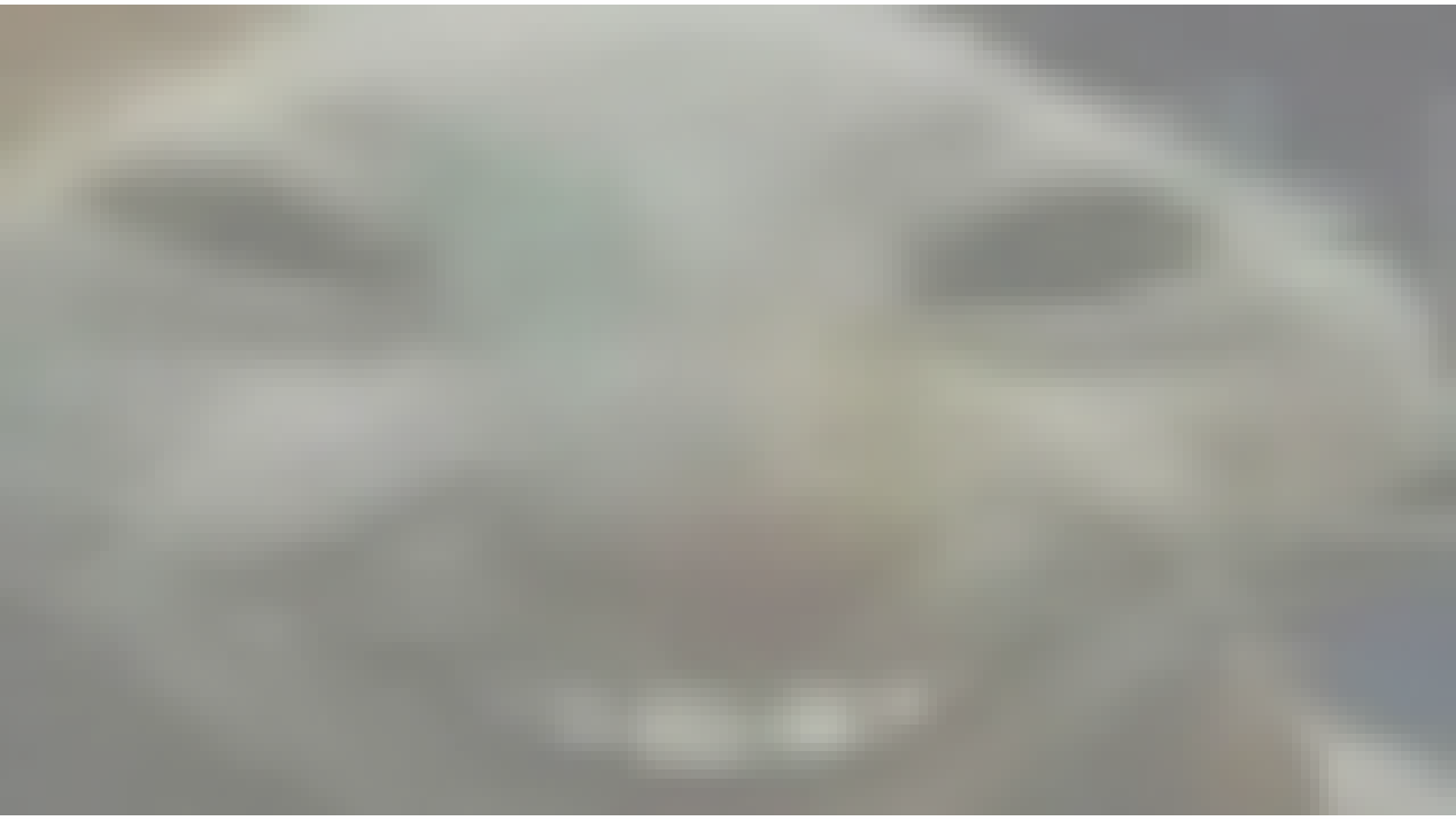
DLA ABSOLWENTÓW SZKÓŁ PODSTAWOWYCH

Informatyka na czasie 2

Podręcznik dla liceum ogólnokształcącego i technikum

Zakres podstawowy

nowa era



Janusz Mazur
Paweł Perekietka
Zbigniew Talaga
Janusz S. Wierzbicki

Informatyka na czasie 2

Podręcznik dla liceum ogólnokształcącego i technikum

Zakres podstawowy

nowa era Twoje mocne strony

Informatyka na czasie

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, na podstawie opinii rzeczoznawców:
mgr inż. Anny Koludo, dr. hab. Henryka Nogi, mgr Teresy Zawiszy-Chlebowskiej.

Etap edukacyjny: III
Typ szkoły: liceum ogólnokształcące i technikum
Rok dopuszczenia: 2020

Numer ewidencyjny w wykazie MEN: 990/2/2020

Podręcznik został opracowany na podstawie *Programu nauczania informatyki w liceach i technikach. „Informatyka na czasie". Zakres podstawowy* autorstwa Janusza Mazura.

Nabyta przez Ciebie publikacja jest dziełem twórcy i wydawcy. Prosimy o przestrzeganie praw, jakie im przysługują. Zawartość publikacji możesz udostępnić nieodpłatnie osobom bliskim lub osobiście znanym, ale nie umieszczaj jej w internecie. Jeśli cytujesz jej fragmenty, to nie zmieniaj ich treści i koniecznie zaznacz, czyje to dzieło. Możesz skopiować część publikacji jedynie na własny użytek.

Szanujmy cudzą własność i prawo. Więcej na **www.legalnakultura.pl**

© Copyright by Nowa Era Sp. z o.o. 2020
ISBN 978-83-267-3885-2

Konsultacja merytoryczno-dydaktyczna: Beata Rząca, Paweł Mateja, Paweł Koszałka.

Redakcja merytoryczna: Przemysław Szydzik.
Współpraca redakcyjna: Anna Nasiadka, Dorota Rzeszewska, Joanna Sawicka, Krzysztof Fit, Agnieszka Sieczak.
Redakcja językowa: Marta Zuchowicz, Paulina Szulim.
Nadzór artystyczny: Kala Pichler.
Opieka graficzna: Ewa Kaletyn.
Projekt okładki: Maciej Galiński.
Projekt graficzny: Sławomir Włodarczyk.
Opracowanie graficzne: Sławomir Włodarczyk, Studio Graficzne Fastwork Grzegorz Afeltowicz.
Ilustracje: Sławomir Włodarczyk, Natalia Helman.
Realizacja projektu graficznego: Artur Polakowski.
Fotoedycja: Katarzyna Iwan-Malawska.

Nowa Era Sp. z o.o.
Aleje Jerozolimskie 146 D, 02-305 Warszawa
www.nowaera.pl, e-mail: nowaera@nowaera.pl, tel. 801 88 10 10

Druk i oprawa: Walstead Central Europe

Spis treści

1. Arkusz kalkulacyjny i bazy danych 7

Kartka z historii 8

1. Jak pobierać dane do arkusza kalkulacyjnego 10

1.1. Zasady pracy w arkuszu kalkulacyjnym 10
1.2. Formuły i funkcje w arkuszu kalkulacyjnym 11
1.3. Typy adresów w arkuszu kalkulacyjnym 14
1.4. Kopiowanie danych i formuł 14
1.5. Podstawowe funkcje arkusza kalkulacyjnego 16
1.6. Importowanie danych do arkusza kalkulacyjnego 18
1.7. Przetwarzanie importowanych danych 23

2. Wyciągamy wiedzę z danych 30

2.1. Dane w arkuszu kalkulacyjnym 30
2.2. Tabele przestawne w programie Excel 35
2.3. Wykresy przestawne 50

3. Korespondencja seryjna 54

3.1. Czym jest korespondencja seryjna? 54
3.2. Korespondencja seryjna w programie Word 55
3.3. Filtrowanie adresatów i personalizacja listów 59
3.4. Zarządzanie danymi korespondencji seryjnej w arkuszu kalkulacyjnym 62
3.5. Kończenie pracy z korespondencją seryjną 65

4. Relacyjne bazy danych 67

4.1. Projektowanie bazy danych 67
4.2. Od projektu do bazy danych 71
4.3. Formularze – zarządzanie danymi i ich przeglądanie 80
4.4. Kwerendy – wyszukiwanie i łączenie danych 82
4.5. Raporty – wizualne przedstawienie danych 84

Z informatyką w przyszłość 88

Projekt zespołowy 89

2. Algorytmika i programowanie w języku C++ 91

Kartka z historii 92

5. Od problemu do programu 94
   5.1. Algorytm, czyli rozwiązanie problemu 94
   5.2. Od algorytmu do programu w języku C++ 99

6. Wyszukujemy i sumujemy 112
   6.1. Szukamy największej liczby. Pętla while 112
   6.2. Sumujemy liczby. Pętla for 118
   6.3. Wyznaczamy noty sędziowskie 120

7. Binarny system liczbowy 126
   7.1. Liczby dziesiętne i dwójkowe 126
   7.2. Cyfrowy zapis informacji 128
   7.3. Algorytmy zamiany reprezentacji liczb 132
   7.4. Program zamieniający liczbę dziesiętną na binarną 134

8. Czy ta liczba jest pierwsza? 141
   8.1. Liczby złożone i liczby pierwsze 141
   8.2. Podzielność liczb 145
   8.3. Test pierwszości liczby 148

9. Algorytm Euklidesa i działania na ułamkach 153
   9.1. Patchworkowe kołdry 153
   9.2. Największy wspólny dzielnik 154
   9.3. Działania na ułamkach zwykłych 161

Z informatyką w przyszłość 168

3. Algorytmika i programowanie w języku Python 169

Kartka z historii 170

10. Od problemu do programu 172
   10.1. Algorytm, czyli rozwiązanie problemu 172
   10.2. Od algorytmu do programu w języku Python 177

11. Wyszukujemy i sumujemy 189
   11.1. Szukamy największej liczby. Pętla while 189
   11.2. Sumujemy liczby. Pętla for 194
   11.3. Wyznaczamy noty sędziowskie 198

12. Binarny system liczbowy 203
   12.1. Liczby dziesiętne i dwójkowe 203
   12.2. Cyfrowy zapis informacji 205
   12.3. Algorytmy zamiany reprezentacji liczb 209
   12.4. Program zamieniający liczbę dziesiętną na binarną 211

13. Czy ta liczba jest pierwsza? 218
13.1. Liczby złożone i liczby pierwsze 218
13.2. Podzielność liczb 222
13.3. Test pierwszości liczby 225
14. Algorytm Euklidesa i działania na ułamkach 230
14.1. Patchworkowe kołdry 230
14.2. Największy wspólny dzielnik 231
14.3. Działania na ułamkach zwykłych 238
Z informatyką w przyszłość 244

Dodatki 245
1. Podstawowe funkcje programu Microsoft Excel 246
2. Typy danych i maski w programie Microsoft Access 248
3. Jak zaprojektować i utworzyć bazę danych 249
4. Sposoby zapisu algorytmów 250
5. Logika matematyczna i operatory logiczne 252
6. Typy danych w języku C++ 255
7. Podstawowe typy instrukcji w języku C++ 257
8. Operatory i funkcje matematyczne w języku C++ 259
9. Słowa kluczowe w języku C++ 260
10. Typy danych w języku Python 261
11. Podstawowe typy instrukcji w języku Python 262
12. Operatory i funkcje matematyczne w języku Python 265
13. Słowa kluczowe w języku Python 266
14. Systemy liczbowe 267

Indeks 268

O czym jest podręcznik?

Podręcznik *Informatyka na czasie 2* pomoże ci pogłębić wiedzę na temat przetwarzania danych w arkuszu kalkulacyjnym. Dowiesz się, do czego wykorzystuje się bazy danych, i zaprojektujesz własną bazę danych. Nauczysz się też pisać proste programy komputerowe w języku C++ lub Python 3.



Czemu służą poszczególne elementy podręcznika?

Cele lekcji

Informacje na temat tego, czym będziesz się zajmować na danej lekcji.

Zapamiętaj

Najważniejsze pojęcia omawiane na lekcji, które należy zapamiętać.

Ćwiczenie 1

Proste zadania w toku lekcji. Odpowiedzi nie należy zapisywać w podręczniku.

Dobra rada

Podpowiedzi ułatwiające wykonanie omawianych czynności.

Warto wiedzieć

Informacje poszerzające wiedzę o omawianych zagadnieniach.

A to ciekawe

Ciekawostki pokazujące omawiane zagadnienia z zaskakującej perspektywy.

Kartka z historii

Wydarzenia najważniejsze dla rozwoju danej dziedziny informatyki.

Z informatyką w przyszłość

Przykłady zastosowania treści omawianych w danym dziale w przyszłym życiu zawodowym.

Oznaczenia w podręczniku

- Odsyłacz do zagadnienia omawianego szerzej w innym miejscu
- Wskaźnik pojęcia; pojęcia obok wskaźników układają się w siatkę pojęć z danej lekcji
- *** Oznaczenie stopnia trudności zadań

Podsumowanie

Najważniejsze informacje z lekcji podane w punktach, które porządkują nowo zdobytą wiedzę.

Zadania

Zestaw poleceń o różnym stopniu trudności. Odpowiedzi nie należy zapisywać w podręczniku.

Projekt zespołowy

Zadanie do wykonania w grupie, które pozwala wykorzystać w praktyce wiedzę z kilku działów.

Dodatki

Przydatne informacje uzupełniające wiedzę z podręcznika.

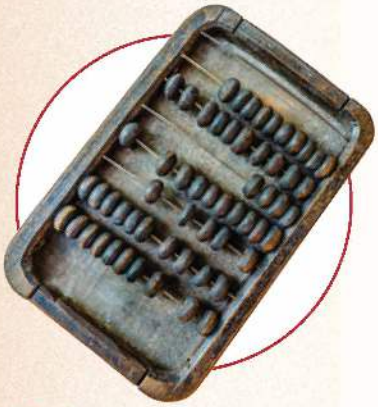


1 Arkusz kalkulacyjny i bazy danych

1. Jak pobierać dane do arkusza kalkulacyjnego
2. Wyciągamy wiedzę z danych
3. Korespondencja seryjna
4. Relacyjne bazy danych

Arkusz kalkulacyjny i bazy danych

Od czasów starożytnych ludzie gromadzą i przetwarzają różnego rodzaju informacje. Wraz z wynalezieniem pieniądza pojawiła się także potrzeba sprawnego wykonywania obliczeń, dlatego zaczęły powstawać kolejne maszyny liczące. Dziś w zarządzaniu informacjami pomagają nam systemy baz danych oraz arkusze kalkulacyjne.



Abakusy wymyślili Sumerowie, prawdopodobnie ok. 2300 r. p.n.e. Początkowo były to linie rysowane na piasku. Z czasem pojawiły się tabliczki z wycięciami oraz znane nam koraliki na patyczkach, czyli liczydła.

2300 r. p.n.e.



W starożytnych bibliotekach gromadzono cenne pisma. Katalogi biblioteczne, z których korzystano, można uznać za pierwsze **bazy danych**. Jedną z najsłynniejszych bibliotek tego czasu była Biblioteka Aleksandryjska – jej zbiory liczyły ok. 5 tysięcy zwojów.

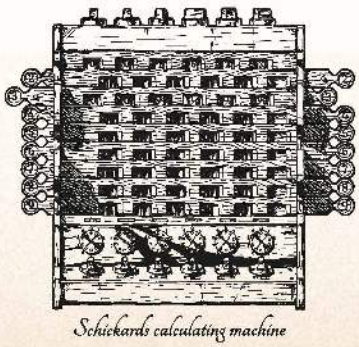
III w. p.n.e

W średniowiecznych kantorach i domach kupieckich używano **stołów obliczeniowych**. Na blacie miały one siatkę, na której układano żetony. Zasada liczenia była zbliżona do liczydła.



średniowiecze

W 1623 r. niemiecki matematyk Wilhelm Schickard skonstruował na zamówienie Jana Keplera drewnianą **maszynę liczącą** do obliczeń astronomicznych. Wykonywała ona cztery podstawowe działania matematyczne.



Schickards calculating machine

1623 r.



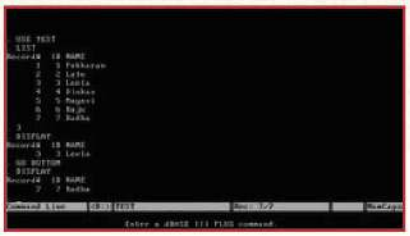
Maszynę licząco-analityczną na karty dziurkowane zbudował ok. 1890 r. inżynier i statystyk Herman Hollerith na potrzeby spisu powszechnego w USA. Sprzedano setki egzemplarzy tego wynalazku, a w 1924 r. firma Holleritha zmieniła nazwę na IBM.

1890 r.



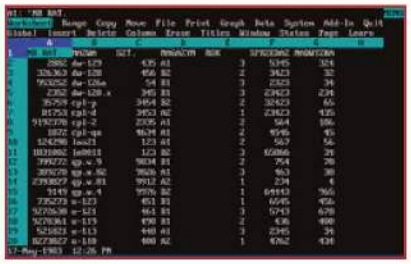
Pierwszy **system zarządzania bazami danych** o nazwie IDS (*Integrated Data Store*) został opracowany w latach 60. XX w. przez Charlesa Bachmana. Za wybitny wkład w rozwój dziedziny baz danych naukowiec otrzymał prestiżową nagrodę Turinga.

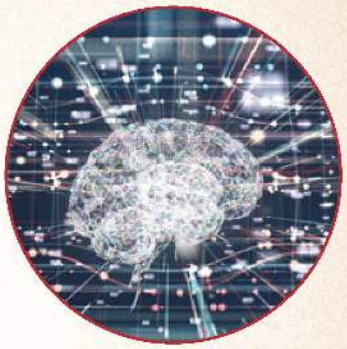
lata 60. XX w.



W 1970 r. Edgar F. Codd przedstawił pracę *A Relational Model of Data for Large Shared Data Banks,* w której opisał **relacyjny model danych**. Jedną z udanych implementacji tego modelu na mikrokomputery był program dBase (na zdjęciu). Na podstawie prac Codda powstał język SQL, który do dziś jest standardem w komunikacji z serwerami relacyjnych baz danych.

Arkusz kalkulacyjny z funkcją rysowania wykresów Lotus 1-2-3 podbił rynek w 1983 r. Był przeznaczony na komputery osobiste IBM z systemem DOS. Pozwalał też na pisanie makr w języku VBA i dopasowanie wyglądu arkusza na wydruku.





W 2019 r. Ośrodek Przetwarzania Informacji jako jedna z pierwszych polskich instytucji publicznych wdrożył technologię **autonomicznej bazy danych**. Jest to baza danych w chmurze obliczeniowej, która wykorzystuje sztuczną inteligencję do zautomatyzowania prac, które wykonywali dotąd administratorzy baz danych (np. tworzenie kopii zapasowych czy aktualizacji).

1970 r.

1979 r.

1983 r.

1985 r.

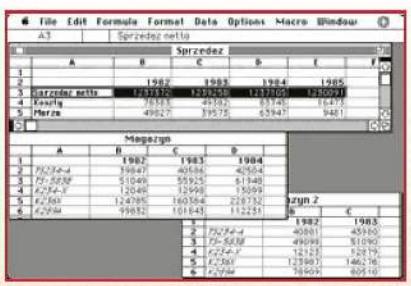
1992 r.

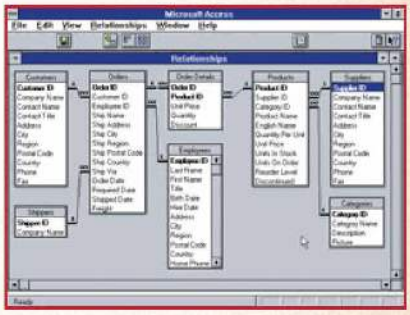
2019 r.



Arkusz kalkulacyjny na komputery osobiste VisiCalc (od ang. *visible calculator* – widoczny kalkulator) stworzyli w 1979 r. Dan Bricklin i Bob Frankston. Mimo ograniczonego rozmiaru arkusza (254 wiersze i 63 kolumny) sprzedano ok. miliona egzemplarzy.

Arkusz kalkulacyjny **Microsoft Excel** powstał w 1985 r. Był przeznaczony na komputery Mac firmy Apple, co pozwoliło na obsługę programu myszką. Wersja na PC pojawiła się dopiero dwa lata później.





W 1992 r. wraz z pakietem Microsoft Office 92 debiutowała pierwsza polska wersja systemu zarządzania bazami danych **Microsoft Access 2.0**.

1. Jak pobierać dane do arkusza kalkulacyjnego

Pozyskiwanie, przyswajanie i przekształcanie informacji pochodzących z różnych źródeł jest podstawą naszego uczenia się. Kiedy pracujesz na komputerze, grasz w gry, zapisujesz kontakty w telefonie, robisz zdjęcia – wytwarzasz i gromadzisz dane. Nawet oglądając telewizję lub rozmawiając z drugą osobą, możesz przetwarzać i zapamiętywać informacje, które potem wykorzystasz w szkole lub w pracy. Dane można dziś łatwo analizować za pomocą komputera i odpowiedniego oprogramowania – m.in. arkuszy kalkulacyjnych. Warto poznać ich możliwości.

Cele lekcji

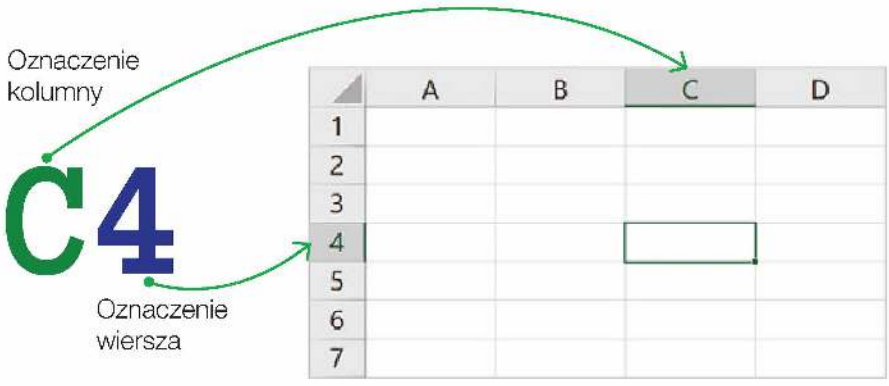
- Przypomnisz sobie zasady pracy w arkuszu kalkulacyjnym.
- Przypomnisz sobie i wykorzystasz podstawowe funkcje arkusza kalkulacyjnego operujące na różnych typach danych.
- Zrozumiesz, dlaczego warto stosować narzędzia wymiany danych.
- Poznasz sposoby importowania danych do arkusza kalkulacyjnego z różnych źródeł.
- Nauczysz się modyfikować dane na etapie importowania do arkusza w zależności od potrzeb.

1.1. Zasady pracy w arkuszu kalkulacyjnym

Skoroszyt to plik, który zawiera jeden lub więcej arkuszy programu Excel.

Arkusz to jedna z zakładek skoroszytu; skoroszyt może mieć wiele arkuszy.

Z arkuszami kalkulacyjnymi pracowaliście już w szkole podstawowej. Na początku przypomnimy kilka podstawowych wiadomości. W programie Microsoft Excel mamy możliwość wyboru pustego **skoroszytu** lub skorzystania z gotowych szablonów. Główne okno programu (s. 12–13) jest podzielone na wiersze i kolumny. Wiersze są oznaczone kolejnymi liczbami naturalnymi, a kolumny – literami alfabetu. Oznaczenia te pozwalają precyzyjnie zidentyfikować każdą komórkę **arkusza**.



Rys. 1.1. Budowa adresu komórki w arkuszu kalkulacyjnym

Adres komórki to litera określająca kolumnę i numer odpowiadający wierszowi. Na rysunku 1.1 wyróżniono komórkę o adresie C4.

Adres komórki

W komórkach arkusza możemy umieszczać dane różnego typu, np. liczby, daty, tekst. Komórki i dane w nich zapisane możemy formatować, czyli modyfikować ich wygląd, po wybraniu z menu kontekstowego opcji **Formatuj komórki...**

Dobra rada

Menu kontekstowe wywołasz, klikając prawym przyciskiem myszy w wybranym miejscu programu. Menu będzie ściśle związane z elementem, który został wskazany przez kliknięcie.

1.2. Formuły i funkcje w arkuszu kalkulacyjnym

Komórki arkusza kalkulacyjnego mogą zawierać również tzw. **formuły**, czyli wzory pozwalające na wykonywanie obliczeń. Istotną zaletą jest to, że wewnątrz formuł możemy umieszczać adresy komórek zamiast ich zawartości (np. liczb lub napisów). Dzięki temu arkusze mogą być uniwersalnym narzędziem obliczeniowym, niezależnym od konkretnych danych. Każda formuła rozpoczyna się od znaku =.

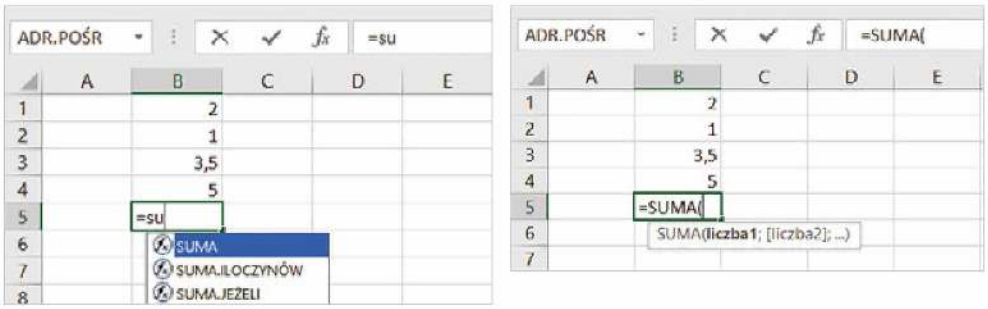
Formuła

Arkusze kalkulacyjne mają bibliotekę wbudowanych **funkcji**, które znacznie ułatwiają tworzenie formuł i pozwalają wykonywać różnego rodzaju obliczenia na różnych typach danych.

Funkcja (arkusz kalkulacyjny)

Jeżeli chcemy np. dodać do siebie liczby znajdujące się w komórkach od B1 do B4 (rys. 1.2), wskazujemy komórkę, w której ma się pokazać wynik, a następnie wpisujemy formułę zawierającą **funkcję** SUMA lub wybieramy ją, klikając **Formuły → Biblioteka funkcji → Wstaw funkcję**.

Funkcja SUMA



Rys. 1.2. Tworzenie formuły z wykorzystaniem funkcji

Dobra rada

Formuły możesz zapisywać bezpośrednio w komórce (pamiętając o znaku =) lub w pasku formuły. Możesz również wyszukać i wybrać funkcję, klikając na ikonę *fx* po lewej stronie paska formuły.

W formule zwykle nie wstawiamy konkretnych wartości liczbowych (choć jest to możliwe), ale korzystamy z adresów komórek, aby formuła była uniwersalna. Możemy też zastosować **zakres adresów**, który zapisujemy z dwukropkiem, np. B1:B4. A zatem w przykładzie z rysunku 1.2 zapis =SUMA(2; 1; 3,5; 5) da taki sam wynik jak zapis =SUMA(B1:B4).

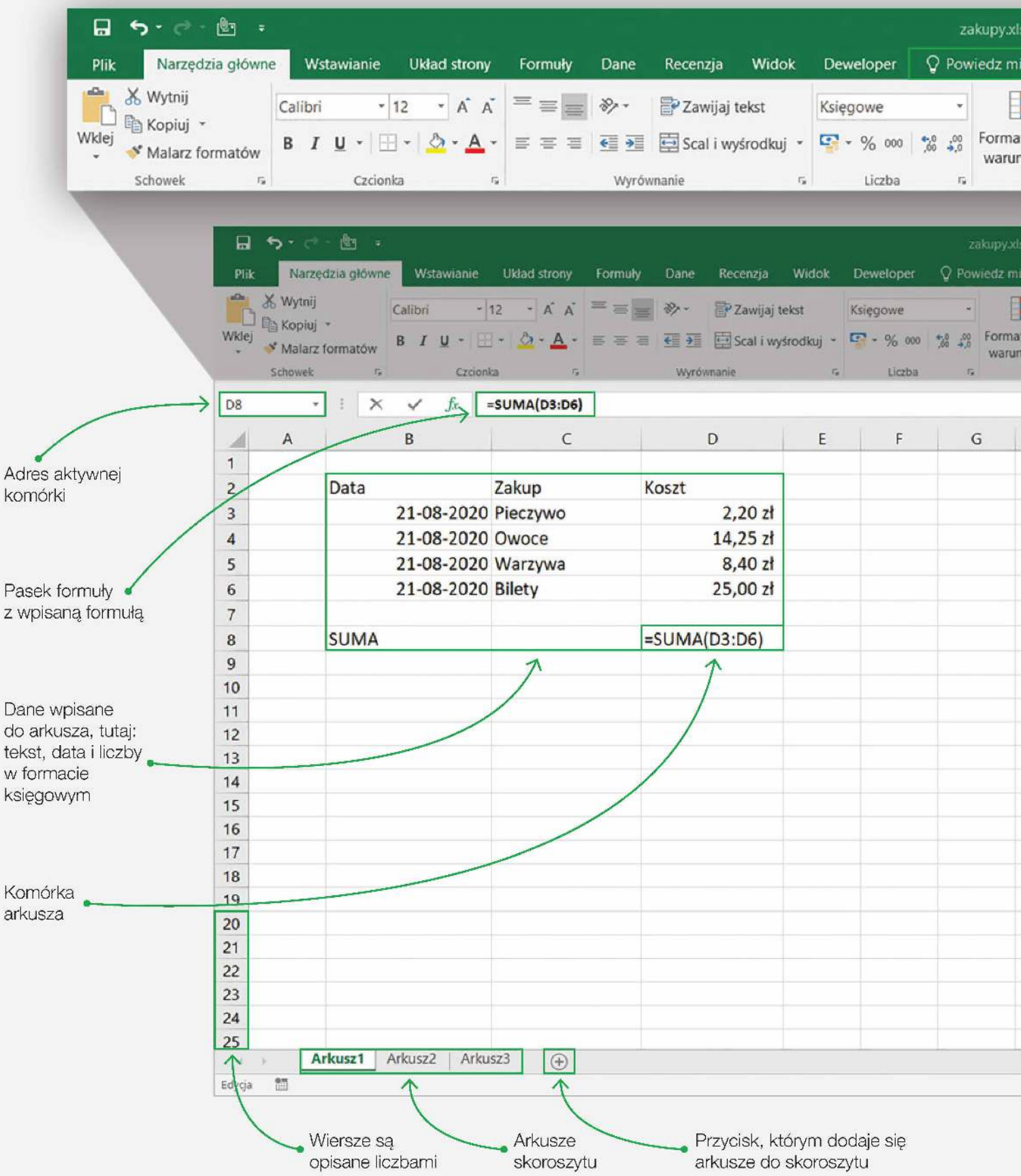
Zakres adresów

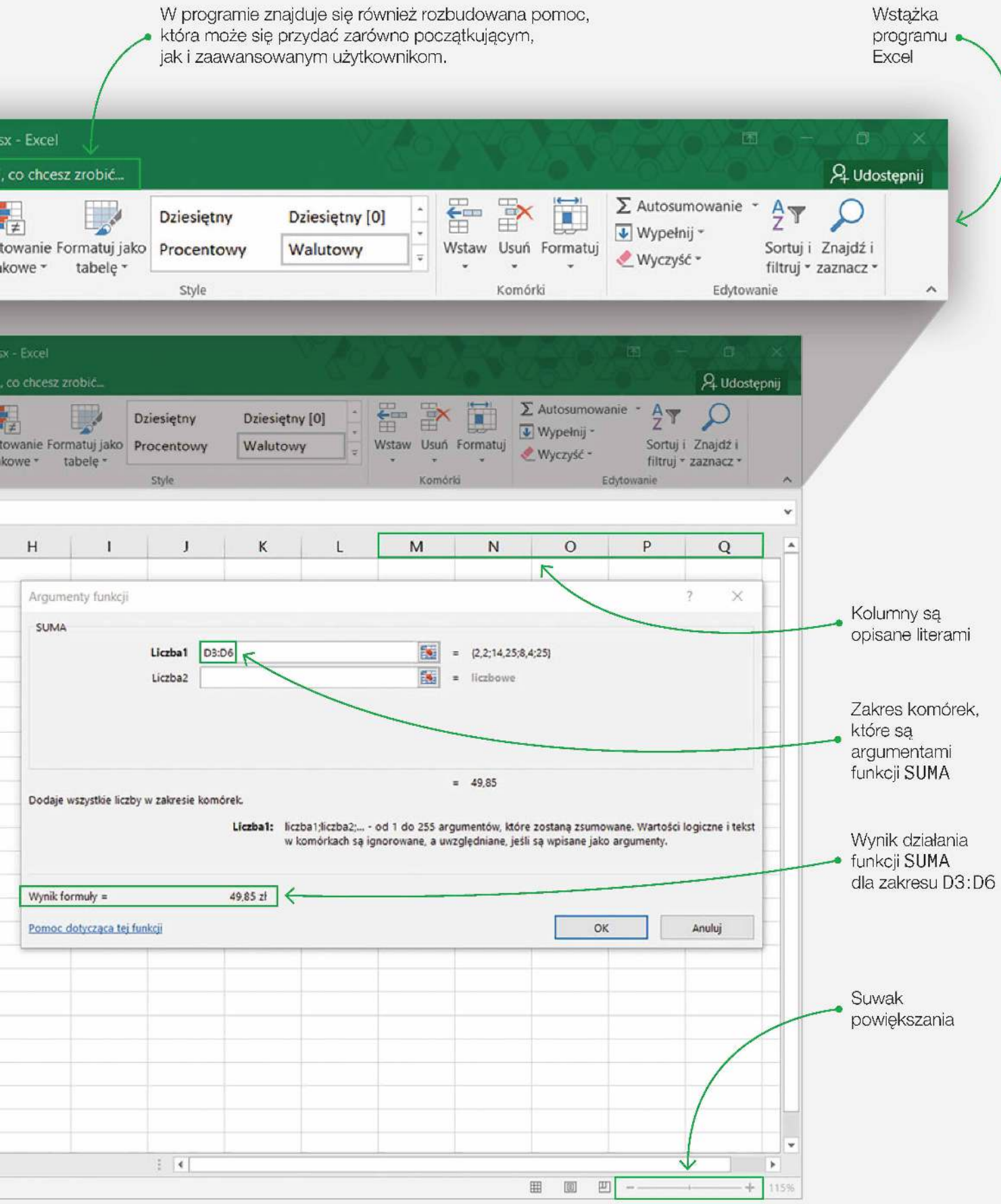
Ćwiczenie 1

W arkuszu kalkulacyjnym wpisz dowolne liczby w dowolnych komórkach. Następnie wybierz dowolną pustą komórkę i zsumuj w niej wstawione wcześniej liczby. Jedną z podanych wcześniej liczb zwiększ o 10 i sprawdź, czy suma się powiększyła.

Okno programu Microsoft Excel 2016

W programie Excel dostępne są karty główne, które grupują narzędzia o podobnym przeznaczeniu. Karta **Narzędzia główne** zawiera podstawowe narzędzia do pracy z arkuszem. Pozwalają one m.in.: kopiować dane, formatować komórki i dane, sortować i filtrować dane w tabelach.




W programie znajduje się również rozbudowana pomoc, która może się przydać zarówno początkującym, jak i zaawansowanym użytkownikom.
Wstążka programu Excel
, co chcesz zrobić...
Udostępnij
Dziesiętny
Dziesiętny [0]
Procentowy
Walutowy
Formatuj jako tabelę
Style
Wstaw
Usuń
Formatuj
Komórki
Autosumowanie
Wypełnij
Wyczyść
Sortuj i filtruj
Znajdź i zaznacz
Edytowanie
H
I
J
K
L
M
N
O
P
Q
Argumenty funkcji
SUMA
Liczba1
D3:D6
(2,2;14,25;8,4;25)
Liczba2
liczbowe
49,85
Dodaje wszystkie liczby w zakresie komórek.
Liczba1: liczba1;liczba2;... - od 1 do 255 argumentów, które zostaną zsumowane. Wartości logiczne i tekst w komórkach są ignorowane, a uwzględniane, jeśli są wpisane jako argumenty.
Wynik formuły = 49,85 zł
Pomoc dotycząca tej funkcji
OK
Anuluj
Kolumny są opisane literami
Zakres komórek, które są argumentami funkcji SUMA
Wynik działania funkcji SUMA dla zakresu D3:D6
Suwak powiększania

1.3. Typy adresów w arkuszu kalkulacyjnym

W formułach możemy wykorzystać różne metody adresowania komórek, czyli wskazywania tych komórek, których chcemy użyć w obliczeniach. Umiejętność zastosowania odpowiedniego typu adresu okazuje się bardzo pomocna, a niekiedy niezbędna do prowadzenia obliczeń w obszernych tabelach arkusza kalkulacyjnego.

Adresowanie względne

Adresowanie względne polega na podawaniu wprost adresu komórki bez żadnych dodatkowych znaków, np. A3, K67, AB23.

Adresowanie bezwzględne

Adresowanie bezwzględne narzuca bezwzględne odwoływanie się do adresowanej komórki. Aby zaadresować komórkę w ten sposób, wstawiamy znak dolara przed nazwą kolumny i numerem wiersza, np. A6, K13.

W praktyce stosuje się również znak dolara tylko w odniesieniu do wiersza albo kolumny, np. $A6 lub A$6. Taki sposób adresowania nazywamy **adresowaniem mieszanym**.

Adresowanie mieszane

Wstawienie znaku $ w adresie komórki powoduje zablokowanie nazwy wiersza lub kolumny albo wiersza i kolumny podczas kopiowania formuły zawierającej ten adres. Kiedy natomiast kopiujemy komórki zawierające adresy względne, adresy komórek zmieniają się zgodnie z kierunkiem kopiowania (w pionie lub poziomie).

W arkuszu możemy się również odwoływać do komórek z innych arkuszy tego samego skoroszytu. Adres komórki z innego arkusza poprzedzamy nazwą arkusza i wykrzyknikiem, np. dla komórki A23 z arkusza o nazwie Arkusz2 odwołanie będzie miało postać =Arkusz2!A23.

1.4. Kopiowanie danych i formuł

Kiedy kopiujemy dane między komórkami lub arkuszami skoroszytu, mamy kilka możliwości ich wklejania. Dzieje się tak dlatego, że komórka kopiowana może zawierać nie tylko surowe dane w postaci liczb,

A to ciekawe

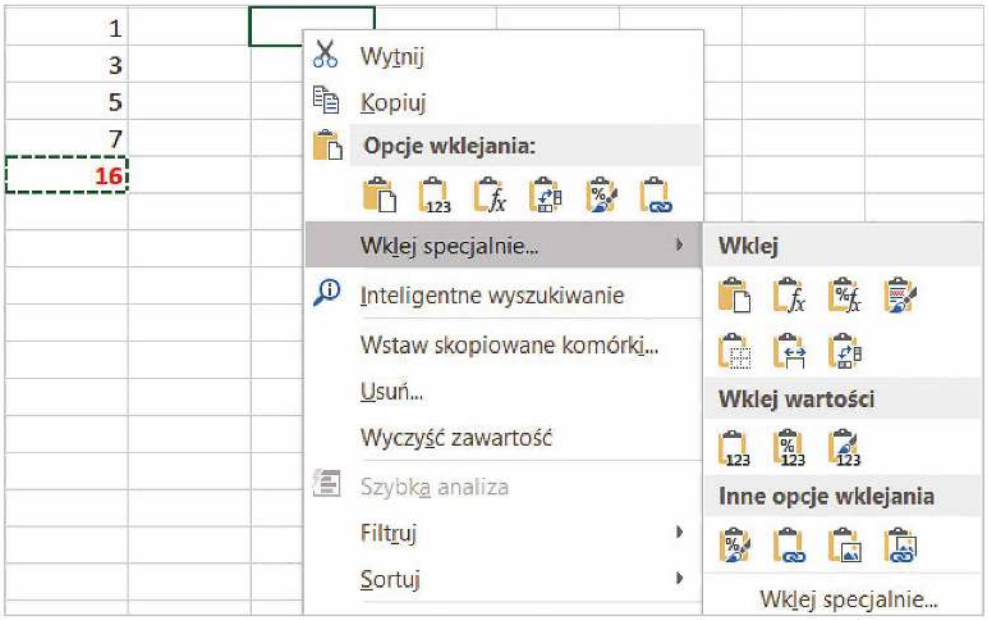
Co potrafi arkusz kalkulacyjny?

Okazuje się, że arkusz kalkulacyjny to nie tylko tabele i formuły. Udowodnił to Japończyk Tatsuo Horiuchi, który w programie Microsoft Excel stworzył wiele pięknych rysunków. Zaczął malować w arkuszu kalkulacyjnym w 2000 r. Miał wówczas 60 lat. Japończyk wykorzystuje podstawowe narzędzia grafiki wektorowej, m.in. krzywe, i składa swoje obrazy z mniejszych elementów. Jego prace można obejrzeć np. na stronie http://www2.odn.ne.jp/~cbl97790/. Artysta udostępnił w internecie również instrukcje pokazujące, jak wykonać rysunki.



napisów lub dat, lecz także np. wyniki złożonych formuł z dodatkowym komentarzem.

Możemy m.in. wkleić tylko wartość z zachowaniem źródłowego **formatowania** lub bez niego, wkleić całą formułę albo wkleić całą formułę wraz z komentarzem. Możliwości te widać po skopiowaniu komórki zawierającej formułę i wybraniu z menu kontekstowego opcji **Wklej specjalnie...** (rys. 1.3).



Rys. 1.3. Możliwości wklejania danych po skopiowaniu komórek

Formuły oraz wartości można również kopiować z wykorzystaniem tzw. **uchwytu wypełniania**, który znajduje się w prawym dolnym rogu zaznaczonej komórki. Po wskazaniu uchwytu wypełniania kursor myszy zamieni się w znak +. Uchwyt należy złapać i przeciągnąć do wybranej komórki w tym samym wierszu lub tej samej kolumnie. Skopiujemy wtedy nie tylko formułę, lecz także adresy komórek będące odwołaniami do jej argumentów.

Uchwyt wypełniania

Kopiując formuły w ten sposób, należy pamiętać, że Excel będzie zmieniał adresy komórek z argumentami, jeżeli nie użyliśmy w formule blokowania komórek, a więc adresowania **bezwzględnego** lub **mieszanego**.

Adresowanie bezwzględne i mieszane, s. 14

Ćwiczenie 2

W komórkach A1:A8 wprowadź dowolne liczby, a w komórce C1 oblicz ich sumę, korzystając z funkcji SUMA. Dodaj w komórce C1 komentarz „Użyto funkcji SUMA". Skopiuj komórkę C1, a następnie za pomocą opcji **Wklej specjalnie...** wklej w komórkach obok:

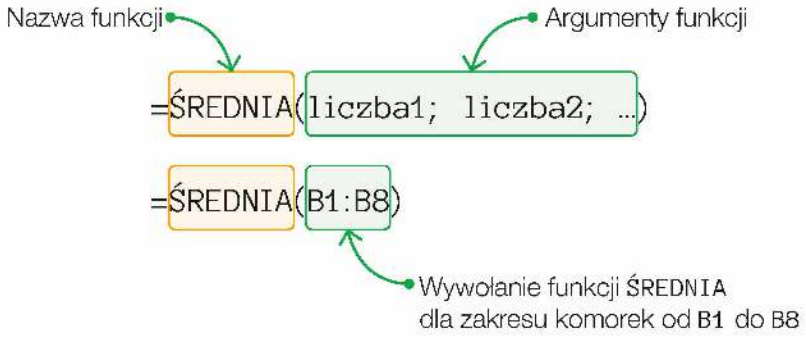
a. tylko wartości,
b. formuły,
c. tylko komentarz.

1.5. Podstawowe funkcje arkusza kalkulacyjnego

Wykorzystaliśmy już funkcję SUMA, pozwalającą na sumowanie liczb. Przypomnijmy teraz kilka kolejnych: ŚREDNIA, MAX, MIN, DŁ i JEŻELI.

Funkcja ŚREDNIA

Funkcja ŚREDNIA umożliwia obliczenie wartości średniej arytmetycznej podanych argumentów. Jej składnia jest identyczna jak dla funkcji SUMA, co przedstawiono na rysunku 1.4.



Rys. 1.4. Budowa funkcji ŚREDNIA

Dobra rada

Nazwy funkcji możesz również zapisywać małymi literami, co w praktyce jest wygodniejsze.

Zastosowanie funkcji ŚREDNIA do wyznaczenia średniej arytmetycznej 8 liczb pokazano na rysunku 1.5.



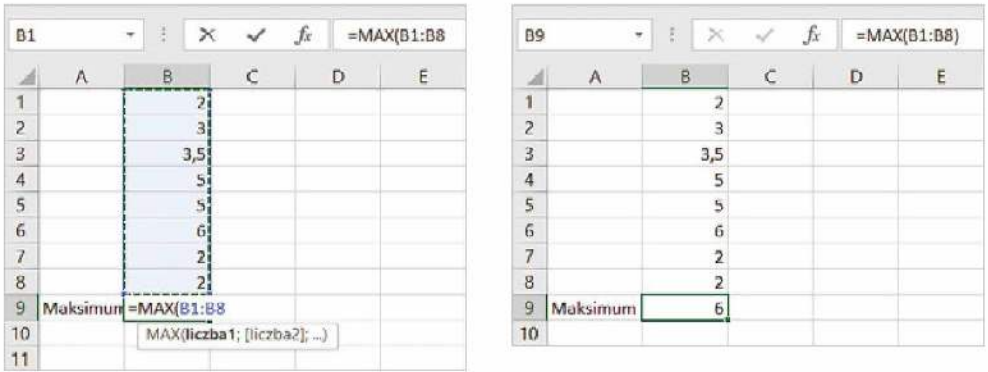
Rys. 1.5. Przykład zastosowania funkcji ŚREDNIA

Funkcja MAX

Funkcja MAX zwraca największą wartość ze zbioru wartości, przy czym ignoruje wartości logiczne oraz tekst (rys. 1.6), natomiast

Funkcja MIN

funkcja MIN zwraca wartość najmniejszą. Obie funkcje mają dokładnie taką samą budowę.



Rys. 1.6. Przykład zastosowania funkcji MAX

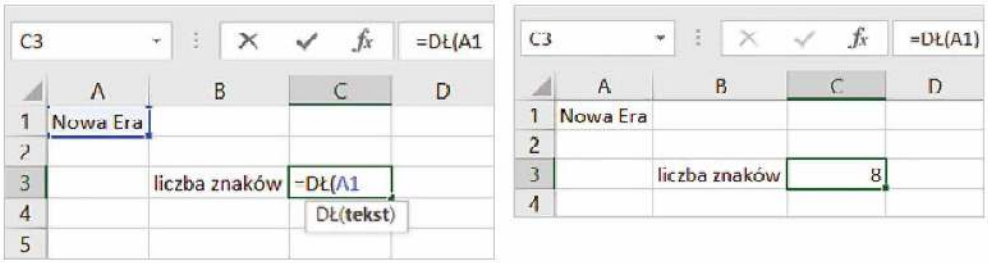
Ćwiczenie 3

Korzystając z pliku otrzymanego od nauczyciela (np. *wzrost.xlsx*) oraz funkcji ŚREDNIA, MIN i MAX, wyznacz:

a. wzrost najwyższego i najniższego ucznia w klasie,

b. średni wzrost oraz różnicę między uczniem najwyższym i najniższym.

Funkcja DŁ zwraca liczbę znaków w tekście łącznie ze spacjami (rys. 1.7). Funkcja DŁ



Rys. 1.7. Przykład zastosowania funkcji DŁ

Warto wiedzieć

Serwis dane.gov.pl zawiera dane ponad 100 instytucji publicznych, które udostępniają pewne informacje obywatelom na podstawie ustawy o dostępie do informacji publicznej. Zasoby w tym serwisie często mają postać plików tekstowych lub plików arkusza kalkulacyjnego.

Ćwiczenie 4

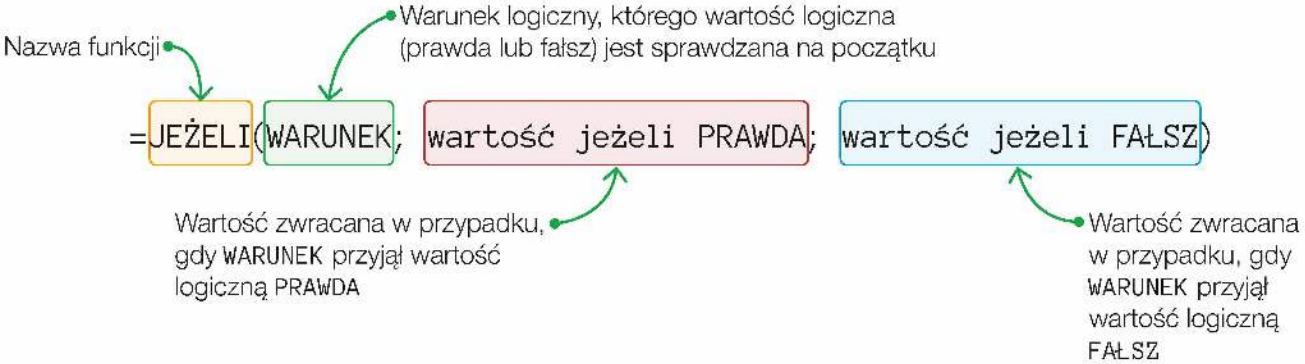
Z serwisu internetowego https://dane.gov.pl pobierz plik XLSX zawierający listę imion nadanych dzieciom w bieżącym roku. Korzystając z poznanych funkcji, wyznacz:

a. długość najdłuższego i najkrótszego imienia,

b. średnią długość imienia.

Funkcja JEŻELI to funkcja, w której zwracana wartość zależy od spełnienia podanego warunku. Funkcja ta należy do grupy funkcji **logicznych** i ma nieco inną budowę niż funkcje dotychczas omówione. Pokazano ją na rysunku 1.8. Funkcja JEŻELI

Logika matematyczna i operatory logiczne, s. 252



Rys. 1.8. Budowa funkcji JEŻELI

Rysunek 1.9 na s. 18 przedstawia przykład zastosowania funkcji JEŻELI do wypisania komunikatu o liczbie wolnych miejsc (lub ich braku) dla kandydatów ubiegających się o miejsce w szkole. Sprawdzany jest warunek C7>C5, a zatem czy liczba podań uczniów przekracza liczbę wolnych miejsc. Dla danych widocznych na rysunku warunek jest spełniony, dlatego wyświetla się komunikat „Za dużo podań".

Dobra rada

Aby szybko zmieniać wielkość widoku danych w arkuszu, możesz nacisnąć klawisz **Ctrl** i kręcić rolką myszy do przodu lub do tyłu.

Warto wiedzieć

Funkcja JEŻELI działa analogicznie do bloku instrukcji warunkowej w języku Scratch.



C9 =JEŻELI(C7>C5; "Za dużo podań"; "Są wolne miejsca")

	A	B	C	D	E	F	G
1							
2							
3		Liczba klas	8				
4		Liczba uczniów w klasie	23				
5		Łączna liczba uczniów	184				
6							
7		Liczba podań	200				
8							
9			Za dużo podań				

Rys. 1.9. Przykład zastosowania funkcji JEŻELI

Dobra rada

Aby w formule umieścić nierówność nieostrą, zastosuj zapis <= lub >=.

Ćwiczenie 5

Aby pokonać grawitację planety, trzeba wystartować z pewną prędkością początkową, zwaną prędkością ucieczki. Znajdź w dostępnych źródłach tę wartość dla Ziemi, a następnie utwórz skoroszyt, w którym po wpisaniu dowolnej prędkości (w km/s) w komórce obok pojawi się odpowiedni komunikat: „Polecisz w kosmos" lub „Po starcie wrócisz na Ziemię".

Zapamiętaj

Wszystkie funkcje programu Excel znajdują się na karcie **Formuły** i są posegregowane według kategorii. Jeżeli znamy nazwy i składnię funkcji, możemy wpisywać je bezpośrednio w komórkach, zaczynając od znaku równości.

1.6. Importowanie danych do arkusza kalkulacyjnego

Większość programów komputerowych oraz aplikacji mobilnych przechowuje i przetwarza różnego rodzaju dane. Często można je wyeksportować, czyli zapisać w plikach. Pliki te mogą mieć różne formaty. Podzielimy je na dwie grupy:

Pliki tekstowe ● ▸ **pliki tekstowe** – zawierają jedynie zakodowane znaki pisma,

Pliki binarne ● ▸ **pliki binarne** – pozostałe pliki, które mogą zawierać dowolne dane, trudne do interpretacji bez odpowiedniego oprogramowania.

Warto wiedzieć

W starszych wersjach programu Excel (2010 i 2013) narzędzia do przetwarzania danych są dostępne jako dodatek Power Query.

Oba typy plików mogą zostać zaimportowane przez inne programy. Mechanizmy eksportowania i importowania umożliwiają wymianę danych między aplikacjami. Z tych mechanizmów korzystamy np. wtedy, gdy chcemy wykonać kopię zapasową konfiguracji smartfona, a następnie skonfigurować nowe urządzenie.

Narzędzia programu Excel pozwalają na importowanie z różnych źródeł różnych typów danych. Mogą to być np. pliki tekstowe (w formatach TXT, CSV) lub strony internetowe.

Importowanie danych z pliku tekstowego

Dane zapisane w pliku tekstowym muszą mieć odpowiednią strukturę, aby można je było poprawnie zaimportować i umieścić w komórkach arkusza kalkulacyjnego.

Powszechnie stosowanym formatem wymiany danych tekstowych jest **format CSV** (ang. *comma-separated values* – wartości rozdzielone przecinkiem). Plik w tym formacie odzwierciedla tabelaryczną strukturę danych. Wiersze w pliku CSV odpowiadają wierszom w tabeli, a dane w wierszu są rozdzielone **separatorem** – wybranym znakiem, który wyznacza podział na kolumny (rys. 1.10). Separatorem w pliku CSV może być przecinek lub inny symbol, np. spacja, średnik, tabulator.

Format CSV

Separator

waluta	kod	liczba jednostek	kurs średni
dolar amerykański	USD	1	3,5482
euro	EUR	1	4,2016
forint (Węgry)	HUF	100	1,342
frank szwajcarski	CHF	1	3,5938
funt szterling	GBP	1	4,7628
hrywna (Ukraina)	UAH	1	0,1288
jen (Japonia)	JPY	100	3,1424
korona czeska	CZK	1	0,1639
peso chilijskie	CLP	100	0,558

kursy_walut.csv

```
waluta→kod→liczba jednostek→kurs średni
dolar amerykański→USD→1→3,5482
euro→EUR→1→4,2016
forint (Węgry)→HUF→100→1,342
frank szwajcarski→CHF→1→3,5938
funt szterling→GBP→1→4,7628
hrywna (Ukraina)→UAH→1→0,1288
jen (Japonia)→JPY→100→3,1424
korona czeska→CZK→1→0,1639
peso chilijskie→CLP→100→0,558
```

Rys. 1.10. Przykładowa tabela z danymi o kursach walut oraz odpowiadający jej plik tekstowy (separatorem danych jest tabulator, oznaczony strzałką)

Pokażemy, jak za pomocą **Kreatora importu tekstu** pobrać do programu Excel tabelę wyników Grand Prix Brazylii 2018 w Formule 1. W tym celu w nowym skoroszycie wybieramy **Dane → Pobieranie danych zewnętrznych → Z tekstu**. Następnie w **Kreatorze importu tekstu** wskazujemy plik z danymi (odpowiedni plik CSV można znaleźć w sieci). W kreatorze w kolejnych krokach (rys. 1.11, s. 20) określamy następujące parametry:

Kreator importu tekstu

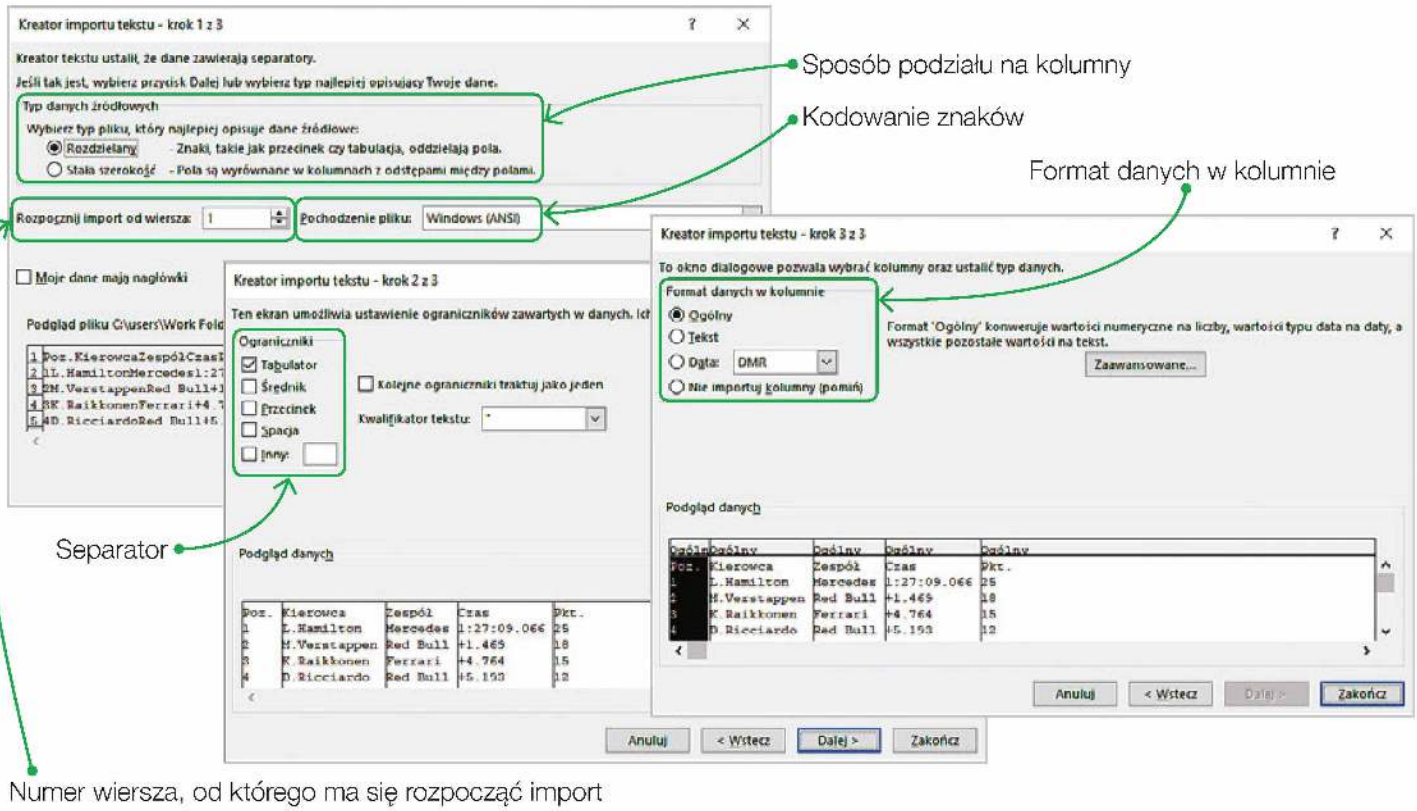
- typ danych źródłowych – **Rozdzielany**, jeżeli w danych są separatory (np. przecinek, średnik, tabulator), lub **Stała szerokość**, gdy dane w kolumnie mają taką samą długość (w naszym pliku CSV dane są rozdzielone tabulatorami, więc wybieramy opcję **Rozdzielany**);
- numer wiersza, od którego rozpocznie się importowanie (w naszym przypadku od pierwszego);
- pochodzenie pliku, czyli informacje o systemie kodowania znaków (pozostawiamy wartość domyślną);
- ograniczniki, czyli rodzaj separatora danych (zaznaczamy opcję **Tabulator**);
- formaty danych dla poszczególnych kolumn, np. **Data**, **Tekst** lub **Ogólny** (jak w naszym przypadku).

Dobra rada

Dane z pliku tekstowego (TXT i CSV) możesz zaimportować również, wybierając **Dane → Pobieranie i przekształcanie → Nowe zapytanie → Z pliku → Z pliku tekstowego**.

Dobra rada

Oprócz ograniczników możesz też ustawić kwalifikator tekstu (" lub '). Tekst widoczny między kwalifikatorami trafi do jednej komórki, nawet jeśli zawiera znak separatora.



Rys. 1.11. Kreator importu danych z pliku tekstowego

Dobra rada

Jeśli w wersji arkusza kalkulacyjnego, której używasz, okna narzędzi wyglądają inaczej niż w podręczniku, poszukaj na nich omawianych opcji lub skorzystaj z pomocy programu.

Po przejściu wszystkich kroków kreatora w kolejnym oknie wybieramy, gdzie mają się znaleźć zaimportowane dane (w istniejącym lub nowym arkuszu) – zaznaczamy **Istniejący arkusz**. Dane z pliku tekstowego zostaną umieszczone w arkuszu Excela w sposób pokazany na podglądzie danych w kroku 3 kreatora (rys. 1.11).

Program automatycznie tworzy łącze do pliku źródłowego. Oznacza to, że aby zaktualizować dane w arkuszu (gdy w pliku źródłowym coś się zmieni), wystarczy odświeżyć połączenie. Można to zrobić, wybierając **Dane → Połączenia → Odśwież wszystko**.

Dobra rada

Kiedy ręcznie odświeżasz dane zaimportowane za pomocą **Kreatora importu tekstu**, Excel domyślnie prosi o wskazanie pliku źródłowego.

Ćwiczenie 6

Zaimportuj do arkusza kalkulacyjnego wyniki Grand Prix Abu Zabi 2018 w Formule 1 z pliku, który otrzymasz od nauczyciela (np. *F1_GP_Abu_Zabi_2018.csv*). Wiersze z trzema najlepszymi czasami oznacz kolorami (złotym, srebrnym, brązowym).
Uwaga: Tylko najlepszy czas jest zapisany wprost, kolejne wartości to opóźnienie względem pierwszego.

Dobra rada

Aby efektywnie importować dane z serwisów internetowych, będziesz potrzebować wersji Excela od 2016 (lub starszej z dodatkiem Power Query).

Importowanie danych ze źródeł internetowych

Program Excel dobrze radzi sobie z danymi, które wczytujemy do arkusza wprost ze strony internetowej. W ten sposób możemy importować i przetwarzać wiele ciekawych informacji. Podobnie jak w przypadku importu z pliku, Excel zapisuje łącze do danych źródłowych.

Takie połączenie można odświeżyć i w ten sposób zaktualizować dane. Oczywiście gdy źródłem jest strona WWW, do synchronizacji niezbędne jest połączenie z internetem.

W serwisie Narodowego Banku Polskiego (https://www.nbp.pl) znajdziemy m.in. tabele z aktualnymi i archiwalnymi średnimi kursami walut. Wybierzemy dla przykładu tabelę zawierającą bieżące kursy średnie walut obcych (*tabela A*) i skopiujemy adres tej strony WWW. W nowym arkuszu programu Excel wybieramy **Dane → Pobieranie i przekształcanie → Nowe zapytanie → Z innych źródeł → Z sieci Web** i wklejamy skopiowany wcześniej adres strony. Po zatwierdzeniu przyciskiem **OK** może się zdarzyć, że zostaniemy poproszeni o potwierdzenie tożsamości, aby uzyskać dostęp do wybranych zasobów – w tym przypadku wystarczy dostęp anonimowy (rys. 1.12).

Dobra rada

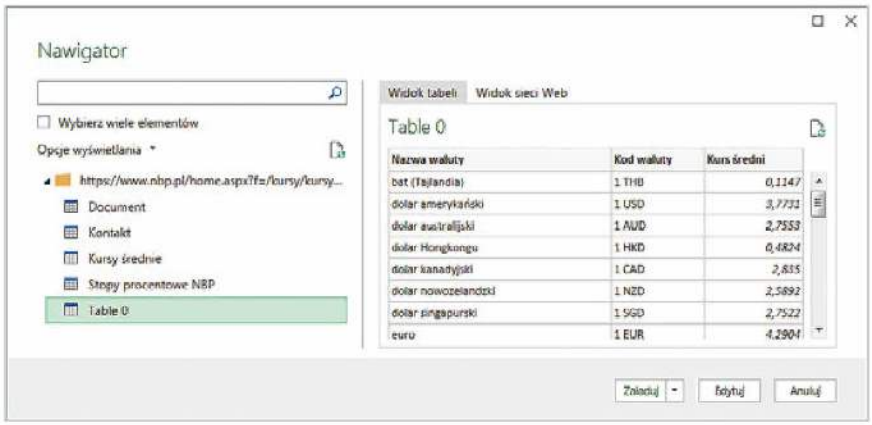
Dane z internetu możesz też zaimportować, wybierając **Dane → Pobieranie danych zewnętrznych → Z sieci Web**. Jednak skorzystanie z narzędzia **Nowe zapytanie** daje większe możliwości.



Rys. 1.12. Uzyskiwanie dostępu do zawartości sieci Web

Kiedy wybierzemy **Połącz**, program pobierze dane znajdujące się pod wskazanym wcześniej adresem WWW. Zanim dane trafią do arkusza programu Excel, musimy je jeszcze zweryfikować. Strona internetowa może bowiem zawierać wiele różnych danych (np. dane kontaktowe). Musimy z nich wybrać tylko te, które nas interesują.

W oknie **Nawigator** z listy po lewej stronie wybieramy **Table 0**. W **Widoku tabeli** (w prawej części okna) zobaczymy zawartość właściwej tabeli. Następnie klikamy przycisk **Załaduj** (rys. 1.13).



Rys. 1.13. Wybór tabeli, którą załadujemy do arkusza

Dobra rada

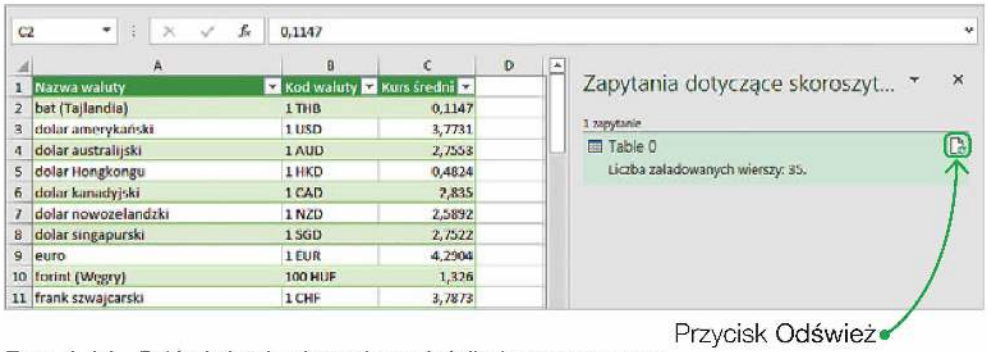
Aby zobaczyć podgląd strony WWW, wybierz **Widok sieci Web** w prawej części okna **Nawigator**.

Tabela programu Excel, s. 32

Tak zaimportowane do arkusza dane są sformatowane w szczególny sposób – jako **tabela**. Aby je zaktualizować, należy w panelu **Zapytania dotyczące skoroszytów**, po prawej stronie naszego zapytania, wybrać **Odśwież** (rys. 1.14).

Dobra rada

Aby wyświetlić **Zapytania dotyczące skoroszytów**, wybierz **Dane → Pobieranie i przekształcanie → Pokaż zapytania**.

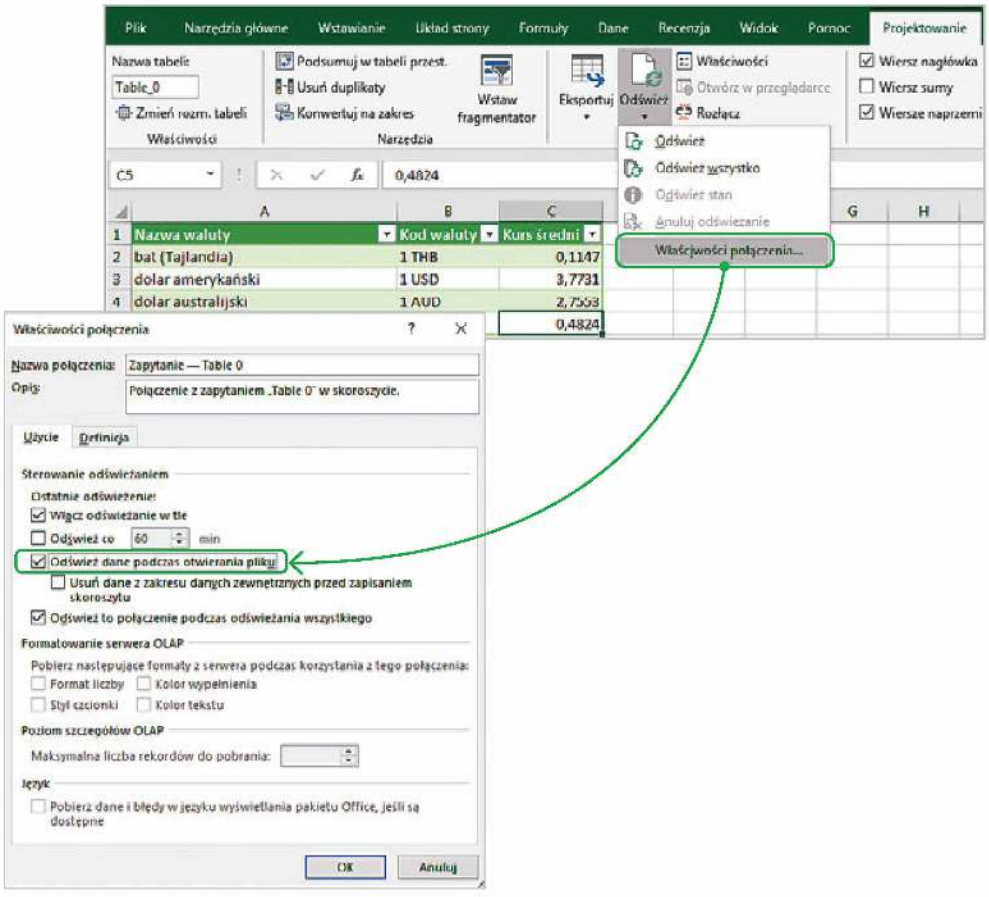


Rys. 1.14. Odświeżanie danych ze źródła internetowego

Dobra rada

Przycisk **Odśwież** znajdziesz również na karcie **Dane**.

Możemy również określić sposób odświeżania danych. W tym celu ustawiamy kursor w dowolnej komórce z zaimportowanymi danymi. Następnie wybieramy **Projektowanie → Dane tabeli zewnętrznej → Odśwież → Właściwości połączenia...** W oknie **Właściwości połączenia** można ustawić np. odświeżanie w momencie, gdy otwieramy skoroszyt, co określoną liczbę minut lub w tle (nawet kiedy pracujemy z arkuszem). My zaznaczymy **Odśwież dane podczas otwierania pliku** (rys. 1.15).



Rys. 1.15. Sposoby odświeżania danych pobranych do arkusza

Ćwiczenie 7

Zaimportuj do arkusza kalkulacyjnego wyniki zawodów w rzucie młotem kobiet i mężczyzn na wybranych przez ciebie igrzyskach olimpijskich. Dane potrzebne do wykonania tego ćwiczenia znajdziesz na stronie https://www.olympic.org/.

1.7. Przetwarzanie importowanych danych

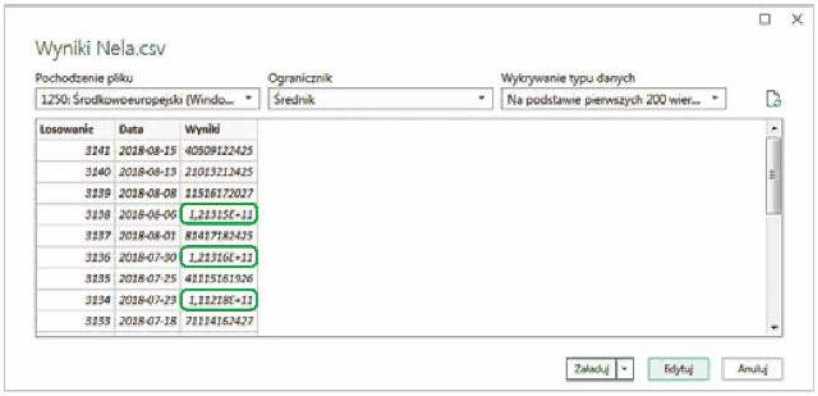
Kiedy pobieramy dane do arkusza, możemy je przekształcić, jeszcze zanim do niego trafią. Wykorzystamy **narzędzie Nowe zapytanie**, aby wczytać do arkusza i poddać dalszej analizie wyniki 100 losowań przykładowej loterii o nazwie Nela: 6 liczb ze zbioru od 1 do 27. Docelowo chcemy określić częstotliwość występowania poszczególnych liczb. Załóżmy, że plik z danymi zawiera rozdzielone średnikami: numer losowania, datę losowania oraz wylosowane liczby (liczby są oddzielone spacjami).

Narzędzie Nowe zapytanie

Wybieramy **Dane → Pobieranie i przekształcanie → Nowe zapytanie → Z pliku → Z pliku CSV** i wskazujemy plik z wynikami losowań. Już na tym etapie widzimy, że istotną dla nas informację o liczbach, które padły w danym losowaniu, program interpretuje nie jako 6 poszczególnych wartości, ale skleja je w jedną liczbę, stosując w niektórych przypadkach notację naukową (liczby w zielonych ramkach na rys. 1.16).

Warto wiedzieć

Notacja naukowa to wygodny sposób zapisu bardzo dużych lub bardzo małych liczb rzeczywistych, np. zapis 2,328E+11 oznacza liczbę $2{,}328 \cdot 10^{11}$.



Rys. 1.16. Pobieranie wyników losowań

A to ciekawe

Gdzie spadają owoce?

Falling Fruit to organizacja non profit, która wyszukuje i gromadzi informacje o lokalizacji roślin jadalnych w miastach na całym świecie. Przetworzone dane są publikowane na stronie http://fallingfruit.org w postaci interaktywnej mapy, dzięki której można odkrywać kulinarne dary natury. Wśród tych roślin są pospolite, m.in. jabłonie i grusze, oraz mniej znane, jak np. dereń jadalny. Na koniec 2019 r. na mapie oznaczonych było ponad 2800 rodzajów roślin w około 1,5 mln miejsc. Organizacja zachęca w ten sposób do kontaktu z naturą i innymi ludźmi.



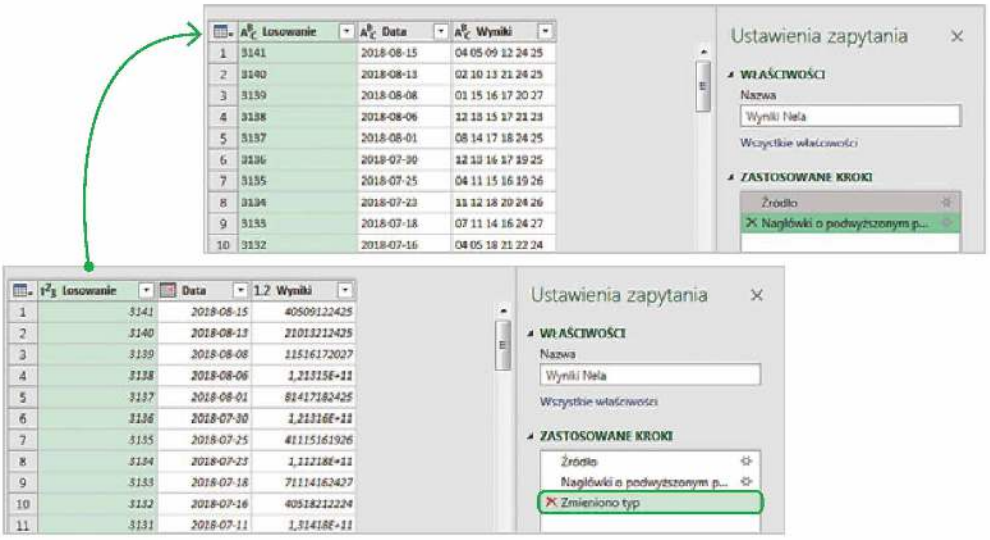
Z tak sformatowanych danych nie skorzystamy, musimy je więc przekształcić. W oknie podglądu wybieramy opcję **Edytuj**, co spowoduje uruchomienie **Edytora Power Query** i załadowanie do niego tabeli.

Edytor Power Query

Poszczególnym kolumnom został automatycznie przypisany wykryty przez program typ danych (liczba całkowita, data oraz liczba dziesiętna). Z prawej strony okna, w **Ustawieniach zapytania** widzimy, jakie kroki wykonał program, aby zaimportowane dane miały bieżącą strukturę (rys. 1.17). Pierwszy wiersz pliku źródłowego program przekształcił na nagłówki kolumn, a typy danych w każdej kolumnie przypisał automatycznie. Z wyjątkiem kolumny z wynikami wszystko się zgadza, spróbujemy zatem zmodyfikować dane tak, abyśmy mogli je potem wykorzystać do dalszej analizy. Ostatni z zastosowanych kroków odpowiada za zmianę typu danych. Usuwamy go, klikając ikonę ✕ po lewej stronie napisu „Zmieniono typ". Wszystkie kolumny będą z powrotem miały przypisany typ **Tekst**.

Dobra rada

Jeśli nie wiesz, jak wymawiać obcojęzyczne nazwy umieszczone w podręczniku, skorzystaj z translatora, który daje możliwość odsłuchania tekstu, np. Tłumacza Google lub Bing Microsoft Translator.



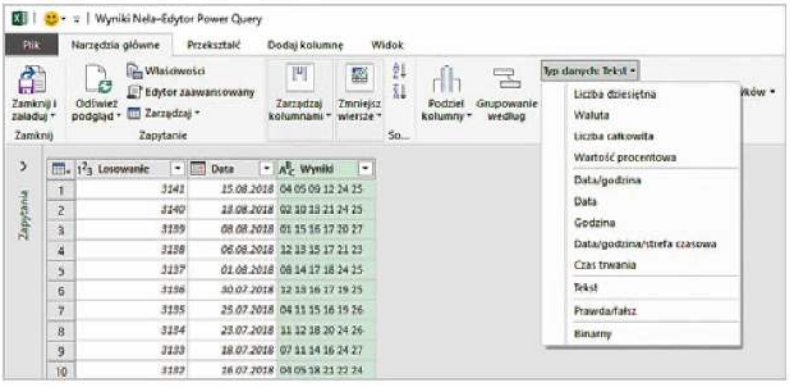
Rys. 1.17. Wycofanie automatycznej zmiany typu danych

Aby ustawić typ danych, zaznaczamy kolumnę i wybieramy **Narzędzia główne → Przekształć → Typ danych**. Dla kolumny Losowanie wybieramy typ **Liczba całkowita**, a dla kolumny Data – typ **Data**. Nie zmieniamy typu danych w kolumnie Wyniki, pozostawiamy je na razie jako **Tekst** (rys. 1.18).

Dobra rada

Typ danych możesz zmienić również, klikając w ikonę po lewej stronie nazwy kolumny. Ikony są różne dla różnych typów danych.



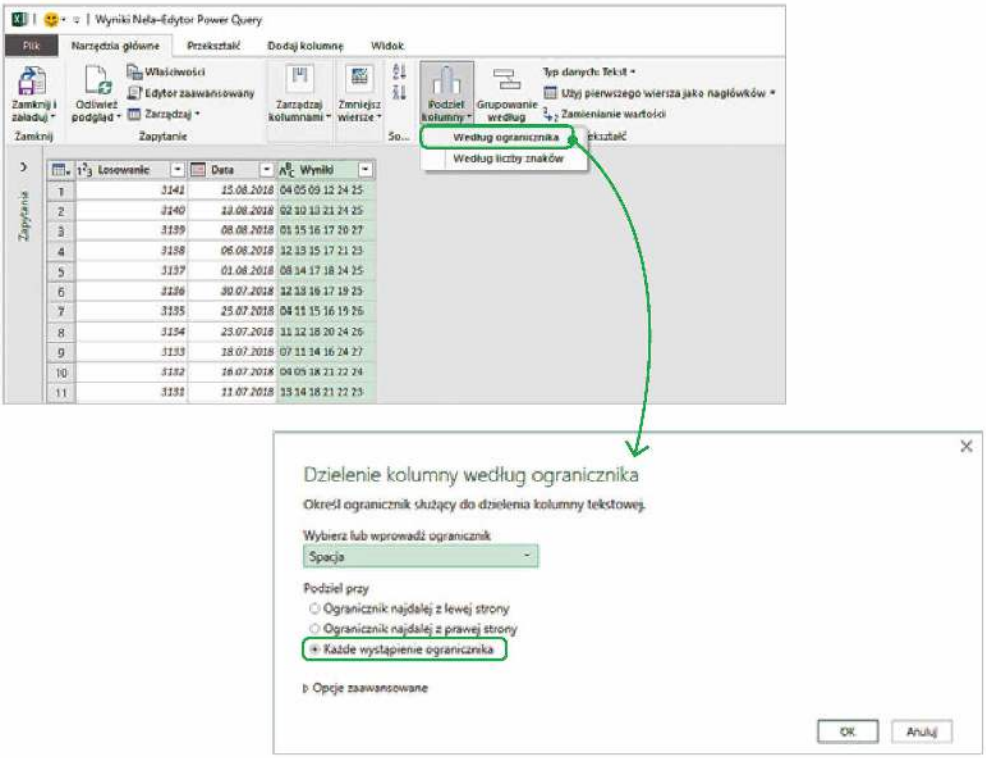


Rys. 1.18. Tabela po ustawieniu odpowiednich typów danych

W kolejnym kroku rozdzielimy kolumnę z wynikami na poszczególne elementy. W tym celu zaznaczamy kolumnę `Wyniki` (lub dowolną komórkę w tej kolumnie), a następnie wybieramy **Narzędzia główne → Przekształć → Podziel kolumny → Według ogranicznika**. W wyświetlonym oknie wskazujemy ogranicznik – **Spację** (o ile nie został wykryty automatycznie) i zaznaczamy opcję **Każde wystąpienie ogranicznika** (rys. 1.19).

Warto wiedzieć

Edytor Power Query umożliwia przekształcanie kolumn tekstowych, m.in. zmianę wielkości liter oraz usuwanie znaków niedrukowalnych.



Rys. 1.19. Rozdzielanie danych z kolumny `Wyniki`

Program nie tylko rozdzielił wyniki losowań, lecz także ponownie zinterpretował typ danych każdej z nowych kolumn. Możemy już załadować dane do arkusza. W tym celu wybieramy **Narzędzia główne → Zamknij → Zamknij i załaduj → Zamknij i załaduj do...** Następnie wybieramy **Istniejący arkusz** i klikamy **Załaduj**.

Warto wiedzieć

Zaawansowane działania Power Query można zaprogramować w języku skryptowym M, który znacząco zwiększa funkcjonalność narzędzia.

Ćwiczenie 8

Plik, który otrzymasz od nauczyciela (np. *zaproszeni_autorzy.csv*), zawiera listę osób (imię i nazwisko) oraz liczbę ich publikacji. Zaimportuj te dane do arkusza, do trzech kolumn: `Imię`, `Nazwisko`, `Liczba publikacji`.

Zapamiętaj

Podczas importowania danych do arkusza kalkulacyjnego zawsze należy zwracać uwagę na sposób podziału danych na kolumny oraz typy danych automatycznie przypisane przez program.

Jak dobrać wykres do rodzaju danych?

Współczesne arkusze kalkulacyjne dają duże możliwości zarówno wyboru typu wykresu, jak i jego modyfikacji, włącznie z nadaniem mu efektu trójwymiarowości. Warto jednak zastanowić się, jaki typ wykresu najlepiej zaprezentuje przygotowane przez nas dane.

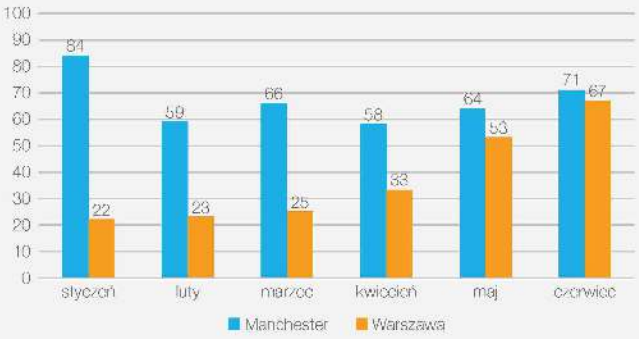
Porównanie wartości nominalnych

Aby zaprezentować dane dotyczące częstości występowania wybranej cechy lub zjawiska, warto skorzystać z **wykresów kolumnowych** i **słupkowych**.

Wykres kolumnowy jest zbudowany z pionowych prostokątów, natomiast **wykres słupkowy** – z prostokątów ułożonych poziomo.

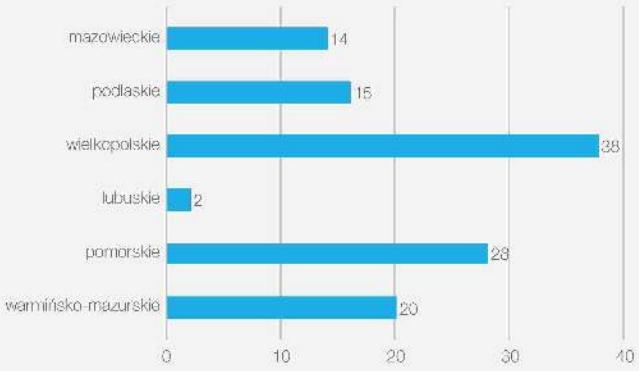
Wykres kolumnowy bardzo dobrze sprawdza się przy porównywaniu co najmniej dwóch serii danych. Serie warto oznaczyć różnymi kolorami.

Średnia opadów w milimetrach w pierwszym półroczu



Wykres słupkowy możemy zastosować wtedy, gdy nazwy etykiet są długie i mogłyby przeszkadzać w odczytaniu danych.

Turniej tenisa stołowego – liczba uczestników



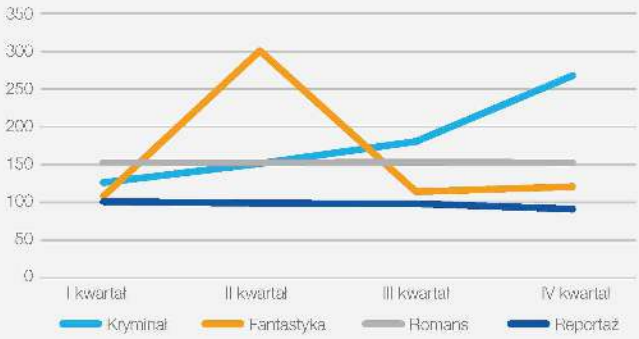
Prezentacja danych zmieniających się w czasie

Wykres liniowy pozwala zaprezentować zmiany pewnych wartości w czasie, a także pokazać trendy i efekty sezonowe.

Wykres panelowy składa się z wielu wykresów o podobnej budowie, które dają się łatwo porównać. Ten typ wykresu jest również przydatny do obserwacji dynamiki zmian.

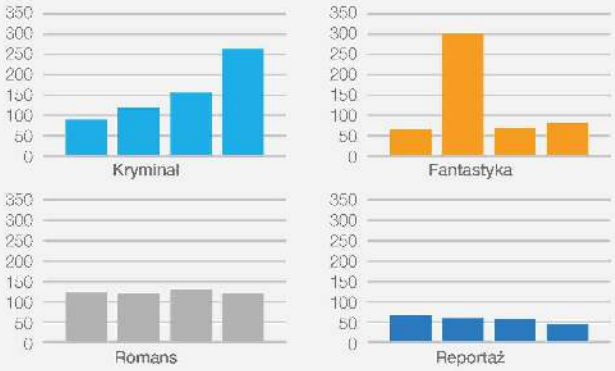
Wykres liniowy dobrze pokazuje dynamikę zmian w czasie. Należy dobrać odpowiednią jednostkę czasu i kolory dla serii danych.

Wypożyczenia książek według kategorii



Wykres panelowy składa się z wielu mniejszych wykresów, które łatwo porównać. Należy zadbać o jednakową skalę na osi pionowej.

Wypożyczenia książek w kolejnych kwartałach





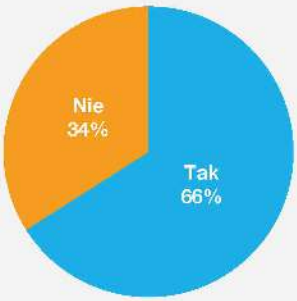
Prezentacja podziału całości na części

Aby przedstawić podział pewnej całości, można zastosować **diagram kołowy**. Długości łuków wycinków koła są proporcjonalne do wartości, których dotyczą. Warto opisywać je dodatkowo wartością procentową lub liczbową, ponieważ porównywanie podobnych wycinków koła jest trudne.

Można też zastosować **wykres słupkowy**, który dobrze pokazuje podział całości na wiele części.

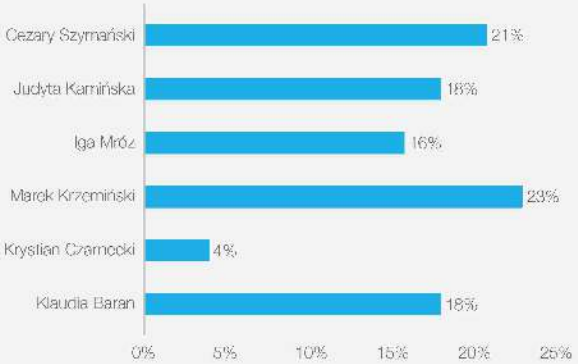
Wykres kołowy dobrze nadaje się do prezentowania małej liczby wartości, w których wyraźnie widać podział całości.

Czy jesteś za utworzeniem szkolnego radiowęzła?



Wykres słupkowy przejrzyście obrazuje podział całości na wiele części, nawet o niewielkim zróżnicowaniu wartości.

Wyniki wyborów do samorządu szkolnego



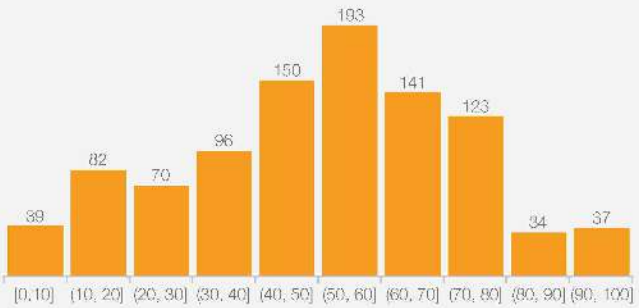
Rozkład cechy w określonych przedziałach

Histogram przedstawia rozkład cechy w ustalonych przedziałach. W każdym z nich wartość oznacza liczbę wystąpień danej cechy w tym przedziale.

Dane tego typu równie dobrze prezentuje się na **wykresie Pareto**, który dodatkowo porządkuje przedziały od tych o największej częstości.

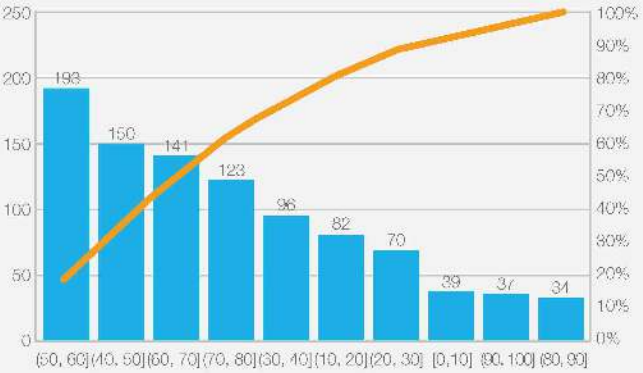
Histogram można zastosować do pokazania rozkładu wyników egzaminu maturalnego. Punktacja od 0 do 100 została podzielona na 10 przedziałów.

Rozkład wyników egzaminu maturalnego



Wykres Pareto też można wykorzystać do pokazania wyników egzaminu maturalnego. Linia pokazuje przyrost udziału procentowego wyników z każdego przedziału punktów.

Rozkład wyników egzaminu maturalnego



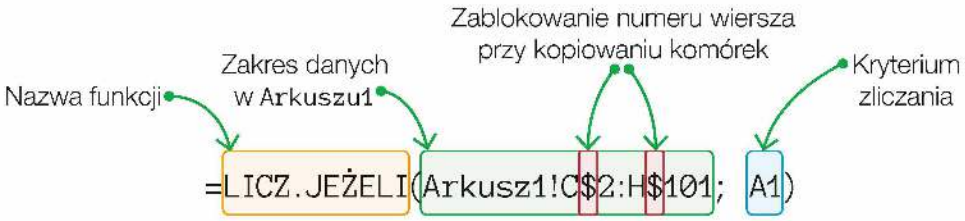
Analiza wyników losowań

Zbadajmy częstotliwość występowania poszczególnych liczb od 1 do 27 w 100 losowaniach loterii Nela. W nowym arkuszu (Arkusz2) wprowadzamy w komórkach A1:A27 liczby od 1 do 27 – posłużą one jako argumenty **funkcji** LICZ.JEŻELI. Funkcja ta oblicza, ile komórek we wskazanym zakresie spełnia dane kryterium. Aby obliczyć, ile razy została wylosowana liczba 1, w komórce B1 Arkusza2 wprowadzamy formułę =LICZ.JEŻELI(Arkusz1!C$2:H$101;A1). Znaczenie poszczególnych elementów tej formuły wyjaśnia rysunek 1.20.

Funkcja LICZ.JEŻELI

Dobra rada

Aby zmienić sposób odwołania do komórki, wystarczy po wpisaniu jej adresu naciskać **F4**. W ten sposób możesz się przełączać między adresowaniem względnym (C2), adresowaniem bezwzględnym (C2), blokowaniem wiersza (C$2) i blokowaniem kolumny ($C2).

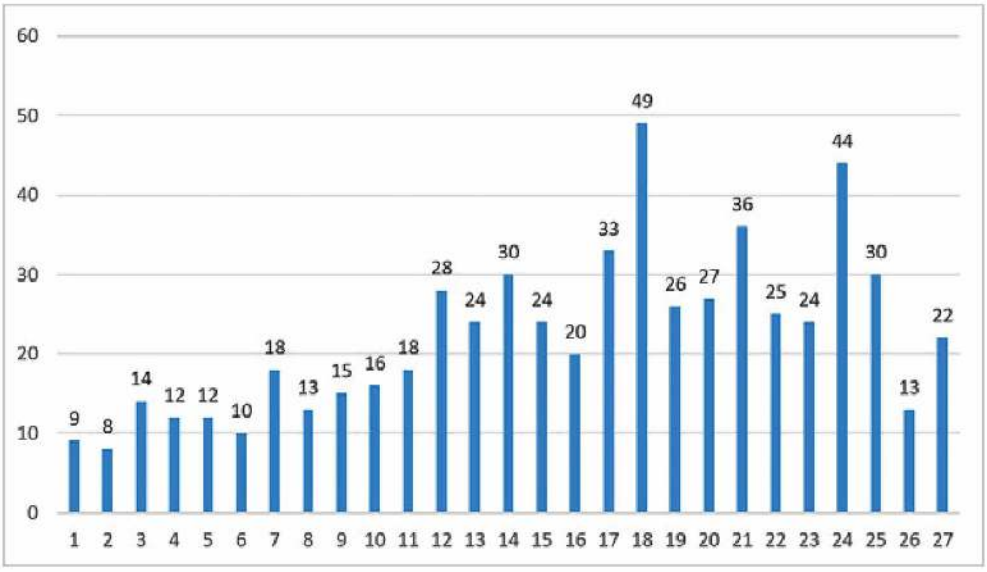


Rys. 1.20. Formuła z użyciem funkcji LICZ.JEŻELI

Dobra rada

Formułę łatwo skopiujesz, korzystając z uchwytu wypełniania. Złap komórkę dokładnie za prawy dolny wierzchołek. Kursor zmieni się wtedy w +.

Analogicznie obliczymy częstotliwości dla pozostałych liczb (w formule zmieni się jedynie kryterium zliczania). W tym celu kopiujemy formułę z komórki B1, przeciągając ją w dół aż do komórki B27. Wyniki obliczeń możemy przedstawić na wykresie. Aby go wstawić, zaznaczamy kolumnę z danymi, wybieramy **Wstawianie → Wykresy** i określamy rodzaj wykresu, który chcemy utworzyć – my wybieramy kolumnowy (rys. 1.21).



Rys. 1.21. Wykres rozkładu częstotliwości występowania liczb od 1 do 27 dla 100 losowań

Ćwiczenie 9

Na podstawie pliku z danymi, który otrzymasz od nauczyciela (np. *zabiegi_pielegnacyjne.csv*), oblicz, ile razy w pewnym salonie urody został wykonany każdy z zabiegów.

Podsumowanie

- Arkusz kalkulacyjny to program, który pozwala przechowywać i przetwarzać duże zbiory danych. Wykorzystuje do tego komórki, które mają swoje adresy, oraz formuły, które pozwalają przetwarzać dane.
- Arkusz kalkulacyjny ma wiele wbudowanych funkcji umożliwiających wykonywanie obliczeń.
- Narzędzia eksportowania i importowania służą do wymiany danych między programami.
- Arkusz kalkulacyjny umożliwia importowanie danych w różnych formatach tekstowych. Jednym z najczęściej stosowanych formatów tekstowych jest CSV.
- W arkuszu kalkulacyjnym można przedstawić dane w formie wykresu. Typ wykresu należy zawsze dobierać do rodzaju danych, które mają zostać zaprezentowane.

Zadania

* **1** Sprawdź działanie funkcji: FRAGMENT.TEKSTU, LEWY, PRAWY, ZŁĄCZ.TEKSTY. Opracuj skoroszyt prezentujący ich zastosowanie.

* **2** Pobierz do arkusza kalkulacyjnego listę 25 najgłębszych jezior świata wraz z ich głębokością.

* **3** Na stronie internetowej Głównego Urzędu Statystycznego znajdź plik CSV dotyczący np. stopy bezrobocia w Polsce lub Produktu Krajowego Brutto (PKB). Zaimportuj te dane do arkusza kalkulacyjnego. Zwróć uwagę na poprawność typów danych.

* **4** Oblicz w arkuszu kalkulacyjnym sumę kolejnych liczb całkowitych od 21 do 235.

* **5** Pobierz z wiarygodnego źródła dane dotyczące składu powietrza na Ziemi. Udział procentowy poszczególnych gazów przedstaw na wykresie lub diagramie.

** **6** Z serwisu https://dane.gov.pl zaimportuj do arkusza kalkulacyjnego wykaz muzeów w Polsce.

** **7** Znajdź stronę internetową zawierającą dane tabelaryczne na temat liczby ludności Polski w ostatnich 10 latach. Zaimportuj te dane do arkusza kalkulacyjnego i przedstaw za pomocą dowolnego wykresu.

** **8** Pobierz z internetu do arkusza kalkulacyjnego wyniki 30 losowań dowolnej loterii liczbowej. Wylicz, ile razy każda liczba występowała w wynikach losowania.

*** **9** Do dwóch arkuszy tego samego skoroszytu załaduj aktualną *tabelę A* średnich kursów walut oraz tabelę archiwalną z tego samego dnia miesiąca, ale z poprzedniego roku. W kolejnym arkuszu pokaż na wykresie słupkowym porównanie tych danych.

*** **10** W pliku, który otrzymasz od nauczyciela (np. *pracownicy.xlsx*), znajduje się kolumna o nazwie Imię i nazwisko. Rozdziel ją na dwie, stosując funkcje: LEWY, PRAWY, DŁ, SZUKAJ.TEKST.

2. Wyciągamy wiedzę z danych

Wyobraź sobie, że jesteś właścicielem dużej sieci sklepów. Z każdego sklepu codziennie zbierane są informacje o każdym sprzedanym produkcie, takie jak: dokładny czas transakcji, cena sprzedaży, parametry towaru (np. rodzaj, kolor, rozmiar), numer sprzedawcy. Rocznie to miliony transakcji. Jeśli dane gromadzone są przez lata, można na ich podstawie wyciągać wiele ciekawych wniosków, opracowywać strategie sprzedaży, zamówień towarów, obsady pracowników. Aby jednak zapanować nad taką ilością danych, potrzebne są narzędzia do ich przetwarzania, analizy, wnioskowania, prognozowania, obliczania i prezentowania wyników. W tym temacie pokażemy, jak wykorzystać w tym celu arkusz kalkulacyjny.

Cele lekcji

- Przypomnisz sobie, jak sortować i filtrować dane w arkuszu kalkulacyjnym.
- Nauczysz się generować losowe dane w programie Excel.
- Poznasz tabele Excela i ich zastosowanie.
- Dowiesz się, czym są tabele przestawne.
- Nauczysz się budować, modyfikować i formatować tabele przestawne.
- Poznasz analityczne możliwości tabel przestawnych.
- Nauczysz się prezentować informacje za pomocą wykresów przestawnych.

2.1. Dane w arkuszu kalkulacyjnym

Arkusz kalkulacyjny jest doskonałym narzędziem do złożonej analizy danych. Poprawnie przeprowadzona analiza powinna dostarczyć odpowiedzi na pytania, które sobie stawiamy. Postawienie właściwych pytań i znalezienie na nie odpowiedzi pozwala odkryć zależności i wzorce ukryte w danych – w tym sensie umożliwia wyciąganie z nich wiedzy.

Sortowanie i filtrowanie danych

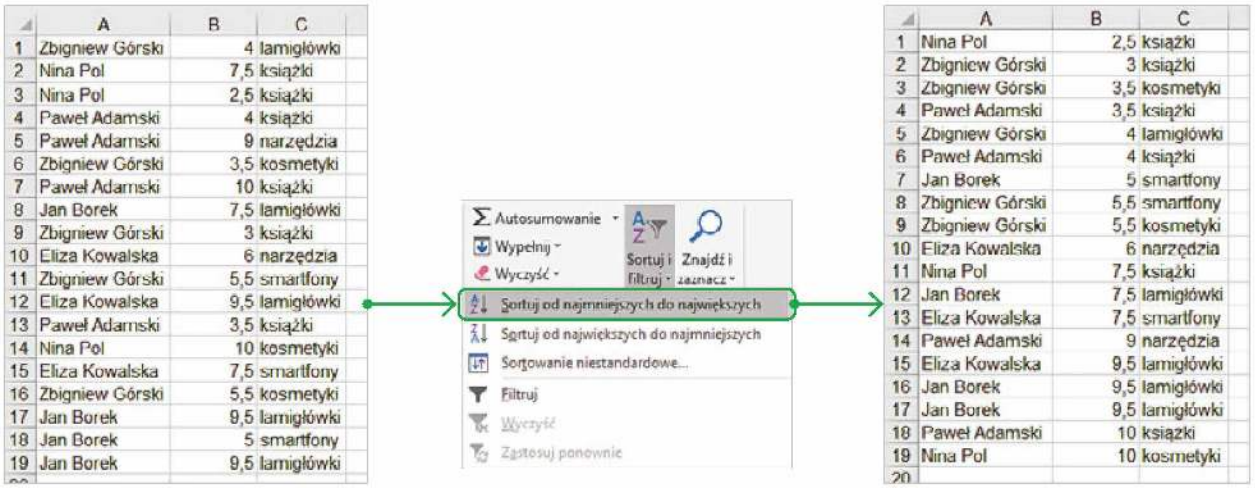
Podczas analizowania danych często stosuje się dwie operacje:

Sortowanie ▸ **sortowanie** – umożliwia zmianę kolejności elementów w zbiorze, np. wyniki biegu na 1000 m możemy uporządkować od najlepszych do najsłabszych;

Filtrowanie ▸ **filtrowanie** – pozwala zawęzić listę obiektów do tych, które spełniają określone kryteria, np. ze zbioru samochodów wyodrębnić kabriolety wyprodukowane przed 1945 r.

Zbiory danych w Excelu można w prosty sposób sortować i filtrować dzięki opcji **Sortuj i filtruj**, znajdującej się na karcie **Narzędzia główne** w grupie **Edytowanie**. Ponieważ z sortowania i filtrowania będziemy korzystać w całym temacie, omówimy je na początku.

Aby posortować dane w kolumnie, należy najpierw zaznaczyć dowolną komórkę w tej kolumnie. Następnie wybieramy, w jakiej kolejności chcemy sortować – rosnąco czy malejąco. Przykład sortowania pokazano na rysunku 2.1 – listę osób, które oceniały produkty z różnych kategorii, uporządkowano według wystawionej przez nie oceny od najmniejszej wartości do największej (kolumna B). Sortowanie według jednej kolumny układa odpowiednio całe wiersze.

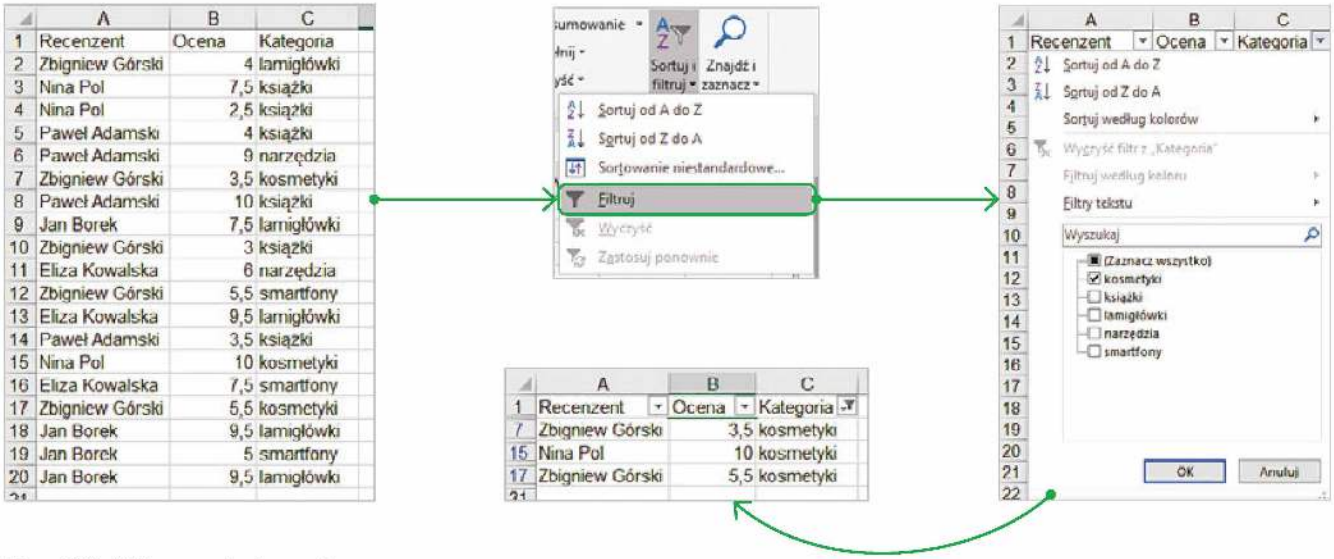


	A	B	C
1	Zbigniew Górski	4	lamigłówki
2	Nina Pol	7,5	książki
3	Nina Pol	2,5	książki
4	Paweł Adamski	4	książki
5	Paweł Adamski	9	narzędzia
6	Zbigniew Górski	3,5	kosmetyki
7	Paweł Adamski	10	książki
8	Jan Borek	7,5	lamigłówki
9	Zbigniew Górski	3	książki
10	Eliza Kowalska	6	narzędzia
11	Zbigniew Górski	5,5	smartfony
12	Eliza Kowalska	9,5	lamigłówki
13	Paweł Adamski	3,5	książki
14	Nina Pol	10	kosmetyki
15	Eliza Kowalska	7,5	smartfony
16	Zbigniew Górski	5,5	kosmetyki
17	Jan Borek	9,5	lamigłówki
18	Jan Borek	5	smartfony
19	Jan Borek	9,5	lamigłówki

	A	B	C
1	Nina Pol	2,5	książki
2	Zbigniew Górski	3	książki
3	Zbigniew Górski	3,5	kosmetyki
4	Paweł Adamski	3,5	książki
5	Zbigniew Górski	4	lamigłówki
6	Paweł Adamski	4	książki
7	Jan Borek	5	smartfony
8	Zbigniew Górski	5,5	smartfony
9	Zbigniew Górski	5,5	kosmetyki
10	Eliza Kowalska	6	narzędzia
11	Nina Pol	7,5	książki
12	Jan Borek	7,5	lamigłówki
13	Eliza Kowalska	7,5	smartfony
14	Paweł Adamski	9	narzędzia
15	Eliza Kowalska	9,5	lamigłówki
16	Jan Borek	9,5	lamigłówki
17	Jan Borek	9,5	lamigłówki
18	Paweł Adamski	10	książki
19	Nina Pol	10	kosmetyki
20			

Rys. 2.1. Sortowanie danych

Jeżeli kolumna z danymi zawiera elementy, które się powtarzają, można je bardzo łatwo wydzielić przy użyciu filtrowania. Pokazano to na rysunku 2.2. Zauważ, że każda kolumna z danymi ma nazwę. W przykładzie ukryto wszystkie elementy kolumny `Kategoria` z wyjątkiem kosmetyków.



