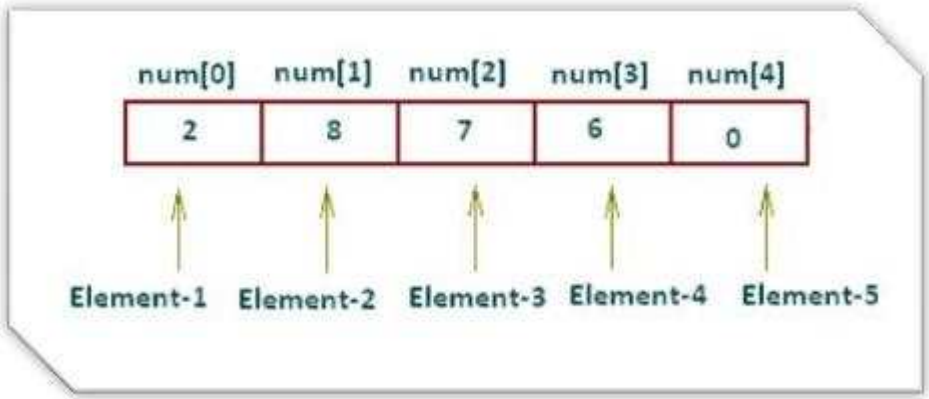
Nizovi

predavač: Nadežda Jakšić

Programiranje programski jezik C

Jednodimenzionalni niz

- najjednostavniji i najčešće korišćen složeni tip podatka
- niz podataka istog tipa koji se u memoriji smeštaju jedan iza drugog
- nazivaju se elementima niza i identifikuju se pomoću rednog broja (indeksa)
- prvi element niza ima indeks **nula**
- ako niz ima **n** elemenata, onda poslednji element niza ima indeks **n-1**



Jednodimenzionalni niz

- deklarisanje
  tip identifakator [n]; //n je broj elemenata niza

- elementi niza mogu da pripadaju nekom od elementarnih tipova (**char**, **int**, **float**, **double**), mogu da budu pokazivači, strukture ili nizovi

- funkcije nije dozvoljeno direktno koristiti kao elemente, ali se mogu predstaviti deklarisanjem pokazivača na funkciju koji će dobiti element niza

- elementima niza se pristupa preko indeksa

Inicijalizacija i unos

- inicijalizacija podrazumeva postavljanje početnih vrednosti

```
dani[0]=31;
dani[1]=28;
…
dani[11]=31;
```

ili

```
int dani[12]={31,28,31,30,31,30,31,31,30,31,30,31};
```

ili unos vrednosti preko tastature

```
int dani[12], i;
for (i=0;i<12;i++)
{
    printf ("Unesite broj dana\n"); //printf ("meses[%2d]=\n", i+1);
    scanf ("%d",&dani[i]);}
```

Niz kao argument funkcije

```
//funkcija vraća aritmetičku sredinu elemenata niza
float nadjiProsek (int n, float niz[])
   {int i; float s=0;
    for (i=0;i<n;i++)
    s+=niz[i];
return (s/n);}
int main ()
{ int i,n;f loat niz[10];
    printf ("Koliko elemenata ima niz?\n");
    scanf ("%d",&n);
    for (i=0;i<n;i++)
          {printf ("\nUnesite %d. element niza  ", i+1);
           scanf ("%f",&niz[i]);}
printf ("Aritmeticka sredina unetih vrednosti je %.2f", nadjiProsek
    (n,niz));return 0;}
```

Programi

1. učitati niz od 10 brojeva i sabrati ih
2. učitati niz od **n** brojeva i pomnožiti ih
3. učitati niz, svaki element niza uvećati za 5 i prikazati novi niz
4. učitati niz i prikazati koliko je unetih brojeva deljivo sa **tri**
5. prikazati najveći element niza
6. prikazuje se svaki drugi element niza počevši od poslednjeg
7. napisati funkcije za unos elemenata u niz i za prikazivanje niza
8. pronaći indeks najmanjeg elementa niza
9. od učitanih nizova **a** i **b** formira se niz **c** tako što se sabiraju parovi **a** i **b**, ako je indeks niza paran broj, a množe se parovi **a** i **b** , ako je indeks niza neparan broj

