Polja

Slide 1

- Polje je niz varijabli istog tipa numeriranih cjelobrojnim indeksom.
- Indeks uvijek počinje od nule.
- Radi efikasnosti pristupa elementi polja smještaju se u uzastopne memorijske lokacije.

Primjer:

```
double x[3];  /* polje tipa double od  */
...........   /* tri clana */
x[0]=0.2;
x[1]=0.7;
x[2]=5.15;
/* x[3]=4.4;  greska: x[3] nije definirano.  */
```

Slide 2

Definicija polja:

Jednodimenzionalno polje definira se na sljedeći način:

```
mem_klasa tip ime[izraz];
```

gdje je `mem_klasa` memorijska klasa, `tip` je tip podatka, `ime` je ime polja, a `izraz` mora biti cjelobrojni pozitivni izraz.

- Deklaracija memorijske klase nije obavezna.
- Unutar funkcije polje deklarirano bez memorijske klase je automatska varijabla, a izvan svih funkcija je statička varijabla.
- Unutar funkcije polje se može učiniti statičkim pomoću identifikatora memorijske klase `static`.
- `izraz` u definiciji polja je najčešće pozitivna konstanta ili simbolička konstanta.

Slide 3

Inicijalizacija polja:

Sintaksa:

```
mem_klasa tip ime[izraz]={v_1,v_2,...,v_n};
```

što daje `ime[0]= v_1`, `ime[1]= v_2`, itd.

Primjer,

```
float v[3]={1.17,2.43,6.11};
```

je ekvivalentmo s

```
float v[3];
v[0]=1.17;
v[1]=2.43;
v[2]=6.11;
```

Dimenzija polja se izračunava automatski:

```
float v[]={1.17,2.43,6.11};
```

Slide 4

- Ako je broj inicijalizacijskih vrijednosti veći od dimenzije polja javlja se greška.
- Ako je broj inicijalizacijskih vrijednosti manji, onda će preostale vrijednosti biti inicijalizirane nulom.
- Polja znakova mogu se inicijalizirati stringovima.

Primjer,

```
char c[]="tri";
```

definirano je polje od 4 znaka: `c[0]='t', c[1]='r', c[2]='i', c[3]='\0'`. Takav način pridruživanja moguć je samo pri definiciji varijable.

```
c="tri"; /* pogresno */
```

Slide 5

```
#include <stdio.h>  /* Program ucitava ime i prezime */
#include <string.h> /* i ispisuje ih u jednom retku  */
char ime          [128];
char prezime      [128];
char ime_i_prezime[128];
int main(void) {
    printf("Unesite ime:"); gets(ime);
    printf("Unesite prezime:"); gets(prezime);
    strcpy(ime_i_prezime,ime);
    strcat(ime_i_prezime," ");
    strcat(ime_i_prezime,prezime);
    printf("Ime i prezime: %s\n", ime_i_prezime);
    return 0;
}
```

Slide 6

Datoteka `<string.h>`:

deklarira mnoge druge funkcije iz standardne biblioteke koje služe za rad sa stringovima.

`char *strcpy(char *s, const char *t)`

kopira `t` u `s` uključujući `'\0'`; vraća `s`.

`char *strcat(char *s, const char *t)`

nadovezuje niz `t` na niz `s`; vraća `s`.

`int strcmp(const char *s, const char *t)`

uspoređuje dva niza. Vraća nulu ako su nizovi jednaki.

`size_t strlen(const char *t)`

daje duljinu znakovnog niza.

Slide 7

Polje kao argument funkcije:

- Prevodilac ime polja `v` pri pozivu funkcije pretvara u pokazivač na prvi element polja.
- Funkciju `f` koja uzima polje tipa `tip` možemo deklarirati kao:

  `f(tip v[])` ili `f(tip *v)`

- Unutar funkcije elementi polja mogu se dohvatiti (i promijeniti) pomoću indeksa polja.

Primjer: računanje aritmetičke sredine.

Slide 8

```
double srednja_vrijednost(int n, double v[]){
    int i;
    double rez=0.0;

    for(i=0;i<n;i++) rez+=v[i];
    return rez/n;
}

int main(void){
    int n;
    double v[]={1.0,2.0,3.0},sv;

    n=3;
    sv=srednja_vrijednost(n,v);
    return 0;
}
```

Slide 9

Višedimenzionalna polja

Deklaracija:

`mem_klasa tip ime[izraz_1][izraz_2]...[izraz_n];`

Na primjer, polje `m` deklarirano sa

`static float m[2][3];`

predstavlja matricu s dva retka i tri stupca. Njene elemente možemo prostorno zamisliti na sljedeći način:

`m[0][0]` `m[0][1]` `m[0][2]`
`m[1][0]` `m[1][1]` `m[1][2]`

Slide 10

- Dvodimenzionalno polje je jednodimenzionalno polje čiji su elementi jednodimenzionalna polja.

Npr. polje

`float m[2][3];`

ima dva elementa `m[0]` i `m[1]` tipa `float[3]` (jednodimenzionalna polja dimenzija 3). To su reci matrice `m`.

Poredak elemenata u memoriji:

`m[0][0]` `m[0][1]` `m[0][2]` `m[1][0]` `m[1][1]` `m[1][2]`

Preciznije element `m[i][j]` biti će na `l`-tom mjestu u memoriji, gdje je `l=i*MAXY+j`, a `MAXY=3` je broj stupaca matrice.

