Основи програмирања 1
Лекција 8

др Зоран Бањац

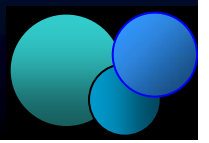
zoran.banjac@viser.edu.rs

Висока школа електротехнике и рачунарства
струковних студија Београд

Садржај

Низови

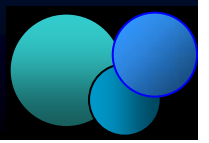
- Појава траженог карактера у стрингу
- Појава траженог подстринга у стрингу
- Проблеми код рада са стринговима
- Сортирање низова
- Претраживање низова



Појава карактера у стрингу

```
#include <string.h>
char * strchr(char *str1, int c);
```

- Тражи прву позицију карактера `c` у стрингу `str1`
- Повратна вредност:
  - Враћа показивач на позицују на којој се налази тражени карактер
  - NULL ако вредност није пронађена



Пример

```
#include <stdio.h>
#include <string.h>
int main ()
{
  char str1[] = "Ovo je test recenica";
  char * pch;

  printf ("Trazim karakter 'e' u \"%s\"\n",str1);

  pch = strchr(str1,'e');

  printf("Slovo 'e' je na %d. mestu\n", pch-str1+1);

}
```

имена низа је његова почетна адреса; тражи од те адресе

```
Trazim karakter 'e' u "Ovo je test recenica"
Slovo 'e' je na 6. mestu
```

Како наћи све позиције?

Пример

```
#include <stdio.h>
#include <string.h>
int main ()
{
  char str1[] = "Ovo je test recenica";
  char * pch;

  printf ("Trazim karakter 'e' u\"%s\"\n",str1);

  pch = strchr(str1,'e');

  while(pch != NULL)
  {
    printf("pronadjeno 'e' na poziciji %d\n",
                                    pch-str1+1);
    pch = strchr(pch + 1,'e');
  }
}
```

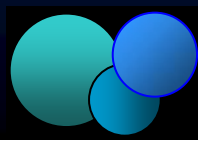
Тражи од ове адресе надаље

```
pronadjeno 'e' na poziciji
6 9 14 16
```

Позиција подстринга

```
#include <string.h>
char *strstr(char *str1, char *str2);
```

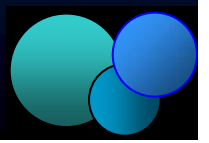
- Тражи позицију стринга str2 у стрингу str1
- Повратна вредност:
  - показивач на позицију на којој се налази први карактер str2 у стрингу str1
  - NULL ако нема поклапања



Пример

```
#include <stdio.h>
#include <string.h>
int main ()
{
 char str[ ] ="Ovo je jedan primer";
 char * pch;

 pch = strstr (str,"dan");
 printf("Pronadjena pozicija je %d\n", pch-str+1);

 return 0;
}
```

```
Pronadjena pozicija je 10
```

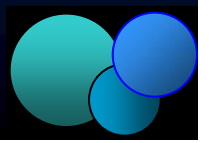
Пример

```
#include<stdio.h>
#include <string.h>
int main ()
{
 char str1[] = "opasan san o ... ";
 char str2[] = "san " ;
 char * pch;

 pch = strstr (str1, str2);
 while(pch)
 {
   printf("Pronadjena pozicija je %d\n",pch-str1+1);
   pch = strstr(pch+1, str2);
 }
 return 0;
}
```

```
Pronadjena pozicija je 4
Pronadjena pozicija je 8
```

Проблеми код рада са стринговима

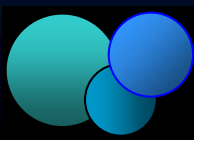
- Buffer overflow
  - настаје када програм покуша да упише податак изван меморије која је резервисана за одређени податак
  - сличан проблем настаје приликом покушаја читања података изван резервисане меморијске локације



Пример

- `char B[10];`

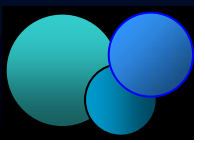
  `B[10] = x;`
- Низ почиње са индексом нула
- `B[10]` је 11-ти елемент низа
- Референциран је један бајт изван резервисане меморије за низ **B**
- Честа грешка!



strcpy

```
char b[20];
char a[20];
  ...
strcpy(b, a); /* (dest, source)*/
```

- Колика је величина стринга смештеног у низу `a`?
- Да ли је `a` заиста стринг (null-terminated)?
- Шта је резултат наредбе
  `strcpy(b, a);`
  ако није све онако како је предвиђено?

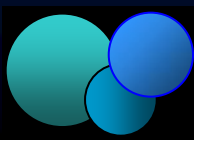


strcpy

- Безбедна употреба **`strcpy(dst, src)`**
- Поставити последњи карактер у стрингу (**`src`**) на **`NULL`**.
- Индекс последњег елемента у низу може да се одреди:

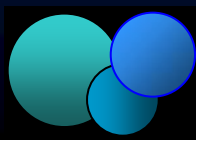
```
sizeof(src)/sizeof(src[0]) - 1
```

- Тиме се не нарушава правилно постављени стринг
- Проверити да ли једужина **`src`** мања или једнака од **`dst`**.



strcpy

- Који је највећи број карактера који смете да ископирате?
  - Да ли је одговор `sizeof(dst)`?
- Шта ће се десити ако се `src` не завршава са NULL карактером?
  - Не треба да се чита изван низа
- Решење:
  - `MIN(sizeof(dst), sizeof(src))-1`
- Да ли треба додати NULL карактер на крај `dst`?

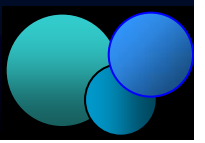


strncpy

- Где може да буде грешка?

```
char b[100];
strncpy(b, a, strlen(a));
```

- стринг **a** може да буде дужи од низа **b**
- треба да се допише **NULL** карактер на низ **b**

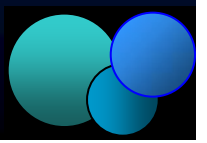


strcat

```
char * strcat(char * s, char * append);
```

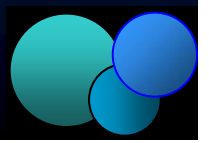
- стринг на који показује **append** се додаје на крај стринга **s**.
- Нема аутоматске провере величине!
- Пример решења:

```
if(sizeof(s)-strlen(s)-1 >= strlen(append) )
    strcat(s, append);
```

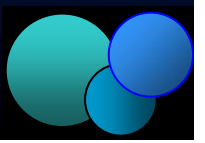
strlen

- Шта се дешава када се позове функција **`strlen`** а њен аргумент није стринг
- **`strlen`** претражује меморију од почетка низа све док не наиђе на NULL карактер (нулу).
  - Може да изађе из опсега низа
  - Последице: ...
- **`strlen`** захтева додатну контролу пре употребе



Сортирање низова

- Основни појмови
- Врсте сортирања



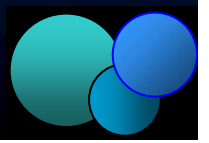
Основни појмови

- **Растући низ**
  Наредни елемент низа је већи од претходног
  `a[i+1] > a[i]`

- **Неопадајући низ**
  Наредни елемент низа је већи или једнак претходном
  `a[i+1] >= a[i]`

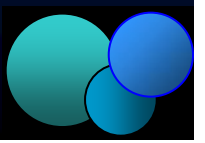
- **Опадајући низ**
  Наредни елемент низа је мањи од претходног
  `a[i+1] < a[i]`

- **Нерастући низ**
  Наредни елемент низа је мањи или једнак претходном
  `a[i+1] <= a[i]`



