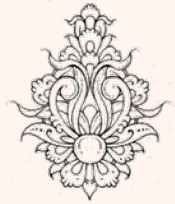
مبانی برنامه‌نویسی
(۱۳۹۹-۱۳-۱۱)
جلسه‌ی بیست و پنجم

آرایه‌ها ۳
روش‌های جستجو

دانشگاه شهید بهشتی
پاییز ۱۳۹۲
دانشکده‌ی مهندسی برق و کامپیوتر
احمد محمودی ازناوه

فهرست مطالب

- مروری بر جلسه‌ی پیش
- آرایه‌های چند بعدی
- ارسال آرایه‌ی چندبعدی به تابع
- روش‌های جستجو



هیستوگرام فراوانی

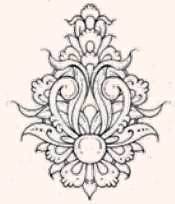

```
 #include <iostream>
 using namespace std;
int main()
{
const int arraySize = 10;
int n[ arraySize ]={12,3,4,6,13,16,2,7,0,9}; // array s has 10 elements



   cout << "Element\t" << "Value\t"<< "Histogram" << endl;


   for ( int i= 0; i < arraySize; i++ )
       {
          cout << i << "\t" << n[ i ]<<"\t";
             for ( int j = 0; j < n[i]; j++ )
                cout<< '*';
          cout << endl;
       }
   return 0; // indicates successful termination
} // end main
```

```
Element Value   Histogram
0       12      ************
1       3       ***
2       4       ****
3       6       ******
4       13      *************
5       16      ****************
6       2       **
7       7       *******
8       0
9       9       *********
```

مثال

```
#include <iostream>
using namespace std;
int sum(int[],int);
int main()
{   int a[] = { 11, 33, 55, 77 };
    int size = sizeof(a)/sizeof(int);
    cout << "sum(a,size) = " << sum(a,size) << endl;
     }
int sum(int a[], int n)
{   int sum=0;
    for (int i=0; i<n; i++)
        sum += a[i];
    return sum;
}
```

```
sum(a,size) = 176
```



```
#include <iostream>
using namespace std;
void read(int[],int&);
void print(int[],int);
const int MAXSIZE=100;
int main()
{
     int a[MAXSIZE]={0},size;
     read(a,size);
     cout << "The array has " << size << " elements: ";
     print(a,size);
}
```

```
void print(int a[],int n)
{
     for (int i=0; i<n; i++)
     cout << a[i] << " ";
}
```

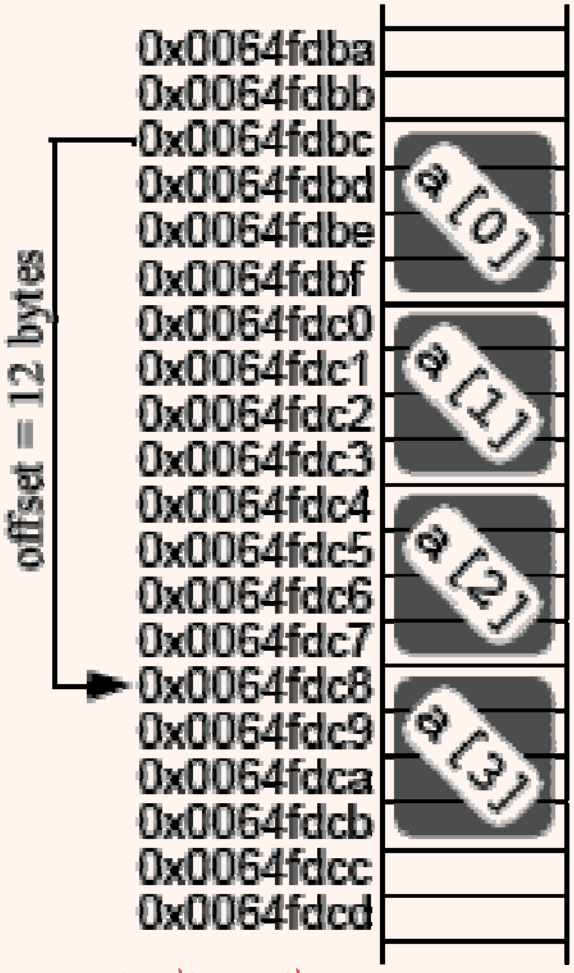
```
Enter integers. Terminate with 0:
a[0]: 12
a[1]: 3
a[2]: 4
a[3]: 5
a[4]: 6
a[5]: 0
The array has 5 elements: 12 3 4 5 6
```

```
void read(int a[],int& n)
{
     cout << "Enter integers. Terminate with 0:\n";
     n = 0;
do
{
     cout << "a[" << n << "]: ";
     cin >> a[n];
} while (a[n++] != 0 && n < MAXSIZE);
--n; // don't count the 0
}
```

