# POKAZIVAČI

Pokazivači predstavljaju poseban izveden tip podataka. Podatak tipa pokazivača sadrži memorijsku adresu nekog drugog podatka.

## Deklaracija pokazivača:

<tip> *<ime>;

<tip> - tip podatka na koji ukazuje pokazivačka promenljiva,

<ime> - ime pokazivačke promenljive.

## Operatori koji se koriste u radu sa pokazivačima

- *Operator referenciranja (&)* - određuje adresu neke promenljive, tj. vraća adresu promenljive ispred koje je napisan;
- *Operator dereferenciranja (*)* - određuje vrednost promenljive na koju ukazuje neka pokazivačka promenljiva, tj. vraća sadržaj lokacije na koju pokazuje pokazivač.

Primeri:

```
int i=0, j=0;      //promenljive su tipa int
int *pi;           //pi je pokazivač na int
pi=&i;             //pi ukazuje na promenljivu i
*pi=2;             //promenljiva na koju ukazuje pi dobija vrednost 2, tj. i=2
j=*pi;             //j dobija vrednost 2 , j=i
pi=&j;             //pi dobija adresu promenljive j, tj. ukazuje na j
```

## Pokazivač tipa void (generički pokazivač)

Pokazivač na tip void može da primi adresu bilo kog objekta, tj. direktno mu se može dodeliti vrednost pokazivača na bilo koji tip. To je "pokazivač na nešto u memoriji".

Primeri:

```
int i=0;
int *pi1=&i, *pi2;
void *pv;              //pokazivac na bilo koji tip
pv=pi1;                //pv ukazuje na i
pi2=pv;                //nemoguca dodela, greska
pi2=(int*)pv;          //ovakva dodela je korektna, izvršena je konverzija tipa
```

## Inicijalizacija pokazivača

Pre korišćenja potrebno je izvršiti inicijalizaciju pokazivača, tj. dodeliti mu neku adresu ili NULL vrednost, oznaku da ne ukazuje ni na šta. NULL je simbolička konstanta definisana u fajlu stdio.h i njena vrednost je 0. Uobičajeno je da se pokazivačima dodeljuje NULL, a ne 0, da bi se jasno istaklo da se radi o pokazivačkoj promenljivoj.

## Veza polja i pokazivača

Ime polja predstavlja pokazivač koji ukazuje na prvi element polja.

Primeri:

```
int polje[10], *pok_polja, a;
pok_polja=&polje[0];
```

Prethodna naredba je ekvivalentna naredbi:

```
pok_polja=polje;
```

Korišćenjem pokazivaca možemo pristupati i elementima polja. Naredbu:

```
a=polje[0];
```

možemo da napišemo u obliku:

```
a=*polje;
```

ili

```
a=*pok_polja;
```

## Adresna aritmetika

U programskom jeziku C dozvoljene su sledeće operacije nad pokazivačima:

- dodela vrednosti jednog pokazivača drugome (samo ako su istog tipa);
- dodavanje ili oduzimanje celobrojnog podatka od vrednosti pokazivača (ako je pokazivač pa=&a[k], pa+1=&a[k+1] bez obzira kolika je dužina jednog elementa vektora a – ne dodaje se bukvalno 1 – već dužina podatka odgovarajućeg tipa);
- upoređivanje 2 pokazivača (ima smisla samo ako ukazuju na elemente istog niza);
- poređenje pokazivača sa nulom (ispituje da li pokazivač uopšte ukazuje na neki objekat – umesto nule koristi se konstanta NULL);

Ako pok_polja ukazuje na *i*-ti element polja, pok_polja+1 i pok_polja-1 ukazuju na (*i*+1). i (*i*-1). element, respektivno. To znači da se korišćenjem pokazivača može pristupiti proizvoljnom elementu polja.

Neka je niz *a* definisan na sledeći način:

```
int a[50];
```

*k*-tom članu niza se može pristupiti korišćenjem indeksa na poznati način:

```
a[k]
```

ili korišćenjem pokazivača:

```
*(a+k)
```

# DINAMIČKA DODELA MEMORIJE

Naredbama za definisanje promenljivih, koje smo do sada pominjali, nalagano je kompilatoru da pri prevođenju automatski rezerviše memorijski prostor za te promenljive. Prostor se zauzima kada se počne sa izvršenjem programa, a oslobađa se po završetku izvršenja. Ovakav način dodele memorije naziva se statička dodela memorije. Kada se polja pamte u statičkoj zoni memorije, potrebno je predvideti njihovu maksimalnu veličinu, čime se određuje koliki se prostor rezerviše za smeštanje njihovih elemenata i toliki deo memorijskog prostora se zauzima pri svakom izvršenju programa i za sve vreme izvršenja programa ili bloka u kojem su polja definisana.