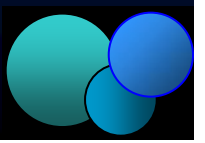
Сортирање низова

- Постављање вредности које садржи низ у одговарајући поредак
- Разлог:
  - Једноставнија употреба података
  - бржи приступ подацима
- Постоји више метода које се разликују по
  - сложености имплементације
  - броју операција које треба да обаве



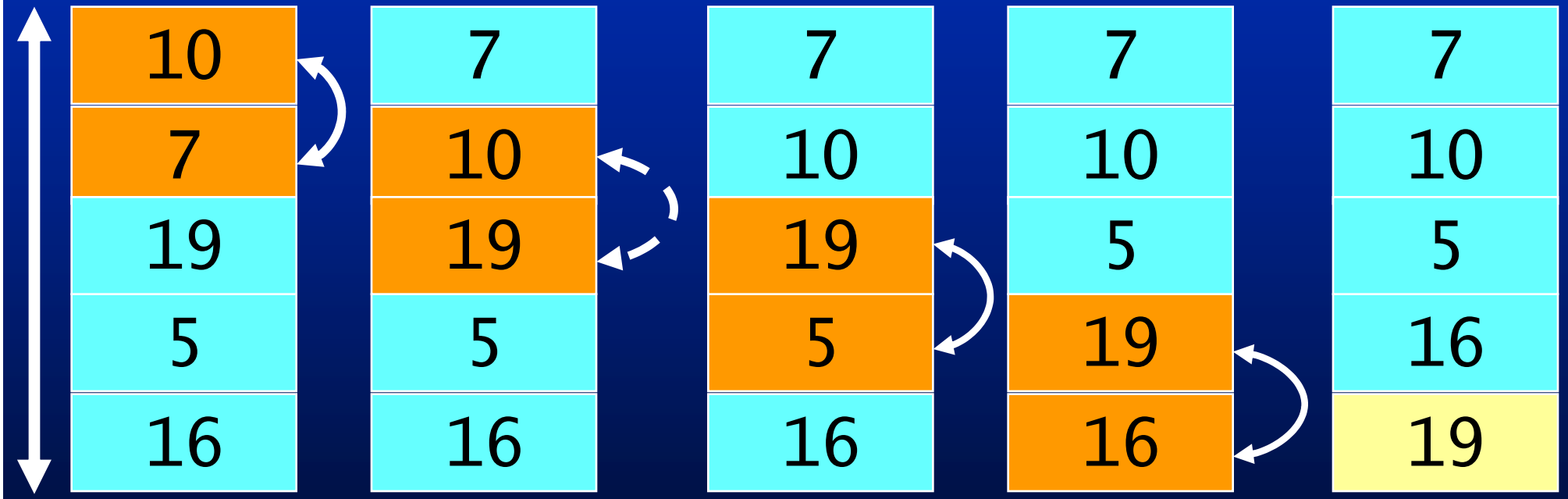
Метода *Bubble sort*

- Нека постоји низ од ***n*** елемената индекс ***0*** до ***n-1***
- У ***n-1*** корака се сортира низ
- Пореде се: **`niz[i]`** и **`niz[i+1]`**
- уколико је **`niz[i] > niz[i+1]`**
  - мењају места та два елемента низа
- Елементи који имају мању вредност се “крећу” ка мањем индексу...



