Osnovi programiranja

2023/24

Funkcije

Potprogram

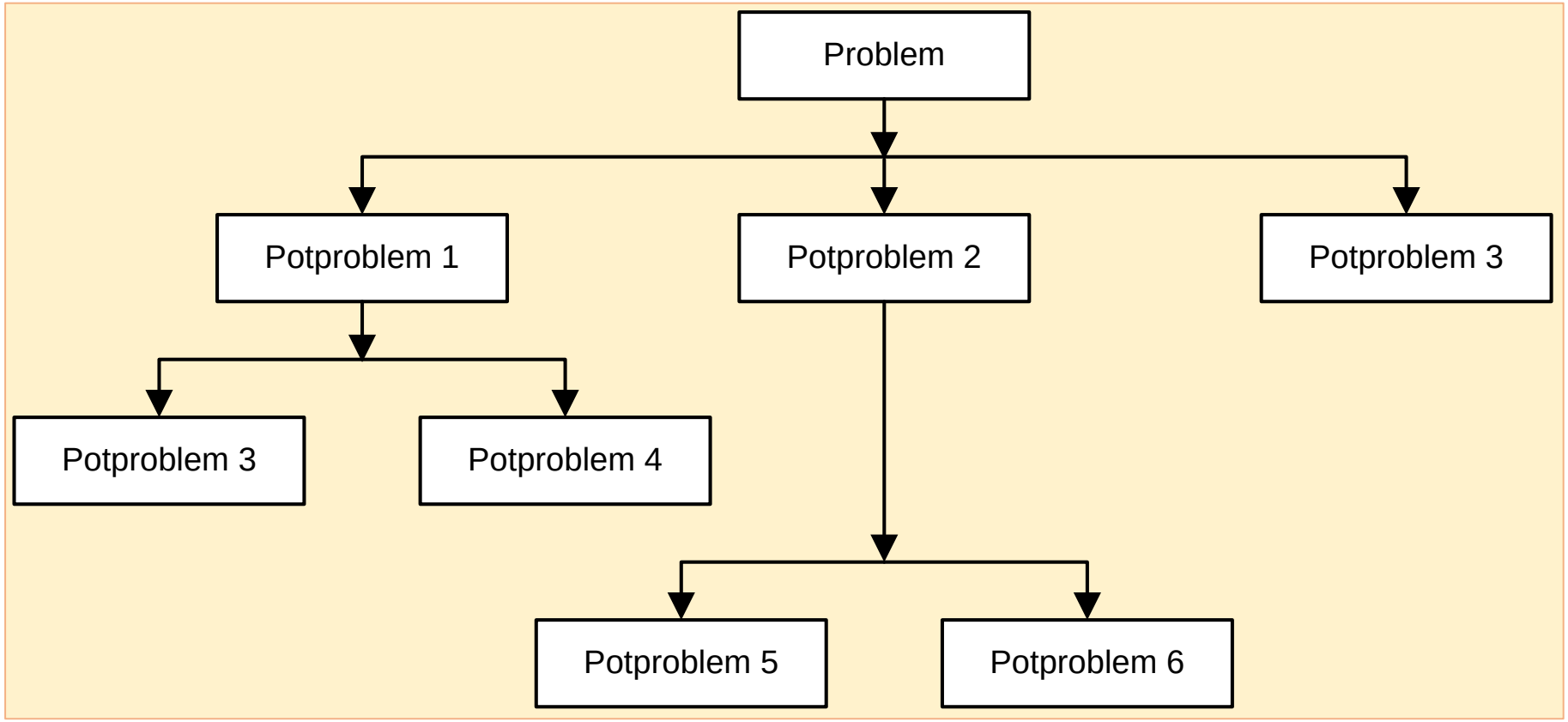
- Složeniji programi se mogu podeliti na odgovarajuće logičke i funkcionalne celine, takozvane `potprograme`.
- Potprogram omogućava organizovanje određenog broja instrukcija u celine koje obavljaju određene zadatak.
- Potprogram preuzima ulazne veličine i vrši njihovu obradu u cilju dobijanja određenih rezultata
- Potprogram možemo pozivati više puta u toku izvršavanja programa.

Prednosti korišćenja potprograma

- Omogućava koncentrisanje na izvršavanje samo određenog zadatka
- Različiti članovi razvojnog tima mogu razvijati različite potprograme koji se kasnije sklapaju u jednu celinu
- Omogućava izvršavanje istog zadatka na više mesta u programu jednostavnim pozivanjem odgovarajućeg potprograma

Programiranje “odozgo na dole”

- Funkcije i procedure omogućavaju realizaciju programiranja "odozgo na dole". Ovaj metod podrazumeva razbijanje problema na potprobleme



Funkcije

- Funkcija je samostalan deo programa koji obavlja određeni zadatak
- Preko svog naziva i liste parametara, delu programa iz koga je pozvana vraća odgovarajuće rezultate
- Svaka funkcija ima jedinstveni naziv preko koga se može pozvati proizvoljan broj puta iz bilo kog dela programa u cilju izvršenja tog zadatka.

