Programiranje u fizici

11. Nizovi i matrice

Prirodno-matematički fakultet u Nišu
Departman za fiziku

Tabele predstavljaju složenu strukturu podataka, čiji svi elementi poseduju isti tip.

Tipični primer tabele predstavljaju rezultati nekog merenja. Ovi podaci sadrže nekoliko desetina ili stotina vrednosti istog tipa. Njihovim grupisanjem u jednu tabelu je moguće u mnogome pojednostaviti operacije nad njima (deklarisanje, inicijalizairanje, itd.)

Tabele mogu biti jednodimenzionalne ili višedimenzionalne.

Jednodimenzionalne tabele se obično nazivaju vektori, dok se dvodimenzionalne tabele nazivaju matrice.

Da bi neki podatak bio deklarisan kao jednodimenzionalna tabela, potrebno je navesti njegovo ime, broj elemenata koje sadrži, kao i tip elemenata.

Opšti oblik deklaracije jednodimenzonalne tabele je dat sa:

```
tip ime[n];
```

gde je **ime** neki identifikator, koji definiše ime tabele,

dok **n** predstavlja broj elemenata tabele a

tip određuje tip svakog pojedinačnog elementa tabele.

Elementi tabele mogu pripadati jednom od elementarnih tipova podataka, a mogu biti pokazivači, tabele ili strukture.

Elementi jednodimeznionalne tabele se uvek smeštaju u memoriju računara neposredno jedan za drugim, pri čemu početni element tabele ima najmanju, poslednji najveću adresu.

Primer 1.

Deklaracija jednodimenzionalnih tabela.

```
int stranica[38];
char red [68];
```

`stranica` predstavlja jednodimenzionalnu tabelu sa 38 elemenata, dok `red` ima 68 elemenata.

Par uglastih zagrada `[...]` predstavlja indeksni operator, koji sadrži indeks tabele.

Indeks tabele je predstavljen nekim konstantnim izrazom i on označava poziciju jednog određenog elementa tabele u odnosu na početni.

Zbog toga prvi element tabele ima uvek index 0.

Drugi element ima indeks 1, jer je udaljen od početnog elementa za jednu poziciju itd.

Poslednji element ima indeks za jedan manji od broja elemenata tabele.

Moguće je inicijalizovati jednodimenzionalnu tabelu neposredno prilikom deklaracije na sledeći način:

```
double racun[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

Moguće je izostaviti broj elemenata niza pri inicijalizaciji:

```
double racun[] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

a sam prevodilac izračunava broj elemenata tabele na osnovu datog broja početnih vrednosti.

Moguće je inicijalizovati članove vektora i na sledeći način:

```
racun[0] = 1000.0;
racun[1] = 2.0;
racun[2] = 3.4;
racun[3] = 17.0;
racun[4] = 50.0;
```

0	1	2	3	4
1000.0	2.0	3.4	7.0	50.0

Primer 2.

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

int main()
{
  int n[10]; // n je vektor od 10 integera
  int i, j ;

  // Inicijalizacija vektora
  for (i = 0; i < 10; i++) { //
    n[i] = i + 100;
  }

 // stampanje vrednosti vektora
 for (j = 0; j < 10; j++) {
   printf("Element[%d] = %d\n", j , n[j] );
 }

 return 0;
}
```

```
Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

Primer 3.

U sledećem programskom primeru je definisana tabela sa 26 elemenata i u njoj su smeštena sva velika slova abecede, koja pripadaju ASCII-kodu.

Program učitava broj, koji unosi korisnik i ispisuje veliko slovo abecede, koje tom broju odgovara.

ASCII tabela

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

65+0 = 65 -> A
65+1 = 66 -> B
65+2 = 67 -> C
65+3 = 68 -> D
65+4 = 69 -> E
65+5 = 70 -> F
65+6 = 71 -> G
65+7 = 72 -> H
65+8 = 73 -> I

.
.
.

65+25 = 90 -> Z

Primer 3.

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
  char slovo[26];
  int i, broj;

  for (i=0; i<=25; i++) {
    slovo[i] = 65 + i; // 65 je decimalna vrednost od A u ASCII kodu
  }

  do {
    printf ("\nUpisite celobrojnu vrednost izmedju 1 i 26 \n");
    scanf("%d", &broj);

    if ((broj < 1) || (broj > 26)){
      printf("\nPogresan broj ! \n");
    }
    else {
      printf("\n Na %d. mestu abecede se nalazi ", broj);
      printf("slovo %c", slovo[broj-1]);
    }
  } while ( (broj >= 1) && (broj <= 26) );
  return 0;
}
```

U programskom jeziku C svaka dimenzija višedimenzionalne tabele se piše unutar jednog para uglastih zagrada.

Na primer, polja šahovske table se mogu deklarisati na sledeći način:

```
char sah_tabla[8][8];
```

pri čemu prva dimenzija označava redove, a druga kolone dvodimenzionalne tabele.

Deklarisanje višedimenzionalne tebele u C-u se razlikuje od uobičajene deklaracije kod drugih viših programskih jezika; dok se kod drugih jezika sve dimenzije obično pišu unutar jednog para zagrada, u C-u se svaka dimenzija piše u zasebnom paru uglastih zagrada.

Inicijalizacija npr. dvodimenzionalne tabele se vrši na sledeći način:

```
int mat[3][4] = {
                  { 35, 46, 32, 40 }
                  { 34, 38, 31, 33 }
                  { 39, 37, 41, 36 }
                };
```

a može i ovako:

```
int mat[3][4] = {35,46,32,40,34,38,31,33,39,37,41,36};
```

```
#include <stdio.h>
#define MAX_SIZE 100

int main() {
 int mat[MAX_SIZE][MAX_SIZE];
 int i,j,n;
                .
                .
                .
                .
                                .
                                .
                                .
                for (i=0; i<n; i++) {
                  for (j=0; j<n; j++)
                    printf("%d ", mat[i][j]);

                  printf("\n");
                }
                                .
                                .
                                .
```

