Programiranje (C) - vježbe

Ajoojjj majko mila... ☹

Udari me gromom ili bar pogodi trulom pomom... ☺

Osnovni programi

- *Zadatak 1.1. :* Što ispisuju sljedeći isječci kôda (poštujte eventualne razmake, skokove u novi red i slično) te koje će vrijednosti varijable poprimiti nakon što se taj kôd izvrši?

a)
```
int a = 1, b = 2, c = 3, d = 4;
printf("%d\n%d", ((a /= 2) ? b++ : d++), --c);
```
4
2
3

b)
```
int a = 4, b = 3, c = 2, d = 1;
printf("%d\n%d", b++, ((c /= 6) ? --d : --a));
```
3
1

c)
```
int a = 4, b = 3, c = 2, d = 1;
printf("%d\n%d", ((b /= 6) ? a++ : d++), --c);
```
1
2

d)
```
int a = 1, b = 2, c = 3, d = 4;
printf("%d\n%d", ++a, ((d /= 8) ? c-- : b--));
```
2

Osnovni programi

- *Zadatak 1.2. :* Napišite program koji:

  a) učitava dva cijela broja i, ako su oba parna ispisuje njihovu sumu; inače treba ispisati produkt

  b) učitava tri realna broja i ispisuje najvećeg među njima

  c) Učitava tri cijela broja i ispisuje najbližeg nuli (najmanjeg po apsolutnoj vrijednosti)

Osnovni programi

- *Zadatak 1.2. a) :* Napišite program koji učitava dva cijela broja i, ako su oba parna ispisuje njihovu sumu; inače treba ispisati produkt

```
#include <stdio.h>

int main (void) {
    int prvibroj , drugibroj ;

    printf("Unesite prvi broj:") ;
    scanf("%d", &prvibroj ) ;
    printf("Unesite drugi broj :" ) ;
    scanf("%d" , &drugibroj ) ;

    if ( ( prvibroj % 2 == 0) && ( drugibroj % 2 == 0 ) )
            printf ("%d\n" , prvibroj + drugibroj ) ;
    else
            printf("%d\n" , prvibroj * drugibroj ) ;

    return 0 ;
}
```

Petlje

- *Zadatak 1.3. :* Napišite dio programa koji učitava cijele brojeve dok ne učita nulu. Program treba ispisati:

  a) koliko je neparnih brojeva učitano

  b) sumu svih prostih učitanih brojeva

  c) koliko je učitanih brojeva strogo veće od prvog

  d) sumu svih znamenaka svih učitanih brojeva

Petlje : *Zadatak 1.3. a) i b)*

```
scanf("%d", &x);
while (x != 0) {
    napravi što vec treba s x;
    scanf("%d", &x);
}

while (1) {
    scanf("%d", &x);
    if (x == 0) break;
    napravi što vec treba s x;
}

prost = 1;
aps_vr_od_x = (x < 0 ? -x : x) ;
for (i=2 ; i< aps_vr_od_x ; i++)
    if (x % i == 0){
            prost = 0; break;
    }
```

Petlje : *Zadatak 1.3. a) i b)*

```
int x, i, br_nep = 0 , suma_prostih = 0 , prost, aps_vr_od_x ;

while ( 1 ) {
    scanf ("%d" , &x ) ;
    if ( x == 0) break ;

    if ( x % 2 == 1) br_nep++;

    aps_vr_od_x = ( x < 0 ? -x : x ) ;
    prost = 1;
    if ( prost )
       for ( i = 2 ; i < aps_vr_od_x ; i++)
            if ( x % i == 0) {
                   prost = 0 ;
                   break ;
            }

    if ( prost ) suma_prostih += x ;
}
printf("Broj neparnih : %d\n" , br nep ) ;
printf("Suma prostih : %d\n" , suma_prostih ) ;
```

```
prost = ( aps_vr_od_x <= 1 ? 0 : 1 ) ;
```

Petlje

- *Zadatak 1.3. :* Napišite dio programa koji učitava cijele brojeve dok ne učita nulu. Program treba ispisati:

d) sumu svih znamenaka svih učitanih brojeva

Isječak rješenja (zbroj znamenki pojedinog broja):

```
scanf("%d", &x);
while (x != 0) {
   znam = x % 10;
   suma = suma + znam;
   x = x / 10 ; /* ili x /= 10 ; */
}
```

– Varijacije na temu:
  - Ispisati znamenke, ali s lijeva na desno
  - Stvoriti broj koji bi nastao okretanjem broja (npr. 15678 -> 87651)

Nizovi

- *Zadatak 1.4. :* Napišite programski isječak koji učitava prirodni broj n ≤ 17 te n cijelih brojeva i zatim:

  a) sortira i ispisuje brojeve uzlazno po vrijednosti zadnje znamenke

  b) ispisuje sve brojeve koji su veći ili jednaki zadnjem učitanom

  c) ispisuje produkt svih brojeva koji su veći ili jednaki predzadnjem učitanom (pretpostavite da je n > 1)

  d) pomoću Hornerovog algoritma izračunava i ispisuje vrijednost $p(a_1)$ gdje su a_i (i = 0 .. n-1) učitani brojevi i

$$p(x) = \sum_{i=0}^{n-1} a_i x^i$$

Nizovi : *Zadatak 1.4. a) i b)*

```
int n, i, j, a[17] ;

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

printf("Svi brojevi veci ili jednaki %d:\n" , a[n-1]);
for ( i = 0 ; i < n ; i++)
    if ( a[i] >= a[n-1] )
        printf("%d\n", a[i]);

for ( i = 0 ; i < n - 1  ; i++)
    for ( j = i + 1  ; j < n  ; j++)
        if ( a[i]%10 >= a[j]%10 ) {
            int temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
printf("Sortirani niz: \n" );
for ( i = 0 ; i < n; i++)
     printf("%d\n", a[i]);
```

Funkcije

```
tip_povratne_vrijednosti ime_funkcije(argumenti){
      tijelo funkcije
}
```

- Vrijednost se vraća s return:

```
return vrijednost_koja_se_vraca;
```

```
double max(double x, double y){
   double veci;
   veci = x > y ? x : y;
   return veci;
}
```

- void funkcije

```
void ispisi(int a, float b, char c){
   ...
   return;
```

Funkcije

- *Zadatak 1.5. :* Napišite funkciju koja kao argumente uzima dva realna broja te vraća većeg od njih. Dodatno napišite program koji prikazuje kako se funkcija upotrebljava

```
#include <stdio.h>

double max(double x, double y) {
   return (x > y ? x : y);
}

int main(void){
   double a,b;
   printf("a="); scanf("%lf", &a);
   printf("b="); scanf("%lf", &b);

   printf("Veci je %g.\n", max(a,b));
   return 0;
}
```

Funkcije

- *Napomena 1.2. :* Argumenti u funkciji su kopije onih s kojima je funkcija pozvana. Promjene vrijednosti argumenata unutar funkcije ne odražavaju se na varijable koje su zadane kao parametri prilikom poziva funkcije.

```
int f(int a){
   a++;
   return a;
}

int main(void){
   int x=17;
   printf("1: %d=" , x);
   printf("2: %d=" , f(x));
   printf("3: %d=" , x);
   return 0;
}
```

17
18
17

- Svojevrsna iznimka: nizovi

Funkcije

- *Zadatak 1.6. :* Napišite funkciju koja kao argumente uzima niz realnih brojeva i cijeli broj n (koji označava duljinu niza). Funkcija treba uzlazno sortirati niz. Napišite i kako se funkcija poziva (pretpostavite da imate učitani niz s n elemenata)

```
#include <stdio.h>

void sort(double x[] , int n) {
    int i,j;
    for ( i = 0 ; i < n - 1  ; i++)
         for ( j = i + 1  ; j < n  ; j++)
              if ( x[i] >= x[j] ) {
                 double temp = x[i];
                 x[i] = x[j];
                 x[j] = temp;
            }
}

int main(void){
    double a[50];int n;
    ...
    sort(a, n);
}
```

