmowych.

## 45.4. Wizualizacja w informatyce

etapy tworzenia wizualizacji

Proces tworzenia wizualizacji w informatyce składa się z czterech etapów. Pierwszy z nich to **modelowanie**, czyli żmudny proces tworzenia obiektów 3D. Tworzy się je za pomocą przekształceń figur geometrycznych. Każda aplikacja czy program, dzięki któremu możemy takie obiekty wykonywać, ma swoje specyficzne narzędzia.

Po wykonaniu modelu 3D należy nadać mu tzw. **teksturę**, czyli wypełnić go tłem, aby był jak najbardziej fotorealistyczny. Dla każdego obiektu z osobna definiuje się odpowiednie parametry rozproszenia, odbicia światła, refleksów, wygładzeń i innych ustawień, które w końcowym efekcie nadadzą mu realistyczny wygląd. Po nadaniu tekstury w programie graficznym, w którym tworzymy naszą wizualizację, ustawiamy scenę. Odpowiednio dobieramy światło i kamery. Musimy pamiętać, że każda aplikacja do obróbki i tworzenia grafiki 3D imituje światło różnego rodzaju lamp, więc na tym etapie bardzo przydatna może okazać się wiedza z zakresu fotografii.

Ostatnim etapem jest renderowanie sceny. Polega ono na przeliczeniu przez komputer wszystkich ustawień, użytych źródeł światła, tekstur, odbić itp., a następnie utworzenie płaskiego rastrowego obrazu 2D. Długość procesu renderowania zależy od wydajności sprzętu, na którym powstaje grafika 3D, a także od złożoności tej grafiki. Ten etap może trwać od kilku minut do kilkudziesięciu godzin.

Najpopularniejszymi programami wykorzystywanymi do tworzenia grafik 3D, a następnie ich animowania, są:

- 3ds Max,
- Blender,
- Cinema 4D,
- Imagine 3D.

## ZADANIA DO WYKONANIA

1. Wykonaj animację spadającego śniegu w programie do grafiki rastrowej GIMP.
2. Wykorzystując program Blender, narysuj wizualizację krzesła, które będzie się obracało wokół osi pionowej.

## PODSUMOWANIE LEKCJI

- **Wizualizacja danych** to graficzne przedstawienie informacji, które zostały zestawione w pewnej schematycznej formie. s. 259
- **Rysunek techniczny** jest wizualizacją konwencjonalnie przedstawiającą plany, konstrukcje lub zasady działania przedmiotów mechanicznych itp. s. 260
- **Animacja** to technika, która wykorzystuje ruchome obrazy w celu przedstawienia pewnego procesu w określonym czasie. s. 260
- Proces tworzenia wizualizacji w informatyce składa się z **czterech etapów**: modelowania, teksturowania, ustawiania sceny i renderowania. s. 261

# 46. Z wydruku do komputera, czyli skanujemy i odczytujemy dokumenty

**NA TEJ LEKCJI:**

- dowiesz się, do czego służy technologia OCR;
- poznasz działanie skanera.

Komputerowa analiza obrazu jest wykorzystywana w wielu urządzeniach. Można nakładać w czasie rzeczywistym na obraz z kamery elementy graficzne, wskazania GPS itp. Ponadto dzięki analizie obrazu tekst widniejący na fotografii da się zmienić na plik edytowalny w edytorze. W jakim celu? Czy to łatwe do wykonania? Sprawdź.

## 46.1. Skąd wziąć kopie, czyli skanujemy dokumenty

Analiza pisma opiera się na przetwarzaniu obrazu (rys. 46.1.). Może on pochodzić z pliku lub kamery (także z aparatu fotograficznego). Analizowanie pisma odręcznego jest skomplikowane i wymaga zazwyczaj próbek oryginalnego charakteru pisma danej osoby.

Szybciej i łatwiej analizować zdjęcie druku. W tym celu opracowano technologię
OCR ▸ OCR (Optical Character Recognition, czytaj: optikal karakter rekognyszyn). Dla uproszczenia programy, które się nią posługują, będziemy nazywać tym skrótem. Do czego może się przydać program OCR?




Rys. 46.1. Kolejność czynności podczas pozyskiwania wersji elektronicznej dokumentu

**Przykład 1.**

Student pisze pracę, na przykład opis projektu na zaliczenie przedmiotu. Na dzień przed terminem oddania uszkodził dysk komputera. Nie robił kopii zapasowych

ani nie pracował w chmurze. Miał jednak wydrukowaną jedną z wersji pracy. Użycie OCR i skanera pozwoliło mu na odzyskanie jej w postaci elektronicznej.

**Przykład 2.**

Osoba przygotowująca kalkulację kosztów projektu wydrukowała tabele z programu MS Excel i dołączyła do dokumentacji. Zgodnie z procedurą musi przesłać kopie w postaci elektronicznej, jednak nie ma dostępu do serwera. Skanowanie tabel i zapisanie plików wynikowych działania OCR pozwoli zyskać wersję elektroniczną kalkulacji (oczywiście formuł tabel nie da się w ten sposób skopiować).

**Przykład 3.**

Dziennikarz szkolnej gazetki internetowej pisze artykuł na pewien temat. Musi posłużyć się cytatami z książek i czasopism. Przepisywanie zabiera czas i może być źródłem błędów. Użycie OCR wyeliminuje te niedogodności.

**Przykład 4.**

Uczeń ma dokument w formacie PDF. Jednak nie może otworzyć go w licencjonowanym programie do jego edycji. OCR zamieni potrzebne fragmenty dokumentu na pliki tekstowe.

Podobnych przykładów można podać więcej i nigdy nie wiadomo, do czego jeszcze technologia OCR może się przydać.

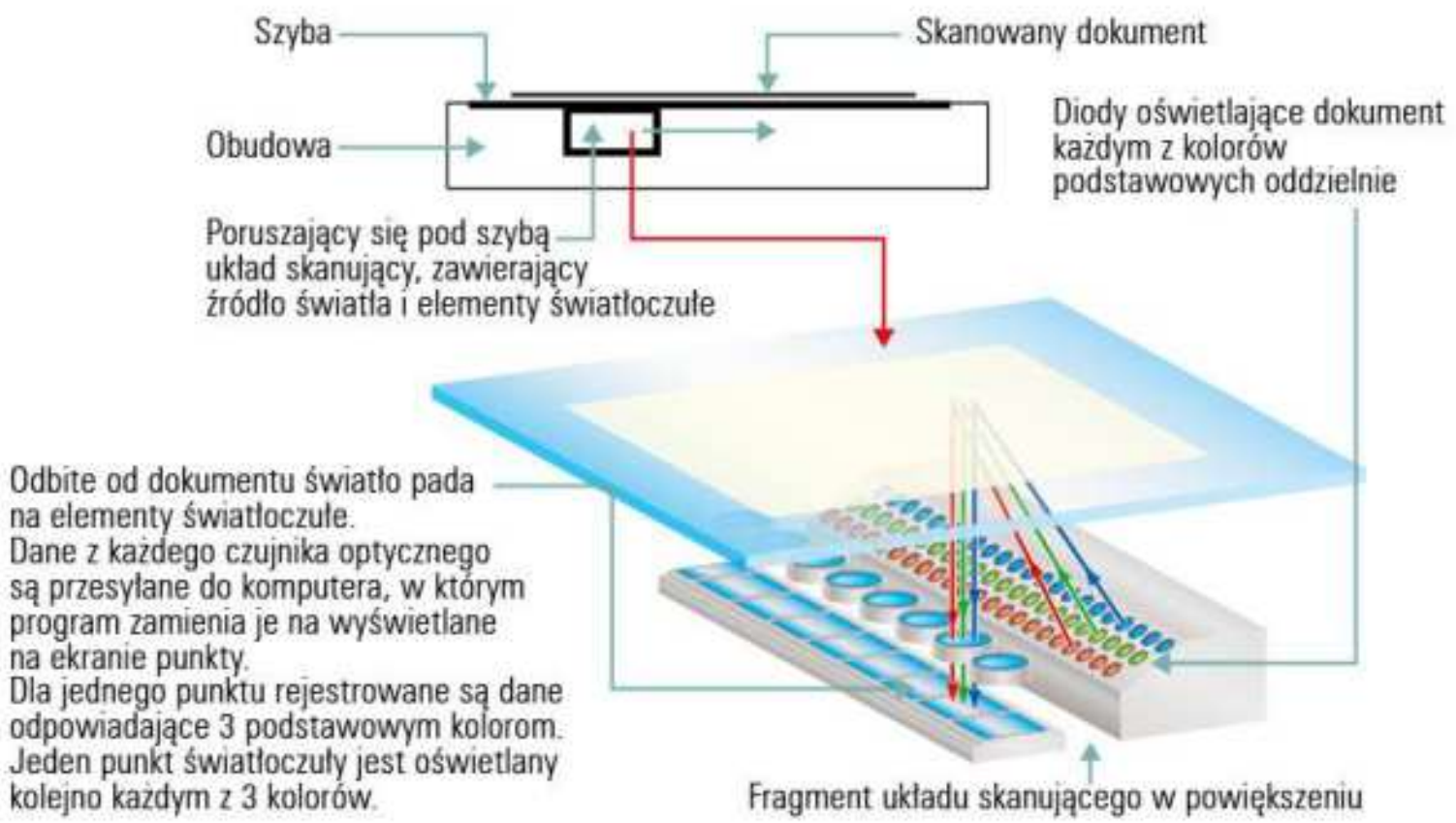
Co musi wiedzieć użytkownik programu OCR? W opisie można znaleźć listę formatów plików, które dany program przetwarza, na przykład JPG, BMP, TIFF, GIF. Pliki graficzne zawierające tekst trzeba pozyskać poprzez skanowanie, robienie zrzutów ekranu, zdjęć itp.




Rys. 46.2. Źródła plików dla programów OCR

**działanie skanera** ▸ Czy wiesz jak działa skaner? Powszechnie są stosowane dwie konstrukcje.

**CCD i CIS** ▸ CIS, w której elementy światłoczułe i źródła światła (RGB) przesuwają się tuż przy skanowanej powierzchni, oraz CCD, w której obraz skanowanego dokumentu (oświetlany białym światłem) dociera poprzez układ luster i soczewek do matrycy podobnej w działaniu do matrycy aparatu fotograficznego (rys. 46.3.).

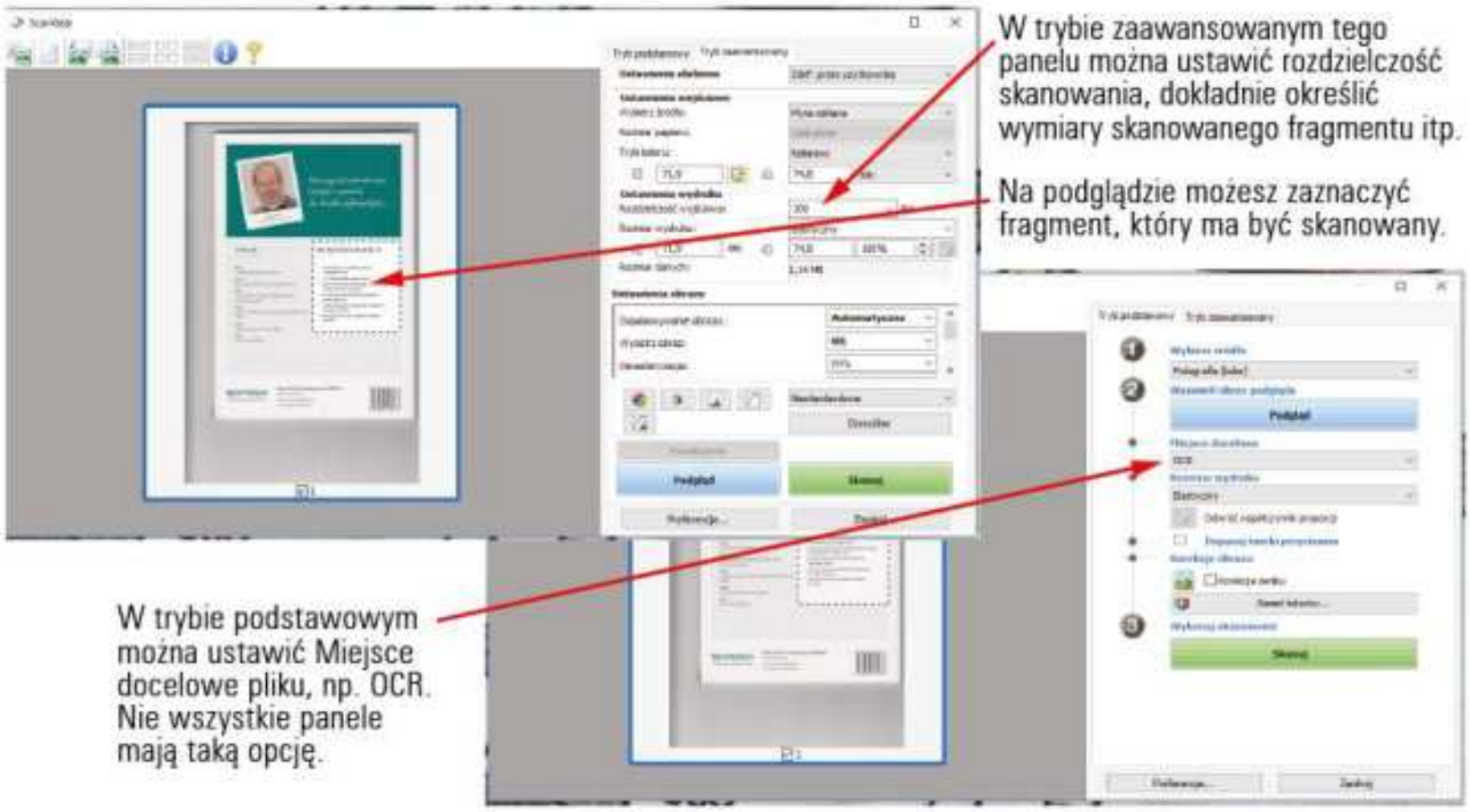



Rys. 46.3. Matryca CIS (u góry) i CCD (na dole) skanerów stołowych

Programy OCR działają skutecznie dla dokumentów zeskanowanych z określoną rozdzielczością. Przyjęło się, że jeśli litery dobrze kontrastują z tłem, to rozdzielczością zapewniającą skuteczność OCR jest minimum 200 dpi (punktów na cal).

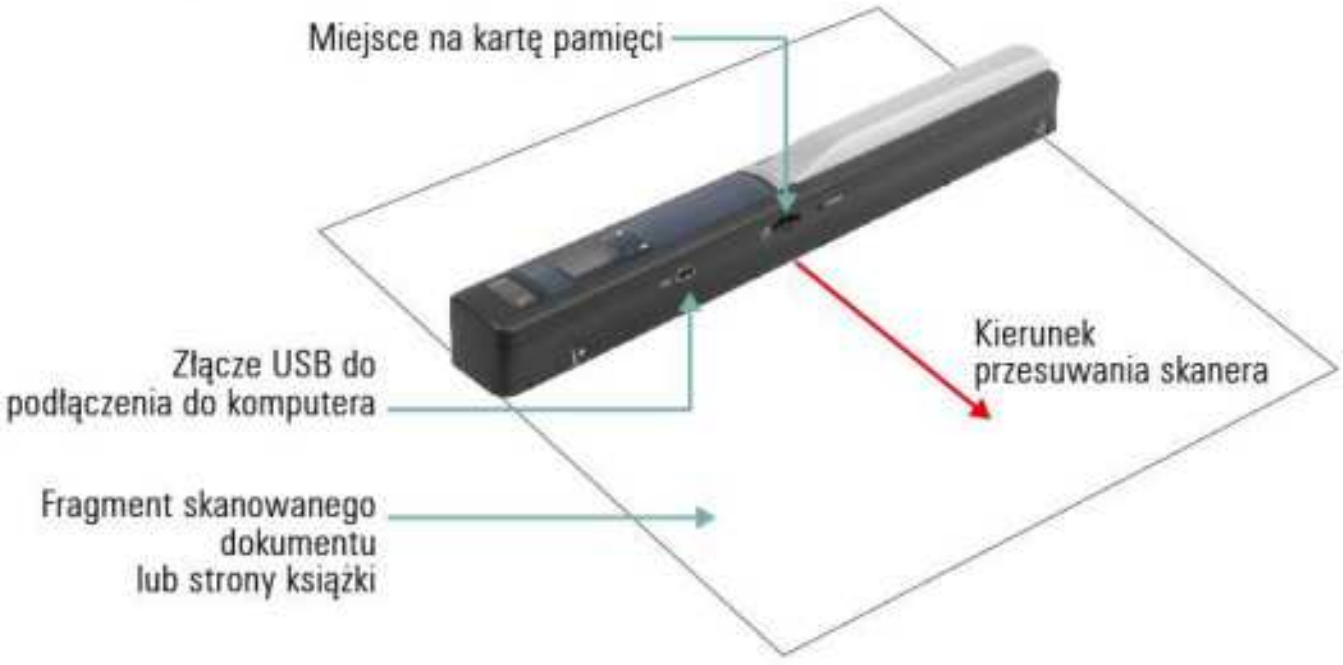
**TWAIN** ▸ Przebieg procesu skanowania zależy od użytego programu. Mimo iż sterowniki skanerów dla systemów operacyjnych używają tego samego standardu komunikacji (protokołu transmisji) TWAIN, panele użytkownika mogą znacznie się różnić. Panel uruchomi się wraz z zainicjowaniem skanowania (rys. 46.4.).

Instaluje się wraz ze sterownikami i stanowi nieocenione narzędzie pozwalające na wykorzystanie wszystkich zalet danego modelu skanera. W zastępstwie skanera można użyć aparatu fotograficznego. Doświadczalnie należy dobrać ustawienia, by tekst był czytelny i kontrastował z tłem.




Rys. 46.4. Przykładowy panel sterujący z ustawieniami parametrów skanowania

Użytkownicy OCR odwiedzający biblioteki czy archiwa zazwyczaj nie mogą korzystać ze skanerów stołowych. Z pomocą przychodzą urządzenia mobilne. Ręczne skanery podobne do długopisów pozwalają na skanowanie wybranych obszarów dokumentów, przechowywanie skanów w pamięci lub przesyłanie do komputera. Analiza i konwersja najczęściej odbywają się w komputerze PC (rys. 46.5.). Zauważ, że w zastosowaniach OCR nie ma znaczenia rodzaj technologii użytej do budowy skanera.




Rys. 46.5. Przykłady skanerów ręcznych do zastosowań OCR

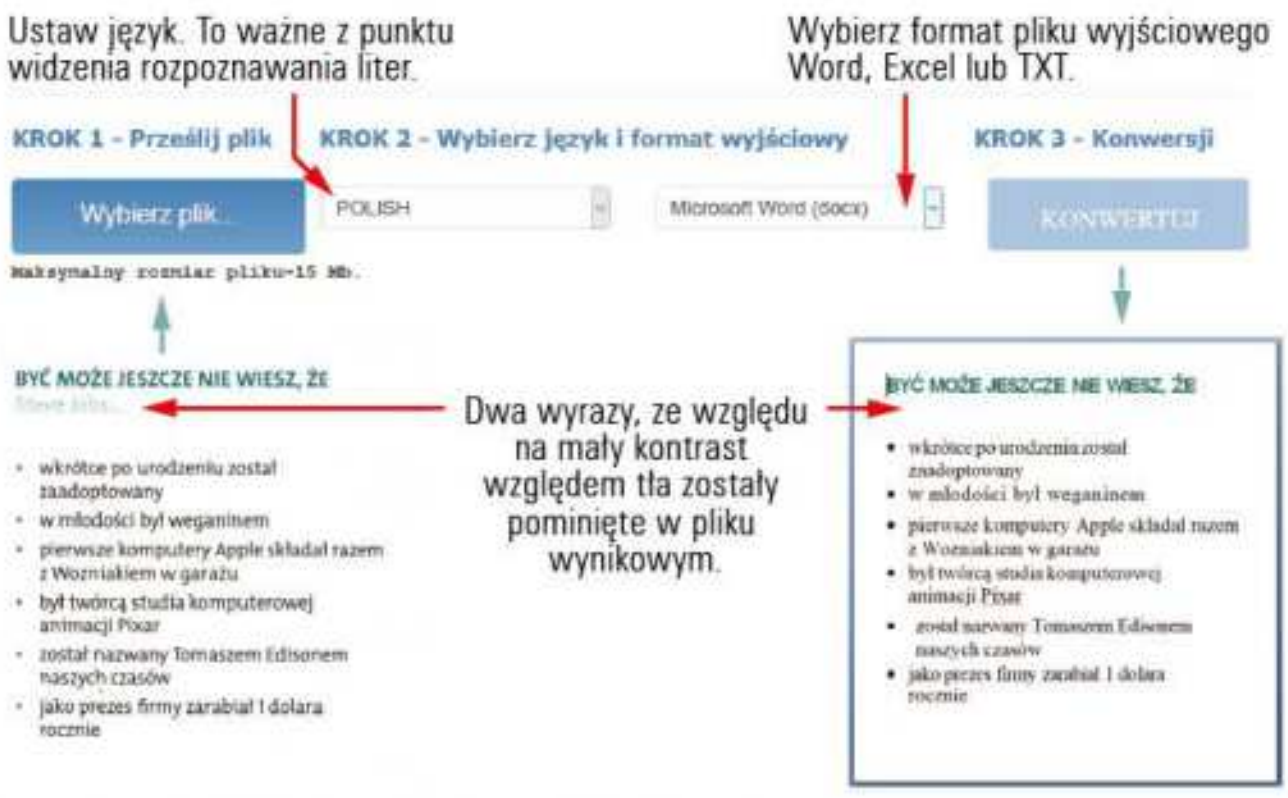
## 46.2. Ostatnia deska ratunku, czyli OCR odzyskuje dokumenty

Przebieg procesu OCR przeanalizujmy na przykładzie programu działającego on-line. Spośród kilku dostępnych w sieci możesz wybrać *onlineocr.net*. Pod tym samym adresem znajduje się także jego komercyjna wersja. Bez założenia konta możesz poddawać procesowi do 15 stron na godzinę. Program nie ma wielu opcji, a proces przebiega automatycznie (rys. 46.6.).

konwersja w programie OCR

W tej wersji do konwersji możesz przesłać pliki w formacie **PDF**, **JPG**, **BMP**, **TIFF**, **GIF**, a formatami wyjściowymi mogą być **Word**, **Excel** lub plik tekstowy **TXT**.

Konwersja nastąpi po wciśnięciu przycisku **KONWERTUJ**.




Rys. 46.6. Konwersja OCR w programie onlineocr.net on-line

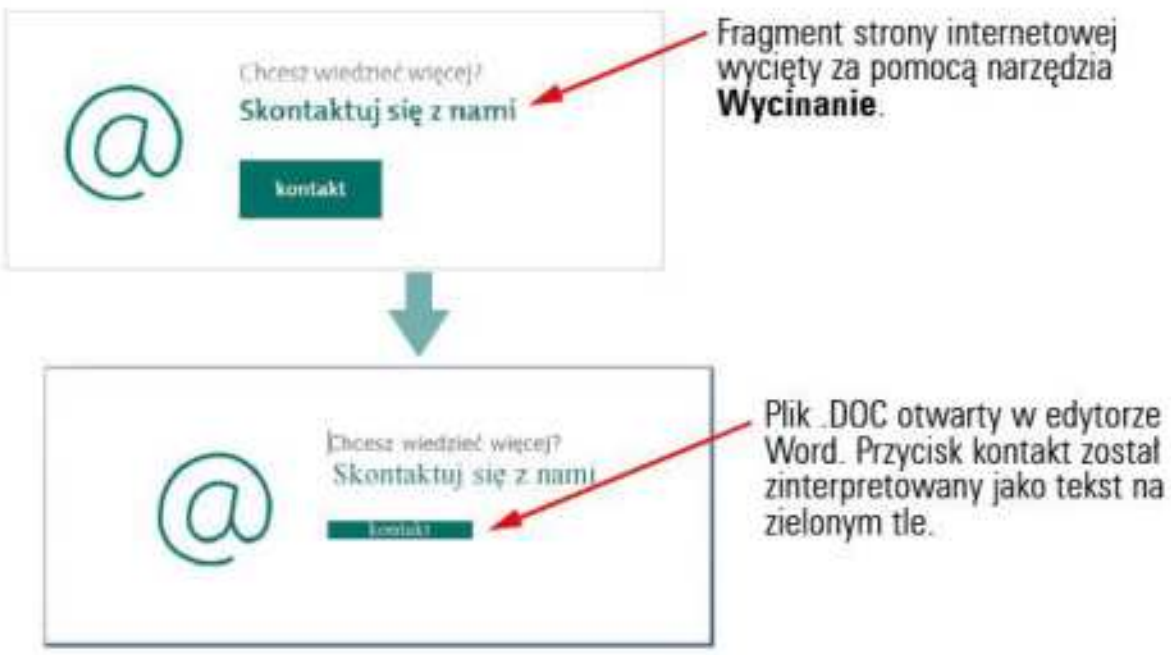
Pobieranie pliku wynikowego zainicjuj, klikając jego nazwę (poniżej przycisku **Wybierz plik...**).

**Uwaga!**

Po konwersji programem OCR zawsze sprawdzaj poprawność procesu. Może się zdarzyć, że litery zostaną źle rozpoznane, a wyraz zupełnie zmieni znaczenie.

Dokumenty często zawierają ilustracje. Czy OCR potrafi je rozpoznać? Wypróbuj to na przykładzie dowolnej strony internetowej. Użyj narzędzia **Wycinanie** (wbudowane w Windows) i zaznacz dany fragment okna przeglądarki. Zapisz obraz w jednym z obsługiwanych formatów. W przykładzie wykorzystano fragment, w którym mamy do czynienia z dwoma elementami graficznymi, w tym przyciskiem z napisem. W automatycznych OCR program może rozpoznać taki

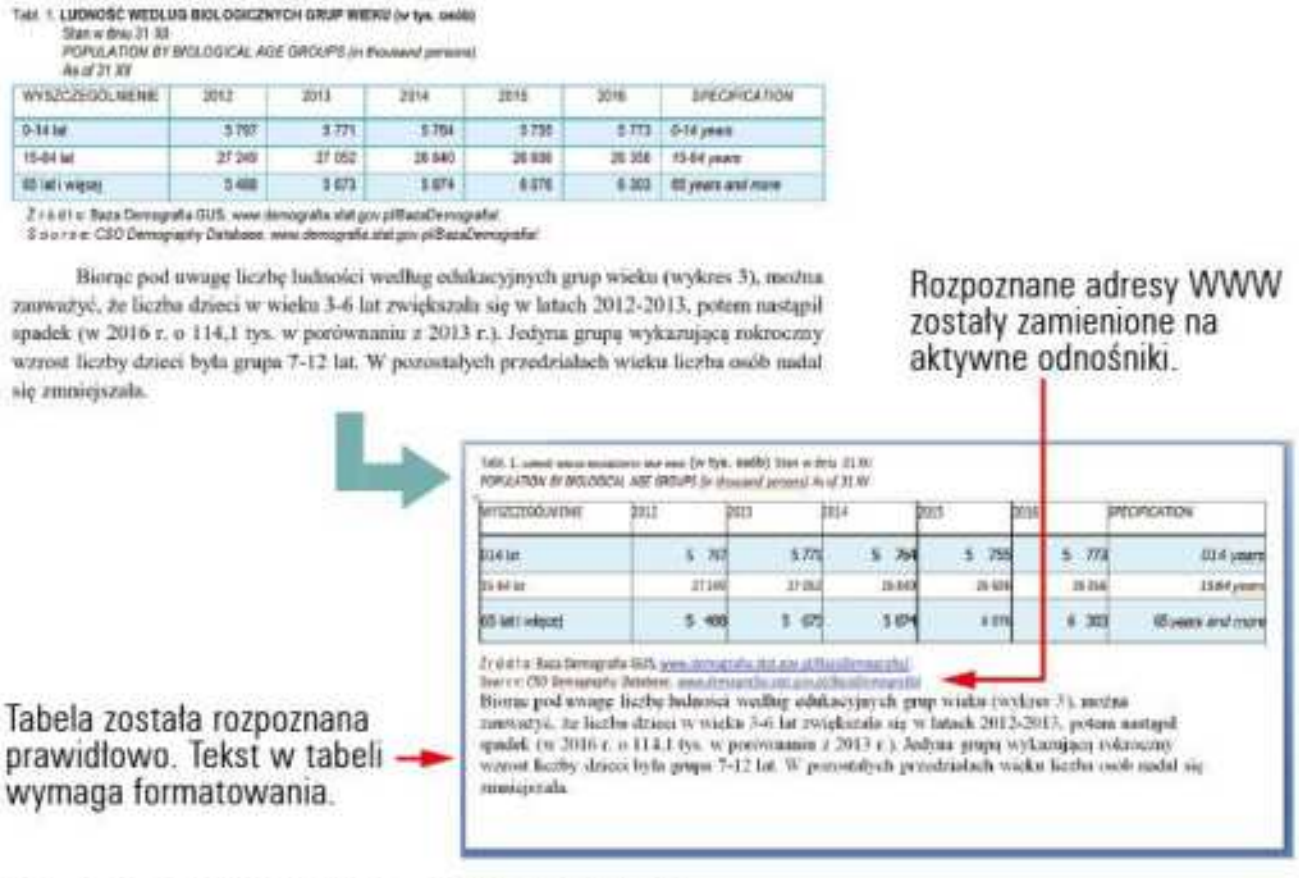
przycisk jako tekst z tłem. Rozbudowane programy OCR dają możliwość zaznaczania fragmentów, które mają być rozpoznane jako ilustracja lub tabela.




Rys. 46.7. Przykład konwersji fragmentu WWW z ilustracją

Kolejnym zadaniem stojącym przed OCR jest rozpoznawanie tabel (rys. 46.8.). Często korzystamy z danych z różnych źródeł, na przykład opracowanych przez Główny Urząd Statystyczny i dostępnych na stronie *stat.gov.pl*. Pobrane dane najczęściej mają format PDF. Jeśli narzędziem **Wycinanie** zapiszemy w pliku graficznym fragment takiego dokumentu, będzie możliwa jego konwersja. Wykonanie tego zadania dostarczyło kilku niespodzianek (rys. 46.8.).

Jak widać, darmowe narzędzia OCR nie zawsze są dokładne w odwzorowywaniu dokumentu. Mimo tych wad są bardzo pożyteczne i zdają egzamin w praktyce szkolnej i nieskomplikowanych przypadkach dokumentów.




Rys. 46.8. Konwersja OCR dokumentu z tabelą

Jeśli zdecydujesz się na założenie konta na *onlineocr.net*, otrzymasz szereg dodatkowych możliwości, a liczba stron możliwych do konwertowania zostanie ograniczona do 50. Warto jednak zapoznać się z możliwościami tej wersji. Przesłane pliki mogą mieć do 200 MB objętości, liczba formatów wejściowych i wyjściowych znacznie się powiększa (rys. 46.9.). Możliwe jest także wybieranie stron z pliku PDF.




Rys. 46.9. Opcje wersji komercyjnej onlineocr.net

**Uwaga!**

Komercyjne programy znanych producentów oparte na OCR zapewniają większą skuteczność konwersji dzięki możliwości zaznaczania obszarów z grafiką, tabelami i tekstem.

## ZADANIA DO WYKONANIA

1. Zrób zdjęcie jednej strony dowolnego czasopisma z ilustracjami. Wybierz taką, na której ilustracje nie zawierają liter. Zdjęcie skopiuj na dysk komputera. Dokonaj konwersji do pliku w formacie MS Word. Porównaj otrzymany dokument z rzeczywistą stroną czasopisma. Opisz krótko różnice. Zastanów się, skąd się wzięły. Jeśli dojdziesz do wniosku, że niektóre zostały spowodowane niedoskonałością zdjęcia, ćwiczenie wykonaj ponownie, starając się zmienić sposób wykonania fotografii.
2. Wczytaj do przeglądarki stronę z artykułem z dowolnego portalu informacyjnego. Wykonaj zrzut fragmentu ekranu z tekstem i ilustracją. Zapisz na dysku. Dokonaj konwersji do pliku MS Word dwoma różnymi programami on-line. Porównaj wyniki. Oceń oba dokumenty. Wskaż różnice.
3. Po wykonaniu zadania 2. znajdź w sieci opinie o obu programach. Porównaj je z własnymi wnioskami. Czy się pokrywają? Jeśli nie, znajdź przyczynę.
4. Załóż, że jesteś pracownikiem wydawnictwa i musisz wybrać program OCR dla dziennikarzy i pracowników redakcji. Zbierz informacje na temat najpopularniejszych programów tego typu i wybierz dwa z najlepszymi recenzjami. Wykonaj tabelę (np. w edytorze tekstu), w której porównasz oba, wpisując ich zalety i wady. Sprawdź, na jakich zasadach można z nich korzystać (zakup licencji stałej, dzierżawa, on-line itp.) i ile to kosztuje.

5*. Wczytaj wersje próbne programów wybranych w zadaniu 2. Przetestuj je. Sprawdź, czy potwierdzają się opinie znalezione w internecie. Wybierz i uzasadnij swoją decyzję.

## PODSUMOWANIE LEKCJI

- Rodzaj zastosowanego skanera do skanowania dokumentów na potrzeby **OCR** nie ma znaczenia. Liczy się rozdzielczość i dokładność odwzorowania oryginału. ◂ s. 262
- **CCD i CIS** to nazwy potocznie określające rodzaj skanera płaskiego. W rzeczywistości określają one rozwiązania techniczne, na podstawie których zbudowano skaner. ◂ s. 264
- **TWAIN** to nazwa standardu komunikacji (protokołu transmisji) pomiędzy skanerem a systemem operacyjnym. Jest on stosowany powszechnie w sterownikach do skanerów. ◂ s. 264
- Wybranie opcji skanowania w programie graficznym najczęściej uruchamia panel sterujący, w którym można ustawić parametry i kierować całym procesem. ◂ s. 264
- Automatycznie działające, najprostsze programy do konwersji na podstawie **OCR** nie zawsze dają pożądany efekt. Trudność sprawiają im ilustracje z tekstem czy tabele. Programy OCR przetwarzają pliki graficzne zapisane w różnych formatach, np. JPG, PNG, TIFF, GIFF oraz tekstowe dokumenty PDF, i eksportują do różnych formatów, np. Word, Excel, TXT, PDF. Poprawność konwersji należy dokładnie sprawdzić. Istnieją programy OCR działające on-line. ◂ s. 266

# 47. Dokumentujemy wydarzenia, czyli aktywna praca z aparatem fotograficznym

**NA TEJ LEKCJI:**

- poznasz najważniejsze funkcje aparatu fotograficznego;
- nauczysz się, jak poprawnie zrobić ciekawe zdjęcie;
- dowiesz się, w jaki sposób przygotować się do sesji zdjęciowej.

Odkąd cyfrowe aparaty fotograficzne stały się standardem, fotografia to jedno z ważniejszych zastosowań informatyki. Dzięki rozpowszechnieniu smartfonów z aparatami i łatwych w obsłudze edytorów każdy może spróbować swoich sił jako reporter lub fotograf. Czy wszyscy osiągają sukces? Dlaczego nie jest to takie proste, jakby się wydawało? Warto się nad tym zastanowić.

## 47.1. Ekspozycja, czyli zupełnie co innego niż myślisz

Aparaty fotograficzne montowane w smartfonach są w pełni zautomatyzowane. Ich oprogramowanie pozwala ponadto na korzystanie z ustawień tematycznych, jednak część użytkowników nie posługuje się tymi opcjami. Dlatego jakość zdjęcia zrobionego automatycznie należy poprawić w edytorze. Niestety nie zawsze jest to możliwe. Co zrobić, by zdjęcia były wyraźnie i dobrze naświetlone? Oczywiście wszystko zależy od aparatu – ale są pewne uniwersalne zasady, których przestrzeganie pozwoli ci uzyskać zadowalające efekty. Aby zrozumieć te zasady, musisz poznać podstawowe pojęcia związane z fotografią.

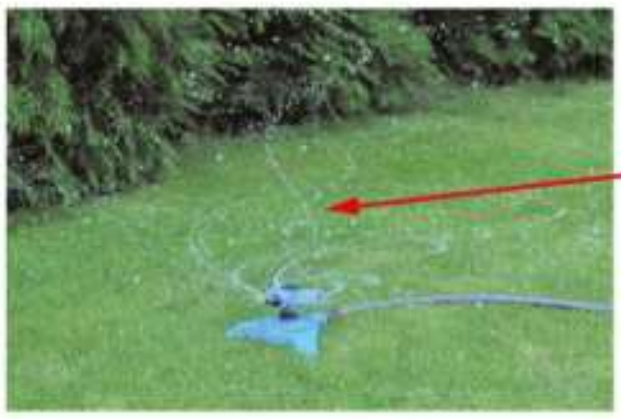
**Ekspozycja** to kombinacja czasu naświetlania i przysłony (rys. 47.4.) dająca pożądane przez fotografa naświetlenie matrycy przy zachowaniu stałej jej czułości. Ogólnie ujmując: krótki czas i otwarta przysłona mogą dać takie samo naświetlenie, co długi czas przy przymkniętej przysłonie. Oczywiście ich poziom zależy od warunków na planie zdjęciowym. W trybie automatycznym ustawienia ekspozycji są dopasowywane w zależności od wybranego programu tematycznego. Na przykład zdjęcia w trybie „Sport" będą wykonywane z krótkim czasem naświetlania, by obiekty nie były poruszone (rys. 47.1.). W trybie „Pejzaż" przysłona zostaje przymknięta, by zwiększyć głębię ostrości (rys. 47.2.).

