Programiranje 1

Beleške sa vežbi

Školska 2008/2009 godina

Matematički fakultet, Beograd

Jelena Graovac (Tomašević)

December 11, 2008

Sadržaj

1

Programski jezik C

[1]

1.1 Switch

Naredba `switch` se naziva još i naredba višestrukog grananja. Ovom naredbom se proverava da li je neki izraz jednak jednoj od više konstantnih celobrojnih vrednosti i u zavisnosti od toga, preduzima se odgovarajuća akcija. Ovo se može realizovati i sa više `if` operatora međutim korišćenjem `switch` operatora programiranje postaje prostije i jasnije. Opšti oblik ove naredbe je:

```
switch (izraz) {
    case konstanta1: naredbe1
    case konstanta2: naredbe2
    ...
    default: naredbe
}
```

Svaka konstanta predstavlja određeni slučaj. Ako neki slučaj odgovara vrednosti izraza, izvršavanje počinje od tog slučaja. Sve konstante odnosno svi slučajevi moraju biti različiti. Slučaj označen sa `default` nije obavezan. Ako se ne navede i nijedan drugi slučaj ne odgovara vrednosti izraza onda se neće izvršiti nikakva akcija.

Primer 1 *Ilustracija switch konstrukcije.*

```
#include<stdio.h>

int main()
{
    int n;
    printf("Unesi paran broj manji od 10:\n");
    scanf("%d",&n);
    switch(n)
    {
        case 0:
            printf("Uneli ste nulu.\n");
            break;
        case 2:
```

[1] Zasnovano na primerima sa sajta `http://www.matf.bg.ac.yu/~filip` i knjige Milana Čabarkape "C - osnovi programiranja".

```
            printf("Uneli ste dvojku.\n");
            break;
        case 4:
            printf("Uneli ste cetvorku.\n");
            break;
        case 6:
            printf("Uneli ste sesticu.\n");
            break;
        case 8:
            printf("Uneli ste osmicu.\n");
            break;
        defalut:
            printf("Uneli ste nesto sto nije paran broj.\n");
    }
    return 0;
}

Ulaz:
Unesi paran broj manji od 10:
2
Izlaz:
Uneli ste dvojku.
```

`break` operator se koristi u `switch` operatoru da bi se obezbedio izlaz neposredno iza njega. Ukoliko se iza neke grupe naredbi u `swith` ispusti `break` operator, tada će se u slučaju izbora te grupe naredbi izvršiti i sve preostale alternativne grupe naredbi koje odgovaraju slučajevima koji su navedeni ispod tog slučaja, sve dok se ne pojavi sledeće `break` ili kraj `swith` naredbe. Na primer, u prethodnom primeru, ako se nigde ne napise `break` operator, izvršenje programa će da izgleda ovako:

```
Ulaz:
Unesi paran broj manji od 10:
2
Izlaz:
Uneli ste dvojku.
Uneli ste cetvorku.
Uneli ste sesticu.
Uneli ste osmicu.
Uneli ste nesto sto nije paran broj.
```

Primer 2 *Napisati program koji vrši brojanje pojavljivanja karaktera* `'0'`, `'1'` *i* `'2'` *korišćenjem* `switch` *naredbe.*

```
#include <stdio.h>

main() {
    int c;
    int br_0=0, br_1=0, br_2=0;

    while ((c = getchar()) != EOF)
    {
        switch(c)
        {
```

```
            /* Obratiti paznju da nije case 0: niti case '0'; */
            case '0':
                br_0++;
                break;  /* Isprobati veziju bez break */
            case '1':
                br_1++;
                break;
            case '2':
                br_2++;
                break;
        }
    }
    printf("Br 0 : %d\nBr 1 : %d\nBr 2 : %d\n",br_0, br_1, br_2);
}
```

1.2 Uslovni izraz

Uslovni izrazi obezbeđuju alternativni način pisanja `if-else` konstrukcije. Tako se

```
if(izraz1)
     izraz2;
else izraz3;
```

može zapisati kao

`izraz1 ? izraz2 : izraz3`

Prvi primer korišćenja uslovnog izraza:

