# Chapter 3

# C++

**Scope of the Syllabus** | **Probable marks : 41**

- Review of C++
- Arrays, pointers, references, strings
- Principle of object oriented programming
- Classes and objects
- Constructors and destructors
- Operator overloading and type conversions
- Inheritance
- Virtual functions and polymorphism
- Working with files

## REVIEW OF C++

Q. 1 What is C++ ? What are the advantages of C++ ? **(March 2003, 13)**

Ans. :

C++ is an object oriented programming language. Initially C++ was named as "C with classes". C++ was developed by Bjarne Stroustrup at AT & T Bell Laboratories, USA, in the early eighties.

The advantages of C++ over C are :

(i) C++ is an incremented version of C. It is a superset of C. Almost all C programs can also run in C++ compiler.

(ii) The important facilities added in C++ are classes, function overloading, operator overloading.

(iii) C++ allow user to create abstract data types, to inherit properties from existing data types.

(iv) C++ supports polymorphism.

(v) Any real life application systems such as editor, compiler, databases, communication systems can be built by C++.

(vi) Object oriented libraries can be built by C++.

(vii) C++ programs can be easily implemented, maintained and expanded.

Q. 2 Differentiate between traditional procedural programming approach and object oriented programming approach. **(Oct. 2002, 2005; March 2011, March 2018**

Ans. :

The differences between traditional procedural programming approach and object oriented programming approach are as follows :

| | Traditional Procedural Programming Approach | | Object Oriented Programming Approach |
|---|---|---|---|
| 1 | In this approach, the problem is viewed as a sequence of things to be done. | 1 | In this approach, the problem is decomposed into a number of entities called objects and then builds data and function around these entities. |
| 2 | Emphasis is on doing things. | 2 | Emphasis is on the data rather than procedure. |
| 3 | Large programs are divided into smaller programs known as functions. | 3 | Programs are divided into entities known as objects. |
| 4 | Data move openly around the system from | 4 | Data is hidden and cannot be accessed by external functions. |
| 5 | Employs top-down approach in program design. | 5 | Follows bottom-up approach in program design. |

Q. 3 What do you mean by Object Based Programming Language and Object Oriented Programming Language ? State the relationship between these languages. **(Oct. 2008, March 2010, 3**

Ans :

Object Based Programming Language :

1) Language that supports programming with objects are said to be object based programming languages.
2) It is a style of programming that primarily supports encapsulation & object or identity.
3) Major features are :
   a) Data encapsulation
   b) Data hiding & access mechanism
   c) Automatic initalization & clean-up objects
   d) operator overloading.
4) They do not support inheritance .& dynamic binding.
5) For eg-Ada.

Object - Oriented Programming Language :

1) This language incorporates all the object based features along with inheritance and dynamic binding.
2) For eg. — C++, Smalltalk.

The relation between them characterized by following statement :

Object oriented programming =. object—based features + inheritance + dynamic binding.

Q. 4 State any six principal advantages of Object Oriented programming. **(March 2012, 3)**

Ans.:

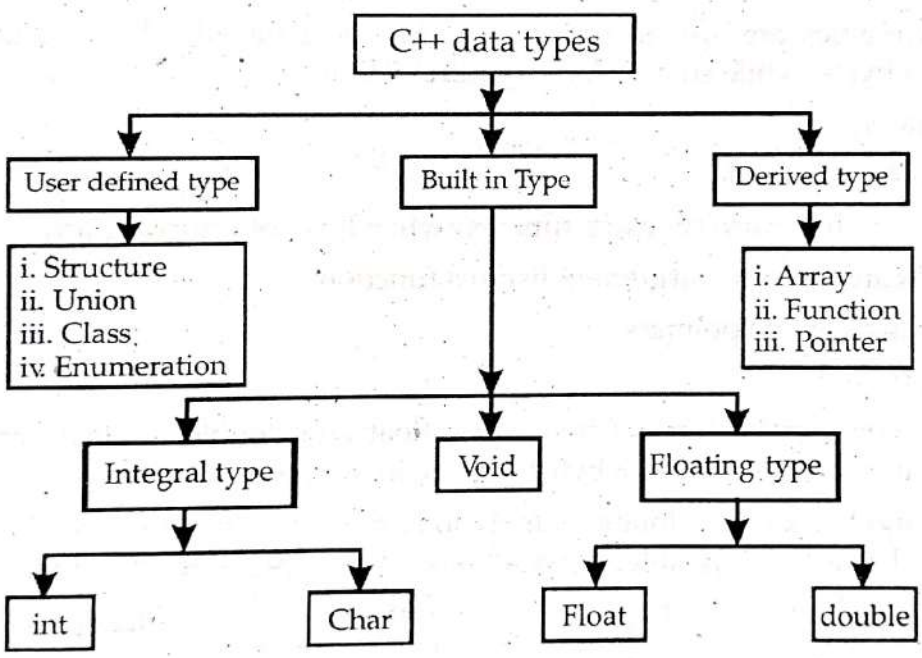
Advantage of Object Oriented Programming:

1. Through, inheritance, eliminate redundant code and extend the use of existing classes.
2. The principle of data hiding helps the programmer to build secure program.
3. It is possible. to have multiple instances of ,an object to co-exist without any interference.
4. It is easy to partition the work in a; project based on object.
5. OOP system can be easily upgraded from small to large system.
6. Software complexity can, be easily managed.
7. It is possible to map objects in the problem domain to objects in the program.
8. Good message passing technique for communication between objects.

Q.5 What are the different data types in C++ ? **(Oct.2015)**

Ans. :

1) Data types in C++ are shown in figure below :




2) C++ allows user to create new abstract data types, which can behave like any buit-in data type. These are called user-defined data types. These include structure, union, class and enumeration.

3) C++ provides three built-in data types which are integral, void and floating.

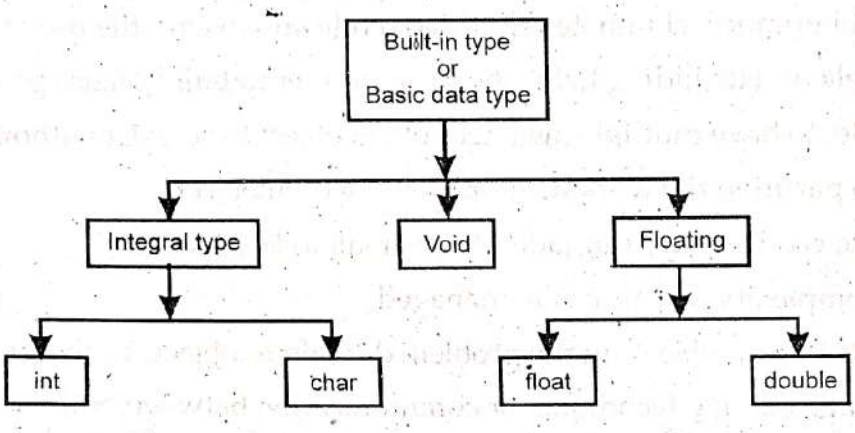
4) Integral includes integer and character (string) while floating type includes float and double.

5) In addition to these data types, C++ provides user with arrays, functions and pointers, which are referred as derived data types.

Q. 6 Enlist the basic data types used in C++ with size of data in terms of bytes for each. (March 2002, 2006, October 2006)

OR Enlist different built in data types in C++ with their sizes. (Oct. 2009)

Ans. : There are three main basic built-in data types used in C++ viz. integral type, void and floating type.




i) Integral data type :

It includes integer (int) and character (char).

An int variable requires 2 bytes to store, while a character variable requires 1 byte.

Integer variables are also of two types : (a) short int and (b) long int. Long integer requires 4 bytes, while short integer requires 2 bytes.

ii) Void data type :

Void data type is used :

(a) to specify the return type of a function when it is not returning any value.

(b) to indicate an empty argument list to a function.

(c) to declare generic pointers.

iii) Floating type : (July 2019)

Floating type variables are of two types; float and double. A float variable requires 4 bytes, while double requires 8 bytes to store in memory.

There is another kind of double namely long double, which requires 10 bytes to store in memory. The following table shows all basic data types, size and range :

| Sr. No. | Type | Bytes | Range |
|---|---|---|---|
| 1 | char (Signed char) | 1 | – 128 to 127 |
| 2 | unsigned char | 1 | 0 to 255 |
| 3 | int (short int or signed int) | 2 | – 32768 to 32767 |
| 4 | unsigned int | 2 | 0 to 65535 |
| 5 | float | 4 | $3.4 \times 10^{-38}$ to $3.4 \times 10^{-38}$ |
| 6 | double | 8 | $1.7 \times 10^{-308}$ to $1.7 \times 10^{308}$ |
| 7 | long double | 10 | $3.4 \times 10^{-4932}$ to $3.4 \times 10^{-4932}$ |

Q. 7 Explain insertion and extraction operators in C++. **(Mar. 2012)**

Ans. :

(i) Insertion operator :

The operator "<<" is called as insertion operator. It is also called as "put to" operator. It inserts the contents of the variables on its right to the object on its left.

It is generally used in output statement in C++.

e.g. (i) Cout << a ;.

(ii) Cout << "program";

In first example, the value of variable 'a' is printed on screen, while in second example the word "program" is printed on screen.

(ii) Extraction operator :

The operator ">>" is called as extraction operator. It is also called as 'get from' operator. It extracts or takes the value from keyboard and assigns it to a variable on its right. It is used in input statement in C++.

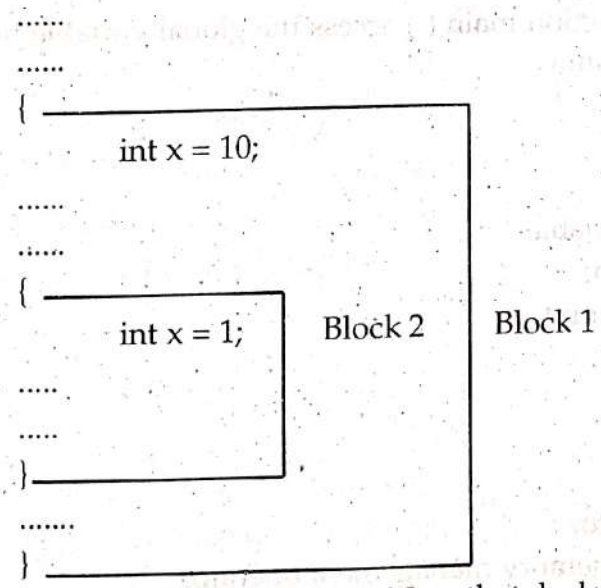
e.g. cin >> a;

This instruction will extract a value from keyboard and assign it to the variable a. C++ allows us to redefine insertion and extraction operators by overloading them.

Q. 8 Write a short note on scope resolution operator. **(Oct. 2014; Mar.16**

Ans. :

1) The operator :: is called as scope resolution operator.
2) C++ is a block structured language i.e. a C++ program may contain one block within another block.
3) When a variable is declared in program, scope extends from the point of declaration till the end of the block in which it is defined.
4) The same variable name can be used to have different meaning in different blocks.
5) Consider the following segment of program.




Here Block 2 is contained in Block 1. Note that declaration of a variable in an inner block hides the declaration of the same variable in an outer block.

6) Scope resolution operator is used to uncover a hidden variable.
It takes the form

:: variable name

e.g.

```
.....
.....
{
    int x = 10;
    .....
    {
    int x = 1;
    cout << "Local x is" << x;
    cout << "\n Global x is" << :: x;
.....
    }
.....
}
```

The output will be as follows :
Local x is 1
Global x is 10

---

Q. 9 Explain the use of scope resolution operator and memory management operators in C++ with examples. (March 2004,16, 17

Ans. : Scope resolution operator :

1) In C++, scope resolution operator (: :) is used to access a global variable from a function in which a local variable is defined with the same name as a global variable.

2) For example :
In following program, the function main ( ) access the global variable num and also the local variable with the same name.

```
int num = 20
void main ( )
{
    int num = 10; // local variable
    cout <<"Local =" << num;
    cout <<"Global =" << : : num;
}
```

The output is as :
Local = 10
Global = 20

Memory management operator :

(1) C++ provides following two memory management operator :
(i) new (ii) delete

(2) The new operator obtains memory block from operating system and returns a pointer to its starting point. The new operator returns NULL, if memory allocation is unsuccessful. The general format of new operator is :

DataType * new DataType [size in integer];

(3) The delete operator is used to return the memory allocated by the new operator back to the memory pool. Thus released memory will be reused by other parts of the program. The general format of delete operator is :

delete pointervariable;

(4) For example :

```
void main ( )
{   char * str = "COMPUTER";
    int len = strlen (str);
    char * ptr;
    ptr = new char [len + 1];
    strcpy (ptr, str);
    cout << "ptr =" << ptr;
    delete ptr;
}
```

In above example, the new operator returns a pointer that point to a memory section large enough to hold the string str plus an extra byte for null character. Then after use of memory delete operator released memory.

Q. 10 What are the different selection (conditional) statements in C++? Give syntax for each.

Ans.

The program has to be able to evaluate conditions and select alternative path in program.

In C++, there are two ways in which selection may be made :

1) The if statement 2) The switch statement

1) The if statement :

The if statement has two forms :

i) Simple if statement ii) if _____ else statement

i) Simple if statement :

Syntax :

```
if (condition)
{
    action 1;
}
    action 2;
```

1) Depending on the condition value, program execution proceeds in one direction or another.

2) If the condition is true, then action 1 will be done.

ii) if .... else statement :

Syntax :

```
if (condition)
{
action 1;
}
else
{
action 2;
}
action 3;
```

If the condition is true, then and then only action 1 will be done, otherwise action 2 will be done.

2) The switch statement :

1) This is a multiple branching statement.
2) Depending on certain condition, it executes only one module out of several. If no condition is satisfied, then default module will be executed.
3) The break statement is used to terminate switch statement.
4) Expression must have int or char value.

Syntax :

```
switch (expression)
{
case 1 :
    {
        action 1;
        break;
    }
case 2 :
    {
        action 2;
        break;
    }
:
:
default :
    {
    action x;
    }
}
```

Q. 11 What are the different looping structures in C++ ? Give syntax for each.

Ans. :

Following are the different looping structures in C++ :

1) For loop 2) While loop 3) Do-while loop

1) The for loop :

The for loop is an entry-controlled loop. It is used when action is to be repeated predetermined number of times.

Syntax :

```
for (initial expression; test expression; increment / decrement expression)
{ .....
action;
} .....
```

where :

(a) Initial expression is executed only once, when the loop starts.

(b) Test-expression evaluated each time through the loop, before the body of the loop is executed.

(c) Increment / Decrement expression changes the value of the loop variable at the end of the loop.

2) The while loop :

The while loop is an entry-controlled loop and it repeats the action until the condition becomes false. When condition is false, that time loop is terminated.

Syntax :

```
while (condition)
{
    action 1;
}
action 2;
```

3) The do-while loop :

The do-while loop is an exit-control loop used to carry out conditional looping.

Syntax :

```
do
{
    action 1;
}
while (condition);
action 2;
```

In do-while, condition is not tested until the body of the loop has been executed once. So even if the condition is false the loop is executed at least once. If the condition is false after the first iteration, the loop is terminated.

Q. 12 What is function prototyping ?

Ans. :

1) Function prototyping is one of the major improvements added to C++ functions.
2) The prototype describes the function interface to the compiler by giving details such as the number and the type of arguments and the type of return values.
3) With function prototyping, a template is always used when declaring and defining a function.
4) When a function is called, the compiler uses the template to ensure that proper arguments are passed, and the return value is treated correctly.
5) Any violation in matching the arguments and the return type will be caught by the compiler at the time of compilation itself.
6) Function prototype is a declaration statement in the calling program and is of the following form

```
return-type function-name (argument-list);
```

The argument list contains the types and names of arguments that must be passed to the function.

e.g.

float volume (int x, float y, float z);

Note that each argument variable should be declared independently. The combined declaration like :

float volume (int x, float y, z); is invalid.

7) In function declaration, the names of arguments are the dummy variables and therefore, they are optional i.e. the declaration :-

float volume (int, float, float); is valid.

Q. 13 Write a short note on inline functions.

Ans. :

1) When a function is called, a lot of time is spent in executing a series of instructions, for tasks such as jumping to the function, saving registers, pushing arguments into stack and returning to the calling function.
2) C++ proposes a solution of inline functions to this problem. Inline function makes a program run faster because the overhead of a function call and return is eliminated.
3) However, it makes program to take up more memory, because the statements that define inline function are reproduced at each point where the function is called.
4) "An inline function is a function that is expanded inline when it is invoked". i.e. the compiler replaces function call with the corresponding function code.
5) The inline functions are defined as follows :

```
inline function header
{
function body
}
```

e.g.
```
inline int area (int a, int b)
{
return (a*b);
}
```

6) The functions are generally made inline, when they are small enough to be defined in one or two lines.

7) The keyword inline is not a command, but it is a request to the compiler.

8) Following are some situations in which compiler may ignore inline request :
   i) For functions returning value, if loop, switch or goto statement exists.
   ii) For functions not returning value, if a return statement exists.
   iii) If functions contain static variables.
   iv) If inline functions are recursive.

---

Q. 14 What are default arguments? Give the advantages of using default arguments.

Ans. :

1) C++ allows to call a function without specifying all its arguments. In such cases, the function assigns a default value to the parameter, which does not have a matching argument in the function call.

2) Default values are specified when the function is declared.

3) Consider a function area declared as follows,

   float area (int r, float Pi = 3.14);

   The above prototype declares default value 3.14 to the argument Pi. A subsequent function call like -

   A = area(7); //one argument missing

   passes the value 7 to r and lets the function use default value 3.14 for Pi.

   The call A = area (7, 2.5) passes an explicit value 2.5 to Pi.

4) Only trailing arguments can have default values. i.e. add defaults from right to left. A default value cannot provide to an argument in the middle of list.

5) Advantages of using default arguments :
   i) These are useful in situations, where some arguments have same values.
   ii) It provides better flexibility to programmers by allowing to use particular arguments that are meaningful to particular solution.
   iii) Use default arguments to add new parameters to the existing functions.
   iv) Default arguments can be used to combine similar functions into a single function.

---

Q. 15 Explain the concept of function overloading with example.

**(March 2008, 15, 17; Oct. 2006**

Ans. :

1) The use of same function name to create functions that perform a variety of different tasks is called as function overloading.

2) Overloading refers to the use of same thing for different purposes. Function overloading or function polymorphism, is an example of compile time polymorphism.

3) Using the concept of function overloading, create a family of functions with one function name but with different argument lists.

4) The function would perform different operation, depending on argument list in function call.

5) The correct function to be invoked is determined by checking the number and the type of the arguments and not on the function type.

6) e.g.

```
# include <iostream.h>
int area (int s);                  //prototype declaration
int area (int l, int b);           //for overloading area( )
main ( )
{
cout <<area (10);                  //function calls
cout <<area (5, 10);
}
int area (int s)                   //function definition
{
return (s*s);
}
int area (int l, int b)
{
return (l*b);
}
```

In above example the function area( ) is overloaded. The first function is used to calculate area of square. It has one integer parameter.

The second function is used to calculate area of rectangle. It has two integer parameters.

7) When a function is called, the compiler first matches the prototype having same number and types of arguments and then calls appropriate function for execution. A best match must be unique.

Q. 16 Explain the structure of a general C++ program. (March 2019)

Ans. :

1) A typical C++ program contains 4 sections as shown in following figure These sections may be placed in different code files and then compiled independently or jointly.

| Include files |
|---|
| Class declaration |
| Class functions definitions |
| Main function program |

Structure of C++ program

2) It is a common practice to organize a program into three separate files.

3) The class declarations are placed in a header file and the definitions of the member go in other file.

4) This approach enables the programmer to separate the abstract of the interface from the implementation details.

5) Finally the main program that uses the class is placed in third file, which includes the previous two files as well as any other files required.

