Temat C2. Tworzenie programów w języku C++

Warto powtórzyć

1. Jaką funkcję pełni pamięć operacyjna (RAM)?
2. Jakie informacje należy umieścić w specyfikacji problemu?
3. Wyjaśnij pojęcia: *język programowania*, *program komputerowy*, *słowo kluczowe*.
4. Dlaczego o języku programowania mówimy, że jest językiem formalnym?
5. Czym jest kod źródłowy?
6. Z czego powinna wynikać kolejność instrukcji programu?
7. Na czym polega kompilacja programu?
8. Dlaczego błędnie napisane instrukcje nie zostaną wykonane?

1. Środowisko programistyczne języka C++

Aby pisać programy w języku C++, należy zainstalować wybraną aplikację zawierającą edytor i kompilator języka C++, np., Code::Blocks, Dev-C++. Są one dostępne bezpłatnie w Internecie. Można także skorzystać z rozwiązań komercyjnych, np. Visual Studio, C++Builder.

Wyspecjalizowany pakiet programistyczny zawiera **edytor kodu źródłowego**, kompilator i inne narzędzia wspomagające programowanie. Taką aplikację (lub zespół aplikacji) określa się też jako **zintegrowane środowisko programistyczne**. W tym podręczniku będziemy korzystać ze środowiska Dev-C++.

Z **Zintegrowane środowisko programistyczne**
(IDE – z ang. *Integrated Development Environment*)

Aplikacja lub zespół aplikacji do tworzenia, modyfikowania, testowania i utrzymywania oprogramowania.

1.1. Struktura programu w języku C++

Prosty program w języku C++ składa się z **poleceń preprocesora** (tzw. **dyrektyw**) oraz **funkcji głównej** `main()`.

W programie musi wystąpić funkcja `main()`, od której rozpoczyna się wykonanie programu. Pozostałe elementy programu są opcjonalne. Instrukcje języka C++ kończymy średnikami.

W języku C++ **blok programu** wyznaczają nawiasy klamrowe `{}`. **Komentarze** (ignorowane przez kompilator) umieszczamy po znakach `//` w tej samej linii lub wewnątrz znaków `/*` (początek bloku komentarza) i `*/` (koniec bloku komentarza).

Przykładowe elementy programu, ułożone w zwykle stosowanej kolejności, pokazujemy na rysunku 1.

Preprocesor P

Program, który przetwarza wstępnie kod źródłowy przed rozpoczęciem zasadniczej kompilacji. Polecenia preprocesora (zwane dyrektywami) rozpoczynają się od znaku #. Np. dyrektywa `#include` nakazuje dołączenie do kodu źródłowego kodu z innych plików źródłowych.

C++

```
#include <iostream>
using namespace std;
// deklaracja zmiennych globalnych (jeśli występują)
// deklaracja funkcji (jeśli występują)

int main()
{
      // deklaracja zmiennych lokalnych (jeśli występują)
      // część wykonawcza programu
   return 0;
}
```

Rys. 1. *Ogólna struktura programu w języku C++*

1.2. Etapy tworzenia programu w języku C++

W każdym języku kompilowanym możemy wyróżnić następujące etapy tworzenia programu:

1. W edytorze wbudowanym w środowisko programistyczne piszemy kod źródłowy programu (rys. 2.).
2. Zapisujemy kod źródłowy programu w pliku (w języku C++ zwykle w pliku z rozszerzeniem *cpp*).
3. Kompilujemy program, korzystając z opcji **Kompiluj** (**Compile**). Przebieg kompilacji obserwujemy w dolnej części okna programu (rys. 2.). Jeśli program został skompilowany, zostanie utworzony tzw. **plik wykonywalny** z rozszerzeniem *exe*.
4. Skompilowany program uruchamiamy, zazwyczaj korzystając z opcji **Uruchom** (**Run**). Wynik działania programu pokaże się w osobnym oknie (rys. 3.).
5. Jeżeli w trakcie kompilacji wystąpiły błędy, należy je poprawić (kompilator wskazuje prawdopodobne miejsce popełnienia błędu – rys. 4a i 4b) i ponownie przejść do kroku 2.

