Notes

16

ARRAY

In the previous lesson you have learnt about C++ functions. Now you will learn about arrays. Sometimes, it is required to store a large number of variables of the same type. Instead of declaring many variables of the same data type you can use array. Array is a collection of values of same type. The array can be one dimensional (one subscript), two dimensional (two subscripts) or multidimensional (n-subscripts). In this lesson, you will learn to declare an array, initialize array elements, operate on one dimensional array and use array of strings.

OBJECTIVES

After reading this lesson, you will be able to:

- define an array;
- declare and initialize array elements;
- operate on one dimensional array;
- use array of strings;
- declare multidimensional array.

16.1 ARRAY

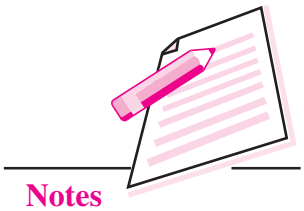
Array is a collection of values of same data type. For declaring an array you have to use following syntax:

```
datatype array_name[size];
```

For example

```
int A[5];
```

This statement declares an array A which allocates 5 continuous memory locations for storing integer data type. These locations are A[0], A[1], A[2],



A[3] and A[4]. The number within the square bracket is called index or subscript and in an array index starts with 0 (zero).

Array can be initialized by setting the elements while writing the program, those values will be hard coded values or can be initialized at run-time.

Syntax for initialization of array elements

type array_name[size]={ list of elements separated by comma };

The list of elements should be separated by commas.

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

Here A is an integer array of size 5. A[0] will have the value 1, A[1] will have the value 2, A[2] will have the value 3, A[3] will have the value 4, A[4] will have the value 5.

While initializing the array elements we can leave size as blank, it will automatically count the elements and will set the size.

For example, int A[ ] ={1,2,3,4,5};

In the above example we have not initialized the size. It will set the size of the array automatically by counting the number of elements. If the size of array is more than the initialized elements then for non-initialized array elements it will take garbage value.

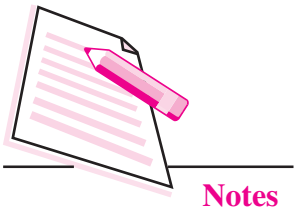
Initialization of array elements at run-time

```
int A[5];  // declaration of Array
cout<< "Enter  elements " ;
for(i=0;i<=4;i++)
{
cin>>A[i];      // elements will be entered one by one for i=0 to i=4
}
```

Here, A is an integer array of size 5. When you run this program, it will ask the value for A[0], A[1]…. A[4]. Those values will be stored in the array A. Similarly you can declare array of other data types. Now let us learn to access array elements.

Accessing Array Elements

You have to use array's index for accessing the array elements.



Notes

For example,

```
int i;
int A[2]={10,20,30};
for (i=0; i <= 2; i++)
{
        cout << A [i] ;
}
```

This program will display A[0], A[1] and A[2] values as 10, 20 and 30 respectively.

16.1.1 Processing an Array

The various operations that can be performed on arrays are

- Traversal
- Searching
- Sorting
- Insertion
- Deletion

Traversal

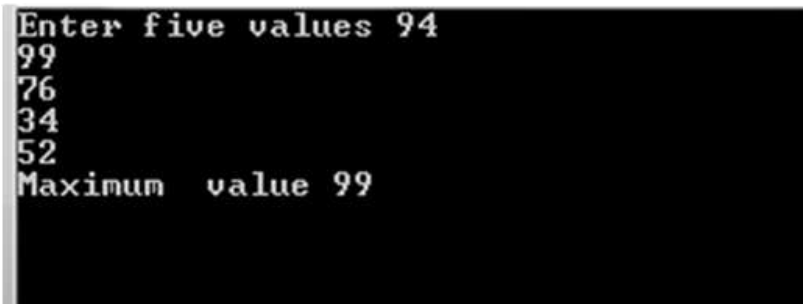
It means to access each location of an array may be for display purpose. Let us consider a program which will read five values from the user and finds out the maximum value.

Example 1

```
# include < iostream.h >
void main ( )
{
        int T, A[5], i;
        cout < < "Enter five values";
        for ( i = 0; i < 5; i++)
        cin > > A [ i ];
        T = A [ 0 ];
        for (i = 1; i < 5; i++)
```

```
        {
        if (T < A [i])
        T = A [ i ];
        }
    cout < < "Maximum value" < < T;
}
```

Output of the program



Searching

This method finds out whether the data entered by the user is present in an array or not. There are two types of searching method.

(i) Linear or Sequential search

(ii) Binary search

Linear or sequential search

This method is slower, inefficient and works on unsorted list. If the data we are searching is not present in the list, we come to know at the end of the list.

