

STD: XI

OCTOBER MONTHLY TEST

Lessons: 11 To 13

Time: 3.00 Hrs.

COMPUTER SCIENCE

Marks: 70

Note: i) Answer all the questions.

PART-I

15 X 1 = 15

ii) Choose the most suitable answer and write the code with corresponding answer.

1. which function begins the program execution ?

- a) isalpha() b) isdigit() c) main() d) islower()

2. which of the following function is with a return value and without any argument ?

- a) x=display(int,int) b) x=display()
c) y=display(float) d) display(int)

3. which of the following is the scope operator ?

- a) > b) & c) % d) ::

4. A structure declaration is given below.

Struct employee

```
{
int empno;
char ename[10];
}e[5];
```

Using above declaration which of the following statement is correct.

- a) .cout<<e[0].empno<<e[0].ename; b) . cout<<[0].empno<<ename;
c) .cout<<e[0]->empno<<e[0]->name; d) . cout<<e.empno<<e.ename;

5. Which of the following is a properly defined structure?

- a) struct {int num;} b) struct sum {int num;}
c) struct sum int sum; d) struct sum {int num;}

6. cin>>n[3]; To which element does this statement accept the value?

- a) 2 b) 3 c) 4 d) 5

7. which of the following supports the transitive nature of data?

- a) Inheritance b) Encapsulation c) Polymorphism d) abstraction

8. Insulation of the data from direct access by the program is called as

- a) data hiding b) encapsulation c) Polymorphism d) abstraction

9. The identifiable entity with some characteristics and behavior is.

- a) class b) object c) structure d) member

10. The __ function tells the compiler that the function returns nothing.

- a) display() b) Compute() c) get() d) void()

11. A __ function looks like normal function.

- a) main() b) Built in c) inline d) String

12. In C++ the pow() function take ----- arguments

- a) 1 b) 2 c) 3 d) 4

13. A function calls itself is known as -----

- a) Built in function b) Library function
c) predefined function d) Recursive function

14. The array size should be specified with -----,
 a) () b) [] c) { } d) {[]}
15. In c++ the attributes are also called -----,
 a) Data variable b) function data
 c) data members d) member function

PART-II

II. Answer any six Questions. Q. No. 24 is compulsory.

6 X 2 = 12

16. Write short note about void data type. 17. What is recursive function?
 18. Write the purpose of "Struct" keyword.
 19. Define a global object with give an example.
 20. Differentiate array and structure. 21. What is member function?
 22. What is Traversal in array? 23. What is polymorphism?
 24. Write the short note about the Pow() function with an its arguments and its example.

PART-III

III. Answer any six Questions. Q. No. 33 is compulsory.

6 X 3 = 18

25. Write short note about : i) get char() ii) put char() iii) gets() iv) puts with an example.
 26. Write the syntax of declaring const modification with an example.
 27. How to access members of a structure? Give an example.
 28. Write the syntax of character array can be initialized at the time of its declaration with an example.
 29. How you will declaring an array of strings with an example.
 30. Write the important features of modular programming with give an example programming language.
 31. Define an array? What are the types.
 32. List some of the features of modular programming.
 33. Write about structure assignment with an example.

PART-D

IV. Answer the following questions:

5 X 5 = 25

34. a) Explain about character functions with its syntax. [OR]
 b) Explain scope of variable with an example
35. a) Explain about default arguments with an example of C++ program. [OR]
 b) Write a program to accept any integer number and reverse it.
36. a) Write difference between object oriented programming and procedural programming. [OR]
 b) How will you pass two dimensional array to a function explain with example..
37. a) Explain detail about advantages of OOPs. [OR]
 b) Write the important features of object oriented programming and list out the OOPs programming languages.
38. a) Write a C++ program to display values from two-dimensional array. [OR]
 b) Explain call by value with respect to structure.