آدرس عناصر آرایه

```
cin >> a[n];
```

- اگر مقدار n برابر با ۳ باشد، به مقدار ۳×۴ از a[0] جلورفته، به آدرس مورد نظر دسترسی پیدا می‌کنیم.



نام آرایه در حقیقت آدرس ابتدای آرایه است که آدرسی ثابت و غیرقابل تغییر است

Defined data types (typedef)

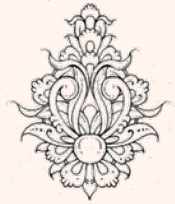
به وسیله‌ی typedef می‌توان نام انواع موجود را به نوع‌هایی جدید تغییر نام داد:

```
typedef  existing_type  new_type_name;
```

با استفاده از typedef نوع جدیدی به‌وجود نمی‌آید. بلکه نام دیگری برای نوعی که‌وجود دارد در نظر گرفته می‌شود

```
typedef  int  integer;
typedef  unsigned int  uint;
```

- در این صورت
  - برنامه به صورت خواناتر کدنویسی می‌شود.
  - اگر در آینده بخواهیم نوع داده‌ای را عوض نماییم کار ساده‌تر می‌شود.
  - برای داده‌هایی با اسامی طولانی و پیچیده کاربرد دارد.



مثال

```
typedef int arrayType[];
// Function prototypes
void doubleArray(arrayType, int);
void showValues(arrayType, int);
int main()
{
const int ARRAY_SIZE = 7;
arrayType set = {1, 2, 3, 4, 5, 6, 7};
cout << "The arrays values are:\n";

showValues(set, ARRAY_SIZE);
doubleArray(set, ARRAY_SIZE);

cout << "\nAfter calling doubleArray, the values are:\n";
showValues(set, ARRAY_SIZE);
cout << endl;
return 0;
}
```

```
The arrays values are:
1 2 3 4 5 6 7

After calling doubleArray, the values are:
2 4 6 8 10 12 14
```

```
void showValues (arrayType nums, int size)
{
for (int index = 0; index < size; index++)
  cout << nums[index] << " ";
cout << endl;
}

void doubleArray(arrayType nums, int size)
{
for (int index = 0; index < size; index++)
  nums[index] *= 2;
}
```

ارسال آرایه همانند ارسال از طریق ارجاع عمل می‌کند پس می تواند مقادیر را تغییر دهد

چند تابع

```
double sumArray(double[], int);
double getHighest(double[], int);
double getLowest(double[], int);
int main()
{
const int NUM_DAYS = 5;
double sales[NUM_DAYS],
total, average, highest, lowest;
cout << "Enter the sales for this week.\n"
for (int day = 0; day < NUM_DAYS; day++)
{
    cout << "Day " << (day + 1) <<": ";
   cin >> sales[day];
}
// Get total sales and compute average sale
total = sumArray(sales, NUM_DAYS);
average = total / NUM_DAYS;
// Get highest and lowest sales amounts
highest = getHighest(sales, NUM_DAYS);
lowest = getLowest(sales, NUM_DAYS);

// Display results
cout << "The total sales are $" <<  total << e
cout << "The average sales amount is $"  << av
cout << "The highest sales amount is $"  << hi
cout << "The lowest sales amount is $"  << low
return 0;
}
```

```
double sumArray(double array[], int size)
{
double total = 0; // Accumulator

for (int count = 0; count < size; count++)
total += array[count];
return total;
}
```

```
 double getHighest(double array[], int size)
{
double highest = array[0];
for (int count = 1; count < size; count++)
{ if (array[count] > highest)
highest = array[count];
}
return highest;
}
```

```
double getLowest(double array[], int size)
{
double lowest = array[0];
for (int count = 1; count < size; count++)
{ if (array[count] < lowest)
lowest = array[count];
}
return lowest;
}
```

```
Enter the sales for this week.
Day 1: 23
Day 2: 4
Day 3: 5
Day 4: 41
Day 5: 2
The total sales are $75
The average sales amount is $15
The highest sales amount is $41
The lowest sales amount is $2
```