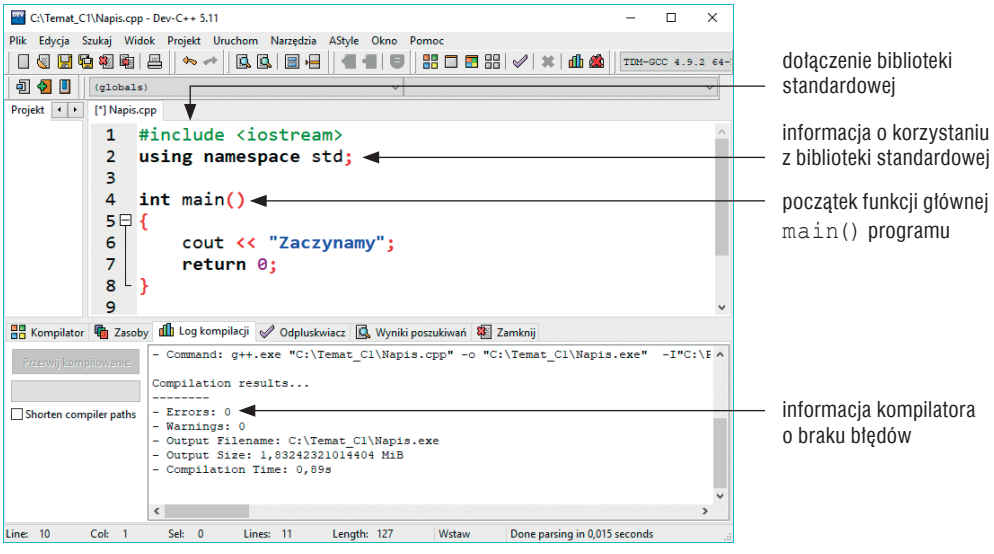
W programowaniu w języku C++, jak w każdym języku, bardzo istotna jest prawidłowa kolejność zapisu poleceń. Instrukcje języka C++, podobnie jak innych języków programowania, realizują takie czynności, jak: wprowadzanie danych, wyprowadzanie wyników, wykonywanie obliczeń, sprawdzanie warunków czy powtarzanie operacji. Poznamy i zastosujemy te instrukcje w ćwiczeniach i zadaniach w tematach C3, C4 i C5.

Uruchomiony i wykonywany może być program, który został poprawnie skompilowany. Jeśli kompilacja nie przebiegła prawidłowo, należy poprawić program i ponownie go skompilować. Raz skompilowany program nie wymaga już powtórnego tłumaczenia.

W trakcie wykonywania programu procesor rozpoznaje i wykonuje instrukcje swojego wewnętrznego języka. Po wykonaniu programu wyniki pojawią się na wybranym urządzeniu zewnętrznym, np. na ekranie monitora (rys. 3.).

Po uruchomieniu środowiska programistycznego języka C++ zobaczymy okno z głównym menu programu. Okno edytora wbudowanego do środowiska programistycznego otwieramy w menu **Plik** – możemy otworzyć istniejący plik (plik źródłowy) lub utworzyć nowy.

Aby utworzyć nowy plik źródłowy, należy w menu **Plik** wybrać opcję **Nowy/Plik źródłowy**.
Aby skompilować program, należy w menu **Uruchom** wybrać opcję **Kompiluj**.
Aby uruchomić program, należy w menu **Uruchom** wybrać opcję **Uruchom**.



Rys. 2. *Okno środowiska programistycznego języka C++ z programem źródłowym – kompilacja przebiegła prawidłowo*

Uwagi:

- Funkcja `main()` zawiera główny program. Przed nazwą funkcji powinno się znajdować słowo kluczowe **`int`**, informujące, że funkcja `main()` zwraca wartość liczbową.
- Każda instrukcja programu powinna być zakończona średnikiem.
- Instrukcja **`return`** `0` na końcu funkcji powoduje zwrócenie do systemu operacyjnego wartości 0, informującej o prawidłowym wykonaniu programu.

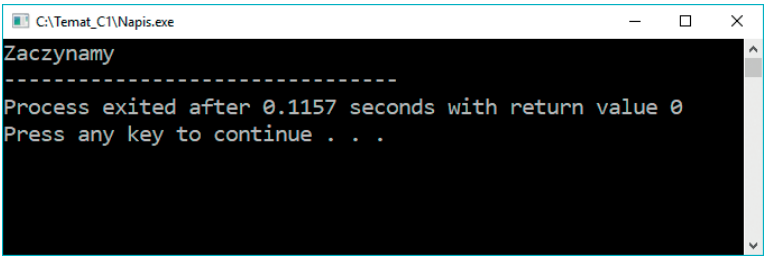
Ćwiczenie 1. Poznajemy etapy tworzenia programu w języku C++

1. Utwórz nowy plik źródłowy i napisz program wyświetlający na ekranie napis „Zaczynamy" – przepisz instrukcje pokazane na rysunku 2.
2. Zapisz program w pliku pod nazwą *Napis*.
3. Skompiluj program.
4. Uruchom program.
5. Jeśli kompilator wykrył błędy, popraw je. Zapisz program w pliku pod tą samą nazwą. Ponownie skompiluj i uruchom program.

Aby powiększyć czcionkę w oknie wykonania programu, należy nacisnąć przycisk w lewym górnym rogu okna i wybrać polecenie **Właściwości/Czcionka**.

Dobra rada

Staraj się samodzielnie poprawiać program do momentu, gdy da się go skompilować i program będzie można uruchomić. W nauce programowania umiejętność wyszukiwania i poprawiania błędów w programie jest bardzo ważna.