MOUNT CARMEL MISSION MATRIC HR. SEC. SCHOOL – KALLAKURICHI

STD: XI

OCTOBER MONTHLY TEST

LESSON: 11 – 13

TIME: 3 : 00 Hrs

COMPUTER SCIENCE

MARK: 70

Part – I

I. CHOOSE THE CORRECT ANSWER:

15 X 1 = 15

- | | |
|----------------------------------|---------------------------|
| 1. c) main() | 11. c) inline |
| 2. b) x = display() | 12. b) 2 |
| 3. d) :: | 13. d) recursive function |
| 4. a) cout<<e[0].empno[0].ename; | 14. b) [] |
| 5. d) struct sum { int num; }; | 15. c) data members |
| 6. c) 4 | |
| 7. a) Inheritance | |
| 8. a) data hiding | |
| 9. b) object | |
| 10. d) void() | |

Part – II

II. ANSWER ANY SIX QUESTIONS Q. No: 24 IS COMPULSORY:

6 X 2 = 12

16. write short note about void data type.

Ans: void data type has two important purpose:

- i. To indicate the function does not return a value
- ii. To declare a generic pointer

17. what is recursive function?

Ans: A function that calls itself is known as recursive function.

Example: int sum(int x)

```
{
    .....
    .....
    c=sum(x-1);
    ..
}
```

18. write the purpose of “struct” keyword.

Ans: Structure is declared using the keyword “struct”. The syntax of creating a structure is given below,

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

19. define a global object with give an example.

Ans: If an object is declared outside all the function bodies or by placing their names immediately after the closing brace of the class declaration then it is called as Global object. These objects can be used by any function in the program.

Example:

```
// Declaring outside all the function bodies
```

```
class global
{
public:
    int a,b;
    }x,y;
    Global m,n;
int main()
{
    .....
}
```

m, n, x and y are global objects.

20. differentiate array and structure.**Ans:**

Array	Structure
i. An array is a collection of variables of the same data type that are referenced by a common name.	i. structure is a user-defined which has the combination of data items with different data types.
ii. syntax: <data type><array_name> [<array_size>; <data type> - declares the basic data type of the array <array_name> - specifies the name with which the array will be referenced <array_size> - defines how many elements the array will hold. Size should be specified with square brackets []	ii. syntax: struct structure_name{ type member_name1; type member_name2; } reference_name;
iii. example: int num[10];	iii. example: struct Student { long rollno; int age; float weigh; };

21. what is member function?

Ans: member functions are the functions that perform specific tasks in a class. Member functions are called as methods.

example:

```
class example
{
public:
int x,y;
void print()
{ cout<<x<<y; }
};
```

Name of the class: example
Data members: x and y
Member function: void print();

22. what is traversal in array?

Ans: 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”.

23. what is polymorphism?

Ans: polymorphism is the ability of a message or function to be displayed in more than one form. Polymorphism is achieved by overloading.

24. write the short note about the pow() function with an its arguments and its example.

Ans: The pow() function returns base raised to the power of exponent. If any argument passed to pow() is long double, the return type is promoted to long double. If not, the return type is double.

The pow() function takes two arguments:

- i. base – the base value
- ii. exponent – exponent of the base

example:

```
cout<<pow(5,2);
```

output:

25

Part – III**III. ANSWER ANY SIX QUESTIONS Q. No: 33 IS COMPULSORY:****6 X 3 = 18****25. write short note about: i) getchar() ii) putchar() iii) gets() iv) puts() with an example.**

Ans: The predefined functions getchar(), putchar(), gets(), puts() are standard I/O functions which is accessed by the header file with “stdio.h”

i) getchar(): It is used to get a single character from keyboard

example: char ch = getchar();

ii) putchar(): It is used to display the received character from keyboard

example: putchar(ch);

iii) gets(): This function reads a string from standard input and stores in into the string pointed by the variable.

example: char str[50];
cout<<”Enter a string:”;
gets(str);

iv) puts(): This function prints the string read by gets() function in a newline.

example: cout<<”you entered:”;
puts(str);

26. write the syntax of declaring const modification with an example.

