# EE-163

## Computers and Programming

## FE Electrical

Lesson 08

Arrays

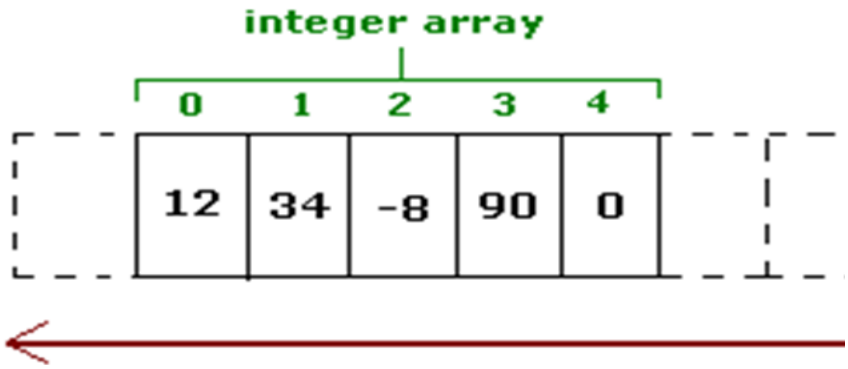
## Array (definition)

An array is a list of variables of a certain data type having a si
name, defined contiguously in the memory.

In C language an array is also called a subscripted variable
obvious reasons. It is the very first step towards data structur

## Graphical Representation of an Array

An integer array of 5 elements would look like this in men
The memory locations occupied by the array are numbered




## Declaring an Array

C++ language has five (5) basic variable data types. Since an array is a list of variables it can also be defined as any one of them namely; integer, float, character, long and double.

## 1) Array declaration and initialization

```
1   #include<iostream>
2   using namespace std;
3
4   int main(void)
5   {   //array declaration
6      int num1[5];
7
8      //another declaration method
9      const int SIZE=3;
10     int num2[SIZE];
11
12     //array declaration with initialization
13     int num3[5]={12,2,54,7,100};
14
15     //declaration with incomplete initialization
16     int num4[5]={12,2};
17
18     //displaying array contents
19     cout<<num1[0]<<" "<<num1[1]<<" "<<num1[2]
20        <<" "<<num1[3]<<" "<<num1[4]<<endl;
21     cout<<num2[0]<<" "<<num2[1]<<" "<<num2[2]
22        <<endl;
23     cout<<num3[0]<<" "<<num3[1]<<" "<<num3[2]
24        <<" "<<num3[3]<<" "<<num3[4]<<endl;
25     cout<<num4[0]<<" "<<num4[1]<<" "<<num4[2]
26        <<" "<<num4[3]<<" "<<num4[4]<<endl;
27     return 0;
28  }
```

## 2) Some more declaration and initialization

```
1  #//array declaration and initialization
2  #include<iostream>
3  using namespace std;
4  int main(void)
5  {  //array declaration
6     char name[5];
7
8     //another declaration method
9     const int SIZE=3;
10    char pet[SIZE];
11    //array declaration with initialization
12    char vowels[5]={'a','e','i','o','u'};
13    //declaration with incomplete initialization
14    char consonants[21]={'b','c'};
15    //displaying array contents
16    cout<<name[0]<<" "<<name[1]<<" "<<name[2]
17       <<" "<<name[3]<<" "<<name[4]<<endl;
18    cout<<pet[0]<<" "<<pet[1]<<" "<<pet[2]
19       <<endl;
20    cout<<vowels[0]<<" "<<vowels[1]<<" "<<vowels[2]
21       <<" "<<vowels[3]<<" "<<vowels[4]<<endl;
22    cout<<consonants[0]<<" "<<consonants[1]<<" "<<
23       " "<<consonants[3]<<" "<<consonants[4]<<endl;
24    return 0;
25 }
```

**The general method for defining an array is:**

**datatype arrayname[no. of elements]**

**Points to note here: 1) no. of elements is always constant and can't be taken from user.**

**2) Usually no. of elements are given as a #define directive.**
**Example: #define LIM 3**

**3) C++ never stops the programmer from exceeding the array limit (eg. cin>>num[5]), so you need to be cautious regarding that**

## 3) Array initialization, input and processing

```
1   #include<iostream>
2   #include<conio2.h>
3   using namespace std;
4   #define SIZE 6
5
6   int main(void)
7   {
8      // Declaring arrays
9      int subjectscores[SIZE];
10     float weeklytemperature[7];
11     char name[25];
12     // similarly long and double can be defined
13     // NOTE: strings are not arrays
14     // Initializing arrays
15     int monthlyprofits[12]=
16                          {1200000,6434564,5564664,5456456,5467787,9788445,4456
17                         ,39996534,8745647,3142355,6574347,89676};
18     float subjectpercentages[6]={88.5,85.2,85.0,79.5,81.0,75.1};
19     char vowels[5]={'a','e','i','o','u'};
20     // use curly brackets containing values separated by comma
21
22     // taking input in an array from location 0 to 5 (total 6 elements)
23     cout<<"\nEnter element no.0 :";
24     cin>>subjectscores[0]; // scanning is just like any variable
25     cout<<"\nEnter element no.1 :"; // but using the appropriate index no.
26     cin>>subjectscores[1];
27     cout<<"\nEnter element no.2 :";
28     cin>>subjectscores[2];
29     cout<<"\nEnter element no.3 :";
30     cin>>subjectscores[3];
31     cout<<"\nEnter element no.4 :";
32     cin>>subjectscores[4];
33     cout<<"\nEnter element no.5 :";
34     cin>>subjectscores[5];
35
```

```
36 // Some manipulation/processing with array:
37    // adding 2 to each element
38    subjectscores[0]=subjectscores[0]+2;
39    subjectscores[1]=subjectscores[1]+2;
40    subjectscores[2]=subjectscores[2]+2;
41    subjectscores[3]=subjectscores[3]+2;
42    subjectscores[4]=subjectscores[4]+2;
43    subjectscores[5]=subjectscores[5]+2;
44
45    // printing the processed array
46    cout<<"\nElement 0 = "<<subjectscores[0];
47    cout<<"\nElement 1 = "<<subjectscores[1];
48    cout<<"\nElement 2 = "<<subjectscores[2];
49    cout<<"\nElement 3 = "<<subjectscores[3];
50    cout<<"\nElement 4 = "<<subjectscores[4];
51    cout<<"\nElement 5 = "<<subjectscores[5];
52    getch();
53    return 0;
54 }
```

