

Programowanie I

Wykład 1

dr inż. Rafał Brociek

Wydział Matematyki Stosowanej
Politechnika Śląska



04.10.2021

Karta przedmiotu - zasady zaliczenia

- Dwa kolokwia praktyczne na laboratoriach 2×20 pkt.
- Egzamin na wykładzie 20 pkt.
- Projekt zaliczeniowy 25 pkt.
- Aktywność na zajęciach, zadania domowe 15 pkt.

- Jerzy Grębosz: *Opus Magnum C++11*, tom 1,2,3, Helion.
- Stephen Prata: *Język C++. Szkoła programowania. Wydanie VI*, Helion.
- Bruce Eckel: *Thinking in C++*, Helion.

Na laboratoriach używać będziemy **Microsoft Visual Studio 2017 (2019)**. Inne możliwe środowiska programistyczne: *Code::Blocks*, *NetBeans*, *KDevelop*, etc.
Standard języka: **C++11** (znany również jako *C++0x*).

Preprocesor - program mający za zadanie przetworzenie kodu źródłowego przy użyciu dyrektyw preprocesora. Następnie tak przetworzony kod jest poddawany analizie składniowej, kompilacji.

Kompilator - program służący do tłumaczenia **kodu źródłowego** napisanego w danym języku programowania (np. w C++) na język maszynowy (procesora). Często skompilowana wersja programu musi zostać jeszcze połączona z tak zwanymi *bibliotekami*.

Linkowanie (konsolidacja) - proces łączenia programu z bibliotekami (za proces ten odpowiada **linker**).

Biblioteka - zbiór konstrukcji programistycznych (np. funkcji).

Kod źródłowy - program napisany w danym języku programowania (np. C++) w postaci zrozumiałej dla człowieka.

Kod maszynowy - program wyrażony w postaci zrozumiałej dla maszyny (w postaci binarnej).

Wykonanie programu zapisanego w postaci zrozumiałej dla maszyny, w bardzo uproszczonym modelu, polega na umieszczeniu tego programu w pamięci operacyjnej komputera oraz wskazaniu procesorowi jego pierwszej instrukcji.

Podział języków programowania

- ❶ niskopoziomowe – stosuje się proste wyrażenia symboliczne, które odpowiadają zestawowi rozkazów maszynowych. Przykładami takich języków są assembly, w których operuje się bezpośrednio na rejestrach procesora. Są blisko powiązane ze sprzętem. Brak w tych językach takich konstrukcji jak np. pętla.
- ❷ wysokopoziomowe – składnia i słowa kluczowe są zrozumiałe dla człowieka, posiadają abstrakcje programistyczne jak np. pętle. Nie są bezpośrednio zrozumiałe dla komputera. Możemy podzielić je na dwie grupy:
 - interpretowane – wymagają interpretacji (brak kompilacji), są wczytywane, analizowane i wykonywane przez interpreter języka dopiero podczas uruchomienia. Przykłady: *JavaScript, Python, PHP*.
 - kompilowane – kod źródłowy wymaga kompilacji do kodu maszynowego (po przez kompilator), dopiero wówczas możliwe jest uruchomienie programu. Przykłady: *C, C++, C#, Java*.

Zacznijmy od definicji słowa *paradygmat*. Zgodnie ze słownikiem języka polskiego (źródło: <https://sjp.pwn.pl/slowniki/paradygmat.html>):

przyjęty sposób widzenia w danej dziedzinie, doktrynie itp.

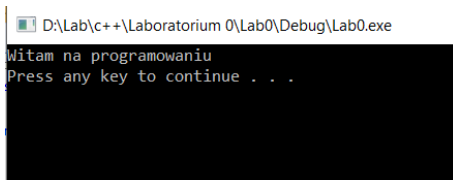
Paradygmat programowania - zbiór reguł, mechanizmów, zasad jakich programista używa do napisania programu oraz sposób w jaki ów program będzie wykonywany przez komputer.

Paradygmaty programowania

- 1 programowanie proceduralne – pisany program jest dzielony na procedury (funkcje) - fragmenty kodu wykonujące ściśle określone operacje. Następnie poszczególne procedury „składane są” w finalny program i wykonywane w określonej kolejności.
- 2 programowanie obiektowe – program definiuje się za pomocą obiektów, które to obiekty posiadają stan (dane obiektu) oraz zachowanie (metody obiektu). Językami wspierającymi programowanie obiektowe są np. C# oraz Java.
- 3 programowanie funkcyjne – kod programu definiuje się przy użyciu funkcji. Często pętle są zastępowane przez rekurencję. Wynik otrzymywany jest na podstawie złożenia funkcji. Czysty język funkcyjny nie posiada zmiennych. Przykładami języków funkcyjnych są LISP, ML.
Język C++ posiada elementy zarówno programowania proceduralnego, jak i obiektowego.