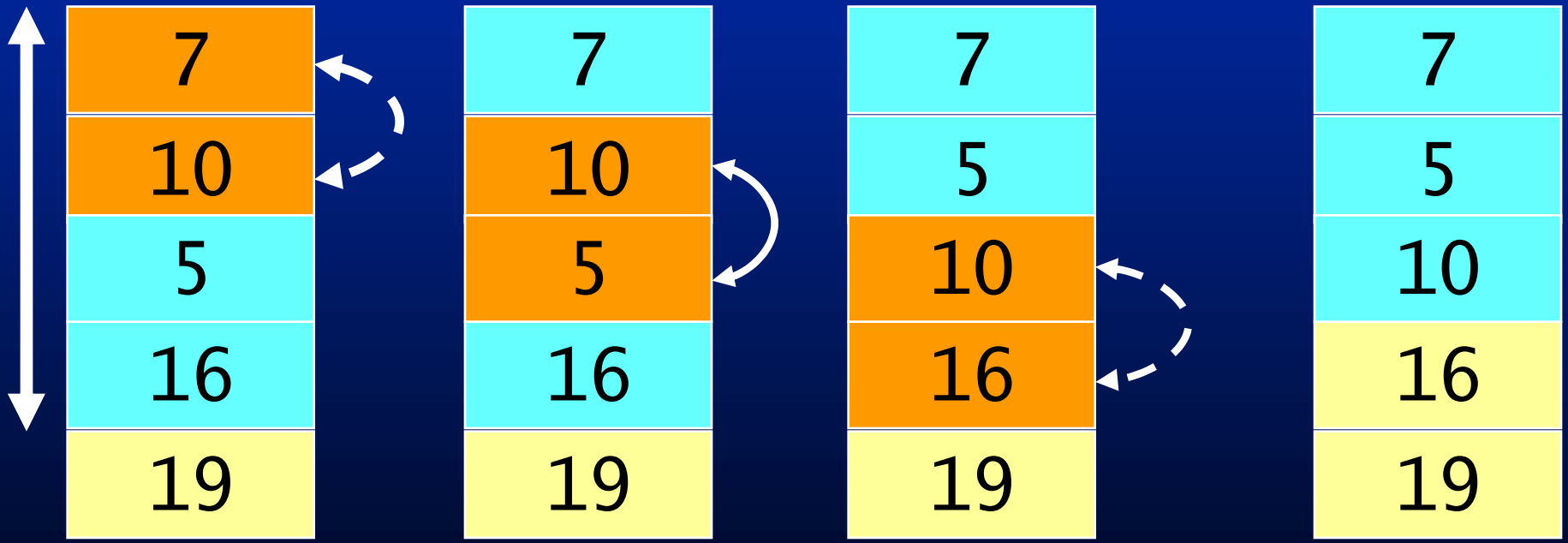
Пример *Bubble sort*

- Корак 1 од ***n-1*** **(*n* je 5)**
- Првих ***n-1*** итерација



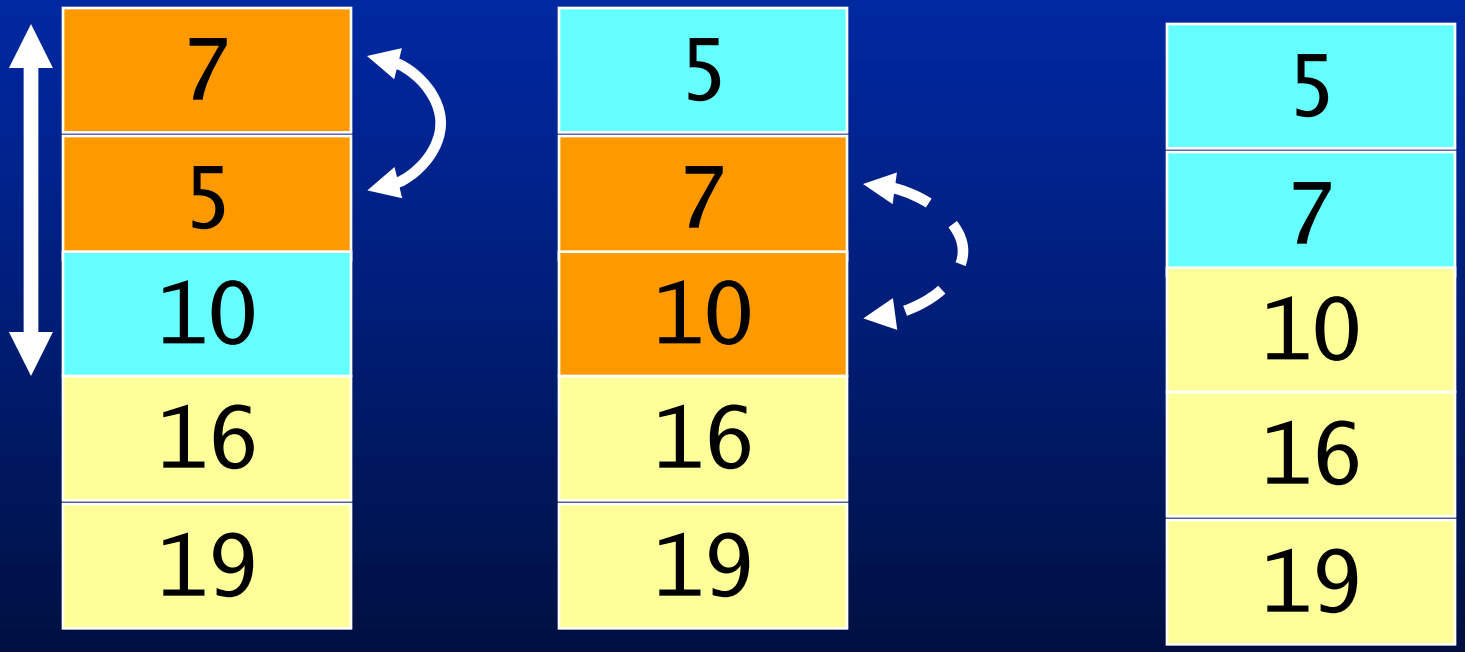
Пример *Bubble sort*

- Корак 2 од ***n-1*** **(*n* je 5)**
- Следећих ***n-2*** итерација



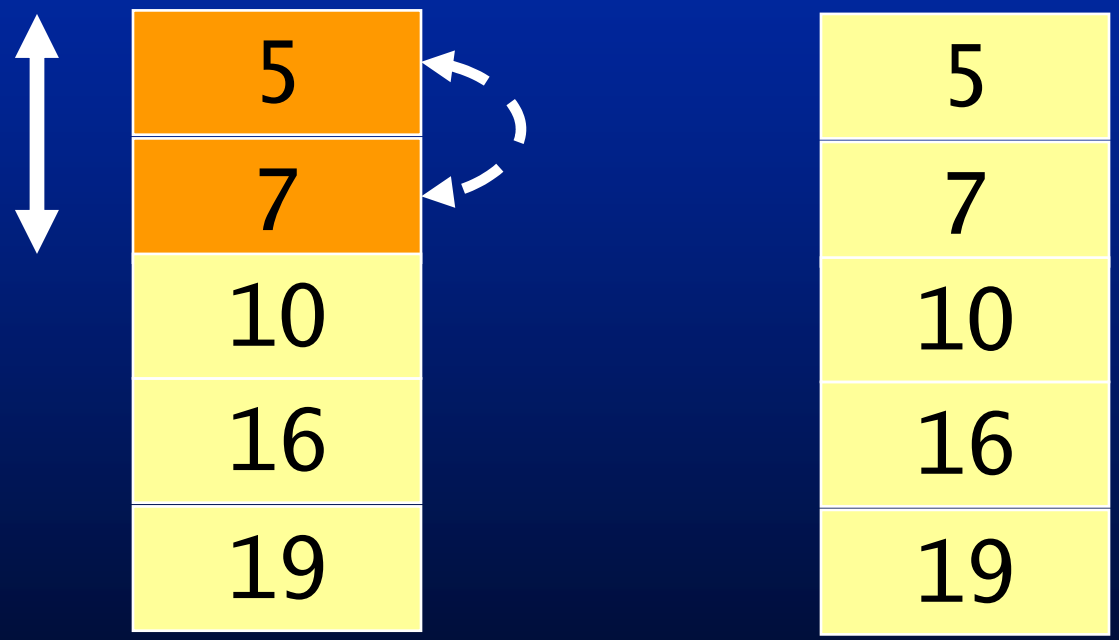
Пример *Bubble sort*

- Корак 3 од ***n-1*** **(*n* je 5)**
- Следећих ***n-3*** итерација



Пример *Bubble sort*

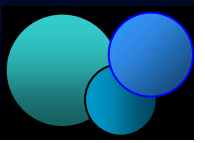
- Корак 4 од ***n-1*** **(*n* je 5)**
- Следећих ***n-4*** итерација



Bubble sort

- Укупан број корака: n-1
  - n димензија низа
  - Спољашња *for* петља

- Број итерација (поређење и замене) унутар корака:
  - први корак: n-1
  - други корак: n-2
  - ... ...
  - n-1 корак: 1
  - Унутрашња *for* петља (променљив број итерација)



Замена елемената низа

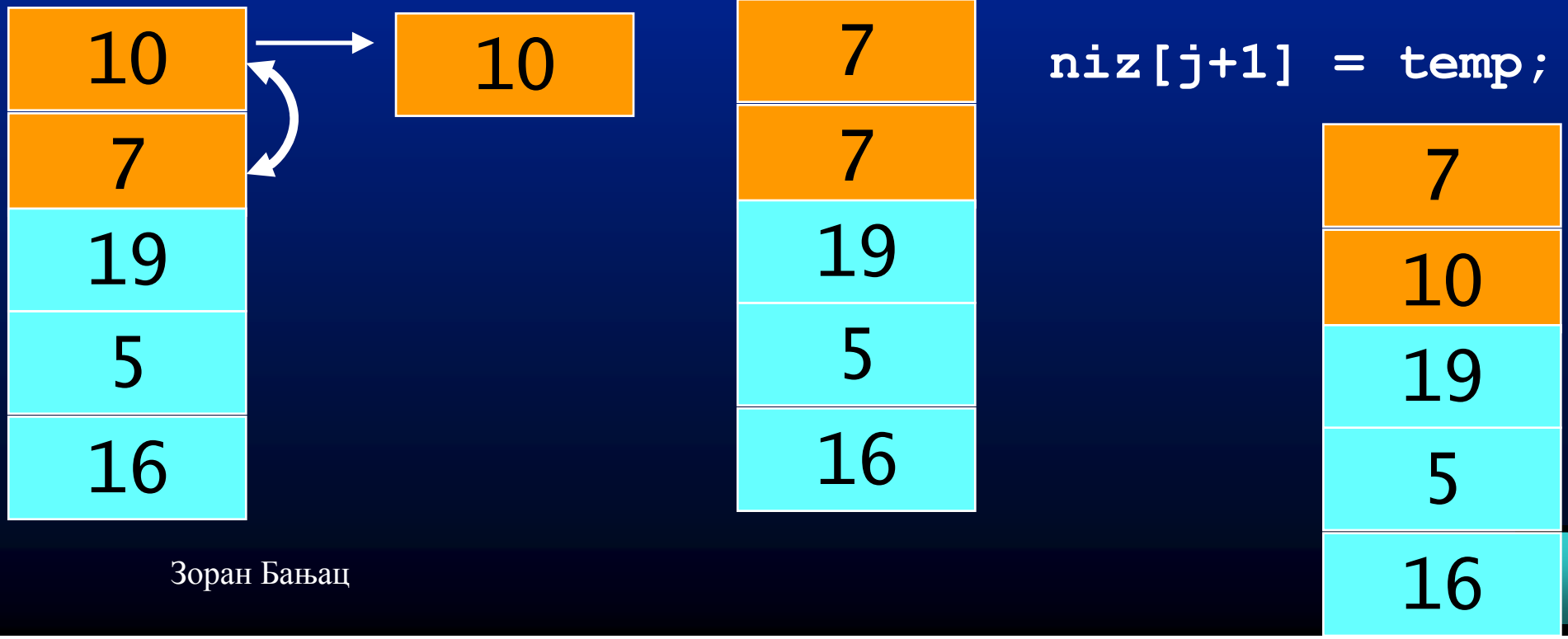
```
if(niz[j] > niz[j+1])
{
        temp = niz[j];
        niz[j] = niz[j+1];
        niz[j+1] = temp;
}
```

j=0

temp=niz[j]

niz[j] = niz[j+1];

niz[j+1] = temp;

