PiPI

VJEŽBE ZA BLITZ 01

Grupa 07,

Z. Šimić, 2008.

Teme za 1. blitz

- Prikaz pozitivnih cijelih brojeva
- Raspon cijelih brojeva
- Prikaz negativnih cijelih brojeva
- Raspon brojeva različitih tipova podataka
- Prikaz realnih brojeva (komponente)
- Prikaz realnih brojeva (vrijednosti)
- Brojevni sustavi (hexa, octal itd.)
- Tipovi konstanti i imena varijabli u C-u

Prikaz pozitivnih cijelih brojeva

- Odrediti dekadsku vrijednost u registru (memoriji) za broj bez korištenja i uz korištenje dvojnog komplementa
  - Za registar od 3 bita i broj 2+2:

    bez 2k.: 2_{10} = $_2$ -> **10**

    sa 2k.: 2_{10} =

    $_2$ -> **10**
  - Za registar od 3 bita i broj $3_{10}+3_{10}$ (rješenje $_{10}$ i $_{10}$).
- Odrediti dekadsku vrijednost cijelog broja uz korištenje tehnike dvojnog komplementa
  - Primjer 01011_2 i 11011_2

    01011_2 = $_{10}$

    11011_2 =

    $_2$ -> **10**

Prikaz pozitivnih cijelih brojeva

- Nakon zadane računske operacije odrediti sadržaj registra
  - Za 3+3+1 u 3 bitnom registru
    - Ili isti broj u 2 bitnom registru

- Odrediti maksimalni cijeli broj koji se može prikazati u registru s dvostrukim komplementom i bez 2k.:
  - Za 2 bitni i 4 bitni registar

- Odrediti broj različitih vrijednosti koje se može prikazati
  - Za prethodni primjer
    za 2 bita __ vrijednosti i za 4 bita __ vrijednosti; općenito =

Prikaz pozitivnih cijelih brojeva

- Kako izgleda zbroj dva binarna broja binarno i dekadski
  - $1101_2 + 1011_2$

2 10

- Koji broj je prikazan u prethodnom primjeru ukoliko se primjeni tehnika dvojnog komplementa?

- Pretvoriti dekadski broj u binarni
  - Primjer 199_{10}

2

- Množenje razlomljenog binarnog broja
  - Primjer množenje s 2_{10}
    binarni zarez (točka) se pomiče __ x u lijevo/desno

Raspon cijelih brojeva

- Različiti pozitivni brojevi u binarnom prikazu
  - Recimo registar od 3 bita:

    -> __ različita broja

    općenito se može prikazati brojeva?
- Različiti negativni brojevi u binarnom prikazu
  - Recimo registar od 3 bita:

    -> __ različita broja

    općenito vrijedi se može prikazati brojeva?
- Raspon brojeva u binarnom prikazu
  - Recimo registar od 5 bita:

    bez predznaka s predznakom u tehnici dvojnog komplementa

    [,] [,]

Raspon cijelih brojeva

- Raspon brojeva u binarnom prikazu

  - char (veličina?) raspon:

    s predznakom u tehnici dvojnog komplementa
    [,]

    unsigned (bez predznaka)
    [,]

  - short int (veličina?) raspon:

    unsigned (bez predznaka)
    [,]

    s predznakom u tehnici dvojnog komplementa
    [,]

Prikaz negativnih brojeva

- Potreban broj bita za prikaz nekog negativnog broja
  - Broj -42_{10}:

    42_{10} =

    $_{2}$

    Potrebno je minimalno _ bitova.

- Zapis negativnog broja tehnikom dvojnog komplementa
  - Broj -3 u 16 bitnom registru:

    3_{10} =

    2

Prikaz negativnih brojeva

- Prikazati dekadski broj heksadecimalno primjenom tehnike dvojnog komplementa
  - Broj **-21_{10}** u 8 bitnom registru :

    21_{10} = 2

    16
  - Broj **-121_{10}** u 9 bitnom registru :

    121_{10} = 2

    2 = 16
- Sadržaj registra opisan hexadecimalno u dekadski
  - Kada je u 4 bitnom registru zapisano **B_{16}**

B_{16} = **2** 2 = **10**

Vrijednost u varijabli nakon pridruživanja

- char vrijednost$_{10}$ (bin. ili hex.)

```
char c = -8;
```

```
char c = 121;
c = c + 7;
```

```
char c = 125;
c = c + 9;
```

```
char c = -125;
c = c - 6;
```

- short

```
short i =32767;
i = i + 1;
```

Prikaz realnih brojeva

- Raspon binarnog eksponenta (BE)
  - [,] za jednostruku preciznost
    - K=BE+__ (__ bita za karakteristiku)
    - K=255 posebno značenje (M nije 0 -> ___, M=0 -> ____)
    - K=0 posebno značenje (M nije 0 -> __, M=0 -> _____ broj)
  - [,] za dvostruku preciznost
    - K=BE+__ (__ bita za karakteristiku)
    - K=2047 posebno značenje (M nije 0 -> ___, M=0 -> ___)
- Duljina mantise – određuje preciznost
  - __ bita za jednostruku preciznost (+ skriveni bit) - prec. __ dek. znam.
  - __ bita za dvostruku preciznost (+ skriveni bit) - prec. __ dek. znam.
- Dodavanje jako malog broja jako velikom broju
  - Na primjer 1,0e33 + 1,0e-33
    Ne mijenja veliki broj jer _____!

Prikaz realnih brojeva

- Binarni rezultat množenja binarnog broja
  - $101101 \times 2^{-5} =$
  - $101.101 \times 2^{2} =$
- Karakteristika za realni broj u jednostrukoj preciznosti
  - Realni broj 33,257

    Dovoljno je gledati samo cijelobrojni dio: $33_d = 1_b$

    $\Rightarrow$ treba množiti sa __ da se dobije = => BE=

    $\Rightarrow$ K= $_d$ => $_b$

Broj bita mantise i dekadskih znamenaka

- Broj potrebnih bita **b** za dekadski broj sa **d** znamenaka
  -
  - Za dekadski broj sa 6 znamenaka:
    b = bita -> bita mantisa
- Broj dekadskih znamenaka **d** uz zadani broj bita **b**
  -
  - Za mantisu sa 23 bita:
    d = decimalnih znamenaka
-
  -

Određivanje mantise za IEEE 754 realni broj

Prikaz u binarnom i heksadecimalnom obliku?

Primjer -118.625_{10}

Binarno: $_2=$ $\cdot 2^-$

Binarno u standardnoj preciznosti:

BE= -> K=

```
1      8             23           broj bita
+-+--------+-----------------------+
|S| Karakt.| Mantisa               |
| |        |                       |
+-+--------+-----------------------+
31 30          23 22          redni broj bita (0 je desno)     0
```

BE +127

16

Primjer 5.25_{10}

Binarno: $_2 =$ $\cdot 2^-$

Binarno u standardnoj preciznosti:

BE= -> K=

2

16

Primjer: Prikazati -9.125 u binarnoj jednostrukoj preciznosti te hexadecimalno i oktalno

Još neka pitanja:

1. Koji se najmanji broj po apsolutnoj vrijednosti može pohraniti u registru korištenjem tehnike dvojnog komplementa?
2. Kolika se vrijednost dobije nakon dodavanja broja 2 najvećem pozitivnom broju koji se može prikazati u jednostrukoj preciznosti?
3. Koliko se puta poveća najveća vrijednost pozitivnog broja u registru sa **b** bita kada se broj bita poveća na **2*b**?
4. Koliko treba bita za prikaz pozitivnih brojeva do **6** znamenki u registru bez dvojnog komplementa?
5. Koliko različitih cijelih brojeva se može prikazati korištenjem dvojnog komplementa u registru od **11** bita?

Tipovi konstanti i imena varijabli u C-u

- Koliko mjesta će u memoriji zauzeti konstanta `4.0f`

  `2, 4, 8, 10` ili nedefinirano okteta

- Identificirajte element izraza: `x=y+`**`40.`**

  pogrešno, konstanta, `float` konstanta, `double` varijabla ili `double` konstanta

- Konstanta 44U prikazana hexadecimalno:

  `1B, FF2C, FF44, 2C ili 44`

Tipovi konstanti i imena varijabli u C-u

- Za izraz `a = 6U` što označava slovo U?
- Binarni prikaz konstante u C-u `0x66`?
- Odrediti **neispravno** ime za varijablu:
  `produkt, int, Z9, _77`
- Odrediti **ispravno** ime za varijablu:
  `void, 6puta, a_9, x+y, suma!0`

Tipovi konstanti i imena varijabli u C-u

- Odrediti `tip` konstante i potreban broj okteta u memoriji:

`3ed-2`	`3`	`21f`	`4u`
`3e+2`	`9.1u`	`2.1E10u`	`0x77L`
`3e-2`	`34.1f`	`21.`	`3.24`
`07.7f`	`0x1A.1f`	`0xff`	`0101u`

Tipovi konstanti i imena varijabli u C-u

- Koji znak predstavlja `'\x45'`?
  - ASCII tablica predstavlja slovo 'A' brojem __

    `E, F, 45, f,` ništa navedeno
- Konstanta `0214` predstavljena binarno

  `10001001, 11001001, 10001100, 01001100, 11101100`
- Ispravan zapis `signed long` hexadecimalne konstante

  `2.4l, -987B1F, 1234ful, 0x101l, 0x14.fel`

PiPI

VJEŽBE ZA BLITZ 02

Grupa 07,

Z. Šimić, 2008.

Teme za 2. blitz

- Apsolutne, relativne i numeričke pogreške
- Raspon realnih brojeva
- Cjelobrojne i realne konstante i varijable u C-u
- ASCII tablica i operacije sa znakovnim tipom podatka
- Redoslijed obavljanja aritmetičkih operacija i konverzija tipova podataka
- Konstantni niz znakova
- Logički tip podataka, relacijski operatori, logički izrazi i naredbe
- Logički izrazi s logičkim operatorima
- Jednostrana selekcija
- Dvostrana selekcija
- Jednostrana selekcija sa složenijim uvjetom
- Dvostrana selekcija sa složenijim uvjetima

Raspon realnih brojeva

- Ovisi o broju bita karakteristike
- Jednostruka preciznost
  - najveći broj ili
    najmanji broj ili
- Dvostruka preciznost
  - najveći broj ili
  - najmanji broj ili

Prikaz realnog broja u IEEE 754

- Odrediti sadržaj registra u heksadekadskom obliku koji za prikaz broja -1.25_{10} prema IEEE 754 standardu u **dvostrukoj preciznosti.**

- Za registar koji sadrži broj $C029000000000000_{16}$ treba odrediti realnu dekadsku vrijednost podatka tipa double.

- Napisati u heksadekadskom prikazu sadržaj registra koji prikazuje: 0, $+\infty$ i NaN

Preciznost realnog broja u IEEE 754

- Realni broj se pohranjuje u memoriju u jednostrukoj i dvostrukoj preciznosti te u prikaz koji ima 15 bita za karakteristiku i 65 bita za mantisu (uključujući skriveni bit). Treba prvo odrediti najveću moguću relativnu pogrešku, a potom najveću apsolutnu pogrešku za broj 1000.1 za sva tri slučaja.

Znakovni tip, ASCII tablica, nizovi znakova

```c
#include <stdio.h>
int main( ) {
char z='c';
printf(" = %c ", z/2);
printf(" = %d ", z/2);
printf("z= %c ", z-1);
printf(" = %c ", z-32);
printf(" = %d ", z - 'a');
z = z + 1;
printf("z= %c i %d", z, z);
return 0;
}
```

```
Ispis:

  =
  =
z=
  =
  =

z=
```

Znakovni tip, ASCII tablica, nizovi znakova

```
char z = '\x41';
printf("z= %c\n", z);
z = '1';
printf("z= %c\n", z+1);
z = (z+1)/'1';
printf("z= %d\n", z);
z = '1'+'1';
printf("z= %c\n", z);
```

```
Ispis:
 z= 
 z= 
 z= 
 z= 
```

Aritmetički operatori i izrazi

- Koliko iznose varijable nakon izvršavanja?

```
int main() {
        int a=1, b=1;
        b = (a = a - b) + b;
        a = a + b;
}

/* rezultat:  a =     b =   */
```

Aritmetički operatori i izrazi

- Koliko iznosi varijabla **x** nakon izvršavanja?

```
int  y, z;
float x;
y=4; z=14;
x = z / y * y  + z % y;
printf(“x = %f ", x);
```

x =

Aritmetički operatori i izrazi

- Što ispisuje program?

```
int x=5, y;
y = 2*x + x * 5 % x;
printf(“x=%d y=%d", x, y);
```

x= y=

Aritmetički operatori i izrazi

- Kolika je vrijednost varijabli nakon izvođenja?

```
int x, y;
float w;
x = 5;
y = x;
w = x / 4 * y - x * 5 * y;

/* rezultat:  x=  y=  w=                */
```

