Složene strukture podataka: nizovi (polja)

Složene strukture podataka

- Nakon odslušanog bit ćete u stanju:
  - objasniti što je polje
  - navesti primjere definicije i inicijalizacije polja
  - koristiti polje kao argument funkcije
  - objasniti vezu pokazivača i polja.

Osnovne karakteristike

- Polje je niz varijabli istog tipa koje su numerirane cjelobrojnim indeksom.
- Indeks uvijek počinje od nule.
- Elementi polja smještaju se u „uzastopne” memorijske lokacije.

- Jednodimenzionalno polje definira se na sljedeći način:

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

gdje su:

Jednodimenzionalna polja

- `mem_klasa` - memorijska klasa; deklaracija memorijske klase nije obavezna
- `tip` - tip podatka
- `ime` - ime polja
- `izraz` - cjelobrojni pozitivni izraz

- Jednodimenzionalno polje inicijalizira se na sljedeći način:

$$\texttt{mem\_klasa tip ime[izraz]=\{}v_0,v_1,\ldots,v_{n-1}\texttt{\};}$$

Definicija polja

- Definicija:

  `int` polje[10];

  `float` v[MAX];

  `char` ime_i_prezime[128];

- Pridruživanje:

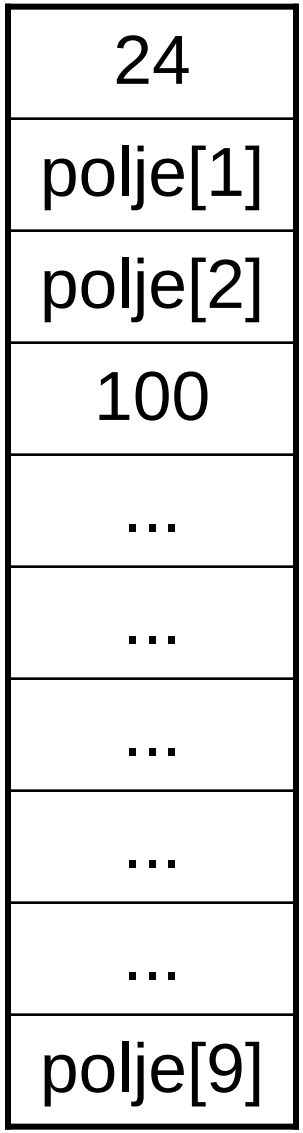
  polje[0] = 24;

  polje[3] = 100;

  polje[3]++;

  v[0] = 1.17f;

  v[1] = 2.34f;



Inicijalizacija polja

```
int tjedan[7] = {1, 2, 3, 4, 5, 6, 7};
char znak[10] = {'+', '-'};
```

- Dimenzija polja se izračunava automatski:

```
float v[ ] = {1.17, -2.34, 9.23};
```

Primjer: s

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

definirano je polje od 4 znaka: c[0] = 'd', c[1] = 'v', c[2] = 'a' i c[3] = '\0'.

Primjer: računanje aritmetičke sredine

```
int main( ){
   int i, n;
   double a_sredina = 0.0;
   double v[ ] = {2.0, 3.11, 4.05, -1.07};

   n = sizeof(v)/8;
   for(i = 0; i < n; i++) a_sredina += v[i];

   a_sredina /= n;
   printf("Aritmetička sredina iznosi %20.12f\n", a_sredina);
   return 0;
}
```

Pokazivači i jednodimenzionalna polja

- Ime jednodimenzionalnog polja je konstantni pokazivač na prvi element polja!

Primjer:

```
int a[10], b[10];
……
a = a + 1; /* Greška, a je konstantni pokazivač. */
b = a; /* Greška! */
```

Pokazivači i jednodimenzionalna polja

Primjer:

```
int a[10], *pa;
……
pa = a; /* ekvivalentno je s pa = &a[0]; */
pa = pa + 2;  /* Nije pogreška - &a[2] */
pa++;  /* &a[3] */
```

Primjer:

```
int a[10], *pa;
……
pa = &a[0];
*(pa + 3) = 20; /* ekvivalentno je s a[3] = 20; */
*(a + 1) = 10; /* ekvivalentno je s a[1] = 10; */
```

Primjer: važnost prioriteta

```
int a[4] = {0, 10, 20, 30};
int *ptr, x;
ptr = a;
```

izraz	x	ptr
x = *ptr;	0	1245040
x = *ptr++;	0	1245044
x = (*ptr)++;	10	1245044
x = *++ptr;	20	1245048
x = ++(*ptr);	21	1245048