Funkcije

- *Zadatak 1.6. :* Napišite funkciju koja kao argumente uzima niz realnih brojeva i cijeli broj n (koji označava duljinu niza). Funkcija treba uzlazno sortirati niz. Napišite i kako se funkcija poziva (pretpostavite da imate učitani niz s n elemenata)

```
int main(void){
   double a[50];int n;
   ...
   sort(a, n);
}
```

- Česte greške:

```
sort(a[], n);
sort(a[n], n);
sort(a[0], n);
```

Funkcije

- *Zadatak 1.7. :* Napišite funkciju koja kao parametre prima prirodne brojeve a i b, a,b > 1 , a kao rezultat ne vraća ništa. Funkcija treba "nacrtati" graf za sve brojeve k takve da je b < k ≤ a + b, koji u svakom retku ispisuje broj k, te f(k) znakova # pri čemu se s f(k) označava broj brojeva x takvih da je a ≤x ≤ b i k djeljiv s x (f(k) služi za opis zadatka te ga ne morate definirati kao posebnu funkciju), te preko trećeg parametra vratiti najveći broj znakova # ispisanih u jednom retku (ako nije ispisan niti jedan znak, treba vratiti nulu).

Npr. za a=3 , b=9, funkcija treba ispisati

`10: #` *(od brojeva iz skupa {3,4,…,9} broj 10 je djeljiv samo s brojem 5)*

`11:` *(broj 11 nije djeljiv s niti jednim brojem iz skupa {3,4,…,9})*

`12: ###` *(od brojeva iz skupa {3,4,…,9} broj 12 je djeljiv samo s 3,4 i 6)*

Te preko trećeg parametra vratiti vrijednost 3 (= max{1,0,3})

Napomena: Nije dozvoljeno korištenje funkcija iz math.h i nizova!

Funkcije – Zadatak 1.6

```
int f ( int k , int a , int b) {
    int cnt = 0 , x ;
    for ( x = a ; x <= b ; x++)
       if ( k % x == 0) cnt++;
    return cnt ;
}
void zad ( int a , int b , int *br ) {
    int k , i ;
    *br = 0 ;
    for ( k = b + 1 ; k <= a + b ; k++) {
            int fk = f (k , a , b ) ;
            if ( fk > *br ) *br = fk ;
            printf ( "%d : " , k ) ;
            for ( i = 0 ; i < fk ; i++) printf("#");
            printf( "\n" ) ;
    }
}
int main(void) {
    int res;
    zad(3, 9, &res);
    printf("Rezultat: %d\n", res);
    return 0;
}
```

Zadaci za samostalnu vježbu

Zadatak 1.8.

Napišite program (ne samo dio programa!) koji učitava niz cijelih brojeva x dok ne učita broj 17 ili ukupno 314 brojeva. Program treba niz x sortirati silazno prema sumi druge (s lijeva) i zadnje znamenke (ako broj nema neku od traženih znamenaka, za njenu vrijednost se uzima nula), te ispisati tako dobiveni niz.

Rekurzije - uvod

```
#include <stdio.h>
int a=1 , b=2 , c=3 ,d=4;

void f ( int a ) {
    int b = 12 ;
    printf(" a=%d , b=%d , c=%d , d=%d\n" , a , b , c , d ) ;
    a++; b++; c++;
    printf(" a=%d , b=%d , c=%d , d=%d\n" , a , b , c , d ) ;
}

int main ( void) {
    int a = 21 , b = 22 , d = 24 ;

    printf("main() 1.put :") ;
    printf("a=%d , b=%d , c=%d , d=%d\n" , a , b , c , d ) ;
    printf("Funkcija 1.put : \n") ;
    f(c) ;
    printf("Funkcija 2.put : \n") ;
    f(c) ;
    printf("main() 2.put :") ;
    printf("a=%d , b=%d , c=%d , d=%d\n" , a , b , c , d ) ;

    return 0 ;
}
```

```
main() 1.put : a=21 , b=22 , c=3 , d=24
Funkcija 1.put :
 a=3 , b=12 , c=3 , d=4
 a=4 , b=13 , c=4 , d=4
Funkcija 2.put :
 a=4 , b=12 , c=4 , d=4
 a=5 , b=13 , c=5 , d=4
main() 2.put : a=21 , b=22 , c=5 , d=24
```

Rekurzije - uvod

```
#include <stdio.h>
int glob_a = 1 , glob_b = 2 , glob_c = 3 , glob_d = 4 ;

void f (int f_a) {
     int f_b = 12 ;
     printf("a=%d , b=%d , c=%d , d=%d\n",f_a, f_b, glob_c, glob_d );
     f_a++; f_b++; glob_c++;
     printf("a=%d , b=%d , c=%d , d=%d\n",f_a, f_b, glob_c, glob_d );
}

int main ( void) {
    int main_a = 21 , main_b = 22 , main_d = 24 ;
    printf( "main() 1.put :" ) ;
    printf( " a=%d , b=%d , c=%d , d=%d\n" ,
                        main_a, main_b, glob_c, main_d );
    printf( "Funkcija 1. put : \n" ) ;
    f(glob_c) ;
    printf( "Funkcija 2. put : \n" ) ;
    f(glob_c) ;
    printf( "main ( ) 2. put : \n" ) ;
    printf( " a=%d , b=%d , c=%d , d=%d\n" ,
                        main_a, main_b, glob_c, main_d );
    return 0 ;
}
```

Rekurzije

- *Primjer 2.1. :* Fibonaccijeve brojeve definiramo sljedećom formulom:

$$F_n := \begin{cases} 0 & , n = 0, \\ 1 & , n = 1, \\ F_{n-2} + F_{n-1}, & n \in \mathbb{N},\ n > 1. \end{cases}$$

Funkcija za računanje n-tog Fibonaccijevog broja direktno prema definiciji (rekurzivno)

```
long int fib(long int n) {
   if (n <= 1) return n ;
   return fib(n-1) + fib(n-2);
}
```

Evaluacija rekurzija

- *Zadatak 2.1. :* Koju vrijednost vraća poziv funkcije fib(5), pri čemu je funkcija fib() ona koju smo definirali u primjeru 2.1?

- Crtamo tijek izvođenja:
  - \+ ulazak u funkciju
  - \ kraj funkcijskog poziva
  - | tijek funkcijskog poziva

Rekurzije

- *Napomena 2.2. :* Fibonaccijeve brojeve definiramo sljedećom formulom:

$$F_n := \begin{cases} 0 & , n = 0, \\ 1 & , n = 1, \\ F_{n-2} + F_{n-1}, & n \in \mathbb{N},\ n > 1. \end{cases}$$

Nerekurzivno rješenje:

```
long int fib ( int n) {
   long int fib1 = 0 , fib2 = 1 , f i b 3 ;
   int i ;
   if (n <= 1) return n ;
   for ( i = 2 ; i <= n ; i++) {
      fib3 = fib1 + fib2 ;
      fib1 = fib2 ;
      fib2 = fib3 ;
   }
   return fib2 ;
}
```

Kreiranje rekurzija

- *Napomena 2.3. :* Svaka rekurzija mora imati i terminalne uvjete, tj. način izvršavanja koji nema rekurzivnih poziva!

```
long int fib(long int n) {
  if (n <= 1) return n ;
  return fib(n-1) + fib(n-2);
}
```

Što bi bilo da je pisalo `if (n==0) return 0;` ?

fib(2) = fib(1) + fib(0)
= fib(0) + fib(-1) + fib(0)
= 0 + fib(-2) + fib(-3) + fib(0)
= ...

