NOWOŚĆ

NOWA Informatyka na czasie

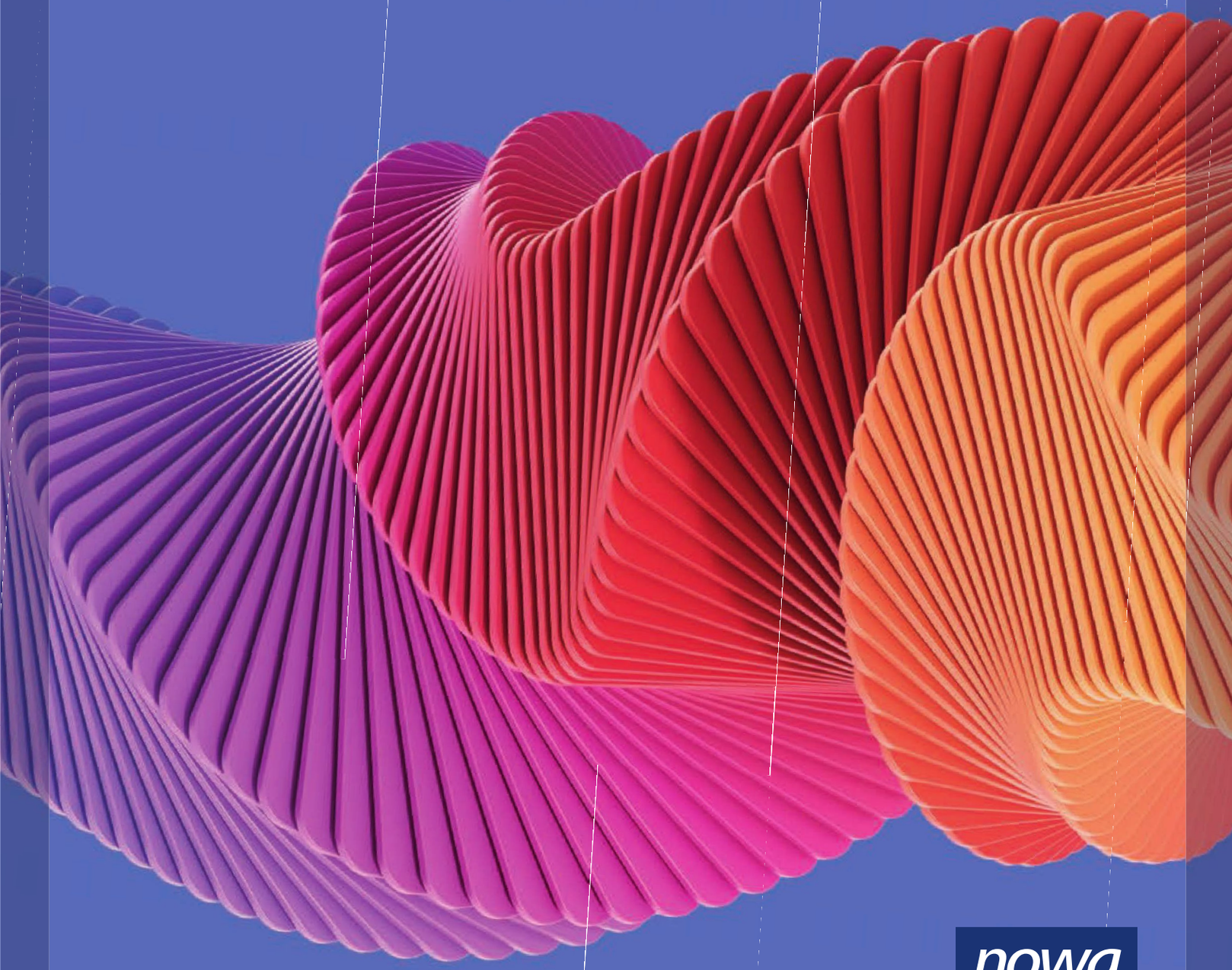
ZMIANY W PODSTAWACH 2024

podręcznik w pełni dostosowany

3

PODRĘCZNIK • LICEUM • TECHNIKUM

ZAKRES PODSTAWOWY



nowa era

NOWA

Informatyka na czasie

3

PODRĘCZNIK • LICEUM • TECHNIKUM

ZAKRES PODSTAWOWY

JANUSZ MAZUR
PAWEŁ PEREKIETKA
ZBIGNIEW TALAGA
JANUSZ S. WIERZBICKI

nowa era Twoje mocne strony

NOWA
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: 2026

Numer ewidencyjny w wykazie MEN: 1220/3/2026

Podręcznik przygotowano na podstawie *Programu nauczania informatyki w liceach i technikach. „NOWA 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**

Wykorzystywanie treści niniejszego Utworu, w jakiejkolwiek formie lub w jakimkolwiek procesie, do celów eksploracji tekstu i danych, uczenia maszynowego, rozwoju i/lub szkolenia i/lub wzbogacania sztucznej inteligencji jakiegokolwiek rodzaju jest surowo zabronione.

© Copyright by Nowa Era Sp. z o.o. 2026
ISBN 978-83-267-5347-3

Konsultacja merytoryczno-dydaktyczna: dr hab. Bartosz Bieganowski, Paweł Koszałka.

Redakcja merytoryczna: Przemysław Szydzik, Krzysztof Fit, Anna Nasiadka, Katarzyna Pietrzak.
Współpraca redakcyjna: Jacek Detlaf, Dorota Rzeszewska, Joanna Sawicka, Agnieszka Sieczak.
Redakcja językowa: Aleksandra Hofman, Paulina Szulim, Agnieszka Szymanowska-Pancer, Marta Zuchowicz.
Nadzór artystyczny: Ewa Kaletyn.
Opieka graficzna: Maciej Galiński.
Projekt okładki: Maciej Galiński.
Projekt graficzny: Sławomir Włodarczyk.
Opracowanie graficzne: Alicja Skupin-Podwojewska, Elena Grikurova-Wojnowska, Klaudia Jarocka, Marcin Kołacz, Sławomir Włodarczyk.
Ilustracje: Natalia Helman, Marcin Ptak.
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, Centrum Kontaktu: 58 721 48 00

Druk i oprawa: DRUK-SERWIS Sp. z o.o. Ciechanów

Spis treści

(F) Tym symbolem oznaczono treści spoza podstawy.

1. Algorytmika i programowanie w języku Python 7
1. Algorytm Euklidesa 8
1.1. Patchworkowe kołdry 8
1.2. Największy wspólny dzielnik 9
1.3. Działania na ułamkach zwykłych 16
2. Obliczanie wartości elementów ciągu 21
2.1. Definicja ciągu liczbowego 21
2.2. Ciąg Fibonacciego 23
3. Algorytmy na tekstach 29
3.1. Litery i inne znaki jako liczby 29
3.2. Szukanie literówek 33
3.3. Lista zakupów. Usuwanie duplikatów w tekście 40
3.4. Wyszukiwanie wzorca w tekście 42
4. Szyfrowanie wiadomości 47
4.1. Podstawowe pojęcia kryptografii 47
4.2. Szyfr Cezara 48
4.3. Metody łamania klasycznych szyfrów 53
Z informatyką w przyszłość 58

2. Algorytmika i programowanie w języku C++ 59
5. Algorytm Euklidesa 60
5.1. Patchworkowe kołdry 60
5.2. Największy wspólny dzielnik 61
5.3. Działania na ułamkach zwykłych 69
6. Obliczanie wartości elementów ciągu 74
6.1. Definicja ciągu liczbowego 74
6.2. Ciąg Fibonacciego 76
7. Algorytmy na tekstach 82
7.1. Litery i inne znaki jako liczby 82
7.2. Szukanie literówek. Łańcuchy znaków w języku C++ 86
7.3. Lista zakupów. Usuwanie duplikatów w tekście 93
7.4. Wyszukiwanie wzorca w tekście 95
8. Szyfrowanie wiadomości 101
8.1. Podstawowe pojęcia kryptografii 101
8.2. Szyfr Cezara 102
8.3. Metody łamania klasycznych szyfrów 107
Z informatyką w przyszłość 112

3. Projektowanie 3D 113

9. Grafika 3D 114

9.1. Modelowanie 3D 114
9.2. Tworzenie trójwymiarowego modelu szafki w programie SketchUp 118
9.3. Tworzenie bryły obrotowej w programie SketchUp 121
9.4. Łączenie różnych technik tworzenia grafiki 3D 122

F **10.** Druk 3D 126

10.1. Zasada działania drukarek 3D 127
10.2. Przygotowanie drukarki 3D do pracy 131
10.3. Przygotowanie wydruku obiektu 3D 134
10.4. Drukowanie modelu 3D 139

Z informatyką w przyszłość 142

4. Rozwiązywanie problemów z wykorzystaniem komputera 143

F **11.** Sztuczna inteligencja 144

11.1. Czym jest sztuczna inteligencja? 144
11.2. Poznawanie sztucznej inteligencji 148
11.3. Sztuczny neuron i uczenie z nauczycielem 151
11.4. Na czym polega uczenie sztucznych neuronów? 154
11.5. Praktyczne zastosowania sztucznej inteligencji 155

12. Sterowanie robotem 159

12.1. Czym są roboty? 159
12.2. Roboty w edukacji 162
12.3. Robot LEGO MINDSTORMS EV3 163
12.4. Mikrokontroler BBC micro:bit 165
12.5. Programowanie robotów i symulatory 166
12.6. Symulacja robota EV3 w środowisku Open Roberta Lab 166
12.7. Symulacja płytki micro:bit w środowisku Microsoft MakeCode 171
12.8. Środowisko do symulacji VEXcode VR 173

13. Korespondencja seryjna 179

13.1. Czym jest korespondencja seryjna? 179
13.2. Korespondencja seryjna w programie Word 180
13.3. Filtrowanie adresatów i personalizacja listów 184
13.4. Zarządzanie danymi korespondencji seryjnej w arkuszu kalkulacyjnym 187
13.5. Kończenie pracy z korespondencją seryjną 190

Z informatyką w przyszłość 192

Projekt zespołowy 193

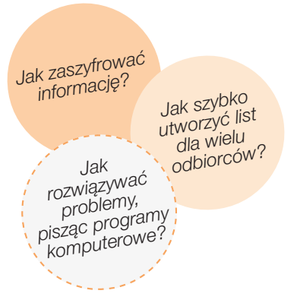
Dodatki 195

1. Sposoby zapisu algorytmów 196
2. Logika matematyczna i operatory logiczne 198
3. Typy danych w języku Python 201
4. Podstawowe typy instrukcji w języku Python 203
5. Operatory i funkcje matematyczne w języku Python 206
6. Słowa kluczowe w języku Python 207
7. Obiekty, klasy i metody w języku Python 208
8. Typy danych w języku C++ 209
9. Podstawowe typy instrukcji w języku C++ 211
10. Operatory i funkcje matematyczne w języku C++ 214
11. Słowa kluczowe w języku C++ 215
12. Obiekty, klasy i metody w języku C++ 216
13. Systemy liczbowe 217
14. Narzędzia grafiki 3D w programie SketchUp 218
15. Podstawowe funkcje programu Microsoft Excel 220

Indeks 222

O czym jest podręcznik?

Podręcznik *NOWA Informatyka na czasie 3* pomoże ci zaprojektować i wydrukować modele 3D. Napiszesz proste programy w języku Python lub C++. Poznasz podstawy robotyki oraz nauczysz się, jak szybko i sprawnie przygotować korespondencję kierowaną do wielu odbiorców.



Czemu służą poszczególne elementy podręcznika?

Ważne w tym temacie

Informacje o najważniejszych zagadnieniach poruszanych w danym temacie.

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.

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 w podręczniku.
- Odsyłacz do omówienia danego zagadnienia poza podręcznikiem.
- Wskaźnik pojęcia; pojęcia obok wskaźników układają się w siatkę pojęć z danej lekcji.
- *** Oznaczenie stopnia trudności zadań.
- (F) Oznaczenie treści spoza podstawy.

Zadania

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

Projekt zespołowy

Zadanie do wykonania w grupie, pozwalające wykorzystać w praktyce wiedzę z kilku rozdziałów.

Dodatki

Przydatne informacje uzupełniające wiedzę z podręcznika.

1. Algorytmika i programowanie w języku Python

1 Algorytm Euklidesa

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. Współcześnie stosuje się go m.in. w kryptograficznym algorytmie RSA, wykorzystywanym np. w podpisach cyfrowych. W tym temacie użyjemy algorytmu Euklidesa do wykonywania operacji na ułamkach.

Ważne w tym temacie

- Wyznaczanie największego wspólnego dzielnika dwóch liczb z użyciem algorytmu Euklidesa
- Algorytmy do wykonywania działań na ułamkach zwykłych
- Funkcje języka Python, które nie zwracają wartości

1.1. 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 w wyniku zszycia wielu mniejszych kawałków tkanin. Wyobraźmy sobie, że przygotowujemy projekt małej kołdry o wymiarach 112 cm × 88 cm, która ma być uszyta z jak największych kwadratowych kawałków takiej samej wielkości (rys. 1.1). Jaką długość powinien mieć bok takiego kawałka?

Rys. 1.1. Patchworkowa kołdra uszyta z kwadratów z różnych materiałów

Zauważmy, 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 cm × 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, jak obliczyć, z ilu kawałków materiału w kształcie kwadratu o boku 8 cm będzie się składać patchworkowa kołdra o wymiarach 112 cm × 88 cm.

1.2. 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 dwóch różnych liczb bez potrzeby rozkładania ich na czynniki pierwsze. Algorytm wykorzystujący najbardziej intuicyjną metodę to **algorytm naiwny**.

Algorytm naiwny

Oto specyfikacja problemu:

Specyfikacja

Dane: dwie różne liczby całkowite dodatnie a i b.

Wynik: największy wspólny dzielnik liczb a i b.

Rozwiązanie 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 wtedy, gdy wspólny dzielnik będzie równy 1.

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 3–5.
   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.

Oto kod źródłowy programu realizującego podany algorytm:

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 wielkość liter. Jeśli stosujesz zmienną lub funkcję zawierającą w nazwie duże litery, to w dalszej części kodu musisz konsekwentnie używać ustalonej nazwy.

Funkcja w liniach 1.–6. zapisana jest w liniach 1.–6. Fragment kodu w liniach 2. i 3. służy temu, aby liczba zapisana w zmiennej `a` była mniejsza od liczby zapisanej w zmiennej `b`. Jeśli wartość zmiennej `a` jest większa od wartości zmiennej `b`, to wartości zmiennych są zamieniane. W języku Python taki efekt można uzyskać, gdy zapisze się nazwy zmiennych, rozdzielone przecinkiem, po prawej stronie znaku przypisania (=) w odwrotnej kolejności niż po lewej stronie znaku przypisania.

Kod realizujący kluczową ideę algorytmu znajduje się w liniach 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` – odpowiada za to instrukcja `return`.

Pętla `for` → s. 204

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`.

W linii 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 tej linii wywołana jest również funkcja `NWD`, zdefiniowana na początku programu.

Funkcja `str`

Rysunek 1.2 przedstawia przykład wywołania powyższego programu.

```
Wykonywanie: nwd.py
Podaj liczbę a: 112
Podaj liczbę b: 88
NWD(112, 88) = 8
>>>
Python
```

Rys. 1.2. Wywołanie programu wyznaczającego NWD dwóch liczb

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. 1.2, s. 10).

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 linii 11. wypisującej wynik bez zastosowania funkcji `str`, np.:

```
print("NWD(", a, ", ", b, ") =", NWD(a,b))
```

Zapamiętaj

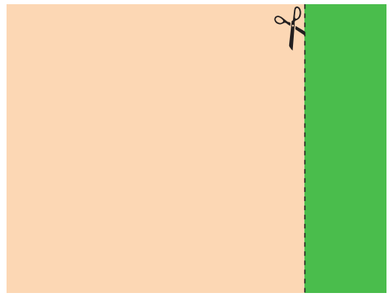
Funkcja `str` służy do przekształcania danych na napisy (ciągi znaków). Operator + pozwala na scalanie napisów i można go wykorzystać m.in. w funkcji `print`.

Warto wiedzieć

Wywołanie kodu `print(1, 2, 3)` wyświetli na ekranie: 1 2 3. Aby zmienić domyślny separator między argumentami funkcji `print`, należy użyć dodatkowego parametru `sep`. Na przykład dla kodu `print(1, 2, 3, sep=", ")` wyświetli się: 1, 2, 3.

Wycinanka Euklidesa

Wróćmy do przykładu wyznaczania długości boku kwadratowego kawałka kołdry o wymiarach 112 cm × 88 cm. Jak szybko można wyznaczyć wymiary takiego kwadratu bez rozkładania liczb 112 i 88 na czynniki pierwsze?

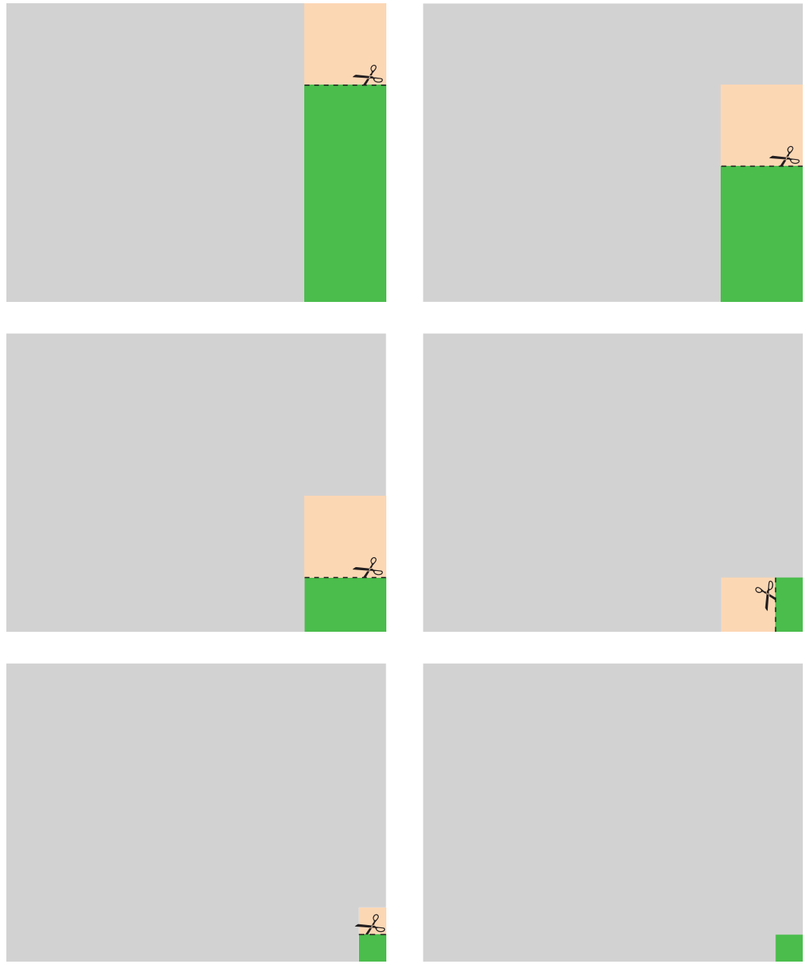


Rys. 1.3. Prostokąt o wymiarach 112 cm × 88 cm z odciętym kwadratem o boku 88 cm

Rysunek 1.3 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 1.4 na s. 12 i opisano w tabeli 1.1 na s. 12.

Warto wiedzieć

Odcinanie jak największego kwadratu od danego prostokąta to przykład zastosowania strategii algorytmicznej nazywanej podejściem zachłannym.



Rys. 1.4. 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	8

Tabela 1.1. Wymiary prostokątów i długości boków kwadratów w kolejnych krokach

Ćwiczenie 3

Zastosuj wycinankę Euklidesa dla prostokąta o bokach 21 cm × 12 cm. Użyj kartki w kratkę i nożyczek.

Dzięki wycinance Euklidesa wiemy, że kwadratami o boku 8 cm można wypełnić cały prostokąt o wymiarach 112 cm × 88 cm. 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 cm, gdyby istniał taki kwadrat dla prostokąta o wymiarach 112 cm × 88 cm. Tymczasem prostokąta o wymiarach 16 cm × 8 cm, z przedostatniego kroku, nie da się wypełnić kwadratami o boku dłuższym niż 8 cm.

To spostrzeżenie można przedstawić w postaci równości: $\mathrm{NWD}(a, b) = \mathrm{NWD}(a, b - a)$ dla $a < b$. Jest ona 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 – wersja z odejmowaniem

Algorytm naiwny ze s. 9 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.

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 dziele *Elementy*.

Algorytm Euklidesa w wersji z odejmowaniem można zapisać w postaci następującej listy kroków:

Algorytm Euklidesa – wersja z odejmowaniem

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.

Oto kod funkcji NWD, która realizuje algorytm Euklidesa:

Kod źródłowy funkcji realizującej algorytm Euklidesa w wersji z odejmowaniem

```
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 liczb 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.

Czy dla dowolnej pary liczb *a* i *b* w skończonej liczbie kroków osiągnie się 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 opisującymi długości **odcinków współmiernych**.

Odcinki współmierne to dwa odcinki, dla których istnieje trzeci odcinek, który w obu danych odcinkach mieści się całkowitą liczbę razy.

Ćwiczenie 5

a. Zmodyfikuj kod źródłowy programu *nwd.py* z ćwiczenia 2., 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 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 wykorzystuje się we współczesnej informatyce. Mimo że dzieło Euklidesa powstało ok. 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

Jeśli w algorytmie Euklidesa jedna liczba jest dużo większa od drugiej, to wielokrotnie powtarza się odejmowanie od niej tej samej mniejszej liczby.

W tabeli 1.2 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 1.2. Wyznaczanie NWD zgodnie z algorytmem Euklidesa – wersja z odejmowaniem

Warto wiedzieć

Operacja dzielenia, a więc i modulo, zajmuje 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.

Wielokrotne odejmowanie możemy zastąpić wyznaczeniem reszty z dzielenia większej liczby przez mniejszą – tak jak w tabeli 1.3.

Wartość zmiennej a	Wartość zmiennej b
196	42
196 % 42 = 28	42
28	42 % 28 = 14
28 % 14 = 0	14

Tabela 1.3. 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. Zauważ, że cztery działania zastąpiliśmy jednym.

Spójrzmy na ostatni wiersz tabeli 1.3. 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**:

Algorytm Euklidesa – wersja 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, która realizuje algorytm 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
```

Patrząc na tabelę 1.3 na s. 15, zauważamy, że dzielenie z resztą jest wykonywane naprzemiennie (a % b, b % a, a % b itd.). Dlatego kod funkcji NWD można zapisać bez użycia instrukcji warunkowej.

Kod źródłowy funkcji realizującej algorytm Euklidesa w wersji z dzieleniem (bez instrukcji warunkowej)

```
1.  def NWD(a, b):
2.      while b != 0:
3.          a, b = b, a % b
4.      return a
```

W linii 3. zapisujemy w zmiennej a wartość zmiennej b, natomiast w zmiennej b – resztę z dzielenia a przez b. Gdy reszta będzie równa 0, pętla zakończy działanie. NWD to ostatnio zapamiętana wartość zmiennej a.

Ćwiczenie 6

Zmodyfikuj kod programu *euklides.py* otrzymany w ćwiczeniu 5., używając funkcji NWD realizującej algorytm Euklidesa w wersji z dzieleniem (bez instrukcji warunkowej). Zapisz go pod nazwą *euklides_dzielenie.py*. Uruchom program dla różnych par liczb.

Zapamiętaj

Algorytm Euklidesa pozwala znaleźć największy wspólny dzielnik dwóch liczb naturalnych bez szukania wspólnych dzielników tych liczb.

1.3. Działania na ułamkach zwykłych

Napiszemy teraz programy, które pozwolą wykonać wybrane działania na ułamkach zwykłych 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, należy podzielić liczby e i f przez NWD(e, f). Oto specyfikacja problemu:

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 b != 0:
3.          a, b = b, a % b
4.      return a
5.
6.  def suma(a, b, c, d):
7.      e = a * d + c * b
8.      f = b * d
9.      nwd_ef = NWD(e, f)
10.     print(str(e // nwd_ef) + "/" + str(f // nwd_ef))
11.
12. licz_1 = int(input("Liczba 1 - licznik: "))
13. mian_1 = int(input("Liczba 1 - mianownik: "))
14. licz_2 = int(input("Liczba 2 - licznik: "))
15. mian_2 = int(input("Liczba 2 - mianownik: "))
16.
17. print("\n" + str(licz_1) + "/" + str(mian_1) + " + ", end="")
18. print(str(licz_2) + "/" + str(mian_2) + " = ", end="")
19. suma(licz_1, mian_1, licz_2, mian_2)
```

W liniach 1.–4. zdefiniowano funkcję wyznaczającą NWD zgodnie z algorytmem Euklidesa (w wersji z dzieleniem). W liniach 6.–10. 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 w linii 19. znajduje się wywołanie funkcji suma bez funkcji print.

Funkcja niezwracająca wartości

Ćwiczenie 7

Zapisz kod źródłowy ze s. 17 w pliku *suma_ulamkow.py* i uruchom program dla różnych par ułamków.

Dodawanie ułamków – metoda szkolna

Kiedy na lekcjach matematyki dodaje się ułamki, sprowadza się je najpierw do wspólnego mianownika, którym jest 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**, czyli ich największym wspólnym dzielnikiem jest liczba 1, to $\mathrm{NWW}(a, b) = a \cdot b$.

Liczby względnie pierwsze

Zależność między NWD a NWW

Istnieje **zależność między NWD a NWW** liczb a i b:

$$\mathrm{NWW}(a,b)=\frac{a\cdot b}{\mathrm{NWD}(a,b)}$$

Można ją 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))
```


W kodzie korzystamy z algorytmu Euklidesa zastosowanego w funkcji `NWD` oraz ze wzoru określającego zależność między NWD i NWW (kod funkcji `NWD` – s. 16).

Ćwiczenie 9

W pliku *suma_ulamkow2.py* zapisz kod źródłowy programu dodającego ułamki metodą szkolną. Do wykonania ćwiczenia 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 postępuje się na lekcjach matematyki):

$$\frac{a}{b}\cdot\frac{c}{d}=\frac{\frac{a}{\mathrm{NWD}(a,d)}\cdot\frac{c}{\mathrm{NWD}(b,c)}}{\frac{b}{\mathrm{NWD}(b,c)}\cdot\frac{d}{\mathrm{NWD}(a,d)}}=\frac{e}{f}$$

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.

Zadania

* **1** Napisz program, który po wczytaniu wymiarów (w centymetrach) patchworkowej kołdry 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. Część ułamkową program powinien podawać w postaci skróconej.

* **3** Napisz program, który – korzystając z algorytmu Euklidesa – będzie sprawdzać, czy dane dwie liczby naturalne są liczbami względnie pierwszymi.

** **4** Napisz program, który wczyta licznik i mianownik ułamka o wartości mniejszej niż 1/2 i sprawdzi, 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 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.

** **7** Uzupełnij program *Suma ułamków* ze s. 17 o wyodrębnianie części całkowitej i części ułamkowej dla sumy, która jest ułamkiem niewłaściwym.

** **8** Napisz program zawierający funkcję o nagłówku `NWW(a, b)`, która będzie wyznaczać najmniejszą wspólną wielokrotność liczb a i b.

** **9** Zapisz specyfikację problemu dzielenia dwóch ułamków. Następnie napisz listę kroków algorytmu i program zawierający funkcję `iloraz`, wykonującą dzielenie.

*** **10** Napisz program, który wczyta dwie dodatnie i względnie pierwsze liczby a i b, a następnie wyznaczy najmniejszą możliwą dodatnią liczbę całkowitą x, dla której ułamek $\frac{a+x}{b+x}$ jest nieskracalny.

*** **11** Zmodyfikuj program *Suma ułamków* ze s. 17 tak, aby umożliwiał dodanie trzech ułamków.

*** **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 NWD 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.

2 Obliczanie wartości elementów ciągu

Ciągi liczbowe służą m.in. do opisywania modeli świata rzeczywistego w komputerowym świecie wirtualnym. W mniej lub bardziej dokładny sposób przedstawiają one różnego rodzaju zjawiska. Na przykład pozwalają opisać zmiany temperatury w wybranym miejscu na świecie czy wzrost liczby użytkowników w sklepie internetowym. Do wyznaczania kolejnych liczb tworzących ciąg liczbowy możemy w programowaniu użyć techniki iteracji, czyli wielokrotnego powtarzania instrukcji zgodnie z określonym wzorem.

Ważne w tym temacie

- Definicja ciągu liczbowego
- Iteracyjne wyznaczanie wyrazów ciągu
- Ciąg Fibonacciego

2.1. Definicja ciągu liczbowego

Pojęcia ciągu używamy, mówiąc o obiektach, których kolejność jest ustalona. **Ciąg liczbowy** to taki, którego elementami, zwanymi wyrazami ciągu, są liczby.

Ciąg liczbowy

Kolejne wyrazy ciągu znajdują się na pozycjach określonych za pomocą liczb naturalnych. Na przykład pierwszy wyraz ciągu znajduje się na pozycji pierwszej, piąty wyraz – na piątej, a n-ty wyraz ciągu – na pozycji n-tej. Wyrazy ciągu zapisujemy symbolicznie jako a_1, a_2, a_3, ... Zapis a_n oznacza n-ty wyraz ciągu.

Przykład **skończonego ciągu** liczbowego, czyli takiego, który ma ściśle określoną liczbę elementów, tworzą numery wagonów składu pociągu (rys. 2.1).

Ciąg skończony

Rys. 2.1. Numery wagonów jako przykład skończonego ciągu liczbowego

Ciąg, w którym nie istnieje ostatni element, nazywamy **ciągiem nieskończonym**. Przykładem takiego ciągu jest: 1, 2, 4, 8, 16, … Zauważ, że iloraz dwóch kolejnych wyrazów jest równy 2. Taki ciąg można opisać wzorem:

Ciąg nieskończony

$$a_n = \begin{cases} 1 & \text{dla } n = 1 \\ 2a_{n-1} & \text{dla } n > 1 \end{cases}$$

Warto wiedzieć

Ciąg liczbowy o takiej własności, że iloraz dwóch kolejnych wyrazów jest stały, nazywa się ciągiem geometrycznym.

Przepis na ciasto francuskie

Ciasto francuskie ma w sobie dużo powietrza. Zawdzięcza to warstwowej strukturze uzyskiwanej w wyniku wielokrotnego składania i wałkowania ciasta (rys. 2.2).

Aby przygotować ciasto francuskie, należy wiele razy powtórzyć następujące czynności:

- rozwałkować ciasto na długość równą trzykrotności jego szerokości,
- złożyć ciasto dwa razy, aby uzyskać trzy warstwy, i schłodzić je.

Rys. 2.2. Wałkowanie i składanie ciasta francuskiego

Ciasto powstałe w efekcie pierwszego wałkowania możemy nazwać ciastem początkowym o numerze 1. Z niego powstaje 3-warstwowe ciasto o numerze 2, a z niego z kolei 9-warstwowe ciasto o numerze 3 itd.

Zauważmy, że i-te ciasto powstaje z ciasta o numerze $i - 1$, które po złożeniu tak, żeby miało trzy warstwy, zostaje rozwałkowane. Ciąg określający liczbę warstw ciasta to: 1, 3, 9, 27, ... Można go opisać w następujący sposób:

$$c_n = \begin{cases} 1 & \text{dla } n = 1 \\ 3c_{n-1} & \text{dla } n > 1 \end{cases}$$

Ćwiczenie 1

Oblicz liczbę warstw ciasta przy założeniu, że procedura wałkowania jest powtarzana sześć razy.

Program wyznaczający *n*-ty wyraz ciągu

Napiszemy program wyznaczający n-ty wyraz ciągu. Oto specyfikacja problemu:

Specyfikacja

Dane: liczba naturalna n.

Wynik: n-ty wyraz ciągu opisanego wzorem:

$$c_n = \begin{cases} 1 & \text{dla } n = 1 \\ 3c_{n-1} & \text{dla } n > 1 \end{cases}$$

Do wyznaczenia n-tego wyrazu ciągu zastosujemy pętlę `for`, gdyż liczba cykli pętli potrzebnych do wyznaczenia n-tego wyrazu ciągu jest z góry określona: będzie ich $n - 1$.

```
1.  def Ciag(n):
2.      c = 1
3.      for i in range(2, n+1):
4.          c = 3 * c
5.      return c
```

Kod źródłowy programu wyznaczającego *n*-ty wyraz ciągu

W linii 2. zmiennej `c` przypisywana jest wartość 1. Jeśli użytkownik wywoła funkcję `Ciag` dla wartości 1, to pętla z linii 3.–4. nie wykona się ani razu, ponieważ nie będzie spełniony warunek $n + 1 > 2$. W pozostałych przypadkach pętla wykona się $n - 1$ razy.

Ćwiczenie 2

a. Zapisz kod źródłowy programu *Ciag* wyznaczającego n-ty wyraz ciągu. Uruchom program dla różnych wartości.

b. Zmodyfikuj kod funkcji `Ciag` tak, aby program wyznaczał n-ty wyraz ciągu a_n zdefiniowanego na s. 21.

Zapamiętaj

Ciąg liczbowy to ciąg, którego wyrazami są liczby. Kolejne wyrazy ciągu znajdują się na pozycjach określonych za pomocą liczb naturalnych. Do obliczenia wartości n-tego wyrazu ciągu stosujemy pętlę `for`, gdyż liczba cykli pętli potrzebnych do wyznaczenia n-tego wyrazu ciągu jest z góry określona: będzie ich $n - 1$.

2.2. Ciąg Fibonacciego

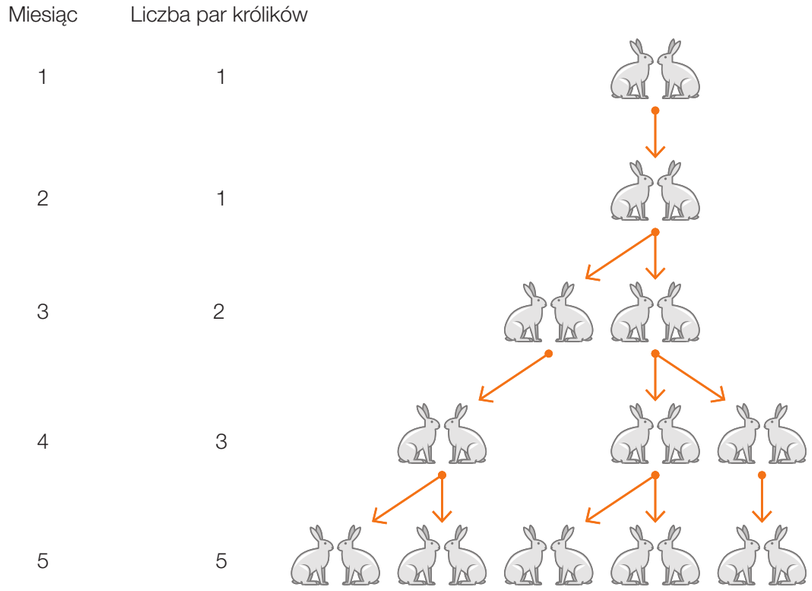
W 1202 r. Leonardo z Pizy, zwany Fibonaccim, przedstawił zadanie problemowe, które opisywało wzrost populacji królików. Pewien gospodarz kupił parę królików. Ile par królików w sumie będzie miał po 12 miesiącach, jeśli każda para rodzi co miesiąc po jednej parze, która staje się zdolna do rozmnażania po miesiącu życia?

Przeanalizujmy, jak rośnie liczba par królików w hodowli. W pierwszym miesiącu mamy parę młodych królików, która na początku drugiego miesiąca staje się płodna. Na początku trzeciego miesiąca tej parze urodzi się para młodych. Razem będą więc 2 pary królików. W czwartym miesiącu ze starszej pary zrodzi się kolejna para – mamy więc 2 + 1 = 3 pary. Ile par królików będzie na początku kolejnego, tj. piątego miesiąca?

Warto wiedzieć

Leonardo z Pizy (1175–1250) był włoskim matematykiem. W młodości towarzyszył ojcu, który był kupcem, w podróżach handlowych i uczył się matematyki u arabskich nauczycieli. W 1202 r. opublikował swoje pierwsze dzieło matematyczne *Liber abaci* (*Księga rachunków*). Zawierało ono prawdopodobnie całą ówczesną wiedzę o arytmetyce. Na stronie tytułowej Leonardo podpisał się *filius Bonacci* (syn Bonacciego). W XIX w. historycy nadali mu przydomek Fibonacci.

Sposób, w jaki zmienia się populacja królików w kolejnych miesiącach, wyjaśnia rysunek 2.3.



Rys. 2.3. Para królików i jej potomstwo po pięciu pierwszych miesiącach

Oznaczmy liczbę par na początku n-tego miesiąca jako f_n. Otrzymamy ciąg liczb: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, … (patrz rys. 2.3). W n-tym miesiącu ($n > 2$) liczba par królików jest równa sumie par z dwóch poprzednich miesięcy: $n - 2$ i $n - 1$. Współcześnie zwykle dodaje się na początku ciągu dodatkowy wyraz $f_0 = 0$.

Ciąg Fibonacciego

Otrzymany ciąg nazywamy **ciągiem Fibonacciego**. Początkowe dwa wyrazy tego ciągu to 0 i 1, a każdy następny jest sumą poprzednich dwóch. Ciąg Fibonacciego ma następującą definicję:

$$f_n = \begin{cases} 0 & \text{dla } n = 0 \\ 1 & \text{dla } n = 1 \\ f_{n-1} + f_{n-2} & \text{dla } n > 1 \end{cases}$$

A to ciekawe

Fibonacci w muzyce

Ciąg Fibonacciego jest chętnie wykorzystywany przez kompozytorów muzyki klasycznej. Zastosowanie kolejnych wyrazów ciągu do wyznaczania punktów kulminacyjnych utworu czy długości następujących po sobie fraz muzycznych pozwala uzyskać intrygującą, a jednocześnie harmonijnie brzmiącą kompozycję.

Napiszemy program obliczający dany wyraz ciągu Fibonacciego. Przyjmijmy następującą specyfikację problemu:

Specyfikacja

Dane: liczba całkowita $n \geqslant 0$.

Wynik: n-ty wyraz ciągu Fibonacciego.

Warto wiedzieć

Definicja ciągu, w której podajemy, w jaki sposób kolejne elementy są tworzone z poprzednich, w matematyce nazywa się wzorem rekurencyjnym ciągu.

Program *Fibonacci*

W każdym momencie wyznaczania wartości nowych wyrazów ciągu wystarczy wiedza o ostatnich dwóch jego wyrazach. Zapamiętajmy w zmiennej `a` wartość ostatniego wyrazu, a w zmiennej `b` – przedostatniego. Jeśli ustalimy początkową wartość zmiennej `b` jako $f_0 = 0$, wartość zmiennej `a` jako $f_1 = 1$ oraz wykorzystamy dodatkową zmienną pomocniczą `pom = a + b`, to będziemy mogli generować kolejne liczby ciągu Fibonacciego. Zmienna `pom` przechowa wartość kolejnej liczby Fibonacciego. Oto kod źródłowy funkcji `Fib`, która wyznacza n-ty wyraz ciągu Fibonacciego:

```
1.  def Fib(n):
2.      a = 1
3.      b = 0
4.      for i in range(0, n):
5.          pom = a + b
6.          b = a
7.          a = pom
8.      return b
```

Kod źródłowy funkcji `Fib` w programie *Fibonacci*

W liniach 2.–3. nadajemy początkowe wartości zmiennym `a` i `b`. W pętli (linie 4.–7.) obliczamy wartość kolejnego wyrazu zapisanego w zmiennej `pom`. Następnie zmieniamy wartości zmiennych tak, aby w zmiennej `b` znalazł się wyraz zapamiętany wcześniej w zmiennej `a` (linia 6.), natomiast w zmiennej `a` następny wyraz (linia 7.).

Po n iteracjach zmienna `b` przyjmie wartość f_n. Zwróćmy uwagę na to, że dla $n = 0$ pętla nie wykona się ani razu.

Warto wiedzieć

Ciąg Fibonacciego można odnaleźć w przyrodzie – kwiat przebiśniegu ma 3 płatki, kaczeńca – 5, ostróżki – 8, nagietka – 13, astra – 21, a stokrotki – 34, 55 lub 89.

Ćwiczenie 3

a. Napisz program obliczający dany wyraz ciągu Fibonacciego według podanej wcześniej specyfikacji. Wykorzystaj funkcję `Fib`. Sprawdź działanie programu dla liczb: 5, 10 i 45.

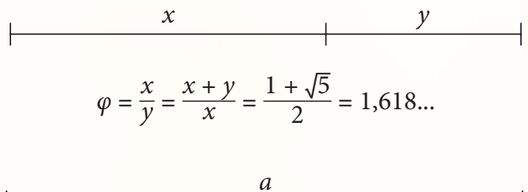
b. Zapisz kod funkcji z użyciem krotki, tj. linie 5.–7. zastąp pojedynczymi instrukcjami.

ZŁOTA PROPORCJA

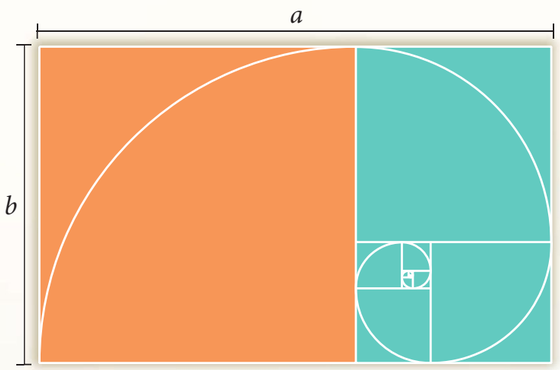
Złota proporcja, nazywana też złotym podziałem, znana jest od wieków. Przyrodnicy zachwycają się jej pięknem, którego przykłady można odnaleźć w naturze. Zainspirowało to wiele osób do poszukiwania i stosowania złotego podziału w różnych dziedzinach życia.

Czym jest złota proporcja?

Złota proporcja wyraża podział odcinka na dwie części w taki sposób, że stosunek jego dłuższej części (x) do krótszej (y) jest taki sam jak stosunek długości całego odcinka ($x + y$) do części dłuższej (x). Stosunek ten oznacza się grecką literą φ (czytaj: fi). Co ciekawe, przybliżeniami liczby φ są ilorazy kolejnych liczb ciągu Fibonacciego: 3/2, 5/3, 8/5 itd.



$$\varphi = \frac{x}{y} = \frac{x+y}{x} = \frac{1+\sqrt{5}}{2} = 1{,}618...$$



Jeżeli w prostokąt o bokach a, b będących w złotej proporcji wpiszemy kwadrat (zaznaczony na pomarańczowo), to uzyskamy mniejszy prostokąt (na rysunku turkusowy) o bokach pozostających w złotej proporcji. Jeżeli powtórzymy tę procedurę wpisywania kwadratów w kolejne mniejsze prostokąty i w każdy kwadrat wpiszemy ćwiartkę okręgu, otrzymamy złotą spiralę.

Złota proporcja w naturze

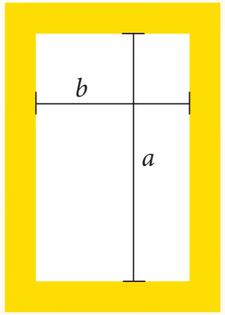
Zaskakująco wiele przykładów złotego podziału odnajdziemy w przyrodzie. Zbliżone do niego są m.in. proporcje w ciele człowieka. Badacze zauważyli też występowanie złotej proporcji na poziomie atomowym.

Huragan obserwowany z kosmosu ma kształt zbliżony do złotej spirali

Złota spirala w muszli

Złota proporcja w projektowaniu

Również graficy i projektanci wykorzystują złotą proporcję. Odnajdziemy ją w logotypach wielu słynnych marek, takich jak Toyota. W złotej proporcji pozostają także m.in. wymiary kart płatniczych.



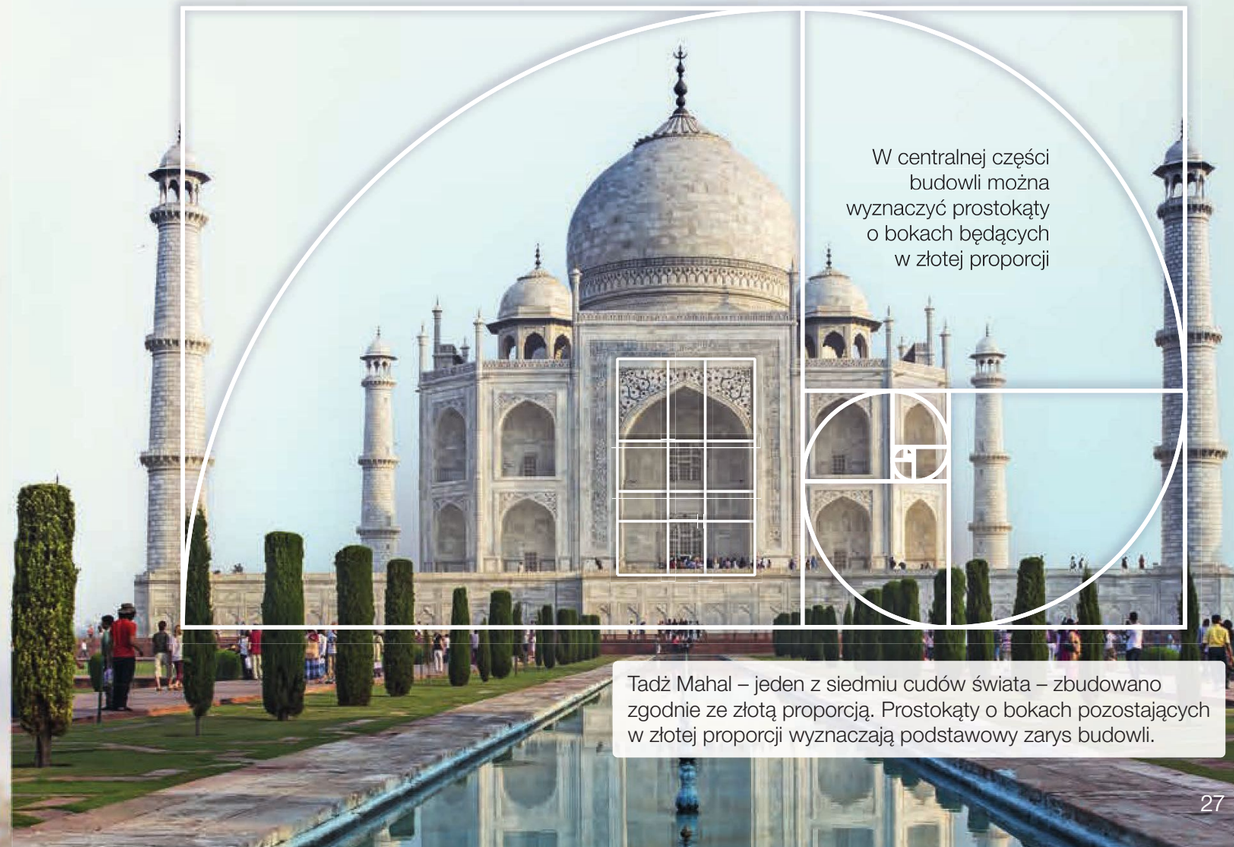
Wewnętrzny prostokąt w logo National Geographic zachowuje złotą proporcję

Na stronie wyszukiwarki Google złoty podział występuje zarówno w logotypie, jak i w układzie strony

$$\frac{a}{b} \approx \frac{b}{c} \approx \varphi$$

Złota proporcja w architekturze i sztuce

Złoty podział stosowano już w starożytnych konstrukcjach, m.in. w Wielkiej Piramidzie i Partenonie. Można go również odnaleźć w obrazach Leonarda da Vinci i Salvadora Dalego.



W centralnej części budowli można wyznaczyć prostokąty o bokach będących w złotej proporcji

Tadż Mahal – jeden z siedmiu cudów świata – zbudowano zgodnie ze złotą proporcją. Prostokąty o bokach pozostających w złotej proporcji wyznaczają podstawowy zarys budowli.

Zadania

* **1** Napisz program, który wypisze na ekranie ciąg liczb: 1, 3, 5, 7, 9, …, 99.

* **2** Napisz program, który wypisze na ekranie ciąg liczb: 1, –2, 4, –8, 16, –32, …, 4096.

* **3** Napisz program, który wypisze na ekranie wszystkie liczby dwucyfrowe.

* **4** Napisz program wypisujący na ekranie kolejne trzycyfrowe liczby parzyste.

* **5** Napisz program, który wypisze 10 elementów ciągu określonego wzorem:

$$b_n = \begin{cases} 1 & \text{dla } n = 1 \\ 1 + 2b_{n-1} & \text{dla } n > 1 \end{cases}$$

** **6** Napisz program, który po podaniu liczby n wypisze na ekranie ciąg liczb: n, $-2n$, $4n$, $-8n$, $16n$, $-32n$, …, $4096n$.

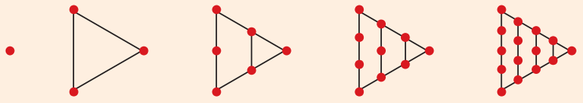
** **7** Napisz program wyznaczający sumę n kolejnych liczb nieparzystych. Na przykład dla $n = 3$ program powinien policzyć sumę $1 + 3 + 5$ i wypisać 9.

** **8** Napisz program, który po podaniu liczby n ($n > 2$) wypisze na ekranie ciąg o dwóch pierwszych wyrazach równych 1. Każdy kolejny wyraz ciągu powinien powstać przez zsumowanie poprzednich trzech wyrazów. Na przykład dla $n = 9$ program powinien wypisać: 1, 1, 2, 4, 7, 13, 24, 44, 81.

*** **9** Iloczyn liczby naturalnej n przez wszystkie liczby naturalne dodatnie mniejsze od tej liczby nazywa się silnią liczby n i oznacza jako $n!$. Napisz program obliczający wartość $n!$.

*** **10** Napisz program, który wyświetli wartości kolejnych ilorazów: f_3/f_2, f_4/f_3, f_5/f_4 itd. aż do momentu, gdy sąsiednie ilorazy różnić się będą o mniej niż 0,0001. Wartości f_n są kolejnymi wyrazami ciągu Fibonacciego.

*** **11** Pitagorejczycy odczuwali mistyczny związek z ciągiem liczb 1, 3, 6, 10, 15, 21, … Pierwszym wyrazem tego ciągu jest 1. Aby obliczyć n-ty wyraz, należy dodać n do poprzedniego wyrazu.



Liczby w tym ciągu nazywane są liczbami trójkątnymi, ponieważ można je sobie wyobrazić jako układ trójkątnych obiektów. Napisz program, który wyznaczy liczbę wszystkich punktów konstrukcji n-tej liczby trójkątnej.

3 Algorytmy na tekstach

Błędy językowe, literówki i inne pomyłki w tekście zwykle świadczą o niewiedzy lub nieuwadze piszącego. Czasami mogą mieć naprawdę poważne konsekwencje. Na przykład skład leku i sposób jego dawkowania powinny być zawsze szczegółowo i precyzyjnie opisane w ulotce. W takim miejscu nawet przypadkowa zmiana znaków w tekście może być kluczowa. Programy przetwarzające tekst oferują wiele narzędzi, m.in. wyszukiwanie, automatyczne podkreślanie powtarzających się wyrazów czy możliwość wielokrotnej zmiany podanej frazy na inną. W tym temacie nauczysz się przetwarzać teksty z wykorzystaniem programów napisanych w języku Python.

Ważne w tym temacie

- Kodowanie znaków
- Algorytmy porównywania tekstów oraz wyszukiwania wzorca w tekście
- Używanie typu tekstowego `str` w języku Python
- Stosowanie funkcji `chr`, `len`, `ord` oraz metod `find` i `rfind` dla klasy `str`

3.1. Litery i inne znaki jako liczby

W serwisach społecznościowych często używa się piktogramów zwanych **emoji**, czyli symboli rysunkowych uzupełniających tekst o graficzne wyrażenie emocji lub przedstawienie jakiegoś obiektu – przedmiotu, miejsca, rośliny itp.

Emoji

Jeśli umieścimy na portalu społecznościowym emoji 🌻, to widzimy je dlatego, że piktogram ten znajduje się w zbiorze znaków czcionki na naszym urządzeniu. Emoji 🌻 jest w takim zbiorze na pozycji o numerze 127 803. Mówimy, że jest to **kod liczbowy znaku** (w tym przypadku piktogramu). Kody wszystkich piktogramów, a także wszelkich innych znaków (liter oraz symboli, m.in. matematycznych i muzycznych) znajdują się w **tablicy znaków Unicode**.

Kod liczbowy znaku

Tablica znaków Unicode

Ćwiczenie 1

a. Uruchom edytor tekstu i wybierz czcionkę Segoe UI Emoji lub Segoe UI Symbol. Następnie, przytrzymując wciśnięty lewy klawisz **Alt**, wybierz w części numerycznej klawiatury kolejno cyfry: **128190**. Jaki symbol kryje się pod tym kodem?

b. Korzystając z dostępnych serwisów internetowych, sprawdź, jaki kod ma emoji z piłką nożną ⚽.

Warto wiedzieć

Emoji wywodzą się z japońskich telefonów komórkowych z końca lat 90. XX w. Nazwa pochodzi od japońskich słów: 絵 (obraz) i 文字 (litera).

Dobra rada

Większość laptopów ma klawisze, które symulują numeryczną część tradycyjnej klawiatury QWERTY. Zazwyczaj są to klawisze **7**, **8**, **9**, **U**, **I**, **O**, **J**, **K**, **L** oraz **M**. Klawisz **Num Lock** znajdziesz zwykle w prawej górnej części klawiatury laptopa.

Tabela 3.1 przedstawia fragment tablicy Unicode dla emoji o kodach od 127 800 do 127 819.

	0	1	2	3	4	5	6	7	8	9
127 80_	🌸	🌹	🌺	🌻	🌼	🌽	🌾	🌿	🍀	🍁
127 81_	🍂	🍃	🍄	🍅	🍆	🍇	🍈	🍉	🍊	🍋

Tabela 3.1. Fragment tablicy znaków Unicode zawierający wybrane piktogramy

Tablica ASCII

Podstawowe znaki, takie jak litery alfabetu łacińskiego, cyfry, znaki interpunkcyjne i symbole (np. @ i ^), które możemy wprowadzić do komputera bezpośrednio z klawiatury, znajdziemy w początkowej części tablicy Unicode (na pozycjach od 0 do 127). Jest ona nazywana **tablicą ASCII** (skrót od ang. *American Standard Code for Information Interchange*). System kodowania znaków ASCII opracowano w latach 60. XX w. na potrzeby telekomunikacji. Powstałe później systemy kodowania znaków są w pełni kompatybilne z ASCII.

Typ danych `str` w języku Python

Typ `str`

Napis (łańcuch znaków, łańcuch)

W języku Python teksty przetwarzamy z użyciem **typu danych** `str`. Teksty nazywa się zazwyczaj **napisami**, **łańcuchami znaków** (ang. *strings of characters*) albo po prostu **łańcuchami**.

Łańcuchy znaków przechowuje się w zmiennej typu `str`, czyli w strukturze danych, w której zapisane są kody Unicode kolejnych znaków łańcucha. W tabeli 3.2 przedstawiono fragment tablicy Unicode ze znakami ASCII, które można wyświetlić na ekranie.

Warto wiedzieć

Kod ASCII jest czasami wykorzystywany w filmach fabularnych. Na przykład w filmie *Marsjanin* główny bohater otrzymał z NASA zakodowaną wiadomość: `48 4F 57 41 4C 49 56 45`. Korzystając z tablicy kodów ASCII, odczytał ją jako `HOW ALIVE`, czyli pytanie o to, jak udało mu się przeżyć na Marsie.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Tabela 3.2. Wybrane znaki ASCII – fragment tablicy Unicode

Kod ASCII

Zazwyczaj tablicę ASCII przedstawia się tak, że znaki w niej zapisane są pogrupowane po 16. W opisie kolumn pojawiają się kolejne cyfry systemu szesnastkowego (od 0 do F). Aby odczytać kod liczbowy, czyli **kod ASCII** znaku, parę liczb, czyli numer wiersza i numer kolumny, traktuje się jako kod znaku zapisany w systemie szesnastkowym. Na przykład znak @ ma kod 40_{16}, czyli 64 dziesiętnie, a znak spacji ma kod 20_{16}, czyli 32 dziesiętnie.

Program *Znaki*. Funkcja `chr`

Napiszemy teraz program, który wyświetli fragment tablicy Unicode pokazany w tabeli 3.2 na s. 30. Kody widocznych tam znaków mieszczą się między 32 a 126. Program powinien wyświetlić znaki z odstępami w wierszach po 16 znaków (nie licząc spacji).

Oto kod źródłowy programu *Znaki*:

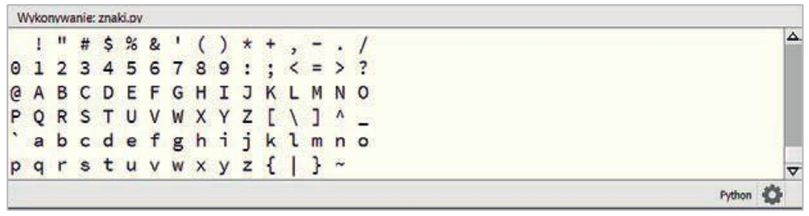
Kod źródłowy programu *Znaki*

```
1.  for i in range(32, 127):
2.      znak = chr(i)
3.      print(znak, end=" ")
4.      if i % 16 == 15:
5.          print()
6.  print()
```

Funkcja `chr`

W rozwiązaniu wykorzystaliśmy wbudowaną w interpreter **funkcję** `chr` (od ang. *character* – znak), która dla podanej liczby zwraca odpowiadający jej znak w tablicy Unicode (jednoznakowy napis typu `str`).

W liniach 1.–5. wykonywana jest pętla `for`, która realizuje wyświetlanie znaków w następujący sposób: zmiennej `znak` jest przypisywany odpowiedni znak ASCII z wykorzystaniem funkcji `chr`. Linia 3. odpowiada za wypisanie na ekranie znaku i pojedynczej spacji. Zapisy w liniach 4. i 5. służą temu, aby znaki wyświetlić w wierszach po 16 znaków. Funkcja `print` wywołana bez żadnego argumentu wstawi wyłącznie przeniesienie do kolejnego wiersza.



Rys. 3.1. Wywołanie programu *Znaki*

Ćwiczenie 2

W pliku *znaki.py* zapisz kod źródłowy programu, który wyświetli fragment tablicy Unicode pokazany w tabeli 3.2 na s. 30. Sprawdź działanie programu.

Układ tablicy Unicode zawierającej kody ASCII jest starannie przemyślany. Na przykład pary wielkich i małych liter („A” i „a” itd.) znajdują się w tych samych kolumnach i są oddalone o dwa wiersze (różnica kodów wynosi 32). Dlatego łatwo z kodu małej litery przejść do kodu odpowiadającej jej wielkiej litery i na odwrót.

Warto wiedzieć

Kod ASCII zawarty w początkowej części tablicy Unicode był pierwotnie przeznaczony na potrzeby usługi telegraficznej zwanej teleksem. Nadawca zapisywał tekst na klawiaturze urządzenia zwanego dalekopisem, a odbiorca otrzymywał wydruk. Dlatego tablica Unicode na pozycjach od 0 do 31 i na pozycji 127 zawiera kody przeznaczone do sterowania drukarką.

Ćwiczenie 3

Zmodyfikuj kod źródłowy programu *Znaki* tak, aby wyświetlił on znaki, które w tablicy Unicode mają kody liczbowe między 128 000 a 128 287.

Program *Znaki 2*. Funkcja `ord`

Gdybyśmy od razu wiedzieli, że program ma wyświetlić znaki znajdujące się w tablicy Unicode między spacją a znakiem ~ (patrz rys. 3.1, s. 31), to kod źródłowy mógłby wyglądać następująco:

Kod źródłowy programu *Znaki 2*

```
1.  znak = " "
2.  kod = ord(znak)
3.  kod2 = ord("~")
4.
5.  while kod <= kod2:
6.      print(znak, end=" ")
7.      if kod % 16 == 15:
8.          print()
9.      kod = kod + 1
10.     znak = chr(kod)
11. print()
```

Dobra rada

W języku Python możesz umieszczać łańcuchy znaków (również pojedyncze znaki) zarówno w cudzysłowach, jak i w apostrofach. Oba zapisy znaczą to samo.

W linii 1. zmiennej `znak` przypisujemy znak spacji. W liniach 2.–3. wykorzystamy wbudowaną **funkcję** `ord` (od ang. *ordinal* – liczba porządkowa), która dla podanego znaku zwraca liczbę określającą jego pozycję w tablicy Unicode. Jest to funkcja odwrotna do **funkcji** `chr`.

Funkcja `ord`

Funkcja `chr` → s. 31

Wygodnie będzie zastosować pętlę, która będzie wykonywana, dopóki liczba określająca kod Unicode danego znaku będzie mniejsza od kodu znaku ~ lub jemu równa. Wykorzystamy więc **pętlę** `while`.

Pętla `while` → s. 204

W pętli wypisywane są: znak (wartość zmiennej `znak`) oraz spacja (linia 6.). Zapisy w liniach 7.–8. odpowiadają za wypisywanie znaków po szesnaście, a w liniach 9.–10. – za przypisanie zmiennej `znak` kolejnego znaku z tablicy Unicode.

Warto wiedzieć

W polskim układzie klawiatury, żeby uzyskać znak **~** (tylda), trzeba przytrzymać klawisz **Shift**, nacisnąć znak tyldy, a potem spację. Wynika to z tego, że pierwsze klawiatury nie rozróżniały prawego klawisza **Alt** od lewego. Polskie znaki uzyskiwało się, wciskając jednocześnie klawisze **Shift**, tyldy i odpowiedniej litery, np. **Shift + ~ + A** dla litery ą.

Ćwiczenie 4

a. Zapisz kod źródłowy powyższego programu pod nazwą *znaki2.py* i sprawdź działanie programu.

b. Napisz program, który wyświetli litery od A do Z w dwóch wierszach w następujący sposób:

```
ABCDEFGHIJKLM
ZYXWVUTSRQPON
```


3.2. Szukanie literówek

Zapobiegać powstawaniu literówek można na różne sposoby. Na przykład w edytorze tekstu słowa zapisywane przez użytkownika zwykle są porównywane ze słowami znajdującymi się w słowniku. Innym sposobem jest wymaganie od użytkownika dwukrotnego wpisania tekstu, np. PIN-u, adresu e-mail lub hasła.

Warto wiedzieć

Edytory tekstu, a także gry słowne (np. Scrabble, Literaki czy Słowotok) zazwyczaj mają swoje oficjalne słowniki, w których sprawdzane są podane wyrazy.

Poprawność adresu e-mail – porównanie dwóch adresów

Podczas zakładania konta w serwisie internetowym często wymaga się od nas, abyśmy dwukrotnie podali adres poczty elektronicznej. Napiszemy program, który będzie symulował taką rejestrację w serwisie internetowym.

Zapiszmy specyfikację problemu:

Specyfikacja

Dane: cztery napisy oznaczające kolejno: imię, nazwisko oraz adres e-mail (podany dwukrotnie).

Wynik: komunikat „Hasło do serwisu wysłaliśmy na adres <adres_e-mail>”, jeśli wpisane adresy są identyczne, w przeciwnym wypadku komunikat „Podane adresy e-mail są różne!”.

Dobra rada

Pamiętaj, że w języku Python rozróżnia się wielkość liter. Nazwy utworzonej zmiennej używaj dokładnie tak samo.

Oto kod źródłowy programu *Porównanie adresów*:

Kod źródłowy programu *Porównanie adresów*

```
1.  INF_OK = "Hasło do serwisu wysłaliśmy na adres "
2.  INF_BLAD = "Podane adresy e-mail są różne!"
3.
4.  imie = input("Podaj imię: ")
5.  nazwisko = input("Podaj nazwisko: ")
6.
7.  adres_1 = input("Podaj adres e-mail: ")
8.  adres_2 = input("Powtórz adres e-mail: ")
9.
10. print()
11. if adres_1 == adres_2:
12.     print(INF_OK + adres_1)
13. else:
14.     print(INF_BLAD)
```

Warto wiedzieć

W przeciwieństwie do wielu innych języków programowania, w których można narzucić, aby zmiana wartości stałej nie była możliwa, w języku Python nie można tego wymusić. Oznaczenie nazwy zmiennej wielkimi literami nie blokuje możliwości zmiany jej wartości, a jedynie jest sposobem na oznaczenie przez programistę zmiennej, która będzie traktowana tak, jakby była stałą.

Ponieważ oba komunikaty ze specyfikacji są dla użytkownika niezmienne, możemy je traktować jako **stałe**, czyli wartości, które nie powinny się zmienić w trakcie wykonywania programu. Dobrą praktyką jest zapisywanie nazw zmiennych, których wartości mają być cały czas takie same, wielkimi literami (tak jak w liniach 1. i 2.).

Stała

Logika matematyczna i operatory logiczne
→ s. 198–200

W liniach 4.–5. i 7.–8. znajduje się fragment kodu określający nazwy zmiennych, w których będą zapamiętane dane wpisane z klawiatury.

W **warunku logicznym** (linia 11.) sprawdza się, czy zachodzi równość między wartościami dwóch zmiennych napisowych (typu `str`). Jeśli tak, wypisywany jest komunikat o poprawnej rejestracji (`INF_OK`) uzupełniony o podany adres e-mail (linia 12.).

Operator +

Operacja złączenia (konkatenacja)

Do złączenia (scalenia) napisów w języku Python wykorzystujemy **operator** `+`. Operację tę nazywa się **złączeniem** lub **konkatenacją** (ang. *concatenation*). Jeśli podane napisy są różne, program wypisze komunikat zapisany w zmiennej `INF_BLAD` (linia 14.).

