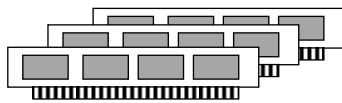


# Pokazivači (*pointers*)

## Organizacija radne memorije (*RAM*)



Memorija računala zapravo je kontinuirani niz bajtova: svaki bajt ima svoj "redni broj", tj. adresu. Memorija od 512 MB može se promatrati na sljedeći način:

0	00110001
1	11010010
...	...
82560	11000001
82561	00001101
82562	00111000
82563	11010101
82564	00000000
82565	10001000
82566	10101011
82567	11101000
82568	11111111
82569	01000010
...	...
536870910	00110001
536870911	00000111

## Definicija "običnih" varijabli

Definicijom varijabli rezervira se prostor u memoriji na nekim adresama, npr.:

...	...		
82560	00000000	}	<b>a</b>
82561	00000010		
82562	11111111	}	<b>b</b>
82563	11111111		
82564	00000000	}	<b>c</b>
82565	00000000		
82566	00000000		
82567	00000111	}	<b>d</b>
82568	00001010		
...	...		

```
short a, b;  
int c;  
char d;  
a = 2;  
b = -1;  
c = 7;  
d = '\n';
```

Za varijablu kažemo da je "na adresi x" ukoliko je prvi bajt sadržaja varijable pohranjen na adresi x.

Primjer: adresa varijable **c** je **82564**.

3

## Adresa ili pokazivač?

Za adresu (*address*) također se koristi pojam pokazivač (*pointer*) zato jer adresa "pokazuje" na neki objekt u memoriji.

...	...		
82560	00000000	}	<b>a</b>
82561	00000010		
82562	11111111	}	<b>b</b>
82563	11111111		
...	...		

```
short a, b;  
a = 2;  
b = -1;
```

**82562** "pokazuje" na jedan objekt tipa **short** (varijablu **b**)

4

## Kako saznati adresu varijable

Ako je **x** varijabla, tada je **&x** njena adresa u memoriji:

...	...		
82560	00000000	} a	short a = 2, b = -1;
82561	00000010		printf("%p %p", &a, &b);
82562	11111111	} b	Ispis:
82563	11111111		82560 82562
...	...		

Kojeg su "tipa" ove dvije adrese? Iako "izgleda" kao `int`, adresa u općem slučaju nije `int` (niti `short`, niti `long`). Zato adresu **nije moguće** pohraniti u varijablu tipa `int`.

```
int a, p;  
p = &a;
```

5

## Kamo pohraniti adresu (pokazivač)

Za pohranu adrese varijable koja je tipa `short`, mora se koristiti varijabla posebnog tipa (pokazivač na `short`):

...	...		
82560	00000000	} a	short a = 2, b = -1;
82561	00000010		short *p1;
82562	11111111	} b	p1 = &b;
82563	11111111		printf("%p", p1);
82564	00000000	} p1	Ispis:
82565	00000001		82562
82566	01000010		
82567	10000010		
...	...		

`p1` je varijabla u koju je moguće smjestiti adresu objekta koji je tipa `short` (tj. pokazivač na objekt tipa `short`). Radi pojednostavljenja, kažemo "`p1` je pokazivač na `short`".

6

## Definiranje pokazivača

---

Pokazivači se definiraju na sličan način kao "obične" varijable. \* ispred imena varijable znači da se ta varijabla koristi za pohranu pokazivača na objekt odgovarajućeg tipa.

```
short a;    /* a služi za pohranu vrijednosti tipa short */
int b;      /* b služi za pohranu vrijednosti tipa int */
short *p1; /* p1 je pokazivač na short */
int *p2;    /* p2 je pokazivač na int */
```

Dopušteno je u istoj naredbi definirati i "obične" varijable i pokazivače:

```
short a, *p1;
int b = 5, *p2;
int *p3 = &b;
```

7

## Tipovi pokazivača

---

Pokazivače na objekte jednog tipa **nije dopušteno** koristiti kao pokazivače na objekte nekog drugog tipa.

```
short a = 2, b = -1;
int c = 7;
short *p1;
int *p2;
p1 = &a;    /* O.K. */
p1 = &b;    /* O.K. */
p2 = &c;    /* O.K. */
p2 = &b;    /* ne valja! */
```

8

## Pristupanje objektu na kojeg pokazuje pokazivač

```
int a = 15;
```

82560 15 ← Koji je sadržaj a?

```
int *p;
```

82564 ? ← Na što pokazuje p?

```
p = &a;
```

82564 82560 ← Na što pokazuje p?