Dinamička dodela memorije podrazumeva dodelu memorijskog prostora promenljivima u toku izvršenja programa. U tom slučaju u statičkoj zoni memorije definiše se jedan pokazivač na taj deo memorijskog prostora. U toku rada programa može da se rezerviše memorijski prostor na koji pokazivač ukazuje, da se menja veličina tog memorijskog prostora i da se on oslobađa te da se u nastavku izvršenja programa isti memorijski prostor koristi za smeštanje drugih podataka.

## Funkcije za upravljanje dinamičkom zonom memorije

Bibliotečke funkcije za upravljanje dinamičkom zonom memorije su deklarisane su u fajlu stdlib.h. To znači da, kad god u svom kodu pozivamo neku od njih, treba uključiti fajl stdlib.h. Ovoj grupi funkcija pripadaju:

1. malloc(<velicina>) – rezerviše deo memorijskog prostora zadate veličine. Veličina se zadaje brojem bajtova. Funkcija vraća generički pokazivač (tipa void*) na rezervisani memorijski prostor, ili NULL ako rezervacija ne može da se izvrši. Sadržaj rezervisanog prostora je nedefinisan.

**Primer**

Rezervisati memorijski prostor za vektor tipa int od N komponenata.

```
int *vektor, N;
printf("unesite broj elemenata u vektoru\n");
scanf("%d\n",&N);
vektror=(int*)malloc(N*sizeof(int));
if (vektor==NULL)
      printf("nema mesta u mem.\n");
else …
```

2. calloc(<broj>,<velicina>) – rezerviše deo memorijskog prostora za pamćenje navedenog broja elemenata zadate veličine. Funkcija vraća generički pokazivač (tipa void*) na rezervisani memorijski prostor, ili NULL ako rezervacija ne može da se izvrši. Rezervisani deo memorisjkog prostora se inicijalno popunjava nulama.

   **Primer**

   vektor=(int*)calloc(N,sizeof(int));

3. realloc(<pokazivac>,<velicina>) – Ova funkcija menja veličinu memorijskog prostora na koji ukazuje zadati pokazivač na zadatu veličinu (tj. na zadati broj bajtova). Veličina memorisjkog prostora, na koji ukazuje navedeni pokazivač, na ovaj način može da se smanji ili da se poveća.

   **Primer**

   realloc(vektor, K*sizeof(int));

4. free(<pokazivac>) – oslobađa deo memorijskog prostora na koji ukazuje navedeni pokazivač.

## Smeštanje matrice u dinamičkoj zoni memorije

Matrica u C-u predstavlja niz nizova i u statičkoj zoni memorije ona se pamti u linearizovanom obliku (matrica reda *m*x*n* se pamti kao niz dužine *m***n* pri čemu se linearizacija vrši po vrstama).

U dinamičkoj zoni memorije matrica može biti predstavljena na tri načina:

1. linearizovano – matrica se zamenjuje vektorom odgovarajuće dužine;
2. tako što se u dinamičkoj zoni memorije rezerviše prostor za onoliko nezavisnih vektora koliko vrsta ima matrica, a u statičkoj zoni memorije se matrica definiše kao vektor pokazivača na vrste;
3. tako što se i vrste i vektor pokazivača na vrste smeštaju u dinamičkoj zoni memorije – u statičkoj zoni memorije u tom slučaju postoji samo pokazivač na vektor pokazivača na vrste.

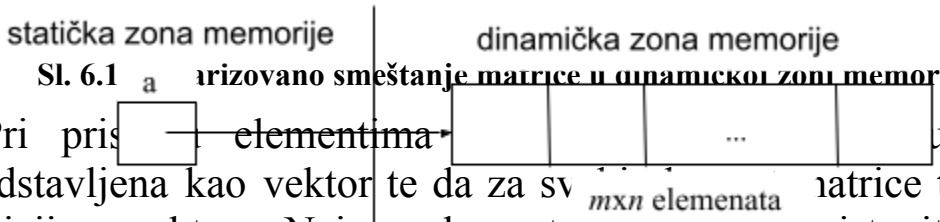
**Primer**

Rezervisati prostor u dinamičkoj zoni memorije za smeštanje elemenata matrice reda *m*x*n*.

***I način:***

```
int *a,m,n,i,j;
a=(int*)calloc(m*n,sizeof(int));
```

Šematski prikaz na ovaj način organizovanih podataka u memoriji prikazan je na Sl. 6.1.