Funkcije

- Sintaksa

```
<tip_funkcije> <naziv>([lista parametara])
<blok_naredbi>
```

- Svaka funkcija mora imati zaglavlje, koje sadrži jedinstveni naziv naveden odmah iza rezervisane reči tipa funkcije. Iza naziva funkcije sledi lista parametara unutar oblih zagrada.
- Za svaki parametar funkcije mora biti naveden i njegov tip, na sličan način kao i kod deklaracija promenljivih.
- Ukoliko funkcija ne zahteva nikakve ulazne podatke, moguće je ne navesti nijedan parametar.

Funkcije

- Nakon definisanja zaglavlja funkcije sledi blok naredbi, odnosno telo funkcije. Kao i svaki drugi blok naredbi, telo funkcije se sastoji od niza komandi ograničenih rezervisanim znakovima **{** i **}**.
- U slučaju da funkcija treba da vrati neku vrednost, neophodno je na kraju tela funkcije pomoću rezervisane reči **return** naglasiti koju vrednost funkcija vraća.

```
return <promenljiva>;
```

- Svaka funkcija može biti pozvana iz glavnog programa (glavne funkcije) ili druge funkcije navođenjem njegovog imena i liste parametara unutar oblih zagrada. Prilikom poziva funkcija potrebno je vrednost koju funkcija vraća dodeliti odgovarajućoj promenljivoj korišćenjem operatora dodele, ili je direktno iskoristiti unutar nekog izraza ili naredbe.

Primer 1

- Napisati program koji izračunava i štampa rastojanje između dve tačke čije su coordinate date na ulazu.

```
#include <stdio.h>
#include <math.h>
float rast(float x1, float y1, float x2, float y2)
{
	return sqrt(pow(x2-x1,2) + pow(y2-y1,2));
}
void stampaj(float d)
{
	printf("Rastojanje izmedju tacaka je %10.4f\n", d);
}
```

Primer 1

```
main()
{
   float x1, y1, x2, y2, r;

   printf("Unesite koordinate prve tacke:\n");
   scanf("%f%f", &x1, &y1);

   printf("Unesite koordinate druge tacke:\n");
   scanf("%f%f", &x2, &y2);

   r = rast(x1, y1, x2, y2);
   stampaj(r);
}
```

Primer 1

- Prva funkcija na osnovu koordinata dve tačke izračunava rastojanje između njih. Druga funkcija vrši štampanje rezultata uz odgovarajući komentar.
- Druga funkcija nema povratnu vrednost, tako da je tip funkcije **void**.

Primer 2

- Napisati program koji za n vrednosti sa ulaza vrši izračunavanje polinoma $x^5 + 3x^4 - 6x^2 + 2$, bez korišćenja biblioteke sa matematičkim funkcijama.

```
#include <stdio.h>
float stepen(float x, int eksp)
{
    int i;
    float s = 1.0;
    for(i = 0; i < eksp; i++)
        s = s*x;
    return s;
}
float poli(float x)
{
        return stepen(x,5)+3.0*stepen(x,4)-6.0*stepen(x,2)+2;
}
```

Primer 2

```
main()
{
    float x;
    int n,i;
    printf("Unesite koliko brojeva zelite:\n");
    scanf("%d", &n);
    for(i = 0; i < n; i++)
    {
        printf("Unesite %d. broj\n", i + 1);
        scanf("%f", &x);
        printf("Vrednost polinoma za %d. broj je %10.4f\n", i, poli(x));
    }
}
```

