

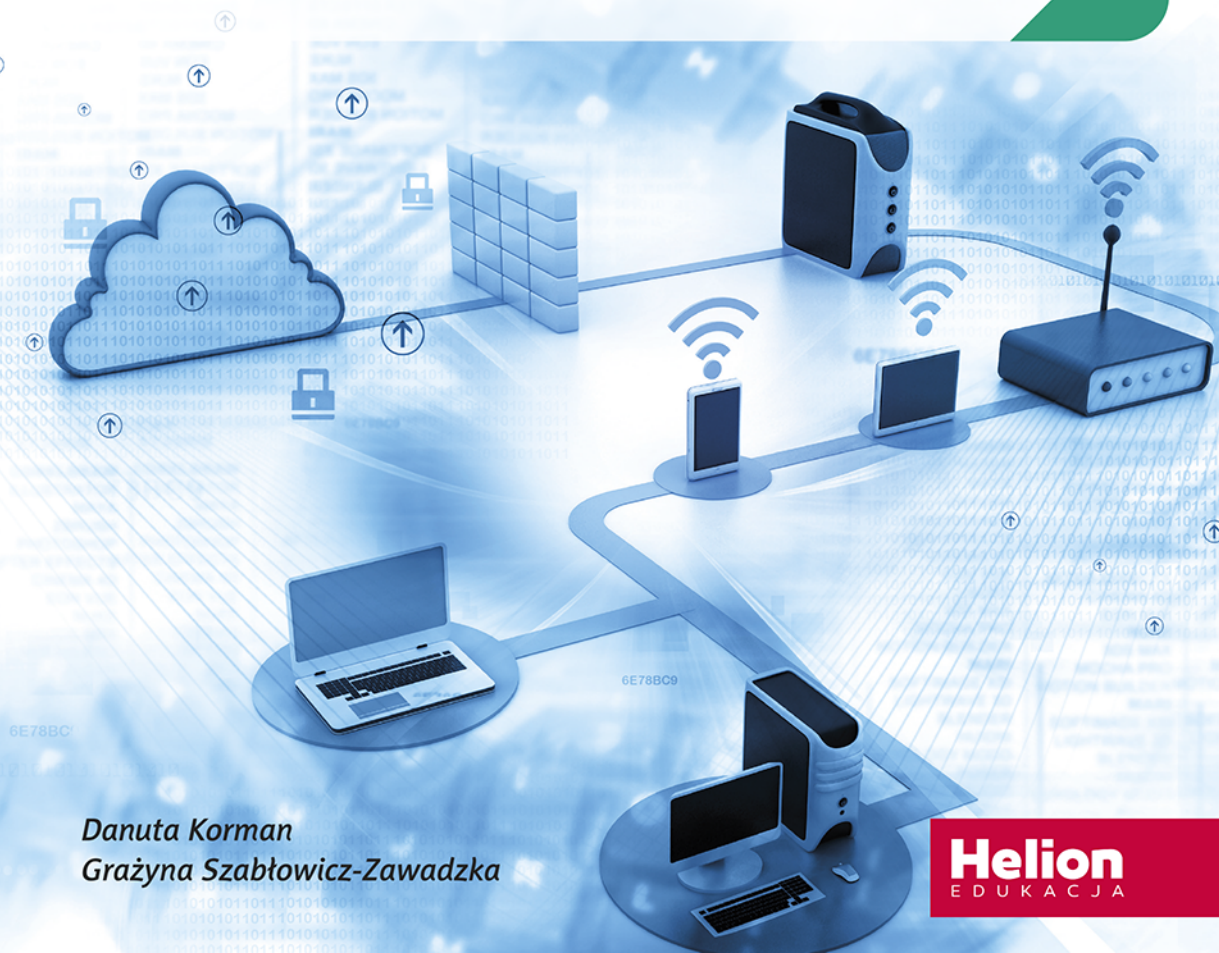
DO NOWEJ PODSTAWY
PROGRAMOWEJ

Część 1

PODRĘCZNIK dla szkół ponadpodstawowych

Informatyka Europejszka

Zakres **rozszerzony**



Danuta Korman
Grażyna Szabłowicz-Zawadzka

Helion
EDUKACJA

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autorki oraz Helion SA dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autorki oraz Helion SA nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Joanna Zaręba

Ilustracja na okładce została wykorzystana za zgodą Shutterstock.

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie?ieppr1>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-4188-3

Copyright © Helion 2020

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

..... Spis treści

Wstęp	9
Rozdział 1. Rozumienie i analizowanie problemów. Algorytmy i ich zastosowanie	11
Temat 1. Sposoby reprezentowania algorytmów	12
1.1. Pseudokod	12
1.2. Rozwiązywanie równania kwadratowego	14
Temat 2. Wprowadzenie do języka C++ — podstawowe konstrukcje i analiza przykładów	22
2.1. Struktura programu	23
2.2. Operacje wejścia-wyjścia	25
2.3. Zmienne, stałe, wskaźniki i referencje	29
2.4. Wyrażenia arytmetyczne, relacje i operatory logiczne	32
2.5. Priorytety relacji i działań	37
2.6. Funkcje matematyczne	38
2.7. Komentarze	40
2.8. Podstawowe konstrukcje algorytmiczne	40
2.9. Proste typy danych	52
2.10. Funkcje w języku C++	53
2.11. Strukturalne typy danych	62
Temat 3. Własności i złożoność obliczeniowa algorytmów	79
3.1. Złożoność obliczeniowa i efektywność algorytmów	79
3.2. Poprawność i skończoność algorytmów	84
3.3. Optymalność algorytmów	85
Temat 4. Iteracja i rekurencja	88
4.1. Obliczanie silni liczby naturalnej	88
4.2. Zastosowania algorytmu Euklidesa	91
4.3. Wieże Hanoi	94
Temat 5. Błędy w obliczeniach komputerowych	99
5.1. Notacja naukowa liczb	99
5.2. Błąd bezwzględny i względny	101

Temat 6. Rozkładanie liczby na czynniki pierwsze i sito Eratostenesa	103
6.1. Faktoryzacja liczby	103
6.2. Sito Eratostenesa	105
Temat 7. Pozycyjne systemy liczbowe i ich zastosowania	108
7.1. Pozycyjne i niepozycyjne systemy liczbowe	108
7.2. Konwersje pozycyjnych systemów liczbowych	110
7.3. Zastosowanie systemu binarnego — szybkie potęgowanie liczb	114
7.4. Operacje arytmetyczne wykonywane w różnych systemach liczbowych ..	116
7.5. Wyznaczanie wartości wielomianu za pomocą schematu Hornera	120
7.6. Zamiana liczb z dowolnego pozycyjnego systemu liczbowego na system dziesiętny z zastosowaniem schematu Hornera	123
Temat 8. Reprezentacja danych liczbowych w komputerze	125
8.1. Reprezentacja liczb naturalnych i całkowitych w komputerze	125
8.2. Stałopozycyjna reprezentacja liczb	125
8.3. Binarna reprezentacja liczb ujemnych	126
8.4. Zmiennopozycyjna reprezentacja liczb	127
Temat 9. Algorytmy liniowe na ciągach liczbowych	129
9.1. Losowe generowanie ciągów liczbowych	129
9.2. Liniowe przeszukiwanie ciągu liczbowego	134
9.3. Liniowe przeszukiwanie ciągu liczbowego z wartownikiem	138
9.4. Znajdowanie minimalnego lub maksymalnego elementu	140
9.5. Sprawdzanie monotoniczności ciągu liczbowego	144
9.6. Lider i idol w zbiorze	146
Temat 10. Algorytmy na tekstach	151
10.1. Palindromy	151
10.2. Sortowanie tekstu	153
10.3. Anagramy	154
Temat 11. Szyfrowanie tekstu	159
11.1. Kryptografia i kryptoanaliza	159
11.2. Szyfrowanie symetryczne	161
11.3. Szyfrowanie asymetryczne	167
Zadania do rozdziału 1.	169

Rozdział 2. Multimedia	171
Temat 12. Cyfrowy zapis dźwięku	171
12.1. Teoria dźwięku	171
12.2. Zapis analogowy dźwięku	173
12.3. Cyfrowy zapis dźwięku	173
Temat 13. Reprezentacja obrazu w komputerze	185
13.1. Co widzi ludzkie oko?	185
13.2. Barwa	186
13.3. Modele barw	186
13.4. Atrybuty barwy	191
13.5. Własności barwy — głębia bitowa (ang. color depth)	195
Temat 14. Kompresja stratna i bezstratna	197
14.1. Kompresja bezstratna	197
14.2. Kompresja stratna	209
Temat 15. Modelowanie 3D	211
15.1. Modelowanie 3D w programie Blender	211
15.2. Renderowanie obrazu i ustawienie kamery	214
Temat 16. Animacje komputerowe	218
16.1. Jak powstaje animacja?	218
16.2. Animacja poklatkowa (ang. stop motion)	219
Zadania do rozdziału 2.	225

Rozdział 3. Arkusz kalkulacyjny

227

Temat 17. Zaawansowane funkcje w przykładach	227
17.1. Składnia funkcji	227
17.2. Widok funkcji	228
17.3. Funkcje generujące dane	228
17.4. Zastosowanie funkcji logicznych	230
17.5. Zastosowanie funkcji czasu	235
17.6. Zastosowanie funkcji daty	240
17.7. Zastosowanie funkcji tekstowych	246
17.8. Zastosowanie funkcji zaokrągleń	253

Temat 18. Makropolecenia i VBA	256
18.1. Nagrywanie makropolecenia	256
18.2. Edycja makra w VBA	259
18.3. Zastosowanie pętli FOR w VBA	262
18.4. Zastosowanie instrukcji warunkowej w VBA	265
18.5. Formularze użytkownika i VBA	268
Zadania do rozdziału 3.	272
Rozdział 4. Relacyjna baza danych	275
Temat 19. Bazy danych i zarządzające nimi systemy	275
19.1. Rodzaje baz danych	276
19.2. Projektowanie relacyjnej bazy danych	278
Temat 20. Tabele w programie Access 2013	284
20.1. Tworzenie tabeli	284
20.2. Typy danych w tabelach	287
20.3. Podstawowe właściwości pól tabeli	291
20.4. Pole kluczowe tabeli	297
20.5. Definiowanie relacji	300
20.6. Operacje na tabelach bazy danych	306
20.7. Zmiana sposobu prezentowania danych	309
Temat 21. Kwerendy — zapytania	313
21.1. Kwerendy wybierające	314
21.2. Kwerendy krzyżowe	325
21.3. Kwerendy funkcjonalne — modyfikujące	329
Temat 22. Język zapytań SQL	335
22.1. Kwerenda oparta na jednej tabeli	335
22.2. Kwerenda wybierająca dane z kilku tabel	337
22.3. Sortowanie	337
22.4. Kryterium wyboru	338
22.5. Pola obliczeniowe	339
22.6. Kwerendy parametryczne	340

22.7. Kwerenda tworząca nową tabelę	340
22.8. Kwerenda krzyżowa	340
22.9. Tworzenie kwerend za pomocą instrukcji języka SQL	341
Temat 23. Formularze	344
23.1. Autoformularze	344
23.2. Projektowanie formularzy	346
Temat 24. Raporty	352
Temat 25. Ochrona i bezpieczeństwo danych	355
Zadania do rozdziału 4.	359
Bibliografia	361
Skorowidz	362

W realizacji programu **informatyki w zakresie rozszerzonym** korzystamy zarówno z podręczników do zakresu podstawowego, jak i rozszerzonego. Nauczyciel omawia materiał z zakresu podstawowego, wplatając w wymagających tego miejscach zagadnienia zawarte w podręcznikach do poziomu rozszerzonego. Rozkład materiału dokładnie wskazuje kolejność omawianych zagadnień. Ponieważ siatka godzin do informatyki rozszerzonej może być różna w różnych szkołach, nie można go podzielić na poszczególne klasy. Nauczyciel, znając przydział godzin lekcyjnych w danym roku, zaplanuje ich rozkład, jednak w sumie musi to być 9 godzin w cyklu kształcenia. Rozkład materiału jest przygotowany dokładnie na tyle.