Kreiranje rekurzija

- *Zadatak 2.2. :* Napišite rekurzivnu funkciju f(a,b) koja je definirana sljedećom formulom:

$$f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$$

$$f(a,b) = \begin{cases} 1 & , a = b = 0, \\ -f(-a,b) & , a < 0, b \geq 0, \\ -f(a,-b) & , a \geq 0, b < 0, \\ f(-a,-b) & , a < 0, b < 0, \\ f(b,a-1)+2, & inače. \end{cases}$$

```
int f ( int a , int b) {
   if ( a == 0 && b == 0) return 1 ;
   else if ( a < 0  && b >= 0) return –f (–a , b ) ;
   else if ( a >= 0 &&  b < 0) return –f ( a, –b ) ;
   else if ( a < 0  &&  b < 0) return f (–a, –b ) ;
   else return f (b , a – 1) + 2 ;
}
```

Krejranje rekurzija

- *Zadatak :* Napišite rekurzivnu funkciju:

```
int suma(int niz[],int n)
```

koja će izračunati sumu elemenata u nizu.

```
int suma(int niz[], int n){
    if (n==0)
        return 0;
    else
        return niz[n-1] + suma(niz, n-1);
}

int main(void) {
    int niz[] = {1,2,7,12,15};
    printf("%d", suma(niz, 5));
    return 0;
}
```

Kreiranje rekurzija

- *Zadatak 2.4:* Napišite rekurzivnu funkciju koja za skup cijelih brojeva (zadan pomoću niza) vraća broj podskupova čija je suma djeljiva sa 17. Uz niz i njegovu duljinu, funkcija smije primati i druge argumente.

  Na primjer, za skup {5,7,12}, funkcija treba vratiti 2, jer traženi uvjet zadovoljavaju podskupovi Ø (suma elemenata je nula) i {5,12}.

  Obavezno napišite i kako se funkcija poziva.

5 je u podskupu
- 7 je u podskupu
  - 12 je u podskupu => suma = 5 + 7 + 12
  - 12 nije u podskupu => suma = 5 + 7
- 7 nije u podskupu
  - 12 je u podskupu => suma = 5 + 12
  - 12 nije u podskupu => suma = 5

5 nije u podskupu
- 7 je u podskupu
  - 12 je u podskupu => suma = 7 + 12
  - 12 nije u podskupu => suma = 7
- 7 nije u podskupu
  - 12 je u podskupu => suma = 12
  - 12 nije u podskupu => suma = 0

Kreiranje rekurzija – zadatak 2.4.

```
int rek ( int niz[ ] , int n , int suma ) {
    if (n) return
           rek (niz , n-1, suma + niz[n-1]) +
           rek (niz , n-1, suma ) ;

    return ( suma % 17 ? 0 : 1 ) ;
}

int main(void){
    int niz[] = {-12, -11, -5, -2, 1, 3, 5, 8};
    int duljina_niza = 8;

    printf("Takvih podskupova ima %d.\n",
           rek(niz, duljina_niza, 0));
    return 0;
}
```

Kreiranje rekurzija – zadatak 2.4.

```
int rek ( int niz[ ] , int n , int suma, int podskup[], int br_elem ) {
        if (n){
                int rez = 0;
                podskup[n-1] = 1;
                rez += rek (niz , n-1, suma + niz[n-1], podskup, br_elem);
                podskup[n-1] = 0;
                rez += rek (niz , n-1, suma, podskup, br_elem ) ;
                return rez;
        }
        if (suma%17) return 0;
        else{
                int i;
                for(i=0; i<br_elem ; i++)
                        if (podskup[i]) printf("%d ", niz[i]);
                printf("\n");
                return 1;
        }
}
int main(void){
        int niz[] = {-12, -11, -5, -2, 1, 3, 5, 8};             int podskup[8] = {0};int duljina_niza = 8;
        printf("Takvih podskupova ima %d.\n",
                rek(niz, duljina_niza, 0, podskup, duljina_niza));
        return 0;
}
```

Kreiranje rekurzija

- *Zadatak 2.5: (pismeni ispit 28.11.2005.)* Napišite rekurzivnu funkciju koja uzima barem jedan argument x (iz N). Funkcija treba vratiti broj na koliko različitih načina se x može prikazati kao suma brojeva 2,3 i 5, neovisno o redoslijedu sumanada (tj. 2+3+3 je isto što i 3+2+3 i 3+3+2, pa se to broji kao jedan način). Napišite i program kojim se testira funkcija (treba samo učitati broj, pozvati funkciju i ispisati rezultat.

  Napomena: Uz argument x, funkcija smije primati dodatne pomoćne argumente, ali nije dozvoljeno korištenje polja, lista te globalnih i static varijabli!

  – *Prvo rješavamo varijantu u kojoj brojimo sve mogućnosti (i one redundantne)*

17 = 2 + 15 ili 17 = 3 + 14 ili 17 = 5 + 12

part(17) = part(15) + part(14) + part(12)
= part(13) + part(12) + part(10) + ...

općenito x = 2 + (x-2) ili x = 3 + (x-3) ili x = 5 + (x-5)
part(x) = part(x-2) + part(x-3) + part(x-5)

Kreiranje rekurzija – zadatak 2.5.

```
part(5) = part(3) + part(2) + part(0)
    = part(1) + part(0) + part(-2)+
      part(0) + part(-1)+
      part(-3)+ part(0)
    = part(-1) + part(-2) + part(-4) + part(0) + part(-2) +
      part(0) + part(-1) +
      part(-3) + part(0)
    = 0 + 0 + 0 + 1 + 0          + 1 + 0        + 0 + 1
```

```
#include <stdio.h>
int part (int x) {
    if ( x < 0) return 0 ; /* ovaj rastav nije moguc */
    if ( x == 0) return 1 ; /* rastav uspjesan */
    return part(x-2) + part(x-3) + part(x-5) ;
}

int main (void) {
 int n ;
 printf ("Upisite broj n:"); scanf ( "%d" , &n ) ;
 printf ("Takvih rastava ima %d.\n" , part(n) ) ;
 return 0 ;
}
```

Kreiranje rekurzija – zadatak 2.5.

- Brojeve u rastavu ćemo složiti uzlazno. Prenosimo varijablu koja označava najmanji broj koji možemo upotrijebiti

```
#include <stdio.h>
int part (int x, int prvi) {
    int cnt=0;
    if ( x < 0) return 0 ; /* ovaj rastav nije moguc */
    if ( x == 0) return 1 ; /* rastav uspjesan */
    if (prvi<=2) cnt += part(x-2, 2);
    if (prvi<=3) cnt += part(x-3, 3);
    if (prvi<=5) cnt += part(x-5, 5);
    return cnt;
}

int main ( void) {
 int n ;
 printf ("Upisite broj n:"); scanf ( "%d" , &n ) ;
 printf ("Takvih rastava ima %d.\n" , part(n,2) ) ;
 return 0 ;
}
```

Kreiranje rekurzija

- *Zadatak 2.6: (sličan 2.5.)* Napišite rekurzivnu funkciju koja uzima barem jedan argument x (iz N). Funkcija treba ispisati sve različite načine na koje se x može prikazati kao suma brojeva 2,3 i 5, neovisno o redoslijedu sumanada (tj. 2+3+3 je isto što i 3+2+3 i 3+3+2, pa se to broji kao jedan način). Napišite i program kojim se testira funkcija (treba samo učitati broj, pozvati funkciju i ispisati rezultat.

  Napomena: Uz argument x, funkcija smije primati dodatne pomoćne argumente, ali nije dozvoljeno korištenje globalnih i static varijabli!