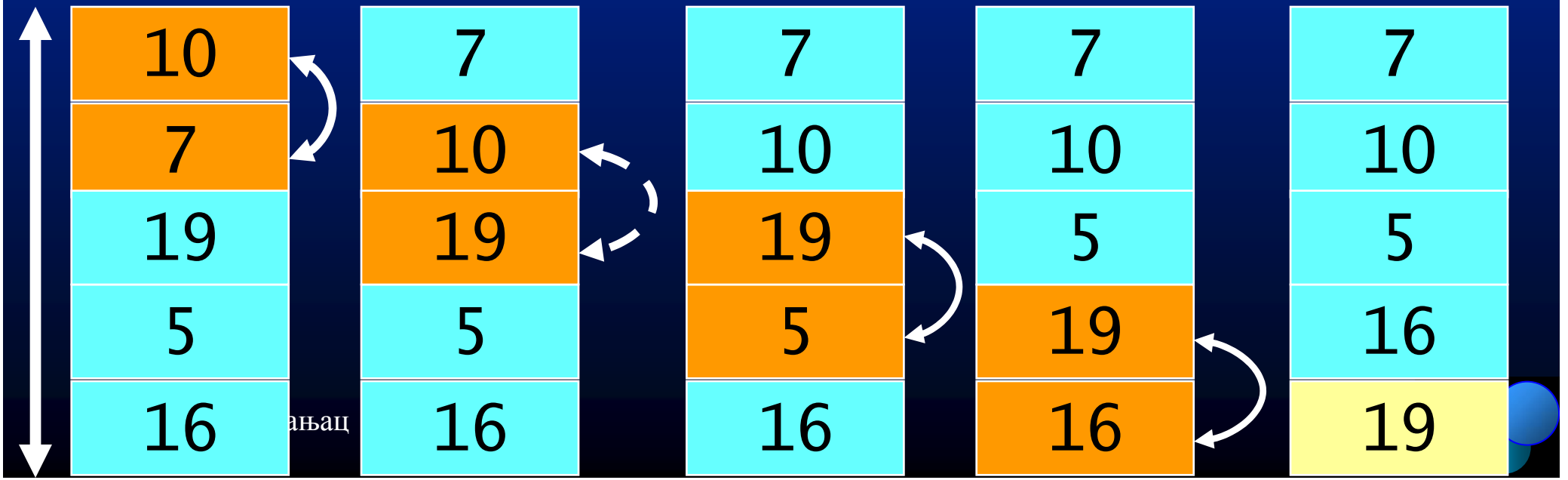


```
for(j=0; j < N-1; j++)
{
  if(niz[j] > niz[j+1])
  {
        temp = niz[j];
        niz[j] = niz[j+1];
        niz[j+1] = temp;
  }
}
```

Први корак
N-1

Други корак
N-2

Последњи корак
1



Пример *Bubble sort*

```
#define N 5
int main() /* rastuci - neopadajuci */
{
  int i, j, temp;
  int niz[N] = {10, 7, 19, 5, 16};

  for(i = 0; i < N-1; i++)
  {
    for(j = 0; j < N-1-i; j++)
      if(niz[j] > niz[j+1])
      {
        temp = niz[j];
        niz[j] = niz[j+1];
        niz[j+1] = temp;
      }
  }
}
```

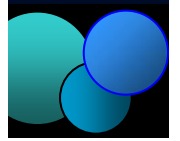
спољашња дужине N-1

замена места

унутрашња дужина се мења N-1-i.

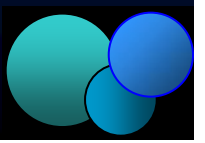
Пример *Bubble sort*

```
#define N 5
int main() /* opadajuci - nerastuci */
{
  int i, j, temp;
  int niz[N] = {10, 7, 19, 5, 16};

  for(i = 0; i < N-1; i++)
  {
    for(j = 0; j < N-1-i; j++)
      if(niz[j] < niz[j+1])
      {
        temp = niz[j];
        niz[j] = niz[j+1];
        niz[j+1] = temp;
      }
  }
}
```

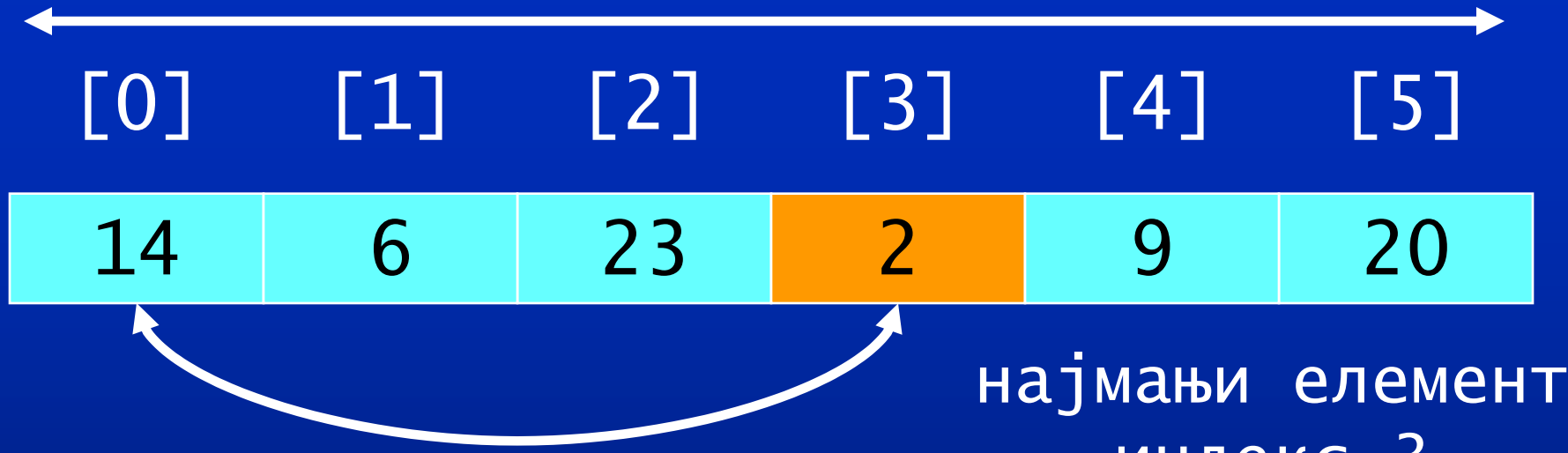


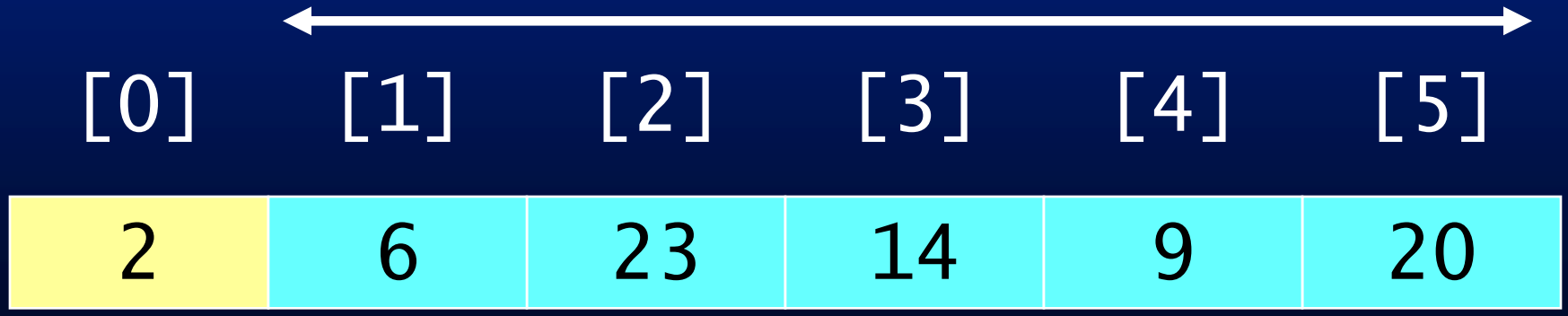
Метода избора (*selection*)

