DLA ABSOLWENTÓW SZKÓŁ PODSTAWOWYCH

Informatyka na czasie 3

Podręcznik dla liceum ogólnokształcącego i technikum

Zakres podstawowy

nowa era

Droga Nowa Ero,

Nigdy bym nie publikowała publicznie książek wydawnictw, które działają na uczciwych zasadach.

Wasza firma jednak promuje masowy dodruk, całkowicie niepotrzebnych książek, które mogłyby zastąpione wersjami elektronicznymi!

Co prawda e-booki są dostępne na waszej stronie, jednak:

- W przeciwieństwie do fizycznej książki, licencja na e-book kończy się po roku. Oznacza to, że jeżeli moja córka chciałaby powtórzyć sobie całą wiedzę do matury, musiałabym jej kupić wszystkie wasze książki od nowa.
- Waszych e-booków nie da się pobrać! Wymagają one dostęp do internetu, co uniemożliwia ich użycie na naszej wsi, gdzie zasięg jest ograniczony.
- Wasze e-booki nie działają na telefonach komórkowych!!!
- Wasze e-booki sprzedawane są po tej samej (albo wyższej) cenie co regularne książki. Cena e-booka powinna być niższa, gdyż e-booki wymagają elektronicznego czytnika (tabletu)!

Czas rozpocząć nową erę (o ironio), w której papier nie jest bezczelnie marnowany dla pieniędzy. Przedstawiam e-book, który spełnia wszystkie oczekiwania uczniów.

Dbajmy o środowisko, zróbmy to dla młodych pokoleń.

Spis treści

1. Algorytmika i programowanie w języku C++ 7

Kartka z historii 8

1. Algorytmy na tekstach 10

1.1. Litery i inne znaki jako liczby 10
1.2. Szukamy literówek. Łańcuchy znaków w języku C++ 14
1.3. Lista zakupów. Usuwanie duplikatów w tekście 21
1.4. Wyszukiwanie wzorca w tekście 23

2. Szyfrujemy wiadomości 30

2.1. Podstawowe pojęcia kryptografii 30
2.2. Szyfry przestawieniowe. Szyfr kolumnowy 31
2.3. Szyfry podstawieniowe. Szyfr Cezara 35
2.4. Metody łamania klasycznych szyfrów 40

3. Porządek ma znaczenie, czyli sortujemy liczby 46

3.1. Do czego służy sortowanie? 46
3.2. Sortowanie a wyszukiwanie 50
3.3. Algorytmy sortowania 52
3.4. Sortowanie bąbelkowe 53
3.5. Sortowanie przez wstawianie 56

4. Podejście zachłanne w rozwiązywaniu problemów 62

4.1. Problemy optymalizacyjne. Algorytm zachłanny 62
4.2. Kolorowanie mapy metodą zachłanną 64
4.3. Wydawanie reszty metodą zachłanną 64
4.4. Problem kinomana 71

5. Rekurencja 77

5.1. Czym jest rekurencja? 77
5.2. Rekurencyjna definicja ciągu 80
5.3. Ciąg Fibonacciego 83
5.4. Rekurencyjny algorytm Euklidesa 89

Z informatyką w przyszłość 92

2. Algorytmika i programowanie w języku Python 93
Kartka z historii 94
6. Algorytmy na tekstach 96
6.1. Litery i inne znaki jako liczby 96
6.2. Szukamy literówek 100
6.3. Lista zakupów. Usuwanie duplikatów w tekście 107
6.4. Wyszukiwanie wzorca w tekście 109
7. Szyfrujemy wiadomości 115
7.1. Podstawowe pojęcia kryptografii 115
7.2. Szyfry przestawieniowe. Szyfr kolumnowy 116
7.3. Szyfry podstawieniowe. Szyfr Cezara 120
7.4. Metody łamania klasycznych szyfrów 125
8. Porządek ma znaczenie, czyli sortujemy liczby 131
8.1. Do czego służy sortowanie? 131
8.2. Sortowanie a wyszukiwanie 135
8.3. Algorytmy sortowania 137
8.4. Sortowanie bąbelkowe 138
8.5. Sortowanie przez wstawianie 141
9. Podejście zachłanne w rozwiązywaniu problemów 147
9.1. Problemy optymalizacyjne. Algorytm zachłanny 147
9.2. Kolorowanie mapy metodą zachłanną 149
9.3. Wydawanie reszty metodą zachłanną 149
9.4. Problem kinomana 155
10. Rekurencja 161
10.1. Czym jest rekurencja? 161
10.2. Rekurencyjna definicja ciągu 164
10.3. Ciąg Fibonacciego 167
10.4. Rekurencyjny algorytm Euklidesa 173
Z informatyką w przyszłość 176
Projekt zespołowy 177

3. Rozwiązywanie problemów z wykorzystaniem komputera 179

Kartka z historii 180

11. Sterujemy robotem 182

11.1. Czym są roboty? 182
11.2. Roboty w edukacji 183
11.3. Robot LEGO MINDSTORMS EV3 186
11.4. Mikrokontroler BBC micro:bit 188
11.5. Programowanie robotów i symulatory 189
11.6. Symulacja robota EV3 w środowisku Open Roberta Lab 190
11.7. Symulacja płytki micro:bit w środowisku Microsoft MakeCode 194
11.8. Środowisko do symulacji VEXcode VR 196

12. Sztuka publikowania w sieci 202

12.1. Jak zainteresować odbiorców swoimi publikacjami? 202
12.2. Teksty informacyjne 207
12.3. Grafika i identyfikacja wizualna 209
12.4. Nagrania audio i wideo oraz wystąpienia na żywo 211
12.5. Jak poznać (lepiej) swojego odbiorcę? 217

13. Grafiki informacyjne 219

13.1. Sposoby przekazywania informacji 219
13.2. Piktogramy 221
13.3. Przykłady infografik 222
13.4. Jak stworzyć poprawną infografikę? 225
13.5. Tworzymy własną infografikę 232

Z informatyką w przyszłość 236

Projekt zespołowy 237

Dodatki 239

1. Sposoby zapisu algorytmów 240
2. Logika matematyczna i operatory logiczne 242
3. Typy danych w języku C++ 245
4. Podstawowe typy instrukcji w języku C++ 247
5. Operatory i funkcje matematyczne w języku C++ 249
6. Słowa kluczowe w języku C++ 250
7. Obiekty, klasy i metody w języku C++ 251
8. Typy danych w języku Python 252
9. Podstawowe typy instrukcji w języku Python 254
10. Operatory i funkcje matematyczne w języku Python 257
11. Słowa kluczowe w języku Python 258
12. Obiekty, klasy i metody w języku Python 259
13. Systemy liczbowe 260

Indeks 261

O czym jest podręcznik?

Podręcznik *Informatyka na czasie 3* pomoże ci pogłębić wiedzę na temat programowania algorytmów w języku C++ lub Python 3. Dowiesz się, jak można programować i symulować działanie robotów. Nauczysz się również przygotowywać atrakcyjne treści do publikacji w sieci oraz tworzyć infografiki.



Czemu służą poszczególne elementy podręcznika?

Cele lekcji

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

Zapamiętaj

Najważniejsze pojęcia omawiane na lekcji, które należy zapamiętać.

Ćwiczenie 1

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

Dobra rada

Podpowiedzi ułatwiające wykonanie omawianych czynności.

Warto wiedzieć

Informacje poszerzające wiedzę o omawianych zagadnieniach.

A to ciekawe

Ciekawostki pokazujące omawiane zagadnienia z zaskakującej perspektywy.

Kartka z historii

Wydarzenia najważniejsze dla rozwoju danej dziedziny informatyki.

Z informatyką w przyszłość

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

Oznaczenia w podręczniku

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

Podsumowanie

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

Zadania

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

Projekt zespołowy

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

Dodatki

Przydatne informacje uzupełniające wiedzę z podręcznika.

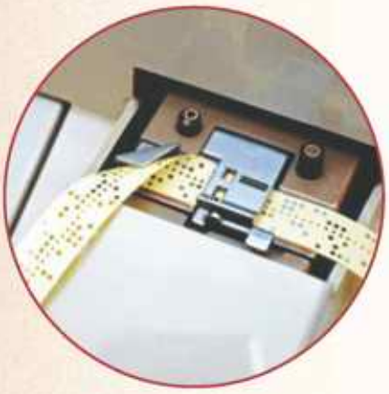


1 Algorytmika i programowanie w języku C++

1. Algorytmy na tekstach
2. Szyfrujemy wiadomości
3. Porządek ma znaczenie, czyli sortujemy liczby
4. Podejście zachłanne w rozwiązywaniu problemów
5. Rekurencja

Algorytmika i programowanie w języku C++

Sprzęt komputerowy jest użyteczny dzięki oprogramowaniu. Narzędziem do jego tworzenia są języki programowania, w których zapisuje się zaprojektowane wcześniej algorytmy, pozwalające efektywnie rozwiązywać różne problemy współczesnego świata.



1901 r.

Kamieniem milowym w rozwoju telekomunikacji był binarny **kod Baudota–Murraya**, ostatecznie ukształtowany ok. 1901 r. Dzięki niemu wysyłanie i odbieranie sygnałów elektrycznych zostało zautomatyzowane. W latach 60. XX w. zastąpił go kod ASCII.

1920 r.

W 1920 r. matematyk Jan Łukasiewicz zaproponował zapis wyrażeń arytmetycznych niewymagający nawiasów: znak operacji umieszcza się w nim przed argumentami. Jest to tzw. notacja Łukasiewicza lub **notacja polska**. Odwrócony zapis (znak operacji umieszczany za argumentami) znalazł zastosowanie w translatorach języków programowania.



1932 r.

Informacje wojskowe szyfrowane przez Niemców za pomocą Enigmy były niemożliwe do odczytania dla obcego wywiadu. **Łamanie szyfrów Enigmy** zapoczątkowali Polacy, już w grudniu 1932 r. Był to efekt wspólnej pracy wojskowych inżynierów i matematyków: Mariana Rejewskiego, Henryka Zygalskiego i Jerzego Różyckiego.



1938 r.



W 1938 r. Claude E. Shannon, amerykański matematyk i inżynier, opracował praktyczne zastosowanie dla teorii G. Boole'a, która wprowadzała wyrażenia logiczne w formie matematycznej. Wykazał, że operacje logiczne i arytmetyczne można wykonywać za pomocą **przełączników**, które otwierają lub zamykają obwód elektryczny.

1944 r.



Około 1944 r. zespół konstruktorów komputera ENIAC zapoczątkował posługiwanie się słowem „**programować**" (ang. *to program*) w dzisiejszym znaczeniu. Wcześniej używano słowa „konfigurować" (ang. *to set up*), ponieważ pierwsze komputery były programowane przez modyfikację sprzętową.

1951 r.

Grace Hopper, doktor matematyki i programistka komputerów elektromechanicznych Mark I i Mark II, tworzyła podprogramy, które można było wielokrotnie wywołać w programie głównym. W 1951 r. Hopper opracowała **translator** (nazwała go A-0), czyli program tłumaczący kody mnemotechniczne (np. ADD na oznaczenie dodawania) na binarny kod maszynowy komputera.





W 1954 r. zespół badawczy firmy IBM pod kierownictwem Johna Backusa zaprojektował język programowania **FORTRAN** (od ang. *FORmula TRANslation*), przeznaczony do obliczeń naukowo-inżynierskich. Jest on rozwijany i stosowany do dzisiaj, zwłaszcza w programowaniu superkomputerów używanych do przeprowadzania symulacji (np. modelowania reakcji chemicznych).

W 1965 r. Thomas E. Kurtz i John Kemeny z Dartmouth College opracowali język programowania **BASIC**. W latach 80. XX w. język ten był bardzo często używany do nauki programowania i pisania prostych gier na popularnych w Polsce komputerach, takich jak Commodore C64 i Amiga. Na składni jednego z dialektów tego języka opiera się język Visual Basic używany współcześnie.

```
10 LET A=112
20 LET B=88
30 LET R=A-B*INT (A/B)
40 IF R=0 THEN GOTO 80
50 LET M=N
60 LET N=R
70 GOTO 30
80 PRINT N_
```



Język programowania **Python** stworzył w 1991 r. holenderski programista Guido van Rossum. W zamierzeniu autora miał to być tzw. język skryptowy, który umożliwia szybkie przetwarzanie plików tekstowych i wykonywanie powtarzalnych zadań (np. na serwerze stron internetowych). Dziś Python jest jednym z najpopularniejszych języków programowania i ma różnorodne zastosowania. Na przykład podczas korzystania z serwisu YouTube lub narzędzi Google uruchamiamy na serwerach oprogramowanie napisane w języku Python.

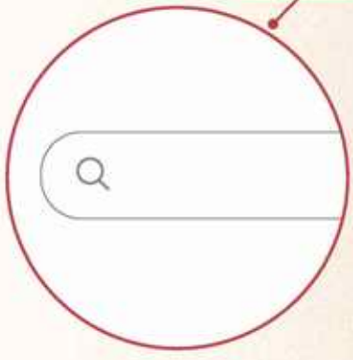
1954 r. **1962 r.** **1965 r.** **1983 r.** **1991 r.** **2018 r.**



W 1962 r. angielski informatyk Charles A.R. Hoare przedstawił nowy sposób sortowania, dziś znany pod nazwą **QuickSort**. Wymyślił go podczas stażu naukowego w Moskwie, pracując na koncepcją maszynowego przekładu tekstu z języka rosyjskiego na angielski.

Język **C++**, opracowany przez Bjarne Stroustrupa, powstał w 1983 r. jako rozszerzenie języka C. Obecnie C++ jest jednym z najważniejszych narzędzi twórców oprogramowania użytkowego oraz gier komputerowych. W tym języku tworzy się oprogramowanie systemów wbudowanych, np. sterujących pracą urządzeń AGD.





Od 2018 r. program wyszukujący Google stosuje **algorytm BERT**. Pozwala on na lepszą analizę języka naturalnego, w której uwzględnia się m.in. końcówki fleksyjne i tzw. *stop words* (np. zaimki, przyimki, spójniki), czyli słowa o mniej istotnym znaczeniu. Algorytm BERT jest praktycznym przykładem zastosowania teorii sieci neuronowych.

1. Algorytmy na tekstach

Błędy językowe, literówki i inne pomyłki w tekście zwykle świadczą o niewiedzy lub nieuwadze piszącego. Czasami jednak mogą mieć naprawdę poważne konsekwencje. Na przykład skład leku i jego dawkowanie powinny być zawsze szczegółowo i precyzyjnie opisane w ulotce. W takim wypadku 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++.

Cele lekcji

- Zrozumiesz, jak komputer zapisuje informacje tekstowe.
- Poznasz i zastosujesz algorytmy przetwarzania tekstów: porównywania tekstów oraz wyszukiwania wzorca.
- Nauczysz się używać zmiennych typu `char` i typu `string` w języku C++.
- Dowiesz się, jak przetwarzać teksty w języku C++.
- Poznasz funkcje z biblioteki `string`: `length`, `find` i `rfind`.

1.1. Litery i inne znaki jako liczby

Emoji

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.

Jeśli umieścisz na portalu społecznościowym emoji 🌻, to widzisz je dlatego, że piktogram ten znajduje się w zbiorze znaków czcionki na twoim 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 specjalnej **tablicy znaków Unicode**. W roku 2020 liczba znaków zakodowanych w tablicy Unicode przekraczała 280 000.

Kod liczbowy znaku

Tablica 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 1.1 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 1.1. 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 podstawowym zestawem znaków ASCII (skrót od ang. *American Standard Code for Information Interchange*), a w skrócie – **tablicą ASCII**. 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 1.2.

Typ `char` (typ znakowy)

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
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 1.2. Fragment tablicy 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 [ma kod $5B_{16}$, czyli 91 dziesiętnie.

Kod ASCII

Napiszemy teraz program, który wyświetli fragment tablicy ASCII pokazany w tabeli 1.2 na s. 11. 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*, który wykonuje opisane zadanie:

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. }
```

Dobra rada

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

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 wyświetlany jest znak, a po nim pojedyncza spacja, zapisana w cudzysłowie (linia 12). Zapisy w liniach 13 i 14 służą temu, aby znaki wyświetlić w wierszach po 16 znaków.

Pętla `for`, s. 247

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



Rys. 1.1. Wywołanie programu *znaki_ASCII.exe*

Na rysunku 1.1 przedstawiono wywołanie programu *Znaki ASCII*. Zauważ, że układ tabeli kodów 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. Dlatego łatwo z kodu małej litery przejść do kodu odpowiadającej jej wielkiej litery i na odwrót. Różnica w kodzie liczbowym odpowiadająca odległości dwóch wierszy wynosi 32.

Ćwiczenie 2

a. Zapisz kod źródłowy programu w pliku *znaki_ASCII.cpp*. Skompiluj kod i sprawdź działanie programu.
b. Zmodyfikuj powyższy kod źródłowy 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 = '~';
```

Dobra rada

Apostrof znajdziesz na tym samym klawiszu co znak cudzysłowu.

Gdybyśmy od razu wiedzieli, że program ma wyświetlić wszystkie znaki znajdujące się w tablicy ASCII między cyfrą 0 a znakiem ~ (patrz rys. 1.1), 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. }
```

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.

W wierszu 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.

Pętla while, s. 248

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

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
```

1.2. Szukamy literówek. Łańcuchy znaków w języku C++

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.

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 są różne!".

Sprawdzenie, czy dwa adresy są identyczne, sprowadza się do porównania wartości dwóch zmiennych, w których są one przechowywane.

Typ `string` W poprzednich tematach stosowaliśmy już zmienne **typu `string`**, czyli zmienne typu tekstowego, w których można zapamiętywać napisy składające się ze znaków ASCII.

Biblioteka `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**.

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*:

```
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. }
```

Fragment kodu źródłowego programu *Porównanie adresów*

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

W **warunku logicznym** (wiersz 17) sprawdza się, 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 nazywamy **złączeniem** lub **konkatenacją** (ang. *concatenation*). Jeśli podane napisy są różne, program wypisze na ekranie komunikat zapisany w stałej `INF_BLAD` (linia 20).

Logika matematyczna i operatory logiczne, s. 242–244

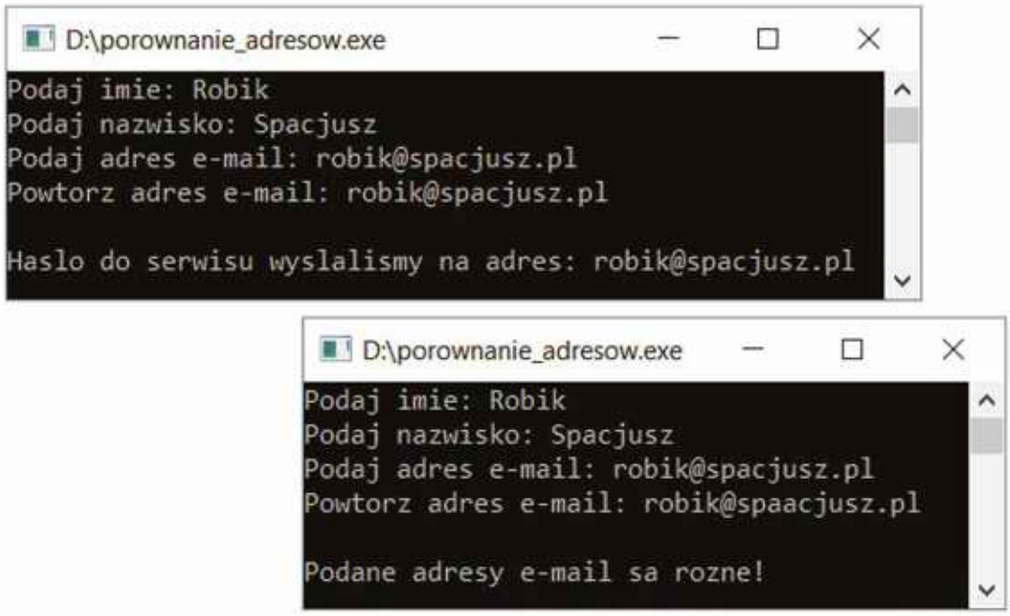
Operator +

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



Rys. 1.2. Wywołania programu *porownanie_adresow.exe*

Ćwiczenie 4

Zapisz kod źródłowy przedstawionego wcześniej programu 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

Poprzedni program 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 @.

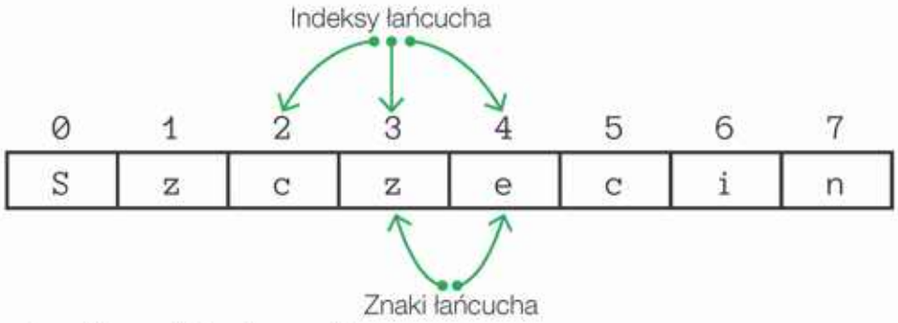
Warto wiedzieć

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 numerze $n - 1$, gdzie n to liczba znaków łańcucha (rys. 1.3). Zauważ, że w **tablicach** stosuje się podobny sposób indeksacji.

Tablice w języku C++, s. 246



Rys. 1.3. Interpretacja graficzna łańcucha znaków

Poprawność adresu e-mail możemy sprawdzić w czterech etapach. Opisuje je schemat z rysunku 1.4.

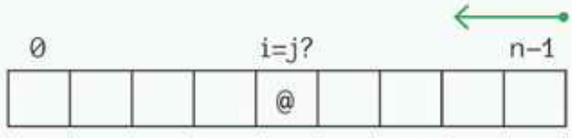
Etap 1

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.



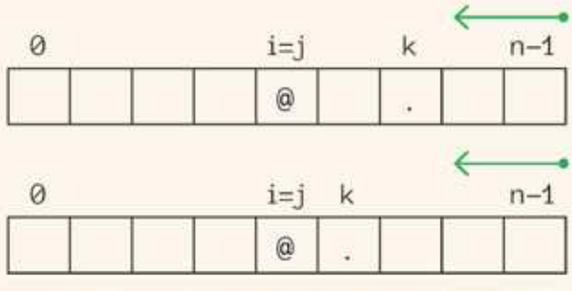
Etap 2

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.



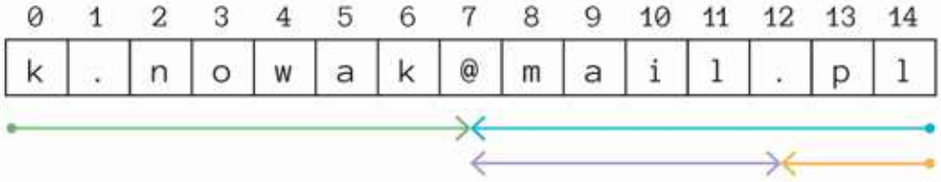
Etap 4

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. 1.4. Etapy sprawdzania poprawności podanego adresu e-mail

Przykład realizacji algorytmu pokazuje rysunek 1.5. Kolory strzałek są zgodne z kolorami tła poszczególnych etapów z rysunku 1.4 ze s. 17.



Rys. 1.5. 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 (adres[i]!='@' && i<dl-1) i++;
7.      if (adres[i]!='@' || i<2) 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 (adres[k]!='.' && k>0) 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 wierszu 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. Używaliśmy go już podczas korzystania ze struktury tablicy.

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

Logika matematyczna i operatory logiczne, s. 242–244

Oto fragment kodu źródłowego zawierający wywołanie funkcji CzyPoprawnyAdres:

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

```
1.  if (CzyPoprawnyAdres(adres_1))
2.  {
3.      cout << "Powtorz adres e-mail: "; cin >> adres_2;
4.      cout << endl;
5.      if (adres_1 == adres_2)
6.          cout << INF_OK + adres_1 << endl;
7.      else
8.          cout << INF_ROZNE_ADRESY << endl;
9.  }
10. else
11.     cout << endl << INF_BLAD_ADRESU << endl;
```

Zwróć uwagę, że funkcja CzyPoprawnyAdres jest typu logicznego, a jej wywołanie znajduje się w miejscu wyrażenia logicznego instrukcji warunkowej (linia 1). W ten sposób funkcję typu logicznego można wykorzystać jako warunek logiczny, np. w instrukcji warunkowej lub pętli.

Warto wiedzieć

Zapis w linii 1 może być również następujący:

```
if (CzyPoprawnyAdres
   (adres_1) == 1)
```

Ćwiczenie 5

a. Zapisz specyfikację do programu *Sprawdzenie adresu*. Zaproponuj brzmienie wykorzystywanych w nim komunikatów.
b. Zapisz pełny kod źródłowy programu *Sprawdzenie adresu*. Skompiluj kod i uruchom program.

A to ciekawe

Jak odróżnić człowieka od maszyny?

W wielu serwisach internetowych stosuje się rozwiązanie nazywane CAPTCHA, którego zadaniem jest upewnienie się, że formularz uzupełnia człowiek, a nie spamujący robot. Test polega zazwyczaj na przepisaniu słowa z obrazka. Fakt, że czynność tę wykonują regularnie miliony internautów, wykorzystano w projekcie reCAPTCHA. Użytkownikom wyświetla się fragmenty tekstu nieczytelne dla oprogramowania OCR (zamieniającego skan na tekst) podczas digitalizacji zasobów na użytek bibliotek cyfrowych. 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. 251

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 poznany 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

Zapis nazwy funkcji po kropce jest właściwy dla tzw. 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, która jest większa 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.

1.3. 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 maksymalnie 20 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 1.3. Analiza działania algorytmu usuwającego powtórzenia

Tabelę 1.3 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, a jedno pojawia się 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 wierszu 1 zadeklarowano stałą N i przypisano jej wartość 20. 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 wierszach 9–10 wczytujemy pierwszy, a w wierszu 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 wierszach 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.

1.4. 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.

Warto wiedzieć

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

Wykreślanka

Na rysunku 1.6 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.

R	O	M	A	N	A	L	I	C	J	A	Z	A	P	M	G	P	R
T	E	L	H	C	E	I	C	J	O	W	S	I	A	I	S	A	K
O	A	I	G	O	Z	U	Z	A	N	N	A	L	W	C	O	W	A
M	A	R	I	A	J	U	L	I	A	I	E	E	A	H	N	E	R
E	W	D	O	M	I	N	I	K	T	N	N	M	W	A	I	Ł	O
K	B	O	G	U	M	I	Ł	O	A	E	Ł	A	E	Ł	A	I	L

Rys. 1.6. Wykreślanka z zaznaczonymi imionami

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. Przyjmiemy pewne uproszczenia. Oto specyfikacja problemu:

Specyfikacja

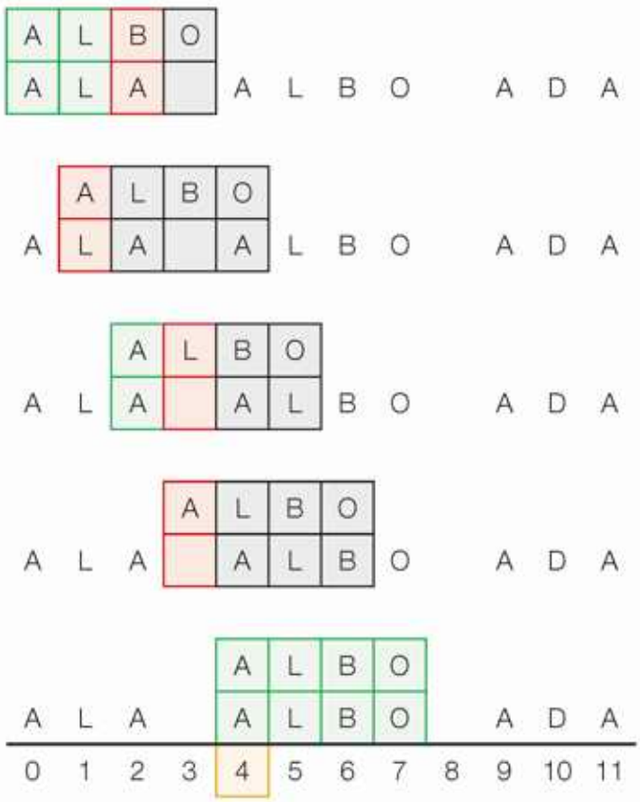
Dane: tekst zapisany wielkimi literami oraz wzorzec – łańcuch zapisany wielkimi literami.

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 1.7 pokazuje działanie algorytmu. Zwracana jest wartość 4.



Rys. 1.7. Naiwne wyszukiwanie wzorca w tekście

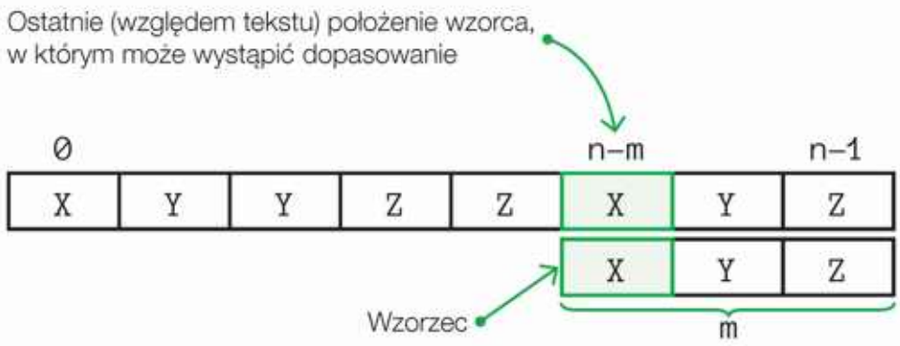
Ćwiczenie 9

a. Zapisz przebieg działania algorytmu poszukiwania w tekście ZA SZCZYTEM ZASZCZYT wzorca ZASZCZYT. Zapisz go w podobny sposób jak na rysunku 1.7.

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 numerze $n - m$, należy sprawdzać zgodność kolejnych znaków tekstu ze znakami tworzącymi wzorzec.

Dlaczego sprawdzamy do znaku o numerze $n - m$? Wyjaśnia to rysunek 1.8. Jeśli tekst ma długość $n = 9$, a wzorzec $m = 3$, to ostatni indeks ma wartość 6 (znaki indeksujemy od 0).



Rys. 1.8. Wyznaczenie indeksu ostatniego znaku tekstu, z którym porównujemy wzorzec

Oto lista kroków algorytmu:

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 wykonuj kolejno kroki 4 i 5.
   4. Wybierz fragment tekstu długości m, począwszy od znaku na pozycji równej licznikowi pętli, i porównaj go ze wzorcem.
   5. Jeśli fragment i wzorzec są identyczne, to zwróć jako odpowiedź liczbę równą licznikowi pętli i zakończ algorytm.
6. Zwróć wartość -1.

Zauważ, że 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

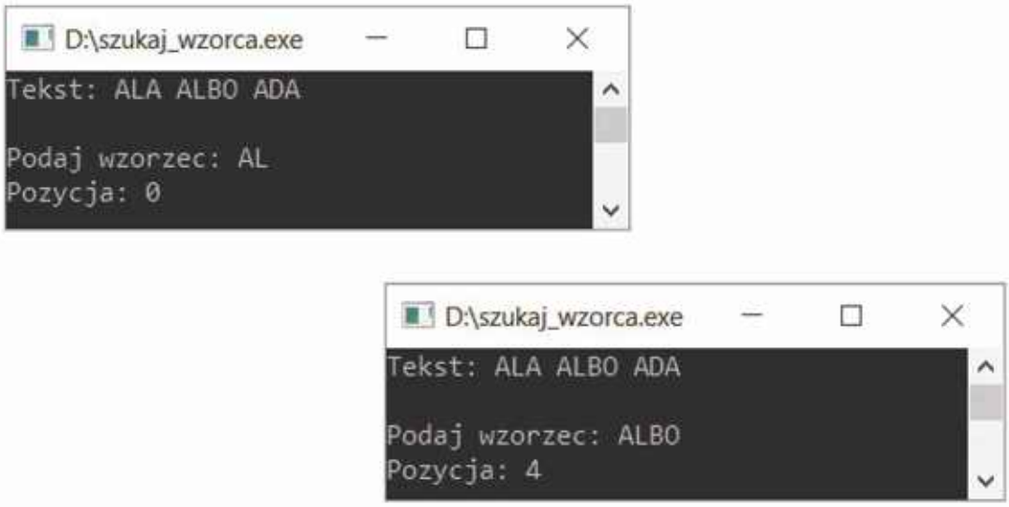
W linii 6 zastosowano **stałą globalną** o nazwie TEKST. Jest to stała, do której dostęp jest możliwy z dowolnego miejsca kodu programu, również z funkcji Porownaj, którą definiujemy w liniach 8–23. Stała globalna musi być zdefiniowana przed wystąpieniem funkcji głównej i każdej innej funkcji lub zmiennej, 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 wierszach 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).

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

Rysunek 1.9 przedstawia przykładowe wywołania programu.



Rys. 1.9. Wywołania programu *szukaj_wzorca.exe* 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 szkielet programu głównego oraz tekst, 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.

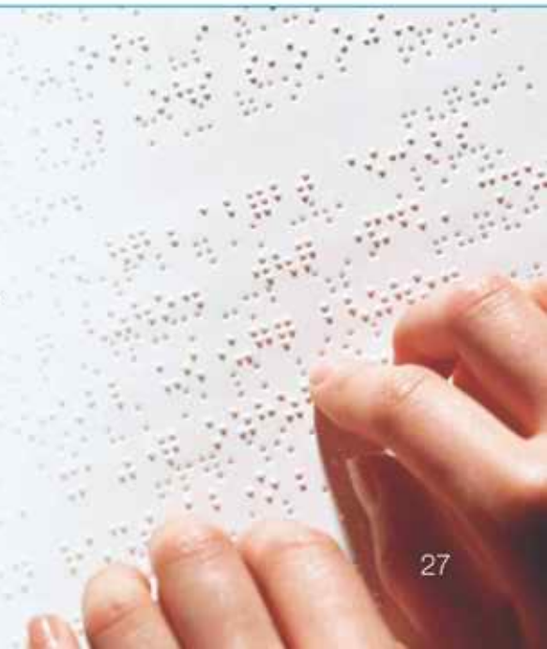
Warto wiedzieć

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.



Podsumowanie

- W pamięci komputera znaki, jak każdy inny rodzaj informacji, są zapisywane jako liczby. Liczby odpowiadające konkretnym znakom (literom, cyfrom, znakom interpunkcyjnym i innym symbolom) są określone przez ich położenie w tablicy Unicode.
- Pojedyncze znaki (litery alfabetu łacińskiego, cyfry, znaki interpunkcyjne i inne znaki) w języku C++ zapisuje się jako zmienne typu **char**.
- Zmienna typu **char** to liczba z zakresu od 0 do 127, która jest kodem liczbowym odpowiedniego znaku w tablicy ASCII.
- Praca z tekstem i z łańcuchami znaków wymaga w języku C++ użycia biblioteki `string`.
- Biblioteka `string` zawiera przydatne funkcje (metody) dotyczące napisów. Na przykład funkcja `length` zwraca długość napisu, funkcja **`find`** zwraca indeks pierwszego od lewej wystąpienia szukanego wzorca w danym łańcuchu, a funkcja `rfind` – indeks pierwszego wystąpienia wzorca od końca łańcucha.
- Dostęp do znaku w napisie uzyskuje się za pomocą nawiasów kwadratowych. Pierwszy znak łańcucha znaków o nazwie `tekst` to `tekst[0]`.
- Łańcuchy znaków (lub łańcuch i pojedynczy znak) można scalać, używając operatora `+`.

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 10 imion osób z twojej klasy i wypisze je w postaci tzw. trójkąta Floyda. Oznacza to, że w pierwszym wierszu będzie jedno imię, a w kolejnych odpowiednio dwa, trzy i cztery.

```
1
2 3
4 5 6
7 8 9 10
```

* **4** 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.

* **5** Napisz program, który wczyta wyraz wzorzec, a następnie będzie wczytywać kolejne wyrazy aż do pojawienia się ciągu znaków ***. Następnie program powinien wyświetlić liczbę powtórzeń wzorca pośród wczytanych wyrazów.

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

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

** **8** Taumatawhakatangihangakoauauotamateaturipukakapikimaungahoronukupokaiwhenuakitanatahu to nazwa jednego ze wzgórz w Nowej Zelandii i jedna z najdłuższych nazw geograficznych na świecie. Korzystając z pliku, który przekaże ci nauczyciel (np. *wzgorze.txt*), sprawdź:
a. długość tej nazwy,
b. liczbę występujących w niej liter: a, e, i, o, u, y,
c. czy w nazwie występuje fraza „eaturip".

** **9** 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.

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

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

** **12** 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.

** **13** 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`.

** **14** Pod kodem $7F_{16}$ (binarnie 0111 1111) w tablicy ASCII kryje się kod znaku sterującego DEL (od ang. *delete*, czyli usunąć). Poszukaj wiarygodnej informacji wyjaśniającej przyczynę umieszczenia kodu tego znaku w tym miejscu. Jaki miało to związek z zapisem na taśmie dziurkowanej używanej w drukarkach dalekopisów? Przygotuj krótką prezentację na temat ewolucji kodów stosowanych w telegrafii i teleksach. Uwzględnij kody Baudota–Murraya, ITA2 oraz ASCII.

** **15** 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.

*** **16** 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.

*** **17** Napisz program, który podane przez użytkownika zdanie (w języku angielskim) zapisze w kodzie Morse'a. Oprócz liter program powinien też kodować cyfry i kropkę.
Wskazówka: Aby w języku C++ wczytać do zmiennej o nazwie `tekst` ciąg znaków rozdzielony spacjami wpisywany z klawiatury, użyj instrukcji `getline(cin,tekst)`.

2. Szyfrujemy 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.

Cele lekcji

- Zrozumiesz podstawowe pojęcia związane z kryptografią.
- Dowiesz się, po co szyfruje się informacje i jakie są zastosowania kryptografii.
- Poznasz przykłady szyfrów podstawieniowych i przestawieniowych, w tym szyfr Cezara i szyfr kolumnowy.
- Napiszesz programy szyfrujące informacje tekstowe poznanymi metodami.
- Poznasz techniki łamania prostych szyfrów.

2.1. Podstawowe pojęcia kryptografii

Szyfrowanie

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**.

Kryptografia

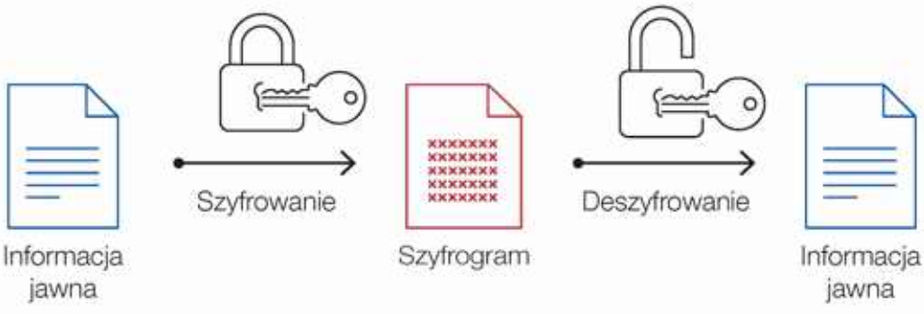
Tekst jawny

Szyfrogram

W trakcie szyfrowania informację jawną, zwaną **tekstem jawnym** (ang. *plaintext*), przekształca się w **szyfrogram** (ang. *ciphertext*), czyli tekst zaszyfrowany (rys. 2.1). W tym procesie stosuje się odpowiedni algorytm oraz **klucz szyfrowania** (ang. *cipher key*). Jest to pewna tajna wartość, niezależna od szyfrowanej informacji, która umożliwia **deszyfrowanie** zaszyfrowanej wiadomości. Warunkiem tajności informacji jest utrzymanie w tajemnicy klucza szyfrowania.

Klucz szyfrowania

Deszyfrowanie



Rys. 2.1. Schemat szyfrowania i deszyfrowania informacji

2.2. Szyfry przestawieniowe. Szyfr kolumnowy

Pionierami w rozwoju klasycznej kryptografii byli prawdopodobnie Spartanie. Grecki poeta Archiloch w jednym z dzieł wspomniał o przyrządzie do szyfrowania zwanym **skytale** (od gr. *skútalon* – długi, gruby kij), używanym w Sparcie od VII w. p.n.e.

Skytale

Zasada korzystania ze skytale była następująca: nadawca wiadomości nawijał na pałkę pasek pergaminu (lub skóry) i pisał tekst na stykających się fragmentach. Po rozwinięciu paska wiadomość stawała się nieczytelna. Adresat posiadał identyczną pałkę, umożliwiającą odczytanie wiadomości. W tej metodzie szyfrowania **kluczem** są grubość i kształt wałka.

Klucz szyfrowania, s. 30

Poniżej zilustrowano sposób odczytywania szyfrogramu na skytale o kształcie graniastosłupa prawidłowego sześciokątnego.

Odczytywanie wiadomości zaszyfrowanej na skytale

Po otrzymaniu szyfrogramu przekazanego na pasku pergaminu należy nawinąć go na wałek o ustalonej wcześniej grubości. Grubość wałka jest kluczem tego szyfru.

Szyfrogram:
TETSKAEKOTSWNSJTTNTTEEJY

Tekst jawny:
TENTEKSTTOJESTTEKSTJAWNY



Załóżmy, że niepowołana osoba przechwyciła zaszyfrowaną wiadomość oraz znała tę metodę szyfrowania. Jeśli nie zna dokładnych wymiarów wałka, trudno jej będzie odczytać informację, ponieważ musiałaby sprawdzić wiele wałków o różnych grubościach.

Ćwiczenie 1

Przygotuj model urządzenia skytale i pergaminu: użyj ołówka i wąskiego paska papieru o długości ok. 25–30 cm. Wykorzystaj zbudowany model skytale do zaszyfrowania krótkiej wiadomości tekstowej.

Dobra rada

Wiele ołówków ma kształt graniastosłupa prawidłowego sześciokątnego. Możesz więc wykorzystać ołówek jako skytale.

Warto wiedzieć

Auguste Kerckhoffs oraz Claude Shannon niezależnie od siebie sformułowali tezę zwaną dziś zasadą Kerckhoffsa. Mówi ona, że zapewnienie poufności informacji nie powinno zależeć od utrzymania metody szyfrowania w tajemnicy – tajny powinien być wyłącznie klucz. Dzięki upublicznieniu szczegółów algorytmów szyfrujących można wykryć luki bezpieczeństwa, zanim algorytm zostanie wdrożony.

Szyfr przestawieniowy

Technikę szyfrowania, z której korzysta się w metodzie z użyciem skytale, nazywa się współcześnie **szyfrem przestawieniowym**. Szyfr ten polega na przestawieniu znaków z informacji jawnej. Szyfrogram i informacja jawna składają się więc z tych samych znaków, ale ustawionych w innej kolejności.

Szyfr kolumnowy

Szyfr kolumnowy

Innym szyfrem przestawieniowym, podobnym do metody z użyciem skytale, jest **szyfr kolumnowy**. Polega on na zapisaniu tekstu jawnego w wierszach, przy czym każdy wiersz (czasem z wyjątkiem ostatniego) składa się z takiej samej liczby znaków odpowiadającej liczbie kolumn. Kluczem szyfrowania jest liczba kolumn. Jeśli ostatni wiersz jest niepełny, można go uzupełnić dowolnymi znakami.

Dla tekstu jawnego PRZYNIESPRZESYLKEDOPARKU i klucza 4 zapis w kolumnach wygląda tak jak na rysunku 2.2.

P	R	Z	Y
N	I	E	S
P	R	Z	E
S	Y	L	K
E	D	O	P
A	R	K	U

Rys. 2.2. Podział tekstu jawnego na kolumny zgodnie z kluczem szyfrowania

Jeśli odczytamy znaki tekstu kolumnami (z góry na dół), otrzymamy tekst zaszyfrowany: PNPSEARIRYDRZEZLOKYSEKPU.

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.

Ćwiczenie 2

Zaszyfruj tekst PRZYNIESPRZESYLKEDOPARKU, używając klucza o wartości 6.

A to ciekawe

Anagram jako szyfrogram

Niektórzy uczeni nie chcieli ogłaszać wprost treści swoich odkryć, dlatego stosowali anagramy, czyli zmieniali kolejność liter w zdaniu. 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.



Szyfr kolumnowy – program szyfrujący

Napiszemy teraz program, który pozwoli zaszyfrować wpisany z klawiatury tekst szyfrem kolumnowym z podanym kluczem.

Oto specyfikacja problemu:

Specyfikacja

Dane: tekst jawny o długości n wprowadzony z klawiatury bez spacji; klucz – liczba całkowita k, spełniająca warunek $2 \leqslant k \leqslant \frac{n}{2}$.

Wynik: tekst zaszyfrowany metodą kolumnową o liczbie kolumn równej k.

Aby zaprogramować szyfrowanie tekstu metodą kolumnową, nie trzeba pamiętać układu kolumn, wystarczy dopisywać do szyfrogramu znaki w odpowiedniej kolejności. Prześledźmy przykład szyfrowania tekstu PRZYNIESPRZESYLKEDOPARKU z kluczem 4.

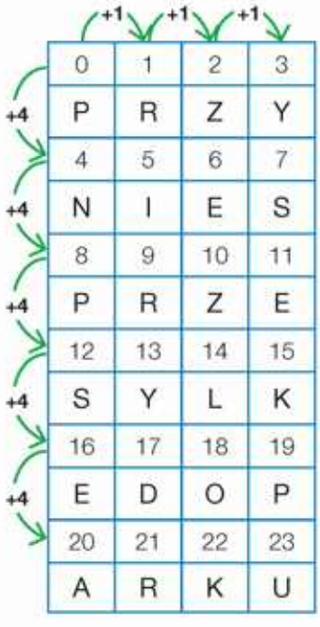
Ponumerujmy znaki tekstu jawnego, począwszy od 0, czyli tak jak numerowane są kolejne znaki w **łańcuchach znaków** (rys. 2.3):

Łańcuchy znaków, s. 14

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
P	R	Z	Y	N	I	E	S	P	R	Z	E	S	Y	L	K	E	D	O	P	A	R	K	U

Rys. 2.3. Tekst jawny z ponumerowanymi znakami

Ustawmy teraz tekst w czterech kolumnach, ponieważ klucz to 4 (rys. 2.4). Łatwo obliczyć, że wierszy musi być 6. Spójrzmy na numery znaków w pierwszej kolumnie: 0, 4, 8, 12, 16, 20 oraz numery znaków w drugiej kolumnie: 1, 5, 9, 13, 17, 21. Kolejne znaki w kolumnie mają numery różniące się o 4, czyli o wartość klucza (liczbę kolumn).



0	1	2	3
P	R	Z	Y
4	5	6	7
N	I	E	S
8	9	10	11
P	R	Z	E
12	13	14	15
S	Y	L	K
16	17	18	19
E	D	O	P
20	21	22	23
A	R	K	U

Rys. 2.4. Numeracja znaków po podziale na kolumny

Warto wiedzieć

Szyfrem podobnym do kolumnowego jest szyfr płotkowy, nazywany również metodą płotu. Kolejne litery tekstu jawnego umieszcza się w tabeli schodkowo, przyjmując pewną stałą wysokość schodków. Na przykład tekst PUŁAPKA zaszyfrowany metodą płotu z kluczem 4 ma postać: PAUKŁPA.

P						A
	U				K	
		Ł		P		
			A			

Lista kroków algorytmu szyfrowania kolumnowego może wyglądać następująco:

1. Wczytaj tekst jawny.
2. Wczytaj klucz szyfrowania i oznacz tę liczbę jako klucz.
3. Oznacz długość tekstu jawnego jako dl.
4. Dla każdej liczby i z zakresu od 0 do klucz - 1, traktowanej jako kolejna wartość licznika pętli, wykonuj krok 5.
   5. Dołącz do tekstu szyfrogramu następujące litery tekstu jawnego: tę na pozycji i oraz wszystkie te, których pozycja jest liczbą i powiększoną o wielokrotność klucza.

Zapiszmy kod źródłowy funkcji głównej programu *Szyfr kolumnowy*:

Fragment kodu źródłowego programu *Szyfr kolumnowy*

```
1.  int main()
2.  {
3.      string jawny, szyfrogram = "";
4.      int klucz, dl;
5.
6.      cout << "Podaj tekst jawny (bez spacji): "; cin >> jawny;
7.      cout << "Podaj klucz szyfrowania: "; cin >> klucz;
8.
9.      dl = jawny.length();
10.
11.     for (int i=0; i<klucz; i++)
12.         for (int j=i; j<dl; j=j+klucz)
13.             szyfrogram = szyfrogram + jawny[j];
14.
15.     cout << "\nSzyfrogram: " << szyfrogram << endl;
16.
17.     return 0;
18. }
```

Warto wiedzieć

W języku C++ zmienna typu `string`, której podczas deklaracji nie przypisano wartości początkowej, domyślnie jest traktowana jako łańcuch pusty. Deklaracje:
`string szyfrogram;`
oraz
`string szyfrogram="";`
są równoważne.

W wierszu 3 deklarujemy zmienną `szyfrogram` i przypisujemy jako początkową wartość **łańcuch pusty** (ang. *empty string*), czyli łańcuch bez znaków (o długości 0). Łańcuch ten będziemy wydłużać o kolejne znaki szyfrogramu. W linii 6, korzystając z **funkcji** (**metody**) `length`, obliczamy długość tekstu jawnego.

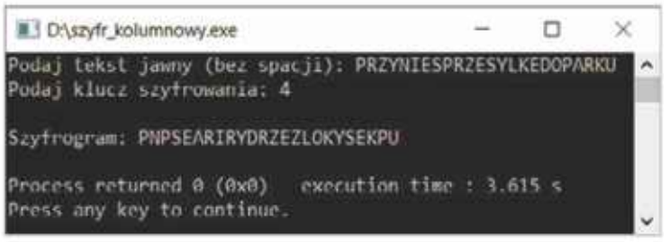
Łańcuch pusty

Funkcja `length`, s. 18

Pętle zagnieżdżone, s. 26

Szyfrogram jest tworzony w liniach 11–13 z wykorzystaniem **pętli zagnieżdżonych**. W zewnętrznej pętli (linie 11–13) przeglądamy kolejne kolumny. Pętla wykona się więc tyle razy, ile jest kolumn (jest to wartość klucza). W wewnętrznej pętli (linie 12–13) przeglądamy kolejne znaki w danej kolumnie (zgodnie z rys. 2.4 ze s. 33). Licznik pętli jest w każdym cyklu (przebiegu pętli) powiększany o wartość klucza, a liczba iteracji ograniczona jest z góry przez warunek `j<dl`. Gwarantuje to poruszanie się tylko w obrębie liter tekstu jawnego.

Rysunek 2.5 przedstawia przykładowe wywołanie programu dla tekstu jawnego PRZYNIESPRZESYLKEDOPARKU oraz klucza 4.



Rys. 2.5. Wywołanie programu *szyfr_kolumnowy.exe*

Ćwiczenie 3

a. Zapisz kod źródłowy programu *Szyfr kolumnowy*. Skompiluj go i sprawdź działanie programu dla tego samego tekstu, ale różnych kluczy – również takich, których wartości nie są dzielnikami długości (liczby znaków) tekstu jawnego.

b. Zmodyfikuj kod źródłowy programu *Szyfr kolumnowy* tak, aby wyświetlał on dodatkowo etap pośredni, tj. przedstawienie tekstu jawnego w układzie kolumnowym.

2.3. Szyfry podstawieniowe. Szyfr Cezara

Jeszcze w czasach starożytnych wymyślono inny rodzaj szyfru – **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 podstawieniowy

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.

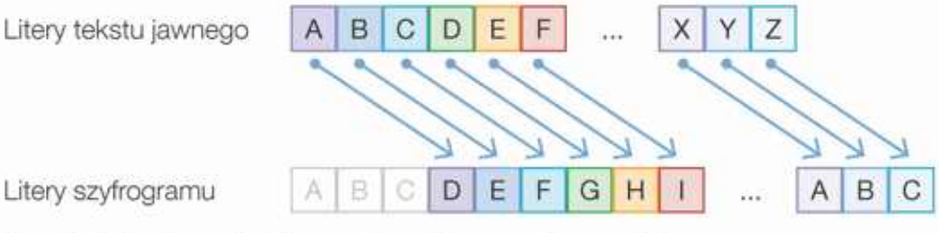
Szyfr Cezara

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.

Tekst jawny, s. 30

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 2.6.

Klucz szyfrowania, s. 30



Rys. 2.6. 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.

Ćwiczenie 4

Wykorzystując szyfr Cezara i alfabet łaciński:

a. zaszyfruj wyraz MEDYCYNA z kluczem 25.

b. odszyfruj szyfrogram LWUQVQSIVI z kluczem 8.

Warto wiedzieć

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

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 2.7 przedstawiono przesunięcie równe 2, tzn. literze A odpowiada litera C, literze B – litera D itd.

Warto wiedzieć

Wykorzystanie do szyfrowania obracających się tarcz z literami alfabetu zaproponował w XV w. Leon Battista Alberti, jeden z największych umysłów tamtych czasów. 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.



Rys. 2.7. Szyfrowanie liter szyfrem Cezara za pomocą obracających się tarcz

Ćwiczenie 5

a. Przygotuj fizyczny model obracających się tarcz pozwalających szyfrować tekst metodą Cezara, podobny jak na rysunku 2.7. Stwórz w programie graficznym projekt tarcz. Wydrukuj je i wytnij. Użyj pinezki, aby umocować przyrząd np. na tablicy korkowej.

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

Warto wiedzieć

Wiele współczesnych algorytmów kryptograficznych łączy elementy podstawiania i przestawiania. Mówi się wtedy o szyfrach kaskadowych (hybrydowych). Twórcą tej koncepcji jest Claude Shannon.

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 podanym kluczem.

Lista kroków algorytmu dla tej specyfikacji 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 nie większą 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 **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.

Typ char, s. 11

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*

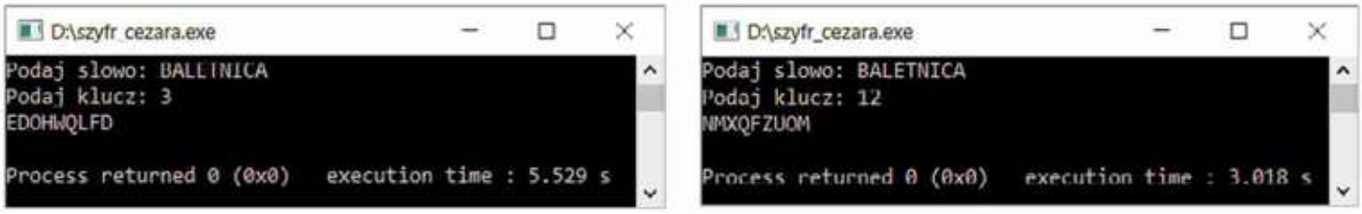
```
1.  int main()
2.  {
3.      string jawny, szyfrogram = "";
4.      int dl, klucz;
5.      char kod;
6.
7.      cout << "Podaj slowo: "; cin >> jawny;
8.      dl = jawny.length();
9.
10.     cout << "Podaj klucz: "; cin >> klucz;
11.
12.     for (int i=0; i<dl; i++)
13.     {
14.         kod = jawny[i] + 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.

Funkcja `length`, s. 18

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). Na rysunku 2.8 przedstawiono wywołania programu *Szyfr Cezara* dla kluczy 3 i 12.



Rys. 2.8. Wywołania programu *szyfr_cezara.exe* dla kluczy 3 oraz 12

Ćwiczenie 6

Zapisz kod źródłowy programu *Szyfr Cezara*. Skompiluj kod, uruchom program i sprawdź jego działanie.

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 dużych 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.

Warto wiedzieć

Współczesne komputerowe algorytmy szyfrowania 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.

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 nie większą 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.

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.

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. }
```

Dobra rada

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

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.

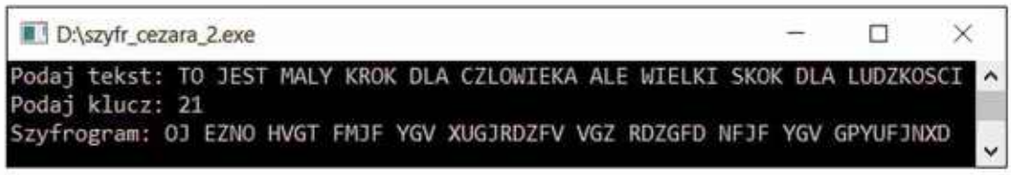
Warto wiedzieć

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`.

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 2.9 przedstawiono wywołanie programu *Szyfr Cezara 2*.



Rys. 2.9. Wywołanie programu *szyfr_cezara_2.exe*

Ćwiczenie 7

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

Deszyfrowanie wiadomości zaszyfrowanych metodą Cezara

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.

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

Ćwiczenie 8

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.

2.4. Metody łamania klasycznych szyfrów

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

Kryptologia ● **kryptologią** (od gr. *kryptós* – ukryty i *lógos* – słowo).

Kryptologia obejmuje wiedzę dotyczącą szyfrowania, deszyfrowania oraz łamania szyfrów w celu bezpiecznego przekazywania informacji.

Kryptoanaliza ● Nauka, która zajmuje się łamaniem szyfrów, to **kryptoanaliza** (od gr. *kryptós* – ukryty i *analýein* – rozluźnić, rozwiązać).

Zastosujemy kilka prostych technik kryptoanalizy do złamania poznanych szyfrów. 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 2.10.

Atak siłowy (brute-force)

Warto wiedzieć

Współcześnie stosowane szyfry używają jako kluczy liczb, które w zapisie binarnym mają po kilkaset, a nawet kilka tysięcy cyfr (bitów). Na przykład liczba potencjalnych kluczy szyfru dla klucza 256-bitowego to 2^{256}. To więcej niż 1000^{25}.



Rys. 2.10. Zastosowanie ataku siłowego do łamania szyfru Cezara

Warto wiedzieć

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

A to ciekawe

(Kryptologiczny) Cud nad Wisłą

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.

	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	

Szyfr kolumnowy, s. 32

W **szyfrze kolumnowym** liczba możliwych kluczy również jest stosunkowo niewielka, ponieważ ogranicza ją długość tekstu jawnego. Liczba potencjalnych kluczy jest w przybliżeniu równa połowie liczby znaków tekstu. W tym przypadku równie łatwo zastosować metodę siłową.

Warto wiedzieć

Za bezpieczny algorytm kryptograficzny uznaje się taki, którego złamanie zajęłoby ok. miliona lat. Ponieważ moc komputerów wzrasta, rośnie też liczba bitów w zapisie kluczy kryptograficznych.

Ćwiczenie 9

Napisz program łamiący szyfr Cezara metodą siłową.

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. Podobnie jest z szyfrem kolumnowym – atak ze znanym tekstem jawnym pozwala odtworzyć podział na kolumny.

Analiza częstości

Analiza częstości liter

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 najczęstszej w danym języku.

Dla krótkich tekstów taka analiza może jednak dać fałszywe wyniki. Metoda ta nie ma zastosowania w wypadku szyfru kolumnowego.

Warto wiedzieć

Krótkie teksty zaszyfrowane w prosty sposób mogą być nie do złamania nawet przez potężne komputery. Claude Shannon wykazał, że nie ma wówczas wystarczającej ilości informacji, by określić rodzaj szyfru. Niektóre teksty wyryte na starych nagrobkach pozostają nieodczytane przez setki lat mimo wysiłków wielu kryptologów.

Ćwiczenie 10

Znajdź informację o tym, jakie litery występują najczęściej w tekstach polskich oraz w innych językach, których się uczysz.

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 i szyfr kolumnowy nie stanowią dziś praktycznie żadnego zabezpieczenia.

Analiza częstości w języku polskim

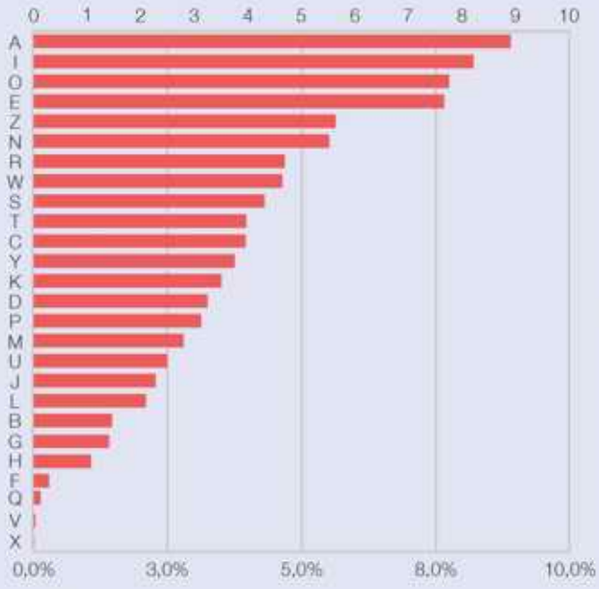
Pokażemy, jak wykorzystać analizę częstości do odszyfrowania szyfrogramu, który powstał po zaszyfrowaniu informacji w języku polskim. Posłużymy się tekstem zapisanym wyłącznie wielkimi literami alfabetu łacińskiego. W tekście nie używamy znaków interpunkcyjnych, a wyrazy są oddzielone spacjami.

Fragment szyfrogramu

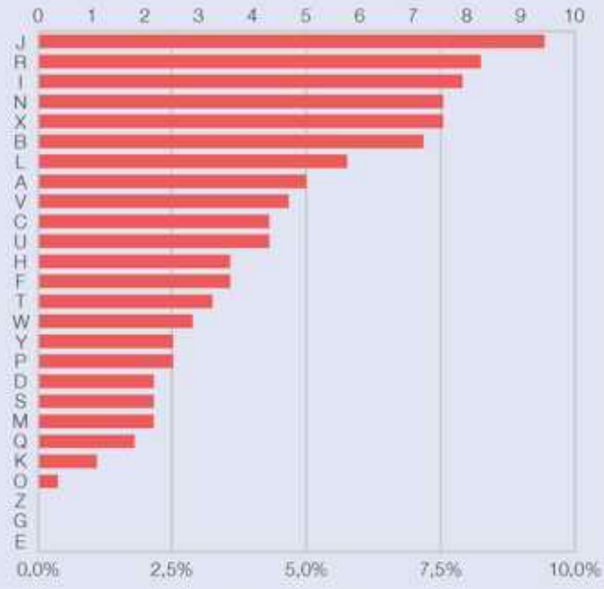
YXMLIJB UJVJWRJ BIHOAXPAJVD VXIN BRN YAIHMJL WRN CHUTX JWJURIJ
LINBCXBLR URCNA UNLI CJTIN IWJSXVXBL LINBCX FHBCNYDSJLHLQ XKXT
BRNKRN PADY MFXLQ UDK CAINLQ IWJTXF LIHUR MRPAJVXF R CARPAJVXF WJ
YAIHTUJM F SNIHTD YXUBTRV WJSLINBCBIHVR MRPAJVJVR BJ LQ LI BI R AI J
CARPAJVNV BUXFX BRN F SNIHTD JWPRNUBTRV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

Podsumowanie

- Podczas szyfrowania tekst jawny jest przekształcany w szyfrogram.
- Algorytmy służące do szyfrowania i do odszyfrowywania używają tajnego klucza, który jest wartością niezależną od szyfrowanej informacji.
- Szyfr przestawieniowy to szyfr, w którym zmienia się kolejność znaków tekstu jawnego. Wówczas w szyfrogramie występują wszystkie znaki tekstu jawnego. Przykładem takiego szyfru jest szyfr kolumnowy.
- Szyfr podstawieniowy to szyfr, w którym informacje utajnia się, zamieniając każdy znak informacji jawnej na inny znak.
- Proste szyfry podstawieniowe można złamać metodą analizy częstości (jeśli tylko tekst szyfrogramu jest odpowiednio długi) oraz metodą siłową.
- Klucz szyfrowania w szyfrze Cezara (wartość przesunięcia) łatwo określić, znajdując litery, które najczęściej występują w szyfrogramie.

Zadania

* **1** Jednym z najprostszych sposobów szyfrowania jest zapis liter tekstu wspak. Napisz program szyfrujący i deszyfrujący tą metodą tekst złożony z liter alfabetu łacińskiego bez spacji.

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

* **3** Szyfr, w którym litery tekstu jawnego pisze się na zmianę w dwóch wierszach, nazywa się czasem parkanem. Dla tekstu AKCJAOTRZYNASTEJ otrzymamy zapis kolumnowy:

```
ACATZNSE
KJORYATJ
```

Sformułuj specyfikację problemu i napisz program szyfrujący tym algorytmem.

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

* **5** 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.

** **6** Napisz program, który podany przez użytkownika tekst jawny, zapisany bez spacji, będzie szyfrować wybranym przez niego szyfrem: albo metodą kolumnową, albo szyfrem Cezara. Wybór szyfru powinien być dokonywany już po uruchomieniu programu.

** **7** 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.

** **8** Przygotuj prezentację na temat maszyny szyfrującej Enigma oraz historii złamania jej szyfru przez polskiego matematyka Mariana Rejewskiego. Przedstaw na osi czasu kluczowe wydarzenia związane z historią łamania szyfru Enigmy przed wybuchem II wojny światowej. Uwzględnij metodę zegara (wymyśloną przez Jerzego Różyckiego) i metodę płacht perforowanych (którą stworzył Henryk Zygalski).

** **9** Szyfr kolumnowy można stosować również w taki sposób, że kluczem jest nie liczba kolumn, ale ich kolejność (kolumny numerowane są od 0). Na przykład dla tekstu: PRZESYLKACZEKAWWILLI oraz klucza 40132 szyfrogram to: SCWIPYZWRLEIEAALZKKL.
Napisz program szyfrujący tekst tą metodą.

P	R	Z	E	S
Y	L	K	A	C
Z	E	K	A	W
W	I	L	L	I

** **10** 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

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

*** **12** Dany jest szyfr polegający na odwróceniu kolejności liter w kolejnych n-literowych grupach, gdzie n jest kluczem. Napisz program szyfrujący tekst tą metodą. Przyjmij, że tekst jawny na początku uzupełnia się dowolnymi literami, aby jego długość była wielokrotnością n.

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

3. Porządek ma znaczenie, czyli sortujemy liczby

Porządkowanie jest ważną częścią ludzkiej aktywności. Zwykle wysiłek w nie włożony opłaca się, bo później wiele zadań można wykonać szybciej i łatwiej. Są sytuacje, w których porządek ma szczególne znaczenie. Na przykład jeśli domy są ponumerowane w zrozumiały sposób, w razie pożaru strażacy sprawniej dotrą na miejsce. W świecie oprogramowania porządkowanie odbywa się dzięki algorytmom sortowania. W tym temacie przyjrzymy im się bliżej.

Cele lekcji

- Uświadomisz sobie, jak ważne jest porządkowanie danych.
- Lepiej zrozumiesz algorytmy wyszukiwania (liniowego i binarnego).
- Poznasz algorytmy sortowania: bąbelkowego i przez wstawianie.
- Dowiesz się, jak w języku C++ używać tablic w argumentach funkcji.

3.1. Do czego służy sortowanie?

Sortowanie (porządkowanie) danych

Sortowanie danych (ang. *sorting*), zwane też **porządkowaniem danych**, to ustawianie danych w zadanej kolejności według określonego kryterium.

Zwykle gdy mówimy o sortowaniu liczb, mamy na myśli uporządkowanie ich w kolejności malejącej albo rosnącej. W rzeczywistości najczęściej jest to kolejność nierosnąca lub niemalejąca, ponieważ sortowane wartości mogą się powtarzać. Wyrazy z reguły ustawiamy w kolejności alfabetycznej (od A do Z) albo odwróconej kolejności alfabetycznej (od Z do A). Natomiast daty i godziny porządkuje się zwykle chronologicznie, czyli od najstarszych do najnowszych (ewentualnie w odwróconej chronologii – od najnowszych do najstarszych).

Uporządkowane informacje, zwłaszcza obszerne zbiory danych, o wiele łatwiej czytać, analizować oraz przeszukiwać.

Rysunek 3.1 przedstawia przykładową tabliczkę przystankową. Zauważ, że dane są uporządkowane na dwóch poziomach: kierunki – alfabetycznie, a godziny dla każdego kierunku – chronologicznie.

LGOTA WIELKA

ROZKŁAD ODJAZDÓW

Kierunek	Przez	Godziny odjazdu		
BOGUMIŁOWICE	SULMIERZYCE	**14:50** S		
ŁÓDŹ	BEŁCHATÓW	**6:58** dnw	**12:08** dnw	**17:03** dnw
PAJĘCZNO	SULMIERZYCE	**09:20** Dm		
RADOMSKO		**05:11** 6s	**07:01** S	**07:09** S
		07:11 S	**08:15** S	**08:15** Dm
		12:05 Dm	**14:35** dnw	**18:05** dnw
		23:18 dnw		

Rys. 3.1. Przykład tabliczki przystankowej

Ćwiczenie 1

Z jakimi uporządkowanymi zbiorami danych masz do czynienia na co dzień? Wymień co najmniej pięć konkretnych przykładów.

Szukanie hasła w słowniku lub skorowidzu

W encyklopediach, słownikach i leksykonach obowiązuje porządek alfabetyczny. Gdy korzystasz z tradycyjnego słownika, np. polsko-angielskiego, zwykle nie szukasz na chybił trafił, lecz intuicyjnie stosujesz bardzo rozsądną strategię poszukiwania słowa. Podobnie postępujesz podczas szukania hasła w indeksie (skorowidzu), umieszczonym zwykle na końcu książki (rys. 3.2).

Warto wiedzieć

Porządek alfabetyczny nazywa się też leksykograficznym. W tym uporządkowaniu spacje (i inne tego typu znaki, np. dwukropek) są umieszczane przed literami alfabetu. Wynika to z kolejności znaków w kodzie ASCII. To dlatego hasło „kod źródłowy" w słowniku informatycznym będzie wcześniej niż np. hasło „kodowanie".

A
adres komórki 11
adresowanie bezwzględne 14
– mieszane 14
– względne 14
algorytm 94, 172
– Euklidesa 158, 235
– – – wersja z dzieleniem 160, 237
– Herona 98, 176
– wybierania największej z trzech liczb 108, 185
arkusz 10
atrybut 69
– JEŻELI 17
– LICZ.JEŻELI 28
– LOS.ZAKR 33
– MAX 16
– MAX.K 63
– MIN 16
– SUMA 11
– ŚREDNIA 16
funkcja (C++) 145
– main 101
– niezwracająca wartości 163
– sqrt 150
– swap 155

Rys. 3.2. Fragment indeksu podręcznika *Informatyka na czasie 2*

Jeśli chcesz znaleźć słowo „algorytm", to prawdopodobnie szukasz go wśród początkowych haseł, a jeśli „mediana" – mniej więcej w środku. Następnie szukasz wyrazów zawierających dwie pierwsze litery – w naszej sytuacji będą to odpowiednio „al" i „me". Za każdym razem zawężasz obszar poszukiwań, pomijając strony, na których dane hasło na pewno nie wystąpi. Posortowanie haseł przez twórców słownika pozwala więc znacznie skrócić czas przeszukiwania.

Warto wiedzieć

Metodę wyszukiwania, którą stosujemy intuicyjnie podczas korzystania ze słownika, nazywa się algorytmem wyszukiwania interpolacyjnego.

A to ciekawe

Algorytmy wyszukiwania

Znalezienie w sieci informacji, których potrzebujemy, byłoby praktycznie niemożliwe bez ich uporządkowania. Systemy rankingowe wyszukiwarek internetowych przeglądają indeksy wyszukiwania, stosując wiele różnych algorytmów, analizujących takie aspekty jak: treść zapytania, przydatność stron, wiarygodność źródła, czas zamieszczenia informacji, lokalizacja użytkownika. Gdyby wykorzystywane algorytmy nie działały bardzo szybko dla dużych zestawów danych, na wyniki dla pojedynczego zapytania trzeba byłoby czekać nawet kilkanaście minut lub dłużej.



Porządek wokół nas

Zwykle nie zwracamy uwagi na to, jak wiele obiektów wokół nas jest uporządkowanych. Ważne jest jednak nie tylko samo uporządkowanie, lecz także jego funkcjonalność, która znacznie ułatwia nam codzienne życie.

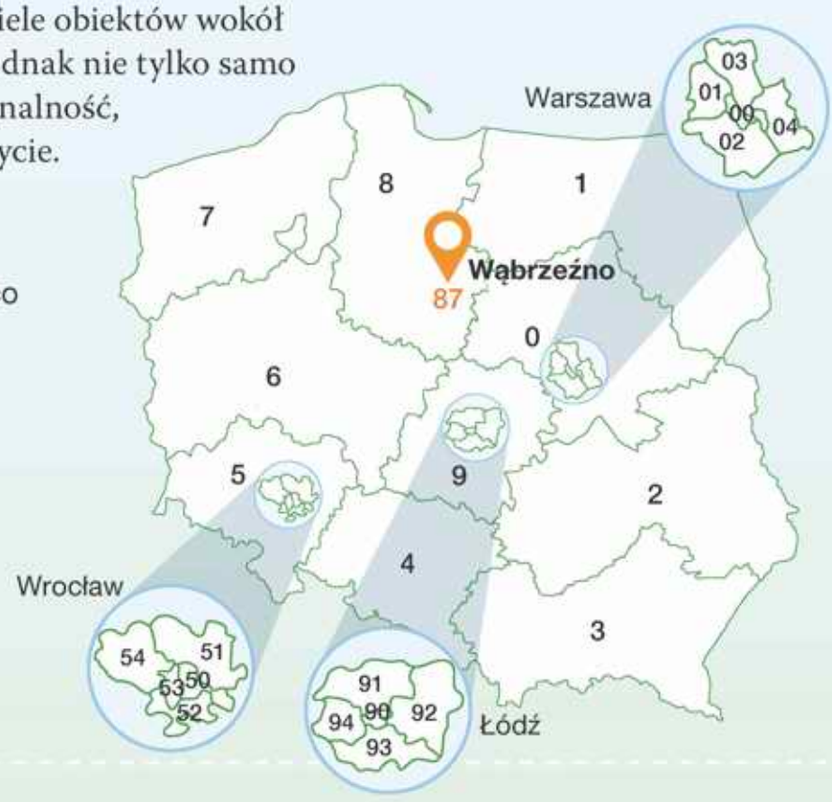
Kody pocztowe

Wprowadzenie kodów pocztowych znacząco usprawniło sortowanie i skróciło czas dostarczania listów. Wcześniej pracownicy poczty musieli sami odszyfrowywać nazwy miejscowości i sprawdzać ich lokalizacje.