```c
#include <stdio.h>
#define MAX_SIZE 100

int main() {
 int mat[MAX_SIZE][MAX_SIZE];
 int i,j,n;

 printf("Unesite n \n");
 scanf("%d", &n);

 for (i=0; i<n; i++) {
   for (j=0; j<n; j++) {
     if (i==j) {
       mat[i][j] = 1;
     }
     else {
       if (i>j)
         mat[i][j] = 3;
       else
         mat[i][j] = 2;
     }
   }
 }
 for (i=0; i<n; i++) {
   for (j=0; j<n; j++)
     printf("%d ", mat[i][j]);

   printf("\n");
 }

 return 0;
} // of main
```

```
Unesite n
6
1 2 2 2 2 2
3 1 2 2 2 2
3 3 1 2 2 2
3 3 3 1 2 2
3 3 3 3 1 2
3 3 3 3 3 1
```

a[0,0]	a[0,1]	a[0,2]	a[0,3]	a[0,4]	a[0,5]
a[1,0]	a[1,1]	a[1,2]	a[1,3]	a[1,4]	a[1,5]
a[2,0]	a[2,1]	a[2,2]	a[2,3]	a[2,4]	a[2,5]
a[3,0]	a[3,1]	a[3,2]	a[3,3]	a[3,4]	a[3,5]
a[4,0]	a[4,1]	a[4,2]	a[4,3]	a[4,4]	a[4,5]
a[5,0]	a[5,1]	a[5,2]	a[5,3]	a[5,4]	a[5,5]

Kriterijum za članove glavne dijagonale:

i == j

Kriterijum za članove sporedne dijagonale:

(i + j)==5

Kriterijum za članove „ispod“ glavne dijagonale:

i > j

Kriterijum za članove „iznad“ glavne dijagonale:

i < j

1	2	2	2	2	2
3	1	2	2	2	2
3	3	1	2	2	2
3	3	3	1	2	2
3	3	3	3	1	2
3	3	3	3	3	1

a[0,0]	a[0,1]	a[0,2]	a[0,3]	a[0,4]	a[0,5]
a[1,0]	**a[1,1]**	a[1,2]	a[1,3]	a[1,4]	a[1,5]
a[2,0]	a[2,1]	**a[2,2]**	a[2,3]	a[2,4]	a[2,5]
a[3,0]	a[3,1]	a[3,2]	**a[3,3]**	a[3,4]	a[3,5]
a[4,0]	a[4,1]	a[4,2]	a[4,3]	**a[4,4]**	a[4,5]
a[5,0]	a[5,1]	a[5,2]	a[5,3]	a[5,4]	**a[5,5]**

Obilazak članova glavne dijagonale:

i = j

```
for (i=0; i<n; i++) {
    for (j=0; j<n; j++){
      if (i==j)
        printf("%d ", mat[i][j]);
    }
}
```

```
1 2 2 2 2 2
3 1 2 2 2 2
3 3 1 2 2 2
3 3 3 1 2 2
3 3 3 3 1 2
3 3 3 3 3 1
```

```
a[0,0] a[0,1]                                  a[0,5]
a[1,0] a[1,1]                          a[1,4]
a[2,0] a[2,1]                  a[2,3]
a[3,0] a[3,1] a[3,2]
a[4,0] a[4,1]
a[5,0] a[5,1]
```

Kriterijum za članove sporedne dijagonale:

(i + j)==(6-1)

```
for (i=0; i<n; i++) {
    for (j=0; j<n; j++){
      if ( (i+j)==(max_dim-1) )
        printf("%d ", mat[i][j]);
    }
}
```

```
1 2 2 2 2 2
3 1 2 2 2 2
3 3 1 2 2 2
3 3 3 1 2 2
3 3 3 3 1 2
3 3 3 3 3 1
```

```
a[0,0] a[0,1] a[0,2]
a[1,0] a[1,1] a[1,2]
a[2,0] a[2,1] a[2,2]
a[3,0] a[3,1] a[3,2]
a[4,0] a[4,1] a[4,2] a[4,3]
a[5,0] a[5,1] a[5,2] a[5,3] a[5,4]
```

Kriterijum za članove „ispod“ glavne dijagonale:

i > j

```
for (i=0; i<n; i++) {
    for (j=0; j<n; j++){
      if (i>j)
        printf("%d ", mat[i][j]);
    }
}
```

```
1 2 2 2 2 2
3 1 2 2 2 2
3 3 1 2 2 2
3 3 3 1 2 2
3 3 3 3 1 2
3 3 3 3 3 1
```

```
a[0,0] a[0,1] a[0,2] a[0,3] a[0,4] a[0,5]
a[1,0] a[1,1] a[1,2] a[1,3] a[1,4] a[1,5]
a[2,0] a[2,1] a[2,2] a[2,3] a[2,4] a[2,5]
a[3,0] a[3,1] a[3,2] a[3,3] a[3,4] a[3,5]
a[4,0] a[4,1] a[4,2] a[4,3] a[4,4] a[4,5]
a[5,0] a[5,1] a[5,2] a[5,3] a[5,4] a[5,5]
```

Kriterijum za članove „iznad“ glavne dijagonale:

i < j

```
for (i=0; i<n; i++) {
    for (j=0; j<n; j++){
      if (i<j)
        printf("%d ", mat[i][j]);
    }
}
```

```
1 2 2 2 2 2
3 1 2 2 2 2
3 3 1 2 2 2
3 3 3 1 2 2
3 3 3 3 1 2
3 3 3 3 3 1
```