**Sl. 6.1 a ...rizovano smeštanje matrice u dinamičkoj zoni memorije**

Pri pris... elementima ...una o tome da je ona predstavljena kao vektor te da za sv... matrice treba preračunati njegovu poziciju u vektoru. Naime, elementu $a_{ij}$ se može pristupiti korišćenjem indeksa na sledeći način:

…a[n*i+j]…

ili, korišćenjem pokazivača na sledeći način:

…*(a+n*i+j)

***II način:***

```
int *a[10],m,n,i,j;
for (i=0;i<m;i++)
   a[i]=(int*)calloc(n,sizeof(int));
```

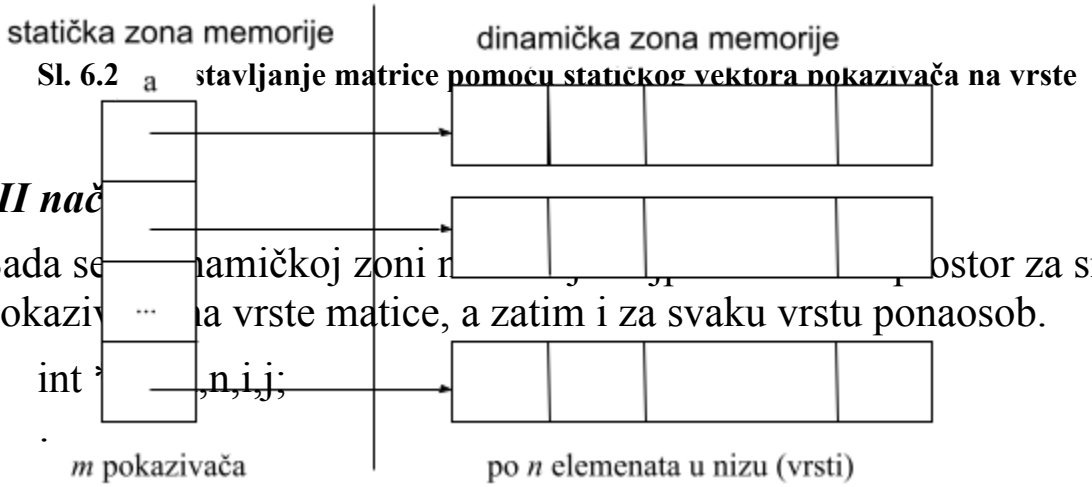
U ovom slučaju, podaci u memoriji su organizovani kao na Sl. 6.2.

Elementima matrice se sada pristupa na standardan način:

… a[i][j]…

ili:

… *(*(a+i)+j)…




**Sl. 6.2 ...stavljanje matrice pomoću statičkog vektora pokazivača na vrste**

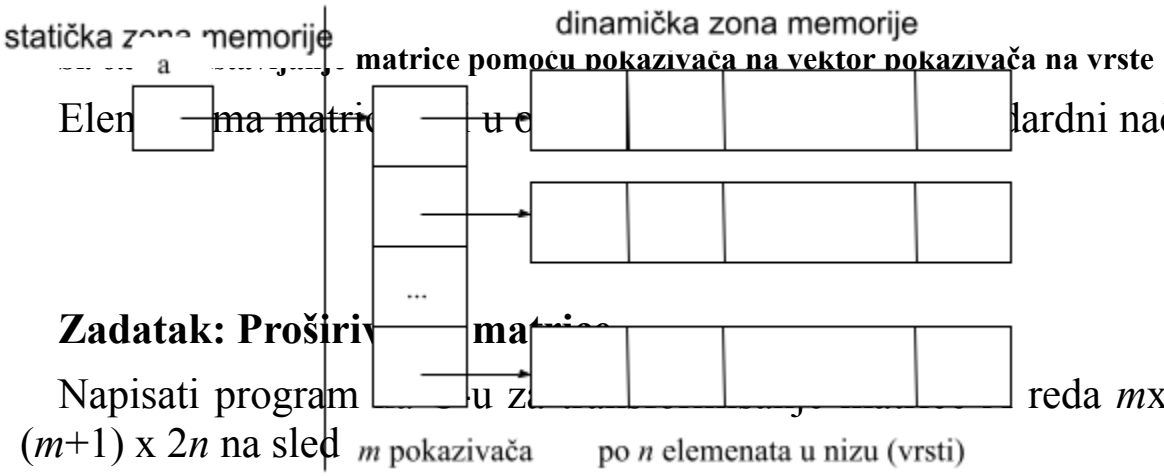
***III nač...***

Sada se ...namičkoj zoni ... ...ostor za smeštanje vektora pokaziv... ...a vrste matice, a zatim i za svaku vrstu ponaosob.

```
int *...,n,i,j;
.
```

```
a=(int**)calloc(m,sizeof(int*));
for (i=0;i<m;i++)
        a[i]=(int*)calloc(n,sizeof(int));
```

Ovakav način smeštanja matrice u memoriji prikazan je na Sl. 6.3




**Zadatak: Proširiv[illegible] matrice**

Napisati program [illegible] reda *m*x*n* u matricu reda (*m*+1) x 2*n* na sled

$$A \rightarrow \begin{bmatrix} A & 0_n \\ A & 0_n \end{bmatrix},$$

gde $0_n$ predstavlja *n* nula u vrsti.

**Napomena:** Elemente matrice A pamtiti u dinamičkoj zoni memorije. Pre i posle transformacije za elemente matrice treba da bude rezervisan onoliki deo memorijskog prostora koliko je stvarno potrebno za njihovo pamćenje.