**Uwaga!**

Przykłady nazw programów tematycznych w różnych aparatach mogą być inne niż w treści podręcznika.



Czas otwarcia migawki 1/15 sekundy, otwór względny f/16, krople wody poruszone



Czas otwarcia migawki 1/1000 sekundy, otwór względny f/5, krople wody „zamrożone"

Rys. 47.1. Wpływ czasu naświetlania na efekt końcowy fotografii





Czas otwarcia migawki 1/640 sekundy, otwór względny f/11, ostrość od planu najbliższego po daleki



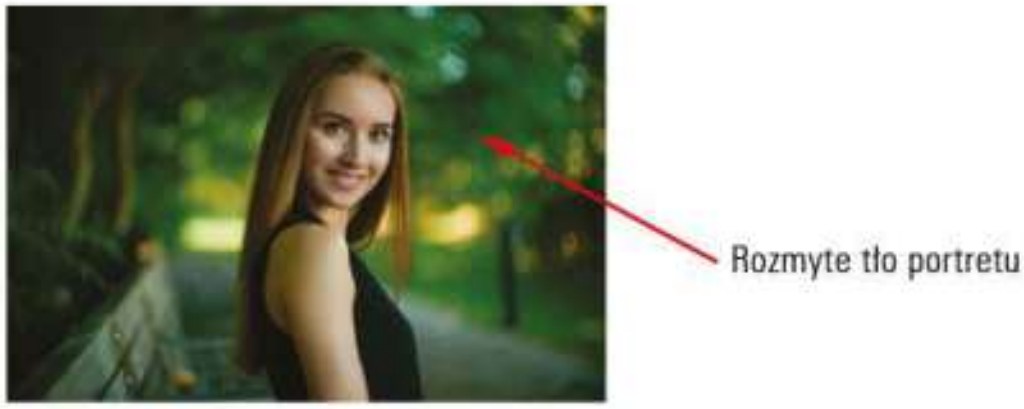


Czas otwarcia migawki 1/1000 sekundy, otwór względny f/5, ostrość na obiekcie, pozostałe plany nieostre

Rys. 47.2. Fotografie z dużą i małą głębią ostrości

**możliwości techniczne aparatu**

W aparatach montowanych w telefonach większość efektów powstaje elektronicznie po wybraniu odpowiedniej opcji. Na przykład użycie opcji rozmycie tła da podobny efekt do zdjęcia motyla z rysunku 47.2. Użyte wraz z programem portretowym da bardzo ciekawe efekty, a rozmyte tło nie będzie rozpraszać odbiorcy, który skupi się na twarzy portretowanej osoby (rys. 47.3.).




Rys. 47.3. Efekt użycia programu rozmywającego tło w aparacie smartfona

**czas naświetlenia i przysłony**

| Czas naświetlania | Otwór względny (przysłona) | Czułość matrycy |
|---|---|---|
| – czas, przez który migawka jest otwarta, czyli naświetla się matryca;<br>– w zależności od konstrukcji aparatu można ustawić w zakresie od kilku sekund do tysięcznych części sekundy;<br>– im krótszy, tym mniej poruszonych obiektów na zdjęciu, ale i „ciemniejsze" może być zdjęcie. | – decyduje o ilości światła, które po otwarciu migawki dociera do matrycy;<br>– im mniejszy otwór, tym mniej światła, ale większa głębia ostrości, czyli bliskie i odległe plany mogą być ostre. | – czułość na światło;<br>– im wyższa, tym większy szum na zdjęciu, ale można fotografować przy niewielkiej ilości światła;<br>– im wyższej klasy aparat, tym mniejszy szum i lepsza ostrość na zdjęciach w wysokich czułościach. |

Rys. 47.4. Podstawowe pojęcia dla aparatów fotograficznych

**1.** Ustawienia automatyczne dobrze sprawdzają się w naturalnych warunkach, gdy słońce nie świeci zbyt ostro i brak dużego zachmurzenia. Dlaczego?

Tab. 47.1. Niektóre cechy ustawień automatyki aparatu fotograficznego

| Warunki | Efekt działania automatyki aparatu | Przyczyna |
|---|---|---|
| Duże nasłonecznienie | Jasne miejsca są dobrze odwzorowane, a cienie ciemne, np. twarz pod daszkiem czapki. Zdarza się, że najjaśniejsze punkty zostają prześwietlone i tracą detale – stają się jasną plamą. | Automatyka ustawia ekspozycję i ostrość na podstawie pomiaru światła. Mocno oświetlone obszary mają na to duży wpływ. |
| Zmrok, miejsca zacienione | Na zdjęciach występują większe szumy (pojawia się efekt „kaszy"). Głębia ostrości jest mniejsza. Na ciemnych miejscach nie widać szczegółów. Ruchome obiekty mogą być poruszone. | Aparat ustawia większą czułość matrycy. Im większa, tym większe szumy. Wydłuża się także czas naświetlania. |
| Obiekty w ruchu, np. rowerzysta, biegacz, dzieci | Obiekty w ruchu na zdjęciu są poruszone. | Od otwarcia do zamknięcia migawki obiekt przebędzie już pewną drogę. Efekt zamrożenia obiektu w ruchu uzyskuje się przez skrócenie czasu naświetlania matrycy. Automatyka ustawia czas zgodnie z pomiarem światła. |
| Zdjęcia z niewielkiej odległości | Mogą być nieostre, gdy tło będzie jaśniejsze. Na ekranie smartfona można dotykiem ustawić miejsca ostrości, co znacznie ułatwia fotografowanie. | Automatyka samoczynnie ustawia ostrość na jasnych obiektach. W zależności od sposobu ostrzenia (w punkt lub na wielu punktach) można uzyskać różne efekty. |

2. Warto stosować programy tematyczne, na przykład sportowe, tylne światło, rozmycie tła, by uzyskać ciekawe efekty i dobrej jakości zdjęcia.

3. Aby zdjęcia miały naturalne kolory, należy stosować automatyczną korektę, na przykład światło żarowe (żarówki, halogeny itp.) – ze zdjęcia zostanie usunięty nadmiar składowej czerwonej. Wykonane z tą nastawą zdjęcia w świetle dziennym będą miały niebieskie zabarwienie.

## 47.2. Reporter i jego aparat, czyli dokumentujemy wydarzenia

Dzięki łatwemu dostępowi do aparatów fotograficznych w komórkach, każdy jest w stanie sfotografować ciekawe wydarzenie i opublikować relację. Ty także możesz zostać szkolnym dziennikarzem.

**Uwaga!**

Upubliczniając swoje zdjęcia lub relacje, musisz przestrzegać podstawowych zasad.

1. Nie publikuj wizerunków osób bez ich zgody. Wyjątek stanowią zdjęcia tłumu, np. młodzież na holu szkoły. Tu jednak także musisz uważać na to, czy ktoś nie został w jakiś szczególny sposób wyróżniony, na przykład ostrością, kolorem, wielkością na zdjęciu.
2. Nigdy nie publikuj zdjęć w złych intencjach, na przykład ośmieszenia kogoś, naruszenia prywatności.
3. Przekazuj rzeczywisty obraz wydarzeń. Zachowaj obiektywizm.
4. Podpisuj swoje relacje.
5. Działaj zgodnie z prawem. Jego nieznajomość nie zwalnia z odpowiedzialności.

**przygotowanie do sesji zdjęciowej**

Jeśli używasz swojego aparatu do zadań reporterskich, odpowiednio go przygotuj.

1. Ustaw automatykę w taki sposób, by jak największa liczba zdjęć nadawała się do publikacji, czyli była odpowiednio naświetlona i ostra.
2. Dbaj o odpowiednią ilość miejsca na karcie telefonu. Musisz liczyć około 4 do 10 MB na jedno zdjęcie. Warto sprawdzić, ile miejsca zajmują w twoim aparacie.
3. Noś przy sobie odpowiedni kabel do połączenia z komputerem lub czytnik kart. Jeśli swoje relacje upubliczniasz bezpośrednio z telefonu, miej zawsze aktywny pakiet internetowy. Zainstaluj odpowiednie oprogramowanie, na przykład aplikację portalu społecznościowego.
4. Naładuj baterię aparatu przed pracą, a jeśli to możliwe, zaopatrz się w zapasową lub powerbank.
5. Dobrze poznaj możliwości i opcje swojego aparatu.

**przesyłanie zdjęć do komputera**

Zdjęcia z twojego aparatu możesz wczytać do komputera, a następnie je edytować. W tym celu wykorzystasz Bluetooth, WiFi lub kabel USB. Dobrym rozwiązaniem jest też odczytanie karty SD bezpośrednio w czytniku komputera. Sprawdź, jakie możliwości ma twój aparat.

**Przykład**

Załóżmy, że relacjonujesz mecz szkolnej drużyny. Fotoreportaż i artykuł musisz napisać jak najszybciej. Jak poradzić sobie z takim zadaniem?

Znasz składy drużyn, termin meczu, miejsce itp., więc część tekstu możesz ułożyć przed meczem. Jeśli przygotujesz relacje na dwa warianty wyniku, to uzupełnienie jej faktami nie zabierze ci dużo czasu.

Zdjęć jednak nie przygotujesz przed wydarzeniem. Zaplanuj swoją pracę, by ułatwić sobie zadanie. W czasie wydarzeń sportowych nie da się niczego powtórzyć czy ustawić ponownie. Zapoznaj się więc z miejscem akcji. Zaplanuj, skąd będziesz fotografować wydarzenia na boisku, a skąd publiczność, ławkę trenera, występy w czasie przerw. Ustaw odpowiednio swój aparat i przeprowadź próbę na miejscu. Jeśli mecz jest wyjazdowy i nie wiesz, jak wygląda hala rywala, znajdź jej zdjęcia w internecie. Narysuj plan swojej pracy, zaznacz miejsca, w których spodziewasz się dobrego widoku. Zaplanuj, w jakiej fazie spotkania staniesz w tych miejscach. W razie potrzeby, na przykład gdy wydarzy się coś nieprzewidywalnego, reaguj spontanicznie i improwizuj. Jednak twój aparat musi być w ciągłej gotowości.

Nie publikuj wszystkich zdjęć. Wybierz takie, które możesz ułożyć w logicznej kolejności. Twój fotoreportaż powinien opowiadać o tym wydarzeniu, by odbiorca poczuł atmosferę i dramaturgię wydarzenia.

Jest więcej zasad, które mogą sprawić, że twoja praca będzie ciekawa. Poczytaj o nich na stronach internetowych i blogach fotografów (rys. 47.5.).




Rys. 47.5. Fragment strony poświęconej komponowaniu fotoreportaży *dg.art.pl/zasady-komponowania-reportazu-fotoreportazu/*

## 47.3. Przypomnij sobie, czyli krótko o kompozycji obrazu

kadrowanie ▸

Zasady komponowania obrazu nie dotyczą tylko fotografii. To ogólnie stosowane sposoby dobrego przekazu za pomocą obrazu. Mają zastosowanie w sztuce, grafice komputerowej, stronach internetowych, grach itp. Spotkałeś/spotkałaś się z nimi nie tylko na lekcjach informatyki, lecz także innych przedmiotów.

1. Przed zrobieniem zdjęcia zawsze pomyśl o tym, co chcesz na nim przedstawić. Zwracaj uwagę na detale, które mogą skupiać na sobie uwagę odbiorcy, ale nie są tematem zdjęcia (rys. 47.6.).

reguła trójpodziału ▸

2. Jeśli nie masz doświadczenia lub pomysłu na kompozycję, stosuj znane reguły, na przykład trójpodziału. Sprawdza się ona w większości przypadków. Podczas kadrowania najważniejszy obiekt umieść w silnym punkcie, czyli w jednym ze skrzyżowań linii trójpodziału (rys. 47.7.). Jeśli fotografujesz pejzaż, możesz wykorzystać pionowe lub poziome linie i zgrać z nimi linie obrazu, na przykład podłużny obiekt, brzeg morza, drzewa itp. Wiele aparatów fotograficznych i w telefonach ma opcję nałożenia na podgląd linii trójpodziału – wykorzystaj ją.




Rys. 47.6. Przykład zdjęcia zakłóconego przez zbędne elementy



Rys. 47.7. Przykład zastosowania reguły trójpodziału

Czytanie o fotografowaniu i komponowaniu reportaży nie nauczy cię praktycznych umiejętności. Musisz wziąć aparat do ręki, wykonać zdjęcia, a potem samodzielnie je posegregować i ułożyć. Im więcej ćwiczeń i sprawdzania się w akcji, tym lepsze będą wyniki. Próbuj więc i baw się dobrze, pamiętając, że opisane w rozdziale przykłady są niewielkim fragmentem wiedzy na temat fotografowania.

## ZADANIA DO WYKONANIA

1. a) Znajdź w internecie instrukcję obsługi swojego aparatu fotograficznego lub telefonu z aparatem. Wskaż programy tematyczne lub nastawienia automatyki i omów ich przeznaczenie. Sprawdź ich działanie, fotografując ten sam obiekt w podobnych warunkach.

**b)** Przygotuj plik, w którym umieścisz obok siebie zdjęcia wykonane w punkcie a. Dla każdego z nich określ, jaki program lub nastawienie automatyki zostały użyte do jego wykonania. Uzasadnij odpowiedź.

2. Wyszukaj w internecie strony lub blogi redagowane przez zawodowców na temat fotografowania wydarzeń sportowych itp. Zbierz najważniejsze rady profesjonalistów i wykonaj prezentację z przykładami. Możesz wykorzystać fragmenty znalezionych stron w charakterze cytatów.
3. Omów w sieci przykłady zastosowania reguły trójpodziału w sztuce na przestrzeni wieków. Opisz kilka ciekawych przykładów, np. na obrazach Jana Matejki. Sporządź prezentację, w której przedstawisz najciekawsze przypadki.
4. Wykonaj zdjęcia wody w ruchu, np. cieknącej z kranu, z różnymi ustawieniami migawki lub programów tematycznych. Uzyskaj dwa efekty – woda jest w ruchu (czyli krople są poruszone) i ruch wody jest „zamrożony". Wykonaj prezentację na temat zjawisk, które pozwoliły na uzyskanie takich efektów.
5. Wybierz ze swoich zbiorów kilka zdjęć z uroczystości, np. zakończenia roku szkolnego. Skomponuj z nich fotoreportaż. Skorzystaj z rad podanych na blogu *dg.art.pl/zasady-komponowania-reportazu-fotoreportazu/* lub innej, wiarygodnej strony na ten temat. Wydrukuj zdjęcia lub utwórz elektroniczny album i zaprezentuj je znajomym nieuczestniczącym w wydarzeniu. W rozmowie zapytaj o ich odczucia i sprawdź, czy odczytali przekaz zgodnie z twoją intencją.

## PODSUMOWANIE LEKCJI

- Każdy aparat fotograficzny nadaje się do dokumentowania wydarzeń. Należy optymalnie wykorzystać jego **możliwości techniczne**. ◂ s. 272
- Optymalne naświetlenie matrycy może być wynikiem różnych kombinacji **czasu naświetlania i przysłony**. Dają one jednak różniące się efekty, np. „zamrożenie" lub poruszenie obiektu, płytką lub głęboką głębię ostrości. Świadome korzystanie z ręcznych ustawień ekspozycji lub programów tematycznych może znacznie poprawić **jakość zdjęcia**. Ma to także wpływ na treść fotografii, np. poprzez wyostrzenie niektórych obiektów, rozmycie tła. ◂ s. 272
- Przed sesją zdjęciową należy zadbać o naładowanie akumulatorów, przygotowanie dodatkowego źródła zasilania, zabezpieczenie odpowiedniej ilości pamięci na karcie. Jeśli będziesz fotografować wydarzenie sportowe, uroczystość szkolną itp., postaraj się zapoznać z miejscem akcji, przygotować plan miejsc, w których staniesz itd. ◂ s. 274
- W czasie fotografowania zwróć uwagę, czy w **kadrze** nie znajdują się niepotrzebne, przykuwające wzrok obiekty. Nie wszystko da się usunąć w programie komputerowym. ◂ s. 276
- Jeśli nie masz doświadczenia lub pomysłu na skadrowanie zdjęcia, skorzystaj z **reguły trójpodziału**. Możesz wykorzystać opcję nałożenia jej na podgląd w swoim aparacie. ◂ s. 276
- Zdjęcia z aparatu możesz przesyłać do komputera za pośrednictwem **łączy bezprzewodowych Bluetooth lub WiFi, kablem USB** lub poprzez odczyt karty wyjętej z urządzenia. ◂ s. 274

# 48. Szturmowiec w chmurze, czyli poprawiamy zdjęcia w edytorze grafiki rastrowej

**NA TEJ LEKCJI:**

- poznasz, jakie są niekomercyjne edytory grafiki rastrowej;
- użyjesz narzędzi niekomercyjnych programów do obróbki zdjęć.

Na pewno masz już pewne doświadczenia związane z przetwarzaniem zdjęć. Wiesz, że można je kadrować, zmieniać jasność, barwy, a także że do tych czynności są potrzebne odpowiednie narzędzia. Do prostych prac nad fotografią nie trzeba wykorzystywać komercyjnych programów graficznych. Istnieje wiele darmowych programów komputerowych, które z powodzeniem zastąpią niektóre narzędzia profesjonalne. Są wśród nich również programy w chmurze. Gdy dojdziesz do wprawy w posługiwaniu się nimi, znacznie szybciej opanujesz narzędzia profesjonalne.

## 48.1. Warstwy w chmurach, czyli edytujemy zdjęcie

edytory grafiki rastrowej w chmurach informatycznych

Wśród edytorów chmurowych możesz znaleźć także takie, których interfejs i podstawowe czynności edycyjne są bardzo zbliżone do profesjonalnych programów, takich jak PhotoShop. Jednym z nich jest **Pixlr** (*www.pixlr.com*). Dzięki niemu wykonasz wszystkie podstawowe czynności edycyjne. Oczywiście wiele z nich pozwala również na zaawansowaną edycję. Warto się z nimi zapoznać. Opisane ćwiczenia możesz także wykonać z pomocą innego edytora grafiki.

Nowy edytor najlepiej poznać poprzez ćwiczenia.

**Przykład**

Pewien reporter wybrał się na zlot miłośników filmów *science fiction*. Jedno ze zdjęć wykonał z dużej odległości spośród tłumu uczestników. Trudno mu było odpowiednio ustawić kadr. Na fotografii znalazło się też kilka niepotrzebnych elementów. Należy więc poddać ją obróbce elektronicznej. Spośród swoich zdjęć wybierz takie, które wymaga retuszu i poddaj je podobnym zabiegom, jak w poniżej opisanym ćwiczeniu.

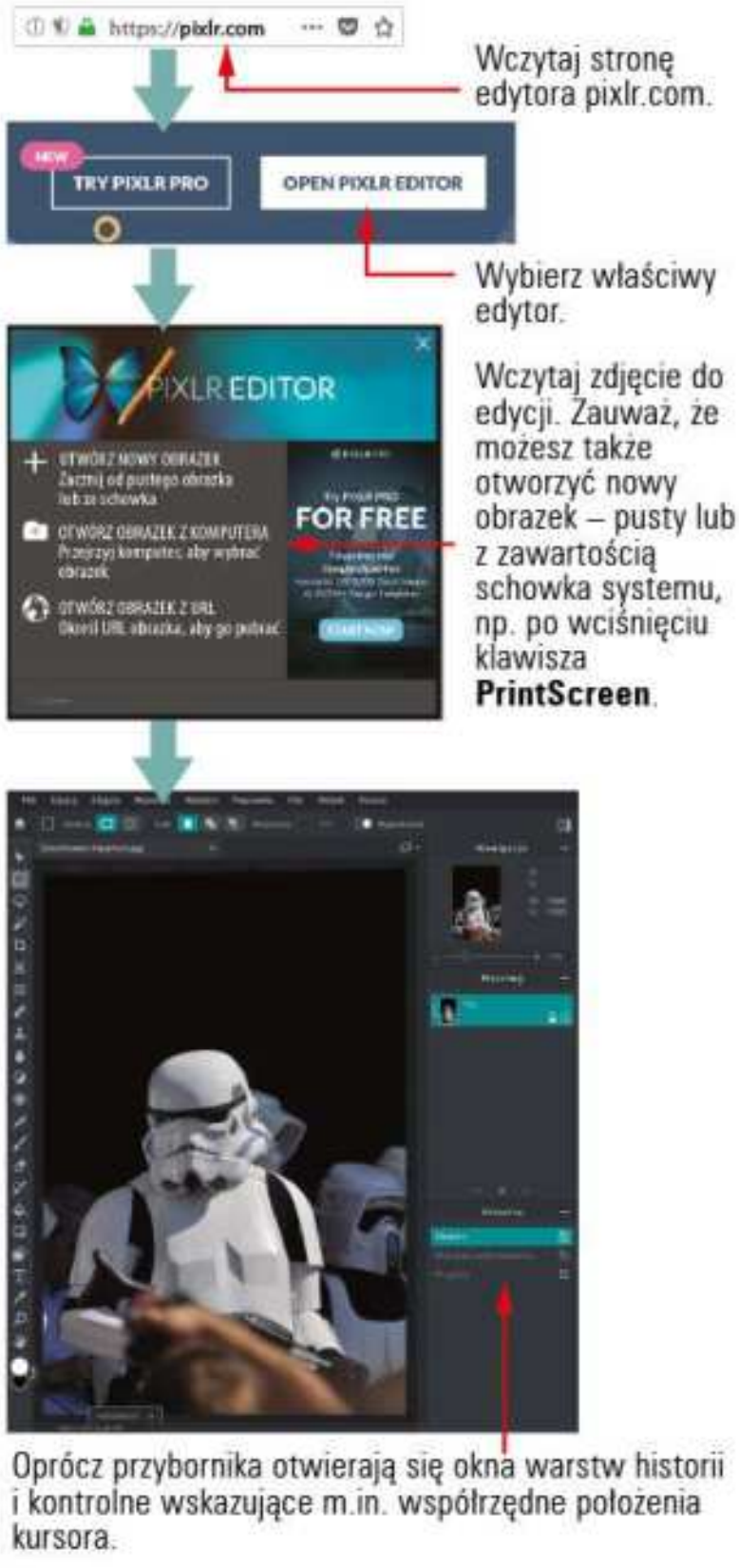
Użyj programu z chmury informatycznej *pixlr.com* (rys. 48.1.).

Przeanalizuj przykładowe zdjęcie i porównaj ze swoim. Być może będzie miało te same wady.

## Analiza zdjęcia

1. Wymaga kadrowania. Prawie połowa powierzchni jest czarna.
2. Na dole kadru głowa jednego z uczestników zbytnio przykuwa uwagę. Niczego nie wnosi do zdjęcia.
3. Po prawej stronie głównej postaci widać fragment hełmu innego uczestnika.

Jak pozbyć się tych niedoskonałości? Użyjmy narzędzi programu Pixlr. Pracę nad korektą rozpocznij od kadrowania. Dla pełnej swobody zaznaczania obszaru dla narzędzia **Kadrowanie** wybierz parametr **Brak ograniczenia**.




Rys. 48.1. Zdjęcie wczytane z dysku do edytora pixlr.com w chmurze




Rys. 48.2. Kadrowanie zdjęcia w pixlr.com

**zapisywanie wyniku pracy** ▸ Wynik swojej pracy co pewien czas zachowuj. Rób to, pobierając kopię na dysk swojego komputera. Oczywiście te same czynności wykonaj po zakończeniu pracy.




Rys. 48.3. Zachowanie zdjęcia edytowanego w chmurze pixlr.com

Czas na usunięcie niechcianego elementu. Posłuż się narzędziem **Lasso**. Wybierz **Wielokątne lasso** i zaznacz pole **Wygładzanie**. Lasso tego typu wyeliminuje problem dokładności zaznaczania, ponieważ będzie możliwe zaznaczanie obszaru za pomocą odcinków (dowolnej długości), co w połączeniu z wygładzaniem krawędzi da oczekiwany efekt. W przypadku korzystania z **Odręcznego lassa** dokładność zaznaczania zależy od precyzji prowadzenia kursora i rozdzielczości myszki. Sprawdza się przy zaznaczaniu skomplikowanych kształtów. Opcja **Pióro** ustala stopień wygładzania. Im większa liczba, tym większy łuk wygładzania. Ponieważ granica zaznaczanego obszaru może być stosunkowo długa, dobrym sposobem jest zaznaczenie fragmentu (rys. 48.4.), wykonanie na tym obszarze odpowiednich działań i powtórzenie ich dla następnych zaznaczeń. Jeśli precyzyjnie operujesz myszką, możesz zaznaczyć cały obszar w jednej operacji.

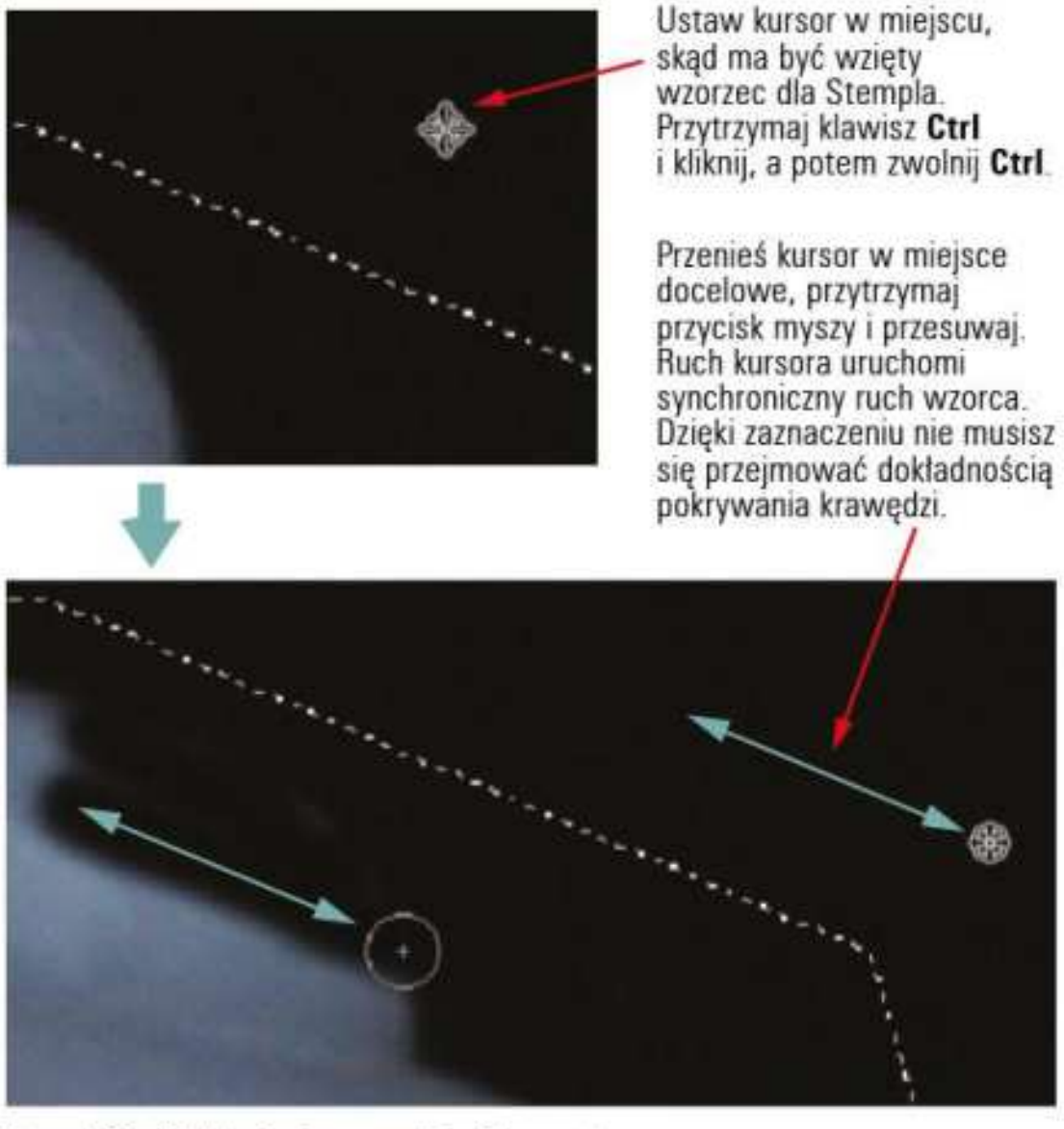



Rys. 48.4. Zaznaczanie narzędziem Wielokątne lasso

Zaznaczony obszar należy wypełnić. W tym przypadku użycie konkretnego koloru nie zda egzaminu. Mimo że tło w przykładzie wydaje się jednolite, w rzeczywistości takie nie jest – ma sporo odcieni. Użyjmy więc narzędzia **Stempel** (rys. 48.5., 48.6.). Jest ono bardzo przydatne przy retuszu zdjęć.




Rys. 48.5. Proces użycia narzędzia Stempel




Rys. 48.6. Efekt użycia narzędzia Stempel




Rys. 48.7. Opcja cofnięcia czynności w edytorze

Efekt, jaki uzyskasz, zależy od precyzji i cierpliwości. W razie nieudanych posunięć możesz wykorzystać z menu **Redaguj** opcję **Cofnij** lub **Ctrl+Z** (rys. 48.7.).

Podobne czynności możesz wykonać na przykład podczas usuwania fragmentu broni szturmowca.

Zauważ, że jego lewe ramię sprawia wrażenie, jakby kończyło się nienaturalnie tuż za zbroją. W rzeczywistości tak nie jest. Poprawmy ten mankament (rys. 48.8.).




Rys. 48.8. Rozjaśnianie fragmentu zdjęcia

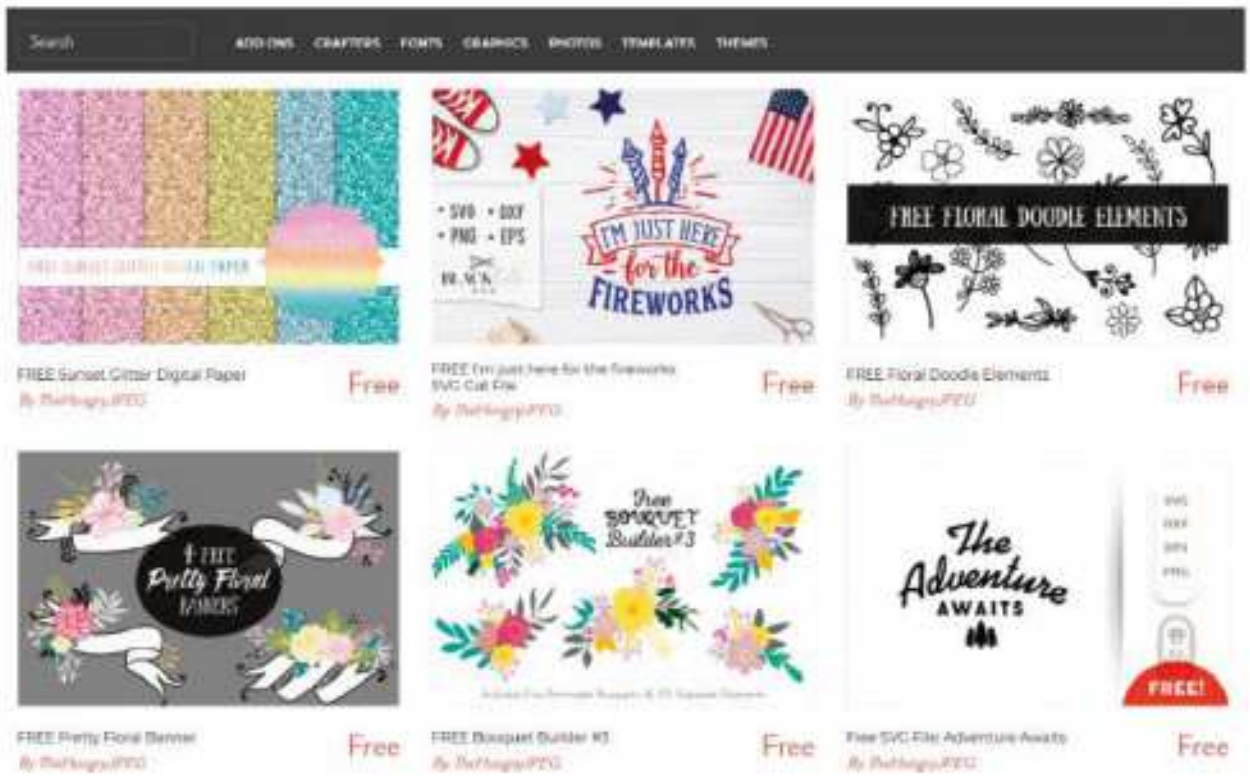


Rys. 48.9. Zdjęcie po korekcie

Jeśli dokładnie wykonałeś/wykonałaś swoje ćwiczenia, to zapewne powstały efekt jest dobry. Teraz możesz użyć go w swoich publikacjach. Użyte narzędzia działają podobnie w innych programach, w tym także komercyjnych. Mogą różnić się liczbą parametrów, które można ustawiać, jednak ogólne zasady ich stosowania są bardzo podobne. Ćwiczenia z darmowymi edytorami dają dobrą podstawę do poznania i używania zaawansowanych edytorów.

Pixlr możesz także wykorzystać do tworzenia grafiki, podobnie jak GIMP w jednym z poprzednich ćwiczeń. Jeśli jesteś dociekliwy/dociekliwa, znajdziesz na stronie głównej jeszcze inne

niespodzianki. Przykładem może być zbiór plików graficznych, które można wykorzystać w swoich projektach (rys. 48.10.).




Rys. 48.10. Darmowe grafiki ze strony pixlr.com

Jeśli nie skończysz edycji w Pixlr, a używane były warstwy, zachowaj swoje zdjęcie lub grafikę w formacie PXD. Dzięki temu będzie możliwa dalsza edycja w późniejszym terminie. Podobnie postępuj z innymi edytorami. Część programów do edycji zdjęć ma także swoje odpowiedniki dla smartfonów. Są one zazwyczaj pozbawione wielu narzędzi z powodu obsługi za pomocą ekranu dotykowego, co znacznie ogranicza precyzję operowania (rys. 48.11.).




