Programowanie

Dariusz Wardecki, wyk. 1

Kontakt

- dward@fuw.edu.pl
- http://www.fuw.edu.pl/~dward/wdp
- ul. Hoża 69, pok. 114
- tel. 22 55 32 181

Zasady zaliczenia

- Wykład (2h/tydzień)
  - Egzamin pisemny
  - Test wyboru 15 pkt.
- Ćwiczenia (2h/tydzień)
  - 3 kolokwia po 15 pkt.
  - 3 zadania po 5 pkt.

Łącznie można zdobyć max 60 pkt.

Materiał na ćwiczenia

- Algorytmy.
- Podstawy programowania w C++
  - Funkcja `main()`, nagłówek i treść funkcji
  - Obsługa wej/wyj
  - Wyrażenia i zmienne
  - Pętle
- Programowanie prostych obliczeń
- Liczby pseudolosowe
- Formatowanie wydruków i drukowanie wzorów
- Tablice i sortowanie liczb
- Wskaźniki i referencje
- Wej/wyj z wykorzystaniem plików
- Złożone typy danych i ich zastosowanie

Plan wykładu

- Kod źródłowy w C++ i kompilator g++
- Proste typy danych
- Składnia języka C++ i przykłady algorytmów
- Operatory i wyrażenia, konwersje typów danych
- Pętle, instrukcja warunkowa i instrukcje sterujące w C++
- Wskaźniki i tablice, operacje na wskaźnikach, Referencje
- Operacje na łańcuchach (C-string)
- Dynamiczne rezerwowanie pamięci i zwalnianie pamięci
- Tablice wskaźników i argumenty programu
- Złożone typy danych i ich zastosowania

Dlaczego warto uczyć się programowania komputerów

- Żeby zobaczyć na czym to polega
- Jeżeli mamy nietypowe zadanie, to czasem szybciej jest napisać własny program, niż szukać gotowego
- Czasami musimy być pewni, że program działa w ściśle określony sposób
- Może to być interesujące
- Stanowiska pracy związane z programowaniem bywają dobrze płatne

Na czym polega programowanie?

- Określenie problemu (co chcemy osiągnąć).
- Wybranie algorytmu, czyli metody poszukiwania rozwiązania (jak będziemy to robić).
- Tworzenie kodu źródłowego (*ang. source code*), stanowiącego reprezentację wybranego algorytmu.
- Kompilowanie kodu źródłowego, czyli tworzenie kodu wykonywalnego (*ang. executable code*) gotowego do wykonania.
- Sprawdzanie i testowanie (*ang. debugging*) programu. Jeżeli znajdziemy błąd, wracamy do etapu tworzenia kodu źródłowego (lub nawet do etapu wyboru algorytmu)

Trochę historii

- język C - początek lat siedemdziesiątych, Dennis Ritchie
- język C++ - koniec lat osimdziesiątych, Bjarne Stroustrup

Kod źródłowy i kompilator

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

int main()
   {
      cout << "Hello World!!!" << endl;
      return 0;
   }
```

```
g++ -o nazwa nazwa.cc
```

```
./nazwa
```

Kod źródłowy w pliku `nazwa.cc`

Kompilator

Program

Logowanie na Primusa

Dom —`ssh`→ primus.okwf.fuw.edu.pl

putty

```
ssh login@tempac.fuw.edu.pl
ssh login@primus.okwf.fuw.edu.pl
```

Kompilator: `g++`

Edytor: `nano, pico`

Kod źródłowy

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

int main()
  {
    cout << ”Hello World!!!” << endl;
    return 0;
  }
```

Kod źródłowy

Polecenie preprocesora

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

int main()
  {
    cout << ”Hello World!!!” << endl;
    return 0;
  }
```

Kod źródłowy

Polecenie preprocesora

Przestrzeń nazw

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

int main()
  {
    cout << ”Hello World!!!” << endl;
    return 0;
  }
```

Kod źródłowy

Polecenie preprocesora

Przestrzeń nazw

Funkcja main

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

int main()
  {
    cout << "Hello World!!!" << endl;
    return 0;
  }
```

Kod źródłowy

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

int main()
  {
    cout << "Hello World!!!" << endl;
    return 0;
  }
```

Polecenie preprocesora

Przestrzeń nazw

Funkcja main

Instrukcje

Proste typy danych

Liczby całkowite

`int` 32 bitów

`char` 8 bitów

char służy do przechowywania znaków alfanumerycznych (wg kodowania ASCII)

Proste typy danych

Liczby całkowite - przykłady

```
char x = 65;
  char x = 0x41;
  char x = 0101;
  char x = 'A';
```

```
#include<iostream>

using namespace std;


int main(){
    char znak1 = '<';
    char znak2 = '<';
    char znak3;

    znak3 = znak2 + znak2;
    cout << znak3 << endl;

    return 0;

}
```

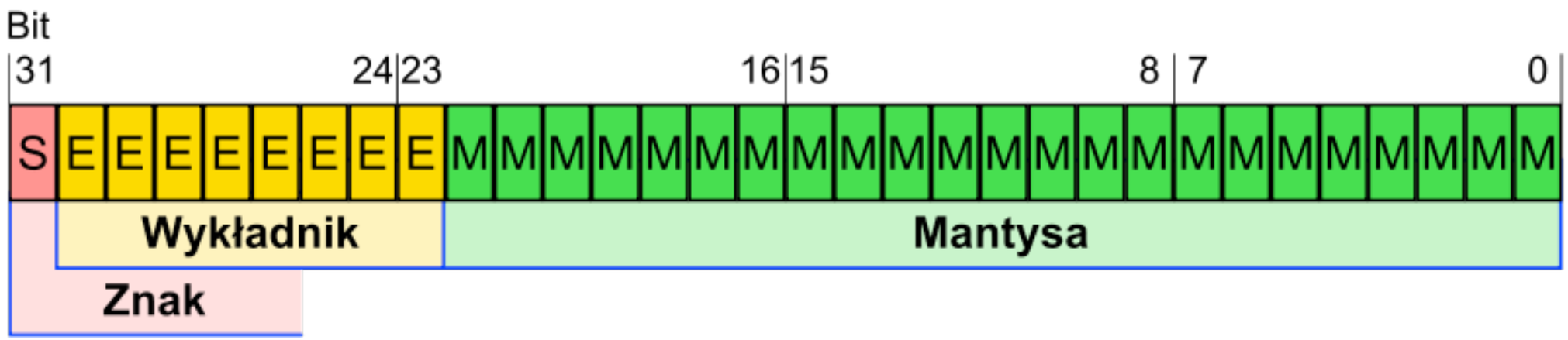
```
int a = 100;
unsigned int a = 30;
int z = 2.9999;
```

liczby całkowite dodatnie

zaokrąglenie w dół

Proste typy danych

Liczby zmiennoprzecinkowe



`float`	32 bity
`double`	64 bity

Proste typy danych

`short , long`

`short int`	16 bitów
`long int`	64 bity
`long double`	128 bitów

Wielkość typu zmiennej można sprawdzić za pomocą operatora `sizeof()`

Operacje wejścia/ wyjścia

Wypisywanie strumienia na ekran

```
cout << "Witaj świecie!!!" << endl;
```

Zczytywanie strumienia z klawiatury

```
cin >> a;
```