## Initializing an Array

Initializing an array means declaring it and assigning some initial values to it. This can be done easily by following the syntax shown in the following program.

```
#define LIM 3
int main(void)
{
float num[LIM]={12.9, 9.0, 986.89}; // Declaration of an array
char ch[]={'a','b','c','d'}; // Declaration of char array, no. of
                              //  elements not assigned
```

# Practical programming with arrays

**Practically speaking we exploit one special property of arrays to work with them. As the array index – the no. written inside square brackets - is always going to be an integer value, we define a separate integer variable and write it within the square brackets (example: num[index]). This empowers us to manipulate the array by changing this index variable.**

**Once the index variable is defined, the best way to manipulate it is through a loop. The following example uses for() loops to perform operations on an array.**

4) Processing arrays with for() loop

```
1  #include<iostream>
2  #include<conio2.h>
3  #include<ctime>
4  #include<cstdlib>
5  using namespace std;
6
7  #define LIM 10
8
9  int main(void)
10 {
11    // variable definition
12    // 10 element float array, numbered 0 to 9
13    float num[LIM]={12.0,3.3,7.6,3.1,9.8,7.9,3.4,8.7,9.0,1.0};
14    int index_count;// an integer variable to access different array locations
15    srand(time(NULL));
16  //Automatic initialization with even numbers
17    for(index_count=0;index_count<LIM;index_count++)
18    {
19       num[index_count]=(index_count+1)*2;
20    }
21
```

```
22  cout<<"\nPress any key to continue";
23   getch();
24
25   // loop for printing output
26   for(index_count=0;index_count<LIM;index_count++)
27   {
28      cout<<"\nElement no."<<index_count<<" = "<<num[index_count];
29   }
30   cout<<endl<<endl;
31
32   //Automatic initialization with random numbers
33   for(index_count=0;index_count<LIM;index_count++)
34   {
35      num[index_count]=(rand()%100)+1;
36   }
37
38   cout<<"\nPress any key to continue";
39   getch();
40
41   // loop for printing output
42   for(index_count=0;index_count<LIM;index_count++)
43   {
44      cout<<"\nElement no."<<index_count<<" = "<<num[index_count];
45   }
46   cout<<endl<<endl;
47
48   // taking array input through for() loop: index_count runs from 0 to 9
49   for(index_count=0;index_count<LIM;index_count++)
50   {
51      cout<<"\nEnter element number "<<index_count<<":";
52      cin>>num[index_count];
53   }
54
55
56
57
58
```

```
// processing the array: adding 2 to each element
   for(index_count=0;index_count<LIM;index_count++)
   {
      num[index_count]=num[index_count]+2.0;
   }
   cout<<"\nTwo (2.0) shall be added to each location now.....";
   cout<<"\nPress any key to continue";
   getch();
   // loop for printing output
   for(index_count=0;index_count<LIM;index_count++)
   {
      cout<<"\nElement no."<<index_count<<" = "<<num[index_count];
   }

   getch();
   return 0;
}
```

## 5) Computing average of 10 numbers with arrays

```
#include<iostream>
#include<conio2.h>
using namespace std;

#define LIM 10

int main(void)
{
   // variable definition
   float num[LIM];      //10 element float array, numbered 0 to 9
   float sum=0.0,average;//
   int index_count;     //an integer variable to access different array locations

   // taking array input through for() loop: index_count runs from 0 to 9

```

```
16 for(index_count=0;index_count<LIM;index_count++)
17    {
18       cout<<"\nEnter element number "<<index_count<<":";
19       cin>>num[index_count];
20    }
21
22    cout<<"\nAverage is being calculated now.....";
23    cout<<"\nPress any key to continue";
24    getch();
25    // processing the array: calculating running sum
26    for(index_count=0;index_count<LIM;index_count++)
27    {
28       sum=sum+num[index_count];
29    }
30    average=sum/LIM;
31    // printing output
32    cout<<"\nAverage of given 10 nos. ="<<average;
33    getch();
34    return 0;
35 }
```

