Pokazivači, algoritmi

Tomo Sjekavica

Ožujak 2012.

Creative Commons

- **slobodno smijete:**
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **remiksirati** — prerađivati djelo
- **pod slijedećim uvjetima:**
  - **imenovanje**. Morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno**. Ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima**. Ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela. Najbolji način da to učinite je linkom na ovu internetsku stranicu.

Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.

Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

Tekst licence preuzet je s http://creativecommons.org/.

Pokazivači

- **pokazivači** su varijable koje sadrže **adresu neke druge varijable**

```
int varijabla;
```

```
int *pokazivac;
```

deklaracija obične varijable

deklaracija pokazivača

- pokazivač se označava sa znakom **zvjezdice *** ispred naziva pokazivača
- pokazivači se najčešće koriste kod predaje polja funkciji kao ulazni parametar, kod korištenja raznih struktura, kod dinamičke alokacije memorije, te kod rada sa datotekama

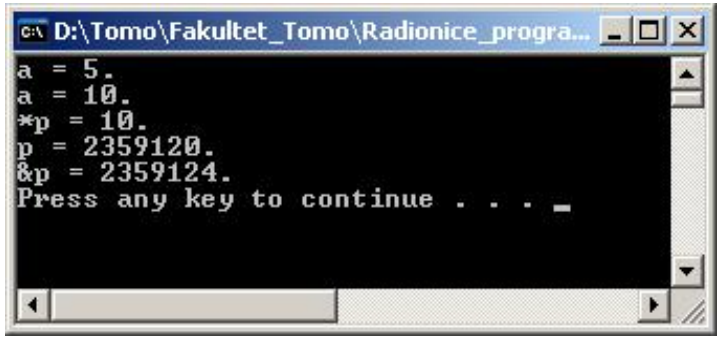
Tipovi pokazivača

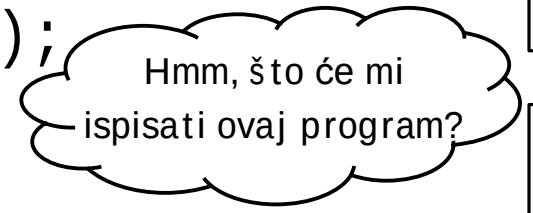
- vrijednost pokazivača će uvijek biti neki cijeli broj bez obzira na koji tip varijable pokazuje jer se u njega sprema adresa te varijable
- pokazivači mogu pokazivati samo na varijable koje su **istog tipa** kao i tip podatka s kojim je deklariran i pokazivač

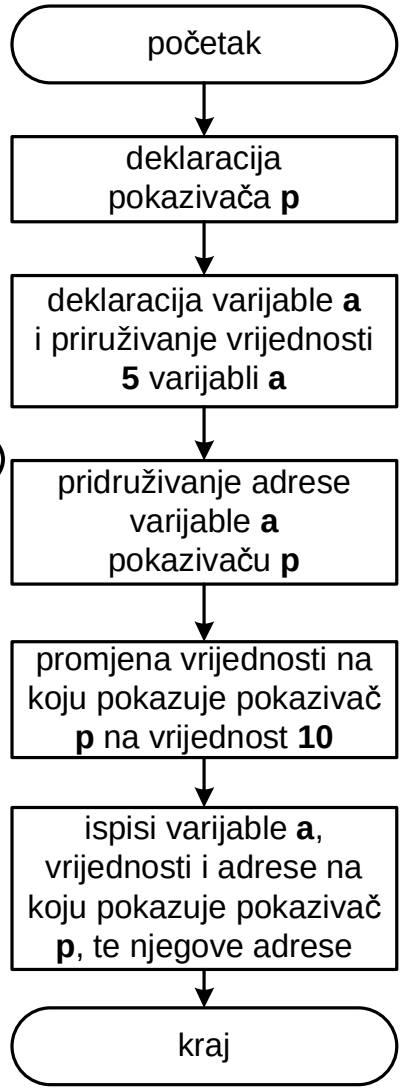
`char *p;`	→ može pokazivati samo na char varijablu →	`char a;`
`int *p;`	→ može pokazivati samo na int varijablu →	`int a;`
`float *p;`	→ može pokazivati samo na float varijablu →	`float a;`
`double *p;`	→ može pokazivati samo na double varijablu →	`double a;`

Primjer s pokazivačima

```
#include <stdio.h>
int main(){
    int *p;
    int a = 5;
    printf("a = %d.\n", a);
    p = &a;
    *p = 10;
    printf("a = %d.\n", a);
    printf("*p = %d.\n", *p);
    printf("p = %d.\n", p);
    printf("&p = %d.\n", &p);
    system("pause");
    return 0;
}
```