Wąbrzeźno

8 7 – 2 0 0

Okręg i strefa kodowa | Sektor kodowy i placówka pocztowa



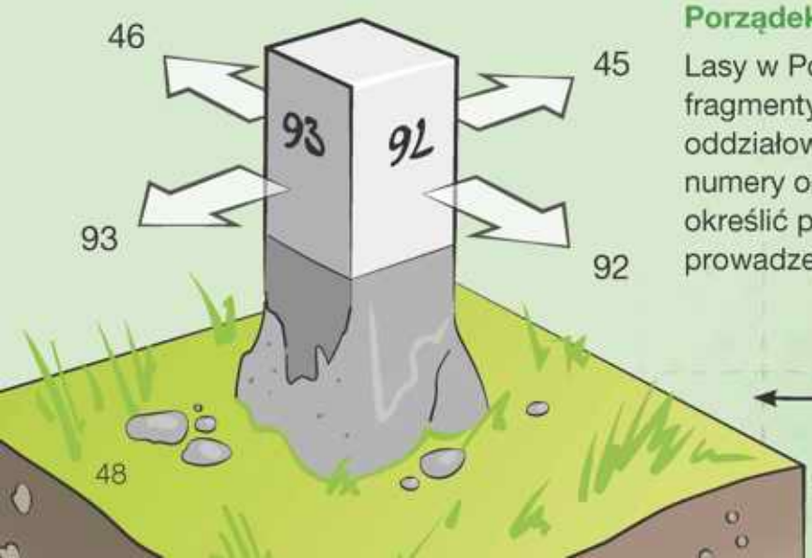
Numeracja domów

Domy znajdujące się w centrach miast są numerowane w uporządkowany sposób. Numery domów położonych wzdłuż głównych ulic rosną w kierunku od centrum miejscowości ku jej krańcom: po lewej stronie są numery nieparzyste, po prawej – parzyste.



Porządek w lesie

Lasy w Polsce są podzielone na oddziały (prostokątne fragmenty), które są identyfikowane przez tzw. słupki oddziałowe. Na słupkach znajdują się liczby oznaczające numery oddziałów leśnych. Słupki oddziałowe pomagają określić położenie, wyznaczyć kierunek oraz ułatwiają prowadzenie akcji ratowniczej.



Klasyfikacja zawodników w pięcioboju nowoczesnym

Przyjrzyjmy się bliżej zasadom ustalania kolejności w pięcioboju nowoczesnym. Składa się on z: szermierki, pływania, jazdy konnej oraz konkurencji łączonej – biegu na przełaj i strzelania z pistoletu. Wyniki w poszczególnych konkurencjach są przeliczane na punkty wielobojowe. W tabeli 3.1 przedstawiono wyniki pierwszych dziesięciu zawodniczek w klasyfikacji końcowej Letnich Igrzysk Olimpijskich w Rio de Janeiro w 2016 r. W rywalizacji brało udział 36 zawodniczek.

Warto wiedzieć

Na igrzyskach w Rio de Janeiro w 2016 r. polska pięcioboistka Oktawia Nowacka z wynikiem 1349 punktów wywalczyła brązowy medal.

			Szermierka			Pływanie			Jazda konna		Bieg ze strzelaniem			Suma
			Wygrane	Miejsce	Pkt	Czas	Miejsce	Pkt	Miejsce	Pkt	Czas	Miejsce	Pkt	
1.	Chloe Esposito	AUS	19	13.	215	00:02:12	7.	303	19.	284	00:12:10	2.	570	1372
2.	Élodie Clouvel	FRA	21	7.	227	00:02:09	2.	315	11.	293	00:12:59	17.	521	1356
3.	Oktawia Nowacka	POL	27	1.	264	00:02:17	16.	290	9.	293	00:13:18	25.	502	1349
4.	Annika Schleu	GER	17	16.	202	00:02:19	21.	282	12.	293	00:12:22	3.	559	1336
5.	Kate French	GBR	17	19.	202	00:02:06	15.	292	1.	300	00:12:43	8.	537	1331
6.	Natalya Coyle	IRL	19	12.	215	00:02:17	18.	288	4.	300	00:12:58	16.	522	1325
7.	Alice Sotero	ITA	20	9.	221	00:02:13	8.	303	23.	279	00:13:00	18.	520	1323
8.	Samantha Murray	GBR	14	31.	192	00:02:11	4.	308	22.	279	00:12:59	7.	542	1321
9.	Jelena Potapienko	KAZ	17	20.	202	00:02:12	5.	306	7.	293	00:13:07	22.	513	1314
10.	Tamara Vega	MEX	15	26.	190	00:02:17	17.	290	5.	300	00:12:49	11.	531	1311

Tabela 3.1. Wyniki pięcioboju nowoczesnego kobiet – igrzyska w Rio de Janeiro w 2016 r.

Ćwiczenie 2

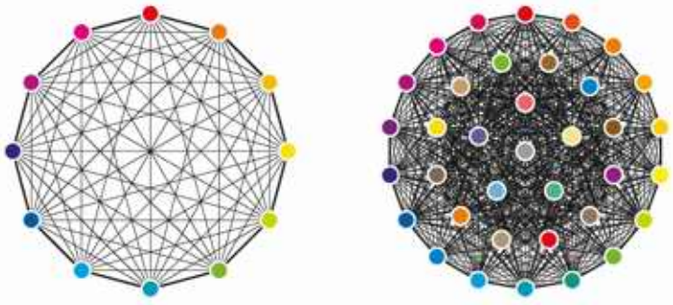
a. Według jakiego kryterium uporządkowano zawodniczki w tabeli 3.1?
b. Czy klasyfikacje dla każdej pojedynczej konkurencji są tworzone na podstawie wyników uporządkowanych malejąco?

Ustalenie kolejności zawodników podczas zawodów w wieloboju jest szczególnie ważne przed rozegraniem ostatniej konkurencji (strzelanie i bieg). Różnice punktowe w klasyfikacji przeliczane są bowiem na sekundy. Lider startuje jako pierwszy, a kolejni z opóźnieniem wyliczonym na podstawie dotychczasowych wyników.

Warto wiedzieć

W języku angielskim system rywalizacji „każdy z każdym" nazywa się *round-robin tournament* (dosł. turniej karuzelowy). Z informatycznego punktu widzenia jest to algorytm sortowania porównawczo-zliczeniowego. Każdy element zbioru danych liczbowych jest porównywany ze wszystkimi pozostałymi, co daje informację, od ilu elementów jest on większy.

Systemy wyłaniania zwycięzców wcześniejszych konkurencji muszą być sprawiedliwe. Na przykład w szermierce stosuje się system „każdy z każdym", zwany systemem kołowym. Przy dużej liczbie rywalizujących jest on jednak bardzo czasochłonny. Dla 36 zawodniczek (każda walczy 35 razy) należy przeprowadzić aż $\frac{1}{2} \cdot 36 \cdot 35 = 630$ pojedynków. Dla porównania – przy 12 zawodniczkach odbyłoby się 66 walk. Różnicę tę pokazuje rysunek 3.3.



Rys. 3.3. Ilustracja liczby pojedynków w rywalizacji „każdy z każdym" dla 12 i 36 zawodniczek

Sortowanie danych w komputerze

Sortowanie zbiorów danych jest jedną z operacji najczęściej wykonywanych przez komputery podczas przetwarzania informacji. Oto kilka przykładów zastosowań sortowania, od prostszych do mniej oczywistych:

- wyświetlanie listy plików w katalogu (folderze) na dysku – przed wyświetleniem zawartości katalogu system operacyjny sortuje pliki zgodnie z określonym kryterium,
- przygotowanie do wydruku danych z arkusza kalkulacyjnego lub bazy danych – często stosujemy sortowanie, aby ułatwić czytanie danych zawartych w tabelach (nadać im odpowiednie znaczenie),
- wyszukiwarki internetowe – zasoby zgromadzone na serwerach wyszukiwarek internetowych muszą być zindeksowane, aby wyszukiwarka mogła zwrócić wynik możliwie szybko,
- grafika komputerowa – obiekty w grze komputerowej lub np. w symulatorze lotu są ułożone na warstwach, więc aby program mógł wyświetlić obiekty na ekranie w dobrej kolejności, może wymagać ich wcześniejszego posortowania (względem odległości od obserwatora).

Warto wiedzieć

Przykłady problemów, do których rozwiązania potrzebne jest sortowanie danych, to:

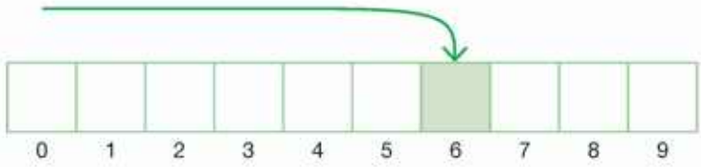
- usuwanie duplikatów,
- sprawdzanie, czy wszystkie elementy zbioru danych są różne,
- znajdowanie dwóch wartości o najmniejszej różnicy,
- wyznaczanie mediany, dominanty i innych statystyk,
- odtwarzanie pierwotnego porządku elementów zbioru danych.

3.2. Sortowanie a wyszukiwanie

Uporządkowanie danych ma zasadnicze znaczenie podczas przeszukiwania ich zbioru. Prześledźmy to na przykładzie wyszukiwania w pamięci komputera.

Przyjmijmy, że po uruchomieniu jakiegoś programu w pamięci komputera mamy zapisaną tablicę jako uporządkowany ciąg komórek pamięci RAM, zaadresowanych kolejnymi liczbami całkowitymi.

Jeśli procesor zna położenie danych w pamięci, to odwołuje się do nich bezpośrednio, poprzez adres komórki, w której się znajdują (rys. 3.4).



Rys. 3.4. Dostęp bezpośredni

Aby odczytać dane zapisane w siódmej komórce fragmentu pamięci, procesor nie odczytuje wszystkich danych w kolejności, w jakiej zostały zapisane, lecz pobiera dane z właściwej komórki (rys. 3.4).

Współczesne algorytmy komputerowe często wymagają, aby czas dostępu do danych z pamięci operacyjnej był jak najkrótszy. Wtedy kluczowe znaczenie – oprócz aspektu technicznego (budowy pamięci) – ma właśnie uporządkowanie danych. Jego brak znacznie spowalnia wyszukiwanie informacji, nawet jeśli korzysta się z szybkich procesorów.

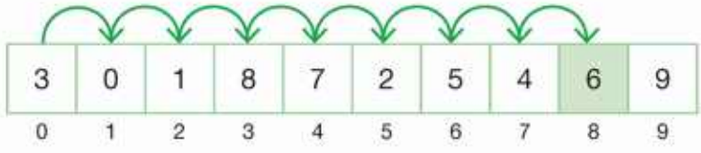
Warto wiedzieć

Czas dostępu do danych zależy nie tylko od ich uporządkowania, lecz także od rodzaju pamięci. Różny czas osiągają rejestry procesora, pamięć podręczna, pamięć RAM czy pamięć zewnętrzna (np. flash, dysk twardy).

Wyszukiwanie sekwencyjne

Kiedy szukamy informacji w zbiorach danych nieuporządkowanych, nie mamy wielkiego wyboru: albo zastosujemy metodę „na chybił trafił", albo algorytm **wyszukiwania sekwencyjnego** (ang. *sequential search*), tj. przeglądanie zbioru danych w kolejności, w jakiej są zapisane (np. według adresów w pamięci). Rysunek 3.5 pokazuje sekwencyjne wyszukiwanie liczby 6 w zbiorze nieuporządkowanym.

Wyszukiwanie sekwencyjne



Rys. 3.5. Wyszukiwanie sekwencyjne

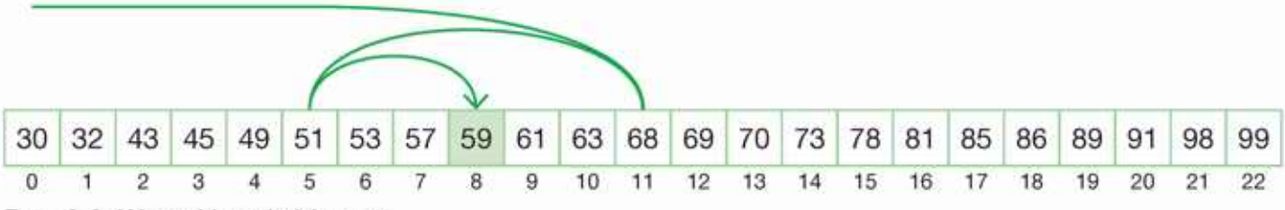
Warto wiedzieć

Metodę „na chybił trafił" nazywa się czasami wyszukiwaniem głupim (ang. *bogo sort*).

Wyszukiwanie binarne

Wyszukiwanie informacji można znacznie przyspieszyć, gdy zbiór danych jest uporządkowany. Stosujemy wówczas strategię połowienia zbioru, czyli zmniejszania liczby jego elementów o połowę w każdej kolejnej próbie. Nazywamy ją **wyszukiwaniem binarnym (połówkowym)**. Algorytm ten, zastosowany do poszukiwania liczby 59 w zbiorze składającym się z 23 elementów, przedstawiono na rysunku 3.6.

Wyszukiwanie binarne (wyszukiwanie połówkowe)



Rys. 3.6. Wyszukiwanie binarne

Warto wiedzieć

Porównanie dwóch liczb jest jedną z podstawowych operacji wykonywanych przez procesor komputerowy. Każdy język programowania musi obsługiwać tę operację.

Wyszukiwanie liczby 59 zaczęło się od sprawdzenia środkowej komórki obszaru pamięci. Jest to komórka o indeksie 11, przechowująca liczbę 68, która jest większa od 59. Ponieważ wiemy, że dane w pamięci są uporządkowane, mamy pewność, że przeszukiwanie komórek o indeksach większych od 11 można pominąć. W drugiej próbie program wyszukujący powinien sprawdzić komórkę o indeksie 5. Znajduje się w niej liczba 51, dlatego wartości należy szukać w komórkach o indeksach od 6 do 10. W kolejnym kroku sprawdzamy komórkę o adresie 8 – znajdziemy w niej poszukiwaną liczbę.

Ćwiczenie 3

Ile sprawdzeń (porównań dwóch liczb) wystarczy, aby rozstrzygnąć, czy w ciągu komórek z rysunku 3.6 znajduje się liczba 80? Zapisz indeksy oraz wartości komórek, które kolejno będą sprawdzane.

3.3. Algorytmy sortowania

Warto wiedzieć

Algorytmy sortowania można podzielić na dwie grupy: te, które działają na zasadzie porównywania elementów zbioru parami (np. sortowanie przez wybór), oraz nieposługujące się tą operacją (np. sortowanie przez zliczanie).

Spójrzmy na porządkowanie jako na problem algorytmiczny i programistyczny. W ostatnich kilkudziesięciu latach odkryto i opisano co najmniej kilkanaście użytecznych algorytmów sortowania danych. Okazuje się, że nie ma algorytmu idealnego o ogólnym przeznaczeniu, najszybszego w każdej sytuacji. Najbardziej uniwersalne są metody hybrydowe, łączące zalety dwóch lub trzech algorytmów sortowania, stosowanych na różnych etapach procesu porządkowania.

Ćwiczenie 4

Przygotuj 8 kartek wielkości połowy kartki z zeszytu. Na każdej z nich zapisz imię oraz datę imienin (dzień i miesiąc). Imiona nie mogą się powtarzać. Następnie odwróć kartki i ułóż je w losowej kolejności.

a. Zaproponuj metodę sortowania informacji na kartkach (alfabetycznie lub chronologicznie). Przyjmij, że kartki trzeba porównywać parami i nie wolno odkryć więcej niż dwóch naraz.

b. W grupie 2- lub 3-osobowej przedyskutujcie zastosowane wcześniej metody porządkowania kartek. Porównajcie czas sortowania 8 i 16 kartek wybraną przez was metodą. Ile razy dłużej trwa porządkowanie 16 kartek?

Programy komputerowe mogą porządkować różne typy danych: liczby, zbiory liter i słów, daty i inne. Wszystkie te wartości są zapisywane cyfrowo w pamięci komputera. Dla uproszczenia ograniczymy się więc do porządkowania tablic liczb całkowitych. Dodatkowo przyjmijmy, że będziemy sortować w porządku niemalejącym.

Oto specyfikacja naszego problemu sortowania:

Specyfikacja

Dane: liczba naturalna `N` i `N`-elementowa tablica `A` liczb całkowitych.

Wynik: tablica `A` po uporządkowaniu: $A[0] \leqslant A[1] \leqslant \ldots \leqslant A[N-1]$.

3.4. Sortowanie bąbelkowe

Jednym z algorytmów porządkowania jest **sortowanie bąbelkowe** (ang. *bubble sort*). W tym algorytmie na każdym etapie porównuje się parę elementów sąsiednich, zaczynając od pierwszej pary w tablicy. Jeżeli podczas porównywania elementy są w niewłaściwej kolejności, to zamienia się je miejscami.

Sortowanie bąbelkowe

Operacja prostej zamiany

Sortowanie bąbelkowe nazywa się też czasami sortowaniem przez **prostą zamianę**, rozumianą jako zamianę miejscami sąsiednich elementów w tablicy.

Operacja prostej zamiany

Podobną operację stosuje się w oprogramowaniu niektórych telewizorów. Pozwala ono za pomocą pilota uszeregować listę kanałów. Są trzy możliwości: zamiana kanałów miejscami (•), przejście w dół (▽) i przejście w górę listy (△). Rysunek 3.7 przedstawia zastosowanie operacji prostej zamiany podczas ustawiania listy kanałów.

TV 3	•	TV 0	▽	TV 0		TV 0		TV 0		TV 0
TV 0		TV 3	•	TV 1	▽	TV 1		TV 1		TV 1
TV 1		TV 1		TV 3	▽	TV 3		TV 3	•	TV 2
TV 6		TV 6		TV 6	▽	TV 6	•	TV 2	△	TV 3
TV 7		TV 7		TV 7	•	TV 2	△	TV 6		TV 6
TV 2		TV 2		TV 2		TV 7		TV 7		TV 7
TV 5		TV 5		TV 5		TV 5		TV 5		TV 5
TV 4		TV 4		TV 4		TV 4		TV 4		TV 4

Rys. 3.7. Pierwsze kroki porządkowania kanałów telewizyjnych przez prostą zamianę

A to ciekawe

Sortowanie w tańcu

Na kanale AlgoRythmics w serwisie YouTube można znaleźć filmy objaśniające działanie różnych algorytmów sortowania. Porządkowanie liczb jest zaprezentowane w tańcu przez członków węgierskiego zespołu folkowego Maros Művészegyüttes. Można tam zobaczyć również m.in. wyszukiwanie binarne.

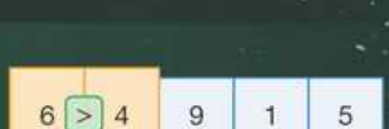


Algorytm sortowania bąbelkowego

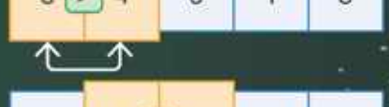
Porządkowanie elementów tą metodą pokażemy na przykładzie pięcioelementowej tablicy, w której na kolejnych pozycjach znajdują się liczby: 6, 4, 9, 1, 5. Elementy uporządkujemy niemalejąco. Pierwszy etap ilustruje, jak umieścić właściwy element na ostatniej pozycji. W kolejnych etapach postępujemy analogicznie.

Etap 1

- Porównujemy pierwszy element z drugim. 6 > 4, więc zamieniamy liczby miejscami.



- Porównujemy drugi element z trzecim. 6 < 9, więc nie zamieniamy liczb miejscami.



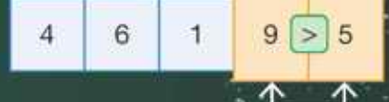
- Porównujemy trzeci element z czwartym. 9 > 1, więc zamieniamy liczby miejscami.



- Porównujemy czwarty element z piątym. 9 > 5, więc zamieniamy liczby miejscami.

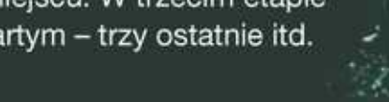


- W wyniku czterech porównań największy element tablicy (liczba 9) znalazł się na ostatniej pozycji.

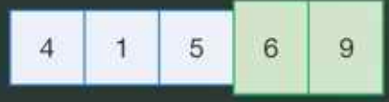


W drugim etapie umieścimy właściwy element na przedostatniej pozycji. Przy porównaniach pomijamy ostatni element tablicy, ponieważ jest już na właściwym miejscu. W trzecim etapie przy porównaniach pomijamy dwa ostatnie elementy tablicy, w czwartym – trzy ostatnie itd. Poniżej przedstawione są wyniki otrzymane w kolejnych etapach.

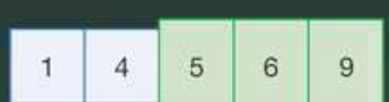
Etap 2 Porównujemy pary liczb: 4 i 6, 6 i 1 oraz 6 i 5. Liczba 6 znalazła się na właściwej, przedostatniej pozycji.



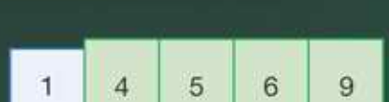
Etap 3 Porównujemy pary liczb 4 i 1 oraz 4 i 5. W wyniku tych operacji liczba 5 jest na właściwej pozycji.



Etap 4 Porównujemy parę liczb 1 i 4. Znalezienie drugiego elementu ciągu wskazuje jednocześnie najmniejszy element ciągu.



Wynikiem działania algorytmu jest tablica:



Ćwiczenie 5

Dana jest tablica A = [3, 0, 1, 7, 5]. Sprawdź, ile prostych zamian potrzeba, by uporządkować ją algorytmem sortowania bąbelkowego:
a. rosnąco,
b. malejąco.

Algorytm porządkowania metodą bąbelkową można zapisać jako **pętle zagnieżdżone**. To oznacza, że na jeden etap algorytmu (cykl pętli zewnętrznej) może przypadać wiele cykli pętli wewnętrznej.

Pętle zagnieżdżone, s. 26

Każdy etap algorytmu kończy się ustawieniem na docelowym miejscu elementu, który w analizowanym fragmencie był tym o największej wartości. Liczba porównań w kolejnym etapie zmniejsza się więc o 1.

Oto zapis algorytmu w postaci listy kroków:

1. Wczytaj tablicę liczb A.
2. Dla wartości i równych kolejno 1, 2, …, N-1 powtarzaj kroki 3 oraz 4.
   3. Dla wartości j równych kolejno 0, 1, …, N-i-1 powtarzaj krok 4.
      4. Jeśli A[j]>A[j+1], to zamień te elementy miejscami.

Liczba powtórzeń zewnętrznej pętli wynosi N–1. Liczba powtórzeń wewnętrznej pętli (tzw. pętli sortującej) jest zależna od wartości zmiennej i, sterującej zewnętrzną pętlą.

Dobra rada

Aby uporządkować elementy nierosnąco, zmień w instrukcji warunkowej znak > na <.

Kod źródłowy funkcji o nazwie Babelkowe w programie *Sortowanie bąbelkowe* może wyglądać następująco:

Kod źródłowy funkcji Babelkowe w programie *Sortowanie bąbelkowe*

```
1.  void Babelkowe (int A[])
2.  {
3.      for (int i=1; i<N; i++)
4.          for (int j=0; j<N-i; j++)
5.              if (A[j]>A[j+1])
6.                  swap(A[j],A[j+1]);
7.  }
```

Zauważ, że funkcja Babelkowe jako parametr przyjmuje tablicę A. W ten sposób przekazuje się do funkcji informację o adresie tablicy, czyli położeniu pierwszej komórki tablicy w pamięci. Wszystkie operacje są wykonywane na tej tablicy, a zmiany są w niej zapamiętane nawet po zakończeniu działania funkcji.

W liniach 3 oraz 4 znajduje się odwołanie do wartości N, określającej rozmiar tablicy do posortowania. Wartość ta jest stałą globalną, zdefiniowaną przed definicją funkcji, która z niej korzysta. Dla tablicy o 10 elementach odpowiedni wiersz kodu będzie miał postać: `const int N = 10;`.

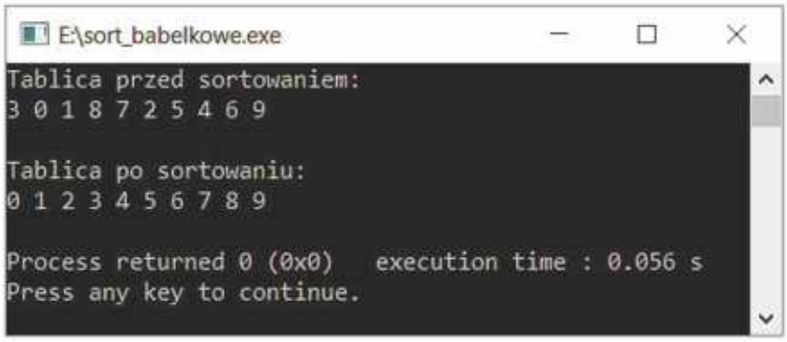
Warto wiedzieć

W nagłówku definiowanej funkcji, np. `void Funkcja (int A[])`, nazwa tablicy nie musi być identyczna z nazwą tablicy, dla której wywołamy funkcję w programie głównym. Funkcję można wywołać dla np. tablicy B i zapisać Funkcja(B).

Dobra rada

Zamiast wprowadzać dane do tablicy bezpośrednio w kodzie źródłowym, możesz napisać funkcję, która pozwoli wczytać liczby podane przez użytkownika.

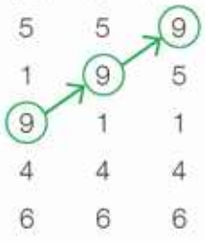
Rysunek 3.8 przedstawia przykładowe wywołanie programu *Sortowanie bąbelkowe*.



Rys. 3.8. Przykładowe wywołanie programu *sort_babelkowe.exe*

Warto wiedzieć

Nazwa „sortowanie bąbelkowe" powstała przez analogię do zjawiska wypływania na powierzchnię wody bąbelków powietrza: ten o największej objętości wypływa jako pierwszy.



Ćwiczenie 6

Napisz kod źródłowy programu o nazwie *Sortowanie bąbelkowe*, w którym wykorzystasz funkcję Babelkowe, a dane do sortowania wprowadzisz bezpośrednio w kodzie źródłowym. Skompiluj kod i sprawdź działanie programu.

Zapamiętaj

Kluczową operacją w wielu algorytmach sortowania danych jest porównywanie danych parami, by ustalić ich porządek.

3.5. Sortowanie przez wstawianie

Sortowanie przez wstawianie

Omówimy teraz kolejny algorytm porządkowania danych – **sortowanie przez wstawianie** (ang. *insertion sort*). Metoda ta polega na przeglądaniu kolejnych elementów tablicy od lewej i przenoszeniu ich, w razie potrzeby, w odpowiednie miejsce już uporządkowanego fragmentu tablicy po lewej stronie. Działanie algorytmu przedstawiono na s. 57.

Sortowanie przez wstawianie przypomina układanie kart przez grających w gry karciane: karta, którą podnosimy ze stołu, jest wstawiana we właściwe miejsce do kart już uporządkowanych (rys. 3.9).

Warto wiedzieć

Doświadczeni gracze w pokera lub brydża nie sortują swoich kart. W ten sposób bowiem dostarczyliby przeciwnikom informacji o tym, jakie karty mają w ręku.



Rys. 3.9. Układanie kart metodą przez wstawianie

Algorytm sortowania przez wstawianie

Porządkowanie elementów tą metodą pokażemy na przykładzie pięcioelementowej tablicy, w której na kolejnych pozycjach znajdują się liczby: 6, 4, 9, 1, 5. Elementy uporządkujemy niemalejąco. Pierwszy etap ilustruje, jak uporządkować niemalejąco dwa elementy tablicy. Liczby przy strzałkach oznaczają kolejność wykonywania operacji.

Etap 1

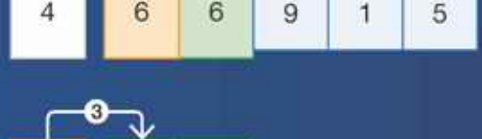
- Przyjmujemy, że pierwszy element tablicy (liczba 6) jest na właściwym miejscu. Drugi element, liczbę 4, zapamiętujemy w zmiennej pomocniczej.



- Liczba 4 powinna się znaleźć przed liczbą 6, dlatego aby zrobić dla niej miejsce, przesuwamy liczbę 6 o jedną pozycję w prawo.



- Liczbę 4 ze zmiennej pomocniczej zapisujemy jako pierwszy element tablicy.



- Wynikiem jest uporządkowanie niemalejąco dwóch pierwszych elementów wyjściowej tablicy.

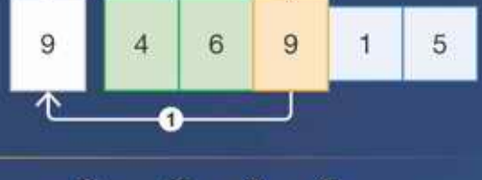


Postępując analogicznie, w drugim etapie uporządkujemy niemalejąco trzy elementy tablicy, w następnym – cztery itd. Poniżej znajduje się skrócony opis kolejnych etapów.

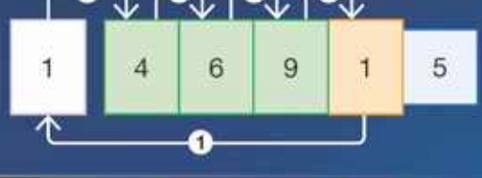
Etap 2 Zapamiętujemy liczbę 9 w zmiennej pomocniczej. Liczba 9 powinna się znaleźć za liczbami 4 i 6, dlatego nie trzeba zmieniać położenia tych liczb, a liczba 9 wraca na swoją pozycję.



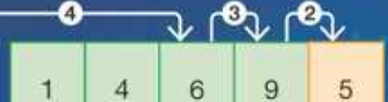
Etap 3 Zapamiętujemy liczbę 1 w zmiennej pomocniczej. Przesuwamy w prawo o jedną pozycję kolejno liczby: 9, 6 i 4. Zapisujemy liczbę 1 na pierwszym miejscu w tablicy.



Etap 4 Zapamiętujemy liczbę 5 w zmiennej pomocniczej. Przesuwamy liczbę 9, a następnie liczbę 6 o jedną pozycję w prawo. Wstawiamy liczbę 5 ze zmiennej pomocniczej w zwolnione miejsce.



Wynikiem działania algorytmu jest tablica:



Warto wiedzieć

Sortowanie bąbelkowe i sortowanie przez wstawianie przeprowadzają operacje w miejscu, co oznacza, że poza pamięcią dla sortowanej tablicy potrzebują bardzo mało dodatkowej pamięci.

Algorytm dzieli tablicę danych na część posortowaną i nieposortowaną (rys. 3.10). Na każdym etapie algorytmu kolejny element z nieposortowanej części tablicy trafia na właściwą pozycję w obszarze posortowanym.



Rys. 3.10. Umieszczenie liczby 10 w częściowo uporządkowanej tablicy

Algorytmy sortowania bąbelkowego i sortowania przez wstawianie są do siebie podobne: pierwszy stopniowo przesuwa elementy zbioru ku końcowi, podczas gdy drugi przenosi elementy zbioru ku początkowi.

Wstawienie elementu tablicy na właściwą pozycję odbywa się dzięki zwolnieniu miejsca w tablicy poprzez przesunięcie, krok po kroku, całego bloku elementów w prawo o jedną pozycję.

Przesuwanie elementów jest realizowane przez serię operacji kopiowania. Element, który będzie docelowo wstawiony we właściwe miejsce uporządkowanej części tablicy, jest tymczasowo przechowywany w zmiennej pomocniczej. Porządkowanie rozpoczynamy od drugiego elementu (o indeksie 1).

Warto wiedzieć

Operację znalezienia właściwego miejsca dla danego elementu w ciągu uporządkowanym można realizować przez przeszukiwanie liniowe lub binarne. Bardziej efektywny jest drugi sposób. Analizował go już w latach 50. XX w. polski matematyk Hugo Steinhaus.

Oto zapis algorytmu w postaci listy kroków:

1. Wczytaj tablicę liczb A.
2. Dla wartości i równych kolejno 1, 2, …, N-1 powtarzaj kroki 3-8.
   3. W zmiennej pom zapamiętaj wartość zapisaną w A[i].
   4. Oznacz bieżącą wartość i-1 literą j.
   5. Dopóki j≥0 i A[j]>pom, wykonuj kroki 6-7.
      6. Skopiuj A[j] na miejsce A[j+1].
      7. Zmniejsz wartość j o 1.
   8. W komórce A[j+1] zapisz wartość pom.

Ćwiczenie 7

Prześledź działanie powyższej listy kroków dla następujących tablic.

a. A = [2, 3, 8, 5]
b. B = [2, 8, 3, 5]
c. C = [8, 2, 3, 5]

Oto kod źródłowy funkcji o nazwie `Wstawianie` w programie *Sortowanie przez wstawianie*.

Kod źródłowy funkcji `Wstawianie` w programie *Sortowanie przez wstawianie*

```
1.  void Wstawianie(int A[])
2.  {
3.      int j, pom;
4.
5.      for (int i=1; i<N; i++)
6.      {
7.          pom = A[i];
8.          j = i - 1;
9.          while (j>=0 && A[j]>pom)
10.         {
11.             A[j+1] = A[j];
12.             j--;
13.         }
14.         A[j+1] = pom;
15.     }
16. }
```

Warto wiedzieć

Podany algorytm sortowania przez wstawianie wykonuje operacje kopiowania. Liczba tych operacji jest w przybliżeniu równa liczbie porównań. Kopiowanie nie jest jednak tak czasochłonne jak zamiana, dlatego sortowanie przez wstawianie będzie kilkukrotnie szybsze od sortowania bąbelkowego.

Pętla zewnętrzna (linie 5–15) przegląda elementy z części nieposortowanej. W kolejnych iteracjach w zmiennej pomocniczej `pom` zapamiętujemy kolejne elementy z tej części (wiersz 7). Jeżeli element z lewej strony aktualnie analizowanego elementu jest większy (`A[j]>pom`), to przesuwamy elementy posortowanej części ciągu, które powinny się znaleźć za danym elementem, a jego wartość zapamiętaną w zmiennej `pom` wstawiamy na właściwe miejsce (linie 9–13).

Warunkiem w wierszu 9 jest **koniunkcja**, czyli złożone wyrażenie logiczne. Program po skompilowaniu i uruchomieniu będzie działał poprawnie, gdyż w działaniu kompilatora C++ stosuje się tzw. **leniwe wartościowanie** (ang. *lazy evaluation*). W naszym przypadku drugie porównanie się nie wykona, jeśli pierwszy warunek nie jest spełniony, tzn. gdy indeks j będzie mniejszy niż indeks pierwszego elementu tablicy.

Logika matematyczna i operatory logiczne, s. 242–244

Leniwe wartościowanie

Ćwiczenie 8

Zapisz kod źródłowy funkcji `Wstawianie` i napisz program, w którym tę funkcję zastosujesz. Sprawdź działanie programu.

Algorytm sortowania przez wstawianie jest szczególnie efektywny przy sortowaniu zbiorów prawie uporządkowanych, np. rankingów aktualizowanych na bieżąco podczas zawodów sportowych. Wykorzystuje się go również w rozwiązaniach hybrydowych, m.in. w ostatniej fazie sortowania szybkiego (ang. *quicksort*), gdy do posortowania jest wiele podzbiorów o niewielkiej liczbie elementów.

Podsumowanie

- Sortowanie danych to ustawianie ich w zadanej kolejności według określonego kryterium.
- Uporządkowanie informacji ma ogromne znaczenie w wielu aspektach życia – zarówno w technologii komputerowej, jak i dziedzinach zupełnie z nią niezwiązanych.
- Sortowanie bąbelkowe polega na cyklicznym porównywaniu par elementów sąsiednich (zaczynając od pierwszej pary w tablicy) i zamianie miejscami tych elementów, które nie są w odpowiedniej kolejności.
- Sortowanie przez wstawianie polega na przenoszeniu jeden po drugim kolejnych elementów z nieuporządkowanego fragmentu tablicy w odpowiednie miejsce już uporządkowanego fragmentu tablicy.
- Aby funkcji zdefiniowanej w C++ przez użytkownika umożliwić dostęp do tablicy zadeklarowanej w części głównej programu (np. `int tabA[]`), w nagłówku funkcji jako parametr trzeba podać taki sam typ i dowolną nazwę tablicy, np. `Funkcja (int A[])`.
- Funkcję wywołuje się, podając jako parametr aktualny wyłącznie nazwę tablicy, np. `Funkcja (tabA)`. Wszystkie operacje są wykonywane przez funkcję na oryginalnej tablicy i zmiany są pamiętane po zakończeniu działania funkcji.

Zadania

* **1** Napisz program sprawdzający, czy tablica 10 liczb jest uporządkowana. Program powinien najpierw wczytać liczby do tablicy.

* **2** Napisz program, który po wczytaniu 10 liczb dodatnich całkowitych znajdzie wśród nich parę liczb o największej różnicy:
a. bez sortowania danych,
b. po posortowaniu danych.

* **3** Napisz program, który będzie porządkować malejąco tablicę 10 różnych liczb metodą:
a. bąbelkową,
b. przez wstawianie.

* **4** Przygotuj krótką prezentację na temat systemu „każdy z każdym", stosowanego w wielu dyscyplinach sportu. W prezentacji przedstaw zasady tworzenia par turniejowych dla kolejnych rund na przykładzie 8 zawodników (lub drużyn).

* **5** Przygotuj krótkie wystąpienie o historii powstania książki *Skafander i motyl*. Wyjaśnij, dlaczego jej autor posługiwał się w komunikacji ze światem alfabetem o zmienionym porządku liter: E S A R I N T U L O M D P C F B V H G J Q Z Y X K W.

** **6** Poszukaj informacji o algorytmie sortowania przez wybór. Napisz program, który będzie sortować liczby całkowite tą metodą.

** **7** Dodaj do programu *Sortowanie bąbelkowe* kod zliczający liczbę wykonanych prostych zamian. Podaj liczbę operacji zamian potrzebnych do uporządkowania zbioru:
a. 10-elementowego,
b. 20-elementowego,
c. 100-elementowego.
Dane do posortowania otrzymasz od nauczyciela w pliku tekstowym (np. o nazwie *posortuj_liczby.txt*).

** **8** Napisz program, który w tablicy zawierającej liczby nieujemne przesuwa na koniec tablicy wszystkie elementy o wartości 0. Porządek pozostałych elementów tablicy powinien zostać zachowany. Program nie powinien używać dodatkowej tablicy podczas przetwarzania danych.

** **9** Napisz program, który zamieni miejscami elementy znajdujące się w pierwszej i drugiej połowie tablicy o parzystej liczbie elementów.

Dane wejściowe:	3	1	4	5	2	8	0	3
Dane wyjściowe:	2	8	0	3	3	1	4	5

Program nie powinien używać dodatkowej tablicy pomocniczej.

** **10** Analizując sortowanie bąbelkowe zbioru [6, 2, 5, 4, 1], można zauważyć, że wartość 6 pojawi się na właściwym miejscu po stosunkowo niewielkiej liczbie operacji, a przeniesienie wartości 1 wymaga o wiele większej liczby operacji. Mówi się, że element o wartości 1 jest w tej sytuacji żółwiem, a element o wartości 6 – zającem. Jak zmodyfikować algorytm, żeby zaradzić tej asymetrii? Przedstaw ulepszoną wersję algorytmu, nazywaną czasami sortowaniem koktajlowym (ang. *shake sort*).

** **11** Poszukaj informacji o algorytmie sortowania bibliotecznego (ang. *library sort*), który został opublikowany w 2006 r. Przygotuj prezentację na temat tego algorytmu. Wyjaśnij pochodzenie jego nazwy.

** **12** Napisz program, który będzie porządkował wyrazy według ich długości. Wyrazy o tej samej długości należy ustawić w kolejności alfabetycznej.

*** **13** Napisz program sortujący metodą przez wstawianie z użyciem wartownika.
Wskazówka: Wartownika umieść jako dodatkowy element w pierwszej komórce tablicy zawierającej liczby do posortowania.

*** **14** W książce *Kalejdoskop matematyczny* Hugo Steinhaus opisał algorytm sortowania przez wstawianie z wyszukiwaniem binarnym. Przygotuj prezentację na temat tego algorytmu i wykorzystania w nim wagi szalkowej. Jakie znaczenie dla tego algorytmu ma ciąg 0, 1, 3, 5, 8, 11, 14, 17, 21, ...?

*** **15** Napisz program, który po wczytaniu 20 liczb dodatnich całkowitych znajdzie wśród nich parę liczb o najmniejszej różnicy.

4. Podejście zachłanne w rozwiązywaniu problemów

Wiele problemów rozwiązywanych metodami komputerowymi to problemy optymalizacji. Przykładem jest modelowanie jak najbardziej opływowego kształtu samochodu czy ustalanie jak najszybszej trasy przejazdu dla kuriera z przesyłkami. Twój komputer również rozwiązuje takiego typu problemy, np. gdy decyduje, jak rozdzielać czas pracy procesora. Z optymalizacją spotykasz się także w życiu codziennym, np. gdy w sklepie otrzymujesz resztę wydaną najmniejszą możliwą liczbą monet. Kasjer stosuje wtedy tzw. podejście zachłanne. Z tego tematu dowiesz się, na czym ono polega.

Cele lekcji

- Dowiesz się, czym są problemy optymalizacyjne.
- Zrozumiesz, na czym polega podejście zachłanne stosowane do rozwiązywania problemów.
- Zaprogramujesz algorytmy zachłanne znajdujące optymalne rozwiązania problemu wydawania reszty oraz problemu kinomana.
- Przekonasz się, jak ważny jest dobór odpowiedniego typu danych w rozwiązaniach problemów, aby uniknąć konsekwencji błędów przybliżeń.
- Nauczysz się stosować tablice równoległe jako struktury danych.

4.1. Problemy optymalizacyjne. Algorytm zachłanny

Warto wiedzieć

Przy masowej produkcji zmniejszenie ilości odpadów choćby o kilka procent daje czasami wielomilionowe oszczędności, a także znacznie redukuje wpływ przemysłu na środowisko.

O optymalizacji można mówić w różnych sytuacjach. Na przykład w ofercie warsztatu wulkanizacyjnego znajdziemy usługę optymalizacji kół. Polega ona na takim nałożeniu opony na felgę, by siły odśrodkowe podczas kręcenia się koła działały najkorzystniej i zapewniały jego prawidłową pracę.

Warsztaty stolarskie starają się, by odpadu produkcyjnego było możliwie jak najmniej, a liczba wykonanych mebli jak największa. Podobnie firmy szyjące ubrania dbają o to, by jak najmniej materiału marnować na ścinki.

W dietetyce mówi się o optymalizacji diety, czyli układaniu indywidualnych planów żywieniowych według zadanych proporcji składników odżywczych (białka, węglowodanów, tłuszczów) oraz oczekiwanej liczby kalorii.

Problem optymalizacyjny

Wymienione przykłady są właśnie **problemami optymalizacyjnymi**, czyli takimi, które wymagają znalezienia jak najlepszego rozwiązania spełniającego określone kryterium (wymaganie).

Jeśli potrafimy uzasadnić, że znalezione rozwiązanie problemu jest najlepsze z możliwych, to nazywamy je **rozwiązaniem optymalnym**. Inaczej możemy mówić tylko o **rozwiązaniu przybliżonym**.

Rozwiązanie optymalne

Rozwiązanie przybliżone

Ćwiczenie 1

Podaj przykład problemu optymalizacyjnego (inny niż opisane).

Podejście zachłanne w optymalizacji

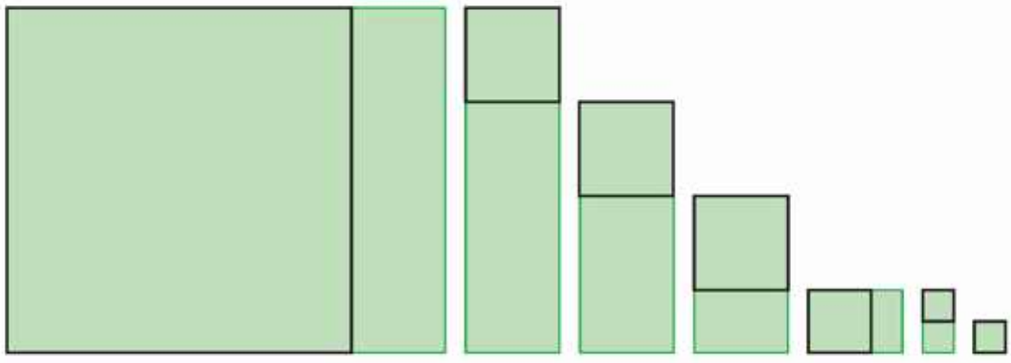
W części drugiej podręcznika omawialiśmy już jeden problem optymalizacyjny: wyznaczanie największego wspólnego dzielnika (NWD) dwóch liczb. Do rozwiązania problemu wykorzystaliśmy algorytm Euklidesa.

W algorytmie tym stosuje się metodę rozwiązywania problemów zwaną **podejściem zachłannym** (ang. *greedy approach*). Polega ono na podejmowaniu szeregu decyzji optymalnych lokalnie, czyli najlepszych (ze względu na przyjęte kryterium) na danym etapie rozwiązywania problemu. Nie sprawdzamy przy tym, jaki wpływ mają takie decyzje na znalezienie rozwiązania optymalnego globalnie, czyli najlepszego rozwiązania całego problemu.

Podejście zachłanne (metoda zachłanna)

Istotę podejścia zachłannego dobrze widać na przykładzie **wycinanki Euklidesa** przedstawionej na rysunku 4.1.

Wycinanka Euklidesa, podręcznik *Informatyka na czasie 2. Zakres podstawowy*, s. 156



Rys. 4.1. Kolejne etapy wycinanki Euklidesa

Dla liczb całkowitych dodatnich a i b wyznaczenie NWD(a,b) jest równoważne znalezieniu największego kwadratu, którym można by wypełnić prostokąt o bokach długości a i b.

W każdym kolejnym kroku algorytmu odcinamy od danego prostokąta jak największy kwadrat (postępujemy zachłannie), a w rezultacie otrzymujemy poszukiwany kwadrat.

Algorytm Euklidesa zawsze znajduje rozwiązanie optymalne. Na przykład dla liczb 112 i 88 algorytm zwraca liczbę 8, która na pewno jest największą liczbą dzielącą 112 i 88.

W części drugiej podręcznika rozważaliśmy różne algorytmy, które rozwiązują problem optymalizacyjny wyznaczenia NWD dwóch liczb, ale tylko algorytm Euklidesa wykorzystuje do tego podejście zachłanne.

Warto wiedzieć

Czasami algorytmy zachłanne pozwalają znaleźć tylko rozwiązania przybliżone. Takie algorytmy zazwyczaj są dość szybkie i można je wykorzystać jako punkt wyjścia do szukania lepszych rozwiązań.

Dobra rada

Nie używaj wyrażenia „najbardziej optymalny", gdyż jest ono niepoprawne. Coś może być najbardziej korzystne albo po prostu optymalne.

4.2. Kolorowanie mapy metodą zachłanną

Warto wiedzieć

W 1853 r. postawiono hipotezę, że cztery kolory zawsze wystarczą do pokolorowania każdej płaskiej mapy. Dowód sformułowano dopiero w 1976 r. Dzięki pracy matematyków problem udało się zredukować do jedynie 1936 układów państw na mapie. Dalsze obliczenia przeprowadził już komputer. Dotąd nikomu nie udało się udowodnić tego twierdzenia bez użycia komputera.

Podejście zachłanne ma zastosowanie nie tylko w algorytmach liczbowych. Dzięki niemu można np. próbować ustalić, ile minimum kolorów wystarczy do wypełnienia mapy w taki sposób, aby dwa sąsiednie obszary, np. państwa lub województwa (jak na rys. 4.2), miały różne kolory.



Rys. 4.2. Mapa Polski z podziałem na województwa

Warto wiedzieć

Mapę można pokolorować trzema kolorami wtedy i tylko wtedy, gdy każdy obszar na mapie (np. województwo) ma parzystą liczbę sąsiadów.

Ćwiczenie 2

W pliku, który otrzymasz od nauczyciela (np. *mapa.png*), znajduje się mapa konturowa Polski z podziałem na województwa. Pokoloruj ją, korzystając z możliwie najmniejszej liczby kolorów tak, aby dwa sąsiednie województwa miały różne barwy.

Problem kolorowania mapy

Problem kolorowania mapy, zwany też problemem ubogiego kartografa, sformułowano w połowie XIX w. Ówczesna technika drukarska pozwalała już na drukowanie barwnych map, w szczególności map politycznych. Liczbę użytych kolorów starano się jednak jak najbardziej ograniczyć, by zmniejszyć koszty konstrukcji maszyny drukarskiej.

Okazuje się, że do pokolorowania każdej płaskiej mapy wystarczą cztery kolory. Nie jest jednak znany algorytm takiego optymalnego kolorowania. Podejście zachłanne dla niektórych map daje tylko rozwiązanie przybliżone: spełniony jest warunek różnych barw dla sąsiednich obszarów, ale w sumie wykorzystuje się np. pięć barw.

4.3. Wydawanie reszty metodą zachłanną

Wyobraź sobie, że podczas zakupów otrzymujesz przy kasie resztę 8 zł i 45 gr wydaną wyłącznie monetami o wartości 1 zł oraz 5 gr. Łącznie to 17 monet. Gdyby w ten sposób wydawano resztę w sklepach, trudno byłoby ją zmieścić w portfelu, a w kasie mogłoby zabraknąć monet. Dlatego dąży się do tego, aby liczba monet była możliwie jak najmniejsza.

Problem wydawania reszty

Jest to tak zwany **problem wydawania reszty**.

Jak automaty wydają resztę?

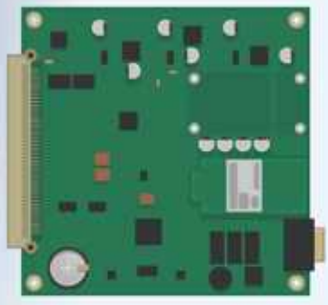
Automaty sprzedające napoje, przekąski lub bilety stały się powszechne. Część z nich pozwala płacić bezgotówkowo, ale nadal wiele przyjmuje banknoty i monety.





Rozpoznawanie monet

Monety wrzucone do automatu są poddawane działaniu fal elektromagnetycznych. Pozwala to określić grubość monet, ich wagę oraz materiał, z którego są wykonane. System zapadni sterowany przez procesor na płycie sterującej kieruje monety do właściwych przegródek.



Sterowanie automatem

Po zaakceptowaniu kwoty płyta sterująca uruchamia mechanizm wydawania produktu. Automat zwraca również resztę – tak, aby użyć najmniejszej możliwej liczby monet.

Jak wydać resztę najmniejszą liczbą monet?

1. Szukamy największego dostępnego nominału nie większego od reszty.
2. Wydajemy nominał tyle razy, ile można, aby nie przekroczyć kwoty reszty.
3. Pomniejszamy resztę do wydania o łączną wartość już wybranych monet, a następnie powtarzamy kroki 1 i 2.

Kwota do wydania	Nominał
9,45 zł	
4,45 zł	
0,45 zł	
0,05 zł	

Ćwiczenie 3

Wyobraź sobie, że korzystasz z automatu na dworcu kolejowym. Z jakich monet składa się reszta, jeśli automat wydaje ją najmniejszą możliwą liczbą monet, a ty płacisz banknotem 10 zł za zakup:

a. wody za 2,70 zł,
b. herbaty za 3,40 zł,
c. przekąski za 4,19 zł?

Zakładamy, że w automacie nie brakuje żadnych monet.

Napiszemy teraz program, który będzie wyznaczał nominały potrzebne do wydania reszty. Przyjmijmy, że reszta jest kwotą mniejszą niż 10 zł. Będzie więc wydana wyłącznie z użyciem monet o nominałach: 5 zł, 2 zł, 1 zł, 0,50 zł, 0,20 zł, 0,10 zł, 0,05 zł, 0,02 zł i 0,01 zł. Zapiszmy specyfikację problemu.

Specyfikacja

Dane: kwota reszty do wydania – liczba z przedziału [0,01; 9,99]; tablica `NOMINALY` zawierająca uporządkowane malejąco liczby od 5 do 0,01, odpowiadające dostępnym nominałom.

Wynik: kolejne wartości nominałów potrzebnych do wydania reszty.

Algorytm wydawania reszty

Zauważ, że przedstawiony na s. 65 **algorytm wydawania reszty** (zwany niekiedy algorytmem kasjera) realizuje **podejście zachłanne**. Wybieramy największy nominał, który jest mniejszy lub równy kwocie reszty, i wydajemy go tyle razy, ile można, aby nie przekroczyć tej kwoty. Następnie wybieramy kolejny nominał i powtarzamy proces. Pamiętamy przy tym, aby za każdym razem zmniejszać kwotę reszty.

Podejście zachłanne, s. 63

Oto zapis algorytmu w postaci listy kroków:

1. Wczytaj kwotę reszty do wydania i zapisz ją w zmiennej reszta.
2. Ustaw na 0 wartość zmiennej i wskazującej komórkę tablicy NOMINALY.
3. Dopóki reszta>0, wykonuj kolejno kroki 4–7.
   4. Jeśli reszta>=NOMINALY[i], to wykonaj kroki 5 i 6.
      5. Wypisz na ekranie wartość NOMINALY[i].
      6. Pomniejsz wartość zmiennej reszta o wartość NOMINALY[i].
   7. W przeciwnym przypadku zwiększ i o 1.

Zakładamy, że wartości groszowe będziemy wprowadzać do programu jako ułamki dziesiętne (stosujemy zapis z kropką zamiast przecinka). Dlatego w kodzie źródłowym programu wykorzystamy typ zmiennoprzecinkowy, a dokładniej użyjemy zmiennych **typu `float`**.

Typ float, s. 245

Oto kluczowy fragment kodu źródłowego programu *Wydawanie reszty*, zawierający definicje zmiennych oraz funkcji `WydajReszte`.

Fragment kodu źródłowego programu *Wydawanie reszty*

```
1.  const int N = 9;
2.  const float NOMINALY[N] = {5, 2, 1, 0.50, 0.20, 0.10,
3.                             0.05, 0.02, 0.01};
4.
5.  void WydajReszte(float reszta)
6.  {
7.      int i = 0;
8.      while (reszta>0 && i<N)
9.      {
10.         if (reszta >= NOMINALY[i])
11.         {
12.             cout << NOMINALY[i] << endl;
13.             reszta = reszta - NOMINALY[i];
14.         }
15.         else i++;
16.     }
17. }
```

W linii 1 zdefiniowaliśmy stałą `N`, która określa liczbę nominałów, a w liniach 2–3 **stałą tablicową** o nazwie `NOMINALY`, wypełnioną wartościami nominałów. Słowo **`const`** przed typem i nazwą tablicy oznacza, że zawartość tablicy nie może się zmienić w trakcie wykonywania programu.

Stała tablicowa

W liniach 5–17 jest zapisana funkcja `WydajReszte` **typu `void`**, której zadaniem jest wyznaczanie i wypisywanie na ekranie kolejnych nominałów potrzebnych do wydania reszty. Monet któregoś nominału w wydawanej reszcie może nie być wcale, a monet innego nominału może być kilka. Dlatego też przed przejściem do kolejnego nominału (linia 15) sprawdza się warunek z linii 10.

Funkcja niezwracająca wartości (typu `void`), s. 248

Rysunek 4.3 przedstawia przykładowe wywołanie programu *Wydawanie reszty*.



Rys. 4.3. Wywołanie programu *wydawanie_reszty.exe* dla kwoty 4 zł 78 gr

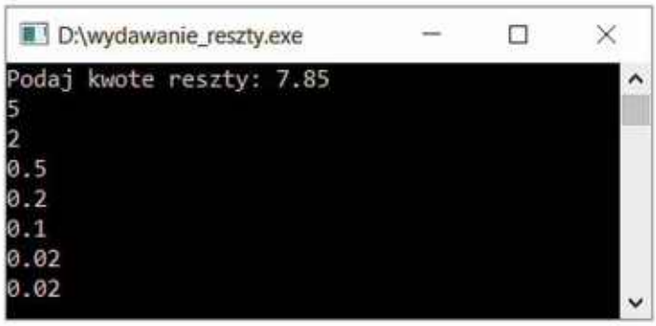
Ćwiczenie 4

W pliku *wydawanie_reszty.cpp* zapisz pełen kod źródłowy programu i skompiluj go. Sprawdź działanie programu dla różnych wartości reszty, w tym dla 7.30, 7.79 i 7.85.

Dobra rada

Pamiętaj, że liczby przechowywane są w komputerze w postaci binarnej, np. liczba 0,1 nie ma skończonego rozwinięcia dwójkowego, lecz okresowe: 0,000(1100). W konsekwencji w zmiennej typu **float** będzie zapisana wartość przybliżona, trochę większa od 0,1.

Okazuje się, że dla niektórych kwot reszty program wyświetla błędne wyniki – po zsumowaniu wartości monet brakuje 1 gr. Na przykład suma monet wyznaczonych przez program dla wartości 7.85 wynosi 7.84 (rys. 4.4).



Rys. 4.4. Błędny wynik dla kwoty 7 zł 85 gr

Niedokładny rezultat nie wynika z błędu w algorytmie ani z zastosowania podejścia zachłannego. Jest on skutkiem dodawania wartości typu **float**, które są tylko przybliżeniami właściwych wartości. Obliczenia z wykorzystaniem tego typu zmiennoprzecinkowego są obarczone konsekwencjami **błędów przybliżeń**.

Błąd przybliżenia

Nawet jeśli zwiększymy precyzję zapisu liczb przez zastosowanie **typu double**, nie uzyskamy poprawy. Dla kwoty 7.30 wyświetli się błędny wynik: 5, 2, 0.2, 0.05, 0.02, 0.02. Błąd jest rezultatem kumulowania się przybliżeń.

Typ double, s. 245

Zauważ, że program *Wydawanie reszty* – choć wyświetla niepoprawne rozwiązanie – kończy swoje działanie, nie zawiesza się. Zawdzięczamy to warunkowi i<N w warunku logicznym pętli **while**. Brak tego zapisu spowoduje zapętlenie się programu na skutek pojawienia się tzw. **pętli nieskończonej**.

Pętla nieskończona

Dobra rada

Aby zatrzymać działanie zapętlonego programu, naciśnij kombinację klawiszy **Ctrl + C**.

Ćwiczenie 5

W pliku *wydawanie_reszty.cpp* utworzonym w poprzednim ćwiczeniu usuń w kodzie drugą część warunku logicznego, tj. i<N. Plik zapisz pod nazwą *wydawanie_reszy_petla.cpp*. Skompiluj kod i sprawdź działanie programu dla różnych wartości reszty, w tym dla 7.30, 7.79 i 7.85. Co obserwujesz?

Poprawiony kod źródłowy dla algorytmu wydawania reszty

Aby rozwiązać problem błędów przybliżeń, możemy zastosować **typ całkowitoliczbowy int** i sprowadzić wszystkie kwoty reszty do wartości wyrażonych w groszach.

Typ int, s. 245

Nominały 5 zł, 2 zł i 1 zł zapisane w groszach mają wartości odpowiednio 500, 200, 100. Sformułujmy poprawioną specyfikację problemu wydawania reszty.

Specyfikacja

Dane: kwota reszty do wydania w postaci dwóch liczby całkowitych: liczba złotych z przedziału [0; 9] i liczba groszy z przedziału [0; 99]; tablica NOMINALY z informacją o dostępnych nominałach – liczby całkowite (wielokrotności 1 grosza) uporządkowane malejąco.

Wynik: kolejne wartości nominałów potrzebnych do wydania reszty podane w złotówkach.

Dobra rada

Zawsze dokładnie czytaj specyfikację problemu. Często narzuca ona typ danych, który należy zastosować w programie.

Oto fragment kodu źródłowego programu *Wydawanie reszty 2*:

Fragment kodu źródłowego programu *Wydawanie reszty 2*

```
1.  const int N = 9;
2.  const int NOMINALY[N] = {500, 200, 100, 50, 20, 10, 5,
3.                           2, 1};
4.
5.  void WydajReszte(int reszta)
6.  {
7.      int i = 0;
8.      while (reszta>0)
9.      {
10.         if (reszta >= NOMINALY[i])
11.         {
12.             cout << NOMINALY[i]/100.0 << endl;
13.             reszta = reszta - NOMINALY[i];
14.         }
15.         else i++;
16.     }
17. }
18.
19. int main()
20. {
21.     int reszta_zlote, reszta_grosze;
22.     cout << "Podaj liczbe zlotych reszty: ";
23.     cin >> reszta_zlote;
24.     cout << "Podaj liczbe groszy reszty: ";
25.     cin >> reszta_grosze;
26.     WydajReszte(reszta_zlote*100 + reszta_grosze);
27.     return 0;
28. }
```

Dobra rada

Zauważ, że po zmianie typu danych w tablicy NOMINALY zmienił się również nagłówek funkcji WydajReszte.

W liniach 22–25 pobieramy wartość kwoty do wydania – oddzielnie złotówki i grosze. Takie rozwiązanie pozwala łatwo przeliczyć złotówki na grosze. Funkcję WydajReszte wywołujemy dla parametru wyrażonego w groszach (linia 26) – w tym celu trzeba przeliczyć złotówki na grosze.

Dobra rada

Zauważ, że warunek pętli **while** (wiersz 8) jest uproszczony. Nie musisz się martwić o przekroczenie indeksów w tablicy NOMINALY, ponieważ funkcja WydajReszte operuje tylko na liczbach całkowitych.

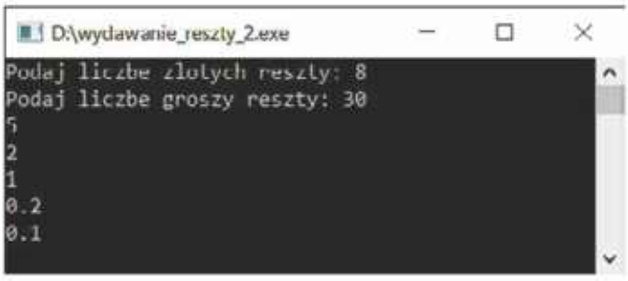
Definicja funkcji `WydajReszte` (linie 5–17) jest podobna do poprzedniej, jednak funkcja operuje już na liczbach całkowitych.

Przy wypisywaniu wartości potrzebnych monet (linia 12) podajemy wynik w postaci liczb niecałkowitych (w złotówkach). Typ niecałkowity jest wymuszony przez zastosowanie dzielenia przez wartość `100.0`, a nie przez `100`.

Przykładowe wywołanie programu pokazuje rysunek 4.5.

Warto wiedzieć

Oprogramowanie stosowane w bankach i innych instytucjach finansowych również wykonuje obliczenia na liczbach całkowitych.



Rys. 4.5. Wywołanie programu *wydawanie_reszty_2.exe* dla kwoty 8 zł 30 gr

Ćwiczenie 6

a. Zapisz kod programu w pliku *wydawanie_reszty_2.cpp*. Skompiluj kod i uruchom program dla kilku różnych kwot reszty, np. 7 zł 30 gr, 7 zł 85 gr.

b. Zmodyfikuj kod programu tak, aby można było do niego wprowadzać kwotę reszty większą niż 10 zł, a resztę można było wydawać z użyciem monet i banknotów.

Warto wiedzieć

Dopiero w 2009 r. został opublikowany artykuł naukowy wyjaśniający, jak sprawdzić, czy dla danego zbioru nominałów algorytm zachłanny na pewno znajdzie rozwiązanie optymalne niezależnie od wydawanej kwoty. Autorami artykułu było dwoje Polaków: Anna Adamaszek i Michał Adamaszek.

Optymalność zachłannego algorytmu wydawania reszty jest ściśle związana ze zbiorem dostępnych nominałów. Można sobie wyobrazić fikcyjny system monetarny, np. o nominałach 25, 10, 1, dla którego algorytm zachłanny nie znajduje optymalnego rozwiązania. Na przykład dla reszty 30 otrzymamy sumę 25 + 1 + 1 + 1 + 1 + 1 zamiast 10 + 10 + 10.

Ten przykład wystarcza, aby stwierdzić, że w ogólnym przypadku wydawanie reszty metodą zachłanną nie musi prowadzić do uzyskania **rozwiązania optymalnego**.

Rozwiązanie optymalne, s. 63

A to ciekawe

Systemy monetarne a wydawanie reszty

Systemy monetarne wielu krajów są tak skonstruowane, aby metoda zachłanna działała w nich optymalnie, tzn. żeby można było wydać resztę najmniejszą liczbą banknotów i monet. Polski system monetarny jest zbudowany na bazie liczb: 5, 2, 1. Wartości banknotów i monet są wielokrotnościami tych liczb.



Zapamiętaj

W pamięci komputera nie możemy zapisać w sposób dokładny wszystkich ułamków dziesiętnych, ponieważ niektóre z nich (np. 0,1) mają nieskończone rozwinięcie binarne. Aby zapobiec kumulowaniu się błędów przybliżeń w programach komputerowych, warto zastosować zapis wykorzystujący liczby całkowite.

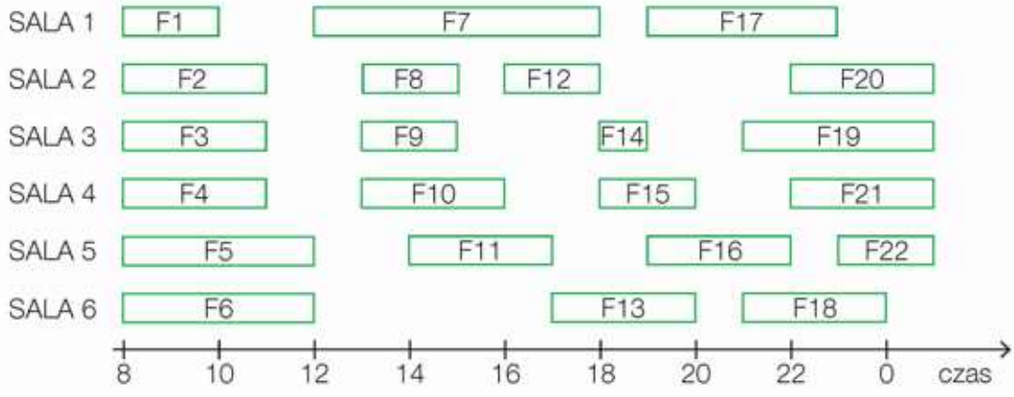
4.4. Problem kinomana

Wyobraźmy sobie, że pewien miłośnik filmu bierze udział w jednodniowym festiwalu filmowym. W opublikowanym programie festiwalu znajdują się informacje takie jak: tytuł filmu, gatunek filmu, godzina rozpoczęcia seansu, czas projekcji, numer sali kinowej.

Kinoman chce zobaczyć tyle filmów, ile tylko będzie możliwe. Zakładamy, że nie może on w tej samej godzinie zakończyć seansu jednego filmu i rozpocząć drugiego. Potrzebuje czasu na przemieszczenie się, a filmy ogląda w całości. W jaki sposób powinien postępować, aby obejrzeć ich jak najwięcej? Tak sformułowany problem znany jest w literaturze jako **problem kinomana**.

Problem kinomana

Harmonogram festiwalu można przedstawić graficznie tak jak na rysunku 4.6. Filmy są reprezentowane przez prostokąty ułożone chronologicznie wzdłuż linii czasu. Długość prostokąta odpowiada długości trwania filmu. Dodatkowo każdy film ma identyfikator od F1 do F22.



Rys. 4.6. Harmonogram festiwalu filmowego

Dobra rada

Zauważ, że numery w identyfikatorach filmów odzwierciedlają kolejność ich rozpoczęcia.

Aby wybrać możliwie najwięcej filmów, możemy postępować zachłannie według różnych kryteriów. Oto niektóre z nich:

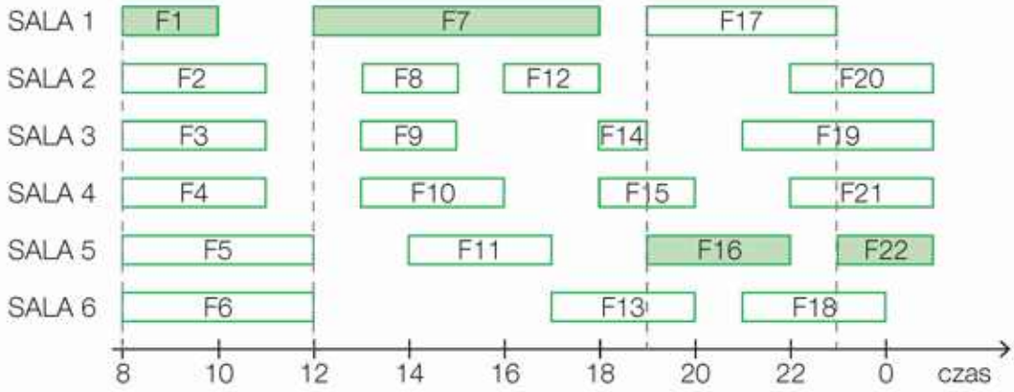
- kryterium 1: filmy zaczynające się najwcześniej,
- kryterium 2: filmy najkrótsze,
- kryterium 3: filmy kończące się najwcześniej.

Załóżmy ponadto, że jeśli kilka filmów spełnia dane kryterium, to wybieramy ten, który kończy się najwcześniej. Jeśli takich filmów jest więcej, wybieramy dowolny.

Warto wiedzieć

Innym kryterium, które można zastosować w problemie kinomana, jest jak najmniejsza liczba konfliktów filmu z innymi filmami.

Rysunek 4.7 przedstawia rozwiązanie z uwzględnieniem kryterium 1. W pierwszej kolejności wybieramy najwcześniej rozpoczynający się film (F1). Następnie szukamy filmu, który rozpocznie się najwcześniej po zakończeniu pierwszego seansu – będzie to film F7. Po jego zakończeniu najwcześniej rozpocznie się film F16, a ostatnim wybranym filmem jest F22.



Rys. 4.7. Filmy wybrane na podstawie kryterium 1

Ćwiczenie 7

W pliku otrzymanym od nauczyciela (np. *kinoman.png*) oznacz filmy, które obejrzy kinoman, jeśli zastosuje kryteria 2 oraz 3.

Można wykazać, że zachłanny wybór najwcześniej kończących się filmów (kryterium 3) daje zawsze **rozwiązanie optymalne**.

Rozwiązanie optymalne, s. 63

Zapiszmy specyfikację problemu kinomana bardziej precyzyjnie i zastanówmy się nad algorytmem wykorzystującym to kryterium.

Specyfikacja

Dane: liczba naturalna `N` – liczba filmów; trzy tablice `N`-elementowe liczb całkowitych:

`P` – z godzinami rozpoczęcia projekcji kolejnych filmów,

`K` – z godzinami zakończenia projekcji kolejnych filmów,

`ID` – z identyfikatorami kolejnych filmów.

Wynik: identyfikatory i godziny rozpoczęcia filmów tworzących optymalne rozwiązanie.

Dobra rada

Jeśli chcesz udowodnić, że wybór najwcześniej kończących się filmów daje optymalne rozwiązanie, załóż, że istnieje lepsza metoda, która da większą liczbę filmów. Spróbuj wykazać fałszywość przyjętego założenia.

Załóżmy dla uproszczenia, że czas projekcji jest wielokrotnością godziny (60 minut).

Aby wyłonić listę filmów, korzystając z metody zachłannej, wystarczy posortować filmy według godziny ich zakończenia, a następnie wybrać pierwszy z tych filmów i wielokrotnie (w pętli) powtórzyć ten sam sposób wyboru dla listy filmów, które rozpoczynają się po zakończeniu poprzedniego filmu.

Przedstawmy zapis algorytmu rozwiązania problemu kinomana w postaci listy kroków:

1. Wczytaj dane do tablic P, K oraz ID.
2. Uporządkuj tablicę K niemalejąco i dostosuj kolejność elementów w tablicach P oraz ID.
3. Wypisz na ekranie ID[0], P[0] oraz K[0]. Wartość K[0] zapisz w zmiennej koniec_poprzedniego.
4. Oznacz jako i zmienną wskazującą komórkę tablicy i ustaw ją na 1.
5. Dla wartości i równych kolejno 1, 2, …, N-1 powtarzaj kroki 6-8.
   6. Jeśli P[i]>koniec_poprzedniego, to wykonaj kroki 7-8.
      7. Wypisz na ekranie ID[i], P[i] oraz K[i].
      8. Zapisz wartość K[i] w zmiennej koniec_poprzedniego.

Warto wiedzieć

Jeśli informacja o jednym filmie znajduje się w kilku tablicach, to zmiana pozycji jednego elementu musi powodować analogiczne zmiany odpowiednich elementów w pozostałych tablicach.

Do sortowania filmów można użyć **algorytmu sortowania przez wstawianie**. Oto kod źródłowy funkcji sortującej Sortuj w programie *Problem kinomana*:

Sortowanie przez wstawianie, s. 56–59

Kod źródłowy funkcji Sortuj w programie *Problem kinomana*

```
1.  void Sortuj(int koniec[], int poczatek[], string id[])
2.  {
3.      int i, j, pom_koniec, pom_poczatek;
4.      string pom_id;
5.
6.      for (i=1; i<N; i++)
7.      {
8.          pom_koniec = koniec[i];
9.          pom_poczatek = poczatek[i];
10.         pom_id = id[i];
11.         j = i - 1;
12.
13.         while (j>=0 && koniec[j]>pom_koniec)
14.         {
15.             koniec[j+1] = koniec[j];
16.             poczatek[j+1] = poczatek[j];
17.             id[j+1] = id[j];
18.             j--;
19.         }
20.
21.         koniec[j+1] = pom_koniec;
22.         poczatek[j+1] = pom_poczatek;
23.         id[j+1] = pom_id;
24.     }
25. }
```

Dobra rada

Pamiętaj, że tablica koniec przechowuje godziny zakończenia kolejnych seansów. Dlatego to dla jej elementów wykonywane są operacje porównania.