نحوه‌ی تعریف آرایه‌های n بعدی

نوع آرایه نام آرایه [طول بعد اول] [طول بعد دوم] ..... [طول بعد nام]

- **تعریف آرایه دو بعدی از نوع صحیح:**
- int a[2][5];
- **یک آرایه سه بعدی از نوع صحیح:**
- int k[3][2][5];
- **میزان حافظه‌ی مصرفی یک آرایه n بعدی:**

میزان حافظه مصرفی= sizeof (نوع آرایه) طول بعد اول x طول بعد دوم x ... xطول بعد nام

برای مثال نخست، میزان حافظه مورد نیاز را حساب کنید:

Sizeof(int)*2*5=40

آرایه‌ی دوبعدی

- آرایه‌ی دو بعدی از کنار هم گذاردن تعدادی آرایه‌ی یک بعدی به دست می‌آید.

	Column 0	Column 1	Column 2	Column 3
Row 0	score[0][0]	score[0][1]	score[0][2]	score[0][3]
Row 1	score[1][0]	score[1][1]	score[1][2]	score[1][3]
Row 2	score[2][0]	score[2][1]	score[2][2]	score[2][3]

```
double score[3][4];
```

Rows Columns

	Column 0	Column 1
Row 0	8	5
Row 1	7	9
Row 2	6	3

```
int hours[3][2] = {{8, 5}, {7, 9}, {6, 3}};
```

```
int hours[3][2] = {8, 5, 7, 9, 6, 3};
```

انواع مقداردهی

- آرایه دو بعدی:

```
int a[2][3]={{3,1,2},{3,5,7}};
```

- ننوشتن بعد اول:

```
int a[ ][3]={{3,1,2},{3,5,7}};
```

```
int table[3][2] = {{1}, {3, 4}, {5}};
```

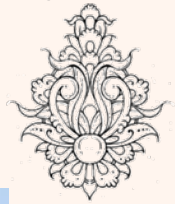
- باقی عناصر اتوماتیک صفر می‌شوند.
- آرایه سه بعدی:

```
int [2][3][4]={{{1,2,3,4},{3,2,3,4},{3,2,2,1}},{{2,3,5,6},{1,2,4,2},{1,2,2,1}}}
```

مثال

```
int main()
{
   const int NUM_ROWS = 3; // Number of rows
   const int NUM_COLS = 5; // Number of columns
   int total = 0; // Accumulator
   int numbers[NUM_ROWS][NUM_COLS] = {{2, 7, 9, 6, 4},
   {6, 1, 8, 9, 4},
   {4, 3, 7, 2, 9}};
   // Sum the array elements
   for (int row = 0; row < NUM_ROWS; row++)
   { for (int col = 0; col < NUM_COLS; col++)
   total += numbers[row][col];
   }
   // Display the sum
   cout << "The total is " << total << endl;
    return 0;
}
```

```
The total is 81
```