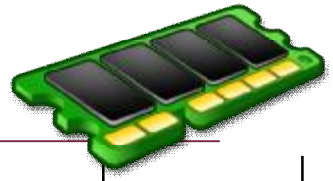






pokazivaci01.c

Anatomija C programa



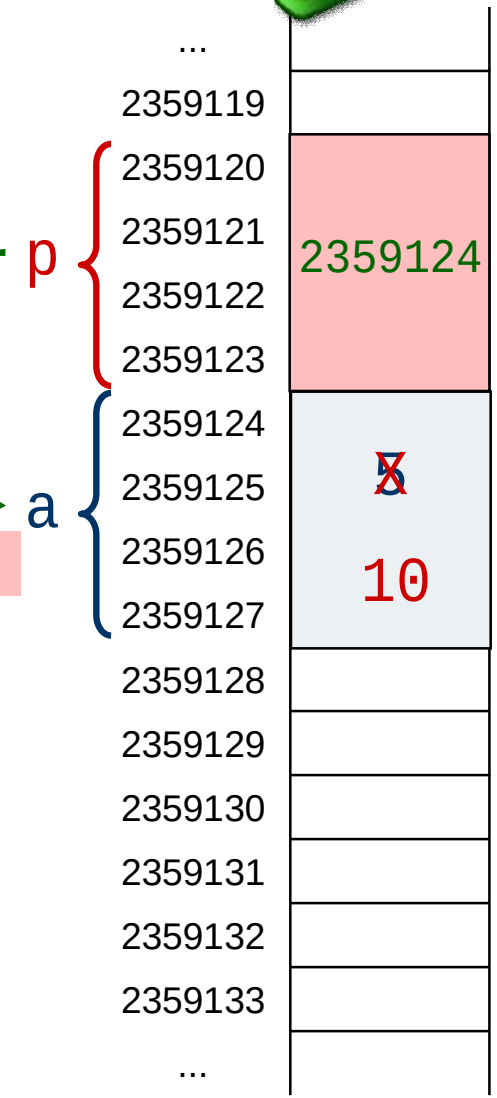
```
#include <stdio.h>
int main(){
    int *p;
    int a = 5;
    printf("a = %d.\n", a);
    p = &a;
    *p = 10;
    printf("a = %d.\n", a);
    printf("*p = %d.\n", *p);
    printf("p = %d.\n", p);
    printf("&p = %d.\n", &p);
    system("pause");
    return 0;
}
```

deklaracija pokazivača koji će pokazivati na cjelobrojnu vrijednost

deklaracija cjelobrojne varijable a i pridruživanje vrijednosti varijabli a

postavljanje da pokazivač pokazuje na varijablu a

mijenjanje vrijednosti varijable a pomoću pokazivača

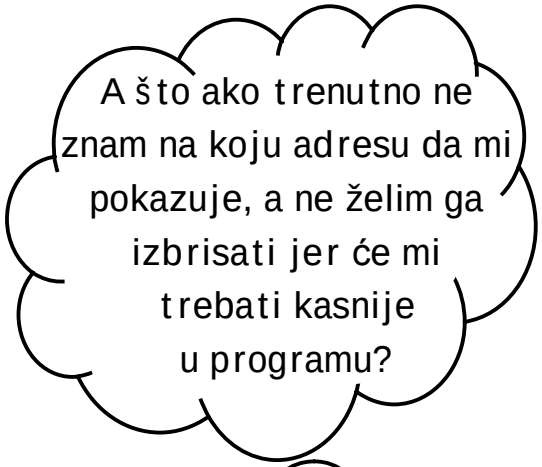


Inicijalizacija pokazivača

- prije korištenja pokazivača **obavezno** je inicijalizirati pokazivač, tj. postaviti da pokazuje na neku adresu

```
int *p;
int a = 5;
p = &a;
```

nakon deklaracije pokazivača i varijable potrebno je pokazivač inicijalizirati



- u slučaju da se ne zna adresu na koju treba pokazivati pokazivač treba ga incijalizirati sa **NULL**

```
int *p;
p = NULL;
```

Operator dereferenciranja *

- nakon inicijalizacije pokazivača vrijednosti te varijable se može pristupiti preko pokazivača