Q. 17 Write a program in C++ that finds larger number among three numbers.

Ans. :

```
//Program to find largest number
#include <iostream.h>
void main( )
{
int a, b, c, max;
cout<<"Enter three numbers" <<endl;
cin>>a>>b>>c;
if (b>c)
        { max = b;}
else
        {max = c;}
if (a>max)
        {max = a;}
cout<<"The larger number is:-";
cout<<max;
}
```

Q. 18 Write a program in C++ to display a fibonacci series of 15 terms.

(March 2004, 07, 09, 17, 22; Oct. 2002

OR Write a program in C++ to display a Fibonacci series of 20 terms (use n < = 18 in this case)

Ans. :

```
#include<iostream.h>
void main( )
{
int f0, f1, f, n;
f0 = 0;
f1 = 1;
clrscr( );
cout<<"Fibbonacci series\n";
cout<<"\n" <<f0<<"\n"<<f1;
for (n=1 ; n<=13; n++)
{
        f = f0 + f1;
        cout<<"\n"<<f;
        f0 = f1;
        f1 = f;
}
}
```

Q. 19 Write a program in C++ to calculate and print factorial of first 10 numbers.

Ans. :

```
//C++ program to calculate and print factorial of first 10 numbers
#include<iostream.h>
#include<conio.h>
void main( )
{
    int fact, n, i;
    clrscr( );
    cout<<"Number"<<"\t"<<"Factorial";
    for (n=1; n<=10; n++)
    {
        fact = 1;
        for (i = 1; i<=n; i++)
        {
        fact = fact*i;
        }.
    cout<<endl<<n<<"\t"<<fact;
    }
}
```

Q. 20 Write a C++ program to find factorial of a natural number input during program execution. (March 2004, 08, 17, Oct. 2002,04,12

Ans. :

```
//Program to find factorial of a number
#include<iostream.h>
#include<conio.h>
void main ( )
{
    int fact, numberi;
    clrscr ( );
    fact = 1;
    cout << "Enter the number" <<endl;
    cin >> number;
    for (i = 1; i < = number; i++)
    {
    fact = fact * i;
    }
    cout << "The factorial of a inputted number is :" << fact;
}
```

Q. 21 Write a program in C++ to check whether the given integer is palindrome or not. (Oct.2012

Ans. :

```
//C++ program to find whether the given integer is palindrome or not
#include<iostream.h>
#include<conio.h>
```

```
void main( )
{
int n, dn, temp=0, d;
clrscr( );
cout<<"Enter a number";
cin>>n;
dn=n;
while (dn!=0)
{
        d=dn%10;
        temp=(temp*10)+d;
        dn=dn/10;
}
if (n==temp)
{
cout<<"The number"<<n<<"is palindrome";
}
else
{
cout<<"The number"<<n<<"is not palindrome";
}
}
```

Q. 22 What is an armstrong number ? Write a program in C++ to check whether the given number is armstrong or not.

Ans. : Armstrong number :

"If sum of the cubes of digits of a number is equal to the original number, then the number is said to be an armstrong number".

e.g. 153 is an armstrong number.

```
//C++ Program to find whether the number is armstrong or not.
#include<iostream.h>
#include<conio.h>
void main( )
{
int n, dn, temp, d;
cout<<"Enter a number";
cin>>n;
dn = n;
temp=0;
while (dn!=0)
{
        d=dn%10;
```

```
        temp=temp+(d*d*d);
        dn=dn/10;
    }
    if(n==temp)
    {
    cout<<n<<"is armstrong no.";
    }
    else
        {
        cout<<n<<"is not an armstrong number";
        }
    }
```

Q. 23 Write a program in C++ to print the numbers in following manner.

```
1
2  2
3  3  3
4  4  4  4
:
:
n terms
```

Ans. :

```
//C++ program to print given pattern
#include<iostream.h>
#include<conio.h>
void main( )
{
int i, j, n;
clrscr( );
cout<<"Enter a number";
cin>>n;
cout<<endl;
for (i=1; i<=n; i++)
{
    for (j=1; j<=i; j++)
    {
        cout<<i<<"\t";
    }
    cout<<endl;
}
}
```

Q. 24 Write a program in C++ to print the numbers in following manner.

```
1  0  1  0  1
1  0  1  0
1  0  1
1  0
1
```

Ans. :

```
//C++ program to print given pattern
#include<iostream.h>
#include<conio.h>
void main( )
{
clrscr( );
int i, j;
      for (i=5; i>=1; i- -)
      {
            for (j=1; j<=i; j++)
            {
                  cout<<j%2<<"\t";
            }
            cout<<endl;
            }
      }
```

Q. 25 Write a program to perform arithmetic calculations such as addition, subtraction, multiplication or division, depending on choice using switch statement.

Ans. :

```
//C++ program to generate simple calculator
#include<iostream.h>
#include<conio.h>
void main( )
{
float a, b, result;
int ch;
clrscr();
cout<<"Enter two numbers";
cin>>a>>b;
cout<<"\n1-addition\n 2-subtraction\n 3-multiplication \n 4-division";
cout<<"Enter Your Choice :";
cin>>ch;
switch (ch)
{
```

```
        case 1:
              result=a+b;
              cout<<"Sum is"<<result;
              break;
        case 2:
              result=a-b;
              cout<<"Difference is"<<result;
              break;
        case 3:
              result=a*b;
              cout<<"Product is"<<result;
              break;
        case 4:
              result=a/b;
              cout<<"Division is"<<result;
              break;
        default:
              cout<<"invalid choice";
              break;
        }
}
```

Q. 26 What is a recursive function ? Write a program in C++ to calculate addition of first n numbers using recursive function.

Ans. : Recursive function :

"A function which is called within the body of the same function itself is called as recursive function."

```
//Program to calculate addition of first n numbers
#include<iostream.h>
#include<conio.h>
int add (int);
void main( )
{
int n, sum;
cout<<"Enter a number \n";
cin>>n;
sum=add(n);
cout<<"Addition of first"<<n<<"numbers is"<<sum;
}
//function to calculate addition
int add (int x)
{
int S = 0;
if (x!=0)
```

```
{
S=x+add(x-1);
}
return(S);
}
```

Q. 27 Write a program in C++ to calculate volume of cube, cylinder and rectangular box, depending on choice by using function overloading.

Ans. : //Program using function overloading

```
#include<conio.h>
#include<iostream.h>
int volume (int s);
float volume (float r, float h);
int volume (int l, int b, int h);
void main( )
{
int ch;
do
{
        clrscr( );
        cout<<"\n1:Volume of cube";
        cout<<"n2:Volume of cylinder";
        cout<<"\n3:Volume of rectangular box";
        cout<<"\n4:Quit";
        cout<<"\n\n Enter Your Choice";
        cin>>ch;
switch (ch)
{
case 1:
        cout<<"Volume of cube is";
        cout<<volume (5);
        break;
case 2:
        cout<<"Volume of cylinder is";
        cout<<volume (7.0, 2.0);
        break;
case 3:
        cout<<"Volume of rectangular box is";
        cout<<volume (3, 5, 7);
        break;
```

```
case 4: break;
default :
        cout<<"Invalid choice";
        cout<<"Reenter your choice";
}
}
        while (ch!=4);
}
        //function to calculate volume of cube
        int volume (int s)
{
        return (s*s*s);
}
        //function to calculate volume of cylinder
        float volume (float r, float h)
{
        return (3.14*r*r*h);
}
        //function to calculate volume of rectangular
        int volume (int l, int b, int h)
{
        return (l*b*h);
}
```

## Arrays, Pointers, References and Strings

Q. 28 What is an array ? Explain how array can be passed onto a function.

Ans. :

1) "An array is a collection of identical data objects, which are stored in consecutive memory locations under common variable name."
2) Arrays may be one dimensional or multidimensional.
3) The general form for declaration of one-dimensional array is

| data-type array-name [expression]; |
|---|

e.g. int a[10];

This declaration creates an array of 10 integers.

4) In general, C++ arrays are zero based. i.e. in above examples, the first array element has index 0 and it is referred as a[0]. Similarly, second array element is a[1] and the last i.e. $10^{th}$ element is a[9].

5) C++ allows to pass the entire array onto a function.

An array name can be used as an argument for the function declaration. No subscript brackets are required to invoke a function using arrays.

e.g. float rev (float b[], int c);

This declares a function rev, with two parameters, out of which one is an array.

Q. 29 What are pointers ? Give the advantages of using pointers **(March 11, 19, July 16, 17**

Ans. :

1) "A pointer is a variable, which holds the memory address of other variable."

2) * operator is used to declare pointer in C++. It takes the form as :

datatype * variable name;

e.g. int *ptr;

The above declaration will create a variable ptr, which is a pointer variable and which will point to a variable, whose data type is integer.

3) The data type of ptr is not integer, but data type of variable which ptr will point is integer.

4) Advantages of using pointers are as :

i) It allows to pass variables, arrays, functions, strings, structures, objects as function arguments.

ii) It allows to return structured variables from functions.

iii) It supports dynamic allocation and deallocation of memory segments.

iv) By using pointers, variables can be swapped, without physically moving them.

v) It allows to establish link between data elements or objects.

Q. 30 What are pointers in C++ ? Explain the use of pointer variables for function definitions using call by value and call by reference OR **(March 2004,07,08,09,22, Oct. 2006,11**

Explain 'Call by value' and 'Call by reference' with one example of each.

Ans. :

1) Pointers in C++ :

A pointer is a variable which holds the memory address of another variable.

* operator is used to declare pointer in C++.

For example : int *ptr;

where ptr is a pointer variable and which will point to a variable whose data type is integer.

2) The use of pointers in a function definition may be classified into two groups :

(1) Call by value (2) Call by reference.

3) Call by value :

(a) When a portion of the program invokes a function, control will be transferred from the main function to the calling function and the value of actual arguments is copied to the function.

(b) Within function the actual value may be altered or changed.

(c) When the control is transferred back from function to the program, altered values are not transferred back. This type of passing formal argument to a function is called as call by value

(d) For example :

```
main ( )
{ void funct (int X, int Y);
        .....
funct (X, Y); // Call by value
.....
}
void funct (int a, int b)
{.....
}
```

4) Call by reference : (Oct.2014)

(a) In call by reference, when a function is called by a program the address of the actual arguments are copied on to the formal arguments. i.e. the formal and actual arguments are referring to same memory location.

(b) Therefore change in value of formal argument affects the value of actual arguments.

(c) The content of a variable that are altered within the function are return to calling portion of a program in the altered form.

(d) For example :

```
main ( )
{
void funct (int X, int Y);
.....
.....
funct (&X, &Y); // Call by reference.
.....
}
void funct (int X, int Y)
{
.....
}
```

Q. 31 Explain how the memory address of a variable can be accessed in C++. (March 2004, 07,14 ; Oct. 2004,12)

Ans. :

1) Computer uses memory for storing the values of variables and the memory is a sequential collection of storage cell. Each cell has a number called address of the cell.

2) In C++, if declare a variable, then it gets associated with certain location where the value of the variable is stored.

3) Consider the declaration

int p = 30;

then p → Location name (variable)

30 → Value at location

7940 → Location number (address)

Computer has selected 7940 memory location to store the value 30.

4) To access the memory address of a particular variable '&' operator is used. The '&' operator returns the memory address of its operand.

For example : a = &p;

assigns the memory address of variable p to the a. This address is the location address of variable. The operator '&' is "the address of" operator.

The variable 'a' is declared as pointer variable as that contains the address. The pointer variable declared in C++ as,

int * a;

where * indicates that a is a pointer variable.

---

Q. 32 What is call by reference ? What is the advantage of call by reference over call by value ? (Mar. 2014)

Ans. :

A function can be called by two methods :

(i) Call by value

(ii) Call by reference

1) When a function call passes arguments by value (call by value) the called function creates a new set of variables and copies the values of arguments into them.

2) The function does not have access to the actual variables in the calling program and can only work on the copies of values.

3) Provision of reference variables in C++ permits to pass parameters to the function by reference.

4) When pass arguments by reference (call by reference) the formal arguments in the called function become alises to the actual arguments in the calling function. This means that when the function is working with its own arguments, it is actually working on the original data.

5) The mechanism of call by value is good, if the function does not need to alter the values of the original variables in the calling program.

6) But, if a situation to change the values of variables in the calling program. e.g. in bubble sort compare two adjacent elements in the list and interchange them if first is greater than second. In such situation, the function should be able to interchange the values of variables of calling program, which is not possible by call by value. But it can be done if the call by reference method is used.

7) e.g. //Program to interchange the values of variable

```
#include <iostream.h>
void swap (int*, int*); //function declaration
```

```
void main( )
{
        int a, b;
        cin>>a>>b;
        swap (&a,&b); //call by reference
        cout<<"a="<<a;
        cout<<"b="<<b;
}
void swap (int*a, int*b) //function definition
{
        int t;
        t=*a; //assign the value at address a to t.
        *a=*b; //put the value at b into a.
        *b=t; //put the value at t into b.
}
```

Q. 33 Explain Library Functions :
(i) Strcpy ( ) (ii) Strcmp ( ) (Oct. 201

Ans. :

(i) Strcpy ( )

If $S_1$ and $S_2$ are string then strepy ($S_1$, $S_2$) copies character string $S_2$ into character string $S_1$. It means it creates a duplicate of string $S_2$.

Char * strcpy (Char * $S_1$, const char * $S_2$)

For example

```
int main ( )
{
char S1 [] = "ABCD"
char S2 [] = "XYZ"
cout << "Before strcpy (S1, S2) \ n";
cout << "\t S1 = [" << S1 <<"], length = " << streln (S1) << endl;
cout << "\t S2 = [" << S2 <<"], length = " << strln (S2) << endl;
strcpy (S1, S2);
cout <<"After strcpy (S1, S2) \n";
cout << "\t S2 = [" << S2 <<"], length = " << strlen (S2) << endl;
}
```

O/P-Before strcpy ($S_1$, $S_2$)

$S_1$ = [ABCDE], length = 5

$S_2$ = [XYZ]; length = 3

After strcpy ($S_1$, $S_2$)

$S_1$ = [XYZ], length = 3

$S_2$ = [XYZ], length = 3

(ii) Strcmp ( ) :

Int strcmp (char * $S_1$, char * $S_2$);

It compares $S_1$ with $S_2$ Returns a negative integer, zero or positive integer according to $S_1$ is less than equal to or grater than $S_2$.

Example :

Char *$S_1$ = "ABCDE"

Char * $S_2$ = " "

If (strcmp ($S_1$, $S_2$) <0)

Cout << $S_1$ << "<" <<$S_2$ << endl;

else

cout <$S_1$<< "> = " << $S_2$ << endl;

O/P is → ABCDE > =

Q. 34 Write a C++ program to read 'n' numbers input from keyboard and sort them in ascending order. (Oct. 2002)

Ans. :

```
//Program to sort numbers in ascending order
#include <iostream.h>
voide main ( )
{
    int a [100];
    int n, i, j, temp;
    cout <<"How many numbers ?" <<endl;
    cin >> n;
    cout << "Enter the elements :";
    for (i=1; i<=n; i++)
    {
        cin >> a[i];
    }
    for (i=1; i<=(n–1); i++)
    {
        for (j=1; j<=n–i; j++)
        {
            if (a[j] > a[j+1])
            {
                temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
    }
    cout<<"Ascending order is :\n";
```

```
for (i = 1; i< = n; i++)
{
cout <<a[i] <<endl;
}
}
```

Q. 35 Write a C++ program to sort a set of 10 floats in descending order using bubble sort method. (March 2003)

Ans. : Hint : In this program, write instruction as,

```
if (a[j] < a[j+1])
```

instead of

```
if (a[j] > a[j+1])
```

in above program

Q. 36 Write a program in C++, that first initialize an array of ten integers. Then sort that array descending order using sort() by call by reference method.

Ans. :

```
//Program to sort array (use call by reference)
#include<iostream.h>
#include<conio.h>
void sort (int*, int*);
void main( )
{
int a[10], i, j;
clrscr( );
cout<<"Enter ten numbers \n";
for (i=0; i<=9; i++)
        {
                cin>>a[i];
        }
cout<<"\n After Sort \n";
for (i=1; i<=9; i++)
{
        for (j = 0 ; j< = (10 -i) ; j++)
        {
        sort (&a[j], &a[j+1]); //call by passing address of variable
        }
}
for (i=0; i<=9; i++) //print sorted array
{
        cout<<a[i]<<"\n";
}
}
//function to sort array
```

```
void sort (int *x, int *y) //function definition
{
        int temp;
        if(*x < *y)
        {
        temp=*x; //assign the value at address x to temp
        *x=*y; //put the value at y into x
        *y=temp; //put the value at temp into y.
        }
}
```

Q. 37 Write the following power ( ) function in C++ that returns X raised to the power n, where n can be any integer. (March 2002, 2006)

Double power (double X, int p);

Use the algorithm that would compute X20 by multiplying 1 by X 20 times.

Ans. :

```
//Power ( ) function in C++
#include<iostream.h>
double power (double, int); //function prototype
void main ( )
{
        double X;
        int n;
        cout << "Enter the value of X\n";
        cin >> X;
        cout << "Enter the value of n\n";
        cin >> n;
        cout << "The result is :" <<power (X, n);
}
double power (double x, int n)
{
        if (x = = 0)
            return 0;
        if (n = = 0)
            return 1;
        double y = 1;
        for (int i = 0; i < n; i++)
            y = y * x;
        for (int j = 0; j > n; j--)
            y = y/x;
        return y;
}
```

Q. 38 Write a .C++ program to accept a set of 10 numbers and print the numbers using pointers. (March 2002, Oct. 2005,12,13, July 2017)

Ans. :

```
//Program to print numbers using pointers.
#include<iostream.h>
#include<conio.h>
void main( )
{
int a[10], i, *ptr;
clrscr( );
cout<<"Enter 10 numbers"<<endl;
for (i=0; i<=9;.i++)
{
cin>>a[i];
}
ptr=&a[0]; //or use ptr = &a;
cout<<"\n The numbers are\n";
for (i=0; i<=9; i++)
{
cout<<*ptr<<"\n";
ptr++;
}
}
```

Q. 39 Write a program in C++ to read a line of text and to count number of words in a text

Ans. : (March 2002, 08, 09; Oct.2013, March 2020)

```
//Count words in a line of text
#include<iostream.h>
#include<string.h>
void main( )
{
        char line [80];
        int count = 1, len, i;
        cout<<"\n Enter a line of text \n";
        cin.getline (line, 80);
        len = strlen (line);
        for (i = 0; i < = len; i++)
 {
        if (line [i] = = ' ')
        count++;
 }
        cout<<"No. of words are";
        cout<<count;
 }
```

## Principles of object oriented programming

Q. 40 What is object oriented programming ? Enlist the features of object oriented programming. (March 2009, Oct.2010, March 2018

Ans. : Definition of OOP :

1) "Object oriented programming is an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand."
2) In object oriented programming, the program is designed around the data being operated upon rather than upon the operations themselves.
3) It ties data more closely to functions that operate on it. OOP allows to decompose a problem into a number of entities called objects and then builds data and functions around these entities.
4) When a program is executed, the objects interact by sending messages to one another.
5) Each object contains data and code to manipulate the data.

Features of OOP : (Oct. 2003,11,13 ; March 2005,07,22

1) Emphasis is on data rather than procedure.
2) Programs are divided into number of objects.
3) Data structures are designed such that they characterize the objects.
4) Functions that operate on the data of an object are tied together in the data structure.
5) Data is hidden and cannot be accessed by external functions.
6) Objects may communicate with each other through functions.
7) New data and functions can be easily added wherever required.

---

Q. 41 Explain the following concepts related to object oriented programming :
(i) Objects (ii) Classes
(iii) Inheritance (iv) Polymorphism
(v) Data Encapsulation (vi) Data Abstraction (vii) Data hiding

Ans. :

(i) Objects : (March 2006, 2008 ; Oct. 2004, 2006,2007, Dec. 2020

1) Objects are the basic runtime entities in object oriented system. for eg. they may represent a person, place, a bank account or any item that the program must handle.
2) Programming problem is analyzed in terms of objects and the nature of communication between them.
3) Program objects should be chosen such that they match closely with the real-world objects.

(ii) Classes : (Mar. 2006,08,14 ; Oct. 2006,07,14, Dec. 2020

1) Class is a way to bind data and its associated functions together.
2) The entire set of data and code of an object can be made a user defined data type with the help of a class.
3) In fact an object is nothing but a variable, whose data type is class.
4) Once a class has been defined, user can define any number of objects belonging to that class.
5) A class is a collection of objects of similar type.

(iii) Inheritance : (Oct. 2007)

1) The mechanism of deriving a new class from an existing one is called as inheritance.
2) Inheritance is the process by which objects of one class can acquire the properties of objects of another class.
3) In OOP, inheritance stands for reusability. This means that additional features can be added to an existing class without modifying it.

(iv) Polymorphism : (March 2008, Oct. 2004)

1) Polymorphism is an important OOP concept. Polymorphism means ability to take more than one form.
2) Polymorphism plays an important role in allowing objects having different internal structures to share the same external interface.
3) This means that a general class of operations may be accessed in the same manner even though specific actions associated with each operation may differ.
4) Polymorphism is extensively used in implementing inheritance.

(v) Data encapsulation (March 2015, Dec. 2020, 3

Encapsulation is the packing of data and functions into a single component. Data encapsulation, also known as data hiding, is the mechanism whereby the implementation details of a class are kept hidden from the user. The user can only perform a restricted set of operations on the hidden members of the class by executing special functions commonly called methods. Encapsulation can be used to hide data member and member function.