Deklaracija funkcije

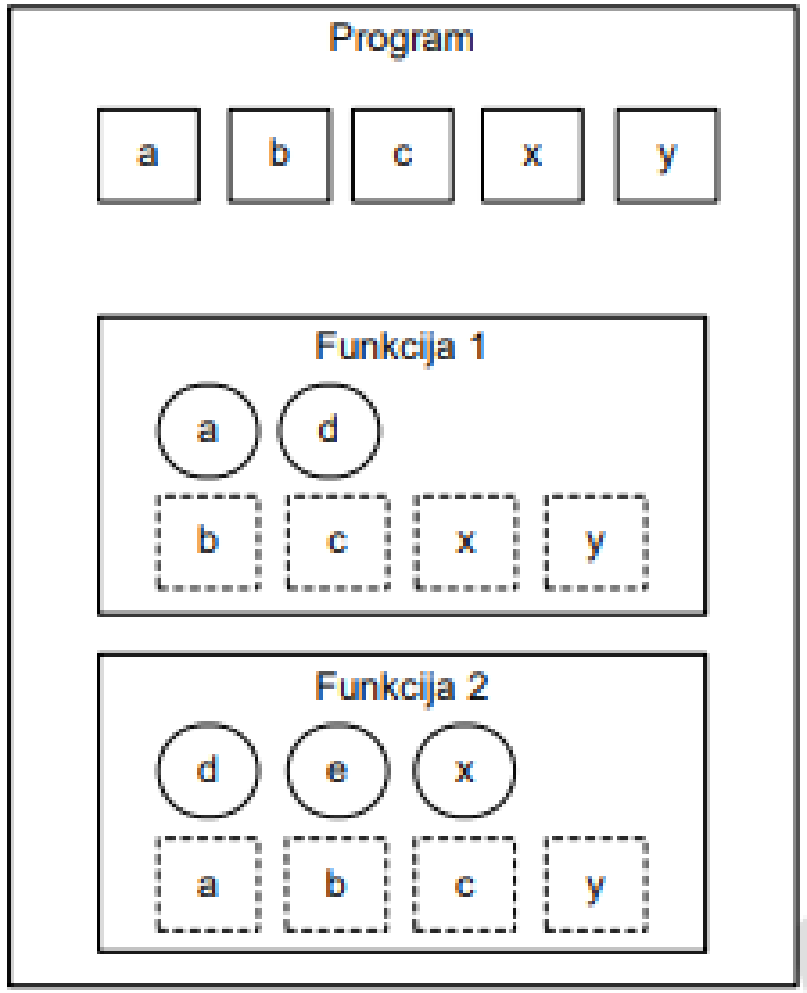
- Da bi određena funkcija mogla da se pozove iz neke druge funkcije, neophodno je da funkcija koja se poziva bude definisana pre funkcije koja je poziva.
- Ukoliko raspored funkcija nije takav da je ovaj zahtev obezbeđen, moguće je deklarisati funkciju kako bi se naznačilo da će definicija funkcije koja se poziva uslediti kasnije u kodu. To se postiže tako što se ispred funkcije koja poziva navede samo zaglavlje funkcije koja se poziva, a sama funkcija koja se poziva se navodi kasnije u kodu.

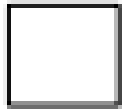
```
int f2(...);
void f1(...)
{
    ...
    f2(...);
    ...
}
int f2(...)
{
    ...
}
```

Opseg važenja i životni vek promenljivih

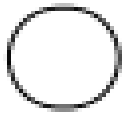
- **Opseg važenja** neke promenljive je deo programa u kome ta promenljiva može biti korišćena. U tom delu programa data promenljiva ima značenje i za nju kažemo da je "vidljiva" u tom opsegu.
- Na primer, promenljiva deklarisana unutar funkcije ima značenje i vidljiva je samo unutar te funkcije. Takvu promenljivu nazivamo **lokalna promenljiva**. Sa druge strane, **globalna promenljiva** se deklariše van svih funkcija i vidljiva je iz bilo kog dela programa.

Opseg važenja i životni vek promenljivih





Globalna promenljiva



Lokalna promenljiva

Globalna promenljiva vidljiva u funkciji

Primer 3

```
#include <stdio.h>
int a, b, c;
float x, y;

void Funkcija1()
{
   int a, d;

   a = 11;
   d = c + 12;
   b = 13;
   x = 3.14;
}
```

```
int Funkcija2()
{
   int d, e, x;

   x = 5;
   d = a+x;
   a = d+1;

   return -1;
}
```

Primer 3

```
main()
{
   a = 1;
   b = 2;
   c = 3;
   x = 4.0;
   y = 5.0;
   Funkcija1(); /* Nakon izvrsenja ove linije vrednosti
   promenljivih su: a=1, b=13, c=3, d nije vidljivo, e nije
   vidljivo, x=3.14, y=5.0 */
   Funkcija2(); /* Nakon izvrsenja ove linije vrednosti
   promenljivih su: a=7, b=13, c=3, d nije vidljivo, e nije
   vidljivo, x=3.14, y=5.0 */
}
```

Prenos parametara

- Izvršavanje svakog programa u programskom jeziku C započinje izvršavanjem funkcije **main**. Iz tog razloga funkciju **main** često nazivamo **glavni program**, kao što je uobičajeno u nekim drugim programskim jezicima.
- Svaka funkcija u svom zaglavlju imaju definisanu listu parametara koje zahteva od programa ili funkcije koja ih poziva. Ove parametre nazivamo **formalni parametri**. U većini programskih jezika postoje dve vrste formalnih parametara:
  - vrednosni
  - promenljivi (varijabilni).
- U programskom jeziku C se promenljivi parametri realizuju korišćenjem pokazivačkih promenljivih, o čemu će biti reči u višim kursevima o programskom jeziku C. U okviru ovog kursa se vrednosni i promenljivi parametri razmatraju na nivou koji je uobičajen za većinu programskih jezika, bez ulaženja u mehanizme koji stoje iza toga.

Prenos parametara

- Kada glavni program ili neka funkcija pozove neku funkciju, on joj prosleđuje potrebne vrednosti koje nazivamo stvarni parametri.
- Lista stvarnih parametara mora da odgovara listi formalnih parametara po broju, tipu i redosledu.
- To znači da glavni program mora funkciji da prosledi tačno onoliko parametara koliko funkcija zahteva, pri čemu parametri moraju da budu istog tipa i navedeni u potpuno istom redosledu kao što su navedeni u zaglavlju funkcije.

Prenos parametara