Ans: The constant variable can be declared using **const** keyword. The **const** keyword makes variable value stable. The constant variable should be initialized while declaring. The **const** modifier enables to assign an initial value to a variable that cannot be changed later inside the body of the function.

Syntax :

<returntype><functionname> (const <datatype variable=value>)

Example:

- int minimum(const int a=10);
- float area(const float pi=3.14, int r=5);

If the variable value “r” is changed as **r=25**; inside the body of the function “area” then compiler will throw an error as “assignment of read-only parameter 'r'”

```
double area(const double r,const double pi=3.14)
```



```
{
r=25;
return(pi*r*r);
}
```

27. how to access members of a structure? Give an example.

Ans: once objects of a structure type are declared, their members can be accessed directly. The syntax for that is using a dot(.) between the object name and the member name.

example: student.name;

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

28. write the syntax of character array can be initialized at the time of its declaration with an example.

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

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

For example,

char country[6]="INDIA";

In the above example, the text "INDIA" has 5 letters which is assigned as initial value to array country.

The text is enclosed within double quotes.

In the above memory representation, each character occupies one byte in memory. At the end of the string, a null character is automatically added by the compiler. **C++ also provides other ways of initializing the character array:**

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

char country[]="INDIA";

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

29. how you will declaring an array of strings with an example.

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

Usually, array of strings are declared in such a way to accommodate the null character at the end of each string.

For example, the 2-D array has the declaration:

char Name[6][10];

In the above declaration, the 2-D array has two indices which refer to the row size and column size, that is 6 refers to the number of rows and 10 refers to the number of columns.

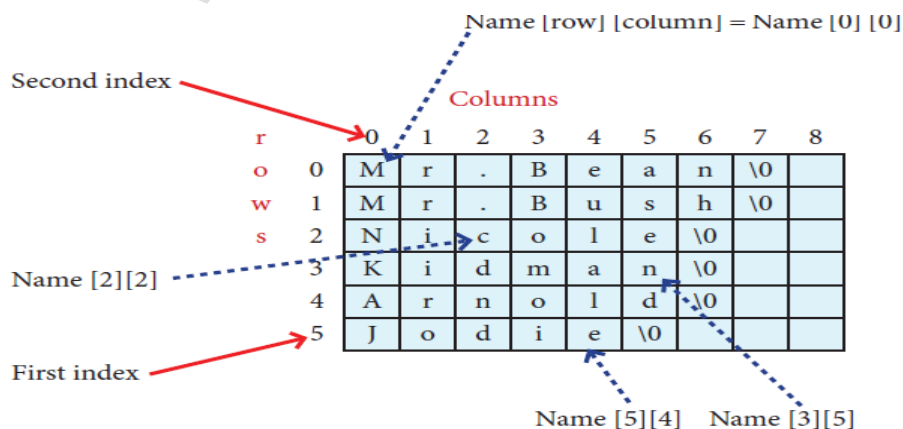
Initialization

For example

char Name[6][10] = {"Mr. Bean", "Mr.Bush", "Nicole", "Kidman", "Arnold", "Jodie"};

In the above example, the 2-D array is initialized with 6 strings, where each string is a maximum of 9 characters long, since the last character is null.

The memory arrangement of a 2-D array is shown below and all the strings are stored in continuous locations.



30. write the important features of modular programming with give an example programming language.

Ans: Important features of Modular programming

- Emphasis on algorithm rather than data
- Programs are divided into individual modules
- Each modules are independent of each other and have their own local data
- Modules can work with its own data as well as with the data passed to it.
- Example: **Pascal** and **C**

31. define an array? what are the types?

Ans: An array is a collection of variables of the same type that are referenced by a common name. types of arrays are,

- i. one-dimensional array
- ii. two-dimensional array
- iii. multi-dimensional array

32. list some of the features of modular programming.