Aritmetički operatori i izrazi

- Kolika je vrijednost varijabli nakon izvođenja?

```
int  a = 9, b = 10;

a = (b = b - a) + a;
b = b + a;



/* rezultat  a =     b =    */
```

Aritmetički operatori i izrazi

- Odrediti vrijedosti za varijable d i j:

```
int b = 91, d, j;

d = b/10;  /* b=  */

j = b%10;  /* j= */
```

Aritmetički operatori i izrazi

- Dodavanje cijelom broju realnog broja:

```
int i;
float k;
k = 6.;
i = k + 0.5;

/* rezultat:   i =      */
```

- Dijeljenje dva cijela broja:

```
int i = 8;
float r = 2*i/3%2;

/* rezultat u   r =        */
```

Vrijednost izraza

```
int x=1, y=-1, z=0;
                  z =!z && !y || x;
            z =(x >= y) && (x != y);
               z = x >= y && x != y;
                        x = y+x>-1;
                         x = y+x>1;
        z=0, x=0; x = !x || y && z;
                  x = (!x || y) && z;
                   z=5,   z = !z - 1;
```

Rezultat:

- z =
- z –
- z =
- x =
- x =
- x =
- x =
- z =

Vrijednost izraza

```
int x=10, y=20, z=0;
```

Rezultat:

```
z = x>y || x==10 && y==2;
```

➢ z =

```
z = x >= x * x - 90;
```

➢ z =

```
z = y <= y / y + 10;
```

➢ z =

Aritmetički operatori i izrazi

- Koliko iznosi varijabla **a** nakon izvršavanja?

```
int main() {
        int a=1, b=0;
        if (b=!a) a=!b;
        a = 1 + a&&1;
}

/* rezultat:  a =
                                */
```

Kontrolna naredba selekcije

```
int x = 10, y = 0;
if (!x - y) x = y+1;
else y = x-1;
```

Rezultat u varijablama:

`x =` `y =`

```
x = y = 0;
if (x - !y) x = y+1;
else y = x-1;
```

`x =` `y =`

Kontrolna naredba selekcije

```
int x = 1, y = 0;
if (x == !y)
  x = x+y+1;
else
  x = x+y-1;
```

Rezultat u varijablama:

x = y =

```
x = 2, y = 1;
if (x == !y)
  x = x+y+1;
else
  x = x+y-1;
```

x = y =

Kontrolna naredba selekcije

```
int x=1;
if (x%2) x=11;
else if (x) x=22;
```

Rezultat u varijabli:

x =

```
x = 2;
if (x%2) x=11;
else if (x) x=22;
```

x =

Kontrolna naredba selekcije

```
int x=-1, y=-1, z=1;
if(x - z == 0) x = x - 1;
  else z = x && 0;
       y = y + z + 1;
```

Rezultat u varijablama:

x= y= z=

```
x = z = -1;
y = 0;
if(x - z == 0) x = x - 1;
  else z = x && 0;
       y = y + z + 1;
```

x= y= z=

Kontrolna naredba selekcije

```
int x=-1, y=1;
if(x && y) x=x+1;
else y=y+1;
```

Rezultat u varijablama:

x= y=

```
x=-1, y=1
if(x > 0 && y > 0) x=x+1;
  else y=y+1;
```

x= y=

Kontrolna naredba selekcije

```
int x=1, y=1, z=1;
char c;
 if(x < y){
   if(y < z)
     c = ‘A’;
   else
     c = ‘B’;
  } else {
    if(y > z)
      c = ‘C’;
    else
      c = ‘D’;
  }
// Rezultat:  c =
```

```
int x = -1, y = 1;
if(x+1 > y-1) x=x+1;
else y=y+1;

// Rezultat: x=   y=
```

```
x = -1, y = 1;
if(x = y-1) x=x+1;
else y=y+1;

// Rezultat:  x=   y=
```

Istinitost izraza

```
int x=0, y=1;
                    if(x==y)   ➢
                     if(x=y)   ➢
              x=0; if(!(y=x))  ➢
              y=1; if(!(y==x)  ➢
                       if(x+1)  ➢
                  x=0; if(1+x)  ➢
                       if(y-1)  ➢
                  y=1; if(1-y)  ➢
```

Zamke selekcije

Sadržaj varijabli nakon izvođenja, slijedom:

```
int x=1, y=0, i=1;

if (x=y+1) x=1; y=5;
if (!y) x=5; y=0;

if (i%2) i=11;
else if (i) i=22;

if (x=i<y) x=10;
```

Vrijednost:

```
x=   y=

i=

x=
```

Zamke selekcije

```
int x, y;
x = -1; y = 6;
if (x = y)
   printf (“>%d%d", x, y);
else
   y = -1;
   printf ("%d%d<", x, y);
```

Ispis:

Rezultat:

```
int x=1, y=-1, z=0;
if (!(x + y>z))
    x = x – y + 2;
    y = y - x - 1;
    z = z * (x – y);
```

```
x=   y=    z=
```

Još neka pitanja:

1. Koliko se mjesta u memoriji zauzme kada se u C-u želi prikazati logički tip podataka (*boolean*)?
2. Koliko puta je veća najveća moguća apsolutna pogreška realnog broja jednostruke preciznosti kada bi imao mantisu od 16 bita (uključujući skriveni bit)?
3. Koliko puta je manja najveća moguća relativna pogreška realnog broja dvostruke preciznosti od realnog broja jednostruke preciznosti?
4. Koliko se smanji najveća moguća relativna pogreška u IEEE 754 formatu jednostruke preciznosti, ako se za karakteristiku koristi 10 bitova uz nepromijenjenu mantisu?

