Programski jezici i strukture podataka

10

NIZ

Niz je linearna struktura podataka koja se sastoji od konačno mnogo elemenata istog skalarnog ili strukturnog tipa.

Svaki element niza je jednako dostupan, i svakom se elementu može pristupati u proizvoljnom redosledu.

Nizovi mogu biti statički ili dinamički.

Vrste nizova

- **Jednodimenzionalni niz** ili **vektor**, za identifikaciju pojedine pozicije u vektoru koristi se indeks, vrednost indeksa u C programskom jeziku uzima vrednosti iz skupa 0,...,n-1 pri čemu je **n** dužina vektora.
- **Višedimenzionalni niz** (dvodimenzionalni se naziva **matrica**), predstavlja generalizaciju vektora.

Operacije sa nizovima

- Operacije matričnog računa
- Operacije po elementima

Smeštanje nizova u memoriji

- Za smeštanje nizova u memoriji se koristi sekvencijalna reprezentacija.
- Logička struktura niza i fizička reprezentacija se poklapaju.
- Niz se implementira kao kontinualni skup susednih memorijskih lokacija.

Dinamički niz

- U statičkoj memoriji uvek zauzimamo maksimalnu potrebnu količinu memorije.
- Statička memorija se koristi za smeštanje malih nizova.
- Za kreiranje nizova tokom izvršavanja koristimo heap.

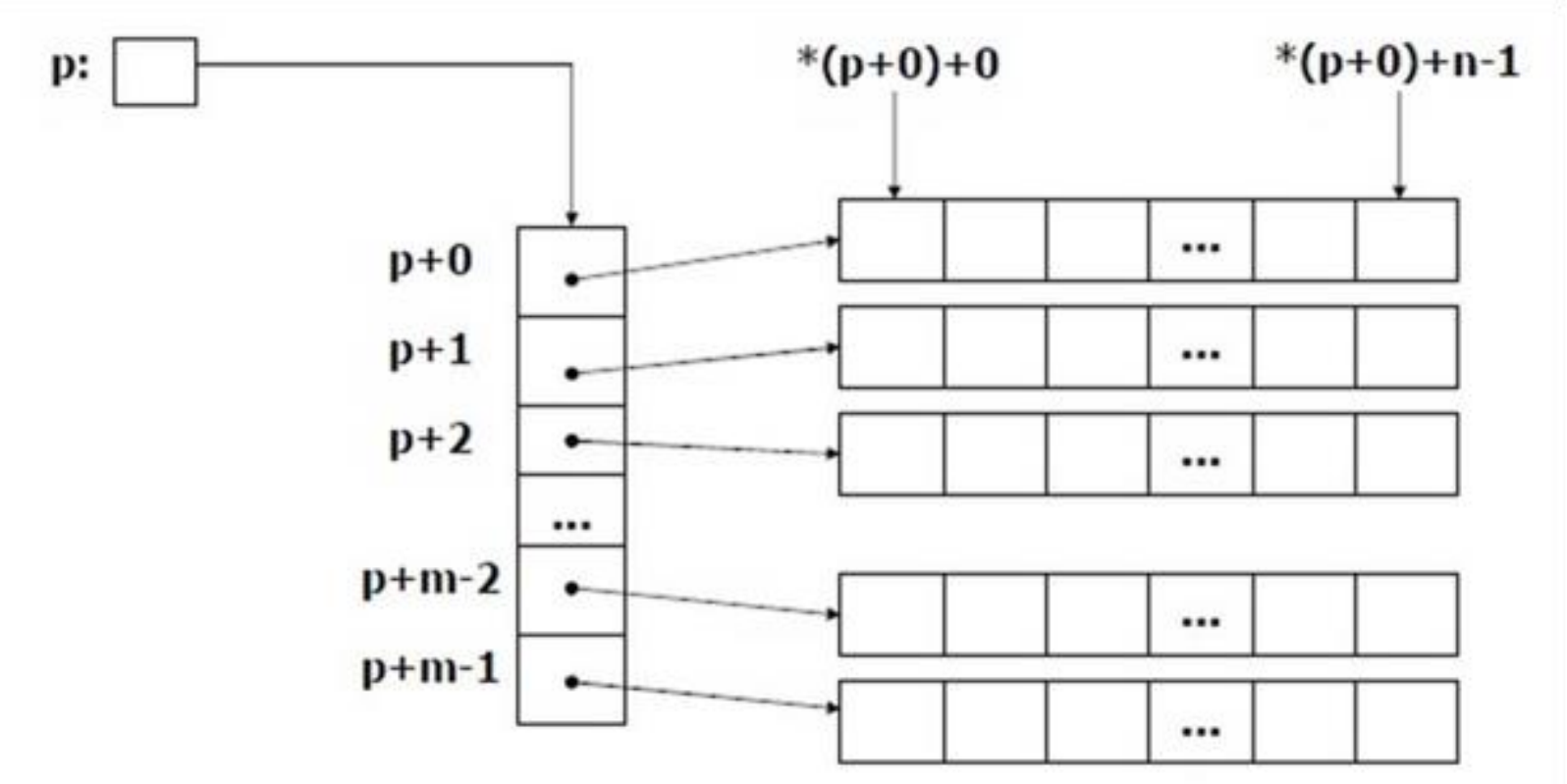
(z38.c)

Dinamička matrica

- Šta ispisuje sledeći program (**z38a.c**):
  - sadržaj matrice vrstu po vrstu, a potom sporednu dijagonalu, sleva-udesno
  - sadržaj matrice vrstu po vrstu, a potom glavnu dijagonalu, sdesna-ulevo
  - sadržaj matrice vrstu po vrstu, a potom sporednu dijagonalu, sdesna-ulevo

