STRUKTURE PODATAKA I ALGORITMI

Međuispit 1

CLASS

Klasa (eng. class) je korisnički – definiran tip podataka koji koristimo u našem programu. Klasa služi kao svojevrsni "blueprint" za kreiranje objekata. Članovi koji se nalaze u klasi su po default-u privatni, dok su u slučaju strukture javni.

Koja je razlika između klase i strukture ako su oboje "ista" stvar? **Nema je**… Razlika je u tome što točno planiramo raditi.

- Ako imamo potrebu za tipom čija je glavna namjena čuvanje podataka bez puno operacija na njima ---> **Struktura**
- Za sve ostalo ---> **Klasa**

Kako definiramo klasu? Klasa se definira na sljedeći način:

```
#pragma once
class Rectangle
{
private:
	double a, b;
public:
	// konstruktori
	Rectangle(double a, double b);

	// seteri
	void set_a(double a);
	void set_b(double b);

	// geteri
	double get_a();
	double get_b();

	// funkcije s kojima radimo
	double opseg();
	double povrsina();
};
```

Na slici se može vidjeti da se klasa radi po istom principu kao i struktura, jedina je razlike u tome što sada postoje **private** i **public** koji definiraju da li će se varijable moći koristiti samo unutar klase ili van nje.

- Varijable koje se nalaze u private dijelu mogu se koristiti samo unutar klase, ne i u mainu.
- Varijable koje se nalaze u public dijelu mogu se koristiti van klase, tj. u mainu.

Što sve spremamo u public dio klase, a što u private dio?

- **PRIVATE** – private sadržava sve vaše varijable koje ćete koristiti, bilo to int, double, float. Uz to, tu se stavljaju sve varijable koje se ne koriste u mainu.
- **PUBLIC** – u public dio se stavljaju sve metode (funkcije), konstruktori i destruktori koje definiramo i koje želimo koristiti u mainu

Kako kreiramo klasu?

Desni klik na projekt ---> Add ---> Class ---> Klasi damo neki naziv ---> OK ---> Dobili smo Header file (.h) i cpp file. (CPP file će kasnije biti objašnjen)

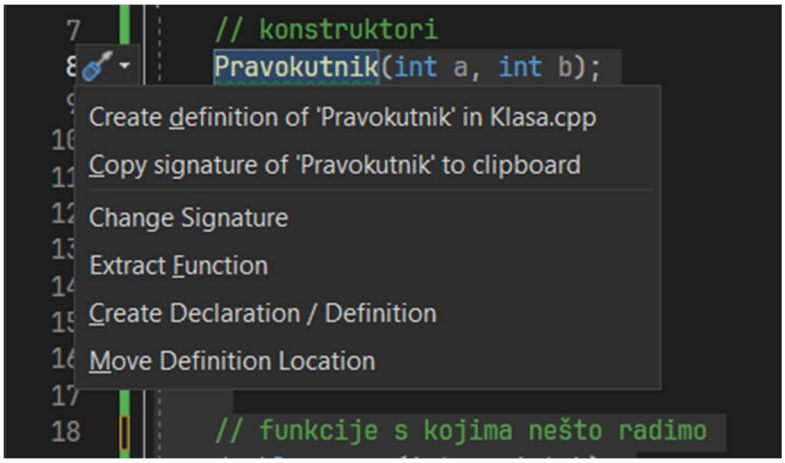
#pragma once dobijate po default-u u vašem header file-u, on služi kao include zaštita i njega ostavljate takvim kakav jest (detaljnije od ovoga nije potrebno znati).

Kako jednostavno deklariramo sve funkcije i inicijaliziramo varijable i metode?