Example 2

```
/ / Linear search
# include < iostream.h >
void main ( )
{
        int A[5], i, data, flag = 0;
        cout < < "Enter five values";
        for (i = 0; i < 5; i++)
        cin > > A [i];
        cout << 'Enter data to be searched";
```

```
        cin >> data;
        for (i=0; i < 5; i ++)
                {
                if (A[i] = = data)
                flag = 1;
                }
        if (flag = = 1)
        cout < < "Data present";
        else cout << "Data not present";
}
```

Output of the program

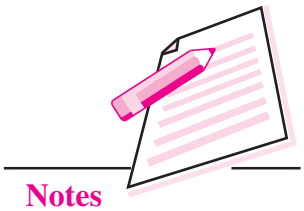
```
Enter five values
34
87
67
56
12
Enter data to be searched 67
Data present
```

Binary search

This method requires the array to be either in ascending or descending order. This method calculates the mid location from initial and final locations and compares the data with the value present in mid location. Consider the case where the list is in ascending order. If data is less than a [mid] then data is present in the upper half otherwise data is present in the lower half. The value of either final or initial will be changed. The mid value is calculated again and searching continues till the data is found or initial is greater than final. The values of initial, final and mid for searching a value of **35** in an ascending order sorted list containing 9 elements is shown below:

Let the array have value:

	10	15	18	20	27	30	35	40	45
location	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)

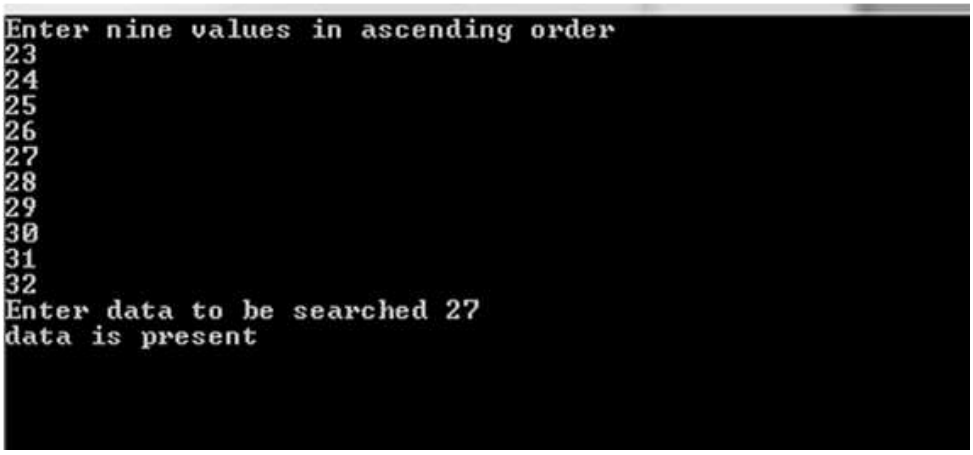


Initial	Final	Mid = $\frac{\text{Initial + Final}}{2}$	A[Mid]
0	8	4	27
5	8	6	35
			Search is successful

Example 3

```
/ / Binary search
# include < iostream.h >
const int N = 10;
void main ( )
{
int A[N], i, initial, final, mid, data;
cout < < "Enter nine values in ascending order";
for (i = 0; i < N; i++)
cin >> A [ i ];
cout << "Enter data to be searched";
cin >> data;
initial = 0;
final = N - 1;
mid = (initial + final) / 2;
While ((initial < = final) & & (A[mid]! = data))
{
if (A [mid] > data)
final = mid - 1;
else
initial = mid + 1;
}
if (A [mid] = = data)
cout << "data is present";
if (initial > final)
cout <<"data not present in the list";
}
```

Output of the program



The advantage of Binary search is that each search cuts the list in to half. A list of 10,000 names can be searched in just 12 searches.

Sorting

It is a method to arrange the list either in ascending or descending order.

Bubble sort

Consider an array of five locations to be sorted in ascending order:

	–1	9	5	0	2
Index	0	1	2	3	4

In this sorting method, A[0] is compared with A[1]. If A[0] is greater than A[1], the values are swapped. Then A[1] is compared with A[2], A[2] is compared with A[3], and A[3] is compared with A[4]. In all cases if the i^{th} location has value greater than $i + 1^{th}$ location, the values are swapped. The entire process is repeated N-1 times where N is the number of data in an array.