Pierwszy program

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     cout << "Witam na programowaniu\n";
7     system("pause");
8 }
```



The screenshot shows a Windows command prompt window with the title bar "D:\Lab\c++\Laboratorium 0\Lab0\Debug\Lab0.exe". The window contains the text "Witam na programowaniu" followed by "Press any key to continue . . ." on the next line. The background of the command prompt is black, and the text is white.

Rysunek: Wynik działania programu

Pierwszy program

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     cout << "Witam na programowaniu\n";
7     system("pause");
8 }
```

- linia 1 - dołączenie pliku nagłówkowego (obsługa strumienia wejścia/wyjścia),
- linia 2 - użycie standardowej przestrzeni nazw,
- linia 4 - funkcja `main`, to od niej rozpoczyna się program,
- linia 6 - wypisanie treści na standardowe wyjście,
- linia 7 - zatrzymanie okna konsoli.

- *.sln – plik rozwiązania (sln – skrót od solution). Przechowuje globalne informacje o rozwiązaniu. Rozwiązanie może zawierać jeden lub więcej projektów (również wspierających różne technologie i języki programowania).
- *.vcxproj – plik projektu, zawiera dane charakterystyczne dla danego projektu.
- *.cpp – plik z kodem źródłowym.

Komentarze

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     // wypisywanie na ekran konsoli
7     cout << "Komentarze\n" // komentarz
8
9     /* komentarz wieloliniowy
10    linia 1
11    linia 2
12    ...
13    */
14 }
```

Całkowitoliczbowe

typ	szerokość	najczęściej spotykany zakres wartości
int	4 bajty	-2147483648 do 2147483647
unsigned int	4 bajty	0 do 4294967295
short int	2 bajty	-32768 do 32767
long int	4 bajty	-2147483648 do 2147483647
long long int	8 bajtów	-9223372036854775808 do 9223372036854775807

Znakowe

- char - pojedynczy znak ASCII (1 bajt)

Zmiennoprzecinkowe

typ	szerokość	najczęściej spotykany zakres wartości
float	4 bajty	1.17549e-38 do 3.40282e+38
double	8 bajtów	2.22507e-308 do 1.79769e+308

Logiczne

- `bool` - wartość `true` albo `false` (1 bajt)

Definiowanie zmiennych

Nazwa zmiennej składa się z ciągu liter języka angielskiego, cyfr oraz znaków podkreślenia, przy czym nie może zaczynać się od cyfry. Rozróżniane są małe i wielkie litery, nie można używać słów kluczowych.

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int wiek = 24;
6     int rozmiar{ 31 };
7     int zm_1, zm_2; //tylko deklaracja
8     float temperatura = 27.45;
9     double pomiar = -1.458964;
10    bool czyKoniecGry = true;
11    char litera = 'a';
12    cout << "Zmienna wiek ma wartosc " << wiek << endl;
13    cout << "Zmienna rozmiar ma wartosc " << rozmiar << endl;
14    // cout << "Zmienna zmienna_1 ma wartosc " << zm_1 << endl;
15    cout << "Zmienna czyKoniecGry ma wartosc " << czyKoniecGry << ↵
        endl;
16    cout << "Zmienna litera ma wartosc " << litera << endl;
17 }
```


Tablica kodów ASCII

ASCII (American Standard Code for Information Interchange) - siedmiobitowy system kodowania znaków. Przyporządkowuje liczbom całkowitym z zakresu 0 – 127 litery alfabetu angielskiego, cyfry, polecenia sterujące oraz inne symbole.

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     char plus = 43;
6     char litera_A = 65;
7     cout << "plus: " << plus << endl;
8     cout << "litera_A: " << litera_A << endl;
9     // kodowanie cp852
10    char znak1 = 180;
11    char znak2 = 165;
12    cout << "znak1: " << znak1 << endl;
13    cout << "znak2: " << znak2 << endl;
14
15 }
```

Operatory

Operatory arytmetyczne

- + dodawania $a+b$
- - odejmowania $a-b$
- * mnożenia $a*b$
- / dzielenia a/b
- % reszty z dzielenia (tzw. modulo) $a\%b$

Operator przypisania

- = przypisanie $a=3$, $a=b$, $a=a+b$

Złożone operatory przypisania

- +=, -=, *=, /=, %= ($a += 5$ jest równoważne $a = a+5$)

Operator

