FUNKCIJE I POTPROGRAMI

Do sada smo koristili "gotove" funkcije iz standardnih biblioteka. Sve ove funkcije smo pozivali iz glavne funkcije našeg programa - main() funkcije, koja predstavlja specijalnu funkciju, jer izvršavanje programa počinje od nje i sve programe (bez obzira na njihovu dužinu) smo pisali u toj jednoj (main) funkciji.

Veliki računski zadaci mogu se razbiti u manje delove i time se omogućava ljudima da iskoriste ono što su neki drugi već uradili, umesto da počinju sve od početka. Odgovarajuće funkcije skrivaju detalje postupka od delova programa i time čine ceo program jasnijim i jednostavnijim za menjanje.

Napisati funkciju koja računa zbir dva cela broja i program koji testira rad ove funkcije.

```
#include <stdio.h>

/* Definicija funkcije */
int zbir (int a, int b) {return a+b;}

int main() {

/* Poziv funkcije */
printf("%d\n", zbir(3,5));

return 0;

}
```

C++ rešenje

```
#include <iostream>
using namespace std;
int zbir (int a, int b)
 {return a+b;}

int main()
{
   cout << zbir(3,5) << endl;
   return 0;
}
```

Uraditi isti primer, u kome se demonstrira razlika između definicije i deklaracije funkcije
Deklaracija funkcije može da stoji nezavisno od definicije funkcije.
Deklaracija je neophodna u situacijama kada se definicija funkcije navodi nakon upotrebe date funkcije u kodu.

```
#include <stdio.h>

/* Deklaracija funkcije zbir() */
int zbir(int, int);          /* int zbir(int a, int b); */

int main() {

/* Poziv funkcije */
printf("%d\n", zbir(3,5));

return 0;
}

/* Definicija funkcije */
int zbir(int a, int b) {return a+b;}
```

Hajde da ponovo pogledamo primer iz lekcije "Unos podataka" tj. Program koji komunicira sa

korisnikom na sledeći način:

```
using namespace std;
#include <iostream>
int main()
{
    int zbir = 0;
    float prosek;
    const int brPredmeta = 11;
    int ocena;

    cout << "Sta imas iz srpskog?\n";
    cin >> ocena;
    zbir += ocena;

    cout << "Sta imas iz matematike?\n";
    cin >> ocena;
    zbir += ocena;

    cout << "Sta imas iz fizickog?\n";
    cin >> ocena;
    zbir += ocena;

    cout << "Sta imas iz engleskog?\n";
    cin >> ocena;
    zbir += ocena;

    cout << "Sta imas iz fizike?\n";
    cin >> ocena;
    zbir += ocena;

    cout << "Sta imas iz hemije?\n";
    cin >> ocena;
    zbir += ocena;

    cout << "Sta imas iz ruskog?\n";
    cin >> ocena;
```

```
    zbir += ocena;

    cout << "Sta imas iz muzickog?\n";
    cin >> ocena;
    zbir += ocena;

    cout << "Sta imas iz likovnog?\n";
    cin >> ocena;
    zbir += ocena;

    cout << "Sta imas iz kung-fu-a?\n";
    cin >> ocena;
    zbir += ocena;

    cout << "Sta imas iz drugarstva?\n";
    cin >> ocena;
    zbir += ocena;

    prosek = (float) zbir / brPredmeta;
    cout << "Tvoj prosek je:" << prosek;
}
```

Kao što vidimo dobili smo prilično dugačak program u kome ima dosta ponavljanja sličnih ili istih operacija i blokova naredbi. Kako možemo da poboljšamo ovaj program i napišemo ga na kraći i jednostavniji način? Tu nam pomažu funkcije i na primeru ispod je pokazano kako se piše jedna jednostavna funkcija. Ova funkcija nas podseća na main funkciju iz naših prethodnih programa.

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

int StaImasIz(std::string imePredmeta)
{
    int ocena;

    cout << "Sta imas iz " << imePredmeta << "?\n";
    cin >> ocena;

    return ocena;
```

```
}

int main()
{
    int zbir = 0;
    float prosek;
    const int brPredmeta = 11;

    zbir += StaImasIz("srpskog");
    zbir += StaImasIz("matematike");
    zbir += StaImasIz("fizickog");
    zbir += StaImasIz("engleskog");
    zbir += StaImasIz("fizike");
    zbir += StaImasIz("hemije");
    zbir += StaImasIz("ruskog");
    zbir += StaImasIz("muzickog");
    zbir += StaImasIz("likovnog");
    zbir += StaImasIz("kung-fu-a");
    zbir += StaImasIz("drugarstva");

    prosek = (float) zbir / brPredmeta;

