آرایه‌ها ۳

مبانی برنامه‌نویسی

(۱۳۹۹-۱۳-۱۱)

جلسه‌ی بیست‌وسوم

دانشگاه شهید بهشتی

پاییز ۱۳۹۹

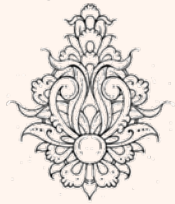
دانشکده‌ی مهندسی برق و کامپیوتر

احمد محمودی ازناوه

http://faculties.sbu.ac.ir/~a_mahmoudi/

فهرست مطالب

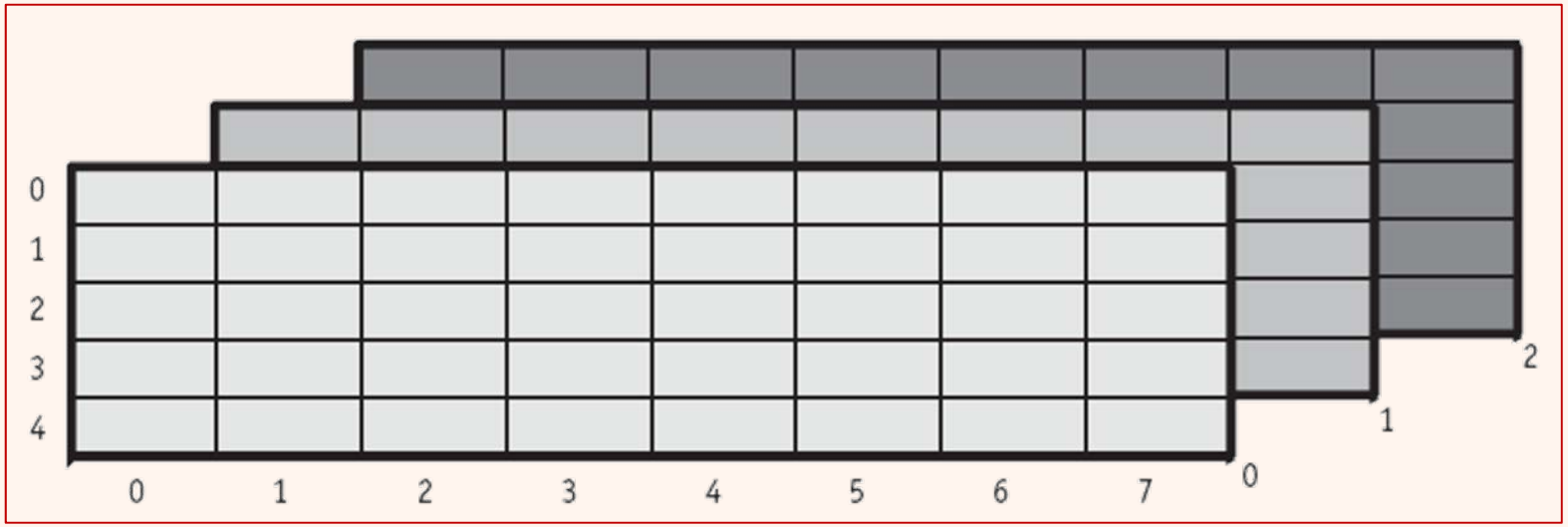
- ارسال آرایه‌ی چندبعدی به تابع
- روش‌های جستجو و مرتب‌سازی



آرایه‌های چندبعدی

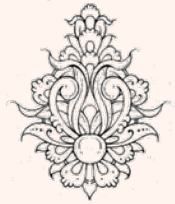
- C++ اجازه می‌دهد آرایه‌ی سه یا بیشتر بعدی تعریف کنیم.

```
double  seat[3][5][8];
```