## 6) Using for loop to display array contents – a better method

```
1  #include<iostream>
2  #include<conio2.h>
3  using namespace std;
4
5  #define LIM 10
6
7  int main(void)
8  {
9     // variable definition
10    float num[LIM]; //10 element float array, numbered 0 to 9
11    int index_count;//an integer variable to access different array locations
12
13    // taking array input through for() loop: index_count runs from 0 to 9
14
```

```
15  for(index_count=0;index_count<LIM;index_count++)
16    {
17       cout<<"\nEnter element number "<<index_count<<":";
18       cin>>num[index_count];
19    }
20    cout<<endl<<endl;
21    cout<<"press any key to continue.....";
22    getch();
23    // printing array: method 1
24    for(index_count=0;index_count<LIM;index_count++)
25    {
26       cout<<index_count<<":"<<num[index_count]<<endl;
27    }
28    cout<<endl<<endl;
29    // printing array: method 2
30    for(index_count=0;index_count<LIM;index_count++)
31    {
32       cout<<index_count<<":"<<num[index_count]<<"  ";
33       if((index_count+1)%5==0)
34       {
35          cout<<endl;
36       }
37    }
38    return 0;}
```

## Practical programming with arrays- use of for() and while()

- **Arrays can be manipulated with for() loops. This is ideal in the case of arrays that need to be processed completely – all elements, first to last.**
- **But when we need only a part of array which has some unused locations, we need while() loops.**
- **In case of using while() loops for controlling array operations where all array locations are not utilized there are two (02) special considerations.**

## 7) Array with while() loop

```
1  #include<iostream>
2  #include<conio2.h>
3  using namespace std;
4
5  int main(void)
6  {
7     const int MAX=100;
8     // Array/variable definition
9     char name[MAX]; // char array of 100 elements
10    int index;      // integer variable to address array locations
11    int maxindex;   // integer variable to store number of elements utilised
12
13    cout<<"\nEnter your name (press ESC to stop)\n";
14
15    // Taking input in array locations: index goes from 0 to unknown value
16    index=0;           // setting index to 0 before loop
17    name[index]=getche();// Initialising first element from user
18                    // before running for the first time
19
20    while( name[index]!=27 )    // loop ends when ESC is pressed
21    {
22       index++;
23       if(index==100)                //
24       {                        // Extremely Important 1:
25          cout<<"\nArray Overflow\n"; // Provide breaking mechanism
26          break;                   // in case of overflow
27       }                        //
28       name[index]=getche();
29    }
30
31    maxindex=index; //Extremely Important 2: saving the the last element
32                 //number given by user
33    // Simple processing of name[] array, converting small into capital case
34    // index goes upto maxindex less 1
35
36
```

```
for(index=0; index<maxindex; index++)
  {
    if((name[index]>=97)&&(name[index]<=122))
    {
      name[index]=name[index]-32;
    }
  }
  cout<<"\nPrinting the processed array.\n";
  // Printing array elements to see the change made
  for(index=0;index<maxindex; index++)
  {
    cout<<name[index]; // displaying each location
  }
  getch();
  return 0;
}
```

## 8) Average of array elements with while loop ( UIY)

```
// Proper method of processing arrays with while() loop
#include<iostream>
#include<conio2.h>
using namespace std;
#define LIM 100
// when we want to process arrays partially, while() loop
// provides an elegant structure but requires careful thinking.
int main(void)
{
  // variable definition
  float num[LIM];   // 100 element float array, numbered 0 to 99
  int index_count=0;// an integer variable to access different array locations
  int max_index;    // an integer variable to store the number of elements
                    // actually used
  char option;      // char variable to control the while loop
  float sum=0.0,average;    //running sum and average
 // asking user to start array input before starting while() loop
  cout<<"\nWould you like to enter array value (y or n):";
  option=getche();
```

```
/ while(option=='y') // condition as per given message
  {
    cout<<"\nEnter element number "<<index_count<<":";
    cin>>num[index_count];
    cout<<"\nWould you like to enter another value? (y or n):";
    option=getche();
    ++index_count;
    if(index_count>=1000)         // separate breaking condition
    {
      cout<<"\nArray Overflow.";
      break;                  // we can also write option='n';
    }
  }
  max_index=index_count;  // saving the the last element number given by user
  cout<<"\n\n";
  // calculating running sum
  for(index_count=0;index_count<max_index;index_count++)
  {
    sum=sum+num[index_count];
  }
  //calculating and printing average
  average=sum/index_count;
  cout<<"\n\nAverage of given "<<index_count<<" numbers is "<<average;
  getch();
  return 0;
}
// Note: 1) A while() loop is feasible when we have to process a subset of array
//         elements from the total array locations.
//       2) When taking array input with while() loop, two steps can be taken to
//         reduce further complexity:
//         a) Storing the total number of elements actually used
//           in a separate variable after the loop.
//         b) Writing a separate breaking condition inside the
//           loop for array overflow.
//       After taking these steps we can do further processing using for() loops.

```

## 9) Array exercise 1:reversing an array within

```
1  //Array exercise 1: Reversing an array within
2  //These exercises are provided for self practice
3  //so UIY (Understand It Yourself)
4  #include<iostream>
5  #include<conio2.h>
6  using namespace std;
7  #define MAX 100
8  int main (void)
9  {              // Variable declaration and definition
10                int num[MAX]={1,2,3,4,5,6,7,8,9,10};
11                int index, reverse_index, buffer, max_index=9;
12                // Displaying original array
13                cout<<"original array is:";
14                for(index=0;index<10;index++)
15                {
16                              cout<<num[index]<<"  ";
17                }
18                // Reversing loop
19                reverse_index=max_index;
20                for(index=0;index<=(max_index/2);index++)
21                {
22                              // Swapping the top and bottom elements
23                              buffer=num[index];
24                              num[index]=num[reverse_index];
25                              num[reverse_index]=buffer;
26                              --reverse_index;
27                }
28                // Displaying reversed array
29                cout<<"\nthe reversed array is:";
30                for(index=0;index<10;index++)
31                {
32                              cout<<num[index]<<"  ";
33                }
34                return 0;
35 }
36
```

