Napredne tehnike programiranja

Vježba 4

Teme

- Osnove objektnog pristupa
- Klase
- Objekti

Korištenje strukture – proceduralni pristup

- Želimo definirati novi tip podatka koji će reprezentirati datum
- U programskom jeziku C definirali bismo strukturu Date:

```
struct Date {
        int day;
        int month;
        int year;
};
```

- •Primjer upotrebe:

```
Date d;
d.day = 21;
d.month = 3;
d.year = 2007;
Date* pd = &d;
(*pd).day = 20;
pd->month = 4;
```

Korištenje strukture – proceduralni pristup

- Neke funkcije za rad s tipom Date:

```
struct Date {
        int day;
        int month;
        int year;
};
void setDate(Date* pd, int d, int m, int y){
        // trivijalna implementacija
        pd->day = d;
        pd->month = m;
        pd->year = y;
}
void printDate(const Date* pd) {
        std::cout << pd->day << " " << pd->month << " "
        << pd->year << "\n";
}
void nextDate(Date*);
```

Struktura

Funkcija koja radi sa strukturom

- Kako onemogućiti korisnika da napravi npr. d.day = 32 ?

Strukture i funkcije

- Ideja: svaka varijabla (objekt) nekog složenog tipa (strukture) se treba sama brinuti za svoju funkcionalnost, tj. svoj sadržaj to ne treba biti uloga globalnih funkcija
- Moramo objediniti definiciju strukture i funkcije koje rade sa njom

Korištenje strukture

- U C++u možemo deklarirati i definirati funkcije unutar strukture:

```
struct Date {
        int day;
        int month;
        int year;
        void set(int, int, int);
        void next();
        void print();
};
```

Funkcije deklarirane unutar strukture

Korištenje strukture

- Definicija funkcije može biti unutar strukture:

```
struct Date {
        int day;
        int month;
        int year;
        void set(int d, int m, int y){
                day = d;
                month = m;
                year = y;
        }
        void next();
        void print();
};
```

Funkcija definirana unutar strukture

Funkcije deklarirane unutar strukture

Korištenje strukture

- ili izvan strukture:

```
struct Date {
        int day;
        int month;
        int year;
        void set(int , int , int );//deklaracija
        void next();
        void print();
};
void Date::set(int d, int m, int y) {
        day = d;
        month = m;
        year = y;
}
```

Operator pripadnosti
(operator dodjele područja)

Strukture i funkcije

- Prednosti spajanja definicije strukture i njenih funkcija:
  - Preglednji program
  - Elementi programa koji rješavaju neki problem su svi na istom mjestu
  - Modularnost programa
  - Povećana mogućnost kasnijeg korištenja gotovog koda

Deklaracija strukture

Prije

```
struct stack {
        int vrh, podaci[100];
};
void makeNull( stack &s  );
void push( stack &s,int  x);
void pop( stack &s );
int top( stack  s);
```

Sad

```
struct stack {
        int vrh, podaci[100];
        void makeNull( void );
        int top( void );
        ...
};
void stack::makeNull( void )
{ vrh = 100; }

int stack::top( void )
{ return podaci[vrh]; }
...
```

Što se mijenja ako promijenimo način na koji deklariramo strukturu?

Promjene u funkciji

Prije

```
struct proba{
        int podatak;
};
void ispisi (proba A){
        cout<<A.podatak<<endl;
}
```

```
proba A;
ispisi(A);
```

Sad

```
struct proba{
        int podatak;
        void ispisi ();
};
void proba::ispisi (){
        cout<<podatak<<endl;
}
```

```
proba A;
A.ispisi();
```

Funkcije kao elementi strukture

```
void stack::makeNull( void )
{ vrh = 100; }

int main( void )
{
      stack S, T;
      S.makeNull();
      T.makeNull();
}
```

- Ako je funkcija deklarirana u strukturi mijenja se način njenog pozivanja
- Funkcija `makeNull` "zna" koja je varijabla (objekt) poziva i ovisno o tome mijenja S.vrh ili T.vrh

Korištenje funkcija

- funkcijama strukture pristupamo jednako kao i varijablama

```
tocka P = {10, 20}, Q, *T = &P;
Q.x = 10; Q.y = 30;
T->x = 15;

P.ispisiTocku();
T->ispisiTocku();
Q.ispisiTocku();
cout << P.udaljenost( Q );

tocka R = T->simetricnaTocka();
```