Na poziomie podstawowym realizowane są 3 godziny lekcyjne w cyklu kształcenia, czyli łącznie 90 godzin. Tutaj realizacja godzin jest dokładnie określona: po 30 godzin w klasach I, II i III.

Na poziomie rozszerzonym dodatkowo dochodzi 6 godzin lekcyjnych w cyklu kształcenia, czyli łącznie 180 godzin. W sumie na tym poziomie jest 9 godzin lekcyjnych w cyklu kształcenia (3+6), czyli 270 godzin. Materiał omawiany w zakresie rozszerzonym obejmuje zarówno zakres podstawowy, jak i rozszerzony.

Podręczniki zawierają **wiele przykładów i ćwiczeń**, na podstawie których uczeń może przystąpić do **rozwiązywania zadań**. Każdy rozdział zaczyna się od podania informacji, co uczeń powinien potrafić po zakończeniu nauki w szkole podstawowej i czego się nauczy na lekcjach na poziomie rozszerzonym. Na końcu każdego rozdziału są **zadania podsumowujące** omówiony materiał.

Na podstawie części pierwszej podręcznika do poziomu rozszerzonego uczeń:

- Znacznie rozszerzy znajomość **tekstowych języków programowania dopuszczonych do egzaminu maturalnego** z informatyki rozszerzonej. Będzie to kontynuacja nauki języka Python. Ponadto zostanie wprowadzony język C++, również dopuszczony do egzaminu maturalnego.
- Zajmie się **własnościami oraz analizą efektywności algorytmów**. Ponadto pozna **źródła błędów występujących w obliczeniach komputerowych**.
- Pozna i będzie stosować **zaawansowane algorytmy iteracyjne i rekurencyjne** z użyciem typów złożonych, realizowane na liczbach i na tekstach. Będzie wykorzystywał nabyte umiejętności w zadaniach z **kryptografii i kryptoanalizy**, analizując i konstruując **algorytmy szyfrujące i deszyfrujące**.
- Pozna sposoby **reprezentowania w komputerze obrazów, dźwięków i animacji**. Nauczy się **edycji dźwięku** i tworzenia dwuwymiarowych oraz trójwymiarowych **wizualizacji i animacji**. Będzie stosować właściwe formaty plików graficznych.
- Pozna **metody kompresji danych**.
- Będzie stosować **zaawansowane funkcje arkusza kalkulacyjnego** w zależności od rodzaju danych. Nauczy się definiowania **makropoleczeń**. Pozna możliwości **wbudowanego języka programowania VBA**.

- Zaprojektuje i stworzy **relacyjną bazę złożoną z wielu tabel**. Będzie formułować kwerendy, tworzyć i modyfikować formularze oraz raporty. Pozna i zastosuje **język SQL** do wyszukiwania informacji w bazie danych. Będzie uwzględniać kwestie **integralności danych, bezpieczeństwa i ochrony danych** w bazie.

Danuta Korman
Grażyna Szabłowicz-Zawadzka

Rozdział 1.

ROZUMIENIE I ANALIZOWANIE PROBLEMÓW. ALGORYTMY I ICH ZASTOSOWANIE

Czego się nauczysz na poziomie rozszerzonym?

Rozszerzysz znajomość **tekstowych języków programowania dopuszczonych do egzaminu maturalnego** z informatyki rozszerzonej. Będzie to kontynuacja nauki języka Python. Ponadto poznasz język C++, również dopuszczony do egzaminu maturalnego, w tym:

- strukturę programu zapisanego w języku C++,
- strumieniowe operacje wejścia-wyjścia i manipulatory strumieniowe,
- wyrażenia arytmetyczne, relacje i operatory logiczne,
- dostęp do bibliotek i funkcje matematyczne w języku C++,
- podstawowe konstrukcje algorytmiczne, na przykład instrukcje warunkowe, instrukcję wyboru, instrukcje iteracyjne, instrukcje sterujące,
- strukturalizację programów, czyli zastosowanie funkcji w języku C++,
- proste typy danych,
- strukturalne typy danych, na przykład tablice, łańcuchy znaków.

Poznasz i będziesz wykorzystywać reprezentację algorytmów w postaci **pseudokodu**.

Zajmiesz się **własnościami i złożonością obliczeniową poznawanych algorytmów**. Będziesz **analizować efektywność algorytmów**. Poznasz pojęcia **poprawności, skończoności i optymalności algorytmów**.

Poznasz tematykę **błędów pojawiających się w obliczeniach komputerowych**, w tym notację naukową, błędy względne i bezwzględne.

Rozszerzysz znajomość **pozycyjnych i niepozycyjnych systemów liczbowych**, w tym konwersje liczb rzeczywistych, konwersje między systemami niedziesiętnymi, operacje arytmetyczne wykonywane w różnych systemach liczbowych.

Poznasz **reprezentacje liczb naturalnych i całkowitych w komputerze**, w tym stałopozycyjną i zmiennopozycyjną reprezentację liczb oraz kod U2.

Poznasz i będziesz stosować **zaawansowane algorytmy iteracyjne i rekurencyjne**, z wykorzystaniem typów złożonych, realizowane na liczbach, ciągach liczbowych i tekstach.

Będziesz wykorzystywać nabyte umiejętności, realizując zadania z **kryptografii i kryptoanalizy**, analizując i konstruując **algorytmy szyfrujące i deszyfrujące**.

..... Temat 1. Sposoby reprezentowania algorytmów

Istnieje wiele sposobów przedstawiania algorytmów. Wybór odpowiedniej metody powinien zależeć od rozwiązywanego problemu. Do wykonywanego zadania należy dobrać taki sposób reprezentowania algorytmu, który najdokładniej i najczytelniej pokaże jego przebieg. Najważniejsze **metody przedstawiania algorytmów wykorzystywane w tym podręczniku** to:

- lista kroków,
- schemat blokowy,
- pseudokod,
- programy w tekstowych językach programowania Python i C++.

Do prezentowania algorytmów można stosować też inne metody, na przykład opis słowny, drzewo algorytmu, drzewo wyrażenia, inne języki programowania. Większość metod zastosowanych w podręczniku poznaliście już w szkole podstawowej. Jedyną nową techniką jest pseudokod, który opisano w podrozdziale 1.1. W tym temacie przypominamy również inne metody, które pojawiają się w podręczniku.

W podręczniku algorytmy będą zapisywane w dwóch językach programowania dopuszczonych do egzaminu maturalnego z informatyki:

- **Python**, którego podstawy już poznaliście,
- **C++**, którego opis jest w temacie 2.

1.1. Pseudokod

Pseudokod to taki sposób zapisu algorytmu, który zachowuje strukturę charakterystyczną dla kodu zapisanego w języku programowania. Rezygnujemy tutaj ze ścisłych reguł składniowych na rzecz prostoty i czytelności. Pseudokod nie zawiera niektórych szczegółów implementacyjnych, takich jak deklaracja zmiennych i alokacja pamięci. Kroki algorytmu opisywane są za pomocą formuł matematycznych lub zdań w języku naturalnym.

Nie istnieją obecnie standardy zapisu pseudokodu. Przeważnie używa się składni, która opiera się na istniejących językach programowania.

Przykład 1.1.

Przeanalizujemy zapisany w postaci pseudokodu prosty algorytm, który po wczytaniu liczby sprawdza, czy jest ona dodatnia. Jeśli warunek jest spełniony, wypisuje wczytaną liczbę, w przeciwnym razie wyświetla informację o błędzie.

Specyfikacja:

Dane: liczba rzeczywista: *liczba*.

Wynik: jeśli liczba jest dodatnia, wypisanie zmiennej *liczba*, w przeciwnym razie wypisanie komunikatu o błędzie.

Pseudokod:

```
wprowadź liczbę
jeżeli liczba jest dodatnia, to
    wypisz liczbę
w przeciwnym razie
    wypisz informację o błędzie
```

Ten sam algorytm możemy zapisać wszystkimi poznanymi metodami. Na rysunku 1.1 przedstawiono schemat blokowy tego algorytmu, a poniżej jego zapis w postaci listy kroków i programów w językach programowania Python i C++. Zauważ, że tylko reprezentacja w postaci programu umożliwi jego uruchomienie. Ponadto zapis algorytmu w pseudokodzie bardzo przypomina kod programu.

Lista kroków:

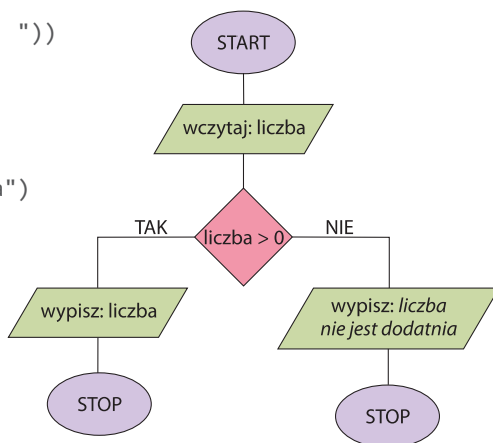
- Krok 1.** Jeżeli *liczba* jest dodatnia, przejdź do kroku 2., w przeciwnym razie przejdź do kroku 3.
- Krok 2.** Wypisz zmienną *liczba* i zakończ algorytm.
- Krok 3.** Wypisz komunikat „liczba nie jest dodatnia” i zakończ algorytm.

Program w języku Python:

```
liczba = float(input("podaj liczbę: "))
if liczba > 0:
    print(liczba)
else:
    print("liczba nie jest dodatnia")
```

Program w języku C++:

```
#include <iostream>
using namespace std;
int main()
{
    double liczba;
    cout << "podaj liczbę: ";
    cin >> liczba;
    if (liczba > 0) cout << liczba << endl;
    else cout << "liczba nie jest dodatnia" << endl;
}
```



Rysunek 1.1. Schemat blokowy algorytmu do przykładu 1.1

1.2. Rozwiązywanie równania kwadratowego

Zajmijmy się analizą i rozwiązaniem równania kwadratowego podawanego przeważnie w postaci $ax^2 + bx + c = 0$. Współczynniki rzeczywiste a , b i c są tutaj danymi wejściowymi. Należy pamiętać, aby współczynnik a był różny od zera. W przeciwnym razie, czyli dla a równego zero, otrzymalibyśmy równanie liniowe.

Znane metody realizujące to zadanie to **algorytm „delty”** oraz **wzory Viète’a**. Jednak aby skonstruować algorytm poprawny dla obliczeń wykonywanych przez komputer, należy wykorzystać obie podane metody. Taki sposób wyznaczania pierwiastków równania kwadratowego nazywamy **stabilnym algorytmem rozwiązującym równanie kwadratowe**.

Analizowany algorytm zapiszemy wszystkimi poznanymi sposobami, które dopuszczone są do egzaminu maturalnego.

Bez względu na zastosowaną metodę rozwiązywania równania kwadratowego specyfikacja tego zadania jest taka sama i ma następującą postać:

Specyfikacja:

Dane: liczba rzeczywista: $a \neq 0$;
dowolne liczby rzeczywiste: b , c .

Wynik: pierwiastki rzeczywiste równania kwadratowego lub komunikat informujący o braku rozwiązania.

1.2.1. Niestabilny algorytm rozwiązujący równanie kwadratowe — algorytm „delty”

Najczęściej stosowaną metodą rozwiązywania równania kwadratowego jest algorytm „delty”. Pierwiastki trójmianu wyznaczone są tutaj z wykorzystaniem „delty”, którą obliczamy następująco:

$$\Delta = b^2 - 4ac.$$

Następnie analizujemy wartość uzyskanej „delty” i w zależności od niej wyznaczamy pierwiastki. Jeśli „delta” jest większa od zera, to otrzymujemy dwa różne pierwiastki x_1 i x_2 , które obliczamy za pomocą podanych poniżej wzorów:

$$x_1 = \frac{-b - \sqrt{\Delta}}{2a},$$

$$x_2 = \frac{-b + \sqrt{\Delta}}{2a}.$$

Jeśli „delta” jest równa zero, to otrzymujemy dwa identyczne pierwiastki x_1 i x_2 , które wyznaczamy za pomocą poniższego wzoru:

$$x_1 = x_2 = \frac{-b}{2a}.$$

Mówimy wówczas, że równanie ma jeden pierwiastek.