- Проналази најмањи елемент у низу и поставља га на најнижи индекс не сортираног дела листе.
- Први пут се нађе најмањи елемент у целом низу, постави се на индекс `0`
- Други пут се претражује од индекса `1`, поставља најмањи елемент на индекс `1`
- Трећи пут се претражује од индекса `2`
- ...

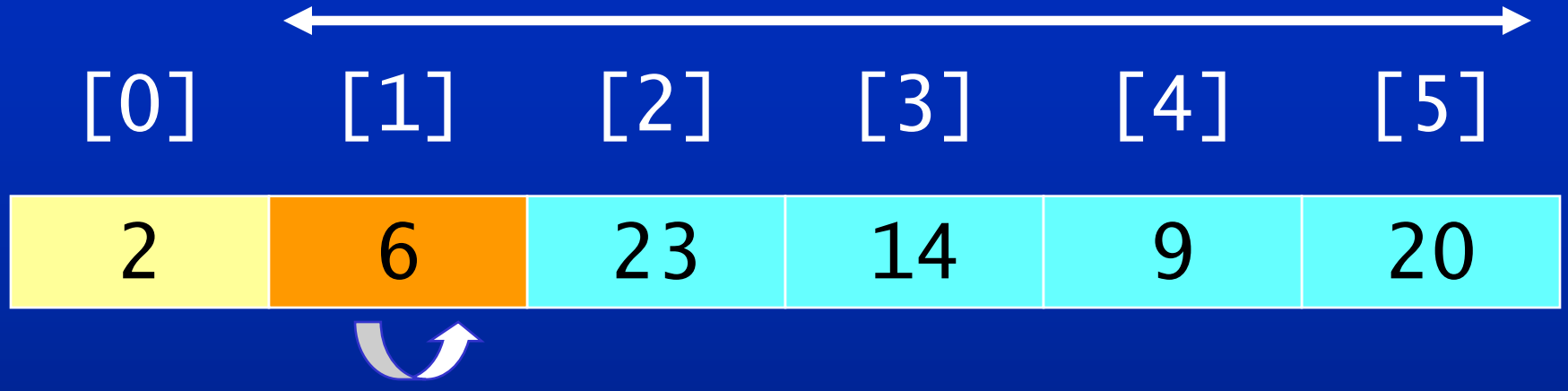


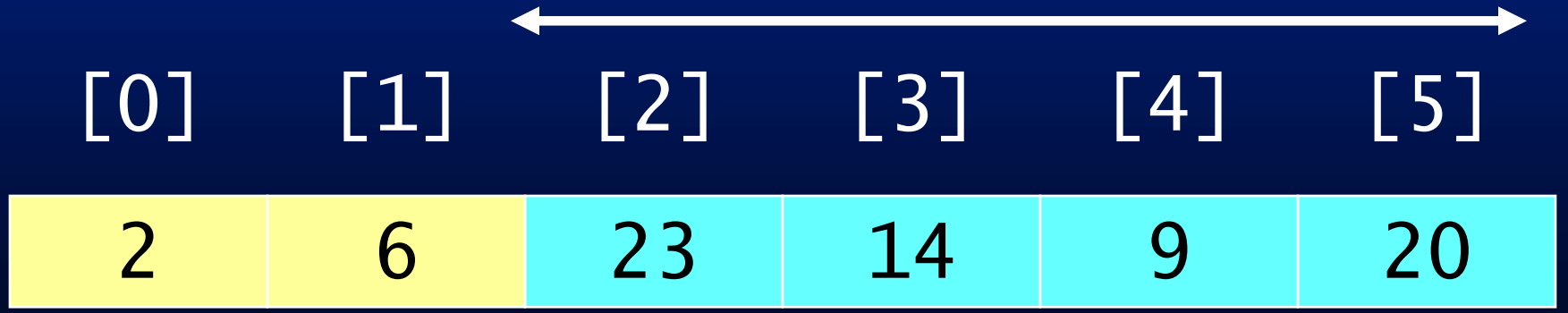
Пример методе избора



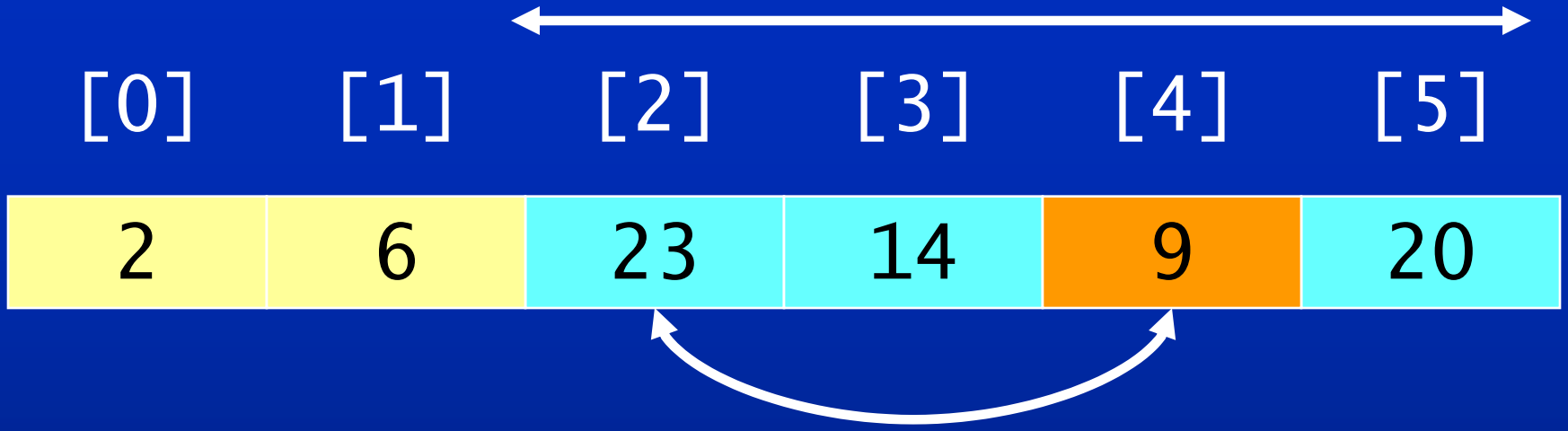


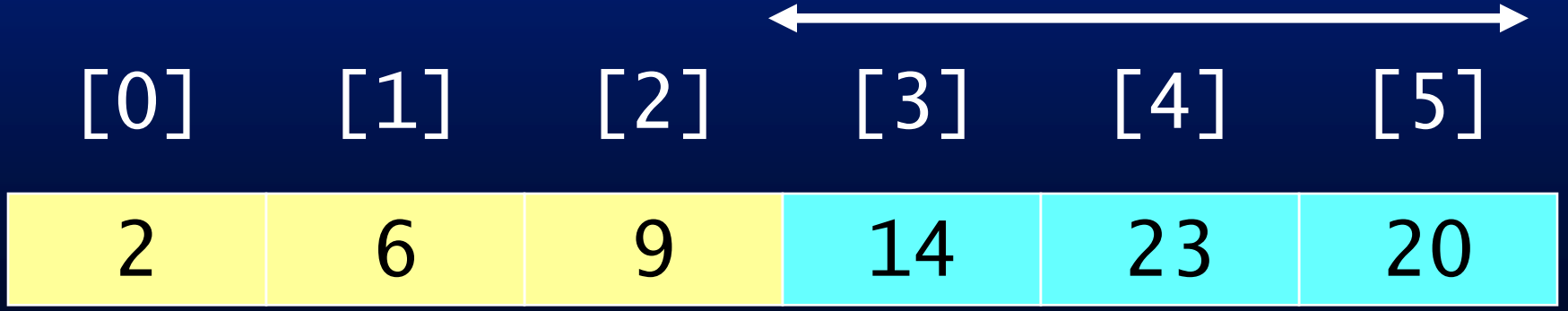
Пример методе избора



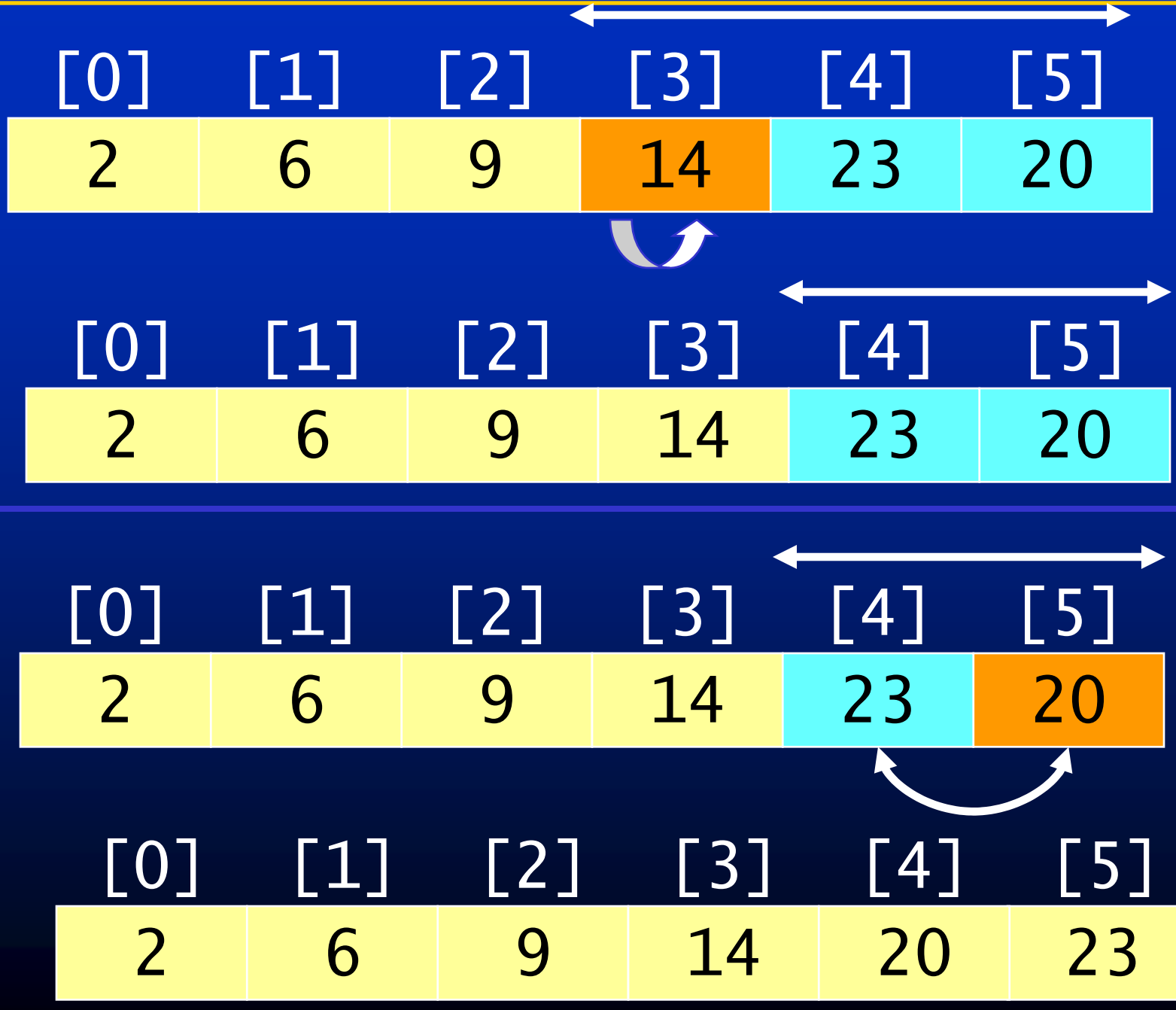


Пример методе избора





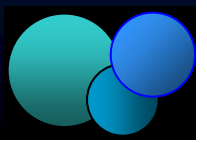
Пример методе избора



Минимални елемент дела низа

[0]	[1]	[2]	[3]	[4]	[5]
2	6	23	14	9	20

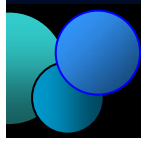
```
    min = niz[i];  /* min = 23 */
    minIndeks = i; /* i = 2 */


    for(j = i+1; j < DUZ; j++)
    {
      if(min > niz[j])
      {   min = niz[j];
          minIndeks = j;
      }
   }
/* min = 9, minIndeks = 4 */
```

Пример методе избора

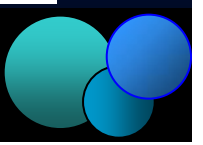
```
#define DUZ 6

int main() /* rastuci - neopadajuci */
{
  int i, j, min, minIndeks;
  int  niz[DUZ] = {14, 6, 23, 2, 9, 20};

