Lokalne i globalne promenljive
predavač: Nadežda Jakšić

Programiranje programski jezik C

Opseg vidljivosti (scope)

- opseg vidljivosti promenljivih predstavlja programski blok u kom kreirana promenljiva "postoji ", što znači da joj se u tom bloku može pristupati i da joj se može menjati vrednost; izvan tog bloka promenljiva ne postoji

```
int prostBroj (int broj)
{ int i, prost=1;
 for (i=2;i<=broj/2;i++)
     {
     if (broj%i==0)
           {prost=0; break;}
      }
      return prost;
}
```

u funkciji **prostBroj** su deklarisane dve lokalne promenljive tipa **int**, to su **i** i **prost** njihova vidljivost je na nivou funkcije **prostBroj**, one izvan funkcije ne postoje; u funkciji **main** može biti takođe definisana promenljiva **i**, ali ova promenljiva **i** nema nikakve veze sa promenljivom **i** u funkciji **prostBroj**, jer funkcije **main** i **prostBroj** predstavljaju dva različita opsega vidljivosti promenljivih

Vrste promenljivih

na osnovu opsega vidlijivosti promenljive se razvrstavaju u tri grupe:

- promenljive definisane u bloku
- interne ili lokalne promenljive
- eksterne ili globalne promenljive

Promenljive definisane u bloku

- to su promenljive koje su definisane u jednom bloku naredbi koji je označen vitičastim zagradama i može se odnositi na neku od naredbi grananje (**if**, **switch**) ili naredbi ponavljanja, a može biti i samostalni blok označen vitičastim zagradama (ovde jedino ne ubrajamo blok naredbi koji predstavlja implementaciju funkcije)
- ova vrsta promenljivih vidljiva je samo u bloku u kom je definisana, kao i u svim unutrašnjim blokovima

```
int main()
{
    {    //spoljašnji blok
    int x=10;
        {   //unutrašnji blok
        printf ("x=%d",x);
        }
printf ("\nx=%d",x);
   }
return 0;
}
```

Promenljive definisane u bloku

- u sledećem primeru promenljivoj **x** se pristupa izvan bloka u kom je definisana

```
int main()
{
 if (10<20)
    {
        int x=10;
        printf ("x=%d",x); //ovde se može pristupiti promenljivoj x
    }
printf ("\nx=%d",x); //ovde se ne može pristupiti promenljivoj x
return 0;
}
```

- programski kôd će izazvati grešku prilikom kompajliranja, a poruka će glasiti: **eerror: 'x' undeclared (first use in this function)**
- ova poruka znači da kompajler na tom mestu ne vidi promenljivu **x**, odnosno ne vidi da je ona definisana

Promenljive definisane u bloku

- postoji mogućnost da se promenljiva redefiniše u unutrašnjem bloku

```
int main()
{
int x=40;
if (10<20)
    {
     float x=10.25; //redefinisanje promenlj.
     printf ("x=%f",x);
     }
 printf ("\nx=%d",x); //pristup promenljivoj
 return 0;                iz spoljašnjeg bloka
 }
```

promenljiva **x** je definisana jednom u spoljašnjem bloku kao tip **int**, a zatim redefinisana u unutrašnjem kao tip **float** (može biti i isti tip kao u spoljašnjem bloku); ovo je situacija kada je promenljiva iz spoljašnjeg bloka "sakrivena" u unutrašnjem bloku i umesto nje se koristi druga promeljiva sa istim imenom

Lokalne promenljive

- to su promenljive koje su definisane u telu funkcije; opseg vidljivosti ovih promenljivih ograničen je na funkciju u kojoj su definisane; promenljivoj koja je definisana u jednoj funkciji **ne može** se pristupiti iz druge funkcije

```
void prikazi();
int main()
{
    int x=10;
    prikazi();
    return 0;
}
void prikazi()
{
    printf ("x=%d",x);
}
```

u funkciji **main** definisana je promenljiva **x**, i u funkciji **prikazi()** pristupamo promenljivoj **x**, međutim **x** u funkciji **prikazi()** nema nikakve veze sa promenljivom **x** iz funkcije **main** i prilikom kompajliranja ovog programa dobićemo grešku da promenljiva **x** nije definisana u funkciji **prikazi**

Lokalne promenljive

- ako se u bilo koje dve funkcije kreira promenljiva sa istim imenom i jedna funkcija poziva drugu, ove dve promenljive su potpuno nezavisne; lokalne promenljive se kreiraju u momentu kad se pozove funkcija (**alokacija memorije**), a prestaju da postoje kada se završi izvršavanje funkcije (**dealokacija memorije**); ne čuvaju vrednost za sledeće pozive iste funkcije

```
void prikazi ()
   { int x=10;
     x=x+10;
     printf ("%d\n",x); }
int main()
{   prikazi ();
    prikazi ();
    prikazi ();
    return 0; }
```

u ispisu ovog programa, uvek će biti ista vrednost promenljive **x**, što znači da se pri svakom pozivu funkcije **prikazi** kreira nova promenljiva **x**

za promenljive koje su definisane u funkciji **main** (direktno u bloku funkcije, ne u unutrašnjem bloku), memorija se alocira na početku programa a dealocira po završetku rada programa

