Mirosław J. Kubiak

C++

Zadania z programowania z przykładowymi rozwiązaniami

WYDANIE III

Helion

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite / Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki
Grafika na okładce została wykorzystana za zgodą Shutterstock.com

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/cppza3
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-6955-9

Printed in Poland.

Spis treści

Rozdział 1.

Proste operacje wejścia – wyjścia

W tym rozdziale zamieściłem proste zadania z przykładowymi rozwiązaniami ilustrujące, w jaki sposób komputer komunikuje się z użytkownikiem w języku C++.

Każda aplikacja powinna mieć możliwość komunikowania się z użytkownikiem. Wykorzystując proste przykłady, pokażę, jak program napisany w języku C++ komunikuje się z użytkownikiem poprzez standardowe operacje wejścia – wyjścia.

Plik nagłówkowy z instrukcji:

```
#include <iostream>
```

zawiera definicje klas[1] umożliwiających wykonywanie operacji wejścia – wyjścia na strumieniach. Do wyprowadzania danych na ekran służy standardowy strumień wyjściowy `cout`, który w języku C++ domyślnie przypisuje ekran do standardowego urządzenia wyjściowego systemu operacyjnego. Aby wyświetlić komunikat lub dane, trzeba do strumienia wyjściowego `cout` zastosować symbol podwójnego znaku mniejszości << (operacja wstawiania). Dwa znaki mniejszości należy wprowadzić z klawiatury. Do wprowadzania danych do programu służy standardowy strumień wejściowy `cin` oraz operator >> (dwa znaki większości, które również wprowadzamy z klawiatury), np. `cin >> a;`.

[1] Więcej informacji na temat klas Czytelnik znajdzie w rozdziale 6.

W programie, dla wygody, użyłem **przestrzeni nazw** (ang. *namespace*):

```
using namespace std;
```

Dzięki temu wygodniejsze będzie posługiwanie się strumieniami wejściowymi `cin` i wyjściowymi `cout`[2].

Do formatowania strumienia wyjściowego będziemy używali flagi formatującej `fixed` i manipulatora `setprecision(n)`. Flaga `fixed` stosuje do liczb zmiennoprzecinkowych ustaloną kropkę dziesiętną, natomiast manipulator `setprecision(n)` ustala ich precyzję na *n*, np. zapis `cout << setprecision(2);` oznacza, że liczby zmiennoprzecinkowe będą wyświetlane z dokładnością do dwóch miejsc po kropce.

Zastosowanie manipulatora `setprecision(n)` wymaga włączenia do programu pliku nagłówkowego:

```
#include <iomanip>
```

Opisane powyżej podejście do operacji wejścia – wyjścia nazywa się obiektowym[3].

Zadanie

1.1 Napisz program, który oblicza pole prostokąta. Wartości boków `a` i `b` wprowadzamy z klawiatury. W programie należy przyjąć, że zmienne `a` i `b` oraz `pole` są typu `float` (rzeczywistego). Przyjmujemy format wyświetlania ich na ekranie z dokładnością do dwóch miejsc po kropce.

Przykładowe rozwiązanie — listing 1.1

```
// Zadanie 1.1

#include <iostream>
#include <iomanip>
#include <conio.h>

using namespace std;
```

[2] Umieszczenie dyrektywy `using` na poziomie globalnym (na poziomie pliku) czyni zawartość przestrzeni nazw dostępną globalnie (tzn. w całym pliku) [1].

[3] Więcej informacji na temat obiektowych operacji wejścia – wyjścia, flag i manipulatorów znajdzie Czytelnik w bibliografii [1, 2] lub na stronach WWW poświęconych językowi programowania C++ pod adresem *http://www.cplusplus.com/*.

```
int main()
{
   float a, b, pole;

   cout << "Program oblicza pole prostokąta." << endl;
   cout << "Podaj bok a." << endl;
   cin >> a;
   cout << "Podaj bok b." << endl;
   cin >> b;

   pole = a*b;

   fixed; // flaga
   cout << setprecision(2); // ustalenie precyzji
   cout << "Pole prostokąta o boku a = " << a << " i boku b = " << b;
   cout << " wynosi " << pole << "." << endl;

   _getch(); // naciśnij klawisz Enter

   return 0;
}
```

Linijka kodu:

```
float a, b, pole;
```

umożliwia zadeklarowanie zmiennych `a`, `b` i `pole` (wszystkie zmienne w programie są typu rzeczywistego `float`). Instrukcja:

```
cout << "Program oblicza pole prostokąta." << endl;
```

wyświetla na ekranie komputera komunikat *Program oblicza pole prostokąta*. Instrukcja `cin >> a;` czeka na wprowadzenie z klawiatury komputera liczby, która następnie zostanie przypisana zmiennej `a`. Pole prostokąta zostaje obliczone w wyrażeniu:

```
pole = a*b;
```