  – Uputa: uvesti dodatni niz u koji će se spremati sumandi + pratiti duljinu tog niza.

    - Bolja varijanta : u niz spremati broj pojavljivanja svakog od sumanada.

Kreiranje rekurzija – zadatak 2.6

```
void part (int x, int prvi, int sumandi[], int duljina) {
    if ( x < 0) return; /* ovaj rastav nije moguc */
    if ( x == 0){
       int i;
       for(i=0; i<duljina ; i++) printf("%d ", sumandi[i]);
       printf("\n");
       return;
    }
    if (prvi<=2){
       sumandi[duljina]=2;
       part(x-2, 2, sumandi, duljina+1);
    }
    if (prvi<=3){
       sumandi[duljina]=3;
       part(x-3, 3, sumandi, duljina+1);
    }
    if (prvi<=5){
       sumandi[duljina]=5;
       part(x-5, 5, sumandi, duljina+1);
    }
}
```

Kreiranje rekurzija – zadatak 2.6. – varijanta 2

```
void part (int x, int prvi, int sumandi[]) {
    if ( x < 0) return; /* ovaj rastav nije moguc */
    if ( x == 0){
       int i;
       for(i=0; i<sumandi[0] ; i++) printf("%d ", 2);
       for(i=0; i<sumandi[1] ; i++) printf("%d ", 3);
       for(i=0; i<sumandi[2] ; i++) printf("%d ", 5);
       printf("\n");
       return;
    }
    if (prvi<=2){
       sumandi[0]++; part(x-2, 2, sumandi); sumandi[0]--;
    }
    if (prvi<=3){
       sumandi[1]++; part(x-3, 3, sumandi); sumandi[1]--;
    }
    if (prvi<=5){
       sumandi[2]++; part(x-5, 5, sumandi); sumandi[2]--;
    }
}
```

Kreiranje rekurzija

- *Zadatak 2.7:* Napišite rekurzivnu verziju Euklidovog algoritma za računanje najvećeg zajedničkog djelitelja (eng. GCD, greatest common divisor) dva prirodna broja.

```
int gcd ( int a , int b) {
   if (b == 0)
       return a ;
   else
       return gcd (b , a % b ) ;
}
```

ili kraće:

```
int gcd ( int a , int b) {
   return (b ? gcd (b , a % b) : a ) ;
}
```

Kreiranje rekurzija – zadatak za DZ

- *Zadatak 2.8 (pismeni ispit 27.6.2005.)* Napišite rekurzivnu funkciju koja uzima barem jedan argument x iz N. Funkcija treba vratiti broj na koliko različitih načina se x može prikazati kao umnožak prirodnih brojeva većih od 1.

  Na primjer, za x = 12, funkcija treba vratiti 4

  ```
  (2*2*3 , 2*6 , 3*4 , 12)
  ```

  Napišite i program kojim se testira funkcija (treba samo učitati broj, pozvati funkciju i ispisati rezultat.

  Napomena: Uz argument x, funkcija smije primati dodatne pomoćne argumente, ali nije dozvoljeno korištenje polja, lista te globalnih i static varijabli!

Kreiranje rekurzija

- *Zadatak 2.9* Napišite funkciju koja kao argument prima niz cijelih brojeva duljine n = 3^k za neki prirodni broj k (ne treba provjeravati da je duljina zaista potencija broja 3) te eventualne pomoćne argumente (koje možete sami odabrati). Funkcija treba srednju trećinu niza ispuniti nulama, a rubne jedinicama. Zatim za rubne trećine treba primijeniti isti postupak (dok te trećine ne postanu jednočlane). Vrijednosti niza koje treba dobiti za neke vrijednosti n su:

n	Niz
3	$1, 0, 1$
9	$\underbrace{1, 0, 1}_{3}, \underbrace{0, 0, 0}_{3}, \underbrace{1, 0, 1}_{3}$
27	$\underbrace{1, 0, 1, 0, 0, 0, 1, 0, 1}_{9}, \underbrace{0, 0, 0, 0, 0, 0, 0, 0, 0}_{9}, \underbrace{1, 0, 1, 0, 0, 0, 1, 0, 1}_{9}$

Napišite i kako se funkcija poziva

Kreiranje rekurzija

- *Zadatak 2.10 :* Napišite program koji testira funkciju iz prethodnog zadatka za vrijednosti 3, 9, 27, 81, 243

```
void niz ( int x[], int from, int to ) {
     int i;
     if (from == to ) {
              x[from] = 1;
              return;
     }
     for ( i = from ; i <= to ; i++) x[i] = 0 ;
     niz(x , from , (2*from+to)/3 ) ;
     niz(x , (from+2*to)/3 + 1 , to ) ;
}
int main ( void) {
     int x [243] , i , n  = 3;
     while (n <= 243) {
              niz (x , 0 , n - 1 );
              printf ("%3d : %d" , n , x[0] ) ;
              for ( i = 1 ; i < n ; i++) printf ("%d" , x [i] ) ;
                       printf("\n");
              n*= 3 ;
     }
     return 0 ;
 }
```

Kreiranje rekurzija

- *Zadatak 2.12 :* Riješite zadatak 2.9 tako da funkcije ne kreira niz, nego ispisuje tražene elemente na ekran, bez upotrebe nizova

```
void niz (int from, int to ) {
     int i;
     if (from == to ) {
              printf("1"); return;
     }
     niz(from , (2*from+to)/3 ) ;
     for ( i = (2*from+to)/3 + 1 ; i <= (from+2*to)/3 ; i++)
              printf("0");
     niz((from+2*to)/3 + 1 , to ) ;
}

int main ( void) {
     int i , n  = 3;
     while (n <= 243) {
              printf ("%3d : " , n) ;
              niz (0 , n - 1 );
              printf("\n");
              n*= 3 ;
     }
     return 0 ;
 }
```

Kreiranje rekurzija

- *Zadatak 2.13 (pismeni ispit 1.9.2004.)* : Žaba želi prijeći rijeku skačući preko n listova lopoča. To radi u skokovima po dva ili tri lista prema naprijed ili prema natrag. Povratka na kopno nema (dakle, ne može otići "ispred" prvog lista). Također ne može skočiti niti iza zadnjeg lista. Skakutanje je gotovo kad žaba dođe na n-ti list. Napišite rekurzivnu funkciju koja za zadani n vraća broj načina kojima žaba može izvesti opisano skakanje u najviše 17 koraka. Treba napisati i kako se funkcija poziva.

```
int zaba (int n, int pozicija, int korak) {
    if (korak > 17) return 0;
    if (pozicija > n || (korak !=1 && pozicija < 1))
            return 0;

    if (pozicija == n){
            return 1;
    }
    return
            zaba(n, pozicija+2, korak+1)
            + zaba(n, pozicija+3, korak+1)
            + zaba(n, pozicija-2, korak+1)
            + zaba(n, pozicija-3, korak+1);
}

printf("Broj nacina=%d\n", zaba(n, 0, 1));
```

Statičke varijable

- Zadržavaju vrijednost prilikom izlaska iz funkcije

```
void f(){
   static int a=2;int b=2;
   a++;b++;
   printf("%d %d", a, b);
}

f(); f(); f(); će ispisati 3 3 4 3 5 3
```

- Razlikovati

```
static int a=2
i
static int a;
a=2;
```

Višedimenzionalna polja

```
int x[10];
int y[10][20];
int z[15][25][35];
...
printf("%d", y[7][13]);
scanf("%d", &y[2][0]);
scanf("%d", &z[6][24][10]);
```

Kod prosljeđivanja višedimenzionalnih polja u funkciju, u zaglavlju funkcije se moraju navesti dimenzije polja osim prvog.

```
void funkcija(int z[][25][35]);
```

