Programski jezik C

Orodja,
Sintaksa,
Vhod-izhod,
Polja,nizi

Zakaj naj bi se učili C?

Praktični razlogi:
C je *de facto* standard za sistemsko programiranje
Programi, pisani v jeziku C so bolj hitri

Izobraževalni razlogi:
Java ne omogoča vpogleda v nizkonivojske mehanizme
Računalnikar bi moral poznati izvajanje programov na različnih nivojih

Prednosti uporabe C:
Boljši nadzor nad obnašanjem programa med njegovim izvajanjem
Več možnosti za uglaševanje performans programa
Dostop do nizkonivojskih mehanizmov sistema

Slabosti uporabe C:
Sami moramo skrbeti za upravljanje s pomnilnikom
Tipično potrebujemo več vrstic kode za izvedbo neke naloge
Več možnosti za napake

Kaj pa C++ ?

C++ je dodal objektno usmerjenost osnovnemu semantičnemu modelu C.
Java je predstavljala implementacijo novega, objektno usmerjenega jezika na osnovi nekaterih konceptov C.

Mnenje: C++ ni čist ... Če hočete programirati objektno usmerjeno, uporabljajte objektno usmerjen jezik.

Primer preprostega programa

```
/* malo komentarja */

#include <stdio.h>

main()
{
   printf("Pozdravljen:\n");
   printf("Kako ti je ime:");
}
```

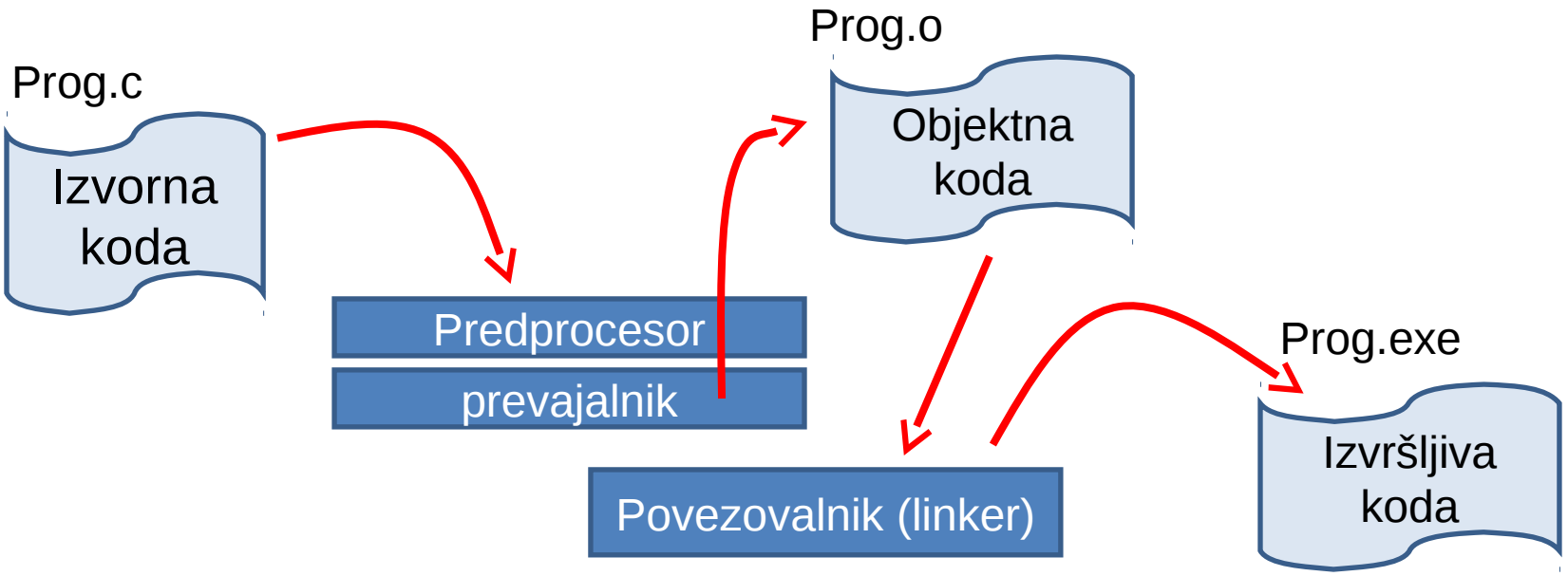
Prevajalnik razlikuje velike in male črke v našem programu.

Namesto besed *begin* in *end* uporabljamo zaviti oklepaj in zaklepaj.

Namesto glavnega programa imamo v kodi obvezno funkcijo *main()*.

Demo

Priprava programa v jeziku C



Program v jeziku C zapišemo v datoteko, ki ima končnico **.c**

Sam prevajalnik ima več podmodulov. Eden od njih je predprocesor, ki obravnava različna navodila oziroma direktive. Te se tipično začno z znakom #

Datoteka s prevodom izvornega programa ima kratico **.o** (objektna datoteka).
Povezovalnik doda druge potrebne podprograme.
Prevedeni (in sestavljen) program ima običajno ime ***a.out***, vendar mu normalno damo bolj pametno ime.

Primeri ukazov za prevajanje

Po kodiranju datoteke “program”, se povrnemo v lupino in vpišemo:

```
gcc program.c
```

Če uporabimo matematične funkcije (exp, sqrt, cos,…), moramo vključiti še matematično knjižnico:

```
gcc program.c -lm
```

Prevajalnik napiše za morebitne napake, v katerih vrsticah so, in opiše napake. .

Opombe:

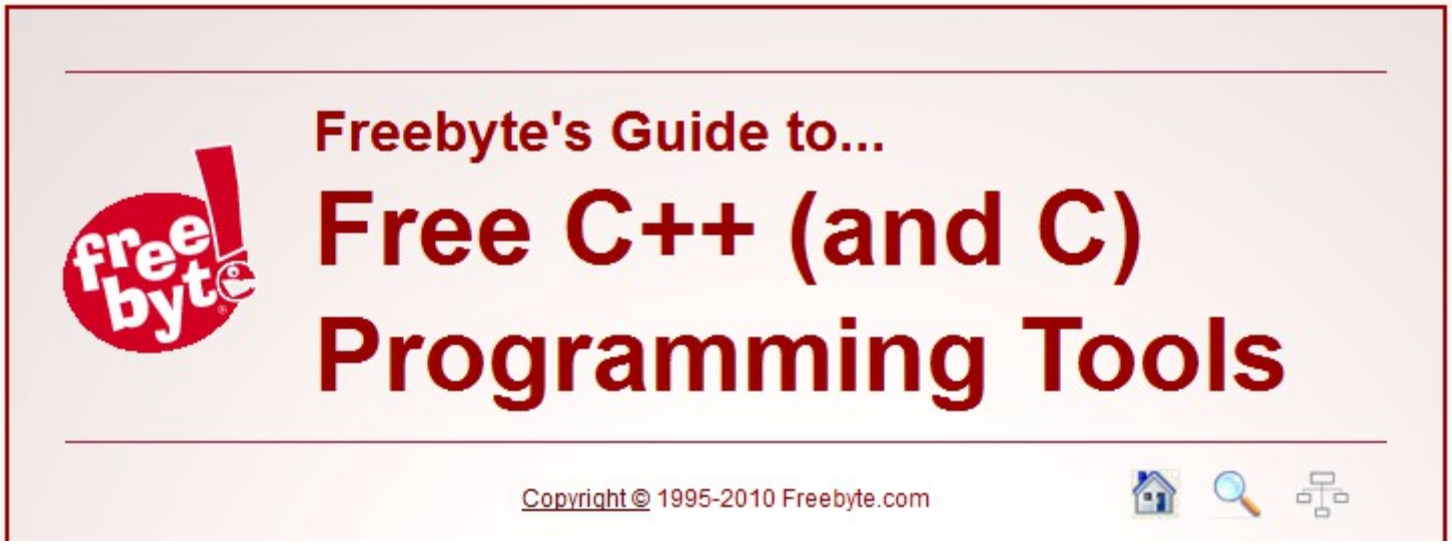
- c-jevske datoteke imajo podaljšek .c, C++ datoteke imajo podaljšek .C (včasih .cpp)
- Knjižnice navajamo v obliki –imeKnjiznice
- Prevedeni (in povezani) program dobi ime a.out (na Linux) ali a.exe (na Windows)

