

Uvod u pokazivače u C-u

Pokazivači u C-u

- Pokazivači (*pointers*) su osnovni dio C jezika
- Snažan alat koji se često koristi jer:
 - se jedino sa pokazivačima mogu izvesti neki računi i programske tehnike
 - se dobiva kompaktni i efikasni kod
- Ali *nepravilno korištenje =>*
 - *velika opasnost od grešaka u programu!*

"s velikom moći dolazi velika odgovornost"

Memorija računala

- Jednostavan primjer:
(32bit)
- Adrese memorijskih lokacija
od $00000000_{(16)}$ do $FFFFFFFF_{(16)}$
- Na svakoj adresi pohranjen
1 byte, dakle broj od $00_{(16)}$ do $FF_{(16)}$
(0 do 255)

00000000	00
00000001	00
...	
0001C702	1A
0001C703	00
0001C704	34
0001C705	00
0001C706	00
0001C707	00
0001C708	00
...	
FFFFFFFFFE	00
FFFFFFFFF	00

Definicija varijabli

- definicijom varijable rezerviramo prostor u memoriji

```
int a;
```

```
char b;
```

```
short c;
```

```
a = 1;
```

```
b = 2;
```

```
c = 3;
```

a je "na adresi"
0001C702₍₁₆₎

00000000	00
00000001	00
...	
0001C702	01
0001C703	00
0001C704	00
0001C705	00
0001C706	02
0001C707	03
0001C708	00
...	
FFFFFFFFE	00
FFFFFFFFF	00

- za varijablu **a** su rezervirana 4 bajta počevši od adrese 0001C702

Operator &

Kako saznati adresu od **x** ?

```
int x=0 , y=10 ;
```

&x → 020BC40A

&y → 020BC40E

x	020BC40A	00
	020BC40B	00
	020BC40C	00
	020BC40D	00
y	020BC40E	0A
	020BC40F	00
	020BC410	00
	020BC411	00

& čitati "adresa od"

Operator &

Primjer:

```
int a;  
a = 1;  
printf("a = %i, ", a);  
printf("adresa od a %x\n", &a);
```

a {

0001C702	01
0001C703	00
0001C704	00
0001C705	00
0001C706	02
0001C707	03
0001C708	00

Ispis: **a = 1, adresa od a je 1C702**

Pokazivač (pointer)

Kamo pohraniti adresu? ... u pokazivač!

- Pokazivač je varijabla koja sadrži adresu

```
int a;
```

```
int *pa; /* pokazivac na int */
```

```
pa = &a; /* uzmi adresu od a i pohrani  
je u pokazivac pa */
```

- Definira se kao i "obična" varijabla, s time da se prije imena stavi ' * '

Definiranje pokazivača (primjeri)

```
short a, *pa;  
a = 5;  
pa = &a;  
printf("%x ", a);  
printf("%x ", &a);  
printf("%x\n", pa);
```

a	0001C702	05
	0001C703	00
pa	0001C704	02
	0001C705	C7
	0001C706	01
	0001C707	00
	0001C708	00

Ispis: 5 1C702 1C702

- Kada pokazivač sadrži adresu neke varijable kaže se da "pokazuje" na tu varijablu

Definiranje pokazivača

```
int a;  
int *pa; /* pokazivac na int */  
char *pb; /* pokazivac na char */  
pa = &a; /* ok */  
pb = &a; /* nije dopusteno! */
```

moraju biti istog tipa



Operator *

(dereferenciranje pokazivača)

- imamo pokazivač koji pokazuje na varijablu a:

```
int a=25, *pa;
```

```
pa=&a;      /* sada pa pokazuje na a */
```

- s operatorom * možemo pristupiti sadržaju memorijske lokacije koja je zapisana u pokazivaču

```
printf("%i", *pa);
```

Ispis: 25

pristupamo sadržaju od a



Operator *

(dereferenciranje pokazivača)

- imamo pokazivač koji pokazuje na varijablu a:

```
int a=25, *pa;          DEFINICIJA  
pa=&a;                 /* pa pokazuje na a */
```

- s operatorom * možemo pristupiti sadržaju memorijske lokacije koja je zapisana u pokazivaču

```
printf("%i", *pa);    PRISTUP SADRŽAJU  
                     (dereferencing)
```

Ispis: **25** pristupamo sadržaju od a

Operator *

(primjeri)