Rysunek 3.2 przedstawia przykładowe wywołania programu.

Warto wiedzieć

W teorii literatury konkatenacją nazywa się konstrukcję, która polega na powieleniu tego samego słowa lub wyrażenia (niekoniecznie w tej samej formie fleksyjnej) na końcu jednego wersu i na początku następnego. Przykład zastosowania konkatenacji można znaleźć w wierszu Juliana Tuwima *Rzepka*.

```
Wykonywanie: porownanie adresow.py
Podaj imię: Robik
Podaj nazwisko: Spacjusz
Podaj adres e-mail: robik@spacjusz.pl
Powtórz adres e-mail: robik@spacjusz.pl

Hasło do serwisu wysłaliśmy na adres robik@spacjusz.pl
Python
```

```
Wykonywanie: porownanie adresow.py
Podaj imię: Robik
Podaj nazwisko: Spacjusz
Podaj adres e-mail: robik@spacjusz.pl
Powtórz adres e-mail: robik@wpacjusz.pl

Podane adresy e-mail są różne!
Python
```

Rys. 3.2. Wywołania programu *Porównanie adresów*

Ćwiczenie 5

Zapisz kod źródłowy programu *Porównanie adresów* i sprawdź działanie programu dla takich samych oraz różnych adresów e-mail.

A to ciekawe

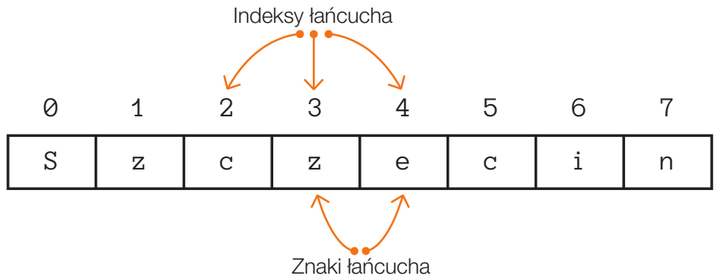
Czeski błąd

Jednym z rodzajów literówek jest przestawienie znaków w tekście, nazywane czeskim błędem. Zwykle zdarza się to osobom, które piszą na klawiaturze obiema rękami. Często klawisze z literami E oraz I wciska się w niewłaściwej kolejności, ponieważ znajdują się po przeciwnych stronach klawiatury. Powstaje wtedy np. „nei" zamiast „nie". Nie wiadomo, skąd wzięła się nazwa „czeski błąd". Według niektórych po prostu słowa w języku czeskim wyglądają dla Polaków tak, jakby ktoś poprzestawiał w nich litery. Inni twierdzą, że ma to związek z dwoma sposobami zapisu czeskich liczebników, np. 27 to *dvacet sedm* albo *sedmadvacet*. Podczas zapisywania takich liczb cyframi łatwo się pomylić i odwrócić kolejność.

Dostęp do znaków łańcucha

Aby uzyskać dostęp do znaków ciągu tekstowego, używa się indeksów. Na łańcuch znaków typu `str` można patrzeć jak na ciąg komórek tworzących strukturę podobną do listy. Pierwszy znak jest przechowywany w komórce o indeksie 0, drugi – w komórce o indeksie 1, a ostatni – w komórce o indeksie $n - 1$, gdzie n to liczba znaków łańcucha (rys. 3.3). Taki sposób indeksacji stosuje się również w **listach**.

Listy w języku Python
→ s. 202



Rys. 3.3. Interpretacja graficzna łańcucha znaków

Warto wiedzieć

W przypadku łańcuchów znaków nie można używać wyrażeń w postaci `imie[5] = "y"`. Próba zmiany znaku o indeksie 5 zakończy się komunikatem o błędzie. Tego typu zmiana możliwa jest m.in. na listach.

Aby wskazać konkretny znak łańcucha, należy zapisać nazwę zmiennej wraz z indeksem znaku w napisie ujętym w nawiasy kwadratowe, np. pierwszy znak zmiennej `imie` to `imie[0]`.

Język Python pozwala też na użycie tzw. **indeksów ujemnych**. Dla łańcucha `"Dorota"` zapisanego w zmiennej `imie` ostatni znak (`"a"`) wyznacza instrukcja `imie[-1]`, przedostatni (`"t"`) – `imie[-2]` itd.

Indeks ujemny

Próba odwołania się do znaku na pozycji, której w łańcuchu nie ma, zakończy się wyświetleniem błędu. Dlatego warto stosować **funkcję** `len`, która zwraca długość łańcucha tekstu. Na przykład dla zmiennej `miasto = "Szczecin"` wartość `len(miasto)` wynosi 8.

Funkcja `len`

Poprawność adresu e-mail – sprawdzenie struktury adresu

Program *Porównanie adresów* można rozbudować o sprawdzanie, czy napis podany przez użytkownika w polu „Podaj adres e-mail:" spełnia warunki poprawnego adresu poczty elektronicznej. Napiszemy program *Sprawdzenie adresu*, który będzie wykonywał to zadanie.

Dla uproszczenia przyjmiemy, że za poprawny adres e-mail będzie uznany napis, w którym znajdują się: dokładnie jeden znak @, który jest co najmniej trzecim znakiem napisu, oraz kropka, która jest czwartym lub trzecim znakiem od końca napisu i nie występuje bezpośrednio po znaku @.

Przykładami poprawnych adresów będą: nowak@abcd.pl oraz nk@ab.com. Niepoprawne adresy to np.: n@abc.pl, nk@w@.pl, n.@pl, a także nk@.com. W poprzednim programie porównywaliśmy całe napisy – tym razem będziemy porównywać pojedyncze znaki.

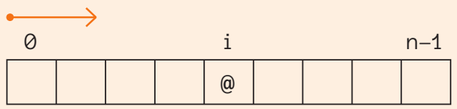
Poprawność adresu e-mail możemy sprawdzić w czterech etapach. Opisuje je schemat z rysunku 3.4 na s. 36.

Warto wiedzieć

Domeny najwyższego rzędu zazwyczaj są dwu- lub trzyliterowe. Należą do nich m.in. domeny: .pl, .us, .org, .com, .gov.

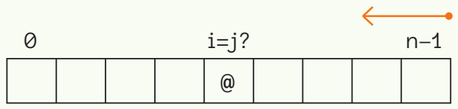
Etap 1

Szukamy w napisie (zaczynając od lewej strony) znaku @. Jeśli go nie znajdziemy albo będzie on pierwszym lub drugim znakiem napisu, sprawdzanie kończy się niepowodzeniem. W przeciwnym razie zapamiętujemy w zmiennej `i` pozycję znaku @ w napisie. Przechodzimy do etapu 2.



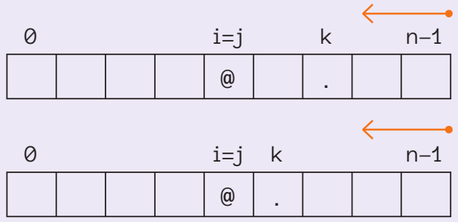
Etap 2

Sprawdzamy, czy wewnątrz napisu nie ma innego znaku @. Szukamy pierwszego wystąpienia znaku @, począwszy od prawego końca napisu. Zapamiętujemy jego pozycję w zmiennej `j` i sprawdzamy, czy `i` jest równe `j`. Jeśli wartości są różne, sprawdzanie kończy się niepowodzeniem. W przeciwnym razie przechodzimy do etapu 3.



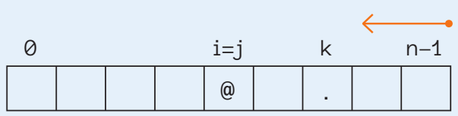
Etap 3

Sprawdzamy, czy znak kropki występuje w napisie na pozycji trzeciej lub czwartej od końca. Jeśli nie, sprawdzanie kończy się niepowodzeniem. Jeśli tak, zapamiętujemy pozycję kropki, np. w zmiennej `k`, i przechodzimy do etapu 4.



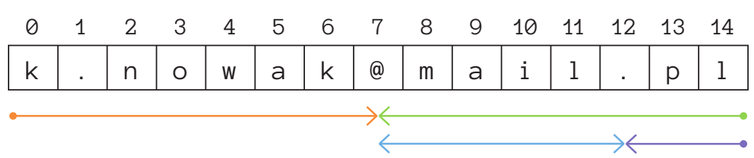
Etap 4

Sprawdzamy, czy znak @ znajduje się po lewej stronie znaku kropki w odstępie co najmniej jednego znaku. Jeśli nie, sprawdzanie kończy się niepowodzeniem. Jeśli tak, podany adres uznajemy za poprawny.



Rys. 3.4. Etapy sprawdzania poprawności podanego adresu e-mail

Przykład realizacji algorytmu pokazuje rysunek 3.5. Kolory strzałek są zgodne z kolorami tła poszczególnych etapów z rysunku 3.4.



Rys. 3.5. Sprawdzanie poprawności adresu

Algorytm sprawdzający poprawność adresu e-mail wygodnie będzie zapisać w kodzie źródłowym w formie funkcji:

Kod źródłowy funkcji `CzyPoprawnyAdres` w programie *Sprawdzenie adresu*

```
1.  def CzyPoprawnyAdres(adres):
2.      dl = len(adres)
3.
4.      i = 0  # etap 1
5.      while i < dl-1 and adres[i] != "@":
6.          i = i + 1
7.      if i < 2 or adres[i] != "@":
8.          return False
9.
10.     j = dl - 1  # etap 2
11.     while adres[j] != "@":
12.         j = j - 1
13.     if i != j:
14.         return False
15.
16.     k = dl - 1  # etap 3
17.     while k > 0 and adres[k] != ".":
18.         k = k - 1
19.     if adres[k] != "." or not (k == dl-3 or k == dl-4):
20.         return False
21.
22.     if k - i <= 1:  # etap 4
23.         return False
24.
25.     return True
```

Dobra rada

Pamiętaj, że w języku Python wcięcia mają kluczowe znaczenie, dlatego stosuj je konsekwentnie w całym kodzie. W edytorze Mu do wstawienia wcięcia możesz używać klawisza **Tab**.

Funkcja `CzyPoprawnyAdres` zwraca wartość logiczną – `True`, gdy podany napis jest adresem e-mail, lub `False` – w przeciwnym przypadku. W linii 2. definiujemy zmienną `dl`, która przechowa długość podanego napisu. Wykorzystaliśmy do tego **funkcję** `len`. Kolejne etapy sprawdzania poprawności adresu zaznaczono w komentarzach.

Funkcja `len`
→ s. 35

W etapie 1. (linie 4.–8.) zmienna `i` określa pozycję pierwszego znalezionego podczas przeszukiwania od lewej znaku @, o ile taki znak w napisie w ogóle jest. Aby uzyskać dostęp do znaku w napisie, stosuje się zapis z parą nawiasów kwadratowych.

Etap 2. zapisany jest w liniach 10.–14. Jeżeli `i` różni się od `j` (warunek z linii 13.), to w napisie występują co najmniej dwa znaki @.

W etapie 3. (linie 16.–20.) szukamy znaku kropki i określamy jego położenie. W linii 19. znajduje się zapis `not (k == dl – 3 or k == dl – 4)`:. Oznacza on **formułę logiczną** „Nieprawda, że zachodzi jeden z przypadków `k == dl – 3` lub `k == dl – 4`”.

Logika matematyczna i operatory logiczne
→ s. 198–200

Etap 4. (linie 22.–23.) to proste sprawdzenie, czy między kropką a znakiem @ znajdują się co najmniej dwa inne znaki.

Oto fragment kodu źródłowego z wywołaniem funkcji:

Fragment kodu źródłowego programu *Sprawdzanie adresu* z wywołaniem funkcji CzyPoprawnyAdres

```
1.  INF_OK = "Hasło do serwisu wysłaliśmy na adres "
2.  INF_ROZNE_ADRESY = "Podane adresy e-mail są różne!"
3.  INF_BLAD_ADRESU = "Adres jest niepoprawny!"
4.
5.  imie = input("Podaj imię: ")
6.  nazwisko = input("Podaj nazwisko: ")
7.  adres_1 = input("Podaj adres e-mail: ")
8.
9.  if CzyPoprawnyAdres(adres_1):
10.     adres_2 = input("Powtórz adres e-mail: ")
11.     print()
12.     if adres_1 == adres_2:
13.         print(INF_OK + adres_1)
14.     else:
15.         print(INF_ROZNE_ADRESY)
16. else:
17.     print(INF_BLAD_ADRESU)
```

Typy danych w języku Python
→ s. 201–202

Funkcja `CzyPoprawnyAdres` zwraca wartość **typu logicznego**, a jej wywołanie znajduje się w miejscu wyrażenia logicznego instrukcji warunkowej (linia 9.). W ten sposób funkcję można wykorzystać jako warunek logiczny, np. w instrukcji warunkowej lub pętli.

Ćwiczenie 6

Zapisz kod źródłowy programu *Sprawdzenie adresu*. Uruchom program i sprawdź jego działanie dla różnych danych.

A to ciekawe

Jak odróżnić człowieka od maszyny?

W wielu serwisach internetowych stosuje się rozwiązanie CAPTCHA, by upewnić się, że formularz wypełnia człowiek, a nie robot. Test polega zwykle na przepisaniu słowa z obrazka. Projekt reCAPTCHA wykorzystuje tę czynność do digitalizacji zasobów – użytkownikom wyświetla się fragmenty tekstu nieczytelne dla oprogramowania OCR (zamieniającego skan na tekst). Jeśli wystarczająco dużo osób tak samo odczyta dane słowo, można je uznać za poprawnie rozpoznane.

Poprawność adresu e-mail – sprawdzenie struktury adresu z wykorzystaniem metod klasy `str`

Język Python jest językiem obiektowym. Utworzenie w nim zmiennej typu `str`, np. `imie = "Dorota"`, oznacza utworzenie **obiektu** (ang. *object*) o nazwie `imie` należącego do **klasy** (ang. *class*) `str` (interpreter rozpozna klasę `str` po użyciu cudzysłowu). Klasa jest pewnym ogólnym, abstrakcyjnym opisem podobnych do siebie obiektów, np. w klasie o nazwie „kolory” obiektami będą „niebieski” oraz „biały”, ale nie będzie nim np. „mokry”.

Obiekt
Klasa

Klasa w języku Python, również klasa `str`, oferuje opracowane specjalnie dla niej funkcje, nazywane **metodami** (ang. *methods*). Metody są związane z obiektami danej klasy, np. dla klasy `str` istnieją **metody** zwracające pozycję znaku w tekście. Metody zapisuje się z kropką po nazwie zmiennej (obiektu), np. `imie.find("a")`.

Metoda

Obiekty, klasy i metody w języku Python
→ s. 208

Zapiszemy teraz opracowany wcześniej algorytm z wykorzystaniem metod `find` oraz `rfind`. **Metoda** `find` zwraca indeks pierwszego wystąpienia poszukiwanego ciągu znaków, a **metoda** `rfind` – ostatniego, czyli szuka frazy, zaczynając od końca (od prawej strony).

Metoda `find`
Metoda `rfind`

Oto zapis funkcji `CzyPoprawnyAdres` z użyciem powyższych metod:

Kod źródłowy funkcji `CzyPoprawnyAdres` w programie *Sprawdzenie adresu 2*

```
1.  def CzyPoprawnyAdres(adres):
2.      dl = len(adres)
3.
4.      i = adres.find("@")    # etap 1
5.      if i < 2 or i == -1:
6.          return False
7.
8.      j = adres.rfind("@")    # etap 2
9.      if i != j:
10.         return False
11.
12.     k = adres.rfind(".")    # etap 3
13.     if k == -1:
14.         return False
15.     if not (k == dl-3 or k == dl-4):
16.         return False
17.
18.     if k - i <= 1:    # etap 4
19.         return False
20.
21.     return True
```

Metodę `find` zastosowaliśmy w linii 4. do odszukania indeksu pierwszego wystąpienia znaku @. Warunki `i == -1` z linii 5. oraz `k == -1` z linii 13. odnoszą się do sytuacji, w której funkcje `find` i `rfind` nie znajdą poszukiwanych znaków. Wówczas metody te zwracają wartość –1.

Warto wiedzieć

Umieszczanie nazwy metody po kropce to cecha programowania obiektowego.

3.3. Lista zakupów. Usuwanie duplikatów w tekście

Warto wiedzieć

Jeśli dane wejściowe są uporządkowane, znacznie prościej jest rozwiązać wiele zadań. Na przykład porządek alfabetyczny w indeksie książki lub w słowniku pozwala szybko odnaleźć hasło.

Stworzymy teraz program, którego zadaniem będzie usunąć zbędne powtórzenia produktów z listy zakupów. Dla ułatwienia przyjmijmy, że napisy będą podawane w porządku alfabetycznym w oddzielnych wierszach, bez żadnych znaków interpunkcyjnych.

Specyfikacja

Dane: lista słów wpisywanych alfabetycznie, jedno pod drugim, zakończona trzema gwiazdkami (***).

Wynik: lista słów po usunięciu powtórzeń występujących w sąsiednich wierszach.

Sposób rozwiązywania tego problemu można sformułować następująco: dopóki na wejściu pojawiają się nowe wyrazy, należy porównywać ostatnio wczytany wyraz z poprzednim i w razie potrzeby usuwać powtórzenia, a nieusunięte wyrazy dołączać do listy wynikowej.

Precyzyjniej ten algorytm można zapisać jako listę kroków:

1. Wczytaj pierwszy wyraz. Zapamiętaj go jako pierwszy na liście wynikowej i wczytaj kolejny wyraz.
2. Dopóki ostatnio wczytany wyraz nie jest ciągiem znaków ***, wykonuj kolejno kroki 3 i 4.
   3. Jeśli ten wyraz jest różny od poprzedniego, to dodaj go jako kolejny na listę wynikową.
   4. Wczytaj kolejny wyraz.

Przeanalizujemy teraz działanie algorytmu dla listy słów. Wygodnie będzie to zrobić w tabeli:

Napis	Czy dublet?	Czy koniec listy?	Lista wynikowa
chleb	–	nie	chleb
cytryna	nie	nie	chleb cytryna
mleko	nie	nie	chleb cytryna mleko
ser	nie	nie	chleb cytryna mleko ser
ser	tak	nie	chleb cytryna mleko ser
szynka	nie	nie	chleb cytryna mleko ser szynka
***	nie	tak	chleb cytryna mleko ser szynka

Tabela 3.3. Analiza działania algorytmu usuwającego powtórzenia

Warto wiedzieć

Wyszukiwanie duplikatów jest operacją szczególnie istotną w pracy analityków danych. Niektóre aplikacje wspomagające ich pracę, np. arkusze kalkulacyjne, zawierają wbudowane funkcje pozwalające usuwać zduplikowane wartości.

Tabelę 3.3 uzupełniamy od lewej do prawej, według kolejnych kroków algorytmu. Kolorem zaznaczono etap, w którym nowy wyraz był powtórzeniem.

Ćwiczenie 7

Przeanalizuj działanie algorytmu dla listy złożonej z ośmiu słów, wśród których są dwa powtórzenia (jedno z nich jest na końcu).

Oto fragment kodu źródłowego programu *Usuwanie powtórzeń*, który uwzględnia ograniczenie listy słów do 20.

Fragment kodu źródłowego programu *Usuwanie powtórzeń*

```
1.  N = 20
2.
3.  wynik = []
4.  i = 1
5.  nowy = input()
6.  wynik.append(nowy)
7.
8.  stary = nowy
9.  nowy = input()
10.
11. while nowy != "***" and i < N:
12.     if nowy != stary:
13.         i = i + 1
14.         wynik.append(nowy)
15.     stary = nowy
16.     nowy = input()
17.
18. for wyraz in wynik:
19.     print(wyraz)
```

Warto wiedzieć

Wszelkiego rodzaju dane zapisane bezpośrednio w kodzie programu nazywa się literałami. Liczba 23 w instrukcji `suma = 23` to literał całkowity, a napis ABC w instrukcji `napis = "ABC"` to literał napisowy.

W linii 1. zmiennej `N` przypisano wartość 20. W linii 3. tworzymy pustą listę o nazwie `wynik` – przechowamy na niej listę słów. Zmienna `i` zdefiniowana w linii 4. będzie licznikiem. W liniach 5.–6. wczytujemy pierwszy napis i umieszczamy go na liście `wynik`. Wykorzystujemy do tego celu **metodę** `append`, która dopisuje na końcu listy nową wartość.

Metoda `append`

W liniach 8.–9. wczytujemy drugi napis. W pętli `while` (linie 11.–16.) sprawdzamy, czy podany ostatnio napis jest różny od *** oraz czy licznik napisów nie przekracza wartości `N`. Jeśli oba warunki są spełnione, kod w liniach 12.–14. zapobiega dodawaniu powtórzonej nazwy do listy. W liniach 18.–19. wypisywana jest lista produktów.

Dobra rada

Łańcuch *** w programie *Usuwanie powtórzeń* pełni funkcję wartownika. Używaj wartowników wtedy, kiedy chcesz oznaczyć warunki brzegowe realizacji programu, np. nie chcesz wyjść poza listę.

Ćwiczenie 8

Zapisz kod źródłowy programu *Usuwanie powtórzeń* i sprawdź działanie programu dla różnych danych wejściowych.

3.4. Wyszukiwanie wzorca w tekście

Wyszukiwanie wzorca w tekście

Wyszukiwanie wzorca w tekście, nazywane też dopasowywaniem wzorca (ang. *pattern matching*), to szukanie wystąpienia danego łańcucha znaków w tekście. Celem jest określenie, w którym miejscu w tekście znajduje się wzorzec, o ile w ogóle w nim występuje.

Wykreślanka

Warto wiedzieć

Wzorca szukamy np. wtedy, gdy chcemy znaleźć konkretne pojęcie w spisie treści lub indeksie podręcznika. Wówczas nasze oko stara się dostrzec dany ciąg znaków.

Na rysunku 3.6 znajduje się przykład zadania zwanego wykreślanką. Polega ono na szukaniu ukrytych wyrazów. Mogą one być zapisane poziomo, pionowo lub na ukos, także wspak. Wyszukiwanie wyrazów pośród porozrzucanych na planszy liter to przykład szukania wzorca.

Rys. 3.6. Wykreślanka z zaznaczonymi imionami

Ćwiczenie 9

W pliku, który otrzymasz od nauczyciela (np. *wykreslanka.xlsx*), znajduje się wykreślanka. Znajdź jak najwięcej imion, które w niej ukryto.

Program *Szukaj wzorca*

Napiszemy teraz program, który będzie poszukiwał podanego wzorca w tekście. Oto specyfikacja problemu:

Specyfikacja

Dane: tekst zapisany wielkimi literami oraz wzorzec – łańcuch zapisany wielkimi literami.

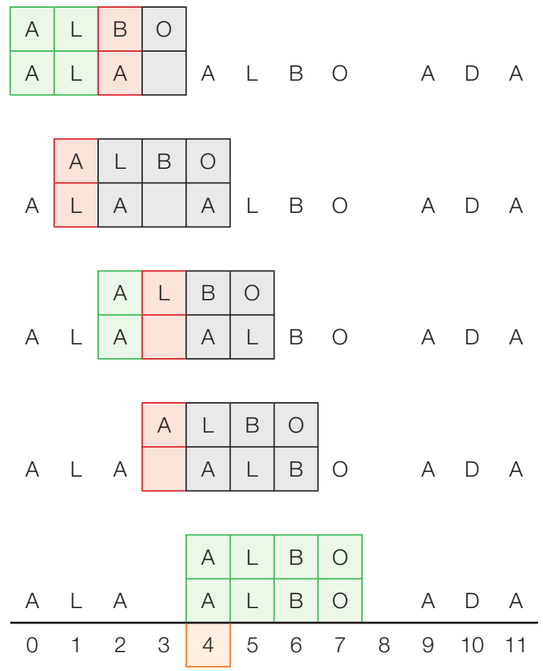
Wynik: pozycja pierwszego wystąpienia wzorca w tekście, rozumiana jako liczba znaków tekstu poprzedzających wystąpienie wzorca, lub liczba –1, jeśli wzorca nie ma w tekście.

Algorytm wyszukiwania wzorca w tekście

Przedstawimy działanie prostego **algorytmu wyszukiwania wzorca w tekście**, tzw. algorytmu naiwnego. Wzorzec będzie przesuwany stopniowo wzdłuż tekstu o jeden znak. Za każdym razem wzorzec będzie ustalał początek fragmentu tekstu, który będzie z nim porównywany znak po znaku (od pierwszego znaku wzorca).

Gdy analizowany znak jest zgodny ze znakiem wzorca, porównuje się zgodność kolejnego znaku fragmentu tekstu z kolejnym znakiem wzorca. Jeśli analizowany fragment tekstu jest identyczny ze wzorcem, to wyszukiwanie się kończy i jako odpowiedź zwracana jest pozycja dopasowania, czyli liczba znaków tekstu poprzedzających wystąpienie wzorca.

W tekście ALA ALBO ADA poszukamy wzorca ALBO. Rysunek 3.7 pokazuje działanie algorytmu. Zwracana jest wartość 4.



Rys. 3.7. Naiwne wyszukiwanie wzorca w tekście

Warto wiedzieć

Od rozwiązania problemu wyszukiwania wzorca w tekście można też oczekiwać, że wskaże ono wszystkie wystąpienia wzorca, jeśli jest ich więcej.

Ćwiczenie 10

a. Zapisz przebieg działania algorytmu poszukiwania w tekście ZA SZCZYTEM ZASZCZYT wzorca ZASZCZYT. Zapisz go w podobny sposób jak na rysunku 3.7.

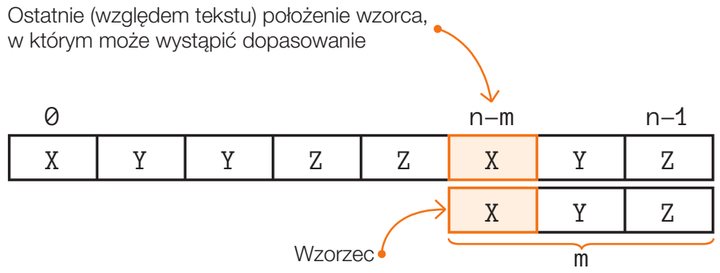
b. Ile prób dopasowania wzorca trzeba by było wykonać dla tekstu z podpunktu a), gdyby wyszukiwanie zacząć od końca?

Liczbę znaków tekstu do przeszukiwania oznaczmy literą n, a liczbę znaków wzorca – literą m. Aby stwierdzić, czy wzorzec występuje w tekście, dla każdego znaku tekstu, aż do znaku na pozycji o indeksie $n - m$, należy sprawdzać zgodność kolejnych znaków tekstu ze znakami tworzącymi wzorzec.

Dlaczego sprawdzamy do znaku o indeksie $n - m$? Wyjaśnia to rysunek 3.8 na s. 44. Jeśli tekst ma długość $n = 9$, a wzorzec $m = 3$, to ostatni indeks ma wartość 6 (znaki indeksujemy od 0).

Warto wiedzieć

W teorii algorytmów metodę rozwiązywania problemu, która wymaga sprawdzenia wszystkich przypadków, nazywa się metodą wyczerpującego wyszukiwania (ang. *exhaustive search*) albo metodą siłową (ang. *brute-force method*). O algorytmie mówi się wówczas, że jest naiwny.



Rys. 3.8. Wyznaczenie indeksu ostatniego znaku tekstu, z którym porównujemy wzorzec

Oto lista kroków naiwnego algorytmu wyszukiwania wzorca:

1. Wczytaj tekst i wzorzec.
2. Oznacz długość tekstu literą n, a długość wzorca – literą m.
3. Dla każdej liczby z zakresu od 0 do n - m (nazywanej bieżącą pozycją znaku) wykonuj kolejno kroki 4 i 5.
   4. Wybierz fragment tekstu długości m, począwszy od bieżącej pozycji znaku, i porównaj go ze wzorcem.
   5. Jeśli fragment i wzorzec są identyczne, to zwróć jako odpowiedź bieżącą pozycję znaku i zakończ algorytm.
6. Zwróć wartość -1.

W powyższym algorytmie licznik pętli, w momencie znalezienia wzorca w tekście, wskazuje jednocześnie indeks (pozycję) pierwszego wystąpienia wzorca w tekście. Jest to możliwe, ponieważ wartości licznika zaczynają się od 0, czyli tej wartości, od której indeksuje się znaki w łańcuchach. Oto kod źródłowy programu *Szukaj wzorca*:

Kod źródłowy programu *Szukaj wzorca*

```
1.  TEKST = "ALA ALBO ADA"
2.
3.  def Porownaj(wzorzec):
4.      n = len(TEKST)
5.      m = len(wzorzec)
6.      for poz in range(0, n-m+1):
7.          j = 0
8.          while j < m and TEKST[poz+j] == wzorzec[j]:
9.              j = j + 1
10.         if j == m:
11.             return poz
12.     return -1
13.
14. print("Tekst:", TEKST)
15. print("Podaj wzorzec:", end=" ")
16. wzorzec = input()
17.
18. print("Pozycja:", Porownaj(wzorzec))
```

Dobra rada

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

W linii 1. zastosowano zmienną o nazwie TEKST. Zmienne n i m (linie 4. i 5.) pamiętają długość odpowiednio tekstu i wzorca.

Wewnątrz funkcji Porownaj zastosowano **pętle zagnieżdżone** (ang. *nested loops*), czyli pętlę umieszczoną wewnątrz innej pętli. Zewnętrzna pętla (for), w liniach 6.–11., służy do przesuwania wzorca po kolejnych znakach tekstu, a wewnętrzna (while), w liniach 8.–9., przechodzi przez kolejne znaki wzorca (krok 4 z listy kroków).

Pętle zagnieżdżone

Pętla while oraz instrukcja warunkowa z linii 10.–11. sprawdzają, czy w napisie występuje wzorzec – wartość zmiennej j będzie równa długości wzorca. Wówczas funkcja kończy działanie i zwraca wartość zmiennej poz (linia 11.).

Rysunek 3.9 przedstawia przykładowe wywołania programu.

```
Wykonywanie: szukai wzorca.ov
Tekst: ALA ALBO ADA
Podaj wzorzec: AL
Pozycja: 0
```

```
Wykonywanie: szukai wzorca.ov
Tekst: ALA ALBO ADA
Podaj wzorzec: ALBO
Pozycja: 4
```

Rys. 3.9. Wywołania programu *Szukaj wzorca* dla dwóch różnych wzorców

Dobra rada

Pamiętaj, że wielkie i małe litery mają różne kody ASCII. Jeśli tekst wyjściowy jest zapisany wielkimi literami, wówczas wzorzec należy zapisać również wielkimi literami.

Ćwiczenie 11

a. W pliku otrzymanym od nauczyciela (np. *wzorzec_w_tekscie.py*) znajduje się fragment programu *Szukaj wzorca* z tekstem, w którym należy wyszukać podany wzorzec. Uzupełnij go o odpowiednie instrukcje, a następnie uruchom program dla wzorców: AGA oraz ALBO.

b. Sprawdź działanie programu dla własnego tekstu i wzorca.

Naiwnego algorytmu wyszukiwania wzorca nie można by było z powodzeniem zaimplementować np. w czytniku PDF. Okazuje się, że dla dłuższych tekstów jest on bardzo powolny. Znane są jednak bardzo efektywne algorytmy dopasowywania wzorca w tekście i to z nich korzysta się w praktyce przy przetwarzaniu tekstów.

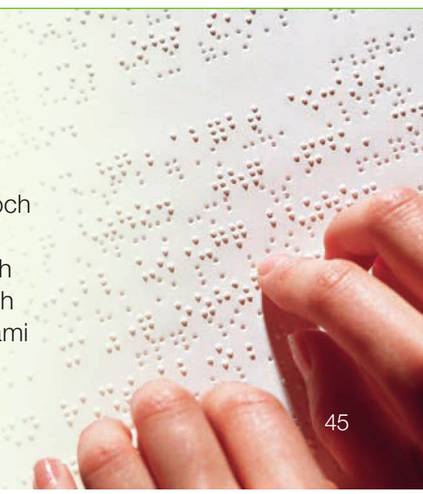
Warto wiedzieć

Efektywna metoda wyszukania wzorca w tekście jest zaimplementowana w języku Python jako metoda find. Używaliśmy jej w programie sprawdzającym poprawność adresu e-mail.

A to ciekawe

Alfabet Braille'a, czyli szukanie wzorca

Alfabet Braille'a to system zapisu informacji z wykorzystaniem dwóch stanów: punkt wypukły i niewypukły. Znaki przedstawione są jako sześciopunkty, czyli kombinacje punktów umieszczonych w dwóch kolumnach, po trzy punkty w każdej. Czytanie książek brajlowskich oznacza w praktyce wielokrotne dopasowywanie wzorca opuszkami palców w celu odczytania kolejnych znaków tekstu.



Zadania

* **1** Napisz program, który wczyta z klawiatury nazwę miejscowości (bez spacji) i wypisze co drugą jej literę. Litery powinny być rozdzielone spacjami.

* **2** Napisz program, który wczyta z klawiatury 10 nazw kolorów i wypisze najdłuższą z nich. Jeśli takich nazw jest więcej, to program powinien wyświetlić nazwę podaną najpóźniej.

* **3** Napisz program, który wczyta z klawiatury łańcuch znaków (składający się z kombinacji cyfr oraz małych i wielkich liter) i wypisze tylko wielkie litery, które pojawiły się w tekście.

* **4** Napisz program, który dla podanego wyrazu wyświetli jego zapis wspak (od ostatniej do pierwszej litery).

* **5** W pliku, który otrzymasz od nauczyciela (np. *emoji.pdf*), znajdują się emoji myszy (🐭) oraz biedronki (🐞). Zmodyfikuj program *Znaki 2* tak, aby wyświetlił znaki zawarte między tymi symbolami.

** **6** Napisz program, który będzie zliczał liczbę samogłosek wewnątrz łańcucha znaków.

** **7** Napisz program, który wyświetli litery występujące jednocześnie w dwóch słowach wpisanych z klawiatury. Na przykład dla słów ARKA i BARAK program powinien wypisać litery A, K i R.

** **8** Wyrazy różniące się dowolną jedną literą nazywa się metagramami, np.: kasa i kara. Napisz program sprawdzający, czy podana para wyrazów to metagramy.

** **9** Napisz program, który z łańcucha znaków będzie usuwał wszystkie samogłoski i zastępował je znakiem podkreślenia (_).

** **10** Napisz funkcję o nagłówku `ile(wzorzec)`, która zwróci liczbę całkowitą oznaczającą liczbę wystąpień wzorca w tekście. Przyjmij, że tekst jest zapisany w zmiennej o nazwie `TEKST`.

*** **11** Napisz program, który po podaniu daty urodzenia (dzień i miesiąc) wypisze odpowiadający jej znak zodiaku. Na przykład dla danych `21.07` wynikiem będzie napis `rak`.

*** **12** Napisz funkcję, która kompresuje tekst prostą metodą opartą na zliczaniu powtarzających się liter (jedna litera może się powtarzać maksymalnie 9 razy). Na przykład wynikiem dla łańcucha `aaabbcaa` będzie `a3b2c1a2`. Jeśli wynik miałby być łańcuchem dłuższym niż ten dany na wejściu, to funkcja powinna zwracać pierwotny łańcuch znaków.

*** **13** Napisz program, który poda liczbę wystąpień każdej litery alfabetu łacińskiego w tekście wprowadzonym do zmiennej o nazwie `TEKST`. Program nie powinien rozróżniać wielkości liter.

4 Szyfrowanie wiadomości

Szyfrów używano od wielu stuleci do utajniania korespondencji dworskiej oraz w wojsku do przesyłania pomiędzy oddziałami wiadomości, które nie powinny być czytane przez wroga. W XIX w., kiedy zaczęła się rozwijać kolej żelazna i upowszechniały się sposoby komunikacji na odległość (telegraf i teleks), metodami szyfrowania zainteresowali się również ludzie biznesu. W II połowie XX w., wraz z rozwojem telekomunikacji i globalnych sieci komputerowych, praktyka szyfrowania informacji stawała się coraz bardziej powszechna. Dziś dotyczy każdego z nas, gdyż stanowi podstawę bezpieczeństwa m.in. transakcji bankowych i płatności elektronicznych.

Ważne w tym temacie

- Podstawowe pojęcia związane z kryptografią
- Zastosowania kryptografii
- Szyfr Cezara
- Programy szyfrujące informacje tekstowe szyfrem Cezara
- Metody łamania prostych szyfrów

4.1. Podstawowe pojęcia kryptografii

Szyfrowaniem nazywamy przekształcanie informacji do postaci trudnej do odczytania bez dodatkowej wiedzy. Celem szyfrowania jest zapewnienie tajności (poufności) informacji. Zagadnieniami związanymi z szyfrowaniem zajmuje się nauka zwana **kryptografią**.

Szyfrowanie

Kryptografia

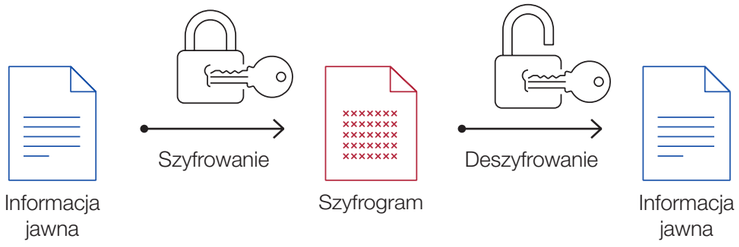
W trakcie szyfrowania informację jawną, zwaną **tekstem jawnym** (ang. *plaintext*), przekształca się w **szyfrogram** (ang. *ciphertext*), czyli tekst zaszyfrowany. W tym procesie stosuje się odpowiedni algorytm oraz **klucz szyfrowania** (ang. *cipher key*). Jest to pewna wartość, niezależna od szyfrowanej informacji. Klucz szyfrowania jest potrzebny do odczytania informacji jawnej. Przekształcanie zaszyfrowanych danych w informację pierwotną to **deszyfrowanie** (ang. *decryption*) (rys. 4.1). Warunkiem zachowania tajności informacji jest utrzymanie w tajemnicy klucza szyfrowania.

Tekst jawny

Szyfrogram

Klucz szyfrowania

Deszyfrowanie



Rys. 4.1. Schemat szyfrowania i deszyfrowania informacji

4.2. Szyfr Cezara

Szyfr podstawieniowy

Jeszcze w czasach starożytnych wymyślono **szyfr podstawieniowy**. Polega on na zamianie znaków tekstu jawnego na inne znaki alfabetu według określonego algorytmu. W szyfrogramie mogą więc pojawić się znaki alfabetu, które nie zostały użyte w tekście jawnym.

Szyfr Cezara

Szyfr Cezara

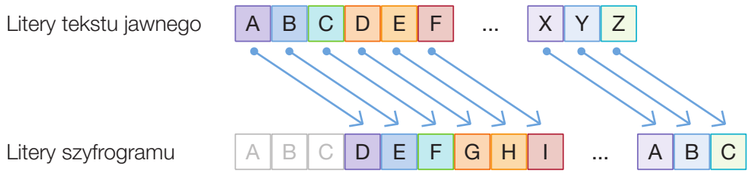
Jednym z najbardziej znanych klasycznych szyfrów podstawieniowych jest **szyfr Cezara**. Był on stosowany przez Juliusza Cezara – żołnierza, polityka i dyktatora Republiki Rzymskiej w I w. p.n.e.

Tekst jawny
→ s. 47

Metoda szyfrowania, której używał Cezar, polega na zastąpieniu każdej litery **tekstu jawnego** literą oddaloną od niej o stałą liczbę pozycji w alfabecie. Kierunek zmian musi być zachowany. Przesunięcie odbywa się cyklicznie, tzn. po ostatniej literze alfabetu następuje pierwsza.

Klucz szyfrowania
→ s. 47

Kluczem dla szyfru Cezara jest liczba miejsc, o jaką przesuwane są litery w alfabecie. Juliusz Cezar stosował klucz o wartości 3, tzn. literę A zastępował literą D, literę B – literą E itd. Pokazuje to rysunek 4.2.



Rys. 4.2. Szyfrowanie liter szyfrem Cezara z kluczem 3

Warto wiedzieć

Juliusz Cezar, jak każdy wykształcony człowiek w tamtych czasach, regularnie pisywał listy do przyjaciół oraz polityków. Aby utajnić poufną korespondencję, szyfrował jej treść.

Warto wiedzieć

Alfabet łaciński w czasach Juliusza Cezara składał się z 23 liter. Nie używano wówczas liter J, U i W.

Ćwiczenie 1

Wykorzystując szyfr Cezara i alfabet łaciński:

a. zaszyfruj wyraz MEDYCYNA z kluczem 25.

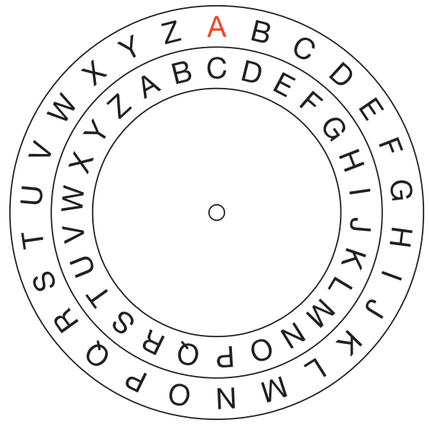
b. odszyfruj szyfrogram LWUQVQSIVI z kluczem 8.

A to ciekawe

Anagram jako szyfrogram

Niektórzy uczeni nie chcieli ogłaszać wprost treści swoich odkryć, dlatego stosowali anagramy, czyli zmieniali kolejność liter w tekście. Istotę odkrycia, zazwyczaj w formie łacińskiego anagramu, ogłaszali drukiem. W 1610 r. Galileusz opublikował w ten sposób swoje odkrycie dotyczące rzekomych księżyców Saturna: SMAISMRMILMEPOETALEVMIBVNENVGTTAVIRAS. Oznacza to: ALTISSIMUM PLANETAM TERGEMINUM OBSERVARI (dosł. widziałem najwyższą z planet potrójną). Liter U i V wtedy nie rozróżniano.

Przesunięcie liter w szyfrze Cezara można również przedstawić w postaci dwóch tarcz, z których wewnętrzna się obraca. Po obróceniu wewnętrznej tarczy otrzymujemy przyporządkowanie: litera tekstu jawnego i odpowiadająca jej litera tekstu zaszyfrowanego. Na rysunku 4.3 przedstawiono przesunięcie równe 2, tzn. literze A odpowiada litera C, literze B – litera D itd.



Rys. 4.3. Szyfrowanie liter szyfrem Cezara za pomocą obracających się tarcz

Warto wiedzieć

Wykorzystanie do szyfrowania obracających się tarcz z literami alfabetu zaproponował w XV w. Leon Battista Alberti. Wewnętrzna tarcza zawierała litery zapisane w losowej kolejności. Oprócz tego założył, że po zaszyfrowaniu kilku wyrazów tekstu jawnego tarcze należy obrócić, aby otrzymać zupełnie nowe podstawienie. Tak powstał tzw. szyfr polialfabetyczny. Podstawienie Cezara zapisane na dwóch tarczach jest przykładem szyfru monoalfabetycznego.

Ćwiczenie 2

a. Przygotuj fizyczny model obracających się tarcz pozwalających szyfrować tekst metodą Cezara, podobny jak na rysunku 4.3. Użyj pinezki, aby umocować przyrząd np. na tablicy korkowej. Wskazówka: Możesz wykonać projekt tarcz w programie graficznym, a następnie wydrukować je i wyciąć.

b. Korzystając z utworzonego modelu, zaszyfruj wiadomość ALARM z kluczem 4 oraz odszyfruj szyfrogram TEDBY zaszyfrowany z kluczem 10.

Szyfr Cezara – program szyfrujący pojedyncze słowo

Napiszemy program, który pozwoli zaszyfrować napis złożony wyłącznie z wielkich liter alfabetu łacińskiego.

Oto specyfikacja problemu:

Specyfikacja

Dane: tekst jawny składający się wyłącznie z wielkich liter alfabetu łacińskiego; klucz szyfrowania w postaci liczby całkowitej z zakresu od 1 do 25.

Wynik: tekst zaszyfrowany szyfrem Cezara z wykorzystaniem podanego klucza.

Warto wiedzieć

Informację zaszyfrowaną nazywa się czasami niepoprawnie informacją zakodowaną. Słowa „szyfr” i „kod” nie są synonimami. Kodu używamy, gdy nie można bezpośrednio przekazać informacji, np. ze względu na odległość. Przykładem kodu jest zapis za pomocą liter na papierze albo binarnie w komputerze. Szyfr to przekształcenie zapisu jawnego w niejawny, np. przez przestawienie kolejności liter lub bitów.

Lista kroków algorytmu szyfrującego metodą Cezara pojedyncze słowo może wyglądać następująco:

1. Wczytaj tekst jawny.
2. Wczytaj klucz szyfrowania i oznacz tę liczbę jako klucz.
3. Dla każdej litery tekstu jawnego wykonuj kolejno kroki 4 i 5.
   4. Oznacz kod ASCII tej litery jako kod.
   5. Jeżeli kod + klucz jest liczbą niewiększą od kodu ASCII litery Z, to dołącz do szyfrogramu literę o kodzie ASCII równym kod + klucz. W przeciwnym razie dołącz do szyfrogramu literę o kodzie ASCII wynoszącym kod + klucz – 26.

Kody ASCII kolejnych liter szyfrogramu zapamiętamy w zmiennej pomocniczej kod. Aby wyznaczyć kod ASCII litery szyfrogramu, do kodu litery tekstu jawnego, otrzymanego jako wartość **funkcji** `ord`, dodajemy przesunięcie, czyli wartość klucza. Jeśli otrzymana wartość będzie większa od kodu litery Z (liczby 90), musimy odjąć 26 (liczbę liter alfabetu), aby otrzymać kod litery z początku alfabetu.

Funkcja `ord` → s. 32

Na przykład podczas szyfrowania litery Y w słowie INFORMATYKA z kluczem 3 otrzymamy literę B. Kod ASCII litery Y to 89. Po dodaniu 3 uzyskujemy 92, a więc wartość większą od kodu litery Z. Odejmujemy 26 i otrzymujemy 66, czyli kod litery B.

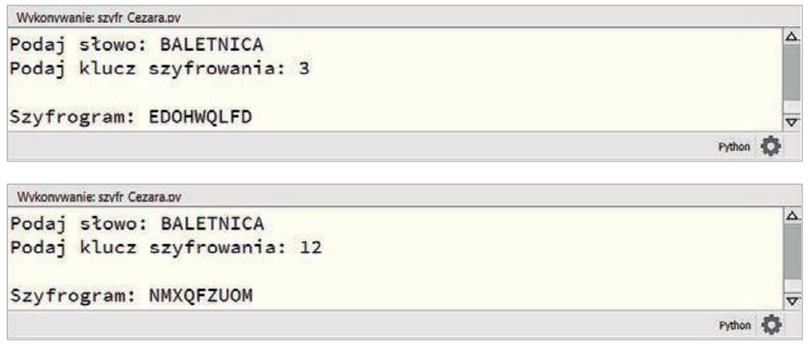
Oto kod źródłowy programu *Szyfr Cezara* realizującego podany powyżej algorytm:

Kod źródłowy programu *Szyfr Cezara*

```
1.  szyfrogram = ""
2.
3.  jawny = input("Podaj słowo: ")
4.  klucz = int(input("Podaj klucz szyfrowania: "))
5.
6.  for znak in jawny:
7.      kod = ord(znak) + klucz
8.      if kod > ord("Z"):
9.          kod = kod - 26
10.     szyfrogram = szyfrogram + chr(kod)
11.
12. print("\nSzyfrogram:", szyfrogram)
```

W linii 1. zmiennej `szyfrogram` przypisujemy jako początkową wartość łańcuch pusty, czyli łańcuch bez znaków (o długości 0). Do tego łańcucha będziemy dołączać kolejne znaki szyfrogramu. W pętli z linii 6.–10. obliczamy kod ASCII pojedynczego znaku szyfrogramu (jednej litery) i zapisujemy go w zmiennej pomocniczej `kod` (linia 7.). Jeśli wartość zmiennej `kod` jest większa od kodu litery Z (wartości funkcji `ord`), to odejmujemy od niej 26 (linia 9.). W linii 10. do wartości zmiennej `szyfrogram` dołączamy otrzymaną literę szyfrogramu.

Rysunek 4.4 przedstawia przykłady wywołań programu *Szyfr Cezara*.



Rys. 4.4. Wywołania programu *Szyfr Cezara* dla kluczy 3 oraz 12

Ćwiczenie 3

Zapisz kod źródłowy programu *Szyfr Cezara*. Uruchom program i sprawdź jego działanie dla różnych słów i kluczy szyfrowania.

Szyfr Cezara – program szyfrujący długi tekst

Teraz napiszemy program, który pozwoli zaszyfrować szyfrem Cezara dłuższy tekst składający się ze słów zapisanych alfabetem łacińskim oddzielonych spacjami. Oto specyfikacja problemu:

Specyfikacja

Dane: tekst jawny składający się z wielkich liter alfabetu łacińskiego oraz spacji; klucz szyfrowania w postaci liczby całkowitej z przedziału od 1 do 25.

Wynik: tekst zaszyfrowany szyfrem Cezara z podanym kluczem. Znaki spacji pozostają w szyfrogramie niezmienione.

A oto lista kroków algorytmu dla tej specyfikacji:

1. Wczytaj tekst jawny.
2. Wczytaj klucz szyfrowania i oznacz tę liczbę jako klucz.
3. Dla każdego znaku tekstu jawnego wykonuj kolejno kroki 4–6.
   4. Oznacz kod ASCII tego znaku jako kod.
   5. Jeśli kod jest kodem ASCII spacji, to dołącz do szyfrogramu spację. W przeciwnym razie wykonaj krok 6.
   6. Jeżeli kod + klucz jest liczbą niewiększą od kodu ASCII litery Z, to dołącz do szyfrogramu literę o kodzie ASCII równym kod + klucz. W przeciwnym razie dołącz do szyfrogramu literę o kodzie ASCII wynoszącym kod + klucz – 26.

Dobra rada

Pamiętaj, że funkcja `ord` zwraca liczbę całkowitą odpowiadającą pozycji znaku w tablicy Unicode, a funkcja `chr` zwraca jednoznakowy napis typu `str`.

Warto wiedzieć

Współczesne algorytmy kryptograficzne pozwalają nie tylko utajnić przesyłaną wiadomość, lecz także zrealizować inne cele. Zapewniają m.in.:
- spójność danych, dzięki czemu mamy pewność, że wiadomość nie była modyfikowana,
- uwierzytelnienie, czyli potwierdzenie tożsamości nadawcy i odbiorcy lub autentyczności przesyłanych danych.

Zapiszmy kod źródłowy programu *Szyfr Cezara 2*, w którym do szyfrowania metodą Cezara z wybranym kluczem wykorzystamy funkcję o nazwie `Cezar`.

Kod źródłowy programu *Szyfr Cezara 2*

```
1.  def Cezar(klucz, jawny):
2.      szyfrogram = ""
3.
4.      for znak in jawny:
5.          if znak == " ":
6.              kod = ord(" ")
7.          else:
8.              kod = ord(znak) + klucz
9.              if kod > ord("Z"):
10.                 kod = kod - 26
11.         szyfrogram = szyfrogram + chr(kod)
12.
13.     return szyfrogram
14.
15. jawny = input("Podaj tekst: ")
16. klucz = int(input("Podaj klucz szyfrowania: "))
17.
18. print("\nSzyfrogram:", Cezar(klucz, jawny))
```

Dobra rada

Jeśli pewne instrukcje są wykonywane kilka razy w tym samym programie, zapisz je w postaci funkcji.

Definicja funkcji `Cezar` znajduje się w liniach 1.–13., a zasadnicza część programu – w liniach 15.–18. Funkcja `Cezar` z parametrami `klucz` i `jawny` wykorzystuje poznaną wcześniej technikę budowania szyfrogramu z liter będących przesunięciami kolejnych liter tekstu jawnego o stałą liczbę (wartość klucza).

Program *Szyfr Cezara* → s. 50

Pętla `for` z linii 4.–11. działa analogicznie do pętli w **programie *Szyfr Cezara***, jednak dodatkowo podczas szyfrowania znaki spacji pozostają niezmienione. Odpowiadają za to wiersze 5.–6.

Na rysunku 4.5 przedstawiono wywołania programu *Szyfr Cezara 2* dla dwóch różnych kluczy.

```
Wykonywanie: szyfr Cezara 2.py
Podaj tekst: TO JEST MALY KROK DLA CZLOWIEKA ALE WIELKI SKOK DLA LUDZKOSCI
Podaj klucz szyfrowania: 7

Szyfrogram: AV QLZA THSF RYVR KSH JGSVDPLRH HSL DPLSRP ZRVR KSH SBKGRVZJP
Python
```

```
Wykonywanie: szyfr Cezara 2.py
Podaj tekst: TO JEST MALY KROK DLA CZLOWIEKA ALE WIELKI SKOK DLA LUDZKOSCI
Podaj klucz szyfrowania: 21

Szyfrogram: OJ EZNO HVGT FMJF YGV XUGJRDZFV VGZ RDZGFD NFJF YGV GPYUFJNXD
Python
```

Rys. 4.5. Wywołania programu *Szyfr Cezara 2* dla kluczy 7 i 21

Ćwiczenie 4

Zapisz kod źródłowy programu *Szyfr Cezara 2* i uruchom program. Sprawdź jego działanie dla napisu: PRZYJDZ DO SKLEPU I ZAPYTAJ O OLGE i klucza szyfrowania równego 13.

Deszyfrowanie wiadomości zaszyfrowanych metodą Cezara

Aby odszyfrować wiadomość zapisaną alfabetem łacińskim, zaszyfrowaną szyfrem Cezara ze znaną wartością klucza, wystarczy od kodu znaku napisu odjąć wartość klucza.

Aby odczytać wiadomość zaszyfrowaną szyfrem Cezara ze znanym kluczem, można też ponownie ją zaszyfrować z nowym kluczem o wartości 26 – *k*, gdzie *k* jest kluczem zastosowanym do zaszyfrowania pierwotnego tekstu.

Warto wiedzieć

Współcześnie stosowane szyfry używają jako kluczy liczb, które w zapisie binarnym mają po kilkaset, a nawet kilka tysięcy cyfr (bitów).

Ćwiczenie 5

Napisz program deszyfrujący tekst zaszyfrowany metodą Cezara ze znaną wartością klucza. Sprawdź działanie programu dla szyfrogramów:

a. PKYZKY SOYZXFKS QXEVZUMXGLOO oraz klucza 6,

b. RSGNMTFCKOBWS GNMTFI QSNOFO NS NBOBMA YZIQNSA XSGH DFCGHS oraz klucza 14.

Tekst powyższych szyfrogramów znajduje się w pliku, który przekaże ci nauczyciel (np. o nazwie *szyfrogramy_cezar.txt*).

Dobra rada

W edytorze Mu po uruchomieniu programu możesz wstawiać tekst ze schowka. W tym celu skorzystaj z menu kontekstowego, które pojawi się po naciśnięciu prawego przycisku myszy, i wybierz **Wklej** (**Paste**).

Zapamiętaj

Szyfr Cezara to szyfr podstawieniowy, który polega na zastąpieniu każdej litery tekstu jawnego literą oddaloną od niej o stałą liczbę pozycji w alfabecie. Przesunięcie odbywa się cyklicznie, tzn. po ostatniej literze alfabetu następuje pierwsza.

4.3. Metody łamania klasycznych szyfrów

Kryptografia (od gr. *kryptós* – ukryty i *gráphō* – piszę) jako nauka o utajnianiu wiadomości jest częścią szerszej dziedziny nazywanej **kryptologią** (od gr. *kryptós* – ukryty i *lógos* – słowo).

Kryptologia

Kryptologia obejmuje między innymi wiedzę dotyczącą szyfrowania, deszyfrowania oraz łamania szyfrów w celu bezpiecznego przekazywania informacji. Nauka, która zajmuje się łamaniem szyfrów, to **kryptoanaliza** (od gr. *kryptós* – ukryty i *analýein* – rozluźnić, rozwiązać).

Kryptoanaliza

Zastosujemy kilka prostych technik kryptoanalizy do złamania poznanego szyfru. Choć klasyczne szyfry dziś już nie chronią skutecznie przesyłanych informacji, warto je analizować również pod kątem technik ich łamania. Oto kilka z nich.

Atak siłowy

Aby odczytać tekst jawny składający się z liter alfabetu łacińskiego zaszyfrowany metodą Cezara, wystarczy wygenerować 25 możliwych tekstów i wybrać właściwy. Dla współczesnych komputerów to bardzo niewiele. Technika ta jest nazywana **atakiem siłowym** lub **brute-force** (dosł. brutalnej siły). Przykład działania programu łamiącego szyfrogram pokazuje rysunek 4.6.

Atak siłowy (brute-force)

Warto wiedzieć

W języku łacińskim przymiotnik *brutus* pierwotnie oznaczał „ciężki”. Później zaczęto go też używać w znaczeniu „głupi”.

```
Wykonywanie: Cezar silowy.py
Podaj szyfrogram: FA VQEF YMXK WDAW PXM OLXAIUQWM MXQ IUQXWU EWAW PXM XGPLWAEOU

GB WRFG ZNYL XEBX QYN PMYBJVRXN NYR JVRYXV FXBX QYN YHQMXBFPV Klucz:  25
HC XSGH AOZM YFCY RZO QNZCKWSYO OZS KWSZYW GYCY RZO ZIRNYCGQW Klucz:  24
ID YTHI BPAN ZGDZ SAP ROADLXTZP PAT LXTAZX HZDZ SAP AJSOZDHRX Klucz:  23
JE ZUIJ CQBO AHEA TBQ SPBEMYUAQ QBU MYUBAY IAEA TBQ BKTPAEISY Klucz:  22
KF AVJK DRCP BIFB UCR TQCFNZVBR RCV NZVCBZ JBFB UCR CLUQBFJTZ Klucz:  21
LG BWKL ESDQ CJGC VDS URDGOAWCS SDW OAWDCA KCGC VDS DMVRCGKUA Klucz:  20
MH CXLM FTER DKHD WET VSEHPBXDT TEX PBXEDB LDHD WET ENWSDHLVB Klucz:  19
NI DYMN GUFS ELIE XFU WTFIQCYEU UFY QCYFEC MEIE XFU FOXTEIMWC Klucz:  18
OJ EZNO HVGT FMJF YGV XUGJRDZFV VGZ RDZGFD NFJF YGV GPYUFJNXD Klucz:  17
PK FAOP IWHU GNKG ZHW YVHKSEAGW WHA SEAHGE OGKG ZHW HQZVGKOYE Klucz:  16
QL GBPQ JXIV HOLH AIX ZWILTFBHX XIB TFBIHF PHLH AIX IRAWHLPZF Klucz:  15
RM HCQR KYJW IPMI BJY AXJMUGCIY YJC UGCJIG QIMI BJY JSBXIMQAG Klucz:  14
SN IDRS LZKX JQNJ CKZ BYKNVHDJZ ZKD VHDKJH RJNJ CKZ KTCYJNRBH Klucz:  13
TO JEST MALY KROK DLA CZLOWIEKA ALE WIELKI SKOK DLA LUDZKOSCI Klucz:  12
UP KFTU NBMZ LSPL EMB DAMPXJFLB BMF XJFMLJ TLPL EMB MVEALPTDJ Klucz:  11
VQ LGUV OCNA MTQM FNC EBNQYKGMC CNG YKGNMK UMQM FNC NWFBMQUEK Klucz:  10
WR MHVW PDOB NURN GOD FCORZLHND DOH ZLHONL VNRN GOD OXGCNRVFL Klucz:  9
XS NIWX QEPC OVSO HPE GDPSAMIOE EPI AMIPOM WOSO HPE PYHDOSWGM Klucz:  8
YT OJXY RFQD PWTP IQF HEQTBNJPF FQJ BNJQPN XPTP IQF QZIEPTXHN Klucz:  7
                                                                  Python
```

Rys. 4.6. Zastosowanie ataku siłowego do łamania szyfru Cezara

A to ciekawe

(Kryptologiczny) Cud nad Wisłą

	0	1	2	3	4	5	6	7	8	9	
0		Д	А	З	М	Щ	Ш	Р		К	0
1	Ч	В	П	Ф		С	Ч		Й		1
2			Ю		Ж			О	Г		2
3											3
4											4
5											5
6											6
7											7
8											8
9											9
	0	1	2	3	4	5	6	7	8	9	

Choć o zwycięstwie Polaków nad Armią Czerwoną w 1920 r. często mówi się w kategoriach cudu, udział w nim miały m.in. osiągnięcia kryptologiczne. Dzięki pracom polskiego Biura Szyfrów w latach 1919–1920 udało się odszyfrowywać kilka tysięcy szyfrogramów przesyłanych przez Rosjan drogą radiową. Zespołem kierował porucznik Jan Kowalewski, który do współpracy zaprosił wybitnych matematyków: Wacława Sierpińskiego, Stefana Mazurkiewicza i Stanisława Leśniewskiego. Opracowali oni nowatorską metodę lingwistyczno-matematycznego ataku na szyfr. Była ona potem rozwijana przez kolejne pokolenia kryptologów, w tym polskich matematyków, którzy złamali szyfr Enigmy.

Ćwiczenie 6

Napisz program łamiący szyfr Cezara techniką ataku siłowego.

Ataki ze znanym tekstem jawnym

Jeśli atakujący zna część tekstu zaszyfrowanego i odpowiadającą jej część tekstu jawnego, może to wykorzystać do odszyfrowania całego tekstu. Na przykład jeśli przechwytujący szyfrogram wie, że autor wszystkie swoje wiadomości zaczyna od słów „dzień dobry”, to może dzięki tej informacji łatwiej znaleźć zastosowany klucz szyfrowania.

Atak ze znanym tekstem jawnym od razu łamie szyfr Cezara. Aby wyznaczyć klucz, wystarczy bowiem sprawdzić przesunięcie zastosowane dla pierwszych liter analizowanej wiadomości.

Analiza częstości

Szyfr Cezara można złamać z wykorzystaniem **analizy częstości** wystąpień poszczególnych liter w szyfrogramie. Jeśli wiemy, w jakim języku zapisana jest informacja jawna, możemy założyć, że najczęściej występująca litera w szyfrogramie odpowiada najczęściej używanej literze w tym języku. Wówczas kluczem użytym do zaszyfrowania informacji jest różnica między pozycjami litery najczęściej występującej w szyfrogramie oraz litery najczęściej pojawiającej się w danym języku. W przypadku języka polskiego, jeśli litery alfabetu numerujemy od 0, tą różnicą będzie pozycja najczęściej występującej litery w szyfrogramie. Wynika to z tego, że w języku polskim najczęściej pojawia się litera A.

Analiza częstości liter

Warto podkreślić, że łamanie szyfru Cezara z wykorzystaniem analizy częstości nie zawsze jest skuteczne. Ta metoda może dać fałszywe wyniki np. w przypadku krótkich informacji.

Warto wiedzieć

Krótkie teksty zaszyfrowane w prosty sposób mogą być nie do złamania nawet przez potężne komputery. Claude Shannon wykazał, że im krótszy szyfrogram, tym trudniej określić rodzaj szyfru. Niektóre teksty wyryte na starych nagrobkach pozostają nieodczytane przez setki lat mimo wysiłków wielu kryptologów.

Ćwiczenie 7

Znajdź informację o tym, jaka litera występuje najczęściej w tekstach zapisanych w:

a. języku angielskim,

b. wybranym przez ciebie języku, innym niż angielski i polski.

Zapamiętaj

Każdy szyfr można złamać przez odnalezienie pasującego klucza. Stopień bezpieczeństwa algorytmu szyfrowania mierzy się ilością czasu, jakiej wymaga jego złamanie (wyczerpujące przeszukanie zbioru potencjalnych kluczy). Szyfr Cezara nie stanowi dziś praktycznie żadnego zabezpieczenia.

ANALIZA CZĘSTOŚCI W JĘZYKU POLSKIM

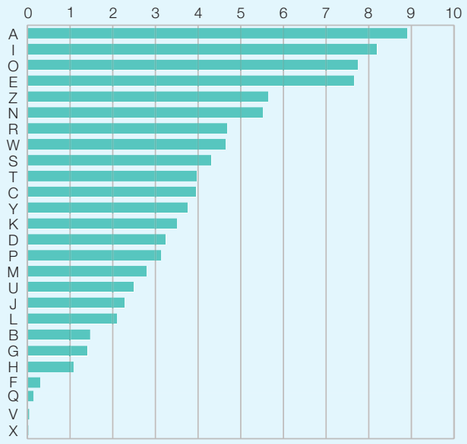
Pokażemy, jak wykorzystać analizę częstości do odszyfrowania szyfrogramu, który powstał po zaszyfrowaniu informacji w języku polskim. Posłużymy się tekstem zapisanym wyłącznie wielkimi literami alfabetu łacińskiego. W tekście nie używamy znaków interpunkcyjnych, a wyrazy są oddzielone spacjami.

Fragment szyfrogramu

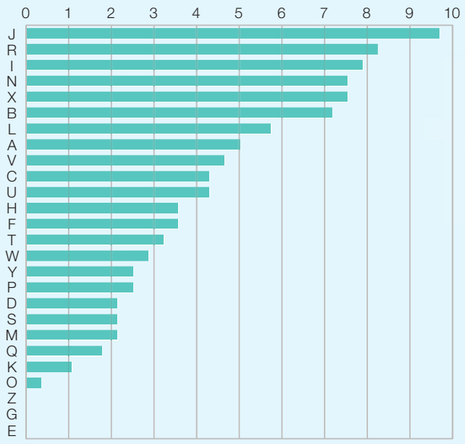
YXMLIJB UJVJWRJ BIHOAXPAJVD VXIN BRN YAIHMJL WRN CHUTX JWJURIJ
LINBCXBLR URCNA UNLI CJTIN IWJSXVXBL LINBCX FHBCNYDSJLHLQ XKXT
BRNKRN PADY MFXLQ UDK CAINLQ IWJTXF LIHUR MRPAJVXF R CARPAJVXF WJ
YAIHTUJM F SNIHTD YXUBTRV WJSLINBCBIHVR MRPAJVJVR BJ LQ LI BI R AI J
CARPAJVNV BUXFX BRN F SNIHTD JWPRNUBTRV LINBCX YXSJFRJ BRN BUXFX CQN

Deszyfrujemy szyfrogram

Przeciętna częstość występowania liter w języku polskim* [%]



Częstość występowania liter w szyfrogramie [%]



Z wykresu po prawej stronie odczytujemy, że najczęściej występującą literą w szyfrogramie jest litera J. Może to oznaczać, że litery alfabetu przesunięto o 9 pozycji. Po odszyfrowaniu szyfrogramu z użyciem klucza 9 otrzymujemy właściwą informację.