PiPI

VJEŽBE ZA BLITZ 03

Grupa 07,

Z. Šimić, 2008.

Teme za 3. *blitz*

- Višestruko pridruživanje i skraćeno pridruživanje
- Odvajanje naredbi, uvjetno pridruživanje
- Programske petlje:
  - `while` (kratki i dugi oblik naredbe)
  - `do-while` (kratki i dugi oblik naredbe)
  - `for`
  - kombinacije više tipova petlji
  - u kombinaciji s `break`, `continue`
  - kombinacije selekcija i programskih petlji
  - degenerirane (beskonačne, one koje se ne izvršavaju)
- Korištenje bitovnih operatora
- Korištenje `goto`
- Naredba `switch`

Operatori – rezultati slijedom

Ispis:

```
char z='5';
printf("%d", z & 0x0f);
printf("%d", z^z+2);
printf("%d", ~(z-53));
int x=1, y=2;
x *= y/2 + 4;
y /= x/3 + 1;
printf("x=%d y=%d",x,y);
printf("x=%d", x%=(y+=2));
printf("y=%d", y);
printf("%d",z>>=y/x);
```

Uvjetni operator

- Rješenje:

1. a, b, c?

```
int a=0, b=-2, c=-1;
 a= (c && a) ? c += a : b == a;
```

2. a, c?

```
int a=1, b=-1, c=1;
 a= (b<c)<<1 ? c+1 : c-1;
```

3. z?

```
char z, c='5';
 z= c<48 || c>57 ? 'Z' : 'B';
```

Uvjetni operator

4. Ispis?

```
int n=1, uvjet=0;
printf("%d. odgovor je %s!\n", n,
   uvjet ? “DA” : “NE”);
```

5. Ispis?

```
int a=5, b=-1, c=1;
c = (a=c&&b) ? a = b : ++c;
printf("a=%d b=%d c=%d",a ,b , c);
```

6. x, y?

```
int x=-1, y=1;
x= ++x >= y-- ? x && y++ : x-(++y);
```

- Rješenje:

Programska petlja s ispitivanjem uvjeta na početku – napisati kod za izraz:

$$\prod_{n=1}^{5} \frac{1}{(n+3)}$$

$$\sum_{n=1}^{5} \frac{1}{(n+3)}$$

$$\sum_{n=1}^{5} \frac{1}{n \cdot 3}$$

Programska petlja s ispitivanjem uvjeta na početku

Koliko puta se obavlja petlja?

```
char c = 1;
c = 1;
while(c>0) c=c+1;
```

/* Petlja se obavlja ___ puta

Sličan problem sa **`short int`** varijablom! */

Koliko puta se obavlja petlja?

```
int n = 33;
  while (n > 9)
    n -= 3;
```

/* Petlja se ponavlja __ puta */

Programska petlja s ispitivanjem uvjeta na početku

Što je rezultat?

```
int n=0, m=10;
while( !(n==3) ) {
   m = m - n;
   n++;
}
```

// Na kraju: n= m=

Što je rezultat?

```
int n, m;
n=m=10;
while (m>0 && n/m) m--;
```

// Na kraju: n= m=

Do while petlja

```
int x=1, y=1;
do {
 ++x;
 y *= x / 2;
} while (y%x);
```

Prolaz x y y%x

Rezultat: x= y=

```
int x = -1;
float r = 2.;
do {
   x++;
  r += x;
  printf("%f %d ", r, x);
} while (r<4 && x);
printf("DNO");
```

Do while petlja

```
char z='1';
do {
 z+=2;
 printf("%c%d ", z, z);
} while ('5'-z);
```

Ispis:

```
int x=9;
do {
 x-=2;
} while (++x>5);
```

Rezultat:
Nakon prolaza x =

```
int a=10, b=10, c=10;
do {
   a = --b;
    do {
      b = c--;
   }while(b>9);
}while(!(a<10));
```

Prolaz a b c

Kraj

v. i u. – vanjska i unutrašnja petlja

For petlja

```
int i, j, x=10;
for (i=0; i<2; i++)
   for (j=1; j<3; ++j)
       x -= i + j;
```

Prolaz i j x

Na kraju: i= j= x=

v. i u. – vanjska i unutrašnja petlja

```
int j, ukupno=0;
for (j=1; ukupno<10; j+=3)
   ukupno += j;
   ukupno /= j;
```

Rezultat: j= ukupno=

```
int j, ukupno;
for (j=15, ukupno=0; j>10; j--)
   if (ukupno%j) ukupno += j;
```

Rezultat: j= ukupno=

For petlja

```
int i, j, x=0;
for (i=10; i>6; i-=2)
   for (j=8; j>6; --j)
       x += i - j;
```

Prolaz i j x

Na kraju: i= j= x=

v. i u. – vanjska i unutrašnja petlja

```
int i, j;
for (i=1,j=0; i<4||j<9; i++,j+=3)
    printf("%d %d ", i, j);
```

Ispis:
Na kraju: i= j=

```
int i, x;
for (i=0, x=0; i<10; x+=i, i++){
   printf ("%d %d ", i, x);
   if (i&&x&&i%3!=1) break;
}
```

Ispis:
Na kraju: i= x=

Naredba **switch**

```
int x=0;
switch(x=1){
 case 0: x*=100;
 case 1: x+=10;
 case 2: x+=10;
 default: x/=10;
}

Rezultat:   x=
```

```
switch('c'-'a'){
 case 1: printf("b");
 case 2: printf("c");
 case 3: printf("d");
 default: printf("*");
}

Ispis:
```

Naredba **switch**

```
int x=1, y=5;
++x;
switch(y%x){
 case 0: y+=x;
 case 1: y-=x;
 case 2: y--; break;
 default: y/=5;
}
```

Rezultat: x= y=

```
char z='C';
switch(z-'A'){
 case 1:
   printf("%d ", z++);
 case 2:
   printf("%c=%d",z,z);
 case 3:
   printf(",z=%d", z+=2);
   break;
 default: printf("*");
}
```

Ispis:

Petlje za kraj

```
i=0;
while(++i){
 if (i%3==0) continue;
 if (!(i%10)) break;
 printf("%d ",i);
}
```

Za ___. prolaza izvršava se **break**, a ___ puta se izvršava **continue**.

Ispis:

Na kraju: i=

```
for(i=j=1; i!=0; i++){
 if (i%2) continue;
 if (i+j>10) break;
 j+=2;
 printf("%d %d ",i, j);
}
```

Za ___. prolaza izvršava se **break**, a ___ puta se izvršava **continue**.