Primer 3 *Bez korišćenja operatora if napisati operatore kojima se za zadate a i b izračunavaju njihov minimum i maksimum.*

```
min = (a<b)? a : b;
max = (a>b)? a : b;
```

Drugi primer korišćenja uslovnog izraza:

Primer 4 *Bez korišćenja operatora if napisati operator kojim se za zadate x i y izračunava z po formuli:*

$$z = \begin{cases} min(x,y) & ako\ je\ y >= 0 \\ max(x^2,y^2) & ako\ je\ y < 0 \end{cases}$$

```
Resenje:
z=(y>=0)?((x<y)?x:y):((x*x>y*y)?x*x:y*y)
```

1.3 Operator sizeof

Operator `sizeof` izračunava veličinu tipa (ili promenljive) u broju BAJTOVA (ne broju bitova) koji zauzima u memoriji.

Primer 5 *Demonstracija sizeof operatora.*

```
#include<stdio.h>

main()
{
```

```
    int i;
    float f;

    printf("sizeof(int)=%d\n", sizeof(int));
    printf("sizeof(long)=%d\n", sizeof(long));
    printf("sizeof(short)=%d\n", sizeof(short));
    printf("sizeof(signed)=%d\n", sizeof(signed));
    printf("sizeof(unsigned)=%d\n", sizeof(unsigned));
    printf("sizeof(char)=%d\n", sizeof(char));
    printf("sizeof(float)=%d\n", sizeof(float));
    printf("sizeof(double)=%d\n", sizeof(double));

    printf("sizeof(i)=%d\n", sizeof(i));
    printf("sizeof(f)=%d\n", sizeof(f));
}

Izlaz iz programa(u konkretnom slucaju):
sizeof(int)=4
sizeof(long)=4
sizeof(short)=2
sizeof(signed)=4
sizeof(unsigned)=4
sizeof(char)=1
sizeof(float)=4
sizeof(double)=8
sizeof(i)=4
sizeof(f)=4
```

Napomena: Ako nas zanima koliki broj bitova zauzima neki tip onda se samo rezultat od `sizeof(tip)` pomnoži sa 8: Na primer, broj bitova koje zauzima tip char je `sizeof(char)*8`.

1.4 Operator zarez

Dva izraza razdvojena zarezom se izračunavaju sleva na desno, ali se vrednost levog izraza odbacuje. Tip i vrednost rezultata su tip i vrednost desnog operanda. Najčešće se koristi u okviru for petlje. Tako se na primer sledeći fragment programa:

```
s=0;
for(i=0; i<=3; i++)
s+=i;
```

korišćenjem operatora zarez može zapisati na sledeći način:

```
for(s=0, i=0; i<=3; s+=i, i++);
```

1.5 Nizovi — osnovni pojmovi

Ogroman je broj problema koji bi se upotrebom prostih pormenljivih vrlo teško rešili, ili se uopšte ne bi mogli rešiti. Na primer, ako želimo da nađemo maksimalnu vrednost između nekoliko brojeva, upotrebom prostih promenljivih i `if` operatora to se lako rešava. Međutim, ako se radi o hiljadu brojeva, koristiti niz if operatora je krajnje nepraktično, jer treba porediti 1000 vrednosti promenljivih:

```
int br1, br2, br3, ... br1000;
```

Efikasno oruđe kojim raspolaže C za rešavanje ovakvih problema su **nizovi**. Niz pretstavlja kolekciju elemenata istog tipa. Primer deklaracije niza je:

```
int broj[1000]; /* niz sa imenom \verb"broj" od 1000 elemenata tipa int*/
```

Time se uvodi niz `broj` koji se sastoji iz 1000 celobrojnih pormenljivih sa imenima `broj[0], broj[1],...,broj[999]`. Brojeve od 0 do 999 nazivamo indeksima niza.