Rys. 48.11. Pixlr w wersji mobilnej

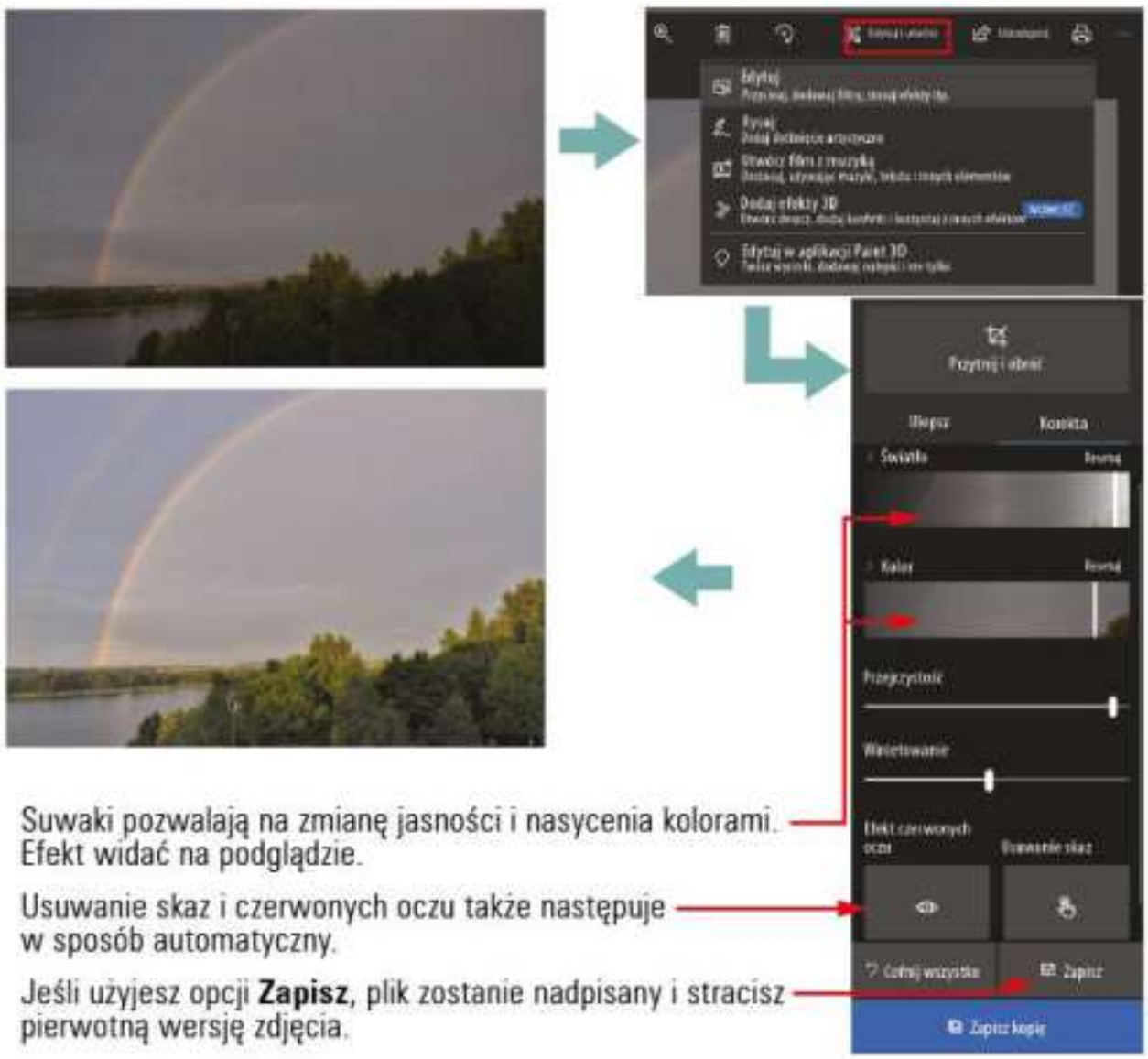
## 48.2. Nie wyważaj otwartych drzwi, czyli narzędzia do korekty grafiki

Do tej pory zajmowaliśmy się edycją szczegółów fotografii lub grafiki. Obok niedokładnego kadrowania najczęściej występującą wadą zdjęć są źle naświetlone kadry. Przyczyną mogą być złe warunki oświetleniowe. Zdjęcie może więc być niedoświetlone, prześwietlone lub zabarwione, na przykład światłem odbitym od kolorowej ściany. Korektę można przeprowadzić zaawansowanym edytorem, jak GIMP czy PhotoShop.

uproszczone edytory zdjęć

Jednak w wielu sytuacjach, szczególnie w przypadku zdjęć amatorskich, wystarczy użycie prostszych programów, na przykład dostępnej w Windows aplikacji **Zdjęcia**.

Przykładowe zdjęcie jest niedoświetlone. Użyjmy programu **Zdjęcia** i poprawmy tę wadę. Miniaturę pliku zdjęcia kliknij prawym przyciskiem myszy i wybierz **Otwórz za pomocą \ Zdjęcia**.




Rys. 48.12. Edycja podstawowych parametrów programem „Zdjęcia"

Program **Zdjęcia** (dostępny w Windows) jest dowodem na to, że nieskomplikowane programy mogą także być skuteczne w amatorskim przygotowaniu fotografii do publikacji. Podobnych aplikacji można znaleźć więcej. Warto z nich korzystać, by nie instalować zaawansowanych programów do wykonania prostych czynności.

## ZADANIA DO WYKONANIA

1. Użyj wyszukiwarki internetowej i znajdź kilka edytorów grafiki rastrowej. Porównaj ich możliwości i sposób posługiwania się podstawowymi narzędziami edycyjnymi z Pixlr. Opisz różnice i ustaw programy w kolejności od najlepszego do najmniej przydatnego. Ponieważ będzie to twoja subiektywna ocena, uzasadnij swój wybór.
2. Wykorzystaj wybrany w zadaniu 1. sieciowy edytor grafiki rastrowej (nie Pixlr) do zmiany nasycenia barw, jasności i kontrastu zdjęcia z własnej kolekcji. Staraj się zapamiętać działanie poszczególnych opcji i narzędzi.
3. Załóż konto za pośrednictwem programu Pixlr i sprawdź, jakie są zalety posiadania takiego konta. Omów funkcje i przydatność **Biblioteki obrazków**.
4. Uruchom program **Zdjęcia**. Odnajdź opcję **Utwórz film z muzyką**. Wykorzystaj ją do utworzenia filmu ze zdjęć z własnej kolekcji. Postaraj się odpowiednio dobrać kolejność i tematykę zdjęć. Zaprezentuj swoje dzieło i uzasadnij dobór zdjęć i ich kolejności.
5. Sprawdź działanie pozostałych opcji programu **Zdjęcia**. Dodaj do zdjęcia ze swojego zbioru efekty 3D (opcja **Dodaj efekty 3D**). W portretach usuń efekt czerwonych oczu. Podaj kilka przykładów wykorzystania dodanych efektów.

## PODSUMOWANIE LEKCJI

- Większość czynności podczas amatorskiej obróbki zdjęć możesz wykonać za pomocą **niekomercyjnych programów**. **Narzędzia i opcje niektórych niekomercyjnych edytorów** zdjęć lub grafiki rastrowej są podobne do narzędzi komercyjnych. Różnią się liczbą parametrów i zakresem działania. Opanowanie ich w programie darmowym znacznie ułatwi późniejszą pracę z profesjonalnymi programami. ◂ s. 278
- Edytory grafiki rastrowej znajdziesz także w **chmurach informatycznych**. ◂ s. 279
- Jeśli wykonujesz wiele prób **modyfikacji zdjęcia,** zawsze zachowuj je w oddzielnych plikach. Dzięki temu zachowasz oryginał zdjęcia. ◂ s. 280
- **Uproszczone edytory zdjęć** przydają się w szybkim kadrowaniu, regulacji jasności i nasycenia kolorów. Ułatwiają też likwidację skaz i tzw. czerwonych oczu. ◂ s. 284

# 49. Tego tu nie było, czyli poprawiamy rzeczywistość na fotografiach

**NA TEJ LEKCJI:**

- przypomnisz sobie zasady pracy z warstwami;
- poznasz narzędzia do nakładania warstw na zdjęciu.

Umiesz już wykorzystać edytor grafiki rastrowej do kadrowania i poprawiania jakości zdjęć. Posługujesz się edytorami grafiki z chmur informatycznych i aplikacjami lokalnymi. Twoje zdjęcia są coraz lepsze i możesz ich używać do tworzenia różnych elementów graficznych, na przykład tła, nagłówków stron, banerów reklamowych. W tego rodzaju projektach bardzo przydatne są warstwy. Używałeś/używałaś ich już podczas tworzenia trójwymiarowego napisu. Czas na zastosowanie w edycji zdjęcia.

## 49.1. Warstwy jak folie, czyli odbudowujemy fasadę

Wyobraź sobie kilka przezroczystych folii, na których kolejno narysowano tło i fazy ruchu postaci. Jeśli na tło położymy warstwę (folię) z pierwszą fazą ruchu i sfotografujemy, otrzymamy początkowy kadr filmu animowanego (rys. 49.1.). Teraz wystarczy powtórzyć to dla następnych etapów poruszania się bohatera. Dysponując takimi zdjęciami, możemy utworzyć film animowany. Dokładnie takiej metody używali twórcy filmów rysunkowych, zanim wprowadzono animacje komputerowe.



Rys. 49.1. Kadr z filmu wykonany z wykorzystaniem warstw

**warstwy w edycji zdjęcia**

Wykorzystajmy warstwy do operacji na fotografii. Użyj programu GIMP lub innego edytora grafiki rastrowej obsługującego warstwy, na przykład pixlr.com.

Użyj fotografii ze swoich zbiorów, która wymaga korekty, ze względu na znajdujące się na niej zbędne elementy. Trzeba je usunąć poprzez przykrycie innym fragmentem, na przykład rośliną, latarnią. W przykładzie użyto zdjęcia wejścia do grobowca ozdobionego czterema rzeźbami, a właściwie trzema, ponieważ jedna z nich została zniszczona. Czy można to naprawić i przywrócić prawdopodobny wygląd tego miejsca sprzed zniszczenia? W rzeczywistości będzie to trudne, ale za pomocą edytora grafiki możesz „wyrzeźbić" brakujący fragment posągu.

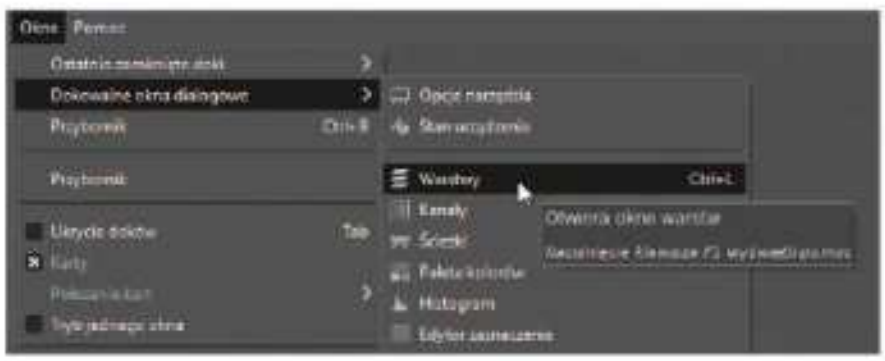
◂ edycja zdjęcia

Pracę rozpocznij od wczytania zdjęcia do edytora (rys. 49.2). W razie potrzeby popraw kadrowanie, jasność, kontrast i nasycenie kolorami. Opis dotyczy edytora GIMP. Użycie innego nie powinno sprawić ci problemów. Narzędzia i sposoby ich użycia czy obsługa warstw są bardzo podobne w różnych programach. Musisz jednak odnaleźć odpowiadające sobie narzędzia i opcje. Zanim rozpoczniesz edycję, sprawdź, czy okno z warstwami jest widoczne (rys. 49.3.).



Rys. 49.2. Oryginał zdjęcia starożytnych posągów

Obejrzyj dokładnie swoje zdjęcie i zaplanuj działania. Zlokalizuj fragment do poprawy i określ zakres prac. Gdy już ustalisz wielkość i kształt obszaru, dopasuj do niego fragment, którym go zakryjesz. Może on pochodzić z innego zdjęcia. Kolejność następnych działań może być taka, jak na rysunku 49.4.



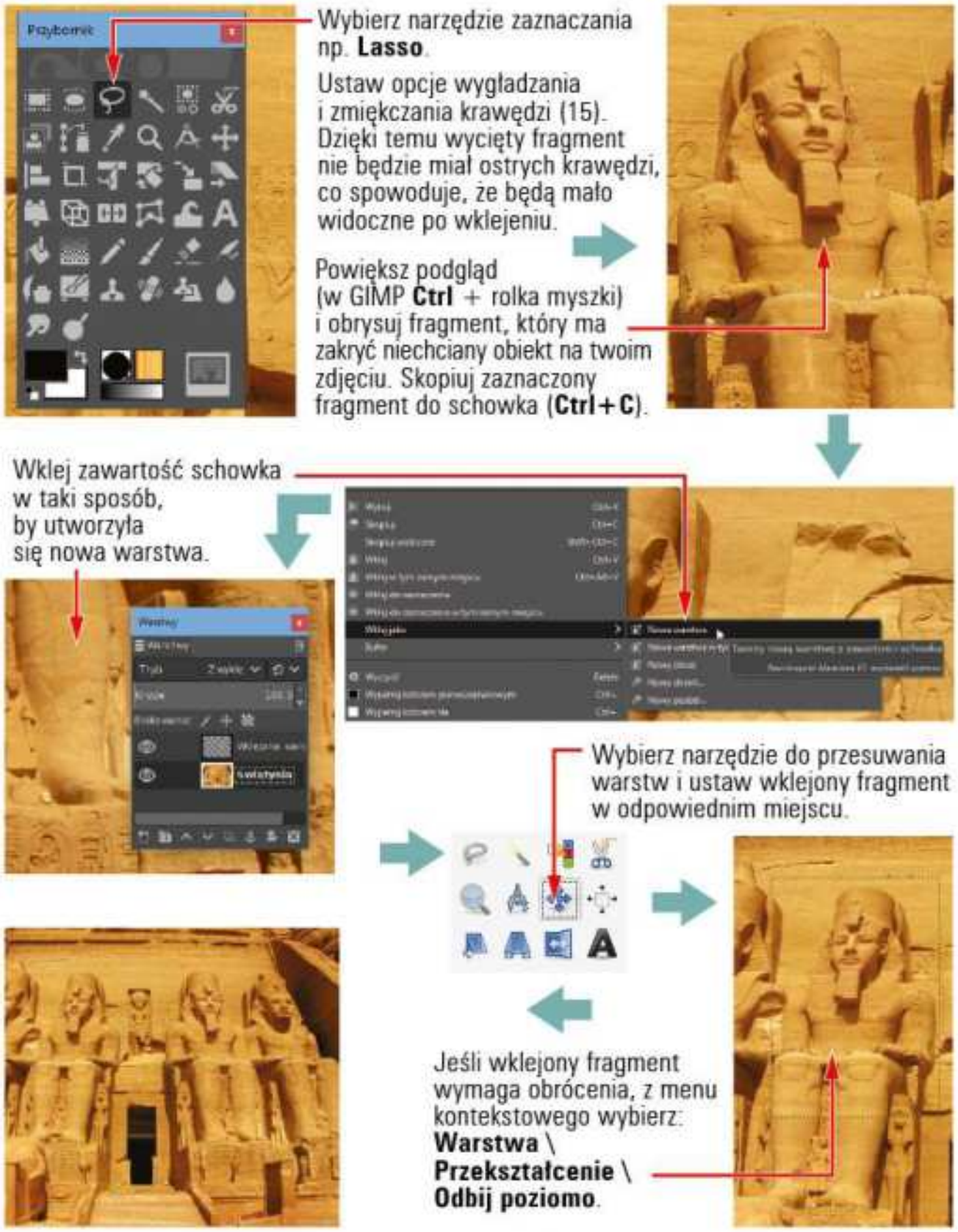

Rys. 49.3. Otwieranie okna Warstwy w GIMP

Przed edycją upewnij się, czy okno warstw jest aktywne. Możesz je otworzyć z menu głównego.




Rys. 49.4. Proces wycinania i wklejania fragmentów zdjęcia za pośrednictwem schowka i warstw

Efekt końcowy zależy już tylko od twojej inwencji, cierpliwości i sprawności posługiwania się myszką. Pomocny będzie rysunek 49.5.




Rys. 49.5. Proces rekonstrukcji z wykorzystaniem innego fragmentu zdjęcia

**Uwaga!**

Fragment rysunku obrysowuj odcinkami o różnej długości, klikając **Lassem** w odpowiednie miejsca (rys. 49.6.). Większość kątów między poszczególnymi fragmentami zostanie zaokrąglona przez program (proces wygładzania) po zakończeniu zaznaczania. Każdy z punktów końcowych odcinków możesz przemieszczać, dzięki temu łatwo poprawisz niedokładności.



Rys. 49.6. Użycie narzędzia „Lasso" do zaznaczania odcinkami

Twoje zdjęcie zostało poprawione. Do czego można wykorzystać tę technikę? Spore możliwości daje podczas odrestaurowywania starych fotografii. Jeśli masz takie w rodzinnych zbiorach, możesz przeprowadzić renowację i wydrukować. Często to jedyna zachowana odbitka i kopia może być atrakcyjnym upominkiem dla członków rodziny (rys. 49.7.).

renowacja starych zdjęć



Rys. 49.7. Archiwalne zdjęcie po renowacji

Innym pomysłem na wykorzystanie warstw i kopiowanie fragmentów zdjęć może być przygotowanie zdjęć znanych miejsc na konkurs „Znam naszą okolicę". Fałszywe i autentyczne zdjęcia mogą ilustrować ciekawe pytania (rys. 49.8.).



Rys. 49.8. Przykład montażu zdjęcia na potrzeby konkursu

## ZADANIA DO WYKONANIA

1. Obejrzyj dokładnie zdjęcie wykorzystane w ćwiczeniu. Jeśli znajdziesz inne niedoskonałości, napraw je znaną ci już metodą. O końcowym efekcie w dużej mierze zadecyduje wybór obszaru użytego do ich zakrycia.
2. Wróć do edycji zdjęcia. Zbadaj różnice pomiędzy zaznaczaniem obszarów z różnymi ustawieniami parametru **Zmiękczanie krawędzi**. Sprawdź także wpływ opcji **Wygładzanie** na efekt zaznaczenia. Oceń ich wpływ na skopiowany i wklejony fragment. Określ granice (wartość parametrów), przy których wklejony fragment odróżnia się od tla i gdy wtapia się niezauważalnie.
3. Znajdź w swoich zbiorach zdjęcie ulicy lub placu, na którym jest mało zieleni. Wklej do niego odpowiednio skalowane drzewa i inne rośliny w taki sposób, by pokazać, jak mogłaby wyglądać okolica, gdyby zadbano o zieleń. Wykorzystaj fragmenty innych zdjęć, np. alei parkowej lub lasu.
4. Wykonaj lub znajdź w swoich zbiorach zdjęcie dużego drzewa. Po uzyskaniu zgody kolegów i koleżanek wykonaj ich portrety. Umieść ich miniatury na drzewie. Dołóż tekst zawierający nazwę szkoły, rok i klasę (użyj oddzielnej warstwy). W ten sposób powstanie ciekawa pamiątka szkolna.
5. Zeskanuj stare rodzinne zdjęcia. Wybierz te najbardziej zniszczone i przeprowadź ich renowację. Przed uzupełnianiem braków i zagnieceń popraw kontrast i jasność zdjęcia. Jeśli zajdzie potrzeba, dokonaj zmiany tych parametrów jedynie dla fragmentu zdjęcia (np. twarzy lub detalu), wykorzystaj warstwy. Przed skanowaniem ustaw odpowiednią rozdzielczość skanera, np. 300 dpi.

## PODSUMOWANIE LEKCJI

s. 286 ▸ • **Warstwy** są jednym z najważniejszych mechanizmów tworzenia grafiki i montażu oraz retuszu zdjęć.

s. 286 ▸ • Każdą warstwę możesz edytować osobno. Możesz włączać i wyłączać **widzialność każdej warstwy**. Warstwę można także **skalować i obracać**, co ułatwia dopasowanie wklejanego elementu.

s. 288 ▸ • Do zaznaczania obszarów możesz użyć dowolnego **narzędzia zaznaczania**. Wybieraj najodpowiedniejsze do danego kształtu.

s. 288 ▸ • Wklejony element będzie wyglądał naturalnie, gdy podczas zaznaczania użyjesz opcji **zmiękczania krawędzi i wygładzania**. Wartości tych parametrów można regulować. Warto wypróbować ich działanie.

s. 289 ▸ • Warstw można także używać do **renowacji starych zdjęć**.

# 50. Własny film, czyli jak twórczo wykorzystać kamerę

**NA TEJ LEKCJI:**

- poznasz podstawy filmowania;
- dowiesz się, jak układać scenariusze do filmów.

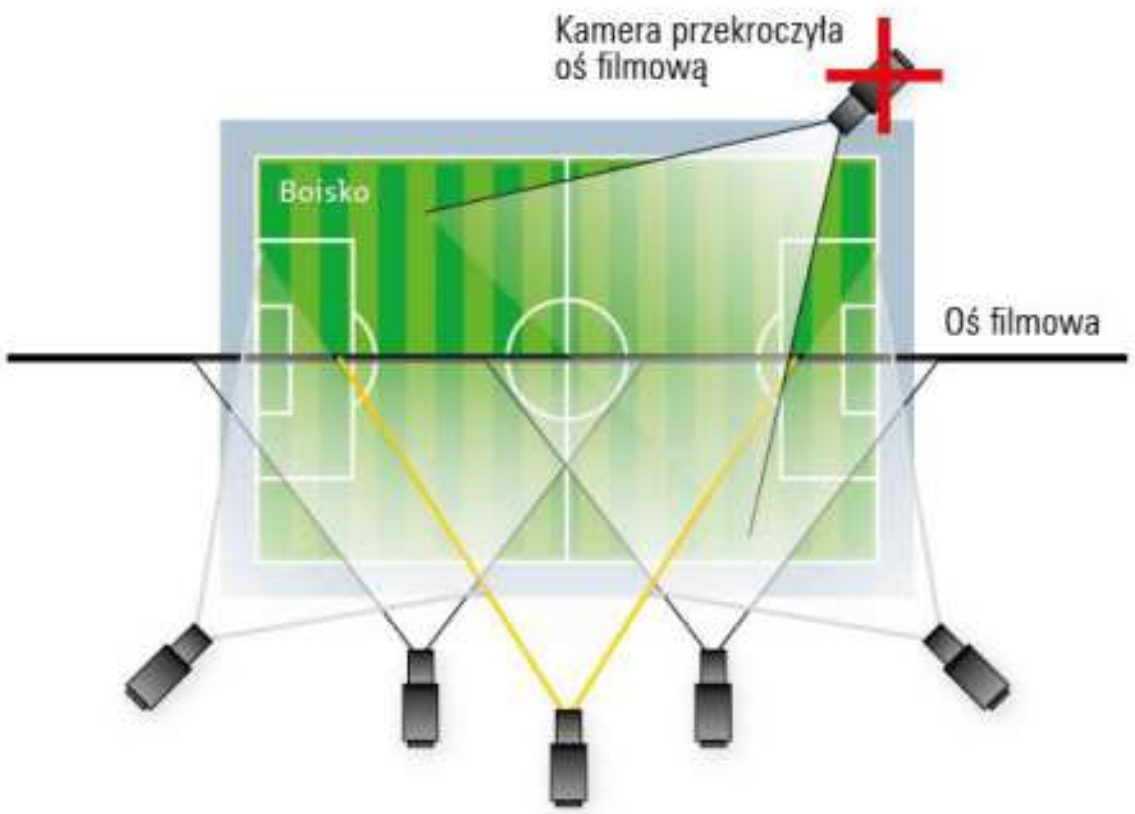
Fotografia przeszła wielką ewolucję od czasów szklanych negatywów do aparatów cyfrowych. Podobną ewolucję przeszły kamery filmowe i technika tworzenia ruchomych obrazów. Każdy smartfon może stać się narzędziem dla reportera lub filmowca amatora. Filmowałeś/filmowałaś swoim aparatem już nie raz, ale czy wiesz, dlaczego niektóre twoje filmy nie udały się tak, jak zaplanowałeś/zaplanowałaś? Podobnie jak w fotografii, także w filmie obowiązują pewne zasady. Warto je poznać, by móc później twórczo je stosować.

## 50.1. Ustawienia kamer

zasada nieprzekraczania osi filmowej

Rejestracja i późniejszy montaż ruchomych obrazów oprócz zachowania zasad kadrowania wymaga także dbałości o inne szczegóły techniczne. Jednym z nich jest zachowanie osi filmowej. Jeśli ją zignorujesz, widzowie mogą stracić orientację w przestrzeni twojego filmu. Przykładowo mogą nie zorientować się, kto kogo goni, kto na kogo patrzy lub z kim rozmawia. Wszystko wyjaśni przykład.

Wielokrotnie oglądałeś/oglądałaś relację z meczów piłkarskich. Ustawienie kamer nie jest przypadkowe.



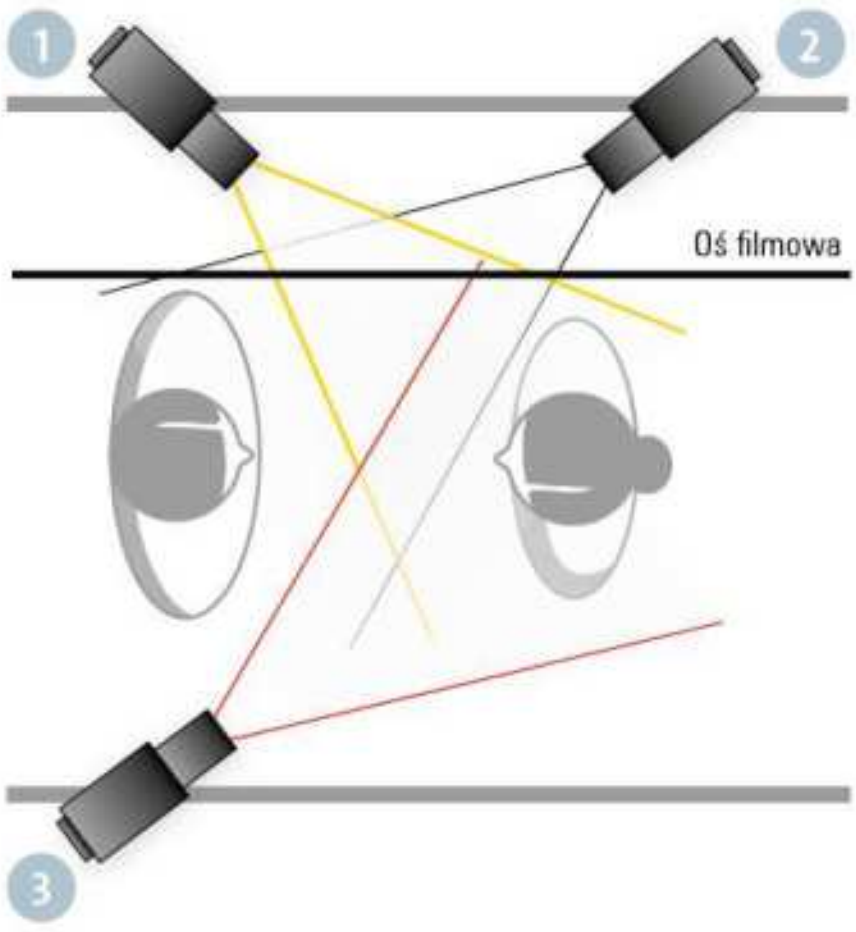

Rys. 50.1. Przykład rozstawienia kamer na meczu piłkarskim

Widz wie, w którą stronę atakuje dana drużyna. Jeśli zostanie zachowana zasada nieprzecinania osi filmowej, z punktu widzenia widza kierunek ataku dla danej drużyny jest stały bez względu na to, z której kamery ogląda obraz. Co stałoby się, gdyby oś została przekroczona (rys. 50.1.)? Obserwator odniósłby wrażenie, że obie drużyny strzelają do tej samej bramki.

**Uwaga!**

Obecnie prowadzi się transmisję za pomocą bardzo wielu kamer i niektóre są ustawione po przeciwnej stronie osi. Pokazują one tylko detale, np. nogi zawodnika kopiącego piłkę czy wyrzucającego ją z autu. Widz nie traci przez to orientacji.

Tę samą zasadę stosuje się przy filmowaniu dialogu dwóch osób (rys. 50.2.). Jeśli rozmowę pokazują na zmianę kamery 1 i 2 widz wie, z której strony stoi dana osoba i prawidłowo odczytuje scenę – rozmówcy stoją naprzeciwko siebie i prowadzą dialog. Jeśli tę samą rozmowę pokazywałyby kamery 2 i 3, widz miałby wrażenie, że obie osoby rozmawiają z kimś trzecim, patrząc w tę samą stronę, w tym przypadku w lewo.







Rys. 50.2. Oś filmowa podczas rejestracji sceny rozmowy dwóch osób

Można podać jeszcze inne takie przykłady – chociażby sceny pościgów, w których wyraźnie musi być widać, kto ściga, a kto ucieka. Przy złamaniu osi widz odniesie wrażenie, że biegną naprzeciwko siebie.

Podczas prowadzenia kamery i kadrowania oczywiście obowiązują także zasady dobrej fotografii. Są one często naginane przez wybitnych twórców dla uzyskania zaplanowanego efektu artystycznego lub narracyjnego. Jednak początkujący filmowcy powinni przestrzegać zasad, co pozwoli im uniknąć błędów.

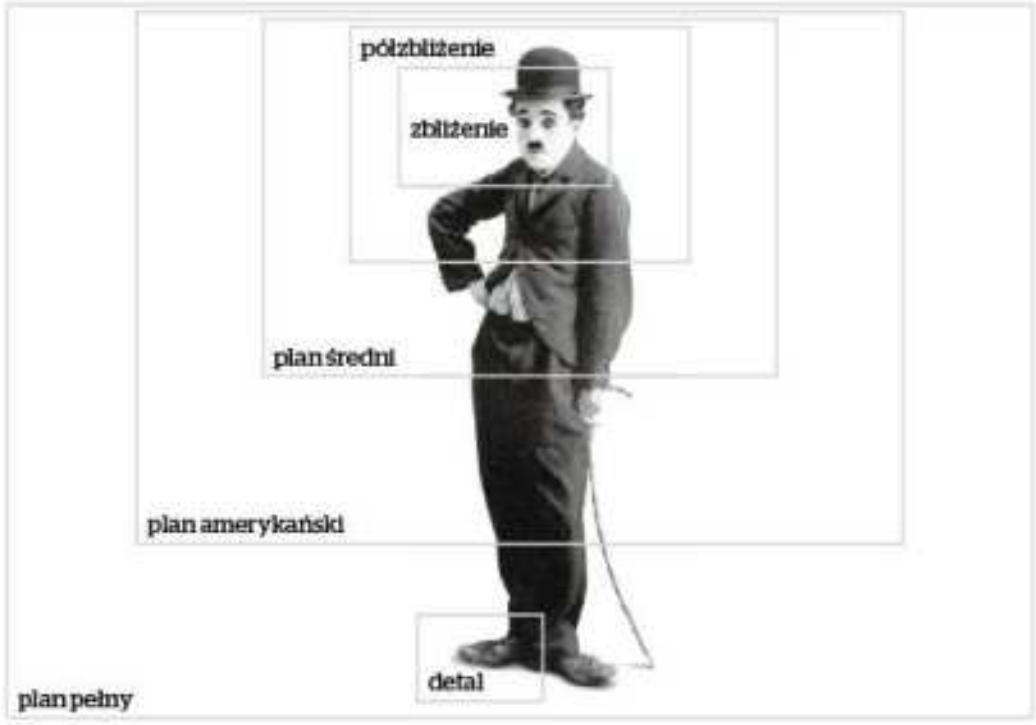
## 50.2. Własny tutorial, czyli filmujemy nie tylko kamerą

Zapewne często korzystasz z filmowych tutoriali zamieszczanych w sieci przez zawodowców i amatorów. Jak one powstają? Co jest potrzebne do ich zrealizowania? Okazuje się, że technicznie niewiele, a najważniejszy jest jak zwykle scenariusz. Nawet największe środki przeznaczone na film nie pomogą stworzyć dzieła na podstawie nieciekawego scenariusza.

◂ scenariusz

Prace nad własnym tutorialem rozpocznij więc od wymyślenia scenariusza. W przykładzie zajmiemy się prezentacją planów filmowych. Ich nazwy wchodzą w skład podstawowych pojęć, jakimi posługują się filmowcy (rys. 50.3.).

Oprócz planów z rysunku 50.3. istnieją jeszcze dwa szersze. W planie ogólnym widać okolicę, w której dzieje się akcja, a bohater może być rozpoznawalny, jednak nie dominuje. Plan totalny jest bardzo szeroki, a bohater praktycznie niewidoczny. Oba są używane do pokazania widzowi miejsca akcji.




Rys. 50.3. Podstawowe plany filmowe związane z bohaterem



Rys. 50.4. Plan ogólny



Rys. 50.5. Plan totalny

Klasyczny scenariusz składa się z trzech części.

**Początek**, inaczej zwany w filmie **ekspozycją**, prezentuje tło i czas akcji oraz postaci.

**Rozwinięcie akcji** – tu rozgrywa się fabuła, na przykład perypetie bohaterów.

**Zakończenie** jest rozwiązaniem akcji. Może być jednoznaczne lub pozostawiać widzowi swobodną interpretację. Jednak powinno doprowadzić do końca najważniejsze wątki.

nagrywanie scen ▸

Czy wszystkie części mogą wystąpić w scenariuszu samouczka? Wróćmy do przykładu. Jeśli jeszcze nie znasz nazw planów filmowych, będzie on dla ciebie ważny w kontekście tworzenia własnego filmu.

*Wstęp.*

*Autor–narrator przy biurku w typowym, uczniowskim pokoju.*

AUTOR

Cześć, dzisiejszy odcinek poradnika filmowca amatora poświęcimy planom filmowym. I nie chodzi tu o plany związane z produkcją, a o sposób pokazywania obiektów na ekranie. Czasami zdarza się wam mówić do operatora „Bliżej. Jeszcze. Za blisko."? Koniec z tym. Dziś poznacie pojęcia, które jednoznacznie określą plan ujęcia.

*Na ekranie szybko zmieniające się po kolei zdjęcia różnych planów filmowych tej samej sceny. Od najszerszych do najbliższych.*

AUTOR

Zdjęcia przedstawiają różne plany filmowe. Najszerszy pokazuje miejsce akcji, a najbliższy detal postaci. Zacznijmy od najszerszego.

*Na ekranie zdjęcie planu ogólnego.*

AUTOR

Taki plan nazywamy ogólnym. Widać na nim miejsce akcji, otoczenie bohatera. Bohater nie jest głównym obiektem na ekranie. Można jednak określić relacje pomiędzy nim a otoczeniem.

*Na ekranie zdjęcie planu pełnego.*

AUTOR

A to plan pełny. Widać całą postać lub obiekt.

*Na ekranie zdjęcie planu amerykańskiego.*

AUTOR

W planie amerykańskim postać jest pokazywana od głowy do kolan.

*Na ekranie zdjęcie planu średniego.*

AUTOR

Reporterów, dziennikarzy wiadomości, bohaterów w scenie, w której ważny jest ruch ciała, np. rąk, pokazuje się w planie średnim, czyli od pasa w górę.

*Na ekranie zdjęcie planu półzbliżenie.*

AUTOR

To jest półzbliżenie. W tym planie widać już emocje bohatera. Zaczynają być ważne umiejętności aktora.

*Na ekranie zdjęcie planu zbliżenie.*

AUTOR

Zbliżenie pokazuje obiekt lub aktora z niewielkiej odległości. Stawia wysokie wymagania przed aktorami.

*Na ekranie zdjęcie planu detal.*

AUTOR

Detal jest najbliższym planem. Za jego pomocą zwracamy uwagę na drobne elementy.

*Zdjęcia zebrane na jednym ekranie.*

AUTOR

Totalny widok na wszystkie plany. Dlaczego totalny? To nazwa najszerszego planu, pokazującego miejsce akcji, np. miasto w ujęciu z drona.

*Na ekranie narrator przy biurku.*

AUTOR

I to już wszystko. Od dziś, jeśli powiecie kamerzyście: teraz plan amerykański, będzie wiedział, o co chodzi. Oczywiście pod warunkiem, że też obejrzał mój film. Do zobaczenia w następnym odcinku samouczków dla amatorów filmowania.



Rys. 50.6. Przykładowe zdjęcia do filmu o planach filmowych

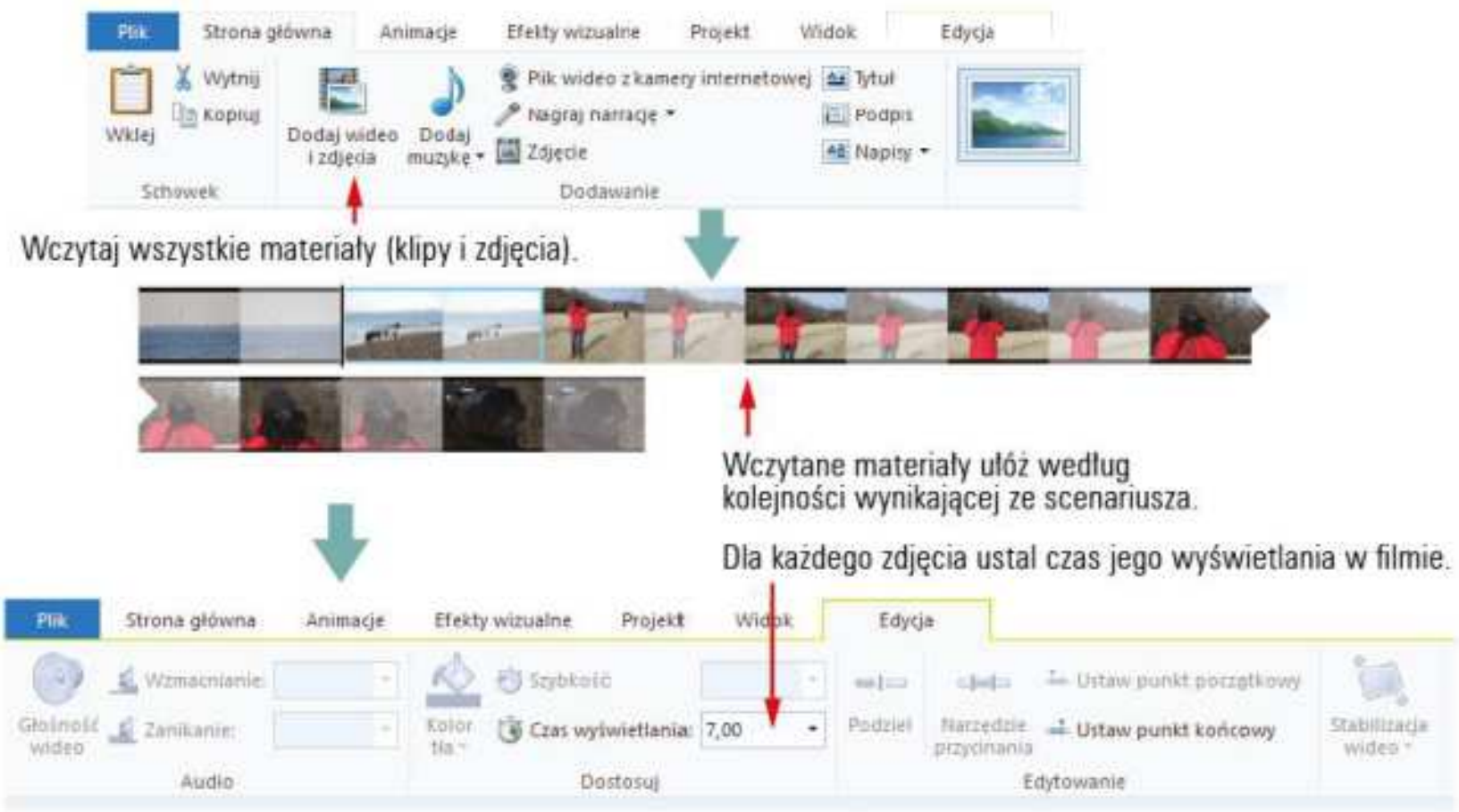
Jako ilustracji możesz użyć kolejno kadrowanego zdjęcia (rys. 50.6.).

