OBJECT: To Understand Array Fundamentals, Defining Arrays, Array Elements, Accessing Array Elements, Initializing Arrays, Operations on Arrays Multidimensional Arrays, Two Dimensional Array

ARRAY INTRODUCTION

An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type. Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index. All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

DECLARING ARRAYS

To declare an array in C++, the programmer specifies the type of the elements and the number of elements required by an array as follows:

```
type arrayName [ arraySize ];
```

This is called a single-dimension array. The arraySize must be an integer constant greater than zero and type can be any valid C++ data type. For example, to declare a 10-element array called balance of type double, use this statement

```
double balance[10];
```

INITIALIZING ARRAYS

You can initialize C++ array elements either one by one or using a single statement as follows −

```
double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

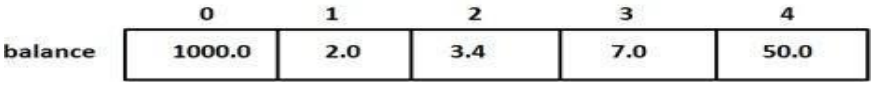
The number of values between braces { } can not be larger than the number of elements that we declare for the array between square brackets [ ]. Following is an example to assign a single element of the array −
If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write

```
double balance[] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

You will create exactly the same array as you did in the previous example.

```
balance[4] = 50.0;
```

The above statement assigns element number 5th in the array a value of 50.0. Array with 4th index will be 5th, i.e., last element because all arrays have 0 as the index of their first element which is also called base index. Following is the pictorial representation of the same array we discussed above



ACCESSING ARRAY ELEMENTS

An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array. For example:

```
double salary = balance[9];
```

The above statement will take 10th element from the array and assign the value to salary variable. Following is an example, which will use all the above-mentioned three concepts viz. declaration, assignment and accessing arrays

SOME IMPORTANT POINTS ABOUT ARRAYS

- We can access array randomly by using any index
- We can use a Loop to access all elements of an array in one by one
- Reduces code as we create many variables in one statement and use loop to access them
- Sorting and searching becomes easy as it can be accomplished by writing less line of code
- Allows a fixed number of elements to be entered which is decided at the time of declaration

PROGRAM 1: Demonstrating Simple Array.

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

int main() {
int arr[5];
int k;

cout<<"An Integer Array..."<<endl;
cout<<"Enter 10 elements in array"<<endl;

for (k=0 ; k<5 ; k++) {
cout<<"Enter arr[" << k << "]:";
cin>>arr[k];
}

cout<<"The elements in array are:"<<endl;

for (k=0 ; k<5 ; k++)
cout<<"arr[" << k << "]: " << arr[k] << endl;

return 0;
}
```

Exercise 1: Write a program that takes input of 5 elements for array int a[5] and the calculate the sum and average of the elements of array.

PROGRAM 2: Finding largest elements in an array.

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

int main() {
int arr[10];
int loc=0, large, k;
for (k=0 ; k<10 ; k++){
cout<<"Enter element No." << k+1 << ":   ";
cin>>arr[k];
}
large=arr[0];
for (k=0 ; k<10 ; k++)
if (large < arr[k]){
loc = k;
large=arr[k];
}
cout<<"The largest element is:"    << arr[loc];

return 0;
}
```

Exercise 2: Write a program which finds the smallest element in an array

PROGRAM 3: Demonstrates array initialization and shows days from start of year to date specified