    cout << "Tvoj prosek je:" << prosek;
}
```

Kao što vidite dobili smo značajno kraći i pregledniji program u kome smo deo programa stavili u odvojenu funkciju StaImasIz(). Inače telo funkcije može biti veoma složeno za razliku od ovog primera. Posle kreiranja funkcije možemo je iz ostalog dela programa pozivati po imenu a funkcije mogu da pozivaju i jedne druge, čak funkcija može da poziva samu sebe (to se koristi u rekurzivnim algoritmima).

Za sad, nemojte kreirati funkciju koja poziva samu sebe.

Većina komandi programskog jezika C++ predstavljaju funkcije. Na primer “sqrt” je funkcija koja izračunava kvadratni koren prosleđenog joj broja. Ove funkcije su definisane u fajlovima C++ biblioteke koji su uključeni u C++ prevodilac (eng. compiler). Kada koristimo C++ funkciju, kompajler dodaje njeno telo vašem programu i kreira krajnji izvršni program (EXE, a.out,...).

Zaključak tj. navedimo razloge za upotrebu potprograma(funkcija):

- Kada imamo više ponavljanja istih blokova naredbi (operacija) u različitim delovima programa

(time skraćujemo tekst programa)

- Upotrebom funkcija, program se deli na odvojene blokove. Svaki blok radi određeni posao. Ovako olakšavamo čitljivost/razumevanje programa. Osmišljavanje, razumevanje i održavanje manjih blokova koda je lakše.

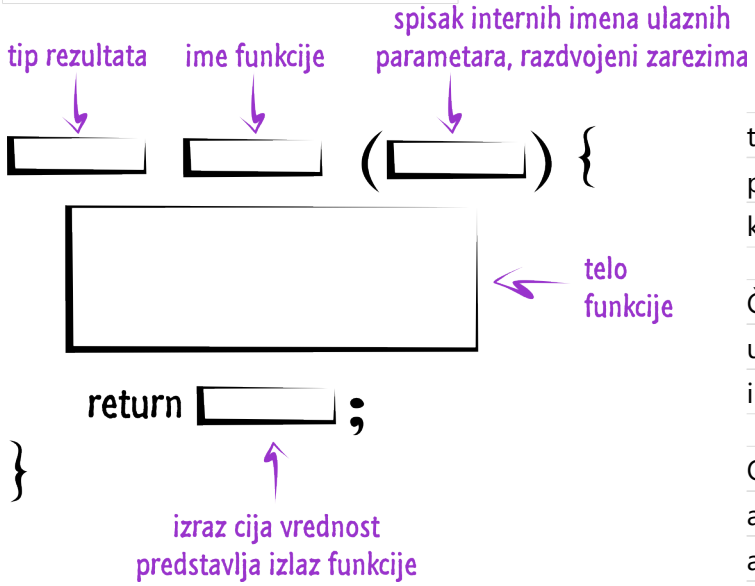
- Kada se deo programa moze izvršiti sa različitim parametrima. Ovi parametri se prosleđuju funkciji kao argumenti funkcije.

Definisanje funkcije

Funkcija se definše na sledeći način:

```
tip_rezultata ime_funkcije(opis argumenata funkcije)
{
    /*telo funkcije*/
}
```

ili prikazano na sledećem dijagramu:



tip_rezultata je neki od već poznatih tipova podataka i on označava kakav je rezultat koji funkcija vraća.

Često se kaže i "tip funkcije", a tada se misli upravo na tip njenog rezultata.
ime_funkcije je identifikator funkcije

Opis argumenata funkcije je lista argumenata funkcije koja sadrži tipove argumenata i identifikatore argumenata.
Npr. funkcija Izracunaj(int a, char ch, float x)
U listi su upisana tri argumenta i to: a, ch i x čiji su tipovi podataka navedeni ispred.

Telo funkcije je niz naredbi koje počinju sa deklaracijama promenljivih kao u programu tj. funkciji main().

U telu funkcije naredba za vraćanje rezultata koji funkcija daje je: return rezultat;

Svaka funkcija se sastoji od skupa naredbi koje određuju šta i kako funkcija radi. Taj skup naredbi čini telo funkcije. Telo funkcije započinje nakon otvorene vitičaste zagrade ({), a završava zatvorenom vitičastom zagradom (}).

Prilikom definicije funkcije navodi se:
– tip povratne vrednosti funkcije (ako se ne navede podrazumeva se int)
– ime funkcije
– lista argumenata
– telo funkcije.

U deklaraciji (prototipu) funkcije je sve isto kao i u definiciji, osim što izostaje telo funkcije.
(Obratiti pažnju gde treba koristiti deklaraciju a gde definiciju funkcije)

Primer definicije funkcije:

```
tip_rezultata ime_funkcije (tip param1, tip param2, ..., tip paramN)
{
/*telo funkcije*/
}
```

Primer deklaracije funkcije:

```
tip_rezultata ime_funkcije (tip param1, tip param2, ..., tip paramN);
```

Poziv funkcije

Funkcija se izvršava tako što se „poziva“ u (glavnom) programu ili drugoj funkciji. Funkcija se poziva navođenjem njenog naziva i argumenata zapisanih u redosledu koji je zadan deklaracijom, kao u primeru StaImaslz("kung-fu-a") gde učenika pitamo koju ocenu ima iz Kung-Fu-a.

Povratna vrednost funkcije

U programskom jeziku C++ očekujemo da funkcija vrati neku vrednost. Ova povratna vrednost ima svoj tip kao i ostale vrednosti u C++, pa može biti tipa integer, float, char ili bilo šta drugo. Vrednost funkcije vraćamo sa komandom return.

```
float Kvadriraj(float x)
{
    float rezultat;
    rezultat = x * x;
    return rezultat;
}
```

Funkcija iz gornjeg primera vraća vrednost promenljive “rezultat” kao povratnu vrednost funkcije.

Takođe, možemo koristiti i izraz u return komandi. Na primer možemo zameniti poslednje dve linije funkcije sa 'return (x * x);' Ako zaboravite povratnu vrednost u nekoj funkciji dobićete poruku upozorenja (eng. warning) od C++ kompajlera da funkcija mora da vrati vrednost. Upozorenja kompajlera ne zaustavljaju izvršavanje programa ali greške zaustavljaju. Ipak, dobra je praksa obraćati pažnju i na ova upozorenja i popraviti kod tako da se prevodi bez upozorenja.

Poziv funkcije u programu se ostvaruje navođenjem:
– imena funkcije
– liste stvarnih argumenata funkcije
npr. faktorijel(5);

Mehanizam za vraćanje vrednosti iz funkcije predstavlja naredba return
return izraz;
Tip izraza se evenutualno konvertuje u tip rezultata.

Funkcije koje ne vraćaju vrednost

Prema pravilima pisanja programa na jezicima C i C++, svaka funkcija ima tip. Ako nam nije potrebno da funkcija na mesto poziva vrati vrednost, deklarišemo (objavljujemo) da je tip funkcije, to jest njene povratne vrednosti, tip void (srpski: nevažeći, prazan):

```
void test ()
{
    /* kod funkcije ali bez povratne vrednosti */
}
```

Upotrebom tipa void u deklaraciji samo saopštavamo da povratnu vrednost potprograma nećemo koristiti. Zato se pozivi funkcija tipa void pišu u programu kao naredbe, a ne kao izrazi.

Funkcija tipa void može (i ne mora) da sadrži naredbu return, a ako je sadrži, onda bez ikakvog izraza.

Evo kako to izgleda na primeru:

```
void ispisUpozorenja()
{
    cout << "Upozorenje!";
}
```

Poziv ove funkcije sa nekog drugog mesta u programu bi izgledao ovako:

```
...
ispisUpozorenja();
...
Iza return može i da se ne stavi ništa, ali u tom slučaju se ni jedna vrednost
```

ne vraća pozivaocu.

• Funkcija koja nema povratnu vrednost deklariše se da ima povratni tip void.

• Slično, ako funkcija nema argumente, u deklaraciji se umesto liste argumenata navodi void.

Deklaracija funkcije

Svaka funkcija se pre upotrebe u programu najavljuje (deklariše) kao što se deklarišu promenljive. Ako pri čitanju programa u trenutku kad naiđemo na upotrebu nekog imena, do tada nije već najavljeno to ime i način njegove upotrebe, naš program ne može da se pripremi za izvršenje jer mu to ime nije poznato i u tom trenutku mu nije jasno šta treba da se radi. U najavi je dovoljno navesti samo tipove podataka koje funkcija koristi kao argumente ili vraća kao rezultat. Ajde sad da pogledamo koji je redosled deklaracija funkcije ispravan kroz primere:

- Ovo je ispravno

```
int StaImasIz(string imePredmeta)
{
    …
}

void main()
{
     …
     zbir += StaImasIz("mađioničarskih trikova");
     …
}
```

- I ovo je ispravno

```
int StaImasIz(string imePredmeta);

void main()
{
    …
    zbir += StaImasIz("mađioničarskih trikova");
    …
}

int StaImasIz(string imePredmeta)
```

```
{
    …
}
```

ali ovo nije ispravno

```
void main()
{
    …
    zbir += StaImasIz("mađioničarskih trikova");
    …
}

int StaImasIz(string imePredmeta)
{
    …
}
```

Jer kad naiđemo na StaImaslz, ne znamo da li je to ime funkcije ili nešto drugo.

Argumenti funkcije

Funkcije mogu da prihvataju ulazne parametre u formi promenljivih i izraza i ove ulazne promenljive se mogu koristiti u telu funkcije.

Do sad su parametri služili da prenesu vrednost sa mesta poziva u funkciju i to su ulazni parametri, a moguće je preneti i vrednost iz funkcije (tako da parametri budu izlazni), ali o tome kasnije.

ZADACI ZA SAMOSTALAN RAD

1. Napisati funkciju int min(int x, int y, int z) koja izračunava minimum tri broja. Napisati program koji sa standardnog ulaza učitava tri cela broja i ispisuje rezultat poziva funkcije.

ULAZ
1 8 -2
IZLAZ
Minimum je: -2

ULAZ
-6 -100 -1
IZLAZ
Minimum je: -100

REŠENJE na 2. načina (korisnička i bibliotečka funkcija)

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

/*Funkcija koja racuna minimum tri cela broja*/
int min(int x, int y, int z){
  int min;
  min=x;
  if(min>y)
        min=y;
  if(min>z)
        min=z;
  return min;
}
int main(){
  int x,y,z;
cin >> x>> y>>z;
/* Pozivamo funkciju i ispisujemo rezultat */
cout << "Minimum je " << min(x,y,z) << endl;
return 0;
}
```

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main()
{  //bibliotečka funkcija min koja traži min u listi
   int x,y,z;
   cin >> x>> y>>z;
   // nalazenje najmanjeg broja u listi
  cout << std::min({ x, y, z }) << "\n";
   return 0;
}
```

2. Napisati funkciju *unsigned int apsvrednost(int x)* koja izračunava apsolutnu vrednost broja x. Napisati program koji sa standardnog ulaza učitava jedan ceo broj i ispisuje rezultat poziva funkcije.

ULAZ
Unesite broj: -100
IZLAZ
Apsolutna vrednost: 100

ULAZ
Unesite broj: 8
Apsolutna vrednost: 8

REŠENJE

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

```
/* Funkcija koja racuna apsolutnu vrednost */
unsigned int apsvrednost(int x){
  /* Kako funkcija vraca unsigned, a x je tipa int, vrsimo kastovanje rezultata u tip unsigned */
  return (unsigned)(x<0?-x:x);

  /* if (x<0) return (unsigned)(-x); else return (unsigned)x; */
  /* if (x<0) return (unsigned)(-x); return (unsigned)x; */
}

int main(){
        int n;

    cout << "Unesite broj: ";
    cin >> n;
        cout << "Apsolutna vrednost: " << apsvrednost(n) << endl;

        return 0;
}
```

3. Napisati funkciju void romb(int n) koja iscrtava romb čija je stranica dužine n. Napisati program koji učitava ceo pozitivan broj i ispisuje rezultat poziva funkcije. U slučaju pogrešnog unosa, ispisati poruku o grešci.

```
ULAZ
Unesite broj n: 5
IZLAZ
      *****
     *****
    *****
   *****
  *****
*****

ULAZ
Unesite broj n: 2
IZLAZ
 **
**

ULAZ
Unesite broj n: -1
IZLAZ
Greska: pogresna dimenzija!
```

REŠENJE

```
#include<stdio.h>

/* Funkcija koja iscrtava romb */
void romb(int n){
  int i, j;
    /* U svakoj liniji */
  for(i=0; i<n; i++){
```

```
        /* Prvo ispisujemo n-i-1 razmaka */
        for(j=0; j<n-i-1; j++)
          printf(" ");

        /* Zatim ispisujemo n zvezdica */
        for(j=0; j<n; j++)
          printf("*");

        /* Na kraju svake linije stoji oznaka za novi red */
        printf("\n");
  }

}

int main(){
  int n;

  /* Ucitavamo broj n */
  printf("Unesite broj n: ");
  scanf("%d", &n);