Mamy więc **wstęp**, w którym autor wyjaśnia, o czym będzie film, **rozwinięcie**, czyli omówienie planów, i **zakończenie**.

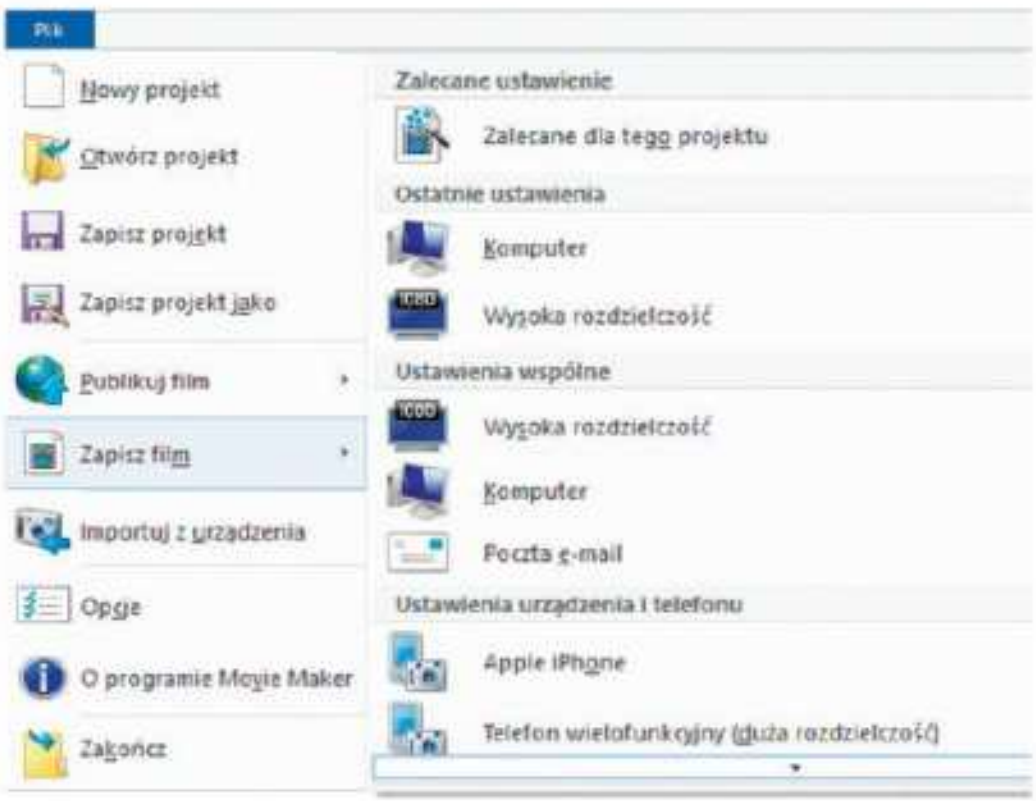
**sprzęt i narzędzia** ▸ Do zrealizowania tego filmiku potrzeba jeszcze oprogramowania do montażu, kamery i mikrofonu. Ponieważ film jest przeznaczony do internetu, a oglądany będzie na telefonach lub ekranach komputerów, wystarczą najprostsze narzędzia.

Dla systemu Windows 7 i 8 ogólnodostępny program do montażu to Microsoft MovieMaker. Obecnie nie jest on już dostępny w sklepie Microsoft, a jego pobranie z innych źródeł grozi zainfekowaniem komputera. W systemie Windows 10 użyj aplikacji **Zdjęcia**. Czas więc wziąć się do pracy.

**dźwięk filmu** ▸ Ustaw kamerę w taki sposób, by kadr odpowiadał założeniom scenariusza. Zarejestruj wszystkie sceny, w których występuje autor, a następnie dźwięk do podłożenia pod zdjęcia.



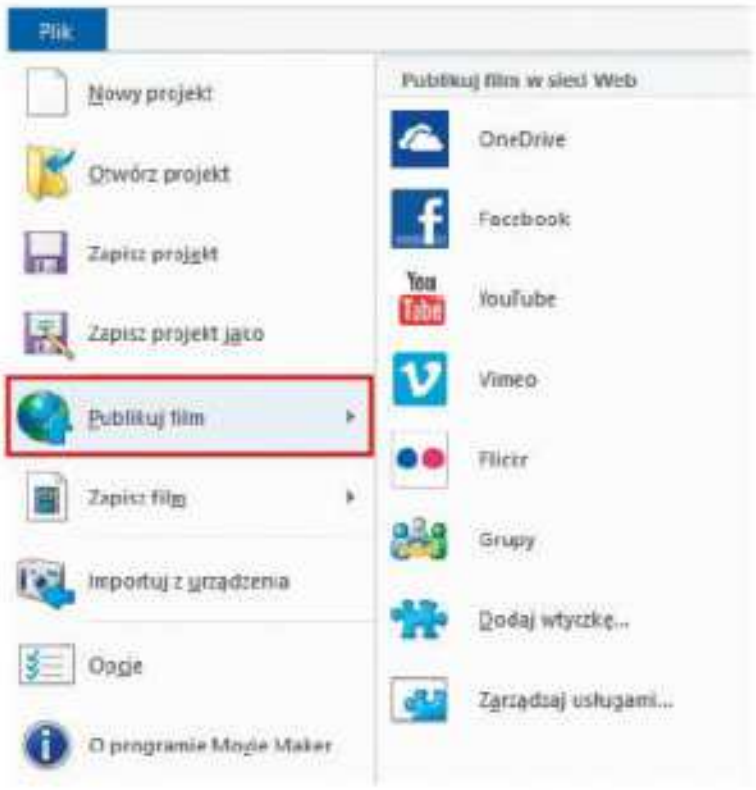

Rys. 50.7. Proces szybkiego tworzenia filmu w MovieMaker




Gdy film będzie gotowy, dodaj do niego napisy. Nie zapomnij wymienić autorów zdjęć i nazwisk uczestników produkcji.

Pozostaje już tylko zapisać film w postaci pliku (wyrenderować; rys. 50.8.) lub od razu upublicznić w portalach społecznościowych (rys. 50.9).

Rys. 50.8. Opcje MovieMakera do zapisu filmu w pliku




Rys. 50.9. Opcje programu MovieMaker do upublicznienia filmu w portalach społecznościowych

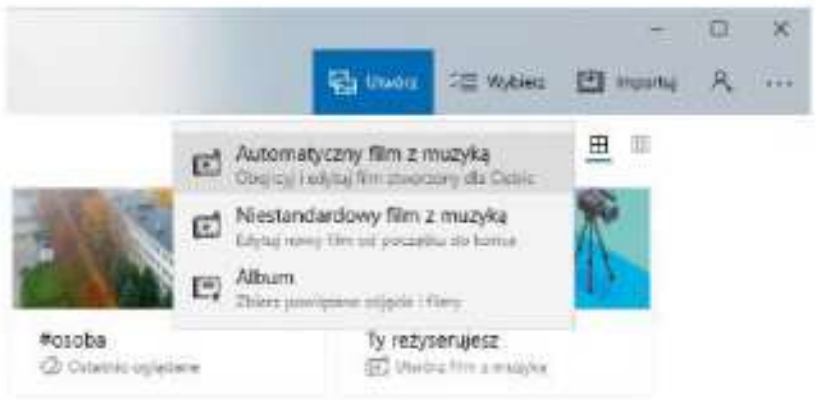
Aplikacja **Zdjęcia** (dostępna w Windows 10) jest również bardzo łatwa w obsłudze (rys. 50.10.).




Rys. 50.10. Edycja filmu w aplikacji Zdjęcia – przycinanie klipu

**Uwaga!**

Opcja **Zapisz projekt** nie eksportuje filmu w postaci gotowej do obejrzenia w odtwarzaczu. To plik, w którym program zapisuje dane potrzebne do odtworzenia projektu, czyli nazwy i miejsce pobrania klipów lub zdjęć, czas ich trwania, miejsca cięć itp. Film zostanie zapisany w pliku dopiero po użyciu opcji **Zapisz film** lub **Publikuj film.**




Rys. 50.11. Opcje szybkiego tworzenia filmów w aplikacji Zdjęcia

Aplikacja ta umożliwia dodawanie efektów 3D, na przykład fajerwerków, balonników i innych animacji nakładanych na film. Jeśli chcesz szybko zrobić film ze zdjęć i klipów (tak jak w ćwiczeniu z programem MovieMaker), użyj jednej z opcji menu **Utwórz** (rys. 50.11).

Zaawansowani amatorzy używają rozbudowanych edytorów. Są wśród nich także niekomercyjne, darmowe aplikacje, jak **Nuke** – edytor w wersji komercyjnej używany przez znanych producentów filmowych oraz **DaVinci Resolve** – program znanego producenta sprzętu studyjnego.

YouTube także udostępnia ciekawy i łatwy w obsłudze program do edycji filmu. Możesz w nim przycinać sceny, dodawać dźwięki, darmową muzykę, napisy itp. Program działa po przesłaniu klipów do chmury YouTube.

Jeśli udało ci się stworzyć ciekawy film, upublicznij go. Może stanie się przebojem internetu?

Istnieją także programy do rejestracji ekranu lub poszczególnych okien. Dzięki nim można tworzyć samouczki, na przykład o obsłudze któregoś z programów. Przykładem może być **Debut Video Capture Software** (rys. 50.12.). Może on także pobierać i rejestrować obraz ze źródeł zewnętrznych – kamery internetowej, sieciowej, kamery analogowej lub kamery USB.




Rys. 50.12. Okno programu Debut Video Capture Software

## ZADANIA DO WYKONANIA

1. W czasie oglądania filmu lub programu TV zwróć uwagę na pracę kamer. Analizuj kadrowanie, czas trwania ujęć, miejsce ustawienia kamery itp. Policz, ile kamer filmowało sceny. Wynotuj wszystkie spostrzeżenia i omów krótko zasadność użycia takich środków. Przedstaw własny pomysł na daną scenę.
2. Wyszukaj w sieci informacje o darmowych programach służących do rejestracji obrazu z ekranu komputera. Sprawdź działanie dwóch wybranych na podstawie opinii użytkowników lub testów w portalach branżowych. Opisz ich najciekawsze funkcje, np. efekt kliknięcia myszką, powiększanie.

3*. Ułóż scenariusz tutoriala (samouczka) wideo opisującego zakładanie konta pocztowego w dowolnej darmowej usłudze. Zrealizuj swój pomysł za pomocą programu do rejestracji ekranu komputerowego lub dowolnej kamery. Zmontuj i upublicznij swój film.

4. Odnajdź we własnych zasobach zdjęcia ze szkolnej wycieczki lub uroczystości. Ułóż scenariusz filmu, na który będą się one składały. Zanotuj, jak długo wybrane zdjęcia mają być widoczne na ekranie. Ułóż komentarze lektora. Za pomocą programu MovieMaker lub dowolnego innego edytora wideo zrealizuj ten film-prezentację. Postaraj się ułożyć zdjęcia w ciekawej kolejności. Zorganizuj pokaz. Pamiętaj, że upublicznienie filmu może odbyć się jedynie z poszanowaniem praw do wizerunku osób ze zdjęć.
5. Sprawdź działanie twojej kamery w różnych warunkach oświetleniowych – od zaciemnionego pomieszczenia do pełnego światła w słoneczny dzień. Oceń subiektywnie zarejestrowane obrazy. Zwróć uwagę na szumy, wyrazistość, kontrast i barwy. Oceń, przy jakim najmniejszym oświetleniu twoja kamera rejestruje akceptowalny co do jakości obraz. Użyj programów tematycznych. Zanotuj swoje spostrzeżenia. Będą przydatne podczas rejestracji ujęć filmu.

## PODSUMOWANIE LEKCJI

- Zachowuj **zasadę nieprzekraczania osi filmowej**. ◂ s. 291
- Nie zaczynaj pracy nad filmem, dopóki nie skończysz układać **scenariusza**. ◂ s. 293
- **Sceny** z aktorami rejestruj kilka razy. Za każdym razem staraj się uniknąć zrobionych wcześniej błędów. ◂ s. 294
- Mów naturalnie, ale głośno i wyraźnie, by mikrofon mógł zarejestrować dźwięk na odpowiednim poziomie. Wykonaj **próby dźwięku** dla wszystkich kamer i aktorów w celu wyrównania ich poziomu głośności. Będzie mniej pracy przy montażu. ◂ s. 296
- **Do rejestracji i montażu** możesz użyć dowolnego programu zgodnie z licencją. Na początku korzystaj z mało rozbudowanych aplikacji. ◂ s. 296

# 51. Ze stratą lub bez, czyli jak kompresować pliki

**NA TEJ LEKCJI:**

- dowiesz się, na czym polega kompresja;
- poznasz kodowanie arytmetyczne, LZW i RLE.

Na samym początku wyjaśnijmy, czym jest kompresja.

kompresja ▸ Polega ona na zmianie sposobu zapisu informacji tak, aby zmniejszyć redundancję, czyli nadmiarowość informacji (danych). Krótko mówiąc, kompresja to zmniejszenie ilości danych w taki sposób, aby podstawowe informacje zawarte w pliku pozostały, a jego objętość się zmniejszyła. Wyróżniamy dwa rodzaje kompresji – stratną i bezstratną. Właśnie nimi zajmiemy się w tym rozdziale. Działaniem przeciwnym do kompresji jest dekompresja.

## 51.1. Kompresja stratna

kompresja stratna ▸ Zadaniem kompresji jest zmniejszenie ilości miejsca zajmowanego przez dane lub zmniejszenie ilości danych przesyłanych przez łącza sieciowe. Kompresja stratna umożliwia zmniejszenie liczby danych potrzebnych do wyrażenia informacji w taki sposób, że nie jesteśmy w stanie odróżnić oryginału od pliku skompresowanego. W większości przypadków, poddając plik czynności odwrotnej, czyli dekompresji, nie odzyskamy wszystkich informacji. Natomiast dla niektórych danych algorytm kompresji stratnej może odtworzyć informacje niemal w sposób identyczny.

Kompresja stratna jest możliwa, ponieważ zmysły człowieka są niedoskonałe. Algorytmy podczas kompresji stratnej zazwyczaj posługują się modelami psychoakustycznymi lub psychowizualnymi, aby odrzucić informacje najmniej istotne dla ludzkich zmysłów.

Kompresję stratną stosuje się tylko do informacji odczytywalnych przez nasze zmysły. Nie wykorzystamy jej do plików wykonywalnych (programów uruchamianych przez komputer), które są odczytywane tylko przez maszyny.

Zazwyczaj kompresja stratna przydaje się do modyfikowania dźwięków, obrazów statycznych i ruchomych. Przy danych audiowizualnych algorytmy kompresji stratnej osobno poddają obróbce dźwięk i osobno obraz.

## Kompresja dźwięku

Do najpopularniejszych algorytmów stosowanych przy kompresji dźwięku należy kodowanie oparte na MDCT przy zapisie plików w formacie MP3. Kompresja stratna polega na redukcji informacji nadmiarowej w celu zmniejszenia przestrzeni pamięci potrzebnej do jej zapisania. Działa najczęściej przez usunięcie najmniej słyszalnych przez ludzkie ucho skrajnych tonów niskich oraz wysokich. Przy większych stopniach kompresji usuwa się również w muzyce ciche szczegóły. Dźwięk staje coraz mniej zbliżony do naturalnego.

Oprócz MP3 na MDCT jest oparty również format Ogg Vorbis.

## Kompresja obrazu

Najpopularniejszym algorytmem w kompresji obrazu jest JPEG, który został oparty na DCT – dyskretnej transformacji kosinusowej. Standardowy algorytm dzieli obraz na bloki o stałych rozmiarach. Następnie bloki te są poddawane transformacji, kwantyzacji (czyli nieodwracalnemu zmniejszeniu dokładności danych przez ograniczenie ich zbioru wartości) i kompresji bezstratnej.

## Kompresja wideo

Najpopularniejszy format wideo stosujący algorytm kompresji stratnej to DivX/XviD. Jest on zgodny z MPEG-4 i pozwala na zapisanie na standardowej płycie kompaktowej o pojemności 700 MB filmu o długości 90 minut.

MPEG to kompresja występująca w czterech odmianach (MPEG-1, MPEG-2, MPEG-3 i MPEG-4). Wykorzystuje ona różne metody kompresji, najczęściej bazujące na ograniczeniach postrzegania obrazu przez człowieka. Zaliczamy do nich kompresję ruchu czy kodowanie Huffmana.

## Kompresja bezstratna

kompresja bezstratna

Pliki poddawane kompresji bezstratnej mają mniejszą objętość przy zachowaniu tej samej ilości informacji. Metoda ta pozwala na odtworzenie (dekompresję) pliku do stanu sprzed skompresowania. W grafice na przykład każdy piksel po dekompresji ma tę samą wartość informacji.

## Kodowanie Huffmana

Kodowanie Huffmana zwane również metodą Huffmana jest jedną z najprostszych metod kompresji bezstratnej. Algorytm wykorzystywany podczas kodowania nie należy do najefektywniejszych obliczeniowo, dlatego też zazwyczaj jest używany w końcowym etapie kompresji. Skuteczność kodowania Huffmana wynosi 20–40% objętości danych. Działa ono niezależnie od typu danych. Kod Huffmana to kod prefiksowy, co oznacza, że żadne słowo kodowane nie jest początkiem innego słowa. Polega ono na przypisaniu znakom występującym

w najczęściej kodowanej wiadomości kodów krótkich. Znaki pojawiające się rzadziej otrzymują kody długie. Dzięki takiemu zróżnicowaniu możemy otrzymać kod o mniejszej liczbie bitów, niż gdybyśmy zastosowali kody o stałej długości. Kompresja w kodowaniu Huffmana polega na zastąpieniu symboli kodami.

## Kodowanie arytmetyczne

Kodowanie arytmetyczne, podobnie jak kodowanie Huffmana, opiera się na rozkładzie prawdopodobieństwa wystąpienia symbolu w strumieniu danych wejściowych. Każde słowo kodowe może być traktowane jako jednostronnie domknięty podprzedział przedziału [0, 1). Dzięki przypisaniu każdemu słowu kodowemu wystarczająco dużej liczby znaczących bitów, można odróżnić jeden podprzedział od innego, i w ten sposób zdekodować ciąg bitów, przypisując mu zbiór wiadomości wygenerowanych przez źródło.

## Kodowanie LZW

Kodowanie LZW (od nazwisk twórców Abrahama Lempela, Jacoba Ziva i Terry'ego Welcha, czytaj: ejbrahama lempela, dżejkoba ziwa, teriego łelcza) jest metodą strumieniowej bezstratnej kompresji słownikowej. Wykorzystywane jest w formacie graficznym GIF, formacie PDF i PostScript. Zmniejsza znacząco tylko rozmiary plików zawierające obrazy o dużych obszarach monokolorowych.

## Kodowanie RLE

Kodowanie RLE (Run-Lenght Encoding, czytaj: ran-lengt enkodin), czyli metoda kodowania długich ciągów, zazwyczaj jest wykorzystywana w takich formatach, jak: PCX, BMP, PDF. Polega ona na zastępowaniu serii takich samych znaków rekordem danych zawierających kod wzorca oraz liczbę jego wystąpień.

Załóżmy, że mamy następujące dane ciągi:

aaaaabbbaaaaaaaaaa – 5 pikseli czerwonych, 3 piksele niebieskie, 10 pikseli czerwonych

aaabbbbbbbbaaaabbb – 3 piksele czerwone, 8 pikseli niebieskich, 4 piksele czerwone i 3 piksele niebieskie

aaaaabbbaaabbbaaaa – 5 pikseli niebieskich, 3 piksele czerwone, 3 piksele niebieskie, 3 piksele czerwone, 4 piksele niebieskie

Bez kompresji – 3 * 18 bajtów = 54 bajty

Sposób kodowania: a5b3a10 – 6 bajtów, a3b8a4b3 – 8 bajtów, a5b3a3b3a4 – 10 bajtów – suma: 24 bajty.

Stopień kompresji 2:1

## ZADANIA DO WYKONANIA

1. Jaki rodzaj kompresji jest wykorzystywany w formacie PNG? Podaj wady i zalety tego formatu. Porównaj go z formatem JPEG.
2. Podaj wady i zalety metody Huffmana.
3. Do jakiego rodzaju kompresji zaliczysz formaty ARJ, RAR i ZIP? Wymień wady i zalety każdego z tych formatów.

## PODSUMOWANIE LEKCJI

- **Kompresja** polega na zmianie sposobu zapisu danych tak, aby zmniejszyć ich nadmiarowość (redundancję). Wyróżniamy dwa rodzaje kompresji – stratną i bezstratną. ◂ s. 300
- **Kompresja stratna** zmniejsza liczbę bitów, ale nie daje gwarancji, że podczas czynności odwracającej kompresję (dekompresji), odtworzona informacja będzie identyczna z oryginałem. ◂ s. 300
- **Kompresja bezstratna** jest metodą kompresji informacji w taki sposób, że podczas dekompresji jesteśmy w stanie odzyskać oryginalne informacje. ◂ s. 301

# 52. Klatki do komputera, czyli jak transmitować strumieniowo

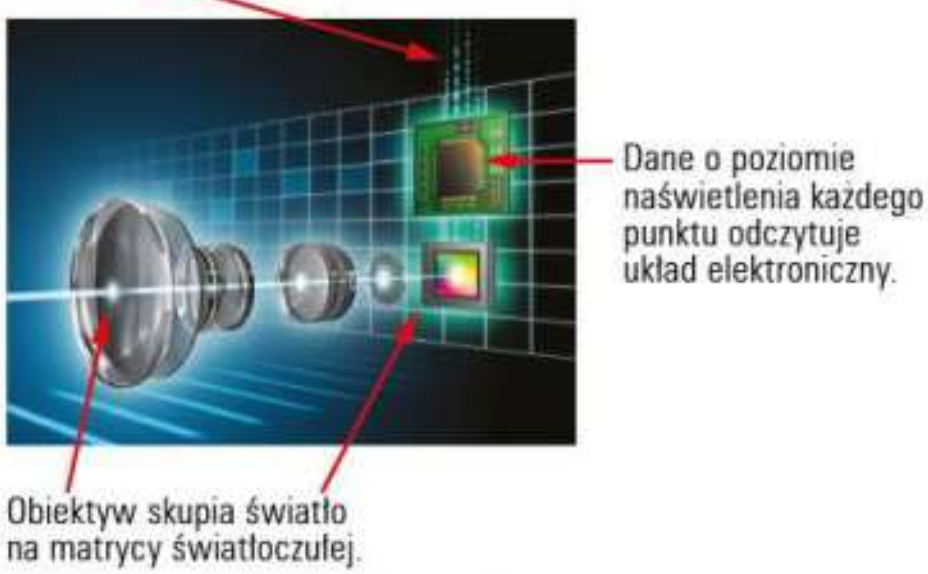
**NA TEJ LEKCJI:**

- dowiesz się, do czego służy strumieniowanie i czym jest transmisja strumieniowa;
- poznasz narzędzia służące do uruchomienia i odbierania transmisji strumieniowej i ich użyjesz

Podłączenie aparatu do komputera lub odczytanie zdjęć lub filmu z karty pamięci nie jest trudne. Umiesz już to zrobić. Jednak kamery mogą przesyłać obraz także na żywo do komputera, sieci lokalnej lub internetu. Można je wtedy podłączyć na kilka sposobów, wykorzystując łącza kablowe lub bezprzewodowe. Tak podłączona kamera może spełniać wiele funkcji, z których większość już znasz. Współczesna telewizja odbierana za pośrednictwem anteny działa na podobnej zasadzie. Strumień danych płynie z nadajnika i jest przetwarzany na obraz i dźwięk w odbiorniku.

## 52.1. Strumienie, czyli jak kamera przesyła obraz

**strumieniowanie** ▸ Czy wiesz, jak to się dzieje, że obraz z kamery można obserwować na ekranie komputera oddalonego o wiele kilometrów? Projektanci takich urządzeń użyli znanych rozwiązań – postanowili przesyłać dane o obrazie i dźwięku strumieniowo.




Rys. 52.1. Uproszczony schemat kamery cyfrowej

Kamerę tworzą trzy zasadnicze części: obiektyw, matryca światłoczuła i układ elektroniczny. Wpadające przez obiektyw światło dociera do matrycy, gdzie punkt po punkcie jest odczytywany stopień naświetlenia. Te dane pobiera sterujący matrycą układ elektroniczny. Po zebraniu danych z całej matrycy powstaje mapa bitowa klatki, czyli zdjęcie albo pojedyncza klatka filmu. Aby przesłać je do komputera, monitora lub sieci, należy postąpić podobnie. Dane każdego punktu klatki z pamięci

kamery są przesyłane po kolei – tworzą strumień. W urządzeniu wyświetlającym punkty zostają ułożone z zachowaniem miejsca z oryginału. Jeśli obraz ma być jedynie przeglądany, na przykład gdy oglądasz film z portalu kinowego, na tym proces się kończy. Jeśli jednak zdjęcie lub film ma być zachowany, komputer musi zapisać dane ze strumienia w pliku (rys. 52.1.).

◂ strumień danych

Przesłanie strumienia do komputera może nastąpić przewodowo lub bezprzewodowo. Najczęściej używanym złączem kablowym jest magistrala USB, w którą wyposażono każdy współczesny komputer. Połączenie bezprzewodowe może odbywać się za pośrednictwem karty sieciowej WiFi lub interfejsu Bluetooth.

Pobieranie plików z kamery przebiega standardowo, ponieważ system jej pamięć traktuje tak samo, jak inne pamięci masowe, na przykład dyski, karty SD, płyty, pendrive'y.

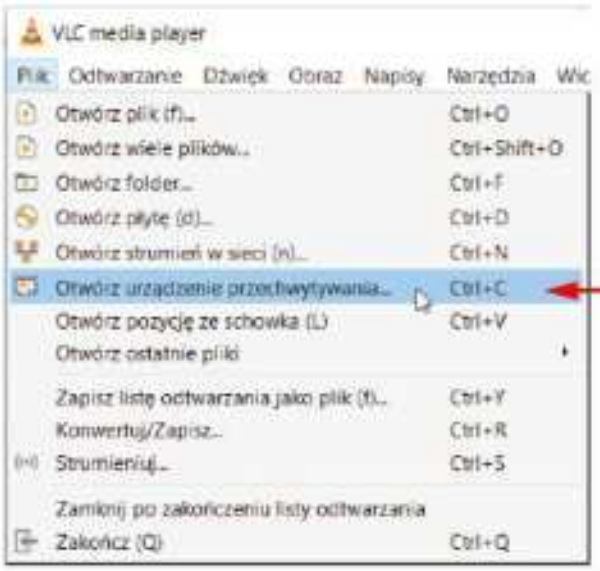
Przesyłanie strumienia z kamery do komputera wymaga użycia odpowiedniego programu lub opcji aplikacji, przykładowo komunikatora. Taki program może rejestrować obraz w pliku lub przesyłać go dalej, chociażby do internetu.

## 52.2. Na dysk, czyli jak zarejestrować obraz

◂ transmisja na żywo

Aparat fotograficzny lub kamera (np. w smartfonie) może rejestrować zdjęcia i filmy w pamięci i zapisywać je w pliku. Zdarza się jednak, że zachodzi potrzeba rejestracji obrazów z kamer pozbawionych takiej możliwości, na przykład kamer USB (także tych zainstalowanych w komputerach przenośnych). W takiej sytuacji należy użyć odpowiedniego programu – najlepiej skorzystać z aplikacji dołączonej przez producenta kamery lub z darmowych programów, jak VLC. Ciekawe możliwości VLC wykorzystamy w dalszej części rozdziału.

Jeśli do komputera dołączamy kamerę USB lub inne źródło obrazu i dźwięku (rys. 52.2.), musimy zadbać o prawidłowe zainstalowanie sterowników. Umożliwi to programowi odnalezienie źródła i wykorzystanie jego możliwości.




Wybierz tę opcję lub wciśnij **Ctrl+C**, aby wyświetlić i zarejestrować obraz z dźwiękiem z kamery USB.

Rys. 52.2. Opcje menu Plik programu VLC

Rejestrację z kamery lub innego urządzenia podłączonego do USB należy zacząć od wybrania odpowiedniego źródła (rys. 52.3.). Zauważ, że VLC (podobnie jak wiele innych odtwarzaczy) może odtwarzać multimedia z wielu źródeł.

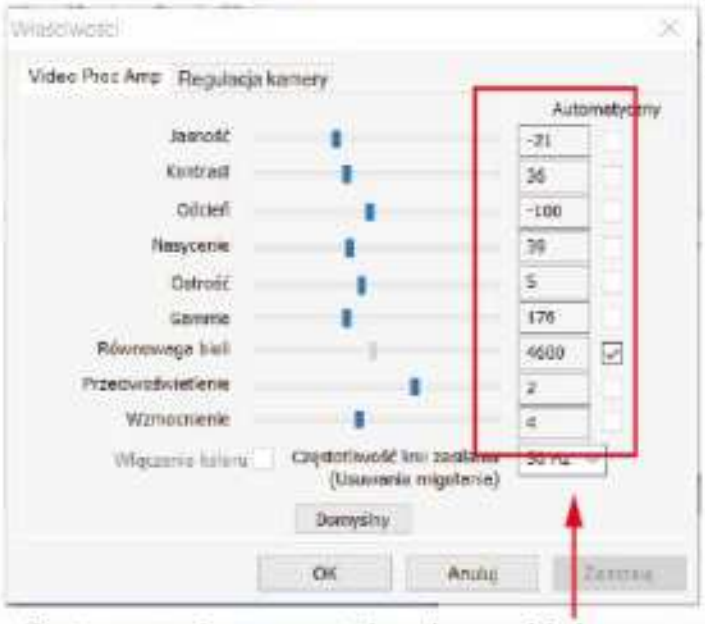
Jeśli do USB dołączono więcej źródeł, wybierz jedno z nich. W przeciwnym przypadku zostanie wybrane domyślne.

Źródło dźwięku nie musi pochodzić z kamery.

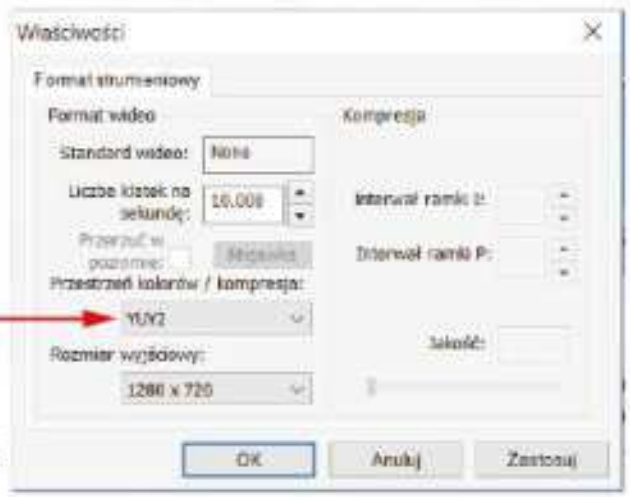
W opcjach zaawansowanych możesz ustawić m.in. proporcje obrazu, np. 4:3 lub 16:9, oraz liczbę klatek na sekundę dla twojej kamery, np. 30 kl./s.




Jeśli nie znasz parametrów urządzenia wejściowego, np. kamery, wybierz opcję **Właściwości urządzenia**. Część ustawień zostanie wpisana zgodnie z jego właściwościami.



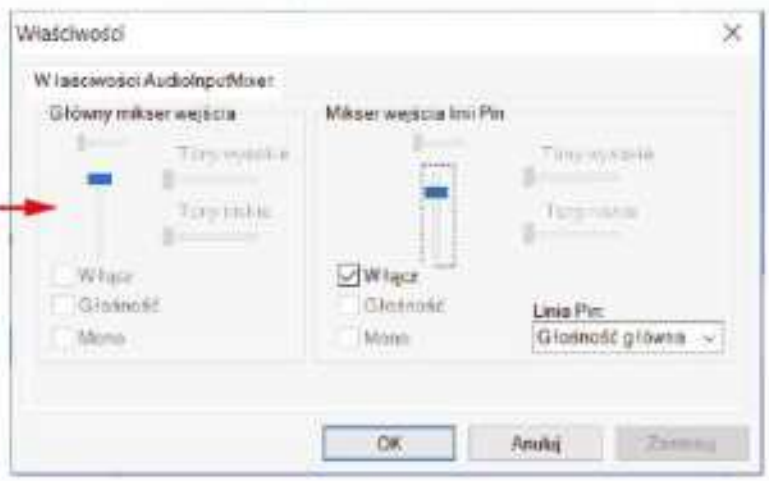

Dostosowanie parametrów obrazu. Możesz zaznaczyć opcję automatycznego ustawienia dla każdego z parametrów.




W kolejnym kroku ustawiania opcji zaawansowanych możesz ustawić parametry formatu strumieniowego. Z nimi będzie rejestrowany obraz i dźwięk.

VLC jest gotowy do wyświetlenia i rejestrowania obrazu ze wskazanego źródła.



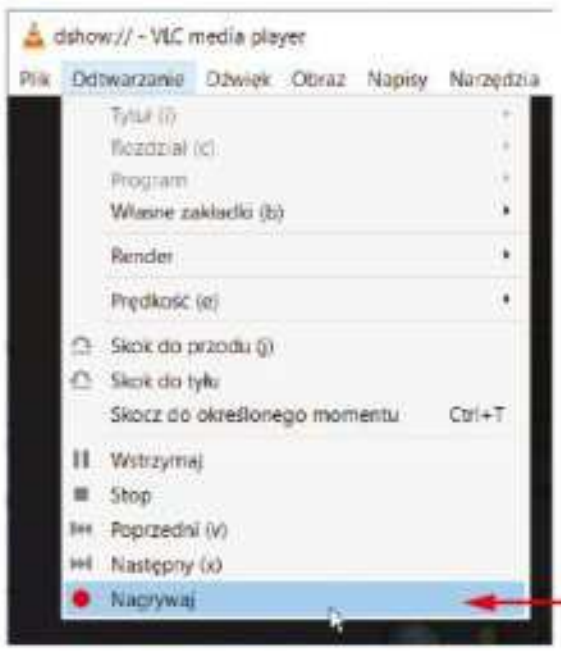



Rys. 52.3. Proces przygotowania programu VLC do rejestrowania obrazu i dźwięku z zewnętrznego źródła

Takie podłączenie kamery i ustawienie opcji służy do wyświetlania obrazu na dużym ekranie (np. za pomocą projektora) podłączonym do komputera. Można to wykorzystać w pracowni fizyki lub chemii do prezentacji przebiegu doświadczenia. Zdarza się, że z różnych przyczyn takie doświadczenia wykonuje się tylko raz, ponieważ obiekt w jego trakcie ulega zniszczeniu. Warto więc je zarejestrować. Zrobisz to za pomocą innej opcji VLC (rys. 52.4.). Plik z nagraniem trafi do folderu domyślnego dla filmów w Windows.

**Ten komputer/Wideo**

VLC zapisuje go w formacie AVI. Pliki zajmują więc sporo miejsca na dysku. Oczywiście można je przekonwertować na inny format, na przykład mp4. W tym także pomoże VLC.




Gdy obraz ze źródła pojawi się w oknie VLC, możesz nagrać go wraz z dźwiękiem za pomocą opcji **Nagrywaj**.