W przypadku gdy „delta” jest mniejsza od zera, równanie nie ma rozwiązania w zbiorze liczb rzeczywistych.

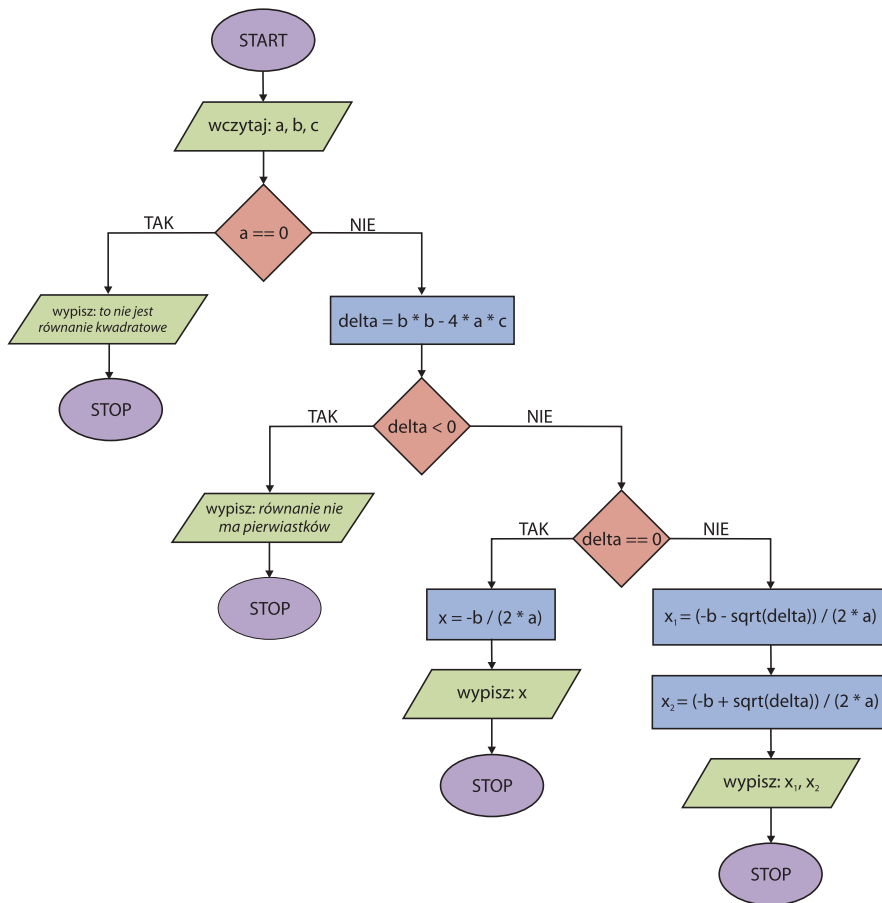
Algorytm „delty” jest najczęściej stosowanym sposobem wyznaczania pierwiastków równania kwadratowego. Metoda ta jednak nie sprawdza się w sytuacji, gdy do obliczeń wykorzystywany jest komputer. Dla niektórych wartości współczynników a , b , c generuje ona niedokładne wartości pierwiastków. Wynika to z błędów zaokrągleń wyników pośrednich, które mogą mieć wpływ na niedokładność wyników końcowych. Stąd algorytm „delty” określamy jako **niestabilny algorytm rozwiązujący równanie kwadratowe**. Poniżej przedstawiono ten algorytm w postaci schematu blokowego (rysunek 1.2), listy kroków, pseudokodu oraz programów w językach Python i C++.

Lista kroków:

- Krok 1.** Jeśli $a = 0$, to wypisz komunikat „to nie jest równanie kwadratowe” i zakończ algorytm.
- Krok 2.** (W tym przypadku $a \neq 0$). Przypisz $\Delta = b^2 - 4ac$.
- Krok 3.** Jeśli $\Delta < 0$, to wypisz komunikat „równanie nie ma pierwiastków” i zakończ algorytm.
- Krok 4.** (W tym przypadku $\Delta \geq 0$). Jeśli $\Delta = 0$, to oblicz pierwiastek $x = \frac{-b}{2a}$, wypisz wyznaczony pierwiastek x i zakończ algorytm.
- Krok 5.** (W tym przypadku $\Delta > 0$). Oblicz pierwiastki $x_1 = \frac{-b - \sqrt{\Delta}}{2a}$ i $x_2 = \frac{-b + \sqrt{\Delta}}{2a}$, wypisz wartości wyznaczonych pierwiastków x_1 i x_2 . Zakończ algorytm.

Pseudokod:

```
jeżeli a = 0, to
    wypisz komunikat "to nie jest równanie kwadratowe"
w przeciwnym razie
    przypisz  $\Delta = b^2 - 4ac$ 
    jeżeli  $\Delta < 0$ , to
        wypisz komunikat "równanie nie ma pierwiastków"
    w przeciwnym razie
        jeżeli  $\Delta = 0$ , to
            oblicz pierwiastek  $x = \frac{-b}{2a}$ 
            wypisz x
        w przeciwnym razie
            oblicz pierwiastki  $x_1 = \frac{-b - \sqrt{\Delta}}{2a}$  i  $x_2 = \frac{-b + \sqrt{\Delta}}{2a}$ 
            wypisz x1 i x2
```



Rysunek 1.2. Schemat blokowy algorytmu „delta” rozwiązującego równanie kwadratowe

Program w języku Python:

```

from math import *

def rownanie_kwadratowe(a, b, c):
    if a == 0:
        return "to nie jest równanie kwadratowe"
    else:
        delta = b * b - 4 * a * c
        if delta < 0:
            return "równanie nie ma pierwiastków"
        elif delta == 0:
            x1 = -b / (2 * a)
            return x1
  
```



```

else:
    x1 = (-b - sqrt(delta)) / (2 * a)
    x2 = (-b + sqrt(delta)) / (2 * a)
    return x1, x2

```

```
print(rownanie_kwadratowe(2, 1, 3))
```

Wynik:

równanie nie ma pierwiastków

Program w języku C++:

```

#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    double a, b, c, delta, x1, x2;
    cout << "podaj współczynniki równania a, b, c:" << endl;
    cin >> a >> b >> c;
    if (a == 0) cout << "to nie jest równanie kwadratowe" << endl;
    else
    {
        delta = b * b - 4 * a * c;
        if (delta < 0) cout << "równanie nie ma pierwiastków" << endl;
        else if (delta == 0)
        {
            x1 = -b / (2 * a);
            cout << "x=" << x1 << endl;
        }
        else
        {
            x1 = (-b - sqrt(delta)) / (2 * a);
            x2 = (-b + sqrt(delta)) / (2 * a);
            cout << "x1 = " << x1 << "\tx2 = " << x2 << endl;
        }
    }
    return 0;
}

```

Ćwiczenie 1.1.

Przeanalizuj przedstawiony powyżej algorytm, który został zapisany za pomocą różnych metod. Porównaj czytelność tych technik. Przetestuj przedstawione rozwiązania dla przykładowych danych.

1.2.2. Wzory Viète'a

Obliczając pierwiastki równania kwadratowego, możemy również korzystać ze **wzorów Viète'a**. Wzory te zostały sformułowane już w XVI wieku przez francuskiego matematyka François Viète'a.

Definicja

Jeżeli liczby x_1 i x_2 są pierwiastkami równania kwadratowego $ax^2 + bx + c = 0$, to zachodzą równości:

$$x_1 + x_2 = -\frac{b}{a},$$

$$x_1 \cdot x_2 = \frac{c}{a}.$$

Należy zwrócić uwagę na to, że podane powyżej wzory Viète'a można stosować tylko wtedy, gdy równanie kwadratowe ma rozwiązania.

Wzory te wykorzystywane są na przykład do określania znaków pierwiastków trójmianu bez konieczności rozwiązywania tego równania. Zastosować je można również do skorygowania błędów zaokrągleń występujących w algorytmie „deltę”.

1.2.3. Stabilny algorytm rozwiązujący równanie kwadratowe

Aby uniknąć błędów zaokrągleń w algorytmie „deltę”, do obliczania pierwiastków równania kwadratowego należy zastosować obie podane wcześniej metody. Wybór wzorów uzależniony jest od znaku współczynnika b . Wynika to stąd, że błąd zaokrąglenia spowodowany jest odejmowaniem bliskich co do wartości liczb. Zachodzi to w liczniku wzorów na pierwiastki równania w algorytmie „deltę”. Jeśli $b > 0$, to odejmowanie pojawia się we

wzorze $x_2 = \frac{-b + \sqrt{\Delta}}{2a}$. W sytuacji gdy $b \leq 0$, odejmowanie wykonywane jest we wzorze

$x_1 = \frac{-b - \sqrt{\Delta}}{2a}$. Aby wyeliminować to działanie, w miejsce wzoru generującego błędy

wstawiamy odpowiednio przekształcony wzór Viète'a $x_1 \cdot x_2 = \frac{c}{a}$, wyznaczający iloczyn pierwiastków równania.

Poprawiając algorytm „delty”, wprowadzamy zmiany, gdy spełniony jest warunek $\Delta > 0$. Wówczas, jeśli $b > 0$, wykonujemy działania:

$$x_1 = \frac{-b - \sqrt{\Delta}}{2a} \text{ i } x_2 = \frac{\frac{c}{a}}{x_1},$$

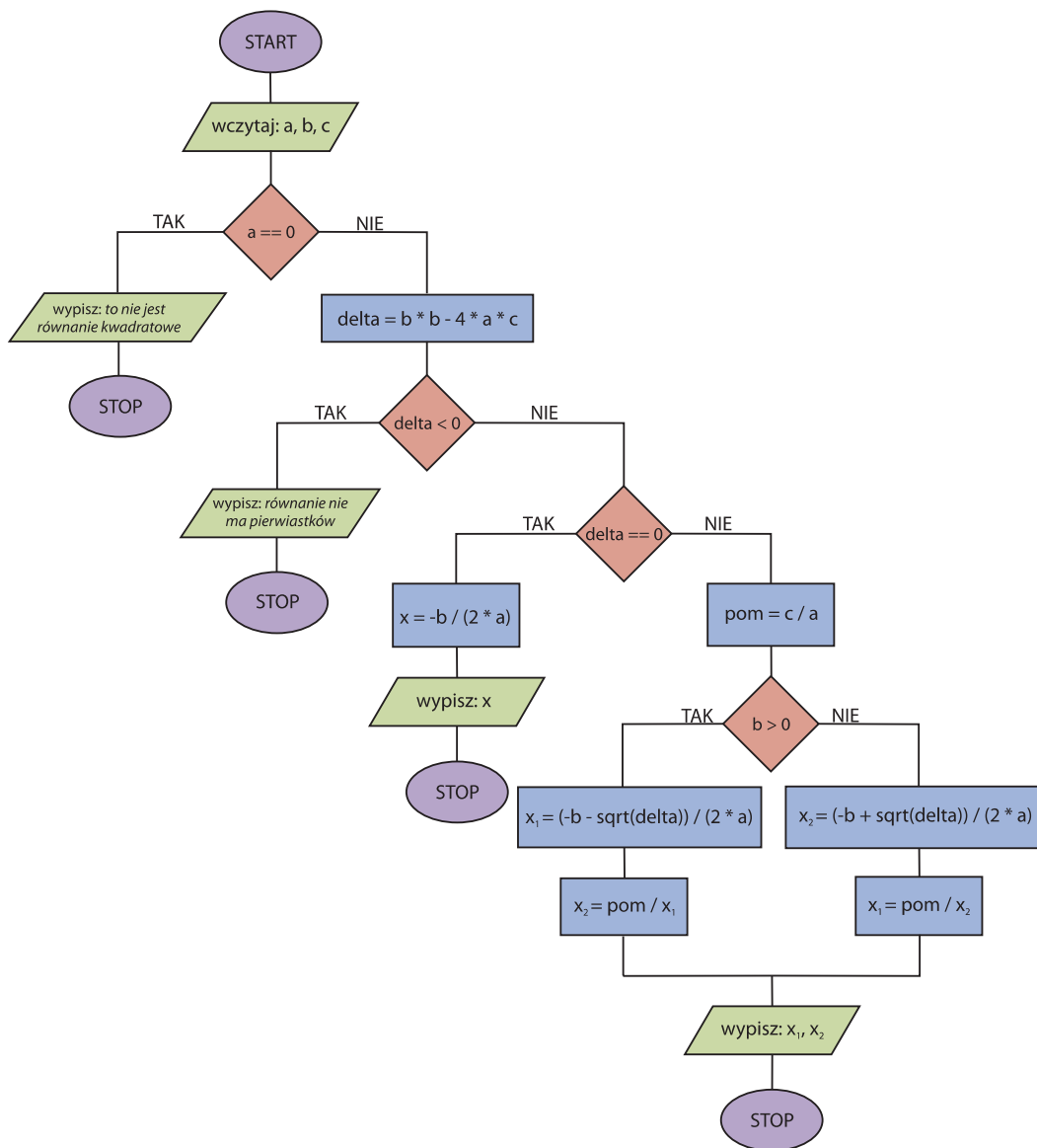
natomiast jeśli $b \leq 0$, realizujemy operacje:

$$x_2 = \frac{-b + \sqrt{\Delta}}{2a} \text{ i } x_1 = \frac{\frac{c}{a}}{x_2}.$$

Realizacja omawianego algorytmu została przedstawiona w postaci listy kroków i schematu blokowego (rysunek 1.3).