Drugi primer deklaracije niza je:

```
int niz[5]; /* niz sa imenom \verb"niz" od 5 elemenata tipa int*/
```

Pristupanje elementima niza se ostvaruje putem indeksa na sledeći način:

```
niz[0] = 4;
niz[1] = 2 * niz[0];        /*niz[1] = 8*/
niz[2] = niz[0] * niz[1];   /*niz[2] = 32*/
niz[3] = 5;
niz[4] = 7;
```

Napomena: Prvi element u nizu se označava sa indeksom 0 a ne 1!

Napomena: Programer mora voditi računa o tome da ne prekorači dimenziju niza. Na primer, ako je niz definisan sa `int niz[5];` moze se pristupiti samo članovima sa indeksom od 0 do 4, ne više od toga!!!

Indeks niza može da bude proizvoljan izraz celobrojne vrednosti. Na primer može se napisati `niz[i*2]=5`.

Primer unosa vrednosti elemenata niza sa standardnog ulaza:

```
for(i=0; i<5; i++)
    scanf("%d ", &a[i]);
```

Stampanje elemenata niza na standardni izlaz.

```
for(i=0; i<5; i++)
    printf("%d ", a[i]);
```

Primer 6 *Program ilustruje korišćenje statičkih nizova. Ispisuje 10 unetih brojeva unazad.*

```
#include <stdio.h>

main()
{
    int a[10];
    int i;
    for (i = 0; i<10; i++)
    {   printf("a[%d]=",i);
        scanf("%d",&a[i]);
    }

    printf("Unazad : \n");

    for (i = 9; i>=0; i--)
        printf("a[%d]=%d\n",i,a[i]);
}
```

Primer 7 *Napisati program koji vrši brojanje pojavljivanja svake od cifara na standardnom ulazu. Koristiti niz brojača.*

```
#include <stdio.h>
#include <ctype.h>
main()
{
    /* Niz brojaca za svaku od cifara */
    int br_cifara[10];
    int i, c;

    /* Resetovanje brojaca */
    for (i = 0; i < 10; i++)
        br_cifara[i] = 0;

    /* Citamo sa ulaza i povecavamo odgovarajuce brojace */
    while ((c = getchar()) != EOF)
        if (isdigit(c))  /* moze i if(c>='0'&& c<='9') */
            br_cifara[c-'0']++;

    /* Ispis rezultata */
    for (i = 0; i < 10; i++)
        printf("Cifra %d se pojavila %d puta.\n",i, br_cifara[i]);
}
```

Primer 8 *Program ilustruje inicijalizaciju nizova.*

```
#include <stdio.h>

main()
{
    /* Niz inicijalizujemo tako sto mu navodimo vrednosti
       u viticasnim zagradama. Dimenzija niza se odredjuje
       na osnovu broja inicijalizatora */
    int a[] = {1, 2, 3, 4, 5, 6};

    /* Isto vazi i za nizove karaktera kao i za nizove bilo kog drugog tipa.*/
    char s[] = {'a', 'b', 'c'};

    /* Ekvivalentno prethodnom bi bilo
    char s[] = {97, 98, 99};
    */

    /* Broj elemenata niza */
    int a_br_elem = sizeof(a)/sizeof(int);
    int s_br_elem = sizeof(s)/sizeof(char);

    /* Ispisujemo nizove */

    int i;
    for (i = 0; i < a_br_elem; i++)
        printf("a[%d]=%d\n",i, a[i]);
```

```
    for (i = 0; i < s_br_elem; i++)
        printf("s[%d]=%c\n",i, s[i]);
}
```

1.6 Funkcije

Veliki računski zadaci mogu se razbiti u manje delove i time se omogućava programerima 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. Funkcije se dele na dve kategorije: funkcije koje vraćaju vrednost (rezultat) i funkcije koje ne vraćaju vrednost pozivajućoj funkciji. Prilikom deklaracije funkcije navodi se tip povratne vrednosti funkcije ili tip rezultata (ako se ne navede podrazumeva se `int`), ime funkcije, lista formalnih parametara ili argumenata i telo funkcije. Kod funkcija koje ne vraćaju vrednost kao tip rezultata navodi se `void`. Funkcija ima sledeći oblik:

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

Funkcija se poziva navođenjem imena funkcije i liste stvarnih parametara (argumenata).