  /* Proveravamo korektnost ulaza i ispisujemo rezultat */
  if(n<=0)
        printf("Greska: pogresna dimenzija!\n");
  else
        romb(n);
  return 0;
}
```

4. Napisati funkciju int prestupna(int godina) koja za zadatu godinu proverava da li je prestupna. Funkcija treba da vrati 1 ako je godina prestupna ili 0 ako nije. Napisati program koji učitava dva cela broja g1 i g2 i ispisuje sve godine iz intervala [g1; g2] koje su prestupne.

ULAZ
Unesite dve godine: 2001 2010
IZLAZ
Prestupne godine su: 2004 2008

ULAZ
Unesite dve godine: 2005 2015
IZLAZ
Prestupne godine su: 2008 2012

ULAZ
Unesite dve godine: 2010 2001
IZLAZ
Greska: pogresan unos!

ULAZ
Unesite dve godine: 2001 2002
IZLAZ

Nema prestupnih godina u ovom intervalu!

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

/* Funkcija koja proverava da li je godina prestupna */
int prestupna(int godina){
  if((godina %100 != 0 && godina%4 == 0) || godina%400 == 0)
        return 1;
  else
        return 0;
}

/* Funkcija koja proverava da li postoji prestupna godina u datom intervalu */
int postoji_prestupna(int g1, int g2){
  for(; g1<=g2; g1++){
        if(prestupna(g1))
          return 1;
  }
  return 0;
}

int main(){

  int g1, g2;

  /* Ucitavamo godine */
  cout << "Unesite dve godine: ";
  cin >>  g1 >> g2;

  /* Proveravamo korektnost ulaza */
  if(g1 < 0 || g2 < 0 || g1>g2){
        cout << "Greska: pogresan unos!" << endl;
  }
  else{

        /* Proveravamo da li uopste postoji prestupna godina u datom intervalu */
        if(postoji_prestupna(g1,g2)){
          /* Ako postoje, ispisujemo ih */
          cout << "Prestupne godine su: ";
          for(; g1<=g2; g1++){
                if(prestupna(g1))
                  cout << g1 << " ";
          }
          cout << endl;

        }else{
          /* U suprotnom, stampamo odgovarajucu poruku */
          cout << "Nema prestupnih godina u ovom intervalu!" << endl;
        }
  }
```

```
  return 0;
}
```

5. Napisati funkciju int zbir_delilaca(int n) koja izračunava zbir delilaca broja n. Napisati program koji sa standardnog ulaza učitava ceo broj k i ispisuje zbir delilaca svakog broja od 1 do k.

Primer 1
Interakcija sa programom:
Unesite broj k: 6
1 3 4 7 6 12

Primer 2
Interakcija sa programom:
Unesite broj k: -2
Greska: pogresan unos!

6. Napisati funkciju int ukloni_stotine(int n) koja modifikuje zadati broj tako što iz njegovog zapisa uklanja cifru stotina (ako postoji). Napisati program koji za brojeve koji se unose sa standardnog ulaza sve do pojave broja 0 ispisuje rezultat primene funkcije.

Primer 1
Interakcija sa programom:
Unesite broj: 1210
110
Unesite broj: 18
18
Unesite broj: 3856
356
Unesite broj: 0

Primer 2
Interakcija sa programom:
Unesite broj: -9632
-932
Unesite broj: 246
46
Unesite broj: -52
-52
Unesite broj: 0

7. Napisati funkciju int rotacija(int n) koja rotira cifre zadatog broja za jednu poziciju u levo. Napisati program koji za brojeve koji se unose sa standardnog ulaza sve do pojave broja 0 ispisuje rezultat primene funkcije.

Primer 1
Interakcija sa programom:
Unesite broj: 146
461
Unesite broj: 18
81

Unesite broj: 3856
8563
Unesite broj: 7
7
Unesite broj: 0

Primer 2
Interakcija sa programom:
Unesite broj: 89
98
Unesite broj: -369
-693
Unesite broj: -55281
-52815
Unesite broj: 0

8. Napisati funkciju float aritmeticka_sredina(int n) koja računa aritmetičku sredinu cifara datog broja. Napisati i program koji testira rad napisane funkcije. Rezultat ispisivati na tri decimale.

Primer 1
Interakcija sa programom:
Unesite broj: 461
3.667

Primer 2
Interakcija sa programom:
Unesite broj: 1001
0.500

Primer 3
Interakcija sa programom:
Unesite broj: -84723
4.800

9. Napisati funkciju int zapis(int x; int y) koja proverava da li se brojevi x i y zapisuju pomoću istih cifara. Funkcija treba da vrati vrednost 1 ako je uslov ispunjen, odnosno 0 ako nije. Napisati i program koji učitava dva cela broja i ispisuje rezultat primene funkcije.

Primer 1
Interakcija sa programom:
Unesite dva broja: 251 125
Uslov je ispunjen!

Primer 2
Interakcija sa programom:
Unesite dva broja: 8898 9988
Uslov nije ispunjen!

Primer 3
Interakcija sa programom:

Unesite dva broja: -7391 1397
Uslov je ispunjen!

10. Napisati funkciju int faktorijel(int n) koja računa faktorijel broja n. Napisati i program koji učitava dva cela broja x i y (0 <=x, y <= 12) i ispisuje vrednost zbira x! + y!.

Primer 1
Interakcija sa programom:
Unesite dva broja: 4 5
144

Primer 2
Interakcija sa programom:
Unesite dva broja: 18 -5
Greska: pogresan unos!