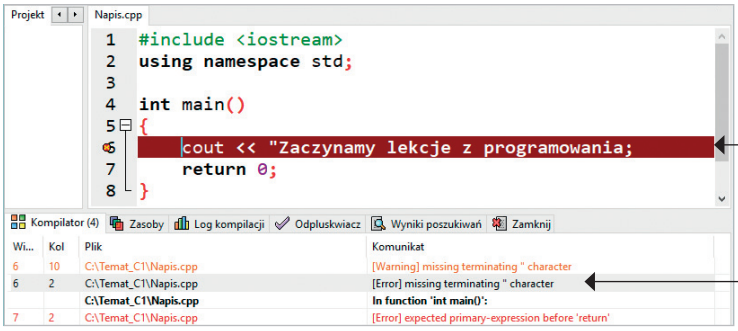


Rys. 3. *Wynik działania programu – ćwiczenie 1.*

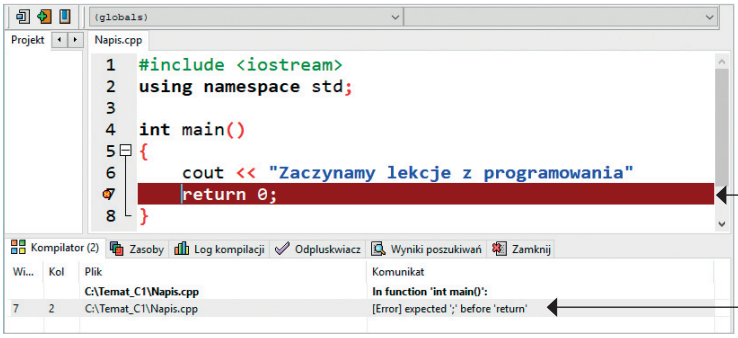
Ćwiczenie 2. Modyfikujemy program

1. Zmodyfikuj program utworzony w ćwiczeniu 1., dodając dalszą część napisu: „lekcje z programowania".
2. Zapisz plik pod tą samą nazwą.
3. Jeśli kompilator wykrył błędy, popraw je. Ponownie zapisz plik i skompiluj program.
4. Jeśli kompilacja przebiegła pomyślnie, uruchom program.

Wskazówka: Na rysunkach 4a i 4b pokazano przykładowe błędy wykryte przez kompilator. Oba błędy są w szóstym wierszu programu, ale na rysunku 4b kompilator podświetlił wiersz siódmy. W tym przypadku bowiem brakuje średnika na końcu poprzedniego wiersza.

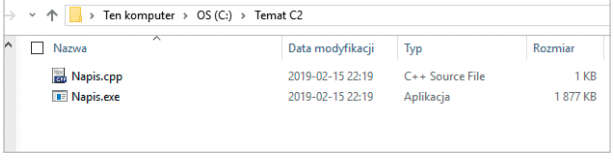


Rys. 4a. *Okno środowiska programistycznego języka C++ z programem źródłowym – kompilator wykrył błąd w programie*



Rys. 4b. *Okno środowiska programistycznego języka C++ z programem źródłowym – kompilator wykrył błąd w programie*

Po skompilowaniu programu źródłowego zapisanego w pliku *Napis.cpp* zostanie utworzony plik wykonywalny *Napis.exe* (aplikacja) o takiej samej nazwie własnej (*Napis*), ale z rozszerzeniem *exe* (rys. 5.). Aby uruchomić program po zamknięciu okna środowiska programistycznego, wystarczy kliknąć dwukrotnie nazwę lub ikonę *Napis.exe*. Program uruchomi się w oknie pokazanym na rysunku 3.



Rys. 5. *W oknie Eksploratora plików widać dwa pliki – źródłowy i wykonywalny*

2. Stosowanie zmiennych

Chcemy obliczyć iloczyn dwóch liczb całkowitych. Jak napisać program w języku C++ umożliwiający wprowadzenie dwóch liczb całkowitych z klawiatury, obliczający ich iloczyn i wyprowadzający wynik obliczeń na ekran?

Program, który utworzyliśmy w ćwiczeniach 1. i 2., wyświetlał na ekranie napis – nie używaliśmy w nim zmiennych i nie wykonywaliśmy obliczeń. Struktura kolejnych programów będzie bardziej rozbudowana, m.in. dodamy **deklarację zmiennych**.

2.1. Deklarowanie zmiennych

Komórka pamięci K

Część pamięci operacyjnej komputera (pamięci RAM). Pamięć operacyjna jest podzielona logicznie na komórki pamięci. Każda komórka pamięci ma swój unikatowy adres. W komórkach pamięci mogą być przechowywane dane reprezentujące instrukcje procesora lub dane dla tych instrukcji.

Typ zmiennej T

Określa rodzaj danych przechowywanych w zmiennej – mogą to być np. liczby, znaki, wartości logiczne (**typy proste**) lub teksty, tablice, rekordy i obiekty (**typy złożone**).

W wielu językach programowania, w tym w języku C++, zmienne przed użyciem należy **zadeklarować**. Deklaruje się zmienne, w których będą przechowywane dane wejściowe dla programu, wyniki działania programu, a także dane pomocnicze niezbędne do wykonania obliczeń. W deklaracji zmiennej podaje się jej **typ** oraz nazwę. Na tym etapie programowania będziemy korzystać wyłącznie z **prostych typów danych**.