	A	B	C
1	Recenzent	Ocena	Kategoria
2	Zbigniew Górski	4	lamigłówki
3	Nina Pol	7,5	książki
4	Nina Pol	2,5	książki
5	Paweł Adamski	4	książki
6	Paweł Adamski	9	narzędzia
7	Zbigniew Górski	3,5	kosmetyki
8	Paweł Adamski	10	książki
9	Jan Borek	7,5	lamigłówki
10	Zbigniew Górski	3	książki
11	Eliza Kowalska	6	narzędzia
12	Zbigniew Górski	5,5	smartfony
13	Eliza Kowalska	9,5	lamigłówki
14	Paweł Adamski	3,5	książki
15	Nina Pol	10	kosmetyki
16	Eliza Kowalska	7,5	smartfony
17	Zbigniew Górski	5,5	kosmetyki
18	Jan Borek	9,5	lamigłówki
19	Jan Borek	5	smartfony
20	Jan Borek	9,5	lamigłówki

	A	B	C
1	Recenzent	Ocena	Kategoria
7	Zbigniew Górski	3,5	kosmetyki
15	Nina Pol	10	kosmetyki
17	Zbigniew Górski	5,5	kosmetyki

Rys. 2.2. Filtrowanie danych

Ćwiczenie 1

W pliku, który otrzymasz od nauczyciela (np. *recenzje.xlsx*), wyszukaj recenzenta, który wystawił najwyższą ocenę w kategorii „kosmetyki".

Tabela programu Excel

Tabela programu Excel

Do danych w arkuszu Excela możemy odwoływać się poprzez adresy komórek lub zakresy komórek. Powiązane ze sobą dane możemy również organizować w postaci **tabel**. Daje to możliwość nie tylko wizualnego grupowania danych, lecz także łatwiejszego ich analizowania, sortowania, filtrowania i formatowania.

Dobra rada

Strzałki na klawiaturze pozwalają przełączać się między komórkami arkusza. Aby zaznaczyć komórki położone obok siebie, użyj strzałki, przytrzymując jednocześnie klawisz **Ctrl**. Aby zaznaczyć cały wiersz tabeli, ustaw kursor na początku wiersza i naciśnij jednocześnie **Ctrl**, **Shift** oraz strzałkę w prawo.

Aby dane pokazane na rysunku 2.2 ze s. 31 przekształcić w tabelę, zaznaczamy odpowiedni zakres komórek (łącznie z nagłówkami) i wybieramy **Wstawianie → Tabele → Tabela**. W wyświetlonym oknie musi być zaznaczona opcja **Moja tabela ma nagłówki**. Nazwę tabeli możemy zmienić, wybierając **Projektowanie → Właściwości → Nazwa tabeli** (rys. 2.3). Jeśli zastosujemy tę nazwę w formułach, będzie ona oznaczać zakres komórek tabeli bez nagłówków kolumn, w naszym przypadku – `A2:C20`.



	A	B	C
1	Recenzent	Ocena	Kategoria
2	Zbigniew Górski	4	lamigłówki
3	Nina Pol	7,5	książki
4	Nina Pol	2,5	książki
5	Paweł Adamski	4	książki
6	Paweł Adamski	9	narzędzia
7	Zbigniew Górski	3,5	kosmetyki
8	Paweł Adamski	10	książki
9	Jan Borck	7,5	lamigłówki
10	Zbigniew Górski	3	książki
11	Eliza Kowalska	6	narzędzia
12	Zbigniew Górski	5,5	smartfony
13	Eliza Kowalska	9,5	lamigłówki
14	Paweł Adamski	3,5	książki
15	Nina Pol	10	kosmetyki
16	Eliza Kowalska	7,5	smartfony
17	Zbigniew Górski	5,5	kosmetyki
18	Jan Borek	9,5	lamigłówki
19	Jan Borck	5	smartfony
20	Jan Borek	9,5	lamigłówki

Rys. 2.3. Tabela programu Excel

Warto wiedzieć

Tabele programu Excel muszą mieć nagłówki. Jeżeli zakres komórek, które przekształcamy w tabelę, nie ma nagłówków, program doda je automatycznie (`Kolumna1`, `Kolumna2` itd.).

Bardzo ważną cechą tabel w programie Excel jest to, że ich rozmiar automatycznie dostosowuje się do zmian. Jeżeli dodamy lub usuniemy wiersz albo kolumnę, nazwa tabeli będzie wskazywała właściwy zakres komórek. Tabel będziemy używać w dalszej części tematu.

Ćwiczenie 2

Utwórz tabelę taką jak na rysunku 2.3. Następnie:

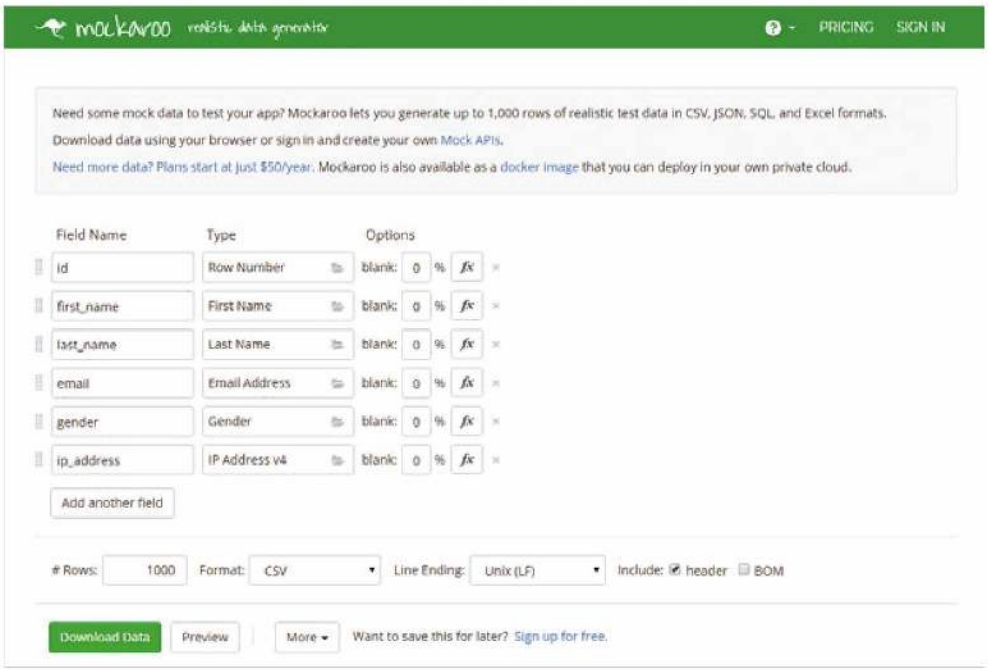
a. dodaj na końcu tabeli nowego recenzenta, uzupełnij ocenę i kategorię,
b. zmień nazwę tabeli na `Oceny_recenzentów`,
c. posortuj dane w tabeli malejąco według oceny.

Generowanie zbiorów danych za pomocą narzędzi online

Dobra rada

Generatory online danych losowych, z których możesz skorzystać, to np.:
www.mockaroo.com
www.generatedata.com
www.random.org/integers

Aby analizować dane, potrzebujemy jakiegoś ich zbioru. Dane do analiz często są dostarczane w postaci plików z różnych systemów informatycznych, mogą być również pobierane wprost z baz danych. Na potrzeby ćwiczeń, które będziemy wykonywać, zbiór danych możemy wygenerować samodzielnie, korzystając z funkcji programu Excel lub dostępnych generatorów online. Jeden z takich generatorów znajdziemy na stronie https://www.mockaroo.com/ (rys. 2.4).



Rys. 2.4. Przykład generatora danych online

Ćwiczenie 3

Za pomocą generatora online (np. mockaroo) wygeneruj zbiór danych zawierający:

a. 30 haseł zabezpieczających konto użytkownika,

b. listę 100 miast i krajów, w których leżą.

Generowanie zbiorów danych za pomocą formuł arkusza kalkulacyjnego

Zbiór danych możemy wygenerować bezpośrednio w programie Excel za pomocą odpowiednich formuł. Przyjmijmy, że chcemy wygenerować 50 losowych dat z zakresu od 1.01.2017 do 31.12.2018, a następnie posortować je od najwcześniejszej do najpóźniejszej. Kiedy wpiszemy takie daty do dwóch komórek arkusza, program automatycznie przypisze im format **Data**. Zmienimy format komórek na **Ogólny**, co spowoduje zamianę dat na liczby całkowite. W naszym przypadku będą to liczby 42736 (co odpowiada dacie 1.01.2017) oraz 43465 (dla daty 31.12.2018). Liczby te potraktujemy jako zakres do generowania liczb losowych, które następnie zamienimy z powrotem na daty.

Aby wylosować liczbę całkowitą z wybranego zakresu, użyjemy **funkcji** LOS.ZAKR, która ma dwa argumenty. Pierwszy z nich (dół) oznacza najmniejszą liczbę, jaka może zostać wylosowana, a drugi (góra) największą liczbę. W tym celu wybieramy komórkę C1, w której ma się znaleźć pierwsza wylosowana data, wpisujemy w niej formułę =LOS.ZAKR(42736;43465) i potwierdzamy klawiszem **Enter**. Następnie kopiujemy tę komórkę, przeciągając w dół do zakresu C1:C50. Aby zamienić liczby z powrotem na daty, zmieniamy format komórek.

Warto wiedzieć

Excel przechowuje daty jako liczby naturalne, które oznaczają kolejne dni numerowane od pewnej ustalonej daty – zwykle liczba 1 oznacza 1 stycznia 1900 r. Na przykład 27 września 2020 r. to dla Excela liczba 44 101.

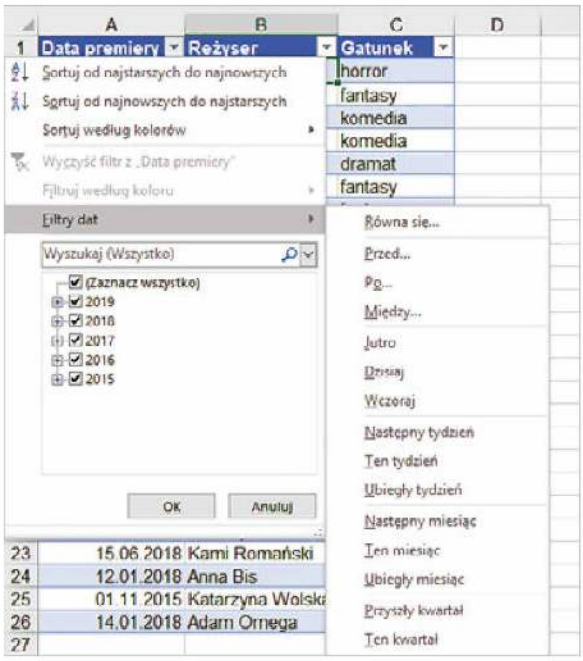
Dobra rada

Aby ustawić typ danych w komórce, kliknij w niej prawym przyciskiem myszy i wybierz opcję **Formatuj komórki...** W zakładce **Liczby** możesz wybrać odpowiednią kategorię.

Funkcja LOS.ZAKR

Sortowanie, s. 30

Aby **posortować** daty, musimy wykonać kilka czynności. Najpierw pozbędziemy się formuł z komórek zawierających daty. Gdybyśmy tego nie zrobili, każda próba sortowania skończyłaby się ponownym obliczeniem formuł, a więc wygenerowaniem na nowo losowych dat. Zaznaczamy wszystkie komórki z datami i kopiujemy je, a następnie wybieramy **Narzędzia główne → Schowek → Wklej → Wartości i formatowanie źródłowe (M)**. Teraz możemy już posortować daty, korzystając ze sposobu opisanego na s. 31. W przypadku filtrowania dat dostępnych jest znacznie więcej możliwości niż dla innych typów danych (rys. 2.5).



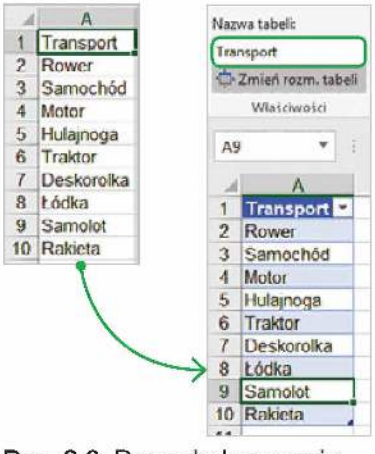
Rys. 2.5. Filtrowanie dat

Ćwiczenie 4

Korzystając z arkusza kalkulacyjnego, wygeneruj 10 adresów IPv4. Każdy adres ma się składać z 4 liczb od 0 do 255 rozdzielonych kropkami, np. 158.69.31.17. Wykorzystaj funkcję ZŁĄCZ.TEKSTY.

Generowanie losowej listy spośród elementów tabeli

Pokażemy teraz, jak wygenerować losową listę produktów na podstawie utworzonej wcześniej tabeli. Tworzymy listę, która posłuży jako źródło danych losowych, np. listę środków transportu, i zamieniamy ją w tabelę. Następnie klikamy w dowolnej komórce tabeli i na karcie **Projektowanie** zmieniamy nazwę tabeli na Transport (rys. 2.6).



Rys. 2.6. Dane do losowania

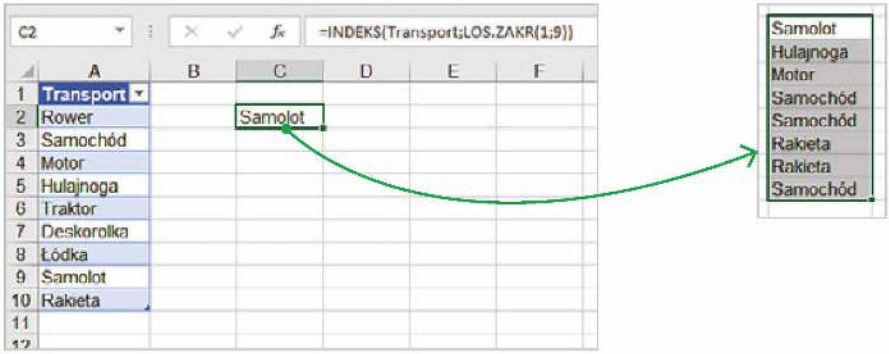
Dobra rada

Aby przekształcić zakres komórek w tabelę programu Excel, możesz zastosować skrót **Ctrl + T**.

Skorzystamy teraz z funkcji INDEKS oraz – znanej już z poprzedniego przykładu – funkcji LOS.ZAKR. **Funkcja INDEKS** zastosowana do tabeli zwróci wartość elementu, który znajduje się w określonym wierszu (i/lub kolumnie). Jeżeli wylosujemy wiersz z tabeli zawierającej dane i połączymy to z funkcją INDEKS, otrzymamy losowy element z tabeli. Do komórki C2, w której ma się znaleźć wylosowany element, wpisujemy następującą formułę: =INDEKS(Transport;LOS.ZAKR(1;9)). Nazwa Transport wskazuje na tabelę bez nagłówków kolumn, co w naszym przypadku oznacza zakres komórek A2:A10. Wiersze w tym zakresie numerowane są od 1 do 9, stąd takie argumenty funkcji LOS.ZAKR.

Funkcja INDEKS

Przeciągając w dół, możemy teraz skopiować formułę z komórki C2 i stworzyć w ten sposób losową listę elementów z tabeli Transport (rys. 2.7).



Rys. 2.7. Generowanie losowych danych na podstawie tabeli

Warto wiedzieć

Formuły zwracają wynik działania do komórki, w której zostały zapisane. To znaczy, że np. formuła z komórki A1 nie może zmienić zawartości komórki B3. Wprowadzanie formuły zawsze rozpoczynamy znakiem = (równości).

Ćwiczenie 5

Utwórz tabele o nazwach: Kolor, Marka samochodu, Rocznik, Moc silnika. Wpisz po kilka wartości do każdej tabeli. Następnie na podstawie danych ze wszystkich tabel wygeneruj 20 wierszy losowych danych – skorzystaj z opisanego powyżej sposobu.

Zapamiętaj

Tabela programu Excel to specjalny zakres komórek, do którego możemy się odwołać, podając jego nazwę. Ponadto tabela dynamicznie zmienia rozmiar wraz ze zmianą liczby kolumn lub wierszy z danymi. Dzięki temu doskonale sprawdza się jako źródło danych (również tych generowanych za pomocą formuł arkusza).

2.2. Tabele przestawne w programie Excel

Arkusz kalkulacyjny oferuje wydajne narzędzie do analizowania, przeglądania i prezentowania danych – **tabele przestawne**. Pozwala ono w prosty sposób podsumowywać (m.in. zliczać i sumować) filtrowane i grupowane dane. Przykładową tabelę przestawną pokazuje rysunek 2.8 na s. 36.

Tabela przestawna

	A	B	C
1	Miasto	Szczecin	
2	Kolor	Szary	
3			
4	**Etykiety wierszy**	**Suma z Cena [zł]**	
5	Fotel	705	
6	Kubek	38	
7	Parasol	45	
8	Plecak	69	
9	Poduszka	40	
10	Ręcznik	32	
11	Rękawiczki	28	
12	Termos	33	
13	Zegarek	398	
14	**Suma końcowa**	**1388**	
15			
16			

Rys. 2.8. Wartość produktów, które są szare, sprzedanych w Szczecinie

Tworzenie tabeli przestawnej

Warto wiedzieć

LibreOffice Calc również umożliwia tworzenie tabel przestawnych. W starszych wersjach programu ta funkcjonalność nazywa się Pilot danych.

Teraz pokażemy, jak utworzyć tabelę przestawną w programie Excel. Skorzystamy z danych o sprzedaży różnych produktów na przestrzeni dwóch lat w pewnej sieci sklepów (rys. 2.9). Informacje pogrupowano w następujące kolumny: `Data sprzedaży`, `Produkt`, `Cena [zł]`, `Kolor`, `Symbol sklepu`, `Miasto`, `Nr sprzedawcy`. W dalszej części tematu będziemy korzystać z takich danych sprzedażowych, aby pokazać analityczne możliwości tabel przestawnych.

	A	B	C	D	E	F	G
1	Data sprzedaży	Produkt	Cena [zł]	Kolor	Symbol sklepu	Miasto	Nr sprzedawcy
2	2017-01-01	Termos	33 zł	Czarny	SK-6	Toruń	P-06
3	2017-01-01	Fotel	235 zł	Niebieski	SK-4	Olsztyn	P-04
4	2017-01-02	Ręcznik	32 zł	Zielony	SK-6	Lublin	P13
5	2017-01-02	Kubek	19 zł	Szary	SK-7	Zakopane	P-06
6	2017-01-02	Zegarek	199 zł	Żółty	SK-4	Rzeszów	P-06
7	2017-01-03	Rękawiczki	28 zł	Różowy	SK-3	Poznać	P-06
8	2017-01-03	Ręcznik	32 zł	Czarny	SK-9	Warszawa	P-06
9	2017-01-04	Rękawiczki	28 zł	Czerwony	SK-9	Olsztyn	P-07
10	2017-01-04	Rękawiczki	28 zł	Czarny	SK-4	Toruń	P-14
11	2017-01-04	Termos	33 zł	Fioletowy	SK-5	Opole	P-04
12	2017-01-04	Portfel	58 zł	Szary	SK-4	Lublin	P-02
13	2017-01-05	Ręcznik	32 zł	Szary	SK-1	Kraków	P-12

Rys. 2.9. Dane sprzedażowe sieci sklepów

Zwróć uwagę na to, że każda kolumna z danymi ma nazwę. Jest to bardzo ważne, gdyż w tabeli przestawnej operuje się właśnie tymi nazwami.

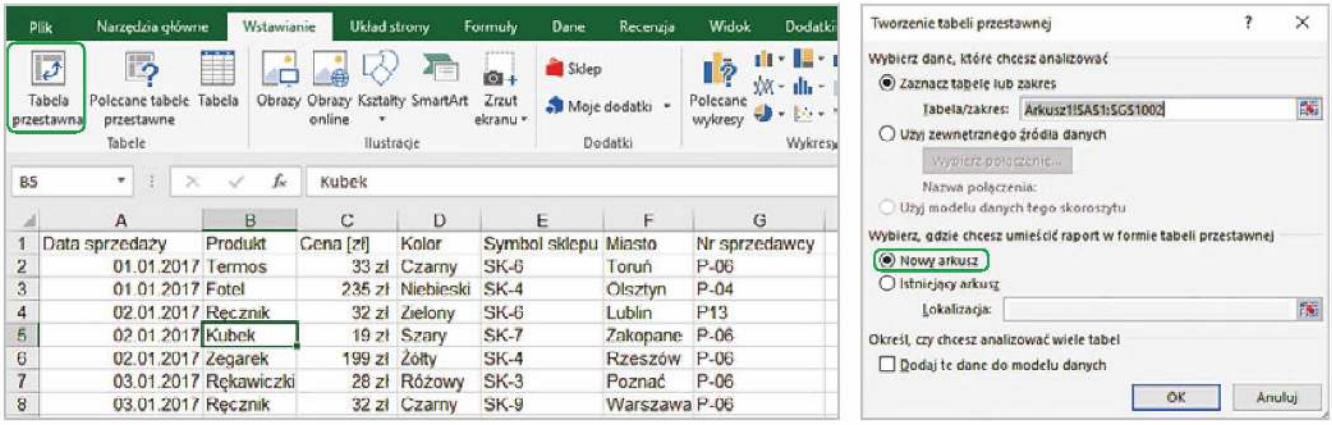
Zapamiętaj

Zakres danych wykorzystywany w tabeli przestawnej musi zawierać wiersz nagłówka z nazwami kolumn.

Aby utworzyć tabelę przestawną, zaznaczamy dowolną komórkę z zakresu danych i wybieramy **Wstawianie → Tabele → Tabela przestawna**. W oknie **Tworzenie tabeli przestawnej** możemy zdecydować o źródle danych – może to być utworzona wcześniej tabela w arkuszu, zakres danych (jak w tym przypadku) lub źródło zewnętrzne (np. baza danych). Wybieramy tu również miejsce, gdzie ma zostać wstawiona tabela przestawna – domyślnie w nowym arkuszu (rys. 2.10).

Dobra rada

Jeśli przy wstawianiu tabeli przestawnej umieścisz kursor w komórce z danymi, dane zaznaczą się automatycznie. Zawsze warto jednak sprawdzić, czy zaznaczony został właściwy zakres.



	A	B	C	D	E	F	G
1	Data sprzedaży	Produkt	Cena [zł]	Kolor	Symbol sklepu	Miasto	Nr sprzedawcy
2	01.01.2017	Termos	33 zł	Czarny	SK-6	Toruń	P-06
3	01.01.2017	Fotel	235 zł	Niebieski	SK-4	Olsztyn	P-04
4	02.01.2017	Ręcznik	32 zł	Zielony	SK-6	Lublin	P13
5	02.01.2017	Kubek	19 zł	Szary	SK-7	Zakopane	P-06
6	02.01.2017	Zegarek	199 zł	Żółty	SK-4	Rzeszów	P-06
7	03.01.2017	Rękawiczki	28 zł	Różowy	SK-3	Poznać	P-06
8	03.01.2017	Ręcznik	32 zł	Czarny	SK-9	Warszawa	P-06

Rys. 2.10. Tworzenie tabeli przestawnej

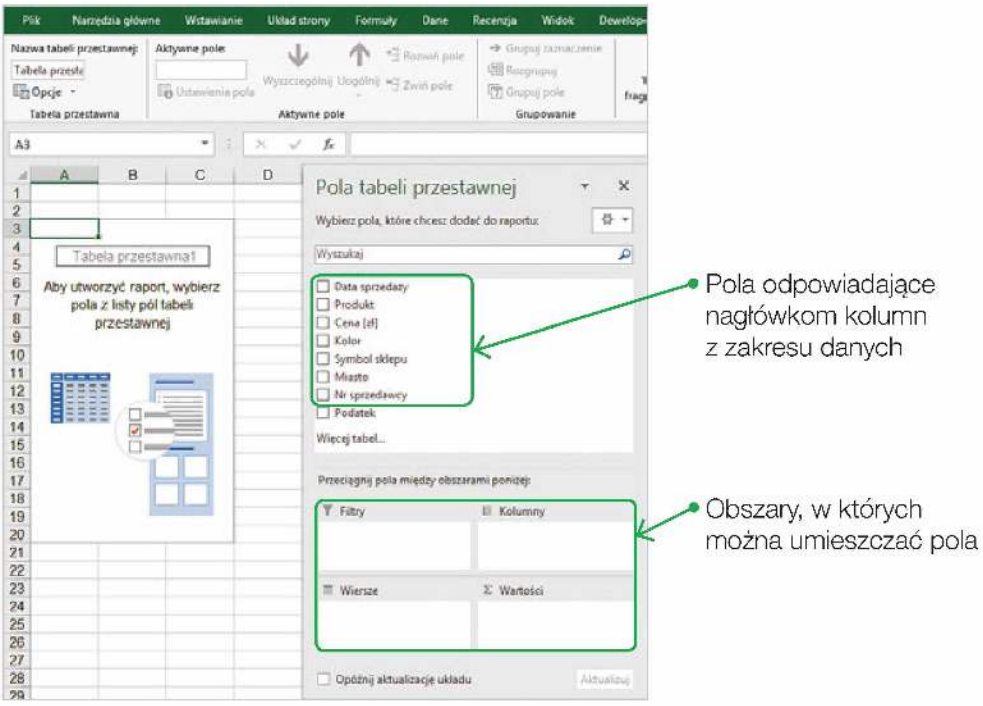
Tabela przestawna, która pojawiła się w nowym arkuszu, jest pusta (rys. 2.11). Z prawej strony ekranu znajduje się panel o nazwie **Pola tabeli przestawnej**, zawierający pola o nazwach takich jak nagłówki kolumn z zakresu z danymi źródłowymi (na podstawie których utworzyliśmy tabelę przestawną). Widać tu również cztery obszary: **Filtry**, **Kolumny**, **Wiersze**, **Wartości**.

Dobra rada

Aby utworzyć tabelę przestawną na podstawie danych z tabeli Excela, zaznacz jej dowolną komórkę i wybierz **Projektowanie → Narzędzia → Podsumuj w tabeli przest.**

Warto wiedzieć

Aby wstawić tabelę przestawną w programie LibreOffice Calc, należy wybrać **Dane → Tabela przestawna → Utwórz.**



Rys. 2.11. Pusta tabela przestawna i elementy do jej budowania

Podstawowa praca z tabelą przestawną sprowadza się do wybierania pól i umieszczania ich w odpowiednich obszarach. Aby umieścić pole w obszarze, należy je tam przeciągnąć i upuścić. To, które pola wybierzemy i w których obszarach je umieścimy, zależy od tego, jakie informacje chcemy uzyskać i w jaki sposób chcemy zaprezentować je w tabeli.

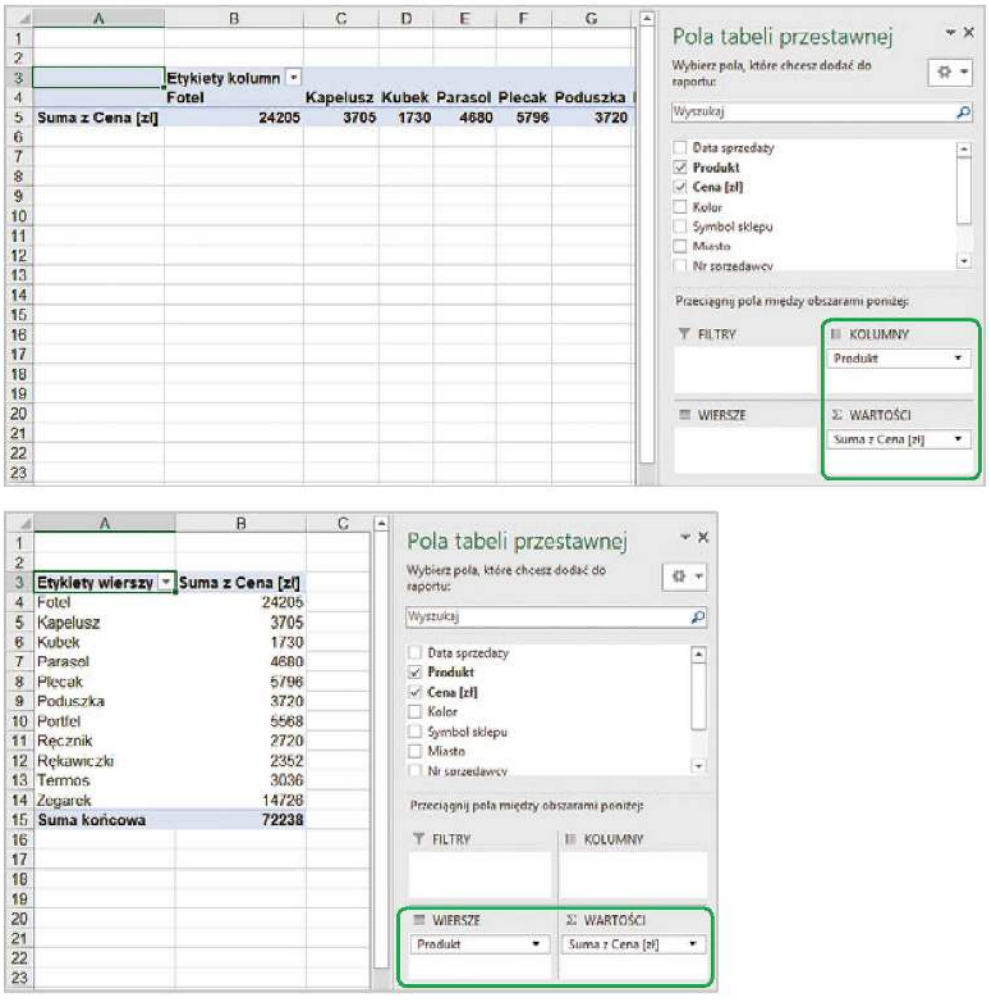
Umieszczenie pola w obszarze wierszy lub kolumn powoduje, że do tabeli przestawnej wstawiane są wartości z odpowiedniej kolumny danych źródłowych (rys. 2.12). **Obszary tabeli przestawnej** pełnią następujące funkcje:

Obszary tabeli przestawnej

- obszar **Filtry** – umożliwia filtrowanie analizowanego zbioru według danych z pól umieszczonych w tym obszarze;
- obszar **Kolumny** – elementy pola z tego obszaru będą rozmieszczone poziomo w kolejnych kolumnach;
- obszar **Wiersze** – wartości z pól, które umieścimy w tym obszarze, będą układane w tabeli przestawnej pionowo, w kolejnych wierszach arkusza;
- obszar **Wartości** – na elementach pola umieszczonego w tym obszarze możemy wykonywać obliczenia, np. zliczać je lub sumować.

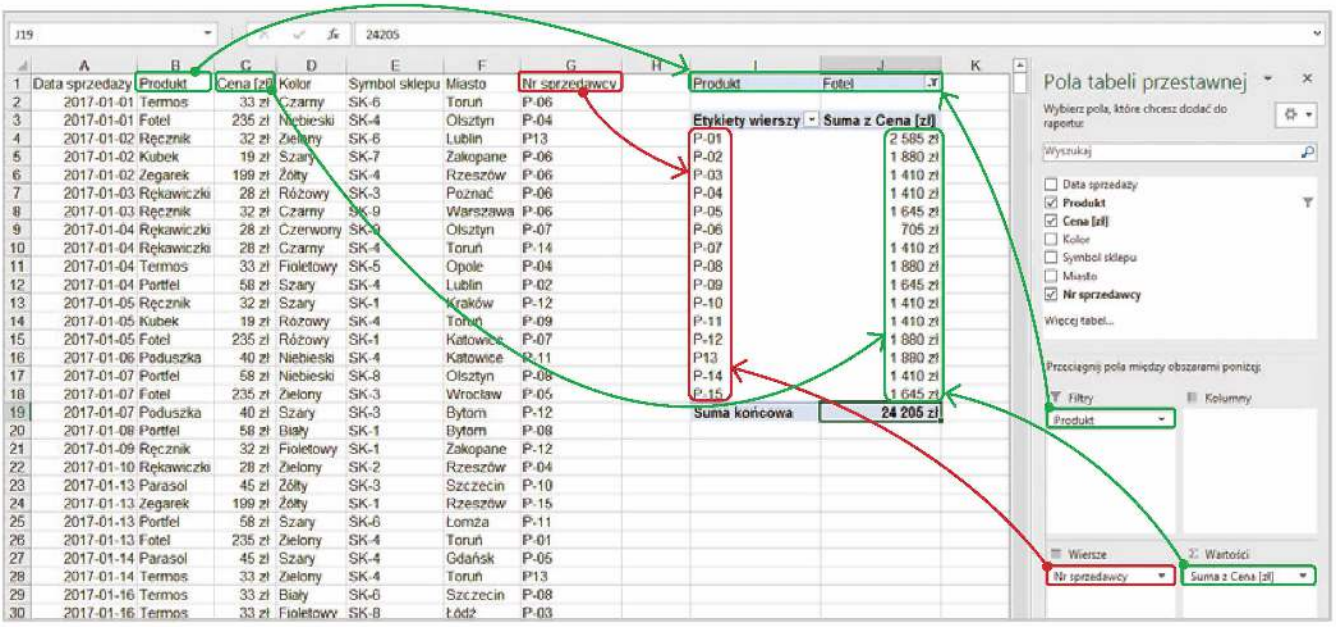
Warto wiedzieć

Wartości wyświetlane jako `Etykiety wierszy` lub `Etykiety kolumn` nie powtarzają się.



Rys. 2.12. Projektowanie tabeli przestawnej

Schemat działania tabel przestawnych pokazuje rysunek 2.13. Na podstawie danych zawartych w kolumnach arkusza utworzono w tym samym arkuszu tabelę przestawną. Znajduje się ona w komórkach z zakresu I1:J19. Strzałki z prawej strony tabeli wskazują elementy tabeli, które zostały utworzone przez umieszczenie odpowiednich pól w wybranych obszarach. Strzałki z lewej strony tabeli wskazują nagłówki kolumn z danych źródłowych: B1, C1 oraz G1. Danych z tych kolumn użyto do utworzenia tabeli przestawnej.

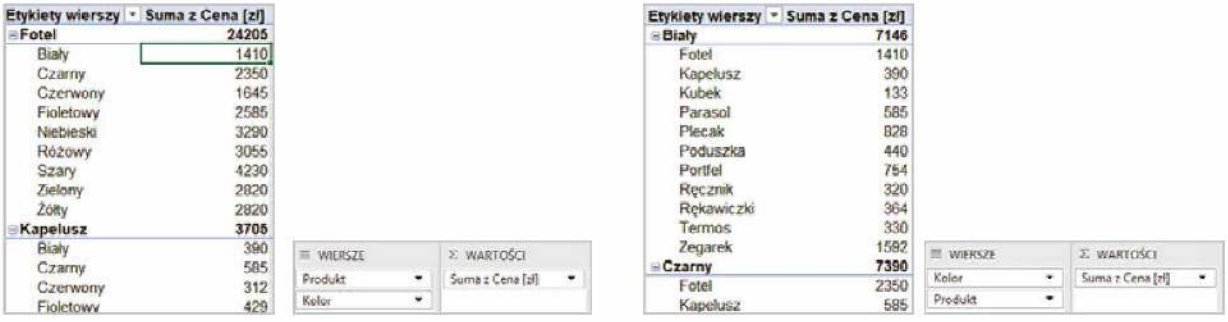


Rys. 2.13. Zależność między danymi a tabelą przestawną, w której pokazano wartość sprzedaży foteli, jaką uzyskał każdy sprzedawca

Ćwiczenie 6

Na podstawie pliku otrzymanego od nauczyciela (np. *rozmowy_tel.xlsx*), wykonaj tabelę przestawną i ustal, ile było łącznie połączeń wychodzących i przychodzących.

W jednym obszarze możemy umieszczać wiele pól. Kolejność pól w obszarze **Kolumny** lub **Wiersze** decyduje, które pole będzie elementem nadrzędnym w tabeli. Przy nim właśnie będzie możliwość zwijania i rozwijania widoku w tabeli przestawnej. Pokazuje to rysunek 2.14.



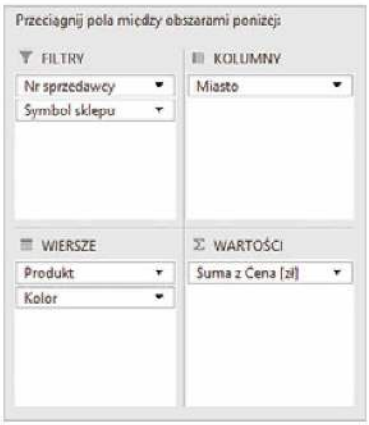
Rys. 2.14. Dwie konfiguracje etykiet wierszy

Warto wiedzieć

Żadne pole tabeli przestawnej nie może występować jednocześnie w obszarze **Wiersze** i obszarze **Kolumny**.

Przy tworzeniu tabeli przestawnej można umieszczać pola w kilku obszarach jednocześnie. Dzięki temu można zarządzać dużą ilością informacji. Informacje te możemy jeszcze zawężać dzięki filtrowaniu. Na rysunku 2.15 pokazano wyniki sprzedaży pracownika o numerze P–06 w sklepie o symbolu SK–9. Ponieważ nie używamy tutaj dat, chodzi o cały okres, którego dotyczą dane. Pracownik ten pracował w dwóch miastach (Warszawie i Krakowie), gdzie sprzedał jedynie parasol (w Krakowie) i ręcznik (w Warszawie). Łączna wartość sprzedaży to 77 zł.

	A	B	C	D
1	Nr sprzedawcy	P-06		
2	Symbol sklepu	SK-9		
3				
4	**Suma z Cena [zł]**	**Etykiety kolumn**		
5	**Etykiety wierszy**	**Kraków**	**Warszawa**	**Suma końcowa**
6	**Parasol**	**45**		**45**
7	Fioletowy	45		45
8	**Ręcznik**		**32**	**32**
9	Czarny		32	32
10	**Suma końcowa**	**45**	**32**	**77**
11				



Rys. 2.15. Wpływ rozmieszczenia pól w obszarach na tabelę przestawną

Możemy szybko sporządzić zestawienie, które pokaże wszystkie dane składające się na wartość wybraną w tabeli przestawnej. W tym celu wystarczy dwukrotnie kliknąć w komórkę z interesującą nas wartością – Excel utworzy nowy arkusz z odpowiednią szczegółową tabelą (rys. 2.16).

	A	B	C	D	E	F	G
1	**Data sprzedaży**	**Produkt**	**Cena [zł]**	**Kolor**	**Symbol sklepu**	**Miasto**	**Nr sprzedawcy**
2	03.01.2017	Ręcznik	32	Czarny	SK-9	Warszawa	P-06
3	19.04.2017	Parasol	45	Fioletowy	SK-9	Kraków	P-06

Rys. 2.16. Tabela w nowym arkuszu zawierająca wybrany fragment danych źródłowych

Ćwiczenie 7

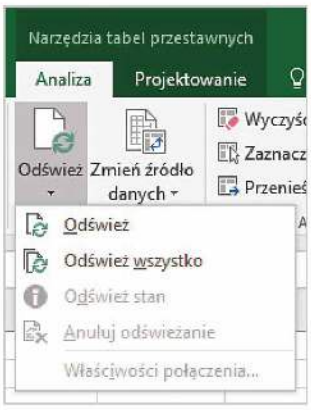
Na podstawie danych, które otrzymasz od nauczyciela (np. z pliku *malopolskie.xlsx*), utwórz tabelę przestawną pokazującą całkowitą wartość sprzedaży (sumę cen) wszystkich produktów w Krakowie.

Zapamiętaj

Tabele przestawne to interakcyjne narzędzie umożliwiające podsumowywanie, analizowanie i prezentowanie danych. Pomagają one podejmować przemyślane decyzje na podstawie zbiorów danych, którymi dysponujemy.

Aktualizacja tabeli przestawnej

Tabela przestawna nie jest automatycznie aktualizowana, nawet jeśli zmienią się dane w arkuszu, na podstawie których ją utworzono. Aby zaktualizować tabelę przestawną, należy umieścić kursor w dowolnej komórce jej obszaru i wybrać **Narzędzia tabel przestawnych → Analiza → Dane → Odśwież**. Aktualizacja może dotyczyć tylko jednej tabeli lub wszystkich tabel w pliku, na którym pracujemy (rys. 2.17).



Rys. 2.17. Odświeżenie tabeli po zmianie danych

Jeżeli tabelę przestawną utworzyliśmy na podstawie **tabeli Excela**, to zakres danych źródłowych zawsze będzie właściwy – nawet gdy w tabeli źródłowej dodamy lub usuniemy wiersze. Inaczej jest, gdy tabelę przestawną utworzyliśmy na podstawie zakresu komórek. Wówczas zanim zaktualizujemy tabelę przestawną, musimy się upewnić, że zakres komórek będący źródłem danych odpowiada zakresowi danych w tabeli.

Tabela programu Excel, s. 32

Dobra rada

Aby zmienić zakres danych tabeli przestawnej, wybierz **Analiza → Dane → Zmień źródło danych**.

Zapamiętaj

Tabele przestawne nie są aktualizowane automatycznie w wyniku zmiany danych źródłowych.

Formatowanie tabeli przestawnej

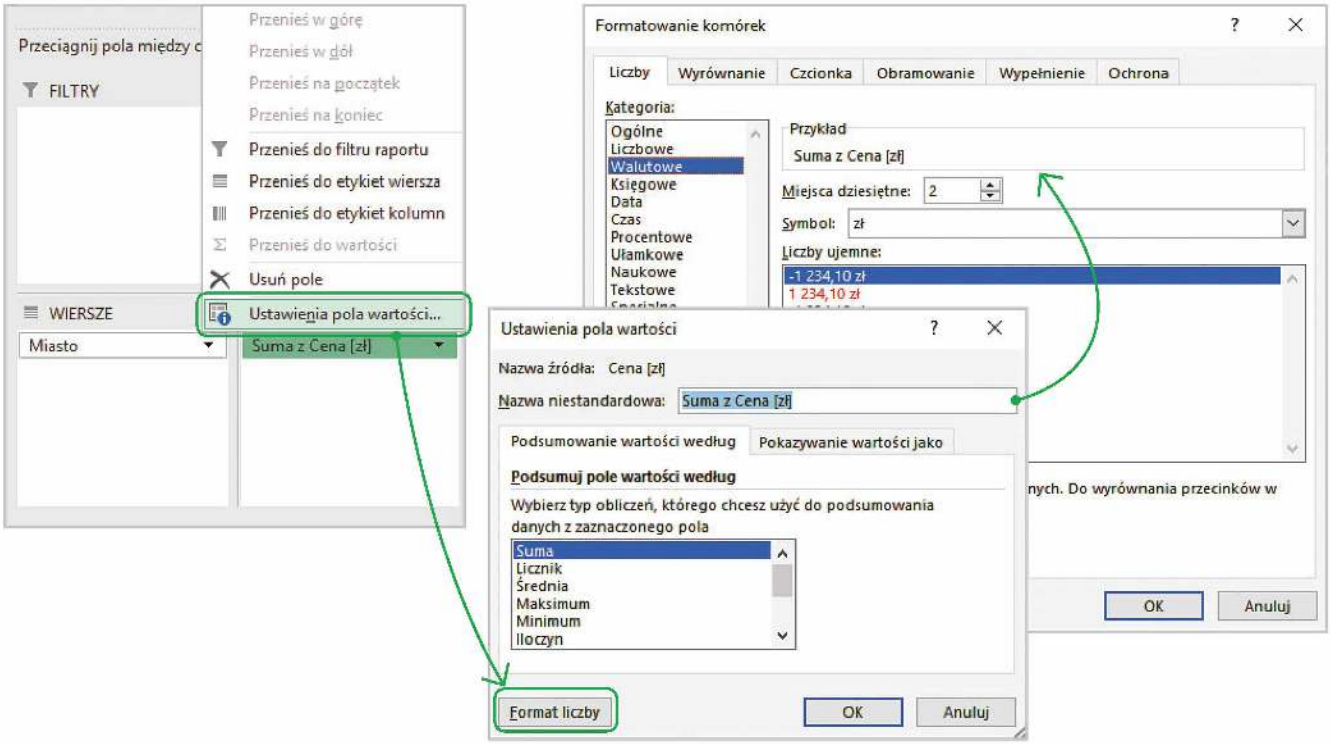
Komórki tabeli przestawnej można formatować tak jak zwykłe komórki arkusza – przydatne opcje znajdziemy np. na karcie **Narzędzia główne**. Z kolei dzięki opcji **Ustawienia pola wartości...**, dostępnej dla elementów w obszarze **Wartości**, można zmieniać format liczb w tabeli, by ustawić np. walutowy (rys. 2.18, s. 42).

A to ciekawe

Dane jak kostki w hurtowni?

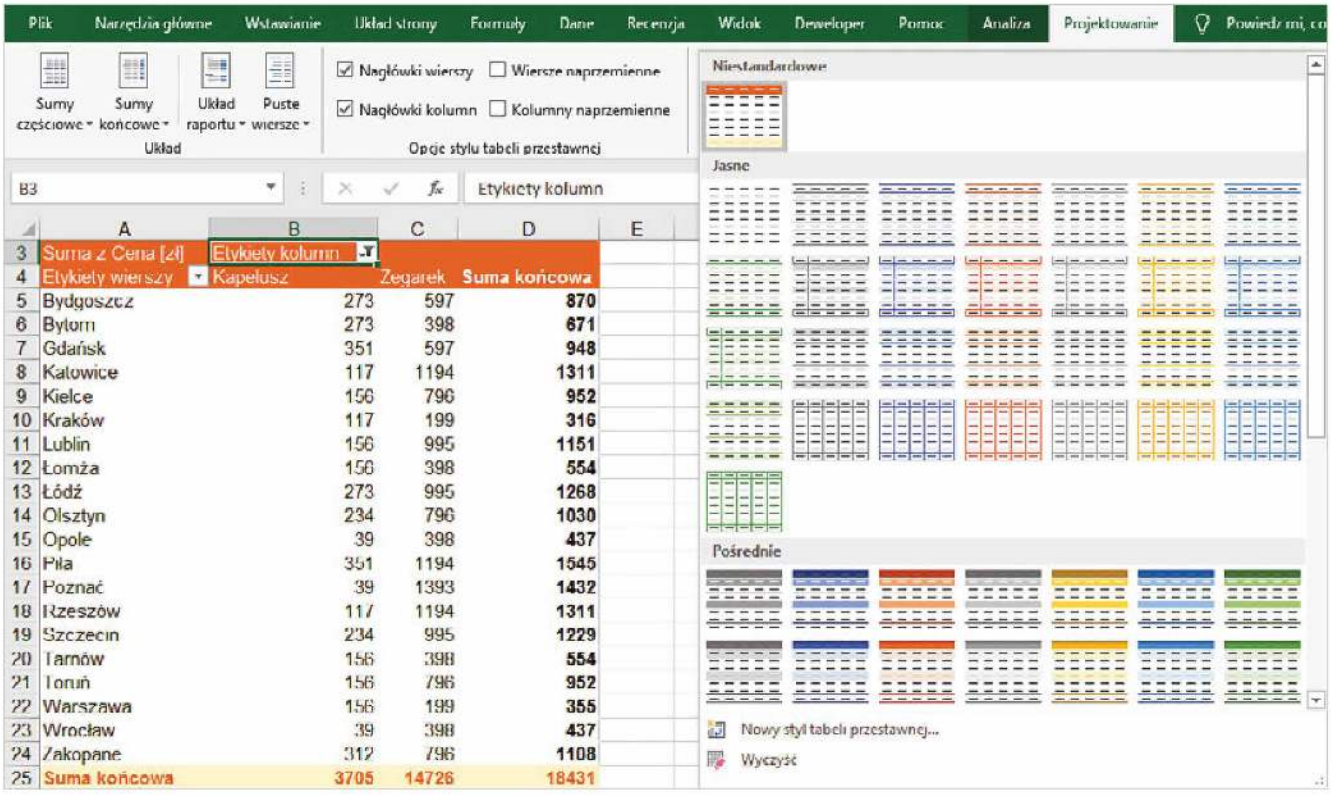
Ogromna ilość danych, które gromadzimy i przetwarzamy, przyczyniła się do powstania tzw. hurtowni danych (ang. *data warehouses*).
Nie są to fizyczne magazyny, ale cyfrowe bazy o pojemności setek terabajtów. Integrują one dane z wielu różnych źródeł, np. z systemów ERP, CRM lub arkuszy kalkulacyjnych. Dane przed załadowaniem do hurtowni są czyszczone i odpowiednio przekształcane.
Aby zapewnić maksymalną wydajność odczytu, przechowywane są najczęściej w bazach OLAP (ang. *online analytical processing*) w wielowymiarowych strukturach, tzw. kostkach (ang. *cubes*).
W Excelu można analizować dane z takich kostek.





Rys. 2.18. Formatowanie liczb w tabeli przestawnej

Wygląd tabeli możemy dostosować na karcie **Projektowanie**. Możemy tu wybrać gotowy styl lub stworzyć własny. Utworzony styl pojawi się na liście **Niestandardowe** (rys. 2.19) – można go przypisać tabeli, modyfikować lub usunąć (po uaktywnieniu menu kontekstowego prawym przyciskiem myszy).



Rys. 2.19. Style tabel przestawnych

Na karcie **Projektowanie** znajdziemy opcję **Układ raportu** – umożliwia ona dobranie formy prezentacji tabeli przestawnej. Domyślną opcją jest forma kompaktowa – optymalna pod względem czytelności.

Warto wiedzieć

W układzie raportu można również wybrać formę konspektu lub tabelaryczną.

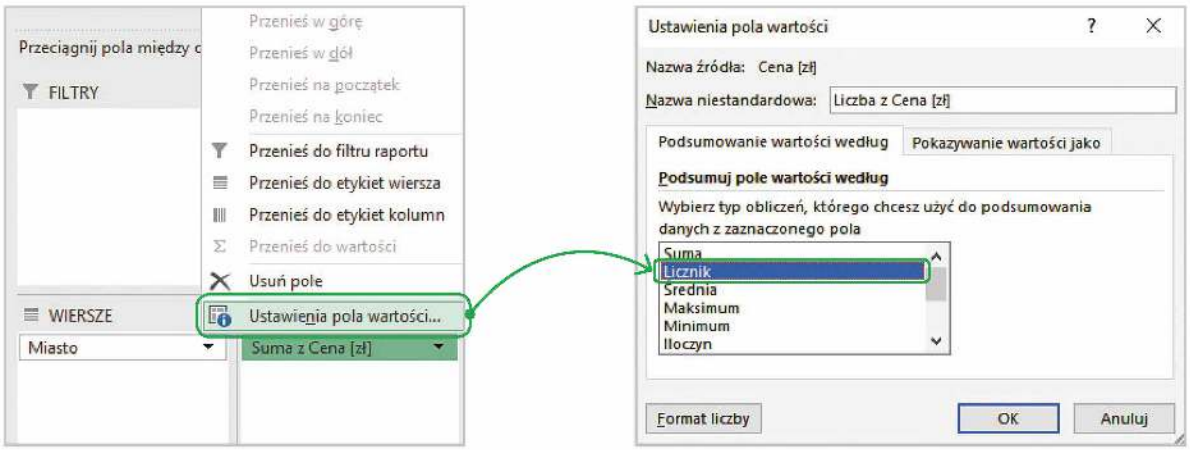
Ćwiczenie 8

Utwórz styl tabeli przestawnej według wzoru na rysunku 2.19. W tym celu wybierz opcję **Nowy styl tabeli przestawnej...** i sformatuj **Wiersz nagłówka** (kolor czcionki biały, wypełnienie pomarańczowe) oraz **Kolumnę sumy końcowej** (czcionka pogrubiona) i **Wiersz sumy końcowej** (czcionka pogrubiona, kolor pomarańczowy, wypełnienie jasnopomarańczowe).

Sposoby podsumowywania danych w tabelach przestawnych

Zwróć uwagę, że umieszczenie pola Cena w obszarze **Wartości** powoduje zmianę nazwy tego elementu na Suma z Cena (widać to np. na rys. 2.15, s. 40). Dzieje się tak dlatego, że sumowanie jest domyślnym sposobem podsumowywania pól w obszarze **Wartości**. Jeśli więc w tym obszarze umieścimy cenę produktu z naszego przykładu, tabela przestawna będzie zawierała podsumowanie cen wszystkich produktów, które spełnią kryteria określone w polach wierszy, kolumn czy filtrów. Otrzymamy zatem całkowitą wartość sprzedaży dla takiej konfiguracji tabeli przestawnej.

Wynik jednak nie musi być sumą cen, możemy równie dobrze policzyć, ile elementów spełnia określone kryteria. W tym celu należy kliknąć na polu Suma z Cena w obszarze **Wartości**, następnie wybrać opcję **Ustawienia pola wartości...** i zmienić **Suma** na **Licznik** (rys. 2.20).

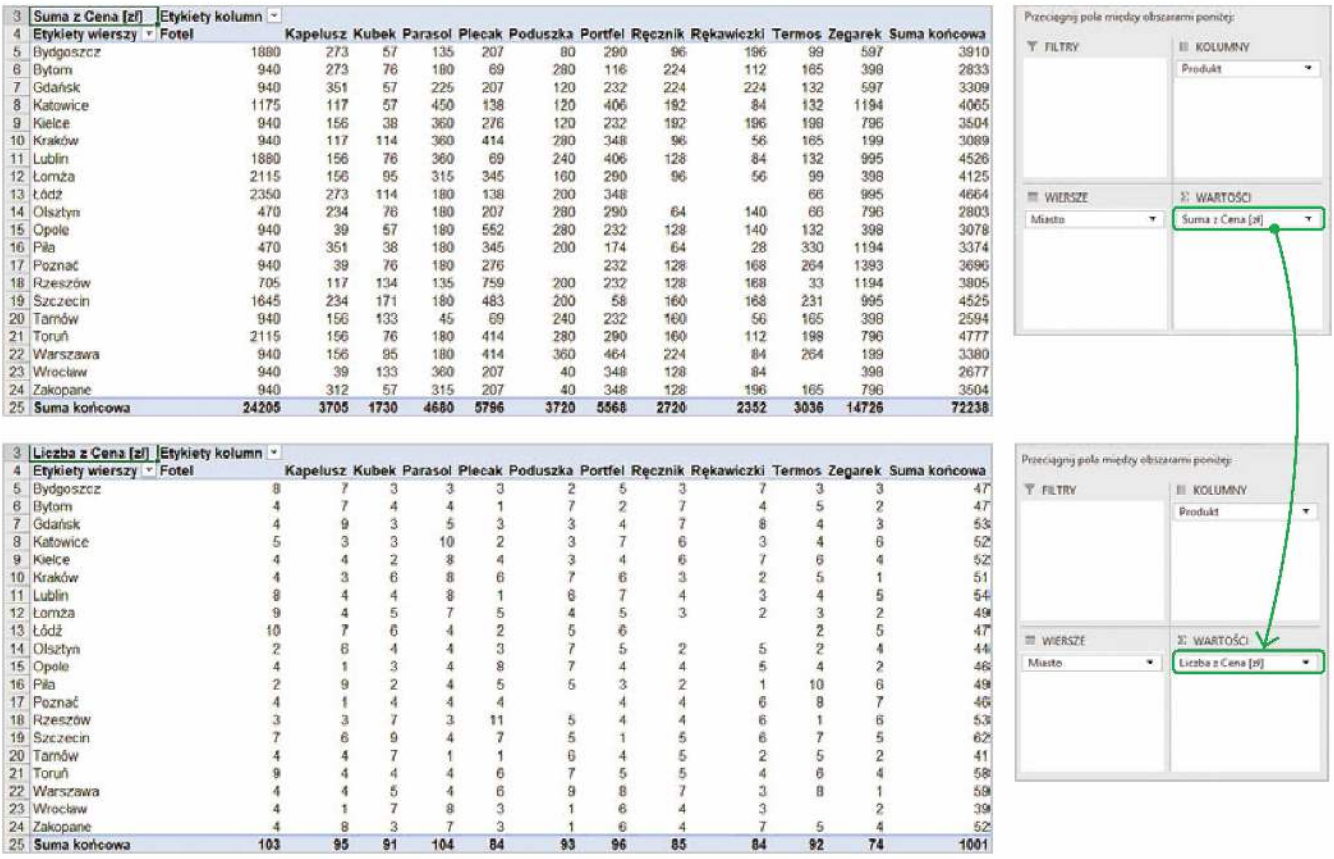


Rys. 2.20. Ustawienia pola wartości

Efekt zastosowania różnych ustawień dla pola wartości ilustruje rysunek 2.21 na s. 44. Pierwsza tabela pokazuje całkowitą wartość sprzedaży produktów w poszczególnych miastach – łącznie sprzedano towar za 72 238 zł. W drugiej tabeli widzimy natomiast, ile produktów sprzedano w każdym z miast – łącznie 1001 sztuk.

Warto wiedzieć

Listę wszystkich typów obliczeń stosowanych do podsumowania danych można znaleźć w **Ustawieniach pola wartości**.



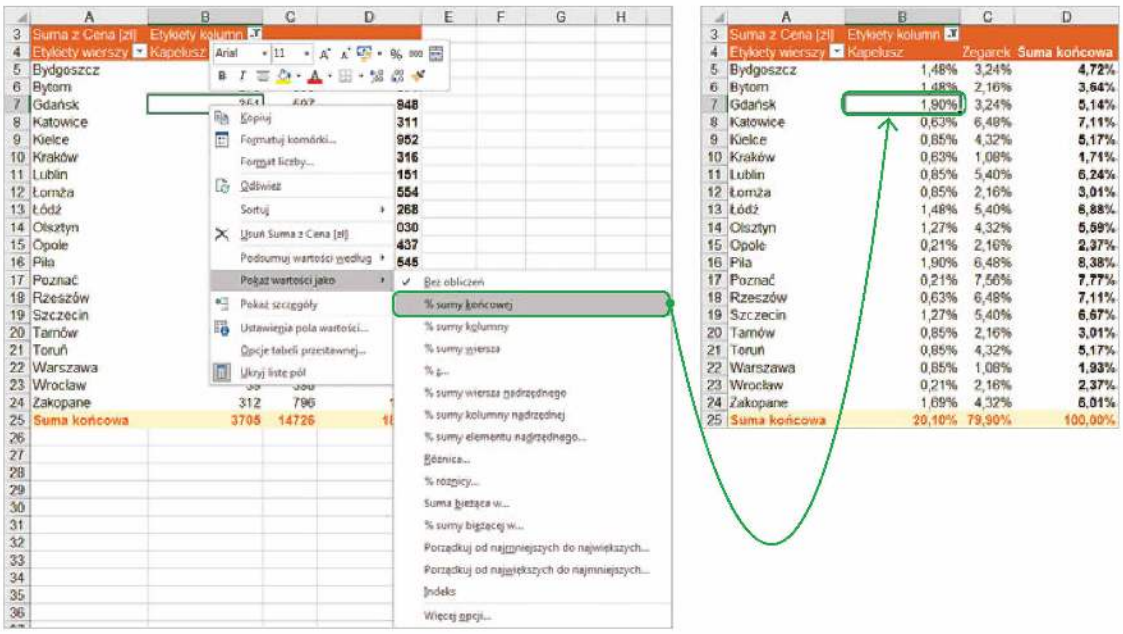
Suma z Cena [zł]	Etykiety kolumn											
Etykiety wierszy	Fotel	Kapelusz	Kubek	Parasol	Plecak	Poduszka	Portfel	Ręcznik	Rękawiczki	Termos	Zegarek	Suma końcowa
Bydgoszcz	1880	273	57	135	207	80	290	96	196	99	597	3910
Bytom	940	273	76	180	69	280	116	224	112	165	398	2833
Gdańsk	940	351	57	225	207	120	232	224	224	132	597	3309
Katowice	1175	117	57	450	138	120	406	192	84	132	1194	4065
Kielce	940	156	38	360	276	120	232	192	196	198	796	3504
Kraków	940	117	114	360	414	280	348	96	56	165	199	3089
Lublin	1880	156	76	360	69	240	406	128	84	132	995	4526
Łomża	2115	156	95	315	345	160	290	96	56	99	398	4125
Łódź	2350	273	114	180	138	200	348			66	995	4664
Olsztyn	470	234	76	180	207	280	290	64	140	66	796	2803
Opole	940	39	57	180	552	280	232	128	140	132	398	3078
Piła	470	351	38	180	345	200	174	64	28	330	1194	3374
Poznać	940	39	76	180	276		232	128	168	264	1393	3696
Rzeszów	705	117	134	135	759	200	232	128	168	33	1194	3805
Szczecin	1645	234	171	180	483	200	58	160	168	231	995	4525
Tarnów	940	156	133	45	69	240	232	160	56	165	398	2594
Toruń	2115	156	76	180	414	280	290	160	112	198	796	4777
Warszawa	940	156	95	180	414	360	464	224	84	264	199	3380
Wrocław	940	39	133	360	207	40	348	128	84		398	2677
Zakopane	940	312	57	315	207	40	348	128	196	165	796	3504
Suma końcowa	24205	3705	1730	4680	5796	3720	5568	2720	2352	3036	14726	72238

Liczba z Cena [zł]	Etykiety kolumn											
Etykiety wierszy	Fotel	Kapelusz	Kubek	Parasol	Plecak	Poduszka	Portfel	Ręcznik	Rękawiczki	Termos	Zegarek	Suma końcowa
Bydgoszcz	8	7	3	3	3	2	5	3	7	3	3	47
Bytom	4	7	4	4	1	7	2	7	4	5	2	47
Gdańsk	4	9	3	5	3	3	4	7	8	4	3	53
Katowice	5	3	3	10	2	3	7	6	3	4	6	52
Kielce	4	4	2	8	4	3	4	6	7	6	4	52
Kraków	4	3	6	8	6	7	6	3	2	5	1	51
Lublin	8	4	4	8	1	8	7	4	3	4	5	54
Łomża	9	4	5	7	5	4	5	3	2	3	2	49
Łódź	10	7	6	4	2	5	6			2	5	47
Olsztyn	2	6	4	4	3	7	5	2	5	2	4	44
Opole	4	1	3	4	8	7	4	4	5	4	2	46
Piła	2	9	2	4	5	5	3	2	1	10	6	49
Poznać	4	1	4	4	4		4	4	6	8	7	46
Rzeszów	3	3	7	3	11	5	4	4	6	1	6	53
Szczecin	7	6	9	4	7	5	1	5	6	7	5	62
Tarnów	4	4	7	1	1	6	4	5	2	5	2	41
Toruń	9	4	4	4	6	7	5	5	4	6	4	58
Warszawa	4	4	5	4	6	9	8	7	3	8	1	59
Wrocław	4	1	7	8	3	1	6	4	3		2	39
Zakopane	4	8	3	7	3	1	6	4	7	5	4	52
Suma końcowa	103	95	91	104	84	93	96	85	84	92	74	1001

Rys. 2.21. Zmiana sumowania wartości w tabeli przestawnej na zliczanie elementów

Sposoby wyświetlania wartości w tabelach przestawnych

Podczas prezentacji danych przydaje się możliwość wyświetlania ich w procentach. Aby z niej skorzystać, wystarczy kliknąć w komórce tabeli przestawnej prawym przyciskiem myszy i wybrać **Pokaż wartości jako**. Mamy do wyboru wiele opcji, np. **% sumy końcowej**, który pozwala wyświetlić procentowy rozkład wartości sprzedaży w stosunku do wartości całkowitej (rys. 2.22).



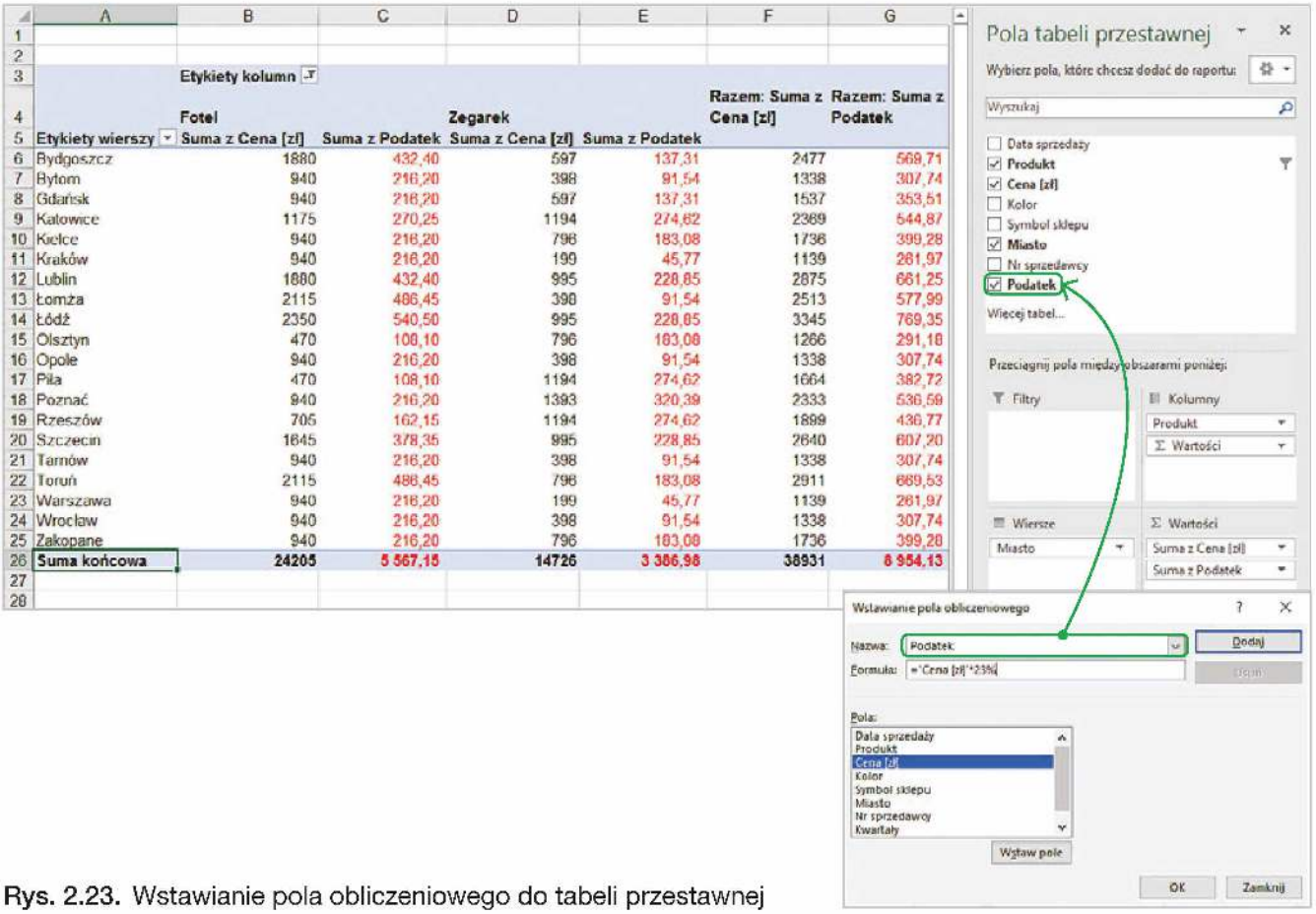
Suma z Cena [zł]	Etykiety kolumn		
Etykiety wierszy	Kapelusz	Zegarek	Suma końcowa
Bydgoszcz	1,48%	3,24%	4,72%
Bytom	1,48%	2,16%	3,64%
Gdańsk	1,90%	3,24%	5,14%
Katowice	0,63%	6,48%	7,11%
Kielce	0,85%	4,32%	5,17%
Kraków	0,63%	1,08%	1,71%
Lublin	0,85%	5,40%	6,24%
Łomża	0,85%	2,16%	3,01%
Łódź	1,48%	5,40%	6,88%
Olsztyn	1,27%	4,32%	5,59%
Opole	0,21%	2,16%	2,37%
Piła	1,90%	6,48%	8,38%
Poznać	0,21%	7,56%	7,77%
Rzeszów	0,63%	6,48%	7,11%
Szczecin	1,27%	5,40%	6,67%
Tarnów	0,85%	2,16%	3,01%
Toruń	0,85%	4,32%	5,17%
Warszawa	0,85%	1,08%	1,93%
Wrocław	0,21%	2,16%	2,37%
Zakopane	1,69%	4,32%	6,01%
Suma końcowa	20,10%	79,90%	100,00%

Rys. 2.22. Zmiana sposobu wyświetlania wartości w tabeli przestawnej

W tabeli przestawnej możemy również wykonywać bardziej złożone obliczenia. Policzmy podatek VAT w wysokości 23% od wartości sprzedaży zegarków i foteli w poszczególnych miastach (rys. 2.23). Wstawimy do tabeli **pole obliczeniowe**, które pozwoli wyznaczyć tę wartość.

Pole obliczeniowe

Tworzymy tabelę przestawną zawierającą wszystkie miasta i wybrane produkty (fotel i zegarek). W obszarze **Wartości** umieszczamy pole Cena. Zaznaczamy dowolną komórkę tabeli przestawnej i wybieramy **Analiza → Obliczenia → Pola, elementy i zestawy → Pole obliczeniowe...** W wyświetlonym oknie wpisujemy nazwę pola – „Podatek". Aby określić sposób obliczania wartości tego pola, na liście dostępnych pól zaznaczamy pole **Cena [zł]** i klikamy przycisk **Wstaw pole**. Następnie uzupełniamy zapis formuły w polu **Formuła**: ='Cena [zł]'*23% i zatwierdzamy przyciskiem **OK** (rys. 2.23). Na liście pól tabeli przestawnej pojawiło się pole Podatek, a w obszarze **Wartości** – Suma z Podatek. Kolumny z wyliczonym podatkiem są oznaczone na rysunku kolorem czerwonym.



Etykiety wierszy	Fotel Suma z Cena [zł]	Fotel Suma z Podatek	Zegarek Suma z Cena [zł]	Zegarek Suma z Podatek	Razem: Suma z Cena [zł]	Razem: Suma z Podatek
Bydgoszcz	1880	432,40	597	137,31	2477	569,71
Bytom	940	216,20	398	91,54	1338	307,74
Gdańsk	940	216,20	597	137,31	1537	353,51
Katowice	1175	270,25	1194	274,62	2369	544,87
Kielce	940	216,20	796	183,08	1736	399,28
Kraków	940	216,20	199	45,77	1139	261,97
Lublin	1880	432,40	995	228,85	2875	661,25
Łomża	2115	486,45	398	91,54	2513	577,99
Łódź	2350	540,50	995	228,85	3345	769,35
Olsztyn	470	108,10	796	183,08	1266	291,18
Opole	940	216,20	398	91,54	1338	307,74
Piła	470	108,10	1194	274,62	1664	382,72
Poznań	940	216,20	1393	320,39	2333	536,59
Rzeszów	705	162,15	1194	274,62	1899	436,77
Szczecin	1645	378,35	995	228,85	2640	607,20
Tarnów	940	216,20	398	91,54	1338	307,74
Toruń	2115	486,45	796	183,08	2911	669,53
Warszawa	940	216,20	199	45,77	1139	261,97
Wrocław	940	216,20	398	91,54	1338	307,74
Zakopane	940	216,20	796	183,08	1736	399,28
Suma końcowa	24205	5 567,15	14726	3 386,98	38931	8 954,13

Rys. 2.23. Wstawianie pola obliczeniowego do tabeli przestawnej

Ćwiczenie 9

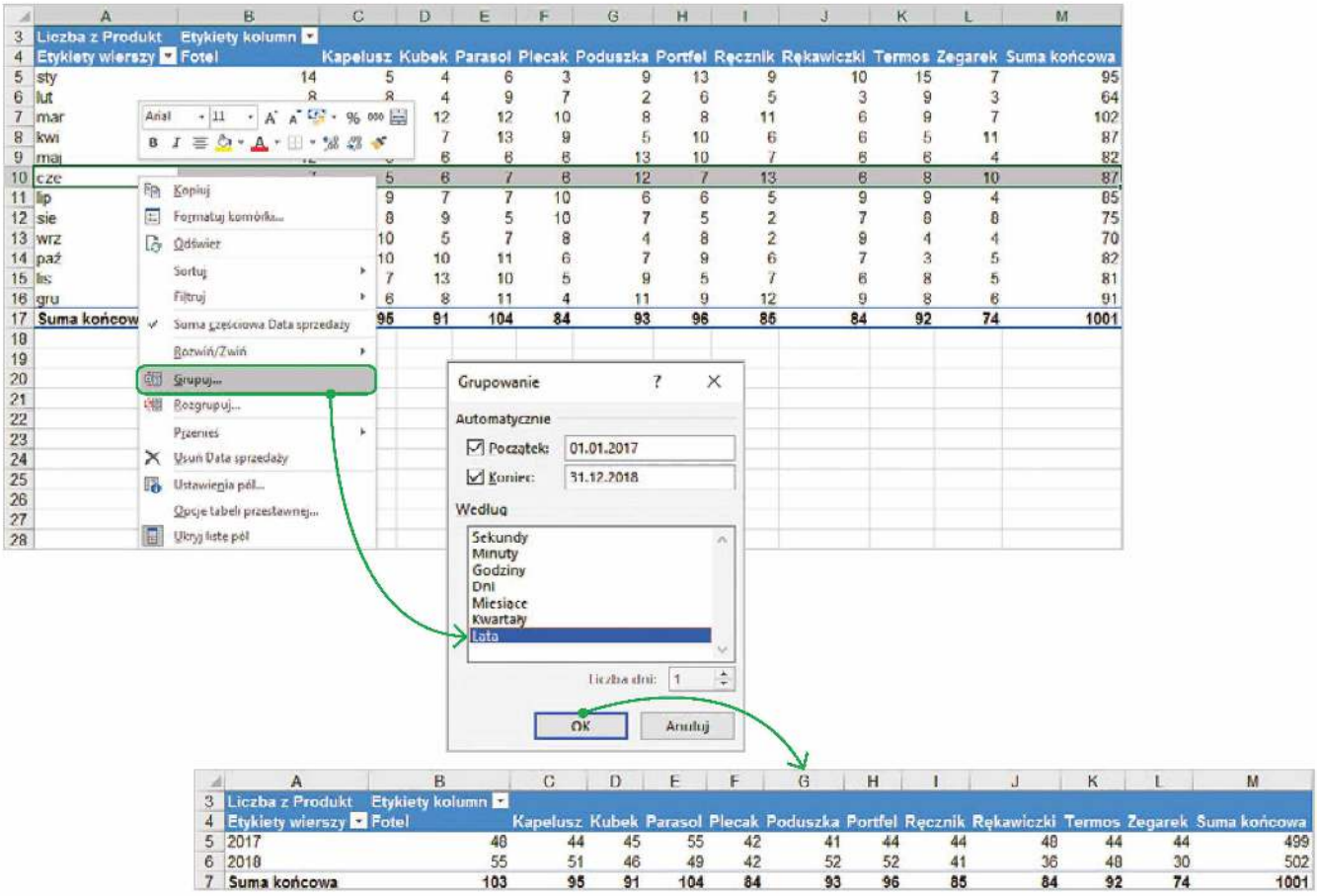
Na podstawie danych otrzymanych od nauczyciela (np. z pliku *handlowcy.xlsx*) utwórz tabelę przestawną, w której określisz procentowy udział każdego sprzedawcy w sprzedaży w danym mieście. Wskazówka: Skorzystaj z pokazywania wartości jako **% sumy wiersza nadrzędnego**.

Grupowanie dat w tabeli przestawnej

Warto wiedzieć

Dane można grupować hierarchicznie na kilku poziomach, np. dla czasu transakcji: godziny, minuty, sekundy.

Arkusz danych sprzedażowych, który w naszych przykładach wykorzystujemy do tworzenia tabel przestawnych, zawiera daty wszystkich transakcji. Daty te możemy grupować według miesięcy, kwartałów lub lat. Jeśli w obszarze **Wiersze** dodamy pole `Data sprzedaży`, program Excel automatycznie utworzy nowe pola `Lata` i `Kwartały` oraz wstępnie pogrupuje daty. Kliknięcie prawym przyciskiem myszy dowolnej komórki w kolumnie `Etykiety wierszy` daje dostęp do opcji grupowania dat (rys. 2.24).



Rys. 2.24. Grupowanie dat w tabeli przestawnej

Ćwiczenie 10

Korzystając z danych, które otrzymasz od nauczyciela (np. z pliku *media.xlsx*), znajdź kwartał o najwyższym zużyciu energii elektrycznej w 2017 r. Utwórz w tym celu tabelę przestawną.

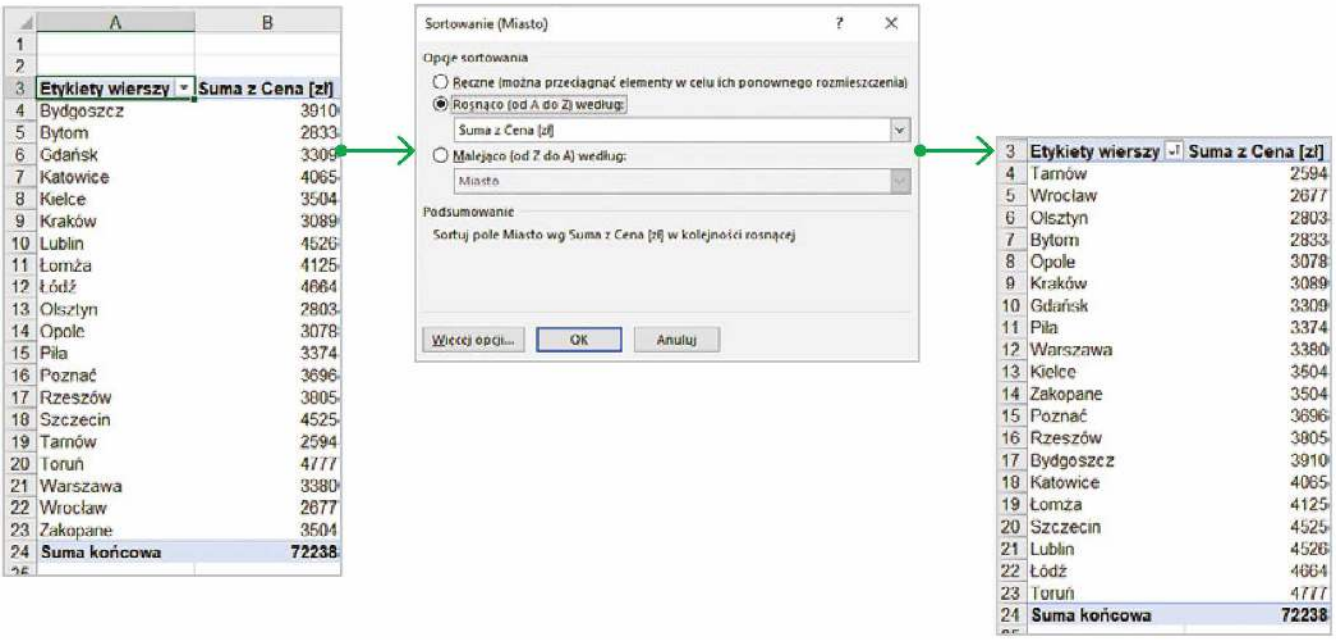
Sortowanie danych w tabeli przestawnej

Sortowanie w tabeli przestawnej

Podobnie jak w przypadku tabel Excela, dane w tabeli przestawnej możemy **sortować**. Jeżeli klikniemy małą strzałkę w kolumnie `Etykiety wierszy` (komórka A3 na rys. 2.25), możemy wybrać opcje alfabetycznego sortowania danych lub **Więcej opcji sortowania**.

Tu mamy możliwość **sortowania** ręcznego lub automatycznego (rosnąco albo malejąco) według wybranego elementu z tabeli przestawnej.

Sortowanie, s. 30



Etykiety wierszy	Suma z Cena [zł]
Bydgoszcz	3910
Bytom	2833
Gdańsk	3309
Katowice	4065
Kielce	3504
Kraków	3089
Lublin	4526
Łomża	4125
Łódź	4664
Olsztyn	2803
Opole	3078
Piła	3374
Poznać	3696
Rzeszów	3805
Szczecin	4525
Tarnów	2594
Toruń	4777
Warszawa	3380
Wrocław	2677
Zakopane	3504
Suma końcowa	72238

Etykiety wierszy	Suma z Cena [zł]
Tarnów	2594
Wrocław	2677
Olsztyn	2803
Bytom	2833
Opole	3078
Kraków	3089
Gdańsk	3309
Piła	3374
Warszawa	3380
Kielce	3504
Zakopane	3504
Poznać	3696
Rzeszów	3805
Bydgoszcz	3910
Katowice	4065
Łomża	4125
Szczecin	4525
Lublin	4526
Łódź	4664
Toruń	4777
Suma końcowa	72238

Rys. 2.25. Sortowanie danych w tabeli przestawnej

Ćwiczenie 11

Na podstawie danych, które otrzymasz od nauczyciela (np. z pliku *sprzedaz.xlsx*), wykonaj tabelę przestawną prezentującą wartości sprzedaży dla poszczególnych miast i posortuj wyniki rosnąco, jak na rysunku 2.25.

Filtrowanie danych w tabeli przestawnej

Podobnie jak dane w zakresach komórek czy tabelach programu Excel, dane w tabelach przestawnych również można filtrować. **Filtrowanie w tabelach przestawnych** ogranicza zestaw danych, który analizujemy. Dodatkowo dzięki operatorom warunkowym możemy tworzyć złożone filtry etykiet lub wartości w tabeli przestawnej.

Filtrowanie w tabeli przestawnej

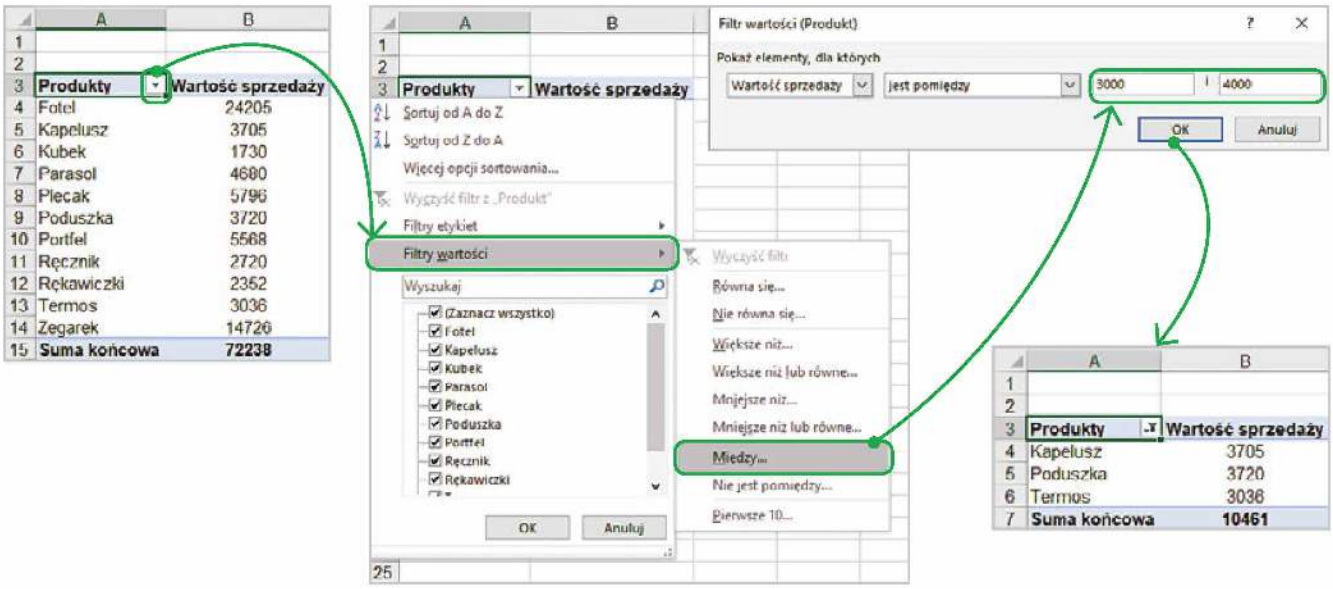
Jeden ze sposobów **filtrowania danych** w tabeli, który pokazaliśmy wcześniej, polega na umieszczeniu wybranego pola w obszarze **Filtry** (rys. 2.15, s. 40).

Filtrowanie, s. 30

Na podstawie danych sprzedażowych wyświetlimy w tabeli przestawnej produkty, których całkowita wartość sprzedaży mieści się pomiędzy 3000 a 4000 zł (rys. 2.26, s. 48). W tym celu tworzymy tabelę przestawną i przeciągamy pole Produkt do obszaru **Wiersze**, a pole Cena – do obszaru **Wartości**. Po rozwinięciu etykiety Produkty (nazwy kolumn zostały tu zmienione względem oryginalnych elementów) wybieramy filtr wartości **Między...** Określamy zakres pomiędzy 3000 a 4000 i zatwierdzamy przyciskiem **OK**.

Warto wiedzieć

Nazwy kolumn w tabeli przestawnej pochodzą od nazw elementów, które umieszczamy w polach tabeli. Nazwy te można jednak dowolnie zmieniać w samej tabeli przestawnej.



Rys. 2.26. Filtrowanie produktów względem wartości sprzedaży

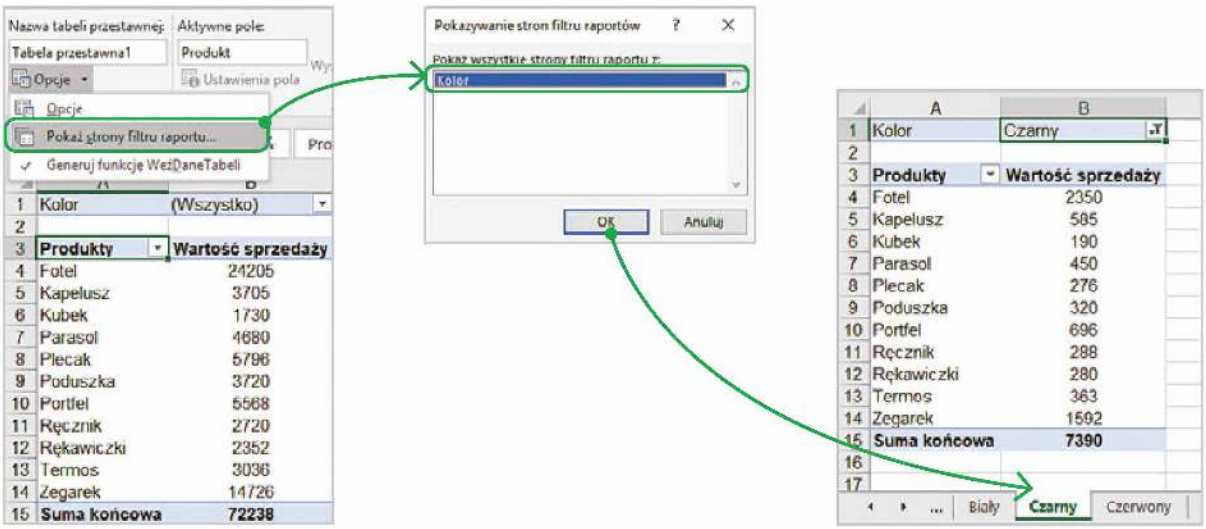
Ćwiczenie 12

Na podstawie danych, które otrzymasz od nauczyciela (np. z pliku *koncerty.xlsx*), wyświetl pięć miast, w których odbyło się najwięcej koncertów. Skorzystaj z **Filtrów wartości** – wybierz filtr o nazwie **Pierwsze 10...** Pamiętaj, aby wcześniej zmienić sumowanie wartości na ich zliczanie.

Warto wiedzieć

Kolejność pól umieszczonych w obszarze **Filtry** nie wpływa na wynik filtrowania.

Obszaru **Filtry** w tabeli przestawnej możemy również użyć do wygenerowania dodatkowych arkuszy zawierających raporty na temat elementów, które filtrujemy. Jeśli np. umieścimy w obszarze **Filtry** kolor produktu (pole Kolor), a tabela zawiera wartości sprzedaży produktów, po użyciu opcji **Pokaż strony filtru raportu...** z karty **Analiza** otrzymamy arkusze o nazwach poszczególnych kolorów. Każda tabela przestawna w osobnym arkuszu będzie dotyczyła danego koloru (rys. 2.27).



Rys. 2.27. Tworzenie arkuszy z tabelami przestawnymi na podstawie pola w obszarze **Filtry**

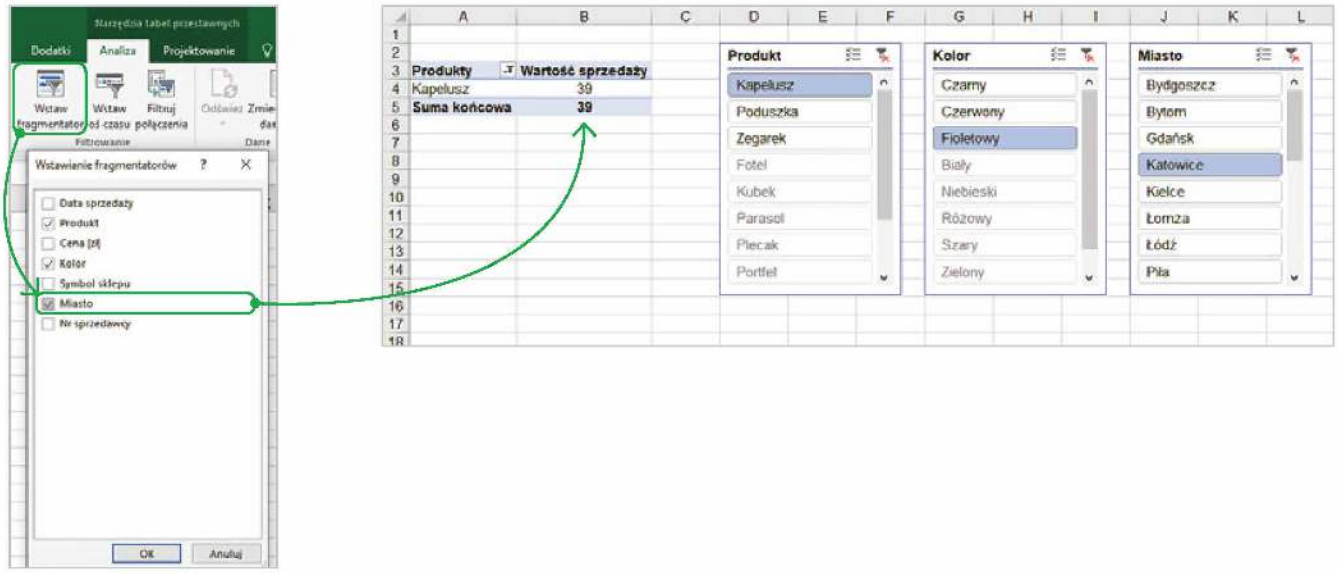
Fragmentatory i oś czasu

Przydatnym narzędziem wizualnego filtrowania danych w tabeli przestawnej są tzw. **fragmentatory**. Są to zgrupowane w okienka przyciski odpowiadające wybranym filtrom. Aby wstawić fragmentator, umieszczamy kursor w tabeli i wybieramy **Analiza → Filtrowanie → Wstaw fragmentator**. Następnie wybieramy pola, które mają decydować o filtrowaniu (rys. 2.28). Teraz bardzo łatwo sprawdzimy np. wartość sprzedaży fioletowych kapeluszy w Katowicach.

Fragmentator

Dobra rada

W programie Excel 2013 (lub nowszym) możesz stosować fragmentatory również do filtrowania zwykłych tabel.



Rys. 2.28. Wstawianie fragmentatora i filtrowanie za jego pomocą

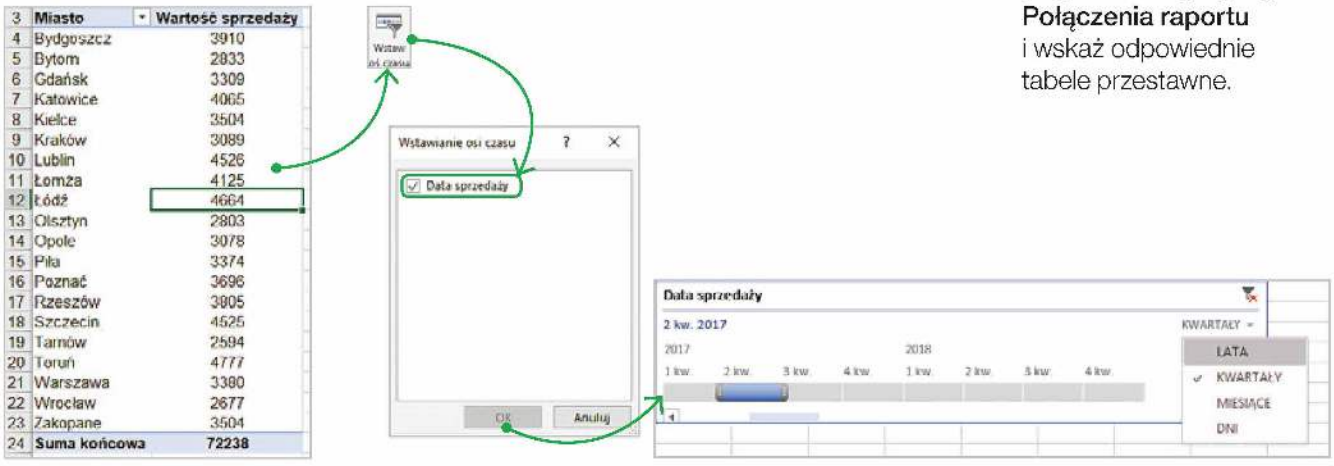
Obok nagłówka fragmentatora są dwa przyciski. Pierwszy po włączeniu pozwala wybrać wiele przycisków, drugi – usuwa filtr.

Oprócz fragmentatorów mamy możliwość wstawienia **osi czasu** (**Analiza → Filtrowanie → Wstaw oś czasu**). Ta funkcjonalność pozwala interaktywnie filtrować tabelę przestawną, uwzględniając daty pogrupowane w okresy – lata, kwartały, miesiące, dni (rys. 2.29). Klikając w szarych polach osi czasu, możemy łatwo zaznaczać wybrane okresy, a rozciągając zaznaczenie w lewo lub w prawo – zmieniać zakres dat.

Oś czasu

Dobra rada

Aby za pomocą fragmentatora jednocześnie filtrować kilka tabel przestawnych, kliknij na nim prawym przyciskiem myszy, wybierz **Połączenia raportu** i wskaż odpowiednie tabele przestawne.



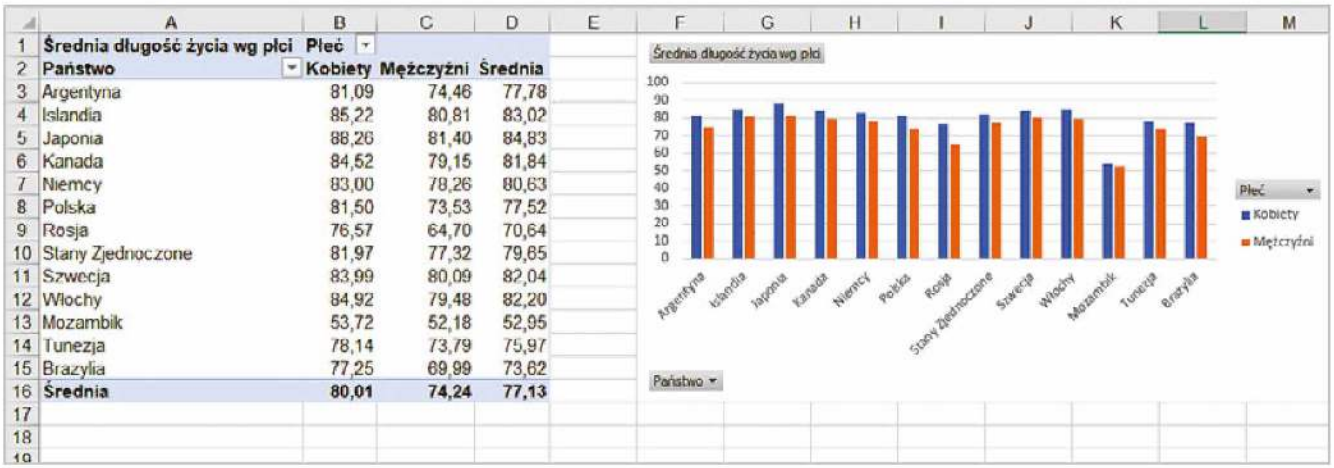
Rys. 2.29. Filtrowanie przy użyciu osi czasu

2.3. Wykresy przestawne

Tabele przestawne w programie Excel są użytecznym narzędziem do analizy dużych zbiorów danych. Jednak oprócz prezentacji danych w postaci tabel czy zestawień wyników Excel daje również możliwość tworzenia **wykresów przestawnych** (rys. 2.30).

Wykres przestawny

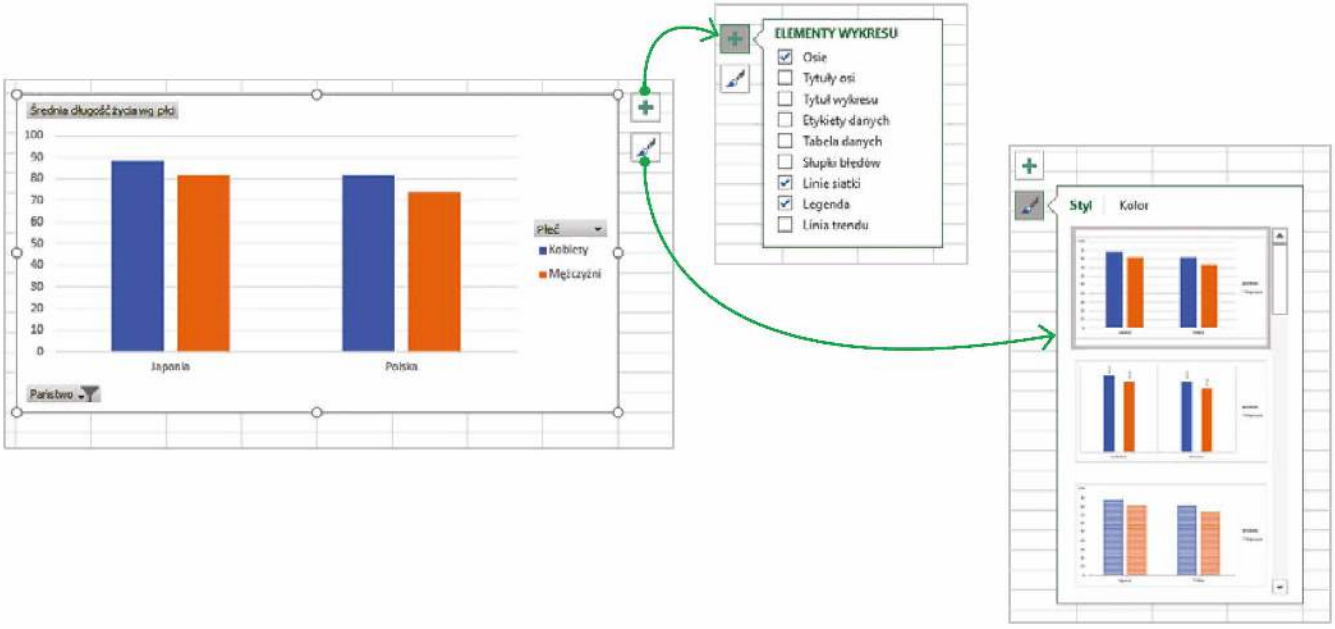
Wykres na podstawie danych tabeli przestawnej wstawiamy, kiedy kursor jest wewnątrz tabeli, wybierając **Analiza** → **Narzędzia** → **Wykres przestawny**. Większość wykresów jest dostępna. Te, których nie można wstawić, są nieaktywne. Elementy znajdujące się na wykresie oraz sam wykres możemy edytować za pomocą narzędzi z karty **Projektowanie** (aktywnej, kiedy wykres jest zaznaczony).



Średnia długość życia wg płci	Płeć		
Państwo	Kobiety	Mężczyźni	Średnia
Argentyna	81,09	74,46	77,78
Islandia	85,22	80,81	83,02
Japonia	88,26	81,40	84,83
Kanada	84,52	79,15	81,84
Niemcy	83,00	78,26	80,63
Polska	81,50	73,53	77,52
Rosja	76,57	64,70	70,64
Stany Zjednoczone	81,97	77,32	79,65
Szwecja	83,99	80,09	82,04
Włochy	84,92	79,48	82,20
Mozambik	53,72	52,18	52,95
Tunezja	78,14	73,79	75,97
Brazylia	77,25	69,99	73,62
Średnia	**80,01**	**74,24**	**77,13**

Rys. 2.30. Przykład wykresu przestawnego

Po zaznaczeniu wykresu pojawiają się obok niego przyciski formatowania, dzięki którym można wybierać elementy pokazywane na wykresie oraz zmieniać styl wykresu (rys. 2.31).



Rys. 2.31. Wybrane opcje formatowania zaznaczonego wykresu

Etykiety na wykresie mają postać przycisków – dzięki nim prezentowane dane można sortować i filtrować dokładnie tak samo jak w tabeli przestawnej.

Ćwiczenie 13

Na podstawie danych, które otrzymasz od nauczyciela (np. z pliku *zycie.xlsx*), wykonaj wykres przestawny prezentujący średnią długość życia kobiet w poszczególnych krajach. Posortuj dane rosnąco według średniej długości życia. W tym celu kliknij na wykresie przycisk **Kontynent** i wybierz **Więcej opcji sortowania**.

Dla bardziej złożonych tabel przestawnych, zawierających więcej niż jedno pole w obszarze **Kolumny** lub **Wiersze**, zarówno tabela przestawna, jak i wykres przestawny dają możliwość rozwijania bądź zwijania widoku odpowiednich elementów. Na wykresie w prawym dolnym rogu pojawiają się wtedy znaki + i – (rys. 2.32).

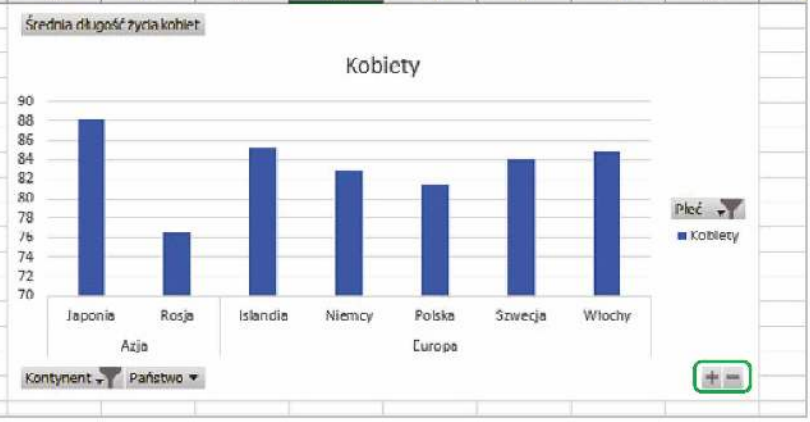
Dobra rada

Jeśli chcesz, aby na wykresie nie było przycisków, możesz je ukryć. Kliknij na przycisku prawym przyciskiem myszy i wybierz odpowiednią opcję. Przywrócić widoczność przycisku można na karcie **Analiza** → **Przyciski pól**.

Dobra rada

Jeśli chcesz, aby przyciski na wykresie miały takie nazwy jak na rysunku 2.32, zanim zrobisz wykres, nazwij odpowiednie kolumny w tabeli przestawnej.

Średnia długość życia kobiet	Płeć
Lokalizacja	Kobiety
Azja	82,42
Japonia	88,26
Rosja	76,57
Europa	83,73
Islandia	85,22
Niemcy	83,00
Polska	81,50
Szwecja	83,99
Włochy	84,92
Suma końcowa	83,35



Rys. 2.32. Wykres prezentujący średnią długość życia kobiet

A to ciekawe

Sztuczna inteligencja wokół nas

Doskonałym narzędziem do przetwarzania dużych zbiorów danych i pozyskiwania z nich wiedzy są sztuczne sieci neuronowe. Te bazujące na rzeczywistych strukturach i systemach uczących się modele stosuje się w różnych dziedzinach życia. Ciekawym przykładem sztucznych sieci neuronowych są GAN (ang. *generative adversarial network*). Sieci tego typu potrafią tworzyć realistyczne obrazy, m.in. twarze, które wyglądają jak autentyczne fotografie. Linki do przykładowych obrazów można znaleźć na stronie https://research.nvidia.com/publication/2017-10_Progressive-Growing-of.



Podsumowanie

- Zbiory danych wygodnie jest przechowywać w tabelach programu Excel. Każda tabela ma nazwę, a pierwszy wiersz tabeli zawiera nazwy kolumn.
- Tabele automatycznie dostosowują swój rozmiar, gdy dodamy lub usuniemy wiersz albo kolumnę tabeli. Nazwa tabeli wskazuje na zakres danych bez nagłówków kolumn.
- Tabele przestawne to interakcyjne narzędzia arkusza kalkulacyjnego ułatwiające analizę dużych zestawów danych.
- Praca z tabelami przestawnymi polega głównie na ich projektowaniu przez umieszczanie wybranych pól w odpowiednich obszarach (**Kolumny**, **Wiersze**, **Filtry**, **Wartości**). Tabelę przestawną można utworzyć na podstawie zakresu komórek lub tabeli programu Excel.
- Analizy z wykorzystaniem tabel przestawnych są stosowane w wielu firmach do wspomagania procesów decyzyjnych.
- Tabele przestawne umożliwiają wykonywanie różnych obliczeń, m.in. sumowanie i zliczanie elementów. Dane w tabelach przestawnych mogą być sortowane i filtrowane.
- Fragmentatory i osie czasu są wizualnymi elementami, które pozwalają filtrować dane w tabelach.
- Prezentacje analiz wykonywanych za pomocą tabel przestawnych można wzbogacić o wykresy przestawne, w których na bieżąco można wpływać na przedstawiane charakterystyki. Wykresy przestawne to interaktywne wykresy utworzone na podstawie tabel przestawnych.
- Tabele programu Excel oraz tabele przestawne możemy formatować, stosując dostępne w programie style. Możemy również definiować własne style.

Zadania

* **1** Na podstawie pliku, który otrzymasz od nauczyciela (np. *rosliny.xlsx*), wykonaj następujące polecenia.

a. Dodaj wiersz poprzedzający dane i nadaj nazwy poszczególnym kolumnom.

b. Przekształć zakres komórek z danymi w tabelę Excela o nazwie `Rośliny`.

* **2** Na podstawie danych z pliku, który otrzymasz od nauczyciela (np. *slaskie.xlsx*), podaj całkowitą wartość sprzedaży zegarków w Katowicach.

* **3** Na podstawie danych, które przekaże ci nauczyciel (np. z pliku *wielkopolskie.xlsx*), odpowiedz, ile czerwonych portfeli sprzedano.

* **4** Na podstawie danych z pliku otrzymanego od nauczyciela (np. *lubelskie.xlsx*) wskaż, w którym roku wartość sprzedaży wszystkich produktów była najwyższa.