**\*p** → rezultat je `int` vrijednost na koju pokazuje p (tj. vrijednost koja se nalazi na adresi 82560)

```
printf("%d", *p);
```

→ 15

9

## Izmjena vrijednosti objekta na kojeg pokazuje pokazivač

```
short a;
```

82560 ? ← Koji je sadržaj a?

```
short *p = &a;
```

82562 82560 ← Na što pokazuje p?

**\*p = 16;** kao vrijednost objekta na kojeg pokazuje p, tj. na adresu 82560, upiši vrijednost 16.

82560 16 ← Koji je sadržaj a?

10

Primjer:

pretpostavka o adresama varijabli - a: 8560, b: 8564, c: 8568

	a	b	c	ap	bp	cp
<code>int a=1, b=2, c=3;</code>	1	2	3			
<code>int *ap, *bp, *cp;</code>	1	2	3	?	?	?
<b>*ap = 100; ap</b> pokazuje na "tko zna što". Naredbom se zapisuje vrijednost 100 na "tko zna koju adresu". <b>Ne činiti to!</b>						
<code>ap = &amp;a;</code>	1	2	3	8560	?	?
<code>bp = ap;</code>	1	2	3	8560	8560	?
<code>*bp = 200;</code>	200	2	3	8560	8560	?
<code>c = *ap;</code>	200	2	200	8560	8560	?
<code>cp = &amp;c;</code>	200	2	200	8560	8560	8568
<code>a = 300;</code>	300	2	200	8560	8560	8568
<code>*cp = *ap + 10;</code>	300	2	310	8560	8560	8568

11

Primjer:

pretpostavka o adresama varijabli - x: 8560, y: 8564

```
int x = 5, y = 10;
int *px, *py;
px = &x; /* px: 8560 */
py = &y; /* py: 8564 */
```

Što će se ispisati?

a) `*px = *py;`  
`printf("%d %d %p %p", x, y, px, py);`  
→ 10 10 8560 8564

b) `px = py;`  
`printf("%d %d %p %p", x, y, px, py);`  
→ 5 10 8564 8564

12

## Aritmetika s pokazivačima

Iako "izgleda" kao `int`, pokazivač nije `int` (niti `short`, niti `long`). Što je s aritmetičkim operacijama s pokazivačima? Operacije koje "imaju smisla" su:

- pokazivaču pribrojiti cijeli broj
- od pokazivača oduzeti cijeli broj

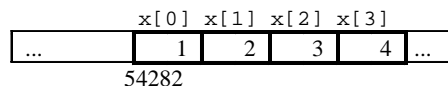
```
char c, *cp = &c; /* neka je cp: 38000 */
short s, *sp = &s; /* neka je sp: 48000 */
int i, *ip = &i; /* neka je ip: 58000 */
double d, *dp = &d; /* neka je dp: 68000 */

cp + 1 → 38001      cp - 1000 → 37000
sp + 1 → 48002      sp - 1000 → 46000
ip + 1 → 58004      ip - 1000 → 54000
dp + 1 → 68008      dp - 1000 → 60000
```

13

## Polja i pokazivači

```
#include <stdio.h>
int main () {
    int x[4] = {1, 2, 3, 4};
    int *p = &x[0]; /* 54282 */
    printf("%d %d %d %d", *p, *(p+1), *(p+2), *(p+3));
    return 0;
}
```



Ispis na zaslonu:  
1 2 3 4

Što bi se dogodilo da se npr. prije `return 0;` obave naredbe:

```
*(p+4) = 1000;
*(p-1) = 1000;
```

14

## Primjer: inicijalizirati članove polja pri čemu članovima treba pristupiti preko pokazivača

```
#include <stdio.h>
int main () {
    float x[10], *p;
    int i;
    for (i = 0, p = &x[0]; i < 10; i++)
        *(p+i) = 1.0;
    /* kontrolni ispis */
    for (i = 0; i < 10; i++)
        printf ("%f\n", *(p+i)); ili *(p++)
    return 0;
}
```