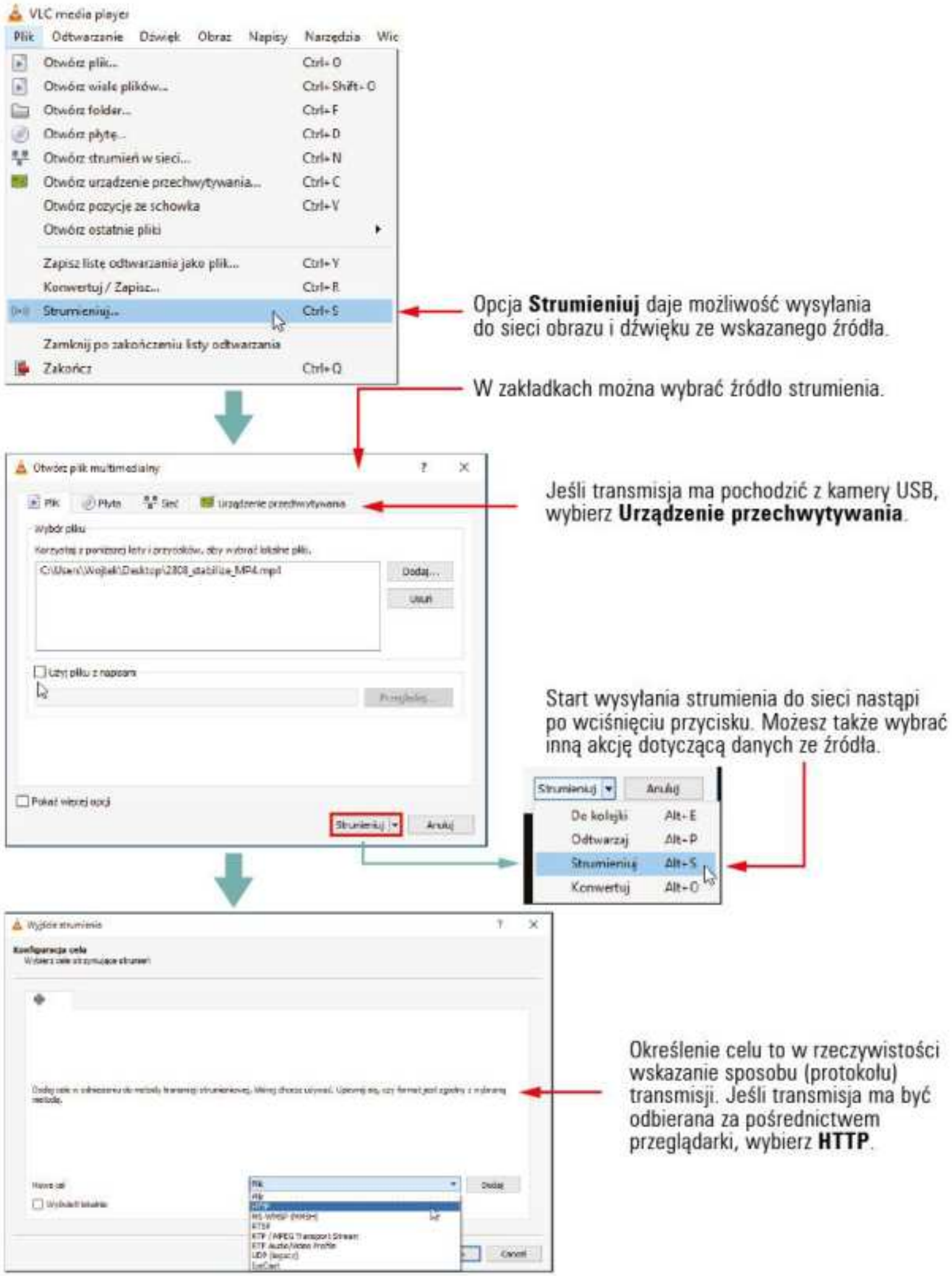
Rys. 52.4. Opcje odtwarzania i rejestrowania obrazu i dźwięku w VLC

## 52.3. Nadajemy, czyli jak uruchomić transmisję w sieci

Obraz i dźwięk można transmitować i nagrywać wprost z kamery USB za pomocą VLC lub innego programu o podobnych właściwościach (rys. 52.5.). Przykładem zastosowania transmisji w sieci lokalnej może być relacja na żywo z ważnej uroczystości szkolnej lub emisja filmu w salach lekcyjnych odbywająca się w tym samym czasie.

**Uwaga!**

Ćwiczenie możesz wykonać w pracowni, prowadząc transmisję pomiędzy komputerami – wykorzystaj do tego swój komputer z dwiema sesjami programu VLC.




Rys. 52.5. Opcja Strumieniuj w programie VLC

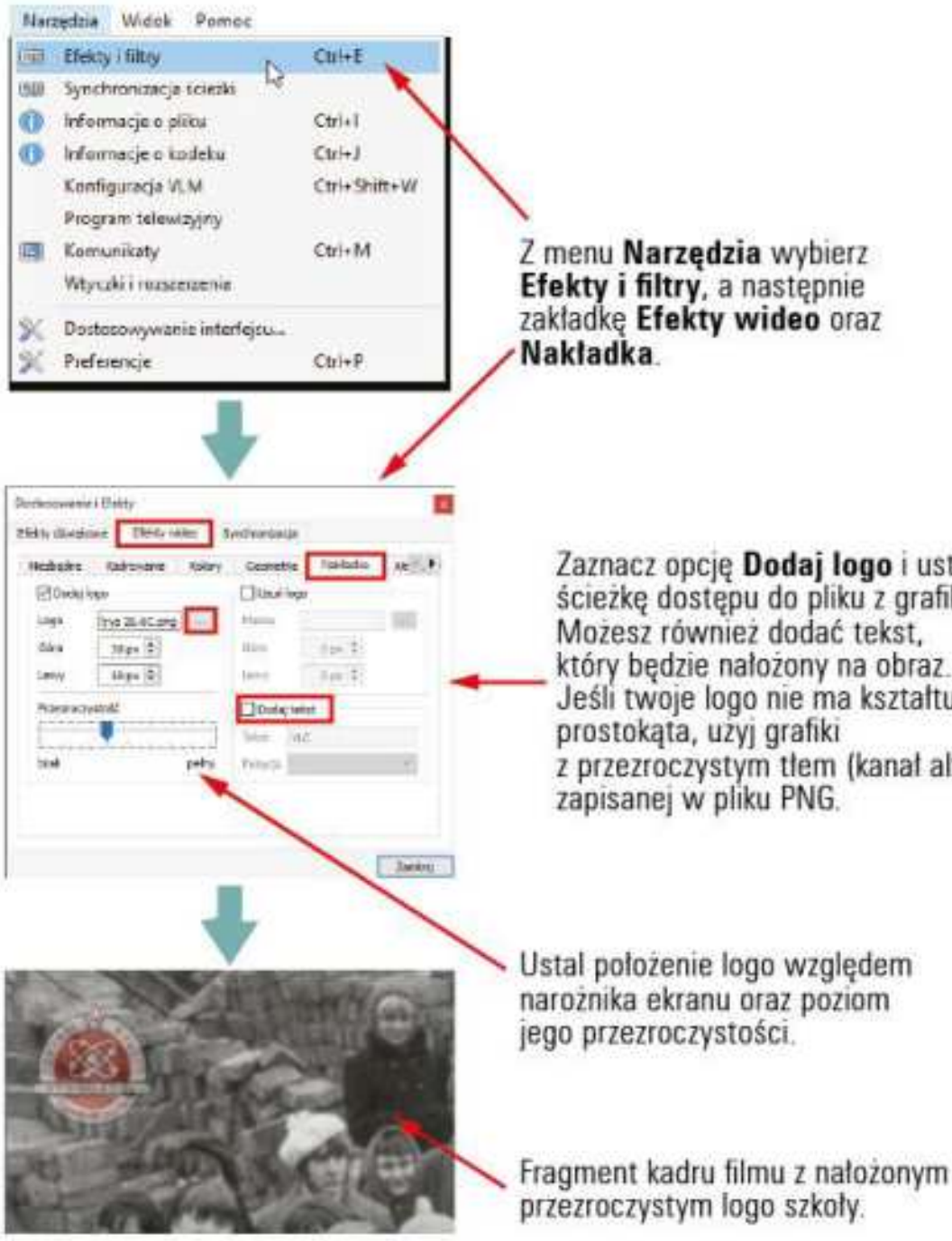
Jeśli chcesz swoją transmisję oznaczyć logo, jak na materiałach komercyjnych stacji telewizyjnych, wybierz opcję **Dostosowanie i efekty** (rys. 52.6.).

Jeśli chcesz wstawić logo do pliku z filmem, wykonaj czynności z rysunku 52.6. i użyj opcji **Strumieniowania do pliku**, wybierając opcję **Plik** w oknie konfiguracji celu (rys. 52.4.).

Odbieranie transmisji – w salach lekcyjnych lub w trakcie ćwiczenia na komputerach w pracowni – może odbywać się na kilka sposobów. Jeśli została wybrana transmisja HTTP, wystarczy uruchomić przeglądarkę i wpisać adres IP komputera strumieniującego w sieci oraz numer portu (domyślnie 8080). Przykładowa postać pełnego adresu wygląda następująco:

http://192.168.1.120:8080

W przypadku wysyłania strumienia za pośrednictwem innego protokołu po stronie odbiorczej będzie potrzebny program VLC lub inny odtwarzacz wyposażony w odpowiednią opcję.




Rys. 52.6. Nakładanie logo na strumieniowany obraz

## 52.4. Do portalu, czyli nadajemy z komórki

transmisja strumieniowa

Zapewne transmisje strumieniowe znasz z portali społecznościowych i informacyjnych. Każdy, kto ma smartfon podłączony do internetu, może nadawać własne relacje na żywo. Również w szkole przydałaby się taka relacja – transmisja z jakiejś uroczystości czy debaty wyborczej do samorządu uczniowskiego. W salach lekcyjnych lub indywidualnie na ekranach telefonów każdy zainteresowany mógłby obejrzeć transmisję. Można także zorganizować transmisję ważnej lekcji dla chorego ucznia, ale to jednak wymaga zgody nauczyciela. Do uruchomienia transmisji potrzebujesz na przykład aplikacji Facebook i połączenia z internetem. Transmisje strumieniowe są bardzo przydatne w różnych okolicznościach. Pamiętaj jednak, że twoi odbiorcy muszą o nich wiedzieć. Przed relacją powiadom o jej terminie jak największą liczbę znajomych i poproś ich, by rozesłali tę informację wśród innych osób. Gdy rozpoczniesz transmisję, twoi zalogowani do portalu społecznościowego znajomi zostaną o tym powiadomieni i mogą obserwować twój przekaz.

**Uwaga!**

Do transmisji strumieniowej za pośrednictwem telefonu komórkowego można wykorzystać różne aplikacje. Podczas ich doboru zwróć uwagę na bezpieczeństwo. Zanim je zainstalujesz, zbierz informacje i opinie na ich temat.

Podczas prowadzenia transmisji na żywo, pamiętaj o poszanowaniu prawa do wizerunku osób pokazywanych w trakcie strumieniowania.

## ZADANIA DO WYKONANIA

1. Odnajdź w sieci darmowe odpowiedniki programu VLC. Na podstawie opinii i opisów ich opcji oceń ich przydatność do transmisji strumieniowych. Porównaj ich możliwości z VLC.
2. Przeprowadź transmisję dowolnego pliku zawierającego film z dźwiękiem w sieci lokalnej pracowni komputerowej. Zwróć uwagę na odpowiednie dobranie portu. Zaproponuj odebranie transmisji koleżankom i kolegom z klasy. Zwróć uwagę na odpowiednie wybranie źródła. Podaj odbiorcom dane do odbioru transmisji i wytłumacz ich znaczenie.
3. Dowolny film, np. z wycieczki klasowej, opatrz swoim zdjęciem. Użyj odpowiednio przygotowanego swojego selfie. Dla lepszego efektu użyj kanału alfa dla przezroczystego tła. Zwróć uwagę na odpowiednie umiejscowienie logo. Zapisz strumień w pliku.
4. Przeprowadź transmisję na żywo, posługując się kamerką USB lub wbudowaną w komputer. Zwróć uwagę na odpowiednie wybranie źródła. Podaj odbiorcom dane do odbioru transmisji i wytłumacz ich znaczenie.
5. Wyszukaj w sieci informacje na temat strumieniowania w sieci dźwięków za pośrednictwem darmowego serwera radia internetowego. Zapoznaj się z możliwościami i zaletami takiego rozwiązania. Zastanów się nad tym, czego potrzebujesz do uruchomienia własnego radia internetowego. Pamiętaj o poszanowaniu praw autorskich.

## PODSUMOWANIE LEKCJI

s. 304 ▸ • **Strumieniowanie** pozwala przesyłać dane obrazu lub dźwięku za pośrednictwem sieci w taki sposób, by można je było odtwarzać w tym samym czasie po stronie odbiorcy. Jest to podobne do działania tradycyjnej telewizji cyfrowej odbieranej za pośrednictwem anteny.

s. 305 ▸ • Twój komputer może być zarówno nadajnikiem, jak i odbiornikiem **strumienia danych**. Strumienie mogą być nadawane w sieci lub zapisywane w pliku i odtwarzane po zakończeniu transmisji.

s. 305 ▸ • Większość smartfonów umożliwia **transmitowanie na żywo** za pomocą wbudowanej kamery i mikrofonu.

s. 309 ▸ • Do **transmisji strumieniowych** można użyć wielu różnych programów i aplikacji. Należy je wybierać po sprawdzeniu wiarygodności źródła i opinii o ich działaniu.

## PODSUMOWANIE PRZED SPRAWDZIANEM

### 40. Modele w przestrzeni, czyli podstawy druku 3D

- **Drukarka 3D** nakłada kolejno na siebie warstwy roztopionego **filamentu.** ▸ s. 222
- **Modele** (obiekty) do drukowania w trzech wymiarach tworzy się za pomocą **edytora trójwymiarowej grafiki wektorowej** lub poprzez **skanowanie 3D.** ▸ s. 222
- Opracowano wiele **formatów plików dla projektów 3D**. Należy zwrócić uwagę na to, z którymi z nich dana drukarka jest kompatybilna. Najpopularniejsze z nich to **STL i 3MF.** ▸ s. 223

### 41. Modelujemy, czyli jak projektować obiekty 3D

- Są **edytory 3D w chmurze**. Przykładem może być **Tinkercad**. ▸ s. 227
- **Pliki dla drukarki 3D** w formacie STL można przeglądać w **Paint 3D.** ▸ s. 230

### 42. Wizualizacja pomysłów, czyli projektujemy dom w edytorze 3D

- **Obiekty 3D** to nie tylko elementy do drukowania. Z modeli 3D są zbudowane m.in. gry komputerowe, a także animacje w filmach. ▸ s. 236
- Niektóre edytory, np. **SketchUp,** są połączone z **mapami elektronicznymi**. Dzięki temu można tworzyć modele obiektów architektonicznych i zaznaczać ich rzeczywiste lokalizacje. ▸ s. 236
- Edytory 3D są wyposażone w szereg **mechanizmów** ułatwiających **kreślenie brył** i **wypełnianie powierzchni.** ▸ s. 239

### 43. Urządzamy mieszkanie, czyli wizualizacje 3D

- Od lat 90. XX w. trwa intensywny rozwój **grafiki 3D**, która obecnie jest używana na szeroką skalę w technice i kinematografii. ▸ s. 245
- Razem z grafiką 3D zaczęło rozwijać się również **komputerowe wspomaganie projektowania w grafice 2D i 3D**. ▸ s. 245
- **Sweet Home 3D** jest darmowym programem służącym do projektowania wnętrz budynków. Pozwala on na projektowanie planów budynku, ustawianie mebli w gotowym już projekcie na planie 2D z podglądem 3D. ▸ s. 245

### 44. Wypukłości nie tylko 3D, czyli tworzymy elementy graficzne publikacji

- Umiejętność wykorzystania **warstw** pozwala na stworzenie zaawansowanych projektów graficznych. Dzięki **filtrom** można przekształcać elementy graficzne. Wypróbuj ich działanie z różnymi ustawieniami parametrów. ▸ s. 250
- **Efekt wypukłości** elementów graficznych uzyskuje się dzięki odpowiedniemu użyciu **filtrów** i **warstw.** ▸ s. 255

### 45. Dynamicznie, czyli jak tworzyć ruchome wizualizacje

- **Wizualizacja danych** to graficzne przedstawienie informacji, które zostały zestawione w pewnej schematycznej formie. ▸ s. 259
- **Rysunek techniczny** jest wizualizacją konwencjonalnie przedstawiającą plany, konstrukcje lub zasady działania przedmiotów mechanicznych itp. ▸ s. 260
- **Animacja** jest dziedziną nauki, techniką, która wykorzystuje ruchome obrazy w celu przedstawienia pewnego procesu w określonym czasie. ▸ s. 260
- **Proces tworzenia wizualizacji** w informatyce składa się z **czterech etapów**: modelowania, teksturowania, ustawiania sceny i renderowania. ▸ s. 261

### 46. Z wydruku do komputera, czyli skanujemy i odczytujemy dokumenty

- **OCR** – technologia zamiany skanowanych dokumentów na edytowalny tekst. ▸ s. 262
- **CCD**, **CIS**, **TWAIN** – pojęcia związane z działaniem skanera i jego użytkowaniem. ▸ s. 264

## 47. Dokumentujemy wydarzenia, czyli aktywna praca z aparatem fotograficznym

s. 272 • Optymalne naświetlenie matrycy może być wynikiem różnych kombinacji **czasu naświetlania i przysłony.**

s. 274 • Zdjęcia z aparatu możesz przesyłać do komputera za pośrednictwem **łączy bezprzewodowych Bluetooth lub WiFi, kablem USB** lub poprzez bezpośredni odczyt z wyjętej z urządzenia karty.

s. 276 • Jeśli nie masz doświadczenia lub pomysłu na skadrowanie zdjęcia, skorzystaj z **reguły trójpodziału.**

## 48. Szturmowiec w chmurze, czyli poprawiamy zdjęcia w edytorze grafiki rastrowej

s. 278 • Edytory grafiki rastrowej znajdziesz także w **chmurach informatycznych**.

s. 279 • Większość czynności podczas obróbki zdjęć (np. retuszowanie, zmiana barw) możesz wykonać za pomocą **niekomercyjnych edytorów grafiki rastrowej.**

## 49. Tego tu nie było, czyli poprawiamy rzeczywistość na fotografiach

s. 286 • Retusz i inne operacje na fotografiach najwygodniej wykonywać z wykorzystaniem **mechanizmu warstw.** Warstwy są jednym z najważniejszych mechanizmów tworzenia grafiki i montażu oraz retuszu zdjęć.

s. 288 • **Opcje narzędzi,** np. **wycinanie,** pozwolą na uzyskanie odpowiednich efektów, np. dodania obiektu do zdjęcia.

## 50. Własny film, czyli jak twórczo wykorzystać kamerę

s. 291 • **Przestrzegaj zasad,** np. nieprzekraczania **osi filmowej** i dobrego **kadrowania.**

s. 293 • Zanim zaczniesz filmować, poznaj **podstawy filmowania** i ułóż **scenariusz** filmu.

s. 296 • Wykorzystaj dostępne **narzędzia** i **sprzęt**.

## 51. Ze stratą lub bez, czyli jak kompresować pliki

s. 300 • **Kompresja** polega na zmianie sposobu zapisu danych, tak aby zmniejszyć ich nadmiarowość (redundancję).

s. 300 • Wyróżniamy **dwa rodzaje kompresji** – stratną i bezstratną.

s. 300 • **Kompresja stratna** zmniejsza liczbę bitów, ale nie daje gwarancji, że podczas czynności odwracającej kompresję (dekompresji) odtworzona informacja będzie identyczna z oryginałem.

s. 301 • **Kompresja bezstratna** jest metodą kompresji informacji w taki sposób, że podczas dekompresji jesteśmy w stanie odzyskać oryginalne informacje.

## 52. Klatki do komputera, czyli jak transmitować strumieniowo

s. 304 • **Strumieniowanie** pozwala przesyłać dane obrazu lub dźwięku za pośrednictwem sieci w taki sposób, by można je było odtwarzać w tym samym czasie po stronie odbiorcy.

s. 305 • Twój **telefon** lub **komputer** mogą zarówno **strumieniować** media, jak i je odbierać.

s. 305 • **Strumieniowane** dane można **rejestrować** w pliku.

# V. Komputer w sieci

- Działanie internetu
- Łączenie sieci z internetem
- Topologia sieci komputerowych
- Odczytywanie danych z karty sieciowej
- Konfigurowanie połączenia
- Parametry urządzeń peryferyjnych

# 53. Nie wszystko jest takie oczywiste, czyli jak działa internet

**NA TEJ LEKCJI:**

- dowiesz się, jak działa internet oraz czym są protokoły komunikacyjne.

Internetem posługujesz się na co dzień. Wiesz, jak działa? Odpowiedź nie jest taka prosta, jakby mogło się wydawać. Aby zrozumieć procesy składające się na funkcjonowanie internetu, należy odpowiedzieć sobie na kilka pytań: Skąd biorą się dane w sieci i gdzie są składowane? Jak je odnajdujemy i w jaki sposób trafiają do komputera? W jaki sposób miliardy komputerów mogą funkcjonować jednocześnie w jednej sieci? W tym rozdziale się tego dowiesz.

## 53.1. Szkielet, czyli łączymy sieć z internetem

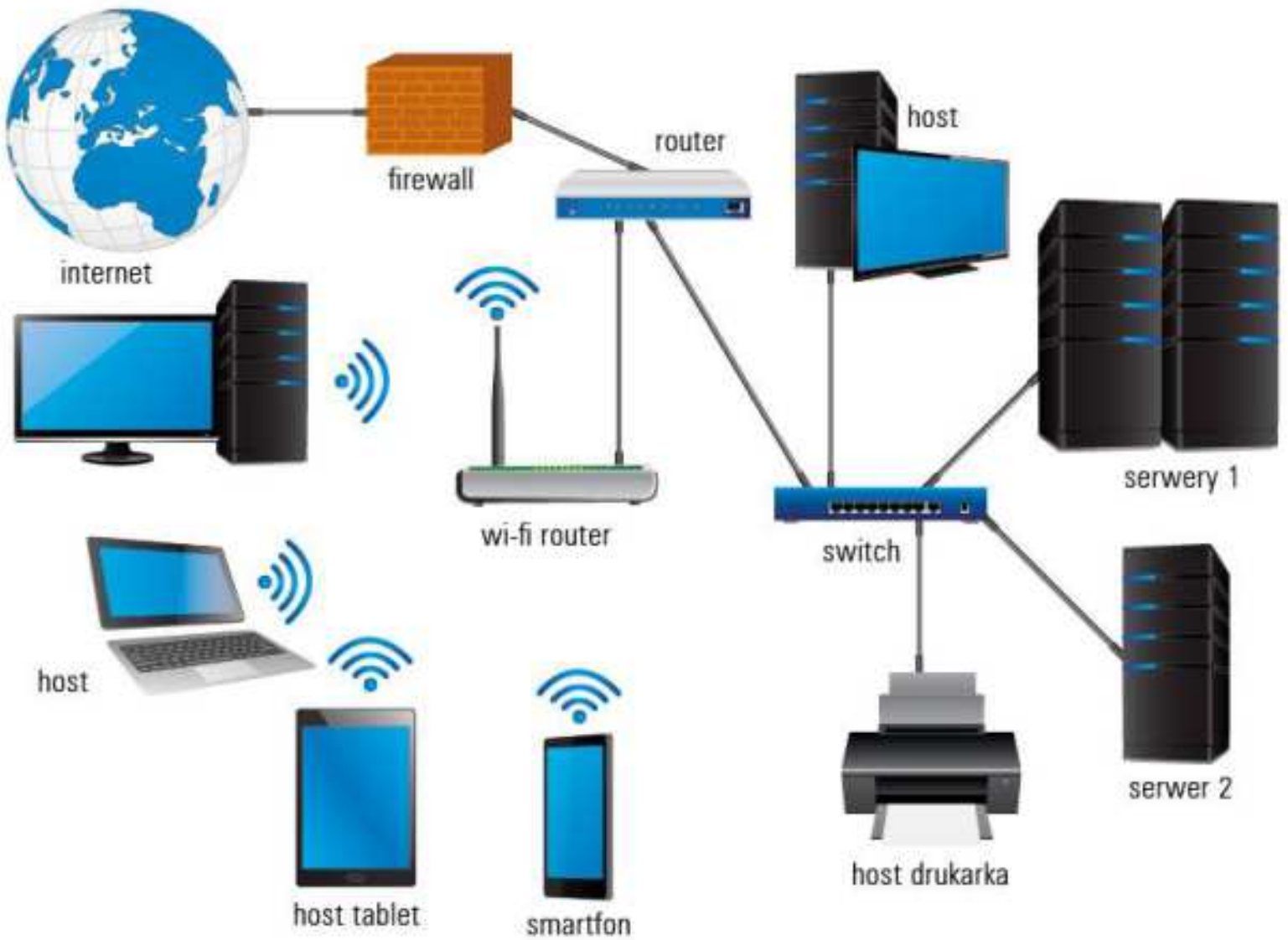
internet ▸

Ułożenie prostej definicji internetu nie sprawi ci kłopotu. Można powiedzieć, że **internet** jest siecią łączącą mniejsze sieci (rys. 53.1.), by możliwa była wymiana danych pomiędzy tworzącymi je komputerami i innymi urządzeniami komputerowymi (jak kamery, dyski sieciowe itp.).




Rys. 53.1. Uproszczony schemat połączeń w sieci internet

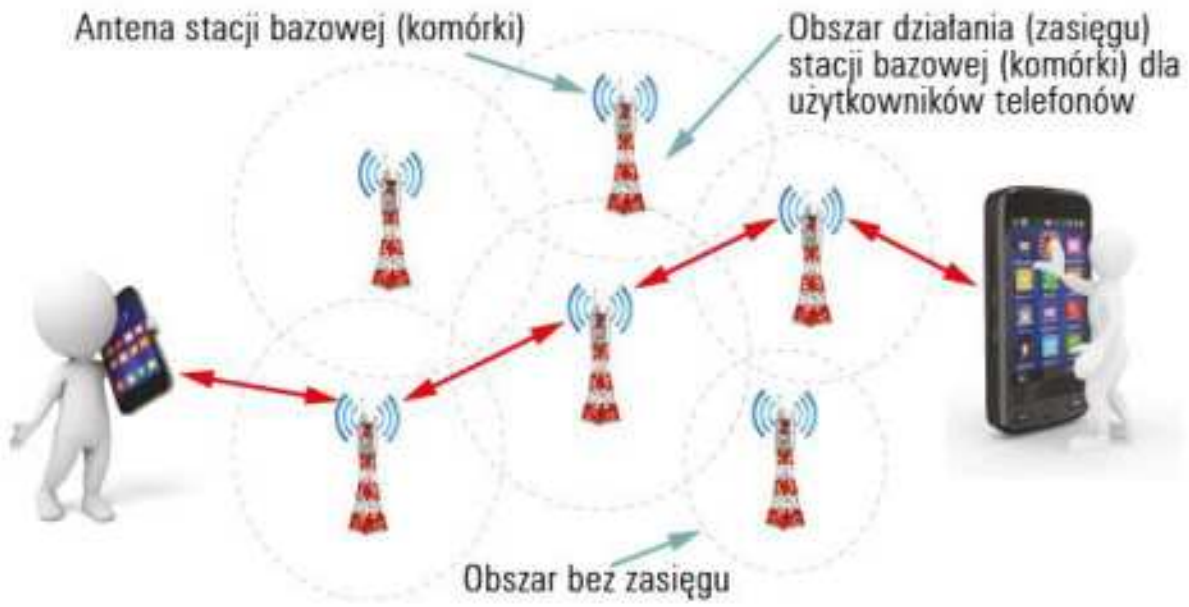
Czym są sieci, które w uproszczeniu nazwaliśmy mniejszymi? Domyślasz się zapewne, że mogą to być sieci o zasięgu lokalnym, a więc na przykład obejmujące dzielnicę, wieś, budynek (biurowiec, dom mieszkalny itp.), biuro, mieszkanie lub pokój. Skoro tak, to potrzebna jest infrastruktura i oprogramowanie. Małe sieci łączą swoje komputery (często nazywane klientami) za pośrednictwem **przełącznika** (**switcha**). Przełączniki mogą być podłączone do osobnego przełącznika, tworząc podsieci. W jakim celu? Po pierwsze – takie rozwiązanie pozwala oddzielić od siebie różne części sieci, którymi łatwiej zarządzać. Po drugie – w ten sposób zostaje uproszczona infrastruktura. Można także utworzyć podsieć, w której pracują serwery, co ułatwi zapewnienie im bezpieczeństwa. Przełącznik łączący podsieci może być połączony z **routerem**, za pośrednictwem którego cała sieć uzyska połączenie z internetem (rys. 53.2. na stronie obok).




Rys. 53.2. Przykładowy schemat sieci lokalnej LAN

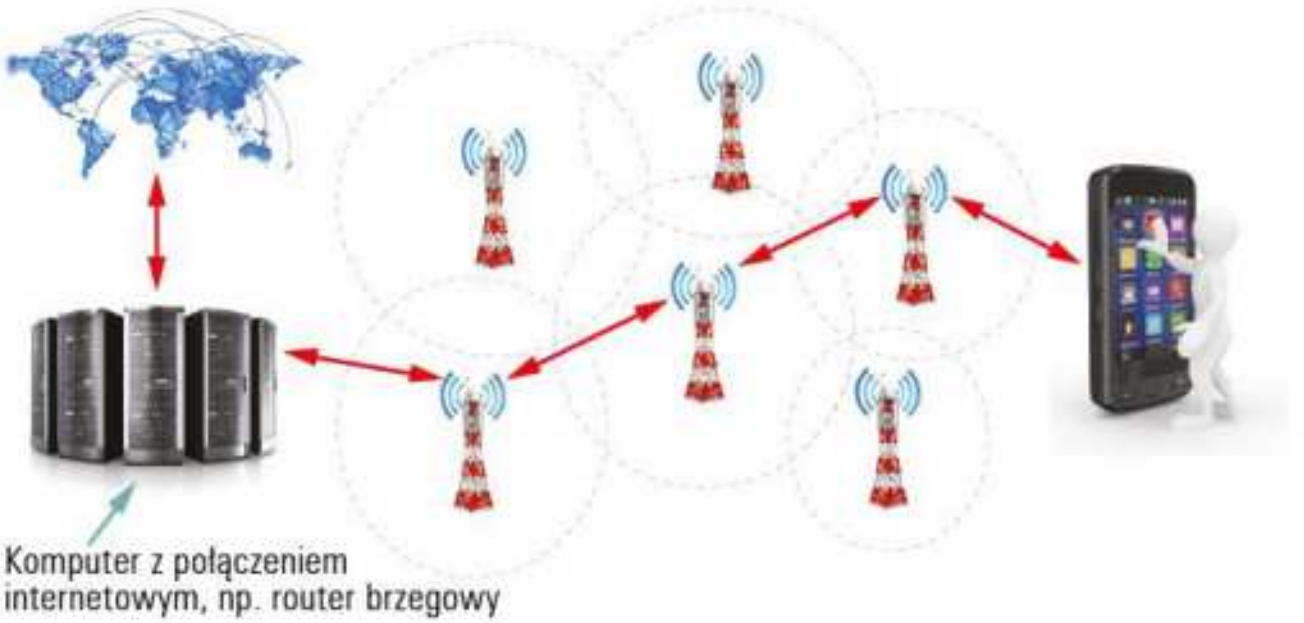
Czy wiesz, czym różni się router od przełącznika (switcha)? Router odpowiada za łączenie ze sobą różnych sieci, na przykład internetu z siecią lokalną w małym biurze lub podsieci w przedsiębiorstwie. Przełącznik łączy ze sobą poszczególne urządzenia w sieci, na przykład komputery czy drukarki. Dzięki niemu można tworzyć sieci lokalne, które mogą ze sobą współpracować dzięki routerom. Oba urządzenia działają w różnych warstwach modelu ISO/OSI, o którym przeczytasz w rozdziale 56.

Wiesz już, jak łączy się w sieć komputery i inne urządzenia. Medium transmisyjnym może być kabel miedziany, światłowód lub połączenie bezprzewodowe WiFi. Taki model sieci komputerowej nie obejmuje połączeń za pośrednictwem sieci telefonii komórkowej. Jeśli łączysz się z innym telefonem, twój smartfon nie musi korzystać z internetu. Jednak z racji tego, że transmisja rozmowy to w rzeczywistości przesyłanie danych cyfrowych, wykorzystywane są te same stacje przekaźnikowe (rys. 53.3.).




Rys. 53.3. Połączenie w trakcie rozmowy w telefonii komórkowej

Podczas połączeń smartfona z internetem ta sama sieć radiowa łączy go z centrum komputerowym pełniącym funkcję routera brzegowego. Takie urządzenia pracują na granicy między siecią komórkową danego operatora a internetem (rys. 53.4.). Niektóre współczesne stacje przekaźnikowe także są wykorzystywane jako routery.




Rys. 53.4. Przykładowe połączenie pomiędzy smartfonem a internetem w sieci komórkowej

## 53.2. Zasady ruchu, czyli protokoły w internecie

Połączenia internetowe pomiędzy komputerami muszą zapewniać nie tylko bezpośrednią łączność, lecz także sprawić, by komputery wzajemnie się rozumiały. To możliwe dzięki **protokołom komunikacyjnym**, czyli ściśle ustalonym i wspólnym w całej sieci zasadom przesyłania informacji i nawiązywania połączeń. Opisują one zarówno zagadnienia informatyczne, jak i elektryczne, wyznaczają parametry dla konstruktorów urządzeń sieciowych i oprogramowania.

Urządzenia sieciowe realizują je automatycznie. W internecie funkcjonuje wiele protokołów komunikacyjnych, najważniejsze to:

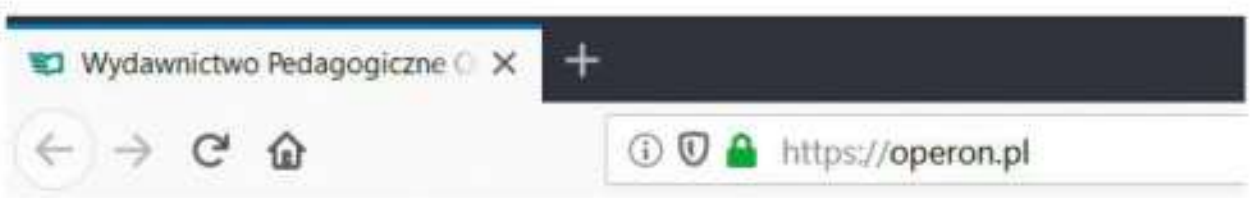
◂ IP i TCP

**IP** (Internet Protocol, czytaj: internet protokol) – odpowiedzialny za wybór trasy i przesłanie informacji w postaci pakietów. Został on utworzony w taki sposób, by w razie awarii automatycznie zestawił alternatywne połączenie.

**TCP** (Transmission Control Protocol, czytaj: transmyszyn kontrol protokol) – zapewnia bezpieczne przesyłanie danych, potwierdza odebrane dane i zapewnia kontrolę poprawności ich przesyłania. Dzięki niemu możesz bez błędów przeglądać strony internetowe, przesyłać pocztę, przesyłać pliki itp.

Oba protokoły są wykorzystywane przez inne, bardziej wyspecjalizowane:

**HTTP** (Hypertext Transfer Protocol, czytaj: hajpertekst transfer protokol) lub **HTTPS** (Hypertext Transfer Protocol Secure, czytaj: sekjur) – umożliwia przesyłanie i wyświetlanie stron internetowych, czego ślad widać w pasku adresów przeglądarki, w których pokazuje się adres poprzedzony nazwą protokołu (rys. 53.5.). Odmiana HTTPS pozwala na szyfrowanie połączenia. Dzięki temu łączenie się ze stronami (zakładanie kont, logowanie się na własne profile, na przykład konta bankowe, aukcje, sklepy, pocztę) staje się bezpieczniejsze.




Rys. 53.5. Nazwa protokołu przed adresem strony

**FTP** (File Transfer Protocol, czytaj: fajl transfer protokol) – służy do przesyłania plików. Korzystają z niego programy do zarządzania zawartością dysków sieciowych, w tym także przeglądarki internetowe (rys. 53.6.).




Rys. 53.6. Przykłady działania protokołów dla tego samego adresu

**SMTP** (Simple Mail Transfer Protocol, czytaj: simpyl mejl transfer protokol) – został stworzony do przesyłania poczty elektronicznej.

Warunki i zasady opisane w specyfikacji protokołów obowiązują w całym internecie. Dzięki temu nie ma problemu z kompatybilnością. Przeglądarka i klient usług internetowych akceptują je bez względu na zainstalowany system operacyjny.