Primer 3
Interakcija sa programom:
Unesite dva broja: 6 0
721

11. **Napisati funkciju float razlomljeni_deo(float x) koja izračunava razlomljeni deo broja x. Napisati program koji sa standardnog ulaza učitava jedan realan broj i ispisuje rezultat poziva funkcije.**

Primer 1
Interakcija sa programom:
Unesite broj: 8.235
Razlomljeni deo: 0.235000
Primer 2
Interakcija sa programom:
Unesite broj: -5.11
Razlomljeni deo:

REŠENJE

```
#include<stdio.h>
#include<math.h>

/* Funkcija koja vraca razlomljeni deo prosledjenog broja */
float razlomljeni_deo(float x){

  /* Funkcija fabs vraca apsolutnu vrednost realnog broja
   * NAPOMENA: funkcija fabs se nalazi u zaglavlju math.h
   * NAPOMENA2: funkcija abs se nalazi u zaglavlju stdlib.h, ali se koristi samo za cele brojeve!
  */
  x = fabs(x);

  /* Razlomljeni deo broja dobijamo tako sto od samog broja oduzmemo njegov ceo deo*/
  return x - (int)x;
}
```

```
int main(){
  float n;

  /* Ucitavamo broj */
  printf("Unesite broj:");
  scanf("%f", &n);

  /* Ispisujemo rezultat */
  printf("Razlomljeni deo: %.6f\n", razlomljeni_deo(n));

  return 0;
}
```

DOMAĆI ZADATAK
Napisati sledeće funkcije kao i programe koji ih testiraju
– maksimum dva broja
– faktorijel datog broja
– n-ti stepen realnog broja x (omogu ćiti da funkcija radi i za negativne stepene)
– ojlerova funkcija datog broja (za zadati broj računamo koliko ima brojeva koji su veći ili jednaki od 1, a manji od datog broja, a u isto vreme su uzajamno prosti sa datim brojem)

1. Zadate su tri verzije C programa. Koja od njih nije korektna i zašto?

```
#include <stdio.h>
void ispisati (void) ;
main()
{ int a,b,c;
   a=b=c=2;
   ispisati (a, b, c);
   c=4;
   ispisati (a, b, c);
}

/*f-ja koja stampa na glavni izlaz*/
void ispisati(int i, int j, int k)
{

  printf("1. argument je %d\n", i);
  printf("2. argument je %d\n", j);
  printf("3. argument je %d\n", k);

}
```

```
#include <stdio.h>
void ispisati (int a, int b, int c) ;
main()
{ int a,b,c;
   a=b=c=2;
   ispisati (a, b, c);
   c=4;
   ispisati (a, b, c);
}

/*f-ja koja stampa na glavni izlaz*/
void ispisati(int i, int j, int k)
{

  printf("1. argument je %d\n", i);
  printf("2. argument je %d\n", j);
  printf("3. argument je %d\n", k);

}
```

```
#include <stdio.h>
void ispisati (int, int, int) ;
main()
{ int a,b,c;
   a=b=c=2;
   ispisati (a, b, c);
   c=4;
   ispisati (a, b, c);
}

/*f-ja koja stampa na glavni izlaz*/
void ispisati(int i, int j, int k)
{

  printf("1. argument je %d\n", i);
  printf("2. argument je %d\n", j);
  printf("3. argument je %d\n", k);

}
```

ARGUMENTI FUNKCIJE - PRENOS PARAMETARA PO VREDNOSTI

Parametri se funkciji prenose po vrednosti, što znači da funkcija zapravo radi sa njihovim lokalnim kopijama.

Sve promene koje funkcija načini nad parametrima, zapravo se odnose na lokalne kopije parametara, a ne na same parametre.

Zaključak je da se preneti parametri NE MOGU promeniti u telu funkcije.

```
#include <stdio.h>

void f (int x)
{
   x *= 2;
   x++;
}

int main()
{
   int x = 3;
   f(x);
   printf("%d\n", x);

   return 0;
}
```

Analiza prethodnog primera

U prethodnom primeru, biće ispisano 3,jer se promene na x-u, vrše nad njegovom lokalnom kopijom u funkciji f, a ne baš nad x-om.

•Ako hoćemo da u funkciji promenimo x, jedan od načina je da iz funkcije vratimo to promenjeno x (na kraju funkcije da stavimo return x;), dok u main funkciji da kažemo x = f(x);

Analiza prenosa parametara u primeru sa funkcijom zbir

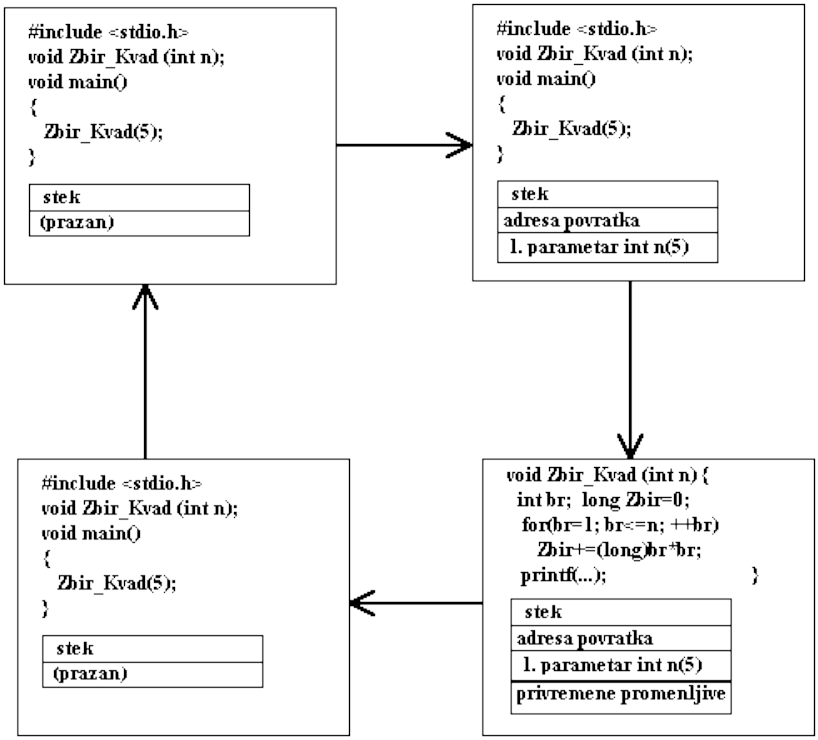
```
#include <stdio.h>
/* Deklaracija funkcije zbir() */
int zbir(int, int);          /* int zbir(int a, int b); */
int main() {
/* Poziv funkcije */
printf("%d\n", zbir(3,5));
return 0;
}
/* Definicija funkcije */
int zbir(int a, int b) {return a+b;}
```

Kada se u main funkciji pojavi poziv funkcije rezultat = zbir (5,3) realizuju se sledeće akcije:
– Rezerviše se memorijski prostor za promenljive deklarisane u funkciji zbir();
– Formalnim parametrima se dodeljuju vrednosti stvarnih parametara: a=3; b=5;
– Izvršava se telo funkcije;
– Rezultat izračunavanja u funkciji se postavlja na mesto obraćanja toj funkciji, odnosno dodeljuje se promenljivoj rezultat, i prelazi se na izvršenje sledeće naredbe u funkciji main().