Ispis:

Na kraju: i= j=

Rezultat izvršavanja koda

Vrijednosti varijabli na kraju?

```
int a, b=0, c=2, d=4;
a = ++b > c ? d : --c;
if (d = 4) a += b;
if (d == 4) a -= b;
printf("a=%d b=%d c=%d
 d=%d\n",a,b,c,d);

/* a=   b=   c=   d=   */
```

Što se i koliko ispisuje?

```
int i = 1;
 for(;;i++) {
  if(i % 5 == 0) {
    printf("%d, ",i);
    continue;
  }
  if(i % 25 == 0) {
    i=0;
    break;
  }
 }

/*
```

goto

```
int x=1, y=1;
do {
  if (x>3) goto dosta;
  ++x;
  y *= x + 2;
} while (y%x);
dosta:
```

Prolaz x y y%x

Rezultat:

```
int x = 1, y = -2;
do {
   if (x==y) goto van;
   x++;
  y += x;
  printf("%d %d ", x, y);
} while (x<4 || y);
van:
```

Ispis:

goto

```
char z='1';
do {
  if (!('5'-z)) goto dno;
  z+=2;
  printf("%c %d ", z, z);
} while (1);
dno:
```

Ispis:

```
int x=9;
gore:
do {
  if (x/3) goto gore;
  x-=2;
} while (x>5);
```

Koliko se puta izvršava petlja?

```
int a=10, b=5, c=5;
prvo:
if (a-b-c) goto zadnje;
b -= c-a;
if (a-b) goto drugo;
c += a+b;
drugo:
if (b) goto prvo;
a -= b+c;
goto prvo;
zadnje:
```

Nakon svega:

a= b= c=

goto

```
i=0;
while(++i){
  if (i%3==0) goto dalje;
  if (!(i%10)) goto stani;
  printf("%d ",i);
  dalje:
}
```
stani:

Za prolaza izvršava se **goto stani**, a puta se izvršava **goto dalje**.

Ispis:

Na kraju: i=

```
for(i=j=1; i!=0; i++){
  if (i%2) goto ponovi;
  if (i+j>10) goto prekini;
  j+=2;
  printf("%d %d ", i, j);
  ponovi:
}
prekini:
```

Za prolaza izvršava se **break**, a puta se izvršava **continue**.

Ispis:

Na kraju: i= j=

Na kraju

Vrijednosti varijabli?

```
int a=0,b=0;
if(a=b) a-=1; b+=1;
if(!b) b-=1; a+=1;

/* a=   b=  */
```

Vrijednosti varijabli?

```
int a=1, b=1, c=1;
while (a){
  for(; b<11; b++){
    if (b%2)
      break;
    else
      continue;
    a--;
    c+=10;
  }
  c++;
  if (c>10) break;
}

/* a=   b=   c=   */
```

PiPI

VJEŽBE ZA BLITZ 04

Grupa 07,

Z. Šimić, 2008.

Teme za 4. *blitz*

- Jednodimenzionalnih polja
  - Deklaracija, dodjeljivanje početnih vrijednosti (bez znakovnih polja)
  - Korištenje (pristupanje članovima polja, indeksni izrazi)
  - Znakovna polja, dodjeljivanje početnih vrijednosti nizu znakova
  - Algoritmi
    - s numeričkim poljima
    - sa znakovnim poljima
    - složeniji
- Deklaracija dvodimenzionalnih i višedimenzionalnih polja i dodjeljivanje početnih vrijednosti dvodimenzionalnim poljima
- Jednostavniji algoritmi s dvodimenzionalnim poljima
- Zauzeće memorije varijablama i poljima (sizeof, ručno brojanje, procjena)
- Učitavanje polja i ispis polja (jednostavni formati kao npr. %5d, %15.7f, %s, %c)
- Pokazivači
  - Definiranje, tipovi i inicijalizacija
  - Korištenje (pristup podatku, izmjena podatka, aritmetika)

Polja – kratko ponavljanje

- Broj elemenata u polju nije ograničen
- Indeks prvog ili početnog elementa polja je uvijek **0**
- Upisivanje i čitanje elementa polja izvan polja je moguće, ali krivo i nepredvidivih posljedica:
  - takvo upisivanje može ‘srušiti’ program
  - takvo čitanje daje vrijednost ovisnu o stanju memorije
- Indeks zadnjeg elementa polja za jedan je manji od broja elemenata polja
- Polje može sadržavati elemente bilo kojeg tipa podataka (char, int, float), ali samo jednog tipa!
- Polje je moguće inicijalizirati kod definiranja, kompletno ili dijelom (kod djelomične inicijalizacije svim ne inicijaliziranim elementima polja pridružuje se **0**)

Definiranje, inicijaliziranje i indeksiranje polja

Ispis/posljedica:

```
int a[5] = {4}, i;
printf("%d\n", a[4]);
printf("%d\n", a[5]);
a[5]=1; a[-1]=1;
for (i=0; i<5; i++)
   printf("%d ", a[i]);
for (i=0; i<5; i++) {
   a[i]=(4-i)*10;
   printf(" %d", a[i]);
}
for (i=0; i<5; i++) {
   a[i]/=10;
   printf(" %d %d",a[i],a[i/2]);
}
```

Definiranje, inicijaliziranje i indeksiranje polja

Ispis:

```
char z[] = {'0', 'a'};
int c[2][3]={0};
printf("%c%c%d\n", z[0], z[1], c[1][2]);
```

```
int p[3][2]={9, 8, 7, 6, 5};
printf("%d %d\n", p[0][1], p[2][1]);
```

```
int m[4][3]={8,9,0,9,8,7,6,5,4,3,2,1};
printf("%d %d\n", m[3][1], m[1][2]);
```

```
int n[4][3]={{0}, {2}, {6,5}};
printf("%d %d\n", n[0][2], n[2][1]);
```

```
int a[3][4]={9,8,7,6,5,4,3,2,1};
printf("%d %d\n", m[0][3], m[2][0]);
```

Definiranje, inicijaliziranje i indeksiranje polja

Ispravno ili ne:

```
char z[4] = {‘a’, ‘b’, 99, 100};
unsigned int i[4][0][1][2][3];
```

```
int i[6]={1, 2, 3, 4, 5, 6};
char c[] = {};
```

```
float f[4]={5, 3., 1.};
int i_1[1][2]={1, 2, 3, 4};
```

```
float _f[4]={2.f, 4.f, 6.f, 8.f};
int [2,2] = {1};
```

```
int i=0, e;
float a[4]={0}, b[4], c[e]={i};
for(i=1;i<=4;i++) b[i-1]=0;
```

Koliko polje zauzima memorije?