Example 4

```
# include < iostream.h >
const int N = 5;
void main ( )
{
        int A [N], i, j, T;
        cout << "Enter values";
        for (i = 0; i = N; i ++)
```

```
        cin >> A [ i ];
        / / sorting
        for ( i = 0; i < N - 1; i ++)
        for (j = 0; j < N - i; j ++)
        if (A[j] > A [j + 1])
                {
                T = A [ j ];
                A [ j ] = A [ j + 1 ];
                A [ j + 1 ] = T;
                }
        cout << "Sorted array is \n";
        for (i = 0; i < N; i ++)
        cout << A [ i ]<< "\n";
}
```

Output of the program

Selection sort

Consider an array having N elements to be sorted in ascending order. Initially, first element is compared with others so that it holds the smallest value. In the next pass, second element is compared with others so that it holds the second smallest value. This procedure is repeated for the entire array.

Example 5

```
# include < iostream.h >
const int N = 5;
void main ( )
{
int A [N], i, j, T;
cout << "Enter values";
for ( i = 0; i < N; i ++ )
cin >> A [ i ];
/ / sorting
for ( i = 0; i < N - 1; i ++ )
for (j = l; j < N; j ++ )
if (A [ i ] > A [j])
{
T = A [i];
A [i] = A [j];
A [j] = T;
}
// printing the sorted data
for ( i = 0; i < N; i ++ )
cout < < A [ i ];
}
```

Output of the program

```
Enter values
6
45
84
62
59
Sorted array is
6
45
59
62
84
```

Insertion

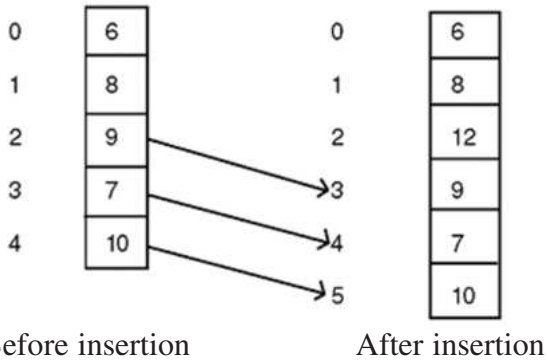
It means addition of a data item in the middle or at the end of the array. If data is to be added after a given data item then the location of the data item is first

determined by applying search procedure and then the insertion procedure is implemented.

Example 6

```
# include < iostream.h >
void main ( )
{
int x [20];
int i, loc, n, data;
cout << "Enter the no. of elements";
cin >> n;
for (i = 0; i < n; i ++ )
{
cout << "Enter the array element";
cin >> x [ i ];
}
cout << "Enter the location after which data is to be inserted";
cin >> loc;
for (i = n - 1; i > = loc; i - - )
x [ i + 1 ] = x [ i ];
cout << "enter the new data to be added";
cin >> x [loc];
n ++ ;
cout << "Array elements after insertion";
for (i = 0; i < n; i ++)
cout << x [ i ];
}
```

Let data 12 to be inserted at location 2



Before insertion After insertion

Notes

Deletion

It means removal of a data. First the location of the item to be deleted is determined by applying an appropriate search procedure and then the value present at particular location will be deleted.

Example 7

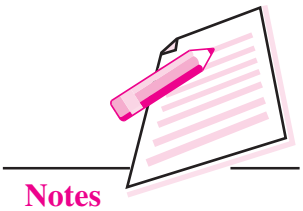
```
# include < iostream.h >
void main ( )
{
int x [20];
int i, j, n, loc, data;
cout << "Enter the no. of elements";
cin >> n;
for (i = 0; i < n; i ++)
{
cout << "Enter value";
cin >> x [ i ];
}
cout << "Enter the location to be deleted";
cin >> loc;
if (loc ! = 0)
{
data = x [loc ];
for {j = loc; j < n - 1; j ++ )
x [ j ] = x [ j + 1];
}
n = n - 1;
cout << "Elements after deletion";
for ( i = 0; i < n; i ++)
cout << x [ i ];
}
```

16.2 ARRAY OF STRINGS

Array of string is nothing but an array of characters.

Syntax for declaring an array of string is:

char array_name [size]={array of character in single quotes separated by commas}



For example, char name[5]={'G','A','U','R','A','V'};

String is a series of character stored in continuous memory location. The last character of each string is a null character '\0'. Actually, you do not place the null character at the end of a string constant. The C++ compiler automatically places the '\0' at the end of the string when it initializes the array.

For example, char name[6]={'G','A','U','R','A','V','\0'};

INTEXT QUESTIONS 16.1

1. Write a statement that defines one dimensional array called student of type int that holds 5 data elements.

2. In the following array declarations find the number of elements in each array and total no. of bytes required to store each array.