Warto wiedzieć

Sortowanie danych w powiązanych ze sobą tablicach jest podobne do sortowania w tabelach arkusza kalkulacyjnego. Zmiana kolejności danych w jednej kolumnie powoduje odpowiednie modyfikacje w pozostałych kolumnach tabeli.

Funkcja `Sortuj` ma 3 parametry – tablice zawierające informacje o końcach i początkach seansów oraz identyfikatorach filmów. Zwróć uwagę, że choć warunek pętli **`while`** (linia 13) odnosi się wyłącznie do tablicy `koniec`, to równolegle modyfikowane są pozostałe tablice: `początek` oraz `id`. Pierwszy warunek pętli **`while`** (linia 13) sprawdza, czy indeks jest liczbą mniejszą od rozmiaru tablicy, a drugi warunek sprawdza, czy w tablicy są jeszcze filmy do uporządkowania.

Dzięki temu te same identyfikatory we wszystkich trzech tablicach nadal będą się odnosić do tego samego filmu. Tak powiązane ze sobą tablice nazywa się **tablicami równoległymi**.

Tablice równoległe

Poniżej znajdują się: kod źródłowy funkcji `Wybierz`, której zadaniem jest wybranie możliwie największej liczby filmów metodą zachłanną, oraz kod źródłowy funkcji `Wypisz`, która ułatwia wypisanie interesujących nas informacji o filmie.

Kod źródłowy funkcji `Wypisz` oraz `Wybierz` w programie *Problem kinomana*

```
1.  void Wypisz(int koniec[], int poczatek[], string id[],
2.               int k)
3.  {
4.      cout << id[k] << ": " << poczatek[k] << "-";
5.      cout << koniec[k] << endl;
6.  }
7.
8.  void Wybierz(int koniec[], int poczatek[], string id[])
9.  {
10.     int i, koniec_poprzedniego;
11.
12.     Wypisz(koniec, poczatek, id, 0);
13.     koniec_poprzedniego = koniec[0];
14.
15.     for (i=1; i<N; i++)
16.         if (poczatek[i] > koniec_poprzedniego)
17.         {
18.             Wypisz(koniec, poczatek, id, i);
19.             koniec_poprzedniego = koniec[i];
20.         }
21. }
```

Dobra rada

Pamiętaj, że kompilator analizuje kod źródłowy linijka po linijce. Dlatego jeśli chcesz napisać własną funkcję wykorzystującą inną własną funkcję (np. tak jak w funkcji `Wybierz` korzystamy z funkcji `Wypisz`), to wykorzystywaną funkcję zdefiniuj przed jej wywołaniem.

Funkcja `Wypisz` pozwala wypisać informacje o filmie: identyfikator oraz godzinę rozpoczęcia i zakończenia seansu. Kod funkcji `Wybierz` znajduje się w liniach 8–21. W linii 12 wypisujemy na ekranie dane filmu o indeksie 0. Jeśli wcześniej posortujemy filmy, to pierwszy film (o indeksie 0) jest tym, który powinien obejrzeć kinoman. Następnie wybierane są filmy, które rozpoczną się bezpośrednio po aktualnie wybranym (linia 16) i mają najwcześniejszą godzinę zakończenia.

Oto kod źródłowy funkcji głównej programu *Problem kinomana*:

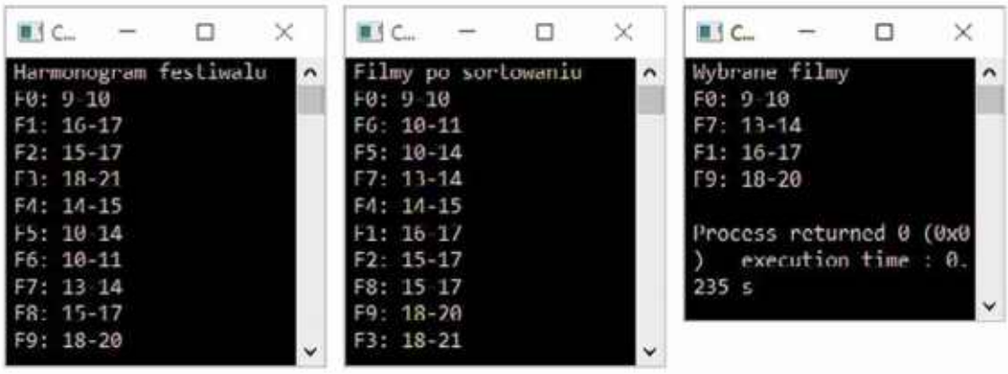
Kod źródłowy funkcji głównej w programie *Problem kinomana*

```
1.  int main()
2.  {
3.      int P[N] = {9, 16, 15, 18, 14, 10, 10, 13, 15, 18};
4.      int K[N] = {10, 17, 17, 21, 15, 14, 11, 14, 17, 20};
5.      string ID[N] = {"F0", "F1", "F2", "F3", "F4", "F5",
6.                      "F6", "F7", "F8", "F9"};
7.
8.      cout << "Harmonogram festiwalu" << endl;
9.      for (int i=0; i<N; i++)
10.         Wypisz(K, P, ID, i);
11.
12.     Sortuj(K, P, ID);
13.     cout << endl << "Filmy po sortowaniu" << endl;
14.     for (int i=0; i<N; i++)
15.         Wypisz(K, P, ID, i);
16.
17.     cout << endl << "Wybrane filmy" << endl;
18.     Wybierz(K, P, ID);
19.
20.     return 0;
21. }
```

Dobra rada

Jeśli w pętli wykonywana jest tylko jedna instrukcja (np. instrukcja wywołania funkcji), to możesz pominąć nawiasy klamrowe. Pamiętaj jednak, by po instrukcji wstawić średnik.

Rysunek 4.8 przedstawia przykładowe wywołanie programu *Problem kinomana*. Poniższe ekrany pokazują kolejne etapy: wyświetlenie harmonogramu, listy filmów po posortowaniu oraz listy filmów do obejrzenia.



Rys. 4.8. Przykładowe wywołanie programu *problem_kinomana.exe*

Warto wiedzieć

Znajomość algorytmu rozwiązującego problem kinomana może się przydać osobom chcącym uczestniczyć w największych festiwalach filmowych na świecie. Na przykład berliński festiwal Berlinale co roku oferuje do wyboru ok. 400 filmów z całego świata.

Ćwiczenie 8

Zapisz pełen kod źródłowy programu *Problem kinomana*. Skompiluj kod i uruchom program.

Podsumowanie

- Problemy optymalizacyjne to problemy, w których szukamy rozwiązania jak najlepszego, spełniającego określone kryterium.
- Podejście zachłanne w rozwiązywaniu problemu polega na podejmowaniu szeregu decyzji, z których każda wydaje się najkorzystniejsza w danym momencie, bez analizowania ich wpływu na optymalność całego rozwiązania. Stosowanie tej strategii prowadzi czasami do rozwiązania optymalnego, a czasami tylko do rozwiązania przybliżonego.
- Problem wydawania reszty polega na przedstawieniu danej kwoty reszty za pomocą jak najmniejszej liczby banknotów i monet. Aby rozwiązać ten problem metodą zachłanną, należy wybierać w danym kroku jak największy nominał nie większy od kwoty, która pozostała jeszcze do wydania.
- Aby zapobiec kumulowaniu się błędów przybliżeń w języku C++, należy unikać używania typów zmiennoprzecinkowych, np. `float` i `double`.

Zadania

* 1 Napisz program, który liczbę wpisaną z klawiatury, mniejszą niż 256, zapisze jako sumę potęg liczby 2. Zdefiniuj w kodzie tablicę o wartościach 128, 64, 32, 16, 8, 4, 2, 1.

* 2 W pliku, który otrzymasz od nauczyciela (np. *mapy_konturowe.pdf*), znajdują się różne mapy. Korzystając z metody zachłannej, wyznacz minimalną liczbę kolorów, której trzeba użyć do pokolorowania każdej mapy tak, aby każde dwa sąsiednie obszary miały różne kolory.

* 3 Napisz program poszukujący rozwiązania problemu wydawania reszty, gdy masz ograniczoną liczbę każdego z nominałów. Zwróć uwagę, że nie każdą resztę uda się wydać (program powinien wtedy wypisać odpowiedni komunikat).

* 4 Zmodyfikuj kod źródłowy programu *Problem kinomana* tak, aby dane wczytywać z klawiatury.

** 5 Napisz program, który wczyta z klawiatury informacje o kolegach i koleżankach z klasy: imię, pseudonim oraz kolor oczu. Po wprowadzeniu danych program powinien uporządkować je według koloru oczu i wyświetlić informacje o osobach rozdzielone na grupy w zależności od koloru oczu. Grupa najliczniejsza powinna być wyświetlona na początku, a grupa najmniej liczna – na końcu. Zastosuj tablice równoległe.

*** 6 Na świecie rzadko stosuje się systemy monetarne, w których nie ma nominału 2. Przykładem jest szyling kenijski, którego monety mają wartości: 1, 5, 10 i 20 szylingów, a banknoty: 50, 100, 200, 500 i 1000 szylingów. Napisz program wydający resztę w tym systemie z użyciem podejścia zachłannego. Czy algorytm zachłanny jest w tym wypadku optymalny?

5. Rekurencja

W programowaniu rozwiązań problemów stosowaliśmy do tej pory iterację, czyli technikę polegającą na wielokrotnym powtarzaniu – w pętli – instrukcji zapisanych w tekście programu. Istnieje jednak inna technika programistyczna, która również wiąże się z powtarzaniem tego samego fragmentu kodu, choć instrukcja pętli jawnie w kodzie źródłowym nie występuje. Jest to rekurencja. W tym temacie dowiesz się, czym jest funkcja rekurencyjna, jakie ma zalety, a także dlaczego należy jej używać z rozwagą.

Cele lekcji

- Dowiesz się, czym jest rekurencja, oraz zrozumiesz istotę rekurencyjnego rozwiązywania problemów.
- Poznasz przykład prostego fraktala w postaci drzewa binarnego.
- Nauczysz się zapisywać funkcje rekurencyjne w języku C++.
- Poznasz ograniczenia, jakie ma stosowanie techniki rekurencji.
- Nauczysz się na przykładach, jak rekurencję zastąpić iteracją i na odwrót.

5.1. Czym jest rekurencja?

Rekurencja lub **rekursja** (ang. *recursion*) w programowaniu oznacza odwołanie się w wybranym kroku algorytmu rozwiązywania problemu do tego samego problemu, ale dla mniejszego rozmiaru danych. Algorytm, w którym stosuje się takie podejście, nazywamy **algorytmem rekurencyjnym**.

Rekurencja (rekursja)

Algorytm rekurencyjny

Podejście rekurencyjne w rozwiązywaniu problemu algorytmicznego jest procesem dwufazowym. Pierwsza faza to redukcja rozmiaru problemu, druga to łączenie częściowych wyników pochodzących z rozwiązania tych mniejszych problemów.

Jeśli danego problemu nie potrafimy rozwiązać, gdyż n-elementowy zbiór danych jest zbyt złożony, to umiejętnie redukujemy go do podobnego problemu, ale dla zbioru o np. $n - 1$ elementach. Taki problem może być łatwiej rozwiązać. Jeśli ten mniejszy problem wciąż jest zbyt złożony, to możemy dalej redukować jego złożoność, aż dojdziemy do takiego problemu, który potrafimy rozwiązać od razu. Później, w fazie zbierania i łączenia częściowych wyników, uzyskujemy rozwiązanie całego problemu.

Aby lepiej zrozumieć rekurencję, przyjrzyjmy się różnym jej przykładom.

Warto wiedzieć

Efekt podobny do rekurencyjnego można uzyskać, wykorzystując lustra i aparat fotograficzny w sposób pokazany na zdjęciu. Można też stanąć plecami do dużego lustra i zrobić zdjęcie przednim aparatem telefonu (tzw. selfie).

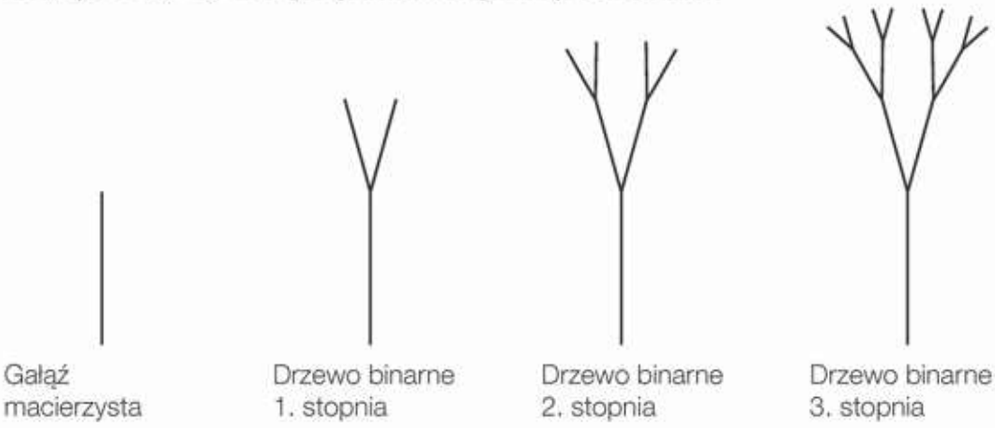


Rekurencyjna budowa drzewa binarnego

Fraktal to figura geometryczna, której podstawową cechą jest samopodobieństwo (podobieństwo wewnętrzne, powtarzalność). Oznacza to, że we fraktalu można wyodrębnić fragmenty podobne do całej figury.

Fraktal

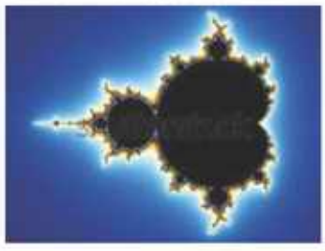
Drzewo binarne Oto kilka przykładów **drzew binarnych** o różnym stopniu złożoności (rys. 5.1). Są one przykładami prostych fraktali.



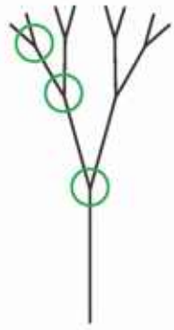
Rys. 5.1. Drzewa binarne o różnym stopniu złożoności

Warto wiedzieć

Pojęcie fraktala wprowadził w latach 70. XX w. Benoît B. Mandelbrot, urodzony w Warszawie w rodzinie litewskich Żydów, którzy po I wojnie światowej zamieszkali w Polsce. Jednym z najbardziej znanych fraktali jest zbiór Mandelbrota, nazywany również żukiem Mandelbrota.



Drzewo 1. stopnia składa się z dwóch gałęzi odchodzących od gałęzi macierzystej. Gałęzie te mają długość równą 0,6 długości gałęzi macierzystej i są odchylone od kierunku przez nią wyznaczonego o ten sam kąt – 15 stopni w prawo i w lewo. Gałęzie końcowe drzewa 2. stopnia mają te same parametry w odniesieniu do gałęzi drzewa 1. stopnia. Analogicznie są zbudowane drzewa kolejnych stopni.



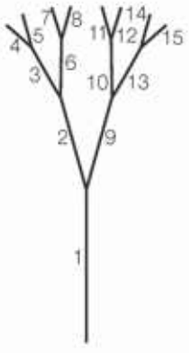
Rys. 5.2. Części samopodobne drzewa binarnego

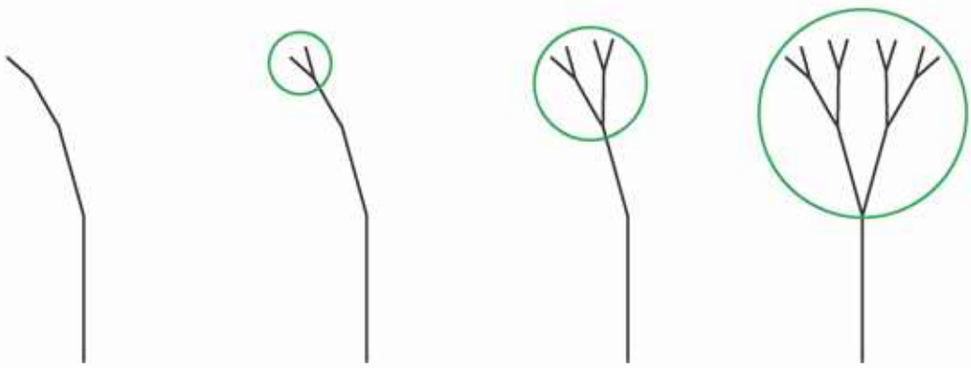
Rysunek 5.2 przedstawia drzewo binarne z zaznaczonymi miejscami rozwidlenia, które wskazują na cechę samopodobieństwa: fragmenty z wyższych części drzewa są podobne do całości. Drzewo 3. stopnia składa się z gałęzi macierzystej oraz dwóch drzew 2. stopnia odpowiednio pomniejszonych i obróconych.

Programy rysujące fraktale najczęściej wykorzystują funkcje rekurencyjne. Rysunek 5.3 przedstawia wybrane etapy rysowania drzewa binarnego 3. stopnia realizowanego przez funkcję rekurencyjną.

Warto wiedzieć

Odcinki tworzące drzewo binarne 3. stopnia są rysowane w następującej kolejności:





Rys. 5.3. Proces rekurencyjnego rozrastania się drzewa binarnego 3. stopnia

Najpierw rysowana jest gałąź macierzysta. Następnie powstaje drzewo 2. stopnia (w odpowiedniej skali) odchylone w lewo, a dopiero po jego narysowaniu – drzewo 2. stopnia odchylone w prawo.

Aby narysować każde z drzew 2. stopnia, należy w podobny sposób narysować drzewa 1. stopnia (rozpoczynając od lewej strony).

Drzewo binarne rozrasta się w sposób rekurencyjny: z każdego rozwidlenia wyrastają dokładnie dwa drzewa niższego stopnia.

Rekurencyjny przepis na ciasto francuskie

Ciasto francuskie charakteryzuje się dużą ilością zawartego w nim powietrza. Zawdzięczamy to warstwowej strukturze uzyskiwanej przez wielokrotne składanie ciasta (rys. 5.4).

Aby przygotować ciasto francuskie, należy na pewnym etapie wielokrotnie wykonywać takie same polecenia: ciasto rozwałkuj na długość równą 3-krotności jego szerokości. Następnie złóż je na 2 razy, aby uzyskać 3 warstwy i schłodź. Powtórz te czynności jeszcze 5 razy.



Rys. 5.4. Wałkowanie i składanie ciasta francuskiego

Warto wiedzieć

Gdyby fragment przepisu na ciasto francuskie dotyczący wałkowania, składania i chłodzenia ciasta zapisać nieco bardziej formalnie, mógłby on brzmieć następująco:

```
Rozwałkuj-Złóż-Schłodź(n):
  jeśli n = 1, to zakończ
  rozwałkuj ciasto
  złóż je na 3 warstwy
  schłodź przez 1 godzinę
  Rozwałkuj-Złóż-Schłodź(n-1)
```

Zauważ, że tak sformułowany opis odwołuje się do samego siebie w słowach „Powtórz te czynności". Jest to cecha każdego algorytmu rekurencyjnego.

Ćwiczenie 1

Tabliczka czekolady składa się z 16 kawałków (4 rzędy po 4 kawałki). Zapisz rekurencyjny algorytm łamania czekolady na 16 pojedynczych kostek.

A to ciekawe

Efekt Droste – rekurencja w projektowaniu

Około 1900 r. na puszce kakao produkowanego przez holenderską firmę cukierniczą Droste pojawił się wizerunek pielęgniarki, która trzymała w ręku tacę z filiżanką oraz puszką kakao Droste. Na tej puszce widać pielęgniarkę trzymającą w ręku tacę z filiżanką oraz puszką kakao Droste itd. Ten sposób prezentacji grafiki został pod koniec lat 70. XX w. nazwany efektem Droste. Jest to rodzaj rekurencyjnego obrazu. Również dziś można zaobserwować ten zabieg m.in. na puszce polskiego mleka zagęszczonego.



5.2. Rekurencyjna definicja ciągu

Warto wiedzieć

Ciąg (c_n) jest ciągiem geometrycznym. Do obliczania jego wyrazów można więc użyć wzoru: $c_n = 3^{n-1}$

Wróćmy raz jeszcze do przykładu ciasta francuskiego. Ciasto powstałe jako efekt pierwszego wałkowania możemy nazwać ciastem początkowym o numerze 1. Z niego powstaje 3-warstwowe ciasto o numerze 2, a z niego 9-warstwowe ciasto o numerze 3 itd.

Zauważmy, że i-te ciasto powstaje z ciasta o numerze $i - 1$, które po złożeniu na trzy części zostaje rozwałkowane. Ciąg opisujący liczbę warstw ciasta to: 1, 3, 9, 27, ... Z matematycznego punktu widzenia można

Ciąg zdefiniowany rekurencyjnie

mówić o ciągu c_n **zdefiniowanym rekurencyjnie** w następujący sposób:

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

Warunek początkowy

Pierwszą część definicji nazwiemy **warunkiem początkowym**. Określa on, kiedy zakończyć proces odwołań rekurencyjnych z drugiej części definicji. Zgodnie z drugą częścią definicji kolejny wyraz ciągu wyznacza się, korzystając z wartości wyrazu poprzedniego.

Aby obliczyć wartość c_n dla danego n, należy wcześniej wyznaczyć wyraz c_{n-1}. Żeby jednak obliczyć tę wartość, należy znać wyraz c_{n-2} itd. Proces redukowania wartości indeksu n skończy się, gdy n przyjmie wartość 1 (warunek początkowy).

Wyrazy ciągów zdefiniowanych rekurencyjnie można łatwo obliczyć w arkuszu kalkulacyjnym. Dla powyższego ciągu wystarczy wpisać w komórce C1 wartość wyrazu początkowego, czyli 1, a w komórce C2 – formułę opisującą zależność rekurencyjną: =3∗C1, i skopiować formułę w dół. Wartość w komórce C4 (rys. 5.5) jest obliczana błyskawicznie. Arkusz wyznacza ją jednak krok po kroku, ponieważ formuła w komórce C4 odwołuje się do formuły z komórki C3, która odwołuje się do formuły z komórki C2 itd. Łańcuch kolejnych wywołań rekurencyjnych (rys. 5.6) kończy się w komórce C1, zawierającej stałą wartość. Dalej następuje faza zbierania wyników, która kończy się wstawieniem do komórki C4 wyniku 27.

	C
1	1
2	3
3	9
4	=3*C3

Rys. 5.5. Ciąg w arkuszu kalkulacyjnym

Dobra rada

Jeśli chcesz prześledzić krok po kroku obliczenia wykonywane w formułach arkusza kalkulacyjnego, skorzystaj z narzędzia **Szacuj formułę**, dostępnego na karcie **Formuły**.



Rys. 5.6. Podgląd wywołań rekurencyjnych w narzędziu **Szacuj formułę**

Napiszemy teraz programy obliczające wybrany wyraz ciągu rekurencyjnego. Sformułujmy specyfikację problemu.

Specyfikacja

Dane: liczba naturalna n.

Wynik: n-ty wyraz ciągu danego wzorem:

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

Program *Ciąg rekurencyjnie*

Zapiszmy funkcję o nazwie `CiagRek`, która będzie wyznaczała n-ty wyraz ciągu w sposób rekurencyjny. Oto jej kod źródłowy:

Kod źródłowy funkcji `CiagRek` w programie *Ciąg rekurencyjnie*

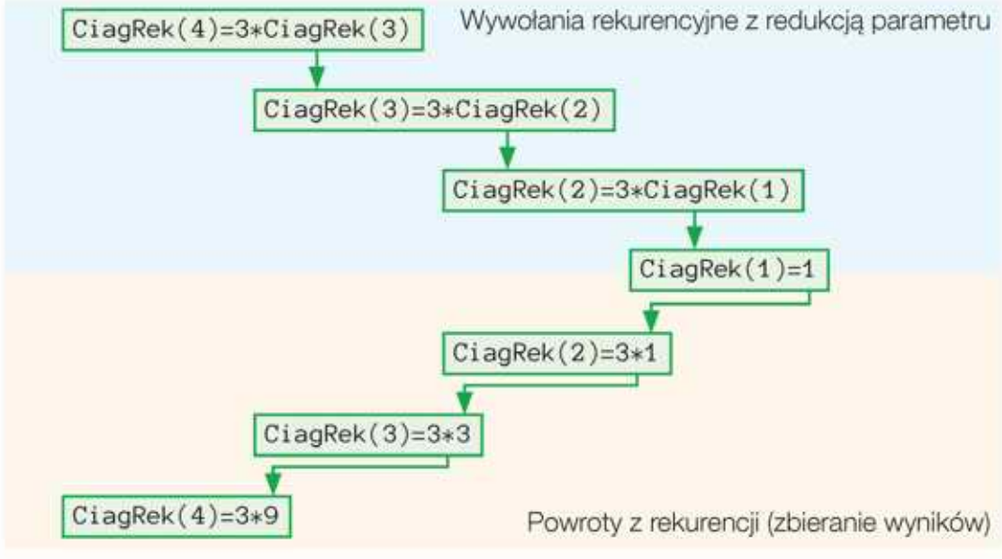
```
1.  int CiagRek(int n)
2.  {
3.      if (n == 1) return 1;
4.      return 3 * CiagRek(n-1);
5.  }
```

Budowa wnętrza **funkcji rekurencyjnej** odzwierciedla dwa możliwe przypadki. Pierwszy to warunek początkowy (linia 3), drugi to kod **wywołania rekurencyjnego**, czyli wywołania funkcji przez samą siebie, ale dla innej wartości parametru (linia 4).

Funkcja rekurencyjna

Wywołanie rekurencyjne

Na rysunku 5.7 przedstawiono fazy działania funkcji `CiagRek`: wywołania rekurencyjne dla coraz mniejszych wartości parametru n oraz powroty z rekurencji (zbieranie wyników).



Rys. 5.7. Schemat wywołań rekurencyjnych funkcji `CiagRek` i powrotów z rekurencji

W pierwszej części schematu strzałki wskazują kolejne wywołania rekurencyjne funkcji `CiagRek` dla coraz mniejszych parametrów. Druga część schematu to faza zbierania wyników – strzałki pokazują, które wartości są przekazywane dalej.

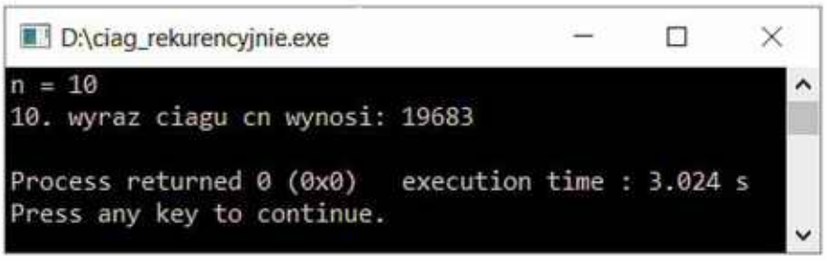
Dobra rada

Nazwa „rekurencja" pochodzi od łacińskiego słowa *recurrere*, czyli „przybiec z powrotem". Jeśli chcesz zrozumieć daną funkcję rekurencyjną, przeanalizuj dokładnie właśnie powroty z rekurencji.

Warto wiedzieć

Kody źródłowe funkcji, które są zapisem algorytmów rekurencyjnych, same mają strukturę podobną do rekurencyjnej definicji ciągu. Składają się z dwóch części: pierwsza określa warunek początkowy, a druga opisuje, jak przypadek dla parametru n zredukować do prostszego (np. $n - 1$).

Rysunek 5.8 przedstawia przykładowe wywołanie programu *Ciąg rekurencyjnie*.



Rys. 5.8. Przykładowe wywołanie programu *ciag_rekurencyjnie.exe*

Ćwiczenie 2

a. W pliku otrzymanym od nauczyciela (np. *wyraz_ciagu_rek.cpp*) znajduje się fragment kodu źródłowego zawierający funkcję `CiagRek`. Dopisz kod funkcji głównej, skompiluj kod programu i sprawdź jego działanie.

b. Napisz program, który obliczy wartość n-tego wyrazu ciągu a_n danego wzorem:

$$a_n = \begin{cases} 6 & \text{dla } n = 1 \\ 2 \cdot a_{n-1} & \text{dla } n > 1 \end{cases}$$

Sprawdź działanie programu i podaj wynik dla $n = 12$

Program *Ciąg iteracyjnie*

Pętla `for`, s. 247

Wyraz ciągu c_n zdefiniowanego rekurencyjnie można również obliczyć w sposób iteracyjny. Zastosujemy w tym celu pętlę **`for`**, gdyż liczba cykli pętli potrzebnych do wyznaczenia n-tego wyrazu ciągu jest z góry określona: będzie ich $n - 1$.

Kod źródłowy funkcji `CiagIter` w programie *Ciąg iteracyjnie*

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

Dobra rada

Instrukcję przypisania `c = 3 * c;` możesz zapisać krócej jako `c *= 3;`.

W linii 3 zmiennej `c` przypisywana jest wartość 1. Jeśli użytkownik wywoła funkcję `CiagIter` 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.

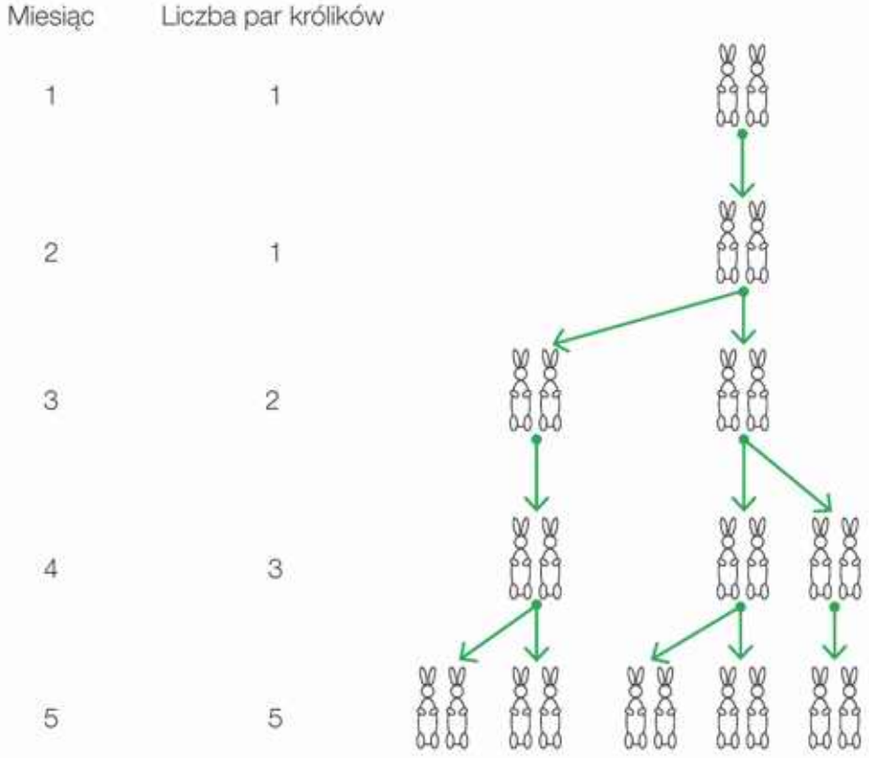
Ćwiczenie 3

Zapisz kod źródłowy programu *Ciąg iteracyjnie*. Skompiluj kod i uruchom program dla różnych wartości.

5.3. 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? Wyjaśnia to rysunek 5.9.



Rys. 5.9. 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. 5.9). 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 rekurencyjną:

○ Ciąg Fibonacciego

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

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.

Złota proporcja

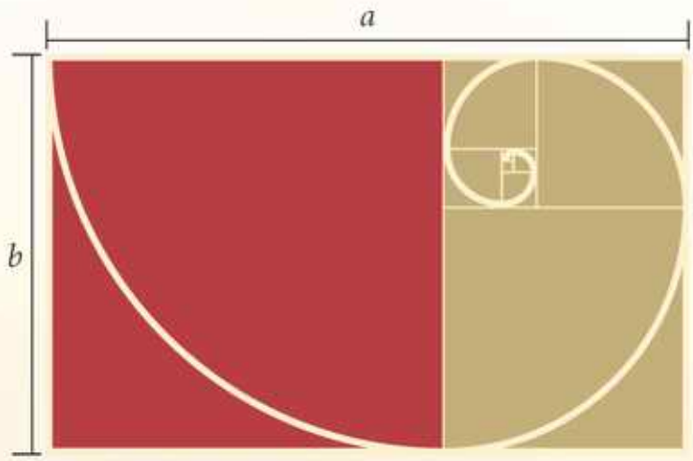
Złota proporcja, nazywana też złotym podziałem, znana jest od wieków. Przyrodnicy zachwycają się jej pięknem, którego przykłady można odnaleźć w naturze. Zainspirowało to wiele osób do poszukiwania i stosowania złotego podziału w różnych dziedzinach życia.

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 czerwono), to uzyskamy mniejszy prostokąt (na rysunku złoty) 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



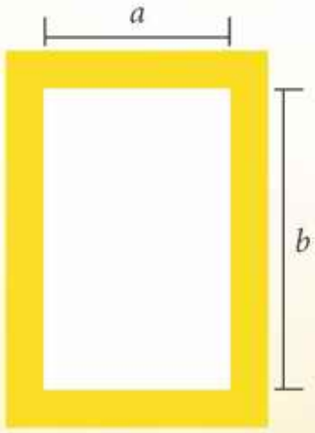
Złota proporcja w projektowaniu

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



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

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

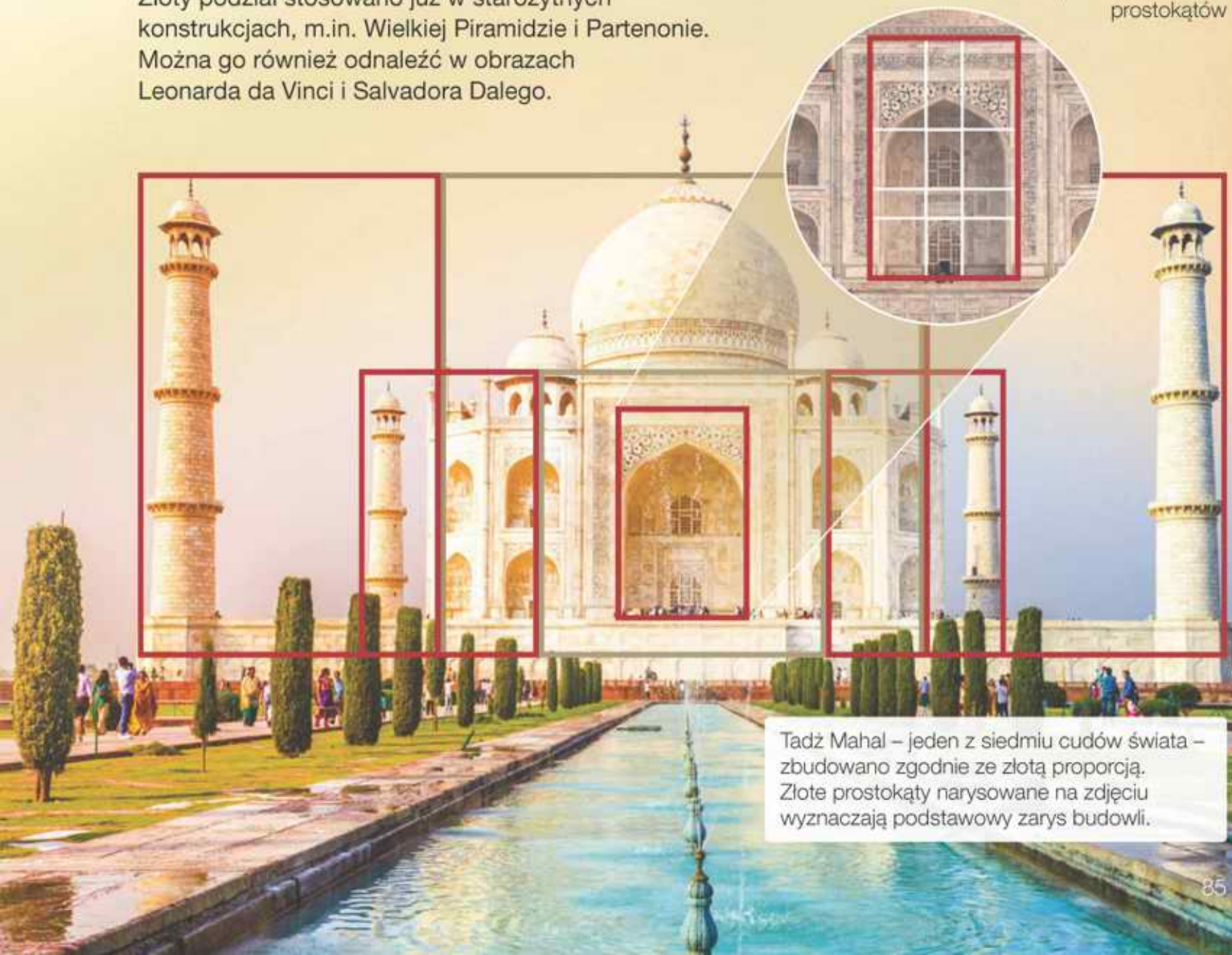


Wewnętrzny prostokąt w logo National Geographic zachowuje złotą proporcję

Złota proporcja w architekturze i sztuce

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

Okna i drzwi wyznaczają linie przecięcia złotych prostokątów



Tadż Mahal – jeden z siedmiu cudów świata – zbudowano zgodnie ze złotą proporcją. Złote prostokąty narysowane na zdjęciu wyznaczają podstawowy zarys budowli.

Warto wiedzieć

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

Napiszemy programy obliczające dany wyraz ciągu Fibonacciego. Przyjmijmy następującą specyfikację problemu:

Specyfikacja

Dane: liczba całkowita $n \geqslant 0$.

Wynik: n-ty wyraz ciągu Fibonacciego.

Program *Fibonacci rekurencyjnie*

Zapisanie funkcji rekurencyjnej sprowadza się do przepisania na wybrany język programowania definicji matematycznej.

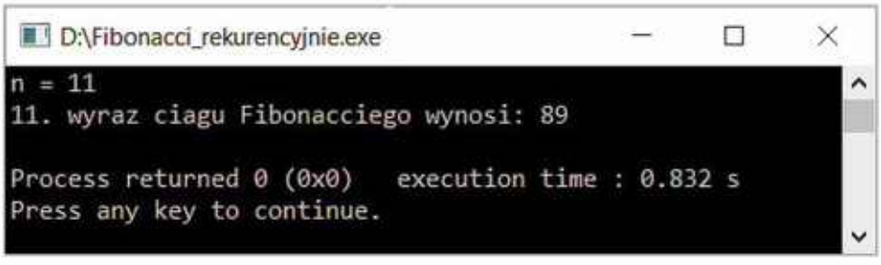
Oto kod źródłowy funkcji FibRek w programie *Fibonacci rekurencyjnie*:

Kod źródłowy funkcji FibRek w programie *Fibonacci rekurencyjnie*

```
1.  int FibRek(int n)
2.  {
3.      if (n == 0) return 0;
4.      if (n == 1) return 1;
5.      return FibRek(n-1) + FibRek(n-2);
6.  }
```

Kod z linii 3 i 4 odpowiada za wyznaczenie wartości funkcji dla wartości n równych 0 lub 1. Jest to sprawdzenie **warunku początkowego** definicji ciągu. W linii 5 następuje podwójne (w dwóch krokach) wywołanie rekurencyjne funkcji FibRek dla parametrów: n–1 oraz n–2. Na rysunku 5.10 przedstawiono przykładowe wywołanie programu:

Warunek początkowy, s. 80



Rys. 5.10. Przykładowe wywołanie programu *Fibonacci_rekurencyjnie.exe*

Ćwiczenie 4

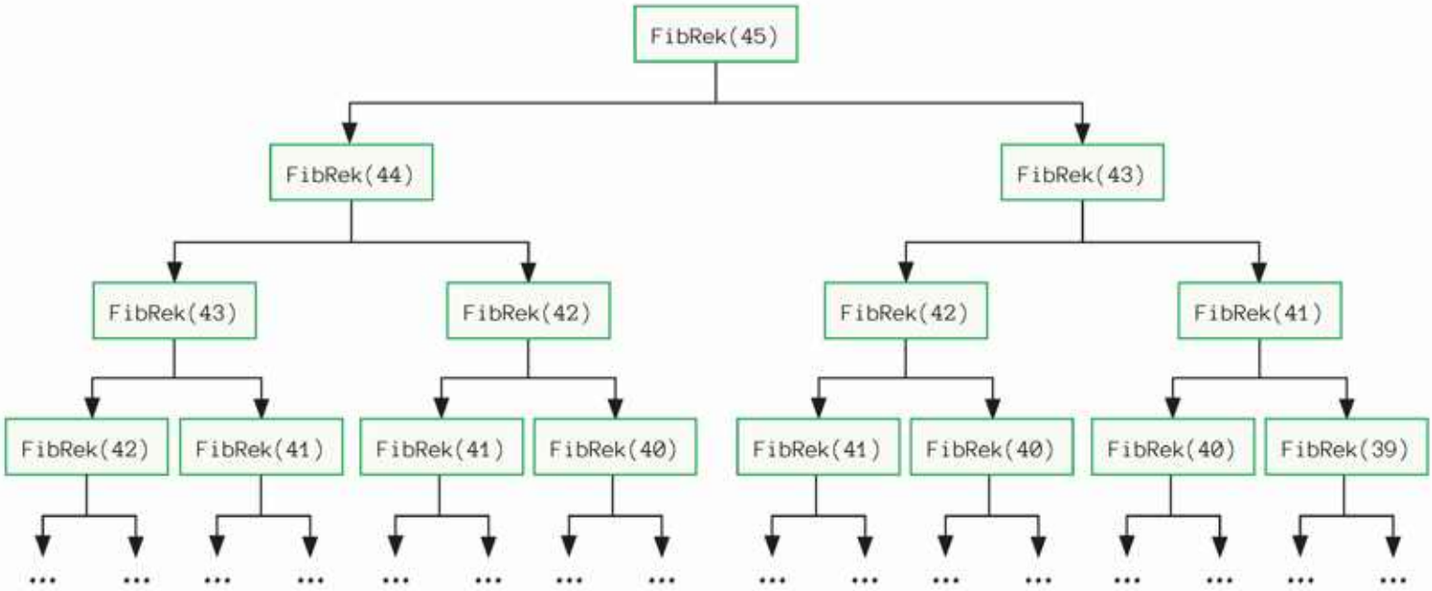
Napisz program według powyższej specyfikacji z wykorzystaniem funkcji FibRek. Sprawdź działanie programu dla liczb: 5, 10 i 45.

Okazuje się, że funkcja rekurencyjna FibRek, wyznaczająca kolejne liczby Fibonacciego, choć krótka w zapisie, może prowadzić do znaczącego marnowania zasobów komputera na wielokrotne wywoływanie funkcji rekurencyjnej z tymi samymi argumentami.

Na przykład już obliczenie 45. wyrazu ciągu Fibonacciego jest czasochłonne nawet dla współczesnych komputerów. Rysunek 5.11 przedstawia początek schematu wywołań rekurencyjnych funkcji `FibRek`. Zwróć uwagę na liczbę powtarzających się identycznych wywołań.

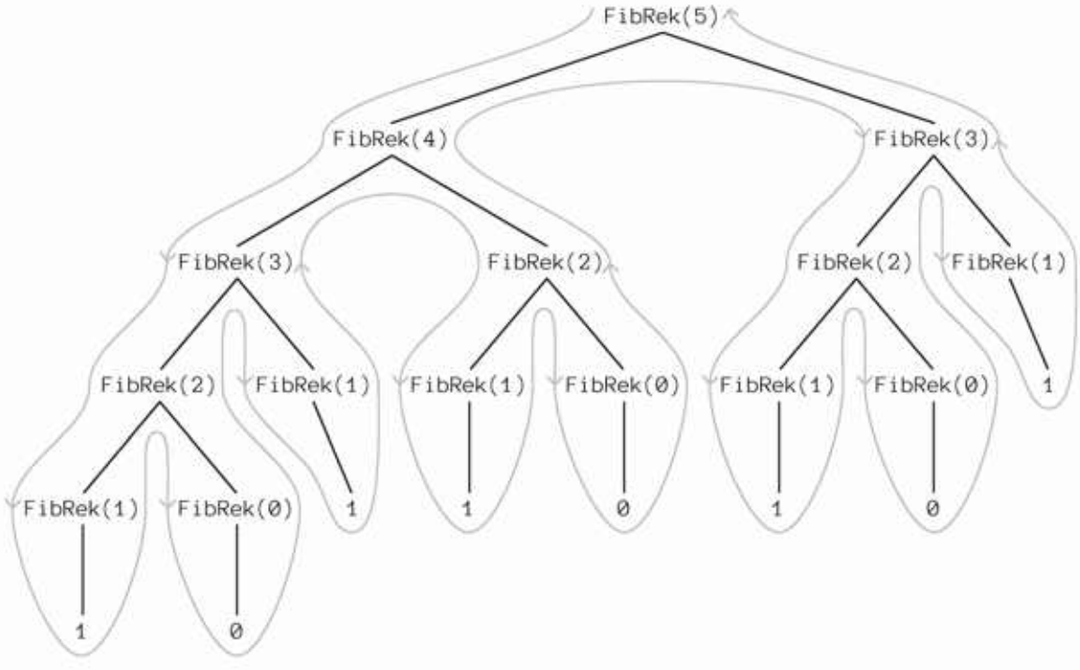
Warto wiedzieć

46. liczba Fibonacciego jest największą liczbą, którą da się zapisać, używając typu `int`.



Rys. 5.11. Schemat wywołań rekurencyjnych funkcji `FibRek` dla $n = 45$

Rysunek 5.12 przedstawia fazy: wywołań rekurencyjnych i zbierania wyników, które składają się na obliczenie wartości FibRek(5). Strzałki wskazują kolejność wywołań i powrotów rekurencyjnych.



Rys. 5.12. Drzewo wywołań rekurencyjnych funkcji `FibRek` dla $n = 5$

Powyższe obserwacje skłaniają do poszukiwania innego rozwiązania, które nie będzie wykorzystywało tak wielu zasobów komputera do wyznaczenia stosunkowo niskich wartości.

Warto wiedzieć

Gdyby w kodzie źródłowym funkcji `FibRek` zabrakło zapisu dotyczącego wartości funkcji dla `n = 0` oraz `n = 1`, to efektem byłaby rekurencja nieskończona. Prowadziłaby ona do zawieszenia się programu po wyczerpaniu zasobów pamięci komputera.

Program *Fibonacci iteracyjnie*

Zdecydowanie mniej pamięci komputerowej będzie potrzebował program, w którym liczby Fibonacciego będziemy obliczać iteracyjnie.

Zauważ, że w każdym momencie tworzenia nowych wyrazów ciągu wystarczy wiedza o dwóch ostatnich wyrazach ciągu. Zapamiętajmy w zmiennej a większą wartość, a w zmiennej b – mniejszą. 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.

W kolejnych cyklach pętli obliczymy wartość zmiennej pom, zmienna b przyjmie dotychczasową wartość zmiennej a, natomiast zmienna a – wartość zmiennej pom. Po n-krotnym zastosowaniu tych przekształceń b będzie równe f_n.

Oto kod źródłowy funkcji FibIter, która wyznacza n-ty wyraz ciągu Fibonacciego w sposób iteracyjny:

Kod źródłowy funkcji FibIter w programie *Fibonacci iteracyjnie*

```
1.  int FibIter(int n)
2.  {
3.      int pom;
4.      int a = 1;
5.      int b = 0;
6.
7.      for (int i=0; i<n; i++)
8.      {
9.          pom = a + b;
10.         b = a;
11.         a = pom;
12.     }
13.
14.     return b;
15. }
```

Warto wiedzieć

Każdy algorytm, który da się zapisać iteracyjnie, można zapisać również rekurencyjnie i na odwrót. Często zapis rekurencyjny algorytmu jest bardziej zwięzły.

Ćwiczenie 5

Napisz program według podanej wcześniej specyfikacji. Wykorzystaj funkcję FibIter. Sprawdź działanie programu dla liczb: 5, 10 i 45.

Zapamiętaj

Używanie rekurencji zwykle pozwala uzyskać bardziej zwięzły kod niż w przypadku iteracji. Czasami jednak ze względu na bardzo dużą liczbę wywołań rekurencyjnych rozwiązanie rekurencyjne jest bezużyteczne.

5.4. Rekurencyjny algorytm Euklidesa

Do tej pory poznaliśmy iteracyjną wersję **algorytmu Euklidesa**. Jego wersja z dzieleniem opiera się na równości:

$$NWD(a,b) = NWD(b \bmod a, a) \quad \text{dla } a \leqslant b$$

gdzie $b \bmod a$ oznacza resztę z dzielenia liczby b przez liczbę a.

Działanie algorytmu sprowadza się do naprzemiennego wykonywania operacji wyznaczania reszty z dzielenia: $a \bmod b$ dla $a > b$ oraz $b \bmod a$ dla $b > a$, dopóki reszta z dzielenia nie jest równa 0.

Algorytm Euklidesa – wersja z dzieleniem, podręcznik *Informatyka na czasie 2. Zakres podstawowy*, s. 160

Ćwiczenie 6

Zapisz w tabeli przebieg algorytmu Euklidesa z dzieleniem dla liczb 192 i 42.

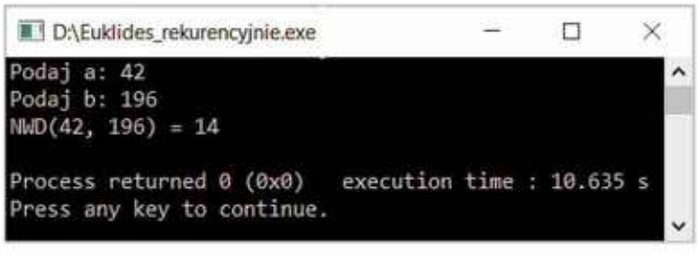
Algorytm wyznaczania największego wspólnego dzielnika dwóch liczb dodatnich a i b ($a \leqslant b$) możemy zapisać rekurencyjnie, opierając się na następującym spostrzeżeniu: jeśli b można podzielić bez reszty przez a (czyli gdy $b \bmod a = 0$), to wtedy $NWD(a,b) = a$. W przeciwnym wypadku $NWD(a,b) = NWD(b \bmod a, a)$.

Zapiszmy kod źródłowy funkcji `NWDRek`, realizującej rekurencyjny algorytm Euklidesa w wersji z dzieleniem:

Kod źródłowy funkcji `NWDRek` w programie *Euklides rekurencyjnie*

```
1.  int NWDRek(int a, int b)
2.  {
3.      if (b%a == 0) return a;
4.      return NWDRek(b%a, a);
5.  }
```

Oto wywołanie programu *Euklides rekurencyjnie*:



Rys. 5.13. Przykładowe wywołanie programu *Euklides_rekurencyjnie.exe*

Warto wiedzieć
Wywołanie rekurencyjne funkcji `NWDRek` z parametrami `b%a` oraz `a` można sobie wyobrazić następująco: procesor komputera działa na kopii kodu funkcji `NWDRek`, w którym zmiennym `a` i `b` są przypisane wartości parametrów, równe liczbowo odpowiednio `b%a` i `a`.

Ćwiczenie 7

a. W funkcji `NWDRek` wskaż warunek początkowy oraz instrukcję wywołania rekurencyjnego.
b. Zapisz kod programu, w którym użyjesz funkcji rekurencyjnej `NWDRek`. Skompiluj kod i uruchom program dla różnych a i b.
c. Przedstaw graficznie fazę wywołań rekurencyjnych i fazę zbierania wyniku.

Podsumowanie

- Rekurencja w informatyce to odwołanie się w rozwiązaniu problemu do tego samego problemu, ale dla mniejszego rozmiaru danych.
- Funkcja rekurencyjna musi zawierać dwa warianty kodu: dla warunku początkowego, gdy funkcja zwraca konkretną wartość, oraz dla wywołania rekurencyjnego, w którym złożony przypadek zostaje zredukowany do prostszego.
- Proces rekurencyjnego rozwiązywania problemu składa się z dwóch faz: redukowania rozmiaru problemu (aż do takiego, który można rozwiązać bezpośrednio) oraz zbierania i łączenia wyników częściowych w rozwiązanie wyjściowego problemu.
- Bardzo duża liczba wywołań rekurencyjnych może prowadzić do niewydajnego działania programu. Jeśli dostępna pamięć przydzielona programowi przez system operacyjny zostanie przekroczona, program przestanie działać.

Zadania

* **1** Aby obliczyć kolejną potęgę liczby 2, wystarczy daną potęgę pomnożyć przez 2, tzn. jeśli chcemy obliczyć wartość 2^6, a wiemy, że $2^5 = 32$, to 2^6 obliczymy jako $2 \cdot 32 = 64$. Korzystając z tej obserwacji, napisz funkcję o nagłówku `int Potega2(int n)`, która będzie obliczać 2^n dla $n > 0$.

* **2** Napisz program obliczający sumę n kolejnych liczb nieparzystych, począwszy od 1. Przygotuj dwie wersje programu: iteracyjną i rekurencyjną.

* **3** Poszukaj informacji o obrazach holenderskiego malarza Pieta Mondriana (1872–1944), który prostokątne tło płótna dzielił rekurencyjnie na coraz mniejsze prostokąty. Jakie było kryterium podziału? Przygotuj szkic przykładowego obrazu.

* **4** Przygotuj prezentację, w której przedstawisz fraktale opisane przez polskiego matematyka Wacława Sierpińskiego: dywan Sierpińskiego oraz trójkąt Sierpińskiego. W dowolnym programie graficznym narysuj fraktale wybranego stopnia i omów etapy ich powstawania.

* **5** Znajdź i popraw błąd w kodzie źródłowym funkcji `Szczescie`, która sprawdza, czy cyfra 8 występuje w danej liczbie. Przetestuj poprawność działania programu.

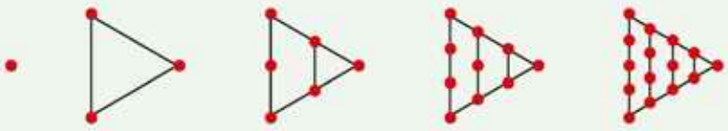
```
1. bool Szczescie(int liczba)
2. {
3.     int ostatnia = liczba % 10;
4.     if (ostatnia == 8)
5.         return true;
6.     return Szczescie(liczba / 10);
7. }
```

** **6** Na ile sposobów n osób ($n > 0$) może usiąść na n-osobowej ławce? Wzór określający liczbę osób zapisz w postaci rekurencyjnej.

** **7** Algorytm rekurencyjny można zastosować do obliczenia iloczynu liczby naturalnej n przez wszystkie liczby naturalne dodatnie mniejsze od tej liczby. Taką wartość nazywa się silnią liczby n i oznacza jako $n!$. Napisz program obliczający rekurencyjnie wartość $n!$.

** **8** 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.

** **9** 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 z użyciem rekurencji wyznaczy liczbę wszystkich punktów konstrukcji n-tej liczby trójkątnej.

*** **10** Zdefiniuj rekurencyjnie ciąg liczb n-cyfrowych ($n > 4$), w których żadna cyfra 8 nie sąsiaduje bezpośrednio z inną cyfrą 8.
Wskazówka: Ile jest takich liczb pięciocyfrowych, w których żadna cyfra 8 nie sąsiaduje bezpośrednio z inną cyfrą 8? Jak można tę wartość policzyć, gdy wiemy, ile jest takich liczb 4-cyfrowych i 3-cyfrowych?

*** **11** Liczba 145 ma taką własność, że jest równa sumie silni jej cyfr:
$1! + 4! + 5! = 1 + 24 + 120 = 145$
Oprócz liczb 1 i 2 istnieje jeszcze tylko jedna liczba o podobnej własności. Napisz program, który znajdzie tę liczbę.

*** **12** Przeanalizuj poniższy kod źródłowy funkcji obliczającej n-ty wyraz ciągu c_n.

```
1. int cn(int n, int iloczyn)
2. {
3.     if (n == 1) return iloczyn;
4.     return cn(n-1, 3*iloczyn);
5. }
```

Przedstaw przebieg działania algorytmu krok po kroku dla wywołania `cn(10, 3)`.

*** **13** Zdanie palindrom to ciąg znaków, który składa się z tych samych znaków czytany od lewej do prawej i wspak. Pomijamy przy tym spacje i znaki interpunkcyjne. Przykład zdania palindromu to: Ada raportuje, że jutro parada. Napisz funkcję rozstrzygającą, czy ciąg znaków będący jej argumentem jest palindromem. Funkcja powinna zwracać wartość logiczną.

Algorytmika i programowanie w języku C++

Wyobraź sobie, że...



...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 zaszyfrowanej wiadomości SMS wysłanej z bezpiecznego komunikatora.



...rozpoczynasz pracę w firmie produkującej gry planszowe, w dziale odpowiedzialnym za opakowania do tych gier. Już po kilku tygodniach twoją uwagę zwraca **problem optymalizacyjny** występujący w procesie produkcji. Duża część materiału przeznaczonego do wytwarzania pudełek jest marnowana i wyrzucana. Po rozmowie z kierownictwem, korzystając z **podejścia zachłannego**, opracowujesz szczegółowy raport, w którym proponujesz **optymalne rozwiązanie** problemu. Firma wprowadza zmiany w produkcji, co pozwala znacząco obniżyć koszty.



...ukończyłeś studia prawnicze i dostałeś się na staż w znanej kancelarii adwokackiej. Często musisz sięgać po różne akta, które nie są posegregowane. Wiesz, że w uporządkowanym zbiorze łatwiej **wyszukiwać** określone elementy. Przypominasz sobie o algorytmach **sortowania danych** i postanawiasz je wykorzystać. Sortujesz akta według dwóch kryteriów: najpierw chronologicznie, a potem alfabetycznie.



...postanowiłaś zwodowo 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**, o której usłyszałaś na lekcjach informatyki. Zaskakujesz prowadzących serią zdjęć z wykorzystaniem techniki **rekurencyjnej** i lustra.



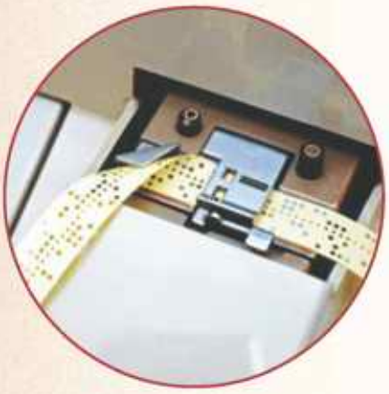
2 Algorytmika i programowanie w języku Python

6. Algorytmy na tekstach
7. Szyfrujemy wiadomości
8. Porządek ma znaczenie, czyli sortujemy liczby
9. Podejście zachłanne w rozwiązywaniu problemów
10. Rekurencja

Kartka z historii

Algorytmika i programowanie w języku Python

Sprzęt komputerowy jest użyteczny dzięki oprogramowaniu. Narzędziem do jego tworzenia są języki programowania, w których zapisuje się zaprojektowane wcześniej algorytmy, pozwalające efektywnie rozwiązywać różne problemy współczesnego świata.



1901 r.

Kamieniem milowym w rozwoju telekomunikacji był binarny **kod Baudota–Murraya**, ostatecznie ukształtowany ok. 1901 r. Dzięki niemu wysyłanie i odbieranie sygnałów elektrycznych zostało zautomatyzowane. W latach 60. XX w. zastąpił go kod ASCII.

1920 r.

W 1920 r. matematyk Jan Łukasiewicz zaproponował zapis wyrażeń arytmetycznych niewymagający nawiasów: znak operacji umieszcza się w nim przed argumentami. Jest to tzw. notacja Łukasiewicza lub **notacja polska**. Odwrócony zapis (znak operacji umieszczany za argumentami) znalazł zastosowanie w translatorach języków programowania.



1932 r.

Informacje wojskowe szyfrowane przez Niemców za pomocą Enigmy były niemożliwe do odczytania dla obcego wywiadu. **Łamanie szyfrów Enigmy** zapoczątkowali Polacy, już w grudniu 1932 r. Był to efekt wspólnej pracy wojskowych inżynierów i matematyków: Mariana Rejewskiego, Henryka Zygalskiego i Jerzego Różyckiego.



1938 r.



W 1938 r. Claude E. Shannon, amerykański matematyk i inżynier, opracował praktyczne zastosowanie dla teorii G. Boole'a, która wprowadzała wyrażenia logiczne w formie matematycznej. Wykazał, że operacje logiczne i arytmetyczne można wykonywać za pomocą **przełączników**, które otwierają lub zamykają obwód elektryczny.

1944 r.



Około 1944 r. zespół konstruktorów komputera ENIAC zapoczątkował posługiwanie się słowem „**programować**" (ang. *to program*) w dzisiejszym znaczeniu. Wcześniej używano słowa „konfigurować" (ang. *to set up*), ponieważ pierwsze komputery były programowane przez modyfikację sprzętową.

1951 r.

Grace Hopper, doktor matematyki i programistka komputerów elektromechanicznych Mark I i Mark II, tworzyła podprogramy, które można było wielokrotnie wywołać w programie głównym. W 1951 r. Hopper opracowała **translator** (nazwała go A-0), czyli program tłumaczący kody mnemotechniczne (np. ADD na oznaczenie dodawania) na binarny kod maszynowy komputera.



W 1954 r. zespół badawczy firmy IBM pod kierownictwem Johna Backusa zaprojektował język programowania **FORTRAN** (od ang. *FORmula TRANslation*), przeznaczony do obliczeń naukowo-inżynierskich. Jest on rozwijany i stosowany do dzisiaj, zwłaszcza w programowaniu superkomputerów używanych do przeprowadzania symulacji (np. modelowania reakcji chemicznych).

1954 r.

1962 r.



W 1962 r. angielski informatyk Charles A.R. Hoare przedstawił nowy sposób sortowania, dziś znany pod nazwą **QuickSort**. Wymyślił go podczas stażu naukowego w Moskwie, pracując na koncepcją maszynowego przekładu tekstu z języka rosyjskiego na angielski.

W 1965 r. Thomas E. Kurtz i John Kemeny z Dartmouth College opracowali język programowania **BASIC**. W latach 80. XX w. język ten był bardzo często używany do nauki programowania i pisania prostych gier na popularnych w Polsce komputerach, takich jak Commodore C64 i Amiga. Na składni jednego z dialektów tego języka opiera się język Visual Basic używany współcześnie.

```
10 LET A=112
20 LET B=88
30 LET R=A-B*INT (A/B)
40 IF R=0 THEN GOTO 80
50 LET M=N
60 LET N=R
70 GOTO 30
80 PRINT N_
```

1965 r.

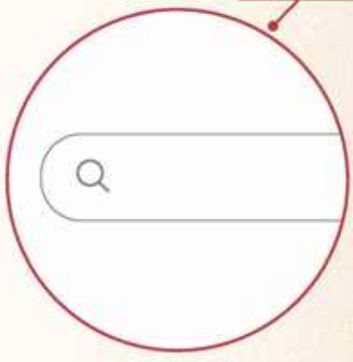
1983 r.

Język **C++**, opracowany przez Bjarne Stroustrupa, powstał w 1983 r. jako rozszerzenie języka C. Obecnie C++ jest jednym z najważniejszych narzędzi twórców oprogramowania użytkowego oraz gier komputerowych. W tym języku tworzy się oprogramowanie systemów wbudowanych, np. sterujących pracą urządzeń AGD.

Język programowania **Python** stworzył w 1991 r. holenderski programista Guido van Rossum. W zamierzeniu autora miał to być tzw. język skryptowy, który umożliwia szybkie przetwarzanie plików tekstowych i wykonywanie powtarzalnych zadań (np. na serwerze stron internetowych). Dziś Python jest jednym z najpopularniejszych języków programowania i ma różnorodne zastosowania. Na przykład podczas korzystania z serwisu YouTube lub narzędzi Google uruchamiamy na serwerach oprogramowanie napisane w języku Python.

1991 r.

2018 r.



Od 2018 r. program wyszukujący Google stosuje **algorytm BERT**. Pozwala on na lepszą analizę języka naturalnego, w której uwzględnia się m.in. końcówki fleksyjne i tzw. *stop words* (np. zaimki, przyimki, spójniki), czyli słowa o mniej istotnym znaczeniu. Algorytm BERT jest praktycznym przykładem zastosowania teorii sieci neuronowych.

6. Algorytmy na tekstach

Błędy językowe, literówki i inne pomyłki w tekście zwykle świadczą o niewiedzy lub nieuwadze piszącego. Czasami jednak mogą mieć naprawdę poważne konsekwencje. Na przykład skład leku i jego dawkowanie powinny być zawsze szczegółowo i precyzyjnie opisane w ulotce. W takim wypadku 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.

Cele lekcji

- Zrozumiesz, jak komputer zapisuje informacje tekstowe.
- Poznasz i zastosujesz algorytmy przetwarzania tekstów: porównywania tekstów oraz wyszukiwania wzorca.
- Dowiesz się, jak przetwarzać teksty w języku Python.
- Nauczysz się używać typu tekstowego `str` w języku Python.
- Poznasz funkcje `chr`, `len`, `ord` oraz metodę `find` dla klasy `str`.

6.1. Litery i inne znaki jako liczby

Emoji

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.

Jeśli umieścisz na portalu społecznościowym emoji 🌻, to widzisz je dlatego, że piktogram ten znajduje się w zbiorze znaków czcionki na twoim urządzeniu. Emoji 🌻 jest w takim zbiorze na pozycji o numerze

Kod liczbowy znaku

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 specjalnej

Tablica Unicode

tablicy znaków Unicode. W roku 2020 liczba znaków zakodowanych w tablicy Unicode przekraczała 280 000.

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).

Tabela 6.1 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 6.1. Fragment tablicy znaków Unicode zawierający wybrane piktogramy

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, w tym Unicode, są w pełni kompatybilne z ASCII.

Tablica ASCII

Typ danych `str` w języku Python

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**.

Typ `str`

Napis (łańcuch znaków, łańcuch)

Łańcuchy znaków przechowuje się w zmiennej typu `str`, czyli w strukturze danych, w której zapisane są kody Unicode kolejnych znaków łańcucha. W tabeli 6.2 przedstawiono fragment tablicy Unicode ze znakami ASCII, które można wyświetlić na ekranie.

	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 6.2. Znaki ASCII – fragment tablicy Unicode

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 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.

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.

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.

Program *Znaki*. Funkcja chr

Napiszemy teraz program, który wyświetli fragment tablicy Unicode pokazany w tabeli 6.2 na s. 97. 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 wykorzystamy wbudowaną w interpreter **funkcję chr** (od ang. *character* – znak), która dla podanej liczby (pozycji w tablicy Unicode) zwraca odpowiadający jej znak (jednoznakowy napis typu `str`).

Pętla for, s. 255

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.

Na rysunku 6.1 przedstawiono wywołanie programu *Znaki*.



Rys. 6.1. Wywołanie programu *Znaki*

Dobra rada

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

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 2

Zapisz kod źródłowy programu w pliku *znaki.py* i 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.

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 *Znaki* ma wyświetlić znaki znajdujące się w tablicy Unicode między spacją a znakiem ~ (patrz rys. 6.1), 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 wierszu 1 zmiennej `znak` przypisujemy znak spacji. W wierszach 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. 98

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. 255

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

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
```

6.2. Szukamy 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!".

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)
```

Dobra rada

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

Stała

Ponieważ oba komunikaty ze specyfikacji są dla użytkownika niezmienne, możemy je traktować jako **stałe**, czyli wartości, które nie mogą się zmienić w trakcie wykonywania programu. Dobrą praktyką jest zapisywanie nazw stałych wielkimi literami (tak jak w liniach 1 i 2). Dzięki temu nie będą się one mylić z nazwami zmiennych.

W wierszach 4–5 i 7–8 znajduje się kod określający nazwy zmiennych, w których będą zapamiętane dane wpisane z klawiatury przez użytkownika.

W **warunku logicznym** (wiersz 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).

Logika matematyczna i operatory logiczne, s. 242–244

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

- **Operator** `+`
- **Operacja złączenia (konkatenacja)**

Rysunek 6.2 przedstawia przykładowe wywołania programu dla dwóch sytuacji: poprawnego i niepoprawnego podania adresów e-mail.

```
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. 6.2. Wywołania programu *Porównanie adresów*

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*.

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

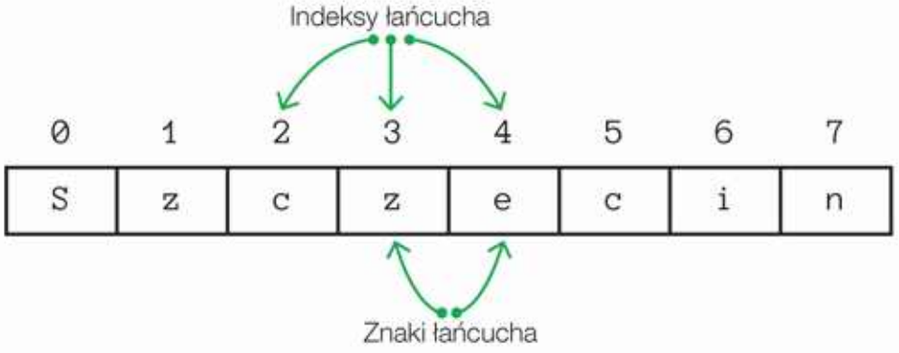


Jednym z rodzajów literówek jest przestawienie znaków w tekście, nazywane czeskim błędem. Zwykle zdarza się to osobom, które piszą na klawiaturze obiema rękami. Często klawisze z literami E oraz I wciska się w niewłaściwej kolejności, ponieważ znajdują się po przeciwnych stronach klawiatury. Powstaje 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 numerze $n - 1$, gdzie n to liczba znaków łańcucha (rys. 6.3). Taki sposób indeksacji stosuje się również w **listach**.

Listy w języku Python, s. 253



Rys. 6.3. 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]`.

Indeks ujemny

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.

Próba odwołania się do znaku na pozycji, której w łańcuchu nie ma, zakończy się wyświetleniem błędu. Dlatego warto stosować

Funkcja len

funkcję `len`, która zwraca długość łańcucha tekstu. Na przykład dla zmiennej `miasto = "Szczecin"` wartość `len(miasto)` wynosi 8.

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

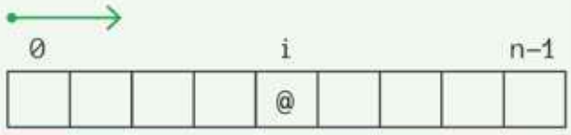
Domeny najwyższego rzędu zazwyczaj są dwu- lub trzyliterowe. Należą do nich m.in. domeny .pl, .us, .org, .com, .gov.

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.

Poprawność adresu e-mail możemy sprawdzić w czterech etapach. Opisuje je schemat z rysunku 6.4.

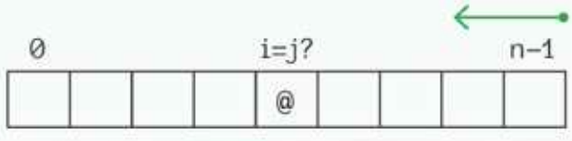
Etap 1

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.



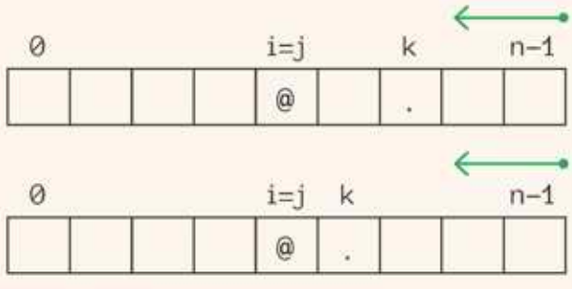
Etap 2

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.



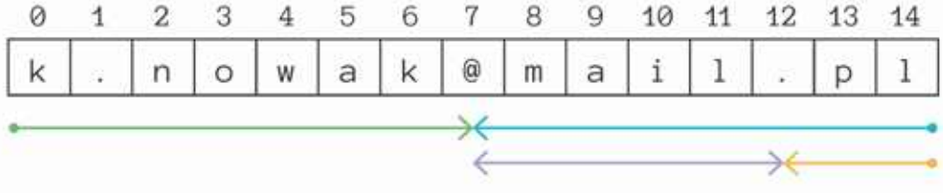
Etap 4

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. 6.4. Etapy sprawdzania poprawności podanego adresu e-mail

Przykład realizacji algorytmu pokazuje rysunek 6.5. Kolory strzałek są zgodne z kolorami tła poszczególnych etapów z rysunku 6.4.



Rys. 6.5. 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 adres[i] != "@" and i < dl-1:
6.          i = i + 1
7.      if adres[i] != "@" or i < 2:
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 adres[k] != "." and k > 0:
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 wierszu 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. 102

W etapie 1 (linie 4–8) 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 10–14. Jeżeli i różni się od j (warunek z linii 13), to w napisie występują co najmniej dwa znaki @.

Wiersze 16–20 dotyczą etapu 3, w którym szukamy znaku kropki i określamy jego położenie. Zauważ, że 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. 242–244

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

Oto fragment kodu źródłowego zawierający wywołanie funkcji CzyPoprawnyAdres:

Wywołanie funkcji CzyPoprawnyAdres w programie *Sprawdzenie adresu*

```
1.  imie = input("Podaj imię: ")
2.  nazwisko = input("Podaj nazwisko: ")
3.  adres_1 = input("Podaj adres e-mail: ")
4.
5.  if CzyPoprawnyAdres(adres_1):
6.      adres_2 = input("Powtórz adres e-mail: ")
7.      print()
8.      if adres_1 == adres_2:
9.          print(INF_OK + adres_1)
10.     else:
11.         print(INF_ROZNE_ADRESY)
12. else:
13.     print(INF_BLAD_ADRESU)
```

Warto wiedzieć

Wartość `True` jest reprezentowana w pamięci jako liczba 1, dlatego zapis w linii 5 może być również następujący:

```
if CzyPoprawnyAdres
   (adres_1) == 1:
```

Zauważ, że funkcja `CzyPoprawnyAdres` zwraca wartość **typu logicznego**, a jej wywołanie znajduje się w miejscu wyrażenia logicznego instrukcji warunkowej (linia 5). W ten sposób funkcję można wykorzystać jako warunek logiczny np. w instrukcji warunkowej lub pętli.

Typy danych w języku Python, s. 252–253

Ćwiczenie 6

a. Zapisz specyfikację do programu *Sprawdzenie adresu*. Zaproponuj brzmienie wykorzystywanych w nim komunikatów.

b. Zapisz pełny kod źródłowy programu *Sprawdzenie adresu* i uruchom program. Sprawdź jego działanie dla różnych danych.

A to ciekawe

Jak odróżnić człowieka od maszyny?

W wielu serwisach internetowych stosuje się rozwiązanie nazywane CAPTCHA, którego zadaniem jest upewnienie się, że formularz uzupełnia człowiek, a nie spamujący robot. Test polega zazwyczaj na przepisaniu słowa z obrazka. Fakt, że czynność tę wykonują regularnie miliony internautów, wykorzystano w projekcie reCAPTCHA. Użytkownikom wyświetla się fragmenty tekstu nieczytelne dla oprogramowania OCR (zamieniającego skan na tekst) podczas digitalizacji zasobów na użytek bibliotek cyfrowych. 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 tzw. językiem obiektowym. Utworzenie w nim zmiennej typu `str`, np. `imie = "Dorota"`, oznacza właściwie 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

Każda klasa w języku Python, również klasa `str`, oferuje opracowane specjalnie dla niej funkcje, nazywane **metodami** (ang. *methods*). Metody są ściśle 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. 259

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

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
```

Warto wiedzieć

Zapis nazwy metody po kropce jest właściwy dla tzw. programowania obiektowego.