(vi) Data Abstraction (March 2015; Oct. 2004, Dec. 2020

(1) Abstraction refers to the act of representing essential features without including the background details of explanations.

(2) Classes are the concept of abstraction and are defined as a list of abstract attributes and functions to operate on these attributes.

(3) They encapsulate all the essential properties of the object that are to be created.

(vii) Data hiding : (July 2017)

(i) Data hiding means keeping details private while giving access to an object only through messages.

(ii) It is hiding unnecessary complexity from outside world which prevents accidental modification.

## Classes and Objects

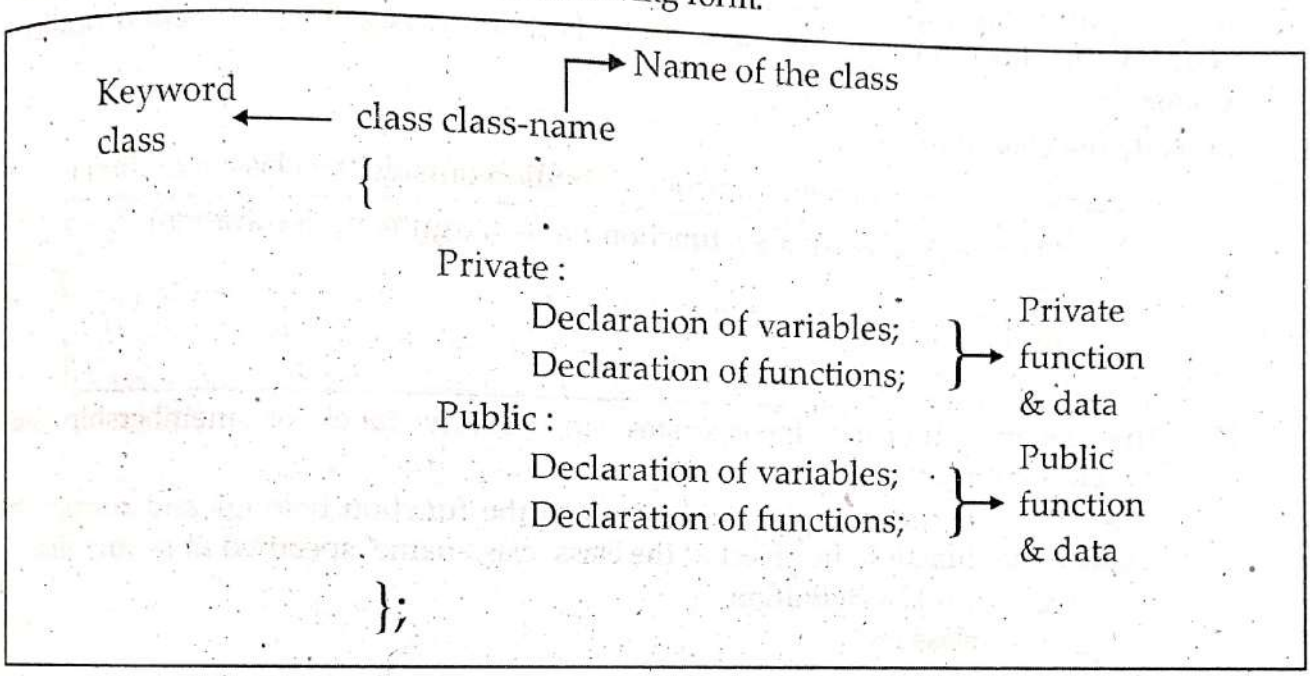
Q. 42 What is a class ? Explain general form of class declaration. - (Mar. 2003,08,11, 13; Oct.2005,11,14; July 2019

Ans. :

1) Class is a way to bind data and its associated functions together.
2) It allows the data (and functions) to be hidden, if necessary, from external use.
3) When defining a class, a new abstract data type that can be created that treated like any other built-in data type.

4) Generally, a class specification has two parts :
1. Class declaration. 2. Class functions definitions.
The class declaration describes the type and scope of its members. The class function definitions describes how class functions are implemented.

5) Class declaration :
Generally a class declaration has following form.




a) The keyword class specifies that what follows is an abstract data type class-name.
b) The body of a class is enclosed within braces and terminated by a semicolon.
c) The body of the class contains declaration of class members, which are variables and functions. They are generally grouped under two sections namely private and public, which are known as visibility labels. These keywords are followed by a colon.
d) The members, declared as private can be accessed only from within the class. It hides data from external use. It is a key feature of OOP.
e) The public members can be accessed from outside the class also.
f) If both the visibility labels are missing, then by default, the members of the class are private. Such a class is completely hidden from outside world and does not serve any purpose.

Example : A class declaration would look like :

```
class item                    //specify a class
{
        int number;           //class data
        float cost;
public:
        void getdata (int a, float b);
        void putdata (void);
};
```

Q. 43 Describe how member functions of class can be defined outside the class definit and inside the class definition.

(March 04, 08,12,14 ; Oct. 04,12,15, 18, 20; July 18, Oct. 20

Ans. : Member functions of class can be defined at two places :

(i) Outside the class definition. (ii) Inside the class definition.

Irrespective of the place of definition, the function performs same operation. Hence code for the function body is identical in both the cases. Only function header i changed.

(i) Outside the class definition :

1) The general form of member function definition outside the class definition is :

```
return-type class-name :: function-name (Argument declaration)
{
function body
}
```

2) The member function incorporates an identity label or membership label (i.e. class-name : :).

3) This label tells compiler the class to which the function belongs and restricts the scope of that function the object of the class 'class-name' specified in header line.

e.g.

```
//class definition
class try1
{
    public:
    void display (void);
};
    //member function definition outside class
    void try1 :: display (void)
{
    cout<<"Programming is fun";
}
```

(ii) Inside the class definition :

1) Another method for defining a member function is to replace the function declaration by the actual function definition.

e.g.

```
class try1
{
    public:
    void display (void)
    {
    cout<<"Programming is fun";
    }
};
```

2) When a function is defined inside a class, it is treated as an inline function Normally, only small functions are defined inside the class definition.

Q. 44 What is an object ? Describe how members of a class be accessed using object of that class.

Ans. :

1) An object is a variable whose datatype is a class. (Oct.2011)

2) User have more than one object for a class. The objects can be declared as :

```
class-name object1-name, object2-name, ...
```

3) The declaration of an object is similar to that of a variable of any basic type. The necessary memory space is allocated to an object at this stage.

Accessing members of class using objects :

1) The private data of a class can be accessed only through the member functions of that class.

2) To use a member function, the dot operator (period) connects the object name and the member function. The dot operator also called as class member access operator.

3) The following format is used for calling a member function :

objects-name.function-name (actual argument);

4) e.g. Consider a class item defined as follows :

```
class item
{
        private:
        int number;
        public:
        int cost;
        void getdata (int a, int b) //inline function
        {
                number=a;
                cost=b;
        }
        void show (void)
        {
        cout<<a<<b;
        }
};
        void main( )
{
        item X;
        X.getdata (20, 30) ;
        .....
}
```

Here item is a class with private variable number and public members cost, getdata (int, int) and show( ). The X is an object of item.

e.g. X.getdata (20, 30);

This declaration will apply values 20 and 30 to number and cost respectively.

5) Similarly public variables of class can be accessed within main( ) function. But private variables cannot access inside main program, they can be accessed only by public functions of the same class.

Instead of using function as X.getdata (20, 30), if directly apply 20 and 30 to number and cost in main program as :-

```
X.number = 20; //error
X.cost = 20;
```

Then, there will be an error in first statement, because number is private member of class, while there will be no error in second statement, because cost is public member of class.

---

Q. 45 Write a C++ program to find the Greatest Common Divisor of two numbers. Define a method find to accept the values and calculate GCD of two numbers and print the GCD value. **(March 2002, 2006 ; Oct. 2002, 2005**

Ans. :

```
//Program to find GCD value
#include<iostream.h>
class gcd
{
        int a, b;
        public :
                void find ( );
};
void gcd :: find (void)
{
        cout<<"Enter the value of a and b\n";
        cin>>a>>b;
        while (a ! = b)
        {
        if (a > b)
                a = a – b;
        if (b > a)
                b = b – a;
        }
                cout << "The gcd is :" << a;
}
void main ( )
{
        gcd obj1;
        obj1.find ( );
}
```

Q. 46 What are friendly functions ? Give the characteristics of a friend function. (Mar:2016, 17, 20; Oct. 2010, 21, July 2016, 17, 18, 19

Ans. :

1) C++ allows the common function to be made friendly with more than one classes, thereby allowing the function to have access to the private data of classes. Such a function need not be a member of any classes.

2) Non-member function cannot have access to the private data of a class. However, there could be a situation where user would like two classes to share a particular function. At this situation friend function is used.

3) To make an outside function "friendly" to a class, simply declare the function as the friend of the class as shown below :-

```
class class-name
{
private:
.....
.....
public:
.....
friend return-type function-name (arguments); //declaration
};
```

4) The keyword "friend' declares the function to be friendly with that class. This function is defined as a normal C++ function. The function definition does not use class-name, keyword friend or scope resolution operator.

5) A friend function has following characteristics :

(i) It is not in the scope of the class to which it has been declared as friend.

(ii) Since it is not in scope of the class, it cannot be called by using object of that class. It is called like a normal C++ function.

(iii) It can be declared either in public or the private part of a class without affecting its meaning.

(iv) Usually, it has the objects as arguments.

(v) It cannot access the member function directly and has to use an object name and dot operator with each member name.

Q. 47 Write any three characteristics of friend function ? (Oct. 2003, 2008)

Ans. : A friend function has following characteristics :

(1) It is not in the scope of the class to which it has been declared as friend.

(2) Since it is not in scope of the class, it cannot be classed by using object of that class. It is called like a normal C++ function.

(3) It can be declared either in public or the private part of a class without affecting its meaning.

Q. 48 Write a program in C++ to demonstrate how a common friend function can be used to exchange the private values of two classes. (Use call by reference method.)

Ans. :

```
//Swapping private data of classes using friend function
#include<iostream.h>
#include<conio.h>
class A; //forward declaration
class B
{
    private:
    int val2;
    public:
        void getdata (void)
        {
        val2=21;
    }
    void display (void)
        {
        cout<<"Value 2:"<<val2;
        }
    friend void exchange (B&, A&);
    };
class A
{
    private:
    int val1;
    public:
    void get (void)
    {
    val1=12;
    }
    void disp (void)
        {
        cout<<"Value 1:"<<val1;
        }
        friend void exchange (B&, A&);
    };
    void exchange (B &x, A &y)
        {
                int temp;
                temp = x.val2;
                x.val2 = y.val1
```

```
        y.val1 = temp;
    }
void main( )
{
    A abc;
    B pqr;
    abc.get( );
    pqr.getdata( );
    cout<<"Before exchange:-";
    abc.disp( );
    pqr.display( );
    exchange (pqr, abc); //Swapping
    cout<<"\n After exchange:";
    abc.disp( );
    pqr.display( );
}
```

Note :When a common function is to be made friendly with two classes, forward declaration is necessary, because when we declare friend function in first class, the object of second function may also be passed as argument to the friend function. But, at that time compiler does not have knowledge about the second class. Therefore an error will arise. Hence, forward declaration is necessary.

---

Q. 49 How members of class are accessed by using pointer object ?

Ans. :

1) Generally in C++, pointer object is used along with normal object and then the address of normal object is given to pointer object.

2) The pointer object is declared as follow in main() program as :

```
class name object-name, pointer object;
pointer object = address of object name;
```

e.g. emp e, *ptr;
ptr=&e;

Here e and ptr are objects of emp, ptr is pointer object, which holds address of e.

3) Now, for access member functions and variables of e, use ptr and '→' (arrow) operator. Suppose getdata( ) is function of class emp. Then by using pointer object, it can be accessed as

ptr → getdata( );

Another way of accessing is (*ptr).getdata( );

4) For eg. following program declare a class employee, which contains name, phone number and salary of employee as data. Member functions are used to read data and display it.

```
#include<iostream.h>
#include<conio.h>
```

```
class emp
{
        private:
        char name[20];
        int ph, sal;
        public:
        void getdata (void);
        void display (void);
};
        void emp::getdata (void)
{
        cout<<"\n Enter employee name-";
        cin>>name;
        cout<<"\n Enter employee salary";
        cin>>sal;
        cout<<"\n Enter phone number-";
        cin>>ph;
}
        void emp::display (void)
{
        cout<<"\n Name:"<<name;
        cout<<"\n Salary:"<<sal;
        cout<<"\n Phone number"<<ph;
}
        void main( )
{
        emp e, *ptr;
        ptr=&e;
        ptr → getdata( );
        ptr → display( );
}
```

Note : This method of using pointer objects in class is same as that of using pointer variables in case of structure.

We can also access array of objects by using single pointer object.

The procedure is

```
emp e[10], *ptr;
ptr=&e[0];
```

This procedure is also same as that in case of structure.

Q. 50 Explain in short the three special characteristics of a static data member in a class.

(March 2005, 11,16,17 ; Oct. 2007,13; July 18

Ans. :

The three special characteristics of a static data member in a class are as follows :

(1) It is initialized to zero when the first object of its class is created. No other initialization is permitted.

(2) Only one copy of that member is created for the entire class and is shared by all the objects of that class, no matter how many objects are created.

(3) It is visible only within the class, but its life time is the entire program.

## Constructor and Destructor

Q. 51 What is a constructor ? Why it is called so ?

(March 2005, 06, 07,11,12,13,14,15; Oct. 2004,05,07,10,12,13,14,1
July 2017, 19, March 18, 19, 20

Ans. :

1) "A constructor is a special member function of a class. Its task is to initialize the objects of its class."

2) It is special because its name is same as that of the class to which it belongs.

3) The constructor is invoked whenever an object of its associated class is created.

4) It is called constructor because it constructs the values of data members of the class.

5) A constructor can never return any value. Hence, it is written with no return type (even void is not written).

6) e.g. A constructor is declared and defined as follows

```
//class with constructor
class integer
{
    private:
    int m, n;
    public:
    integer (void); //constructor declared
};
integer::integer (void) //constructor defined
{
    m=0;
    n=0;
}
```

7) Whenever a class contains a constructor like one above, it will be initialized automatically, whenever an object of that class is created.

i.e. the declaration - integer int1;

This not only creates the object int1 of type integer, but also initializes its data members m and n to zero.

Q. 52 Give the characteristics of a constructor function. OR What are the syntax rules for writing constructors ? (Mar. 2003,13, 19; Oct. 2002, 04,12,13, July 201

Ans. :

(i) The constructor name is always same as the class name.
(ii) They do not have return types, not even void and therefore, they cannot return values.
(iii) They cannot be static or virtual.
(iv) They should be declared in public section.
(v) They cannot be inherited though a derived class can call base class constructor.
(vi) Like other C++ functions, they can have default arguments.
(vii) We cannot refer to their address.
(viii) An object with a constructor cannot be used as a member of union.
(ix) They make implicit calls to the operators 'new' and 'delete' when memory allocation is required.
(x) When a constructor is declared for a class, initialization of class objects become mandatory, since constructor is invoked automatically when the objects are created.

Q. 53 Explain parameterized constructors with default arguments.

Ans. :

1) Generally a constructor initializes the class object to predetermined values. But, in practice, it may be necessary to initialize data elements of objects to different values.
2) C++ permits to achieve this objective by passing arguments to the constructor function when the objects are created.
3) The constructors that can take arguments are called parameterized constructor.
4) For e.g.

```
class fib
{
        private:
        int f0, f1;
        public:
        fib (int x, int y); //Parameterized constructor
};
fib::fib (int x, int y)
{
        f0=x;
        f1=y;
}
```

5) When a constructor has been parameterized, the object declaration statement such as, Fib F; will not initialize the data elements.
6) Pass the initial values to the constructor function when an object is declared. This can be done in two ways :
   (i) By calling the constructor explicitly.
   (ii) By calling the constructor implicitly.

The explicit call can be made as follows.

```
fib F = fib (0, 1); //Explicitly call
```

or

```
fib F;
F = fib (0, 1);
```

The implicit call can be made as follows

```
fib F (0, 1); //Implicit call
```

7) User also pass default arguments to the constructor.

e.g

```
class fib
{
int f0, f1;
public :
fib (int x, int y = 1) //constructor with default arguments
{
f0 = x;
f1 = y;
}
```

The above declaration sets default value 1 to y.

8) In main ( ), such constructor can be called as

```
fib F (0, 1);
```

or

```
fib F (0); //One argument missing.
```

This call sets f0 to zero and f1 to default value 1.

9) User also call such constructor as -

```
fib F (0, 2); // No argument missing.
```

This call sets f0 to zero and f1 to two, instead of one.

Q. 54 What is a Constructor ? Explain Copy Constructor with example. (Oct. 2009, 3)

Ans :

Constructor :

A constructor is a special member function of a class. Its task is to initialize the objects of its class. It is a special because its name is same as that of the class to which it belongs.

Copy constructor

i) Copy constructor are always used when the compiler has to create a temporary object of a class object.

ii) The copy constructors are used in following situations :
   a) The initialization of an object by another object of the same class.
   b) Return of object as by value parameters of a functions.
   c) Stating the object as by value parameters of a functions.

iii) The general format is :
   class_name : : class_name (class_name & ptr)

iv) For eg : x : : x (x & ptr)
   ptr is a pointer to a class object x.

v) The following program segment shows how to define copy constructor :

```
fib  : : fib ( )        // constructor
{
fo = 0;
f1 = 1;
f = f0 + f1;
}
fib  : : fib (fib & ptr) // copy constructor
{
fo = ptr . f0;
f1 = ptr . f1;
f = ptr . f;
}
```

Q. 55 What is a destructor? Give syntax rules for writing destructor function.

(Mar. 2006,20,09,11,12,14,15, 18, 20, 22; Oct. 2005,07, 21, July 2016, 1

Ans. :

1) A destructor, as the name implies, is used to destroy the objects that have been created by a constructor.

2) The destructor is invoked implicitly by the compiler upon exit from the program to clean up storage that is no longer accessible. In other words, a destructor function gets executed whenever an instance of the class to which it belongs goes out of existence.

3) It is a good practice to declare destructors in a program since it releases memory space for future use.

Syntax rules for writing a destructor function :

(i) A destructor function name is same as that of its class name. But it is preceded by a tilde

e.g. ~fib( ){.... message.....}

(ii) It is declared with no return type since it can never return any value.

(iii) It takes no arguments.

(iv) It should have public access in the class declaration.

Q. 56 What is Constructor and Destructor ? State the difference between them.

(Oct. 2007, March 2011, 3

Ans : (Ch. 3 / Q. 51 and Q. 55 / Pg. 3-39 and Pg. 3-42)

Difference between constructor & destructor

| Constructor | Destructor |
|---|---|
| 1) A constructor is a special member of a class. Its task is to initialize the object of its class. | 1) A Destructor is called to destruction of the object that have been created constructor. |
| 2) The constructor is invoked whenever an object of its associated class is created. | 2) The destructor is invoked important by the complier upon exit from the |

| Constructor | Destructor |
|---|---|
| 3) A constructor, constructs the values of data members of the class. | 3) It is a good practice to declare destructors in a program since in releases memory space for future |

Q. 57 Write a program in C++ to calculate fibonacci series of 'n' numbers using constructor.

Ans. :

```
//Program to generate fibonacci series
#include<iostream.h>
#include<conio.h>
class fibonacci
{
        private:
        long int f0, f1, fib;
        public:
        fibonacci (void);
        void process (void);
        void display (void);
};
fibonacci::fibonacci (void)
{
        f0=0;
        f1=1;
}
void fibonacci::process (void)
{
        fib=f0+f1;
        f0=f1;
        f1=fib;
}
void fibonacci::display (void)
{
        cout<<fib<<"\t";
}
void main( )
{
        int i, n;
        fibonacci F;
        cout<<"\n Enter number of elements"<<endl;
        cin>>n;
        for (i=1; i<=n; i++)
```

```
{
        F.process( );
        F.display( );
}
}
```

Q. 58 Implement a class electricity to calculate electricity bill. The class contains following member functions.

(i) Getdata ( ) : to get meter number, previous units and current units.

(ii) Process ( ) : to check whether current units are greater than previous unit or not. If not display appropriate message and restart program.

(iii) Calculate ( ) : to calculate electricity bill.

(iv) Display ( ) : to display bill and other details.

Use constructor and destructor.

