PROGRAMIRANJE II

PROGRAMSKI JEZIK C++

Predavanje 6

Statički članovi klase

Statički članovi

- Klase koje smo definisiali u dosadašnjim primjerima imale su zasebne skupove članova za svaki objekat koji je kreiran.
- U nekim slučajevima potrebno je dijeliti podatke između objekata iste klase.
- To se može ostvariti statičkim podacima članovima i statičkim funkcijama članicama.
- Ponekad je potrebno definisati podatak član koji će biti zajednički svim objektima (članovima) klase.
- To može, na primjer, biti:
  - Brojač koji broji Koliko je napravljeno objekata te klase.
  - Statusni član koji određuje ponašanje klase.
- Taj problem bi se klasičnim postupkom mogao riješiti tako što bi se deklarisala globalna promjenljiva kojoj pristupaju funkcijski članovi klase.
- Međutim, to rješenje nije elegantno, jer je globalna promjenljiva dostupona I izvan klase i pristup joj ne može biti privatan.

Statički članovi

- Rješenje problema daju statički članovi.
- U ovom slučaju promjenljiva se nalazi u području imena klase (ne u globalnom prostoru imena) i pristup joj je ograničen.

```
#include <iostream>
using namespace std;

class Brojanje {
public:
          static int Brojac;            // statički podatak član
          int MojBroj;

          Brojanje();
          ~Brojanje();
          void IspisiBrojace();
};

Brojanje::Brojanje() : MojBroj(Brojac++) {
          cout << "Kreiran element br.: " << MojBroj << endl;
}

Brojanje::~Brojanje() {
          cout << "Unisten element br.: " << MojBroj << endl;
}

int Brojanje::Brojac = 0;      //Inicijalizacija statičkog člana
```

Statički član

- Podatak člna `brojac` nije sadržan u svakom objektu klase nego je globalan za cijelu klasu.
- Statički podaci članovi se vežu za klasu, ne za objekat, jer je vrijednost statičkog podatka člana jedinstvena za sve objekte klase.
- Takav član se deklariše tako da se prije specifikacije tipa ubaci ključna riječ `static`.
- Za statičke članove vrijede standardna pravila pristupa.
- U prethodnom primjeru, podatak član `brojac` je deklarisan kao privatan i može mu se pristupiti isključivo unutar klase.

Statički članovi su zajednički za sve objekte neke klase. Oni su slični globalnim promjenljivim s tom razlikom što su njihova imena vidljiva ISKLJUČIVO u klasi.

Statički članovi

- Deklaracija statičkog člana unutar klase služi samo kao najava ostatku programa da će negdje u memoriji postojati jedan član koji će biti zajednički svim objektima klase.
- Sam član time nije smješten u memoriju.
- Inicijalizacija statičkog člana se mora eksplicitno obaviti izvan klase – tada se odvaja memorijski prostor i član se inicijalizuje.
- U prethodnom primjeru to je učinjeno naredbom:

```
int Brojanje::Brojac = 0;
```

- Ova naredba odvaja potreban prostor i inicijalizuje člna na nulu.
- Prilikom inicijalizacije izostavljena je ključna riječ `static` ispred oznake tipa – ona se navodi SAMO prilikom deklaracije unutar klase.

Statički članovi

- U programskok jeziku C++ definicija klase se često izdvaja u zasebnu datoteku zaglavlja koja se uključuje u sve segmente programa koji koriste tu klasu.
- Na taj način će definicija klase svim djelovima programa objaviti da u memoriji postoji jedan zaajednički član.
- U jednoj datoteci se mora naći i inicijalizacija tog člana ili će se u protivnom javiti greška prilikom povezivanja - kako član nije pronađen u memoriji.
- Definiciju klase `Brojanje` možemo smjetiti npr. u datoteku `broj.h`, a definiciju funkcijskih i statičkih članova u datoteku `broj.cpp`.
- Ako bi se inicijalizacija statičkih članova (također) smjestila u datoteku zaglavlja, prilikom svakog uključenja te datoteke pretprocesorskom naredbom `#include` kreirao bi se po jedan član i na braju bismo prilikom prevođenja (povezivanja) dobili poruku o grešci da je član `Brojanje::Brojac` višestruko definisan.