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.

6.3. 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 maksymalnie 20 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 6.3. Analiza działania algorytmu usuwającego powtórzenia

Tabelę 6.3 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 7

Przeanalizuj działanie algorytmu dla listy zakupów składającej się z ośmiu wyrazów, wśród których są dwa powtórzenia, a jedno pojawia się na końcu.

Oto fragment kodu źródłowego programu *Usuwanie powtórzeń*:

Kod źródłowy 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 j in range(i):
19.     print(wynik[j])
```

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.

Dobra rada

Wyświetlanie wszystkich elementów listy `wynik` możesz zapisać prościej, np.:

```
for j in wynik:
    print(j)
```

W linii 1 zdefiniowano stałą `N` i przypisano jej wartość 20. W linii 3 dodano instrukcję tworzącą pustą listę o nazwie `wynik`. Docelowo na tej liście będzie przechowywana lista produktów. Zmienna `i` zdefiniowana w linii 4 będzie licznikiem.

W wierszach 5–6 wczytujemy pierwszy napis i umieszczamy go na liście `wynik`. Wykorzystujemy do tego celu **metodę** `append`, która powiększa rozmiar listy i umieszcza na jej końcu wartość.

Metoda append

W wierszach 8–9 wczytujemy drugi napis. W pętli **`while`** 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 wierszach 12–16 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.

6.4. 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

Wykreślanka

Na rysunku 6.6 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.

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.

R	O	M	A	N	A	L	I	C	J	A	Z	A	P	M	G	P	R
T	E	L	H	C	E	I	C	J	O	W	S	I	A	I	S	A	K
O	A	I	G	O	Z	U	Z	A	N	N	A	L	W	C	O	W	A
M	A	R	I	A	J	U	L	I	A	I	E	E	A	H	N	E	R
E	W	D	O	M	I	N	I	K	T	N	N	M	W	A	I	Ł	O
K	B	O	G	U	M	I	Ł	O	A	E	Ł	A	E	Ł	A	I	L

Rys. 6.6. 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.

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).

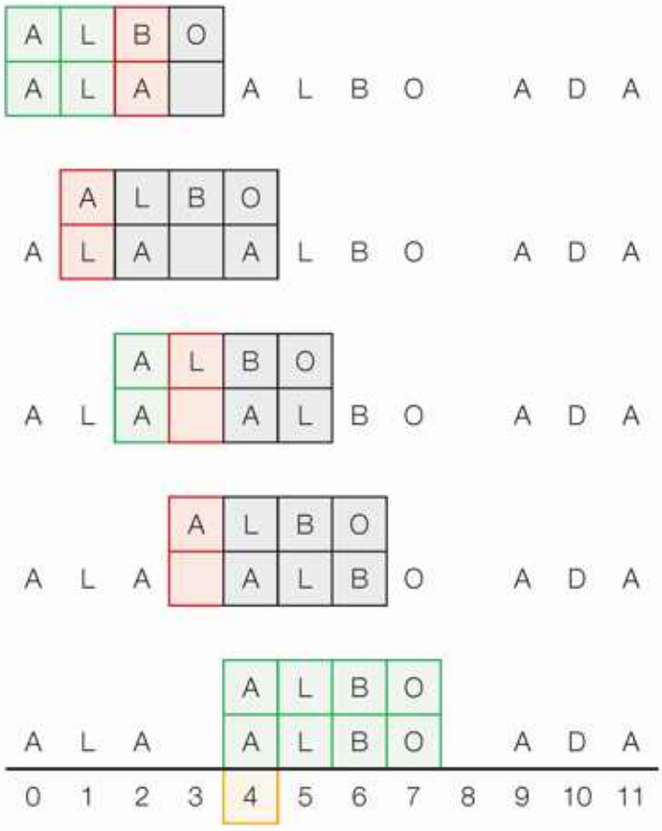
Algorytm wyszukiwania wzorca w tekście

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.

Warto wiedzieć

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

W tekście ALA ALBO ADA poszukamy wzorca ALBO. Rysunek 6.7 pokazuje działanie algorytmu. Zwracana jest wartość 4.



Rys. 6.7. Naiwne wyszukiwanie wzorca w tekście

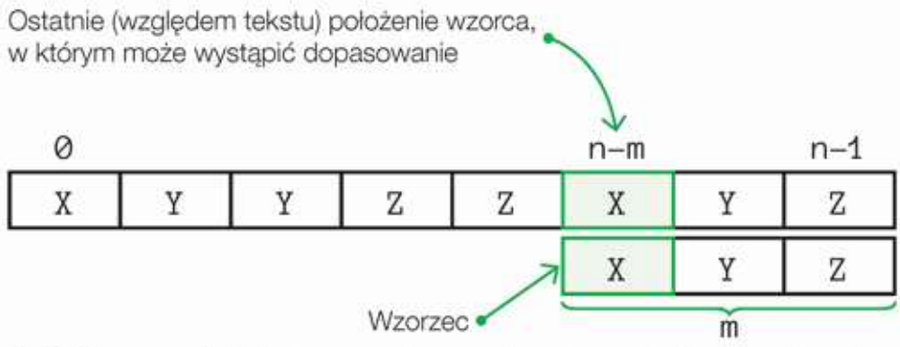
Ćwiczenie 10

a. Zapisz przebieg działania algorytmu poszukiwania w tekście ZA SZCZYTEM ZASZCZYT wzorca ZASZCZYT. Zapisz go w podobny sposób jak na rysunku 6.7.

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 numerze $n - m$, należy sprawdzać zgodność kolejnych znaków tekstu ze znakami tworzącymi wzorzec.

Dlaczego sprawdzamy do znaku o numerze $n - m$? Wyjaśnia to rysunek 6.8. Jeśli tekst ma długość $n = 9$, a wzorzec $m = 3$, to ostatni indeks ma wartość 6 (znaki indeksujemy od 0).



Rys. 6.8. Wyznaczenie indeksu ostatniego znaku tekstu, z którym porównujemy wzorzec

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 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 wykonuj kolejno kroki 4 i 5.
   4. Wybierz fragment tekstu długości m, począwszy od znaku na pozycji równej licznikowi pętli, i porównaj go ze wzorcem.
   5. Jeśli fragment i wzorzec są identyczne, to zwróć jako odpowiedź liczbę równą licznikowi pętli i zakończ algorytm.
6. Zwróć wartość -1.

Zauważ, że w powyższym algorytmie licznik pętli, w momencie znalezienia wzorca w tekście, wskazuje jednocześnie indeks pierwszego wystąpienia wzorca. 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.

Stała, s. 100

W linii 1 zastosowano **stałą** o nazwie TEKST. Zmienne `n` i `m` pamiętają 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`), 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).

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.

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`.

Rysunek 6.9 przedstawia przykładowe wywołania programu.

```
Wykonywanie: szukaj wzorca.py
Tekst: ALA ALBO ADA
Podaj wzorzec: AL
Pozycja: 0
Python
```

```
Wykonywanie: szukaj wzorca.py
Tekst: ALA ALBO ADA
Podaj wzorzec: ALBO
Pozycja: 4
Python
```

Rys. 6.9. Wywołania programu *Szukaj wzorca*

Ćwiczenie 11

a. W pliku otrzymanym od nauczyciela (np. *wzorzec_w_tekscie.py*) znajduje się fragment programu *Szukaj wzorca*. Uzupełnij go o odpowiednie instrukcje i uruchom program dla wzorców: AGA oraz ALBO.
b. Sprawdź działanie programu dla własnego tekstu i wzorca.

Warto wiedzieć

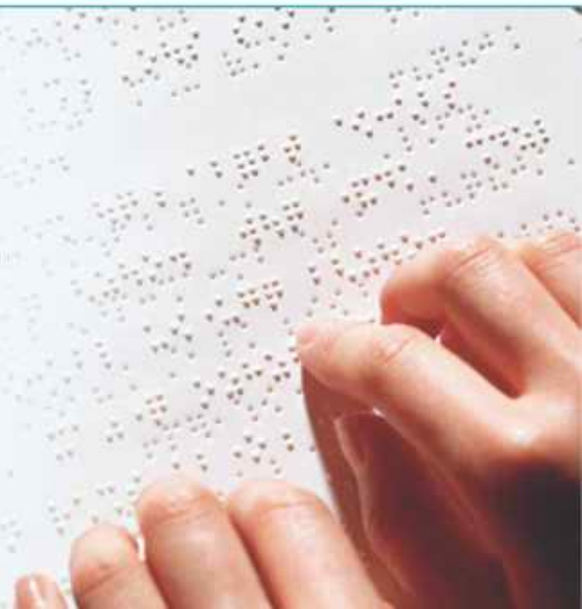
Efektywna metoda wyszukania wzorca w tekście jest zaimplementowana w języku Python jako metoda `find`. Używaliśmy jej w programie sprawdzającym poprawność adresu e-mail.

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.

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.



Podsumowanie

- W pamięci komputera znaki tekstu, jak każdy inny rodzaj informacji, są zapisywane jako liczby. Liczby odpowiadające konkretnym znakom (literom, cyfrom, znakom interpunkcyjnym i innym symbolom) są określone przez ich położenie w tablicy Unicode.
- Klasa `str` zawiera przydatne metody dotyczące napisów. Na przykład metoda `find` zwraca indeks pierwszego od lewej wystąpienia szukanego wzorca w danym łańcuchu, a metoda `rfind` – indeks pierwszego wystąpienia wzorca od końca łańcucha.
- Funkcja `len` zwraca długość napisu.
- Metoda `append` służy do dodawania na końcu listy nowego elementu. Aby dodać do listy `imiona` element `"Agnieszka"`, należy zapisać: `imiona.append("Agnieszka")`.
- Dostęp do znaku w napisie uzyskuje się za pomocą nawiasów kwadratowych. Znaki numerowane są od 0. Pierwszy znak łańcucha znaków o nazwie `tekst` to `tekst[0]`, a ostatni to `tekst[-1]`.
- Łańcuchy znaków (lub łańcuch i pojedynczy znak) można scalać, używając operatora +.

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 10 imion osób z twojej klasy i wypisze je w postaci tzw. trójkąta Floyda. Oznacza to, że w pierwszym wierszu będzie jedno imię, a w kolejnych odpowiednio dwa, trzy i cztery.

```
1
2 3
4 5 6
7 8 9 10
```

* **4** 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.

* **5** Napisz program, który wczyta wyraz wzorzec, a następnie będzie wczytywać kolejne wyrazy aż do pojawienia się ciągu znaków `***`. Następnie program powinien wyświetlić liczbę powtórzeń wzorca pośród wczytanych wyrazów.

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

* **7** 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.

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

** **9** Taumatawhakatangihangakoauauotamateaturipukakapikimaungahoronukupokaiwhenuakitanatahu to nazwa jednego ze wzgórz w Nowej Zelandii i jedna z najdłuższych nazw geograficznych na świecie. Korzystając z pliku, który przekaże ci nauczyciel (np. *wzgorze.txt*), sprawdź:

a. długość tej nazwy,
b. liczbę występujących w niej liter: a, e, i, o, u, y,
c. czy w nazwie występuje fraza „eaturip".

** **10** 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.

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

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

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

** **14** 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`.

** **15** Pod kodem $7F_{16}$ (binarnie 0111 1111) w tablicy ASCII kryje się kod znaku sterującego DEL (od ang. *delete*, czyli usunąć). Poszukaj wiarygodnej informacji wyjaśniającej przyczynę umieszczenia kodu tego znaku w tym miejscu. Jaki miało to związek z zapisem na taśmie dziurkowanej używanej w drukarkach dalekopisów? Przygotuj krótką prezentację na temat ewolucji kodów stosowanych w telegrafii i teleksach. Uwzględnij kody Baudota–Murraya, ITA2 oraz ASCII.

** **16** 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.

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

*** **18** Napisz program, który podane przez użytkownika zdanie (w języku angielskim) zapisze w kodzie Morse'a. Oprócz liter program powinien też kodować cyfry i kropkę.

7. Szyfrujemy 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.

Cele lekcji

- Zrozumiesz podstawowe pojęcia związane z kryptografią.
- Dowiesz się, po co szyfruje się informacje i jakie są zastosowania kryptografii.
- Poznasz przykłady szyfrów podstawieniowych i przestawieniowych, w tym szyfr Cezara i szyfr kolumnowy.
- Napiszesz programy szyfrujące informacje tekstowe poznanymi metodami.
- Poznasz techniki łamania prostych szyfrów.

7.1. 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

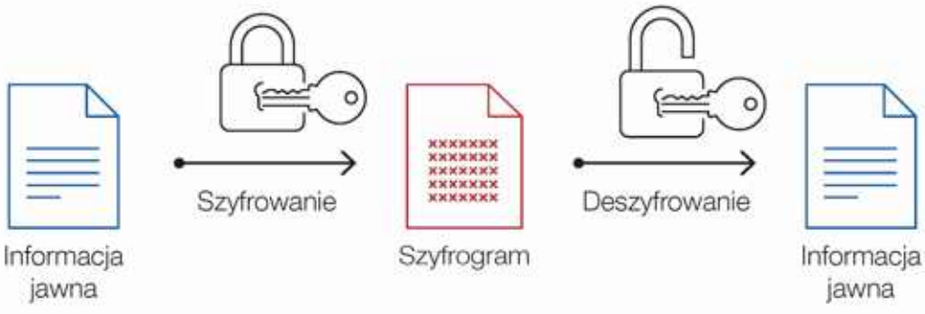
W trakcie szyfrowania informację jawną, zwaną **tekstem jawnym** (ang. *plaintext*), przekształca się w **szyfrogram** (ang. *ciphertext*), czyli tekst zaszyfrowany (rys. 7.1). W tym procesie stosuje się odpowiedni algorytm oraz **klucz szyfrowania** (ang. *cipher key*). Jest to pewna tajna wartość, niezależna od szyfrowanej informacji, która umożliwia **deszyfrowanie** zaszyfrowanej wiadomości. Warunkiem tajności informacji jest utrzymanie w tajemnicy klucza szyfrowania.

Tekst jawny

Szyfrogram

Klucz szyfrowania

Deszyfrowanie



Rys. 7.1. Schemat szyfrowania i deszyfrowania informacji

7.2. Szyfry przestawieniowe. Szyfr kolumnowy

Pionierami w rozwoju klasycznej kryptografii byli prawdopodobnie Spartanie. Grecki poeta Archiloch w jednym z dzieł wspomniał o przyrządzie do szyfrowania zwanym **skytale** (od gr. *skútalon* – długi, gruby kij), używanym w Sparcie od VII w. p.n.e.

Skytale

Zasada korzystania ze skytale była następująca: nadawca wiadomości nawijał na pałkę pasek pergaminu (lub skóry) i pisał tekst na stykających się fragmentach. Po rozwinięciu paska wiadomość stawała się nieczytelna. Adresat posiadał identyczną pałkę, umożliwiającą odczytanie wiadomości. W tej metodzie szyfrowania **kluczem** są grubość i kształt wałka.

Klucz szyfrowania, s. 115

Poniżej zilustrowano sposób odczytywania szyfrogramu na skytale o kształcie graniastosłupa prawidłowego sześciokątnego.

Odczytywanie wiadomości zaszyfrowanej na skytale

Po otrzymaniu szyfrogramu przekazanego na pasku pergaminu należy nawinąć go na wałek o ustalonej wcześniej grubości. Grubość wałka jest kluczem tego szyfru.

Szyfrogram:
TETSKAEKOTSWNSJTTNTTEEJY

Tekst jawny:
TENTEKSTTOJESTTEKSTJAWNY



Dobra rada

Wiele ołówków ma kształt graniastosłupa prawidłowego sześciokątnego. Możesz więc wykorzystać ołówek jako skytale.

Warto wiedzieć

Auguste Kerckhoffs oraz Claude Shannon niezależnie od siebie sformułowali tezę zwaną dziś zasadą Kerckhoffsa. Mówi ona, że zapewnienie poufności informacji nie powinno zależeć od utrzymania metody szyfrowania w tajemnicy – tajny powinien być wyłącznie klucz. Dzięki upublicznieniu szczegółów algorytmów szyfrujących można wykryć luki bezpieczeństwa, zanim algorytm zostanie wdrożony.

Załóżmy, że niepowołana osoba przechwyciła zaszyfrowaną wiadomość oraz znała tę metodę szyfrowania. Jeśli nie zna dokładnych wymiarów wałka, trudno jej będzie odczytać informację, ponieważ musiałaby sprawdzić wiele wałków o różnych grubościach.

Ćwiczenie 1

Przygotuj model urządzenia skytale i pergaminu: użyj ołówka i wąskiego paska papieru o długości ok. 25–30 cm. Wykorzystaj zbudowany model skytale do zaszyfrowania krótkiej wiadomości tekstowej.

Technikę szyfrowania, z której korzysta się w metodzie z użyciem skytale, nazywa się współcześnie **szyfrem przestawieniowym**. Szyfr ten polega na przestawieniu znaków z informacji jawnej. Szyfrogram i informacja jawna składają się więc z tych samych znaków, ale ustawionych w innej kolejności.

Szyfr przestawieniowy

Szyfr kolumnowy

Innym szyfrem przestawieniowym, podobnym do metody z użyciem skytale, jest **szyfr kolumnowy**. Polega on na zapisaniu tekstu jawnego w wierszach, przy czym każdy wiersz (czasem z wyjątkiem ostatniego) składa się z takiej samej liczby znaków odpowiadającej liczbie kolumn. Kluczem szyfrowania jest liczba kolumn. Jeśli ostatni wiersz jest niepełny, można go uzupełnić dowolnymi znakami.

Szyfr kolumnowy

Dla tekstu jawnego PRZYNIESPRZESYLKEDOPARKU i klucza 4 zapis w kolumnach wygląda tak jak na rysunku 7.2.

P	R	Z	Y
N	I	E	S
P	R	Z	E
S	Y	L	K
E	D	O	P
A	R	K	U

Rys. 7.2. Podział tekstu jawnego na kolumny zgodnie z kluczem szyfrowania

Jeśli odczytamy znaki tekstu kolumnami (z góry na dół), otrzymamy tekst zaszyfrowany: PNPSEARIRYDRZEZLOKYSEKPU.

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.

Ćwiczenie 2

Zaszyfruj tekst PRZYNIESPRZESYLKEDOPARKU, używając klucza o wartości 6.

A to ciekawe

Anagram jako szyfrogram

Niektórzy uczeni nie chcieli ogłaszać wprost treści swoich odkryć, dlatego stosowali anagramy, czyli zmieniali kolejność liter w zdaniu. 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.



Szyfr kolumnowy – program szyfrujący

Napiszemy teraz program, który pozwoli zaszyfrować wpisany z klawiatury tekst szyfrem kolumnowym z podanym kluczem.

Oto specyfikacja problemu:

Specyfikacja

Dane: tekst jawny o długości n wprowadzony z klawiatury bez spacji; klucz – liczba całkowita k, spełniająca warunek $2 \leqslant k \leqslant \frac{n}{2}$.

Wynik: tekst zaszyfrowany metodą kolumnową o liczbie kolumn równej k.

Aby zaprogramować szyfrowanie tekstu metodą kolumnową, nie trzeba pamiętać układu kolumn, wystarczy dopisywać do szyfrogramu znaki w odpowiedniej kolejności. Prześledźmy przykład szyfrowania tekstu PRZYNIESPRZESYLKEDOPARKU z kluczem 4.

Ponumerujmy znaki tekstu jawnego, począwszy od 0, czyli tak jak numerowane są kolejne znaki w **łańcuchach znaków** (rys. 7.3):

Łańcuchy znaków, s. 97

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
P	R	Z	Y	N	I	E	S	P	R	Z	E	S	Y	L	K	E	D	O	P	A	R	K	U

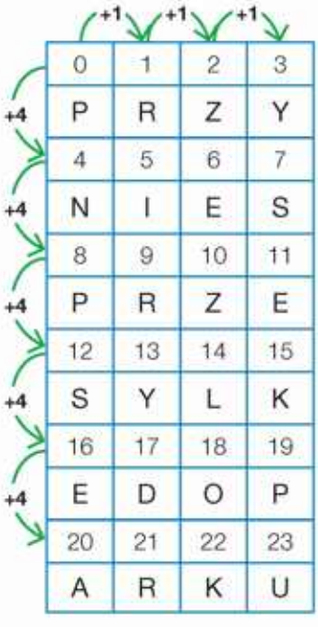
Rys. 7.3. Tekst jawny z ponumerowanymi znakami

Ustawmy teraz tekst w czterech kolumnach, ponieważ klucz to 4 (rys. 7.4). Łatwo obliczyć, że wierszy musi być 6. Spójrzmy na numery znaków w pierwszej kolumnie: 0, 4, 8, 12, 16, 20 oraz numery znaków w drugiej kolumnie: 1, 5, 9, 13, 17, 21. Kolejne znaki w kolumnie mają numery różniące się o 4, czyli o wartość klucza (liczbę kolumn).

Warto wiedzieć

Szyfrem podobnym do kolumnowego jest szyfr płotkowy, nazywany również metodą płotu. Kolejne litery tekstu jawnego umieszcza się w tabeli schodkowo, przyjmując pewną stałą wysokość schodków. Na przykład tekst PUŁAPKA zaszyfrowany metodą płotu z kluczem 4 ma postać: PAUKŁPA.

P						A
	U				K	
		Ł		P		
			A			



Rys. 7.4. Numeracja znaków po podziale na kolumny

Lista kroków algorytmu szyfrowania kolumnowego może wyglądać następująco:

1. Wczytaj tekst jawny.
2. Wczytaj klucz szyfrowania i oznacz tę liczbę jako klucz.
3. Oznacz długość tekstu jawnego jako dl.
4. Dla każdej liczby i z zakresu od 0 do klucz - 1, traktowanej jako kolejna wartość licznika pętli, wykonuj krok 5.
   5. Dołącz do tekstu szyfrogramu następujące litery tekstu jawnego: tę na pozycji i oraz wszystkie te, których pozycja jest liczbą i powiększoną o wielokrotność klucza.

Zapiszmy kod źródłowy programu *Szyfr kolumnowy*:

Kod źródłowy programu *Szyfr kolumnowy*

```
1.  szyfrogram = ""
2.
3.  jawny = input("Podaj tekst jawny (bez spacji): ")
4.  klucz = int(input("Podaj klucz szyfrowania: "))
5.
6.  dl = len(jawny)
7.
8.  for i in range(0, klucz):
9.      for j in range(i, dl, klucz):
10.         szyfrogram = szyfrogram + jawny[j]
11.
12. print("\nSzyfrogram:", szyfrogram)
```

W wierszu 1 zmiennej `szyfrogram` przypisujemy jako początkową wartość **łańcuch pusty** (ang. *empty string*), czyli łańcuch bez znaków (o długości 0). Łańcuch ten będziemy wydłużać o kolejne znaki szyfrogramu. W linii 6, korzystając z **funkcji** `len`, obliczamy długość tekstu jawnego.

Łańcuch pusty

Funkcja `len`, s. 102

Szyfrogram jest tworzony w liniach 8–10 z wykorzystaniem **pętli zagnieżdżonych**. W zewnętrznej pętli (linie 8–10) przeglądamy kolejne kolumny. Pętla wykona się więc tyle razy, ile jest kolumn (jest to wartość klucza). W wewnętrznej pętli (linie 9–10) przeglądamy kolejne znaki w danej kolumnie (zgodnie z rys. 7.4). Licznik pętli jest w każdym cyklu (przebiegu pętli) powiększany o wartość klucza, a liczba iteracji ograniczona jest z góry przez wartość zmiennej `dl`. Gwarantuje to poruszanie się tylko w obrębie liter tekstu jawnego.

Pętle zagnieżdżone, s. 112

W linii 10 napis aktualnie przechowywany w zmiennej `szyfrogram` powiększamy z prawej strony o kolejny znak z wykorzystaniem **operacji złączenia**, czyli konkatenacji.

Operacja złączenia, s. 101

Rysunek 7.5 na s. 120 przedstawia wywołanie programu dla tekstu jawnego PRZYNIESPRZESYLKEDOPARKU oraz klucza 4.

```
Wykonywanie: szyfr kolumnowy.py
Podaj tekst jawny (bez spacji): PRZYNIESPRZESYLKEDOPARKU
Podaj klucz szyfrowania: 4

Szyfrogram:  PNPSEARIRYDRZEZLOKYSEKPU
>>>
                                                    Python
```

Rys. 7.5. Wywołanie programu *Szyfr kolumnowy*

Ćwiczenie 3

a. Zapisz kod źródłowy programu *Szyfr kolumnowy* i sprawdź działanie programu dla tego samego tekstu, ale różnych kluczy – również takich, których wartości nie są dzielnikami długości (liczby znaków) tekstu jawnego.

b. Zmodyfikuj kod źródłowy programu *Szyfr kolumnowy* tak, aby wyświetlał on dodatkowo etap pośredni, tj. przedstawienie tekstu jawnego w układzie kolumnowym.

7.3. Szyfry podstawieniowe. Szyfr Cezara

Szyfr podstawieniowy

Jeszcze w czasach starożytnych wymyślono inny rodzaj szyfru – **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. 115

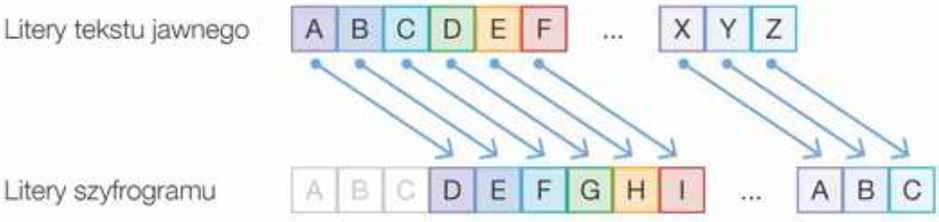
Metoda szyfrowania, której używał Cezar, polega na zastąpieniu każdej litery **tekstu jawnego** literą oddaloną od niej o stałą liczbę pozycji w alfabecie. Kierunek zmian musi być zachowany. Przesunięcie odbywa się cyklicznie, tzn. po ostatniej literze alfabetu następuje pierwsza.

Klucz szyfrowania, s. 115

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 7.6.

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.



Rys. 7.6. Szyfrowanie liter szyfrem Cezara z kluczem 3

Ćwiczenie 4

Wykorzystując szyfr Cezara i alfabet łaciński:

a. zaszyfruj wyraz MEDYCYNA z kluczem 25.

b. odszyfruj szyfrogram LWUQVQSIVI z kluczem 8.

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 7.7 przedstawiono przesunięcie równe 2, tzn. literze A odpowiada litera C, literze B – litera D itd.



Rys. 7.7. Szyfrowanie liter szyfrem Cezara za pomocą obracających się tarcz

Warto wiedzieć

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

Warto wiedzieć

Wykorzystanie do szyfrowania obracających się tarcz z literami alfabetu zaproponował w XV w. Leon Battista Alberti, jeden z największych umysłów tamtych czasów. 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 5

a. Przygotuj fizyczny model obracających się tarcz pozwalających szyfrować tekst metodą Cezara, podobny jak na rysunku 7.7. Stwórz w programie graficznym projekt tarcz. Wydrukuj je i wytnij. Użyj pinezki, aby umocować przyrząd np. na tablicy korkowej.

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 podanym kluczem.

Warto wiedzieć

Wiele współczesnych algorytmów kryptograficznych łączy elementy podstawiania i przestawiania. Mówi się wtedy o szyfrach kaskadowych (hybrydowych). Twórcą tej koncepcji jest Claude Shannon.

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 Unicode tej litery jako kod.
   5. Jeżeli kod + klucz jest liczbą nie większą od kodu Unicode litery Z, to dołącz do szyfrogramu literę o kodzie Unicode równym kod + klucz. W przeciwnym razie dołącz do szyfrogramu literę o kodzie Unicode wynoszącym kod + klucz – 26.

Kody Unicode kolejnych liter szyfrogramu zapamiętamy w zmiennej pomocniczej kod. Aby wyznaczyć kod Unicode 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. 99

Na przykład podczas szyfrowania litery Y w słowie INFORMATYKA z kluczem 3 otrzymamy literę B. Kod Unicode 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*:

Kod źródłowy programu *Szyfr Cezara*

```
1.  szyfrogram = ""
2.
3.  jawny = input("Podaj słowo: ")
4.  klucz = int(input("Podaj klucz szyfrowania: "))
5.
6.  dl = len(jawny)
7.
8.  for i in range(0, dl):
9.      kod = ord(jawny[i]) + klucz
10.     if kod > ord("Z"):
11.         kod = kod - 26
12.     szyfrogram = szyfrogram + chr(kod)
13.
14. print("\nSzyfrogram:", szyfrogram)
```

W linii 6, korzystając z **funkcji** len, obliczamy długość podanego napisu. W pętli z linii 8–12 obliczamy kod Unicode pojedynczego znaku szyfrogramu (jednej litery) i zapisujemy go w zmiennej pomocniczej kod (linia 9). Jeśli wartość zmiennej kod jest większa od kodu litery Z (wartości funkcji ord), to odejmujemy od niej 26 (linia 11).

Funkcja len, s. 102

Rysunek 7.8 przedstawia wywołania programu *Szyfr Cezara* dla kluczy 3 i 12.



Rys. 7.8. Wywołania programu *Szyfr Cezara* dla kluczy 3 oraz 12

Ćwiczenie 6

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 dużych 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 Unicode tego znaku jako kod.
   5. Jeśli kod jest kodem Unicode spacji, to dołącz do szyfrogramu spację. W przeciwnym razie wykonaj krok 6.
      6. Jeżeli kod + klucz jest liczbą nie większą od kodu Unicode litery Z, to dołącz do szyfrogramu literę o kodzie Unicode równym kod + klucz. W przeciwnym razie dołącz do szyfrogramu literę o kodzie Unicode 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 komputerowe algorytmy szyfrowania 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, tekst):
2.      szyfrogram = ""
3.      dl = len(tekst)
4.
5.      for i in range(0, dl):
6.          if tekst[i] == " ":
7.              kod = ord(" ")
8.          else:
9.              kod = ord(tekst[i]) + klucz
10.             if (kod > ord("Z")):
11.                 kod = kod - 26
12.         szyfrogram = szyfrogram + chr(kod)
13.
14.     return szyfrogram
15.
16. jawny = input("Podaj tekst: ")
17. klucz = int(input("Podaj klucz szyfrowania: "))
18.
19. 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 1–14, a zasadnicza część programu – w liniach 16–19. Funkcja `Cezar` z parametrami `klucz` i `tekst` wykorzystuje poznaną wcześniej technikę budowania szyfrogramu z liter będących przesunięciami kolejnych liter tekstu jawnego o stałą liczbę (wartość klucza).

Funkcja `len`, s. 102

W linii 3 korzystamy z **funkcji** `len`, która wyznacza długość tekstu jawnego. Przypisujemy tę wartość zmiennej `dl`. Pętla **`for`** z linii 5–12 działa analogicznie do pętli w **programie *Szyfr Cezara***, jednak dodatkowo podczas szyfrowania pomija się znak spacji. Odpowiadają za to wiersze 6–7.

Program *Szyfr Cezara*, s. 122

Na rysunku 7.9 przedstawiono wywołania programu *Szyfr Cezara 2* dla dwóch różnych kluczy.

```
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
```

```
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
```

Rys. 7.9. Wywołania programu *Szyfr Cezara 2*

Ćwiczenie 7

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

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). Na przykład liczba potencjalnych kluczy szyfru dla klucza 256-bitowego to 2^{256}. To więcej niż 1000^{26}.

Ćwiczenie 8

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.

7.4. Metody łamania klasycznych szyfrów

Kryptografia (od gr. *kryptós* – ukryty i *gráphō* – piszę) jako nauka o utajnianiu wiadomości jest częścią ogólniejszej 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 poznanych szyfrów. 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 7.10.

Atak siłowy (brute-force)

Warto wiedzieć

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

```
Wykonywanie: Cezar siłowy.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. 7.10. Zastosowanie ataku siłowego do łamania szyfru Cezara

A to ciekawe

(Kryptologiczny) Cud nad Wisłą

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.

	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	

W **szyfrze kolumnowym** liczba możliwych kluczy również jest stosunkowo niewielka, ponieważ ogranicza ją długość tekstu jawnego. Liczba potencjalnych kluczy jest w przybliżeniu równa połowie liczby znaków tekstu. W tym przypadku równie łatwo zastosować metodę siłową.

Szyfr kolumnowy, s. 117

Ćwiczenie 9

Napisz program łamiący szyfr Cezara metodą siłową.

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. Podobnie jest z szyfrem kolumnowym – atak ze znanym tekstem jawnym pozwala odtworzyć podział na kolumny.

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 najczęstszej w danym języku.

Analiza częstości liter

Dla krótkich tekstów taka analiza może jednak dać fałszywe wyniki. Metoda ta nie ma zastosowania w wypadku szyfru kolumnowego.

Ćwiczenie 10

Znajdź informację o tym, jakie litery występują najczęściej w tekstach polskich oraz w innych językach, których się uczysz.

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 i szyfr kolumnowy nie stanowią dziś praktycznie żadnego zabezpieczenia.

Warto wiedzieć

Za bezpieczny algorytm kryptograficzny uznaje się taki, którego złamanie zajęłoby ok. miliona lat. Ponieważ moc komputerów wzrasta, rośnie też liczba bitów w zapisie kluczy kryptograficznych.

Warto wiedzieć

Krótkie teksty zaszyfrowane w prosty sposób mogą być nie do złamania nawet przez potężne komputery. Claude Shannon wykazał, że nie ma wówczas wystarczającej ilości informacji, by określić rodzaj szyfru. Niektóre teksty wyryte na starych nagrobkach pozostają nieodczytane przez setki lat mimo wysiłków wielu kryptologów.

Analiza częstości w języku polskim

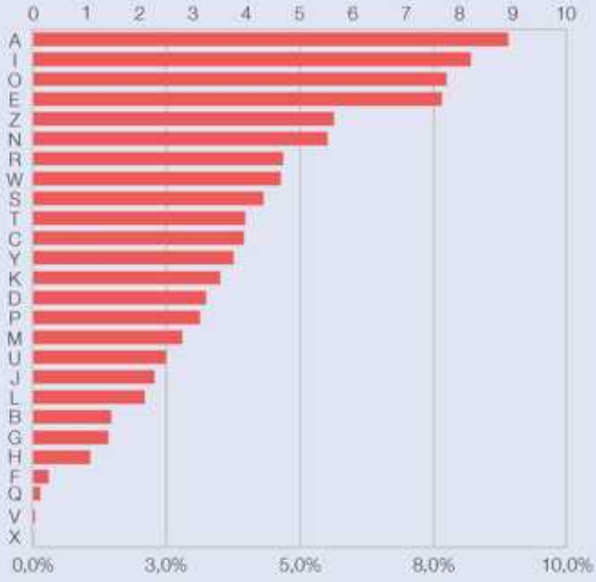
Pokażemy, jak wykorzystać analizę częstości do odszyfrowania szyfrogramu, który powstał po zaszyfrowaniu informacji w języku polskim. Posłużymy się tekstem zapisanym wyłącznie wielkimi literami alfabetu łacińskiego. W tekście nie używamy znaków interpunkcyjnych, a wyrazy są oddzielone spacjami.

Fragment szyfrogramu

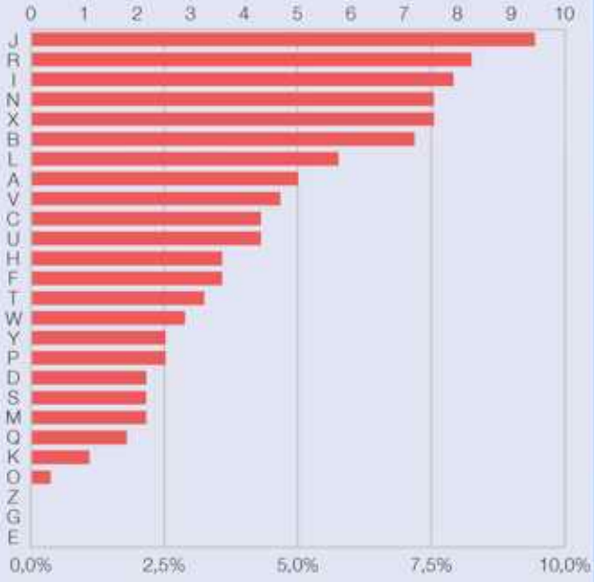
YXMLIJB UJVJWRJ BIHOAXPAJVD VXIN BRN YAIHMJL WRN CHUTX JWJURIJ
LINBCXBLR URCNA UNLI CJTIN IWJSXVXBL LINBCX FHBCNYDSJLHLQ XKXT
BRNKRN PADY MFXLQ UDK CAINLQ IWJTXF LIHUR MRPAJVXF R CARPAJVXF WJ
YAIHTUJM F SNIHTD YXUBTRV WJSLINBCBIHVR MRPAJVJVR BJ LQ LI BI R AI J
CARPAJVNV BUXFX BRN F SNIHTD JWPRNUBTRV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

Podsumowanie

- Podczas szyfrowania tekst jawny jest przekształcany w szyfrogram.
- Algorytmy służące do szyfrowania i do odszyfrowywania używają tajnego klucza, który jest wartością niezależną od szyfrowanej informacji.
- Szyfr przestawieniowy to szyfr, w którym zmienia się kolejność znaków tekstu jawnego. Wówczas w szyfrogramie występują wszystkie znaki tekstu jawnego. Przykładem takiego szyfru jest szyfr kolumnowy.
- Szyfr podstawieniowy to szyfr, w którym informacje utajnia się, zamieniając każdy znak informacji jawnej na inny znak.
- Proste szyfry podstawieniowe można złamać metodą analizy częstości (jeśli tylko tekst szyfrogramu jest odpowiednio długi) oraz metodą siłową.
- Klucz szyfrowania w szyfrze Cezara (wartość przesunięcia) łatwo określić, znajdując litery, które najczęściej występują w szyfrogramie.

Zadania

* **1** Jednym z najprostszych sposobów szyfrowania jest zapis liter tekstu wspak. Napisz program szyfrujący i deszyfrujący tą metodą tekst złożony z liter alfabetu łacińskiego bez spacji.

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

* **3** Szyfr, w którym litery tekstu jawnego pisze się na zmianę w dwóch wierszach, nazywa się czasem parkanem. Dla tekstu AKCJAOTRZYNASTEJ otrzymamy zapis kolumnowy:

```
ACATZNSE
KJORYATJ
```

Sformułuj specyfikację problemu i napisz program szyfrujący tym algorytmem.

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

* **5** 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.

** **6** Napisz program, który podany przez użytkownika tekst jawny, zapisany bez spacji, będzie szyfrować wybranym przez niego szyfrem: albo metodą kolumnową, albo szyfrem Cezara. Wybór szyfru powinien być dokonywany już po uruchomieniu programu.

** **7** 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.

** **8** Przygotuj prezentację na temat maszyny szyfrującej Enigma oraz historii złamania jej szyfru przez polskiego matematyka Mariana Rejewskiego. Przedstaw na osi czasu kluczowe wydarzenia związane z historią łamania szyfru Enigmy przed wybuchem II wojny światowej. Uwzględnij metodę zegara (wymyśloną przez Jerzego Różyckiego) i metodę płacht perforowanych (którą stworzył Henryk Zygalski).

** **9** Szyfr kolumnowy można stosować również w taki sposób, że kluczem jest nie liczba kolumn, ale ich kolejność (kolumny numerowane są od 0). Na przykład dla tekstu: PRZESYLKACZEKAWWILLI oraz klucza 40132 szyfrogram to: SCWIPYZWRLEIEAALZKKL.
Napisz program szyfrujący tekst tą metodą.

P	R	Z	E	S
Y	L	K	A	C
Z	E	K	A	W
W	I	L	L	I

** **10** 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

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

*** **12** Dany jest szyfr polegający na odwróceniu kolejności liter w kolejnych n-literowych grupach, gdzie n jest kluczem. Napisz program szyfrujący tekst tą metodą. Przyjmij, że tekst jawny na początku uzupełnia się dowolnymi literami, aby jego długość była wielokrotnością n.

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

8. Porządek ma znaczenie, czyli sortujemy liczby

Porządkowanie jest ważną częścią ludzkiej aktywności. Zwykle wysiłek w nie włożony opłaca się, bo później wiele zadań można wykonać szybciej i łatwiej. Są sytuacje, w których porządek ma szczególne znaczenie. Na przykład jeśli domy są ponumerowane w zrozumiały sposób, w razie pożaru strażacy sprawniej dotrą na miejsce. W świecie oprogramowania porządkowanie odbywa się dzięki algorytmom sortowania. W tym temacie przyjrzymy im się bliżej.

Cele lekcji

- Uświadomisz sobie, jak ważne jest porządkowanie danych.
- Lepiej zrozumiesz algorytmy wyszukiwania (liniowego i binarnego).
- Poznasz algorytmy sortowania: bąbelkowego i przez wstawianie.
- Dowiesz się, jak w języku Python używać list w argumentach funkcji.

8.1. Do czego służy sortowanie?

Sortowanie danych (ang. *sorting*), zwane też **porządkowaniem danych**, to ustawianie danych w zadanej kolejności według określonego kryterium.

Sortowanie (porządkowanie) danych

Zwykle gdy mówimy o sortowaniu liczb, mamy na myśli uporządkowanie ich w kolejności malejącej albo rosnącej. W rzeczywistości najczęściej jest to kolejność nierosnąca lub niemalejąca, ponieważ sortowane wartości mogą się powtarzać. Wyrazy z reguły ustawiamy w kolejności alfabetycznej (od A do Z) albo odwróconej kolejności alfabetycznej (od Z do A). Natomiast daty i godziny porządkuje się zwykle chronologicznie, czyli od najstarszych do najnowszych (ewentualnie w odwróconej chronologii – od najnowszych do najstarszych).

Uporządkowane informacje, zwłaszcza obszerne zbiory danych, o wiele łatwiej czytać, analizować oraz przeszukiwać.

Rysunek 8.1 przedstawia przykładową tabliczkę przystankową. Zauważ, że dane są uporządkowane na dwóch poziomach: kierunki – alfabetycznie, a godziny dla każdego kierunku – chronologicznie.

LGOTA WIELKA

ROZKŁAD ODJAZDÓW

Kierunek	Przez	Godziny odjazdu		
BOGUMIŁOWICE	SULMIERZYCE	**14:50** S		
ŁÓDŹ	BEŁCHATÓW	**6:58** dnw	**12:08** dnw	**17:03** dnw
PAJĘCZNO	SULMIERZYCE	**09:20** Dm		
RADOMSKO		**05:11** 6s	**07:01** S	**07:09** S
		07:11 S	**08:15** S	**08:15** Dm
		12:05 Dm	**14:35** dnw	**18:05** dnw
		23:18 dnw		

Rys. 8.1. Przykład tabliczki przystankowej

Ćwiczenie 1

Z jakimi uporządkowanymi zbiorami danych masz do czynienia na co dzień? Wymień co najmniej pięć konkretnych przykładów.

Szukanie hasła w słowniku lub skorowidzu

Warto wiedzieć

Porządek alfabetyczny nazywa się też leksykograficznym. W tym uporządkowaniu spacje (i inne tego typu znaki, np. dwukropek) są umieszczane przed literami alfabetu. Wynika to z kolejności znaków w kodzie ASCII. To dlatego hasło „kod źródłowy" w słowniku informatycznym będzie wcześniej niż np. hasło „kodowanie".

W encyklopediach, słownikach i leksykonach obowiązuje porządek alfabetyczny. Gdy korzystasz z tradycyjnego słownika, np. polsko-angielskiego, zwykle nie szukasz na chybił trafił, lecz intuicyjnie stosujesz bardzo rozsądną strategię poszukiwania słowa. Podobnie postępujesz podczas szukania hasła w indeksie (skorowidzu), umieszczonym zwykle na końcu książki (rys. 8.2).

A
adres komórki 11
adresowanie bezwzględne 14
— mieszane 14
— względne 14
algorytm 94, 172
— Euklidesa 158, 235
— — — wersja z dzieleniem 160, 237
— Herona 98, 176
— wybierania największej z trzech liczb 108, 185
arkusz 10
atrybut 69
— JEŻELI 17
— LICZ.JEŻELI 28
— LOS.ZAKR 33
— MAX 16
— MAX.K 63
— MIN 16
— SUMA 11
— ŚREDNIA 16
funkcja (C++) 145
— main 101
— niezwracająca wartości 163
— sqrt 150
— swap 155

Rys. 8.2. Fragment indeksu podręcznika *Informatyka na czasie 2*

Warto wiedzieć

Metodę wyszukiwania, którą stosujemy intuicyjnie podczas korzystania ze słownika, nazywa się algorytmem wyszukiwania interpolacyjnego.

Jeśli chcesz znaleźć słowo „algorytm", to prawdopodobnie szukasz go wśród początkowych haseł, a jeśli „mediana" – mniej więcej w środku. Następnie szukasz wyrazów zawierających dwie pierwsze litery – w naszej sytuacji będą to odpowiednio „al" i „me". Za każdym razem zawężasz obszar poszukiwań, pomijając strony, na których dane hasło na pewno nie wystąpi. Posortowanie haseł przez twórców słownika pozwala więc znacznie skrócić czas przeszukiwania.

A to ciekawe

Algorytmy wyszukiwania

Znalezienie w sieci informacji, których potrzebujemy, byłoby praktycznie niemożliwe bez ich uporządkowania. Systemy rankingowe wyszukiwarek internetowych przeglądają indeksy wyszukiwania, stosując wiele różnych algorytmów, analizujących takie aspekty jak: treść zapytania, przydatność stron, wiarygodność źródła, czas zamieszczenia informacji, lokalizacja użytkownika. Gdyby wykorzystywane algorytmy nie działały bardzo szybko dla dużych zestawów danych, na wyniki dla pojedynczego zapytania trzeba byłoby czekać nawet kilkanaście minut lub dłużej.



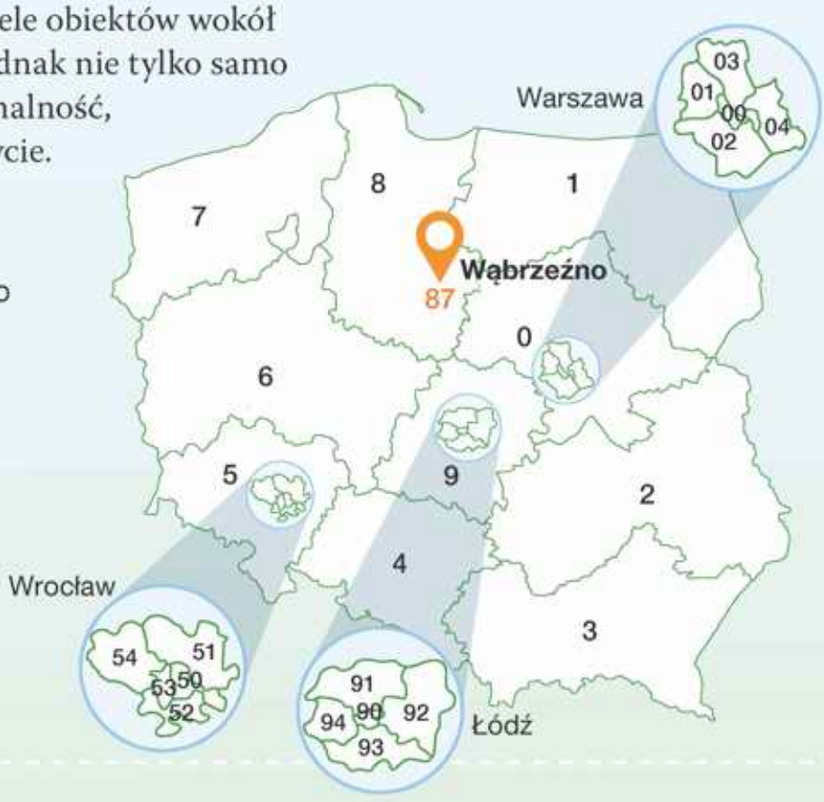
Porządek wokół nas

Zwykle nie zwracamy uwagi na to, jak wiele obiektów wokół nas jest uporządkowanych. Ważne jest jednak nie tylko samo uporządkowanie, lecz także jego funkcjonalność, która znacznie ułatwia nam codzienne życie.

Kody pocztowe

Wprowadzenie kodów pocztowych znacząco usprawniło sortowanie i skróciło czas dostarczania listów. Wcześniej pracownicy poczty musieli sami odszyfrowywać nazwy miejscowości i sprawdzać ich lokalizacje.





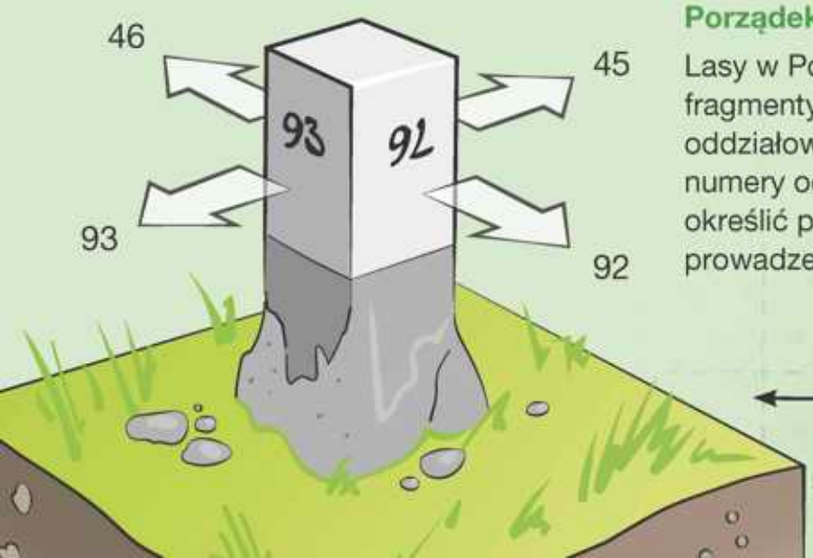
Numeracja domów

Domy znajdujące się w centrach miast są numerowane w uporządkowany sposób. Numery domów położonych wzdłuż głównych ulic rosną w kierunku od centrum miejscowości ku jej krańcom: po lewej stronie są numery nieparzyste, po prawej – parzyste.



Porządek w lesie

Lasy w Polsce są podzielone na oddziały (prostokątne fragmenty), które są identyfikowane przez tzw. słupki oddziałowe. Na słupkach znajdują się liczby oznaczające numery oddziałów leśnych. Słupki oddziałowe pomagają określić położenie, wyznaczyć kierunek oraz ułatwiają prowadzenie akcji ratowniczej.



Klasyfikacja zawodników w pięcioboju nowoczesnym

Warto wiedzieć

Na igrzyskach w Rio de Janeiro w 2016 r. polska pięcioboistka Oktawia Nowacka z wynikiem 1349 punktów wywalczyła brązowy medal.

Przyjrzyjmy się bliżej zasadom ustalania kolejności w pięcioboju nowoczesnym. Składa się on z: szermierki, pływania, jazdy konnej oraz konkurencji łączonej – biegu na przełaj i strzelania z pistoletu. Wyniki w poszczególnych konkurencjach są przeliczane na punkty wielobojowe. W tabeli 8.1 przedstawiono wyniki pierwszych dziesięciu zawodniczek w klasyfikacji końcowej Letnich Igrzysk Olimpijskich w Rio de Janeiro w 2016 r. W rywalizacji brało udział 36 zawodniczek.

			Szermierka			Pływanie			Jazda konna		Bieg ze strzelaniem			Suma
			Wygrane	Miejsce	Pkt	Czas	Miejsce	Pkt	Miejsce	Pkt	Czas	Miejsce	Pkt	
1.	Chloe Esposito	AUS	19	13.	215	00:02:12	7.	303	19.	284	00:12:10	2.	570	1372
2.	Élodie Clouvel	FRA	21	7.	227	00:02:09	2.	315	11.	293	00:12:59	17.	521	1356
3.	Oktawia Nowacka	POL	27	1.	264	00:02:17	16.	290	9.	293	00:13:18	25.	502	1349
4.	Annika Schleu	GER	17	16.	202	00:02:19	21.	282	12.	293	00:12:22	3.	559	1336
5.	Kate French	GBR	17	19.	202	00:02:06	15.	292	1.	300	00:12:43	8.	537	1331
6.	Natalya Coyle	IRL	19	12.	215	00:02:17	18.	288	4.	300	00:12:58	16.	522	1325
7.	Alice Sotero	ITA	20	9.	221	00:02:13	8.	303	23.	279	00:13:00	18.	520	1323
8.	Samantha Murray	GBR	14	31.	192	00:02:11	4.	308	22.	279	00:12:59	7.	542	1321
9.	Jelena Potapienko	KAZ	17	20.	202	00:02:12	5.	306	7.	293	00:13:07	22.	513	1314
10.	Tamara Vega	MEX	15	26.	190	00:02:17	17.	290	5.	300	00:12:49	11.	531	1311

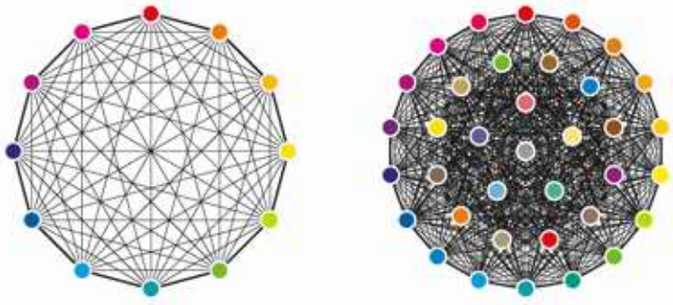
Tabela 8.1. Wyniki pięcioboju nowoczesnego kobiet – igrzyska w Rio de Janeiro w 2016 r.

Ćwiczenie 2

a. Według jakiego kryterium uporządkowano zawodniczki w tabeli 8.1?
b. Czy klasyfikacje dla każdej pojedynczej konkurencji są tworzone na podstawie wyników uporządkowanych malejąco?

Ustalenie kolejności zawodników podczas zawodów w wieloboju jest szczególnie ważne przed rozegraniem ostatniej konkurencji (strzelanie i bieg). Różnice punktowe w klasyfikacji przeliczane są bowiem na sekundy. Lider startuje jako pierwszy, a kolejni z opóźnieniem wyliczonym na podstawie dotychczasowych wyników.

Systemy wyłaniania zwycięzców wcześniejszych konkurencji muszą być sprawiedliwe. Na przykład w szermierce stosuje się system „każdy z każdym", zwany systemem kołowym. Przy dużej liczbie rywalizujących jest on jednak bardzo czasochłonny. Dla 36 zawodniczek (każda walczy 35 razy) należy przeprowadzić aż $\frac{1}{2} \cdot 36 \cdot 35 = 630$ pojedynków. Dla porównania – przy 12 zawodniczkach odbyłoby się 66 walk. Różnicę tę pokazuje rysunek 8.3.



Rys. 8.3. Ilustracja liczby pojedynków w rywalizacji „każdy z każdym" dla 12 i 36 zawodniczek

Warto wiedzieć

W języku angielskim system rywalizacji „każdy z każdym" nazywa się *round-robin tournament* (dosł. turniej karuzelowy). Z informatycznego punktu widzenia jest to algorytm sortowania porównawczo-zliczeniowego. Każdy element zbioru danych liczbowych jest porównywany ze wszystkimi pozostałymi, co daje informację, od ilu elementów jest on większy.

Sortowanie danych w komputerze

Sortowanie zbiorów danych jest jedną z operacji najczęściej wykonywanych przez komputery podczas przetwarzania informacji. Oto kilka przykładów zastosowań sortowania, od prostszych do mniej oczywistych:

- wyświetlanie listy plików w katalogu (folderze) na dysku – przed wyświetleniem zawartości katalogu system operacyjny sortuje pliki zgodnie z określonym kryterium,
- przygotowanie do wydruku danych z arkusza kalkulacyjnego lub bazy danych – często stosujemy sortowanie, aby ułatwić czytanie danych zawartych w tabelach (nadać im odpowiednie znaczenie),
- wyszukiwarki internetowe – zasoby zgromadzone na serwerach wyszukiwarek internetowych muszą być zindeksowane, aby wyszukiwarka mogła zwrócić wynik możliwie szybko,
- grafika komputerowa – obiekty w grze komputerowej lub np. w symulatorze lotu są ułożone na warstwach, więc aby program mógł wyświetlić obiekty na ekranie w dobrej kolejności, może wymagać ich wcześniejszego posortowania (względem odległości od obserwatora).

Warto wiedzieć

Przykłady problemów, do których rozwiązania potrzebne jest sortowanie danych, to:

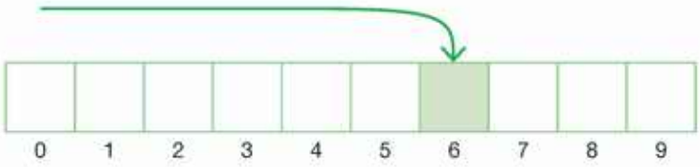
- usuwanie duplikatów,
- sprawdzanie, czy wszystkie elementy zbioru danych są różne,
- znajdowanie dwóch wartości o najmniejszej różnicy,
- wyznaczanie mediany, dominanty i innych statystyk,
- odtwarzanie pierwotnego porządku elementów zbioru danych.

8.2. Sortowanie a wyszukiwanie

Uporządkowanie danych ma zasadnicze znaczenie podczas przeszukiwania ich zbioru. Prześledźmy to na przykładzie wyszukiwania w pamięci komputera.

Przyjmijmy, że po uruchomieniu jakiegoś programu w pamięci komputera mamy zapisaną listę jako uporządkowany ciąg komórek pamięci RAM, zaadresowanych kolejnymi liczbami całkowitymi.

Jeśli procesor zna położenie danych w pamięci, to odwołuje się do nich bezpośrednio, poprzez adres komórki, w której się znajdują (rys. 8.4).



Rys. 8.4. Dostęp bezpośredni

Warto wiedzieć

Czas dostępu do danych zależy nie tylko od ich uporządkowania, lecz także od rodzaju pamięci. Różny czas osiągają rejestry procesora, pamięć podręczna, pamięć RAM czy pamięć zewnętrzna (np. flash, dysk twardy).

Aby odczytać dane zapisane w siódmej komórce fragmentu pamięci, procesor nie odczytuje wszystkich danych w kolejności, w jakiej zostały zapisane, lecz pobiera dane z właściwej komórki (rys. 8.4).

Współczesne algorytmy komputerowe często wymagają, aby czas dostępu do danych z pamięci operacyjnej był jak najkrótszy. Wtedy kluczowe znaczenie – oprócz aspektu technicznego (budowy pamięci) – ma właśnie uporządkowanie danych. Jego brak znacznie spowalnia wyszukiwanie informacji, nawet jeśli korzysta się z szybkich procesorów.

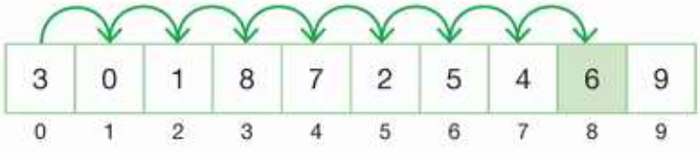
Wyszukiwanie sekwencyjne

Wyszukiwanie sekwencyjne

Kiedy szukamy informacji w zbiorach danych nieuporządkowanych, nie mamy wielkiego wyboru: albo zastosujemy metodę „na chybił trafił", albo algorytm **wyszukiwania sekwencyjnego** (ang. *sequential search*), tj. przeglądanie zbioru danych w kolejności, w jakiej są zapisane (np. według adresów w pamięci). Rysunek 8.5 pokazuje sekwencyjne wyszukiwanie liczby 6 w zbiorze nieuporządkowanym.

Warto wiedzieć

Metodę „na chybił trafił" nazywa się czasami wyszukiwaniem głupim (ang. *bogo sort*).

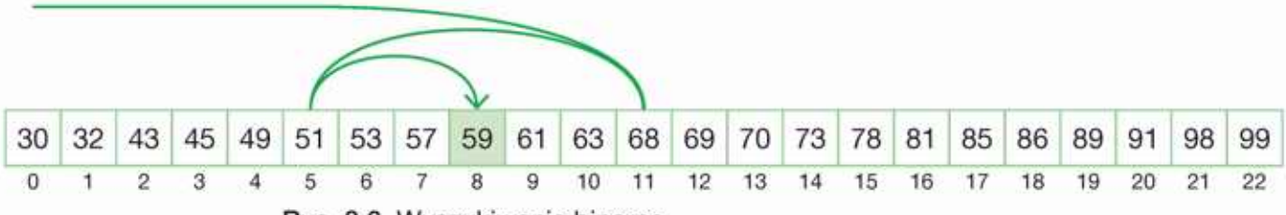


Rys. 8.5. Wyszukiwanie sekwencyjne

Wyszukiwanie binarne

Wyszukiwanie binarne (wyszukiwanie połówkowe)

Wyszukiwanie informacji można znacznie przyspieszyć, gdy zbiór danych jest uporządkowany. Stosujemy wówczas strategię połowienia zbioru, czyli zmniejszania liczby jego elementów o połowę w każdej kolejnej próbie. Nazywamy ją **wyszukiwaniem binarnym** (**połówkowym**). Algorytm ten, zastosowany do poszukiwania liczby 59 w zbiorze składającym się z 23 elementów, przedstawiono na rysunku 8.6.



Rys. 8.6. Wyszukiwanie binarne

Wyszukiwanie liczby 59 zaczęło się od sprawdzenia środkowej komórki obszaru pamięci. Jest to komórka o indeksie 11, przechowująca liczbę 68, która jest większa od 59. Ponieważ wiemy, że dane w pamięci są uporządkowane, mamy pewność, że przeszukiwanie komórek o indeksach większych od 11 można pominąć. W drugiej próbie program wyszukujący powinien sprawdzić komórkę o indeksie 5. Znajduje się w niej liczba 51, dlatego wartości należy szukać w komórkach o indeksach od 6 do 10. W kolejnym kroku sprawdzamy komórkę o adresie 8 – znajdziemy w niej poszukiwaną liczbę.

Warto wiedzieć

Porównanie dwóch liczb jest jedną z podstawowych operacji wykonywanych przez procesor komputerowy. Każdy język programowania musi obsługiwać tę operację.

Ćwiczenie 3

Ile sprawdzeń (porównań dwóch liczb) wystarczy, aby rozstrzygnąć, czy w ciągu komórek z rysunku 8.6 znajduje się liczba 80? Zapisz indeksy oraz wartości komórek, które kolejno będą sprawdzane.

8.3. Algorytmy sortowania

Spójrzmy na porządkowanie jako na problem algorytmiczny i programistyczny. W ostatnich kilkudziesięciu latach odkryto i opisano co najmniej kilkanaście użytecznych algorytmów sortowania danych. Okazuje się, że nie ma algorytmu idealnego o ogólnym przeznaczeniu, najszybszego w każdej sytuacji. Najbardziej uniwersalne są metody hybrydowe, łączące zalety dwóch lub trzech algorytmów sortowania, stosowanych na różnych etapach procesu porządkowania.

Warto wiedzieć

Algorytmy sortowania można podzielić na dwie grupy: te, które działają na zasadzie porównywania elementów zbioru parami (np. sortowanie przez wybór), oraz nieposługujące się tą operacją (np. sortowanie przez zliczanie).

Ćwiczenie 4

Przygotuj 8 kartek wielkości połowy kartki z zeszytu. Na każdej z nich zapisz imię oraz datę imienin (dzień i miesiąc). Imiona nie mogą się powtarzać. Następnie odwróć kartki i ułóż je w losowej kolejności.

a. Zaproponuj metodę sortowania informacji na kartkach (alfabetycznie lub chronologicznie). Przyjmij, że kartki trzeba porównywać parami i nie wolno odkryć więcej niż dwóch naraz.
b. W grupie 2- lub 3-osobowej przedyskutujcie zastosowane wcześniej metody porządkowania kartek. Porównajcie czas sortowania 8 i 16 kartek wybraną przez was metodą. Ile razy dłużej trwa porządkowanie 16 kartek?

Programy komputerowe mogą porządkować różne typy danych: liczby, zbiory liter i słów, daty i inne. Wszystkie te wartości są zapisywane cyfrowo w pamięci komputera. Dla uproszczenia ograniczymy się więc do porządkowania list liczb całkowitych. Dodatkowo przyjmijmy, że będziemy sortować w porządku niemalejącym.

Oto specyfikacja naszego problemu sortowania:

Specyfikacja

Dane: liczba naturalna n i n-elementowa lista A liczb całkowitych.

Wynik: lista A po uporządkowaniu: $A[0] \leqslant A[1] \leqslant \ldots \leqslant A[n-1]$.

8.4. Sortowanie bąbelkowe

Sortowanie bąbelkowe

Jednym z algorytmów porządkowania jest **sortowanie bąbelkowe** (ang. *bubble sort*). W tym algorytmie na każdym etapie porównuje się parę elementów sąsiednich, zaczynając od pierwszej pary na liście. Jeżeli podczas porównywania elementy są w niewłaściwej kolejności, to zamienia się je miejscami.

Operacja prostej zamiany

Operacja prostej zamiany

Sortowanie bąbelkowe nazywa się też czasami sortowaniem przez **prostą zamianę**, rozumianą jako zamianę miejscami sąsiednich elementów na liście.

Podobną operację stosuje się w oprogramowaniu niektórych telewizorów. Pozwala ono za pomocą pilota uszeregować listę kanałów. Są trzy możliwości: zamiana kanałów miejscami (•), przejście w dół (▽) i przejście w górę listy (△). Rysunek 8.7 przedstawia zastosowanie operacji prostej zamiany podczas ustawiania listy kanałów.

TV 3	•	TV 0	▽	TV 0		TV 0		TV 0		TV 0
TV 0		TV 3	•	TV 1	▽	TV 1		TV 1		TV 1
TV 1		TV 1		TV 3	▽	TV 3		TV 3	•	TV 2
TV 6		TV 6		TV 6	▽	TV 6	•	TV 2	△	TV 3
TV 7		TV 7		TV 7	•	TV 2	△	TV 6		TV 6
TV 2		TV 2		TV 2		TV 7		TV 7		TV 7
TV 5		TV 5		TV 5		TV 5		TV 5		TV 5
TV 4		TV 4		TV 4		TV 4		TV 4		TV 4

Rys. 8.7. Pierwsze kroki porządkowania kanałów telewizyjnych przez prostą zamianę

A to ciekawe

Sortowanie w tańcu

Na kanale AlgoRythmics w serwisie YouTube można znaleźć filmy objaśniające działanie różnych algorytmów sortowania. Porządkowanie liczb jest zaprezentowane w tańcu przez członków węgierskiego zespołu folkowego Maros Művészegyüttes. Można tam zobaczyć również m.in. wyszukiwanie binarne.

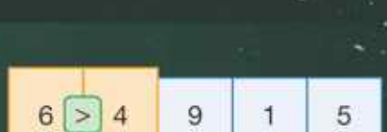


Algorytm sortowania bąbelkowego

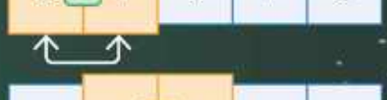
Porządkowanie elementów tą metodą pokażemy na przykładzie pięcioelementowej listy, w której na kolejnych pozycjach znajdują się liczby: 6, 4, 9, 1, 5. Elementy uporządkujemy niemalejąco. Pierwszy etap ilustruje, jak umieścić właściwy element na ostatniej pozycji. W kolejnych etapach postępujemy analogicznie.

Etap 1

- Porównujemy pierwszy element z drugim. 6 > 4, więc zamieniamy liczby miejscami.



- Porównujemy drugi element z trzecim. 6 < 9, więc nie zamieniamy liczb miejscami.



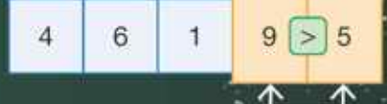
- Porównujemy trzeci element z czwartym. 9 > 1, więc zamieniamy liczby miejscami.



- Porównujemy czwarty element z piątym. 9 > 5, więc zamieniamy liczby miejscami.

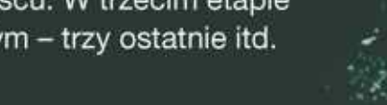


- W wyniku czterech porównań największy element listy (liczba 9) znalazł się na ostatniej pozycji.



W drugim etapie umieścimy właściwy element na przedostatniej pozycji. Przy porównaniach pomijamy ostatni element listy, ponieważ jest już na właściwym miejscu. W trzecim etapie przy porównaniach pomijamy dwa ostatnie elementy listy, w czwartym – trzy ostatnie itd. Poniżej przedstawione są wyniki otrzymane w kolejnych etapach.

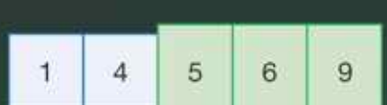
Etap 2 Porównujemy pary liczb: 4 i 6, 6 i 1 oraz 6 i 5. Liczba 6 znalazła się na właściwej, przedostatniej pozycji.



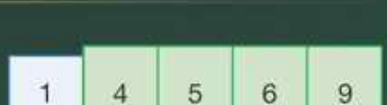
Etap 3 Porównujemy pary liczb 4 i 1 oraz 4 i 5. W wyniku tych operacji liczba 5 jest na właściwej pozycji.



Etap 4 Porównujemy parę liczb 1 i 4. Znalezienie drugiego elementu ciągu wskazuje jednocześnie najmniejszy element ciągu.



Wynikiem działania algorytmu jest lista:



Ćwiczenie 5

Dana jest lista A = [3, 0, 1, 7, 5]. Sprawdź, ile prostych zamian potrzeba, by uporządkować ją algorytmem sortowania bąbelkowego:
a. rosnąco,
b. malejąco.

Pętle zagnieżdżone, s. 112

Algorytm porządkowania metodą bąbelkową można zapisać jako **pętle zagnieżdżone**. To oznacza, że na jeden etap algorytmu (cykl pętli zewnętrznej) może przypadać wiele cykli pętli wewnętrznej.

Każdy etap algorytmu kończy się ustawieniem na docelowym miejscu elementu, który w analizowanym fragmencie był tym o największej wartości. Liczba porównań w kolejnym etapie zmniejsza się więc o 1.

Oto zapis algorytmu w postaci listy kroków:

1. Wczytaj listę liczb A.
2. Dla wartości i równych kolejno 1, 2, …, n-1 powtarzaj kroki 3 oraz 4.
3. Dla wartości j równych kolejno 0, 1, …, n-i-1 powtarzaj krok 4.
4. Jeśli A[j]>A[j+1], to zamień te elementy miejscami.

Dobra rada

Aby uporządkować elementy nierosnąco, zmień w instrukcji warunkowej znak > na <.

Liczba powtórzeń zewnętrznej pętli wynosi n–1. Liczba powtórzeń wewnętrznej pętli (tzw. pętli sortującej) jest zależna od wartości zmiennej i, sterującej zewnętrzną pętlą.

Kod źródłowy funkcji Babelkowe w programie *Sortowanie bąbelkowe* może wyglądać następująco:

Kod źródłowy funkcji Babelkowe w programie *Sortowanie bąbelkowe*

```
1.  def Babelkowe(A):
2.      n = len(A)
3.      for i in range(1, n):
4.          for j in range(0, n-i):
5.              if (A[j] > A[j+1]):
6.                  A[j], A[j+1] = A[j+1], A[j]
```

Warto wiedzieć

W nagłówku definiowanej funkcji, np. `def Funkcja(A)`, nazwa listy nie musi być identyczna z nazwą listy, dla której wywołamy funkcję. Funkcję można wywołać dla np. listy B i zapisać `Funkcja(B)`.

Zauważ, że funkcja Babelkowe jako parametr przyjmuje listę. W ten sposób przekazuje się do funkcji informację o adresie listy w pamięci. Wszystkie operacje są wykonywane bezpośrednio na liście, dla której funkcja została wywołana.