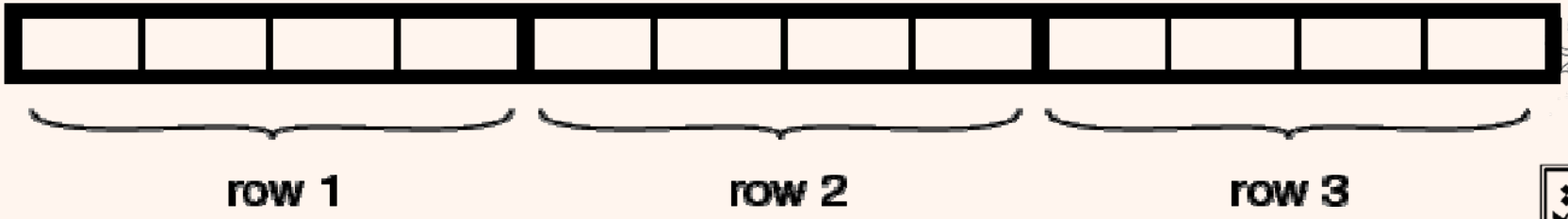


ارسال آرایه‌ی دو بعدی به تابع

- برای ارسال یک آرایه‌ی دو بعدی به تابع می‌باید ستون ها مشخص باشد.

```
void showArray(int array[][NUM_COLS],int numRows)
```

- در c++ ساختاری که یک آرایه‌ی دوبعدی قرار می‌گیرد به صورت زیر است:



- در این حالت هر ردیف پس از ردیف دیگر قرار می‌گیرد.

مثال

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

 const int NUM_COLS = 4; // Number of columns in each array
 const int TBL1_ROWS = 3; // Number of rows in table1
 const int TBL2_ROWS = 4; // Number of rows in table2

 void showArray(int [][NUM_COLS], int); // Function prototype

 int main()
 {
 int table1[TBL1_ROWS][NUM_COLS] = {{1, 2, 3, 4},
 {5, 6, 7, 8},
 {9, 10, 11, 12}};
 int table2[TBL2_ROWS][NUM_COLS] = {{ 10, 20, 30, 40},
 { 50, 60, 70, 80},
 { 90, 100, 110, 120},
 {130, 140, 150, 160}};

 cout << "The contents of table1 are:\n";
 showArray(table1, TBL1_ROWS);
 cout << "\nThe contents of table2 a
 showArray(table2, TBL2_ROWS);
 return 0;
 }
```

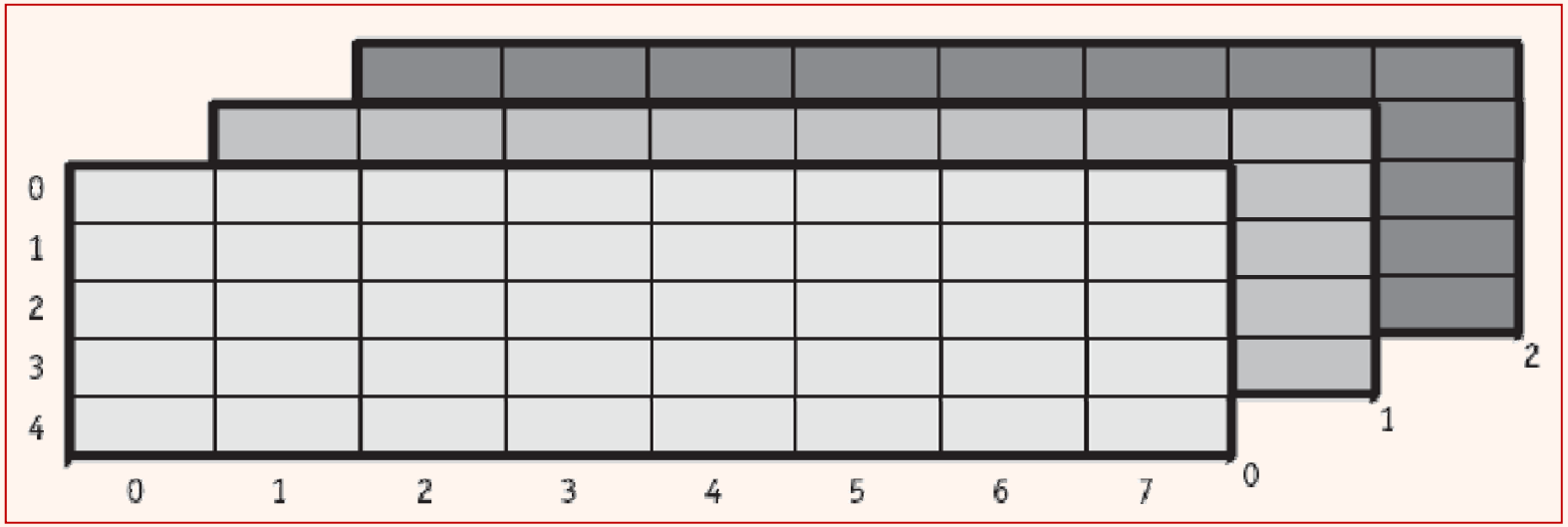
```
The contents of table1 are:
    1     2     3     4
    5     6     7     8
    9    10    11    12