Među operatorima u telu funkcije se može naći operator povratka u pozivajuću funkciju `return`. Ako se on ne navede funkcija se završava izvršenjem poslednjeg operatora u telu funkcije. Ako se navede samo

```
return;
```

funkcija u kojoj se on nalazi prekida izvršavanje i prelazi se na izvršavanje sledećeg operatora pozivajuće funkcije. Ako se navede

```
return izraz;
```

tada funkcija predaje vrednost izraza pozivajućoj funkciji.

Primer 9 *Napisati funkciju koja vrši sabiranje dva cela broja i program koji testira rad ove funkcije.*

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

main() //pozivajuca funkcija
{
    /* Poziv funkcije zbir */
    int rezultat=zbir(3,5);
    printf("%d\n", rezultat);
}
```

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.

Primer 10 *Funkcija koja sabira dva broja (deklaracija stoji nezavisno od definicije).*

```
int zbir(int, int); /* Deklaracija funkcije zbir() */

main()
{
    /* Poziv funkcije */
    int rezultat=zbir(3,5);
    printf("%d\n", rezultat);
}

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

Kada se u main() funkciji pojavi operator

```
rezultat=zbir(3,5);
```

realizuju se sledeće akcije:

1. Rezerviše se memorijski prostor za promenljive definisane u funkciji zbir();
2. Formalnim parametrima se dodeljuju vrednosti stvarnih parametara: `a=3; b=5;`
3. Izvršava se telo funkcije;
4. 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().

Primer 11 *power - funkcija koja stepenuje realan broj na celobrojni izlozilac.*

```
 #include <stdio.h>

/* stepenuje x^k tako sto k puta pomnozi x */
float power(float x, int k)
{
    int i;
    float s = 1;
    for (i = 0; i<k; i++)
        s*=x;

    return s;
}

main()
{
    /* Poziv funkcije */
    float s = power(2.0,8);
    printf("%f\n", s);
}
```

Primer 12 *Verzija funkcije power koja radi i za negativne izlozioce.*

```
float power_n(float x, int k)
{
    int i;
    int negative = k<0;

    if (negative)
        k = -k;

    float s = 1;
    for (i = 0; i<k; i++)
        s*=x;

    return negative ? 1.0/s : s;
}

main()
{
    /* Poziv funkcije */
    float s = power(2.0,-1);
    printf("%f\n", s);
}
```

Primer 13 *Napisati funkciju koja izračunava zbir n-tih stepena brojeva od 1 do granice i program koji ilustruje rad ove funkcije.*

```
#include <stdio.h>
long Zbir_stepena (int n, int granica);
main()
{
    printf(" Zbir 2. stepena od 1 do 5 jeste %ld\n",Zbir_stepena(2,5));
    printf(" Zbir 3. stepena od 1 do 5 jeste %ld\n",Zbir_stepena(3,5));
    printf(" Zbir 4. stepena od 1 do 10 jeste %ld\n",Zbir_stepena(4,10));
    return 0;
}

long Zbir_stepena (int n, int granica)
{

    int i,j;   /*brojaci u for petljama */
    long Zbir=0  ,  stepenovan ;

     /*spoljasnji for ciklus obavlja sumiranja*/
    for (i=1; i<=granica; Zbir +=stepenovan, ++i)
        /*unutrasnji for ciklus obavlja stepenovanje */
         for( stepenovan=1,j=1;  j<=n;  stepenovan*=i, ++j) ;
   return Zbir;
}
```

Ako izuzmemo korišćenje operatora zarez, ova funkcija može da se napiše jednostavnije na sledeći način:

```
long Zbir_stepena (int n, int granica)
{

    int i,j;   /*brojaci u for petljama */
    long Zbir=0, stepenovan ;

     /*spoljasnji for ciklus obavlja sumiranja*/
    for (i=1; i<=granica; ++i)
    {
        stepenovan=1;
        /*unutrasnji for ciklus obavlja stepenovanje */
         for(j=1;  j<=n; ++j)
            stepenovan*=i;
         Zbir +=stepenovan;
    }
    return Zbir;
}
```