Wykorzystywanym w programie zmiennym typu prostego przyporządkowane są określone **komórki pamięci**.

Dla każdej zadeklarowanej zmiennej rezerwowany jest fragment pamięci o określonym adresie i wielkości odpowiadającej danemu typowi danych. Na przykład dla zmiennych typu całkowitego `int` to cztery bajty lub osiem bajtów (zależnie od używanego kompilatora), dla zmiennych typu rzeczywistego `float` – cztery bajty, a wartość zmiennej znakowej `char` pamiętana jest w jednym bajcie lub w dwóch bajtach.

Użycie w programie niezadeklarowanej zmiennej zostanie zgłoszone podczas kompilacji jako błąd (program nie zostanie skompilowany).

Języki programowania oferują różne typy danych, które dobieramy zależnie od zadania, np. liczby całkowite, gdy mamy obliczyć sumę liczb całkowitych, lub znaki, gdy mamy sprawdzić, czy wprowadzona litera jest samogłoską.

Deklarowanie zmiennych	`opis_typu lista_zmiennych;`	C++

Przykład 1. Deklarowanie zmiennych w języku C++

Deklarowanie zmiennych typu całkowitego (`int`)

```
int liczba;
int a, b, Suma;
int a, b, iloczyn;
int liczba_sztuk, dlugosc;
```

Deklarowanie zmiennych typu rzeczywistego (`float`)

```
float minimum;
float a, b, c, suma, srednia;
float Cena_jedn, Dlugosc;
```

Zasady dotyczące nazw zmiennych w języku C++

1. Wielkie i małe litery w nazwach traktowane są odmiennie (np. `suma` i `Suma` oznaczać będą różne zmienne). Pisząc program w języku C++, należy zwracać uwagę na poprawne używanie małych i wielkich liter.
2. W nazwach zmiennych można stosować litery (bez polskich znaków diakrytycznych), cyfry i znak podkreślenia. Nazwa nie może zaczynać się od cyfry.
3. W nazwach zmiennych nie wolno stosować spacji.

Uwaga

W nazwach plików, w których zapisujemy programy tworzone w języku C++, nie będziemy stosować polskich znaków diakrytycznych i spacji (w nazwach wieloczłonowych użyjemy znaku podkreślenia).

W kilku kolejnych ćwiczeniach napiszemy program obliczający iloczyn dwóch liczb całkowitych wprowadzanych z klawiatury. Zaczniemy od zadeklarowania zmiennych.

Ćwiczenie 3. Zapisujemy dane i wyniki do zadania

Wzorując się na opisie podanym w przykładzie 6. z tematu C1, zapisz w zeszycie przedmiotowym dane i wyniki do zadania: *Napisz w wybranym języku programowania program umożliwiający wprowadzenie z klawiatury dwóch liczb całkowitych, obliczający ich iloczyn i wyprowadzający wynik obliczeń na ekran. Przed wprowadzeniem pierwszej danej wyświetlaj napis „Podaj pierwsza liczbe", przed wprowadzaniem drugiej – „Podaj druga liczbe", a przed wyprowadzeniem wyniku – „Iloczyn wynosi:".*

Uwaga

Aby uniknąć problemów z poprawnym wyświetlaniem komunikatów, nie będziemy w nich używać polskich znaków diakrytycznych.

Ćwiczenie 4. Deklarujemy zmienne

1. Utwórz nowy plik źródłowy. Zgodnie z zapisaną specyfikacją zadeklaruj zmienne do obliczenia iloczynu dwóch liczb całkowitych.
2. Zapisz program w pliku pod nazwą *Iloczyn* i skompiluj. Jeśli kompilator wykrył błędy, popraw program, ponownie zapisz, skompiluj i uruchom.

2.2. Nadawanie zmiennym wartości

Aby nadać zmiennej wartość, można wprowadzić wartość ze **standardowego wejścia** (zwykle klawiatury), stosując polecenie `cin >>`, lub zastosować **instrukcję przypisania**.

W języku C++ `cin` jest **obiektem** reprezentującym standardowe wejście programu. **Operator** >> oznacza wprowadzenie danych do odpowiedniej zmiennej, podanej po prawej stronie. Jeżeli chcemy wprowadzić dane do kilku zmiennych, należy operatora >> użyć wielokrotnie.

Wprowadzanie danych z klawiatury	`cin >> nazwa_zmiennej;`	C++

Przykład 2. Wprowadzanie danych

```
cin >> dlugosc;
cin >> a >> b;
cin >> Liczba1 >> Liczba2 >> Liczba3;
```

W **instrukcji przypisania** zmiennej podanej po lewej stronie instrukcji zostanie przypisana obliczona przez komputer wartość wyrażenia znajdującego się po prawej stronie instrukcji.

Instrukcja przypisania	`zmienna = wyrażenie;`	C++

Wyrażeniem mogą być m.in. konkretna wartość, zmienna, wyrażenie algebraiczne.