Napisati C program koji će izračunati sume $\sum_{i=1..5} i^2$ i $\sum_{i=1..23} i^2$ i obe sume ispisati u zasebnim linijama.

```
#include <stdio.h>
void  Zbir_Kvad(int n);   /*f-ja koja vrsi zeljeno izracunavanje */
main()
{  Zbir_Kvad( 5);
   Zbir_Kvad( 23);
}
void  Zbir_Kvad(int n)
{  int br; /* lokalna promenljiva funkcije, brojac u ciklusu */
    long Zbir=0; /* lokalna promenljiva funkcije, suma kvadrata brojeva od 1..n */
   for  (br=1; br<=n;  Zbir+= (long)  br*br,  ++br)  ;
   printf(" Zbir kvadrata brojeva od 1 do %d jese %ld\n", n,Zbir);
}
```

Poziv funkcije: pozivajuća jedinica(main) -> funkcija Zbir_Kvad -> pozivajuća jedinica(main)



SPOLJAŠNJE (GLOBALNE)PROMENLJIVE

C program sastoji se od skupa spoljašnjih objekata, koji mogu biti:

– promenljive

– funkcije

Spoljašnje promenljive su definisane izvan svake funkcije i zato su potencijalno na raspolaganju raznim funkcijama.

Funkcije su uvek spoljašnje, jer C ne dozvoljava definisanje funkcija unutar drugih funkcija

Prenošenje podataka između funkcija:

– preko argumenata

– preko povratnih vrednosti

– preko spoljašnjih promenljivih

Ukoliko se veliki broj promenljivih mora deliti između funkcija, bolje je koristiti spoljašnje promenljive nego dugačke liste argumenata.

Međutim, ovo treba pažljivo primenjivati, jer u suprotnom se dobija program sa previše veza između funkcija.

Spoljašnje promenljive – primer

```
#include <stdio.h>
/* Deklaracija funkcije prebroj */
void prebroj();
/* Globalni brojaci. Podrazumevano su inicijalizovani nulom. */
int br_malih, br_velikih, br_cifara, br_belina, br_redova;
int main() {
/* Pozivamo funkciju za prebrojavanje */
prebroj();
/* Prikazujemo vrednosti globalnih brojaca, koje je funkcija
prebroj() izmenila */
printf("Broj malih slova: %d\n", br_malih);
printf("Broj velikih slova: %d\n", br_velikih);
printf("Broj cifara: %d\n", br_cifara);
printf("Broj belina: %d\n", br_belina);
printf("Broj redova: %d\n", br_redova);
/* Povratna vrednost funkcije main() */
return 0;
}

/* Definicija funkcije prebroj() */
void prebroj() {
int c;
while((c = getchar()) != EOF) {
    if(c >= 'a' && c <= 'z')  ++br_malih;
    else if(c >= 'A' && c <= 'Z')   ++br_velikih;
    else if(c >= '0' && c <= '9')  br_cifara++;
   else if (c == '\n' || c == '\t' || c ==' ') {
```

```
            br_belina++;
            if(c == '\n') br_redova++;
        }
    }
}
```

Poređenje spoljašnjih i automatskih promenljivih

Automatske promenljive su unutrašnje za funkciju. One nastaju kada otpočinje izvršavanje funkcije i nestaju kada se ono završava.

Spoljašnje promenljive su permanentne i stoga zadržavaju svoje vrednosti između poziva funkcija. Dakle, životni ciklus ovih promenljivih je duži.

STATIČKE PROMENLJIVE

Deklaracija static, koja se primenjuje na spoljašnju promenljivu ili funkciju, ograničava domet tog objekta na preostali deo izvorne datoteke.

Deklaracija static se može primeniti i na unutrašnje promenljive. Unutrašnje statičke promenljive su lokalne za određenu funkciju baš kao i automatske promenljive, ali za razliku od njih, one nastavljaju da postoje i nakon završetka funkcije. Dakle, one ne nastaju i ne nestaju sa svakim pozivom funkcije.

INICIJALIZACIJA

Spoljašnje i statičke promenljive se uvek inicijalizuju na nulu ukoliko nisu eksplicitno inicijalizovane. Inicijalizator mora biti konstantan izraz. Inicijalizacija se obavlja samo jednom, pre početka izvršavanja programa.

Automatske promenljive imaju nedefinisane početne vrednosti ukoliko nisu eksplicitno inicijalizovane. Inicijalizator može biti svaki izraz u kojem učestvuju prethodno definisane vrednosti, pa čak i pozivi funkcija. Inicijalizacija se obavlja svaki put prilikom izvršavanja funkcije ili bloka.

Životni vek i oblast važenja promenljivih – primer

```
#include <stdio.h>
/* Globalna promenljiva */
int a = 0;
/* Uvecava se globalna promenljiva */
void uvecaj() {a++;  printf("uvecaj::a = %d\n", a);}

/* Umanjuje se lokalna promenljiva a. Globalna promenljiva sa istim imenom zadrzava svoju vrednost. */
void umanji() {
```

```
/* Ova promenljiva a je nezavisna u odnosu na globalnu promenljivu a */
int a = 0;
a--;
printf("umanji::a = %d\n", a);
}

void nestaticka_prom() {
/* Nestaticke promenljive ne cuvaju vrednosti kroz pozive funkcije */
int s = 0;
s++;
printf("nestaticka promenljiva::s = %d\n", s);
}

void staticka_prom() {
/* Staticke promenljive cuvaju vrednosti kroz pozive funkcije.
Inicijalizacija se odvija samo u okviru prvog poziva. */
static int s = 0;
s++;
printf("staticka promenljiva::s = %d\n", s);
}

int main() {
int i, x = 3; /* Ovo su promenljive lokalne za funkciju main */
printf("main::x = %d\n", x);
for(i=0; i<3; i++) {
/* Promenljiva u okviru bloka je nezavisna od spoljne promenljive. Ovde se koristi promenljiva x
lokalna za blok for petlje koja ima vrednost 5, dok originalno x i dalje ima vrednost 3 */
int x = 5;
printf("for::x = %d\n", x);
}
/* U ovom bloku x ima vrednost 3 */
printf("main::x = %d\n", x);
uvecaj();
umanji();
/* Globalna promenljiva a */       printf("main::a = %d\n", a);
/* Demonstracija nestatickih promenljivih */
for(i=0; i<3; i++) nestaticka_prom();
/* Demonstracija statickih promenljivih */
for(i=0; i<3; i++)  staticka_prom();
return 0;
}
```

12.