  for(i = 0; i < DUZ-1; i++)
  {
     min = niz[i]; /* pretpostavka */
     minIndeks = i; /* zapamti indeks */

     for(j = i+1; j < DUZ; j++)
     {
       if(min > niz[j])
       {
          /*promena pretpostavke*/
          min = niz[j];
          minIndeks = j;
       }
     } /* kraj for */
```

минимални елемент дела низа



Пример методе избора

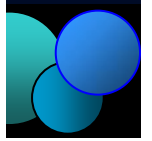
```
    /* zamena najmanjeg elementa sa onim na
       pocetku dela niza */
    niz[minIndeks] = niz[i];
    niz[i] = min;
  } /* kraj spoljasnje for petlje */

  for(i = 0; i < DUZ-1; i++)
    printf("%d ", niz[i]);

  printf("\n");

  return 0;
} /* kraj main */
```

Пример методе избора

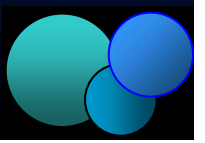
```
#define DUZ 6

int main() /* opadajuci - nerastuci */
{
  int i, j, max, maxIndeks;
  int  niz[DUZ] = {14, 6, 23, 2, 9, 20};

  for(i = 0; i < DUZ-1; i++)
  {
     max = niz[i]; /* pretpostavka */
     maxIndeks = i; /* zapamti indeks */

     for(j = i+1; j < DUZ; j++)
     {
       if(max < niz[j])
       {
          /*promena pretpostavke*/
          max = niz[j];
          maxIndeks = j;
       }
     } /* kraj for */
```