## 53.3. Komputery ponumerowane, czyli adresy w internecie

Wiesz już, że komputery w sieciach mają swoje adresy IP. Adresy IP w sieci mają hosty. Host to komputer lub urządzenie sieciowe. Jednak w sieci pracuje niezliczona ilość urządzeń, czy zatem wystarczy dla wszystkich adresów? Przede wszystkim internetowe adresy IP są przypisane jedynie urządzeniom podłączonym bezpośrednio do internetu. Na przykład szkolna pracownia może być włączona do internetu za pośrednictwem modemu z routerem, a w sieci globalnej cała będzie funkcjonować pod jednym adresem przydzielonym modemowi (adres publiczny). Komputery wewnątrz sieci dostaną adresy przypisane przez router (adresy prywatne). Twój komputer może mieć taki sam adres, jak inny w sąsiedniej pracowni obsługiwanej przez inny router (pracujący w innej podsieci).

protokół DHCP ▸ Jak komputer uzyskuje IP? W twojej sieci lokalnej odpowiada za to protokół **DHCP** (Dynamic Host Configuration Protocol, czytaj: dajnamik hołst konfigurejszyn protokol), czyli usługa działająca w urządzeniu zarządzającym siecią. To dzięki niemu komputer podłączony do sieci pobiera niezbędne dane ustawień parametrów, które odpowiadają za poprawną pracę. Wśród nich jest także adres IP.

adresy dynamiczne i statyczne ▸ Urządzenie zarządzające siecią może przyznawać **adresy dynamicznie** – za każdym razem, gdy host loguje się do sieci, może otrzymać różne IP, najczęściej pierwsze wolne. IP w sieci może być także **statyczne**, czyli przydzielone danemu hostowi na stałe. Za każdym razem, gdy loguje się on do sieci, otrzymuje ten sam adres.

A kto przyznaje adresy urządzeniom sieciowym w internecie? Aby zachować zasadę, że każdy z adresów jest niepowtarzalny, zarządza nimi instytucja o światowym zasięgu – The Internet Corporation for Assigned Names and Numbers (czytaj: de internet korporejszyn for asajnd nejms end nambers) (ICANN). Robi to za pośrednictwem organizacji regionalnych; na terenie Europy jest to Réseaux IP Européens (czytaj: rezu ipe europeą) (RIPE). Te z kolei przydzielają adresy instytucjom zarządzającym w poszczególnych krajach. W Polsce adresy dostawcom internetu nadaje Naukowa i Akademicka Sieć Komputerowa (NASK).

ICANN ▸




Rys. 53.7. Instytucje czuwające nad przyznawaniem IP w internecie

Strony internetowe i usługi mają więc adres komputera, na którym je zainstalowano. Taki komputer nazywamy **hostem**. Udostępnianie przez dostawcę miejsca na usługi internetowe i strony to **hosting**.

Protokół IP zestawia połączenia między hostami i może je zmieniać (tak samo jak w przypadku wystąpienia zakłóceń lub awarii). Zdarza się, że podczas wczytywania strony internetowej dane biegną do nas inną drogą. Taki proces nazywamy routingiem dynamicznym wyboru trasy. Czy możliwe jest śledzenie takiego połączenia? Wypróbuj dwa sposoby.

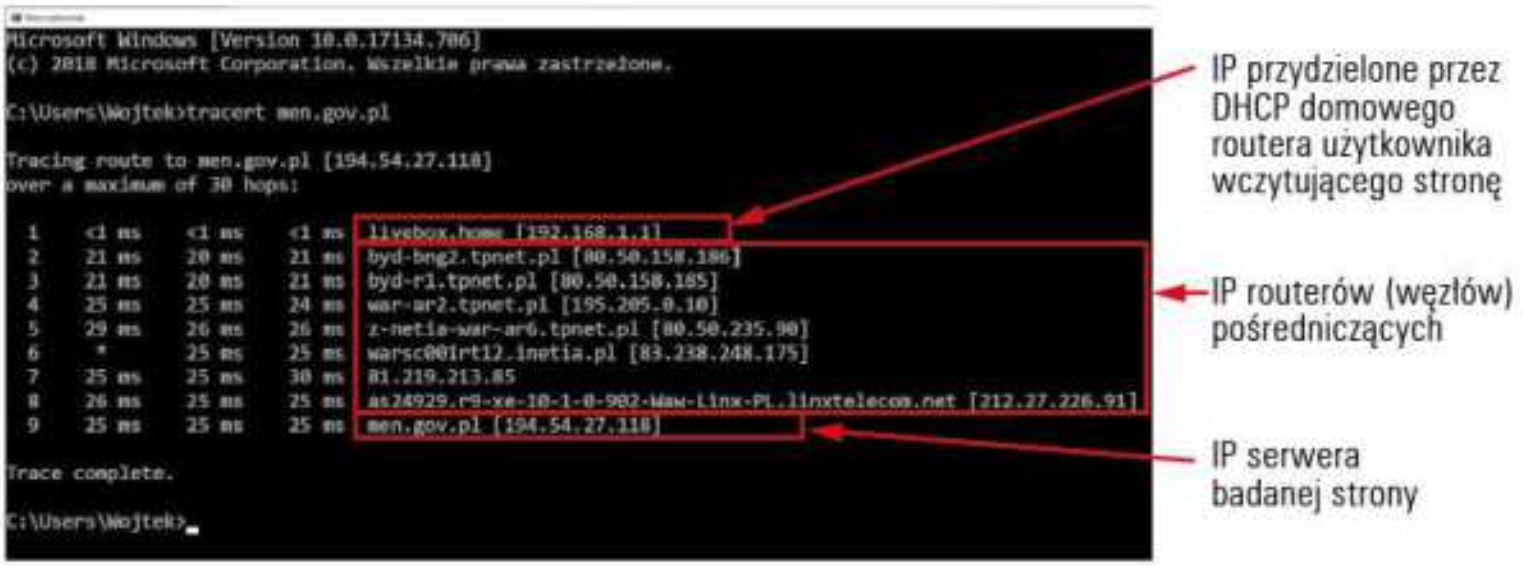
Uruchom **Wiersz polecenia** (**CMD**) systemu Windows (rys. 53.8.). Następnie wpisz polecenie **tracert** i adres strony, do której drogę chcesz prześledzić. **Tracert** wyświetla informacje o trasie i punktach węzłowych (routerach), przez które muszą przejść pakiety danych, by dotrzeć z serwera do twojego komputera. Polecenie to wyświetla także czas, w którym pakiety przebywają dany odcinek. Jako pierwsze zostaną wyświetlone węzły najbliższe twojemu komputerowi (rys. 53.9.).

◀ polecenie tracert




W opcji wyszukiwania wpisz **cmd**, a następnie kliknij w ikonę odnalezionej aplikacji.

Rys. 53.8. Uruchamianie Wiersza polecenia w systemie Windows 10




Rys. 53.9. Efekt działania polecenia tracert na przykładzie strony men.gov.pl

Drugim sposobem jest użycie jednego z programów, jak **VisualRoute** (rys. 53.10.). Dane są przedstawione w formie wykresu. W wersji komercyjnej można również śledzić połączenia na mapie świata. Zauważ, że oba sposoby wskazały inne połączenie. Nie jest to błędne wskazanie, lecz efekt działania protokołu IP, który zestawił połączenia.




Rys. 53.10. Śledzenie połączenia z serwerem strony internetowej za pomocą programu VisualRoute

W obu przypadkach końcowym IP jest 194.54.27.118, ponieważ pod tym adresem umieszczono daną stronę.

## 53.4. Zrozumiałe dla użytkownika, czyli adresy symboliczne

W ćwiczeniu obok IP wyświetlano też nazwy, które łatwo zidentyfikowałeś/zidentyfikowałaś jako adresy internetowe. Można więc uznać, że IP w postaci czterech liczb oddzielonych kropkami są z nimi tożsame. Sprawdź to na przykładzie badanej przez ciebie strony – men.gov.pl.

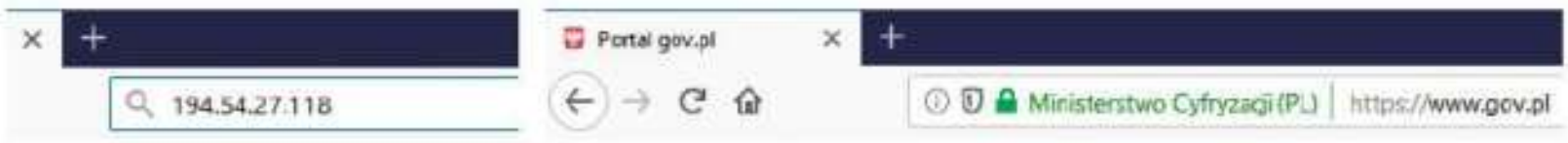
Otwórz przeglądarkę i w osobnych kartach wpisz adres symboliczny men.gov.pl oraz 194.54.27.118 (rys. 53.11.). Wciśnij **Enter** i poczekaj na efekt. Czy wczytała się ta sama strona w obu kartach? Przeglądarka w obu przypadkach połączyła się z tym samym serwerem stron i wczytała tę samą stronę.

Dlaczego tak się dzieje? Odpowiedź jest prosta – chodzi o wygodę korzystania z sieci WWW. Czy w trakcie czytania rozdziału zapamiętałeś/zapamiętałaś IP strony internetowej? Właśnie. A adres symboliczny zapisał się w pamięci bez problemu.

Jak to możliwe, że wpisanie adresu symbolicznego jest także zrozumiałe dla sieci? Zawdzięczamy to komputerom w centrach DNS, w których są przechowywane informacje o adresach IP i odpowiadających im domenach, czyli adresach symbolicznych. Przeglądarka wysyła zapytanie do DNS o adres symboliczny i po jego otrzymaniu łączy się z tym adresem. Skąd dane w DNS? Wykupując domenę, czyli adres symboliczny, właściciel musi ją powiązać z konkretnym serwerem, który świadczy usługę hostingową, co oznacza, że udostępnia stronę w sieci. A ten ma konkretne IP.

◂ DNS

Za pomocą **tracert** możesz sprawdzić IP znanych ci stron, na przykład strony szkoły.




Rys. 53.11. Adresy tej samej strony w różnej postaci

W sieci funkcjonują także strony, które odnajdują dane właścicieli domen oraz przypisane im IP lub na podstawie IP podają domenę (rys. 53.12.). Zauważ, że te same dane pokażą się po wpisaniu zarówno IP, jak i adresu symbolicznego.




Rys. 53.12. Dane adresowe strony internetowej odczytane ze strony z usługą geolokalizacji

## ZADANIA DO WYKONANIA

1. Wczytaj do przeglądarki stronę men.gov.pl i sprawdź, jakie symbole pojawiają się po prawej stronie adresu wyświetlanego w polu adresowym przeglądarki. Korzystając z dowolnych, wiarygodnych źródeł, wytłumacz ich znaczenie. Odpowiedz, dlaczego pojawiły się w tym miejscu i jaki to ma wpływ na relacje pomiędzy użytkownikiem a właścicielem strony.
2. Skorzystaj z programu VisualRoute w wersji lite (darmowej), by zbadać trasę zestawioną przez protokół IP od twojego komputera do twojej ulubionej strony informacyjnej. Powtórz ćwiczenie i porównaj wyniki działania programu. Wskaż różnice i wyjaśnij, dlaczego się pojawiły.
3. Użyj polecenia tracert do odczytania drogi połączenia ze stroną twojej szkoły. Odczytaj nazwy węzłów i sprawdź, jak długo trwało łączenie pomiędzy nimi. Na podstawie informacji pozyskanych z wiarygodnych stron wyjaśnij, skąd wynikają różnice.
4. Odszukaj w internecie strony, które świadczą usługi udostępniania danych na temat właścicieli domen internetowych (adresów symbolicznych). Odczytaj także inne dane np. o serwerze, na którym znajduje się strona o danym adresie.

5*. Znajdź w sieci usługi, w których można zarezerwować i kupić domenę internetową (adresu symbolicznego). Sprawdź, czy domena zawierająca twoje nazwisko jest wolna i ile kosztuje jej wykupienie. Porównaj ceny domen .pl, .com, .eu. Opisz, czym się różnią z punktu widzenia użytkownika i odbiorcy strony internetowej spod danego adresu.

## PODSUMOWANIE LEKCJI

s. 314 • **Internet** jest siecią szkieletową, która łączy ze sobą sieci lokalne. Mogą one być niewielkie, np. w mieszkaniu, i rozlegle, np. w dzielnicy lub wsi.

s. 317 • Wszystkie urządzenia i programy sterujące pracą sieci muszą działać według zasad opisanych w specyfikacji protokołów sieciowych. Dwa podstawowe to **IP** – opisuje sposób łączenia komputerów, i **TCP** – reguluje zasady przesyłania informacji. Uwzględniając te zasady, działają pozostałe protokoły, np. HTTP – protokół stron internetowych, FTP – przesyłanie plików, SMTP – odpowiedzialny za przesyłanie poczty i inne.

s. 318 • Każdy komputer wpięty bezpośrednio do internetu, w tym routery i serwery, ma swój adres IP, który przydziela **ICANN** (odpowiednikiem tej instytucji w Polsce jest NASK).

s. 318 • Adres w sieci lokalnej przydziela **usługa protokołu DHCP**.

s. 319 • Drogę połączenia z danym punktem mającym własne IP w internecie można śledzić za pomocą **polecenia tracert** wydanego w **Wierszu poleceń** systemu Windows lub odpowiedniego programu.

s. 321 • Adresy symboliczne i odpowiadające im adresy IP są przechowywane w **serwerach DNS**. Przeglądarka wysyła zapytanie do DNS, przesyłając adres symboliczny, i otrzymuje odpowiadający mu IP.

# 54. Zabezpieczenia firewall

**NA TEJ LEKCJI:**

- dowiesz się, czym jest zapora sieciowa i jakie jest jej zadanie;
- poznasz serwery NAT.

Korzystając ze sprzętu komputerowego i internetu, jesteśmy narażeni na ataki z zewnątrz. Pojęcia takie jak: sniffing (czytaj: snifin), spoofing (czytaj: spufin), DDoS, wirusy, robaki, konie trojańskie, na stałe weszły do języka użytkowników sieci. Jak się obronić przed atakiem? Jednym ze sposobów jest zastosowanie zapory sieciowej.

## 54.1. Firewall, czyli ściana ogniowa

Zapora sieciowa, nazywana również zaporą ogniową (ang. firewall, czytaj: fajerłol), ma przede wszystkim za zadanie chronić nasz komputer czy też sieć komputerową przed atakami z zewnątrz bądź nieautoryzowanym dostępem do naszych zasobów. W systemie informatycznym taką funkcję pełnią serwery NAT (ang. Network Address Translation, czytaj: netłork adres translejszyn) i proxy (czytaj: proksy).

◀ serwery NAT

### Jak działa zapora sieciowa?

Jedną z ważniejszych funkcji zapory sieciowej jest filtrowanie ruchu, który odbywa się w lokalnej sieci komputerowej. Zapora sprawdza pochodzenie pakietów i akceptuje te pożądane. Stosowane są algorytmy identyfikacyjne użytkowników (hasła, cyfrowe certyfikaty) oraz zabezpieczanie programów obsługujących niektóre protokoły.

Zapora sieciowa pracuje we wszystkich warstwach modelu ISO/OSI (o którym więcej dowiesz się w rozdziale 56), wykonując na ich różnych poziomach inne zadania.

Adres MAC (sprzętowy adres karty sieciowej) jest filtrowany w warstwach fizycznej i łącza danych. W warstwie sieciowej są filtrowane adresy IP. Porty natomiast firewall filtruje w warstwie transportowej. Ochrona antywirusowa i współpraca z serwerami proxy odbywa się w warstwach sieci, aplikacji i prezentacji.

## 54.2. Typy zapór sieciowych

Jednym z typów zapór sieciowych jest zapora filtrująca. Monitoruje ona przepływające przez nią pakiety sieciowe i przepuszcza tylko te zgodne z regułami ustanowionymi na danej zaporze. Zapora sieciowa najczęściej jest stawiana na komputerach z systemem operacyjnym Linux. Obecnie stosowana metoda filtrowania w tym systemie to iptables.

◀ zapora sieciowa

Translacja adresów sieciowych – serwer NAT – polega na zmianie adresu IP hosta wewnętrznego w celu ukrycia go przed zewnętrznym monitorowaniem go. Czynność ta nazywa się również maskowaniem adresu IP.

Zapory pośredniczące, czyli proxy, wykonują połączenia z serwerem w imieniu użytkownika. Zamiast uruchomienia sesji http bezpośrednio do zdalnego serwera WWW zostaje uruchomiona sesja z zaporą, a stamtąd dopiero zostaje uruchomione połączenie z systemem zdalnym. Cała komunikacja na serwer http przechodzi najpierw przez serwer proxy, który zazwyczaj filtruje ruch. Jeśli serwer pośredniczący posiada taką funkcję, może rozpoznawać komendy http, a także analizować zawartość pobieranych stron WWW, działając w warstwie aplikacji modelu ISO/OSI. Działanie zabezpieczające zapory polega na tym, że możliwe jest zablokowanie wybranej treści (mogą to być np. wulgaryzmy lub treści pornograficzne).

filtrowanie ruchu ▸

## 54.3. Wady zapór sieciowych

Choć zapora sieciowa ma wiele zalet, to są ataki, którym się nie zapobiegnie przy użyciu firewall. Zapora sieciowa nie ochroni nas przed nasłuchiwaniem ruchu sieciowego czy przechwytywaniem e-maili. Firewalla należy używać jako jednego z zabezpieczeń sieci. Podstawową zasadą jest przeszkolenie użytkowników w zakresie używania sieci.

polityka bezpieczeństwa ▸

Zapory sieciowe mają duży problem z atakami typu IP Spooding, czyli podszywaniem się pod komputer z innym adresem IP, który jest akceptowany przez odbiorcę. Standardowo używanie zapór ogniowych powinno być uzupełniane programami antywirusowymi, systemami szyfrowania danych, a przede wszystkim odpowiednią polityką bezpieczeństwa, która obejmuje nie tylko systemy teleinformatyczne.

## ZADANIA DO WYKONANIA

1. Wyjaśnij, na czym polega tworzenie przez zaporę sieciową strefy ograniczonego zaufania nazywanej również strefą zdemilitaryzowaną DMZ. Skorzystaj z informacji w internecie.
2. Wyjaśnij etapy działania usługi NAT.
3. Co to jest zapora sieciowa dedykowana? Wyjaśnij zasady jej działania. Skorzystaj z informacji w internecie.

## PODSUMOWANIE LEKCJI

s. 323 ▸ • **Zapora sieciowa**, nazywana również zaporą ogniową, ma za zadanie chronić nasz komputer czy sieć komputerową przed atakami z zewnątrz bądź nieautoryzowanym dostępem do naszych zasobów.

s. 323 ▸ • **NAT** jest techniką przesyłania ruchu sieciowego przez router, która wiąże się ze zmianą źródłowych lub docelowych adresów IP.

s. 324 ▸ • Jedną z ważniejszych funkcji zapory sieciowej jest **filtrowanie ruchu**, który odbywa się w lokalnej sieci komputerowej.

s. 324 ▸ • Podstawą bezpieczeństwa sieci komputerowej jest zapewnienie odpowiedniej **polityki bezpieczeństwa**, która obejmuje nie tylko systemy teleinformatyczne.

# 55. Sieć to nie tylko internet, czyli poznajemy topologię sieci komputerowych

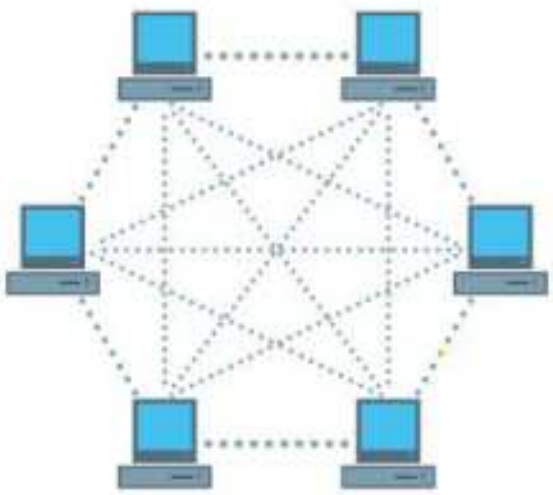
**NA TEJ LEKCJI:**

- dowiesz się, czym jest sieć P2P oraz topologia gwiazdy.

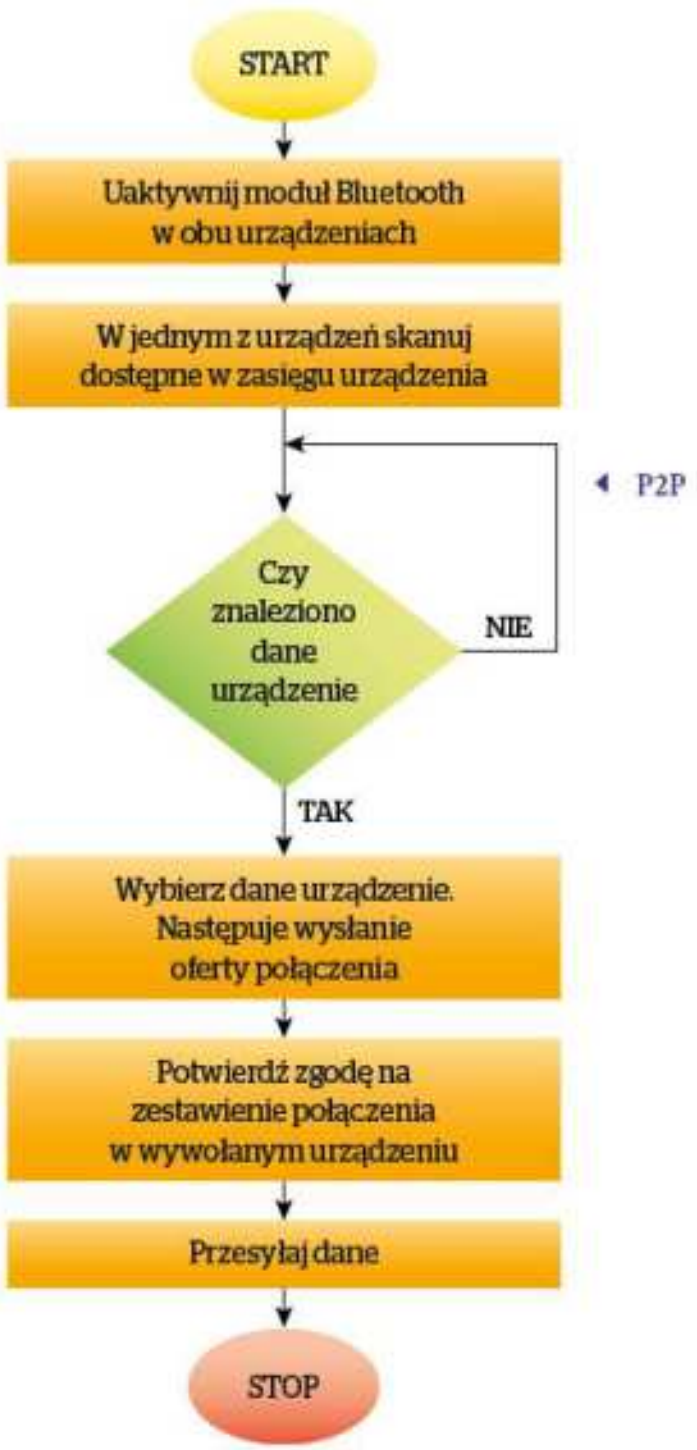
Internet to sieć szkieletowa łącząca mniejsze sieci. Ale jak są zbudowane mniejsze sieci? Czy rządzą nimi takie same prawa, jakie obowiązują w internecie? Jakich urządzeń potrzeba do uruchomienia małej sieci w domu lub biurze?

## 55.1. Komputer z komputerem, czyli sieć P2P

Najprostszą sieć mogą tworzyć dwa komputery połączone ze sobą w taki sposób, by możliwa była wymiana danych. Gdy łączysz się poprzez Bluetooth lub WiFi z komputerem, by skopiować na jego dysk na przykład zdjęcia, tworzysz małą, lokalną, najprostszą sieć typu P2P (Peer-to-Peer, czytaj: pir-tu-pir). Może być ich oczywiście więcej. W tego rodzaju sieci komputery łączą się ze sobą bez pośrednictwa innych urządzeń i wszystkie mają takie same prawa (rys. 55.1.). Strukturę z rysunku 55.1. nazywa się także strukturą pełnej siatki.



Rys. 55.1. Komputery w sieci P2P, w której wszystkie komputery są ze sobą połączone




Rys. 55.2. Proces zestawienia połączenia P2P urządzeń z Bluetooth

W sieci P2P każdy komputer może pełnić funkcję hosta (komputera użytkownika) lub serwera. Wymiana danych odbywa się bezpośrednio pomiędzy nimi. Takie komputery nazywa się węzłami sieci. Z racji jej budowy P2P ma zastosowanie głównie do wymiany plików – podobnie jak we wspomnianym przykładzie z telefonem komórkowym i komputerem. Jest też stosowana do nielegalnego rozpowszechniania plików multimedialnych. Udostępnianie ich w ten lub inny sposób bez zezwolenia jest przestępstwem. Jeśli takie połączenie komputerów wykorzysta się do budowy sieci lokalnej, może to zwiększyć niezawodność połączeń. Oczywiście warunkiem jest prawidłowe działanie wszystkich komputerów.

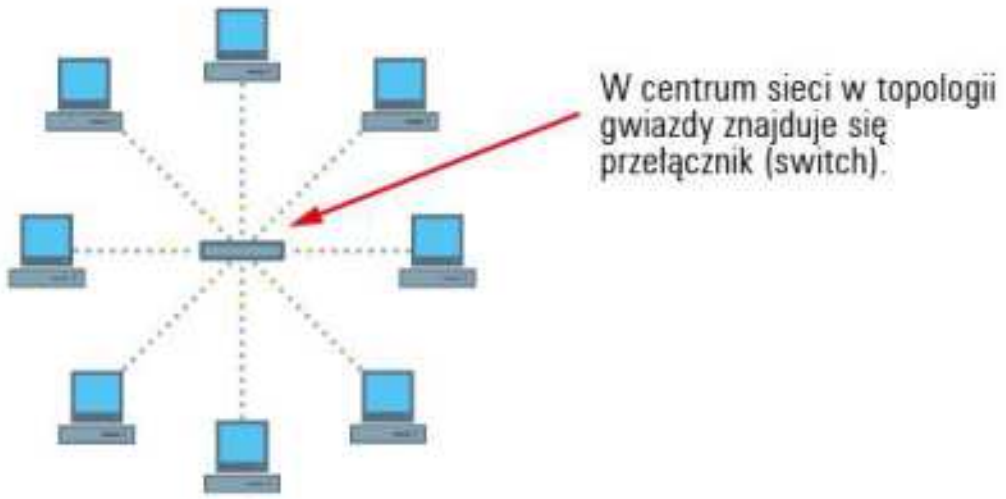
Jak nawiązać połączenie między smartfonem a komputerem? W pewnym stopniu zależy to od wersji ich systemów, ale ogólne zasady są niezmienne (rys. 55.2.).

## 55.2. Przełączniki, czyli więcej o infrastrukturze sieci lokalnych

Sieci lokalne, bez względu na liczbę połączonych komputerów, muszą współdziałać w ramach pewnej infrastruktury. Najważniejsze w niej są łącza – przewodowe lub bezprzewodowe. Sposób połączenia komputerów (fizyczny i logiczny) nazywamy topologią. Istnieje wiele różnych rozwiązań technicznych.

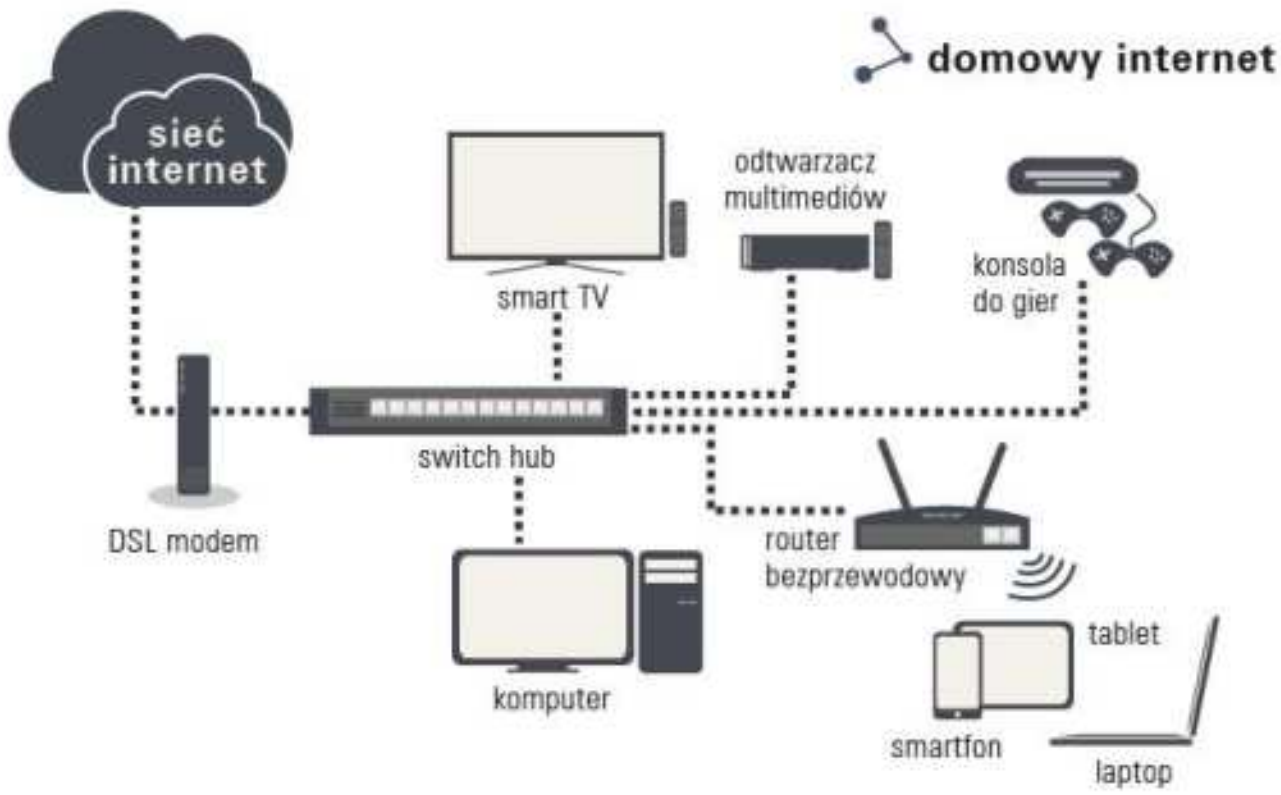
Najpopularniejsze obecnie spotykane w przedsiębiorstwach, biurach i mieszkaniach nosi nazwę **topologia gwiazdy** (rys. 55.3.).

topologia gwiazdy ▸




Rys. 55.3. Sieciowa topologia gwiazdy

Z takim połączeniem zapewne masz do czynienia w swojej domowej sieci. To rozwiązanie zakłada, że głównym elementem infrastruktury jest przełącznik. W ten sposób łączy się nie tylko komputery, lecz także inne urządzenia (rys. 55.4.).




Rys. 55.4. Przykładowa domowa sieć łącząca różne urządzenia

Transfer danych w takiej sieci odbywa się za pośrednictwem przełącznika, z którym są połączone wszystkie urządzenia, w tym także router sieci bezprzewodowej. Dostawcy internetu często udostępniają urządzenia, które łączą w sobie modem, przełącznik sieciowy i router wraz z punktem dostępowym WiFi.

Do zalet topologii gwiazdy należą:

- duża odporność na awarie,
- awaria pojedynczego przewodu lub komputera nie zakłóca pracy sieci,
- większa przepustowość,
- łatwa lokalizacja uszkodzeń,
- duża wydajność,
- łatwa rozbudowa.

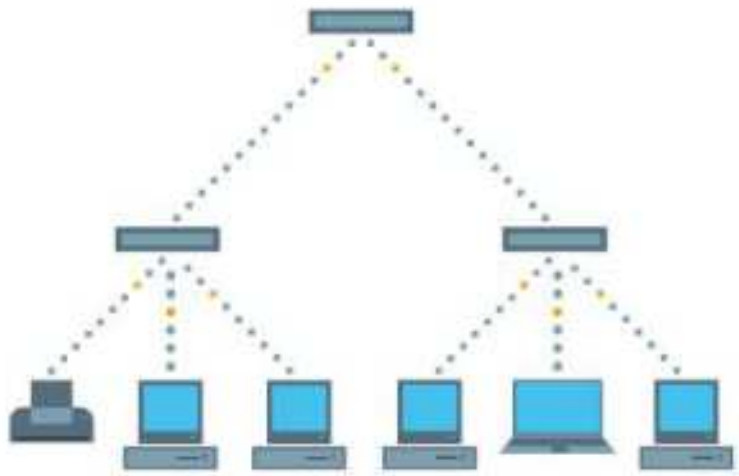
Wśród wad topologii gwiazdy można wymienić:

- zależność funkcjonowania sieci od jednego urządzenia centralnego,
- konieczność doprowadzenia przewodów od przełącznika do urządzeń podłączanych do sieci, co w niektórych przypadkach łączy się z koniecznością robienia przepustów przez ściany i stropy.

topologia gwiazdy rozszerzonej

**Topologia gwiazdy rozszerzonej** pozwala na łączenie ze sobą sieci zbudowanych w topologii gwiazdy (rys. 55.5.). Można ją zastosować, gdy odległości pomiędzy poszczególnymi grupami komputerów, na przykład między biurami na różnych kondygnacjach, są duże. Prowadzenie kabli od przełącznika do każdego z komputerów byłoby niewskazane (grube, długie wiązki). Dlatego w takim

przypadku przełączniki z poszczególnych pięter łączy się jednym przewodem z głównym przełącznikiem. Można też pozostawić opcję wymiany danych między wszystkimi urządzeniami sieci. Niektóre rodzaje przełączników się konfiguruje, aby ograniczyć dostęp do podsieci pozostałym komputerom.



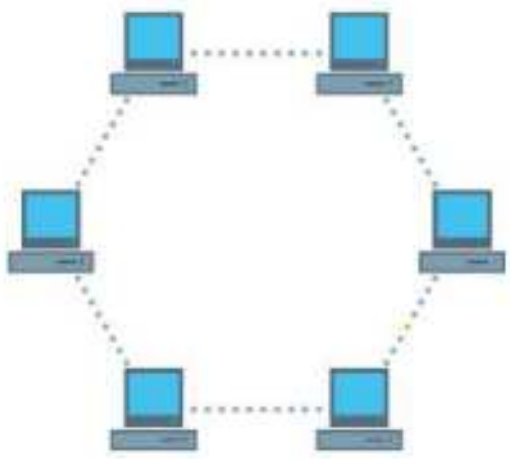
Rys. 55.5. Przykład sieci o topologii gwiazdy rozszerzonej

Zaletami topologii gwiazdy rozszerzonej są:

- możliwość łączenia ze sobą mniejszych sieci,
- ograniczenie liczby kabli między głównym przełącznikiem a znacznie oddalonymi grupami komputerów i urządzeń.

Do wad topologii gwiazdy rozszerzonej należą:

- konieczność rozbudowy infrastruktury o dodatkowe urządzenia,
- zwiększenie poboru energii przez infrastrukturę,
- konieczność przygotowania odpowiednich miejsc na dodatkowe przełączniki.



Rys. 55.6. Przykład sieci o topologii pierścienia

Topologia pierścienia to sposób połączenia komputerów, jaki pokazano na rysunku 55.6. Liczba łączy i ich długość są ograniczone. Organizacja logiczna takiej sieci opiera się na tak zwanym żetonie. W danej sieci istnieje jeden komputer, który przejmie żeton i może transmitować dane. Żeton oznacza określoną sekwencję bitów.

W praktyce można spotkać jeszcze sieci o topologii mieszanej, mające cechy kilku topologii.

## ZADANIA DO WYKONANIA

1. Zbadaj, jaką topologię zastosowano przy budowie sieci w twojej szkolnej pracowni komputerowej. Ustal, czy jest ona siecią jednego typu i czy łączy się z innymi sieciami lokalnymi w szkole.
2. Podaj założenia do budowy sieci w bloku mieszkalnym składającym się z 20 mieszkań położonych w 2 klatkach schodowych. Załóż, że w każdym mieszkaniu może działać więcej niż 1 komputer. Jakiego typu będzie to sieć?
3. Wykonaj swoim telefonem wspólne zdjęcie z kolegą lub koleżanką z klasy i połącz się za pośrednictwem Bluetooth z jego/jej telefonem. Po prawidłowym zestawieniu połączenia prześlij wykonane wcześniej zdjęcie. Rozłącz połączenie.
4. Ułóż algorytm w postaci listy kroków opisujący czynności podczas łączenia dwóch telefonów za pośrednictwem Bluetooth.

5*. Użyj jednego z narzędzi systemu Windows, aby sprawdzić, które z komputerów w pracowni są podłączone do sieci. Zwróć uwagę na ich nazwy. Spróbuj przesłać pliki na ich dysk. Jaki warunek musi być spełniony, by przesłanie się powiodło?