* Nie uwzględniamy częstości występowania liter z polskimi znakami diakrytycznymi.

Tekst jawny po odszyfrowaniu fragmentu kryptogramu

PODCZAS LAMANIA SZYFROGRAMU MOZE SIE PRZYDAC NIE TYLKO ANALIZA
CZESTOSCI LITER LECZ TAKZE ZNAJOMOSC CZESTO WYSTEPUJACYCH
OBOK SIEBIE GRUP DWOCH LUB TRZECH ZNAKOW CZYLI DIGRAMOW
I TRIGRAMOW NA PRZYKLAD W JEZYKU POLSKIM NAJCZESTSZYMI DIGRAMAMI
SA CH CZ SZ I RZ A TRIGRAMEM SLOWO SIE W JEZYKU ANGIELSKIM
CZESTO POJAWIA SIE SLOWO THE

Zadania

* **1** Poszukaj informacji na temat historii steganografii, czyli sztuki ukrywania informacji. Przygotuj krótką prezentację na ten temat. Uwzględnij następujące przykłady: akrostych, mikropunkt, szyfr szpilkowy.

* **2** Przygotuj prezentację na temat sposobu szyfrowania tajnej korespondencji w służbie dyplomatycznej Stanisława Augusta Poniatowskiego, ostatniego króla Polski.

* **3** Przygotuj prezentację na temat maszyny szyfrującej Enigma oraz historii złamania jej szyfru przez polskich matematyków w latach 30. XX w. Przedstaw na osi czasu kluczowe wydarzenia związane z historią łamania szyfru Enigmy przed wybuchem II wojny światowej.

** **4** Jednym z prostych szyfrów przestawieniowych jest szyfr polegający na zamianie miejscami znaków na pozycjach parzystych i nieparzystych (pierwszy znak zamieniamy miejscem z drugim, trzeci z czwartym itd.). Napisz program szyfrujący i program deszyfrujący tekst tą metodą. Przyjmij, że tekst jawny składa się z parzystej liczby liter.

** **5** Szyfr Cezara z przesunięciem (kluczem) o wartości 13 jest nazywany algorytmem ROT13. Jest on współcześnie używany w dyskusjach internetowych (w języku angielskim), aby jakaś część wiadomości nie była możliwa do odczytania przez przypadkowe osoby. Dowiedz się, dlaczego używa się przesunięcia o 13 liter. Napisz program do szyfrowania i deszyfrowania wiadomości algorytmem ROT13.

** **6** Zmodyfikuj program do szyfrowania algorytmem Cezara tak, aby pozwalał na szyfrowanie i deszyfrowanie z wykorzystaniem odwróconego szyfru Cezara.

W odwróconym szyfrze Cezara wyjściowy układ alfabetów dla klucza równego 0 ma postać:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

Alfabet szyfrowy dla klucza równego 16 ma postać:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q

*** **7** Przygotuj prezentację na temat kilku wybranych metod klasycznej kryptoanalizy. W swoim opracowaniu uwzględnij nowożytny szyfr Vigenère'a i okoliczności jego złamania w XIX w.

*** **8** Napisz program szyfrujący dla szyfru Cezara, w którym wprowadzisz rozróżnienie wielkich i małych liter.
Uwaga: Liczba kluczy wzrośnie dwukrotnie.

Algorytmika i programowanie w języku Python

Wyobraź sobie, że...

...zostałeś zatrudniony jako młodszy specjalista ds. transportu i magazynowania. Zauważasz problem związany z jednoczesnymi dostawami do magazynu części z dwóch różnych firm, realizowanymi w cyklach co 4 i 6 dni roboczych. Stwierdzasz, że można go łatwo rozwiązać, wykorzystując pojęcie **NWW**. Uzasadniasz, że dostawy w cyklach co 5 i 6 dni będą się pokrywać tylko raz na 30 dni. Wiedzę o **algorytmie Euklidesa** wykorzystujesz przy planowaniu przestrzeni magazynowych – stosujesz podejście znane ci z **wycinanki Euklidesa**.

...znalazłaś pracę jako asystentka marketingu cyfrowego. Analizując listy subskrybentów, zauważasz **zduplikowane** imiona i nazwiska, co może oznaczać, że klienci wielokrotnie zakładali konta. Dostrzegasz też literówki w adresach mailowych, spowodowane brakiem mechanizmu **sprawdzania adresu**. Zgłaszasz problem do działu IT. Poprawiasz błędy, korzystając z funkcji sprawdzania pisowni w przeglądarce internetowej – **algorytmu wyszukiwania wzorca w tekście**.

...pracujesz w dziale administracji dużej firmy. Twoim zadaniem jest wysłanie pocztą elektroniczną poufnych dokumentów do ważnego kontrahenta. Postanawiasz **zaszyfrować** przesyłane informacje. Z lekcji informatyki pamiętasz, że warunkiem tajności informacji jest utrzymanie w tajemnicy **klucza szyfrowania**. Przygotowaną wiadomość przesyłasz kontrahentowi e-mailem, ale hasło **deszyfrujące** przekazujesz w wiadomości wysłanej z bezpiecznego komunikatora.

...postanowiłaś zawodowo zająć się fotografią. Kupiłaś profesjonalny aparat, tzw. lustrzankę, i zapisałaś się na kurs. Na pierwsze zajęcia masz przynieść ze sobą kilka zdjęć. Pomimo niewielkiej wiedzy na temat fotografii starasz się wykorzystać zasadę **złotej proporcji**. Zaskakujesz prowadzących, objaśniając, że fotografowany obiekt umieściłaś dokładnie 5 jednostek od lewej krawędzi kadru i 8 jednostek od prawej, gdyż są to kolejne liczby **ciągu Fibonacciego**.

2. Algorytmika i programowanie w języku C++

5 Algorytm Euklidesa

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. Współcześnie stosuje się go m.in. w kryptograficznym algorytmie RSA, wykorzystywanym np. w podpisach cyfrowych. W tym temacie użyjemy algorytmu Euklidesa do wykonywania operacji na ułamkach.

Ważne w tym temacie

- Wyznaczanie największego wspólnego dzielnika dwóch liczb z użyciem algorytmu Euklidesa
- Algorytmy do wykonywania działań na ułamkach zwykłych
- Funkcje typu `void`, które nie zwracają wartości

5.1. Patchworkowe kołdry

Kołdra patchworkowa powstaje w wyniku zszycia wielu mniejszych kawałków tkanin. Wyobraźmy sobie, że przygotowujemy projekt małej kołdry o wymiarach 112 cm × 88 cm, która ma być uszyta z jak największych kwadratowych kawałków takiej samej wielkości (rys. 5.1). Jaką długość powinien mieć bok takiego kawałka?

Rys. 5.1. Patchworkowa kołdra uszyta z kwadratów z różnych materiałów

Zauważmy, 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 cm × 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, jak obliczyć, z ilu kawałków materiału w kształcie kwadratu o boku 8 cm będzie się składać patchworkowa kołdra o wymiarach 112 cm × 88 cm.

5.2. 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 dwóch różnych liczb bez potrzeby rozkładania liczb na czynniki pierwsze. Algorytm wykorzystujący najbardziej intuicyjną metodę to **algorytm naiwny**.

Algorytm naiwny

Oto specyfikacja problemu:

Specyfikacja

Dane: dwie różne liczby całkowite dodatnie a i b.

Wynik: największy wspólny dzielnik liczb a i b.

Rozwiązanie 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 wtedy, gdy wspólny dzielnik będzie równy 1.

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 3–5.
   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.

Oto kod źródłowy programu realizującego podany algorytm:

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 wielkość liter. Jeśli deklarujesz zmienną lub funkcję zawierającą w nazwie duże litery, to w dalszej części kodu musisz konsekwentnie używać ustalonej nazwy.

W liniach 5.–13. zdefiniowana jest funkcja `NWD`. W liniach 8.–9. znajduje się fragment kodu, który zapewnia, że liczba zapisana w zmiennej `a` będzie mniejsza od liczby zapisanej w zmiennej `b`. Jeśli wartość zmiennej `a` jest większa od wartości zmiennej `b`, to wartości zmiennych są zamieniane.

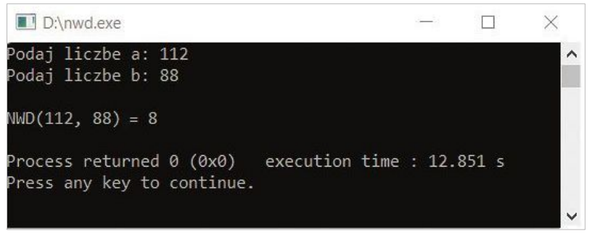
Funkcja swap

Zamianę wartości zmiennych realizujemy z wykorzystaniem wbudowanej w język C++ **funkcji** `swap` – jej argumentami są zmienne, których wartości należy zamienić (linia 9.).

Kod realizujący kluczową ideę algorytmu znajduje się w liniach 11. i 12. 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 (funkcji `main`) zwracana jest wartość `d` – odpowiada za to instrukcja `return`.

Pętla for
→ s. 211

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 5.2 na s. 63 przedstawia przykład wywołania tego programu.



Rys. 5.2. Wywołanie programu wyznaczającego NWD dwóch liczb

Zauważmy, że zmienne `a` i `b` zadeklarowano 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` zadeklarowane w funkcji `NWD` są zapisane w innych komórkach pamięci komputera.

Zwróćmy uwagę na linie 20. i 21. – w linii 20. przeplatają się napisy stałe oraz wartości zmiennych liczbowych, a w linii 21. jest wywołana funkcja `NWD`. Pozwala to wyświetlić wynik tak jak na rysunku 5.2.

Warto wiedzieć

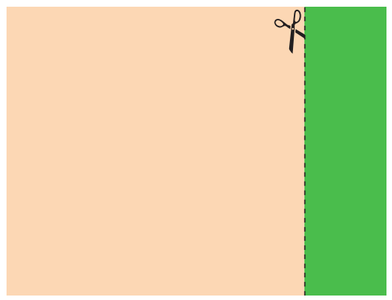
Jedną z cech dobrego języka programowania jest domyślne ograniczanie widoczności zmiennej do fragmentu programu. To programista powinien świadomie decydować, w którym fragmencie kodu dostęp do zmiennej jest naprawdę potrzebny.

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 kwadratowego kawałka kołdry o wymiarach 112 cm × 88 cm. Jak szybko można wyznaczyć wymiary takiego kwadratu bez rozkładania liczb 112 i 88 na czynniki pierwsze?

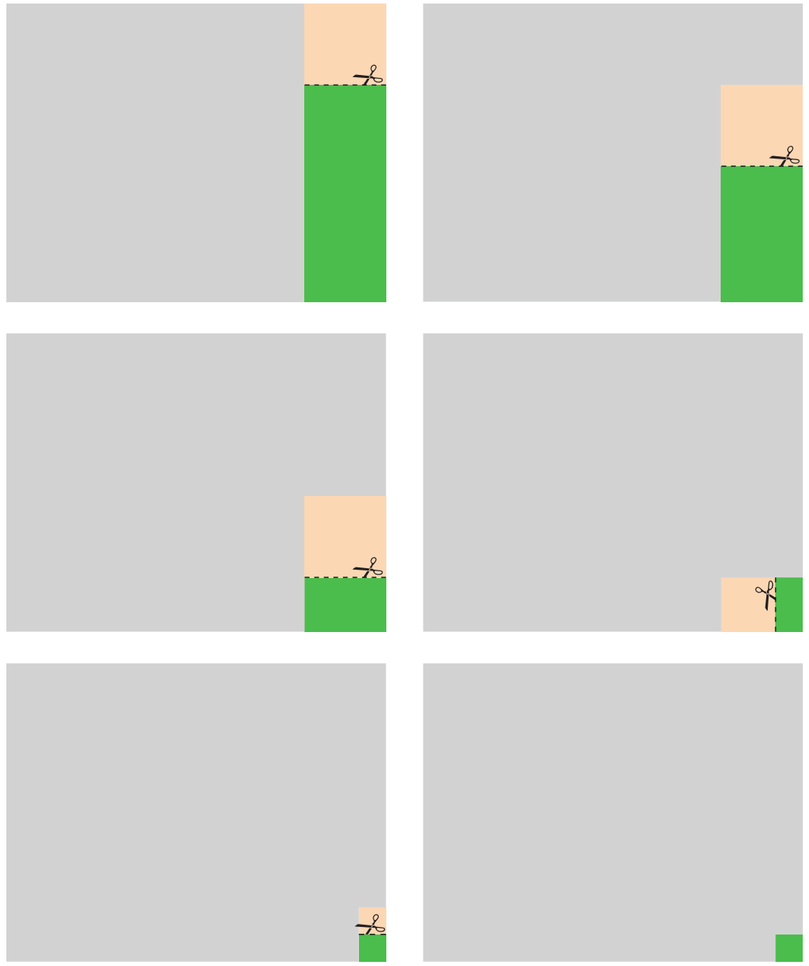


Rys. 5.3. Prostokąt o wymiarach 112 cm × 88 cm z odciętym kwadratem o boku 88 cm

Rysunek 5.3 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 5.4 na s. 64 i opisano w tabeli 5.1 na s. 64.

Warto wiedzieć

Odcinanie jak największego kwadratu od danego prostokąta to przykład zastosowania strategii algorytmicznej nazywanej podejściem zachłannym.



Rys. 5.4. 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	8

Tabela 5.1. Wymiary prostokątów i długości boków kwadratów w kolejnych krokach

Ćwiczenie 3

Zastosuj wycinankę Euklidesa dla prostokąta o bokach 21 cm × 12 cm. Użyj kartki w kratkę i nożyczek.

Dzięki wycinance Euklidesa wiemy, że kwadratami o boku 8 cm można wypełnić cały prostokąt o wymiarach 112 cm × 88 cm. 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 cm, gdyby istniał taki kwadrat dla prostokąta o wymiarach 112 cm × 88 cm. Tymczasem prostokąta o wymiarach 16 cm × 8 cm, z przedostatniego kroku, nie da się wypełnić kwadratami o boku dłuższym niż 8 cm.

To spostrzeżenie można przedstawić w postaci równości: $\mathrm{NWD}(a, b) = \mathrm{NWD}(a, b - a)$ 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 · d i b = m · d, gdzie n i m to liczby całkowite dodatnie. Wówczas:
b – a = m · d – n · d = = (m – n) · d,
czyli d dzieli różnicę b – a.

Algorytm Euklidesa – wersja z odejmowaniem

Algorytm naiwny ze s. 61 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.

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?

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 $2^{31} - 1$).

Skorzystamy z obserwacji wynikających z wycinanki i stworzymy program realizujący algorytm opisany w starożytności przez Euklidesa w jego dziele *Elementy*.

Algorytm Euklidesa w wersji z odejmowaniem można zapisać w postaci następującej listy kroków:

Algorytm Euklidesa – wersja z odejmowaniem

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.

Oto kod źródłowy funkcji NWD, która realizuje algorytm Euklidesa.

Kod źródłowy funkcji realizującej algorytm Euklidesa w wersji z odejmowaniem

```
1. int NWD(int a, int b)
2. {
3.     while (a!=b)
4.         if (a>b) a=a-b;
5.         else b=b-a;
6.     return a;
7. }
```

Po zakończeniu działania pętli while wartości obu zmiennych a i b są równe największemu wspólnemu dzielnikowi początkowych wartości liczb a i b. Dlatego funkcja NWD zwraca jedną z tych wartości (linia 6.). 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 osiągnie się 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 opisującymi długości **odcinków współmiernych**.

Odcinki współmierne to dwa odcinki, dla których istnieje trzeci odcinek, który w obu danych odcinkach mieści się całkowitą liczbę razy.

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 wykorzystuje się we współczesnej informatyce. Mimo że dzieło Euklidesa powstało ok. 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

Jeśli w algorytmie Euklidesa jedna liczba jest dużo większa od drugiej, to wielokrotnie powtarza się odejmowanie od niej tej samej liczby.

W tabeli 5.2 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 5.2. Wyznaczanie NWD zgodnie z algorytmem Euklidesa – wersja z odejmowaniem

Warto wiedzieć

Operacja dzielenia, a więc i modulo, zajmuje 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.

Wielokrotne odejmowanie możemy zastąpić wyznaczeniem reszty z dzielenia większej liczby przez mniejszą – tak jak w tabeli 5.3.

Wartość zmiennej a	Wartość zmiennej b
196	42
196 % 42 = 28	42
28	42 % 28 = 14
28 % 14 = 0	14

Tabela 5.3. 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. Zauważ, że cztery działania zastąpiliśmy jednym.

Spójrzmy na ostatni wiersz tabeli 5.3. 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**:

Algorytm Euklidesa – wersja 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, która realizuje algorytm Euklidesa w wersji z dzieleniem.

Kod źródłowy funkcji realizującej algorytm Euklidesa w wersji z dzieleniem

```
1. int NWD(int a, int b)
2. {
3.     while (a!=0 && b!=0)
4.         if (a>b) a=a%b;
5.         else b=b%a;
6.
7.     if (a!=0) return a;
8.     else return b;
9. }
```

Patrząc na tabelę 5.3 na s. 67, zauważamy, że dzielenie z resztą jest wykonywane naprzemiennie (a%b, b%a, a%b itd.). Dlatego kod funkcji NWD można zapisać bez użycia instrukcji warunkowej.

Kod źródłowy funkcji realizującej algorytm Euklidesa w wersji z dzieleniem (bez instrukcji warunkowej)

```
1.  int NWD(int a, int b)
2.  {
3.      while (b!=0)
4.      {
5.          int r=a%b;
6.          a=b;
7.          b=r;
8.      }
9.      return a;
10. }
```

W linii 6. zapisujemy w zmiennej a wartość zmiennej b, natomiast w linii 7. w zmiennej b – resztę z dzielenia a przez b, zapamiętaną w zmiennej pomocniczej r. Kiedy reszta będzie równa 0, pętla zakończy działanie. NWD to ostatnio zapamiętana wartość zmiennej a.

Ćwiczenie 6

Zmodyfikuj kod programu *euklides.cpp* otrzymany w ćwiczeniu 5., używając funkcji NWD realizującej algorytm Euklidesa w wersji z dzieleniem (bez instrukcji warunkowej). Zapisz go pod nazwą *euklides_dzielenie.cpp*. Uruchom program dla różnych par liczb.

Zapamiętaj

Algorytm Euklidesa pozwala znaleźć największy wspólny dzielnik dwóch liczb naturalnych bez szukania wspólnych dzielników tych liczb.

5.3. Działania na ułamkach zwykłych

Napiszemy teraz programy, które pozwolą wykonać wybrane działania na ułamkach zwykłych 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 fragment kodu źródłowego programu *Suma ułamków*, który realizuje powyższą specyfikację.

Fragment kodu źródłowego programu *Suma ułamków*

```
1.  int NWD(int a, int b)
2.  {
3.      while (b!=0)
4.      {
5.          int r=a%b;
6.          a=b;
7.          b=r;
8.      }
9.      return a;
10. }
11.
12. void suma(int a, int b, int c, int d)
13. {
14.     int e, f, nwd_ef;
15.     e = a*d+c*b;
16.     f = b*d;
17.     nwd_ef = NWD(e,f);
18.     cout << e/nwd_ef << "/" << f/nwd_ef << endl;
19. }
20.
```

```
21. int main()
22. {
23.     int licz_1, mian_1, licz_2, mian_2;
24.
25.     cout << "Liczba 1 - licznik: "; cin>>licz_1;
26.     cout << "Liczba 1 - mianownik: "; cin>>mian_1;
27.     cout << "Liczba 2 - licznik: "; cin>>licz_2;
28.     cout << "Liczba 2 - mianownik: "; cin>>mian_2;
29.
30.     cout << endl << licz_1 << "/" << mian_1 << " + ";
31.     cout << licz_2 << "/" << mian_2 << " = ";
32.     suma(licz_1, mian_1, licz_2, mian_2);
33.
34.     return 0;
35. }
```

W liniach 1.–10. zdefiniowano funkcję wyznaczającą NWD zgodnie z algorytmem Euklidesa (w wersji z dzieleniem). W liniach 12.–19. zdefiniowano funkcję `suma`. Jest to funkcja typu `void` (z ang. próżnia), czyli **funkcja niezwracająca wartości**. Zadaniem tej funkcji jest wykonanie obliczeń i wyświetlenie efektu na ekranie.

Funkcja niezwracająca wartości

Ćwiczenie 7

Zapisz kod źródłowy ze s. 69–70 w pliku *suma_ulamkow.cpp*. Następnie dodaj na początku pliku zapisy `#include <iostream>` oraz `using namespace std;`
Skompiluj kod i uruchom program dla różnych par ułamków.

Dodawanie ułamków – metoda szkolna

Kiedy na lekcjach matematyki dodaje się ułamki, sprowadza się je najpierw do wspólnego mianownika, którym jest 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**, czyli ich największym wspólnym dzielnikiem jest liczba 1, to $\mathrm{NWW}(a, b) = a \cdot b$.

Liczby względnie pierwsze

Istnieje **zależność między NWD a NWW** liczb a i b:

Zależność między NWD a NWW

$$\mathrm{NWW}(a, b)=\frac{a\cdot b}{\mathrm{NWD}(a, b)}$$

Można ją 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`, 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. }
```

Kod źródłowy funkcji dodającej ułamki metodą szkolną

W kodzie korzystamy z algorytmu Euklidesa zastosowanego w funkcji `NWD` oraz ze wzoru określającego zależność między NWD i NWW (kod funkcji `NWD` – s. 68).

Ćwiczenie 9

W pliku *suma_ulamkow2.cpp* zapisz kod źródłowy programu dodającego ułamki metodą szkolną. Możesz wykorzystać plik *suma_ulamkow.cpp* z ćwiczenia 7. Skompiluj kod i 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 postępuje się 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.  void iloczyn(int a, int b, int c, int d)
2.  {
3.      int e, f, nwd_ef;
4.
5.      e = (a/nwd(a,d)) * (c/nwd(b,c));
6.      f = (b/nwd(b,c)) * (d/nwd(a,d));
7.
8.      nwd_ef = nwd(e,f);
9.
10.     cout << e/nwd_ef << endl;
11.     cout << f/nwd_ef << endl;
12. }
```

Ćwiczenie 10

W pliku *iloczyn_ulamkow.cpp* zapisz 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.

Zadania

* **1** Napisz program, który po wczytaniu wymiarów (w centymetrach) patchworkowej kołdry 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. Część ułamkową program powinien podawać w postaci skróconej.

* **3** Napisz program, który – korzystając z algorytmu Euklidesa – będzie sprawdzać, czy dane dwie liczby naturalne są liczbami względnie pierwszymi.

** **4** Napisz program, który wczyta licznik i mianownik ułamka o wartości mniejszej niż 1/2 i sprawdzi, 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 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.

** **7** Uzupełnij program *Suma ułamków* ze s. 69–70 o wyodrębnianie części całkowitej i części ułamkowej dla sumy, która jest ułamkiem niewłaściwym.

** **8** Napisz program zawierający funkcję o nagłówku `int NWW(int a, int b)`, która będzie wyznaczać najmniejszą wspólną wielokrotność liczb a i b.

** **9** Zapisz specyfikację problemu dzielenia dwóch ułamków. Następnie napisz listę kroków algorytmu i program zawierający funkcję `iloraz`, wykonującą dzielenie.

*** **10** Napisz program, który wczyta dwie dodatnie i względnie pierwsze liczby a i b, a następnie wyznaczy dla nich najmniejszą możliwą dodatnią liczbę całkowitą x, dla której ułamek $\frac{a+x}{b+x}$ jest nieskracalny.

*** **11** Zmodyfikuj program *Suma ułamków* ze s. 69–70 tak, aby umożliwiał dodanie trzech ułamków.

*** **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 NWD 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.

6 Obliczanie wartości elementów ciągu

Ciągi liczbowe służą m.in do opisywania modeli świata rzeczywistego w komputerowym świecie wirtualnym. W mniej lub bardziej dokładny sposób przedstawiają one różnego rodzaju zjawiska. Na przykład pozwalają opisać zmiany temperatury w wybranym miejscu na świecie czy wzrost liczby użytkowników w sklepie internetowym. Do wyznaczania kolejnych liczb tworzących ciąg liczbowy możemy w programowaniu użyć techniki iteracji, czyli wielokrotnego powtarzania instrukcji zgodnie z określonym wzorem.

Ważne w tym temacie

- Definicja ciągu liczbowego
- Iteracyjne wyznaczanie wyrazów ciągu
- Ciąg Fibonacciego

6.1. Definicja ciągu liczbowego

Ciąg liczbowy

Pojęcia ciągu używamy, mówiąc o obiektach, których kolejność jest ustalona. **Ciąg liczbowy** to taki, którego elementami, zwanymi wyrazami ciągu, są liczby.

Kolejne wyrazy ciągu znajdują się na pozycjach określonych za pomocą liczb naturalnych. Na przykład pierwszy wyraz ciągu znajduje się na pozycji pierwszej, piąty wyraz – na piątej, a n-ty wyraz ciągu – na pozycji n-tej. Wyrazy ciągu zapisujemy symbolicznie jako a_1, a_2, a_3, ... Zapis a_n oznacza n-ty wyraz ciągu.

Ciąg skończony

Przykład **skończonego ciągu** liczbowego, czyli takiego, który ma ściśle określoną liczbę elementów, tworzą numery wagonów składu pociągu (rys. 6.1).

Rys. 6.1. Numery wagonów jako przykład skończonego ciągu liczbowego

Ciąg nieskończony

Ciąg, w którym nie istnieje ostatni element, nazywamy **ciągiem nieskończonym**. Przykładem takiego ciągu jest: 1, 2, 4, 8, 16, … Zauważ, że iloraz dwóch kolejnych wyrazów jest równy 2. Taki ciąg można opisać wzorem:

$$a_n = \begin{cases} 1 & \text{dla } n = 1 \\ 2a_{n-1} & \text{dla } n > 1 \end{cases}$$

Warto wiedzieć

Ciąg liczbowy o takiej własności, że iloraz dwóch kolejnych wyrazów jest stały, nazywa się ciągiem geometrycznym.

Przepis na ciasto francuskie

Ciasto francuskie ma w sobie dużo powietrza. Zawdzięcza to warstwowej strukturze uzyskiwanej w wyniku wielokrotnego składania i wałkowania ciasta (rys. 6.2).

Aby przygotować ciasto francuskie, należy wiele razy powtórzyć następujące czynności:

- rozwałkować ciasto na długość równą trzykrotności jego szerokości,
- złożyć ciasto dwa razy, aby uzyskać trzy warstwy, i schłodzić je.

Rys. 6.2. Wałkowanie i składanie ciasta francuskiego

Ciasto powstałe w efekcie pierwszego wałkowania możemy nazwać ciastem początkowym o numerze 1. Z niego powstaje 3-warstwowe ciasto o numerze 2, a z niego z kolei 9-warstwowe ciasto o numerze 3 itd.

Zauważmy, że i-te ciasto powstaje z ciasta o numerze $i - 1$, które po złożeniu tak, żeby miało trzy warstwy, zostaje rozwałkowane. Ciąg określający liczbę warstw ciasta to: 1, 3, 9, 27, ... Można go opisać w następujący sposób:

$$c_n = \begin{cases} 1 & \text{dla } n = 1 \\ 3c_{n-1} & \text{dla } n > 1 \end{cases}$$

Ćwiczenie 1

Oblicz liczbę warstw ciasta przy założeniu, że procedura wałkowania jest powtarzana sześć razy.

Program wyznaczający n-ty wyraz ciągu

Napiszemy program wyznaczający n-ty wyraz ciągu. Oto specyfikacja problemu:

Specyfikacja

Dane: liczba naturalna n.

Wynik: n-ty wyraz ciągu opisanego wzorem:

$$c_n = \begin{cases} 1 & \text{dla } n = 1 \\ 3c_{n-1} & \text{dla } n > 1 \end{cases}$$

Do wyznaczenia *n*-tego wyrazu ciągu zastosujemy pętlę `for`, gdyż liczba cykli pętli potrzebnych do wyznaczenia *n*-tego wyrazu ciągu jest z góry określona: będzie ich $n - 1$.

Kod źródłowy programu wyznaczającego *n*-ty wyraz ciągu

```
1. int Ciag(int n)
2. {
3.     int c = 1;
4.     for (int i=2; i<=n; i++) c = 3 * c;
5.     return c;
6. }
```

W linii 3. zmiennej `c` przypisywana jest wartość 1. Jeśli użytkownik wywoła funkcję `Ciag` dla wartości 1, to pętla z linii 4. nie wykona się ani razu, ponieważ nie będzie spełniony warunek `i<=n`. W pozostałych przypadkach pętla wykona się $n - 1$ razy.

Ćwiczenie 2

a. Zapisz kod źródłowy programu *Ciag* wyznaczającego *n*-ty wyraz ciągu. Skompiluj kod i uruchom program dla różnych wartości.

b. Zmodyfikuj kod funkcji `Ciag` tak, aby program wyznaczał *n*-ty wyraz ciągu a_n zdefiniowanego na s. 74.

Warto wiedzieć

Leonardo z Pizy (1175–1250) był włoskim matematykiem. W młodości towarzyszył ojcu, który był kupcem, w podróżach handlowych i uczył się matematyki u arabskich nauczycieli. W 1202 r. opublikował swoje pierwsze dzieło matematyczne *Liber abaci* (*Księga rachunków*). Zawierało ono prawdopodobnie całą ówczesną wiedzę o arytmetyce. Na stronie tytułowej Leonardo podpisał się *filius Bonacci* (syn Bonacciego). W XIX w. historycy nadali mu przydomek Fibonacci.

Zapamiętaj

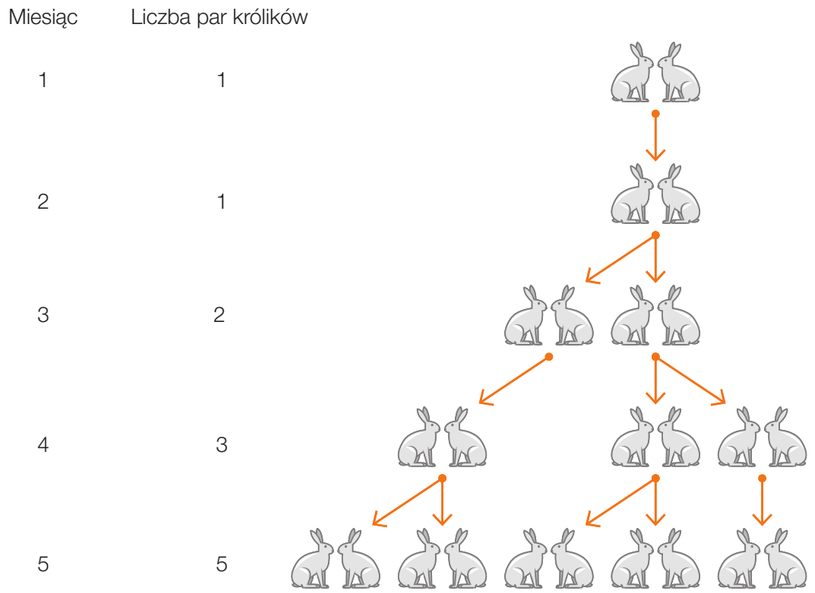
Ciąg liczbowy to ciąg, którego wyrazami są liczby. Kolejne wyrazy ciągu znajdują się na pozycjach określonych za pomocą liczb naturalnych. Do obliczenia wartości *n*-tego wyrazu ciągu stosujemy pętlę `for`, gdyż liczba cykli pętli potrzebnych do wyznaczenia *n*-tego wyrazu ciągu jest z góry określona: będzie ich $n - 1$.

6.2. Ciąg Fibonacciego

W 1202 r. Leonardo z Pizy, zwany Fibonaccim, przedstawił zadanie problemowe, które opisywało wzrost populacji królików. Pewien gospodarz kupił parę królików. Ile par królików w sumie będzie miał po 12 miesiącach, jeśli każda para rodzi co miesiąc po jednej parze, która staje się zdolna do rozmnażania po miesiącu życia?

Przeanalizujmy, jak rośnie liczba par królików w hodowli. W pierwszym miesiącu mamy parę młodych królików, która na początku drugiego miesiąca staje się płodna. Na początku trzeciego miesiąca tej parze urodzi się para młodych. Razem będą więc 2 pary królików. W czwartym miesiącu ze starszej pary zrodzi się kolejna para – mamy więc 2 + 1 = 3 pary. Ile par królików będzie na początku kolejnego, tj. piątego miesiąca?

Sposób, w jaki zmienia się populacja królików w kolejnych miesiącach, wyjaśnia rysunek 6.3.



Rys. 6.3. Para królików i jej potomstwo po pięciu pierwszych miesiącach

Oznaczmy liczbę par na początku *n*-tego miesiąca jako f_n. Otrzymamy ciąg liczb: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, … (patrz rys. 6.3). W *n*-tym miesiącu ($n > 2$) liczba par królików jest równa sumie par z dwóch poprzednich miesięcy: $n - 2$ i $n - 1$. Współcześnie zwykle dodaje się na początku ciągu dodatkowy wyraz $f_0 = 0$.

Otrzymany ciąg nazywamy **ciągiem Fibonacciego**. Początkowe dwa wyrazy tego ciągu to 0 i 1, a każdy następny jest sumą poprzednich dwóch. Ciąg Fibonacciego ma następującą definicję:

Ciąg Fibonacciego

$$f_n = \begin{cases} 0 & \text{dla } n = 0 \\ 1 & \text{dla } n = 1 \\ f_{n-1} + f_{n-2} & \text{dla } n > 1 \end{cases}$$

A to ciekawe

Fibonacci w muzyce

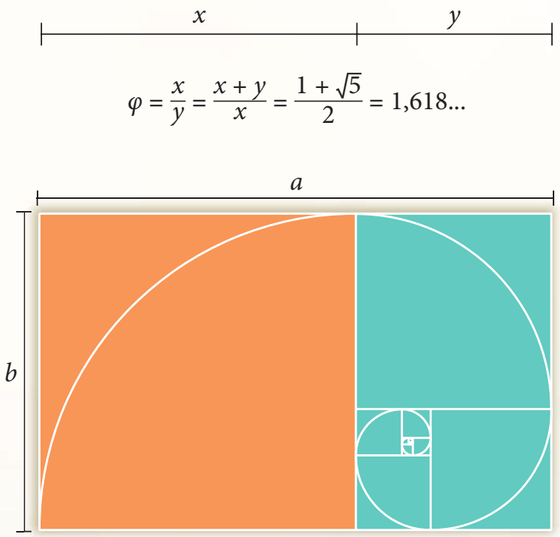
Ciąg Fibonacciego jest chętnie wykorzystywany przez kompozytorów muzyki klasycznej. Zastosowanie kolejnych wyrazów ciągu do wyznaczania punktów kulminacyjnych utworu czy długości następujących po sobie fraz muzycznych pozwala uzyskać intrygującą, a jednocześnie harmonijnie brzmiącą kompozycję.

ZŁOTA PROPORCJA

Złota proporcja, nazywana też złotym podziałem, znana jest od wieków. Przyrodnicy zachwycają się jej pięknem, którego przykłady można odnaleźć w naturze. Zainspirowało to wiele osób do poszukiwania i stosowania złotego podziału w różnych dziedzinach życia.

Czym jest złota proporcja?

Złota proporcja wyraża podział odcinka na dwie części w taki sposób, że stosunek jego dłuższej części (x) do krótszej (y) jest taki sam jak stosunek długości całego odcinka ($x + y$) do części dłuższej (x). Stosunek ten oznacza się grecką literą φ (czytaj: fi). Co ciekawe, przybliżeniami liczby φ są ilorazy kolejnych liczb ciągu Fibonacciego: 3/2, 5/3, 8/5 itd.



$$\varphi = \frac{x}{y} = \frac{x+y}{x} = \frac{1+\sqrt{5}}{2} = 1{,}618...$$

Jeżeli w prostokąt o bokach a, b będących w złotej proporcji wpiszemy kwadrat (zaznaczony na pomarańczowo), to uzyskamy mniejszy prostokąt (na rysunku turkusowy) o bokach pozostających w złotej proporcji. Jeżeli powtórzymy tę procedurę wpisywania kwadratów w kolejne mniejsze prostokąty i w każdy kwadrat wpiszemy ćwiartkę okręgu, otrzymamy złotą spiralę.

Złota proporcja w naturze

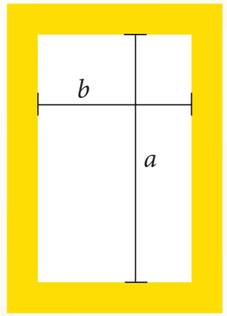
Zaskakująco wiele przykładów złotego podziału odnajdziemy w przyrodzie. Zbliżone do niego są m.in. proporcje w ciele człowieka. Badacze zauważyli też występowanie złotej proporcji na poziomie atomowym.

Huragan obserwowany z kosmosu ma kształt zbliżony do złotej spirali

Złota spirala w muszli

Złota proporcja w projektowaniu

Również graficy i projektanci wykorzystują złotą proporcję. Odnajdziemy ją w logotypach wielu słynnych marek, takich jak Toyota. W złotej proporcji pozostają także m.in. wymiary kart płatniczych.



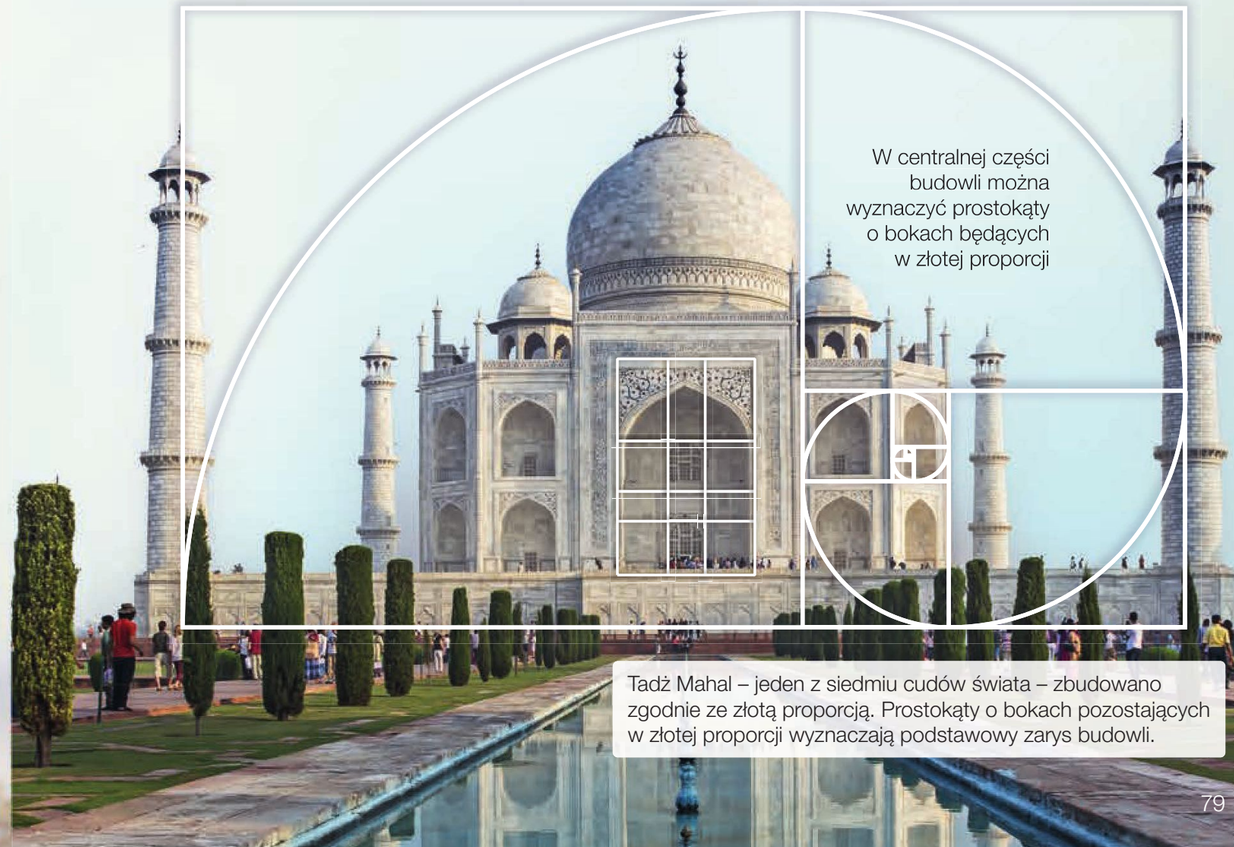
Wewnętrzny prostokąt w logo National Geographic zachowuje złotą proporcję

Na stronie wyszukiwarki Google złoty podział występuje zarówno w logotypie, jak i w układzie strony

$$\frac{a}{b} \approx \frac{b}{c} \approx \varphi$$

Złota proporcja w architekturze i sztuce

Złoty podział stosowano już w starożytnych konstrukcjach, m.in. w Wielkiej Piramidzie i Partenonie. Można go również odnaleźć w obrazach Leonarda da Vinci i Salvadora Dalego.



W centralnej części budowli można wyznaczyć prostokąty o bokach będących w złotej proporcji

Tadż Mahal – jeden z siedmiu cudów świata – zbudowano zgodnie ze złotą proporcją. Prostokąty o bokach pozostających w złotej proporcji wyznaczają podstawowy zarys budowli.

Warto wiedzieć

Definicja ciągu, w której podajemy, w jaki sposób kolejne elementy są tworzone z poprzednich, w matematyce nazywa się wzorem rekurencyjnym ciągu.

Napiszemy program obliczający dany wyraz ciągu Fibonacciego. Przyjmijmy następującą specyfikację problemu:

Specyfikacja

Dane: liczba całkowita $n \geqslant 0$.

Wynik: n-ty wyraz ciągu Fibonacciego.

Program *Fibonacci*

W każdym momencie wyznaczania wartości nowych wyrazów ciągu wystarczy wiedza o dwóch ostatnich jego wyrazach. Zapamiętajmy w zmiennej `a` wartość ostatniego wyrazu, a w zmiennej `b` – przedostatniego. Jeśli ustalimy początkową wartość zmiennej `b` jako $f_0 = 0$, wartość zmiennej `a` jako $f_1 = 1$ oraz wykorzystamy dodatkową zmienną pomocniczą `pom = a + b`, to będziemy mogli generować kolejne liczby ciągu Fibonacciego. Zmienna `pom` przechowa wartość kolejnej liczby Fibonacciego.

Oto kod źródłowy funkcji `Fib`, która wyznacza n-ty wyraz ciągu:

Kod źródłowy funkcji `Fib` w programie *Fibonacci*

```
1.  int Fib(int n)
2.  {
3.      int pom;
4.      int a = 1, b = 0;
5.
6.      for (int i=0; i<n; i++)
7.      {
8.          pom = a + b;
9.          b = a;
10.         a = pom;
11.     }
12.     return b;
13. }
```

Warto wiedzieć

Ciąg Fibonacciego można odnaleźć w przyrodzie – kwiat przebiśniegu ma 3 płatki, kaczeńca – 5, ostróżki – 8, nagietka – 13, astra – 21, a stokrotki – 34, 55 lub 89.

W linii 4. nadajemy początkowe wartości zmiennym `a` i `b`. W pętli (linie 6.–11.) obliczamy wartość kolejnego wyrazu zapisanego w zmiennej `pom`. Następnie zmieniamy wartości zmiennych tak, aby w zmiennej `b` znalazł się wyraz zapamiętany wcześniej w zmiennej `a` (linia 9.), natomiast w zmiennej `a` następny wyraz (linia 10.). Po n iteracjach zmienna `b` przyjmie wartość f_n. Zwróćmy uwagę na to, że dla $n = 0$ pętla nie wykona się ani razu.

Ćwiczenie 3

Napisz program obliczający dany wyraz ciągu Fibonacciego według podanej wcześniej specyfikacji. Wykorzystaj funkcję `Fib`. Sprawdź działanie programu dla liczb: 5, 10 i 45.

Zadania

* **1** Napisz program, który wypisze na ekranie ciąg liczb: 1, 3, 5, 7, 9, …, 99.

* **2** Napisz program, który wypisze na ekranie ciąg liczb: 1, –2, 4, –8, 16, –32, …, 4096.

* **3** Napisz program, który wypisze na ekranie wszystkie liczby dwucyfrowe.

* **4** Napisz program wypisujący na ekranie kolejne trzycyfrowe liczby parzyste.

* **5** Napisz program, który wypisze 10 elementów ciągu określonego wzorem:

$$b_n = \begin{cases} 1 & \text{dla } n = 1 \\ 1 + 2b_{n-1} & \text{dla } n > 1 \end{cases}$$

** **6** Napisz program, który po podaniu liczby n wypisze na ekranie ciąg liczb: n, $-2n$, $4n$, $-8n$, $16n$, $-32n$, …, $4096n$.

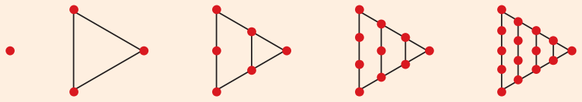
** **7** Napisz program wyznaczający sumę n kolejnych liczb nieparzystych. Na przykład dla $n = 3$ program powinien policzyć sumę $1 + 3 + 5$ i wypisać 9.

** **8** Napisz program, który po podaniu liczby n ($n > 2$) wypisze na ekranie ciąg o dwóch pierwszych wyrazach równych 1. Każdy kolejny wyraz ciągu powinien powstać przez zsumowanie poprzednich trzech wyrazów. Na przykład dla $n = 9$ program powinien wypisać: 1, 1, 2, 4, 7, 13, 24, 44, 81.

*** **9** Iloczyn liczby naturalnej n przez wszystkie liczby naturalne dodatnie mniejsze od tej liczby nazywa się silnią liczby n i oznacza jako $n!$. Napisz program obliczający wartość $n!$.

*** **10** Napisz program, który wyświetli wartości kolejnych ilorazów: f_3/f_2, f_4/f_3, f_5/f_4 itd. aż do momentu, gdy sąsiednie ilorazy różnić się będą o mniej niż 0,0001. Wartości f_n są kolejnymi wyrazami ciągu Fibonacciego.

*** **11** Pitagorejczycy odczuwali mistyczny związek z ciągiem liczb 1, 3, 6, 10, 15, 21, … Pierwszym wyrazem tego ciągu jest 1. Aby obliczyć n-ty wyraz, należy dodać n do poprzedniego wyrazu.



Liczby w tym ciągu nazywane są liczbami trójkątnymi, ponieważ można je sobie wyobrazić jako układ trójkątnych obiektów. Napisz program, który wyznaczy liczbę wszystkich punktów konstrukcji n-tej liczby trójkątnej.

7 Algorytmy na tekstach

Błędy językowe, literówki i inne pomyłki w tekście zwykle świadczą o niewiedzy lub nieuwadze piszącego. Czasami mogą mieć naprawdę poważne konsekwencje. Na przykład skład leku i sposób jego dawkowania powinny być zawsze szczegółowo i precyzyjnie opisane w ulotce. W takim miejscu nawet przypadkowa zmiana znaków w tekście może być kluczowa. Programy przetwarzające tekst oferują wiele narzędzi, m.in. wyszukiwanie, automatyczne podkreślanie powtarzających się wyrazów czy możliwość wielokrotnej zmiany podanej frazy na inną. W tym temacie nauczysz się przetwarzać teksty z wykorzystaniem programów napisanych w języku C++.

Ważne w tym temacie

- Kodowanie znaków
- Algorytmy porównywania tekstów oraz wyszukiwania wzorca w tekście
- Używanie zmiennych typu `char` i typu `string` w języku C++
- Stosowanie funkcji z biblioteki `string`: `length`, `find` i `rfind`

7.1. Litery i inne znaki jako liczby

W serwisach społecznościowych często używa się piktogramów zwanych **emoji**, czyli symboli rysunkowych uzupełniających tekst o graficzne wyrażenie emocji lub przedstawienie jakiegoś obiektu – przedmiotu, miejsca, rośliny itp.

Emoji

Jeśli umieścimy na portalu społecznościowym emoji 🌻, to widzimy je dlatego, że piktogram ten znajduje się w zbiorze znaków czcionki na naszym urządzeniu. Emoji 🌻 jest w takim zbiorze na pozycji o numerze 127 803. Mówimy, że jest to **kod liczbowy znaku** (w tym przypadku piktogramu). Kody wszystkich piktogramów, a także wszelkich innych znaków (liter oraz symboli m.in. matematycznych i muzycznych) znajdują się w **tablicy znaków Unicode**.

Kod liczbowy znaku

Tablica znaków Unicode

Warto wiedzieć

Emoji wywodzą się z japońskich telefonów komórkowych z końca lat 90. XX w. Nazwa pochodzi od japońskich słów: 絵 (obraz) i 文字 (litera).

Ćwiczenie 1

a. Uruchom edytor tekstu i wybierz czcionkę Segoe UI Emoji lub Segoe UI Symbol. Następnie, przytrzymując wciśnięty lewy klawisz **Alt**, wybierz w części numerycznej klawiatury kolejno cyfry: **128190**. Jaki symbol kryje się pod tym kodem?

b. Korzystając z dostępnych serwisów internetowych, sprawdź, jaki kod ma emoji z piłką nożną ⚽.

Tabela 7.1 przedstawia fragment tablicy Unicode dla emoji o kodach od 127 800 do 127 819.

	0	1	2	3	4	5	6	7	8	9
127 80_	🌸	🌹	🌺	🌻	🌼	🌽	🌾	🌿	🍀	🍁
127 81_	🍂	🍃	🍄	🍅	🍆	🍇	🍈	🍉	🍊	🍋

Tabela 7.1. Fragment tablicy znaków Unicode zawierający wybrane piktogramy

Dobra rada

Większość laptopów ma klawisze, które symulują numeryczną część tradycyjnej klawiatury QWERTY. Zazwyczaj są to klawisze **7**, **8**, **9**, **U**, **I**, **O**, **J**, **K**, **L** oraz **M**. Klawisz **Num Lock** znajdziesz zwykle w prawej górnej części klawiatury laptopa.

Podstawowe znaki, takie jak litery alfabetu łacińskiego, cyfry, znaki interpunkcyjne i symbole (np. @ i ^), które możemy wprowadzić do komputera bezpośrednio z klawiatury, znajdziemy w początkowej części tablicy Unicode (na pozycjach od 0 do 127). Jest ona nazywana **tablicą ASCII** (skrót od ang. *American Standard Code for Information Interchange*). System kodowania znaków ASCII opracowano w latach 60. XX w. na potrzeby telekomunikacji. Powstałe później systemy kodowania znaków są w pełni kompatybilne z ASCII.

Tablica ASCII

Typ znakowy w języku C++

W języku C++ możemy deklarować zmienne przechowujące znaki. Służy do tego **typ `char`**, nazywany **znakowym typem danych** (ang. *character data type*). Zmienna tego typu to liczba z zakresu od 0 do 127, która jest kodem liczbowym odpowiedniego znaku w tablicy ASCII. Fragment tej tablicy przedstawia tabela 7.2.

Typ `char` (typ znakowy)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Tabela 7.2. Wybrane znaki ASCII – fragment tablicy Unicode

Warto wiedzieć

Kod ASCII jest czasami wykorzystywany w filmach fabularnych. Na przykład w filmie *Marsjanin* główny bohater otrzymał z NASA zakodowaną wiadomość: `48 4F 57 41 4C 49 56 45`. Korzystając z tablicy kodów ASCII, odczytał ją jako `HOW ALIVE`, czyli pytanie o to, jak udało mu się przeżyć na Marsie.

Zazwyczaj tablicę ASCII przedstawia się tak, że znaki w niej zapisane są pogrupowane po 16. W opisie kolumn pojawiają się kolejne cyfry systemu szesnastkowego (od 0 do F). Aby odczytać kod liczbowy, czyli **kod ASCII** znaku, parę liczb, czyli numer wiersza i numer kolumny, traktuje się jako kod znaku zapisany w systemie szesnastkowym. Na przykład znak @ ma kod 40_{16}, czyli 64 dziesiętnie, a znak spacji ma kod 20_{16}, czyli 32 dziesiętnie.

Kod ASCII

Program *Znaki ASCII*

Napiszemy teraz program, który wyświetli fragment tablicy ASCII pokazany w tabeli 7.2 na s. 83. Kody widocznych tam znaków mieszczą się między 48 a 126. Program powinien wyświetlić znaki z odstępami w wierszach po 16 znaków (nie licząc spacji).

Oto kod źródłowy programu *Znaki ASCII*:

Kod źródłowy programu *Znaki ASCII*

```
1.  #include <iostream>
2.
3.  using namespace std;
4.
5.  int main()
6.  {
7.      char znak;
8.
9.      for (int i=48; i<=126; i++)
10.     {
11.         znak = i;
12.         cout << znak << " ";
13.         if (i%16 == 15)
14.             cout << endl;
15.     }
16.     cout << endl;
17.     return 0;
18. }
```

W linii 7. znajduje się deklaracja zmiennej znak typu char. W liniach 9.–15. wykonywana jest pętla for, która realizuje wyświetlanie znaków w następujący sposób: zmiennej znak przypisywana jest liczba (kod ASCII znaku) równa wartości zmiennej i sterującej pętlą (linia 11.). Następnie w linii 12. realizujemy wyświetlenie znaku, a po nim pojedynczej spacji (zapisujemy ją w cudzysłowie). Zapisy w liniach 13. i 14. służą temu, aby znaki wyświetlić w wierszach po 16 znaków.

Rys. 7.1. Wywołanie programu *Znaki ASCII*

Na rysunku 7.1 przedstawiono wywołanie programu *Znaki ASCII*.

Układ tablicy Unicode zawierającej kody ASCII jest starannie przemyślany. Na przykład pary wielkich i małych liter („A” i „a”, „B” i „b” itd.) znajdują się w tych samych kolumnach i są oddalone o dwa wiersze (różnica kodów wynosi 32). Dlatego łatwo z kodu małej litery przejść do kodu odpowiadającej jej wielkiej litery i na odwrót.

Warto wiedzieć

Deklaracja zmiennej typu char skutkuje zarezerwowaniem w pamięci jednego bajta (czyli 8 bitów), w którym znajdzie się binarny zapis liczby będącej kodem ASCII danego znaku. Oto zapis binarny kodu litery A (wynoszącego 65):

0	1	0	0	0	0	0	1

Ćwiczenie 2

Zmodyfikuj kod źródłowy programu *Znaki ASCII* z poprzedniej strony tak, aby program wyświetlił znaki, które w tablicy ASCII mają kody liczbowe między 32 a 47.

Przypisanie kodu znaku z wykorzystaniem apostrofów

Aby przypisać wartość zmiennej typu char, nie trzeba znać liczb przyporządkowanych znakom w tablicy ASCII. Zmiennej można nadać wartość liczbową niejawnie, podając wybrany znak między apostrofami, np.:

```
char znak = '~';
```

Gdybyśmy od razu wiedzieli, że program ma wyświetlić wszystkie znaki znajdujące się w tablicy ASCII między cyfrą 0 a znakiem ~ (patrz rys. 7.1, s. 84), to kod źródłowy mógłby wyglądać następująco:

Kod źródłowy programu *Znaki ASCII 2*

```
1.  #include <iostream>
2.
3.  using namespace std;
4.
5.  int main()
6.  {
7.      char znak = '0';
8.
9.      while (znak <= '~')
10.     {
11.         cout << znak << " ";
12.         if (znak%16 == 15)
13.             cout << endl;
14.         znak = znak + 1;
15.     }
16.     cout << endl;
17.     return 0;
18. }
```

W linii 7. deklarujemy zmienną znak i przypisujemy jej cyfrę 0. Wygodnie będzie zastosować **pętlę** while, która będzie wykonywana, dopóki liczba określająca kod ASCII danego znaku będzie mniejsza od kodu ASCII znaku ~ lub jemu równa.

W pętli wypisywane są: znak, którego kod ASCII jest przechowywany w zmiennej znak, oraz spacja (linia 11.). Zapisy w liniach 12.–13. odpowiadają za wypisywanie znaków po szesnaście, a w linii 14. zmiennej znak przypisywany jest kod ASCII kolejnego znaku. Zapisy w liniach 9. i 14. są poprawne dlatego, że w pamięci komputera znaki są zapisane jako kolejne liczby całkowite, a więc są w ustalonym porządku.

Dobra rada

Apostrof znajdziesz na tym samym klawiszu co znak cudzysłowu.

Warto wiedzieć

Kod ASCII był pierwotnie przeznaczony na potrzeby usługi telegraficznej zwanej teleksem. Nadawca zapisywał tekst na klawiaturze urządzenia zwanego dalekopisem, a odbiorca otrzymywał wydruk. Dlatego tablica ASCII na pozycjach od 0 do 31 i na pozycji 127 zawiera kody przeznaczone do sterowania drukarką.

Pętla while
→ s. 212

Ćwiczenie 3

a. Zapisz kod źródłowy programu *Znaki ASCII 2* pod nazwą *znaki_ASCII_2.cpp*. Skompiluj kod i sprawdź działanie programu.

b. Napisz program, który wyświetli litery od A do Z w dwóch wierszach w następujący sposób:

```
ABCDEFGHIJKLM
ZYXWVUTSRQPON
```

7.2. Szukanie literówek. Łańcuchy znaków w języku C++

Warto wiedzieć

Edytory tekstu, a także gry słowne (np. Scrabble, Literaki czy Słowotok) zazwyczaj mają swoje oficjalne słowniki, w których sprawdzane są podane wyrazy.

Zapobiegać powstawaniu literówek można na różne sposoby. Na przykład w edytorze tekstu słowa zapisywane przez użytkownika zwykle są porównywane ze słowami znajdującymi się w słowniku. Innym sposobem jest wymaganie od użytkownika dwukrotnego wpisania tekstu, np. PIN-u, adresu e-mail lub hasła.

Poprawność adresu e-mail – porównanie dwóch adresów

Podczas zakładania konta w serwisie internetowym często wymaga się od nas, abyśmy dwukrotnie podali adres poczty elektronicznej. Napiszemy program, który będzie symulował taką rejestrację w serwisie internetowym.

Oto specyfikacja problemu:

Specyfikacja

Dane: cztery napisy oznaczające kolejno: imię, nazwisko oraz adres e-mail (podany dwukrotnie).

Wynik: komunikat „Hasło do serwisu wysłaliśmy na adres: <adres_e-mail>”, jeśli wpisane adresy są identyczne, w przeciwnym wypadku komunikat „Podane adresy e-mail są różne!”.

Sprawdzenie, czy dwa adresy są identyczne, sprowadza się do porównania wartości dwóch zmiennych, w których są one przechowywane. Napisy składające się ze znaków ASCII można zapamiętać w zmiennych **typu** `string`, czyli zmiennych typu tekstowego.

Typ `string`

Aby móc deklarować zmienne typu `string`, należy użyć **biblioteki** `string`, czyli wpisać na początku kodu źródłowego programu dyrektywę: `#include <string>`. Zmienne typu tekstowego nazywać będziemy **napisami**, **łańcuchami znaków** (ang. *strings of characters*) lub po prostu **łańcuchami**.

Biblioteka `string`

Napis (łańcuch znaków, łańcuch)

Podobnie jak w przypadku wypisywania napisów na ekranie, aby przypisać zmiennej typu `string` wartość (w postaci napisu), umieszcza się ją w cudzysłowie.

Oto kod źródłowy funkcji `main` programu *Porównanie adresów*:

Fragment kodu źródłowego programu *Porównanie adresów*

```
1.  int main()
2.  {
3.      const string INF_OK =
4.          "Haslo do serwisu wyslalismy na adres: ";
5.      const string INF_BLAD =
6.          "Podane adresy e-mail sa rozne!";
7.
8.      string imie, nazwisko, adres_1, adres_2;
9.
10.     cout << "Podaj imie: "; cin >> imie;
11.     cout << "Podaj nazwisko: "; cin >> nazwisko;
12.
13.     cout << "Podaj adres e-mail: "; cin >> adres_1;
14.     cout << "Powtorz adres e-mail: "; cin >> adres_2;
15.     cout << endl;
16.
17.     if (adres_1 == adres_2)
18.         cout << INF_OK + adres_1 << endl;
19.     else
20.         cout << INF_BLAD << endl;
21.
22.     return 0;
23. }
```

Dobra rada

Aby kod był czytelniejszy, długie napisy przypisywane stałym umieszczaj w tekście programu od nowej linii. Podczas kompilacji znak nowej linii (podobnie jak spacja) po operatorze przypisania jest ignorowany.

Warto wiedzieć

Wszelkiego rodzaju dane zapisane bezpośrednio w kodzie programu nazywa się literałami. Liczba 23 w instrukcji `int suma = 23;` to literał całkowity, a napis ABC w instrukcji `string napis = "ABC";` to literał napisowy.

Ponieważ oba komunikaty ustalone w specyfikacji są dla użytkownika niezmienne, możemy je zdefiniować jako **stałe** typu `string` (linie 3.–6.). Zauważ, że słowo kluczowe `const` znajduje się przed nazwą i typem zapamiętywanej wartości. Kompilator potraktuje ją jako wartość tylko do odczytu (ang. *read-only*) i nie pozwoli na jej zmianę. Dobrą praktyką jest zapisywanie nazw stałych wielkimi literami (tak jak w liniach 3. i 5.). Dzięki temu nie będą się one mylić z nazwami zmiennych.

Stała

Dobra rada

Pamiętaj, że w języku C++ rozróżniana jest wielkość liter. Nazwy zadeklarowanej zmiennej używaj dokładnie tak samo.

W linii 8. deklarujemy zmienne typu `string`, w których będą zapamiętane dane wpisane z klawiatury przez użytkownika (pobierane w liniach 10.–11. i 13.–14.). Zauważ, że napisy jako wartości zmiennych i stałych typu `string` deklarujemy pomiędzy parą znaków `" "`.

Logika matematyczna i operatory logiczne
→ s. 198–200

Warunek logiczny w linii 17. sprawdza, czy zachodzi równość między wartościami dwóch zmiennych typu `string`. Jeśli tak, wypisywany jest komunikat o poprawnej rejestracji (`INF_OK`) uzupełniony o podany adres e-mail (linia 18.). Do złączenia (scalenia) napisów wykorzystujemy **operator** `+`. Operację tę nazywa się **złączeniem** lub **konkatenacją** (ang. *concatenation*). Jeśli podane napisy są różne, program wypisze na ekranie komunikat zapisany w stałej `INF_BLAD` (linia 20.).

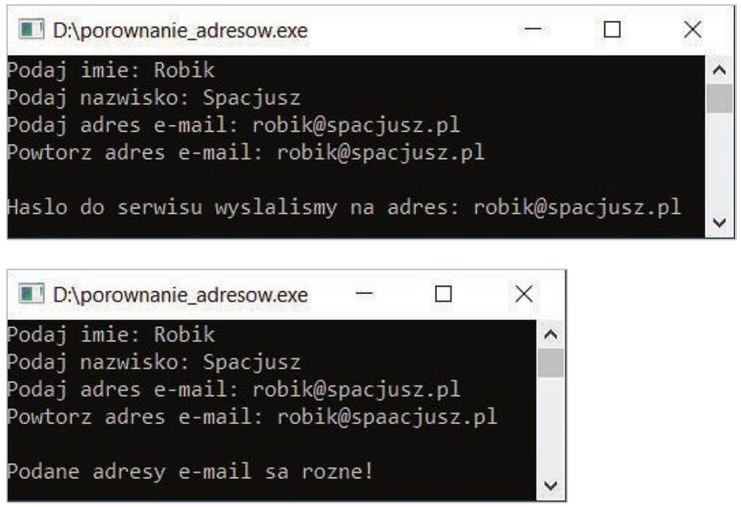
Operator +

Operacja złączenia (konkatenacja)

Warto wiedzieć

W teorii literatury konkatenacją nazywa się konstrukcję, która polega na powieleniu tego samego słowa lub wyrażenia (niekoniecznie w tej samej formie fleksyjnej) na końcu jednego wersu i na początku następnego. Przykład zastosowania konkatenacji można znaleźć w wierszu Juliana Tuwima *Rzepka*.

Rysunek 7.2 przedstawia przykładowe wywołania programu dla dwóch sytuacji: poprawnego i niepoprawnego podania adresów e-mail.



Rys. 7.2. Wywołania programu *Porównanie adresów*

Ćwiczenie 4

Zapisz kod źródłowy programu *Porównanie adresów* w pliku pod nazwą *porownanie_adresow.cpp*. Skompiluj kod i sprawdź działanie programu dla takich samych oraz różnych adresów e-mail.

Poprawność adresu e-mail – sprawdzenie struktury adresu

Program *Porównanie adresów* można rozbudować o sprawdzanie, czy napis podany przez użytkownika w polu „Podaj adres e-mail:" spełnia warunki poprawnego adresu poczty elektronicznej. Napiszemy program *Sprawdzenie adresu*, który będzie wykonywał to zadanie.

Dla uproszczenia przyjmiemy, że za poprawny adres e-mail będzie uznany napis, w którym znajdują się: dokładnie jeden znak @, który jest co najmniej trzecim znakiem napisu, oraz kropka, która jest czwartym lub trzecim znakiem od końca napisu i nie występuje bezpośrednio po znaku @.

