Uvod: OOP s C++ jezikom

Predavač: Dr. sc. Ivo Mateljan

Asistenti: Mr.sc. Marijan Sikora i Mia Čizmić

WWW: Materijali s predavanja na :

http://www.fesb.hr/~mateljan/cpp/

Konzultacije: soba 412, četvrtkom 11h.

Dodatni udžbenici i kompajleri na CD-romu:

- **R. Astrahan : Computer science tapestry, McGraw-Hill, 2000.**
  - **dobra knjiga za one koji neznaju C – moderni pristup programiranju – nastala po programu koji je prihvatio veliki broj sveučilišta u SAD – tzv. advanced placement cours**
- **Robert Lafore: Interactive C++ Course, *Macmillan Computer Publishing, 1996***
- **Bruce Eckel: Thinking in C++, MindViev Inc., 2000.**
- **Kate Gregory: Special Edition Using Visual C++ 6, Que Publishing, 1998.**
- **ISO Standard**

Knjige za napredno učenje C++

- **Stanley B. Lippman, C++ Primer, Second Edition, Addison- Wesley, 1991**
- **Magaret A. Ellis and Bjarne Stroustrup, The Annotated C++ Reference Manual, Addison-Wesley, 1990.**
- **Scott Meyers, Effective C++, Addison- Wesley, 1992.**
- **James O. Coplien, Advanced C++: Programming Styles and Idioms, Addison- Wesley, 1992.**
- **Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Design Patterns, Addison Wesley, Reading, MA, 1995.**
- **David R. Musser and Atul Saini, STL Tutorial and Reference Guide, Addison-Wesley, 1996.**

Četiri pristupa programiranju:

1. **Proceduralno programiranje**
   - **Jezici: Fortran, Basic, Pascal, Modula 2, C - nazivaju se i algoritamski jezici**
     Slijedi se tok izvršenja programa u računalu: sekvenca, selekcija, iteracija
   - **Organizacija podataka: prosti tipovi, nizovi i strukture**
   - **Modularno programiranje olakšava organizaciju velikih programa**
2. **Objektno orjentirano programiranje**
   - **Jezici: Smalltalk, Eiffel, Modula3, Objective C, C++, Java, C#**
   - **Proširuje se koncept apstraktnog tipa podataka**
   - **Tip se specificira u programskoj strukturi – klasi, koja određuje ponašanje sličnih objekata**
   - **Objekt je pojavnost neke klase**
   - **Ponašanje i stanje objekta određuje članovi klase:**
     i. **podaci – bilježe stanje objekta**
     ii. **funkcije – određuju ponašanje objekta – nazivaju se metode klase**
3. **Funkcionalno programiranje**
   - **Jezici: Lisp, Sheme, Haskell, ML**
   - **Matematički pogled na programiranje u kojem su funkcije entiteti prve vrste:**
     i. **funkcije se tretira kao podatak koji se može prenositi u drugu funkciju**
     ii. **rezultat primjene funkcije može biti neka druga funkcija (funkcija vraća funkciju)**
   - **Prihvaćeni od matemetički usmjerenih umova, jer se pokazuje da bi se ovim jezicima mogli dobiti „sigurniji“ programi.**
4. **Logičko programiranje**
   - **Jezici: Prolog ...**
   - **Temelji se na matemetičkoj logici - predikatnom računu**
   - **Primjena im je ograničena na probleme koji se mogu logički specificirati**

Je li C++ dobar jezik

C jezik je bio dugo kritiziran u znanstvenoj zajednici, a danas se svi slažu da je to jezik koji će nadživjeti sve druge jezike.

Što je s C++ jezikom ?

C++ je C jezik kojem su dodane mogućnosti objektno orijentiranog programiranja.
"C with Classes" dizajnira Bjarne Stroustrup 1980; i od toga nastaje C++.

Uspjeh C++ jezika je dijelom i u tome što je potpuno kompatibilan s C jezikom.

Mana C++ jezika je dijelom i u tome što je potpuno kompatibilan s C jezikom.

Proces učenja C++ je relativno dug.

Java i C# predstavljaju pokušaj da se naprave nešto jednostavniji jezici od C++. To je samo dijelom ispunjeno, jer je uz upoznavanje jezika potrebno upoznati i neophodne programske biblioteke. U tom slučaju ovi jezici nisu bitno lakši za učenje od C++ jezika.

Što ćemo učiti?

- **Kako implementirati ADT - “abstract data types” pomoću C++ clasa**
- **C++ tipove podataka, kontrolu tijeka programa i funkcije**
- **Nasljeđivanje o polimorfizam – temelj za OOP i “code reuse”**
- **Generičke klase i STL biblioteka kontenjerski objekti – string, vector, list, map –**
- **Efikasno i sigurno korištenje memorije**
- **Kako koristiti “assertions” i “exceptions” za razvoj korektnih programa**
- **Organiziranje C++ programa u više izvornih .cpp (izvorni kod) i .h (header-prototip) datoteka**
- **Upotreba makefile, debugger-a, IDE alata. GUI**
- **GUI - MFC**

Objekti i njihova implementacija u C i C++ jeziku:

Točka – (point)

atributi - koordinate x,y i boja
metode - pomakni u položaj x1,y1
- postavi boju

Kružnica – (circle)

atributi - koordinate središta x,y, boja i radijus
metode - pomakni u položaj x,y
- postavi boju

Pravokutnik (square)

atributi - koordinate točke x1,y1 i točke x2,y2 te boja
metode - pomakni u položaj x,y
- postavi boju

OOP ideja

Objekti imaju neke zajedničke atribute i metode, a neki se atributi razlikuju.

