Вовед во ООП – класи, објектии функции членки, дефинирање на класа со функција членка

1. Објектно ориентирано програмирање

Почетоците на ООП може да се пронајдат во 1960те. Како што хардверот и софтверот ста-нуваше се посложен, квалитетот често беше компромитиран. Истражувачите бараа начини да го задржат квалитетот на софтверот и да го развиваат ООП и да посветат внимание на честите проб-леми и посебно силно нагласување на повеќекратните единици од програмската логика. Методологијата се фокусира повеќе на податоци отколку на процеси, со програми составени од самостојни модули (објекти) секој си содржи сите информации потребни за да манипулира со сопствената податочна структура. Тоа е спротивно со постоечкото модуларно програмирање што беше доминатно многу години и се фокусираше на функциите од модулите повеќе отколку на податоците но исто така обезбедуваше повеќекратна употреба на кодот и повеќекратна употреба на самостојните единици од програмската логика, овозможуваше соработка преку употреба на поврзани модули (потпрограми). Овој конвенционален пристап, којшто сè уште постои, насочен е да ги третира податоците и однесувањата посебно.

На објектно-ориентирана програма може да се гледа како на колекција од кооперативни објекти, за разлика од вообичаениот модел, во којшто на програмата се гледа како на листа од задачи (потпрограми) пред да се извршат. Во ООП секој објект е способен да прими порака, обработи податок и да прати порака на друг објект и на него може да се гледа како на независна "машина" со посебна улога или одговорност. Операторите на тие објекти се тесно поврзани со објектот. На пример, структурите на податоци имаат тенденција ка ги носат нивните оператори со нив (или барем да ги "наследат" нив од слични објекти или класи).

Програмскиот јазик Simula бил привиот јазик што вовел ООП концепти (објекти, класи, поткласи, виртуелни методи, coroutines и посебна симулација на настан) како дел од Algol. Simula исто така користел автоматско собирање на ѓубре коешто претходно било измислено за функцио-налниот програмски јазик Lisp. Simula се користел за физичко моделирање, како на пример модели за изучување и подобрување на движењето на бродовите и нивниот товар низ товарните порти. Smalltalk бил првиот јазик којшто бил наречен "објектно-ориентиран".

2. Податоци од тип објект и класа

Во програмирањето се користат класи и објекти. **Објект** во реалниот свет може да биде автомо-бил, велосипед, во компјутерските игри топка или некој лик. Секој објект е дефиниран со состојба и однесување: пр. ликот во играта може да трча, скока, а однесувањето на ликот може да се опише до брзината на трчање или висината на скокањето. Во програмата **објектот** е опишан со **промен-ливи** кои му одредуваат **состојба** и **методи** кои му одредувааат **однесување.**

Класа е основа на објектот. Затоа прво се креираат класи врз основа на кои се креираат објекти. Пр. кога креираме класа лик, во играта може да се креираат повеќе ликови имат различни состојби (пр. некој лик може да лета), и однесување (има брзина на летање)

1)**Објект** е сложена структура на податок кој се опишува со два дела
Објект=Податоци+Методи (функции и процедури)

2)**Класа** е проширена податочна структура што содржи не само **податоци**, туку ги содржи и **функциите** што се применуваат врз тие податоци. **Класа е множество** на објекти што делат исти карактеристики, однесување, релации и функции.

Примери:
1) класа – ученик-податоци: име, презиме, низа со оценки по предмети,
методи – функција за пресметка на среден успех : збирот на оценките се дели со бројот на предмети;
објект – ученик од III-9 клас,или со име и презиме
2) класа – продавница-податоци: име, цена и количество за секој артикал, **методи** – функција за пресметка на попис во продавницита ...
објект – аптека , маркет и сл.
3) класа – возило , објект- автомобил, авион, воз и сл.