Programi

10. meteorološka stanica je za **n** dana posmatranja formirala tablicu vrednosti atmosferskog pritiska, temperature i vlažnosti vazduha; napisti program koji:
    - učitava vrednosti za pritisak, temperaturu i vlažnosti vazduha
    - određuje maksimalne i minimalne vrednosti za sve tri veličine koje se posmatraju, kao i redni broj odgovarajućeg dana
    - određuje srednju vrednost u periodu posmatranja za sve tri veličine

    napisati funkcije koje određuju minimalnu, maksimalnu i srednju vrednost niza

Nizovi i sizeof

- operator **sizeof** se koristi kada treba da se odredi broj elemenata niza u slučaju kada dimenzija niza nije navedena u toku inicijalizacije

```
int niz[] = { 0, 1, 2, 3, 4 }; //deklaracija niza od 5 elemenata
printf ("%d",sizeof(niz)); //prikazuje 20 (5 elemenata * 4 bajta)
```

- ne postoji direktan način da se ispita koliko elemenata ima niz, ali je moguće to utvrditi korišćenjem operatora **sizeof**

int nElemenata = sizeof (niz) / sizeof (niz[0]);

- s ozbirom da svi elementi niza imaju istu veličinu (pošto se radi o istom tipu podatka), deljenjem ukupne količine memorije niza sa veličinom memorije koja odgovara jednom (obično prvom) elementu niza dobija se broj elemenata niza

Pretraživanje nizova

Linerano pretraživanje – uzastopno upoređivanje elemenata niza sa traženom vrednošću; jednostavan algoritam ali prilično spor

- **primer:** niz **a** je dužine **n**, funkcija traži **indeks** traženog elementa **t** unutar niza **s**, ako ga ne pronađe vraća **-1**

```
int trazi (int a[], int n, int t)
{
    int i;
    for (i=0;i<n;i++)
        if (a[i]==t) return i;
        return (-1);}
```

Linerano pretraživanje

```
include <stdio.h>
linearnaPretraga (int niz[], int brElem, int x)
{int i;
for (i = 0; i<brElem; i++)
if (niz[i] == x) return i;
return -1;}
int main()
{int a[] = {4, 3, 2, 6, 7, 9, 11};
int brElem = sizeof(a)/sizeof(int);
int x,i;
printf ("Unesite broj koji trazite : ");
scanf ("%d",&x);
i = linearnaPretraga (a, brElem, x);
```

```
if (i == -1)
printf ("Element %d nije nadjen\n",x);
else
printf ("Indeks elementa %d je  %d a
redni broj u nizu je %d\n",x,i,i+1);
return 0;}
//korisnik unosi elemente niza
```

Binarna pretraga

binarna pretraga – niz mora da bude uređen u rastućem ili opadajućem poretku; poredi se traženi element sa središnjim elementom niza; ako nije jednak, određuje se u kojoj polovini se može naći traženi element, a druga polovina se odbacuje; proces se ponavlja

- **primer**: binarno pretraživanje u **rastućem** nizu

```
int trazi (int a[]), int n, int t) //n je dužina niza, t je tražena vrednost
{ int levi, desni, srednji;
 levi=0;
 desni=n-1;
 while (levi<= desni)
{ srednji=(levi+desni)/2;
      if (a[srednji] == t) return (srednji);
      if (t<a[srednji]) desni=srednji-1;
      else levi=srednji+1;
} return(-1); }
```

binarna pretraga je brža od linearne
mana – ne može da se pretražuje niz koji nije sortiran

Binarna pretraga - primer

```
#include <stdio.h> //niz je uređen u neopadajućem redosledu
int binarnaPretraga (int a[], int n, int x) //n je dužina niza, tražimo x
{int levi = 0;
int desni = n-1;
while (levi <= desni)
   {
    int sr = (levi+desni)/2;
    if (x == a[sr])
       return sr;
    else
       if (x < a[sr])
           desni = sr-1;
       else  levi = sr+1;
 }
return -1;}
```

```
int main()
{
 int a[] = {3, 5, 7, 9, 11, 13, 15};
 int x;
 int i;
 printf ("Unesi element koji trazimo : ");
 scanf ("%d",&x);
 i = binarnaPretraga (a, sizeof(a)/sizeof(int), x);
 if (i==-1)
        printf ("Elementa %d nema\n", x);
 else
         printf ("Pronadjen na poziciji %d\n", i+1);
return 0;}
```

Binarna pretraga - rekurzivno

```
int binarnaPretraga (int a[], int levi, int desni, int x)
{
int sr;
if (levi > desni)
  return -1;
sr=levi+(desni-levi)/2;
if (x>a[sr])
return binarnaPretraga (a,sr+1,desni,x);
else if (x<a[sr])
  return binarnaPretraga (a,levi,sr-1,x);
else
  return sr;
}
```

```
int main()
{
int a[] = {3, 5, 7, 9, 11, 13, 15};
int x,i,levi=1,desni;
printf ("Unesi element koji trazimo : ");
scanf ("%d",&x);
d=sizeof(a)/sizeof(int);
i = binarnaPretraga (a,levi,desni,x);
if (i == -1)
printf ("Elementa %d nema\n", x);
else
printf (“Na poziciji %d\n", i+1);
return 0;}
```