Ako iskoristimo funkciju power iz prethodnog primera, ova funkcija može da se napiše još jednostavnije:

```
long Zbir_stepena (int n, int granica)
{

    int i,j;   /*brojaci u for petljama */
    long Zbir=0;

     /*spoljasnji for ciklus obavlja sumiranja*/
    for (i=1; i<=granica; ++i)
         Zbir +=power(i,n);

    return Zbir;
}

Izlaz:
 Zbir 2. stepena od 1 do 5 jeste 55
 Zbir 3. stepena od 1 do 5 jeste 225
 Zbir 4. stepena od 1 do 10 jeste 25333
```

Primer 14 *Napisati funkciju koja izračunava zbir kvadrata brojeva od 1 do date granice kao i program koji ilustruje korišćenje date funkcije.*

```
#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+=br*br,  ++br)  ;
    printf(" Zbir kvadrata brojeva od 1 do %d jese %ld\n", n,Zbir);
}

Izlaz:
Zbir kvadrata brojeva od 1 do 5 jese 55
Zbir kvadrata brojeva od 1 do 23 jese 4324
```

Primer 15 *Napisati program u C-u koji prikazuje sve proste brojeve u datom intervalu kojima je zbir cifara složen broj. Interval se zadaje učitavanjem gornje i donje granice (dva prirodna broja). Brojeve prikazati u opadajućem poretku.*

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

int prost (int n);  /*testira da li je broj n prost broj */

/* Svi prirodni brojevi (sem 1) imaju najmanje dva delioca: jedinicu i samog sebe.
Brojevi koji nemaju drugih delioca, sem ova dva, nazivaju se prostim brojevima. */

int zbirCifara (int n);   /* vraca zbir cifara broja n */
main()
{
    int donja,gornja;  /*granice intervala */
    int i;  /*brojac u petlji */
    int pom;   /*posrednik u eventualnoj zameni */

  /*ucitavanja granice intervala */
    scanf("%d%d", &donja, &gornja);
     if (donja > gornja)   /*obezbedjivanje relacije: donja <=gornja */
    {
        pom=donja;
        donja=gornja;
         gornja=pom;
     }
     for(i=gornja;i>=donja; i--)
     if (prost (i) && !prost(zbirCifara(i) )  ) printf("%d\n",i);
}

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 n/2.  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 prost;  /*indikator slozenosti broja n */
    int i;    /*potencijalni delitelj broja n */
    if (n==1) return 0;
     /*parni brojevi razliciti od  od dva nisu prosti brojevi */
     prost= (n%2!=0) || (n==2);

     /*najmanji potencijalni kandidat za delitelje medju
```

```
      neparnim brojevima razlicitim od jedan */
     i=3;
    while ( (prost) && (i<=n/2) )
   {
    prost=n%i != 0;
    i=i+2; /*proveravamo kandidate za delitelje samo medju neparnim brojevma */
   }
 return prost;
}
int zbirCifara (int n)
{
   int  Suma=0;
   while (n>0)
   {
    Suma+= n%10;  /*dodavanje cifre tekuceg razreda,pocev od razreda jedinica ,
                    a iduci ka visim razredima cifara */
    n=n/10;      /*prelaz ka visem razredu */
   }
 return Suma;
}

Ulaz:
1 20
Izlaz:
19
17
13
```

Zadaci za praktikum:

Zadatak 1 *Sledeći program koji prepisuje standardni ulaz na standardni izlaz pokrenuti sa: (ilustracija redirekcije standardnog ulaza i izlaza)*

```
./a.out <zadatak.c
./a.out >tekst.txt
./a.out <zadatak.c >kopija.c
```

```
#include <stdio.h>

main() {
    int c;
    /* Obratiti paznju na raspored zagrada */
    while ((c = getchar()) != EOF)
        putchar(c);
}
```

Zadatak 2 *Napisati program koji prepisuje ulaz na izlaz pri čemu više blanko znakova zamenjuje jednim.*

Zadatak 3 *Napisati program koji prepisuje ulaz na izlaz pri čemu velika slova pretvara u mala a mala u velika.*

Zadatak 4 *Napisati funkciju koja izračunava broj redova u tekstu i program koji vrši njeno testiranje.*

Zadaci za vežbu:
Swith

Zadatak 5 *Napisati switch operator ekvivalentan sledećem if operatoru:*