# 10) Array exercise 2:Inserting an element at specified location

```
1   ///Array exercise 2: Inserting an element at specified location
2   //These exercises are provided for self practice
3   //so UIY (Understand It Yourself)
4   #include<iostream>
5   #include<conio2.h>
6   using namespace std;
7   #define MAX 100
8   int main(void)
9   {
10     // variable definition
11     float temp[MAX]={ 1.0, 2.3, 4.8, -5.3, 6.6, 9.8, 0.5, 2.3, 1.0, 12.7};
12     int index_count, insert_index, max_index=9;
13     float insert_element;
14     // Displaying the original array
15     for(index_count=0;index_count<=max_index;index_count++)
16     {
17        cout<<"\nElement "<<index_count<<" = "<<temp[index_count];
18     }
19     // Asking user to enter element value and index to be inserted
20     cout<<"\n\nEnter value to be inserted (float number):";
21     cin>>insert_element;
22     cout<<"\n\nEnter location where value is to be inserted:";
23     cin>>insert_index;
24
25     // Inserting
26     // Shifting all elements 1 location down to make space
27     // for new value (from insert_index to last)
28     for(index_count=max_index;index_count>=insert_index;index_count--)
29     {
30        temp[index_count+1]=temp[index_count];
31     }
32     temp[insert_index]=insert_element;// Now put the new value at specified index
33     max_index++;                  // Increasing max_index as array size increased
34
35     // Displaying the final, processed array
36     for(index_count=0;index_count<=max_index;index_count++)
37     {
38        cout<<"\nElement "<<index_count<<" = "<<temp[index_count];
39     }
40     getch();
41     return 0;
   }
```

## 11) Array exercise 3: Deleting an element from specified location

```
1  //These exercises are provided for self practice
2  //so UIY (Understand It Yourself)
3  #include<iostream>
4  #include<conio2.h>
5  using namespace std;
6  #define MAX 100
7  int main(void)
8  {
9     // variable definition
10    float temp[MAX]={ 1.0, 2.3, 4.8, -5.3, 6.6, 9.8, 0.5, 2.3, 1.0, 12.7};
11    int index_count, delete_index, max_index=9;
12    // Displaying the original array
13    for(index_count=0;index_count<=max_index;index_count++)
14    {
15       cout<<"\nElement "<<index_count<<" = "<<temp[index_count];
16    }
17    // Asking user to enter element index to be deleted
18    cout<<"\n\nEnter location to be deleted:";
19    cin>>delete_index;
20
21    // Deleting
22    // Shifting all elements 1 location up (from delete_index to last)
23    for(index_count=delete_index;index_count<max_index;index_count++)
24    {
25       temp[index_count]=temp[index_count+1];
26    }
27    max_index--; // reducing total tally count as one element is deleted
28
29    // Displaying the final, processed array
30    for(index_count=0;index_count<=max_index;index_count++)
31    {
32       cout<<"\nElement "<<index_count<<" = "<<temp[index_count];
33    }
34    getch();
35    return 0;}
36
```

## 12) Array exercise 4: Shifting an array downward by given amount

```
1  #include<iostream>
2  #include<conio2.h>
3  using namespace std;
4  #define MAX 100
5  int main (void)
6  {
7                  int num[MAX]={1,2,3,4,5,6,7,8,9,0};
8                  int index, downshift, max_index=9 ;
9                  cout<<"original array is:";
10                 for(index=0;index<=max_index;index++)
11                 {
12                                 cout<<endl<<num[index];
13                 }
14                 // shifting inputs
15                 cout<<"\nEnter the amount of down shift:";
16                 cin>>downshift;
17   //**********Actual Shifting Takes Place Here***********
18                 for(index=max_index;index>=downshift;index--)
19                 {
20                                 num[index]=num[index-downshift];
21                 }
22                 // zero padding
23                 for(index=0;index<downshift;index++)
24                 {
25                                 num[index]=0;
26                 }
27   // ********Shifting ends here**********************
28                 // display the final output
29                 cout<<"\n the shifted array is:";
30                 for(index=0;index<10;index++)
31                 {
32                                 cout<<endl<<num[index];
33                 }
34                 return 0;
35 }
36
```

## 12) Array exercise 4: Shifting an array downward by given amount

```
1  #include<iostream>
2  #include<conio2.h>
3  using namespace std;
4  #define MAX 100
5  int main (void)
6  {
7                  int num[MAX]={1,2,3,4,5,6,7,8,9,0};
8                  int index, downshift, max_index=9 ;
9                  cout<<"original array is:";
10                 for(index=0;index<=max_index;index++)
11                 {
12                                 cout<<endl<<num[index];
13                 }
14                 // shifting inputs
15                 cout<<"\nEnter the amount of down shift:";
16                 cin>>downshift;
17   //**********Actual Shifting Takes Place Here***********
18                 for(index=max_index;index>=downshift;index--)
19                 {
20                                 num[index]=num[index-downshift];
21                 }
22                 // zero padding
23                 for(index=0;index<downshift;index++)
24                 {
25                                 num[index]=0;
26                 }
27   // ********Shifting ends here**********************
28                 // display the final output
29                 cout<<"\n the shifted array is:";
30                 for(index=0;index<10;index++)
31                 {
32                                 cout<<endl<<num[index];
33                 }
34                 return 0;
35 }
36
```