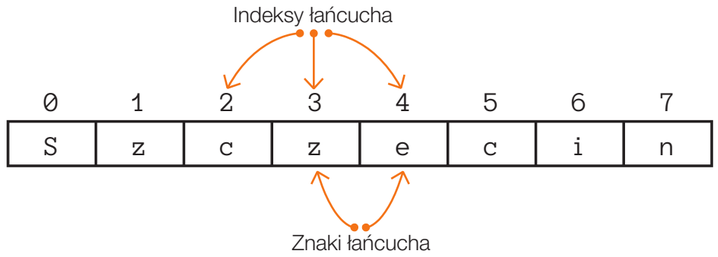
Warto wiedzieć

Domeny najwyższego rzędu zazwyczaj są dwu- lub trzyliterowe. Należą do nich m.in. domeny: .pl, .us, .org, .com, .gov.

Przykładami poprawnych adresów będą: nowak@abcd.pl oraz nk@ab.com. Niepoprawne adresy to np.: n@abc.pl, nk@w@.pl, n.@pl, a także nk@.com. W poprzednim programie porównywaliśmy całe napisy – tym razem będziemy porównywać pojedyncze znaki.

Na łańcuch znaków można patrzeć jak na ciąg komórek w pamięci komputera. Pierwszy znak jest przechowywany w komórce o indeksie 0, drugi – w komórce o indeksie 1, a ostatni – w komórce o indeksie $n - 1$, gdzie n to liczba znaków łańcucha (rys. 7.3). Zauważ, że w **tablicach** stosuje się podobny sposób indeksacji.

Tablice w języku C++
→ s. 210

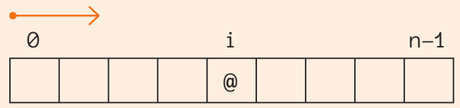


Rys. 7.3. Interpretacja graficzna łańcucha znaków

Poprawność adresu e-mail możemy sprawdzić w czterech etapach. Opisuje je schemat z rysunku 7.4.

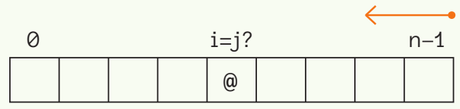
Etap 1

Szukamy w napisie (zaczynając od lewej strony) znaku @. Jeśli go nie znajdziemy albo będzie on pierwszym lub drugim znakiem napisu, sprawdzanie kończy się niepowodzeniem. W przeciwnym razie zapamiętujemy w zmiennej `i` pozycję znaku @ w napisie. Przechodzimy do etapu 2.



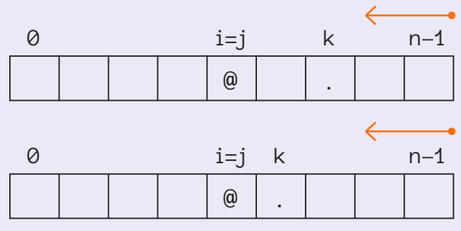
Etap 2

Sprawdzamy, czy wewnątrz napisu nie ma innego znaku @. Szukamy pierwszego wystąpienia znaku @, począwszy od prawego końca napisu. Zapamiętujemy jego pozycję w zmiennej `j` i sprawdzamy, czy `i` jest równe `j`. Jeśli wartości są różne, sprawdzanie kończy się niepowodzeniem. W przeciwnym razie przechodzimy do etapu 3.



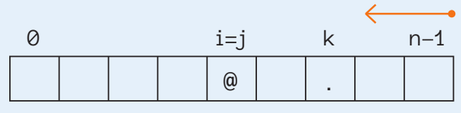
Etap 3

Sprawdzamy, czy znak kropki występuje w napisie na pozycji trzeciej lub czwartej od końca. Jeśli nie, sprawdzanie kończy się niepowodzeniem. Jeśli tak, zapamiętujemy pozycję kropki, np. w zmiennej `k`, i przechodzimy do etapu 4.



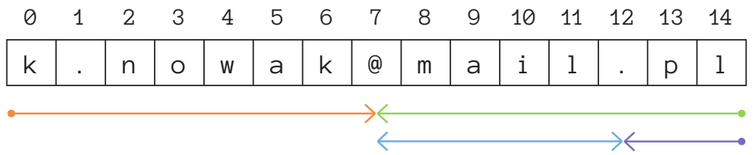
Etap 4

Sprawdzamy, czy znak @ znajduje się po lewej stronie znaku kropki w odstępie co najmniej jednego znaku. Jeśli nie, sprawdzanie kończy się niepowodzeniem. Jeśli tak, podany adres uznajemy za poprawny.



Rys. 7.4. Etapy sprawdzania poprawności podanego adresu e-mail

Przykład realizacji algorytmu pokazuje rysunek 7.5. Kolory strzałek są zgodne z kolorami tła poszczególnych etapów z rysunku 7.4 ze s. 89.



Rys. 7.5. Sprawdzanie poprawności adresu

Algorytm sprawdzający poprawność adresu e-mail wygodnie będzie zapisać w kodzie źródłowym w formie funkcji:

Kod źródłowy funkcji CzyPoprawnyAdres w programie *Sprawdzenie adresu*

```
1.  bool CzyPoprawnyAdres(string adres)
2.  {
3.      int dl = adres.length();
4.
5.      int i = 0; // etap 1
6.      while (i<dl-1 && adres[i]!='@') i++;
7.      if (i<2 || adres[i]!='@') return false;
8.
9.      int j = dl-1; // etap 2
10.     while (adres[j]!='@') j--;
11.     if (i!=j) return false;
12.
13.     int k = dl-1; // etap 3
14.     while (k>0 && adres[k]!='.') k--;
15.     if (adres[k]!='.'||!(k==dl-3||k==dl-4)) return false;
16.
17.     if (k-i<=1) return false; // etap 4
18.
19.     return true;
20. }
```

Warto wiedzieć

Łańcuchy znaków w języku C++ nie są zwykłymi tablicami znaków, czyli tablicami zmiennych typu `char`. Projektując bibliotekę `string`, zastosowano podejście właściwe tzw. programowaniu obiektowemu. W praktyce oznacza to tyle, że rozmiar łańcucha (liczba znaków) może się zmieniać, a miejsce w pamięci komputera dla zmiennej jest przydzielane dynamicznie w zależności od długości łańcucha, który ma być zapamiętany.

Funkcja `CzyPoprawnyAdres` zwraca wartość logiczną – `true`, gdy podany napis jest adresem e-mail, lub `false` – w przeciwnym przypadku. W linii 3. deklarujemy zmienną `dl`, która przechowa długość podanego napisu. Wykorzystaliśmy do tego **funkcję** `length` zdefiniowaną w bibliotece `string`. Funkcja ta zwraca długość napisu. Zapisujemy ją po nazwie zmiennej i kropce. Kolejne etapy sprawdzania poprawności adresu zaznaczono w komentarzach.

Funkcja `length`

Warto wiedzieć

W bibliotece `string` zdefiniowana jest struktura tzw. klasy (ang. *class*) wraz z funkcjami wbudowanymi. Funkcje te, zwane metodami, pozwalają wykonywać operacje na obiektach danej klasy.

W etapie 1. (linie 5.–7.) zmienna `i` określa pozycję pierwszego znalezionego w przeszukiwaniu od lewej znaku @, o ile taki znak w napisie w ogóle jest. Aby uzyskać dostęp do znaku w napisie, stosuje się zapis z parą nawiasów kwadratowych.

Etap 2. zapisany jest w liniach 9.–11. Jeżeli `i` różni się od `j` (linia 11.), to w napisie występują co najmniej dwa znaki @. Linie 13.–15. dotyczą etapu 3. Zapis `!(k==dl-3||k==dl-4)` w linii 15. oznacza **formułę logiczną** „Nieprawda, że zachodzi jeden z przypadków `k==dl-3` lub `k==dl-4`”.

Oto fragment kodu źródłowego z wywołaniem funkcji:

Logika matematyczna i operatory logiczne
→ s. 198–200

Fragment kodu źródłowego funkcji `main` programu *Sprawdzenie adresu*

```
1.  const string INF_OK =
2.      "Haslo do serwisu wyslalismy na adres: ";
3.  const string INF_ROZNE_ADRESY =
4.      "Podane adresy e-mail sa rozne!";
5.  const string INF_BLAD_ADRESU =
6.      "Adres jest niepoprawny!";
7.  if (CzyPoprawnyAdres(adres_1))
8.  {
9.      cout << "Powtorz adres e-mail: "; cin >> adres_2;
10.     cout << endl;
11.     if (adres_1 == adres_2)
12.         cout << INF_OK + adres_1 << endl;
13.     else
14.         cout << INF_ROZNE_ADRESY << endl;
15. }
16. else
17.     cout << endl << INF_BLAD_ADRESU << endl;
```

Warto wiedzieć

Zapis w linii 7. może być również następujący: `if (CzyPoprawnyAdres (adres_1) == 1)`

Funkcja `CzyPoprawnyAdres` jest **typu logicznego**. Jest wywołana w miejscu wyrażenia logicznego instrukcji warunkowej (linia 7.).

Typy danych w języku C++
→ s. 209–210

Ćwiczenie 5

Zapisz kod źródłowy programu *Sprawdzenie adresu*. Skompiluj kod i sprawdź działanie programu dla różnych danych.

A to ciekawe

Jak odróżnić człowieka od maszyny?

W wielu serwisach internetowych stosuje się rozwiązanie CAPTCHA, by upewnić się, że formularz wypełnia człowiek, a nie robot. Test polega zwykle na przepisaniu słowa z obrazka. Projekt reCAPTCHA wykorzystuje tę czynność do digitalizacji zasobów – użytkownikom wyświetla się fragmenty tekstu nieczytelne dla oprogramowania OCR (zamieniającego skan na tekst). Jeśli wystarczająco dużo osób tak samo odczyta dane słowo, można je uznać za poprawnie rozpoznane.

Poprawność adresu e-mail – sprawdzenie struktury adresu z wykorzystaniem funkcji z biblioteki `string`

Obiekty, klasy i metody w języku C++
→ s. 216

Biblioteka `string` poza **funkcją** (**metodą**) `length` oferuje kilkanaście innych funkcji, nazywanych również metodami, które mogą usprawnić pisanie kodu źródłowego oraz znacznie go skrócić.

Zapiszemy teraz opracowany wcześniej algorytm z wykorzystaniem funkcji `find` oraz `rfind`. **Funkcja `find`** zwraca indeks pierwszego wystąpienia poszukiwanej frazy. **Funkcja `rfind`** zwraca indeks ostatniego wystąpienia poszukiwanej frazy, czyli szuka frazy, zaczynając od końca (od prawej strony) analizowanego łańcucha znaków.

Funkcja `find`
Funkcja `rfind`

Oto zapis funkcji `CzyPoprawnyAdres` z użyciem powyższych funkcji:

Kod źródłowy funkcji `CzyPoprawnyAdres` wykorzystujący funkcje z biblioteki `string` w programie *Sprawdzenie adresu 2*

```
1.  bool CzyPoprawnyAdres(string adres)
2.  {
3.      int dl = adres.length();
4.
5.      if (adres.find("@")<2 || adres.find("@")>dl)
6.          return false; // etap 1
7.      int i = adres.find("@");
8.
9.      int j = adres.rfind("@"); // etap 2
10.     if (i!=j) return false;
11.
12.     if (adres.rfind(".")>dl) return false; // etap 3
13.     int k = adres.rfind(".");
14.     if (!(k==dl-3 || k==dl-4)) return false;
15.
16.     if (k-i<=1) return false; // etap 4
17.
18.     return true;
19. }
```

Warto wiedzieć

Umieszczanie nazwy metody po kropce to cecha programowania obiektowego.

Funkcję `find` zastosowaliśmy w linii 7. do odszukania indeksu pierwszego wystąpienia znaku @. Warunki `adres.find("@")>dl` z linii 5. oraz `adres.rfind(".")>dl` z linii 12. odnoszą się do sytuacji, w której funkcje `find` i `rfind` nie znajdą poszukiwanych znaków. Wówczas funkcje te zwracają pewną ustaloną wartość, większą niż długość napisu. W obu przypadkach użyto zapisu z nazwą funkcji po kropce.

Zapamiętaj

W języku C++ rozróżniamy zmienne znakowe typu `char` i zmienne łańcuchowe typu `string`. Pierwsze służą do zapamiętania pojedynczego znaku (np. litery, cyfry), a drugie pozwalają na przechowanie całego łańcucha znaków.

7.3. Lista zakupów. Usuwanie duplikatów w tekście

Stworzymy teraz program, którego zadaniem będzie usunąć zbędne powtórzenia produktów z listy zakupów. Dla ułatwienia przyjmijmy, że napisy będą podawane w porządku alfabetycznym w oddzielnych wierszach, bez żadnych znaków interpunkcyjnych.

Warto wiedzieć

Jeśli dane wejściowe są uporządkowane, znacznie prościej jest rozwiązać wiele zadań. Na przykład porządek alfabetyczny w indeksie książki lub w słowniku pozwala szybko odnaleźć hasło.

Specyfikacja

Dane: lista słów wpisywanych alfabetycznie, jedno pod drugim, zakończona trzema gwiazdkami (***).

Wynik: lista słów po usunięciu powtórzeń występujących w sąsiednich wierszach.

Sposób rozwiązywania tego problemu można sformułować następująco: dopóki na wejściu pojawiają się nowe wyrazy, należy porównywać ostatnio wczytany wyraz z poprzednim i w razie potrzeby usuwać powtórzenia, a nieusunięte wyrazy dołączać do listy wynikowej.

Precyzyjniej ten algorytm można zapisać jako listę kroków:

1. Wczytaj pierwszy wyraz. Zapamiętaj go jako pierwszy na liście wynikowej i wczytaj kolejny wyraz.
2. Dopóki ostatnio wczytany wyraz nie jest ciągiem znaków ***, wykonuj kolejno kroki 3 i 4.
3. Jeśli ten wyraz jest różny od poprzedniego, to dodaj go jako kolejny na listę wynikową.
4. Wczytaj kolejny wyraz.

Przeanalizujemy teraz działanie algorytmu dla listy słów. Wygodnie będzie to zrobić w tabeli:

Napis	Czy dublet?	Czy koniec listy?	Lista wynikowa
chleb	–	nie	chleb
cytryna	nie	nie	chleb cytryna
mleko	nie	nie	chleb cytryna mleko
ser	nie	nie	chleb cytryna mleko ser
ser	tak	nie	chleb cytryna mleko ser
szynka	nie	nie	chleb cytryna mleko ser szynka
***	nie	tak	chleb cytryna mleko ser szynka

Tabela 7.3. Analiza działania algorytmu usuwającego powtórzenia

Tabelę 7.3 uzupełniamy od lewej do prawej, według kolejnych kroków algorytmu. Kolorem zaznaczono etap, w którym nowy wyraz był powtórzeniem.

Warto wiedzieć

Wyszukiwanie duplikatów jest operacją szczególnie istotną w pracy analityków danych. Niektóre aplikacje wspomagające ich pracę, np. arkusze kalkulacyjne, zawierają wbudowane funkcje pozwalające usuwać zduplikowane wartości.

Ćwiczenie 6

Przeanalizuj działanie algorytmu dla listy zakupów składającej się z ośmiu wyrazów, wśród których są dwa powtórzenia (jedno z nich jest na końcu).

Oto fragment kodu źródłowego programu *Usuwanie powtórzeń*:

Fragment kodu źródłowego programu *Usuwanie powtórzeń*

```
1.  const int N = 20;
2.
3.  int main()
4.  {
5.      string wynik[N];
6.      string nowy, stary;
7.      int i = 0;
8.
9.      cin >> nowy;
10.     wynik[0] = nowy;
11.
12.     stary = nowy;
13.     cin >> nowy;
14.     while (nowy != "***" && i < N-1)
15.     {
16.         if (nowy != stary)
17.         {
18.             i++;
19.             wynik[i] = nowy;
20.         }
21.         stary = nowy;
22.         cin >> nowy;
23.     }
24.
25.     for (int j=0; j<=i; j++)
26.         cout << wynik[j] << endl;
27.
28.     return 0;
29. }
```

Dobra rada

Łańcuch *** w programie *Usuwanie powtórzeń* pełni funkcję wartownika. Używaj wartowników wtedy, kiedy chcesz oznaczyć warunki brzegowe realizacji programu, np. nie chcesz wyjść poza listę.

W linii 1. zadeklarowano stałą N i przypisano jej wartość 20 – maksymalną liczbę słów na liście. W linii 5. zadeklarowano tablicę N łańcuchów, w której będzie przechowywana lista produktów. W linii 6. zadeklarowano łańcuchy znaków, które będą służyły do porównania dwóch podanych po sobie napisów. Zmienna i deklarowana w linii 7. będzie licznikiem. W liniach 9.–10. wczytujemy pierwszy, a w linii 13. drugi napis.

W pętli (linie 14.–23.) sprawdza się, czy podany ostatnio napis jest różny od *** oraz czy licznik napisów nie przekracza wartości stałej N. Jeśli oba warunki są spełnione, kod w liniach 16.–20. zapobiega wstawianiu powtórzonej nazwy do tablicy. W liniach 25.–26. wypisywana jest lista produktów.

Ćwiczenie 7

Zapisz kod źródłowy programu *Usuwanie powtórzeń*. Skompiluj kod i sprawdź działanie programu dla różnych danych wejściowych.

7.4. Wyszukiwanie wzorca w tekście

Wyszukiwanie wzorca w tekście, nazywane też dopasowywaniem wzorca (ang. *pattern matching*), to szukanie wystąpienia danego łańcucha znaków w tekście. Celem jest określenie, w którym miejscu w tekście znajduje się wzorzec, o ile w ogóle w nim występuje.

Wyszukiwanie wzorca w tekście

Wzorca szukamy np. wtedy, gdy chcemy znaleźć konkretne pojęcie w spisie treści lub indeksie podręcznika. Wówczas nasze oko stara się dostrzec dany ciąg znaków.

Wykreślanka

Na rysunku 7.6 znajduje się przykład zadania nazywanego wykreślanką. Polega ono na odnajdowaniu ukrytych wyrazów – w tym wypadku imion. Mogą one być zapisane poziomo, pionowo lub na ukos, także wspak. Wyszukiwanie imion pośród porozrzucanych na planszy liter jest przykładem szukania wzorca.

Rys. 7.6. Wykreślanka z zaznaczonymi imionami

Warto wiedzieć

Od rozwiązania problemu wyszukiwania wzorca w tekście można też oczekiwać, że wskaże ono wszystkie wystąpienia wzorca, jeśli jest ich więcej.

Ćwiczenie 8

a. W pliku, który otrzymasz od nauczyciela (np. *wykreslanka.xlsx*), znajduje się wykreślanka. Znajdź jak najwięcej imion, które w niej ukryto.

b. Przygotuj własną wykreślankę, w której umieścisz jak najwięcej imion osób z twojej klasy.

Program *Szukaj wzorca*

Napiszemy teraz program, który będzie poszukiwał podanego wzorca w tekście. Oto specyfikacja problemu:

Specyfikacja

Dane: tekst zapisany wielkimi literami oraz wzorzec – łańcuch zapisany wielkimi literami.

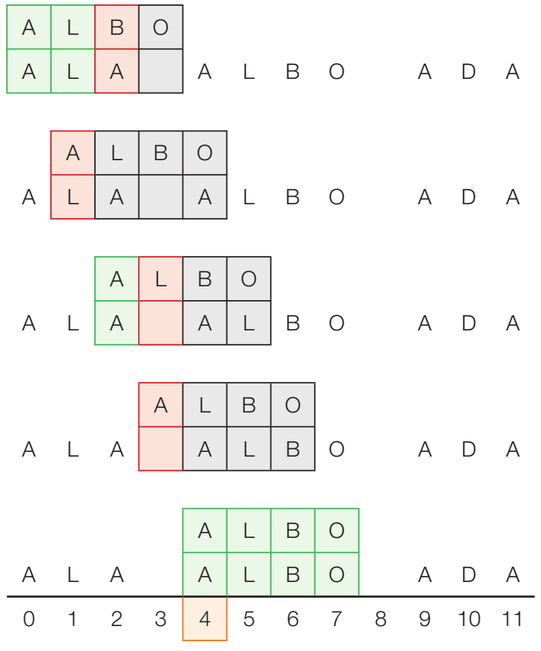
Wynik: pozycja pierwszego wystąpienia wzorca w tekście, rozumiana jako liczba znaków tekstu poprzedzających wystąpienie wzorca, lub liczba –1, jeśli wzorca nie ma w tekście.

Algorytm wyszukiwania wzorca w tekście

Przedstawimy działanie prostego **algorytmu wyszukiwania wzorca w tekście**, tzw. algorytmu naiwnego. Wzorzec będzie przesuwany stopniowo wzdłuż tekstu o jeden znak. Za każdym razem wzorzec będzie ustalał początek fragmentu tekstu, który będzie z nim porównywany znak po znaku (od pierwszego znaku wzorca).

Gdy analizowany znak jest zgodny ze znakiem wzorca, porównuje się zgodność kolejnego znaku fragmentu tekstu z kolejnym znakiem wzorca. Jeśli analizowany fragment tekstu jest identyczny ze wzorcem, to wyszukiwanie się kończy i jako odpowiedź zwracana jest pozycja dopasowania, czyli liczba znaków tekstu poprzedzających wystąpienie wzorca.

W tekście ALA ALBO ADA poszukamy wzorca ALBO. Rysunek 7.7 pokazuje działanie algorytmu. Zwracana jest wartość 4.



Rys. 7.7. Naiwne wyszukiwanie wzorca w tekście

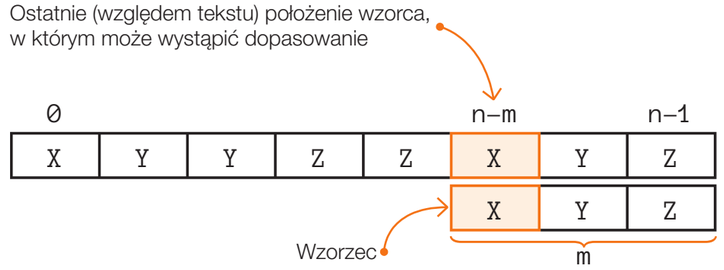
Ćwiczenie 9

a. Zapisz przebieg działania algorytmu poszukiwania w tekście ZA SZCZYTEM ZASZCZYT wzorca ZASZCZYT. Zapisz go w podobny sposób jak na rysunku 7.7 ze s. 96.

b. Ile prób dopasowania wzorca trzeba by było wykonać w powyższym ćwiczeniu, gdyby wyszukiwanie zacząć od końca?

Liczbę znaków tekstu do przeszukiwania oznaczmy literą n, a liczbę znaków wzorca – literą m. Aby stwierdzić, czy wzorzec występuje w tekście, dla każdego znaku tekstu, aż do znaku na pozycji o indeksie $n - m$, należy sprawdzać zgodność kolejnych znaków tekstu ze znakami tworzącymi wzorzec.

Dlaczego sprawdzamy do znaku o indeksie $n - m$? Wyjaśnia to rysunek 7.8. Jeśli tekst ma długość $n = 9$, a wzorzec $m = 3$, to ostatni indeks ma wartość 6 (znaki indeksujemy od 0).



Rys. 7.8. Wyznaczenie indeksu ostatniego znaku tekstu, z którym porównujemy wzorzec

Oto lista kroków naiwnego algorytmu wyszukiwania wzorca:

1. Wczytaj tekst i wzorzec.
2. Oznacz długość tekstu literą n, a długość wzorca – literą m.
3. Dla każdej liczby z zakresu od 0 do n - m (nazywanej bieżącą pozycją znaku) wykonuj kolejno kroki 4 i 5.
   4. Wybierz fragment tekstu długości m, począwszy od bieżącej pozycji znaku, i porównaj go ze wzorcem.
   5. Jeśli fragment i wzorzec są identyczne, to zwróć jako odpowiedź bieżącą pozycję znaku i zakończ algorytm.
6. Zwróć wartość -1.

W powyższym algorytmie licznik pętli, w momencie znalezienia wzorca w tekście, wskazuje jednocześnie indeks (pozycję) pierwszego wystąpienia wzorca w tekście. Jest to możliwe, ponieważ wartości licznika zaczynają się od 0, czyli tej wartości, od której indeksuje się znaki w łańcuchach.

Warto wiedzieć

W teorii algorytmów metodę rozwiązywania problemu, która wymaga sprawdzenia wszystkich przypadków, nazywa się metodą wyczerpującego wyszukiwania (ang. *exhaustive search*) albo metodą siłową (ang. *brute-force method*). O algorytmie mówi się wówczas, że jest naiwny.

Oto kod źródłowy programu *Szukaj wzorca*:

Kod źródłowy programu *Szukaj wzorca*

```
1.  #include <iostream>
2.  #include <string>
3.
4.  using namespace std;
5.
6.  const string TEKST = "ALA ALBO ADA";
7.
8.  int Porownaj(string wzorzec)
9.  {
10.     int n = TEKST.length();
11.     int m = wzorzec.length();
12.
13.     for (int poz=0; poz<=n-m; poz++)
14.     {
15.         int j = 0;
16.         while (j<m && TEKST[poz+j] == wzorzec[j])
17.             j = j + 1;
18.
19.         if (j == m)
20.             return poz;
21.     }
22.     return -1;
23. }
24.
25. int main()
26. {
27.     cout << "Tekst: " << TEKST << endl;
28.     string wzorzec;
29.     cout << "\nPodaj wzorzec: "; cin >> wzorzec;
30.     cout << "Pozycja: " << Porownaj(wzorzec) << endl;
31.     return 0;
32. }
```

Stała globalna

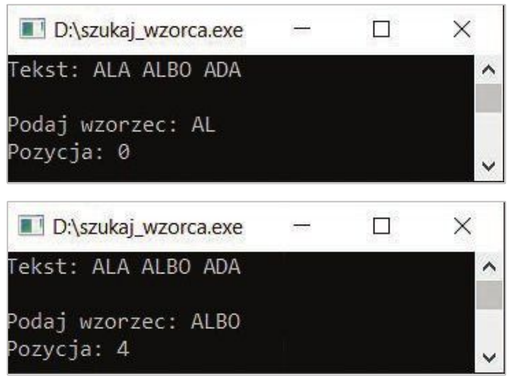
W linii 6. zastosowano **stałą globalną** o nazwie TEKST. Jest to stała, do której dostęp jest możliwy z dowolnego miejsca kodu programu, również z funkcji Porownaj, którą definiujemy w liniach 8.–23. Stała globalna musi być zdefiniowana przed wystąpieniem funkcji głównej i każdej innej funkcji lub zmiennej, które mogłyby ją wykorzystywać. W liniach 10. i 11. zadeklarowano zmienne n i m, które będą pamiętały długość odpowiednio tekstu i wzorca.

Pętle zagnieżdżone

Wewnątrz funkcji Porownaj zastosowano **pętle zagnieżdżone** (ang. *nested loops*), czyli pętlę umieszczoną wewnątrz innej pętli. Zewnętrzna pętla (for), zapisana w liniach 13.–21., służy do przesuwania wzorca po kolejnych znakach tekstu, a wewnętrzna (while), w liniach 16.–17., przechodzi przez kolejne znaki wzorca (krok 4 z listy kroków).

Pętla while oraz instrukcja warunkowa w linii 19. sprawdzają, czy w napisie występuje wzorzec – wartość zmiennej j będzie równa długości wzorca. Wówczas funkcja kończy działanie i zwraca wartość zmiennej poz (linia 20.).

Rysunek 7.9 przedstawia przykładowe wywołania programu.



Rys. 7.9. Wywołania programu *Szukaj wzorca* dla dwóch różnych wzorców

Dobra rada

Pamiętaj, że wielkie i małe litery mają różne kody ASCII. Jeśli tekst wyjściowy jest zapisany wielkimi literami, wówczas wzorzec należy zapisać również wielkimi literami.

Ćwiczenie 10

a. W pliku otrzymanym od nauczyciela (np. *wzorzec_w_tekscie.cpp*) znajdują się fragment programu *Szukaj wzorca* z tekstem, w którym należy wyszukać podany wzorzec. Uzupełnij brakujące fragmenty kodu źródłowego, włącznie z funkcją Porownaj. Skompiluj kod i uruchom program dla wzorców: AGA oraz ALBO.

b. Sprawdź działanie programu dla własnego tekstu i wzorca.

Naiwnego algorytmu wyszukiwania wzorca nie można by było z powodzeniem zaimplementować np. w czytniku PDF. Okazuje się, że dla dłuższych tekstów jest on bardzo powolny. Znane są bardzo efektywne algorytmy dopasowywania wzorca w tekście i to z nich korzysta się w praktyce przy przetwarzaniu tekstów.

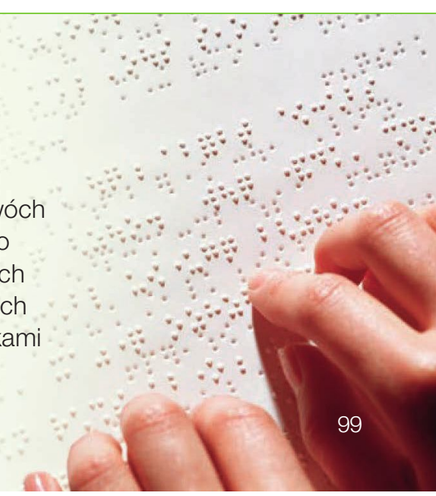
Warto wiedzieć

Efektywna metoda wyszukania wzorca w tekście jest zaimplementowana w języku C++ jako funkcja (metoda) find. Używaliśmy jej w programie sprawdzającym poprawność adresu e-mail.

A to ciekawe

Alfabet Braille'a, czyli szukanie wzorca

Alfabet Braille'a to system zapisu informacji z wykorzystaniem dwóch stanów: punkt wypukły i niewypukły. Znaki przedstawione są jako sześciopunkty, czyli kombinacje punktów umieszczonych w dwóch kolumnach, po trzy punkty w każdej. Czytanie książek brajlowskich oznacza w praktyce wielokrotne dopasowywanie wzorca opuszkami palców w celu odczytania kolejnych znaków tekstu.



Zadania

* **1** Napisz program, który wczyta z klawiatury nazwę miejscowości (bez spacji) i wypisze co drugą jej literę. Litery powinny być rozdzielone spacjami.

* **2** Napisz program, który wczyta z klawiatury 10 nazw kolorów i wypisze najdłuższą z nich. Jeśli takich nazw jest więcej, to program powinien wyświetlić nazwę podaną najpóźniej.

* **3** Napisz program, który wczyta z klawiatury łańcuch znaków (składający się z kombinacji cyfr oraz małych i wielkich liter) i wypisze tylko wielkie litery, które pojawiły się w tekście.

* **4** Napisz program, który dla podanego wyrazu wyświetli jego zapis wspak (od ostatniej do pierwszej litery).

** **5** Napisz program, który będzie zliczał liczbę samogłosek wewnątrz łańcucha znaków.

** **6** Napisz program, który wyświetli litery występujące jednocześnie w dwóch słowach wpisanych z klawiatury. Na przykład dla słów ARKA i BARAK program powinien wypisać litery A, K i R.

** **7** Wyrazy różniące się dowolną jedną literą nazywa się metagramami, np.: kasa i kara. Napisz program sprawdzający, czy podana para wyrazów to metagramy.

** **8** Napisz program, który z łańcucha znaków będzie usuwał wszystkie samogłoski i zastępował je znakiem podkreślenia (_).

** **9** Napisz funkcję o nagłówku `int ile(string wzorzec)`, która zwróci liczbę całkowitą oznaczającą liczbę wystąpień wzorca w tekście. Przyjmij, że tekst jest zapisany w stałej globalnej o nazwie TEKST.

*** **10** Napisz program, który po podaniu daty urodzenia (dzień i miesiąc) wypisze odpowiadający jej znak zodiaku. Na przykład dla danych `21.07` wynikiem będzie napis `rak`.

*** **11** Napisz funkcję, która kompresuje tekst prostą metodą opartą na zliczaniu powtarzających się liter (jedna litera może się powtarzać maksymalnie 9 razy). Na przykład wynikiem dla łańcucha `aaabbcaa` będzie `a3b2c1a2`. Jeśli wynik miałby być łańcuchem dłuższym niż ten dany na wejściu, to funkcja powinna zwracać pierwotny łańcuch znaków.

*** **12** Napisz program, który poda liczbę wystąpień każdej litery alfabetu łacińskiego w tekście wprowadzonym do stałej globalnej o nazwie TEKST. Program nie powinien rozróżniać wielkości liter.

8 Szyfrowanie wiadomości

Szyfrów używano od wielu stuleci do utajniania korespondencji dworskiej oraz w wojsku do przesyłania pomiędzy oddziałami wiadomości, które nie powinny być czytane przez wroga. W XIX w., kiedy zaczęła się rozwijać kolej żelazna i upowszechniały się sposoby komunikacji na odległość (telegraf i teleks), metodami szyfrowania zainteresowali się również ludzie biznesu. W II połowie XX w., wraz z rozwojem telekomunikacji i globalnych sieci komputerowych, praktyka szyfrowania informacji stawała się coraz bardziej powszechna. Dziś dotyczy każdego z nas, gdyż stanowi podstawę bezpieczeństwa m.in. transakcji bankowych i płatności elektronicznych.

Ważne w tym temacie

- Podstawowe pojęcia związane z kryptografią
- Zastosowania kryptografii
- Szyfr Cezara
- Programy szyfrujące informacje tekstowe szyfrem Cezara
- Metody łamania prostych szyfrów

8.1. Podstawowe pojęcia kryptografii

Szyfrowaniem nazywamy przekształcanie informacji do postaci trudnej do odczytania bez dodatkowej wiedzy. Celem szyfrowania jest zapewnienie tajności (poufności) informacji. Zagadnieniami związanymi z szyfrowaniem zajmuje się nauka zwana **kryptografią**.

Szyfrowanie

Kryptografia

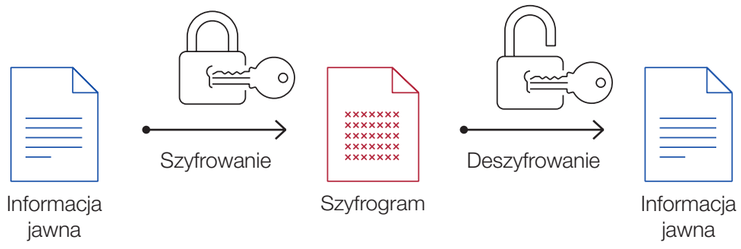
W trakcie szyfrowania informację jawną, zwaną **tekstem jawnym** (ang. *plaintext*), przekształca się w **szyfrogram** (ang. *ciphertext*), czyli tekst zaszyfrowany. W tym procesie stosuje się odpowiedni algorytm oraz **klucz szyfrowania** (ang. *cipher key*). Jest to pewna wartość, niezależna od szyfrowanej informacji. Klucz szyfrowania jest potrzebny do odczytania informacji jawnej. Przekształcanie zaszyfrowanych danych w informację pierwotną to **deszyfrowanie** (ang. *decryption*) (rys. 8.1). Warunkiem zachowania tajności informacji jest utrzymanie w tajemnicy klucza szyfrowania.

Tekst jawny

Szyfrogram

Klucz szyfrowania

Deszyfrowanie



Rys. 8.1. Schemat szyfrowania i deszyfrowania informacji

8.2. Szyfr Cezara

Szyfr podstawieniowy

Jeszcze w czasach starożytnych wymyślono **szyfr podstawieniowy**. Polega on na zamianie znaków tekstu jawnego na inne znaki alfabetu według określonego algorytmu. W szyfrogramie mogą więc pojawić się znaki alfabetu, które nie zostały użyte w tekście jawnym.

Szyfr Cezara

Szyfr Cezara

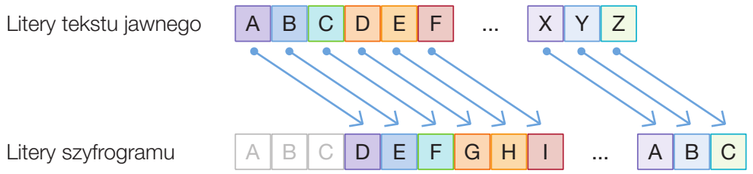
Jednym z najbardziej znanych klasycznych szyfrów podstawieniowych jest **szyfr Cezara**. Był on stosowany przez Juliusza Cezara – żołnierza, polityka i dyktatora Republiki Rzymskiej w I w. p.n.e.

Tekst jawny
→ s. 101

Metoda szyfrowania, której używał Cezar, polega na zastąpieniu każdej litery **tekstu jawnego** literą oddaloną od niej o stałą liczbę pozycji w alfabecie. Kierunek zmian musi być zachowany. Przesunięcie odbywa się cyklicznie, tzn. po ostatniej literze alfabetu następuje pierwsza.

Klucz szyfrowania
→ s. 101

Kluczem dla szyfru Cezara jest liczba miejsc, o jaką przesuwane są litery w alfabecie. Juliusz Cezar stosował klucz o wartości 3, tzn. literę A zastępował literą D, literę B – literą E itd. Pokazuje to rysunek 8.2.



Rys. 8.2. Szyfrowanie liter szyfrem Cezara z kluczem 3

Warto wiedzieć

Juliusz Cezar, jak każdy wykształcony człowiek w tamtych czasach, regularnie pisywał listy do przyjaciół oraz polityków. Aby utajnić poufną korespondencję, szyfrował jej treść.

Warto wiedzieć

Alfabet łaciński w czasach Juliusza Cezara składał się z 23 liter. Nie używano wówczas liter J, U i W.

Ćwiczenie 1

Wykorzystując szyfr Cezara i alfabet łaciński:

a. zaszyfruj wyraz MEDYCYNA z kluczem 25.

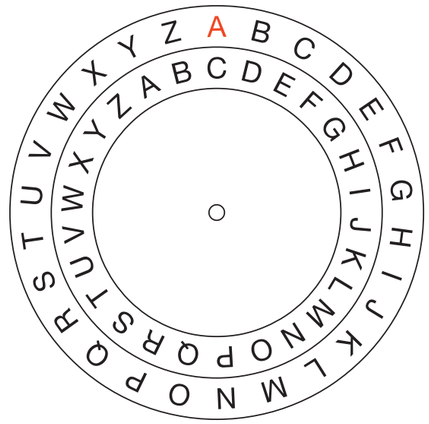
b. odszyfruj szyfrogram LWUQVQSIVI z kluczem 8.

A to ciekawe

Anagram jako szyfrogram

Niektórzy uczeni nie chcieli ogłaszać wprost treści swoich odkryć, dlatego stosowali anagramy, czyli zmieniali kolejność liter w tekście. Istotę odkrycia, zazwyczaj w formie łacińskiego anagramu, ogłaszali drukiem. W 1610 r. Galileusz opublikował w ten sposób swoje odkrycie dotyczące rzekomych księżyców Saturna: SMAISMRMILMEPOETALEVMIBVNENVGTTAVIRAS. Oznacza to: ALTISSIMUM PLANETAM TERGEMINUM OBSERVARI (dosł. widziałem najwyższą z planet potrójną). Liter U i V wtedy nie rozróżniano.

Przesunięcie liter w szyfrze Cezara można również przedstawić w postaci dwóch tarcz, z których wewnętrzna się obraca. Po obróceniu wewnętrznej tarczy otrzymujemy przyporządkowanie: litera tekstu jawnego i odpowiadająca jej litera tekstu zaszyfrowanego. Na rysunku 8.3 przedstawiono przesunięcie równe 2, tzn. literze A odpowiada litera C, literze B – litera D itd.



Rys. 8.3. Szyfrowanie liter szyfrem Cezara za pomocą obracających się tarcz

Warto wiedzieć

Wykorzystanie do szyfrowania obracających się tarcz z literami alfabetu zaproponował w XV w. Leon Battista Alberti. Wewnętrzna tarcza zawierała litery zapisane w losowej kolejności. Oprócz tego założył, że po zaszyfrowaniu kilku wyrazów tekstu jawnego tarcze należy obrócić, aby otrzymać zupełnie nowe podstawienie. Tak powstał tzw. szyfr polialfabetyczny. Podstawienie Cezara zapisane na dwóch tarczach jest przykładem szyfru monoalfabetycznego.

Ćwiczenie 2

a. Przygotuj fizyczny model obracających się tarcz pozwalających szyfrować tekst metodą Cezara, podobny jak na rysunku 8.3. Użyj pinezki, aby umocować przyrząd np. na tablicy korkowej. Wskazówka: Możesz wykonać projekt tarcz w programie graficznym, a następnie wydrukować je i wyciąć.

b. Korzystając z utworzonego modelu, zaszyfruj wiadomość ALARM z kluczem 4 oraz odszyfruj szyfrogram TEDBY zaszyfrowany z kluczem 10.

Szyfr Cezara – program szyfrujący pojedyncze słowo

Napiszemy program, który pozwoli zaszyfrować napis złożony wyłącznie z wielkich liter alfabetu łacińskiego.

Oto specyfikacja problemu:

Specyfikacja

Dane: tekst jawny składający się wyłącznie z wielkich liter alfabetu łacińskiego; klucz szyfrowania w postaci liczby całkowitej z zakresu od 1 do 25.

Wynik: tekst zaszyfrowany szyfrem Cezara z wykorzystaniem podanego klucza.

Warto wiedzieć

Informację zaszyfrowaną nazywa się czasami niepoprawnie informacją zakodowaną. Słowa „szyfr” i „kod” nie są synonimami. Kodu używamy, gdy nie można bezpośrednio przekazać informacji, np. ze względu na odległość. Przykładem kodu jest zapis za pomocą liter na papierze albo binarnie w komputerze. Szyfr to przekształcenie zapisu jawnego w niejawny, np. przez przestawienie kolejności liter lub bitów.

Lista kroków algorytmu może wyglądać następująco:

1. Wczytaj tekst jawny.
2. Wczytaj klucz szyfrowania i oznacz tę liczbę jako klucz.
3. Dla każdej litery tekstu jawnego wykonuj kolejno kroki 4 i 5.
   4. Oznacz kod ASCII tej litery jako kod.
   5. Jeżeli kod + klucz jest liczbą niewiększą od kodu ASCII litery Z, to dołącz do szyfrogramu literę o kodzie ASCII równym kod + klucz. W przeciwnym razie dołącz do szyfrogramu literę o kodzie ASCII wynoszącym kod + klucz – 26.

Typ char
→ s. 83

Kody ASCII kolejnych liter szyfrogramu zapamiętamy w zmiennej pomocniczej kod **typu** `char`. Aby wyznaczyć kod ASCII litery szyfrogramu, do kodu litery tekstu jawnego dodajemy przesunięcie, czyli wartość klucza. Jeśli otrzymana wartość będzie większa od kodu litery Z (liczby 90), musimy odjąć 26 (liczbę liter alfabetu), aby otrzymać kod litery z początku alfabetu.

Na przykład podczas szyfrowania litery Y w słowie INFORMATYKA z kluczem 3 otrzymamy literę B. Kod ASCII litery Y to 89. Po dodaniu 3 uzyskujemy 92, a więc wartość większą od kodu litery Z. Odejmujemy 26 i otrzymujemy 66, czyli kod litery B.

Oto kod źródłowy funkcji głównej programu *Szyfr Cezara*, realizującego omówiony algorytm:

Kod źródłowy funkcji głównej programu *Szyfr Cezara*

```
1.  int main()
2.  {
3.      string jawny, szyfrogram = "";
4.      int dl, klucz;
5.      char kod;
6.
7.      cout << "Podaj slowo: "; cin >> jawny;
8.      dl = jawny.length();
9.
10.     cout << "Podaj klucz: "; cin >> klucz;
11.
12.     for (int i=0; i<dl; i++)
13.     {
14.         kod = jawny[i] + klucz;
15.         if (kod > 'Z')
16.             kod = kod - 26;
17.         szyfrogram = szyfrogram + kod;
18.     }
19.
20.     cout << szyfrogram << endl;
21.
22.     return 0;
23. }
```

Dobra rada

Pamiętaj, że w języku C++ znaki (wartości typu `char`) są reprezentowane przez wartości liczbowe (kody ASCII). Możesz wykonywać na nich działania arytmetyczne w celu obliczenia kodu ASCII właściwego znaku.

W linii 3. zmiennej `szyfrogram` przypisujemy jako początkową wartość łańcuch pusty, czyli łańcuch bez znaków (o długości 0). Do tego łańcucha będziemy dołączać kolejne znaki szyfrogramu. W linii 8., korzystając z **funkcji** (**metody**) `length`, obliczamy długość podanego napisu. W pętli z linii 12.–18. obliczamy kod ASCII pojedynczego znaku szyfrogramu (jednej litery) i zapisujemy go w zmiennej pomocniczej `kod` (linia 14.). Jeśli wartość zmiennej `kod` jest większa od kodu litery Z, to odejmujemy od niej 26 (linia 16.). Rysunek 8.4 przedstawia przykłady wywołań programu *Szyfr Cezara*.

Funkcja `length`
→ s. 90

Rys. 8.4. Wywołania programu *Szyfr Cezara* dla kluczy 3 oraz 12

Ćwiczenie 3

Zapisz kod źródłowy programu *Szyfr Cezara*. Skompiluj kod i sprawdź działanie programu dla różnych słów i kluczy szyfrowania.

Szyfr Cezara – program szyfrujący długi tekst

Napiszemy program, który zaszyfruje szyfrem Cezara tekst składający się ze słów oddzielonych spacjami. Poniżej znajduje się specyfikacja problemu i lista kroków algorytmu dla tej specyfikacji.

Specyfikacja

Dane: tekst jawny składający się z wielkich liter alfabetu łacińskiego oraz spacji; klucz szyfrowania w postaci liczby całkowitej z przedziału od 1 do 25.

Wynik: tekst zaszyfrowany szyfrem Cezara z podanym kluczem. Znaki spacji pozostają w szyfrogramie niezmienione.

1. Wczytaj tekst jawny.
2. Wczytaj klucz szyfrowania i oznacz tę liczbę jako klucz.
3. Dla każdego znaku tekstu jawnego wykonuj kolejno kroki 4–6.
   4. Oznacz kod ASCII tego znaku jako kod.
   5. Jeśli kod jest kodem ASCII spacji, to dołącz do szyfrogramu spację. W przeciwnym razie wykonaj krok 6.
      6. Jeżeli kod + klucz jest liczbą niewiększą od kodu ASCII litery Z, to dołącz do szyfrogramu literę o kodzie ASCII równym kod + klucz. W przeciwnym razie dołącz do szyfrogramu literę o kodzie ASCII wynoszącym kod + klucz – 26.

Warto wiedzieć

Współczesne algorytmy kryptograficzne pozwalają nie tylko utajnić przesyłaną wiadomość, lecz także zrealizować inne cele. Zapewniają m.in.:
- spójność danych, dzięki czemu mamy pewność, że wiadomość nie była modyfikowana,
- uwierzytelnienie, czyli potwierdzenie tożsamości nadawcy i odbiorcy lub autentyczności przesyłanych danych.

Do tej pory teksty pobierane przez nas z klawiatury miały formę łańcucha znaków bez spacji. Użycie instrukcji `cin` do pobrania całego zdania nie zadziałałoby poprawnie, ponieważ `cin` pobiera znaki do wystąpienia pierwszej spacji. Aby w języku C++ wczytać i zapamiętać wiersz tekstu łącznie ze znakami spacji, można użyć **funkcji** `getline` z biblioteki `string`. Funkcja ta ma następującą budowę:

Funkcja `getline`

```
getline(cin, zmienna)
```

Pierwszym parametrem funkcji `getline` jest standardowe wejście (`cin`), a drugim zmienna typu `string`, w której przechowamy wszystkie znaki wprowadzone z klawiatury aż do naciśnięcia klawisza **Enter**. Mogą to być również np. znaki interpunkcyjne.

Dobra rada

Pamiętaj, że w języku C++ standardowym wejściem jest klawiatura, a standardowym wyjściem – ekran monitora.

Oto kluczowy fragment kodu źródłowego programu *Szyfr Cezara 2* – funkcja `Cezar` oraz funkcja główna:

Fragment kodu źródłowego programu *Szyfr Cezara 2*

```
1.  string Cezar(int klucz, string tekst)
2.  {
3.      char kod;
4.      int dl = tekst.length();
5.      string szyfrogram = "";
6.
7.      for (int i=0; i<dl; i++)
8.      {
9.          if (tekst[i] == ' ') kod = ' ';
10.         else
11.         {
12.             kod = tekst[i] + klucz;
13.             if (kod>'Z') kod = kod - 26;
14.         }
15.         szyfrogram = szyfrogram + kod;
16.     }
17.     return szyfrogram;
18. }
19.
20. int main()
21. {
22.     string jawny;
23.     int klucz;
24.
25.     cout << "Podaj tekst: ";
26.     getline(cin,jawny);
27.
28.     cout << "Podaj klucz: ";
29.     cin >> klucz;
30.
31.     cout << "Szyfrogram: " << Cezar(klucz,jawny) << endl;
32.
33.     return 0;
34. }
```

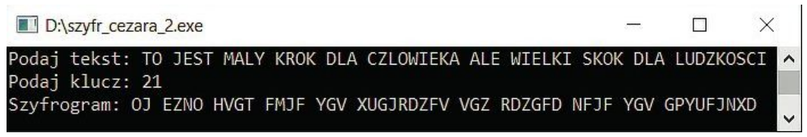
Warto wiedzieć

Choć spacja jest znakiem niedrukowanym, można jej użyć w wyrażeniu logicznym, wstawiając odstęp między apostrofami. Można też zapisać informację o spacji w zmiennej znakowej, np.:
`char spacja = ' ';`

Dobra rada

Jeśli pewne instrukcje są wykonywane kilka razy w tym samym programie, zapisz je w postaci funkcji.

W linii 26. do wczytania i zapamiętania wiersza tekstu, łącznie ze znakami odstępu, użyliśmy funkcji `getline` z biblioteki `string`. Tekst wpisany z klawiatury (`cin`) jest wczytany do zmiennej `jawny`. Na rysunku 8.5 przedstawiono wywołanie programu *Szyfr Cezara 2*.



Rys. 8.5. Wywołanie programu *Szyfr Cezara 2* dla klucza 21

Warto wiedzieć

Funkcja `getline` ma również postać bardziej ogólną:
`getline(cin, zmienna, znak)`
Pobiera ona ze standardowego wejścia (`cin`) łańcuch znaków do momentu wystąpienia wskazanego znaku i zapisuje ten łańcuch w zmiennej typu `string`.

Ćwiczenie 4

Zapisz kod źródłowy programu *Szyfr Cezara 2*, skompiluj go i uruchom program. Sprawdź jego działanie dla napisu: PRZYJDZ DO SKLEPU I ZAPYTAJ O OLGE i klucza szyfrowania równego 13.

Deszyfrowanie wiadomości zaszyfrowanych metodą Cezara

Aby odszyfrować wiadomość zapisaną alfabetem łacińskim, zaszyfrowaną szyfrem Cezara ze znaną wartością klucza, wystarczy od kodu znaku napisu odjąć wartość klucza.

Dobra rada

Aby odczytać wiadomość zaszyfrowaną szyfrem Cezara ze znanym kluczem, możesz też ponownie ją zaszyfrować z nowym kluczem o wartości 26 – *k*, gdzie *k* jest kluczem zastosowanym do zaszyfrowania pierwotnego tekstu.

Ćwiczenie 5

Napisz program deszyfrujący tekst zaszyfrowany metodą Cezara ze znaną wartością klucza. Sprawdź działanie programu dla szyfrogramu PKYZKY SOYZXFKS QXEVZUMXGLOO i klucza 6.

Zapamiętaj

Szyfr Cezara to szyfr podstawieniowy, który polega na zastąpieniu każdej litery tekstu jawnego literą oddaloną od niej o stałą liczbę pozycji w alfabecie. Przesunięcie odbywa się cyklicznie, tzn. po ostatniej literze alfabetu następuje pierwsza.

8.3. Metody łamania klasycznych szyfrów

Kryptografia (od gr. *kryptós* – ukryty i *gráphō* – piszę) jako nauka o utajnianiu wiadomości jest częścią szerszej dziedziny nazywanej **kryptologią** (od gr. *kryptós* – ukryty i *lógos* – słowo).

Kryptologia

Kryptologia obejmuje wiedzę dotyczącą szyfrowania, deszyfrowania oraz łamania szyfrów w celu bezpiecznego przekazywania informacji. Nauka, która zajmuje się łamaniem szyfrów, to **kryptoanaliza** (od gr. *kryptós* – ukryty i *analýein* – rozluźnić, rozwiązać).

Kryptoanaliza

Warto wiedzieć

Współcześnie stosowane szyfry używają jako kluczy liczb, które w zapisie binarnym mają po kilkaset, a nawet kilka tysięcy cyfr (bitów).

Zastosujemy kilka prostych technik kryptoanalizy do złamania poznanego szyfru. Choć klasyczne szyfry dziś już nie chronią skutecznie przesyłanych informacji, warto je analizować również pod kątem technik ich łamania. Oto kilka z nich.

Atak siłowy

Atak siłowy (brute-force)

Aby odczytać tekst jawny składający się z liter alfabetu łacińskiego zaszyfrowany metodą Cezara, wystarczy wygenerować 25 możliwych tekstów i wybrać właściwy. Dla współczesnych komputerów to bardzo niewiele. Technika ta jest nazywana **atakiem siłowym** lub **brute-force** (dosł. brutalnej siły). Przykład działania programu łamiącego szyfrogram pokazuje rysunek 8.6.

Warto wiedzieć

W języku łacińskim przymiotnik *brutus* pierwotnie oznaczał „ciężki”. Później zaczęto go też używać w znaczeniu „głupi”.

Rys. 8.6. Zastosowanie ataku siłowego do łamania szyfru Cezara

A to ciekawe

(Kryptologiczny) Cud nad Wisłą

	0	1	2	3	4	5	6	7	8	9	
0		Д	А	З	М	Щ	Ш	Р		К	0
1	Ч	В	П	Ф		С	Ч		Й		1
2			Ю		Ж			О	Г		2
3											3
4											4
5											5
6											6
7											7
8											8
9											9
	0	1	2	3	4	5	6	7	8	9	

Choć o zwycięstwie Polaków nad Armią Czerwoną w 1920 r. często mówi się w kategoriach cudu, udział w nim miały m.in. osiągnięcia kryptologiczne. Dzięki pracom polskiego Biura Szyfrów w latach 1919–1920 udało się odszyfrowywać kilka tysięcy szyfrogramów przesyłanych przez Rosjan drogą radiową. Zespołem kierował porucznik Jan Kowalewski, który do współpracy zaprosił wybitnych matematyków: Wacława Sierpińskiego, Stefana Mazurkiewicza i Stanisława Leśniewskiego. Opracowali oni nowatorską metodę lingwistyczno-matematycznego ataku na szyfr. Była ona potem rozwijana przez kolejne pokolenia kryptologów, w tym polskich matematyków, którzy złamali szyfr Enigmy.

Ćwiczenie 6

Napisz program łamiący szyfr Cezara techniką ataku siłowego.

Ataki ze znanym tekstem jawnym

Jeśli atakujący zna część tekstu zaszyfrowanego i odpowiadającą jej część tekstu jawnego, może to wykorzystać do odszyfrowania całego tekstu. Na przykład jeśli przechwytujący szyfrogram wie, że autor wszystkie swoje wiadomości zaczyna od słów „dzień dobry”, to może dzięki tej informacji łatwiej znaleźć zastosowany klucz szyfrowania.

Atak ze znanym tekstem jawnym od razu łamie szyfr Cezara. Aby wyznaczyć klucz, wystarczy bowiem sprawdzić przesunięcie zastosowane dla pierwszych liter analizowanej wiadomości.

Analiza częstości

Szyfr Cezara można złamać z wykorzystaniem **analizy częstości** wystąpień poszczególnych liter w szyfrogramie. Jeśli wiemy, w jakim języku zapisana jest informacja jawna, możemy założyć, że najczęściej występująca litera w szyfrogramie odpowiada najczęściej używanej literze w tym języku. Wówczas kluczem użytym do zaszyfrowania informacji jest różnica między pozycjami litery najczęściej występującej w szyfrogramie oraz litery najczęściej pojawiającej się w danym języku. W przypadku języka polskiego, jeśli litery alfabetu numerujemy od 0, tą różnicą będzie pozycja najczęściej występującej litery w szyfrogramie. Wynika to z tego, że w języku polskim najczęściej pojawia się litera A.

Analiza częstości liter

Warto podkreślić, że łamanie szyfru Cezara z wykorzystaniem analizy częstości nie zawsze jest skuteczne. Ta metoda może dać fałszywe wyniki np. w przypadku krótkich informacji.

Warto wiedzieć

Krótkie teksty zaszyfrowane w prosty sposób mogą być nie do złamania nawet przez potężne komputery. Claude Shannon wykazał, że im krótszy szyfrogram, tym trudniej określić rodzaj szyfru. Niektóre teksty wyryte na starych nagrobkach pozostają nieodczytane przez setki lat mimo wysiłków wielu kryptologów.

Ćwiczenie 7

Znajdź informację o tym, jaka litera występuje najczęściej w tekstach zapisanych w:

a. języku angielskim,
b. wybranym przez ciebie języku, innym niż angielski i polski.

Zapamiętaj

Każdy szyfr można złamać przez odnalezienie pasującego klucza. Stopień bezpieczeństwa algorytmu szyfrowania mierzy się ilością czasu, jakiej wymaga jego złamanie (wyczerpujące przeszukanie zbioru potencjalnych kluczy). Szyfr Cezara nie stanowi dziś praktycznie żadnego zabezpieczenia.

ANALIZA CZĘSTOŚCI W JĘZYKU POLSKIM

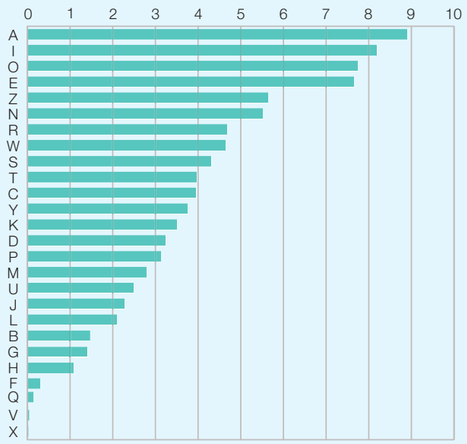
Pokażemy, jak wykorzystać analizę częstości do odszyfrowania szyfrogramu, który powstał po zaszyfrowaniu informacji w języku polskim. Posłużymy się tekstem zapisanym wyłącznie wielkimi literami alfabetu łacińskiego. W tekście nie używamy znaków interpunkcyjnych, a wyrazy są oddzielone spacjami.

Fragment szyfrogramu

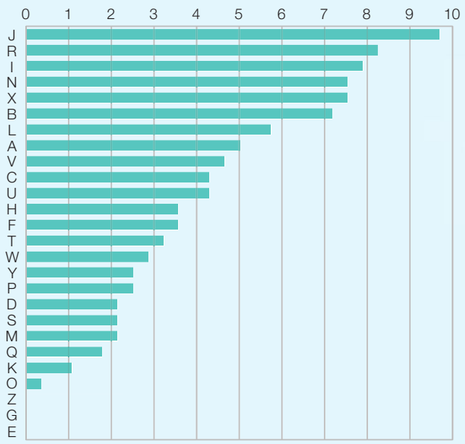
YXMLIJB UJVJWRJ BIHOAXPAJVD VXIN BRN YAIHMJL WRN CHUTX JWJURIJ
LINBCXBLR URCNA UNLI CJTIN IWJSXVXBL LINBCX FHBCNYDSJLHLQ XKXT
BRNKRN PADY MFXLQ UDK CAINLQ IWJTXF LIHUR MRPAJVXF R CARPAJVXF WJ
YAIHTUJM F SNIHTD YXUBTRV WJSLINBCBIHVR MRPAJVJVR BJ LQ LI BI R AI J
CARPAJVNV BUXFX BRN F SNIHTD JWPRNUBTRV LINBCX YXSJFRJ BRN BUXFX CQN

Deszyfrujemy szyfrogram

Przeciętna częstość występowania liter w języku polskim* [%]



Częstość występowania liter w szyfrogramie [%]



Z wykresu po prawej stronie odczytujemy, że najczęściej występującą literą w szyfrogramie jest litera J. Może to oznaczać, że litery alfabetu przesunięto o 9 pozycji. Po odszyfrowaniu szyfrogramu z użyciem klucza 9 otrzymujemy właściwą informację.

* Nie uwzględniamy częstości występowania liter z polskimi znakami diakrytycznymi.

Tekst jawny po odszyfrowaniu fragmentu kryptogramu

PODCZAS LAMANIA SZYFROGRAMU MOZE SIE PRZYDAC NIE TYLKO ANALIZA
CZESTOSCI LITER LECZ TAKZE ZNAJOMOSC CZESTO WYSTEPUJACYCH
OBOK SIEBIE GRUP DWOCH LUB TRZECH ZNAKOW CZYLI DIGRAMOW
I TRIGRAMOW NA PRZYKLAD W JEZYKU POLSKIM NAJCZESTSZYMI DIGRAMAMI
SA CH CZ SZ I RZ A TRIGRAMEM SLOWO SIE W JEZYKU ANGIELSKIM
CZESTO POJAWIA SIE SLOWO THE

Zadania

* **1** Poszukaj informacji na temat historii steganografii, czyli sztuki ukrywania informacji. Przygotuj krótką prezentację na ten temat. Uwzględnij następujące przykłady: akrostych, mikropunkt, szyfr szpilkowy.

* **2** Przygotuj prezentację na temat sposobu szyfrowania tajnej korespondencji w służbie dyplomatycznej Stanisława Augusta Poniatowskiego, ostatniego króla Polski.

* **3** Przygotuj prezentację na temat maszyny szyfrującej Enigma oraz historii złamania jej szyfru przez polskich matematyków w latach 30. XX w. Przedstaw na osi czasu kluczowe wydarzenia związane z historią łamania szyfru Enigmy przed wybuchem II wojny światowej.

** **4** Jednym z prostych szyfrów przestawieniowych jest szyfr polegający na zamianie miejscami znaków na pozycjach parzystych i nieparzystych (pierwszy znak zamieniamy miejscami z drugim, trzeci z czwartym itd.). Napisz program szyfrujący i program deszyfrujący tekst tą metodą. Przyjmij, że tekst jawny składa się z parzystej liczby liter.

** **5** Szyfr Cezara z przesunięciem (kluczem) o wartości 13 jest nazywany algorytmem ROT13. Jest on współcześnie używany w dyskusjach internetowych (w języku angielskim), aby jakaś część wiadomości nie była możliwa do odczytania przez przypadkowe osoby. Dowiedz się, dlaczego używa się przesunięcia o 13 liter. Napisz program do szyfrowania i deszyfrowania wiadomości algorytmem ROT13.

** **6** Zmodyfikuj program do szyfrowania algorytmem Cezara tak, aby pozwalał na szyfrowanie i deszyfrowanie z wykorzystaniem odwróconego szyfru Cezara.

W odwróconym szyfrze Cezara wyjściowy układ alfabetów dla klucza równego 0 ma postać:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

Alfabet szyfrowy dla klucza równego 16 ma postać:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q

*** **7** Przygotuj prezentację na temat kilku wybranych metod klasycznej kryptoanalizy. W swoim opracowaniu uwzględnij nowożytny szyfr Vigenère'a i okoliczności jego złamania w XIX w.

*** **8** Napisz program szyfrujący dla szyfru Cezara, w którym wprowadzisz rozróżnienie wielkich i małych liter.
Uwaga: Liczba kluczy wzrośnie dwukrotnie.

Algorytmika i programowanie w języku C++

Wyobraź sobie, że...

...zostałeś zatrudniony jako młodszy specjalista ds. transportu i magazynowania. Zauważasz problem związany z jednoczesnymi dostawami do magazynu części z dwóch różnych firm, realizowanymi w cyklach co 4 i 6 dni roboczych. Stwierdzasz, że można go łatwo rozwiązać, wykorzystując pojęcie **NWW**. Uzasadniasz, że dostawy w cyklach co 5 i 6 dni będą się pokrywać tylko raz na 30 dni. Wiedzę o **algorytmie Euklidesa** wykorzystujesz przy planowaniu przestrzeni magazynowych – stosujesz podejście znane ci z **wycinanki Euklidesa**.

...znalazłaś pracę jako asystentka marketingu cyfrowego. Analizując listy subskrybentów, zauważasz **zduplikowane** imiona i nazwiska, co może oznaczać, że klienci wielokrotnie zakładali konta. Dostrzegasz też literówki w adresach mailowych, spowodowane brakiem mechanizmu **sprawdzania adresu**. Zgłaszasz problem do działu IT. Poprawiasz błędy, korzystając z funkcji sprawdzania pisowni w przeglądarce internetowej – **algorytmu wyszukiwania wzorca w tekście**.

...pracujesz w dziale administracji dużej firmy. Twoim zadaniem jest wysłanie pocztą elektroniczną poufnych dokumentów do ważnego kontrahenta. Postanawiasz **zaszyfrować** przesyłane informacje. Z lekcji informatyki pamiętasz, że warunkiem tajności informacji jest utrzymanie w tajemnicy **klucza szyfrowania**. Przygotowaną wiadomość przesyłasz kontrahentowi e-mailem, ale hasło **deszyfrujące** przekazujesz w wiadomości wysłanej z bezpiecznego komunikatora.

...postanowiłaś zawodowo zająć się fotografią. Kupiłaś profesjonalny aparat, tzw. lustrzankę, i zapisałaś się na kurs. Na pierwsze zajęcia masz przynieść ze sobą kilka zdjęć. Pomimo niewielkiej wiedzy na temat fotografii starasz się wykorzystać zasadę **złotej proporcji**. Zaskakujesz prowadzących, objaśniając, że fotografowany obiekt umieściłaś dokładnie 5 jednostek od lewej krawędzi kadru i 8 jednostek od prawej, gdyż są to kolejne liczby **ciągu Fibonacciego**.