You may use following rates :

| Units | Rates |
|---|---|
| 0-50 | Rs 2 per unit |
| 50-200 | Rs 3.5 per unit |
| 200-500 | Rs 4.5 per unit |
| 500 and above | Rs 5 per unit |

Ans. :

```
//Program to calculate electricity bill
#include<iostream.h>
#include<conio.h>
class electricity
{
    private:
    int mn, pu, cu, n;
    float bill;
    public:
    electricity( ) // constructor
    {
        bill=0;
    }
    void Getdata (void);
    int Process (void);
    void Calculate (void);
    void Display (void);
    ~electricity( ) { } // destructor
};
void electricity::Getdata (void) ( )
{
```

```
cout<<"Enter meter No:";
cin>>mn;
cout<<"\n Enter previous units:";
cin>>pu;
cout<<"\n Enter current units:";
cin>>cu;
}
int electricity::Process (void)
{
    if(cu>pu)
    {
            n=cu-pu;
            return(1);
    }
    else
            return(0);
}
void electricity::Calculate (void)
{
    int dn;
    dn=n;
    if (dn<=50)
    {
    bill=bill+(dn*2);
    }
    else
    {
            bill=bill+(50*2);
            if (dn<=200)
            {
            dn=dn-50;
            bill=bill+(dn*3.5);
    }
    else
    {
            bill=bill+(150*3.5);
            if(dn<=500)
            {
            dn=dn-200;
            bill=bill+(dn*4.5);
            }
            else
            {
            bill=bill+(300*4.5);
```

```
                    dn=dn-500;
                    bill=bill+(dn*5);
                    }
                }
            }
        }
        void electricity::Display (void)
        {
            cout<<"\n Meter no:"<<mn;
            cout<<"\n Previous unit"<<pu;
            cout<<"\n Current units"<<cu;
            cout<<"\n No. of units consumed:"<<n;
            cout<<"\n Bill:"<<bill;
        }
        void main( )
        {
            electricity E;
            int a;
        abc:clrscr( );
            E.Getdata( );
            a=E.process( );
            if(a==0)
            {
            cout<<"\n Wrong data Reenter it";
            goto abc;
        }
        else
        {
            E.Calculate( );
            E.Display( );
        }
    }
```

Q. 59 Write a program in C++ to show how multiple constructors can be used in a class.

OR

Write a program in C++ to show overloading of constructors.

Ans. :

```
//Program for overloading constructor
#include<iostream.h>
#include<conio.h>
class interest
{
    private:
    float amount, rate, total;
    public:
```

```
        interest( )      //Default constructor
        {                //1st constructor
        amount=2000.0;
        rate=10.0;
        }
        interest (float x, float y) //IInd constructor
        {                //Parameterized constructor
        amount=x;
        rate=y;
        }
    void process (void);
    };
    void interest::process (void)
    {
        total=amount+((rate*amount)/100);
        cout<<"Total="<<total;
    }
    void main( )
    {
        interest I1, I2 (500.0, 5.0);
        cout<<"Default constructor:\n";
        I1.process( );
        cout<<"\n Parameterized constructor:\n";
        I2.process( );
    }
```

## Operator Overloading and Type Conversions

Q. 60 What is operator overloading ? Explain with suitable example. OR (March 2004, 15 Oct. 2004)

Explain operator overloading with illustration. Write the advantages of operator overloading. (Oct. 2003)

Ans. :

1) The mechanism of giving some special meaning to an operator is called as operator overloading.
2) In C++, the user defined data types behave in much the same way as the built-in data types.
3) For instance, C++ permits to add two variables of user-defined data types with the same syntax as the basic types. This means that C++ has the ability to provide operators with a special meaning for a data type. This is nothing but operator overloading.
4) Operator overloading provides a flexible option for the creation of new definitions for most of the C++ operators.

5) When an operator is overloaded, its original meaning is not lost. For instance, the operator+ has been overloaded to add two vectors, can still be used to add two integers.

6) To define an additional task to an operator, a special function called 'operator function' is used to specify the relation of the operator to the class.

7) Following program shows overloaded ++ operator

e.g.

```
//increment counter variable with ++ operator
#include<iostream.h>
class counter
{
    private:
        int count;
    public:
        counter ( )
            {count = 0;}
        int get_count ( )
            {return count;}
        void operator ++ ( )
            {count ++ ;}
};
void main ( )
{   counter C1, C2;
    cout <<"C1 = "<<C1.get_count ( );
    cout < "C2 = "<<C2.get_count ( );
    C1++;
    C2++;
    ++C2;
    cout <<"C1 = "<<C1.get_count ( );
    cout <<"C2 = "<<C2.get_count ( );
}
```

In the above program, two objects of class counter : C1 and C2 are created. They are initially 0. Then using overloaded ++ operator, increment C1 once and C2 twice and display resulting values.

8) Advantages of operator overloading :

(i) Operator overloading concept extends capability of operators to operate on user-defined data.

(ii) It can also be applied to data conversion.

(iii) Using operator overloading technique, user-defined data types behave in much the same way as the build-in types.

Q. 61 What is operator overloading ? State the three steps involved in operator overloading ? **(Mar. 2003,13, 19; Oct. 2008, 10,11,12,13,21**

Ans. :

1) The mechanism of giving special meanings to an operator is known as operator overloading.

2) Operator overloading provides a flexible option for the creation of new definitions for most of the C++ operators.

3) To define an additional task to an operator, a special function called operator function is used.

4) The process of overloading involves the following steps :
   (a) First create a class that defines the data type that is to be used in the overloading operation.
   (b) Declare the operator function operator op ( ) in the public part of the class. It may be either a member function or a friend function.
   (c) Define the operator function to implement the required operations.

Q. 62 What is an operator function ? Describe the syntax of an operator function. Explain the difference between operator function as member function and friend function.
(Mar. 2002,06,07,11,16, 17, 19; Oct.2010, July 2016

Ans. :

1) To define an additional task to an operator, it specify what it means in relation to the class to which the operator is applied. This is done with the help of a special function, called operator function, which describes the task.

2) In short, a function which defines additional task to an operator or which gives a special meaning to an operator is called the operator function.

3) The general form of operator function is,

```
return-type class_name::operator op(argument list)
{
function body//task defined
}
```

Where return type is the type of value returned by the specified operation and op is the operator being overloaded.

The op is preceded by the keyword operator. Operator op is the function name.

4) Operator functions must be either member functions (non-static) or friend functions.

5) The basic difference between operator function as a friend function and as a member function is that a friend function will have only one argument for unary operators and two for binary operators, while a member function has no arguments for unary operators and only one for binary operators. This is because the object used to invoke the member function is passed implicitly and therefore is available for the member function. This is not the case with friend function. Arguments may be passed either by value or by reference.

Q. 63 State any eight rules for overloading operators.
(Mar.13, 19; Oct. 07,10,12,15, July 17, Dec. 2020

Ans. :

There are certain restrictions and limitations for overloading operators. Some of them are listed below :

1) Only existing operators can be overloaded. New operators cannot be created.

2) The overloaded operator must have atleast one operand that is of user-defined type.

3) The basic meaning of an operator cannot change. i.e. we cannot redefine the plus (+) operator to subtract one value from the another.

4) The overloaded operators follow the syntax rules of original operators.

5) Following are some operators that cannot be overloaded.

| | |
|---|---|
| Size of | Size of operator |
| . | Membership operator |
| .* | Pointer to member operator |
| : : | Scope resolution operator |
| ? : | Conditional opeator |

6) Following certain operators cannot be overloaded using friend functions but member functions can be used to overload them.

| | |
|---|---|
| = | Assignment operator |
| ( ) | Function call operator |
| [ ] | Subscripting operator |
| → | Class member access operator |

7) Unary operators, overloaded by means of a member function, take no explicit arguments and return no explicit values.

8) Unary operators, overloaded by means of a friend function take one reference argument.

9) Binary operators overloaded through a member function take one explicit argument.

10) Binary operators overloaded through a friend function takes two explicit arguments.

11) When using binary operators overloaded through a member function, the left hand operand must be an object of the relevant class.

12) Binary arithmetic operators such as +, –, * and / must explicitly return a value. They must not attempt to change their own arguments.

Q. 64 Enlist the operators which cannot be overloaded and the operators where friend functions cannot be used.

Ans. :

1) Operators which cannot be overloaded :

| | | |
|---|---|---|
| (i) | sizeof | - Size of operator. |
| (ii) | . | - Membership operator |
| (iii) | .* | - Pointer to member operator. |
| (iv) | :: | - Scope resolution operator. |
| (v) | ?: | - Conditional operator. |

2) Operators where friend function cannot be used :

| | | |
|---|---|---|
| (i) | = | - assignment operator |
| (ii) | ( ) | - function call operator |
| (iii) | [ ] | - subscripting operator |
| (iv) | → | - class member access operator |

Q. 65 Write a short note on type conversions.

Ans. :

1) When constants and variables of different types are mixed in an expression, compiler applies automatic type conversions to the operands as per certain rules.
2) Similarly, an assignment operator also causes the automatic type conversions.
3) The type of data to the right of an assignment operator is automatically converted to the type of variable on the left.
4) e.g.

```
int x;
float y;
y=29.123;
x=y;
```

These statements convert y to an integer before its value is applied to x. Thus, the fractional part is truncated.

5) The type conversions are automatic as long as data types involved are built-in types.
6) Consider the following statement that adds two objects and then assigns the result to a third object.

V3 = V1 + V2; //V1, V2, V3 are class type objects.

When the objects are of same class type, the operations of addition and assignment are carried out smoothly and compiler makes no complaint.

7) But, if one operand is an object and other is built-in type variable or the objects are of different classes, then compiler gives error.
8) Since, the user-defined data types are designed by us to suit our requirements, the compiler does not support automatic type conversions for such data types. Therefore, design the conversion routines, if such operations are required.
9) Three types of situations may arise in the data conversion between uncompatible types.
   1. Conversion from built-in type to class type.
   2. Conversion from class type to built-in type.
   3. Conversion from one class type to another class type.

Q. 66 Explain the three types of data conversion in C++ with a suitable example.

(March 2005, 10, 19; July 18

Ans. : Three types of data conversion in C++ are as follows :

(i) Conversion from built-in type to class type.
(ii) Conversion from class type to built-in type.
(iii) Conversion from one class to another class.

(i) Basic to class type :

The constructor can be used for default type conversion from argument's type to the constructor's class type.

For e.g.

```
class time
{
int hr;
int min;
public :
```

```
time (int t) // constructor
{
hr = t / 60; //t in minutes
min = t % 60;
}
};
```

The following conversion statements can be used in a function.

```
time T1; // object T1 is created.
int duration = 90;
T1 = duration; // int to class type
```

After this conversion, the hr member of T1 will contain a value of 1 and min contain 30 means 1 hour and 30 minutes.

(ii) Class to basic type :

Overloaded casting operator is used to convert a class type data to basic type. The general form is as :

```
operator typename ( )
{
.....(function statement)
.....
}
```

The conversion function must satisfy following conditions :

(a) It must be a class member.

(b) It must not specify a return value.

(c) It must not have any arguments.

For e.g.

```
Time : : operator int ( )
{
int min1 = hr * 60;
min1 = min1 + min;
return min1;
}
```

The operator int ( ) can be used as follow :

```
Time T1 ; // T1 object
int m = T1; // Class to basic
```

After the conversion 1 hr 30 mins can be converted into 90 minutes.

(iii) One class to another class type :

Use one-argument constructor or conversion function depends upon the defining conversion routine in source class or destination class.

For e.g. :

obj A = obj B

↓ ↓

Destination Source

Constructor is placed in the destination class and conversion function is placed in source class.

Q. 67 Write a program in C++ to overload unary minus operator, so that unary minus operator when applied to an object should change the sign of each of its data items.

Ans. :

```
//Negate all data items of class with - operator
#include<iostream.h>
#include<conio.h>
class space
{
    private:
    int x, y, z;
    public:
        space (int a, int b, int c)
        {
        x=a;
        y=b;
        z=c;
        }
        void display (void);
        void operator-( ); //overloaded unary-
        };
        void space::display( )
        {
        cout<<x<<"\t"<<y<<"\t"<<z;
        }
        void space::operator-(void)
        {
        x=-x;
        y=-y;
        z=-z;
        }
        void main( )
        {
        int l, m, n;
        space S;
        cout<<"Enter three numbers \n";
        cin>>l>>m>>n;
```

```
S=space (l, m, n);
cout<<"S:";
S.display( );
- S;
cout<<"-S:";
S.display( );
}
```

**Q. 68 Write a program in C++ to overload binary + operator for addition of two complex numbers.**

**Ans. :**

```
//C++ program to overload binary + operator for addition of two complex numbers.
#include<iostream.h>
#include<conio.h>
class complex
{
        private:
        float x;        //real part
        float y;        //imaginary part
        public:
        complex (float, float);
        complex operator + (complex);
        void display (void);
};
complex::complex (float real, float imag)
{
        x=real;
        y=imag;
}
void complex::display (void)
{
        cout<<x<<"+"<<y<<"i\n";
}
complex complex::operator+(complex c)
{
        complex temp;
        temp.x=x+c.x;
        temp.y=y+c.y;
        return(temp);
}
void main( )
{
        complex C1, C2, C3;
        C1=complex (3.5, 2.5);
        C2=complex (1.1, 1.7);
        C3=C1+C2;               //operator function invoked
```

```
        cout<<"C1=";
        C1.display( );
        cout<<"C2=";
        C2.display( );
        cout<<"C3=";
        C3.display( );
}
```

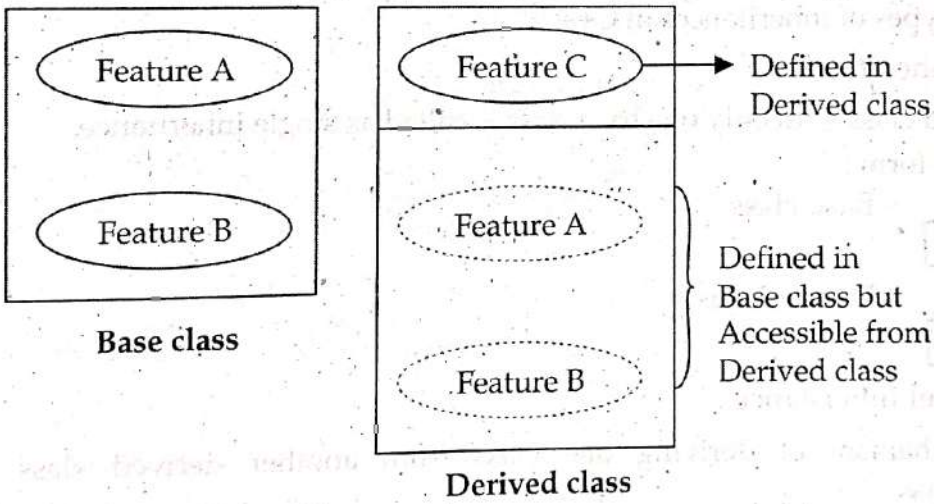
## INHERITANCE

Q. 69 What is inheritance ? Explain with suitable example.

(Mar.2015; Oct. 2004,06,08,10,11, July 2016, Dec. 2020

Ans. :

1) The mechanism of deriving a new class from an old one is called as inheritance.
2) The old class is referred as base class and new class is referred as derived class.
3) C++ strongly supports the concept of reusability. Once a class has been written and tested, it can be adapted by other programmers to suit their requirements.
4) This is basically done by creating new classes, reusing the properties of the existing ones.
5) Functions and variables of a class that has been tested can be used by object of another class. This is known as inheritance.
6) The reuse of a class that has already been tested, debugged and used many times, can save the efforts of developing and testing the same again.
7) Figure shows inheritance :




8) The syntax of declaration of derived class is :

```
class derived class_name:visibility_mode base_class_name
{
...
...        // Members of derived class
...
};
```

where visibility mode is optional and if present may be either private or public.

```
class base
{
        public :
        void showbase ( )
        {
        cout<<"This is the base";
        }
};
class derived:public base //Declaration of derived class
{
        public:
        void showderived (void)
        {
        showbase ( ); //Base class function used
        cout<<"\n This is derived class";
        }
};
```

Q. 70 Explain different types of inheritances with suitable diagram.

(March 2003,05,09,14,15; 17 ; Oct. 2002,04,06,08,10,11, July 2016,19

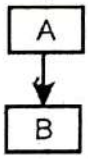
Ans. :

There are five types of inheritances in C++ :

(i) Single inheritance :

A derived class with only one base class is called as single inheritance.

It has the form :




– Base class

– Derived class

(ii) Multilevel inheritance :

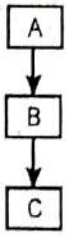
The mechanism of deriving one class from another derived class is multilevel inheritance.

It has the form :

Base class .... Grand father

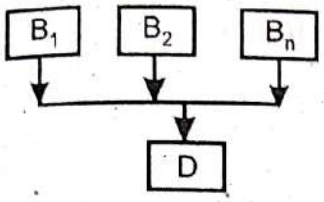
Intermediate base class .... Father

Derived class ....Child



(iii) Multiple inheritance :

When a class is derived from several base classes, it is called as multiple inheritance.



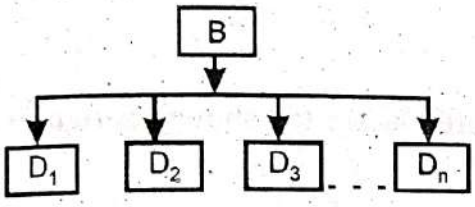

....Base classes

.....Derived class

(iv) Hierarchical inheritance :

The traits of one class may be inherited by more than one classes. This process is known as hierarchical inheritance.

It has the form :

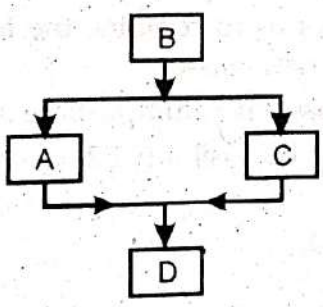



.....Base class

.....Derived classes

(v) Hybrid inheritance :

The inheritance which involves more than one inheritances is called as hybrid inheritance. For e.g. : Above figure involves hierarchical, multiple and multilevel inheritances and the resultant inheritance is called hybrid inheritance.



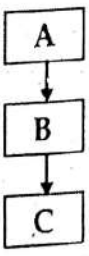

Q. 71 Explain multilevel and multiple inheritance in detail.

Ans. :

1) Multilevel inheritance :

The mechanism of deriving one class from another derived class is called as multilevel inheritance.

Following figure shows multilevel inheritance.




Base class – Grand father

Intermediate base class – Father

Derived class – Child

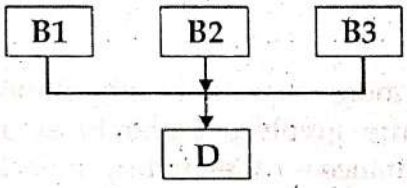
The class A serves as a base class for derived class B which in turn serves as a base class for the derived class C. The class B is known as intermediate base class since it provides link for the inheritance between A and C. The chain ABC is known as inheritance path.

A derived class with multilevel inheritance is declared as follows -

```
class A                 //Base class
{___
};
class B:public A        //B derived from A
{___
};
class C:public B        //C derived from B
{___
};
```

2) Multiple inheritance :

A class can inherit the attributes of two or more classes (as shown in figure). This is known as multiple inheritance.




– Base class

– Derived class

Multiple inheritance allows us to combine the features of several existing classes as a starting point for defining new classes.

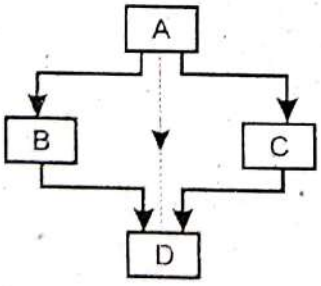
The syntax of a derived class with multiple base classes is as follows :

```
class D:visibility B1, visibility B2, ....
{
(Body of class D)
};
```

where, visibility may be either public or private. The base classes are separated by commas.

Q. 72 What is virtual base class ? Why is it necessary to define virtual base classes in some cases of hybrid inheritance ?

Ans. :




- Grandparent

- Parent

- Child

1) Sometimes, when hybrid inheritance is used, there are at least three levels as shown in below figure Here, hierarchical, multiple and multilevel inheritances are used to implement hybrid inheritance.

2) In figure classes B and C are derived from class A and class D is derived from classes B and C. So, class D can inherit members of class A by two paths :

Path I$^{st}$ : - A → B → D

Path II$^{nd}$ :- A → C → D

3) Class D can also inherit members of A directly as shown in figure by dotted line. The grand parent (i.e. class A) is sometimes referred as indirect base class.

4) This means that class D may contain duplicate sets of members of class A i.e. the members of class A are inherited in class D twice via class B and via class C, This produces ambiguity.