## PODSUMOWANIE LEKCJI

- Połączenie twojego telefonu z komputerem za pośrednictwem Bluetooth jest siecią typu **P2P**. ◂ s. 325
- Najpopularniejszą topologią sieci jest **topologia gwiazdy**. ◂ s. 326
- Sieci zbudowane zgodnie z topologią gwiazdy mogą być łączone w **gwiazdę rozszerzoną**. Urządzenia komputerowe można łączyć w sieci na różne sposoby. ◂ s. 327

# 56. Warstwowy model sieci komputerowej

**NA TEJ LEKCJI:**

- poznasz modele warstwowe sieci ISO/OSI i TCP/IP;
- dowiesz się, w jakim celu oba modele zostały stworzone.

Komunikacja między różnymi urządzeniami w sieci komputerowej składa się z różnych etapów. Każdy z nich realizuje niezbędne zadania do prawidłowej komunikacji.

modele warstwowe sieci

Etapy te są określane przez tzw. modele warstwowe. W sieciach komputerowych istnieją dwa takie modele – ISO/OSI i TCP/IP.

## 56.1. Model Wzorcowy Połączeń w Systemach Otwartych ISO/OSI

model ISO/OSI

ISO/OSI (ang. Open Systems Interconnection Reference Model, czytaj: ołpen systems interkonekszyn refrens model) został stworzony w celu normalizacji zasad komunikacji w sieci. Całym przedsięwzięciem kieruje od 1978 r. Międzynarodowa Organizacja Normalizacji, która opublikowała model opisujący poszczególne warstwy sieci z określeniem zasad, na jakich one działają, nadając im standardowe nazwy i określając zadania.

Model ISO/OSI został przygotowany z myślą o tworzeniu tzw. systemu otwartego, który nie będzie należał do żadnej zamkniętej sieci. Model został podzielony na siedem warstw:

**Warstwa aplikacji** – jest warstwą siódmą i najbliższą użytkownikowi. Zapewnia aplikacjom dostęp do sieciowych usług komunikacyjnych. Świadczy usługi aplikacjom spoza modelu. Zawiera protokoły takich działań, jak obsługa poczty elektronicznej czy przesyłanie danych.

**Warstwa prezentacji** – warstwa szósta odpowiada za translację formatów danych między różnymi systemami. Przekształca dane u nadawcy z reprezentacji wykorzystywanej na komputerze użytkownika do standardowej reprezentacji w sieci oraz na odwrót, od standardowej reprezentacji w sieci do reprezentacji wykorzystywanej na komputerze odbiorcy. Zajmuje się również szyfrowaniem i deszyfrowaniem danych oraz ich kompresją.

**Warstwa sesji** – jest warstwą piątą w modelu ISO/OSI. Dostarcza usługi użyteczne w komunikacji między warstwami prezentacji – jest odpowiedzialna m.in. za synchronizację komunikacji w sieci – ustanawia, steruje i zamyka sesje między komunikującymi się ze sobą komputerami.

**Warstwa transportu** – warstwa czwarta musi zagwarantować bezbłędną transmisję danych między użytkownikami. Ma za zadanie wyszukiwanie błędów

i odrzucanie pakietów zawierających błędy. Dane przesyłane z warstwy sesji w warstwie transportu są dzielone na segmenty, którym są nadawane numery porządkowe. Ta warstwa również gromadzi u odbiorcy wszystkie przesyłane do niego segmenty, z których składa się informacja, składa je w całość w odpowiedniej kolejności.



Rys. 56.1. Model ISO/OSI

**Warstwa sieciowa** – jest trzecią warstwą modelu, w której segmenty danych są wzbogacane o nagłówek sieci zawierający adres IP źródła i celu. W ten sposób tworzy się pakiety danych. Głównym zadaniem warstwy sieciowej jest wybranie najlepszej drogi od nadawcy do odbiorcy dla stworzonych w niej pakietów.

**Warstwa łącza danych** – warstwa druga. Jej zadaniem jest wykrywanie i korygowanie błędów, na jakie narażona jest komunikacja w sieci. Poza tym zajmuje się ona podziałem wysyłanego pakietu na ramki odpowiadające architekturze sieci oraz uzupełnieniem ramek o informacje, które umożliwiają wykrycie błędów transmisji. Warstwa ta jest odpowiedzialna również za sterowanie dostępem do medium transmisyjnego oraz nadawanie adresów przesyłkom z wykorzystaniem adresów MAC kart sieciowych.

Kontroluje poprawność komunikacji poprzez umieszczenie specjalnego szablonu bitów na początku i końcu każdej ramki. Oblicza sumę kontrolną dopisywaną do ramki. U odbiorcy stwierdza, czy obliczona przez niego suma kontrolna jest identyczna z tą, która została zapisana w ramce. Suma kontrolna służy do sprawdzenia, czy nie nastąpiło jakieś przekłamanie w czasie przesyłania ramek.

**Warstwa fizyczna** – warstwa pierwsza jest odpowiedzialna za fizyczny transport mediami sieciowymi komunikatu złożonego z ciągu bitów. W warstwie tej są określone wymagania techniczne dotyczące medium transmisyjnego. Protokoły tej warstwy określają parametry elektryczne, mechaniczne i sygnalizacyjne tak, aby przesyłanie informacji było bezawaryjne.

## 56.2. Protokoły TCP/IP

TCP/IP (ang. Transmission Control Protocol/Internet Protocol, czytaj: transmiszyn kontrol protokol/internet protokol) jest zbiorem protokołów, które służą do transmisji danych przez sieć komputerową. TCP/IP wdraża najważniejsze funkcje siedmiu warstw modelu ISO/OSI.

◀ model TCP/IP

Tab. 56.1. Porównanie modeli ISO/OSI i TCP/IP

| ISO/OSI | TCP/IP |
|---|---|
| aplikacji<br>prezentacji<br>sesji | aplikacji |
| transportowa | transportowa |
| sieciowa | internetu |
| łącza danych<br>fizyczna | dostępu do sieci |

Model TCP/IP został stworzony w latach 70. XX w. Jego twórcom przyświecał zamysł stworzenia modelu odpornego na ataki sieci komputerowych. W późniejszym etapie prac oraz funkcjonowania model TCP/IP stał się podstawową strukturą internetu. Model TCP/IP składa się tylko z czterech warstw:

**Warstwa aplikacji** jest najwyższym poziomem, w którym pracują takie aplikacje, jak serwery WWW czy przeglądarki internetowe. Obejmuje ona zestawy protokołów, które aplikacje wykorzystują do przesyłania informacji w sieci.

**Warstwa transportowa** ma za zadanie zagwarantowanie pewności przesyłu danych oraz kierowania właściwych informacji do konkretnych aplikacji. Opiera się ona na wykorzystaniu portów, które zostają określone dla każdego połączenia. W jednym komputerze może być wiele aplikacji wymieniających dane z innymi komputerami w sieci, warstwa transportowa musi zadbać o to, aby nie nastąpiło wymieszanie się przesyłanych informacji. Nawiązuje ona i zrywa połączenia między komputerami oraz musi zapewnić pewność transmisji.

**Warstwa internetu** – zwana również warstwą protokołu internetowego. Jej zadaniem jest przetwarzanie datagramów (podstawowych jednostek przekazu) mających adresy IP. Warstwa ta ustala odpowiednią drogę do komputera docelowego w sieci. Niektóre z urządzeń sieciowych – routery, które zajmują się kierowaniem ruchem w internecie, mają tę warstwę jako warstwę najwyższą. Zajmują się trasowaniem, czyli określaniem właściwej drogi w sieci.

**Warstwa dostępu** do sieci – to najniższa z warstw w tym modelu. Zajmuje się ona przekazywaniem danych przez fizyczne połączenia między różnymi urządzeniami sieciowymi – najczęściej tymi urządzeniami są karty sieciowe. Poza tym ta warstwa jest czasami wyposażona w protokół do dynamicznego określania adresów IP.

## ZADANIA DO WYKONANIA

1. Wyjaśnij, czym różni się model ISO/OSI od modelu TCP/IP.
2. Wyjaśnij, za co jest odpowiedzialna warstwa sieciowa w modelu ISO/OSI oraz która z warstw realizuje jej zadania w modelu TCP/IP.
3. Które warstwy z modelu ISO/OSI realizują zadania warstwy aplikacji z modelu TCP/IP?

## PODSUMOWANIE LEKCJI

s. 330 ▸ • Etapy komunikacji w sieci są opisane poprzez modele warstwowe sieci – **ISO/OSI** i **TCP/IP**.

s. 330 ▸ • **Model ISO/OSI** składa się z siedmiu warstw i jest przygotowany z myślą o tworzeniu tzw. systemu otwartego, który nie będzie należał do żadnej zamkniętej sieci.

s. 331 ▸ • **Model TCP/IP** został stworzony w latach 70. XX w. Jego twórcom przyświecał zamysł stworzenia modelu odpornego na ataki sieci komputerowych. Model TCP/IP stał się podstawową strukturą internetu i składa się tylko z czterech warstw.

# 57. Funkcje urządzeń i protokoły

**NA TEJ LEKCJI:**

- przeanalizujesz, czym jest sieć komputerowa i z czego się składa;
- dowiesz się, czym są protokoły komunikacyjne i gdzie się je stosuje.

Sieć komputerowa to komputery i urządzenia sieciowe połączone kanałem komunikacyjnym za pomocą oprogramowania; pozwala dwóm urządzeniom elektronicznym (bądź większej liczbie) komunikować się ze sobą i przesyłać między nimi informacje. Dzięki sieci możemy wykorzystać te urządzenia do współdzielenia informacji oraz współpracy przy realizacji projektów lub zadań. Sieć składa się z wielu elementów, ale do najważniejszych należy sprzęt i oprogramowanie, dzięki któremu będzie ona funkcjonować.

◂ sieć komputerowa

## 57.1. Urządzenia sieciowe

◂ podział urządzeń niezbędnych do obsługi i funkcjonowania sieci

Urządzenia niezbędne do obsługi i funkcjonowania sieci możemy podzielić na trzy grupy:

- urządzenia transmisji,
- urządzenia dostępu,
- urządzenia wzmacniania przesyłanych sygnałów.

**Urządzenia transmisji** są wykorzystywane i przeznaczone do transportu sygnału biegnącego przez sieć. Takimi nośnikami będą media transmisyjne, czyli kable koncentryczne, skrętka dwużyłowa i pięciożyłowa, kable światłowodowe. Możemy do tej grupy również zaliczyć nośniki niematerialne, takie jak światło, fale radiowe czy też mikrofale. Natomiast **urządzenia dostępu** są odpowiedzialne za przygotowywanie danych do przesyłu w sieci, następnie umieszczanie tych danych w sieci i w końcu odbieranie ich przez adresatów.

W sieci lokalnej LAN (ang. Local Area Network, czytaj: lokal erija netłork) takimi urządzeniem będzie na pewno karta sieciowa pełniąca funkcję portu, za pomocą którego komputer jest przyłączony do sieci.

W sieciach rozległych WAN (ang. Wide Area Network, czytaj: łajd erija netłork) urządzeniami dostępu są routery. Działają one na poziomie sieci modelu ISO/OSI i wykorzystują protokoły trasowania (używane do wymiany informacji o trasach między sieciami komputerowym), na przykład: RIP, IGRP, BG), oraz protokoły trasowane (najczęściej IP). Protokoły trasowane (protokół IP) odpowiadają za dostarczanie danych do celu (czyli przenoszą pakiety zawierające dane użytkowników sieci), natomiast protokoły trasowania udostępniają wszystkie funkcje, które określają w sieci WAN optymalne ścieżki dla każdego adresu oraz odbierają pakiety i przesyłają je dalej do miejsc docelowych z wykorzystaniem tych optymalnych ścieżek.

Wzmacniak (koncentrator) jest natomiast urządzeniem, które ma za zadanie odbierać przesyłany sygnał – wzmacnia go i wysyła z powrotem do sieci. Umożliwia także przyłączenie do sieci wielu urządzeń. Ważną funkcją koncentratorów również regenerowanie sygnału.

## 57.2. Protokoły modeli warstwowych

**Protokoły warstw aplikacji:**

protokoły komunikacyjne

Najważniejsze spośród ogólnie znanych protokołów komunikacyjnych stosowanych w warstwie aplikacji wymieniono poniżej:

- **DNS** (ang. Domain Name System, czytaj: domejn nejm system) – jest używany do odwzorowania nazw w internecie na adresy IP.
- **HTTP** (ang. Hypertext Transfer Protocol, czytaj: hajpertekst transfer protokol) – jest używany do przesyłania plików hipertekstowych, czyli tworzących strony WWW.
- **HTTPS** (ang. Hypertext Transfer Protocol Secure, czytaj: hajpertekst transfer protokol sekjur) – szyfrowana wersja protokołu HTTP. Protokół ten wykorzystuje szyfrowanie TLS.
- **SMTP** (ang. Simple Mail Transfer Protocol, czytaj: simpyl mejl transfer protokol) – służy do **wysyłania** wiadomości poczty elektronicznej wraz z załącznikami.
- **POP3** (ang. Post Office Protocol, czytaj: post ofis protokol) – służy do odbioru poczty elektronicznej.
- **IMAP** (ang. Internet Message Access Protocol, czytaj: internet mesydż akses protokol) – używa się go do odbioru poczty elektronicznej. Protokół ten posiada funkcje POP3, a także pozwala na zarządzanie zdalnym folderami znajdującymi się na serwerze.
- **Telnet** (ang. Telecommunication Network, czytaj: telekomunikejszyn netłork) – protokół do obsługi terminala zdalnego urządzenia.
- **FTP** (File Transfer Protocol, czytaj: fajl transfer protokol) – służy do przesyłania plików między klientem a serwerem FTP.
- **DHCP** (ang. Dynamic Host Configuration Protocol, czytaj: dajnamik hołst konfigurejszyn protokol) – jest używany do dynamicznej konfiguracji urządzeń. Odpowiada za przydzielanie adresów IP, adresu domyślnej bramy i adresów serwerów DNS.
- **SH** (ang. Secure Shell, czytaj: sekjur szel) – protokół, którego używa się do emulacji (symulowania działania) terminala sieciowego. Zapewnia szyfrowanie połączenia.
- **NFS** (ang. Network File System, czytaj: netłork fajl system) – protokół używany do udostępniania plików w systemach Unix\Linux.

**Protokoły warstwy transportowej:**

- **UDP** (ang. User Datagram Protocol, czytaj: juzer datagram protokol) – bezpołączeniowy protokół, którego najważniejszą zaletą jest niewielki narzut dodawany do dostarczanych danych. Porcje danych UDP nazywa się datagramami. Nie zapewnia on mechanizmów ochrony i potwierdzenia dostarczenia datagramu czy retransmisji.
- **TCP** (ang. Transmission Control Protocol, czytaj: transmiszyn control protokol) – protokół połączeniowy, który tworzy segmenty danych. Główną zaletą tego protokołu jest jego niezawodność, która uzyskiwana jest dzięki konieczności wysyłania potwierdzenia przez hosta odbierającego dane, zanim zostanie wysłany kolejny segment.

**Protokoły warstwy sieciowej:**

- **Protokół IP** – protokół komunikacyjny warstwy modelu ISO/OSI i warstwy internetu w modelu TCP/IP. Odpowiada za przesyłanie pakietów pomiędzy użytkownikami sieci. Najpopularniejszą obecnie stosowaną wersją protokołu jest starsza jego wersja IPv4. Ma on zapewnić niezbędne funkcje umożliwiające dostarczenie pakietów od nadawcy do adresata. To protokół bezpołączeniowy – przed wysłaniem informacji nie jest zestawiane żadne połączenie. Jest zawodny, gdyż nie ma żadnych mechanizmów zapewniających dostarczenie pakietu. Nie jest zależny od używanego medium transmisyjnego.

## ZADANIA DO WYKONANIA

1. Sprawdź na swoim komputerze adres IP i adres MAC.
2. Wyślij pakiet testowy (ping) do komputera odgrywającego rolę bramy w szkolnej sieci.
3. Sprawdź trasę, po której pakiety są przesyłane do komputera www.cke.gov.pl.
4. Przedstaw, za co odpowiadają protokoły ICMP, ARP.
5. Omów protokoły wirtualnych sieci prywatnych: PPP, PPTP i L2TP.

## PODSUMOWANIE LEKCJI

- **Sieć** składa się z wielu elementów, ale do najważniejszych należą sprzęt i oprogramowanie, dzięki któremu będzie ona funkcjonować. ◂ s. 333
- Urządzenia niezbędne do obsługi i funkcjonowania sieci można podzielić na **trzy grupy**: urządzenia transmisji, urządzenia dostępu oraz urządzenia wzmacniania przesyłanych sygnałów. ◂ s. 333
- Każda warstwa w modelu **ISO/OSI** i **TCP/IP** ma swoje **protokoły komunikacyjne.** ◂ s. 334

# 58. Kto tam, czyli identyfikujemy komputery w sieci

**NA TEJ LEKCJI:**

- nauczysz się odczytywać dane z karty sieciowej;
- dowiesz się, jak skonfigurować kartę sieciową.

Sieć komputerowa działa dzięki protokołom i infrastrukturze technicznej. W każdym urządzeniu sieciowym znajduje się karta sieciowa. Jej skonfigurowanie decyduje o poprawnym działaniu. Od protokołów zależy sprawne komunikowanie się urządzeń i działanie usług. Czym są protokoły i jak konfigurować kartę? Warto to wiedzieć, by świadomie korzystać z możliwości sieci.

## 58.1. Sprawdzamy konfigurację, czyli jak odczytać dane z karty sieciowej

ipconfig/all ▸

Wiesz już, że jeśli komputer ma pracować w sieci, należy mu przydzielić adres IP. Musi tak być zarówno w sieci lokalnej, jak i internecie. Do sprawdzenia stanu połączenia i odczytania parametrów połączenia można użyć polecenia **ipconfig** z parametrem **all** (rys. 58.1.). Co otrzymaliśmy w wyniku działania tego polecenia? Odczytaj informacje z ekranu swojego komputera.

```
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Wojtek>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : komp
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : home

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : home
   Description . . . . . . . . . . . : Qualcomm Atheros AR8611 PCI-E Gigabit Ethernet Controller (NDIS 6.30)
   Physical Address. . . . . . . . . : 96-F4-34-6E-F4-5C
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::3d95:3dc4:b53d:1f1e%11(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.1.13(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : 27 października 2018 17:04:07
   Lease Expires . . . . . . . . . . : 28 października 2018 17:04:07
   Default Gateway . . . . . . . . . : 192.168.1.1
   DHCP Server . . . . . . . . . . . : 192.168.1.1
   DHCPv6 IAID . . . . . . . . . . . : 131843188
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-1F-55-96-A5-F4-5C-34-6E-F4-5C
   DNS Servers . . . . . . . . . . . : 192.168.1.1
                                       192.168.1.1
   NetBIOS over Tcpip. . . . . . . . : Enabled

C:\Users\Wojtek>_
```

Rys. 58.1. Efekt działania polecenia „ipconfig/all"

**Host Name** (czytaj: hołst nejm) to w rzeczywistości nazwa twojego komputera nadana w trakcie instalacji systemu operacyjnego.

**Description** (czytaj: deskrypszyn) jest pełną nazwą modelu karty sieciowej. Z ilustracji można odczytać, że to karta Ethernet (czytaj: eternet; to karta przewodowa) o przepustowości 1 gigabita z układem Qualicom Atheros (czytaj: kłalikom ateros).

**Physical Address** (czytaj: fyzykal adres) to unikatowy, fizyczny adres karty sieciowej zwany także **MAC adresem** (Media Access Control Address, czytaj: midia akses kontrol adres). Każda karta sieciowa musi mieć niepowtarzalny adres fizyczny. Producenci wpisują je na stałe, wybierając z przydzielonej im puli.

Karta z ilustracji ma także włączone automatyczne pobieranie adresu IP (**DHCP Enabled**, czytaj: enejbled) oraz opcję automatycznej konfiguracji (**Autoconfiguration Enabled**, czytaj: ałtokonfigurejszyn).

**IPv4 Address** oznacza adres IP komputera w sieci; w tym przypadku przydzielony przez router w ramach usługi DHCP.

Ponieważ nasz komputer łączy się z inną siecią (internetem), musi korzystać z bramy sieciowej (**Gateway**, czytaj: getełej).

**DHCPv6** pełni podobną funkcję co DHCPv4. Wprowadzono go po modyfikacjach protokołu IP.

Adres IP został przydzielony karcie (komputerowi) przez router, więc adresem urządzenia z usługą **DNS** jest router (**DNS Servers**, czytaj: serwers) pod swoim adresem.

## 58.2. Ping, czyli sprawdzamy poprawność połączenia

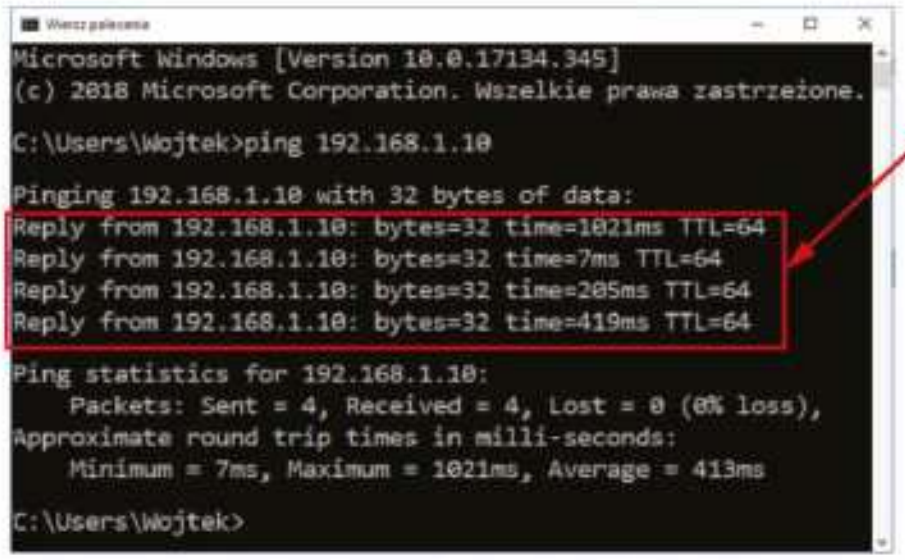
ping

Jak sprawdzić, czy twój komputer może nawiązać kontakt z innym urządzeniem podłączonym do sieci? Załóżmy, że twój telefon łączy się z siecią za pośrednictwem karty WiFi. Oznacza to, że może pracować w tej samej sieci lokalnej, co twój komputer. Aby sprawdzić, czy rzeczywiście tak jest, użyj polecenia **ping** z **Wiersza polecenia** systemu Windows. Zanim jednak rozpoczniesz doświadczenie, odczytaj adres IP przydzielony w sieci karcie twojego telefonu (rys. 58.2.). W tym celu przytrzymaj chwilę ikonę połączenia WiFi i wybierz opcję konfiguracji karty.



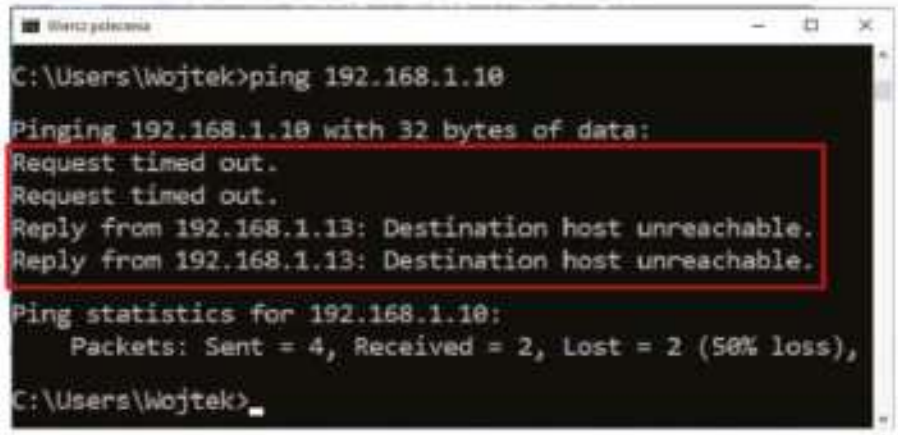

Rys. 58.2. Odczyt IP przydzielonego karcie telefonu z systemem Android

Otwórz **Wiersz polecenia (CMD)** i użyj polecenia **ping**. Ponieważ sprawdza ono połączenie z urządzeniem wyposażonym w kartę sieciową podłączoną do sieci, jako argumentu użyj odczytany z komórki IP (rys. 58.3.).



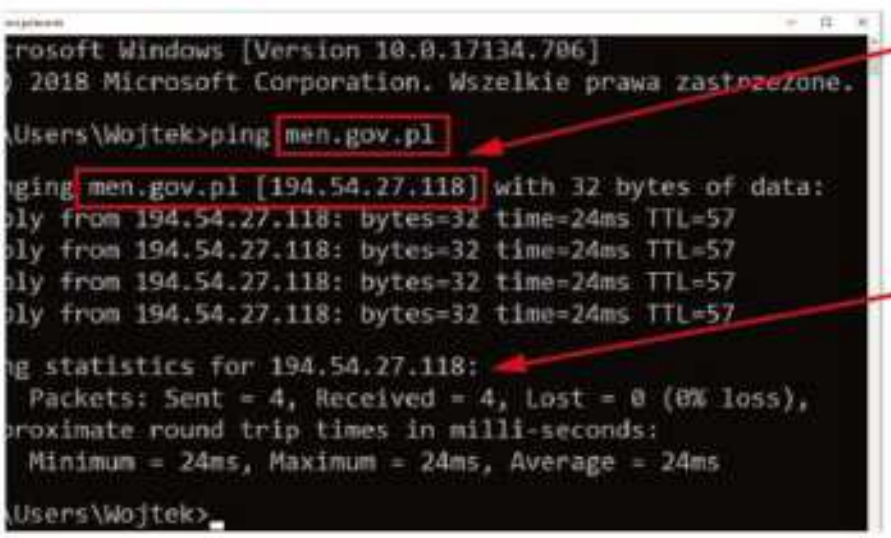

Rys. 58.3. Polecenie ping nawiązało kontakt z telefonem z kartą sieciową WiFi

W przypadku nienawiązania kontaktu z kartą sieciową telefonu otrzymasz komunikat jak na rysunku 58.4. Oczekiwanie na odpowiedź przekroczy czas ustalony dla testu.




Rys. 58.4. Polecenie ping nie nawiązało kontaktu z telefonem z kartą sieciową WiFi

Czy tym samym poleceniem można sprawdzić poprawność podłączenia naszego komputera do internetu? Domyślasz się, jak to zrobić? Wykorzystaj polecenie **ping**. W miejscu adresu IP komputera wpisz domenę (rys. 58.5.).



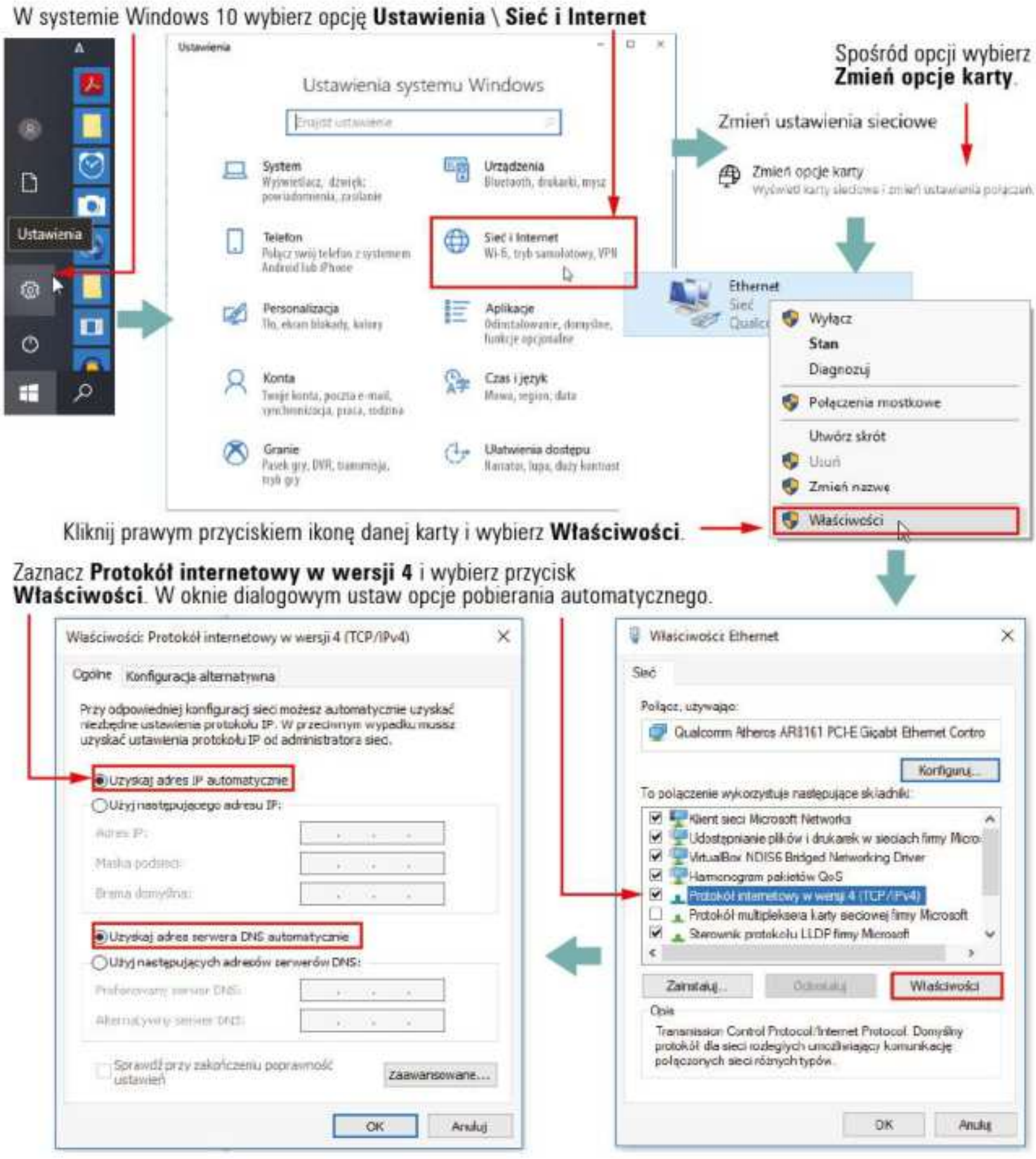

Rys. 58.5. Sprawdzenie połączenia z internetem za pomocą polecenia ping

## 58.3. Kłopoty z siecią, czyli jak skonfigurować połączenie

konfiguracja karty sieciowej

Współczesne systemy operacyjne potrafią same skonfigurować kartę sieciową, umożliwiając komputerowi pracę w sieci i w internecie. Może się jednak zdarzyć, że będziesz musiał/musiała samodzielnie wykonać te czynności.

Zacznijmy od konfiguracji powodującej automatyczne wczytywanie adresu z pomocą protokołu DHCP (rys. 58.6.).




Rys. 58.6. Ustawienie opcji automatycznego pobierania adresów z routera

Jeśli pracujesz w systemie Windows 7, rozpocznij od **Panelu sterowania** (rys. 58.7.).




Rys. 58.7. Opcja „Właściwości karty sieciowej" w systemie Windows 7

Gdy wybierzesz opcję ręcznego ustawienia parametrów połączenia, musisz znać parametry danej sieci (rys. 58.1.). Adresy serwerów DNS dla połączeń internetowych mogą być różne dla różnych dostawców internetu, na przykład dla Neostrady Orange są to: 194.204.152.34 i 194.204.159.1.

## ZADANIA DO WYKONANIA

1. Sprawdź ustawienia połączenia sieciowego karty sieciowej twojego komputera. Wykorzystaj polecenie **ipconfig**. Porównaj je z danymi innego komputera pracującego w tej samej sieci, np. na sąsiednim stanowisku. Wskaż różnice i wyjaśnij, skąd się wzięły.
2. Za pomocą polecenia ping znajdź adresy IP kilku znanych ci stron internetowych. Porównaj czasy odpowiedzi ich serwerów. Doświadczenie powtórz kilka razy. Sprawdź, czy czasy odpowiedzi dla tych samych adresów się różnią. Jeśli tak, zastanów się nad przyczyną i zweryfikuj swoje przypuszczenia w wiarygodnych źródłach.
3. Odnajdź w swoim smartfonie lub tablecie dane karty sieciowej WiFi. Zanotuj jej MAC adres. Ponieważ jest on niepowtarzalny i na stałe wpisany w jej pamięć, może posłużyć do identyfikacji telefonu.

4*. Użyj opcji pomocy **Wiersza polecenia** dla polecenia **ipconfig**. Użyj argumentu **/?** i odczytaj komunikaty. Na podstawie dostępnych źródeł, np. wiarygodnych stron internetowych lub książek, ustal, jakie znaczenie mają poszczególne argumenty tego polecenia.

5*. Skorzystaj z opcji pomocy **Wiersza polecenia** dla polecenia **ping**. Użyj argumentu **/?** i odczytaj komunikaty. Korzystając z dostępnych źródeł, np. wiarygodnych stron internetowych lub książek, ustal, jakie znaczenie mają poszczególne argumenty tego polecenia.

## PODSUMOWANIE LEKCJI

s. 336 • Konfigurację połączenia i dane karty sieciowej poznasz, używając polecenia **ipconfig/all.**

s. 337 • Za pomocą polecenia **ping** sprawdzisz połączenie pomiędzy komputerami w sieci lokalnej, używając ich adresów IP, a także poprawność połączenia z internetem, używając np. IP lub domeny znanej ci strony.

s. 339 • Gdy zajdzie taka potrzeba, np. dla kamery sieciowej, która powinna mieć stałe IP w sieci, możesz **skonfigurować kartę sieciową**, wpisując ręcznie parametry sieci i adres IP.

# 59. Kupujemy świadomie, czyli poznajemy parametry urządzeń peryferyjnych

**NA TEJ LEKCJI:**

- poznasz parametry urządzeń peryferyjnych;
- dowiesz się, na co zwracać uwagę przy kupnie urządzenia.

Na półkach marketów i witrynach sklepów internetowych sprzedawcy oferują mnóstwo urządzeń peryferyjnych. Które wybrać? Jak kupić sprzęt dopasowany do naszych potrzeb? Na co zwracać uwagę? Czy droższe zawsze oznacza lepsze? Aby dokonać właściwego wyboru, musisz poznać podstawowe parametry, które decydują o przeznaczeniu i jakości sprzętu. Zapoznaj się z nimi i samodzielnie oceń wartość urządzeń.

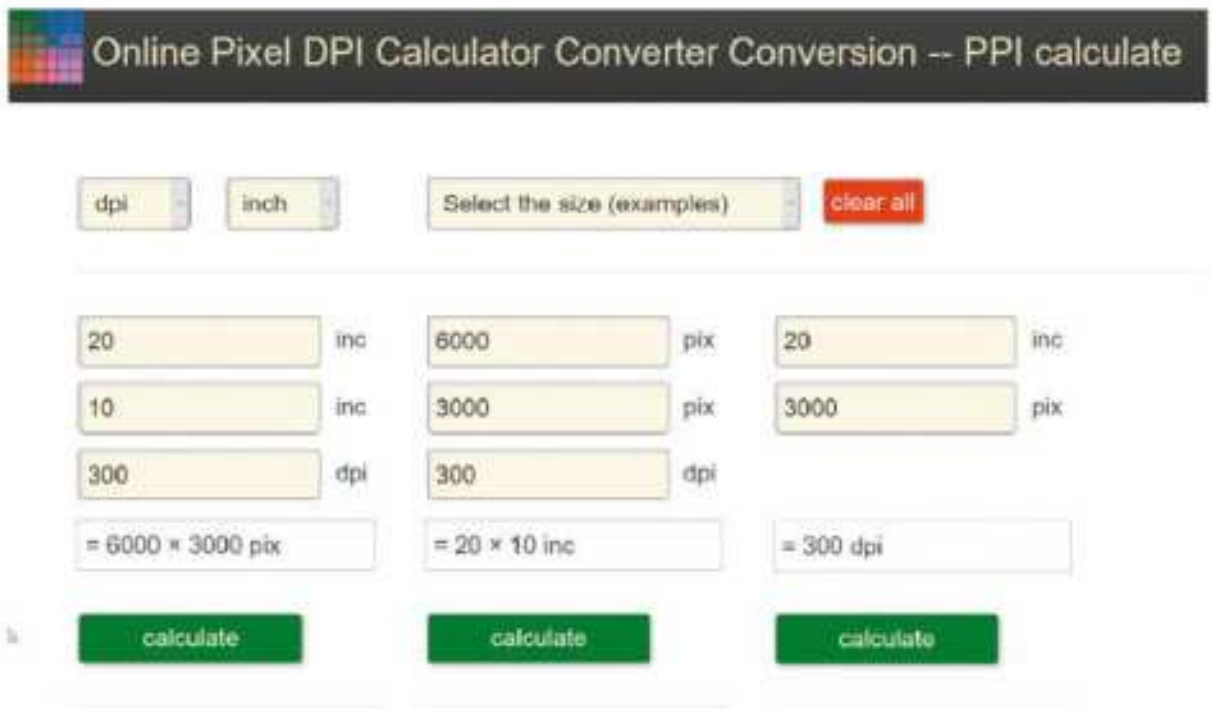
## 59.1. PPI czy DPI, czyli parametry drukarek

Czy zastanawiałeś/zastanawiałaś się kiedyś, co oznaczają skróty w nazwie danego parametru urządzenia peryferyjnego? Czym na przykład różni się PPI od DPI? Zacznijmy od nich.