Lista kroków:

- Krok 1.** Jeśli $a = 0$, wypisz komunikat „to nie jest równanie kwadratowe” i zakończ algorytm.
- Krok 2.** (W tym przypadku $a \neq 0$). Przypisz $\Delta = b^2 - 4ac$.
- Krok 3.** Jeśli $\Delta < 0$, wypisz komunikat „równanie nie ma pierwiastków” i zakończ algorytm.
- Krok 4.** (W tym przypadku $\Delta \geq 0$). Jeśli $\Delta = 0$, oblicz pierwiastek $x = \frac{-b}{2a}$, wypisz wyznaczony pierwiastek x i zakończ algorytm.
- Krok 5.** (W tym przypadku $\Delta > 0$). Przypisz $pom = \frac{c}{a}$.
- Krok 6.** Jeśli $b > 0$, oblicz pierwiastki $x_1 = \frac{-b - \sqrt{\Delta}}{2a}$, $x_2 = \frac{pom}{x_1}$, wypisz wartości wyznaczonych pierwiastków x_1 i x_2 oraz zakończ algorytm.
- Krok 7.** (W tym przypadku $b \leq 0$). Oblicz pierwiastki $x_2 = \frac{-b + \sqrt{\Delta}}{2a}$ i $x_1 = \frac{pom}{x_2}$, wypisz wartości wyznaczonych pierwiastków x_1 i x_2 . Zakończ algorytm.



Rysunek 1.3. Schemat blokowy stabilnego algorytmu rozwiązującego równanie kwadratowe

Ćwiczenie 1.2.

Zapisz algorytm rozwiązujący równanie kwadratowe metodą stabilną (na podstawie listy kroków i schematu blokowego przedstawionego na rysunku 1.3) w postaci pseudokodu i programu. Przetestuj algorytm dla przykładowych danych.

Zadanie 1.1.

Przeanalizuj podany poniżej kod programu:

```
x, y, z = 0, 1, 2
if x: x += 1
elif x + y: y *= x + 3
elif y: x -= y * 3
x += y
print("x = ", x, "\ty = ", y)
```

Podaj wartość zmiennych x i y po wykonaniu podanego algorytmu. Utwórz schemat blokowy, listę kroków i pseudokod dla tego algorytmu.

Zadanie 1.2.

Podaj specyfikację zadania i skonstruuj algorytm w postaci schematu blokowego i programu wyznaczający wartość podanej funkcji:

$$f(x) = \begin{cases} 2x & \text{dla } x < 1 \\ -10 & \text{dla } x = 1 \\ (x-1)^4 & \text{dla } x = 3 \\ \sqrt{x-4} & \text{dla } x = 6 \\ 0 & \text{dla innych } x \end{cases}$$

Zmienną rzeczywistą x należy wprowadzić z klawiatury.

Zadanie 1.3.

Podaj specyfikację zadania i skonstruuj algorytm w postaci listy kroków i pseudokodu wyznaczający wartość podanej funkcji:

$$f(x) = \begin{cases} x^3 + 1 & \text{dla } x < 7 \\ \cos(x-1) & \text{dla } x = 7 \\ \sqrt{3x} & \text{dla } x = 9 \\ -8x & \text{dla innych } x \end{cases}$$

Zmienną rzeczywistą x należy wprowadzić z klawiatury.

Zadanie 1.4.

Poniżej podany jest pewien algorytm zapisany w pseudokodzie. Oblicza on wartość sumy dla danej liczby naturalnej n większej od 0. Użyty symbol \leftarrow często stosowany jest w pseudokodzie i oznacza operację przypisania wartości zmiennej.

```
suma  $\leftarrow$  0
liczba_1  $\leftarrow$  1
dla i  $\leftarrow$  1...n wykonuj
    liczba_1  $\leftarrow$  liczba_1 + n
    liczba_2  $\leftarrow$  1
    dla j  $\leftarrow$  1...i wykonuj
        liczba_2  $\leftarrow$  liczba_2 * j
    suma  $\leftarrow$  suma + liczba_1 + liczba_2
```

Wykonaj poniższe polecenia:

- Podaj specyfikację algorytmu.
- Podaj, jakie wartości przyjmą zmienne *liczba_1*, *liczba_2* i *suma* w wyniku działania powyższego algorytmu dla $n = 3$. Obliczenia wykonaj bez użycia komputera.
- Czy potrafisz zapisać wartość sumy obliczanej przez algorytm w postaci wzoru?
- Zapisz powyższy algorytm w postaci schematu blokowego, listy kroków i programu. Przetestuj działanie tego algorytmu dla przykładowych danych.

..... Temat 2. Wprowadzenie do języka C++ — podstawowe konstrukcje i analiza przykładów

Znasz już podstawy języka Python, jednak nie jest to jedyny język programowania dopuszczony do matury. W tym podręczniku na poziomie rozszerzonym algorytmy zapisywane są w dwóch językach: Python i C++. Temat ten wprowadza Cię w świat nowego języka programowania C++. Kiedy poznasz podstawy języka C++, będziesz mógł wybrać jeden z tych dwóch języków do realizacji zagadnień algorytmicznych. Możesz też poznawać obydwa języki, co na pewno przyda Ci się w przyszłości.

Temat ten omawia podstawy języka programowania C++ niezbędne do realizacji zagadnień algorytmicznych zawartych w tym podręczniku. Rozwój projektowania i programowania w kierunku obiektowym spowodował zmiany w zakresie języka C. Rozszerzono go o elementy umożliwiające programowanie zorientowane obiektowo. C++ stanowi więc nadzbiór języka C. Różnica pomiędzy tymi językami jest znacząca, jednak wszystkie poprawne programy napisane w języku C są również poprawne w C++.

W podręczniku będziemy korzystać z **międzynarodowego standardu języka C++**, stworzonego przez komitet ASC (*Accredited Standards Committee*), który działa w ramach ANSI (*American National Standards Institute*). Standard ten określanym jest najczęściej mianem **ANSI/ISO**.

Standard ANSI C++ stara się zapewnić **przenośność** tego języka. Oznacza to, że kod programu powinien być poprawny dla różnego typu komputerów pracujących w wielu systemach. Ponadto standard ten powinien być czytelny dla różnych kompilatorów.

2.1. Struktura programu

Program w języku C++ zbudowany jest z dwóch części:

- części deklaracyjnej,
- części operacyjnej.

Część deklaracyjna obejmuje wszystkie deklarowane elementy programu, które mogą zostać wykorzystane w części operacyjnej. Ta część służy do definiowania bądź deklarowania różnych elementów, jednak nie można ich w tym miejscu uruchomić. Do tego wykorzystuje się część operacyjną.

Definicja

Część operacyjna to program główny, który w tym języku jest funkcją o nazwie `main()`. Po uruchomieniu programu wykonywane są wszystkie polecenia zawarte w funkcji `main()`. W tej części programu korzystamy ze zdefiniowanych lub zadeklarowanych przez siebie elementów znajdujących się w części deklaracyjnej.

Definicja

Na początkowym etapie nauki języka C++ będziesz korzystać z uproszczonej budowy programu. W następnych podrozdziałach poznasz bardziej zaawansowane możliwości konstruowania programu.

Przykład 2.1.

Przyjrzyjmy się konstrukcji pierwszego programu, który został przedstawiony w tabeli 2.1. Opisy umieszczone z prawej strony wyjaśniają znaczenie poszczególnych elementów implementacji. W języku C++ **rozdzielane są małe i wielkie litery**, kod programu należy więc zapisywać dokładnie tak, jak podano w podręczniku.

Tabela 2.1. Pierwszy program w języku C++

Kod programu	Opis
CZĘŚĆ DEKLARACYJNA	
<code>#include <iostream></code>	Dyrektywa preprocesora — preprocesor to program dokonujący wstępnego przeglądu kodu źródłowego programu Deklaracja bibliotek (czyli plików nagłówkowych), z których korzystamy w programie Biblioteka <code>iostream</code> zawiera definicje standardowego strumienia wejścia <code>cin</code> i standardowego strumienia wyjścia <code>cout</code>
<code>using namespace std;</code>	Deklaracja korzystania ze standardowej przestrzeni nazw <code>std</code> (czyli biblioteki standardowej)

Kod programu	Opis
CZĘŚĆ OPERACYJNA	
<code>int main()</code>	Nagłówek funkcji głównej
<code>{</code>	Nawias otwierający (początek funkcji)
<code>int a, b, c;</code>	Deklaracja zmiennych lokalnych <code>a, b, c</code>
<code>a = 2;</code>	Przypisanie zmiennej <code>a</code> wartości 2
<code>cout << "podaj wartość zmiennej b:";</code>	Komunikat wypisany na ekranie „podaj wartość zmiennej b:”
<code>cin >> b;</code>	Wczytanie z klawiatury wartości zmiennej <code>b</code>
<code>c = a + b;</code>	Przypisanie zmiennej <code>c</code> sumy zmiennych <code>a</code> i <code>b</code>
<code>cout << a << " + " << b << " = " << c << endl;</code>	Wypisanie wyniku na ekranie
<code>return 0;</code>	Przypisanie funkcji <code>main()</code> wartości 0
<code>}</code>	Nawias zamykający (koniec funkcji)

Wskazówka

Nagłówek funkcji głównej `int main()` w pełnej postaci zapisywany jest następująco: `int main(void)`, gdzie:

- `int` oznacza, że typ wartości zwracanej z funkcji `main()` jest całkowity, funkcji tej przypisywana jest więc wartość typu `int`;
- `main` to nazwa funkcji;
- `void` oznacza „pusty”, a w tym przypadku brak parametrów, które wpisuje się w nawiasie; pusty nawias jest równoważny `void`.

W razie pominięcia deklaracji przestrzeni nazw standardowych `std` operacje wejścia i wyjścia dostępne są dopiero po podaniu nazwy klasy `std`, czyli `std::cout`, `std::cin`. Program z takimi zmianami przedstawiono poniżej:

```
#include <iostream>
int main()
{
    int a, b, c;
    a = 2;
    std::cout << "podaj wartość zmiennej b:";
    std::cin >> b;
    c = a + b;
    std::cout << a << " + " << b << " = " << c << std::endl;
    return 0;
}
```


2.2. Operacje wejścia-wyjścia

Strumieniowe operacje wejścia-wyjścia dostępne są po dołączeniu biblioteki `iostream`. Korzystać możemy z dwóch standardowych strumieni:

- `cout` — standardowy strumień wyjściowy, który zwykle skojarzony jest z monitorem; stosujemy tutaj operator kierunkowy wstawiania znaków do strumienia wyjściowego: `<<`;
- `cin` — standardowy strumień wejściowy, który zwykle skojarzony jest z klawiaturą; wykorzystujemy w tym przypadku operator kierunkowy pobierania danych ze strumienia do podanej zmiennej: `>>`.

