

PiPI

RJEŠENJA VJEŽBI ZA BLITZ 04

Grupa 08,
Z. Šimić, 2007.

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:
<code>int a[5] = {4}, i;</code>	
<code>printf("%d\n", a[4]);</code>	0
<code>printf("%d\n", a[5]);</code>	<i>indeksirano izvan polja: ispis nepredvidiv</i>
<code>a[5]=1; a[-1]=1;</code>	<i>1 se upisuje na krivo mjesto u memoriji - opasno!</i>
<code>for (i=0; i<5; i++) printf("%d ", a[i]);</code>	4 0 0 0 0
<code>for (i=0; i<5; i++) { a[i]=(4-i)*10; printf(" %d", a[i]); }</code>	40 30 20 10 0
<code>for (i=0; i<5; i++) { a[i]/=10; printf(" %d %d",a[i],a[i/2]); }</code>	4 4 3 4 2 3 1 3 0 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]);
```

0a0

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

8 0

```
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]);
```

2 7

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

0 5

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

6 1

Definiranje, inicijaliziranje i indeksiranje polja

	Ispravno ili ne:
<code>char z[4] = {'a', 'b', 99, 100};</code>	Ispravno
<code>unsigned int i[4][0][1][2][3];</code>	<i>0 nedopuštena dimenzija</i>
<code>int i[6]={1, 2, 3, 4, 5, 6};</code>	Ispravno
<code>char c[] = {};</code>	<i>Kriva sintaksa.</i>
<code>float f[4]={5, 3., 1.};</code>	Ispravno
<code>int i_1[1][2]={1, 2, 3, 4};</code>	<i>Previše inicijal. elemenata.</i>
<code>float _f[4]={2.f, 4.f, 6.f, 8.f};</code>	Ispravno
<code>int [2,2] = {1};</code>	<i>Kriva sintaksa.</i>
<code>int i=0, e;</code>	
<code>float a[4]={0}, b[4], c[e]={i};</code>	
<code>for(i=1;i<=4;i++) b[i-1]=0;</code>	- polja a i b su ispunjena 0-ma, polje c ima sintaksne pogreške

Koliko polje zauzima memorije?

Odgovor:

```
char z[7] = {'a', 'b'};
```

7 bytes

```
long int i[10];
```

40 bytes

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

20 bytes

```
char c[5+1];
```

6 bytes

```
float f[50][2];
```

400 bytes

```
long a[10][5][2];
```

400 bytes

```
double d[20][50];
```

8000 bytes

```
short s[10][10] = {1, 2};
```

200 bytes

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);
```

Ispis/učitavanje:

-2 4 -3

```
for (i=4; i>=1; i--)
    printf("%d", a[i]=i*i);
printf("a[%d]=%d", i, a[i]);
```

16941a[0]==1

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

Učitava jednu
vrijednost; za a[4]

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==45

x==45

```
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 == 8

c == 16

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 na glavnoj
dijagonali na 1.
z == 9

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

Nakon petlji:
Postavlja elemente polja ispod
sporedne dijagonale na 1.
s == 15

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=='N' i==0 j==1 d==2.0

Aritmetika pokazivača

Dozvoljeno je i smisleno

- Zbrajati i oduzimati cijeli broj od pokazivača
- Za dva pokazivača na isto polje:
 - oduzimati ih
 - uspoređivati ih (\leq i \geq)

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

<code>short n=1, m=2, k;</code>	
<code>short *pi = &n, *pj = &m, *pp;</code>	
<code>pp = &k;</code>	
<code>*pp = *pi;</code>	
<code>*pi = m;</code>	
<code>m = *pp;</code>	
<code>*pp = k * m + n;</code>	
<code>pp = pi;</code>	
<code>printf("%d %d %d\n", n, m, k);</code>	2 1 3
<code>printf("%d %d %d", *pi, *pj, *pp);</code>	2 1 2
<code>printf("\n%d ", *pi + *pj);</code>	3
<code>printf("\n%d ", sizeof(n));</code>	2
<code>printf("\n%d \n", sizeof(pi));</code>	4
<code>n = sizeof(n) + sizeof(pj);</code>	
<code>printf("\n%d %d", n, n + *pi);</code>	6 12

Ispis:

Pokazivači

<pre>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);</pre>	<p><u>Ispis:</u></p> <p>4.0 3.0 12.0 4.0 3.0 3.0 6.0 8 4 12.0 15.0</p>
--	--

Pokazivači – radi, ali nema smisla:

Ispis:

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

Neka je početna adresa varijable **x**
4377768 (0042CCA8₁₆)

```
pj = (int)x;  
printf("%lu %p\n", pj, pj);           5 00000005  
*pa = 100 * (int>(&x));  
printf("%p %p\n", pa, &x);           0042CCA8 0042CCA8  
printf("%lu %lu\n", pa, &x);          4377768 4377768  
printf("%f %f\n", *pa, x);            437776800.0 437776800.0  
printf("%lu", 10 * (int>(&x)));         43777680
```

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

Pokazivači – aritmetika

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

`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);` 1000016 1.100000
`pb += 2;`
`printf("%lu %f\n", pb, *pb);` 1000024 3.300000
`printf("%lu %f\n", ar+2, *(ar+2));` 1000024 3.300000
`printf("%lu %f\n", &ar[2], ar[2]);` 1000024 3.300000
`printf("%d\n", pb-pa);` 2
`printf("%d\n", (int)pb-(int)pa);` 8
`printf("%lu %f\n", pb-1, *(pb-1));` 1000020 2.200000
`pz += 4;`
`printf("%d\n", pz-az);` 4
`printf("%d\n", (int)pz-(int)az);` 4
`pz = (char *) ((int *) az +4);`
`printf("%d\n", pz-az);` 16
`printf("%d\n", (int)pz-(int)az);` 16

Ispis: