Instytut Informatyki UMCS
Zakład Technologii Informatycznych

Wykład 1

Wstęp do C++

dr Marcin Denkowski

Lublin, 2019

AGENDA

1. Informacje organizacyjne
2. Różnice między C a C++
3. Wstęp do klas i obiektów
4. Struktura wieloplikowa
5. Przykłady `iostream`

WARUNKI ZALICZENIOWE

- Wykład + laboratorium
  - Wykład – na **zal**
  - Laboratorium na **ocenę**
- Strona przedmiotu na kampus.umcs.pl:
  - *http://kampus.umcs.pl/course/… {Programowanie obiektowe}*
- Kolokwia zaliczeniowe

  I – 06.05.2019 (50p)

  II – 21.06.2018 (50p)
  - zaliczenie >50p
- Kolokwium poprawkowe (wrzesień 2019)

LITERATURA

- Pozycje książkowe

1. Gręboszj., „**Opus magnum C++11. Programowanie w języku C++**”, Helion, 2017
2. Prata S., „**Szkoła programowania Język C++**”, Wydanie VI, Helion, 2012
3. Stroustrup B., „**Język C++. Kompendium wiedzy**”, Wydanie IV, Helion, 2014
4. Krzaczkowski J., „**Zbiór zadań z programowania w języku C/C++**” cz. 1 i cz. 2, Instytut Informatyki UMCS, Lublin, 2012.

- Manuale internetowe

1. **CppReference**, https://cppreference.com
2. **Cplusplus**, http://www.cplusplus.com
3. **WikiBooks**, https://pl.wikibooks.org/wiki/C++

PRZECIĄŻANIE FUNKCJI

- *Function overloading*
- Możliwość wiązania z jedną nazwą wielu funkcji
- Rozróżnienie na podstawie sygnatury funkcji (listy parametrów)
- Podczas kompilacji funkcja dostanie unikatowy identyfikator (**`extern "C"`** dla kompatybilności)

```
float   add(float a, float b)        { return a+b; }
int     add(int   a, int   b)        { return a+b; }
complex add(complex a, complex b)    { return a+b; }

struct vec2 {float x,y; };
vec2    add(vec2 a, vec2 b) { return (vec2){a.x+b.x,a.y+b.y};}

int main() {
    add(3, 4);      // add(int,int)
    add(3.14, 3); //add(float, float)
    vec2 a,b;
    add(a,b);       //add(vec2, vec2)
}
```

SPECYFIKATOR TYPU: AUTO

1. Do wersji C++11
   – zachowanie identyczne jak dla C, tj. specyfikator zmiennej automatycznej
2. Od wersji C++11
   – zmienna zadeklarowana ze specyfikatorem **`auto`** ma typ dedukowany automatycznie podczas inicjalizacji

```
auto add(float a, float b) { return a+b; }

int main() {
    auto a = 1+2;          // a: int
    auto b = add(1,2);     // b: float

    auto arr = {1,2,3}; // arr: int[3]

    auto int x; // poprawne w C++98, bledne w C++11
    auto x;     //poprawne w C, bledne w C++
}
```

NOWA WERSJA PĘTLI FOR

- *Range-based for loop* (od C++11)
- Iteracja po wszystkich elementach kontenera (kolekcji)

```
for (declaration variable: range_expression)
  intrukcje
```

- Może być używana dla tablic automatycznych oraz obiektów, które mają zdefiniowane metody `begin()` i `end()`

```
int array[] = {1,2,3,4,5,6};

for(int i : array)        // tablica automatyczna
   cout << i << endl;

for(auto i : {2,3,4,5,6,11} )     // lista inicjalizacyjna
   cout << i << endl;

struct vec3 { float x,y,z; };
for(auto v : (vec3[]){{2,3,4}, {0,9,8} )   // tablica automatyczna z wlasnym typem
   cout << "[" << v.x << ", " << v.y << ", " << v.z << endl;
```

INNE DROBNE RÓŻNICE

1. Natywny typ **`bool`** i wartości **`true`** i **`false`**
2. Stałe **`const`** są stałe w dosłownym rozumieniu
3. Nieznaczne różnice w łączności zmiennych
4. Pusta lista parametrów funkcji oznacza **`void`** (a nie niewyspecyfikowaną listę)
5. Drobne różnice w listach inicjalizacyjnych

KLASA

1. Klasa (*class*) – typ złożony, własny programisty

Słowo kluczowe class
zaczyna deklarację/definicję klasy

Składowe klasy mogą
być polami lub metodami

```
class nazwa_klasy       //deklaracka klasy
{
    typ pole1;                      //pole prywatne
    typ metoda1(parametry);         //funkcja skladowa prywatna

public:                             //etykieta widocznosci (dostepu)
    typ pole3;                      //pole publiczne
    typ metoda3(parametry);         //funkcja skladowa publiczna

private:                            //etykieta widocznosci  (dostepu)
    typ pole4;                      //pole prywatne
    typ metoda4(parametry);         //funkcja skladowa prywatna

};  //srednik konczacy definicje
```

Sekcja publiczna, interfejs klasy

Sekcja prywatna, składowe dostępne
tylko dla metod tej klasy

METODY KLASY

1. Metoda (*method*) – funkcja składowa (*member function*) klasy
2. Może być wywoływana tylko na rzecz konkretnego obiektu
3. Definicja może być zawarta wewnątrz definicji klasy lub poza nią

```
class Player
{
    ...
public:
    void move(int x, int y)   //definicja wewnetrzna, inline
    {
        … definicja funkcji
    }

    void jump(int);     //tylko deklaracja metody
};

void Player::jump(int height)   //zewnetrzna definicja metody klasy Player
{
    … definicja funkcji
}
```