Wypisując komunikat poleceniem `cout`, w łańcuchu oznaczonym cudzysłowem można stosować **znaki specjalne**. W tabeli 2.2 podano zestawienie podstawowych znaków specjalnych.

Tabela 2.2. Zestawienie znaków specjalnych

Znak specjalny	Opis
<code>\n</code>	<i>new line</i> (przejdźcie do nowej linii)
<code>\t</code>	<i>tab</i> (tabulator poziomy)
<code>\v</code>	<i>vertical tab</i> (tabulator pionowy)
<code>\b</code>	<i>backspace</i> (przesunięcie kursora o jeden znak w lewo)
<code>\r</code>	<i>carriage return</i> (przesunięcie kursora na początek bieżącej wiersza)
<code>\'</code>	' (wypisanie znaku zastrzeżonego — apostrof)
<code>\"</code>	" (wypisanie znaku zastrzeżonego — cudzysłów)
<code>\?</code>	? (wypisanie znaku zastrzeżonego — znak zapytania)
<code>\\</code>	\ (wypisanie znaku zastrzeżonego — <i>backslash</i>)

Przykład 2.2.

Przyjrzyjmy się przykładom zastosowania strumieniowych operacji wejścia i wyjścia przedstawionym w tabeli 2.3. Z prawej strony umieszczono komentarze wyjaśniające wykonywane operacje.

Tabela 2.3. Przykłady zastosowania operacji wejścia i wyjścia

Przykładowe operacje wejścia i wyjścia	Efekt działania polecenia	Opis
<pre>cout << "tekst" << " " << "tekst";</pre>	<pre>tekst tekst</pre>	Nie ma potrzeby wydzielenia poszczególnych słów wypisywanego tekstu, taki sam efekt uzyskamy poleceniem: <pre>cout << "tekst tekst";</pre>
<pre>cout << "tekst\ttekst\ntekst";</pre>	<pre>tekst tekst tekst</pre>	Wewnątrz wyświetlanego łańcucha można stosować znaki specjalne
<pre>int a = 5, b = 3; cout << "a=" << a << "\n" << a + b;</pre>	<pre>a=5 8</pre>	W strumieniu wyjściowym można umieszczać zarówno komunikaty, jak i zmienne, które należy od siebie oddzielać operatorem << Znak specjalny \n stosowany pojedynczo można zapisywać "\n" lub '\n'
<pre>int a; cin >> a;</pre>	Oczekiwanie na wpisanie z klawiatury wartości zmiennej <i>a</i> i potwierdzenie tego klawiszem <i>Enter</i>	Wczytanie z klawiatury wartości zmiennej <i>a</i>
<pre>int a, b, c; cin >> a >> b >> c;</pre>	Oczekiwanie na wpisanie z klawiatury wartości zmiennych <i>a</i> , <i>b</i> i <i>c</i> oddzielonych przez spacje lub <i>Enter</i> oraz potwierdzenie tego klawiszem <i>Enter</i>	Przy wczytywaniu z klawiatury wartości wielu zmiennych wygodnie jest wykonać to za pomocą jednej operacji <i>cin</i> Wczytywane zmienne mogą być różnego typu

Do formatowania strumieniowych operacji wejścia i wyjścia najczęściej wykorzystuje się **manipulatory**. Najważniejsze manipulatory przedstawione zostały w tabeli 2.4.

Tabela 2.4. Zestawienie manipulatorów strumieniowych

Manipulator	Opis
<code>dec</code>	Włącza konwersję dziesiętną
<code>hex</code>	Włącza konwersję szesnastkową
<code>oct</code>	Włącza konwersję ósemkową
<code>endl</code>	Umieszcza w strumieniu znak końca wiersza i opróżnia strumień
<code>ends</code>	Umieszcza znak <code>'\0'</code> na końcu łańcucha
<code>setfill(char c)</code>	Ustala znak wypełniania wolnych miejsc pola na <code>c</code> (dostępny po dołączeniu biblioteki <code>iomanip</code>)
<code>setprecision(int n)</code>	Określa precyzję liczb rzeczywistych równą <code>n</code> (dostępny po dołączeniu biblioteki <code>iomanip</code>)
<code>setw(int n)</code>	Wyznacza szerokość pola równą <code>n</code> znaków (dostępny po dołączeniu biblioteki <code>iomanip</code>)

Przykład 2.3.

Przyjrzyj się przykładom zastosowania manipulatorów strumieniowych w przedstawionym poniżej programie:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int k = 241;
    cout << "k = " << k << endl;
    cout << "konwersja systemów liczbowych:" << endl;
    cout << "szesnastkowy = " << hex << k << endl;
    cout << "dziesiętny = " << dec << k << endl;
    cout << "szesnastkowy = " << setfill('0') << setw(6) << hex << k << endl;
    cout << "ósemkowy = " << setfill('*') << setw(10) << oct << k << endl;
    return 0;
}
```

Wyniki:

<code>k = 241</code>	<code>dziesiętny = 241</code>
<code>konwersja systemów liczbowych:</code>	<code>szesnastkowy = 0000f1</code>
<code>szesnastkowy = f1</code>	<code>ósemkowy = *****361</code>

Powyższy program wypisuje wartość zmiennej `k` w różnych systemach liczbowych z podaną szerokością pola i znakiem wypełniania.

Przykład 2.4.

Przykład pokazuje zastosowanie manipulatorów dla zmiennych typu rzeczywistego:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    double x = 22.11111111, y = 3.555555555555, w;
    cout << "podaj liczbę rzeczywistą: ";
    cin >> w;
    cout << "x = " << setw(10) << x << endl;
    cout << "y = " << setfill('0') << setw(12) << y << endl;
    cout << "y = " << setprecision(8) << setfill('0') << setw(12) << y << endl;
    cout << "w = " << setfill('*') << setw(15) << w << endl;
    cout << "x + y + w = " << setprecision(9) << x + y + w << endl;
    cout << "x * y = " << setprecision(8) << x * y << endl;
    return 0;
}
```

Przykładowe wyniki:

podaj liczbę rzeczywistą: 2.5

x = 22.1111

y = 000003.55556

y = 0003.5555556

*w = *****2.5*

x + y + w = 28.1666667

*x * y = 78.617284*

Ćwiczenie 2.1.

Na podstawie kodu programu podanego w przykładzie 2.4 i wyników jego uruchomienia określ, w jaki sposób ustawiana jest precyzja dla liczb rzeczywistych za pomocą manipulatora `setprecision()`.

Podaj, z jaką precyzją wypisywane są wartości zmiennych rzeczywistych bez zastosowania manipulatora precyzji.

2.3. Zmienne, stałe, wskaźniki i referencje

2.3.1. Zmienne

Zmienną nazywamy identyfikator, który może mieć przypisaną określoną wartość (można ją zmieniać podczas realizacji programu). Aby korzystać ze zmiennej, należy ją wcześniej zadeklarować i jednocześnie określić jej typ.

Definicja

Deklaracja zmiennej ma postać:

```
typ identyfikator;
```

Nazwy zmiennych powinny być czytelne i ułatwiać piszącemu program dokonywanie w nim poprawek.

Przykład 2.5.

Przyjrzyjmy się przykładom **deklaracji zmiennych**:

```
int A;
float b, c, d;
char nazwa;
unsigned long int odleglosc;
string kolor_samochodu;
```

Podczas deklaracji zmiennych możliwe jest nadawanie im wartości początkowych przez **inicjalizację zmiennych**:

```
double liczba = 0.00125;
short int suma = 0, iloczyn = 1;
char znak = 'a';
```

2.3.2. Stałe

Stałą nazywamy identyfikator o przypisanej określonej wartości, której podczas realizacji programu nie można zmieniać. Aby korzystać ze stałej, należy ją wcześniej zadeklarować i jednocześnie określić jej typ.

Definicja

Deklaracja stałej ma postać:

```
const typ identyfikator = wartość;
```

Nazwy stałych, podobnie jak zmiennych, powinny być czytelne i ułatwiać piszącemu program dokonywanie w nim poprawek.

Przykład 2.6.

Przjrzyjmy się przykładom deklaracji stałych:

```
const int B = 10;
const double a = 2.34, b = 3.125;
const char znak = 'W';
```

2.3.3. Wskaźniki

Definicja

Wskaźniki to adresy danych, struktur danych, obiektów klas lub funkcji. Wskaźnik zawiera informację o adresie oraz typie wskazywanego elementu w pamięci komputera.

Za pomocą operatora `*` możemy deklarować zmienne wskaźnikowe:

```
typ *identyfikator;
```

Wówczas `*identyfikator` to wartość zmiennej wskaźnikowej (czyli dynamicznej), a `identyfikator` to jej adres.

W języku C++ do dynamicznego przydzielania i zwalniania pamięci stosuje się operatory `new` i `delete`. Operator `new` służy do przydzielania pamięci i ma składnię:

```
identyfikator = new typ_obiektu;
```

Wartością wyrażenia jest tutaj wskaźnik do utworzonego obiektu.

Operator `delete` wykorzystywany jest do zwalniania pamięci i ma następującą składnię:

```
delete identyfikator;
```

Operator `delete` usuwa obiekt, który jest wskazywany przez `identyfikator`.

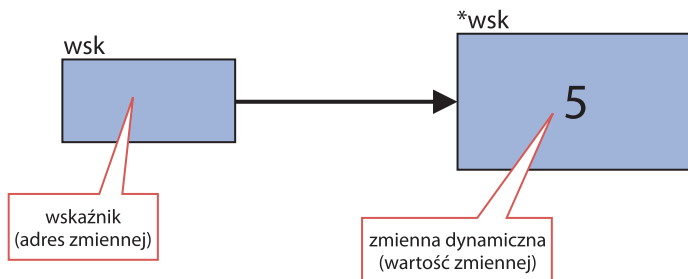
Przykład 2.7.

Poniżej przedstawiono przykład dynamicznej alokacji pamięci dla zmiennej typu `int`:

```
int *wsk;
wsk = new int;
// operacje wykonywane na zmiennej *wsk, na przykład: *wsk=5;
delete wsk;
```

Na rysunku 2.1 pokazano efekt wykonania operacji `new`. Zmienna `wsk` zawiera adres wskazujący na zmienną dynamiczną `*wsk`, której następnie przypisywana jest wartość 5.

Do wyznaczenia wskaźnika zmiennej zadeklarowanej statycznie używany jest operator adresu `&`. Wykorzystuje się go do nadawania wartości początkowych zmiennym wskaźnikowym, a następnie dokonywania na nich zmian.



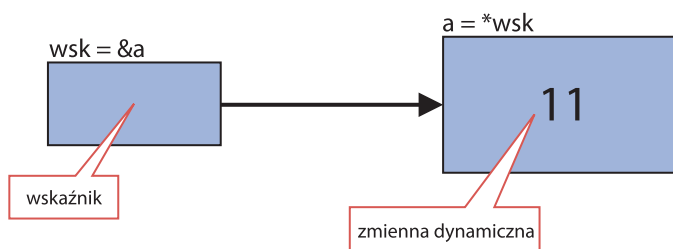
Rysunek 2.1. Powiązanie wskaźnika ze zmienną dynamiczną

Przykład 2.8.

Przeanalizujmy przykład zastosowania operatora adresu:

```
int a;
int *wsk;
wsk = &a;
*wsk = 11;
```

Na rysunku 2.2 przedstawiono efekt wykonania tych operacji.



Rysunek 2.2. Dwie nazwy dla jednej zmiennej dynamicznej

Wskaźnik `wsk` zawiera adres zmiennej dynamicznej `a`. Powoduje to, że zmienne `*wsk` i `a` są równoważne. Zmiana wartości zmiennej dynamicznej `*wsk` powoduje zmiany zmiennej `a`, i odwrotnie. Przyczyna jest prosta — obydwie zmienne mają ten sam adres, a więc przeznaczone dla nich miejsce w pamięci komputera ma dwie nazwy.

2.3.4. Referencje

Referencje, oznaczane znakiem `&`, zastępują zmienne, struktury danych lub obiekty klas. Operacje realizowane na referencji wykonywane są na reprezentowanej przez nią zmiennej, strukturze czy obiekcie.