3. Projektowanie 3D

9 Grafika 3D

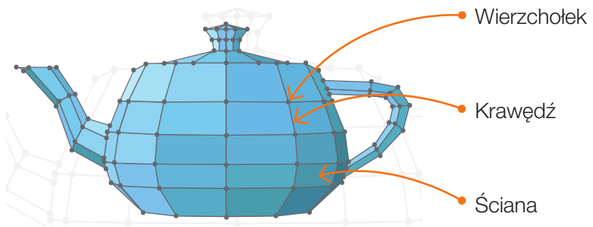
Grafika trójwymiarowa ma dużą zaletę. Pozwala pokazać to, co nie jest widoczne na grafice dwuwymiarowej, czyli właśnie trzeci wymiar. Płaski obraz ma wysokość i szerokość, a grafika 3D – dodatkowo także głębokość. To daje nowe możliwości. W tym temacie nauczysz się tworzyć modele trójwymiarowe.

Ważne w tym temacie

- Modelowanie 3D
- Widok trójwymiarowy
- Tworzenie modeli 3D

W temacie o grafice 3D poznamy nowe pojęcia, którymi opisuje się tworzone obiekty. Podstawowymi komponentami obiektów w grafice 3D są siatki wielokątne (ang. *meshes*) składające się z wierzchołków (ang. *vertices*), krawędzi (ang. *edges*) i ścian (ang. *faces*). Wszystkie te elementy tworzą **geometrię obiektów** (rys. 9.1).

Geometria obiektów



Rys. 9.1. Geometria obiektu 3D

9.1. Modelowanie 3D

Modelowanie 3D

Z grafiką 3D spotykamy się od dawna w bajkach, animacjach i grach komputerowych. Programy pozwalające na **modelowanie 3D**, czyli tworzenie przestrzennych obiektów, działają nawet z poziomu przeglądarki internetowej. Dzięki rozwojowi druku 3D nie musimy kończyć pracy na wykonaniu modelu w programie graficznym. Kolejnym krokiem może być zmaterializowanie wyniku za pomocą drukowania przestrzennego.

Programy do modelowania lub animacji 3D to często skomplikowane w obsłudze narzędzia wymagające odpowiednich umiejętności. Pozwalają one tworzyć bardzo rozbudowane, wieloelementowe konstrukcje, które dodatkowo można wprawić w ruch. Przygotowanie mniej skomplikowanych projektów, nawet przeznaczonych do wydruku 3D, wcale nie jest trudne.

Warto wiedzieć

Żeby zarządzać zaawansowanym projektem 3D, grafik wykorzystuje funkcjonalność warstw. Widoczność warstw można włączać i wyłączać, aby skupić się na jednej, konkretnej części projektu.

Jednym z najpopularniejszych bezpłatnych programów służących do modelowania 3D jest Blender, dostępny na licencji GNU GPL. Spośród programów płatnych warto wymienić programy: 3ds Max, Maya, Maxon Cinema 4D, Modo, Rhino 3D.

Do narzędzi pozwalających tworzyć grafikę 3D należy również program SketchUp, którego możliwości zaprezentujemy w tym temacie. Program można uruchomić wprost z przeglądarki internetowej na stronie https://www.sketchup.com/products/sketchup-free. Kliknięcie przycisku **Start Modeling** spowoduje załadowanie w przeglądarce internetowej aplikacji do modelowania.

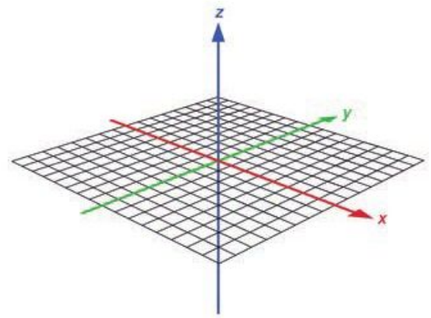
Dobra rada

Aby skorzystać z aplikacji SketchUp (zapisywać prace, nadawać im nazwy, uruchamiać ponownie), musisz mieć konto użytkownika. Możesz je założyć bezpłatnie.

Widok trójwymiarowy

Podstawowym widokiem w programie SketchUp jest układ trójwymiarowy z widocznymi liniami w kolorach zielonym, czerwonym i niebieskim. Są to osie trójwymiarowego układu współrzędnych. W dwuwymiarowym układzie współrzędnych osie są oznaczone jako *x* i *y*, a każdy punkt opisują dwie współrzędne. Na przykład początek układu współrzędnych znajduje się w punkcie (0, 0).

W układzie trójwymiarowym (rys. 9.2) osie są oznaczone jako *x*, *y* i *z*, a każdy punkt składa się z trzech współrzędnych. Początek układu ma więc współrzędne (0, 0, 0). Do sprawnego poruszania się w tym układzie służą narzędzia **Zoom** (przybliżanie), **Orbit** (obracanie) oraz **Pan** (przesuwanie bez obracania).



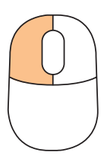
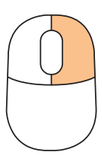
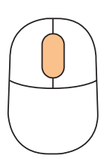
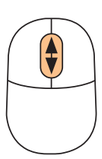
Rys. 9.2. Układ trójwymiarowy

Dobra rada

Jeśli nie widzisz jakiegoś narzędzia, kliknij 3 kropki widoczne na końcu paska narzędzi. Pojawi się wtedy panel z dodatkowymi narzędziami, a trzy ostatnio wybrane będą widoczne na końcu paska narzędzi.

Obsługa myszy

Aby w programie SketchUp sterować widokiem kamery, można skorzystać z wymienionych wcześniej narzędzi. Jednak każdą z tych operacji można też wykonać za pomocą myszy komputerowej (rys. 9.3).

 Lewy przycisk myszy pozwala na operowanie wybranymi narzędziami. Na przykład dla narzędzia **Select** (skrót: **Spacja**) pozwala zaznaczać krawędzie, ściany i całe obiekty. Wiele obiektów zaznaczamy z klawiszem **Shift**.	 Prawy przycisk myszy po kliknięciu obiektu uruchamia menu z dodatkowymi opcjami.
 Naciśnięcie środkowego przycisku myszy pozwala na operowanie kamerą w trójwymiarowym układzie współrzędnych.	 Kiedy kręcimy środkowym przyciskiem myszy, przybliżamy bądź oddalamy widok.

Rys. 9.3. Obsługa myszy w programie SketchUp

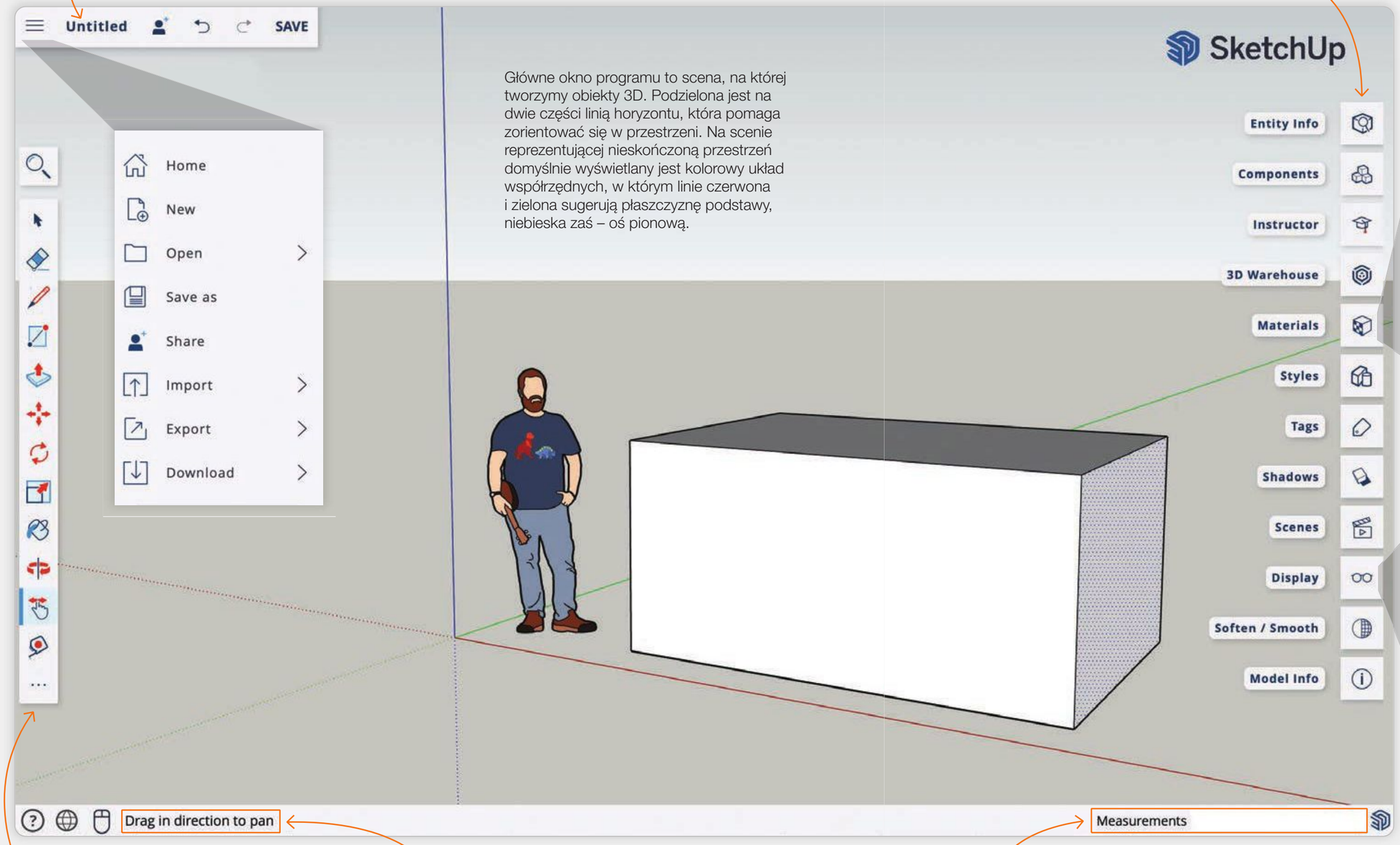
Warto wiedzieć

Pojedyncze kliknięcie w obiekt zaznacza tylko jego ścianę lub krawędź (tę, którą klikniemy), podwójne kliknięcie zaznacza ścianę wraz z jej krawędziami, natomiast potrójne kliknięcie zaznacza cały obiekt.

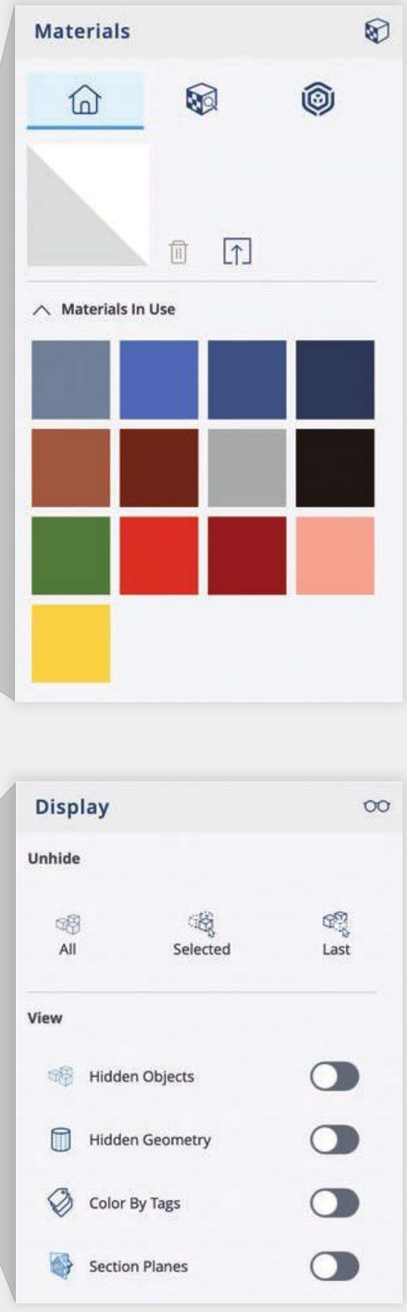
OKNO PROGRAMU SKETCHUP

Górne menu programu umożliwia tworzenie i zapisywanie nowego projektu oraz ponowne otwarcie projektu zapisanego wcześniej. Tu wyświetlana jest nazwa pliku, nad którym pracujemy.

Z prawej strony ekranu znajduje się pionowy pasek z ikonami dodatkowych okien. Rozwijamy je po kliknięciu wybranej ikony. To tu np. możemy wybrać materiał, który nałożymy na ścianki swojego modelu, ustawić styl widoku dla całego projektu lub takie efekty jak mgła i cienie. Można ustawiać parametry komponentów oraz wyszukiwać online potrzebne obiekty.



Główne okno programu to scena, na której tworzymy obiekty 3D. Podzielona jest na dwie części linią horyzontu, która pomaga zorientować się w przestrzeni. Na scenie reprezentującej nieskończoną przestrzeń domyślnie wyświetlany jest kolorowy układ współrzędnych, w którym linie czerwona i zielona sugerują płaszczyznę podstawy, niebieska zaś – oś pionową.



Dzięki narzędziom programu można tworzyć obiekty 3D na scenie. Więcej opcji jest dostępnych po kliknięciu ikony z trzema kropkami znajdującej się na dole paska narzędzi.

Wiele narzędzi ma dodatkowe możliwości, jeżeli używamy ich z odpowiednim przyciskiem na klawiaturze. To w tym miejscu pojawiają się podpowiedzi, których przycisków klawiatury można użyć dla wybranego narzędzia, aby mogło wykonywać dodatkowe czynności.

To niewielkie pole tekstowe jest bardzo przydatne. Wyświetla wielkości miar, kątów, odległości itp. Widać w nim również parametry wpisywane za pomocą klawiatury. Aby np. narysować trójkąt, nie wystarczy wybrać narzędzia **Polygon**, trzeba też – po jego wybraniu – napisać na klawiaturze „3s”, gdzie „3” oznacza liczbę boków.

Ćwiczenie 1

Uruchom aplikację SketchUp. Pojawi się widok postaci w układzie trójwymiarowym. Korzystając z narzędzi programu lub przycisków myszy, ustaw kamerę tak, aby patrzyła na model postaci:

a. z góry,
b. z dołu,
c. z oddali,
d. z bliska.

Rysowanie obiektów

Obiekty można tworzyć z wykorzystaniem bogatej listy narzędzi, począwszy od prostych narzędzi **Line** oraz **Freehand** (rysowanie swobodne), **Circle**, **Rectangle**, aż po narzędzia pozwalające rysować w różnych płaszczyznach, np. **Rotated Rectangle**, **Pie**.

Dobra rada

Po kliknięciu trzech kropek na końcu paska narzędzi i wybraniu przycisku **Edit** możesz zmienić układ narzędzi na pasku.

Materiały

Utworzonym obiektom 3D można nadawać kolor i **fakturę**, aby oddać ich przeznaczenie. Służy do tego narzędzie **Paint**. Po kliknięciu w nie otworzy się okno **Materials**, w którym dostępne są kolory oraz m.in. różne rodzaje drewna, metalu, szkła i kamienia. W górnej części okna **Materials** są ikony narzędzi: **InModel** (kolory i faktury znajdujące się w scenie), **Browse** (biblioteka dowolnych kolorów i materiałów), **3D Warehouse** (obszerna biblioteka online).

Faktura to sposób ukształtowania powierzchni materiału, np. drewna, kamienia.

Zapis do różnych formatów graficznych

Program umożliwia zapisywanie stworzonych projektów w formacie SKP do pobrania na komputer lub zapisu w chmurze obliczeniowej producenta. Możliwy jest eksport do plików PNG oraz STL, który jest formatem dla drukarek 3D.

9.2. Tworzenie trójwymiarowego modelu szafki w programie SketchUp

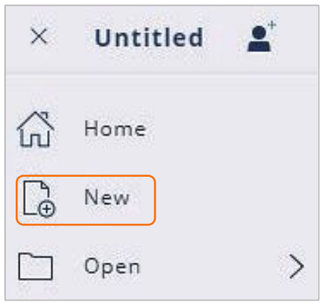
Programy do grafiki 3D są powszechnie wykorzystywane przez projektantów wystroju wnętrz. W programie SketchUp w kilku prostych krokach możemy sami stworzyć trójwymiarowy model szafki z szufladami, taki jak na rysunku 9.4, s. 119.

Warto wiedzieć

Jeżeli po uruchomieniu aplikacji rozwinie się strzałkę obok przycisku **Create new**, będzie można wybrać jednostki, w których będą tworzone obiekty, np. Decimal-Meters.

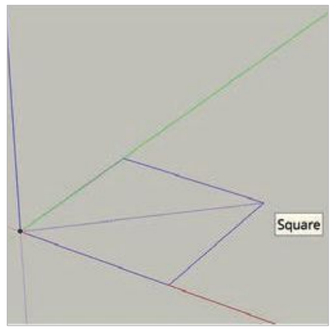
Po uruchomieniu aplikacji online rozwijamy menu w lewym górnym rogu i wybieramy **New** (rys. 9.5, s. 119). Nie musimy od razu usuwać domyślnej postaci znajdującej się obok środka układu współrzędnych. Możemy to zrobić później klawiszem **Delete**. Ta postać pozwala nam zobaczyć proporcje tworzonego obiektu w odniesieniu do wysokości człowieka. W zależności od potrzeb możemy zmieniać położenie postaci.

Rys. 9.4. Trójwymiarowy model szafki stworzony w programie SketchUp



Rys. 9.5. Tworzenie nowego modelu

Na początku korzystamy z narzędzia **Rectangle** i rysujemy kwadrat w płaszczyźnie poziomej (rys. 9.6). W tym celu klikamy w dowolnym miejscu płaszczyzny podstawy, by zaczepić narzędzie, a następnie zwalniamy lewy przycisk myszy i przesuwamy mysz, aby sprawdzić, czy poprawnie zaczepiono wierzchołek prostokąta. Na koniec wpisujemy z klawiatury wymiary 1, 1 i zatwierdzamy je klawiszem **Enter**. Przecinek oddziela długości boków figury.



Rys. 9.6. Rysowanie kwadratu w płaszczyźnie poziomej

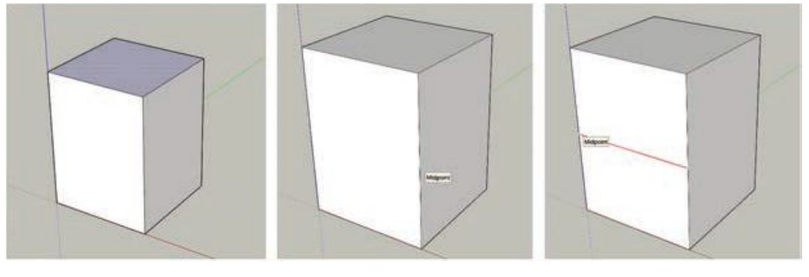
Narzędzia grafiki 3D w programie SketchUp
→ s. 218–219

Dobra rada

Wpisuj wymiary, gdy narzędzie **Rectangle** z zaczepionym wierzchołkiem kwadratu jest aktywne.

W kolejnym kroku za pomocą narzędzia **Push/Pull** wyciągamy wysokość. Tak jak w poprzednim kroku klikamy pole kwadratu, aby tylko zaczepić narzędzie na jego powierzchni. Następnie zwalniamy przycisk myszy, wyciągamy na dowolną wysokość, wpisujemy 1.5 z klawiatury i zatwierdzamy klawiszem **Enter**.

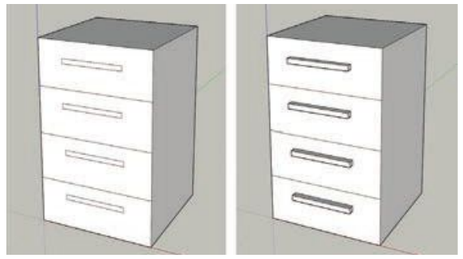
Za pomocą narzędzia **Line** na jednym z boków szafki narysujemy szuflady (rys. 9.7). Musimy zwrócić uwagę na to, aby rysowane linie biegły równolegle do jednej z osi płaszczyzny podstawy – będą wtedy oznaczane kolorem tej osi. Zauważmy też, że kiedy prowadzimy narzędzie **Line** wzdłuż krawędzi, program oznacza punkty skrajne krawędzi (**Endpoint**) na zielono oraz środkowe (**Midpoint**) – na niebiesko. Dzięki temu można łatwo podzielić każdy odcinek na połowę.



Rys. 9.7. Użycie narzędzia **Push/Pull** i wprowadzenie linii w połowie wysokości szafki

Dobra rada

Rysowanie kwadratu lub prostokąta czy wyciąganie wysokości możesz wykonać również, korzystając z operacji **drag and drop** – kliknij lewym przyciskiem myszy i puść go po uzyskaniu odpowiedniego rozmiaru. Dopóki nie wykonasz nowej operacji myszą, możesz aktualizować wymiary, wpisując je z klawiatury i zatwierdzając klawiszem **Enter**.



Rys. 9.8. Tworzenie uchwytów do szuflad

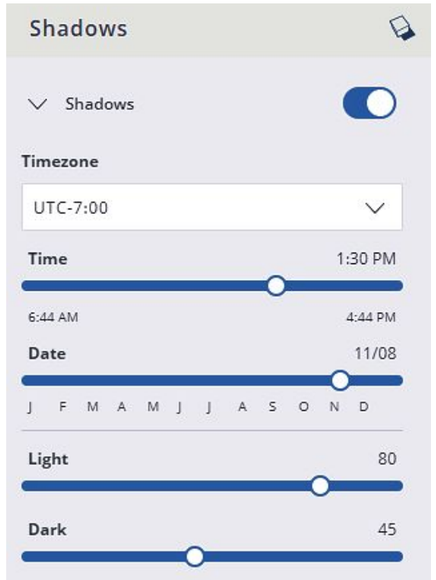
Dodamy prostokątne uchwyty szuflad, jak na rysunku 9.8. Aby zrobić identyczne uchwyty do każdej szuflady, możemy narysować linie pomocnicze, które potem wymażemy. Możemy również skopiować prostokąt (z którego powstanie uchwyt) i wkleić go na kolejne szuflady. Aby skopiować element przy użyciu narzędzia do przemieszczania, zaznaczamy prostokąt narysowany wcześniej na ściance szuflady, wybieramy narzędzie **Move** i naciskamy klawisz **Ctrl** (przy narzędziu **Move** pojawi się znak plusa). Następnie łapiemy za wierzchołek (dowolny narożny punkt kopiowanego prostokąta) i przesuwamy prostokąt na kolejną szufladę. Narzędziem **Push/Pull** wyciągamy powierzchnię każdego uchwytu na niewielką odległość. Aby te odległości dla każdego uchwytu były jednakowe, po wyciągnięciu pierwszego uchwytu wystarczy kliknąć dwukrotnie narzędziem **Push/Pull** na pozostałych.

Dobra rada

Aby powielić zaznaczony element, możesz skorzystać ze skrótów klawiszowych **Ctrl + C** oraz **Ctrl + V**.

Na ostatnim etapie możemy dodać więcej detali, np. za pomocą linii wydzielić boczne fragmenty szuflad na bocznych ściankach (rys. 9.4, s. 119). Dodajemy tekstury: metalu dla uchwytów, a drewna dla frontów szafki i szuflad. W tym celu wybieramy narzędzie **Paint**.

Możemy też dodać blat na górze szafki. Aby to zrobić, potrójnym kliknięciem zaznaczamy szafkę, a potem z menu kontekstowego wybieramy opcję **Make group**. Za pomocą narzędzi **Rectangle** oraz **Push/Pull** tworzymy blat, a następnie dobieramy właściwą teksturę. Na koniec dzięki narzędziu **Shadows** możemy dodać cieniowanie modelu (rys. 9.9).



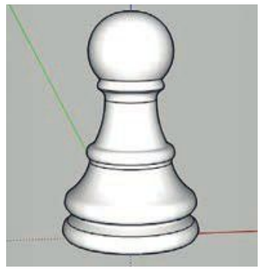
Rys. 9.9. Opcje włączania cienia

Ćwiczenie 2

Wykonaj model szafki, w której uchwyty i układ szuflad będą inne niż na rys. 9.8.

9.3. Tworzenie bryły obrotowej w programie SketchUp

W programie SketchUp można łatwo tworzyć bryły obrotowe, czyli takie, które powstają przez obracanie się figury płaskiej wokół pewnej osi. Poniżej omówimy proces tworzenia modelu pionka szachowego (rys. 9.10). Wykorzystamy fakt, że jest to bryła obrotowa.



Rys. 9.10. Trójwymiarowy model pionka szachowego

Warto wiedzieć

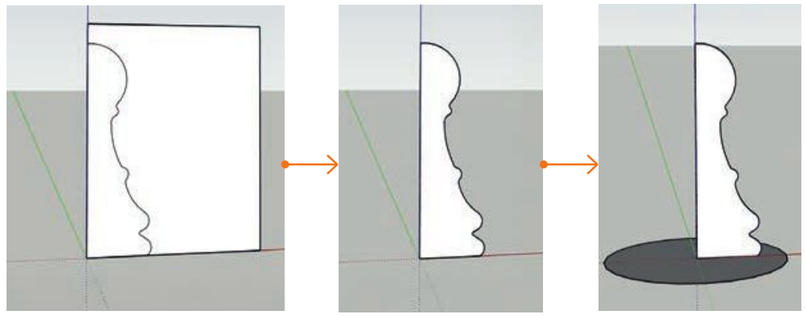
Przykładem bryły obrotowej jest stożek, który powstaje w wyniku obracania się trójkąta prostokątnego wokół osi przechodzącej przez jedną z przyprostokątnych.

Zaczynamy od narysowania prostokąta w płaszczyźnie pionowej. Przy jego lewej krawędzi rysujemy kształt przypominający prawą połowę przekroju pionka (pierwszy obraz na rys. 9.11). Wykorzystujemy do tego celu narzędzia rysunkowe, takie jak linie, łuki, rysunek odręczny. Kształt ten po narysowaniu powinien się dać zaznaczyć.

Usuwamy krawędzie prostokąta. Pozostawiamy tylko te, które należą do narysowanego kształtu.

Dobra rada

Zaznaczony kształt rozpoznasz po tym, że jego powierzchnia wygląda jak wypełniona punktami.



Rys. 9.11. Etapy tworzenia pionka szachowego

Następnie zmieniamy widok tak, aby patrzeć na kształt z góry. W płaszczyźnie, na której znajdzie się podstawa pionka, rysujemy koło w taki sposób, aby jego środek leżał na osi obrotu pionka (ostatni obraz na rys. 9.11). Zwracamy uwagę, by kontur koła był niebieski – wtedy mamy pewność, że leży w płaszczyźnie podstawy.

Pozostaje wykonać symulację obrotu. Zaznaczamy kontur koła narzędziem **Select**, a następnie wybieramy narzędzie **Follow Me** i klikamy nim dwukrotnie wewnątrz kształtu, z którego ma powstać pionek. W efekcie powstanie bryła taka jak na rysunku 9.10.

Ćwiczenie 3

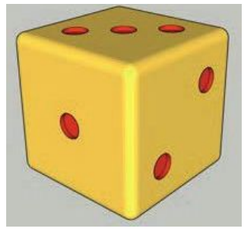
Stwórz bryłę obrotową powstałą w wyniku obrotu:

a. prostokąta wokół osi przechodzącej przez wybrany bok,

b. dowolnego kształtu wokół wybranej osi.

9.4. Łączenie różnych technik tworzenia grafiki 3D

Program SketchUp w wersji online ma niewiele narzędzi, pozwala jednak tworzyć trójwymiarowe modele według własnych pomysłów. Ogranicza nas w tym tylko nasza wyobraźnia. Możemy np. wykonać model sześciennej kostki do gry. Jednak zrobienie kostki, której krawędzie będą zaokrąglone (rys. 9.12), nie jest już takie proste. Spróbujmy ją wykonać.

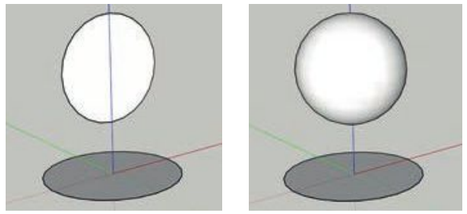


Rys. 9.12. Kostka do gry

Warto wiedzieć

Wersja online programu SketchUp nie pozwala na zaznaczanie i modyfikację wierzchołków siatki. Można jednak zaznaczać krawędzie lub ścianki.

Zaczynamy od utworzenia dwóch kół umieszczonych na prostopadłych względem siebie płaszczyznach. Środki kół powinny być na jednej linii. Zaznaczamy krawędź dolnego koła narzędziem **Select**. Następnie wybieramy narzędzie **Follow me** i klikamy wewnątrz górnego koła, żeby powstała kula (rys. 9.13). Dolne koło możemy usunąć.

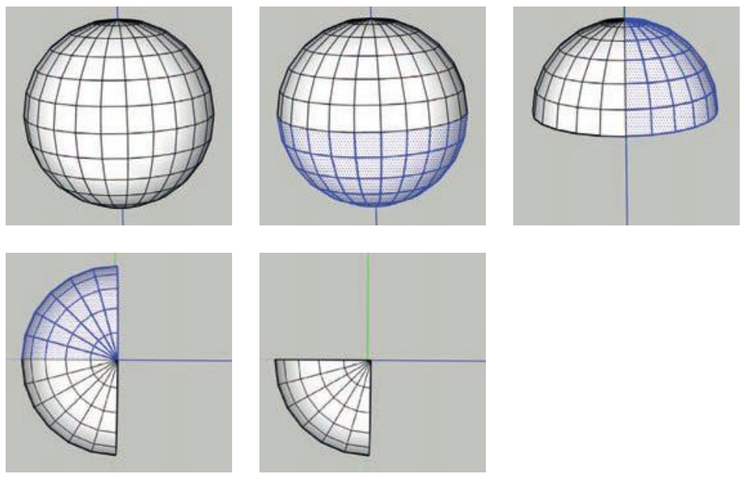


Rys. 9.13. Tworzenie kuli

Warto wiedzieć

W programach grafiki 3D na określenie ścianek siatki często używa się pojęć *faces* i *polygons*.

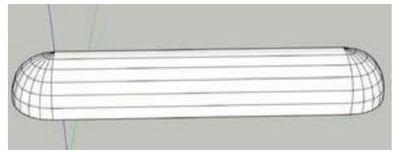
Klikamy trzykrotnie kulę i w oknie **Soften/Smooth** z prawej strony ekranu włączamy **Smooth Edges**. Powierzchnia kuli pokryje się siatką z wydzielonymi elementami (krawędzie, ścianki). Pomoże nam to wykonać kolejny krok. Zaznaczamy i usuwamy dolną połowę kuli, a następnie połowę pozostałej części. Ponownie usuwamy połowę uzyskanej części. To, co zostanie, to jeden narożnik kostki (rys. 9.14).



Rys. 9.14. Tworzenie narożnika kostki do gry

Kopiujemy ten narożnik i przesuwamy go wzdłuż jednej z osi współrzędnych (np. w prawo). Zapamiętujemy też odległość, o jaką przesunęliśmy skopiowany element. Wykonujemy odbicie lustrzane narzędziem **Flip** tak, aby oba narożniki były zwrócone do siebie jak na rysunku 9.15. Za pomocą narzędzia **Line** łączymy odpowiadające sobie wierzchołki ścianek. Otrzymaliśmy jedną krawędź kostki (rys 9.16).

Rys. 9.15. Odbicie lustrzane narożników

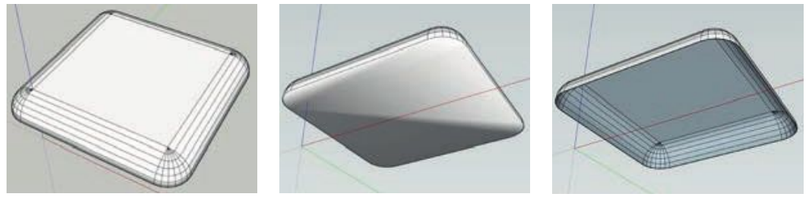


Rys. 9.16. Pojedyncza krawędź kostki

Dobra rada

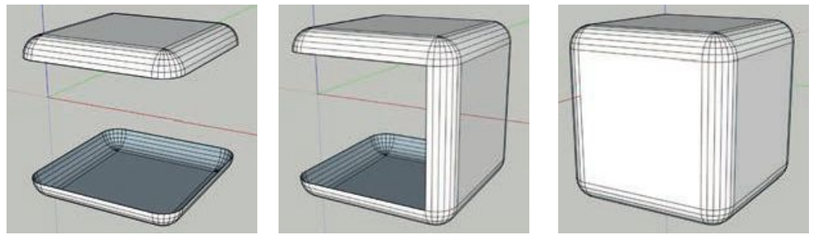
Możesz szybko skopiować element, naciskając klawisz **Ctrl** (nie przytrzymuj go) i przesuwając ten element narzędziem **Move**.

Aby utworzyć kolejne zaokrąglone krawędzie kostki, a potem całą pojedynczą ściankę, powtarzamy kopiowanie, lustrzane odbicie i łączenie wierzchołków. Tym razem jednak wykonujemy te czynności dla całego tworzonego elementu. Potem usuwamy niepotrzebne ścianki (patrząc od spodu), które powstały w wyniku łączenia wierzchołków (rys. 9.17).



Rys. 9.17. Łączenie krawędzi i tworzenie pojedynczej ścianki

Powtórzymy powyższe operacje dla całego elementu (tym razem jest to już górna ścianka kostki). Otrzymamy sześcian, w którym wierzchołki i krawędzie będą zaokrąglone (rys. 9.18).

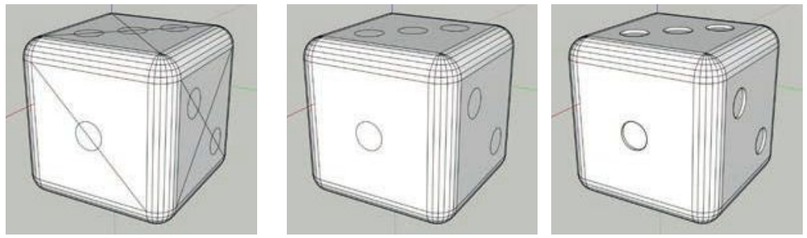


Rys. 9.18. Kopiowanie ścianki i budowanie kostki

Kolejny etap to oczka na ściankach kostki. Korzystając z linii pomocniczych, które możemy dodawać i usuwać w razie potrzeby, rysujemy na ściankach jednakowe koła i wciskamy je narzędziem **Push/Pull** nieznacznie do środka (rys. 9.19, s. 124).

Dobra rada

Kiedy już wciśniesz oczko na jakąś głębokość, możesz wcisnąć pozostałe oczka na taką samą głębokość, klikając w nie dwukrotnie.



Rys. 9.19. Dodawanie oczek na kostce

Ostatni etap to pokolorowanie wszystkich ścianek i wygładzenie modelu za pomocą narzędzia **Soften Coplanar Edges** w oknie **Soften/Smooth**.

Aby otrzymać końcowy efekt, połączyliśmy kilka narzędzi i technik pracy w programie SketchUp. W wielu przypadkach najważniejszy jest jednak pomysł na wykonanie modelu. Zwykle daną rzecz da się zrobić na wiele sposobów, dlatego zawsze na początku pracy warto zastanowić się nad różnymi możliwościami wykonania zadania.

Ćwiczenie 4

W planszowych grach RPG wykorzystuje się kości o różnej liczbie ścianek – od 4 do nawet 20. Poszukaj w Internecie zdjęć takich kości i zaprojektuj kość, która ma 4 ścianki.

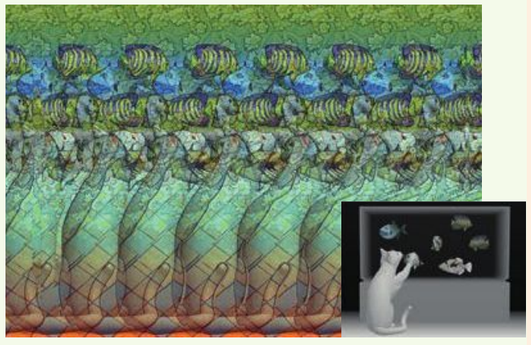
Zapamiętaj

Modelowanie obiektów 3D wymaga pracy w trójwymiarowym układzie współrzędnych. Elementy tych obiektów składamy z prostych kształtów, którym możemy nadawać kolor i fakturę. Jednym ze sposobów rysowania kształtów 3D jest również obracanie kształtu 2D wokół określonej osi.

A to ciekawe

Obraz 3D w obrazie 2D

Autostereogramy to dwuwymiarowe obrazy z ukrytymi obrazami trójwymiarowymi. Na pierwszy rzut oka większy z obrazów zamieszczonych obok nie przedstawia konkretnych obiektów. Jednak kiedy spojrzymy na ten obraz z bliska, a potem powoli będziemy oddalać wzrok, nie zmieniając ustawienia gałek ocznych, zobaczymy ukryte trójwymiarowe obrazy. To, co jest ukryte na większym obrazie, znajdziesz na mniejszym obrazku.



Zadania

* **1** Wykonaj trójwymiarowy model kostki Rubika:
- **a.** ułożonej,
- **b.** nieułożonej.

* **2** Wykonaj przestrzenny model dowolnej kartki okolicznościowej.

* **3** W edytorze grafiki trójwymiarowej przygotuj model 3D wazonu.

* **4** Zaprojektuj model 3D monitora komputerowego.

* **5** Wykonaj uproszczony model 3D sześciennej kostki do gry (na sześcianie bez zaokrąglonych krawędzi). Poprawnie rozmieść wszystkie oczka.

** **6** W edytorze grafiki trójwymiarowej wykonaj model klawiatury komputerowej. Wykorzystaj odpowiednie narzędzie do umieszczenia na przyciskach liter i znaków.

** **7** Zaprojektuj trójwymiarowy model zegara:
- **a.** analogowego,
- **b.** cyfrowego.

** **8** Przygotuj model 3D planszy i pionków do gry w warcaby. Zaprojektuj bryłę pionka.

** **9** Stwórz model globusa na podstawie zdjęcia znalezionego w Internecie.

** **10** Zrób przestrzenny model dowolnej cząsteczki chemicznej. Możesz poszukać przykładów w Internecie lub w podręczniku do chemii.

*** **11** Wykonaj model 3D stanowiska komputerowego. Uwzględnij w nim m.in. takie elementy jak mysz komputerowa i dwa głośniki.

*** **12** Wykonaj trójwymiarowy model dowolnego budynku. Przygotuj wcześniej materiał fotograficzny, który pomoże ci w modelowaniu bryły.

*** **13** Przygotuj w programie do grafiki 3D kompletne klasyczne szachy, czyli pionki oraz szachownicę.

*** **14** Zrób kilka modeli 3D obiektów z wybranej przez siebie gry komputerowej.

*** **15** W jednym pliku wykonaj modele różnych klocków (np. Lego). Pokoloruj je i powiel w liczbie pozwalającej na ułożenie dowolnego prostego zestawu.

F

10 Druk 3D

Drukarki 3D pozwalają w domowym lub szkolnym warsztacie realizować projekty, które do niedawna wymagały profesjonalnego sprzętu, dużych nakładów finansowych oraz niemałych umiejętności. Dzięki takim drukarkom szybko można przygotować prototyp swojego wynalazku lub wydrukować część, która się zepsuła, a której pozyskanie w inny sposób byłoby trudne. To także doskonałe narzędzie dla majsterkowiczów i modelarzy. Ale nie tylko. Możliwości druku 3D ogranicza dzisiaj właściwie tylko nasza wyobraźnia.

Ważne w tym temacie

- Rodzaje drukarek 3D
- Przygotowanie drukarki
- Materiały drukarskie
- Przygotowanie obiektu 3D do wydruku
- Wydruk i jego obróbka

Nawet proste i tanie drukarki 3D do użycia w domu lub szkole pozwalają na precyzyjny druk z wykorzystaniem różnych materiałów. Drukarek 3D używa się obecnie w niemal każdej branży, nawet w medycynie, np. do przygotowania implantów, protez oraz tkanek do przeszczepów. Wybudowano już nawet domy z użyciem drukarek 3D (rys. 10.1). Gdy powstawał ten tekst, trwały eksperymenty z drukowaniem żywności. Być może, gdy to czytasz, można już kupić taką żywność. My zajmiemy się jednak wyłącznie drukiem 3D dostępnym w warunkach szkolnych i domowych.

Rys. 10.1. Drukarka 3D tworząca konstrukcję betonową

Ćwiczenie 1

Zastanów się, do czego może się przydać drukarka 3D w:
a. domu,
b. szkole.

10.1. Zasada działania drukarek 3D

Najpopularniejszy rodzaj druku 3D to **wydruk przyrostowy (addytywny)**. Polega na nanoszeniu niewielkich ilości materiału i tworzeniu w ten sposób kolejnych warstw drukowanego obiektu. Podstawowa technologia tego rodzaju nosi nazwę FDM (od ang. *Fused Deposition Modeling*) lub FFF (od ang. *Fused Filament Fabrication*). Obie nazwy oznaczają to samo.

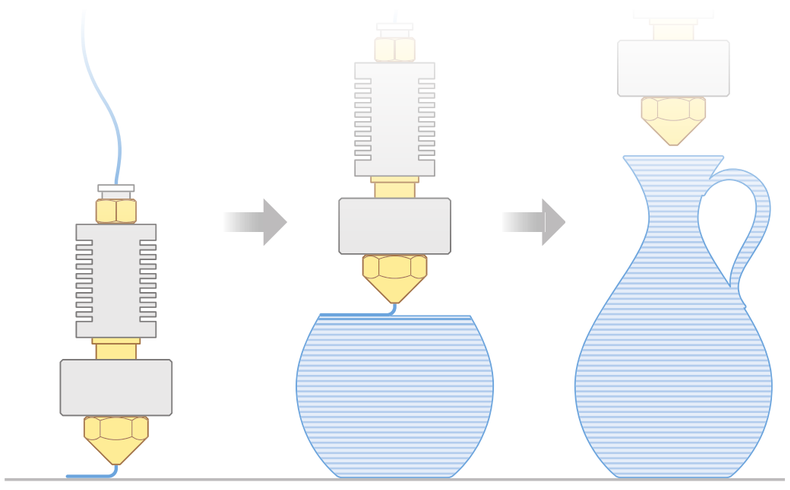
Wydruk przyrostowy (addytywny)

W najpopularniejszych domowych lub szkolnych drukarkach 3D materiał do druku, czyli **filament**, wygląda jak cienki drut lub żyłka z tworzywa. Jest on podawany do **głowicy drukującej**, gdzie jest podgrzewany. Następnie cienkie pasmo roztopionego tworzywa jest precyzyjnie nakładane za pomocą dyszy, warstwa po warstwie, na **stole roboczym**, który stanowi płaskie podłoże dla wydruku. Wydruk stopniowo stygnie i twardnieje – powstaje obiekt (rys. 10.2).

Filament

Głowica drukująca

Stół roboczy



Rys. 10.2. Drukowanie przyrostowe

Aby wydruk był zgodny z oczekiwaniami, ruch poszczególnych elementów drukarki 3D musi być precyzyjny. Dotyczy to m.in. ruchu głowicy nad stołem roboczym oraz ruchu stołu roboczego. Ważna jest także odpowiednia prędkość podawania filamentu. Przy zbyt małej prędkości będzie podawana niewystarczająca ilość filamentu i drukowana warstwa będzie np. za cienka. Z kolei za duża prędkość podawania filamentu sprawi, że będzie go za dużo i warstwa wyjdzie za gruba.

Warto wiedzieć

Temperatura podgrzanych filamentów waha się między 50°C a 300°C.

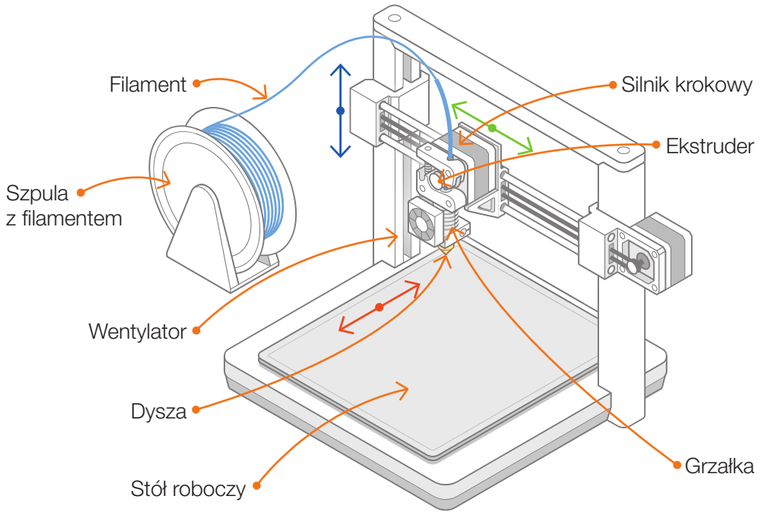
Warto wiedzieć

Głowica i stół roboczy muszą zapewnić drukowanie w przestrzeni trójwymiarowej.
W niektórych drukarkach głowica przesuwa się w kierunkach lewo/prawo, tył/przód, a stół roboczy – w górę i w dół. W innych modelach może być inaczej, np. stół roboczy przesuwa się w kierunku przód/tył. Wszystko zależy od modelu drukarki.

Precyzyjny ruch głowicy drukującej oraz stołu roboczego zapewniają **silniki krokowe**. Po otrzymaniu impulsu elektrycznego obracają się o ściśle ustalony kąt. Za transport (podawanie) filamentu odpowiada **ekstruder** (rys. 10.3).

Silnik krokowy

Ekstruder



Rys. 10.3. Schemat budowy drukarki 3D typu FDM

Materiały używane do druku 3D

Filamenty najczęściej są dostępne w formie szpul (rys. 10.4). Najpopularniejszym rodzajem takiego materiału jest PLA. Filament może występować w różnych kolorach. Może również imitować drewno, stal lub świecić w ciemności.

Rys. 10.4. Szpule z filamentem w różnych kolorach

Istnieje wiele rodzajów filamentów różniących się właściwościami i służących do tworzenia rozmaitych obiektów.

RODZAJE FILAMENTÓW

Istnieje wiele rodzajów filamentów do drukarek 3D. Różnią się właściwościami i służą do wydruku różnych modeli. Poniżej przedstawiamy najpopularniejsze z nich.

PLA

PLA to biopolimer pozyskiwany z roślin. Jest niedrogi i nadaje się niemal do wszystkich drukarek 3D. Jest idealny do prototypów, modeli koncepcyjnych, świetnie sprawdza się w warunkach szkolnych oraz domowych. Jego zaletą i jednocześnie wadą jest biodegradowalność – rozpada się m.in. pod wpływem wody i wysokiej temperatury. Modele z PLA są podatne na pękanie.

ABS

Produkty wykonane z ABS są bardzo trwałe i odporne na wysokie temperatury. Z materiału tego typu robi się często klocki dla dzieci. Użytkownicy drukarek 3D powinni jednak pamiętać, że filament ABS wymaga wysokiej temperatury drukowania, ma tendencję do wypaczania się podczas chłodzenia, a także wydziela intensywne, potencjalnie niebezpieczne opary.

TPU

Elementy wydrukowane z tego tworzywa po ściśnięciu i odkształceniu wracają do pierwotnego kształtu. Dodatkowo powierzchnia elementów wykonanych z TPU nie ściera się i jest odporna na działanie wielu substancji chemicznych. Materiał ten skutecznie wytłumia uderzenia oraz wstrząsy. Nadaje się do drukowania uszczelek, opon do modeli pojazdów itp.

PET-G

PET-G to połączenie materiału PET (materiał plastikowych butelek na napoje) z glikolem. Materiał ten cechuje się dużą elastycznością i wysoką wytrzymałością. Jest bezpieczny w użyciu (nie wydziela szkodliwych oparów) oraz nie rozpuszcza się w wodzie. Może mieć kontakt z produktami spożywczymi. Drukuje się też z niego części użytkowe i techniczne.

Nylon

Materiał lekki, bardzo trwały, odporny na ścieranie i nierozpuszczalny. Świetny do drukowania elementów eksploatacyjnych, ponieważ wytrzymuje duże obciążenia mechaniczne. Podczas pracy z nim powstają szkodliwe opary, dlatego wymaga korzystania z zamkniętej i filtrowanej komory roboczej lub druku w osobnym pomieszczeniu.

Filamenty pomocnicze

Zaliczamy do nich materiały, które pozwalają drukować podpory łatwe do usunięcia (np. przez rozpuszczenie w wodzie lub w odpowiedniej substancji chemicznej). Należą do nich m.in. PVA oraz HIPS. Wybierając odpowiedni filament do stworzenia podpór, trzeba wziąć pod uwagę to, czy będzie się on prawidłowo łączył z materiałem docelowym.

Warto wiedzieć

Filament PLA może być produkowany z naturalnych składników, takich jak skrobia kukurydziana lub ziemniaczana, trzcina cukrowa itp.

W drukarkach szkolnych i domowych najczęściej stosuje się filament PLA, ponieważ jest bezwonny i biodegradowalny, czyli ulega szybkiemu, przyjaznemu dla środowiska rozkładowi, np. pod wpływem wody lub temperatury. Może być wykorzystywany w większości drukarek oraz jest łatwy w użyciu. Nie wymaga specjalnych warunków, np. dobrej wentylacji czy stałej temperatury otoczenia podczas druku. Jest wytrzymały na rozciąganie, mało się kurczy i ma odpowiedni połysk. Nie jest jednak odporny na wilgoć, dlatego należy go przechowywać w suchym miejscu. W skrajnych przypadkach wilgoć może spowodować pękanie tego filamentu w trakcie drukowania lub już po wydrukowaniu obiektu. Do drukowania przedmiotów, które będą narażone na kontakt z wodą, trzeba więc wybrać inny rodzaj filamentu niż PLA.

Ćwiczenie 2

Zastanów się, jaki typ filamentu mógłby wykorzystać modelarz hobbysta do wydrukowania modelu łodzi motorowej. Sprawdź w wiarygodnych źródłach informacji, czy twoja odpowiedź jest poprawna.

Fotopolimeryzacja tworzyw

Wydruk przyrostowy
→ s. 127

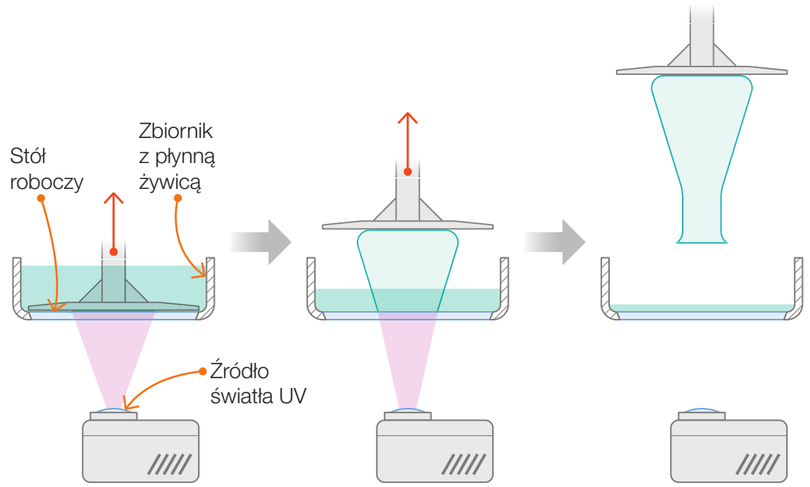
Fotopolimeryzacja

Istnieje też wiele innych odmian drukowania 3D. Większość jednak opiera się na wspomnianym **wydruku przyrostowym**. Popularną techniką wydruku przyrostowego stosowaną w domach i szkołach jest **fotopolimeryzacja** tworzyw, czyli utwardzanie płynnych żywic za pomocą promieni ultrafioletowych (UV). Proces takiego wydruku przebiega następująco:

1. W drukarce znajduje się zbiornik z płynną żywicą. Zbiornik jest przezroczysty, aby światło mogło przez niego przenikać. Stół roboczy zanurza się w zbiorniku z płynną żywicą i opuszcza tak, by od dna zbiornika dzieliła go bardzo cienka warstwa żywicy.
2. Źródło światła precyzyjnie oświetla stół roboczy od dołu i utwardza określone miejsca warstwy żywicy.
3. Po utwardzeniu danej warstwy stół przesuwa się minimalnie w górę wraz z utwardzonymi warstwami.
4. Źródło światła ponownie oświetla dolną część zbiornika i utwardza kolejną warstwę wydruku.

Powyższe operacje są powtarzane do momentu zakończenia drukowania (rys. 10.5, s. 131).

Po zakończeniu druku na drukarce żywicznej obiekt należy oczyścić, np. alkoholem izopropylowym (nazwa systematyczna: propan-2-ol). Następnie, aby dokończyć proces utwardzania, obiekt poddaje się dalszemu działaniu promieni UV, np. wkłada do specjalnej maszyny emitującej takie promieniowanie lub wystawia na działanie światła słonecznego. Trzeba ograniczyć kontakt fizyczny z płynną żywicą – pracować w rękawiczkach i traktować płynną żywicę jak substancję drażniącą.



Rys. 10.5. Wydruk 3D na drukarce żywicznej

Ćwiczenie 3

Poszukaj informacji o innych metodach druku 3D oraz używanych w nich materiałach. Sporządź notatkę dotyczącą wybranej metody. W notatce uwzględnij:

a. sposób powstawania wydruku,
b. przykłady materiów, z których może powstać wydruk, oraz ich właściwości,
c. obszary praktycznego zastosowania druku tą metodą.

Zapamiętaj

Wydruk obiektu na drukarce 3D metodą przyrostową polega na układaniu kolejnych jego warstw z wybranego materiału, np. z rozgrzanego filamentu lub utwardzonej żywicy.

10.2. Przygotowanie drukarki 3D do pracy

W drukarkach FDM jednymi z najczęściej wykonywanych czynności są załadunek filamentu oraz jego wymiana.

Załadunek filamentu

Załadunek filamentu krok po kroku:

1. Umieść szpulę filamentu w odpowiednim miejscu drukarki.
2. Wprowadź końcówkę żyłki filamentu rozwiniętego ze szpuli do **ekstrudera** przez kanał prowadzący.

Ekstruder
→ s. 128

3. W menu drukarki wybierz opcję **Załaduj filament**. Nazwa opcji może się różnić w zależności od modelu drukarki.
4. Poczekaj, aż drukarka potwierdzi zakończenie tego procesu.

Wymiana filamentu

Wymiana filamentu krok po kroku:

1. Wycofaj z głowicy drukującej dotychczas używany materiał – wybierz odpowiednią funkcję z menu drukarki.
2. Gdy żyłka zostanie usunięta z głowicy, wyjmij ją ręcznie z kanału prowadzącego, zwijając na szpuli. Wyjmij szpulę, by zrobić miejsce na nowy filament.
3. Zabezpiecz szpulę – włóż ją do szczelnego pojemnika z pochłaniaczem wilgoci.
4. Umieść nową szpulę filamentu w miejscu poprzedniej.

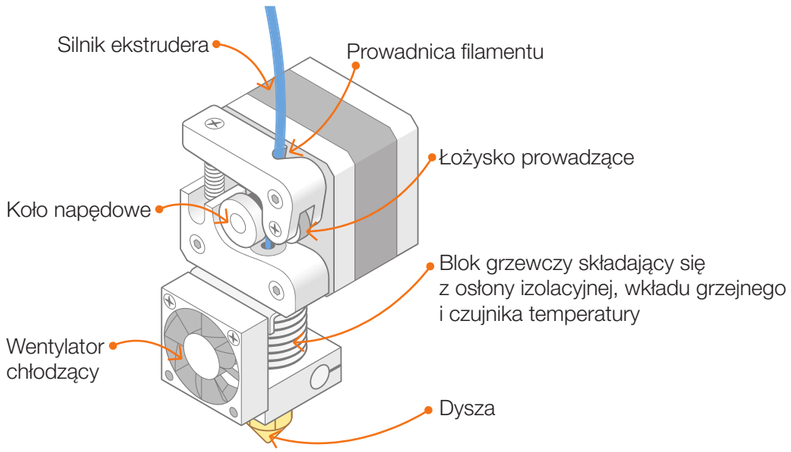
Zespół ekstrudera z poprawnie wprowadzonym filamentem pokazano na rysunku 10.6.

Dobra rada

Postępuj zgodnie z instrukcją dołączoną do sprzętu i nie używaj zbyt dużej siły, umieszczając filament w ekstruderze.

Warto wiedzieć

Część drukarek 3D pozwala na użycie systemów AMS (ang. *Automatic Material System*), dzięki którym można automatycznie drukować z wielu materiałów o różnych kolorach lub właściwościach. Wówczas wymiana filamentu sprowadza się do umieszczenia go zgodnie z instrukcją w automatycznym podajniku.



Rys. 10.6. Zespół ekstrudera z wprowadzonym filamentem

Przygotowanie stołu drukarki

Kalibracja stołu drukarki 3D

Kalibracja stołu drukarki 3D polega na dokładnym ustawieniu jego poziomu oraz odległości między blatem a głowicą drukującą. Jest to bardzo ważny proces, który wpływa na jakość wydruku.

Niewłaściwa odległość między stołem a głowicą osłabia przyczepność wydruku do stołu. To z kolei może sprawić, że obiekt będzie się przesuwał lub wyginał w trakcie drukowania. Brak właściwej kalibracji także powoduje odkształcenia i zdeformowanie części wydruków. Bywa, że takie obiekty są mniej funkcjonalne i nie wyglądają ładnie. W skrajnych przypadkach nieprawidłowa kalibracja stołu może doprowadzić do uszkodzenia drukarki 3D. Naprawa bywa kosztowna, a czasem wręcz niemożliwa, i wówczas konieczna jest wymiana sprzętu.

Kalibracja stołu drukarki 3D powinna być wykonywana regularnie. Szczególnie po przeniesieniu go w inne miejsce oraz gdy pojawiają się problemy z wydrukiem. Na skutek przenoszenia, a także zwykłego użytkowania stół może się rozkalibrować. Systematyczna kalibracja stołu zapobiega problemom związanym z jego nieprawidłowym ustawieniem.

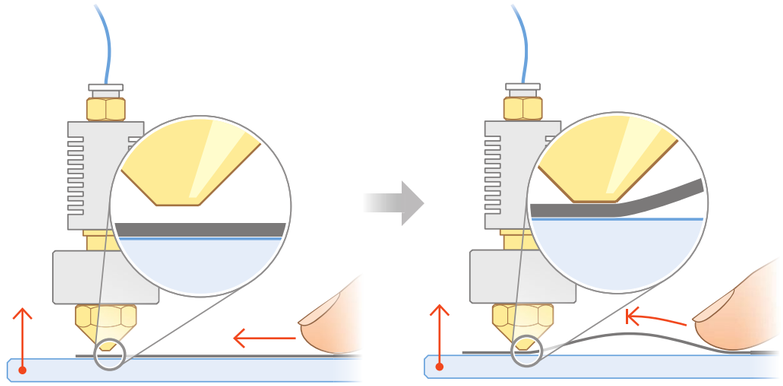
Dobra rada

Kalibrując stół drukarki, pamiętaj, że powinna ona stać na stabilnym i wypoziomowanym podłożu. Dzięki temu nie będzie narażona na potrącenia lub przesunięcia podczas pracy.

Sposób kalibracji stołu drukarki 3D zależy od jej modelu. Przed rozpoczęciem tego procesu należy zapoznać się z instrukcją urządzenia. Modele do zastosowań profesjonalnych mają specjalne czujniki i funkcje umożliwiające automatyczną kalibrację. Wystarczy zaczekać na zakończenie procesu. W przypadku modeli bez automatycznej kalibracji trzeba przeprowadzać ją ręcznie. Zazwyczaj proces rozpoczynamy od wybrania odpowiedniej opcji w menu drukarki. Następnie drukarka automatycznie przemieszcza głowicę nad stół w kilku wyznaczonych punktach. W każdym z tych punktów musimy użyć specjalnej karty kalibracyjnej lub kartki. Za pomocą śrub lub pokręteł trzeba ustawić głowicę tak, aby odległość między nią a stołem pozwoliła na przemieszczanie przez nas z małym oporem karty lub kawałka papieru (rys. 10.7).

Dobra rada

Posmarowanie stołu zwykłym klejem w sztyfcie zmniejsza ryzyko przesunięcia się obiektu podczas druku. Łatwiej będzie też dzięki temu zdjąć obiekt z platformy. Należy jednak regularnie usuwać ze stołu resztki kleju.



Rys. 10.7. Ręczna kalibracja stołu z użyciem kartki: ilustracja z lewej – zbyt duża odległość między dyszą a stołem, ilustracja z prawej – zbyt mała odległość między nimi

Konserwacja drukarki

Należy pamiętać o regularnym czyszczeniu elementów drukarki. Najlepiej używać specjalnych szczotek dołączanych przez producenta i przestrzegać zaleceń z instrukcji. Pozwoli to dłużej cieszyć się dobrze działającym urządzeniem i zmniejszyć ryzyko nieudanych prób wydruku.

A to ciekawe

Czy astronauci będą drukować jedzenie?

NASA rozwija technologię, która umożliwi astronautom drukowanie żywności w kosmosie. Zamiast tradycyjnych posiłków zabiorą ze sobą specjalne kartridże ze składnikami w proszku, które drukarka zamieni w gotowe potrawy. Naukowcy opracowują przepisy, które zapewnią bezpieczeństwo i odpowiednią wartość odżywczą drukowanego jedzenia.

Zapamiętaj

Przygotowanie drukarki do pracy polega na skalibrowaniu stołu roboczego oraz załadowaniu filamentu do urządzenia.

10.3. Przygotowanie wydruku obiektu 3D

Kiedy chcemy wydrukować dokument na zwykłej drukarce atramentowej lub laserowej, zaczynamy od wyboru kilku parametrów. Określamy jakość wydruku, czy ma to być drukowanie w kolorze czy w odcieniach szarości, jednostronne czy dwustronne. Możemy też wydruk przeskalować i wydrukować kilka stron na jednej kartce formatu A4.

Rozdzielczość wydruku 3D

W podobny sposób musimy przygotować wydruk 3D. Należy m.in. ustalić takie parametry jak **rozdzielczość wydruku 3D** (czyli np. wysokość pojedynczej warstwy), temperatura druku, prędkość drukowania i gęstość wypełnienia filamentem. Wszystkie one mają wpływ na jakość i funkcjonalność wydrukowanego obiektu.

Slicer

Do przygotowania wydruku służą programy typu **slicer** (dosł. krajalnica), które dzielą model 3D na warstwy możliwe do wydrukowania. Niektórzy producenci drukarek dołączają do nich własne oprogramowanie tego typu. W przypadku innych drukarek można skorzystać z bezpłatnych aplikacji. Zawsze warto jednak sprawdzić, czy dany program będzie dobrze współpracował z naszą drukarką (tab. 10.1).

Warto wiedzieć

Producenci drukarek 3D zwykle dostarczają oprogramowanie do obsługi swoich urządzeń oraz materiałów eksploatacyjnych (np. filamentów). Jednak w darmowych wersjach oprogramowania też można znaleźć ustawienia dla popularnych drukarek i korzystać z plików konfiguracyjnych tworzonych przez społeczność twórców w Internecie. Predefiniowane ustawienia ułatwiają rozpoczęcie pracy z drukiem 3D.

Dobra rada

Ciekawe informacje i porady dotyczące druku 3D możesz znaleźć na portalach społecznościowych i forach tematycznych poświęconych temu zagadnieniu.

Nazwa programu	Adres strony producenta	Informacje o programie
Cura	https://ultimaker.com/software/ultimaker-cura/	bezpłatny
Slic3r	https://slic3r.org	bezpłatny
PrusaSlicer	https://www.prusa3d.com/pl/strona/prusaslicer_424/	bezpłatny
FlashPrint	https://flashforge.com/blogs/download-software	do drukarek Flashforge
Z-Suite	https://support.zortrax.com/downloads/	do drukarek firmy Zortrax
Bambu Studio	https://bambulab.com/en-eu/download/studio	do drukarek BambuLab

Tabela 10.1. Przykłady programów typu slicer

Przygotowanie modelu 3D do druku

Aby przygotować model do druku, należy zapisać go w formacie zrozumiałym dla programu typu slicer. Często używanymi formatami są STL oraz 3MF. Chociaż oba służą do tego samego celu – opisania geometrii modelu 3D – robią to w inny sposób.