Slide 11

- Višedimenzionalno polje je jednodimenzionalno polje čiji su elementi polja dimenzije manje za jedan.

Primjer:

`float m[2][3][4];`

`m[0]` i `m[1]` su polja tipa `float[3][4]`.

Element `m[i][j][k]` bit će smješten u memoriju na mjesto

`i*MAXY*MAXZ+j*MAXZ+k`,

gdje su `MAXX=2`, `MAXY=3` i `MAXZ=4` dimenzije polja.

- Prva dimenzija (`MAXX`) nije nužna za indeksiranje.

Slide 12

Inicijalizacija (1):

```
static float m[2][3]={1.0,2.0,3.0,4.0,5.0,6.0};
```

Inicijalne vrijednosti će biti pridružene elementima polja onako kako su oni smješteni u memoriji:

```
m[0][0]=1.0,  m[0][1]=2.0,  m[0][2]=3.0,
m[1][0]=4.0,  m[1][1]=5.0,  m[1][2]=6.0.
```

Inicijalne vrijednosti je moguće grupirati:

```
static float m[2][3]={{1.0,2.0,3.0},
                      {4.0,5.0,6.0}
                     };
```

Slide 13

Inicijalizacija (2):

- Prvu dimenziju polja prevodilac može izračunati iz inicijalizacijske liste:

```
char A[][2][2]={ {{'a','b'},{'c','d'}},
                 {{'e','f'},{'g','h'}}
               };
```

$$\texttt{A[0]} = \begin{bmatrix} \texttt{a} & \texttt{b} \\ \texttt{c} & \texttt{d} \end{bmatrix} \quad \texttt{A[1]} = \begin{bmatrix} \texttt{e} & \texttt{f} \\ \texttt{g} & \texttt{h} \end{bmatrix}$$

Slide 14

Višedimenzionalno polje kao argument funkcije:

- Kada je višedimenzionalno polje argument funkcije ono se može deklarirati sa svim svojim dimenzijama ili sa svim dimenzijama osim prve.

```
int mat[MAXX][MAXY];
........
void readinput(int mat[MAXX][MAXY], int n, int m)
```

`n` i `m` su stvarni brojevi redaka i stupaca koje treba učitati. Drugi način:

```
void readinput(int mat[][MAXY], int n, int m)
```

ili

```
void readinput(int (*mat)[MAXY], int n, int m)
```

Broj redaka nije bitan za adresiranje elemenata matrice.

Slide 15

```
#include <stdio.h>
char A[][2][2]={ {{'a','b'},{'c','d'}},
                 {{'e','f'},{'g','h'}}};
void f(char a[2][2]);

int main(void) {
      printf("Matrica A[0]:\n");
      f(A[0]);
      printf("Matrica A[1]:\n");
      f(A[1]);  return 0;
}

void f(char a[2][2]) {
     printf("%c %c\n",a[0][0],a[0][1]);
     printf("%c %c\n",a[1][0],a[1][1]);
}
```

Slide 16

Ispis programa:

```
Matrica A[0]:
a b
c d
Matrica A[1]:
e f
g h
```

Slide 17

Množenje matrica:

```
#include <stdio.h>
double A[2][2]={{1.0,2.0},{3.0,4.0}};
double B[2][3]={{0.0,1.0,0.0},{1.0,0.0,1.0}};
double C[2][3];

int main(void) {
     int i,j,k;
     for(i=0;i<2;++i)
        for(j=0;j<3;++j)
           for(k=0;k<2;++k)
              C[i][j]+=A[i][k]*B[k][j];
     /*  .... ispis matrice C  ... */
}
```

Slide 18

Efikasniji kôd:

```
.....
for(i=0;i<2;++i)
   for(k=0;k<2;++k)
      for(j=0;j<3;++j)
         C[i][j]+=A[i][k]*B[k][j];

.....
```

Ovdje se u najdubljoj petlji dohvaćaju reci matrica `C` i `B`, što je efikasnije od kôda koji dohvaća elemente stupca matrice.

Slide 19

Polja varijabilne dimenzije

Stabdard C99 uvodi polja varijabilne dimenzije: to su automatska polja čije dimenzije mogu biti zadane putem varijabli

```
int n=3;
int m=3;
double a[n][m];   // PVD
```

Osnovna upotreba: pisanje funkcija koje kao argument uzimaju dimenzije polja:

```
double Fnorm(int n, int m, double a[n][m]); // a je PVD
```

Primjer:

Slide 20

Računje Frobeniusove norme matrice:

```
double Fnorm(int n, int m, double a[n][m])
{
     double norm=0.0;
     int i,j;

     for(i=0;i<n;++i)
        for(j=0;j<m;++j)
            norm += a[i][j]*a[i][j];

     return sqrt(norm);
}
```

Glavni program:

Slide 21

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

double Fnorm(int n, int m, double a[n][m]);
void vandermond(int n, int m, double a[n][m]);

int main(void)
{
    int n=3,m=3;
    double a[n][m];
    vandermond(n,m,a);
    print(n,m,a);
    printf("norma matrice = %f\n", Fnorm(n,m,a));
    return 0;
}
```

Slide 22

Funkcija `vandermond` inicijalizira matricu.

```
void vandermond(int n, int m, double a[n][m])
{
     int i,j;

     for(i=0;i<n;++i)
        for(j=0;j<m;++j)
             a[i][j] = 1.0/(2.0+i+j);
}
```

Slide 23