ALGORITMI I STRUKTURE PODATAKA

RAČUNSKE VEŽBE – TERMIN BR. 2 – POKAZIVAČI I NIZOVI

ALDINA AVDIĆ, DIPL. INŽ.

Pokazivači - uvod

- **Primer 1**

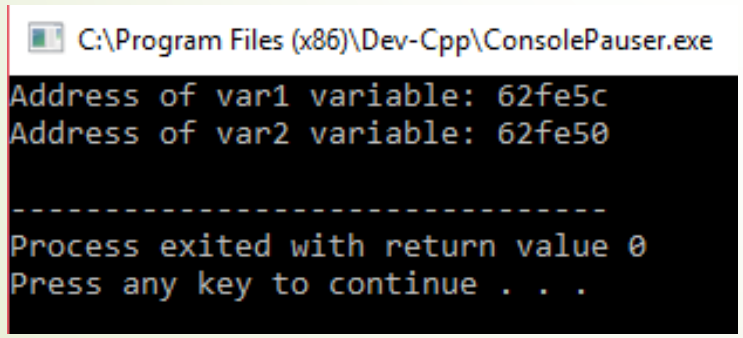
```c
#include <stdio.h>

int main () {

   int  var1;
   char var2[10];

   printf("Address of var1 variable: %x\n", &var1  );
   printf("Address of var2 variable: %x\n", &var2  );

   return 0;
}
```



- **Primer 2**

```c
#include <stdio.h>

int main () {

   int  var = 20;   /* actual variable declaration */
   int  *ip;        /* pointer variable declaration */

   ip = &var;  /* store address of var in pointer varia

   printf("Address of var variable: %x\n", &var  );

   /* address stored in pointer variable */
   printf("Address stored in ip variable: %x\n", ip );

   /* access the value using the pointer */
   printf("Value of *ip variable: %d\n", *ip );

   return 0;
}
```

```
Address of var variable: 62fe54
Address stored in ip variable: 62fe54
Value of *ip variable: 20
```

Pokazivači - operacije

- **Primer 3**

```c
#include <stdio.h>

const int MAX = 3;

int main () {

   int  var[] = {10, 100, 200};
   int  i, *ptr;

   /* let us have array address in pointer */
   ptr = var;

   for ( i = 0; i < MAX; i++) {

      printf("Address of var[%d] = %x\n", i, ptr );
      printf("Value of var[%d] = %d\n", i, *ptr );

      /* move to the next location */
      ptr++;
   }

   return 0;
}
```

```
Address of var[0] = 62fe40
Value of var[0] = 10
Address of var[1] = 62fe44
Value of var[1] = 100
Address of var[2] = 62fe48
Value of var[2] = 200
```

- **Primer 4**

```c
#include <stdio.h>

const int MAX = 3;

int main () {

   int  var[] = {10, 100, 200};
   int  i, *ptr;

   /* let us have array address in pointer */
   ptr = &var[MAX-1];

   for ( i = MAX; i > 0; i--) {

      printf("Address of var[%d] = %x\n", i-1, ptr );
      printf("Value of var[%d] = %d\n", i-1, *ptr );

      /* move to the previous location */
      ptr--;
   }

   return 0;
}
```

```
Address of var[2] = 62fe48
Value of var[2] = 200
Address of var[1] = 62fe44
Value of var[1] = 100
Address of var[0] = 62fe40
Value of var[0] = 10
```

Pokazivači – operacije i nizovi

- **Primer 5**

```c
#include <stdio.h>

const int MAX = 3;

int main () {

   int  var[] = {10, 100, 200};
   int  i, *ptr;

   /* let us have address of the first element in pc
   ptr = var;
   i = 0;

   while ( ptr <= &var[MAX - 1] ) {

      printf("Address of var[%d] = %x\n", i, ptr );
      printf("Value of var[%d] = %d\n", i, *ptr );

      /* point to the previous location */
      ptr++;
      i++;
   }

   return 0;
}
```

```
Address of var[0] = 62fe40
Value of var[0] = 10
Address of var[1] = 62fe44
Value of var[1] = 100
Address of var[2] = 62fe48
Value of var[2] = 200
```

- **Primer 6**

```c
#include <stdio.h>

const int MAX = 3;

int main () {

   int  var[] = {10, 100, 200};
   int i, *ptr[MAX];

   for ( i = 0; i < MAX; i++) {
      ptr[i] = &var[i]; /* assign the address of inte
   }

   for ( i = 0; i < MAX; i++) {
      printf("Value of var[%d] = %d\n", i, *ptr[i] );
   }

   return 0;
}
```

```
Value of var[0] = 10
Value of var[1] = 100
Value of var[2] = 200
```

