Razvoj softvera

vežbe 3

Lista

- Lista – dinamička struktura podataka koja omogućava da se više elemenata veže u sekvencu čija dužina nije unaperd poznata.
- Svaka lista je:
  - ili prazna lista;
  - ili se sastoji od jedne vrednosti elementa (tj. prvog elementa liste) i liste koja predstavlja njen nastavak.

Element liste

```
class ElementListe
{
  public:
     int Vrednost;
     ElementListe* Sledeci;
}
class Lista
{
  private:
     ElementListe* _Pocetak;
}
```

Život objekta

1. alokacija memorije
2. inicijalizacija
3. upotreba
4. deinicijalizacija
5. oslobađanje memorije

- konstruktor – obezbeđuje inicijalizaciju objekta pri pravljenju
- destruktor – obezbeđuje deinicijalizaciju objekta pri uklanjanju

Izrazi `new` i `delete`

- operator `new` - alocira memoriju za nove objekte
- operator `delete` – oslobađa memoriju pri uklanjanju objekata
- izraz `new` – alocira memoriju i inicijalizuje novi objekat (izvršava operator `new` pa zatim konstruktor)
- izraz `delete` – deinicijalizuje objekat i oslobađa memoriju (izvršava destruktor, pa operator `delete`)

Destruktor

- Destruktor – metod koji obezbeđuje neophodnu deinicijalizaciju objekata pre nego što se oni trajno uklone iz memorije.
- Svaka klasa ima tačno jedan destruktor (podrazumevani ili eksplicitno definisan).
- Podrazumevani destruktor se ponaša inverzno podrazumevanom konstruktoru: obezbeđuje da svi podaci objekta budu deinicijalizovani primenom destruktora za odgovarajuće tipove.
- `~ImeKlase()`
- nema rezultat ni argumente

Destruktor

- U telu se navode deinicijalizacije koje ne obavlja podrazumevani destruktor.
- U destruktoru je potrebno osloboditi alocirane resurse, tj. podatke tipova koji ne oslobađaju sami resurse na koje referišu.
- Pokazivači
- Ako objekat klase referiše na resurse koji bi nakon uništavanja objekta mogli ostati rezervisani i nedostupni, onda je takve resurse neophodno oslobađati u destruktoru klase.

Vrste objekata

- Statički objekti – prave se tokom učitavanja programa u memoriju, pre pozivanja funkcije `main`,a uništavaju pri izlasku iz funkcije `main`
- Automatski objekti (lokalno definisani) – prave se kada se pri izvršavanju koda dođe do definicije promenljive koja predstavlja novi objekat. Uništavaju se neposredno pre izlaska iz bloka u kome su definisani.
- Dinamički objekti – eksplicitno se prave i uklanjaju primenom izraza `new` i `delete`

```
int a(1);

main()
{
  …
  int b(2);
  int* c=new int(3);
  …
  delete c;
}
```

Kopiranje objekata

```
Lista l;
  for(int i=0; i<10; i++)
     l.dodajNaPocetak(i);
  …
Lista l1=l;
```

- Šta se dešava?

Kopiranje objekta

- Konstruktor kopije ili kopi-konstruktor – konstruktor čiji je argument objekat istog tipa

```
Lista l1=l;
Lista l1(l);
```

- Podrazumevani konstruktor kopije primenjuje konstruktore kopije za sve pojedinačne članove

Kopiranje objekata

- Plitko kopiranje – kopiraju se reference (ili pokazivači) na neke resurse, ali ne i sami resursi.
- Duboko kopiranje – kopiraju se čitavi resursi.
- Resurse je potrebno duboko kopirati ako i samo ako ih destruktor oslobađa.
- Problemi
  - resurs se oslobađa u destruktoru, a kopiranje je plitko -> višestruko oslobađanje resursa
  - resurs se ne oslobađa, a kopiranje je duboko -> curenje memorije
- `Lista(const Lista& l)`

Operator dodeljivanja

```
Lista l1= l;
…
l1=l;
```

- Implicitno definisan `operator =` izvodi plitko kopiranje.
- `void operator(const Lista& l)`
- Ako je za ispravno funkcionisanje klase neophodan neki od metoda: destruktor, konstruktor kopije ili operator dodeljivanja, tada su neophodna sva tri metoda.

- Zabrana upotrebe kopi-konstruktora i operatora dodeljivanja:

```
class Lista
{
    …
    private:
        Lista(const Lista& l);
        Lista& operator= (const Lista&
l);
}
```

- Metod može biti deklarisan, a da ne bude i definisan. Greška se prijavljuje samo ako se koristi.

operator []

- `int operator [] (unsigned i) const`
- `int& operator [] (unsigned i)`

Izuzeci

- podsistem za rad sa izuzecima obrađuje vandredne situacije
- prijavljivanje neispravnosti – izbacivanje izuzetka ili proglašavanje izuzetka
- obrada izbačenog izuzetka – hvataje izuzetka
- izuzetak može biti objekat bilo kog tipa

Izuzeci

- izbacivanje izuzetka

```
throw <izuzetak>;
```

- blok pokušaja – blok koda u kome se očekuju mogući izuzeci koji se mogu obraditi

```
try {
 <blok koda>
 }
<hvatac izuzetka>
```

Izuzeci

- hvatač izuzetka

```
catch(<argument>)
{
   <kod za obradu izuzetka>
}
```

- izuzetak se hvata i obrađuje od strane prvog navedenog hvatača izuzetaka koji može uhvatiti izuzetak odgovarajućeg tipa

Izuzeci

- univerzalni hvatač – može da uhvati izuzetke svakog tipa

```
catch(...)
{
   <kod za obradu izuzetka>
}
```

Izuzeci

- u telu hvatača izuzetka može se propustiti izuzetak koji se obrađuje navođenjem samo `throw` bez argumenta.
- Zaglavlje `stdexcept`
  - Neke klase:
    - `bad_alloc` – neuspešna alokacija
    - `invalid_argument` – neispravan argument
    - `out_of_range` - podatak van opsega
  - Sve klase imaju konstruktor sa argumentom const string&
  - Metod what