نکات آرایه‌های چندبعدی

- int a[3][5]; آرایه‌ای با سه عنصر تعریف می‌کند که هر عنصر، خود یک آرایه‌ی پنج عنصری از نوع int است. این یک آرایه‌ی دو بعدی است که در مجموع پانزده عضو دارد.
- دستور int a[2][3][5]; آرایه‌ای با دو عنصر تعریف می‌کند که هر عنصر، سه آرایه است که هر آرایه پنج عضو از نوع int دارد. این یک آرایه‌ی سه بعدی است که در مجموع سی عضو دارد.
- **آرایه‌های چند بعدی مثل آرایه‌های یک بعدی به توابع فرستاده می‌شوند با این تفاوت که هنگام اعلان و تعریف تابع مربوطه، باید تعداد عناصر بُعد دوم تا بُعد آخر حتما ذکر شود.**

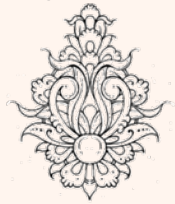


نکات آرایه‌های چندبعدی (ادامه...)

- برای ارسال آرایه‌ی سه‌بعدی به تابع می‌باید دو بعد آخر مشخص گردد.

```
// Function prototype
void displaySeats(double [][5][8], int);

// Function header
void displaySeats(double array[][5][8], int numGroups);
```



مثال

```
int numZeros(int a[][4][3],int n1,int n2,int n3);
int main()
{
    int a[2][4][3] = { { {5, 0, 2}, {0, 0, 9}, {4, 1, 0}, {7, 7, 7} },
                       { {3,0,0}, {8,5,0}, {0,0,0}, {2,0,9} }
};
cout << "This array has " << numZeros(a,2,4,3) << " zeros:\n";
}

int numZeros(int a[][4][3],int n1,int n2,int n3)
{ int count = 0;
for (int i = 0; i < n1; i++)
   for (int j = 0; j < n2; j++)
      for (int k = 0; k < n3; k++)
            if (a[i][j][k] == 0) ++count;
return count;
}
```

```
This array has 11 zeros:
```

تمرین کلاسی

تابعی بنویسید که یک آرایه‌ی ۵×۵ به عنوان آرگومان ورودی دریافت و مجموع عناصر قطر اصلی را بازگرداند.

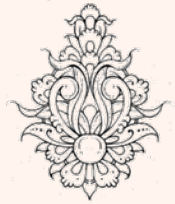
```
int SumDiag(int mat[][5]){
    int sum=0;
    for(int i=0;i<5;i++)
        sum+=mat[i][i];
    return sum;
}
```

تابعی بنویسید که یک آرایه‌ی ۵×۵ به عنوان آرگومان ورودی دریافت و تعیین کند (مقدار bool برگرداند) این ماتریس پایین مثلثی است یا نه.

```
bool isLowerTriangle(int mat[][5]){
    for(int i=0;i<5;i++)
        for(int j=i+1;j<5;j++)
            if(mat[i][j]!=0)
                return false;
    return true;
}
```

جستجوی خطی

- آرایه‌ها اغلب برای پردازش یک زنجیره از داده‌ها به کار می‌روند.
- **جستجوی خطی** به روشی گفته می‌شود که از اولین عنصر آرایه شروع کنیم و یکی یکی همه‌ی عناصر آرایه را جستجو می‌نماییم تا دریابیم مقدار مورد جستجو در کدام عنصر قرار گرفته است.

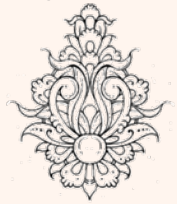


جستجوی خطی (ادامه...)

```
#include <iostream>
using namespace std;
int Lsearch(int,int[],int);

int main(){
  int a[] = { 2, 43, 616, 88, 464, 6, 55};
  cout << "index(88,a,7) = " << Lsearch(88,a,7) << endl;
  cout << "index(55,a,7) = " << Lsearch(55,a,7) << endl;
}
int Lsearch(int x, int a[], int n){
  for (int i=0; i<n; i++)
    if (a[i] == x)
      return i;
  return -1;    // x not found
}
```

```
index(88,a,7) = 3
index(55,a,7) = 6
```

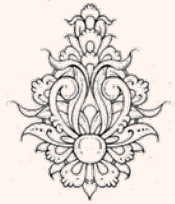


جستجوی دودویی

در روش **جستجوی دودویی** به یک آرایه‌ی مرتب نیاز است.