Odgovor:

```
char z[7] = {‘a’, ‘b’};
long int i[10];
int i[]={1, 2, 3, 4, 5};
char c[5+1];
float f[50][2];
long a[10][5][2];
double d[20][50];
short s[10][10] = {1, 2};
```

1D polja u programu

Dio programa:

```
int a[]={1, 2, 3, 4, 5}, i=3;
do {
   a[i] = a[i-1]-a[i+1];
   printf(“ %d”, a[i]);
} while(--i>0);
/* program koji slijedi nastavak je programa iznad! */
for (i=4; i>=1; i--)
   printf("%d“, a[i]=i*i);
printf("a[%d]=%d", i, a[i]);

for (i=4; i>2&&i!=1; i/=2)
   scanf("%d“, &a[i]);
```

Ispis/učitavanje:

Učitava

2D polja u programu

Dio programa:

```
int a[2][10], x=0, y=0, i;
for (i=9; i!=0; i--) {
   a[0][i] = i;
   a[1][i] = i*2;
   y += a[0][i];
   x += a[1][i] - a[0][i];
}
```

Vrijednost/ispis:

Nakon petlje:

y==

x==

```
int n[4][4]={{1},{1,2},{3,4,5}};
int b=0, c=0, i, j;
for (i=0; i<4; i++){
    b += n[i][i];
    for (j=0; j<4; ++j)
        c += n[i][j];
}
```

Nakon petlji:

b ==

c ==

2D polja u programu

Dio programa:

```
int m[9][9], z=0, i;
for (i=0; i<5; i++)
   m[i][8-i] = m[8-i][i] = 1;
for (i=0; i<9; i++) z+=m[i][8-i];
```

Rezultat:

Nakon petlje:
Postavlja elemente …

z ==

```
int n[6][6], s=0, i, j;
for (i=0; i<6; i++)
    for (j=5-i; j<6; ++j){
        n[i][j]= 1;
        s += n[i][j];
}
```

Nakon petlji:
Postavlja elemente …

s ==

Pokazivači

Pokazivači

- varijable koje sadrže memorijsku adresu
- pored adrese deklaracijom je određen tip podatka na koji pokazivač pokazuje
- operator * omogućava čitanje i spremanje vrijednosti na pokazivanu adresu preko pokazivača
- operator & omogućava pridruživanje adrese bilo koje (uključujući pokazivače) varijable pokazivaču
- svaki pokazivač zauzima 4 bajta
- aritmetika nad pokazivačima mijenja vrijednost (pokazivača) u kvantima koje određuje tip podatka na koji pokazivač pokazuje

```
char *pc, c='Y';
int *pi, *pk, i=1, j;
double *pd, d=0;
pc = &c;
pi = &i;
pk = pi;
pd = &d;
*pc = 'N';
j = *pi;
*pk = 0;
*pd = 2;
```

Stanje na kraju:

c== i== j== d==

Aritmetika pokazivača

Dozvoljeno je i smisleno

- Zbrajati i oduzimati cijeli broj od pokazivača
- Za dva pokazivača na isto polje:
  - oduzimati ih
  - uspoređivati ih (<=>)

Nema smisla pridruživati pokazivaču rezultat:

- množenja, dijeljenja ili % (mod) operacija dvaju cijelih brojeva

Nije dozvoljeno

- Zbrajanje dvaju pokazivača
- Množenje, dijeljenje i % (mod) među pokazivačima

Pokazivači

```
short n=1, m=2, k;
short *pi = &n, *pj = &m, *pp;
pp = &k;
*pp = *pi;
*pi = m;
m = *pp;
*pp = k * m + n;
pp = pi;
printf("%d %d %d\n", n, m, k);
printf("%d %d %d",*pi,*pj,*pp);
printf("\n%d ", *pi + *pj);
printf("\n%d ", sizeof(n));
printf("\n%d \n", sizeof(pi));
n = sizeof(n) + sizeof(pj);
printf("\n%d %d", n, n + *pi);
```

Ispis:

Pokazivači

```
double x=3., y=4., z;
double *pa=&x, *pb=&y, *pd;
pd = &z;
*pa += *pb;
*pb = *pa - y;
x = *pa - *pb;
*pd = x * y ;
pd = pb;
printf("%f %f %f\n", x, y, z);
printf("%f %f %f",*pa,*pb,*pd);
printf("\n%f ", *pb + *pd);
printf("\n%d ", sizeof(x));
printf("\n%d \n", sizeof(pa));
z = sizeof(x) + sizeof(pb);
printf("\n%f %f", z, z + *pd);
```

Ispis:

Pokazivači – radi, ali nema smisla:

```
int n;
int *pi = &n, *pj;
double x = 5.;
double *pa = &x, *pb;

pj = (int *)x;
printf("%lu %p\n", pj, pj);

*pa = 100 * (int)(&x);
printf("%p %p\n", pa, &x);
printf("%lu %lu\n", pa, &x);
printf("%f %f\n", *pa, x);
printf("%lu", 10 * (int)(&x));
```

Ispis:

Neka je početna adresa varijable **x**
4377768 (0042CCA8$_{16}$)

Adresa varijable x je proizvoljna i rezultat se može mijenjati kod ponovnog izvođenja!

%p – format za heksadski ispis adrese
%lu – format `unsigned long` za cjelobrojni ispis bez predznaka

Pokazivači – aritmetika

```
float ar[3]={1.1, 2.2, 3.3};
float *pa, *pb;
char az[40], *pz=&az[0];
pa = pb = ar;
printf("\n%lu %f\n", pa, *pa);
pb += 2;
printf("%lu %f\n", pb, *pb);
printf("%lu %f\n", ar+2, *(ar+2));
printf("%lu %f\n", &ar[2], ar[2]);
printf("%d\n", pb-pa);
printf("%d\n", (int)pb-(int)pa);
printf("%lu %f\n", pb-1, *(pb-1));
pz += 4;
printf("%d\n", pz-az);
printf("%d\n", (int)pz-(int)az);
pz = (char *) ((int *) az +4);
printf("%d\n", pz-az);
printf("%d\n", (int)pz-(int)az);
```

Neka je početna adresa polja **ar** **1000016**!

Ispis:

PiPI

VJEŽBE ZA BLITZ 05

Grupa 07,

Z. Šimić, 2008.

Teme za 5. blitz

- Definicija funkcije
- Naredba `return`
- `void` funkcije i funkcije bez argumenata
- Prijenos kopija vrijednosti (bez polja)
- Prijenos referencija-adresa (bez polja)
- Formalni i stvarni argumenti (izrazi, redoslijed, tipovi pri pozivu funkcije) Ne spominjati stog!
- Prototipovi funkcija, organizacija složenijih programa
- Smještajni razredi (postojanost, područje važenja varijabli)
  Samo elementarni pojmovi, po mogućnosti bez register i external!
- Jednodimenzionalna polja kao argumenti funkcije
  - rad s indeksnim izrazima
  - rad s pokazivačima
- Dvodimenzionalna polja kao argumenti funkcije
  - rad s indeksnim izrazima
  - rad s pokazivačima

void funkcija

Dio programa:

```
void fn2(int n){
    printf(“fn2- %d ”, 2*n);
}
void fn1(int n){
    printf(“fn1- %d ”, 2*n);
    fn2(--n);
}
void main(){
  fn1(2);
}
```