To avoid this ambiguity, concept of virtual base class is used.

5) Thus, the duplication of inherited members due to multiple paths can be avoided by making the common base class (ancestor class or grand- parent) as virtual base class while declaring the direct or intermediate base classes as follows :

```
class A //grandparent
{.....
};
class B:virtual public A     //parent 1
{.....
};
class C:public virtual A     //parent 2
{.....
};
class D:public B, public C   //child
{.....    //only one copy of A will be
.....     //inherited
};
```

6) When a class is made a virtual base class, C++ takes necessary care to see that only one copy of that class is inherited, regardless of how many inheritance paths exist between the virtual base class and the derived class.

7) The keywords virtual and public may be used in either order.

**Q. 73 What is single inheritance ? Write a program to implement single inheritance.**

Ans. :

1) A derived class with only one base class is called single inheritance.

2) For e.g.

Here B is base class and D is derived class. The class B contains one private data member, one public data member and three public member functions. The class D contains one private data member and two public member functions.

```
//Single Inheritance
    #include<iostream.h>
    #include<conio.h>
    class B                 //Base class
    {
    private:
            int a;          //Private member, not inheritable.
    public:
            int b;          //public; ready for inheritance
            void get_ab (void);
            int get_a (void);
            void show_a (void);
    };
    class D:public B //Derived class (public derivation)
    {
             private:
             int c;
             public:
             void mul (void);
             void display (void);
     };
     //....Functions definition .....
     void B::get_ab (void)
     {
             a=5; b=10;
     }
             int B::get_a (void)
     {
             return (a);
     }
             void B::Show_a (void)
     {
             cout<<"a="<<a<<endl;
     }
             void D::mul (void)
     {
             c=b*get_a( );
     }
             void D::display (void)
     {
             cout<<"a="<<get_a( );
```

```
        cout<<"\n b="<<b;
        cout<<"\n c="<<c;
}
        //....main program ....
void main ( )
{
        D d;
        d.get_ab( );
        d.mul( )
        d.display( );
}
```

The output of above program is :

a = 5

b = 10

c = 50

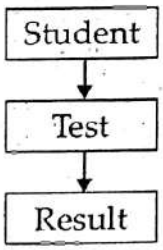
Q. 74 What is multilevel inheritance ? Write a program in C++, to implement multilevel inheritance :

Ans. :

The mechanism of deriving one class from another derived class is called as multilevel inheritance.

For eg.

Here student is a class, which stores roll number. Class test stores the marks obtained in two subjects and class result contains total marks obtained in test. The class result inherits the details of the marks obtained in the test and roll number through multilevel inheritance.




```
//Multilevel Inheritance
    #include<iostream.h>
    #include<conio.h>
    class student //Base class
    {
    protected:
        int roll_number;
    public:
        void get_number (int);
        void put_number (void);
```

```
};
class test:public student      //Intermediate
{                              //base class
        protected:
        float sub1;
        float sub2;
        public:
        void get_marks (float, float);
        void put_marks (void);
};
class result:public test //Derived class
{
private:
        float total;
public:
        void display (void);
};
        //.....functions definition.....
        void student::get_number (int a)
{
        roll_number=a;
}
        void student::put_number (void)
{
        cout<<"Roll number"<<roll_number<<"\n";
}
        void test::get_marks (float x, float y)
{
        sub1=x;
        sub2=y;
}
        void test::put_marks (void)

{
        cout<<"Marks in sub1="<<sub1<<"\n";
        cout<<"Marks in sub2="<<sub2<<"\n";
}
        void result::display (void)
{
        total=sub1+sub2;
        put_number( );
```

```
        put_marks( );
        cout<<"Total="<<total<<"\n";
}
        void main( )
{
        result student1;
        student1.get_number (127);
        student1.get_marks(98, 99.2);
        student1.display( );
}
```

Q. 75 What is multiple inheritance ? Write a program to implement multiple inheritance.

Ans. : "The derivation of one class from several base classes is called as multiple inheritance."

```
//Multiple Inheritance
#include<iostream.h>
#include<conio.h>
class M          //Parent Ist
{
protected:
        int m;
public:
        void get_m(int a)
        {
        m=a;
        }
};
class N          //Parent IInd
{
protected:
int n;
public:
void get_n(int b)
        {
        n=b;
        }
};
class P:public M, public N //child
{
        public:
        void display (void);
};
        void P::display (void)
```

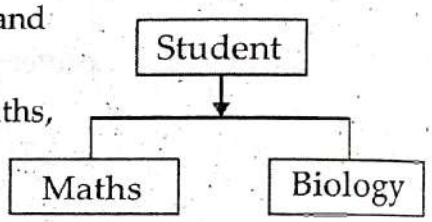
```
{
        cout<<"\n m="<<m;
        cout<<"\n n="<<n;
        cout<<"\n m*n="<<m*n;
}
        void main( )
        {
        P p;
        p.get_m(10);
        p.get_n(20);
        p.display( );
        }
```

Q. 76 What is hierarchical inheritance ? Write a program to implement hierarchical inheritance.

Ans. : "The inheritance, in which two or more classes are derived from same base class is called as hierarchical inheritance."

For e.g.

Here student is a class, which stores the name of student and roll number and marks obtained by student in physics and chemistry. Class maths contain marks obtained in maths, which is derived from class student and calculates PCM grouping. Class Biology, derived from student contains marks obtained in biology and calculates PCB grouping of student.




```
//Hierarchical Inheritance
    #include<iostream.h>
    #include<conio.h>
    class student
    {
    protected:
            char name[30];
            int phy, che, roll;
    public:
            void getdata (void);
            void display (void);
    };
            class maths:public student
    {
    protected:
            int math;
            float pcm;
    public:
            void get_maths (void);
```

```
        void show_maths (void);
};
        class biology: public student
{
protected:
        int bio;
        float pcb;
public:
        void get_bio (void);
        void show_bio (void);
};
        void student::getdata (void)
{
        cout<<"Enter name and roll number";
        cin>>name; cin>>roll;
        cout<<"\n Enter marks in physics:";
        cin>>phy;
        cout<<"\n Enter marks in chemistry:";
        cin>>che;
}
        void student::display (void)
        {
        cout<<"Name:"<<name;
        cout<<"\n Roll No:"<<roll<<endl;
        }
        void maths::get_maths (void)

        {
        cout<<"\n Enter marks in maths:";
        cin>>math;
        pcm=(phy+che+math)/3;
        }
        void maths::show_maths(void)
        {
        cout<<"\n Marks in physics:"<<phy;
        cout<<"\n Marks in chemistry:"<<che;
        cout<<"\n Marks in maths:"<<math;
        cout<<"\n PCM grouping:"<<pcm<<endl;
        }
        void biology::get_bio (void)
        {
        cout<<"Enter marks in biology:";
        cin>>bio;
        pcb=(phy+che+bio)/3;
        }
        void biology::show_bio (void)
```

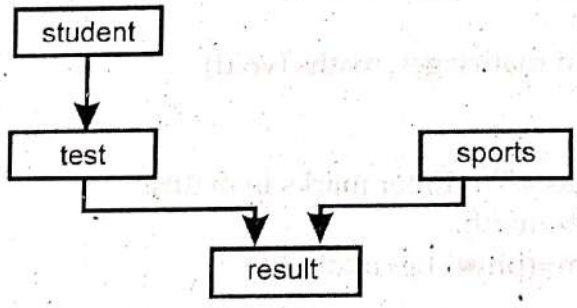
```
{
cout<<"\n Marks in physics:"<<phy;
cout<<"\n Marks in chemistry:"<<che;
cout<<"\n Marks in biology:"<<bio;
cout<<"\n PCB grouping:"<<pcb;
}
void main( )
{
maths M;
biology B;
M.getdata( );
M.get_maths( );
B.get_bio( ); clrscr( );
B.display( );
M.show_maths( );
B.show_bio( );
}
```

Q. 77 What is hybrid inheritance ? Write a program to implement hybrid inheritance.

Ans. : "The inheritance, which involves two or more other inheritances is called hybrid inheritance."

For e.g.

Here student is base class of test. Class result is derived from classes test and sports. The class relationships are shown in following figure.

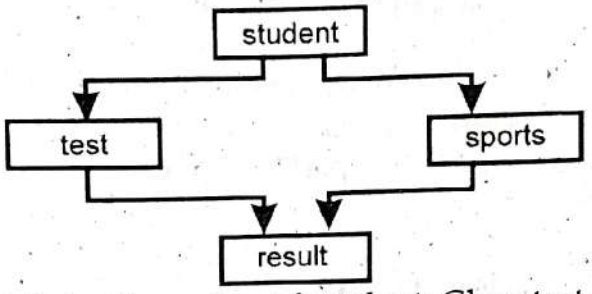



```
//Hybrid Inheritance
#include<iostream.h>
class student
{
protected:
        int roll_number;
public:
        void get_number (int a)
        {
        roll_number=a;
        }
        void put_number (void)
        {
        cout<<"\n Roll No:"<<roll_number;
```

```
        }
};
        class test:public student
        {
        protected:
        float part1, part2;
public:
        void get_marks (float x, float y)
        {
        part1 = x;
        part2 = y;
        }
        void put_marks (void)
        {
        cout<<"Marks obtained:\n";
        cout<<"part1="<<part1;
        cout<<"\n part2="<<part2;
        }
};
        class sports
{
protected:
        float score;
public:
        void get_score (float S)
        {
        score = S;
        }
        void put_score (void)
        {
        cout<<"\n Score="<<score;
        }
};
        class result:public test, public sports
{
protected:
        float total;
public:
        void display (void);
};
        void result::display (void)
        {
        total=part1+part2+score;
        put_number( );
        put_marks( );
        put_score( );
```

```
cout<<"\n Total="<<total;
}
void main( )
{
result R;
R.get_number (127);
R.get_marks(83.0, 83.0);
R.get_score(83.0);
R.display( );
}
```

Q. 78 Write a program to implement hybrid inheritance using virtual base class. The class relationships are shown in following Figure.




Class student contains roll number of student. Class test contains marks obtained in two subjects and class sports contains score. Class result should calculate total and display it.

Ans. :

```
//Virtual Base Class
#include<iostream.h>
class student
{
protected:
        int roll_number;
public:
        void get_number (int a)
        {
        roll_number=a;
        }
        void put_number (void)
        {
        cout<<"\n Roll No:"<<roll_number;
        }
};
        class test:virtual public student
{
protected:
        float part1, part2;
public:
        void get_marks (float x, float y)
        {
```

```
        part1=x;
        part2=y;
        }
        void put_marks (void)
        {
        cout<<"\n Part1="<<part1;
        cout<<"\n Part2="<<part2;
        }
};
        class sports:public virtual student
{
protected:
        float score;
public:
        void get_score (float s)
        {
        score=s;
        }
        void put_score (void)
        {
        cout<<"\n Score:"<<score;
        }
};
        class result:public test, public sports
        {
protected:
        float total;
public:
        void display (void);
};
        void result::display (void)
        {
        total=part1+part2+score;
        put_number( );
        put_marks( );
        put_score( );
        cout<<"\n Total="<<total;
        }
        void main( )
        {
        result R;
        R.get_number (127);
        R.get_marks (83.0, 83.0);
        R.get_score (83.0);
        R.display( );
        }
```

## VIRTUAL FUNCTIONS AND POLYMORPHISM

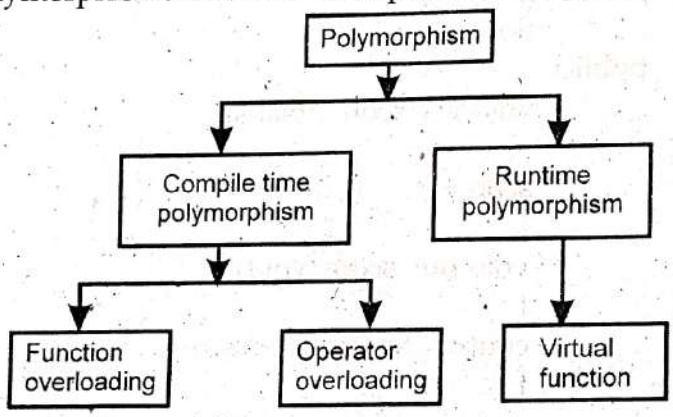
Q. 79 What is polymorphism ? Explain runtime and compile time polymorphism. (Oct. 2006, 21, Mar. 2009, July 2017, Dec. 2020

What does polymorphism in C++ ? How is the same achieved at –

(i) Compile time (ii) Runtime ? (Oct. 2002,03,05,14 ; Mar. 2006,13,16,20

Ans. :

1) "Polymorphism refers to identically named methods (member functions) that have different behavour depending on the type of object they refer."
2) Polymorphism simply means "one name, multiple forms."
3) The types of polymorphisms and their examples are shown in following figure.




I) Compile time polymorphism :

1) Function overloading and operator overloading are the examples of compile time polymorphism.
2) In this case, the overloaded member functions are selected for invoking by matching arguments, both type and number.
3) This information is known to the compiler at the compile time and, therefore the compiler is able to select the appropriate function for a particular call at the compile time itself. This is known as compile time polymorphism.
4) Compile time polymorphism is also called as early binding or static binding or static linking. Early binding simply means that an object is bound to its function at compile time.

II) Runtime polymorphism :

1) In some situations, it is nice to select appropriate member function to be invoked while the program is running. This is known as runtime polymorphism.
2) e.g. consider a situation where the function name and prototype is the same in both the base and derived classes as shown in following class definitions.

```
class A
{
        int x;                  // private by default.
public:
        void show (void)    // show( ) in base class
```

```
        { ... }
};
        class B: public A
{
int y;
public:
        void show (void)    // show( ) in derived class
        { ... }
};
```

Here, show( ) function is used to print values of object of both the classes A and B. The prototype of show( ) is the same in both the places, the function is not overloaded and therefore static binding does not apply.

3) In such situations, the appropriate member function can be selected at runtime and it is known as runtime polymorphism.
4) To achieve runtime polymorphism, C++ supports mechanism of virtual functions.
5) At runtime, it is known what class objects are under consideration, the appropriate version of function is called.
6) Since the function is linked with a particular class much after its compilation, this process is termed as late binding. It is also called as dynamic binding because the selection of the appropriate function is done dynamically at runtime.

Q. 80 Explain the concept of virtual functions. **(July 2019)**

Ans. :

1) When user use the same function name in both the base and derived classes, the function in base class is declared as virtual using the keyword 'virtual' preceding its normal declaration.
2) When a function is made virtual, C++ determines which function to use at runtime, based on the type of object pointed to by the base pointer.
3) Thus, by making the base pointer to point two different objects, it can execute different versions of the virtual function.
4) Virtual functions can be accessed through the use of a pointer declared as a pointer to the base class.
5) Also, the prototypes of the base class version of a virtual function and all the derived class versions must be identical.
6) If two functions with the same name have different prototype, C++ considers them as overloaded functions, and the virtual function mechanism is ignored.

Q. 81 Explain the difference between static and dynamic binding with example. **(March 2003, Dec. 2020)**

Ans. :

1) In static binding, object is bound to its function call at compile time.
While in dynamic binding, selection of the appropriate function is done dynamically at runtime.

2). In static binding, compiler knows the function information (argument type, number etc) at the compile time itself so as able to select appropriate function for a particular call (also called early binding). In dynamic binding, function is linked with a particular class much later after the compilation (also known as late binding).

3) Function overloading and operator overloading are the examples of static binding. Virtual functions are used to implement dynamic binding.

4) For example :

Consider the following class definitions :

```
class A
{       int x;
        public :
        void show ( )
        {       cout << "Base class";
        }
};
class B : public A
{       int y;
        public :
                void show ( )
                { cout << "'Derived class";
                }
};
```

In above example, compiler does not know which show ( ) function is executed either of base class or derived class. So compiler defers this decision and at the run time select appropriate function concept of virtual function.

In dynamic binding, classes are defined as :

```
class A
{       int x;
        public :
        virtual void show ( )
        { cout << "Base class"; }
};
class B : public A
{       int y;
        public :
        void show ( )
        {       cout << "'Derived class";}
};
```

Q. 82 State any eight basic rules for virtual functions that satisfy the compi requirements. (March 2002,07,14,18,22; Oct. 2005,07,13; July 2018,

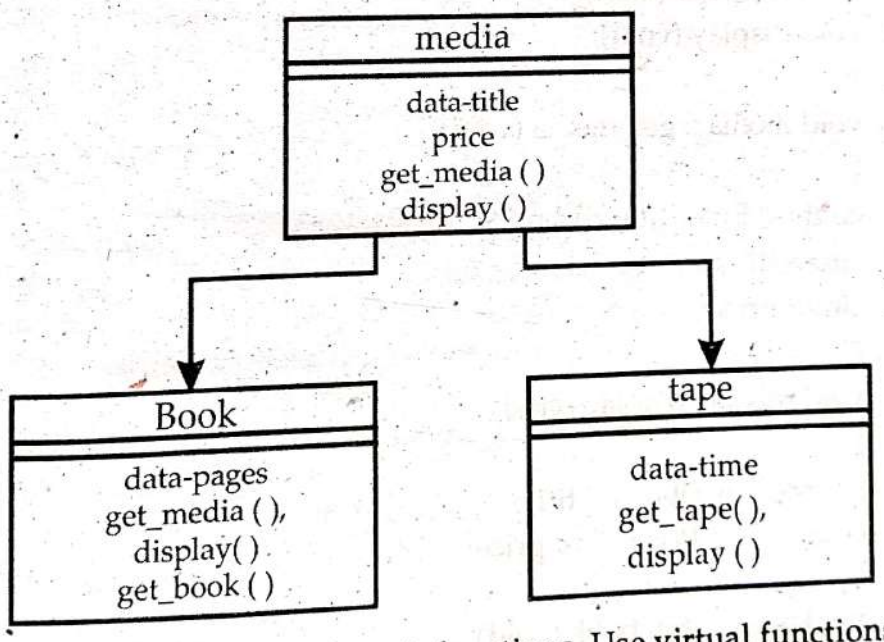
Ans. : When virtual functions are created for implementing late binding, we should observ following basic rules that satisfy the compiler requirements :

1) The virtual functions must be members of some class.
2) They cannot be static members.
3) They are accessed by using object pointers.
4) A virtual function can be a friend of another class.
5) A virtual function in a base class must be defined, eventhough it is not used.
6) The prototypes of the base class version of virtual function and all derived class version must be identical. If two functions have different prototypes, then C++ considers them as overloaded functions and not as virtual functions.
7) We cannot have virtual constructors, but we can have virtual destructors.
8) A base pointer can point to any type of derived object, the reverse is not true. i.e. we cannot use a pointer to derived class to access an object of the base type.
9) When base pointer points to derived class, the incrementation and decrementation is only relative to its base type.
10) Virtual functions are defined in base class, they need not be redefined in derived class.

Q. 83 Write a program to declare a class media, which contains title and price. Declare another two classes book and tape with base class media, which contains pages and time respectively.

Read book details and display it.

The class relationships and functions are shown below :




and You may use additional data & functions. Use virtual functions.

Ans. :
```
// Runtime Polymorphism
#include <iostream.h>
#include <conio.h>
class media
{
protected:
char title [20];
float price;
```

```
public:
void get_media (void);
virtual void display (void);
};
class book : public media
{
protected:
        int pages;
public:
void get_book (void);
void display (void);
};
class tape: public media
{
protected:
        int time;
public:
void get_tape (void);
void display (void);
};
void media :: get_media (void)
{
cout<<"Enter title and price" ;
cin>> title;
cin>> price;
}
void media:: display (void)
{
cout<<"\n Title : << title;
cout<<" \n Price : "<< price;
}
void book :: get_book (void)
{
cout<<"\n Enter number of pages";
cin>> pages;
}
void book :: display (void)
{
cout<<"\n Pages" <<pages;
}
void tape :: get_tape (void)
```

```
{
        cout<< "\n Enter time";
        cin>> time;
}
void tape :: display (void)
{
        cout<<"\n Time" <<time;
}
void main( )
{
media m,*p;
book B;
tape T;
m.get_media( );
B.get_book( );
T.get_tape( );
clrscr( );
p = &m;
p → display( )
p = &B;
p → display( );
p = &T;
p → display( );
}
```

Q. 84 Create a base class shape. Use this class to store two double type values that could be used to compute the area of figures. Derive two specific classes called triangle and rectangle from the base shape. Add to the base class a member function get_data() to initialize base class data members and another member function display_area( ) to compute and display area of figures. Make display_area( ) as a virtual function and redefine this function in the derived classes to suit their requirements.

Using these three classes design a program that will accept dimensions of a triangle or rectangle interactively and display the area.