Definicja

Referencja zawsze musi mieć wartość początkową, która wiąże ją z reprezentowaną zmienną, strukturą lub obiektem:

```
typ &indyfikator = zmienna;
```

Przykład 2.9.

Przyjrzyjmy się przykładowi zastosowania referencji:

```
int a, b;
int &ref = a;
ref = 10;
b = ref;
```

Po wykonaniu powyższych instrukcji zmienna `a` reprezentowana przez referencję `ref` ma wartość 10. Zmienna `b`, będąca zmienną, która nie jest powiązana z referencją `ref`, również ma wartość 10. Operacje wykonywane na referencji `ref` i zmiennej `a` są równoważne.

2.4. Wyrażenia arytmetyczne, relacje i operatory logiczne

2.4.1. Operator przypisania (podstawienia)

Operator przypisania (tabela 2.5) realizuje nadanie wartości zmiennej. Z lewej strony tego operatora musi być określona zmienna, z prawej — wyrażenie, którego wartość przypisujemy tej zmiennej.

Tabela 2.5. Operator przypisania

Operator	Opis
=	Przypisanie wartości zmiennej

Przykład 2.10.

Przeanalizujmy przykłady zastosowania operatora przypisania:

```
a = 4;
b = a + 2;
a = (b = 8) - 3;    Zagnieżdżenie instrukcji przypisania, równoważne
                   instrukcjom b = 8; a = b - 3.
a = b = c = 0;     Wielokrotna instrukcja przypisania, równoważna
                   instrukcjom c = 0; b = c; a = b.
```

2.4.2. Operatory arytmetyczne

Operatory arytmetyczne (tabela 2.6) to operatory dwuargumentowe realizujące operacje arytmetyczne.

Operator dzielenia ma dwa zastosowania, które zależą od typu argumentów. Jeśli obydwa argumenty są całkowite, wynik dzielenia również jest liczbą całkowitą. W tym przypadku uzyskujemy część całkowitą

Tabela 2.6. Zestawienie operatorów arytmetycznych

Operator	Opis
*	Mnożenie
/	Dzielenie
+	Dodawanie
-	Odejmowanie
%	Reszta z dzielenia

wyniku z dzielenia tych liczb. W sytuacji gdy chociaż jeden z argumentów jest rzeczywisty, wynikiem jest również liczba rzeczywista.

Przykład 2.11.

Przeanalizuj przedstawione w tabeli 2.7 przykłady dzielenia liczb całkowitych i rzeczywistych zapisane w języku C++.

Tabela 2.7. Przykłady wyrażeń zapisanych w języku C++ wykorzystujących operator dzielenia

Wyrażenie	Wartość wyrażenia
<code>9.0 / 2.0</code>	4.5
<code>9 / 2.0</code>	4.5
<code>9.0 / 2</code>	4.5
<code>9 / 2</code>	4

Tylko w jednym przypadku uzyskujemy wynik będący liczbą całkowitą. Aby wynik był liczbą całkowitą, obydwa argumenty muszą być wartościami całkowitymi.

Jeśli argumenty wykonywanej operacji dzielenia są zmiennymi całkowitymi, a potrzebny jest wynik rzeczywisty, pojawia się problem. Aby go rozwiązać, należy wymusić, by wynik dzielenia był rzeczywisty. Stosujemy wówczas tak zwane **rzutowanie**, pokazane na poniższym przykładzie:

```
int a = 5, b = 2;
cout << a / b << "\t" << (double)a / b << endl;
```

Wyniki:

```
2 2.5
```

2.4.3. Operatory zmniejszania i zwiększania

Wyróżniamy dwa rodzaje operatorów zmniejszania i zwiększania (tabela 2.8):

- prefiksowe — `++a`, `--a`,
- postfiksowe — `a++`, `a--`.

Tabela 2.8. Zestawienie operatorów zmniejszania i zwiększania

Operator	Opis
<code>++</code>	Zwiększanie argumentu o 1 (inkrementacja)
<code>--</code>	Zmniejszanie argumentu o 1 (dekrementacja)

Jeśli operator występuje wewnątrz wyrażenia, to przy zapisie prefiksowym zwiększanie lub zmniejszanie argumentu jest wykonywane przed obliczeniem wartości wyrażenia, natomiast przy zapisie postfiksowym — po obliczeniu wartości wyrażenia. W tabeli 2.9 podano przykłady zastosowań tych operatorów.

Tabela 2.9. Przykłady zastosowań operatorów zmniejszania i zwiększania w zapisie postfiksowym i prefiksowym

Przykład zastosowania	Wartości zmiennych po wykonaniu operacji
<pre>int a = 3, b = 4, c; ++a; b--; c = a + b;</pre>	<pre>a = 4 b = 3 c = 7</pre>
<pre>int a = 3, b = 4, c; c = ++a + b;</pre>	<pre>a = 4 b = 4 c = 8</pre>
<pre>int a = 3, b = 4, c; c = a + b--;</pre>	<pre>a = 3 b = 3 c = 7</pre>
<pre>int a = 3, b = 4, c; c = ++a + b--;</pre>	<pre>a = 4 b = 3 c = 8</pre>

2.4.4. Operatory relacyjne

W tabeli 2.10 podano operatory relacyjne, z których można korzystać w języku C++.

Tabela 2.10. Zestawienie operatorów relacyjnych

Operator	Opis
<	Mniejsze
<=	Mniejsze lub równe
==	Równe
!=	Różne (nieprawda, że równe)
>=	Większe lub równe
>	Większe

2.4.5. Operatory logiczne

Wartość wyrażenia logicznego, w którym korzystamy z operatorów logicznych (tabela 2.11), jest zawsze typu całkowitego.

Tabela 2.11. Zestawienie operatorów logicznych

Operator	Opis
!	Negacja
&&	Koniunkcja (iloczyn logiczny)
	Alternatywa (suma logiczna)

W języku C++ dowolna niezerowa wartość interpretowana jest jako prawda (czyli `true`), natomiast wartość 0 jako fałsz (czyli `false`).

Przykład 2.12.

Przeanalizujmy następujący przykład:

```
int m = 8, n = 0, wynik;  
wynik = m || n;  
cout << wynik << "\t";  
wynik = m && n;  
cout << wynik << "\t";  
wynik = !m;  
cout << wynik << "\t";  
wynik = !n;  
cout << wynik << endl;
```

Wyniki:

```
1    0    0    1
```

Wynika to stąd, że `m` równe 8 interpretowane jest jako prawda, a `n` równe 0 jako fałsz. Jeśli wynikiem wykonanej operacji jest prawda, to wypisywana jest wartość 1, natomiast jeśli fałsz — wartość 0.

Przykład 2.13.

Następny przykład pokazuje zastosowanie operatora logicznego `&&` w konstruowaniu warunku w instrukcji warunkowej `if`:

```
int m = -3, n = -7;  
if (m < 0 && n < 0) cout << "liczby są ujemne" << endl;  
else cout << "co najmniej jedna liczba nie jest ujemna" << endl;
```

Wynik:

```
liczby są ujemne
```

Liczby `m` i `n` są ujemne, spełnione są więc obydwa warunki, dlatego pojawia się ten komunikat. Gdyby chociaż jedna z tych liczb była nieujemna, wyrażenie byłoby fałszywe i wypisany zostałby komunikat „co najmniej jedna liczba nie jest ujemna”.

2.4.6. Złożone operatory przypisania

Złożone operatory przypisania stosuje się do zapisywania wyrażeń $X = X \cdot Y$ w postaci `X *= Y`, gdzie `·` to operator dwuargumentowy. Do najważniejszych z nich należą:

`+=` `-=` `*=` `/=` `%=`

Przykład 2.14.

W tabeli 2.12 podano przykłady zastosowania złożonych operatorów przypisania w konstruowaniu wyrażeń.

Tabela 2.12. Przykłady zastosowania złożonych operatorów przypisania

Wyrażenie	Zapis wyrażenia bez złożonego operatora przypisania
<code>a *= 4 * b - 1;</code>	<code>a = a * (4 * b - 1);</code>
<code>m %= 3 - n / 2;</code>	<code>m = m % (3 - n / 2);</code>
<code>b -= 3 + d * 4;</code>	<code>b = b - (3 + d * 4);</code>
<code>k += (a *= 2) - (b += 3);</code>	<code>k = k + ((a = a * 2) - (b = b + 3));</code>

Zwróć uwagę na kolejność wykonywanych działań.

2.4.7. Operator warunkowy

Operator warunkowy jest jedynym w języku C++ operatorem trzyargumentowym. Jego składnia jest następująca:

$$w_1 \text{ ? } w_2 \text{ : } w_3$$

gdzie: w_1, w_2, w_3 — wyrażenia.

Wykonanie powyższej operacji przebiega następująco:

1. Obliczenie wartości wyrażenia w_1 .
2. Jeśli wartość wyrażenia w_1 jest różna od 0 (czyli wyrażenie jest prawdziwe), rezultatem operacji jest wartość wyrażenia w_2 . Wyrażenie w_3 nie jest wówczas obliczane.
3. Jeśli wartość wyrażenia w_1 jest równa 0 (czyli wyrażenie jest fałszywe), rezultatem operacji jest wartość wyrażenia w_3 . Wyrażenie w_2 nie jest w tym przypadku obliczane.

Realizacja tego działania przypomina instrukcję warunkową `if`, która została omówiona w punkcie 2.8.2, „Instrukcje warunkowe”.

Przykład 2.15.

Zastosujmy operator warunkowy do wyznaczenia najmniejszej wartości dla dwóch liczb całkowitych wprowadzanych z klawiatury:

```
int a, b, minimum;
cout << "podaj dwie liczby całkowite:" << endl;
cin >> a >> b;
minimum = a < b ? a : b;
cout << "minimum = " << minimum << endl;
```

Przykładowy wynik:

podaj dwie liczby całkowite:

12

7

minimum = 7

A

AAC, Advanced Audio Coding, 182

Access 2013, 284

- autoformularze, 344
- formularze, 344
- funkcje wbudowane, 321
- kwerendy, 313
- projektowanie
 - formularzy, 346
 - tabel, 284, 286
- raporty, 352
- system QBE, 313
- tabele, 284, 296
- typy danych, 287
- widok
 - formularza, 346
 - SQL, 335
- właściwości pól, 291
- zapytania SQL, 313

aktualizowanie danych w tabeli, 332

algorytm, 12

- Euklidesa, 85, 91
- Huffmana, 207, 208, 209
- kompresji MP3, 181
- LZW, 198
- naiwny, 82
- rozwiązujący równanie kwadratowe, 14, 18
- RSA, 167
- szybkiego potęgowania, 116
- Vorbis, 182

algorytmy

- efektywność, 79, 82
- liniowe, 129
- na tekstach, 151
- optymalność, 85
- podstawieniowe, 164
- poprawność, 84
- przetawieniowe, 161
- skończoność, 84
- szyfrujące
 - asymetryczne, 160
 - symetryczne, 160
- uniwersalność, 84
- złożoność obliczeniowa, 79

alokacja pamięci, 30

anagram, 154

analiza

- częstotliwości, 181
- danych w tabeli, 325

animacja

- kamery, 222
- koloru materiału, 222
- komputerowa, 218
- kształtu, 222
- poklatkowa, stop motion, 219
- poklatkowa w programie Blender, 220
- ustawienia renderowania, 224

arkusz kalkulacyjny, 227

- funkcje, 227
- funkcje generujące dane, 228

atrybuty barwy, 191

Audacity, 179

- analiza częstotliwości, 181, 182
- Cofnij w czasie, 179
- Eksportuj jako MP3, 180
- interfejs, 179
- narzędzie
 - do przesuwania w czasie, 183
 - obwiedni, 184
 - zaznaczanie, 179
- Redukcja szumu..., 181
- Uzyskaj profil szumu, 181

autoformularze, 344, 346

Autonumerowanie, 289

B

barwa, 186

- głębina bitowa, color depth, 195
- jaskrawość, brightness, 191
- jasność, lightning, 191
- nasycenie, saturation, 191
- odcień, hue, 191