Za wyświetlenie wartości zmiennych `a` i `b` oraz `pole` wraz z odpowiednim opisem są odpowiedzialne następujące linijki kodu:

```
cout << "Pole prostokątna o boku a = " << a << " i boku b = " << b;
cout << " wynosi " << pole << "." << endl;
```

Flaga `fixed` używa ustalonej kropki dziesiętnej dla liczb zmiennoprzecinkowych. Zapis:

```
cout << setprecision(2);
```

oznacza, że liczby te będą wyświetlane na ekranie z dokładnością do dwóch miejsc po kropce. Natomiast funkcja:

```
_getch(); // naciśnij klawisz Enter
```

(ang. *get character* — wczytaj znak) czeka na naciśnięcie klawisza *Enter* (lub na naciśnięcie dowolnego klawisza). **UWAGA!** Przed literą `g` w instrukcji `_getch()` znajduje się znak podkreślenia `_` .

Prototyp tej funkcji znajduje się w pliku nagłówkowym *conio.h*. Instrukcja:

```
endl;
```

(ang. *end of line* — koniec linii) przenosi kursor na początek następnej linii.

Komentarze w programie oznaczamy dwoma ukośnikami:

```
// to jest komentarz do programu
```

Są one ignorowane w procesie kompilacji.

Rezultat działania programu można zobaczyć na rysunku 1.1.

Rysunek 1.1. *Efekt działania programu Zadanie 1.1*

```
Program oblicza pole prostokątna.
Podaj bok a.
1
Podaj bok b.
2
Pole prostokątna o boku a = 1.00 i boku b = 2.00 wynosi 2.00.
```

Zadanie 1.2

Napisz program, który wyświetla na ekranie komputera wartość predefiniowanej stałej $\pi = 3{,}14\ldots$ Należy przyjąć format prezentowania tej stałej, oznaczanej w języku C++ jako `M_PI`, z dokładnością do pięciu miejsc po kropce.

Stałe matematyczne, w tym `M_PI`, są rozszerzeniem biblioteki standardowej `math.h`. Jeśli chcemy użyć tych stałych w programie w środowisku Microsoft Visual C++, to należy zadeklarować za pomocą preprocesora zmienną `_USE_MATH_DEFINES` przed dołączeniem pliku nagłówkowego `math.h`. Ilustrują to następujące instrukcje zamieszczone w programie:

```
#define _USE_MATH_DEFINES
#include <math.h>
```

Przykładowe rozwiązanie — listing 1.2

```
// Zadanie 1.2

#include <iostream>
#include <iomanip>
#define _USE_MATH_DEFINES
#include <math.h>
#include <conio.h>

using namespace std;

int main()
{
   cout << "Program wyświetla wartość predefiniowanej stałej pi" << endl;
   cout << "z dokładnością do pięciu miejsc po kropce." << endl;
   cout << "pi = " << fixed << setprecision(5) << M_PI << endl;

   _getch(); // naciśnij klawisz Enter

    return 0;
}
```

Rezultat działania programu można zobaczyć na rysunku 1.2.

Rysunek 1.2. *Efekt działania programu Zadanie 1.2*

```
Program wyświetla wartość predefiniowanej stałej pi
z dokładnością do pięciu miejsc po kropce.
pi = 3.14159
```

Zadanie 1.3

Napisz program, który wyświetla na ekranie komputera pierwiastek kwadratowy z wartości predefiniowanej stałej π = 3,14… Należy przyjąć format wyświetlania tego pierwiastka z dokładnością do dwóch miejsc po kropce.

Pierwiastek kwadratowy ze stałej π obliczamy, korzystając z funkcji `sqrt()`. Funkcja ta znajduje się w pliku nagłówkowym *math.h*.

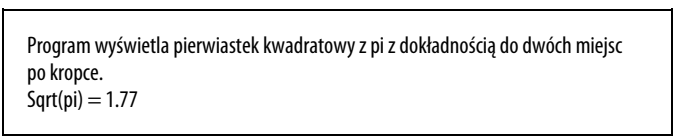
Przykładowe rozwiązanie — listing 1.3

```
// Zadanie 1.3

#include <iostream>
#include <iomanip>
#define _USE_MATH_DEFINES
#include <math.h>
#include <conio.h>
```

```
using namespace std;

int main()
{
   cout << "Program wyświetla pierwiastek kwadratowy z pi" << endl;
   cout << "z dokładnością do dwóch miejsc po kropce." << endl;

   cout << "Sqrt(pi) = " << fixed << setprecision(2) << sqrt(M_PI) << endl;

   _getch(); // naciśnij klawisz Enter

   return 0;
}
```

Rezultat działania programu można zobaczyć na rysunku 1.3.