Remember the two values gives as input will be treated as lengths of two sides in case of rectangle and as base and height in case of triangle. Use the following formulae,

Area of rectangle = x * y

Area of triangle = 1/2 * x * y

Ans. :

```
// C++ program using virtual function
#include <iostream.h>
#include <conio.h>
class shape
{
```

```
protected:
        double x,y;
public:
        void get_data(void);
virtual void display_area(void) // Empty virtual function
{       }
};
class triangle : public shape
{
protected:
        double at;
public:
        void display_area (void);
};
class rectangle: public shape
{
protected:
        double ar;
public:
        void display_area (void);
};
void shape :: get_data (void)
{
        cout <<"\n Enter base and height";
        cin>> x >> y;
}
void triangle :: display_area (void)
{
        at = (1/2)*x*y;
        cout"<<"\n Area of triangle is" << at;
}
void rectangle :: display_area (void)
{
        ar = x*y;
        cout<< "\n Area of rectangle is :" << ar;
}
void main( )
{
        shape s, *P;
        triangle t;
```

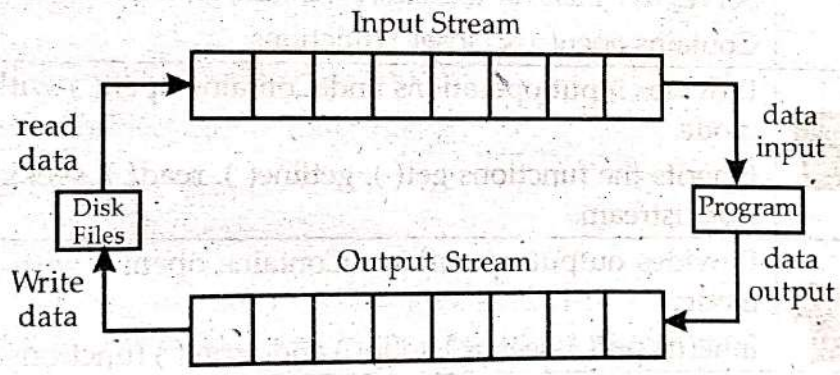
```
rectangle r;
s.get_data( );
P = &t;
P → display_area( );
P = &r;
P → display_area( );
}
```

## WORKING WITH FILES

**Q. 85 What are input and output streams ?** (March 2006, 2009, Oct. 2021)

Ans. :

1) The I/O system of C++ handles file operations which are very much similar to console input and output operations.
2) It uses file streams as an interface between the programs and the files.
3) The streams that supply data to the program is known as input stream, while the stream that receives data from the program is known as output stream.
4) In other words, input stream extracts (or reads) data from the file and the output stream inserts (or writes) data to the file. This is illustrated in following Figure.
5) The input operation involves the creation of an input stream & linking it with the program and the input file. Similarly, the output operation involves establishing an output stream and linking it to the program and the output file.



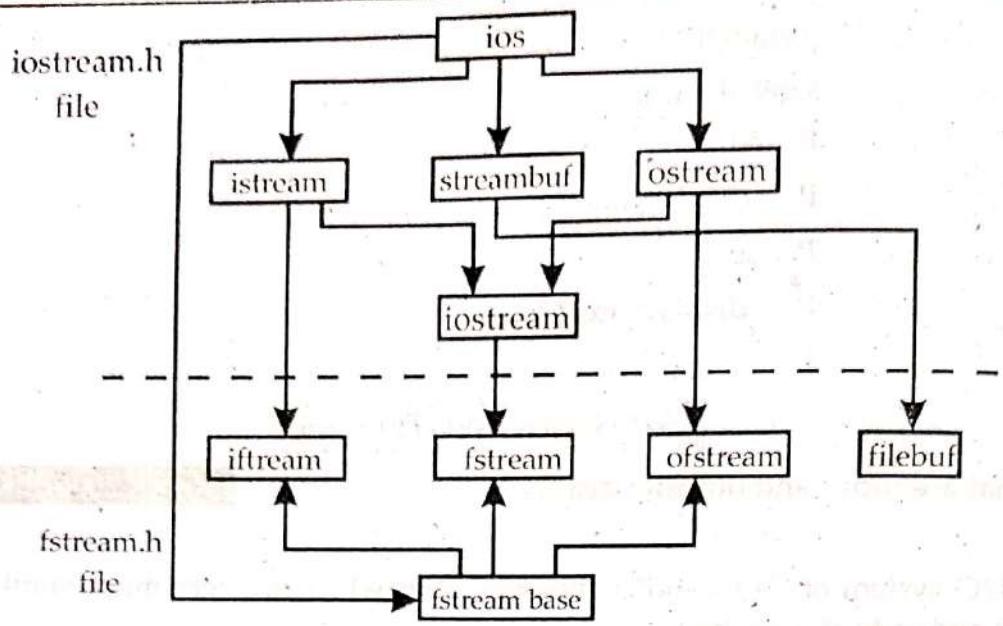

File input and output streams

**Q. 86 Describe the various classes available for file operations.** (Oct. 2002)

OR **Describe briefly the features of I/O system supported by C++ with suitable example.** (Oct. 2004)

Ans. :

1) The I/O system of C++ contains a set of classes that defines the file handling methods. These include ifstream, ofstream and fstream.
2) These classes are derived from fstreambase and the corresponding iostream. h class as shown in the following figure.
3) These classes designed to manage the disk files, are declared in fstream. h and therefore, we must include this file in any program that uses files.




Stream classes for file operations

The details of file stream classes is given in the following table.

| Class | Contents |
|---|---|
| 1. filebuf | Its purpose is to set the file buffers to read and write. Contains 'openprot' constant and used in the 'open( )' of file stream classes. Also contains close ( ) and open( ) as members. |
| 2. fstreambase | Provides operations common to the file streams. Serves as a base for fstream, ofstream, & ifstream classes. Contains open( ) & close( ) functions. |
| 3. ifstream (March 2020 22, Dec. 2020) | Provides input operations and Contains open( ) with default input mode. Inherits the functions get( ), getline( ), read( ), seekg( ), and tellg( ) from istream. |
| 4. ofstream (March 2020 22 Dec. 2020) | Provides output operations. Contains open( ) with default output mode. Inherits put( ), seekp( ), tellp( ) and write( ) functions from ostream. |
| 5. fstream (March 2020 22, Dec. 2020) | Provides support for simultaneous input and output operations. Contains open( ) with default input mode. Inherits all the functions from istream and ostream classes through iostream. |

Q. 87 What is the function of each of each of the following file stream classes ?
(i) ifstream (ii) ofstream (iii) filebuf (Mar. 08 ; Oct. 2003,12

Ans. :

(i) ifstream : Provides input operations. This file stream class is used to read a stream of objects from a file. (March 2020

(ii) ofstream : Provides output operations. Ofstream class is used to write a stream of objects in a file. (March 2020

(iii) filebuf : Its purpose is to set the file buffers to read and write. It contains close( ) and open ( ) as members.

Q. 88 State the details of the following file stream classes : (i) ifstream (ii) ofstream (March 2005)

Ans. :

(a) ifstream :

(1) This class is used for file handling methods.

(2) This class is designed to manage the disk files and user must include this file in any program that uses files.

(3) It provides input operations.

(4) It contains open ( ) function wit default input mode.

(5) It inherits get( ), getline( ), read( ), seekg( ) and tellg( ) functions from istream.

(b) ofstream :

(1) This class is also used for file handling methods.

(2) It provides output operations related to files.

(3) It contains open( ) function with default output mode.

(4) It inherits put( ), seekp( ), tellp( ) and write( ) function from ostream.

Q. 89 How we can open a file using open() function ? (July 2016, 18)

Ans. :

We can open files using open( ) by two ways :

1) In first method, open( ) function takes only one argument and that is file name.

This is done as follows :

```
file-stream-class stream_object;
stream_object.open("file name");
```

e.g. ofstream fout;

fout.open("Try");

2) A file can be also opened using open() by passing two arguments. The first argument is file name and the second argument is used to specify the file mode. The general form of open() with two arguments is,

```
stream_object.open("file name", mode);
```

The second argument mode specifies the purpose for which file is opened.

e.g. fout.open("computer", ios::app);

This opens the file in append mode.

Note : By using open( ), we can open a file explicitly, whereas, we can also open a file implicitly as.

```
file-stream-class-stream-object ("file name");
```

e.g. ofstream outf ("computer")

It will open file computer in output mode

Q. 90 What are different file modes ? (Oct. 2002,11)

Ans. :

The following table lists the file mode parameters and their meanings.

| | Parameter | Meaning |
|---|---|---|
| 1. | ios::app | Append to end-of-file. |
| 2. | ios::ate | Go to end-of-file on opening |
| 3. | ios::binary | Binary file |
| 4. | ios::in | Open a file for reading only |
| 5. | ios::nocreate | Open fails if the file does not exist. |
| 6. | ios::noreplace | Open fails if the file already exist. |
| 7. | ios::out | Open file for writing only. |
| 8. | ios::trunc | Delete contents of file, if it exist. |

Q. 91. What are classes in C++ for file stream operation ? How do you open and close file in C++ ? Explain any four file modes. (March 2004,12,16, July 2016)

Ans. : Classes for file stream operation :

(1) The I/O system of C++ contains a set of classes that define the file handling methods. They include :

(a) ifstream (b) ofstream

(c) fstream (Oct. 2012)

(2) These classes are derived from fstreambase and the corresponding iostream.h class.

(3) They are defined to manage the disk files and declared in fstream.h.

Opening and closing a file :

(1) The general format for opening a file is as :

```
file-stream-class stream-object;
stream-object.open ("filename");
```

(2) Here ifstream class is used to read a stream of objects from a file and ofstream class is used to write a stream of objects in a file.

(3) For example :

Open a file to read stream of object from 'data'.

```
ifstream infile;
infile.open ("data");
```

Open a file to write stream of object from 'data'.

```
ofstream outfile;
outfile.open ("data");
```

(4) Closing a file : Function close ( ) is used to close a file, which is opened for read, write or read and write operations. (July 2016)

For example : infile.close( );

File modes :

With class fstream, file mode can be specified. The form of open( ) function as Stream-object.open ("file name", mode)

File mode parameters are as follows :

1. ios::app – Append to end-of-file.
2. ios::ate – Go to end-of-file on opening.
3. ios::binary – Binary file.
4. ios::in – Open a file for reading only.

Q. 92 What are file pointers ?

Ans. :

1) "Each file has two associated pointers known as the file pointers". One of them is called the input pointer and the other is called the output pointer.
2) These pointers are used to move through the files while reading or writing.
3) The input pointer is used for reading the contents of a given file location and the output pointer is used for writing to a given file location.
4) Each time an input or output operation takes place, the appropriate pointer is automatically advanced.
5) Input pointer is also called as get pointer and output pointer is also called as put pointer.

Q. 93 Explain the purpose of following functions with example :
1) seekg( ) 2) seekp( ) 3) tellg( ) 4) tellp( )

(March 2018; July 2018, Dec. 2020

Ans. :

1) Seekg( ) : This function is used to move the file pointer forwards with given number of bytes. It has the form

```
seekg (unsigned int);
```

e.g.

```
ifstream inf("xyz.dat");
inf.seekg(10);
```

In above example, seekg( ) moves input pointer 10 bytes forward.
seekg is associated with input pointer or get pointer

2) Seekp( ) : This function is used to reposition file pointer to a given number of bytes. This function is associated with output pointer. Corresponding class to process this function is "ofstream" class.
It has the form :-

```
seekp (unsigned int);
```

e.g. ofstream outf("xyz.dat");
outf.seekp(10)
In above example, output pointer will point to 10th byte in file, after execution of outf.seekp(10);

3) tellg( ) : This function is used to return current file pointer position. This function is associated with input file stream. (July 2018)

e.g.

```
ifstream inf;
inf.open("xyz.dat");
int n;
n=inf.tellg( );
```

In above example, tellg( ) will return value zero, because initially, input pointer points to zeroth location.

4) tellp( ) : This function is used to return current file pointer (output pointer) position. It is associated with output file stream.

e.g.

```
ofstream outf;
outf.open("xyz",ios::app);
int n=outf.tellp();
```

In above example file is opened in append mode, therefore file pointer points to end-of-file character. Hence, tellp( ) returns number of characters present in file xyz.

Note : "Seek" functions can also be used with two arguments as :

```
seekg(offset, ref position);
seekp(offset, ref position);
```

Offset represents number of bytes file pointer is to be moved from the location specified by ref position.

The ref position can be one of the three constants :

(i) ios::beg - start of file

(ii) ios::cur - current position of pointer

(iii)i os::end - end-of-file.

---

Q. 94 Explain the purpose of following functions with example :
i) put( ) ii) get( ) iii) read( ) iv) write( )

Ans. :

(i) put( ) : (July 2016)

This function is used to store a single character into file, specified by object of ofstream.

It has the form :

```
ofstream object.put(character variable);
```

e.g.

```
ofstream outf;
outf.open("xyz");
char c='A';
outf.put(c);
```

(ii) get( ) : (July 2016)

This function is used to read a character from a file specified by ifstream object.

It has the form

```
ifstream object.get(character variable);
```

e.g.

```
char c;
ifstream inf("xyz");
while(inf.eof( )==0)
//It will read characters from file till it reaches end of file i.e. eof()
{
inf.get(c);
        cout<<c;
}
```

(iii) write( ) :

It has the form

```
write((char*) & variable, sizeof (variable));
```

This function is used with object of ofstream and it is used store data into file in binary mode i.e. general user cannot read data of file by using 'type' command. Here all the variables of different type are first of all converted into (char*) i.e. pointer to character and second parameter is number of bytes required to store given variable.

e.g.

```
struct s
{
char n[20];
int t;
};
main( )
{
struct s m;
ofstream outf("xyz");
cin>>m.n;
cin>>m.t;
outf.write((char*) & m, sizeof (m));
}
```

(iv) read( ) :

It has the form

```
read((char*)& variable,sizeof (variable));
```

Data which is stored by using write( ) can be read by using read( ). This function is used to read data in binary modefrom file. This function is associated with ifstream object.

Generally read( ) and write( ) functions are used with structures or objects to store the records. These functions has two arguments. First is address of variable of any data type. But, data is converted into pointer to characters and the second parameter is size of that variable.

e.g.

```
struct s
{
char n[20];
int t;
};
        main( )
{
ifstream inf("xyz");
struct s m;
while (inf.eof( )==0)
{
        read((char*) & m, sizeof (m));
        cout<<"\n"<<m.n;
        cout<<"\n"<<m.t;
}
```

Q. 95 Write a program in C++ to read the name of country from one text file and name of corresponding capital from another text file. The program must display the country name and corresponding capitals name in the output.

Ans. :

```
//Working with multiple files
#include<fstream.h>
#include<conio.h>
void main( )
{
        clrscr( );
        ofstream fout;
        fout.open("country");
        fout<<"United States of America \n";
        fout<<"United Kingdom \n";
        fout<<"South Korea \n";
        fout.close( );
        fout.open("capital");
        fout<<"Washington \n";
        fout<<"London \n";
        fout<<"Seoul \n";
        fout.close( );
        const int n=80;
        char line[n];
        ifstream fin;
```

```
        fin.open("country");
        cout<<"Contents of country file: \n";
        while(fin.eof()==0))
        {
        fin.getline(line, n);
        cout<<line;
        }
        fin.close( );
        fin.open("capital");
        cout<<"\n Contents of capital file:\n";
        while (fin) //or equivalent to while (fin.eof)()==0))
        {
        fin.getline(line, n);
                cout<<"line";
        }
        fin.close( );
    }
```

The output of above program will be

Contents of country file :

United States of America
United Kingdom
South Korea

Contents of capital file :

Washington
London
Seoul

Q. 96 Write a program to store records of 10 students into file student.dat. Declare class student with member variables name, roll number, marks of three subjects and total marks. By using object of this class and read( ), write( ) functions store records in file student.dat and display those records whose total is greater than 250.

(Note : It is not necessary to use array of objects.)

Ans. :

```
//C++ program that working with files
        #include<conio.h>
        #include<iostream.h>
        #include<fstream.h>
        class student
        {
        public:
                char name[30];
                int roll_no, m1, m2, m3, total;
```

```
        void getdata (void);
};
void student::getdata (void)
{
cout<<"\n Enter name of student:";
cin>>name;
cout<<"\n Enter roll number:";
cin>>roll_no;
cout<<"\n Enter marks in three subjects:";
cin>>m1>>m2>>m3;
total = m1 + m2 + m3;
}
void main( )
{
        ofstream outf;
        student S;
        int i;
        outf.open("student.dat");
        for (i=0; i<=9; i++)
        {
        S.getdata( );
        outf.write((char*)&S, sizeof (S));
        }
        outf.close( );
        ifstream inf;
        inf.open("student.dat");
        while (inf)
        {
        inf.read((char*)&S, sizeof (S));
        if (total>250)
        {
        cout<<"Name:"<<name<<endl;
        cout<<"Roll number:"<<roll-no<<endl;
        cout<<"M1="<<m1<<"\t"<<"M2="<<m2;
        cout<<"\t M3="<<m3;
        cout<<"\n Total="<<total<<endl;
        }
        }
        inf.close( );
}
```

Q. 97 Write a program in C++ to store records of 10 students in file student.dat using object of class student. Each record contains name, roll number and total marks. Modify the record, if total is less than 50.

Ans. : //C++ program using files that modify the record

```
#include<iostream.h>
#include<fstream.h>
class stud
{
public:
        char name[30];
        int roll_no, total;
        void getdata (void);
};
void stud::getdata (void)
{
cout<<"\n Enter students name:";
cin>>name;
cout<<"\n Enter roll number:";
cin>>roll_no;
cout<<"\n Enter total";
cin>>total;
}
void main( )
{
        stud S;
        ofstream outf;
        outf.open("student.dat");
        for(int i=0; i<=9; i++)
        {
        S.getdata( );
        outf.write ((char*)&S, sizeof(S));
        }
        outf.close( ),
        ifstream inf;
        inf.open("student.dat", ios::in | ios::out);
        while (inf)
        {
        inf.read((char*)&S, sizeof (S));
        if (S.total<50)
{
        S.getdata( );
```

```
        inf.seekg(-(size of (S)), ios::cur);
        inf.write((char*)&S, sizeof(S));
    }
        inf.close( );
    }
```

Q. 98 Write a program in C++ that accepts a string. Store all the characters of string in file "Text", using put(). Use get() to read all the characters of string one by one and display them.

Ans. :

```
//C++ program to read a file and display its contents
#include<fstream.h>
#include<iostream.h>
#include<string.h>
void main( )
{
char string[80];
cout<<"\n Enter a string:"<<endl;
cin>>string;
int len;
len=strlen(string);
fstream file;          //input and output stream
file.open("Text", ios::in | ios::out);
for (int i=0; i<=len; i++)
{
file.put(string [i]);  //put a character to file
}
file.seekg(0);         //go to the start
char ch;
while (file)
{
file.get(ch);          //get a character from file
cout<<ch;              //display it on screen
}
}
```

Q. 99 Write a declaration for each of the following :

(a) An array a of six doubles. (b) An array a of six pointers to double.

(c) A pointer a to an array of 6 doubles. (March 2002)

Ans. :

(a) double a[6];

(b) double *a[6];

(c) double b[6], *a; a=&b[o]; (or a=&b;)

Q. 100 Write declaration for each of the following in C++.
i) An array of 8 floats.
ii) A pointer to an array of 8 doubles.
iii) Function that return a pointer to float. (Oct. 2007, 3)

Ans : i) float rev [8];
ii) double data [8], i.e. data [8], *a * a ; a = & data [0]; (or a = & data)
iii) float * fun ( );

Q. 101 Write a declaration for each of the following :
(i) A pointer to an array of 8 floats.
(ii) A function that returns a float
(iii) An array of 8 pointers to float (Oct. 2008, 3)

Ans. i) float a[8], *ptr; ptr=&a[0]; ii) float fun (); iii) float *a[8];

Q. 102 Write declarations of the following : (March 2014, 3)
(i) An array of 8 (ii) An array of 8 pointers to floats
(iii) Prototype of function that returns pointer to float (no parameters).