Višedimenzionalna polja

Deklaracija i istovremena inicijalizacija dvodimenzionalnog polja

```
int a[3][5];
int a[3][5] =
   {{1,2,3,4,5},{6,7,8,9,10},{2,3,4,5,6}};
int a[][5] = {{1,2,3,4,5},{6,7,8,9,10},{2,3,4,5,6}};
int a[3][5] = {{1,2,3,4,5},{6,7,8,9,10}};
int a[3][5] = {{1,2,3},{6,7,8,9}};
int a[][5] = {{1,2,3},{6,7,8,9},{10,11}};
int a[3][5] = {1,2,3,4,5,6,7};
int a[3][5] = {0};
```

Višedimenzionalna polja

- *Zadatak 3.1.* Napišite program koji učitava dva prirodna broja m, n ≤ 10, te matrice a, b $\in \mathbb{N}^{m \times n}$. Program treba izračunati sumu matrice c := a+b i ispisati ju tablično; možete pretpostaviti da će svi učitani brojevi imati najviše 5 znamenaka).

```
int main(void) {
  int a[10][10], b[10][10], c[10][10], m, n, i, j;

  printf("m = "); scanf("%d", &m); printf("n = "); scanf("%d", &n);

  ucit_mat(a, m, n, 'a'); ucit_mat(b, m, n, 'b');

  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      c[i][j] = a[i][j] + b[i][j];

  printf("Rezultat c =\n");
  for (i = 0; i < m; i++) {
    for (j = 0; j < n; j++)
      printf("%6d", c[i][j]);
    printf("\n");
  }
  return 0;
}
```

Višedimenzionalna polja

- *Zadatak 3.1.* Napišite program koji učitava dva prirodna broja m, n ≤ 10, te matrice a, b $\in \mathbb{N}^{m \times n}$, i treba izračunati sumu matrice c := a+b i ispisati ju tablično (možete pretpostaviti da će svi učitani brojevi imati najviše 5 znamenaka).

```
void ucit_mat(int x[][10], int m, int n, char ime) {
  int i, j;
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++) {
      printf("%c[%d][%d] = ", ime, i, j);
      scanf("%d", &x[i][j]);
    }
}
```

Višedimenzionalna polja

- *Zadatak 3.3.* Neka je u varijablu x učitana kvadratna matrica realnih brojeve reda n. Napišite dio programa koji računa i ispisuje trag te matrice

```
int i; double sum = 0;
for (i = 0; i < n; i++) sum += a[i][i];
printf("tr a = %g\n", sum);
```

- Točno, ali loše rješenje (kvadratna složenost) bi bilo sljedeće:

```
int i, j;double sum = 0;
for (i = 0; i < n; i++)
  for (j = 0; j < n; j++)
    if (i == j) sum += a[i][j];
printf("tr a = %g\n", sum);
```

Višedimenzionalna polja

- *Zadatak 3.4.* Napišite program koji učitava n i kvadratnu matricu reda n te ispisuje produkt elemenata na njenoj sporednoj dijagonali. Pokušajte postići da program ima linearnu složenost.

```
#include <stdio.h>
int main(void) {
   int a[10][10], i, m, n, prod=1;

   printf("m = "); scanf("%d", &m);
   printf("n = "); scanf("%d", &n);

   for (i = 0; i < m; i++)
     for (j = 0; j < n; j++) {
        printf("%a[%d][%d] = ", i, j); scanf("%d", &a[i][j]);
     }

   for(i=0; i<n ; i++)
      prod *= a[i][n-1-i];

   printf("Produkt=%d\n", prod);

   return 0;
}
```

Višedimenzionalna polja

- *Zadatak 3.5.* Zadano je trodimenzionalno polje nula i jedinica dimenzije n (reprezentacija diskretizirane kocke u trodimenzionalnom prostoru). Napišite dio programa koji će ispisati koliko ima jedinica:
  - u vrhovima kocke
  - na bridovima kocke (bez vrhova)
  - na stranama kocke (bez vrhova i bridova)
  - u unutrašnjosti kocke (bez vrhova, bridova i strana)
  - na glavnoj dijagonali kocke

- Uputa:

  Ako je matrica pohranjena u varijablu a, vrhovi kocke se nalaze u

```
a[0][0][0] , a[0][0][n-1] , a[0][n-1][0] , a[0][n-1][n-1],
a[n-1][0][0],a[n-1][0][n-1],a[n-1][n-1][0],a[n-1][n-1][n-
   1]
```

Zadatak 3.5.

Ako je matrica pohranjena u varijablu a, vrhovi kocke se nalaze u
`a[0][0][0] , a[0][0][n-1] , a[0][n-1][0] , a[0][n-1][n-1],`
`a[n-1][0][0],a[n-1][0][n-1],a[n-1][n-1][0],a[n-1][n-1][n-1]`

```
int i, j, k;
int cnt_vrhovi = 0;
for (i = 0; i < n; i += n-1)
  for (j = 0; j < n; j += n-1)
    for (k = 0; k < n; k += n-1)
      cnt_vrhovi += a[i][j][k];
```

ili

```
int i, j, k;
int cnt_vrhovi = 0;

for (i = 0; i < 2; i++)
  for (j = 0; j < 2; j++)
    for (k = 0; k < 2; k++)
      cnt_vrhovi += a[i*(n-1)][j*(n-1)][k*(n-1)];
```

Višedimenzionalna polja

- *Zadatak 3.5.* Zadano je trodimenzionalno polje nula i jedinica dimenzije n (reprezentacija diskretizirane kocke u trodimenzionalnom prostoru). Napišite dio programa koji će ispisati koliko ima jedinica:
  - na bridovima kocke (bez vrhova)

```
for(i=0; i<n ; i+= n-1){
   for(j=0; j<n ; j+=n-1){
         for(k=1 ; k<n-1 ; k++){

               cnt_bridovi += a[i][j][k];
               cnt_bridovi += a[i][k][j];
               cnt_bridovi += a[k][j][i];
         }
   }
}
```

Višedimenzionalna polja

- *Zadatak 3.5.* Zadano je trodimenzionalno polje nula i jedinica dimenzije n (reprezentacija diskretizirane kocke u trodimenzionalnom prostoru). Napišite dio programa koji će ispisati koliko ima jedinica:
  - na stranama kocke (bez vrhova i bridova)

```
for(i=0; i<n ; i+= n-1){
    for(j=1; j<n-1 ; j++){
            for(k=1 ; k<n-1 ; k++){

                    cnt_stranice += a[i][j][k];
                    cnt_stranice += a[j][i][k];
                    cnt_stranice += a[k][j][i];
            }
    }
}
```

Višedimenzionalna polja

- *Zadatak 3.5.* Zadano je trodimenzionalno polje nula i jedinica dimenzije n (reprezentacija diskretizirane kocke u trodimenzionalnom prostoru). Napišite dio programa koji će ispisati koliko ima jedinica:
  - u unutrašnjosti kocke (bez vrhova, bridova i strana)
  - na glavnoj dijagonali kocke

```
for(i=1; i<n-1 ; i++)
    for(j=1; j<n-1 ; j++)
            for(k=1 ; k<n-1 ; k++)
                    cnt_unutra += a[i][j][k];

for(i=0; i<n ; i++)
    cnt_gl_dij += a[i][i][i];
```

Višedimenzionalna polja

- *Zadatak 3.9.:* Napišite dio programa koji za učitanu kvadratnu matricu realnih brojeva reda n provjerava je li ona donje trokutasta.

```
int d_trokut = 1, i, j;

for (i = 0; i < n; i++)
  for (j = i + 1; j < n; j++)
    if (a[i][j]) {
      d_trokut = 0;
      break;
    }

printf("Matrica %sje donje trokutasta.\n",
  d_trokut ? "" : "ni");
```