Pristup statičkom članu

- Funkcije članice klase statičkom članu pristupaju na isti način kao i običnom članu klase.
- Ukoliko statički član ima javno pravo pristupa, može mu se pristupiti i iz ostalih djelova programa i to:
  1. Prvi način - pomoću objekta: navede se objekat, zatim operator . (tačka) pa onda naziv člana. Tada sam objekat služi isključivo za identifikaciju klase u kojoj je podatak član definisan.
  2. Drugi način – pristup bez objekta, navodeći isključivo klasu. Ispred samog člana se mora nevesti naziv klase i operator :: (dvije dvotačke) da bi se naznačilo u kojoj klasi se nalazi član kojem se pristupa.
- Opšti oblik sintakse za pristup članovima je:

```
naziv_klase :: ime_člana
```

Primjer poziva statičkog člana

```
#include <iostream>
using namespace std;

class Brojanje {
public:
          static int Brojac;            // statički podatak član
          int MojBroj;
          void IspisiBrojace();
};

void Brojanje::IspisiBrojace() {
          cout << "Brojac: " << Brojac << endl;   // ovo je pristup iznutra
          cout << "Moj broj: " << MojBroj << endl;
}

int Brojanje::Brojac = 0;      //Inicijalizacija statičkog člana
int main() {
                    Brojanje bb;
                    Brojanje::Brojac = 5;     // pristup izvana operatorom ::
                    bb.Brojac = 5;           // isto kao i prethodna naredba
                    bb.IspisiBrojace();
          return 0;
}
```

Pristup statičkim članovima

- Javnim statičkim članovima se može pristupiti direktno (navodeći cijelo ime člana) jer za cijelu klasu postoji samo jedan statički član.
- Običnim članovima podacima se ne može direktno pristupiti. U sledećem primjeru pridruživanje (inicijalizacija) nije ispravno.

```
Brojanje::MojBroj = 5;
```

- Ova naredba izaziva grešku u prevođenju jer `MojBroj` postoji samo u kontekstu pojedinog objekta pa mu se može pristupiti samo preko imena objekta.

```
bb.MojBroj = 5;
```

- Dok je za pristup statičkom podatku članu `Brojac` dozvoljeno napisati i

```
bb.Brojac = 5;
```

- Objekat `bb` sada prevodiocu samo kazuje naziv klase u kojoj je statički podatak član smješten. Iako gornja naredba sadrži ime objekta, ona mu ništa ne pridružuje već pristupa statičkom članu.

Pristup statičkim članovima

- Inicijalizaciju (kao i u prethodnom primjeru) statičkog podatka člana neizostavno treba izvršiti ispred koda glavne funkcije.
- Nekome (početnicima) ta inicijalizacija može izgledati suvišna, jer se slično pridruživanje obavlja i unutar glavne funkcije.
- Međutim, ta dva pridruživanja treba razlikovati: inicijalizacija ispred glavne funkcije obavlja se prilikom prevođenja programa, dok se pridruživanje u tijelu glavne funkcije obavlja u toku izvršavanja programa.
- Ako se izostavi inicijalizacija, prilikom povezivanja programa (linkovanje) prevodilac neće naći statički član `Brojanje::Brojac` i prijaviće grešku.

Statički objekat

- Statički podaci članovi se dodatno razlikuju od običnih članova po tome što klasa može sadržavati statički objekat iste klase.
- U slučaju običnih članova, klasa može sadržavati samo pokazivače i reference na objekte iste klase, na primjer:

```
class Obj {
    static Obj statickyClan; //Ovo je OK je clan staticky
    Obj& ref; //i ovo je OK jer su članovi referenca i pokazivač
    Obj* pok;
    Obj nestatickiClan; //ovo nije OK
};
```

- Statički članovi mogu biti navedeni kao podrazumijevani argumenti funkcijskim članovima, dok nestatički ne mogu, kao u primjeru:

```
int a;
class Param {
private:
    int a;
    static int b;
public:
    void Func1(int = b);  //ovo je OK jer je član statički
    void Func2(int ::=a); //i ovo je OK jer se odnosi na globalno a
    void Func3(int = a);  //GREŠKA jer se odnosi na nestatički član
};
```

