Wstęp do programowania

Dr Anna Muranova

Semestr zimowy 2023/2024, UWM w Olsztynie

1. Zajęcie

---

1 Podstawy C++

1.1 Najprostsze programy

1.1.1 Pusty program

Najprostszy możliwy program w języku C++ wygląda tak:

```
int main() {}//najmniejszy mozliwy program w jezyku C++
```

Kod ten zawiera definicję funkcji o nazwie `main`, która nie pobiera żadnych argumentów ani nic nie robi.

- { i } klamry w języku C++ służą do grupowania. W tym przypadku wyznaczają początek i koniec treści właściwej funkcji.
- // dwa znajdujące się obok siebie ukośniki oznaczają początek komentarza sięgającego do końca wiersza. Komentarze są dla ludzi. Kompilator je ignoruje.

1.1.2 Program "Hello world!"

```
#include <iostream>

int main()
{
std::cout << "Hello world!\n";
return 0;
}
```

albo

```
#include <iostream>
using namespace std;

int main()
{
cout << "Hello world!\n";
return 0;
}
```

Uwagi:

- `using namespace std;` dzięki temu nazwy z `std` są widoczne bez przedrostka `std`.

- wiersz `#include <iostream>` nakazuje kompilatorowi dołączenie do programu deklaracji standardowych narzędzi wejścia i wyjścia dostępnych w bibliotece iostream. Bez niej wyrażenie:

  ```
  std::cout << "Hello world!\n";
  ```

  nie miałoby sensu.

- operator << (wyślij do) zapisuje swój drugi argument w pierwszym. W tym przypadku literał łańcuchowy "Hello world!\n" zostaje wysłany do standardowego strumienia wyjściowego `std::cout`.

- `return 0` – zwrócenie 0 w funkcji main oznacza, że program wykonał się pomyślnie. Standardowe trzeba zawsze to pisać na końcu main().

1.2 Typy danych

W języku C++ dostępnych jest kilka podstawowych typów danych, np.:

- `bool` – typ logiczny, którego zbiór wartości zawiera tylko prawdę i fałsz
- `char` – znak, np. 'a', 'z', '9'
- `int` – liczba całkowita, np. 1, 42, 1066
- `float` – liczba zmiennoprzecinkowa, np. 3.14, 0.453.
- `double` – liczba zmiennoprzecinkowa o podwójnej precyzji, np. 3.14 albo 299793.0
- `void` – służy do oznaczania braku informacji (używa się jako typ funkcji, kiedy funkcja nic nie zwraca).

  Deklaracja i inicjalizacja:

  ```
  int x;//deklaracja
  x = 2;//inicializacja
  ```

  albo

  ```
  int x=2;
  ```

1.3 Wczytywanie oraz wyświetlenie na konsoli:

`cout, printf, cin`

Przykład:

```
#include <iostream>
using namespace std;

int main()
{
    int a;
    cin >> a;
```

```
    cout << "a is " << a;
    printf("\na is %d", a);
    return 0;
}
```

Uwaga. Opcje dla printf: %c (char), %d (double lub integer), ...
\n – początek nowego wierszu.

1.4 Operatory arytmetyczne

Do wykonywania działań arytmetycznych na tych typach służą operatory arytmetyczne:

```
x+y // plus
+x // jednoargumentowy plus
x-y // minus
-x // jednoargumentowy minus
x/y // dzielenie
x*y // mnozenie
x%y // reszta z dzielenia (modulo) liczb calkowitych
```

1.5 Operatory porównawczy

```
x==y // rowny
x!=y // rozny
x<y // mniejszy niz
x>y // wiekszy niz
x<=y // mniejszy niz lub rowny
x>=y // wiekszy niz lub rowny
```

1.6 Skróty operatorów

Oprócz konwencjonalnych operatorów arytmetycznych i logicznych w języku C++ można także używać specjalnych operatorów do modyfikowania zmiennych:

```
x+=y // x = x+y
++x // inkrementacja: x = x+1
x++ // inkrementacja: x = x+1
x-=y // x = x-y
- -x // dekrementacja: x = x-1
x- - // dekrementacja: x = x-1
x*=y // skalowanie: x = x*y
x/=y // skalowanie: x = x/y
x%=y // x = x%y
```

1.7 Podstawowe operatory logiczny

i	&&	Iloczyn logiczny
lub	||	Suma logiczna
negacja	!	Zanegowanie wartości

1.8 Programowanie proceduralne

Zadecyduj, jakie chcesz mieć procedury, stosuj najlepszy algorytmy, jakie możesz znaleźć.