W linii 6 elementy listy są zamieniane miejscami. Zapis po prawej stronie znaku = oznacza utworzenie dwuelementowej **krotki** (ang. *tuple*), czyli odpowiednika listy, której nie można modyfikować. Interpreter przypisuje obiektom z lewej strony znaku = wartości zapisane w **krotce**. Elementowi A[j+1] przypisana zostaje kopia wartości A[j], a A[j] – kopia wartości A[j+1]. Stąd automatyczna zamiana wartości zapisanych w A[j] i A[j+1].

Krotka

Typy danych w języku Python, s. 252–253

Rysunek 8.8 przedstawia przykładowe wywołanie programu *Sortowanie bąbelkowe.*

```
Wykonywanie: sort babelkowe.py
Tablica przed sortowaniem:
[3, 0, 1, 8, 7, 2, 5, 4, 6, 9]

Tablica po sortowaniu:
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Python
```

Rys. 8.8. Przykładowe wywołanie programu *Sortowanie bąbelkowe*

Ćwiczenie 6

Napisz kod źródłowy programu o nazwie *Sortowanie bąbelkowe*, w którym wykorzystasz funkcję `Babelkowe`, a dane do sortowania wprowadzisz bezpośrednio w kodzie źródłowym. Sprawdź działanie programu.

Zapamiętaj

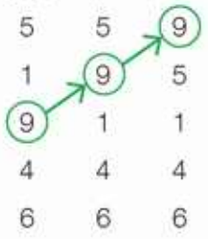
Kluczową operacją w wielu algorytmach sortowania danych jest porównywanie danych parami, by ustalić ich porządek.

Dobra rada

Zamiast wprowadzać dane do listy bezpośrednio w kodzie źródłowym, możesz napisać funkcję, która pozwoli wczytać liczby podane przez użytkownika.

Warto wiedzieć

Nazwa „sortowanie bąbelkowe" powstała przez analogię do zjawiska wypływania na powierzchnię wody bąbelków powietrza: ten o największej objętości wypływa jako pierwszy.



8.5. Sortowanie przez wstawianie

Omówimy teraz kolejny algorytm porządkowania danych – **sortowanie przez wstawianie** (ang. *insertion sort*). Metoda ta polega na przeglądaniu kolejnych elementów listy od lewej i przenoszeniu ich, w razie potrzeby, w odpowiednie miejsce już uporządkowanego fragmentu listy po lewej stronie. Działanie algorytmu przedstawiono na s. 142.

Sortowanie przez wstawianie

Sortowanie przez wstawianie przypomina układanie kart przez grających w gry karciane: karta, którą podnosimy ze stołu, jest wstawiana we właściwe miejsce do kart już uporządkowanych (rys. 8.9).



Rys. 8.9. Układanie kart metodą przez wstawianie

Warto wiedzieć

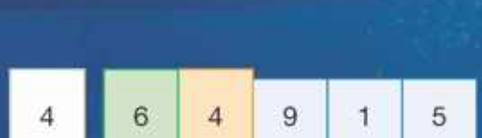
Doświadczeni gracze w pokera lub brydża nie sortują swoich kart. W ten sposób bowiem dostarczyliby przeciwnikom informacji o tym, jakie karty mają w ręku.

Algorytm sortowania przez wstawianie

Porządkowanie elementów tą metodą pokażemy na przykładzie pięcioelementowej listy, w której na kolejnych pozycjach znajdują się liczby: 6, 4, 9, 1, 5. Elementy uporządkujemy niemalejąco. Pierwszy etap ilustruje, jak uporządkować niemalejąco dwa elementy listy. Liczby przy strzałkach oznaczają kolejność wykonywania operacji.

Etap 1

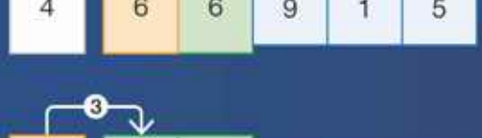
- Przyjmujemy, że pierwszy element listy (liczba 6) jest na właściwym miejscu. Drugi element, liczbę 4, zapamiętujemy w zmiennej pomocniczej.



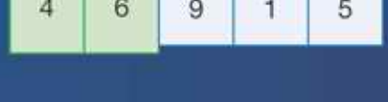
- Liczba 4 powinna się znaleźć przed liczbą 6, dlatego aby zrobić dla niej miejsce, przesuwamy liczbę 6 o jedną pozycję w prawo.



- Liczbę 4 ze zmiennej pomocniczej zapisujemy jako pierwszy element listy.



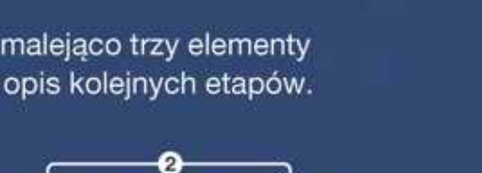
- Wynikiem jest uporządkowanie niemalejąco dwóch pierwszych elementów wyjściowej listy.



Postępując analogicznie, w drugim etapie uporządkujemy niemalejąco trzy elementy listy, w następnym – cztery itd. Poniżej znajduje się skrócony opis kolejnych etapów.

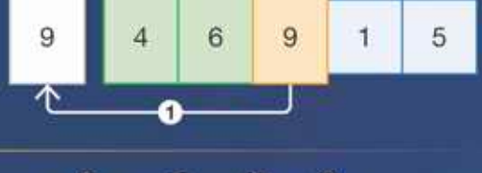
Etap 2

Zapamiętujemy liczbę 9 w zmiennej pomocniczej. Liczba 9 powinna się znaleźć za liczbami 4 i 6, dlatego nie trzeba zmieniać położenia tych liczb, a liczba 9 wraca na swoją pozycję.



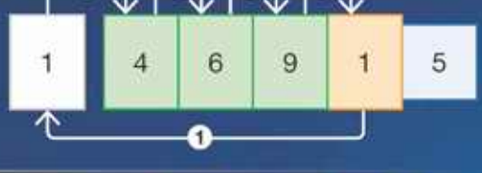
Etap 3

Zapamiętujemy liczbę 1 w zmiennej pomocniczej. Przesuwamy w prawo o jedną pozycję kolejno liczby: 9, 6 i 4. Zapisujemy liczbę 1 na pierwszym miejscu na liście.

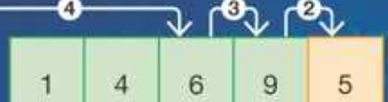


Etap 4

Zapamiętujemy liczbę 5 w zmiennej pomocniczej. Przesuwamy liczbę 9, a następnie liczbę 6 o jedną pozycję w prawo. Wstawiamy liczbę 5 ze zmiennej pomocniczej w zwolnione miejsce.



Wynikiem działania algorytmu jest lista:



Algorytm dzieli listę danych na część posortowaną i nieposortowaną (rys. 8.10). Na każdym etapie algorytmu kolejny element z nieposortowanej części listy trafia na właściwą pozycję w obszarze posortowanym.



Rys. 8.10. Umieszczenie liczby 10 na częściowo uporządkowanej liście

Algorytmy sortowania bąbelkowego i sortowania przez wstawianie są do siebie podobne: pierwszy stopniowo przesuwa elementy zbioru ku końcowi, podczas gdy drugi przenosi elementy zbioru ku początkowi.

Wstawienie elementu listy na właściwą pozycję odbywa się dzięki zwolnieniu miejsca na liście poprzez przesunięcie, krok po kroku, całego bloku elementów w prawo o jedną pozycję.

Przesuwanie elementów jest realizowane przez serię operacji kopiowania. Element, który będzie docelowo wstawiony we właściwe miejsce uporządkowanej części listy, jest tymczasowo przechowywany w zmiennej pomocniczej. Porządkowanie rozpoczynamy od drugiego elementu (o indeksie 1).

Oto zapis algorytmu w postaci listy kroków:

1. Wczytaj listę liczb A.
2. Dla wartości i równych kolejno 1, 2, …, n-1 powtarzaj kroki 3-8.
   3. W zmiennej pom zapamiętaj wartość zapisaną w A[i].
   4. Oznacz bieżącą wartość i-1 literą j.
   5. Dopóki j≥0 i A[j]>pom, wykonuj kroki 6-7.
      6. Skopiuj A[j] na miejsce A[j+1].
      7. Zmniejsz wartość j o 1.
   8. W komórce A[j+1] zapisz wartość pom.

Ćwiczenie 7

Prześledź działanie listy kroków algorytmu sortowania przez wstawianie dla następujących list.

a. A = [2, 3, 8, 5]
b. B = [2, 8, 3, 5]
c. C = [8, 2, 3, 5]

Warto wiedzieć

Sortowanie bąbelkowe i sortowanie przez wstawianie przeprowadzają operacje w miejscu, co oznacza, że poza pamięcią dla sortowanej listy potrzebują bardzo mało dodatkowej pamięci.

Warto wiedzieć

Operację znalezienia właściwego miejsca dla danego elementu w ciągu uporządkowanym można realizować przez przeszukiwanie liniowe lub binarne. Bardziej efektywny jest drugi sposób. Analizował go już w latach 50. XX w. polski matematyk Hugo Steinhaus.

Oto kod źródłowy funkcji o nazwie `Wstawianie` w programie *Sortowanie przez wstawianie*.

Kod źródłowy funkcji `Wstawianie` w programie *Sortowanie przez wstawianie*

```
1.  def Wstawianie(A):
2.      n = len(A)
3.      for i in range(1, n):
4.          pom = A[i]
5.          j = i - 1
6.          while j >= 0 and A[j] > pom:
7.              A[j+1] = A[j]
8.              j -= 1
9.          A[j+1] = pom
```

Warto wiedzieć

Podany algorytm sortowania przez wstawianie wykonuje operacje kopiowania. Liczba tych operacji jest w przybliżeniu równa liczbie porównań. Kopiowanie nie jest jednak tak czasochłonne jak zamiana, dlatego sortowanie przez wstawianie będzie kilkukrotnie szybsze od sortowania bąbelkowego.

Pętla zewnętrzna (linie 3–9) przegląda elementy z części nieposortowanej. W kolejnych iteracjach w zmiennej pomocniczej `pom` zapamiętujemy kolejne elementy z tej części (wiersz 4). Jeżeli element z lewej strony aktualnie analizowanego elementu jest większy (`A[j]>pom`), to przesuwamy elementy posortowanej części ciągu, które powinny się znaleźć za danym elementem, a jego wartość zapamiętaną w zmiennej `pom` wstawiamy na właściwe miejsce (linie 6–9).

Logika matematyczna i operatory logiczne, s. 242–244

Warunkiem w wierszu 6 jest **koniunkcja**, czyli złożone wyrażenie logiczne. Program po uruchomieniu będzie działał poprawnie, gdyż w działaniu interpretera języka Python stosuje się tzw. **leniwe wartościowanie** (ang. *lazy evaluation*). W naszym przypadku drugie porównanie się nie wykona, jeśli pierwszy warunek nie jest spełniony, tzn. gdy indeks `j` będzie mniejszy niż indeks pierwszego elementu listy.

Leniwe wartościowanie

W linii 8 zastosowano skróconą formę operacji zmniejszenia wartości zmiennej `j` o 1, równoważną zapisowi `j = j - 1`.

Ćwiczenie 8

a. Zapisz kod źródłowy funkcji `Wstawianie` i napisz program, w którym tę funkcję zastosujesz. Sprawdź działanie programu.

b. Zmodyfikuj program *Sortowanie przez wstawianie* tak, aby sortował dwie listy liczb całkowitych, np. o nazwach `ListaA` i `ListaB`. Następnie program ma posortować listę, która powstanie z połączenia dwóch list wcześniej posortowanych.

Algorytm sortowania przez wstawianie jest szczególnie efektywny przy sortowaniu zbiorów prawie uporządkowanych, np. rankingów aktualizowanych na bieżąco podczas zawodów sportowych. Wykorzystuje się go również w rozwiązaniach hybrydowych, m.in. w ostatniej fazie sortowania szybkiego (ang. *quicksort*), gdy do posortowania jest wiele podzbiorów o niewielkiej liczbie elementów.

Podsumowanie

- Sortowanie danych to ustawianie ich w zadanej kolejności według określonego kryterium.
- Uporządkowanie informacji ma ogromne znaczenie w wielu aspektach życia – zarówno w technologii komputerowej, jak i dziedzinach zupełnie z nią niezwiązanych.
- Sortowanie bąbelkowe polega na cyklicznym porównywaniu par elementów sąsiednich (zaczynając od pierwszej pary na liście) i zamianie miejscami tych elementów, które nie są w odpowiedniej kolejności.
- Sortowanie przez wstawianie polega na przenoszeniu jeden po drugim kolejnych elementów z nieuporządkowanego fragmentu listy w odpowiednie miejsce fragmentu już uporządkowanego.
- Aby funkcji zdefiniowanej w języku Python przez użytkownika umożliwić dostęp do listy zdefiniowanej w części głównej programu (np. o nazwie `lista`), w nagłówku funkcji jako parametr trzeba podać dowolną nazwę, która będzie w kodzie funkcji używana jako nazwa listy, np. `def Funkcja(lista)`.
- Funkcję wywołuje się, podając jako parametr aktualny nazwę listy, np. `Funkcja(listaA)`. Wszystkie operacje są wykonywane przez funkcję bezpośrednio na liście i zmiany są pamiętane po zakończeniu działania funkcji.

Zadania

* **1** Napisz program sprawdzający, czy lista 10 liczb jest uporządkowana. Program powinien najpierw wczytać liczby na listę.

* **2** Napisz program, który po wczytaniu 10 liczb dodatnich całkowitych znajdzie wśród nich parę liczb o największej różnicy:
- **a.** bez sortowania danych,
- **b.** po posortowaniu danych.

* **3** Napisz program, który będzie porządkować malejąco listę 10 różnych liczb metodą:
- **a.** bąbelkową,
- **b.** przez wstawianie.

* **4** Przygotuj krótką prezentację na temat systemu „każdy z każdym", stosowanego w wielu dyscyplinach sportu. W prezentacji przedstaw zasady tworzenia par turniejowych dla kolejnych rund na przykładzie 8 zawodników (lub drużyn).

* **5** Przygotuj krótkie wystąpienie o historii powstania książki *Skafander i motyl*. Wyjaśnij, dlaczego jej autor posługiwał się w komunikacji ze światem alfabetem o zmienionym porządku liter: E S A R I N T U L O M D P C F B V H G J Q Z Y X K W.

** **6** Poszukaj informacji o algorytmie sortowania przez wybór. Napisz program, który będzie sortować liczby całkowite tą metodą.

** **7** Dodaj do programu *Sortowanie bąbelkowe* kod zliczający liczbę wykonanych prostych zamian. Podaj liczbę operacji zamian potrzebnych do uporządkowania zbioru:
a. 10-elementowego,
b. 20-elementowego,
c. 100-elementowego.
Dane do posortowania otrzymasz od nauczyciela w pliku tekstowym (np. o nazwie *posortuj_liczby.txt*).

** **8** Napisz program, który na liście zawierającej liczby nieujemne przesuwa na koniec listy wszystkie elementy o wartości 0. Porządek pozostałych elementów listy powinien zostać zachowany. Program nie powinien używać dodatkowej listy podczas przetwarzania danych.

** **9** Napisz program, który zamieni miejscami elementy znajdujące się w pierwszej i drugiej połowie listy o parzystej liczbie elementów.

Dane wejściowe:	3	1	4	5	2	8	0	3
Dane wyjściowe:	2	8	0	3	3	1	4	5

Program nie powinien używać dodatkowej listy pomocniczej.

** **10** Analizując sortowanie bąbelkowe zbioru [6, 2, 5, 4, 1], można zauważyć, że wartość 6 pojawi się na właściwym miejscu po stosunkowo niewielkiej liczbie operacji, a przeniesienie wartości 1 wymaga o wiele większej liczby operacji. Mówi się, że element o wartości 1 jest w tej sytuacji żółwiem, a element o wartości 6 – zającem. Jak zmodyfikować algorytm, żeby zaradzić tej asymetrii? Przedstaw ulepszoną wersję algorytmu, nazywaną czasami sortowaniem koktajlowym (ang. *shake sort*).

** **11** Poszukaj informacji o algorytmie sortowania bibliotecznego (ang. *library sort*), który został opublikowany w 2006 r. Przygotuj prezentację na temat tego algorytmu. Wyjaśnij pochodzenie jego nazwy.

** **12** Napisz program, który będzie porządkował wyrazy według ich długości. Wyrazy o tej samej długości należy ustawić w kolejności alfabetycznej.

*** **13** Napisz program sortujący metodą przez wstawianie z użyciem wartownika.
Wskazówka: Wartownika umieść jako dodatkowy element w pierwszej komórce listy zawierającej liczby do posortowania.

*** **14** W książce *Kalejdoskop matematyczny* Hugo Steinhaus opisał algorytm sortowania przez wstawianie z wyszukiwaniem binarnym. Przygotuj prezentację na temat tego algorytmu i wykorzystania w nim wagi szalkowej. Jakie znaczenie dla tego algorytmu ma ciąg 0, 1, 3, 5, 8, 11, 14, 17, 21, ...?

*** **15** Napisz program, który po wczytaniu 20 liczb dodatnich całkowitych znajdzie wśród nich parę liczb o najmniejszej różnicy.

9. Podejście zachłanne w rozwiązywaniu problemów

Wiele problemów rozwiązywanych metodami komputerowymi to problemy optymalizacji. Przykładem jest modelowanie jak najbardziej opływowego kształtu samochodu czy ustalanie jak najszybszej trasy przejazdu dla kuriera z przesyłkami. Twój komputer również rozwiązuje takiego typu problemy, np. gdy decyduje, jak rozdzielać czas pracy procesora. Z optymalizacją spotykasz się także w życiu codziennym, np. gdy w sklepie otrzymujesz resztę wydaną najmniejszą możliwą liczbą monet. Kasjer stosuje wtedy tzw. podejście zachłanne. Z tego tematu dowiesz się, na czym ono polega.

Cele lekcji

- Dowiesz się, czym są problemy optymalizacyjne.
- Zrozumiesz, na czym polega podejście zachłanne stosowane do rozwiązywania problemów.
- Zaprogramujesz algorytmy zachłanne znajdujące optymalne rozwiązania problemu wydawania reszty oraz problemu kinomana.
- Przekonasz się, jak ważny jest dobór odpowiedniego typu danych w rozwiązaniach problemów, aby uniknąć konsekwencji błędów przybliżeń.
- Nauczysz się stosować listy równoległe jako struktury danych.

9.1. Problemy optymalizacyjne. Algorytm zachłanny

O optymalizacji można mówić w różnych sytuacjach. Na przykład w ofercie warsztatu wulkanizacyjnego znajdziemy usługę optymalizacji kół. Polega ona na takim nałożeniu opony na felgę, by siły odśrodkowe podczas kręcenia się koła działały najkorzystniej i zapewniały jego prawidłową pracę.

Warsztaty stolarskie starają się, by odpadu produkcyjnego było możliwie jak najmniej, a liczba wykonanych mebli jak największa. Podobnie firmy szyjące ubrania dbają o to, by jak najmniej materiału marnować na ścinki.

W dietetyce mówi się o optymalizacji diety, czyli układaniu indywidualnych planów żywieniowych według zadanych proporcji składników odżywczych (białka, węglowodanów, tłuszczów) oraz oczekiwanej liczby kalorii.

Wymienione przykłady są właśnie **problemami optymalizacyjnymi**, czyli takimi, które wymagają znalezienia jak najlepszego rozwiązania spełniającego określone kryterium (wymaganie).

Problem optymalizacyjny

Warto wiedzieć

Przy masowej produkcji zmniejszenie ilości odpadów choćby o kilka procent daje czasami wielomilionowe oszczędności, a także znacznie redukuje wpływ przemysłu na środowisko.

Rozwiązanie optymalne
Rozwiązanie przybliżone

Jeśli potrafimy uzasadnić, że znalezione rozwiązanie problemu jest najlepsze z możliwych, to nazywamy je **rozwiązaniem optymalnym**. Inaczej możemy mówić tylko o **rozwiązaniu przybliżonym**.

Ćwiczenie 1

Podaj przykład problemu optymalizacyjnego (inny niż opisane).

Podejście zachłanne w optymalizacji

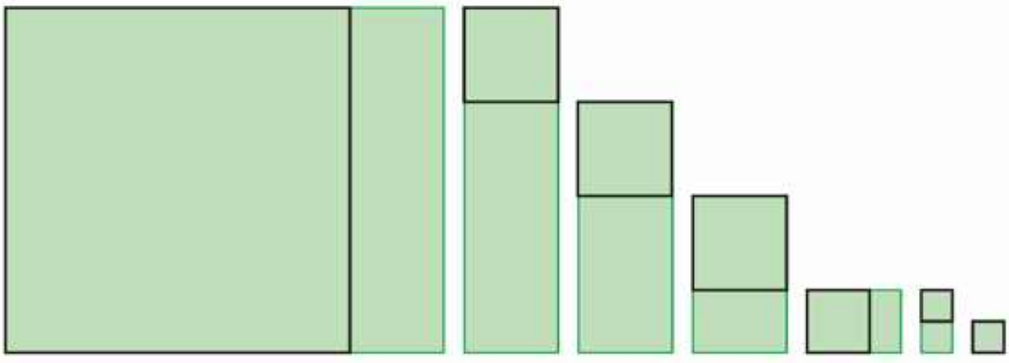
W części drugiej podręcznika omawialiśmy już jeden problem optymalizacyjny: wyznaczanie największego wspólnego dzielnika (NWD) dwóch liczb. Do rozwiązania problemu wykorzystaliśmy algorytm Euklidesa.

Podejście zachłanne (metoda zachłanna)

W algorytmie tym stosuje się metodę rozwiązywania problemów zwaną **podejściem zachłannym** (ang. *greedy approach*). Polega ono na podejmowaniu szeregu decyzji optymalnych lokalnie, czyli najlepszych (ze względu na przyjęte kryterium) na danym etapie rozwiązywania problemu. Nie sprawdzamy przy tym, jaki wpływ mają takie decyzje na znalezienie rozwiązania optymalnego globalnie, czyli najlepszego rozwiązania całego problemu.

Istotę podejścia zachłannego dobrze widać na przykładzie **wycinanki Euklidesa** przedstawionej na rysunku 9.1.

Wycinanka Euklidesa, podręcznik *Informatyka na czasie 2. Zakres podstawowy*, s. 233



Rys. 9.1. Kolejne etapy wycinanki Euklidesa

Warto wiedzieć

Czasami algorytmy zachłanne pozwalają znaleźć tylko rozwiązania przybliżone. Takie algorytmy zazwyczaj są dość szybkie i można je wykorzystać jako punkt wyjścia do szukania lepszych rozwiązań.

Dla liczb całkowitych dodatnich a i b wyznaczenie NWD(a,b) jest równoważne znalezieniu największego kwadratu, którym można by wypełnić prostokąt o bokach długości a i b.

Dobra rada

Nie używaj wyrażenia „najbardziej optymalny", gdyż jest ono niepoprawne. Coś może być najbardziej korzystne albo po prostu optymalne.

W każdym kolejnym kroku algorytmu odcinamy od danego prostokąta jak największy kwadrat (postępujemy zachłannie), a w rezultacie otrzymujemy poszukiwany kwadrat.

Algorytm Euklidesa zawsze znajduje rozwiązanie optymalne. Na przykład dla liczb 112 i 88 algorytm zwraca liczbę 8, która na pewno jest największą liczbą dzielącą 112 i 88.

W części drugiej podręcznika rozważaliśmy różne algorytmy, które rozwiązują problem optymalizacyjny wyznaczenia NWD dwóch liczb, ale tylko algorytm Euklidesa wykorzystuje do tego podejście zachłanne.

9.2. Kolorowanie mapy metodą zachłanną

Podejście zachłanne ma zastosowanie nie tylko w algorytmach liczbowych. Dzięki niemu można np. próbować ustalić, ile minimum kolorów wystarczy do wypełnienia mapy w taki sposób, aby dwa sąsiednie obszary, np. państwa lub województwa (jak na rys. 9.2), miały różne kolory.



Rys. 9.2. Mapa Polski z podziałem na województwa

Warto wiedzieć

W 1853 r. postawiono hipotezę, że cztery kolory zawsze wystarczą do pokolorowania każdej płaskiej mapy. Dowód sformułowano dopiero w 1976 r. Dzięki pracy matematyków problem udało się zredukować do jedynie 1936 układów państw na mapie. Dalsze obliczenia przeprowadził już komputer. Dotąd nikomu nie udało się udowodnić tego twierdzenia bez użycia komputera.

Ćwiczenie 2

W pliku, który otrzymasz od nauczyciela (np. *mapa.png*), znajduje się mapa konturowa Polski z podziałem na województwa. Pokoloruj ją, korzystając z możliwie najmniejszej liczby kolorów tak, aby dwa sąsiednie województwa miały różne barwy.

Warto wiedzieć

Mapę można pokolorować trzema kolorami wtedy i tylko wtedy, gdy każdy obszar na mapie (np. województwo) ma parzystą liczbę sąsiadów.

Problem kolorowania mapy, zwany też problemem ubogiego kartografa, sformułowano w połowie XIX w. Ówczesna technika drukarska pozwalała już na drukowanie barwnych map, w szczególności map politycznych. Liczbę użytych kolorów starano się jednak jak najbardziej ograniczyć, by zmniejszyć koszty konstrukcji maszyny drukarskiej.

Problem kolorowania mapy

Okazuje się, że do pokolorowania każdej płaskiej mapy wystarczą cztery kolory. Nie jest jednak znany algorytm takiego optymalnego kolorowania. Podejście zachłanne dla niektórych map daje tylko rozwiązanie przybliżone: spełniony jest warunek różnych barw dla sąsiednich obszarów, ale w sumie wykorzystuje się np. pięć barw.

9.3. Wydawanie reszty metodą zachłanną

Wyobraź sobie, że podczas zakupów otrzymujesz przy kasie resztę 8 zł i 45 gr wydaną wyłącznie monetami o wartości 1 zł oraz 5 gr. Łącznie to 17 monet. Gdyby w ten sposób wydawano resztę w sklepach, trudno byłoby ją zmieścić w portfelu, a w kasie mogłoby zabraknąć monet. Dlatego dąży się do tego, aby liczba monet była możliwie jak najmniejsza. Jest to tak zwany **problem wydawania reszty**.

Problem wydawania reszty

Jak automaty wydają resztę?

Automaty sprzedające napoje, przekąski lub bilety stały się powszechne. Część z nich pozwala płacić bezgotówkowo, ale nadal wiele przyjmuje banknoty i monety.





Rozpoznawanie monet

Monety wrzucone do automatu są poddawane działaniu fal elektromagnetycznych. Pozwala to określić grubość monet, ich wagę oraz materiał, z którego są wykonane. System zapadni sterowany przez procesor na płycie sterującej kieruje monety do właściwych przegródek.



Sterowanie automatem

Po zaakceptowaniu kwoty płyta sterująca uruchamia mechanizm wydawania produktu. Automat zwraca również resztę – tak, aby użyć najmniejszej możliwej liczby monet.

Jak wydać resztę najmniejszą liczbą monet?

1. Szukamy największego dostępnego nominału nie większego od reszty.
2. Wydajemy nominał tyle razy, ile można, aby nie przekroczyć kwoty reszty.
3. Pomniejszamy resztę do wydania o łączną wartość już wybranych monet, a następnie powtarzamy kroki 1 i 2.

Kwota do wydania	Nominał
9,45 zł	
4,45 zł	
0,45 zł	
0,05 zł	

Ćwiczenie 3

Wyobraź sobie, że korzystasz z automatu na dworcu kolejowym. Z jakich monet składa się reszta, jeśli automat wydaje ją najmniejszą możliwą liczbą monet, a ty płacisz banknotem 10 zł za zakup:

a. wody za 2,70 zł,
b. herbaty za 3,40 zł,
c. przekąski za 4,19 zł?

Zakładamy, że w automacie nie brakuje żadnych monet.

Napiszemy teraz program, który będzie wyznaczał nominały potrzebne do wydania reszty. Przyjmijmy, że reszta jest kwotą mniejszą niż 10 zł. Będzie więc wydana wyłącznie z użyciem monet o nominałach: 5 zł, 2 zł, 1 zł, 0,50 zł, 0,20 zł, 0,10 zł, 0,05 zł, 0,02 zł i 0,01 zł. Nominały te nie powinny się zmieniać w trakcie wykonywania programu, dlatego zapiszemy je w **krotce**.

Krotka, s. 140

Oto specyfikacja problemu:

Specyfikacja

Dane: kwota reszty do wydania – liczba z przedziału [0,01; 9,99]; krotka `NOMINALY` zawierająca uporządkowane malejąco liczby od 5 do 0,01, odpowiadające dostępnym nominałom.

Wynik: kolejne wartości nominałów potrzebnych do wydania reszty.

Zauważ, że przedstawiony na s. 150 **algorytm wydawania reszty** (zwany niekiedy algorytmem kasjera) realizuje **podejście zachłanne**. Wybieramy największy nominał, który jest mniejszy lub równy kwocie reszty, i wydajemy go tyle razy, ile można, aby nie przekroczyć tej kwoty. Następnie wybieramy kolejny nominał i powtarzamy proces. Pamiętamy przy tym, aby za każdym razem zmniejszać kwotę reszty.

Algorytm wydawania reszty

Podejście zachłanne, s. 148

Oto zapis algorytmu w postaci listy kroków:

1. Wczytaj kwotę reszty do wydania i zapisz ją w zmiennej reszta.
2. Ustaw na 0 wartość zmiennej i wskazującej element krotki NOMINALY.
3. Dopóki reszta>0, wykonuj kolejno kroki 4–7.
   4. Jeśli reszta>=NOMINALY[i], to wykonaj kroki 5 i 6.
      5. Wypisz na ekranie wartość NOMINALY[i].
      6. Pomniejsz wartość zmiennej reszta o wartość NOMINALY[i].
   7. W przeciwnym przypadku zwiększ i o 1.

Zakładamy, że wartości groszowe będziemy wprowadzać do programu jako ułamki dziesiętne (stosujemy zapis z kropką zamiast przecinka). Dlatego w kodzie źródłowym programu wykorzystamy **typ zmiennoprzecinkowy** `float` oraz funkcję `float`.

Typ `float`, s. 253

Oto kod źródłowy programu *Wydawanie reszty*.

Kod źródłowy programu *Wydawanie reszty*

```
1.  NOMINALY = (5, 2, 1, 0.50, 0.20, 0.10, 0.05, 0.02, 0.01)
2.  N = len(NOMINALY)
3.
4.  def WydajReszte(reszta):
5.      i = 0
6.      while reszta > 0 and i < N:
7.          if reszta >= NOMINALY[i]:
8.              print(NOMINALY[i])
9.              reszta = reszta - NOMINALY[i]
10.         else:
11.             i += 1
12.
13. reszta = float(input("Podaj kwotę reszty: "))
14. WydajReszte(reszta)
```

Dobra rada

Przy definiowaniu krotki możesz pominąć nawiasy po prawej stronie znaku przypisania =.

Krotka, s. 140

W linii 1 definiujemy **krotkę** NOMINALY, która przechowa wartości kolejnych nominałów w postaci liczb zmiennoprzecinkowych. W drugiej linii określamy stałą N jako liczbę elementów krotki NOMINALY. W liniach 4–11 jest zapisana funkcja WydajReszte **niezwracająca wartości**, która wyznacza i wypisuje na ekranie kolejne nominały potrzebne do wydania reszty. Monet któregoś nominału w wydawanej reszcie może nie być wcale, a monet innego nominału może być kilka. Dlatego też przed przejściem do kolejnego nominału (linia 11) sprawdza się warunek z linii 7.

Funkcja niezwracająca wartości, s. 256

Rysunek 9.3 przedstawia przykładowe wywołanie programu *Wydawanie reszty*.

```
Wykonywanie: wydawanie resztv.pv
Podaj kwotę reszty: 4.78
2
2
0.5
0.2
0.05
0.02
0.01
Python
```

Rys. 9.3. Wywołanie programu *Wydawanie reszty* dla kwoty 4 zł 78 gr

Dobra rada

Pamiętaj, że liczby przechowywane są w komputerze w postaci binarnej, np. liczba 0,1 nie ma skończonego rozwinięcia dwójkowego, lecz okresowe: 0,000(1100). W konsekwencji w pamięci komputera będzie zapisana wartość przybliżona, trochę większa od 0,1.

Ćwiczenie 4

Zapisz kod źródłowy programu *Wydawanie reszty* i uruchom program. Sprawdź jego działanie dla różnych wartości reszty, w tym dla 7.30, 7.79 i 7.85.

Okazuje się, że dla niektórych kwot reszty program wyświetla błędne wyniki – po zsumowaniu wartości monet brakuje 1 gr. Na przykład suma monet wyznaczonych przez program dla wartości 7.85 wynosi 7.84. Przedstawia to rysunek 9.4.



Rys. 9.4. Błędny wynik dla kwoty 7 zł 85 gr

Niedokładny rezultat nie wynika z błędu w algorytmie ani z zastosowania podejścia zachłannego. Jest on skutkiem dodawania wartości typu zmiennoprzecinkowego, które są tylko przybliżeniami właściwych wartości. Obliczenia z wykorzystaniem tego typu zmiennoprzecinkowego są obarczone konsekwencjami **błędów przybliżeń**.

Błąd przybliżenia

Zauważ, że program *Wydawanie reszty* – choć wyświetla niepoprawne rozwiązanie – kończy swoje działanie, nie zawiesza się. Zawdzięczamy to warunkowi `i<N` w warunku logicznym pętli **`while`**. Brak tego zapisu spowoduje błąd: „IndexError: tuple index out of range" (indeks krotki poza zakresem).

Ćwiczenie 5

W programie utworzonym w poprzednim ćwiczeniu usuń w kodzie drugą część warunku logicznego, tj. `i<N`. Plik zapisz pod nazwą *wydawanie_reszy_petla.py*. Sprawdź działanie programu dla różnych wartości reszty, w tym dla 7.30, 7.79 i 7.85. Co obserwujesz?

Poprawiony kod źródłowy dla algorytmu wydawania reszty

Aby rozwiązać problem błędów przybliżeń, możemy zastosować **typ całkowitoliczbowy** i sprowadzić wszystkie kwoty reszty do wartości wyrażonych w groszach.

Typ całkowitoliczbowy, s. 252

Nominały 5 zł, 2 zł i 1 zł zapisane w groszach mają wartości odpowiednio 500, 200, 100. Sformułujmy poprawioną specyfikację problemu wydawania reszty.

Specyfikacja

Dane: kwota reszty do wydania w postaci dwóch liczby całkowitych: liczba złotych z przedziału [0; 9] i liczba groszy z przedziału [0; 99]; krotka `NOMINALY` z informacją o dostępnych nominałach – liczby całkowite (wielokrotności 1 grosza) uporządkowane malejąco.

Wynik: kolejne wartości nominałów potrzebnych do wydania reszty podane w złotówkach.

Dobra rada

Zawsze dokładnie czytaj specyfikację problemu. Często narzuca ona typ danych, który należy zastosować w programie.

Oto kod źródłowy programu *Wydawanie reszty 2*:

Kod źródłowy programu *Wydawanie reszty 2*

```
1.  NOMINALY = (500, 200, 100, 50, 20, 10, 5, 2, 1)
2.
3.  def WydajReszte(reszta):
4.      i = 0
5.      while reszta > 0:
6.          if reszta >= NOMINALY[i]:
7.              print(NOMINALY[i]/100)
8.              reszta = reszta - NOMINALY[i]
9.          else:
10.             i += 1
11.
12. reszta_zlote = \
13.     int(input("Podaj liczbę złotych reszty: "))
14. reszta_grosze = \
15.     int(input("Podaj liczbę groszy reszty: "))
16. WydajReszte(reszta_zlote * 100 + reszta_grosze)
```

Dobra rada

Zauważ, że warunek pętli `while` (wiersz 5) jest uproszczony. Nie musisz się martwić o przekroczenie indeksów krotki `NOMINALY`, ponieważ funkcja `WydajReszte` operuje tylko na liczbach całkowitych.

Dobra rada

Aby podzielić wiersz kodu, zakończ go spacją i znakiem ukośnika \.

W liniach 12–15 pobieramy wartość kwoty do wydania – oddzielnie złotówki i grosze. Takie rozwiązanie pozwala łatwo przeliczyć złotówki na grosze. Funkcję `WydajReszte` wywołujemy dla parametru wyrażonego w groszach (linia 16) – w tym celu trzeba przeliczyć złotówki na grosze.

Definicja funkcji `WydajReszte` (linie 3–10) jest podobna do poprzedniej, jednak funkcja operuje już na liczbach całkowitych. Przy wypisywaniu wartości potrzebnych monet (linia 7) podajemy wynik w postaci liczb niecałkowitych (w złotówkach).

Przykładowe wywołanie programu pokazuje rysunek 9.5.

Warto wiedzieć

Oprogramowanie stosowane w bankach i innych instytucjach finansowych również wykonuje obliczenia na liczbach całkowitych.



Rys. 9.5. Wywołanie programu *Wydawanie reszty 2* dla kwoty 8 zł 30 gr

Warto wiedzieć

Dopiero w 2009 r. został opublikowany artykuł naukowy wyjaśniający, jak sprawdzić, czy dla danego zbioru nominałów algorytm zachłanny na pewno znajdzie rozwiązanie optymalne niezależnie od wydawanej kwoty. Autorami artykułu było dwoje Polaków: Anna Adamaszek i Michał Adamaszek.

Ćwiczenie 6

a. Zapisz kod programu w pliku *Wydawanie reszty 2*. Uruchom program dla kilku różnych kwot reszty, np. 7 zł 30 gr, 7 zł 85 gr.
b. Zmodyfikuj kod programu tak, aby można było do niego wprowadzać kwotę reszty większą niż 10 zł, a resztę można było wydawać z użyciem monet i banknotów.

Optymalność zachłannego algorytmu wydawania reszty jest ściśle związana ze zbiorem dostępnych nominałów. Można sobie wyobrazić fikcyjny system monetarny, np. o nominałach 25, 10, 1, dla którego algorytm zachłanny nie znajduje optymalnego rozwiązania. Na przykład dla reszty 30 otrzymamy sumę 25 + 1 + 1 + 1 + 1 + 1 zamiast 10 + 10 + 10.

Ten przykład wystarcza, aby stwierdzić, że w ogólnym przypadku wydawanie reszty metodą zachłanną nie musi prowadzić do uzyskania **rozwiązania optymalnego**.

Rozwiązanie optymalne, s. 148

Zapamiętaj

W pamięci komputera nie możemy zapisać w sposób dokładny wszystkich ułamków dziesiętnych, ponieważ niektóre z nich (np. 0,1) mają nieskończone rozwinięcie binarne. Aby zapobiec kumulowaniu się błędów przybliżeń w programach komputerowych, warto zastosować zapis wykorzystujący liczby całkowite.

9.4. Problem kinomana

Wyobraźmy sobie, że pewien miłośnik filmu bierze udział w jednodniowym festiwalu filmowym. W opublikowanym programie festiwalu znajdują się informacje takie jak: tytuł filmu, gatunek filmu, godzina rozpoczęcia seansu, czas projekcji, numer sali kinowej.

Kinoman chce zobaczyć tyle filmów, ile tylko będzie możliwe. Zakładamy, że nie może on w tej samej godzinie zakończyć seansu jednego filmu i rozpocząć drugiego. Potrzebuje czasu na przemieszczenie się, a filmy ogląda w całości. W jaki sposób powinien postępować, aby obejrzeć ich jak najwięcej? Tak sformułowany problem znany jest w literaturze jako **problem kinomana**.

Problem kinomana

Zanim sformułujemy specyfikację problemu, zastanówmy się, jak można reprezentować poszczególne filmy oraz co oznacza podejście zachłanne w problemie kinomana.

A to ciekawe

Systemy monetarne a wydawanie reszty

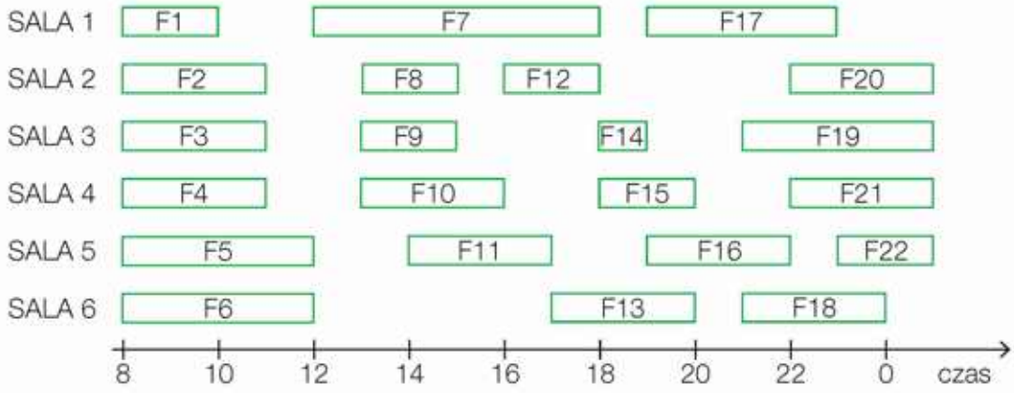
Systemy monetarne wielu krajów są tak skonstruowane, aby metoda zachłanna działała w nich optymalnie, tzn. żeby można było wydać resztę najmniejszą liczbą banknotów i monet. Polski system monetarny jest zbudowany na bazie liczb: 5, 2, 1. Wartości banknotów i monet są wielokrotnościami tych liczb.



Harmonogram festiwalu można przedstawić graficznie tak jak na rysunku 9.6. Filmy są reprezentowane przez prostokąty ułożone chronologicznie wzdłuż linii czasu. Długość prostokąta odpowiada długości trwania filmu. Dodatkowo każdy film ma identyfikator od F1 do F22.

Dobra rada

Zauważ, że numery w identyfikatorach filmów odzwierciedlają kolejność ich rozpoczęcia.



Rys. 9.6. Harmonogram festiwalu filmowego

Warto wiedzieć

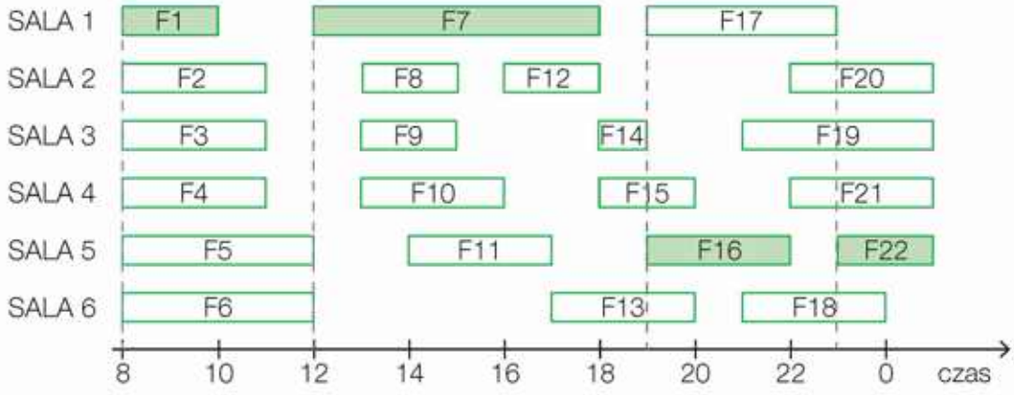
Innym kryterium, które można zastosować w problemie kinomana, jest jak najmniejsza liczba konfliktów filmu z innymi filmami.

Aby wybrać możliwie najwięcej filmów, możemy postępować zachłannie według różnych kryteriów. Oto niektóre z nich:

- kryterium 1: filmy zaczynające się najwcześniej,
- kryterium 2: filmy najkrótsze,
- kryterium 3: filmy kończące się najwcześniej.

Załóżmy ponadto, że jeśli kilka filmów spełnia dane kryterium, to wybieramy ten, który kończy się najwcześniej. Jeśli takich filmów jest więcej, wybieramy dowolny.

Rysunek 9.7 przedstawia rozwiązanie z uwzględnieniem kryterium 1. W pierwszej kolejności wybieramy najwcześniej rozpoczynający się film (F1). Następnie szukamy filmu, który rozpocznie się najwcześniej po zakończeniu pierwszego seansu – będzie to film F7. Po jego zakończeniu najwcześniej rozpocznie się film F16, a ostatnim wybranym filmem jest F22.



Rys. 9.7. Filmy wybrane na podstawie kryterium 1

Ćwiczenie 7

W pliku otrzymanym od nauczyciela (np. *kinoman.png*) oznacz filmy, które obejrzy kinoman, jeśli zastosuje kryteria 2 oraz 3.

Można wykazać, że zachłanny wybór najwcześniej kończących się filmów (kryterium 3) daje zawsze **rozwiązanie optymalne**.

Zapiszmy specyfikację problemu kinomana bardziej precyzyjnie i zastanówmy się nad algorytmem wykorzystującym to kryterium.

Rozwiązanie optymalne, s. 148

Specyfikacja

Dane: liczba naturalna N – liczba filmów; trzy listy N-elementowe liczb całkowitych:
P – z godzinami rozpoczęcia projekcji kolejnych filmów,
K – z godzinami zakończenia projekcji kolejnych filmów,
IDEN – z identyfikatorami kolejnych filmów.

Wynik: identyfikatory i godziny rozpoczęcia filmów tworzących optymalne rozwiązanie.

Załóżmy dla uproszczenia, że czas projekcji jest wielokrotnością godziny (60 minut).

Aby wyłonić listę filmów, korzystając z metody zachłannej, wystarczy posortować filmy według godziny ich zakończenia, a następnie wybrać pierwszy z tych filmów i wielokrotnie (w pętli) powtórzyć ten sam sposób wyboru dla listy filmów, które rozpoczynają się po zakończeniu poprzedniego filmu.

Przedstawmy algorytm rozwiązania problemu kinomana w postaci listy kroków:

1. Wczytaj dane do list P, K oraz IDEN.
2. Uporządkuj listę K niemalejąco i dostosuj kolejność elementów na listach P oraz IDEN.
3. Wypisz na ekranie IDEN[0], P[0] oraz K[0]. Wartość K[0] zapisz w zmiennej koniec_poprzedniego.
4. Oznacz jako i zmienną wskazującą element listy i ustaw ją na 1.
5. Dla wartości i równych kolejno 1, 2, …, N-1 powtarzaj kroki 6-8.
   6. Jeśli P[i]>koniec_poprzedniego, to wykonaj kroki 7-8.
      7. Wypisz na ekranie IDEN[i], P[i] oraz K[i].
      8. Zapisz wartość K[i] w zmiennej koniec_poprzedniego.

Zapiszmy na listach P, K i IDEN odpowiednie dane. Zmienna N przechowa długość listy P, ale jednocześnie reprezentuje długość list K i IDEN (zakładamy, że wszystkie trzy listy mają tę samą długość).

```
1.  P = [9, 16, 15, 18, 14, 10, 10, 13, 15, 18]
2.  K = [10, 17, 17, 21, 15, 14, 11, 14, 17, 20]
3.  IDEN = ["F0", "F1", "F2", "F3", "F4", "F5", \
4.          "F6", "F7", "F8", "F9"]
5.  N = len(P)
```

Fragment kodu źródłowego programu *Problem kinomana* – definicja list

Dobra rada

Jeśli chcesz udowodnić, że wybór najwcześniej kończących się filmów daje optymalne rozwiązanie, załóż, że istnieje lepsza metoda, która da większą liczbę filmów. Spróbuj wykazać fałszywość przyjętego założenia.

Warto wiedzieć

Jeśli informacja o jednym filmie znajduje się na kilku listach, to zmiana pozycji jednego elementu musi powodować analogiczne zmiany odpowiednich elementów na pozostałych listach.

Sortowanie przez wstawianie, s. 141–144

Do sortowania filmów można użyć **algorytmu sortowania przez wstawianie**. Oto kod źródłowy funkcji sortującej `Sortuj` w programie *Problem kinomana*:

Kod źródłowy funkcji `Sortuj` w programie *Problem kinomana*

```
1.  def Sortuj(koniec, poczatek, iden):
2.      for i in range(1, N):
3.          pom_koniec = koniec[i]
4.          pom_poczatek = poczatek[i]
5.          pom_iden = iden[i]
6.          j = i - 1
7.          while j >= 0 and koniec[j] > pom_koniec:
8.              koniec[j+1] = koniec[j]
9.              poczatek[j+1] = poczatek[j]
10.             iden[j+1] = iden[j]
11.             j -= 1
12.
13.         koniec[j+1] = pom_koniec
14.         poczatek[j+1] = pom_poczatek
15.         iden[j+1] = pom_iden
```

Dobra rada

Pamiętaj, że lista `koniec` przechowuje godziny zakończenia kolejnych seansów. Dlatego to dla jej elementów wykonywane są operacje porównania.

Warto wiedzieć

Sortowanie danych na powiązanych ze sobą listach jest podobne do sortowania w tabelach arkusza kalkulacyjnego. Zmiana kolejności danych w jednej kolumnie powoduje odpowiednie modyfikacje w pozostałych kolumnach tabeli.

Funkcja `Sortuj` ma 3 parametry – listy zawierające informacje o końcach i początkach seansów oraz identyfikatorach filmów. Zwróć uwagę, że choć warunek pętli **`while`** (linia 7) odnosi się wyłącznie do listy `koniec`, to równolegle modyfikowane są pozostałe listy: `początek` oraz `iden`. Pierwszy warunek pętli **`while`** (linia 7) sprawdza, czy indeks jest liczbą mniejszą od rozmiaru listy, a drugi warunek sprawdza, czy na liście są jeszcze filmy do uporządkowania.

Dzięki temu te same identyfikatory na wszystkich trzech listach nadal będą się odnosić do tego samego filmu. Tak powiązane ze sobą listy nazywa się **listami równoległymi**.

Listy równoległe

Poniżej znajdują się: kod źródłowy funkcji `Wybierz`, której zadaniem jest wybranie możliwie największej liczby filmów metodą zachłanną, oraz kod źródłowy funkcji `Wypisz`, która ułatwia wypisanie interesujących nas informacji o filmie.

Kod źródłowy funkcji `Wypisz` oraz `Wybierz` w programie *Problem kinomana*

```
1.  def Wypisz(koniec, poczatek, iden, k):
2.      print(iden[k], ":", poczatek[k], "-", koniec[k])
3.
4.  def Wybierz(koniec, poczatek, iden):
5.      Wypisz(koniec, poczatek, iden, 0)
6.      koniec_poprzedniego = koniec[0]
7.      for i in range(1, N):
8.          if poczatek[i] > koniec_poprzedniego:
9.              Wypisz(koniec, poczatek, iden, i)
10.             koniec_poprzedniego = koniec[i]
```

Funkcja `Wypisz` pozwala wypisać informacje o filmie: identyfikator oraz godzinę rozpoczęcia i zakończenia.

Kod funkcji `Wybierz` znajduje się w liniach 4–10. W linii 5 wypisujemy na ekranie dane filmu o indeksie 0. Jeśli wcześniej posortujemy filmy, to pierwszy film (o indeksie 0) jest tym, który powinien obejrzeć kinoman. Następnie wybierane są filmy, które rozpoczną się bezpośrednio po aktualnie wybranym (linia 8) i mają najwcześniejszą godzinę zakończenia.

Oto zasadnicza część kodu źródłowego programu *Problem kinomana*:

Zasadnicza część kodu źródłowego w programie *Problem kinomana*

```
1.  print("Harmonogram festiwalu:")
2.  for i in range(0, N):
3.      Wypisz(K, P, IDEN, i)
4.  print()
5.
6.  Sortuj(K, P, IDEN)
7.  print("Filmy po sortowaniu:")
8.  for i in range(0, N):
9.      Wypisz(K, P, IDEN, i)
10. print()
11.
12. print("Wybrane filmy:")
13. Wybierz(K, P, IDEN)
```

Linie 1–4 odpowiadają za wypisanie harmonogramu festiwalu. W liniach 6–10 filmy są sortowane, a następnie wypisuje się je na ekranie. Wywołanie funkcji `Wybierz` w linii 13 powoduje wybór filmów z użyciem podejścia zachłannego i wypisanie ich na ekranie.

Rysunek 9.8 przedstawia przykładowe wywołanie programu *Problem kinomana*.

```
Harmonogram festiwalu:
F0 : 9 - 10
F1 : 16 - 17
F2 : 15 - 17
F3 : 18 - 21
F4 : 14 - 15
F5 : 10 - 14
F6 : 10 - 11
F7 : 13 - 14
F8 : 15 - 17
F9 : 18 - 20
```

```
Filmy po sortowaniu:
F0 : 9 - 10
F6 : 10 - 11
F5 : 10 - 14
F7 : 13 - 14
F4 : 14 - 15
F1 : 16 - 17
F2 : 15 - 17
F8 : 15 - 17
F9 : 18 - 20
F3 : 18 - 21
```

```
Wybrane filmy:
F0 : 9 - 10
F7 : 13 - 14
F1 : 16 - 17
F9 : 18 - 20

>>>
```

Rys. 9.8. Przykładowe wywołanie programu *Problem kinomana*

Warto wiedzieć

Znajomość algorytmu rozwiązującego problem kinomana może się przydać osobom chcącym uczestniczyć w największych festiwalach filmowych na świecie. Na przykład berliński festiwal Berlinale co roku oferuje do wyboru ok. 400 filmów z całego świata.

Ćwiczenie 8

Zapisz pełen kod źródłowy programu *Problem kinomana* i uruchom program.

Podsumowanie

- Problemy optymalizacyjne to problemy, w których szukamy rozwiązania jak najlepszego, spełniającego określone kryterium.
- Podejście zachłanne w rozwiązywaniu problemu polega na podejmowaniu szeregu decyzji, z których każda wydaje się najkorzystniejsza w danym momencie, bez analizowania ich wpływu na optymalność całego rozwiązania. Stosowanie tej strategii prowadzi czasami do rozwiązania optymalnego, a czasami tylko do rozwiązania przybliżonego.
- Problem wydawania reszty polega na przedstawieniu danej kwoty reszty za pomocą jak najmniejszej liczby banknotów i monet. Aby rozwiązać ten problem metodą zachłanną, należy wybierać w danym kroku jak największy nominał nie większy od kwoty, która pozostała jeszcze do wydania.
- Aby zapobiec kumulowaniu się błędów przybliżeń, należy unikać używania typu zmiennoprzecinkowego `float`.

Zadania

* **1** Napisz program, który liczbę wpisaną z klawiatury, mniejszą niż 256, zapisze jako sumę potęg liczby 2. Zdefiniuj w kodzie listę o wartościach 128, 64, 32, 16, 8, 4, 2, 1.

* **2** W pliku, który otrzymasz od nauczyciela (np. *mapa.pdf*), znajdują się różne mapy. Korzystając z metody zachłannej, wyznacz minimalną liczbę kolorów, której trzeba użyć do pokolorowania każdej mapy tak, aby każde dwa sąsiednie obszary mały różne kolory.

* **3** Napisz program poszukujący rozwiązania problemu wydawania reszty, gdy masz ograniczoną liczbę każdego z nominałów. Zwróć uwagę, że nie każdą resztę uda się wydać (program powinien wtedy wypisać odpowiedni komunikat).

* **4** Zmodyfikuj kod źródłowy programu *Problem kinomana* tak, aby dane wczytywać z klawiatury.

** **5** Napisz program, który wczyta z klawiatury informacje o kolegach i koleżankach z klasy: imię, pseudonim oraz kolor oczu. Po wprowadzeniu danych program powinien uporządkować je według koloru oczu i wyświetlić informacje o osobach rozdzielone na grupy w zależności od koloru oczu. Grupa najliczniejsza powinna być wyświetlona na początku, a grupa najmniej liczna – na końcu. Zastosuj listy równoległe.

*** **6** Na świecie rzadko stosuje się systemy monetarne, w których nie ma nominału 2. Przykładem jest szyling kenijski, którego monety mają wartości: 1, 5, 10 i 20 szylingów, a banknoty: 50, 100, 200, 500 i 1000 szylingów. Napisz program wydający resztę w tym systemie z użyciem podejścia zachłannego. Czy algorytm zachłanny jest w tym wypadku optymalny?

10. Rekurencja

W programowaniu rozwiązań problemów stosowaliśmy do tej pory iterację, czyli technikę polegającą na wielokrotnym powtarzaniu – w pętli – instrukcji zapisanych w tekście programu. Istnieje jednak inna technika programistyczna, która również wiąże się z powtarzaniem tego samego fragmentu kodu, choć instrukcja pętli jawnie w kodzie źródłowym nie występuje. Jest to rekurencja. W tym temacie dowiesz się, czym jest funkcja rekurencyjna, jakie ma zalety, a także dlaczego należy jej używać z rozwagą.

Cele lekcji

- Dowiesz się, czym jest rekurencja, oraz zrozumiesz istotę rekurencyjnego rozwiązywania problemów.
- Poznasz przykład prostego fraktala w postaci drzewa binarnego.
- Nauczysz się zapisywać funkcje rekurencyjne w języku Python.
- Poznasz ograniczenia, jakie ma stosowanie techniki rekurencji.
- Nauczysz się na przykładach, jak rekurencję zastąpić iteracją i na odwrót.

10.1. Czym jest rekurencja?

Rekurencja lub **rekursja** (ang. *recursion*) w programowaniu oznacza odwołanie się w wybranym kroku algorytmu rozwiązywania problemu do tego samego problemu, ale dla mniejszego rozmiaru danych. Algorytm, w którym stosuje się takie podejście, nazywamy **algorytmem rekurencyjnym**.

Rekurencja (rekursja)

Algorytm rekurencyjny

Podejście rekurencyjne w rozwiązywaniu problemu algorytmicznego jest procesem dwufazowym. Pierwsza faza to redukcja rozmiaru problemu, druga to łączenie częściowych wyników pochodzących z rozwiązania tych mniejszych problemów.

Jeśli danego problemu nie potrafimy rozwiązać, gdyż n-elementowy zbiór danych jest zbyt złożony, to umiejętnie redukujemy go do podobnego problemu, ale dla zbioru o np. $n - 1$ elementach. Taki problem może być łatwiej rozwiązać. Jeśli ten mniejszy problem wciąż jest zbyt złożony, to możemy dalej redukować jego złożoność, aż dojdziemy do takiego rozmiaru problemu, który potrafimy rozwiązać od razu. Później, w fazie zbierania i łączenia częściowych wyników, uzyskujemy rozwiązanie całego problemu.

Aby lepiej zrozumieć rekurencję, przyjrzyjmy się różnym jej przykładom.

Warto wiedzieć

Efekt podobny do rekurencyjnego można uzyskać, wykorzystując lustra i aparat fotograficzny w sposób pokazany na zdjęciu. Można też stanąć plecami do dużego lustra i zrobić zdjęcie przednim aparatem telefonu (tzw. selfie).

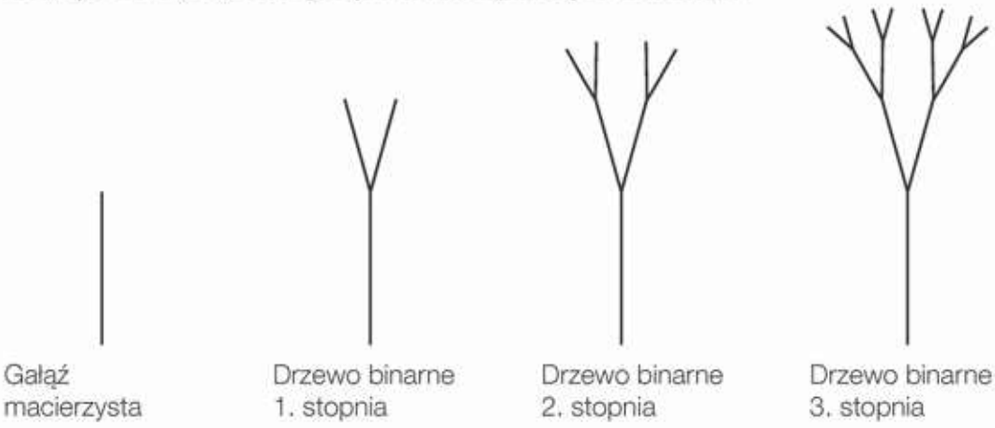


Rekurencyjna budowa drzewa binarnego

Fraktal to figura geometryczna, której podstawową cechą jest samopodobieństwo (podobieństwo wewnętrzne, powtarzalność). Oznacza to, że we fraktalu można wyodrębnić fragmenty podobne do całej figury.

Fraktal

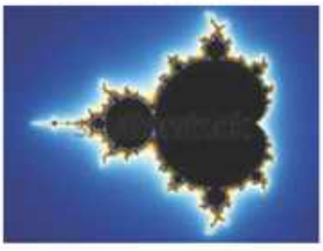
Drzewo binarne ● Oto kilka przykładów **drzew binarnych** o różnym stopniu złożoności (rys. 10.1). Są one przykładami prostych fraktali.



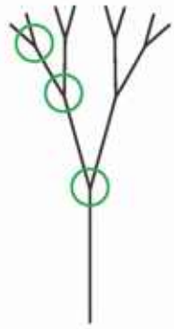
Rys. 10.1. Drzewa binarne o różnym stopniu złożoności

Warto wiedzieć

Pojęcie fraktala wprowadził w latach 70. XX w. Benoît B. Mandelbrot, urodzony w Warszawie w rodzinie litewskich Żydów, którzy po I wojnie światowej zamieszkali w Polsce. Jednym z najbardziej znanych fraktali jest zbiór Mandelbrota, nazywany również żukiem Mandelbrota.



Drzewo 1. stopnia składa się z dwóch gałęzi odchodzących od gałęzi macierzystej. Gałęzie te mają długość równą 0,6 długości gałęzi macierzystej i są odchylone od kierunku przez nią wyznaczonego o ten sam kąt – 15 stopni w prawo i w lewo. Gałęzie końcowe drzewa 2. stopnia mają te same parametry w odniesieniu do gałęzi drzewa 1. stopnia. Analogicznie są zbudowane drzewa kolejnych stopni.



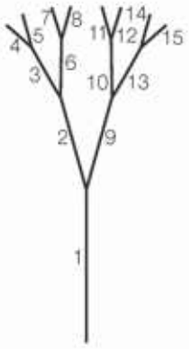
Rys. 10.2. Części samopodobne drzewa binarnego

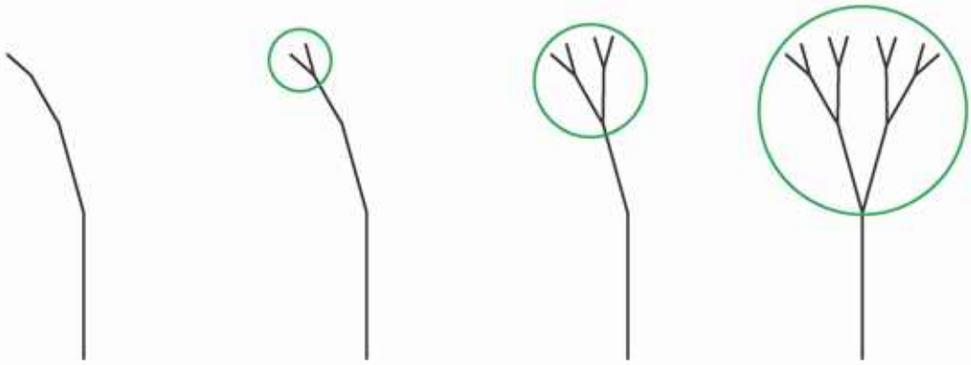
Rysunek 10.2 przedstawia drzewo binarne z zaznaczonymi miejscami rozwidlenia, które wskazują na cechę samopodobieństwa: fragmenty z wyższych części drzewa są podobne do całości. Drzewo 3. stopnia składa się z gałęzi macierzystej oraz dwóch drzew 2. stopnia odpowiednio pomniejszonych i obróconych.

Programy rysujące fraktale najczęściej wykorzystują funkcje rekurencyjne. Rysunek 10.3 przedstawia wybrane etapy rysowania drzewa binarnego 3. stopnia realizowanego przez funkcję rekurencyjną.

Warto wiedzieć

Odcinki tworzące drzewo binarne 3. stopnia są rysowane w następującej kolejności:





Rys. 10.3. Proces rekurencyjnego rozrastania się drzewa binarnego 3. stopnia

Najpierw rysowana jest gałąź macierzysta. Następnie powstaje drzewo 2. stopnia (w odpowiedniej skali) odchylone w lewo, a dopiero po jego narysowaniu – drzewo 2. stopnia odchylone w prawo.

Aby narysować każde z drzew 2. stopnia, należy w podobny sposób narysować drzewa 1. stopnia (rozpoczynając od lewej strony).

Drzewo binarne rozrasta się w sposób rekurencyjny: z każdego rozwidlenia wyrastają dokładnie dwa drzewa niższego stopnia.

Rekurencyjny przepis na ciasto francuskie

Ciasto francuskie charakteryzuje się dużą ilością zawartego w nim powietrza. Zawdzięczamy to warstwowej strukturze uzyskiwanej przez wielokrotne składanie ciasta (rys. 10.4).

Aby przygotować ciasto francuskie, należy na pewnym etapie wielokrotnie wykonywać takie same polecenia: ciasto rozwałkuj na długość równą 3-krotności jego szerokości. Następnie złóż je na 2 razy, aby uzyskać 3 warstwy i schłodź. Powtórz te czynności jeszcze 5 razy.



Rys. 10.4. Wałkowanie i składanie ciasta francuskiego

Warto wiedzieć

Gdyby fragment przepisu na ciasto francuskie dotyczący wałkowania, składania i chłodzenia ciasta zapisać nieco bardziej formalnie, mógłby on brzmieć następująco:

```
Rozwałkuj-Złóż-Schłodź(n):
  jeśli n = 1, to zakończ
  rozwałkuj ciasto
  złóż je na 3 warstwy
  schłodź przez 1 godzinę
  Rozwałkuj-Złóż-Schłodź(n-1)
```

Zauważ, że tak sformułowany opis odwołuje się do samego siebie w słowach „Powtórz te czynności". Jest to cecha każdego algorytmu rekurencyjnego.

Ćwiczenie 1

Tabliczka czekolady składa się z 16 kawałków (4 rzędy po 4 kawałki). Zapisz rekurencyjny algorytm łamania czekolady na 16 pojedynczych kostek.

A to ciekawe

Efekt Droste – rekurencja w projektowaniu

Około 1900 r. na puszce kakao produkowanego przez holenderską firmę cukierniczą Droste pojawił się wizerunek pielęgniarki, która trzymała w ręku tacę z filiżanką oraz puszką kakao Droste. Na tej puszce widać pielęgniarkę trzymającą w ręku tacę z filiżanką oraz puszką kakao Droste itd. Ten sposób prezentacji grafiki został pod koniec lat 70. XX w. nazwany efektem Droste. Jest to rodzaj rekurencyjnego obrazu. Również dziś można zaobserwować ten zabieg m.in. na puszce polskiego mleka zagęszczonego.



10.2. Rekurencyjna definicja ciągu

Warto wiedzieć

Ciąg (c_n) jest ciągiem geometrycznym. Do obliczania jego wyrazów można więc użyć wzoru: $c_n = 3^{n-1}$

Wróćmy raz jeszcze do przykładu ciasta francuskiego. Ciasto powstałe jako efekt pierwszego wałkowania możemy nazwać ciastem początkowym o numerze 1. Z niego powstaje 3-warstwowe ciasto o numerze 2, a z niego 9-warstwowe ciasto o numerze 3 itd.

Zauważmy, że i-te ciasto powstaje z ciasta o numerze $i - 1$, które po złożeniu na trzy części zostaje rozwałkowane. Ciąg opisujący liczbę warstw ciasta to: 1, 3, 9, 27, ... Z matematycznego punktu widzenia można

Ciąg zdefiniowany rekurencyjnie

mówić o ciągu c_n **zdefiniowanym rekurencyjnie** w następujący sposób:

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

Warunek początkowy

Pierwszą część definicji nazwiemy **warunkiem początkowym**. Określa on, kiedy zakończyć proces odwołań rekurencyjnych z drugiej części definicji. Zgodnie z drugą częścią definicji kolejny wyraz ciągu wyznacza się, korzystając z wartości wyrazu poprzedniego.

Aby obliczyć wartość c_n dla danego n, należy wcześniej wyznaczyć wyraz c_{n-1}. Żeby jednak obliczyć tę wartość, należy znać wyraz c_{n-2} itd. Proces redukowania wartości indeksu n skończy się, gdy n przyjmie wartość 1 (warunek początkowy).

Wyrazy ciągów zdefiniowanych rekurencyjnie można łatwo obliczyć w arkuszu kalkulacyjnym. Dla powyższego ciągu wystarczy wpisać w komórce C1 wartość wyrazu początkowego, czyli 1, a w komórce C2 – formułę opisującą zależność rekurencyjną: =3∗C1, i skopiować formułę w dół. Wartość w komórce C4 (rys. 10.5) jest obliczana błyskawicznie. Arkusz wyznacza ją jednak krok po kroku, ponieważ formuła w komórce C4 odwołuje się do formuły z komórki C3, która odwołuje się do formuły z komórki C2 itd. Łańcuch kolejnych wywołań rekurencyjnych (rys. 10.6) kończy się w komórce C1, zawierającej stałą wartość. Dalej następuje faza zbierania wyników, która kończy się wstawieniem do komórki C4 wyniku 27.

	C
1	1
2	3
3	9
4	=3*C3

Rys. 10.5. Ciąg w arkuszu kalkulacyjnym

Dobra rada

Jeśli chcesz prześledzić krok po kroku obliczenia wykonywane w formułach arkusza kalkulacyjnego, skorzystaj z narzędzia **Szacuj formułę**, dostępnego na karcie **Formuły**.



Rys. 10.6. Podgląd wywołań rekurencyjnych w narzędziu **Szacuj formułę**

Napiszemy teraz programy obliczające wybrany wyraz ciągu rekurencyjnego. Sformułujmy specyfikację problemu.

Specyfikacja

Dane: liczba naturalna n.

Wynik: n-ty wyraz ciągu danego wzorem:

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

Program *Ciąg rekurencyjnie*

Zapiszmy funkcję o nazwie `CiagRek`, która będzie wyznaczała n-ty wyraz ciągu w sposób rekurencyjny. Oto jej kod źródłowy:

```
1.  def CiagRek(n):
2.      if n == 1:
3.          return 1
4.      return 3 * CiagRek(n-1)
```

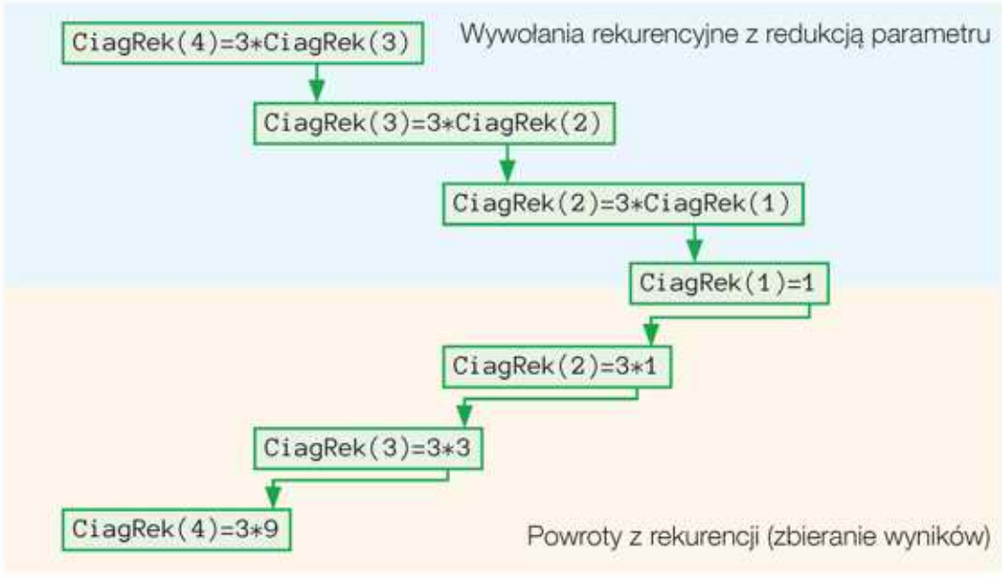
Kod źródłowy funkcji `CiagRek` w programie *Ciąg rekurencyjnie*

Budowa wnętrza **funkcji rekurencyjnej** odzwierciedla dwa możliwe przypadki. Pierwszy to warunek początkowy (linie 2–3), drugi to kod **wywołania rekurencyjnego**, czyli wywołania funkcji przez samą siebie, ale dla innej wartości parametru (linia 4).

Funkcja rekurencyjna

Wywołanie rekurencyjne

Na rysunku 10.7 przedstawiono fazy działania funkcji `CiagRek`: wywołania rekurencyjne dla coraz mniejszych wartości parametru n oraz powroty z rekurencji (zbieranie wyników).



Rys. 10.7. Schemat wywołań rekurencyjnych funkcji `CiagRek` i powrotów z rekurencji

Dobra rada

Nazwa „rekurencja" pochodzi od łacińskiego słowa *recurrere*, czyli „przybiec z powrotem". Jeśli chcesz zrozumieć daną funkcję rekurencyjną, przeanalizuj dokładnie właśnie powroty z rekurencji.

W pierwszej części schematu strzałki wskazują kolejne wywołania rekurencyjne funkcji `CiagRek` dla coraz mniejszych parametrów. Druga część schematu to faza zbierania wyników – strzałki pokazują, które wartości są przekazywane dalej.

Warto wiedzieć

Kody źródłowe funkcji, które są zapisem algorytmów rekurencyjnych, same mają strukturę podobną do rekurencyjnej definicji ciągu. Składają się z dwóch części: pierwsza określa warunek początkowy, a druga opisuje, jak przypadek dla parametru n zredukować do prostszego (np. $n - 1$).