Pokazivači – string i pointer na pointer

- **Primer 7**

```c
#include <stdio.h>

const int MAX = 4;

int main () {

   char *names[] = {
      "Zara Ali",
      "Hina Ali",
      "Nuha Ali",
      "Sara Ali"
   };

   int i = 0;

   for ( i = 0; i < MAX; i++) {
      printf("Value of names[%d] = %s\n", i, names[i] );
   }

   return 0;
}
```

```
Value of names[0] = Zara Ali
Value of names[1] = Hina Ali
Value of names[2] = Nuha Ali
Value of names[3] = Sara Ali
```

- **Primer 8**

```c
#include <stdio.h>

int main () {

   int  var;
   int  *ptr;
   int  **pptr;

   var = 3000;

   /* take the address of var */
   ptr = &var;

   /* take the address of ptr using address of operato
   pptr = &ptr;

   /* take the value using pptr */
   printf("Value of var = %d\n", var );
   printf("Value available at *ptr = %d\n", *ptr );
   printf("Value available at **pptr = %d\n", **pptr);

   return 0;
}
```

```
Value of var = 3000
Value available at *ptr = 3000
Value available at **pptr = 3000
```

Pokazivači – potprograma

- **Primer 9**

```c
#include <stdio.h>

/* function declaration */
double getAverage(int *arr, int size);

int main () {

   /* an int array with 5 elements */
   int balance[5] = {1000, 2, 3, 17, 50};
   double avg;

   /* pass pointer to the array as an argu
   avg = getAverage( balance, 5 ) ;

   /* output the returned value  */
   printf("Average value is: %f\n", avg );
   return 0;
}

double getAverage(int *arr, int size) {

   int  i, sum = 0;
   double avg;

   for (i = 0; i < size; ++i) {
      sum += arr[i];
   }

   avg = (double)sum / size;
   return avg;
}
```

```
Average value is: 214.400000
```

- **Primer 10**

```c
#include <stdio.h>
#include <time.h>
int * getRandom( ) {
   static int  r[10];
   int i;
   srand( (unsigned)time( NULL ) );
   for ( i = 0; i < 10; ++i) {
      r[i] = rand();
      printf("%d\n", r[i] );
   }
   return r;
}
int main () {
   int *p;
   int i;
   p = getRandom();
   for ( i = 0; i < 10; i++ ) {
      printf("*(p + [%d]) : %d\n", i, *(p + i) );
   }
   return 0;
}
```

```
7150
518
4906
4396
20426
17200
4008
22775
18649
19953
*(p + [0]) : 7150
*(p + [1]) : 518
*(p + [2]) : 4906
*(p + [3]) : 4396
*(p + [4]) : 20426
*(p + [5]) : 17200
*(p + [6]) : 4008
*(p + [7]) : 22775
*(p + [8]) : 18649
*(p + [9]) : 19953
```

Nizovi - zadaci

- **Primer 11.** Napisati program kojim se za dato **n** izračunava suma pozitivnih elemenata niza **a[1]**, **a[2]**,..., **a[n]**. Učitavanje i sumiranje izvršiti korišćenjem potprograma.

```
#include <stdio.h>
const int k=30;
void citaj(float a[], int broj)
{
    int i;
    for(i=0;i<broj;i++)
    {
        printf("Niz[%d]=", i);
        scanf("%f", &a[i]);
    }
}
```

```
float sumaPozitivnih(float a[], int broj)
{
    float Suma=0.0;
    int i;
    for(i=0;i<broj;i++)
    {
        if(a[i]>0)
        {
            Suma+=a[i];
        }
    }
    return Suma;
}
```

```
int main()
{
    float niz[k], S;
    int n, i;
    printf("Unesite n \nn=");
    scanf("%d", &n);
    citaj(niz, n);
    S=sumaPozitivnih(niz,n);
    printf("S=%.2f", S);
    return 0;
}
```

Nizovi - zadaci

- **Primer 12.** Ako je dat niz **a[1]**, **a[2]**, …, **a[n]** napisati:

a) Proceduru koja će promeniti znak svim elementima sa parnim indeksima.
b) Funkciju koja određuje broj parnih elemenata sa neparnim indeksima u nizu.