Ispis:

```
char x=1,y=2;
```

```
char *p;
```

```
p=&x;
```

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

1

```
p=&y;
```

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

2

```
y=3;
```

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

3

Operator *

(primjeri)

```
char x=5, *p;
```

```
p=&x; /* p pokazuje na x */
```

- ako pokazivač **p** pokazuje na **x** onda je jednako pisati `printf (*p) ;` ili `printf (x) ;`

```
printf (*p) ;
```

```
printf (x) ;
```

```
*p=3;
```

```
printf (*p) ;
```

```
printf (x) ;
```

Ispis:

5
5

3
3

Operator *

(primjeri)

```
char x=5, *p;
```

```
p=&x; /* p pokazuje na x */
```

- ako pokazivač **p** pokazuje na **x** onda je jednako pisati `printf (*p)` ; ili `printf (x)` ;

Ispis:

```
printf (*p);      5  
printf (x);      5
```

odnose se na sadržaj iste memorijske adrese

Primjeri

<i>adresa od a je 1B60</i>	a	b	pa	pb
<code>int a=1,b=2;</code>	1	2		
<code>int *pa, *pb;</code>				
<code>pa=&a;</code>				
<code>pb=pa;</code>				
<code>*pb=15;</code>				
<code>b=*pa+5;</code>				

Primjeri

<i>adresa od a je 1B60</i>	a	b	pa	pb
int a=1,b=2;	1	2		
int *pa, *pb;	1	2	?	?
pa=&a;				
pb=pa;				
*pb=15;				
b=*pa+5;				

Primjeri

<i>adresa od a je 1B60</i>	a	b	pa	pb
int a=1,b=2;	1	2		
int *pa, *pb;	1	2	?	?
pa=&a;	1	2	1B60	?
pb=pa;				
*pb=15;				
b=*pa+5;				

Primjeri

<i>adresa od a je 1B60</i>	a	b	pa	pb
int a=1,b=2;	1	2		
int *pa, *pb;	1	2	?	?
pa=&a;	1	2	1B60	?
pb=pa;	1	2	1B60	1B60
*pb=15;				
b=*pa+5;				

Primjeri

<i>adresa od a je 1B60</i>	a	b	pa	pb
int a=1,b=2;	1	2		
int *pa, *pb;	1	2	?	?
pa=&a;	1	2	1B60	?
pb=pa;	1	2	1B60	1B60
*pb=15;	15	2	1B60	1B60
b=*pa+5;				

Primjeri

<i>adresa od a je 1B60</i>	a	b	pa	pb
int a=1,b=2;	1	2		
int *pa, *pb;	1	2	?	?
pa=&a;	1	2	1B60	?
pb=pa;	1	2	1B60	1B60
*pb=15;	15	2	1B60	1B60
b=*pa+5;	15	20	1B60	1B60

Novi primjer (pažljivo!)

<i>adresa od a je 1B60</i>	a	b	pa	pb
int a=1,b=2;	1	2		
int *pa, *pb;	1	2	?	?
pa=&a;	1	2	1B60	?
pb=pa;	1	2	1B60	1B60
pb=pa+1;				

?

Novi primjer (pažljivo!)

<i>adresa od a je 1B60</i>	a	b	pa	pb
int a=1,b=2;	1	2		
int *pa, *pb;	1	2	?	?
pa=&a;	1	2	1B60	?
b=a+1;	1	2	1B60	1B60
pb=pa+1;	1	2	1B60	1B64

ne ~~1B61~~ vec 1B64 !



Aritmetika pokazivača

```
char    *pc = 0x00001000;  
short   *ps = 0x00002000;  
int     *pi = 0x00003000;  
float   *pf = 0x00004000;
```

```
printf("%x ", pc+1);  
printf("%x ", ps+1);  
printf("%x ", pi+1);  
printf("%x ", pf+1);
```

Ispis:

1001

2002

3004

4004

Pomak je ovisan o tipu pokazivača

Polja i pokazivači

...	a[0]	a[1]	a[2]	a[3]	...
	10	4	2	25	

```
int x, a[4]={10,4,2,25};
```

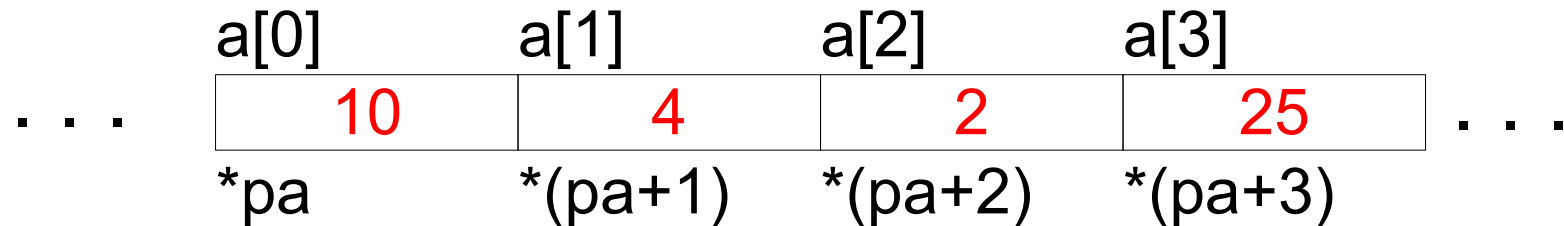
```
int *pa;
```

```
pa = &a[0];    /* pa pokazuje na a[0] */
```

```
x = *pa;      /* x = sadržaj od a[0] */
```

- Što sadrži `x = *(pa+2)` ?

Polja i pokazivači



```
int x, a[4]={10,4,2,25};
```

```
int *pa;
```

```
pa = &a[0]; /* pa pokazuje na a[0] */
```

```
x = *(pa+3); /* isto kao x = a[3] */
```

```
printf("%i",x);
```

Ispis: 25

Polja i pokazivači

	a[0]	a[1]	a[2]	a[3]	
	*a	*(a+1)	*(a+2)	*(a+3)	
...	10	4	2	25	...
	*pa	*(pa+1)	*(pa+2)	*(pa+3)	

Primjer:


```
int a[4]={10,4,2,25};  
int *pa=a;          /* a <=> &a[0] */  
for(int i=0;i<4;i++)  
    printf("%i ",*(pa+i));
```

Ispis: 10 4 2 25

Najčešće greške

1) Sintaksa

```
int x, y = 5;    /* y je na adresi 1BA0 */  
int *p = &y;  
x = p;          /* možda greska? */  
                /* ispravno je x = *p; */
```



x sada sadrži 1BA0

- C prevodilac ne radi probleme i stavlja u **x** adresu sadržanu u **p** kao da je broj.
- Ovakve greške su teške za uočiti

Najčešće greške

2) Inicijalizacija

VAŽNO: kada se definira pokazivač on ne pokazuje nigdje, mora se inicijalizirati prije korištenja!

```
int *p;  
*p = 100;
```

Greška! Može izazvati "crash" programa!

```
int *p;  
int x;  
p = &x;  
*p = 100;
```

Ok!

- Također teška za uočiti. Prevodilac ne javlja grešku!

Najčešće greške

3) Kontrola prije pristupa sadržaju

Kada sami ne inicijaliziramo pokazivač, prije pristupa sadržaju valja uvijek kontrolirati ispravnost adrese.

Opasno:

```
int *p;  
p = fun(); /* neka funkcija koja vraca adresu */  
*p = 100;
```

Bolje:

```
int *p;  
p = fun(); /* neka funkcija koja vraca adresu */  
if (p!=NULL) *p = 100;
```

Sažetak

- Pokazivač je varijabla koja sadrži adresu druge varijable
- Pokazivač može biti bilo kog tipa, ali adrese koje sadrži moraju biti od varijabli istog tipa
- Operator **&** daje "*adresu od varijable*"
- Operator ***** daje "*sadržaj lokacije na koju pokazivač pokazuje*"
- Pokazivač se definira se kao i obična varijabla ali se ispred imena stavlja ***** (Na primjer ***int *p;***)
- Obratiti pažnju da je pokazivač inicijaliziran prije *dereferenciranja* (pristupanje sadržaju).

Zaključci

- Pokazivači su koristan i snažan alat koji pruža veliku fleksibilnost u programiranju
- Opasni su ako se nepravilno koriste!
- Snaga i korist pokazivača dolazi do izražaja u nekim važnih tehnikama programiranja i dijelovima C jezika koje nismo vidjeli u ovoj uvodnoj lekciji:

(Dinamičko alociranje memorije, kompleksne strukture podataka kao npr. vezane liste, stabla i višedimenzionalna polja promjenljivih veličina, prijenos argumenata funkcija, kružni *buffer*-i u memoriji, pokazivači na funkcije itd.)

Kristijan Lenac
klenac@gmail.com

Polja i pokazivači

- Što radi ovaj program?

```
char *grad="akejiR", *p=grad;  
while (*p++!=0);  
while (p--!=grad) printf("%c", *p);
```

Ispis ?

Polja i pokazivači

- Što radi ovaj program?

```
char *grad="akejiR", *p=grad;  
while (*p++!=0);  
while (p--!=grad) printf("%c",*p);
```

Ispis: **Rijeka**