Primjer

- Napišite implementaciju sljedeće strukture:

```
struct tocka {
        int x, y;
        void ispisiTocku();
        void unesiTocku();
        tocka simetricnaTocka();
        float udaljenost( tocka Q );
};
```

i odgovarajuću funkciju main

Rješenje

```
struct Tocka{  int x,y;
               void ispisiTocku();
               Tocka simetricnaTocka();
               float udaljenost( Tocka *b );
};
void Tocka::ispisiTocku() { cout<<"x:"<<x<<" y:"<<y<<endl;}
Tocka Tocka::simetricnaTocka() {
        Tocka z;
        int a;
        z.x=-x;
        z.y=-y;
        return z;
}
float Tocka::udaljenost(Tocka *b) {
        float a;
        a=sqrt(pow(b->x-x,2)+pow(b->y-y,2));
        return a;
}
```

Rješenje

```
int main () {
        Tocka *x, *y, *z;
        x=new Tocka;
        y=new Tocka;
        z=new Tocka;
        x->x=1;
        x->y=1;
        y->x=2;
        y->y=2;
        x->ispisiTocku();
        y->ispisiTocku();
        *z=x->simetricnaTocka();
        y->ispisiTocku();
        cout<<x->udaljenost(y)<<endl;
        delete x,y,z;
        system("pause");
        return 0;
}
```

Zadatak

- Napišite kompletnu strukturu `Date` (dovršite definicije svih funkcija) i isprobajte je sa slijedećim kodom:

```
Date d;
d.set(21, 3, 2015);
d.next();
Date* pd = &d;
pd->set(20, 4, 2015);
pd->print();
```

```
struct Date {
        int day;
        int month;
        int year;
        void
set(int,int,int);
        void next()
        void print();
};
```

Što je objekt

- Objekt je koncept organizacije programa koji ujedinjuje strukturu podataka i funkcije koje rade sa tom strukturom.
- Na taj način sve potrebno za rad sa nekom strukturom podataka imamo objedinjeno u jednoj cjelini (klasi).

Strukture i klase

- Pretvorimo strukturu u klasu:

```
struct Date {
        int day;
        int month;
        int year;
        void set(int,int,int);
        void next()
        void print();
};
```

```
class Date {
        int day;
        int month;
        int year;
    public:
        void set(int,int,int);
        void next();
        void print();
};
```

Prava pristupa članovima klase

- `public` – članovima klase koji imaju ovo pravo pristupa može se pristupiti izvan klase
- `protected` - članovima klase koji imaju ovo pravo pristupa može se pristupiti samo iz klase i podklasa (klasa koji je nasljeđuju)
- `private` - članovima klase koji imaju ovo pravo pristupa može se pristupiti samo unutar klase (preko funkcija članova)

- Ako se pravo pristupa ne navede eksplicitno podrazumijeva se privatni pristup

Dozvola prava pristupa

```
class primjer {
     private:  //defaultno pravo pristupa
          //članovi klase
     protected:
          //članovi klase
     public:
          //članovi klase
};
```

Klase

- Sada imamo klasu:

```
class Date {
        int day;
        int month;
        int year;
    public:
        void set(int,int,int);
        void next();
        void print();
};
void Date::set(int d, int m, int y) {
        day = d;
        month = m;
        year = y;
}
```

specifikator pristupa `public` nalaže da članovi navedeni nakon njega smiju biti vidljivi bilo gdje izvan tijela klase

defaultni pristup članova kod `class` deklaracije je privatni (private)

`struct` je `class` kod kojeg svi članovi po defaultu imaju javni public pristup

Prava pristupa članovima klase

- unutar strukture možemo definirati vrstu pristupa (`private`, `protected` ili `public`) na isti način kao u klasi
- `struct` - defaultna vrsta pristupa je public
- `class` - defaultna vrsta pristupa je private
- funkcije članice također mogu biti `private`, `protected` ili `public`

Prava pristupa članovima klase

- Što smo time dobili?

```
class Date {
  private:
      int day;
      int month;
      int year;
  public:
      void set(int,int,int);
      void next();
      void print();
};
```

Podaci su privatni

Funkcije su javne

Sad više ne možemo napisati:

```
Date d;
d.day=32;    // greška
```