Rysunek 10.8 przedstawia przykładowe wywołanie programu *Ciąg rekurencyjnie*.

```
Wykonywanie: ciag rekurencyjnie.py
n = 10
10. wyraz ciągu cn wynosi: 19683
>>>
```

Rys. 10.8. Przykładowe wywołanie programu *Ciąg rekurencyjnie*

Ćwiczenie 2

a. W pliku otrzymanym od nauczyciela (np. *wyraz_ciagu_rek.py*) znajduje się fragment kodu źródłowego zawierający funkcję `CiagRek`. Dopisz kod, który pozwoli wczytać liczbę `n` z klawiatury oraz wywołać funkcję `CiagRek`. Sprawdź działanie programu.

b. Napisz program, który obliczy wartość n-tego wyrazu ciągu a_n danego wzorem:

$$a_n = \begin{cases} 6 & \text{dla } n = 1 \\ 2 \cdot a_{n-1} & \text{dla } n > 1 \end{cases}$$

Sprawdź działanie programu i podaj wynik dla $n = 12$

Program *Ciąg iteracyjnie*

Wyraz ciągu c_n zdefiniowanego rekurencyjnie można również obliczyć w sposób iteracyjny. Zastosujemy w tym celu **pętlę `for`**, gdyż liczba cykli pętli potrzebnych do wyznaczenia n-tego wyrazu ciągu jest z góry określona: będzie ich $n - 1$.

Pętla `for`, s. 255

Kod źródłowy funkcji `CiagIter` w programie *Ciąg iteracyjnie*

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

Dobra rada

Instrukcję przypisania `c = 3 * c` możesz zapisać krócej jako `c *= 3`.

W linii 2 zmiennej `c` przypisywana jest wartość 1. Jeśli użytkownik wywoła funkcję `CiagIter` 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.

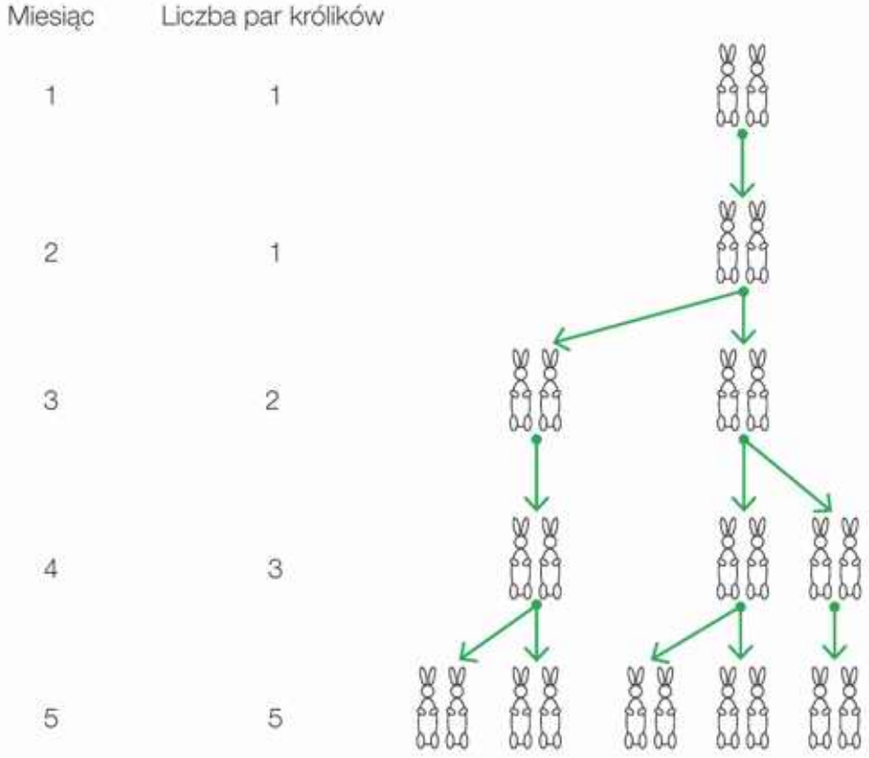
Ćwiczenie 3

Zapisz kod źródłowy programu *Ciąg iteracyjnie*. Uruchom program dla różnych wartości.

10.3. 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? Wyjaśnia to rysunek 10.9.



Rys. 10.9. 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. 10.9). 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 rekurencyjną:

○ Ciąg Fibonacciego

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

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.

Złota proporcja

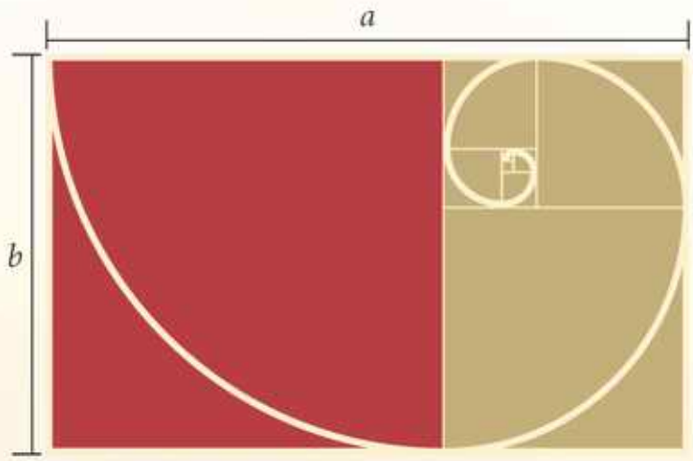
Złota proporcja, nazywana też złotym podziałem, znana jest od wieków. Przyrodnicy zachwycają się jej pięknem, którego przykłady można odnaleźć w naturze. Zainspirowało to wiele osób do poszukiwania i stosowania złotego podziału w różnych dziedzinach życia.

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 czerwono), to uzyskamy mniejszy prostokąt (na rysunku złoty) 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



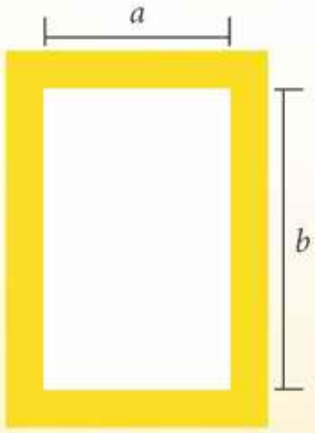
Złota proporcja w projektowaniu

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



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

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

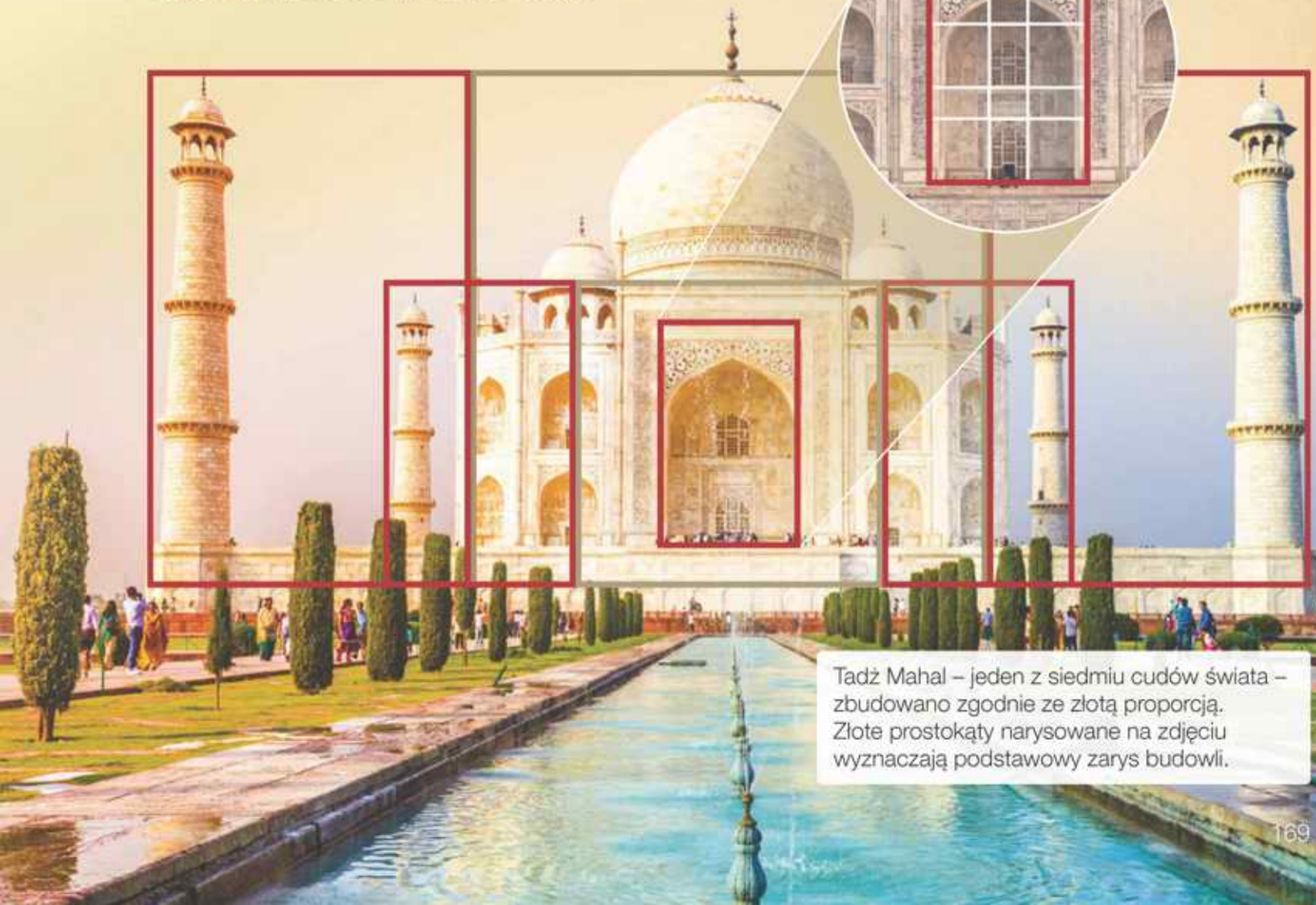


Wewnętrzny prostokąt w logo National Geographic zachowuje złotą proporcję

Złota proporcja w architekturze i sztuce

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

Okna i drzwi wyznaczają linie przecięcia złotych prostokątów



Tadż Mahal – jeden z siedmiu cudów świata – zbudowano zgodnie ze złotą proporcją. Złote prostokąty narysowane na zdjęciu wyznaczają podstawowy zarys budowli.

Warto wiedzieć

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

Napiszemy programy obliczające dany wyraz ciągu Fibonacciego. Przyjmijmy następującą specyfikację problemu:

Specyfikacja

Dane: liczba całkowita $n \geqslant 0$.

Wynik: n-ty wyraz ciągu Fibonacciego.

Program *Fibonacci rekurencyjnie*

Zapisanie funkcji rekurencyjnej sprowadza się do przepisania na wybrany język programowania definicji matematycznej.

Oto kod źródłowy funkcji FibRek w programie *Fibonacci rekurencyjnie*:

Kod źródłowy funkcji FibRek w programie *Fibonacci rekurencyjnie*

```
1.  def FibRek(n):
2.      if n == 0:
3.          return 0
4.      if n == 1:
5.          return 1
6.      return FibRek(n-1) + FibRek(n-2)
```

Kod z linii 2–3 oraz 4–5 odpowiada za wyznaczenie wartości funkcji dla n równego 0 lub 1. Jest to sprawdzenie **warunku początkowego** definicji ciągu. W linii 6 następuje podwójne (w dwóch krokach) wywołanie rekurencyjne funkcji FibRek dla parametrów: n–1 oraz n–2. Na rysunku 10.10 przedstawiono przykładowe wywołanie programu:

Warunek początkowy, s. 164

```
Wykonywanie: Fibonacci rekurencyjnie.py
n = 11
11. wyraz ciągu Fibonacciego wynosi: 89
>>>
```

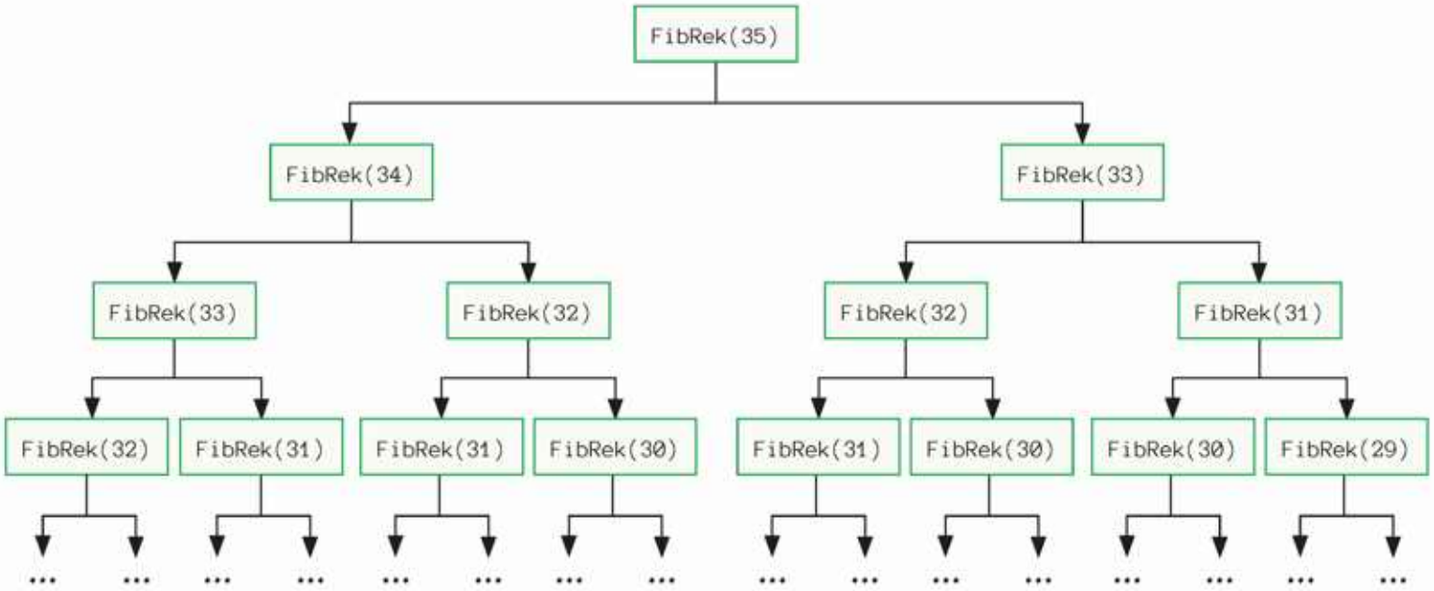
Rys. 10.10. Przykładowe wywołanie programu *Fibonacci rekurencyjnie*

Ćwiczenie 4

Napisz program według powyższej specyfikacji z wykorzystaniem funkcji FibRek. Sprawdź działanie programu dla liczb: 5, 10 i 35.

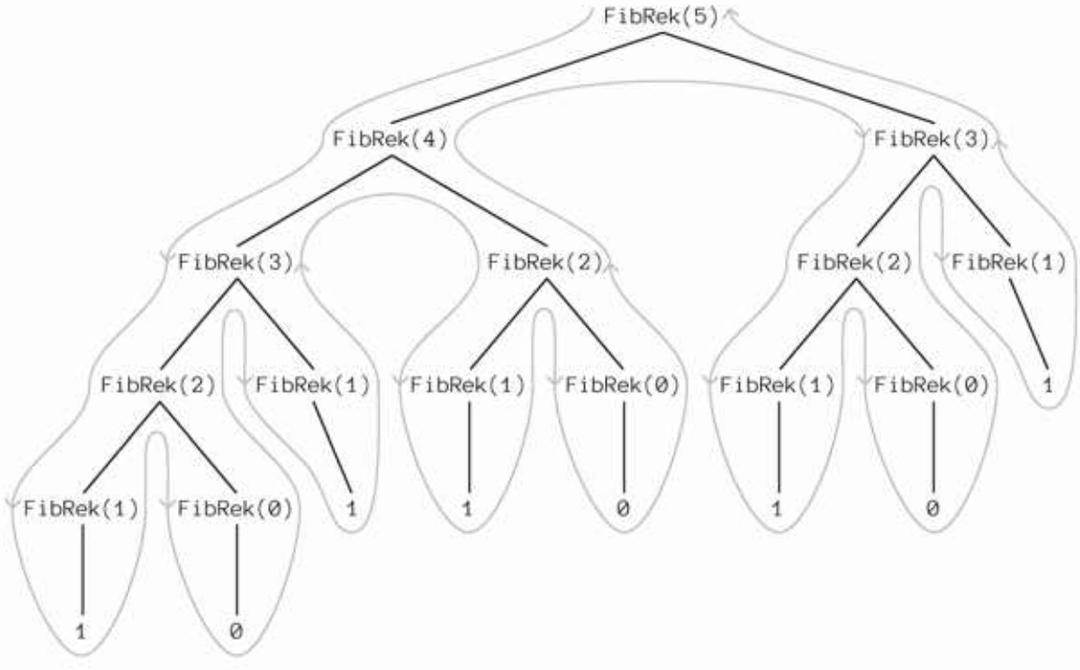
Okazuje się, że funkcja rekurencyjna FibRek, wyznaczająca kolejne liczby Fibonacciego, choć krótka w zapisie, może prowadzić do znaczącego marnowania zasobów komputera na wielokrotne wywoływanie funkcji rekurencyjnej z tymi samymi argumentami.

Na przykład już obliczenie 35. wyrazu ciągu Fibonacciego jest czasochłonne nawet dla współczesnych komputerów. Rysunek 10.11 przedstawia początek schematu wywołań rekurencyjnych funkcji `FibRek`. Zwróć uwagę na liczbę powtarzających się identycznych wywołań.



Rys. 10.11. Schemat wywołań rekurencyjnych funkcji `FibRek` dla $n = 35$

Rysunek 10.12 przedstawia fazy: wywołań rekurencyjnych i zbierania wyników, które składają się na obliczenie wartości `FibRek(5)`. Strzałki wskazują kolejność wywołań i powrotów rekurencyjnych.



Rys. 10.12. Drzewo wywołań rekurencyjnych funkcji `FibRek` dla $n = 5$

Powyższe obserwacje skłaniają do poszukiwania innego rozwiązania, które nie będzie wykorzystywało tak wielu zasobów komputera do wyznaczenia stosunkowo niskich wartości.

Warto wiedzieć

Gdyby w kodzie źródłowym funkcji `FibRek` zabrakło zapisu dotyczącego wartości funkcji dla `n = 0` oraz `n = 1`, to efektem byłaby rekurencja nieskończona. Prowadziłaby ona do przerwania działania programu po wyczerpaniu zasobów pamięci komputera. Interpreter zwróci wtedy komunikat: *maximum recursion depth exceeded.*

Program *Fibonacci iteracyjnie*

Zdecydowanie mniej pamięci komputerowej będzie potrzebował program, w którym liczby Fibonacciego będziemy obliczać iteracyjnie.

Zauważ, że w każdym momencie tworzenia nowych wyrazów ciągu wystarczy wiedza o dwóch ostatnich jego wyrazach. Zapamiętajmy w zmiennej a większą wartość, a w zmiennej b – mniejszą. 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 `FibIter`, która wyznacza n-ty wyraz ciągu Fibonacciego w sposób iteracyjny:

Kod źródłowy funkcji `FibIter` w programie *Fibonacci iteracyjnie*

```
1.  def FibIter(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
```

Warto wiedzieć

Każdy algorytm, który da się zapisać iteracyjnie, można zapisać również rekurencyjnie i na odwrót. Często zapis rekurencyjny algorytmu jest bardziej zwięzły.

W liniach 2–3 nadajemy początkowe wartości zmiennym a i b. W pętli (linie 4–7) obliczamy wartość następnego wyrazu zapisanego w zmiennej pom, a następnie zmieniamy wartości zmiennych tak, aby w zmiennej b znalazł się wyraz mniejszy (linia 6), a w zmiennej a wyraz większy (linia 7).

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

Ćwiczenie 5

a. Napisz program według podanej wcześniej specyfikacji. Wykorzystaj funkcję `FibIter`. Sprawdź działanie programu dla liczb: 5, 10 i 35.

b. Zapisz kod funkcji z użyciem krotki, tj. wiersze 2–3 oraz 5–7 zastąp pojedynczymi instrukcjami.

Zapamiętaj

Używanie rekurencji zwykle pozwala uzyskać bardziej zwięzły kod niż w przypadku iteracji. Czasami jednak ze względu na bardzo dużą liczbę wywołań rekurencyjnych rozwiązanie rekurencyjne jest bezużyteczne.

10.4. Rekurencyjny algorytm Euklidesa

Do tej pory poznaliśmy iteracyjną wersję **algorytmu Euklidesa**. Jego wersja z dzieleniem opiera się na równości:

$$\mathrm{NWD}(a,b) = \mathrm{NWD}(b \bmod a, a) \quad \text{dla } a \leqslant b$$

gdzie $b \bmod a$ oznacza resztę z dzielenia liczby b przez liczbę a.

Działanie algorytmu sprowadza się do naprzemiennego wykonywania operacji wyznaczania reszty z dzielenia: $a \bmod b$ dla $a > b$ oraz $b \bmod a$ dla $b > a$, dopóki reszta z dzielenia nie jest równa 0.

Algorytm Euklidesa – wersja z dzieleniem, podręcznik *Informatyka na czasie 2. Zakres podstawowy*, s. 237

Ćwiczenie 6

Zapisz w tabeli przebieg algorytmu Euklidesa z dzieleniem dla liczb 192 i 42.

Algorytm wyznaczania największego wspólnego dzielnika dwóch liczb dodatnich a i b ($a \leqslant b$) możemy zapisać rekurencyjnie, opierając się na następującym spostrzeżeniu: jeśli b można podzielić bez reszty przez a (czyli gdy $b \bmod a = 0$), to wtedy $\mathrm{NWD}(a,b) = a$. W przeciwnym wypadku $\mathrm{NWD}(a,b) = \mathrm{NWD}(b \bmod a, a)$.

Zapiszmy kod źródłowy funkcji `NWDRek`, realizującej rekurencyjny algorytm Euklidesa w wersji z dzieleniem:

Kod źródłowy funkcji `NWDRek` w programie *Euklides rekurencyjnie*

```
1.  def NWDRek(a, b):
2.      if b % a == 0:
3.          return a
4.      return NWDRek(b % a, a)
```

Oto wywołanie programu *Euklides rekurencyjnie*:

```
Wykonywanie: Euklides rekurencyjnie.py
Podaj a: 42
Podaj b: 196
NWD(42, 196) = 14
>>>
                                                    Python
```

Rys. 10.13. Przykładowe wywołanie programu *Euklides rekurencyjnie*

Warto wiedzieć
Wywołanie rekurencyjne funkcji `NWDRek` z parametrami b%a oraz a można sobie wyobrazić następująco: procesor komputera działa na kopii kodu funkcji `NWDRek`, w którym zmiennym a i b są przypisane wartości parametrów, równe liczbowo odpowiednio b%a i a.

Ćwiczenie 7

a. W funkcji `NWDRek` wskaż warunek początkowy oraz instrukcję wywołania rekurencyjnego.
b. Zapisz kod programu, w którym użyjesz funkcji rekurencyjnej `NWDRek`. Uruchom program dla różnych a i b.
c. Wyjaśnij, dlaczego program działa poprawnie również wtedy, gdy początkowa wartość a jest większa od b.

Podsumowanie

- Rekurencja w informatyce to odwołanie się w rozwiązaniu problemu do tego samego problemu, ale dla mniejszego rozmiaru danych.
- Funkcja rekurencyjna musi zawierać dwa warianty kodu: dla warunku początkowego, gdy funkcja zwraca konkretną wartość, oraz dla wywołania rekurencyjnego, w którym złożony przypadek zostaje zredukowany do prostszego.
- Proces rekurencyjnego rozwiązywania problemu składa się z dwóch faz: redukowania rozmiaru problemu (aż do takiego, który można rozwiązać bezpośrednio) oraz zbierania i łączenia wyników częściowych w rozwiązanie wyjściowego problemu.
- Bardzo duża liczba wywołań rekurencyjnych może prowadzić do niewydajnego działania programu. Jeśli dostępna pamięć przydzielona programowi przez system operacyjny zostanie przekroczona, program przestanie działać.

Zadania

* **1** Aby obliczyć kolejną potęgę liczby 2, wystarczy daną potęgę pomnożyć przez 2, tzn. jeśli chcemy obliczyć wartość 2^6, a wiemy, że $2^5 = 32$, to 2^6 obliczymy jako $2 \cdot 32 = 64$. Korzystając z tej obserwacji, napisz funkcję o nagłówku `Potega2(n)`, która będzie obliczać 2^n dla $n > 0$.

* **2** Napisz program obliczający sumę n kolejnych liczb nieparzystych, począwszy od 1. Przygotuj dwie wersje programu: iteracyjną i rekurencyjną.

* **3** Poszukaj informacji o obrazach holenderskiego malarza Pieta Mondriana (1872–1944), który prostokątne tło płótna dzielił rekurencyjnie na coraz mniejsze prostokąty. Jakie było kryterium podziału? Przygotuj szkic przykładowego obrazu.

* **4** Przygotuj prezentację, w której przedstawisz fraktale opisane przez polskiego matematyka Wacława Sierpińskiego: dywan Sierpińskiego oraz trójkąt Sierpińskiego. W dowolnym programie graficznym narysuj fraktale wybranego stopnia i omów etapy ich powstawania.

* **5** Znajdź i popraw błąd w kodzie źródłowym funkcji `Szczescie`, która sprawdza, czy cyfra 8 występuje w danej liczbie. Przetestuj poprawność działania programu.

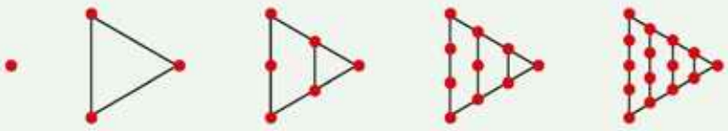
```
1. def Szczescie(liczba):
2.     ostatnia = liczba % 10
3.     if ostatnia == 8:
4.         return True
5.     return Szczescie(liczba // 10)
```

** **6** Na ile sposobów n osób ($n > 0$) może usiąść na n-osobowej ławce? Wzór określający liczbę osób zapisz w postaci rekurencyjnej.

** **7** Algorytm rekurencyjny można zastosować do obliczenia iloczynu liczby naturalnej n przez wszystkie liczby naturalne dodatnie mniejsze od tej liczby. Taką wartość nazywa się silnią liczby n i oznacza jako $n!$. Napisz program obliczający rekurencyjnie wartość $n!$.

** **8** 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.

** **9** 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 z użyciem rekurencji wyznaczy liczbę wszystkich punktów konstrukcji n-tej liczby trójkątnej.

*** **10** Zdefiniuj rekurencyjnie ciąg liczb n-cyfrowych ($n > 4$), w których żadna cyfra 8 nie sąsiaduje bezpośrednio z inną cyfrą 8.
Wskazówka: Ile jest takich liczb pięciocyfrowych, w których żadna cyfra 8 nie sąsiaduje bezpośrednio z inną cyfrą 8? Jak można tę wartość policzyć, gdy wiemy, ile jest takich liczb 4-cyfrowych i 3-cyfrowych?

*** **11** Liczba 145 ma taką własność, że jest równa sumie silni jej cyfr:
$1! + 4! + 5! = 1 + 24 + 120 = 145$
Oprócz liczb 1 i 2 istnieje jeszcze tylko jedna liczba o podobnej własności. Napisz program, który znajdzie tę liczbę.

*** **12** Przeanalizuj poniższy kod źródłowy funkcji obliczającej n-ty wyraz ciągu c_n.

```
1.  def cn(n, iloczyn):
2.      if n == 1:
3.          return iloczyn
4.      return cn(n-1, 3 * iloczyn)
```

Przedstaw przebieg działania algorytmu krok po kroku dla wywołania `cn(10, 3)`.

*** **13** Zdanie palindrom to ciąg znaków, który składa się z tych samych znaków czytany od lewej do prawej i wspak. Pomijamy przy tym spacje i znaki interpunkcyjne. Przykład zdania palindromu to: Ada raportuje, że jutro parada. Napisz funkcję rozstrzygającą, czy ciąg znaków będący jej argumentem jest palindromem. Funkcja powinna zwracać wartość logiczną.

Algorytmika i programowanie w języku Python

Wyobraź sobie, że...



...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 zaszyfrowanej wiadomości SMS wysłanej z bezpiecznego komunikatora.



...rozpoczynasz pracę w firmie produkującej gry planszowe, w dziale odpowiedzialnym za opakowania do tych gier. Już po kilku tygodniach twoją uwagę zwraca **problem optymalizacyjny** występujący w procesie produkcji. Duża część materiału przeznaczonego do wytwarzania pudełek jest marnowana i wyrzucana. Po rozmowie z kierownictwem, korzystając z **podejścia zachłannego**, opracowujesz szczegółowy raport, w którym proponujesz **optymalne rozwiązanie** problemu. Firma wprowadza zmiany w produkcji, co pozwala znacząco obniżyć koszty.



...ukończyłeś studia prawnicze i dostałeś się na staż w znanej kancelarii adwokackiej. Często musisz sięgać po różne akta, które nie są posegregowane. Wiesz, że w uporządkowanym zbiorze łatwiej **wyszukiwać** określone elementy. Przypominasz sobie o algorytmach **sortowania danych** i postanawiasz je wykorzystać. Sortujesz akta według dwóch kryteriów: najpierw chronologicznie, a potem alfabetycznie.



...postanowiłaś zwodowo 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**, o której usłyszałaś na lekcjach informatyki. Zaskakujesz prowadzących serią zdjęć z wykorzystaniem techniki **rekurencyjnej** i lustra.

Projekt zespołowy

Pułapki cyfrowego świata

Rozwój społeczeństw, a szczególnie gospodarki, edukacji i usług publicznych, jest obecnie w dużej mierze możliwy dzięki sprawnemu przepływowi danych w internecie. Dlatego tak ważny stał się problem bezpieczeństwa w sieci. Wiele się o nim mówi zarówno w związku z przesyłaniem danych, jak i zachowaniami użytkowników internetu. W ramach tego projektu zespołowego zbierzecie informacje na wybrany temat, który będzie dotyczył zagrożeń w środowisku usług sieciowych i umiejętności radzenia sobie z nimi. Następnie zorganizujecie dyskusję panelową na ten temat.



Cel projektu

Przygotowanie się do roli ekspertów z wybranego zagadnienia dotyczącego zagrożeń i pułapek pojawiających się podczas korzystania z usług internetowych, a następnie zorganizowanie i przeprowadzenie dyskusji panelowej na ten temat.

To trzeba wiedzieć

Czym jest dyskusja panelowa?

Dyskusja panelowa (zwana też panelem) jest uważana za skuteczny sposób uczenia się, ponieważ łączy przygotowane wypowiedzi ekspertów z możliwością zadawania pytań przez słuchaczy. Najpierw każdy z ekspertów przedstawia swój punkt widzenia na zadany temat w stosunkowo krótkim wystąpieniu. Może ono zawierać także przekaz multimedialny. Po wysłuchaniu wszystkich prezentacji uczestnicy panelu mogą zadawać ekspertom pytania. Całą dyskusję prowadzi moderator.

Specjalistyczne serwisy

Wiarygodne informacje na temat cyberbezpieczeństwa możecie znaleźć w artykułach i raportach zespołu CERT, działającego w strukturach Naukowej i Akademickiej Sieci Komputerowej. Cennym źródłem są też serwisy specjalistyczne takie jak niebezpiecznik.pl czy zaufanatrzeciastrona.pl. Ich twórcy omawiają konkretne przypadki naruszenia bezpieczeństwa, luki w systemach i oprogramowaniu, opracowują także tematyczne podcasty. Warto również sięgnąć po serwisy administracji rządowej oraz bazy aktów prawnych.

Propozycje tematów

1. Jak bezpiecznie kupować w internecie?
2. Czy prywatność jest dobrem luksusowym? Jak ją chronić?
3. Jak bezpiecznie prowadzić edukację zdalną i hybrydową?
4. Biały wywiad. Czy jest się czego obawiać?
5. Jak bezpiecznie kształtować cyfrową tożsamość?
6. Co o mnie wie mój telefon i jak go zabezpieczyć?
7. Jak skutecznie niszczyć dane (nie tylko cyfrowe)?
8. Jak bezpiecznie posługiwać się kartami płatniczymi w Polsce i za granicą?

Etapy projektu

1. Wybierzcie jeden z tematów i podzielcie się na zespoły, w których będziecie go opracowywać.
2. Poszukajcie dodatkowych informacji o tym, czym jest dyskusja panelowa, po co i w jakich sytuacjach się ją stosuje. Wysłuchajcie fragmentu takiej dyskusji na dowolny temat.
3. Ustalcie z nauczycielem, do kogo będzie skierowany panel dyskusyjny. Mogą to być np. wasi rówieśnicy, uczniowie klas pierwszych, nauczyciele lub uczniowie szkoły podstawowej.
4. Przeanalizujcie temat i podzielcie go na mniejsze zagadnienia, a następnie opracujcie je w zespole. Przygotujcie się jak najlepiej, ponieważ jedna osoba z zespołu zostanie ekspertem, któremu słuchacze będą zadawać pytania. Temat możecie opracować w formie mapy myśli.
5. Przygotujcie opisy realnych przykładów naruszeń bezpieczeństwa i przedstawcie kilka różnych scenariuszy rozwoju każdej z takich sytuacji.
6. Wybierzcie wraz z nauczycielem osobę, która będzie moderatorem dyskusji, a z każdego zespołu wyłońcie jednego eksperta. Moderator ustali z nauczycielem zagadnienia oraz czas wystąpień ekspertów.
7. Przygotujcie wskazanego eksperta do wystąpienia na podstawie mapy myśli.
8. Opracujcie materiały, które wzmocnią przekaz podczas panelu, np.:
   - prezentację multimedialną,
   - krótki, prosty film,
   - animację,
   - zestaw dodatkowych rekwizytów.
9. Zorganizujcie w zespole co najmniej jedno spotkanie w trybie zdalnym, które będzie próbnym wystąpieniem eksperta z wykorzystaniem prezentacji multimedialnej lub rekwizytów.
10. Przeprowadźcie spotkanie organizacyjne między ekspertami i moderatorem, aby ustalić zasady panelu.
11. Zastanówcie się, jak możecie udokumentować dyskusję (np. w formie artykułu lub wideo), oraz czy możecie ją odbyć w trybie zdalnym.
12. Przeprowadźcie dyskusję panelową wraz z częścią poświęconą pytaniom od publiczności.
13. Podsumujcie efekty dyskusji.

Przykładowe narzędzia

FreeMind, **Freeplane** – narzędzia do budowy mapy myśli.
Sway – narzędzie do szybkiego tworzenia serwisów WWW oraz prezentacji multimedialnych, ułatwiające łączenie zasobów własnych i dostępnych w sieci.
Microsoft Teams, **Zoom**, **Google Meet** – narzędzia do prowadzenia spotkań zdalnych.
Microsoft PowerPoint, **Keynote** – programy do tworzenia prezentacji w formie slajdów.
OneDrive, **iCloud**, **Google Drive** – dyski wirtualne do przechowywania materiałów.



3 Rozwiązywanie problemów z wykorzystaniem komputera

11. Sterujemy robotem
12. Sztuka publikowania w sieci
13. Grafiki informacyjne

Rozwiązywanie problemów z wykorzystaniem komputera

Pierwsze mechaniczne zabawki pojawiły się już w starożytności. Dziś roboty są ważną częścią codziennego życia – nie tylko zastępują ludzi przy zajęciach monotonnych i niebezpiecznych, lecz także są źródłem rozrywki i pozwalają zdobywać nowe umiejętności.



Robotyce nieodłącznie towarzyszy **mechanika**. Za jej twórcę uznawany jest Archytas z Tarentu, który ok. 420 r. p.n.e. skonstruował drewnianego gołębia, zasilanego sprężonym powietrzem i mogącego przelecieć nawet do 200 metrów.

420 r. p.n.e.



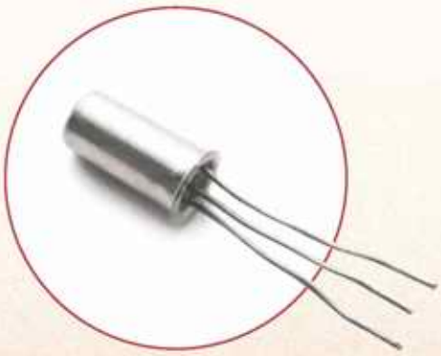
Szwajcarski zegarmistrz i mechanik Pierre Jaquet-Droz skonstruował razem z synami trzy roboty humanoidalne, które nazwał **automatami**. Najbardziej złożonym był Pisarz – składał się z ok. 6 tys. ruchomych elementów. Robot poruszał głową i nadgarstkiem, a tekst, który pisał, można było zaprogramować.

1774 r.

Pierwszą **taśmę montażową** uruchomiono 1913 r. w fabryce samochodów Forda w amerykańskim Detroit. Zmiana organizacji pracy pozwoliła skrócić czas montażu auta z 12,5 do 1,5 godziny. Dzięki temu znacząco spadła cena samochodu i stał się on dobrem masowym.



1913 r.



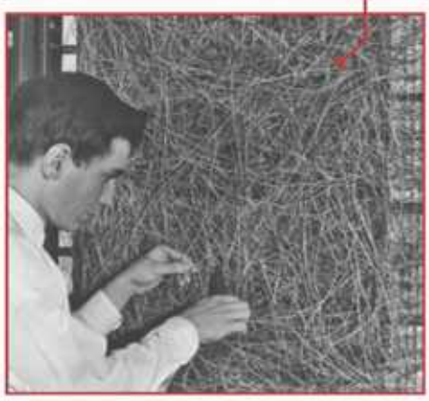
Pierwszy działający **tranzystor** skonstruowali w 1947 r. fizycy pracujący w Bell Laboratories – John Bardeen, Walter Houser Brattain i William Bradford Shockley. Wynalezienie tranzystora, a następnie układów scalonych zapoczątkowało rozwój elektroniki.

1947 r.



W 1950 r. brytyjski matematyk i kryptolog Alan Turing opublikował pionierską pracę dotyczącą sztucznej inteligencji – *Computing Machinery and Intelligence* (*Maszyna licząca a inteligencja*). Turing zaproponował w niej test sprawdzający, czy maszyna może przejawiać inteligentne zachowania tak jak człowiek. Dziś nazywany jest on **testem Turinga**.

1950 r.



W 1957 r. Frank Rosenblatt i Charles Wightman wprowadzili pojęcie **perceptronu**. Jest to prosta sztuczna sieć neuronowa, w której zastosowano proces uczenia się jako metodę programowania systemu. Perceptron był urządzeniem elektromechanicznym i elektronicznym.

1957 r.



Jednego z pierwszych **robotów przemysłowych** opracowali Joseph Engelberger i George Devol. Ramię robotyczne o nazwie Unimate zainstalowano w 1961 r. w zakładach General Motors w New Jersey w Stanach Zjednoczonych.

1961 r.



4 lipca 1997 r. na powierzchni Marsa wylądowała amerykańska sonda kosmiczna Pathfinder. Znajdujący się na jej pokładzie **łazik Sojourner** był pierwszym pojazdem poruszającym się po powierzchni obcej planety, którym zdalnie sterowano z Ziemi.

1997 r.

W 1998 r. odbyła się pierwsza operacja wykonana w całości przez zaprogramowanego do tego **robota medycznego**. Urządzenie składa się z dwóch części – sterowniczej, którą stanowi konsola, oraz wykonawczej, czyli interaktywnych ramion.



1998 r.

W 2012 r. stan Nevada w USA zezwolił na poruszanie się **samochodów autonomicznych** po drogach publicznych. Od 2015 r. w Nevadzie mogą jeździć także autonomiczne samochody ciężarowe.

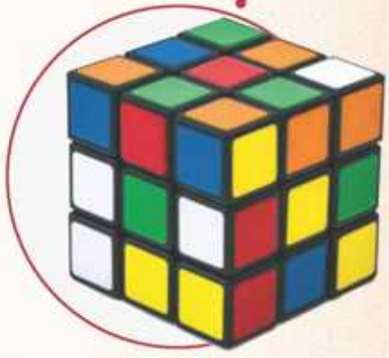


2012 r.



Sophia, humanoidalny robot firmy Hanson Robotics z Hongkongu, to pierwszy **robot**, który uzyskał **obywatelstwo** – w 2017 r. nadała mu je Arabia Saudyjska. Sophia udziela wywiadów na całym świecie, w 2018 r. można było jej wysłuchać w Krakowie.

2017 r.



Głęboka sieć neuronowa, stworzona przez naukowców z Uniwersytetu Kalifornijskiego, **ułożyła kostkę Rubika** trzy razy szybciej niż dotychczasowy rekordzista świata. Sieć neuronowa potrzebowała na to zaledwie 1,2 sekundy.

2019 r.

11. Sterujemy robotem

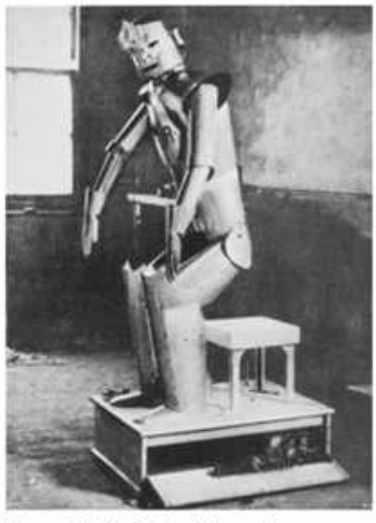
Od wielu lat tematy związane z automatyką i robotyką zyskują na popularności. Uczelnie wyższe wykorzystują roboty na różnych kierunkach studiów, przedsiębiorstwa używają ich m.in. do przyspieszenia pracy, zwiększenia jej precyzji i wydajności oraz do jej automatyzacji. Roboty już od ponad 100 lat nie tylko towarzyszą nam w literaturze i filmie, lecz także pod różnymi postaciami są obecne w naszym życiu codziennym. Ułatwiają wiele prac, a niekiedy wręcz nas wyręczają. W tym temacie nauczysz się programować roboty edukacyjne oraz symulować ich działanie.

Cel lekcji

- Dowiesz się, jak się definiuje roboty.
- Poznasz budowę oraz parametry wybranych robotów edukacyjnych.
- Nauczysz się programować roboty z wykorzystaniem różnych środowisk dostępnych online.
- Przeprowadzisz symulacje zaprogramowanych robotów w środowisku wirtualnym.

11.1. 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 11.1 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. 11.1. 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".

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 mechanicznym. Na podstawie 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.

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.

Robot

11.2. Roboty w edukacji

Dzięki różnorodnym robotom edukacyjnym (rys. 11.2) programowania robotów może się dziś nauczyć niemal każdy, niezależnie od wieku czy etapu edukacji szkolnej.



Rys. 11.2. 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.

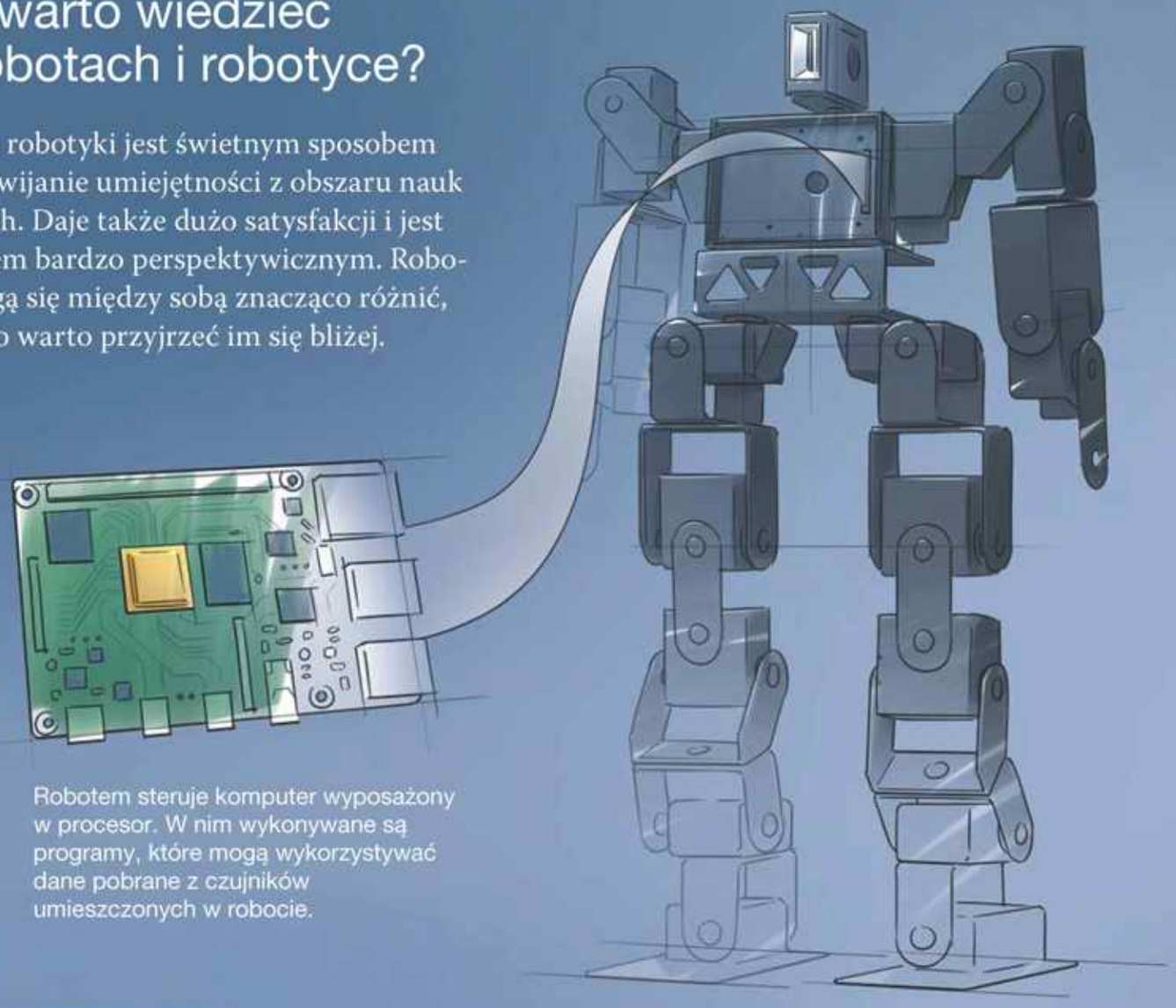
Warto wiedzieć

Popularnymi robotami edukacyjnymi są:

- Edison,
- Dash i Dot,
- mBot,
- Photon,
- Ozobot,
- LEGO MINDSTORMS,
- micro:Maqueen.

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 znacząco 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.

Roboty rozrywkowe

Niektóre roboty dostarczają człowiekowi rozrywki. Mogą mieć różną postać, np. imitować ludzi lub zwierzęta.



Robot zabawka

- **Waga:** 1,6 kg
- **Czas działania:** 1,5 h
- **Liczba kamer:** 1
- **Czujniki:** mikrofon, głośnik, czujnik temperatury

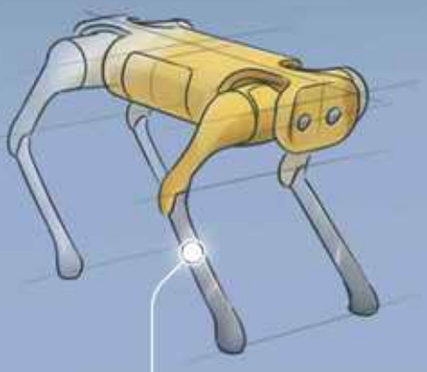


Robot humanoidalny

- **Waga:** 18 kg
- **Udźwig:** 0,6 kg (chwyt dłonią)
- **Liczba kamer:** do 4
- **Czujniki:** widzenie głębi 3D, kamera oka, mikrofon, głośnik, kompas
- Sztuczna skóra twarzy i szyi
- System rozpoznawania i wyrażania emocji

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.



Robot penetracyjny
- **Waga:** 32 kg
- **Maks. prędkość:** 5 km/h
- **Maks. udźwig:** 14 kg
- **Czas działania:** 1,5 h
- **Zakres temperatur:** od –20°C do 45°C



Robot saperski
- **Waga:** 220 kg
- **Maks. prędkość:** 4,8 km/h
- **Maks. udźwig:** 60 kg
- **Czas działania:** 3–4 h
- **Liczba kamer:** 5



Dron
- **Waga:** 4,5 kg
- **Maks. prędkość:** 82 km/h
- **Maks. pułap:** 3 km
- **Maks. czas lotu:** 32 min
- **Liczba kamer:** 1

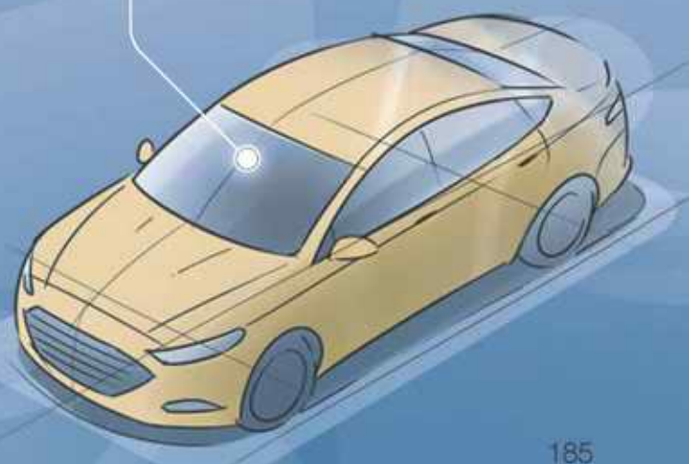
Roboty pomagające człowiekowi

Zadaniem niektórych robotów jest pomoc człowiekowi w codziennych czynnościach, np. opiece nad osobami starszymi lub w przemieszczaniu się.



Robot opiekun
- **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
- **Waga:** 2000 kg
- **Liczba kamer:** 8
- **Czujniki:** 12 czujników ultradźwiękowych, radar, sonar
- **Systemy:** GPS, mapowanie 3D, chmura obliczeniowa, system komunikacji, uczenie maszynowe



11.3. Robot LEGO MINDSTORMS EV3

Warto wiedzieć

Robot EV3 jest następcą zestawu NXT 2.0.

Jednym z popularnych na rynku edukacyjnym robotów jest zestaw LEGO MINDSTORMS EV3, który daje bardzo duże możliwości twórcze. Trzonem tego zestawu jest inteligentna kostka (widoczna na rys. 11.3) – 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. 11.3. 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.

Serwomotor

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:

Czujnik kolorów
- **czujnik kolorów** – mierzy barwę oraz natężenie światła,

Czujnik podczerwieni
- **czujnik podczerwieni** – wykrywa źródła podczerwieni, np. z nadajnika sterującego robotem,

Czujnik żyroskopowy
- **czujnik żyroskoskopowy** – wykrywa ruch obrotowy robota i zmianę jego położenia,

Czujnik dotykowy
- **czujnik dotykowy** – wykrywa naciśnięcie lub zwolnienie przycisku,

Czujnik ultradźwiękowy
- **czujnik ultradźwiękowy** – mierzy odległość od obiektów,

Czujnik temperatury
- **czujnik temperatury** – mierzy temperaturę w zakresie od –20°C do +120°C.

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.

Po uruchomieniu środowiska w systemie Windows i wybraniu opcji **New Program** (rys. 11.4) pojawi się zakładka, w której za pomocą bloków symbolizujących elementy zestawu tworzymy program dla naszego robota. Dodatkowo możemy również zbudować jednego z zaprojektowanych wcześniej robotów według przygotowanej instrukcji (narzędzie **Building Instructions**).



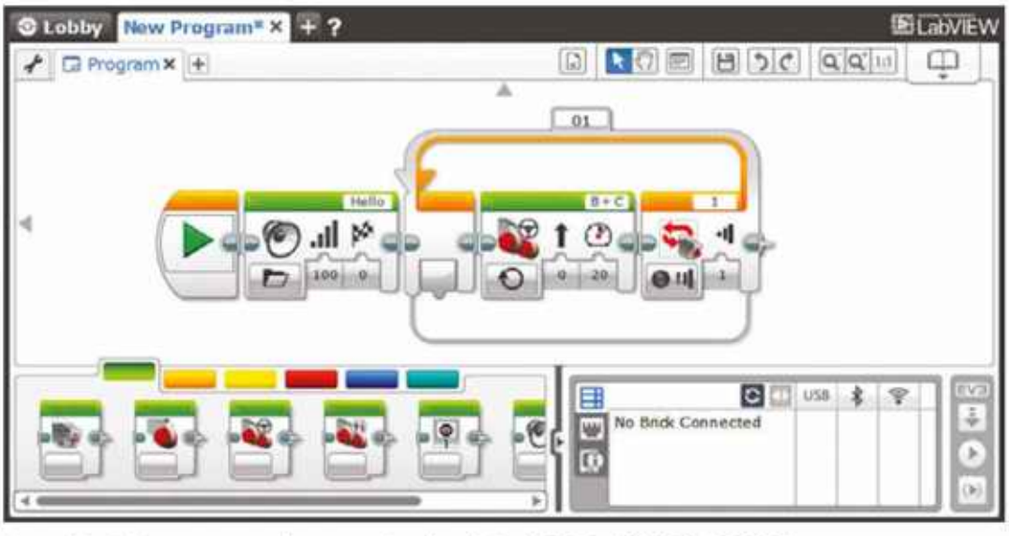
Rys. 11.4. Środowisko LEGO MINDSTORMS

Programowanie robota EV3 w środowisku LEGO MINDSTORMS

Przyjmijmy, że robot ma nam zakomunikować początek wykonywania programu, a następnie poruszać się do przodu, dopóki nie naciśniemy przycisku w podłączonym do robota czujniku dotyku.

Pierwsza instrukcja, z której skorzystamy, będzie odpowiadała za wypowiedzenie komunikatu dźwiękowego, np. „Hello". Następnie użyjemy pętli, aby robot poruszał się do momentu wciśnięcia odpowiedniego czujnika.

Rysunek 11.5 przedstawia przykładowy program.



Rys. 11.5. Program zapisany w środowisku LEGO MINDSTORMS

Dobra rada

Jeśli jesteś użytkownikiem systemu macOS, do zaprogramowania robota EV3 możesz wykorzystać również oprogramowanie LEGO MINDSTORMS. Przypomina ono nieco środowisko Scratch.

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.

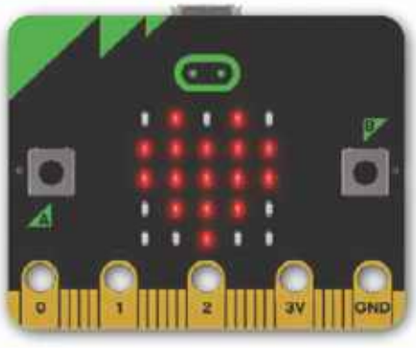
Program składa się z bloku początku i bloku pętli (kolor pomarańczowy) oraz dwóch bloków instrukcji (kolor zielony). Kolejne instrukcje odczytujemy od lewej do prawej. Po przesłaniu programu do kostki EV3 i uruchomieniu go robot przywita się (pierwsza zielona instrukcja), a następnie będzie się poruszał. W pętli wykonywane będą obroty silników (z mocą 20%) podłączonych do portów B i C kostki EV3. Pętla będzie wykonywana, dopóki nie wciśniemy czujnika dotyku podłączonego do portu 1 kostki EV3.

11.4. Mikrokontroler BBC micro:bit

Warto wiedzieć

Kolejna wersja płytki micro:bit o nazwie BBC micro:bit 2 zawiera dodatkowo głośnik i mikrofon oraz przycisk dotykowy. Wyposażona jest w tryb oszczędzania energii i ma większą moc obliczeniową.

Konstruowanie robotów można połączyć z nauką elektroniki. Przykładem takiego rozwiązania jest mikrokontroler BBC micro:bit (rys. 11.6), który miał premierę w 2015 r.



Rys. 11.6. Mikrokontroler BBC micro:bit

Jest to niewielka płytka elektroniczna wyposażona w procesor ARM Cortex-M0, moduły komunikacji Bluetooth i USB, dwa przyciski oraz dwa wbudowane czujniki:

Magnetometr ● ▸ **magnetometr** – czujnik, który działa na zasadzie kompasu, można dzięki niemu wskazać kierunek północny,

Akcelerometr ● ▸ **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).

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. 11.7. Przykład robota micro:Maqueen, wykonanego z użyciem mikrokontrolera micro:bit

Przykładem takiego zastosowania jest robot micro:Maqueen (rys. 11.7), 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

Kod źródłowy można tworzyć w środowisku online opartym na edytorze Mu, dostępnym na stronie python.microbit.org/v/2.0.

Aby zobaczyć działanie napisanego kodu, należy przesłać go na płytkę i uruchomić. Przykładowy kod źródłowy programu wyświetlającego napis „Witaj!" oraz symbol serca przedstawiono na rysunku 11.8.

Dokumentację i dokładny opis poleceń języka MicroPython w języku polskim można znaleźć na stronie:
microbit-micropython.readthedocs.io/pl/latest/index.html

Dobra rada

Kod źródłowy możesz tworzyć również w samym edytorze Mu. Aby to zrobić, wybierz **Tryb** → **BBC micro:bit**.



Rys. 11.8. Programowanie płytki micro:bit w języku MicroPython

11.5. 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.

Omówimy środowiska online, które pozwalają programować roboty oraz symulować ich działanie: Open Roberta Lab, Microsoft Make-Code oraz VEXcode VR.

Warto wiedzieć

Symulatory pozwalają programować i testować roboty w bezpiecznych warunkach, bez obaw o przypadkowe błędy, które mogłyby spowodować uszkodzenia sprzętu.

A to ciekawe

Roboty w muzyce



Niemiecki zespół muzyczny Kraftwerk jest uznawany za prekursora muzyki elektronicznej. Muzycy od lat 70. konstruowali własne elektroniczne instrumenty i komponowali na nich swoje utwory. W roku 1978 wydali album *Die Mensch-Maschine*, który odwoływał się do relacji człowiek–maszyna. W utworze *Die Roboter* robotyczny głos z syntezatora dźwięku wypowiada po rosyjsku „Я твой слуга, я твой работник" („Jestem twoim sługą, jestem twoim pracownikiem"). Na koncertach zespołu stale pojawiają się robotyczne sylwetki muzyków.

11.6. Symulacja robota EV3 w środowisku Open Roberta Lab

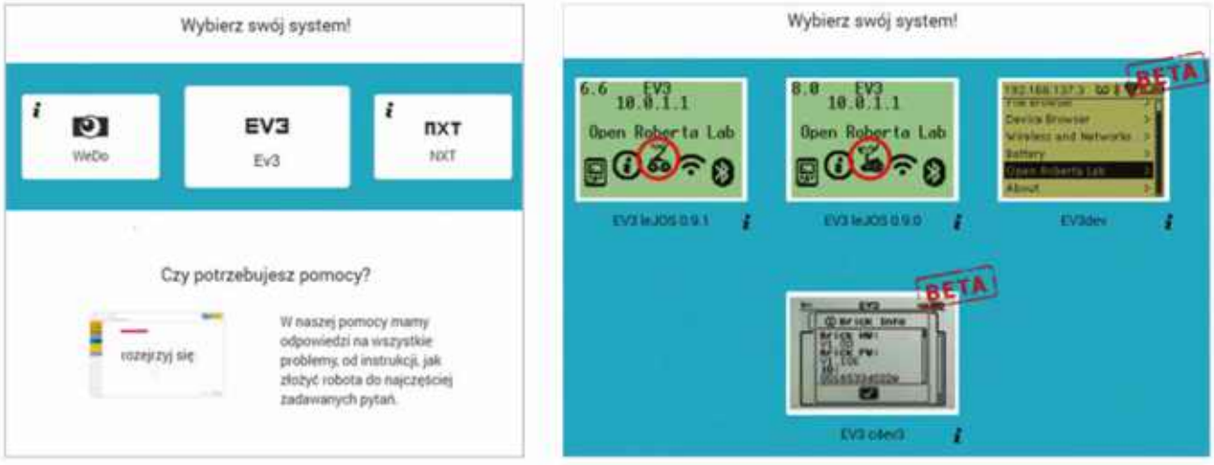
Warto wiedzieć

Okna symulacji robotów są dostępne tylko dla niektórych modeli dostępnych 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.

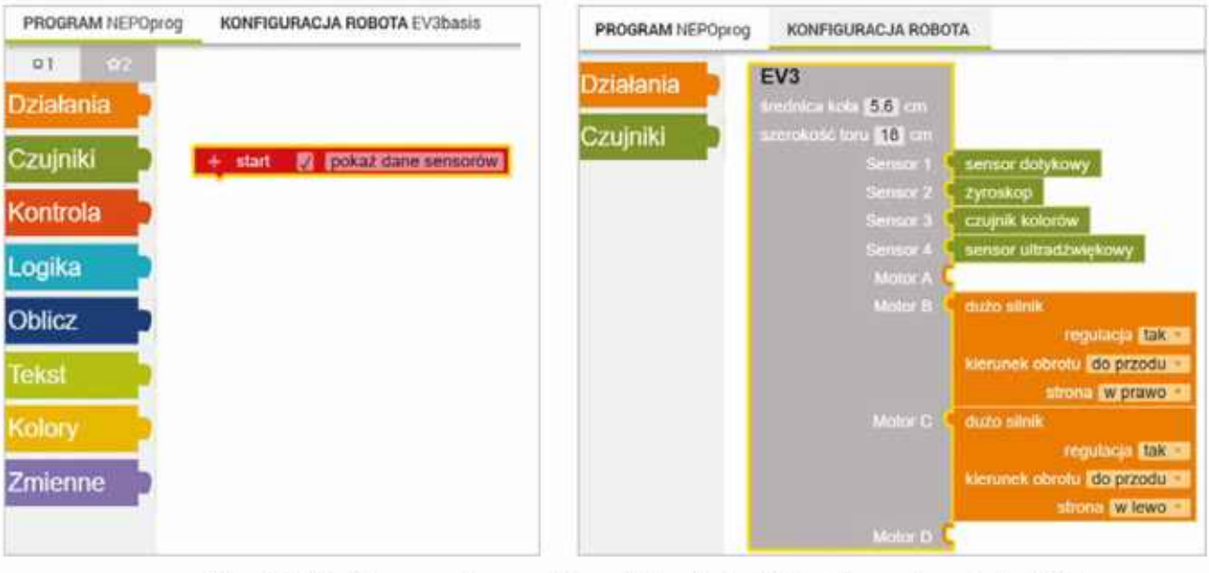
Programowanie odbywa się za pomocą graficznego interfejsu, w którym program układa się z dostępnych bloków. W zależności od wybranego robota można również podejrzeć kod przetłumaczony m.in. na język Python. Utworzony program można następnie zaimplementować w robocie lub przetestować w oknie symulatora. Serwis Open Roberta Lab jest dostępny w języku polskim.

Po uruchomieniu strony projektu lab.open-roberta.org wybieramy robota oraz system operacyjny. W naszym wypadku wybierzmy robota EV3 i system EV3 c4ev3 (rys. 11.9).



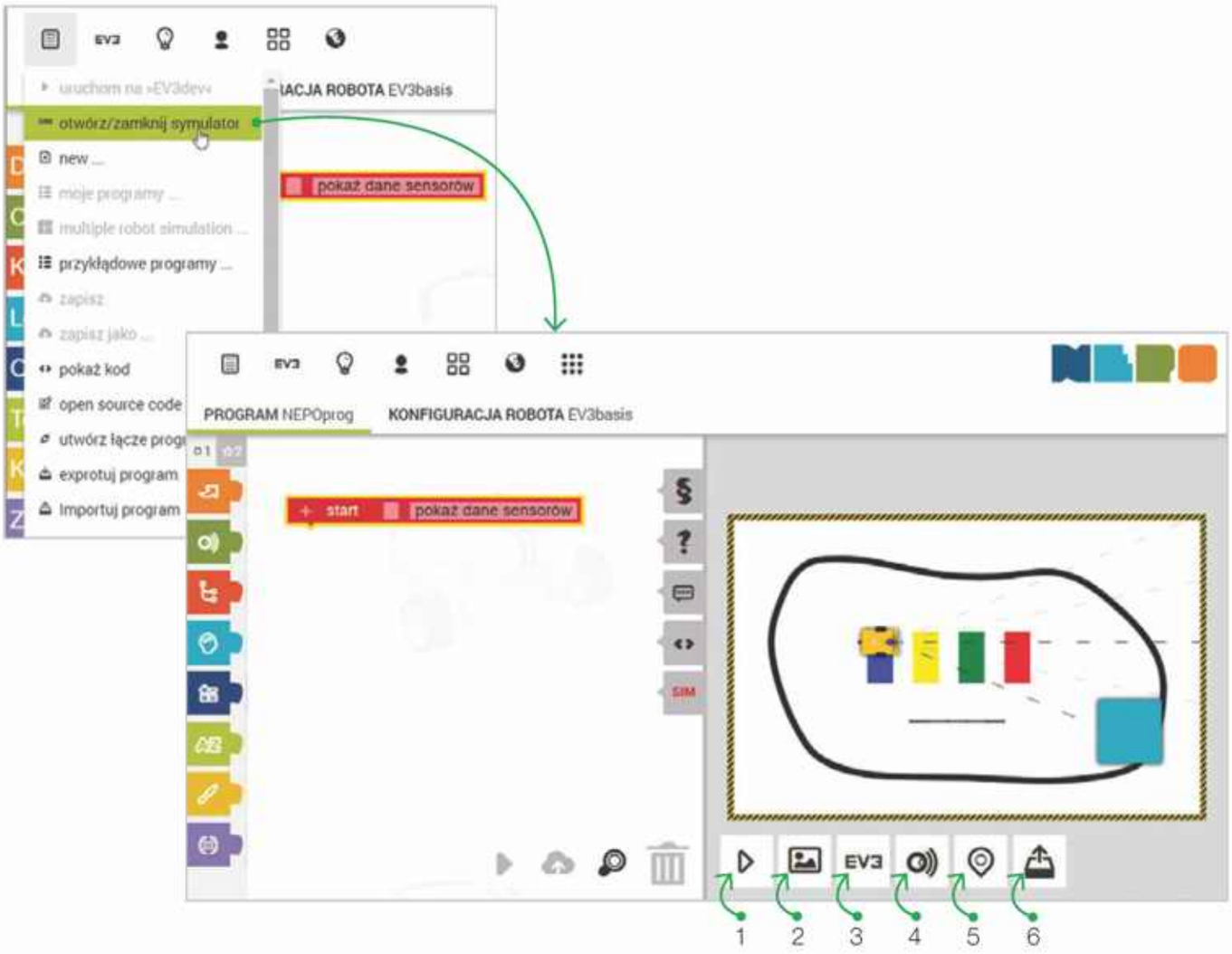
Rys. 11.9. Wybór robota EV3 w środowisku Open Roberta Lab

Rysunek 11.10 przedstawia okno z obszarem roboczym (zrzut z lewej) oraz okno z podglądem konfiguracji robota EV3 (zrzut z prawej).



Rys. 11.10. 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. 11.11).



Rys. 11.11. Włączanie podglądu, w którym można symulować działanie robota

W oknie symulatora znajduje się kilka przycisków – na rysunku 11.11 oznaczone są one kolejnymi numerami od 1 do 6. Przyciski pozwalają na:

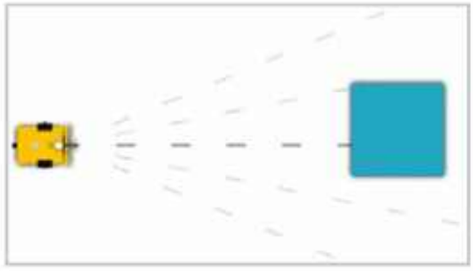
- uruchomienie programu (1),
- zmianę tła sceny (2),
- podgląd programowalnej kostki zestawu EV3 (3),
- podgląd wartości poszczególnych czujników (4),
- umieszczenie robota w punkcie początkowym (5),
- wgranie własnego tła (6).

Program *Zatrzymanie przed przeszkodą*

Zaprogramujemy teraz 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 2 z rys. 11.11, s. 191) na takie, w którym robot jest ustawiony w pewnej odległości przed przeszkodą (rys. 11.12). Linie przerywane obrazują działanie **czujnika ultradźwiękowego**.

Czujnik ultradźwiękowy, s. 186



Rys. 11.12. 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. Poruszanie się robota do przodu będzie wypadkową takiej samej prędkości robota w lewo i w prawo jednocześnie.

Kod źródłowy programu *Zatrzymanie przed przeszkodą* przedstawiono na rysunku 11.13.

Kod źródłowy programu *Zatrzymanie przed przeszkodą*



Rys. 11.13. Kod źródłowy programu *Zatrzymanie przed przeszkodą*

Na rysunku 11.14 przedstawiono wynik symulacji – pogrubiona linia pokazuje tor ruchu robota, a linie wychodzące od robota obrazują działanie sensora ultradźwiękowego.



Rys. 11.14. Efekt symulacji działania robota w programie *Zatrzymanie przed przeszkodą*

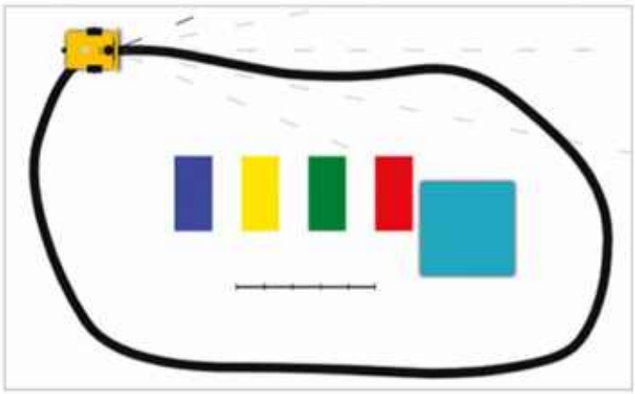
Ćwiczenie 1

a. Zapisz w środowisku Open Roberta Lab kod źródłowy programu *Zatrzymanie przed przeszkodą* i przeprowadź symulację robota.
b. Zmodyfikuj program *Zatrzymanie przed przeszkodą* tak, aby po zatrzymaniu robot dwukrotnie wypowiedział słowa „Poprawne zatrzymanie".

Program *Jazda po ścieżce*

Wykonamy kolejne zadanie: robot będzie jeździł po wyznaczonej ścieżce narysowanej linią ciągłą.

Ustawiamy odpowiednie tło (rys. 11.15) i przesuwamy przeszkodę tak, aby nie kolidowała z robotem podczas jazdy po linii. Robota umieszczamy na czarnej linii.



Rys. 11.15. Ustawienie robota na 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.

W rozwiązaniu możemy wykorzystać **czujnik kolorów**, który pozwoli skorygować tor ruchu robota, gdyby wyjeżdżał poza czarną linię. Rysunek 11.16 przedstawia zapis kodu źródłowego programu *Jazda po ścieżce*.

Czujnik kolorów, s. 186



Rys. 11.16. Kod źródłowy programu *Jazda po ścieżce*

Kod źródłowy programu *Jazda po ścieżce*

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**.

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.



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.

11.7. Symulacja płytki micro:bit w środowisku Microsoft MakeCode

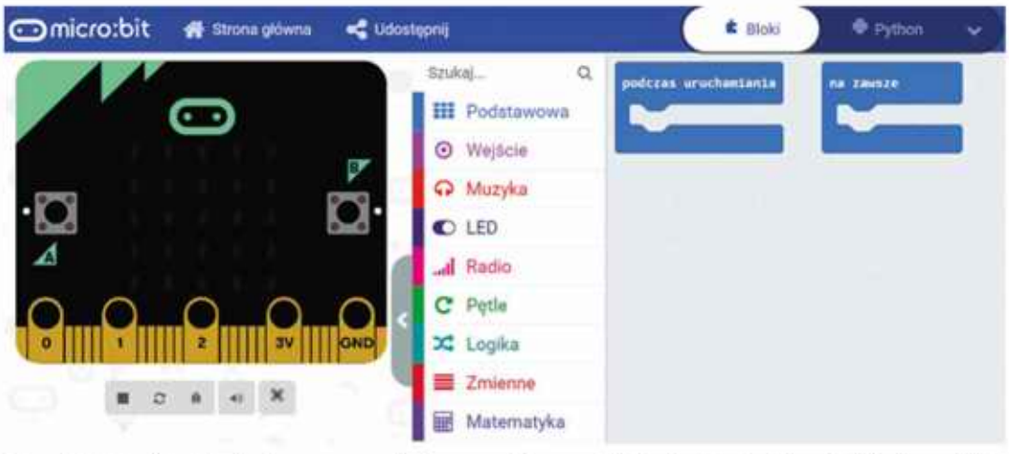
Warto wiedzieć

Środowisko MakeCode obsługuje wiele robotów i innych projektów edukacyjnych, w tym:

- LEGO MINDSTORMS EV3,
- Circuit Playground Express,
- Minecraft,
- Arcade,
- Cue,
- Chibi Chip.

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 11.17 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 charakterystycznymi diodami LED i dwoma przyciskami oznaczonymi literami A i B.



Rys. 11.17. Środowisko programistyczne Microsoft MakeCode dla płytki micro:bit

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.

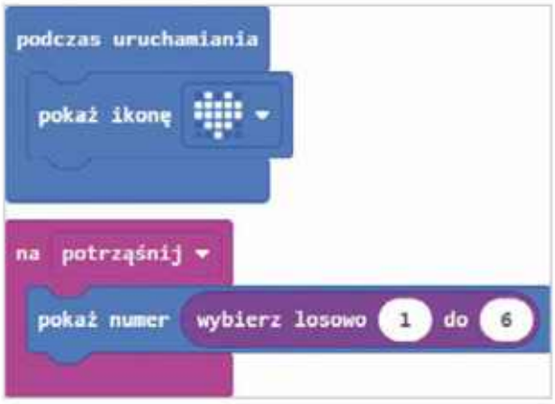
W tym środowisku można wygodnie tworzyć kod źródłowy za pomocą bloków, które zawierają gotowe instrukcje. Pozwalają one wyświetlać m.in. zdefiniowane wcześniej symbole lub napisy, które powstają poprzez zaświecenie się wybranych diod. Można również samodzielnie tworzyć symbole według własnych projektów.

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.

Rysunek 11.18 przedstawia kod źródłowy programu *Rzut kostką*.

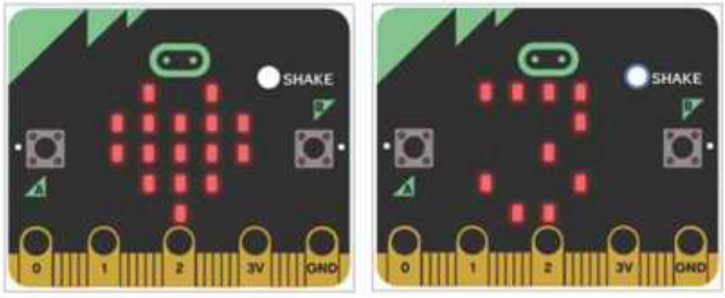
Kod źródłowy programu *Rzut kostką*



Rys. 11.18. 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. 11.19).



Rys. 11.19. 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

Imię dla robota



Amerykańska agencja kosmiczna NASA co jakiś czas ogłasza konkursy na nazwy marsjańskich łazików. Dotychczasowe nazwy zostały wymyślone przez uczniów. Łazik Sojourner z 1997 r. został nazwany przez 12-letnią Vallery Ambroise, która upamiętniła w ten sposób Sojourner Truth, afroamerykańską działaczkę na rzecz praw kobiet i zniesienia niewolnictwa. Łaziki z 2003 r. – Spirit i Opportunity – nazwała 9-letnia Sofi Collins, a Curiosity z 2011 r. – 12-letnia Clara Ma.

11.8. 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.

Dobra rada

W trakcie tworzenia programu z bloków możesz przełączyć się na tryb edycji kodu 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. 11.20) umieszczono następujące narzędzia: zmiana języka, menu **Plik**, samouczki, zmiana tła (narzędzie **Boisko**) oraz narzędzia do sterowania wykonywaniem programu. Poniżej znajdują się: opcja przełączania kodu blokowego na kod języka Python oraz kontrola wartości zmiennych i czujników.



Rys. 11.20. Narzędzia środowiska VEXcode VR

W prawej części okna znajduje się panel **Kod**. Zawiera on bloki instrukcji, z których można tworzyć program sterujący robotem. Fragment okna przedstawia rysunek 11.21.



Rys. 11.21. Panel **Kod** zawierający instrukcje sterujące robotem

Warto wiedzieć

Na niektórych planszach (tłach w narzędziu **Boisko**) dostępna jest jeszcze trzecia kamera, pokazująca obraz, który widzi robot.

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. 11.22).

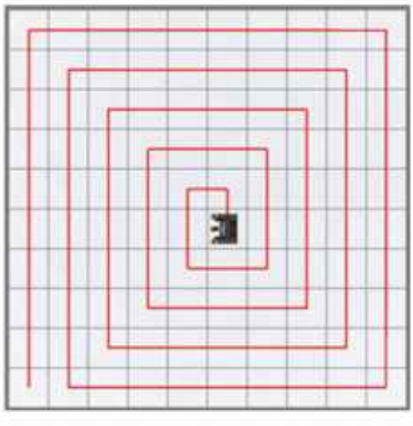


Rys. 11.22. Okno symulacji robota

Narzędzie **Boisko** (ang. *Playground*) 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 11.23.



Rys. 11.23. 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.

Pętla, s. 247–248, 255

Na rysunku 11.24 przedstawiono trasę robota. Przez pierwsze 3 odcinki robot pokona 9 pól, następnie długość dwóch kolejnych odcinków zmniejszy się do 8 pól, następnych dwóch odcinków do 7 pól itd.

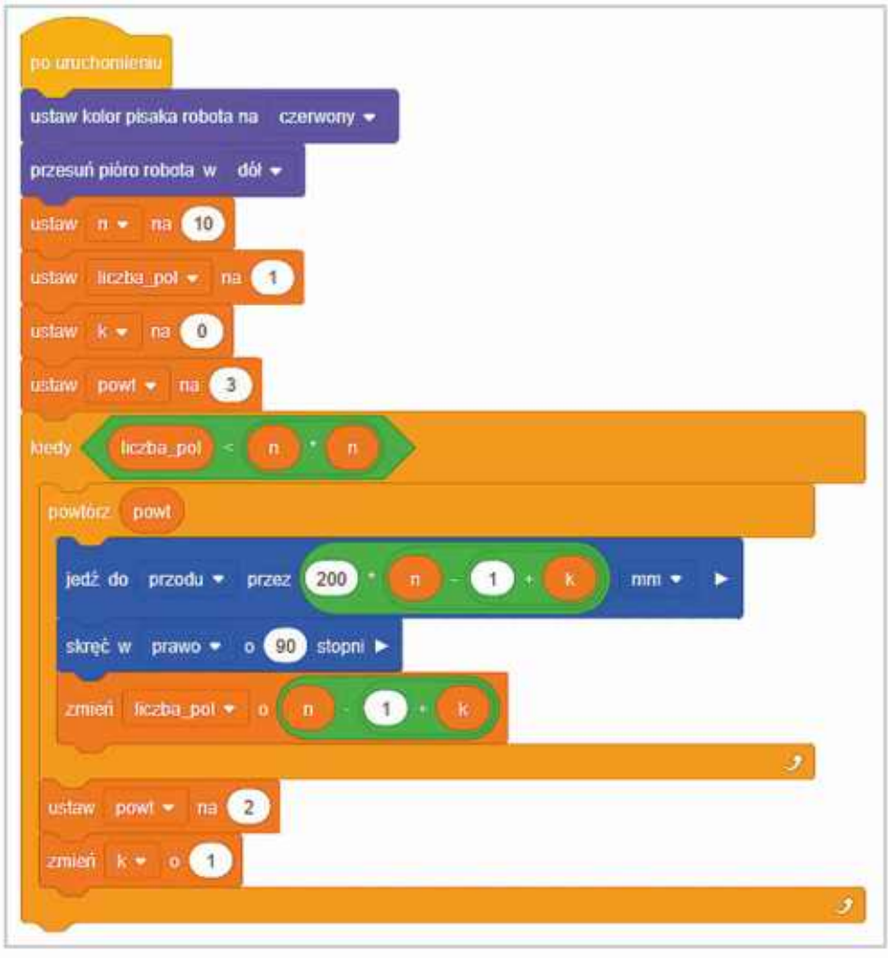


Rys. 11.24. Zmiany długości przemierzanych odcinków

Pętle zagnieżdżone, s. 26, 112

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 11.25 przedstawia kod źródłowy programu *Wszystkie pola planszy*.

Kod źródłowy programu *Wszystkie pola planszy*



Rys. 11.25. 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.

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 **Labirynt ze ścian**.

Rysunek 11.26 przedstawia tę planszę w dwóch widokach: z kamery znajdującej się za robotem oraz z kamery znajdującej się nad planszą.

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.

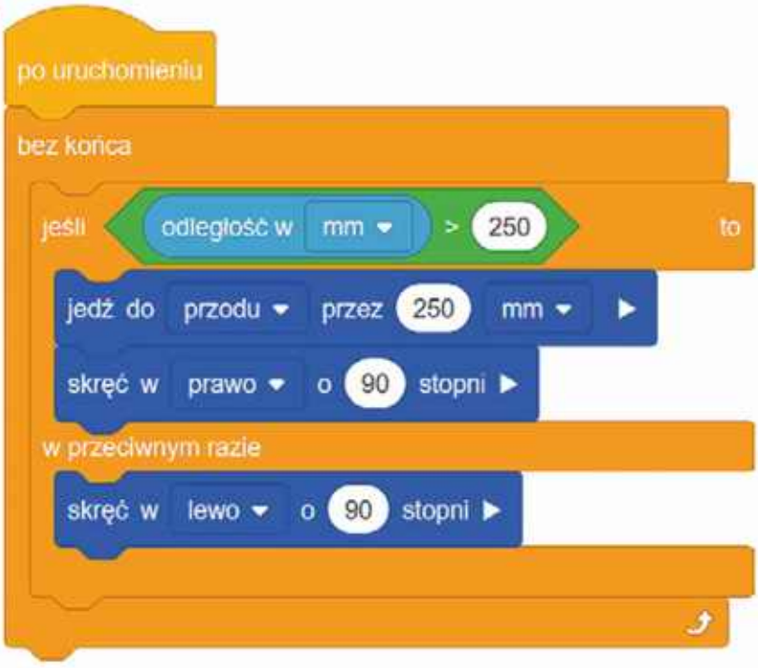


Rys. 11.26. Plansza **Labirynt ze ścian** – widok z góry i zza robota

Z rysunku 11.26 na s. 199 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 11.27 przedstawiono zapis kodu źródłowego programu *Wyjście z labiryntu*.

Kod źródłowy programu ● *Wyjście z labiryntu*



Rys. 11.27. 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.

Warto wiedzieć

Problem poszukiwania wyjścia z labiryntu można rozwiązać między innymi z wykorzystaniem algorytmów: Minty'ego, Bellmana–Forda, Dijkstry oraz Dantziga.

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.

Podsumowanie

- Robot to 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.
- Roboty można programować w wielu językach, np. za pomocą bloków oraz języków Python i C++.
- Środowiska symulacyjne (np. VEXcode VR, Microsoft MakeCode lub Open Roberta Lab) pozwalają na poznawanie robotów, programowanie i obserwowanie ich działania. Niektóre oferują widok trójwymiarowy, dzięki któremu można dokładnie zarejestrować pracę robota.

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 na s. 197–199.

* **2** Zaproponuj, opisz i wykonaj prezentację na temat wybranego zagadnienia, które można rozwiązać za pomocą jednego lub wielu robotów.

* **3** Zademonstruj możliwości wybranego robota. Opisz i zaprezentuj zastosowania dostępnych czujników i silników.

** **4** 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.

** **5** Zaprogramuj robota muzycznego, który będzie tworzył własną muzykę, np. z wykorzystaniem akcelerometru.

** **6** Zaprogramuj robota, który będzie służył jako kontroler do sterowania w grze komputerowej.

** **7** 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.

*** **8** Zaprogramuj robota (lub mikrokontroler) tak, aby za jego pomocą pokazać działanie wybranych praw fizyki, np. siły odśrodkowej.

*** **9** Zaprogramuj robota tak, aby pokonał dowolny labirynt.

*** **10** W kilkuosobowej grupie zaprogramujcie wybranego robota poruszającego się po wyznaczonej ścieżce z możliwością omijania przeszkód na drodze.

12. Sztuka publikowania w sieci

Nasze działania w różnych dziedzinach życia coraz częściej mają miejsce w sieci, a czasem muszą wręcz ograniczyć się tylko do niej. Dlatego szczególnie ważna staje się umiejętność szeroko rozumianego publikowania w internecie – od komunikacji z przyjaciółmi po przygotowywanie materiałów na potrzeby nauki, pracy czy hobby. Z tego tematu dowiesz się, jak tworzyć zrozumiałe, atrakcyjne i profesjonalne treści, które mają szansę zaistnieć w sieci.

Cele lekcji

- Nauczysz się opracowywać interesujące dla odbiorców treści internetowe.
- Poznasz narzędzia, które mogą pomóc w przygotowaniu publikacji wzbogaconych o grafikę, nagrania audio lub filmy.
- Zrozumiesz pojęcia: webcast, webinarium, screencast i podcast.
- Nauczysz się przygotowywać do wystąpienia przed kamerą i mikrofonem.
- Dowiesz się, jakie narzędzia pozwalają monitorować i analizować ruch w internecie.

Warto wiedzieć

Nad zróżnicowaniem formy przekazu warto się zastanowić w przypadku serii publikacji, które będą się pojawiać cyklicznie. Wówczas formę można dopasować do aktualnego tematu.

Twórcy internetowi najczęściej wybierają jedną główną formę publikowania, np. teksty na blogu, filmy w serwisie wideo lub audycje tylko do słuchania. Niezależnie jednak od tego, jaką podstawową formę przekazu wybierzemy, przyda się umiejętność tworzenia pozostałych. Publikacje filmowe wymagają bowiem np. przygotowania tekstów zapowiadających ich premierę, a także komunikatów (tekstu lub grafiki) do zamieszczenia w sieciach społecznościowych.

Ćwiczenie 1

Zastanów się, co przyciąga twoją uwagę w sieci. Omówcie to w grupie i wymieńcie najważniejsze czynniki. Następnie zastanówcie się, co sprawia, że:

a. wracacie do tego samego miejsca w sieci – witryny lub kanału,
b. przestajecie odwiedzać daną witrynę lub kanał.

12.1. Jak zainteresować odbiorców swoimi publikacjami?

Gdy przeglądamy informacje w sieci, pewne czynniki decydują, że przy jednych treściach zatrzymujemy się na dłużej, by poznać szczegóły, a w wypadku innych przeciwnie – przechodzimy dalej. Podobnie jest z całymi serwisami internetowymi.

Zastanówmy się, które elementy sprawiają, że publikowane w internecie treści są atrakcyjne i przyciągają uwagę odbiorców.

Treści dopasowane do odbiorców

Przygotowanie treści dopasowanych do odbiorcy, jego potrzeb i oczekiwań jest w dzisiejszych czasach koniecznością – bez tego nie uda się wzbudzić szerszego zainteresowania.

Zauważ, że inaczej należy przygotować np. zaproszenie do obejrzenia filmu dla rówieśników, a inaczej dla nauczycieli czy specjalistów w danej dziedzinie. Trzeba również odpowiednio dobrać konstrukcję samej publikacji, język przekazu oraz warstwę graficzną. Nie bez znaczenia jest wiek i wykształcenie odbiorcy.

Warto wiedzieć

Tworzenie prasy skierowanej do czytelników z niższym wykształceniem wymaga odpowiedniego dostosowania tekstu. Kiedyś redakcje zatrudniały osoby o takim profilu w roli testerów, którzy oznaczali niezrozumiałe dla nich fragmenty tekstu. Z czasem jednak redaktorzy uczyli się pisać wystarczająco przystępnie, a testerzy tracili pracę.

Ćwiczenie 2

Opisz krótko swoją ulubioną grę, książkę lub film w sposób dostosowany do odbiorców w wieku 70 lat oraz 10 lat.

Co przyciąga uwagę użytkowników?

Oto wybrane elementy, które przyciągają uwagę odbiorców podczas przeglądania informacji w internecie.

- Tytuły (nagłówki) oraz **lidy** (ang. *lead*), czyli krótkie zapowiedzi treści. Wzbudzające emocje i intrygujące tytuły zachęcają do przeczytania, obejrzenia lub odsłuchania całości materiału.
- Dobrze skonstruowane teksty informacyjne – wprowadzenia, streszczenia, zaproszenia oraz zapowiedzi, np. w mediach społecznościowych.
- Ilustracje – wszelkiego rodzaju grafiki związane z danym tematem, włącznie z banerami, oraz pełniące rolę informacyjną, np. odpowiedniki plakatów filmowych.
- Właściwy dobór słów, które przybliżą użytkownikowi zagadnienie, ale również będą odpowiadały zapytaniom w wyszukiwarkach internetowych. W ten sposób można zwiększać szansę na to, że naszą publikację znajdą nowi użytkownicy, bo pojawi się wysoko w **organicznych wynikach wyszukiwania** (ang. *organic search*), czyli takich, które nie są wynikiem opłaconych kampanii.
- **Hashtagi**, czyli krótkie teksty poprzedzone znakiem #, które pełnią funkcję znacznika treści. Hashtagi pozwalają uporządkować pokrewne publikacje, a niekiedy pomagają różnym inicjatywom zyskać rozgłos.

Lid

Warto wiedzieć

Tytuł publikacji, który ma przyciągnąć uwagę, nie musi ściśle odpowiadać jej tematowi. Na przykład artykuł o inteligencji szczurów może mieć tytuł: *Dlaczego tak trudno pozbyć się szczurów?*

Organiczne wyniki wyszukiwania

Hashtag

Ważnym elementem, który może zwrócić uwagę internautów na publikację, są również reklamy. Najczęściej mają one postać graficzną lub tekstową. Warto pamiętać o dwóch ważnych zasadach. Po pierwsze – reklama nie może być zbyt nachalna. Nikt nie lubi często wyskakujących banerów. Po drugie – reklama nie może oszukiwać.

Tworzenie tytułów zachęcających do kliknięcia, ale mylących co do zawartości jest niedopuszczalne. Takie działanie nazywane jest często **clickbaitem** (od ang. *to click* – klikać, *bait* – przynęta) i jest bardzo negatywnie odbierane przez użytkowników.

Clickbait

Jak przygotować treść publikacji

Aby sprawnie przygotować publikację, która zainteresuje wiele osób, warto odpowiednio zorganizować sobie pracę.

1 Zrób plan

Znajdź pomysł na temat publikacji i zaplanuj jej zawartość.

Publikacja tekstowa

- konspekt z punktami do opracowania
- opis grafik ilustrujących tekst

Nagranie wideo lub występ na żywo

- scenariusz lub scenorys z opisem scen i ujęć
- tekst lektora
- opis potrzebnych grafik

2 Poddaj plan krytycznej analizie

Przygotowany plan warto przeanalizować i odpowiedzieć sobie na kilka pytań:



Odpowiedzi na te pytania powinny skłonić do wprowadzenia niezbędnych poprawek.

3 Ustal objętość publikacji

- Im dłuższy materiał, tym więcej czasu zajmie jego przygotowanie.
- Pamiętaj, że wraz ze wzrostem objętości publikacji spada uwaga odbiorców. Publikacja powinna omawiać wszystkie istotne zagadnienia, ale nie powinna być zbyt obszerna.







Unikaj błędów

- Przed publikacją sprawdź: imiona i nazwiska, nazwy, adresy, daty, dane teleadresowe itp.
- Nie używaj wyrażeń typu „dziś", „wczoraj", „w zeszłym tygodniu", bo nie wiesz, kiedy odbiorca zapozna się z publikacją.

4

Wymyśl tytuł

- Dobry tytuł powinien przyciągać uwagę i zachęcać do zapoznania się z treścią.
- Sformułuj tytuł tak, by łatwo było go używać w różnych celach, m.in.: w zapowiedziach, na banerach, plakatach, gadżetach, planszach początkowych i końcowych.
- Tytuł powinien brzmieć tak samo niezależnie od miejsca użycia – nawet drobne odstępstwa mogą dezorientować odbiorców.

5

Wykonaj plan

- Podczas tworzenia publikacji trzymaj się przygotowanego konspektu lub scenariusza.
- Wprowadź modyfikacje, jeśli ulepszą one materiał i sprawią, że będzie lepiej odpowiadał potrzebom odbiorców.
- Opublikuj materiał zgodnie z zapowiedziami. Możesz skorzystać z systemów, które automatycznie publikują treści o określonej porze.

Śledź reakcje i wprowadzaj poprawki

- Czytaj komentarze i utrzymuj kontakt z odbiorcami.
- Staraj się zrozumieć grono swoich odbiorców – pamiętaj, że może się ono zmieniać.
- Analizuj trendy w sieci.
- Dostosowuj nowe publikacje do oczekiwań odbiorców i aktualizuj starsze treści.



Jak utrzymać uwagę odbiorców?

Zastosowanie kilku wymienionych poniżej zasad może sprawić, że odbiorcy nie tylko zapoznają się z przygotowanym przez nas materiałem, lecz także będą wracać do naszego serwisu.

- Twórz interesujące i użyteczne treści, dostosowane do potrzeb i oczekiwań odbiorcy pod względem zawartości i języka przekazu.
- Aktualizuj treści – dbaj o to, aby nawet starsze materiały zawierały aktualne odnośniki, sprawdzaj, czy wszystko działa prawidłowo, oznaczaj nieaktualne treści jako archiwalne lub usuwaj je z serwisu.
- Bierz odpowiedzialność za słowo – publikuj informacje sprawdzone w rzetelnych źródłach, konsekwentnie realizuj zapowiadane działania, zrezygnuj z publikowania treści, których nie rozumiesz.
- Publikuj systematycznie, zwłaszcza na blogach i w serwisach z poradami. Jest to obowiązek, jeśli tworzysz coś w formie serii złożonej z wielu odcinków (części).
- Monitoruj ruch w sieci i **poznawaj własnych odbiorców** – pozwoli to na lepsze dopasowanie treści do ich potrzeb.
- Komunikuj się z odbiorcami – odpowiadaj na ich pytania i uwagi, zachęcaj do wykonania określonych działań (np. udostępniania lub zakupu), badaj preferencje użytkowników (np. za pomocą ankiet) umożliwiaj dyskusję, organizuj konkursy.

Dobra rada

Ponieważ filmy trudniej aktualizować niż teksty, możesz w zamian przygotować odpowiednie adnotacje, np. w opisie filmu.

Jak poznać (lepiej) swojego odbiorcę, s. 217

Ćwiczenie 3

Przypomnij sobie, na czym polega **storytelling**. Zastanów się, czy i jak można go zastosować do treści publikowanych w internecie.

Storytelling, podręcznik *Informatyka na czasie 1. Zakres podstawowy*, s. 109

Zapamiętaj

Publikacja powinna uwzględniać potrzeby i zainteresowania odbiorców. Należy zadbać o to, by materiał był interesujący, przejrzysty, kompletny oraz treściwy, a informacje w nim zawarte były sprawdzone w rzetelnych źródłach.

A to ciekawe

Czym jest content marketing?

Content marketing to jedna z popularnych i efektywnych form marketingu, którą powszechnie wykorzystuje się w internecie. Publikowanie interesujących i przydatnych treści (np. artykułów, webinariów, infografik) przyciąga klienta na stronę firmy i potwierdza, że pracują w niej eksperci w danej dziedzinie. Dzięki temu łatwiej jest przekonać użytkownika np. do zakupów.



12.2. Teksty informacyjne

Teksty opracowuje się nie tylko na potrzeby publikacji ściśle tekstowych, lecz także podczas tworzenia filmów, podcastów czy webinariów. Najczęściej są to **teksty informacyjne**, czyli takie, których głównym zadaniem jest skuteczne przekazanie informacji. Z tego powodu zazwyczaj są pozbawione wszelkich środków artystycznych, cechujących np. teksty literackie.

Tekst informacyjny

Przygotowanie do pisania

Przed rozpoczęciem pisania należy się zastanowić, jakiego rodzaju tekst informacyjny będzie nam potrzebny. Może to być m.in.:

- zaproszenie, np. do obejrzenia filmu lub wysłuchania **podcastu**,
- zapowiedź wydarzenia,
- opis audycji,
- relacja z wydarzenia,
- tekst do działu „o nas" na stronę internetową lub blog.

Podcast, s. 211

Każdy z wymienionych typów ma inną charakterystykę i zazwyczaj zawiera nieco inne informacje. Zanim zaczniemy pisać, musimy ustalić, jaka jest najważniejsza informacja do przekazania, oraz przemyśleć konstrukcję tekstu.

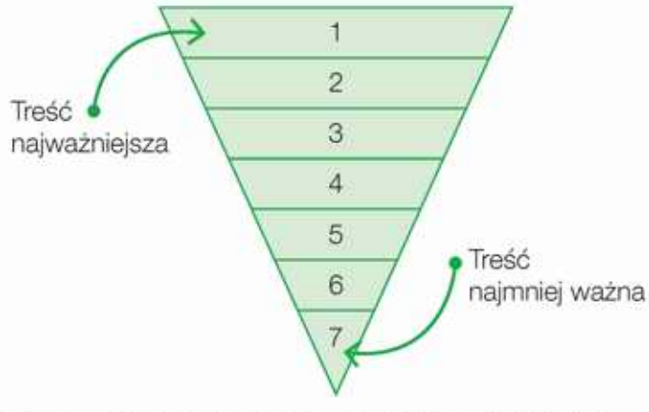
Nawet w wypadku krótkiego tekstu warto przygotować **konspekt**, czyli zarys przedstawiony w punktach. Powinny się w nim znaleźć zagadnienia, które chcemy uwzględnić. Dzięki temu podczas pisania o niczym nie zapomnimy.

Konspekt

Dobra rada

Stworzenie konspektu da ci jeszcze jedną korzyść – liczba pozostałych do opracowania punktów będzie przypominała o ilości pracy do wykonania oraz skłaniała do zwartego opisywania punktów.

Punkty w konspekcie warto na bieżąco ustawiać w formie odwróconej piramidy (rys. 12.1). Najważniejsze treści, które opiszemy najszerzej, umieszczamy na początku, a najmniej ważne, którym poświęcimy mniej miejsca – na końcu.



Rys. 12.1. Konstrukcja konspektu w formie odwróconej piramidy

Dzięki zapisaniu haseł z konspektu w formie odwróconej piramidy już w trakcie tworzenia tekstu zorientujemy się, jaka będzie jego objętość i z jakich treści możemy zrezygnować, aby nadal przekazać wszystkie najważniejsze informacje.

Tworzenie komunikatu tekstowego

Zasada 5W

W napisaniu tekstu może pomóc nam tzw. **zasada 5W**. Jej nazwa pochodzi od pięciu pytań w języku angielskim zaczynających się na literę „w". Należy je sobie postawić w odniesieniu do informacji, którą chcemy przekazać:

1. *Who?* – Kto?
2. *What?* – Co?
3. *Where?* – Gdzie?
4. *When?* – Kiedy?
5. *Why?* – Dlaczego (po co)?

Warto wiedzieć

Zasada 5W jest powszechnie stosowana podczas przygotowywania informacji dziennikarskiej.

Jeśli odpowiemy w tekście na powyższe pytania, w wielu przypadkach to wystarczy, aby zaciekawić czytelnika i przekazać mu wszystko, co najważniejsze. Koniecznie natomiast trzeba uwzględnić te informacje, które będą szczególnie istotne dla naszego odbiorcy. Pomoże w tym również przygotowanie **konspektu** w formie odwróconej piramidy.

Konspekt, s. 207

Jeśli tekst zaprasza do obejrzenia filmu, w którym występuje znana i ceniona przez użytkowników osoba, warto podać jej nazwisko w pierwszych zdaniach i dodatkowo wyróżnić je w tekście.

Jeśli jednak dopiero budujemy swoją markę, na pierwszym miejscu umieśćmy interesujący tytuł oraz opis, który przyciągnie odbiorców. Własne nazwisko należy podać w dalszej części. Jeśli publikacja się spodoba, nazwisko (pseudonim lub logo) autora zacznie być prędzej czy później rozpoznawalne i z czasem może samo w sobie stać się zachętą dla użytkowników.

Ćwiczenie 4

Zastosuj zasadę 5W, aby:

a. przygotować tekst zaproszenia na zbliżające się webinarium z udziałem eksperta,
b. zmodyfikować przygotowany tekst tak, aby zapraszał do obejrzenia nagrania webinarium, które już się odbyło.

A to ciekawe

Klucz do sukcesu

O tym, jak duże znaczenie ma tytuł oraz użyte w nim słownictwo, przekonała się kiedyś firma Microsoft. Ten sam artykuł z poradami dotyczącymi edytora prezentacji został opublikowany pod dwoma bardzo podobnymi tytułami: *10 smart tips for using PowerPoint* oraz *10 key tips for using PowerPoint*. Okazało się, że artykuł opatrzony drugim tytułem przeczytało kilka razy więcej osób.



12.3. Grafika i identyfikacja wizualna

Przygotowanie grafiki, podobnie jak opracowanie tekstów informacyjnych, jest elementem towarzyszącym każdej formie publikacji sieciowej. Do elementów graficznych możemy zaliczyć:

- ilustracje w tekście – zdjęcia, **grafiki informacyjne (infografiki)**, wykresy, schematy,
- plansze tytułowe i końcowe w filmach lub wystąpieniach na żywo,
- podpisy osób w filmie lub w czasie wystąpień na żywo,
- plakaty i banery informacyjne zapraszające na wydarzenia oraz zachęcające do zapoznania się z publikacjami,
- logotypy,
- materiały prezentacyjne i animacje wykorzystywane w filmach lub podczas wystąpień na żywo.

Grafiki informacyjne, s. 219–235

Język wizualny stał się bardzo popularnym sposobem komunikacji w internecie, ponieważ jest prosty i atrakcyjny oraz przyciąga uwagę odbiorców. Warto poświęcić czas na przemyślenie i przygotowanie odpowiednich grafik towarzyszących publikacji sieciowej.

Oprogramowanie i serwisy ułatwiające przygotowanie grafik do publikacji

Istnieje wiele aplikacji i serwisów, dzięki którym można przygotować atrakcyjną grafikę. Niektóre z nich omówiliśmy w poprzednich częściach podręcznika. Warto jednak kilka aplikacji i serwisów wyróżnić. Przedstawia je tabela 12.1.

Dobra rada

Wskazówki dotyczące grafiki możesz znaleźć w tematach *Sztuka prezentacji* oraz *Grafika 2D i 3D* z podręcznika *Informatyka na czasie 1.*

Program lub serwis	Przeznaczenie
Canva	Serwis oferuje wiele narzędzi, szablonów oraz grafik, które można wykorzystać w projektach do stworzenia m.in. plakatów, banerów, prezentacji i plansz tytułowych.
Edytor prezentacji (np. PowerPoint, Impress)	Aplikacje do tworzenia prezentacji mają wbudowane narzędzia graficzne. Dzięki nim można wygodnie przygotować plakat, baner lub planszę tytułową.
Wycinek i szkic (Windows 10)	Narzędzie to pozwala na łatwe wykonywanie i przetwarzanie zrzutów ekranowych, które mogą być przydatne przy tworzeniu samouczków. Można je uruchomić skrótem klawiszowym **Windows + Shift + S**.
Pixabay, Pixels, Unsplash	Te serwisy internetowe udostępniają zdjęcia na otwartych licencjach (często do użytku komercyjnego). Przed użyciem zdjęcia trzeba sprawdzić warunki licencyjne, ponieważ mogą one narzucać sposób oznaczania autora zdjęcia.
Programy graficzne (np. Inkscape, GIMP)	Program do grafiki wektorowej (np. Inkscape) można wykorzystać np. do stworzenia logo lub baneru. Programy do grafiki rastrowej pomogą m.in. odpowiednio wykadrować zdjęcia.

Tabela 12.1. Programy i serwisy ułatwiające przygotowanie grafik do publikacji

Warto wiedzieć

W internecie można znaleźć również serwisy oferujące darmowe zasoby muzyczne, np. Free Music Archive lub Bensound.

Identyfikacja wizualna

Identyfikacja wizualna

Twórcy, którzy publikują treści w internecie, dbają o odpowiednią **identyfikację wizualną**, czyli rozpoznawalność przygotowywanych przez siebie materiałów. Mogą ją zapewnić stałe, powtarzane w każdej publikacji elementy graficzne, nazywane czasem motywem lub szablonem. Należą do nich:

- czcionki,
- tła,
- kolorystyka,
- zdjęcie,
- układ treści.

Warto wiedzieć

W publikacjach multimedialnych stałymi elementami są intra oraz dżingle.

Stałe elementy pozwalają szybko skojarzyć treści z daną firmą, serią publikacji lub autorem. Warto także zadbać o ich spójność nie tylko w samych publikacjach, lecz także na stronie WWW, blogu, w mediach społecznościowych i wszelkich zapowiedziach.

Logo

Sygnet, logotyp

Jednym z elementów graficznych ściśle związanych z identyfikacją wizualną jest **logo**. Może się ono składać z samego symbolu graficznego, zwanego **sygnetem**, z sygnetu i **logotypu** (nazwy) lub samego logotypu. Czasem zawiera też dodatkowy tekst, np. slogan.

Warto wiedzieć

Oryginalne rozwiązania przyciągają uwagę odbiorców, ale nie można z nimi przesadzić. Gdy jedna z firm technologicznych wprowadziła na swoich stronach przewijanie treści lewo–prawo, szybko musiała z pomysłu zrezygnować, bo użytkownicy nie potrafili z tego korzystać.

Zazwyczaj dąży się do tego, by logotyp był oryginalny, ale równocześnie prosty i łatwy do użycia w różnych publikacjach. Przykładowe logo składające się z sygnetu i logotypu przedstawia rysunek 12.2.



Rys. 12.2. Logo Olimpiady Informatycznej, składające się z sygnetu i logotypu

Ćwiczenie 5

Wskaż wizualne i (o ile to możliwe) dźwiękowe elementy identyfikujące:

a. wybrany serwis VOD (wideo na życzenie),
b. Wielką Orkiestrę Świątecznej Pomocy,
c. miasta: Rybnik, Katowice, Melbourne,
d. wybranego blogera lub kanał internetowy.

12.4. Nagrania audio i wideo oraz wystąpienia na żywo

Wiele treści w internecie ma postać różnego rodzaju filmów, audycji lub relacji na żywo. Tabela 12.2. przedstawia najpopularniejsze rodzaje publikacji związane z wideo i dźwiękiem.

Rodzaj publikacji	Opis
Webinarium (webinar)	Rodzaj seminarium organizowanego na żywo w internecie. Może być prowadzone przez jedną osobę lub wiele osób znajdujących się w oddalonych miejscach. Podczas webinarium wykorzystuje się transmisje audio-wideo z kamer, udostępnianie prezentacji lub zawartości ekranu komputera. Możliwa jest interakcja z publicznością, np. dyskusja z uczestnikami lub odpowiedzi na pytania (zadane np. na czacie). Webinaria wykorzystuje się głównie jako narzędzia szkoleniowe i marketingowe.
Webcast	Audycja internetowa w formie wideo (przypominająca audycję telewizyjną) najczęściej realizowana na żywo, choć może być również odtwarzana z przygotowanego wcześniej materiału. Zazwyczaj nie wykorzystuje możliwości bezpośredniej komunikacji między odbiorcami a prowadzącymi. Nie oznacza to jednak całkowitego braku interakcji – widzowie mogą np. głosować w przeprowadzanych podczas programu ankietach.
Podcast	Rodzaj audycji internetowej (najczęściej audio) publikowanej regularnie jako seria odcinków.
Screencast	Film będący zapisem czynności wykonywanych na ekranie komputera uzupełniony o komentarz lektora. Wykorzystuje się go jako materiał szkoleniowy lub demonstracyjny. Nagrywanie i publikowanie tego typu filmu określane jest często jako screencasting, a film nazywany jest tutorialem.

Webinarium (webinar)

Webcast

Podcast

Screencast

Tabela 12.2. Popularne typy internetowych publikacji audiowizualnych

Znaczenie przedstawionych wyżej pojęć może się zmieniać wraz z rozwojem technologii. Wszystkie te działania wymagają jednak odpowiedniej pracy z kamerą oraz mikrofonem.

Praca z kamerą

W zastosowaniach amatorskich i półprofesjonalnych do rejestrowania filmów wykorzystuje się nie tylko kamery, lecz także smartfony. Na przykład w czasie nadawania na żywo użycie smartfonu pozwoli uzyskać o wiele lepszy efekt niż korzystanie z kamery wbudowanej w laptop.

Większość telefonów po zainstalowaniu dodatkowej aplikacji pozwala używać aparatu jak zewnętrznej kamery USB lub kamery bezprzewodowej (z wykorzystaniem wi-fi). Dzięki temu otrzymujemy lepszy obraz i więcej możliwości kadrowania.

W czasie przygotowań do nagrania warto pamiętać o kilku prostych zasadach, które pomogą uzyskać materiał lepszej jakości.

Warto wiedzieć

Wiele aplikacji pozwala ustabilizować zapisany film, czyli pozbawić go drgań. Warto z tego korzystać, ale trzeba pamiętać, że stabilizacja odbywa się w tym przypadku kosztem rozdzielczości filmu.

Praca z kamerą

Podczas nagrywania materiału wideo warto pamiętać o wskazówkach, które pomogą wykorzystać możliwości dostępnych urządzeń i podnieść jakość nagrań.

Nagranie na zewnątrz

Mikrofon

Korzystaj z mikrofonu krawatowego lub ręcznego. Zamontuj też osłonę chroniącą przed szumem.

Oświetlenie

Zadbaj o to, by nagrywany obiekt lub osoba były odpowiednio oświetlone. Uważaj na cień rzucany przez czapkę lub kapelusz.

Naturalne światło

Najlepsze ujęcia powstaną o poranku, gdy słońce jeszcze jest nisko, albo po południu, gdy światło nie jest już tak ostre.



Stabilizacja kamery

Przy ujęciach w plenerze sprawdzi się tzw. gimbal, czyli urządzenie, które stabilizuje kamerę.

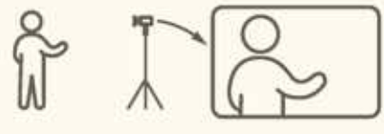


Unikaj kadrowania z funkcją zoomu, ponieważ może to pogorszyć jakość nagrania. Wykorzystuj optyczne możliwości kamery lub przestaw ją tak, aby dopasować kadr.



Podczas nagrań na żywo zabezpiecz przestrzeń przed niepowołanymi osobami. Poproś kogoś, by informował przechodniów o trwającym nagraniu i prosił ich o omijanie filmowanego obszaru.

Kadrowanie



Jeśli dużo gestykulujesz lub korzystasz z rekwizytów, lepiej wypadniesz w dalszym planie, tzw. planie średnim. Nie musisz też być w centrum kadru.

Nagranie w pomieszczeniu



Sztuczne oświetlenie
Światło powinno być rozproszone, miękkie i równomierne. Pozwoli to filmowanej osobie uniknąć cieni pod oczami i refleksów na twarzy.

Mikrofon
Korzystaj z dodatkowego, zewnętrznego mikrofonu. Polepszy on jakość nagrania.

Tylna kamera
Staraj się korzystać z tylnej kamery smartfonu. Zazwyczaj ma ona lepsze parametry od kamery przedniej.

Nagranie z wielu kamer
Jeśli używasz jednocześnie wielu kamer, włącz w każdej z nich nagrywanie obrazu oraz dźwięku. Ułatwisz sobie synchronizację fragmentów.



Przy ujęciach z bliska kamera powinna być na wysokości twoich oczu. Uważaj z gestykulacją, aby dłonie nie pojawiły się w kadrze.



Zadbaj o przestrzeń wokół siebie. Usuń wszystkie niepotrzebne rzeczy, które mogłyby odwracać uwagę od tego, co najważniejsze.



Jeśli nie masz wpływu na swoje otoczenie, ustaw wirtualne tło. Pamiętaj, żeby postać na tle była dobrze oświetlona.

Zachowanie przed kamerą

Aby dobrze wypaść przed kamerą, warto stosować zasady podobne do tych, które obowiązują podczas wystąpień publicznych.

Utrzymuj kontakt wzrokowy



Podczas wywiadu patrz na osobę, z którą rozmawiasz, a nie w obiektyw kamery. W innych okolicznościach zapytaj operatora kamery, jaka pozycja będzie bardziej naturalna, lub zrób próbę i oceń to samodzielnie.

Dobierz odpowiedni strój



Kamera lubi szarości i błękity. Unikaj drobnych wzorów, pasków i zieleni – zwłaszcza gdy występujesz w studiu telewizyjnym. Załóż dłuższą spódnicę lub dłuższe spodnie, aby w pozycji siedzącej niczego niepotrzebnie nie odsłonić.

Przyjmij właściwą postawę



Wyprostuj się, stań pewnie na lekko rozstawionych nogach ze stopami skierowanymi na zewnątrz. Jeśli siedzisz, siedź prosto, ale swobodnie. Nie rozstawiaj nóg i nie zapadaj się w głębokich fotelach. Unikaj kręcenia i bujania się na krześle.

Gestykuluj z umiarem



Wyznacz przestrzeń, w której możesz się poruszać podczas nagrania, by nie wyjść z kadru, gdy kamera jest ustawiona nieruchomo (lub poproś operatora o wyznaczenie tej przestrzeni).

Unikaj biżuterii i daj się pomalować



Mieniąca się biżuteria może powodować refleksy, a duże ozdoby mogą odwracać uwagę od twojej wypowiedzi. Dzięki makijażowi oświetlenie nie będzie wywoływać refleksów na twojej skórze.

Przekazuj właściwe emocje



Uśmiech i pozytywne nastawienie prelegenta udzielają się słuchaczom. Jednak nie okazuj radości, gdy mówisz o rzeczach poważnych lub smutnych – nie będzie to naturalne ani wiarygodne.

Zanim wystąpi się przed kamerą (np. w studiu telewizyjnym lub w sondzie ulicznej), warto wcześniej porozmawiać zarówno z osobą prowadzącą rozmowę, jak i z operatorem kamery. To, jak wypadniemy na nagraniu, będzie zależało również od pracy operatora.

O zasadach zachowania przed kamerą warto pamiętać również podczas przygotowywania np. poradnika kucharskiego czy pokazu tresury ulubionego zwierzęcia. Wtedy oprócz kucharza czy tresera ważne są potrawa lub zwierzę. To one będą głównymi obiektami nagrania – należy zadbać o odpowiednie ich sfilmowanie, a samemu pozostać nieco w tle.

Na planie filmowym mogą się zdarzyć nieprzewidziane sytuacje. Ujęcia z wielu kamer (np. kilku smartfonów lub tabletów) ułatwią pozbycie się niepotrzebnych ujęć na etapie montażu, a także pozwolą zwiększyć dynamikę ostatecznej wersji filmu.

Dobra rada

Jeśli nagrywasz film z kilku kamer, poproś kogoś o pomoc. Możesz także ustawić kilka urządzeń na stałe, a samodzielnie nagrywać ujęcia z ręki.

Ćwiczenie 6

Przeanalizuj filmy dotyczące wybranego zagadnienia, które mają dużą oglądalność w internecie. Zastanów się:

a. w jaki sposób zostały nagrane i jakie techniki w nich zastosowano,

b. jak zachowują się osoby występujące na nagraniach.

Praca z mikrofonem

Tworzenie publikacji dźwiękowych oraz filmowych wymaga odpowiedniej pracy z mikrofonem. Aby uzyskać dobre efekty, warto stosować się do kilku podstawowych zasad.

- Przećwicz pracę z mikrofonem. Wypróbuj różne ustawienia parametrów mikrofonu oraz różne jego położenie. Wiele mikrofonów pozwala wybrać sposób nagrywania (np. dźwięk dookolny lub kierunkowy) oraz czułość. Część parametrów można ustawić w programie do rejestrowania dźwięku. Zazwyczaj powinno się zachować odległość kilkunastu centymetrów między twarzą a mikrofonem. Unikaj zbyt głośnego dźwięku powodującego zniekształcenie nagrania. W razie potrzeby zmniejsz czułość mikrofonu.

Warto wiedzieć

Mikrofony rejestrują nawet dźwięki występujące w trakcie oddychania. Mogą one stanowić istotne zakłócenia dla odbiorcy.

- Zadbaj o wyciszenie pomieszczenia. Większość mikrofonów rejestruje tzw. **szum tła** (ang. *background noise*), czyli dźwięki pochodzące z otoczenia. Sprawdź, czy tam, gdzie planujesz nagrywać, występują takie dźwięki. Zapewnij sobie spokojne miejsce: zamknij okno, drzwi i poinformuj innych o nagraniu (np. przyklej na drzwiach kartkę z odpowiednim komunikatem). Unikaj echa.

Szum tła

- Przygotuj skrypt wypowiedzi. Nie musi to być dokładny zapis tego, co chcesz powiedzieć, lecz ogólny plan, który pozwoli uniknąć pomyłek i przestojów. Wypowiedź będzie żywsza niż w przypadku odczytania przygotowanego tekstu.
- Zadbaj o dykcję i natężenie głosu. Mów do mikrofonu tak jak do osoby siedzącej naprzeciwko ciebie przy stole, w głośnym otoczeniu.
- Odpowiednio zareaguj na pomyłkę. Zrób chwilę przerwy i dopiero potem wznów nagranie. Przerwa ułatwi później montaż.

Warto wiedzieć

W telewizji często używa się tzw. teleprompterа, czyli ekranu, który wyświetla tekst do odczytania przez osobę występującą przed kamerą. Widz odnosi wtedy wrażenie, że tekst jest wygłaszany z pamięci.

Jak mówić do mikrofonu?

Niezależnie od tego, czy wypowiadamy się na filmie czy w materiale audio, pomóc może wyobrażenie sobie osób, które będą nas słuchać. Oprócz tego warto zastosować kilka prostych zasad.

Mów naturalnie



Mów podobnie jak podczas rozmowy: zmieniaj tempo, natężenie i brzmienie głosu. Nie używaj stylu urzędowego i zbyt oficjalnego.

Mów wyraźnie



Lepiej mówić wolniej i ciszej niż zbyt szybko i zbyt głośno. Unikaj zjadania końcówek i wydawania zbędnych dźwięków.

Mów prosto



Dopasuj język do odbiorcy. Jeśli nie jest on sprecyzowany, używaj prostego języka, aby każdy mógł cię zrozumieć.

Mów konkretnie



Używaj krótkich zdań oraz unikaj dygresji. Korzystaj z przygotowanego skryptu.

Nagrywanie i udostępnianie ekranu

Podczas nagrywania operacji wykonywanych na ekranie komputera również warto stosować się do kilku zasad.

Przygotuj ekran komputera



Usuń z pulpitu, paska zadań oraz zasobnika systemowego zbędne dokumenty i ikony. Pozamykaj niepotrzebne okna aplikacji oraz zakładki w przeglądarce internetowej. Ustaw jako tapetę tło w jednolitym, stonowanym kolorze.

Zablokuj powiadomienia



Ustaw tryb pracy komputera tak, aby zablokowane były wszystkie powiadomienia (np. od programu antywirusowego). Zamknij komunikatory, wyłącz automatyczne aktualizacje. Zablokuj wyskakujące okna w przeglądarce internetowej.

Stosuj zasady zachowania się przed kamerą



Jeśli chcesz w materiale uwzględnić siebie, zastosuj się do wskazówek dotyczących wystąpień przed kamerą oraz pamiętaj, aby czasem spojrzeć w obiektyw. Dobrze to zrobić wtedy, kiedy zapowiadasz, co za chwilę zaprezentujesz.

Skup się na wskaźniku myszy



Poruszaj kursorem myszy precyzyjnie – wolnymi, płynnymi ruchami wskazuj wykonywane operacje. Skorzystaj z narzędzi do wygładzania ruchów kursora w programie rejestrującym ekran komputera.

Podziel materiał na mniejsze fragmenty



Materiał zarejestruj jako serię krótkich nagrań. Zacznij nagrywać kilka sekund przed właściwą częścią filmu i zakończ kilka sekund po niej. W kolejnych nagraniach staraj się zachować te same ustawienia ekranu i mikrofonu.

Nie spiesz się



Zarezerwuj sobie odpowiednio dużo czasu, zwłaszcza jeśli masz małe doświadczenie w nagrywaniu. Stworzenie jednominutowego samouczka może zająć nawet ponad godzinę.

Montaż materiału

Dobra rada

Stosuj z umiarem efekty dodatkowe – animacje napisów i przejść między ujęciami oraz dżingle.

Dobra rada

Na urządzaniach mobilnych możesz użyć aplikacji Hokusai Audio Editor (iOS/iPadOS) lub Super Sound (Android).

Przed opublikowaniem materiału trzeba wykonać jego korektę i montaż. Warto wyregulować dźwięk, usunąć szum i ustabilizować obraz. Do wideo można dodać plansze, napisy początkowe i końcowe, a do audio – zapowiedzi, dżingle i zakończenie.

Do montażu prostych projektów filmowych można wykorzystać oprogramowanie dostarczone wraz z systemem, np. aplikację Zdjęcia w Windows, iMovie w macOS czy iPadOS. Przy bardziej złożonych projektach warto sięgać po programy oferujące więcej opcji, np. używany w produkcjach filmowych DaVinci Resolve. Do pracy nad dźwiękiem może się przydać program Audacity, bezpłatny i dający duże możliwości.

Screencast, s. 211

Do nagrywania ekranu oraz podstawowego montażu **screencastu** można wykorzystać usługę Stream (wymagane jest szkolne konto Office 365). Wbudowana w nią funkcja **Zarejestruj ekran** pozwala nagrać czynności wykonywane na pulpicie lub w wybranej aplikacji, narrację oraz obraz z kamery. Czas nagrania nie może jednak w tym wypadku przekroczyć 15 minut.

Innym sposobem jest wykorzystanie wbudowanej w system operacyjny funkcjonalności ukrytej pod skrótem klawiszowym **Windows + G**.

Warto wiedzieć

W systemach macOS i iPadOS również znajdują się wbudowane narzędzia do nagrywania ekranu.

Zapamiętaj

Zapewnienie odbiorcom ciekawej rozrywki lub wartościowej wiedzy może przysłonić niedoskonałości techniczne przygotowanego materiału. Dlatego przed opracowaniem publikacji należy się przede wszystkim zastanowić, jaki jest jej cel.

12.5. Jak poznać (lepiej) swojego odbiorcę?

Przed przygotowaniem publikacji internetowej przyjmujemy wiele założeń na jej temat. Jednym z najważniejszych jest pytanie o głównego odbiorcę. Nie zawsze nasze założenia i wynikające z nich działania odniosą zamierzony skutek. Dlatego warto śledzić efekty i wyciągać wnioski – poprawiać zamieszczone treści i uwzględniać nową wiedzę podczas przygotowywania kolejnych materiałów.

Do zbadania tego, w jaki sposób nasza publikacja oddziałuje na odbiorców, możemy wykorzystać gotowe narzędzia. Przedstawia je tabela 12.3.

Dobra rada

Do narzędzi analitycznych możesz sięgnąć również na etapie planowania publikacji. Pozwolą one m.in. odnaleźć słowa często wyszukiwane przez internautów.

Lid, s. 203

Narzędzie	Przeznaczenie
Google Trends	Pozwala sprawdzać, które hasła są najczęściej wyszukiwane w sieci, oraz badać zmiany ich popularności w czasie. Na podstawie tych wyników można zidentyfikować poszukiwane przez internautów słowa i używać ich we własnych tekstach, np. w opisach, tytułach, **lidach** czy w głównej treści publikacji.
Serwisy analityczne (np. Google Analytics, stat.4u.pl)	Służą do zbierania informacji o osobach odwiedzających dane serwisy. Można się z nich dowiedzieć więcej o swoich odbiorcach, m.in. poznać ich wiek, narodowość, miejsce, z którego łączą się z serwisem, sposób dotarcia do naszych treści (przez wyszukiwanie, bezpośrednie wpisanie adresu czy link z innego serwisu), czas spędzony w serwisie, odwiedzane podstrony.
Narzędzia badające internet (np. Mediapanel)	Warto korzystać z informacji i raportów przekazywanych przez firmy badające internet, przepływy odbiorców oraz trendy. Informacje te są skierowane głównie do reklamodawców, ale mogą być również przydatne przy rozwijaniu własnego serwisu.

Tabela 12.3. Narzędzia analizujące zasoby internetu

Warto wiedzieć

Wiele serwisów (np. tych, które pozwalają publikować filmy) oferuje swoje własne, wbudowane narzędzia analityczne.

Łączenie informacji z różnych źródeł oraz śledzenie trendów wyszukiwania pomoże przygotowywać treści nie tylko lepiej odpowiadające oczekiwaniom odbiorców, lecz także łatwiejsze do znalezienia w gąszczu informacji.

Podsumowanie

- Aby zainteresować odbiorców przygotowaną treścią, należy nadać jej przyciągający uwagę tytuł, wykorzystać wysokiej jakości ilustracje oraz teksty zawierające słowa, które internauci łatwo odnajdą dzięki wyszukiwarkom.
- Treść publikacji należy dostosować do odbiorcy, m.in. jego wieku oraz doświadczeń. Trzeba zadbać o odpowiedni język przekazu.
- W konstruowaniu tekstów informacyjnych może pomóc konspekt zapisany w postaci odwróconej piramidy – najważniejsze treści, najszerzej opisane, umieszczamy na początku komunikatu, a najmniej ważne – na końcu.
- Identyfikacja wizualna oraz dźwiękowa nadaje publikacjom jednolitą, uporządkowaną formę i pozwala skojarzyć je z autorem.
- W pracy z kamerą i mikrofonem warto stosować podstawowe zasady, które pomogą uniknąć niepotrzebnego powtarzania nagrań oraz skrócą czas montażu.
- Narzędzia analityczne pozwalają lepiej poznać odbiorców treści oraz dopasować publikacje do ich potrzeb.

Zadania

* 1 Poszukaj serwisów oferujących gotowe dżingle, które można wykorzystać we własnych nagraniach.

* 2 Sprawdź, czy i w jaki sposób możesz spowodować, by publikowane przez ciebie nagrania w różnych serwisach wideo (np. YouTube, Vimeo) były odtwarzane bez reklam.

* 3 Znajdź informacje o tym, jakie tematy i hasła były najczęściej wyszukiwane przez ostatnich 12 miesięcy w Polsce w kategorii edukacja. Zastanów się, jak możesz wykorzystać tego typu analizy do przygotowania własnych treści na określony temat.

** 4 Przygotuj grafikę do wykorzystania w sieciach społecznościowych reklamującą serię webinariów lub blog na interesujący cię temat. Załóż, że grafika będzie kierowana do twoich kolegów.

** 5 Przygotuj tekst, który będzie relacją z wybranego przez ciebie webinarium lub podcastu. Tekst powinien zachęcić twoich rówieśników do odsłuchania nagrania.

*** 6 Przygotuj podcast na wybrany temat. Opracuj towarzyszące mu materiały promocyjne: zapowiedź tekstową i grafiki do zamieszczenia w sieciach społecznościowych lub na stronach WWW.

*** 7 Przygotuj publikację wideo (webcast, screencast lub webinarium) na wybrany temat związany z edukacją. Opracuj towarzyszące publikacji materiały promocyjne (zapowiedź tekstową, grafiki do zamieszczenia w sieciach społecznościowych lub na stronach WWW).

13. Grafiki informacyjne

Wraz z rozwojem technologii zwiększyła się rola komunikacji wizualnej. Ma ona długą tradycję, gdyż najstarsze zachowane środki przekazu to powstałe przed tysiącami lat malowidła naskalne. Ukazywały one wizerunki zwierząt oraz sceny polowań i nie były tylko dekoracjami – dostarczały także informacji. Współcześnie grafiki informacyjne uznaje się za gatunek dziennikarski, a ich głównym zadaniem jest skuteczne przekazanie wiadomości. W tym temacie dowiesz się, jak pracować nad takimi grafikami.

Cele lekcji

- Dowiesz się, czym jest grafika informacyjna, oraz poznasz przykłady różnych rodzajów takich grafik.
- Nauczysz się projektować proste infografiki.
- Rozwiniesz umiejętność pracy z grafiką komputerową.

Ćwiczenie 1

Opisz swoje pasje i zainteresowania, używając wyłącznie:

a. edytora tekstu i 150–200 słów,
b. edytora grafiki i narzędzia **Pióro**,
c. edytora grafiki i dwóch narzędzi: **Pióro** oraz **Tekst**.

13.1. Sposoby przekazywania informacji

Informacje można przekazywać różnymi metodami. Poniżej przedstawiono trzy sposoby podania danych dotyczących marnowania żywności.

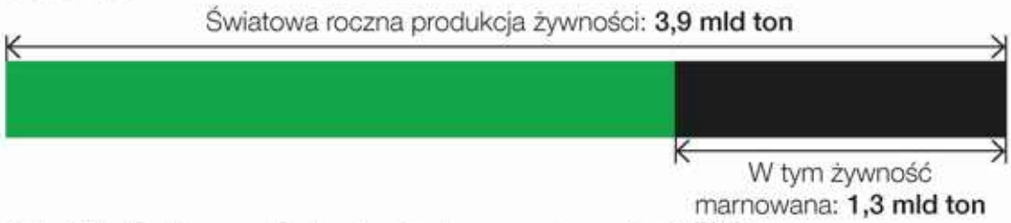
Sposób 1

Organizacja Narodów Zjednoczonych ds. Wyżywienia i Rolnictwa, nazywana w skrócie FAO, zajmuje się redystrybucją żywności i rozwojem obszarów wiejskich. Opracowuje również raporty dotyczące żywności. Z raportu opublikowanego w 2011 r. wynika, że w ciągu roku na świecie produkuje się ok. 3,9 mld ton żywności, z czego marnuje się 1,3 mld ton jej jadalnych części.

Sposób 2

Według raportu *Global food losses and food waste – Extent, causes and prevention*, przygotowanego w 2011 r. przez Organizację Narodów Zjednoczonych ds. Wyżywienia i Rolnictwa, w ciągu roku na świecie marnuje się 1,3 mld ton żywności, co stanowi 1/3 całej produkcji (ok. 3,9 mld ton).

Sposób 3



Źródło: *Global food losses and food waste – Extent, causes and prevention*, FAO, 2011.

Dobra rada

Aby graficznie porównać wartości, możesz użyć prostokątów, których pola są w odpowiednich proporcjach. Taki sposób wizualizacji ma najmniejszy tzw. współczynnik kłamstwa (ang. *lie factor*). Unikaj natomiast wykresów kołowych. Mają one wysoki współczynnik kłamstwa, ponieważ porównywanie wycinków koła jest trudne dla ludzkiego oka.

Jeśli zależy nam na porównaniu dwóch wartości, to rozbudowany opis (sposób 1) nie jest efektywny. Lepiej ten cel realizuje krótszy tekst (sposób 2). Najlepiej jednak sprawdzi się prosta grafika (sposób 3), która do minimum ogranicza wszystkie dodatkowe informacje.

Grafika informacyjna (infografika)

Jednym ze skutecznych sposobów przekazywania informacji są **grafiki informacyjne**, nazywane również **infografikami** (ang. *infographics*). Przedstawiają one treść w sposób wizualny tak, aby była ona możliwie prosta i dostosowana do odbiorcy.

Uporządkowanie informacji

Infografiki powinny zawierać informacje już uporządkowane. Jest to konieczne, aby odbiorca mógł je łatwo zrozumieć i przyswoić. Wyróżniamy 5 głównych **sposobów porządkowania informacji**:

Sposoby porządkowania informacji

- według alfabetu,
- według czasu,
- według hierarchii,
- według kategorii,
- według przestrzeni.

Dobra rada

Pamiętaj, że zebranie informacji na dany temat to dopiero połowa sukcesu. Zanim zaprezentujesz dane odbiorcy, odpowiednio je uporządkuj.

Wszystkie wymienione sposoby zastosowano do uporządkowania informacji o polskich noblistach.

Sposoby porządkowania informacji

Porządek według przestrzeni (miejsce urodzenia)



Porządek alfabetyczny

- Czesław Miłosz
- Władysław Reymont
- Henryk Sienkiewicz
- Maria Skłodowska-Curie
- Wisława Szymborska
- Olga Tokarczuk
- Lech Wałęsa

Porządek według hierarchii ważności

2 nagrody:
Maria Skłodowska-Curie

1 nagroda:

- Henryk Sienkiewicz
- Władysław Reymont
- Czesław Miłosz
- Lech Wałęsa
- Wisława Szymborska
- Olga Tokarczuk

Porządek według kategorii

Nagroda literacka

- Henryk Sienkiewicz
- Władysław Reymont
- Czesław Miłosz
- Wisława Szymborska
- Olga Tokarczuk

Nagroda z fizyki

- Maria Skłodowska-Curie

Nagroda z chemii

- Maria Skłodowska-Curie

Nagroda pokojowa

- Lech Wałęsa

Porządek chronologiczny



Jeśli dobrze wybierzemy sposób uporządkowania informacji, możemy przekazać odbiorcy dodatkową wiedzę – coś, czego nie byłoby widać przy ustawieniu tych samych (lub podobnych) danych w innym porządku.

Na przykład umieszczenie noblistów na mapie pozwala zauważyć, że nie wszyscy urodzili się w obecnych granicach Polski, a uporządkowanie chronologiczne pokazuje, jak długo Polacy czekali na nagrodę po roku 1924.

Dobra rada

Sposoby porządkowania informacji możesz łączyć. Na przykład na osi czasu oprócz dat i nazwisk laureatów nagrody Nobla możesz dodatkowo umieścić informację o kategoriach, w których tę nagrodę otrzymali.

Ćwiczenie 2

a. Jakie informacje o polskich noblistach nie są widoczne w porządku alfabetycznym, a jakie – w porządku według kategorii?
b. Pomyśl o 10 przedmiotach codziennego użytku, z których korzystasz. Uporządkuj je na dwa wybrane przez siebie sposoby.

13.2. Piktogramy

Z przekazywaniem informacji za pomocą obrazu spotykamy się dziś w różnych dziedzinach życia. Na co dzień korzystamy choćby ze znaków drogowych, tabliczek informacyjnych czy emotikonów i emoji. Są to przykłady **piktogramów** (ang. *pictograms*). Dobry piktogram cechują: prostota, uniwersalność i niezależność od kultury. Na przykład znaki drogowe (rys. 13.1) są uniwersalnym sposobem zapisu informacji, zrozumiałym dla kierowców na całym świecie.

Piktogram

Warto wiedzieć

W znakach ostrzegawczych nieprzypadkowo zostały użyte kolory: żółty, czerwony i czarny. Kolor żółty łatwo dostrzec z dużej odległości. Najbardziej kontrastowy do żółtego jest kolor czarny. Natomiast czerwień obramowania podkreśla ważność znaku.



Rys. 13.1. Wybrane znaki ostrzegawcze

Piktogramy wykorzystuje się również w czasie igrzysk olimpijskich. Oznacza się nimi obiekty sportowe, pojawiają się na mapach, biletach oraz podczas transmisji telewizyjnych. Na igrzyska w Monachium w 1972 r. niemiecki projektant Otto Aicher opracował piktogramy przedstawiające różne dyscypliny sportu (rys. 13.2).



Rys. 13.2. Wybrane piktogramy olimpijskie z 1972 r.

Piktogramy pełnią ważną funkcję w komunikacji – niwelują barierę językową oraz pomagają osobom z niepełnosprawnościami (np. osobom niemówiącym).

Ćwiczenie 3

a. Zastanówcie się w grupie, w jakich sytuacjach korzysta się z piktogramów. Podajcie kilka przykładów z różnych dziedzin.
b. Odszukaj piktogramy zastosowane w tym podręczniku i określ ich rolę.

13.3. Przykłady infografik

Infografiki są rodzajem komunikacji wizualnej. Ze względu na cel przekazu można je podzielić na dwie kategorie:

Grafika narracyjna ● **grafikę narracyjną** – należą do niej: diagramy przepływowe (opisujące procesy), osie czasu, harmonogramy, schematy organizacyjne, mapy myśli,

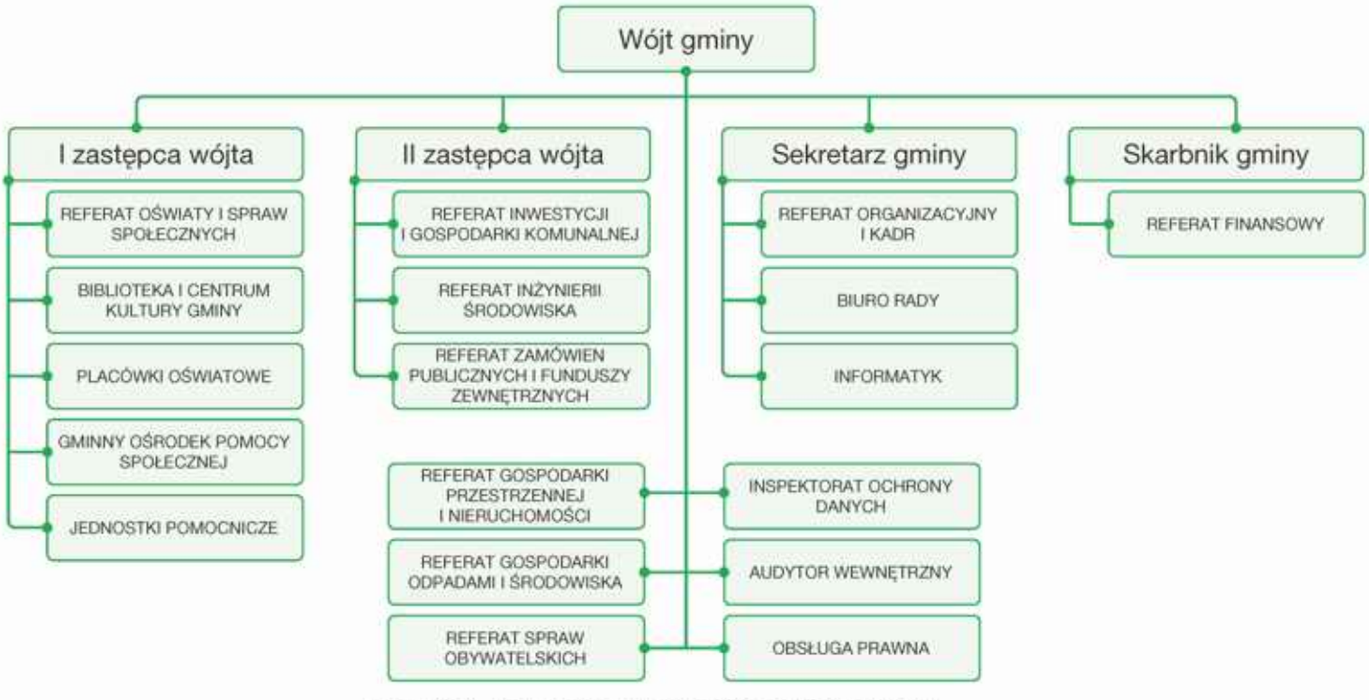
Wizualizacja danych ● **wizualizację danych** – należą do niej: grafiki liczbowe, diagramy statystyczne, grafiki porównawcze (np. do porównywania kilku cech, postaw), mapy, kartogramy i kartodiagramy.

Wiele o tworzeniu infografik można się nauczyć, analizując ich przykłady.

Schemat organizacyjny

Informacje uporządkowane według hierarchii można przedstawić za

Schemat organizacyjny ● pomocą **schematu organizacyjnego** (rys. 13.3). Cechują go prostota oraz precyzja przekazu. Schemat taki pozwala jednoznacznie opisać nawet skomplikowaną strukturę.



Rys. 13.3. Przykładowy schemat organizacyjny

Ćwiczenie 4

Opracuj schemat przedstawiający:
a. wycinek systematyki ssaków w Polsce,
b. hierarchię w zakonie Jedi z *Gwiezdnych wojen*.

Schematy linii komunikacyjnych

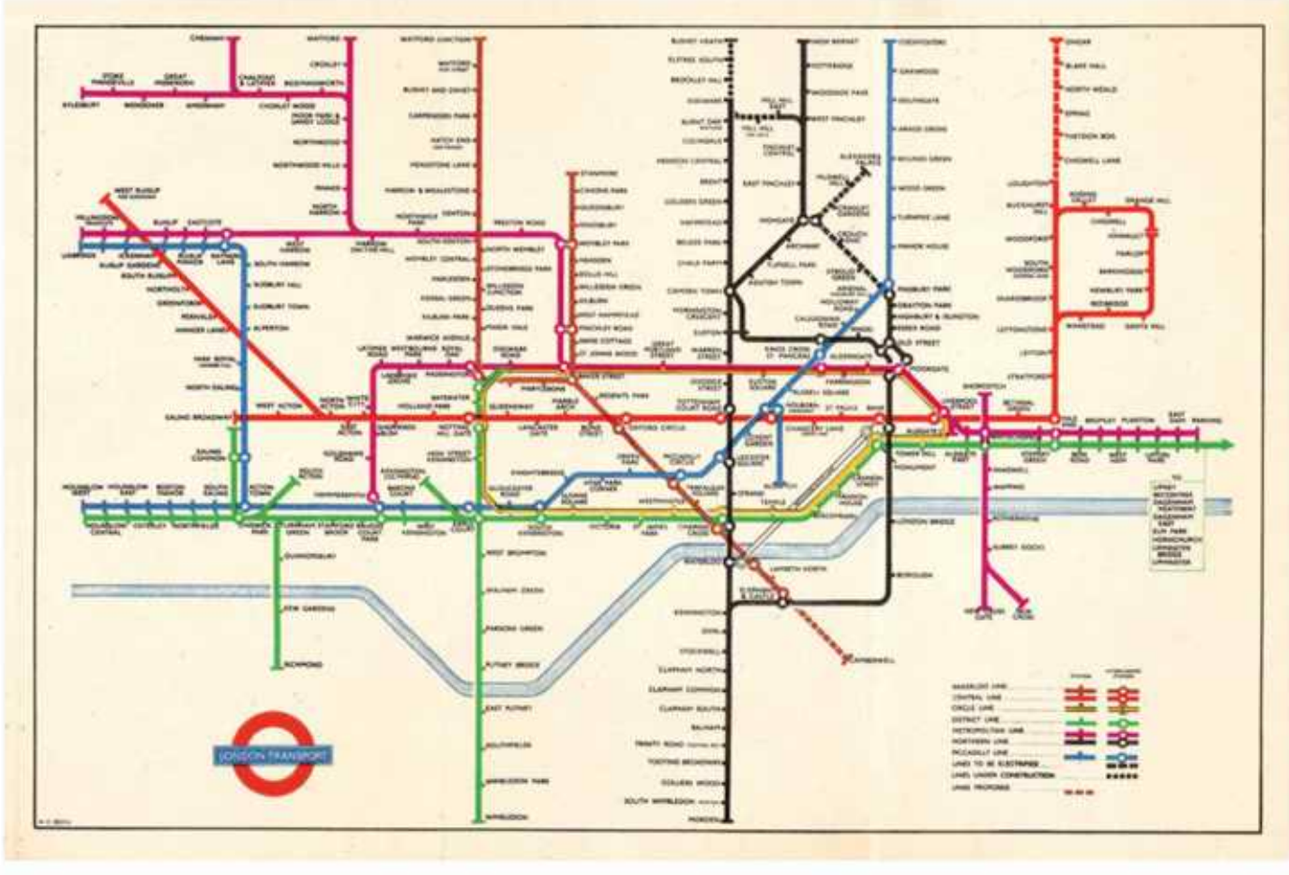
Przykładem infografiki, z którą spotykamy się często, zwłaszcza w dużych miastach, jest **schemat linii komunikacyjnych**. Wiele takich schematów jest wzorowanych na pomyśle inżyniera Henry'ego Becka, który w latach 1931–1933 opracował przejrzysty diagram linii londyńskiego metra, liczącego 200 stacji i mającego ok. 400 km długości.

Schemat linii komunikacyjnych

Warto wiedzieć

Henry Beck pracował jako kreślarz, ale z wykształcenia był inżynierem elektrykiem. Dlatego opracowując schemat linii metra, wzorował się na schematach układów elektronicznych.

Beck zauważył, że główną wadą dotychczasowego schematu było to, że odwzorowywał on skomplikowaną sieć połączeń z topograficzną dokładnością. Inżynier zrozumiał, że dla pasażerów ważne są wyłącznie wzajemne relacje przestrzenne między stacjami metra (np. kolejność stacji, miejsca przesiadek), a nie dokładne odzwierciedlenie odległości (rys. 13.4).



Rys. 13.4. Schemat linii metra w Londynie (projekt H. Becka)

Warto wiedzieć

Beck zastosował nowoczesną w tamtym czasie czcionkę bezszeryfową. Dla każdej z linii metra wybrał inny kolor.

Kolejne stacje na danej linii metra są ułożone w równych odstępach na linii prostej. Trasy na diagramie krzyżują się pod kątem 90 lub 45 stopni. Uproszczenie informacji w tym przypadku przyniosło wymierne efekty: usprawniło życie milionom osób – zarówno mieszkańcom, jak i turystom.

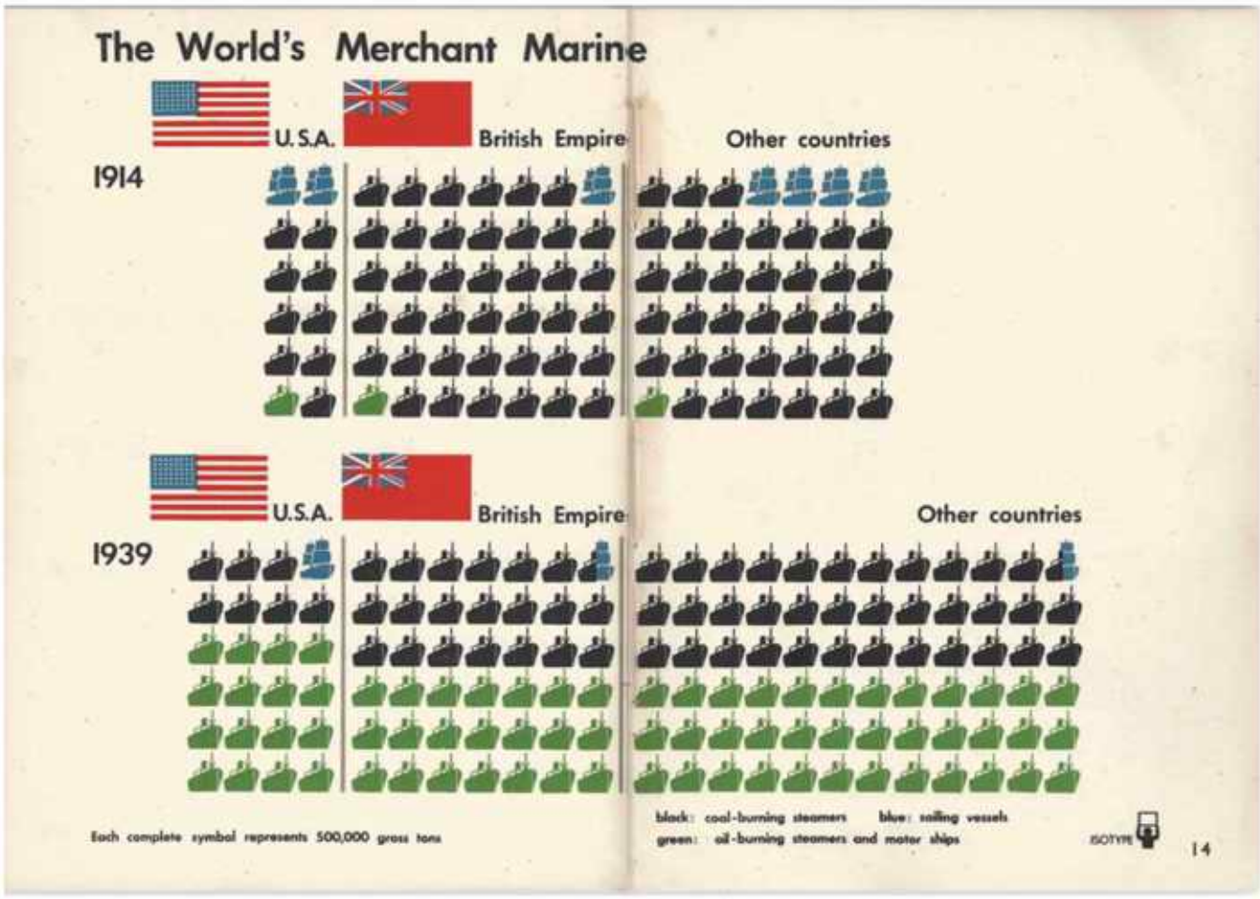
Ćwiczenie 5

Znajdź schemat linii komunikacyjnych wybranego miasta w Polsce. Czy jest inspirowany pomysłem Becka? Zapoznaj się z legendą. Jakiej kolorystyki użyto?