The contents of table2 are:
   10    20    30    40
   50    60    70    80
   90   100   110   120
  130   140   150   160
```

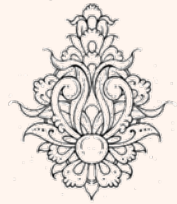
```
void showArray(int array[][NUM_COLS], int numRows)
{
for (int row = 0; row < numRows; row++)
 {
     for (int col = 0; col < NUM_COLS; col++)
      {
        cout << setw(5) << array[row][col] << " ";
      }
cout << endl;
  }
}
```

آرایه‌های چندبعدی

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

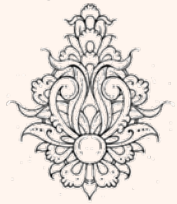
```
double  seat[3][5][8];
```





نکات آرایه‌های چندبعدی

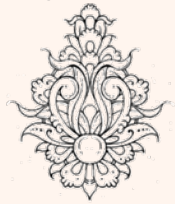
- `int a[3][5];` آرایه‌ای با سه عنصر تعریف می‌کند که هر عنصر، خود یک آرایه‌ی پنج عنصری از نوع int است. این یک آرایه‌ی دو بعدی است که در مجموع پانزده عضو دارد.
- دستور `int a[2][3][5];` آرایه‌ای با دو عنصر تعریف می‌کند که هر عنصر، سه آرایه است که هر آرایه پنج عضو از نوع int دارد. این یک آرایه‌ی سه بعدی 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 به‌عنوان آرگومان ورودی دریافت و مجموع عناصر قطر اصلی را برگرداند.

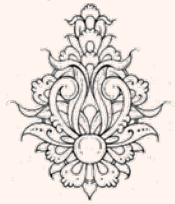
```
int SumDiag(int mat[][5]){
    int sum=0;
    for(int i=0;i<5;i++)
        sum+=mat[i][i];
    return sum;
}
```

تابعی بنویسید که یک آرایه‌ی ۵×۵ به‌عنوان آرگومان ورودی دریافت و تعیین کند (مقدار bool برگرداند) این ماتریس پایین

```
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

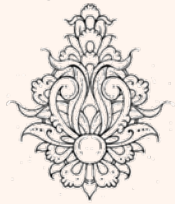
- آرایه‌ها اغلب برای پردازش یک زنجیره از داده‌ها به کار می‌روند.
- **جستجوی خطی** به روشی گفته می‌شود که از اولین عنصر آرایه شروع کنیم و یکی یکی همه‌ی عناصر آرایه را جستجو می‌نماییم تا دریابیم مقدار مورد جستجو در کدام عنصر قرار گرفته است.



جستجوی خطی (ادامه...)

```cpp
#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
```

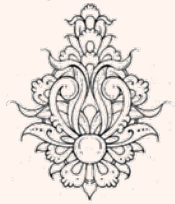


جستجوی دودویی

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.

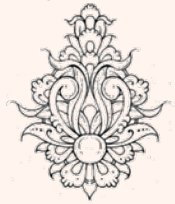
این روش پیچیده‌تر از روش جستجوی خطی است اما بسیار سریع‌تر به جواب می‌رسیم.



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
```