Przykład 3. Stosowanie instrukcji przypisania

```
Rok = 2019;
a = 7.58;
x = y;
Obwod = 2 * a + 2 * b;
```

Uwaga: W języku C++ operacji przypisania można użyć wielokrotnie w ramach jednej instrukcji, np. zapis: `a = b = c = 150;` oznacza, że wartość 150 zostanie przypisana najpierw zmiennej `c`, potem wartość zmiennej `c` zostanie przypisana zmiennej `b`, a na koniec wartość zmiennej `b` zostanie przypisana zmiennej `a`. Po wykonaniu instrukcji zmienne `a`, `b` i `c` będą miały wartość 150.

Ćwiczenie 5. Wprowadzamy dane z klawiatury

1. Otwórz plik *Iloczyn* zapisany w ćwiczeniu 4.
2. Dodaj polecenie wprowadzania danych z klawiatury, zgodnie z programem pokazanym na rysunku 6.
3. Zapisz plik pod tą samą nazwą i skompiluj program. Jeśli kompilator wykrył błędy, popraw je, ponownie zapisz plik i skompiluj program.

[*] Iloczyn.cpp

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int a, b, iloczyn;
6
7      cin >> a;
8      cin >> b;
9
10     return 0;
11 }
```

Rys. 6. *Program umożliwiający wprowadzenie dwóch liczb z klawiatury – ćwiczenie 5.*

2.3. Wykonywanie obliczeń

Aby zapisać w programie obliczenia, możemy skorzystać z instrukcji przypisania.

W języku C++ stosujemy następujące podstawowe **operatory arytmetyczne**: `+` (dodawania), `-` (odejmowania), `*` (mnożenia), `/` (dzielenia) oraz `%` (reszty z dzielenia). Operator `/` oznacza dzielenie, w zależności od typu danych – całkowite (z obcięciem części ułamkowej) lub zmiennoprzecinkowe (z zachowaniem części ułamkowej), a operator `%` – obliczenie reszty z dzielenia dwóch liczb całkowitych.

Operator	Działanie	Przykład instrukcji przypisania	Wynik działania dla danych: a = 11 i b = 4
+	dodawanie	`suma = a + b;`	11 + 4 = 15
–	odejmowanie	`roznica = a - b;`	11 – 4 = 7
*	mnożenie	`iloczyn = a * b;`	11 * 4 = 44
/	dzielenie w zależności od typu danych: • całkowite (z obcięciem części ułamkowej) – dla wartości typu całkowitego	`iloraz = a / b;`	11 / 4 = 2
	• zmiennoprzecinkowe (z zachowaniem części ułamkowej) – jeśli przynajmniej jedna liczba jest wartością typu rzeczywistego	`iloraz = a / b;`	11 / 4 = 2.75
%	obliczenie reszty z dzielenia dwóch liczb całkowitych	`reszta = a % b;`	11 % 4 = 3

Tabela 1. *Podstawowe operatory arytmetyczne w języku C++*

Ćwiczenie 6. Stosujemy instrukcję przypisania do zapisania obliczeń

1. Otwórz plik *Iloczyn* zapisany w ćwiczeniu 5.
2. Dodaj obliczanie iloczynu tak jak pokazano na rysunku 7.
3. Zapisz plik pod tą samą nazwą. Skompiluj i uruchom program. Dlaczego nie widzisz wyniku obliczeń na ekranie? Uzasadnij odpowiedź.

[*] Iloczyn.cpp

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int a, b, iloczyn;
6
7      cin >> a;
8      cin >> b;
9      iloczyn = a * b;
10
11     return 0;
12 }
```

Rys. 7. *Program obliczający iloczyn dwóch liczb wprowadzanych z klawiatury – ćwiczenie 6.*

3. Wyprowadzanie komunikatów i wyników na ekran monitora

W ćwiczeniach 1. i 2. wyprowadzaliśmy komunikaty na ekran monitora, używając polecenia `cout <<`. Za pomocą tego polecenia można wyprowadzać również wyniki obliczeń.

Wyprowadzanie wyników i komunikatów na ekran monitora	`cout << wartość;`	C++

W języku C++ `cout` jest obiektem reprezentującym standardowe wyjście programu (zwykle okno systemu operacyjnego na ekranie monitora). **Operator** << oznacza wyprowadzenie wartości podanej po prawej stronie. Jeżeli chcemy wyprowadzić kilka wartości, należy użyć operatora << wielokrotnie.

Wartością może być zmienna, wyrażenie, a także napis. Napis umieszczany w programie, który ma zostać wyświetlony na ekranie monitora, musi być ujęty w górne cudzysłowy "".

Przykład 4. Wyprowadzanie komunikatów i wyników

```
cout << "Zaczynamy lekcje z programowania";
cout << P;
cout << a + b;
cout << "Obwod = " << 2 * a + 2 * b;
cout << "Obwod wynosi: " << obwod;
```

Umieszczenie w wierszu z instrukcją wyprowadzania danych zapisu `endl` lub znaków `"\n"` powoduje przejście kursora na początek następnego wiersza po wyświetleniu komunikatu lub wyniku, za którym umieszczono te znaki.