Język piktogramów Isotype

Język Isotype

Około 1925 r. zespół ekspertów pracujący pod kierunkiem Ottona Neuratha, socjologa i specjalisty od komunikacji, opracował obrazkowy język nazywany **Isotype** (ang. *International System of Typographic Picture Education*). Jego podstawę stanowiły jednoznaczne piktogramy, które miały za zadanie w uniwersalny i prosty sposób ilustrować złożone treści (rys. 13.5).



Rys. 13.5. Infografika opisująca liczebność marynarki handlowej na świecie w 1914 i 1939 r.

Warto wiedzieć

Piktogramy Isotype uzupełnia się o niezbędny tekst. Na infografice o marynarce handlowej są to m.in. tytuł wykresu i informacja o podziale okrętów na należące do USA, Imperium Brytyjskiego oraz reszty świata. Ponadto kolorem oznaczono sposób zasilania okrętu.

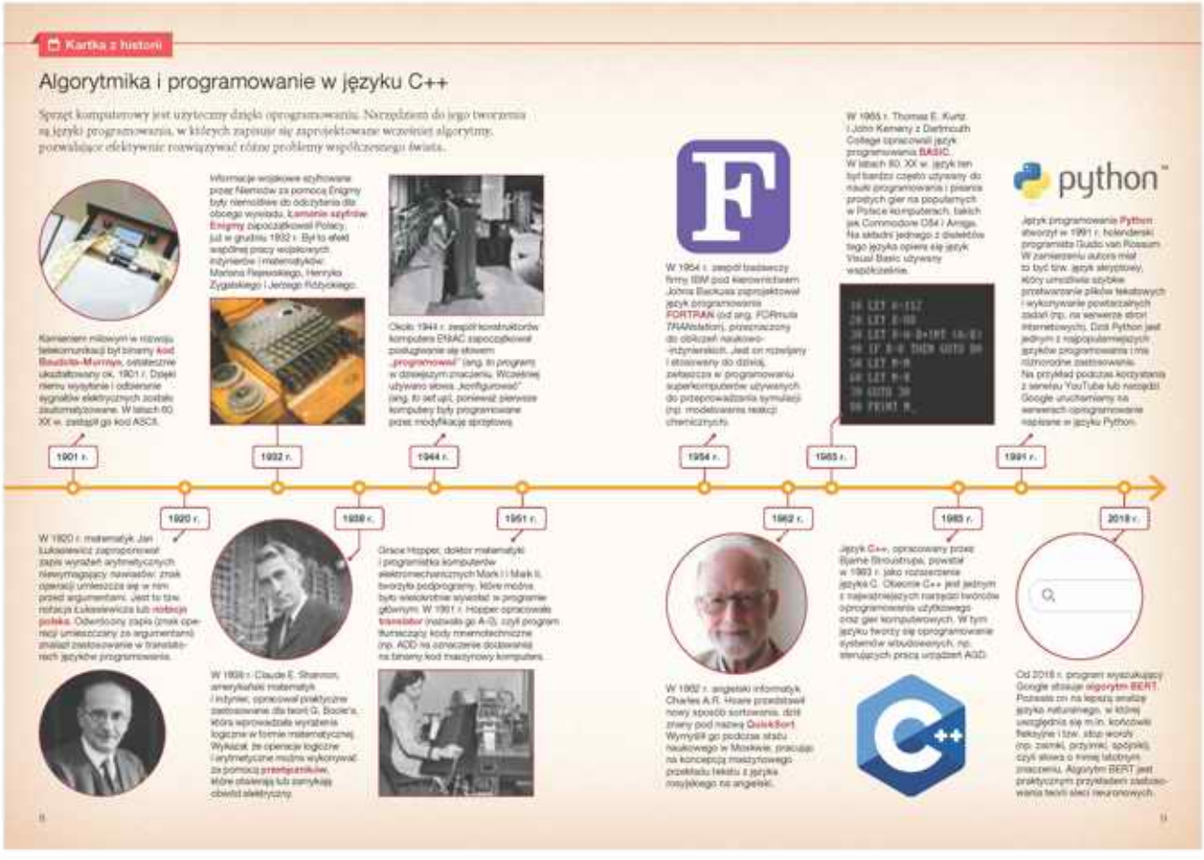
Jak pokazuje powyższy przykład, informację o ilości uzyskiwano nie przez zmianę rozmiaru znaku, lecz przez jego zwielokrotnienie.

Ćwiczenie 6

Przedstaw na infografice liczbę ton jabłek zebranych przez sadowników w Polsce oraz w innych krajach świata (łącznie) w 2000 r. oraz w 2020 r.

Oś czasu

Jeśli chcemy uporządkować informacje chronologicznie, dobrze jest zastosować **oś czasu**. Pokazuje ona przebieg wydarzeń na linii upływającego czasu. Pozwala to odbiorcy jednym spojrzeniem objąć całość zagadnienia, a wprowadzony porządek ułatwia naukę (rys. 13.6). ◀ Oś czasu



Rys. 13.6. Oś czasu pochodząca z podręcznika *Informatyka na czasie 3*

Ćwiczenie 7

Przedstaw graficznie, w postaci równoległych osi czasu, sobotnie godziny pracy wybranych urzędów i instytucji użyteczności publicznej (np. główny urząd pocztowy, muzea, ogród zoologiczny, park miejski) w dowolnym mieście w Polsce. Możesz dodać informacje o dziennej i nocnej komunikacji miejskiej.

13.4. Jak stworzyć poprawną infografikę?

Na infografice tekst i obraz powinny się wzajemnie uzupełniać. Przekazywaną treść należy przedstawić w sposób jawny, bez zbędnych metafor. Obrazowi powinien towarzyszyć tylko konieczny tekst objaśniający.

Poprawnie przygotowana infografika przedstawia tylko jeden problem lub temat. Należy zaprezentować go w sposób prosty, przystępny i dostosowany do docelowego odbiorcy.

Warto wiedzieć

Atrakcyjność wizualna nie musi być nieodłączną cechą infografiki. Ważniejsze jest skuteczne przekazanie informacji. Na przykład schematy organizacyjne dobrze realizują to zadanie, choć najczęściej nie są zbyt efektowne.

Do cech dobrej infografiki możemy zaliczyć:

- odpowiednie uporządkowanie informacji,
- odwołanie się do wiedzy i doświadczeń odbiorcy,
- prostotę i harmonię kompozycji,
- właściwy dobór kolorów,
- odpowiednie rozłożenie akcentów.

Aby odbiorcy łatwiej było przyswajać informacje we właściwej kolejności, warto różnicować wielkość elementów (co podkreśla ich hierarchię) oraz układać je względem siebie według ustalonego porządku. Znaczącą rolę odgrywają również kolory.

Funkcje koloru

Funkcja koloru

Kolory na infografice należy dobierać uważnie, ponieważ są one również nośnikiem treści. Można wyróżnić kilka **funkcji koloru**. Przedstawiono je w tabeli 13.1.

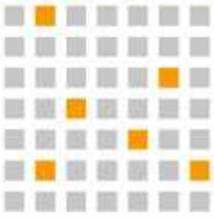
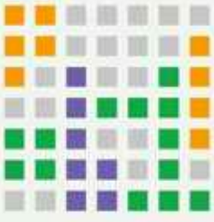
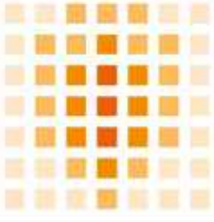
Funkcja koloru	Interpretacja graficzna
Akcentująca funkcja koloru **Funkcja akcentująca** pozwala zwrócić uwagę na wybrany element.	
Rozróżniająca funkcja koloru **Funkcja rozróżniająca** pozwala odróżnić elementy od siebie (i w niektórych wypadkach podzielić infografikę na sekcje).	
Wartościująca funkcja koloru **Funkcja wartościująca** pozwala porównać ze sobą wybrane elementy.	

Tabela 13.1. Funkcje koloru

Warto wiedzieć

Kolory mają również określoną wagę i przeznaczenie, np. do oznaczania zakazu (m.in. na znakach drogowych) używamy koloru czerwonego.

Kolory mają też określoną symbolikę, ściśle związaną z kulturą. Na przykład w kulturze europejskiej czerń oznacza zazwyczaj żałobę lub śmierć, czerwień zaś dynamizm, wysoką temperaturę lub krew.

Ćwiczenie 8

Zastanów się, jakie skojarzenia kulturowe są związane z kolorem:

a. żółtym,
b. zielonym.

Dobór kolorów

W trakcie prac nad infografiką należy zastanowić się nie tylko nad funkcją kolorów, lecz także nad ogólną kolorystyką pracy. W jej wyborze może pomóc tzw. **koło barw** (ang. *color wheel*), na którym umieszcza się kolory podzielone na trzy podstawowe grupy:

- **barwy podstawowe** – czerwony, niebieski, żółty,
- **barwy pochodne** – powstałe w wyniku wymieszania dwóch barw podstawowych: pomarańczowy (czerwony z żółtym), zielony (żółty z niebieskim) oraz fioletowy (niebieski z czerwonym),
- **barwy trzeciego rzędu** – powstałe w wyniku połączenia koloru podstawowego z sąsiadującym z nim kolorem pochodnym.

Koło barw

Barwy podstawowe

Barwy pochodne

Barwy trzeciego rzędu

Rysunek 13.7 przedstawia tradycyjne koło barw utworzone w opisany wyżej sposób dla 12 barw o różnym **nasyceniu**, czyli o różnej zawartości bieli. Kolory ułożone są w kolejności od najbardziej nasyconego, najciemniejszego na brzegu koła do najmniej nasyconego, najjaśniejszego bliżej środka koła.

Nasycenie koloru



Rys. 13.7. Koło barw

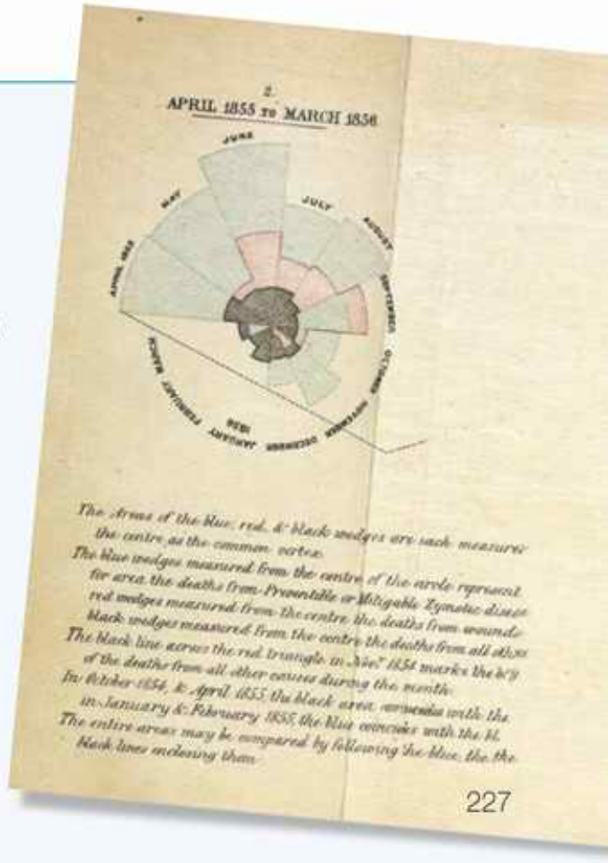
Warto wiedzieć

Założenie, że podstawowymi kolorami są czerwony, niebieski i żółty, jest uwarunkowane historycznie i wywodzi się ze sztuki malarskiej. Współcześnie wiadomo, że dla receptorów ludzkiego oka (czopków) podstawowe barwy to: czerwony, niebieski i zielony. Jeśli postrzegamy wybrany obiekt jako żółty, to znaczy, że do oka dotarło światło czerwone i zielone w proporcjach 1 : 1.

A to ciekawe

Diagram ratujący życie

Podczas wojny krymskiej (1853–1856) angielska pielęgniarka Florence Nightingale opiekowała się rannymi żołnierzami i analizowała dane o ich zgonach. Przygotowała raport, z którego wynikało, że więcej żołnierzy umierało z powodu niedożywienia i fatalnych warunków sanitarnych niż na skutek odniesionych ran. Przełożeni Nightingale nie byli zainteresowani wnioskami płynącymi z raportu, jednak ona pozostawała nieugięta. Ostatecznie swoją analizę zaprezentowała samej królowej Wiktorii. Przygotowała na tę okazję specjalną grafikę, nazywaną dzisiaj diagramem centrycznym albo biegunowym.



Koło barw pozwala dobierać kolory na kilka sposobów. Najczęściej stosowane przedstawiono w tabeli 13.2.

Układ barw	Wybór z koła barw	Opis
Barwy sąsiadujące		Kolory sąsiadujące nazywa się też analogicznymi. Pozwalają uzyskać wrażenie harmonii, gdyż nasz wzrok dostrzega, że mają ze sobą coś wspólnego.
Barwy monochromatyczne		Jeden kolor w różnych odcieniach i o rożnym nasyceniu (ciemne lub jasne, czyli pastelowe). Z reguły projekty tworzone z wykorzystaniem tego schematu są dość spokojne. Unikamy ryzyka błędnego zestawienia różnych kolorów.
Barwy trójkątne		Wyznacza się je w kole barw za pomocą trójkąta równobocznego. Jeden z kolorów powinien dominować nad pozostałymi.
Barwy dopełniające		Kolory umieszczone w kole barw naprzeciw siebie. Trzeba ostrożnie je dobierać, aby nie uzyskać zbyt dużego kontrastu, ponieważ może on wywoływać niepokój.

Tabela 13.2. Dobór kolorów z wykorzystaniem koła barw

Dobra rada

Jeśli stosujesz barwy sąsiadujące, to powinny one mieć takie samo nasycenie.

Warto wiedzieć

Internetowe narzędzie palettable.io pozwala stworzyć kompozycję 5 kolorów, z których każdy kolejny jest dobierany do wybranych poprzednio.

Podczas dobierania kolorów dobrze sprawdzają się narzędzia online. Wiele z nich umożliwia wykorzystanie koła barw i sposobów opisanych w tabeli 13.2. Przykładami darmowych narzędzi są: Adobe Color (color.adobe.com) oraz canva.com/colors/color-wheel/. Często można w nich też skorzystać z gotowych schematów kolorystycznych.

Ćwiczenie 9

W pliku, który otrzymasz od nauczyciela (np. *firma_transportowa.bmp*), znajduje się schemat organizacyjny pewnej firmy transportowej. Otwórz go w wybranym edytorze grafiki i dobierz do niego wielobarwny schemat kolorystyczny tak, aby kolor był dodatkowym nośnikiem informacji.

Zapamiętaj

Podstawową zaletą infografiki jest to, że ułatwia ona zrozumienie danego zagadnienia. Za dobrze zaprojektowaną infografikę należy uznać taką, która jest czytelna i przystępna dla odbiorcy.

Akcentowanie informacji

Najważniejsze informacje znajdujące się na infografice powinny być wyraźnie zaznaczone i odpowiednio zaakcentowane. Dzięki temu uzyskuje się kontrast między informacjami ważnymi i mniej ważnymi.

Istnieje kilka **sposobów akcentowania najważniejszych informacji** (tekstu lub grafiki) – przedstawia je rysunek 13.8.

Sposoby akcentowania informacji



Rys. 13.8. Sposoby akcentowania informacji na infografice

Warto wiedzieć

W projektowaniu graficznym oraz składzie tekstu światłem nazywa się odstęp, np. między literami, wyrazami lub elementami graficznymi.

Ćwiczenie 10

W pliku, który otrzymasz od nauczyciela (np. *akcentowanie.pdf*), znajdują się odnośniki do kilku infografik. Wybierz dwie i określ, jakie sposoby akcentowania informacji na nich zastosowano.

Typowe błędy

Źródłem chaosu na infografice może być m.in. niestosowanie się do nawyków wzrokowych czytelników. Przywykliśmy czytać od lewej do prawej i z góry na dół. Jeśli wzrok odbiorcy ma podążać inną ścieżką, to elementy graficzne i tekstowe muszą taką ścieżkę wyraźnie tworzyć.

Warto wiedzieć

Nienaturalnym sposobem układania treści jest używanie układu odwrotnego do ruchu wskazówek zegara.

Do najczęściej popełnianych błędów podczas tworzenia infografik można zaliczyć:

- nielogiczny układ treści,
- błędny dobór środków ilustrujących przekazywane informacje,
- przesadną ozdobność i zbędne szczegóły,
- źle dobrany typ wykresu lub diagramu,
- powtórzenia informacji,
- nieczytelne metafory,
- błędne użycie koloru.

Przedstawimy teraz przykład poprawnej infografiki, która łączy w sobie wiele różnych technik.

Zastosowanie różnych technik na infografice

Często infografiki są przygotowywane jako samodzielne źródła informacji, w których tekst i obraz wzajemnie się przenikają.

Tytuł infografiki i krótki tekst wprowadzający pomagają zrozumieć przedstawiany temat.

Legendę objaśniającą znaczenie kolorów i piktogramów umieszczono w neutralnym miejscu. Mimo że symbole zegara i krzyża są bardzo sugestywne, wyjaśniono je w legendzie.



LEGENDA
konflikty między państwami
konflikty polityczne (m.in. walka o władzę, zbrojna opozycja, konflikty ideologiczne)
konflikty etniczne (w tym dążenia do niepodległości)
konflikty religijne
liczba ofiar
czas trwania konfliktu

Tło infografiki jest stonowane. Użyto w nim odcieni tylko jednego koloru, który koresponduje z wymową jedynego zdjęcia. Wzmacnia to przekaz o dramacie setek tysięcy ofiar. Na tle wyraźnie wyeksponowano obszary objęte konfliktem.

W ramkach znajduje się krótka charakterystyka wybranych konfliktów. Podstawowe informacje – czas trwania konfliktu oraz liczbę ofiar – oznaczono piktogramami.

ROSJA – UKRAINA (DONBAS)

od 2014 r. † 13 tys.

Rosja wykorzystała trudną sytuację wewnętrzną na Ukrainie i zajęła Półwysep Krymski, a następnie wywołała konflikt na wschodzie Ukrainy, udzielając zbrojnego wsparcia zwolennikom niezależności tych terenów.

ARMENIA–AZERBEJDŻAN (GÓRSKI KARABACH)

od 1991 r. † 30 tys.

Konflikt toczy się o Górski Karabach. Teren leży w granicach Azerbejdżanu, jednak większość mieszkańców tego obszaru stanowią Ormianie, którzy kontrolują go dzięki wsparciu wojsk Armenii.

TURCJA (KURDOWIE)
od 1984 r.
† 34–46 tys.

LIBIA
od 2011 r.
† 33–38 tys.

IRAK
od 2003 r.
† 450 tys.

AFGANISTAN
od 2001 r.
† 150–160 tys.

INDIE–PAKISTAN (KASZMIR)
od 1947 r.
† 47–100 tys.

KOREA PÓŁNOCNA–KOREA POŁUDNIOWA
od 1953 r.
† około 1 tys.

MALI
od 2012 r.
† 4–11 tys.

JEMEN
od 1994 r.
† ponad 100 tys.

MJANMA
od 1948 r.
† 115–200 tys.

SOMALIA
od 1988 r.
† 300–550 tys.

SUDAN (DARFUR)
od 2003 r. † 300–350 tys.

SUDAN POŁUDNIOWY
od 2011 r. † 400 tys.

DEMOKRATYCZNA REPUBLIKA KONGA
od 1996 r. † 2,5–4 mln

REPUBLIKA ŚRODKOWOAFRYKAŃSKA
od 2013 r. † ponad 8 tys.

NIGERIA

od 2009 r. † 27–38 tys.

W kraju zamieszkanym przez chrześcijan i muzułmanów trwa wojna z islamskim ugrupowaniem terrorystycznym Boko Haram. Dąży ono do przekształcenia Nigerii w państwo wyznaniowe i prześladuje chrześcijan – dopuszcza się porwań, mordów.

SYRIA

od 2011 r. † 470–560 tys.

W kraju toczy się wojna domowa między wojskami prezydenta Baszara Al-Asada a opozycyjnymi ugrupowaniami zbrojnymi. Strona rządowa jest wspierana militarnie przez Rosję i Iran, natomiast jej przeciwnicy – przez USA i Turcję.

Informacje uporządkowane są według przestrzeni (mapa) oraz kategorii (rodzaje konfliktów), którym przyporządkowano kolory.

Na infografice zastosowano następujące funkcje kolorów:

- akcentującą – na monochromatycznym tle mocne kolory zwracają uwagę czytelnika,



- różnicującą – różne barwy odpowiadają różnym rodzajom konfliktów.



13.5. Tworzymy własną infografikę

Warto wiedzieć

Do najważniejszych twórców infografik na świecie należy Fernando Baptista. Jest on autorem wielu niezapomnianych prac, m.in. dla National Geographic. Część jego infografik można odnaleźć w internecie.

Praca nad infografiką to w dużej mierze proces twórczy, wymykający się twardym regułom. Określimy jednak kilka kluczowych czynności, które należałoby wykonać podczas tworzenia dowolnej grafiki informacyjnej. Przedstawmy je w postaci listy kroków:

1. Wybierz problem, który infografika ma objaśniać.
2. Określ grupę odbiorców i sformułuj najważniejszy cel infografiki.
3. Zbierz informacje na wybrany temat i zredaguj zwięzły tekst.
4. Zamień zredagowany tekst na język wizualny.
5. Określ podstawowe parametry infografiki i opracuj jej szkic.
6. Wykonaj infografikę zgodnie ze szkicem.
7. Wprowadź poprawki i dopracuj kompozycję.

Pokażemy na przykładzie, jak można zrealizować powyższe kroki.

Infografika *Recykling papieru*

Załóżmy, że mamy za zadanie opracować infografikę omawiającą proces recyklingu papieru. Chcemy, aby uczniowie szkół podstawowych i ponadpodstawowych (grupa docelowa) rozumieli, jak efektywnie można przetwarzać papier, jeśli tylko zadbamy o odpowiednią segregację odpadów (cel). Wykonaliśmy więc kroki 1 i 2.

Podczas opracowania tematu (krok 3) korzystamy tylko z rzetelnych źródeł, aby nie wprowadzić odbiorców w błąd. Jeśli interesującej nas informacji nie możemy potwierdzić w sprawdzonych źródłach, lepiej z niej zrezygnować.

Zebrane informacje najwygodniej przedstawić w formie tekstu. Już na tym etapie powinien on być zwięzły i dobrze zredagowany.

Oto przykładowy tekst:

Warto wiedzieć

Dzięki selektywnej zbiórce makulatury powstają m.in. banknoty, bandaże, opakowania na jajka, fartuchy szpitalne i papier ksero.

Recykling papieru

Co roku ścina się tysiące drzew, aby wyprodukować z nich papier na gazety, książki i opakowania. Tymczasem znaczną część makulatury można by ponownie wykorzystać. Zużyty papier wyrzucamy do niebieskich pojemników na śmieci. Odpady papierowe muszą zostać rozdrobnione na małe kawałeczki. Skrawki papieru miesza się z wodą i gotuje, aż powstanie z nich płynna miazga (pulpa). Tę mokrą pulpę rozsmarowuje się bardzo cienko na metalowej siatce. Następnie pulpa jest ściskana i suszona – tak powstaje nowy arkusz papieru. Papier pochodzący z recyklingu jest ciemniejszy, ponieważ zawiera resztki farby drukarskiej. Makulaturę można przerobić na papier od 5 do 7 razy.

W kroku 4 przetwarzamy tekst na formę wizualną:

- dzielimy tekst na bloki znaczeniowe,
- zaznaczamy słowa oznaczające obiekty (rzeczowniki z określeniami) istotne dla przekazu (wyróżnijmy je kolorem pomarańczowym),
- zaznaczamy powiązania między obiektami (na zielono).

Tytuł: Recykling papieru

Wprowadzenie: Co roku ścina się tysiące drzew, aby wyprodukować z nich papier na gazety, książki i opakowania. Tymczasem znaczną część makulatury można by ponownie wykorzystać.

1. Zużyty papier wyrzucamy do niebieskich pojemników na śmieci.
2. Odpady papierowe muszą zostać rozdrobnione na małe kawałeczki.
3. Skrawki papieru miesza się z wodą i gotuje, aż powstanie płynna miazga (pulpa).
4. Tę mokrą pulpę rozsmarowuje się bardzo cienko na metalowej siatce.
5. Następnie pulpa jest ściskana i suszona – tak powstaje nowy arkusz papieru.

Dodatkowe informacje: Papier pochodzący z recyklingu jest ciemniejszy, ponieważ zawiera resztki farby drukarskiej. Makulaturę można przerobić na papier od 5 do 7 razy.

Dzięki przetworzeniu tekstu łatwiej będzie sporządzić szkic (krok 5): pomarańczowe elementy można zwizualizować wprost, zielone zaś oznaczają fazy procesu – należy rozważyć, czy ich zobrazowanie pomoże odbiorcy ten proces zrozumieć.

Po przemyśleniu tematu oraz analizie przygotowanego tekstu możemy zdecydować się na następujące rozwiązania:

- Recykling papieru jest procesem, dlatego uporządkujemy informacje **według czasu**.
- Ponieważ mamy do czynienia z procesem powtarzalnym, zastosujemy okrąg, na którym osadzimy kolejne etapy.
- Ponumerujemy etapy procesu zgodnie z ruchem wskazówek zegara.
- Informację o kolejności etapów recyklingu wzmocnimy przez dodanie strzałek na okręgu. Będą one szczególnie ważne dla odbiorcy podczas przechodzenia z etapu ostatniego (gotowy arkusz papieru) do pierwszego (zużyty papier).
- Infografika nie wymaga użycia wielu kolorów. Jako główny wybierzemy niebieski, ponieważ jest to kolor pojemnika na makulaturę.
- Użyty kolor będzie miał **funkcję akcentującą** – zwróci uwagę odbiorcy na kolejne etapy procesu oraz utrwali skojarzenie z kolorem pojemnika, do którego wrzuca się odpady papierowe.

Dobra rada

Szkic możesz wykonać odręcznie na kartce. Zadbaj o ogólną kompozycję, nie skupiaj się na zbyt wielu detalach. Samą infografikę możesz stworzyć w programie do grafiki wektorowej, np. Inkscape. W wypadku prostych infografik dobrze sprawdzą się również narzędzia graficzne edytorów prezentacji, np. Microsoft PowerPoint.

Sposoby porządkowania informacji, s. 220

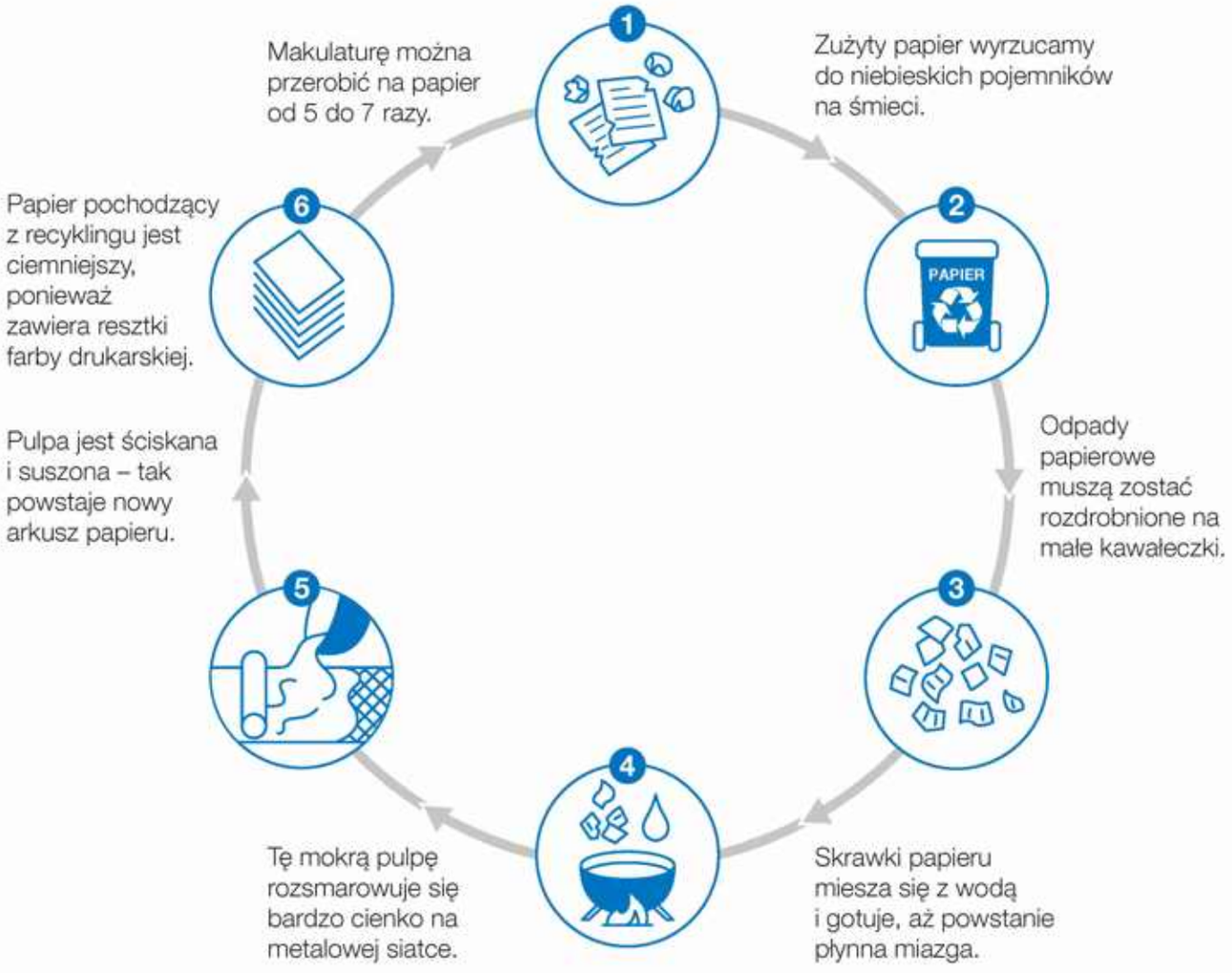
Akcentująca funkcja koloru, s. 226

Następnie należy przygotować infografikę, poddać ją krytycznej analizie i wprowadzić poprawki (kroki 6 i 7). Jeśli mamy taką możliwość, warto pokazać projekt osobie, która reprezentuje docelową grupę odbiorców. Można się w ten sposób dowiedzieć, czy infografika realizuje założony cel.

Oto przykładowy efekt końcowy.

Recykling papieru

Co roku ścina się tysiące drzew, aby wyprodukować z nich papier na gazety, książki i opakowania. Tymczasem znaczną część makulatury można by ponownie wykorzystać. Co się dzieje z makulaturą, gdy poprawnie segregujemy śmieci?



Ćwiczenie 11

Stwórz szkic infografiki, która omówi:

a. zmiany oświetlenia Ziemi wynikające z ruchu obrotowego,
b. obieg wody w przyrodzie,
c. pojęcie łańcucha pokarmowego na dwóch różnych przykładach.

Podsumowanie

- Informacje można przedstawiać na różne sposoby. Jednym z nich jest połączenie tekstu z obrazem w formie grafiki informacyjnej (infografiki).
- Informacje na infografice powinny być uporządkowane. Wyróżniamy uporządkowanie: alfabetyczne, chronologiczne, według kategorii, przestrzenne i hierarchiczne.
- Przygotowanie atrakcyjnych wizualnie infografik wymaga znajomości zasad doboru barw oraz funkcji, jakie mogą pełnić kolory.
- W projektowaniu infografiki należy szczególną uwagę zwracać na to, czy będzie ona zrozumiała dla potencjalnego odbiorcy. Skuteczne przekazanie informacji jest ważniejsze niż atrakcyjność wizualna.

Zadania

* **1** Poszukaj reprodukcji infografiki z 1869 r. autorstwa francuskiego inżyniera Charles'a Minarda. Ukazuje ona marsz armii Napoleona na Moskwę z 1812 r. oraz jej dramatyczny odwrót – wyścig z czasem podczas mroźnej zimy. Postaraj się odczytać z infografiki jak najwięcej informacji.

* **2** Przygotuj prostą grafikę porównawczą, na której wskażesz podobieństwa i różnice:
a. dwóch wybranych dyscyplin sportowych (np. tenisa ziemnego i stołowego),
b. dwóch form nauki – zdalnej i stacjonarnej,
c. dwóch wybranych stylów w modzie (np. sportowego i boho).

* **3** Omów techniki zastosowane na wybranej infografice. Możesz skorzystać z serwisu http://visualoop.com, w którym publikowane są prace autorów z całego świata.

* **4** Przygotuj opis procesu smażenia naleśników w postaci infografiki.

* **5** Odszukaj piktogramy wykorzystywane w twojej szkole. Określ ich funkcję.

** **6** Wyszukaj niezbędne informacje, a następnie opracuj infografikę, na której przedstawisz swoją ulubioną aktorkę oraz najważniejsze filmy, w których występowała. Użyj co najmniej dwóch sposobów porządkowania informacji.

** **7** Wykonaj infografikę przedstawiającą roczne wydatki gminy, w której mieszkasz. Uwzględnij m.in. nakłady na drogi, gospodarkę komunalną, przedszkola i szkoły, kulturę i biblioteki, gospodarkę gruntami, opiekę społeczną, kulturę fizyczną.

*** **8** Przygotuj infografikę przedstawiającą proces publikacji wywiadu w czasopiśmie (np. w tygodniku) – od przygotowań przed rozmową do ukazania się numeru.

*** **9** Opracuj infografikę, która porówna fabułę przynajmniej 6 klasycznych baśni dla dzieci. Pokaż podobieństwa i różnice.

Rozwiązywanie problemów z wykorzystaniem komputera

Wyobraź sobie, że...



...po studiach medycznych zamierzasz zostać chirurgiem. Trafiasz na rezydenturę do szpitala, w którym operuje się również za pomocą **specjalistycznych robotów** chirurgicznych. 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.



...pracujesz w dziale promocji miasta. Kierujesz zespołem przygotowującym kampanię, która ma przyciągnąć turystów oraz zachęcić mieszkańców do odkrywania miasta na nowo. Z pomocą grafików opracowujecie **identyfikację wizualną**, m.in. rozpoznawalne **logo**, które od tej pory będzie widnieć na wszystkich materiałach informacyjnych. Kręcicie także spot promocyjny do mediów społecznościowych i wymyślacie oryginalne hasło, które rozpowszechniacie w formie **hashtagu**.

...jesteś grafikiem i pracujesz jako freelancer. Od portalu informacyjnego dostajesz zlecenie na **infografikę** o zmianach demograficznych w Polsce. Zapoznajesz się z otrzymanymi materiałami i wpadasz na pomysł **wizualizacji danych**. Najpierw **porządkujesz informacje**, a następnie przygotowujesz **szkic**. Odpowiednio **dobierasz kolory** i **akcentujesz** to, co najważniejsze. Twoja praca podoba się zarówno redaktorom, jak i czytelnikom portalu, dlatego otrzymujesz kolejne zlecenie.





...skończyłaś studia przyrodnicze i chcesz popularyzować wiedzę o kryzysie klimatycznym. Na blogu publikujesz rzetelne i przystępne **teksty informacyjne**. Z czasem otwierasz kanał wideo, na którym organizujesz m.in. **webinaria**, a także tworzysz tematyczny **podcast**. Stosujesz zasady **pracy z kamerą** i **mikrofonem** poznane w szkole, dlatego twoje materiały od początku są na wysokim poziomie. Za pomocą **narzędzi analitycznych** regularnie śledzisz ruch w sieci, aby jak najlepiej dopasowywać treści do swoich odbiorców.

Projekt zespołowy

Analiza postępu technologicznego w ostatnich latach

Technologia, z której korzystamy na co dzień, rozwija się właściwie nieustająco. Postęp ten najczęściej jest ewolucyjny, więc nie zawsze go zauważamy. Warto się jednak zastanowić, co sprawia, że niektóre technologie się przyjmują, a inne nie. Jeśli nauczymy się identyfikować takie czynniki, łatwiej nam będzie podejmować życiowe decyzje, np. o wyborze studiów czy założeniu firmy. Choć przewidywanie przyszłości w wypadku rozwoju technologii jest zadaniem niezmiernie trudnym, warto spróbować się z nim zmierzyć.



Cel projektu

Analiza czynników, które spowodowały, że dana technologia odniosła sukces lub poniosła porażkę w stosunku do wcześniejszych przewidywań. Wyniki należy przedstawić w wybranej formie, np. raportu, filmu, prezentacji z nagraną narracją, podcastu.

To trzeba wiedzieć

Analiza trendów

O zmianach w popularności danej technologii można wiele powiedzieć na podstawie tego, jak często hasła z nią związane są wyszukiwane w internecie. Trendy te można prześledzić np. w Google Trends. Analiza archiwalnych artykułów w serwisach informacyjnych pomoże poznać przyczyny zmian i zrozumieć, co wpływało na zainteresowanie daną technologią – jakie wydarzenia, ludzkie potrzeby, sytuacja na świecie.

Badania ankietowe

Więcej informacji można zdobyć, przeprowadzając ankiety wśród rodziny i znajomych w różnym wieku, z różnym wykształceniem i mieszkających w różnych miejscach. Można do tego wykorzystać Formularze Google lub Microsoft Forms. Warto w nich uwzględnić pytania otwarte, aby umożliwić respondentom dołączenie własnych przemyśleń. Ankietę można też przeprowadzić w formie rozmowy – osobiście, telefonicznie lub za pomocą komunikatora internetowego.

Propozycje tematów

1. Jedna z dziedzin opracowanych w ramach projektu w klasie 1: wirtualna oraz rozszerzona rzeczywistość, automatyka i robotyka, druk 3D, sztuczna inteligencja, autonomiczne pojazdy, inteligentne miasta
2. Inna technologia wybrana w ramach projektu w klasie 1
3. Wideo na żądanie (VOD)
4. Filmy 360 stopni oraz filmy interaktywne

Etapy projektu

1. Wybierzcie w zespole technologię (oraz związany z nią produkt lub usługę), której rozwój będziecie analizować.
2. Poszukajcie materiałów opisujących początki wybranej technologii oraz analiz przewidujących jej rozwój i wpływ na ludzi – wybrane grupy społeczne lub całe społeczeństwo (w kraju, na świecie). Dane te powinny być starsze niż 3 lata.
3. Znajdźcie bieżące dane dotyczące wybranej przez was technologii. Mogą to być raporty i artykuły prasowe zawierające informacje o prawdopodobnych przyczynach jej sukcesu i szybkiego rozwoju oraz jej wpływie na różne aspekty życia ludzi. Odnieście się również do własnych doświadczeń i przeprowadźcie badania wśród rodziny i znajomych (ankiety, rozmowy, wywiady).
4. Porównajcie przewidywania dotyczące rozwoju wybranej usługi i jej wpływu na życie ludzi z tym, jak rzeczywiście jest dzisiaj. Przeprowadźcie burzę mózgów, aby zidentyfikować przyczyny takiego rozwoju wydarzeń.
5. Zdecydujcie, w jakiej formie przedstawicie swoje wnioski.
6. Podzielcie się zadaniami, wybierzcie narzędzia odpowiednie do pracy zespołowej oraz wybranej przez was formy przekazu (tekst, film).
7. W opracowaniu uwzględnijcie, jaki wpływ (pozytywny, negatywny lub żaden) miała wybrana technologia na różne dziedziny życia, np. edukację, opiekę zdrowotną, przemysł czy rolnictwo. Podajcie konkretne przykłady, które pokazują, czy przewidywania z przeszłości były trafne czy nie.
8. Do opracowania i zaprezentowania wniosków wykorzystajcie techniki poznane w trakcie nauki dotyczące różnych form publikowania (np. tworzenia prezentacji, animacji czy stron WWW).
9. Gotową pracę opublikujcie w sieci w taki sposób, by dostęp do niej miały jedynie pozostałe osoby w klasie oraz nauczyciel.

Przykładowe narzędzia

Microsoft Teams, **Google Suite** – do pracy zespołowej, wymiany plików, komunikacji oraz udostępniania przygotowanych treści.
Microsoft Forms, **Formularze Google** – do przygotowania i przeprowadzenia ankiet.
Microsoft Excel, **Arkusze Google** – do zebrania i analizy wyników oraz przedstawienia ich na wykresach.
Microsoft PowerPoint, **Prezentacje Google** – do opracowania prezentacji multimedialnej przedstawiającej najważniejsze fakty o danej technologii.
Microsoft Stream – do opublikowania filmu dla zamkniętej grupy odbiorców.

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 można zastosować dla dowolnych dwóch liczb, np. 21 · 57, 123 · 87. Algorytm musi być precyzyjny, choć może mieć różną formę.

Opis słowny

Algorytmy można opisywać słownie. Przykładem może być instrukcja obsługi kuchennego minutnika: „Wykonać pełen obrót wskaźnika aż do pozycji 0. Następnie ustawić wskaźnik na wymaganą liczbę minut".

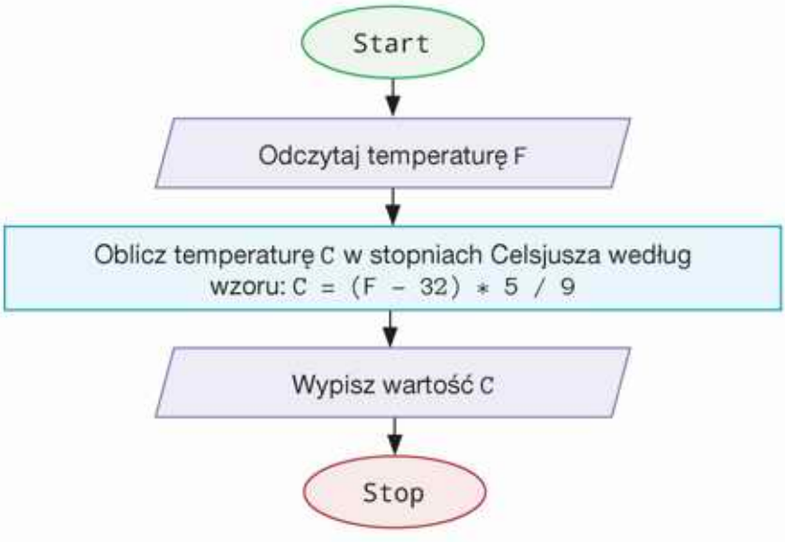
Lista kroków

Lista kroków algorytmu to opis postępowania podzielony na poszczególne kroki, które mogą się odwoływać do siebie nawzajem.
Oto lista kroków algorytmu zaplatania warkocza klasycznego:

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

Schemat blokowy (sieć działań)

Schemat blokowy algorytmu jest zbudowany z umownych kształtów oraz strzałek pokazujących kierunek wykonywania operacji. Oto przykład schematu blokowego algorytmu przeliczania temperatury ze skali Fahrenheita na skalę Celsjusza:



Pseudokod (pseudojęzyk)

Pseudokod to zapis przypominający kod języka programowania, ale pozbawiony szczegółów związanych z implementacją w języku programowania. Przedstawienie algorytmu w postaci pseudokodu ułatwia później zapisanie tego algorytmu w języku programowania. Nie istnieje jedna, powszechnie przyjęta konwencja zapisu pseudokodu. Podstawową zasadą jest jednak wybór intuicyjnych oznaczeń i konsekwentne ich wykorzystanie.
Oto pseudokod algorytmu wyznaczania części całkowitej $\sqrt{n}$ dla dodatniej liczby całkowitej n:

```
wczytaj n
p ← 1
dopóki (p + 1) · (p + 1) ≤ n wykonuj p ← p + 1
wypisz p
```

W powyższym zapisie strzałka (←) oznacza instrukcję przypisania wartości do zmiennej. Symbole +, · i ≤ mają takie samo znaczenie jak w matematyce.

Kod źródłowy

Kod źródłowy to sformalizowany zapis algorytmu zgodny z regułami wybranego języka programowania. Kody źródłowe zazwyczaj zapisuje się w plikach tekstowych. Oto przykłady kodów źródłowych algorytmu obliczania całkowitego pierwiastka kwadratowego z liczby n, zapisane w językach C++ i Python 3:

C++

```
1.  int main()
2.  {
3.      int n; cin >> n;
4.      int p = 1;
5.      while ((p + 1)*(p + 1) <= n)
6.      {
7.          p = p + 1;
8.      }
9.      cout << p << endl;
10.     return 0;
11. }
```

Python

```
1.  n = int(input())
2.  p = 1
3.  while (p + 1)*(p + 1) <= n:
4.      p = p + 1
5.  print(p)
```

Języki programowania, podobnie jak języki naturalne, mają słownictwo podstawowe, czyli **słowa kluczowe**, oraz **składnię**, czyli reguły tworzenia wypowiedzi.

Język programowania służy do porozumiewania się człowieka i komputera. W tej komunikacji pośredniczą specjalne **translatory**, czyli programy tłumaczące, gdyż docelowo kod źródłowy musi być przetłumaczony na **kod maszynowy**, zapisany binarnie, właściwy dla architektury procesora (CPU) i systemu operacyjnego.

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

Wszystkie możliwe przypadki wartości logicznych zdań składowych i odpowiadające im wartości logiczne zdań złożonych można zestawić w tabeli:

p	*q*	*~p*	*p lub q*	*p i q*
1	0	0	1	0
0	1	1	1	0
1	1		1	1
0	0		0	0

Operatory porównań (relacji) w językach C++ i Python

Porównywanie dwóch wartości w językach programowania wykonuje się za pomocą operatorów porównań (relacji). Wynikiem ich działania są wartości `true` albo `false` w języku C++ lub `True` albo `False` w języku Python.

Zapis w matematyce	Zapis w językach C++ i Python
=	==
≠	!=
>	>
<	<
≥	>=
≤	<=

Operatory logiczne w języku C++

Spójniki logiczne („i", „lub", „nieprawda, że") nazywa się operatorami logicznymi. Umożliwiają one stosowanie wyrażeń logicznych w tworzonym kodzie. Język C++ pozwala je łączyć z operatorami porównań przedstawionymi na s. 243.

Nazwa operatora logicznego	Zapis w języku C++	Przykład w języku C++
AND (koniunkcja)	`&&`	Dla zmiennej `x` o wartości 3 i zmiennej `y` o wartości 2 wyrażenie `x < 5 && y > 1` ma wartość `true`.
OR (alternatywa)	`||`	Dla zmiennej `x` o wartości 5 i zmiennej `y` o wartości 5 wyrażenie `x == 5 || y == 5` ma wartość `true`.
NOT (negacja)	`!`	Dla zmiennej `x` o wartości 3 wyrażenie `!(x == 4)` ma wartość `true`.

W języku C++ zamiast operatorów `&&`, `||` oraz `!` można używać odpowiednio operatorów: `and`, `or` i `not`.

Operatory logiczne w języku Python

Spójniki logiczne („i", „lub", „nieprawda, że") nazywa się operatorami logicznymi. Umożliwiają one stosowanie wyrażeń logicznych w tworzonym kodzie. Język Python pozwala je łączyć z operatorami porównań przedstawionymi na s. 243.

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`.

3. Typy danych w języku C++

W języku C++ każda zmienna, niezależnie od tego, czy przechowuje liczbę całkowitą, liczbę z częścią ułamkową, znak, napis czy wartość logiczną, musi mieć 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.

Nazwa typu	Przeznaczenie zmiennej	Deklaracja zmiennej w języku C++
`int`	Liczby całkowite z zakresu od –2 147 483 648 do 2 147 483 647	`int a = 4;`
`long long`	Duże liczby całkowite nie większe niż $9{,}22 \cdot 10^{18}$	`long long a = 300000000000;`
`float`	Ułamki dziesiętne i ich przybliżenia	`float a = -0.5;`
`float`	Liczby z częścią ułamkową, w których liczba cyfr znaczących (pewnych) jest nie większa niż 6	`float a = 2.45123;`
`double`	Bardzo małe liczby, które trzeba zapamiętać z większą dokładnością (nie więcej niż 15 cyfr znaczących)	`double a = 4.00000023e-12;`
`double`	Bardzo duże liczby (z częścią ułamkową) lub liczby, które trzeba zapamiętać z większą dokładnością (nie więcej niż 15 cyfr znaczących)	`double a = 4.00000023e12;`

Uwaga: Zapisy typu `e12` oraz `e-12` oznaczają notację wykładniczą (tj. odpowiednio czynniki 10^{12} oraz 10^{-12}).

Typ znakowy

Pojedyncze znaki można zapisywać w zmiennych typu **`char`**. Przechowują one kody liczbowe znaków z tabeli ASCII. Możemy więc przypisać im wartość, podając kod ASCII, na przykład `char litera=97;`, lub wpisując znak w apostrofach, na przykład `char litera='a';` to deklaracja zmiennej, w której następnie zapisano literę `a`.

Typ napisowy

Napisy, czyli tzw. łańcuchy znaków, zapisuje się z użyciem zmiennych typu **`string`**. Wartość takiej zmiennej można nadać za pomocą operatora przypisania: `napis = "ABC";`. Napisy można scalać za pomocą operatora `+`. Na przykład operacja `"ABC" + "DE"` skutkuje utworzeniem zmiennej o wartości `"ABCDE"`.

Typ napisowy nie jest jednym z typów podstawowych C++, jak np. **`char`**. Typ **`string`** opracowano według koncepcji programowania obiektowego (patrz dodatek 7). Możemy więc wykorzystywać metody dostępne dla tego typu. Na przykład dla zmiennej zadeklarowanej jako **`string`** `napis="ABC";` liczbę znaków napisu wypiszemy na ekranie, jeśli użyjemy w kodzie zapisu **`cout`**`<<napis.length();`.

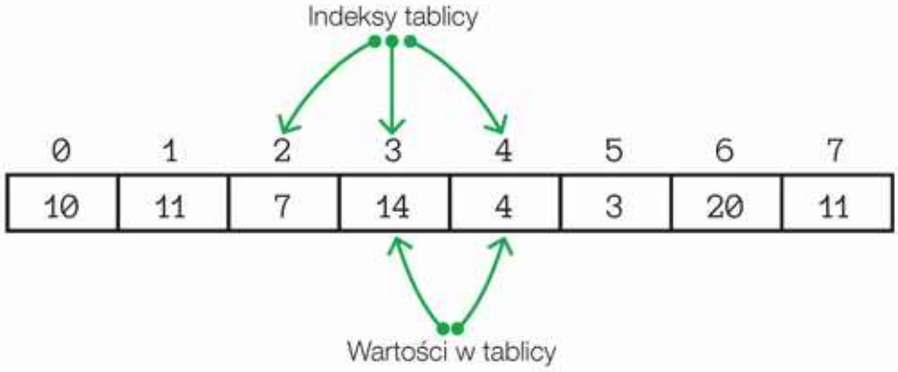
Aby odwołać się do konkretnych znaków w łańcuchu, zapisujemy nazwę zmiennej i numer znaku w nawiasach kwadratowych (znaki numerowane są od 0). Na przykład `cout << napis[1];` wyświetli na ekranie literę B.

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

4. Podstawowe typy instrukcji w języku C++

Instrukcja warunkowa

Składnia prostej instrukcji warunkowej:

Pojedyncza instrukcja	Wiele instrukcji

`if (warunek) instrukcja;`	`if (warunek)`
	`{`
	`  instrukcja1;`
	`  instrukcja2;`
	`  ...`
	`}`

Warunek podawany po słowie **if** powinien być wyrażeniem typu logicznego (patrz dodatek 2).

Składnia instrukcji warunkowej w postaci alternatywy:

Pojedyncza instrukcja	Wiele instrukcji

`if (warunek)`
`  instrukcja na tak;`
`else`
`  instrukcja na nie;`
	`if (warunek)`
	`{`
	`  instrukcja1;`
	`  instrukcja2;`
	`  ...`
	`}`
	`else`
	`{`
	`  instrukcja3;`
	`  instrukcja4;`
	`  ...`
	`}`

W przypadku zagnieżdżania instrukcji warunkowych warto zadbać o odpowiednie wcięcia.

Instrukcje powtarzania. Pętla for

Składnia pętli **for** powtarzającej pojedynczą instrukcję:

```
for (licznik; warunek powtarzania; krok) instrukcja;
```

Składnia pętli **for** powtarzającej wiele instrukcji:

```
for (licznik; warunek powtarzania; krok)
{
    instrukcja1;
    instrukcja2;
    ...
}
```

Instrukcje powtarzania. Pętla `while`

Składnia pętli **`while`** powtarzającej pojedynczą instrukcję:

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

Składnia pętli **`while`** powtarzającej wiele instrukcji:

```
while (warunek logiczny)
{
    instrukcja1;
    instrukcja2;
    ...
}
```

Instrukcja iteracyjna. Pętla `do while`

Składnia pętli **`do while`** powtarzającej pojedynczą instrukcję:

```
do instrukcja while (warunek logiczny);
```

Składnia pętli **`do while`** powtarzającej wiele instrukcji:

```
do
{
    instrukcja1;
    instrukcja2;
    ...
} while (warunek logiczny);
```

Instrukcje sterujące. Funkcje

Funkcje to podprogramy zawarte w programie głównym, które odpowiadają za realizację określonej części całego programu. Funkcje definiuje się przed definicją funkcji `main`.

Definicja funkcji składa się z nagłówka funkcji i bloku instrukcji. Nagłówek zawiera typ wartości zwracanej przez funkcję, nazwę funkcji oraz listę argumentów wraz z typami i nazwami argumentów. Budowa funkcji:

```
typ nazwa_funkcji(typ argument1, typ argument2, ...)
{
    instrukcja1;
    instrukcja2;
    ...
    return wynik;
}
```

Wartość zwracana przez funkcję jest wskazana po słowie **`return`**.
Na przykład instrukcja `return a;` oznacza, że funkcja zwróci wartość zmiennej `a`.

Gdy typem funkcji w nagłówku jest **`void`**, funkcja wykonuje zadania, które nie wymagają zwrócenia wartości wynikowej.

5. 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.

6. Słowa kluczowe w języku C++

Słowa kluczowe nie mogą być nazwami zmiennych, funkcji ani innych obiektów w kodzie źródłowym.

Słowo kluczowe	Opis
`bool`	Nazwa typu logicznego
`const`	Używane przy definiowaniu stałej
`double`	Nazwa typu dla liczb z częścią ułamkową, o tzw. podwójnej precyzji
`else`	Używane w instrukcji warunkowej do wyodrębnienia kodu wykonywanego, gdy warunek po `if` nie jest spełniony
`false`	Stała oznaczająca wartość logiczną fałsz
`float`	Nazwa typu dla liczb z częścią ułamkową
`for`	Używane w instrukcji iteracyjnej
`if`	Używane w instrukcji warunkowej
`int`	Nazwa typu dla liczb całkowitych
`long`	Nazwa typu dla liczb całkowitych, definicja z `long long int` pozwala zapisywać w pamięci komputera liczby z większego zakresu liczb
`namespace`	Używane przy określaniu tzw. przestrzeni nazw
`return`	Używane w instrukcji, która pozwala funkcji zwrócić wartość do miejsca jej wywołania
`true`	Stała oznaczająca wartość logiczną prawda
`using`	Używane przy określaniu tzw. przestrzeni nazw
`void`	Używane w nagłówku funkcji, która nie zwraca wartości
`while`	Używane w instrukcji iteracyjnej

Inne słowa kluczowe języka C++

`asm`	`auto`	`break`	`case`
`catch`	`char`	`class`	`const_cast`
`continue`	`default`	`delete`	`do`
`dynamic_cast`	`enum`	`explicit`	`export`
`extern`	`friend`	`goto`	`inline`
`mutable`	`new`	`operator`	`private`
`protected`	`public`	`register`	`reinterpret_cast`
`short`	`signed`	`sizeof`	`static`
`static_cast`	`struct`	`switch`	`template`
`this`	`throw`	`try`	`typedef`
`typeid`	`typename`	`union`	`unsigned`
`virtual`	`volatile`	`wchar_t`	

7. 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.

8. 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 12). 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`

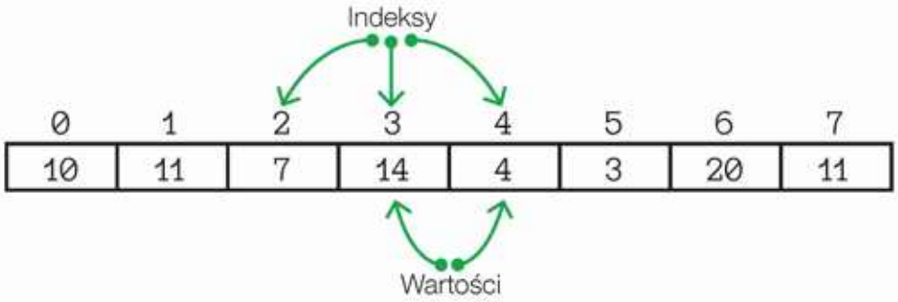
Uwaga: Zapis `e-12` oznacza notację wykładniczą (tj. czynnik 10^{-12}).

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]`.

9. Podstawowe typy instrukcji w języku Python

Instrukcja warunkowa

Składnia prostej instrukcji warunkowej:

```
if warunek:
    instrukcja1
    instrukcja2
    instrukcja3
    ...
```

Warunek podawany po słowie **if** powinien być wyrażeniem typu logicznego (patrz dodatek 2).

Składnia instrukcji warunkowej w postaci alternatywy:

```
if warunek:
    instrukcja1
    instrukcja2
    ...
else:
    instrukcja3
    instrukcja4
    ...
```

W przypadku zagnieżdżania instrukcji warunkowych trzeba zadbać o to, aby kolejne wcięcia były wielokrotnościami pojedynczego wcięcia.

```
if warunek1:
    instrukcja1
    if warunek2:
        instrukcja2
    else:
        instrukcja3
else:
   instrukcja4
```

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))
```

10. Operatory i funkcje matematyczne w języku Python

Operatory arytmetyczne

Operator	Nazwa operatora	Wyrażenie	Wynik
+	Dodawanie	`2 + 3` `2 + 3.5`	5 5.5
–	Odejmowanie	`3 - 2` `3.5 - 2`	1 1.5
*	Mnożenie	`2 * 3` `2.5 * 3.5`	6 8.75
**	Potęgowanie	`2 ** 3`	8
/	Dzielenie	`3 / 2` `3.0 / 2`	1.5 1.5
//	Dzielenie całkowite	`3 // 2` `-3 // 2`	1 –2
%	Reszta z dzielenia (modulo)	`3 % 2`	1

Wybrane wbudowane funkcje matematyczne

Działanie	Wyrażenie	Wynik	Uwagi
Potęgowanie	`pow(2.0,-2)`	0.25	Drugi argument funkcji to wykładnik potęgi.
Wartość bezwzględna	`abs(2)` `abs(-2)`	2 2	
Konwersja na liczbę typu całkowitego	`int(12.5)` `int("12")`	12 12	

Zaokrąglenie	`round(2.5)`
	`round(2.6)`
	`round(3.5)`
	`round(3.6)`
		2
		3
		4
		4
			Liczby z częścią ułamkową 0,5 program zaokrągla inaczej dla liczb o części całkowitej parzystej i nieparzystej.

Wybrane funkcje matematyczne z biblioteki `math`

Działanie	Wyrażenie	Wynik	Uwagi
Pierwiastek kwadratowy	`sqrt(2.0)`	1.41421	Zwracana jest wartość przybliżenia.
Logarytm dziesiętny	`log10(2.0)`	0.30103	Zwracana jest wartość przybliżenia.
Zaokrąglenie w górę (sufit)	`ceil(2.5)` `ceil(-2.5)`	3 –2	
Zaokrąglenie w dół (podłoga)	`floor(2.5)` `floor(-2.5)`	2 –3	

11. Słowa kluczowe w języku Python

Słowa kluczowe nie mogą być nazwami zmiennych, funkcji ani innych obiektów w kodzie źródłowym.

Słowo kluczowe	Opis
`and`	Operator logiczny
`def`	Używane w definicji funkcji
`else`	Używane w instrukcji warunkowej do wyodrębnienia kodu wykonywanego, gdy warunek po `if` nie jest spełniony
`for`	Używane w instrukcji iteracyjnej
`from`	Używane przy imporcie fragmentu biblioteki (modułu)
`if`	Używane w instrukcji warunkowej
`import`	Używane przy imporcie biblioteki (modułu)
`in`	Używane do odwołania się do elementów listy
`is`	Używane do sprawdzania, czy dwa obiekty są tymi samymi obiektami, np. czy zwracana wartość jest równa `True`
`not`	Operator logiczny
`or`	Operator logiczny
`return`	Używane w instrukcji, która pozwala funkcji zwrócić wartość do miejsca jej wywołania
`True`	Stała oznaczająca wartość logiczną prawda
`while`	Używane w instrukcji iteracyjnej
`False`	Stała oznaczająca wartość logiczną fałsz

Inne słowa kluczowe języka Python

`as`	`assert`	`break`	`class`
`continue`	`del`	`elif`	`except`
field	`finally`	`global`	`lambda`
`None`	`nonlocal`	`pass`	`raise`
`try`	`with`		

12. 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"`

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						
$150_{10} = 10010110_2$						$19374_{10} = 4BAE_{16}$					

Indeks

A

akcelerometr 188
akcentująca funkcja koloru 226
algorytm rekurencyjny 77, 161
— wydawania reszty 66, 151
— wyszukiwania wzorca w tekście 24, 109
analiza częstości liter 42, 127
atak siłowy (brute-force) 41, 126

B

barwy dopełniające 228
— monochromatyczne 228
— pochodne 227
— podstawowe 227
— sąsiadujące 228
— trójkątne 228
— trzeciego rzędu 227
biblioteka `string` 14
błąd przybliżenia 68, 153
brute-force (atak siłowy) 41, 126

C

ciąg Fibonacciego 83, 167
— zdefiniowany rekurencyjnie 80, 164
clickbait 203
czujnik dotykowy 186
— kolorów 186
— podczerwieni 186
— temperatury 186
— ultradźwiękowy 186
— żyroskopowy 186

D

deszyfrowanie 30, 115
drzewo binarne 78, 162

E

emoji 10, 96

F

fraktal 77, 161
funkcja `chr` (Python) 98
— `find` (C++) 20
— `getline` (C++) 39
— `len` (Python) 102
— `length` (C++) 18
— `ord` (Python) 99
— rekurencyjna 81, 165
— `rfind` (C++) 20
funkcja koloru 226
—— akcentująca 226
—— rozróżniająca 226
—— wartościująca 226

G

grafika informacyjna (infografika) 220
— narracyjna 222

H

hashtag 203

I

identyfikacja wizualna 210
indeks ujemny 102
infografika (grafika informacyjna) 220

J

język Isotype 224
— MicroPython 189

K

klasa 106
klucz szyfrowania 30, 115
kod ASCII 11
— liczbowy znaku 10, 96
koło barw 227

konkatenacja (operacja złączenia) 15, 101
konspekt 207
krotka 140
kryptoanaliza 40, 125
kryptografia 30, 115
kryptologia 40, 125

L
leniwe wartościowanie 59, 144
lid 203
listy równoległe 158
logo 210
logotyp 210

Ł
łańcuch (napis, łańcuch znaków) (C++) 14
———— (Python) 97
— pusty 34, 119

M
magnetometr 188
metoda 106
— append (Python) 108
— find (Python) 106
— rfind (Python) 106
— zachłanna (podejście zachłanne) 63, 148

N
napis (łańcuch znaków, łańcuch) (C++) 14
———— (Python) 97
nasycenie koloru 227

O
obiekt 106
operacja prostej zamiany 53, 138
— złączenia (konkatenacja) 15, 101
operator + (C++) 15
—— (Python) 101
organiczne wyniki wyszukiwania 203
oś czasu 225

P
pętla nieskończona 68
pętle zagnieżdżone 26, 112
piktogram 221
podcast 211
podejście zachłanne (metoda zachłanna) 63, 148
porządkowanie (sortowanie) danych 46, 131
problem kinomana 71, 155
— kolorowania mapy 64, 149
— optymalizacyjny 62, 147
— wydawania reszty 64, 149

R
rekurencja (rekursja) 77, 161
robot 183
rozróżniająca funkcja koloru 226
rozwiązanie optymalne 63, 148
— przybliżone 63, 148

S
schemat linii komunikacyjnych 223
— organizacyjny 222
screencast 211
serwomotor 186
skytale 31, 116
sortowanie bąbelkowe 53, 138
— (porządkowanie) danych 46, 131
— przez wstawianie 56, 141
sposoby akcentowania informacji 229
— porządkowania informacji 220
stała 15, 100
— globalna 26
— tablicowa 67
sygnet 210
symulator robota 183
szum tła 215
szyfr Cezara 35, 120
— kolumnowy 32, 117
— podstawieniowy 35, 120
— przestawieniowy 32, 117
szyfrogram 30, 115
szyfrowanie 30, 115

T

tablica ASCII 11, 97
— Unicode 10, 96
tablice równoległe 74
tekst informacyjny 207
— jawny 30, 115
typ `char` (typ znakowy) (C++) 11
— `str` (Python) 97
— `string` (C++) 14

W

wartościująca funkcja koloru 226
warunek początkowy 80, 164
webcast 211
webinarium (webinar) 211
wizualizacja danych 222
wyszukiwanie binarne (wyszukiwanie połówkowe) 51, 136
— sekwencyjne 51, 136
— wzorca w tekście 23, 109
wywołanie rekurencyjne 81, 165

Z

zasada 5W 208

Zdjęcie na okładce: Getty Images/iStock/Getty Images Plus/sakkmesterke.

Zdjęcia: Archiwum NE s. 221 (znaki ostrzegawcze); BE&W: Alamy Stock Photo - ADDICTIVE STOCK CREATIVES s. 213, Dumitru Gornet s. 19, 105, John Frost Newspapers s. 223, Kristoffer Tripplaar s. 181 (samochód autonomiczny), PNC Collection s. 79, 163 (puszka), Sandra Foyt s. 181 (robot da Vinci), Science History Images s. 8, 94, 225 (Grace Hopper), s. 180 (fabryka Forda), s. 83, 167, UtCon Collection s. 8, 94, 225 (Jan Łukasiewicz), EXCLUSIVE PIXExclusivepix Media Ltd/Exclusivepix s. 180 (Pierre Jaquet-Droz), Mary Evans Picture Library s. 182; Forum: Interfoto/Friedrich s. 221 (piktogramy olimpijskie); Getty Images: Archive Photos/Frederic Lewis s. 180 (Frank Rosenblatt), Bettmann s. 8, 94, 225 (komputer ENIAC), DigitalVision – Dave & Les Jacobs s. 92, 176 (prawnik), Luis Alvarez s. 177, E+ - AleksandarGeorgiev s. 92, 176 (pracownik w firmie produkującej gry planszowe), martin-dm s. 212, matejmo s. 43, 128 (kłódka), Hulton Archive/Getty Images s. 181 (łazik Sojourner), imagenavi s. 27, 112, iStock/Getty Images Plus/koya79 s. 179, iStockphoto/Cristian Storto Fotografia s. 180 (tranzystor), Moment/Photography taken by Mario Gutiérrez s.92, 176 (fotograf), Science Photo Library/SCIEPRO s. 32, 117, SSPL/Science & Society Picture Library s. 181 (robot Unimate), Tetra Images s. 92, 176 (pracownik działu administracji), The LIFE Picture Collection/Alfred Eisenstaedt s. 8, 94, 225 (Claude E. Shannon), Ulrich Zillmann s. 8, 94, 225 (kod Baudot-Murraya), Universal Images Group Editorial/PYMCA s. 189, VCG/Visual China Group s. 236 (chirurg); Indigo: Alamy Stock Photo/Pictorial Press Ltd s. 180 (Archytas z Tarentu), Science Photo Library - s. 180 (Alan M. Turing), BRITISH LIBRARY s. 227; Shutterstock: Africa Studio s. 237, akramalrasny s. 230, alphaspirit s. 47, 132, amiak s. 43, 128 (tło), Andreas Nilsson s. 78, 162, Andrey_Popov s. 236 (grafik komputerowy), Anton Gvozdikov s. 181 (robot Sophia), April stock s. 208, Creative Instinct s. 9, 95, 225 (wyszukiwarka), Daniel Beckemeier s. 93, elRoce s. 84, 168 (huragan), Evgeniya Lystsova s. 84, 168 (muszla), feeling lucky s. 204 (dziewczyna przy komputerze), freestyle images s. 195 (łazik), gd_project s. 181 (kostka Rubika), Gekkon s. 85, 169 (Taj Mahal – portal), Here s. 54, 139 (woda), Klavdiya Krinichnaya s. 54, 57, 139, 142 (piksele), KSai23 s. 70, 155, Lenscap Photography s. 8, 94, 225 (Enigma), NDAB Creativity s. 236 (zebranie zespołu), ORION PRODUCTION s. 236 (nagrywanie), Paulo Arsand s. 79, 163 (wałkowanie, składanie ciasta), Picsfive s. 204 (kartka papieru), pisaphotography s. 77, 161, Rawpixel.com s. 205 (ludzie z telefonami), s. 206 (laptop), Sergey Chirkov s. 204 (lampa fotograficzna), Stanislaw Tokarski s. 53, 138, Stefano Garau s. 204 (aparat na statywie), Sumit.Kumar.99 s. 85, 169 (Taj Mahal), tanatat s. 205 (lupa), Ulrich Mueller s. 7, Vmaster s. 206 (rakieta); ISOTYPE link: https://medium.com/nightingale/exploring-isotype-charts-only-an-ocean-between-part-1-399f227e1c69 link: http://isotyperevisited.org/2012/08/introduction.html s. 224; Janusz Mazur s. 183, 186, 188 (roboty edukacyjne); Wikipedia: Photograph by Rama, Wikimedia Commons, Cc-by-sa-2.0-fr s. 9, 95, 225 (Tony Hoare); Maciej Wróbel s. 65, 150.

Na stronach 230–231 wykorzystano infografikę „Konflikty na świecie" zamieszczoną w podręczniku *Dziś i jutro. Podręcznik do wiedzy o społeczeństwie dla szkoły podstawowej* autorstwa Iwony Janickiej, Arkadiusza Janickiego, Aleksandry Kuci-Maćkowskiej, Tomasza Maćkowskiego, s. 212–213.

Wydawnictwo dołożyło wszelkich starań, aby odnaleźć posiadaczy praw autorskich do wszystkich utworów zamieszczonych w podręczniku. Pozostałe osoby prosimy o kontakt z Wydawnictwem.

Wydawnictwo Nowa Era nie ponosi odpowiedzialności za treści i zasoby zamieszczone na cytowanych w tej publikacji stronach internetowych.

nowa era *Twoje mocne strony*

Informatyka na czasie

Podręcznik *Informatyka na czasie 3* do zakresu podstawowego zawiera treści dotyczące algorytmiki i programowania – w językach C++ i Python, a także robotyki, publikowania w sieci i tworzenia infografik. Opanowanie tych zagadnień ułatwi aktywny margines z siatką pojęć oraz ramkami *Dobra rada* i *Warto wiedzieć* – informacjami uzupełniającymi wiadomości z lekcji. W nauce pomogą również atrakcyjne infografiki oraz liczne ilustracje, schematy, zrzuty ekranowe i kody źródłowe.

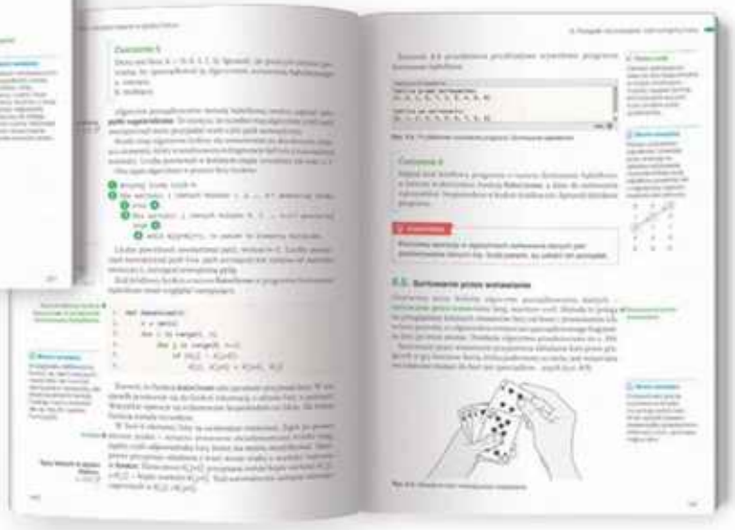


Różnorodne formy przekazu

Rysunki i schematy, a także czytelne zrzuty ekranowe i kody źródłowe ułatwiają zrozumienie i zapamiętanie kluczowych informacji.

Aktywny margines

Zawiera siatkę pojęć, odsyłacze oraz praktyczne wskazówki ułatwiające opanowanie niezbędnych wiadomości.



Informatyka bliska uczniowi

W podręczniku *Informatyka na czasie 3* do zakresu podstawowego trudne zagadnienia są wprowadzane poprzez odwołania do sytuacji bliskich doświadczeniu ucznia. Elementy *A to ciekawe* ukazują informatykę przez pryzmat faktów pozornie niemających z nią związku. Infografiki ujmują ważne treści w przejrzystej i atrakcyjnej dla ucznia formie.



Nowa Era Sp. z o.o.

www.nowaera.pl nowaera@nowaera.pl

Centrum Kontaktu: 801 88 10 10, 58 721 48 00

ISBN 978-83-267-4119-7