```
if(k==0)r++;
else if(k==1)s++;
else if ((k==2)||(k==3)||(k==4)) t++;
```

Zadatak 6 *Napisati program koji za dati redni broj dana u nedelji štampa ime dana.*

Zadatak 7 *Napisati program kojim se učita znak za operaciju* `(+,_,*,/)` *i dva realna operanda, a zatim štampa rezultat.*

Zadatak 8 *Napisati program kojim se za uneti datum sa* `d, m, g` *neprestupne godine određuje datum: a) sledećeg dana; b) prethodnog dana.*

Nizovi

Zadatak 9 *Sa tastature učitati elemente niza koji su celi brojevi i za koje se pretpostavlja da ih nema više od 100. Pronaći maksimalan elemenat niza i ispisati ga na izlaz.*

Zadatak 10 *Sa tastature se unosi 15 karaktera u niz. Ispitati da li uneti niz predstavlja palindrom (primer palindroma je "anavolimilovana").*

Zadatak 11 *Ispisati prvih 15 članova Fibonačijevog niza.*

Zadatak 12 *Napisati program koji ispituje da li dva niza imaju barem jedan zajednički element.*

Zadatak 13 *Napisati program kojim se izračunava suma pozitivnih elemenata niza sa imenom* `a`*, koji se unosi sa ulaza i koji nema više od 100 elemenata. Prvo uneti broj elemenata a zatim i elemente niza.*

Zadatak 14 *Napisati program koji u datom nizu* `a`*, koji se unosi sa ulaza i nema više od 100 elemenata, menja znak svim elementima sa parnim indeksima.*

Zadatak 15 *Napisati program koji u datom nizu* `a`*, koji se unosi sa ulaza i nema više od 100 elemenata, određuje broj parnih elemenata sa neparnim indeksima.*

Zadatak 16 *Napisati program koji u datom nizu* `a`*, koji se unosi sa ulaza, sadrži samo nule i jedinice i nema više od 100 elemenata, proverava da li ima svojstvo da su svaka dva susedna elementa različita.*

Zadatak 17 *Napisati program koji na osnovu datog niza* `a`*, koji se unosi sa ulaza i nema više od 100 elemenata, formira niz b po formuli:* $b[i] = \frac{a[i]+a[2*n+1-i]}{2}$

Funkcije

Zadatak 18 *Definisati funkciju neparno(x) koja daje "tačno" ako je ceo broj x neparan. Testirati rad ove funkcije.*

Zadatak 19 *Koristeći funkciju max4() koja određuje maksimum 4 cela broja, napisati program za štampanje najveće među poslednjim ciframa brojeva a, b, c i d. Na primer, za a=35, b=140, c=127 i d=190 treba da se štampa cifra 7.*

Zadatak 20 *Napisati f-ju koja za uneti broj n izračunava zbir recipročnih vrednosti prvih n brojeva.*

Zadatak 21 *Ilustracija korišćenja funkcije za izračunavanje faktorijela celog broja.*

(a) Napisati funkciju koja izračunava faktorijel celog broja.

(b) Napisati program koji izračunava faktorijel unetog broja koristeći prethodno definisanu funkciju.

Zadatak 22 *Napisati program kojim se štampaju svi trocifreni brojevi (ako ih ima) koji su jednaki sumi faktorijela svojih cifara.*

Zadatak 23 *Ilustracija korišćenja funkcije za proveru da li je broj prost.*

(a) Napisati funkciju koja za ceo broj proverava da li je prost.

(b) Napisati program koji štampa prvih 100 prostih brojeva.

Zadatak 24 *Koristeći funkciju prost() štampa sve brojeve blizance do datog prirodnog broja n. Dva prirodna broja su blizanci ako su prosti i razlikuju se za 2 (npr. 3 i 5, 5 i 7, 11 i 13, itd.).*