- هنگام جستجو آرایه از وسط به دو بخش بالایی و پایینی تقسیم می‌شود.
  - مقدار مورد جستجو با آخرین عنصر بخش پایینی مقایسه می‌شود.
  - اگر این عنصر کوچک‌تر از مقدار جستجو بود، مورد جستجو در بخش پایینی وجود ندارد پس باید در بخش بالایی به دنبال آن گشت.
- دوباره بخش بالایی به دو بخش تقسیم می‌گردد و گام‌های بالا تکرار می‌شود.
- در نهایت محدوده‌ی جستجو به یک عنصر محدود می‌شود که یا آن عنصر با مورد جستجو برابر است و عنصر مذکور یافت شده و یا این که آن عنصر با مورد جستجو برابر نیست و لذا مورد جستجو در آرایه وجود ندارد.

این روش پیچیده‌تر از روش جستجوی خطی است اما بسیار سریع‌تر به جواب می‌رسیم.



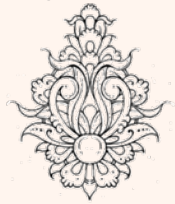
جستجوی دودویی (ادامه...)

```
#include <iostream>
using namespace std;

int Bsearch(int,int[],int);
int main(){  int a[] = { 2, 43, 88, 188, 464, 567, 655};
cout << "index(464,a,7) = " << Bsearch(464,a,7) << endl;
cout << "index(500,a,7) = " << Bsearch(500,a,7) << endl;
}

int Bsearch(int x, int a[], int n){     // binary search:
  int l=0, h=n-1, i;
  while (l <= h){
    i = (l + h)/2;           // the average of l and h
    if (a[i] == x)
      return i;
    if (a[i] < x)
      l = i+1;   // continue search in a[i+1..h]
    else
      h = i-1;              // continue search in a[0..i-1]
  }
  return -1;                    // x was not found in a[0..n-1]
}
```

```
index(464,a,7) = 4
index(500,a,7) = -1
```

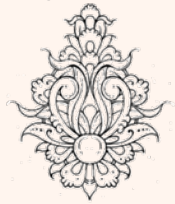


جستجوی دودویی بازگشتی

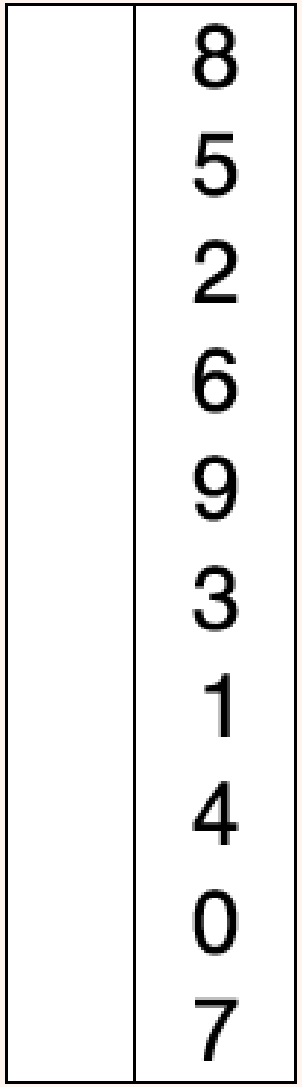
```
int Bsearch(int,int[],int,int);
int main(){
    int a[] = { 2, 43, 88, 188, 464, 567, 655};
    cout << "index(464,a,0,6) = " << Bsearch(464,a,0,6) << endl;
    cout << "index(500,a,0,6) = " << Bsearch(500,a,0,6) << endl;}

int Bsearch(int x, int a[], int l, int h){  // binary search:
    if(l <= h){
        int i = (l + h)/2;
        if (a[i] == x)
            return i;
        if (a[i] < x)
            return Bsearch(x,a,i+1, h);
        else
            return Bsearch(x,a,l,i-1);
    }
    else
        return -1;
}
```

```
index(464,a,0,6) = 4
index(500,a,0,6) = -1
```