Rezultat/ispis:

```
void uradi(int m) {
   int  i, zbroj=0;
   for(i=0; i<m; i++)
       if(i%3) zbroj+=i;
  printf("%d ", zbroj);
 }
```

Formalni i stvarni argumenti funkcija

Definicije (tri reda na početku) vrijede za sve pozive poslije

Prototip funkcije:

```
int n, m;
char z;
double r, u;
```

```
m=uf(z,n,3*r,2e-4);
```

```
uh(m,u,z-32);
```

```
oj(10.f, 3., 45);
```

```
u = ah(n,&r,&z);
```

```
printf("\n%d", eh(u,z,m));
```

Što nam znači prototip funkcije?

```
void fu(char c, int *n);
```

-može vratiti

```
int *bu(char *z);
```

-funkcija vraća _____ i ima _____ argument

```
mu(int *n);
```

- funkcija vraća _____ i prima _____ argument

Kako glasi prototip funkcije `su` koja treba izračunati sumu cijelih brojeva u nekom intervalu?

```
su(          );
su(              );
```
ili

Call by value

Program:

Rezultat/ispis:

```
char gore(int n){
    n+10;
   return n;
}

void ravno(int n){
   n-=10;
}

void main(){
    int n=10;
    n = gore(n);
    ravno(n);
    printf("n=%d", n);
}
```

Call by value

Program:

```
void ispisi(int n){
  n--;
  printf(“ni=%d ", n);
}

void main(){
  int n=10;
  ispisi(n);
  ispisi(++n);
  ispisi(n++);
  n-=2;
  printf(“nm=%d ",n);
}
```

Rezultat/ispis:

Call by value

Program:

Rezultat/ispis:

```
int fdp(int n){
  n+=5;
  return n;
}

void main(){
  int n=0, m, k;
  n = fdp(n);
  fdp(m=0);
  k=fdp(fdp(m-n));
  printf("%d%d%d", n, m, k);
}
```

Jednostavne funkcije

Program:

```
int fp(int n, int m){
  return n*m;
}

int fm(int n, int m){
  return n%m;
}

void main(){
  int n=5, m=2, k=6;
  n = fp(fm(n,m),k);
  m = fm(5,fp(m,1));
  k = fp(fp(2,3),fm(3,2));
}
```

Rezultat:

Jednostavne funkcije

Program:

```
int lim(int m){
  if (m<10) return 10*m;
  return -10*m;
}

int mil(int n){
  return n%12;
}

double rec(int m){
  return 10./m;
}

void main(){
  int n=0, m=10;
  n = lim(n+m);
  m = mil(m+8);
  printf("%f",rec(n));
}
```

Rezultat:

Za početno kao u programu (int n=0, m=10;) rezultat je:

Za početno `int n=1, m=1;` rezultat je:

Stog i varijable

Stog – zauzeće memorije?

Program:

Rezultat:

```
int fdr(long n, char z){
  short k=5*n/z;
  return k;
}

double fpr(int m){
  return 1.*m/fdr(300,'*');
}

void main(){
  double dupla;
  dupla = fpr(10000);
}
```

bajta?

Call by reference – bez polja

Program:

Rezultat/ispis:

```
void pomakni(char *c) {
  *c += 2;
}

void main () {
  char c = 'a', z = 'b';
  pomakni(&c);
  pomakni(&z);
  printf ("c='%c'\n", c);
  printf ("z='%c'", z);
}
```

Call by reference – bez polja

Program:

```
int fopa(char *z, short t) {
  (*z)--;
  t-=2;
  return (*z) * t;
}

main () {
  char c = '3';
  int i;
  short t = 3;

  i = fopa (&c, t);
  printf ("%d %d\n", i, t);
  printf ("%d %c\n", c, c);
}
```

Rezultat/ispis:

Call by reference – bez polja

Program:

Rezultat/ispis:

```
int puk(int *i, int j) {
   int k;
    k=*i * j;
    (*i)++;
    j--;
    *i *= j;
    printf ("f: %d ", *i);
    printf ("%d %d", j, k);
     return *i;
}

main () {
  int i, j=5, k=1;
  i = puk (&k, j);
  printf ("\nm: %d ", i);
  printf ("%d %d", j, k);
}
```

Call by reference – 1D polje

Program:

```
int puni(int *ari, int n) {
  int i=0, p=5;
   for(;i<n;i++) ari[i]=p;
  return p;
}

main () {
  int ari[20];
  int i=20, j;
  j = puni(ari, i);
  printf ("%d %d", *ari, ari[0]);
  printf ("\n%d", j);
}
```

Rezultat/ispis:

Call by reference – 2D polje

Program:

```
void fda(int *ari, int n, int m, int stupmax) {
   printf("%d ", *(ari+n*stupmax+m));
   printf("%d \n", ari[n*stupmax+m]);
}

void fdb(int *ari, int n, int m, int stupmax) {
    int i=0, j;
     for (;i<n;i++)
         for (j=0;j<m;j++)
             printf("%d ", ari[i*stupmax+j]);
}


main () {
  int ari[5][10]={{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
  int i=3, j=3, k=10;
  fda(&ari[0][0], i-1, j-1, k);
  fdb(ari[0], i, j, k);
}
```

Rezultat/ispis:

PiPI

VJEŽBE ZA BLITZ 06

Grupa 07,

Z. Šimić, 2008.

Teme za 6. blitz

- Macro s parametrima
- Matematičke ugrađene funkcije
- Vlastite funkcije za rad s nizovima
  - deklariranim kao polje
  - deklariranim kao pokazivač
- Ugrađene funkcije iz `string.h`
- Ugrađene funkcije iz `ctype.h`
- Ulaz/izlaz (`gets`, `puts`)
- Formati za ispis (`printf`)
- Formati za unos (`scanf`)
- Ulaz/izlaz (`getchar`, `putchar`)
- `typedef` (bez strukture)

Macro s parametrima

Program:

Rezultat:

```
#define ta(a,b,c) a*b*c
#define tb(a,b,c) (a)*(b)*(c)
#define tc(a,b,c) ((a)*(b)*(c))

void main(){
  int x, y, z;
  int n=0, m=2, k=1;
  x = !ta(n,m,k);
  y = !tb(n,m,k);
  z = !tc(n,m,k);

  x = ta(n,m,k);
  y = tb(n,m,k);
  z = tc(n,m,k);
}
```