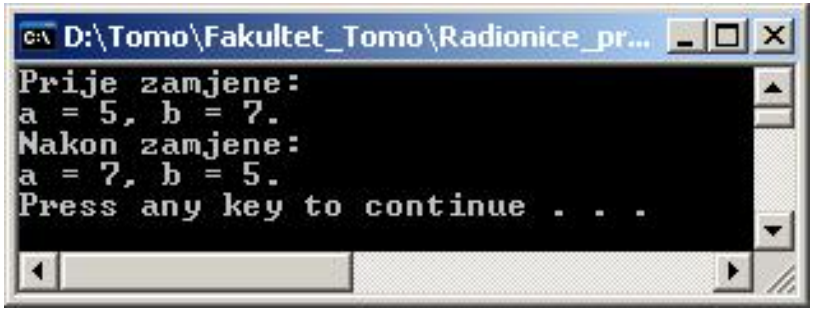
```
int *p;
int a = 5;
p = &a;
printf("%d %d", a, *p);
*p = 10;
printf("%d %d", a, *p);
```

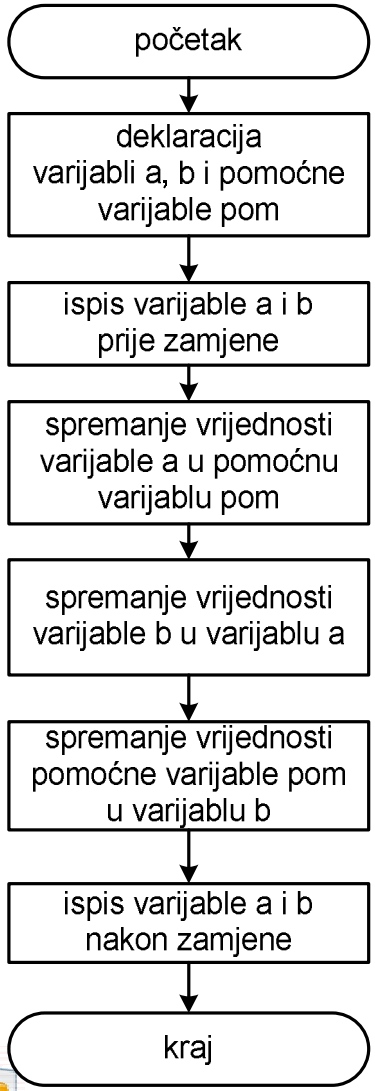
→ 5 5

→ 10 10

Zamjena vrijednosti varijabli v0.1

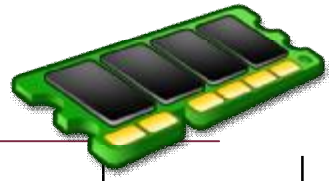
```
#include <stdio.h>
int main(){
    int a = 5;
    int b = 7;
    int pom;
    printf("Prije zamjene:\n");
    printf("a = %d, b= %.d\n", a, b);
    pom = a;
    a = b;
    b = pom;
    printf("Nakon zamjene:\n");
    printf("a = %d, b= %.d\n", a, b);
    return 0;
}
```





zamjena01.c

Anatomija C programa



```
#include <stdio.h>
int main(){
    int a = 5;
    int b = 7;
    int pom;
    printf("Prije zamjene:\n");
    printf("a = %d, b= %.d\n", a, b);
    pom = a;
    a = b;
    b = pom;
    printf("Nakon zamjene:\n");
    printf("a = %d, b= %.d\n", a, b);
    return 0;
}
```

deklaracija cjelobrojne varijable a i pridruživanje vrijednosti varijabli

deklaracija cjelobrojne varijable b i pridruživanje vrijednosti varijabli

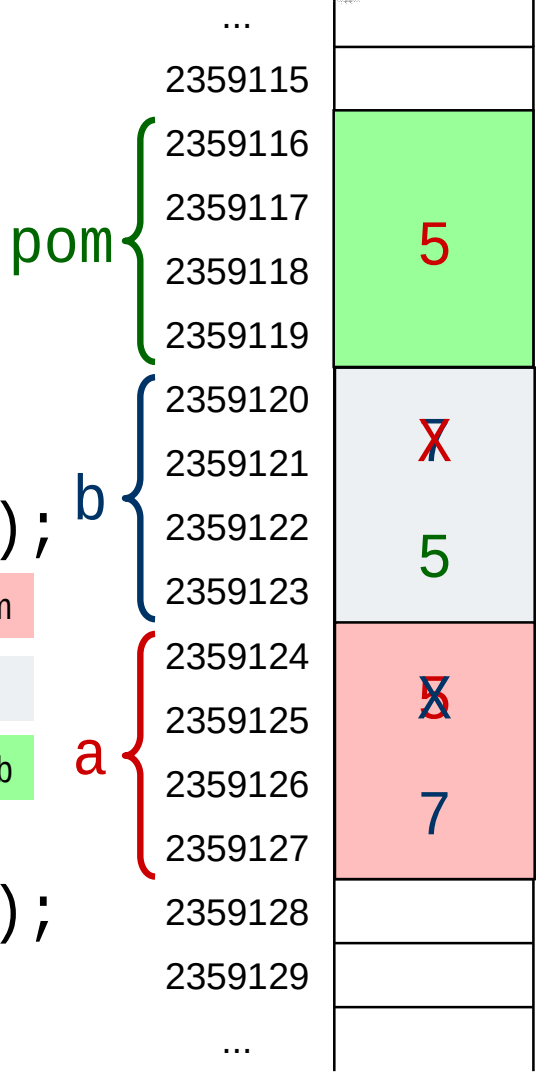
deklaracija cjelobrojne pomoćne varijable