Typowym przykładem dobrego stylu jest funkcja licząca kwadrat liczby.

```
double square (double arg)
{
return arg*arg;
}
```

1.9 Operator warunkowy if

```
if (warunek)
    zmienna = wyrazenie1;
else
    zmienna = wyrazenie2;
```

lub

```
warunek ? wyrażenie1 : wyrażenie2
```

1.10 Operator switch

```
switch(wyrażenie) {
  case x:
    // kod
    break;
  case y:
    // kod
    break;
  default:
    // kod
}
```

Przyklad:

```
void write_day(int a)
{
     switch (a){
     case 1:
          cout << "Monday\n";
         break;
     case 2:
          cout << "Tuesday\n";
         break;
     case 3:
          cout << "Wednesday\n";
         break;
     case 4:
```

```
            cout<<"Thursday\n";
        break;
    case 5:
            cout << "Friday\n";
        break;
    case 6:
            cout << "Saturday\n";
        break;
    case 7:
            cout << "Sunday\n";
        break;
    default:
        cout<<"Error\n";
    }
}
```

2 Ćwiczenia

Ćwiczenie 1. Uruchom program "Hello World!".

Ćwiczenie 2. Spróbuj następne polecenia:

- Jak działa iloraz na rożnych typach danych?

```
int x = 1;
float y = 1.0f;
double z = 1;
cout << x / 3 << '\n';
cout << y / 3 << '\n';
cout << z / 3 << '\n';
```

- Co to jest precyzyjność?

```
cout << 0.3-3*0.1 << '\n';
```

- Porównaj typy double oraz float

```
double  pi_double = 3.14159265358979323846264338;
float     pi_float = 3.14159265358979323846264338f;

printf("3.14159265358979323846264338\n");
printf("double: %1.26lf\n", pi_double);
printf("float : %1.26f\n", pi_float);
```

Ćwiczenie 3. Uruchom następny program, zawierający funkcje:

```
#include <iostream>
using namespace std;

double square(double arg)
```

```
{
    return arg * arg;
}

void print_square(double x) // void - funkcja nie
                            przekazuje zadnej wartosci
{
    cout << "Kwadrat liczby " << x << " wynosi "
                    << square(x) << "\n";
}

int main()
{
    print_square(2);
    return 0;
}
```

Ćwiczenie 4. Zmień następny program (sprawdza, czy liczba jest parzysta) tak, aby liczbą trzeba było wpisywać z konsoli, a parzystość sprawdzała się i wypisywała się na konsol w osobnej funkcji.

```
#include <iostream>
using namespace std;

int main()
{
    int x;
    x = 3;

    if (x % 2 == 0)
        cout << "even";
    else
        cout << "odd";

    return 0;
}
```

inna możliwość dla sprawdzenia:

```
x % 2 == 0 ? cout << "even" : cout << "odd";
```

Ćwiczenie 5. Czym różnią się `x++` oraz `++x`?

```
int x1 = 5;
int x2 = 5;
int y1 = x1++;
int y2 = ++x2;
cout << "x1: " << x1;
cout << "\n x2 : " << x2;
cout << "\n y1 : " << y1;
cout << "\n y2 : " << y2;
```

Ćwiczenie 6. Napisz funkcję, która wyświetla na konsoli porą dnia według podanej godziny

- $5 - 11 \rightarrow$ morning
- $12 \rightarrow$ noon
- $13 - 17 \rightarrow$ afternoon
- $18 - 20 \rightarrow$ evening
- $21 - 23, 1 - 4 \rightarrow$ night
- $24 \rightarrow$ midnight

Co lepiej używać: switch czy if?

Ćwiczenie 7. Napisz funkcje "Calculator" która po podanym dwóm liczbom typu **double** i symbolu typu **char** $(*, +, -, :)$ zwraca iloczyn, sumy, różnicą albo ilora tych liczb.

Dodaj w main() możliwość wprowadzenie z klawiatury jako naprz. $2 + 3$.

3 Oprogramowanie

3.1 Visual Studio

Download page: `https://visualstudio.microsoft.com`

Installation: `https://learn.microsoft.com/en-us/cpp/build/vscpp-step-0-installation view=msvc-170`

Create project: `https://learn.microsoft.com/en-us/cpp/build/vscpp-step-1-create? view=msvc-170`

Build and run project: `https://learn.microsoft.com/en-us/cpp/build/vscpp-step-2-buil source=recommendations&view=msvc-170`

```
File -> New -> Project -> Console App -> C++
```

3.2 Dev-C++

Download page: `https://sourceforge.net/projects/orwelldevcpp/`

Wersja: TDM-GCC 4.9.2 32/64bit