Če hočemo, da prevajalnik/povezovalnik da izvršljivemu programu drugačno ime:

```
gcc program.c -lm -o novoIme
```

DEMO-LINUX

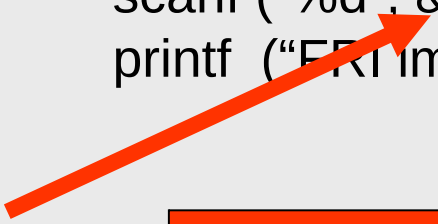
Programska orodja



Quincy 2005
Eclipse
Gcc, lcc
DevC++
Code::Blocks
Pelles C

Še en primer !

```
#include <stdio.h>

void main(void)
{
        int nStudents = 0; /* Initialization, required */

        printf  (“Koliko studentov ima FRI ?:”);
        scanf (“%d”, &nStudents);  /* Read input */
        printf  (“FRI ima %d studentov.\n”, nStudents);

}
```



Pozor na znak & pred imenom spremenljivke

```
$Koliko študentov ima FRI ?: 1600 (enter)
FRI ima 1600 studentov.
$
```

DEMO

Izgled programa v jeziku C

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

main(){

   double a, b;
   printf("Vpisi stevilo\n");
   scanf ("%lf", &a);
   b = sqrt(a);
   printf( "koren od %lf je %lf", a, b);

}
```

header datoteke

Knjižnice funkcij

Skoraj ni programa, ki ne bi imel na začetku navedenih "header" datotek, ki jih mora prevajalnik vključiti v program. Prav tako je običajno, da v programu uporabljamo funkcije, ki so na voljo v posebnih "knjižnicah".

Demo

Spremenljivke in konstante

- **Konstante**
- **spremenljivke**

Atributi:
- ime
- vrednost
- podatkovni tip
- način pomnenja

Podatkovni tipi:
integer
int
long
short
unsigned
realni
float
double
znaki
char

Posebni operator
sizeof( )
pove, koliko bytov zaseda neka podatkovna struktura

Kot pri vsakem programskem jeziku so tudi tu osnovni gradniki programa spremenljivke in konstante.

Preprosti tipi podatkov

Tip	Bytov	Bitov	Območje	
short int	2	16	-16,384 -> +16,383	(16kb)
unsigned short int	2	16	0 -> +32,767	(32 kb)
unsigned int	4	32	0 -> +4,294,967,295	(4Gb)
int	2ali 4	16 ali 32	-2^{15} -> $2^{15}-1$ -2^{31} -> $2^{31}-1$	(2GB)
long int long	4 najmanj	32 najmanj	-2,147,483,648 -> +2,147,483,647	(2GB)
long long	8	64	-2^{63} -> $2^{63}-1$	
char	1	8	-128 -> +127	
unsigned char	1	8	0 -> +255	
float	4	32		
double	8	64		
long double	12	96		

Ni tipa boolean

Isto kot v javi

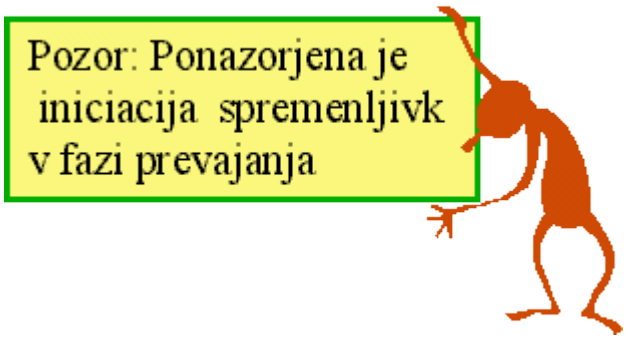
Imena spremenljivk

Začno s črko
Poleg črk lahko vsebujejo številke in znak _
Največ 255 znakov
Razlikujemo velike in male črke
Ne smemo uporabljati rezerviranih besed

Rezervirane besede

auto	break	case	char	const	continue
default	do	double	else	enum	extern
float	for	goto	if	int	long
register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void
volatile	while				

Primeri imen in deklaracij




```
int     a,  numPoints;
char    ch, answer = "\033";
double  value, max_value = 100.0;
```

Pri deklaraciji spremenljivk lahko pomagamo prevajalniku (optimizatorju), tako, da mu napovemo, ali bo neka spremenljivka imela stalno ali spremenljivo vrednost:

```
const double    e= 2.718281828;
volatile char   answer;
```

Razred *volatile prevajalniku pove, da se lahko spremenljivka spreminja v procesih, ki tečejo v ozadju in je torej ni dovoljeno* optimizirati.

Oštevilčeni tipi spremenljivk

Definicija oštevilčenih tipov (enumerated types) ima naslednjo splošno obliko:

enum etiketa {seznam vrednosti};
enum etiketa imeSpremenljivke;

Primer:

```
enum  dnevi {poned,torek,sreda,cetrtek,petek,sobota}; enum
dnevi dan;
. . dan= sreda;
```

Prevajalnik C obravnava oštevilčene označbe kot celoštevilčne konstante (0, 1, 2,..)

Demo

Definicija novih tipov operandov

Splošno:

typedef oldType newName;

Primer:

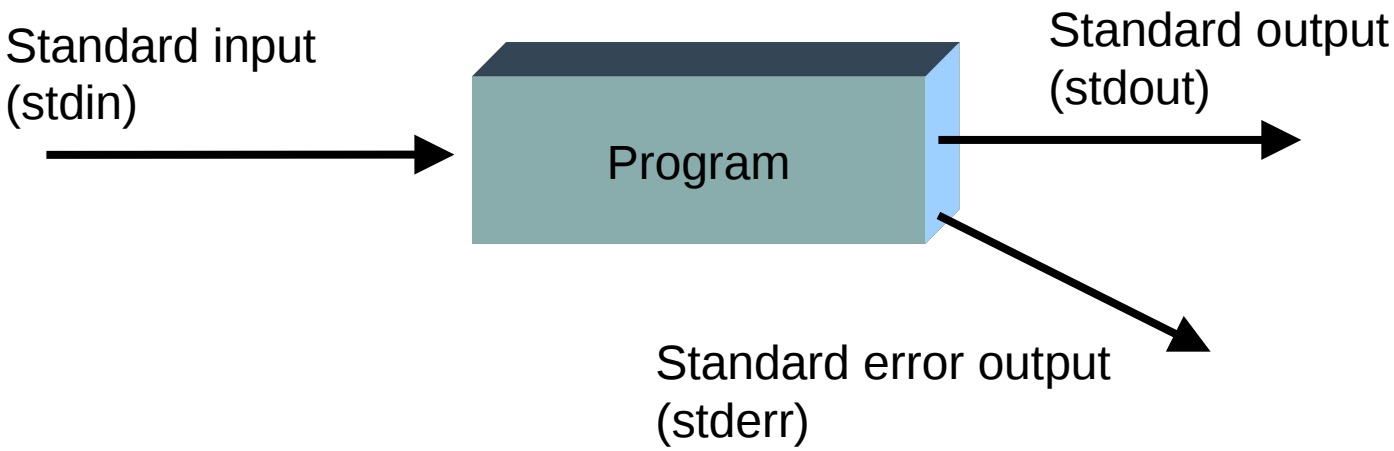
```
enum logical {FALSE, TRUE};
typedef enum logical boolean;
boolean status;
```

Demo

Vhodno izhodne funkcije

Funkcije s standardnim vhodom, izhodom:

int printf (format [,arg, arg,..arg])	Formatiran izpis na standardni izhod
int scanf (format [,kazalec, kazalec, ..])	Formatirano branje s standardnega vhoda
int getchar( )	Branje znaka s standardnega vhoda
int putchar (int)	Izpis znaka na standarni izhod
char* fgets(char str[ ], int length, FILE * stream)	Branje niza s standardnega vhoda
char *puts(char str[80])	Izpis niza na standardni izhod



Formatirano branje in izpis

Primer:

```
#include <stdio.h>
int   starost, stevCevljev;
main( ) {
    printf ("Koliko imas stevilko cevljev:");
    scanf ("%d", &stevCevljev);
    printf ("Torej rabis copate stev %d \n",stevCevljev);
}
```

Pozor na znak & pred imenom spremenljivke v stavku scanf, ker mora biti za vhodni parameter podan naslov in ne vrednost spremenljivke

Splošna oblika:

printf (format, seznam spremenljivk ali konstant);
scanf(format, seznam naslovov spremenljivk);

DEMO

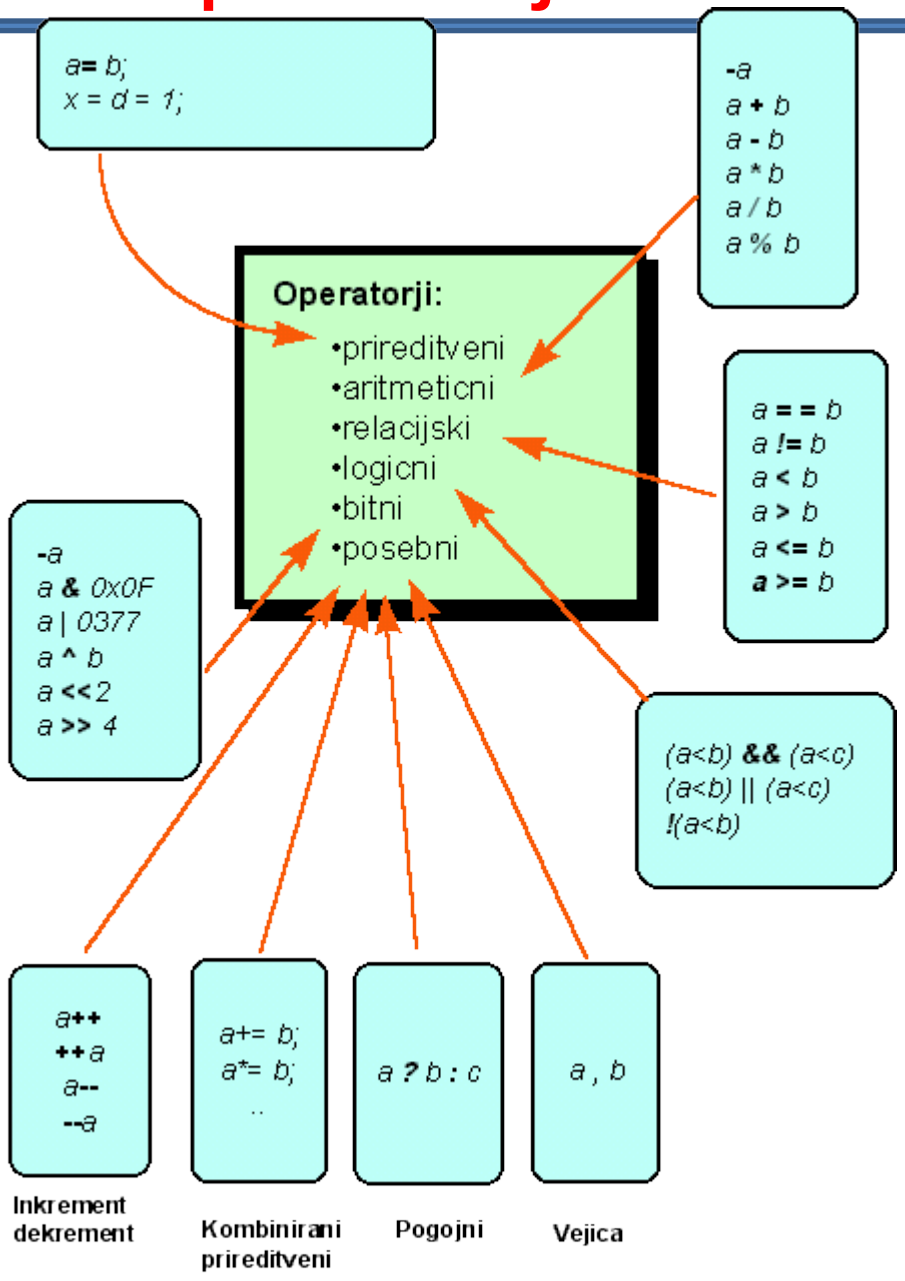
printf in scanf - konverzijske specifikacije

d	Desetiška cela stevila
u	Desetiška cela števila brez predznaka
o	Osmiška števila
x	Šestnajstiška števila (male črke abcdef)
X	Šestnajstiška števila (velike črke ABCDEF)
i	Cela števila, osnova definirana z notacijo
f	Realna števila tipa float
lf	Realna števila tipa double
e	Realna števila,znanstvana notacija (črka e)
E	Realna števila,znanstvena notacija (črka E)
g	Realna števila, format odvisen od velikosti
G	Isto, le črka E namesto e
c	Posamezni znaki
s	Nizi, ki so zaključeni s kodo 0

Demo

Pregled operatorjev in izrazov

Isto kot v javi



Isto kot v javi

Aritmetični operatorji

Operator	Pomen	Primer
+	seštevanje	a+b
-	odštevanje	a-b
*	množenje	a*b
/	deljenje	a/b
%	modulo (ostanek celoštevilčnega deljenja)	a%b
++	pred inkrement (poveča vrednost spremenljivke za 1 in jo nato uporabi)	++a
++	po inkrement (uporabi spremenljivko in ji nato poveča vrednost za 1)	a++
--	pred dekrement (zmanjša vrednost spremenljivke za 1 in jo nato uporabi)	--a
--	po dekrement (uporabi spremenljivko in ji nato zmanjša vrednost za 1)	a--
-	unarni minus	-a
+	unarni plus	+a
+=	seštevanje in prirejanje (a+=b pomeni a=a+b)	a+=b
-=	odštevanje in prirejanje (a-=b pomeni a=a-b)	a-=b
*=	množenje in prirejanje (a*=b pomeni a=a*b)	a*=b
/=	deljenje in prirejanje (a/=b pomeni a=a/b)	a/=b
%=	modulo in prirejanje (a%=b pomeni a=a%b)	a%=b

Logični operatorji

Isto kot v javi

Operator	Pomen	Primer
&	bitni IN (AND)	a&b
|	bitni ALI (OR)	a|b
^	bitni ekskluzivni ALI (XOR)	a^b
~	komplement	~a
!	negacija	!a
<<	pomik levo (a<<b pomakne a za b bitov v levo)	a<<b

Posebni operatorji

Isto kot v javi

Kombinirani prireditveni operatorji:

Splošna oblika:
izraz1 op= izraz2; Primer: a += b;

Pomeni isto kot:
izraz1 = izraz1 op izraz2; Primer: a = a + b;

*(velja za operatorje: + - * / % << & | ^)*

Pogojni operator:

Splošna oblika:
izraz1 ? izraz2 : izraz3 Primer: predznak = (x < 0) ? -1 : 1 ;

Pomen:
Če je vrednost izraz1 TRUE (ni nič), potem je celotni izraz po vrednosti in tipu enak izrazu2 sicer je celotni izraz po tipu in vrednosti enak izrazu3

Operator vejica:
Splošna oblika:
izraz1 , izraz2
Pomen:
Ocenita se oba izraza, celoten izraz ima vrednost in tip desnega izraza.

Isto kot v javi

Izrazi

Imajo tip in vrednost. So kombinacija operandov in operatorjev.

Primeri:

```
pogoj =  a < b;
rezultat = (a > 4) < 6;
stanje = !(a < b);
rezultat = x >> 2; ++a;  /* kar je enako a = a+1 */;
a = --b - 2;
predznak = (x < 0) ? -1 : 1 ;
```

Nepravilna uporaba operatorjev

Isto kot v javi

Izraze pišemo pregledno in nedvoumno!

Slabo:

```
z=++x - y/x--;
z= -x/y;
z= x++ + ++y/z-- *5;
z= (x++==4 || y-- <=5);
```

Dobro:

```
X++; z= x-y/x;  x--;
Z= (-x)/y;
Z= ((x++) + (++y)) / ((z--) *5)
Z= (x==4|| y<=5); x++; y--;
```

WEB

Konverzija tipa podatka

Avtomatična:

Do avtomatične konverzije pride med tipi:
char, short int, int

Potrebna:

V naslednjih dveh primerih imejmo dve spremenljivki:

int a; float b;

Jasno je, da mora priti v naslednjem stavku do konverzije tipa izraza iz **float** v **int**: **a = b;**

Zahtevana:

V naslednjem primeru konverzijo eksplicitno zahtevamo:
a = (int) b;

Konverzija tipa podatka po standardu ANSI

Če je eden od operandov **long double**, bo tak tudi drugi.

Sicer če je en operand **double**, bo tak tudi drugi.

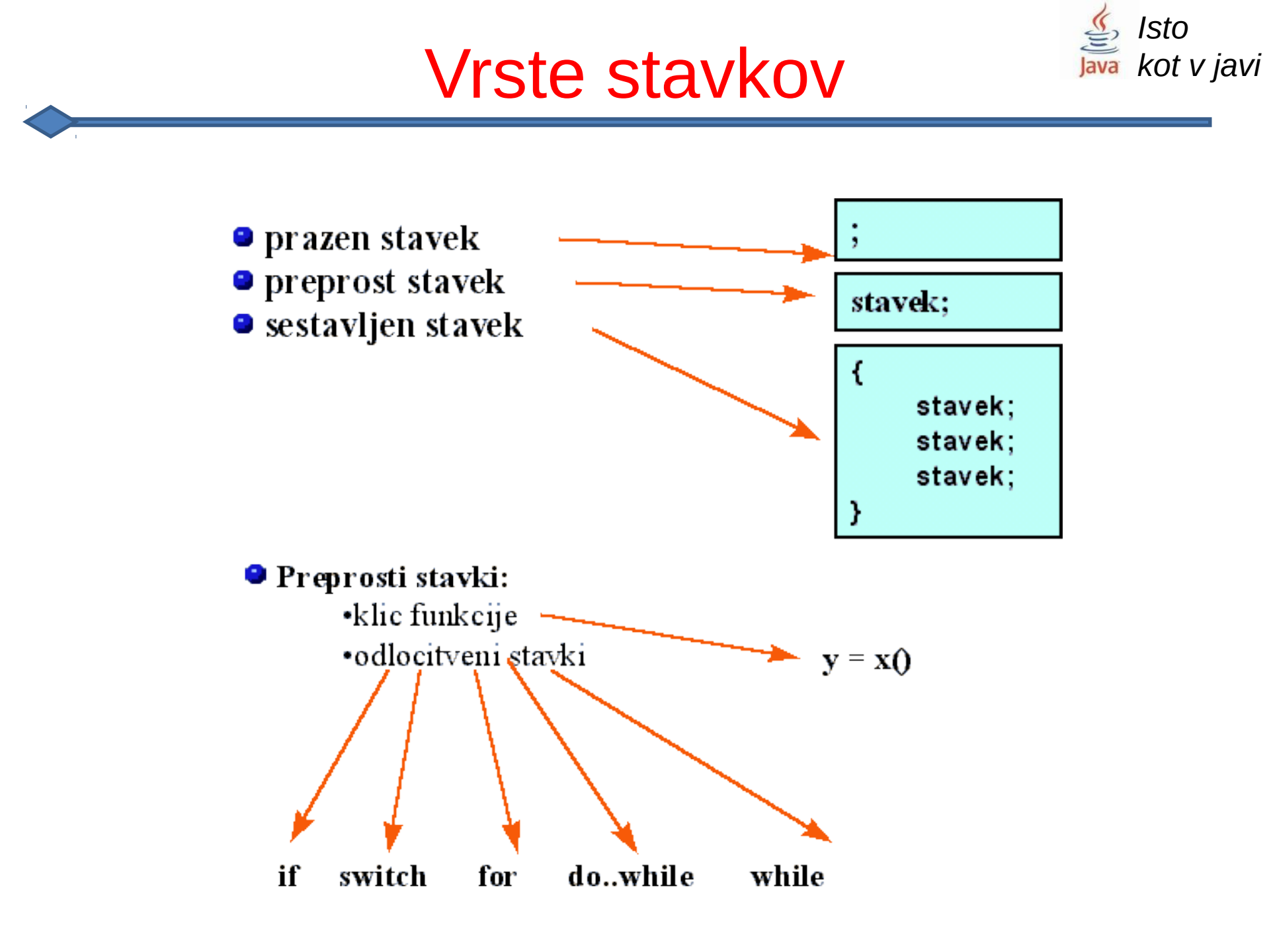
Sicer če je en operand **float**, bo tak tudi drugi.

Operand tipa **char** ali **short int** postane tipa **int**.

Če je en operand tipa **long int**, bo tak tudi drugi.

Sicer pa bo izraz tipa **int**.

Vrste stavkov



Na hitro nekaj o funkcijah

(Ker brez njih pač ne gre)

Splošna oblika klica funkcije:

[vrednost =] imeFunkcije(arg1, arg2,..,argn);

Primeri:
ch = getchar();
printf("Pozdravljeni");

Opomba:
getchar() bere znak, vendar ga dobimo šele po vtipkanju ENTER. Rezultat getchar() je tipa int. To omogoča, da lahko vemo, kdaj je branje neuspešno (ne moremo na primer brati po koncu vhoda, tedaj vrne -1 (kar pomeni EOF (end of file)))

Demo

Odločitveni stavek if

Isto kot v javi

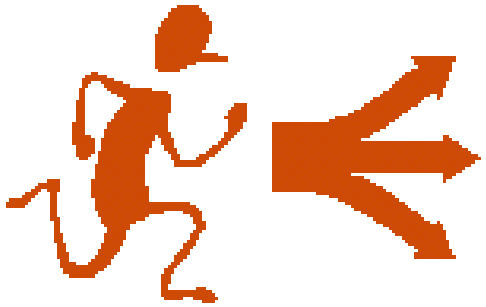
Splošna oblika

if (izraz) stavek1; else stavek2;

Primer:

```
printf("Vpisi x in y:");
scanf("%d %d",&x, &y);
if(x==y)
    printf("Enaki vrednosti\n");
else    printf("Vrednosti sta razlicni \n");
```

Odločitveni stavek switch



Splošna oblika:

```
switch(izraz) {
    case (A):
        stavek_A;
        break;
    case (B):
        stavek_B;
        break;
    default:
        stavek_X;
}
```

Opomba:
Stavek default normalno pišemo na koncu. Lahko pa tudi manjka.

Stavek switch: primer

```
printf ("Izberi eno od moznosti:");
switch( getchar() )   {
        case ('1'):
                vnosPodatkov();
                break;
         case ('2'):
                izracun();
                break;
         case('3');
                izpisRezultatov();
}
```

Iterativni stavki (zanke)

Isto kot v javi

Splošne oblike

for (Inicializacija; Pogoj; Inkrement) stavek;
while (izraz) stavek;
do stavek while (izraz);

Zanke: primeri