15

## Korištenje varijable tipa polje umjesto pokazivača na prvi element polja

Ako je **x** varijabla definirana kao **jednodimenzionalno** polje, tada se adresa prvog člana polja može dobiti na dva načina:

`&x[0]`

`x`

U prethodnom primjeru, umjesto `p = &x[0];` može `p = x;`

**Napomena:** ako je **x** varijabla tipa polje s dvije ili više dimenzija, tada se **x** ne može koristiti kao adresa prvog člana polja **x**

Ako je **x** varijabla definirana kao **dvodimenzionalno** polje, tada se adresa prvog člana polja **x** može dobiti na dva načina:

`&x[0][0]`

`x[0]`

Ako je **x** varijabla definirana kao **trodimenzionalno** polje, tada se adresa prvog člana polja **x** može dobiti na dva načina:

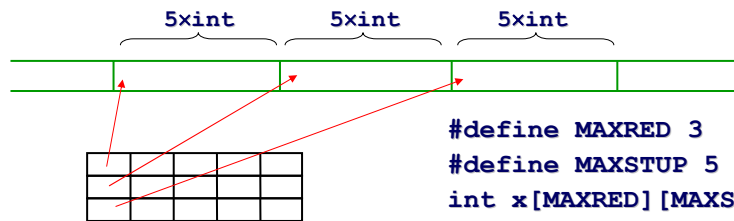
`&x[0][0][0]`

`x[0][0]`

16

## Dvodimenzionalna polja i pokazivači

- Dvodimenzionalno polje: redak za retkom



```
#define MAXRED 3
#define MAXSTUP 5
int x[MAXRED][MAXSTUP]
3 susjedna jednodimenzionalna
polja veličine 5 x int
```

```
int *p = &x[0][0];
*(p + 0*MAXSTUP + 0) → vrijednost člana x[0][0]
*(p + 0*MAXSTUP + 1) → vrijednost člana x[0][1]
*(p + 1*MAXSTUP + 0) → vrijednost člana x[1][0]
*(p + 2*MAXSTUP + 0) → vrijednost člana x[2][0]
*(p + 2*MAXSTUP + 3) → vrijednost člana x[2][3]
```

17

## Primjer određivanja najvećeg člana polja u svakom retku - pomoću pokazivača (1)

- Pročitati vrijednosti za broj redaka  $mr \leq 10$  i broj stupaca  $ms \leq 5$ . Pročitati vrijednosti članova dvodimenzionalnog realnog polja od  $mr$  redaka i  $ms$  stupaca. Ispisati vrijednost najvećeg člana u svakom retku.

18

## Primjer određivanja najvećeg člana polja u svakom retku - pomoću pokazivača (2)

---

```
#include <stdio.h>
#define MAXRED 10
#define MAXSTUP 5

int main() {
    int mr, ms, i, j;
    float najveći, a[MAXRED][MAXSTUP];
    float *p = &a[0][0];

    /* učitavati mr i ms dok ne budu ispravni
    */
    ...
}
```

19

## Primjer određivanja najvećeg člana polja u svakom retku - pomoću pokazivača (3)

---

```
/* učitaj polje a od mr redaka i ms stupaca */
for (i = 0; i < mr; i++) {
    for (j = 0; j < ms; j++) {
        scanf("%f", p + i*MAXSTUP + j);
    }
}
/* Ispiši polje a */
for (i = 0; i < mr; i++) {
    for (j = 0; j < ms; j++) {
        printf("%f ", *(p + i*MAXSTUP + j));
    }
    printf ("\n");    /* skok u novi red */
}
}
```

20

## Primjer određivanja najvećeg člana polja u svakom retku - pomoću pokazivača (4)

---

```
/*Ispis najvećih članova u retcima */
printf("Najveci clanovi polja po retcima:\n");
/* Petlja po svim retcima */
for (i = 0; i < mr; i++) {
    najveci = *(p + i*MAXSTUP); /* prvi clan retka i */
    for (j = 1; j < ms; j++)
        if (*(p + i*MAXSTUP + j) > najveci)
            /* clan retka i stupca j */
            najveci = *(p + i*MAXSTUP + j);
    /*Ispis vrijednosti nadjenog najveceg clana*/
    printf("%f\n", najveci);
}
return 0;
}
```

21