Ans:

- Emphasis on algorithm rather than data.
- Programs are divided into individual modules
- Each modules are independent of each other and have their own local data.
- Modules can work with its own data as well as with data passed to it.
- **example:** Pascal and C

33. write about structure assignment with an example.

Ans: 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.

example:

If Mahesh and Praveen are same age and same height and weight then the values of Mahesh can be copied to Praveen

```
struct Student
{
int age;
float height, weight;
}mahesh;
```

The age of Mahesh is 17 and the height and weights are 164.5 and 52.5 respectively. The following statement will perform the assignment.

```
mahesh = {17, 164.5, 52.5};
```

```
praveen =mahesh;
```

will assign the same age, height and weight to Praveen.

Part – IV

IV. ANSWER THE FOLLOWING QUESTIONS:

5 X 5 = 25

34. a) explain about character functions with its syntax.

Ans: This header file defines various operations on characters. Following are the various character functions available in C++. The header file **ctype.h** is to be included to use these functions in a program.

1. isalnum()

This function is used to check whether a character is **alphanumeric or not**. This function returns non-zero value if c is a digit or a letter, else it returns 0.

General Form:

```
int isalnum (char c)
```

Example:

```
int r = isalnum('5');
cout << isalnum('A') << '\t' << r;
```

But the statements given below assign 0 to the variable n, since the given character is neither an alphabet nor a digit.

```
char c = '$';
int n = isalnum(c);
```

```
cout << c;
```

Output:

0

2. isalpha()

The isalpha() function is used to check whether the given character is an alphabet or not.

General Form:

```
isalpha(char c)
```

This function will return 1 if the given character is an alphabet, and 0 otherwise. The following statement assigns 0 to the variable n, since the given character is not an alphabet.

```
int n = isalpha('3');
```

But, the statement given below displays 1, since the given character is an alphabet.

```
cout << isalpha('a');
```

3. isdigit()

This function is used to check whether a given character is a digit or not. This function will return 1 if the given character is a digit, and 0 otherwise.

General Form:

```
isdigit(char c)
```

4. islower()

This function is used to check whether a character is in lower case (small letter) or not. This functions will return a non-zero value, if the given character is a lower case alphabet, and 0 otherwise.

General Form:

```
islower(char c)
```

After executing the following statements, the value of the variable **n** will be **1** since the given character is in lower case.

```
char ch = 'n';
int n = islower(ch);
```

But the statement given below will assign 0 to the variable n, since the given character is an uppercase alphabet.

```
int n = islower('P');
```

5. isupper()

This function is used to check the given character is uppercase. This function will return 1 if true otherwise 0.

General Form:

```
isupper(char c)
```

For the following examples value **1** will be assigned to **n** and **0** for m.

```
int n=isupper('A');
int m=isupper('a');
```


6. toupper()

This function is used to convert the given character into its uppercase. This function will return the upper case equivalent of the given character. If the given character itself is in upper case, the output will be the same.

General Form:

char toupper(char c);

The following statement will assign the character constant 'K' to the variable c. **char c = toupper('k');**

But, the output of the statement given below will be 'B' itself. **cout <<toupper('B');**

7. tolower()

This function is used to convert the given character into its lowercase. This function will return the lower case equivalent of the given character. If the given character itself is in lower case, the output will be the same.

General Form:

char tolower(char c)

The following statement will assign the character constant 'k' to the variable c.

char c = tolower('K');

But, the output of the statement given below will be 'b' itself.

cout <<tolower('b');

[OR]

b) explain scope of variable with an example.

Ans: scope refers to the accessibility of a variable.

There are four types of scopes in C++.