Dinamička matrica

- **Matrica se predstavlja kao dinamički niz pokazivača na dinamičke nizove elemenata**



Dinamička matrica

- **Moguće definisanje pokazivača na neki drugi pokazivač, do proizvoljne "dubine"**
  - pokazivač na pokazivač na celobrojni podatak:

```
int **p;
```

- **Prvi korak u stvaranju dinamičke matrice je alokacija memorije za niz pokazivača**

```
p = calloc(m,sizeof(int*));
```

  - p je statički alociran dvostruki pokazivač u koji će biti smeštena adresa niza pokazivača
  - **m** je broj vrsta (redova) matrice
  - svaki pokazivač u alociranom nizu pokazivača će ukazivati na jedan niz elemenata koji će predstavljati vrste (redove) matrice
  - p mora biti dvostruki pokazivač, jer da bi se pristupilo jednom elementu matrice (p [i][j]), mora se prvo odrediti adresa i-te vrste i pristupiti i-tom pokazivaču u nizu pokazivača, a zatim odrediti adresa j-tog elementa u nizu elemenata i pristupiti tom elementu

```
p[i][j] isto što i: *(*(p+i)+j)
```

Dinamička matrica

- **Drugi korak u stvaranju dinamičke matrice je alokacija vrsta matrice**

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

  - u svakoj iteraciji petlje se alocira prostor za jednu od m vrsta matrice
  - alocirani nizovi koji predstavljaju vrste matrice ne moraju biti fizički susedni u memoriji, već mogu biti razbacani bilo gde
- **Uništavanje (dealociranje) matrice se radi u obrnutom redosledu od redosleda kreiranja**
  - prvo se dealociraju nizovi koji predstavljaju vrste, a zatim niz pokazivača na vrste

```
for (i=0; i<m; i++)
    free (p[i]);
free (p);
```

Dinamička matrica

```
printf("N="); scanf("%d", &n) ;
/*alocira memoriju za n pokazivaca na vrste*/
a = calloc(n, sizeof(int*));
for (i=0; i<n; i++)
{
/*alocira memoriju za n elemenata vrste koji su tipa int*/
      *(a+i)= calloc(n, sizeof(int));
      for (j=0; j<n; j++)
            *(*(a+i)+j) =
rand()/((double)RAND_MAX+1)*10;
}
```

- Biće stvorena kvadratna dinamička matrica dimenzija n x n
  - dimenzija n se unosi sa glavnog ulaza
- Matrica će biti popunjena pseudoslučajnim brojevima u rasponu od 0 do 9

Dinamička matrica

- **Deo koda:**

```
for (i=0; i<n; printf("\n"), i++)
     for (j=0; j<n; j++)
          printf("%d ", *(*(a+i)+j));
```

- **ispisuje sadržaj matrice, vrstu po vrstu**
  - nakon svake iteracije spoljne petlje će u post uslovu biti odštampan znak za novi red
- **Šta radi sledeći deo koda?**

```
for (i=n-1; i>=0; i--)
printf("%d ", *(*(a+i)+n-1-ì));
```

- **Uzmimo primer kvadrate matrice za n = 4**

Dinamička matrica

- Izraz: *(*(a+i)+n-1-i));

Za i = 3

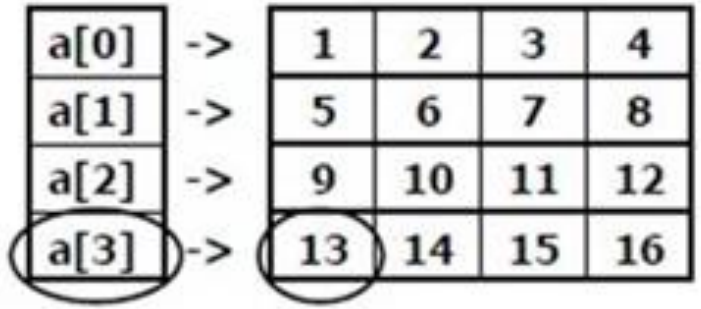
*(a + 3) je isto kao: a[3]

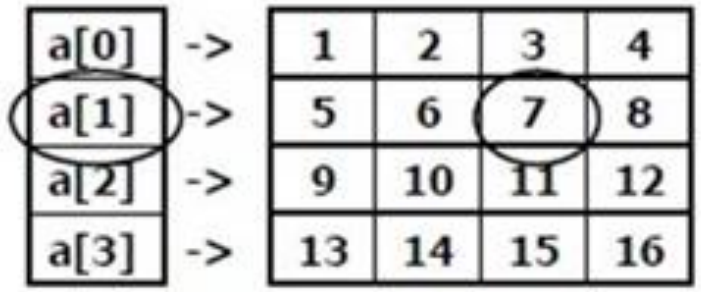
((a + 3) + 4 - 1 - 3) je isto kao: a[3][0]

Za i = 1

*(a + 1) je isto kao: a[1]

((a + 1) + 4 - 1 -1) je isto kao: a[1][2]





- Prednost nizova (kao linearnih struktura podataka) je podudaranje fizičke i logičke strukture, kao i linearna adresna funkcija.
- Nedostatak nizova možemo videti kroz komplikovane operacije umetanja i brisanja elementa na proizvoljnom mestu, a kod sekvencijalne realizacije i kroz statički alociran memorijski prostor.
- Navedeni nedostaci se mogu prevazići spregnutom realizacijom.