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

int main() {
int month, day, total_days;
int days_per_month[12] = { 31, 28, 31, 30, 31, 30,
                           31, 31, 30, 31, 30, 31 };
   cout << "\nEnter month (1 to 12): ";  //get date
   cin >> month;
   cout << "Enter day (1 to 31): ";
   cin >> day;
   total_days = day;                     //separate days
   for(int j=0; j<month-1; j++)          //add days each month
      total_days += days_per_month[j];
   cout << "Total days from start of year is: " << total_days
     << endl;
   return 0;
   }
```

PROGRAM 4: Demonstrates different array types and their use.

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

int main() {
const int size=5;

int roll_no[size];
int marks[size];
float per[size];
char grade[size];

cout<<"\nPlease enter the data for 10 students..."<<endl;
for(int i=0 ; i<size ; i++)
{
cout<<"\nPlease Enter Student No."<<i+1<<" data..."<<endl;
cout<<"Roll No.:";
cin>>roll_no[i];
cout<<"Marks out of (500):";
cin>>marks[i];
per[i] = (float)marks[i] / 500 * 100;

if(per[i]>=80) grade[i]='A';
else if(per[i]>=70 && per[i]<80) grade[i] = 'B';
else if(per[i]>=60 && per[i]<70) grade[i] = 'C';
else if(per[i]>=50 && per[i]<60) grade[i] = 'D';
else if(per[i]>=40 && per[i]<50) grade[i] = 'P';
else grade[i] = 'F';
}

cout<<"\n\nPress any key to continue....";
getch();

cout<<"\n \n \n \t \t RESULT LIST \n \n";
cout<<setw(9)<<"ROLL NO."<<setw(7)<<"MARKS"
<<setw(12)<<"PERCENTAGE"<<"\tGRADE"<<endl;

for(int i=0 ; i<size ; i++)
cout<<setw(9)<<roll_no[i]<<setw(7)<<marks[i]
<<setw(12)
<<setprecision(2)  //Digit after decimal point
<<setiosflags(ios::fixed) //Do not show number in exponential form
<<setiosflags(ios::showpoint) //Alway show decimal point
<<per[i]<<"%\t"<<grade[i]<<endl;

return 0;
}
```

MULTI DIMENSIONAL ARRAY

C++ allows multidimensional arrays. Here is the general form of a multidimensional array declaration

```
type name[size1][size2]...[sizeN];
```

For example, the following declaration creates a three dimensional 5 . 10 . 4 integer array:

```
int threedim[5][10][4];
```

TWO-DIMENSIONAL ARRAYS

The simplest form of the multidimensional array is the two-dimensional array. A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional array of size x,y, you would write something as follows:

```
type arrayName [ x ][ y ];
```

Where type can be any valid C++ data type and arrayName will be a valid C++ identifier.
A two-dimensional array can be think as a table, which will have x number of rows and y number of columns. A 2-dimensional array a, which contains three rows and four columns can be shown as below:

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

Thus, every element in array a is identified by an element name of the form a[i][j], where a is the name of the array, and i and j are the subscripts that uniquely identify each element in a.

INITIALIZING TWO-DIMENSIONAL ARRAYS

Multidimensional arrays may be initialized by specifying bracketed values for each row. Following is an array with 3 rows and each row have 4 columns:

```
int a[3][4] = {
   {0, 1, 2, 3} ,   /*  initializers for row indexed by 0 */
   {4, 5, 6, 7} ,   /*  initializers for row indexed by 1 */
   {8, 9, 10, 11}   /*  initializers for row indexed by 2 */
};
```

ACCESSING TWO-DIMENSIONAL ARRAY ELEMENTS

An element in 2-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array. For example:

```
int val = a[2][3];
```

The above statement will take 4th element from the 3rd row of the array. You can verify it in the above diagram.

PROGRAM 5: Demonstrates a double dimensional array.

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

int main() {
int arr[5][3];
cout<<"An Integer Double Dimensional Array..."<<endl;
cout<<"Enter elements"<<endl;
for (int r=0 ; r<5 ; r++)
    for (int c=0; c<3; c++) {
cout<<"Enter arr[" << r << "]["<<c<<"]:";
cin>>arr[r][c];
}
cout<<"The elements in array are:"<<endl;
for (int r=0 ; r<5 ; r++){
    for (int c=0; c<3; c++)
cout<<arr[r][c]<<"  ";
cout<<endl;
}
return 0;

}
```

Exercise 3: Write a program in C++ for addition of two Matrices of same size (3 x 3).

```
Input the element of the square matrix (3 x 3 )
Input elements in the first matrix:
[0][0] : 9
[0][1] : 8
[0][2] : 7
[1][0] : 6
[1][1] : 5
[1][2] : 4
[2][0] : 3
[2][1] : 2
[2][2] : 1

Input elements in the second matrix:
[0][0] : 1
[0][1] : 2
[0][2] : 3
[1][0] : 4
[1][1] : 5
[1][2] : 6
[2][0] : 7
[2][1] : 8
[2][2] : 9

The First matrix is :
9  8  7
6  5  4
3  2  1
The Second matrix is :
1  2  3
4  5  6
7  8  9
The Addition of two matrix is :
10  10  10
10  10  10
10  10  10
```