مرتب‌سازی انتخابی (صعودی)



- در این روش ابتدا اولین عنصر، با تمامی عنصرها مقایسه و کوچک‌ترین عنصر در نخستین خانه‌ی آرایه قرار می‌گیرد.
- سپس این روند، برای بقیه‌ی آرایه تکرار می‌شود.

```
void selectionsort(int k[],int l){
  int i,j,temp;
  for(i=0;i<l-1;i++)
    for(j=i+1;j<l;j++)
      if(k[i]>k[j]){
        temp=k[j];
        k[j]=k[i];
        k[i]=temp;
      }
}
```

wikipedia

مرتب‌سازی انتخابی (ادامه...)

```
int main(int)
{
    const int n=10;
    int i,a[n];
    cout<<"enter ter integer number:\n";
    for(i=0;i<n;i++)
    {
        cout<<i+1<<"th:";
        cin>>a[i];
        cout<<"\n";
    }
    selectionSort(a,n);
    for(i=0;i<n;i++)
        cout<<a[i]<<"\t";
}
```

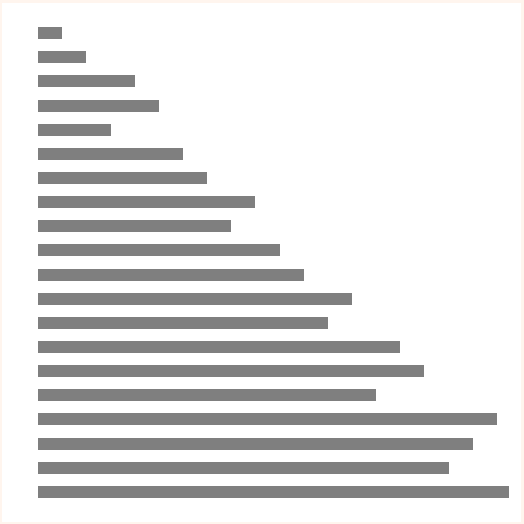
```
void selectionSort(int k[],int len)
{
    int Min,MinInd;
    for(int i=0;i<len-1;i++)
    {
        Min=k[i];
        MinInd=i;
        for(int j=i+1;j<len;j++)
            if(Min>k[j])
            {
                Min=k[j];
                MinInd=j;
            }
        int tmp=k[i];
        k[i]=Min;
        k[MinInd]=tmp;
    }
}
```

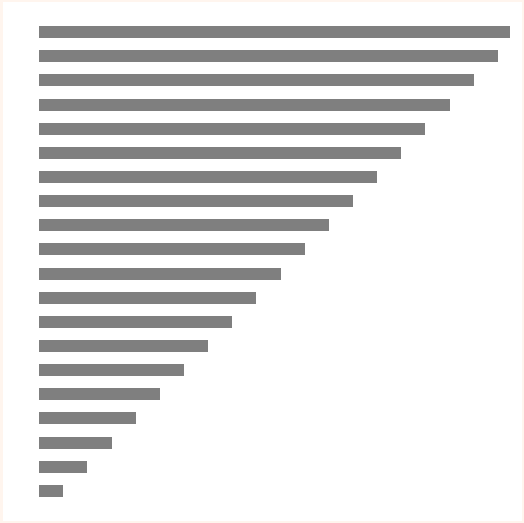
```
1th:8
2th:6
3th:4
4th:2
5th:1
6th:3
7th:9
8th:7
9th:5
10th:0
```

```
0    1    2    3    4    5    6    7    8    9
```

http://www.sorting-algorithms.com/

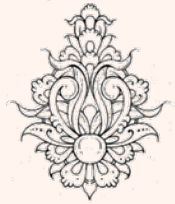
مرتب‌سازی انتخابی (ادامه...)