* **5** Utwórz własny styl formatowania tabeli i zastosuj go do tabeli przestawnej, w której pokażesz całkowitą liczbę koncertów w Katowicach. Zadanie wykonaj na podstawie danych z pliku, który otrzymasz od nauczyciela (np. *koncerty.xlsx*).

** **6** Na podstawie danych z pliku, który przekaże ci nauczyciel (np. *prawo_jazdy.xlsx*), odpowiedz, ile kobiet uzyskało prawo jazdy w Poznaniu, a ile w Zakopanem.

** **7** Na podstawie danych z pliku otrzymanego od nauczyciela (np. *rozgrywki.xlsx*) podaj liczbę punktów uzyskanych w pierwszym kwartale 2018 r. przez gracza o identyfikatorze P–12.

** **8** Przedstaw za pomocą wykresu, jak kształtowała się wartość eksportu jabłek w poszczególnych miesiącach 2018 r. Zadanie wykonaj na podstawie danych z pliku, który przekaże ci nauczyciel (np. *eksport.xlsx*).

** **9** Na podstawie danych z pliku, który otrzymasz od nauczyciela (np. *narodziny.xlsx*), wskaż, w którym miesiącu 2017 r. na świat przyszło najwięcej chłopców.

** **10** Na podstawie danych z pliku przekazanego przez nauczyciela (np. *auta.xlsx*) pokaż na wykresie przestawnym kwartalną sprzedaż (liczbę sztuk) trzech wybranych modeli aut w 2017 r.

** **11** Z serwisu internetowego https://dane.gov.pl/ pobierz rejestr zabytków archeologicznych. Na podstawie pliku:

a. przygotuj zestawienie liczby zabytków w poszczególnych województwach,
b. wskaż trzy najpopularniejsze funkcje zabytków,
c. ustal, z którego okresu historycznego pochodzi najwięcej zabytków,
d. podaj listę dziesięciu miejscowości, w których zarejestrowano najwięcej zabytków,
e. wskaż miesiące, w których wpisów do rejestru było najmniej, oraz miesiąc, w którym było ich najwięcej. Podaj możliwą przyczynę takiego stanu rzeczy.

*** **12** Utwórz tabelę przestawną prezentującą sprzedaż biletów na różne koncerty (np. data zakupu biletu, artysta, temat koncertu, rodzaj muzyki), którą będzie można analizować za pomocą fragmentatorów i osi czasu.

*** **13** Na podstawie danych z pliku otrzymanego od nauczyciela (np. *premie.xlsx*) podaj łączną wartość premii przyznanej pracownikom księgowości w 2018 r. Przyjmij, że premia wynosi 5%.

*** **14** Na podstawie programu festiwalu filmowego, który otrzymasz od nauczyciela (np. *festiwal_filmowy.xlsx*), przygotuj w kolejnych arkuszach:

a. alfabetyczny spis filmów, zawierający informację o reżyserze i roku powstania filmu,
b. listę reżyserów, których filmy będą wyświetlane na festiwalu (nazwiska reżyserów nie powinny się powtarzać),
c. osobny program na każdy dzień festiwalu z informacją o tytule filmu, reżyserze, roku powstania oraz godzinie wyświetlania i sali,
d. spis filmów dla każdej sali, w której odbędzie się projekcja 12.06.2021 r.,
e. listę pięciu sal najczęściej wykorzystywanych podczas festiwalu.

3. Korespondencja seryjna

Czy zdarzyło ci się mieć w ręku list z banku lub urzędu? W takim liście często pojawia się odmienione imię adresata, zwykle występują też odpowiednie zwroty grzecznościowe, dostosowane do płci odbiorcy, np. „Szanowna Pani" lub „Szanowny Panie". Tekst może również zawierać spersonalizowaną ofertę. Takie listy wysyła się bardzo często w dużej ilości do klientów różnych instytucji. Praktycznie zawsze tworzy się je za pomocą narzędzi komputerowych, które pozwalają personalizować treści, a także filtrować adresatów i dzielić ich na kategorie. Warto poznać te narzędzia, aby nauczyć się szybko i sprawnie przygotowywać korespondencję kierowaną do wielu odbiorców.

Cele lekcji

- Nauczysz się tworzyć i personalizować listy seryjne.
- Utworzysz zewnętrzne źródła danych adresowych.
- Dowiesz się, jak filtrować adresy i drukować jedynie wybrane listy.
- Zastosujesz arkusz kalkulacyjny, aby lepiej spersonalizować korespondencję.

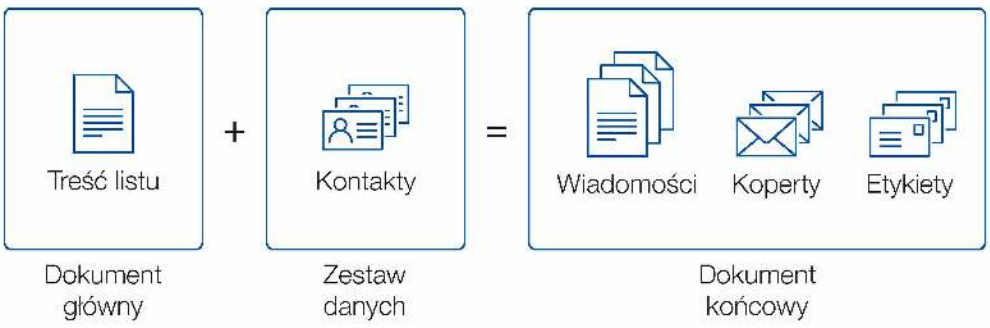
3.1. Czym jest korespondencja seryjna?

Korespondencja seryjna

Korespondencja seryjna to technika, która umożliwia przygotowanie listu jednocześnie do wielu adresatów. Można przy tym spersonalizować treść wiadomości i filtrować osoby, które powinny ją otrzymać. Jest to niezwykle wygodne narzędzie nie tylko w pracy biurowej, ale wszędzie tam, gdzie trzeba utrzymywać kontakt z klientami, prezentować firmowe oferty, wysyłać zaproszenia, informacje, powiadomienia itp.

Dokument główny, zestaw danych, dokument końcowy korespondencji seryjnej

Korespondencja seryjna opiera się na dwóch plikach. Pierwszy z nich, zwany **dokumentem głównym**, zawiera treść listu (szablon). Drugi to **zestaw danych**, zwykle adresowych, które w połączeniu z dokumentem głównym utworzą **dokument końcowy** (rys. 3.1). Na przykład jeśli zestaw danych zawiera 100 adresatów, dokument końcowy korespondencji może mieć 100 stron, gdzie każda strona będzie zawierała tę samą treść listu, lecz inne dane adresowe.



Rys. 3.1. Elementy składowe korespondencji seryjnej

Warto wiedzieć

Mechanizmy korespondencji seryjnej są stosowane w systemach marketingu mailowego, np. Mailchimp, GetResponse czy FreshMail.

3.2. Korespondencja seryjna w programie Word

Pracę z korespondencją seryjną rozpoczynamy zwykle od zredagowania treści listu, który chcemy wysłać. W treści warto zaplanować miejsca na: adres, powitanie, zwroty grzecznościowe i inne elementy, które chcielibyśmy uzależniać od wybranych kryteriów. Na przykład jeżeli chcemy, aby w lewym górnym rogu listu znajdował się adres odbiorcy, zostawiamy kilka linii wolnego miejsca. Nie musimy wpisywać adresu, ponieważ pobierzemy go z bazy zawierającej dane adresatów.

Przygotujemy zaproszenie na pokaz sztucznych ogni. Docelowo w zaproszeniach mają się znaleźć dane adresowe ich odbiorów. W nowym dokumencie programu Word zapisujemy główną treść wiadomości, nie podając na razie żadnych danych adresowych (rys. 3.2).

Dobra rada

List możesz formatować na każdym etapie tworzenia korespondencji seryjnej.

Warto wiedzieć

Narzędzia korespondencji seryjnej są również dostępne w oprogramowaniu biurowym LibreOffice.

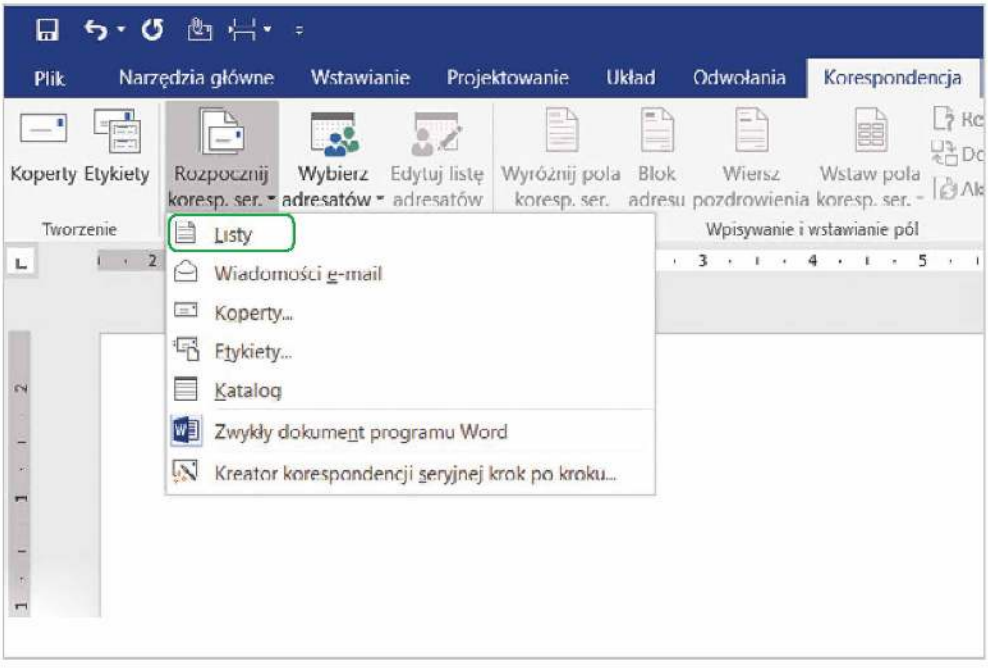
Szanowni Państwo!

Informujemy, że 28 maja br. o godzinie 22 na stadionie przy ulicy Kasztanowej odbędzie się pokaz sztucznych ogni. Serdecznie zapraszamy.

Organizatorzy

Rys. 3.2. Zaproszenie na pokaz sztucznych ogni

Musimy teraz wskazać programowi Word, że pracujemy z dokumentem korespondencji seryjnej. W tym celu wybieramy **Korespondencja → Rozpoczynanie korespondencji seryjnej → Rozpocznij koresp. ser. → Listy** (rys. 3.3). Od tego momentu dokument jest traktowany jako dokument główny korespondencji seryjnej.



Rys. 3.3. Różne rodzaje dokumentów korespondencji seryjnej

Warto wiedzieć

Zmiana typu dokumentu na **Zwykły dokument programu Word** powoduje usunięcie połączenia z listą adresatów.

Dobra rada

Każdą z operacji tworzenia listów, kopert i etykiet możesz wykonać również z poziomu tzw. kreatora. Wystarczy wybrać opcję **Kreator korespondencji seryjnej krok po kroku...** (rys. 3.3, s. 55).



Jak widać na rysunku 3.3 ze s. 55, w ramach korespondencji seryjnej możemy tworzyć nie tylko listy. Inne dostępne opcje to:

- **Wiadomości e-mail** – wybranie tej opcji powoduje zmianę widoku dokumentu na **Układ sieci Web**. Treść listu można wysłać do adresatów pocztą elektroniczną, jeżeli w podłączonej bazie adresowej jednym z pól jest adres e-mail.
- **Koperty...** – jeśli wybierzemy tę opcję, będziemy mieć możliwość edycji danych adresata i nadawcy oraz umieszczenia tych elementów na wybranym formacie koperty. Możemy dzięki temu szybko zaadresować wiele kopert.
- **Etykiety...** – ta opcja pozwala zadrukować stronę identycznymi obiektami. Kolejne elementy pobierane są z bazy danych i rozmieszczane równomiernie na stronie. W ten sposób możemy stworzyć serię naklejek z adresami, etykiety z cenami itp.
- **Katalog** – podobnie jak w przypadku etykiet opcja ta pozwala na zadrukowanie strony wybranymi danymi z kolejnych wierszy listy adresowej. Elementy rozmieszczane są na stronie kolejno, jeden pod drugim.

Zapamiętaj

Plik programu Microsoft Word należy przełączyć w odpowiedni tryb, aby stał się dokumentem głównym korespondencji seryjnej.

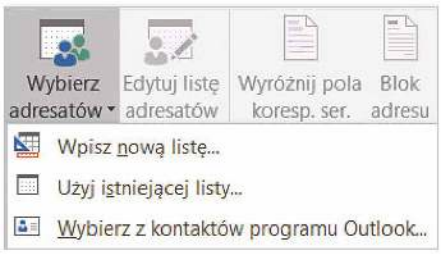
Wybór adresatów

Kiedy mamy już gotowy dokument główny korespondencji seryjnej, możemy połączyć go z listą adresatów. Wybieramy spośród kilku opcji (rys. 3.4). Możemy:

- utworzyć nową listę bezpośrednio w programie Word,
- połączyć dokument główny z istniejącą bazą (np. w pliku Excela),
- wybrać kontakty z programu Microsoft Outlook.

Dobra rada

Dane do personalizacji korespondencji seryjnej mogą być zapisane w postaci pliku tekstowego, arkusza kalkulacyjnego lub bazy danych.



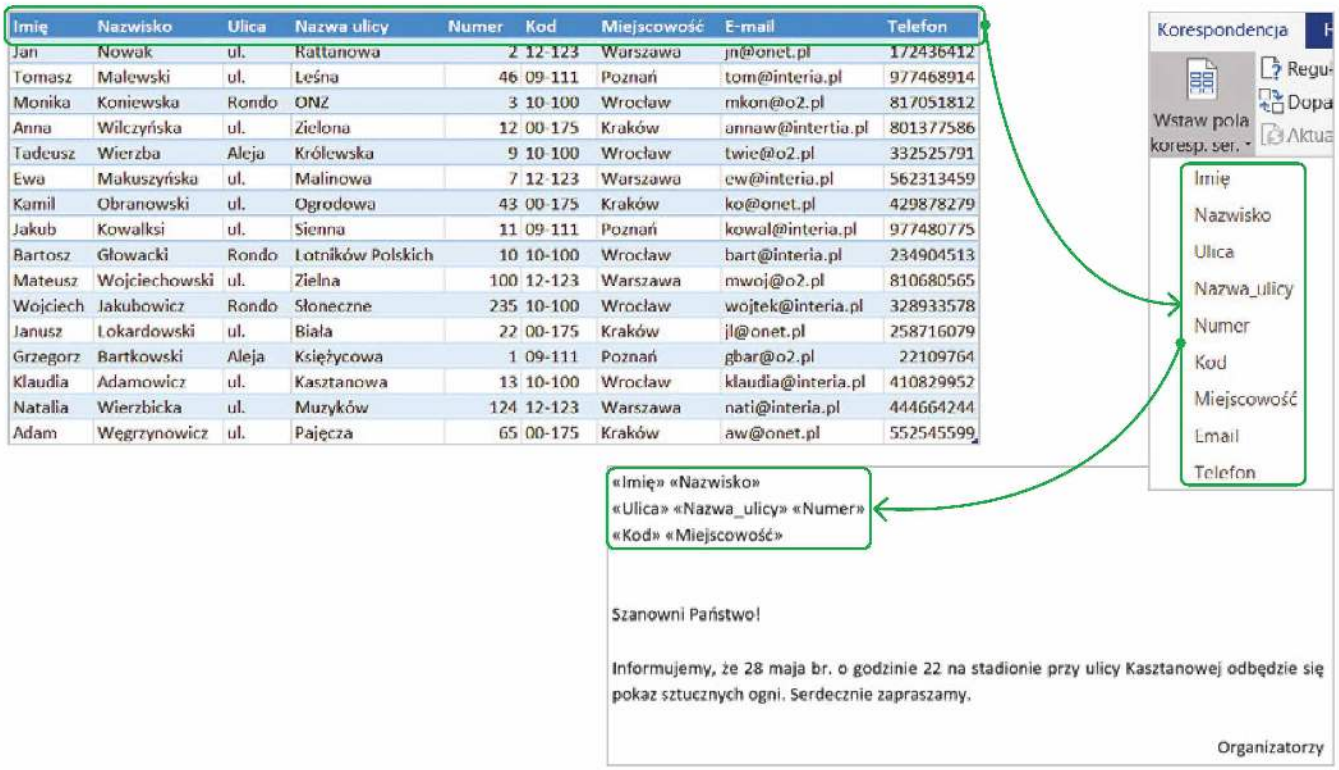
Rys. 3.4. Dodawanie adresatów

Podłączymy przykładową bazę adresów z tabeli programu Excel (np. z pliku *lista_adresatow.xlsx* przekazanego przez nauczyciela). Wybieramy **Korespondencja → Rozpoczynanie korespondencji seryjnej → Wybierz adresatów → Użyj istniejącej listy...** i wskazujemy właściwy plik. W kolejnym oknie wybieramy arkusz skoroszytu, w którym przechowywane są dane adresatów i klikamy **OK**.

Pierwszy wiersz tabeli zawiera nagłówki kolumn opisujące ich zawartość (rys. 3.5). Nazwy kolumn tabeli są reprezentowane w dokumencie w postaci tzw. **pól korespondencji seryjnej**. Możemy je wstawić do dokumentu i formatować jak zwyczajny tekst. Każdy wiersz listy adresowej dotyczy jednego adresata i będziemy go nazywali **rekordem**.

Pole korespondencji seryjnej

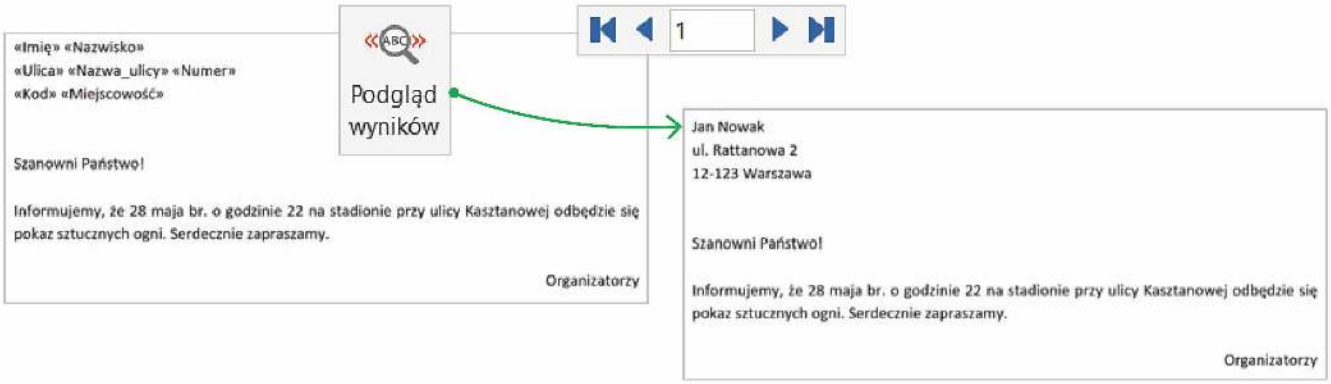
Rekord



Imię	Nazwisko	Ulica	Nazwa ulicy	Numer	Kod	Miejscowość	E-mail	Telefon
Jan	Nowak	ul.	Rattanowa	2	12-123	Warszawa	jn@onet.pl	172436412
Tomasz	Malewski	ul.	Leśna	46	09-111	Poznań	tom@interia.pl	977468914
Monika	Koniewska	Rondo	ONZ	3	10-100	Wrocław	mkon@o2.pl	817051812
Anna	Wilczyńska	ul.	Zielona	12	00-175	Kraków	annaw@intertia.pl	801377586
Tadeusz	Wierzba	Aleja	Królewska	9	10-100	Wrocław	twie@o2.pl	332525791
Ewa	Makuszyńska	ul.	Malinowa	7	12-123	Warszawa	ew@interia.pl	562313459
Kamil	Obranowski	ul.	Ogrodowa	43	00-175	Kraków	ko@onet.pl	429878279
Jakub	Kowalksi	ul.	Sienna	11	09-111	Poznań	kowal@interia.pl	977480775
Bartosz	Głowacki	Rondo	Lotników Polskich	10	10-100	Wrocław	bart@interia.pl	234904513
Mateusz	Wojciechowski	ul.	Zielna	100	12-123	Warszawa	mwoj@o2.pl	810680565
Wojciech	Jakubowicz	Rondo	Słoneczne	235	10-100	Wrocław	wojtek@interia.pl	328933578
Janusz	Lokardowski	ul.	Biała	22	00-175	Kraków	jl@onet.pl	258716079
Grzegorz	Bartkowski	Aleja	Księżycowa	1	09-111	Poznań	gbar@o2.pl	22109764
Klaudia	Adamowicz	ul.	Kasztanowa	13	10-100	Wrocław	klaudia@interia.pl	410829952
Natalia	Wierzbicka	ul.	Muzyków	124	12-123	Warszawa	nati@interia.pl	444664244
Adam	Węgrzynowicz	ul.	Pajęcza	65	00-175	Kraków	aw@onet.pl	552545599

Rys. 3.5. Lista adresowa i odpowiadające jej pola korespondencji seryjnej

Aby zamiast pól korespondencji seryjnej zobaczyć konkretne wartości pobrane z listy adresatów, wystarczy włączyć podgląd wyników: **Korespondencja → Podgląd wyników → Podgląd wyników**. Kolejne adresy można przeglądać dzięki przyciskom nawigacyjnym (rys. 3.6).



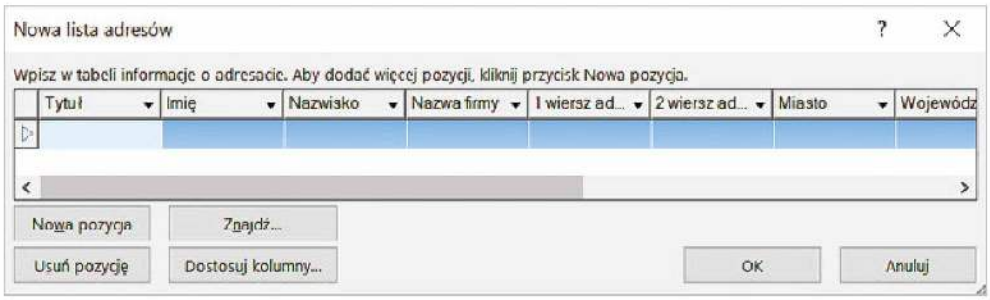
Rys. 3.6. Podgląd wyników

Ręczne wprowadzanie adresatów

Dane adresowe można również uzupełnić samodzielnie na etapie tworzenia korespondencji seryjnej, bez odwoływania się do plików zewnętrznych. W tym celu należy wybrać **Korespondencja → Rozpoczynanie korespondencji seryjnej → Wybierz adresatów → Wpisz nową listę...** Okno **Nowa lista adresów**, które się pojawi, zawiera szereg kolumn i pól (rys. 3.7, s. 58). Należy je uzupełnić danymi adresowymi. Jeden wiersz odpowiada danym jednej osoby. Kolejne adresy dodajemy, klikając przycisk **Nowa pozycja**.

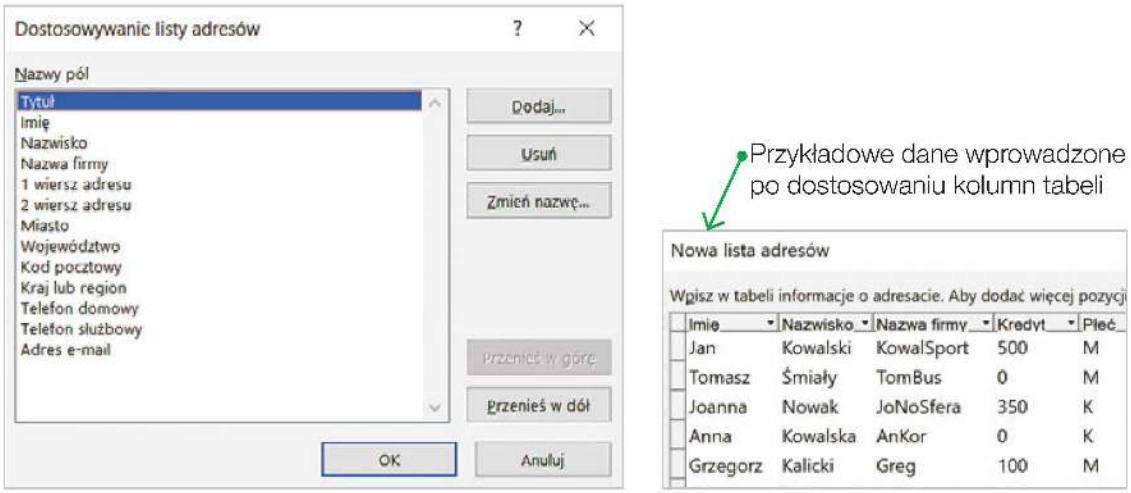
Dobra rada

Jeśli adresatów jest wielu, możesz ich wyszukiwać dzięki opcji **Znajdź...**



Rys. 3.7. Wstawianie nowej listy adresatów

Liczbę kolumn oraz ich nazwy możemy dostosować do naszych potrzeb. Przycisk **Dostosuj kolumny...** uruchamia okno, które pozwala wprowadzać zmiany w polach korespondencji. Pola możemy dodawać, usuwać, zmieniać ich nazwy oraz kolejność (rys. 3.8).

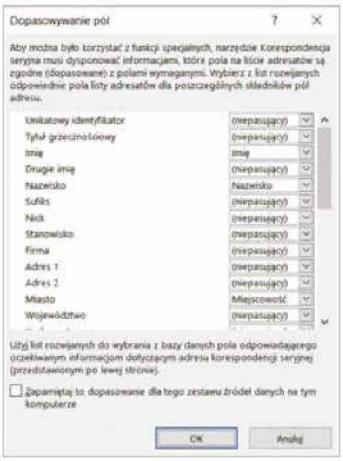


Rys. 3.8. Własny zestaw kolumn i przykładowe dane adresatów

Ćwiczenie 1

Zmodyfikuj domyślną listę pól korespondencji seryjnej do postaci pokazanej w tabeli na rysunku 3.8 i uzupełnij pola danymi (5 wierszy).

Kiedy tworzymy lub modyfikujemy pola korespondencji seryjnej, program Word będzie próbował dopasować je do gotowych, predefiniowanych nazw. Ich listę zobaczymy po wybraniu **Korespondencja → Wpisywanie i wstawianie pól → Dopasuj pola** (rys. 3.9). Te nazwy program stosuje, gdy korzystamy z narzędzi **Blok adresu** lub **Wiersz pozdrowienia** (**Korespondencja → Wpisywanie i wstawianie pól**). Tylko jeśli właściwie przypiszemy pola z listy adresowej do predefiniowanych nazw, program dobrze wstawi dane adresowe w gotowych blokach.



Rys. 3.9. Przypisywanie pól do predefiniowanej listy

3.3. Filtrowanie adresatów i personalizacja listów

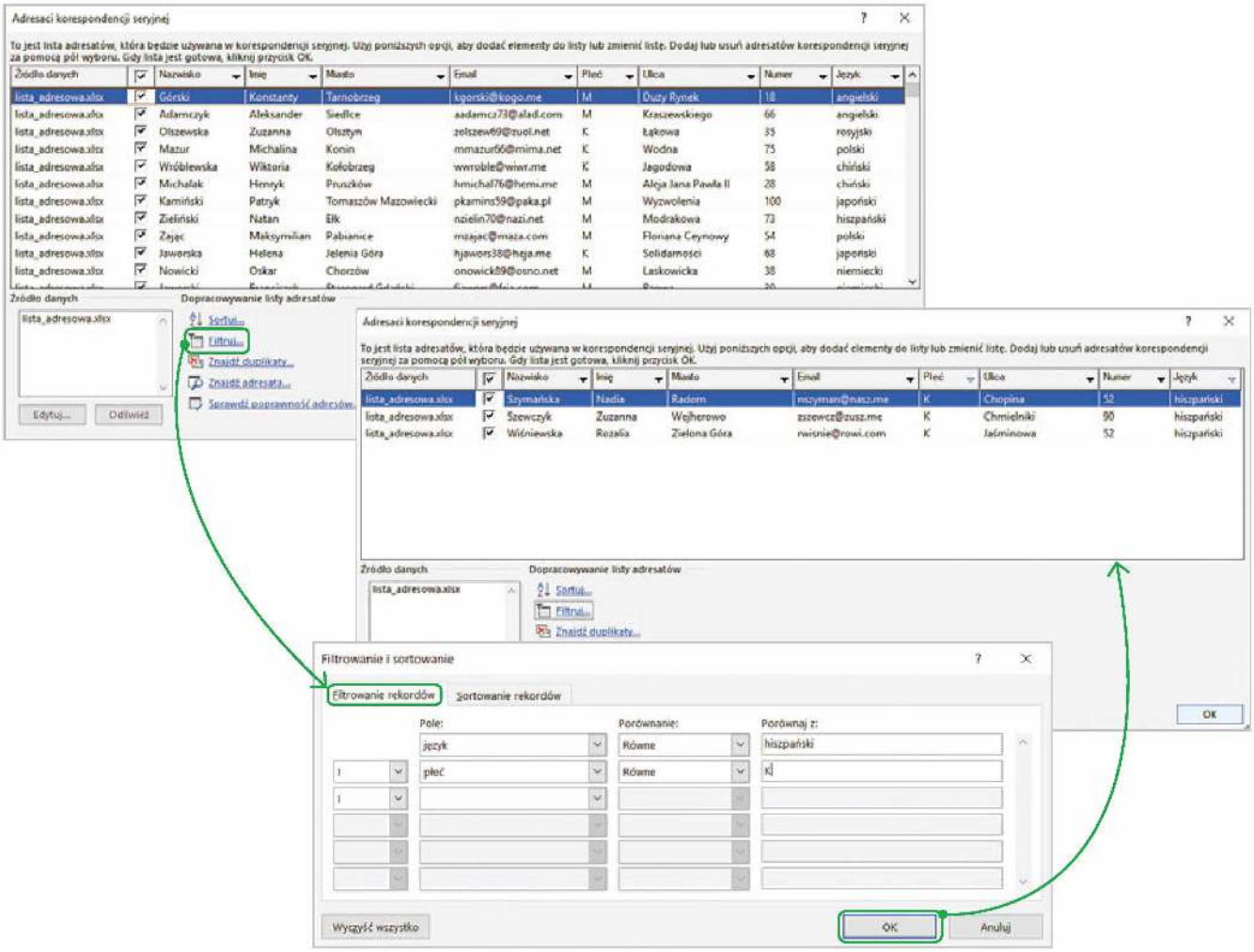
Do tej pory wykorzystaliśmy korespondencję seryjną do tego, aby dodać do gotowego listu dane wszystkich wprowadzonych przez nas adresatów. Pokażemy teraz, jak przygotować ją tylko dla wybranych adresatów oraz jak dostosować treść listu do jego odbiorcy.

Filtrowanie adresatów

Jeśli nasza baza adresów jest bardzo duża, a korespondencję chcemy wysłać jedynie do wybranych osób, możemy użyć **filtrowania**, aby wyselekcjonować właściwych adresatów. W nowym dokumencie programu Word podłączymy listę adresatów z arkusza kalkulacyjnego (**Korespondencja → Rozpoczynanie korespondencji seryjnej → Wybierz adresatów → Użyj istniejącej listy...**).

Filtrowanie, s. 30

Załóżmy, że zarządzamy grupą kilkuset profesjonalnych tłumaczy specjalizujących się w różnych językach obcych. Chcielibyśmy wysłać korespondencję jedynie do kobiet znających język hiszpański. W tym celu wybieramy **Korespondencja → Rozpoczynanie korespondencji seryjnej → Edytuj listę adresatów**, a w oknie, które się pojawi, wybieramy **Filtruj...** (rys. 3.10).



Rys. 3.10. Filtrowanie listy adresatów

Filtrowanie, w którym jako kryteria przyjęliśmy język hiszpański oraz płeć żeńską, ograniczyło listę adresatów do trzech pozycji.

Ćwiczenie 2

Plik, który otrzymasz od nauczyciela (np. *firmy.csv*), zawiera dane adresowe 1000 firm. Użyj filtrowania, aby na liście adresatów korespondencji seryjnej wyświetlić tylko firmy z województwa małopolskiego, które nie mają zobowiązań kredytowych.

Personalizacja treści listu

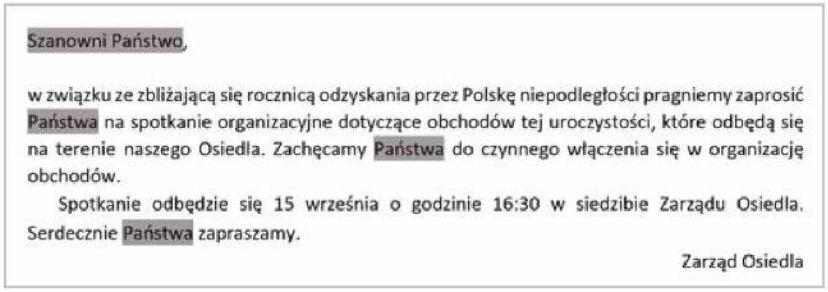
W programie Word można personalizować fragmenty listu na podstawie zastosowanych reguł. Możemy np. umieścić w liście zwroty grzecznościowe dostosowane do płci adresata. W zależności od kraju czy województwa, w którym mieszka adresat, możemy użyć właściwego numeru kierunkowego. Możemy też pomijać informacje zawarte w polach korespondencji, jeśli nie spełniają wybranych kryteriów. Treść listu, choć tworzona w jednym dokumencie dla wielu adresatów, może zatem docelowo być zupełnie inna dla każdego odbiorcy.

Dobra rada

Większość polskich imion żeńskich kończy się literą „a". Tę zależność możesz wykorzystać do określania płci adresata na podstawie imienia.

Stworzymy teraz proste zaproszenie, w którym uzależnimy zwroty grzecznościowe od płci adresata. Aby to było możliwe, dane adresowe muszą zawierać informację, czy odbiorca jest kobietą czy mężczyzną. Wystarczy, że dodamy w pliku z adresami kolumnę o nazwie `Płeć` i wypełnimy ją, przyjmując jednoznaczny opis, np. `M` dla mężczyzn, `K` dla kobiet.

Otwieramy nowy dokument i wybieramy **Korespondencja → Rozpoczynanie korespondencji seryjnej → Rozpocznij koresp. ser. → Listy**. Przykładową treść listu przedstawiono na rysunku 3.11.



Szanowni Państwo,

w związku ze zbliżającą się rocznicą odzyskania przez Polskę niepodległości pragniemy zaprosić Państwa na spotkanie organizacyjne dotyczące obchodów tej uroczystości, które odbędą się na terenie naszego Osiedla. Zachęcamy Państwa do czynnego włączenia się w organizację obchodów.

Spotkanie odbędzie się 15 września o godzinie 16:30 w siedzibie Zarządu Osiedla. Serdecznie Państwa zapraszamy.

Zarząd Osiedla

Rys. 3.11. Zaproszenie skierowane do mieszkańców osiedla

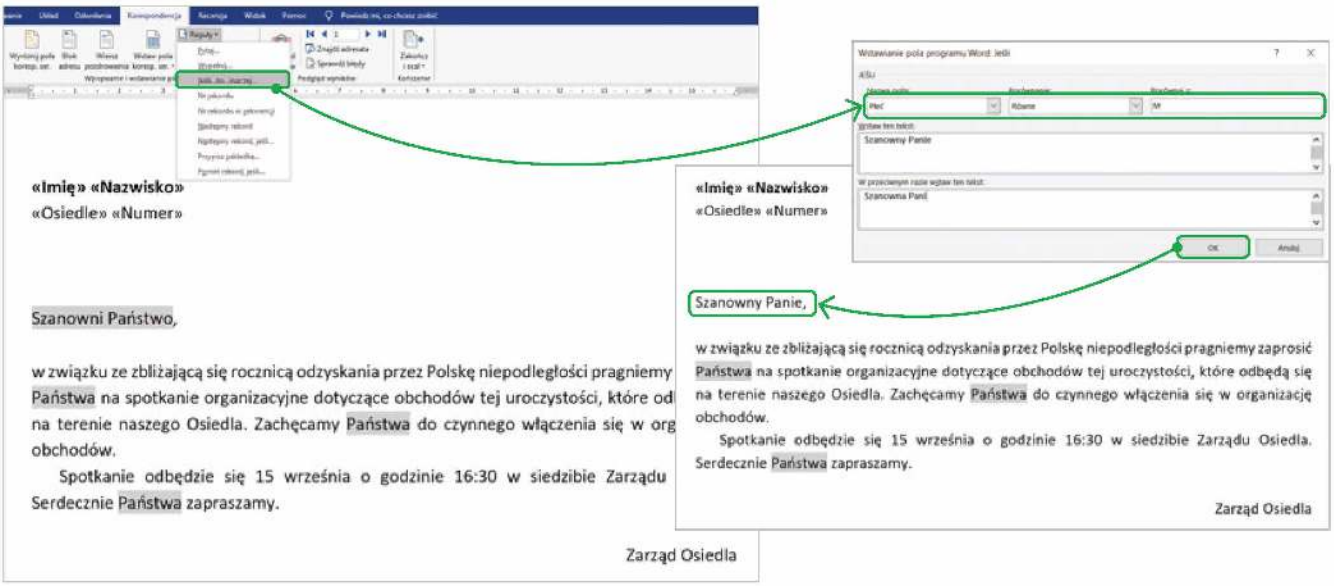
Chcielibyśmy, aby list był kierowany indywidualnie do poszczególnych mieszkańców. Dlatego wszystkie zwroty ogólne (oznaczone w treści listu szarym kolorem) zamienimy w zależności od tego, czy adresatem jest kobieta czy mężczyzna.

Do szablonu wiadomości podłączymy adresatów z pliku Excela, w którym znajdują się dane pogrupowane w następujące kolumny: `Imię`, `Nazwisko`, `Osiedle`, `Numer` oraz `Płeć`. Na początku dodamy pola korespondencji seryjnej z danymi adresowymi.

Aby to zrobić, ustawiamy kursor w odpowiednim miejscu dokumentu (np. w lewym górnym rogu) i wskazujemy potrzebne pola (**Korespondencja → Wpisywanie i wstawianie pól → Wstaw pola koresp. ser.**). Następnie w treści listu zaznaczamy (lub usuwamy) zwrot „Szanowni Państwo" i wybieramy **Korespondencja → Wpisywanie i wstawianie pól → Reguły → Jeśli...to...inaczej...** (rys. 3.12).

Dobra rada

Po zastosowaniu reguły warunkowej sprawdź, czy nie zmieniło się formatowanie wstawionego tekstu.



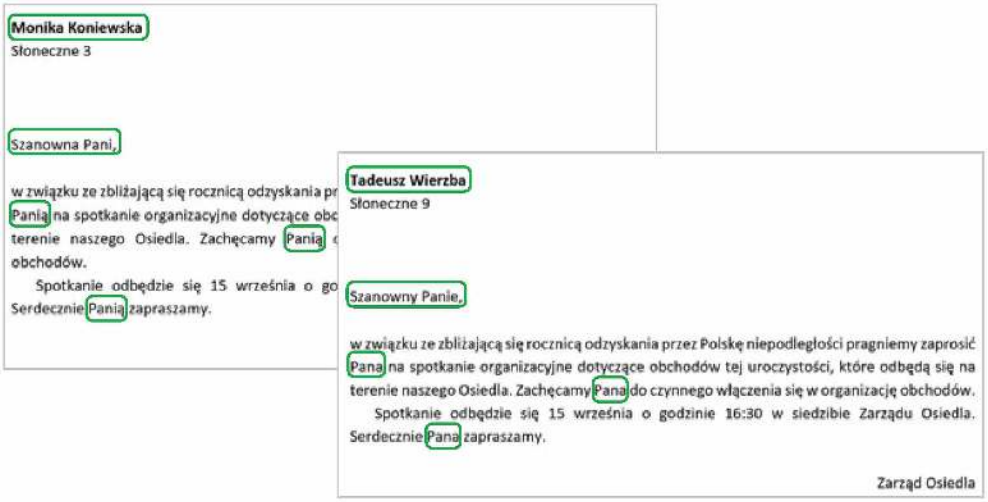
Rys. 3.12. Reguła, która zmienia zwrot grzecznościowy w zależności od płci adresata

Jest to reguła, która pozwala wstawiać do dokumentu tekst w zależności od tego, czy spełniony jest warunek określony w górnej części tego okna. Warunkiem, który wybraliśmy, jest porównanie (Równe) wartości z pola Płeć bieżącego rekordu z tekstem: M. Wtedy jeśli adresat jest mężczyzną, program wstawi tekst „Szanowny Panie", w przeciwnym razie – „Szanowna Pani".

Użyjemy tej reguły dla pozostałych zwrotów oznaczonych szarym kolorem na rysunku 3.12. Efektem będzie personalizacja treści, którą zobaczymy po włączeniu podglądu wyników (rys. 3.13).

Warto wiedzieć

W rozbudowanych tabelach informację o płci oznacza się często literami K i M (lub z ang. M i F). Literami T i N (lub z ang. Y i N) oznacza się zazwyczaj wartości tak i nie. Można używać również liczb 1 i 0, gdzie 1 reprezentuje tak, a 0 – nie.



Rys. 3.13. Spersonalizowana korespondencja

Ćwiczenie 3

Na podstawie opisanego przykładu utwórz serię zaproszeń na imprezę urodzinową, skierowanych do swoich znajomych. W treści listu użyj zwrotów, które spersonalizujesz w zależności od tego, czy piszesz do kolegi czy koleżanki.

3.4. Zarządzanie danymi korespondencji seryjnej w arkuszu kalkulacyjnym

Dobra rada

Jeżeli jakiś element korespondencji seryjnej zależy od wielu warunków, możesz przygotować bazę adresową w arkuszu kalkulacyjnym, stosując zagnieżdżone instrukcje warunkowe.

Dobra rada

Jeśli w korespondencji seryjnej wykorzystujesz dane z arkusza kalkulacyjnego, które są wynikiem działania formuł na liczbach, ustaw typ danych źródłowych na tekstowy. Dzięki temu unikniesz przetwarzania w edytorze tekstu m.in. zapisu rozwinięć dziesiętnych ułamków nieskończonych.

Arkusz kalkulacyjny możemy wykorzystać jako narzędzie do przygotowywania bazy z danymi do korespondencji seryjnej. Mogą to być dane adresowe lub jakiekolwiek inne. Jest to dobra metoda w przypadku dużej liczby danych, których wstępne filtrowanie wymaga narzędzi, jakich nie posiadają mechanizmy korespondencji seryjnej edytora tekstu. Używając formuł arkusza kalkulacyjnego, możemy nie tylko filtrować dane (np. adresowe), lecz także wykonywać na nich różnego rodzaju operacje.

Rysunek 3.14 prezentuje fragment przykładowej tabeli zawierającej informacje o liczbie punktów uzyskanych przez poszczególnych uczestników konkursu. Przyjmijmy, że chcemy przygotować wiadomość o wynikach konkursu, którą wyślemy jedynie do laureatów, tj. dziesięciu uczestników z najwyższą liczbą punktów.

	A	B	C	D	E
1	ID	Imię	Nazwisko	Data zgłoszenia	Liczba punktów
2	U1	Stanisław	Radzik	21-05-2019	130
3	U2	Ida	Bujak	07-07-2019	116
4	U3	Mikołaj	Guzek	17-04-2019	131
5	U4	Michał	Rajkowski	25-04-2019	114
6	U5	Michał	Marciniuk	26-06-2019	155
7	U6	Grzegorz	Łapa	28-05-2019	112
8	U7	Katarzyna	Dobkowska	06-05-2019	141
9	U8	Wiktor	Gomółka	16-04-2019	106
10	U9	Piotr	Gołuch	19-05-2019	126
11	U10	Jakub	Turzyński	29-06-2019	139
12	U11	Jakub	Drozdek	08-06-2019	149
13	U12	Stefan	Gawenda	28-06-2019	115
14	U13	Grzegorz	Ślusarski	10-05-2019	146
15	U14	Jacek	Wojciechowski	30-04-2019	143
16	U15	Karolina	Łukaszewicz	15-07-2019	102
17	U16	Maria	Matuszczyk	09-07-2019	113
18	U17	Marta	Gibas	15-07-2019	132
19	U18	Wiktor	Tomala	06-05-2019	170
20	U19	Katarzyna	Strugała	28-06-2019	118

Rys. 3.14. Uczestnicy konkursu

Mechanizmy korespondencji seryjnej nie pozwolą w prosty sposób rozwiązać tego zadania. Jednak za pomocą arkusza kalkulacyjnego można łatwo oznaczyć w tabeli laureatów.

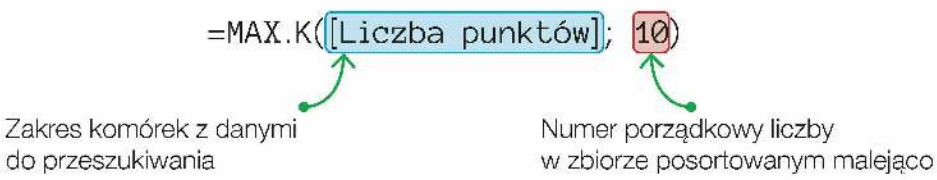
W tym celu dodamy do tabeli nową kolumnę, np. `Laureat`. W pierwszej komórce tej kolumny wstawimy formułę, która wypisze słowo „tak", jeżeli liczba punktów uczestnika należy do zbioru 10 najlepszych.

Skorzystamy tu z **funkcji** `MAX.K`, która dla określonej liczby *k* zwraca *k*-tą największą wartość w wybranym zbiorze danych. Na przykład dla *k* = 2 funkcja zwróci dwie najwyższe wartości ze zbioru. Na rysunku 3.15 przedstawiono zastosowanie funkcji `MAX.K` do wyłonienia dziesięciu największych wartości w kolumnie `Liczba punktów`.

Funkcja `MAX.K`

Warto wiedzieć

Funkcja `MAX.K` może zwrócić większą liczbę rekordów niż *k*, jeśli znajdą się pola o takich samych wartościach.



Rys. 3.15. Formuła z użyciem funkcji `MAX.K`

Kompletna formuła ma postać:

```
=JEŻELI([@[Liczba punktów]] >=
MAX.K([Liczba punktów]; 10); "tak"; "")
```

Zwróć uwagę, że w formule zastosowano zapis z nawiasami kwadratowymi oraz symbolem @. Pojawiają się one podczas pracy w obrębie zdefiniowanej tabeli programu Excel. Zapis `[Liczba punktów]` oznacza pełen zakres kolumny `Liczba punktów`, a zapis `[@[Liczba punktów]]` oznacza odwołanie do aktualnego wiersza kolumny `Liczba punktów`. Są to tak zwane **odwołania strukturalne**.

Odwołania strukturalne

Kopiujemy formułę z komórki `F2` do pozostałych komórek w tej kolumnie tabeli (rys. 3.16).

Warto wiedzieć

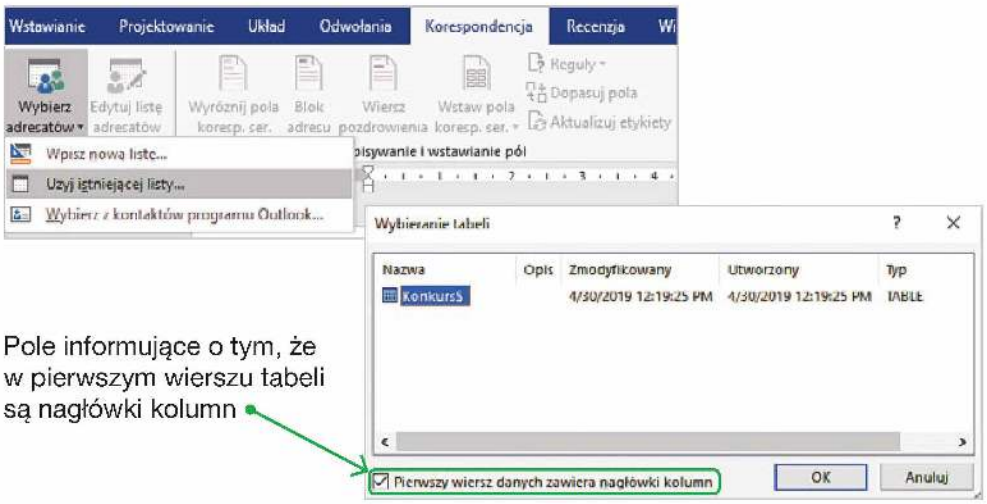
Odwołanie strukturalne do komórki tabeli o nagłówku jednowyrazowym, np. `Miasto`, to: `[@Miasto]`. Jeśli nagłówek składa się z wielu wyrazów, np. `Miasto partnerskie`, to ma on budowę: `[@[Miasto partnerskie]]`.

	A	B	C	D	E	F
1	ID	Imię	Nazwisko	Data zgłoszenia	Liczba punktów	Laureat
2	U1	Stanisław	Radzik	21-05-2019	130	
3	U2	Ida	Bujak	07-07-2019	116	
4	U3	Mikołaj	Guzek	17-04-2019	131	
5	U4	Michał	Rajkowski	25-04-2019	114	
6	U5	Michał	Marciniuk	26-06-2019	155	
7	U6	Grzegorz	Łapa	28-05-2019	112	

	A	B	C	D	E	F
1	ID	Imię	Nazwis	Data zgłoszenia	Liczba p	Laureat
6	U18	Wiktor	Tomala	06-05-2019	170	tak
8	U20	Piotr	Lipski	23-05-2019	156	tak
11	U5	Michał	Marciniuk	26-06-2019	155	tak
12	U11	Jakub	Drozdek	08-06-2019	149	tak
14	U29	Piotr	Kuriata	08-06-2019	148	tak
15	U13	Grzegorz	Ślusarski	10-05-2019	146	tak
19	U24	Jacek	Zarębski	19-04-2019	144	tak
21	U14	Jacek	Wojciecho	30-04-2019	143	tak
25	U7	Katarzyna	Dobkowsk	06-05-2019	141	tak
30	U10	Jakub	Turzyński	29-06-2019	139	tak

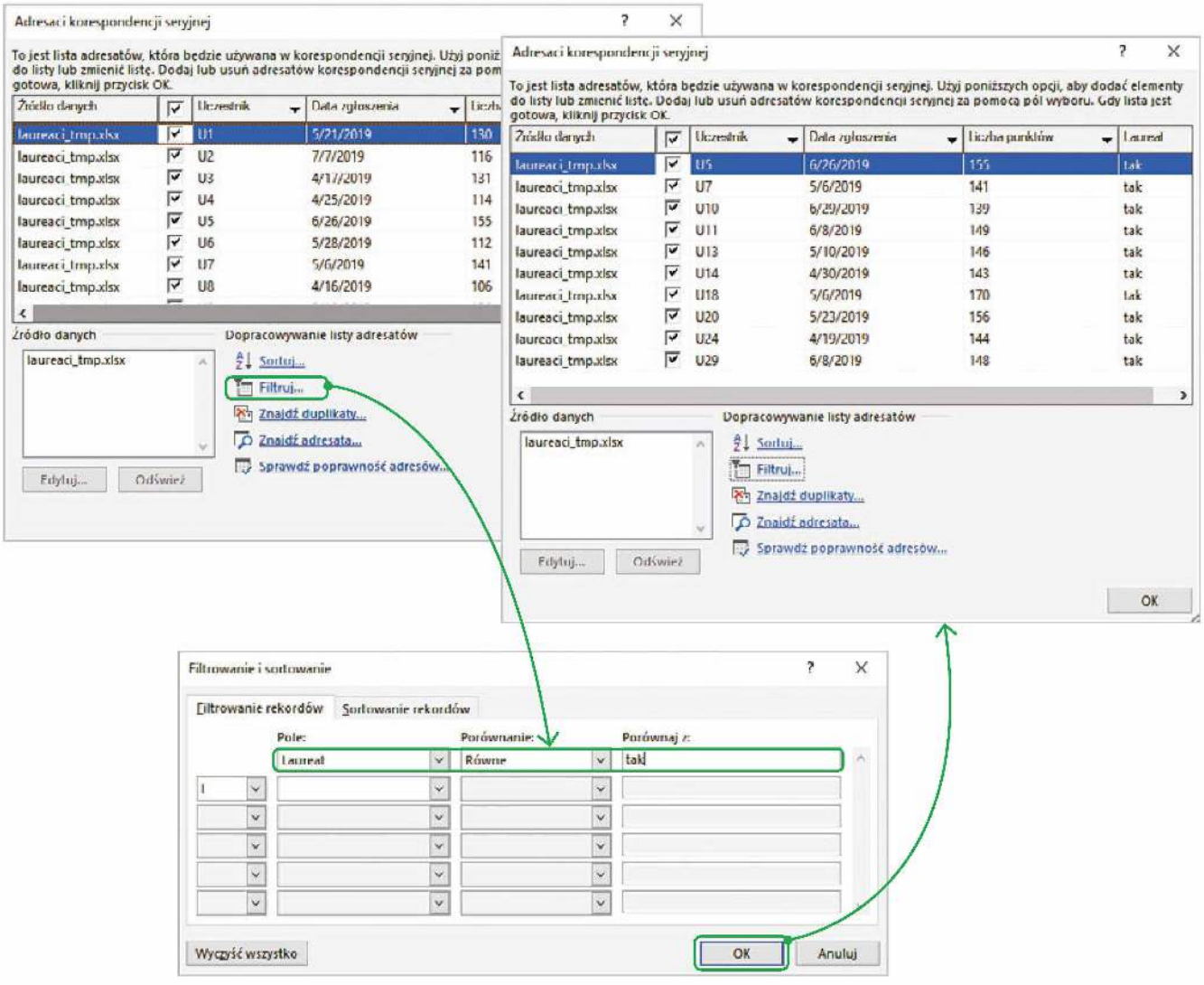
Rys. 3.16. Wytypowanie laureatów konkursu

Aby umieścić dane z tabeli w korespondencji seryjnej, w zakładce **Korespondencja** programu Word za pomocą opcji **Wybierz adresatów → Użyj istniejącej listy...** wskazujemy przygotowany plik Excela. Ponieważ w pierwszym wierszu tabeli znajdują się tytuły dla poszczególnych kolumn, upewniamy się, że włączona jest opcja **Pierwszy wiersz danych zawiera nagłówki kolumn** (rys. 3.17, s. 64).



Rys. 3.17. Importowanie listy adresowej z tabeli Excela

W naszym przykładzie musimy jeszcze przefiltrować listę adresatów, aby zawęzić ją tylko do laureatów konkursu. Wybieramy przycisk **Edytuj listę adresatów** i filtrujemy ją, dodając regułę, w której pole `Laureat` będzie równe `tak` (rys. 3.18).



Rys. 3.18. Filtrowanie listy uczestników

Po wybraniu arkusza, w którym znajduje się tabela, jej nagłówki odpowiadają polom korespondencji seryjnej. Wystarczy teraz przygotować szablon wiadomości i wstawić odpowiednie pola w pliku głównym korespondencji seryjnej.

Ćwiczenie 4

W Polsce platynowe płyty są przyznawane artystom jazzowym, których albumy osiągnęły w danym roku sprzedaż przynajmniej 10 000 egzemplarzy. Na podstawie plików przekazanych przez nauczyciela (np. *sprzedaz_plyt.xlsx* oraz *gratulacje.docx*) przygotuj listy gratulacyjne dla wszystkich wyróżnionych zespołów.

Warto wiedzieć

Oprogramowanie bazodanowe CRM (ang. *customer relatioship management*) do zarządzania relacjami z klientem zwykle umożliwia wysyłkę wiadomości e-mail.

Zapamiętaj

Zaawansowaną personalizację listów w korespondencji seryjnej można przygotować, korzystając z zewnętrznego arkusza kalkulacyjnego. Pozwala to zastosować bardziej złożone warunki filtrowania adresatów oraz dostosować treść listu.

3.5. Kończenie pracy z korespondencją seryjną

Ostatnim etapem pracy z korespondencją seryjną jest wygenerowanie dokumentu końcowego zgodnie z szablonem i zastosowanymi regułami. Proces ten nazywa się **scalaniem** i uruchamiamy go, wybierając **Zakończ i scal** na karcie **Korespondencja**.

Scalanie dokumentów korespondencji seryjnej

Dostępne są trzy opcje scalania dokumentów korespondencji seryjnej:

- **Edytuj poszczególne dokumenty...** powoduje utworzenie nowego pliku – jest to zwykły dokument, w którym pola korespondencji seryjnej zostały zastąpione odpowiednimi wartościami z listy adresowej, a samo połączenie zostało utracone. Jeśli tworzymy jeden dokument końcowy, jego objętość będzie wielokrotnością liczby stron dokumentu głównego. Na przykład gdy dwustronicowy list wysyłamy do 15 adresatów, dokument końcowy będzie liczył 30 stron. W tak utworzonym dokumencie końcowym możemy wprowadzić dowolne zmiany w każdym liście.
- **Drukuj dokumenty...** przesyła listy wprost na drukarkę. Możemy zdecydować, czy drukujemy wszystkie listy czy tylko wybrane.
- **Wyślij wiadomości e-mail...** to możliwość wysłania listów za pomocą poczty elektronicznej. W tym przypadku zostaniemy poproszeni o wskazanie pola, w którym znajdują się adresy e-mail (**Do**), o podanie tematu (**Wiersz tematu**) i określenie **Formatu wiadomości** (może to być **HTML**, **Załącznik** lub **Zwykły tekst**). Możemy ograniczyć liczbę adresatów i wysłać wiadomość tylko do bieżącego adresata lub do wybranych.

Warto wiedzieć

Niektóre programy pocztowe mogą mieć wyłączoną lub zablokowaną możliwość wyświetlania wiadomości w formacie HTML.

Podsumowanie

- Korespondencja seryjna ułatwia tworzenie spersonalizowanych wiadomości skierowanych do wielu odbiorców.
- Aby przygotować korespondencję seryjną, potrzebne są dwa pliki: szablon wiadomości oraz lista adresatów. Dokument edytora tekstu należy przełączyć w odpowiedni tryb, aby stał się szablonem korespondencji seryjnej.
- Listy adresowe mogą być importowane z zewnętrznych plików lub tworzone bezpośrednio w edytorze tekstu. Zaimportowane listy można filtrować.
- Arkusz kalkulacyjny umożliwia przygotowanie bazy adresowej do korespondencji seryjnej. Zastosowanie formuł zwiększa możliwości personalizacji korespondencji.

Zadania

* **1** Za pomocą korespondencji seryjnej utwórz w programie Word plik z 15 adresami. Dla każdego adresata uzupełnij następujące pola: `Imię`, `Nazwisko`, `Ulica`, `Numer`, `Kod pocztowy` oraz `Miasto`, `Płeć`, `Wiek` i `Zainteresowania`.

* **2** Utwórz w programie Excel bazę adresową, która będzie zawierała minimum 10 rekordów z następującymi polami: `Login`, `Data`, `Adres IP`. Przyjmij, że adres IP to adres zgodny z protokołem IPv4.

* **3** Napisz list z informacją o spotkaniu miłośników kolei. Plik zapisz jako list korespondencji seryjnej.

* **4** Do pliku tekstowego (np. *ulotka.docx*) podłącz bazę adresatów utworzoną w arkuszu kalkulacyjnym (np. z pliku *podroznicy.xlsx*). Oba pliki otrzymasz od nauczyciela. Przefiltruj adresy tak, aby otrzymać grupę osób, które interesują się podróżowaniem.

** **5** W dostępnych źródłach sprawdź, w jaki sposób w edytorze tekstu można seryjnie adresować koperty. Następnie zaadresuj koperty o standardowym (pocztówkowym) rozmiarze, korzystając z listy adresowej, którą otrzymasz od nauczyciela (np. z pliku *adresaci.xlsx*).

** **6** Napisz wiadomość e-mail, która zostanie rozesłana tylko do mieszkańców Krakowa. Zastosuj filtrowanie listy adresów z pliku, który przekaże ci nauczyciel (np. *mieszkancy.xlsx*).

** **7** Zastosuj korespondencję seryjną, aby wygenerować zaproszenia na wybraną przez ciebie konferencję. Zaproszenia mają trafić tylko do tych osób, które uiściły opłatę rejestracyjną. Przygotuj dokument z treścią listu oraz plik z danymi uczestników.

*** **8** Zaprojektuj w programie graficznym i wygeneruj z wykorzystaniem korespondencji seryjnej naklejki na plecaki na szkolną wycieczkę. Informacja na etykiecie powinna zawierać dane szkoły oraz dane kontaktowe właściciela plecaka.

4. Relacyjne bazy danych

Portale społecznościowe i serwisy internetowe wykorzystują olbrzymie bazy danych, aby przygotowywać informacje jak najlepiej dopasowane do użytkownika. Jeśli stu twoich znajomych w serwisie społecznościowym udostępni w tym samym czasie jakieś informacje, trzeba je gdzieś przechować i skatalogować, a także zdecydować, które z nich ci wyświetlić i w jakiej kolejności. Bez bazy danych i zaawansowanych algorytmów byłoby to niemal niemożliwe. W profesjonalnych zastosowaniach baz danych dominuje tzw. model relacyjny – w tym temacie dowiesz się, na czym on polega, i stworzysz własną relacyjną bazę danych.

Cele lekcji

- Zrozumiesz, czym jest baza danych, i poznasz związane z nią pojęcia.
- Poznasz praktyczne zastosowanie relacyjnych baz danych.
- Zaprojektujesz prostą bazę danych.
- Nauczysz się zarządzać bazą danych w programie Microsoft Access.
- Dowiesz się, jak przechowywać informacje w bazie danych oraz jak je modyfikować.
- Nauczysz się tworzyć tabele i relacje między nimi, kwerendy, formularze oraz raporty w programie Microsoft Access.

4.1. Projektowanie bazy danych

Bazą danych będziemy nazywać zbiór uporządkowanych i powiązanych ze sobą informacji dotyczących pewnej dziedziny życia, zorganizowanych w taki sposób, aby łatwo można było je przeszukiwać, wyciągać z nich wnioski i na tej podstawie podejmować decyzje. ◀ Baza danych

Podczas projektowania bazy danych przede wszystkim należy się zastanowić, do czego będzie wykorzystana, czyli w jakim celu powstaje.

Załóżmy, że wraz z przyjaciółmi prowadzisz w formie bloga serwis internetowy o nowoczesnych technologiach. Aby utrzymać stałych czytelników oraz pozyskać nowych, powinniście regularnie publikować ciekawe i dobrze opracowane artykuły.

Utworzenie bazy danych w tym wypadku pozwoliłoby:

- gromadzić dane dotyczące opublikowanych i planowanych artykułów,
- kontrolować terminowe przygotowanie artykułów,
- przechowywać dane kontaktowe autorów,
- sprawdzać, czy długości artykułów zgadzają się z ustaleniami.

Aby to wszystko było możliwe, trzeba się zastanowić, które informacje należałoby zebrać w bazie danych. Mogą to być m.in.: tytuł, liczba stron i termin dostarczenia artykułu, nazwa działu, w którym ukaże się artykuł, imię i nazwisko autora oraz kontakt do niego, a także informacja, czy tekst został dostarczony.

Warto wiedzieć

Tematyka serwisu internetowego i związanej z nim bazy danych nie ma większego znaczenia. Projekt bazy po niewielkich modyfikacjach może również posłużyć do prowadzenia np. gazetki szkolnej, bloga modowego lub profilu szkoły w mediach społecznościowych.

Tabela, w której gromadzilibyśmy wymienione dane, mogłaby wyglądać tak jak na rysunku 4.1. Zostały w niej umieszczone informacje, które wydają się niezbędne, aby skutecznie zarządzać pracami nad tekstami na blog.

Lp.	Tytuł	Strony	Termin	Dział	Dostarczono	Imię	Nazwisko	Telefon	Email
1	Sztuczna inteligencja	6	11.10.2020	Nowe technologie	Tak	Jan	Kowalski	991123123	jkow@mail.com
2	Instagram jako narzędzie osobistej marki	4	13.10.2020	Samouczki	Nie	Eliza	Makulska	991231231	emaku@mail.com
3	Autonomiczne pojazdy – analiza SWOT	3	12.10.2020	Rynek	Nie	Jan	Kowalski	991123122	jkow@mail.com

Rys. 4.1. Tabela z danymi dotyczącymi bloga

Warto wiedzieć

Ten sam tekst napisany na komputerze może mieć różną liczbę stron w zależności od szerokości marginesów, wybranej czcionki, jej wielkości itp. Dlatego do rozliczeń wykorzystuje się tzw. znormalizowaną stronę maszynopisu – zwyczajowo zawiera ona 1800 znaków (łącznie ze spacjami i znakami interpunkcyjnymi). W większości edytorów tekstu (np. Wordzie) można sprawdzić liczbę znaków w napisanym tekście.

Uzupełniając tak zaprojektowaną tabelę, możemy jednak zauważyć, że w rzeczywistości przechowujemy w niej informacje o dwóch niezależnych obiektach: artykułach oraz autorach. Łączenie informacji o nich w taki sposób, jak w tabeli na rysunku 4.1, stwarza kilka potencjalnych problemów.

Problem 1: Dodawanie kolejnych artykułów tej samej osoby wymaga wprowadzania za każdym razem tych samych informacji. Zajmuje to czas oraz miejsce, a poza tym łatwo popełnić błąd (np. w numerze telefonu Jana Kowalskiego, jak na rysunku 4.1).

Problem 2: Jeśli wprowadzimy błędne dane dotyczące tego samego autora, trudno potem rozstrzygnąć, które są właściwe.

Problem 3: Jeśli autor zmieni swoje dane (np. adres e-mail), trzeba będzie tę zmianę wprowadzić przy wszystkich jego artykułach.

Projektując bazę danych, powinniśmy więc unikać możliwości powielania informacji o tym samym obiekcie. Dlatego w naszej sytuacji należy oddzielić dane dotyczące autorów od tych dotyczących artykułów. Można w tym celu utworzyć dwie tabele, które są ze sobą powiązane, np. takie jak na rysunkach 4.2 i 4.3.

ID	Tytuł	Strony	Termin	Dział	Dostarczono	Autor
1	Sztuczna inteligencja	6	11.10.2020	Nowe technologie	Tak	1
2	Instagram jako narzędzie osobistej marki	4	13.10.2020	Samouczki	Nie	2
3	Autonomiczne pojazdy – analiza SWOT	3	12.10.2020	Rynek	Nie	1

Rys. 4.2. Tabela `Artykuły`

ID	Imię	Nazwisko	Telefon	Email
1	Jan	Kowalski	991123123	jkow@mail.com
2	Eliza	Makulska	991231231	emaku@mail.com

Rys. 4.3. Tabela `Autorzy`

Tabela z rysunku 4.1 została więc podzielona na dwie tabele o nazwach `Artykuły` i `Autorzy`. Tabela `Artykuły` zawiera teraz dane ściśle powiązane z artykułami, w tym informację o autorze. Dane autorów zostały jednak w niej zapisane nie wprost, lecz za pomocą liczb, które są identyfikatorami w tabeli `Autorzy`. Dlatego też zmieniliśmy nazwę pierwszej kolumny tabeli `Autorzy` z `Lp.` na `ID` (skrót słowa „identyfikator").

W tabeli `Autorzy` znajdują się teraz wszystkie dane dotyczące poszczególnych autorów. Zauważ, że nie trzeba, a wręcz nie powinno się umieszczać w niej dwa razy jednej osoby (w naszym przypadku: Jana Kowalskiego). Zakładamy przy tym, że jeden artykuł ma jednego autora, a przynajmniej jednego autora głównego, który reprezentuje zespół autorów.

Dobra rada

W tabelach bazy danych zadbaj o to, aby każdy wiersz dotyczył tylko jednego obiektu oraz aby ten sam obiekt nie występował w dwóch różnych wierszach.

W opisany powyżej sposób zaprojektowaliśmy dwie tabele bazy danych. W **tabeli bazy danych** (ang. *database table*) są gromadzone informacje na temat obiektów jednego, określonego typu. Obiekty te są opisywane za pomocą cech podanych jako wartości w kolumnach tabeli.

Tabela bazy danych

W bazach danych kolumny tabeli nazywa się **atrybutami** (ang. *attributes*). Nazwy tych atrybutów są nagłówkami kolumn – w naszym przykładzie to np. `Tytuł`, `Termin` czy `Dział`.

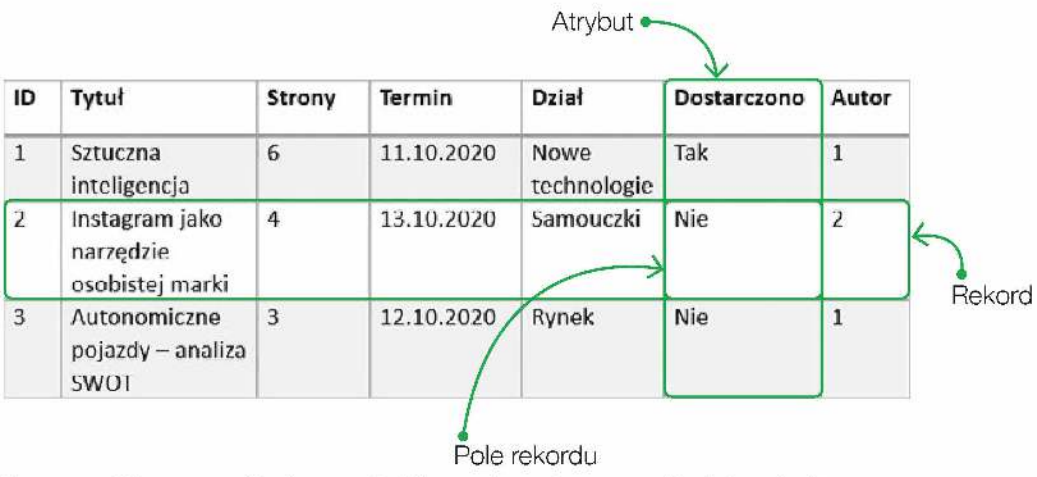
Atrybut

Każdy wiersz tabeli zawiera informacje o jednym obiekcie i jest nazywany **rekordem** (ang. *data record*). Każdy rekord składa się z wielu **pól rekordu** (ang. *values*), które znajdują się na przecięciu rekordu z wybranym atrybutem.

Rekord

Pole rekordu

Rysunek 4.4 przedstawia te pojęcia na przykładzie tabeli `Artykuły`.



ID	Tytuł	Strony	Termin	Dział	Dostarczono	Autor
1	Sztuczna inteligencja	6	11.10.2020	Nowe technologie	Tak	1
2	Instagram jako narzędzie osobistej marki	4	13.10.2020	Samouczki	Nie	2
3	Autonomiczne pojazdy – analiza SWOT	3	12.10.2020	Rynek	Nie	1

Rys. 4.4. Elementy składowe tabeli bazy danych na przykładzie tabeli `Artykuły`

Klucz główny (klucz podstawowy)

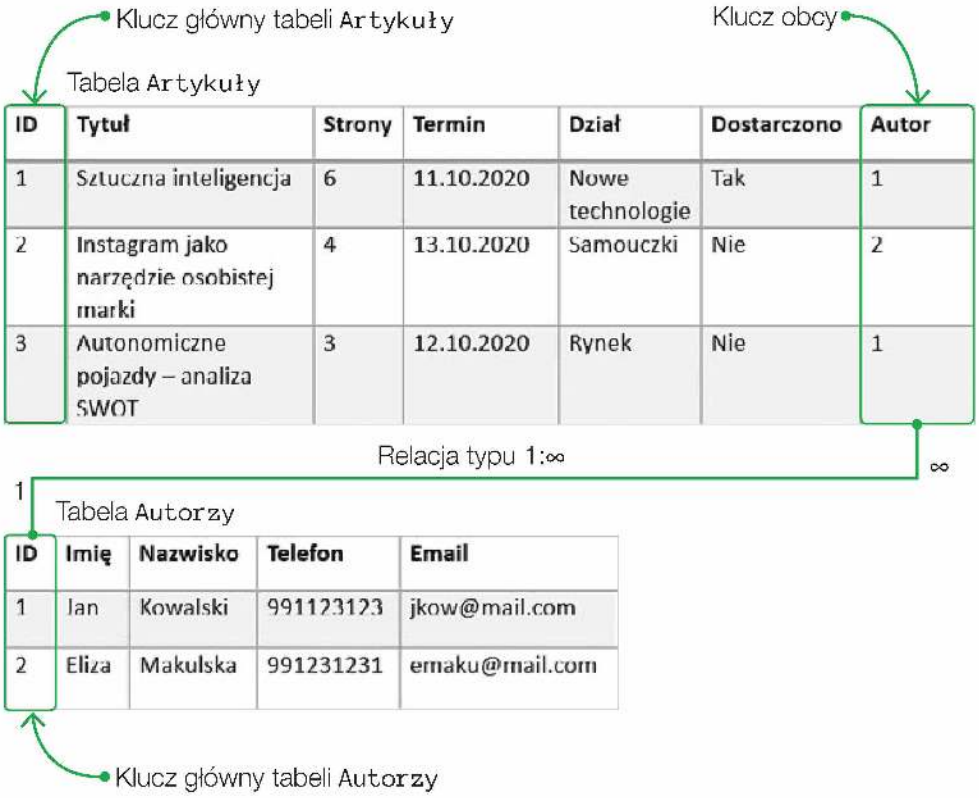
Atrybuty o nazwie ID, znajdujące się w obu tabelach, pełnią funkcję **klucza głównego** (ang. *primary key*), nazywanego również **kluczem podstawowym**, czyli niepowtarzalnego identyfikatora, który jednoznacznie wskazuje konkretny rekord danej tabeli. Klucz główny może się składać z jednej lub wielu kolumn i nie musi przechowywać wartości liczbowych.

Klucz obcy

Kolumna `Autor` tabeli `Artykuły` pełni funkcję tak zwanego **klucza obcego** (ang. *foreign key*), ponieważ wskazuje na klucz główny innej tabeli, w naszym przypadku kolumnę ID tabeli `Autorzy` (rys. 4.5).

Warto wiedzieć

Nazwy kolumn zawierających wartości klucza obcego często tworzy się z wykorzystaniem skrótu ID oraz nazwy tabeli, do której klucza głównego się odnoszą. Na przykład kolumna o nazwie `ID_Autora` zawiera klucze obce odnoszące się do rekordów z tabeli `Autorzy`.



ID	Tytuł	Strony	Termin	Dział	Dostarczono	Autor
1	Sztuczna inteligencja	6	11.10.2020	Nowe technologie	Tak	1
2	Instagram jako narzędzie osobistej marki	4	13.10.2020	Samouczki	Nie	2
3	Autonomiczne pojazdy – analiza SWOT	3	12.10.2020	Rynek	Nie	1

ID	Imię	Nazwisko	Telefon	Email
1	Jan	Kowalski	991123123	jkow@mail.com
2	Eliza	Makulska	991231231	emaku@mail.com

Warto wiedzieć

Atrybut pełniący funkcję klucza głównego tabeli zwyczajowo nazywany jest ID. Jest to skrót słowa „identyfikator".

Rys. 4.5. Klucze główne i klucz obcy tabel `Artykuły` i `Autorzy`

Relacja

Takie powiązanie między rekordami w dwóch tabelach nazywamy relacją. **Relacja** (ang. *relationship*) to połączenie logiczne między dwiema tabelami, określające, które atrybuty pól są wspólne dla tych tabel.

Relacja typu jeden do wielu

W tym przypadku mamy do czynienia z **relacją typu jeden do wielu** (zapisywaną jako 1:∞), ponieważ:

- jeden autor może napisać wiele artykułów, tzn. rekord z tabeli `Autorzy` może być powiązany z wieloma rekordami w tabeli `Artykuły`,
- jeden artykuł ma tylko jednego autora, tzn. jednemu rekordowi z tabeli `Artykuły` odpowiada tylko jeden rekord w tabeli `Autorzy`.

Relacja jeden do wielu jest często występującym typem relacji. Może być zastosowana w bardzo wielu sytuacjach, w których pojedyncza informacja z jednej tabeli jest powiązana z wieloma informacjami z innej tabeli, np. jedna klasa jest powiązana z wieloma uczniami, którzy są do niej przypisani.

Oprócz relacji jeden do wielu istnieją także relacje: jeden do jednego i wiele do wielu. W **relacji typu jeden do jednego** (1:1) jeden rekord tabeli odpowiada dokładnie jednemu rekordowi innej tabeli. Relację tego typu wprowadza się czasem z powodu ograniczeń technicznych, np. kiedy system zarządzania relacyjną bazą danych nie pozwala zgromadzić wymaganej liczby atrybutów w jednej tabeli.

Relacja typu jeden do jednego

W **relacji typu wiele do wielu** (∞:∞) wiele rekordów jednej tabeli może odpowiadać wielu rekordom innej tabeli. Taka relacja sprawdziłaby się również w naszej bazie, jeśli przyjęlibyśmy założenie, że każdy autor może napisać wiele artykułów, a jeden artykuł może być współtworzony przez wielu autorów. Na potrzeby relacji ∞:∞ tworzy się dodatkową tabelę, w której są przechowywane pary kluczy głównych tabel powiązanych ze sobą.

Relacja typu wiele do wielu

Warto wiedzieć

W relacyjnych bazach danych często tabelę nazywa się relacją, ponieważ wiąże ona cechy opisujące określony obiekt.

Ćwiczenie 1

Zaprojektuj bazę danych dowolnego muzeum, zawierającą rejestr zbiorów. Baza powinna składać się z dwóch tabel: `Eksponaty` i `Sale`. Ustal nazwy atrybutów w tabelach, klucze główne oraz relacje między tabelami. Projekt możesz przygotować na kartce lub w wybranym oprogramowaniu.

Zapamiętaj

Relacyjna baza danych to zorganizowany zbiór tabel połączonych relacjami. Każda tabela powinna opisywać jeden typ obiektu i składać się z kolumn zwanych atrybutami, które są zbiorem cech opisujących ten typ. Tabela może zawierać wiele rekordów, czyli pojedynczych wierszy opisujących cechy (atrybuty) różnych obiektów danego (pojedynczego) typu.

4.2. Od projektu do bazy danych

W praktyce do tworzenia baz danych i zarządzania nimi wykorzystuje się tzw. **system zarządzania bazą danych – SZBD** (ang. *Database Management System* – DBMS). Jest to program lub zestaw programów służących do korzystania z relacyjnej bazy danych. Jednym z takich programów jest Microsoft Access.

System zarządzania bazą danych (SZBD)

Dane w programie Microsoft Access są przechowywane w tabelach. Między tabelami można tworzyć relacje, które odpowiadają opisanym wcześniej typom. Za pomocą formularzy można wygodnie wprowadzać i przeglądać pojedyncze rekordy.

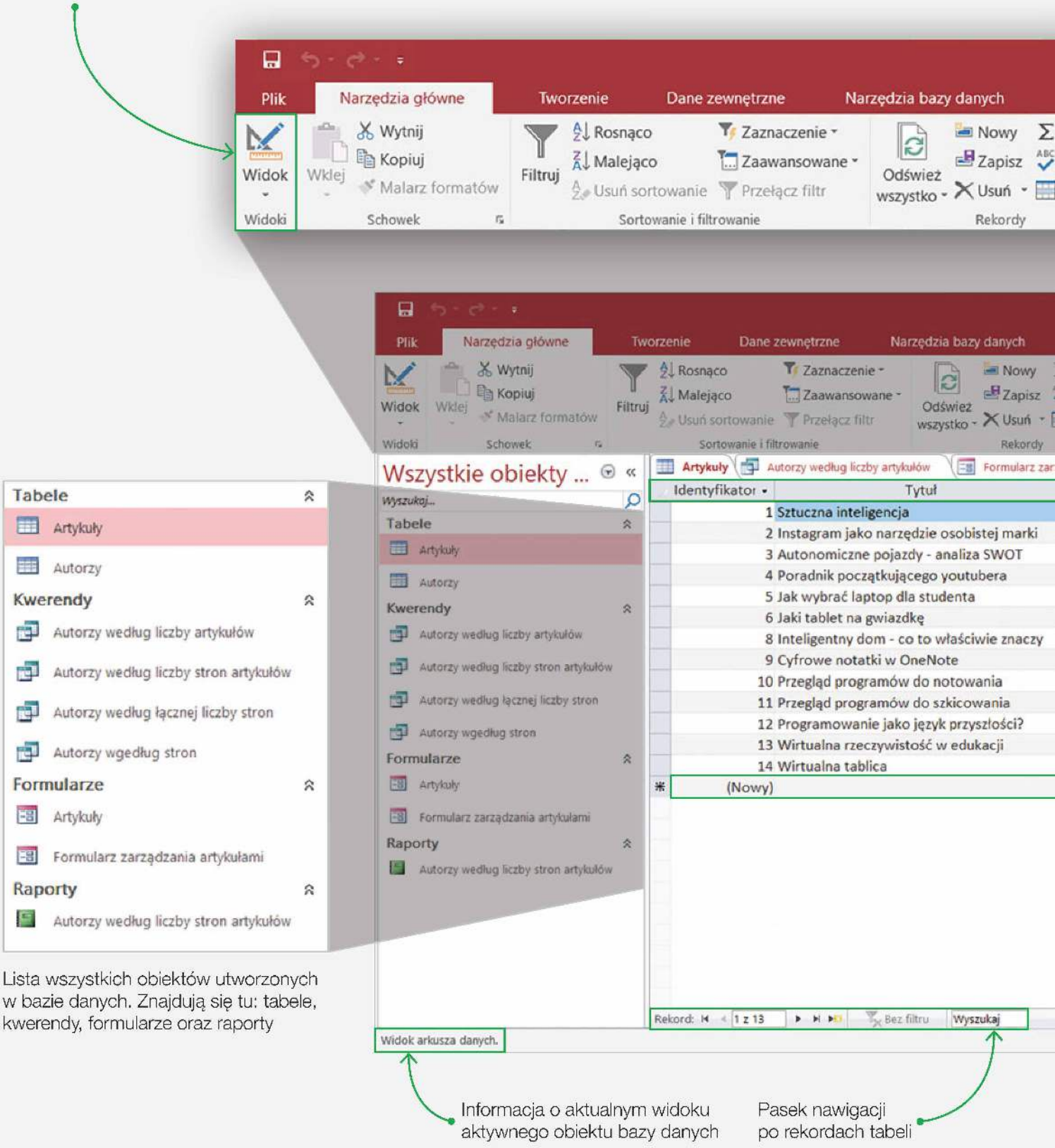
Znajdowanie odpowiednich informacji w bazie danych umożliwiają kwerendy, natomiast dzięki raportom można zbierać wygenerowane przez kwerendy dane i przygotowywać je do wydruku.

Warto wiedzieć

Aby podkreślić relacyjny charakter systemu zarządzania bazą danych, stosuje się niekiedy skrót SZRBD – system zarządzania relacyjną bazą danych.

Okno programu Microsoft Access 2016

W programie Access na kartach głównych są pogrupowane narzędzia o podobnym przeznaczeniu. Jednym z podstawowych narzędzi jest **Widok**, dzięki któremu można się przełączać między widokiem projektu obiektu bazy (np. tabeli lub kwerendy) a np. widokiem arkusza danych



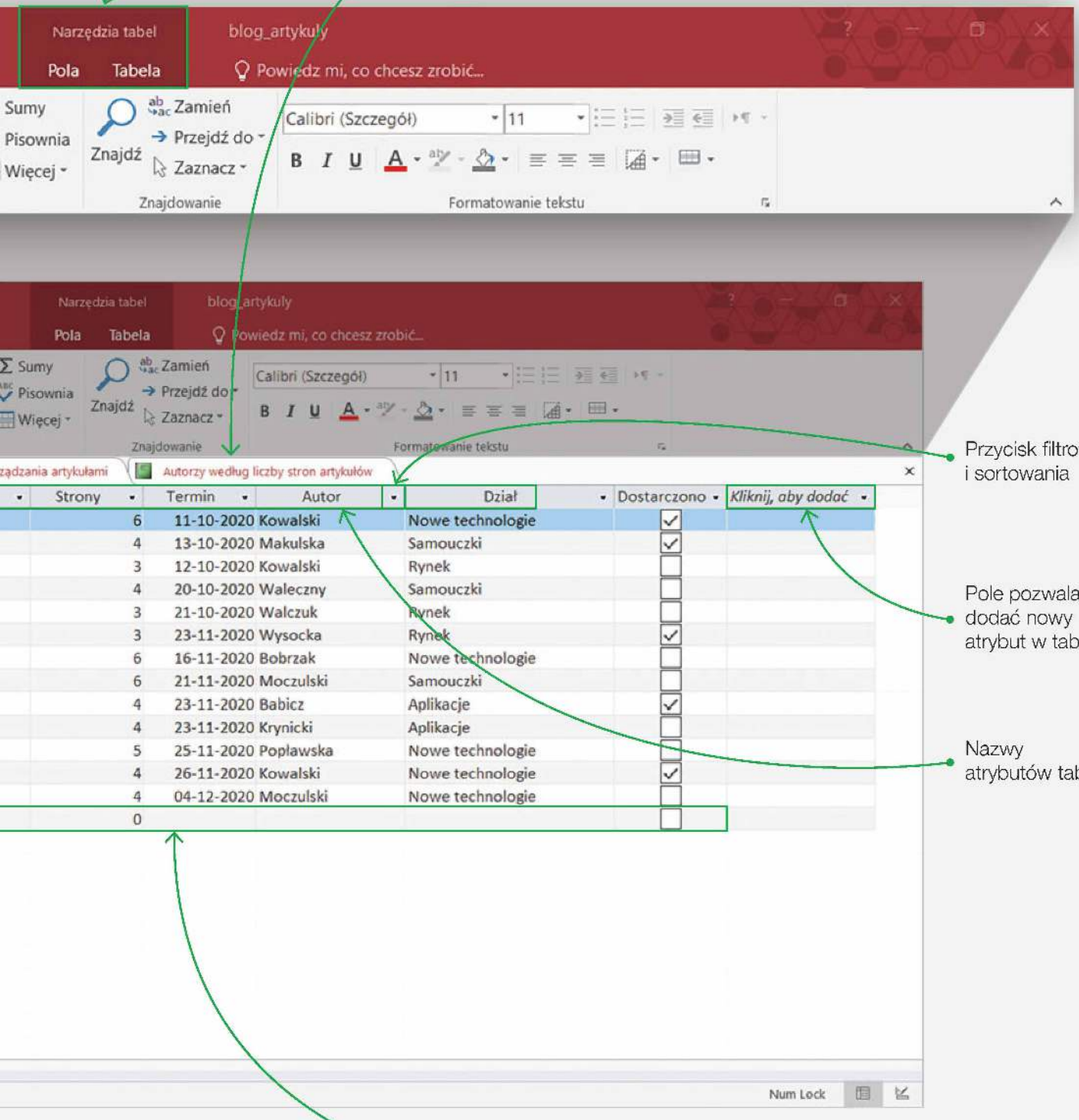
Lista wszystkich obiektów utworzonych w bazie danych. Znajdują się tu: tabele, kwerendy, formularze oraz raporty

Informacja o aktualnym widoku aktywnego obiektu bazy danych

Pasek nawigacji po rekordach tabeli

Po uaktywnieniu wybranego obiektu (w tym wypadku tabeli `Artykuły`) pojawiają się dodatkowe karty, zawierające narzędzia dotyczące tylko tego obiektu (tu: **Narzędzia tabel**, a w nich karty **Pola** oraz **Tabela**)

W kolejnych zakładkach można przeglądać i modyfikować zawartość elementów bazy danych w różnych trybach widoku. Zakładkę danego obiektu możemy otworzyć po dwukrotnym kliknięciu jego nazwy na liście wszystkich obiektów.



Przycisk filtrowania i sortowania

Pole pozwalające dodać nowy atrybut w tabeli

Nazwy atrybutów tabeli

Wypełnienie ostatniego wiersza tabeli powoduje dodanie do niej nowego rekordu z danymi

Tabele w programie Access

Tabela programu Access

Podstawowym elementem programu Access jest **tabela** – w niej przechowywane są dane dotyczące opisywanych obiektów. Aby dodać do bazy nową tabelę, należy wybrać **Tworzenie** → **Tabele** → **Tabela**. Otworzy się nowa zakładka z tabelą zawierającą tylko jeden atrybut o domyślnej nazwie `Identyfikator`.

Załóżmy, że chcemy dodać tabelę opisującą autorów, taką jak zaprojektowana na rysunku 4.5 ze s. 70 . W naszym projekcie nie ma jednak atrybutu `Identyfikator`, tylko ID. Aby zmienić nazwę atrybutu, należy ją kliknąć prawym przyciskiem myszy i z menu kontekstowego wybrać opcję **Zmień nazwę pola**.

Aby dodać kolejny atrybut, wybieramy opcję **Kliknij, aby dodać**, znajdującą się po prawej stronie ostatniego atrybutu tabeli (w naszym przypadku atrybutu ID). Następnie wybieramy typ danych, które będą w nim przechowywane, i wprowadzamy nazwę atrybutu, np. `Imię`.

Projektując tabelę bazy danych i jej atrybuty, należy określić **typ** poszczególnych danych. W przypadku imienia, nazwiska lub adresu e-mail odpowiednie będzie pole tekstowe (**Krótki tekst**) o długości do 255 znaków. W większości przypadków będzie to nawet więcej, niż potrzeba. Dla numeru telefonu można wybrać pole typu **Liczba**. W tabeli `Autorzy` wprowadzamy zatem następujące typy atrybutów: `Nazwisko` (**Krótki tekst**), `Telefon` (**Liczba**), `E-mail` (**Krótki tekst**).

Na rysunku 4.6 pokazano typy danych przypisane do pól tabeli `Autorzy`. Widać na nim również pełną listę typów danych programu Access.

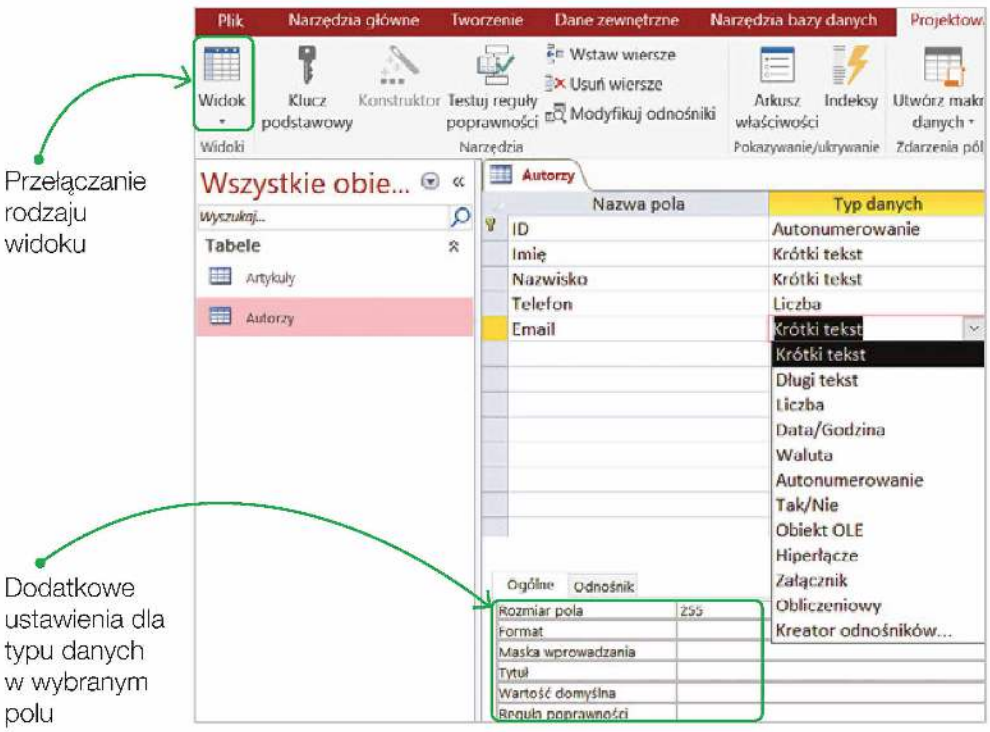
Warto wiedzieć

Atrybut opisuje pewną własność obiektu, np. cenę, kolor, kod pocztowy, nazwisko, markę. Dla danego rekordu wybrany atrybut nazywa się czasem polem. W programie Microsoft Access używa się tego terminu również do określenia nazwy nagłówka tabeli.

Typy danych i maski w programie Microsoft Access, s. 248

Warto wiedzieć

W przypadku numeru telefonu ustalenie typu pola nie jest oczywiste. Zazwyczaj wystarczy podać tylko 9 cyfr, a w polu ustawić typ **Liczba**. Dla numerów z zagranicy to jednak za mało, gdyż trzeba uwzględnić choćby numer kierunkowy kraju (np. +33). Dla tak zapisanych danych można skorzystać z typu **Krótki tekst**.



Rys. 4.6. Zarządzanie tabelą w Widoku projektu

Po dodaniu wszystkich kolumn (atrybutów) należy pamiętać o zapisaniu tabeli. W tym celu możemy kliknąć prawym klawiszem jej zakładkę, a następnie z menu kontekstowego opcję **Zapisz**. Ponieważ zapisujemy tabelę po raz pierwszy, pojawi się okno **Zapisz jako**, w którym możemy wprowadzić nazwę tabeli – w naszym przypadku `Autorzy`.

Dodawać nowe atrybuty oraz określić typ danych możemy również w **Widoku projektu**. Aby przełączyć się na ten widok, należy zaznaczyć zakładkę danego obiektu (np. tabeli `Autorzy`), a następnie wybrać **Narzędzia główne** → **Widoki** → **Widok** → **Widok projektu** (rys. 4.6).

Widok projektu

Pliki bazy danych programu Access zapisywane są w pliku z domyślnym rozszerzeniem *accdb*.

Warto wiedzieć

Po zapisaniu tabeli jej nazwa pojawi się na liście wszystkich obiektów bazy.

Ćwiczenie 2

W bazie danych udostępnionej przez nauczyciela (np. w pliku *baza_blog_2.accdb*), korzystając z narzędzia **Widok projektu**, utwórz tabelę `Autorzy` z atrybutami odpowiednich typów. Plik zapisz pod nazwą *baza_blog_3.accdb*.

Przeglądanie i modyfikowanie danych w tabelach

Aby ułatwić sobie przeglądanie danych w pojedynczej tabeli, warto korzystać z **filtrowania** i **sortowania**. Działają one tak samo jak narzędzia dostępne w arkuszu kalkulacyjnym.

Filtrowanie, sortowanie, s. 30

Jeśli chcemy poprawić dane, w otwartej zakładce wybranej tabeli wskazujemy odpowiednie pole w widoku arkusza danych i wprowadzamy poprawkę lub dodajemy informację, której brakowało.

Aby **zapisać zmiany**, klikamy zakładkę z nazwą tabeli prawym przyciskiem myszy i wybieramy opcję **Zapisz**.

Zapisywanie zmian w tabeli

Ćwiczenie 3

Otwórz plik *baza_blog_3.accdb*, utworzony w ćwiczeniu 2, a następnie:

a. posortuj rekordy tabeli `Artykuły` od najwcześniejszego terminu dostarczenia utworu,
b. wyświetl tylko tytuły artykułów, których nie dostarczono,
c. zmień tytuł artykułu *Sztuczna inteligencja* na *Systemy sztucznej inteligencji*.

Plik bazy zapisz pod nazwą *baza_blog_4.accdb*.

Warto wiedzieć

O tym, że na dane w tabeli został założony filtr, świadczy ikona lejka wyświetlona przy nazwie danej kolumny (obok ikony narzędzia filtrowania).

Usuwanie i dodawanie rekordu

Jeśli wiemy, że któryś z artykułów do naszego serwisu nie zostanie napisany, wówczas możemy usunąć z bazy danych cały rekord, który go dotyczy. W tym celu zaznaczamy odpowiadający mu wiersz, a następnie z menu kontekstowego wybieramy opcję **Usuń rekord**. Po potwierdzeniu operacji (rys. 4.7, s. 76) wiersz zostanie usunięty z tabeli.

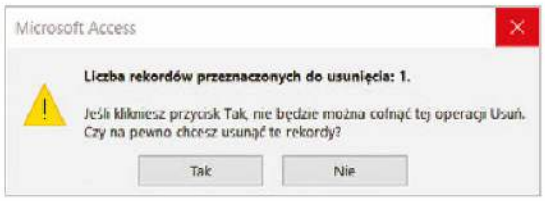
Warto wiedzieć

Program Access pozwala stosować polskie znaki diakrytyczne, ale nie wszystkie SZBD oferują taką możliwość. Dlatego w bazach danych zazwyczaj unika się polskich znaków. Ponadto nazwy tabel przyjęło się zaczynać wielką literą, a nazwy atrybutów – małą.

Warto wiedzieć

Często zamiast usuwać rekordy z bazy, oznacza się je dodatkowym atrybutem jako nieaktywne lub nieaktualne. W niektórych zastosowaniach takie postępowanie wręcz jest konieczne, a usuwanie danych zabronione – np. w Biuletynach Informacji Publicznej (BIP).

Aby dodać nowy rekord, wystarczy uzupełnić pola w pustym wierszu na końcu tabeli. Pole identyfikatora program uzupełni automatycznie.



Rys. 4.7. Potwierdzenie usunięcia rekordu

Ćwiczenie 4

W pliku *baza_blog_4.accdb* z ćwiczenia 3 wykonaj następujące czynności:

a. usuń z tabeli `Artykuły` artykuł o ID 4 i tytule *Roboty wokół nas*,
b. uzupełnij w tabeli `Autorzy` listę autorów zgodnie z projektem bazy danych z rysunku 4.5 ze s. 70,
c. dodaj nowy, wymyślony przez ciebie artykuł autorstwa Elizy Makulskiej.

Warto wiedzieć

Systemy zarządzania relacyjnymi bazami danych zazwyczaj dbają o zachowanie zasady niepowtarzalności klucza głównego i automatycznie uzupełniają pole identyfikatora stanowiącego ten klucz.

W bazie danych nie mogą istnieć dwa rekordy z takim samym kluczem głównym. Jeśli dany rekord zostanie usunięty, przynależny do niego klucz główny nie powinien zostać przyporządkowany innemu rekordowi (również nowemu).

Gdy w tabeli bazy danych programu Access usuniemy rekord o danym identyfikatorze, to żaden z nowych rekordów nie otrzyma już tego, teoretycznie zwolnionego, numeru. Jest to jedno z wielu zabezpieczeń oferowanych przez systemy zarządzania bazami danych.

Maski wprowadzania i format wyświetlania danych

W wielu sytuacjach określenie typu danych nie wystarczy. W przypadku numeru telefonu zapisanego jako liczba system nie powinien dopuszczać wartości składających się np. wyłącznie z jednej cyfry, ponieważ takie numery telefonów nie istnieją.

Aby zapewnić poprawność informacji wpisywanych do bazy, możemy użyć tzw. **maski wprowadzania**, która ma postać ciągu znaków określającego sposób wprowadzania danych. W programie Access maski wprowadzania określa się w **Widoku projektu** wybranej tabeli.

Maska wprowadzania

Warto wiedzieć

Międzynarodowe numery telefoniczne mogą mieć do 15 cyfr. Wynika to z zalecenia Międzynarodowego Związku Telekomunikacyjnego – najstarszej tego typu organizacji na świecie.

Typy danych i maski w programie Microsoft Access, s. 248

W przypadku numeru telefonu możemy założyć, że składa się on przynajmniej z 9 cyfr, ale nie więcej niż z 15 cyfr. W programie Access maska taka będzie mieć postać: 000000000999999, gdzie 0 oznacza cyfrę obowiązkową, a 9 – opcjonalną.

W **Widoku projektu** możemy także zmienić format danych widocznych w polu – m.in. zdecydować, jak będzie wyświetlana data (np. „16.11.2020" czy „poniedziałek, 16 listopada 2020"). **Format** możemy wybrać z listy rozwijanej lub użyć do jego określenia odpowiednich symboli. Aby zobaczyć wprowadzoną zmianę formatu, należy zapisać zmiany w tabeli jeszcze przed przełączeniem się do **Widoku arkusza danych**.

Definiowanie relacji

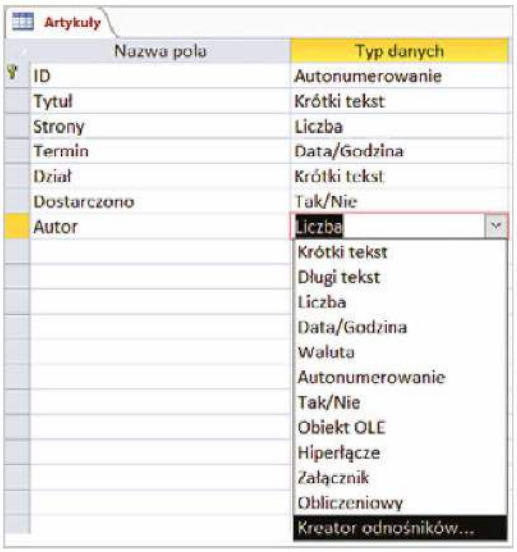
W naszej bazie zaplanowaliśmy relację między tabelami **typu jeden do wielu**. Do tej pory jednak nie stworzyliśmy żadnych mechanizmów, które by z niej korzystały. Możliwość tworzenia relacji jest jednym z podstawowych zadań bazy danych. Dzięki temu będziemy mogli wyświetlać wybrane dane o autorach, mimo że formalnie znajdują się one w odrębnej tabeli.

Relacja typu jeden do wielu, s. 70

Do stworzenia relacji w programie Microsoft Access wykorzystamy **Kreator odnośników**, który pozwoli nie tylko powiązać ze sobą pola tabel, lecz także wyświetlać w obu tabelach powiązane ze sobą informacje (rys. 4.12 i 4.13, s. 79).

Kreator odnośników

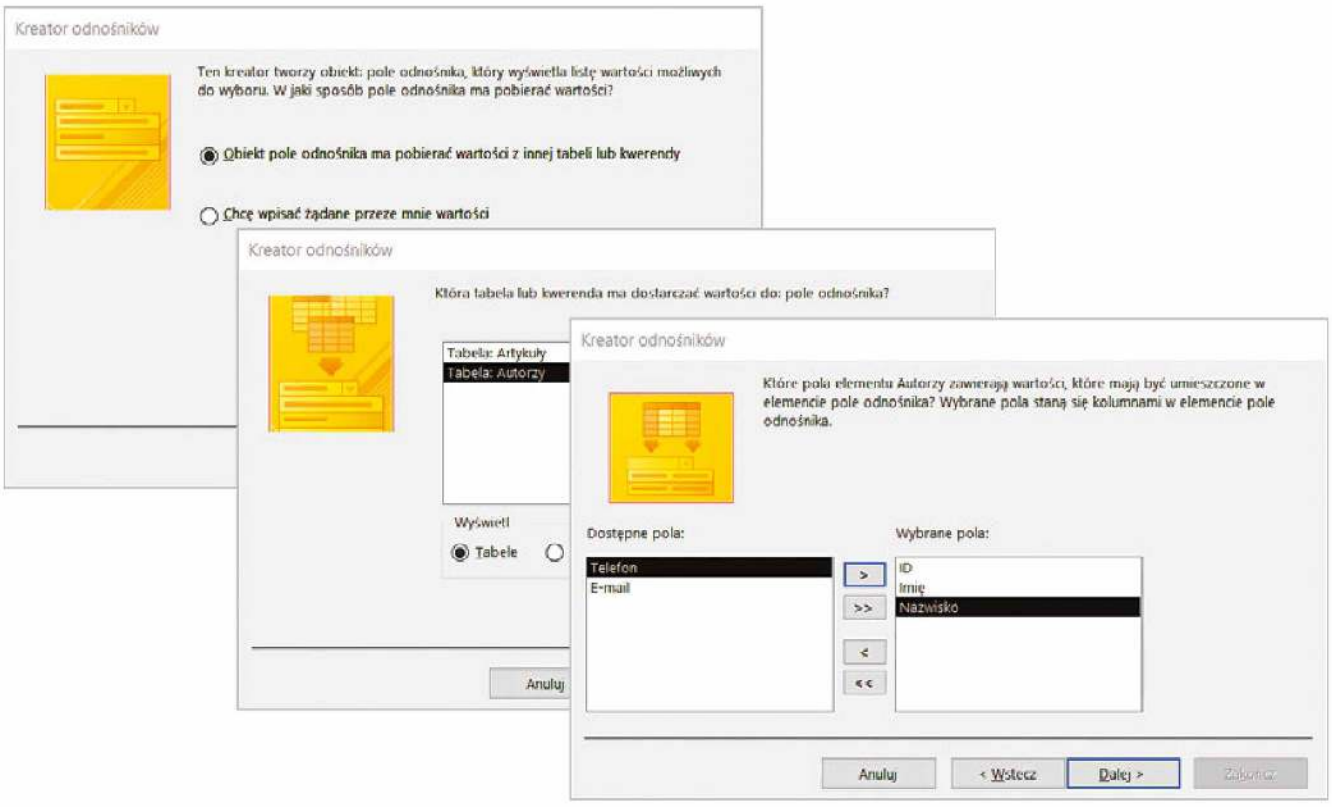
Aby go uruchomić, przechodzimy do **Widoku projektu** dla tabeli Artykuły. Następnie dla pola Autor w kolumnie **Typ danych** rozwijamy menu kontekstowe i wybieramy narzędzie **Kreator odnośników** (rys. 4.8). Najpierw wskazujemy tabelę Autorzy, a następnie pola, których chcemy użyć w tabeli Artykuły (rys. 4.9).



Rys. 4.8. Wybór Kreatora odnośników

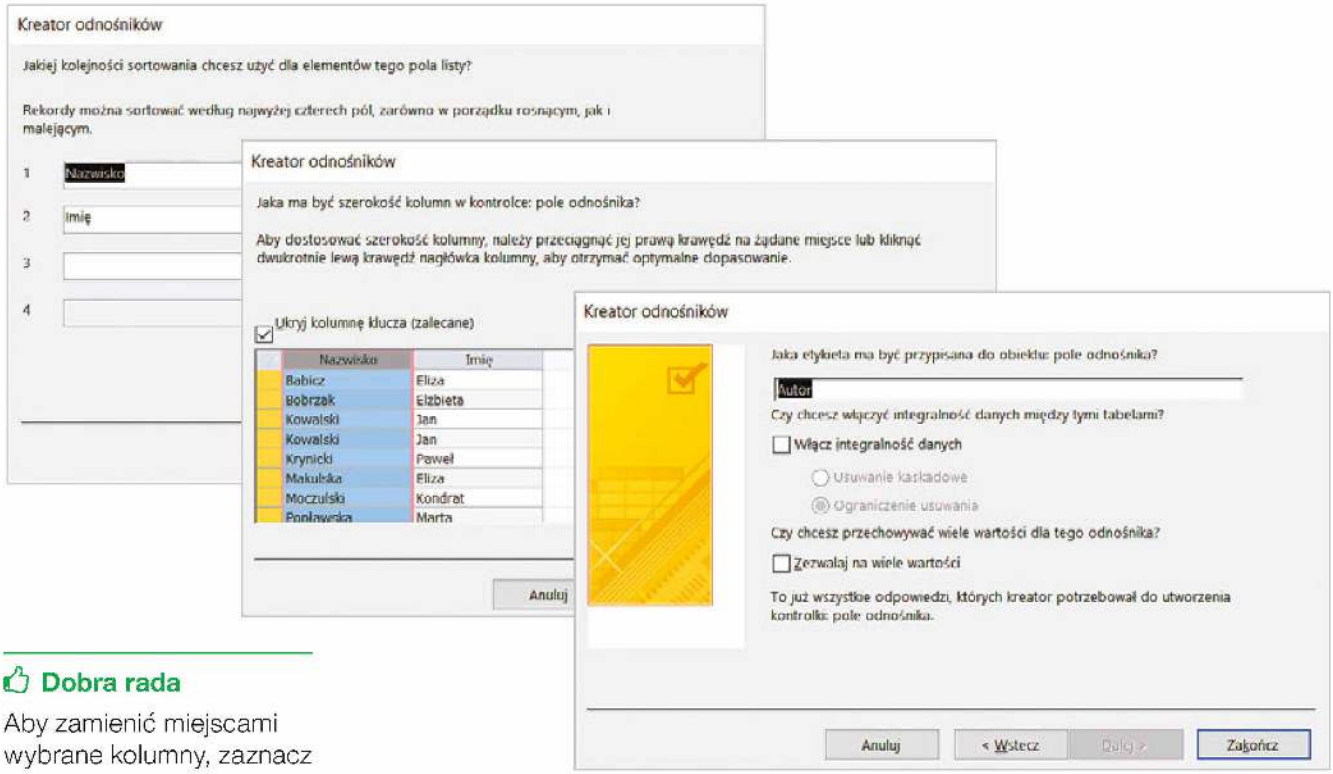
Dobra rada

Jeśli w twojej w bazie danych znajdują się dwie osoby o takim samym imieniu i nazwisku, to podczas tworzenia odnośnika zawierającego te dane dodaj dodatkowy atrybut, który będzie je odróżniał, np. adres e-mail.