spremanje vrijednosti varijable a u varijablu pom

spremanje vrijednosti varijable b u varijablu a

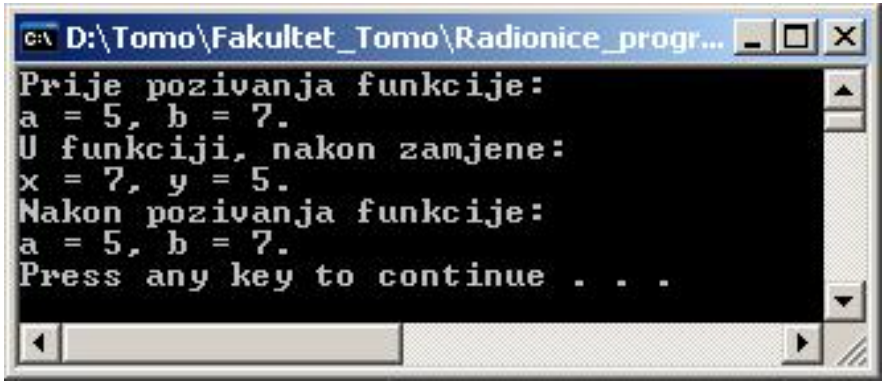
spremanje vrijednosti varijable pom u varijablu b

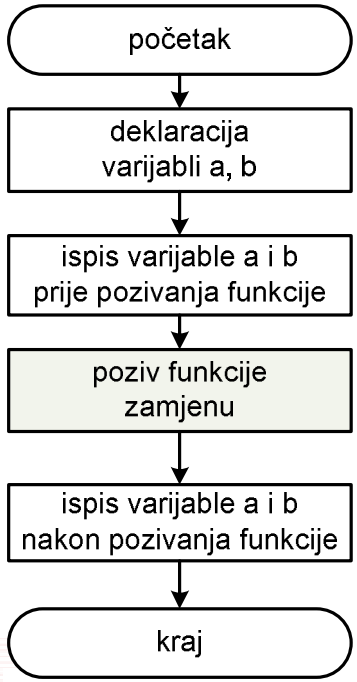
```
Nakon zamjene:
a = 7, b = 5.
```



Zamjena vrijednosti v0.2 (s funkcijom)

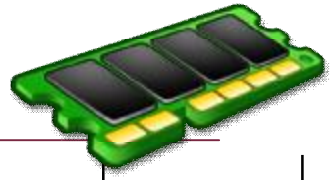
```
#include <stdio.h>
void zamjena(int x, int y){
    int pom;
    pom = x;
    x = y;
    y = pom;
    printf("U funkciji, nakon zamjene:\n");
    printf("x = %d, y = %d.\n", x, y);
}
int main(){
    int a = 5;
    int b = 7;
    printf("Prije pozivanja funkcije:\n");
    printf("a = %d, b = %d.\n", a, b);
    zamjena(a, b);
    printf("Nakon pozivanja funkcije:\n");
    printf("a = %d, b = %d.\n", a, b);
    return 0;
}
```





zamjena02.c

Anatomija C programa



```
#include <stdio.h>
void zamjena(int x, int y){
    int pom;
    pom = x;
    x = y;
    y = pom;
    printf("U funkciji, nakon zamjene:\n");
    printf("x = %d, y = %d.\n", x, y);
}
int main(){
    int a = 5;
    int b = 7;
    printf("Prije pozivanja funkcije:\n");
    printf("a = %d, b = %d.\n", a, b);
    zamjena(a, b);
    printf("Nakon pozivanja funkcije:\n");
    printf("a = %d, b = %d.\n", a, b);
    return 0;
}
```