They are i. Local scope ii. Function scope iii. File scope iv. Class scope

i. Local scope:

- A local variable is defined within a block. A block of code begins and ends with curly braces { }.
- The scope of a local variable is the block in which it is defined
- A local variable cannot be accessed from outside the block and destroyed upon exit.
- A local variable is created upon entry into its block and destroyed upon exit.

example:

```
int main()
{ int a,b } // variables a and b are local variables to main()
```

ii. Function scope:

- The scope of variables declared within a function is extended to the function block, and all sub-blocks therein.
- The life time of a function scope variable, is the life time of the function block.

example:

The scope of formal parameters is function scope.

```
int sum(int x, int y) // x and y has function scope
```

iii. File scope:

- A variable declared above all blocks and functions (including main()) has the scope of a file.
- The lifetime of a file scope variable is the life time of a program.
- the file scope variable is also called as global variable.

example:

```
#include<iostream>
using namespace std;
int x,y; // x and y are global variables
void main()
{ }
```

iv. Class scope:

- Data members declared in a class has the class scope.
- Data members declared in a class be accessed by all member functions of the class.

example:

```
class example
```

```
{
int x,y; // x and y can be accessed by print() and sum()
void print();
void sum();
};
```

35. a) explain about default arguments with an example of c++ program.

Ans: default values can be assigned to the formal parameters of a function prototype. The default arguments allows to omit some arguments when calling the function.

When calling a function,

- for any missing arguments, compiler uses the values in default arguments for the called function.
- The default value is given in the form of variable initialization.
- The default arguments facilitate the function call statement with partial or no arguments.

example: void defaultvalue(int n1=10, int n2=100);

example: defaultvalue(x,y); //n1=x, n2=y

defaultvalue(200,150); //n1=200, n2=150

defaultvalue(150,100); // n1=150, n2=100 (default value)

defaultvalue(x,150); // n1=x, n2=150

[OR]

b) write a program to accept any integer number and reverse it.

Ans:

```
#include<iostream>
using namespace std;
int main()
{
int num, rnum=0, r;
cout<<"Enter a positive number:";
cin>>num;
while(num>0)
{
r = num%10;
num = num/10;
rnum = rnum*10+r;
}
cout<<" The Reversed Number is ="<<rnum;
return 0;
}
```

36. a) write difference between object oriented programming and procedural programming.

Ans:

Object oriented programming	Procedural programming
Emphasizes on data rather than algorithm	Procedural programming aims more at procedures
It implements programs using classes and objects	Programs are organized in the form of subroutines or sub programs.
Data abstraction is introduced in addition to procedural abstraction.	All data items are global
It is easy to maintain and modify existing code as new objects can be created with small differences to existing ones.	Difficult to maintain and enhance the program code as any change in data type needs to be propagated to all subroutines that use the same data type. This is time consuming.
Supports encapsulation and polymorphism.	Do not support encapsulation and polymorphism.
Suitable for all small and large software applications.	Suitable for small sized software application.
Example: C++, java, Vb.net, python	Example: FORTRAN and COBOL

[OR]

b) How will you pass two dimensional array to a function explain with example.

Ans: In C++, we can pass arrays as an argument to a function. And, also we can return arrays from a function.

Syntax for Passing Arrays as Function Parameters

The syntax for passing an array to a function is:

```
returnType functionName(dataType arrayName[arraySize]) {
//code
}
```

Let's see an example,

```
int total(int marks[5]) {
//code
}
```

Here, we have passed an `int` type array named `marks` to the function `total()`. The size of the array is `5`.

//C++ Program to display the elements of two dimensional array by passing it to a function

```
#include<iostream>
using namespace std;
//define a function
// pass a 2d array as a parameter
void display(int n[ ][2])
{
    cout<<"displaying values:"<<endl;
    for(int i = 0; i < 3; ++i)
    {
        for(j=0; j < 2; ++j)
        {
            cout<<"num["<<i<<"]["<<j<<"]:"<<n[i][j]<<endl;
```

```

}
}
}
int main( )
{
// initialize 2d array
int num[3] [2]= { {3, 4}, {9, 5}, {7, 1}};
// call the function
// pass a 2d array as an argument
display(num);
return (0);
}

```

OUTPUT:

Displaying Values:

```

num[0][0]: 3
num[0][1]: 4
num[1][0]: 9
num[1][1]: 5
num[2][0]: 7
num[2] [1]: 1

```

In the above program, we have defined a function named `display()`. The function takes a two dimensional array, `int n[][2]` as its argument and prints the elements of the array.