```
for(i=1; i<=10;i++)printf("2 X %d = %d\n",i,2*i);/* postevanka */
```

```
while ( (ch = getchar())!= EOF) putchar(ch);     /* Prepis vhoda na izhod */
```

```
float stevilo, vsota = 0;
do{
      printf("Vpisi stevilo:");
      scanf("%f", &stevilo);
      vsota += stevilo;
}while (stevilo != 0) ;
printf(" Vsota je %f\n ", vsota);
```

Demo

Stavki break, continue, goto

Stavek *break*
Povzroči izstop iz (najbolj notranje) zanke tipa for, while ali do..while.
Uporabljamo ga tudi pri zaključku alternativ v odločitvenih stavkih switch

Stavek *continue*
Je podoben stavku break in ga uporabljamo v zankah (for, while, do..while). V razliko od stavka break ne povzroči izstopa iz zanke ampak le preskok vseh stavkov (ki so za njim) v dani iteraciji.

Stavek *goto*
Povzroči direkten prehod na stavek z dano etiketo

Primer:

```
if( failure)  goto errorMessage ;
 .....
errorMessage:  printf( "Action failed");
```

Demo

Polja

Primer deklaracije polja:

```
double  vsota, cena[20];    /* polje ima 20 elementov */
int i;
```

Uporaba:

```
cena[0] = 100.0;          /* prvi element ima indeks 0 */
cena[1] = 150.0;
.....
vsota = 0;
for(i=0,i<20;i++) vsota  += cena[i];
```

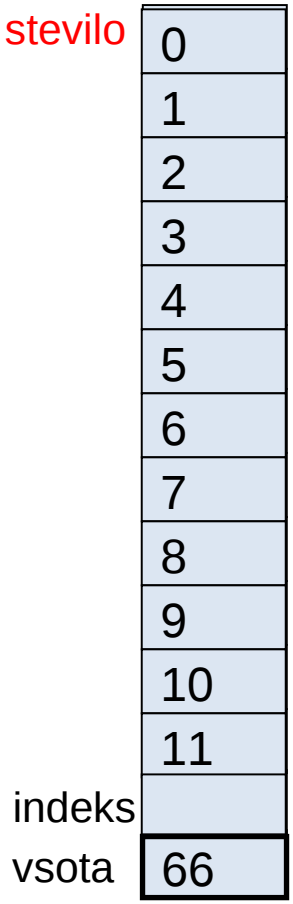
Primeri deklaracije in istočasno iniciacije vrednosti:

```
int dnevi[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
char pozdrav[ ] = {'P','o','z',d','r','a','v','l','j','e','n'};
```

Demo

Enodimenzionalna polja

```
#include <stdio.h>

void main(void)
{
    int stevilo[12];          /* 12 elementov polja */
    int indeks, vsota = 0;
    /* Vedno inicializiraj vrednosti pred uporabo */
    for (indeks = 0; indeks < 12; indeks++) {
            stevilo[indeks] = indeks;
    }
    /* stevilo[indeks]=indeks bi sedaj povzročil napako, zakaj ?*/

    for (indeks = 0; indeks < 12; indeks = indeks + 1) {
            vsota += stevilo[indeks];  /* vsota elementov polja */
    }

    return;
}
```



Demo

Primer s poljem: izpis histograma

Element	Vrednost	Histogram
0	19	**********
1	3	***
2	15	**********
3	7	*******
4	11	**********
5	9	*********
6	13	**********
7	5	*****
8	17	**********
9	1	*

Izpis histograma: koda programa

```
#include <stdio.h>
#define SIZE 10

int main() {
   int n[SIZE] = {19,3,15,7,11,9,13,5,17,1};
   int i,j;

   printf("%s%13s%17s\n", "Element", "Vrednost", "Histogram");

   for (i=0; i<=SIZE-1; i++) {
      printf("%7d%13d        ",i, n[i] );
               for (j=1; j<=n[i]; j++) /* izpis ene vrstice histograma */
                  printf("%c", '*' );
               printf ("\n");
   }
}
```

DEMO

Posredovanje polja funkciji

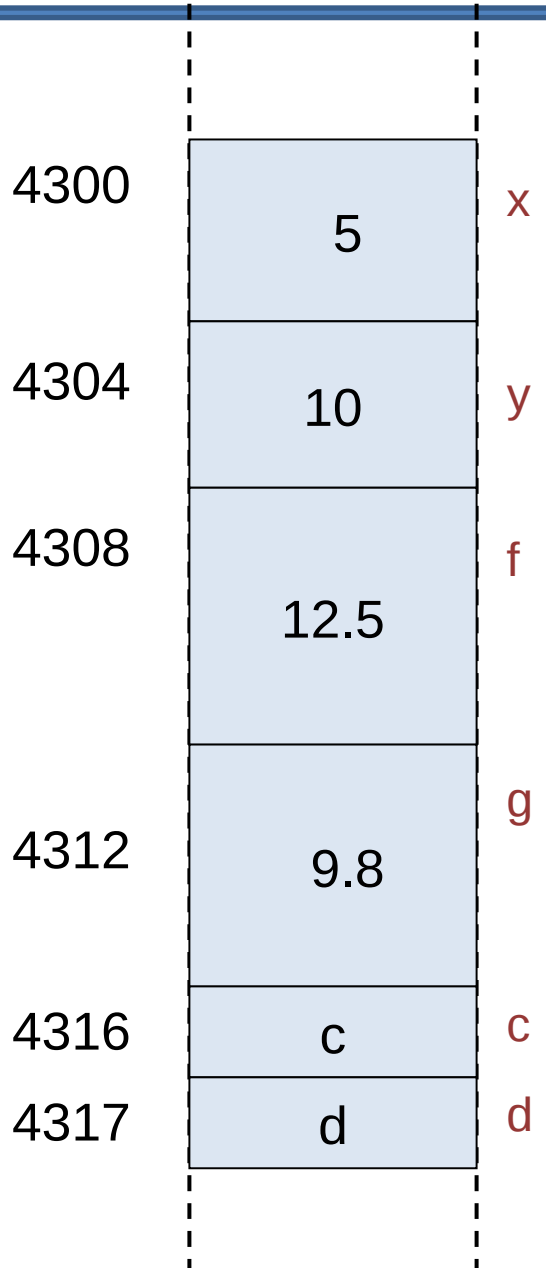
```
void normirajPolje(double p[ ], int n)  {
        /* Funkcija normira polje p z n realnimi stevili */
        double max;
        int i;
        max = fabs(p[0]);
        for (i=0; i<n; i++)
                if (fabs(p[i])> max) max = fabs(p[i]);
        for (i=0; i<n; i++) p[i] = p[i]/max;
}