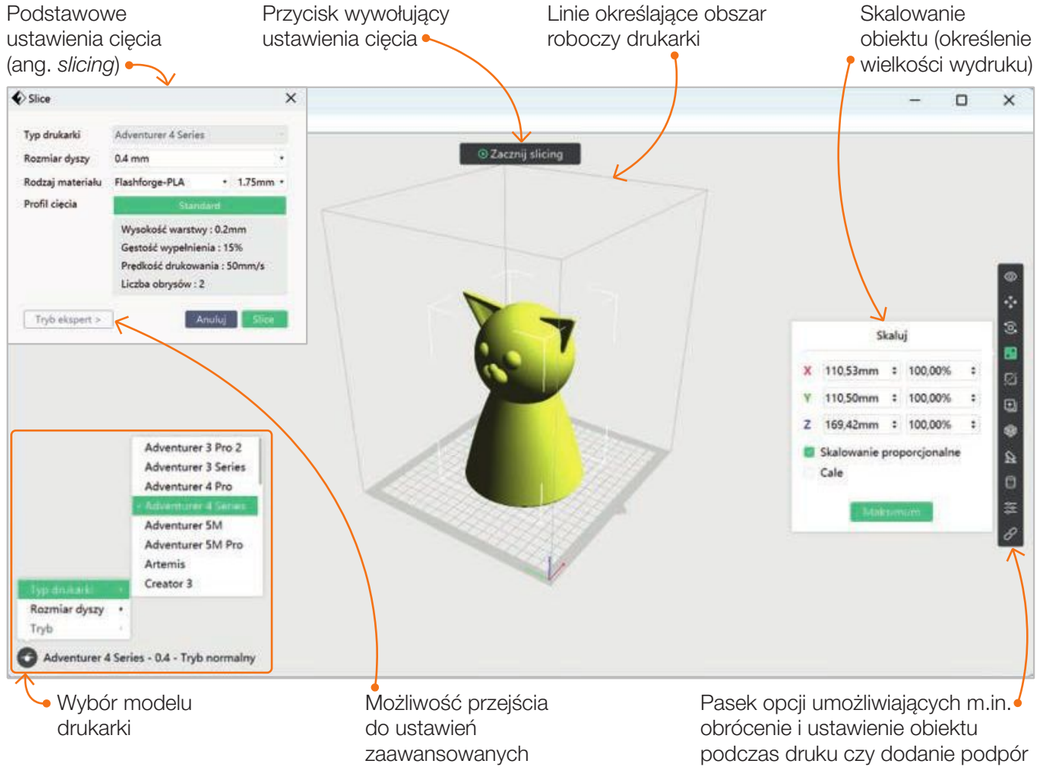
Format STL (od ang. *Standard Tessellation Language*) opisuje jedynie kształt obiektu. Nie zawiera informacji na temat jego innych cech, jak np. kolor, materiał, faktura powierzchni. Nie sprawdzi się przy wydruku kolorowym lub z różnych materiałów.

Format 3MF (od ang. *3D Manufacturing Format*) pozwala zapisać różne informacje na temat obiektu 3D dotyczące, poza kształtem, także koloru, materiału, faktury czy zastosowanej jednostki miary. Dzięki temu jest popularny w profesjonalnych zastosowaniach druku 3D.

Po zapisaniu projektu w wybranym formacie powinniśmy zaimportować model do programu umożliwiającego pocięcie go na warstwy i przygotowanie do wydruku. W przykładzie posłużymy się programem FlashPrint przeznaczonym do drukarek Flashforge. Po uruchomieniu programu wybieramy drukarkę, na której zrobimy wydruk. Dzięki temu część parametrów i ograniczeń, takich jak przestrzeń robocza drukarki czy rozmiar dyszy, zostanie automatycznie wczytana do programu (rys. 10.8).

Dobra rada

Dobierz format zapisu modelu do rodzaju wydruku 3D, który chcesz przygotować. Informacje na temat formatów innych niż opisane w temacie znajdziesz np. w Internecie. Możesz też zapytać o te formaty sztuczną inteligencję.



Rys. 10.8. Opcje programu typu slicer na przykładzie aplikacji FlashPrint

W takim przypadku możemy ograniczyć się do wyboru podstawowych parametrów technicznych wydruku. Najważniejszym z nich jest rodzaj filamentu. Jeśli do programu wczytaliśmy ustawienia odpowiadające naszej drukarce, inne parametry, takie jak wielkość dyszy drukującej i obszaru roboczego, możemy pozostawić jako domyślne.

Kilka parametrów powinniśmy ustalić przed każdym drukowaniem. Dzięki temu wydruk będzie zgodny z naszymi oczekiwaniami, szybszy, a czasem w ogóle możliwy.

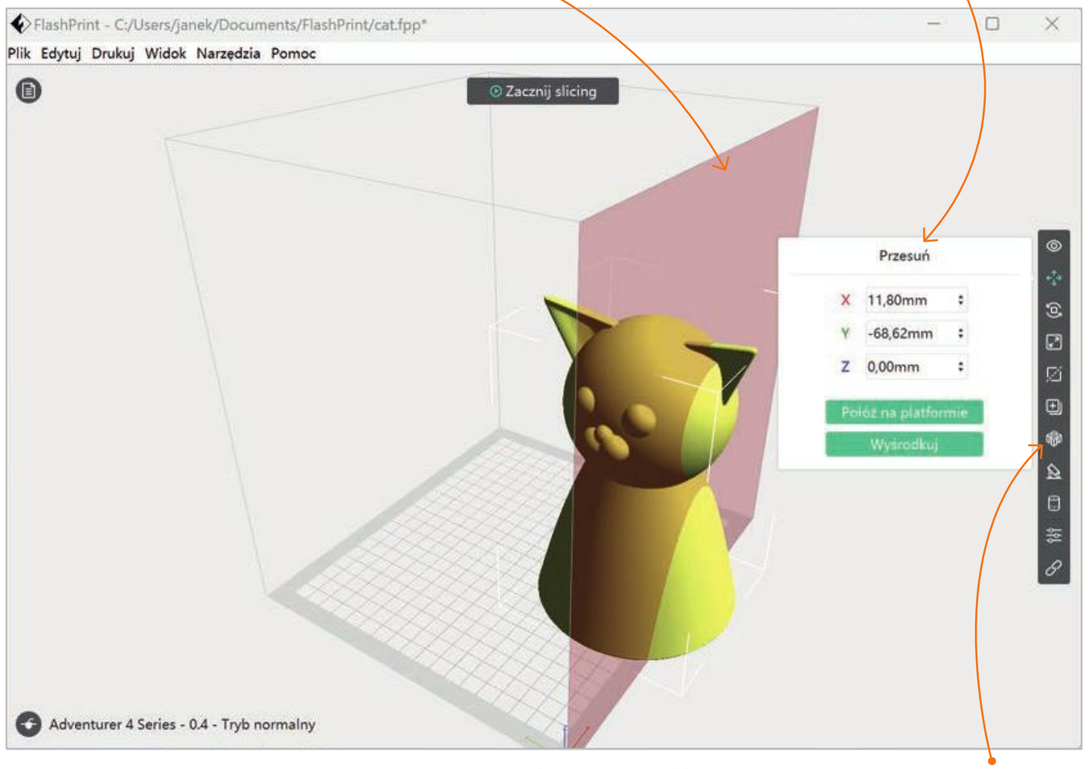
- **Skalowanie obiektu.** Za pomocą tej opcji możemy ustalić, jakiej wielkości obiekt chcemy wydrukować. Możemy dostosować wielkość modelu na jeden z dwóch sposobów: skalując wielkość wydruku w procentach względem obszaru roboczego drukarki albo określając (w centymetrach lub milimetrach) konkretną wysokość, szerokość lub głębokość drukowanego obiektu. W drugim przypadku musimy jednak zadbać o to, by przeskalowany obiekt nie wystawał poza obszar roboczy (rys. 10.9).
- **Przesunięcie (rozmieszczenie automatyczne).** Pozwala przesunąć obiekt w taki sposób, by znajdował się w granicach obszaru roboczego naszej drukarki. Jest to szczególnie przydatne, gdy zależy nam na uzyskaniu maksymalnie dużego wydruku lub chcemy jednocześnie wydrukować więcej niż jeden model (rys. 10.9).

Dobra rada

Jeśli chcesz uzyskać wydruk większy, niż pozwala na to przestrzeń robocza drukarki, podziel go na mniejsze fragmenty i wydrukuj je oddzielnie. Część programów umożliwia przeprowadzenie takiej operacji ręcznie lub automatycznie. Po wydrukowaniu fragmentów możesz je połączyć klejem przeznaczonym do materiału wykorzystanego podczas druku.

Ostrzeżenie, że obiekt jest ustawiony poza obszarem roboczym drukarki

Opcja **Przesuń** pozwala na ustawienie zaznaczonego obiektu w taki sposób, by znajdował się w obszarze roboczym i nie kolidował z innymi obiektami.



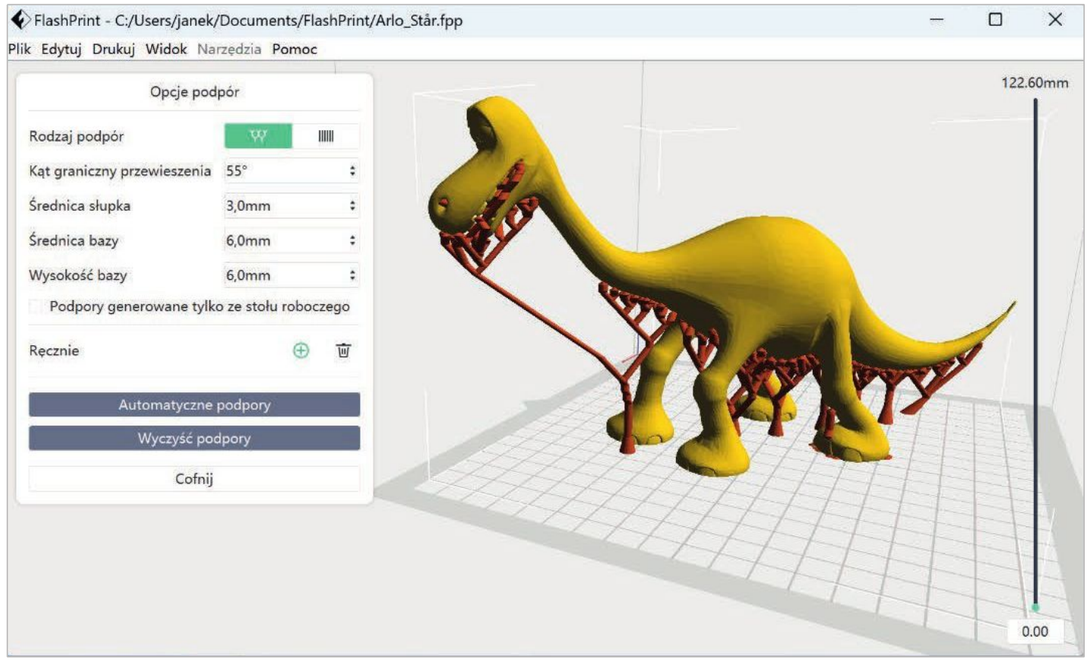
Opcja **Rozmieść automatycznie** pozwala na automatyczne ustawienie obiektów w obszarze roboczym.

Rys. 10.9. Pozycjonowanie modelu przed wydrukiem

- **Obrót modelu.** Także ta opcja ułatwia rozplanowanie drukowania w obszarze roboczym. Dodatkowo może pozwolić zminimalizować liczbę rusztowań (podpór, ang. *supports*) oraz podstawek (ang. *rafts*). Umożliwiają one wydrukowanie niektórych obiektów, ale wydłużają czas drukowania oraz zwiększają zużycie materiału.
- **Rusztowania (podpory).** Niektóre obiekty w trakcie drukowania wymagają podparcia, ponieważ nie mogą zostać wydrukowane „zawieszone w powietrzu". W takich przypadkach oprogramowanie wskazuje obszary modelu wymagające podpór, podświetlając je na czerwono. Należy wtedy utworzyć odpowiednie rusztowania wspierające. Można je zaprojektować samodzielnie albo wygenerować automatycznie. Wyróżnia się dwa podstawowe typy podpór: liniowe – wykorzystywane głównie w przypadku dużych, płaskich powierzchni, a także drzewiaste – przeznaczone do powierzchni o nieregularnych kształtach (ich zastosowanie pozwala na mniejsze zużycie filamentu niż w przypadku podpór liniowych – rys. 10.10).

Dobra rada

Warto zwrócić uwagę na automatycznie wygenerone podpory (rusztowania). Czasem już na pierwszy rzut oka widać, że mogą nie spełnić swojego zadania. Wówczas lepiej przejść na ręczny tryb ich dodawania i poprawić ich ustawienie tak, by zapewnić poprawny wydruk modelu.

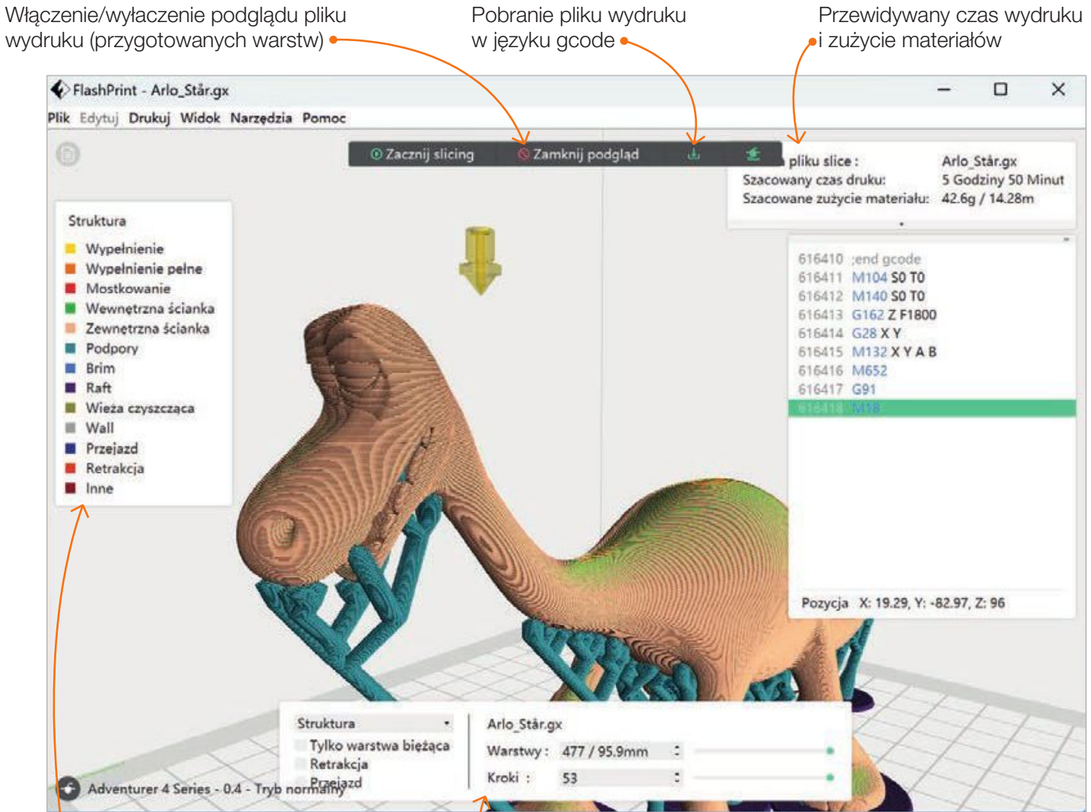


Rys. 10.10. Przykład wygenerowanych podpór (rusztowań) typu drzewiastego

- **Rafty, czyli podstawki.** Mogą zostać zdefiniowane automatycznie przez oprogramowania do cięcia, jeśli wybierzemy podstawową konfigurację wydruku. Możemy też zmienić parametry raftów w ustawieniach zaawansowanych. Dzięki raftom drukowany obiekt lepiej przylega do stołu roboczego, co zapobiega jego przesunięciu podczas drukowania. Podstawki, podobnie jak podpory, należy usunąć z gotowego obiektu.

Slicing

Po wprowadzeniu niezbędnych ustawień możemy rozpocząć proces **slicingu** (dosł. cięcia, krojenia). Po jego zakończeniu warto sprawdzić, na ile warstw został podzielony projekt, ile kroków (czyli ruchów) wykona głowica podczas ich drukowania oraz ile czasu zajmie cały proces. Na podglądzie (rys. 10.11) widać warstwy modelu, a różnymi kolorami oznaczono poszczególne elementy struktury, m.in. właściwy obiekt, podpory i podstawy. Możemy także sprawdzić przewidywany czas wydruku oraz zużycie filamentu. Większość programów do slicingu pozwala również zobaczyć, na ile warstw został podzielony projekt, i przyjrzeć się im, korzystając z suwaków. Odpowiednio obracając model i posługując się suwakami, możemy także zobaczyć oraz ocenić sposób i gęstość wypełnienia modelu.



Rys. 10.11. Podgląd projektu po slicingu (podziale na warstwy)

Przed wydrukiem warto zapisać projekt w edytowalnej formie odpowiadającej wybranemu oprogramowaniu. Jeśli wydruk się nie powiedzie, łatwo będzie do niego wrócić i zmienić ustawienia, które mogły mieć negatywny wpływ na efekt druku.

Ćwiczenie 4

W programie typu slicer przygotuj prosty model do wydruku. Zadbaj o odpowiednie ustawienie modelu, by zmieścił się w obszarze wydruku, a także miał poprawnie dobrane rusztowania (podpory) oraz podstawki.

Zapamiętaj

Aby przygotować model do druku, należy zapisać go np. w formacie STL lub 3MF. Potem trzeba ustawić model w programie typu slicer w taki sposób, by nie wykraczał poza obszar drukowania danej drukarki. Ostatni krok to dobranie optymalnego ustawienia rusztowań oraz podstawek.

10.4. Drukowanie modelu 3D

Jeżeli w programie skonfigurowaliśmy dostęp do naszej drukarki 3D, to po przygotowaniu modelu do wydruku możemy bezpośrednio z tego programu rozpocząć drukowanie. W przeciwnym razie powinniśmy pobrać plik w formacie gcode (rys. 10.11, s. 138). Plik ten będzie zawierał instrukcje dla drukarki dotyczące wydruku i jego ustawień w specjalnie do tego przeznaczonym języku. Każdy wiersz pliku w formacie gcode opisuje krok działania: ruch głowicy lub stołu roboczego, ustawienie temperatury wydruku lub stołu roboczego, włączenie albo wyłączenie określonej funkcji. Po pobraniu plik należy zamieścić w drukarce, np. za pomocą dysku USB.

Podczas drukowania mogą się pojawić trudności. W przypadku drukarki atramentowej lub laserowej kartka może się zakleszczyć albo może się pojawić konieczność wyczyszczenia głowicy drukującej. Z kolei w przypadku drukarki 3D obiekt może się np. przesunąć na stole roboczym podczas drukowania. Należy wtedy jeszcze raz skalibrować urządzenie i ponownie rozpocząć wydruk. Powinniśmy też systematycznie czyścić elementy drukarki 3D, żeby uniknąć nieudanych wydruków. Upewnijmy się również, że wystarczy nam filamentu do wydrukowania zaplanowanego obiektu.

Końcowa obróbka wydruku

Obiekty wydrukowane w technologii przyrostowej (FDM) są często gotowe do użytku zaraz po zdjęciu ze stołu roboczego. Elementy modeli wydrukowanych w częściach należy połączyć. Czasem wydruki wymagają dodatkowych zabiegów wykończeniowych, np. żeby lepiej wyglądały. Dodatkowa obróbka może być także konieczna ze względu na technikę druku, rodzaj materiału lub przeznaczenie modelu.

Warto wiedzieć

Mimo iż format gcode jest uniwersalny, różne urządzenia mogą wykorzystywać jego odmienne warianty. Plik wygenerowany dla konkretnego modelu drukarki niekoniecznie zostanie poprawnie zinterpretowany przez inne urządzenie.

Warto wiedzieć

Drukarki 3D w różny sposób mogą przyjmować pliki do druku: bezpośrednio z programu typu slicer, przez wi-fi lub pendrive (dysk USB). Warto zapoznać się z instrukcją drukarki, ponieważ część modeli obsługuje tylko dyski USB niewielkiej pojemności (np. do 32 GB).

Dobra rada

Drukowanie z niektórych rodzajów filamentu (np. ABS), a także obróbka wykonanych z nich modeli 3D wymagają ostrożności ze względu na wydzielające się podczas tych procesów nieprzyjemne i/lub szkodliwe opary. Dlatego zawsze należy dokładnie zapoznać się z instrukcją użytkowania danego materiału.

Do najczęściej wykonywanych zabiegów należą te podane poniżej.

- **Usuwanie struktur podporowych.** Podpory są niezbędne do drukowania zwisających lub skomplikowanych geometrycznie części modelu. Można je usuwać np. ręcznie (odłamywać) lub za pomocą narzędzi takich jak cążki czy szczypce. W przypadku materiałów rozpuszczalnych (np. PVA, HIPS) wydrukowany obiekt zanurza się w odpowiednim rozpuszczalniku (np. wodzie).
- **Szlifowanie i polerowanie.** Aby pozbyć się widocznych linii warstw i uzyskać gładką powierzchnię, wydrukowane obiekty można szlifować papierem ściernym o różnej gradacji. Polerowanie pastami polerskimi dodatkowo wygładza powierzchnię obiektu i nadaje jej połysk.
- **Wygładzanie chemiczne.** Niektóre tworzywa, np. ABS, można wygładzać oparami acetonu. Wystarczy na krótko umieścić wydrukowany obiekt w zamkniętym pojemniku z acetonem, tak aby opary mogły delikatnie rozpuścić jego zewnętrzną warstwę. Dzięki temu powierzchnia obiektu staje się gładka i błyszcząca.
- **Malowanie i klejenie.** Powierzchnię obiektów wydrukowanych z FDM można zagruntować i pomalować farbami, co pozwala uzyskać dowolny kolor i wybrane wykończenie. Większe modele można drukować w częściach, a następnie sklejać. Zarówno farby, jak i kleje należy dobrać do rodzaju materiału, z którego wydrukowano dany obiekt.
- **Uszczelnianie.** W przypadku modeli przeznaczonych do kontaktu z wodą konieczne może być zabezpieczenie powierzchni lub uszczelnienie łączeń części drukowanych osobno.

Ćwiczenie 5

Poszukaj informacji na temat tego, jakich rodzajów farb oraz klejów można używać do obiektów wydrukowanych z filamentów typu PLA, PET-G i ABS.

A to ciekawe

Części zamienne w kosmosie

Druk 3D sprawdza się w projektach typu *open-source hardware*, w których twórcy dzielą się projektami narzędzi wykorzystywanych na co dzień. To rozwiązanie jest szczególnie przydatne w miejscach, gdzie dostęp do produktów jest ograniczony. Umożliwia np. wydruk części zamiennych na stacjach kosmicznych – bez konieczności czekania na ich dostawę z Ziemi.

Zadania

* **1** Zastanów się, jakiego filamentu powinno się użyć do wydrukowania:
- **a.** modelu fontanny, który będzie miał kontakt z wodą,
- **b.** uszczelki do systemu nawadniania kwiatów na balkonie,
- **c.** pucharu, który będzie nagrodą w szkolnym konkursie.

Uzasadnij odpowiedź.

* **2** Wymień trzy przykłady zastosowania drukarek 3D w codziennym życiu.

* **3** Przygotuj do wydruku zaprojektowany przez siebie model pionka do warcabów. Umieść go odpowiednio na stole roboczym oraz zadbaj o właściwą wielkość wydrukowanego modelu. Zapisz projekt w postaci pliku w formacie gcode oraz edytowalnego projektu w wybranym przez siebie programie typu slicer.

* **4** Przygotuj do wydruku zaprojektowany przez siebie model kostki do gry. Zadbaj o odpowiednie ustawienie modelu na stole roboczym, tak by wydruk nie wymagał zastosowania podpór oraz by kostka miała wysokość 1 cm.

** **5** Przygotuj do wydruku opracowany przez siebie model szafki. Odpowiednio skonfiguruj ustawienie szafki w obszarze roboczym, aby:
- **a.** jej wydruk na pewno nie wymagał podpór,
- **b.** uzyskać maksymalną wielkość wydrukowanego modelu.

Zapisz model w postaci pliku w formacie gcode oraz edytowalnego projektu w wybranym przez siebie programie typu slicer.

** **6** Przygotuj poradnik dla koleżanek i kolegów z klasy, w którym opiszesz kilka najczęstszych problemów występujących podczas druku 3D oraz sposoby ich rozwiązania.

** **7** Wyjaśnij różnicę między drukiem FDM a drukiem z fotopolimeryzacji (żywicznym). Podaj przykłady, kiedy warto wybrać druk żywiczny, a kiedy FDM.

*** **8** Przygotuj do wydruku zaprojektowany model globusa. Wydruk powinien mieć wysokość 2 cm. Ustaw go w obszarze wydruku tak, by zminimalizować liczbę koniecznych podpór. Dobierz także w odpowiedni sposób gęstość i rodzaj wypełnienia, aby wydrukowany model był wytrzymały na upadek z wysokości biurka, ale jednocześnie aby dało się oszczędzić filament oraz zminimalizować czas wydruku. Zapisz projekt w postaci pliku w formacie gcode oraz edytowalnego projektu w wybranym przez siebie programie typu slicer.

*** **9** Znajdź informacje na temat tego, jakie filamenty wymagają zamkniętej komory drukowania i dlaczego tak jest.

*** **10** Podaj, jakie parametry gęstości wypełnienia należy ustawić dla wydruku modelu globusa o średnicy 2 cm, aby:
- **a.** obiekt był odporny na upadek,
- **b.** zużyć jak najmniej filamentu.

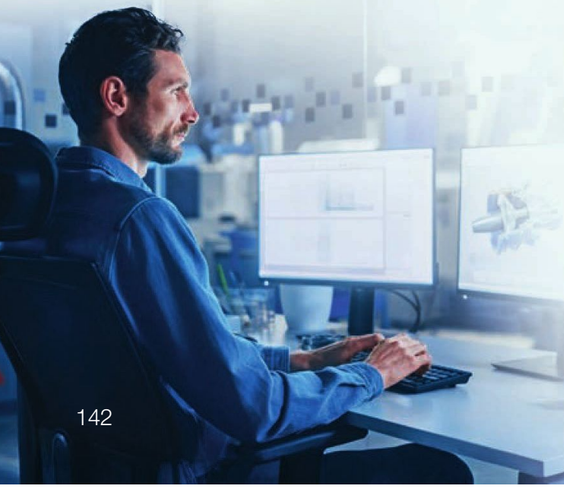
Projektowanie 3D

Wyobraź sobie, że...

...zawsze pasjonowały cię gry komputerowe, a teraz masz możliwość uczestniczenia w procesie ich tworzenia. Dzięki umiejętności **modelowania 3D** znajdujesz zatrudnienie w firmie, która tworzy i wydaje gry komputerowe. Możesz kreować postaci, przedmioty i całe sceny, dbając o **geometrię obiektów** oraz odpowiednie **materiały**, które nadają im realistyczny wygląd. Wszystkie te elementy zostaną wykorzystane w najnowszej grze 3D.

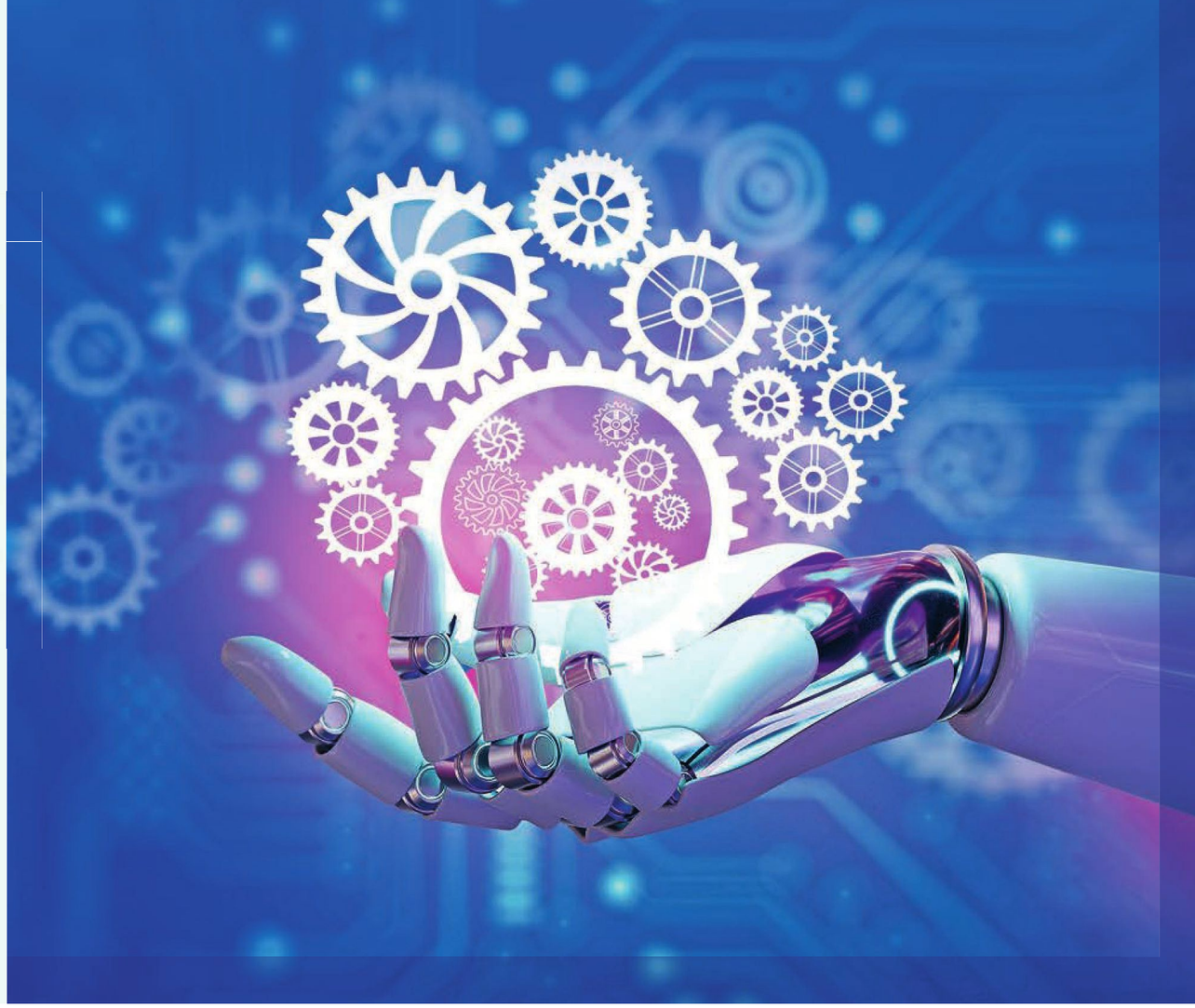
...jesteś architektem. Konkurencja w tym zawodzie jest duża. Jednak twoja firma wyróżnia się na tle innych – prezentuje klientom nie tylko wizualizacje budynków, lecz także makiety drukowane w technologii **wydruku przyrostowego**. Taki budynek można dokładnie obejrzeć z każdej strony, aby mieć lepsze wyobrażenie o jego ostatecznym wyglądzie. Przed rozpoczęciem procesu drukowania przygotowujesz **model 3D do druku**, a następnie wykorzystujesz **filament**, który pozwala uzyskać trwałe i szczegółowe odwzorowanie projektu.

...otrzymujesz w prezencie okulary do wirtualnej rzeczywistości. Dzięki umiejętności **modelowania 3D** możesz za pomocą odpowiednich programów tworzyć wirtualne **obiekty 3D**, wnętrza pomieszczeń i budynków. Pracując w **widoku trójwymiarowym**, tworzysz realistyczne **obrazy 3D**, które prezentujesz na stronie WWW. Dzięki temu łatwo znajdujesz pracę, w której tworzysz wirtualne spacery po zabytkach, muzeach czy innych obiektach użyteczności publicznej.



...jesteś początkującym grafikiem 3D. Pierwsze szkolne projekty z użyciem podstawowych narzędzi do **rysowania linii**, **obrotów obiektu**, **tworzenia złożonej geometrii 3D** i **wypełniania modelu teksturą** były dla ciebie pasjonującym doświadczeniem. Dlatego wybierasz studia na kierunku grafika komputerowa i kończysz specjalistyczne kursy grafiki 3D. Po studiach bez większego problemu znajdujesz pracę.

4. Rozwiązywanie problemów z wykorzystaniem komputera



F

11 Sztuczna inteligencja

Sztuczna inteligencja szybko wkracza do wielu dziedzin nauki i technologii. Nie powinno nas zatem dziwić, że jest różnie definiowana w zależności od kontekstu, przeznaczenia czy dziedziny nauki, której dotyczy. Samo pojęcie bycia sztucznym rozumiemy intuicyjnie. Wiemy np., że kwiat może być naturalny albo sztuczny. Jednak terminy *inteligentny* i *inteligencja* już takie oczywiste nie są. Wiemy, że inteligencję można mierzyć za pomocą testów i że zależy ona zarówno od indywidualnych predyspozycji człowieka, jak i od czynników genetycznych oraz środowiskowych. Jednak czym tak właściwie jest?

Ważne w tym temacie

- Definicja oraz początki sztucznej inteligencji
- Działanie pojedynczego sztucznego neuronu i uczenie go
- Przykładowe narzędzia AI do generowania tekstów i obrazów

Ćwiczenie 1

Przypomnij sobie sytuacje, w których użyto pojęcia *sztuczna inteligencja*. W jakim kontekście je przytoczono i jak je rozumiesz?

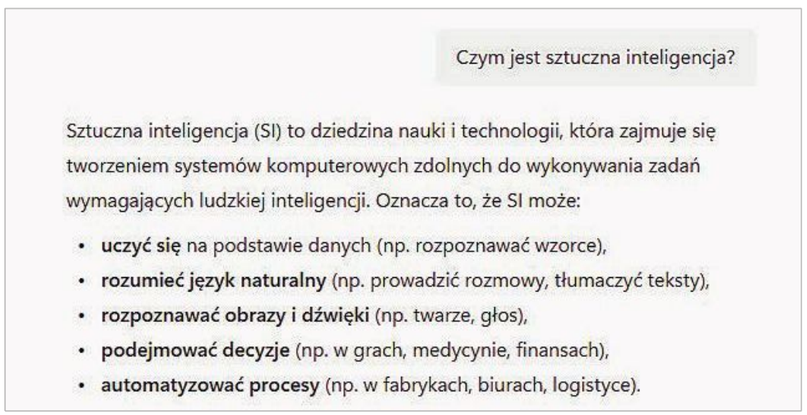
11.1. Czym jest sztuczna inteligencja?

Warto wiedzieć

Skrót AI, pochodzący od angielskiego zwrotu *artificial intelligence*, jest najbardziej znany i rozpowszechniony na całym świecie. W polskich opracowaniach spotkamy się też czasami z polskim skrótem SI.

Na pytanie, czym jest sztuczna inteligencja (SI, ang. AI – *artificial intelligence*), nie znajdziemy jednej odpowiedzi. Przeanalizujmy jednak kilka wyjaśnień pojęcia *sztuczna inteligencja* i sprawdźmy, co je łączy.

Jeśli o wytłumaczenie tego terminu poprosimy sztuczną inteligencję, otrzymamy np. taką odpowiedź jak ta na rysunku 11.1.



Rys. 11.1. Odpowiedź Microsoft Copilot na pytanie „Czym jest sztuczna inteligencja?”

W dokumencie z 2019 r., wydanym przez grupę niezależnych ekspertów Komisji Europejskiej, pojawia się następująca definicja:

> „Sztuczna inteligencja (AI) odnosi się do systemów, które wykazują inteligentne zachowanie polegające na analizowaniu otoczenia i podejmowaniu działań – z pewnym stopniem autonomii – w celu osiągnięcia określonych celów. Systemy oparte na AI mogą być wyłącznie oprogramowaniem działającym w świecie wirtualnym (np. asystenci głosowi, oprogramowanie do analizy obrazów, wyszukiwarki, systemy rozpoznawania mowy i twarzy) lub mogą być zintegrowane z urządzeniami sprzętowymi (np. zaawansowane roboty, autonomiczne samochody, drony czy aplikacje Internetu Rzeczy)”.
>
> Źródło: *https://digital-strategy.ec.europa.eu/en/library/definition-artificial-intelligence-main-capabilities-and-scientific-disciplines* [dostęp: 10.09.2025 r., tłumaczenie na język polski za pomocą DeepL Translator: *www.deepl.com*]

Na rządowym Portalu sztucznej inteligencji, w artykule *Polityka dla rozwoju sztucznej inteligencji w Polsce od roku 2020*, pojawia się następująca definicja:

> „Sztuczną inteligencję (ang. *Artificial Intelligence* – AI) próbuje się definiować jako dziedzinę wiedzy obejmującą m.in. sieci neuronowe, robotykę i tworzenie modeli zachowań inteligentnych oraz programów komputerowych symulujących te zachowania, włączając w to również uczenie maszynowe (ang. *machine learning*), głębokie uczenie (ang. *deep learning*) oraz uczenie wzmocnione (ang. *enforsement learning*)”.
>
> Źródło: *https://www.gov.pl/web/ai/polityka-dla-rozwoju-sztucznej-inteligencji-w-polsce-od-roku-2020* [dostęp: 10.09.2025 r.]

Widzimy, że w przedstawionych definicjach sztucznej inteligencji jest wiele wspólnych pojęć. W każdej z nich jest np. mowa o rozwiązaniach, których działanie naśladuje inteligencję człowieka. Te definicje jednoznacznie kierują nas w stronę informatyki, algorytmów i rozwiązywania problemów, w których tradycyjne algorytmy nie mają zastosowania lub wręcz nie wiemy, jak je napisać.

W tym temacie skupimy się na wybranych zastosowaniach sztucznej inteligencji. Opowiemy też o tym, jak system oparty na AI może się uczyć na podstawie danych. Wyjaśnimy, dlaczego sztuczna inteligencja staje się istotnym elementem współczesnego świata. Zaczniemy jednak od przyjrzenia się historii kształtowania się sztucznej inteligencji.

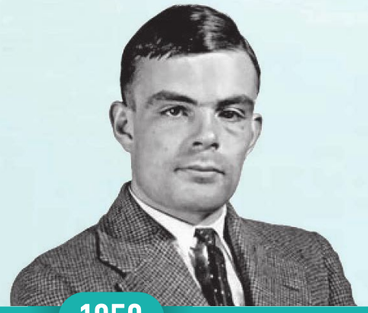
Warto wiedzieć

Przykład zagadnienia, które ciężko opisać w tradycyjny sposób, to rozpoznanie na zdjęciu lub filmie określonego obiektu, np. samochodu. Auta mają różne rodzaje nadwozi, różne kolory, wielkości, różnią się też markami i modelami.

Zapamiętaj

Nie istnieje jedna definicja pojęcia *sztuczna inteligencja*. Wszystkie proponowane wyjaśnienia łączy jednak motyw tworzenia rozwiązań, np. systemów komputerowych, których działanie naśladuje inteligencję człowieka.

HISTORIA SZTUCZNEJ INTELIGENCJI



1950

Alan Turing opublikował artykuł pt. *Computing Machinery and Intelligence*. Opisał w nim test zakładający, że jeden z rozmówców nie jest człowiekiem, lecz maszyną, która stara się naśladować zachowanie człowieka w rozmowie. Test Turinga sprawdza, w jakim stopniu maszyna potrafi upodobnić się do człowieka.

1952

Prof. Arthur Samuel, pracując z komputerami IBM serii 700, opracował program do gry w warcaby. Ten program samodzielnie podejmował decyzje dotyczące kolejnych ruchów. Z czasem uczył się na podstawie rozegranych partii.

1956

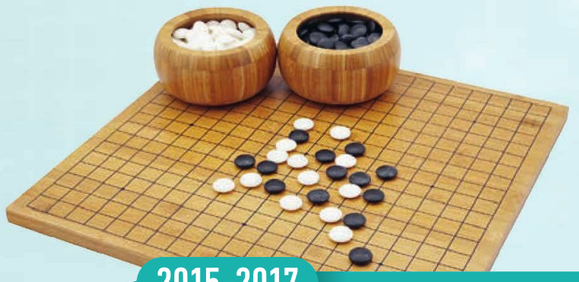
Podczas konferencji na Uniwersytecie w Dartmouth po raz pierwszy pojawiło się pojęcie sztucznej inteligencji.

1958

John McCarthy stworzył język programowania LISP (*List Processing*). Szybko zyskał on popularność jako narzędzie wykorzystywane do badania i rozwijania sztucznej inteligencji.

1966

Joseph Weizenbaum stworzył pierwszego chatbota – ELIZA. Wykorzystał on mechanizmy przetwarzania języka naturalnego (NLP), aby symulować psychoterapeutę w rozmowach z ludźmi.



2015–2017

AlphaGo, program komputerowy stworzony przez Google DeepMind, pokonał czołowych zawodników gry w grze planszowej Go, w tym mistrza świata Lee Sedola.

2011

Apple wprowadziło Siri, pierwszego popularnego asystenta wirtualnego.

1997

Deep Blue, komputer szachowy opracowany przez IBM, w meczu transmitowanym do wszystkich krajów pokonał ówczesnego mistrza świata – Garriego Kasparowa.

1979

Powstało *Association for the Advancement of Artificial Intelligence* (AAAI) – naukowe stowarzyszenie non profit, którego celem jest pogłębianie zrozumienia mechanizmów stojących za myśleniem oraz sposobów jego odwzorowania w maszynach.

2019

AlphaStar, program opracowany przez Google DeepMind, osiągnął poziom mistrzowski w grze wideo StarCraft II. Okazał się lepszy niż 99,8% graczy na świecie.

2020

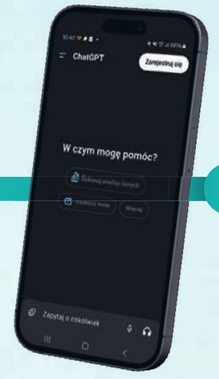
Firma OpenAI rozpoczęła testy wersji beta GPT-3 – modelu językowego służącego do generowania tekstów.

2021

OpenAI opracowało model DALL·E, który generuje obrazy na podstawie opisu tekstowego.

2023

OpenAI udostępniło model GPT-4, Microsoft wprowadził Copilota do swoich aplikacji, a Google zaprezentowało Gemini – nową generację modeli językowych.



2024

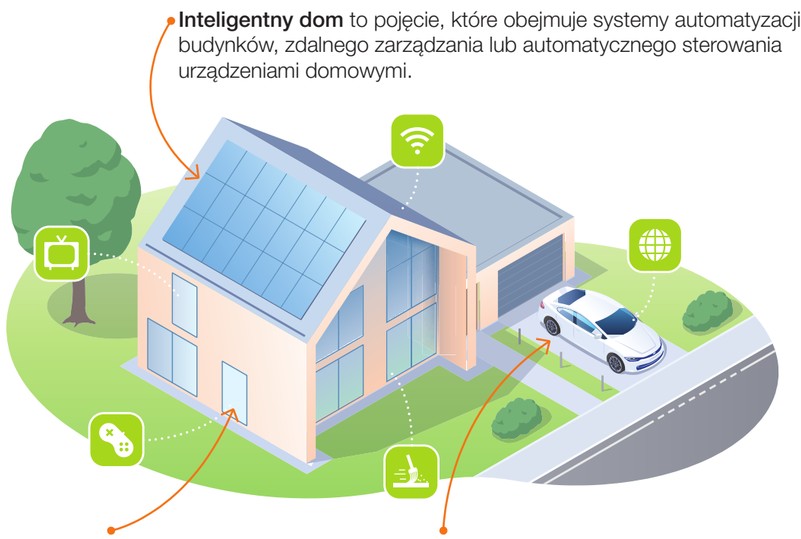
Open AI udostępniło ChatGPT with Canvas, który zmienia sposób rozmowy ze sztuczną inteligencją.

2025

OpenAI zaprezentowało Sora – pierwsze ogólnodostępne narzędzie AI do tworzenia realistycznych filmów z tekstu.

11.2. Poznawanie sztucznej inteligencji

Wiemy już, jak można zdefiniować sztuczną inteligencję. Czasem nawet nie zdajemy sobie sprawy, że z niej korzystamy. Spotykamy się również z przypisywaniem pojęcia inteligencji różnym rzeczom (rys. 11.2).



Rys. 11.2. Przykłady użycia określenia *inteligentny* w stosunku do przedmiotów

Często, kiedy korzystamy z różnych rozwiązań technologicznych i przedmiotów codziennego użytku, takich jak komputery czy smartfony, nie zdajemy sobie sprawy z tego, jak zaawansowane algorytmy odpowiadają za ich działanie. Te urządzenia mogą dostosowywać się do naszych zwyczajów, preferencji i codziennych aktywności. Czasem może nam się nawet wydawać, że nas obserwują i podsłuchują. Potocznie mówimy, że „uczą się nas" i każdego dnia poznają nas coraz lepiej. Jest to możliwe dzięki temu, że dostarczamy im wielu danych, na podstawie których dostosowują się do nas.

Uczymy sztuczną inteligencję

Na przykładzie aplikacji Google Teachable Machine zobaczymy, jak na podstawie danych (prostych obrazów) AI może się czegoś nauczyć.

Google Teachable Machine to platforma internetowa działająca w przeglądarce. Pozwala na tworzenie prostych modeli **uczenia maszynowego** (ang. *machine learning*). Polega ono na tym, że program komputerowy uczy się na podstawie otrzymanych danych.

Uczenie maszynowe

Platforma Google Teachable Machine nie wymaga umiejętności kodowania ani wiedzy o tym, jak działa AI. Dzięki temu jest odpowiednia dla użytkowników na każdym poziomie zaawansowania. Na tej platformie możemy eksperymentować z rozpoznawaniem obrazów (zdjęć, rysunków), dźwięków albo obrazów zarejestrowanych przez kamerę internetową.

Uczenie sztucznej inteligencji krok po kroku:

1. Dostarcz danych, na podstawie których model będzie się uczył.
2. Sprawdź, jakie wyniki zwraca model w procesie uczenia.
3. Oszacuj błędy popełniane przez model w procesie uczenia.
4. Na podstawie błędów skoryguj parametry modelu.
5. Powtarzaj kroki od 1 do 4, dopóki otrzymane wyniki nie spełniają oczekiwań.

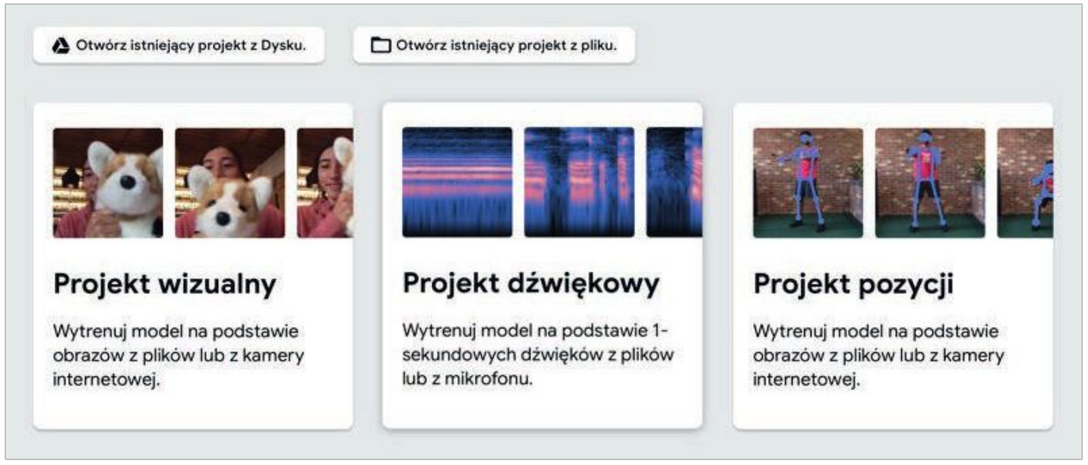
Uczenie sztucznej inteligencji

Na początku przechodzimy na stronę teachablemachine.withgoogle.com i wybieramy projekt. Do dyspozycji mamy trzy możliwości (rys. 11.3).

- **Projekt wizualny** – gdy chcemy stworzyć narzędzie do rozpoznawania obrazów pochodzących z kamery internetowej lub zapisanych w plikach.
- **Projekt dźwiękowy** – jeżeli interesuje nas rozpoznawanie dźwięków zapisanych w plikach albo pochodzących z mikrofonu.
- **Projekt pozycji** – podobny do projektu wizualnego, ale dotyczy rozpoznawania ustawienia ciała człowieka.

Warto wiedzieć

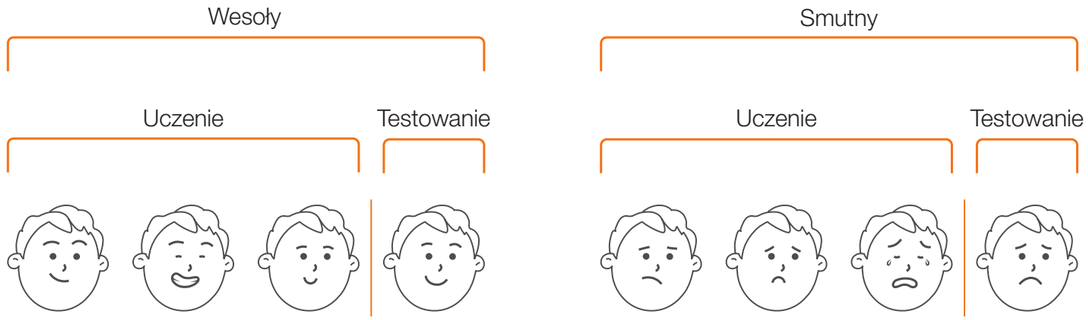
Modelem uczenia maszynowego może być algorytm, który uczy się wzorców z danych i wykorzystuje je do podejmowania decyzji lub przewidywania wyników.



Rys. 11.3. Rodzaje projektów dostępnych w Google Teachable Machine

Przetestujemy to narzędzie na przykładzie rozpoznawania obrazów. Korzystając z dowolnego edytora graficznego, np. programu Microsoft Paint, narysujemy twarze wyrażające radość lub smutek. Przygotujemy dwa zestawy obrazów w formacie JPG. Nasze obrazy będą bardzo proste i będzie ich tylko kilka. Aby uzyskać lepsze rezultaty, trzeba by przygotować znacznie więcej plików, i to dużo lepszej jakości.

Z każdego zestawu wybierzemy kilka obrazów, których nie użyjemy w procesie uczenia. Posłużą nam do testowania (rys. 11.4).

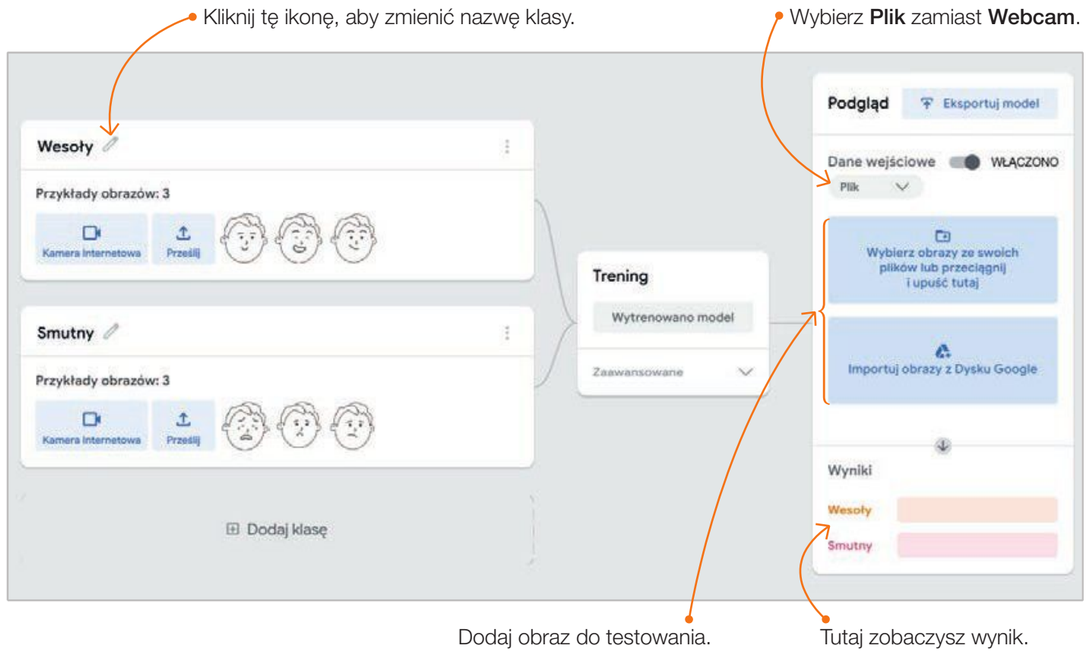


Rys. 11.4. Przykładowe rysunki do uczenia i testowania sztucznej inteligencji

Kiedy mamy już gotowe obrazy, czyli dane, których użyjemy do uczenia sztucznej inteligencji, wybieramy **Projekt wizualny**, a następnie **Standardowy model obrazu**. Ponieważ mamy tylko dwie kategorie obrazów (wesoły i smutny), czyli dwie klasy, nie musimy dodawać kolejnej. Nazwijmy klasy Wesoły i Smutny. Następnie za pomocą przycisków **Prześlij** dodajemy do każdej klasy odpowiednie obrazy – te, które przeznaczyliśmy do uczenia. Po dodaniu obrazów dla każdej klasy klikamy przycisk **Trenuj model** i czekamy na zakończenie uczenia. Po pojawieniu się komunikatu „Wytrenowano model" możemy przetestować skuteczność narzędzia (rys. 11.5).

Dobra rada

Jeśli chcesz wykorzystać więcej kategorii obrazów, dodaj kolejne, naciskając przycisk **Dodaj klasę**.



Rys. 11.5. Uczenie sztucznej inteligencji w Google Teachable Machine

Ćwiczenie 2

Utwórz nowy projekt wizualny w Google Teachable Machine. Przygotuj różne zdjęcia trzech gatunków zwierząt z podziałem na obrazy do trenowania i obrazy do testowania. Wytrenuj model i przetestuj jego działanie na zdjęciach, które nie były użyte do trenowania.

Opisany przykład pokazuje jedno z prostszych zastosowań AI – rozpoznawanie obrazów i przypisywanie ich do danej klasy. To proste narzędzie pozwala na rozpoznawanie i klasyfikowanie nie tylko obrazów, lecz także dźwięków.

Zapamiętaj

Narzędzia, które korzystają ze sztucznej inteligencji, potrzebują do nauki odpowiednich danych. Jakość i ilość tych danych to jedne z podstawowych parametrów decydujących o skuteczności działania tych narzędzi.

11.3. Sztuczny neuron i uczenie z nauczycielem

Na przykładzie prostej aplikacji internetowej poznaliśmy możliwości uczenia się sztucznej inteligencji. Warto się jednak zastanowić, jak to się dzieje, że program komputerowy może się czegoś uczyć, a potem wykorzystywać zdobytą wiedzę. Wyobraźmy sobie sytuację, w której to człowiek ma się czegoś nauczyć.

Gdy po raz pierwszy pokażemy dziecku jabłko, ale nie powiemy mu, jak ta rzecz się nazywa i do czego służy, to nie będzie ono wiedziało, co to jest. Ten nowy obiekt może być jednak przez nie kojarzony na podstawie cech, takich jak kolor i kształt, z czymś, co widziało w przeszłości. Nie zmienia to jednak faktu, że nazwa „jabłko" nie będzie mu znana.

A to ciekawe

Kucharz Flippy

Firma Miso Robotics opracowała robota o nazwie Flippy, który potrafi m.in. robić burgery. Ma on czujniki termiczne i kamery, dzięki którym może dokładnie kontrolować temperaturę oraz czas smażenia. Flippy analizuje warunki procesu i zawsze we właściwym momencie przewraca mięso, żeby było odpowiednio wysmażone.

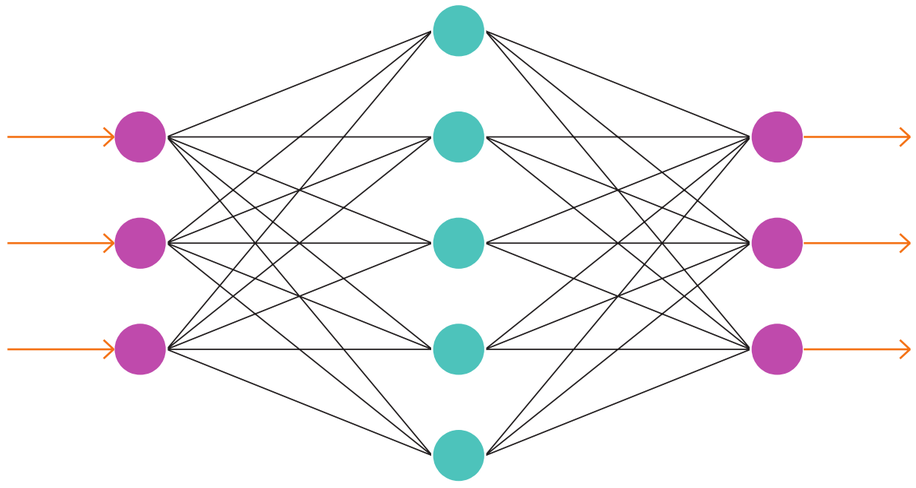
Wielokrotne pokazywanie dziecku różnych jabłek i nazywanie ich za każdym razem jabłkiem spowoduje, że dziecko nie tylko przyswoi sobie to pojęcie, lecz także będzie je kojarzyć z jabłkami różniącymi się od siebie. Dzięki temu będzie rozpoznawać jabłko za każdym razem, kiedy je zobaczy (rys. 11.6).

Rys. 11.6. Uczenie dziecka rozpoznawania jabłka

Mechanizm ten można w skrócie opisać jako relację ucznia (dziecko, które po raz pierwszy coś widzi) i nauczyciela (pokazującego i opisującego, co to jest). W tym procesie ważne są dane, które nauczyciel przekazuje uczniowi, sposób, w jaki to robi, oraz weryfikowanie i korygowanie przez nauczyciela procesu uczenia się.

U człowieka procesy związane z uczeniem się zachodzą w układzie nerwowym, a kluczową rolę odgrywa w nich mózg. Gdybyśmy mieli wskazać miejsce, w którym „mieszka" inteligencja, byłby to właśnie mózg – centrum sterowania całym naszym organizmem. Sztuczna inteligencja, z jaką mamy dziś do czynienia, opiera się głównie na **sztucznych sieciach neuronowych**, a więc modelach, dla których inspiracją były prawdziwe neurony w mózgu (rys. 11.7).

Sztuczna sieć neuronowa

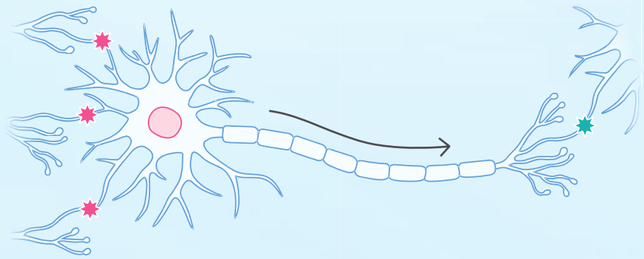


Rys. 11.7. Schemat sztucznej sieci neuronowej

MODEL SZTUCZNEGO NEURONU

Neuron jest najmniejszym elementem ludzkiego układu nerwowego i nie może funkcjonować samodzielnie. Działa dzięki połączeniom z innymi neuronami, tworząc złożoną sieć. Na podobnej zasadzie zaprojektowano sztuczny neuron.

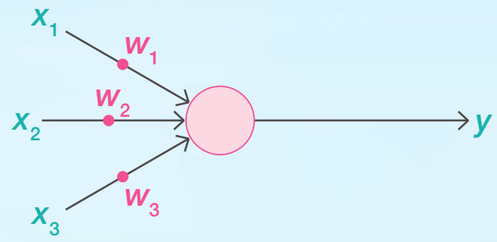
Neuron odbiera sygnały od innych neuronów w sieci. Wzmacnia je lub osłabia i przekazuje dalej do kolejnych neuronów.



Cechy wspólne neuronów typowego biologicznego i sztucznego

- Mogą mieć wiele wejść, ale mają tylko jedno wyjście.
- Sygnały wejściowe przekazywane są w jednym kierunku.
- Sygnał wyjściowy z jednego neuronu może być przekazywany do wielu kolejnych neuronów.

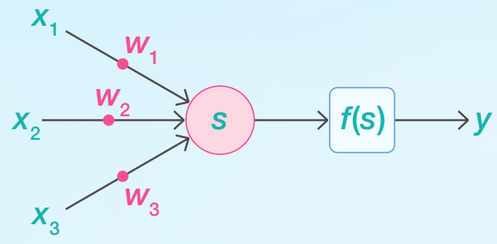
Sztuczny neuron również odbiera sygnały – w postaci liczb. Aby je wzmocnić lub osłabić, mnoży je przez odpowiadające im wagi. Następnie sumuje uzyskane iloczyny i zwraca wartość wyjściową, którą przekazuje dalej do kolejnych neuronów.



$$x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 = y$$

x_1, x_2, x_3 – sygnały wejściowe przekazywane do neuronu
w_1, w_2, w_3 – wagi, przez które neuron mnoży poszczególne sygnały
y – sygnał wyjściowy, czyli suma iloczynów sygnałów wejściowych i odpowiadających im wag

Aby neuron był w pełni funkcjonalny i działał jak przełącznik, decydując, czy otrzymane dane przekazać dalej, musi mieć funkcję aktywacji. Na przykład prosta funkcja aktywacji zwraca 0 lub 1. Gdy wyliczona suma $s < 0$, funkcja zwraca 0, czyli $y = 0$. W przeciwnym razie zwraca 1, czyli $y = 1$.



$$x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 = s$$
$$y = f(s)$$

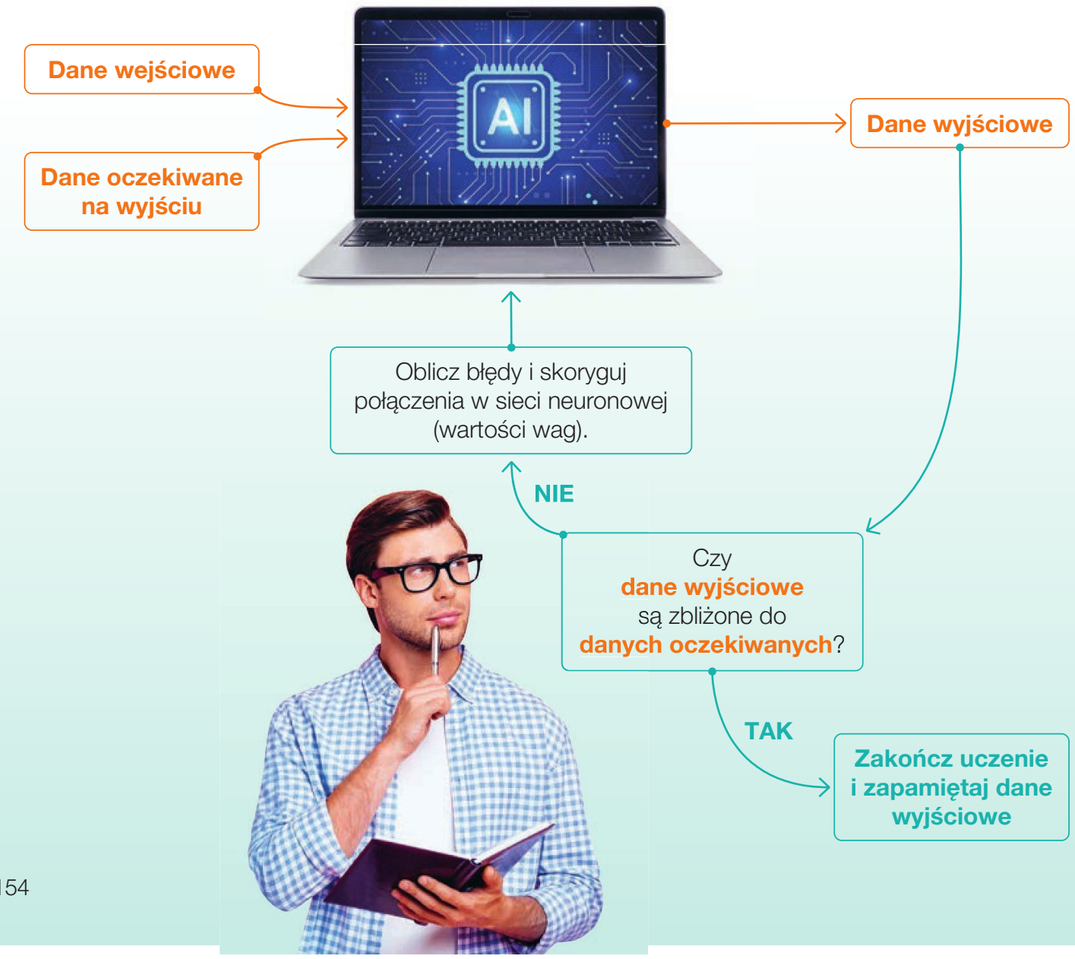
s – suma iloczynów sygnałów wejściowych i odpowiadających im wag
$f(s)$ – funkcja aktywacji neuronu
y – sygnał wyjściowy, czyli wynik działania funkcji $f(s)$

11.4. Na czym polega uczenie sztucznych neuronów?

Wiemy już, jak wygląda model sztucznego neuronu i w jaki sposób przetwarza on sygnały wejściowe. Jeśli odpowiednio połączymy wiele takich neuronów, powstanie sztuczna sieć neuronowa. Takie sieci uczą się, analizując duże ilości danych i dostosowując swoje wewnętrzne parametry, czyli wartości wag, aby coraz lepiej rozpoznawać wzorce i wykonywać zadania.

Proces uczenia z nauczycielem w przypadku sztucznych sieci neuronowych polega na tym, że sieć otrzymuje dane wejściowe jednocześnie z oczekiwanym dla tych danych wynikiem, oblicza swoją odpowiedź, a następnie porównuje ją z wartością oczekiwaną. Jeśli sieć pomyli się w obliczeniach, czyli odpowiedź będzie inna niż spodziewana, możemy wyznaczyć wartość błędu. Kiedy znamy błąd, możemy korygować wagi tak, by przy kolejnych próbach zmniejszać wartość tego błędu. Ten cykl powtarza się tak długo, aż sieć osiągnie zakładaną przez nas dokładność w wykonywaniu zadania.