```
1 int a = 3;
2 a += 5; // a = 8
3 a /= 2; // a = 4
4
5
6 double b = 1/2; // b = 0
7 double c = 1.0/2.0; // c = 0.5
8 int d = 10/3; // d = 3
9
10 int reszta = 10 % a; // reszta = 2
11
12 int duza = 2100000000;
13 duza = duza*2; // duza = -94967296
14
15 long long int duza1 = 2100000000;
16 duza1 = duza1*2; // duza1 = 4200000000
```

Operatory inkrementacji i dekrementacji

- postinkrementacja $a++$ - wartość a zostaje zwiększona o 1 po wykonaniu operacji
- preinkrementacja $++a$ - wartość a zostaje zwiększona o 1 przed wykonaniem operacji
- postdekrementacja $a--$ - wartość a zostaje zmniejszona o 1 po wykonaniu operacji
- predekrementacja $--a$ - wartość a zostaje zmniejszona o 1 przed wykonaniem operacji

Operator

```
1 int a = 1;
2 cout << a++ << endl; // 1
3 cout << a << endl; // 2
4 cout << ++a << endl; // 3
5 cout << a << endl; // 3
6 cout << a-- << endl; // 3
7 cout << a << endl; // 2
8 cout << --a << endl; // 1
9 cout << a << endl; // 1
```

Operatory relacyjne

- `==` równe `a == b`
- `!=` nierówne `a != b`
- `>` większy `a > b`
- `>=` większy lub równy `a >= b`
- `<` mniejszy `a < b`
- `<=` mniejszy lub równy `a <= b`

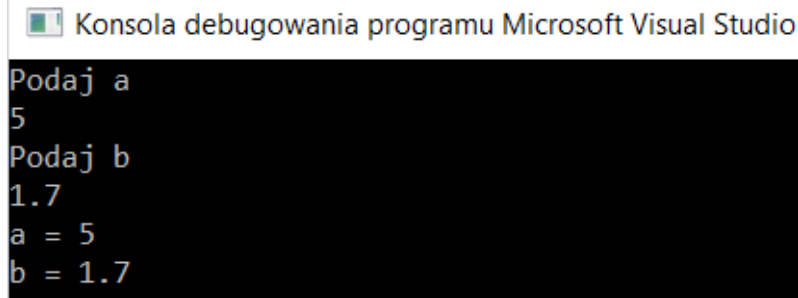
Operatory logiczne

- `||` suma logiczna (alternatywa)
- `&&` iloczyn logiczny (koniunkcja)
- `!` negacja logiczna

Operatory

```
1 int a = 5, b = 7, c = 10;
2 cout << (a < b) << endl;    // 1
3 cout << (b > a) << endl;    // 1
4 cout << ((a < b) || (c < b)) << endl; // 1
5 cout << ((a < b) && (c < b)) << endl; // 0
6 cout << boolalpha << (5 > 2) << endl; // true
7 cout << boolalpha << (4 < 1) << endl; // false
```

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a;
7     double b;
8     // wczytanie danych
9     cout << "Podaj a " << endl;
10    cin >> a;
11    cout << "Podaj b " << endl;
12    cin >> b;
13    // wypisanie danych
14    cout << "a = " << a << endl;
15    cout << "b = " << b << endl;
16 }
```

Konsola debugowania programu Microsoft Visual Studio

```
Podaj a
5
Podaj b
1.7
a = 5
b = 1.7
```

Rysunek: Wynik działania programu

- funkcja `main()` jest punktem wejściowym programu,
- w celu wprowadzania danych ze standardowego wejścia (klawiatury) oraz wyprowadzania danych na standardowe wyjście (ekran monitora) korzystamy z biblioteki `iostream` (obiekty `cin`, `cout`),
- do dołączania nagłówków biblioteki używamy dyrektywy preprocesora np. `#include<iostream>`,
- nazwy funkcji z biblioteki `iostream` zawarte są w przestrzeni nazw `std`,
- rozróżniamy różne typy danych: całkowitoliczbowe (np. `int`), zmiennoprzecinkowe (np. `double`), logiczne (`bool`),
- zmienne deklarujemy podając wcześniej ich typ,
- wyróżniamy następujące operatory: arytmetyczne (`+`, `-`, `*`, `/`, `%`), operator przypisania (`=`), relacyjne (np. `>`), logiczne (np. `&&`), operatory wprowadzania/wyprowadzania danych.

Zadanie 1.

```
1 int a = 3, b = 0, c = 0;
2 b = ++a * 3 + ++a;
3 // ile wynosi a oraz b?
4 cout << "a=" << a << " b=" << b << endl;
5 a = 3;
6 c = ++a * 3 + a++;
7 // ile wynosi a oraz c?
8 cout << "a=" << a << " c=" << c << endl;
```

Dziękuję za uwagę