Rys. 4.9. Wybór pól tabeli w Kreatorze odnośników

Po wybraniu pól i ich zatwierdzeniu możemy zdecydować, w jaki sposób zostanie posortowana lista z ich wartościami. Wybieramy atrybuty `Nazwisko` oraz `Imię` i w obu przypadkach zostawiamy opcję sortowania rosnącego, które dla tekstu oznacza kolejność alfabetyczną. Ustawiamy również szerokość kolumn i ich kolejność, a także nazwę etykiety (rys. 4.10). Domyślnie pole `ID` (które jest kluczem obcym) będzie ukryte.



Rys. 4.10. Ustawienia wyświetlania pól w **Kreatorze odnośników**

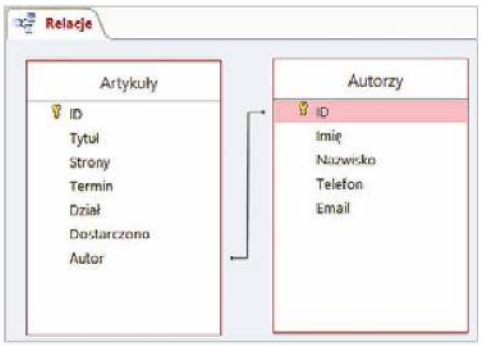
Dobra rada

Aby zamienić miejscami wybrane kolumny, zaznacz jedną z nich, a następnie chwyć kursorem nazwę kolumny i przeciągnij ją na właściwe miejsce.

Dobra rada

Jeśli po utworzeniu odnośnika kolejność kolumn jest niewłaściwa, kliknij prawym przyciskiem myszy nazwę kolumny, wybierz z menu kontekstowego opcję **Modyfikuj odnośniki** i przejdź do kroku, w którym możesz zmienić kolejność kolumn.

Po zapisaniu zmian w tabeli możemy sprawdzić utworzoną relację między tabelami `Autorzy` oraz `Artykuły`. W tym celu zaznaczamy jedną z tabel i wybieramy: **Narzędzia tabel → Tabela → Relacje → Relacje** (rys. 4.11). Powinna być widoczna linia łącząca atrybut `Autor` tabeli `Artykuły` z atrybutem `ID` tabeli `Autorzy`.



Rys. 4.11. Relacja między tabelami `Artykuły` i `Autorzy`

Poza utworzeniem relacji między tabelami w tabeli zawierającej **klucz obcy** pojawią się dane z powiązanej z nią tabeli. Aby to zobaczyć, należy przełączyć się na **Widok arkusza danych**. W kolumnie `Autor` powinny być widoczne nazwiska autorów zamiast ich identyfikatorów. Po wybraniu pola z nazwiskiem autora powinna się pojawić lista wszystkich autorów. Pokazano to na rysunku 4.12.

Klucz obcy, s. 70

Artykuły

ID	Tytuł	Strony	Termin	Dział	Dostarczono	Autor	Kliknij, aby
1	Sztuczna inteligencja	6	11-10-2020	Nowe technologie	☑	Kowalski	
2	Instagram jako narzędzie osobistej marki	4	13-10-2020	Samouczki	☑	Babicz	Eliza
3	Autonomiczne pojazdy - analiza SWOT	3	12-10-2020	Rynek	☐	Makulska	Eliza
4	Poradnik początkującego youtubera	4	20-10-2020	Samouczki	☐	Bobrzak	Elżbieta
5	Jak wybrać laptop dla studenta	3	21-10-2020	Rynek	☐	Wysocka	Gabriela
6	Jaki tablet na gwiazdkę	3	23-11-2020	Rynek	☑	Kowalski	Jan
8	Inteligentny dom - co to właściwie znaczy	6	16-11-2020	Nowe technologie	☐	Kowalski	Jan
9	Cyfrowe notatki w OneNote	6	21-11-2020	Samouczki	☐	Wierzbicki	Janusz
10	Przegląd programów do notowania	4	23-11-2020	Aplikacje	☑	Moczulski	Kondrat
11	Przegląd programów do szkicowania	4	23-11-2020	Aplikacje	☐	Popławska	Marta
12	Programowanie jako język przyszłości?	5	25-11-2020	Nowe technologie	☐	Krynicki	Paweł
13	Wirtualna rzeczywistość w edukacji	4	26-11-2020	Nowe technologie	☑	Walczuk	Paweł
						Waleczny	Piotr

Rys. 4.12. Widok tabeli Artykuły po powiązaniu z tabelą Autorzy

W **Widoku arkusza danych** tabeli Autorzy przy każdym rekordzie pojawia się znak plusa. Jego wybranie spowoduje wyświetlenie wszystkich rekordów z tabeli Artykuły powiązanych z danym rekordem tabeli Autorzy (rys. 4.13).

Autorzy

ID	Imię	Nazwisko	Email	Telefon	Kliknij, aby dodać
1	Jan	Kowalski	janwie@mail.com	991123123	

ID	Tytuł	Strony	Termin	Dział	Dostarczono
1	Sztuczna inteligencja	6	11-10-2020	Nowe technologie	☑
3	Autonomiczne pojazdy - analiza SWOT	3	12-10-2020	Rynek	☐
13	Wirtualna rzeczywistość w edukacji	4	26-11-2020	Nowe technologie	☑
(Nowy)		0			☐

2	Eliza	Makulska	kowal@mail.com	991231231
3	Piotr	Waleczny	piowa@mail.com	990212121

Dane z tabeli Artykuły związane z wybranym autorem

Rys. 4.13. Widok tabeli Autorzy po utworzeniu relacji

Ćwiczenie 5

W pliku bazy danych, który otrzymasz od nauczyciela (np. *baza_blog_5.accdb*), stwórz odnośnik powodujący, że w kolumnie Autor tabeli Artykuły wyświetli się nazwisko autora z odpowiadającego mu rekordu w tabeli Autorzy. Plik zapisz pod nazwą *baza_blog_6.accdb*.

Dobra rada

Jeśli dwie tabele twojej bazy danych są związane relacją, to w **Widoku arkusza danych** jednej z tych tabel możesz wprowadzać zmiany dotyczące rekordów drugiej tabeli. Możesz również dodawać nowe rekordy.

A to ciekawe

Bieżące raportowanie sprzedaży

Jedna z największych firm technologicznych wdrożyła we wszystkich swoich sklepach system, który w czasie rzeczywistym zbiera dane na temat sprzedanych produktów. Zaprojektowano w tym celu specjalną bazę danych. Dzięki temu rozwiązaniu można przewidywać zapotrzebowanie na produkty i wytwarzać je tak, żeby zaspokoić popyt bez ryzyka nadprodukcji.



Zapamiętaj

Za pomocą **Kreatora odnośników** w programie Microsoft Access można jednocześnie stworzyć relację między tabelami bazy danych oraz spowodować, że w polu klucza obcego będą wyświetlane odpowiadające mu dane z tabeli, na którą ten klucz wskazuje.

4.3. Formularze – zarządzanie danymi i ich przeglądanie

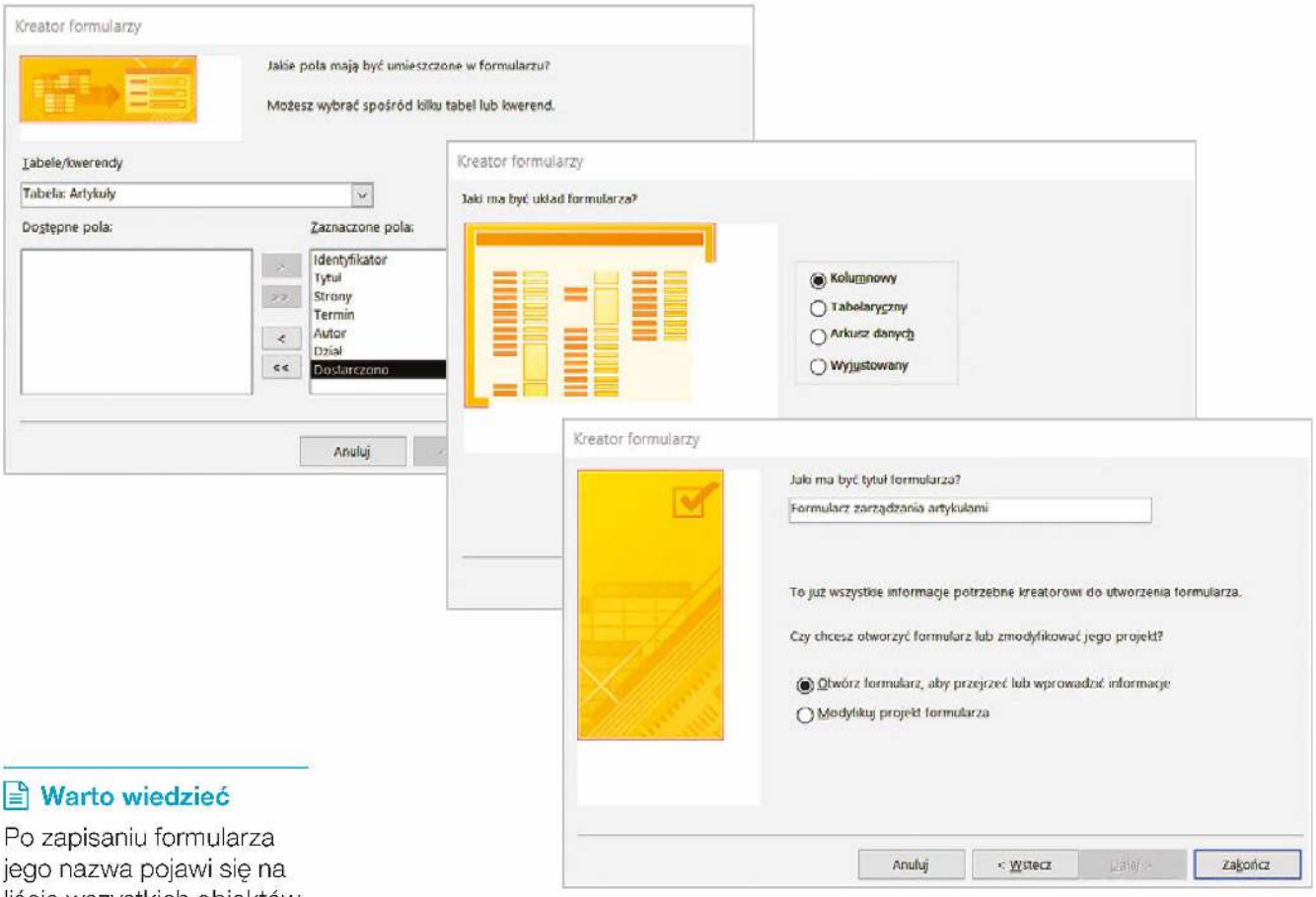
Formularze

Formularze programu Microsoft Access umożliwiają przeglądanie i modyfikowanie danych zgromadzonych w bazie oraz dodawanie nowych. Dzięki formularzom możemy skupić się na pojedynczym rekordzie zamiast na całej tabeli.

Tworzenie formularza

Aby utworzyć nowy formularz, wybieramy **Tworzenie** → **Formularze** → **Kreator formularzy**. Wskazujemy wszystkie pola tabeli `Artykuły`.

W naszym przykładzie możemy zostawić domyślnie wybrany układ tabelaryczny i nadać formularzowi nazwę: `Formularz zarządzania artykułami` (rys. 4.14).



Rys. 4.14. Tworzenie formularza za pomocą **Kreatora formularzy**

Warto wiedzieć

Po zapisaniu formularza jego nazwa pojawi się na liście wszystkich obiektów bazy.

W formularzu wyświetlone są wartości poszczególnych pól wybranego rekordu (rys. 4.15). Możemy je modyfikować, a zmiany automatycznie zostaną dodane do tabeli po przejściu do innego rekordu.



Rys. 4.15. Widok formularza utworzonego dla danych z tabeli `Artykuły`

Dobra rada

Korzystając z przycisków umieszczonych na pasku pod formularzem, możesz przechodzić do pierwszego, poprzedniego, kolejnego, ostatniego lub nowego rekordu tabeli.



Ćwiczenie 6

W pliku *baza_blog_6.accdb* z poprzedniego ćwiczenia utwórz formularz dla tabeli `Artykuły` o nazwie `Formularz zarządzania artykułami`. Plik zapisz pod nazwą *baza_blog_7.accdb*.

Modyfikowanie formularza

Formularz możemy dostosować poprzez zmianę wielkości pól, czcionki czy dopasowanie tła i kolorystyki poszczególnych elementów. Możemy także dodać przyciski, np. ułatwiające mniej zaawansowanym użytkownikom korzystanie z bazy.

Aby zmodyfikować formularz, wybieramy **Narzędzia główne** → **Widoki** → **Widok** → **Widok projektu**. Wygląd poszczególnych elementów (pól i etykiet) zmieniamy w podobny sposób jak w innych programach pakietu Microsoft Office – zaznaczamy wybrany element, a następnie korzystamy z odpowiednich narzędzi formatowania, które znajdziemy m.in. na kartach **Formatowanie**, **Rozmieszczanie** oraz **Projektowanie**.

Dobra rada

Możesz powiększyć domyślny obszar formularza, chwytając i przeciągając jego brzegi kursorem myszy. Możesz też przesunąć do dołu **Stopkę formularza**, by zyskać więcej miejsca np. na przyciski.

Ćwiczenie 7

Otwórz plik *baza_blog_7.accdb*, utworzony w poprzednim ćwiczeniu. Zmodyfikuj wygląd istniejącego w nim formularza, zmieniając wielkość i kolor czcionek etykiet. Dodaj miejsce pod polami formularza, przesuwając jego stopkę nieco w dół. Zapisz plik pod nazwą *baza_blog_8.accdb*.

Dobra rada

Do formularza możesz dodać przyciski, np. pozwalające przełączać się między rekordami. Aby wstawić przycisk, w **Widoku projektu** formularza wybierz: **Narzędzia projektowania formularzy** › **Projektowanie** → **Formanty** → **Przycisk** i narysuj przycisk. Następnie postępuj zgodnie z instrukcjami **Kreatora przycisków poleceń**.

4.4. Kwerendy – wyszukiwanie i łączenie danych

Kwerenda

Kwerenda pozwala znajdować i pobierać z tabel lub innych kwerend dane spełniające określone kryteria. Tworzenie kwerend jest jedną z podstawowych operacji wykonywanych przez użytkowników baz danych. Dzięki nim można odpytywać bazę z informacji dotyczących przechowywanych w niej danych. W naszej bazie utworzymy kwerendę, dzięki której dowiemy się, ile w sumie stron artykułów zobowiązał się dostarczyć każdy z autorów.

Aby utworzyć nową kwerendę, wybieramy **Tworzenie** → **Kwerendy** → **Projekt kwerendy**. W ten sposób przejdziemy do widoku projektu pustej kwerendy. Na ekranie pojawi się okno **Pokazywanie tabeli**, w którym możemy dodać wybrane tabele do projektu. Dodajemy tabele `Artykuły` oraz `Autorzy`, ponieważ w obu tych tabelach znajdują się dane, które chcemy wyświetlić. Dodane tabele pojawią się w widoku projektu, pokaże się także łącząca je relacja.

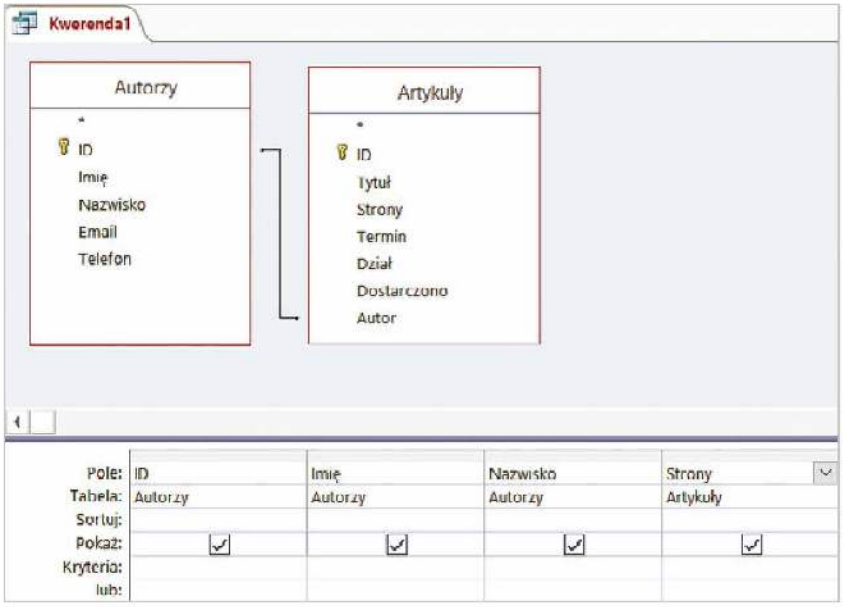
Z obu tabel wybieramy nazwy pól, przeciągamy je do dolnej części ekranu projektu i umieszczamy na liście pól tworzących daną kwerendę (rys. 4.16). Ponieważ chcemy mieć dane autora, na początek przeciągniemy z tabeli `Autorzy` pola `ID`, `Imię` oraz `Nazwisko`. Następnie z tabeli `Artykuły` wybierzemy pole `Strony`, bo to dzięki niemu będzie można obliczyć, ile stron zobowiązał się napisać każdy z autorów.

Warto wiedzieć

Do kwerendy można dodawać nie tylko tabele, lecz także inne, wcześniej utworzone kwerendy. Jest to możliwe, ponieważ wynikiem działania kwerendy jest w rzeczywistości zbiór wybranych danych ułożonych w formie tabeli.

Warto wiedzieć

SQL to język stworzony do tworzenia zapytań w relacyjnych bazach danych. W programie Access po utworzeniu kwerendy możemy zobaczyć jej zapis w języku SQL. W tym celu wystarczy przełączyć się z **Widoku projektu** lub **Widoku arkusza danych** na **Widok SQL**.



Rys. 4.16. Dodawanie tabel i wybieranie z nich pól tworzących kwerendę

Dobra rada

Do **Widoku projektu** kwerendy przejdziesz analogicznie jak w przypadku innych obiektów (np. formularzy), wybierając: **Narzędzia główne** → **Widoki** → **Widok** › **Widok projektu**.

Na tym etapie możemy już sprawdzić działanie kwerendy – wystarczy na karcie **Projektowanie** wybrać przycisk **Uruchom**. Powinniśmy zobaczyć listę autorów, a przy każdym nazwisku – liczbę stron.

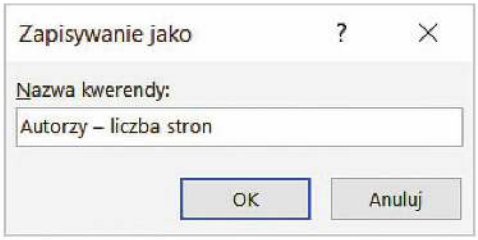
Autorzy się jednak powtarzają, a my chcemy znać łączną liczbę stron dla każdego autora. Należy więc zmodyfikować kwerendę. W tym celu przechodzimy do **Widoku projektu** kwerendy i wybieramy **Projektowanie** → **Pokazywanie/ukrywanie** → **Sumy**.

Zauważ, że na liście pól wybranych do kwerendy pojawiła się opcja **Grupuj według**. Aby zsumować liczbę stron, rozwijamy tę opcję pod polem Strony i wybieramy **Suma**. Jeżeli chcemy, żeby lista była posortowana w kolejności od autora z największą łączną liczbą stron, powinniśmy jeszcze wybrać opcję sortowania **Malejąco**. Możemy także odznaczyć wyświetlanie identyfikatora autora. Na rysunku 4.17 pokazano ustawienia kwerendy oraz jej wywołanie.



Rys. 4.17. Ustawienia kwerendy wyświetlającej liczbę wszystkich stron

Jeżeli kwerenda działa zgodnie z naszymi oczekiwaniami, możemy ją zapisać. W tym celu klikamy nazwę otwartej zakładki kwerendy prawym przyciskiem myszy i z menu kontekstowego wybieramy opcję **Zapisz**. W oknie **Zapisywanie jako** należy wprowadzić nazwę, która będzie odpowiadała działaniu kwerendy (rys. 4.18).



Rys. 4.18. Zapisywanie kwerendy

Jeżeli wynik kwerendy jest inny od oczekiwanego, należy przełączyć się do **Widoku projektu** i wprowadzić poprawki.

Warto wiedzieć

Po zapisaniu kwerendy jej nazwa pojawi się na liście wszystkich obiektów bazy.

A to ciekawe

Bazy danych a *big data*

W biznesie dużą popularność zyskują rozwiązania typu *big data*, czyli nierelacyjne bazy danych, w których informacje są zapisywane nie w tabelach, ale w plikach o dowolnej strukturze. Rozwiązania te umożliwiają powiązanie różnorodnych danych, znajdowanie relacji między nimi i wyciąganie wniosków. W ten sposób można np. na podstawie analizy zapytań w wyszukiwarkach internetowych ustalić miejsce wybuchu epidemii grypy lub przewidywać trendy na rynkach finansowych. Nazwa *big data* ma swoje uzasadnienie, bowiem odnosi się do gigantycznych ilości danych zmieniających się w czasie. Dane te liczone są w tera- lub petabajtach.



Ćwiczenie 8

W pliku *baza_blog_8.accdb* z poprzedniego ćwiczenia utwórz kwerendę wyświetlającą imiona i nazwiska autorów oraz łączną liczbę stron artykułów, do których napisania się zobowiązali. Posortuj listę według nazwisk. Plik zapisz pod nazwą *baza_blog_9.accdb*.

4.5. Raporty – wizualne przedstawienie danych

Raport

Dzięki **raportom** można w czytelny sposób przedstawić dane pochodzące z bazy, aby je potem wyświetlić lub wydrukować.

Tworzenie raportu z użyciem kreatora

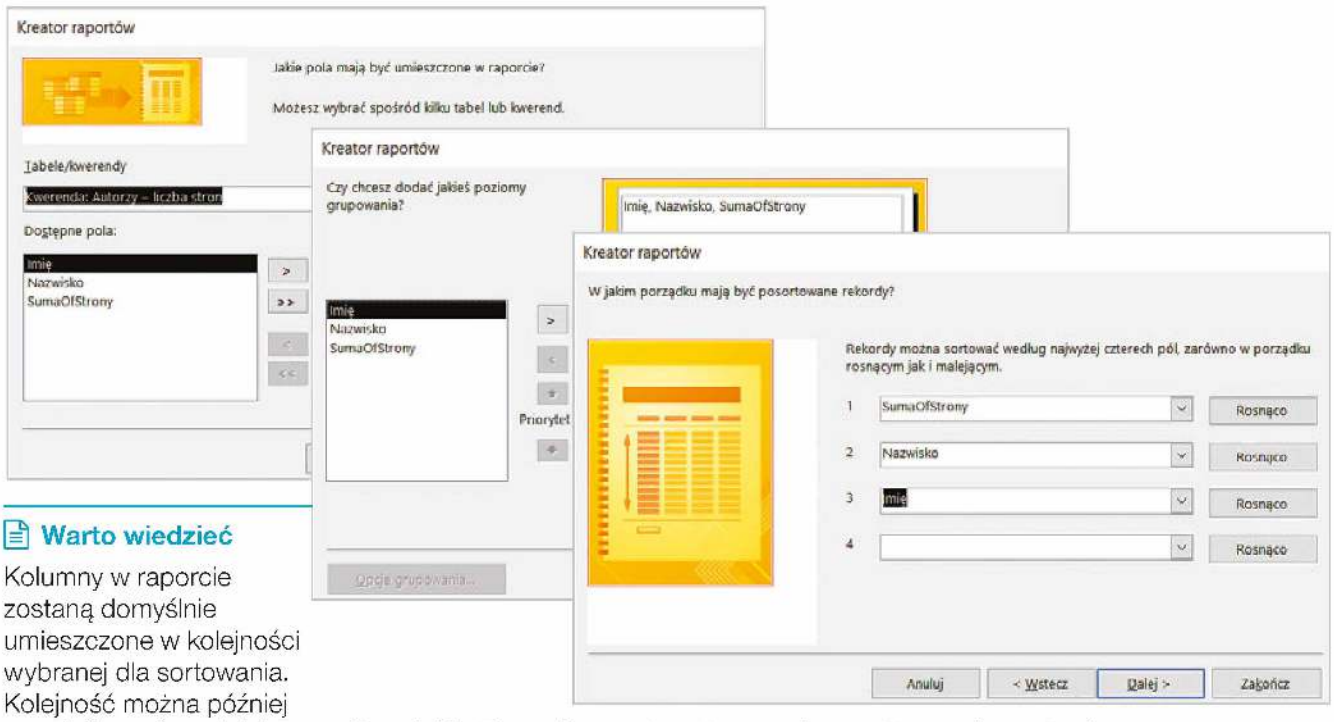
Załóżmy, że chcemy przygotować wydruk zawierający listę autorów wraz z łączną liczbą stron wszystkich ich artykułów. Lista ma być posortowana malejąco według liczby stron. Jeśli kilku autorów ma przygotować tę samą liczbę stron, powinni być w ramach grupy posortowani alfabetycznie według nazwisk, a następnie według imion.

Warto wiedzieć

Dzięki określeniu poziomów grupowania można grupować informacje w raporcie według wartości wybranych atrybutów.

Zauważ, że w bazie znajduje się już kwerenda, która zwraca potrzebne nam dane. Do utworzenia prostego raportu możemy wykorzystać **Kreator raportów** (karta **Tworzenie**). W kreatorze (rys. 4.19) wybieramy kolejno:

- obiekt, na którego podstawie zostanie przygotowany raport – np. utworzoną wcześniej kwerendę,
- pole, według którego chcielibyśmy ustalić poziomy grupowania,
- pola, według których dane zostaną posortowane: `SumaOfStrony` (malejąco), `Nazwisko` (rosnąco) i `Imię` (rosnąco).



Warto wiedzieć

Kolumny w raporcie zostaną domyślnie umieszczone w kolejności wybranej dla sortowania. Kolejność można później zmodyfikować w widoku projektu formularza.

Rys. 4.19. Początkowe etapy tworzenia raportu z wykorzystaniem **Kreatora raportów**

W kolejnych krokach możemy wybrać typ raportu. W naszym przypadku dobrze się sprawdzi układ tabelaryczny (rys. 4.20), ponieważ zależy nam na odpowiednio posortowanej liście osób. Lista będzie zawierała tylko trzy kolumny, dlatego zostawimy układ pionowy.


Autorzy – liczba stron

SumaOfStrony	Nazwisko	Imię
3	Walczuk	Paweł
3	Wysocka	Gabriela
4	Babicz	Eliza
4	Krynicki	Paweł
4	Makulska	Eliza
4	Waleczny	Piotr
5	Popławska	Marta
6	Bobrzak	Elżbieta
10	Moczulski	Kondrat
13	Kowalski	Jan

czwartek, 5 grudnia 2019 Strona 1 z 1

Rys. 4.20. Przykładowy podgląd wydruku raportu

Po zakończeniu pracy kreatora warto zobaczyć, jak wygląda raport, i zastanowić się nad wprowadzeniem ewentualnych zmian.

Dobra rada

Jeśli tabela wykorzystywana w raporcie ma wiele kolumn, rozważ poziomy układ raportu.

Ćwiczenie 9

Otwórz plik *baza_blog_9.accdb*, utworzony w poprzednim ćwiczeniu, i przygotuj raport na podstawie kwerendy *Autorzy według liczby stron artykułów*. Raport powinien zawierać imię, nazwisko oraz łączną liczbę stron artykułów napisanych lub zadeklarowanych przez danego autora. Plik zapisz pod nazwą *baza_blog_10.accdb*.

Dostosowanie raportu

W celu zwiększenia czytelności raportu możemy w **Widoku projektu** zmienić automatycznie utworzony tekst etykiety `SumaOfStrony` na `Liczba stron`. Możemy też zmienić kolejność wyświetlanych kolumn – na pierwszym miejscu ustawić imię, potem nazwisko, a na końcu liczbę stron.

Jeśli chcemy zamienić miejscami dwie kolumny raportu, należy w **Widoku projektu** przenieść zarówno etykiety kolumn w sekcji **Nagłówek strony**, jak i same kolumny w sekcji **Szczegóły strony**.

Dobra rada

W widoku projektu raportu możesz zaznaczyć wiele obiektów i przenieść je jednocześnie.

Drukowanie raportu i eksport do pliku

Po dopracowaniu raportu możemy przełączyć się do podglądu wydruku, wybierając: **Narzędzia główne** → **Widoki** → **Widok** → **Podgląd wydruku**. Jeśli chcemy wydrukować raport, wybieramy **Podgląd wydruku** → **Drukowanie** → **Drukuj**.

Raport możemy również wyeksportować do formatów XLSX, PDF lub TXT, korzystając z odpowiednich przycisków na karcie **Podgląd wydruku** w sekcji **Dane**. Aby wrócić do pracy z raportem, należy wybrać przycisk **Zamknij podgląd wydruku**.

Ćwiczenie 10

Otwórz plik *baza_blog_10.accdb*, utworzony w poprzednim ćwiczeniu. Wyeksportuj przygotowany przez siebie raport do pliku PDF.

Podsumowanie

- Bazą danych nazywamy zbiór powiązanych ze sobą informacji dotyczących pewnej dziedziny życia i zorganizowanych w taki sposób, aby łatwo było je przeszukiwać i na ich podstawie wyciągać wnioski oraz podejmować decyzje.
- Relacyjna baza danych w programie Microsoft Access składa się z tabel połączonych relacjami, kwerend, formularzy oraz raportów.
- W tabeli bazy danych przechowywane są informacje o jednym typie obiektów.
- Kolumny tabeli, zwane atrybutami, stanowią zbiór cech opisujących dany typ obiektu.
- Każdy wiersz tabeli, zwany rekordem, opisuje jeden obiekt danego typu poprzez określenie wartości poszczególnych jego atrybutów.
- Relacje opisują powiązania między danymi gromadzonymi w poszczególnych tabelach. Do tworzenia tych powiązań służą klucze główne oraz obce.
- Kwerendy pozwalają wyszukiwać w bazie danych informacje spełniające określone kryteria.
- Raporty umożliwiają wydrukowanie znalezionych danych w czytelnej, łatwej do zrozumienia formie.
- Formularze ułatwiają wprowadzanie oraz modyfikowanie danych w bazie.

Zadania

* **1** W bazie danych otrzymanej od nauczyciela (np. w pliku *baza_blog_z1.accdb*) przefiltruj i posortuj tabelę `Artykuły` tak, aby wyświetlić tylko nieoddane artykuły przeznaczone do działu *Nowe technologie* w kolejności, w jakiej powinny zostać dostarczone.

* **2** W bazie danych otrzymanej od nauczyciela (np. w pliku *baza_blog_z2.accdb*) przy użyciu kreatora stwórz na podstawie kwerendy `Nieoddane artykuły` raport zawierający listę niedostarczonych artykułów. Raport powinien zawierać: tytuł artykułu, nazwę działu, termin oddania, nazwisko autora. Lista powinna zostać posortowana według planowanej daty dostarczenia, a następnie – tytułu artykułu. Gotowy raport wyeksportuj do pliku PDF.

* **3** Stwórz krótką notatkę tekstową na temat innych niż relacyjne rodzajów baz danych.

* **4** W bazie danych otrzymanej od nauczyciela (np. w pliku *baza_blog_z4.accdb*) utwórz formularz pozwalający łatwo zarządzać danymi autorów artykułów.

* **5** Zaproponuj listę atrybutów tabeli zawierającej informacje o:
   a. uczniach danej szkoły,
   b. artykułach w magazynie ogrodniczym,
   c. kołach zainteresowań prowadzonych przez dom kultury.

* **6** W krótkiej notatce opisz, w jakiej sytuacji lepiej użyć poziomego układu strony podczas tworzenia nowego raportu.

** 7 W bazie danych otrzymanej od nauczyciela (np. w pliku *baza_blog_z7.accdb*) dla pól `Imię` i `Nazwisko` ustal maskę wprowadzania, która będzie wymuszała podanie imienia i nazwiska od wielkiej litery.

** 8 W bazie danych otrzymanej od nauczyciela (np. w pliku *baza_blog_z8.accdb*) dodaj w tabeli `Autorzy` atrybuty pozwalające przechowywać adres korespondencyjny autora, czyli: ulicę, numer domu, numer lokalu, kod pocztowy. Przyporządkuj właściwy typ danych do każdego z tworzonych atrybutów. Dla kodu pocztowego określ maskę wprowadzania.

** 9 W bazie danych otrzymanej od nauczyciela (np. w pliku *baza_blog_z9.accdb*) stwórz kwerendę wyświetlającą informacje na temat liczby artykułów, do których napisania zobowiązali się poszczególni autorzy.

** 10 W bazie danych otrzymanej od nauczyciela (np. w pliku *baza_blog_z10.accdb*) zmodyfikuj maskę wprowadzania adresu e-mail w taki sposób, aby można było wprowadzić tylko poprawny adres.

*** 11 Zaprojektuj bazę danych, która pozwoliłaby samorządowi szkolnemu gromadzić informacje na temat uczniów gotowych pomagać kolegom w nauce przed maturą. Baza powinna zawierać dane kontaktowe wolontariuszy oraz umożliwiać odnajdywanie ich ze względu na przedmioty, w których mogą pomóc. Uwzględnij to, że jeden wolontariusz może pomagać w wielu przedmiotach oraz że dla jednego przedmiotu pomoc może oferować wielu wolontariuszy.

*** 12 Zaproponuj, jak zmodyfikować bazę danych autorów bloga opisaną w podręczniku tak, aby można w niej było gromadzić dodatkowo informacje na temat nie tylko autorów, lecz także innych osób współpracujących przy tworzeniu bloga – np. grafików, animatorów, redaktorów itd.

*** 13 W bazie danych otrzymanej od nauczyciela (np. w pliku *baza_blog_z13.accdb*) dodaj nową tabelę służącą do przechowywania informacji na temat działów, w których są publikowane artykuły. Wprowadź w niej nazwy działów oraz ich opisy na podstawie danych gromadzonych dotychczas w kolumnie `Dział` tabeli `Artykuły`. Następnie dodaj nowe pole `IdDziału` w tabeli `Artykuły`. Za pomocą kreatora odnośników stwórz powiązanie między tabelami `Artykuły` oraz `Działy`. Przyporządkuj artykuły do działów, których nazwy zostały umieszczone w tabeli `Działy`. Następnie usuń kolumnę `Dział` z tabeli `Artykuły`.

*** 14 W bazie danych otrzymanej od nauczyciela (np. w pliku *baza_blog_z14.accdb*) stwórz kwerendę wyświetlającą liczbę artykułów opublikowanych na blogu w danym roku. Wyświetl dane z lat 2017–2020.

Arkusz kalkulacyjny i bazy danych

Wyobraź sobie, że...



...jesteś dietetykiem w ośrodku sportowym. Do układania planów żywieniowych wykorzystujesz zaprojektowaną przez siebie **bazę danych**. Dzięki temu możesz gromadzić, a następnie wyszukiwać przepisy na posiłki skomponowane z określonych produktów i składników odżywczych. Dane przechowujesz w **tabelach**, a nowe informacje wprowadzasz z wykorzystaniem **formularzy**. Dzięki **raportom** przygotowujesz przepisy uwzględniające różne kryteria, takie jak „płeć", „dyscyplina", „BMI" oraz „alergie".



...działasz jako wolontariusz w organizacji humanitarnej. Twoim zadaniem jest koordynowanie zbiórki żywności oraz pieniędzy na zakup narzędzi i nasion dla lokalnej społeczności w Afryce. Masz już arkusz kalkulacyjny z danymi darczyńców, datami i sposobem przekazania darów oraz kwotami pieniędzy i adresami hurtowni rolniczych. Analizę tych danych ułatwiasz sobie poprzez ich **filtrowanie** i **sortowanie**. Po zakończeniu akcji umieszczasz na stronie internetowej organizacji sprawozdanie uzupełnione odpowiednimi **wykresami**.

...założyłaś własną agencję marketingową i pomagasz klientom biznesowym w prowadzeniu kampanii reklamowych. Wiesz, jak ważne w twojej działalności są aktualne statystyki, dlatego regularnie **importujesz z sieci** tabele z danymi demograficznymi, pokazujące strukturę wiekową potencjalnych klientów. Przymierzasz się także do kampanii mailingowej, w czym pomoże ci umiejętność personalizowania treści oraz posługiwania się **korespondencją seryjną**.

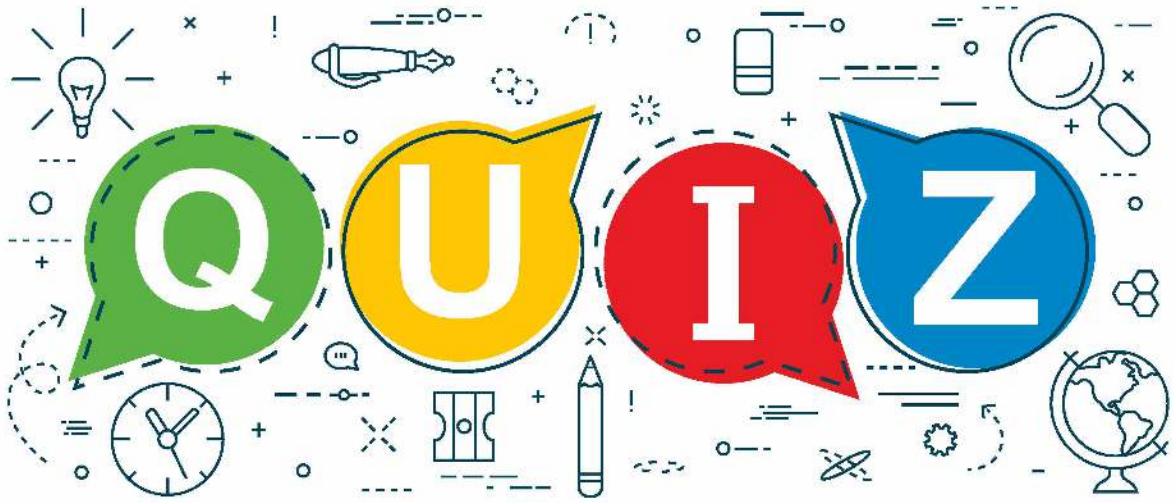




...pracujesz w dziale kontrolingu dużego przedsiębiorstwa. Twoim zadaniem jest gromadzenie i przetwarzanie danych kluczowych dla funkcjonowania firmy, takich jak wydajność produkcyjna, przewidywane koszty decyzji strategicznych czy polityki kadrowej. Dane pobierasz z **kwerend** i **raportów** bazy danych przedsiębiorstwa. Następnie przetwarzasz je w arkuszu kalkulacyjnym, korzystając z możliwości, jakie dają **tabele przestawne**.

Współdziałanie aplikacji

Wykorzystując bezpłatne aplikacje, służące m.in. do projektowania grafiki, pisania tekstów i wykonywania obliczeń, możemy stworzyć ciekawe projekty – np. quizy. Są one znakomitym narzędziem edukacyjnym. Dzięki quizom możecie zweryfikować swoją wiedzę na dany temat lub przygotować się do sprawdzianu.



Cel projektu

Utworzenie quizu dotyczącego wybranej tematyki oraz certyfikatów dla osób, które rozwiążą quiz. Zadania powinny być wykonane z wykorzystaniem aplikacji użytkowych.

To trzeba wiedzieć

Przykładem programu, w którym można stworzyć quiz, jest arkusz kalkulacyjny. Podczas pracy warto wykorzystać poniższe funkcje i narzędzia.

`JEŻELI` – funkcja warunkowa, dzięki której można m.in. wstawić komentarz dla rozwiązującego quiz, w zależności od udzielonej odpowiedzi, np. DOBRZE lub ŹLE. Takie komentarze mogą być następnie zliczane, aby pokazać końcowy wynik.

`ILE.NIEPUSTYCH` / `LICZ.PUSTE` – te funkcje pozwolą policzyć, ile pytań jest w quizie. Można to wykorzystać na różne sposoby, np. prezentować na bieżąco, ilu odpowiedzi już udzielono oraz ile pytań pozostało do końca.

`LICZ.JEŻELI` – dzięki tej funkcji można zliczać poprawne lub błędne odpowiedzi.

`POZYCJA` – funkcja ta pozwala na utworzenie rankingu uczestników quizu.

Power Query – narzędzie programu Microsoft Excel do pobierania danych z różnych źródeł, przetwarzania ich i przygotowywania do dalszej analizy. W starszych wersjach programu Excel funkcjonuje jako darmowy dodatek.

Propozycje tematów

1. Powtórka słówek lub gramatyki z języka obcego
2. Mistrzowie sportu
3. Najważniejsze wydarzenia z historii XX w.
4. Stolice świata
5. Mistrz ortografii

Etapy projektu

1. Wybierzcie temat zadania oraz ustalcie reguły tworzenia i rozwiązywania quizu. Przydzielcie role – powinny się wśród nich znaleźć: redaktor, projektant, twórca rozwiązania oraz tester.
2. Przygotujcie pytania i odpowiedzi do quizu. Skorzystajcie z dostępnych narzędzi współpracy online, aby ocenić propozycje pytań i wspólnie wybrać najlepsze. Zaplanujcie komunikat, który poinformuje użytkownika o konieczności przesłania pliku z uzupełnionymi odpowiedziami pod wskazany adres (np. e-mail lub katalog na dysku sieciowym).
3. W programie graficznym stwórzcie projekt graficzny:
   a. quizu – powinien być przyjazny i czytelny dla użytkownika oraz musi uwzględniać specyfikę arkusza kalkulacyjnego, w którym będzie przeprowadzany,
   b. certyfikatu dla uczestnika quizu – powinien być imienny, musi informować o wyniku i miejscu uczestnika w rankingu wszystkich osób biorących udział w quizie.
4. Wykonajcie quiz w skoroszycie arkusza kalkulacyjnego:
   a. wprowadźcie pytania do quizu oraz sformatujcie arkusz tak, jak zakładał to projekt graficzny. Możecie wykorzystać narzędzie **Poprawność danych**;
   b. dodajcie pole do wpisania imienia i nazwiska użytkownika;
   c. w dodatkowym arkuszu tego samego skoroszytu wprowadźcie poprawne odpowiedzi do pytań oraz mechanizm zliczania punktów wraz z punktacją uzyskaną przez uczestnika. Zbierzcie też w jedną tabelę dane identyfikujące uczestnika, m.in. jego imię i nazwisko, liczbę zdobytych przez niego punktów oraz nazwę quizu;
   d. ukryjcie arkusz z odpowiedziami i zabezpieczcie silnym hasłem cały skoroszyt przed możliwością jego modyfikacji przez uczestnika quizu.
5. Wykonajcie certyfikat uczestnika quizu:
   a. utwórzcie plik edytora tekstu zgodnie z przyjętym projektem graficznym;
   b. przygotujcie dodatkowy skoroszyt, który będzie bazą danych zawierającą wyniki quizów wszystkich uczestników. Dane do pliku można pobrać z wykorzystaniem narzędzia **Power Query**. Umożliwia ono importowanie do arkusza kalkulacyjnego zawartości innych plików, które znajdują się w tym samym katalogu. W tym celu wystarczy wybrać: **Dane → Pobieranie i przekształcanie → Nowe zapytanie → Z pliku → Z folderu**, a następnie wybrać ścieżkę do katalogu z zebranymi odpowiedziami do quizu;
   c. w pliku z certyfikatem zastosujcie mechanizmy korespondencji seryjnej, aby spersonalizować informacje dla uczestników. Wykorzystajcie bazę danych z danymi uczestników.
6. Przetestujcie rozwiązanie i poprawcie ewentualne błędy.
7. Udostępnijcie quiz wszystkim uczestnikom.
8. Zbierzcie wszystkie pliki od uczestników i umieśćcie je w jednym katalogu. Uruchomcie plik z bazą uczestników i odświeżcie połączenie utworzone wcześniej przez **Power Query**. Otwórzcie plik z certyfikatem, jako źródło danych wskażcie utworzoną wcześniej bazę danych w arkuszu kalkulacyjnym i wygenerujcie certyfikaty z wynikami.

Przykładowe narzędzia

Edytor tekstu, np. Microsoft Word, LibreOffice Writer, Dokumenty Google.
Edytor grafiki, np. Inkscape, Gravit Designer.
Arkusz kalkulacyjny, np. Microsoft Excel, LibreOffice Calc, Arkusze Google.



2 Algorytmika i programowanie w języku C++

5. Od problemu do programu
6. Wyszukujemy i sumujemy
7. Binarny system liczbowy
8. Czy ta liczba jest pierwsza?
9. Algorytm Euklidesa i działania na ułamkach

Algorytmika i programowanie w języku C++

Technika i matematyka są związane ze sobą od wieków. Urządzenia ułatwiające obliczenia i przetwarzające ich wyniki zawsze były rezultatem wykorzystania algorytmów matematycznych i rozwiązań technicznych dostępnych w danej epoce. Współcześnie algorytmy zapisuje się w językach programowania, które pozwalają tworzyć oprogramowanie.



1800 r. p.n.e.

Około 1800 r. p.n.e. starożytni Egipcjanie odkryli algorytm mnożenia – nazwany potem **mnożeniem egipskim** – w którym używano wyłącznie operacji podwajania i dodawania.

VII w. p.n.e.



Grecki poeta Archiloch w jednym z dzieł wspomniał o przyrządzie do szyfrowania zwanym *skutálē*, używanym w Sparcie od VII w. p.n.e. Nadawca nawijał na laskę pasek pergaminu i pisał tekst na stykających się brzegach. Adresat posiadał identyczną laskę, umożliwiającą odczytanie wiadomości. Dziś ten **szyfr przestawieniowy** nazywa się metodą płotu.

300 r. p.n.e.

Jednym z najstarszych algorytmów obliczeniowych jest **algorytm Euklidesa**, czyli metoda wyznaczania największego wspólnego dzielnika dwóch liczb, opisana przez greckiego matematyka Euklidesa w dziele *Elementy*. Pierwotnie algorytm ten służył do szukania odcinka, który byłby wspólną miarą dla dwóch dłuższych odcinków.



I w. p.n.e.

Pierwszym opisanym **szyfrem podstawieniowym** był prawdopodobnie szyfr z czasów wojen galijskich, stosowany w korespondencji przez Juliusza Cezara. Szyfr polega na cyklicznym przesuwaniu liter alfabetu: za literę A podstawiano literę D, za B – E itd., a za Z – C.



I w. n.e.

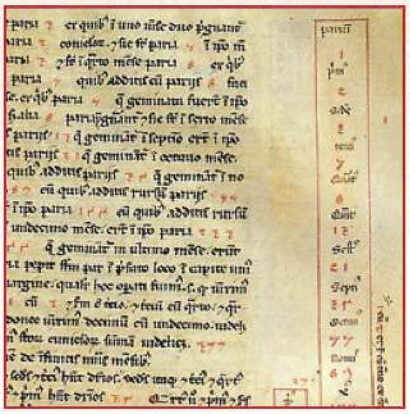


Heron z Aleksandrii w dziele *Metrica* opisał efektywną metodę iteracyjnego wyznaczania przybliżeń pierwiastka kwadratowego z użyciem operacji dodawania, mnożenia i dzielenia. Dziś tę metodę nazywa się **algorytmem Herona**.

IX w.



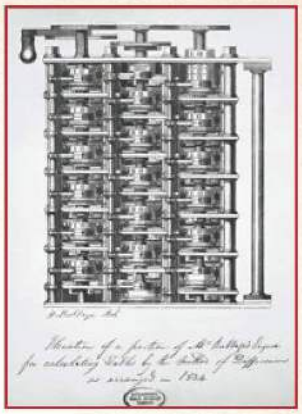
Muhammad Ibn Musa Al-Chuwarizmi, bibliotekarz Domu Mądrości w Bagdadzie, opisał efektywne sposoby pisemnego wykonywania obliczeń z użyciem systemu dziesiętnego. Od łacińskiej wersji nazwiska tego uczonego pochodzi słowo **algorytm**.



Leonardo Fibonacci w dziele *Liber abaci* z 1202 r. rozważał problem rozmnażania się populacji królików. W rozwiązaniu pojawia się ciąg liczb:
1, 1, 2, 3, 5, 8, 13, 21, ...,
w którym: 2 = 1 + 1, 3 = 1 + 2, 5 = 2 + 3, 8 = 5 + 3 itd. Ciąg ten opisuje **zależność rekurencyjna**: następna liczba jest sumą dwóch poprzednich. Kolejne liczby Fibonacci wypisał na prawym marginesie rękopisu.

W II połowie XVII w. Gottfried Wilhelm Leibniz opisał **dwójkowy system liczbowy**, w którym każdą liczbę można zapisać za pomocą kombinacji cyfr 0 i 1. Obecnie taki kod zero-jedynkowy (binarny) stanowi podstawę działania komputerów.





W 1837 r. Charles Babbage ogłosił koncepcję parowej **maszyny analitycznej**. Miała się ona składać z magazynu danych i pośrednich wyników obliczeń, młyna (jednostki liczącej) oraz mechanizmu sterującego, programowanego z użyciem kart perforowanych. Projekt nie doczekał się realizacji z powodu zbyt niskiego poziomu ówczesnej techniki.

1202 r.

1614 r.



John Napier w 1614 r. opisał metodę logarytmów, czyli metodę mnożenia liczb za pomocą dodawania innych liczb, które można zapisać w tablicy. Napier wynalazł przyrząd, nazywany kostkami Napiera, który uznaje się za **prototyp komputera analogowego**.

1679 r.

1804 r.

Na początku XIX w. Joseph Marie Jacquard zbudował **programowalną maszynę** tkacką, która automatycznie wykonywała wzór tkaniny dzięki kartom perforowanym. Aż do lat 80. XX w. karty i taśmy perforowane były nośnikami danych dla komputerów.



1837 r.

1843 r.



Za **pierwszego programistę** niektórzy uważają Adę Lovelace. Jej notatki dotyczące maszyny analitycznej, wydane w 1843 r., zawierały diagram algorytmu napisany z zamiarem jego wykonania na fizycznej maszynie. W pracach Lovelace pojawiają się koncepcje: zmiennych, instrukcji warunkowej, pętli i funkcji.

5. Od problemu do programu

Być może tytuł tego tematu wydaje ci się nieco tajemniczy. W tym krótkim haśle zawiera się jednak główna idea całego rozdziału: programowanie pozwala urzeczywistnić pomysły, które mogą rozwiązać konkretne problemy współczesnego świata i zmienić życie ludzi na lepsze. Nawet jeśli nie zamierzasz w przyszłości zostać programistą, warto poznać elementy sztuki programowania. W tym temacie nauczysz się podstaw informatycznego rozwiązywania problemów z użyciem języka C++. Dzięki temu lepiej zrozumiesz otaczający cię świat cyfrowy, a zdobyte umiejętności przydadzą ci się w przyszłej współpracy z programistami.

Cele lekcji

- Zrozumiesz etapy myślenia informatycznego.
- Poznasz przykłady algorytmów wykorzystywanych w życiu codziennym.
- Dowiesz się, jak poprawnie zapisać algorytm i czym jest lista kroków.
- Nauczysz się posługiwać zmiennymi oraz stosować konstrukcje programistyczne: instrukcje wejścia i wyjścia, wyrażenia arytmetyczne i logiczne oraz instrukcję warunkową.
- Napiszesz w języku programowania C++ proste programy rozwiązujące problemy z różnych dziedzin.

5.1. Algorytm, czyli rozwiązanie problemu

Każde urządzenie cyfrowe, z którego korzystamy, zawiera oprócz systemu operacyjnego pewne oprogramowanie użytkowe. Zanim jednak powstanie program, musi się pojawić potrzeba jego stworzenia. Okoliczność, która tworzy tę potrzebę, będziemy nazywać problemem lub sytuacją problemową.

Aby stworzyć program komputerowy rozwiązujący konkretny problem, trzeba najpierw zaprojektować samo rozwiązanie problemu. Musi ono mieć formę **algorytmu** (ang. *algorithm*). W tym i kolejnych tematach będziemy postępować według schematu: problem – algorytm – program komputerowy.

Algorytm to opis postępowania prowadzącego do rozwiązania problemu.

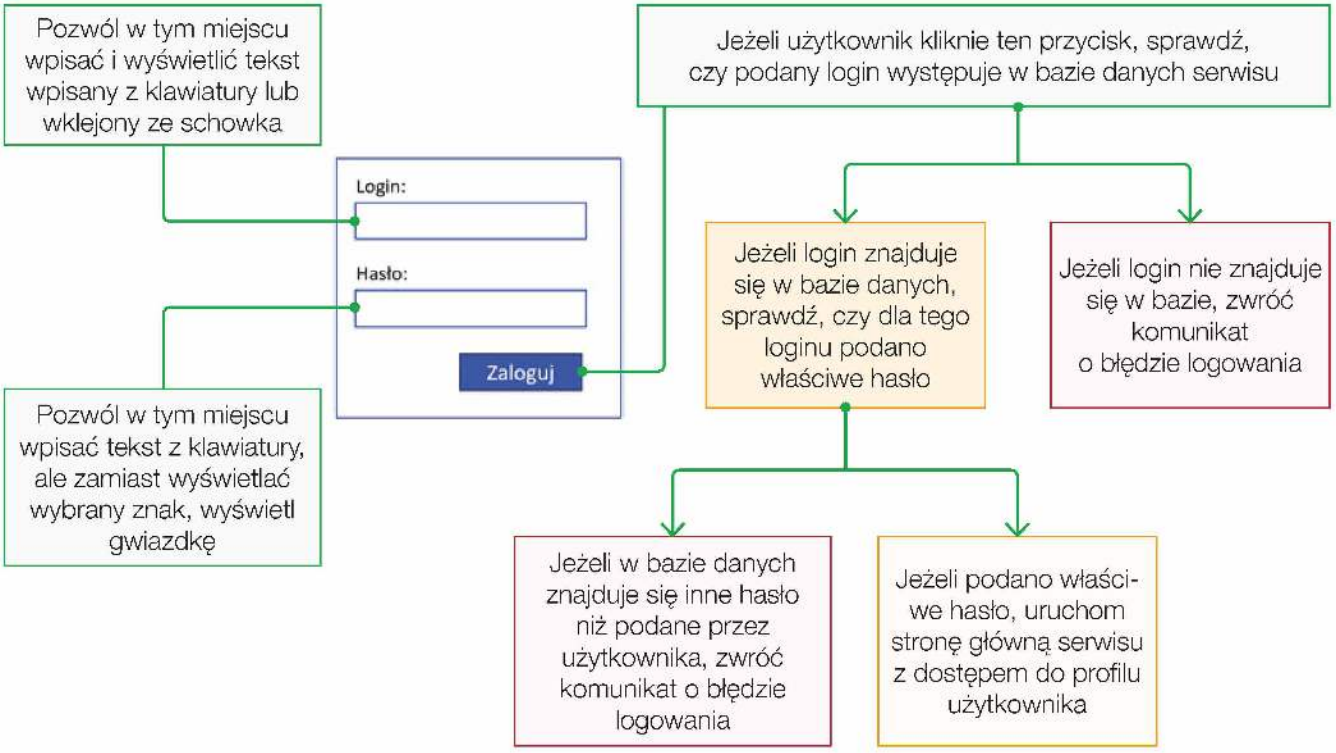
Ćwiczenie 1

Zastanów się, jak utworzyć silne hasło zabezpieczające konto internetowe. Przedstaw jeden ze sposobów konstruowania hasła w formie kolejnych kroków.

Omówimy teraz kilka przykładów problemów i pokażemy algorytmy ich rozwiązania.

Problem 1. Logowanie do serwisu internetowego

Zapewne nie raz zdarzyło ci się logować do serwisu internetowego. Zastanówmy się, jak mogłyby wyglądać wytyczne dla programisty takiego serwisu dotyczące okna logowania (rys. 5.1).



Rys. 5.1. Logowanie do serwisu internetowego – opis działania elementów

Stworzenie takiego mechanizmu wynikało z potrzeby zidentyfikowania (uwierzytelnienia) użytkownika. W tym celu sprawdza się poprawność loginu i hasła, które podał logujący się. Jeśli obie te wartości zgadzają się z danymi zapisanymi w bazie danych serwisu (serwera), to użytkownik może wejść na swoje konto w serwisie. Wystarczy jednak, że jedna z danych się nie zgadza, aby zwrócić komunikat o błędnym loginie lub haśle.

To jest właśnie najprostsza, opisowa wersja algorytmu logowania, którą następnie można by zaprogramować.

Warto wiedzieć

Hasła zazwyczaj są przechowywane na serwerze w formie zahaszowanej (czyli zaszyfrowanej specjalnym algorytmem). Podczas logowania podane hasło jest ponownie haszowane, a następnie sprawdza się jego zgodność z kodem haszowania zapisanym w bazie.

Ćwiczenie 2

Zaproponuj schemat objaśniający proces:

a. wypożyczania książki w szkolnej bibliotece,

b. zakładania konta poczty elektronicznej.

Problem 2. Jak zapleść warkocz klasyczny?

Wyobraź sobie, że ktoś zapytał cię, jak zapleść warkocz klasyczny. Jeśli potrafisz to zrobić, zapewne najłatwiej będzie ci pokazać to na przykładzie. W języku informatyki powiemy w takiej sytuacji, że zademonstrujesz działanie algorytmu zaplatania warkocza.

Nie zawsze jednak można taki pokaz przeprowadzić, dlatego czasem potrzebujemy uniwersalnego i jednoznacznego opisu rozwiązania zadania. Może on mieć formę np. listy kroków.

Lista kroków

Lista kroków to jeden ze sposobów precyzyjnego opisu postępowania. Składa się z zapisanych słownie kolejnych pojedynczych instrukcji, nazywanych krokami, które mogą się odwoływać do siebie nawzajem.

Lista kroków algorytmu zaplatania warkocza może wyglądać następująco:

1. `Rozdziel włosy na trzy pasma.`
2. `Dopóki nie spleciesz warkocza po końcówki włosów, wykonuj na zmianę krok 3 i krok 4.`
   3. `Przełóż lewe pasmo nad środkowym. Teraz to pasmo staje się środkowym.`
   4. `Przełóż prawe pasmo nad środkowym. Teraz to pasmo staje się środkowym.`

Warto wiedzieć

Ludzie od tysięcy lat zaplatają warkocze – przedstawiają je m.in. rysunki pochodzące ze starożytnego Egiptu. Dziś znanych jest wiele sposobów zaplatania warkocza. Wersja podstawowa to warkocz klasyczny (zdjęcie poniżej).



Zauważ, że kroki 3 i 4 są wcięte względem kroków 1 i 2. W ten sposób zaznaczamy, że zależą one od warunku podanego w kroku 2 (ostatnim kroku przed wcięciem). W naszym przypadku wykonywanie kroków 3 i 4 zależy od tego, czy dotychczas zapleciony warkocz sięga końcówek włosów. Zauważ ponadto, że tak podany opis:

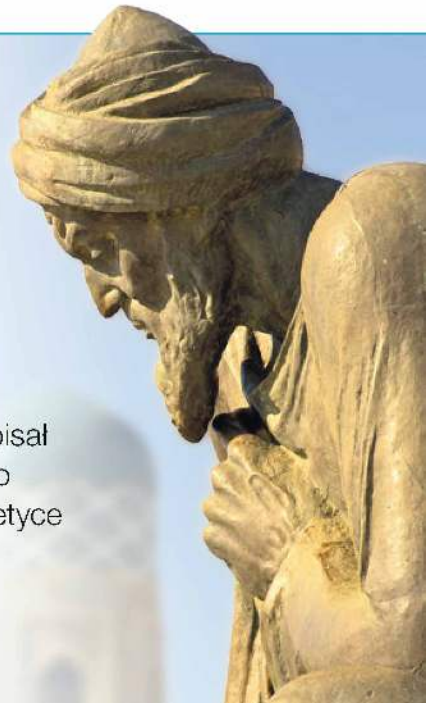
- zawiera informację o tym, na jakich danych wejściowych będziemy wykonywać operacje (w naszym przykładzie są to włosy),
- zwraca określony wynik (zapleciony warkocz),
- jest skończony – można go zrealizować w skończonej liczbie kroków (warunek zakończenia znajduje się w kroku 2),
- jest precyzyjny (pasma włosów mają swoje nazwy, czynności są jednoznacznie opisane),
- jest uniwersalny – można go zastosować do każdych włosów odpowiedniej długości.

Cechy poprawnego algorytmu

Są to **cechy poprawnego algorytmu**.

A to ciekawe

Skąd pochodzi słowo „algorytm”?



Słowo „algorytm” pochodzi od *Algorithmus* – łacińskiego przekładu ostatniego z członów nazwiska perskiego matematyka i astronoma, który był znany jako Muhammad Ibn Musa Al-Chuwarizmi. Żył on na przełomie VIII i IX w. n.e. i był bibliotekarzem Domu Mądrości w Bagdadzie. Zajmował się m.in. pozycyjnym systemem dziesiętnym zapisu liczb, znanym wcześniej wyłącznie w Indiach, i opisał efektywne sposoby pisemnego wykonywania obliczeń. W XII w. przetłumaczono jego dzieło na łacinę, co zapoczątkowało w Europie stopniowe odejście w arytmetyce handlowej od posługiwania się żetonami i stołami obliczeniowymi (abakami).

Ćwiczenie 3

a. W podanej instrukcji zaplatania warkocza wskaż te czynności, które można uznać za podstawowe (niepodzielne). Kiedy wykonywanie algorytmu powinno się zakończyć?

b. Dowiedz się, jak zawiązać tzw. węzeł ósemkowy. Zapisz algorytm w formie listy kroków.

Problem 3. Wyznaczanie pierwiastka kwadratowego

Dokładność przybliżeń stosowanych w komputerach odgrywa kluczową rolę w wielu dziedzinach, np. w medycynie, kosmonautyce, architekturze czy na rynkach finansowych.

Zapoznamy się z metodą znajdowania przybliżonej wartości pierwiastka kwadratowego przedstawioną przez Herona z Aleksandrii w I w. n.e. Używa się w niej wyłącznie prostych operacji arytmetycznych: dodawania oraz dzielenia i można ją zastosować do współczesnych obliczeń wykorzystujących wartość pierwiastka kwadratowego.

Opis algorytmu poprzedzimy stworzeniem tzw. **specyfikacji problemu**, czyli określeniem związku między danymi a wynikiem.

Specyfikacja problemu

Specyfikacja

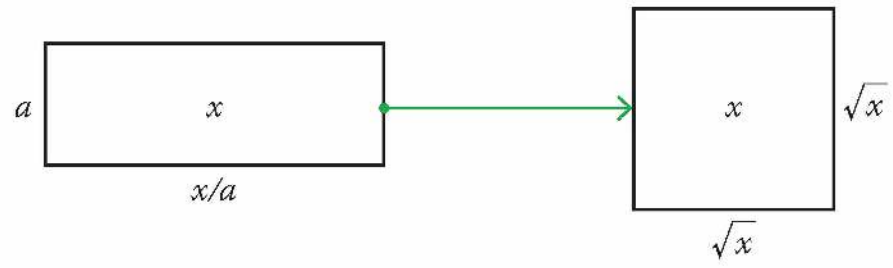
Dane: liczba x.

Wynik: przybliżenie $\sqrt{x}$ z dokładnością do 0,001.

Liczbę x możemy potraktować jako pole powierzchni prostokąta o długościach boków a oraz x/a (rys. 5.2). Metoda Herona polega na przemyślanej zmianie długości boku a (przy jednoczesnej zmianie długości drugiego boku prostokąta) tak, aby bez zmiany pola powierzchni prostokąta otrzymać figurę jak najbardziej zbliżoną do kwadratu. Po kilkukrotnym powtórzeniu tej procedury długość boku a przyjmie wartość równą w przybliżeniu $\sqrt{x}$.

Warto wiedzieć

Heron pisał dzieła o charakterze praktycznym, dziś powiedzielibyśmy – inżynierskie. Pokazywał na przykładach m.in. metody obliczania pola powierzchni działek rolnych oraz objętości brył.



Rys. 5.2. Interpretacja geometryczna szukania przybliżenia liczby $\sqrt{x}$

Modyfikacja długości boku a oznacza, że jeden z boków prostokąta skracamy, a drugi wydłużamy, aż różnica między nimi będzie wynosiła 0 albo będzie nie większa niż błąd podany w specyfikacji.

Do właściwej wartości a (przybliżenia $\sqrt{x}$) dochodzimy w kolejnych krokach, w których tworzymy kolejne prostokąty.

Warto wiedzieć

Prawdziwa jest zależność: jeśli $|a - x/a| < 0{,}001$, to $|\sqrt{x} - a| < 0{,}001$. Wynika to stąd, że a i x/a są przybliżeniami $\sqrt{x}$ z góry i z dołu (jeśli $a > \sqrt{x}$, to $x/a < \sqrt{x}$ i na odwrót). Jeśli odległość między a i x/a jest mniejsza od 0,001, to tym bardziej odległość a od $\sqrt{x}$ musi być mniejsza od 0,001.

Za długość boku a kolejnego prostokąta przyjmiemy średnią arytmetyczną długości boków poprzedniego prostokąta: $(a + x/a)/2$. Drugi z boków tego prostokąta wyznaczymy już jako nowy iloraz x/a dla zmienionego a. Długość a traktujemy jako zmienną, do której przypisaliśmy wartość $(a + x/a)/2$. Jeżeli różnica między długościami boków nowego prostokąta jest dostatecznie mała (w tym wypadku wynosi mniej niż 0,001), to a jest szukanym przybliżeniem. Jeżeli tak nie jest, tworzymy kolejny prostokąt o boku $(a + x/a)/2$ i polu x. I tak dalej.

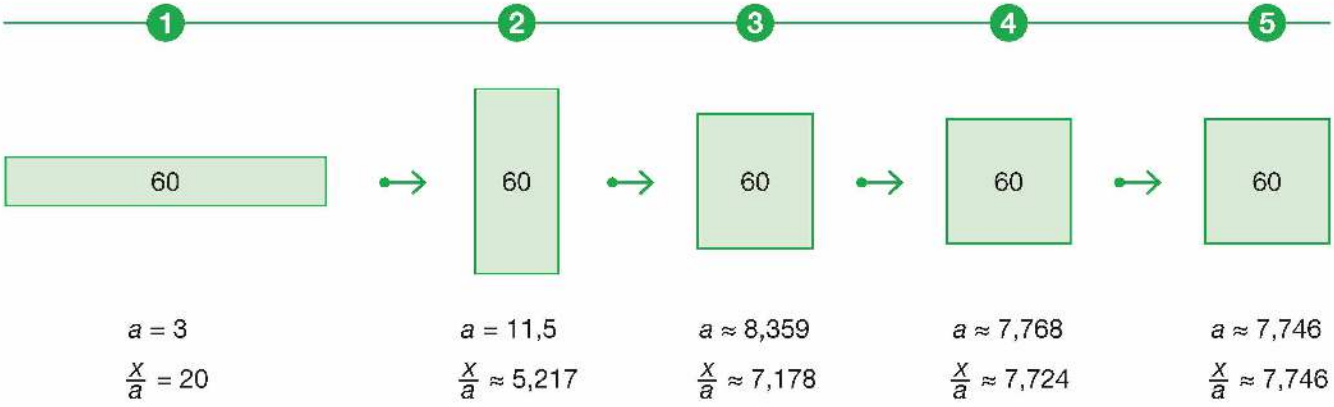
Skąd wziąć początkową wartość a? Może nią być dowolna liczba, np. 1. Jednak im bliższa pierwiastkowi z danej liczby jest pierwsza wartość a, tym szybciej algorytm wyznaczy przybliżenie pierwiastka.

Algorytm Herona

Zapiszmy listę kroków **algorytmu Herona**:

1. `Za a przyjmij dowolną liczbę większą od 0.`
2. `Dopóki wartość |a - x/a| jest większa niż 0,001, powtarzaj kroki` 3 `i` 4`.`
   3. `Przyjmij za a średnią arytmetyczną a i x/a.`
   4. `Oblicz różnicę a i x/a.`

Korzystając z listy kroków algorytmu Herona, obliczmy przybliżenie $\sqrt{60}$. Za wartość początkową a przyjmijmy 3. Na rysunku 5.3 pokazano kolejne przybliżenia kwadratu o boku długości $\sqrt{60} \approx 7{,}746$.



Rys. 5.3. Zastosowanie algorytmu Herona dla $x = 60$

Warto wiedzieć

Komputery przeznaczają znaczną część czasu pracy na obliczenia wykorzystujące operacje arytmetyczne. Są one potrzebne np. w grafice komputerowej lub analizie danych. Jedną z takich operacji jest obliczanie pierwiastka kwadratowego.

Zauważ, że na schemacie (rys. 5.3) etap 5 kończy działanie algorytmu, ponieważ udało się znaleźć przybliżenie z dokładnością do trzech miejsc po przecinku. Algorytm można by wykonywać dalej, jeśli zależałoby nam na większej precyzji. Można by go również zakończyć na etapie 4 – wówczas dla wartości $a \approx 7{,}768$ mamy $a - x/a \approx 0{,}044$. Jeśli kwadrat tej wartości jest akceptowalnym przybliżeniem, możemy przyjąć, że $\sqrt{60} \approx 7{,}768$.

Ćwiczenie 4

Opracuj plik arkusza kalkulacyjnego, który będzie obliczał przybliżenie pierwiastka kwadratowego zgodnie z algorytmem Herona.

Zapamiętaj

Algorytm to opis postępowania prowadzącego do rozwiązania problemu. Specyfikacja problemu algorytmicznego to opis danych (wraz z warunkami, jakie mają spełniać) oraz wyników i ich związku z danymi.

5.2. Od algorytmu do programu w języku C++

Zapis algorytmu w języku programowania nazywamy tekstem programu lub **kodem źródłowym** programu komputerowego. Na kod źródłowy programu, niezależnie od języka programowania, składają się dwie podstawowe konstrukcje programistyczne: instrukcje i funkcje.

Kod źródłowy programu komputerowego musi być przetłumaczony przez **translatory** na kod wykonywalny (maszynowy) właściwy dla procesora i systemu operacyjnego. Translatory, które tłumaczą od razu cały kod, nazywamy **kompilatorami** (ang. *compilers*). Natomiast translatory, w których tłumaczenie odbywa się instrukcja po instrukcji, nazywamy **interpreterami** (ang. *interpreters*). Przykładem języka kompilowanego jest **język C++**, a interpretowanego – **język Python**.

Instrukcja to polecenie, które kompilator lub interpreter potrafi zrozumieć i wykonać. Przyjmuje ono formę zdania rozkazującego (zawierającego zwroty typu „Powiedz", „Zatrzymaj się", „Wczytaj", „Wyświetl"). Pojedyncze instrukcje składa się zazwyczaj w większą całość nazywaną **funkcją**.

Podamy teraz kilka podstawowych zasad zapisywania kodu w języku C++. Język ten, jak każdy inny język programowania, opiera się na sformalizowanych regułach tworzenia kodu. Zawiera również listę zastrzeżonych słów, które są nazywane **słowami kluczowymi**. Można ich używać tylko w konkretnym celu, który określili twórcy języka.

- Kod źródłowy
- Translator
- Kompilator
- Interpreter
- Język C++, język Python
- Instrukcja
- Funkcja
- Słowa kluczowe języka programowania

A to ciekawe

Ranking języków programowania

Istnieje wiele internetowych rankingów popularności języków programowania. Część z nich mierzy liczbę zapytań pozwalających odnaleźć w internecie kursy programowania w danym języku. Inne rankingi biorą pod uwagę serwisy społecznościowe dla programistów i mierzą liczbę wątków dotyczących danego języka. W czołówce najpopularniejszych języków programowania ostatnich lat znajdują się języki: Java, JavaScript, Pyhton, C/C++, C#, PHP.



Zapisywanie programów w języku C++

Kody źródłowe języka C++ zapisujemy w edytorze kodu połączonym z kompilatorem. Powszechnie korzysta się ze zintegrowanych środowisk programistycznych (IDE). Jednym z nich jest Code::Blocks, który automatycznie koloruje elementy składniowe kodu (np. słowa kluczowe języka, komentarze i liczby).



Przykładowy kod źródłowy programu w języku C++

To polecenie dołącza bibliotekę z obsługą standardowego wejścia i wyjścia. Dzięki niemu program będzie mógł m.in. wczytywać dane z klawiatury i wypisywać wyniki na ekranie monitora

```
1. #include <iostream>
2.
3. using namespace std;
4.
5. int main() //nagłówek funkcji głównej
6. {
7.     cout << "Hello world!" << endl;
8.     return 0;
9. }
```

Ten zapis pozwala programowi wykorzystać tzw. standardową przestrzeń nazw. Dzięki temu instrukcje w kodzie można zapisywać w wygodniejszej krótszej wersji, np. `cout` zamiast `std::cout`

Funkcja `main` to główny blok kodu każdego programu. Początek i koniec bloku oznacza się nawiasami klamrowymi

Zasady tworzenia kodu w języku C++

Wielkość liter w kodzie ma znaczenie – zmienne `a` i `A` mogą przechowywać różne wartości.

Nie używamy polskich znaków w nazwach tworzonych zmiennych i funkcji.

Każdą instrukcję kończymy średnikiem.

Tekst po znakach `//` jest komentarzem – można używać w nim polskich znaków.

Warto stosować w kodzie wcięcia i puste wiersze. Zwiększają one czytelność programu.

Spacje, wcięcia i inne tzw. białe znaki nie mają znaczenia w procesie kompilacji.

Program *Hello world* – zapisywanie programu w środowisku Code::Blocks

Na poprzedniej stronie zapisano tekst programu, którego jedynym zadaniem jest wypisanie na ekranie tekstu „Hello world!". Jest to tradycyjny przykład, od którego często rozpoczyna się naukę programowania.

Z lewej strony kodu znajdują się liczby, którymi oznaczono kolejne linie (wiersze) kodu. Wiersze będziemy numerować, żeby łatwiej było omawiać poszczególne zapisy.

Wiersz 7 kodu odpowiada za wypisanie na ekranie tekstu „Hello world!". Instrukcja **cout** to tak zwana **instrukcja wyjścia** (ang. *output*), której działanie polega na wypisaniu tekstu na ekran. Bezpośrednio za tą instrukcją znajduje się operator <<, którego zadaniem jest przekazywanie danych. W tym przypadku operator wykorzystano do przekazania występującego za nim tekstu (podanego w cudzysłowie) do polecenia **cout**. Za ciągiem znaków znajduje się kolejny operator <<. Informuje on kompilator o istnieniu kolejnego parametru polecenia **cout**. Jest nim specjalne słowo **endl**, które służy do wstawiania znaku końca wiersza. W efekcie po wypisaniu tekstu na ekranie kursor zostanie ustawiony na początku nowego wiersza.

Instrukcja **return** 0; z wiersza 8 informuje system operacyjny o poprawnym zakończeniu działania programu. Jest ona zawsze ostatnią instrukcją **funkcji** main, czyli funkcji głównej programu.

Przedstawione na sąsiedniej stronie instrukcje, poza tymi z linii 7–8, stanowią **podstawową strukturę kodu źródłowego w języku C++**:

Warto wiedzieć

Polecenie **cout** pochodzi od angielskiego wyrażenia *console output*, które oznacza wypisanie tekstu na konsolę (czyli standardowe wyjście). Wymawia się je tak jak *see out*.

Instrukcja wyjścia

Warto wiedzieć

Zapis **endl** pochodzi od angielskiego wyrażenia *end line*, które oznacza koniec linii (wiersza) tekstu.

Funkcja main

Podstawowa struktura kodu źródłowego w języku C++

```
1.  #include <iostream>
2.
3.  using namespace std;
4.
5.  int main()
6.  {
7.
8.  }
```

Warto wiedzieć

Nazwa **iostream** pochodzi od angielskiego wyrażenia *input-output stream*, które oznacza strumień danych wejścia i wyjścia.

Kody źródłowe programów w języku C++ będziemy zapisywać za pomocą edytora środowiska Code::Blocks. Pliki źródłowe zawierające kod w języku C++ są plikami tekstowymi z rozszerzeniem *cpp*.

Aby rozpocząć tworzenie kodu w tym środowisku, wybieramy z menu kolejno **File → New → Empty file**, a następnie, jeszcze przed edycją kodu, zapisujemy nowo utworzony plik na dysku (**File → Save file as...**), np. pod nazwą *hello_world.cpp*.

Program domyślnie zapisuje pliki z rozszerzeniem *c* (właściwym dla kodu w języku C), a więc należy zmienić rozszerzenie na *cpp*.

Dobra rada

Aktualną wersję programu Code::Blocks możesz pobrać ze strony http://www.codeblocks.org/. Jeśli używasz systemu Windows, pobierz wersję oprogramowania zawierającą kompilator języka C++ (GCC/G++).

Ćwiczenie 5

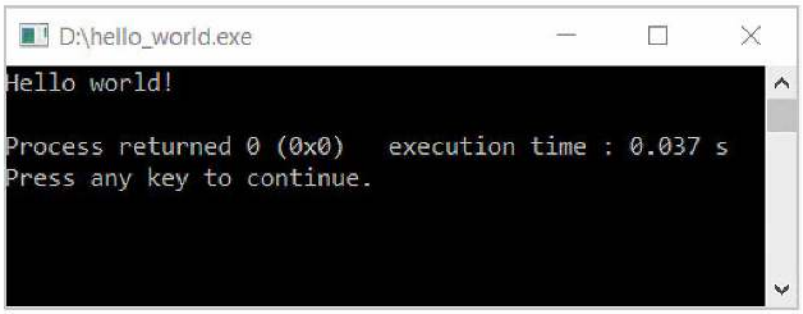
W edytorze Code::Blocks utwórz plik *hello_world.cpp* i zapisz w nim kod źródłowy programu *Hello world*. Zachowaj odstępy w pionie i w poziomie. Po wpisaniu całości kodu zapisz zmiany, wybierając **File → Save File**. Jeśli twój kod źródłowy wygląda inaczej niż w podręczniku, poszukaj błędów i je popraw.

Plik wykonywalny

W środowisku Code::Blocks aby skompilować kod źródłowy z pliku *hello_world.cpp* i uzyskać **plik wykonywalny**, należy wybrać z menu **Build → Build and Run** lub nacisnąć przycisk **F9**. W efekcie otrzymujemy nowy plik na dysku: w systemie Windows będzie to plik o rozszerzeniu *exe*, który zostanie zapisany na dysku w tym samym katalogu co plik źródłowy. Jednocześnie Code::Blocks uruchomi program (rys. 5.4).

Warto wiedzieć

Edytory programistyczne są częścią zintegrowanych środowisk programistycznych zwanych IDE (ang. *integrated development environment*). Umożliwiają one m.in. kompilowanie lub interpretowanie kodu źródłowego oraz ułatwiają usuwanie błędów. Popularnymi środowiskami programistycznymi są: Code::Blocks, Dev-C++, Visual C++ czy Visual Studio.



Rys. 5.4. Wywołanie programu *hello_world.exe*

Program *hello_world.exe* można uruchomić bez pośrednictwa Code::Blocks, z wykorzystaniem Wiersza polecenia (*cmd.exe*). Jeśli np. plik został zapisany w katalogu *Kody* lokalnego konta Windows, to w Wierszu polecenia należy wpisać `cd Kody`, a następnie `hello_world`.

Ćwiczenie 6

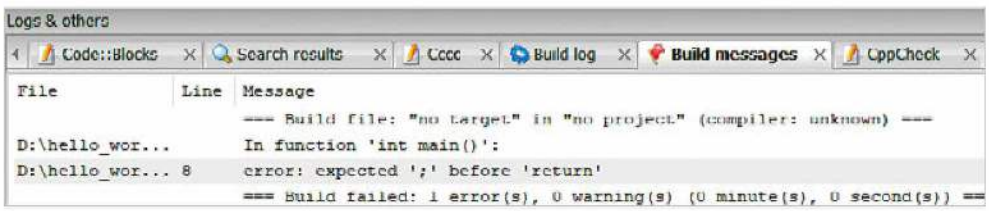
a. Skompiluj kod źródłowy programu *Hello world* i uruchom go. Porównaj wynik z rysunkiem 5.4.
b. Usuń dowolny średnik i skompiluj kod ponownie. Co zauważasz?

Błąd składniowy

Jeśli w kodzie wystąpi **błąd składniowy**, czyli taki, który wynika z zapisania kodu niezgodnie z regułami (składnią) danego języka programowania, poniżej okna z kodem źródłowym wyświetli się komunikat o błędzie. Na przykład jeśli wiersz 7 nie będzie się kończył średnikiem (;), to komunikat będzie miał taką postać jak na rysunku 5.5.

Warto wiedzieć

W systemach Linux i macOS pliki wykonywalne nie są wyróżniane żadnym rozszerzeniem. Aby można było je uruchomić, muszą mieć ustawiony atrybut wykonywalności.



Rys. 5.5. Komunikat o błędzie składniowym – brakuje średnika przed instrukcją `return`

Przykład ten pokazuje, jak bardzo skrupulatnym narzędziem jest **kompilator**. Precyzja zapisu instrukcji ma w programowaniu kluczowe znaczenie. Każda instrukcja musi się kończyć średnikiem.

Kompilator, s. 99

Program *Przeliczanie temperatury* – pojęcie zmiennej i proste operacje arytmetyczne w C++

Program *Hello world* wyświetlał jedynie tekst na ekranie. Napiszemy teraz program bardziej praktyczny, który zwróci wynik na podstawie informacji podanych przez użytkownika.

Mamy do rozwiązania następujący problem. Wyobraź sobie, że zespół naukowców z Polski i USA pracuje nad robotem, który weźmie udział w misji kosmicznej. Naukowcy używają różnych skali do zapisu temperatury – stopni Fahrenheita i stopni Celsjusza. Celem jest stworzenie programu, który pozwoli przeliczać wartości temperatury ze skali Fahrenheita na skalę Celsjusza.

Dla uproszczenia przyjmijmy, że odczyt temperatury, a także wyliczona temperatura będą podane w postaci liczb całkowitych (w razie potrzeby jako liczby całkowite utworzone przez ucięcie części ułamkowej).

Oto specyfikacja tego problemu:

Dobra rada

W programach w języku C++, które wypisują napisy za pomocą instrukcji `cout`, nie stosuj polskich znaków diakrytycznych.

Specyfikacja

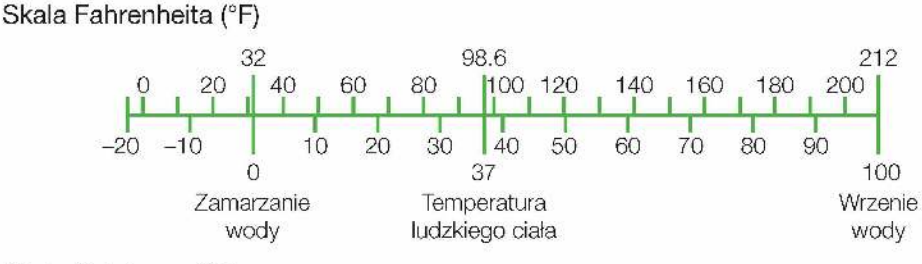
Dane: wartość temperatury w stopniach Fahrenheita podana jako liczba całkowita *f*.

Wynik: wartość temperatury w stopniach Celsjusza (ucięta do części całkowitej).

Warto wiedzieć

Skala Fahrenheita jest obecnie używana m.in. w USA i Belize. W Kanadzie powszechnie wykorzystuje się obie skale temperatur.

Na rysunku 5.6 porównano skalę Fahrenheita ze skalą Celsjusza. Skala Fahrenheita jest przesunięta o 32 stopnie (temperatura zamarzania wody to 0°C i 32°F), ale to nie wszystko. Różnica pomiędzy temperaturami zamarzania i wrzenia wody (w warunkach normalnych) w skali Fahrenheita wynosi 180 stopni (od 32 do 212), a w skali Celsjusza 100 stopni (od 0 do 100). Zatem aby przeliczyć skale, od temperatury podanej w stopniach Fahrenheita odejmujemy 32 i wynik mnożymy przez 100/180, czyli 5/9 po skróceniu ułamka.



Rys. 5.6. Porównanie skali Fahrenheita ze skalą Celsjusza

Algorytm powinien zatem przeliczać temperaturę podaną w skali Fahrenheita (f) na temperaturę w skali Celsjusza (c) według wzoru matematycznego: $c = (f - 32) \cdot \frac{5}{9}$.

Oto kod źródłowy programu *Przeliczanie temperatury*:

Kod źródłowy programu *Przeliczanie temperatury*

```
1.  #include <iostream>
2.
3.  using namespace std;
4.
5.  int main()
6.  {
7.      int f;
8.      cout << "Temperatura w skali Fahrenheita: ";
9.      cin >> f;
10.     cout << f << " stopni F to " << (f-32)*5/9;
11.     cout << " stopni C" << endl;
12.     return 0;
13. }
```

Warto wiedzieć

W języku Scratch za operację wypisania na ekranie komunikatu odpowiada blok **powiedz**:



Warto wiedzieć

W języku Scratch operację wczytania liczby podanej z klawiatury do zmiennej realizuje się z użyciem bloków **zapytaj i ustaw**:



Zwróć uwagę, że powyższy kod ma budowę zbliżoną do poprzedniego – różnice widać dopiero w liniach 7–11, znajdujących się wewnątrz funkcji `main`.

Ponieważ program ma przeliczać wartość podaną przez użytkownika, potrzebujemy elementu, który przechowa tę informację. Dlatego w wierszu 7 wprowadzono **deklarację zmiennej**, czyli określono jej nazwę (`f`) oraz typ (`int`). **Zmienna** to obszar w pamięci komputerowej, któremu nadano nazwę (w tym przypadku `f`) i w którym można zapisać informację – tu wartość temperatury. **Typ danych** jest to zbiór wartości możliwych do zapisania w pamięci urządzenia komputerowego.

Deklaracja zmiennej

Zmienna

Typ danych

O zmiennych w językach programowania możesz myśleć jak o pudełkach na dane – mogą one mieć różny rozmiar i mogą przechowywać różne rodzaje informacji. Typ określa rodzaj danych, które będą przechowywane w zmiennej, i jednocześnie wielkość przydzielonej (zarezerwowanej) pamięci. Typ danych `int` jest przeznaczony dla zmiennych przechowujących liczby całkowite (ang. *integer*). Wszystkie zmienne występujące w tekście programu w języku C++ muszą być zadeklarowane przed pierwszym użyciem.

Warto wiedzieć

Polecenie `cin` pochodzi od angielskiego wyrażenia *console input*, które oznacza pobranie tekstu podanego z klawiatury (czyli standardowego wejścia). Wymawia się je tak jak *see in*.

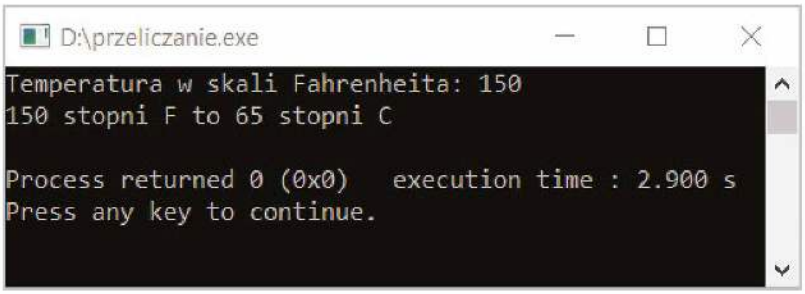
W wierszu 9 instrukcja `cin` pozwala na wczytanie do zmiennej liczby podanej z klawiatury. Jest to tak zwana **instrukcja wejścia** (ang. *input*). Operator `>>` wskazuje, dokąd zostaną przekazane dane wpisane na klawiaturze. W tym przypadku będzie to zmienna `f`.

Instrukcja wejścia

Wiersze 10–11 to kod instrukcji, która wyświetla na ekranie wartość wyrażenia arytmetycznego `(f-32)*5/9`. Dla wartości temperatury podanej w skali Fahrenheita zwraca ono wartość w skali Celsjusza.

Zauważ, że podstawowe operatory arytmetyczne (dodawania, odejmowania, mnożenia i dzielenia) są zapisywane w języku C++ z użyciem znaków znanych np. z arkusza kalkulacyjnego (+, –, *, /).

Rysunek 5.7 przedstawia okno programu *Przeliczanie temperatury*.



Rys. 5.7. Wywołanie programu *przeliczanie.exe*

Warto wiedzieć

Jeżeli w działaniu `(f-32)*5/9` zmienna `f` jest typu `int`, to w języku C++ wynik działania zostanie przycięty do części całkowitej.

Ćwiczenie 7

Utwórz plik *przeliczanie.cpp* i zapisz w nim kod źródłowy programu *Przeliczanie temperatury*, a następnie wykonaj poniższe czynności.

a. Skompiluj kod źródłowy programu i przetestuj go dla różnych danych.
b. Stwórz program, który będzie przeliczał temperaturę ze skali Celsjusza na skalę Fahrenheita. Zapisz kod źródłowy w pliku *przeliczanie2.cpp*, skompiluj go i sprawdź działanie programu.

Program *Logowanie* – instrukcja warunkowa i operatory logiczne

Poprzedni program zawsze wykonywał wiersz po wierszu wszystkie instrukcje zapisane w kodzie źródłowym – przeliczał temperaturę i zwracał jej wartość na ekranie. Napiszemy teraz program, który będzie działał wariantowo – w zależności od wprowadzonych danych wybierze jedną z dwóch ścieżek: wykona następną instrukcję kodu albo ją pominie.

Załóżmy, że mamy do rozwiązania następujący problem: należy utworzyć symulator logowania do konta w serwisie internetowym. W zależności od tego, czy wprowadzone dane będą poprawne, symulator będzie zwracał odpowiedni komunikat. Potrzebna więc będzie konstrukcja programistyczna, która pozwoli uzależnić działanie programu od spełnienia określonych warunków. Taką konstrukcją jest **instrukcja warunkowa** (warunkowe rozgałęzienie):

Instrukcja warunkowa

```
if (warunek logiczny) instrukcja na tak;
else instrukcja na nie;
```

Jeśli warunek logiczny (nazywany również wyrażeniem testowym) ma **wartość logiczną** prawda, to wykonywana jest instrukcja na tak, w przeciwnym razie – instrukcja na nie. Część `else`, oznaczająca „w przeciwnym przypadku", jest opcjonalna, czyli może nie występować w kodzie.

Logika matematyczna i operatory logiczne, s. 252–254

Poniżej zapisano kod źródłowy programu *Logowanie*.

Kod źródłowy programu *Logowanie*

```
1.  #include <iostream>
2.  #include <string>
3.  using namespace std;
4.
5.  int main()
6.  {
7.      string login, login_baza="Robik";
8.      string haslo, haslo_baza="Gwi@zdA#99";
9.
10.     cout << "Login: "; cin >> login;
11.     cout << "Haslo: "; cin >> haslo; cout << endl;
12.     if (login==login_baza && haslo==haslo_baza)
13.         cout << "Logowanie prawidlowe" << endl;
14.     else
15.         cout << "Blad logowania" << endl;
16.
17.     return 0;
18. }
```

Warto wiedzieć

W algorytmie z warunkami może wystąpić kilka alternatywnych ciągów działań. Wybór jednego z nich następuje w zależności od spełnienia określonych warunków.

Dobra rada

Biblioteka `string` zawiera wiele funkcji pozwalających wykonywać różne operacje na łańcuchach znaków. Korzystaj z tej biblioteki, nawet jeśli środowisko programistyczne jej nie wymaga, ponieważ w przeciwnym razie kompilacja tego samego kodu w innym środowisku może się nie udać.

W wierszach 7 i 8 zadeklarowano zmienne typu tekstowego **`string`**. W linii 2 dołączono bibliotekę `string`, która obsługuje łańcuchy znaków (teksty). Zauważ, że w jednej linii kodu można jednocześnie zadeklarować kilka zmiennych tego samego typu (oddziela się je przecinkiem). Zmiennym `login_baza` oraz `haslo_baza` nadano wartości od razu (używa się do tego operatora przypisania `=`). Wartości przechowywane przez te zmienne będą stanowiły uproszczony wzorzec danych przechowywanych na serwerze serwisu internetowego.

W wierszu 10 pojawiają się instrukcje **`cout`** i **`cin`**. Pierwsza wyświetli na ekranie napis „Login: ", a druga pobierze podany przez użytkownika napis. Analogicznie działają instrukcje w wierszu 11, ale dotyczą one hasła (nie będziemy zamieniać znaków hasła na gwiazdki).

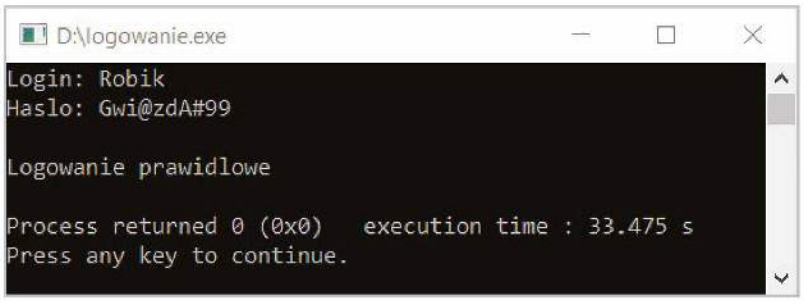
Wiersze 12–15 to przykład użycia instrukcji warunkowej **`if`**, która sprawdza poprawność podanych loginu i hasła. W zależności od wartości logicznej warunku wyświetlany jest napis „Logowanie prawidłowe" lub „Błąd logowania".

Koniunkcja

Zauważ, że wyrażenie testowe jest wyrażeniem złożonym, zwanym przez matematyków **koniunkcją** (dwa wyrażenia połączone spójnikiem „i"), zapisywanym w C++ za pomocą operatora `&&`. Wykonywane są dwa porównania typu „czy zachodzi równość?" (z użyciem operatora `==`): porównanie pierwszego tekstu wprowadzonego z klawiatury z loginem zapisanym w bazie oraz porównanie drugiego tekstu wprowadzonego z klawiatury z hasłem zapisanym w bazie. Zgodnie z logiką schematu logowania (rys. 5.1, s. 95) oczekuje się, że oba porównania zakończą się sukcesem (przyjmą logiczną wartość prawda). Okno programu *Logowanie* przedstawiono na rysunku 5.8.

Warto wiedzieć

W nazwach zmiennych oprócz liter (małych i wielkich) oraz cyfr (te nie mogą być na początku nazwy) można używać również znaku podkreślenia _.



Rys. 5.8. Wywołanie programu *logowanie.exe*

Ćwiczenie 8

Utwórz plik *logowanie.cpp* i zapisz w nim kod źródłowy programu *Logowanie*. Następnie skompiluj kod źródłowy programu i uruchom go kilkukrotnie, sprawdzając różne przypadki (dla pary: login i hasło).

W pracy ze zmiennymi tekstowymi często używa się również innych **operatorów**:

- operator porównań `!=` stosuje się wtedy, kiedy chcemy sprawdzić, czy dwie wartości są różne, np. wartość logiczna porównania `"kot" != "pies"` będzie prawdą,
- operator logiczny `||` stosuje się do zapisania alternatywy dwóch wyrażeń, np. `a<5 || a>10`.

Ćwiczenie 9

Zmodyfikuj program *Logowanie* tak, aby wykorzystać w nim operatory `!=` oraz `||`. Kod źródłowy zapisz w pliku *logowanie2.cpp* i sprawdź działanie programu.

Warto wiedzieć

W edytorze języka Scratch instrukcja warunkowa znajduje się wśród bloków kontrolnych.



Dobra rada

W języku C++ możesz wstawiać odstępy między operatorami, ale ich brak nie wpływa na poprawność kodu.

Logika matematyczna i operatory logiczne, s. 252–254

Program *Warunek trójkąta* – zamiana miejscami dwóch wartości

Załóżmy, że mamy do rozwiązania następujący problem: chcemy rozstrzygnąć, czy z trzech odcinków o podanych długościach można zbudować trójkąt.

Specyfikacja

Dane: trzy różne liczby całkowite dodatnie a, b, c.

Wynik: komunikat „TAK”, jeśli z odcinków o długościach a, b, c można zbudować trójkąt, lub „NIE” – w przeciwnym przypadku.

Warto wiedzieć

Sprawdzanie warunku istnienia trójkąta wydaje się banalne w przypadku małych liczb, ale jest o wiele trudniejsze np. dla liczb 299 654, 589 522, 289 877. Opracowanie algorytmu rozwiązującego ten problem pozwoli napisać program, który w krótkim czasie wykona milion takich rozstrzygnięć.

Z lekcji matematyki możesz pamiętać, że tzw. **warunek istnienia trójkąta** mówi o tym, że z odcinków o długościach a, b, c można zbudować trójkąt tylko wtedy, gdy najdłuższy odcinek jest krótszy od sumy dwóch pozostałych.

Warunek istnienia trójkąta

Aby komputer mógł sprawdzać warunek trójkąta, trzeba opracować dla niego algorytm postępowania: liczby *a*, *b*, *c* będą reprezentowane przez zmienne a, b, c. Program komputerowy wybierze największą wartość, zamieni miejscami tę wartość z wartością zmiennej c (o ile to konieczne) i znajdzie rozwiązanie problemu przez sprawdzenie wartości logicznej wyrażenia a + b > c.

Algorytm wybierania największej z trzech liczb

Algorytm wybierania największej z trzech liczb *a*, *b* i *c* może mieć następującą postać:

1. Sprawdź, czy a > b i a > c. Jeśli tak, to a jest największą liczbą, więc zamień miejscami wartości zmiennych a i c i przejdź do kroku 3.
2. Sprawdź, czy b > c. Jeśli tak, to b jest największą liczbą, więc zamień miejscami wartości zmiennych b i c.
3. Sprawdź, czy a + b > c. Jeśli tak, to wypisz komunikat „TAK". W przeciwnym wypadku wypisz „NIE".

Oto kod źródłowy powyższego algorytmu w języku C++:

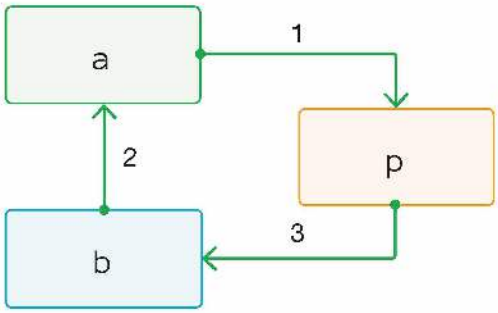
Kod źródłowy programu *Warunek trójkąta*

```
1.  #include <iostream>
2.
3.  using namespace std;
4.
5.  int main()
6.  {
7.      int a,b,c,p; // p to zmienna pomocnicza
8.      cin >> a >> b >> c;
9.
10.     if (a>b && a>c)
11.     { // jeśli a jest największe
12.         p = a;
13.         a = c;
14.         c = p;
15.     }
16.     else
17.         if (b>c)
18.         { // jeśli b jest największe
19.             p = b;
20.             b = c;
21.             c = p;
22.         }
23.
24.     if (a + b > c)
25.         cout << "TAK" << endl;
26.     else
27.         cout << "NIE" << endl;
28.
29.     return 0;
30. }
```

Warto wiedzieć

Pozornie łatwe do rozwiązania problemy mogą się skomplikować, kiedy dodamy do nich nowe założenia lub zwiększymy ilość danych wejściowych. Na przykład wyświetlenie mniejszej z dwóch liczb wydaje się proste, ale wybranie najmniejszej spośród 15 liczb wymaga nieco innego podejścia.

Kluczowa operacja zamiany liczb miejscami, a precyzyjniej: **zamiana wartości zmiennych** a i c albo b i c, jest zapisana w wierszach 12–14 i 19–21. Użyliśmy do tego zmiennej pomocniczej p, która przechowa tymczasowo wartość jednej ze zmiennych.

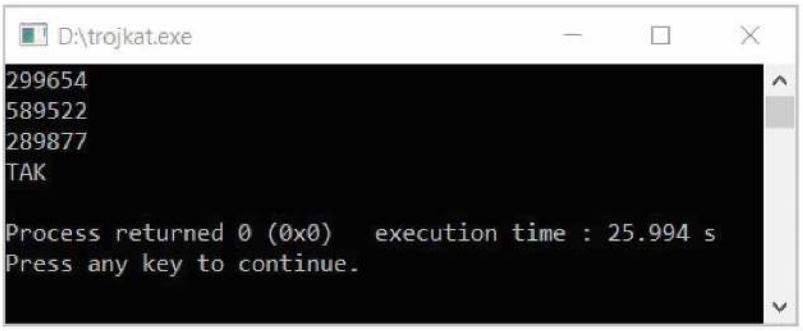


Rys. 5.9. Ilustracja zamiany wartości dwóch zmiennych

Zamiana wartości zmiennych

Pomysł ten zilustrowano na rysunku 5.9. Liczby wskazują kolejność wykonywanych podstawień (przypisywania wartości zmiennym). Istotne fragmenty programu opisano komentarzami. Dzięki nim kod staje się łatwiejszy do zrozumienia dla osoby, która ma z niego korzystać.

Rysunek 5.10 przedstawia okno programu *Warunek trójkąta*.



Rys. 5.10. Wywołanie programu *trojkat.exe*

W programie pojawiają się elementy składni, których do tej pory nie stosowaliśmy:

- w wierszach 11–15 oraz 18–22 zgrupowano instrukcje z użyciem pary nawiasów { i } – są one konieczne przy grupach instrukcji, ale można je pominąć przy pojedynczej instrukcji,
- w wierszach 17–22 pojawia się instrukcja **if** po **else**, co oznacza, że instrukcje z wierszy 19–21 będą wykonane tylko wtedy, gdy będzie spełniona nierówność z wiersza 17 (b>c),
- w wierszach 7, 11 i 18 wstawiono komentarze, które rozpoczynają się od znaku //, a kończą na końcu wiersza – w procesie kompilacji komentarze są ignorowane.

Warto wiedzieć

Zagadnienie zamiany wartości dwóch zmiennych można zobrazować następująco: mamy dwie szklanki – jedną z sokiem, drugą z wodą. Jak przelać sok do szklanki, w której jest woda, a wodę do szklanki z sokiem? Wykorzystanie trzeciej szklanki pozwoli rozwiązać ten problem.

Dobra rada

Stosuj w kodzie komentarze. Ułatwiają one czytanie kodu i pozwalają szybko znaleźć jego właściwy fragment po dłuższym czasie, np. gdy chcemy go zmodyfikować.

Ćwiczenie 10

a. Utwórz plik *trojkat.cpp* i zapisz w nim kod źródłowy programu *Warunek trójkąta*. Skompiluj go i uruchom program kilkukrotnie, aby sprawdzić jego działanie dla różnych wartości a, b i c.

b. Ile jest sposobów uszeregowania trzech liczb a, b, c o różnych wartościach (tzn. takich, że $a \neq b \neq c$)?

Podsumowanie

- Algorytm to opis postępowania prowadzącego do rozwiązania problemu.
- Algorytmiczne rozwiązywanie problemu jest procesem, na który składają się: zrozumienie problemu, zaprojektowanie algorytmu jego rozwiązania, komputerowa realizacja algorytmu, testowanie programu.
- Zapis algorytmu z użyciem instrukcji języka programowania nazywamy kodem źródłowym lub tekstem programu.
- Typ danych `int` reprezentuje liczby całkowite, a typ `string` – łańcuchy znaków, czyli napisy.
- Instrukcja warunkowa umożliwia wykonanie pewnych operacji w zależności od prawdziwości wyrażenia logicznego. Zapisuje się ją przy użyciu konstrukcji `if` oraz `if else`.
- Do podstawowych konstrukcji języka programowania należą: deklaracja zmiennych i instrukcja przypisania wartości do zmiennej, instrukcje wejścia i wyjścia, wyrażenia arytmetyczne i logiczne oraz instrukcja warunkowa.

Zadania

* **1** Wiele instrukcji służących ratowaniu życia człowieka przyjmuje postać algorytmu (np. sposób użycia defibrylatora lub instrukcja ewakuacji z budynku). Podaj kilka przykładów czynności z życia codziennego, które można opisać za pomocą algorytmów. Wybierz jedną z tych czynności i zapisz w postaci listy kroków.

* **2** Podaj specyfikację dla problemu zaplatania warkocza ze s. 96.

* **3** Napisz program, który wczyta dwie różne liczby całkowite dodatnie (np. wiek twój i najstarszej osoby w twojej rodzinie) i sprawdzi, ile razy mniejsza z nich mieści się w większej liczbie.

* **4** Napisz program, który obliczy pole powierzchni trapezu dla podanych długości podstaw oraz wysokości trapezu. Wynik powinien być obliczony z dokładnością do części całkowitej.

* **5** Napisz program, który będzie wyznaczał średnią prędkość pojazdu pokonującego podaną odległość w podanym czasie. Aby sprawdzić działanie programu, oblicz średnią prędkość pociągu dla wybranych przez siebie połączeń w rozkładzie jazdy.

* **6** Napisz program, który wczyta dwie liczby całkowite, a następnie wypisze większą z nich wraz z komunikatem „jest większa". Jeśli liczby są równe, to program wypisze komunikat „Liczby są równe".

* **7** Napisz program, który przeliczy wielkość podaną w bajtach na bity.

* **8** Napisz program, który wartość prędkości podaną w m/s zapisze w km/h. Wartość powinna być wyznaczona z dokładnością do części całkowitych.

** **9** Napisz program, który będzie przeliczał złotówki na dolary (według ustalonego kursu). Przed utworzeniem kodu zapisz specyfikację problemu.

** **10** Napisz program, który wczyta trzy różne liczby całkowite i sprawdzi, czy były podane w porządku niemalejącym. Program powinien zwrócić odpowiedź „TAK" lub „NIE".

** **11** Napisz program, który wczyta trzy wyrazy i zwróci informację o liczbie wystąpień każdego z nich wśród wyrazów wczytanych. Na przykład po wczytaniu wyrazów „pies kot pies" na ekranie pojawi się informacja: „pies 2 kot 1".

** **12** Na lekcjach geometrii wykorzystuje się cyrkiel i linijkę do tworzenia konstrukcji geometrycznych. Zapisz listę kroków pozwalającą:
a. zbudować trójkąt równoboczny o zadanej długości a,
b. wyznaczyć środek odcinka łączącego punkty A i B.
Operacje typu „ustaw nóżkę cyrkla...", „zakreśl okrąg...", „poprowadź prostą...", „zaznacz punkt przecięcia..." uznaj za elementarne, niewymagające rozkładania na czynności prostsze.

** **13** Napisz program, który dla podanego wzrostu (w centymetrach) i wagi (w kilogramach) obliczy wskaźnik masy ciała (BMI).
Uwaga: Przyjmij, że użytkownik wprowadza liczby całkowite, bez części ułamkowej.

*** **14** Napisz program, który zapyta użytkownika o bieżący rok, miesiąc i dzień miesiąca oraz o datę urodzenia, a w odpowiedzi zwróci informację, czy użytkownik jest pełnoletni. Jeśli nie jest, program poda, za ile miesięcy osiągnie on pełnoletniość.