   (a) int A [20];

   (b) char z [18] ;

3. What will be the output of the following programs?

   (a)
```
# include < iostream.h >
void main ( )
 {
      int a [ 5 ], t ;
      for (i= 0; i < 5; i ++ )
      a [i] = 5 * i;
      for (i = 0; i < 5; i ++)
      cout << a [ i ];
 }
```

   (b)
```
# include < iostream.h >
void main ( )
{
      char title [ ] = "Computer";
      for ( int i = 0; title [ i ]; i ++)
      cout << "\n" << (title + i);
}
```

16.3 TWO-DIMENSIONAL ARRAY

It has two subscript or index, first for row and second for column. For example:

```
int A [ 5 ] [ 4 ];
```

The above has five rows and four columns. The total number of elements are 20. The subscripts of 20 elements are:

A[0][0]	A[0][1]	A[0] [2]	A[0][3]
A[1][0]	A[1][1]	A[1] [2]	A[1][3]
A[2][0]	A[2][1]	A[2] [2]	A[2][3]
A[3][0]	A[3][1]	A[3] [2]	A[3][3]
A[4][0]	A[4][1]	A[4] [2]	A[4][3]

Initialization of Two Dimensional Array

The initialization is done at the time of declaration of an array.

You can create multi-dimentional arrays like int A[ ] [ ] [ ], float B [ ] [ ] [ ] [ ] etc.

For example:

```
int A [2] [4] = {1, 2, 3, 4, 5, 6, 7, 8};
```

For more clarity

```
int A [2] [4] = { {1, 2, 3, 4 }, {5, 6, 7, 8} };
```

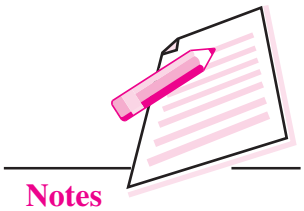
The above data can be grouped. The inner braces are ignored by the compiler.

Initialization of array elements at run-time

You can initialize array elements at run time.

Let int A[3][2]; // declaration of Array

```
cout<<"Enter  elements "
for(i=0;i<3;i++)              //1st for loop for row
{
        for(j=0;j<2;j++)      // 2nd for loop for column
        cin>>A[i][j];
}
```



Accessing Array Elements

Accessing array elements is same as reading elements.

Let us take an array int A[3][2]= {{1,2},{3,4},{5,6}};

For accessing/reading elements we will use following code:-

```
int A[3][2];                    // of Array
cout<<"Enter elements"
for(i=0;i<3;i++)                //1st for loop for row
{
        for(j=0;j<2;j++)
        cout<<A[i][j];
}
```

The following program finds out the maximum value stored in two dimensional array.

Example 8

```
# include , <iostream.h >
const int M = 5;
void main ( )
{ int A [M] [M], i, j, T;
cout >> "\n Enter Array Elements";
for (i = 0; i < = M - 1; i ++ )
for (j = 0; j < = M - 1; j ++ )
cin >> A [i] [j];
T = A [0] [0];
for ( i = 0; i < = M – 1; i ++)
for (j = 0; j < = M – 1; j ++)
{ (if (T < A [ i ] [ j ])
T = A [ i ] [ j ];
}
cout << "Largest value" << T << "\n";
}
```

INTEXT QUESTIONS 16.2

1. Define a two dimensional array A having 2 rows, 3 columns and stores int data type.

2. Consider the following array declarations. Find the number of elements in each array and total number. of bytes required by each array.

   (a) int x [5] [10];

   (b) long y [5] [10];

WHAT YOU HAVE LEARNT

- Array is a collection of values of same type.
- On one dimensional array you can perform the operations like traversing, searching, sorting.
- Traversing of an array means accessing all the elements of the array one by one.
- Searching is the process of finding an element within the array list.
- Sorting is a method to arrange the list either in ascending or descending order.
- Two dimensional array has two subscripts or index.
- You can initialize array elements at run time.

TERMINAL EXERCISE

1. What is the need of an array?
2. What do you mean by searching?
3. What do you mean by sorting?
4. Write a program that will read 20 float values in a one dimensional array and find out the following:

   (i) Number of values greater than zero.

   (ii) Number of values equal to zero.

   (iii) Number of values less than zero.

5. Write a program that will find out the sum of two diagonals of two dimensional array of A [N] [N].

6. Write a program to search whether the element entered by the user is present in a one dimensional array of 10 elements or not.

ANSWERS TO INTEXT QUESTIONS

16.1

1. int student [5];

2. (a) Number of elements - 20 Total number of bytes - 40

   (b) Number of elements - 18 Total number of bytes - 18

3. (a) 0 5 1015 20

   (b) COMPUTER

   OMPUTER

   MPUTER

   PUTER

   UTER

   TER

   ER

   R

16.2

1. int A [2] [3];

2. (a) Number of elements - 50

   Total number of bytes - 100

   (b) Number of elements – 50

   Total number of bytes - 200