UNIT -III | Introduction of C++ CHAPTER 12

Arrays and Structures

1.What is an Arrays?

- An array is a collection of variables of the same type that are referenced by a common name.
- It is derived data type.

There are different types of arrays used in C++. They are:

- One-dimensional arrays
- Two-dimensional arrays
- Multi-dimensional arrays

2.Define One-dimensional array

- This is the simplest form of an array.
- A one dimensional array represents values that are stored in a single row or in a single column.

Declaration

Syntax:

data type array_name [size];

Example:

int num[10];

3.State the following array declaration are valid or Invalid.

(a)int array [100.5]; - invalid
(b)int a [10]; - valid
(c)char name [15]; - valid
(d)const **j** = 15; double val [**j**]; - valid
(e)int d[ ] ={ 1, 2, 3, 4, 5, 6, 7 } - valid

4.Write a statement for the following.

(a) Read 6 th element = **cin >> n [5]**
(b)assigns the contents of the 4 th element of the array to its 5 th element **= n [4] = n [3]**
(c) increments the value stored as 5 th element by 1 **= n [4] ++**

5.Explain the types of Array Initialization

- An array can be initialized at the time of its declaration.
- Unless an array is initialized, all the array elements contain **garbage** values.

Syntax:

Datatype array_name [size] = {val-1,val-2,. ,val-n};

Ex. int a[3] = { 2,3,4);
a[1] = 5; a[0]= 10;

More examples of array initialization:

```
float  x[5] = {5.6, 5.7, 5.8, 5.9, 6.1};
char   vowel[6] = {'a', 'e', 'i', 'o', 'u', '\0'};
```

Accepting values to an array during run time :

by using **cin** cin>>a[2] ;

```
#include <iostream>
using namespace std;
int main()
{
 int num[5];
 for(int i=0; i<5; i++)
 {
  cin>>num[i];
 }
}
```

In the above program, a for loop has been constructed to execute the statements within the loop for 5 times.

6.How to Accessing array elements

- Array elements can be used anywhere in a program like a normal variable.
- The elements of an array are accessed with the array name followed by the subscript index within the square bracket.

Ex. **cout<<num[3];**

The following for statement is used to display the values.

```
for(int i=0; i<5; i++)
 {
  cin>>num[i];
 }
```

7.What is Traversal in an Array?

- Accessing each element of an array at least once to perform any operation is known as “Traversal”.
- Displaying all the elements in an array is an example of “traversal”.

8.What are strings? Give an example.What is Array of Characters?

- A string is defined as a sequence of characters where each character may be a letter,number or a symbol.
- Each element occupies one byte of memory.
- Every string is terminated by a null (‘\0’)
- a string as an one-dimensional character array.

To declare Character array

Syntax: **char array_name[size];**

```
#include <iostream>
void main()
 {
  char country[6];
```

```
cout<< "Enter the name of the country: ";
cin>>country;
cout<<" The name of the country is "<<country;
}
```

OUTPUT

Enter country the name: INDIA
The country name is INDIA

9.How to Initialize one dimension character array ?

- The character array can be initialized **at the time of** its **declaration**. The syntax is shown below:

char a_nam[size]={ list of characters separated by comma or a string } ;

```
char country[6]="INDIA";
char country[6]={'I', 'N', 'D', 'I', 'A', '\0'};
char country[]="INDIA";
char country[]={'I', 'N', 'D', 'I', 'A', '\0'};
```

- During initialization, the array of elements cannot be initialized more than its size.

```
char str[2]={'5','+','A','B'}; // Invalid
```

10.Define cin.get().

- In C++, **cin.get()** is used to read a line of text **including blank spaces**.
- This function takes two arguments.
- The first argument is the **name of the string** and second argument is the **maximum size** of the array.

```
char str[100];
cin.get(str, 100);
```

11.Define cin.getline()

- In C++, getline() is also used to read a line of text **including blank spaces** from the input stream.
- It can read the characters till it encounters a newline character or a delimiter specified by the user.
- This function is available in the **<string.h>** header.

12.Define Two-dimensional array

- Two-dimensional (2D) arrays are collection of **similar elements** where the elements are stored in certain number of rows and columns.
  ex.**int arr[3][3];**

13.How to Declare 2-D array in C++? Write the syntax of Declaration of 2-D array.

data-type array_name[row-size][col-size];

Ex. int a[3][4];

- **Array size** must be **Positive integer** value
- In arrays, **column size is compulsory** but **row size** is **optional**.

14.How to calculate the size of the array?

one dimension array size = memory required (data type) x No. of the elements in the array

Ex. int n[5] = 2 x 5 = **10** bytes (one integer is 2 bytes)

Two dimension array = Number of elements (Row x column) x memory required

Ex. int n [2] [3] = (2 x 3) x 2 = **12** bytes

15.How to initialize a two dimensional array .