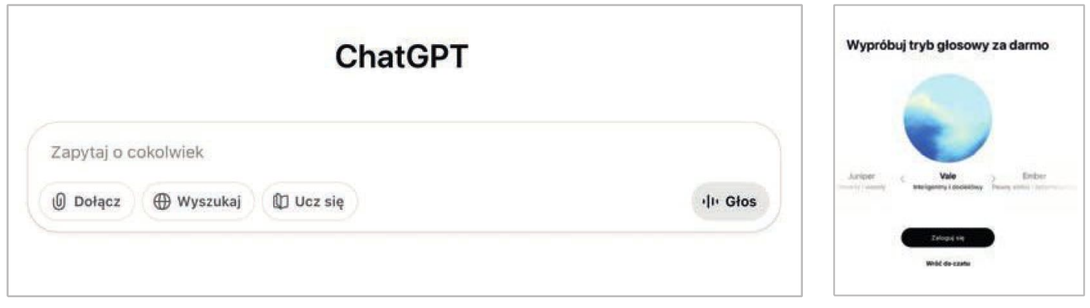
Proces uczenia sieci neuronowej



11.5. Praktyczne zastosowania sztucznej inteligencji

Jednymi z narzędzi często kojarzonych ze sztuczną inteligencją są **chatboty**, czyli programy prowadzące z użytkownikiem rozmowę zbliżoną do naturalnej. Przykładami takich narzędzi są ChatGPT, Copilot i Gemini. Można ich używać do uzyskiwania informacji, pomocy w nauce, pisania tekstów czy nawet rozwiązywania problemów technicznych. Pomagają realizować pomysły, tłumaczą teksty na różne języki, piszą kody źródłowe programów i wyjaśniają trudne zagadnienia w prosty sposób. Są to narzędzia, które są przydatne zarówno w pracy, jak i w nauce (rys. 11.8).

Chatbot



Rys. 11.8. Pytanie skierowane do chatbota GPT oraz jego odpowiedź

Polecenie, które ma wykonać system sztucznej inteligencji, nazywa się **promptem**. Aby uzyskać satysfakcjonującą odpowiedź, musi ono być precyzyjnie sformułowane. Jeśli prompt będzie zbyt ogólny lub niejasny, możemy otrzymać odpowiedź, która nie będzie odpowiadała naszym oczekiwaniom. Im bardziej konkretne polecenie, tym większa szansa na trafną odpowiedź. Zawsze możemy też doprecyzować polecenie albo poprosić o rozwinięcie lub wyjaśnienie udzielonej odpowiedzi.

Prompt

Ćwiczenie 3

Napisz prompt, dzięki któremu sztuczna inteligencja udzieli informacji o tym, czym jest budowla Czerwony Kamień. Sprawdź w wybranym chatbocie, jaką odpowiedź otrzymasz.

Chatboty naśladujące konkretne postaci

Model językowy to algorytm sztucznej inteligencji, który analizuje, rozumie i generuje tekst w języku naturalnym na podstawie wzorców i zależności wyuczonych na dużych zbiorach danych.

Na stronie character.ai możemy na wiele sposobów testować możliwości sztucznej inteligencji opartej na **modelach językowych**. Znajdziemy tutaj m.in. chatboty odpowiadające znanym postaciom. Możemy poczuć się, jakbyśmy rzeczywiście rozmawiali z tymi osobami (rys. 11.9).

Rys. 11.9. Rozmowa z Albertem Einsteinem na stronie character.ai

Możemy również przetestować w ten sposób swoje możliwości w rozmowach kwalifikacyjnych, zaplanować podróż, poćwiczyć wybrany język czy po prostu porozmawiać z chatbotem na dowolny temat.

Jednym z przykładów wykorzystania tego narzędzia jest też wirtualna gra typu Escape Room (rys. 11.10).

Rys. 11.10. Gra typu Escape Room na stronie character.ai

Ćwiczenie 4

Na stronie character.ai zadaj Napoleonowi Bonaparte pytanie dotyczące wygranych przez niego bitew. Zweryfikuj uzyskane odpowiedzi w innych dostępnych źródłach.

Generatory obrazów

Sztuczna inteligencja może też generować obrazy. Jednym z narzędzi, które to umożliwiają, jest DALL·E. Po opisaniu, jaki obraz chcemy otrzymać, program przygotowuje propozycje, które będzie można dalej dostosowywać (rys. 11.11). Możemy np. określić szczegóły dotyczące stylu czy kolorystyki. Im lepiej opiszemy, co chcemy uzyskać, tym większa szansa, że rezultat będzie zgodny z naszymi oczekiwaniami.

Rys. 11.11. Przykładowy prompt oraz uzyskane propozycje obrazów

Ćwiczenie 5

W wybranym narzędziu korzystającym ze sztucznej inteligencji wygeneruj obraz przedstawiający bohatera filmu lub książki.

Zapamiętaj

Prompt to polecenie, które ma wykonać system sztucznej inteligencji. Jeśli nie zostanie sformułowane precyzyjnie, odpowiedź może odbiegać od naszych oczekiwań. Im bardziej konkretny i jasny prompt sformułujemy, tym większa szansa, że otrzymamy satysfakcjonującą odpowiedź.

Zadania

* **1** W dwóch wybranych narzędziach korzystających ze sztucznej inteligencji wygeneruj obraz pokazujący, jak mógłby wyglądać idealny świat za 20 lat. Które z narzędzi twoim zdaniem lepiej poradziło sobie z zadaniem? Uzasadnij odpowiedź.

* **2** Wykonaj prezentację, w której przedstawisz po trzy zastosowania sztucznej inteligencji w dwóch wybranych dziedzinach nauki.

* **3** W dokumencie tekstowym zapisz trzy pytania, na które chcesz poznać odpowiedzi. Zdobądź odpowiedzi, wykorzystując jeden z dostępnych chatbotów. Następnie zapisz te odpowiedzi w utworzonym dokumencie. Sprawdź w wiarygodnych źródłach informacji, czy odpowiedzi udzielone przez chatbota są poprawne.

** **4** W wybranym narzędziu, które korzysta ze sztucznej inteligencji, wygeneruj obraz przedstawiający zachód słońca nad jeziorem. Obraz powinien być w takim stylu, w jakim jest *Krzyk* Edvarda Muncha.

** **5** Przeprowadź rozmowę z wybranym naukowcem w aplikacji character.ai. Żeby dobrze się do niej przygotować, poszukaj wcześniej w różnych źródłach odpowiedzi na pytania, które zadasz. Sprawdź, czy wiedza zdobyta podczas przygotowań pokrywa się z odpowiedziami chatbota.

** **6** Przygotuj prezentację, w której omówisz główne zagrożenia związane z rozwojem narzędzi wykorzystujących sztuczną inteligencję. Możesz skupić się na jednym obszarze życia (np. edukacja, praca, zdrowie, relacje z innymi ludźmi) lub przygotować przekrojowy materiał identyfikujący możliwe zagrożenia w różnych obszarach.

*** **7** Za pomocą Google Teachable Machine przygotuj projekt związany z rozpoznawaniem dźwięków. Mogą się w nim znaleźć np. dwie klasy: Powitania i Pożegnania. W pierwszej z klas powinny być zwroty typu „dzień dobry”, „cześć”, a w drugiej – zwroty typu „do widzenia”, „pa”. Przetestuj działanie modelu.

*** **8** Korzystając z kamery (np. w laptopie) i aplikacji Google Teachable Machine, wytrenuj model rozpoznający twój nastrój.

*** **9** Korzystając z dowolnego narzędzia AI do generowania muzyki, skomponuj utwór w stylu, który lubisz (np. pop, rap, rock itp.). Przygotuj i wprowadź tekst piosenki, którą sztuczna inteligencja ma zaśpiewać. Tekst możesz napisać samodzielnie lub wygenerować w jednym z dostępnych chatbotów.

12 Sterowanie robotem

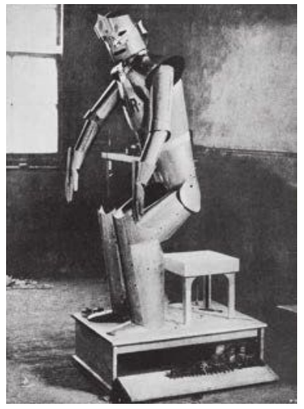
Od wielu lat tematy związane z automatyką i robotyką zyskują na popularności. Uczelnie wyższe wykorzystują roboty na różnych kierunkach studiów, przedsiębiorstwa używają ich m.in. do przyspieszenia pracy, zwiększenia jej precyzji i wydajności oraz do jej automatyzacji. Roboty już od ponad 100 lat nie tylko są bohaterami literatury i filmu, lecz także pod różnymi postaciami są obecne w naszym życiu codziennym. Ułatwiają wiele prac, a niekiedy nawet nas wyręczają. W tym temacie nauczysz się programować roboty edukacyjne oraz symulować ich działanie.

Ważne w tym temacie

- Definiowanie robotów
- Budowa oraz parametry wybranych robotów edukacyjnych
- Programowanie robotów z wykorzystaniem różnych środowisk dostępnych online
- Symulacje zaprogramowanych robotów w środowisku wirtualnym

12.1. Czym są roboty?

Słowo „robot” upowszechniło się w latach 20. XX w. po publikacji sztuki *R.U.R.* autorstwa Karela Čapka – czeskiego pisarza, jednego z pionierów fantastyki naukowej (rysunek 12.1 przedstawia robota inspirowanego tą sztuką). *R.U.R.* to skrót od *Rossumovi Univerzální Roboti*, co można przetłumaczyć jako „Roboty Uniwersalne Rossuma”. Sztuka ta miała swoją polską prapremierę w 1922 r. w warszawskim Teatrze Rozmaitości.



Rys. 12.1. Robot inspirowany sztuką *R.U.R.*

Zaledwie kilka lat później roboty pojawiły się w filmie *Metropolis* w reżyserii Fritza Langa i towarzyszą nam do dziś na ekranach kin i telewizorów. Do najpopularniejszych należą: para C-3PO i R2-D2 z *Gwiezdnych wojen*, roboty serii T (T-800, T-X) z filmów o Terminatorze czy też tytułowy bohater animacji *WALL-E*.

Samo pojęcie robota definiowane jest różnie w zależności od przeznaczenia. Według Encyklopedii Powszechnej PWN robot to „cybernetyczna (sterowana) maszyna lokomocyjno-manipulacyjna przeznaczona do wspomagania lub realizacji czynności energetyczno-ruchowych, sensualnych i intelektualnych człowieka” [https://encyklopedia.pwn.pl/szukaj/robot.html, dostęp: 24.09.2025].

Warto wiedzieć

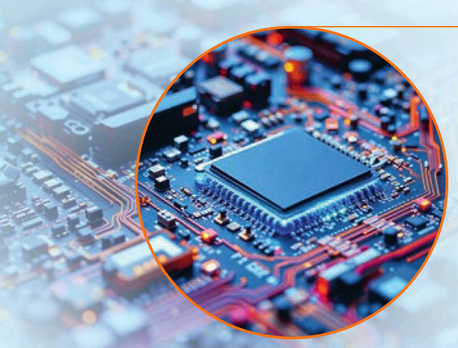
Pod koniec lat 50. XX w. polski pisarz i futurysta Stanisław Lem napisał opowiadanie *Czy pan istnieje, Mr. Johns?*, które dotyka problemu cyborgizacji, czyli zastępowania narządów człowieka urządzeniami mechanicznymi. Na podstawie tego opowiadania Andrzej Wajda nakręcił film pt. *Przekładaniec*.

Warto wiedzieć

Ze względu na powszechne użycie robotów w przemyśle powstała norma PN-EN ISO 8373:2001, która definiuje robota przemysłowego.

CO WARTO WIEDZIEĆ O ROBOTACH I ROBOTYCE?

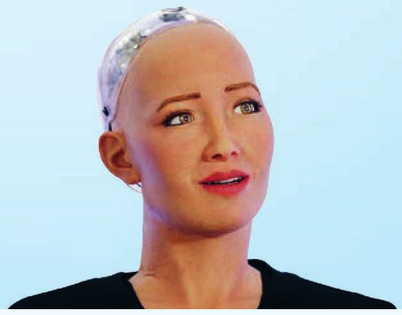
Nauka robotyki jest świetnym sposobem na rozwijanie umiejętności z obszaru nauk ścisłych. Daje także dużo satysfakcji i jest zajęciem bardzo perspektywicznym. Roboty mogą się między sobą istotnie różnić, dlatego warto przyjrzeć im się bliżej.



Robotem steruje komputer wyposażony w procesor. W nim wykonywane są programy, które mogą wykorzystywać dane pobrane z czujników umieszczonych w robocie. W niektórych robotach stosuje się moduły AI.

Roboty rozrywkowe

Niektóre roboty dostarczają człowiekowi rozrywki. Mogą mieć różną postać, np. imitować ludzi lub zwierzęta.



Robot humanoidalny
Przykładowe parametry:
- **waga:** 40 kg
- **liczba kamer:** 4
- **czujniki:** dotyku, widzenie głębi 3D, kamera oka, mikrofon, głośnik, kompas, żyroskop
- **udźwig:** 2,7 kg (chwyt dłonią)
- sztuczna skóra twarzy i szyi
- system rozpoznawania i wyrażania emocji.

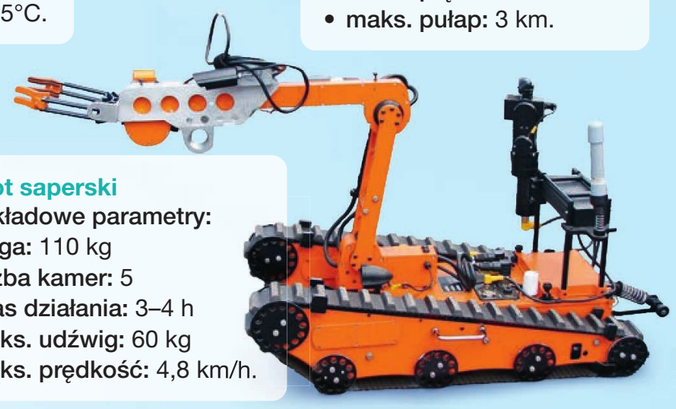
Robot zabawka
Przykładowe parametry:
- **waga:** 1,6 kg
- **liczba kamer:** 1
- **czujniki:** mikrofon, głośnik, czujnik temperatury
- **czas działania:** 1,5 h.

Roboty specjalistyczne

Konstrukcja robotów specjalistycznych jest zróżnicowana. Dron musi być lekki i szybki. Robot saperski ma rozbudowany system kamer i precyzyjne ramię. Robot penetracyjny powinien sprawnie docierać do trudno dostępnych miejsc.

Dron
Przykładowe parametry:
- **waga:** 4,5 kg
- **liczba kamer:** 1
- **maks. czas lotu:** 32 min
- **maks. prędkość:** 82 km/h
- **maks. pułap:** 3 km.

Robot penetracyjny
Przykładowe parametry:
- **waga:** 32 kg
- **czas działania:** 1,5 h
- **maks. udźwig:** 14 kg
- **maks. prędkość:** 5 km/h
- **zakres temperatur:** od –20°C do 45°C.



Robot saperski
Przykładowe parametry:
- **waga:** 110 kg
- **liczba kamer:** 5
- **czas działania:** 3–4 h
- **maks. udźwig:** 60 kg
- **maks. prędkość:** 4,8 km/h.

Roboty pomagające człowiekowi

Zadaniem niektórych robotów jest pomoc człowiekowi w codziennych czynnościach, np. opiece nad osobami starszymi lub w przemieszczaniu się.

Robot opiekun
Przykładowe parametry:
- **waga:** 3 kg
- **czujniki:** kamera, mikrofon, głośnik
- **funkcje:** rozrywka (np. karaoke, filmy, zabawy), ciśnieniomierz, rozmowy wideo, stymulowanie pamięci, alarmy (np. na żądanie lub po dłuższej nieobecności podopiecznego).

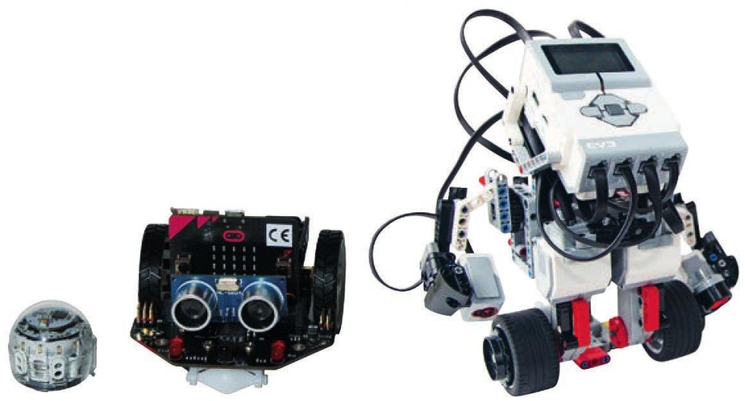
Samochód autonomiczny
Przykładowe parametry:
- **waga:** 2000 kg
- **liczba kamer:** 8
- **czujniki:** 12 czujników ultradźwiękowych, radar, sonar
- **systemy:** GPS, mapowanie 3D, chmura obliczeniowa, system komunikacji, uczenie maszynowe.

Robot

Przyjmijmy nieco uproszczoną definicję. **Robotem** będziemy nazywać urządzenie możliwe do zaprogramowania lub sterowania, które jest zdolne do poruszania się (wykonywania różnych ruchów w przestrzeni), a jego zadaniem jest ułatwienie człowiekowi pracy lub zastąpienie go w skomplikowanych, powtarzalnych lub monotonnych czynnościach.

12.2. Roboty w edukacji

Dzięki różnorodnym robotom edukacyjnym (rys. 12.2) programowania robotów może się dziś nauczyć niemal każdy, niezależnie od wieku i etapu edukacji szkolnej.



Rys. 12.2. Przykłady robotów edukacyjnych

Wraz z robotem otrzymuje się dostęp do środowiska, w którym można zaprogramować jego działanie. Istnieją też środowiska programistyczne obsługujące roboty różnych producentów.

Brak robota nie stanowi problemu, ponieważ mogą go zastąpić **symulatory robotów**. Są to środowiska, które umożliwiają programowanie oraz obserwację zachowania wirtualnych robotów (często cyfrowych odpowiedników rzeczywistych robotów).

Symulator robota

Robotów edukacyjnych jest wiele, regularnie pojawiają się też nowe modele. Dlatego omówimy tylko wybrane roboty, które spełniają następujące kryteria ważne w edukacji:

- popularność w literaturze i zasobach internetowych – zapewnia to dostęp do obszernej bazy wiedzy na temat robota, jego budowy, programowania, zastosowań i przykładów realizacji,
- różnorodność dostępnych środowisk programowania – pozwoli to wybrać odpowiedni język i środowisko programowania, niezależnie od posiadanego systemu operacyjnego, z możliwością użycia narzędzi online i offline, języka tekstowego lub wizualnego,
- możliwość symulacji robota – dzięki temu można uczyć się robotyki na ekranie komputera, symulując działanie robota dokładnie tak, jak zachowywałby się w rzeczywistości.

Warto wiedzieć

Popularnymi robotami edukacyjnymi są:

- Edison,
- Dash i Dot,
- mBot,
- Photon,
- Ozobot,
- LEGO MINDSTORMS,
- micro:Maqueen.

12.3. Robot LEGO MINDSTORMS EV3

Jednym z popularnych na rynku edukacyjnym robotów jest zestaw LEGO MINDSTORMS EV3, który daje bardzo duże możliwości twórcze. Trzonem tego zestawu jest inteligentna kostka (rys. 12.3) – programowalny komputer wyposażony w procesor ARM9.

Kostka EV3 zawiera:

- podświetlany interfejs,
- przyciski sterujące,
- ekran,
- głośnik,
- port USB,
- czytnik kart miniSD,
- cztery porty wejściowe,
- cztery porty wyjściowe.

Rys. 12.3. Kostka EV3 wraz z podłączonymi do niej silnikami i czujnikami

Zestaw EV3 wspiera takie technologie jak Bluetooth oraz wi-fi. Pozwala również na programowanie zarówno z wykorzystaniem narzędzi komputerowych (specjalnych środowisk programistycznych), jak i bezpośrednio z poziomu samej kostki.

W zestawie EV3 wykorzystywane są **serwomotory**, czyli silniki służące do poruszania elementów konstrukcji robota. Do jednej kostki EV3 można podłączyć cztery niezależne silniki. EV3 można również wyposażyć w następujące czujniki:

Serwomotor

- **czujnik kolorów** – mierzy barwę oraz natężenie światła,
- **czujnik podczerwieni** – wykrywa źródła podczerwieni, np. z nadajnika sterującego robotem,
- **czujnik żyroskopowy** – wykrywa ruch obrotowy robota i zmianę jego położenia,
- **czujnik dotykowy** – wykrywa naciśnięcie lub zwolnienie przycisku,
- **czujnik ultradźwiękowy** – mierzy odległość od obiektów,
- **czujnik temperatury** – mierzy temperaturę w zakresie od –20°C do +120°C.

Czujnik kolorów
Czujnik podczerwieni
Czujnik żyroskopowy
Czujnik dotykowy
Czujnik ultradźwiękowy
Czujnik temperatury

Dzięki możliwości łączenia kostki EV3 z innymi klockami producenta zestaw ten pozwala konstruować wiele różnorodnych robotów edukacyjnych.

Środowisko LEGO MINDSTORMS

Podstawowym narzędziem programowania kostki EV3 i podłączonych do niej silników oraz czujników jest oprogramowanie firmy LEGO przeznaczone dla zestawu MINDSTORMS. Jest ono darmowe i można je pobrać ze strony producenta.

Warto wiedzieć

Robot EV3 jest następcą zestawu NXT 2.0.
Po EV3 w 2020 r. firma Lego wprowadziła zestawy Robot Inventor (następcę EV3) oraz nieco prostszy w budowie i działaniu SPIKE Prime.

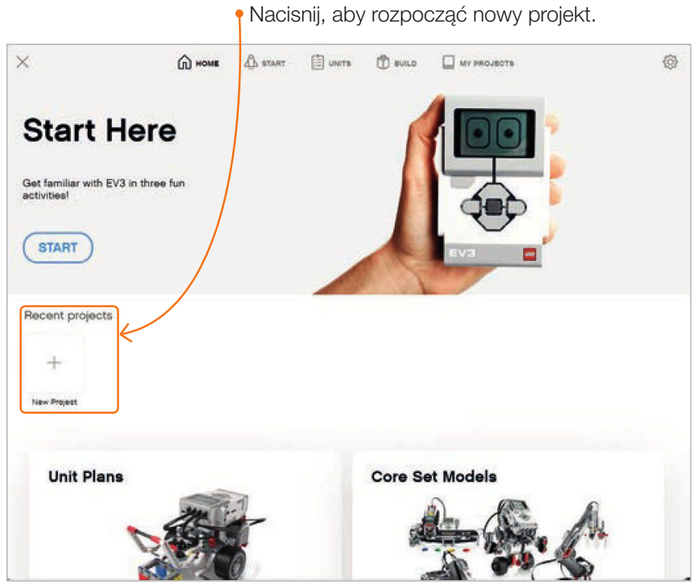
Dobra rada

Jeśli jesteś użytkownikiem systemu macOS, do zaprogramowania robota EV3 możesz wykorzystać również oprogramowanie LEGO MINDSTORMS.

Dobra rada

Jeśli jesteś użytkownikiem systemu Linux, robota EV3 możesz zaprogramować za pomocą systemu operacyjnego ev3dev (opartego na dystrybucji Debian). Robota możesz uruchomić z wykorzystaniem karty pamięci microSD i programować m.in. w językach C++ lub Python.

Po uruchomieniu środowiska zobaczymy ekran, na którym można wybrać gotowe przykłady, propozycje lekcji lub stworzyć bardziej zaawansowane roboty. Własny, pusty projekt uruchamiamy przyciskiem „+” (**New Project**) w środkowej części okna (rys. 12.4).

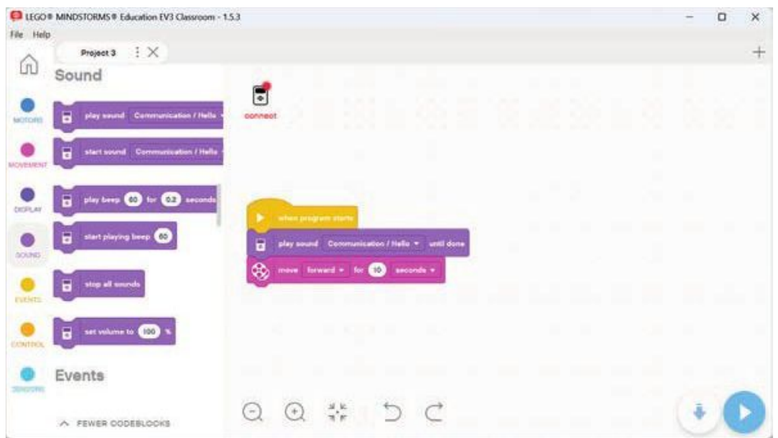


Rys. 12.4. Środowisko LEGO MINDSTORMS

Programowanie robota EV3 w środowisku LEGO MINDSTORMS

Programowanie robota odbywa się w środowisku blokowym podobnym do Scratcha. Bloki są podzielone na kategorie, które dotyczą np. silników, czujników, ekranu na kostce EV3 czy sygnałów dźwiękowych. Po zbudowaniu programu z sekwencji bloków przesyłamy go do podłączonej wcześniej kostki EV3 (na której jest zbudowany robot).

Rysunek 12.5 przedstawia przykładowy program.

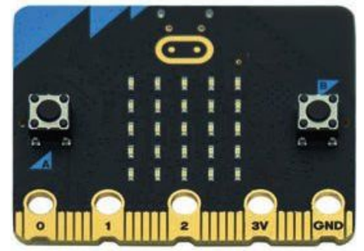


Rys. 12.5. Program zapisany w środowisku LEGO MINDSTORMS

Po uruchomieniu programu widocznego na rysunku 12.5 na s. 164 usłyszymy powitanie dźwiękowe. Potem na 10 sekund zostaną uruchomione silniki podłączone do kostki.

12.4. Mikrokontroler BBC micro:bit

Konstruowanie robotów można połączyć z nauką elektroniki. Przykładem takiego rozwiązania jest mikrokontroler BBC micro:bit (rys. 12.6).



Rys. 12.6. Mikrokontroler BBC micro:bit 2

Warto wiedzieć

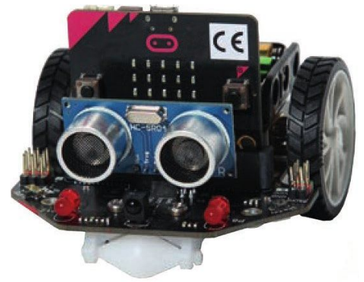
Poprzednia wersja płytki micro:bit nie była wyposażona w głośnik i mikrofon oraz przycisk dotykowy. Nie miała też trybu oszczędzania energii i miała mniejszą moc obliczeniową.

Jest to niewielka płytka elektroniczna wyposażona m.in. w procesor ARM Cortex-M4, moduły komunikacji Bluetooth i USB, dwa przyciski, mikrofon, głośnik oraz wbudowane czujniki:

- **magnetometr** – czujnik, który działa na zasadzie kompasu, można dzięki niemu wskazać kierunek północny, ◂ Magnetometr
- **akcelerometr** – czujnik mierzący ruch wzdłuż trzech osi: X (przechylenie w lewo/w prawo), Y (przechylenie w przód/w tył), Z (poruszanie się w górę/w dół). ◂ Akcelerometr

Charakterystycznym elementem płytki micro:bit jest umieszczony na niej ekran składający się z 25 diod LED (5 × 5), na którym można wyświetlać proste komunikaty graficzne lub tekstowe. Płytka wyposażona jest również w złącze rozszerzeń pozwalające rozbudować ją o dodatkowe akcesoria.

Płytka micro:bit pojawia się w wielu zestawach edukacyjnych z zakresu robotyki. Można dzięki nim budować roboty lub realizować ciekawe projekty rozwijające m.in. umiejętności programistyczne.



Rys. 12.7. Przykład robota micro:Maqueen wykonanego z użyciem mikrokontrolera micro:bit

Przykładem takiego zastosowania jest robot micro:Maqueen (rys. 12.7), który wykorzystuje właśnie płytkę micro:bit oraz czujniki: ultradźwiękowy, wykrywania linii oraz podczerwieni (umożliwia on sterowanie robotem za pomocą pilota).

Programowanie płytki micro:bit w języku Python

Do programowania płytki micro:bit wykorzystuje się język Python, a właściwie jego odmianę – **język MicroPython**, który został zoptymalizowany pod kątem stosowania go w mikrokontrolerach. ◂ Język MicroPython

Dobra rada

Kod źródłowy możesz tworzyć również w samym edytorze Mu. Aby to zrobić, wybierz **Tryb → BBC micro:bit**.

Kod źródłowy można tworzyć w środowisku online opartym na edytorze Mu, dostępnym na stronie python.microbit.org/v/3.

Aby zobaczyć działanie napisanego kodu, należy przesłać go na płytkę i uruchomić. Można też wykorzystać do tego symulator dostępny bezpośrednio w edytorze (rys. 12.8).

Rys. 12.8. Programowanie płytki micro:bit w języku MicroPython

Warto wiedzieć

Dokumentację i dokładny opis poleceń języka MicroPython w języku polskim można znaleźć na stronie: microbit-micropython.readthedocs.io/pl/latest/index.html.

12.5. Programowanie robotów i symulatory

Jeśli nie mamy dostępu do fizycznego robota, warto wykorzystać narzędzia do programowania i symulacji działania robotów. Dzięki nim można zrealizować projekty o takim samym stopniu skomplikowania i mieć z nich taką samą satysfakcję jak w wypadku pracy z rzeczywistym robotem.

Warto wiedzieć

Symulatory pozwalają programować i testować roboty w bezpiecznych warunkach, bez obaw o przypadkowe błędy, które mogłyby spowodować uszkodzenia sprzętu.

Omówimy środowiska online, które pozwalają programować roboty oraz symulować ich działanie: Open Roberta Lab, Microsoft MakeCode oraz VEXcode VR.

12.6. Symulacja robota EV3 w środowisku Open Roberta Lab

Open Roberta Lab to darmowe środowisko programowania i symulacji kilku popularnych robotów, w tym LEGO MINDSTORMS EV3. Wykorzystuje ono rozwiązania chmurowe, więc nie wymaga instalacji. Założenie konta na stronie projektu umożliwia zapisanie online wszystkich utworzonych programów.

A to ciekawe

Roboty na sali operacyjnej

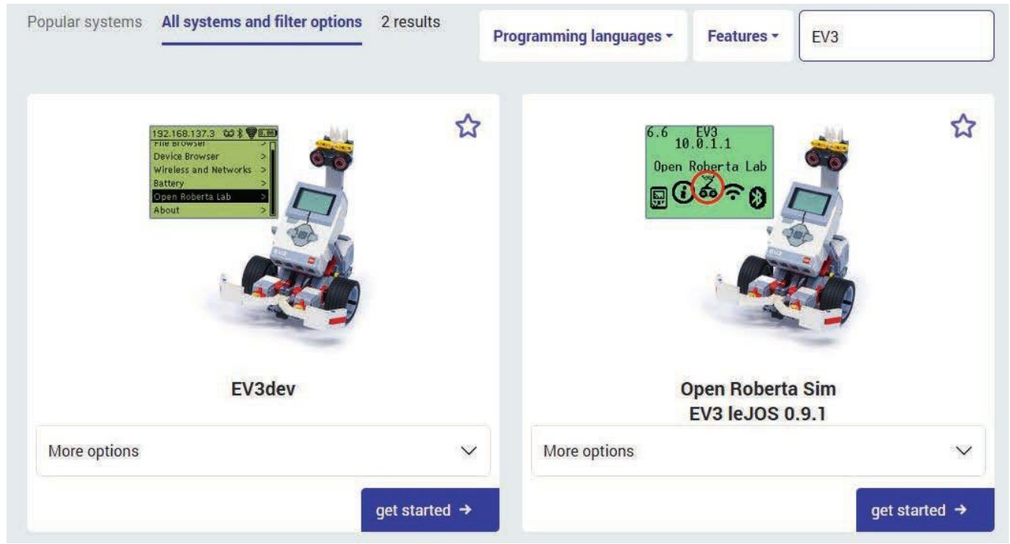
W 1999 r. na rynku pojawił się, stworzony przez firmę Intuitive Surgical, robot chirurgiczny da Vinci. Umożliwia on przeprowadzanie skomplikowanych zabiegów, ponieważ ma cztery ramiona pozwalające na wykonywanie bardzo precyzyjnych czynności. Został już wykorzystany w kilkunastu milionach operacji na całym świecie.

Programowanie odbywa się za pomocą graficznego interfejsu, w którym program układa się z dostępnych bloków. W zależności od wybranego robota można również podejrzeć kod przetłumaczony m.in. na język Python. Utworzony program można następnie zaimplementować w robocie lub przetestować w oknie symulatora. Serwis Open Roberta Lab jest dostępny w języku polskim.

Warto wiedzieć

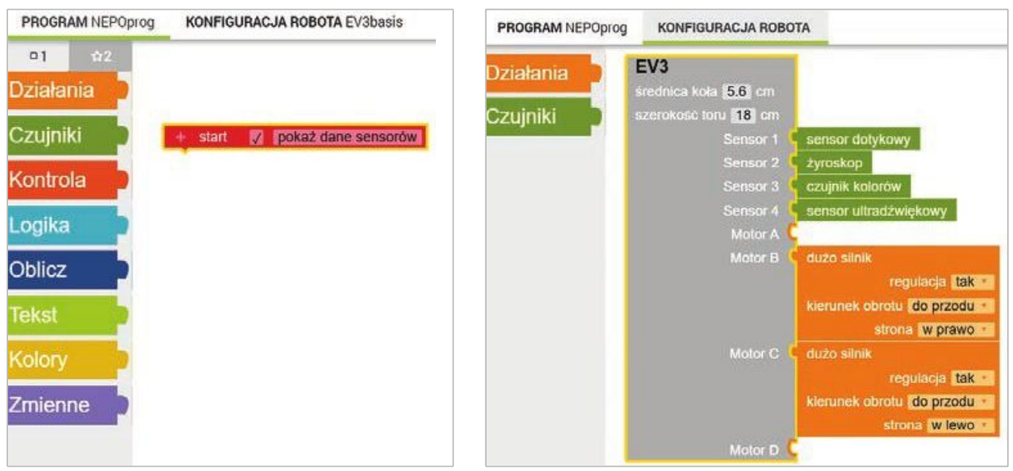
Okna symulacji robotów są dostępne tylko dla niektórych modeli w środowisku Open Roberta Lab.

Po uruchomieniu strony projektu lab.open-roberta.org wybieramy robota oraz system operacyjny. W naszym wypadku będzie to robot EV3 i system EV3 dev (rys. 12.9).



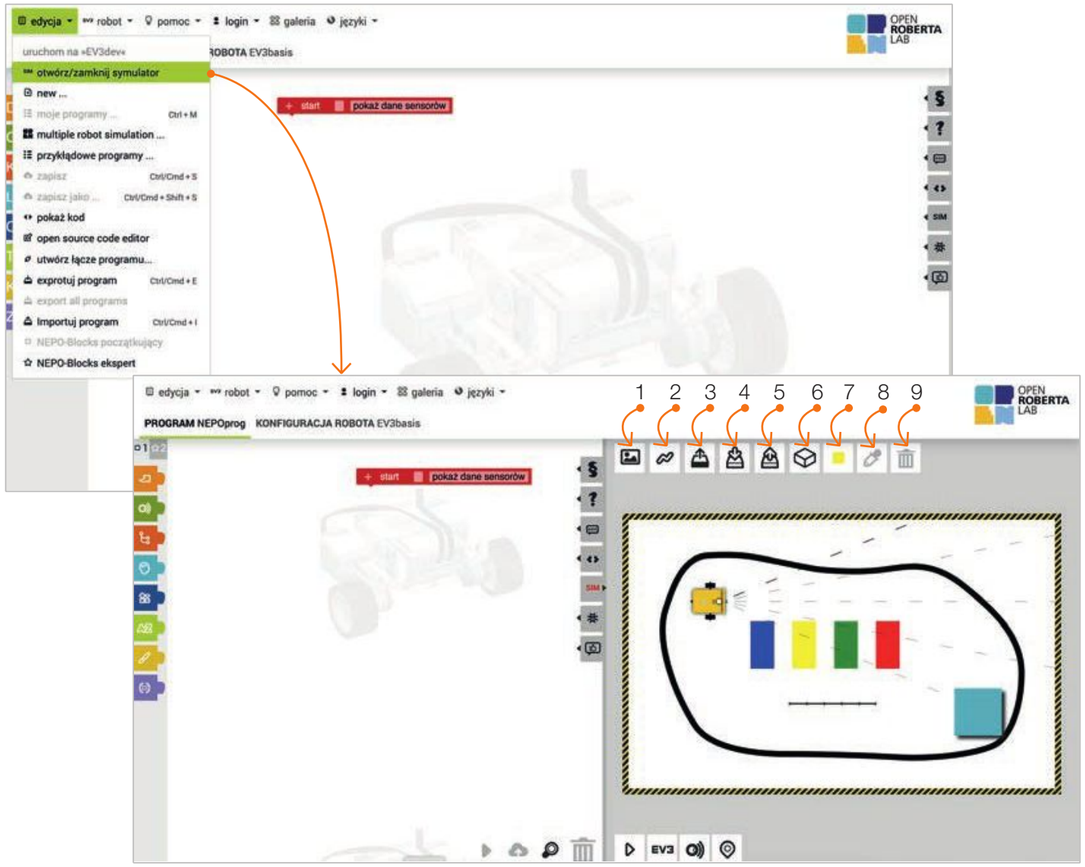
Rys. 12.9. Wybór robota EV3 w środowisku Open Roberta Lab

Rysunek 12.10 przedstawia okno z obszarem roboczym (zrzut z lewej) oraz okno z podglądem konfiguracji robota EV3 (zrzut z prawej).



Rys. 12.10. Obszar roboczy Open Roberta Lab i konfiguracja robota EV3

Aby włączyć podgląd robota, wystarczy wybrać przycisk edycji i opcję **otwórz/zamknij symulator**. Podgląd przedstawia robota wyposażonego w dwa koła, promienie czujnika ultradźwiękowego wychwytujące przeszkodę (niebieski kwadrat), kolorowe prostokąty oraz ścieżkę, po której robot może jeździć, jeśli zostanie tak zaprogramowany. W tym widoku symuluje się działanie robota (rys. 12.11).



Rys. 12.11. Włączanie podglądu, w którym można symulować działanie robota

W oknie symulatora znajduje się kilka przycisków – na rysunku 12.11 oznaczone są kolejnymi numerami od 1 do 9. Przyciski pozwalają na:

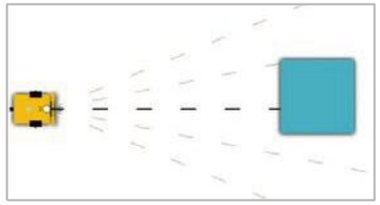
- zmianę tła sceny (1),
- rysowanie ścieżki ruchu robota (2),
- wgranie własnego tła (3),
- pobranie ustawień symulatora (4),
- wczytanie ustawień symulatora (5),
- dodawanie przeszkód (6),
- dodawanie kolorowych obszarów do sceny (7),
- pobranie koloru z obiektu lub powierzchni (8),
- usunięcie obiektu lub kolorowej powierzchni (9).

Program *Zatrzymanie przed przeszkodą*

Zaprogramujemy reakcję robota na spotkanie z przeszkodą: robot zatrzyma się, jeśli wykryje, że przeszkoda znajduje się w odległości 10 cm.

Aby to zrobić, zmienimy najpierw tło w oknie symulatora (przycisk 1 z rys. 12.11, s. 168) na takie, w którym robot jest ustawiony w pewnej odległości przed przeszkodą (rys. 12.12). Linie przerywane obrazują działanie **czujnika ultradźwiękowego**.

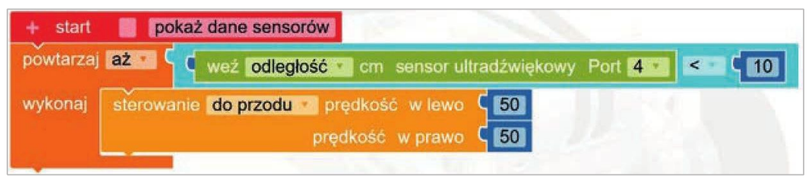
Czujnik ultradźwiękowy
→ s. 163



Rys. 12.12. Robot ustawiony przed przeszkodą

Robotem będzie sterowała pętla, wykonywana do momentu zwrócenia przez czujnik ultradźwiękowy informacji o tym, że odległość od przeszkody jest mniejsza lub równa 10 cm. Aby robot poruszał się do przodu, trzeba dla sterowania do przodu ustawić taką samą prędkość w lewo i w prawo.

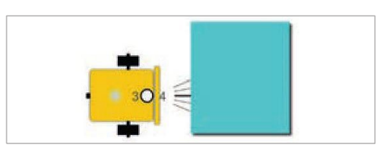
Kod źródłowy programu *Zatrzymanie przed przeszkodą* przedstawiono na rysunku 12.13.



Kod źródłowy programu *Zatrzymanie przed przeszkodą*

Rys. 12.13. Kod źródłowy programu *Zatrzymanie przed przeszkodą*

Na rysunku 12.14 przedstawiono wynik symulacji – robot zatrzymuje się przed przeszkodą, a linie wychodzące od robota obrazują działanie sensora ultradźwiękowego.



Rys. 12.14. Efekt symulacji działania robota w programie *Zatrzymanie przed przeszkodą*

Ćwiczenie 1

Zapisz w środowisku Open Roberta Lab kod źródłowy programu *Zatrzymanie przed przeszkodą* i sprawdź działanie programu.

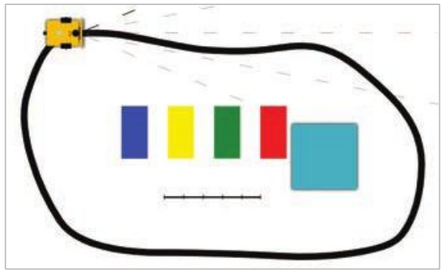
Program *Jazda po ścieżce*

Warto wiedzieć

Podczas jazdy robota po ścieżce czujnik koloru może działać w jednym z trzech trybów: rozpoznawania kolorów, określania natężenia światła odbitego i określania natężenia światła otoczenia. W polskim tłumaczeniu opisu tego czujnika dwa razy pojawia się „światło otoczenia". Należy pamiętać, że pierwsza opcja dotyczy natężenia światła odbitego, a druga – światła otoczenia.

Wykonamy kolejne zadanie: robot będzie jeździł po wyznaczonej ścieżce narysowanej linią ciągłą.

Ustawiamy odpowiednie tło (rys. 12.15) i przesuwamy przeszkodę tak, aby nie kolidowała z robotem podczas jazdy po linii. Robota umieszczamy na czarnej linii.



Rys. 12.15. Ustawienie robota na ścieżce

Czujnik kolorów
→ s. 163

W rozwiązaniu możemy wykorzystać **czujnik kolorów**, który pozwoli skorygować tor ruchu robota, gdyby wyjeżdżał poza czarną linię. Rysunek 12.16 przedstawia zapis kodu źródłowego programu *Jazda po ścieżce*.

Kod źródłowy programu *Jazda po ścieżce*

Rys. 12.16. Kod źródłowy programu *Jazda po ścieżce*

Dobra rada

Open Roberta Lab udostępnia dla robota EV3 dwie listy instrukcji: jedna jest oznaczona kwadratem, a druga – gwiazdką. Na tej drugiej znajdziesz bardziej zaawansowane instrukcje sterujące.

W pętli sprawdzany jest kolor wykrywany przez czujnik koloru. Jedno koło (port B) obraca się do przodu z prędkością 25, drugie zaś (port C) – w przeciwnym kierunku z prędkością 3 (wartość –3). Kiedy czujnik koloru nie rozpoznaje czarnego koloru ścieżki, silniki poruszają się w przeciwne strony. W konsekwencji robot jeździ zygzakiem po czarnej linii. Blok **port silnika** można znaleźć na liście instrukcji oznaczonych gwiazdką i numerem 2, w części **Działania** → **Ruch**.

Ćwiczenie 2

a. Zapisz w środowisku Open Roberta Lab kod źródłowy programu *Jazda po ścieżce* i przeprowadź symulację działania robota.

b. Zmodyfikuj kod programu *Jazda po ścieżce* tak, aby robot pokonał trasę, jadąc tyłem.

Zapamiętaj

Producenci robotów poza samymi robotami udostępniają również środowiska pozwalające na ich programowanie. Można korzystać także z innych serwisów – często oferują one wygodne narzędzia do symulacji.

12.7. Symulacja płytki micro:bit w środowisku Microsoft MakeCode

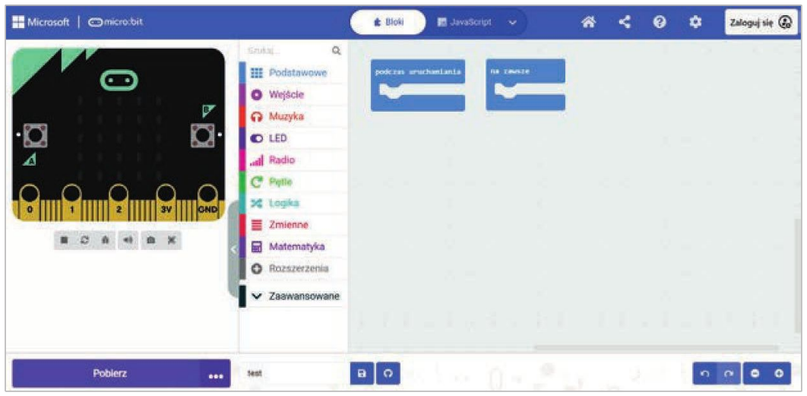
Płytka micro:bit może być programowana również z wykorzystaniem interfejsu graficznego. Przykładem takiego środowiska jest serwis Microsoft MakeCode (makecode.microbit.org). Na tej stronie poza samym środowiskiem programowania znajdziemy również dużą bazę przykładów, lekcji i filmów instruktażowych z wykorzystaniem tego mikrokontrolera.

Rysunek 12.17 pokazuje fragment okna środowiska Microsoft MakeCode uruchomionego dla płytki micro:bit. Po lewej stronie okna znajduje się obraz reprezentujący wygląd jednej strony płytki micro:bit z charakterystycznymi diodami LED i przyciskami.

Warto wiedzieć

Środowisko MakeCode obsługuje wiele robotów i innych projektów edukacyjnych, w tym:

- LEGO MINDSTORMS EV3,
- Circuit Playground Express,
- Minecraft,
- Arcade,
- Cue,
- Chibi Chip.



Rys. 12.17. Środowisko programistyczne Microsoft MakeCode dla płytki micro:bit

W tym środowisku można wygodnie tworzyć kod źródłowy za pomocą bloków, które zawierają gotowe instrukcje. Pozwalają one wyświetlać m.in. zdefiniowane wcześniej symbole lub napisy, które powstają poprzez zaświecenie się wybranych diod. Można również samodzielnie tworzyć symbole według własnych projektów.

Program *Rzut kostką*

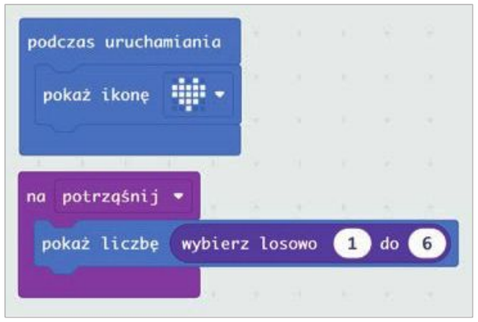
Zaprogramujemy płytkę tak, aby na początku zaświecone diody utworzyły symbol serca, a po potrząśnięciu płytki został zasymulowany rzut kostką do gry – diody powinny zaświecić się tak, aby pojawiła się liczba od 1 do 6.

Warto wiedzieć

Programowanie za pomocą elementów graficznych jest charakterystyczne nie tylko dla robotów edukacyjnych. Takie rozwiązanie występuje również w zastosowaniach profesjonalnych i komercyjnych. Przykładem może być środowisko Simulink.

Rysunek 12.18 przedstawia kod źródłowy programu *Rzut kostką*.

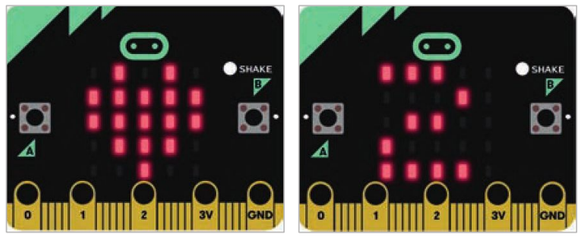
Kod źródłowy programu *Rzut kostką*



Rys. 12.18. Kod źródłowy programu *Rzut kostką*

Aby wyświetlić symbol serca, w bloku `podczas uruchamiania` wykorzystamy instrukcję `pokaż ikonę`. Do zaprogramowania działania płytki po potrząśnięciu nią użyjemy bloku z instrukcją `na`.

Po uruchomieniu programu na ekranie mikrokontrolera pojawia się ikona serca, a kiedy nim potrząsamy (lub naciskamy przycisk **SHAKE**), wyświetla się losowa liczba z zadanego zakresu (rys. 12.19).



Rys. 12.19. Symulacja działania programu *Rzut kostką*

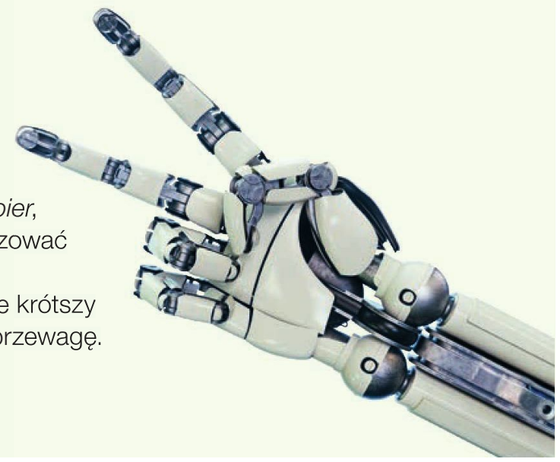
Ćwiczenie 3

Zapisz w środowisku Microsoft MakeCode kod źródłowy programu *Rzut kostką* i przeprowadź symulację działania płytki micro:bit.

A to ciekawe

Robot zawsze wygrywa?

Japoński robot Janken ma czujniki wizyjne oraz robotyczną dłoń, którą może zagrać z człowiekiem w popularną grę *Papier, kamień, nożyce*. Czujniki Jankena mogą odebrać i przeanalizować sygnał w ciągu zaledwie kilku milisekund oraz odpowiednio zareagować. Czas reakcji robota jest więc nieporównywalnie krótszy niż czas reakcji człowieka. Dzięki temu Janken zawsze ma przewagę.



12.8. Środowisko do symulacji VEXcode VR

VEXcode VR (vr.vex.com) jest darmowym środowiskiem do symulacji robota przeznaczonym do celów edukacyjnych. Pozwala ono programować wirtualnego robota z wykorzystaniem bloków i przypomina środowisko Scratch. Bardziej zaawansowani użytkownicy mogą również programować tego wirtualnego robota w języku Python.

Robot, którego programujemy, jest wyposażony m.in. w:

- elektromagnes (do podnoszenia przedmiotów),
- wirtualne oczy (np. do rozpoznawania przeszkód),
- czujniki dotykowe (lewy i prawy),
- czujnik odległości,
- żyroskop.

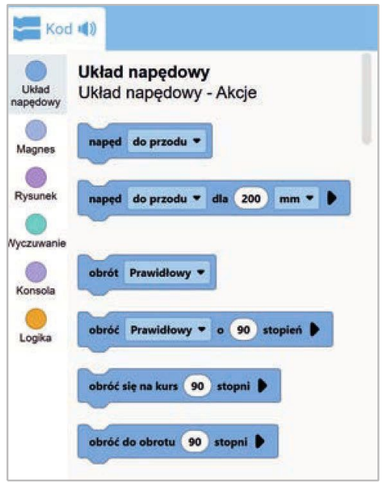
W środowisku VEXcode VR na głównej belce (rys. 12.20) umieszczono następujące narzędzia: zmiana języka, menu **Plik**, samouczki, zmiana tła (narzędzie **WYBIERZ PLAC ZABAW**) oraz narzędzia do sterowania wykonywaniem programu. Poniżej znajdują się opcja przełączania kodu blokowego na kod języka Python oraz kontrola wartości zmiennych i czujników.

Warto wiedzieć

Można wykupić roczną subskrypcję premium symulatora VEXcode VR. Dzięki temu otrzymuje się dostęp do większej liczby robotów, plansz i dodatkowych funkcji.

Rys. 12.20. Narzędzia środowiska VEXcode VR

W lewej części okna znajduje się panel **Kod**. Zawiera on bloki instrukcji, z których można tworzyć program sterujący robotem. Fragment okna przedstawia rysunek 12.21.

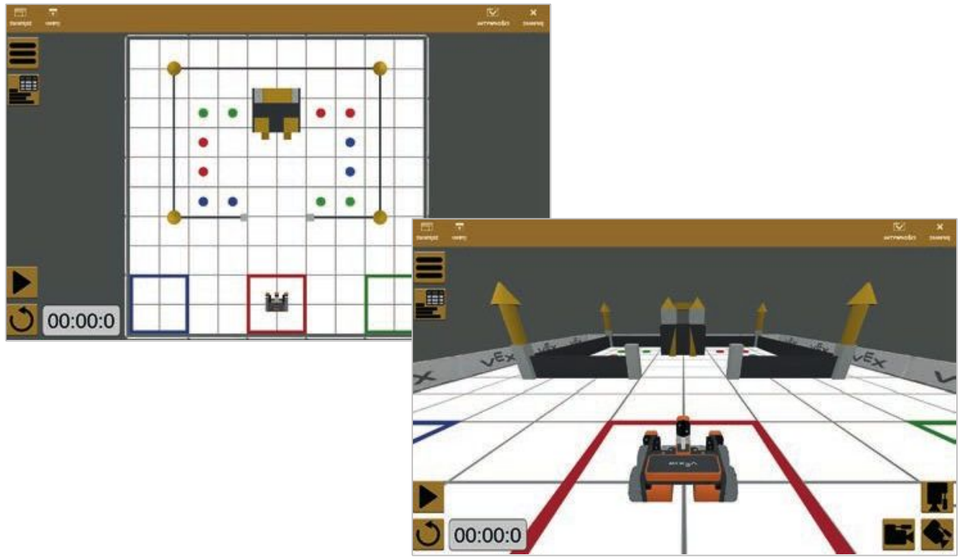


Rys. 12.21. Panel **Kod** zawierający instrukcje sterujące robotem

W środowisku VEXcode VR możemy obserwować zachowanie robota z różnych perspektyw – mamy do dyspozycji m.in. widok z góry oraz widok trójwymiarowy z kamery podążającej za robotem (rys. 12.22, s. 174).

Warto wiedzieć

Na niektórych planszach (tłach w narzędziu **WYBIERZ PLAC ZABAW**) dostępna jest jeszcze trzecia kamera, pokazująca obraz, który widzi robot.



Rys. 12.22. Okno symulacji robota

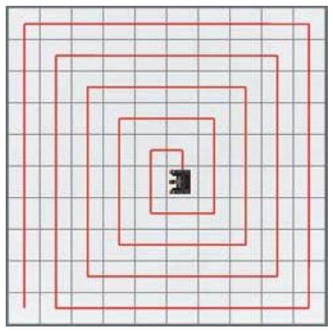
Dobra rada

Testuj swoje kody krok po kroku, uruchamiając symulację na każdym etapie. Tak łatwiej wychwycić i poprawić ewentualne błędy niż dopiero po utworzeniu całego kodu i uruchomieniu pełnego programu.

Narzędzie **WYBIERZ PLAC ZABAW** służy do ustawiania różnych plansz, po których porusza się robot, oraz do kontrolowania zachowania robota realizującego zapisany program. W wybranym widoku możemy uruchomić program sterujący robotem i w wygodny sposób zobaczyć, czy wykonuje on zakładane zadanie.

Program *Wszystkie pola planszy*

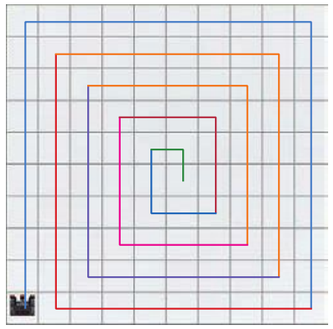
Zaprogramujemy teraz robota tak, aby przeszedł przez wszystkie pola na planszy o rozmiarach 10 × 10 i oznaczył swoją drogę czerwoną linią. Zakładamy, że robot startuje z lewego dolnego pola, a kończy na środku planszy – tak jak na rysunku 12.23.



Rys. 12.23. Plansza, po której porusza się robot

Robot wielokrotnie będzie jechał prosto przez określoną liczbę pól oraz będzie zawsze skręcał w prawo. Dlatego warto użyć pętli do zautomatyzowania i uproszczenia kodu źródłowego.

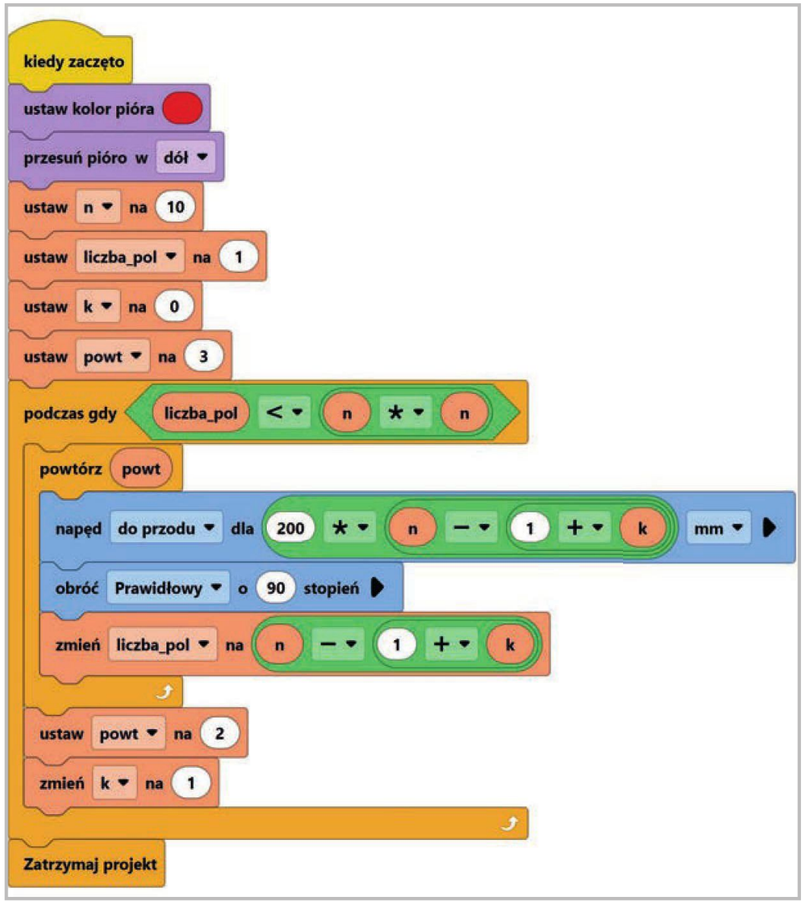
Na rysunku 12.24 przedstawiono trasę robota. Przez pierwsze 3 odcinki robot pokona 9 pól, następnie długość dwóch kolejnych odcinków zmniejszy się do 8 pól, dwóch następnych do 7 pól itd.



Rys. 12.24. Zmiany długości przemierzanych odcinków

Instrukcje sterujące robotem można zapisać jako pętle zagnieżdżone: wewnętrzna pętla sterująca jazdą na wprost i skrętem w prawo najpierw wykona się 3 razy, a następnie wielokrotnie wykona się 2 razy. Liczbę jej powtórzeń kontroluje zewnętrzna pętla, która będzie się wykonywała, dopóki robot nie odwiedzi wszystkich pól. Rysunek 12.25 przedstawia kod źródłowy programu *Wszystkie pola planszy*.

Kod źródłowy programu *Wszystkie pola planszy*



Rys. 12.25. Kod źródłowy programu *Wszystkie pola planszy*

Program rozpoczynają instrukcje dotyczące zaznaczania trasy kolorem czerwonym. Następnie definiujemy zmienne: `n` – oznaczającą długość boku kwadratu, `liczba_pol` – zliczającą liczbę odwiedzonych przez robota pól na planszy, `k` – wskazującą liczbę krawędzi kwadratu, które należy wykluczyć na danym etapie, `powt` – określającą liczbę powtórzeń wewnętrznej pętli.

Wartość zmiennej `powt` po pierwszym obrocie pętli zmienia się z 3 na 2, aby po dwóch odcinkach zmniejszyła się długość kolejnego odcinka. Długość pojedynczego kwadratu wynosi 200 mm, dlatego wyrażenie `200 * (n - (1 + k))` poprawnie opisuje długość trasy na każdym etapie.

Ćwiczenie 4

a. Zapisz kod źródłowy programu *Wszystkie pola planszy* i przeprowadź symulację ruchu robota.
b. Wywołaj program dla planszy o boku 6.

Program *Wyjście z labiryntu*

Jednym z klasycznych przykładów zadań dla robota jest znajdowanie wyjścia z labiryntu. Problem ten można rozwiązać na wiele sposobów, wykorzystując wiele różnych algorytmów.

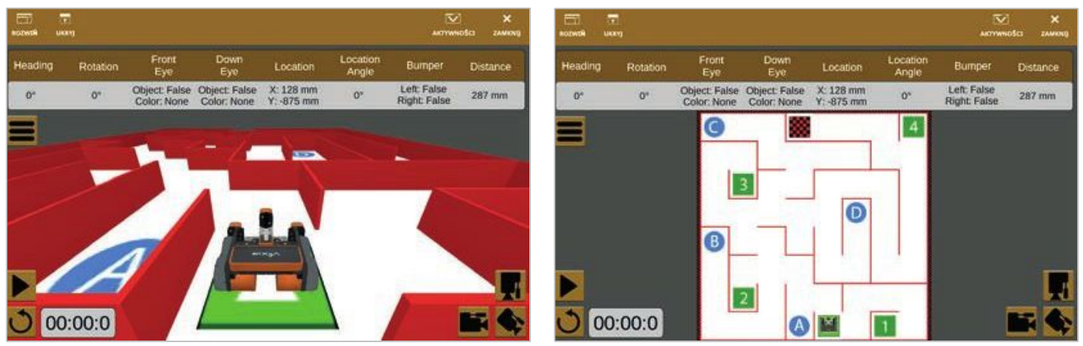
Dobra rada

Zauważ, że jeśli po pierwszym skręcie w prawo robot napotka na ścianę, to pierwszy obrót w lewo oznacza powrót do poprzedniego ustawienia. Robot będzie wtedy kontynuował jazdę prosto.

Jednym z możliwych rozwiązań jest takie, w którym robot porusza się do przodu o pewną ustaloną odległość (w naszym przypadku jest to długość pojedynczej ściany w labiryncie), a następnie obraca się w prawo i próbuje jechać prosto. Jeśli napotka przed sobą ścianę, obraca się w lewo do momentu, w którym nie będzie przed nim ściany i znów będzie mógł jechać prosto.

W środowisku VEXcode VR można skorzystać z kilku zdefiniowanych rodzajów plansz, które pozwalają realizować różnorodne algorytmy. Jedną z nich jest plansza z labiryntem.

Rysunek 12.26 przedstawia tę planszę w dwóch widokach: z kamery znajdującej się za robotem oraz z kamery znajdującej się nad planszą.

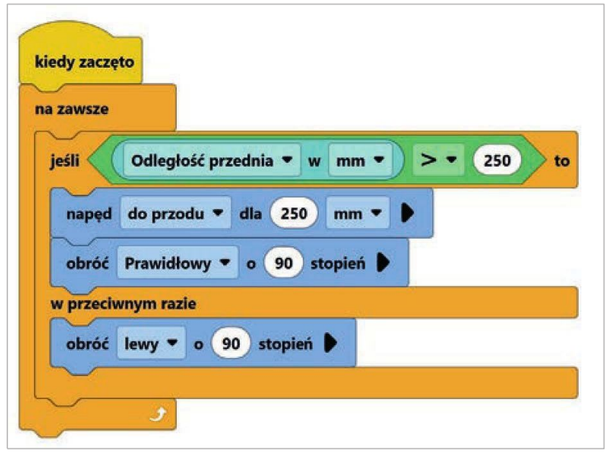


Rys. 12.26. Plansza z labiryntem – widok zza robota i z góry

Z rysunku 12.26 na s. 176 wiemy, że plansza zbudowana jest z kwadratowych pól. Każde z nich ma długość boku równą 250 mm. Ściany labiryntu są ułożone wzdłuż krawędzi tej siatki, a więc pod kątem 90 stopni.

Na rysunku 12.27 przedstawiono zapis kodu źródłowego programu *Wyjście z labiryntu*.

Kod źródłowy programu *Wyjście z labiryntu*



Rys. 12.27. Kod źródłowy programu *Wyjście z labiryntu*

W programie robot porusza się do przodu o jedno pole i skręca w prawo o 90 stopni, jeżeli odległość od przeszkody jest większa niż 250 mm, w przeciwnym razie skręca w lewo. Powtarza tę czynność aż do opuszczenia labiryntu.

Przedstawiona wcześniej metoda sprawdzi się tylko dla prostych labiryntów, w których wyjście jest połączone z krawędzią labiryntu. Nie jest jednak optymalna.

Ćwiczenie 5

a. Zapisz kod źródłowy programu *Wyjście z labiryntu* i przeprowadź jego symulację.
b. Zmodyfikuj kod źródłowy programu *Wyjście z labiryntu* tak, aby robot w pierwszej kolejności szukał możliwości skrętu w lewo.

Zapamiętaj

Symulatory robotów mogą skutecznie zastępować fizyczne urządzenia. Dają możliwość realizacji ciekawych projektów oraz korzystania z gotowych plansz, po których poruszają się roboty. Wiele środowisk symulacyjnych działa w trybie online.