1. parametru funkcije x se predaje vrijednost varijable a

2. parametru funkcije y se predaje vrijednost varijable b

deklaracija pomoćne varijable

spremanje vrijednosti varijable x u varijablu pom

spremanje vrijednosti varijable y u varijablu x

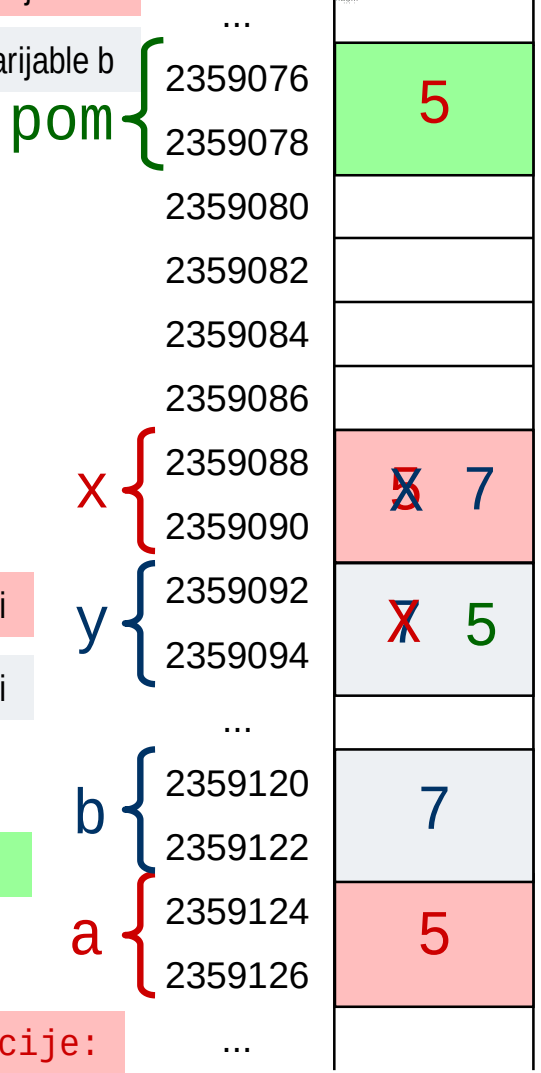
spremanje vrijednosti varijable pom u varijablu y