مرتب سازی حبابی

- «مرتب‌سازی حبابی» یکی از ساده‌ترین الگوریتم‌های مرتب‌سازی است.
- در این روش، آرایه چندین مرتبه پویش می‌شود و عناصر متوالی با هم مقایسه می‌شوند.
- در هر مرتبه کوچک‌ترین عنصر موجود به سمت بالا هدایت می‌شود، سپس از محدوده‌ی مرتب‌سازی برای مرتبه‌ی بعدی یکی کاسته می‌شود.
- در پایان تمامی آرایه مرتب شده است.



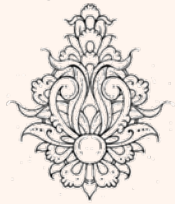
شیوه مرتب‌سازی حبابی

1) آخرین عنصر آرایه با عنصر قبل از خود مقایسه می‌شود.
2) اگر کوچک‌تر بود، جای این دو با هم عوض می‌شود.
3) سپس عنصر ماقبل آخر با عنصر قبل از خود مقایسه می‌شود.
4) اگر عنصر دوم بزرگ‌تر بود، جای این دو با هم عوض می‌شود.

- و به همین ترتیب مقایسه و جابه‌جایی زوج‌های همسایه ادامه می‌یابد تا وقتی به ابتدای آرایه رسیدیم، کوچک‌ترین عضو آرایه در خانه‌ی نخست قرار خواهد گرفت.
- در ادامه محدوده‌ی جستجو یکی کاسته می‌شود.
- به همین ترتیب زوج‌های کناری یکی یکی مقایسه می‌شوند تا عدد کوچک‌تر بعدی به جایگاه خود منتقل شود. این پویش ادامه می‌یابد تا این که وقتی محدوده جستجو به عنصر آخر محدود شد، آرایه مرتب شده است.

مثال

- برنامه‌ای بنویسید که یک آرایه ۱۰ عنصری را از ورودی دریافت کند، آن را به یک تابع مرتب‌سازی ارسال نماید و بعد از مرتب‌سازی آن را در تابع اصلی چاپ نماید.



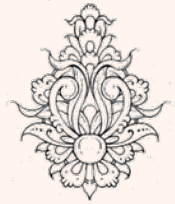
پاسخ

```
#include <iostream>
using namespace std;
void bubblesort(int a[],int);

int main(int){
   const int n=10;
   int i,a[n];
   cout<<"enter ten integer number
   for(i=0;i<n;i++){
      cout<<i+1<<"th:";
      cin>> a[i];
      cout<<"\n";
   }
   bubblesort(a,n);
   for(i=0;i<n;i++)
      cout<<a[i]<<"\
}
```

```
void bubblesort(int k[],int l){
   int i,j,temp;
   for(i=0;i<l-1;i++)
      for(j=l-1;j>i;j--)
         if(k[j]<k[j-1]){
            temp=k[j];
            k[j]=k[j-1];
            k[j-1]=temp;
         }
}
```

```
enter ten integer number:
1th:1
2th:4
3th:7
4th:4
5th:55
6th:8
7th:4
8th:1
9th:9
10th:33
```

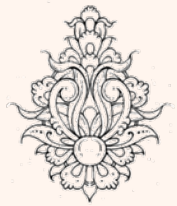


1 1 4 4 4 7 8 9 33 55

مرتب‌سازی حبابی (ادامه...)

```
void bubblesort(int k[],int l){
     int i,j,temp;
     bool isSorted=false;
     for(i=0;i<l-1 && !isSorted;i++){
          isSorted=true;
          for(j=l-1;j>i;j--)
               if(k[j]<k[j-1]){
                    isSorted=false;
                    temp=k[j];
                    k[j]=k[j-1];
                    k[j-1]=temp;
               }
     }
}
```

بدین ترتیب برای آرایه‌های مرتب شده این کار را تا پایان ادامه نمی‌دهد.



http://www.sorting-algorithms.com/

مرتب‌سازی حبابی (ادامه...)