NCP koji ispisuje prvih n prostih brojeva, gde se broj n ucitava sa stanadrdnog ulaza

```
#include<stdio.h>
int prost(int n);  /*Ispituje se da li je broj n prost, vraca 0 ako nije, a nenula ako jeste */
main()
{
int n; /* broj zeljenih prostih brojeva */
int i; /* brojac ispisanih prostih brojeva */
int br; /* kandidat za prost broj */


printf("Unesite koliko prostih brojeva zelite da dobijete: \n");
scanf("%d", &n);


/* Inicijalizujemo brojac i - koliko smo prostih brojeva nasli do sad */
i = 0;


/* Pocetni broj za koji proveravamo da li je prost */
br = 2;


/* Trazimo i-ti prost broj */
while(i < n) {
   /* Ako je broj prost... */
   if (prost(br)) {
      /* stampamo ga... */
     printf("Broj %d je prost.\n", br);
      /* i uvecavamo broj pronadjenih prostih brojeva. */
     i++;
 }
/* U svakom slucaju prelazimo na proveru da li je sledeci broj prost */
br++;


}
}
int prost(int n)  /*Ispituje se da li je broj n prost tako sto se proverava da li ima delioce medju brojevima od 2 do sqrt(n).  Pri implementaciji se koristi tvrdjenje da je broj prost ako je jednak 2, ili  ako je neparan i ako nema delitelja medju neparnim brojevima od 3 do n/2 */
{
  int i;    /*potencijalni delitelj broja n */
if (n<=1) return 0;
  for(i=2; (i*i<=n);i=i+1+(i>2) )
```

```
   if(n%i == 0) return 0;
 return 1;
}
```

13.

```
Ako je dat niz prirodnih prirodnih brojeva, napisati program kojim se
vraca najveca duzina uzastopnih prostih brojeva.
ulaz                                                izlaz
1 2 3 4 5                                           2
1 2 3 5 7 8 9 11 13 17 18   4
```

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 50

int prost(int n)
{
  int i;    /*potencijalni delitelj broja n */
  if (n<=1) return 0;
  for(i=2; (i*i<=n);i=i+1+(i>2) )
    if(n%i == 0) return 0;
 return 1;
}

int najduza_serija(int a[], int n)
{
int i;
  int ts, ns; //duzina tekuce serije, duzina max serije
  ts = 1;
  ns = 0;
for(i=1; i<n; i++)
{
        if(prost(a[i]) && prost (a[i-1])) ts++;
        else ts = 1;
        if(ts > ns) ns = ts;
}
return ns;
}

int main()
{   int a[DIM],i,n;
    printf("Unesite dimenziju niza\n"); scanf("%d", &n);
    printf("Unesite clanove niza\n");
    for(i=0;i<n;i++) scanf("%d", &a[i]);
     printf("Najduza serija prostih ima duzinu %d\n", najduza_serija(a,n));
    return 0;
}
```

14.

```
/* strlen, strcpy, strcat, strcmp, strchr, strstr
-manipulacija niskama karaktera */
Napisati C funkcije koje implementiraju funkcije iz
biblioteke string.h i to:
1. funkciju string_length koja implementira funkciju
strlen iz zaglavlja string.h
```

2. funkciju string_copy koja impementira funkciju strcpy iz string.h
3. funkciju string_concatenate koja implementira funkciju strcat iz string.h
4. funkciju string_char koja implementira funkciju strchr iz string.h
5. funkciju string_last_char koja pronalazi poslednju pojavu karaktera u nisci
6. funkciju string_string koja slicno funkciji strstr iz string.h proverava da li je neka niska podniska druge
NCP koji ilustruje pozive ovih funkcija.

```
#include <stdio.h>

/* Izracunava duzinu stringa */
int string_length(char s[])
{
   int i;
   for (i = 0; s[i]; i++) ;
   return i;
}



/* Kopira string src u string dest. Pretpostavlja da u dest ima dovoljno prostora. */
void string_copy(char dest[], char src[])
{
   /*Kopira karakter po karakter,sve dok nije iskopiran karakter '\0' */
    int i;
      for (i = 0; (dest[i]=src[i]) != '\0'; i++) ;

                /* Uslov != '\0' se, naravno, moze izostaviti : for (i = 0; dest[i]=src[i]; i++);   */
}


/* Nadovezuje string t na kraj stringa s. Pretpostavlja da u s ima dovoljno prostora. */
void string_concatenate(char s[], char t[])
{
    int i, j;

     /* Pronalazimo kraj stringa s */
      for (i = 0; s[i]; i++);

      /* Vrsi se kopiranje, slicno funkciji string_copy, ali ne od pocetka niske s, vec od kraja niske s */
      for (j = 0; s[i] = t[j]; j++, i++) ;

}
```

```
/* Vrsi leksikografsko poredjenje dva stringa.
Vraca :
0 - ukoliko su stringovi jednaki
<0 - ukoliko je s leksikografski ispred t
>0 - ukoliko je s leksikografski iza t
*/
int string_compare(char s[], char t[])
{
   int i;

  /* Petlja tece sve dok ne naidjemo na prvi razliciti karakter */
  for (i = 0; s[i]==t[i]; i++)
         if (s[i] == '\0')  return 0; /* Naisli smo na kraj oba
stringa, a nismo nasli razliku */


