Osnovi računarskih sistema

Jelena Tomašević 2005/2006

Čas 1

Beleške sa vežbi iz ORS-a (18.10.2005.)

1.1 Podsećanje sa prošlog časa

Pokazivač- adresa neke promenljive.

```
int *p;
int m, n;
p=&n;
m=*p;

p=0; ili p=NULL; nula pokazivac(ne pokazuje ni na jednu zonu u memoriji).
```

New - operator koji služi za odvajanje zone u memoriji i usmeravanje pokazivača na tu zonu.

Na primer, ako napišemo:

```
int *p;
p=new int(4);
```

U memoriji se odvaja zona u kojoj se upisuje broj 4 i p se usmerava da pokazuje na to.

Delete - operator za oslobadjanje zone u memoriji koja je alocirana sa new. Na primer:

```
delete p;
```

Posmatrajmo sledeći deo koda:

```
int *p,*q, r;
p = new int(2);
q = new int(3);
r = *p + *q;
delete q;
q=&r;
delete q;
```

(ne znamo šta radi. Briše r a r postoji u programu) - brišemo samo ono što smo odvojili sa new!!!

Malkov-skripta (reference-strukture)

1.2 Strukture

Struktura je skup od jedne ili više promenljivih, koje mogu biti različitih tipova, grupisanih zajedno pod jednim imenom. Pomažu da se komplikovani podaci oraganizuju tako da se omogući da se grupa medjusobno povezanih promenljivih tretira kao jedna celina.

Uvode se preko ključne reči struct (u c++ -u postoji bolje rešenje - klase). Npr.

```
 struct Osoba
{
int god;
double pr; //prosecna ocena
}
```

Članovima ove strukture se može pristupiti na sledeći način:

```
Osoba x; //deklaracija
x.god = 25; // pristup clanu god
x.pr = 7.24; // pristup clanu pr

Inicijalizacija je ista kao kod nizova:
Osoba x = {25, 7.24};

Ako je pp pok na strukturu:

Struct Osoba *pp;
```

Članovima te strukture se pristupa na sledeći način:

```
(*pp).god ili  pp->god.
```

1.3 Unije

Uvode se preko ključne reči union i predstavljaju posebnu vrstu strukture (svi članovi dele isti memorijski prostor.) Jedna promenljiva koja može da sadrži objekte različitih tipova i veličina.

```
Npr.
union U
{
    int i;
    char c;
    float f;
}

Pristup je kao kod struktura:
```

```
U a; // deklaracija
a.i = 25; // a sada cuva int
int j;
j = a.i;
a.c = '?'; //a je sada char
j += a.i; //nedefinisan efekat jer je a char
```

U C++ - umesto struktura koriste se klase.

1.4 KLASE

Koncept slogova je u C++ - u doveden na neki novi nivo. Klasa se može shvatiti kao skup objekata (instanci te klase) koji imaju zajedničku strukturu i ponasanje. Dakle, klasa određuje strukturu i ponašanje objekata.

Kako se definiše neka klasa?

```
class A
{
//podaci
//f-je
};
```

Podaci koji opisuju tu klasu zovu se još i atributi (opisuju svojstva), a f-je se zovu metodi. Kako koristimo klasu A?

```
class A
{
int x,y,z;
int f(int x);
}
A a,b; - a i b su objekti klase A (promenljive tipa A)
Kako da pristupimo tim podacima?
a.x - prostup polju x
a.y
a.z
a.f(1)
```

Kad se pojavi neki veliki problem više ljudi radi na njemu. Svako po neku klasu. Ako bi svako mogao da pristupi podacima u klasi ovog drugog javio bi se problem. Zato je obezbeđena zaštita podataka u klasi. Razlikujemo više nivoa pristupa:
1. nivo pristupa je **private** (podacima i metodima navedenim u ovom delu mogu pristupati samo objekti te klase. Ostalima je pristup zabranjen.)
2. **protected** - uvešćemo kod budemo radili nasleđivanje.
3. **public** (podatke i metode navedene u tom delu svi mogu da koriste.)

Konstruktor: Posebna f-ja koja služi za inicijalizaciju objekata neke klase. Ima isto ime kao klasa (po tome ga prevodilac prepoznaje) i nema tip rezultata. Može postojati više konstruktora u okviru jedne klase.

Destruktor: Konstruktori mogu i da rezervišu memoriju koja je potrebna pa se zato javlja potreba i za destruktorima. On postoji samo jedan u klasi i služi da vrati sistemu memoriju koju je konstruktor rezervisao. Ako konstruktor

nije nista alocirao onda ne moramo eksplicitno da navodimo destruktor u klasi. Destruktor ima isto ime kao klasa sa prefiksom $\sim$. Nema argumenata i nema rezultata Kao sto se konstruktor poziva automatski svaki put kad se definise promenljiva neke klase isto tako se i automatski poziva destruktor po zavrsetku oblasti važenja te klase.

Inspektori i mutatori: Inspektori su metode koje ne menjaju vrednosti objekta (njegovo stanje i atribute) a mutatori menjaju. Inspektori imaju const na kraju naslova.

```
int f()const
{//} - inspector
int g()
{} - mutator
```

Operatorske f-je: Deklarišu se tako što njihovo ime počinje sa **operator**. Tako na primer možemo definisati **operator *** i onda može da se napiše $a * b$. Ukoliko pak napisemo $a * b$ a ovaj operator nije definisan u klasi, dobijamo poruku o grešci. Mnogo je čitljivije nego npr a.puta(b);

Primer 1 *Napisati klasu za rad sa kompleksnim brojevima i program koji ilustruje njeno korišćenje.*

```
#include <iostream>

using namespace std;


class KompleksanBroj {

private:
    double re;
    double im;

public:

KompleksanBroj(double r=0.0, double i =0.0)
        {
            re = r;
            im = i;
}
    double real () const
        {return re;}

double imag () const
        {return im;}

KompleksanBroj operator+ ( KompleksanBroj c) const
        {return KompleksanBroj(re+c.re, im+c.im);}

KompleksanBroj operator- ( KompleksanBroj c) const
        {return KompleksanBroj(re - c.re, im- c.im);}
```

```
KompleksanBroj operator* ( KompleksanBroj c) const
{return KompleksanBroj(re * c.re - im * c.im, re * c.im + im * c.re);}

KompleksanBroj operator/ ( KompleksanBroj c) const
        {double x = c.re * c.re + c.im * c.im;
return KompleksanBroj((re * c.re + im*c.im)/x,
(-re * c.im + im * c.re)/x);}

void pisanje (ostream& str) const
{str << '(' << re << ','<< im << ')';}

void citanje (istream& str)
        {char c;
  str>>c>>re>>c>>im>>c;}

}; //kraj klase



inline ostream & operator<< (ostream & str, const KompleksanBroj & c)
    {c.pisanje (str);
  return str;}


inline istream & operator>> (istream & str,  KompleksanBroj & c)
    {c.citanje(str);
  return str;}


main()
{KompleksanBroj a(2,1), b(5,3);
cout << "Racunske operacije:\n";
cout << a << "+" << b << "=" << (a+b)<< endl;
cout << a << "-" << b << "=" << (a-b)<< endl;
cout <<'(' << a << "/" << b << ")*" <<b<<  "=" << (a/b)*b<< endl;
cout<<"unesite kompleksan broj:";
cin >> a;
cout << a;
return 0;
}
```