11
Array

11.1 Introduction

The ordered collection of identical elements is called an array. Sometimes, it is required to store a large number of variables of the same type, one can use an array. For example:

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

The above statement declares an array A which reserves 5 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. The array can be one dimensional (one subscript) or two dimensional (two subscripts).

11.2 Objectives

After going through this lesson, you would be able to:

- explain the concept of an array
- define, access and initialize array elements
- perform common operations on arrays
- explain two dimensional array
- define, access and initialize two dimensional array elements

11.3 Initialization of One Dimensional Array

An array can be unitialized along with declaration. For array initialization it is required to place the elements separated by commas enclosed within braces.

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

It is possible to leave the array size open. The compiler will count the array size.

```
int B [ ] = {6, 7, 8, 11, 10};
```

11.4 Initialization of String

An array of characters known as character string may be initialized by placing the string in double quotes.

```
Char A [20] = "Computer Science";
Char B [  ] = "Calcutta";
```

A string is a series of characters stored in consecutive bytes of memory. This implies that one can store a string in an array of characters, with each character kept in its own array element. The last character of every C++ string is Null character. Therefore

```
char city [ ] = {'c, 'a', 'l', 'c', 'u', 't', 't', 'a'};
```

is an array of characters while

```
char city [ ] = {'c, 'a', 'l', 'c', 'u', 't', 't', 'a', '\0'};
```

is a string because it is terminated by a null character.

11.5 Processing an Array

The various operations possible on arrays are :

- Traversal
- Searching
- Sorting
- Insertion
- Deletion

Traversal

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

```
# 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;
}
```

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.

```
/ / 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";
}
```

Binary search

This method requires the array to be either in ascending or descending order. This method calculates the mid location form 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:

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

Let the array have value:

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

```
/ / Binary search
# include < iostream.h >
const int N = 9;
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";
}
```

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, the values are swapped. The entire process is repeated N-1 times where N is the number of data in an array.

```
# 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 - 1; j ++)
If (A[j] > A [j + 1])
{
    T = A [ j ];
    A [ j ] = A [ j + 1 ];
    A [ j + 1 ] = T;
}
    cout << "Sorted array is";
    for (I = 0; I < N; I ++)
    cout << A [ I ];
}
```

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 smallest value. This procedure is repeated for the entire array.

```
# 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 = I; 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 ];
}
```

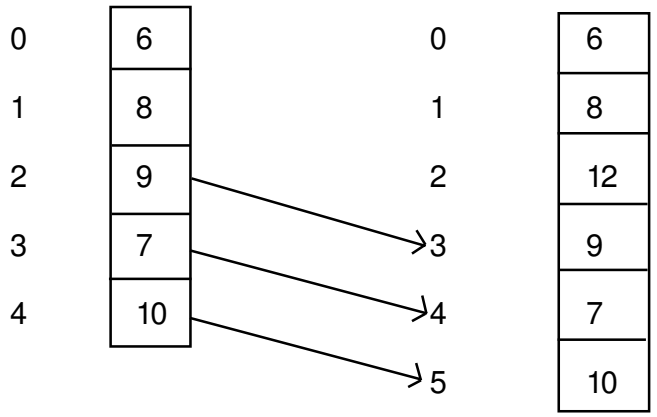
Insertion

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.

```
# include < iostream.h >
void main ( )
{
    int x [20] ;
```

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

Let data 12 to be inserted at location 2



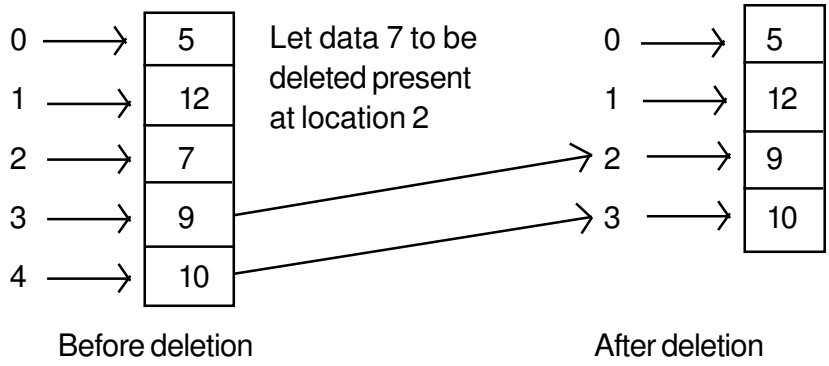
Before insertion After insertion

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 is deleted.

```
# 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 ];
}
```



In-Text Questions 11.1

1. What is an array ?

2. What do you mean by index of an array?

3. The elements of a five element array are numbered from ..... to ..... .

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

5. 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] ;

6. 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 = o; 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);
}
```

11.6 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.

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.

For two dimensional array, two loops are required. The following program finds out the maximum value stored in two dimensional array.

```
# 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”;
}
```

In-Text Questions 11.2

1. Define a two dimensional array A having 2 rows, 3 columns and stores int data type.
2. How many elements are there in the array defined in question 1 above ?
3. How many bytes the above declaration (see question 1) will take in the memory?
4. Consider the following array declarations. Find the number of elements in each array and total no. of bytes required to each array.

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

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

11.7 What you have learnt

In this lesson we learnt how to define, access and assign values to one dimensional and two dimensional array. We have discussed the operations one can perform with one dimensional array.

11.8 Terminal Questions

1. What is the need of an array?
2. What do you mean by searching?
3. Give one advantage and one disadvantage of Linear and Binary search?
4. What do you mean by sorting?

5. Which method is better: Bubble or Selection sorting? Why ?
6. 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.
7. Write a program that will find out the sum of two diagonals of two dimensional array of A [N] [N].
8. Write a program that will find out whether the data entered by the user present in a one dimensional array of 10 elements using:
   (i) Linear search
   (ii) Binary search
9. Write a program that will sort the given ten numbers in descending order using:
   (i) Bubble sort
   (ii) Selection sort

11.9 Feedback to In-Text Questions

In-text Questions 11.1

1. Array is a collection of the same data type.
2. The index designates the position in array ordering.
3. 0, 4
4. int student [5];
5. (a) Number of elements - 20
   Total number of bytes - 40
   (b) Number of elements - 18
   Total number of bytes - 18

6. (a) 0 5 10 15 20

   (b) COMPUTER
   OMPUTER
   MPUTER
   PUTER
   UTER
   TER
   ER
   R

In-text Questions 11.2

1. int A [2] [3]; 2. 6 3. 12

4. (a) Number of elements - 50
   Total number of bytes - 100

   (b) Number of elements - 50
   Total number of bytes - 200