## 13) Sequential Search: Searching an element in array

```
1  #include<iostream>
2  #include<conio2.h>
3  #include<cmath>
4  using namespace  std;
5  #define MAX 100
6  int main(void)
7  {   //Variable Definition and Declaration
8     float num[MAX]={12.0,5.5,7.3,6.0,3.0,0.5,9.8,7.5,2.9,66.0};
9     int index,check_flag=0, max_index=9, search_index;
10    float search_value;
11    // Entering value to be searched
12    cout<<"Enter the number to search in the array:";
13    cin>>search_value;
14
15    // Searching location No. of the search_value
16    for(index=0;index<=max_index;index++)
17    {
18       if(fabs(num[index]-search_value)<0.0001) // num[index]==search_value
19       {
20          check_flag=1;
21          search_index=index;
22          break;
23       }
24    }
25
26    if(check_flag==1)
27    {
28       cout<<"\nsearch value "<<search_value<<" found at location:"<<search_index;
29    }
30    else
31    {
32       cout<<"\nNo search found ";
33    }
34    return 0;
35 }
```

## 14) Bubble sort (Exchange sort): Sorting an array in order

```
1  #include<iostream>
2  #include<conio2.h>
3  using namespace std;
4  #define MAX 100
5
6  int main (void)
7  {              // Variable definition and declaration
8                 int num[MAX]={10,2,34,4,-5,6,77,18,9,0};
9                 int pivot, moving_index, buffer, max_index=9;
10               //Displaying the original array
11                cout<<"original array is:";
12                for(pivot=0;pivot<=max_index;pivot++)
13                {
14                                cout<<num[pivot]<<" ";
15                }
16                // sorting with Bubble Sort Algorithm
17                //Outer Loop: controls pivot 0 to (max_index-1)
18                for(pivot=0;pivot<max_index;pivot++)
19                {
20                  //Inner Loop controls moving_index=pivot+1 to max_index
21
22                for(moving_index=pivot+1;moving_index<=max_index;moving_index++)
23                              {
24                                 //if the following element is less than preceeding element
25                                 // then swap: the '<' can be replaced with '>' to sort in
26                                 // descending order
27                                              if(num[moving_index]<num[pivot])
28                                              {
29                                                              buffer=num[moving_index];
30                                                              num[moving_index]=num[pivot];
31                                                              num[pivot]=buffer;
32                                              }
33                              }
34                }
35
```

```
36              // display final array
37              cout<<"\nthe sorted array is:";
38              for(pivot=0;pivot<=max_index;pivot++)
39              {
40                              cout<<num[pivot]<<" ";
41              }
42              return 0;
43  }
```

## Calling functions to process arrays

Arrays can also be passed to functions for input, processing and output. One vital difference between passing variable and passing array to a function is that 'Calling' a function with array as argument is a 'Call by Reference' - When array is passed to function, it's address is passed to it. What is the consequence of this little difference?

## Method for writing function that passes arrays

- Write prototype with name of array to be used inside function with proper data-type.

- Write the function inside which the array is processed. Note that this processing shall modify the original array in main() function, thus no need to return the array.

## 15) Passing arrays to functions

```
1  //Passing arrays to functions
2  #include<iostream>
3  #include<conio2.h>
4  using namespace std;
5  #define MAX 100
6
7  //Prototype 1:function to display a double array
8  void displaydoublearray(const double arr[],int start,int stop, int SIZE);
9
10 //Prototype 2:function to find and return maximum value of double array
11 double maxofdoublearray(const double arr[],int start,int stop);
12
13 //Prototype 3:Input double array from user
14 void inputdoublearray(double arr[],int start,int stop, int SIZE);
15
16 int main (void)
17 {
18                 // Variable definition and declaration
19                 double num[MAX];
20                 double largest;
21                 int elements;
22                 cout<<"How many array elements do you need:";
23                 cin>>elements;
24                 //input array elements from user
25                 inputdoublearray(num,0,elements-1,MAX);
26    //display array
27    cout<<"\nThe array you entered is:\n";
28    displaydoublearray(num,0,elements-1,MAX);
29    //find largest element of the given array
30                 largest=maxofdoublearray(num,0,elements);
31                 cout<<"\nThe largest among the given numbers"
32                    <<"is "<<largest;
33                 return 0;
34 }
35
```

```
//****************** Definition of displaydoublearray()**********************
//Definition 1:
void displaydoublearray(const double arr[],int start,int stop,int SIZE)
{
   int index;
   //Displaying the array
   if((SIZE>stop)&&(start>=0))
   {
      for(index=start;index<=stop;index++)
      {
         cout<<arr[index]<<" ";
         if((index+1)%5==0)
         {
            cout<<endl;
         }
      }
   }
   else
   {
      cout<<"\n\nERROR:Array limits exceeded.";
   }
}
//******************Definition of maxofdoublearray()*********************
//Definition 2:function to find and return maximum value of double array
double maxofdoublearray(const double arr[],int start,int stop)
{
   int index;
   float maximum=arr[start];//assuming the first element to be maximum
   //Defensive condition: program runs only if this condition is met
   //Loop to go through all elements start_index---->stopindex
   for(index=start;index<=stop;index++)
   {
      if(arr[index]>maximum)//if current assumption is wrong
      {
         maximum=arr[index];// select a new max
      }
   }
```

```
return maximum;
}
//*****************Definition of inputdoublearray()**************************
//Definition 3:Input double array from user
void inputdoublearray(double arr[],int start,int stop,int SIZE)
{
   int index;
   if((SIZE>stop)&&(start>=0))
   {
      for(index=start;index<=stop;index++)
      {
         cout<<"Enter location "<<index<<":";
         cin>>arr[index];
      }
   }
   else
   {
      cout<<"\n\nERROR:Array limits exceeded.";
   }
}
```