Polecenia:

```
cout << "Liczba dni wolnych" << endl;
cout << liczba;
```

lub

```
cout << "Liczba dni wolnych" << endl << liczba;
```

lub

```
cout << "Liczba dni wolnych" << "\n";
cout << liczba;
```

lub

```
cout << "Liczba dni wolnych\n" << liczba;
```

wyświetlą w taki sam sposób te same wyniki

```
Liczba dni wolnych
12
```

Uwaga: Jeśli wprowadzamy z klawiatury podwójny cudzysłów, edytor środowiska Dev-C++ od razu wypisuje obydwa górne cudzysłowy i ustawia pomiędzy nimi kursor: `""`. Podobnie dzieje się, gdy wprowadzamy otwierający nawias okrągły po nazwie funkcji, np. `main`: `int main()`, czy otwierający nawias klamrowy zaczynający blok.

Ćwiczenie 7. Wyprowadzamy napisy i wynik na ekran

1. Otwórz plik zapisany w ćwiczeniu 6.
2. Dodaj wyświetlanie komunikatów, podobnie jak pokazano na rysunku 8a.
3. Zapisz plik pod tą samą nazwą. Skompiluj i uruchom program.
4. Co sądzisz o programach pokazanych na rysunkach 8a i 8b? Wskaż kilka zasadniczych różnic.

[*] Iloczyn.cpp

```
#include <iostream>
using namespace std;
int main()
{
    int a, b, iloczyn;

    cout << "podaj pierwsza liczbe: ";
    cin >> a;
    cout << "podaj druga liczbe: ";
    cin >> b;
    iloczyn = a * b;
    cout <<"Iloczyn wynosi: "<< iloczyn;

    return 0;
}
```

DOBRZE

Dobra rada

Mimo że kompilator pomija dodatkowe spacje i wcięcia, kod programu powinien być napisany czytelnie, przejrzyście i jednolicie, (rys. 8a). Innej osobie, trudniej jest pracować dalej nad programem napisanym w złym stylu (rys. 8b).

Rys. 8a. *Program napisany czytelnie i przejrzyście – ćwiczenie 7.*

[*] Iloczyn.cpp

```
#include <iostream>
using namespace std;
int main()
{int a , b,iloczyn ;
cout << "podaj pierwsza liczbe: " ;cin>> a;
    cout << "podaj druga liczbe: ";
cin >>b;iloczyn =a * b;cout <<"Iloczyn wynosi: "<<iloczyn;
        return 0;
}
```

ŹLE

Rys. 8b. *Program mniej czytelny i przejrzysty – ćwiczenie 7.*

Ćwiczenie 8. Testujemy działanie programu obliczającego średnią dla danych różnego typu

1. Do programu zapisanego w ćwiczeniu 7. dodaj obliczanie średniej arytmetycznej liczb *a* i *b*. Zadeklaruj zmienną *srednia* typu całkowitego **`int`**. Wynik wyprowadź na ekran. Dodaj odpowiednie komunikaty.
2. Zapisz program w pliku pod nazwą *Srednia_c*. Skompiluj go i uruchom. Przetestuj program dla par liczb (wartości zmiennych *a* i *b*): (2; 3), (14; 8); (7; 6).
3. Zmień typ wszystkich danych na **`float`**. Zapisz program pod nazwą *Srednia_r*, skompiluj i uruchom. Przetestuj program dla tych samych danych. Porównaj otrzymane wyniki z wynikami działania programu *Srednia_c*. Wyjaśnij, dlaczego dla tych samych danych otrzymujemy różne wyniki. Dodatkowo przetestuj program *Srednia_r* dla liczb: (342,3; 25,7) (3763,82; 109,87).

Uwaga

W większości języków programowania, w tym w języku C++, część dziesiętną liczby zapisujemy po kropce, a nie po przecinku, jak na matematyce. W ten sam sposób liczby dziesiętne są również wyświetlane na ekranie.

Wskazówka: Pamiętaj, aby liczby dziesiętne wprowadzać z kropką, a nie z przecinkiem.

4. Zapisywanie rozwiązania problemu w języku C++

Chcesz przygotować tradycyjną sałatkę jarzynową na imprezę urodzinową. W Internecie są przepisy, ale w proporcjach dla dwóch, pięciu czy dziesięciu osób. Nie możesz znaleźć przepisu dla trzynastu osób – uczestników spotkania. Jak napisać program w języku C++, który obliczy, ile potrzebujemy danego składnika sałatki i wyświetli wynik na ekranie?

Przykład 5. Sformułowanie zadania, opis specyfikacji i projekt rozwiązania

Zadanie: Napisz program obliczający, ile potrzeba danego składnika sałatki dla *o2* osób, gdy znasz ilość składnika dla *o1* osób.

Dane: liczba całkowita $o1 > 0$ oznaczająca liczbę osób uwzględnioną w przepisie, liczba rzeczywista $s1 > 0$, określająca ilość składnika dla *o1* osób, liczba całkowita $o2 > 0$ oznaczająca liczbę uczestników spotkania.