Ans. : (i) float a[8] ; (ii) float *a[8] ; (iii) float *f()

Q. 102(A) Write C++ declaration for the following. (March 2020, 3)
(i) Array of 10 integers. , (ii) Pointer to character variable
(iii) Object of the class

Ans. : (i) int a[10]; (ii) char* p; (iii) test t;

Q. 103 Write a C++ program that right justifies text. It should read and echo a sequence of left justified lines and print them in right justified format.
(March 05, 06, 07 ; Oct. 03, 06

Ans. : //C++ program that right justifies text.

```
#include <iostream.h>
int main ( )
{
    const int SIZE = 100; // max. no. of lines stored
    string line [SIZE], S;
    int n = 0, len, maxlen = 0;
    while (! cin.eof ( ))
    {
        getline (cin, S);
        len = S.length ( );
        if (len > 0)
            cout << S << endl;
        if (len > maxlen)
            maxlen = len;
        line [n++] = S;
    }
```

```
        - - n;          // n = number of lines read
        for (int i = 0; i < n; i++)
        {
                S = line [i];
                len = S.length ( );
                cout << string (maxlen - len, ' ' ) << S << endl;
        }
}
```

Q. 104 Implement a circle class. Each object of this class will represent a circle, accepting its radius value as float. Include an area ( ) function which will calculate the area of circle. (Oct. 2003, 05 ; March 2015)

Ans. :

```
//C++ program to implement a circle class
#include <iostream.h>
class circle
{
        float a;
        float r;
    public :
        void area (void);
};
void circle : : area (void)
{
    cout << "Enter radius of circle";
    cin >> r;
    a = 3.142 *r*r;
    cout << "The area of a circle is";
    cout << a;
}
void main ( )
{
    circle C;
    C.area ( );
}
```

Q. 105 Write a function that uses pointer to search for the address of a given integer of given array. If the given integer is found, the function returns its address, otherwise it returns NULL. (October 2003, March 2005)

Ans. :

```
#include <iostream.h>
    int *location (int a[ ], int n, int target)
    {
        for (int i = 0; i < n; i++)
```

```
                if (a [i] == target)
                        return (& a[i]);
                return NULL;
}
int main ( )
{
int a[8] = {1, 3, 7, 9, 10, 12, 17, 25};
int *p, n;
do
        {
                cout <<"Enter the number";
                cin >> n;
                if (p = location (a, 8, n))
                        cout <<"The number" << n <<
                                "is found at location" << p ;
                else
                        cout << "The number" << n << "is not found" << endl;
        }
while (n > 0);
}
```

Q. 106 Write a C++ program to accept a number and test whether it is prime or not. (October 2003, 08; March 2007, 18, 22

Ans. : //C++ program to test whether the inputted number is prime or not

```
#include <iostream.h>
void main ( )
{
        int prime, C = 0;
        cout << "Enter the number";
        cin >> prime;
        for (int i = 2; i < prime; i++)
{
        if (prime % i == 0)
                C = 1;
}
if (C == 0)
        cout << "The number" << prime
                << "is prime number";
else
        cout << "The number" << prime
                << "is not a prime number";
}
```

Q.107 Write a C++ program to replace every space in an inputed string (less than 80 characters) with a hyphen (i.e. -) (Mar. - 2004,07; Oct.2012)

Ans. : //C++ program to replace every space in string with a hyphen.

```
#include <iostream.h>
#include <string.h>
int main ( )
{       int len;
        char str [80];
        cout << "Enter a string";
        cin.get (str, 80);
        len = strlen (str);
        for (int i = 0; i < len; i++)
{       if (str [i] == ' ')
        str [i] = ' _ ';
}
cout << "The final string is";
cout << str ;
}
```

Q. 108 Write an object oriented program in C++ to read an integer number and find the sum of digits of integer [Hint : input 125 output 8 i.e. 1 + 2 + 5 = 81] (March 2018)

Ans. :

```
#include<iostream.h>
#include<conio.h>
void main()
{
int val, num, sum = 0;
cout « "Enter the number : ";
cin » val;
num = val;
while (num ! = 0)
{
    sum = sum + num % 10;
    num = num / 10;
}
    cout « "The sum of the digits of" << val << "is" << sum;
}
```

Q. 109 Write a C++ program to accept a sentence and print sentence using pointer. (July 2018)

Ans. :

```
#include<iostream.h>
#include<conio.h>
```

```
void main()
{
    char a[80],i,*ptr;
    clrscr();
    cout<<"Enter Sentence:";
    cin.gentline(a,80);
    ptr=&a[0];
    cout<<"sentence is:";
    for(i=0;a[i]!='\0';i++)
    {
        cout<<*ptr;
        ptr++;
    }
    getch();
}
```

Q.110 Write the output of the following C++ program : (Mar. 2004)

Ans. :

```
#include <iostream.h>
long comb (int n, int k);
int main ( )
{
    const int m = 5;
for (int i = 0; i < m; i++)
  {
    for (int j = 1; j < m - i; j++)
        cout << setw (2) <<" ";
    for (int j = 0; j <= i ; j++)
        cout << setw (4) << comb (i, j);
    cout << endl;
  }
}
long comb (int n, int k)
{
    if (n < 0 || k < 0 || k > n) return 0;
    long c = 1;
    for (int i = 1; i <=k; i++, n - - )
        c = c * n/i;
    return c;
}
```

Ans. : Output of the program is as follows :

```
                1
            1       1
        1       2       1
    1       3       3       1
1       4       6       4       1
```

Q.111 Write a C++ Program to exchange the contents of two variables using call by reference. (Oct.04)

Ans. :
```
// Program to exchange the contents of two variables using call by // reference
# include <iostream.h>
void swap (int *, int *); // function prototype
void main ( )
{
int a, b;
cout << "Enter the values";
cin >> a >> b;
cout << "Before Swapping";
cout << "a =" << a;
cout << "b=" << b;
swap (& a, & b);        // call by reference
cout << "After Swapping";
cout << "a =" << a;
cout << "b =" << b;
}
void swap (int *a, int *b)     // function definition
{
int temp;
temp = *a ;    // assign the value at address a to temp
*a = *b;       // put the value at b into a
*b = temp;     // put the value at temp into b
}
```

Q.112 Write a program in C++ to read a set of numbers from keyboard and find out the largest number in the given array. (Oct. 2004)

Ans. :
```
// Program to find out largest number from the given array
# include <iostream.h>
void main ( )
{
    int num [ 10 ], max ;
    cout << "Enter the number";
    for (int i = 0; i < 10; i++)
```

```
        cin >> num [ i ];
        max = num [ 0 ];
        for (int j = 1; j < 10; j++)
{
        if (max < num [ j ])
        max = num [ j ];
}
        cout << "The largest number in the array is" << max;
}
```

Q.113 Write a program in C++ to find Greatest Common Divisor (GCD) of two natural numbers. (Oct. 2004, March 2006, 2011, July 2017

Ans. : // To find Greatest Common Divisor of two natural numbers.

```
# include <iostream.h>
void main ( )
{
int n1, n2;
cout << "Enter the two natural numbers";
cin >> n1 >> n2;
while (n1 ! = n2)
{
if (n1 > n2)
n1 = n1 - n2;
if (n2 > n1)
n2 = n2 - n1;
}
cout << "The GCD is :" << n1;
}
```

Q.114 Write a program in C++ that inputs and stores 10 numbers in an array and prints the sum and average of the array elements. (Mar.2005, 09, 20 ; July 2019

Ans. : // Program to print the sum and average of the array elements.

```
# include <iostream.h>
void main ( )
{
int num [ 10 ], sum ;
float avg = 0.0;
cout <<"Enter the 10 elements";
for (int i = 1; i < = 10; i++)
cin >> num [ i ];
sum = 0;
for (i = 1; i < = 10; i ++)
```

```
{
sum = sum + num [ i ];
}
avg = sum / 10;
cout <<"The sum of numbers is :" <<sum << endl;
cout << "The average of the array element is : "<< avg;
}
```

Q.115 Write a C++ program to find the smallest of four given integers using min ( ) function to return the smallest of four given integers. int min (int, int, int, int) (March 2005,13, 18,19)

Ans. : //Program to find the smallest of four given integers using function.

```
# include <iostream.h>
void main ( )
{
    int a, b, c, d, small ;
    int min (int, int, int, int); //Prototype
    cout << "Enter the four numbers :" <<endl;
    cin >> a >> b >> c >> d;
    small = min (a, b, c, d); //function call
    cout << "The smallest number is :" <<small;
}
// function definition
int min (int n1, int n2, int n3, int n4)
{
    int low;
    if (n1 < n2)
    low = n1;
    else
    low = n2;
    if (n3 < low)
    low = n3;
    if (n4 < low)
    low = n4;
    return (low);
}
```

Q.116 Write a C++ program to accept the string from the user and reverse a string. (Oct.2005,2007)

Ans. : //Reverse a string

```
#include <iostream.h>
void main ( )
```

```
{
    char str [80], reverse [80];
    int len, i, k;
    cout << "Enter a string :" <<endl;
    cin>>str;
    for (i = 0; str [i]! = '\0'; i++);
    len = i;
    len - -;
    for (k = 0; len >=0; len - -, k++)
    reverse [k] = str [len];
    reverse [k] = '\0';
    cout <<"The reverse string is :";
    cout <<reverse;
}
```

Q.117 Implement a class temperature to convert degree Fahrenheit Value to degree Celsius Value. [Hint : C/5 = F - 32/9 , where C is temperature in degree Celsius and F is temperature in Fahrenheit degree] (March 2005, 2006, 2009

Ans. : //C++ program to convert degree.Fahrenheit value to degree celsius value

```
#include <iostream.h>
class temperature
{
    float tcel;
    float tfht;
    public:
    void getdata ( );
    void display ( );
};
void temperature : : getdata (void)
{
    cout <<"Enter the degree Fahrenheit";
    cin >> tfht;
}
    void temperature : : display (void)
{
    tcel = 5/9 * (tfht - 32);
    cout << "The degree celsius:";
    cout << tcel;
```

```
}
    void main ( )
{
    temperature t1;
    t1.getdata ( );
    t1.display ( );
}
```

Q.118 Write a program in C++ to read a set of numbers from the keyboard and to find out the largest number in the given array. (The numbers are stored in a random order.) (Oct.2006)

Ans. :

```
#include<iostream.h>
    void main (void)
        {
            int a[100];
            int i, n, larg;
            cout <<"How many numbers are in the array ?" << endl;
            cin>>n;
            cout <<."Enter the elements"<<endl;
            for (i = 0; i <= n - 1; ++i)
            {
                cin>>a[i];
            }
            cout << "contents of the array" << endl;
            for (i =0; i <= n - 1; ++i)
            {
                cout << a [i] << '\t';
            }
            cout << endl;
            larg = a[0];
            for (i = 1; i <= n - 1; ++i)
            {
                if (larg < a[i])
                larg = a[i];
            }
            cout << "Largest value in the array =" << larg;
        }
```

Q.119 Write a function that uses pointers to copy an array of double. (Oct.2006)

Ans. :
```
double* copy (double a[], int n)
{
    double* p = new double [n];
    for (int i = 0; i < n; i++)
        p [i] = a[i];
    return p;
}
void print (double [], int);
int main ()
    {
        double a[8] = {22.2, 33.3, 44.4, 55.5, 66.6, 77.7, 88.8, 99.9};
        print (a, 8);
        double*b = copy (a, 8);
        print (a, 8);
        print (b, 8);
    }
```

Q.120 Write a function that has passed an array of n pointers to floats and returns a pointer to the maximum of a floats. (Oct.2006)

Ans. :
```
float* max (float* p[ ], int n)
    {
    float* pmax = p[0];
    for (int i = 1; i<n; i++)
    { if (*p[i] > *pmax) pmax = p[i]; }
    return pmax;
    }
    float * max (float *P [ ], int n);
    main ()
    {
    float a[8] = {44.4, 77.7, 22.2, 88.8, 66.6, 33.3, 99.9, 55.5};
    float* p[8];
    for (int i = 0; i<8; i++)
    p[i]=&a[i]; //p[i] points to a[i]
    float* M = max (p, 8);
    cout<<M<<","<<*M<<endl;
    }
```

Q.121 Write a C++ program with ComputeTriangle ( ) function that returns the area a and perimeter p of a traingle with given side lengths x, y and z. (Oct.2007, July 2016)

(void ComputeTriangle (float & a, float & p, float x, float y, float z).

Ans :
```
# include <iostream. h>
# include <math . h>
```

```
void computerTriangle (float & a, float & p, float X, float y, float z) ;
void main ()
{
float a, p, x, y, z;
cout <<"Enter three side values of triangle";
cin>>x>>y>>z;
computerTriangle (a, p, x, y, z);
cout<<"The area with three side values are "<<x<<end1<<y<<endl<<z<<"is"
      <<a<<" and its perimeter is "<<p<<end1;
getch( );
}
void ComputeTriangle (float & a, float & p, float X, float y, float z)
{
p = x + y + z;
double s = p / 2.0;
a = sqrt (s * (s - x) * (s - y) * (s - z) );
}.
```

Q.122 Write a C++ program with ComputeCircle ( ) function that returns the area a and circumference c of a circle with given radius r. (Oct.2007) (July 2016)
void ComputerCircle (float & a, float & c, float r).

OR

Void Circle (float & a, float & c, float & r) (Oct. 2010)

(Note : Give function name as per hint)

Ans :

```
# include <iostream. h>
# include <conio. h>
void ComputeCircle (float & a, float & c, float r);
void main ( )
{
clrscr ( ) ;
float r, a, c ;
cout << " Enter the value for radius r";
cin >> r;
ComputeCircle (a, c, r); cout << " The area with radius "
<< r << "is"<< a <" and its ciramference is " << c << endl ;
getch ( );
}
void ComputeCircle (float & a, float & c, float r)
{
a = 3.14 * r * r;
c = 3.14 * 2 * r;
}
```

Q.123 Write C++ program to print the input string in a reverse order using function, which first locates the end of string. Then it swaps the first character with the last character, the second character with the second last character and so on. (Oct.2007; March 2019

Ans :
```
//c++ program to reverse the string
# include <iostream.h>
# include <stdio.h>
# include <string.h>
void reverse (char str[ ], int);
void main ( )
{
        char str [80];
        cout << "Enter the string";
        gets (str);
        int len = strlen (str);
                reverse (str,len);
}
void reverse (char str1[ ], int l)
{
        int mid = l/2;
        for (int i = 1; i < = mid ; i + +)
        {
                char temp = str1 [i];
                str1 [i] = str1 [l];
                str1 [l] = temp;
                l– –;
        }
        cout << "Reverse of string is :";
        puts (str1);
}
```

Q.124 Write a C+ + program with ComputeSphere ( ) function that returns the volume V and the Surface area S of a sphere with given radius r.

void ComputeSphere (float & S, float& V, float r) (Mar.2013, 20; Oct.2007, 2008

Ans.

```
#include<iostream.h>
#include<conio.h>
void ComputeSphere (float & s, float & v, float r);
void main ( )
{
float s, v, r;
cout<<"Enter the radius";
cin >> r;
ComputeSphere(s, v, r);
```

```
cout<<"The area with radius "<<r<<" is "<<s
      <<"and its volume is "<<v<<endl;
getch( );
}
void ComputeSphere (float & s, float & v, float r)
{
const float PI = 3.142;
s = 4.0 * PI *r *r;
v = s * r /3.0;
}
```

Q.125 Write a C++ program to read 5 elements of int array in reserve order and print the array (i.e. Read A [5] first and while printing , print A[0] first). (Oct.2008)

Ans.

```
#include<iostream.h>
#include<conio.h>
void main ( )
{
const int SIZE = 5 ;
double a [SIZE];
cout<<"Enter "<<SIZE <<"numbers : " <<endl;
for ( int i = SIZE - 1; i > = 0 ; i -- )
{
cout << "\t a ["<< i <<"] = ";
cin>> a[i];
}
cout<<"In reverse order they are :"<<endl;
for ( i = 0 ; i < SIZE ; i ++)
cout << " \t a["<<i>>"] = "<< a[i]<<endl;
getch ( );
}
```

Q.126 Write a C++ program with average () function that returns the average of four input numbers. (Oct.2008, July 2016)
(double ave (double x1, double x2, double x3, double x4);)

Ans.

```
#include<iostream.h>
#include<conio.h>
double ave (double, double, double, double);
void main ( )
{     double a, b, c, d;
      cout << "Enter four numbers : ";
      cin >> a >> b >> c >> d;
```

```
        cout <<"The average of all four is : "
                << ave (a, b, c, d) << endl;
        getch ( );
}
double ave (double x1, double x2, double x3, double x4)
{
        double sum = x1 + x2 + x3 + x4;
        return (sum / 4.0);
}
```

Q.127 Write an object oriented program to implement a class convert to convert degree Centigrade values to Fahrenheit degree. (Mar.2009)

(Hint C= 5/9 (F-32), where C is temperature in degree Celcius and F is temperature in Fahrenheit degree )

Ans :

```
//c++ program to convert degree celsius value to degree Fahrenheit
# include <iostream.h>
class temperature
{
        float tcel;
        float tfht;
        public :
                void getdata ( );
                void display ( );
};
void temperature : : getdata (void)
{
        cout <<"Enter the degree celcius:";
        cin >> tcel;
}
void temperature : : display ( )
{
        tfht = 9/5 * (tcel + 32);
        cout << "The Fahrenheit temperature :";
        cout << tfht;
}
void main ( )
{       temperature t;
        t.getdata ( );
        t. display ( ); }
```

Q. 128 Write a program in C++ to find sum of First 100 Natural Numbers. (October. 2009)

Ans :

```
#include <iostream.h
void main( )
      {int d, sum = 0;
      d = 1;
      while (d<=100)
            {sum = sum + d;
            d = d + 1;
            }
      cout <<"sum" <<sum;
}
```

Q. 129 Write a C++ program to input a word (max. length 15 characters) from user and print each of its characters on new line in Reverse Order. (October. 09, March 14, July 17)

Ans :

```
#include <iostream.h>
# include < string.h>
      void main ( )
{
      char x [16] ; int i ;
      cout<<"Enter a word" l;
      cin>>x;
      l = strlen (x);
      for (i = l ; i> = 0 ; i --)
{
      cout <<x [i] <<endl;
      }
}
```

Q. 130 Write a program in C++ using OOP technique to find AREA of Circle. (October 2009)

Ans :

```
#include <iostream.h>
class circle
{
      private;
      int r ;
      float A ;
      public:
void getradius()
{
      cin>>r;
```

```
}
        void print( )
{
        A = 3.14 * r * r;
cout<< "Area of circle is"<<A;
}
        };
        void main( )
{
        circle c;
        c.getradius( );
        c.print( );
}
```

Q. 131 Write a C++ Program by using swap function to interchange given two numbers void swap (int & x, int & y) ; (Mar.2010)

Ans : //C++ program for interchange values.

```
# include <iostream.h>
void swap (int & x, int & y);
void main ( )
{
        int a, b ;
        cout << "Enter values for a and b";
        cin >> a >> b;
        swap (a, b);
        cout << "After swapping" <<end l;
        cout << "a = " << a;
        cout << "b = " << b;
}
void swap (int & x, int & y)
{
        int temp = x;
        x = y;
        y = temp;
}
```

Q. 132 Write a C++ Program to count occurrence of a character 'J' in a given string. (Mar. 2010 ; July 2019

Ans : //C++ program to count occurrence of character 'J'

```
# include <iostream.h>
# include <string.h>
void main ( )
{
```

```
char string [80];
int len, count, i;
count = 0;
cout <<"Enter a string : "<<endl;
gets (strlen);
len = strelen (string);
for (i=0; i<len; i ++)
{
if (string [i] = = 'J')
    count ++;
}
cout <<"Occurence of character 'J' in given string is :" <<count;
}
```

Q. 133 Show output of the following C++ Program : (Mar. 2010)

```
class test
{
    private :
    int i, j;
    public :
    void calculate ( )
    {
        for (i = 1; i < = 5; i ++)
        {
            for (j = 1; j< = i; j++)
            { cout <<j % 2 << "\t";
            }
            cout << endl;
            }
    }
};
void main ( )
{
    test a ;
    a. calculate ( );
}
```

Ans : output of given C+ + program is as fallows ;

```
1
1   0
1   0   1
1   0   1   0
1   0   1   0   1
```

Q. 134 Write a C++ program to sort 10 integer numbers in ascending order. (Oct.2010)

Ans :

```
void main ()
{ int n [10], i, j, temp ;
cout << " Enter 10 integer" <<endl;
for ( i = 0; i < = 9; i++)
{ cin>>n[i] ;
}
for (i = 0; i < = 9; i ++)
{
for (j = i; j < = 9; j++)
{ if (n [i] > n[j])
{ temp = n[j];
n[j] = n[i];
    n[j] = temp ;
     }
  }
}
cout << " sorted list is" << endl ;
for (i =0; i < =9, i ++)
{ cout << n [i] << endl ;
  }
}
```

Q. 135 Write a C++ program with a function to find whether the year entered is a Leap year or not. 5

Ans :

```
# include <iostream.h>
void main ()
{ int leap (int) ;
int n ;
cout << " Enter year" << endl ;
cin >> n ;
cout << leap (n)
}
int leap (int n)
{
if (n% 4 == 0 && n% 100!= 0 | | n% 400 ==0)
cout << " Year is Leap Year";
else
cout<<< " Year is not Leap Year";
return n;
}
```

**Q. 136** Write an Object Oriented Program in C++ to read a set of 10 numbers and store it as an one dimensional array; again read a number 'd' and check whether the number 'd' is present in the array, if it is so, print out how many times the number 'd' is repeated in the array.