## 16) Passing arrays to functions: Checking for Palindromes

```
#include<iostream>
#include<conio2.h>
using namespace std;
#define MAX 100

//Prototype 1:function to copy a char array into another
void copychararray(const char arr1[],char arr2[], int SIZE);

//Prototype 2:function to reverse a char array within itself
void reversechararray(char arr[],int start,int stop, int SIZE);

//Prototype 3:compare char arrays of same size, element by element
bool comparechararray(const char arr1[],const char arr2[],int SIZE);

```

```
int main (void)
{
                // Variable definition and declaration
   char word1[10]={'r','a','c','e','e','a','r'};
   char word2[10];
   cout<<"The first word is:";
   for(int index=0;index<7;index++)
   {
      cout<<word1[index];
   }
   cout<<endl<<endl;
   copychararray(word1,word2,7);
   cout<<"The 2nd word is:";
   for(int index=0;index<7;index++)
   {
      cout<<word2[index];
   }
   cout<<endl<<endl;
   reversechararray(word2,0,6,10);
   cout<<"The 2nd word, reversed is:";
   for(int index=0;index<7;index++)
   {
      cout<<word2[index];
   }
   cout<<endl<<endl;
   if(comparechararray(word1,word2,7)==1)
   {
      cout<<"word1 is a palindrome";
   }
   else
   {
      cout<<"word1 is not a palindrome";
   }
                return 0;
}
```

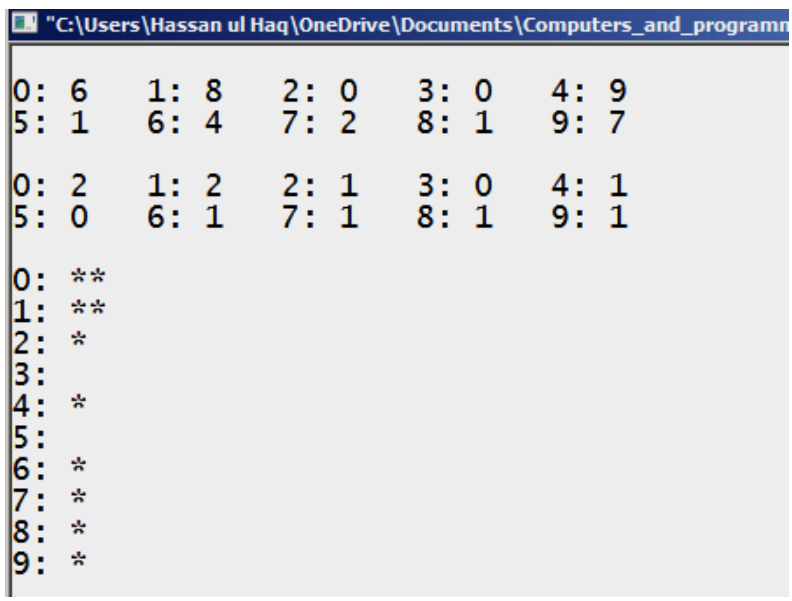
```
//Definition 1:
void copychararray(const char arr1[],char arr2[], int SIZE)
{
   int index;
   for(index=0;index<SIZE;index++)
   {
      arr2[index]=arr1[index];
   }
}
//Definition 2:
void reversechararray(char arr[],int start,int stop, int SIZE)
{
   int index, reverse_index=stop;
   char temp;
   if(start>=0 && start<stop && stop<SIZE )
   {
      for(index=0;index<(stop/2);index++)
      {
         // Swapping the top and bottom elements
         temp=arr[index];
         arr[index]=arr[reverse_index];
         arr[reverse_index]=temp;
         --reverse_index;
      }
   }
   else
   {
      cout<<"\n\ncopy operation failed";
   }
}
//Definition 3:
bool comparechararray(const char arr1[],const char arr2[], int SIZE)
{
   for(int index=0;index<SIZE;index++)
   {
      if(arr1[index]!=arr2[index])
      {
         return 0;
      }
   }
   return 1;
}

```

# 17) Application Program: Making Histogram for frequency distribution

```
1  #include <iostream>
2  #include <cstdlib>
3  #include <ctime>
4
5  using namespace std;
6
7  void buildrandomarray(int arr[], int size);
8
9  void clearintarray(int arr[], int size);
10
11 void displayintarray(int arr[], int size);
12
13 void histogram(int arr[], int size);
14
15
16 int main(void)
17 {
18    const int size = 10;
19    int numbers[size];
20    buildrandomarray(numbers, size);
21    cout << endl;
22    displayintarray(numbers, size);
23    cout << endl;
24    int distrib[size];
25    clearintarray(distrib, size);
26    for(int i = 0; i < size; ++i)
27       distrib[numbers[i]] += 1;
28    displayintarray(distrib,size);
29    cout << endl;
30    histogram(distrib, size);
31    cout << endl;
32    return 0;
33 }
34
35
```

```
36 void buildrandomarray(int arr[], int size)
37 {
38    srand(time(NULL));
39    for(int i = 0; i < size; ++i)
40       arr[i] = rand() % 10 + 0;
41 }
42
43 void clearintarray(int arr[], int size)
44 {
45    for(int i = 0; i < size; ++i)
46       arr[i] = 0;
47 }
48
49 void displayintarray(int arr[], int size)
50 {
51    for(int i = 0; i < size; ++i)
52    {
53       cout << i << ": " << arr[i]<<"   ";
54       if((i+1)%5==0)
55       {
56          cout<<endl;
57       }
58    }
59
60 }
61
62 void histogram(int arr[], int size)
63 {
64    for(int i = 0; i < size; ++i) {
65       cout << i << ": ";
66       for(int j = 1; j <= arr[i]; ++j)
67          cout << "*";
68       cout << endl;
69    }
70 }
71
72
```

Output:



# Two Dimensional Arrays

C language allows the programmer to define and work with multi-dimensional arrays. Multidimensional arrays find extensive usage in programming specially when handling large data-set. In this second part of the lab we will see how to use 2-dimensional arrays.