barwy czyste, 192

bazy danych, 275, 277

- bezpieczeństwo, 355
- dekodowanie, 358
- diagram obiektowo-związkowy, 282
- funkcje zabezpieczeń, 358

jednoznaczna identyfikacja danych, 281

język SQL, 313

kartotekowe, 276

klucz główny, 282

klucze obce, 282

kodowanie, 358

nadmiarowość, 279

ochrona hasłem, 356

operacje na tabelach, 306

proste, 276

relacyjne, 275

- integralność, 283

- projektowanie, 278

systemy zarządzania, 275, 277

tabele, 281

typu klient-serwer, 276

złożone, 277

Bell Alexander Graham, 173

bezpieczeństwo danych, 355

biblioteka

- ctime, 131

- iomanip, 27

- iostream, 23

- cstdlib, 131

bitrate, 177

Blender, 211

- animacja poklatkowa, 220

- modelowanie 3D, 211

- modyfikowanie położenia kamery, 215

- okno programu, 212

- oś czasu, Timeline, 221

- przekształcenie sześcianu, 216

- renderowanie

- obrazu, 214

- sceny, 223

- skalowanie, 214

- tryb

- edycji, 216

- obiektu, 216

- ustawienia

- renderowania animacji,

- 224

- kamery, 214

- wstawienie i przesuwanie

- obiektu, 213

błąd

- bezwzględny, 101

- względny, 101

bpp, bits per pixels, 195

C

C++, 22
CBR, Constant Bit Rate, 176
ciąg Fibonacciego, 99
ciągi liczbowe, 134, 138
 generowanie, 129
 liniowe przeszukiwanie, 134, 138
 sprawdzanie
 monotoniczności, 144
CIE La^*b^* , 187
CIE XYZ, 186
CMY, Cyan, Magenta, Yellow, 190
CMYK, Cyan, Magenta, Yellow, black, 190
CZAS, TIME, 236
CZAS.WARTOŚĆ, TIMEVALUE, 238
częstotliwość próbkowania, 175, 176
CZY, IS, 234

D

DATA, DATE, 241
DATA.RÓŻNICA, DATEDIF, 244
decybel, 172
definiowanie
 kryteriów wyboru, 314
 relacji, 300
 wyrażeń, 319
deklaracja
 funkcji, 53
 przestrzeni nazw, 24
 stałej, 29
 standardowej przestrzeni nazw, 23
 zmiennej, 29
 zmiennej typu string, 75
dekrementacja, 33
diagram
 obiektowo-związkowy, 282
 relacji
 jeden do jednego, 303
 wiele do wielu, 305, 306
DŁ, LEN, 251
Długi tekst, 287
drzewo binarne, binary tree, 207
drzewo regularne, 207

drzewo, tree, 207
dynamika dźwięku, 175
DZIEŃ, DAY, 242
DZIEŃ.TYG, WEEKDAY, 245
DZIŚ, TODAY, 240
dźwięki
 edycja, 179
 nagrywanie, 179
 zapis
 analogowy, 173
 cyfrowy, 171, 173

E

edycja makra, 259
element minimalny, 140
enkoder, 179

F

faktoryzacja liczby, 103
filtrowanie
 danych, 309
 zaawansowane, 312
filtry, 310
 zaawansowane, 311
FLAC, Free Lossless Audio Codec, 183
FOR, 262
formanty, 347
 ActiveX, 268
 formularza, 268
format
 AAC, 181, 182
 BMP, 197
 FLAC, 183
 MP3, 176
 WAV, 176
 WAVE, 176
formularze, 269
 interfejs graficzny, 348
 użytkownika, 268
 w programie Access, 344
 z przybornikiem, 269
fps, frames per second, 218
FRAGMENT.TEKSTU, MID, 247
funkcja, 53
 abs, 39
 append, 131
 at, 77
 choice, 130
 clear, 77

cos, 39
CZAS, TIME, 236
CZAS.WARTOŚĆ, TIMEVALUE, 238
CZY, IS, 234
DATA, DATE, 241
DATA.RÓŻNICA, DATEDIF, 244
DŁ, LEN, 251
DZIEŃ, DAY, 242
DZIEŃ.TYG, WEEKDAY, 245
DZIŚ, TODAY, 240
empty, 77
FRAGMENT.TEKSTU, MID, 247
getline, 74, 77
GODZINA, HOUR, 238
GÓRA, ROUNDUP, 254
isalnum, 78
isalpha, 78
isdigit, 78
JEŻELI, IF, 230
LEWY, LEFT, 247
LITERY.MAŁE, LOWER, 246
LITERY.WIELKIE, UPPER, 246
LOS, RAND, 228
LOS.ZAKR, RANDBETWEEN, 229
LUB, OR, 234
main, 24
MIESIĄC, MONTH, 242
MINUTA, MINUTE, 240
NIE, NOT, 232
ORAZ, AND, 232
PODSTAW, SUBSTITUTE, 250
PORÓWNAJ, EXACT, 232, 253
pow, 38
PRAWY, RIGHT, 247
rand, 132
randint, 129
random, 130
randrange, 130
range, 130
ROK, YEAR, 242
s.c_str, 77
s.find, 77
s.substr, 77

funkcja
SEKUNDA, SECOND, 240
sin, 39
size, 77
sqrt, 38
srand, 132
strcat, 73
strcmp, 73
strcpy, 73
strlen, 73
strncpy, 73
SZUKAJ.TEKST, SEARCH,
251
tan, 39
TERAZ, NOW, 241
time, 132
uniform, 130
USUŃ.ZBĘDNE.ODSTĘPY,
TRIM, 250
WARTOŚĆ, VALUE, 252
Z.WIELKIEJ. LITERY,
PROPER, 246
ZAKR, ROUND, 253
ZAKR.DO.CAŁK, INT,
255
ZAKR.DO.NPARZ, ODD,
255
ZAKR.DO.PARZ, EVEN,
255
ZAKR.DÓŁ,
ROUNDDOWN, 254
ZAKR.W.DÓŁ, FLOOR,
254
ZAKR.W.GÓRĘ,
CEILING, 254
ZASTĄP, REPLACE, 250
ZŁĄCZ.TEKSTY,
CONCATENATE, 248
ZNAJDŹ, FIND, 251
ZNAK, CHAR, 252

funkcje
argumenty, 54
arkusza kalkulacyjnego, 227
czasu, 235
daty, 240
deklaracja, 53
generujące dane, 228
matematyczne, 38
nagłówek, 53
programu Access, 321
przekazywanie parametrów,
57

tekstowe, 246, 248
wywołanie, 55
zabezpieczeń bazy danych,
358
zaokrąglenia, 253

G

generowanie ciągów
liczbowych, 129
GIMP, 218
głębina bitowa, color depth, 195
GODZINA, HOUR, 238
GÓRA, ROUNDUP, 254

H

hałas, 184
hasła, 356
hierarchia tabel, 302
Hiperłącze, 291
HLS, 191
HSB, 191
HSV, 191
Huffman David, 207

I

idol, 146, 150
indeks, 295
infradźwięki, 172
inicjalizacja
tablicy, 63
zmiennej typu string, 75
inkrementacja, 33
Inkscape, 189
mieszanie barw RGB, 189
narzędzie Wskaźnik, 189
Wypełnienie i kontur..., 189
instrukcja
break, 44, 51
continue, 51
do-while, 49
for, 47
GROUP BY, 341
ORDER BY, 337
return, 54
SELECT, 336
switch, 42
TRANSFORM, 340
while, 46
instrukcje
wyboru, 42
złożone, 40

iteracyjne, 46
sterujące, 51
warunkowe, 41, 265
integralność bazy danych, 283
interfejs graficzny formularza,
348
iteracja, 88

J

jaskrawość, brightness, 194
jasność, lightness, 194
jednostka głębi bitowej, bpp,
195
JEŻELI, IF, 230
język
C++, 22
SQL, 313, 335

K

kamera, 214
modyfikowanie położenia,
215
klatki
kluczowe, key frames, 220
pośrednie, frames, 220
klauszula
FROM, 337
INTO, 340
Is Not Null, 339
WHERE, 338, 342
klucz
główny, 282
obcy, 282, 297
podstawowy, 297
prywatny, 161
publiczny, 161
w bazie danych, 281
kodowanie
długości ciągów, 197
Huffmana, 205
kolor, 186
koło chromatyczne, 192
komentarze, 40
kompresja
bezstratna, 197
kodowanie Huffmana,
205
LZW, 198
RLE, 197
MP3, 181
stratna, 209

konstruktor wyrażeń, 320, 321
konwersja
AC, 173
liczb, 123
liczb rzeczywistych, 110
łańcuchów, 78
kreator
formularza, 346
kwerend krzyżowych, 327
masek, 292
krotka, 284
Krótki tekst, 287
kryptoanaliza, 159
kryptografia, 159
kryptogram, 165, 166
kryteria wyboru, 338
złożone, 314
znaki wieloznaczne, 316
kwantyzacja, 175
kwerenda tworząca tabelę, 329, 340
kwerendy
aktualizujące, 332
dołączające, 330, 331
funkcjonalne, 329
instrukcje języka SQL, 341
konfiguracja, 320
krzyżowe, 325, 328, 340
modyfikujące, 329
parametryczne, 324, 340
usuwające, 333
widok
arkusza danych, 313
projektu, 314
SQL, 314
wybierające, 314, 337
z polem obliczeniowym, 323, 339
z wyrażeniem, 319

L
LEWY, LEFT, 247
liczba
klatek na sekundę, fps, 218
Smitha, 105
superpierwsza, 169
liczby
błąd
bezwzględny, 101
względny, 101
całkowite, 125
czynniki pierwsze, 103

faktoryzacja, 103
lustrzane, 108
naturalne, 125
notacja naukowa, 99
półpierwsze, 108
pseudolosowe, 129, 131
rzeczywiste, 110
ujemne, 126
w polach programu Access, 288
lider, 146
LITERY.MAŁE, LOWER, 246
LITERY.WIELKIE, UPPER, 246
LOS, RAND, 228
LOS.ZAKR, RANDBETWEEN, 229
LUB, OR, 234
LZW, Lempel-Ziv-Welch, 198

Ł
łańcuchy, 72
konwersje, 78

M
makropolecenia, 256
edycja, 259
nagrywanie, 256
przypisanie do przycisku formularza, 266
uruchamianie, 258
manipulator
dec, 27
endl, 27
ends, 27
hex, 27
oct, 27
setfill, 27
setprecision, 27
setw, 27
mapa bitowa, 197
maska
kreator, 292
symbole, 292
wprowadzania, 291
MIESIĄC, MONTH, 242
mieszanie
barw RGB, 189, 190
kolorów CMY, 190
minimum, 140
MINUTA, MINUTE, 240
mnożenie, 118

model barw
CIE La*b*, 187
CIE XYZ, 186
CMY, 190
CMYK, 190
HLS, 191
HSB, 191
HSV, 191
RGB, 187
modelowanie 3D, 211
monotoniczność ciągu
liczbowego, 144
MP3, MPEG-1/2 audio layer-3, 176

N
nagrywanie
makropolecenia, 256
muzyki, 178
najmniejsza wspólna wielokrotność, 92
największy wspólny dzielnik, 92
nasylenie, saturation, 193
nawias
klamrowy, 40
kwadratowy, 324
NIE, NOT, 232
notacja naukowa liczb, 99
Null, 290

O
obiekt OLE, 290, 308
obliczanie najmniejszej wspólnej wielokrotności, 92
obraz, 185
odcień, hue, 191
Ogg, 182
OLE, Object Linking and Embedding, 290, 308
opcje zgrywania muzyki, 178
operacje
arytmetyczne, 116
wejścia-wyjścia, 25
operator
&, 248
, 30
adresu &, 30
delete, 30
INNER JOIN, 337
LEFT JOIN, 337
łączenia napisów, 235

operator
 new, 30
 przypisania =, 32
 warunkowy, 36
operatory
 arytmetyczne, 32
 logiczne, 34
 przypisania złożone, 35
 relacyjne, 34
 zmniejszania i zwiększania,
 33
ORAZ, AND, 232
oś czasu, Timeline, 220