/*********************************************************/
void main()  {
        int i;
        int num = 5;  /* stevilo elementov v polju */
        double polje[ ] = {10.0, 20.0, 22.0, 15.0, 30.0 };
        normirajPolje(polje, num);
        printf("\nPo normiranju:   ");
        for(i=0;i<num;i++) printf("%lf  ",polje[i]);
        printf("\n");
}
```

*Kopija **naslova** istega polja*

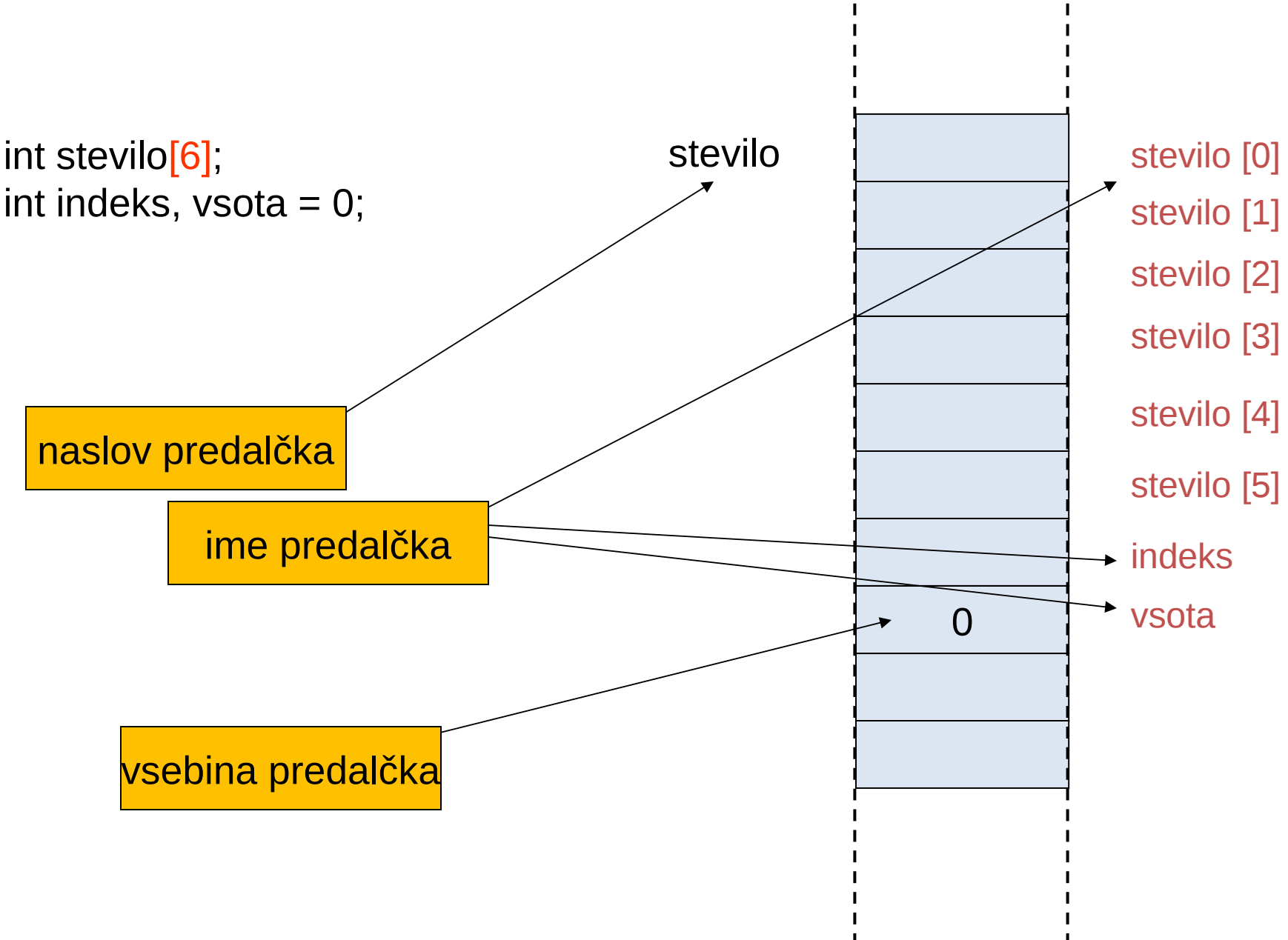
Demo

Izgled pomnilnika in naslovi

```
int    x = 5, y = 10;
float f = 12.5, g = 9.8;
char c = ‘c’, d = ‘d’;
```

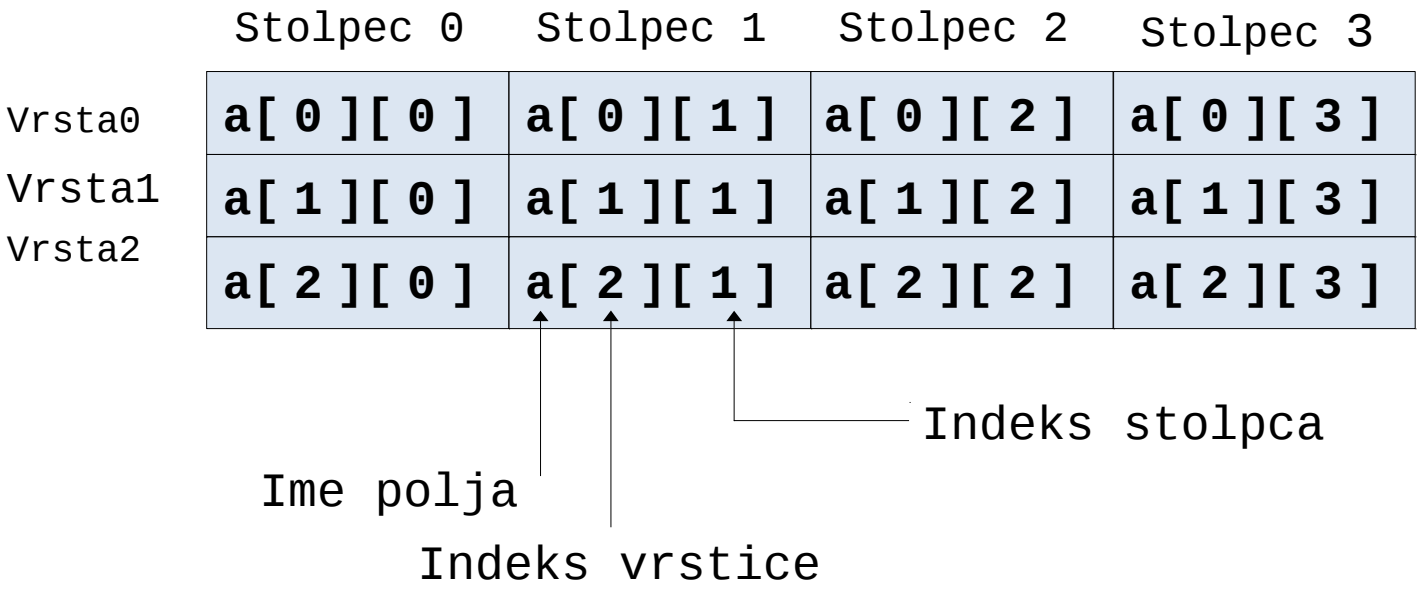


Izgled pomnilnika in naslovi (2)



Večdimenzijska polja

- Polja z več indeksi
  - Tabele z vrsticami in stolpci (polje **m** krat **n**)
  - Kot pri matrikah: najprej povemo vrstico, nato stolpec

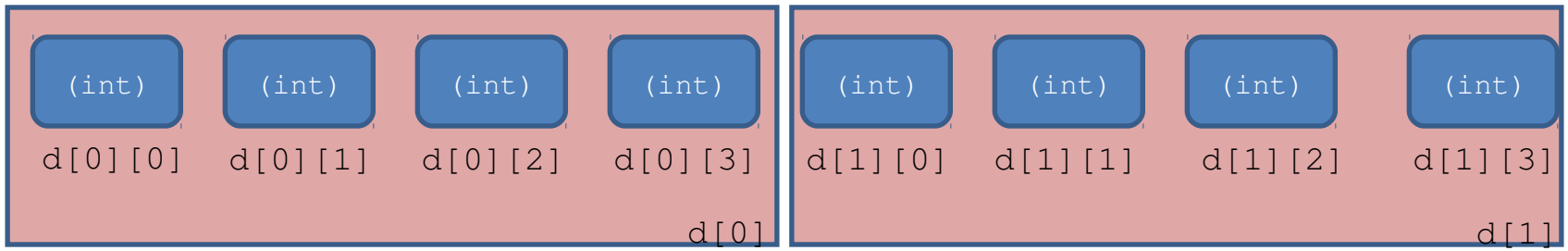


Večdimenzijska polja

- Kako razumemo deklaracijo:

```
int d[2][4];
```

- To je polje dveh elementov:
  - Vsak element je polje štirih vrednosti tipa int
- Elementi so v pomnilniku razporejeni zaporedno, tako kot pri eno dimenzionalnih poljih



Zakaj je pomemben vrstni red pomnenja?

- Če se zadovoljimo s “paradigmo” večdimenzijskega polja, je vrstni red nepomemben…
- Če pa uporabljamo trike z aritmetiko s kazalci, pa je zelo važno
- Pomembno je tudi za inicializacijo
  - Če želimo d inicializirati tako:

0	**1**	**2**	**3**
4	**5**	**6**	**7**

- Uporabimo raje to:

```
int d[2][4] = {0, 1, 2, 3, 4, 5, 6, 7};
```

- Namesto tega

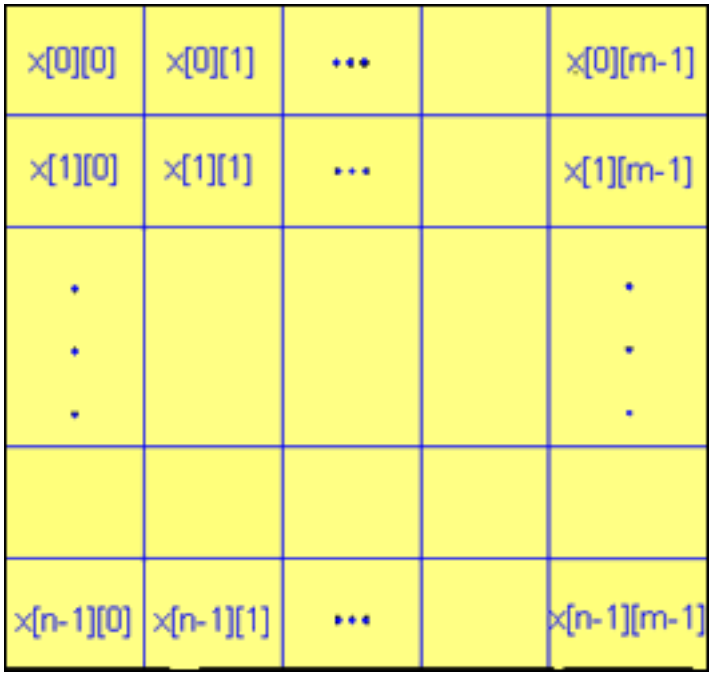
```
int d[2][4] = {0, 4, 1, 5, 2, 6, 3, 7};
```

Večdimenzionalna polja (2)

Primeri polj z numeričnimi vrednostmi:

1	2	3
2	4	6

```
char x [25][80];
int mat[2][3] = {{1,2, 3},{2,4,6}};

/* polje mat prepisemo v polje x   */
/* polje x  naj vsebuje crke !!            */

for(i=0; i<2;i++) {
    for(j=0; j<3;j++) x[i][j] = mat [i][j]+'0';
}
```



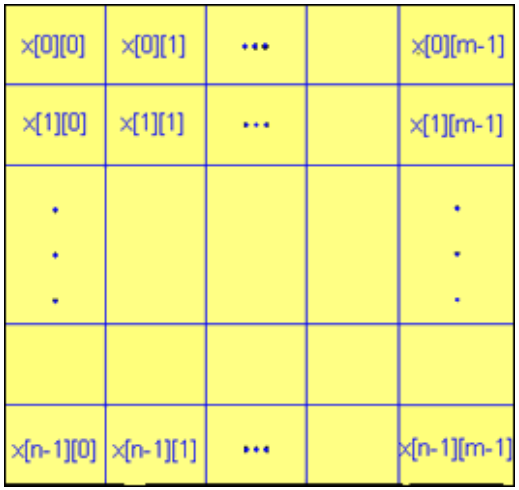
Dvodimenzionalna polja so v C definirana kot enodimenzionalno polje, katerega vsak element je spet polje

Demo

Primer: Branje matrike, račun povprečja

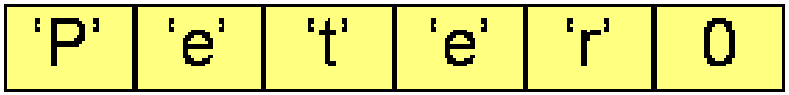
```
#include <stdio.h>
int main (void) {
    double x[10] [10], povpVrstice[10], vsotaMatrike vsotaVrstice, povpMatrike;
    int i,j,m,n;
    FILE *inp, *out;
    inp = fopen("vhod.dat", "r");
    fscanf(inp, "%d%d", &n, &m);
    for (i=0; i<n; ++i){
        for (j=0; j<m; ++j)
           fscanf(inp, "%lf", &x[i][j]);
    }
    fclose (inp);
    vsotaMatrike = 0.0;
    for (i=0; i<n; ++i){
        vsotaVrstice = 0.0;
        for (j=0; j<m; ++j){
            vsotaVrstice += x[i][j];   vsotaMatrike+= x[i][j];
        }
        povpVrstice[i] = vsotaVrstice / (double)m;
    }
    povpMatrike = vsotaMatrike /(double) (n*m);
    out = fopen ("izhod.dat", "w");
    fprintf(out, "POVPRECJA VRSTIC\n");
    for (i=0; i<n; ++i)
      fprintf(out,"%8.3f\n", povpVrstice[i]);
    fprintf(out,"\n\nCELOTNO POVPRECJE = %8.3f", povpMatrike);
    fclose(out);    return(0);
}
```

Pozor na lokacijo teh stavkov glede na zanke for



Demo

Polja znakov (nizi)



char Priimek[6] = {'P','e','t','e','r','\0'};

Lahko pa to deklariramo tudi na boljši način:

char Priimek[ ] = "Peter";

Nize zaključuje znak '\0'

char name[6] = {'L','j','u','b','l' ,'j' ,'a' ,'n' ,'a','\0'}; /* '\0'= konec niza */
printf("%s", name); /* izpisuje do '\0' */

Vhodno izhodne operacije z nizi

Primer 1:

```
#include <stdio.h>
int main (void) {
   char ime[20];
   printf("Kako ti je ime:");
   scanf ("%s",ime);   /* pozor, ni znaka & pred ime */
   printf("Pozdravljen %s", ime);
}
```

In računalnik bo napisal: *Pozdravljen Peter*

Primer 2:

```
char ime[20]; printf("Kako ti je ime:");
fgets(ime,20,stdin);
printf("Pozdravljen %s",ime);
```

In računalnik bo napisal: *Pozdravljen Peter Novak*

DEMO

C reference

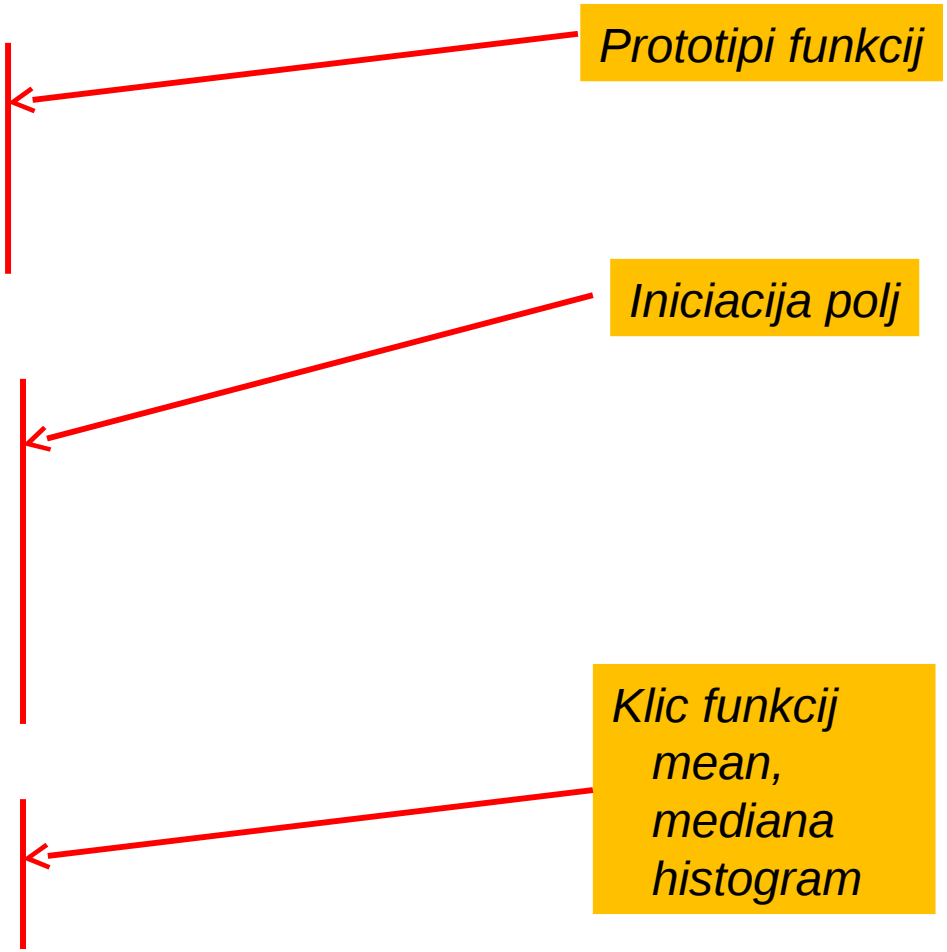
Študij primera: računanje povprečja, mediane, pogostosti

- Računanje povprečja
- Urejanje (sortiranje) elementov polja
- mediana – število na sredini urejenega seznama
  - 1, 2, 3, 4, 5
  - 3 je mediana
- pogostost
  - Kolikokrat nastopa neko število
  - 1, 1, 1, 2, 3, 3, 4, 5
  - Katero število največkrat nastopa? (v našem primeru 1)

DEMO

Študij primera (1 del)