Wynik: liczba rzeczywista $s2 > 0$ oznaczająca ilość składnika dla *o2* osób.

W rozwiązaniu skorzystamy z zależności:
jeśli

s1 ⟶ dla *o1* osób
s2 ⟶ dla *o2* osób

to

$$s2 = \frac{s1 \cdot o2}{o1}$$

Aby zapisać te obliczenia w języku C++, zastosujemy instrukcję przypisania:

```
s2 = (s1 * o2) / o1;
```

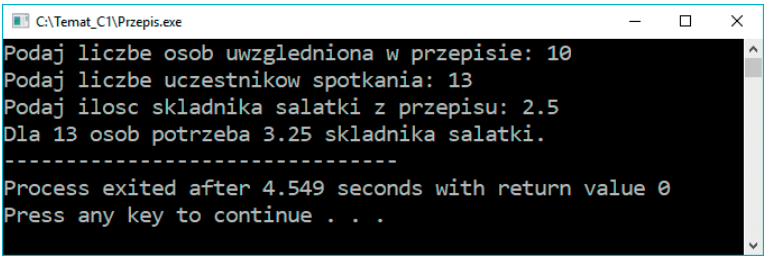
Uwaga: Z matematycznego punktu widzenia nawiasy są zbędne – umieściliśmy je w programie dla zwiększenia czytelności.

Ćwiczenie 9. Realizujemy algorytm obliczania ilości składnika sałatki w języku C++

Uwaga

Na razie w programach nie sprawdzamy poprawności danych wprowadzanych przez użytkownika. Zakładamy, że użytkownik podaje poprawne dane. Problem poprawności danych omówimy w temacie C4.

1. Utwórz program realizujący zadanie sformułowane w przykładzie 5. Zadeklaruj zmienne zgodnie ze specyfikacją zadania. Dodaj przed wprowadzaniem zmiennych komunikaty. Dodaj również komunikat przed wyprowadzeniem wyniku. Komunikaty i wynik wyprowadź na ekran monitora tak jak pokazano na rysunku 9.
2. Zapisz program w pliku pod nazwą *Przepis*. Skompiluj i uruchom program. Przetestuj go dla następujących trójek liczb (wartości zmiennych *o1, o2*, *s1*): (10; 13; 2,5), (3; 12; 0,45), (15; 4; 3,20).
3. Przetestuj program dla trzech innych trójek liczb.



Rys. 9. *Wynik działania programu – ćwiczenie 9.*

Wskazówki:

- Jeśli musimy zadeklarować zmienne różnych typów, warto dla czytelności programu zapisać deklaracje w dwóch kolejnych wierszach, np.:

```
int o1, o2;
float s1, s2;
```

- Aby wyświetlić wyniki tak jak pokazano na rysunku 9., należy wpisać instrukcję:

```
cout << "Dla " << o2 << " osob potrzeba " << s2
<< " skladnika salatki.";
```

Zauważmy, że program utworzony w ćwiczeniu 9. oblicza ilość tylko jednego składnika sałatki. Jeśli chcemy obliczyć ilość kolejnego składnika, musimy ponownie uruchomić program. W temacie C5 zmodyfikujemy ten program tak, aby obliczał i wyświetlał na ekranie kolejno ilości wszystkich składników.

Warto zapamiętać

- Aby utworzyć program w języku C++, piszemy kod źródłowy w specjalnym edytorze wbudowanym w środowisko programistyczne, zapisujemy program w pliku i kompilujemy. Jeśli kompilacja przebiegła prawidłowo, uruchamiamy program. Wynik działania programu pokaże się w osobnym oknie i zostanie utworzony plik wykonywalny z rozszerzeniem *exe*.
- Funkcja główna `main()`, rozpoczynająca działanie programu, musi wystąpić w każdym programie w języku C++. Inne elementy są opcjonalne.
- W języku C++ każda zmienna używana w programie musi zostać zadeklarowana przed pierwszym użyciem poprzez określenie jej typu i nazwy.
- Wartość zmiennej możemy wprowadzić z klawiatury (stosując polecenie `cin >> zmienna`) lub nadać zmiennej wartość za pomocą instrukcji przypisania.
- Obliczenia zapisujemy, korzystając z instrukcji przypisania. Zmiennej po lewej stronie znaku = zostanie przypisana wartość wyrażenia podanego po prawej stronie.
- Komunikaty i wyniki wyprowadzamy na ekran monitora, stosując polecenie `cout <<`.

Pytania i polecenia

1. Przedstaw i omów na konkretnym przykładzie etapy tworzenia programu komputerowego.
2. Jaka jest ogólna struktura programu w języku C++?
3. Czym jest komórka pamięci?