Statički funkcijski članovi

- Postoje funkcije članice koje pristupaju samo statičkim članovima klase.
- Funkcije članice se u tom slučaju mogu realizovati kao i obične, međutim, u tom slučaju treba kreirati objekat da bi se pristupilo statičkom članu.
- Ovo je nepraktično, jer kreiramo objekat (koji u suštini nije ni potreban).
- Funkcija koju pozivamo pristupa isključivo statičkim članovima i ne mijenja niti jedan nestatički član objekta.
- Takva se funkcija može deklarisati kao statička fukcija klase tako što se ispred povratnog tipa funkcije stavi ključna riječ `static`.
- Za poziv takve funkcije nije potrebno imati objekat, nego se jednostavno navodi puno ime funkcije.

Statičke funkcije

- Da bismo ilustrovali concept statičkih funkcija, dodajmo klasi `Brojanje` statičku funkciju `VrijednostBrojaca()`, koja vraća vrijednost statičkog člana `Brojac`.

```
#include <iostream>
using namespace std;

class Brojanje {
public:
          static int Brojac;              // statički podatak član
          int MojBroj;
          // …
          static int VrijednostBrojaca() { return Brojac; }
};
          //…
int Brojanje::Brojac = 0;       //Inicijalizacija statičkog člana

int main() {
                    Brojeni::Brojac = 5;      // pristup izvana operatorom ::
                    cout << Brojanje::VrijednostBrojača() <<endl;
          return 0;
}
```

- Iz primjera se vidi da prilikom poziva statičke funkcije nije potrebno navoditi objekat pomoću kojeg se funkcija poziva, nego se jednostavno navede cijelo ime funkcije.

Statičke funkcije

- Važno je uočiti da statičke funkcije, slično kao i statički podaci članovi imaju puno ime oblika *naziv_klase::ime_funkcije*.
- Zato je potrebno prilikom referenciranja na funkciju navesti ime klase.
- Za statičke članove klase važe ista pravila pristupa, pa ako funkciju želimo pozivati van objekta klase, ona mora imati javni pristup.

Statički funkcijski članovi se mogu pozivati bez objekta, navodeći puno ime funkcije. Kompletno ime obuhvata naziv klase, odvojen operatorom :: od naziva člana.

- Statičke funkcije članice se, kao i statički podaci članovi mogu pozivati i pomoću objekta klase. Sledeći poziv je potpuno ispravan (uz moguće upozorenje prevodioca da objekat `bb` nigdje nije korišćen)

```
int main() {
     Brojanje bb;
     Brojanje::brojac = 5;
     cout >> bb.VrijednostBrojaca() << endl;
     return 0;
}
```

Statičke funkcije

- Pošto se statičke funkcije ne pozivaju pomoću objekta, iz tog razloga one i ne sadrže implicitni `this` pokazivač.
- Svaka upotreba te ključne riječi u statičkoj funkciji će rezultirati pojavom grešeke prilikom prevođenja programa.
- Kako nema pokazivača this, jasno je da i pokušaj pristupa bilo kojem nestatičkom podatku članu ili funkciji članici klase rezultira greškom prilikom prevođenja, kao na primjer:

```
//greška prilikom prevođenja
static int VrijednostMojMojBroj() {return mojBroj;}
```

- Definicija statičkog funkcijskog člana koji bi trebalo da vrati vrijednost nestatičkog člana. GREŠKA.

Ključna riječ `static` se navodi samo ispred deklaracije statičkog člana.

Statičke funkcije

- Da smo u primjeru klase Brojanje definiciju statičke funkcije članice izvuklji izvan tijela klase, tada bismo morali pisati:

```
class Brojanje {
public:
        static int VrijednostBrojaca(); // statička funkcija članca
};
//nema riječi static ispred definicije
int Brojanje::VrijednostBrojaca(){
        return Brojač;
}
```

- Statičke fukkcije članice ne mogu biti deklarisane kao const ili volatile. Također, dozvoljeno je koristiti pokazivače na statičke podatke članove i statičke funkcije članice kao u primjeru

```
//…
int* pok = &Brojanje::Brojac;
Int (*pokNaFunkciju)() = Brojanje::VrijednostBrojača;
```