Programowanie I
Wykład 12

dr inż. Adam Zielonka

Instytut Matematyki,
Politechnika Śląska

Gliwice 11.01.2019

Klasa vector

```
#include<vector>

vector< int > liczby; \\utworzenie pustego wektora

vector< int > liczby(10); \\10 - komórek zainicjowanych zerami

vector< int > liczby(5, 3); \\5 komórek zainicjowanych 3

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

vector<int> d( t, t+6 ); //1, 2, 3, 4, 5, 6

vector<int> e( t+2, t+4 ); //3, 4

vector<int> f( e ); //kopia wektora e

vector<string> napisy; //wektor łańcuchów znaków
```

Klasa vector – metody

```
vector<int> w(10);

w.at(5) = 11; //podobnie jak w[5] = 11;
//ale w przypadku przekroczenia zakresu wyrzuci wyjątek

w.size(); //rozmiar -- ilość elementów

w.empty(); //test -- czy wektor jest pusty?

w.clear(); //usunięcie wszystkich komórek

w.push_back( 4 ); //dodaje na koniec liczby 4

w.pop_back(); //usuwa ostatnią komórkę
//ale jej nie zwraca

w.front(); //zerowy element
w.back(); //ostatni element
```

Klasa

Klasa

- Określa nowy typ.
- Klasa jet zbiorem danych różnych typów i funkcji, które na nich operują.
- Składowa dana klasy – pola klasy (data member).
- Składowa funkcja klasy – metoda klasy (member function).
- Klasa może być modelem jakiegoś rzeczywistego obiektu.
- Dzięki utworzeniu klas programując można myśleć w kategoriach rozważanego problemu.

Klasa

Deklaracja klasy

```
class MojaKlasa
{
    int pole1;
    double pole2;
    void metoda1(void);
};
```

Użytkownik klasy – (programista ją używający) zazwyczaj nie zna szczegół implementacji klasy. Zasadniczo klasa (jej dane oraz metody) podzielone są na dwie główne części:

- publiczną – dostępna z zewnątrz klasy,
- prywatną – wewnętrzna, dostępna tylko w zakresie klasy.

Klasa

Interfejs publiczny klasy

Metody publiczne klasy tworzą tzw. **interfejs publiczny klasy**. Użytkownik klasy zna tylko interfejs publiczny (nie zna szczegółów implementacji). Używając klasę, w której nie zmienił się interfejs publiczny a zmieniła się implementacja nie trzeba ingerować w implementację, w której klasa ta została użyta.

```
class MojaKlasa
{
    public:
    // interfejs publiczny
    private:
    //składowe prywatne
};
```

Obiekt

Klasa a obiekt

Klasa jest "przepisem", "projektem ", który opisuje czym ma być i w co ma być wyposażony nowy typ. Obiektem natomiast będzie konkretny egzemplarzem danej klasy, który posiada określone wartości poszczególnych pól klasy.

```
class Punkt
{
    public:
     int x;
     int y;
    private:
     int cos;
};
Punkt p;
Punkt p1* = new Punkt();

p.x=3;
p1->x=4;

p.cos=5;  // błąd brak dostępu
```

Podział kodu klasy na pliki

MojaKlasa.h

```
#include<iostream>
class MojaKasa{
    public:
        int a;
        int b;
        void metoda(void);
    private:
        void metodaPrywatna(void);
};
```

MojaKlasa.cpp

```
#include"MojaKlasa.h"

void MojaKlasa::metoda(void){
   metodaPrywatna();
   std::cout<<"Pole a="<< a <<std::endl;
}

void MojaKlasa::metodaPrywatna(void){
   std::cout<<"Obiekt typu KlasaMoja" <<std::endl;
}
```

Konstruktor klasy

Konstruktor nie zwraca żadnej wartości.

MojaKlasa.h

```
#include<iostream>
class MojaKasa{
    public:
        int a;
        int b;
        MojaKlasa( void );
};
```

MojaKlasa.cpp

```
#include"MojaKlasa.h"

MojaKlasa::MojaKlasa( void ){
   a=12;
   b=13;
}
```

Przeciążenie konstruktorów

Konstruktor nie zwraca żadnej wartości.

MojaKlasa.h

```
#include<iostream>
class MojaKasa{
    public:
        int a;
        int b;
        MojaKlasa( void );
        //domyślna inicjalizacja danych

        MojaKlasa( int a, int b);
        //inicjalizacja danymi zadanymi

        MojaKlasa(std::string s);
        //ustawienia konwertowane z łańcucha znaków
};
```

Konstruktory – użycie

Przykłady

```
MojaKlasa mk();
MojaKlasa* mk1 = new MojaKlasa();
//domyślny

MojaKlasa mk2( 10, -3 );
MojaKlasa* mk3 = new MojaKlasa(10 , -3);
//inicjalizacja danymi zadanymi

MojaKlasa mk4("4,3");
MojaKlasa* mk5 = new MojaKlasa("4,3");
//ustawienia konwertowane z łańcucha znaków
```

Destruktor

```
#include<iostream>
class MojaKasa{
    public:
        int a;
        int b;
        MojaKlasa( void );
        MojaKlasa( int a, int b);
        MojaKlasa(std::string s);

       ~MojaKlasa(void);
};
```

```
MojaKlasa::~MojaKlasa(void)
{
    //sprzątanie
}
```

Hermetyzacja

Hermetyzacja jest jednym z filarów programowania obiektowego. Implementując powinniśmy zadbać, żeby użytkownik klasy nie miał bezpośredniego dostępu do wszystkich danych obiektu.Operować nimi można jedynie za pomocą manipulatorów.

```
#include<iostream>
class MojaKasa{
    public:
        MojaKlasa( void );
        MojaKlasa( int a, int b);
        MojaKlasa(std::string s);
       ~MojaKlasa(void);

       void setA(int a);
       int getA(void);

    private:
        int a;
};
```