1. KORAK – selectamo public dio

```
6   public:
7       // konstruktori
8       Pravokutnik(double a, double b);
9
10      // seteri
11      void set_a(double a);
12      void set_b(double b);
13
14      // geteri
15      double get_a();
16      double get_b();
17
18      // funkcije s kojima nešto radimo
19      double opseg(double a, double b);
20      double povrsina(double a, double b);
21  };
```

2. KORAK – kliknemo ALT + ENTER



3. KORAK – Kliknemo na Create Declaration / Definition

Kada smo to kliknuli, napokon ulazimo u nas .cpp file koji smo dobili uz .h file.

.cpp file bi vam inicijalno trebao izgledati ovako:

```cpp
#include "Rectangle.h"

Rectangle::Rectangle(double a, double b)
{
}

void Rectangle::set_a(double a)
{
}

void Rectangle::set_b(double b)
{
}

double Rectangle::get_a()
{
    return 0.0;
}

double Rectangle::get_b()
{
    return 0.0;
}

double Rectangle::opseg()
{
    return 0.0;
}

double Rectangle::povrsina()
{
    return 0.0;
}
```

Na slici se može vidjeti na nema onih varijabli koje su definirane u .h file-u. Iako ih ne vidimo, one su tu i aktivno ih možemo koristiti u daljnjem radu.

Za inicijalizaciju varijabli postoji nekoliko metoda, dvije najvaznije su:

- Inicijalizacija konstruktorom
- Inicijalizacija seterima

Inicijalizacija seterima izgleda ovako:

```
void Rectangle::set_a(double a)
{
    this->a = a;
}

void Rectangle::set_b(double b)
{
    this->b = b;
}
```

Kao što se može primijetiti, za inicijalizaciju varijable a i b koristili smo pointer **this. This** je pointer na samoga sebe (u prijevodu pokazuje na našu varijablu). This pointere koristimo umjesto drugih načina iz razloga što ne stvaramo dodatne varijable (ili kopije varijabli) kako bi inicijalizirali varijablu.

Inicijalizacija konstruktorom izgleda ovako:

```
Rectangle::Rectangle(double a, double b)
{
    this->a = a;
    this->b = b;
}
```

Kasnije ćemo detaljnije objasniti kako konstruktori funkcioniraju, te kako zapravo ovo radi. Za sada se držimo inicijalizacije seterima.

Za dohvaćanje potrebnih varijabli koristimo getere koje smo definirali ranije. Getere nije preporučljivo previše koristiti jer je cilj napraviti program koji se neće u potpunosti “otvoriti” korisniku (u prijevodu, getane varijable postaju javne, a to nije nužno dobro).

Dohvaćanje geterima izgleda ovako:

```
double Rectangle::get_a()
{
    return a;
}

double Rectangle::get_b()
{
    return b;
}
```

Za što nam služe geteri? Suština getera je da vrati traženu varijablu. U ovom slučaju, ako u mainu pozovemo funkciju iz klase naziva **get_b();** , ona će ga dohvatiti i vratiti varijablu nama (kasnije ćemo demonstrirati kako to izgleda).

Defaultni konstruktor vs. Korisnički definiran konstruktor:

Konstruktor je po definiciji metoda koja stvara objekt na klasi. Konstruktor može biti definiran na više načina, a jedan od njih je onaj defaultni.

Defaultni konstruktor je automatski generiran konstruktor koji se poziva onog trenutka kada u main stavimo **Pravokutnik p1;** Defaultni konstruktor je izuzetno opasan. Zašto je opasan? Opasan je iz razloga što one varijable a i b koje smo prije definirali neće biti inicijalizirane, nego će se u njima naći smeće.

Korisnički definiran konstruktor je onaj konstruktor koji je jasno definiran u klasi i na kraju pozvan u mainu. To je u suštini konstruktor kojemu najčešće šaljemo vrijednosti pri pozivu. Korisnički definiranom konstruktoru kojemu nije poslana niti jedna varijabla možemo namjestiti da iste defaultno inicijalizira. Objasnimo pomoću slika…

```
Rectangle::Rectangle(double a, double b)
{
    this->a = a;
    this->b = b;
}
```