- (a)

```
void promenaZnaka(float a[], int broj)
{
    int i;
    for(i=0;i<broj;i=i+2)
    {
        a[i]=-a[i];
    }
}
```

```
int brojParnih(int a[], int broj)
{
    int i, S=0;
    for(i=1;i<broj;i=i+2)
    {
        if(a[i]%2==0)
        {
            S++;
        }
    }
    return S;

}
```

Nizovi - zadaci

- **Primer 13.** Napisati funkciju koja određuje broj promene znaka u nizu **a[1]**, **a[2]**, …, **a[n]** elemenata različitih od nule.

```
int brojPromenaZnaka(int a[], int broj)
{
    int S=0, P, i;
    for(i=0;i<(broj-1);i++)
    {
        P=a[i]*a[i+1];
        if(P<0)
        {
            S++;
        }
    }
    return S;
}
```

- **Primer 14.** Napisati funkciju koja proverava da li niz **a[1]**, **a[2]**, …, **a[n]** koji se sastoji samo od nula i jedinica, ima svojstvo da su svaka dva susedna elementa različita.

```
int razliciti Susedni(int a[], int broj)
{
    int i;
    int razliciti=1;
    for(i=0;i<(broj-1);i++)
    {
        razliciti=a[i]+a[i+1];
    }
    return razliciti;
}
```

Nizovi - zadaci

- **Primer 15.** Napisati program kojim se za zadate nizove **a[1]**, **a[2]**, …, **a[n]** i **b[1]**, **b[2]**, …, **b[n]** izračunava **a[1]*b[n]+a[2]*b[n-1]+…+a[n]*b[1]**.
- **Primer 16.** Napisati program kojim se na osnovu nizova **b[1]**, **b[2]**, …, **b[n]** i **c[1]**, **c[2]**, …, **c[n]** formira niz **a[1]**, **a[2]**, …, **a[2n]** čije su vrednosti: **b[1]**, **c[1]**, **b[2]**, **c[2]**, …, **b[n]**, **c[n]**.
- **Primer 17.** Napisati program kojim se na osnovu niza **a[1]**, **a[2]**, …, **a[2*n]** formiraju nizovi **b[1]**, **b[2]**, …, **b[n]** i **c[1]**, **c[2]**, …, **c[n]** čiji su elementi redom jednaki: **a[1]**, **a[3]**, …, **a[2*n-1]** i **a[1]**, **a[4]**, …, **a[2*n]** .
- **Primer 18.** Napisati program kojim se realizuje ciklično premeštanje vrednosti elemenata niza **a[1]**, **a[2]**, …, **a[n]** za **m** mesta ulevo.

Nizovi - zadaci

- **Primer 19.** Napisati program kojim se ispituje da li je tekst koji se unosi nizom **a[1]**, **a[2]**, …, **a[n]** palindrom. Uneti tekst je palindrom ako se isto čita od početka kao i sa kraja (na primer, MADAM, ANAVOLIMILOVANA).
- **Primer 20.** Napisati program koji umeće zadati broj **r** u dati neopadajući niz **a[1]**, **a[2]**, …, **a[n]**, pri čemu rezultujući niz **a[1]**, **a[2]**, …, **a[n+1]** ostaje u neopadajućem poretku.

Nizovi – binarno pretraživanje

- **Primer 21.** Napisati funkciju koja u neopadajućem nizu **a[1]**, **a[2]**, …, **a[n]** određuje indeks elementa koji je jednak broju **b** metodom polovljenja. Ako ne postoji element jednak **b**, funkcija treba da vrati -1.
- Metoda polovljenja se realizuje na sledeći način: Poredi se **b** sa srednjim elementom niza (ili elementom oko sredine). Ako su jednaki, pretraživanje je završeno. Ako je **b** manje od srednjeg elementa, tada se pretraživanje nastavlja u levoj polovini niza, a suprotno u desnu. U izabranoj polovini se primenjuje isti algoritam.

Binarno pretraživanje - iterativno

```c
#include <stdio.h>
int binarySearch(int arr[], int l, int r, int x)
{
    while (l <= r)
    {
        int m = l + (r-l)/2;

        if (arr[m] == x)
            return m;

        if (arr[m] < x)
            l = m + 1;

        else
            r = m - 1;
    }

    return -1;
}

int main(void)
{
    int arr[] = {2, 3, 4, 10, 40};
    int n = sizeof(arr)/ sizeof(arr[0]);
    int x = 10;
    int result = binarySearch(arr, 0, n-1, x);
    (result == -1)? printf("Element is not present"
                           " in array")
                  : printf("Element is present at "
                           "index %d", result);
    return 0;
}
```

Binarno pretraživanje - rekurzivno

```c
#include <stdio.h>

int binarySearch(int arr[], int l, int r, int x)
{
    if (r >= l)
    {
        int mid = l + (r - l)/2;

        if (arr[mid] == x)
            return mid;

        if (arr[mid] > x)
            return binarySearch(arr, l, mid-1, x);

        return binarySearch(arr, mid+1, r, x);
    }
    return -1;
}

int main(void)
{
    int arr[] = {2, 3, 4, 10, 40};
    int n = sizeof(arr)/ sizeof(arr[0]);
    int x = 10;
    int result = binarySearch(arr, 0, n-1, x);
    (result == -1)? printf("Element is not present in array")
                  : printf("Element is present at index %d",
                                                  result);
    return 0;
}
```

Hvala na pažnji!