Rysunek 1.3. *Efekt działania programu Zadanie 1.3*



Zadanie

1.4 Napisz program, który oblicza objętość kuli o promieniu `r`. Wartość promienia wprowadzamy z klawiatury. W programie należy przyjąć, że `r` jest typu `double` (rzeczywistego). Dla zmiennych `r` oraz `objetosc` należy przyjąć format wyświetlania ich na ekranie z dokładnością do dwóch miejsc po kropce.

Przykładowe rozwiązanie — listing 1.4

```
// Zadanie 1.4

#include <iostream>
#include <iomanip>
#define _USE_MATH_DEFINES
#include <math.h>
#include <conio.h>

using namespace std;

int main()
{
   double r, objetosc;

   cout << "Program oblicza objętość kuli o promieniu r." << endl;
   cout << "Podaj promień r." << endl;
   cin >> r;
   objetosc = 4*M_PI*r*r*r/3;
```

```
    cout << fixed << setprecision(2);
    cout << "Objętość kuli o promieniu r = " << r << " wynosi ";
    cout << objetosc << "." << endl;

    _getch(); // naciśnij klawisz Enter

    return 0;
}
```

Objętość kuli o promieniu `r` oblicza następująca linijka kodu:

```
objetosc = 4*M_PI*r*r*r/3;
```

gdzie potęgowanie zostało zamienione na mnożenie.

Rezultat działania programu można zobaczyć na rysunku 1.4.

Rysunek 1.4. *Efekt działania programu Zadanie 1.4*

```
Program oblicza objętość kuli o promieniu r.
Podaj promień r.
1
Objętość kuli o promieniu r = 1.00 wynosi 4.19.
```

Zadanie

1.5 Napisz program, który oblicza wynik dzielenia całkowitego bez reszty dla dwóch liczb całkowitych: $a = 37$ i $b = 11$.

W języku C++ w przypadku zastosowania operatora dzielenia / dla liczb całkowitych reszta wyniku jest pomijana.

Przykładowe rozwiązanie — listing 1.5

```
// Zadanie 1.5

#include <iostream>
#include <conio.h>

using namespace std;

int main()
{
    auto a = 37, b = 11;

    cout << "Program wyświetla wynik dzielenia całkowitego" << endl;
    cout << "dla dwóch liczb całkowitych." << endl;
    cout << "Dla liczb a = " << a << " i b = " << b << endl;
    cout << a << "/" << b << " = " << a/b << "." << endl;
```

```
    _getch();  // naciśnij klawisz Enter

    return 0;
}
```

W standardzie C++ 11 wprowadzono udogodnienie pozwalające kompilatorowi dedukować typ zmiennej na podstawie typu wartości inicjalizującej — jest to **automatyczna deklaracja typów**. Słowo kluczowe `auto` umożliwia kompilatorowi przydzielenie zmiennej tego samego typu co wartość inicjalizująca. W ten sposób omija się deklarację typu obowiązującą w starej składni języków C i C++, np. `int a = 37`. Ilustruje to następująca linijka kodu:

```
auto a = 37, b = 11;
```

W standardzie C++ 11 również wprowadzono inną inicjalizację dla pojedynczych zmiennych. Zastosowano inicjalizator klamrowy zamiast znaku przypisania =. Następujące linijki kodu są równoważne:

```
auto a = 37, b = 11;
```

i

```
auto a{37}, b{11};
```

W tej książce stosuję wyłącznie starą składnię. Zachęcam jednak Czytelnika, aby w ramach ćwiczeń dodatkowych zastosował inicjalizator klamrowy zamiast znaku przypisania =.

Rezultat działania programu można zobaczyć na rysunku 1.5.

Rysunek 1.5. *Efekt działania programu Zadanie 1.5*

```
Program wyświetla wynik dzielenia całkowitego
dla dwóch liczb całkowitych.
Dla liczb a = 37 i b = 11
37/11 = 3.
```

Zadanie 1.6

Napisz program, który oblicza resztę z dzielenia całkowitego dla dwóch liczb całkowitych $a = 37$ i $b = 11$.

Należy zastosować operator reszty z dzielenia całkowitego modulo, który oznaczamy w języku C++ symbolem %. Operator ten umożliwia uzyskanie tylko reszty z dzielenia, natomiast całkowita wartość liczbowa jest odrzucana.

Przykładowe rozwiązanie — listing 1.6

```
// Zadanie 1.6

#include <iostream>
#include <conio.h>

using namespace std;

int main()
{
   auto a = 37, b = 11;

   cout << "Program oblicza resztę z dzielenia całkowitego" << endl;
   cout << "dla dwóch liczb całkowitych." << endl;
   cout << "Dla liczb a = " << a << " i b = " << b << endl;
   cout << a << "%" << b << " = " << a%b << "." << endl;

   _getch();  // naciśnij klawisz Enter

   return 0;
}
```

Rezultat działania programu można zobaczyć na rysunku 1.6.