Warto wiedzieć

Problem poszukiwania wyjścia z labiryntu można rozwiązać m.in. z wykorzystaniem algorytmów: Minty'ego, Bellmana–Forda, Dijkstry oraz Dantziga.

Zadania

* **1** W środowisku VEXcode VR zaprogramuj robota tak, aby odwiedził wszystkie pola planszy o rozmiarze 10 na 10 w inny sposób niż opisany w tym temacie.

* **2** W środowisku VEXcode VR zaprogramuj robota tak, aby rysował na planszy serce.

* **3** Zaproponuj, opisz i wykonaj prezentację na temat wybranego zagadnienia, które można rozwiązać za pomocą jednego robota lub wielu robotów.

* **4** Zademonstruj możliwości wybranego robota. Opisz i zaprezentuj zastosowania dostępnych czujników i silników.

** **5** W kilkuosobowej grupie zaprojektujcie proste ramię robota służące do wykonywania jakiejś czynności, np. chwytak przenoszący przedmiot z lewej strony ramienia na prawą. Zaprogramujcie kostkę EV3 tak, aby sterowała tym ramieniem. Zademonstrujcie w środowisku Microsoft MakeCode działanie programu i kostki EV3.

** **6** Zaprogramuj robota muzycznego, który będzie tworzył własną muzykę, np. z wykorzystaniem akcelerometru.

** **7** Zaprogramuj robota, który będzie służył jako kontroler do sterowania w grze komputerowej.

** **8** W środowisku VEXcode VR wybierz planszę o nazwie **Siatka liczb**. Zaprogramuj robota tak, aby odwiedził wszystkie pola planszy oznaczone liczbami nieparzystymi przy możliwie najmniejszej liczbie najazdów na pola parzyste.

*** **9** Do programu *Wyjście z labiryntu*, utworzonego w tym temacie, dodaj instrukcję zamykającą działanie skryptu po dojściu robota do wyjścia z labiryntu.

*** **10** Zaprogramuj robota (lub mikrokontroler) tak, aby za jego pomocą pokazać działanie wybranych praw fizyki, np. siły odśrodkowej.

*** **11** Zaprogramuj robota tak, aby pokonał dowolny labirynt.

*** **12** W kilkuosobowej grupie zaprogramujcie wybranego robota poruszającego się po wyznaczonej ścieżce z możliwością omijania przeszkód na drodze.

13 Korespondencja seryjna

W liście z banku lub urzędu 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.

Ważne w tym temacie

- Tworzenie i personalizowanie listów seryjnych
- Tworzenie zewnętrznych źródeł danych adresowych
- Filtrowanie adresów i drukowanie wybranych listów
- Wykorzystanie arkusza kalkulacyjnego do personalizowania korespondencji

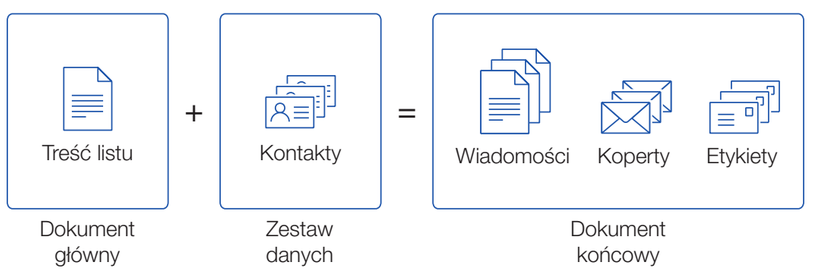
13.1. Czym jest korespondencja seryjna?

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, lecz także wszędzie tam, gdzie trzeba utrzymywać kontakt z klientami, prezentować firmowe oferty, wysyłać zaproszenia, informacje, powiadomienia itp.

Korespondencja seryjna

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. 13.1). Na przykład jeśli zestaw danych zawiera 100 adresatów, dokument końcowy korespondencji może mieć 100 stron, z których każda strona będzie zawierała tę samą treść listu, lecz inne dane adresowe.

Dokument główny
Zestaw danych
Dokument końcowy korespondencji seryjnej



Rys. 13.1. Elementy składowe korespondencji seryjnej

Warto wiedzieć

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

13.2. Korespondencja seryjna w programie Word

Dobra rada

List możesz formatować na każdym etapie tworzenia korespondencji seryjnej.

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 uzależnić 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. W zaproszeniach mają się znaleźć dane adresowe ich odbiorców. W nowym dokumencie programu Word zapisujemy główną treść wiadomości, nie podając na razie żadnych danych adresowych (rys. 13.2).

Szanowni Państwo!

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

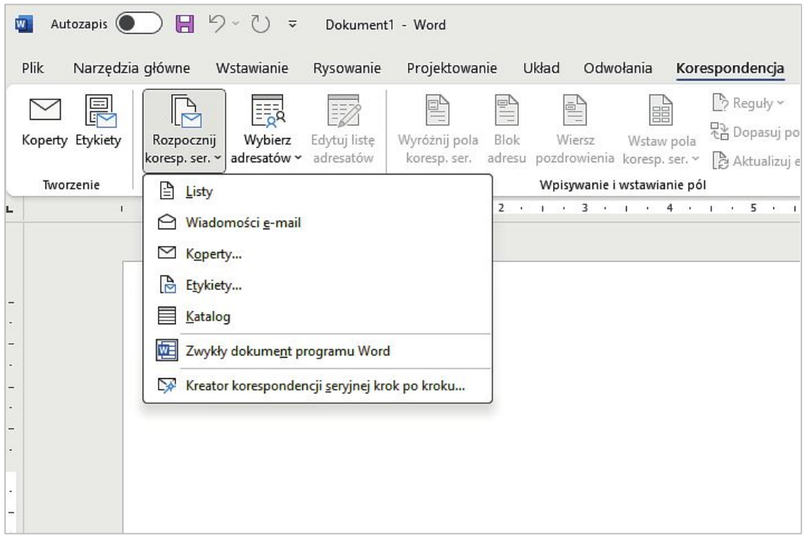
Organizatorzy

Rys. 13.2. Zaproszenie na pokaz sztucznych ogni

Warto wiedzieć

Narzędzia korespondencji seryjnej są również dostępne w oprogramowaniu biurowym LibreOffice.

Musimy teraz wskazać programowi Word, że pracujemy z dokumentem korespondencji seryjnej. W tym celu wybieramy **Korespondencja → Rozpoczynanie korespondencji seryjnej → Rozpocznij koresp. ser. → Listy** (rys. 13.3). Od tego momentu dokument jest traktowany jako dokument główny korespondencji seryjnej.



Rys. 13.3. Różne rodzaje dokumentów korespondencji seryjnej

Warto wiedzieć

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

Jak widać na rysunku 13.3 ze s. 180, w ramach korespondencji seryjnej możemy tworzyć nie tylko listy. Inne dostępne opcje to:

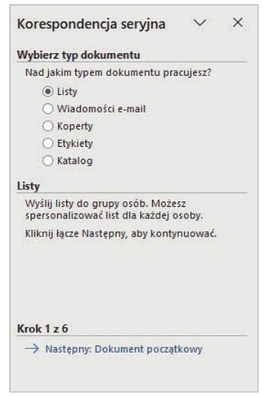
- **Wiadomości e-mail** – wybranie tej opcji powoduje zmianę widoku dokumentu na **Układ sieci Web**. Treść listu można wysłać do adresatów pocztą elektroniczną, jeżeli w podłączonej bazie adresowej jednym z pól jest adres e-mail.
- **Koperty...** – jeśli wybierzemy tę opcję, będziemy mieć możliwość 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.

Dobra rada

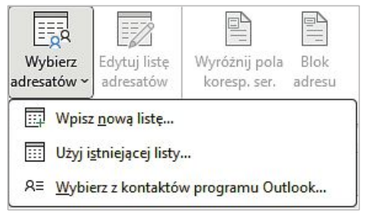
Każdą z operacji tworzenia listów, kopert i etykiet możesz wykonać również z poziomu kreatora. Wystarczy wybrać opcję **Kreator korespondencji seryjnej krok po kroku...** (rys. 13.3, s. 180).



Wybór adresatów

Kiedy mamy już gotowy dokument główny korespondencji seryjnej, możemy połączyć go z listą adresatów. Wybieramy spośród kilku opcji (rys. 13.4). Możemy:

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



Rys. 13.4. 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**.

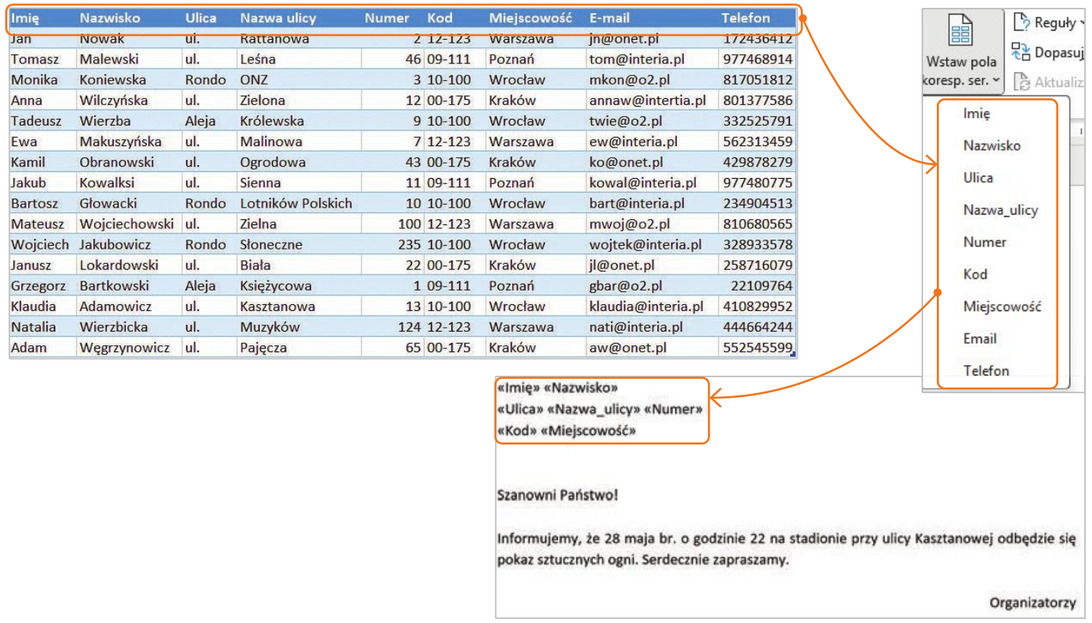
Pierwszy wiersz tabeli zawiera nagłówki kolumn opisujące ich zawartość (rys. 13.5, s. 182). Nazwy kolumn tabeli są reprezentowane w dokumencie w postaci **pól korespondencji seryjnej**. Możemy je wstawić do dokumentu i formatować jak zwykły tekst. Każdy wiersz listy adresowej dotyczy jednego adresata i będziemy go nazywali **rekordem**.

Pole korespondencji seryjnej

Rekord

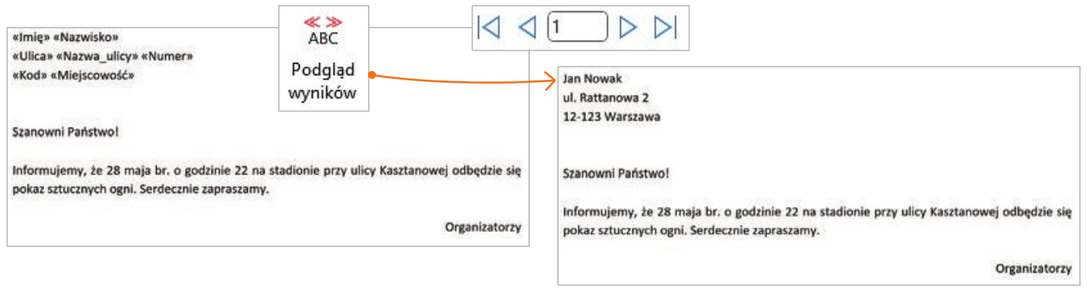
Dobra rada

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



Rys. 13.5. 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. 13.6).



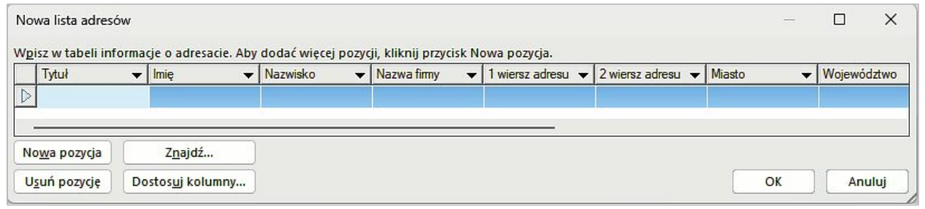
Rys. 13.6. Podgląd wyników

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 wiele kolumn i pól (rys. 13.7, s. 183). 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...**

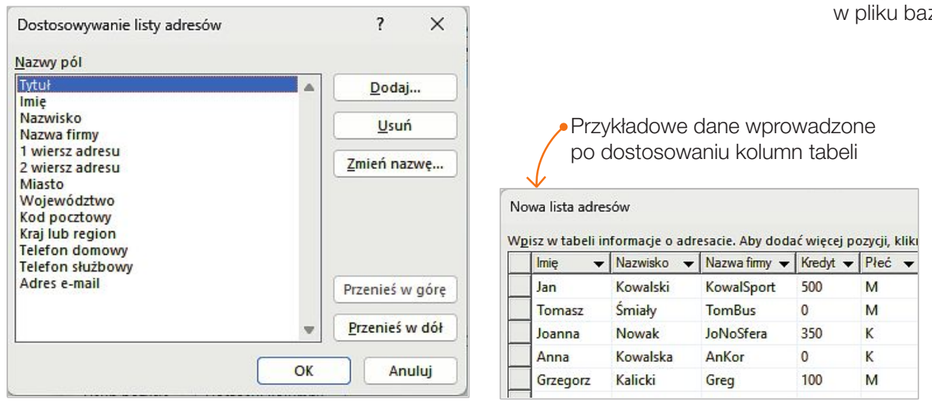


Rys. 13.7. Wstawianie nowej listy adresatów

Liczbę kolumn oraz ich nazwy możemy dostosować do naszych potrzeb. Przycisk **Dostosuj kolumny...** uruchamia okno, które pozwala wprowadzać zmiany w polach korespondencji (rys. 13.8).

Warto wiedzieć

Dane, które podasz po wybraniu opcji **Wpisz nową listę**, są zapisywane w pliku bazy danych.

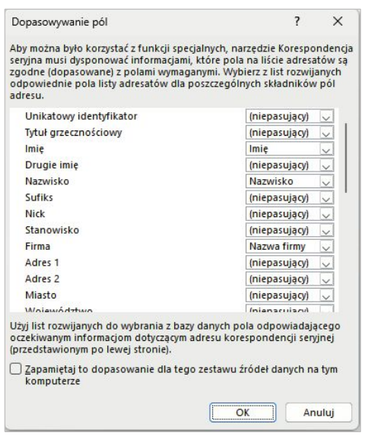


Rys. 13.8. Własny zestaw kolumn i przykładowe dane adresatów

Ćwiczenie 1

Zmodyfikuj domyślną listę pól korespondencji seryjnej do postaci pokazanej w tabeli na rysunku 13.8 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. 13.9). 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. 13.9. Przypisywanie pól do listy

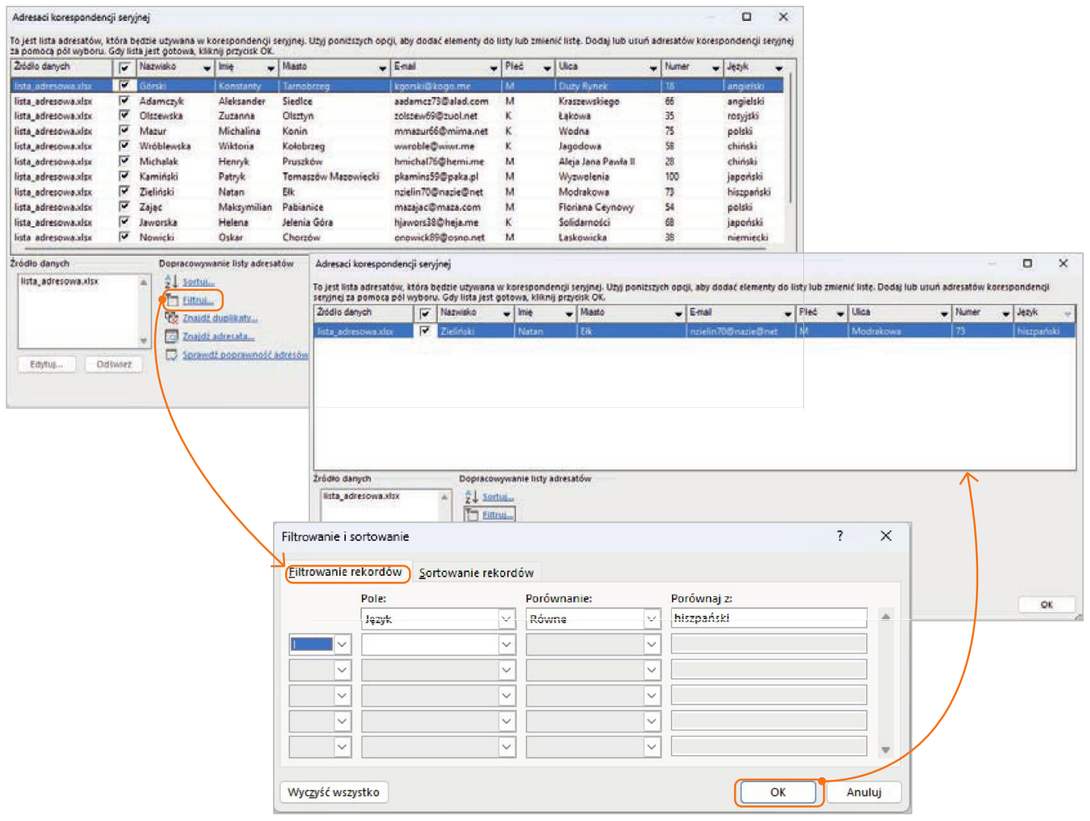
13.3. Filtrowanie adresatów i personalizacja listów

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...**).

Załóżmy, że zarządzamy grupą kilkuset profesjonalnych tłumaczy specjalizujących się w różnych językach obcych. Chcielibyśmy wysłać korespondencję jedynie do osób znających język hiszpański. W tym celu wybieramy **Korespondencja → Rozpoczynanie korespondencji seryjnej → Edytuj listę adresatów**, a w oknie, które się pojawi, wybieramy **Filtruj...** (rys. 13.10).



Rys. 13.10. Filtrowanie listy adresatów

Filtrowanie, w którym jako kryterium przyjęliśmy język hiszpański, ograniczyło listę adresatów do trzech pozycji.

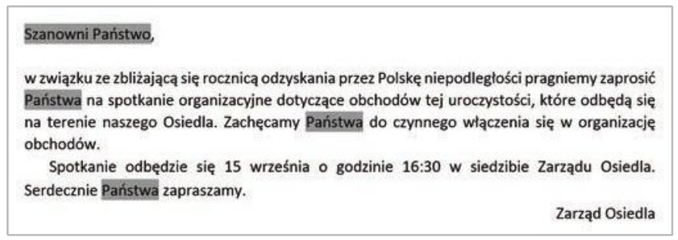
> **Ćwiczenie 2**
>
> Plik, który otrzymasz od nauczyciela (np. *firmy.csv*), zawiera dane adresowe 1000 firm. Użyj filtrowania, aby na liście adresatów korespondencji seryjnej wyświetlić tylko firmy z województwa małopolskiego, które nie mają zobowiązań kredytowych.

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.

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 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 13.11.



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. 13.11. 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.

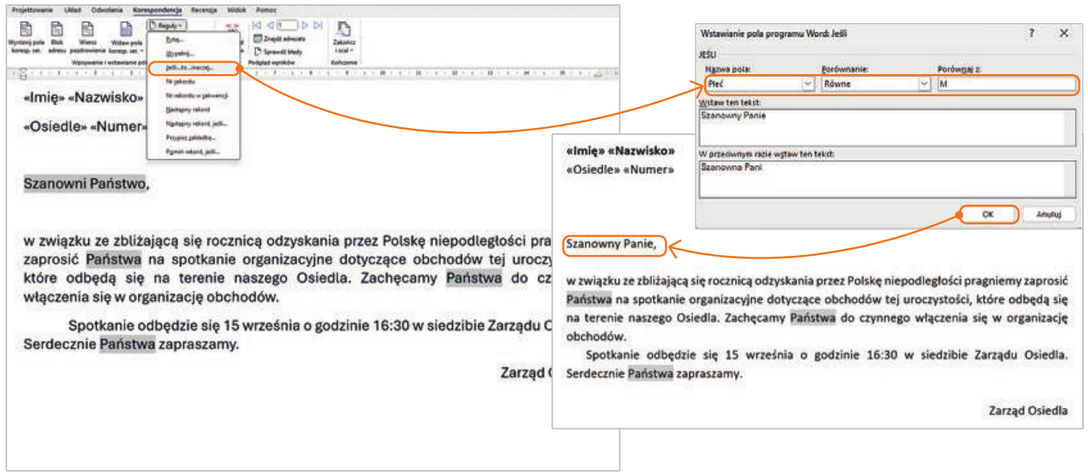
Do szablonu wiadomości podłączymy adresatów z pliku Excela, w którym znajdują się dane pogrupowane w następujące kolumny: `Imię`, `Nazwisko`, `Osiedle`, `Numer` oraz `Płeć`. Na początku dodamy pola korespondencji seryjnej z danymi adresowymi.

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

Dobra rada

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

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. 13.12).



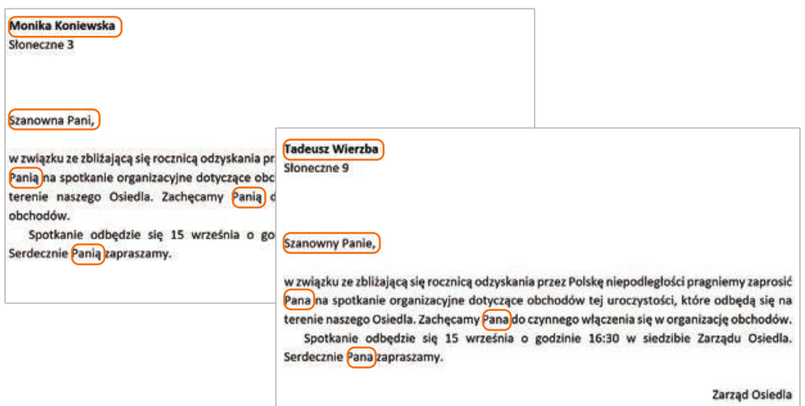
Rys. 13.12. Reguła, która zmienia zwrot grzecznościowy w zależności od płci adresata

Warto wiedzieć

W rozbudowanych tabelach informację o płci oznacza się często literami K i M (lub z ang. F i M). 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.

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 13.12. Efektem będzie personalizacja treści, którą zobaczymy po włączeniu podglądu wyników (rys. 13.13).



Rys. 13.13. Spersonalizowana korespondencja

Ćwiczenie 3

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

13.4. Zarządzanie danymi korespondencji seryjnej w arkuszu kalkulacyjnym

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 13.14 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.

Dobra rada

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

	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. 13.14. 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.

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.

Funkcja MAX.K

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 13.15 przedstawiono zastosowanie funkcji MAX.K do wyłonienia dziesięciu największych wartości w kolumnie Liczba punktów.

Warto wiedzieć

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

```
=MAX.K([Liczba punktów]; 10)
```

Zakres komórek z danymi do przeszukiwania

Numer porządkowy liczby w zbiorze posortowanym malejąco

Rys. 13.15. Formuła z użyciem funkcji MAX.K

Kompletna formuła ma postać:

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

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. 13.16).

Warto wiedzieć

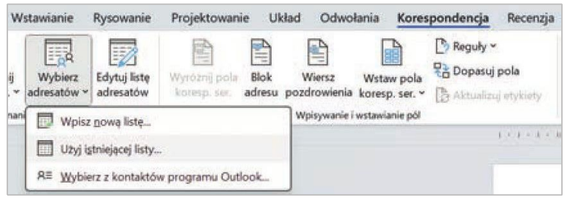
Odwołanie strukturalne do komórki tabeli o nagłówku jednowyrazowym, np. Miasto, to: [@Miasto]. Jeśli nagłówek składa się z wielu wyrazów, np. Miasto partnerskie, to ma on budowę: [@[Miasto partnerskie]].

	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	Nazwisk	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	Wojciechc	30-04-2019	143	tak
25	U7	Katarzyna	Dobkowsk	06-05-2019	141	tak
30	U10	Jakub	Turzyński	29-06-2019	139	tak

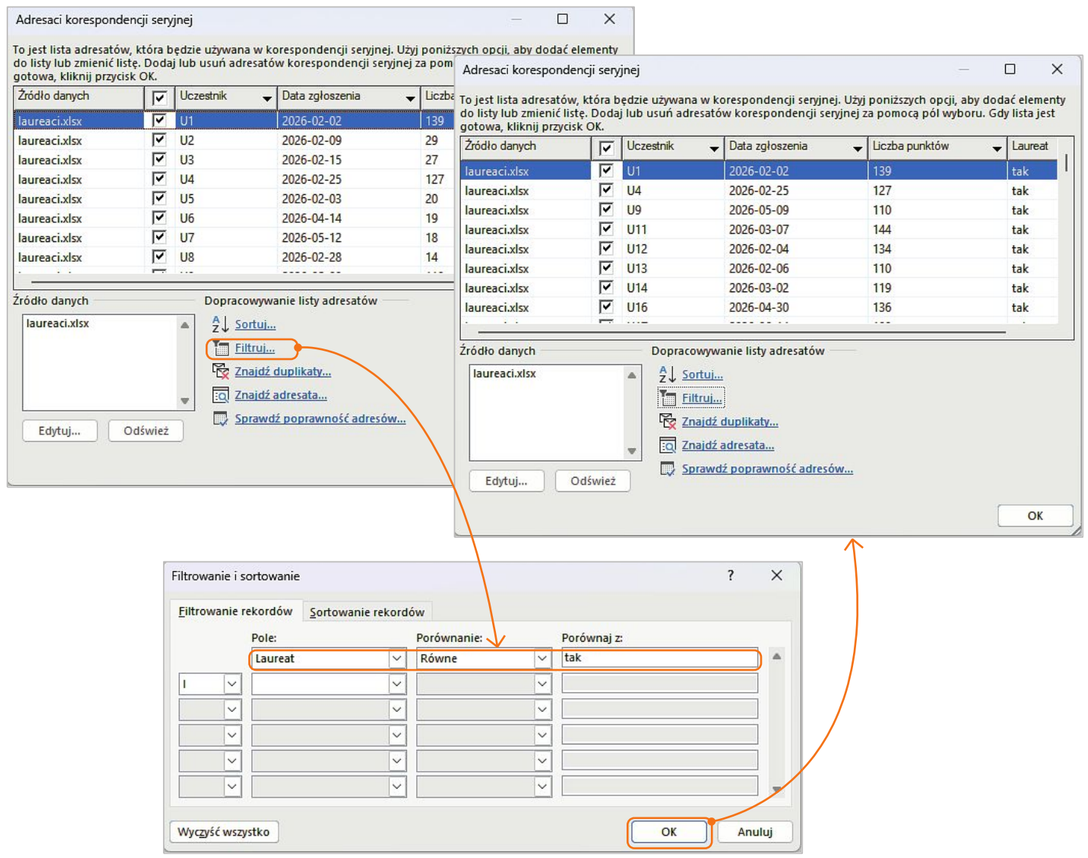
Rys. 13.16. Wytypowanie laureatów konkursu

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. 13.17 , s. 189).



Rys. 13.17. 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 listę, dodając regułę, w której pole Laureat będzie równe tak (rys. 13.18).



Rys. 13.18. Filtrowanie listy uczestników

Warto wiedzieć

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

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.

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.

13.5. Kończenie pracy z korespondencją seryjną

Scalanie dokumentów korespondencji seryjnej

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**.

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.

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.

*** 9 Za pomocą korespondencji seryjnej oraz z wykorzystaniem graficznych możliwości edytora tekstu przygotuj dyplom ukończenia szkoły. Zadbaj o to, aby znalazły się w nim odpowiednie informacje, które program pobierze z zewnętrznego pliku.

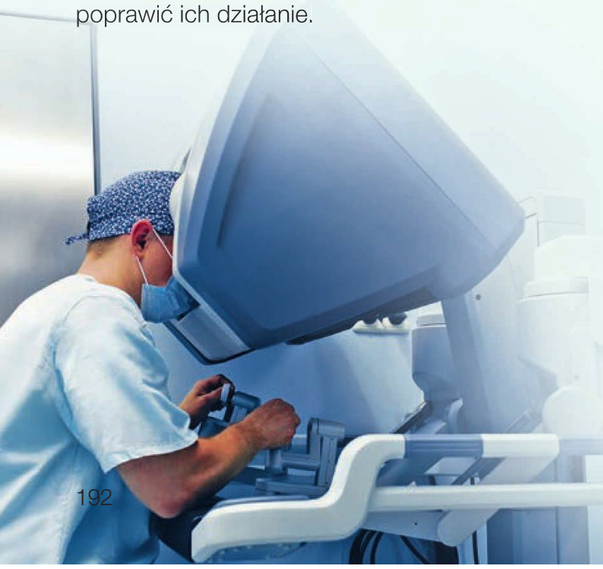
Rozwiązywanie problemów z wykorzystaniem komputera

Wyobraź sobie, że...

...po kilku latach pracy jako organizatorka eventów zdobywasz rozpoznawalność i zaufanie klientów. Planujesz duże wydarzenie dla swojej społeczności i chcesz przyspieszyć proces wysyłania zaproszeń. Wykorzystujesz do tego znajomość tworzenia **korespondencji seryjnej**, dzięki czemu możesz **personalizować treść wiadomości**, korzystając z przygotowanego **zestawu danych**. Zdecydowanie przyspiesza to twoją pracę, a zaoszczędzony czas przeznaczasz na aktywność fizyczną.

...z grupą znajomych pracujesz nad pierwszą własną aplikacją komputerową działającą w przeglądarce internetowej. Będzie to aplikacja edukacyjna przeznaczona dla dzieci. Utknęliście jednak nad pewnym algorytmem, który miał rozwiązywać jeden z problemów. Z pomocą przychodzi wam **sztuczna inteligencja** – a dokładniej **model językowy**, któremu przekazujecie jasno sformułowany **prompt**, czyli polecenie opisujące problem. Dzięki udzielonym podpowiedziom z powodzeniem kończycie i publikujecie swój projekt.

...trafiasz do firmy, w której montuje się nowoczesne samochody elektryczne. Większość prac na linii produkcyjnej wykonują **roboty**, a twoim zadaniem jest poszukiwanie możliwości optymalizacji ich działania. Dzięki doświadczeniom w **programowaniu robotów** potrafisz analizować i przetwarzać dane, które wykorzystujesz w **symulatorach robotów**, próbując poprawić ich działanie.



...po studiach medycznych zamierzasz zostać chirurgiem. Trafiasz na rezydenturę do szpitala, w którym operuje się również za pomocą **specjalistycznych robotów** chirurgicznych, wykorzystujących **serwomotory**. Jesteś pod wrażeniem tego, jak szybko pacjenci wracają do zdrowia po takich operacjach oraz jak dużą zręczność i precyzję robot zapewnia lekarzowi. Masz okazję uczestniczyć w szkoleniu, na którym uczysz się obsługi **robota na symulatorze**. Z czasem zaczynasz operować pacjentów.

Projekt zespołowy

Pułapki cyfrowego świata

Cyfryzacja obejmuje dziś cały świat. Coraz większe znaczenie mają e-usługi i nowe technologie wspierane przez sztuczną inteligencję (AI). Te rozwiązania pojawiają się w niemal każdej dziedzinie życia i sprawiają, że musimy podejmować trudne decyzje. W ramach projektu zespołowego przygotujcie się do dyskusji na ważne tematy związane z cyfryzacją. Wymianę poglądów przeprowadźcie w formie dyskusji panelowej lub debaty oksfordzkiej.

Cel projektu

Rozwijanie umiejętności argumentowania, prezentowania i pracy zespołowej przez przygotowanie i przeprowadzenie dyskusji panelowej lub debaty oksfordzkiej na temat społecznych skutków postępu w cyfryzacji.

To trzeba wiedzieć

Czym jest dyskusja panelowa?
Dyskusja panelowa (zwana też panelem) jest uważana za skuteczny sposób uczenia się, ponieważ łączy wypowiedzi ekspertów z możliwością zadawania pytań przez słuchaczy. Najpierw każdy z ekspertów krótko przedstawia swój punkt widzenia na dany temat – podczas wypowiedzi może korzystać z prezentacji multimedialnej. Po wysłuchaniu wszystkich ekspertów uczestnicy panelu mogą im zadawać pytania. Całą dyskusję prowadzi moderator.

Czym jest debata oksfordzka?
Debata oksfordzka to sformalizowany sposób obrony lub obalenia postawionej tezy. Punktem wyjścia jest sformułowanie takiej tezy, którą można uzasadniać merytoryczną argumentacją. Debacie przewodniczą marszałek oraz sekretarz. Argumenty przedstawiają dwie drużyny: jedna broni postawionej tezy, druga stara się ją obalić. Przebieg debaty ocenia jury. Istotnym elementem są głosowania publiczności (na początku, w trakcie oraz na końcu debaty), które pokazują, jak zmienia się stanowisko publiczności wobec tezy.

Propozycje tematów

1. Prywatność w cyfrowym świecie – przywilej czy standard?
2. Czy biały wywiad jest zagrożeniem dla prywatności?
3. Jak bezpiecznie kształtować cyfrową tożsamość?
4. Czy sztuczna inteligencja spowoduje, że zadawanie zadań domowych w szkole straci sens?
5. Telefon komórkowy gromadzi zbyt dużo danych osobowych.
6. Rozwój sztucznej inteligencji może ograniczać ludzką kreatywność i samodzielność.
7. Popularność zakupów internetowych spowoduje zahamowanie inwestycji w handlu detalicznym.
8. Nowe technologie zagrażają jakości edukacji.

Etapy projektu

1. Zapoznajcie się z zasadami przeprowadzania dyskusji panelowej oraz debaty oksfordzkiej, a następnie zdecydujcie się na jedną z tych form.
2. Wybierzcie temat dyskusji – z podręcznika lub zaproponowany przez nauczyciela.
3. Ustalcie z nauczycielem grupę docelową. Rozważcie zaproszenie gości.
   W przypadku dyskusji panelowej ustalcie, kim będą słuchacze (wasi rówieśnicy, uczniowie klas pierwszych, nauczyciele).
   W przypadku debaty oksfordzkiej określcie, kto może odnieść korzyści edukacyjne z udziału w debacie.
4. Ustalcie role uczestników oraz czas trwania wydarzenia i poszczególnych wypowiedzi.
   W przypadku dyskusji panelowej wyznaczcie moderatora, ekspertów oraz słuchaczy.
   W przypadku debaty oksfordzkiej (zakładając, że funkcję marszałka pełni nauczyciel) przydzielcie sobie pozostałe role: sekretarza, członków drużyn, jury i publiczności.
5. Przeanalizujcie temat i zaplanujcie szczegółowy przebieg wydarzenia.
   W przypadku dyskusji panelowej podzielcie temat na zagadnienia, które zostaną omówione przez ekspertów; możecie przygotować prezentacje lub inne materiały pomocnicze.
   W przypadku debaty oksfordzkiej ustalcie szczegółowy plan debaty; wylosujcie, która drużyna będzie bronić tezy, a która ją obalać.
6. Zorganizujcie spotkanie w grupach.
   W przypadku dyskusji panelowej omówcie swoje role, pytania i przebieg dyskusji.
   W przypadku debaty oksfordzkiej opracujcie argumenty, pytania oraz przypomnijcie sobie zasady debaty.
7. Przeprowadźcie debatę oksfordzką lub dyskusję panelową na wybrany wcześniej temat.
8. Podsumujcie na kolejnej lekcji wybraną metodę wymiany poglądów. W podsumowaniu powinna zabrać głos co najmniej jedna osoba, która pełniła jakąś funkcję podczas debaty lub dyskusji.

Przykładowe narzędzia

FreeMind, **Freeplane** – narzędzia do tworzenia map myśli.
Microsoft Whiteboard – tablica online do planowania, notowania i współpracy zespołowej.
Microsoft Sway – narzędzie do tworzenia prezentacji multimedialnych i prostych stron WWW, umożliwiające łączenie własnych materiałów z zasobami z Internetu.
Microsoft PowerPoint, **Keynote** – programy do tworzenia prezentacji w formie slajdów.
Canva – narzędzie do projektowania m.in. plakatów, infografik i prezentacji.
Microsoft OneDrive, **iCloud**, **Google Drive** – dyski wirtualne do przechowywania materiałów.
Microsoft Teams, **Zoom**, **Google Meet** – narzędzia do prowadzenia spotkań zdalnych, np. w przypadku zdalnego udziału gości.

Dodatki

1. 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".

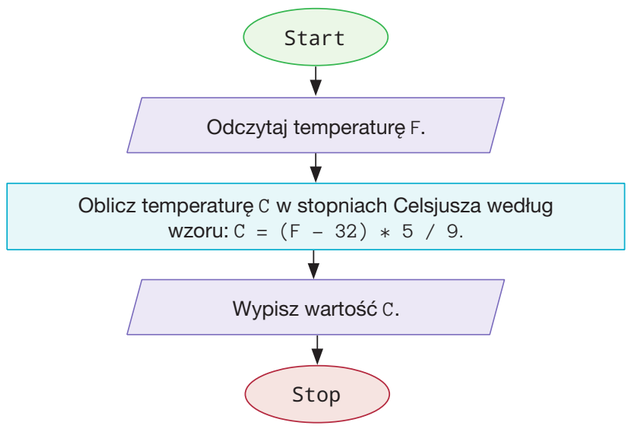
Lista kroków

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 ich konsekwentne 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 Python i C++:

Python

```python
n = int(input())
p = 1
while (p + 1)*(p + 1) <= n:
    p = p + 1
print(p)
```

C++

```cpp
#include <iostream>
using namespace std;
int main()
{
    int n; cin >> n;
    int p = 1;
    while ((p + 1)*(p + 1) <= n)
    {
        p = p + 1;
    }
    cout << p << endl;
    return 0;
}
```

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.

2. 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

Przykładowe 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 Python i C++

Porównywanie dwóch wartości w językach programowania wykonuje się za pomocą operatorów porównań (relacji). Wynikiem ich działania są wartości `True` albo `False` w języku Python lub `true` albo `false` w języku C++.

Zapis w matematyce	Zapis w językach Python i C++
=	==
≠	!=
>	>
<	<
≥	>=
≤	<=

Operatory logiczne w językach Python i C++

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 poprzedniej stronie.

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`.

W języku C++ zamiast operatorów `&&`, `||` oraz `!` można używać odpowiednio operatorów: `and`, `or` i `not`.

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`.

3. Typy danych w języku Python

W języku Python typ zmiennych określany jest, już po uruchomieniu programu, na podstawie wartości, które są przypisane zmiennym.

Typ napisowy

Napisy, czyli tzw. łańcuchy znaków, Python zapisuje w pamięci komputera jako obiekty klasy `str` (patrz dodatek 7). Ze zmiennej typu napisowego korzysta się jak ze zmiennej każdego innego typu. Można np. nadać jej wartość za pomocą operatora przypisania:

```
napis = "ABC"
```

Zamiast cudzysłowów można użyć apostrofów: napis = `'ABC'`.

Aby odwołać się do konkretnych znaków w łańcuchu, zapisujemy nazwę zmiennej i numer znaku w nawiasach kwadratowych (znaki numerowane są od 0). Na przykład `print(napis[1])` wyświetli na ekranie literę B. Możemy też używać indeksowania ujemnego i indeksowania pustego. Instrukcja `print(napis[-1])` wyświetli ostatni znak, czyli C. Natomiast instrukcja `print(napis[1:])` wyświetli wszystkie znaki, zaczynając od tego o indeksie 1. Do porównywania napisów stosuje się operatory relacji <, >, <=, >=, == i !=.

Napisy można scalać za pomocą operatora +. Na przykład operacja `"ABC" + "DE"` skutkuje utworzeniem zmiennej o wartości `"ABCDE"`. W poniższej tabeli zebrano kilka najczęściej używanych funkcji dla typu napisowego.

Nazwa funkcji	Parametry	Wynik	Przykład zastosowania
`chr()`	Liczba oznaczająca pozycję znaku w tabeli Unicode	Jednoznakowy napis typu `str` odpowiadający znakowi o danym kodzie Unicode	Instrukcja: `print(chr(97))` Wynik: a
`len()`	Łańcuch znaków, lista lub krotka	Liczba elementów obiektu lub struktury danych, np. łańcucha znaków, listy lub krotki	Instrukcja: `print(len("abc"))` Wynik: 3
`ord()`	Jednoznakowy napis	Kod Unicode dla znaku zapisanego w zmiennej	Instrukcja: `print(ord("a"))` Wynik: 97

`type()`	Nazwa obiektu lub zmiennej	Typ danych zapisany w obiekcie lub zmiennej	Instrukcja:
			`print(type("abc"))`
			Wynik:
			`<class 'str'>`

Typy liczbowe

Wśród typów liczbowych wyróżnia się typy całkowite oraz 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`

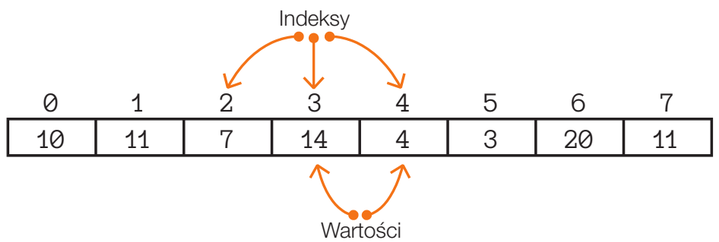
Uwaga: Zapis `e-12` oznacza notację wykładniczą (tj. czynnik 10^{-12}).

Typ logiczny

Do deklarowania zmiennych logicznych używa się typu `bool`, który przyjmuje wartości `True` i `False`. Są one słowami kluczowymi jezyka Python. Wartość `False` reprezentowana jest w pamięci komputera przez liczbę 0, a wartość `True` – przez liczbę 1.

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]`.



W języku Python możemy też używać indeksowania ujemnego i indeksowania pustego. Dla powyższej listy instrukcja `print(L[-1])` wyświetli ostatni element listy, czyli `11`, natomiast instrukcja `print(L[1:])` wyświetli wszystkie elementy, zaczynając od tego o indeksie 1.

Krotki

Krotka to struktura danych podobna do listy. Różnica między nimi polega na tym, że zawartości krotki nie można zmieniać.
Aby utworzyć krotkę, należy podać jej nazwę, następnie znak równości, a na końcu tworzące ją elementy. Elementy wchodzące w skład krotki rozdzielamy przecinkiem:

```
Krotka = 1, 2, 3, 4
```

Można użyć dodatkowo nawiasu okrągłego:

```
Krotka = (1, 2, 3, 4)
```

Krotka, podobnie jak lista, obsługuje indeksowanie. Do pierwszego elementu krotki odwołamy się, pisząc: `Krotka[0]`.

4. Podstawowe typy instrukcji w języku Python

Instrukcja warunkowa

Składnia prostej instrukcji warunkowej `if`:

```
if warunek:
    instrukcja1
    instrukcja2
    instrukcja3
    ...
```

Warunek podawany po słowie `if` powinien być wyrażeniem typu logicznego (patrz: dodatek 2).

Składnia instrukcji warunkowej `if else` (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))
```

5. 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 modułu `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	

6. 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`
`finally`	`global`	`lambda`	`None`
`nonlocal`	`pass`	`raise`	`try`
`with`			

7. Obiekty, klasy i metody w języku Python

Język Python jest językiem obiektowym. Oznacza to, że wykorzystuje się w nim obiekty mające cechy zdefiniowane w klasach, do których należą. Oprócz opisu cech obiektów definicja klasy zawiera również metody, czyli operacje, które można wykonywać na należących do niej obiektach. Jeśli tworzymy grę, klasa może być opisem postaci, na który składają się np.: imię, liczba punktów doświadczenia, wygląd. Obiektem w tej klasie będzie konkretna postać, np. o imieniu Alex, która ma 25 punktów doświadczenia, wygląd maga.
Metodami dla obiektu w tej klasie mogą być np.: bieg, podskok, użycie magicznej różdżki.

Klasa `str`

Jedną z wielu wbudowanych klas języka Python jest klasa `str`, opisująca obiekty zbudowane z łańcuchów znaków. Aby utworzyć zmienne (obiekty) należące do tej klasy, wystarczy wpisać ich wartości w cudzysłowach lub apostrofach.

```
napis1 = "ABC"
napis2 = 'XYZ'
```

Choć zmienne `napis1` i `napis2` zawierają różne dane, należą do tej samej klasy. Metody to funkcje zdefiniowane w tej klasie. Aby je zastosować, po nazwie obiektu dodajemy kropkę, a następnie wpisujemy nazwę metody i parametry podane w nawiasie:
`obiekt.metoda(parametry)`.

Metody najczęściej używane dla klasy `str`

Nazwa metody	Parametry	Wynik	Przykład zastosowania metody dla wartości `"Ala ma kota"` zapamiętanej w zmiennej `s`
`find(szukane)`	Szukany wzorzec	Pozycja pierwszego wystąpienia wzorca w napisie. Jeśli dany wzorzec nie występuje, wynikiem jest liczba –1	Instrukcja: `print(s.find("ma"))` Wynik: 4
`islower()`	–	Efekt sprawdzenia, czy napis składa się wyłącznie z małych liter (wartość `True` albo `False`)	Instrukcja: `print(s.islower())` Wynik: `False`
`split(separator)`	`separator` – domyślnie jest nim odstęp (spacja)	Lista napisów, na które napis został rozdzielony	Instrukcja: `print(s.split())` Wynik: `"Ala"`, `"ma"` i `"kota"`

8. 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 niewiększe 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 niewiększa niż 6	`float a = 2.45123;`
`double`	Bardzo małe liczby, które trzeba zapamiętać z większą dokładnością (niewięcej niż 15 cyfr znaczących)	`double a = 4.00000023e-12;`
`double`	Bardzo duże liczby (z częścią ułamkową) lub liczby, które trzeba zapamiętać z większą dokładnością (niewięcej 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

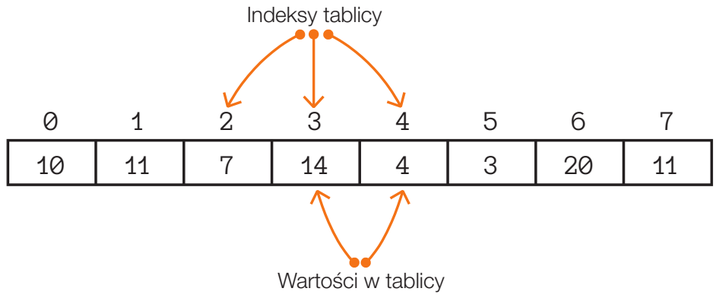
Pojedyncze znaki można zapisywać w zmiennych typu `char`. Na przykład deklaracja `char litera='a';` to deklaracja zmiennej, w której następnie zapisano literę `a`. Napisy, czyli łańcuchy znaków, zapisuje się z użyciem zmiennych typu `string`. Typ ten w sensie ścisłym nie jest typem prostym C++, ale ma 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

9. Podstawowe typy instrukcji w języku C++

Instrukcja warunkowa

Składnia prostej instrukcji warunkowej `if`:

Pojedyncza instrukcja	Wiele instrukcji

`if (warunek) instrukcja;`	`if (warunek)`
	`{`
	`  instrukcja1;`
	`  instrukcja2;`
	`  ...`
	`}`

Warunek podawany po słowie `if` powinien być wyrażeniem typu logicznego (patrz: dodatek 2).

Składnia instrukcji warunkowej `if else` (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 powtarzania. 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.

Instrukcja wyboru `switch`

Składnia instrukcji `switch` jest następująca:

```
switch (zmienna)
{
    case wartość_1:
        // instrukcje, gdy zmienna==wartość_1
        break;
    case wartość_2:
        // instrukcje, gdy zmienna==wartość_2
        break;
    …
    case wartość_n:
        // instrukcje, gdy zmienna==wartość_n
        break;
    default:
        // instrukcje, gdy zmienna nie przyjmuje
        // żadnej z wymienionych wcześniej wartości
}
```

Wartości, jakie może przyjąć `zmienna`, muszą być typu całkowitoliczbowego. W zależności od wartości zmiennej wykonywana jest jedna z części kodu zaczynających się od słowa kluczowego `case`. Na przykład gdy `zmienna` przyjmie wartość `wartosc_2`, program wykona instrukcje od części rozpoczynającej się od `case wartosc_2`. Część `default` jest opcjonalna (może nie wystąpić) – podobnie jak część `else` w instrukcji warunkowej `if`. Dzięki instrukcji `break` można przerwać wykonywanie kolejnych instrukcji zapisanych w `switch`. Gdyby zabrakło `break`, po wykonaniu instrukcji dla danej wartości zmiennej program przeszedłby do kolejnych linii w instrukcji `switch`. Jeśli nie wystąpi część `default`, nie ma potrzeby używania `break` w części z ostatnim `case`.

```
1.  switch (dzialanie)
2.  {
3.      case 1:
4.          c = a + b;
5.          break;
6.      case 2:
7.          c = a - b;
8.          break;
9.      case 3:
10.         c = a * b;
11.         break;
12.     case 4:
13.         c = a / b;
14. }
```

10. 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.

11. 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 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 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 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`	`typename`	`union`	`unsigned`
`virtual`	`volatile`	`wchar_t`	

12. Obiekty, klasy i metody w języku C++

Język C++ jest językiem obiektowym. Oznacza to, że wykorzystuje się w nim obiekty mające cechy zdefiniowane w klasach, do których należą. Oprócz opisu cech obiektów definicja klasy zawiera również metody, czyli operacje, które można wykonywać na należących do niej obiektach. Jeśli tworzymy grę, klasa może być opisem postaci, na który składają się np.: imię, liczba punktów doświadczenia, wygląd. Obiektem w tej klasie będzie konkretna postać, np. o imieniu Alex, która ma 25 punktów doświadczenia, wygląd maga.
Metodami dla obiektu w tej klasie mogą być np.: bieg, podskok, użycie magicznej różdżki.

Klasa `string`

Po dołączeniu do programu biblioteki `string` poprzez umieszczenie w kodzie źródłowym dyrektywy `#include <string>` można utworzyć zmienne (obiekty) typu napisowego:

```
string napis1 = "ABC";
string napis2 = "XYZ";
```

Choć zmienne `napis1` i `napis2` zawierają różne dane, należą do tej samej klasy. Metody to funkcje zdefiniowane w tej klasie. Aby je zastosować, po nazwie obiektu dodajemy kropkę, a następnie wpisujemy nazwę metody i parametry podane w nawiasie:
`obiekt.metoda(parametry)`

Metody najczęściej używane dla klasy `string`

Nazwa metody	Parametry	Wynik	Przykład zastosowania metody dla zmiennej `s` typu `string` `s="Ala ma kota"`

`find(szukane)`	`szukane` – fraza lub znak	Pozycja pierwszego wystąpienia szukanej frazy w napisie	Instrukcja:
			`cout<<s.find("ma");`
			Wynik:
			`4`

`substr(pozycja,ile)`	`pozycja` – indeks znaku w napisie, `ile` – liczba znaków	Fragment ciągu znaków zapisanych od indeksu `pozycja`. Liczbę znaków określa parametr `ile`	Instrukcja:
			`cout<<s.substr(4,2);`
			Wynik:
			`ma`

`erase(pozycja,ile)`	`pozycja` – indeks znaku w napisie, `ile` – liczba znaków	Napis bez ciągu znaków o długości `ile` liczonego od indeksu `pozycja`	Instrukcje:
			`s.erase(4,3);`
			`cout<<s;`
			Wynik:
			`Ala kota`

`size()`	–	Długość ciągu znaków	Instrukcja:
			`cout<<s.size();`
			Wynik:
			`11`

Oprócz metod wymienionych w tabeli często korzysta się także ze stałej `npos`. Metoda `find` przyjmuje wartość równą tej stałej, gdy szukana fraza nie zostanie odnaleziona w tekście. Na przykład dla napisu `s="Ala ma kota"` instrukcja `s.find("mak")==s.npos` przyjmie wartość prawda.

13. 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						Zamiana zapisu liczby z systemu dziesiętnego na szesnastkowy					
150	:	2	=	75	**0** ↑	19374	:	16	=	1210	**E** ↑
75	:	2	=	37	**1**	1210	:	16	=	75	**A**
37	:	2	=	18	**1**	75	:	16	=	4	**B**
18	:	2	=	9	**0**	4	:	16	=	0	**4**
9	:	2	=	4	**1**						
4	:	2	=	2	**0**						
2	:	2	=	1	**0**						
1	:	2	=	0	**1**						
$150_{10} = 10010110_2$						$19374_{10} = 4BAE_{16}$					

14. Narzędzia grafiki 3D w programie SketchUp

Ikona narzędzia	Opis
	Search SketchUp – wyszukiwanie i aktywowanie narzędzi, poleceń oraz rozszerzeń
	Select – wybieranie obiektów
	Eraser – wymazywanie krawędzi i linii
	Line – rysowanie linii prostych
	Rectangle – rysowanie prostokąta
	Push/Pull – dodawanie trzeciego wymiaru do dwuwymiarowych ścianek
	Move – przesuwanie obiektu
	Rotate – obracanie obiektu
	Scale – zmiana rozmiaru obiektu
	Paint – nakładanie koloru/tekstury na modelowane obiekty lub poszczególne ścianki
	Orbit – obracanie kamery wokół modelu
	Pan – przesuwanie widoku w poziomie/pionie
	Tape Measure – mierzenie odległości oraz tworzenie linii i punktów konstrukcyjnych
	Zoom – przybliżanie/oddalanie widoku względem obiektu
	Lasso – zaznaczanie obiektów
	Sample Material – pobieranie materiału użytego na obiekcie
	Circle – rysowanie okręgu; kolejnymi kliknięciami wskazuje się: środek i promień
	Polygon – rysowanie wielokąta; wskazuje się środek, wpisuje liczbę boków (np. 3s) i naciska **Enter**
	Arc – rysowanie łuku; kolejnymi kliknięciami wskazuje się: środek, początek łuku i kąt
	2 Point Arc – rysowanie łuku; kolejnymi kliknięciami określa się: dwa punkty podstawy i promień
	3 Point Arc – rysowanie łuku przez ustawienie wycinka koła

	Pie – rysowanie wycinka koła; kolejnymi kliknięciami wskazuje się: środek, początek łuku i kąt
	Freehand – rysowanie linii odręcznych; tak narysowany kształt zamknięty staje się ścianką
	Rotated Rectangle – rysowanie prostokąta i jego obrót wokół jednej ze ścian
	3D Text – tworzenie trójwymiarowych napisów
	Offset – tworzenie obrysu wewnątrz lub na zewnątrz narysowanej ściany
	Follow Me – tworzenie brył obrotowych, wyciąganie dowolnych kształtów wzdłuż ścieżek
	Flip – odbijanie obiektów względem wybranej osi
	Outer Shell – usuwanie elementów znajdujących się wewnątrz połączonych grup lub komponentów; pozostaje tylko zewnętrzna powłoka bryły 3D
	Intersect – tworzenie krawędzi w miejscach przecięcia obiektów
	Union – łączenie obiektów w jedną bryłę
	Subtract – odejmowanie jednej bryły od drugiej
	Trim – przycinanie obiektów poprzez zachowanie tylko części wspólnych
	Split – dzielenie obiektów w miejscach przecięcia
	Dimensions – wymiarowanie elementów graficznych
	Protractor – mierzenie kątów
	Axes – zmiana orientacji i środka układu współrzędnych względem rysowanego modelu
	Text – dodawanie etykiet tekstowych do obiektów lub sceny
	Section Plane – tworzenie płaszczyzny przecinania brył 3D; umożliwia podgląd wewnętrznej struktury modelu
	Tag – przypisywanie obiektom etykiet w celu organizacji i kontroli widoczności
	Zoom Extents – dopasowanie widoku sceny tak, aby wszystkie obiekty znajdujące się w niej były widoczne i wyśrodkowane
	Zoom Window – powiększanie konkretnej części obiektu
	Position Camera – umieszczanie kamery na zadanej wysokości
	Look Around – obracanie i oglądanie sceny wokół stałego punktu
	Walk – poruszanie się po scenie, jak podczas spaceru; z klawiszem **Shift** porusza się w górę/dół, z **Ctrl** – biegnie

15. 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)	Dodaje 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()	Zwraca 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 określoną liczbę znaków ciągu tekstowego od znaku o podanej pozycji.
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.

Indeks

A
akcelerometr 165
algorytm Euklidesa (wersja z dzieleniem) 15, 67
—— (wersja z odejmowaniem) 13, 65
— naiwny 9, 61
— wyszukiwania wzorca w tekście 42, 96
analiza częstości liter 55, 109
atak siłowy (brute-force) 54, 108

B
biblioteka `string` 86

C
chatbot 155
ciąg Fibonacciego 24, 77
— liczbowy 21, 74
— nieskończony 21, 74
— skończony 21, 74
czujnik dotykowy 163
— kolorów 163
— podczerwieni 163
— temperatury 163
— ultradźwiękowy 163
— żyroskopowy 163

D
deszyfrowanie 47, 101
dokument główny 179
— końcowy korespondencji seryjnej 179

E
ekstruder 128
emoji 29, 82

F
faktura 118
filament 127
fotopolimeryzacja 130
funkcja `chr` 31
— `find` 92
— `getline` 106
— `len` 35
— `length` 90
— `MAX.K` 188
— niezwracająca wartości 17, 70
— `ord` 32
— `rfind` 92
— `str` 10
— `swap` 62

G
geometria obiektów 114
głowica drukująca 127

I
indeks ujemny 35

J
język MicroPython 165

K
kalibracja stołu drukarki 3D 132
klasa 39
klucz szyfrowania 47, 101
kod ASCII 30, 83
— liczbowy znaku 29, 82
konkatenacja (operator złączenia) 34, 87
korespondencja seryjna 179
kryptoanaliza 53, 107
kryptografia 47, 101
kryptologia 53, 107

L
liczby względnie pierwsze 18, 70

Ł
łańcuch znaków (łańcuch, napis) 30, 86

M
magnetometr 165
metoda 39
— `append` 41
— `find` 39
— `rfind` 39
model językowy 156
modelowanie 3D 114

N
najmniejsza wspólna wielokrotność (NWW) 18, 70
największy wspólny dzielnik (NWD) 9, 61
napis (łańcuch znaków, łańcuch) 30, 86

O
obiekt 39
odcinki współmierne 14, 66
odwołania strukturalne 188
operacja złączenia (konkatenacja) 34, 87
operator + 34, 87
— scalania napisów 11

P
pętle zagnieżdżone 45, 98
pole korespondencji seryjnej 181
prompt 155

R
rekord 181
robot 162
rozdzielczość wydruku 3D 134

S
scalanie dokumentów korespondencji seryjnej 190
serwomotor 163
silnik krokowy 128
slicer 134
slicing 138
stała 33, 87
— globalna 98
stół roboczy 127
symulator robota 162
sztuczna sieć neuronowa 152
szyfr Cezara 48, 102
szyfr podstawieniowy 48, 102
szyfrogram 47, 101
szyfrowanie 47, 101

T
tablica ASCII 30, 83
— znaków Unicode 29, 82
tekst jawny 47, 101
typ `char` (typ znakowy) 83
— `str` 30
— `string` 86

U
uczenie maszynowe 148
— sztucznej inteligencji 149

W
wydruk przyrostowy (addytywny) 127
wymiana filamentu 132
wyszukiwanie wzorca w tekście 42, 95

Z
zależność między NWD a NWW 18, 71
załadunek filamentu 131
zestaw danych 179

Zdjęcie na okładce: Getty Images: Moment/Eugene Mymrin.

Zdjęcia:
Agencja Gazeta: Archiwum IBM/link https://www.ibm.com/history/early-games s. 146 (Arthur Samuel); **BE&W:** Alamy Stock Photo – Christoph Keller s. 147 (Joseph Weizenbaum), Dumitru Gornet s. 38, 91, IanDagnall Computing s. 146 (Alan Tuning), Independent s. 147 (Garri Kasparow), Science History Images s. 23, 76, Mary Evans Picture Library s. 159; **Getty Images:** DigitalVision/Alistair Berg s. 58 (specjalista ds. transportu i magazynowania), s. 112 (specjalista ds. transportu i magazynowania), E+ - Just_Super s. 59, matejmo s. 56 (tło), s. 110 (tło), vm s. 58 (dział administracji), s. 112 (dział administracji), Getty Images News/Pier Marco Tacca s. 161 (robot opiekun), Hulton Archive/Heritage Images s. 14, 66, imagenavi s. 9, 45, iStock/Getty Images Plus – Alexandr Muşuc s. 192 (aplikacja komputerowa), dima_zel s. 140, Erikona s. 152, gorodenkoff s. 142 (grafik 3D), guteksk7 s. 126, JFLAURIN s. 161 (robot saperski), MarioGuti s. 58 (fotograf), s. 112 (fotograf), MF3d s. 7, NanoStockk s. 142 (architect), Peach_iStock s. 143, treedartist s. 147 (smartfon), Moment – Andriy Onufriyenko s. 172 (dłoń robota), istetiana s. 151, Kriangkrai Thitimakorn s. 26 (Taj Mahal), s. 79 (Taj Mahal), Xuanyu Han s. 113, Science Photo Library/SCIEPRO s. 48, 102; **Shutterstock:** 19 STUDIO s. 154 (laptop), 3Dimka s. 124, Ahmet Misirligul s. 192 (organizatorka eventów), AlesiaKan s. 162 (robot edukacyjny), amiak s. 56 (tło), s. 110 (tło), Angelatriks s. 129, Anton Gvozdikov s. 160 (robot humanoidalny), Artit Wongpradu s. 142 (gry komputerowe), ASdesign0714 s. 146–147 (tło), B..Robinson s. 165 (płytka micro:bit), bbernard s. 193, BEST-BACKGROUNDS s. 26 (huragan), s. 78 (huragan), Bragin Alexey s. 146 (gra planszowa Go), By Artistdesign.13 s. 154 (ilustracja na ekranie laptopa), ClassicVector s. 147 (klaps filmowy), Crop3d s. 146 (dymek, góry), s. 147 (dymki – znak zapytania, wielokropek), Dorota Szymczyk s. 161 (robot penetracyjny), Evgeniya Lystsova s. 26 (muszla), s. 78 (muszla), funkyfrogstock s. 161 (robot dron), Gerain0812 s. 192 (chirurg), goran gjorovski s. 160 (płyta główna z procesorem), Gorodenkoff s. 142 (okulary do wirtualnej rzeczywistości), s. 192 (samochody elektryczne), insta_photos s. 58 (asystentka marketingu cyfrowego), s. 112 (asystentka marketingu cyfrowego), isky1989 s. 146–147 (tło), Iv-olga s. 161 (samochód autonomiczny), Kaspars Grinvalds s. 146 (smartfon), Kryuchka Yaroslav s. 160 (robot zabawka), MarinaGrigorivna s. 128, matthew25 s. 153, Mikhail Leonov s. 24, 77, Paulo Arsand s. 22, 75, Pe3k s. 146 (postać z gry StarCraft), PP77LSK s. 160 (robot przemysłowy), pyrozhenka s. 8, 60, Roman Samborskyi s. 154 (człowiek analizujący dane), Stone/Dana Neely s. 166 (sala operacyjna), sumkinn s. 160–161 (tło), Surlan Denis s. 114; **Janusz Mazur** s. 162 (roboty edukacyjne), s. 163, 165 (przykład robota micro:Maqueen).

Obrazy wygenerowane przy pomocy AI: Chat GPT s. 133 (ciekawostka – Czy astronauci będą drukować jedzenie?), Dall-e s. 157 (pies w przestrzeni kosmicznej).

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. Adresy stron internetowych zweryfikowano 15.01.2025 r.

nowa era *Twoje mocne strony*

NOWA
Informatyka na czasie

Za co polubisz podręcznik *NOWA Informatyka na czasie 3*?

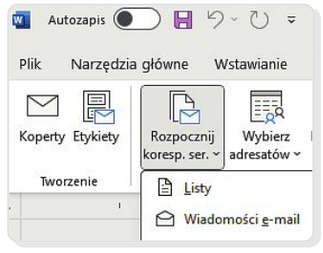
Powód 1

Jest tu dokładnie tyle treści, ile trzeba, nie ma nadmiaru. Podręcznik jest dostosowany do zmienionej podstawy programowej – wycięliśmy z niego treści, które od roku 2024 nie są wymagane.

ZMIANY W PODSTAWACH 2024
podręcznik w pełni dostosowany

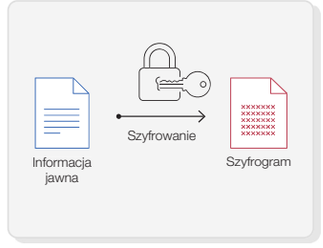
Powód 2

Dużo **zrzutów ekranu** i czytelne **infografiki**. Łatwo zorientujesz się, gdzie dokładnie znajdują się omawiane narzędzia, i nauczysz się z nich korzystać.



Powód 3

Przykłady z życia codziennego – bo pokazują, jak możesz wykorzystać wiedzę w praktyce – na przykład zaszyfrować wiadomość za pomocą prostego algorytmu szyfrującego.



Nowa Era Sp. z o.o.
www.nowaera.pl nowaera@nowaera.pl
Centrum Kontaktu: 58 721 48 00

ISBN 978-83-267-5347-3