*** **15** Napisz program, który po wczytaniu trzech liczb całkowitych wpisanych z klawiatury wyświetli je w porządku niemalejącym. Program nie powinien wykonywać operacji zamiany liczb miejscami.

*** **16** Napisz program, który po wczytaniu liczby trzycyfrowej wyświetli jej cyfry w odwróconym porządku.
Uwaga: Wczytywana liczba nie powinna być podzielna przez 10.

*** **17** Napisz program, który dla liczb a, b, c i d obliczy i wyświetli sumę wartości bezwzględnych różnic: $|a-b|+|b-c|+|c-d|$. Program powinien się posługiwać wyłącznie instrukcjami `if` oraz dodawaniem i odejmowaniem.

6. Wyszukujemy i sumujemy

Do rozwiązania wielu problemów przydają się algorytmy, w których wielokrotnie trzeba powtarzać szereg obliczeń. Takie obliczenia, zwane cyklicznymi, potrzebne są np. w grafice komputerowej do uzyskania efektu cienia w scenie 3D. Komputer o mocy obliczeniowej liczonej w miliardach operacji na sekundę jest w stanie realizować tego typu zadania znacznie szybciej niż człowiek. Oprócz szybkości komputery mają jeszcze jedną zaletę: nie mylą się podczas wykonywania żmudnych, powtarzalnych czynności. Dlatego w tym temacie nauczysz się zapisywać w języku C++ proste algorytmy obliczeń cyklicznych z wykorzystaniem pętli.

Cele lekcji

- Zrozumiesz, na czym polega iteracyjne rozwiązywanie problemów.
- Nauczysz się tworzyć algorytmy zawierające instrukcje powtarzania.
- Napiszesz w języku C++ programy z użyciem pętli.

Warto wiedzieć

Słowo „iteracyjny" pochodzi od łacińskiego czasownika *iterare*, który znaczy „powtarzać".

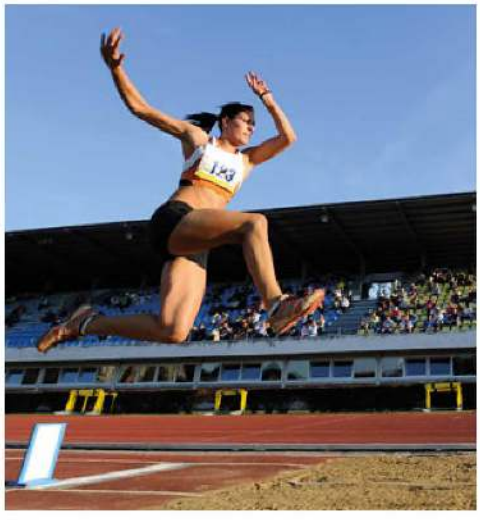
Do tej pory każdy nasz program zakładał jednokrotne wykonanie instrukcji zapisanych w tekście programu. Napiszemy teraz programy, które będą wielokrotnie wykonywały ten sam fragment kodu. Będą one komputerowymi realizacjami algorytmów nazywanych iteracyjnymi, czyli takich, w których aby uzyskać określony efekt, powtarza się jakąś czynność wielokrotnie.

6.1. Szukamy największej liczby. Pętla `while`

Warto wiedzieć

Określenie jednostek, w których wyrażone są różne wielkości, jest dla programisty bardzo ważne, ponieważ musi on odpowiednio wybrać typ danych. Na przykład jeśli dokładność pomiaru skoku w dal wynosi 1 cm, to wyniki podawane w centymetrach będą liczbami całkowitymi. Jeśli jednak wyniki podajemy w metrach, mogą być ułamkami.

Rozważmy następujący problem: trener skoku w dal mierzy podczas treningu długości kolejnych skoków i wpisuje wyniki na bieżąco do programu komputerowego. Chcemy stworzyć program, który na zakończenie treningu wyświetli długość najdłuższego skoku. Zakładamy, że wyniki podawane są w metrach z dokładnością do 0,01 m, czyli 1 cm.



Rys. 6.1. Skok w dal

Nie wiemy, ile skoków lekkoatleta wykona podczas treningu. Nawet jeśli trener zaplanuje jakąś ich liczbę, to i tak niektóre skoki będą prawdopodobnie spalone (nieważne).

Program komputerowy trzeba więc napisać tak, aby trener mógł poinformować komputer o tym, że trening został zakończony. Jak to zrobić?

Jednym ze sposobów może być wpisanie przez trenera liczby 0, która zakończy ciąg podawanych liczb. Wtedy program komputerowy powinien wypisać długość najdłuższego skoku.

Dobra rada

Ciąg potraktuj jako układ liczb, w którym następują one w określonej kolejności.

Specyfikacja

Dane: ciąg liczb zapisanych z dokładnością do dwóch miejsc po przecinku zakończony liczbą 0.

Wynik: największa z podanych liczb.

Aby wykonać to zadanie, program musi porównać długości skoków. Komputer nie potrafi jednak jednocześnie porównywać wielu liczb, nawet gdyby miał je wszystkie zapisane w pamięci. Pracuje on sekwencyjnie i nie może np. objąć wzrokiem kilku liczb, aby wskazać największą. W jednej operacji może brać pod uwagę tylko dwie liczby. Pozostałe są niedostępne.

Na poniższym schemacie zaprezentowano, jaką strategię wyboru największej spośród czterech podanych liczb stosuje człowiek, a jaką – komputer.

Warto wiedzieć

W żargonie programistów specjalną wartość (np. 0), która wskazuje koniec ciągu wczytywanych do programu wartości, nazywa się wartownikiem (ang. *sentinel*). Wartość ta musi być unikatowa, aby nie można było jej pomylić z wartościami podanymi przez użytkownika programu.

Jak widzi komputer?

Zadanie:
Która z podanych liczb jest największa?

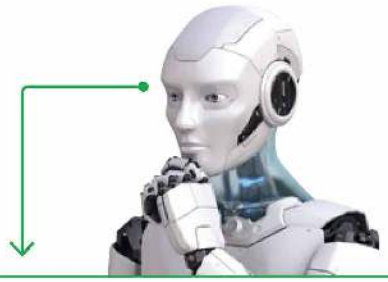
6,40	6,53	6,34	6,55



Człowiek

Przy małej liczbie danych człowiekowi wystarczy jedno spojrzenie, aby wskazać największą liczbę:

6,40	6,53	6,34	6,55



Komputer

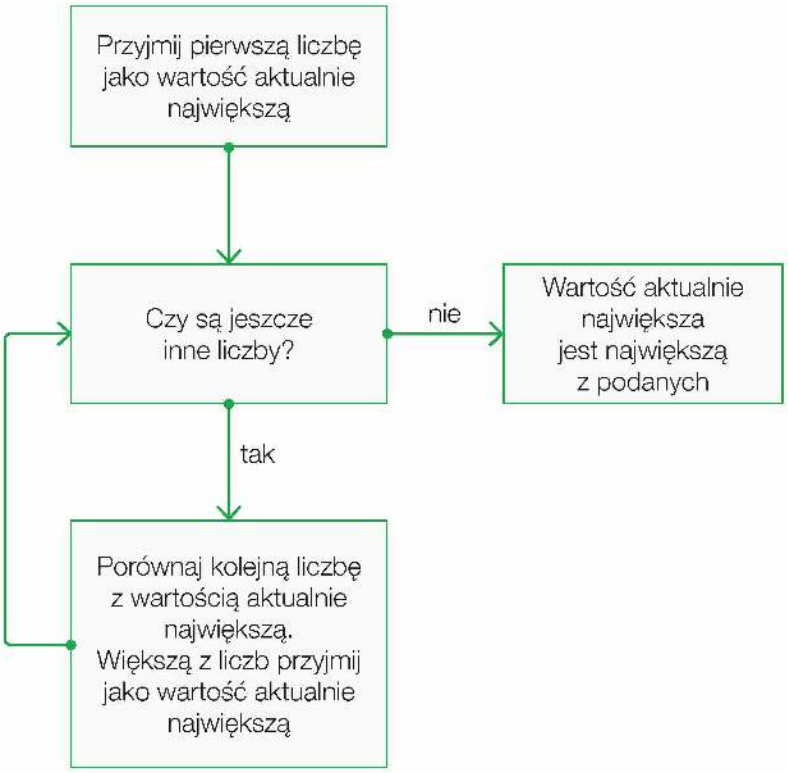
Komputer, nawet przy małej liczbie danych, etapami porównuje kolejne pary liczb i wskazuje większą liczbę z pary:

Etap 1	6,40	6,53		
Etap 2		6,53	6,34	
Etap 3		6,53		6,55

Aby program mógł wyznaczyć najdłuższy z podanych skoków, będziemy porównywać ich długości parami: porównamy pierwszy skok z drugim, dłuższy z nich porównamy z trzecim skokiem, dłuższy z nich z czwartym itd. Operacje te program będzie wykonywał do momentu, aż użytkownik wpisze liczbę 0.

Dzięki takiemu rozwiązaniu program będzie musiał zapamiętywać tylko dwie wartości: aktualnie podaną przez użytkownika i największą z dotychczas podanych. W naszej sytuacji pamiętanie pozostałych wartości nie jest potrzebne.

Poniższy diagram (rys. 6.2) pokazuje, na czym polega cykliczne wykonywanie tej samej operacji na kolejnych danych. Zauważ, że graficznie ma ono postać zbliżoną do pętli.



Rys. 6.2. Diagram przedstawiający algorytm wyszukiwania największej wartości

Lista kroków algorytmu wyznaczania największej liczby z ciągu liczb mogłaby wyglądać następująco:

1. Jako wartość aktualnie największą (maksimum) przyjmij 0 i wczytaj pierwszą liczbę.
2. Dopóki ostatnia wczytana liczba jest różna od 0, wykonuj krok 3 i krok 4.
   3. Jeśli ostatnia wczytana liczba jest większa od maksimum, to podstaw ją za maksimum.
   4. Wczytaj kolejną liczbę z klawiatury.
5. Wypisz maksimum na ekranie.

Warto wiedzieć

Procesor porównujący ze sobą dwie liczby używa sprzętowej instrukcji porównywania. Jej działanie polega na sprawdzeniu, czy różnica porównywanych liczb jest dodatnia, ujemna czy równa 0.

Przeanalizujemy teraz działanie algorytmu na przykładowych danych. Wygodnie będzie zrobić to w tabeli.

Wartość aktualnie największa (maksimum)	Wczytana liczba	Czy ostatnia wczytana liczba jest różna od 0?	Czy ostatnia wczytana liczba jest większa od aktualnego maksimum?
0	6,20	tak	tak
6,20	6,23	tak	tak
6,23	6,34	tak	tak
6,34	6,30	tak	nie
6,34	6,41	tak	tak
6,41	6,14	tak	nie
6,41	6,34	tak	nie
6,41	6,27	tak	nie
6,41	0	nie	–

Tabela 6.1. Działanie algorytmu wyznaczania największej liczby

Dobra rada

Jeżeli trudno ci zrozumieć, jak działa algorytm, możesz utworzyć tabelę zawierającą pośrednie wartości zmiennych i warunków logicznych.

Tabelę uzupełniamy wierszami od lewej do prawej, według kolejnych kroków algorytmu. Kolorem zielonym zaznaczono pewien etap analizy działania algorytmu: wczytano w tym momencie liczbę 6,41. Kolejnymi operacjami będą: sprawdzenie, czy liczba 6,41 jest różna od 0 (krok 2), a następnie – czy jest większa od 6,34 (krok 3). Ponieważ oba warunki są spełnione, liczba 6,41 staje się aktualnie największą. W kolejnych iteracjach podawane są liczby od niej mniejsze, dlatego w momencie wpisania liczby 0 liczba aktualnie największa staje się po prostu największą spośród wszystkich podanych liczb.

A to ciekawe

Muzyczne pętle

Pętlą lub loopem w muzyce elektronicznej nazywa się odpowiednio przygotowany fragment utworu (sampel), który wielokrotnie powtarza się w innym utworze. Jedną z najczęściej wykorzystywanych pętli perkusyjnych w muzyce rozrywkowej jest fragment z utworu Jamesa Browna *Funky Drummer*. Według serwisu whosampled.com wykorzystano go w ponad 1500 utworach innych artystów.



Ćwiczenie 1

Zastanówcie się w grupie, czy jeśli na liście kroków ze s. 114 zamienimy miejscami kroki 3 i 4, lista nadal będzie przedstawiać poprawny algorytm wyszukiwania wartości największej. W jaki sposób można to sprawdzić?

Zauważ, że program wyznaczający największą liczbę z ciągu powinien cyklicznie:

1. sprawdzać, czy liczba wprowadzona przez użytkownika jest liczbą 0, której podanie kończy działanie programu,
2. sprawdzać, czy wprowadzona liczba jest większa od aktualnie największej,
3. wczytywać kolejną wprowadzoną liczbę.

Instrukcja iteracyjna (instrukcja powtarzania, pętla)

W języku C++ operacje cykliczne realizuje się w specjalnych **instrukcjach iteracyjnych** (**powtarzania**), potocznie nazywanych **pętlami** (ang. *loop*). W językach programowania stosuje się różne rodzaje instrukcji iteracyjnych. W poniższym przykładzie wykorzystamy jeden z rodzajów pętli – **pętlę `while`**, która ma następującą budowę:

Pętla `while`

```
while (warunek) instrukcja;
```

Logika matematyczna i operatory logiczne, s. 252

Ta pętla pozwala wykonywać daną instrukcję (prostą lub złożoną), dopóki warunek jest spełniony, tzn. przyjmuje **wartość logiczną** prawda. Wartość logiczną wyrażenia program sprawdza przed każdym wykonaniem instrukcji, może więc: nie wykonać danej instrukcji, wykonać ją raz albo wiele razy.

Kod źródłowy programu *Najdłuższy skok* może wyglądać następująco:

Kod źródłowy programu *Najdłuższy skok*

```
1.  #include <iostream>
2.
3.  using namespace std;
4.
5.  int main()
6.  {
7.      float a,maks;
8.
9.      maks=0;
10.     cout << "Podaj kolejne dlugosci skokow" << endl;
11.     cin >> a;
12.
13.     while (a!=0)
14.     {
15.         if (a>maks) maks = a;
16.         cin >> a;
17.     }
```

Warto wiedzieć

Zmiennym zadeklarowanym w funkcji `main` kompilator nie przypisuje żadnej domyślnej wartości. Zmienna ma więc domyślnie wartość przypadkową.

```
18.
19.     cout << "\nNajdluzszy skok: " << maks << " m" << endl;
20.
21.     return 0;
22. }
```

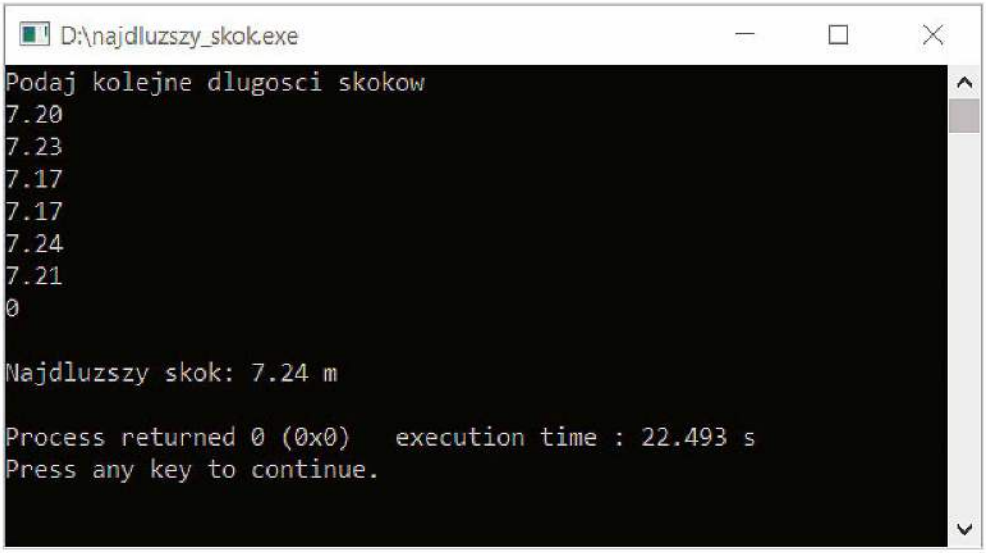
W wierszu 7 deklarujemy dwie zmienne typu **float** (a oraz maks), które przechowują liczby wymierne. W zmiennej a będzie przechowywana liczba jako ostatnia wczytana z klawiatury, a w zmiennej maks – aktualnie największa z wczytanych liczb. Na początku przypisujemy tej zmiennej wartość 0. W wierszu 11 wczytujemy wartość zmiennej a.

W wierszach 13–17 pojawia się pętla **while**, która najpierw sprawdza, czy wartość zmiennej a jest różna od 0. Jeżeli tak, to program wykonuje grupę instrukcji w nawiasach klamrowych. Instrukcje wykonywane są tyle razy, ile razy użytkownik podał liczbę różną od zera.

W wierszu 15 sprawdzamy, czy podana przez użytkownika liczba jest większa od wartości zmiennej maks. Jeśli tak, to do zmiennej maks przypisywana jest aktualna liczba. Po zakończeniu wprowadzania liczb (czyli po wpisaniu przez użytkownika liczby 0) aktualnie największa liczba staje się liczbą największą dla tego zestawu danych. Zauważ, że cała instrukcja **if** jest zapisana w jednym wierszu, ponieważ wykonywana jest w niej tylko jedna operacja.

W linii 19 użyto zapisu \n, który odpowiada za przeniesienie kursora do kolejnego wiersza. Zapis **cout** << "\n" można stosować wymiennie z zapisem **cout** << **endl**.

Okno programu *Najdłuższy skok* przedstawiono na rysunku 6.3. Zwróć uwagę, że w programie liczby ułamkowe podaje się z kropką zamiast przecinka.



Rys. 6.3. Wywołanie programu *najdluzszy_skok.exe*

Dobra rada

Jeżeli używasz w pętli **while** warunku porównującego zmienne typu **float**, staraj się stosować operatory porównania <, > i != zamiast <=, >= oraz ==. Pozwoli to uniknąć niektórych błędów wynikających z dokładności przybliżeń wartości zmiennych.

Dobra rada

We wnętrzu pętli może wystąpić również pojedyncza instrukcja. Wówczas nie musisz stosować nawiasów klamrowych.

Warto wiedzieć

Od lewego ukośnika zapisuje się tzw. stałe znakowe znaków niegraficznych kodu ASCII (np. \n dla nowego wiersza, \t dla tabulacji poziomej) lub znaków specjalnych (np. \" dla znaku ", gdy chcemy go wyświetlić na ekranie).

Warto wiedzieć

Zapisywanie liczb ułamkowych z kropką, a nie z przecinkiem, jest zwyczajem anglosaskim. To dlatego na klawiaturze numerycznej znajduje się właśnie znak kropki.

Ćwiczenie 2

Utwórz plik *najdluzszy_skok.cpp* i zapisz w nim kod źródłowy programu *Najdłuższy skok*. Skompiluj kod i uruchom program kilkukrotnie dla różnych danych wpisanych z klawiatury. Aby przyspieszyć wprowadzanie danych do programu, możesz podać je w jednym wierszu rozdzielone spacjami, np. `6.11 6.24 6 6.01 6.12 0`, a następnie nacisnąć **Enter**.

6.2. Sumujemy liczby. Pętla for

Pętla for

Drugą z instrukcji iteracyjnych w języku C++ jest **pętla for**. W odróżnieniu od instrukcji **while**, stosuje się ją w przypadkach, gdy można z góry określić liczbę cykli, czyli wykonań pętli, nawet jeśli dokładna liczba powtórzeń zależy od danych podanych przez użytkownika. Pętlę **for** wykorzystamy w kolejnym programie.

Problem, który sobie postawimy, to: wyznaczyć sumę n kolejnych liczb parzystych.

Specyfikacja

Dane: liczba naturalna n.

Wynik: suma n kolejnych liczb parzystych.

Aby rozwiązać problem od strony algorytmicznej, posłużymy się prostą matematyką: będziemy wielokrotnie dodawać kolejne składniki (kolejne liczby parzyste) do sumy częściowej, aż liczba dodanych składników będzie równa n. Zaczniemy od sumy równej 0 i składnika 2.

Sposobem na wielokrotne wykonanie instrukcji zwiększania sumy będzie instrukcja pętli. Do zapamiętania, ile razy została wykonana pętla, użyjemy **zmiennej sterującej** pętlą **for**, nazywanej też licznikiem.

Zmienna sterująca

Warto wiedzieć

W języku Scratch do tworzenia pętli służą bloki **powtarzaj aż** oraz **powtórz**, znajdujące się wśród bloków kontrolnych.





Składnia instrukcji **for** ma następującą postać:

```
for (licznik; warunek powtarzania; krok) instrukcja
```

W nawiasie po słowie **for** zapisuje się:

- instrukcję inicjującą, która nadaje wartość początkową zmiennej sterującej pętlą (licznik),
- wyrażenie testowe, czyli warunek logiczny, od którego zależy powtarzanie instrukcji znajdujących się w pętli,
- operację zwiększenia wartości zmiennej sterującej (ustalenie kroku pętli).

Jeśli powtarzana ma być instrukcja złożona, czyli składająca się z wielu instrukcji, to powinna być zapisana między nawiasami { i }.

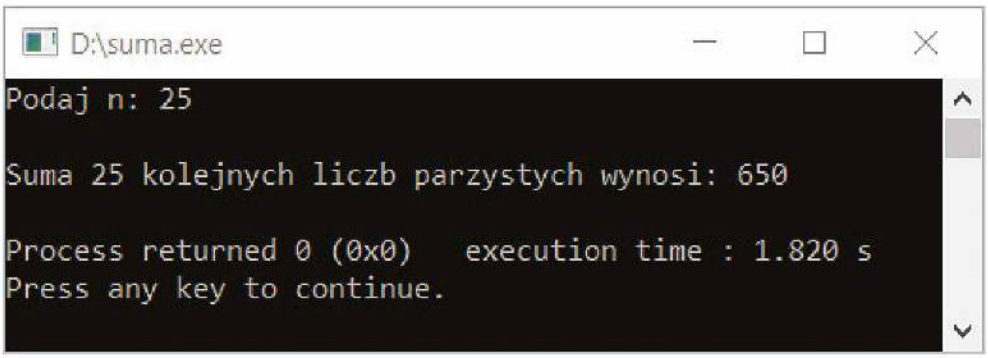
Kod źródłowy programu sumującego *n* kolejnych liczb parzystych może wyglądać następująco:

Kod źródłowy programu *Suma*

```
1.  #include <iostream>
2.
3.  using namespace std;
4.
5.  int main()
6.  {
7.      int n;
8.      cout << "Podaj n: "; cin>>n;
9.
10.     int i=1, suma=0, skladnik=2;
11.     for (i=1; i<=n; i++)
12.     {
13.         suma = suma + skladnik;
14.         skladnik = skladnik + 2;
15.     }
16.
17.     cout << "Suma " << n << " kolejnych liczb ";
18.     cout << "parzystych wynosi: " << suma <<  endl;
19.
20.     return 0;
21. }
```

Zwróć uwagę na wiersze 11–15, czyli na składnię instrukcji **for**. Podczas wykonywania pętli najpierw jest ustalana wartość początkowa zmiennej sterującej: i=1. Wartość zmiennej sterującej zwiększa się o 1 w każdym cyklu pętli do momentu osiągnięcia przez nią wartości zmiennej n. Aby zwiększyć wartość zmiennej typu **int** o 1, stosuje się zwykle specjalną składnię. Zamiast i=i+1 można napisać i++ (analogicznie zmniejszenie wartości o 1 można zapisać i-- zamiast i=i-1). Każde powtórzenie w pętli powoduje zwiększenie wartości zmiennej suma oraz ustalenie kolejnej wartości zmiennej skladnik (jako kolejnej liczby parzystej).

Okno programu *Suma* przedstawiono na rysunku 6.4.



Rys. 6.4. Wywołanie programu *suma.exe*

Warto wiedzieć

W języku C++ stosuje się również zapisy ++i oraz --i. Zapis ++i powoduje zwiększenie licznika zmiennej i jeszcze przed jej zastosowaniem, podczas gdy i++ powoduje zwiększenie licznika po jej zastosowaniu. Na przykład wartości i oraz a po wykonaniu kodu:

```
i=1;
a=i++;
```

będą odpowiednio równe: 2 i 1. Natomiast po wykonaniu kodu:

```
i=1;
a=++i;
```

będą odpowiednio równe: 2 i 2.

Analogicznie działają zapisy --i oraz i--.

Ćwiczenie 3

a. W pliku *suma.cpp* zapisz kod źródłowy programu *Suma*, skompiluj go i sprawdź działanie programu dla różnych wartości *n*.

b. Zmodyfikuj program *Suma* tak, aby wyznaczał sumę określonej liczby kolejnych liczb nieparzystych (począwszy od 1).

Warto wiedzieć

Suma kolejnych liczb nieparzystych jest kwadratem liczby składników:

$1 + 3 = 2^2$,

$1 + 3 + 5 = 3^2$,

$1 + 3 + 5 + 7 = 4^2$

itd.

Zapamiętaj

Pętla `while` jest wykonywana, dopóki jej warunek logiczny jest spełniony. Liczba powtórzeń nie jest określona z góry przez programistę.

Pętla `for` jest wykonywana określoną liczbę razy, ustaloną przez programistę lub przez użytkownika już po uruchomieniu programu.

6.3. Wyznaczamy noty sędziowskie

W ostatniej części tematu zmierzymy się z kolejną sytuacją problemową. Wykorzystamy umiejętności zdobyte wcześniej do opracowania odpowiedniego algorytmu i napiszemy program o bardziej rozbudowanej strukturze.

Rozważmy następujący problem: w konkursie skoków narciarskich chcemy wyznaczyć ocenę za styl, którą otrzyma dany zawodnik. Aby to zrobić, spośród pięciu ocen przyznanych przez pięciu sędziów należy odrzucić dwie skrajne noty: najwyższą i najniższą, a pozostałe zsumować.

Specyfikacja

Dane: pięć liczb z przedziału $[0; 20]$ podanych z dokładnością do 0,5.

Wynik: największa i najmniejsza liczba oraz suma liczb pomniejszona o liczbę najmniejszą i liczbę największą.

Warto wiedzieć

W statystyce stosuje się pojęcie średniej ucinanej. Średnia ta wyliczana jest na podstawie próbki wyników po odrzuceniu pewnej części wartości skrajnych – znacznie większych oraz znacznie mniejszych od pozostałych. W arkuszu kalkulacyjnym taką średnią wyznacza funkcja `ŚREDNIA.WEWN`.

Załóżmy, że nasz symulator wyznaczania oceny za styl będzie programem, który nie będzie zapamiętywał wszystkich pięciu liczb (not sędziowskich). Dzięki temu rozwiązanie stanie się bardziej uniwersalne: program będzie można łatwo zmodyfikować i zastosować do innego podobnego problemu, w którym stosuje się zasadę odrzucania wartości skrajnych (np. skrajnych wyników pomiarów w doświadczeniu przeprowadzanym na lekcji fizyki).

Algorytm będzie wyznaczał szukaną sumę przez odjęcie od sumy wszystkich pięciu liczb wartości najmniejszej oraz wartości największej. W przypadku gdy wartości skrajnych jest więcej, będziemy odejmować każdą tylko raz.

Punktacja w skokach narciarskich

Zawodnicy startujący w zawodach skoków narciarskich otrzymują za każdy skok ocenę, która składa się z dwóch części: punktów za uzyskaną odległość oraz noty za styl. Tę sumę koryguje się w zależności od warunków pogodowych i długości rozbiegu.



NOR	19.0
SUI	~~19.5~~
SVK	19.0
AUT	19.0
ITA	~~18.5~~
WIND:	+11.1
1	144.7

Noty sędziowskie za styl od 0 do 20 punktów: skrajne noty (najwyższa i najniższa) są usuwane, a pozostałe sumowane

Korekta punktacji ze względu na siłę i kierunek wiatru oraz długość rozbiegu

Suma punktów za odległość i styl z uwzględnieniem korekty

W zagadnieniu 6.1 analizowaliśmy przykład wyznaczania największej z liczb wprowadzonych z klawiatury. Zauważ, że tamto rozwiązanie można zaadaptować do obecnego problemu.

Oto propozycja listy kroków algorytmu:

1. Wczytaj pierwszą liczbę i przyjmij ją jako wartość minimum oraz jednocześnie wartość maksimum. Jako wartość sumy przyjmij wartość wczytanej liczby.
2. Powtórz cztery razy kroki od 3 do 6.
   3. Wczytaj kolejną liczbę.
   4. Zwiększ wartość sumy o wartość wczytanej liczby.
   5. Jeśli wczytana liczba jest mniejsza od minimum, to podstaw ją za minimum.
   6. Jeśli wczytana liczba jest większa od maksimum, to podstaw ją za maksimum.
7. Wypisz na ekranie minimum, maksimum oraz sumę pomniejszoną o minimum i maksimum.

Kod źródłowy programu może wyglądać następująco:

Kod źródłowy programu ● *Noty sędziowskie*

```
1.  #include <iostream>
2.
3.  using namespace std;
4.
5.  int main()
6.  {
7.      float nota,mini,maks,suma;
8.
9.      cout<<"Wprowadz noty za skok"<<endl;
10.     cin>>nota;
11.     mini=nota;
12.     maks=nota;
13.     suma=nota;
14.
15.     for (int i=1; i<=4; i++)
16.     {
17.         cin>>nota;
18.         suma=suma+nota;
19.         if (nota<mini) mini=nota;
20.         if (nota>maks) maks=nota;
21.     }
22.
23.     suma=suma-mini-maks;
24.     cout<<"\nNoty skrajne: "<<mini;
25.     cout<<" i "<<maks<<endl;
```

```
26.     cout<<"Nota za skok: "<<suma<<endl;
27.
28.     return 0;
29. }
```

W liniach 9–13 pobieramy pierwszą notę za skok i nadajemy tę wartość zmiennym `mini`, `maks` i `suma`. W linii 23 od sumy odejmujemy wartości najmniejszą i największą, aby uzyskać docelową notę.

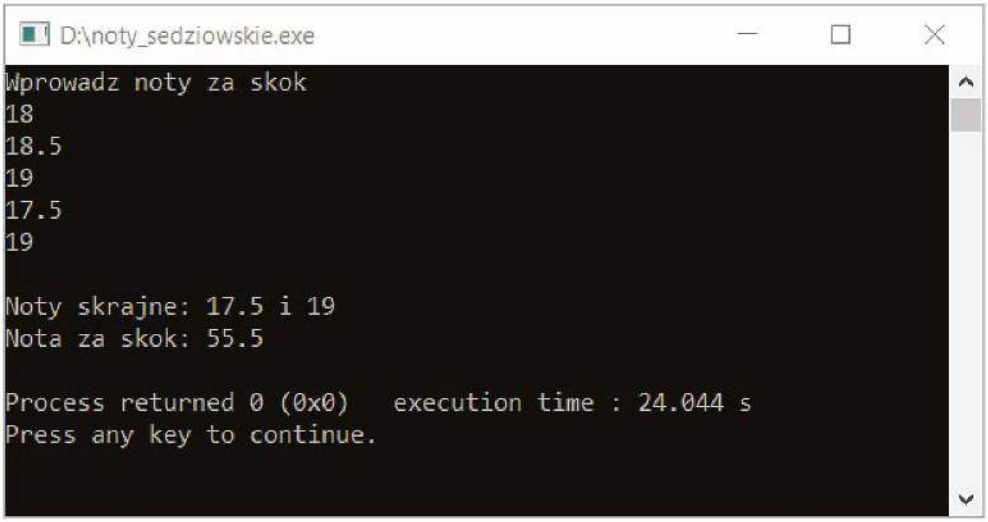
Kod programu wymaga jeszcze jednego dodatkowego wyjaśnienia: zwróć uwagę na wiersz 15. Zmienna `i`, czyli licznik pętli, została zadeklarowana dopiero w tym wierszu. Wielu programistów poleca taki sposób deklaracji zmiennych kontrolnych pętli, ponieważ:

- tak zadeklarowana zmienna ma tzw. mniejszy zasięg – poza kodem tworzącym daną pętlę **for** zmienna o takiej nazwie nie może być przypadkowo zastosowana (kompilator pokaże błąd),
- jest to wygodne – deklarujemy tę pomocniczą zmienną w momencie, gdy tworzymy kod dla pętli.

Okno programu *Noty sędziowskie* przedstawiono na rysunku 6.5.

Warto wiedzieć

W kodzie można użyć ponownie zmiennej o takiej samej nazwie, pod warunkiem że na nowo ją zadeklarujemy, np. w innej pętli **for**.



Rys. 6.5. Wywołanie programu *noty_sedziowskie.exe*

Ćwiczenie 4

a. Utwórz plik *noty_sedziowskie.cpp* i zapisz w nim kod źródłowy programu. Skompiluj kod i sprawdź działanie programu dla różnych zestawów danych.

b. Zmodyfikuj program tak, aby pozwalał obliczać sumę bez wartości skrajnych dla dowolnej liczby składników (wprowadzonych z klawiatury).

Podsumowanie

- Oprócz algorytmów liniowych i algorytmów z warunkami w programowaniu stosuje się algorytmy iteracyjne.
- Iteracyjne rozwiązywanie problemów polega na powtarzaniu jakiejś czynności skończoną liczbę razy, aż uzyska się oczekiwany efekt. Wykorzystuje się do tego instrukcje powtarzania (iteracyjne).
- W języku C++ instrukcjami powtarzania, pozwalającymi wielokrotnie wykonać ten sam fragment kodu, są `while` oraz `for`.
- Pętla `while` charakteryzuje się tym, że instrukcja wykonywana w pętli jest powtarzana, dopóki pewien warunek logiczny jest spełniony.
- W pętli `for` instrukcje wykonywane w pętli są powtarzane określoną liczbę razy.

Zadania

* **1** Napisz program, który wyznaczy najmniejszą liczbę z ciągu liczb całkowitych dodatnich podawanych z klawiatury. Przyjmij, że podanie liczby 0 kończy wprowadzanie liczb.

* **2** Napisz program wypisujący 10 kolejnych wielokrotności dodatniej liczby naturalnej wczytanej z klawiatury.

* **3** Zmodyfikuj kod źródłowy programu *Logowanie* z tematu 5 (s. 106) tak, aby program kończył działanie po trzech nieudanych próbach logowania.

* **4** Zmodyfikuj program *Najdłuższy skok* tak, aby trener mógł dodatkowo zliczać liczbę skoków nieważnych (spalonych).

* **5** Napisz program, który będzie obliczać średnią wartość (obciętą do części całkowitej) wszystkich liczb wprowadzonych z klawiatury, z pominięciem liczb najmniejszej i największej.

* **6** Przeanalizuj dokładnie działanie kodu źródłowego programu *Najdłuższy skok* na trzech konkretnych przykładach. Efekty przedstaw w tabeli w pliku tekstowym:

a	maks
4	4
2	
5	5
7	7
5	
0	

a	maks
8	
6	
2	
8	
9	
0	

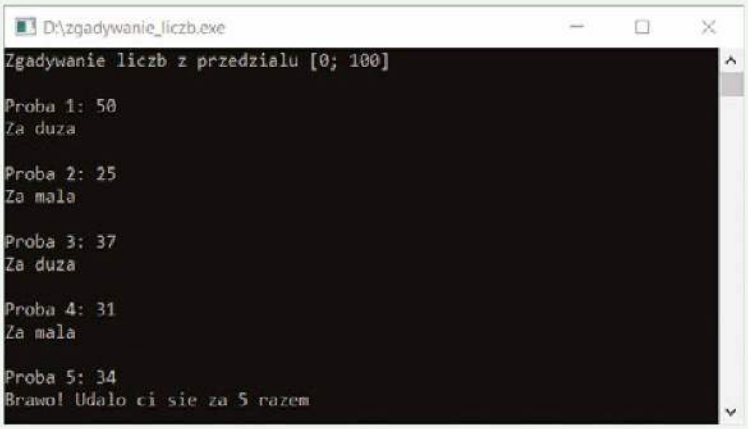
a	maks
12	
9	
12	
9	
12	
0	

** **7** Napisz program, który zliczy osobno, ile liczb ujemnych i liczb dodatnich wprowadzono z klawiatury. Przyjmij, że wczytanie liczby 0 kończy wczytywanie liczb.

** **8** Napisz program, który po wczytaniu 10 liczb wskaże te, które potraktowane jako długości boków utworzą prostokąt o największym polu.

** **9** Napisz kod programu, który pokaże różnicę w działaniu operatorów `++i` oraz `i++` dla zmiennej `i` typu całkowitego.

** **10** Napisz program, który będzie grą w zgadywanie ustalonej liczby z przedziału od 0 do 100. Program powinien działać do momentu podania przez użytkownika właściwej liczby. Po każdej próbie odgadnięcia liczby program powinien wypisać jeden z komunikatów: „Za mała", „Za duża" lub „Brawo! Udało ci się za *x* razem", gdzie *x* oznacza liczbę prób, które wykonał użytkownik (rysunek poniżej).



** **11** Napisz program, który będzie wczytywał podane przez użytkownika jedno- lub dwucyfrowe liczby całkowite i sumował je do momentu otrzymania sumy przekraczającej 123. Program powinien zwrócić informację o liczbie składników sumy i wartość sumy.

** **12** Napisz program, który wypisze pierwszych 100 liczb całkowitych dodatnich podzielnych przez:

a. 3,

b. 7,

c. 3 i 7.

*** **13** Zmodyfikuj program *Najdłuższy skok* tak, aby dodatkowo znajdował najdłuższą serię, w której każdy kolejny skok był dłuższy od poprzedniego, a jako wynik wyświetlał liczbę skoków w tej serii. Na przykład dla skoków o długościach 6,20; 6,30; 6,28; 6,33; 6,40; 6,30; 6,23 program powinien wyświetlić odpowiedź 3.

*** **14** Dla sześciu liczb wprowadzonych z klawiatury zawsze wystarczy 7 porównań, aby wyznaczyć najmniejszą oraz największą z nich. Jak to zrobić? Zapisz algorytm w postaci listy kroków, a następnie jego realizację w języku programowania.

*** **15** Napisz program, który pozwoli rozstrzygnąć, czy wśród *n* liczb całkowitych dodatnich wprowadzonych z klawiatury była taka trójka liczb *x*, *y* i *z*, dla których nie da się zbudować trójkąta o bokach długości *x*, *y*, *z*.

7. Binarny system liczbowy

Kiedy korzystasz z komputera lub smartfona, wykonujesz wiele różnych czynności, m.in. słuchasz muzyki, przetwarzasz zdjęcia, redagujesz teksty, wysyłasz e-maile i oglądasz transmisje wideo. To wszystko jest możliwe dzięki temu, że zarówno oprogramowanie, jak i różnego typu dane są zapisywane w pamięci komputera w jednolity sposób – cyfrowo, czyli jako liczby. Z tego tematu dowiesz się, jaką postać mają takie liczby.

Cele lekcji

- Dowiesz się, jak się zapisuje liczby w binarnym systemie liczbowym.
- Przypomnisz sobie, czym są bit i bajt.
- Zrozumiesz istotę cyfrowego zapisu informacji.
- Poznasz algorytmy zamiany reprezentacji liczby między systemami dziesiętnym i dwójkowym.
- Nauczysz się korzystać ze struktury tablicy w języku C++.
- Zaprogramujesz algorytm konwersji liczby z systemu dziesiętnego na dwójkowy.

7.1. Liczby dziesiętne i dwójkowe

System dziesiętny (decymalny)

Gdy na co dzień wykorzystujemy dane liczbowe, posługujemy się zazwyczaj **systemem dziesiętnym** (**decymalnym**). Dlaczego właśnie ten system się upowszechnił? Pewnie dlatego, że pierwszymi liczydłami były palce u dłoni człowieka. Liczby zapisujemy więc za pomocą dziesięciu cyfr: 0, 1, 2, 3, 4, 5, 6, 7, 8 i 9.

System pozycyjny

Naturalne jest dla nas, że zapis 2350 oznacza liczbę dziesięć razy większą od 235. Wynika to z faktu, że system dziesiętny jest **systemem pozycyjnym**. To oznacza, że z pozycją cyfry (liczoną od prawej strony) wiąże się jej waga – w tym wypadku: 1, 10, 100, 1000 itd. Są to kolejne

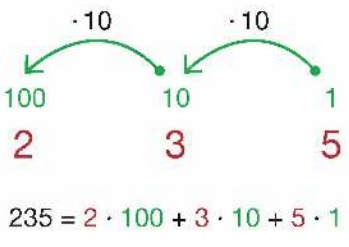
Podstawa systemu liczbowego

potęgi liczby 10, która dla kodu decymalnego jest **podstawą systemu liczbowego**. Każdą liczbę przedstawia się w tym systemie jako sumę wielokrotności potęg liczby 10. Na przykład liczba 235 jest sumą dwóch potęg 10^2, trzech potęg 10^1 i pięciu jedności (czyli potęg 10^0).

Na rysunku 7.1 pokazano zapis liczby 235 w systemie dziesiętnym.

Dziesiętny system pozycyjny
Podstawa systemu: 10
Cyfry: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Wagi cyfr na kolejnych pozycjach są kolejnymi potęgami liczby dziesięć: 10^0, 10^1, 10^2, 10^3, ...



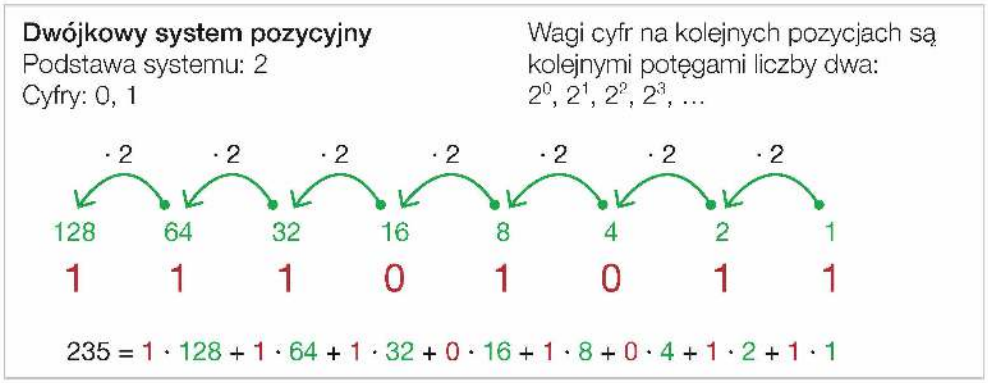
Rys. 7.1. Zapis dziesiętny liczby 235

W technice komputerowej szczególne znaczenie ma **system dwójkowy (binarny)**, w którym występują tylko dwie cyfry: 0 i 1. Wagi cyfr na kolejnych pozycjach w zapisie liczby w systemie dwójkowym są potęgami liczby 2. Na rysunku 7.2 przedstawiono binarną postać liczby 235.

System dwójkowy (binarny)

Warto wiedzieć

O liczbie przedstawionej w systemie dwójkowym mówi się w skrócie: liczba dwójkowa.



Rys. 7.2. Zapis dwójkowy liczby 235

Zauważ, że liczba 235 jest sumą 2^7, 2^6, 2^5, 2^3, 2 oraz 1 i jest – w zapisie dwójkowym – liczbą ośmiocyfrową 11101011.

Tabela 7.1 zawiera zapis binarny liczb od 0 do 15. Możemy na tej podstawie sformułować kilka **podstawowych własności liczb binarnych**:

1. Liczby nieparzyste w zapisie binarnym kończą się cyfrą 1.
2. Liczby binarne kończące się cyfrą 0 dzielą się przez 2, a więc są parzyste.
3. Aby uzyskać liczbę binarną dwa razy większą od wyjściowej, należy na końcu dopisać cyfrę 0. Oznacza to, że jeśli usuniemy ostatnią cyfrę parzystej liczby binarnej, otrzymamy połowę tej liczby.
4. Zapisy dwóch kolejnych liczb binarnych – mniejszej parzystej i większej nieparzystej – różnią się tylko ostatnią cyfrą. Liczba parzysta kończy się cyfrą 0, a nieparzysta – cyfrą 1, np. 12 to 1100, a 13 to 1101.

Takich własności można sformułować więcej i wszystkie można matematycznie udowodnić.

Podstawowe własności liczb binarnych

Liczba	Zapis binarny	Uzasadnienie
0	0	0
1	1	2^0
2	10	2^1
3	11	$2^1 + 2^0$
4	100	2^2
5	101	$2^2 + 2^0$
6	110	$2^2 + 2^1$
7	111	$2^2 + 2^1 + 2^0$
8	1000	2^3
9	1001	$2^3 + 2^0$
10	1010	$2^3 + 2^1$
11	1011	$2^3 + 2^1 + 2^0$
12	1100	$2^3 + 2^2$
13	1101	$2^3 + 2^2 + 2^0$
14	1110	$2^3 + 2^2 + 2^1$
15	1111	$2^3 + 2^2 + 2^1 + 2^0$

Tabela 7.1. Zapis binarny liczb od 0 do 15

Dobra rada

Liczb dwójkowych nie nazywaj tak samo jak liczb w systemie dziesiętnym. Na przykład liczbę zapisaną jako 1010 w systemie dziesiętnym nazywamy „tysiąc dziesięć", ale w systemie binarnym należy ten zapis odczytać „jeden zero jeden zero".

Warto wiedzieć

Liczby większe od 1 zapisane w postaci binarnej składają się z większej liczby cyfr niż ich odpowiedniki w postaci dziesiętnej. Z punktu widzenia człowieka to z pewnością wada, dlatego binarnym systemem liczbowym nie posługujemy się na co dzień.

Zapamiętaj

Dwójkowy system liczbowy (podobnie jak system dziesiętny) opiera się na zasadzie zapisu wagowo-pozycyjnego. Cyfry w zapisie liczby znajdują się na ponumerowanych pozycjach. Każdej pozycji (licząc od prawej do lewej) odpowiada pewna waga: 1, 2, 4, 8 itd. Wagi są kolejnymi potęgami liczby 2, którą nazywa się podstawą tego systemu liczbowego.

Ćwiczenie 1

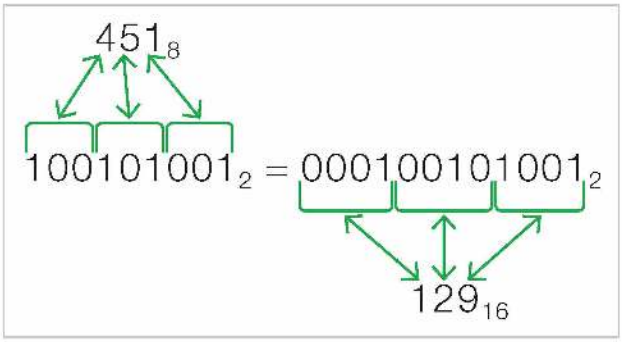
Jaką największą liczbę można zapisać w systemie dwójkowym za pomocą ośmiu cyfr?

Inne systemy liczbowe

System ósemkowy
System szesnastkowy

W informatyce często wykorzystuje się również systemy ósemkowy i szesnastkowy. W **systemie ósemkowym** liczby tworzy się z ośmiu cyfr (od 0 do 7), natomiast w **systemie szesnastkowym** jest ich aż szesnaście: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Do zapisu sześciu ostatnich cyfr wykorzystuje się znaki będące kolejnymi literami alfabetu łacińskiego. Aby oznaczyć, w jakim systemie została zapisana liczba, w indeksie dolnym zapisuje się podstawę systemu (rys. 7.3).

Podstawy systemów szesnastkowego i ósemkowego są potęgami liczby 2, dlatego stosunkowo łatwo można zamienić liczby zapisane w tych systemach na reprezentację w systemie binarnym i odwrotnie. Każdą cyfrę ósemkową można zapisać za pomocą trzech cyfr binarnych, a każdą cyfrę szesnastkową – za pomocą czterech cyfr binarnych, przy czym mniejsze wartości uzupełniamy z lewej strony zerami (rys. 7.3).



Rys. 7.3. Zamiana zapisu liczby pomiędzy systemami dwójkowym, ósemkowym i szesnastkowym

Warto wiedzieć

System ósemkowy wykorzystuje się m.in. do ustalania praw do pliku w systemach Linux. System szesnastkowy ma zastosowanie np. w adresach IPv6.

7.2. Cyfrowy zapis informacji

Wszelkiego typu informacje (liczby, teksty, grafika, dźwięk itd.) w systemach komputerowych są przedstawiane (kodowane) jako liczby.

Bit

Najmniejszą, niepodzielną porcją informacji jest **bit** (od ang. ***binary digit***). Pojedynczy bit może przyjmować dwie wartości – zapisywane umownie jako 0 albo 1. Są one cyfrową reprezentacją pewnych fizycznych właściwości nośnika danych lub medium transmisyjnego, np. właściwości elektrycznych lub magnetycznych.

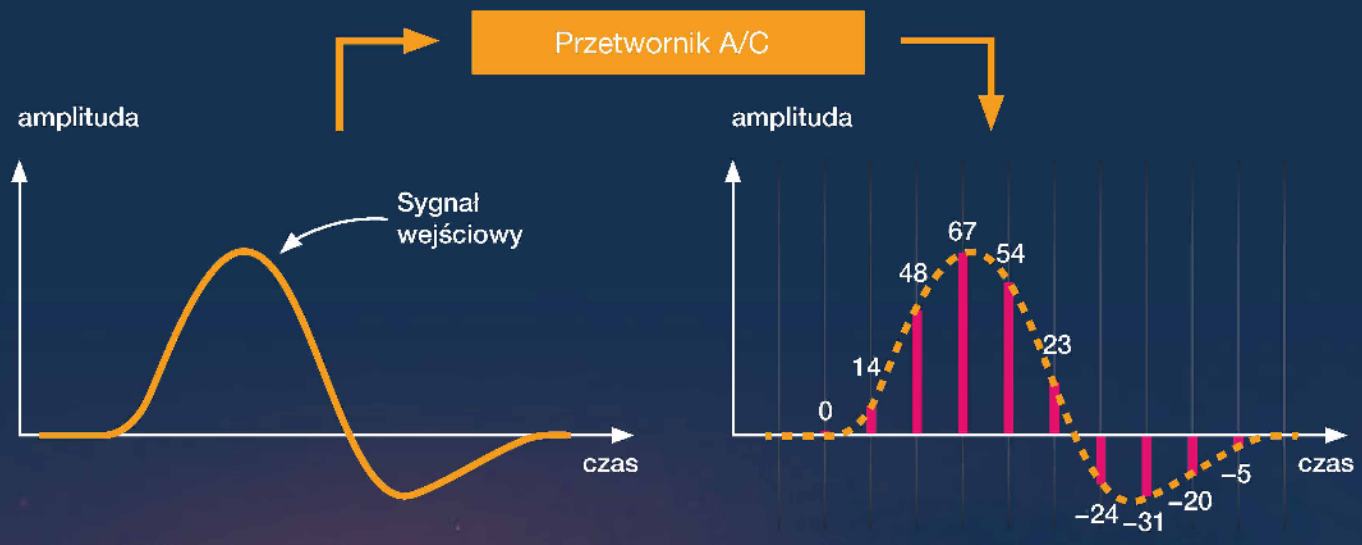
Jak dane analogowe są przekształcane na cyfrowe

Podczas nagrywania filmu smartfonem dźwięk i ruchomy obraz są przetwarzane na postać cyfrową. Dzięki temu możemy przechowywać je w pamięci telefonu, a następnie odtwarzać. Jak działają cyfrowe urządzenia rejestrujące dźwięk i obraz? Przeanalizujmy to na przykładzie.



Mikrofon wbudowany w telefon zamienia falę dźwiękową na sygnały elektryczne odpowiadające amplitudzie i częstotliwości dźwięku.

Przetwornik A/C (analogowo-cyfrowy) próbkuje amplitudy sygnałów elektrycznych, czyli mierzy je tysiące razy w ciągu jednej sekundy i zamienia na ciągi liczb binarnych. Aby zapis cyfrowy był zgodny z dźwiękiem, który słyszy ludzkie ucho, próbkowanie musi się odbywać z odpowiednio wysoką częstotliwością i rozdzielczością. Na przykład dla płyt CD standardem są częstotliwość 44,1 kHz i rozdzielczość 16 bitów na próbkę, czyli $2^{16} = 65\,536$ możliwych wartości amplitudy fali dźwiękowej.

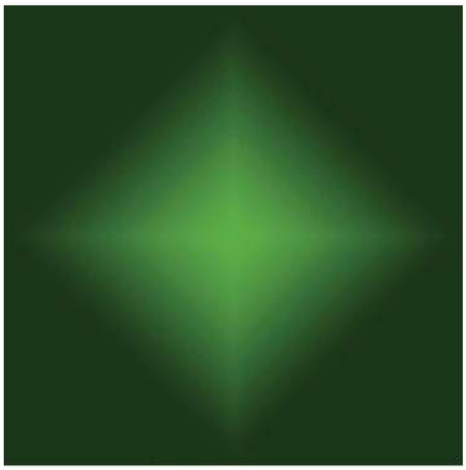


Jak w komputerze zapisuje się obrazy cyfrowe?

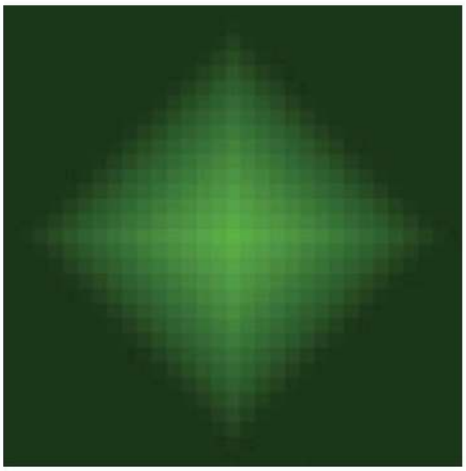
Aby wyjaśnić dokładniej istotę cyfrowego zapisu informacji, posłużymy się przykładem z dziedziny grafiki komputerowej.

Rysunek 7.4 to przykład gradientu, czyli wypełnienia powierzchni równomiernym przejściem z jednego odcienia tej samej barwy do innego.

Rysunek 7.5 powstał z rysunku 7.4 w wyniku ograniczenia liczby użytych odcieni koloru do 16 i zmniejszenia liczby pikseli.



Rys. 7.4. Gradient



Rys. 7.5. Gradient po ograniczeniu liczby pikseli i odcieni

Rozdzielczość obrazu, podręcznik *Informatyka na czasie 1. Zakres podstawowy*, s. 188

Dzięki zastosowaniu odpowiedniej **rozdzielczości** mamy wrażenie, że obraz z rysunku 7.4 jest ciągły – kolory płynnie przechodzą jeden w drugi. Obraz z rysunku 7.5 sprawia wrażenie zbioru kwadratów o różnych odcieniach zieleni, które są ułożone schodkowo.

Przyjrzyjmy się dokładniej rysunkowi 7.5. W poziomie i pionie mamy po 31 pikseli. Na rysunku 7.6 pokazano środkową linię tego obrazu w powiększeniu:



Rys. 7.6. Środkowa linia obrazu w powiększeniu

Warto wiedzieć

Czasami zamiast słowa piksel używa się słowa pel (ang. ***p**icture **el**ement*). Stosuje się je zwłaszcza wtedy, gdy mówimy o elementach obrazu wydrukowanego.

W cyfrowej reprezentacji każda informacja, również informacja o obrazie, musi mieć postać liczbową. Każdemu z odcieni można przypisać liczbę całkowitą, począwszy od 0.

Jeśli przyjąć, że odcień o najmniejszej intensywności (jasności) koloru zielonego opiszemy liczbą 0, a ten o największej – 15, to piksele tworzące tę linię obrazu opiszemy ciągiem 31 liczb. Przedstawiono to na rysunku 7.7.



Rys. 7.7. Środkowa linia obrazu z ponumerowanymi odcieniami

Każda z tych liczb dziesiętnych w pamięci komputera lub na nośniku danych jest w rzeczywistości zapisywana jako czterocyfrowa liczba dwójkowa. Aby zrozumieć, w jaki sposób tworzone są kody binarne kolorów, przeanalizujmy diagram z rysunku 7.8.

Różnym odcieniom koloru zielonego przyporządkowano liczby binarne od 0000 do 1111. **Najbardziej znaczący bit** – reprezentowany przez pierwszą od lewej cyfrę liczby binarnej – decyduje o tym, czy dany odcień jest w grupie ciemniejszych czy jaśniejszych. Diagram z rysunku 7.8 ma formę tzw. **drzewa decyzyjnego** – czytamy je od góry. Jeśli postawilibyśmy pytanie „Czy dany odcień jest jednym z jaśniejszych?", to odpowiedź „tak" jest reprezentowana przez cyfrę 1, a odpowiedź „nie"– przez 0. Dlatego wszystkie odcienie bliższe czerni mają pierwszy bit o wartości 0.

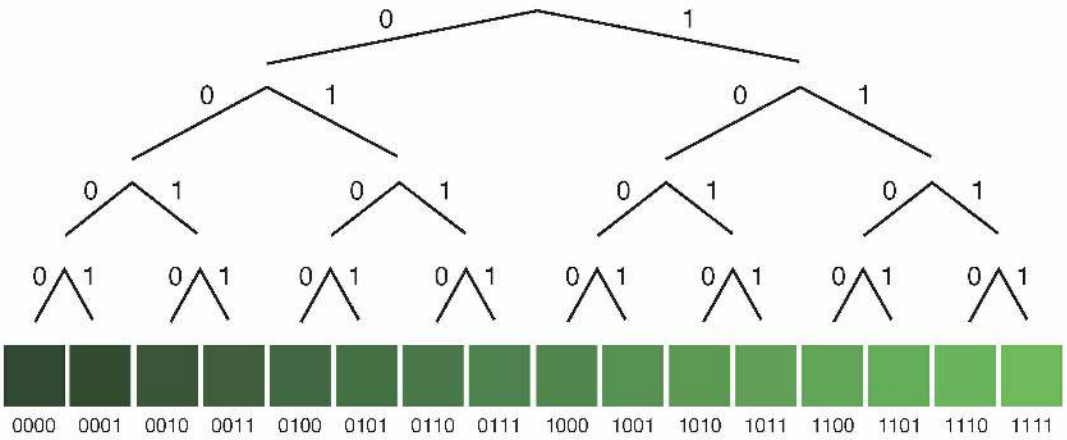
Bit najbardziej znaczący

Drzewo decyzyjne

Drugi bit określa, czy odcień jest wśród $\frac{1}{4}$ najciemniejszych czy wśród $\frac{1}{4}$ średnio ciemnych (lub analogicznie $\frac{1}{4}$ najjaśniejszych czy w $\frac{1}{4}$ średnio jasnych).

Po wyznaczeniu trzeciego bitu wybór organicza się do dwóch odcieni, a **najmniej znaczący bit** – reprezentowany przez ostatnią cyfrę liczby binarnej – pozwala jednoznacznie określić odcień docelowy.

Bit najmniej znaczący



Rys. 7.8. Diagram ilustrujący tworzenie kodów odcieni w obrazie

Każdy odcień koloru zielonego na obrazie można więc jednoznacznie przedstawić jako czterocyfrową liczbę binarną. Do opisania jednego piksela obrazu z rysunku 7.7 potrzeba zatem 4 bitów.

Zapamiętaj

Wszelkiego typu informacje przechowywane i przetwarzane w systemach komputerowych są przedstawiane (kodowane) jako liczby binarne, składające się z dwóch cyfr: 0 i 1. Cyfry te reprezentują dwa fizyczne stany, np. magnetyczne, elektryczne lub optyczne.

Warto wiedzieć

W pierwszych komputerach w latach 40. XX w. nie korzystano z systemu binarnego. W słynnym komputerze ENIAC zastosowano system dwupiątkowy. Każdą cyfrę dziesiętną zastępowano dwoma cyframi: jedną o wartości 0, 1, 2, 3 lub 4, drugą o wartości 0 lub 5, tak aby suma tych dwóch cyfr dawała pierwotną cyfrę dziesiętną.

7.3. Algorytmy zamiany reprezentacji liczb

W komputerze liczby zapisuje się w systemie dwójkowym, jednak użytkownicy są przyzwyczajeni do używania systemu dziesiętnego. Dlatego np. podczas pracy z kodem HTML, tworzenia grafiki komputerowej czy konfiguracji sieci pojawia się potrzeba konwersji między systemami liczbowymi.

Zamiana liczby binarnej na liczbę dziesiętną

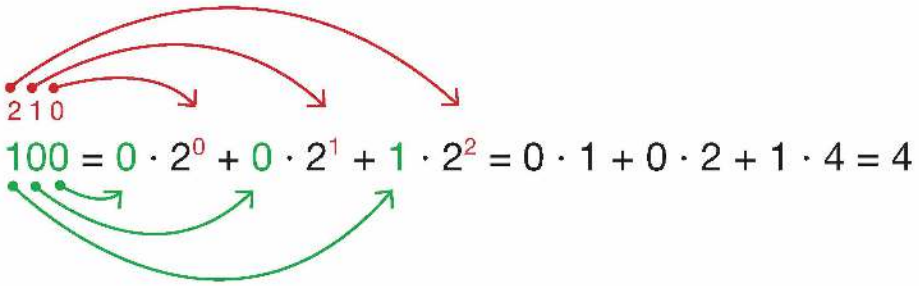
Zapis dziesiętny liczby binarnej można wyznaczyć, sumując kolejne potęgi liczby 2 odpowiadające wagom kolejnych cyfr. Algorytm można zapisać następująco:

1. Ponumeruj cyfry liczby binarnej od prawej do lewej, zaczynając od 0.
2. Każdą cyfrę potraktuj odpowiednio jako liczbę dziesiętną 0 albo 1 i pomnóż przez liczbę 2 podniesioną do takiej potęgi, jaki jest numer cyfry.
3. Zsumuj otrzymane iloczyny.

Dobra rada

Jeśli posługujesz się jednocześnie przynajmniej dwoma systemami liczbowymi, stosuj dodatkowy indeks dolny i określaj w nim podstawę systemu, w którym liczba została zapisana. Jest to szczególnie ważne w przypadku liczb, których postaci wyglądają tak samo w różnych systemach, ale mają różne wartości, np.: $100_{10} = 1100100_2$, a $100_2 = 4_{10}$.

Zamianę liczby binarnej 100 na postać dziesiętną można zilustrować w następujący sposób (rys. 7.9):



Rys. 7.9. Schemat zamiany liczby binarnej na dziesiętną

Ćwiczenie 2

Zapisz poniższe liczby binarne w systemie dziesiętnym.

a. 111
b. 10001
c. 1101101

Zamiana liczby dziesiętnej na liczbę binarną

Załóżmy, że chcemy zapisać w postaci binarnej liczbę dziesiętną 123. Aby wyznaczyć kolejne cyfry (bity) zapisu binarnego, musimy wskazać potęgi liczby 2, które w sumie dają tę liczbę.

Zacznijmy od wyznaczenia najmniej znaczącego, czyli ostatniego bitu. Z podanych wcześniej **własności liczb binarnych** wiemy, że wszystkie liczby nieparzyste kończą się cyfrą 1 (własność nr 1), dlatego zapis binarny liczby 123 będzie się kończyć cyfrą 1. Liczby 122 i 123

Podstawowe własności liczb binarnych, s. 127

to kolejne liczby całkowite, więc z własności nr 4 wiemy, że wszystkie bity poza ostatnim będą takie same jak bity liczby 122. Przedostatni bit liczby 122 (a więc i 123) będzie taki jak ostatni bit liczby o połowę mniejszej, a więc 61 (wynika to z własności nr 3). Ta liczba jest nieparzysta, więc jej postać binarna będzie się kończyć cyfrą 1. To oznacza, że zapis binarny liczby 122 kończy się parą cyfr 10 (jeden zero), a liczby 123 – parą 11 (jeden jeden).

Jak znaleźć kolejną cyfrę (trzecią od prawej) zapisu binarnego liczby 123? Wystarczy powtórzyć powyższe rozumowanie: określić parzystość liczby (61 – 1)/2. Wyznaczanie parzystości kolejnych ilorazów należy powtarzać do momentu, gdy otrzymamy iloraz równy 0.

Przebieg tego algorytmu można przedstawić w formie tabeli:

Liczba	Iloraz z dzielenia przez 2	Reszta z dzielenia przez 2
123	61	1
61	30	1
30	15	0
15	7	1
7	3	1
3	1	1
1	0	1

Tabela 7.2. Przebieg algorytmu zapisu liczby 123 w postaci binarnej

Dobra rada

Kolejne etapy realizacji danego algorytmu możesz w wygodny sposób umieścić w tabeli. Może ona pomóc w zrozumieniu trudniejszego algorytmu oraz pozwala znaleźć ewentualne błędy.

Ostatnią kolumnę czyta się od dołu i zapisuje cyfry od lewej do prawej. Liczba 123 w zapisie binarnym to 1111011.

A to ciekawe

Komputery analogowe

Maszyny analogowe, inaczej niż urządzenia cyfrowe, przetwarzały dane w postaci analogowej, tj. za pomocą wielkości fizycznych, zmieniających się w sposób ciągły, takich jak napięcie elektryczne. Komputerem analogowym był, prawdopodobnie pierwszy na świecie, analizator równań różniczkowych AKAT-1, zaprojektowany i skonstruowany w 1957 r. przez zespół pod kierunkiem inż. Jacka Karpińskiego. Analizator pomagał tworzyć złożone systemy opisujące dynamicznie zmieniające się procesy, takie jak przepływy cieczy, wymiana ciepła, resorowanie pojazdów. Ergonomiczną obudowę komputera zaprojektowano na warszawskiej Akademii Sztuk Pięknych.



Dobra rada

Jeśli chcesz wyznaczyć postać binarną wielu liczb naraz, możesz skorzystać z funkcji `DZIES.NA.DWÓJK` lub `DEC2BIN` w arkuszu kalkulacyjnym.

Ćwiczenie 3

Wyznacz postać binarną liczby dziesiętnej 117. Zapisz przebieg algorytmu w tabeli. Jego poprawność możesz sprawdzić np. w kalkulatorze systemu Windows (tryb programisty) albo korzystając z odpowiedniej aplikacji w telefonie lub serwisu internetowego.

7.4. Program zamieniający liczbę dziesiętną na binarną

Przygotujemy się teraz do stworzenia programu, który po wpisaniu z klawiatury liczby naturalnej, nie większej niż 255, wypisze ciąg zer i jedynek przedstawiający binarną reprezentację komputerową tej liczby. Ograniczymy się do liczb, które da się zapisać w jednym **bajcie** (ang. *byte*), czyli jako liczby binarne co najwyżej ośmiocyfrowe, złożone z ośmiu bitów.

Bajt

Zapiszmy specyfikację problemu algorytmicznego.

Warto wiedzieć

Jeden bajt reprezentuje aż $2^8 = 256$ różnych liczb binarnych. Tę właściwość wykorzystano w kodzie ASCII, w którym za pomocą jednobajtowych kodów przedstawia się litery alfabetu łacińskiego (języka angielskiego), cyfry, znaki przestankowe, symbole i inne znaki służące do komunikacji tekstowej.

Specyfikacja

Dane: liczba całkowita (nieujemna) nie większa niż 255.

Wynik: zapis binarny liczby całkowitej.

Aby wyznaczyć postać binarną liczby dziesiętnej, możemy postępować w sposób przedstawiony w tabeli 7.2 ze s. 133. Wystarczy zapamiętywać kolejne ilorazy z dzielenia przez 2 oraz otrzymywane reszty z dzielenia. Dokładny zapis działania algorytmu przedstawia poniższa lista kroków.

1. Podziel liczbę przez 2, zapisz resztę z dzielenia i zapamiętaj iloraz (część całkowitą).
2. Dopóki ostatnio otrzymany iloraz jest większy od 0, powtarzaj kroki 3, 4 i 5.
   3. Wyznacz resztę z dzielenia ostatnio zapamiętanego ilorazu przez 2.
   4. Zapisz resztę z dzielenia przed ostatnią wcześniej zapisaną resztą.
   5. Podziel iloraz przez 2 i zapamiętaj część całkowitą wyniku jako nowy iloraz.

Po otrzymaniu ilorazu zerowego postać binarną można utworzyć z zapamiętanych reszt zapisanych od prawej do lewej w kolejności ich pojawiania się.

Do zapisania algorytmu w języku C++ potrzebna będzie nam nowa wiedza, ponieważ będziemy musieli wyznaczać reszty z dzielenia oraz zapamiętywać kolejne liczby. Do tego celu wykorzystamy operator wyznaczania reszty z dzielenia oraz strukturę tablicy.

Dzielenie całkowite i reszta z dzielenia

Program ***Przeliczanie temperatury*** z tematu 5 zawierał operację dzielenia na liczbach całkowitych. Wynik tego działania był obcięty do części całkowitej według zasady: jeśli dzielna i dzielnik są typu `int`, to iloraz też musi być typu `int`. Jeśli więc użyjemy deklaracji `int iloraz=123/2;`, zmiennej `iloraz` zostanie przypisana wartość 61. Ta wiedza przyda nam się podczas zapisywania kroku algorytmu zamiany, w którym dzielimy ostatnio wyznaczony iloraz przez 2.

Program *Przeliczanie temperatury*, s. 104

Do wyznaczania reszty z dzielenia służy operator %, zwany **modulo**. Na przykład wartością zmiennej `reszta` po deklaracji i przypisaniu `int reszta = 123%2;` będzie 1.

Modulo to operacja wyznaczania reszty z dzielenia jednej liczby przez drugą.

Ćwiczenie 4

Sprawdź efekt działania programu, którego kod wewnątrz funkcji `main` ma postać:

```
int iloraz, reszta;
iloraz = 123/2;
reszta = 123%2;
cout<<iloraz<<endl;
cout<<reszta<<endl;
```

Warto wiedzieć

Reszta z dzielenia dowolnej liczby całkowitej przez liczbę k może przyjąć wartości od 0 do $k-1$.

Tablica – struktura danych

Do przechowywania określonej liczby elementów tego samego typu można posłużyć się strukturą danych nazywaną **tablicą** (ang. *array*).

Tablica

Aby korzystać z tablicy, trzeba ją wcześniej zadeklarować, czyli określić jej nazwę, rozmiar i typ przechowywanych informacji. Na przykład tablicę o nazwie `tab`, w której będziemy przechowywać wartości ośmiu liczb całkowitych, zadeklarujemy w C++ następująco:

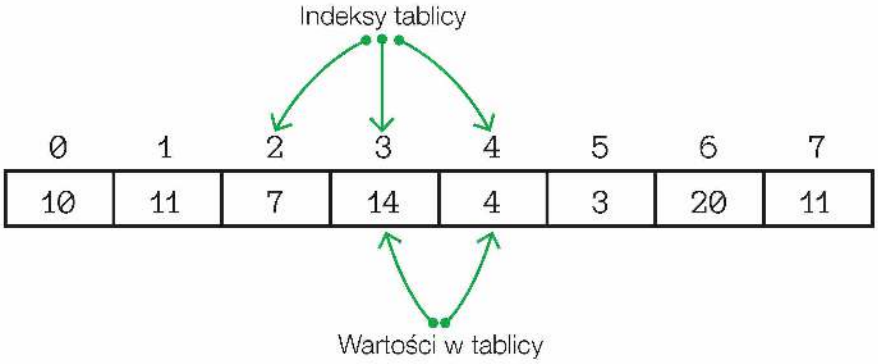
```
int tab[8];
```

Dobra rada

10-elementową tablicę liczb rzeczywistych (typu `float`) możesz zadeklarować następująco:
`float tablica[10];`

Tablicę można porównać do jednowierszowej tabeli, w której komórkach zapisujemy kolejne liczby. Każda komórka tablicy ma numer, zwany **indeksem**. Komórki numeruje się od 0 (rys. 7.10).

Indeks



Rys. 7.10. Interpretacja graficzna tablicy 8-elementowej

Aby odwołać się do jakiegoś elementu tablicy, wystarczy podać jego współrzędną (indeks). Instrukcje:

```
tab[0] = 10;
tab[7] = 11;
```

powodują przypisanie wartości 10 pierwszemu elementowi tablicy (o indeksie 0) i przypisanie wartości 11 ostatniemu elementowi (o indeksie 7).

Nadawanie wartości początkowych tablicy

Dobra rada

Jeśli wyświetlisz zawartość tablicy, której nie nadano żadnych wartości, to otrzymasz w pewnym sensie losowe liczby – takie, które w momencie utworzenia tablicy były przechowywane w tym fragmencie pamięci komputera. Dlatego warto rozważyć nadanie wartości 0 elementom tablicy już przy jej deklaracji.

W momencie deklarowania tablicy można określić wartości początkowe jej elementów. Można to zrobić na różne sposoby:

```
1. int a[4] = {0};
2. int b[4] = {86, 12, 24, 1000};
3. int c[4] = {3, 5};
```

Czteroelementowa tablica o nazwie a z wiersza 1 będzie wypełniona zerami, tablica b z wiersza drugiego będzie przechowywała wartości 86, 12, 24 i 1000. Dwa pierwsze elementy tablicy c z wiersza trzeciego będą przechowywały odpowiednio wartości 3 i 5, a pozostałe elementy – będą przechowywały wartość 0.

Tablice – wczytywanie i wypisywanie danych

Napiszemy krótki program o nazwie *Cyfry wspak*, który po wczytaniu z klawiatury ośmiu liczb jednocyfrowych wyświetli te liczby w odwróconej kolejności. Do wykonania operacji wczytywania i wypisywania danych użyjemy instrukcji **for**.

Zasadniczy fragment programu, czyli kod znajdujący się wewnątrz funkcji main, będzie miał postać:

Fragment kodu źródłowego programu *Cyfry wspak*

```
1. int tab[8];
2. for(int i=0; i<=7; i++)
3.     cin >> tab[i];
4.
5. for(int j=7; j>=0; j--)
6.     cout << tab[j] << " ";
```

W wierszu 1 znajduje się deklaracja tablicy, w której można przechować liczby całkowite (typu **int**). W wierszach 2 i 3 zapisano kod pętli, która służy do wprowadzenia 8 liczb z klawiatury: kolejne liczby będą zapisywane w komórkach o indeksach: 0, 1, 2, ..., 7. Do wskazania położenia liczby w tablicy użyto zmiennej pomocniczej i, która jest jednocześnie licznikiem pętli. Ponieważ liczby mają być przez program

wypisane w odwróconej kolejności, w pętli z wierszy 5 i 6 wypisywanie zawartości tablicy rozpoczyna się od elementu o indeksie 7. Do kontroli liczby iteracji i jednocześnie ustalenia pozycji liczby w tablicy służy zmienna pomocnicza j.

Ćwiczenie 5

a. Sprawdź działanie powyższego kodu.
b. Przyjmij, że liczby w tablicy powinny być zapamiętane w kolejności odwróconej. Napisz program realizujący takie założenie.

Program *Zamiana systemów liczbowych*

Teraz możemy już napisać w języku C++ cały program zamieniający liczbę dziesiętną na liczbę dwójkową. Poniżej znajduje się kompletny kod programu *zamiana.cpp*, realizującego przedstawiony wcześniej algorytm.

Kod źródłowy programu *Zamiana*

```
1.  #include <iostream>
2.
3.  using namespace std;
4.
5.  int main()
6.  {
7.      int liczba, iloraz, cyfry[8]={0};
8.
9.      cout<<"Podaj liczbe: "; cin>>liczba;
10.
11.     cyfry[7] = liczba%2;
12.     iloraz = liczba/2;
13.
14.     int i=1;
15.     while(iloraz>0)
16.     {
17.         cyfry[7-i] = iloraz%2;
18.         iloraz = iloraz/2;
19.         i++;
20.     }
21.
22.     cout << "Zapis binarny liczby " << liczba << " to: ";
23.     for(int i=0; i<=7; i++) cout << cyfry[i];
24.     cout << endl;
25.
26.     return 0;
27. }
```

W wierszu 7 deklarujemy zmienne oraz tablicę, której wartości zerujemy. Wiersze 14–20 są realizacją w języku C++ algorytmu zamiany liczby dziesiętnej na binarną w postaci listy kroków (s. 134). Linie 22–24 odpowiadają za wyświetlenie postaci binarnej.

Reszty z dzielenia zapamiętywane są od razu w kolejności odwróconej, dlatego pierwsza reszta jest zapisywana jako ostatni element tablicy `cyfry` (wiersz 11). W linii 15 zastosowano pętlę `while` z warunkiem z kroku 2 algorytmu zamiany liczby dziesiętnej na liczbę binarną (s. 134). Zapis `cyfry[7-i]` pozwala na poprawne ustalenie pozycji kolejno wyznaczanych cyfr zapisu binarnego.

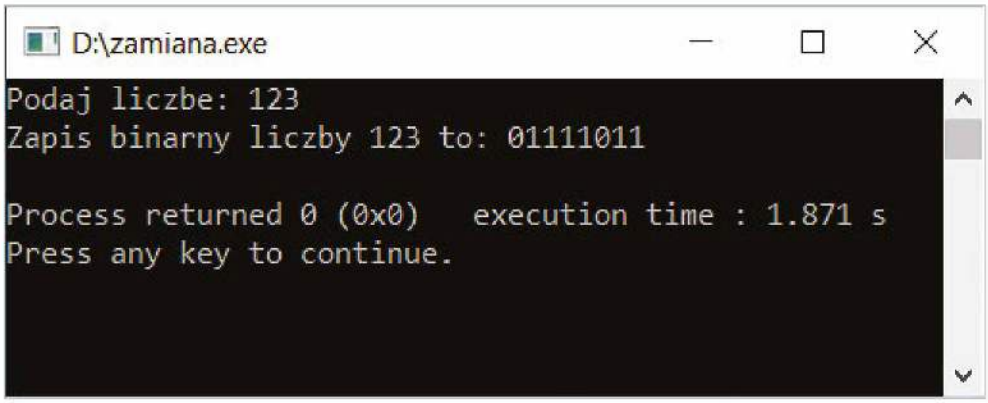
Warto wiedzieć

Zera poprzedzające pierwszą jedynkę w zapisie binarnym, np. pierwsze zero w zapisie 01111011, nazywa się czasem cyframi (zerami) nieznaczącymi.

Zauważ, że program wyświetla zawsze zapis liczby z użyciem ośmiu bitów. Dlatego rozmiar tablicy jest ustalony na 8 (wiersz 7).

Takie rozwiązanie daje lepsze wyobrażenie o tym, jak w pamięci komputera zapisuje się dane liczbowe: współczesne komputery zawsze przechowują i przetwarzają dane, używając liczby bitów o wielkości bajta lub jego wielokrotności.

Przykład wywołania programu *zamiana.exe* przedstawia rysunek 7.11.



Rys. 7.11. Wywołanie programu *zamiana.exe*

Zapamiętaj

W tablicy *n*-elementowej pierwszy element ma indeks 0, a ostatni: $n - 1$.

Ćwiczenie 6

a. Zapisz kod źródłowy programu *zamiana.cpp*, skompiluj go i uruchom program dla różnych liczb (mniejszych niż 256) wpisanych z klawiatury. Zacznij od sprawdzenia liczby 123.

b. Zmień kod źródłowy programu *zamiana.cpp* tak, aby program pozwalał zapisywać liczby za pomocą dwóch bajtów (16 bitów). Metodą prób, uruchamiając program jak najmniejszą liczbę razy, poszukaj największej liczby, którą da się zapisać w ten sposób.

Podsumowanie

- Komputery przechowują oprogramowanie i przetwarzają informacje zapisane cyfrowo w kodzie binarnym. Dotyczy to zarówno liczb, jak i tekstów, obrazów, filmów oraz dźwięków.
- Dane wprowadzane do pamięci komputera albo są w postaci cyfrowej, albo trzeba je poddać konwersji analogowo-cyfrowej.
- Bit jest najmniejszą porcją danych możliwą do zapisania w urządzeniach komputerowych. Jest reprezentowany przez cyfry 0 i 1, które wykorzystuje się do zapisywania liczb w systemie binarnym.
- Grupę ośmiu bitów nazywa się bajtem. Podstawową jednostką pamięci we współczesnych komputerach są bajty i ich wielokrotności.
- Wyznaczenie postaci binarnej liczby zapisanej dziesiętnie jest równoważne wskazaniu potęg liczby 2, które w sumie dają tę liczbę.
- Aby zamienić liczbę dziesiętną na dwójkową, należy kolejno dzielić liczbę dziesiętną, a następnie jej dziesiętne ilorazy częściowe, przez liczbę 2, aż otrzymany iloraz będzie równy 0. Postać binarną liczby stanowią reszty uzyskane z kolejnych dzieleń przez 2 zapisane wspak.
- W języku C++ wynikiem dzielenia jednej liczby całkowitej przez inną liczbę całkowitą jest część całkowita ilorazu.
- Tablica to struktura danych w języku C++ służąca do przechowywania określonej liczby elementów tego samego typu.

Zadania

* **1** Najprostszym systemem liczbowym jest kod jedynkowy (unarny). Poszukaj informacji na temat tego systemu liczbowego. Podaj przykład jego praktycznego zastosowania w życiu codziennym.

* **2** Napisz program, który będzie określać parzystość liczby wprowadzonej z klawiatury. W przypadku liczby parzystej powinien się wyświetlić komunikat „Liczba jest parzysta", natomiast w przypadku liczby nieparzystej – komunikat „Liczba jest nieparzysta".

* **3** Napisz program, który zapisze w tablicy dziesięć kolejnych liczb nieparzystych, począwszy od 1, i wyświetli zawartość tablicy.

* **4** Napisz program, który wczyta do tablicy podane przez użytkownika liczby, a następnie wypisze je w odwrotnej kolejności.

* **5** Napisz program, który dla podanych dwóch liczb dwucyfrowych wypisze ich postać binarną, a następnie postać binarną ich sumy.

* **6** Napisz program, który wczyta 10 liczb podanych przez użytkownika oraz wypisze je oddzielone przecinkami, z kropką po ostatniej liczbie.

** **7** Napisz program, który dla liczby wprowadzonej z klawiatury wyznaczy największą potęgę liczby 2 nie większą od tej liczby.

** **8** W słynnym komputerze ENIAC stosowano kodowanie oparte na systemie dwupiątkowym, znane wcześniej z japońskich liczydeł, tzw. sorobanów. Przygotuj krótką prezentację lub artykuł na stronę internetową szkoły na temat tego systemu liczbowego i jego zastosowania do kodowania liczb w komputerze ENIAC.

** **9** Napisz program przechowujący w tablicy kwadraty 100 kolejnych liczb naturalnych. W tablicy w komórce o indeksie 0 powinna się znajdować liczba 0, w komórce o indeksie 1 – liczba 1, w komórce o indeksie 2 – liczba 4, w komórce o indeksie 3 – liczba 9 itd.

** **10** Napisz program, który dla liczby wprowadzonej z klawiatury wypisze po przecinku wszystkie jej dzielniki jednocyfrowe (łącznie z liczbą 1).

** **11** Napisz program odwracający kolejność cyfr w liczbie całkowitej. Możesz zapamiętywać cyfry w tablicy.

** **12** Napisz program, który będzie określać parzystość liczby wprowadzonej z klawiatury. Program nie powinien korzystać z operatora %.

*** **13** Napisz program tworzący zestawienie liczb binarnych odpowiadających liczbom dziesiętnym od 0 do 255. Efektem działania programu powinny być dwie kolumny: po lewej – liczby dziesiętne, a po prawej – liczby binarne.

*** **14** Napisz program odwracający kolejność cyfr w liczbie całkowitej. Nie możesz zapamiętywać cyfr w tablicy.

*** **15** Poniżej przedstawiono zawartość bitową – w kodzie szesnastkowym – czarno-białej grafiki zapisanej w formacie BMP. Oprócz właściwych danych, które wyróżniono pogrubieniem, plik zawiera w nagłówku tzw. metadane (dane o danych), np. dotyczące rozmiaru obrazu.

```
42 4D 4E 00 00 00 00 00 00 00 3E 00 00 00 28 00 00 00 20 00 00
00 04 00 00 00 01 00 01 00 00 00 00 00 10 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF 00 FF
FF FF FF CC CC CC CC AA AA AA AA FF FF FF FF
```

Odszukaj w wiarygodnym źródle znaczenie poszczególnych bajtów nagłówka. Narysuj na kartce grafikę zakodowaną w tym pliku.

*** **16** Napisz program, który będzie zamieniał liczby binarne na liczby dziesiętne, np. po wprowadzeniu przez użytkownika z klawiatury ciągu cyfr 100011 wypisze odpowiedź: 35. Program powinien używać wyłącznie typu całkowitego. Przyjmij, że użytkownik nie używa w zapisie zer nieznaczących oraz że liczba jest co najwyżej 8-bitowa.

8. Czy ta liczba jest pierwsza?

Liczby pierwsze znasz z lekcji matematyki w szkole podstawowej. Choć odkryto je już w starożytności, przez lata twierdzenia dotyczące liczb pierwszych uchodziły wśród matematyków za piękne, ale bezużyteczne. Dopiero w drugiej połowie XX w. znalazły praktyczne zastosowania w informatyce, zwłaszcza w kryptografii. Twierdzenia te są podstawą działania niektórych protokołów szyfrowania, czyli algorytmów, które zapewniają poufność komunikacji w internecie i bezpieczeństwo transakcji bankowych. W tym temacie dowiesz się więcej na temat liczb pierwszych i ich znaczenia.

Cele lekcji

- Przypomnisz sobie i lepiej zrozumiesz zagadnienie podzielności liczb całkowitych.
- Poznasz algorytmy sprawdzające, czy liczba jest pierwsza.
- Nauczysz się stosować funkcje typu `int` oraz `bool` w języku C++.
- Zastosujesz funkcję `sqrt` z biblioteki funkcji matematycznych `cmath`.

8.1. Liczby złożone i liczby pierwsze

Dodatnie liczby naturalne, które można przedstawić w postaci iloczynu liczb naturalnych mniejszych od nich, nazywamy **liczbami złożonymi**. Ich przykładami są: $9 = 3 \cdot 3$, $10 = 2 \cdot 5$, $120 = 2 \cdot 3 \cdot 4 \cdot 5$, $1115 = 3 \cdot 5 \cdot 7 \cdot 11$.

Liczby złożone

Pozostałe liczby, z wyjątkiem liczby 1, są **liczbami pierwszymi**. Liczbą pierwszą nazywamy więc liczbę, która jest podzielna wyłącznie przez 1 i samą siebie. Wśród liczb mniejszych od 100 jest dwadzieścia pięć liczb pierwszych:
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.
Na rysunku 8.1 oznaczono je zielonym tłem.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	

Liczby pierwsze

Rys. 8.1. Liczby złożone i liczby pierwsze mniejsze niż 100

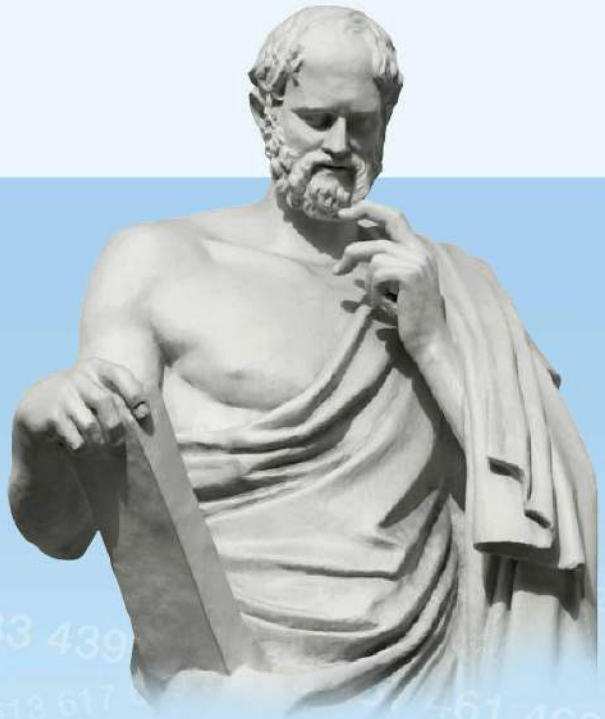
Warto wiedzieć

Liczba 1 jest wyjątkowa. Przez setki lat uważano ją za liczbę pierwszą. Dziś przyjmuje się, że nie jest ani pierwsza, ani złożona.

Jedno z najważniejszych twierdzeń matematycznych dotyczących liczb pierwszych nazywane jest **podstawowym twierdzeniem arytmetyki** i brzmi następująco: każda liczba naturalna (większa od 1) jest albo liczbą pierwszą, albo iloczynem liczb pierwszych.

Podstawowe twierdzenie arytmetyki

Co warto wiedzieć o liczbach pierwszych?



Ile jest liczb pierwszych?

Liczby pierwsze odkryto i badano już w starożytności. W IV w. p.n.e. Euklides udowodnił, że jest ich nieskończenie wiele. Dotąd nie odkryto jednak wzoru, według którego można byłoby dla konkretnej liczby n wyznaczyć n-tą liczbę pierwszą.

2 to najmniejsza i jedyna parzysta liczba pierwsza

Liczby postaci 2^n-1, gdzie n jest liczbą naturalną, nazywamy liczbami Mersenne'a. Wiele z nich jest liczbami pierwszymi

Liczby pierwsze większe od 5 kończą się cyframi 1, 3, 7, 9

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89

Liczby czworacze to cztery liczby pierwsze mające postać: $p, p+2, p+6, p+8$

Liczby bliźniacze to dwie liczby pierwsze, których różnica wynosi 2

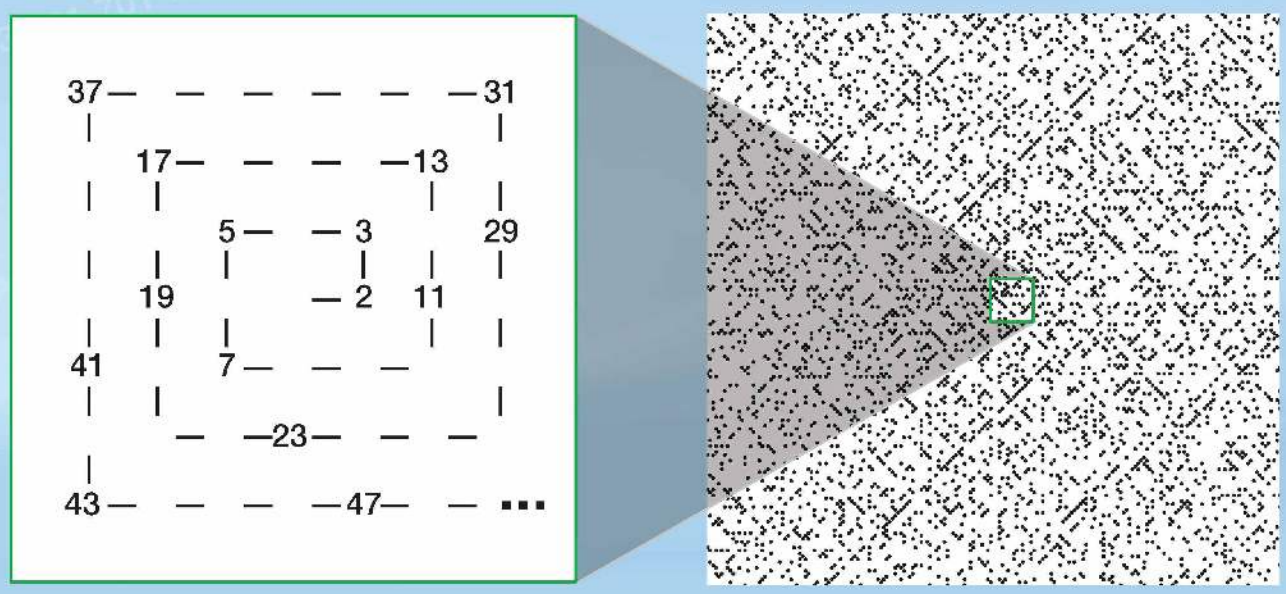
Liczby pierwsze w kryptografii

Liczbę utworzoną z iloczynu dwóch dużych liczb pierwszych bardzo trudno w rozsądnym czasie rozłożyć na czynniki pierwsze, nawet z użyciem najszybszych komputerów. Wykorzystali to twórcy algorytmu RSA, który stosuje się do szyfrowania komunikacji w sieci Internet oraz w bankowości elektronicznej. Liczby pierwsze używane w tym algorytmie mają nawet 2048 bitów, czyli składają się z 2048 cyfr binarnych.



Spirala Ulama

W 1963 r. polski matematyk Stanisław Ulam przedstawił graficznie rozmieszczenie liczb pierwszych. Na kwadratowej tablicy spiralnie wypisał kolejne liczby naturalne. Po naniesieniu bardzo wielu liczb okazało się, że na niektórych liniach liczby pierwsze grupują się częściej niż na innych. Do tej pory nie wyjaśniono, dlaczego tak się dzieje.



Jak liczby pierwsze pomagają rozpoznawać anagramy?

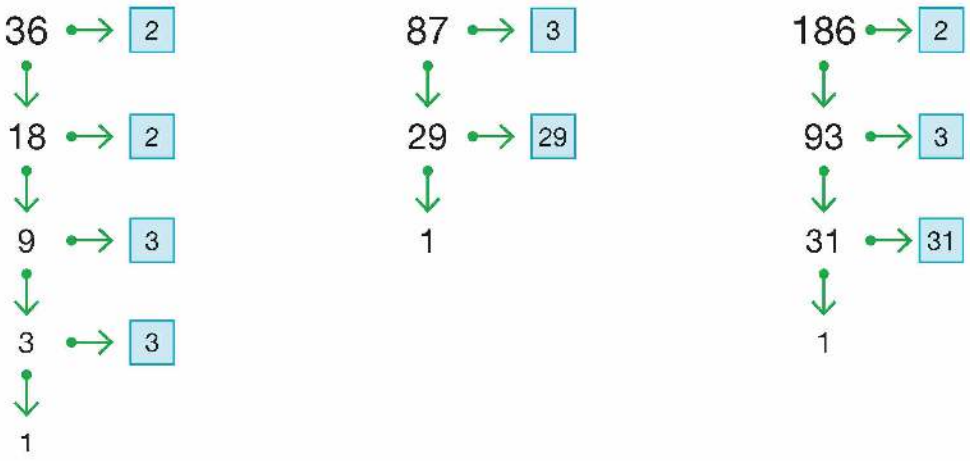
Rozkład dowolnej liczby na czynniki pierwsze jest jednoznaczny (np. dla 30 jest to wyłącznie rozkład $2 \cdot 3 \cdot 5$). Ta własność jest podstawą działania pewnego algorytmu sprawdzającego, czy dwa wyrazy są anagramami (składają się z tych samych liter). Kolejnym literom alfabetu przypisuje się kolejne liczby pierwsze. Jeśli iloczyny liczb odpowiadających literom w wyrazach są równe, wyrazy są anagramami.





Rozkład na czynniki pierwsze

Iloczyn liczb pierwszych, który tworzy daną liczbę, nazywamy **rozkładem na czynniki pierwsze**. Jest on jedyny w swoim rodzaju, tzn. każda liczba ma tylko jedną unikatową formułę rozkładu, która do niej przynależy. Na rysunku 8.2 przedstawiono rozkład na czynniki pierwsze liczb 36, 87 i 186. Liczby te możemy zapisać następująco: $36 = 2 \cdot 2 \cdot 3 \cdot 3$; $87 = 3 \cdot 29$; $186 = 2 \cdot 3 \cdot 31$.



Rys. 8.2. Rozkład liczb 36, 87 i 186 na czynniki pierwsze

Warto wiedzieć

Stosunkowo łatwo sprawdzić, czy podana liczba nieparzysta jest podzielna przez 5, ponieważ kończy się ona cyfrą 5.

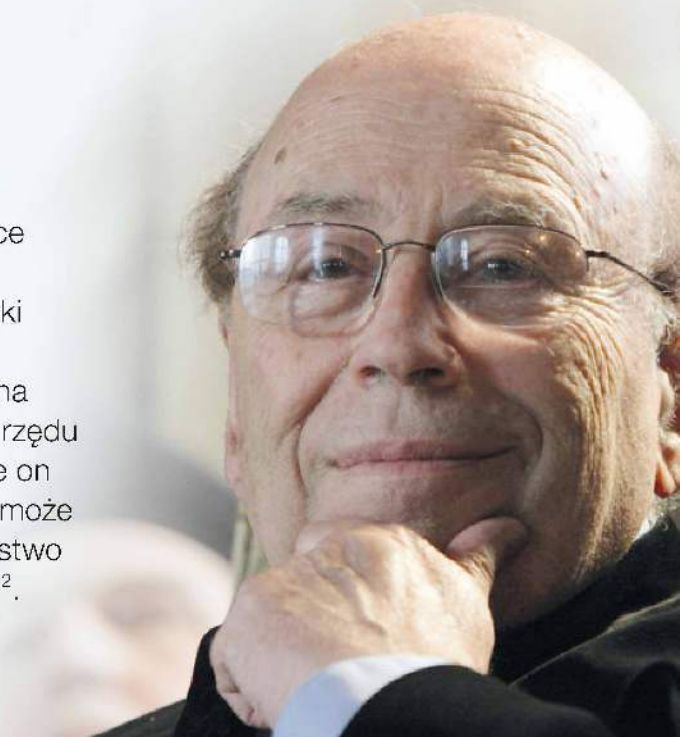
Okazuje się, że im większa liczba, tym trudniej sprawdzić, czy jest pierwsza czy złożona, a więc czy da się ją rozłożyć na czynniki pierwsze. W przypadku dużych liczb rozstrzygnięcie tego w rozsądnym czasie stanowi zadanie praktycznie niewykonalne, nawet dla współczesnych komputerów. Ta trudność jest podstawą bezpieczeństwa działania niektórych algorytmów kryptograficznych (np. algorytmu RSA).

Liczba 2 jest jedyną parzystą liczbą pierwszą, więc w praktyce poszukiwania liczb pierwszych wymagają sprawdzania tylko liczb nieparzystych.

A to ciekawe

Szybki test pierwszości

Dla bardzo dużych liczb bardzo trudno ustalić, czy są one pierwsze czy złożone. Powszechnie stosowanym i w praktyce najszybszym testem pierwszości jest test Millera–Rabina. Algorytm stworzył w 1976 r. amerykański profesor informatyki Gary Miller, a w 1980 r. ulepszył go urodzony we Wrocławiu Michael O. Rabin (na zdjęciu). Dzięki temu algorytmowi można przeprowadzić test pierwszości nawet dla ogromnych liczb, rzędu 150 cyfr. Ponieważ jest to test probabilistyczny, nie wskazuje on żadnych dzielników testowanej liczby. Oznacza to, że liczba może zostać błędnie rozpoznana jako pierwsza. Prawdopodobieństwo takiego zdarzenia jest jednak znikome – wynosi ok. 1 do 10^{12}.



8.2. Podzielność liczb

Pojęcie liczby pierwszej jest ściśle związane z podzielnością liczb. Dlatego zanim stworzymy program sprawdzający, czy podana liczba jest pierwsza, napiszemy prostsze programy związane z podzielnością liczb. Wprowadzimy przy tej okazji nową konstrukcję programistyczną – funkcje.

Funkcje w języku C++

Aby rozwiązać bardziej złożony problem algorytmiczny, najlepiej rozłożyć go na prostsze podproblemy. W praktyce oznacza to wyodrębnianie fragmentów programów (podprogramów), które w języku C++ nazywa się **funkcjami**.

Funkcja

Każda funkcja musi mieć nadaną nazwę. Po nazwie zawsze występują nawiasy okrągłe, w których podaje się **parametry funkcji**.

Parametr funkcji

```
typ nazwa_funkcji(typ parametr1, typ parametr2, ...)
{
    //blok instrukcji tworzących treść funkcji
}
```

Definicja funkcji składa się z nagłówka funkcji i bloku instrukcji. Nagłówek zawiera typ wartości zwracanej przez funkcję, nazwę funkcji oraz listę parametrów, czyli określenie typu i nazwy każdego z nich. Funkcja nie zawiera zwykle instrukcji wyświetlania wyniku na ekranie, ale przesyła efekt swego działania za pomocą **instrukcji `return`** z powrotem do instrukcji programu, która funkcję wywołała.

Instrukcja `return`

Dzięki konsekwentnemu stosowaniu funkcji tworzymy kod bardziej przejrzysty i łatwiejszy do modyfikowania w przyszłości. Unikamy także powtórzeń, ponieważ tę samą funkcję można wywołać wiele razy dla różnych parametrów.

Badanie podzielności przez 3

Aby zobaczyć, jak w praktyce korzysta się z funkcji, napiszemy program, który będzie sprawdzał, czy podana przez użytkownika liczba jest podzielna przez 3. Sprawdzenie podzielności danej liczby przez 3 sprowadza się do ustalenia, czy reszta z dzielenia tej liczby przez 3 jest równa zero. Oto specyfikacja problemu:

Specyfikacja

Dane: dwie liczby naturalne a i b.

Wynik: reszta z dzielenia liczby a przez liczbę b.

Stworzymy funkcję w postaci ogólnej, sprawdzającą resztę z dzielenia liczby a przez liczbę b, a wywołamy ją dla podanej przez użytkownika liczby i b równego 3. Podamy teraz kod źródłowy programu.

Dobra rada

Staraj się tworzyć funkcje o możliwie szerokim zastosowaniu. Dzięki temu będzie można je wywołać dla różnych parametrów. Zwróć uwagę, że tak tworzone są również funkcje w arkuszach kalkulacyjnych.

Kod źródłowy programu sprawdzającego podzielność liczby przez 3

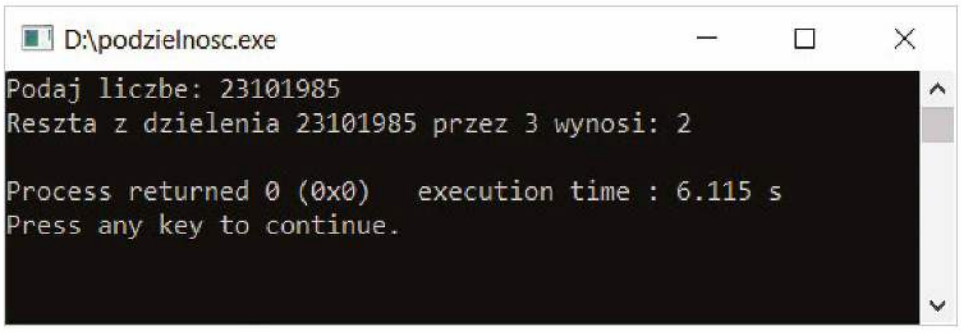
```
1.  #include <iostream>
2.
3.  using namespace std;
4.
5.  int ResztaZDzielenia(int a, int b)
6.  {
7.      return a % b;
8.  }
9.
10. int main()
11. {
12.     int liczba;
13.     cout << "Podaj liczbe: "; cin >> liczba;
14.     cout << "Reszta z dzielenia " << liczba;
15.     cout << " przez 3 wynosi: ";
16.     cout << ResztaZDzielenia(liczba, 3) << endl;
17.
18.     return 0;
19. }
```

Dobra rada

Dłuższe wiersze kodu, zawierające teksty, które chcesz wyświetlić na ekranie, możesz podzielić na mniejsze. Pamiętaj, aby wstawiać odstępy między tekstem a wartościami zmiennych, które mają być wypisane na ekranie.

W linii 16 znajduje się wywołanie funkcji `ResztaZDzielenia` przy użyciu parametrów `liczba` i 3. W ten sposób sprawdzimy podzielność podanej liczby przez 3.

Definicja tej funkcji znajduje się w wierszach 5–8, czyli przed kodem funkcji `main`. W wierszu 5 przed nazwą funkcji umieszczono zapis **`int`**, który określa typ wartości zwracanej przez funkcję. W nawiasach znajduje się lista parametrów funkcji oraz informacja o ich typie. Parametry te przekazują wartości z programu głównego (odpowiednio: wartość zmiennej `liczba` oraz liczbę 3). Wiersz 7 to treść funkcji, która wyznacza resztę z dzielenia (z wykorzystaniem operatora `%`) i zwraca rezultat, stąd słowo kluczowe **`return`**. Na rysunku 8.3 przedstawiono przykładowe wywołanie programu wyznaczającego resztę z dzielenia przez 3.



Rys. 8.3. Wywołanie programu sprawdzającego podzielność przez 3

Parametry formalny i aktualny funkcji

Parametry określone w momencie definiowania nowej funkcji nazywa się **parametrami formalnymi**. Natomiast parametry, dla których funkcja wywołana jest w programie głównym, to **parametry aktualne**.

Ćwiczenie 1

a. Przepisz kod źródłowy programu i sprawdź jego działanie na różnych przykładach.
b. Zmodyfikuj treść programu tak, aby sprawdzał on podzielność podanej liczby przez 17.

Badanie podzielności przez kilka liczb jednocześnie

Zmodyfikujmy kod naszego programu tak, aby wyznaczał reszty z dzielenia przez cztery kolejne liczby pierwsze: 2, 3, 5, 7.

Wynikiem działania programu powinny być cztery liczby: reszta z dzielenia przez 2, reszta z dzielenia przez 3, reszta z dzielenia przez 5 i reszta z dzielenia przez 7. Dzięki temu, że zdefiniowaliśmy wcześniej funkcję `ResztaZDzielenia`, modyfikacja programu polega wyłącznie na dopisaniu do programu głównego odpowiednich wywołań funkcji.

Oto kod źródłowy funkcji głównej programu:

Fragment kodu źródłowego programu obliczającego reszty z dzielenia przez 2, 3, 5, 7

```
10. int main()
11. {
12.     int liczba;
13.     cout << "Podaj liczbe: "; cin>>liczba;
14.
15.     cout << "Reszta z dzielenia liczby " << liczba;
16.     cout << " przez 2, 3, 5, 7: ";
17.     cout << "wynosi odpowiednio:" << endl;
18.     cout << ResztaZDzielenia(liczba, 2) << endl;
19.     cout << ResztaZDzielenia(liczba, 3) << endl;
20.     cout << ResztaZDzielenia(liczba, 5) << endl;
21.     cout << ResztaZDzielenia(liczba, 7) << endl;
22.
23.     return 0;
24. }
```

Ćwiczenie 2

Przepisz kod źródłowy programu i uzupełnij go o instrukcje z linii 1–9. Sprawdź jego działanie na różnych liczbach.

Zapamiętaj

Funkcje definiujemy, gdy chcemy nazwać pewną grupę instrukcji i oddzielić fragment kodu od reszty programu – zwykle po to, aby go wielokrotnie wykorzystać.

8.3. Test pierwszości liczby

Matematycy od wieków zastanawiali się, jak sprawdzić, czy liczba jest pierwsza. Dziś naukowcy szukają przede wszystkim szybkich algorytmów, które będą to rozstrzygać.

Test pierwszości – algorytm najprostszy

Aby stwierdzić, czy dana liczba n jest pierwsza czy złożona, należy sprawdzić, czy ma ona jakikolwiek dzielnik różny od 1 i od niej samej. Można to zrobić, dzieląc n kolejno przez 2, przez 3 itd. aż do $n - 1$.

Przed napisaniem programu komputerowego zapiszemy specyfikację, w której założymy, że liczba podana przez użytkownika mieści się w przedziale obsługiwanym przez typ **int**.

Specyfikacja

Dane: liczba naturalna z przedziału $[2; 2\,147\,483\,647]$.

Wynik: komunikat „Liczba pierwsza", jeśli liczba jest liczbą pierwszą, komunikat „Liczba złożona" – w przeciwnym wypadku.

Poniżej znajduje się kod źródłowy programu *Test pierwszości*.