4. Na czym polega deklaracja zmiennych w programie komputerowym? Wyjaśnij sposób deklarowania zmiennych w języku C++.
5. Przedstaw na przykładzie sposób wprowadzania danych z klawiatury.
6. Do czego służy instrukcja przypisania?
7. Jaka jest rola zapisów `endl` i `"\n"` umieszczonych w programie?
8. Wyjaśnij zapisy:
   a. `int liczba1, liczba2;`
   b. `float iloraz;`
   c. `cin >> liczba1;`
   d. `iloraz = liczba1 / liczba2;`
   e. `cout << "Iloraz wynosi: " << iloraz << endl;`
9. Wskaż nieprawidłowo zapisane instrukcje. Wyjaśnij, na czym polegają błędy:
   a. `cin >> 78;`
   b. `cin >> droga >> czas;`
   c. `cout >> "Podaj liczbę: ";`
   d. `cout << "Suma wynosi: " << a + b;`

Zadania

Uwagi:
- W edytorze języka C++ można kopiować (**Edycja**/**Kopiuj**) i wklejać (**Edycja**/**Wklej**) fragmenty kodu programu – w tym samym dokumencie i pomiędzy dokumentami (tak jak w edytorze tekstu). Ta możliwość jest przydatna, gdy ponownie wykorzystujemy te same lub podobne fragmenty kodu.
- Staraj się dodawać do programu komunikaty (również komentarze) oraz pisać przejrzyste programy.
- Każdy program należy skompilować, uruchomić i przetestować dla różnych danych, nawet jeśli w zadaniu nie ma takiego polecenia.

1. Napisz specyfikację zadania i program obliczający pole powierzchni prostokąta dla danych długości boków *a* i *b*, wprowadzanych z klawiatury. Po uruchomieniu programu na ekranie powinny pojawiać się w kolejnych wierszach komunikaty: „Podaj dlugosc boku a:", „Podaj dlugosc boku b:". Po podaniu danych w trzecim wierszu powinny wyświetlić się: napis „Pole prostokąta wynosi" oraz wartość pola. Zapisz program w pliku pod nazwą *Pole.*
2. Napisz specyfikację zadania i program realizujący algorytm obliczania kwadratu i sześcianu wprowadzanej z klawiatury liczby rzeczywistej. Zapisz program w pliku pod nazwą *Potegi*.
3. Napisz specyfikację i program do zadania: *Oblicz drogę* S *przebytą w czasie* t *przez pojazd poruszający się ze średnią prędkością* v. Zapisz program w pliku pod nazwą *Droga*.
   Wskazówka: Zadanie wykonaj na podstawie specyfikacji zapisanej w ćwiczeniu 2., punkt 1. z tematu C1.
4. Przepisz program pokazany na rysunku 10. Zapisz go w pliku pod nazwą *Test* i skompiluj. Po wprowadzeniu z klawiatury liczby program powinien wyświetlić na ekranie liczbę o 2 większą od wprowadzonej. Jakie błędy pojawiły się w zapisie programu? Zmodyfikuj program, aby działał poprawnie. Zapisz ponownie plik, skompiluj program, uruchom i sprawdź jego działanie.

[*] TC1_z4.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int liczba1;
7      cin >> liczba1
8      cout << liczba2
9      liczba2 = liczba1 + 2;
10     return 0;
11 }
```

Rys. 10. *Błędny program – zadanie 4.*

5. Napisz specyfikację i program do zadania: *Dane są: suma długości przekątnych rombu i długość jednego boku. Oblicz długości przekątnych rombu*. Zapisz program w pliku pod nazwą *Romb*.
6. Napisz specyfikację i program do zadania: *Jaki procent liczby* a *stanowi liczba* b*?* Zapisz program w pliku pod nazwą *Procenty*.
7. Napisz program obliczający wysokość trójkąta, gdy podane są jego pole i podstawa. Zapisz program w pliku pod nazwą *Trojkat*.
8. Napisz program obliczający, ile litrów wody spadło na plac o powierzchni P m^2, jeśli pokryła go warstwa wody o grubości d milimetrów. Zapisz program w pliku pod nazwą *Deszcz*.

Dla zainteresowanych

9. Napisz specyfikację zadania i opisz rozwiązanie: *Dany jest średni wiek trzech osób: S. Najmłodsza osoba ma* x *lat, najstarsza:* y. *Ile lat ma trzecia osoba? S, x, y to liczby naturalne różne od zera*. Napisz program na podstawie zapisanej specyfikacji. Przetestuj program dla następujących wartości zmiennych (S, x, y): (15, 10, 20), (27, 20, 40), (24, 14, 33). Dodaj kilka swoich propozycji danych. Jakie powinny być wartości zmiennych: S oraz x i y, aby otrzymany wynik był sensowny? Uwzględnij te warunki w opisie algorytmu.
10. Dane są suma i iloczyn dwóch liczb całkowitych. Napisz specyfikację zadania i program znajdujący te liczby. Zapisz program w pliku pod nazwą *Liczby*.
11. Opisz wymyślony samodzielnie problem podobnie jak w punkcie 4. tego tematu. Ułóż zadanie, zapisz specyfikację i przygotuj rozwiązanie podobnie jak w przykładzie 5. Na koniec napisz program realizujący to zadanie. Zapisz program w pliku pod nazwą *Problem*.