Višedimenzionalna polja

- *Zadatak 3.9.:* Napišite program koji učitava cijele brojeve `i` i `j` (`0≤i<10, 0≤j<10`) te kreira i ispisuje tablicu M s 10 x 10 znakova koja na svim mjestima ima točkice, osim na horizontalnoj i vertikalnoj liniji koje prolaze elementom M[i][j] (na te linije treba staviti zvjezdice). Npr. za `i=1, j=2`, tablica treba izgledati ovako:

```
. . * . . . . . . . .
* * * * * * * * * * *
. . * . . . . . . . .
. . * . . . . . . . .
. . * . . . . . . . .
. . * . . . . . . . .
. . * . . . . . . . .
. . * . . . . . . . .
. . * . . . . . . . .
. . * . . . . . . . .
```

Zadatak 3.9.

```
#include <stdio.h>
int main(void) {
  char a[10][10];
  int i, j, k, l;

  printf("i = "); scanf("%d", &i); printf("j = "); scanf("%d", &j);

  for (k = 0; k < 10; k++)
    for (l = 0; l < 10; l++)
      a[k][l] = '.';

  for (k = 0; k < 10; k++) {
    a[i][k] = '*';
    a[k][j] = '*';
  }

  for (k = 0; k < 10; k++) {
    for (l = 0; l < 10; l++)
      printf("%2c", a[k][l]);
    printf("\n");
  }
  return 0;
}
```

Višedimenzionalna polja

- *Zadatak 3.11.:* Napišite dio programa koji ispisuje koliko se prostih brojeva nalazi u učitanoj matrici sa m redaka i n stupaca.

```
int i, j, k, cnt = 0;

for (i = 0; i < m; i++)
  for (j = 0; j < n; j++) {
    for (k = 2; k < a[i][j]; k++)
      if (!(a[i][j] % k)) break;
    if (k == a[i][j]) cnt++;
  }

printf("Broj prostih brojeva u matrici: %d\n", cnt);
```

Višedimenzionalna polja

- *Zadatak 3.13.:* Neka je učitana matrica x sa m redaka i n stupaca. Napišite dio programa koji ispisuje indeks **retka** s najvećom sumom elemenata. Ako takvih ima više, dovoljno je ispisati indeks jednog od njih.

```
int i, j, max, maxi;

for (i = 0; i < m; i++) {
  int sum = 0;
  for (j = 0; j < n; j++) sum += x[i][j];
  if (i == 0 || sum > max) {
    max = sum;
    maxi = i;
  }
}
printf("Trazeni redak=%d", maxi);
```

Višedimenzionalna polja

- *Zadatak 3.14.:* Neka je učitana matrica x sa m redaka i n stupaca. Napišite dio programa koji ispisuje indeks stupca s najvećom sumom elemenata. Ako takvih ima više, dovoljno je ispisati indeks jednog od njih.

```
int i, j, max, maxj;

for (j = 0; j < n; i++) {
  int sum = 0;
  for (i = 0; i < m; i++) sum += x[i][j];
  if (j == 0 || sum > max) {
    max = sum;
    maxj = j;
  }
}
printf("Trazeni stupac=%d", maxj);
```

Višedimenzionalna polja

- *Zadatak 3.15.:* Neka je učitana matrica x sa m redaka i n stupaca. Napišite dio programa koji ispisuje **indekse redaka** s najvećom sumom elemenata.

```
int i, j, max, maxi;

for (i = 0; i < m; i++) {
  int sum = 0;
  for (j = 0; j < n; j++) sum += x[i][j];
  if (i == 0 || sum > max) max = sum;
}

printf("indexi redaka s najvecom sumom:\n");

for (i = 0; i < m; i++) {
  int sum = 0;
  for (j = 0; j < n; j++) sum += x[i][j];
  if (sum == max) printf("%d\n", i);
}
```

Višedimenzionalna polja

- *Zadatak 3.18.:* Napišite dio programa koji transponira elemente matrice m x n (pri tome je potrebno prilagoditi i dimenzije matrice tako da odgovaraju novonastaloj matrici.

Pokušaj rješenja:

```
for (i = 0; i < m; i++) {
  for (j = 0; j < n; j++) {
    double tmp = x[i][j];
    x[i][j] = x[j][i];
    x[j][i] = tmp;
  }
}
tmp = m; m = n; n = tmp;
```

Višedimenzionalna polja

- *Zadatak 3.18.:* Napišite dio programa koji transponira elemente matrice m x n (pri tome je potrebno prilagoditi i dimenzije matrice tako da odgovaraju novonastaloj matrici.

```
for (i = 0; i < m; i++) {
  for (j = i + 1; j < n; j++) {
    double tmp = x[i][j];
    x[i][j] = x[j][i];
    x[j][i] = tmp;
  }
}
tmp = m; m = n; n = tmp;
```

Rješenje je još uvijek krivo!

Provjerite što se događa kad je m=2 i n=3, a što kad je m=3, a n=2

Višedimenzionalna polja

- *Zadatak 3.18.:* Napišite dio programa koji transponira elemente matrice m x n (pri tome je potrebno prilagoditi i dimenzije matrice tako da odgovaraju novonastaloj matrici.

```
int max = m>n ? m : n
 for (i = 0; i < max ; i++) {
   for (j = i + 1; j < max ; j++) {
     double tmp = x[i][j];
     x[i][j] = x[j][i];
     x[j][i] = tmp;
   }
 }
 tmp = m; m = n; n = tmp;
```

Varijabilni argumenti funkcija

- Parametri se prenose “po vrijednosti”.
- Funkcija ima lokalnu varijablu u kojoj se nalazi kopija vrijednosti.
- Promjene argumenata u funkciji ne odražavaju se na vrijednosti varijabli na mjestu s kojeg je funkcija pozvana

- Primjer 4.1:

```
#include <stdio.h>
void f(int a) {
  printf("Funkcija 1: a = %d\n", a);
  a++;
  printf("Funkcija 2: a = %d\n", a);
}

int main(void) {
  int x = 1;

  printf("Gl. prog. 1: x = %d\n", x);
  f(x);
  printf("Gl. prog. 2: x = %d\n", x);

  return 0;
}
```

```
Gl. prog. 1: x = 1
Funkcija 1: a = 1
Funkcija 2: a = 2
Gl. prog. 2: x = 1
```