максимални елемент дела низа



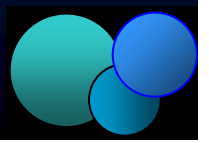
Пример методе избора

```
    /* zamena najmanjeg elementa sa onim na
       pocetku dela niza */
    niz[maxIndeks] = niz[i];
    niz[i] = max;
  } /* kraj spoljasnje for petlje */

  for(i = 0; i < DUZ-1; i++)
     printf("%d ", niz[i]);

  printf("\n");

  return 0;
} /* kraj main */
```

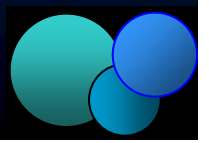
Претраживање низова

- Претраживање је веома чест захтев
  - код великих низова може бити временски захтеван
- Треба разликовати поступак претраживања
  - Несортираних низова
  - Сортираних низова



Линеарно претраживање

- Несортирани низови
- Мора се проћи кроз цео низ и проверити сваки елемент
- У најгорем случају је број провера једнак броју елемената низа



Линеарно претраживање несортирана листа

```
#define DUZ 6
int main()
{
 int i, trazenaVrednost = 23;
 int  niz[DUZ]={14, 6, 23, 2, 9, 20};

 for(i = 0; i < DUZ; i++)
    if(niz[i] == trazenaVrednost)
    {
      printf("Trazena vrednost se nalazi na %d.
                                        indeksu\n", i);
      break;
    }

 if(i == DUZ)
   printf("Trazena vrednost se ne nalazi u
                                          nizu\n");
}
```

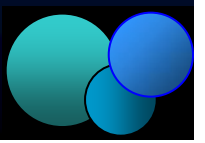
Линеарно претраживање сортирана листа

```
#define DUZ 6
int main()
{ int i, trazenaVrednost = 14;
  int  niz[DUZ]={5, 6, 9, 14, 20, 23};

  for(i = 0; i < DUZ; i++)
  {
    if(niz[i] >= trazenaVrednost)
      break;
  }

  if(i < DUZ && niz[i] == trazenaVrednost)
    printf("Trazeni broj se nalazi na %d.
                                    indeksu\n", i);
  else
    printf("Trazeni broj se ne nalazi u nizu\n");
}
```

Бинарно претраживање

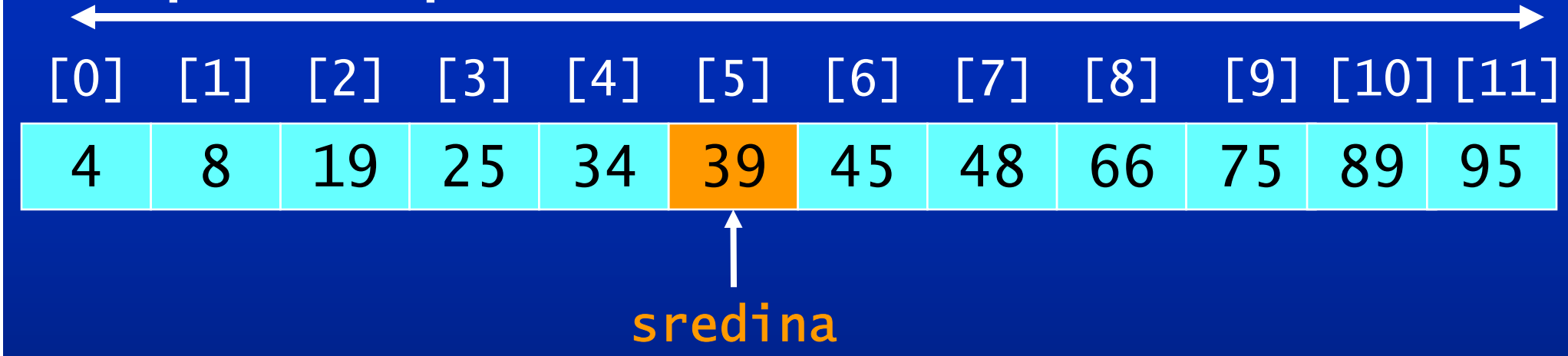
- Низ мора да буде сортиран

1. Тражи се индекс <= половини дужине низа
2. Пореди се елемент на том индексу са траженим
3. Ако је већи од траженог, наставља са претрага на доњем делу низа
4. У противном на горњем делу низа
5. Актуелни део низа дели на 2 дела
6. Понављати кораке од 2 до 5