deklaracija varijable a i pridruživanje vrijednosti

deklaracija varijable b i pridruživanje vrijednosti

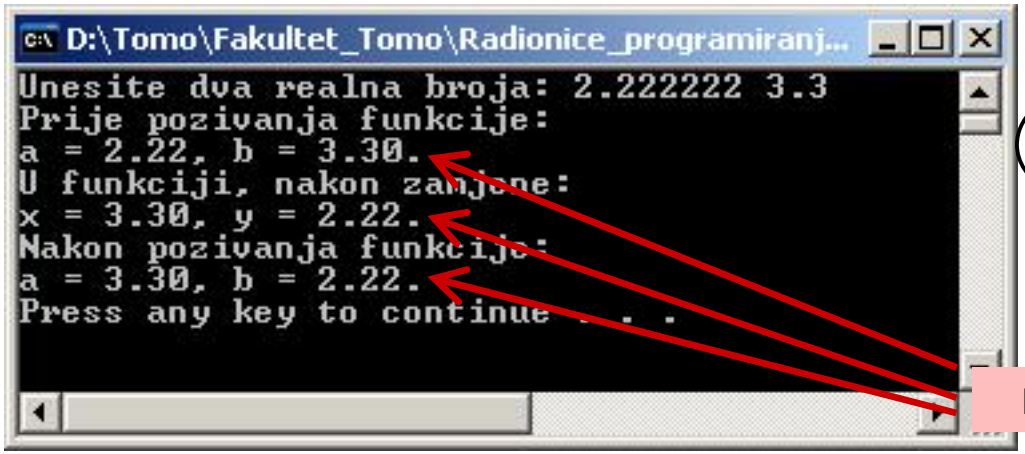
poziv funkcije zamjena

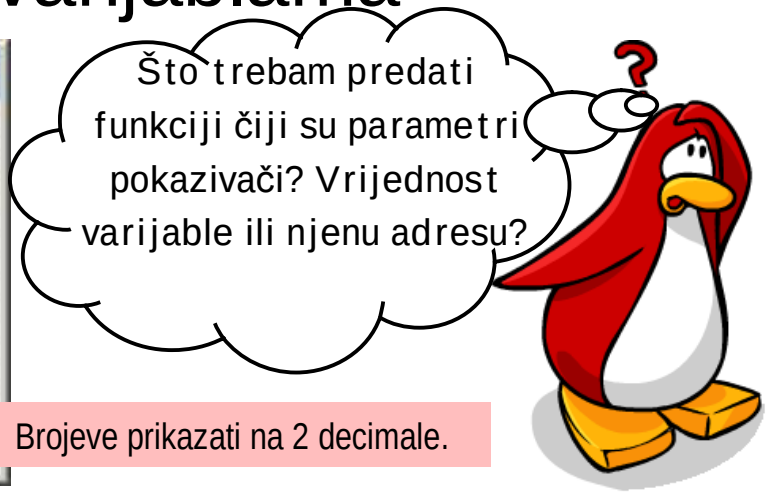
Nakon pozivanja funkcije:
a = 5, b = 7.



Pokazivači kao parametri funkcije

- **Izmijenite** prethodnu funkciju tako da su ulazni parametri **pokazivači** koji pokazuju na realni broj, a ne obične varijable
- U glavnom programu se trebaju unositi dva realna broja s tipkovnice umjesto pridruživanja vrijednosti varijablama





Brojeve prikazati na 2 decimale.

Aritmetika pokazivača

Prvi izraz će povećati vrijednost varijable na koju pokazuje pokazivač p za 10 puta!

```
*p = *p * 10;
```

Što ćemo dobiti s prvim izrazom *p = *p * 10;?

- unarni operator dereferenciranja * ima veći prioritete od svih matematičkih i logičkih operacija

```
p = p + 1;
```

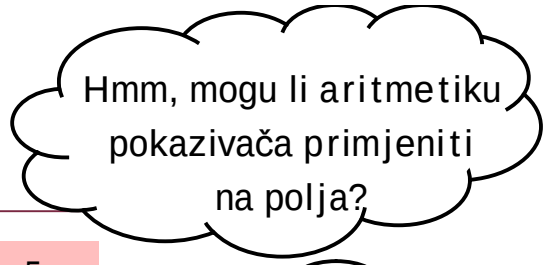
- ovisno o tipu pokazivača vrijednost adrese na koju pokazuje će se povečati za **1**, **4** ili **8**.

Drugi izraz će povećati adresu na koju pokazuje pokazivač p, tako da sad pokazivač p pokazuje na sljedeću adresu!

`char` `int, float` `double`

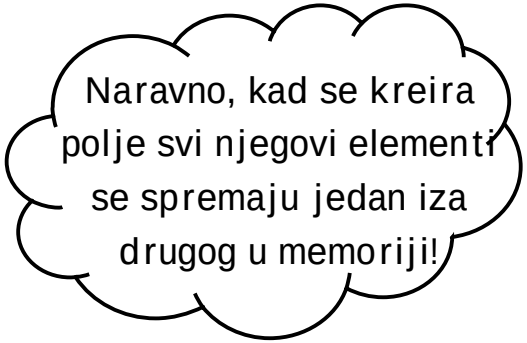
A što ćemo dobiti s ovim drugim izrazom p = p + 1;?

Pokazivači i polja




```
int a[5], *p;
```

→ deklaracija polja a s 5 elemenata i pokazivača p




```
p = a;   <==>   p = &a[0];
```

postavljanje da pokazivač pokazuje na adresu prvog elementa polja

- pristup elementima polja

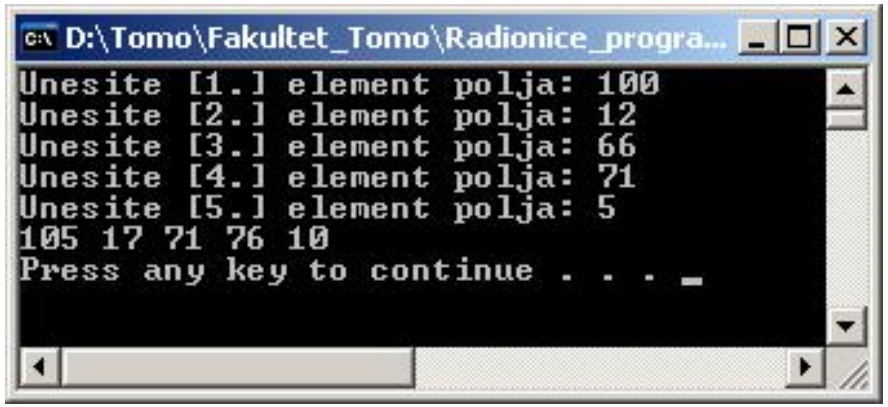
`&a[0]`	⟷	`p`
`&a[1]`	⟷	`p + 1`
`&a[2]`	⟷	`p + 2`
...	...	...
`&a[i]`	⟷	`p + i`

