Vaje 13

Barbara Nežmah

5.1.1998

Predmetno orientirano ali usmerjeno programiranje

Kaj je predmet? Odgovor: miza, stol.... Imamo 24 stolov in ti stoli sodijo v razred stolov.

Razred stolov:

1. noge,
2. naslonjalo za hrbet,
3. naslonjalo za pisanje...

Vzamemo enega iz tega razreda. Če bi rekli, da smo vzeli rdeč stol, potem ta stol ne sodi v ta razred, ker na začetku nismo predpostavili, da so stoli rdeči.

Primeri

Definirajmo razred geometrijskih likov z lastnostmi:

1. ploščina,
2. obseg.

Razred mnogokotnikov, ki je podrazred geometrijskih likov ima še:

1. število ogljišč,
2. dolžina stranice.

Kako je razred definiran? Razred je definiran s skupkom lastnosti. Predmet ima skupek lastnosti, ki jih mora zadovoljiti.

Kakšna je razlika med razredom skladov in enim skladom? Odgovor: En sklad vedno obstaja. Predstavnik razreda skladov pa lahko da ne.

Razred je opis kako stvar izgleda in kako se obnaša. Predmet je pa eden izmed predstavnikov iz tega razreda. To nas spominja na tip. V navadnem c-ju smo imeli, na primer tip `integer`. K temu tipu smo nanizali razne spremenljivke, ki smo jih potrebovali. V c++ pa se spremenljivka danega tipa imenuje predmet.

PRIMER: Definirajmo razred vseh skladov: Pod sklad 1 spada 1, 2, 3, 4 =4 številke.

Pod sklad 2 spada 1, 2 =2 številki.

V c-ju smo imeli:

```
SKL_Ustvari
SKL_Uniči
SKL_Nalozi(sk, st);
SKL_Zlozi(sk)       -> st
SKL_Poln(sk)        -> T/F
SKL_Prazen(sk)      -> T/F
```

kar prevedemo v razred celoštevilčnih skladov z naslednjimi lastnostmi:

1. lahko ga ustvarimo in lahko ga uničimo;
2. nanj lahko naložimo celo število;
3. z njega lahko zložimo celo število;
4. pogledamo, če je poln;
5. pogledamo, če je prazen.

Torej, če imamo predmet sklad, lahko nanj nalagamo ali z njega zlagamo cela števila; pogledamo, če je prazen ali poln nimamo nobenih implementacij. To je *razred celoštevilskih skladov.*

Na začetku imamo *definicijski modul* nato izdelamo *izvedbeni modul* v *obj* obliki. V c++ ima definicijski modul obliko:

```
class IntSklad \{
public:              // metode
  IntSklad();        // s tem stvar ustvarimo
  ~IntSklad();       // unici predmet tega razreda
```

```
  void Nalozi(const int stev);
  int Zlozi();
  bool Poln();
  bool Prazen();

\} // IntSklad
```

Defincije razreda sestoji iz:

1. privatnega dela,
2. zaščitnega dela,
3. javnega dela.

Predmet iz razreda celoštevilskih skladov je sedaj definiran kot:

```
int main(void) \{
  int i;                 // i je iz razreda celih stevil
                         // in int je razred
  IntSklad skl;          // s tem se ustvari predmet iz razreda
                         // IntSklad in predmet se imenuje skl --
                         // pozene se kreator IntSkl()
  .
  .
  .
\}
```

Vsak definicijski modul ima podprograme za uničenje in ustvarjanje. Prav tako tudi vsak razred:

```
  i=10;                    // i-ju smo priredili neko vrednost
  printf("\%d",i);          // stvar se izpise
  .
  .
  skl.Nalozi(10);          // na sklad smo nalozili 10
  skl.Nalozi(20);          // na sklad smo nalozili 20
  skl.Nalozi(i);           // na sklad smo nalozili i
                           // predmetu skl, ki je iz razreda
                           // celostevilcnih skladov posljemo
                           // sporocilo, da naj na sebe nalozi
                           // stevilo i
  if sklad.Poln()
```

```
    printf("sklad je poln");
\end{verbatin}
}
%
Lahko imamo več predmetov iz razreda celoštevilčnih skladov:
{\tt
\begin{verbatim}
  int i;
  IntSklad skl,skl1;
  .
  .
  .
  if skl.Prazen()
    printf("sklad je prazen");
  if skl1.Prazen()
    printf("sklad je prazen");
```

in funkciji `skl.Prazen()` ter `skl1.Prazen()` nimata takorekoč nič skupnega.

C++ poskrbi, da za seboj vse avtomatično počisti. Tako, po koncu izvajanja bloka, pokliče sistem destruktorja za vsakega od predmetov ustvarjenih v bloku (npr. `~IntSklad()`).

Prej smo omenili, da `int` v bitsvu definira razred celih števil. Recimo, predmeti iz tega razreda imajo lastnost, da jim lahko prištejemo drugo celo število:

```
  int i;
  i= 3;                   // je isto kot i.=(3);
  i+= 10;                 // je isto kot i.+=(10)
```