- The array can be initialized in more than one way at the time of 2-D array declaration.
  Ex. int a [2][2] = { {1,2},{3,4}};
- Array's **row** size is **optional** but **column** size is **compulsory**.
- Accessing the two-dimensional array
  A[0][1] = 10; assign 10 to 2nd element of first row

16.Define Memory representation of 2-D array

- A 2-D array is stored in sequential memory blocks.
- There are two types of 2-D array memory representations. They are:

Row-Major order

- In row-major order, all the elements are stored row by row in continuous memory locations,

Column-Major order

- In column-major order, all the elements are stored column by column in continuous memory locations,

17.What is array of string?

- An array of strings is a two-dimensional character array.
- The size of first index (rows) determines the **number of strings** and the size of second index determines **maximum length of each string**.

Ex. char day[2][10] ={"Sunday \0","Monday\0"};

S	u	N	d	a	Y	\0			
M	O	N	D	A	Y	\0			

18.Write a C++ program to accept and print your Name.

```
#include <iostream>
using namespace std;
int main()
{
char n[]="ELANGO";
cout<<"Myname is …"<<n;
}
```

19.How will you pass two dimensional array to a function explain with example?

- In C++, arrays can be passed to a function as an argument.
- The actual parameter is passed **only the array name** as an argument ignoring dimensions.

Passing a two-dimensional array to a function

```
#include <iostream>
using namespace std;
int main()
{
 int marks[5]={88, 76, 90, 61, 69};
 display(marks);
 }
```

```
void display (int m[5])
{
   for (int i=0; i<5; i++)
   {
     cout << m[i];
  }
}
```

20.Write a C++ program to find the sum of two matrix.

```
#include <iostream>
using namespace std;
int i;
int main()
{
 int m1[10][10], m2[10][10], sub[10][10];
 cout<< "Enter the elements of first matrix:\n ";
 for (i = 0;i<2;i++ )
 for (j = 0;j<2;j++ )
 cin>>m1[i][j];
 cout<< "Enter the elements of second matrix:\n ";
 for (i = 0;i<2;i++ )
 for (j = 0;j<2;j++ )
 cin>>m2[i][j];
cout<<"Output: \n";
 for (i = 0;i<2;i++ )
 for (j = 0;j<2;j++ )
            {
            sub[i][j]=m1[i][j] - m2[i][j];
            cout<<sub[i][j]<<'\t';
            }
 }
getch();
}
```

21.Write a C++ program to find the difference between two matrix.

```
#include <iostream>
using namespace std;
int i;
int main()
{
 int m1[10][10], m2[10][10], sum[10][10];

 cout<< "Enter the elements of first matrix:\n ";
 for (i = 0;i<2;i++ )
 for (j = 0;j<2;j++ )
 cin>>m1[i][j];

cout<< "Enter the elements of second matrix:\n ";
 for (i = 0;i<2;i++ )
 for (j = 0;j<2;j++ )
 cin>>m2[i][j];

cout<<"Output: \n";
 for (i = 0;i<2;i++ )
 for (j = 0;j<2;j++ )
            {
            sum[i][j]=m1[i][j]+m2[i][j];
            cout<<sum[i][j]<<'\t';
            }
 }
getch();
}
```

Structures …

22.What is Structure? Or Define Structure.

- Structure is a **user-defined data type**.
- This allows to **group of variables** with **different data types** together into a **single unit.**

23. Declaring and defining structures

Structure is declared using the keyword '**struct**'.

Syntax:

```
struct structure_name
{
type member_name1;
type member_name2;
} object ;
```

24.Define global objects.

- Objects declared along with structure definition are called global objects

25.What is an Anonymous Structure?

- A structure without a name/tag is called anonymous structure.

```
struct
{
long rollno;
int age;
float weight;
} student;
```

26.To store 100 integer number which of the following is good to use? Array or Structure

Array because Array is a set of variable of same data type

27.What is the error in the following structure definition.

```
 struct   employee
{
Int eno;
Char ename[20];
char dept;
}
   Employee e1,e2;
```

- Structure is not terminated with **;**
- Data type and variable name should be separate(int eno; char ename;)
- In Structure tag Employee e should be in small

Correct definition

```
struct   employee
{
int eno;
char ename[20];
char dept;
} ;
   employee e1,e2;
```

28.Write a structure definition for the structure student containing *examno, name and an array for storing five subject marks*.

```
struct student
{
int examno;
char name;
int marks[5];
};
```

29.What is the size of the following highlighted variable in terms of byte if it is compiled in dev c++

struct A{ float f[3]; char ch[5];long double d;};
struct B{ A a; int arr[2][3];}b[3]

struct A{ float f[3]; char ch[5];long double d;};
1.4x3 = 12
2.1x 5 =5
3.2+8 =10
Total = 27 bytes
struct B{ ~~A a;~~ int arr[2][3];}b[3]
4 x 6 = 24
24+24+24 =72bytes

30.Is the following snippet is fully correct. If not identify the error.

```
struct sum1{ int n1,n2;}s1;
struct sum2{int n1,n2}s2;
cin>>s1.n1>>s1.n2;
s2=s1;
```

Error because s1,s2 are separate objects for two separate structures.