Možemo li iskoristiti tu činjenicu tako da se jedan objekt specificira pomoću drugog objekta. To je predmet našeg zanimanja.

Najprije pogledajmo kako se realiziraju apstraktni tipovi u C jeziku, a zatim u C++ jeziku

Pristup u C jeziku - ADT point

```
/* specifikacija apstraktnog objekta point */

typedef struct _point
{   int m_x;
    int m_y;
    int m_color
} point, *POINT;

point *new_point() {
    return (point *) malloc(sizeof point)
}

void delete_point(point * this){
    free(this)
}

void set_point_pos(point * this, int x, int y){
    this->m_x = x;  this->m_y = y
}

void set_point_color(point * this, int color) {
    this->m_color = color;
}
```

```
/* aplikacija objekta */

int main()
{

   POINT p1 = new_point();
   PIONT p2 = new_point() ;

   set_point_pos(p1, 2, 3);
   set_point_pos(p2, 2, 3);
   set_color(p1, 7);
   set_color(p2, 7);

   /* ................... */

   delete_point(p2);
   delete_point(p2);

   return 0;
}
```

```
/* specifikacija objekta circle */

typedef struct _circle {
   int m_x;
   int m_y;
   int m_color;
   int m_rad;
} circle, *CIRCLE;

circle *new_circle(){
   return (circle *) malloc(sizeof circle)
}

void delete_circle(circle * this){
   free(this);
}

void set_circle_pos(circle * this, int x, int y ){
   this->m_x = x;  this->m_y = y;
}

void set_circle_radius(circle * this, int r){
   this->m_rad= r;
}

void set_circle_color(circle * this, int color){
   m_color = color;
}
```

```
/* aplikacija objekta/

int main()
{

   CIRCLE c1 = new_circle() ;
   CIRCLE c2 = new_circle() ;

   set_circle_pos(c1, 2, 3);
   set_circle_pos(c2, 2, 3);
   set_circle_radius(c1,7);
   set_circle_radius(c2,6);

   set_color(c1, 7);
   set_color(p2, 7);


   /* ..................*/

   delete_circle(c2);
   delete_circle(c2);

   return 0;
}
```

Uočimo da point i circle imaju zajedničke članove , m_y i m_color i zajedničko tijelo funkcija set_obj_pos(), i set_obj_color().

Zgodno bi bilo napraviti jezik koji zna iskoristiti zajedničke članove i metode, te kojim se objekt može specificirati kao regularni tip podataka. Recimo, C++ .....

```
struct point
{  int m_x;
   int m_y;
   int    m_color;

   set_color(int c)
   {m_color=c;}

   set_position(int x, int y)
   {m_x=x; m_y=y;}
};

struct circle : public point
{
// circle nasljeđuje članove od point;
// treba definirati samo one članove koji
// nisu definirani u klasi point:

   int m_radius;
   set_radius(int r) { m_radius = r;}
}
```

```
// Aplikacija objekta
int main {

// deklaracija dva objekta
// „tipa" point i circle

    point p1;
    circle c1;

// točka operator služi da se dosegnu
// članovi: podaci i funkcije

   p1.set_pos(7,2);
   c1.set_pos(2,3);
   c1.set_radius(2);

//ili

   c1. set_pos(p1.m_x, p1.m_y);
}
```

Gornje strukture opisuju klase objekata tipa `point` i `circle`. Stoga se u C++ za definiranje klase umjesto riječi `struct` češće koristi riječ `class`.

Komponiranje i nasljeđivanje objekata

Komponiranje objekata

```
class Rectangle
{
public:
  point m_p1;
  point m_p2;
  set_pos(int x, int y) { m_p1.x = x; m_p1.y=y; }
  set_width(int w)      { m_p2.x = m_p1.x + w;  }
  set_height(int h)     { m_p2.x = m_p1.y + h;  }
}
```

Nasljeđivanje objekata

```
class Rectangle:  public Point
{
public:
  int m_height;
  int m_width;
  set_pos(int x, int y) { m_p1.x = x; m_p1.y=y; }
  set_width(int w)      { m_width = w; }
  set_height(int h)     { m_height = m_p1.y + h;}
}
```

OOP principi koji su implementirani u C++ jeziku

1. **Sustavno definiranje novih tipova kao podrška za „data abstraction“**
2. **Nasljeđivanje**
3. **Komponiranje objekata**
4. **Polimorfizam funkcija i klasa (virtuelne klase)**

C++ također omogućuje programske tehnike:

5. **Modularno programiranje kao podrška za „data hidding“**
6. **Generičko programiranje**
7. **Rukovanje iznimkama**