```
#include <stdio.h>
#define SIZE 50
void mean( int [] );
void median( int [] );
void histogram( int [], const int [] ) ;
void bubbleSort( int [] );
void printArray( const int [] );

int main() {
        int pogostost [ 10 ] = { 0 };
        int rezultat[ SIZE ] =  {
            6, 7, 8, 9, 8, 7, 8, 9, 8, 9,
            7, 8, 9, 5, 9, 8, 7, 8, 7, 8,
            6, 7, 8, 9, 3, 9, 8, 7, 8, 7,
            7, 8, 9, 8, 9, 8, 9, 7, 8, 9,
            6, 7, 8, 7, 8, 7, 9, 8, 9, 2};

        mean( rezultat );
        median( rezultat );
        histogram( pogostost, rezultat );
        return 0;
}
```



Srednja vrednost in mediana

```
void mean( int podatki[ ] ) {
        int j, vsota = 0;
    for ( j = 0; j <= SIZE - 1; j++ )  vsota+= podatki[j];
        printf("%d podatkov, vsota je %d, srednja vrednost je %.4f\n\n",
            SIZE, vsota,  (  double  ) vsota / SIZE );
}

void median( int podatki[] ) {
        printf( "Neurejeno polje ocen je" );
        printArray( podatki );
        bubbleSort( podatki );
        printf( "\n\nUrejeno polje ocen je" );
        printArray( podatki );
        printf( "\nV nasem primeru je mediana  %d\n\n",   SIZE / 2, SIZE, podatki[ SIZE / 2 ] );
}
```

Srednja vrednost

mediana

Pogostost in histogram

```
void histogram( int freq[], const int podatki[] ) {
   int ocena, j, h, largest = 0, modeValue = 0;
            for ( ocena = 1; ocena <= 9; ocena++ )  freq[ ocena ] = 0;
            for ( j = 0; j <= SIZE - 1; j++ )          ++freq[ podatki[ j ] ];

            printf( "%s%11s%19sn\n",  "rezultat", "pogostost", "Histogram" );

           for ( ocena = 1; ocena <= 9; ocena++ ) {
                 printf( "%8d%11d          ", ocena, freq[ ocena ] );
                 if ( freq[ ocena ] > largest ) {
                    largest = freq[ ocena ];
                    modeValue = ocena;
                 }
                 for ( h = 1; h <= freq[ ocena ]; h++ ) printf( "*" );
                 printf( "\n" );
            }

   printf( "\nNajbolj pogosta vrednost je %d, ki nastopa %d krat.\n",
              modeValue, largest );
}
```

Pogostost = št. nastopov

Sortiranje (urejanje) podatkov v poljih

- Sortiranje podatkov
  - Pomembno v računalniških aplikacijah
- Bubble sort
  - Preko polja moramo izvesti več prehodov
  - Primerjamo zaporedne pare elementov
    - Če sta elementa v paru v natraščajočem zaporedju ali enaka, ni spremembe
    - Če sta elementa v paru v padajočem zaporedju, ju zamenjamo
  - To ponavljamo v zanki
- Primer:
  - original: 3 4 2 6 7
  - prehod 1: 3 2 4 6 7
  - prehod 2: 2 3 4 6 7
  - Majhni elementi se kot mehurčki vzpenjajo navzgor

Sortiranje (bubble sort)

```
void bubbleSort( int a[] ) {
    int prehod, j, temp;
    for ( prehod = 1; prehod <= SIZE - 1; prehod++ )
        for ( j = 0; j <= SIZE - 2; j++ )
            if ( a[ j ] > a[ j + 1 ] ) {
                temp = a[ j ];
                a[ j ] = a[ j + 1 ];
                a[ j + 1 ] = temp;
            }
}
```

Animacija

Študijski primer: izpis(1)

****
Srednja vrednost
****

Srednja vrednost je povprecje podatkov.
Je enaka vsoti vseh podatkov, deljeno s stevilom podatkov
Imamo 50 podatkov, vsota je 380, srednja vrednost je 7.6000

****
Mediana
****

Neurejeno polje ocen je
6 7 8 9 8 7 8 9 8 9 7 8 9 5 9 8 7 8 7 8
6 7 8 9 3 9 8 7 8 7 7 8 9 8 9 8 9 7 8 9
6 7 8 7 8 7 9 8 9 2

Urejeno polje ocen je
2 3 5 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 8
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 9 9 9
9 9 9 9 9 9 9 9 9 9

Mediana je element 25 urejenega polja z 50 elementi.
V nasem primeru je mediana 8

Študijski primer: izpis(2)

```
**********
Histogram pogostosti
**********
rezultat  pogostost        Histogramn
      1         0
      2         1        *
      3         1        *
      4         0
      5         1        *
      6         3        ***
      7        13         *************
      8        18         ******************
      9        13         *************
```

Najbolj pogosta vrednost je 8, ki nastopa 18 krat.

Iskanje v polju: linearno in binarno

- Iščemo določeno vrednost v polju
- Linearno iskanje
  - preprosto
  - Primerjamo vsak element polja z dano vrednostjo
  - Primerno pri majhnih in neurejenih poljih
- Binarno iskanje
  - Za urejena polja, zelo hitro
  - Primerja **`srednji`** element z dano vrednostjo
    - If equal, najdeno
    - If **`vrednost < srednji`**, pogledamo v prvo polovico polja
    - If **`vrednost > srednji`**, pogledamo v drugo polovico polja
    - ponavljamo

DEMO

Kaj so torej polja?

Sosednje lokacije podatkov enakega tipa
Vrednosti vsakega podatka so seveda lahko različne

Ali lahko imamo na sosednjih lokacijah podatke različnih tipov?

Da! V jeziku C pravimo temu strukture.

Strukture v C

Včasih želimo skupini podatkov zaradi lažje obravnave dati skupno ime. Uvedemo **strukturo**.

Primer:

```
struct address {
        unsigned int houseNumber;
        char streetName[50];
        int zipCode;
        char country[50];
};
```

Več o tem kasneje

Struktura v C je podobna javanskim razredom.
Člani strukture so le spremenljivke,
V razliko od Jave člani niso metode!!

WEB