Lokalne promenljive

```
float nadjiProsek (int n, float niz[])
   {int i;
     float s=0;
     for (i=0;i<n;i++)
     s+=niz[i];
     return (s/n);}
int main ()
{ int i,n;
     float niz[10];
     printf ("Koliko elemenata ima niz?\n");
     scanf ("%d",&n);
     for (i=0;i<n;i++)
           {printf ("\nUnesite %d. element niza  ", i+1);
            scanf ("%f",&niz[i]);}
printf ("Aritmeticka sredina %.2f", nadjiProsek (n,niz));
```

Prosleđivanje parametara

```
void increment (int x)
{
  x = x+1;
  return;
}

int main()
{
int x = 10;
printf ("x pre funkcije %d\n", x);
increment (x);
printf ("x posle funkcije %d", x);
return 0;
}
```

funkcija **main** poziva funkciju **increment** i vrši ispis vrednosti promenljive pre i posle poziva funkcije; vrednost promenljive **x** je ostala nepromenjena; to je prosleđivanje **"po vrednosti"**, kao parametar funkcije prosleđuje se samo vrednost promenljive, a ne i promenljiva na određenoj memorijskoj lokaciji; prosleđivanjem **"po referenci”** bi se prosledila memorijska lokacija i vrednost promenljive bi se promenila po izlasku iz funkcije (simulacija koncepta se realizuje preko pokazivača u **C**-u, a koncept je podržan u većini objektno orijentisanih jezika

Prosleđivanje parametara

```
int increment (int x)
{
  x = x+1;
  return x; //funkcija vraća vrednost kopije koju je napravila u svom steku
}
int main()
{
   int x = 10;
   printf ("x pre funkcije %d\n", x);
   printf ("x posle funkcije %d", increment (x));
return 0;
}
```

Zamena ne radi

```
void zamena (int a, int b)
   {
       int pom;
       pom=a;
       a=b;
       b=pom;
       return;
   }
   int main ()
   {    int br1, br2;
        printf ("unesite dva broja\n");
        scanf ("%d%d",&br1,&br2);
        zamena (br1,br2);
        printf ("zamenjeni brojevi su: %d    %d", br1, br2);
    return 0;}
```

Zamena radi

```
void zamena (int a, int b)
   {
      int pom;
      pom=a;
      a=b;
      b=pom;
      printf ("zamenjeni brojevi su: %d    %d", a, b);
      return;    }
   int main ()
   {
         int br1, br2;
         printf ("unesite dva broja\n");
         scanf ("%d%d",&br1,&br2);
         zamena (br1,br2);
    return 0;}
```

Zamena pokazivači

```
void zamena (int *a, int *b)
{
    int pom;
    pom=*a;
    *a=*b;
    *b=pom;
return;
}
int main ()
{
      int br1, br2;
      printf ("unesite dva broja\n");
      scanf ("%d%d",&br1,&br2);
      zamena (&br1,&br2);
      printf ("zamenjeni brojevi su: %d    %d", br1, br2);  return 0;}
```

Globalne promenljive

- definišu se izvan tela funkcije

```
#include<stdio.h>
int x=10; //globalna promenljiva
void povecaj ();
int main ()
{
   printf ("x=%d",x); //prikaz 1
   povecaj ();
   printf ("\nx=%d",x); //prikaz 2
return 0;
}
void povecaj ()
{
   x=x+50; //promena vrednosti
}                globalne promenljive
```

pre funkcije **main** definisana je globalna promenljiva **x** tipa **int**, koja se koristi u funkciji **main** za ispis, a u funkciji **povecaj** joj se menja vrednost;
pri prikazu 1 dobija se vrednost promenljive **x 10**, a u prikazu 2 dobija se **60**, što znači da je promena napravljena u funkciji **povećaj** ostala sačuvana u globalnoj promenljivoj **x** i posle izvršavanja funkcije

Globalne promenljive

- vidljive u svim funkcijama koje su navedne ispod definicije promenljive u programskom kôdu; upotrebljavaju se kada je neophodno pristupati promenljivoj u više funkcija i ako ne želimo da koristimo parametre funkcija
- korišćenje globalnih promenljivih može imati određene negativne posledice i zahteva opreznost - veliki broj funkcija može da menja vrednost promenljivoj; zbog toga je u velikim programima teško ispratiti stanje neke globalne promenljive; fenomen promene vrednosti neke globalne promenljive u funkcijama naziva se sporedni efekat funkcije (**side effect**)
- ako se u nekoj funkciji definiše promenljiva sa istim imenom kao globalna promenljiva, u toj funkciji se koristi vrednost lokalne promenljive