Kod źródłowy programu *Test pierwszości*

```
1.  #include <iostream>
2.
3.  using namespace std;
4.
5.  bool CzyPierwsza(int n)
6.  {
7.      for (int i=2; i<n; i++)
8.          if (n%i==0) return 0;
9.      return 1;
10. }
11.
12. int main()
13. {
14.     int liczba;
15.     cout <<"Podaj liczbe: "; cin >> liczba;
16.     if (CzyPierwsza(liczba)==1)
17.         cout << "Liczba pierwsza" <<endl;
18.     else
19.         cout << "Liczba zlozona" <<endl;
20.
21.     return 0;
22. }
```

Dobra rada

Ponieważ pętla:

```
for (int i=2;i<n;i++)
{
  if (n%i==0)
    return 0;
}
```

składa się z pojedynczej instrukcji **if**, w której umieszczona jest pojedyncza instrukcja **return**, zapis możesz skrócić następująco:

```
for (int i=2;i<n;i++)
  if (n%i==0) return 0;
```

W liniach 5–10 znajduje się definicja funkcji `CzyPierwsza`, która jest typu `bool`, tzn. wynikiem jej działania będzie wartość typu logicznego. Zadaniem funkcji jest sprawdzenie podzielności podanej liczby przez liczby od 2 do $n - 1$. W przypadku, kiedy zostanie znaleziony dzielnik, wynikiem funkcji (instrukcja `return`) jest wartość logiczna 0 (linia 8), która jest interpretowana w języku C++ jako `false` (fałsz). Wówczas funkcja kończy swoje działanie i nie wykonuje kolejnych dzieleń. Jeśli nie zostanie znaleziony dzielnik, to wynikiem funkcji jest wartość 1, interpretowana – jak każda wartość większa od 0 – jako `true` (prawda). Funkcja kończy wtedy swoje działanie (linia 9).

W programie głównym w linii 16 wywoływana jest funkcja `CzyPierwsza`.

Warto wiedzieć

W latach 90. XX w. pewien producent procesorów był zmuszony wymienić jeden z modeli, ponieważ wykonywał on operacje dzielenia z mniejszą dokładnością, niż zakładano. Wynosiła ona 9 zamiast 19 miejsc po przecinku. Wada ta ujawniła się przy szacowaniu tzw. stałej Bruna, która dotyczny liczb bliźniaczych.

Ćwiczenie 3

Przepisz kod źródłowy programu i sprawdź jego działanie.

Program *Test pierwszości*, choć daje poprawne wyniki, wykonuje wiele niepotrzebnych operacji wyznaczania reszt z dzielenia. Liczbę tych operacji możemy stosunkowo łatwo ograniczyć.

Test pierwszości – ulepszony algorytm

Sprawdzanie podzielności podanej liczby przez liczby parzyste większe od 2 nie ma sensu. Wystarczy więc zbadać podzielność przez 2, a potem sprawdzać wyłącznie liczby nieparzyste.

Liczbę wykonywanych dzieleń można ograniczyć jeszcze bardziej. Oto rozkłady liczb 36, 100 i 42 na czynniki:

$36 = 2 \cdot 18$	$100 = 2 \cdot 50$	$42 = 2 \cdot 21$
$36 = 3 \cdot 12$	$100 = 4 \cdot 25$	$42 = 3 \cdot 14$
$36 = 4 \cdot 9$	$100 = 5 \cdot 20$	$42 = 6 \cdot 7$
$36 = 6 \cdot 6$	$100 = 10 \cdot 10$	

Przykłady te pokazują, że gdy liczbę złożoną n przedstawiamy w postaci iloczynu dwóch czynników większych od 1, to przynajmniej jeden z tych czynników musi być mniejszy lub równy $\sqrt{n}$. Dla liczby 36 są to odpowiednio liczby 2, 3, 4 i 6, przy czym $6 = \sqrt{36}$. Podobnie dla pozostałych liczb – mniejszy z dzielników liczby nie jest większy od pierwiastka z tej liczby. Tak więc najmniejszy dzielnik właściwy liczby złożonej n musi być mniejszy bądź równy $\sqrt{n}$.

Warto wiedzieć

Dzielnik właściwy liczby to każdy dzielnik mniejszy od tej liczby.

Ćwiczenie 4

Sprawdź za pomocą kalkulatora, czy 401 jest liczbą pierwszą. Ile dzieleń było koniecznych, aby to rozstrzygnąć? Zastosuj wskazówkę zapisaną powyżej.

Operacja pierwiastkowania

Operacja pierwiastkowania nie jest standardowym działaniem matematycznym w języku C++, dlatego w kodzie musimy dołączyć

Biblioteka cmath

bibliotekę o nazwie cmath. Zawiera ona zdefiniowane funkcje matematyczne, m.in. trygonometryczne, wykładnicze, logarytmiczne.

Poniżej znajduje się tekst programu *Ulepszony test pierwszości*.

Kod źródłowy programu *Ulepszony test pierwszości*

```
1.  #include <iostream>
2.  #include <cmath>
3.  using namespace std;
4.
5.  bool CzyPierwsza(int n)
6.  {
7.      if (n==2) return 1;
8.      if (n%2==0) return 0;
9.
10.     int pierwiastek = sqrt(n);
11.     for (int i=3; i<=pierwiastek; i=i+2)
12.         if (n%i==0) return 0;
13.     return 1;
14. }
15. int main()
16. {
17.     int liczba;
18.     cout << "Podaj liczbe: "; cin >> liczba;
19.
20.     if (CzyPierwsza(liczba)==1)
21.         cout << "Liczba pierwsza" << endl;
22.     else
23.         cout << "Liczba zlozona" << endl;
24.
25.     return 0;
26. }
```

Ulepszenia wprowadzono w definicji funkcji `CzyPierwsza`. W liniach 7–8 nadajemy status liczby pierwszej liczbie 2 (instrukcja `return 1;`) oraz status liczby złożonej wszystkim liczbom podzielnym przez 2 większym od 2 (instrukcja `return 0;`). W linii 10 wprowadzono zmienną `pierwiastek`, której wartość stanowi przybliżenie $\sqrt{n}$

Funkcja sqrt

otrzymane dzięki **funkcji** `sqrt`, znajdującej się w bibliotece cmath (linia 2). W pętli modyfikujemy liczbę wykonań – ograniczamy ją do $\sqrt{n}$ oraz testujemy tylko liczby nieparzyste (instrukcja `i=i+2`).

Ćwiczenie 5

Przepisz kod źródłowy programu i sprawdź jego działanie.

Podsumowanie

- Liczbą złożoną nazywamy liczbę większą od 1, która ma przynajmniej jeden dzielnik różny od niej samej i różny od 1.
- Liczba pierwsza to liczba większa od 1, która dzieli się wyłącznie przez samą siebie oraz przez 1.
- Efektywność algorytmu można poprawić przez ograniczenie zbędnych działań (operacji matematycznych). Na przykład przy badaniu pierwszości liczby wystarczy sprawdzać podzielność przez 2 oraz przez liczby nieparzyste nie większe niż pierwiastek kwadratowy z danej liczby.
- Korzystanie z funkcji poprawia czytelność programu i umożliwia wielokrotne wykorzystywanie tego samego kodu.
- Typ logiczny w języku C++ to `bool`. Przyjmuje on dwie wartości: `true` oraz `false`. Typ ten jest używany w nagłówkach funkcji zwracających wartość logiczną, których celem jest rozstrzygnięcie, czy dana równość jest prawdziwa.
- W języku C++ można korzystać z biblioteki matematycznej `cmath`, która zawiera wiele wbudowanych funkcji matematycznych, ułatwiających wykonanie w programie bardziej złożonych obliczeń.

Zadania

* **1** Napisz program wyznaczający wszystkie dzielniki liczby 80.

* **2** Napisz program znajdujący najmniejszą i największą trzycyfrową liczbę pierwszą.

* **3** Napisz program, który wyświetli największą liczbę pierwszą mniejszą od dodatniej trzycyfrowej liczby całkowitej podanej przez użytkownika.

* **4** Napisz funkcję, która będzie badała podzielność podanej liczby jednocześnie przez 3 i 7. W teście użyj pojedynczej instrukcji logicznej (bez operatora &&).

* **5** Napisz program, który dla dwóch podanych przez użytkownika liczb wyświetli informację o tym, czy ich suma jest liczbą pierwszą. Utwórz w tym celu funkcję typu `bool`.

* **6** Napisz program, który dla danej liczby rzeczywistej a wyświetli wartości: $\cos(a)$ oraz $\sin(a)$. Skorzystaj z gotowych funkcji `sin` oraz `cos` z biblioteki `cmath`.

* **7** Napisz program wyświetlający liczby, których pierwiastki mieszczą się w podanym przez użytkownika przedziale. Na przykład dla przedziału [5; 6] są to liczby: 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36. W kodzie wykorzystaj bibliotekę `cmath` i funkcję `sqrt`. Sprawdź działanie programu dla przedziału [14; 19].

** **8** Stwórz program, który wyznaczy wszystkie trzycyfrowe liczby pierwsze.

** 9 Przygotuj w arkuszu kalkulacyjnym skoroszyt, który pozwoli rozstrzygnąć, czy prawdziwe jest następujące twierdzenie: każda liczba złożona mniejsza od 100 dzieli się przynajmniej przez jedną z liczb: 2, 3, 5, 7. Skorzystaj z funkcji MOD.

** 10 Napisz program, który po podaniu przez użytkownika liczby typu `float` wyświetli zaokrąglenia tej liczby w górę i w dół do najbliższej liczby całkowitej. Skorzystaj z funkcji `ceil` i `floor` z biblioteki `cmath`.
Uwaga: Zaokrąglenie do najbliższej liczby całkowitej w górę nazywa się sufitem, a w dół – podłogą. Stąd nazwy funkcji pochodzące od angielskich słów *ceiling* i *floor*.

** 11 Napisz program wyznaczający wszystkie dzielniki pierwsze liczby 80.

** 12 Napisz w języku C++ prostą grę, w której dwóch użytkowników podaje po jednej liczbie 6-cyfrowej. Wygrywa ten, kto wskazał liczbę, która ma więcej dzielników właściwych.

** 13 Znajdź pary dwucyfrowych i trzycyfrowych liczb bliźniaczych.
Uwaga: Liczby bliźniacze to dwie liczby pierwsze, których różnica wynosi 2.

** 14 Liczba 362 881 ma tę własność, że przy dzieleniu przez 2, 3, 4, 5, 6, 7, 8, 9 daje resztę 1. Znajdź najmniejszą liczbę o tej własności.

*** 15 Napisz program, który wyznaczy wszystkie trzycyfrowe liczby doskonałe. Liczba doskonała to liczba równa sumie swych dzielników różnych od niej samej, np. 28 = 1 + 2 + 4 + 7 + 14.

*** 16 Wśród poniższych liczb są iloczyny dwóch liczb pierwszych nieparzystych. Napisz program zawierający funkcję `sprawdz`, która pozwoli znaleźć te liczby.
1 838 947 883
1 004 098 109
851 260 967
809 509 261
669 266 311
Wskazówka: Funkcja `sprawdz` powinna zwracać wartość znalezionego dzielnika liczby, o ile liczba nie jest pierwsza. Wartość tę można zapamiętać. Wówczas pozostaje sprawdzić jedną liczbę. Jaką?

*** 17 Znajdź jak największą liczbę, która jest iloczynem dwóch liczb pierwszych.
Uwaga: Największa liczba, którą możesz uzyskać, używając typu `int`, to prawdopodobnie 2 147 483 647. Zależy to od używanego kompilatora.

*** 18 Napisz program, który wypisze wszystkie pary liczby bliźniaczych z przedziału [2; 3000].
Uwaga: Liczby bliźniacze to dwie liczby pierwsze, których różnica wynosi 2.

*** 19 Napisz program, który wyświetli listę wszystkich par liczb zaprzyjaźnionych mniejszych niż milion.
Uwaga: Dwie liczby uważa się za zaprzyjaźnione, jeśli sumy dzielników właściwych tych liczb są równe.

9. Algorytm Euklidesa i działania na ułamkach

Jednym z najstarszych algorytmów obliczeniowych jest procedura wyznaczania największego wspólnego dzielnika dwóch liczb, opisana przez greckiego matematyka Euklidesa. Algorytm ten służył najpierw do szukania odcinka, który byłby wspólną miarą dla dwóch odcinków. Dzisiaj znalazł zastosowanie m.in. w kryptograficznym algorytmie RSA, wykorzystywanym np. w podpisach cyfrowych. W tym temacie użyjemy algorytmu Euklidesa do wykonywania operacji na ułamkach.

Cele lekcji

- Przypomnisz sobie pojęcia: największy wspólny dzielnik, najmniejsza wspólna wielokrotność oraz algorytm Euklidesa.
- Poznasz przykład zastosowania algorytmu Euklidesa do wykonywania działań na ułamkach zwykłych.
- Nauczysz się stosować funkcje typu `void` w języku C++.

9.1. Patchworkowe kołdry

Kołdra patchworkowa powstaje ze zszycia wielu mniejszych kawałków tkanin. Wyobraź sobie, że przygotowujesz projekt małej kołdry o wymiarach 112 × 88 cm, która ma być uszyta z jak największych kwadratowych kawałków równej wielkości. Jakiej długości bok powinien mieć taki kawałek?

Warto wiedzieć

Jeszcze na początku XX w. w krajach anglosaskich patchworkowe kołdry były elementem posagu, który panna młoda wnosiła do domu męża.



Rys. 9.1. Patchworkowa kołdra uszyta z kwadratów

Zauważ, że długość boku takiego kwadratu musi być największą możliwą liczbą, która dzieli liczby wyrażające długość i szerokość całej kołdry. Z matematycznego punktu widzenia liczba określająca długość boku największego możliwego kwadratu to największy wspólny dzielnik liczb 112 i 88. Dla kołdry o wymiarach 112 × 88 cm długość boku kawałka materiału powinna wynosić 8 cm, ponieważ $88 = 8 \cdot 11$ oraz $112 = 8 \cdot 2 \cdot 7$.

Ćwiczenie 1

Wyjaśnij, dlaczego podane na s. 153 rozkłady liczb 112 i 88 na czynniki jednoznacznie wskazują, że największy kwadratowy kawałek materiału ma bok długości 8 cm.

9.2. Największy wspólny dzielnik

Największy wspólny dzielnik (NWD)

Przypomnijmy, że **największy wspólny dzielnik** (w skrócie: **NWD**) dwóch liczb całkowitych dodatnich a i b to największa liczba całkowita, która jest dzielnikiem obu tych liczb. Jeśli a jest dzielnikiem b, to liczba a jest największym wspólnym dzielnikiem liczb a i b.

W większości przypadków nie jest łatwo natychmiast wskazać największy wspólny dzielnik konkretnych dwóch liczb. Dla par liczb dwu- albo trzycyfrowych można rozłożyć na czynniki każdą z liczb i wyznaczyć iloczyn wspólnych czynników obu rozkładów. Tę metodę zastosowaliśmy w przykładzie z kołdrą. Jednak już dla liczb większych niż 1000 byłoby to bardzo czasochłonne.

Warto wiedzieć

Rozkład liczb o setkach cyfr na czynniki pierwsze jest problemem trudnym obliczeniowo również dla współczesnych komputerów. Nie istnieje efektywny algorytm, który pozwalałby na znalezienie takiego rozkładu w rozsądnym czasie.

Algorytm naiwny wyznaczania NWD

Zastanówmy się, czy można znaleźć NWD bez potrzeby rozkładania liczb na czynniki pierwsze.

Zapiszmy specyfikację problemu nieco ogólniej:

Specyfikacja

Dane: liczby całkowite dodatnie a i b.

Wynik: największy wspólny dzielnik liczb a i b.

Jedno z rozwiązań może być następujące: będziemy szukać wspólnego dzielnika liczb a i b wśród coraz mniejszych liczb, zaczynając od dzielnika o wartości równej mniejszej z tych liczb. Poszukiwanie NWD zakończy się najpóźniej w momencie, gdy wspólny dzielnik będzie równy 1.

Warto wiedzieć

O dwóch liczbach, które nie mają wspólnego dzielnika większego od 1, mówi się, że są względnie pierwsze.

Algorytm, na podstawie którego stworzymy program wyznaczający NWD, można zapisać w postaci następującej listy kroków:

1. Jeśli a > b, to zamień wartości a i b miejscami.
2. Dla wartości d równych kolejno a, a - 1, a - 2, …, 1 powtarzaj kroki 3, 4 i 5.
   3. Wyznacz resztę z dzielenia a przez d.
   4. Wyznacz resztę z dzielenia b przez d.
   5. Jeśli obie reszty wyznaczone w krokach 3 i 4 są równe 0, to zwróć wartość d i zakończ algorytm.

Poniżej znajduje się tekst programu komputerowego, który dla dwóch liczb całkowitych wczytanych z klawiatury wyznacza NWD tą metodą.

Kod źródłowy programu wyznaczającego NWD dwóch liczb

```
1.  #include<iostream>
2.
3.  using namespace std;
4.
5.  int NWD(int a, int b)
6.  {
7.      int p;
8.      if (a>b)
9.          swap(a,b);
10.
11.     for(int d=a; d>=1; d--)
12.         if (a%d==0 && b%d==0) return d;
13. }
14.
15. int main()
16. {
17.     int a, b;
18.     cout << "Podaj liczbe a: "; cin>>a;
19.     cout << "Podaj liczbe b: "; cin>>b;
20.     cout << "\nNWD(" << a << ", " << b << ")= ";
21.     cout << NWD(a,b) << endl;
22.
23.     return 0;
24. }
```

Dobra rada

Pamiętaj, że kompilator języka C++ rozróżnia duże i małe litery. Jeśli deklarujesz zmienną lub funkcję zawierającą w nazwie duże litery, to w dalszej części kodu musisz konsekwentnie używać ustalonej nazwy.

W wierszach 5–13 zdefiniowana jest funkcja NWD. W wierszach 8–9 znajduje się kod służący temu, aby zawsze w zmiennej b zapisana była większa z wartości, a w zmiennej a – mniejsza. Jeśli tak nie jest, wartości zmiennych są **zamieniane**.

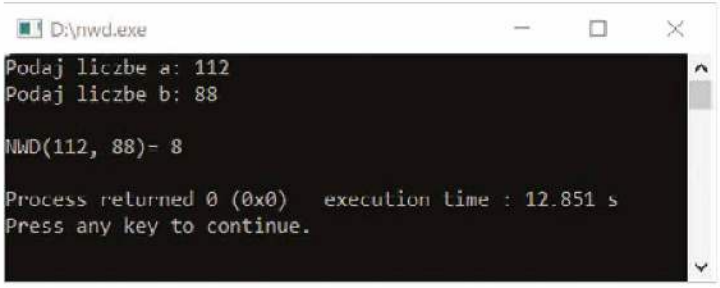
Zamiana wartości zmiennych, s. 109

Zamianę wartości zmiennych realizujemy z wykorzystaniem wbudowanej w język C++ **funkcji swap** – jej argumentami są zmienne, których wartości należy zamienić (linia 9).

Funkcja swap

Kod realizujący kluczową ideę algorytmu znajduje się w wierszach 11 i 12. W zmiennej d będziemy przechowywać aktualną wartość potencjalnego wspólnego dzielnika. Pętla **for** zakończy działanie w momencie znalezienia wspólnego dzielnika a i b (zapisano to w warunku logicznym instrukcji **if**). Wówczas do programu głównego (funkcji main) zwracana jest wartość d – odpowiada za to instrukcja **return**.

Ponieważ wartości zmiennej d są przy każdej kolejnej iteracji zmniejszane o jeden, znaleziony dzielnik na pewno jest największym wspólnym dzielnikiem liczb a i b. Rysunek 9.2 na s. 156 przedstawia przykład wywołania programu.



Rys. 9.2. Wywołanie programu *nwd.exe*

Warto wiedzieć

Jedną z cech dobrego języka programowania jest domyślne ograniczanie widoczności zmiennej do fragmentu programu. To programista powinien świadomie decydować, w którym fragmencie kodu dostęp do zmiennej jest naprawdę potrzebny.

Zauważ, że zmienne `a` i `b` zdefiniowano w programie dwukrotnie: raz dla funkcji `main`, a raz dla funkcji `NWD` w jej nagłówku. Jest to możliwe, ponieważ zmienne zadeklarowane w funkcji `main` nie są dostępne (widoczne) dla kompilatora podczas przetwarzania linii kodu z funkcji `NWD`. Zmienne `a` i `b` zdefiniowane w funkcji `NWD` są zapisane w innych komórkach pamięci komputera.

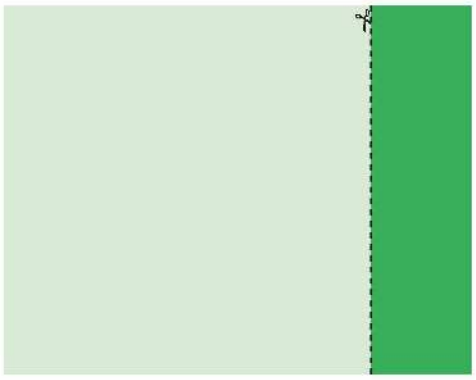
Zwróć uwagę na linie 20 i 21 – przeplatają się w nich napisy stałe oraz wartości zmiennych. Pozwala to wyświetlić wynik tak jak na rysunku 9.2.

Ćwiczenie 2

Zapisz kod źródłowy programu *nwd.cpp*, skompiluj go i uruchom program dla różnych par liczb wpisanych z klawiatury.

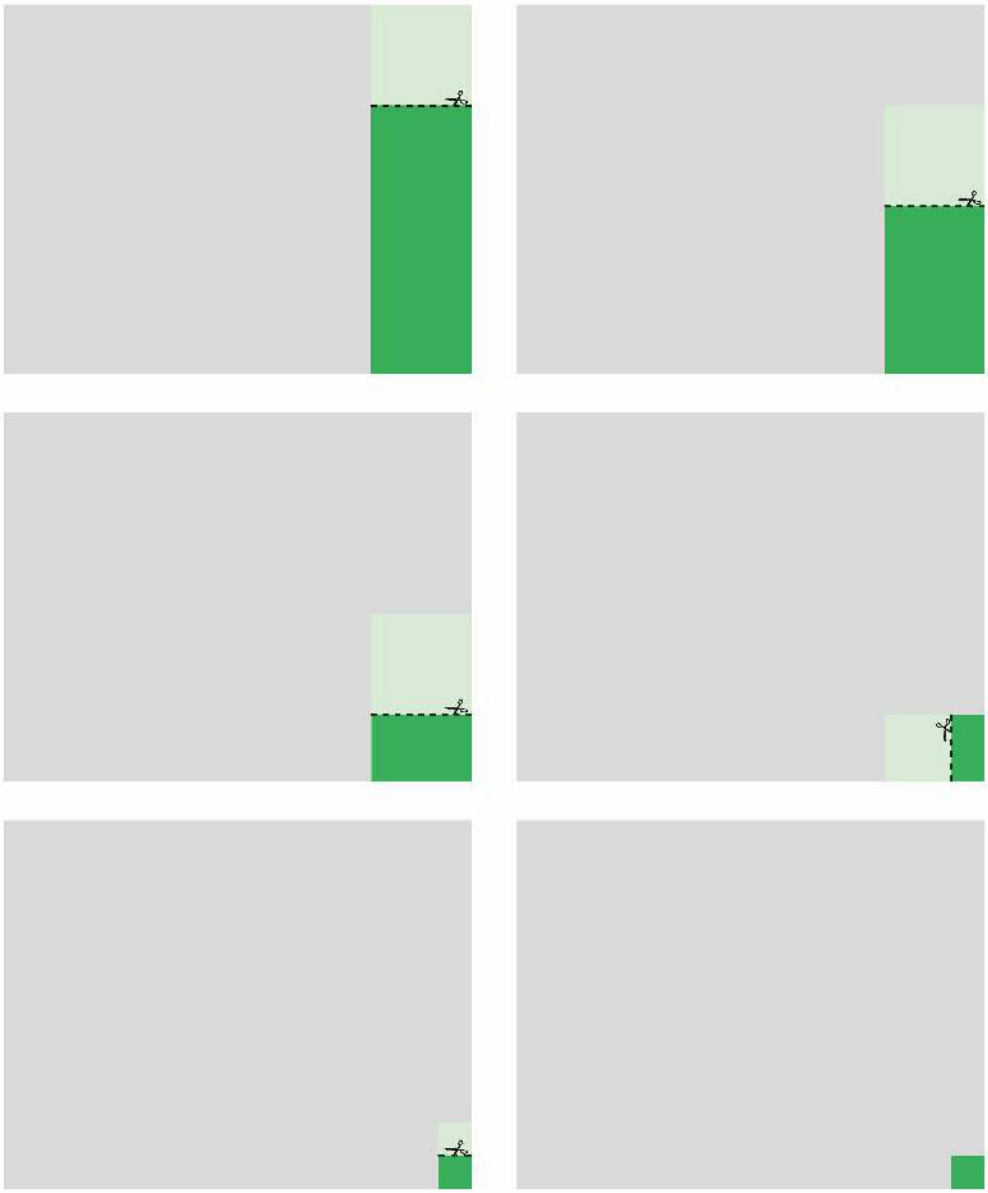
Wycinanka Euklidesa

Wróćmy do przykładu wyznaczania długości boku największego kwadratowego kawałka kołdry o wymiarach 112 × 88 cm. Jak można by możliwie szybko wyznaczyć wymiary takiego kwadratu bez rozkładania liczb 112 i 88 na czynniki pierwsze?



Rys. 9.3. Prostokąt o wymiarach 112 na 88 z odciętym kwadratem o boku 88

Rysunek 9.3 przedstawia pierwszy krok procedury, którą nazwiemy wycinanką Euklidesa. Krok po kroku będziemy odcinać jak największy kwadrat od prostokąta, który powstał w poprzednim kroku. Po pewnej liczbie kroków otrzymamy prostokąt, który jest kwadratem. Kolejne kroki pokazano na rysunku 9.4 i opisano w tabeli 9.1.



Warto wiedzieć

Odcinanie jak największego kwadratu od danego prostokąta to przykład zastosowania strategii algorytmicznej nazywanej podejściem zachłannym.

Rys. 9.4. Kolejne kroki wycinanki Euklidesa

Wymiary prostokąta	Długość boku kwadratu
112 × 88	88
88 × 24	24
64 × 24	24
40 × 24	24
24 × 16	16
16 × 8	8
8 × 8	koniec

Tabela 9.1. Wymiary prostokątów i boków kwadratów w kolejnych krokach wycinanki

Ćwiczenie 3

Zastosuj procedurę wycinania w prostokącie o bokach 51 i 30. Użyj kartki w kratkę i nożyczek.

Dzięki wycinance wiemy, że kwadratami o boku 8 można wypełnić cały prostokąt o wymiarach 112 × 88. Czy można by go wypełnić kwadratami o większym boku? Istnieje dowód matematyczny na to, że nie znajdziemy takiego kwadratu. Dowód ten opiera się na następującym spostrzeżeniu: jeśli prostokąt o bokach a i b (gdzie $a < b$) da się wypełnić kwadratami o boku d, to prostokąt o bokach a i $b - a$ także da się wypełnić kwadratami o boku d.

To oznacza, że każdy z prostokątów odcinanych w wycinance Euklidesa dałby się wypełnić kwadratami o boku większym niż 8, gdyby istniał taki kwadrat dla prostokąta o wymiarach 112 × 88. Tymczasem prostokąta o wymiarach 16 × 8, uzyskanego w przedostatnim kroku procedury, nie da się wypełnić kwadratami o boku dłuższym niż 8.

Spostrzeżenie zapisane powyżej można przedstawić w postaci równości: $NWD(a, b) = NWD(a, b - a)$ dla $a < b$. Ta równość jest podstawą algorytmu Euklidesa.

Warto wiedzieć

Jeśli pewna liczba dzieli bez reszty dwie liczby, to dzieli bez reszty również ich różnicę. Szkic dowodu można napisać tak: Niech $a < b$ oraz $a = n \cdot d$ i $b = m \cdot d$, gdzie n i m to liczby całkowite dodatnie. Wówczas:
$b - a = m \cdot d - n \cdot d =$
$= (m - n) \cdot d$,
czyli d dzieli różnicę $b - a$.

Algorytm Euklidesa

Algorytm naiwny ze s. 154 zastosowany dla pary liczb 112 i 88 sprawdzi każdą z liczb: 88, 87, 86, ..., 9, 8, a więc wykona ponad 80 dzieleń. Dla komputera to niedużo – efekt wyświetli się niemal natychmiast. Jednak jeśli będziemy sprawdzać parę liczb wielocyfrowych, np. 1 000 000 014 oraz 999 999 999, wynik na ekranie komputera nie pokaże się od razu, gdyż miliard operacji dzielenia jest już poważnym obciążeniem dla komputera i wymaga dłuższego czasu pracy procesora.

Warto wiedzieć

Największą liczbą całkowitą typu `int`, którą program skompilowany w C++ poprawnie rozpozna, jest liczba 2 147 483 647. Wynika to z ograniczeń reprezentacji liczb w komputerze: zwykle kompilatory dla zapisu zmiennych typu `int` stosują 32 bity pamięci (od liczby -2^{31} do $2^{31} - 1$).

Ćwiczenie 4

Uruchom program z ćwiczenia 2 i porównaj czas jego działania dla dwóch par liczb: 112 i 88 oraz 1 000 000 014 i 999 999 999. Co zauważasz? Jak można to wytłumaczyć?

Skorzystamy z obserwacji wynikających z wycinanki i stworzymy program realizujący algorytm opisany w starożytności przez Euklidesa w jego słynnym dziele *Elementy*.

Algorytm Euklidesa

Algorytm Euklidesa można zapisać w postaci następującej listy kroków:

1. Dopóki a ≠ b, powtarzaj kroki 2 i 3.
   2. Jeśli a > b, to a = a - b.
   3. W przeciwnym razie b = b - a.
4. Zwróć wartość a.

Poniżej znajduje się kod funkcji NWD, który jest realizacją algorytmu Euklidesa.

```
5.  int NWD(int a, int b)
6.  {
7.      while (a!=b)
8.          if (a>b) a=a-b;
9.          else b=b-a;
10.     return a;
11. }
```

Kod źródłowy funkcji realizującej algorytm Euklidesa

Po zakończeniu działania pętli **while** wartości obu zmiennych a i b są równe największemu wspólnemu dzielnikowi początkowych wartości a i b. Dlatego funkcja NWD zwraca jedną z tych wartości (linia 10). W przypadku liczb, które nie mają wspólnego dzielnika większego niż 1, funkcja zwraca wartość 1.

Czy dla dowolnej pary liczb *a* i *b* w skończonej liczbie kroków zostanie osiągnięty warunek wyjścia z pętli (czyli *a* będzie równe *b*)? Od czasów Euklidesa wiemy, że stanie się tak wtedy i tylko wtedy, gdy *a* i *b* będą liczbami współmiernymi.

Warto wiedzieć

Liczby współmierne to takie, dla których można wyznaczyć wspólną miarę, czyli liczbę dzielącą obie liczby bez reszty.

Ćwiczenie 5

a. Zmodyfikuj kod źródłowy programu *nwd.cpp*, używając zapisanej wyżej funkcji NWD, i zapisz go pod nazwą *euklides.cpp*. Skompiluj kod i uruchom program dla różnych par liczb wpisanych z klawiatury.

b. Porównaj czas działania programu dla dwóch par liczb: 112 i 88 oraz 1 000 000 014 i 999 999 999. Czy zauważasz różnicę w działaniu algorytmu naiwnego i algorytmu Euklidesa?

A to ciekawe

Matematyczny bestseller wszech czasów

Elementy Euklidesa to traktat matematyczny, który opisuje osiągnięcia starożytnej geometrii. Jest on uznawany za dzieło wyjątkowe, ponieważ zastosowano w nim podejście dedukcyjne (od ogółu do szczegółu), precyzję definicji i dowodów twierdzeń. W traktacie znalazł się również algorytm, który jest wykorzystywany we współczesnej informatyce. Mimo że dzieło Euklidesa powstało około 300 r. p.n.e., używano go w kolejnych przekładach jako podręcznika do geometrii przez niemal 2000 lat. Uznaje się, że *Elementy* są drugą po Biblii książką najczęściej drukowaną i tłumaczoną w historii świata.



Warto wiedzieć

Operacja dzielenia, a więc i modulo, pochłania stosunkowo dużo czasu pracy procesora. Dlatego w najszybszych implementacjach algorytmu Euklidesa dzielenie z resztą zastępuje się odejmowaniem oraz połowieniem (przesunięciami bitów o jeden w prawo), które procesory wykonują szybko. W ten sposób NWD można wyznaczyć ok. 60% szybciej.

Algorytm Euklidesa – wersja z dzieleniem

Zauważ, że jeśli w algorytmie Euklidesa jedna liczba jest dużo większa od drugiej, to wielokrotnie powtarza się odejmowanie od niej tej samej liczby. W tabeli 9.2 przedstawiono znajdowanie NWD liczb 196 i 42.

Wartość zmiennej a	Wartość zmiennej b
196	42
196 – 42 = 154	42
154 – 42 = 112	42
112 – 42 = 70	42
70 – 42 = 28	42
28	42 – 28 = 14
28 – 14 = 14	14

Tabela 9.2. Wyznaczanie NWD zgodnie z algorytmem Euklidesa

Wielokrotne odejmowanie możemy zastąpić wyznaczeniem reszty z dzielenia większej liczby przez mniejszą – tak jak w tabeli 9.3.

Wartość zmiennej a	Wartość zmiennej b
196	42
196 % 42 = 28	42
28	42 % 28 = 14
28 % 14 = 0	14

Tabela 9.3. Wyznaczanie NWD zgodnie z algorytmem Euklidesa, w którym kolejne odejmowania zastąpiono resztą z dzielenia

Wynikiem dzielenia 196 przez 42 jest 4 i reszta 28. Część całkowita określa liczbę operacji odejmowania, a reszta z dzielenia – wynik ostatniego odejmowania. W tym przypadku cztery działania zastąpiliśmy jednym.

Zwróć uwagę na ostatni wiersz tabeli 9.3. Inny jest tym razem warunek zakończenia działania algorytmu. Nie otrzymujemy dwóch takich samych wartości – jedna ze zmiennych przyjmuje wartość 0. Przed ostatnim dzieleniem większa liczba jest wielokrotnością mniejszej, a więc ta mniejsza jest największym wspólnym dzielnikiem.

Algorytm Euklidesa – wersja z dzieleniem

Oto lista kroków **algorytmu Euklidesa w wersji z dzieleniem**:

1. Dopóki a ≠ 0 i b ≠ 0, powtarzaj kroki 2 i 3.
   2. Jeśli a > b, to a = a % b.
   3. W przeciwnym razie b = b % a.
4. Jeśli a ≠ 0, to zwróć a.
5. W przeciwnym razie zwróć b.

Poniżej znajduje się kod funkcji NWD, który jest realizacją algorytmu Euklidesa w wersji z dzieleniem.

Kod źródłowy funkcji realizującej algorytm Euklidesa w wersji z dzieleniem

```
5.  int NWD(int a, int b)
6.  {
7.      while (a!=0 && b!=0)
8.          if (a>b) a=a%b;
9.          else b=b%a;
10.
11.     if (a!=0) return a;
12.     else return b;
13. }
```

Ćwiczenie 6

a. Zmodyfikuj kod programu *euklides.cpp* z ćwiczenia 5, używając zapisanej wyżej funkcji NWD, i zapisz go pod nazwą *euklides_dzielenie.cpp*. Skompiluj kod i uruchom program dla różnych par liczb wpisanych z klawiatury.
b. Dodaj do kodu instrukcję zliczającą wykonane iteracje.
c. Dołącz do kodu funkcję NWD (w wersji z odejmowaniem). Zmodyfikuj kod programu tak, aby porównywał liczbę iteracji wykonanych w obu wersjach.

Zapamiętaj

Największy wspólny dzielnik dwóch liczb naturalnych można wyznaczyć bez konieczności szukania wspólnych dzielników tych liczb. Należy zastosować algorytm Euklidesa.

9.3. Działania na ułamkach zwykłych

W szkole na lekcjach różnych przedmiotów wielokrotnie wykonywaliście podstawowe działania arytmetyczne na ułamkach, m.in. dodawanie i mnożenie. Napiszemy teraz programy, które pozwolą wykonywać obliczenia dla ułamków o wartościach dodatnich.

Licznik i mianownik ułamka będziemy reprezentować w postaci liczb całkowitych, zadeklarowanych w języku C++ jako dwie zmienne typu **int**.

Dodawanie ułamków

Żeby dodać dwa ułamki zwykłe, wystarczy pomnożyć ich mianowniki i dodać liczniki mnożone odpowiednio przez mianowniki:

$$\frac{a}{b}+\frac{c}{d}=\frac{a\cdot d+c\cdot b}{b\cdot d}=\frac{e}{f}$$

Aby doprowadzić ułamek do postaci nieskracalnej, wystarczy podzielić liczby e i f przez NWD(e, f).

Oto specyfikacja problemu dodawania dwóch ułamków:

Specyfikacja

Dane: cztery liczby całkowite – licznik i mianownik jednego ułamka oraz licznik i mianownik drugiego ułamka.

Wynik: dwie liczby całkowite – licznik i mianownik sumy ułamków, zapisane w postaci nieskracalnej.

Poniżej znajduje się kod programu *Suma ułamków*, który realizuje powyższą specyfikację.

Kod źródłowy programu ● *Suma ułamków*

```
1.  #include<iostream>
2.
3.  using namespace std;
4.
5.  int NWD(int a, int b)
6.  {
7.      while (a!=0 && b!=0)
8.      if (a>b) a=a%b;
9.          else b=b%a;
10.     if (a!=0) return a;
11.         else return b;
12. }
13.
14. void suma(int a, int b, int c, int d)
15. {
16.     int e, f, nwd_ef;
17.     e = a*d+c*b;
18.     f = b*d;
19.     nwd_ef = NWD(e,f);
20.     cout << e/nwd_ef << "/" << f/nwd_ef << endl;
21. }
22.
23. int main()
24. {
25.     int licz_1, mian_1, licz_2, mian_2;
26.
27.     cout << "Liczba 1 - licznik: "; cin>>licz_1;
28.     cout << "Liczba 1 - mianownik: "; cin>>mian_1;
29.     cout << "Liczba 2 - licznik: "; cin>>licz_2;
30.     cout << "Liczba 2 - mianownik: "; cin>>mian_2;
31.
32.     cout << endl << licz_1 << "/" << mian_1 << " + ";
```

```
33.     cout << licz_2 << "/" << mian_2 << " = ";
34.     suma(licz_1, mian_1, licz_2, mian_2);
35.
36.     return 0;
37. }
```

W programie zastosowaliśmy funkcję wyznaczającą NWD zgodnie z algorytmem Euklidesa (w wersji z dzieleniem). W linii 14 użyto funkcji typu **void** (z ang. próżnia), która jest typem **funkcji niezwracającej wartości**. Jej zadaniem jest wykonanie obliczeń i wyświetlenie efektu na ekranie.

Funkcja niezwracająca wartości

Ćwiczenie 7

Zapisz powyższy kod źródłowy w pliku *suma_ulamkow.cpp*. Skompiluj kod i uruchom program dla różnych par ułamków.

Dodawanie ułamków – metoda szkolna

Kiedy na lekcjach matematyki dodawaliście ułamki, sprowadzaliście je najpierw do wspólnego mianownika, którym była najmniejsza wspólna wielokrotność mianowników ułamków.

Na przykład:

$$\frac{5}{12}+\frac{5}{18}+\frac{5\cdot\frac{36}{12}+5\cdot\frac{36}{18}}{36}=\frac{15+10}{36}=\frac{25}{36}$$

Przypomnijmy, że **najmniejsza wspólna wielokrotność** (w skrócie: **NWW**) dwóch liczb całkowitych dodatnich a i b to najmniejsza liczba naturalna, która jest podzielna przez każdą z tych liczb. Jeśli liczby a i b są względnie pierwsze, to $\mathrm{NWW}(a, b) = a \cdot b$.

Najmniejsza wspólna wielokrotność (NWW)

Okazuje się, że prawdziwa jest równość pokazująca **zależność między NWD a NWW** liczb a i b:

Zależność między NWD a NWW

$$\mathrm{NWW}(a, b)=\frac{a\cdot b}{\mathrm{NWD}(a, b)}$$

Równość tę można wyjaśnić następująco: jeśli a i b mają wspólne dzielniki, to w iloczynie $a \cdot b$ każdy z nich występuje dwukrotnie – raz w a, a raz w b. Zatem aby uzyskać $\mathrm{NWW}(a, b)$, należy iloczyn $a \cdot b$ podzielić przez iloczyn wspólnych dzielników, którym jest $\mathrm{NWD}(a, b)$.

Aby dodać dwa ułamki $\frac{a}{b}$ oraz $\frac{c}{d}$, gdzie a i b oraz c i d są liczbami względnie pierwszymi, można wykorzystać następujący wzór:

$$\frac{a}{b}+\frac{c}{d}=\frac{a\cdot\frac{\mathrm{NWW}(b, d)}{b}+c\cdot\frac{\mathrm{NWW}(b, d)}{d}}{\mathrm{NWW}(b, d)}=\frac{e}{f}$$

Ćwiczenie 8

a. Zgodnie z przedstawionym wzorem dodaj ułamki $\frac{5}{12}$ i $\frac{3}{20}$.

b. Czy po zastosowaniu tego wzoru licznik i mianownik sumy dwóch ułamków (liczby e i f) zawsze będą względnie pierwsze?

Oto zapis kodu funkcji suma, realizującej algorytm dodawania ułamków metodą szkolną:

Kod źródłowy funkcji dodającej ułamki metodą szkolną

```
1.  void suma(int a, int b, int c, int d)
2.  {
3.      int NWW, e, f;
4.      NWW = b*d/NWD(b,d);
5.      e = a*NWW/b+c*NWW/d;
6.      f = NWW;
7.      cout << e << "/" << f << endl;
8.  }
```

Zauważ, że w kodzie korzystamy z algorytmu Euklidesa zastosowanego w funkcji NWD oraz z wzoru określającego zależność między NWD i NWW (kod funkcji NWD – s. 162).

Ćwiczenie 9

W pliku *suma_ulamkow2.cpp* zapisz kod źródłowy programu obliczającego sumę ułamków z wykorzystaniem NWW. Możesz wykorzystać plik *suma_ulamkow.cpp* utworzony w ćwiczeniu 7. Skompiluj kod i uruchom program dla różnych par ułamków.

A to ciekawe

Starożytne ułamki

Dzięki odkryciom archeologicznym wiemy, że starożytni Egipcjanie potrafili stosować ułamki, a dokładniej rozkład liczb na ułamki o licznikach równych 1. Pochodzący z ok. 1650 r. p.n.e. papirus Rhinda (od nazwiska XIX-wiecznego szkockiego kolekcjonera, który go kupił) zawiera następujące zadanie: podziel 7 bochenków chleba między 8 osób. Stosując dzisiejszy zapis, rozwiązanie Egipcjan można w skrócie przedstawić: 7/8 = 1/2 + 1/4 + 1/8. To znaczy, że każdej osobie należy dać połowę, ćwiartkę i jeszcze 1/8 bochenka.



Mnożenie ułamków

Kiedy mamy za zadanie pomnożyć dwa ułamki, możemy pomnożyć przez siebie liczniki i mianowniki, a następnie skrócić otrzymany wynik.

Żeby wykonywać operacje na mniejszych liczbach, skracamy najpierw ułamki „na krzyż" (tak samo postępujecie na lekcjach matematyki):

$$\frac{a}{b}\cdot\frac{c}{d}=\frac{\frac{a}{\mathrm{NWD}(a,d)}\cdot\frac{c}{\mathrm{NWD}(b,c)}}{\frac{\mathrm{b}}{\mathrm{NWD}(b,c)}\cdot\frac{d}{\mathrm{NWD}(a,d)}}=\frac{e}{f}$$

Zapiszmy specyfikację problemu:

Specyfikacja

Dane: cztery liczby całkowite – licznik i mianownik jednego ułamka oraz licznik i mianownik drugiego ułamka.

Wynik: dwie liczby całkowite – licznik i mianownik iloczynu ułamków, zapisane w postaci nieskracalnej.

Oto zapis kodu funkcji `iloczyn`, realizującej algorytm mnożenia ułamków z wykorzystaniem NWD:

Kod źródłowy funkcji mnożącej ułamki

```
5.  void iloczyn(int a, int b, int c, int d)
6.  {
7.      int e, f, nwd_ef;
8.
9.      e = a/nwd(a,d) * c/nwd(b,c);
10.     f = b/nwd(b,c) * d/nwd(a,d);
11.
12.     nwd_ef = nwd(e,f);
13.
14.     cout << e/nwd_ef << endl;
15.     cout << f/nwd_ef << endl;
16. }
```

Ćwiczenie 10

W pliku *iloczyn_ulamkow.cpp* zapisz pełny kod źródłowy programu wyznaczającego iloczyn dwóch ułamków. Możesz skorzystać z kodów źródłowych utworzonych w trakcie rozwiązywania ćwiczeń 7 i 9. Skompiluj kod i uruchom program dla różnych par ułamków.

Podsumowanie

- Największy wspólny dzielnik (NWD) dwóch liczb całkowitych dodatnich a i b to największa liczba całkowita, która jest dzielnikiem obu tych liczb. Symbolicznie zapisuje się ją jako NWD(a,b).
- Algorytm Euklidesa służy do znajdowania największego wspólnego dzielnika (NWD) dwóch liczb naturalnych. W algorytmie tym można wykorzystać odejmowanie – wtedy jeśli od większej liczby odejmiemy mniejszą, to mniejsza liczba i otrzymana różnica będą miały NWD taki jak pierwotne liczby. Ten schemat powtarzamy tak długo, aż liczby się zrównają – będą one równe szukanemu NWD.
- W algorytmie Euklidesa zamiast odejmowania można wykorzystać dzielenie. W tej wersji algorytmu wyznaczamy resztę z dzielenia większej liczby przez mniejszą. Działanie to wykonujemy, aż reszta będzie równa 0. Wtedy mniejsza liczba (dzielnik) jest poszukiwanym NWD.
- Liczbę wymierną można reprezentować w programie jako dwie zmienne całkowite określające odpowiednio licznik i mianownik ułamka.
- W działaniach na liczbach wymiernych wykorzystujemy algorytm Euklidesa do skracania ułamka oraz sprowadzania ułamków do wspólnego mianownika.
- Funkcja typu `void` nie zwraca żadnego wyniku, wykonuje tylko określone czynności.

Zadania

* **1** Napisz program, który po wczytaniu wymiarów patchworkowej kołdry (w centymetrach) wyznaczy liczbę kawałków materiału potrzebnych do jej wykonania. Kołdra ma się składać z jak największych kwadratów tej samej wielkości.

* **2** Napisz program, który z ułamka niewłaściwego wyodrębni część całkowitą i część ułamkową.

* **3** Napisz program, który – korzystając z algorytmu Euklidesa – będzie sprawdzać, czy dane dwie liczby naturalne są liczbami względnie pierwszymi.

* **4** Napisz program, który będzie wczytywać licznik i mianownik ułamka o wartości mniejszej niż 1/2 i sprawdzać, ile razy ten ułamek mieści się w liczbie 1.

* **5** Napisz program, który wczyta dwa ułamki i wyświetli większy z nich wraz z komunikatem „jest większą liczbą". Jeśli użytkownik wprowadzi te same ułamki, program wyświetli komunikat „Ułamki są równe".

** **6** Napisz program, który wczytuje dwie dodatnie i względnie pierwsze liczby a i b, a następnie wyznacza dla nich najmniejszą możliwą liczbę x, dla której ułamek $\frac{a+x}{b+x}$ jest nieskracalny.

** **7** Napisz program, który wczyta trzy liczby całkowite dodatnie i wyznaczy ich największy wspólny dzielnik.
Wskazówka: Największy wspólny dzielnik trzech liczb to największy wspólny dzielnik jednej z liczb i liczby, która jest największym dzielnikiem pozostałych.

** **8** Zmodyfikuj program *Suma ułamków* tak, aby można było w nim dodawać trzy ułamki.

** **9** Uzupełnij program *Suma ułamków* o wyodrębnianie części całkowitej i części ułamkowej dla sumy, która jest ułamkiem niewłaściwym.

** **10** Napisz program zawierający funkcję o nagłówku `int NWW(int a, int b)`, która będzie wyznaczać najmniejszą wspólną wielokrotność liczb a i b.

** **11** Zapisz specyfikację problemu dzielenia dwóch ułamków. Następnie napisz listę kroków algorytmu i program zawierający funkcję `iloraz`, wykonującą dzielenie.

** **12** Napisz w języku C++ prostą grę dwuosobową, w której pierwszy z graczy podaje dwie liczby naturalne mniejsze od 200, a drugi gracz próbuje odgadnąć ich NWD. Kolejne próby powinny być numerowane. Po odgadnięciu gracze zamieniają się rolami. Wygrywa ten, który poda poprawny wynik w mniejszej liczbie prób.

*** **13** Zapisz specyfikację problemu odejmowania dwóch ułamków. Następnie napisz listę kroków algorytmu i program zawierający funkcję `roznica`, odejmującą dwa ułamki. Umieść w nim również podprogram, który przetestuje działanie tej funkcji.

*** **14** Udowodnij prawdziwość wzoru na mnożenie ułamków, w którym skracamy ułamki „na krzyż".

*** **15** Napisz program, który spośród pięciu różnych dodatnich liczb całkowitych podanych przez użytkownika wybierze takie dwie, które będą stanowiły licznik i mianownik ułamka o możliwie najmniejszej wartości. Rozwiązanie przygotuj dwoma sposobami:
a. z wykorzystaniem tablicy pięcioelementowej,
b. bez korzystania z tablic.

*** **16** Napisz program, który wczyta trzy dodatnie liczby jednocyfrowe a, b i c takie, że $a \neq b$. Efektem powinna być informacja o położeniu cyfry c w rozwinięciu dziesiętnym ułamka a/b. Jeśli taka cyfra się nie pojawia, program powinien wypisać liczbę -1.

*** **17** Napisz program, który zrealizuje algorytm Herona według listy kroków podanej na s. 98.

Algorytmika i programowanie w języku C++

Wyobraź sobie, że...



...zaczynasz studia na politechnice, ponieważ chcesz zostać inżynierem geodetą. Okazuje się, że na pierwszym roku masz zajęcia z programowania. Na szczęście w szkole poznałaś **podstawowe konstrukcje programistyczne**, **typy danych** i **proste struktury danych** oraz zdobyłaś pewne doświadczenie w programowaniu w **środowisku tekstowym**. Radzisz sobie, rozumiesz omawiane algorytmy. Potrafisz je też zaprogramować, mimo że używasz innego języka programowania niż w szkole.



...jesteś inżynierem pożarnictwa i pracujesz w firmie opracowującej zasady bezpieczeństwa dla obiektów biurowych. Wiesz, że instrukcje zawierające sposób postępowania na wypadek pożaru lub innego zagrożenia muszą być bardzo precyzyjne. Jednym ze sposobów ich zapisania jest forma **listy kroków**, z którą zetknąłeś się w szkole podczas nauki programowania. Twoja praca cieszy się dużym uznaniem, ponieważ instrukcje tworzone przez ciebie są dla wszystkich zrozumiałe.



...skończyłaś studia medyczne i odbyłaś staż. Trafiasz na ofertę pracy w projekcie badawczym, w ramach którego ma powstać oprogramowanie wykorzystywane do kształcenia przyszłych lekarzy w Afryce. W czasie rozmowy kwalifikacyjnej sprawdzana jest m.in. znajomość podstawowych pojęć związanych z projektowaniem oprogramowania, takich jak **sytuacja problemowa** i **specyfikacja problemu**. Dostajesz pracę. Współpracując z programistami, możesz zmieniać świat na lepsze.



...pracujesz w firmie produkującej meble, która wdraża system wsparcia produkcji wyposażony w sztuczną inteligencję. Na szkoleniu pokazano model sztucznego neuronu wraz z **kodem źródłowym**. Na jego podstawie wyjaśniono podstawę działania całego systemu. Jesteś zachwycony, ponieważ do zrozumienia kodu wystarczyła wiedza o **instrukcji warunkowej** i **pętlach** wyniesiona ze szkoły.



3 Algorytmika i programowanie w języku Python

10. Od problemu do programu
11. Wyszukujemy i sumujemy
12. Binarny system liczbowy
13. Czy ta liczba jest pierwsza?
14. Algorytm Euklidesa i działania na ułamkach

Algorytmika i programowanie w języku Python

Technika i matematyka są związane ze sobą od wieków. Urządzenia ułatwiające obliczenia i przetwarzające ich wyniki zawsze były rezultatem wykorzystania algorytmów matematycznych i rozwiązań technicznych dostępnych w danej epoce. Współcześnie algorytmy zapisuje się w językach programowania, które pozwalają tworzyć oprogramowanie.



1800 r. p.n.e.

Około 1800 r. p.n.e. starożytni Egipcjanie odkryli algorytm mnożenia – nazwany potem **mnożeniem egipskim** – w którym używano wyłącznie operacji podwajania i dodawania.

VII w. p.n.e.



Grecki poeta Archiloch w jednym z dzieł wspomniał o przyrządzie do szyfrowania zwanym *skytálē*, używanym w Sparcie od VII w. p.n.e. Nadawca nawijał na laskę pasek pergaminu i pisał tekst na stykających się brzegach. Adresat posiadał identyczną laskę, umożliwiającą odczytanie wiadomości. Dziś ten **szyfr przestawieniowy** nazywa się metodą płotu.

300 r. p.n.e.

Jednym z najstarszych algorytmów obliczeniowych jest **algorytm Euklidesa**, czyli metoda wyznaczania największego wspólnego dzielnika dwóch liczb, opisana przez greckiego matematyka Euklidesa w dziele *Elementy*. Pierwotnie algorytm ten służył do szukania odcinka, który byłby wspólną miarą dla dwóch dłuższych odcinków.



I w. p.n.e.

Pierwszym opisanym **szyfrem podstawieniowym** był prawdopodobnie szyfr z czasów wojen galijskich, stosowany w korespondencji przez Juliusza Cezara. Szyfr polega na cyklicznym przesuwaniu liter alfabetu: za literę A podstawiano literę D, za B – E itd., a za Z – C.



I w. n.e.

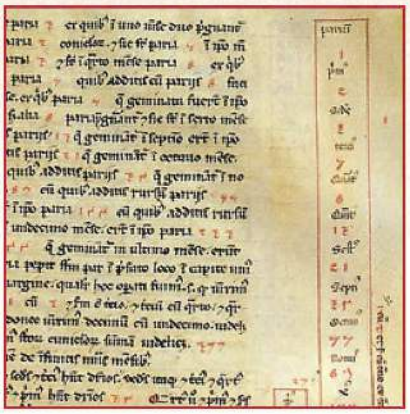


Heron z Aleksandrii w dziele *Metrica* opisał efektywną metodę iteracyjnego wyznaczania przybliżeń pierwiastka kwadratowego z użyciem operacji dodawania, mnożenia i dzielenia. Dziś tę metodę nazywa się **algorytmem Herona**.

IX w.



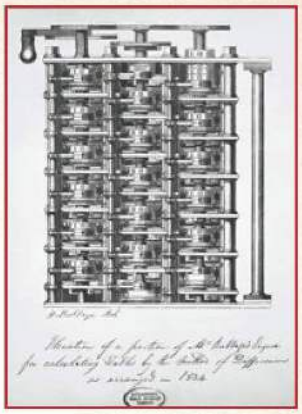
Muhammad Ibn Musa Al-Chuwarizmi, bibliotekarz Domu Mądrości w Bagdadzie, opisał efektywne sposoby pisemnego wykonywania obliczeń z użyciem systemu dziesiętnego. Od łacińskiej wersji nazwiska tego uczonego pochodzi słowo **algorytm**.



Leonardo Fibonacci w dziele *Liber abaci* z 1202 r. rozważał problem rozmnażania się populacji królików. W rozwiązaniu pojawia się ciąg liczb:
1, 1, 2, 3, 5, 8, 13, 21, ...,
w którym: 2 = 1 + 1, 3 = 1 + 2, 5 = 2 + 3, 8 = 5 + 3 itd. Ciąg ten opisuje **zależność rekurencyjna**: następna liczba jest sumą dwóch poprzednich. Kolejne liczby Fibonacci wypisał na prawym marginesie rękopisu.

W II połowie XVII w. Gottfried Wilhelm Leibniz opisał **dwójkowy system liczbowy**, w którym każdą liczbę można zapisać za pomocą kombinacji cyfr 0 i 1. Obecnie taki kod zero-jedynkowy (binarny) stanowi podstawę działania komputerów.





W 1837 r. Charles Babbage ogłosił koncepcję parowej **maszyny analitycznej**. Miała się ona składać z magazynu danych i pośrednich wyników obliczeń, młyna (jednostki liczącej) oraz mechanizmu sterującego, programowanego z użyciem kart perforowanych. Projekt nie doczekał się realizacji z powodu zbyt niskiego poziomu ówczesnej techniki.

1202 r.

1614 r.



John Napier w 1614 r. opisał metodę logarytmów, czyli metodę mnożenia liczb za pomocą dodawania innych liczb, które można zapisać w tablicy. Napier wynalazł przyrząd, nazywany kostkami Napiera, który uznaje się za **prototyp komputera analogowego**.

1679 r.

1804 r.

Na początku XIX w. Joseph Marie Jacquard zbudował **programowalną maszynę** tkacką, która automatycznie wykonywała wzór tkaniny dzięki kartom perforowanym. Aż do lat 80. XX w. karty i taśmy perforowane były nośnikami danych dla komputerów.



1837 r.

1843 r.



Za **pierwszego programistę** niektórzy uważają Adę Lovelace. Jej notatki dotyczące maszyny analitycznej, wydane w 1843 r., zawierały diagram algorytmu napisany z zamiarem jego wykonania na fizycznej maszynie. W pracach Lovelace pojawiają się koncepcje: zmiennych, instrukcji warunkowej, pętli i funkcji.

10. Od problemu do programu

Być może tytuł tego tematu wydaje ci się nieco tajemniczy. W tym krótkim haśle zawiera się jednak główna idea całego rozdziału: programowanie pozwala urzeczywistnić pomysły, które mogą rozwiązać konkretne problemy współczesnego świata i zmienić życie ludzi na lepsze. Nawet jeśli nie zamierzasz w przyszłości zostać programistą, warto poznać elementy sztuki programowania. W tym temacie nauczysz się podstaw informatycznego rozwiązywania problemów z użyciem języka Python. Dzięki temu lepiej zrozumiesz otaczający cię świat cyfrowy, a zdobyte umiejętności przydadzą ci się w przyszłej współpracy z programistami.

Cele lekcji

- Zrozumiesz etapy myślenia informatycznego.
- Poznasz przykłady algorytmów wykorzystywanych w życiu codziennym.
- Dowiesz się, jak poprawnie zapisać algorytm i czym jest lista kroków.
- Nauczysz się posługiwać zmiennymi oraz stosować konstrukcje programistyczne: instrukcje wejścia i wyjścia, wyrażenia arytmetyczne i logiczne oraz instrukcję warunkową.
- Napiszesz w języku programowania Python proste programy rozwiązujące problemy z różnych dziedzin.

10.1. Algorytm, czyli rozwiązanie problemu

Każde urządzenie cyfrowe, z którego korzystamy, zawiera oprócz systemu operacyjnego pewne oprogramowanie użytkowe. Zanim jednak powstanie program, musi się pojawić potrzeba jego stworzenia. Okoliczność, która tworzy tę potrzebę, będziemy nazywać problemem lub sytuacją problemową.

Aby stworzyć program komputerowy rozwiązujący konkretny problem, trzeba najpierw zaprojektować samo rozwiązanie problemu. Musi ono mieć formę **algorytmu** (ang. *algorithm*). W tym i kolejnych tematach będziemy postępować według schematu: problem – algorytm – program komputerowy.

Algorytm to opis postępowania prowadzącego do rozwiązania problemu.

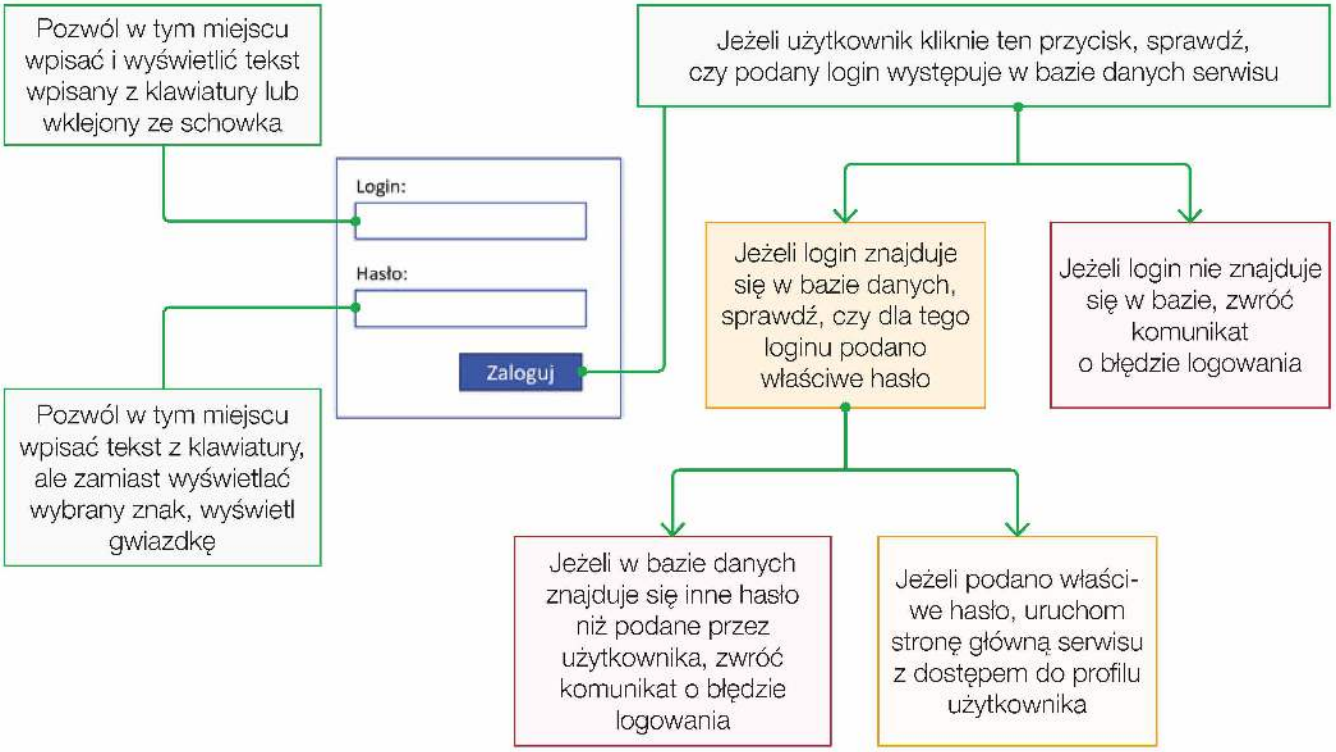
Ćwiczenie 1

Zastanów się, jak utworzyć silne hasło zabezpieczające konto internetowe. Przedstaw jeden ze sposobów konstruowania hasła w formie kolejnych kroków.

Omówimy teraz kilka przykładów problemów i pokażemy algorytmy ich rozwiązania.

Problem 1. Logowanie do serwisu internetowego

Zapewne nie raz zdarzyło ci się logować do serwisu internetowego. Zastanówmy się, jak mogłyby wyglądać wytyczne dla programisty takiego serwisu dotyczące okna logowania (rys. 10.1).



Rys. 10.1. Logowanie do serwisu internetowego – opis działania elementów

Stworzenie takiego mechanizmu wynikało z potrzeby zidentyfikowania (uwierzytelnienia) użytkownika. W tym celu sprawdza się poprawność loginu i hasła, które podał logujący się. Jeśli obie te wartości zgadzają się z danymi zapisanymi w bazie danych serwisu (serwera), to użytkownik może wejść na swoje konto w serwisie. Wystarczy jednak, że jedna z danych się nie zgadza, aby zwrócić komunikat o błędnym loginie lub haśle.

To jest właśnie najprostsza, opisowa wersja algorytmu logowania, którą następnie można by zaprogramować.

Warto wiedzieć

Hasła zazwyczaj są przechowywane na serwerze w formie zahaszowanej (czyli zaszyfrowanej specjalnym algorytmem). Podczas logowania podane hasło jest ponownie haszowane, a następnie sprawdza się jego zgodność z kodem haszowania zapisanym w bazie.

Ćwiczenie 2

Zaproponuj schemat objaśniający proces:

a. wypożyczania książki w szkolnej bibliotece,

b. zakładania konta poczty elektronicznej.

Problem 2. Jak zapleść warkocz klasyczny?

Wyobraź sobie, że ktoś zapytał cię, jak zapleść warkocz klasyczny. Jeśli potrafisz to zrobić, zapewne najłatwiej będzie ci pokazać to na przykładzie. W języku informatyki powiemy w takiej sytuacji, że zademonstrujesz działanie algorytmu zaplatania warkocza.

Nie zawsze jednak można taki pokaz przeprowadzić, dlatego czasem potrzebujemy uniwersalnego i jednoznacznego opisu rozwiązania zadania. Może on mieć formę np. listy kroków.

Lista kroków

Lista kroków to jeden ze sposobów precyzyjnego opisu postępowania. Składa się z zapisanych słownie kolejnych pojedynczych instrukcji, nazywanych krokami, które mogą się odwoływać do siebie nawzajem.

Lista kroków algorytmu zaplatania warkocza może wyglądać następująco:

1. `Rozdziel włosy na trzy pasma.`
2. `Dopóki nie spleciesz warkocza po końcówki włosów, wykonuj na zmianę krok 3 i krok 4.`
   3. `Przełóż lewe pasmo nad środkowym. Teraz to pasmo staje się środkowym.`
   4. `Przełóż prawe pasmo nad środkowym. Teraz to pasmo staje się środkowym.`

Warto wiedzieć

Ludzie od tysięcy lat zaplatają warkocze – przedstawiają je m.in. rysunki pochodzące ze starożytnego Egiptu. Dziś znanych jest wiele sposobów zaplatania warkocza. Wersja podstawowa to warkocz klasyczny (zdjęcie poniżej).



Zauważ, że kroki 3 i 4 są wcięte względem kroków 1 i 2. W ten sposób zaznaczamy, że zależą one od warunku podanego w kroku 2 (ostatnim kroku przed wcięciem). W naszym przypadku wykonywanie kroków 3 i 4 zależy od tego, czy dotychczas zapleciony warkocz sięga końcówek włosów. Zauważ ponadto, że tak podany opis:

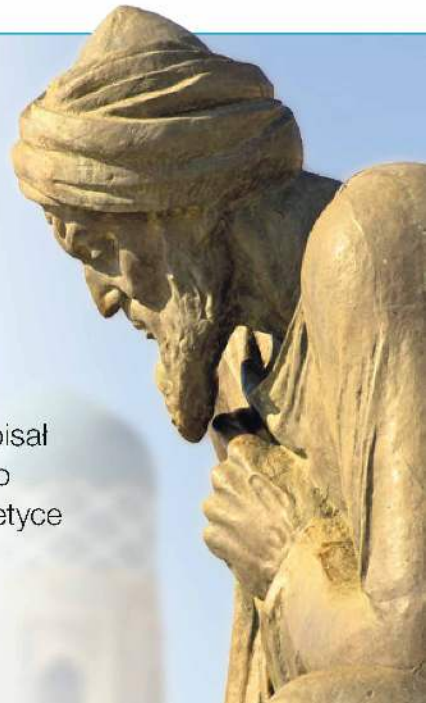
- zawiera informację o tym, na jakich danych wejściowych będziemy wykonywać operacje (w naszym przykładzie są to włosy),
- zwraca określony wynik (zapleciony warkocz),
- jest skończony – można go zrealizować w skończonej liczbie kroków (warunek zakończenia znajduje się w kroku 2),
- jest precyzyjny (pasma włosów mają swoje nazwy, czynności są jednoznacznie opisane),
- jest uniwersalny – można go zastosować do każdych włosów odpowiedniej długości.

Cechy poprawnego algorytmu

Są to **cechy poprawnego algorytmu**.

A to ciekawe

Skąd pochodzi słowo „algorytm”?



Słowo „algorytm” pochodzi od *Algorithmus* – łacińskiego przekładu ostatniego z członów nazwiska perskiego matematyka i astronoma, który był znany jako Muhammad Ibn Musa Al-Chuwarizmi. Żył on na przełomie VIII i IX w. n.e. i był bibliotekarzem Domu Mądrości w Bagdadzie. Zajmował się m.in. pozycyjnym systemem dziesiętnym zapisu liczb, znanym wcześniej wyłącznie w Indiach, i opisał efektywne sposoby pisemnego wykonywania obliczeń. W XII w. przetłumaczono jego dzieło na łacinę, co zapoczątkowało w Europie stopniowe odejście w arytmetyce handlowej od posługiwania się żetonami i stołami obliczeniowymi (abakami).

Ćwiczenie 3

a. W podanej instrukcji zaplatania warkocza wskaż te czynności, które można uznać za podstawowe (niepodzielne). Kiedy wykonywanie algorytmu powinno się zakończyć?

b. Dowiedz się, jak zawiązać tzw. węzeł ósemkowy. Zapisz algorytm w formie listy kroków.

Problem 3. Wyznaczanie pierwiastka kwadratowego

Dokładność przybliżeń stosowanych w komputerach odgrywa kluczową rolę w wielu dziedzinach, np. w medycynie, kosmonautyce, architekturze czy na rynkach finansowych.

Zapoznamy się z metodą znajdowania przybliżonej wartości pierwiastka kwadratowego przedstawioną przez Herona z Aleksandrii w I w. n.e. Używa się w niej wyłącznie prostych operacji arytmetycznych: dodawania oraz dzielenia i można ją zastosować do współczesnych obliczeń wykorzystujących wartość pierwiastka kwadratowego.

Opis algorytmu poprzedzimy stworzeniem tzw. **specyfikacji problemu**, czyli określeniem związku między danymi a wynikiem.

Specyfikacja problemu

Specyfikacja

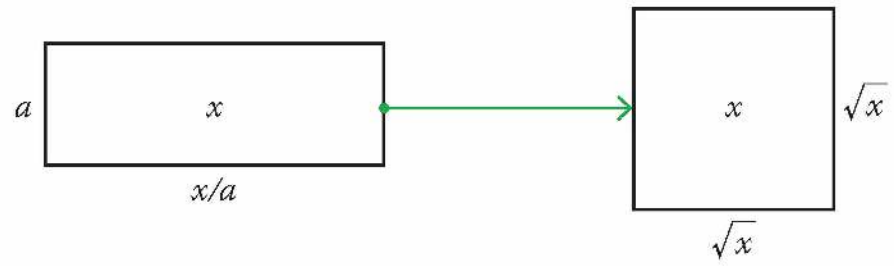
Dane: liczba x.

Wynik: przybliżenie $\sqrt{x}$ z dokładnością do 0,001.

Liczbę x możemy potraktować jako pole powierzchni prostokąta o długościach boków a oraz x/a (rys. 10.2). Metoda Herona polega na przemyślanej zmianie długości boku a (przy jednoczesnej zmianie długości drugiego boku prostokąta) tak, aby bez zmiany pola powierzchni prostokąta otrzymać figurę jak najbardziej zbliżoną do kwadratu. Po kilkukrotnym powtórzeniu tej procedury długość boku a przyjmie wartość równą w przybliżeniu $\sqrt{x}$.

Warto wiedzieć

Heron pisał dzieła o charakterze praktycznym, dziś powiedzielibyśmy – inżynierskie. Pokazywał na przykładach m.in. metody obliczania pola powierzchni działek rolnych oraz objętości brył.



Rys. 10.2. Interpretacja geometryczna szukania przybliżenia liczby $\sqrt{x}$

Modyfikacja długości boku a oznacza, że jeden z boków prostokąta skracamy, a drugi wydłużamy, aż różnica między nimi będzie wynosiła 0 albo będzie nie większa niż błąd podany w specyfikacji.

Do właściwej wartości a (przybliżenia $\sqrt{x}$) dochodzimy w kolejnych krokach, w których tworzymy kolejne prostokąty.

Warto wiedzieć

Prawdziwa jest zależność: jeśli $|a - x/a| < 0{,}001$, to $|\sqrt{x} - a| < 0{,}001$. Wynika to stąd, że a i x/a są przybliżeniami $\sqrt{x}$ z góry i z dołu (jeśli $a > \sqrt{x}$, to $x/a < \sqrt{x}$ i na odwrót). Jeśli odległość między a i x/a jest mniejsza od 0,001, to tym bardziej odległość a od $\sqrt{x}$ musi być mniejsza od 0,001.

Za długość boku a kolejnego prostokąta przyjmiemy średnią arytmetyczną długości boków poprzedniego prostokąta: $(a + x/a)/2$. Drugi z boków tego prostokąta wyznaczymy już jako nowy iloraz x/a dla zmienionego a. Długość a traktujemy jako zmienną, do której przypisaliśmy wartość $(a + x/a)/2$. Jeżeli różnica między długościami boków nowego prostokąta jest dostatecznie mała (w tym wypadku wynosi mniej niż 0,001), to a jest szukanym przybliżeniem. Jeżeli tak nie jest, tworzymy kolejny prostokąt o boku $(a + x/a)/2$ i polu x. I tak dalej.

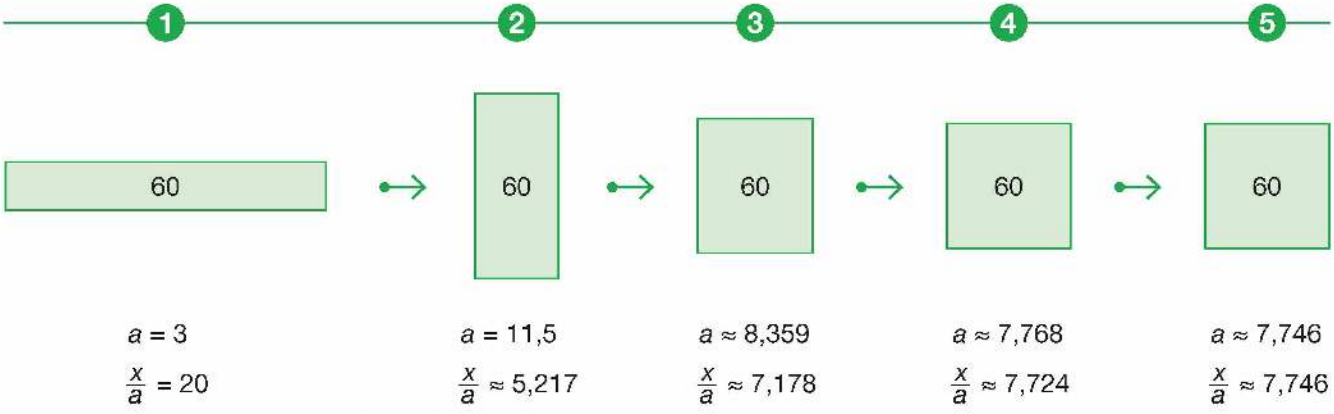
Skąd wziąć początkową wartość a? Może nią być dowolna liczba, np. 1. Jednak im bliższa pierwiastkowi z danej liczby jest pierwsza wartość a, tym szybciej algorytm wyznaczy przybliżenie pierwiastka.

Algorytm Herona

Zapiszmy listę kroków **algorytmu Herona**:

1. `Za a przyjmij dowolną liczbę większą od 0.`
2. `Dopóki wartość |a - x/a| jest większa niż 0,001, powtarzaj kroki 3 i 4.`
   3. `Przyjmij za a średnią arytmetyczną a i x/a.`
   4. `Oblicz różnicę a i x/a.`

Korzystając z listy kroków algorytmu Herona, obliczmy przybliżenie $\sqrt{60}$. Za wartość początkową a przyjmijmy 3. Na rysunku 10.3 pokazano kolejne przybliżenia kwadratu o boku długości $\sqrt{60} \approx 7{,}746$.



Rys. 10.3. Zastosowanie algorytmu Herona dla $x = 60$

Warto wiedzieć

Komputery przeznaczają znaczną część czasu pracy na obliczenia wykorzystujące operacje arytmetyczne. Są one potrzebne np. w grafice komputerowej lub analizie danych. Jedną z takich operacji jest obliczanie pierwiastka kwadratowego.

Zauważ, że na schemacie (rys. 10.3) etap 5 kończy działanie algorytmu, ponieważ udało się znaleźć przybliżenie z dokładnością do trzech miejsc po przecinku. Algorytm można by wykonywać dalej, jeśli zależałoby nam na większej precyzji. Można by go również zakończyć na etapie 4 – wówczas dla wartości $a \approx 7{,}768$ mamy $a - x/a \approx 0{,}044$. Jeśli kwadrat tej wartości jest akceptowalnym przybliżeniem, możemy przyjąć, że $\sqrt{60} \approx 7{,}768$.

Ćwiczenie 4

Opracuj plik arkusza kalkulacyjnego, który będzie obliczał przybliżenie pierwiastka kwadratowego zgodnie z algorytmem Herona.

Zapamiętaj

Algorytm to opis postępowania prowadzącego do rozwiązania problemu. Specyfikacja problemu algorytmicznego to opis danych (wraz z warunkami, jakie mają spełniać) oraz wyników i ich związku z danymi.

10.2. Od algorytmu do programu w języku Python

Zapis algorytmu w języku programowania nazywamy tekstem programu lub **kodem źródłowym** programu komputerowego. Na kod źródłowy programu, niezależnie od języka programowania, składają się dwie podstawowe konstrukcje programistyczne: instrukcje i funkcje.

Kod źródłowy programu komputerowego musi być przetłumaczony przez **translatory** na kod wykonywalny (maszynowy) właściwy dla procesora i systemu operacyjnego. Translatory, które tłumaczą od razu cały kod, nazywamy **kompilatorami** (ang. *compilers*). Natomiast translatory, w których tłumaczenie odbywa się instrukcja po instrukcji, nazywamy **interpreterami** (ang. *interpreters*). Przykładem języka kompilowanego jest **język C++**, a interpretowanego – **język Python**.

Instrukcja to polecenie, które kompilator lub interpreter potrafi zrozumieć i wykonać. Przyjmuje ono formę zdania rozkazującego (zawierającego zwroty typu „Powiedz", „Zatrzymaj się", „Wczytaj", „Wyświetl"). Pojedyncze instrukcje składa się zazwyczaj w większą całość nazywaną **funkcją**.

Podamy teraz kilka podstawowych zasad zapisywania kodu w języku Python. Język ten, jak każdy inny język programowania, opiera się na sformalizowanych regułach tworzenia kodu. Zawiera również listę zastrzeżonych słów, które są nazywane **słowami kluczowymi**. Można ich używać tylko w konkretnym celu, który określili twórcy języka.

- Kod źródłowy
- Translator
- Kompilator
- Interpreter
- Język C++, język Python
- Instrukcja
- Funkcja
- Słowa kluczowe języka programowania

A to ciekawe

Ranking języków programowania

Istnieje wiele internetowych rankingów popularności języków programowania. Część z nich mierzy liczbę zapytań pozwalających odnaleźć w internecie kursy programowania w danym języku. Inne rankingi biorą pod uwagę serwisy społecznościowe dla programistów i mierzą liczbę wątków dotyczących danego języka. W czołówce najpopularniejszych języków programowania ostatnich lat znajdują się języki: Java, JavaScript, Pyhton, C/C++, C#, PHP.



Zapisywanie programów w języku Python

Kody źródłowe języka Python zapisujemy w edytorze kodu połączonym z interpreterem. Powszechnie korzysta się ze zintegrowanych środowisk programistycznych (IDE). Jednym z nich jest program Mu, który automatycznie koloruje elementy składniowe kodu (np. słowa kluczowe języka, teksty i liczby).



Przykładowy kod źródłowy programu w języku Python

Funkcja `print` pozwala wyświetlić tekst na ekranie

Ten zapis sprawia, że program wyświetla komunikat zachęcający do podania nazwy użytkownika, a następnie w zmiennej `login` zapamiętuje wprowadzony tekst, zatwierdzony klawiszem **Enter**

```
1. print("*** Program powitalny ***")
2. login = input("Podaj nazwę użytkownika: ")
3.
4. print("_____________________________________")
5. print("Użytkownik <" + login + "> został programistą języka Python")
6. print("_____________________________________")
```

Funkcja `print` może wyświetlać napisy stałe oraz wartości zmiennych. Znak + jest w tym zapisie operatorem scalającym napisy

Zasady tworzenia kodu w języku Python

Wielkość liter w kodzie ma znaczenie – zmienne a i A mogą przechowywać różne wartości.

Między wartościami a operatorami można wstawiać spacje (np. a + 2).

Kolejne instrukcje należy umieszczać w kolejnych wierszach.

Tekst po znaku # jest komentarzem – interpreter go ignoruje.

O typie zmiennej decyduje nadana jej wartość. Wartość wprowadzona z klawiatury domyślnie traktowana jest jako tekst.

Warto stosować w kodzie puste wiersze, ponieważ zwiększają czytelność programu.

Program *Powitanie* – zapisywanie programu w środowisku Mu

Na poprzedniej stronie zapisano tekst programu, który pobiera od użytkownika jego login, a następnie wyświetla na ekranie tekst powitalny.

Obok kodu znajdują się liczby, które są numerami wierszy (linii) kodu. Linie kodu będziemy numerować, żeby łatwiej było omawiać poszczególne zapisy. W linii 1 znajduje się wywołanie **funkcji** `print`, która pozwala wyświetlać tekst (podany w cudzysłowie) na ekranie. Taka instrukcja jest nazywana **instrukcją wyjścia** (ang. *output*). W linii 2 znajduje się tzw. instrukcja wejścia, która pozwala pobrać nazwę użytkownika i przechować ją w zmiennej `login`. Instrukcję wejścia i zmienne omówimy dokładniej na s. 181–182.

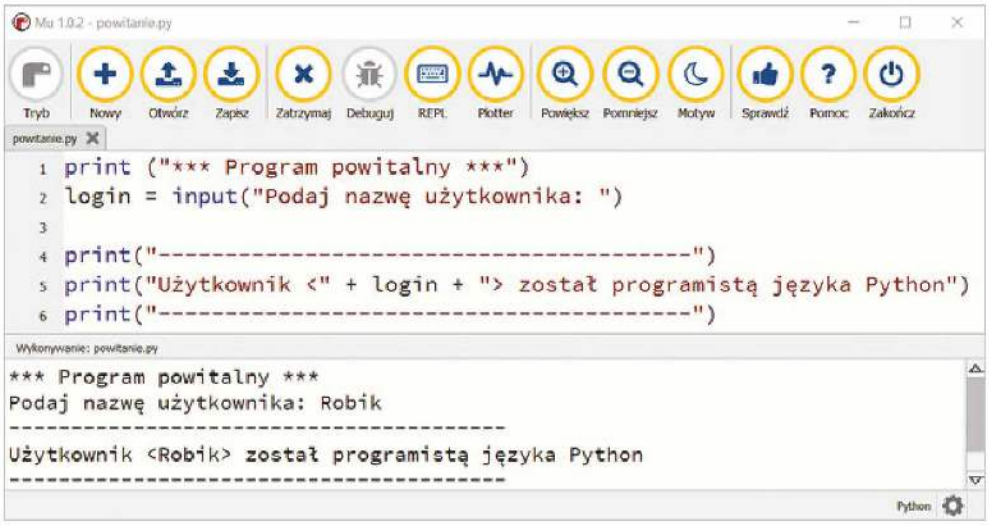
Funkcja `print`

Instrukcja wyjścia

W liniach 4–6 ponownie zastosowano funkcję `print`, ale w linii 5 wykorzystuje ona napisy stałe oraz wartość przechowaną w zmiennej `login` (nazwę podaną przez użytkownika).

Kody źródłowe programów będziemy zapisywać za pomocą edytora środowiska Mu. Pliki źródłowe zawierające kod w języku Python są plikami tekstowymi z rozszerzeniem *py*.

Efekt działania programu *Powitanie* przedstawiono na rysunku 10.4.



Rys. 10.4. Wywołanie programu *Powitanie*

Aby rozpocząć tworzenie kodu w tym środowisku, wybieramy z menu przycisk **Nowy**, a następnie zapisujemy nowo utworzony plik na dysku (przycisk **Zapisz**), np. pod nazwą *powitanie.py*.

Ćwiczenie 5

W edytorze Mu w pliku *powitanie.py* wpisz kod źródłowy programu *Powitanie*. Zapisz zmiany.

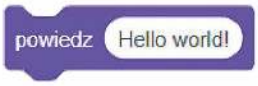
Aby uruchomić napisany program, wystarczy w środowisku Mu wybrać z menu przycisk **Wykonuj** lub nacisnąć klawisz **F5**. Aby zatrzymać działanie programu, należy kliknąć przycisk **Zatrzymaj** lub ponownie wcisnąć klawisz **F5**.

Dobra rada

Aktualną wersję programu Mu możesz pobrać ze strony https://codewith.mu/. W wersji dla systemu Windows do edytora dołączony jest interpreter języka Python 3.

Warto wiedzieć

W języku Scratch za operację wypisania na ekranie komunikatu odpowiada blok **powiedz**:



Dobra rada

Pamiętaj, żeby w edytorze Mu po napisaniu kodu najpierw zapisać plik na komputerze, a dopiero potem uruchomić program. Niezapisany kod nie uruchomi się i wyświetli się okno pozwalające zapisać plik na dysku.

Ćwiczenie 6

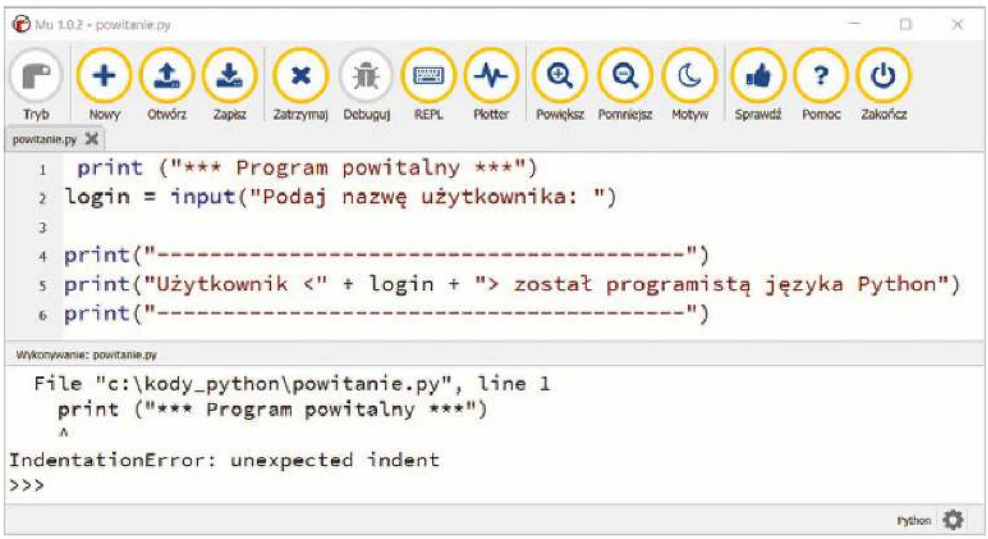
a. Uruchom kod programu zapisany w pliku *powitanie.py*.
b. W pliku *powitanie.py* dodaj spację na początku pierwszego wiersza, przed instrukcją `print`. Zapisz zmianę w pliku źródłowym i spróbuj uruchomić program. Co zauważasz?

Błąd składniowy

Jeśli w kodzie źródłowym wystąpi **błąd składniowy**, czyli taki, który wynika z zapisania kodu niezgodnie z regułami (składnią) danego języka programowania, to próba uruchomienia takiego programu zakończy się wyświetleniem komunikatu o błędzie.

Na przykład jeśli na początku wiersza kodu w programie *powitanie.py* pojawi się dodatkowy odstęp (choćby pojedyncza spacja), to po próbie uruchomienia programu pojawia się komunikat o błędzie składniowym: `unexpected indent`, czyli nieoczekiwane wcięcie, a znak ^ wskazuje miejsce, w którym **interpreter** wykrył błąd (rys. 10.5).

Interpreter, s. 177



Rys. 10.5. Komunikat o błędzie składniowym – zbędna spacja przed instrukcją `print`

W języku Python wcięcia mają kluczowe znaczenie i trzeba je konsekwentnie stosować w całym kodzie. Jeśli przed instrukcjami znajdą się zbędne spacje, program będzie interpretować to jako błąd.

Program *Przeliczanie temperatury* – pojęcie zmiennej i proste operacje arytmetyczne w języku Python

Program *Powitanie* wyświetlał jedynie tekst na ekranie i nie wykonywał żadnych obliczeń. Napiszemy teraz program bardziej praktyczny, który zwróci wynik na podstawie informacji podanych przez użytkownika.

Mamy do rozwiązania następujący problem. Wyobraź sobie, że zespół naukowców z Polski i USA pracuje nad robotem, który weźmie udział w misji kosmicznej. Naukowcy używają różnych skali do zapisu temperatury – stopni Fahrenheita i stopni Celsjusza. Celem jest stworzenie programu, który pozwoli przeliczać wartości temperatury ze skali Fahrenheita na skalę Celsjusza.

Dla uproszczenia przyjmijmy, że odczyt temperatury w stopniach Fahrenheita będzie liczbą całkowitą, a wynik będzie podany w postaci liczby całkowitej z ułamkiem dziesiętnym.

Oto specyfikacja tego problemu:

Specyfikacja

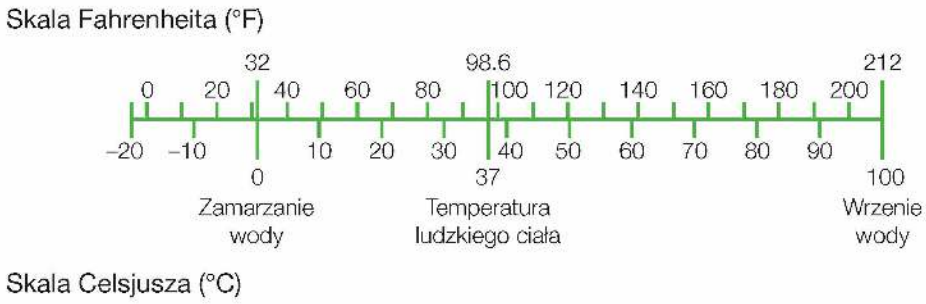
Dane: wartość temperatury w stopniach Fahrenheita podana jako liczba całkowita *f*.

Wynik: wartość temperatury w stopniach Celsjusza.

Na rysunku 10.6 porównano skalę Fahrenheita ze skalą Celsjusza. Skala Fahrenheita jest przesunięta o 32 stopnie (temperatura zamarzania wody to 0°C i 32°F), ale to nie wszystko. Różnica pomiędzy temperaturami zamarzania i wrzenia wody (w warunkach normalnych) w skali Fahrenheita wynosi 180 stopni (od 32 do 212), a w skali Celsjusza 100 stopni (od 0 do 100). Zatem aby przeliczyć skale, od temperatury podanej w stopniach Fahrenheita odejmujemy 32 i wynik mnożymy przez 100/180, czyli 5/9 po skróceniu ułamka.

Warto wiedzieć

Skala Fahrenheita jest obecnie używana m.in. w USA i Belize. W Kanadzie powszechnie wykorzystuje się obie skale temperatur.



Rys. 10.6. Porównanie skali Fahrenheita ze skalą Celsjusza

Algorytm powinien zatem przeliczać temperaturę podaną w skali Fahrenheita (f) na temperaturę w skali Celsjusza (c) według wzoru matematycznego: $c = (f - 32) \cdot \frac{5}{9}$.

Oto kod źródłowy programu *Przeliczanie temperatury*:

Warto wiedzieć

W języku Scratch operację wczytania liczby podanej z klawiatury do zmiennej realizuje się z użyciem bloków **zapytaj i ustaw**:



Kod źródłowy programu *Przeliczanie temperatury*

```
1.  f = int(input("Temperatura w skali Fahrenheita: "))
2.  print(f, " stopni F to ", (f-32)*5/9, " stopni C")
```

Ponieważ program ma przeliczać wartość podaną przez użytkownika, potrzebujemy elementu, który przechowa tę informację. Dlatego w wierszu 1 wprowadzono przypisanie wartości do **zmiennej** o określonej nazwie (f). Oznacza to tyle, że pewien obszar w pamięci komputerowej otrzymał nazwę f i można w nim zapisać informację – w tym przypadku wartość temperatury.

Zmienna

O zmiennych w językach programowania możesz myśleć jak o pudełkach na dane – mogą one mieć różny rozmiar i mogą przechowywać różne rodzaje informacji. W Pythonie typ tych danych jest ustalany podczas uruchamiania programu na podstawie wartości przypisanej do zmiennej. Po rozpoznaniu typu danych program rezerwuje odpowiednią ilość pamięci komputerowej, aby przechowywać wartości w tej zmiennej.

Funkcja input
Instrukcja wejścia

Instrukcja wywołująca **funkcję input** pozwala wczytać do zmiennej liczbę podaną z klawiatury. Jest to tzw. **instrukcja wejścia** (ang. *input*). Domyślnie dane wczytane z klawiatury interpreter traktuje jako ciąg znaków (np. liter, cyfr) – tak jak to robi każdy edytor tekstu. To oznacza, że jeśli podczas pisania programu odczyt z klawiatury chcemy traktować jak liczbę, trzeba poinformować o tym interpreter. W tym celu stosuje się **funkcję int**, która przekształca napis na liczbę całkowitą, np. `int(input("Podaj liczbę"))`.

Funkcja int

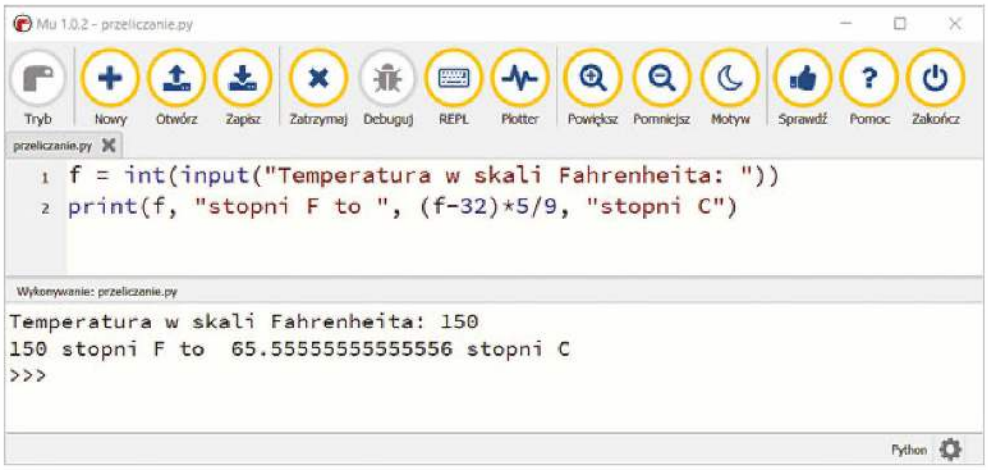
Warto wiedzieć

W języku Scratch operację wczytania liczby podanej z klawiatury do zmiennej realizuje się z użyciem bloków **zapytaj** i **ustaw**:



Wiersz 2 to kod instrukcji, która wyświetla na ekranie wartość wyrażenia arytmetycznego `(f-32)*5/9`. Wyrażenie to dla wartości temperatury podanej w skali Fahrenheita zwraca wartość w skali Celsjusza. Zauważ, że podstawowe operatory arytmetyczne (dodawania, odejmowania, mnożenia i dzielenia) są zapisywane w języku Python z użyciem znaków znanych np. z arkusza kalkulacyjnego (+, –, ∗, /).

Okno programu *Przeliczanie temperatury* przedstawiono na rysunku 10.7.



Rys. 10.7. Wywołanie programu *Przeliczanie temperatury*

Warto wiedzieć

Gdy wartość wyrażenia `(f-32)*5/9` nie jest liczbą całkowitą, to wynik jest wyświetlany łącznie z cyframi rozwinięcia dziesiętnego, np. dla `f = 33` będzie to: `0.5555555555555556`. Jeśli chcemy ograniczyć wyświetlanie wyniku do części całkowitej, należy użyć funkcji `int`, która przekształca napis na liczbę całkowitą `int`, czyli zapisać `int((f-32)*5/9)`.

Ćwiczenie 7

W pliku *przeliczanie.py* zapisz kod źródłowy programu *Przeliczanie temperatury*, a następnie wykonaj poniższe czynności.

a. Uruchom program i sprawdź jego działanie dla różnych danych.
b. Stwórz program, który będzie przeliczał temperaturę ze skali Celsjusza na skalę Fahrenheita. Zapisz kod źródłowy w pliku *przeliczanie2.py*, uruchom go i sprawdź jego działanie dla różnych danych wejściowych.

Program *Logowanie* – instrukcja warunkowa i operatory logiczne

Poprzedni program zawsze wykonywał wiersz po wierszu wszystkie instrukcje zapisane w kodzie źródłowym – przeliczał temperaturę i zwracał jej wartość na ekranie. Napiszemy teraz program, który będzie działał wariantowo – w zależności od wprowadzonych danych wybierze jedną z dwóch ścieżek: wykona następną instrukcję kodu albo ją pominie.

Załóżmy, że mamy do rozwiązania następujący problem: należy utworzyć symulator logowania do konta w serwisie internetowym. W zależności od tego, czy wprowadzone dane są poprawne, symulator będzie zwracał odpowiedni komunikat. Potrzebna więc będzie konstrukcja programistyczna, która pozwoli uzależnić działanie programu od spełnienia określonych warunków.

Taką konstrukcją jest **instrukcja warunkowa** (warunkowe rozgałęzienie): ◀ Instrukcja warunkowa

```
if warunek logiczny:
    instrukcja na tak
else:
    instrukcja na nie
```

Jeśli warunek logiczny (nazywany również wyrażeniem testowym) ma **wartość logiczną** prawda, to wykonywana jest instrukcja na tak, w przeciwnym razie – instrukcja na nie. Część **else**, oznaczająca „w przeciwnym przypadku", może nie występować w kodzie. Oto kod źródłowy programu *Logowanie*:

Logika matematyczna i operatory logiczne, s. 252

◀ Kod źródłowy programu *Logowanie*

```
1. login_baza = "Robik"
2. haslo_baza = "Gwi@zdA#99"
3. login = input("Login: ")
4. haslo = input("Hasło: ")
5. if login == login_baza and haslo == haslo_baza:
6.     print("Logowanie prawidłowe")
7. else:
8.     print("Błąd logowania")
```

Warto wiedzieć

W algorytmie z warunkami może wystąpić kilka alternatywnych ciągów działań. Wybór jednego z nich następuje w zależności od spełnienia określonych warunków.

W wierszach 1 i 2 zmiennym login_baza oraz haslo_baza przypisano wartości, które będą stanowiły uproszczony wzorzec danych przechowywanych na serwerze serwisu internetowego.

W wierszu 3 pojawia się instrukcja input. Wyświetli ona na ekranie napis „Login: " i pozwoli na odczytanie ciągu znaków z klawiatury. Odczytana wartość będzie przypisana zmiennej login. Analogicznie działa instrukcja w wierszu 4, ale dotyczy ona hasła (nie będziemy zamieniać znaków hasła na gwiazdki).

Wiersze 5–8 to przykład użycia instrukcji warunkowej **if**, która sprawdza poprawność podanych loginu i hasła. W zależności od wartości logicznej warunku wyświetlany jest napis „Logowanie prawidłowe" lub „Błąd logowania".

Dobra rada

W kodzie zapisanym w języku Python stosuj wcięcia, które są wielokrotnościami czterech spacji.

Zwróć uwagę na to, że po słowach `if` i `else` występują dwukropki, a kody w wierszach 6 i 8 są odsunięte od marginesu o tę samą liczbę spacji (w tym wypadku cztery spacje). W języku Python bardzo istotne jest, aby instrukcje (lub bloki instrukcji) w wierszach odpowiednio po `if` i po `else` były wcięte o tyle samo spacji.

Zauważ, że warunek logiczny występujący po słowie kluczowym `if` jest wyrażeniem złożonym, zwanym przez matematyków **koniunkcją** (dwa wyrażenia połączone spójnikiem „i"), zapisywanym w języku Python za pomocą operatora `and`. Wykonywane są dwa porównania typu „czy zachodzi równość?" (z użyciem operatora ==): porównanie pierwszego tekstu wprowadzonego z klawiatury z loginem zapisanym w bazie oraz porównanie drugiego tekstu wprowadzonego z klawiatury z hasłem zapisanym w bazie. Zgodnie z logiką schematu logowania (rys. 10.1, s. 173) oczekuje się, że oba porównania zakończą się sukcesem (przyjmą logiczną wartość prawda). Okno programu *Logowanie* przedstawiono na rysunku 10.8.

Koniunkcja

Warto wiedzieć

W języku Python, podobnie jak w C++, podczas wyznaczania wartości logicznej wyrażeń złożonych stosuje się tzw. leniwe wartościowanie (ang. *lazy evaluation*), aby uniknąć zbędnych operacji. Na przykład w koniunkcji nie wykonuje się drugiego porównania, jeśli pierwsze nie zakończyło się sukcesem.



Rys. 10.8. Wywołanie programu *Logowanie*

Ćwiczenie 8

Utwórz plik *logowanie.py* i zapisz w nim kod źródłowy programu *Logowanie*. Następnie uruchom program kilkukrotnie, sprawdzając różne przypadki (dla pary: login i hasło).

W pracy ze zmiennymi tekstowymi często używa się również innych **operatorów**:

- operator porównań != stosuje się wtedy, kiedy chcemy sprawdzić, czy dwie wartości są różne, np. wartość logiczna porównania `"kot" != "pies"` będzie prawdą,
- operator logiczny `or` stosuje się do zapisania alternatywy dwóch wyrażeń, np. `a < 5 or a > 10`.

Logika matematyczna i operatory logiczne, s. 252–254

Dobra rada

W języku Python możesz wstawiać odstępy między operatorami, ale ich brak nie wpływa na poprawność kodu.

Ćwiczenie 9

Zmodyfikuj program *Logowanie* tak, aby wykorzystać w nim operatory != oraz `or`. Kod źródłowy zapisz w pliku *logowanie2.py* i sprawdź działanie programu.

Program *Warunek trójkąta* – zamiana miejscami dwóch wartości

Załóżmy, że mamy do rozwiązania następujący problem: chcemy rozstrzygnąć, czy z trzech odcinków o podanych długościach można zbudować trójkąt.

Specyfikacja

Dane: trzy różne liczby całkowite dodatnie a, b, c.

Wynik: komunikat „TAK”, jeśli z odcinków o długościach a, b, c można zbudować trójkąt, lub „NIE” – w przeciwnym przypadku.

Warto wiedzieć

Sprawdzanie warunku istnienia trójkąta wydaje się banalne w przypadku małych liczb, ale jest o wiele trudniejsze np. dla liczb 299 654, 589 522, 289 877. Opracowanie algorytmu rozwiązującego ten problem pozwoli napisać program, który w krótkim czasie wykona milion takich rozstrzygnięć.

Z lekcji matematyki możesz pamiętać, że tzw. **warunek istnienia trójkąta** mówi o tym, że z odcinków o długościach a, b, c można zbudować trójkąt tylko wtedy, gdy najdłuższy odcinek jest krótszy od sumy dwóch pozostałych.

Warunek istnienia trójkąta

Aby komputer mógł sprawdzać warunek trójkąta, trzeba opracować dla niego algorytm postępowania: liczby a, b, c będą reprezentowane przez zmienne a, b, c. Program komputerowy wybierze największą wartość, zamieni miejscami tę wartość z wartością zmiennej c (o ile to konieczne) i znajdzie rozwiązanie problemu przez sprawdzenie wartości logicznej wyrażenia `a + b > c`.

Algorytm wybierania największej z trzech liczb a, b i c może mieć następującą postać:

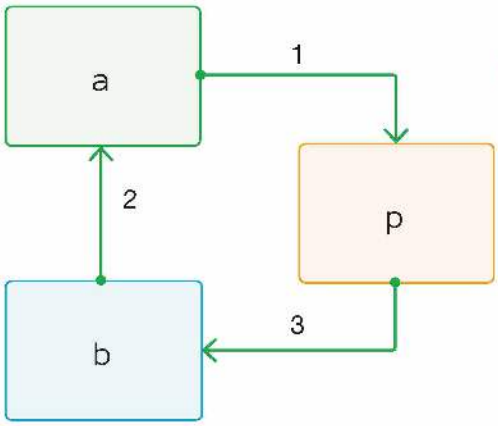
Algorytm wybierania największej z trzech liczb

1. Sprawdź, czy a > b i a > c. Jeśli tak, to a jest największą liczbą, więc zamień miejscami wartości zmiennych a i c i przejdź do kroku 3.
2. Sprawdź, czy b > c. Jeśli tak, to b jest największą liczbą, więc zamień miejscami wartości zmiennych b i c.
3. Sprawdź, czy a + b > c. Jeśli tak, to wypisz komunikat „TAK”. W przeciwnym wypadku wypisz „NIE”.

Jednym z kluczowych elementów algorytmu jest **zamiana wartości dwóch zmiennych** np. a i b. Można do tego celu użyć zmiennej pomocniczej, np. p, która przechowa tymczasowo wartość jednej ze zmiennych.

Zamiana wartości zmiennych

Pomysł ten zilustrowano na rysunku 10.9. Liczby wskazują kolejność wykonywanych podstawień (przypisywania wartości zmiennym).



Rys. 10.9. Ilustracja zamiany wartości dwóch zmiennych

Warto wiedzieć

W edytorze języka Scratch instrukcja warunkowa znajduje się wśród bloków kontrolnych.



Oto kod źródłowy powyższego algorytmu w języku Python:

Kod źródłowy programu *Warunek trójkąta*

```
1.  a = int(input())
2.  b = int(input())
3.  c = int(input())
4.  if a > b and a > c:     # a jest największą liczbą
5.      p = a
6.      a = c
7.      c = p
8.  else:
9.      if b > c:     # b jest największą liczbą
10.         p = b
11.         b = c
12.         c = p
13.
14. if a + b > c:
15.     print("TAK")
16. else:
17.     print("NIE")
```

Warto wiedzieć

Pozornie łatwe do rozwiązania problemy mogą się skomplikować, kiedy dodamy do nich nowe założenia lub zwiększymy ilość danych wejściowych. Na przykład wyświetlenie mniejszej z dwóch liczb wydaje się proste, ale wybranie największej spośród 15 liczb wymaga nieco innego podejścia.

Rysunek 10.10 przedstawia wywołanie programu *Warunek trójkąta*.



Rys. 10.10. Wywołanie programu *Warunek trójkąta*

Warto wiedzieć

Zagadnienie zamiany wartości dwóch zmiennych można zobrazować następująco: mamy dwie szklanki – jedną z sokiem, drugą z wodą. Jak przelać sok do szklanki, w której jest woda, a wodę do szklanki z sokiem? Wykorzystanie trzeciej szklanki pozwoli rozwiązać ten problem.

W programie pojawiają się nowe elementy składni:

- w wierszach 9–12 instrukcja **if** pojawia się po instrukcji **else**, co oznacza, że instrukcje z wierszy 10–12 będą wykonane tylko wtedy, gdy nierówność z wiersza 9 będzie spełniona (b > c),
- w wierszach 4 i 9 wstawiono komentarze, które rozpoczynają się od znaku #, a kończą na końcu wiersza. W procesie interpretowania kodu komentarze są ignorowane.

Ćwiczenie 10

Utwórz plik *trojkat.py* i zapisz w nim kod źródłowy programu *Warunek trójkąta*. Uruchom program kilkukrotnie, aby sprawdzić jego działanie dla różnych wartości a, b i c.

Podsumowanie

- Algorytm to opis postępowania prowadzącego do rozwiązania problemu.
- Algorytmiczne rozwiązywanie problemu jest procesem, na który składają się: zrozumienie problemu, zaprojektowanie algorytmu jego rozwiązania, komputerowa realizacja algorytmu, testowanie programu.
- Zapis algorytmu z użyciem instrukcji języka programowania nazywamy kodem źródłowym lub tekstem programu.
- Funkcja `int` pozwala zamienić napis przedstawiający cyfry liczby na liczbę całkowitą. Używa się jej np. podczas odczytywania danych z klawiatury.
- Instrukcja warunkowa umożliwia wykonanie pewnych operacji w zależności od prawdziwości wyrażenia logicznego. Zapisuje się ją przy użyciu konstrukcji **`if`** oraz **`if else`**.
- Do podstawowych konstrukcji języka programowania należą: instrukcja przypisania wartości do zmiennej, instrukcje wejścia i wyjścia, wyrażenia arytmetyczne i logiczne oraz instrukcja warunkowa.