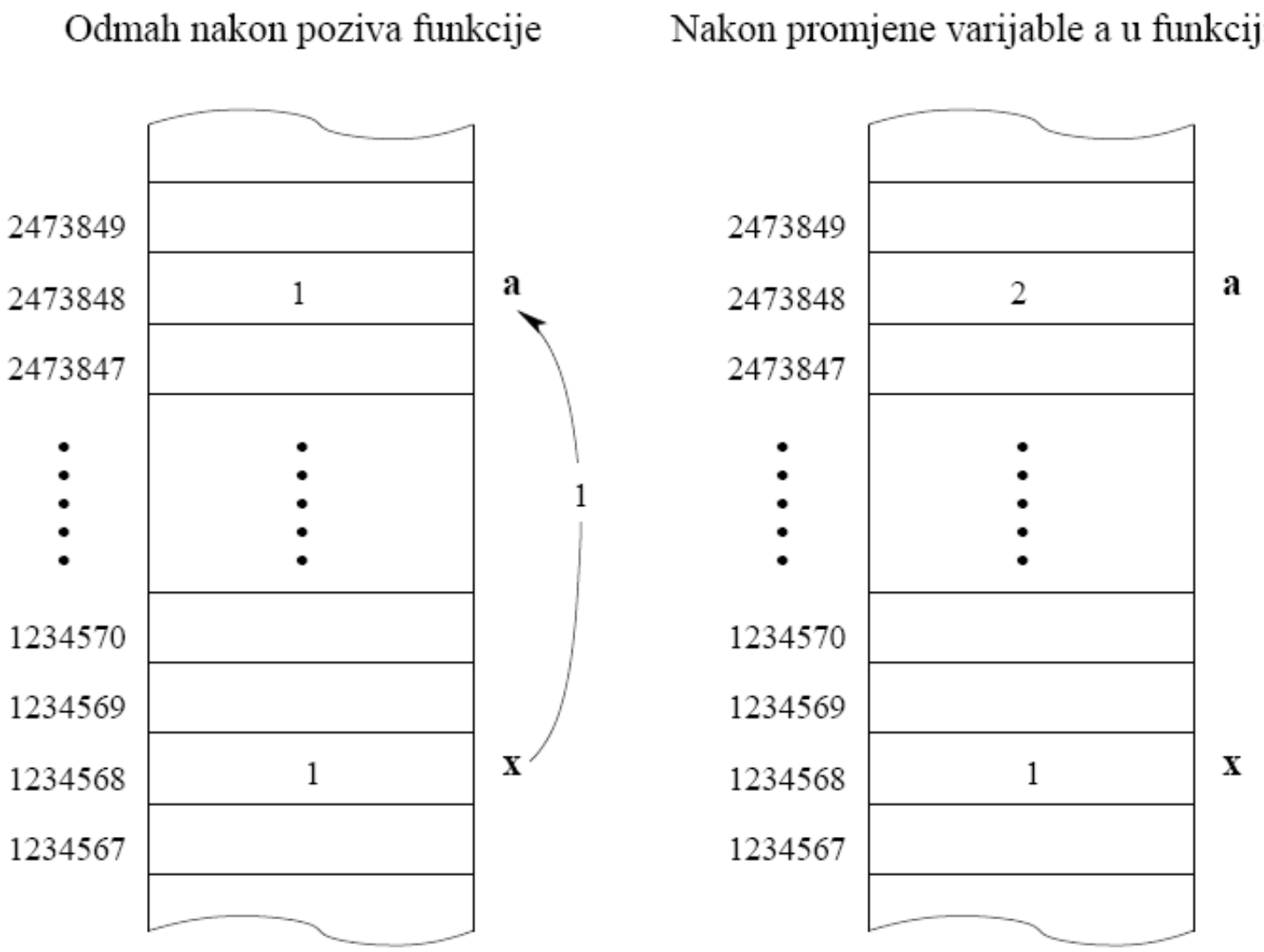
Varijabilni argumenti funkcija

- Kako mijenjati vrijednosti u funkciji? – Koristiti pokazivače
- Prenosimo **adresu** varijable, a ne njenu vrijednost.

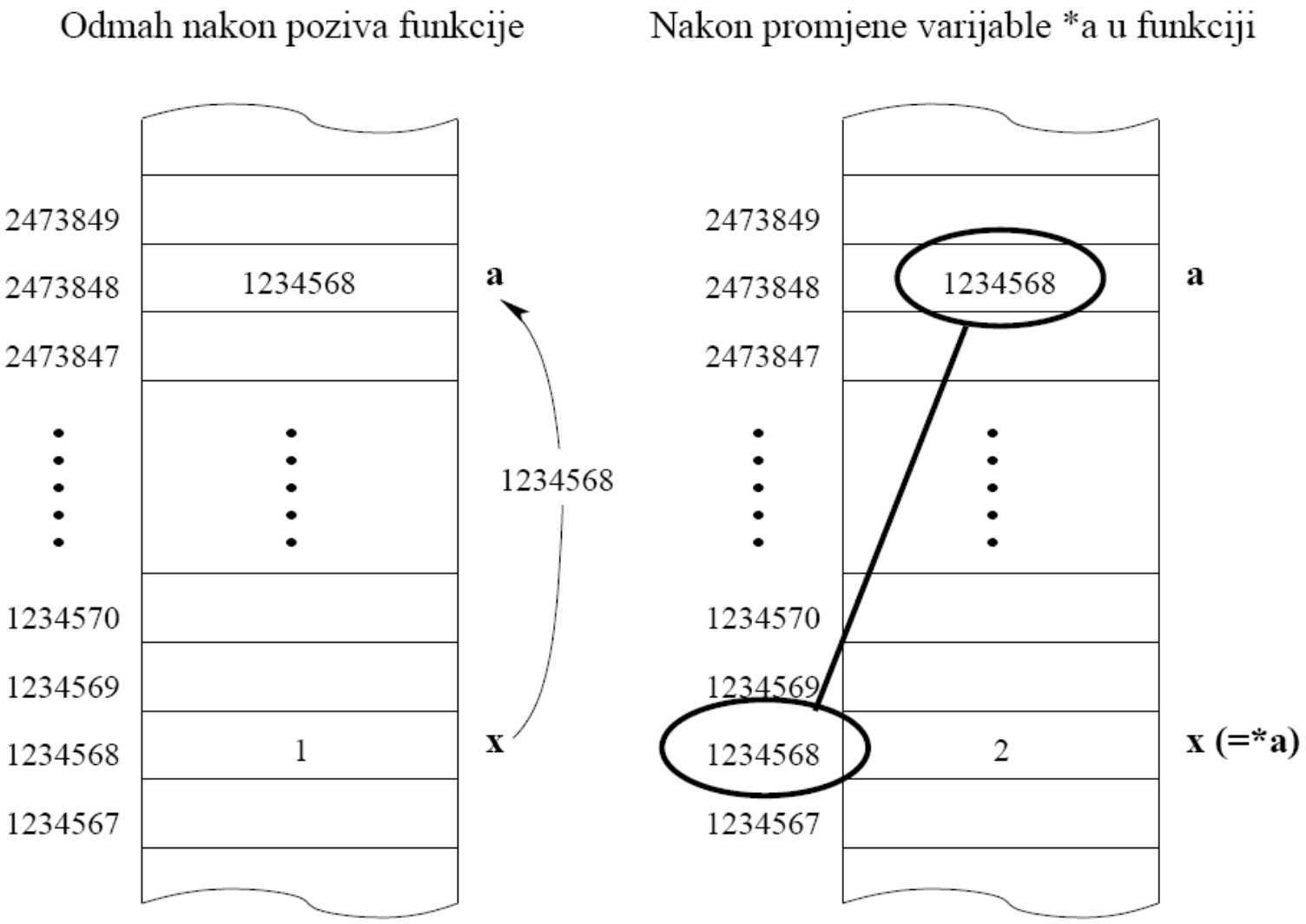
- Primjer 4.2:

```
#include <stdio.h>
void f(int *a) {
  printf("Funkcija 1: a = %d\n", *a);
  (*a)++;
  printf("Funkcija 2: a = %d\n", *a);
}

int main(void) {
  int x = 1;

  printf("Gl. prog. 1: x = %d\n", x);
  f(&x);
  printf("Gl. prog. 2: x = %d\n", x);

  return 0;
}
```

```
Gl. prog. 1: x = 1
Funkcija 1: a = 1
Funkcija 2: a = 2
Gl. prog. 2: x = 2
```



SLIKA 1. Memorija u primjeru 4.1



SLIKA 2. Memorija u primjeru 4.2

Varijabilni argumenti funkcije

- *Zadatak 4.1* : Što ispisuje sljedeći program:

```
#include <stdio.h>

void f(int *a, int b) {
  int *c;
  c = &b;
  (*a)++; b++; (*c)++;
}

int main(void) {
  int a = 1, b = 10, c = 100;

  printf("a = %d, b = %d, c = %d\n", a, b, c);
  f(&a, b);
  printf("a = %d, b = %d, c = %d\n", a, b, c);

  return 0;
}
```

a = 1, b = 10, c = 100

a = 2, b = 10, c = 100

Varijabilni argumenti funkcije

- *Zadatak 4.2* : Napišite funkciju koja omogućuje zamjenu vrijednosti dva realna broja.

```
void swap(double *x, double *y) {
  double temp;
  temp = *x;
  *x = *y;
  *y = temp;
}
```