## Representation of a 2D array

A 2D array, also called an array of arrays, is practically nothing but an a two dimensional grid of numbers or characters. Just like a 1D array which has a max length, a 2D array has both max length and max width.

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 12.9 | -78.0 | 90.8 | 0.0 |
| 1 | 0.0 | 0.0 | 569.7 | 100.0 |
| 2 | 55.0 | -10.0 | 988.1 | 12.0 |

**A 3x2 (2D) float array**

## Defining and Declaring a 2D Array

A 2D array requires both ROWS and COLUMNS to be defined in separate square

brackets.

Examples: int num[5][10] ; // a 5 row and 10 column integer array.

char names[5][20]; // a character array in which 5 names can be stored each with 20 characters length.

Declaration is also similar for a 2D array that is by using curly brackets {}.

Examples: float price[2][3] = { {12.0 ,3.4 ,56.8} , {0.0 ,23.8 ,65.8} } ;

char cars[3][7]= { {‘t’,’o’,’y’,’o’,’t’,’a’} , {‘k’,’i’,’a’} , {‘h’,’o’,’n’,’d’,’a’} };

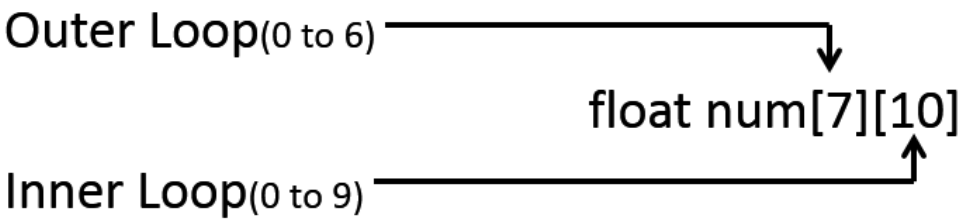
## 18) 2D Array Introduction: Using nested loop to manipulate 2D array

```
1  //2D Arrays Intro
2  #include<iostream>
3  #include<conio2.h>
4  using namespace std;
5
6  #define row 5
7  #define col 3
8
9  int main(void)
10 {
11                 int i,j;
12    float num[row][col];
13    cout<<"Enter elements of array:";
14   //***************nested loop for scanning**************
15    for(i=0;i<row;i++) // i controls rows
16    {
17       for(j=0;j<col;j++)  // j controls columns
18       {
19          cout<<"\nEnter location "<<i<<","<<j<<":";
20                                          cin>>num[i][j];
21       }
22    }
23    cout<<"\n \n";
24    //***************nested loop for printing**************
25                 for(i=0;i<row;i++)  // i controls rows
26                 {
27                              for(j=0;j<col;j++)  // j controls columns
28                              {
29                                          cout<<i<<","<<j<<"="<<num[i][j]<<"  ";
30                              }
31                              cout<<"\n";
32                 }
33                 getch();
34    return 0;
35 }
```

## Practical Programming with 2D Arrays

Considerations for practical programming with 2D arrays are same as those for 1D arrays. However there is one understandable difference ; to access all locations of a 2D array, you need a nested loop.

➔ A 2D array requires nested loop for accessing all of its locations.




## 19) 2D Array Introduction: Part 2

```
1  //2D Arrays Intro:Part 2
2  #include<iostream>
3  #include<conio2.h>
4  using namespace std;
5  int main(void)
6  {
7     const int row=5,col=5;
8                  int i,j;
9                  //2D float array initialization
10    float num[row][col]={{2,6,23},{43,6,18},{1,5,9}};
11    //2D char array initialization
12    char animals[row][col]={{'c','a','t'},{'b','a','t'},{'r','a','t'}};
13 //***************nested loop for printing**************
14                 for(i=0;i<row;i++)  // i controls rows
15                 {
16                               for(j=0;j<col;j++)  // j controls columns
17                               {
18                                            cout<<i<<","<<j<<"="<<num[i][j]<<"  ";
19                               }
20                               cout<<"\n";
21                 }
```

```
22 cout<<"\n\n";
23                 //***************nested loop for printing**************
24                 for(i=0;i<row;i++)  // i controls rows
25                 {
26                                 for(j=0;j<col;j++)  // j controls columns
27                                 {
28                                                 cout<<i<<","<<j<<"="<<animals[i][j]<<"  ";
29                                 }
30                                 cout<<"\n";
31                 }
32                 cout<<"\n\n";
33                 //another way to print 2D char array
34                 //*********this method prints one row at once**************
35                 for(i=0;i<row;i++)  // i controls rows
36                 {
37       cout<<animals[i]<<endl;
38                 }
39                 getch();
40    return 0;
41 }
```

## Practical programming with arrays- use of for() and while()

In the previous example, the array was manipulated with nested for() loops. This is ideal in the case of arrays that need to be processed completely – all elements, first to last. But when we need only a part of array which has some unused locations, we need nested while() loops.