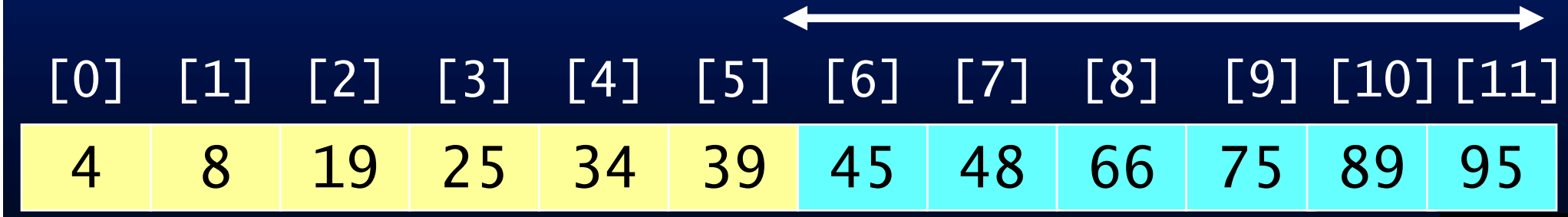
Бинарно претраживање

Тражена вредност 75

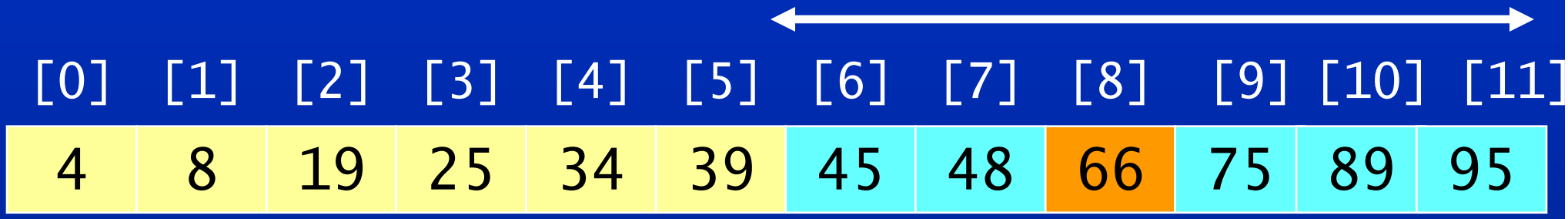


sredina = (prvi indeks + poslednji indeks)/2

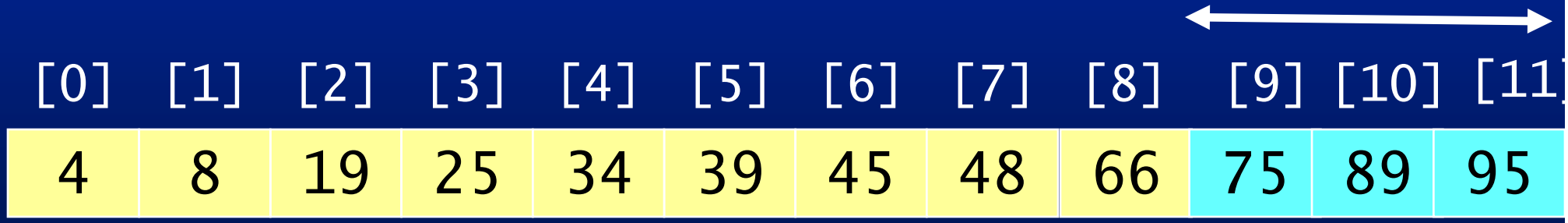
niz[5]< 75, горња половина низа

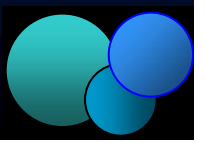


Бинарно претраживање



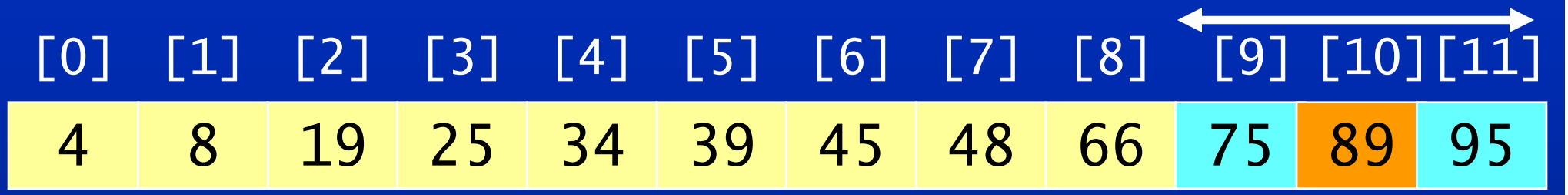

```
sredina = (prvi indeks +poslednji indeks)/2
```





Бинарно претраживање

```
sredina = (prvi indeks +poslednji indeks)/2 = 10
```



Бинарно претраживање

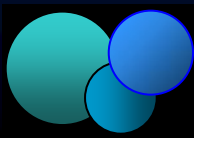
```
#include<stdio.h>
#define DUZ 12

int main()
{
  int prvi = 0;
  int poslednji = DUZ - 1;
  int srednji;
  int nadjen = 0;
  int trazenaVrednost = 75;
  int niz[DUZ]={4,8,19,25,34,39,45,48,66,75,89,95};
```

Бинарно претраживање

```
while(prvi <= poslednji && !nadjen)
{
  srednji=(poslednji + prvi)/2;

  if(niz[srednji] == trazenaVrednost)
     nadjen = 1;
  else if(niz[srednji] > trazenaVrednost)
     poslednji = srednji - 1;
  else
     prvi = srednji + 1;

}/* kraj while */
```

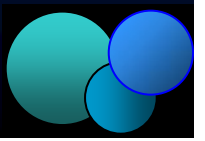
Бинарно претраживање

```
  if(nadjen)
     printf("Vrednost se nalazi na %d. indeksu\n",
             srednji);
  else
     printf("Vrednost se ne nalazi u nizu\n");
} /* kraj main */
```

Показивачи и низови

```
int my_array[6]={1, 23, 17, 4, -5, 100};
int *ptr;
ptr=&my_array[0]; /* pokazivac na prvi
                     element niza*/
```

Пример

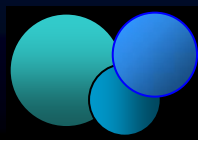
```
my_array[0]=  1     *(ptr + 0) =  1
my_array[1]= 23     *(ptr + 1) = 23
my_array[2]= 17     *(ptr + 2) = 17
my_array[3]=  4     *(ptr + 3) =  4
my_array[4]= -5     *(ptr + 4) = -5
my_array[5]=100     *(ptr + 5) = 100
```

```
#include <stdio.h>
int main()
{ /* ispis elemenata niza primenom pokazivaca */
 int my_array[6] = {1, 23, 17, 4, -5, 100};
 int *ptr;
 int i;

 ptr = &my_array[0]; /* prvi element niza*/
 printf("\n\n");
 for (i = 0; i < 6; i++)
 {
  printf("my_array[%d]=%3d\t",i,my_array[i]);
  printf("*(ptr + %d) =%3d\n",i,*(ptr + i));
 }
}
```

Пример *наст.*

```
for (i = 0; i < 6; i++)
 {
  printf("my_array[%d]=%3d\t", i, my_array[i]);
  printf("*(ptr + %d) = %3d\n", i, *(ptr + i));
 }
```

```
my_array[0] =   1    *(ptr + 0) =   1
my_array[1] =  23    *(ptr + 1) =  23
my_array[2] =  17    *(ptr + 2) =  17
my_array[3] =   4    *(ptr + 3) =   4
my_array[4] =  -5    *(ptr + 4) =  -5
my_array[5] = 100    *(ptr + 5) = 100
```

Показивачи и низови

- У програмском језику *C* је дозвољено да се уместо наредбе

```
ptr = &my_array[0];
```

користи наредба

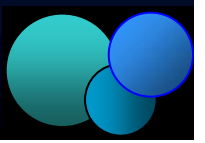
```
ptr = my_array;
```

- Име низа (без навођења индекса) је адреса првог елемента низа.

- погрешно је написати

```
my_array = ptr;    /* greska! */
```

јер је **`ptr`** променљива а **`my_array`** је константа



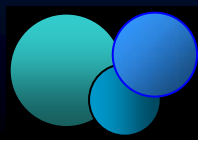
Показивачи и низови

- Показивач може да садржи адресу било ког елемента низа а не само првог.

```
ptr = &my_array[2];
ptr = &my_array[0] + 2;
ptr = my_array + 2;
```

што је исто као и односно

- јер је `my_array` исто као и `&my_array[0]`



Пример

```
int main()
{
 int a[8] = {1,23,17,4,-5,100,55,44};
 int *pa, *pb;
 int x, y;
 pa=&a[4];  /* pa ukazuje na a[4] tj -5 */
 x=*(pa+3); /* x=a[7]=44 */
 y=*pa+3;   /* y=a[4]+3=-2*/
 x=*pa++;   /* x=*pa ili x=-5
               potom pa++ ili pa = pa + 1
               pa sadrži adresu od a[5] */
 y=(*pa)++; /* y=a[5], a[5]=101 */
 pb=&a[2];
}
```

Сортирање 2*D* низа (низ стрингова)

```
#include <stdio.h>
#include<string.h>
#define BROJ_IMENA 30
#define DUZINA_IMENA 80

int main( )
{
  char spisak[BROJ_IMENA][DUZINA_IMENA+1];
  char pom_ime[DUZINA_IMENA+1];
  int i, j, n = 0;

  /* Poruka korisniku*/
  printf("Unesite ime, potom ENTER\n");
  printf("Za manje od 30 imena poslednji unos ");
  printf("treba da bude: dve kose crte //\n");
```


Сортирање 2*D* низа (низ стрингова)

```
/* prihvat imena */
do
{
  gets(pom_ime);
  if(! strcmp(pom_ime, "//"))
     break;
  strcpy(spisak[n],pom_ime);
  n++;
}while( n < BROJ_IMENA );
```


Сортирање 2*D* низа (низ стрингова)

```
/*Sortiranje po abecednom redu*/
   printf("\nNiz imena, sortiran:\n");

    for(i = 0; i < n-1; i++)
        for(j = i+1; j < n; j++)
          if(strcmp(spisak[i], spisak[j]) > 0)
          {
           strcpy(pom_ime, spisak[i]);
           strcpy(spisak[i], spisak[j]);
           strcpy(spisak[j], pom_ime);
          }
  /*Prikaz sortiranog niza imena*/
    for(i=0; i<n; i++)
      puts(spisak[i]);

    printf("\n");
} /* kraj funkcije main*/
```

Хвала на пажњи

Питања?