While calling the function, we only pass the name of the two dimensional array as the function argument `display(num)`.

37. a) explain detail about advantages of OOPs.

Ans:

- **Re-usability:** "Write once and use it multiple times" you can achieve this by using class.
- **Redundancy:** Inheritance is the good feature for data redundancy. If we need a same functionality in multiple class we can write a common class for the same functionality and inherit that class to sub class.
- **Easy maintenance:** It is easy to maintain and modify existing code as new objects can be created with small differences to existing ones.
- **Security:** using data hiding and abstraction only necessary data will be provided thus maintains the security of data.

[OR]

b) write the important features of object oriented programming and list out the OOPs programming languages.

Ans: Important features of Object oriented programming:

- Emphasizes on data rather than algorithm
- Data abstraction is introduced in addition to procedural abstraction
- Data and its associated operations are grouped in to single unit
- Programs are designed around the data being operated
- Relationships can be created between similar, yet distinct data types
- **Example:** C++, Java, VB.Net, Python etc.

Main Features of Object Oriented Programming

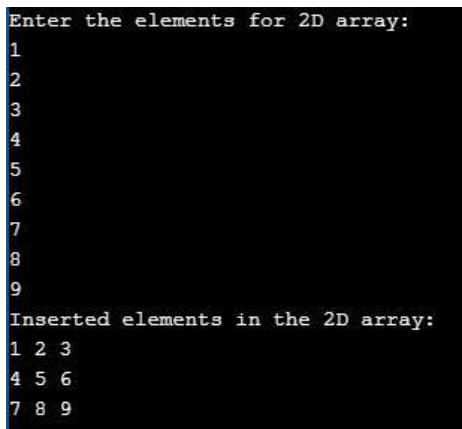
- Data Abstraction
- Encapsulation
 - Modularity
 - Inheritance
 - Polymorphism

38. a) write a C++ program to display values from two-dimensional array.

Ans:

```
#include <iostream>
using namespace std;
int main()
{
    int array2d[3][3]; // 2D array declared
    int i, j; // variables for loop iteration
    cout << "Enter the elements for 2D array:" << endl;
    for(i = 0; i <= 2; i++)
    {
        for(j = 0; j <= 2; j++)
        {
            // inserting the elements in the 2D array
            cin >> array2d[i][j];
        }
    }
    cout << "Inserted elements in the 2D array:" << endl;
    for(i = 0; i <= 2; i++)
    {
        for(j = 0; j <= 2; j++)
        {
            cout << array2d[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

OUTPUT:



```
Enter the elements for 2D array:
1
2
3
4
5
6
7
8
9
Inserted elements in the 2D array:
1 2 3
4 5 6
7 8 9
```

[OR]

b) explain call by value with respect to structure.

Ans: This method copies the value of an actual parameter into the formal parameter of the function. in this case, changes made to formal parameter within the function will have no effect on the actual parameter.

example:

```
#include<iostream>
using namespace std;
void display(int x)
{
    int a=x * x;
    cout<<"\n\n The Value inside display function ( a * a ):"<<a;
}

int main()
{
    int a;
    cout<<"\n Example: Function call by value";
    cout<<"\n\n Enter the Value for A:";
    cin>>a;
    display(a);
    cout<<"\n\n The Value inside main function:"<<a;
    return(0);
}
```

OUTPUT:

Example: Function call by value

Enter the Value for A: 5

The Value inside display function (a * a): 25

The Value inside main function: 5

– Prepared By
S.Vinoth Kumar,
B.Sc.(CS), MCA., B.Ed.,
Mobile Number: (+91)9786845143,
PG.Asst.in Computer Science Dept.,
Mount Carmel Mission Matriculation
Higher Secondary School,
Kallakurichi – 606 202.