## 20) 2D Array Application: Computing student average and score average

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7      const int rows = 5;
8      const int cols = 5;
9      int total = 0;
10     double average = 0.0;
11     //each row contains scores of 5 test of a student
12     //each column contains score in a single test taken
13     //by all 5 students
14     int grades[rows][cols] = {{75, 82, 84, 79, 91},
15                               {85, 81, 94, 96, 89},
16                               {92, 91, 94, 89, 90},
17                               {74, 72, 81, 78, 80},
18                               {84, 82, 82, 83, 81}};
19     //printing student average for all 5 tests
20     for(int r = 0; r < rows; ++r)
21        {
22           cout << "Student " << r+1 << ": ";
23           for(int c = 0; c < cols; ++c)
24           {
25              cout << grades[r][c] << " ";
26              total += grades[r][c];
27           }
28        average = total / cols;
29        cout << "Average: " << average << endl;
30        total = 0;
31        average = 0.0;
32        }
33        cout << endl;
34     //printing test average for all 5 stuents
35
```

```
36  for(int c = 0; c < cols; ++c)
37        {
38           cout << "Test " << c+1 << ": ";
39           for(int r = 0; r < rows; ++r)
40           {
41              cout << grades[r][c] << " ";
42              total += grades[r][c];
43           }
44        average = total / rows;
45        cout << "Average: " << average << endl;
46        total = 0;
47        average = 0.0;
48        }
49     return 0;
50  }
```

## 21) 2D array to functions

```
1   #include<iostream>
2   #include<conio2.h>
3   using namespace std;
4   #define ROW 3
5   #define COL 3
6   //Prototype for square-matrix addition function
7   void matadd(double a[][COL],double b[][COL],double c[][COL],int dimension);
8
9   //Prototype for square-matrix subtraction function
10  void matsub(double a[][COL],double b[][COL],double c[][COL],int dimension);
11
12  //Prototype for square-matrix multiplication function
13  void matmul(double a[][COL],double b[][COL],double c[][COL],int dimension);
14
15  int main(void)
16  {
17     // A[][] and B[][] arrays for input, C[][] for output. All are initially 0.
18                  double A[ROW][COL],B[ROW][COL],C[ROW][COL];
19                  int index_row,index_col,k;
20                  char option;
```

```
21  for(int c = 0; c < cols; ++c)
22      {
23         cout << "Test " << c+1 << ": ";
24         for(int r = 0; r < rows; ++r)
25         {
26            cout << grades[r][c] << " ";
27            total += grades[r][c];
28         }
29      average = total / rows;
30      cout << "Average: " << average << endl;
31      total = 0;
32      average = 0.0;
33      }
34    return 0;
35  }
36
37  //*****************Enter contents in A[][]************************
38  cout<<"Enter the contents of matrix A";
39  for(index_row=0;index_row<ROW;index_row++)
40  {
41                  for(index_col=0;index_col<COL;index_col++)
42                  {
43                              cout<<"\nEnter element "<<index_row<<","<<index_col<<":";
44                                          cin>>A[index_row][index_col];
45                  }
46  }
47
48  //***************Enter contents in B[][]************************
49  cout<<"\n\nEnter the contents of matrix B";
50  for(index_row=0;index_row<ROW;index_row++)
51  {
52                  for(index_col=0;index_col<COL;index_col++)
53                  {
54                              cout<<"\nEnter element "<<index_row<<","<<index_col<<":";
55                                          cin>>B[index_row][index_col];
56                  }
57  }
```

```
cout<<"\n\nWhat operation do you need to perform?";
cout<<"\nEnter +, - or *:";
option=getche();

switch(option)
{
   case '+':
      matadd(A,B,C,3);
      break;
   case '-':
      matsub(A,B,C,3);
      break;
   case '*':
      matmul(A,B,C,3);
      break;
   default:
      cout<<"\nInvalid Operator.";
      return 0;
}
cout<<"\n\n";
// ***************display result ************************
for(index_row=0;index_row<ROW;index_row++)
{
                for(index_col=0;index_col<COL;index_col++)
                {

                cout<<index_row<<","<<index_col<<":"<<C[index_row][index_col]<<"   ";
                }
                cout<<"\n";
}
return 0;
}

//definition for square-matrix addition function
void matadd(double a[][COL],double b[][COL],double c[][COL],int dimension)
{

```

```
for(int index_row=0;index_row<dimension;index_row++)
                {
                        for(int index_col=0;index_col<dimension;index_col++)
                        {

                    c[index_row][index_col]=a[index_row][index_col]
                                                +b[index_row][index_col];
                        }
                }
}

//Prototype for square-matrix subtraction function
void matsub(double a[][COL],double b[][COL],double c[][COL],int dimension)
{
   for(int index_row=0;index_row<dimension;index_row++)
                {
                        for(int index_col=0;index_col<dimension;index_col++)
                        {
                                        c[index_row][index_col]=a[index_row][index_col]
          -b[index_row][index_col];
                        }
                }
}

//Prototype for square-matrix multiplication function
void matmul(double a[][COL],double b[][COL],double c[][COL],int dimension)
{
   for(int index_row=0;index_row<dimension;index_row++)
   {
                for(int index_col=0;index_col<dimension;index_col++)
                {
                  c[index_row][index_col]=0.0;
                    for(int k=0;k<dimension;k++)   // loop for running sum
                    {
                         c[index_row][index_col]=c[index_row][index_col]
                                +(a[index_row][k]*b[k][index_col]);
                    }
                }
    }
}
```