Rysunek 1.6. *Efekt działania programu Zadanie 1.6*

```
Program oblicza resztę z dzielenia całkowitego dla dwóch liczb całkowitych.
Dla liczb a = 37 i b = 11
37%11 = 4.
```

Zadanie

1.7 Napisz program, który oblicza sumę, różnicę, iloczyn i iloraz dla dwóch liczb `x` i `y` wprowadzanych z klawiatury. W programie przyjmujemy, że liczby `x` i `y` są typu `float` (rzeczywistego). Dla zmiennych `x`, `y`, `suma`, `roznica`, `iloczyn` i `iloraz` należy przyjąć format wyświetlania ich na ekranie z dokładnością do dwóch miejsc po kropce.

Przykładowe rozwiązanie — listing 1.7

```
// Zadanie 1.7

#include <iostream>
#include <iomanip>
#include <conio.h>

using namespace std;

int main()
```

```
{
   float x, y, suma, roznica, iloczyn, iloraz;

   cout << "Program oblicza sumę, różnicę, iloczyn i iloraz" << endl;
   cout << "dla dwóch liczb x i y wprowadzanych z klawiatury." << endl;
   cout << endl; // wydruk pustej linii

   cout << "Podaj x." << endl;
   cin >> x;
   cout << "Podaj y." << endl;
   cin >> y;

   suma = x+y;
   roznica = x-y;
   iloczyn = x*y;
   iloraz = x/y;

   cout << fixed << setprecision(2) << endl;
   cout << "Dla x = " << x << " i y = " << y << endl;
   cout << endl;
   cout << "suma = " << suma << "," << endl;
   cout << "różnica = " << roznica << "," << endl;
   cout << "iloczyn = " << iloczyn << "," << endl;
   cout << "iloraz = " << iloraz << "." << endl;

   _getch(); // naciśnij klawisz Enter

   return 0;
}
```

Rezultat działania programu można zobaczyć na rysunku 1.7.

Rysunek 1.7. *Efekt działania programu Zadanie 1.7*

```
Program oblicza sumę, różnicę, iloczyn i iloraz
dla dwóch liczb x i y wprowadzanych z klawiatury.

Podaj x.
1
Podaj y.
2
Dla x = 1.00 i y = 2.00

suma = 3.00,
różnica = -1.00,
iloczyn = 2.00,
iloraz = 0.50.
```


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

Poznaj język C++ od strony praktycznej

C++ to bez wątpienia jeden z najpopularniejszych i najpotężniejszych języków programowania. Znajduje zastosowanie w tworzeniu systemów operacyjnych, sterowników przemysłowych, bibliotek, gier komputerowych, najrozmaitszych aplikacji desktopowych, programów bazodanowych i oprogramowania serwerowego — słowem, wszędzie tam, gdzie liczy się wydajność i niskie zużycie zasobów. Co więcej, napisane w nim programy są przenośne, dzięki czemu można je kompilować pod różne platformy sprzętowe i systemowe. C++ jest też doskonałym językiem do nauki programowania, również dlatego, że jego składnię wykorzystuje się w wielu innych językach.

Teoretyczna nauka programowania jest jak czytanie o lataniu — można się w ten sposób dużo dowiedzieć, ale z pewnością nie zapewni to doświadczenia niezbędnego, by naprawdę wystartować. Dlatego z językiem programowania warto zapoznać się od strony praktycznej: pisać kod, wykonywać ćwiczenia programistyczne, wykorzystywać kolejne techniki i konstrukcje języka, a przede wszystkim mierzyć się z coraz trudniejszymi zadaniami. Świetnym wsparciem w tym działaniu będzie najnowsze wydanie książki *C++. Zadania z programowania z przykładowymi rozwiązaniami*. Dzięki niej dowiesz się, jak wykorzystać bezpłatne środowisko Visual Studio Community 2019 firmy Microsoft do tworzenia aplikacji konsolowych o prostym, przejrzystym kodzie, oraz szybko opanujesz C++.

- **Proste operacje wejścia-wyjścia**
- **Instrukcje warunkowe i iteracje**
- **Operacje na tablicach i macierzach**
- **Podprogramy i rekurencja**
- **Programowanie obiektowe**
- **Przetwarzanie plików tekstowych**
- **Wskaźniki i zmienne dynamiczne**
- **Szablony i standardowa biblioteka STL**

Zostań mistrzem programowania w C++

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

ISBN 978-83-283-6955-9

Cena: 39,90 zł