31.Differentiate array and structure.

Array	Structure
An array is a collection of variables of the **same** type that are referenced by a **common name**	This allows to **group of variables** with **different data types** together into a **single unit.**
It is a **derived** data type	It is a **user-defined data type**.
There are **different types.**	Only one type

32.How to referencing structure elements in C++? How to success members of a structure ?Give example.

- The structure members can be accessed directly.
- The syntax for that is using a **dot (.)** between the object name and the member name.

Ex. x.rollno , x.age .

33.How pointer type elements reference in structure?

- If the members are a pointer types then **'->'** is used
- to access the members.
- Let name is a character pointer ins student like char * name
- It can be accessed student -> name

34.What are the different ways to initialize the structure members?
How values are assigned to structure elements?
How to Initializing structure elements?

- Values can be assigned to structure elements similar to assigning values to variables.
  balu.rollno= "702016";
  balu.age= 18;
  balu.weight= 48.5;
- Also, values can be assigned directly as similar to assigning values to Arrays.
  balu={702016, 18, 48.5};
- Structures can be assigned directly instead of assigning the values of elements individually.

35.Define Structure Assignments in c++

- Structures can be assigned directly instead of assigning the values of elements individually.
- Structure assignment is possible only if both structure variables/objects are same type.

Ex.

```
struct student
{
I
int age;
float height,weight;
} priya,usha;
priya ={19,165.7,56.4};
usha=priya;
```

It will assign the same age,height and weight to usha.

36.Define nested Structures.

- The structure declared **within another structure** is called a nested structure.

Ex.

```
struct student
{
int age;
struct dob
        {
        Int date;
        char mon[4];
        int year;
        }y;
}x;
void main()
{
cin>>x.age >>x.y.date;
}
```

37.Explain Array of Structure with an example.

- An array of structures is declared in the same way as declaring an array with built-in data types like int or char.

For example

- If the class has 20 students, then 20 individual structures are required.
- For this purpose, an array of structures can be used.

```
#include <iostream.h>
struct student
{
 int  age;
 float height, weight;
 char name[30];
};
void main( )
{
student std[20];
int i;
 for(i=0;i<20;i++)
{
 cout<< " Enter the age:"<< '\n';    cin>>std[i].age;
 cout<< "Enter the height:"<< '\n';    cin>>std[i].height;
 cout<< "Enter the weight:"<< '\n';    cin>>std[i].weight;
}
 cout<< "To enter the value…<'\n';
 for(i=0;i<20;i++)
cout<<"Student "<<i+1<< "\t"<<std[i].age<<
"\t"<<std[i].height<< "\t"<<std[i].weight;    }
```

38.Explain call by value with respect to structure in c++
Explain call by reference with respect to structure in c++
What are the method to pass structures to function?

A structure variable can be passed to a function in two types 1.call by value 2. Call by reference

Call by value.

- When a structure is passed as argument to a function using call by value method,
- Any change made to the contents of the structure **do not affect** the argument of the function.

```
#include <iostream>
using namespace std;
struct employee
{
char name[50];
float salary;
};
void printdata(employee q)
{
 cout<<"\nDisplay..";
 cout<< "\nName: " << q.name ';
   cout<< "\nSalary: " <<q.salary;
}
int main()
{
 employee p;
 cout<< "\nEnter Full name: ";
 cin>>p.name;
 cout<< "\nEnter salary: ";
 cin>>p.salary;
printdata(p);
}
```

Output:

Enter Full name: **Kumar**
Enter salary: **34000.0**
Display
Name: **Kumar**
Salary: **34000.0**

- In the above example, a structure name is **employee**
- The values are name, age and salary .
- A function **printdata()** used to display employee.

Call by reference

- In this method, the **address of a structure** variable /object is passed to the function using address of(**&**) operator.
- So Any change made to the contents of the structure **affect** the argument of the function.
- Structures are usually passed by **reference** method because it saves the **memory space** and executes **faster**.

```
#include <iostream>
using namespace std;
struct employee
{
char name[50];
float salary;
};
void printdata(employee  &q)
{
 cout<<"\nDisplay..";
 cout<< "\nName: " << q.name ';
   cout<< "\nSalary: " <<q.salary;
}
void main()
{
 employee p;
 cout<< "\nEnter Full name: ";
 cin>>p.name;
 cout<< "\nEnter salary: ";
 cin>>p.salary;
printdata(p);
}
```

Output:
Enter Full name: **Kumar**
Enter salary: **34000.0**
Display
Name: **Kumar**
Salary: **34000.0**