- pristup vrijednostima elemenata polja

`a[0]`	⟷	`*p`
`a[1]`	⟷	`*(p + 1)`
`a[2]`	⟷	`*(p + 2)`
...	...	...
`a[i]`	⟷	`*(p + i)`

Pokazivači i polja

- napisati funkciju koja pronalazi najmanji element polja, pa ga zatim dodaje svim elementima polja
- ulazni parametri funkcije će biti pokazivač na polje i broj elemenata polja, a kod rada s poljem u funkciji treba koristiti **aritmetiku pokazivača**
- funkcija nema povratnu vrijednost
- u glavnom programu se unose vrijednosti elemenata polja, poziva funkcija i ispisuje polje s novim vrijednostima.



Algoritmi

- algoritam predstavlja precizno opisan način za rješenje nekog problema
- algoritam uvijek treba jednoznačno odrediti što treba napraviti
- kao primjer algoritma se često koriste razne vrste sortiranja

Mjehuričasto sortiranje (bubble sort)

- kod mjehuričastog sortiranja se zamjenjuju susjedni elementi ako nisu u dobrom redoslijedu
- postupak:
  - kreće se od početka polja do kraja polja
  - ako je sljedeći element manji od prethodnog zamjeni ih
  - ako se prođe kroz cijelo polje bez ijedne zamjene, polje je sortirano

Primjer sortiranja bubble sortom

1. prolaz kroz polje				
4	2	5	6	1
2	4	5	6	1
2	4	5	6	1
2	4	5	6	1
2	4	5	1	6

2. prolaz kroz polje				
2	4	5	1	6
2	4	5	1	6
2	4	5	1	6
2	4	1	5	6

3. prolaz kroz polje				
2	4	1	5	6
2	4	1	5	6
2	1	4	5	6
2	1	4	5	6

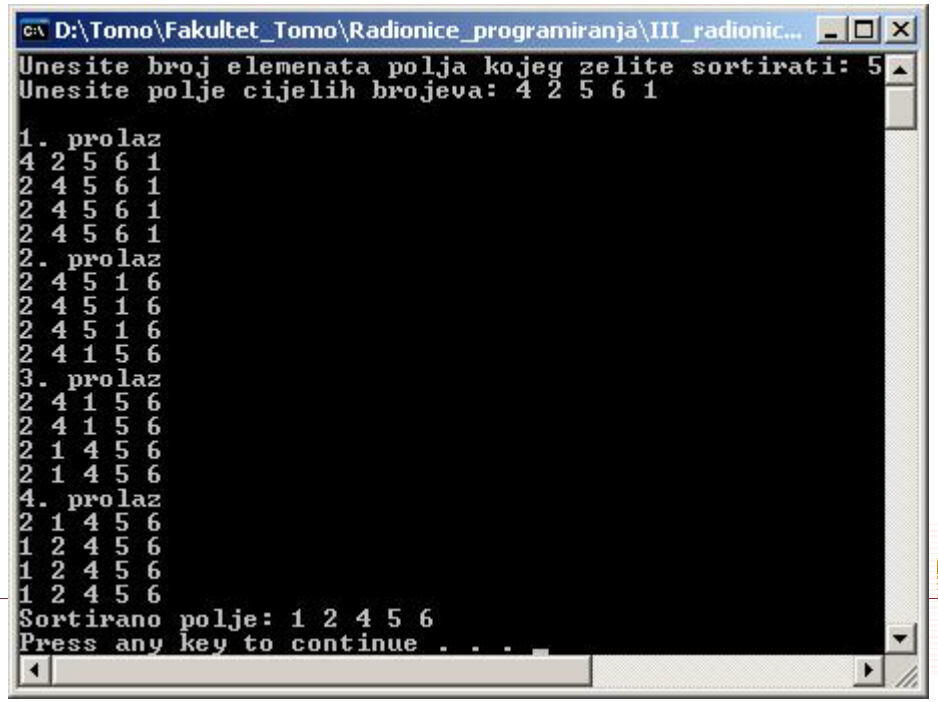
4. prolaz kroz polje				
2	1	4	5	6
1	2	4	5	6
1	2	4	5	6
1	2	4	5	6