Formati za unos (scanf)

```
char slova[10];

scanf("%6s", slova);
```

Za unos: **Primjer za**

slova=

```
scanf("%[ samo]", slova);
```

Za unos: **samo se**

slova=

```
scanf("%[^NeTo]", slova);
```

Za unos: **nEtONe**

slova=

```
char slova[10];
int i, j;

scanf("%2d%3d", &i, &j);
```

Za unos: **987653**

i= , j=

```
scanf("%2d %s", &i, slova);
```

Za unos: **34 Neki unos**

i= , slova=

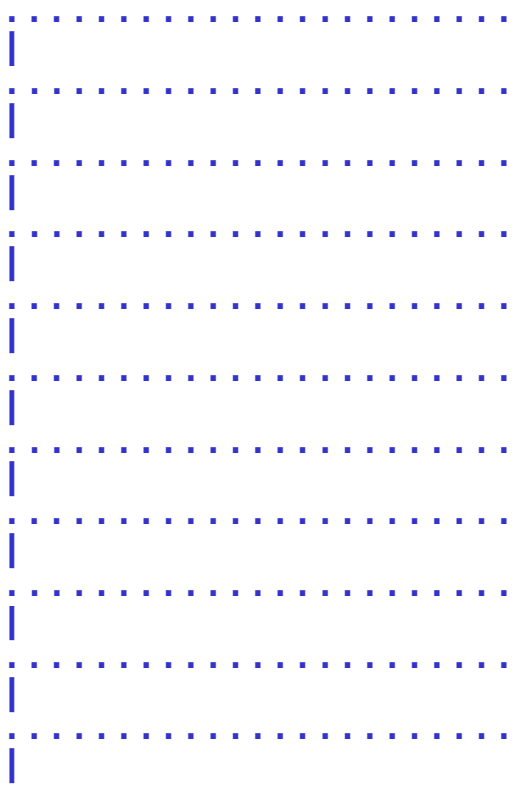
```
scanf("%o", &i);
printf("x%X d%d o%o",i,i,i);
```

Za unos: **17**

Formati za ispis (`printf`)

Ispis:

```
char at[] ="\n.....................\n";
int m = 8, n=100, d = 0xa;
float x = 1.61803, y=-3.14159, q = 2.718;
char ac[20] = "NEKI TEKST", az[]="Znakovi";
printf("%s|%-3d %05.3f %.3s", at, m, x, &ac[5]);
printf("%s|n=%05d, y=%07.3f", at, n, y);
printf("%s|%4.1f %4.2f %4.0f", at, q, q, q);
printf("%s|Broj %.0f.", &at[0], q);
printf("%s|%-10s%-3X-%+3x%3d", at, az, d, d, d);
printf("%s|%-10s %04d %6.3f", at, "DA", 987, -y);
printf("%s|%5.2f%-6.4s", at, 10*x, "zlatni");
printf("%s|%-3d %03d", at, m, -m);
printf("%s|%03d%-5.1f", at, 2, 3.14);
printf("%s|%04d%4d%02d%2d", at, 4, 4, 20, 20);
printf("%s|%05.2f%5.3s\n", at, 1.2345, "Simbol");
```



Ugrađene funkcije iz `string.h`

```
#include <string.h>
...
char st[] = "neki tekst", *nz="znakovi-";
char slova[20];
strncpy(slova,st,4);
printf("%s\n", slova);
slova[4]='\0';
printf("%s\n", slova);
printf("%s\n", strcpy(slova, nz));
printf("%s %d\n", slova+4, strlen(&slova[4]));
strcpy(slova, nz);
printf("%s\n", strstr(st, "tek"));
printf("%s\n", strstr(st, "t")+1);
printf("%c\n", *(strstr(st, "t")+2));
printf("%s\n", strcat(slova, st));
printf("%s\n", strchr(st, 'i'));
printf("%d\n", strcmp(st, nz));
printf("%d\n", strcmp("abc", "ABC"));
```

Ispis:

Ugrađene funkcije iz `ctype.h`

Ispis:

```
#include <ctype.h>

int i=0, v=0, m=0;
char c='c', z='C', s='1';
char niz[]="Testiranje 123 ABCD";

printf("%d %d\n", isupper(z), isalpha(s));
printf("%d %c\n", isupper(c), toupper(c));
for(; niz[i]!='\0'; i++){
   if (islower(niz[i]) m++;
   v += isupper(niz[i]);
   niz[i] = toupper(niz[i]);
}
printf("%d %d\n", v, m);
printf("%s\n", niz);
printf("%d %c\n", tolower(z), tolower(z));
printf("%c\n", z);
```

Ulaz/izlaz (getchar, putchar)

Ispis:

```
int i;
char niz[]="123 abc ABC";
char *ps = "pointer na string";

i=strlen(niz);
while(isupper(niz[--i])){
   putchar(niz[i]+32);
   if (niz[i]=='A') niz[i]='a';
   if (islower(niz[i])) break;
}
putchar('\n');
for(i=0; i<strlen(niz); i++)
   putchar(*(niz+i));
putchar('\n');
for (; *ps; ps++)
   if (*ps >= 'd' && *ps <= 'r')
         putchar(*(++ ps));
printf("\n%s", ps-4);
```

Ulaz/izlaz (getchar, putchar)

```
void fprva(int i, char str[]) {
   for(; str[i]; i++)
      if (isdigit(str[i])) putchar(str[i]);
}
void fdruga (int i, char *str) {
  for (i=strlen(str)-1; i>=0; i--)
      putchar(*(str+i));
}
char *ftreca(int i, char *str) {
   do {
      str[++i] = getchar();
   } while( str[i] != 'X');
   str[i] = 0;
   return str;
}
void main () {
   char niz[]="\nT: ABC 123\n", as[40];
        fprva(0, niz);
        fdruga(0, niz);
        strcpy(as, ftreca(-1, niz));
    printf("%s %s\n", niz, as);
}
```

Ispis:

Unos s tastature:

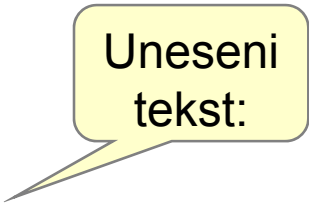
```
test 1X↵
```

Ulaz/izlaz (`gets`, `puts`)

```
char niz[80], z;
char str[] = "Neki tekst", *ps;

puts(str);

str[6]='\0';

puts(str);

puts("Unesi tekst: ");
z = getchar();
ps = gets(niz)+5;

printf("%c%s\n", z, niz);
printf("%s %d", ps, strlen(niz));
```

Rezultat/Ispis:

```
Unesi tekst:
jos jedan test
```



Podsjetnik

- Postoji MS Word dokument sa zadatcima u vezi datoteka