   /* s[i] i t[i] su prvi karakteri u kojima se niske razlikuju.
    Na osnovu njihovog odnosa (razlike ASCII vrednosti), odredjuje
se odnos stringova < ili > ili ==*/
      return s[i] - t[i];
}



/* Pronalazi prvu poziciju karaktera c u stringu s, odnosno -1
ukoliko s ne sadrzi c */
int string_char(char s[], char c)
{
  int i;

   for (i = 0; s[i]; i++)
       if (s[i] == c)  return i;

/* NIKAKO
  else  return -1;
*/

/* Nije nadjeno */
  return -1;
}


/* Pronalazi poslednju poziciju karaktera c u stringu s, odnosno
-1 ukoliko s ne sadrzi c */
int string_last_char(char s[], char c)
{
  int i;

 /* Najpre pronalazimo kraj stringa s */
  for (i = 0; s[i]; i++) ;

   /* Krecemo od kraja i trazimo c unazad */
     for (i--; i>=0; i--)
        if (s[i] == c)  return i;
```

```
   /* Nije nadjeno */
     return -1;


   /* ILI drugi nacin bi bio koristeci string_length :
              for (i = string_length(s) - 1; i>0; i--)
                  if (s[i] == c)  return i;
            return -1;
    */

}


/* Proverava da li string str sadrzi string sub.
Vraca poziciju na kojoj sub pocinje, odnosno -1 ukoliko ga nema
*/
int string_string(char str[], char sub[])
{
   int i, j;

/* Proveravamo da li sub pocinje na svakoj poziciji i */
  for (i = 0; str[i]; i++)
       /* Poredimo sub sa str pocevsi od poziciji i sve dok ne
naidjemo na razliku */
     for (j = 0; str[i+j] == sub[j]; j++)
       /* Nismo naisli na razliku a ispitali smo sve karaktere
niske sub */
        if (sub[j+1]=='\0') return i;

/* Nije nadjeno */
return -1;

}

main()
{
char s[100];
char t[] = "Zdravo";
char u[] = " svima";

string_copy(s, t);
printf("%s\n", s);

string_concatenate(s, u);
printf("%s\n", s);

printf("%d\n",string_char("racunari", 'n'));

printf("%d\n",string_last_char("racunari", 'a'));

printf("%d\n",string_string("racunari", "rac"));

printf("%d\n",string_string("racunari", "ari"));

printf("%d\n",string_string("racunari", "cun"));
```

```
printf("%d\n",string_string("racunari", "cna"));

}
```

```
Izlaz:
Zdravo
Zdravo svima
4
5
0
5
2
-1
```

/* Linearna pretraga u niza karaktera
NCP koji sa standardnog ulaza unosi pomocu funkcije scanf nisku sa ne vise od 20 karaktera i proverava da li niska sadrzi znak @ */

Postupak linearnog pretraživanja

Traženi element x se može naći lineranim (sekvencijalnim) pretraživanjem.
Redom se porede elementi ulaznog niza v sa vrednošću od x, sve do prvog elementa v[i] za koji važli v[i] >=x.
Tada, ako je:

- v[i] = = x, onda je poz=i tražena pozicija
- v[i] > x, onda se x ne nalazi u nizu v

Ako je svaki element niza v manji od vrednosti od x, onda se x ne nalazi u nizu v (izvrsi se n poredjenja).

```
#include <stdio.h>
#include <string.h>
/* Funkcija proverava da li se dati element x nalazi
   u datom nizu a.
   Funkcija vraca poziciju u nizu na kojoj je x pronadjen
   odnosno -1 ukoliko elementa nema.
*/
int linear_search(char a[], int n, int x)
{
        int i;
        for (i = 0; i<n; i++)
                if (a[i] == x)
                        return i;
                /* nema potrebe za  else!!! */
        return -1;
}

main()
{
        char a[21] ; /*niska sa stdin */
                  int n; /*broj unetih karaktera niske a*/
        int i; /*brojacka promenljiva*/
        printf("Unesite nisku : ");
         scanf("%s",a);
        n = strlen(a);
```

```
        i = linear_search(a, n, '@');
        if (i == -1)
                printf("Karakter @ nije nadjen\n");
        else
                printf("Karakter @  je na poziciji %d\n", i);
}
```

/* Binarna pretraga u niza karaktera
NCP koji sa standardnog ulaza unosi pomocu funkcije scanf leksikografski uredjenu nisku sa ne vise od 20 karaktera i proverava da li niska sadrzi znak @. ASCII(@) = 64 */

```
/*----------------------Binarna pretraga niza celih brojeva - iterativna
verzija-------------------------------*/

#include <stdio.h>
#include <string.h>

/* Funkcija proverava da li se element x javlja unutar niske a.
   Funkcija vraca poziciju na kojoj je element nadjen odnosno
   -1 ako ga nema.

   !!!!! VAZNO !!!!!
   Pretpostavka je da je niska a sortirana
*/

int binary_search(char a[], int n, int x)
{
        /* Pretrazujemo interval [l, d] */
        int l = 0;
        int d = n-1;

        /* Sve dok interval [l, d] nije prazan */
        while (l <= d)
        {
                /* Srednja pozicija intervala [l, d] */
                int s = (l+d)/2;

                /* Ispitujemo odnos x i srednjeg elementa a[s]*/
                if (x == a[s])
                        /* Element je pronadjen */
                        return s;
                else if (x < a[s])
                {
                        /* Pretrazujemo interval [l, s-1] */
                        d = s-1;
                }
                else
                {
                        /* Pretrazujemo interval [s+1, d] */
                        l = s+1;
                }
        }

        /* Element je nadjen */
        return -1;
}

main()
{
```

```
        char a[21] ; /*niska sa stdin */
                  int n; /*broj unetih karaktera niske a*/
        int i; /*brojacka promenljiva*/
        printf("Unesite nisku : ");
        scanf("%s",a);

        i = binary_search(a, strlen(a), '@');

        if (i==-1)
                printf("Nema karaktera @\n");
        else
                printf("Pronadjen @ na poziciji %d\n", i);
}
```