P

paleta barw, 225
palindrom, 151
parametry
 formalne, 54
 aktualne, 54
PESEL, 248
pętla FOR, 262
płyta CD, 175
podformularz, 351
podpis elektroniczny, 161
PODSTAW, SUBSTITUTE, 250
pola
 Format, 291
 Indeksowane, 295
 kluczowe tabeli, 297
 Komunikat o błędzie, 294
 Maska wprowadzania, 291
 obliczeniowe, 339
 Reguła poprawności, 294
 Rozmiar pola, 291
 Wartość domyślna, 293
 właściwości indeksowania,
 295
 Wymagane, 294
 Zerowa dł. dozwolona, 295
typu
 Autonumerowanie, 289
 Długi tekst, 287
 Hiperłącze, 291
 Krótki tekst, 287
 Liczba, 288
 obiekt OLE, 290
 Tak/Nie, 290
 Waluta, 289
 Załącznik, 290
pomiar poziomu hałasu, 184

PORÓWNAJ, EXACT, 232, 253
potomek, child node, 207
PRAWY, RIGHT, 247
priorytety relacji i działań, 37
program, 23
 Access 2013, 284
 Audacity, 179
 Blender, 211
 GIMP, 218
 Inkscape, 189
 Windows Media Player, 177
programy
 część
 deklaracyjna, 23
 operacyjna, 24
projektowanie
 bazy danych, 278
 formularzy, 346
próbkiwanie, 174
przekazywanie parametrów
 przez referencję, 58
 przez wartość, 57
 przez wskaźnik, 58
przetawienie kolumnowe, 161
przeźrzeń nazw, 24
przeszukiwanie
 binarne, 85
 ciągu liczbowego, 134
 z wartownikiem, 138
 liniowe, 85
pseudokod, 12

Q

QBE, Query By Example, 313

R

raporty, 352
referencje, 31
reguła poprawności, 294
rekurencja, 88
relacje, 300
 jeden do jednego, 283, 300,
 301
 jeden do wielu, 283, 300, 304
 wiele do wielu, 284, 300, 305
renderowanie, 214
 sceny, 223
reprezentacja liczb
 binarnych, 126
 całkowitych, 125
 naturalnych, 125

 stałopozycyjna, 125
 zmiennopozycyjna, 127
RGB, Red, Green, Blue, 187
 mieszanie barw, 189
RLE, Run-Length Encoding,
 197
rodzic, parent node, 207
ROK, YEAR, 242
równanie kwadratowe, 14

S

schemat blokowy
 algorytm delty, 16
 ciąg rosnący, 145
 instrukcja
 do-while, 49
 for, 47
 if, 41
 switch, 44
 while, 46
 palindrom, 151
 przeszukiwanie ciągu
 liczbowego, 134
 z wartownikiem, 138
 rozwiązanie równania
 kwadratowego, 20
 schemat Hornera, 122
 sprawdzanie warunków, 136,
 137
 wyznaczanie maksimum,
 142
schemat Hornera, 85, 120, 123
SEKUNDA, SECOND, 240
silnia liczby naturalnej, 88
sito Eratostenesa, 103, 105
słowo kluczowe
 break, 43, 44, 51
 case, 42
 char, 29
 cin, 25
 continue, 51
 cout, 25
 default, 42
 delete, 30
 do, 49
 else, 41
 float, 29
 for, 47
 if, 41, 42
 int, 24
 long, 29
 new, 30

- return, 54
- std, 24
- string, 75
- switch, 42
- void, 24
- while, 46
- sortowanie, 153, 156, 337
- SQL, Structured Query Language, 313, 335
- kryterium wyboru, 338
- kwerendy, 335
 - krzyżowe, 340
 - parametryczne, 340
 - wybierające, 337
- pola obliczeniowe, 339
- sortowanie, 337
- tworzenie
 - kwerend, 341
 - tabeli, 340
- stała, 29
- standardowy strumień
 - wejściowy, 25
 - wyjściowy, 25
- struktura programu, 23
- sygnał
 - analogowy, 174
 - cyfrowy, 174
- symbole dla masek, 292
- systemy liczbowe
 - binarne, 112, 114
 - heksadecymalne, 112
 - konwersje, 110, 112
 - niepozycyjne, 108
 - oktalne, 112
 - operacje arytmetyczne, 116
 - pozycyjne, 108
- SZUKAJ.TEKST, SEARCH, 251
- szybkie potęgowanie liczb, 114
- szyfrowanie
 - asymetryczne, 167
 - podstawieniowe, 160, 164
 - przetawieniowe, 160
 - symetryczne, 161
 - tekstu, 159
 - wieloalfabetyczne, 166
- szyfry
 - monoalfabetyczne, 164
 - wieloalfabetyczne, 165

T

- tabele, 281, 284
 - aktualizowanie danych, 332
 - analizowanie danych, 325

- dodawanie rekordów, 330
- filtrowanie
 - danych, 309
 - zaawansowane, 312
- filtry zaawansowane, 311
- klucz
 - podstawowy, 297
 - obcy, 297
- kryteria złożone, 313
- kwerenda tworząca, 329, 340
- operacje edycyjne, 306
- pobieranie danych, 314
- pole kluczowe, 297
- relacje, 300
- typy danych, 287
- usuwanie rekordów, 333
- w programie Access, 284
- widok
 - arkusza danych, 296, 309
 - projektu, 296
 - właściwości pól, 291
 - wstawianie obiektów OLE, 308
- tablice, 62
 - barw, 188
 - wielowymiarowe, 66
 - znaków, 72
- tekst
 - funkcje Excela, 246
 - łączenie, 248
 - sortowanie, 153, 156
 - szyfrowanie, 159
- teksty, 151
- TERAZ, NOW, 241
- tworzenie
 - kwerend, 331, 341
 - tabel, 284, 329
- typ danych
 - bool, 52
 - char, 29, 52
 - double, 29, 52
 - float, 29, 52
 - int, 24, 52
 - long double, 52
 - pola, *Patrz także* pola
 - short int, 52
 - string, 75
 - unsigned char, 52
 - unsigned int, 52
 - unsigned long, 29
 - unsigned short int, 52
 - void, 24

- typy danych
 - całkowite, 52
 - logiczne, 52
 - proste, 52
 - rzeczywiste, 52
 - strukturalne, 62
 - w tabelach, 287
 - znakowe, 52

U

- ultradźwięki, 172
- ułamki zwykłe, 93
- uruchamianie makra, 258
- UserForm, 268
- ustawienia
 - renderowania animacji, 224
 - kamery, 214
- USUŃ.ZBĘDNE.ODSTĘPY, TRIM, 250
- usuwanie rekordów w tabeli, 333

V

- VBA, 256
 - formularz, 269
 - instrukcja warunkowa, 265
 - pętla FOR, 262
 - UserForm, 268
- VBR, Variable Bit Rate, 177

W

- Waluta, 289
- WARTOŚĆ, VALUE, 252
- wartość Null, 290
- wartownik, 139
- WAV, 176
- węzeł, node, 207
 - główny, root node, 207
- widok
 - formularza, 346
 - raportu, 354
- wielomian, 120
- wieża Hanoi, 94
- Windows Media Player, 177
- własności
 - barwy, 195
 - pól tabeli, 291
- wskazniki, 30, 31
- wrażenia, 319
 - konstruktor, 320, 321
- wzory Viete'a, 18

Z

- Z.WIELKIEJ. LITERY,
PROPER, 246
- zakres światła widzialnego, 185
- Załącznik, 290
- ZAKR. ROUND, 253
- ZAKR.DO.CAŁK. INT, 255
- ZAKR.DO.NPARZ. ODD, 255
- ZAKR.DO.PARZ. EVEN, 255
- ZAKR.DÓŁ,
ROUNDDOWN, 254
- ZAKR.W.DÓŁ. FLOOR, 254
- ZAKR.W.GÓRĘ. CEILING,
254
- zapytania, *Patrz* kwerendy
- zarządzanie bazą danych, 275
- ZASTĄP. REPLACE, 250
- zastosowanie
 - algorytmu Euklidesa, 91
 - manipulatorów, 28
 - operatorów zmniejszania, 34
 - schematu Hornera, 123
 - złożonych operatorów
przypisania, 36
- zbiory
 - idol, 146, 150
 - lider, 146
- zliczanie znaków, 158
- ZŁĄCZ.TEKSTY,
CONCATENATE, 248
- złączenie
 - wewnętrzne, 337
 - zewnętrzne, 337
- złożoność obliczeniowa
 - algorytmu, 79
 - czasowa, 80
 - kwadratowa, 81
 - liniowa, 81
 - liniowo-logarytmiczna, 81
 - logarytmiczna, 81
 - oczekiwana, 80
 - pamięciowa, 80
 - pesymistyczna, 80
 - stała, 81
 - wykładnicza, 81, 82
- zmienne, 29
 - dynamiczne, 30, 31
 - globalne, 56
 - lokalne, 56
 - wskaźnikowe, 30
 - adres, 30
 - wartość, 30
- znajdowanie
 - anagramów, 156, 158
 - lidera w zbiorze, 146
 - minimalnego elementu, 140
- ZNAJDŹ. FIND, 251
- ZNAK. CHAR, 252
- znak specjalny, 25
- znaki wieloznaczne, 316

PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion 

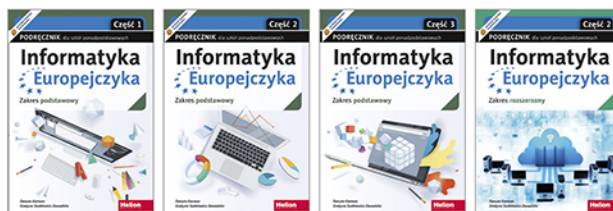
Technologia informacyjna i informatyka to dziedziny, których wykorzystanie i dostępność stale rosną, a tempo zachodzących w nich zmian jest nieporównywalne z rozwojem innych dyscyplin. Umiejętność biegłego posługiwania się komputerem i urządzeniami peryferyjnymi oraz znajomość obsługi pakietów biurowych są obowiązkowe na rynku pracy. Wkrótce każdy z nas będzie musiał się również wykazać wiedzą z zakresu programowania, tworzenia grafiki czy zagadnień związanych z prawem w sieci.

Informatyka w szkole ponadpodstawowej na poziomie rozszerzonym:

- bazuje na wiedzy, którą zdobywa się na poziomie podstawowym, i stanowi jej uzupełnienie
- przygotowuje do studiów informatycznych
- jest przedmiotem maturalnym

Dzięki pracy z książką *Informatyka Europejczyka. Podręcznik dla szkół ponadpodstawowych. Zakres rozszerzony. Część 1* poszerzysz znajomość języków tekstowych, takich jak Python i C++. Zgłębisz wiedzę na temat algorytmów i zaczniesz ją wykorzystywać między innymi w zadaniach z zakresu kryptografii i kryptoanalizy. Opanujesz przydatne metody edycji dźwięku i kompresji danych. Przygotujesz w programie Blender przestrzenne wizualizacje i animacje. Zastosujesz zaawansowane funkcje arkusza kalkulacyjnego dla różnego typu danych, a do zdefiniowania makropoleczeń użyjesz wbudowanego do niego języka programowania VBA. Ponadto zaprojektujesz i utworzysz relacyjną bazę danych, przygotujesz formularze, kwerendy i raporty. Poznasz język zapytań SQL. Zadbasz o integralność i bezpieczeństwo zgromadzonych danych.

W skład kompletnego pakietu edukacyjnego dla szkoły ponadpodstawowej wchodzi także podręczniki:



Podręczniki z serii *Informatyka Europejczyka* ułatwią uczniom zdobywanie wiedzy i umiejętności podczas wykonywania ćwiczeń praktycznych, a nauczycielom — przekazywanie nowego materiału w interesujący i niebanalny sposób.

Wciśnij Enter i do dzieła!

<http://edukacja.helion.pl>

Helion EDUKACJA	księgarnia internetowa	ISBN 978-83-283-4188-3
	http://helion.pl	
zamówienia telefoniczne		
0 801 339900	Helion SA ul. Kościuszki 1c, 44-100 Gliwice tel.: 32 230 98 63 e-mail: helion@helion.pl http://helion.pl	
0 601 339900		
Informatyka w najlepszym wydaniu		9 788328 341883