Zasadniczą różnicą między tymi parametrami jest to, że DPI dotyczy obrazu wydrukowanego, a PPI – wyświetlanego na ekranie.

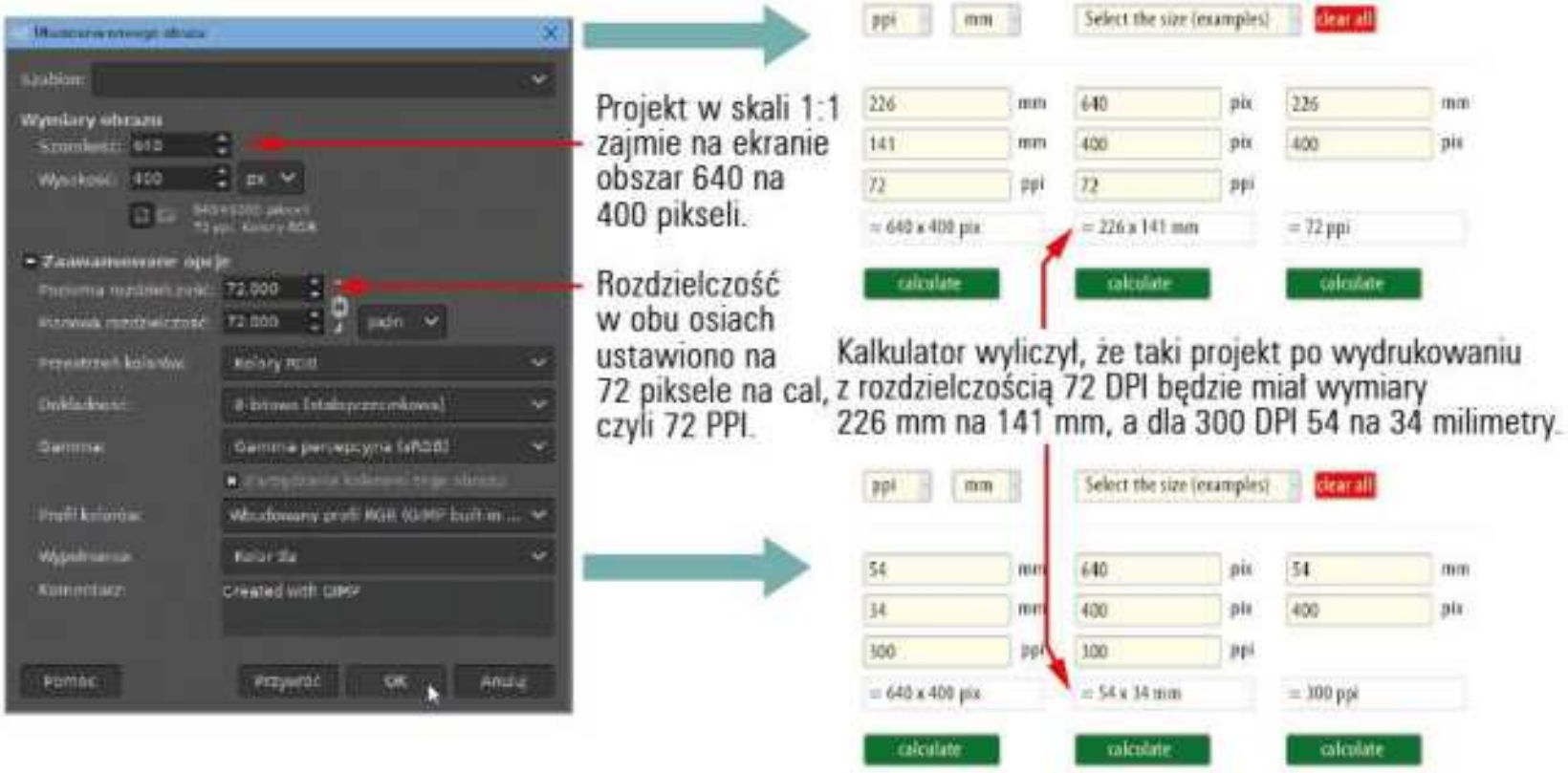
◂ DPI

Wartość podana jako **DPI** (Dots Per Inch, czytaj: dots per incz) oznacza liczbę wydrukowanych punktów na długości jednego cala (2,54 cm). Ma to bezpośrednie przełożenie także na wielkość wydrukowanego obrazka. Przykładowo grafika przygotowana w edytorze zbudowana z 6000 punktów w pionie i 3000 punktów w poziomie wydrukowana w rozdzielczości 300 DPI ma wymiary 20 na 10 cali. Istnieje wiele programów pomagających przeliczać wartości związane z wydrukiem, w tym także on-line (rys. 59.1.). Ta sama grafika drukowana z rozdzielczością 150 DPI zajmie na wydruku obszar 40 na 20 cali. Z tego wniosek, że dla danego obrazu zapisanego w pliku uzyska się różne rozmiaru wydruku w zależności od liczby DPI ustawionej dla drukarki. Wartość ta służy więc do określania rozmiaru obrazu na wydruku względem faktycznych wymiarów obrazu liczonych liczbą punktów powstałych podczas tworzenia danej grafiki.




Rys. 59.1. Kalkulator on-line wspomagający obliczanie wielkości wydruku w zależności od wartości PPI

Czym w takim razie jest **PPI** (Pixels Per Inch, czytaj: piksels per incz)? Szybko to zrozumiesz, jeśli posłużysz się edytorem grafiki rastrowej, na przykład GIMP (rys. 59.2.). Ustawienia nowego projektu zawierają podstawowe jego parametry, w tym także wielkość w pikselach i rozdzielczość PPI.




Rys. 59.2. Wpływ ustawienia parametrów projektu graficznego na wielkość wydruku

cechy i parametry ▸ Domyślasz się, że znajomość tych parametrów pomoże w wyborze drukarki (DPI) i monitora (PPI). Ale nie tylko one. Zanim podejmiesz decyzję o zakupie, odwiedź stronę producenta i pobierz pełną instrukcję obsługi, w której

znajdziesz także komplet parametrów. Nie opieraj swoich sądów na informacjach reklamowych. Na co zwrócić uwagę?

- **Technologia druku** – w powszechnym, domowym i biurowym użytkowaniu wykorzystuje się drukarki laserowe i atramentowe. Na podstawie wydajności i ceny materiałów eksploatacyjnych, tonera lub atramentu można wyliczyć cenę wydruku jednej strony. Technologia druku znacznie wpływa na pozostałe parametry.
- **Rozdzielczość** – ten parametr określa liczbę DPI. W przypadku drukarek laserowych często spotykanymi wartościami są 300, 600 lub 1200. 600 jest wystarczająca do uzyskania dobrej jakości druku i dlatego jest najczęściej używana. Większość drukarek atramentowych drukujących barwnie dla domowego lub biurowego użytku ma rozdzielczość na przykład 1200 x 4800 dpi. Tak duża liczba punktów wynika z tego, że druk powstaje z kropli emitowanych przez wiele dysz dla każdego z kolorów oddzielnie.
- **Szybkość druku** – jest ważna, gdy drukarka ma być użytkowana do druku dokumentów o wielu stronach. Dla drukarek laserowych może sięgać ponad 35 stron na minutę, co daje 1 kartkę co około 2 sekundy. Drukarki atramentowe tej samej klasy drukują wolniej. W czerni osiągają prędkość do 20 stron na minutę. Wydruk kolorowy trwa około dwukrotnie dłużej. Oczywiście dane te mogą być różne w zależności od modelu i producenta drukarki.
- **Format papieru** – najczęściej spotykanym formatem papieru, na którym można drukować dokumenty, jest A4. Większość drukarek drukuje także na formatach mniejszych, na przykład na różnej wielkości kopertach. Niektóre drukarki atramentowe drukują też na płytach CD lub DVD. Z formatem wiąże się dopuszczalna gramatura papieru, czyli jego waga na 1 metr kwadratowy. Typowa kartka do drukowania dokumentów ma gramaturę 80 gramów. Każda drukarka ma ponadto określoną maksymalną liczbę kartek, jaka mieści się w zasobniku, przykładowo 250 sztuk (pół ryzy). Jeśli planuje się drukowanie na grubszym papierze, należy zwrócić uwagę, czy drukarka ma możliwość podawania pojedynczych arkuszy.
- **Obciążalność** – ten parametr jest bardzo ważny, gdy drukarka pracuje w miejscu, w którym wykonuje się wiele wydruków w miesiącu. Drukarki laserowe przewyższają pod tym względem atramentowe. Ich dopuszczalna miesięczna liczba wydrukowanych stron może sięgać nawet do kilkudziesięciu tysięcy. Dla drukarek atramentowych liczba ta sięga kilkunastu tysięcy.
- **Sposób transmisji danych** – każda drukarka domowa i biurowa łączy się z komputerem za pośrednictwem magistrali USB. Wiele modeli ma także wbudowaną kartę sieciową. Dzięki temu staje się ona dostępna dla wielu użytkowników bez konieczności przenoszenia drukarki i przełączania przewodów. Jeśli drukarka ma pracować w sieci firmowej, należy sprawdzić dostępne protokoły transmisji danych.

- **Koszt wydruku** – tego parametru nie podają oficjalne opisy drukarek. Producent materiałów eksploatacyjnych (tonerów lub atramentów) podaje średnią liczbę stron, jaką można wydrukować z użyciem danego materiału, a także normy określające poziom wydajności tych materiałów (laserowe – ISO/IEC 19752, atramentowe – ISO/IEC 24711). Ustalono, że takie dane określa się dla pięcioprocentowego pokrycia strony drukiem dla tekstu i dwudziestoprocentowego – dla grafiki (dla każdego koloru).

| | |
|---|---|
| Technologia druku | Laserowa, monochromatyczna |
| Maksymalny format nośnika | A4 |
| Podajnik papieru | 250 arkuszy |
| Rodzaje podajników papieru | Kasetowy |
| Odbiornik papieru | 150 arkuszy |
| Szybkość druku w mono | 28 str./min |
| Maksymalna rozdzielczość druku | 1200 x 1200 dpi |
| Miesięczne obciążenie | 30000 str./miesiąc |
| Maksymalna gramatura papieru | 163 g/m² |
| Druk dwustronny (dupleks) | Automatyczny |
| Interfejsy | USB<br>Wi-Fi<br>LAN (Ethernet) |
| Wyświetlacz | Brak |
| Szerokość | 371 mm |
| Wysokość | 360 mm |
| Głębokość | 625 mm |
| Waga | 6,9 kg |
| Dołączone akcesoria | Kabel zasilający<br>Kabel USB<br>Toner startowy |

Rys. 59.3. Przykładowe parametry biurowej drukarki laserowej

| | |
|---|---|
| Przeznaczenie produktu | Dom<br>Małe biuro |
| Technologia druku | Atramentowa, kolorowa |
| Maksymalny format nośnika | A3 |
| Podajnik papieru | 500 arkuszy |
| Rodzaje podajników papieru | Kasetowy + tacka |
| Szybkość druku w kolorze | 10 str./min |
| Szybkość druku w mono | 18 str./min |
| Maksymalna rozdzielczość druku | 4800 x 2400 dpi |
| Miesięczne obciążenie | 20000 str./miesiąc |
| Maksymalna gramatura papieru | 256 g/m² |
| Druk dwustronny (dupleks) | Automatyczny |
| Interfejsy | USB<br>Wi-Fi<br>LAN (Ethernet)<br>NFC |
| Wyświetlacz | Wbudowany |
| Szerokość | 567 mm |
| Wysokość | 304 mm |
| Głębokość | 424 mm |
| Waga | 15,5 kg |

Rys. 59.4. Przykładowe parametry biurowej drukarki laserowej formatu A3

## 59.2. 4:3 czy 16:10, czyli parametry monitorów

wybór monitora

Zanim wybierzesz monitor, musisz zapoznać się z kilkoma parametrami. Podobnie jak w przypadku drukarek ważne jest przeznaczenie urządzenia i oczekiwania użytkownika. Zacznijmy od rozdzielczości.

**Rozdzielczość monitora** określa się maksymalną liczbą pikseli matrycy w poziomie i w pionie, na przykład 1920 na 1080 (ful HD). Rozdzielczość ekranu determinuje proporcje obrazu. Dla rozdzielczości określanej jako pełne HD (High Definition, czytaj: haj definyszyn) to 16:9, co łatwo wyliczyć, dzieląc 1920 przez 1080. Im większa rozdzielczość, tym więcej szczegółów na ekranie (rys. 59.5.).

Dobór proporcji ekranu zależy od upodobań użytkownika i właściwości oprogramowania, którym się posługuje. Gracz wybierze zapewne monitor panoramiczny, na przykład 22:9 z jak największą rozdzielczością, użytkownikowi korzystającemu z arkuszy kalkulacyjnych lub edytorów tekstu zapewne wystarczy 16:9.




Rys. 59.5. Porównanie wielkości obrazu w zależności od rozdzielczości

**Przekątna ekranu**, podobnie jak w przypadku telewizorów, jest podawana w calach. Wartość ta powinna być brana pod uwagę w ścisłym połączeniu z proporcjami ekranu.

**Rodzaj matrycy** określa technologię, w jakiej została ona wykonana. Wszystkie obecnie stosowane matryce są podświetlane diodami LED i wykonane na bazie ciekłych kryształów.

**Czas reakcji** decyduje o jakości obrazu (efekt smużenia) w czasie wyświetlania ruchomych obrazów, na przykład w grach. Jego wartość zależy od technologii użytej do budowy matrycy ekranowej. Według norm czas reakcji to czas, jaki upłynął w czasie przejścia pojedynczego piksela ze stanu całkowitego zgaszenia (czarny) poprzez całkowicie rozjaśniony do ponownego wygaszenia. Jednak w niektórych monitorach czas reakcji mierzy się według innych norm. Czas od 1 ms do 5 ms można uznać za wystarczający, by nie odczuwać efektu smużenia.

**Kąt widzenia** ma znaczący wpływ na wrażenia wizualne. Zbyt mały może spowodować zmianę kolorów i jasności podczas obserwowania obrazu z pozycji innej niż prostopadła do ekranu. Widać to szczególnie w osi *y*. Kąt ten mierzy się dla osi *x* i *y*.

**Matryca TN** (Twisted Nematic, czytaj: tłisted nematik) jest szybka, na przykład czas reakcji może osiągnąć 1 ms, jednak nie zapewnia szerokich kątów widzenia. Można to zauważyć zwłaszcza na osi *y*.

▸ rodzaje wyświetlaczy LCD

**Matryca VA** charakteryzuje się podobną szybkością jak TN, jednak znacznie lepszymi kątami widzenia.

**Matryca IPS** (in-plane switching, czytaj: inplejn sliczing) zapewnia duże, zbliżone do 180° kąty widzenia w pionie i poziomie, dużą liczbę kolorów (do ponad miliarda), wysoki kontrast i czas reakcji kilku milisekund.

**Uwaga!**

Producenci nieustannie pracują nad nowymi technologiami wyświetlania obrazu w monitorach. Zazwyczaj są one najpierw wykorzystywane w produkcji telewizorów. Ekrany QLED, OLED lub te, które jeszcze do produkcji nie weszły, mogą niebawem trafić także do monitorów.

**Częstotliwość odświeżania** to liczba klatek (obrazów) wyświetlanych w ciągu sekundy. Standardem jest 60 Hz (60 na sekundę). Niektóre monitory mogą wyświetlać obrazy z wyższą częstotliwością, na przykład 75 Hz lub większą.

interfejsy wejściowe




Rys. 59.6. Wejścia sygnału wideo w monitorze komputerowym

**Interfejsy wejściowe** (rys. 59.6.) muszą być zgodne z wyjściami karty grafiki komputera. Coraz rzadziej spotyka się wejście analogowe VGA. Wśród cyfrowych najpopularniejszy jest HDMI (używany także w domowym sprzęcie telewizyjnym). Oprócz niego można spotkać DVI i DisplayPort. Ten ostatni pozwala na dołączenie do karty grafiki (jeśli ma odpowiednią liczbę portów) kilku monitorów jednocześnie.

Inne parametry są w większości zależne od producentów i zazwyczaj mają poprawić jakość pracy przy monitorze. Redukcja migotania ekranu (Flicker free, czytaj: fliker fri) czy regulacje kolorów mogą znacznie ograniczyć negatywny wpływ wyświetlanego obrazu na wzrok.



Specyfikacja

| | |
|---|---|
| Przekątna ekranu | 27" |
| Powłoka matrycy | Matowa |
| Rodzaj matrycy | LED, IPS |
| Rozdzielczość ekranu | 2560 x 1440 (WQHD) |
| Format ekranu | 16:9 |
| Częstotliwość odświeżania ekranu | 60 Hz |
| Technologia ochrony oczu | Redukcja migotania (Flicker free) |
| Wielkość plamki | 0,233 mm |
| Jasność | 350 cd/m² |
| Kontrast statyczny | 1 000:1 |
| Kontrast dynamiczny | 3 000 000:1 |
| Kąt widzenia w poziomie | 178 stopni |
| Kąt widzenia w pionie | 178 stopni |
| Czas reakcji | 4 ms |
| Liczba wyświetlanych kolorów | 1,07 mld |
| Rodzaje wejść / wyjść | HDMI - 1 szt.<br>DisplayPort - 1 szt.<br>DC-in (wejście zasilania) - 1 szt. |

Rys. 59.7. Przykładowe parametry monitora komputerowego

Czy wiesz, czym jest **rozdzielczość natywna** monitora? To rozdzielczość wyświetlacza określona fizycznie występującą liczbą pikseli w osi pionowej i poziomej. Ustawiając w sterowniku karty grafiki rozdzielczość natywną, uzyskuje się najlepszy efekt wyświetlania obrazu.

## 59.3. Z papieru na dysk bez problemów, czyli jak wybierać skaner

Wiesz już, jak działają skanery i do czego służą. Podobnie jak w przypadku drukarek i monitorów przed wyborem odpowiedniego sprzętu należy przeanalizować potrzeby użytkownika i wymagania oprogramowania.

Od domowych skanerów wymaga się prostoty obsługi, niewielkich rozmiarów i rozdzielczości wystarczającej do otrzymania dobrych cyfrowych kopii fotografii. Najczęściej w takich przypadkach wybiera się tańsze skanery CIS. Przeanalizuj specyfikację przykładowego skanera (rys. 59.8.). Przykładowe urządzenie zgodnie ze specyfikacją ma bardzo dobre parametry. Jednak tabelka z danymi nie przedstawia wszystkich zalet tego skanera. Jedną z nich jest to, że model nie wymaga dodatkowego zasilacza, a energię czerpie ze złącza USB. Inne cechy można odczytać z oprogramowania dołączonego do urządzenia. W wielu skanerach możliwe jest skanowanie uruchamiane wciśnięciem klawisza w obudowie, automatyczne tworzenie wielostronicowych dokumentów PDF, automatyczna korekta, a także uzyskanie osobnych plików dla zdjęć zeskanowanych w czasie jednego przebiegu. Warto więc wczytać się w opis udostępniany przez producenta lub sprzedawcę.

Specyfikacja

| | |
|---|---|
| Typ skanera | Płaski |
| Typ sensora | CIS |
| Interfejs | USB |
| Rozdzielczość optyczna | 4800 x 4800 dpi |
| Prędkość skanowania w kolorze | 6 str./min |
| Prędkość skanowania w czerni | 10 s |
| Głębia koloru (wejście) | 48-bitowa |
| Głębia koloru (wyjście) | 48-bitowa |
| Skala szarości (wejście) | 16 bitów |
| Skala szarości (wyjście) | 8 bitów |
| Obsługiwane nośniki | A4<br>A5<br>A6<br>B5<br>B6<br>Pocztówki<br>Wizytówki |

Wysoka rozdzielczość.

Dokument A4 z rozdzielczością 300 dpi zostanie zeskanowany w czasie 10 s.

Każdy z kolorów RGB jest skanowany i zapisywany z rozdzielczością 16 bitów.

Można skanować dokumenty o dowolnych wymiarach równych A4 lub mniejszych.

Rys. 59.8. Parametry przykładowego skanera CIS

Powyższe rozważania dotyczyły głównie skanerów płaskich. Istnieją także skanery ręczne, które można zabrać chociażby do biblioteki i skanować bez udziału komputera. Niektóre skanują jednocześnie całą powierzchnię, na przykład albumu A4, inne pojedyncze wiersze tekstu, co w połączeniu z programem OCR pozwala zamienić obraz na tekst edytowalny w komputerze.

## ZADANIA DO WYKONANIA

1. Określ rozmiary rysunku wydrukowanego z rozdzielczością 300 dpi, jeśli jego rzeczywisty rozmiar to 6000 na 1500 punktów. Jaką rozdzielczość dpi należy dobrać, by wydruk miał rozmiar poziomy 10 cali? Wyniki swoich obliczeń sprawdź w internecie (poszukaj strony z kalkulatorem rozdzielczości).
2. Ustal, dla jakiej rozdzielczości pionowej proporcje ekranu będą równe 16:10, jeśli pozioma jest równa 1920 pikseli.
3. Oblicz, dla jakiej rozdzielczości poziomej proporcje ekranu będą równe 4:3, jeśli pionowa jest równa 768 pikseli.
4. Załóż, że do realizacji swojego hobby potrzebujesz monitora o minimalnej wielkości ekranu 27 cali i rozdzielczości WQHD. Znajdź najkorzystniejszą ofertę w sklepach internetowych. Uzasadnij swój wybór, biorąc pod uwagę wiarygodność sprzedawcy (sprawdź opinie), cenę i sposób płatności oraz dostawy.
5. Oblicz liczbę kolorów, jaką można otrzymać, skanując dokument z 48-bitową głębią kolorów. Oceń wynik i określ, jaki ma to wpływ na jakość skanowanych zdjęć.
6. Ile odcieni szarości można uzyskać, skanując dokument z 16-bitową głębią? Czy zapisanie go po przetworzeniu na 8-bitową głębię pogorszy jakość powstałego obrazu?

## PODSUMOWANIE LEKCJI

s. 341 ▸ • Wartość podana jako **DPI** oznacza liczbę wydrukowanych punktów na długości jednego cala.

s. 342 ▸ • **Przy wyborze urządzenia** zwróć uwagę przede wszystkim na jego istotne cechy i parametry. Następnie oceń dodatkowe zalety, np. kolor, kształt, dołączone oprogramowanie. Znajomość **parametrów urządzeń peryferyjnych** pozwala na dokonanie świadomego wyboru urządzenia odpowiedniego do potrzeb i preferencji użytkownika. **Dane techniczne** można znaleźć na stronach producentów i sprzedawców sprzętu.

s. 344 ▸ • Zanim dokonasz **wyboru monitora**, sprawdź, czy twoja karta grafiki ma **wymagany interfejs** i czy potrafi wygenerować obraz o natywnej rozdzielczości monitora.

# 60. Projektuję rozbudowę i zakup nowego zestawu

**NA TEJ LEKCJI:**

- poznasz najważniejsze cechy komputera przenośnego i stacjonarnego;
- dowiesz się, na co zwracać uwagę przy zakupie komputera.

Dzięki obecnemu rozwojowi technologicznemu zastosowanie komputerów czy urządzeń elektronicznych jest powszechne. Wymusza wręcz na nas korzystanie z nich w życiu codziennym. Postęp w rozwoju technologicznym, a co za tym idzie dostosowanie oprogramowania nakłada na nas konieczność regularnej wymiany sprzętu komputerowego. Zestawy komputerowe stacjonarne czy przenośne zakupione 4–5 lat temu są już przestarzałe technologicznie, a część oprogramowania może w ogóle na nich nie działać.

Przyzwyczailiśmy już się do tego, że „żywotność" telefonu komórkowego nie jest dłuższa niż 2 lata. Tak szybki i intensywny rozwój urządzeń i technologii stawia przed nami co jakiś czas zadanie wymiany sprzętu na nowy bądź rozbudowę używanego do tej pory zestawu.

## 60.1. Stacjonarny czy przenośny – dylemat człowieka XXI wieku

komputer stacjonarny i przenośny

Dwadzieścia lat temu przy zakupie zestawu komputerowego takiego dylematu nie było. Uważano, że warto kupować komputery stacjonarne, najlepiej z drukarką, skanerem i zestawem głośników w technologii Surround 5.1. Dzisiaj, kupując sprzęt komputerowy, też nie mamy takiego dylematu – wiadomo, że trzeba kupić komputer przenośny, najlepiej o wadze do 1,5 kg, dostosowany do korzystania z sieci i gier – krótko mówiąc, laptop powinien być „gamingowy" (czyt. gejmingowy, od ang. game – gra). Oczywiście za komputer spełniający powyższe warunki, zarówno dawniej, jak i obecnie, należy zapłacić odpowiednią cenę.

Wymieniając sprzęt komputerowy na nowy, zwracamy uwagę na jego komponenty: płytę główną, procesor, pamięć, kartę graficzną, kartę sieciową, dysk twardy, zasilacz, monitor, klawiaturę i urządzenia wskazujące, na przykład mysz. Przed zmianą sprzętu komputerowego na nowy musimy rozważyć wszystkie za i przeciw.

komputery przenośne

Oczywiście komputer przenośny powinien charakteryzować się sprawnie działającym systemem operacyjnym i oprogramowaniem biurowym, gdyż na tego typu sprzęcie będziemy korzystać przede wszystkim z tego oprogramowania. W laptopie ważne są również łączność sieciowa i możliwość wielogodzinnej pracy na zasilaniu baterią. Jak już decydujemy się na komputer przenośny, będziemy szukali sprzętu lekkiego, o wymiarach dostosowanych do naszych potrzeb. Najczęściej

wybieranymi laptopami są te o przekątnych ekranu wielkości 14" i 15", ale również sporym powodzeniem cieszą się mniejsze – 10" i 13", a także laptopy z monitorami o przekątnej 17". Rozwój technologiczny spowodował również to, że obecnie na rynku jest coraz trudniej kupić laptop z wbudowanym napędem optycznym – planując zakup nowego sprzętu, również to musimy wziąć pod uwagę.

Jeśli szukamy komputera do gier – zarówno sprzętu przenośnego, jak i zestawu stacjonarnego – musimy zwrócić uwagę na wydajność całego zestawu, a zwłaszcza karty graficznej, pojemność dysku twardego oraz wielkość monitora.

## 60.2. Misja „nowy komputer"

analiza potrzeb przed kupnem nowego komputera

Przed podjęciem decyzji o zakupie nowego komputera powinniśmy określić swoje potrzeby. Jeżeli chcemy tylko korzystać z internetu i robić drobne prace w arkuszu kalkulacyjnym, to w zupełności wystarczy nam zwykły laptop o niewygórowanych parametrach. Jeżeli planujemy zająć się obróbką multimediów lub grać na nowym komputerze, to wymagania sprzętowe są o wiele wyższe, a co za tym idzie – także cena sprzętu.

Tab. 60.1. Tabela analizy potrzeb projektowanego komputera

| Komponenty | Definicja | Kluczowe cechy |
|---|---|---|
| CPU – procesor | Odpowiedzialny za przetwarzanie informacji. Jego szybkość ma kluczowy wpływ na szybkość komputera. | – ilość rdzeni<br>– częstotliwość procesora mierzona w gigahercach (GHz) |
| Pamięć RAM | Służy do tymczasowego przechowywania aktualnie używanych danych. Im większa pamięć RAM, tym komputer szybciej pracuje (systemy operacyjne 32-bitowe nie mogą używać więcej niż 4 GB pamięci RAM). | ilość pamięci mierzona w gigabajtach (GB) |
| Dysk twardy – HDD lub SSD | Odpowiedzialny za przechowywanie danych. Istnieją dwa rodzaje dysków twardych:<br>– dysk magnetyczny HDD (w zamkniętej obudowie z wirującym talerzem oraz głowicą),<br>– dysk półprzewodnikowy SSD (nie ma ruchomych części, dyski takie są szybsze i cichsze, ale znacznie droższe). | pojemność dysku mierzona w gigabajtach (GB) bądź w terabajtach (TB) |
| Połączenie z internetem | Komputer łączy się z internetem poprzez:<br>– port Ethernet (podłączenie do routera),<br>– adapter bezprzewodowy albo kartę bezprzewodową. | łączność przewodowa lub bezprzewodowa |
| Porty | Dzięki portom urządzeń komputer łączy się z innymi urządzeniami, np. klawiaturą, drukarką lub dyskiem zewnętrznym. Najpopularniejsze rodzaje portów to:<br>– USB – obecnie popularne standardy to USB 2.0 i USB 3.0,<br>– Firewire – znany również jako IEEE 1394 lub iLink – zapewnia szybsze połączenie do szybszego transferu danych.<br>Porty audio i wideo umożliwiają podłączenie głośników i ekranu. | urządzenia, jakie podłącza się do komputera |
| Karta graficzna | Pozwala na przetwarzanie i wyświetlanie informacji na ekranie. Wyróżniamy:<br>– zintegrowaną kartę graficzną, która jest wbudowana w płytę główną lub procesor – używa ona zazwyczaj pamięci komputera,<br>– odpowiednią kartę graficzną, która ma swoją pamięć. | ilość pamięci systemowej (RAM) |

| Komponenty | Definicja | Kluczowe cechy |
|---|---|---|
| Napędy optyczne | Pozwalają na tworzenie informacji i nagrywanie ich na płyty CD, DVD i czytniki Blu-ray. | zależnie od używanych nośników |
| Urządzenia dodatkowe | – monitor,<br>– klawiatura,<br>– urządzenia wskazujące,<br>– drukarki, skanery i inne urządzenia dodatkowe. | rozdzielczość wyświetlacza monitora, drukarka atramentowa lub laserowa |
| Baterie i pobór prądu | Niektóre laptopy mają dodatkową baterię. Są one wtedy większe i cięższe, ale mogą znacznie dłużej pracować bez ładowania baterii. W niektórych modelach komputerów przenośnych nie ma możliwości samodzielnej wymiany baterii. | czas pracy baterii |

Ostatnim krokiem w planowaniu zakupu nowego sprzętu lub rozbudowy już używanego jest udzielenie odpowiedzi następujące pytania:

- Czy wybrany przez nas produkt spełnia moje oczekiwania?
- Co wiem na temat firmy, która wyprodukowała dany komputer, czy ma dobrą reputację?
- Jakie opinie ma wybrany przez nas model komputera?
- Jaką gwarancję mogę otrzymać od producenta lub sprzedawcy i jak długo ona mi przysługuje?
- Jakie komponenty wchodzą w skład mojego komputera?
- Jakie oprogramowanie jest już zainstalowane na nowym komputerze?

## ZADANIA DO WYKONANIA

1. Przeanalizuj komponenty własnego komputera lub komputera, na którym pracujesz w szkole. Użyj programu diagnostycznego, np. Everest Home Edition.
2. Na podstawie analizy z zadania 1. omów obszar zastosowań twojego komputera. Uzasadnij i określ, co można zmodernizować w tym sprzęcie. Wskaż parametry, wyznaczając granicę opłacalności takich ulepszeń.
3. Zaprojektuj zestaw komputerowy będący kompromisem pomiędzy wydajnością a ceną. Sporządź kosztorys kilku wersji własnego projektu.

## PODSUMOWANIE LEKCJI

- Każde urządzenie wyposażone w procesor możemy nazwać **komputerem**. ▸ s. 349
- Komputery dzielimy na **stacjonarne i przenośne** (laptopy, tablety, netbooki). ▸ s. 349
- Najważniejszą **cechą komputera PC** jest jego łączność z internetem. ▸ s. 349
- **Komputery przenośne** powinny mieć wydajną baterię, która zapewni im długą pracę bez ponownego ładowania. ▸ s. 349
- Przed zakupem nowego sprzętu powinniśmy dokonać **analizy potrzeb.** ▸ s. 350

## PODSUMOWANIE PRZED SPRAWDZIANEM

### 53. Nie wszystko jest takie oczywiste, czyli jak działa internet

s. 317 ▸ • **IP, TCP, HTTP, FTP** i **SMTP** to protokoły, bez których sieci i internet nie mogłyby funkcjonować.

s. 321 ▸ • **Adres symboliczny,** np. strony internetowej, w rzeczywistości służy jedynie wygodzie użytkowników. W sieci funkcjonuje jako adres **IP**. Za ich zamianę odpowiadają **DNS**.

### 54. Zabezpieczenia firewall

s. 323 ▸ • **Zapora sieciowa**, nazywana również zaporą ogniową, ma za zadanie chronić nasz komputer czy też sieć komputerową przed atakami z zewnątrz bądź nieautoryzowanym dostępem do naszych zasobów.

s. 323 ▸ • **NAT** jest techniką przesyłania ruchu sieciowego przez router, która wiąże się ze zmianą źródłowych lub docelowych adresów IP.

s. 324 ▸ • Jedną z ważniejszych funkcji zapory sieciowej jest **filtrowanie ruchu**, który odbywa się w lokalnej sieci komputerowej.

s. 324 ▸ • Podstawą bezpieczeństwa sieci komputerowej jest zapewnienie odpowiedniej **polityki bezpieczeństwa**, która obejmuje nie tylko systemy teleinformatyczne.

### 55. Sieć to nie tylko internet, czyli poznajemy topologię sieci komputerowych

s. 325 ▸ • Połączenie twojego **telefonu** z komputerem za pośrednictwem **Bluetooth** jest siecią typu **P2P.**

s. 326 ▸ • Najpopularniejsza topologia sieci to **topologia gwiazdy.**

### 56. Warstwowy model sieci komputerowej

s. 330 ▸ • Etapy komunikacji w sieci są opisane poprzez **modele warstwowe sieci** – ISO/OSI i TCP/IP.

s. 330 ▸ • **Model ISO/OSI** składa się z siedmiu warstw i jest przygotowany z myślą o tworzeniu tzw. systemu otwartego, który nie będzie należał do żadnej zamkniętej sieci.

s. 331 ▸ • **Model TCP/IP** został stworzony tak, aby był odporny na ataki sieci komputerowych.

s. 331 ▸ • **Model TCP/IP** stał się podstawową strukturą internetu.

s. 331 ▸ • **Model TCP/IP** składa się tylko z czterech warstw.

### 57. Funkcje urządzeń i protokoły

s. 333 ▸ • **Sieć** składa się z wielu elementów, ale do najważniejszych należą sprzęt i oprogramowanie, dzięki któremu będzie ona funkcjonować.

s. 333 ▸ • Urządzenia niezbędne do obsługi i funkcjonowania sieci możemy podzielić na **trzy grupy**:
- urządzenia transmisji,
- urządzenia dostępu,
- urządzenia wzmacniania przesyłanych sygnałów.

s. 334 ▸ • Każda warstwa w modelu ISO/OSI i TCP/IP ma swoje **protokoły komunikacyjne**.

### 58. Kto tam, czyli identyfikujemy komputery w sieci

s. 336 ▸ • **ipconfig/all** to polecenie systemowe wyświetlające stan sieci i karty sieciowej.

s. 337 ▸ • **Poprawność połączenia** sieciowego można sprawdzić poleceniem **ping.**

s. 339 ▸ • **Konfigurację połączenia sieciowego** w Windows wykonuje się w opcji **Ustawienia.**

### 59. Kupujemy świadomie, czyli poznajemy parametry urządzeń peryferyjnych

s. 342 ▸ • Wybór **drukarki** zależy od wielu czynników, np. **rozdzielczości, techniki druku,** szybkości.

s. 344 ▸ • Trafny wybór odpowiedniego urządzenia peryferyjnego zależy od znajomości jego parametrów.

s. 345 ▸ • **TN, VA, IPS** to różne rodzaje wyświetlaczy LCD monitorów komputerowych.

s. 346 ▸ • **VGA, HDMI, DisplayPort, DVI** oznaczają podstawowe rodzaje złączy dla **monitorów.**

### 60. Projektuję rozbudowę i zakup nowego zestawu

s. 349 ▸ • Każde urządzenie wyposażone w procesor możemy nazwać **komputerem.**

s. 349 ▸ • Komputery dzieli się na **stacjonarne** i **przenośne** (laptopy, tablety, netbooki).

s. 349 ▸ • Najważniejszą cechą komputera PC jest jego łączność z internetem.

s. 349 ▸ • Komputer przenośny powinien mieć wydajną baterię, która zapewni mu długą pracę bez ponownego ładowania.

s. 350 ▸ • Przed dokonaniem zakupu nowego sprzętu powinniśmy dokonać **analizy potrzeb.**