Пристап до класите:
public - членовите од класата се достапни за сите функции во програмата
private - членовите од класта се достапни само за функциите од иста класа
protected - членовите од класата се на располагање за сите функции од иста класа и поврзани функции на класи кои се добиени од почетната класа, а не се видливи за други функции

Дефинирање на класа во C++:
```
#include <iostream>
using namespace std;
class име на класата
{
 обезбедување на пристап до класата:
 листа на атрибути и однесување/функции;
};
```

Примери на програми со класи:

1) Класа со едноставна функција – членка, што нема влезни податоци и не враќа вредност, туку само печати податоци (за тоа се користи резервираниот збор – void):

```
#include <iostream>
using namespace std;
class  Pozdrav
{
public:   //пристап до класата
  void  Poraka()   // функција за порака
{
cout << "Klasa so ednostavna funkcija so poraka!"<< endl;
}
};   // завршува дефинирањето на класата
int  main()  //главна дел на програма (главна функција)
 {
   Pozdrav  objPozdrav;   //креирање на објект  со име objPozdrav од класата Pozdrav
  objPozdrav.Poraka ();  // повикување на функција – членка на класа
  return 0;
 }
```

2) Класа со функција – членка на која и требаат влезни вредности (параметри, аргументи-податоци) за да испечати некаков резултат:

```
#include <iostream>
```

```
using namespace std;

class  Kvadrat

{

public:  //пристап до класата

  void  Poraka(int broj)  // функција со порака што ќе печати квадрат на број

{

   cout<<"Kvadratot na brojot e:"<<broj*broj<<endl;

}

};

int  main()

 {

  int br;

  cout<<"Vnesi broj";

  cin>>br;

  Kvadrat  objKvadrat;  // креирање на објект  со име objKvadrat од класата Kvadrat

  objKvadrat.Poraka (br);   // повикување на функција – членка на класа

  return 0;

 }
```

3) Програма со која се пресметува плоштината, периметарот и должината на дијагоналата на правоаголник. При тоа, се креира класа правоаголник, која има податоци за должината и ширината на правоаголникот и три функции за пресметување на плоштината, периметарот и должината на дијагоналата.

```
#include <iostream>
#include <cmath>
using namespace std;
class Pravoagolnik
{
  public:
      float visina, sirina;
      float plostina(float visina, float sirina)
      {
          return visina * sirina;
      }

     float perimetar(float visina, float sirina)
      {
          return(2*(visina + sirina));
      }
```

```
      float  dijagonala(float visina, float sirina)
      {
         return(sqrt(pow(visina,2)) + pow(sirina,2));
      }
};
int  main()
{
      Pravoagolnik mal;
      float  v,s;
      cout<<"Vnesi sirina i  visina na pravoagolnikot"<<endl;
      cin>>s>>v;
      mal.sirina=s;
      mal.visina=v;
      cout<< " Plostinata na pravoagolnikot e " << mal.plostina(v,s)<<endl;
      cout<< " Perimetarot na pravoagolnikot e " << mal.perimetar(v,s)<<endl;
      cout<< " Dijagonalata  na pravoagolnikot e " << mal.dijagonala(v,s)<<endl;
  return 0;
}
```

4. Програма со која се пресметува плоштината и периметарот на круг. При тоа, се креира класа круг, која има две функции за пресметување на плоштината и периметарот на кругот.

```
//p-2 C Plostina perimetar na krug
//Programa so koja se presmetuva plostinata i perimetarot na krug so radius r
//so primena na klasi
#include <iostream>
#include <math.h>
using namespace std;

class Krug
{
  public:
    double perimetar(double r)
    {
       return 2*r*3.1415;
    }
   double plostina(double r)
  {
       return pow(r,2)*3.1415;
   }
};

main()
{
  double r;
  Krug k;
  cout<<"Vnesi ja vrednosta na radiusot"<<endl;
  cout<<"r=";
  cin>> r;
  cout<<"Perimetarot na krugot e " << k.perimetar(r) << endl;
  cout<<"Plostinata na krugot e " << k.plostina(r) << endl;
//system("pause");
return 0;
}
```

Или:

```
//p-2 C Plostina perimetar na krug
//Programa so koja se presmetuva plostinata i perimetarot na krug so radius r
//so primena na klasi
#include <iostream>
#include <math.h>
```

```
using namespace std;

class Krug
{
  public:
      double r;
    double perimetar(double r)
    {
        return 2*r*3.1415;
    }
   double plostina(double r)
  {
        return pow(r,2)*3.1415;
   }
};

int main()
{
     Krug k;
  double r1;

  cout<<"Vnesi ja vrednosta na radiusot"<<endl;
  cout<<"r=";
  cin>> r1;
  k.r=r1;
  cout<<"Perimetarot na krugot e " << k.perimetar(r1) << endl;
  cout<<"Plostinata na krugot e " << k.plostina(r1) << endl;
//system("pause");
return 0;
}
```