Polje kao argument funkcije

- Funkciju f koja uzima polje v 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

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

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

Polje kao argument funkcije (2)

```
int main( ){
   int n;
   double a_sredina = 0.0;
   double v[ ] = {2.0, 3.11, 4.05, -1.07};

   n = sizeof(v)/8;

   a_sredina = as(n, v);

   printf("Aritmeticka sredina iznosi %20.12f\n", a_sredina);
   return 0;
}
```

Višedimenzionalna polja

- Višedimenzionalno polje definira se na sljedeći način:

  `mem_klasa tip ime[izraz_1]…[izraz_n];`

- Na primjer, `static int` m[2][3] predstavlja matricu s 2 retka i 3 stupca.
- Poredak elemenata polja m u memoriji:

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

- m[i][j] $\longrightarrow$ pozicija_u_memoriji=i$*$broj_stupaca+j (Pozicija u memoriji ne ovisi o broju redaka!)
- Dvodimenzionalno polje je jednodimenzionalno polje čiji su elementi jednodimenzionalna polja.

Inicijalizacija

```
int M[4][3] = {{10, 5, -3},
               {9, 18, 0},
               {32, 20, 10},
               {-1, 0, 8}};
M[2][1] = 20;
```

M	[0]	[1]	[2]
[0]	10	5	-3
[1]	9	18	0
[2]	32	20	10
[3]	-1	0	8

- Inicijalne vrijednosti ne moraju biti grupirane:

```
int M[4][3]={10, 5, -3, 9, 18, 0, 32, 20, 10, -1, 0, 8};
```

Inicijalizacija (2)

Primjer:

```
int M[4][3] = {{10, 5, -3}, {9, 18, 0},
               {32, 20, 1}, {-1, 0, 8}};

printf ("%d\n", sizeof(M));
printf ("%d\n", sizeof(M[0]));
printf ("%d\n", sizeof(M[0][0]));
```

Primjer: množenje matrica

```
/* c = a * b, a = a[m][n], b = b[n][p] */

int i, j, k;
……
for(i = 0; i < m; i++)
   for(j = 0; j < p; j++){
        c[i][j] = 0;
        for(k = 0; k < n; k++)
                c[i][j] += a[i][k] * b[k][j];
   }
```

Pokazivači i višedimenzionalna polja

Primjer:

```
int a[2][3] = {{1, 2, 3}, {4, 5, 6}}, *pa;
……
pa = a; /* ekvivalentno je s pa = &a[0][0]; */
pa = pa + 3;
printf("%d\n", *pa);          /* 4 */
printf("%d\n", *(a[1]));      /* 4 */
```

- a[i][j] ⇔ *(a[i] + j) ⇔ *(*(a + i) + j)

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 funkcija(int mat[MAXX][MAXY], int n, int m);
```

gdje su m i n stvarni brojevi redaka i stupaca.

- Drugi načini su:

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

Primjer: množenje matrica

```
#include <stdio.h>
#define MAX_m 10
#define MAX_n 10
#define MAX_p 20

int main( )
{  int a[MAX_m][MAX_n], b[MAX_n][MAX_p], c[MAX_m][MAX_p];
   int m, n, p;
   int i, j;
   void umnozak(int,int,int,int mat1[MAX_m][MAX_n], int
   mat2[][MAX_p], int mat3[][MAX_p]);

   printf("Učitajte dimenzije matrica\n");
   scanf("%d %d %d", &m, &n, &p);
```

Primjer: (2)

```
/* Učitavanje matrica */
    for(i = 0; i < m; i++)
        for(j = 0; j< n; j++)
            scanf("%d", &a[i][j]);
    for(i = 0; i < n; i++)
        for(j = 0; j <p ; j++)
            scanf("%d", &b[i][j]);
    umnozak(m, n, p, a, b, c);
    /* Ispis rezultata */
    for(i = 0; i < m; i++){
        for(j = 0; j < p; j++)
            printf("%d ", c[i][j]);
        printf("\n");}
    return 0;
}
```

Primjer: (3)

```
void umnozak(int m, int n, int p, int mat1[MAX_m][MAX_n],
   int mat2[][MAX_p], int mat3[][MAX_p]){

   int i, j, k;

   /* Mnozenje matrica */
   for(i = 0; i < m; i++)
        for(j = 0; j < p; j++){
                mat3[i][j] = 0;
                for(k = 0; k < n; k++)
                        mat3[i][j] += mat1[i][k] * mat2[k][j];
       }
}
```