POLA KLASY

1. Pole klasy (*data member*) – zmienne będące składowymi klasy
2. Mogą to być zmienne typów wbudowanych, pochodnych (tablice, wskaźniki) i złożonych (obiekty innych klas)

```
class Player
{
    int hp;              //składowa prywatna

public:
    vec3 position;       //składowa publiczna
};
```

INSTANCJA KLASY

- Instancja klasy (*class instance*, *class object*) – konkretny obiekt w pamięci realizujący daną klasę, zmienna typu złożonego zgodna ze specyfikacją klasy

```
class Time      //definicja klasy
{
public:
    int hour, minute, second;
};

int main()
{
    Time time;              //instancja (obiekt) klasy Time
    Time arrayTime[40];     //tablica obiektów klasy Time
    Time* p_time;           //wskaznik na obiekt klasy Time
}
```

- Każdy obiekt przechowuje kopie swoich własnych pól składowych
- Metody klasy występują w jednym egzemplarzu i są współdzielone przez wszystkie obiekty tej klasy

KLASA A STRUKTURA

W Języku C++ jedyną różnicą pomiędzy klasą **`class`** a strukturą **`struct`** jest domyślna widoczność składowych:

– **`struct`** – domyślnie **`public`**

– **`class`** – domyślnie **`private`**

PROJEKT WIELOPLIKOWY

I Pliki nagłówkowe (*header file*)

- rozszerzenie .h lub .hpp
- nie są dosłownymi jednostkami translacji
- dołączane do plików źródłowych (`include`)
- zawierają:
  - deklaracje funkcji
  - definicje funkcji `inline`
  - deklaracje/definicje klas
  - definicje makr

II Pliki źródłowe (*source file*)

- rozszerzenie .c, .cpp, .cxx
- są jednostkami translacji (.cpp → .o)
- zawierają:
  - definicje funkcji i metod
  - instancje obiektów

PLIK NAGŁÓWKOWY, PRZYKŁAD

```
#ifndef MY_H    // zabezpieczenie przed wielokrotnym dolaczaniem
#define MY_H

class Array       //definicja klasy
{
    int _size;              //prywatne pole
    int _elements[64];      //prywatne pole, tablica automatyczna

public:
    int element(int i); //publiczna metoda, deklaracja
    int size()          //publiczna metoda wraz z definicja
    { return _size; }
};

void funkcja(float);    //deklaracja funkcji globalnej

inline int add(int a, int b) //definicja funkcji globalnej typu inline
{ return a+b; }

#endif
```

PLIK ŹRÓDŁOWY, PRZYKŁAD

```
#include "my.h"     // zabezpieczenie przed wielokrotnym dolaczaniem

int Array::element(int i)  //definicja metody publicznej klasy Array
{
    if( i>=0 && i < _size )
        return _element[i];
    return 0;
}


void funkcja(float i)   //definicja funkcji globalnej
{
    return i;
}

#endif
```

PRZESTRZENIE NAZW

1. Klasyczny problem przestrzeni nazw
2. Nazwane przestrzenie nazw

```
namespace NaszaNazwa {
// zmienne,  klasy, funkcje
   int droga;
}
```

3. Odwołanie do nazwy zdefiniowanej w przestrzeni nazw

```
NaszaNazwa::droga = 5;
```

4. Globalna przestrzeń nazw

```
::droga = 5;
```

5. Włączenie nazw z przestrzeni nazwanej do globalnej (lub lokalnej)

```
using namespace NaszaNazwa;
using namespace NaszaNazwa::droga;
```

KONWENCJE PLIKÓW NAGŁÓWKOWYCH

Rodzaj nagłówka	Konwencja	Przykład	Uwagi
Stary styl C++	Kończy się `.h`	`iostream.h`	Używane w C++
Stary styl C	Kończy się `.h`	`math.h`	Używane w C i C++
Nowy styl C++	Brak rozszerzenia	`iostream`	Używane w C++ z **namespace std**
Konwersja z C	Przedrostek c Brak rozszerzenia	`cmath`	Używane w C++ z **namespace std**

IOSTREAM

- Klasy definiujące strumienie

```
class std::ostream;   //definicja strumienia wyjsciowego
class std::istream;   //definicja strumienia wejsciowego
```

- Obiekty klas strumieni

```
std::ostream std::cout; //obiekt globalny, strumienia
                        wyjsiowego, skojarzony z ekranem
std::istream std::cin; //obiekt globalny, strumienia
                       wejściowego, skojarzony z klawiaturą
```

- Operatory wstawiania i wyjmowania do/ze strumienia

```
ostream& operator<<(typ);  //cout << 5;
istream& operator>>(typ);  //cin >> var;
```

IOSTREAM, PRZYKŁAD

```
#include <iostream>
//using namespace std;   //wlaczenie calej przestrzeni std
//using namespace std::cout;  //wlaczenie tylko obiektu cout

int main(void)
{
    std::cout << „Witaj swiecie” << std::endl;
    std::cout << „Wyswietlenie liczby „ << 3.1415f;
    std::cout << „ „ << 2.81 << „, koniec” << std::endl;

    //pobranie liczb
    int k;
    float x;
    std::cin >> k >> x;

    //pobranie napisu
    char str[8] = {0};
    std::cin >> str;  //niebezpieczne – mozliwy blad przepelnienia
    std::cin.get(str, 8);
    return 0;
}
```