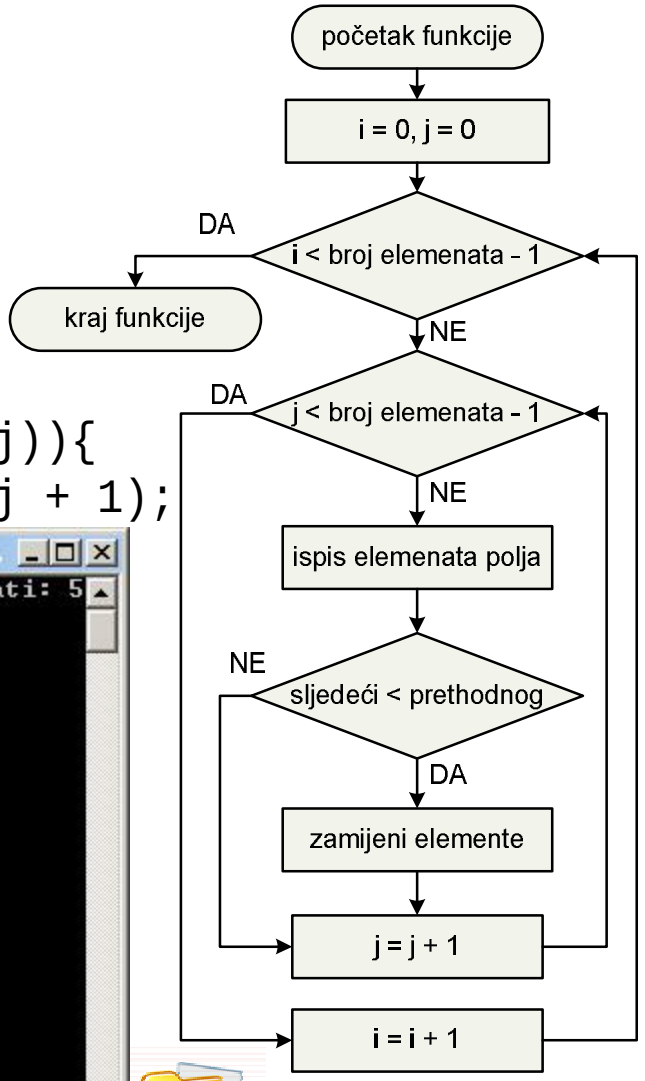
5. prolaz kroz polje				
1	2	4	5	6
1	2	4	5	6
1	2	4	5	6
1	2	4	5	6

Nakon što se prođe kroz cijelo polje bez ijedne zamjene možemo reći da je polje sortirano!

Sortiranje bubble sortom

```
void bubble_sort(int *polje, int brel){
    int i, j;
    for(i=0; i < brel-1; i++){
        printf("%d. prolaz\n", i+1);
        for(j=0; j < brel-1; j++){
            ispis(polje, brel);
            if(*(polje + j + 1) < *(polje + j)){
                zamjena(polje + j, polje + j + 1);
            }
        }
    }
}
```





bubble_sort.c

Anatomija C programa

```
void zamjena(int *x, int *y);
void ispis(int *polje, int brel);
void bubble_sort(int *polje, int brel){
    int i, j;
    for(i=0; i < brel-1; i++){
        printf("%d. prolaz\n", i+1);
        for(j=0; j < brel-1; j++){
            ispis(polje, brel);
              if(*(polje + j + 1) < *(polje + j)){
                  zamjena(polje + j, polje + j + 1);
              }
        }
    }
}
```

funkcija za zamjenu vrijednosti 2 cijela broja

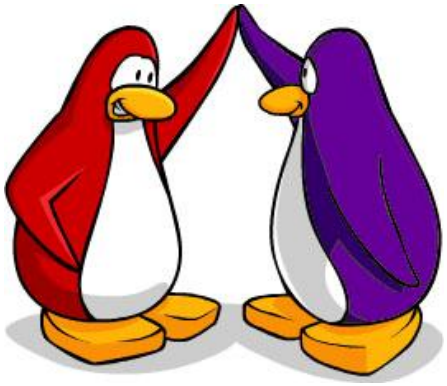
funkcija za ispis elemenata polja

funkcija za sortiranje polja bubble sortom

da bi se polje sortiralo treba proći broj elemenata - 1 puta kroz polje

u svakom prolasku kroz polje uspoređujemo susjedne elemente od početka polja

ako je sljedeći element veći od prethodnog poziva se funkcija zamjena koja će zamijeniti njihove vrijednosti



Selection sort

- izmijenite prethodni program tako da se umjesto korištenja funkcije za bubble sort koristi funkcija za **selection sort**
- selection sort pronalazi najmanji element polja i zamijeni ga s prvim elementom, to se onda ponavlja s ostatkom niza

4	2	5	6	1
1	2	5	6	4
1	2	5	6	4
1	2	4	6	5
1	2	4	5	6
1	2	4	5	6