Zadania

* **1** Wiele instrukcji służących ratowaniu życia człowieka przyjmuje postać algorytmu (np. sposób użycia defibrylatora lub instrukcja ewakuacji z budynku). Podaj kilka przykładów czynności z życia codziennego, które można opisać za pomocą algorytmów. Wybierz jedną z tych czynności i zapisz w postaci listy kroków.

* **2** Podaj specyfikację dla problemu zaplatania warkocza ze s. 174.

* **3** Napisz program, który wczyta dwie różne liczby całkowite dodatnie (np. wiek twój i najstarszej osoby w twojej rodzinie) i sprawdzi, ile razy mniejsza z nich mieści się w większej liczbie.

* **4** Napisz program, który obliczy pole powierzchni trapezu dla podanych długości podstaw oraz wysokości trapezu. Dane powinny być liczbami całkowitymi. Wynik zaokrąglij do części całkowitej.

* **5** Napisz program, który będzie wyznaczał średnią prędkość pojazdu pokonującego podaną odległość w podanym czasie. Aby sprawdzić działanie programu, oblicz średnią prędkość pociągu dla wybranych przez siebie połączeń w rozkładzie jazdy. Wynik zaokrąglij do części całkowitej.

* **6** Napisz program, który wczyta dwie liczby całkowite, a następnie wypisze większą z nich wraz z komunikatem „jest większa". Jeśli liczby są równe, to program wypisze komunikat „Liczby są równe".

* **7** Napisz program, który przeliczy wielkość podaną w bajtach na bity.

* **8** Napisz program, który wartość prędkości podaną w m/s zapisze w km/h. Wartość powinna być wyznaczona z dokładnością do części całkowitych.

* **9** Napisz program, który wczyta trzy liczby całkowite i sprawdzi, czy były podane w porządku niemalejącym. Program powinien zwrócić odpowiedź „TAK" lub „NIE".

** **10** Napisz program, który będzie przeliczał złotówki na dolary (według ustalonego kursu). Określ specyfikację problemu.

** **11** Napisz program, który wczyta trzy wyrazy i zwróci informację o liczbie wystąpień każdego z nich wśród wyrazów wczytanych. Na przykład po wczytaniu wyrazów „pies kot pies" na ekranie pojawi się informacja: „pies 2 kot 1".

** **12** Na lekcjach geometrii wykorzystuje się cyrkiel i linijkę do tworzenia konstrukcji geometrycznych. Zapisz listę kroków pozwalającą:

a. zbudować trójkąt równoboczny o zadanej długości a,

b. wyznaczyć środek odcinka łączącego punkty A i B.

Operacje typu „ustaw nóżkę cyrkla...", „zakreśl okrąg...", „poprowadź prostą...", „zaznacz punkt przecięcia..." uznaj za elementarne, niewymagające rozkładania na czynności prostsze.

** **13** Napisz program, który dla podanego wzrostu (w centymetrach) i wagi (w kilogramach) obliczy wskaźnik masy ciała (BMI).
Uwaga: Przyjmij, że użytkownik wprowadza liczby całkowite, bez części ułamkowej.

*** **14** Napisz program, który zapyta użytkownika o bieżący rok, miesiąc i dzień miesiąca oraz o datę urodzenia, a w odpowiedzi zwróci informację, czy użytkownik jest pełnoletni. Jeśli nie jest, program poda, za ile miesięcy osiągnie on pełnoletniość.

*** **15** Napisz program, który po wczytaniu trzech liczb całkowitych wpisanych z klawiatury wyświetli je w porządku niemalejącym. Program nie powinien wykonywać operacji zamiany liczb miejscami.

*** **16** Napisz program, który po wczytaniu liczby trzycyfrowej wyświetli jej cyfry w odwróconym porządku.
Uwaga: Wczytywana liczba nie powinna być podzielna przez 10.

*** **17** Napisz program, który dla liczb a, b, c i d obliczy i wyświetli sumę wartości bezwzględnych różnic: $|a-b|+|b-c|+|c-d|$. Program powinien się posługiwać wyłącznie instrukcjami `if` oraz dodawaniem i odejmowaniem.

11. Wyszukujemy i sumujemy

Do rozwiązania wielu problemów przydają się algorytmy, w których wielokrotnie trzeba powtarzać szereg obliczeń. Takie obliczenia, zwane cyklicznymi, potrzebne są np. w grafice komputerowej do uzyskania efektu cienia w scenie 3D. Komputer o mocy obliczeniowej liczonej w miliardach operacji na sekundę jest w stanie realizować tego typu zadania znacznie szybciej niż człowiek. Oprócz szybkości komputery mają jeszcze jedną zaletę: nie mylą się podczas wykonywania żmudnych, powtarzalnych czynności. Dlatego w tym temacie nauczysz się zapisywać w języku Python proste algorytmy obliczeń cyklicznych z wykorzystaniem pętli.

Cele lekcji

- Zrozumiesz, na czym polega iteracyjne rozwiązywanie problemów.
- Nauczysz się tworzyć algorytmy zawierające instrukcje powtarzania.
- Napiszesz w języku Python programy z użyciem pętli.

Do tej pory każdy nasz program zakładał jednokrotne wykonanie instrukcji zapisanych w tekście programu. Napiszemy teraz programy, które będą wielokrotnie wykonywały ten sam fragment kodu. Będą one komputerowymi realizacjami algorytmów nazywanych iteracyjnymi, czyli takich, w których aby uzyskać określony efekt, powtarza się jakąś czynność wielokrotnie.

Warto wiedzieć

Słowo „iteracyjny" pochodzi od łacińskiego czasownika *iterare*, który znaczy „powtarzać".

11.1. Szukamy największej liczby. Pętla `while`

Rozważmy następujący problem: trener skoku w dal mierzy podczas treningu długości kolejnych skoków i wpisuje wyniki na bieżąco do programu komputerowego. Chcemy stworzyć program, który na zakończenie treningu wyświetli długość najdłuższego skoku. Zakładamy, że wyniki podawane są w metrach z dokładnością do 0,01 m, czyli 1 cm.

Nie wiemy, ile skoków lekkoatleta wykona podczas treningu. Nawet jeśli trener zaplanuje jakąś ich liczbę, to i tak niektóre skoki będą prawdopodobnie spalone (nieważne).



Rys. 11.1. Skok w dal

Warto wiedzieć

Określenie jednostek, w których wyrażone są różne wielkości, jest dla programisty bardzo ważne, ponieważ musi on odpowiednio wybrać typ danych. Na przykład jeśli dokładność pomiaru skoku w dal wynosi 1 cm, to wyniki podawane w centymetrach będą liczbami całkowitymi. Jeśli jednak wyniki podajemy w metrach, mogą być ułamkami.

Dobra rada

Ciąg potraktuj jako układ liczb, w którym występują one w określonej kolejności.

Warto wiedzieć

W żargonie programistów specjalną wartość (np. 0), która wskazuje koniec ciągu wczytywanych do programu wartości, nazywa się wartownikiem (ang. *sentinel*). Wartość ta musi być unikatowa, aby nie można było jej pomylić z wartościami podanymi przez użytkownika programu.

Program komputerowy trzeba więc napisać tak, aby trener mógł poinformować komputer o tym, że trening został zakończony.

Jak to zrobić? Jednym ze sposobów może być wpisanie przez trenera liczby 0, która zakończy ciąg podawanych liczb. Wtedy program komputerowy powinien wypisać długość najdłuższego skoku.

Specyfikacja

Dane: ciąg liczb nieujemnych, zapisanych z dokładnością do 0,01, zakończony liczbą 0.

Wynik: największa z podanych liczb.

Aby wykonać to zadanie, program musi porównać długości skoków. Komputer nie potrafi jednak jednocześnie porównywać wielu liczb, nawet gdyby miał je wszystkie zapisane w pamięci. Pracuje on sekwencyjnie i nie może jak człowiek objąć wzrokiem kilku liczb, aby wskazać największą. W jednej operacji może brać pod uwagę tylko dwie liczby. Pozostałe są niedostępne.

Na poniższym schemacie zaprezentowano, jaką strategię wyboru największej spośród czterech podanych liczb stosuje człowiek, a jaką – komputer.

Jak widzi komputer?

Zadanie:
Która z podanych liczb jest największa?

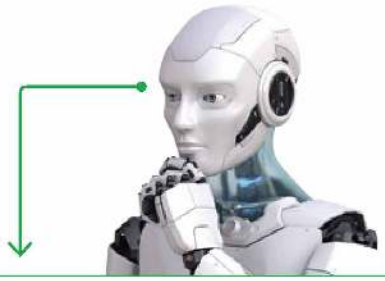
6,40	6,53	6,34	6,55



Człowiek

Przy małej liczbie danych człowiekowi wystarczy jedno spojrzenie, aby wskazać największą liczbę:

6,40	6,53	6,34	**6,55**



Komputer

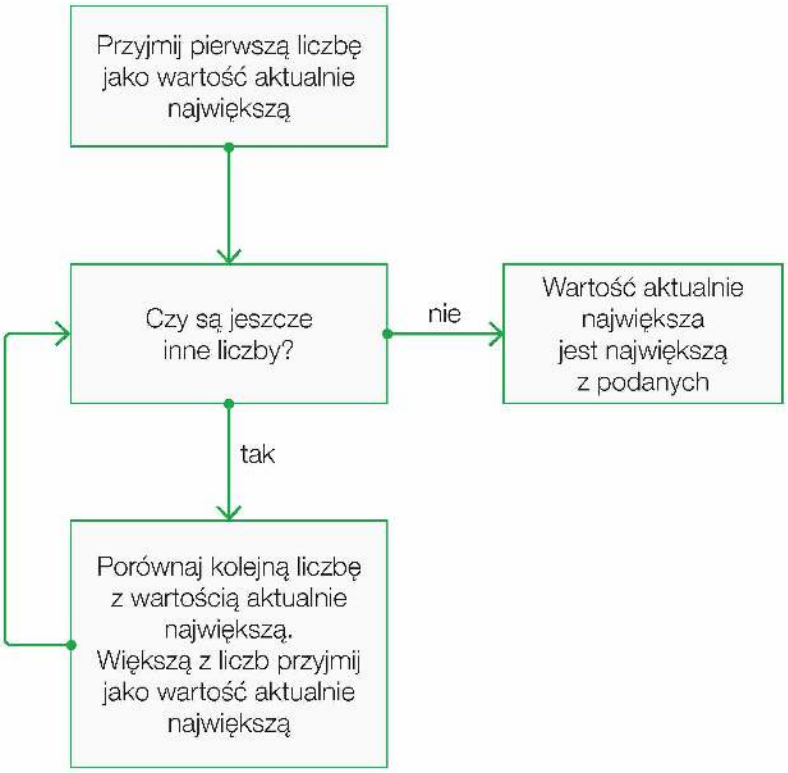
Komputer, nawet przy małej liczbie danych, etapami porównuje kolejne pary liczb i wskazuje większą liczbę z pary:

Etap 1	6,40	**6,53**		
Etap 2		6,53	6,34	
Etap 3		6,53		**6,55**

Aby program mógł wyznaczyć najdłuższy z podanych skoków, będziemy porównywać ich długości parami: porównamy pierwszy skok z drugim, dłuższy z nich porównamy z trzecim skokiem, dłuższy z nich z czwartym itd. Operacje te program będzie wykonywał do momentu, aż użytkownik wpisze liczbę 0.

Dzięki takiemu rozwiązaniu program będzie musiał zapamiętywać tylko dwie wartości: aktualnie podaną przez użytkownika i największą z dotychczas podanych. W naszej sytuacji pamiętanie pozostałych wartości nie jest potrzebne.

Poniższy diagram (rys. 11.2) pokazuje, na czym polega cykliczne wykonywanie tej samej operacji na kolejnych danych. Zauważ, że graficznie ma ono postać zbliżoną do pętli.



Rys. 11.2. Diagram przedstawiający algorytm wyszukiwania największej wartości

Warto wiedzieć

Procesor porównujący ze sobą dwie liczby używa sprzętowej instrukcji porównywania. Jej działanie polega na sprawdzeniu, czy różnica porównywanych liczb jest dodatnia, ujemna czy równa 0.

Lista kroków algorytmu wyznaczania największej liczby z ciągu liczb mogłaby wyglądać następująco:

1. Jako wartość aktualnie największą (maksimum) przyjmij 0 i wczytaj pierwszą liczbę.
2. Dopóki ostatnia wczytana liczba jest różna od 0, wykonuj krok 3 i krok 4.
   3. Jeśli ostatnia wczytana liczba jest większa od maksimum, to podstaw ją za maksimum.
   4. Wczytaj kolejną liczbę z klawiatury.
5. Wypisz maksimum na ekranie.

Przeanalizujemy teraz działanie algorytmu na przykładowych danych. Wygodnie będzie zrobić to w tabeli.

Dobra rada

Jeżeli trudno ci zrozumieć, jak działa algorytm, możesz utworzyć tabelę zawierającą pośrednie wartości zmiennych i warunków logicznych.

Wartość aktualnie największa (maksimum)	Wczytana liczba	Czy ostatnia wczytana liczba jest różna od 0?	Czy ostatnia wczytana liczba jest większa od aktualnego maksimum?
0	6,20	tak	tak
6,20	6,23	tak	tak
6,23	6,34	tak	tak
6,34	6,30	tak	nie
6,34	6,41	tak	tak
6,41	6,14	tak	nie
6,41	6,34	tak	nie
6,41	6,27	tak	nie
6,41	0	nie	–

Tabela 11.1. Działanie algorytmu wyznaczania największej liczby

Tabelę uzupełniamy wierszami od lewej do prawej, według kolejnych kroków algorytmu. Kolorem zielonym zaznaczono pewien etap analizy działania algorytmu: wczytano w tym momencie liczbę 6,41. Kolejnymi operacjami będą: sprawdzenie, czy liczba 6,41 jest różna od 0 (krok 2), a następnie – czy jest większa od ostatniej największej wartości równej 6,34 (krok 3). Ponieważ oba warunki są spełnione, liczba 6,41 staje się aktualnie największą. W kolejnych iteracjach podawane są liczby od niej mniejsze, dlatego w momencie podania liczby 0 liczba aktualnie największa staje się po prostu największą spośród wszystkich podanych liczb.

A to ciekawe

Muzyczne pętle

Pętlą lub loopem w muzyce elektronicznej nazywa się odpowiednio przygotowany fragment utworu (sampel), który wielokrotnie powtarza się w innym utworze. Jedną z najczęściej wykorzystywanych pętli perkusyjnych w muzyce rozrywkowej jest fragment z utworu Jamesa Browna *Funky Drummer*. Według serwisu whosampled.com wykorzystano go w ponad 1500 utworach innych artystów.



Ćwiczenie 1

Zastanówcie się w grupie, czy jeśli na liście kroków ze s. 191 zamienimy miejscami kroki 3 i 4, lista nadal będzie przedstawiać poprawny algorytm. W jaki sposób można to sprawdzić?

Zauważ, że program wyznaczający największą liczbę z ciągu powinien cyklicznie:

1. sprawdzać, czy liczba wprowadzona przez użytkownika jest liczbą 0, której podanie kończy działanie programu,
2. sprawdzać, czy wprowadzona liczba jest większa od aktualnie największej,
3. wczytywać kolejną wprowadzoną liczbę.

W języku Python operacje cykliczne realizuje się w specjalnych **instrukcjach iteracyjnych** (**powtarzania**), potocznie nazywanych **pętlami** (ang. *loop*). W językach programowania stosuje się różne rodzaje instrukcji iteracyjnych. W poniższym przykładzie wykorzystamy jeden z rodzajów pętli – **pętlę `while`**, która ma następującą budowę:

Instrukcja iteracyjna (instrukcja powtarzania, pętla)

Pętla `while`

```
while warunek:
    instrukcje
```

Ta pętla pozwala wykonywać dane instrukcje, dopóki warunek jest spełniony, tzn. przyjmuje **wartość logiczną** prawda. Wartość logiczną wyrażenia program sprawdza przed każdym wykonaniem instrukcji, może więc: nie wykonać danej instrukcji, wykonać ją raz albo wiele razy.

Logika matematyczna i operatory logiczne, s. 252

Kod źródłowy programu *Najdłuższy skok* może wyglądać następująco:

Kod źródłowy programu *Najdłuższy skok*

```
1.  maks = 0
2.
3.  print("Podaj kolejne długości skoków:")
4.  a = float(input())
5.
6.  while a != 0:
7.      if a > maks:
8.          maks = a
9.      a = float(input())
10.
11. print("\nNajdłuższy skok:", maks, "m")
```

Warto wiedzieć

W języku Python zmienna musi mieć jawnie przypisaną wartość, zanim będzie można jej użyć, np. w warunku pętli. Jeśli programista nie nada wartości początkowej zmiennej, to po uruchomieniu programu jego działanie zostanie przerwane i pojawi się komunikat o błędzie.

W wierszach 4 i 9, podczas wczytywania długości skoków, zastosowaliśmy **funkcję `float`**, która pozwala liczby wpisane z klawiatury przechować jako liczby wymierne. W zmiennej `a` będzie przechowywana liczba jako ostatnia wczytana z klawiatury, a w zmiennej `maks` – aktualnie największa z wczytanych liczb. W wierszu 1 przypisujemy tej zmiennej wartość 0.

Funkcja `float`

W wierszach 6–9 pojawia się pętla **while**, która najpierw sprawdza, czy wartość zmiennej a jest różna od 0. Jeżeli tak, to program wykonuje grupę instrukcji zapisanych w wierszach 7–9. Są one w kodzie programu odsunięte od marginesu tak, aby interpreter mógł rozpoznać, że należą do pętli. Instrukcje wykonywane są tyle razy, ile razy użytkownik podał liczbę różną od zera.

W wierszu 7 sprawdzamy, czy podana przez użytkownika liczba jest większa od wartości zmiennej maks. Jeśli tak, to ta liczba jest przypisywana do zmiennej maks. Po zakończeniu wprowadzania liczb (czyli po wpisaniu przez użytkownika liczby 0) aktualnie największa liczba staje się liczbą największą dla tego zestawu danych. Zauważ, że instrukcja przypisania wartości a do zmiennej maks (wiersz 8) jest zapisana z dodatkowym odstępem po to, aby poinformować interpreter, że wiersze 7 i 8 stanowią zamkniętą całość. W linii 11 użyto zapisu \n, który odpowiada za przeniesienie kursora do kolejnego wiersza.

Okno programu *Najdłuższy skok* przedstawiono na rysunku 11.3. Zwróć uwagę, że w programie liczby ułamkowe podaje się z kropką zamiast przecinka.

Warto wiedzieć

Od lewego ukośnika zapisuje się tzw. stałe znakowe znaków niegraficznych kodu ASCII (np. \n dla nowego wiersza, \t dla tabulacji poziomej) lub znaków specjalnych (np. \" dla znaku ", gdy chcemy go wyświetlić na ekranie).

Dobra rada

Jeżeli używasz w pętli **while** warunku porównującego wartości typu **float** (liczby wymierne), staraj się stosować operatory porównania <, > i != zamiast <=, >= oraz ==. Pozwoli to uniknąć niektórych błędów wynikających z dokładności przybliżeń wartości zmiennych.



Rys. 11.3. Wywołanie programu *Najdłuższy skok*

Warto wiedzieć

Zapisywanie liczb ułamkowych z kropką, a nie z przecinkiem, jest zwyczajem anglosaskim. To dlatego na klawiaturze numerycznej znajduje się właśnie znak kropki.

Ćwiczenie 2

Utwórz plik *najdluzszy_skok.py* i zapisz w nim kod źródłowy programu *Najdłuższy skok*. Sprawdź kod i uruchom program kilkukrotnie dla różnych danych wpisanych z klawiatury.

11.2. Sumujemy liczby. Pętla for

Pętla for

Drugą z instrukcji iteracyjnych w języku Python jest **pętla for**. W odróżnieniu od instrukcji **while**, stosuje się ją w przypadkach, gdy można z góry określić liczbę cykli, czyli wykonań pętli, nawet jeśli dokładna liczba powtórzeń zależy od danych podanych przez użytkownika. Pętlę **for** wykorzystamy w kolejnym programie.

Problem, który sobie postawimy, to: wyznaczyć sumę n kolejnych liczb parzystych.

Oto specyfikacja problemu algorytmicznego:

Specyfikacja

Dane: liczba naturalna n.

Wynik: suma n kolejnych liczb parzystych.

Aby rozwiązać problem od strony algorytmicznej, posłużymy się prostą matematyką: będziemy wielokrotnie dodawać kolejne składniki (kolejne liczby parzyste) do sumy częściowej, aż liczba dodanych składników będzie równa n. Zaczniemy od sumy równej 0 i składnika 2.

Sposobem na wielokrotne wykonanie instrukcji zwiększania sumy będzie instrukcja pętli. Do zapamiętania, ile razy została wykonana pętla, użyjemy **zmiennej sterującej** pętlą **for**, nazywanej też zmienną kontrolną.

Zmienna sterująca

Składnia instrukcji **for** ma następującą postać:

```
for zmienna in lista_wartości:
    instrukcje
```

Po słowie **for** zapisuje się nazwę zmiennej sterującej pętlą (licznik), natomiast po słowie **in** umieszcza się listę wartości zmiennej sterującej, dla których instrukcja pętli będzie wykonywana.

Jeśli powtarzana ma być instrukcja złożona, to kolejne tworzące ją instrukcje zapisuje się w kolejnych wierszach, wcięte względem słowa **for**.

Aby podać listę wartości w pętli **for**, najprościej wypisać te wartości w nawiasach kwadratowych, rozdzielone przecinkami. Lista może zawierać wartości różnych typów, np.:

```
1. for i in [5, 10, 15]:
2.     print(i)
3.
4. for k in ["pięć", 10, "piętnaście"]:
5.     print(k)
```

Warto wiedzieć

W języku Scratch do tworzenia pętli służą bloki **powtarzaj aż** oraz **powtórz**, znajdujące się wśród bloków kontrolnych.





Do utworzenia w pętli **for** listy wartości, która często jest bardzo długa, służy wbudowana **funkcja** `range`. Umożliwia ona generowanie ciągów liczb całkowitych.

Funkcja `range`

Funkcja `range` może być wywołana w następujących wariantach:

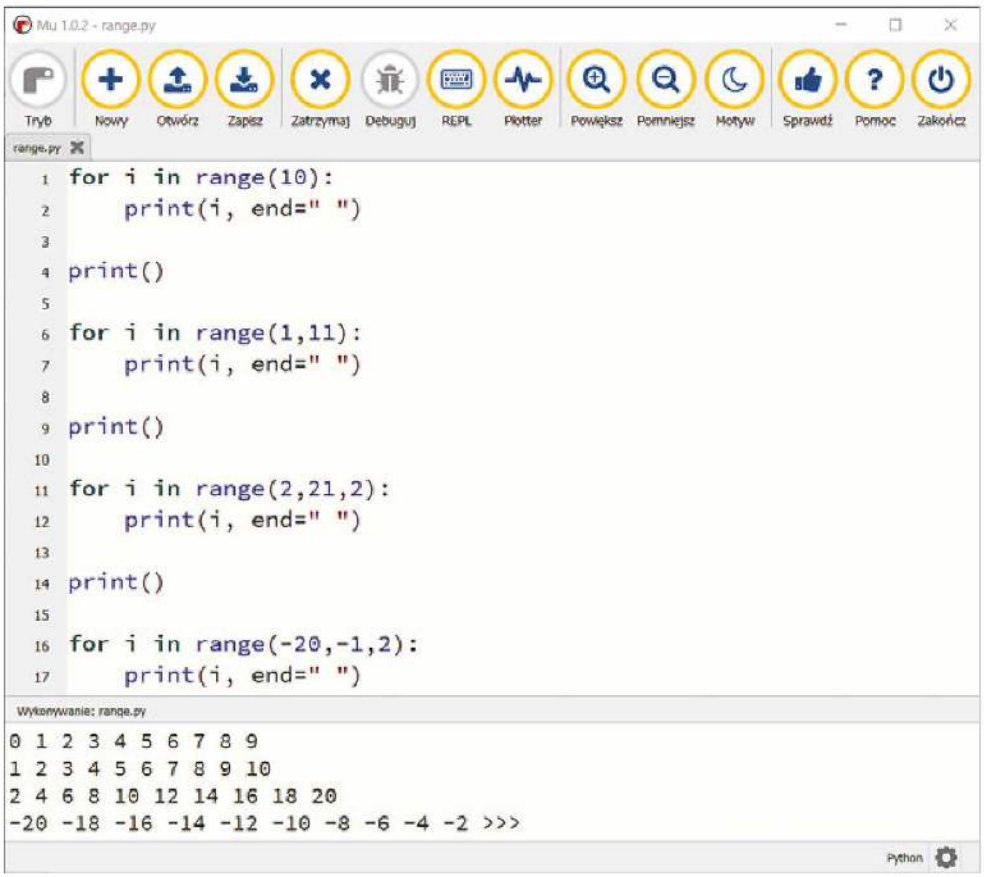
- `range(n)` – wówczas funkcja wygeneruje listę liczb od 0 do $n - 1$,
- `range(k, m)` – wówczas funkcja wygeneruje listę kolejnych liczb od k do $m - 1$,
- `range(k, m, krok)` – wówczas funkcja wygeneruje listę liczb od k do $m - 1$, które będą się zmieniały o ustaloną wartość kroku.

Dobra rada

Aby sprawdzić, czy twój kod jest poprawny oraz sformatowany zgodnie z zasadami (np. ma zachowane odstępy), skorzystaj z narzędzia **Sprawdź** w edytorze Mu.

Rysunek 11.4 przedstawia przykład wykorzystania funkcji `range` w pętli `for` do generowania różnych ciągów liczbowych.

Zastosowano tu cztery pętle. Pierwsza wypisuje kolejne liczby od 0 do 9, druga – liczby od 1 do 10, trzecia – liczby parzyste od 2 do 20, a czwarta – parzyste liczby ujemne od –20 do –2.



Rys. 11.4. Różne warianty wywołania funkcji `range` w pętli `for`

Warto wiedzieć

Parametr `end` funkcji `print` odpowiada za wstawienie znaku po wypisaniu na ekranie napisu lub wartości zmiennej. Domyślnie parametr `end` ma wartość `"\n"`, czyli wartość znaku przeniesienia do kolejnego wiersza.

Kod źródłowy programu *Suma*

Kod źródłowy programu sumującego n kolejnych liczb parzystych może wyglądać następująco:

```
1.  n = int(input("Podaj n:"))
2.
3.  suma = 0
4.  skladnik = 2
5.
6.  for i in range(1, n+1):
7.      suma = suma + skladnik
8.      skladnik = skladnik+2
9.
10. print("Suma", n, end="")
11. print(" kolejnych liczb parzystych wynosi:", suma)
```

Dobra rada

Zapis `range(0, n)` możesz uprościć do postaci `range(n)`. Efekt działania funkcji będzie taki sam.

Zwróć uwagę na wiersze 6–8, czyli na zapis instrukcji `for`. Podczas wykonywania pętli najpierw jest ustalana wartość początkowa zmiennej sterującej `i`.

Wartość zmiennej sterującej powinna się zwiększać o 1 w każdym cyklu pętli do momentu osiągnięcia przez nią wartości zmiennej n.

Zmienna `i` przyjmuje wartości z listy liczb całkowitych zwróconej przez funkcję `range`. Parametry tej funkcji określają wartość początkową i końcową listy, przy czym wartość końcowa nie należy do listy. Kod `range(1, n+1)` tworzy listę wartości: 1, 2, ..., *n*.

Każde powtórzenie w pętli powoduje zwiększenie wartości zmiennej `suma` oraz ustalenie kolejnej wartości zmiennej `skladnik` (jako kolejnej liczby parzystej).

W linii 1 funkcja `input` zostaje wykorzystana jako tekst zachęty do wprowadzenia danych. W naszym przykładzie znakiem zachęty jest napis „Podaj n: ". Podany przez użytkownika napis jest zamieniany przez funkcję `int` na liczbę całkowitą i tak zapisywany w zmiennej. Dzięki temu program może wykonać obliczenia.

W wierszu 10 występuje dodatkowy parametr `end` funkcji `print`, którego zadaniem jest zamiana standardowego przeniesienia do kolejnej linii na pusty znak.

Okno programu *Suma* przedstawiono na rysunku 11.5.

```
Wykonywanie: suma.py
Podaj n: 25
Suma 25 kolejnych liczb parzystych wynosi: 650
>>>
                                                    Python
```

Rys. 11.5. Wywołanie programu *Suma*

Warto wiedzieć

Suma kolejnych liczb nieparzystych jest kwadratem liczby składników:

$1 + 3 = 2^2$,

$1 + 3 + 5 = 3^2$,

$1 + 3 + 5 + 7 = 4^2$

itd.

Ćwiczenie 3

a. W pliku *suma.py* zapisz kod źródłowy programu *Suma* i sprawdź działanie programu dla różnych wartości *n*.

b. Zmodyfikuj program *Suma* tak, aby wyznaczał sumę określonej liczby kolejnych liczb nieparzystych (począwszy od 1). Program zapisz pod nazwą *suma_nieparzystych.py*.

c. Korzystając z programu *Suma*, utwórz program, który poda sumę 100, 1000 oraz 10 000 kolejnych wielokrotności liczby 11.

Zapamiętaj

Pętla `while` jest wykonywana, dopóki jej warunek logiczny jest spełniony. Liczba powtórzeń nie jest określona z góry przez programistę.

Pętla `for` jest wykonywana określoną liczbę razy, ustaloną przez programistę lub przez użytkownika już po uruchomieniu programu.

11.3. Wyznaczamy noty sędziowskie

W ostatniej części tematu zmierzymy się z kolejną sytuacją problemową. Wykorzystamy umiejętności zdobyte wcześniej do opracowania odpowiedniego algorytmu i napiszemy program o bardziej rozbudowanej strukturze.

Rozważmy następujący problem: w konkursie skoków narciarskich chcemy wyznaczyć ocenę za styl, którą otrzyma dany zawodnik. Aby to zrobić, spośród pięciu ocen przyznanych przez pięciu sędziów należy odrzucić dwie skrajne noty: najwyższą i najniższą, a pozostałe zsumować.

Specyfikacja

Dane: pięć liczb z przedziału $[0; 20]$ podanych z dokładnością do 0,5.

Wynik: największa i najmniejsza liczba oraz suma liczb pomniejszona o liczbę najmniejszą i liczbę największą.

Warto wiedzieć

W statystyce stosuje się pojęcie średniej ucinanej. Średnia ta wyliczana jest na podstawie próbki wyników po odrzuceniu pewnej części wartości skrajnych – znacznie większych oraz znacznie mniejszych od pozostałych. W arkuszu kalkulacyjnym taką średnią wyznacza funkcja `ŚREDNIA.WEWN`.

Załóżmy, że nasz symulator wyznaczania oceny za styl będzie programem, który nie będzie zapamiętywał wszystkich pięciu liczb (not sędziowskich). Dzięki temu rozwiązanie stanie się bardziej uniwersalne: program będzie można łatwo zmodyfikować i zastosować do innego podobnego problemu, w którym stosuje się zasadę odrzucania wartości skrajnych (np. skrajnych wyników pomiarów w doświadczeniu przeprowadzanym na lekcji fizyki).

Algorytm będzie wyznaczał szukaną sumę przez odjęcie od sumy wszystkich pięciu liczb wartości najmniejszej oraz wartości największej. W przypadku gdy wartości skrajnych jest więcej, będziemy odejmować każdą tylko raz.

W zagadnieniu 11.1 analizowaliśmy przykład wyznaczania największej z liczb wprowadzonych z klawiatury. Zauważ, że tamto rozwiązanie można zaadaptować do obecnego problemu.

Oto propozycja listy kroków algorytmu:

1. Wczytaj pierwszą liczbę i przyjmij ją jako wartość minimum oraz jednocześnie wartość maksimum. Jako wartość sumy przyjmij wartość wczytanej liczby.
2. Powtórz cztery razy kroki od 3 do 6.
   3. Wczytaj kolejną liczbę.
   4. Zwiększ wartość sumy o wartość wczytanej liczby.
   5. Jeśli wczytana liczba jest mniejsza od minimum, to podstaw ją za minimum.
   6. Jeśli wczytana liczba jest większa od maksimum, to podstaw ją za maksimum.
7. Wypisz na ekranie minimum, maksimum oraz sumę pomniejszoną o minimum i maksimum.

Punktacja w skokach narciarskich

Zawodnicy startujący w zawodach skoków narciarskich otrzymują za każdy skok ocenę, która składa się z dwóch części: punktów za uzyskaną odległość oraz noty za styl. Tę sumę koryguje się w zależności od warunków pogodowych i długości rozbiegu.



NOR	19.0
SUI	~~19.5~~
SVK	19.0
AUT	19.0
ITA	~~18.5~~
WIND:	+11.1
1	144.7

Noty sędziowskie za styl od 0 do 20 punktów: skrajne noty (najwyższa i najniższa) są usuwane, a pozostałe sumowane

Korekta punktacji ze względu na siłę i kierunek wiatru oraz długość rozbiegu

Suma punktów za odległość i styl z uwzględnieniem korekty

Kod źródłowy programu może wyglądać następująco:

Kod źródłowy programu *Noty sędziowskie*

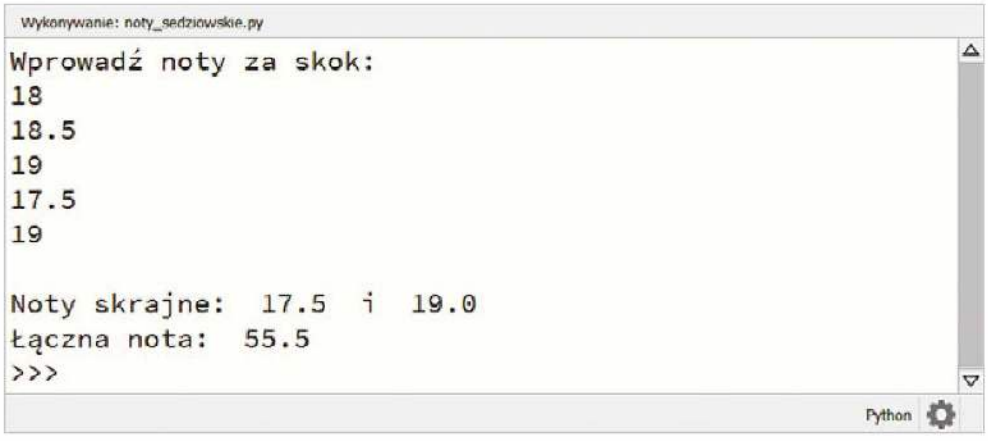
```
1.  print("Wprowadź noty za skok:")
2.  nota = float(input())
3.  mini = nota
4.  maks = nota
5.  suma = nota
6.
7.  for i in range(4):
8.      nota = float(input())
9.      suma = suma + nota
10.     if nota < mini:
11.         mini = nota
12.     if nota > maks:
13.         maks = nota
14.
15. suma = suma-mini-maks
16. print("\nNoty skrajne: ", mini, " i ", maks)
17. print("Łączna nota: ", suma)
```

Dobra rada

W programie, który wyznacza największą wartość ze zbioru liczb, jako początkową wartość zmiennej `maks` przyjmij pierwszą dostępną wartość.

Pętla **for** wykona się 4 razy i w każdym cyklu zostanie pobrana kolejna nota za skok (linia 8), a wartość zmiennej `suma` zostanie zwiększona o aktualnie podaną notę (linia 9). Oprócz tego w liniach 10–13 program sprawdza, czy podana nota ma wartość aktualnie największą lub najmniejszą.

Okno programu *Noty sędziowskie* przedstawiono na rysunku 11.6.



Rys. 11.6. Wywołanie programu *Noty sędziowskie*

Ćwiczenie 4

Przepisz kod programu *Noty sędziowskie* i sprawdź jego działanie dla różnych danych. Następnie zmodyfikuj program tak, aby obliczał sumę bez wartości skrajnych dla dowolnej liczby składników (wprowadzonych z klawiatury).

Podsumowanie

- Oprócz algorytmów liniowych i algorytmów z warunkami w programowaniu stosuje się algorytmy iteracyjne.
- Iteracyjne rozwiązywanie problemów polega na powtarzaniu jakiejś czynności skończoną liczbę razy, aż uzyska się oczekiwany efekt. Wykorzystuje się do tego instrukcje powtarzania (iteracyjne).
- W języku Python instrukcjami powtarzania, pozwalającymi wielokrotnie wykonać ten sam fragment kodu, są `while` oraz `for`.
- Pętla `while` charakteryzuje się tym, że instrukcja wykonywana w pętli jest powtarzana, dopóki pewien warunek logiczny jest spełniony.
- W pętli `for` instrukcje wykonywane w pętli są powtarzane określoną liczbę razy.
- Funkcja `range` służy do generowania ciągów liczb całkowitych. Można ją wykorzystywać do generowania listy wartości w pętli `for`.

Zadania

* **1** Napisz program, który wyznaczy najmniejszą liczbę z ciągu liczb całkowitych dodatnich podawanych z klawiatury. Przyjmij, że podanie liczby 0 kończy wprowadzanie liczb.

* **2** Napisz program wypisujący 10 kolejnych wielokrotności dodatniej liczby naturalnej wczytanej z klawiatury.

* **3** Zmodyfikuj kod źródłowy programu *Logowanie* z tematu 10 (s. 183) tak, aby program kończył działanie po trzech nieudanych próbach logowania.

* **4** Zmodyfikuj program *Najdłuższy skok* tak, aby trener mógł dodatkowo zliczać liczbę skoków nieważnych (spalonych).

* **5** Napisz program, który będzie obliczać średnią wartość (obciętą do części całkowitej) wszystkich liczb wprowadzonych z klawiatury, z pominięciem liczb najmniejszej i największej.

* **6** Przeanalizuj dokładnie działanie kodu źródłowego programu *Najdłuższy skok* na trzech konkretnych przykładach. Efekty przedstaw w tabeli w pliku tekstowym:

a	maks
4	4
2	
5	5
7	7
5	
0	

a	maks
8	
6	
2	
8	
9	
0	

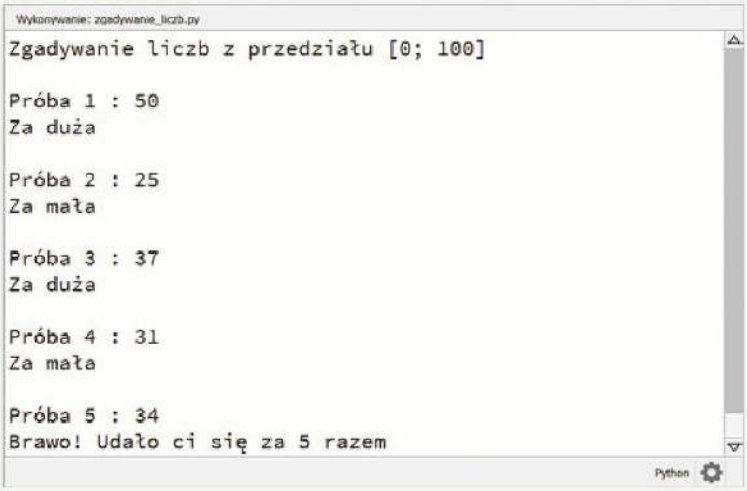
a	maks
12	
9	
12	
9	
12	
0	

** **7** Napisz program, który zliczy osobno, ile liczb ujemnych i liczb dodatnich wprowadzono z klawiatury. Przyjmij, że wczytanie liczby 0 kończy wczytywanie liczb.

** **8** Napisz program, który po wczytaniu 10 liczb wskaże te, które potraktowane jako długości boków utworzą prostokąt o największym polu.

** **9** Napisz kod programu obliczającego sumę n kolejnych liczb naturalnych, w którym nie będziesz używać zmiennej `skladnik`, a wartość składnika będzie ustalana na postawie zmiennej sterującej `i`.

** **10** Napisz program, który będzie grą w zgadywanie ustalonej liczby z przedziału od 0 do 100. Program powinien działać do momentu podania przez użytkownika właściwej liczby. Po każdej próbie odgadnięcia liczby program powinien wypisać jeden z komunikatów: „Za mała", „Za duża" lub „Brawo! Udało ci się za x razem", gdzie x oznacza liczbę prób, które wykonał użytkownik (rysunek poniżej).



** **11** Napisz program, który będzie wczytywał podane przez użytkownika liczby jedno- i dwucyfrowe całkowite i sumował je do momentu otrzymania sumy przekraczającej 123. Program powinien zwrócić informację o liczbie składników sumy i wartość sumy.

** **12** Napisz program, który wypisze pierwszych 100 liczb całkowitych dodatnich podzielnych przez:

a. 3, **b.** 7, **c.** 3 i 7.

*** **13** Zmodyfikuj program *Najdłuższy skok* tak, aby dodatkowo znajdował najdłuższą serię, w której każdy kolejny skok był dłuższy od poprzedniego, a jako wynik wyświetlał liczbę skoków w tej serii. Na przykład dla skoków o długościach 6,20; 6,30; 6,28; 6,33; 6,40; 6,30; 6,23 program powinien wyświetlić odpowiedź 3.

*** **14** Dla sześciu liczb wprowadzonych z klawiatury zawsze wystarczy 7 porównań, aby wyznaczyć najmniejszą oraz największą z nich. Jak to zrobić? Zapisz algorytm w postaci listy kroków, a następnie jego realizację w języku programowania.

*** **15** Napisz program, który pozwoli rozstrzygnąć, czy wśród n liczb całkowitych dodatnich wprowadzonych z klawiatury była taka trójka liczb x, y i z, dla których nie da się zbudować trójkąta o bokach długości x, y, z.

12. Binarny system liczbowy

Kiedy korzystasz z komputera lub smartfona, wykonujesz wiele różnych czynności, m.in. słuchasz muzyki, przetwarzasz zdjęcia, redagujesz teksty, wysyłasz e-maile i oglądasz transmisje wideo. To wszystko jest możliwe dzięki temu, że zarówno oprogramowanie, jak i różnego typu dane są zapisywane w pamięci komputera w jednolity sposób – cyfrowo, czyli jako liczby. Z tego tematu dowiesz się, jaką postać mają takie liczby.

Cele lekcji

- Dowiesz się, jak się zapisuje liczby w binarnym systemie liczbowym.
- Zrozumiesz istotę cyfrowego zapisu informacji.
- Poznasz algorytmy zamiany reprezentacji liczby między systemami dziesiętnym i dwójkowym.
- Nauczysz się korzystać ze struktury listy w języku Python.
- Zaprogramujesz algorytm konwersji liczby z systemu dziesiętnego na dwójkowy.

12.1. Liczby dziesiętne i dwójkowe

Gdy na co dzień wykorzystujemy dane liczbowe, posługujemy się zazwyczaj **systemem dziesiętnym** (**decymalnym**). Dlaczego właśnie ten system się upowszechnił? Pewnie dlatego, że pierwszymi liczydłami były palce u dłoni człowieka. Liczby zapisujemy więc za pomocą dziesięciu cyfr: 0, 1, 2, 3, 4, 5, 6, 7, 8 i 9.

System dziesiętny (decymalny)

Naturalne jest dla nas, że zapis 2350 oznacza liczbę dziesięć razy większą od 235. Wynika to z faktu, że system dziesiętny jest **systemem pozycyjnym**. To oznacza, że z pozycją cyfry (liczoną od prawej strony) wiąże się jej waga – w tym wypadku: 1, 10, 100, 1000 itd. Są to kolejne potęgi liczby 10, która dla kodu decymalnego jest **podstawą systemu liczbowego**. Każdą liczbę przedstawia się w tym systemie jako sumę wielokrotności potęg liczby 10. Na przykład liczba 235 jest sumą dwóch potęg 10^2, trzech potęg 10^1 i pięciu jedności (czyli potęg 10^0).

System pozycyjny

Podstawa systemu liczbowego

Na rysunku 12.1 pokazano zapis liczby 235 w systemie dziesiętnym.



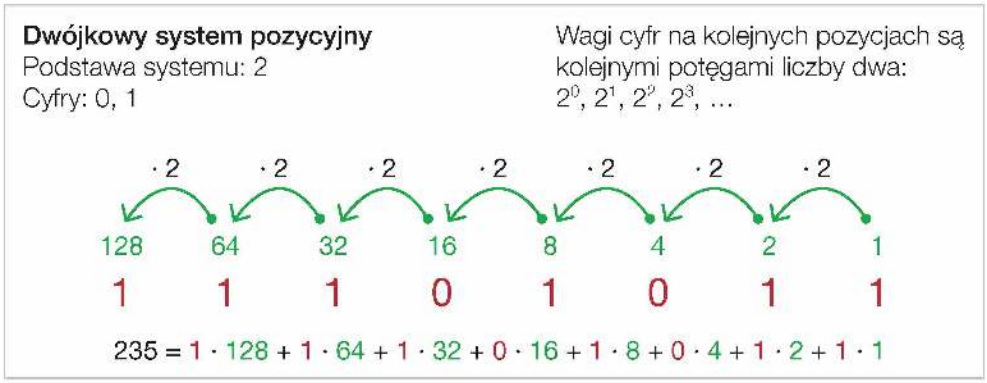
Rys. 12.1. Zapis dziesiętny liczby 235

System dwójkowy (binarny)

W technice komputerowej szczególne znaczenie ma **system dwójkowy (binarny)**, w którym występują tylko dwie cyfry: 0 i 1. Wagi cyfr na kolejnych pozycjach w zapisie liczby w systemie dwójkowym są potęgami liczby 2. Na rysunku 12.2 przedstawiono binarną postać liczby 235.

Warto wiedzieć

O liczbie przedstawionej w systemie dwójkowym mówi się w skrócie: liczba dwójkowa.



Rys. 12.2. Zapis dwójkowy liczby 235

Zauważ, że liczba 235 jest sumą 2^7, 2^6, 2^5, 2^3, 2 oraz 1 i jest – w zapisie dwójkowym – liczbą ośmiocyfrową 11101011.

Tabela 12.1 zawiera zapis binarny liczb od 0 do 15. Możemy na tej podstawie sformułować kilka **podstawowych własności liczb binarnych**:

Podstawowe własności liczb binarnych

1. Liczby nieparzyste w zapisie binarnym kończą się cyfrą 1.
2. Liczby binarne kończące się cyfrą 0 dzielą się przez 2, a więc są parzyste.
3. Aby uzyskać liczbę binarną dwa razy większą od wyjściowej, należy na końcu dopisać cyfrę 0. Oznacza to, że jeśli usuniemy ostatnią cyfrę parzystej liczby binarnej, otrzymamy połowę tej liczby.
4. Zapisy dwóch kolejnych liczb binarnych – mniejszej parzystej i większej nieparzystej – różnią się tylko ostatnią cyfrą. Liczba parzysta kończy się cyfrą 0, a nieparzysta – cyfrą 1, np. 12 to 1100, a 13 to 1101.

Takich własności można sformułować więcej i wszystkie można matematycznie udowodnić.

Dobra rada

Liczb dwójkowych nie nazywaj tak samo jak liczb w systemie dziesiętnym. Na przykład liczbę zapisaną jako 1010 w systemie dziesiętnym nazywamy „tysiąc dziesięć", ale w systemie binarnym należy ten zapis odczytać „jeden zero jeden zero".

Liczba	Zapis binarny	Uzasadnienie
0	0	0
1	1	2^0
2	10	2^1
3	11	$2^1 + 2^0$
4	100	2^2
5	101	$2^2 + 2^0$
6	110	$2^2 + 2^1$
7	111	$2^2 + 2^1 + 2^0$
8	1000	2^3
9	1001	$2^3 + 2^0$
10	1010	$2^3 + 2^1$
11	1011	$2^3 + 2^1 + 2^0$
12	1100	$2^3 + 2^2$
13	1101	$2^3 + 2^2 + 2^0$
14	1110	$2^3 + 2^2 + 2^1$
15	1111	$2^3 + 2^2 + 2^1 + 2^0$

Tabela 12.1. Zapis binarny liczb od 0 do 15

Zapamiętaj

Dwójkowy system liczbowy (podobnie jak system dziesiętny) opiera się na zasadzie zapisu wagowo-pozycyjnego. Cyfry w zapisie liczby znajdują się na ponumerowanych pozycjach. Każdej pozycji (licząc od prawej do lewej) odpowiada pewna waga: 1, 2, 4, 8 itd. Wagi są kolejnymi potęgami liczby 2, którą nazywa się podstawą tego systemu liczbowego.

Warto wiedzieć

Liczby większe od 1 zapisane w postaci binarnej składają się z większej liczby cyfr niż ich odpowiedniki w postaci dziesiętnej. Z punktu widzenia człowieka to z pewnością wada, dlatego binarnym systemem liczbowym nie posługujemy się na co dzień.

Ćwiczenie 1

Jaką największą liczbę można zapisać w systemie dwójkowym za pomocą ośmiu cyfr?

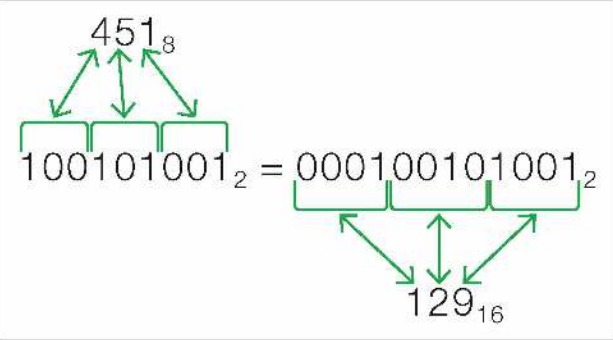
Inne systemy liczbowe

W informatyce często wykorzystuje się również systemy ósemkowy i szesnastkowy. W **systemie ósemkowym** liczby tworzy się z ośmiu cyfr (od 0 do 7), natomiast w **systemie szesnastkowym** jest ich aż szesnaście: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Do zapisu sześciu ostatnich cyfr wykorzystuje się znaki będące kolejnymi literami alfabetu łacińskiego. Aby oznaczyć, w jakim systemie została zapisana liczba, w indeksie dolnym zapisuje się podstawę systemu (rys. 12.3).

System ósemkowy

System szesnastkowy

Podstawy systemów szesnastkowego i ósemkowego są potęgami liczby 2, dlatego stosunkowo łatwo można zamienić liczby zapisane w tych systemach na reprezentację w systemie binarnym i odwrotnie. Każdą cyfrę ósemkową można zapisać za pomocą trzech cyfr binarnych, a każdą cyfrę szesnastkową – za pomocą czterech cyfr binarnych, przy czym mniejsze wartości uzupełniamy z lewej strony zerami (rys. 12.3).



Rys. 12.3. Zamiana zapisu liczby pomiędzy systemami dwójkowym, ósemkowym i szesnastkowym

Warto wiedzieć

System ósemkowy wykorzystuje się m.in. do ustalania praw do pliku w systemach Linux. System szesnastkowy ma zastosowanie np. w adresach IPv6.

12.2. Cyfrowy zapis informacji

Wszelkiego typu informacje (liczby, teksty, grafika, dźwięk itd.) w systemach komputerowych są przedstawiane (kodowane) jako liczby. Najmniejszą, niepodzielną porcją informacji jest **bit** (od ang. ***binary digit***). Pojedynczy bit może przyjmować dwie wartości – zapisywane umownie jako 0 albo 1. Są one cyfrową reprezentacją pewnych fizycznych właściwości nośnika danych lub medium transmisyjnego, np. właściwości elektrycznych lub magnetycznych.

Bit

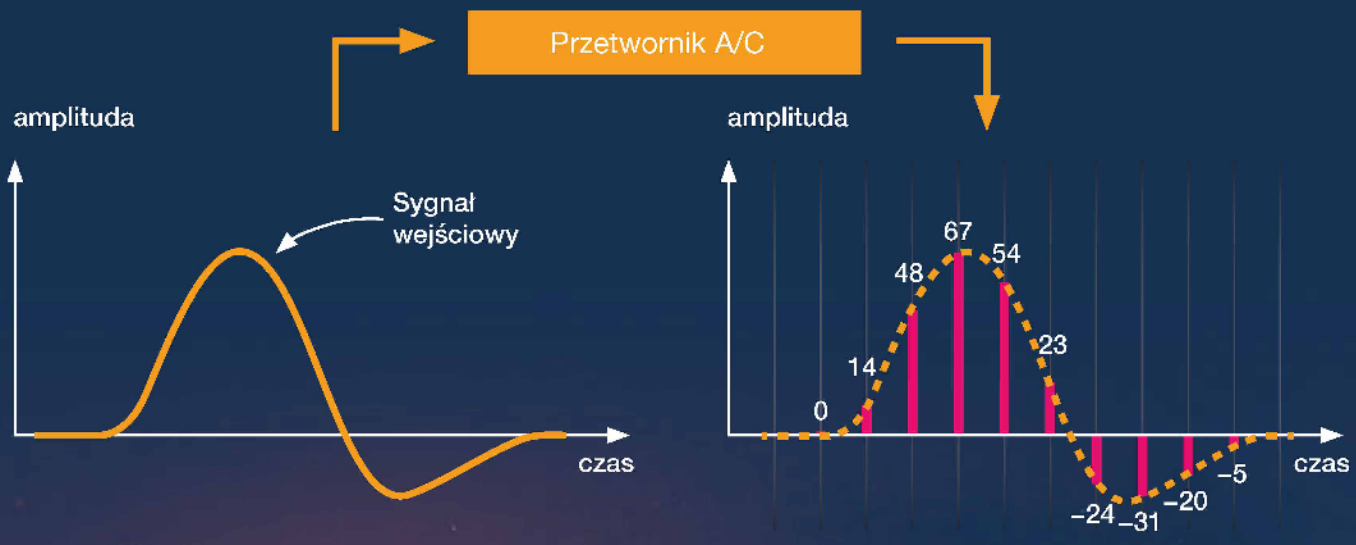
Jak dane analogowe są przekształcane na cyfrowe

Podczas nagrywania filmu smartfonem dźwięk i ruchomy obraz są przetwarzane na postać cyfrową. Dzięki temu możemy przechowywać je w pamięci telefonu, a następnie odtwarzać. Jak działają cyfrowe urządzenia rejestrujące dźwięk i obraz? Przeanalizujmy to na przykładzie.



Mikrofon wbudowany w telefon zamienia falę dźwiękową na sygnały elektryczne odpowiadające amplitudzie i częstotliwości dźwięku.

Przetwornik A/C (analogowo-cyfrowy) próbkuje amplitudy sygnałów elektrycznych, czyli mierzy je tysiące razy w ciągu jednej sekundy i zamienia na ciągi liczb binarnych. Aby zapis cyfrowy był zgodny z dźwiękiem, który słyszy ludzkie ucho, próbkowanie musi się odbywać z odpowiednio wysoką częstotliwością i rozdzielczością. Na przykład dla płyt CD standardem są częstotliwość 44,1 kHz i rozdzielczość 16 bitów na próbkę, czyli 2^{16} = 65 536 możliwych wartości amplitudy fali dźwiękowej.

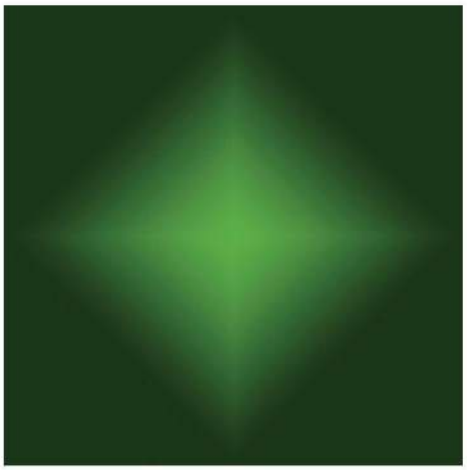


Jak w komputerze zapisuje się obrazy cyfrowe?

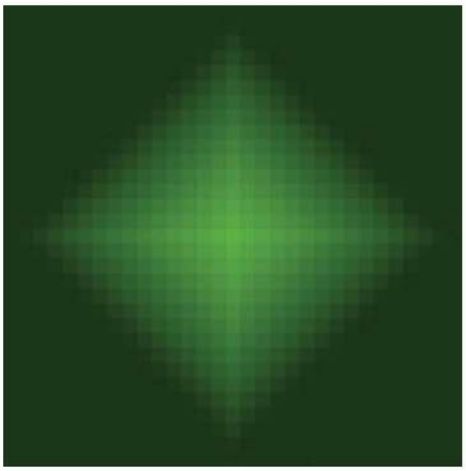
Aby wyjaśnić dokładniej istotę cyfrowego zapisu informacji, posłużymy się przykładem z dziedziny grafiki komputerowej.

Rysunek 12.4 to przykład gradientu, czyli wypełnienia powierzchni równomiernym przejściem z jednego odcienia tej samej barwy do innego.

Rysunek 12.5 powstał z rysunku 12.4 w wyniku ograniczenia liczby użytych odcieni koloru do 16 i zmniejszenia liczby pikseli.



Rys. 12.4. Gradient



Rys. 12.5. Gradient po ograniczeniu liczby pikseli i odcieni

Dzięki zastosowaniu odpowiedniej **rozdzielczości** mamy wrażenie, że obraz z rysunku 12.4 jest ciągły – kolory płynnie przechodzą jeden w drugi. Obraz z rysunku 12.5 sprawia wrażenie zbioru kwadratów o różnych odcieniach zieleni, które są ułożone schodkowo.

Rozdzielczość obrazu, podręcznik *Informatyka na czasie 1. Zakres podstawowy*, s. 188

Przyjrzyjmy się dokładnie rysunkowi 12.5. W poziomie i pionie mamy po 31 pikseli. Na rysunku 12.6 pokazano środkową linię tego obrazu w powiększeniu:



Rys. 12.6. Środkowa linia obrazu w powiększeniu

W cyfrowej reprezentacji każda informacja, również informacja o obrazie, musi mieć postać liczbową. Każdemu z odcieni można przypisać liczbę całkowitą, począwszy od 0.

Jeśli przyjąć, że odcień o najmniejszej intensywności (jasności) koloru zielonego opiszemy liczbą 0, a ten o największej – 15, to piksele tworzące tę linię obrazu opiszemy ciągiem 31 liczb. Przedstawiono to na rysunku 12.7.

Warto wiedzieć

Czasami zamiast słowa piksel używa się słowa pel (ang. ***pic**ture **el**ement*). Stosuje się je zwłaszcza wtedy, gdy mówimy o elementach obrazu wydrukowanego.



Rys. 12.7. Środkowa linia obrazu z ponumerowanymi odcieniami

Każda z tych liczb dziesiętnych w pamięci komputera lub na nośniku danych jest w rzeczywistości zapisywana jako czterocyfrowa liczba dwójkowa. Aby zrozumieć, w jaki sposób tworzone są kody kolorów, przeanalizujmy diagram z rysunku 12.8.

Różnym odcieniom koloru zielonego przyporządkowano liczby binarne od 0000 do 1111. **Najbardziej znaczący bit** – reprezentowany przez pierwszą od lewej cyfrę liczby binarnej – decyduje o tym, czy dany odcień jest w grupie ciemniejszych czy jaśniejszych.

Bit najbardziej znaczący

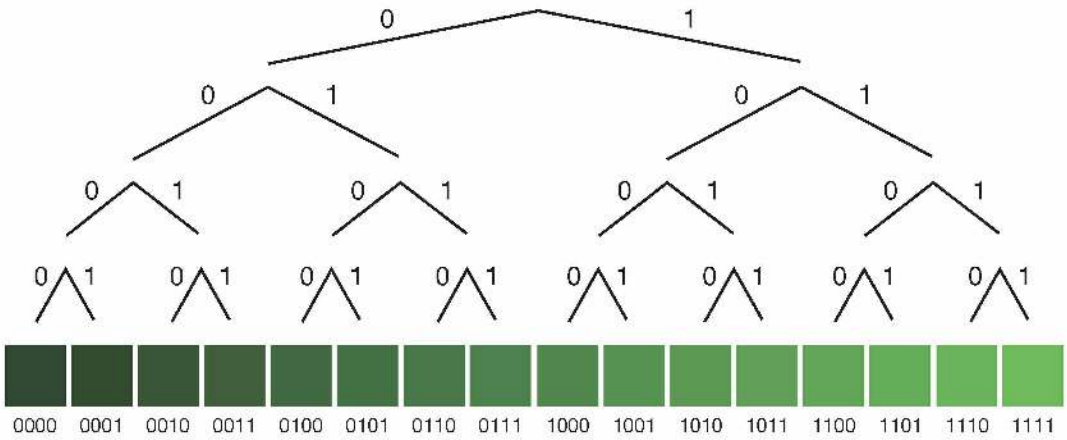
Drzewo decyzyjne

Diagram z rysunku 12.8 ma formę tzw. **drzewa decyzyjnego** – czytamy je od góry. Jeśli postawilibyśmy pytanie „Czy dany odcień jest jednym z jaśniejszych?", to odpowiedź „tak" jest reprezentowana przez cyfrę 1, a odpowiedź „nie"– przez 0. Dlatego wszystkie odcienie bliższe czerni mają pierwszy bit o wartości 0.

Drugi bit określa, czy odcień jest wśród $\frac{1}{4}$ najciemniejszych czy wśród $\frac{1}{4}$ średnio ciemnych (lub analogicznie $\frac{1}{4}$ najjaśniejszych czy w $\frac{1}{4}$ średnio jasnych).

Po wyznaczeniu trzeciego bitu wybór organicza się do dwóch odcieni, a **najmniej znaczący bit** – reprezentowany przez ostatnią cyfrę liczby binarnej – pozwala jednoznacznie określić odcień docelowy.

Bit najmniej znaczący



Rys. 12.8. Diagram ilustrujący tworzenie kodów odcieni w obrazie

Warto wiedzieć

W pierwszych komputerach w latach 40. XX w. nie korzystano z systemu binarnego. W słynnym komputerze ENIAC zastosowano system dwupiątkowy. Każdą cyfrę dziesiętną zastępowano dwoma cyframi: jedną o wartości 0, 1, 2, 3 lub 4, drugą o wartości 0 lub 5, tak aby suma tych dwóch cyfr dawała pierwotną cyfrę dziesiętną.

Każdy odcień koloru zielonego na obrazie można więc jednoznacznie przedstawić jako czterocyfrową liczbę binarną. Do opisania jednego piksela obrazu z rysunku 12.7 ze s. 207 potrzeba zatem 4 bitów.

Zapamiętaj

Wszelkiego typu informacje przechowywane i przetwarzane w systemach komputerowych są przedstawiane (kodowane) jako liczby binarne, składające się z dwóch cyfr: 0 i 1. Cyfry te reprezentują dwa fizyczne stany, np. magnetyczne, elektryczne lub optyczne.

12.3. Algorytmy zamiany reprezentacji liczb

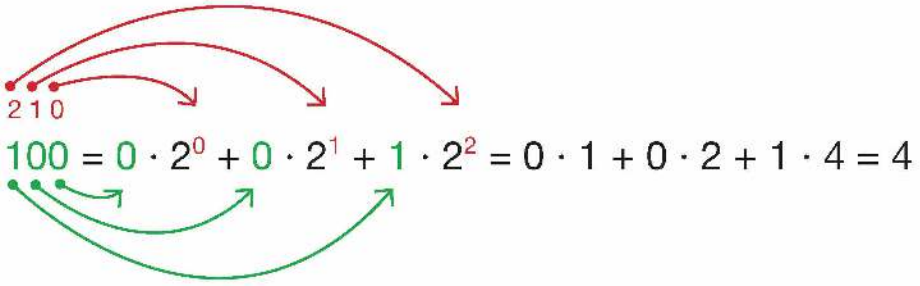
W komputerze liczby zapisuje się w systemie dwójkowym, jednak użytkownicy są przyzwyczajeni do używania systemu dziesiętnego. Dlatego np. podczas pracy z kodem HTML, tworzenia grafiki komputerowej czy konfiguracji sieci pojawia się potrzeba konwersji między systemami liczbowymi.

Zamiana liczby binarnej na liczbę dziesiętną

Zapis dziesiętny liczby binarnej można wyznaczyć, sumując kolejne potęgi liczby 2 odpowiadające wagom kolejnych cyfr. Algorytm można zapisać następująco:

1. Ponumeruj cyfry liczby binarnej od prawej do lewej, zaczynając od 0.
2. Każdą cyfrę potraktuj odpowiednio jako liczbę dziesiętną 0 albo 1 i pomnóż przez liczbę 2 podniesioną do takiej potęgi, jaki jest numer cyfry.
3. Zsumuj otrzymane iloczyny.

Zamianę liczby binarnej 100 na postać dziesiętną można zilustrować w następujący sposób:



Rys. 12.9. Schemat zamiany liczby binarnej na dziesiętną

Dobra rada

Jeśli posługujesz się jednocześnie przynajmniej dwoma systemami liczbowymi, stosuj dodatkowy indeks dolny do określenia podstawy systemu, w którym liczba została zapisana. Jest to szczególnie ważne w przypadku liczb, których postaci wyglądają tak samo w różnych systemach, ale mają różne wartości, np.: $100_{10} = 1100100_2$, a $100_2 = 4_{10}$.

Ćwiczenie 2

Zapisz poniższe liczby binarne w systemie dziesiętnym.

a. 111
b. 10001
c. 1101101

Zamiana liczby dziesiętnej na liczbę binarną

Załóżmy, że chcemy zapisać w postaci binarnej liczbę dziesiętną 123. Aby wyznaczyć kolejne cyfry (bity) zapisu binarnego, musimy wskazać potęgi liczby 2, które w sumie dają tę liczbę.

Zacznijmy od wyznaczenia najmniej znaczącego, czyli ostatniego bitu. Z podanych wcześniej **własności liczb binarnych** wiemy, że wszystkie liczby nieparzyste kończą się cyfrą 1 (wynika to z własności nr 1), dlatego zapis binarny liczby 123 będzie się kończyć cyfrą 1.

Podstawowe własności liczb binarnych, s. 204

Liczby 122 i 123 to kolejne liczby całkowite, więc z własności nr 4 wiemy, że wszystkie bity poza ostatnim będą takie same jak bity liczby 122. Przedostatni bit liczby 122 (a więc i 123) będzie taki jak ostatni bit liczby o połowę mniejszej, a więc 61 (wynika to z własności nr 3). Ta liczba jest nieparzysta, więc jej postać binarna będzie się kończyć cyfrą 1. To oznacza, że zapis binarny liczby 122 kończy się parą cyfr 10 (jeden zero), a liczby 123 – parą 11 (jeden jeden).

Jak znaleźć kolejną cyfrę (trzecią od prawej) zapisu binarnego liczby 123? Wystarczy powtórzyć powyższe rozumowanie: określić parzystość liczby (61 – 1)/2. Wyznaczanie parzystości kolejnych ilorazów należy powtarzać do momentu, gdy otrzymamy iloraz równy 0.

Przebieg tego algorytmu można przedstawić w formie tabeli:

Dobra rada

Kolejne etapy realizacji danego algorytmu możesz w wygodny sposób umieścić w tabeli. Może ona pomóc w zrozumieniu trudniejszego algorytmu oraz pozwala znaleźć ewentualne błędy.

Liczba	Iloraz z dzielenia przez 2	Reszta z dzielenia przez 2
123	61	1
61	30	1
30	15	0
15	7	1
7	3	1
3	1	1
1	0	1

Tabela 12.2. Przebieg algorytmu zapisu liczby 123 w postaci binarnej

Ostatnią kolumnę czyta się od dołu i zapisuje cyfry od lewej do prawej. Liczba 123 w zapisie binarnym to 1111011.

A to ciekawe

Komputery analogowe

Maszyny analogowe, inaczej niż urządzenia cyfrowe, przetwarzały dane w postaci analogowej, tj. za pomocą wielkości fizycznych, zmieniających się w sposób ciągły, takich jak napięcie elektryczne. Komputerem analogowym był, jeden z pierwszych na świecie, analizator równań różniczkowych AKAT-1, zaprojektowany i skonstruowany w 1957 r. przez zespół pod kierunkiem inż. Jacka Karpińskiego. Analizator pomagał tworzyć złożone systemy opisujące dynamicznie zmieniające się procesy, takie jak przepływy cieczy, wymiana ciepła, resorowanie pojazdów. Ergonomiczną obudowę komputera zaprojektowano na warszawskiej Akademii Sztuk Pięknych.



Ćwiczenie 3

Wyznacz postać binarną liczby dziesiętnej 117. Zapisz przebieg algorytmu w tabeli. Jego poprawność możesz sprawdzić np. w kalkulatorze systemu Windows (tryb programisty) albo korzystając z odpowiedniej aplikacji w telefonie lub serwisu internetowego.

Dobra rada

Jeśli chcesz wyznaczyć postać binarną wielu liczb, możesz skorzystać z funkcji `DZIES.NA.DWÓJK` lub `DEC2BIN` w arkuszu kalkulacyjnym.

12.4. Program zamieniający liczbę dziesiętną na binarną

Przygotujemy się teraz do stworzenia programu, który po wpisaniu z klawiatury liczby naturalnej, nie większej niż 255, wypisze ciąg zer i jedynek przedstawiający binarną reprezentację komputerową tej liczby. Ograniczymy się do liczb, które da się zapisać w jednym **bajcie** (ang. *byte*), czyli jako liczby binarne co najwyżej ośmiocyfrowe, złożone z ośmiu bitów.

Bajt

Zapiszmy specyfikację problemu algorytmicznego.

Specyfikacja

Dane: liczba całkowita (nieujemna) nie większa niż 255.

Wynik: zapis binarny liczby całkowitej.

Warto wiedzieć

Jeden bajt reprezentuje aż $2^8 = 256$ różnych liczb binarnych. Tę właściwość wykorzystano w kodzie ASCII, w którym za pomocą jednobajtowych kodów przedstawia się litery alfabetu łacińskiego (języka angielskiego), cyfry, znaki przestankowe, symbole i inne znaki służące do komunikacji tekstowej.

Aby wyznaczyć postać binarną liczby dziesiętnej, możemy postępować w sposób przedstawiony w tabeli 12.2. Wystarczy zapamiętywać kolejne ilorazy z dzielenia przez 2 oraz otrzymywane reszty z dzielenia. Dokładny zapis działania algorytmu przedstawia poniższa lista kroków.

1. Podziel liczbę przez 2, zapisz resztę z dzielenia i zapamiętaj iloraz (część całkowitą).
2. Dopóki ostatnio otrzymany iloraz jest większy od 0, powtarzaj kroki 3, 4 i 5.
   3. Wyznacz resztę z dzielenia ostatnio zapamiętanego ilorazu przez 2.
   4. Zapisz resztę z dzielenia przed ostatnią wcześniej zapisaną resztą.
   5. Podziel iloraz przez 2 i zapamiętaj część całkowitą wyniku jako nowy iloraz.

Po otrzymaniu ilorazu zerowego postać binarną można utworzyć z zapamiętanych reszt zapisanych od prawej do lewej w kolejności ich pojawiania się.

Do zapisania algorytmu w języku Python potrzebna będzie nam nowa wiedza, ponieważ będziemy musieli wyznaczać reszty z dzielenia oraz zapamiętywać kolejne ilorazy (części całkowite). Do tego celu wykorzystamy operator wyznaczania reszty z dzielenia oraz strukturę listy.

Dzielenie całkowite i reszta z dzielenia

W języku Python, podobnie jak w matematyce, rozróżnia się dwie operacje dzielenia, gdy dzielna i dzielnik są liczbami całkowitymi: dzielenie w sensie znanym ze szkoły podstawowej (np. operacja 123/2 daje iloraz równy 61,5) i dzielenie całkowite (operacja 123/2 daje iloraz całkowity 61 i resztę 1).

W języku Python do wyznaczania ilorazu z możliwą częścią ułamkową służy operator /, a do wyznaczania ilorazu całkowitego (części całkowitej ilorazu) – operator //. Resztę z dzielenia wyznaczamy z wykorzystaniem operatora %, zwanego **modulo**. Na przykład wartością zmiennej `iloraz` po przypisaniu `iloraz = 123//2` będzie 61, a wartością zmiennej `reszta` po przypisaniu `reszta = 123%2` będzie 1.

Modulo to operacja wyznaczania reszty z dzielenia jednej liczby przez drugą.

Ćwiczenie 4

Sprawdź efekt działania programu, którego kod ma postać:

```
iloraz = 123//2
reszta = 123%2
print(iloraz)
print(reszta)
```

Warto wiedzieć

Reszta z dzielenia dowolnej liczby całkowitej przez liczbę k może przyjąć wartości od 0 do $k - 1$.

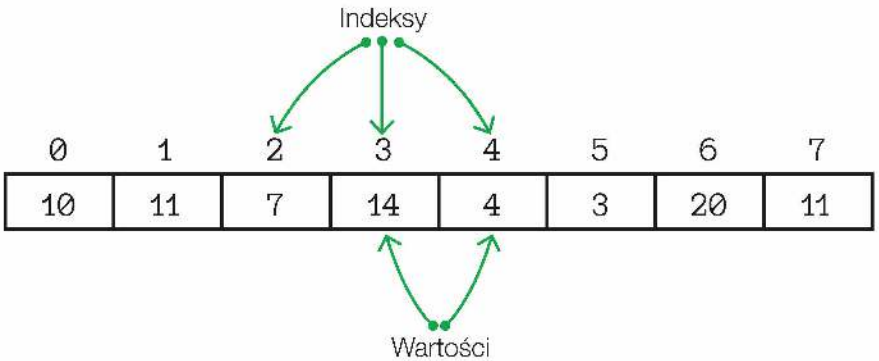
Lista – struktura danych

Aby zapisać w pamięci i przetwarzać w programie większą liczbę danych, np. liczby całkowite, bez konieczności tworzenia osobnej zmiennej dla każdej z nich, można posłużyć się strukturą danych nazywaną **listą** (ang. *list*).

Lista

Lista jest uporządkowana, tzn. każda zapisana na niej wartość ma określone miejsce. Listę liczb można porównać do wiersza tabeli, gdzie w kolejnych komórkach są zapisane kolejne liczby. Każda wartość zapisana na liście ma swoją współrzędną, zwaną **indeksem**. Listy są indeksowane od 0 (rys. 12.10).

Indeks



Rys. 12.10. Interpretacja graficzna listy o 8 wartościach liczbowych

Warto wiedzieć

Przypisanie wartości do wybranego elementu listy oznacza zastąpienie wartości dotychczas przechowywanej nową wartością.

Aby odwołać się do jakiegoś elementu listy, wystarczy podać jego współrzędną (indeks).

Instrukcje:

```
lista[0] = 10
lista[7] = 11
```

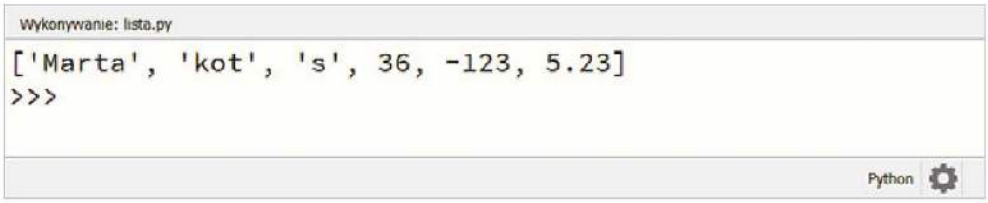
powodują przypisanie wartości 10 pierwszemu elementowi listy (o indeksie 0) i przypisanie wartości 11 ostatniemu elementowi (o indeksie 7).

Listy wartości tworzy się z użyciem nawiasów kwadratowych, które zapisuje się po znaku przypisania. Między nawiasami wstawia się kolejne wartości, oddzielone przecinkami. Aby utworzyć listę o nazwie `lista` zawierającą osiem konkretnych wartości całkowitych, należy napisać: `lista = [1, 11, 7, 14, 4, 3, 20, 10]`.

Struktura listy może przechowywać elementy różnego typu. Przeanalizujmy zapis:

```
1. a = 12 * 3
2. lista = ["Marta", 'kot', 's', a, -123, 5.23]
3. print(lista)
```

W powyższym kodzie do listy wprowadzono napisy, literę, wartość zmiennej a, a także liczby – całkowitą oraz wymierną. Instrukcja z linii 3 umożliwia wypisanie całej zawartości listy (rys. 12.11).



Rys. 12.11. Wyświetlenie elementów listy

Liczba elementów listy może się zmieniać podczas działania programu. Można również utworzyć w pamięci komputera listę bez elementów, pisząc w kodzie źródłowym np. `lista = []`.

Listy – wczytywanie i wypisywanie danych

Napiszemy krótki program o nazwie *Cyfry wspak*, który po wczytaniu z klawiatury ośmiu cyfr (liczb jednocyfrowych) wyświetli je w odwróconej kolejności.

Kod źródłowy programu *Cyfry wspak*

```
1. lista = [0, 0, 0, 0, 0, 0, 0, 0]
2.
3. for i in range(8):
4.     lista[i] = int(input())
5.
6. for cyfra in reversed(lista):
7.     print(cyfra, end="")
```

Dobra rada

W Pythonie napisy możesz zapisywać również między apostrofami. Część programistów woli taki zapis, gdyż nie wymaga on używania klawisza **Shift**.

Dobra rada

Aby wyświetlić wartości znajdujące się na liście o nazwie L, napisz `print(L)`. Wartości zostaną wypisane w nawiasie kwadratowym oddzielone przecinkami.

Kod w wierszu 1 pozwala na utworzenie listy składającej się z ośmiu zer. W dalszej części programu będziemy zamieniać te wartości (zera na liście) na wartości, które wprowadzi użytkownik. W wierszach 3 i 4 zapisano kod pętli, która służy do wprowadzenia docelowych ośmiu liczb z klawiatury: kolejne liczby będą zapisywane na liście w miejsce zer jako elementy o indeksach: 0, 1, 2, ..., 7.

Do wskazania położenia liczby na liście użyto zmiennej pomocniczej `i`, która jest jednocześnie licznikiem pętli. Ponieważ liczby mają być przez program wypisane w odwróconej kolejności (od elementu z indeksem 7 do elementu z indeksem 0), w pętli z wierszy 6 i 7 zastosowano **funkcję `reversed`**. Umożliwia ona dostęp do elementów listy w kolejności od ostatniego.

Funkcja `reversed`

Zauważ, że zmienna pomocnicza `cyfra` służy zarówno do kontroli liczby iteracji (powtórzeń pętli), jak i do ustalenia pozycji cyfry w liczbie.

Ćwiczenie 5

a. Przepisz kod programu *Cyfry wspak* i sprawdź jego działanie dla różnych zestawów danych.

b. Przyjmij, że liczby na liście od razu powinny być zapamiętane w kolejności odwróconej. Napisz program realizujący takie założenie. Kod programu zapisz w pliku *cyfry_wprost.py*.

Zapamiętaj

Lista to struktura programistyczna służąca do przechowywania wielu danych w uporządkowany sposób. Dane są przechowywane w ponumerowanych (indeksowanych) komórkach pamięci. Na liście n-elementowej pierwszy element ma indeks 0, a ostatni: $n - 1$.

Program *Zamiana systemów liczbowych*

Teraz możemy wykorzystać listy do napisania programu *Zamiana*, realizującego przedstawiony wcześniej algorytm zamiany liczby dziesiętnej na liczbę dwójkową.

Oto kod programu znajdujący się w pliku *zamiana.py*:

Kod źródłowy programu *Zamiana*

```
1. cyfry = [0, 0, 0, 0, 0, 0, 0, 0]
2.
3. liczba = int(input("Podaj liczbę:"))
4.
5. cyfry[7] = liczba % 2
6. iloraz = liczba // 2
7.
```

Dobra rada

Listę, której wszystkie wartości są identyczne, możesz utworzyć w prosty sposób. Na przykład pięcioelementową listę zer zapisz: `lista = [0] * 5`

```
8.  i = 1
9.  while iloraz > 0:
10.     cyfry[7-i] = iloraz % 2
11.     iloraz = iloraz // 2
12.     i = i + 1
13.
14. print("Zapis binarny liczby ", liczba, " to: ", end=" ")
15.
16. for cyfra in cyfry:
17.     print(cyfra, end="")
```

Wiersze 5–12 są realizacją w języku Python listy kroków ze s. 211. Linie 16–17 odpowiadają za wyświetlenie postaci binarnej podanej liczby. Reszty z dzielenia zapamiętywane są od razu w kolejności odwróconej, dlatego pierwsza reszta jest zapisywana jako ostatni element listy cyfr (wiersz 5). W linii 9 zastosowano pętlę **while** z warunkiem z kroku 2 algorytmu.

Zapis `cyfry[7-i]` pozwala ustalić pozycje kolejno wyznaczanych cyfr zapisu binarnego.

Zauważ, że program wyświetla zawsze zapis liczby z użyciem ośmiu **bitów**. Takie rozwiązanie daje lepsze wyobrażenie o tym, jak w pamięci komputera zapisuje się dane liczbowe: współczesne komputery zawsze przechowują i przetwarzają dane, używając liczby bitów o wielkości bajta lub jego wielokrotności.

Bit, s. 205

Rysunek 12.12 przedstawia przykład wywołania programu *zamiana.py*.

```
Wykonywanie: zamiana.py
Podaj liczbę: 123
Zapis binarny liczby  123  to: 01111011>>>

Python
```

Rys. 12.12. Wywołanie programu *Zamiana*

Ćwiczenie 6

a. Zapisz kod źródłowy programu *zamiana.py* i uruchom program dla różnych liczb (mniejszych niż 256) wpisanych z klawiatury. Zacznij od sprawdzenia liczby 123.

b. Zmień kod źródłowy programu *zmiana.py* tak, aby program pozwalał zapisywać liczby za pomocą dwóch bajtów (16 bitów). Metodą prób, uruchamiając program jak najmniejszą liczbę razy, poszukaj największej liczby, którą da się zapisać w ten sposób.

Podsumowanie

- Komputery przechowują oprogramowanie i przetwarzają informacje zapisane cyfrowo w kodzie binarnym. Dotyczy to zarówno liczb, jak i tekstów, obrazów, filmów oraz dźwięków.
- Dane wprowadzane do pamięci komputera albo są w postaci cyfrowej, albo trzeba je poddać konwersji analogowo-cyfrowej.
- Bit jest najmniejszą porcją danych możliwą do zapisania w urządzeniach komputerowych. Jest reprezentowany przez cyfry 0 i 1, które wykorzystuje się do zapisywania liczb w systemie binarnym.
- Grupę ośmiu bitów nazywa się bajtem. Podstawową jednostką pamięci we współczesnych komputerach są bajty i ich wielokrotności.
- Wyznaczenie postaci binarnej liczby zapisanej dziesiętnie jest równoważne wskazaniu potęg liczby 2, które w sumie dają tę liczbę.
- Aby zamienić liczbę dziesiętną na dwójkową, należy kolejno dzielić liczbę dziesiętną, a następnie jej dziesiętne ilorazy częściowe, przez liczbę 2, aż otrzymany iloraz będzie równy 0. Postać binarną liczby stanowią reszty uzyskane z kolejnych dzieleń przez 2 zapisane wspak.
- Do wyznaczania ilorazu całkowitego (części całkowitej ilorazu) w języku Python służy operator //, a do wyznaczania reszty z dzielenia – operator %.
- Do przechowywania określonej liczby elementów różnych typów, bez konieczności tworzenia zmiennej dla każdej z nich, służy struktura danych zwana listą.

Zadania

* **1** Najprostszym systemem liczbowym jest kod jedynkowy (unarny). Poszukaj informacji na temat tego systemu liczbowego. Podaj przykład jego praktycznego zastosowania w życiu codziennym.

* **2** Napisz program, który będzie określać parzystość liczby wprowadzonej z klawiatury. W przypadku liczby parzystej powinien się wyświetlić komunikat „Liczba jest parzysta", natomiast w przypadku liczby nieparzystej – komunikat „Liczba jest nieparzysta".

* **3** Napisz program, który zapisze na liście dziesięć kolejnych liczb nieparzystych, począwszy od 1, i wyświetli zawartość listy.

* **4** Napisz program, który wczyta na listę podane przez użytkownika liczby, a następnie wypisze je w odwrotnej kolejności. Przyjmij, że liczba 0 kończy wprowadzanie danych.

* **5** Napisz program, który dla podanych dwóch liczb dwucyfrowych wypisze ich postać binarną, a następnie postać binarną ich sumy.

* **6** Napisz program, który wczyta 10 liczb podanych przez użytkownika oraz wypisze je oddzielone przecinkami, z kropką po ostatniej liczbie.

** **7** Napisz program, który dla liczby wprowadzonej z klawiatury wyznaczy największą potęgę liczby 2 nie większą od tej liczby.

** **8** W słynnym komputerze ENIAC stosowano kodowanie oparte na systemie dwupiątkowym, znane wcześniej z japońskich liczydeł, tzw. sorobanów. Przygotuj krótką prezentację lub artykuł na stronę internetową szkoły na temat tego systemu liczbowego i jego zastosowania do kodowania liczb w komputerze ENIAC.

** **9** Napisz program przechowujący w strukturze listy kwadraty 100 kolejnych liczb naturalnych. Elementem listy o indeksie 0 powinna być liczba 0, elementem o indeksie 1 – liczba 1, elementem o indeksie 2 – liczba 4, elementem o indeksie 3 – liczba 9 itd.

** **10** Napisz program, który dla liczby wprowadzonej z klawiatury wypisze po przecinku wszystkie jej dzielniki jednocyfrowe (łącznie z liczbą 1).

** **11** Napisz program odwracający kolejność cyfr w liczbie całkowitej. Możesz zapamiętywać cyfry w strukturze listy.

** **12** Napisz program, który będzie określać parzystość liczby wprowadzonej z klawiatury. Program nie powinien korzystać z operatora %.

*** **13** Napisz program tworzący zestawienie liczb binarnych odpowiadających liczbom dziesiętnym od 0 do 255. Efektem działania programu powinny być dwie kolumny: po lewej – liczby dziesiętne, a po prawej – liczby binarne.

*** **14** Napisz program odwracający kolejność cyfr w liczbie całkowitej. Nie możesz zapamiętywać cyfr w w strukturze listy.

*** **15** Poniżej przedstawiono zawartość bitową – w kodzie szesnastkowym – czarno-białej grafiki zapisanej w formacie BMP. Oprócz właściwych danych, które wyróżniono pogrubieniem, plik zawiera w nagłówku tzw. metadane (dane o danych), np. dotyczące rozmiaru obrazu.

```
42 4D 4E 00 00 00 00 00 00 00 3E 00 00 00 28 00 00 00 20 00 00
00 04 00 00 00 01 00 01 00 00 00 00 00 10 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF 00 FF
FF FF FF CC CC CC CC AA AA AA AA FF FF FF FF
```

Odszukaj w wiarygodnym źródle znaczenie poszczególnych bajtów nagłówka. Narysuj na kartce grafikę zakodowaną w tym pliku.

*** **16** Napisz program, który będzie zamieniał liczby binarne na liczby dziesiętne. Na przykład po wprowadzeniu przez użytkownika z klawiatury ciągu cyfr 100011 wypisze odpowiedź: 35. Program powinien używać wyłącznie typu całkowitego. Przyjmij, że użytkownik nie używa w zapisie zer nieznaczących oraz że liczba jest co najwyżej 8-bitowa.

13. Czy ta liczba jest pierwsza?

Liczby pierwsze znasz z lekcji matematyki w szkole podstawowej. Choć odkryto je już w starożytności, przez lata twierdzenia dotyczące liczb pierwszych uchodziły wśród matematyków za piękne, ale bezużyteczne. Dopiero w drugiej połowie XX w. znalazły praktyczne zastosowania w informatyce, zwłaszcza w kryptografii. Twierdzenia te są podstawą działania niektórych protokołów szyfrowania, czyli algorytmów, które zapewniają poufność komunikacji w internecie i bezpieczeństwo transakcji bankowych. W tym temacie dowiesz się więcej na temat liczb pierwszych i ich znaczenia.

Cele lekcji

- Przypomnisz sobie i lepiej zrozumiesz zagadnienie podzielności liczb całkowitych.
- Poznasz algorytmy sprawdzające, czy liczba jest pierwsza.
- Nauczysz się stosować funkcje w języku Python.
- Zastosujesz funkcję `sqrt` z modułu `math`.

13.1. Liczby złożone i liczby pierwsze

Liczby złożone

Dodatnie liczby naturalne, które można przedstawić w postaci iloczynu liczb naturalnych mniejszych od nich, nazywamy **liczbami złożonymi**. Ich przykładami są: $9 = 3 \cdot 3$, $10 = 2 \cdot 5$, $120 = 20 \cdot 3 \cdot 4 \cdot 5$, $1115 = 3 \cdot 5 \cdot 7 \cdot 11$.

Liczby pierwsze

Pozostałe liczby, z wyjątkiem liczby 1, są **liczbami pierwszymi**. Liczbą pierwszą nazywamy więc liczbę, która jest podzielna wyłącznie przez 1 i samą siebie. Wśród liczb mniejszych od 100 jest dwadzieścia pięć liczb pierwszych:
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.
Na rysunku 13.1 oznaczono je zielonym tłem.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	

Rys. 13.1. Liczby złożone i liczby pierwsze mniejsze niż 100

Warto wiedzieć

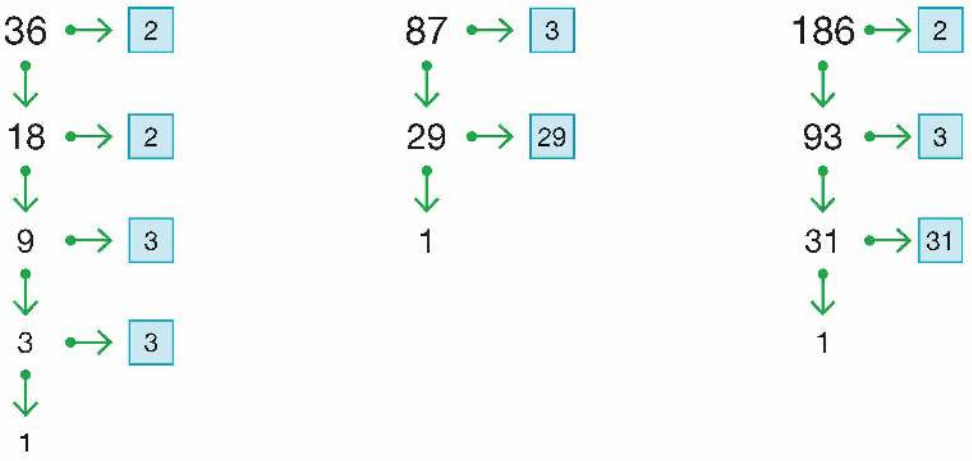
Liczba 1 jest wyjątkowa. Przez setki lat uważano ją za liczbę pierwszą. Dziś przyjmuje się, że nie jest ani pierwsza, ani złożona.

Podstawowe twierdzenie arytmetyki

Jedno z najważniejszych twierdzeń matematycznych dotyczących liczb pierwszych nazywane jest **podstawowym twierdzeniem arytmetyki** i brzmi następująco: każda liczba naturalna (większa od 1) jest albo liczbą pierwszą, albo iloczynem liczb pierwszych.

Iloczyn liczb pierwszych, który tworzy daną liczbę, nazywamy **rozkładem na czynniki pierwsze**. Jest on jedyny w swoim rodzaju, tzn. każda liczba ma tylko jedną unikatową formułę rozkładu, która do niej przynależy. Na rysunku 13.2 przedstawiono rozkład na czynniki pierwsze liczb 36, 87 i 186. Liczby te możemy zapisać następująco: $36 = 2 \cdot 2 \cdot 3 \cdot 3$; $87 = 3 \cdot 29$; $186 = 2 \cdot 3 \cdot 31$.

Rozkład na czynniki pierwsze



Rys. 13.2. Rozkład liczb 36, 87 i 186 na czynniki pierwsze

Okazuje się, że im większa liczba, tym trudniej sprawdzić, czy jest pierwsza czy złożona, a więc czy da się ją rozłożyć na czynniki pierwsze. W przypadku dużych liczb rozstrzygnięcie tego w rozsądnym czasie stanowi zadanie praktycznie niewykonalne, nawet dla współczesnych komputerów. Ta trudność jest podstawą bezpieczeństwa działania niektórych algorytmów kryptograficznych (np. algorytmu RSA).

Liczba 2 jest jedyną parzystą liczbą pierwszą, więc w praktyce poszukiwania liczb pierwszych wymagają sprawdzania tylko liczb nieparzystych.

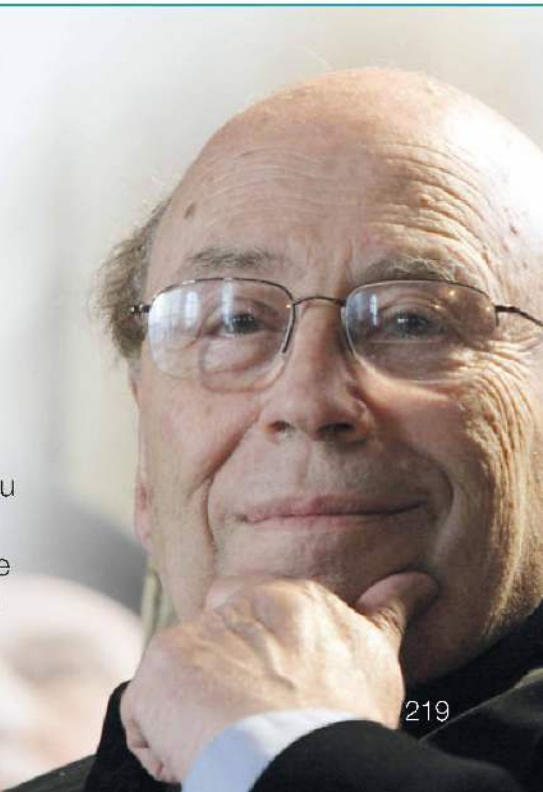
Warto wiedzieć

Stosunkowo łatwo sprawdzić, czy podana liczba nieparzysta jest podzielna przez 5, ponieważ kończy się ona cyfrą 5.

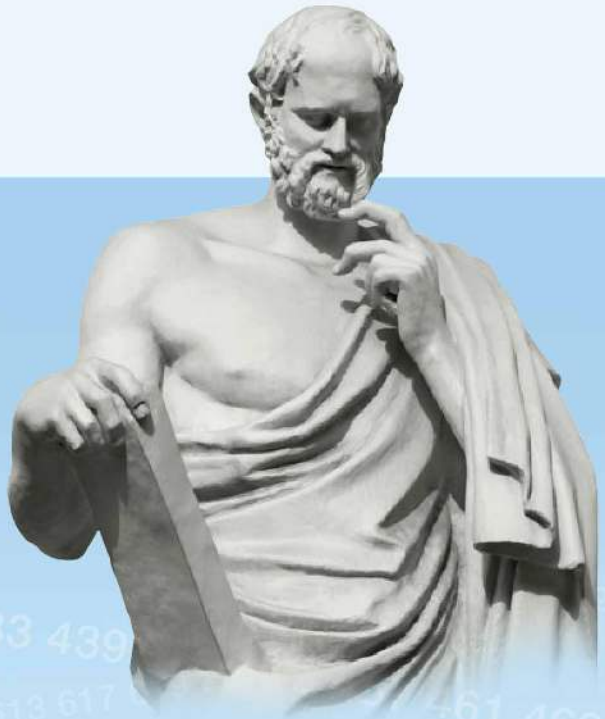
A to ciekawe

Szybki test pierwszości

Dla bardzo dużych liczb bardzo trudno ustalić, czy są one pierwsze czy złożone. Powszechnie stosowanym i w praktyce najszybszym testem pierwszości jest test Millera–Rabina. Algorytm stworzył w 1976 r. amerykański profesor informatyki Gary Miller, a w 1980 r. ulepszył go urodzony we Wrocławiu Michael O. Rabin (na zdjęciu). Dzięki temu algorytmowi można przeprowadzić test pierwszości nawet dla ogromnych liczb, rzędu 150 cyfr. Ponieważ jest to test probabilistyczny, nie wskazuje on żadnych dzielników testowanej liczby. Oznacza to, że liczba może zostać błędnie rozpoznana jako pierwsza. Prawdopodobieństwo takiego zdarzenia jest jednak znikome – wynosi ok. 1 do 10^{12}.



Co warto wiedzieć o liczbach pierwszych?



Ile jest liczb pierwszych?

Liczby pierwsze odkryto i badano już w starożytności. W IV w. p.n.e. Euklides udowodnił, że jest ich nieskończenie wiele. Dotąd nie odkryto jednak wzoru, według którego można byłoby dla konkretnej liczby n wyznaczyć n-tą liczbę pierwszą.

2 to najmniejsza i jedyna parzysta liczba pierwsza

Liczby postaci 2^n-1, gdzie n jest liczbą naturalną, nazywamy liczbami Mersenne'a. Wiele z nich jest liczbami pierwszymi

Liczby pierwsze większe od 5 kończą się cyframi 1, 3, 7, 9

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89

Liczby czworacze to cztery liczby pierwsze mające postać: $p, p+2, p+6, p+8$

Liczby bliźniacze to dwie liczby pierwsze, których różnica wynosi 2

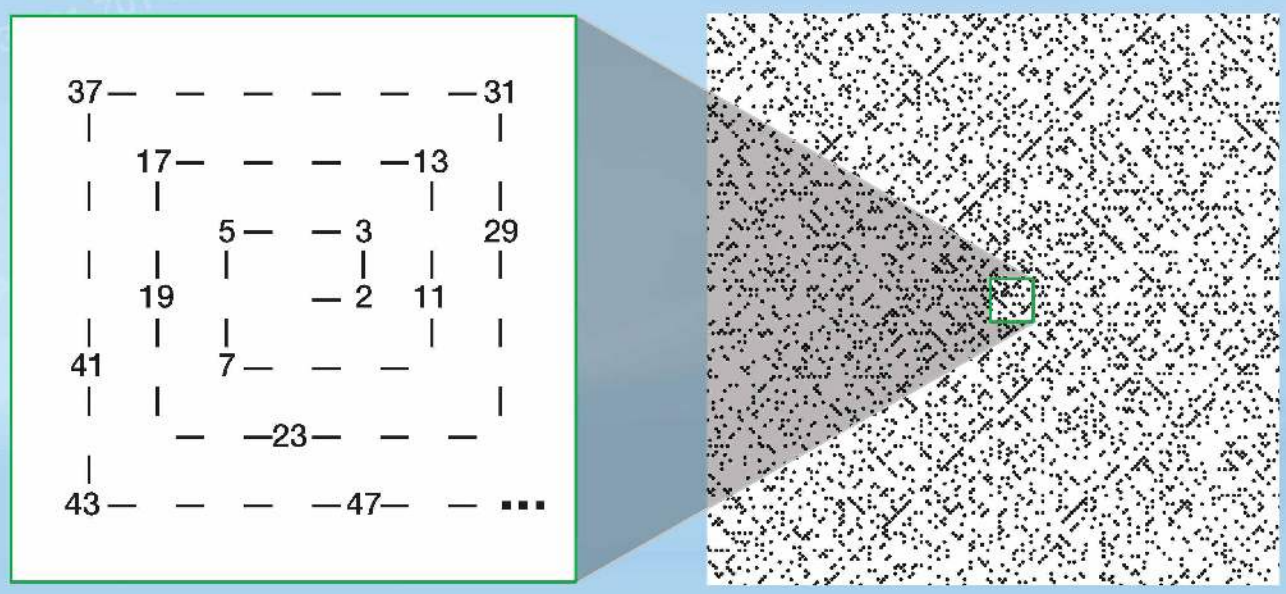
Liczby pierwsze w kryptografii

Liczbę utworzoną z iloczynu dwóch dużych liczb pierwszych bardzo trudno w rozsądnym czasie rozłożyć na czynniki pierwsze, nawet z użyciem najszybszych komputerów. Wykorzystali to twórcy algorytmu RSA, który stosuje się do szyfrowania komunikacji w sieci Internet oraz w bankowości elektronicznej. Liczby pierwsze używane w tym algorytmie mają nawet 2048 bitów, czyli składają się z 2048 cyfr binarnych.



Spirala Ulama

W 1963 r. polski matematyk Stanisław Ulam przedstawił graficznie rozmieszczenie liczb pierwszych. Na kwadratowej tablicy spiralnie wypisał kolejne liczby naturalne. Po naniesieniu bardzo wielu liczb okazało się, że na niektórych liniach liczby pierwsze grupują się częściej niż na innych. Do tej pory nie wyjaśniono, dlaczego tak się dzieje.



Jak liczby pierwsze pomagają rozpoznawać anagramy?

Rozkład dowolnej liczby na czynniki pierwsze jest jednoznaczny (np. dla 30 jest to wyłącznie rozkład $2 \cdot 3 \cdot 5$). Ta własność jest podstawą działania pewnego algorytmu sprawdzającego, czy dwa wyrazy są anagramami (składają się z tych samych liter). Kolejnym literom alfabetu przypisuje się kolejne liczby pierwsze. Jeśli iloczyny liczb odpowiadających literom w wyrazach są równe, wyrazy są anagramami.





13.2. Podzielność liczb

Zanim stworzymy program sprawdzający, czy podana liczba jest pierwsza, napiszemy prostsze programy związane z podzielnością. Wprowadzimy przy tej okazji nową konstrukcję programistyczną – funkcje.

Funkcje w języku Python

Aby rozwiązać bardziej złożony problem algorytmiczny, najlepiej rozłożyć go na prostsze podproblemy. W praktyce oznacza to wyodrębnianie fragmentów programów (podprogramów), które w języku Python nazywa się **funkcjami**.

Funkcja

Każda funkcja musi mieć nadaną nazwę. Po nazwie zawsze występują nawiasy okrągłe, w których podaje się **parametry funkcji**.

Parametr funkcji

```
def nazwa_funkcji(lista parametrów):
    instrukcje
```

Definicja funkcji składa się z nagłówka funkcji i bloku instrukcji, które stanowią jej treść. Nagłówek rozpoczyna się słowem kluczowym **def**, po którym zapisuje się nazwę funkcji, a w nawiasie listę parametrów (nazwę każdego z nich), o ile funkcja ma jakieś parametry. Wiersz nagłówka kończy się dwukropkiem.

Warto wiedzieć

Zazwyczaj nie zaleca się używania polskich znaków w nazwach funkcji i zmiennych.

W treści funkcji umieszcza się instrukcje, które tworzą podprogram (stanowią pewną całość). Wszystkie linie kodu wchodzące w skład bloku instrukcji są przesunięte w prawo, podobnie jak w przypadku **pętli** i instrukcji warunkowej.

Podstawowe typy instrukcji w języku Python, s. 262

Efekt działania funkcji albo wyświetla się na ekranie, albo za pomocą **instrukcji `return`** jest przekazywany z powrotem do instrukcji programu, która funkcję wywołała.

Instrukcja `return`

Dzięki konsekwentnemu stosowaniu funkcji tworzymy kod bardziej przejrzysty i łatwiejszy do modyfikowania w przyszłości. Unikamy także powtórzeń, ponieważ tę samą funkcję można wywołać wiele razy dla różnych parametrów.

Badanie podzielności przez 3

Aby zobaczyć, jak w praktyce korzysta się z funkcji, napiszemy program, który będzie sprawdzał, czy podana przez użytkownika liczba jest podzielna przez 3.

Sprawdzenie podzielności danej liczby przez 3 sprowadza się do ustalenia, czy reszta z dzielenia tej liczby przez 3 jest równa zero.

Oto specyfikacja problemu:

Specyfikacja

Dane: dwie liczby naturalne a i b.

Wynik: reszta z dzielenia liczby a przez liczbę b.

Stworzymy funkcję o nazwie ResztaZDzielenia, sprawdzającą resztę z dzielenia liczby *a* przez liczbę *b*, a wywołamy ją dla podanej przez użytkownika liczby i *b* równego 3.

Poniżej znajduje się kod źródłowy programu.

Kod źródłowy programu sprawdzającego podzielność liczby przez 3

```
1. def ResztaZDzielenia(a, b):
2.     return a % b
3.
4. liczba = int(input("Podaj liczbę: "))
5.
6. print("Reszta z dzielenia ", liczba, end=" ")
7. print("przez 3 wynosi:", end=" ")
8. print(ResztaZDzielenia(liczba, 3))
```

W linii 8 znajduje się wywołanie funkcji ResztaZDzielenia przy użyciu parametrów: wartości zmiennej `liczba` oraz 3. W ten sposób sprawdzimy podzielność podanej liczby przez 3. Kod w linii 4 odpowiada za nadanie przez użytkownika wartości zmiennej `liczba`.

Definicja funkcji ResztaZDzielenia znajduje się w wierszach 1 i 2. W nawiasach znajduje się lista parametrów funkcji. Parametry te przekazują wartości z programu głównego (odpowiednio: wartość zmiennej `liczba` oraz liczbę 3). Wiersz 2 to treść funkcji, która wyznacza resztę z dzielenia (z wykorzystaniem operatora %) i zwraca rezultat, stąd słowo kluczowe `return`.

Treść funkcji może się składać z wielu instrukcji. Wszystkie jednak muszą być wcięte względem nagłówka o tę samą liczbę spacji (np. 4) lub wielokrotność tej liczby.

Zauważ, że tak zdefiniowana funkcja jest ogólna i może być zastosowana do sprawdzania podzielności dowolnych liczb.

Na rysunku 13.3 przedstawiono przykładowe wywołanie programu wyznaczającego resztę z dzielenia przez 3.

Dobra rada

Staraj się tworzyć funkcje o możliwie szerokim zastosowaniu. Dzięki temu będzie można je wywołać dla różnych parametrów. Tak tworzone są również funkcje w arkuszach kalkulacyjnych.

```
Wykonywanie: podzielnosc_3.py
Podaj liczbę: 23101985
Reszta z dzielenia 23101985 przez 3 wynosi: 2
>>>
                                                    Python
```

Rys. 13.3. Wywołanie programu sprawdzającego podzielność przez 3

Parametry określone w momencie definiowania nowej funkcji nazywa się **parametrami formalnymi**. Natomiast parametry, dla których funkcja wywołana jest w programie głównym, to **parametry aktualne**. Parametr aktualny przekazuje więc faktyczną wartość (np. podaną przez użytkownika) do funkcji, która została wcześniej zdefiniowana.

Parametry formalny i aktualny funkcji

Ćwiczenie 1

a. Przepisz kod źródłowy programu sprawdzającego podzielność liczby przez 3 i sprawdź jego działanie na różnych przykładach.

b. Zmodyfikuj treść programu tak, aby sprawdzał on podzielność podanej liczby przez 17.

Badanie podzielności przez kilka liczb jednocześnie

Zmodyfikujmy kod naszego programu tak, aby wyznaczał reszty z dzielenia podanej liczby przez cztery kolejne liczby pierwsze: 2, 3, 5, 7.

Wynikiem działania programu powinny być cztery liczby: reszta z dzielenia przez 2, reszta z dzielenia przez 3, reszta z dzielenia przez 5 i reszta z dzielenia przez 7.

Dzięki temu, że zdefiniowaliśmy wcześniej funkcję `ResztaZDzielenia`, modyfikacja programu polega wyłącznie na dopisaniu do programu głównego odpowiednich wywołań funkcji.

Oto kod źródłowy programu:

Fragment kodu źródłowego programu obliczającego reszty z dzielenia przez 2, 3, 5, 7

```
1.  def ResztaZDzielenia(a, b):
2.      return a % b
3.
4.  liczba = int(input("Podaj liczbę: "))
5.
6.  print("Reszty z dzielenia", liczba, end=" ")
7.  print("przez 2, 3, 5, 7:", end=" ")
8.  print("wynoszą odpowiednio:")
9.  print(ResztaZDzielenia(liczba, 2))
10. print(ResztaZDzielenia(liczba, 3))
11. print(ResztaZDzielenia(liczba, 5))
12. print(ResztaZDzielenia(liczba, 7))
```

Ćwiczenie 2

a. Przepisz kod źródłowy programu obliczającego reszty z dzielenia przez 2, 3, 5, 7 i sprawdź jego działanie dla różnych liczb. Instrukcję z linii 6–8 zapisz w jednej linii.

b. Skróć powyższy kod, korzystając z pętli **for**. Zastosuj listę [2, 3, 5, 7].