Sortiranje nizova

selection sort

- u prvoj iteraciji poredi se prvi element niza sa ostalim članovima niza i ako se naiđe na element koji je manji od prvog, zamene se mesta (na kraju iteracije na prvom mestu se nalazi najmanji element niza)
- u drugoj iteraciji poredi se drugi element sa ostalim elementima niza...u poslednjoj iteraciji se porede pretposlednji i poslednji element niza

```
int niz[10], i, j, temp;
for (i=0;i<9;i++) //od prvog do pretposlednjeg
for (j=1;j<10;j++) //od drugog do poslednjeg
    if (niz[i]<niz[j])
        {temp=n[i]; //zamena mesta preko promenjljive temp
        n[i]=n[j];
        n[j]=temp;}
```

efikasniji algoritmi za sortiranje su **quick sort**, **bubble sort** i **merge sort**

Sortiranje nizova - primer

```
#include <stdio.h> //niz se sortira u nerastućem poretku a[0]>=a[1]>=...a[n-1]
#include <stdlib.h> //zbog exit
#define MAXDUZ 100
int main()
{int a[MAXDUZ];
 int n,pom,i,j;
 printf ("Unesite dimenziju niza\n");
 scanf ("%d",&n);
 if (n>MAXDUZ)
   {printf ("Nedozvoljena vrednost za n\n");
   exit(1);}
for (i=0; i<n; i++)
   {
   printf ("Unesite %d. clan niza\n",i+1);
   scanf ("%d",&a[i]);}
```

```
for (i=0; i<n-1; i++)
   for (j=i+1; j<n; j++)
      if (a[i]<a[j])
        {
        pom=a[i];
        a[i]=a[j];
        a[j]=pom;
        }
printf ("Sortirani niz:\n");
    for (i=0; i<n; i++)
    printf ("%d\t",a[i]);
    printf ("\n");
return 0;}
```

Dvodimenzionalni nizovi

to je niz čiji su elementi jednodimenzionalni nizovi

- deklarisanje

tip identifakator [n] [m];

//matrica **nxm**, **n**Redova x **m**Kolona

- inicijalizacija

int Matrica [3][4]={{0,10,8,0},{4,7,8,2},{56,12,3,9,}}

//**3** jednodimenzionalna niza sa po **4** elementa

a_{00}	a_{01}	a_{02}	a_{03}
a_{10}	a_{11}	a_{12}	a_{13}
a_{20}	a_{21}	a_{22}	a_{23}

0	10	8	0
4	7	8	2
56	12	3	9

unos elemenata matrice

```
int a[3][4], i,j;
for (i=0;i<3;i++) //redovi od 0 do 2
for (j=0;j<4;j++) //kolone od 0 do 3
    scanf ("%d",&a[i][j])
```

prikazivanje na ekran

```
for (i=0;i<3;i++)
{
    printf ("\n");
    for (j=0;j<4;j++)
        printf ("%d", a[i][j]);
}
```

Kvadratna matrica

- matrica koja ima isti broj redova i kolona (a[i][j] tj. a[4][4])

pojmovi

$$\begin{matrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{matrix}$$

- glavna dijagonala
  ako je **i = j** onda je a[i][j] na glavnoj dijagonali
- sporedna dijagonala
  i+j = n-1
- elementi iznad glavne dijagonale
  ako je **i<j** onda je a[i][j] iznad glavne dijagonale
- elementi ispod glavne dijagonale
  ako je **i>j** onda je a[i][j] ispod glavne dijagonale

Programi

1. učitati matricu veličine **n**x**m** (sa tastature), a zatim prikazati novu matricu čiji su elementi za jedan veći od elemenata početne matrice

2. učitati kvadratnu matricu (**n**x**n**), a zatim prikazati:
   - matricu u obliku tabele
   - broj neparnih elemenata ispod glavne dijagonale
   - najveći element na glavnoj dijagonali
   - najmanji element na sporednoj dijagonali
   - broj nula iznad glavne dijagonale
   - zbir svih parnih elemenata matrice

Programi

3. matricom dimenzije **n** data je tabela jesenjeg dela fudbalskog šampionata, čiji su elementi:

   0 ako je ekipa **i** izgubila od ekipe **j**

   1 ako je rezultat nerešen

   2 ako je ekipa **i** pobedila ekipu **j**

   napisati program kojim se izračunava:

   broj ekipa koje su imale više pobeda nego poraza

   broj ekipa koje su prošle prvenstvo bez poraza

   sadržaj na glavnoj dijagonali zanemariti