  - Česta greška:

```
void swap(double *x, double *y) {
  double *temp;
  *temp = *x;
  *x = *y;
  *y = *temp;
}
```

Varijabilni argumenti funkcije*

- *Napomena 4.2.* To što je varijabilni parametar pokazivač i dalje ne znači da možemo njega mijenjati tako da promjena afektira parametar s kojim je funkcija pozvana. Možemo mijenjati isključivo onu varijablu na koju varijabilni parametar (pokazivač) pokazuje!
- *Zadatak 4.3.* Napišite funkciju koja preko varijabilnog parametra poništava pokazivač na cijeli broj, tj. postavlja ga na vrijednost NULL.

```
void nullify(int **x) {
  *x = NULL;
}
```

Varijacija na temu: Promijeniti x tako da pokazuje na neku drugu varijablu

```
void f(int **x, int *b) {
  *x = b; /* što bi bilo x=b, **x = *b , *x = *b ? */
}
int main(void) {
  int a = 1, b = 2, *x;
  x = &a;
  f(&x, &b);
  printf("%d", *x);  /* ispisuje 2, tj. vrijednost od b*/
  return 0;
}
```

Malo ponavljanja o rekurzijama i matricama

- Zadatak V1:Napisati program koji će učitati cijeli broj n (n<=20), a zatim i kvadratnu matricu reda n. Program treba izmijeniti matricu na način da je nova vrijednost pojedinog elementa na sporednoj dijagonali jednaka sumi elemenata koji se nalaze lijevo i ispod navedenog elementa. Nakon

$$\begin{bmatrix} 1 & 3 & 0 & 12 & 34 & 3 \\ 3 & 4 & 13 & 12 & 16 & -2 \\ 8 & -8 & 8 & 0 & 10 & 6 \\ 10 & -2 & 3 & 9 & 11 & 8 \\ -4 & 1 & 4 & 9 & -5 & 3 \\ 7 & -2 & 6 & 2 & -8 & 4 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 3 & 0 & 12 & 34 & 69 \\ 3 & 4 & 13 & 12 & 40 & -2 \\ 8 & -8 & 8 & 28 & 10 & 6 \\ 10 & -2 & 18 & 9 & 11 & 8 \\ -4 & -6 & 4 & 9 & -5 & 3 \\ 0 & -2 & 6 & 2 & -8 & 4 \end{bmatrix}$$

Zadatak V1

```
#include <stdio.h>
int main(void) {
  int i,j,k,n,a[20][20],suma;
  printf("Unesite n:");scanf("%d", &n);
  for(i=0;i<n;i++){
     for(j=0;j<n;j++){
          printf("a[%d][%d]=", i, j);
          scanf("%d", &a[i][j]);
     }
  }

   ... promjena matrice ...

  printf("\n");
  for(i=0;i<n;i++){
     for(j=0;j<n;j++)
          printf("%3d", a[i][j]);
     printf("\n");
  }
  return 0;
}
```

Zadatak V1

Varijanta 1:

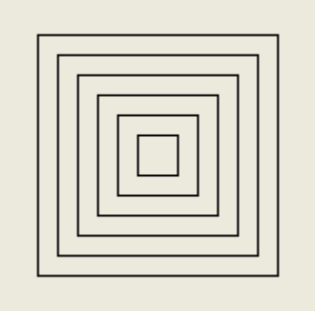
```
for(i=0; i < n ; i++){
     suma = 0;
     for(k=0; k<n-i-1 ; k++)
          suma += a[i][k];
     for(k=i+1 ; k<n ; k++)
          suma += a[k][n-i-1];
     a[i][n-i-1] = suma;
}
```

Varijanta 2:

```
for(i=0; i < n ; i++){
     suma = 0;
     for(k=0; k<n-i-1 ; k++){
          suma += a[i][k];
          suma += a[n-1-k][n-i-1];
     }
     a[i][n-i-1] = suma;
}
```

Malo ponavljanja o rekurzijama i matricama

- *Zadatak V2:(FER, ASP, 1. međuispit):* Napisati **rekurzivnu** funkciju koja za zadane x i y koordinate lijeve gornje točke kvadrata, duljinu st
  brojKvadrata crta "koncentrične kvadrate"
  kao na slici.



  Točka (0, 0) koordinatnog sustava se nalazi u gornjem
  lijevom kutu. Svaki kvadrat ima središte kao i prethodni,
  a stranicu duljine 20 piksela veću od prethodnog. Za crtanje koristite funkciju DrawRectangle koja iscrtava pravokutnik s gornjim lijevim kutom u (x, y) i duljinom stranica sirina i visina, koja ima prototip:

```
DrawRectangle(int x, int y, int sirina, int visina)
```

  Funkciju DrawRectangle ne treba implementirati!
  Na slici je zadan brojKvadrata = 6. Funkcija koju treba napisati ima prototip:

```
void KoncentricniKvadrati(int x, int y, int stranica, int
    brojKvadrata);
```

Zadatak V2

```
void KoncentricniKvadrati(int x, int y, int stranica, int
   brojKvadrata)
{

   if (brojKvadrata > 0)
   {
       DrawRectangle(x, y, stranica, stranica);
       KoncentricniKvadrati(x-10, y-10, stranica+20,
              brojKvadrata - 1);
   }
}
```

Zadatak V2 – primjer “izuzetno” krivog rješenja

```
void KoncentricniKvadrati(int x, int y, int stranica,
    int brojKvadrata)
{
   if (brojKvadrata == 0)
        break; /* ili npr. exit(0);  ili npr. return 0;*/
   DrawRectangle(int x, int y, int stranica, int stranica);
   return KoncentricniKvadrati(int(x-10), int(y+10),
        stranica+20, brojKvadrata--);
}
```

- Pronađite pogreške.

Malo ponavljanja o rekurzijama i matricama

- *Zadatak V3 :* Neka je zadana funkcija `int izracunaj(int niz[], int n)` koja će za niz od n binarnih vrijednosti izračunati "vrijednost" tog niza. Vrijednosti mogu biti i pozitivne i negativne. Potrebno je napisati funkciju `trazi` koja osim cijelog broja n može imati i neke druge argumente, a koja će vratiti najveću moguću vrijednost koju n binarnih znamenki može postići.

- Uputa:
  - Neka prototip funkcije trazi bude sljedeći:

```
int trazi(int niz[], int pos, int n)
```

    - `niz` = sadrži nule ili jedinice
    - `pos` = trenutna pozicija u nizu
    - `n` = ukupan broj znamenki

Zadatak V3

```
int trazi(int niz[], int pos, int n){
   if (pos<n){
        int max, val;
        niz[pos] = 0;
        max = trazi(niz, pos+1, n);
        niz[pos] = 1;
        val = trazi(niz, pos+1, n);
        if (val > max)
                max = val;
        return max;
   }
   else
        return izracunaj(niz, n);
}
```

gl. program:

```
int main(void) {
  int niz[10];
  printf("\nMax = %d\n", trazi(niz, 0, 7));
}
```

Zadatak V3

Što ako se ne radi samo o binarnim znamenkama nego npr. o dekadskim?

```
int trazi(int niz[], int pos, int n){
   if (pos<n){
        int i,max, val;
        niz[pos] = 0;
        max = trazi(niz, pos+1, n);
        for(i=1; i<10 ; i++){
               niz[pos] = i;
               val = trazi(niz, pos+1, n);
               if (val > max)
                      max = val;
        }
        return max;
   }
   else
        return izracunaj(niz, n);
}
```