Zapamiętaj

Funkcje definiujemy, gdy chcemy nazwać pewną grupę instrukcji i oddzielić fragment kodu od reszty programu – zwykle po to, aby go wielokrotnie wykorzystać.

13.3. Test pierwszości liczby

Matematycy od wieków zastanawiali się, jak sprawdzić, czy liczba jest pierwsza. Dziś naukowcy szukają przede wszystkim szybkich algorytmów, które będą to rozstrzygać.

Test pierwszości – algorytm najprostszy

Aby stwierdzić, czy jakaś liczba n jest pierwsza czy złożona, należy sprawdzić, czy ma ona jakikolwiek dzielnik różny od 1 i od niej samej. Można to zrobić, dzieląc n kolejno przez 2, przez 3 itd. aż do $n - 1$.

Przed napisaniem programu komputerowego zapiszemy specyfikację.

Specyfikacja

Dane: liczba naturalna większa od 1.

Wynik: komunikat „Liczba pierwsza", jeśli liczba jest liczbą pierwszą, komunikat „Liczba złożona" – w przeciwnym wypadku.

Poniżej znajduje się kod źródłowy programu *Test pierwszości*.

Kod źródłowy programu *Test pierwszości*

```
1.  def CzyPierwsza(n):
2.      for i in range(2, n):
3.          if n % i == 0:
4.              return 0
5.      return 1
6.
7.  liczba = int(input("Podaj liczbę: "))
8.
9.  if CzyPierwsza(liczba) == 1:
10.     print("Liczba pierwsza")
11. else:
12.     print("Liczba złożona")
```

W liniach 1–5 znajduje się definicja funkcji `CzyPierwsza`. Jej celem jest sprawdzenie podzielności podanej przez użytkownika liczby przez liczby od 2 do $n - 1$. W przypadku, kiedy zostanie znaleziony dzielnik, wynikiem funkcji jest wartość 0 (linia 4), którą możemy traktować jako wartość logiczną fałsz. Wówczas funkcja kończy swoje działanie i nie wykonuje kolejnych dzieleń. Jeśli nie zostanie znaleziony dzielnik, to wynikiem funkcji jest wartość 1, interpretowana jako wartość logiczna prawda. Funkcja kończy wtedy swoje działanie (linia 5). W Pythonie można też w kodzie funkcji po instrukcji `return` stosować słowa kluczowe `True` (prawda) lub `False` (fałsz) zamiast wartości 1 i 0.

W programie głównym w linii 9 wywoływana jest funkcja `CzyPierwsza`. W zależności od zwróconej przez funkcję liczby na ekranie wyświetla się odpowiedni komunikat (linie 10 i 12).

Warto wiedzieć

W latach 90. XX w. pewien producent procesorów był zmuszony wymienić jeden z modeli, ponieważ wykonywał on operacje dzielenia z mniejszą dokładnością, niż zakładano. Wynosiła ona 9 zamiast 19 miejsc po przecinku. Wada ta ujawniła się przy szacowaniu tzw. stałej Bruna, która dotyczy liczb bliźniaczych.

Ćwiczenie 3

a. Przepisz kod źródłowy programu i sprawdź jego działanie.
b. Przygotuj tabelę, w której rozpiszesz kolejne etapy działania funkcji `CzyPierwsza` dla $n = 9$ oraz $n = 11$.

Program *Test pierwszości*, choć daje poprawne wyniki, wykonuje wiele niepotrzebnych operacji wyznaczania reszt z dzielenia. Liczbę tych operacji możemy stosunkowo łatwo ograniczyć.

Test pierwszości – ulepszony algorytm

Sprawdzanie podzielności podanej liczby przez liczby parzyste większe od 2 nie ma sensu. Wystarczy więc zbadać podzielność przez 2, a potem sprawdzać wyłącznie liczby nieparzyste.

Liczbę wykonywanych dzieleń można ograniczyć jeszcze bardziej. Oto rozkłady liczb 36, 100 i 42 na czynniki:

$36 = 2 \cdot 18$	$100 = 2 \cdot 50$	$42 = 2 \cdot 21$
$36 = 3 \cdot 12$	$100 = 4 \cdot 25$	$42 = 3 \cdot 14$
$36 = 4 \cdot 9$	$100 = 5 \cdot 20$	$42 = 6 \cdot 7$
$36 = 6 \cdot 6$	$100 = 10 \cdot 10$	

Warto wiedzieć
Dzielnik właściwy liczby to każdy dzielnik mniejszy od tej liczby.

Przykłady te pokazują, że gdy liczbę złożoną n przedstawiamy w postaci iloczynu dwóch czynników większych od 1, to przynajmniej jeden z tych czynników musi być mniejszy lub równy $\sqrt{n}$.

Dla liczby 36 są to odpowiednio liczby 2, 3, 4 i 6, przy czym $6 = \sqrt{36}$. Podobnie dla pozostałych liczb – mniejszy z dzielników liczby nie jest większy od pierwiastka z tej liczby. Tak więc najmniejszy dzielnik właściwy liczby złożonej n musi być mniejszy bądź równy $\sqrt{n}$.

Ćwiczenie 4

Sprawdź za pomocą kalkulatora, czy 401 jest liczbą pierwszą. Ile dzieleń było koniecznych, aby to rozstrzygnąć? Zastosuj wskazówkę zapisaną powyżej.

Operacja pierwiastkowania

Operacja pierwiastkowania nie jest standardowym działaniem matematycznym w języku Python, dlatego skorzystamy z dodatkowej funkcji matematycznej. Takie funkcje przechowywane są w tzw. modułach.

Moduł

Moduły są to pliki z rozszerzeniem *py*, które zawierają definicje wybranych funkcji dotyczących pewnego zagadnienia.

Funkcja `sqrt`

W naszym programie wykorzystamy **funkcję** `sqrt` znajdującą się

Moduł `math`

w **module** `math`. Moduł ten zawiera wiele funkcji, które znasz z lekcji matematyki, m.in. funkcje trygonometryczne, wykładnicze, logarytmiczne.

Aby zaimportować funkcję `sqrt`, należy na początku kodu dodać instrukcję:

```
from math import sqrt
```

Poniżej znajduje się tekst programu *Ulepszony test pierwszości*.

Kod źródłowy programu *Ulepszony test pierwszości*

```
1.  from math import sqrt
2.
3.  def CzyPierwsza(n):
4.      if n == 2:
5.          return 1
6.      if n % 2 == 0:
7.          return 0
8.
9.      pierwiastek = int(sqrt(n))
10.     for i in range(3, pierwiastek+1, 2):
11.         if n % i == 0:
12.             return 0
13.     return 1
14.
15. liczba = int(input("Podaj liczbę:"))
16.
17. if CzyPierwsza(liczba) == 1:
18.     print("Liczba pierwsza")
19. else:
20.     print("Liczba złożona")
```

Dobra rada

Pamiętaj, że funkcja `range` postaci `range(k, n)` generuje indeksy (ciąg liczb całkowitych) od k do $n - 1$.

Ulepszenia wprowadzono w definicji funkcji `CzyPierwsza`. W liniach 4–7 nadajemy status liczby pierwszej liczbie 2 (linia 5) oraz status liczby złożonej wszystkim liczbom parzystym większym od 2 (linia 7).

W linii 9 wprowadzono zmienną `pierwiastek`, której wartość stanowi część całkowita przybliżenia $\sqrt{n}$, otrzymanego dzięki funkcji `sqrt`, znajdującej się w module `math` (wskazanym w linii 1). W pętli z wierszy 10–13 ograniczamy liczbę wykonań – w tym celu używamy wartości zmiennej `pierwiastek` oraz testujemy tylko liczby nieparzyste (trzecim argumentem funkcji `range` jest krok o długości 2).

Zwróć uwagę na to, że drugim argumentem funkcji `range` jest wartość `pierwiastek+1`. W przeciwnym przypadku np. liczba 9 byłaby uznana za liczbę pierwszą ograniczamy ją do $\sqrt{n}$.

Ćwiczenie 5

a. Przepisz kod źródłowy programu i sprawdź jego działanie.

b. Przygotuj tabelę, w której rozpiszesz kolejne etapy działania funkcji `CzyPierwsza` dla $n = 9$ oraz $n = 11$.

Podsumowanie

- Liczbą złożoną nazywamy liczbę większą od 1, która ma przynajmniej jeden dzielnik różny od 1 i od niej samej.
- Liczba pierwsza to liczba większa od 1, która dzieli się wyłącznie przez samą siebie oraz przez 1.
- Efektywność algorytmu można poprawić przez ograniczenie zbędnych działań (operacji matematycznych). Na przykład przy badaniu pierwszości liczby wystarczy sprawdzać podzielność przez 2 oraz przez liczby nieparzyste nie większe niż pierwiastek kwadratowy z danej liczby.
- Korzystanie z funkcji poprawia czytelność programu i umożliwia wielokrotne wykorzystywanie tego samego kodu.
- Instrukcja `return` przerywa wykonywanie instrukcji i zwraca do programu głównego wartość określoną w funkcji.
- W języku Python można korzystać z modułu matematycznego o nazwie `math`, który zawiera wiele wbudowanych funkcji matematycznych, ułatwiających wykonanie w programie bardziej złożonych obliczeń.
- Funkcja `sqrt` znajduje się w module `math` i zwraca przybliżoną wartość pierwiastka kwadratowego z podanej liczby.
- Wartości logiczne w języku Python można zapisywać jako 1 i 0 lub jako `True` i `False`.

Zadania

* **1** Napisz program wyznaczający wszystkie dzielniki liczby 80.

* **2** Napisz program znajdujący najmniejszą i największą trzycyfrową liczbę pierwszą.

* **3** Napisz funkcję, która będzie badała podzielność podanej liczby jednocześnie przez 3 i 7. W teście użyj pojedynczej instrukcji logicznej (bez operatora **and**).

* **4** Napisz program, który dla dwóch podanych przez użytkownika liczb wyświetli informację o tym, czy ich suma jest liczbą pierwszą.

* **5** Napisz program, który dla danej liczby rzeczywistej a wyświetli wartości $\log_{10} a$. Skorzystaj z gotowej funkcji `log10` wbudowanej w moduł `math`.

* **6** Napisz program wyświetlający liczby, których pierwiastki mieszczą się w podanym przez użytkownika przedziale. Na przykład dla przedziału [5; 6] są to liczby: 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36. W kodzie wykorzystaj moduł `math` i funkcję `sqrt`. Sprawdź działanie programu dla przedziału [14; 19].

** **7** Stwórz program, który wyznaczy wszystkie trzycyfrowe liczby pierwsze.

** 8 Przygotuj w arkuszu kalkulacyjnym skoroszyt, który pozwoli rozstrzygnąć, czy prawdziwe jest następujące twierdzenie: każda liczba złożona mniejsza od 100 dzieli się przynajmniej przez jedną z liczb: 2, 3, 5, 7. Skorzystaj z funkcji MOD.

** 9 Napisz program, który po podaniu przez użytkownika liczby wymiernej (typu `float`) wyświetli zaokrąglenia tej liczby w górę i w dół do najbliższej liczby całkowitej. Skorzystaj z funkcji `ceil` i `floor` z modułu `math`.
Uwaga: Zaokrąglenie do najbliższej liczby całkowitej w górę nazywa się sufitem, a w dół – podłogą. Stąd nazwy funkcji pochodzące od angielskich słów *ceiling* i *floor*.

** 10 Napisz program wyznaczający wszystkie dzielniki pierwsze liczby 80.

** 11 Napisz w języku Python prostą grę, w której dwóch użytkowników podaje po jednej liczbie 6-cyfrowej. Wygrywa ten, kto wskazał liczbę, która ma więcej dzielników właściwych.

** 12 Znajdź pary dwucyfrowych i trzycyfrowych liczb bliźniaczych.
Uwaga: Liczby bliźniacze to dwie liczby pierwsze, których różnica wynosi 2.

** 13 Liczba 362 881 ma tę własność, że przy dzieleniu przez 2, 3, 4, 5, 6, 7, 8, 9 daje resztę 1. Znajdź najmniejszą liczbę o tej własności.

*** 14 Napisz program, który wyznaczy wszystkie trzycyfrowe liczby doskonałe. Liczba doskonała to liczba równa sumie swych dzielników różnych od niej samej, np. 28 = 1 + 2 + 4 + 7 + 14.

*** 15 Wśród poniższych liczb są iloczyny dwóch liczb pierwszych nieparzystych. Napisz program zawierający funkcję `sprawdz`, która pozwoli znaleźć te liczby.
1 838 947 883
1 004 098 109
851 260 967
809 509 261
669 266 311
Wskazówka: Funkcja `sprawdz` powinna zwracać wartość znalezionego dzielnika liczby, o ile liczba nie jest pierwsza. Wartość tę można zapamiętać. Wówczas pozostaje sprawdzić jedną liczbę. Jaką?

*** 16 Znajdź jak największą liczbę mniejszą od 2 000 000 000, która jest iloczynem dwóch liczb pierwszych.

*** 17 Napisz program, który wypisze wszystkie pary liczby bliźniaczych z przedziału [2; 3000].
Uwaga: Liczby bliźniacze to dwie liczby pierwsze, których różnica wynosi 2.

*** 18 Napisz program, który wyświetli listę wszystkich par liczb zaprzyjaźnionych mniejszych niż milion.
Uwaga: Dwie liczby uważa się za zaprzyjaźnione, jeśli sumy dzielników właściwych tych liczb są równe.

14. Algorytm Euklidesa i działania na ułamkach

Jednym z najstarszych algorytmów obliczeniowych jest procedura wyznaczania największego wspólnego dzielnika dwóch liczb, opisana przez greckiego matematyka Euklidesa. Algorytm ten służył najpierw do szukania odcinka, który byłby wspólną miarą dla dwóch odcinków. Dzisiaj znalazł zastosowanie m.in. w kryptograficznym algorytmie RSA, wykorzystywanym np. w podpisach cyfrowych. W tym temacie użyjemy algorytmu Euklidesa do wykonywania operacji na ułamkach.

Cele lekcji

- Przypomnisz sobie pojęcia: największy wspólny dzielnik, najmniejsza wspólna wielokrotność oraz algorytm Euklidesa.
- Poznasz przykład zastosowania algorytmu Euklidesa do wykonywania działań na ułamkach zwykłych.
- Nauczysz się stosować funkcje języka Python, które nie zwracają wartości.

14.1. Patchworkowe kołdry

Warto wiedzieć

Jeszcze na początku XX w. w krajach anglosaskich patchworkowe kołdry były elementem posagu, który panna młoda wnosiła do domu męża.

Kołdra patchworkowa powstaje ze zszycia wielu mniejszych kawałków tkanin. Wyobraź sobie, że przygotowujesz projekt małej kołdry o wymiarach 112 × 88 cm, która ma być uszyta z jak największych kwadratowych kawałków równej wielkości. Jakiej długości bok powinien mieć taki kawałek?



Rys. 14.1. Patchworkowa kołdra uszyta z kwadratów

Zauważ, że długość boku takiego kwadratu musi być największą możliwą liczbą, która dzieli liczby wyrażające długość i szerokość całej kołdry. Z matematycznego punktu widzenia liczba określająca długość boku największego możliwego kwadratu to największy wspólny dzielnik liczb 112 i 88. Dla kołdry o wymiarach 112 × 88 cm długość boku kawałka materiału powinna wynosić 8 cm, ponieważ $88 = 8 \cdot 11$ oraz $112 = 8 \cdot 2 \cdot 7$.

Ćwiczenie 1

Wyjaśnij, dlaczego podane rozkłady liczb 112 i 88 na czynniki jednoznacznie wskazują, że największy kwadratowy kawałek materiału ma bok długości 8 cm.

14.2. Największy wspólny dzielnik

Przypomnijmy, że **największy wspólny dzielnik** (w skrócie: **NWD**) dwóch liczb całkowitych dodatnich a i b to największa liczba całkowita, która jest dzielnikiem obu tych liczb. Jeśli a jest dzielnikiem b, to liczba a jest największym wspólnym dzielnikiem liczb a i b.

Największy wspólny dzielnik (NWD)

W większości przypadków nie jest łatwo natychmiast wskazać największy wspólny dzielnik konkretnych dwóch liczb. Dla par liczb dwu- albo trzycyfrowych można rozłożyć na czynniki każdą z liczb i wyznaczyć iloczyn wspólnych czynników obu rozkładów. Tę metodę zastosowaliśmy w przykładzie z kołdrą. Jednak już dla liczb większych niż 1000 byłoby to bardzo czasochłonne.

Warto wiedzieć

Rozkład liczb o setkach cyfr na czynniki pierwsze jest problemem trudnym obliczeniowo również dla współczesnych komputerów. Nie istnieje efektywny algorytm, który pozwalałby na znalezienie takiego rozkładu w rozsądnym czasie.

Algorytm naiwny wyznaczania NWD

Zastanówmy się, czy można znaleźć NWD bez potrzeby rozkładania liczb na czynniki pierwsze.

Zapiszmy specyfikację problemu nieco ogólniej:

Specyfikacja

Dane: liczby całkowite dodatnie a i b.

Wynik: największy wspólny dzielnik liczb a i b.

Jedno z rozwiązań może być następujące: będziemy szukać wspólnego dzielnika liczb a i b wśród coraz mniejszych liczb, zaczynając od dzielnika o wartości równej mniejszej z tych liczb. Poszukiwanie NWD zakończy się najpóźniej w momencie, gdy wspólny dzielnik będzie równy 1.

Warto wiedzieć

O dwóch liczbach, które nie mają wspólnego dzielnika większego od 1, mówi się, że są względnie pierwsze.

Algorytm, na podstawie którego stworzymy program wyznaczający NWD, można zapisać w postaci następującej listy kroków:

1. Jeśli a > b, to zamień wartości a i b miejscami.
2. Dla wartości d równych kolejno a, a - 1, a - 2, …, 1 powtarzaj kroki 3, 4 i 5.
   3. Wyznacz resztę z dzielenia a przez d.
   4. Wyznacz resztę z dzielenia b przez d.
   5. Jeśli obie reszty wyznaczone w krokach 3 i 4 są równe 0, to zwróć wartość d i zakończ algorytm.

Poniżej znajduje się tekst programu komputerowego, który dla dwóch liczb całkowitych wczytanych z klawiatury wyznacza NWD tą metodą.

Kod źródłowy programu wyznaczającego NWD dwóch liczb

```
1.  def NWD(a, b):
2.      if a > b:
3.          a, b = b, a
4.      for d in range(a, 0, -1):
5.          if a % d == 0 and b % d == 0:
6.              return d
7.
8.  a = int(input("Podaj liczbę a: "))
9.  b = int(input("Podaj liczbę b: "))
10.
11. print("NWD(" + str(a) + "," + str(b) + ")= ", NWD(a, b))
```

Dobra rada

Pamiętaj, że interpreter języka Python rozróżnia duże i małe litery. Jeśli stosujesz zmienną lub funkcję zawierającą w nazwie duże litery, to w dalszej części kodu musisz konsekwentnie używać ustalonej nazwy.

Funkcja o nazwie NWD zapisana jest w wierszach 1–6. W wierszu 3 znajduje się kod służący temu, aby zawsze w zmiennej b zapisana była większa z wartości, a w zmiennej a – mniejsza. Jeśli tak nie jest, wartości zmiennych są zamieniane. Zastosowaliśmy skrótowy zapis operacji **zamiany wartości zmiennych**, opisanej w temacie 10.

Zamiana wartości zmiennych, s. 185

Kod realizujący kluczową ideę algorytmu znajduje się w wierszach 4–6. W zmiennej d będziemy przechowywać aktualną wartość potencjalnego wspólnego dzielnika. Pętla **for** zakończy działanie w momencie znalezienia wspólnego dzielnika liczb a i b (zapisano to w warunku logicznym instrukcji **if**). Wówczas do programu głównego zwracana jest wartość d. Ponieważ funkcja range generuje wartości zmiennej d zmniejszane przy każdej kolejnej iteracji o jeden, znaleziony dzielnik na pewno jest największym wspólnym dzielnikiem liczb a i b.

Rysunek 14.2 przedstawia przykład wywołania tego programu:

```
Wykonywanie: nwd.py
Podaj liczbę a: 112
Podaj liczbę b: 88
NWD(112,88) = 8
>>>
Python
```

Rys. 14.2. Wywołanie programu wyznaczającego NWD dwóch liczb

Zauważ, że w wierszu 11 przeplatają się napisy stałe oraz wartości zmiennych liczbowych zamienione na napisy z użyciem wbudowanej w język Python **funkcji str**, która przekształca dane (np. liczby) na napisy (łańcuchy znaków). W tym wierszu wywołana jest również funkcja NWD, zdefiniowana na początku programu.

Funkcja str

Do tej pory w naszych programach podawaliśmy wyniki, korzystając z funkcji `print`, w której argumentami były zarówno napisy, jak i wartości zmiennych lub funkcji. Argumenty te rozdzielaliśmy przecinkiem. W tym przypadku wszystkie wartości liczbowe zamieniamy na napisy, a następnie łączymy z wykorzystaniem znaku +, który jest **operatorem scalania napisów** w języku Python. Dzięki takiemu rozwiązaniu mamy kontrolę nad odstępami, które są wyświetlane przed nawiasami i po nich (rys. 14.2).

Operator scalania napisów

Ćwiczenie 2

a. Zapisz kod źródłowy programu *nwd.py* i uruchom program dla różnych par liczb wpisanych z klawiatury.

b. Sprawdź działanie instrukcji z wiersza 11 wypisującej wynik bez zastosowania funkcji `str`, np.:

```
print("NWD(", a, ",", b, ") =", NWD(a,b))
```

Jaką różnicę dostrzegasz?

Warto wiedzieć

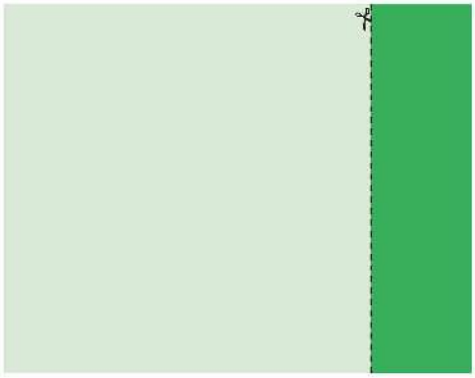
Funkcja `print` wywołana dla kilku argumentów oddzielonych przecinkami domyślnie wstawia między tymi argumentami odstępy (spacje). Na przykład dla `print(1, 2, 3)` zobaczymy na ekranie: 1 2 3.

Zapamiętaj

Funkcja `str` służy do przekształcania danych na napisy (ciągi znaków). Operator + pozwala na scalanie napisów i może być wykorzystany np. jako element funkcji `print`.

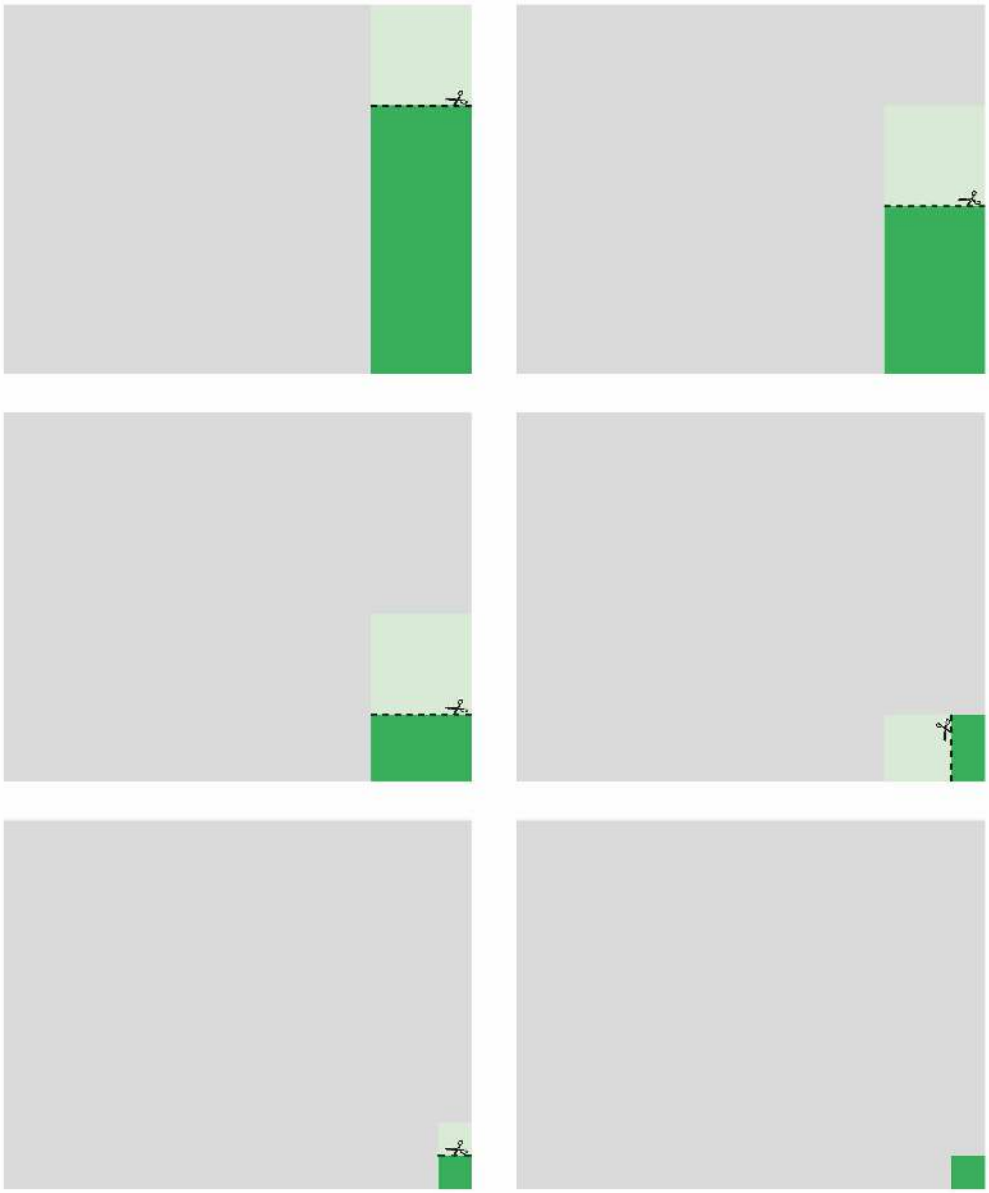
Wycinanka Euklidesa

Wróćmy do przykładu wyznaczania długości boku największego kwadratowego kawałka kołdry o wymiarach 112 × 88 cm. Jak można by możliwie szybko wyznaczyć wymiary takiego kwadratu bez rozkładania liczb 112 i 88 na czynniki pierwsze?



Rys. 14.3. Prostokąt o wymiarach 112 na 88 z odciętym kwadratem o boku 88

Rysunek 14.3 przedstawia pierwszy krok procedury, którą nazwiemy wycinanką Euklidesa. Krok po kroku będziemy odcinać jak największy kwadrat od prostokąta, który powstał w poprzednim kroku. Po pewnej liczbie kroków otrzymamy prostokąt, który jest kwadratem. Kolejne kroki pokazano na rysunku 14.4 i opisano w tabeli 14.1 na s. 234.



Warto wiedzieć

Odcinanie jak największego kwadratu od danego prostokąta to przykład zastosowania strategii algorytmicznej nazywanej podejściem zachłannym.

Rys. 14.4. Kolejne kroki wycinanki Euklidesa

Wymiary prostokąta	Długość boku kwadratu
112 × 88	88
88 × 24	24
64 × 24	24
40 × 24	24
24 × 16	16
16 × 8	8
8 × 8	koniec

Tabela 14.1. Wymiary prostokątów i boków kwadratów w kolejnych krokach wycinanki

Ćwiczenie 3

Zastosuj procedurę wycinania w prostokącie o bokach 51 i 30. Użyj kartki w kratkę i nożyczek.

Dzięki wycinance Euklidesa wiemy, że kwadratami o boku 8 można wypełnić cały prostokąt o wymiarach 112 × 88. Czy można by go wypełnić kwadratami o większym boku? Istnieje dowód matematyczny na to, że nie znajdziemy takiego kwadratu. Dowód ten opiera się na następującym spostrzeżeniu: jeśli prostokąt o bokach a i b (gdzie $a < b$) da się wypełnić kwadratami o boku d, to prostokąt o bokach a i $b - a$ także da się wypełnić kwadratami o boku d.

To oznacza, że każdy z prostokątów odcinanych w wycinance Euklidesa dałby się wypełnić kwadratami o boku większym niż 8, gdyby istniał taki kwadrat dla prostokąta o wymiarach 112 × 88. Tymczasem prostokąta o wymiarach 16 × 8, uzyskanego w przedostatnim kroku procedury, nie da się wypełnić kwadratami o boku dłuższym niż 8.

Spostrzeżenie zapisane powyżej można przedstawić w postaci równości: $NWD(a, b) = NWD(a, b - a)$ dla $a < b$. Ta równość jest podstawą algorytmu Euklidesa.

Warto wiedzieć

Jeśli pewna liczba dzieli bez reszty dwie liczby, to dzieli bez reszty również ich różnicę. Szkic dowodu można napisać tak:
Niech $a < b$ oraz $a = n \cdot d$ i $b = m \cdot d$, gdzie n i m to liczby całkowite dodatnie. Wówczas:
$b - a = m \cdot d - n \cdot d =$
$= (m - n) \cdot d$,
czyli d dzieli różnicę $b - a$.

Algorytm Euklidesa

Algorytm naiwny ze s. 231 zastosowany dla pary liczb 112 i 88 sprawdzi każdą z liczb: 88, 87, 86, ..., 9, 8, a więc wykona ponad 80 dzieleń. Dla komputera to niedużo – efekt wyświetli się niemal natychmiast. Jednak jeśli będziemy sprawdzać parę liczb wielocyfrowych, np. 100 000 014 oraz 99 999 999, wynik na ekranie komputera nie pokaże się od razu, gdyż miliard operacji dzielenia jest już poważnym obciążeniem dla komputera i wymaga dłuższego czasu pracy procesora.

Ćwiczenie 4

Uruchom program z ćwiczenia 2 i porównaj czas jego działania dla dwóch par liczb: 112 i 88 oraz 100 000 014 i 99 999 999. Co zauważasz? Jak można to wytłumaczyć?

Skorzystamy z obserwacji wynikających z wycinanki i stworzymy program realizujący algorytm opisany w starożytności przez Euklidesa w jego słynnym dziele *Elementy*.

Algorytm Euklidesa można zapisać w postaci następującej listy kroków:

Algorytm Euklidesa

1. Dopóki a ≠ b, powtarzaj kroki 2 i 3.
   2. Jeśli a > b, to a = a - b.
   3. W przeciwnym razie b = b - a.
4. Zwróć wartość a.

Poniżej znajduje się kod funkcji NWD, który jest realizacją algorytmu Euklidesa.

Kod źródłowy funkcji realizującej algorytm Euklidesa

```
1.  def NWD(a, b):
2.      while a != b:
3.          if a > b:
4.              a = a - b
5.          else:
6.              b = b - a
7.      return a
```

Po zakończeniu działania pętli **while** wartości obu zmiennych a i b są równe największemu wspólnemu dzielnikowi początkowych wartości a i b. Dlatego funkcja NWD zwraca jedną z tych wartości (linia 7). W przypadku liczb, które nie mają wspólnego dzielnika większego niż 1, funkcja zwraca wartość 1.

Warto wiedzieć

Liczby współmierne to takie, dla których można wyznaczyć wspólną miarę, czyli liczbę dzielącą obie liczby bez reszty.

Czy dla dowolnej pary liczb a i b w skończonej liczbie kroków zostanie osiągnięty warunek wyjścia z pętli (czyli a będzie równe b)? Od czasów Euklidesa wiemy, że stanie się tak wtedy i tylko wtedy, gdy a i b będą liczbami współmiernymi.

Ćwiczenie 5

a. Zmodyfikuj kod źródłowy programu *nwd.py*, używając zapisanej wyżej funkcji NWD, i zapisz go pod nazwą *euklides.py*. Uruchom program dla różnych par liczb wpisanych z klawiatury.

b. Porównaj czas działania programu dla dwóch par liczb: 112 i 88 oraz 100 000 014 i 99 999 999. Czy zauważasz różnicę w działaniu algorytmu naiwnego i algorytmu Euklidesa?

A to ciekawe

Matematyczny bestseller wszech czasów

Elementy Euklidesa to traktat matematyczny, który opisuje osiągnięcia starożytnej geometrii. Jest on uznawany za dzieło wyjątkowe, ponieważ zastosowano w nim podejście dedukcyjne (od ogółu do szczegółu), precyzję definicji i dowodów twierdzeń. W traktacie znalazł się również algorytm, który jest wykorzystywany we współczesnej informatyce. Mimo że dzieło Euklidesa powstało około 300 r. p.n.e., używano go w kolejnych przekładach jako podręcznika do geometrii przez niemal 2000 lat. Uznaje się, że *Elementy* są drugą po Biblii książką najczęściej drukowaną i tłumaczoną w historii świata.



Algorytm Euklidesa – wersja z dzieleniem

Zauważ, że jeśli w algorytmie Euklidesa jedna liczba jest dużo większa od drugiej, to wielokrotnie powtarza się odejmowanie od niej tej samej liczby. W tabeli 14.2 przedstawiono znajdowanie NWD liczb 196 i 42.

Wartość zmiennej a	Wartość zmiennej b
196	42
196 – 42 = 154	42
154 – 42 = 112	42
112 – 42 = 70	42
70 – 42 = 28	42
28	42 – 28 = 14
28 – 14 = 14	14

Tabela 14.2. Wyznaczanie NWD zgodnie z algorytmem Euklidesa

Wielokrotne odejmowanie możemy zastąpić wyznaczeniem reszty z dzielenia większej liczby przez mniejszą – tak jak w tabeli 14.3.

Wartość zmiennej a	Wartość zmiennej b
196	42
196 % 42 = 28	42
28	42 % 28 = 14
28 % 14 = 0	14

Tabela 14.3. Wyznaczanie NWD zgodnie z algorytmem Euklidesa, w którym kolejne odejmowania zastąpiono resztą z dzielenia

Wynikiem dzielenia 196 przez 42 jest 4 i reszta 28. Część całkowita określa liczbę operacji odejmowania, a reszta z dzielenia – wynik ostatniego odejmowania. W tym przypadku cztery działania zastąpiliśmy jednym.

Zwróć uwagę na ostatni wiersz tabeli 14.3. Inny jest tym razem warunek zakończenia działania algorytmu. Nie otrzymujemy dwóch takich samych wartości – jedna ze zmiennych przyjmuje wartość 0. Przed ostatnim dzieleniem większa liczba jest wielokrotnością mniejszej, a więc ta mniejsza jest największym wspólnym dzielnikiem.

Oto lista kroków **algorytmu Euklidesa w wersji z dzieleniem**:

1. Dopóki a ≠ 0 i b ≠ 0, powtarzaj kroki 2 i 3.
   2. Jeśli a > b, to a = a % b.
   3. W przeciwnym razie b = b % a.
4. Jeśli a ≠ 0, to zwróć a.
5. W przeciwnym razie zwróć b.

◀ Algorytm Euklidesa – wersja z dzieleniem

Warto wiedzieć

Operacja dzielenia, a więc i modulo, pochłania stosunkowo dużo czasu pracy procesora. Dlatego w najszybszych implementacjach algorytmu Euklidesa dzielenie z resztą zastępuje się odejmowaniem oraz połowieniem (przesunięciami bitów o jeden w prawo), które procesory wykonują szybko. W ten sposób NWD można wyznaczyć ok. 60% szybciej.

Poniżej znajduje się kod funkcji NWD, który jest realizacją algorytmu Euklidesa w wersji z dzieleniem.

Kod źródłowy funkcji realizującej algorytm Euklidesa w wersji z dzieleniem

```
1.  def NWD(a, b):
2.      while a != 0 and b != 0:
3.          if a > b:
4.              a = a % b
5.          else:
6.              b = b % a
7.      if a != 0:
8.          return a
9.      else:
10.         return b
```

Ćwiczenie 6

a. Zmodyfikuj kod programu *euklides.py* z ćwiczenia 5, używając funkcji NWD, i zapisz go pod nazwą *euklides_dzielenie.py*. Uruchom program dla różnych par liczb wpisanych z klawiatury.
b. Dodaj do kodu instrukcję zliczającą wykonane iteracje.
c. Dołącz do kodu funkcję NWD (w wersji z odejmowaniem). Zmodyfikuj kod programu tak, aby porównywał liczbę iteracji wykonanych w obu wersjach.

Zapamiętaj

Największy wspólny dzielnik dwóch liczb naturalnych można wyznaczyć bez konieczności szukania wspólnych dzielników tych liczb. Należy zastosować algorytm Euklidesa.

14.3. Działania na ułamkach zwykłych

W szkole na lekcjach różnych przedmiotów wielokrotnie wykonywaliście podstawowe działania arytmetyczne na ułamkach, m.in. dodawanie i mnożenie. Napiszemy teraz programy, które pozwolą wykonywać obliczenia dla ułamków o wartościach dodatnich.

Licznik i mianownik ułamka będziemy reprezentować w postaci pary liczb całkowitych.

Dodawanie ułamków

Żeby dodać dwa ułamki zwykłe, wystarczy pomnożyć ich mianowniki i dodać liczniki mnożone odpowiednio przez mianowniki:

$$\frac{a}{b}+\frac{c}{d}=\frac{a\cdot d+c\cdot b}{b\cdot d}=\frac{e}{f}$$

Aby doprowadzić ułamek do postaci nieskracalnej, wystarczy podzielić liczby e i f przez NWD(e,f).

Oto specyfikacja problemu dodawania dwóch ułamków:

Specyfikacja

Dane: cztery liczby całkowite – licznik i mianownik jednego ułamka oraz licznik i mianownik drugiego ułamka.

Wynik: dwie liczby całkowite – licznik i mianownik sumy ułamków, zapisane w postaci nieskracalnej.

Poniżej znajduje się kod źródłowy programu *Suma ułamków*, który realizuje powyższą specyfikację.

Kod źródłowy programu *Suma ułamków*

```
1.  def NWD(a, b):
2.      while a != 0 and b != 0:
3.          if a > b:
4.              a = a % b
5.          else:
6.              b = b % a
7.      if a != 0:
8.          return a
9.      else:
10.         return b
11.
12. def suma(a, b, c, d):
13.     e = a * d + c * b
14.     f = b * d
15.     nwd_ef = NWD(e, f)
16.     print(str(e // nwd_ef) + "/" + str(f // nwd_ef))
17.
18. licz_1 = int(input("Liczba 1 - licznik: "))
19. mian_1 = int(input("Liczba 1 - mianownik: "))
20. licz_2 = int(input("Liczba 2 - licznik: "))
21. mian_2 = int(input("Liczba 2 - mianownik: "))
22.
23. print("\n" + str(licz_1) + "/" + str(mian_1) + " + ", end="")
24. print(str(licz_2) + "/" + str(mian_2) + " = ", end="")
25. suma(licz_1, mian_1, licz_2, mian_2)
```

W liniach 1–10 zdefiniowano funkcję wyznaczającą NWD zgodnie z algorytmem Euklidesa (w wersji z dzieleniem), a w liniach 12–16 zdefiniowano funkcję `suma`, która jest **funkcją niezwracającą wartości**, ponieważ nie zawiera instrukcji **`return`**. Zadaniem tej funkcji jest wykonanie obliczeń i wyświetlenie efektu na ekranie. Dlatego też w linii 25 znajduje się wywołanie funkcji `suma` bez funkcji `print`.

Funkcja niezwracająca wartości

Ćwiczenie 7

Zapisz kod źródłowy ze s. 239 w pliku *suma_ulamkow.py* i uruchom program dla różnych par ułamków.

Dodawanie ułamków – metoda szkolna

Kiedy na lekcjach matematyki dodawaliście ułamki, sprowadzaliście je najpierw do wspólnego mianownika, którym była najmniejsza wspólna wielokrotność mianowników ułamków. Na przykład:

$$\frac{5}{12}+\frac{5}{18}+\frac{5\cdot\frac{36}{12}+5\cdot\frac{36}{18}}{36}=\frac{15+10}{36}=\frac{25}{36}$$

Najmniejsza wspólna wielokrotność (NWW)

Przypomnijmy, że **najmniejsza wspólna wielokrotność** (w skrócie: **NWW**) dwóch liczb całkowitych dodatnich a i b to najmniejsza liczba naturalna, która jest podzielna przez każdą z tych liczb. Jeśli liczby a i b są względnie pierwsze, to $\mathrm{NWW}(a, b) = a \cdot b$.

Zależność między NWD a NWW

Okazuje się, że prawdziwa jest równość pokazująca **zależność między NWD a NWW** liczb a i b:

$$\mathrm{NWW}(a,b)=\frac{a\cdot b}{\mathrm{NWD}(a,b)}$$

Można to wyjaśnić następująco: jeśli a i b mają wspólne dzielniki, to w iloczynie $a \cdot b$ każdy z nich występuje dwukrotnie – raz w a, a raz w b. Zatem aby uzyskać $\mathrm{NWW}(a, b)$, należy iloczyn $a \cdot b$ podzielić przez iloczyn wspólnych dzielników, którym jest $\mathrm{NWD}(a, b)$.

Aby dodać dwa ułamki $\frac{a}{b}$ oraz $\frac{c}{d}$, gdzie a i b oraz c i d są liczbami względnie pierwszymi, można wykorzystać następujący wzór:

$$\frac{a}{b}+\frac{c}{d}=\frac{a\cdot\frac{\mathrm{NWW}(b,d)}{b}+c\cdot\frac{\mathrm{NWW}(b,d)}{d}}{\mathrm{NWW}(b,d)}=\frac{e}{f}$$

Ćwiczenie 8

a. Zgodnie z powyższym wzorem dodaj ułamki $\frac{5}{12}$ i $\frac{3}{20}$.

b. Czy po zastosowaniu tego wzoru licznik i mianownik sumy dwóch ułamków (liczby e i f) zawsze będą względnie pierwsze?

Oto zapis kodu funkcji suma, dodającej ułamki metodą szkolną:

Kod źródłowy funkcji dodającej ułamki metodą szkolną

```
1.  def suma(a, b, c, d):
2.      NWW = b * d // NWD(b, d)
3.      e = a * NWW // b + c * NWW // d
4.      f = NWW
5.      print(str(e) + "/" + str(f))
```

Zauważ, że w kodzie korzystamy z algorytmu Euklidesa zastosowanego w funkcji `NWD` oraz z wzoru określającego zależność między NWD i NWW (kod funkcji `NWD` – s. 239).

Ćwiczenie 9

W pliku *suma_ulamkow2.py* zapisz kod programu dodającego ułamki metodą szkolną. Możesz wykorzystać plik *suma_ulamkow.py* z ćwiczenia 7. Sprawdź działanie programu dla różnych par ułamków.

Mnożenie ułamków

Aby pomnożyć dwa ułamki, możemy pomnożyć przez siebie liczniki i mianowniki, a następnie skrócić otrzymany wynik.

Żeby wykonywać operacje na mniejszych liczbach, skracamy najpierw ułamki „na krzyż" (tak samo postępujecie na lekcjach matematyki):

$$\frac{a}{b} \cdot \frac{c}{d} = \frac{\frac{a}{\mathrm{NWD}(a,d)} \cdot \frac{c}{\mathrm{NWD}(b,c)}}{\frac{\mathrm{b}}{\mathrm{NWD}(b,c)} \cdot \frac{d}{\mathrm{NWD}(a,d)}} = \frac{e}{f}$$

Zapiszmy specyfikację problemu:

Specyfikacja

Dane: cztery liczby całkowite – licznik i mianownik jednego ułamka oraz licznik i mianownik drugiego ułamka.

Wynik: dwie liczby całkowite – licznik i mianownik iloczynu ułamków, zapisane w postaci nieskracalnej.

Oto zapis kodu funkcji `iloczyn`, realizującej algorytm mnożenia ułamków z wykorzystaniem NWD:

Kod źródłowy funkcji mnożącej ułamki

```
1.  def iloczyn(a, b, c, d):
2.      e = a // NWD(a, d) * c // NWD(b, c)
3.      f = b // NWD(b, c) * d // NWD(a, d)
4.      nwd_ef = NWD(e, f)
5.      print(str(e // nwd_ef) + "/" + str(f // nwd_ef))
```

Ćwiczenie 10

W pliku *iloczyn_ulamkow.py* zapisz kod źródłowy programu mnożącego ułamki z wykorzystaniem NWD. Sprawdź jego działanie dla różnych par ułamków.

Podsumowanie

- Algorytm Euklidesa służy do znajdowania największego wspólnego dzielnika (NWD) dwóch liczb naturalnych. W algorytmie tym można wykorzystać odejmowanie – wtedy jeśli od większej liczby odejmiemy mniejszą, to mniejsza liczba i otrzymana różnica będą miały NWD taki jak pierwotne liczby. Ten schemat powtarzamy tak długo, aż liczby się zrównają – będą one równe szukanemu NWD.
- W algorytmie Euklidesa zamiast odejmowania można wykorzystać dzielenie. W tej wersji algorytmu wyznaczamy resztę z dzielenia większej liczby przez mniejszą. Działanie to wykonujemy, aż reszta będzie równa 0. Wtedy mniejsza liczba (dzielnik) jest poszukiwanym NWD.
- Liczbę wymierną można reprezentować w programie jako dwie zmienne całkowite określające odpowiednio licznik i mianownik ułamka.
- W działaniach na liczbach wymiernych wykorzystujemy algorytm Euklidesa do skracania ułamka oraz sprowadzania ułamków do wspólnego mianownika.
- Funkcja bez instrukcji `return` nie zwraca żadnego wyniku, wykonuje tylko określone czynności.

Zadania

* **1** Napisz program, który po wczytaniu wymiarów patchworkowej kołdry (w centymetrach) wyznaczy liczbę kawałków materiału potrzebnych do jej wykonania. Kołdra ma się składać z jak największych kwadratów tej samej wielkości.

* **2** Napisz program, który z ułamka niewłaściwego wyodrębni część całkowitą i część ułamkową.

* **3** Napisz program, który – korzystając z algorytmu Euklidesa – będzie sprawdzać, czy dane dwie liczby naturalne są liczbami względnie pierwszymi.

* **4** Napisz program, który będzie wczytywać licznik i mianownik ułamka o wartości mniejszej niż 1/2 i sprawdzać, ile razy ten ułamek mieści się w liczbie 1.

* **5** Napisz program, który wczyta dwa ułamki i wyświetli większy z nich wraz z komunikatem „jest większą liczbą". Jeśli użytkownik wprowadzi te same ułamki, program wyświetli komunikat „Ułamki są równe".

** **6** Napisz program, który wczytuje dwie dodatnie i względnie pierwsze liczby a i b, a następnie wyznacza dla nich najmniejszą możliwą liczbę x, dla której ułamek $\frac{a+x}{b+x}$ jest nieskracalny.

** **7** Napisz program, który wczyta trzy liczby całkowite dodatnie i wyznaczy ich największy wspólny dzielnik.
Wskazówka: Największy wspólny dzielnik trzech liczb to największy wspólny dzielnik jednej z liczb i liczby, która jest największym dzielnikiem pozostałych.

** 8 Zmodyfikuj program *Suma ułamków* tak, aby można było w nim dodawać trzy ułamki.

** 9 Uzupełnij program *Suma ułamków* o wyodrębnianie części całkowitej i części ułamkowej dla sumy, która jest ułamkiem niewłaściwym.

** 10 Napisz program zawierający funkcję o nagłówku `NWW(a, b)`, która będzie wyznaczać najmniejszą wspólną wielokrotność liczb a i b.

** 11 Zapisz specyfikację problemu dzielenia dwóch ułamków. Następnie napisz listę kroków algorytmu i program zawierający funkcję `iloraz`, wykonującą dzielenie.

*** 12 Napisz w języku Python prostą grę dwuosobową, w której pierwszy z graczy podaje dwie liczby naturalne mniejsze od 200, a drugi gracz próbuje odgadnąć ich NWD. Kolejne próby powinny być numerowane. Po odgadnięciu gracze zamieniają się rolami. Wygrywa ten, który poda poprawny wynik w mniejszej liczbie prób.

*** 13 Zapisz specyfikację problemu odejmowania dwóch ułamków. Następnie napisz listę kroków algorytmu i program zawierający funkcję `roznica`, odejmującą dwa ułamki. Umieść w nim również podprogram, który przetestuje działanie tej funkcji.

*** 14 Udowodnij prawdziwość wzoru na mnożenie ułamków, w którym skracamy ułamki „na krzyż".

*** 15 Napisz program, który spośród pięciu różnych dodatnich liczb całkowitych podanych przez użytkownika wybierze takie dwie, które będą stanowiły licznik i mianownik ułamka o możliwie najmniejszej wartości. Rozwiązanie przygotuj dwoma sposobami:
a. z wykorzystaniem listy pięcioelementowej,
b. bez korzystania z listy.

*** 16 Napisz program, który wczyta trzy dodatnie liczby jednocyfrowe a, b i c takie, że $a \neq b$. Efektem powinna być informacja o położeniu cyfry c w rozwinięciu dziesiętnym ułamka a/b. Jeśli taka cyfra się nie pojawia, program powinien wypisać liczbę –1.

*** 17 Napisz program, który zrealizuje algorytm Herona według listy kroków podanej na s. 176.

Algorytmika i programowanie w języku Python

Wyobraź sobie, że...



...zaczynasz studia na politechnice, ponieważ chcesz zostać inżynierem geodetą. Okazuje się, że na pierwszym roku masz zajęcia z programowania. Na szczęście w szkole poznałaś **podstawowe konstrukcje programistyczne**, **typy danych** i **proste struktury danych** oraz zdobyłaś pewne doświadczenie w programowaniu w **środowisku tekstowym**. Radzisz sobie, rozumiesz omawiane algorytmy. Potrafisz je też zaprogramować, mimo że używasz innego języka programowania niż w szkole.



...jesteś inżynierem pożarnictwa i pracujesz w firmie opracowującej zasady bezpieczeństwa dla obiektów biurowych. Wiesz, że instrukcje zawierające sposób postępowania na wypadek pożaru lub innego zagrożenia muszą być bardzo precyzyjne. Jednym ze sposobów ich zapisania jest forma **listy kroków**, z którą zetknąłeś się w szkole podczas nauki programowania. Twoja praca cieszy się dużym uznaniem, ponieważ instrukcje tworzone przez ciebie są dla wszystkich zrozumiałe.



...skończyłaś studia medyczne i odbyłaś staż. Trafiasz na ofertę pracy w projekcie badawczym, w ramach którego ma powstać oprogramowanie wykorzystywane do kształcenia przyszłych lekarzy w Afryce. W czasie rozmowy kwalifikacyjnej sprawdzana jest m.in. znajomość podstawowych pojęć związanych z projektowaniem oprogramowania, takich jak **sytuacja problemowa** i **specyfikacja problemu**. Dostajesz pracę. Współpracując z programistami, możesz zmieniać świat na lepsze.



...pracujesz w firmie produkującej meble, która wdraża system wsparcia produkcji wyposażony w sztuczną inteligencję. Na szkoleniu pokazano model sztucznego neuronu wraz z **kodem źródłowym**. Na jego podstawie wyjaśniono podstawę działania całego systemu. Jesteś zachwycony, ponieważ do zrozumienia kodu wystarczyła wiedza o **instrukcji warunkowej** i **pętlach** wyniesiona ze szkoły.

Dodatki

1. Podstawowe funkcje programu Microsoft Excel

Każda formuła w arkuszu kalkulacyjnym musi się zaczynać od znaku =. W arkuszu można wykorzystywać standardowe symbole +, –, *, /, ^, które pełnią funkcję operatorów dodawania, odejmowania, mnożenia, dzielenia i potęgowania.
Znak : pozwala na określenie zakresu komórek, np. zakres od A1 do A10 to A1:A10.

Nazwa funkcji	Przykład zastosowania	Opis
SUMA	SUMA(A1:A8) SUMA(A1;A2;A3)	Umożliwia dodawanie wartości w komórkach.
ILOCZYN	ILOCZYN(A1:A4)	Oblicza iloczyn liczb ze wskazanych komórek lub podanych bezpośrednio w formule.
PIERWIASTEK	PIERWIASTEK(9)	Zwraca pierwiastek kwadratowy liczby.
POTĘGA	POTĘGA(A1;B1)	Zwraca wartość liczby podniesionej do potęgi.
MODUŁ.LICZBY	MODUŁ.LICZBY(A15)	Zwraca wartość bezwzględną liczby.
PI	PI()	Zwaraca wartość liczby π zaokrągloną do 15 miejsc po przecinku.
SILNIA	SILNIA(A1)	Wylicza wartość silni liczby.
DZIES.NA.DWÓJK	DZIES.NA.DWÓJK(25)	Zamienia liczbę dziesiętną na binarną.
DWÓJK.NA.DZIES	DWÓJK.NA.DZIES(B2)	Zamienia liczbę binarną na dziesiętną.
ŚREDNIA	ŚREDNIA(B5:B10)	Oblicza wartość średniej arytmetycznej podanych argumentów.
MEDIANA	MEDIANA(A1:A100)	Oblicza medianę, czyli wartość środkową dla podanego zakresu liczb.
MAX	MAX(A2:A20)	Zwraca największą wartość ze zbioru wartości.
MIN	MIN(A2:A30)	Zwraca najmniejszą wartość ze zbioru wartości.
LOS	LOS()	Zwraca liczbę losową z przedziału [0; 1].
LOS.ZAKR	LOS.ZAKR(1;100)	Zwraca losową liczbę całkowitą z zadanego zakresu liczb.
DZIŚ	DZIŚ()	Zwraca bieżącą datę.
DZIEŃ.TYG	DZIEŃ.TYG(DZIŚ())	Zwraca liczbę od 1 do 7 określającą numer dnia tygodnia na podstawie daty.
ORAZ	ORAZ(A1>1;A2<100)	Funkcja logiczna, której argumenty są typu logicznego. Zwraca wartość PRAWDA tylko wtedy, gdy wszystkie argumenty mają wartość PRAWDA. W przeciwnym wypadku zwraca wartość FAŁSZ.

LUB	LUB(B1>1;B2<100)	Funkcja logiczna, której argumenty są typu logicznego. Zwraca wartość PRAWDA, jeśli przynajmniej jeden z argumentów ma wartość PRAWDA. W przeciwnym wypadku zwraca wartość FAŁSZ.
JEŻELI	JEŻELI(A1>0;A2;A3)	Funkcja, w której zwracana wartość zależy od spełnienia podanego warunku, np. jeśli wartość w komórce A1 jest większa od 0, zwracana jest wartość komórki A2, w przeciwnym razie A3.
LICZ.JEŻELI	LICZ. JEŻELI(A2:A15;6)	Oblicza, ile komórek we wskazanym zakresie spełnia dane kryterium, np. ile razy w zakresie A2 do A15 występuje wartość 6.
JEŻELI.BŁĄD	JEŻELI.BŁĄD(A4/B4; "Nie można dzielić przez 0")	Zwraca wartość podaną jako pierwszy parametr, a jeśli wystąpi błąd, wykonuje inną operację, np. zwraca odpowiedni tekst.
ZŁĄCZ.TEKSTY	ZŁĄCZ.TEKSTY(A1;" ";A2)	Umożliwia połączenie dwóch lub więcej ciągów tekstowych w jeden ciąg.
INDEKS	INDEKS(A2:B5;2;2)	Zwraca wartość lub odwołanie do wartości znajdującej się w tabeli lub zakresie, np. wartość na przecięciu drugiego wiersza i drugiej kolumny w zakresie od A2 do B5.
ZNAJDŹ	ZNAJDŹ("N";A1)	Zwraca pozycję podanego znaku lub ciągu znaków we wskazanym tekście.
FRAGMENT.TEKSTU	FRAGMENT. TEKSTU(A1;1;5) FRAGMENT. TEKSTU("Nowa Era";6;3)	Zwraca fragment ciągu tekstowego od znaku o podanej pozycji, o określonej liczbie znaków.
PODSTAW	PODSTAW(F8;"a";"b")	Zastępuje stary tekst nowym tekstem w podanym ciągu tekstowym.
KOD	KOD(K10)	Zwraca wartość kodu liczbowego (w systemie znaków ANSI) pierwszego znaku w ciągu tekstowym.
ZNAK	ZNAK(K12)	Zwraca znak o podanym numerze kodu w systemie ANSI.

2. Typy danych i maski w programie Microsoft Access

Wybrane typy danych

Typ danych	Opis
Krótki tekst	Pozwala na przechowywanie 255 znaków alfanumerycznych (np. liter, liczb, znaków przestankowych). Wykorzystywany jest do zapisywania imion, nazwisk, danych teleadresowych i liczb, na których nie wykonuje się obliczeń (np. numerów telefonów).
Długi tekst	Podobnie jak **Krótki tekst** pozwala zapisać teksty, może jednak zawierać do 64 tysięcy znaków.
Liczba	Przechowuje dane liczbowe, na których można wykonywać obliczenia, z wyjątkiem obliczeń walutowych (patrz: typ **Waluta**).
Data/Godzina	Przechowuje informacje na temat czasu: daty i godziny.
Waluta	Przechowuje dane liczbowe z dokładnością do 4 miejsc dziesiętnych i zapobiega zaokrągleniom podczas obliczeń.
Autonumerowanie	Przechowuje unikatową wartość generowaną przez program dla każdego nowego rekordu.
Tak/Nie	Przechowuje wartość logiczną typu prawda (1) lub fałsz (0).
Załącznik	Pozwala dołączać pliki takie jak obrazy, dokumenty tekstowe, arkusze kalkulacyjne lub wykresy. Każde pole typu **Załącznik** może zawierać dowolną liczbę załączników na rekord.
Hiperłącze	Przechowuje adres (ścieżkę) do zasobów znajdujących się w sieci Internet, sieci lokalnej lub na dysku komputera.
Kreator odnośników	Nie jest faktycznym typem danych, pozwala jednak uruchomić kreator, za pomocą którego można wybrać wartości z innej tabeli lub listy.

Wybrane znaki stosowane do definiowania masek wprowadzania

Znak	Opis
0	Użytkownik musi wprowadzić cyfrę.
9	Użytkownik może wprowadzić cyfrę.
L	Użytkownik musi wprowadzić literę.
A	Użytkownik musi wprowadzić cyfrę lub literę.
?	Użytkownik może wprowadzić literę.
#	Użytkownik może wprowadzić cyfrę, spację, znak plus lub minus.
>	Wszystkie znaki po tym symbolu wyświetlane są jako duże litery.
<	Wszystkie znaki po tym symbolu wyświetlane są jako małe litery.
&	Użytkownik musi wprowadzić dowolny znak lub spację.

3. Jak zaprojektować i utworzyć bazę danych

Kiedy tworzysz nową bazę danych, postępuj według następujących kroków.

1. Zastanów się, w jakim celu tworzysz bazę danych, jakie informacje chcesz w niej przechowywać, a także jakie informacje będą potrzebne w przyszłości.
2. Zaplanuj tabele opisujące poszczególne obiekty w twojej bazie. Pamiętaj, że jedna tabela powinna opisywać tylko pojedynczy obiekt – np. adresy, uczniów, sklepy, modele samochodów.
3. W każdej tabeli jedna kolumna powinna opisywać pojedynczą cechę obiektu, np. kod pocztowy, ale nie kod pocztowy i miasto. Imię i nazwisko również wprowadzaj w odrębnych kolumnach.
4. Zaplanuj, jakiego typu dane będą przechowywane w poszczególnych atrybutach (kolumnach) każdej z tabel. Przykładowe typy to: tekst, data, liczba.
5. Upewnij się, że te same dane nie znajdują się w dwóch różnych tabelach lub polach tabeli.
6. Utwórz nową, pustą bazę w programie, z którego korzystasz, np. Microsoft Access.
7. Utwórz tabele według stworzonego planu, nadaj nazwy poszczególnym kolumnom (atrybutom) oraz określ, jakiego typu dane będą w nich przechowywane.
8. Dodaj testowe dane w każdej z tabel.
9. W razie potrzeby wykorzystaj maski wprowadzania danych dla wybranych pól, aby ograniczyć możliwość popełnienia błędu.
10. Utwórz relacje między tabelami, korzystając np. z kreatora odnośników. Przetestuj działanie odnośników: w widoku arkusza danych dodaj, usuń i zmodyfikuj dane w każdej z dostępnych tabel.
11. W razie potrzeby wprowadź modyfikacje dla typów danych przechowywanych w poszczególnych polach tabel i ponownie je przetestuj.
12. Utwórz formularze, kwerendy oraz raporty ułatwiające korzystanie z bazy danych i zarządzanie nią.
13. Zacznij używać swojej bazy.

Wybrane systemy zarządzania bazami danych:

- Microsoft Access (https://products.office.com/pl-pl/access)
- LibreOffice Base (https://www.libreoffice.org/)
- Kexi (http://kexi-project.org/)
- Symphytum (https://github.com/giowck/symphytum)

Wybrane systemy zarządzania bazami danych pracujące w architekturze klient–serwer:

- MySQL
- Microsoft SQL Server
- Oracle
- PostgreSQL

4. Sposoby zapisu algorytmów

Algorytm to opis postępowania prowadzącego do rozwiązania problemu. Istotą algorytmu jest jego uniwersalność, czyli to, że służy do rozwiązywania wielu zadań tego samego rodzaju. Na przykład szkolny algorytm mnożenia pisemnego możesz zastosować dla dowolnych dwóch liczb, np. 21 · 57, 123 · 87. Algorytm musi być precyzyjny, choć może mieć różną formę.

Opis słowny

Algorytmy mogą być opisane słownie. Przykładem może być instrukcja obsługi kuchennego minutnika: „Wykonać pełen obrót wskaźnika aż do pozycji 0. Następnie ustawić wskaźnik na wymaganą liczbę minut".

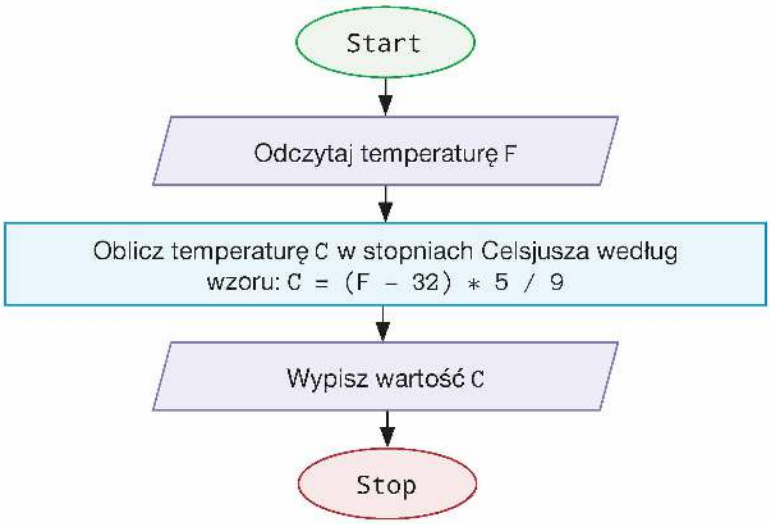
Lista kroków

Lista kroków algorytmu to opis postępowania podzielony na poszczególne kroki, które mogą się odwoływać do siebie nawzajem.
Oto lista kroków algorytmu zaplatania warkocza klasycznego:

1. `Rozdziel włosy na trzy pasma.`
2. `Dopóki nie spleciesz warkocza po końcówki włosów, wykonuj krok` 3 `i krok` 4.
   3. `Przełóż lewe pasmo nad środkowym. Teraz to pasmo staje się środkowym.`
   4. `Przełóż prawe pasmo nad środkowym. Teraz to pasmo staje się środkowym.`

Schemat blokowy (sieć działań)

Schemat blokowy algorytmu jest zbudowany z umownych kształtów oraz strzałek pokazujących kierunek wykonywania operacji. Oto przykład schematu blokowego algorytmu przeliczania temperatury ze skali Fahrenheita na skalę Celsjusza:



Pseudokod (pseudojęzyk)

Pseudokod to zapis przypominający kod języka programowania, ale pozbawiony szczegółów związanych z implementacją w języku programowania. Przedstawienie algorytmu w postaci pseudokodu ułatwia później zapisanie tego algorytmu w języku programowania. Nie istnieje jedna, powszechnie przyjęta konwencja zapisu pseudokodu. Podstawową zasadą jest jednak wybór intuicyjnych oznaczeń i konsekwentne ich wykorzystanie.
Oto pseudokod algorytmu wyznaczania części całkowitej $\sqrt{n}$ dla dodatniej liczby całkowitej n:

```
wczytaj n
p ← 1
dopóki (p + 1) · (p + 1) ≤ n wykonuj p ← p + 1
wypisz p
```

W powyższym zapisie strzałka (←) oznacza instrukcję przypisania wartości do zmiennej. Symbole +, · i ≤ mają takie samo znaczenie jak w matematyce.

Kod źródłowy

Kod źródłowy to sformalizowany zapis algorytmu zgodny z regułami wybranego języka programowania. Kody źródłowe zazwyczaj zapisuje się w plikach tekstowych. Oto przykłady kodów źródłowych algorytmu obliczania całkowitego pierwiastka kwadratowego z liczby n, zapisane w językach C++ i Python 3:

C++

```
1.  int main()
2.  {
3.      int n; cin >> n;
4.      int p = 1;
5.      while ((p + 1)*(p + 1) <= n)
6.      {
7.          p = p + 1;
8.      }
9.      cout<< p <<endl;
10.     return 0;
11. }
```

Python

```
1.  n = int(input())
2.  p = 1
3.  while (p + 1)*(p + 1) <= n:
4.      p = p + 1;
5.  print(p)
```

Języki programowania, podobnie jak języki naturalne, mają słownictwo podstawowe, czyli **słowa kluczowe**, oraz **składnię**, czyli reguły tworzenia wypowiedzi.

Język programowania służy do porozumiewania się człowieka i komputera. W tej komunikacji pośredniczą specjalne **translatory**, czyli programy tłumaczące, gdyż docelowo kod źródłowy musi być przetłumaczony na **kod maszynowy**, zapisany binarnie, właściwy dla architektury procesora (CPU) i systemu operacyjnego.

5. Logika matematyczna i operatory logiczne

Zdania logiczne

Zdaniem logicznym jest każde zdanie, któremu możemy przyporządkować jednoznacznie wartość logiczną, czyli jedną z dwóch możliwych ocen: prawdę (ang. *true*) albo fałsz (ang. *false*).

Przykłady zdań logicznych:

- Liczba 8 jest liczbą pierwszą. (zdanie fałszywe)
- $2 + 2 < 5$ (zdanie prawdziwe)
- $2 + 8 = 20$ (zdanie fałszywe)
- Liczba parzysta jest podzielna przez 2. (zdanie prawdziwe)

Tradycyjnie przyjęło się, że prawdzie odpowiada wartość 1, a fałszowi – wartość 0.

Formy zdaniowe i zdania złożone

Formą zdaniową jest każde zdanie, które zawiera zmienne i które staje się zdaniem logicznym z chwilą zastąpienia zmiennych konkretnymi wartościami. Formie zdaniowej nie można więc jednoznacznie przypisać wartości logicznej, gdyż ta wartość zależy od wyboru wartości dla zmiennych.

Przykłady form zdaniowych:

- Liczba x jest podzielna przez 3. (np. dla $x = 6$ zdanie jest prawdziwe, a dla $x = 7$ – fałszywe)
- $y < 4$ (np. dla $y = 0$ zdanie jest prawdziwe, a dla $y = 4$ – fałszywe)
- Liczba z jest liczbą złożoną. (np. dla $z = 5$ zdanie jest fałszywe, a dla $z = 12$ – prawdziwe)

Zdania logiczne łączymy w zdania złożone, a formy zdaniowe – w formy zdaniowe złożone. Najczęściej używane konstrukcje to: zaprzeczenie (negacja), alternatywa oraz koniunkcja.

Zaprzeczenie (negacja)

Zaprzeczenie to zdanie logiczne, które ma postać: *nieprawda, że p*, gdzie p jest zdaniem logicznym. Negacją może być też forma zdaniowa o postaci *nieprawda, że* $p(x)$, gdzie $p(x)$ jest formą zdaniową. Zaprzeczenie zdania p zapisuje się często jako $\sim p$.

p	$\sim p$
$x < 4$	$x \geqslant 4$
$2 + 2 = 5$	$2 + 2 \neq 5$

Alternatywa

Alternatywą jest połączenie zdań p i q spójnikiem „lub", które ma postać: *p lub q*. Może to być także połączenie form zdaniowych $p(x)$ i $q(x)$ w postaci: $p(x)$ *lub* $q(x)$.

p	q	*p lub q*
$x < 4$	$x > 9$	$x < 4$ lub $x > 9$
y jest liczbą parzystą	z jest liczbą nieparzystą	y jest liczbą parzystą lub z jest liczbą nieparzystą

Zdanie złożone zwane alternatywą jest – jako całość – fałszywe jedynie wtedy, gdy tworzące je zdania proste są jednocześnie fałszywe. W pozostałych przypadkach alternatywa jest prawdziwa.

Koniunkcja

Koniunkcja to połączenie zdań p i q spójnikiem „i", które ma postać: p *i* q. Może to być także połączenie form zdaniowych $p(x)$ *i* $q(x)$ w postaci: $p(x)$ *i* $q(x)$.

p	q	p i q
$x > 0$	$x < 4$	$x > 0$ i $x < 4$
y jest liczbą pierwszą	z jest liczbą złożoną	y jest liczbą pierwszą i z jest liczbą złożoną

Koniunkcja jest zdaniem prawdziwym jedynie wtedy, gdy tworzące ją zdania proste są jednocześnie prawdziwe. W pozostałych przypadkach koniunkcja jest fałszywa.

Tabela wartości logicznych

Wszystkie możliwe przypadki wartości logicznych zdań składowych i odpowiadające im wartości logiczne zdań złożonych można zestawić w tabeli:

p	q	$\sim p$	p lub q	p i q
1	0	0	1	0
0	1	1	1	0
1	1		1	1
0	0		0	0

Operatory porównań (relacji) w językach C++ i Python

Porównywanie dwóch wartości w językach programowania wykonuje się za pomocą operatorów porównań (relacji). Wynikiem ich działania są wartości `true` albo `false` w języku C++ lub `True` albo `False` w języku Python.

Zapis w matematyce	Zapis w językach C++ i Python
=	==
≠	!=
>	>
<	<
≥	>=
≤	<=

Operatory logiczne w językach C++ i Python

Spójniki logiczne („i", „lub", „nieprawda, że") nazywa się operatorami logicznymi. Umożliwiają one stosowanie wyrażeń logicznych w tworzonym kodzie. Można je łączyć z operatorami porównań przedstawionymi na s. 253.

Nazwa operatora logicznego	Zapis w języku C++	Przykład w języku C++
AND (koniunkcja)	`&&`	Dla zmiennej `x` o wartości 3 i zmiennej `y` o wartości 2 wyrażenie `x < 5 && y > 1` ma wartość `true`.
OR (alternatywa)	`||`	Dla zmiennej `x` o wartości 5 i zmiennej `y` o wartości 5 wyrażenie `x == 5 || y == 5` ma wartość `true`.
NOT (negacja)	`!`	Dla zmiennej `x` o wartości 3 wyrażenie `!(x == 4)` ma wartość `true`.

W języku C++ zamiast operatorów `&&`, `||` oraz `!` można używać odpowiednio operatorów: `and`, `or` i `not`.

Nazwa operatora logicznego	Zapis w języku Python	Przykład w języku Python
AND (koniunkcja)	`and`	Dla zmiennej `x` o wartości 3 i zmiennej `y` o wartości 2 wyrażenie `x < 5 and y > 1` ma wartość `True`.
OR (alternatywa)	`or`	Dla zmiennej `x` o wartości 5 i zmiennej `y` o wartości 5 wyrażenie `x == 5 or y == 5` ma wartość `True`.
NOT (negacja)	`not`	Dla zmiennej `x` o wartości 3 wyrażenie `not x == 4` ma wartość `True`.

6. Typy danych w języku C++

W języku C++ każda zmienna, w zależności od tego, czy jest liczbą całkowitą, liczbą z częścią ułamkową, znakiem, napisem czy wartością logiczną, musi mieć jawnie przypisany odpowiedni typ danych. Określa on sposób zapisu zmiennej w pamięci komputera oraz zbiór operacji, które można wykonać na tej zmiennej.

Typy liczbowe

Wśród typów liczbowych wyróżnia się typy całkowite oraz zmiennoprzecinkowe (zmiennopozycyjne), czyli takie, które pozwalają zapisać liczby rzeczywiste. W programowaniu wartości zapisane jako 2 i 2.0 to liczby różnych typów. Pierwsza z nich jest typu całkowitego, a druga jest typu zmiennoprzecinkowego, ponieważ zawiera zapis części ułamkowej.

Nazwa typu	Przeznaczenie zmiennej	Deklaracja zmiennej w języku C++
`int`	Liczby całkowite z zakresu od –2 147 483 648 do 2 147 483 647	`int a = 4;`
`long long`	Duże liczby całkowite nie większe niż $9{,}22 \cdot 10^{18}$	`long long a = 300000000000;`
`float`	Ułamki dziesiętne i ich przybliżenia	`float a = -0.5;`
`float`	Liczby z częścią ułamkową, w których liczba cyfr znaczących (pewnych) jest nie większa niż 6	`float a = 2.45123;`
`double`	Bardzo małe liczby, które trzeba zapamiętać z większą dokładnością (nie więcej niż 15 cyfr znaczących)	`double a = 4.00000023e-12;`
`double`	Bardzo duże liczby (z częścią ułamkową) lub liczby, które trzeba zapamiętać z większą dokładnością (nie więcej niż 15 cyfr znaczących)	`double a = 4.00000023e12;`

Uwaga: Zapisy typu `e12` oraz `e-12` oznaczają notację wykładniczą (tj. odpowiednio czynniki 10^{12} oraz 10^{-12}).

Typ znakowy i typ napisowy

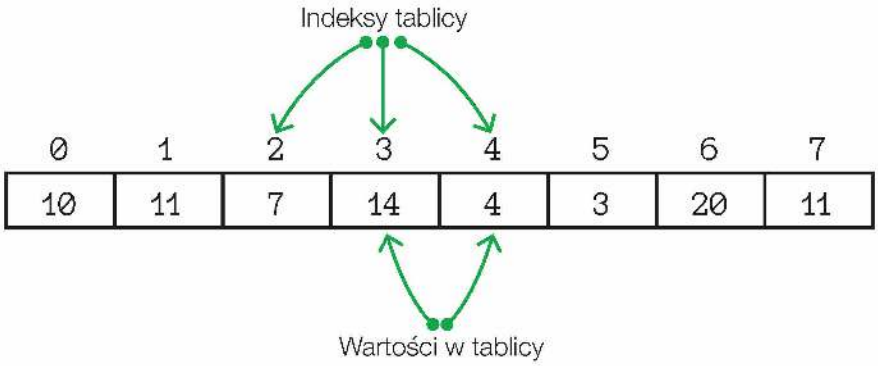
Pojedyncze znaki można zapisywać w zmiennych typu `char`. Na przykład deklaracja `char litera='a';` to deklaracja zmiennej, w której następnie zapisano literę `a`. Napisy, czyli tzw. łańcuchy znaków, zapisuje się z użyciem zmiennych typu `string`. Typ ten w sensie ścisłym nie jest typem prostym C++, ale ma charakter obiektowy, tzn. po deklaracji zmiennej uzyskujemy dostęp do wbudowanych funkcji (tzw. metod), które pozwalają łatwo wykonać operacje na napisach. Dla zmiennej zadeklarowanej jako `string napis="ABC";` liczbę znaków napisu wypiszemy na ekranie, jeśli użyjemy w kodzie zapisu `cout<<napis.size();`.

Typ logiczny

Do deklarowania zmiennych logicznych używa się typu **bool**, który przyjmuje wartości **true** i **false**. Są to słowa kluczowe języka C++. Wartość **false** reprezentowana jest w pamięci komputera przez liczbę 0, a wartość **true** – przez liczbę różną od 0.

Tablice

Tablica to struktura danych grupująca elementy określonego typu w sposób uporządkowany. Na przykład deklaracja `int tablica[4];` zastępuje deklarację czterech zmiennych typu **int**.



Tablice upraszczają proces zapisywania algorytmów w języku programowania, gdyż można odwołać się do wielu wartości za pomocą jednej nazwy z liczbowymi indeksami. Dostęp do poszczególnych elementów zapisanych w tablicy uzyskuje się za pomocą mechanizmu indeksowania, tj. podania pozycji elementu w tablicy, licząc od 0, np. `tablica[0]`, `tablica[1]`, `tablica[2]` i `tablica[3]`.

Zapis	Wyjaśnienie
`int tablica[4];`	Deklaracja tablicy czterech liczb typu **int**
`const int n = 4;` `int tablica[n];`	Deklaracja tablicy, której liczbę elementów określa stała n
`int tablica[3] = {1,2,3};`	Deklaracja tablicy trzech liczb typu **int** połączona z przypisaniem im wartości
`int tablica[] = {1,2,3};`	Deklaracja tablicy liczb typu **int** połączona z przypisaniem im wartości – rozmiar tablicy wyznacza się na podstawie liczby wartości
`int tablica[4] = {0};`	Deklaracja tablicy czterech liczb typu **int** połączona z przypisaniem im wszystkim wartości 0
`int tablica[4] = {1, 2};`	Deklaracja tablicy czterech liczb typu **int** połączona z przypisaniem im wartości: 1, 2, 0 i 0

7. Podstawowe typy instrukcji w języku C++

Instrukcja warunkowa

Składnia prostej instrukcji warunkowej:

Pojedyncza instrukcja	Wiele instrukcji

`if (warunek) instrukcja;`	`if (warunek)`
	`{`
	`  instrukcja1;`
	`  instrukcja2;`
	`  ...`
	`}`

Warunek podawany po słowie **if** powinien być wyrażeniem typu logicznego (patrz dodatek 5).

Składnia instrukcji warunkowej w postaci alternatywy:

Pojedyncza instrukcja	Wiele instrukcji

`if (warunek)`
`  instrukcja;`
`else`
`  instrukcja;`
	`if (warunek)`
	`{`
	`  instrukcja1;`
	`  instrukcja2;`
	`  ...`
	`}`
	`else`
	`{`
	`  instrukcja1;`
	`  instrukcja2;`
	`  ...`
	`}`

W przypadku zagnieżdżania instrukcji warunkowych warto zadbać o odpowiednie wcięcia.

Instrukcje powtarzania. Pętla `for`

Składnia pętli **for** powtarzającej pojedynczą instrukcję:

```
for (licznik; warunek powtarzania; krok) instrukcja;
```

Składnia pętli **for** powtarzającej wiele instrukcji:

```
for (licznik; warunek powtarzania; krok)
{
    instrukcja1;
    instrukcja2;
    ...
}
```

Instrukcje powtarzania. Pętla `while`

Składnia pętli **`while`** powtarzającej pojedynczą instrukcję:

```
while (warunek logiczny) instrukcja;
```

Składnia pętli **`while`** powtarzającej wiele instrukcji:

```
while (warunek logiczny)
{
    instrukcja1;
    instrukcja2;
    ...
}
```

Instrukcja iteracyjna. Pętla `do while`

Składnia pętli **`do while`** powtarzającej pojedynczą instrukcję:

```
do instrukcja while (warunek logiczny);
```

Składnia pętli **`do while`** powtarzającej wiele instrukcji:

```
do
{
    instrukcja1;
    instrukcja2;
...
} while (warunek logiczny);
```

Instrukcje sterujące. Funkcje

Funkcje to podprogramy zawarte w programie głównym, które odpowiadają za realizację określonej części całego programu. Funkcje definiuje się przed definicją funkcji `main`.

Definicja funkcji składa się z nagłówka funkcji i bloku instrukcji. Nagłówek zawiera typ wartości zwracanej przez funkcję, nazwę funkcji oraz listę argumentów wraz z typami i nazwami argumentów. Budowa funkcji:

```
typ nazwa_funkcji(typ argument1, typ argument2, ...)
{
    instrukcja1;
    instrukcja2;
    ...
    return wynik;
}
```

Wartość zwracana przez funkcję jest wskazana po słowie **`return`**.
Na przykład instrukcja `return a;` oznacza, że funkcja zwróci wartość zmiennej `a`.

Gdy typem funkcji w nagłówku jest **`void`**, funkcja wykonuje zadania, które nie wymagają zwrócenia wartości wynikowej.

8. Operatory i funkcje matematyczne w języku C++

Operatory arytmetyczne

Operator	Nazwa operatora	Wyrażenie	Wynik
`+`	Dodawanie	`2 + 3` `2 + 3.5`	`5` `5.5`
`-`	Odejmowanie	`3 - 2` `3.5 - 2`	`1` `1.5`
`*`	Mnożenie	`2 * 3` `2.5 * 3.5`	`6` `8.75`

`/`	Dzielenie	`3 / 2`
		`-3 / 2`
		`3.0 / 2`
			`1`
			`-1`
			`1.5`

`%`	Reszta z dzielenia (modulo)	`3 % 2`	`1`

W języku C++ nie ma operatora potęgowania. Kwadrat lub sześcian liczby najłatwiej zapisać jako iloczyn, odpowiednio: `2 * 2` lub `2 * 2 * 2`.
Do zapisywania potęg można też użyć funkcji `pow` z biblioteki `cmath`.

Wybrane funkcje matematyczne z biblioteki `cmath`

Działanie	Wyrażenie	Wynik	Uwagi
Potęgowanie	`pow(2.0; -2)`	`0.25`	Drugi argument funkcji to wykładnik potęgi. Może być liczbą niecałkowitą.
Pierwiastek kwadratowy	`sqrt(2.0)`	`1.41421`	Zwracana jest wartość przybliżenia.
Pierwiastek sześcienny	`cbrt(2.0)`	`1.25992`	Zwracana jest wartość przybliżenia.
Zaokrąglenie w górę (sufit)	`ceil(2.5)` `ceil(-2.5)`	`3` `-2`	
Zaokrąglenie w dół (podłoga)	`floor(2.5)` `floor(-2.5)`	`2` `-3`	
Zaokrąglenie (tradycyjne)	`round(2.5)` `round(-2.5)`	`3` `-3`	
Obcięcie części ułamkowej	`trunc(2.5)` `trunc(-2.5)`	`2` `-2`	
Wartość bezwzględna	`abs(2)` `abs(-2)`	`2` `2`	Można też stosować funkcję `fabs` z biblioteki `math.h`
Logarytm dziesiętny	`log10(2.0)`	`0.30103`	Zwracana jest wartość przybliżenia.

9. Słowa kluczowe w języku C++

Słowa kluczowe nie mogą być nazwami zmiennych, funkcji ani innych obiektów w kodzie źródłowym.

Słowo kluczowe	Opis
`bool`	Nazwa typu logicznego
`const`	Używane przy definiowaniu stałej
`double`	Nazwa typu dla liczb z częścią ułamkową, o tzw. podwójnej precyzji
`else`	Używane w instrukcji warunkowej do wyodrębnienia kodu wykonywanego, gdy warunek po `if` nie jest spełniony
`false`	Stała oznaczająca wartość logiczną fałsz
`float`	Nazwa typu dla liczb z częścią ułamkową
`for`	Używane w instrukcji iteracyjnej
`if`	Używane w instrukcji warunkowej
`int`	Nazwa typu dla liczb całkowitych
`long`	Nazwa typu dla liczb całkowitych, definicja z `long long int` pozwala zapisywać w pamięci komputera liczby z większego zakresu liczb
`namespace`	Używane przy określaniu tzw. przestrzeni nazw
`return`	Używane w instrukcji, która pozwala funkcji zwrócić wartość do miejsca jej wywołania
`true`	Stała oznaczająca wartość logiczną prawda
`using`	Używane przy określaniu tzw. przestrzeni nazw
`void`	Używane w nagłówku funkcji, która nie zwraca wartości
`while`	Używane w instrukcji iteracyjnej

Inne słowa kluczowe języka C++

`asm`	`auto`	`break`	`case`
`catch`	`char`	`class`	`const_cast`
`continue`	`default`	`delete`	`do`
`dynamic_cast`	`enum`	`explicit`	`export`
`extern`	`friend`	`goto`	`inline`
`mutable`	`new`	`operator`	`private`
`protected`	`public`	`register`	`reinterpret_cast`
`short`	`signed`	`sizeof`	`static`
`static_cast`	`struct`	`switch`	`template`
`this`	`throw`	`try`	`typedef`
`typeid`	`type_info`	`typename`	`union`
`unsigned`	`virtual`	`volatile`	`wchar_t`

10. Typy danych w języku Python

W języku Python typ zmiennych określany jest na podstawie wartości, które są przypisane zmiennym.

Typy liczbowe

Wśród typów liczbowych wyróżnia się typy całkowite i zmiennoprzecinkowe (zmiennopozycyjne), pozwalające zapisać również ułamki (liczby wymierne). W programowaniu wartości zapisane jako 2 i 2.0 to liczby różnych typów. Pierwsza z nich jest typu całkowitego, a druga jest typu zmiennoprzecinkowego, ponieważ zawiera zapis części ułamkowej.

Nazwa typu	Przeznaczenie zmiennej	Przykład przypisania wartości
`int`	Liczby całkowite	`a = 4`
`float`	Ułamki dziesiętne i ich przybliżenia	`a = -0.5;`
`float`	Liczby z częścią ułamkową	`a = 2.45123;` `a = 4.00000023e-12`

Uwaga: Zapis `e-12` oznacza notację wykładniczą (tj. czynnik 10^{-12}).

Typ logiczny

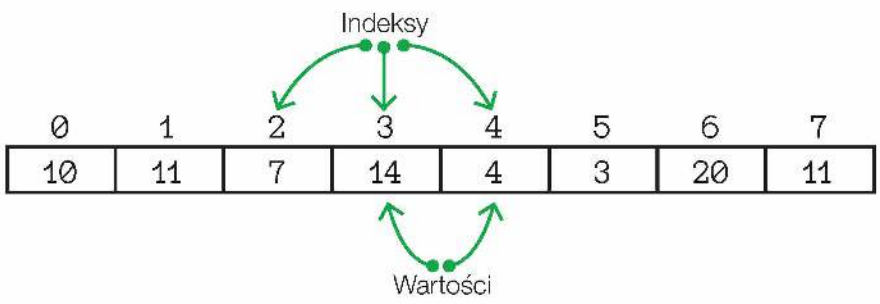
Do deklarowania zmiennych logicznych używa się typu `bool`, który przyjmuje wartości `True` i `False`. Są one słowami kluczowymi języka Python. Wartość `False` reprezentowana jest w pamięci komputera przez liczbę 0, a wartość `True` – przez liczbę 1.

Typ znakowy i typ napisowy

Napisy, czyli tzw. łańcuchy znaków, Python zapisuje w pamięci komputera z użyciem typu `str`. Podczas przypisania wartości do zmiennej stosuje się cudzysłowy (`napis = "ABC"`) albo pary apostrofów (`napis = 'ABC'`).
Napisy można łączyć za pomocą operatora złączania (konkatenacji) +: np. kod
`napis = "ABC" + "DE"` skutkuje zapamiętaniem napisu ABCDE.

Listy

Lista to struktura danych grupująca elementy dowolnych typów w sposób uporządkowany. Na przykład dzięki utworzeniu listy `L = [1, 2, 3, 4]` można zastąpić cztery zmienne typu całkowitego. Listy upraszczają proces kodowania algorytmów, gdyż można odwołać się do wielu wartości za pomocą jednej nazwy z liczbowymi indeksami. Dostęp do elementów zapisanych na liście uzyskuje się za pomocą mechanizmu indeksowania, tj. podania pozycji elementu na liście, licząc od 0, np. `L[0]`, `L[1]`, `L[2]` i `L[3]`.



11. Podstawowe typy instrukcji w języku Python

Instrukcja warunkowa

Składnia prostej instrukcji warunkowej:

```
if warunek:
    instrukcja1
    instrukcja2
    instrukcja3
    ...
```

Warunek podawany po słowie **if** powinien być wyrażeniem typu logicznego (patrz dodatek 5).

Składnia instrukcji warunkowej w postaci alternatywy:

```
if warunek:
    instrukcja1
    instrukcja2
    ...
else:
    instrukcja1
    instrukcja2
    ...
```

W przypadku zagnieżdżania instrukcji warunkowych trzeba zadbać o to, aby kolejne wcięcia były wielokrotnością pojedynczego wcięcia.

```
if warunek1:
    instrukcja
    if warunek2:
        instrukcja
    else:
        instrukcja
else:
   instrukcja
```

Oto przykładowe wykorzystanie instrukcji warunkowej:

```
if artykul == "buty":
    if kolor == "zielony":
        rabat = 0.5
    else:
        rabat = 0.2
else:
    rabat = 0.1
```

Instrukcje powtarzania. Pętla `for`

Składnia pętli **for** wykonującej iterację według elementów listy ma postać:

```
for k in nazwa_listy:
    instrukcja1
    instrukcja2
    ...
```

Przykład: wypisanie elementów listy [1, 3, 5] z użyciem pętli **for**.

```
liczby = [1, 3, 5]
for k in liczby:
    print(k)
```

Składnia pętli **for** wykonującej iterację według liczb z zakresu:

```
for k in range(zakres):
    instrukcja1
    instrukcja2
    ...
```

Przykład: wypisanie liczb od 1 do 10 z użyciem pętli **for**.

```
for k in range(1, 11):
    print(k)
```

Instrukcje powtarzania. Pętla `while`

Składnia pętli **while** ma postać:

```
while warunek_logiczny:
    instrukcja1
    instrukcja2
    ...
```

Instrukcję **while** stosujemy wówczas, gdy nie znamy dokładnej liczby powtórzeń (cykli). Program wykonuje tę instrukcję, dopóki warunek jest spełniony. Jest więc możliwe, że nie wykona jej ani razu.

Oto przykładowe wykorzystanie pętli **while**:

```
imie = "Joanna"
proba = ""
i = 0

print("Spróbuj odgadnąć moje imię")
while proba != imie:
    i = i + 1
    print("Próba", i, ": ", end="")
    proba = input()

print("Brawo! Udało ci się odgadnąć moje imię za", i, "razem.")
```

Funkcje

Funkcje to podprogramy zawarte w programie głównym, które odpowiadają za realizację określonej części całego programu.

Budowa funkcji:

```
def nazwa_funkcji(argument1, argument2, ...):
    instrukcja1
    instrukcja2
    ...
    return wynik
```

Definicja funkcji składa się z nagłówka funkcji i bloku instrukcji. Nagłówek zawiera nazwę funkcji oraz listę nazw argumentów funkcji. Wartość zwracana przez funkcję jest wskazana po słowie **return**. Na przykład instrukcja `return a` oznacza, że funkcja zwróci wartość zmiennej a.

Gdy w definicji funkcji instrukcja **return** się nie pojawia, funkcja wykonuje zadania, które nie wymagają zwrócenia wartości wynikowej.

Oto przykład funkcji PoleTrapezu, która oblicza pole trapezu dla podanych długości podstaw i wysokości:

```
def PoleTrapezu(a, b, h):
    return (a+b)*h/2

print("*** Program Pole trapezu *** \n")
a = float(input("Podaj długość jednej podstawy trapezu: "))
b = float(input("Podaj długość drugiej podstawy trapezu: "))
h = float(input("Podaj wysokość trapezu: "))

print("Pole trapezu wynosi", PoleTrapezu(a, b, h))
```

12. Operatory i funkcje matematyczne w języku Python

Operatory arytmetyczne

Operator	Nazwa operatora	Wyrażenie	Wynik
+	Dodawanie	`2 + 3` `2 + 3.5`	`5` `5.5`
–	Odejmowanie	`3 - 2` `3.5 - 2`	`1` `1.5`
*	Mnożenie	`2 * 3` `2.5 * 3.5`	`6` `8.75`
**	Potęgowanie	`2 ** 3`	`8`
/	Dzielenie	`3 / 2` `3.0 / 2`	`1.5` `1.5`
//	Dzielenie całkowite	`3 // 2` `-3 // 2`	`1` `-2`
%	Reszta z dzielenia (modulo)	`3 % 2`	`1`

Wybrane wbudowane funkcje matematyczne

Działanie	Wyrażenie	Wynik	Uwagi
Potęgowanie	`pow(2.0,-2)`	`0.25`	Drugi argument funkcji to wykładnik potęgi.
Wartość bezwzględna	`abs(2)` `abs(-2)`	`2` `2`	
Konwersja na liczbę typu całkowitego	`int(12.5)` `int("12")`	`12` `12`	

Zaokrąglenie	`round(2.5)`
	`round(2.6)`
	`round(3.5)`
	`round(3.6)`
		`2`
		`3`
		`4`
		`4`
			Liczby z częścią ułamkową 0,5 program zaokrągla inaczej dla liczb o części całkowitej parzystej i nieparzystej.

Wybrane funkcje matematyczne z biblioteki `math`

Działanie	Wyrażenie	Wynik	Uwagi
Pierwiastek kwadratowy	`sqrt(2.0)`	`1.41421`	Zwracana jest wartość przybliżenia.
Logarytm dziesiętny	`log10(2.0)`	`0.30103`	Zwracana jest wartość przybliżenia.
Zaokrąglenie w górę (sufit)	`ceil(2.5)` `ceil(-2.5)`	`3` `-2`	
Zaokrąglenie w dół (podłoga)	`floor(2.5)` `floor(-2.5)`	`2` `-3`	

13. Słowa kluczowe w języku Python

Słowa kluczowe nie mogą być nazwami zmiennych, funkcji ani innych obiektów w kodzie źródłowym.

Słowo kluczowe	Opis
`and`	Operator logiczny
`def`	Używane w definicji funkcji
`else`	Używane w instrukcji warunkowej do wyodrębnienia kodu wykonywanego, gdy warunek po `if` nie jest spełniony
`for`	Używane w instrukcji iteracyjnej
`from`	Używane przy imporcie fragmentu biblioteki (modułu)
`if`	Używane w instrukcji warunkowej
`import`	Używane przy imporcie biblioteki (modułu)
`in`	Używane do odwołania się do elementów listy
`is`	Używane do sprawdzania, czy dwa obiekty są tymi samymi obiektami, np. czy zwracana wartość jest równa `True`
`not`	Operator logiczny
`or`	Operator logiczny
`return`	Używane w instrukcji, która pozwala funkcji zwrócić wartość do miejsca jej wywołania
`True`	Stała oznaczająca wartość logiczną prawda
`while`	Używane w instrukcji iteracyjnej
`False`	Stała oznaczająca wartość logiczną fałsz

Inne słowa kluczowe języka Python

`as`	`assert`	`break`	`class`
`continue`	`del`	`elif`	`except`
field	`finally`	`global`	`lambda`
`None`	`nonlocal`	`pass`	`raise`
`try`	`with`		

14. Systemy liczbowe

Systemem liczbowym nazywamy zbiór reguł jednolitego zapisu liczb. Do zapisu liczb w danym systemie liczbowym używa się cyfr.

System dziesiętny

Podstawą systemu jest liczba 10. Do zapisu wykorzystujemy cyfry: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Liczbę w systemie dziesiętnym możemy przedstawić jako sumę liczb, z których każda jest iloczynem potęgi liczby 10 oraz liczby od 0 do 9.

System dwójkowy (binarny)

Podstawą systemu jest liczba 2. Do zapisu wykorzystujemy zbiór cyfr: {0, 1}. Liczbę w systemie binarnym możemy przedstawić jako sumę liczb, z których każda jest iloczynem potęgi liczby 2 oraz liczby 0 lub 1.

System szesnastkowy (heksadecymalny)

Podstawą systemu jest liczba 16. Do zapisu wykorzystujemy zbiór cyfr: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}. Liczbę w systemie szesnastkowym możemy przedstawić jako sumę liczb, z których każda jest iloczynem potęgi liczby 16 oraz liczby od 0 do F (gdzie: A = 10, B = 11, C = 12, D = 13, E = 14, F = 15).

Przeliczanie liczb pomiędzy systemami

Zamiana liczby zapisanej w systemie dziesiętnym na liczbę zapisaną w innym systemie liczbowym polega na jej wielokrotnym dzieleniu całkowitym przez podstawę danego systemu i zapisywaniu reszty z dzielenia. Można to zrobić w tabeli – liczbę po zamianie odczytujemy od dołu do góry. Ilustruje to poniższy przykład.

Zamiana zapisu liczby z systemu dziesiętnego na dwójkowy					
150	:	2	=	75	0
75	:	2	=	37	1
37	:	2	=	18	1
18	:	2	=	9	0
9	:	2	=	4	1
4	:	2	=	2	0
2	:	2	=	1	0
1	:	2	=	0	1
$150_{10} = 10010110_2$					

Zamiana zapisu liczby z systemu dziesiętnego na szesnastkowy					
19374	:	16	=	1210	E
1210	:	16	=	75	A
75	:	16	=	4	B
4	:	16	=	0	4
$19374_{10} = 4BAE_{16}$					

Indeks

A
adres komórki 11
adresowanie bezwzględne 14
— mieszane 14
— względne 14
algorytm 94, 172
— Euklidesa 158, 235
—— – wersja z dzieleniem 160, 237
— Herona 98, 176
— wybierania największej z trzech liczb 108, 185
arkusz 10
atrybut 69

B
bajt 134, 211
baza danych 67
biblioteka cmath 150
bit 128, 205
— najbardziej znaczący 131, 208
— najmniej znaczący 131, 208
błąd składniowy 102, 180

C
cechy poprawnego algorytmu 96, 174

D
deklaracja zmiennej 104
dokument główny korespondencji seryjnej 54
— końcowy korespondencji seryjnej 54
drzewo decyzyjne 131, 208

E
Edytor Power Query 24

F
filtrowanie 30
— w tabeli przestawnej 47
format CSV 19
formularze 80
formuła 11
fragmentator 49
funkcja (arkusz kalkulacyjny) 11
— DŁ 17
— INDEKS 35
— JEŻELI 17
— LICZ.JEŻELI 28
— LOS.ZAKR 33
— MAX 16
— MAX.K 63
— MIN 16
— SUMA 11
— ŚREDNIA 16
funkcja (C++) 145
— main 101
— niezwracająca wartości 163
— sqrt 150
— swap 155
funkcja (Python) 222
— float 193
— input 182
— int 182
— niezwracająca wartości 239
— print 179
— range 195
— reversed 214
— sqrt 226
— str 232
funkcja (w językach programowania) 99, 177

I
indeks (C++) 135
— (Python) 212
instrukcja 99, 177
— iteracyjna (i. powtarzania, pętla) (C++) 116
—————— (Python) 193
— return (C++) 145
— return (Python) 222
— warunkowa (C++) 105
—— (Python) 183
— wejścia (C++) 104
—— (Python) 182
— wyjścia (C++) 101
—— (Python) 179
interpreter 99, 177

K
klucz główny (klucz podstawowy) 70
— obcy 70
kod źródłowy 99, 177

kompilator 99, 177
koniunkcja 106, 184
korespondencja seryjna 54
Kreator importu tekstu 19
— odnośników 77
kwerenda 82

L
liczby pierwsze 141, 218
— złożone 141, 218
lista 212
— kroków 96, 174

M
maska wprowadzania 76
modulo (C++) 135
— (Python) 212
moduł 226
— `math` 226

N
najmniejsza wspólna wielokrotność (NWW) 163, 240
największy wspólny dzielnik (NWD) 154, 231
narzędzie Nowe zapytanie 23

O
obszary tabeli przestawnej 38
odwołania strukturalne 63
operacja pierwiastkowania (C++) 150
—— (Python) 226
operator scalania napisów 233
oś czasu 49

P
parametr aktualny funkcji 146, 223
— formalny funkcji 146, 223
— funkcji 145, 222
pętla (instrukcja iteracyjna, i. powtarzania) (C++) 116
————— (Python) 193
— `for` (C++) 118
— `for` (Python) 194
— `while` (C++) 116
— `while` (Python) 193
plik wykonywalny 102
pliki binarne 18
— tekstowe 18
podstawa systemu liczbowego 126, 203
podstawowa struktura kodu źródłowego w języku C++ 101
podstawowe twierdzenie arytmetyki 141, 218
— własności liczb binarnych 127, 204
pole korespondencji seryjnej 56
— obliczeniowe 45
— rekordu 69

R
raport 84
rekord 56, 69
relacja 70
— typu jeden do jednego 71
——— do wielu 70
—— wiele do wielu 71
rozkład na czynniki pierwsze 144, 219

S
scalanie dokumentów korespondencji seryjnej 65
separator 19
skoroszyt 10
słowa kluczowe języka programowania 99, 177
sortowanie 30
— w tabeli przestawnej 46
specyfikacja problemu 97, 175
system dwójkowy (binarny) 127, 204
— dziesiętny (decymalny) 126, 203
— ósemkowy 128, 205
— pozycyjny 126, 203
— szesnastkowy 128, 205
— zarządzania bazą danych (SZBD) 71

T
tabela bazy danych 69
— programu Access 74
—— Excel 32
— przestawna 35
tablica 135
translator 99, 177
typ danych 104

U
uchwyt wypełniania 15

W
warunek istnienia trójkąta 107, 185
Widok projektu 75
wykres przestawny 50

Z

zakres adresów 11
zależność między NWD a NWW 163, 240
zamiana wartości zmiennych 109, 185
zapisywanie zmian w tabeli 75
zestaw danych korespondencji seryjnej 54
zmienna 104, 181
— sterująca 118, 195

Zdjęcie na okładce: Getty Images/iStockphoto/Greyfebruary

Zdjęcia: Agencja Gazeta: Arkadiusz Scichocki s. 133, 210, Łukasz Giza s. 144, 219; BE&W: Alamy Stock Photo – Art Collection 2 s. 164, FLHC 80 s. 93, 171 (Liber abaci), GL Archive s. 93, 171 (Leibniz), J Marshall – Tribaleye Images s. 96, 174 (posąg Al-Chuwarizmiego), Kumar Sriskandan s. 92, 170 (mnożenie egipskie), urbanbuzz s. 142, 220 (tokeny); Getty Images: Archive Photos/Ann E. Yow-Dyson s. 9 (VisiCalc), Corbis Historical/Leemage s. 8 (Biblioteka Aleksandryjska), DigitalVision Vectors s. 92, 170 (Euklides), E+ – FatCamera s. 88 (wolontariusz), loops7 s. 91, PeopleImages s. 115, 192, technotr s. 112, 189, Getty Images Europe/Getty Images Sport/Mike Hewitt s. 121, 199, iStock/Getty Images Plus – AnatolyM s. 41, AntonioGuillem s. 88 (marketingowiec), g-stockstudio s. 88 (kontroler finansowy), matejmo s. 169, Mirexon s. 7, stefanamer s. 88 (dietetyczka), Moment/MR.Cole_Photographer s. 9 (autonomiczna baza danych), SSPL – Science & Society Picture Library s. 93, 171 (kostki Napiera), Science Museum s. 93, 171 (maszyna Jacquarda), Universal Images Group/SVF2 s. 92, 170 (Al-Chuwarizmi); Indigo: Alamy Stock Photo: Pictorial Press Ltd s. 93, 171 (Ada Lovelace), UPI s. 8 (Charles Bachman), De Agostini Editore/De Agostini/Chomon & Perino s. 93, 171 (maszyna Babbage'a), Science Photo Library/KING'S COLLEGE LONDON s. 8 (stół obliczeniowy); Shutterstock: A_stockphoto s. 168, 244 (kontrola systemu ppoż.), Alex Alekseev s. 8 (abakus), Alexander Raths s. 168, 244 (naukowcy), Axel Bueckert s. 113, 190 (człowiek), Babich Alexander s. 8 (maszyna Schickarda), Chaosamran_Studio s. 26–27, dencg s. 99, 177, DmitriyRazinkov s. 129, 206 (fale dźwiękowe), Dmitry Kalinovsky s. 168, 244 (inżynier), itechno s. 142, 220 (posąg Euklidesa), metamorworks s. 51, NicoElNino s. 79, pyrozhenka s. 153, 230, SeventyFour s. 168, 244 (logistyka), solarseven s. 83, stscheb s. 89, Tatiana Shepeleva s. 113, 190 (robot), Titima Ongkantong s. 129, 206 (fale dźwiękowe), topseller s. 23, Voyagerix s. 96, 174 (warkocz), Wlad74 s. 8 (maszyna Holleritha), Yuriy Lukin s. 129, 206 (koncert), zffoto s. 142–143, 220–221 (fale w tle); Wikipedia/Wikimedia Commons s. 92, 170 (szyfr przestawieniowy, Heron), Hubert Berberich (HubiB) s. 92, 170 (szyfr podstawieniowy), Jeremy Kratz s. 100, link: http://www.math.ubc.ca/~cass/Euclid/papyrus/tha.jpg s. 159, 236; Tatsuo Horiuchi link: www2.odn.ne.jp/~cbl97790 (http://pasokonga.com/) s. 14; link: www.python.org s. 178.

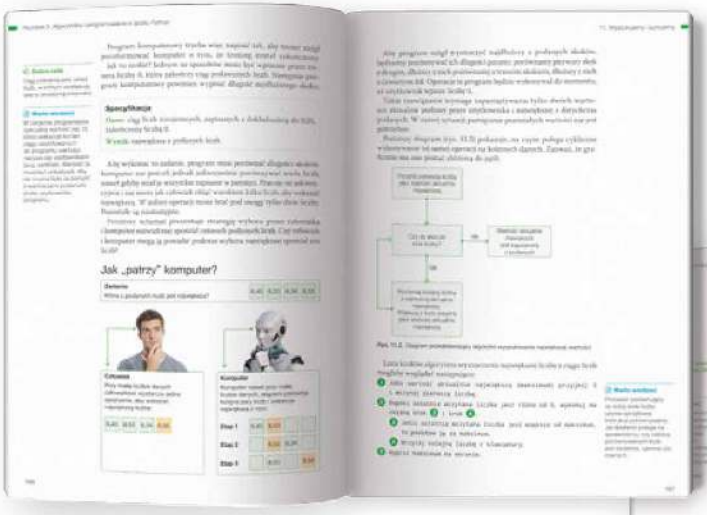
Wydawnictwo dołożyło wszelkich starań, aby odnaleźć posiadaczy praw autorskich do wszystkich utworów zamieszczonych w podręczniku. Pozostałe osoby prosimy o kontakt z Wydawnictwem.

Wydawnictwo Nowa Era nie ponosi odpowiedzialności za treści i zasoby zamieszczone na cytowanych w tej publikacji stronach internetowych.

nowa era *Twoje mocne strony*

Informatyka na czasie

Podręcznik *Informatyka na czasie 2* do zakresu podstawowego zawiera treści dotyczące arkusza kalkulacyjnego, baz danych oraz algorytmiki i programowania – w językach C++ i Python. Opanowanie tych zagadnień ułatwi aktywny margines z siatką pojęć oraz ramkami *Dobra rada* i *Warto wiedzieć* – informacjami uzupełniającymi wiadomości z lekcji. W nauce pomogą również atrakcyjne infografiki oraz liczne ilustracje, schematy, zrzuty ekranowe i kody źródłowe.



Różnorodne formy przekazu

Rysunki i schematy, a także czytelne zrzuty ekranowe i kody źródłowe ułatwiają zrozumienie i zapamiętanie kluczowych informacji.

Aktywny margines

Zawiera siatkę pojęć, odsyłacze oraz praktyczne wskazówki ułatwiające opanowanie niezbędnych wiadomości.



Informatyka bliska uczniowi

W podręczniku *Informatyka na czasie 2* do zakresu podstawowego trudne zagadnienia są wprowadzane poprzez odwołania do sytuacji bliskich doświadczeniu ucznia. Elementy *A to ciekawe* ukazują informatykę przez pryzmat faktów pozornie niemających z nią związku. Infografiki ujmują ważne treści w przejrzystej i atrakcyjnej dla ucznia formie.



Nowa Era Sp. z o.o.

www.nowaera.pl nowaera@nowaera.pl

Centrum Kontaktu: 801 88 10 10, 58 721 48 00

ISBN 978-83-267-3885-2