Ans:

```
# include <iostream. h>
class search
{
private :
        int num [10] ;
public :
        void getdata( ) ;
        void find ( ) ;
} ;
void search :: getdata ()
{
        cout << " Enter elements";
        for (int i =0 ; i < 10 ; i ++)
           cin >> num [i];
int n ;
cout < < " Enter number ";
cin >> n;
}
void search : : find ()
{ int count = 0 ;
        for ( i = 0 ; i < = 9 ; i ++)
{
        if ( num [i] == n)
        count ++ ;
}
count << " The number " << n << " is
        present in array " << count << " times" ;
}
void main ( )
{
search s ;
s. getdata ( ) ;
s. find ( ) ;
}
```

Q. 137 Write an Object Oriented Program in C++ to read an integer number and find out the sum of all the digits of a number. 5

(For eg. n = 1256, SUM = 1 + 2 + 5 + 6 =14)

Ans:
```
// C++ program to read an integer number & find out the sum of all the digits
# include <iostream.h>
class digit
{
    private :
        int num ;
    public :
        void getdata ( ) :
        void sum ( ) ;
};
void digit : : getdata ( )
{
    cout << " Enter the number" ;
    cin >> num ;
}
void digit : : sum ( )
{
    int rem, add = 0;
    while ( num > 0)
{
    rem = num % 10;
    add = add + rem;
    num = num /10;
}
    cout << " The sum of all digit of number= " << add ;
}
void main ( )
{
    digit S ;
S.getdata ( ) ;
S.sum ( ) ;
}
```

Q. 138 Write a program in C++ to read a set of numbers from keyboard and find out largest number in the given array using pointers. (Oct. 2011, 5 Marks)

Ans.: find largest number in given array using pointers,

```
void main ( )
{
```

```
int a [100], b, *ptr, d, Max = 0;
cout <<"Enter ten numbers"<<endl;
for (d = 0; d <= 9; d++)
    {
        Cin >> a [d];
    }
    ptr = a; // ptr = &a [0]
for (d = 0; d <= 9; d++)
{
If (* ptr > max)
    {
        Max = * ptr;
    }
}
cout <<"largest Number="<<max;
}
```

Q. 139 Write a program in C++ to convert the given binary number into decimal equivalent using method convert ( ). (Oct. 2011, 5 Marks)

Ans. :

```
class binary
{
    private :
        int bin;
    public :
        void getdata ( )
        {
            cout <<"Enter binary number"<<endl;
            cin >> bin;
        }
        void convert ( )
        {
            int digit, dec = 0, i = 0;
            do
            {
                digit = bin%10;
                dec = dec + digit*pow (2,'i);
                i++;
                bin = bin/10;
            } while (bin! = 0);
        cout << "Decimal equivalent="<<dec;
```

```
    }
};
void main ( )
{
binary a;
a.getdata ( );
a.convert ( );
}
```

Q. 140 Implement a class rectangle. Each object of this class, will :represent a rectangle accepting its length and width as float. Include an area function which will calculate the area of, the rectangle.

Ans. : (Oct. 2011, 5 Marks)

```
Class rectangle
{
    private :
        float len, width;
    public :
        void getdata ( );
        void area ( );
};
void rectangle : : getdata ( )
{
    Cout<<"Enter length and width"<<endl;
    Cin>>len>>width;
}
void rectangle : : area ( )
{
cout << "Area of rectangle =" <<len*width;
}
void main ( )
{
    rectangle r
    r.getdata ( );
    r.area ( );
}
```

Q. 141 Write a program in C1 to find GCD and. LCM of two inputted numbers using methods enter ( ) and compute ( ): (March 2012, 5 Marks)

Ans.: Find GCD and LCM of two input number,

```
Class lcmgcd
{
    private :
```

```
        int a, b, x,
    public :
        void enter ( )
        {
            cout<<"Enter two number";
            cin>>a>>b;
            x = a * b;
        }
        void compute ( );
};
void lcmgcd : : compute (void)
{
    while (a! = b)
    {
        if (a > b)
            a = a – b;
        if (b > a)
            b = b – a;
    }
    l = x/a;
    cout<<"GCD = "<<a;
    cout<<"LCM = "<<l;
}
void main ( )
    lcmgcd d;
    d.enter ( );
    d.compute( );
}
```

Q. 142 Implement a point class for two dimensional points (x, y). Include a function dist() to return points distance from origin (0, 0) and a display ( ) function to print result. (March 2012, 5 Marks)

Ans. :

```
class point
{
private :
    double x, y;
public :
    void getdata ( )
    {
        cin>>x>>y;
```

```
    }
    double dist ( )
    {
        return sqrt (x*x+y*y);
    }
    void display ( )
    {
        cout<<"Distance between"<<x<<"and"<<y<<dist ( );
    }
};
void main ( )
{
    point c;
    c.getdata ( );
    c.display ( );
}
```

Q. 143 Write an Object Oriented Program in C++ that prints the factorial of a given number. (Oct. 2012, 5 Marks)

Ans. :

Factorial of given no.

```
#include<iostream.h>
class fact
{   private:
        int n;
    public:
        void getaata();
        void display();
};
void fact : : getdata ()
{
    cout<<"Enter value of n\n";
    cin>>n;
}
void fact :: display( )
{
    for (int i=1,f = 1; i<=n; i++)
    {
        f = f * i;
    }
```

```
        cout <<"Factorial of number"<<n
             <<"=" <<f<<endl;
}
void main ()
{
        fact obj;
        obj. get data();
        obj. Display();
}
```

Q. 144 Write a C++ program to find largest of four given integers using max ( ) function to return largest of four given integers : int max (int, int, int, int) (March 2013, 5 Marks)

Ans. :

```
Void main( )
{
        int max(int, int, int, int);
        cout << "enter four integer";
        int w, x, y, z;
        cin >> w >> x >> y >> z;
        cout << "minimum =" <<max (w, x, y, z);
}
int max (int n1, int n2, int n3, int n4)
{
        int max = n1;
        if (n2 > max) max = n2;
        if (n3 > max) max = n3;
        if (n4 > max) max = n4;
        return max;
}
```

Q. 145 Write a program in C++ that inputs and stores 10 numbers in any array and prints numbers in reverse order. (March 2013, 5 Marks)

Ans. :

```
void main( )
{
        double a[10];
        cout <<"enter number";
        for (int i = 0 ; i= 10; i++)
        {
             count <<"\t a ["<< i <<"]";
             cin>>a[i];
        }
        cont <<"The array in reverse order is\n";
```

```
    for (int j = 9 ; j > 0 ; j –)
    {
        Cout <<"\t a["<<j<<"]";
        Cout << a[j] << endl;
    }
}
```

Q. 146 Write a C++ Program to find factorial of 1 of 5 numbers.

(Oct. 2013, March 2022, 5 Marks

Ans. :

```
#include <iostream.h>
void main ( )
{
    int fact, i, j;
    fact = 1;
    cout << "Number " << "\t" << "Factorial";
        for (j = 1; j< = 5; j++)
        {
            fact = fact * j;
        }
        cout << i << "\t" << fact <<endl;
    }
```

Q. 147 Write a program in C+ to find sum of contents of an array with the help of a pointer. (March 2014, 5 Marks

Ans. :

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

Q. 148 Implement a class circle. Include a constructor in it which accepts value of radius from user. Include two more functions in it, one of which calculates area and circumference and the other prints answers (March 2014, 5 Marks

Ans. :

```
#include<iostream.h>
class circle
```

```
{
    private :
        int r ;
        float a, c;
    public :
        circle()
        {
            cout<<"Enter Radius";
            Cin>>r;
        }
        void cal()
        {
            a = 3.14 *r*r;
            c = 2*3.14*r;
        }
        void print()
        {
        cout<<a<<c;
        }
        ~ circle(){}
    };
void main()
{
    circle obj;
    obj.cal();
    obj.print();
}
```

Q. 149 Write a program in C++ display the following output using for loop : (Oct. 2015)

```
1
1  2
1  2  3
1  2  3  4
1  2  3  4  5
```

Ans. :

(a)

```
# include <isotream.h>
void main ( )
{
    int i, j;
    for (i =1; i < = 5; i++)
    {
        for (j = 1; j < = i; j++)
        {
```

```
        for (j = 1; j <= i; j++)
        cout << j << " ";
        cout << endl;
        }
    }
```

Q. 150 Write a C++ program to find the GCD (Greatest Common Divisor) of two numbers entered by user. (March 2014, 5 Marks)

Ans. :

```
#include<iostream.h>
void main()
{
    int a, b,i;
    count <<"Enter Two Numbers";
    cin >>a>>b;
    int gcd =1, min;
    if (a>b)
    {
        min = a;
    }
    else
        min=b;
    for (i=1, i<=min; i++)
    {
    if (a%i==0&&b%i==0)
        gcd = i;
    }
        cout <<"GCD="<<gcd ;
}
```

Q. 151 Write a program in C++ to initialize the array of 10 integers and find the sum of all the elements of array. (Oct. 2014, 5 Marks)

Ans. :

```
    void main ( )
    {
        int a [10], i, sum;
        cout<< " Enter 10 Elements ";
            for ( i=0; i < 10; i++)
    {
        cin>> a [1];
    }
    sum = 0;
```

```
cout<<"/n The Array = ";
    for ( i=0; i<10; i++ )
{   cout<< a[i] << "  ";
    sum = sum + a[i];
}
cout<<"\n sum of all element = "  << sum;
    getch ( );
}
```

Q. 152 Write a program in C++ to accept a line from keyboard and count total no. of blank spaces in a line. The program should print the original string and blank spaces. (Oct. 2014, 5 Marks)

OR Write a C++ program to count and print occurrence of the character 'M' in a given string of maximum 79 characters. (March 2022)

Ans. :

```
void main ( )
    {
        char a[80] ;
        int i, bsp;
        cout<< " Enter a sentence = ";
        gets (a);
        bsp = i = 0;
        while ( a[i]  ! = '\0' )
    {   if ( a[i] == ' ' )
                {       bsp++ ;
                }
    i++;
}
cout<<"/n Total Blank space = "  << bsp;
getch ( );
}
```

Q. 153 Write a program in C++ to accept three numbers from keyboard and find the smallest one and print it. (Oct. 2014, 5 Marks)

Ans. :

```
void main ( )
{   int a, b, c;
    cout<< " Enter three no. = ";
    cin >> a>>b>>c;
    int small = a;
    if ( b < small )
{   small = b;  }
if ( c < small )
```

```
{ small = c ; }
cout<<"/n smallest no = " << small;
getch ( );
}
```

Q. 154 Write a program in C++ to initialize the array to 10 floats and print all the array elements using pointer. (Oct. 2014, 5 Marks)

Ans. :

```
void main ( )
{
float a [10] = {.1.2, 2.3, 4.5, 5.6, 7.8, 8.9, 9.1, 6.7, 10.2, 10.4 }
        float *p;
    int i ;
p = & a [0];
        cout <<"\n Array=" ;
    for ( i=0; i<=9; i++ )
{   cout<< * p   << " " ;
    p++;
}
    getch ( );
}
```

Q. 155 Write a program in C++ to accept two integer values in main function, pass them to function great() using call by value and find greater number, function great() should not return any value. (March 2015, 5 Marks)

Ans. :

```
void great( int, int);
    void main( )
    {   int a, b;
        cout<<"Enter Two No=";
        cin >> a>>b;
        great(a,b);
        getch( );
    }
    void great( int p, int q)
    {   if( p>q)
        {   cout<<p<<"is big" ;}
            else
        {   cout<<q<<"is big"; }
    }
```

Q. 156 Write a program in C++ to accept three integers from keyboard and find greatest number with using condition control. (March 2015, 5 Marks)

Ans. :

```
void main( )
{     int a, b, c;
      cout<<"Enter three no. = ";
      cin >> a>>b>>c;
      int big = a;
      if ( b < big )
{     big = b; }
if ( c < big )
{ big = c ; }
cout<<"\n Greatest no = " << big;
getch( );
}
```

Q. 157 Write a program in C++ to accept a string from keyboard and copy string into another string without using the library function. (March 2015, 5 Marks)

Ans.

```
void main( )
      {     char a[80], b[80];
            int i;
            i = 0
          cout<< " Enter a String = ";
      cin>> a;
      while( a[i]! ='\0' )
      {     b[i] = a[i];
              i++;
      }
      b[i] = '\0' ;
      cout<<"\n copied string="<<b;
      getch( );
      }
```

Q. 158 Write a program in C++ to declare the array of 10 floats and find the largest. (Oct. 2015, 5 Marks, March 2017)

Ans. :

(b)

```
void main ( )
      {     float a[10];
            cout <<"Enter 10 float values" <<endl\
            for (int i = 0; i<=9; i++)
                  {
                        cin>>a[i];
```

```
        }
        int largest = a[0];
        for (i = 1; i<=9; i++)
        {
                if(a[i] > Largest)
                {
                        Largest = a[i];
                }
        }
    cout<<"Largest of all 10 values = " << Largest;
}
```

Q. 159 Write a program in C++ to accept a string from keyboard and find the length of the string without using library function. (Oct. 2015, 5 Marks)

Ans. :

```
void main ( )
{
    char S[80];
    cout<<"Enter a string" <<endl;
    cin >> getline (S, 80);
    int i = 0, L = 0;
    while (S[i] ! = '\0')
        {
            i++;
            L++;
        }
    cout<<"length of a string = " <<L;
}
```

Q. 160 Write a program in C++ to accept two integer number in main ( ) and find sum of those no using function of by address. (Oct. 2015, 5 Marks)

Ans. :

(b)

```
int Add (int *, int *);
void main ( )
        {       int  N1, N2;
                cout <<"Enter two No";
                cin>>N1>>N2;
                int sum = Add (&N1 & N2);
                cout << "Addition = "<<SUM;
        }
int Add (int * x, int * y)
        {
                int S = *x + *y
                return S;
        }
```

Q. 161 Implement a class temperature. Include a constructor in it which accepts value of temperature from user in degree Celsius. Include two functions in it, one of which calculates its equivalent temperature in degree Fahrenheit and other function prints the answer :

$$\left[\text{Formula} : \frac{C}{5} = \frac{F-32}{9}\right]$$

(March 2016, 5 Marks)

Ans. :

```
# include <iostream.h>
class temperature
    {   private :
        float f, t;
        public :
            temperature()
            {
                cout <<"Enter temp. in celcin";
                cin >> t;
            }
            void cal ( )
            {
                f = 9* c/5 + 32;
            }
            void print( )
            {
                cout <<"temp in farenheit"<< f
                ~ temperature( ) {  }
            };
            void main( )
                {   temperature ob;
                    ob . cal();
                    ob.print( );
                }
```

Q. 162 Implement a class average. Include a constructor in it which will accept value of three variables from user. Include two more functions in it, one functions calculates average and other prints it.

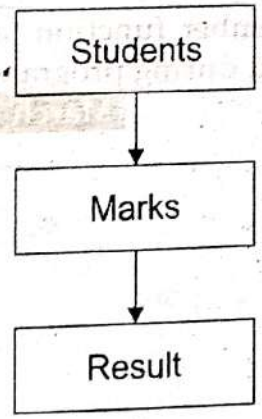
(March 2016, 5 Marks)

Ans. :

```
# include <iostream.h>
class average
    { private :
        float a, b, c, s;
      Public :
        average( )
```

```
        {
                cout <<"Enter 3 nos." ;
                cin >> a >> b >> c;
        }
    void cal()
        {
                s = (a +b + c)/3;
        }
    void print( )
        {
                cout <<"average=" << s;
        }
        ~ average( ) { }
        };
        void main( )
        { average ob;
          ob.cal( );
          ob.print( );
        }
```

Q. 163 Implement the above class hierarchy of inheritance. Class student accepts roll number of student, class marks accepts marks of three subjects and class result calculates the total and prints all details. (5)
(Create object of class result)




Ans. :

```
# include <iostream.h>
    class student
    { protected : int r ;
      public :
            void get1( )
               {cout <<"Enter roll no.";
               cin >>r; }
            };
            class marks : public student
```

```
{ protected : int m1, m2, m3;
  public :
  void get 2( )
  {
    cout <<"Enter marks of 3 sub";
    cin >> m1 >> m2 > m3;
  }
};
class result : public marks
{ protected : int tot ;
 public :
        void cal( )
          { tot = m1 + m2 + m3;
          }
        void print ( )
          {
          cout <<"roll no". << r;
          cout <<"total =" << tot; }
          };
void main( )
{       result ob;
        ob.get1( );
        ob.get2( );
        ob.cal( );
        ob.print( );
}
```

Q. 164 Impliment class GCD which have member function (a/c), which calculate greate common divisor of two number entered during program execution. Print() will Pri GCD of two number. (March 2017, July 2019, March 2020

Ans. :

```
#include<iostream.h>
#include<conio.h>
class GCD
{
private:
    int a, b;
public:
    void gcd()
    {
        cout<<"\n Enter Two numbers:";
        cin>>a>>b;
        while(a!=b)
        {
            if(a>b)
```

```
                    a=a-b;
                if(b>a)
                    b=b-a;
            }
        }
        void print()
        {
            cout<<"\n GCD="<<a;
        }
    };
    void main()
    {
    GCD g;
    g.gcd();
    g.print();
    }
```

Q. 165 Write an object oriented program in CH to read an integer number and find the sum of digits of integer [Hint : input 125 output 8 i.e. 1 + 2 + 5 = 81] (March 2018)

Ans. :

```
#include<iostream.h>
#include<conio.h>
void main()
{
    int val, num, sum = 0;
    cout « "Enter the number : ";
    cin » val;
    num = val;
    while (num ! = 0)
    {
        sum = sum + num % 10;
        num = num / 10;
    }
        cout « "The sum of the digits of" << val << "is" << sum;
}
```

Q. 166 Write a C++ program to accept a sentence and print sentence using pointer. (July 2018)

Ans. :

```
#include<iostream.h>
#include<conio.h>
void main()
{
    char a[80],i,*ptr;
```

```
        clrscr();
        cout<<"Enter Sentence:";
        cin.gentline(a,80);
        ptr=&a[0];
        cout<<"sentence is:";
        for(i=0;a[i]!='\0';i++)
        {
            cout<<*ptr;
            ptr++;
        }
        getch();
    }
```

Q. 167 Write a class based C++ program to print 20 terms of fihonacci series.
[Hint : Fibonacci series 0, 1, 1, 2, 3, 5,.....] (July 2018)

Ans. :

Method 1 :

```
#include<iostream.h>
#include<conio.h>
class fibonaaci
{
    private:
    long int f0,f1,fib;
    public:
    fibonacci(void);
    void process(void);
    void display(void);
    void display1(void);
};
fibonacci::fibonacci(void)
{
    f0=0;
    f1=1;
}
void fibonacci::display1(void)
{
    cout<<f0<<"\t"<<f1<<"\t";
}
void fibonacci::process(void)
{
    fib=f0+f1;
```

Method 2 :

```
#include<iostream.h>
#include<conio.h>
class fibonacci
{
        private;
        long inte f0,f1,fib;
        public:
        fibonacci(void);
        void process(void);
};
fibonacci::fibonacci(void)
{
        f0=0;
        f1=1;
}
void fibonacci::process(void)
{
        int i, n;
        cout<<"\n Enter number of
        elements"<<endl;
        cin>>n;
        cout<<f0<<"\t"<<f1<<"\t";
        for(i=3;i<n;i++)
        {
```

```
        f0=f1;
        f1=fib;
    }
    void fibonacci::display(void)
    {
        cout<<fib<<"\t";
    }
    void main()
    {
        int i, n;
        fibonacci f;
        cout<<"\n Enter number of elements"<<endl;
        cin>>n;
        f.display1();
        for(i=3;i<=n;i++)
        {
            f.process();
            f.display();
        }
}
```

```
            fib=f0+f1;
            cout<<fib<<"\t";
            f0=f1;
            f1=fib;
        }
}
void main()
{
        clrscr();
        fibonacci f;
        f.process();
        getch();
}
```

Q. 168 Write a C++ program to overload add ( ) function which will add two integer [add(int, int)] and three integers [add (int, int, int)]. (July 2018)

Ans. :

```
#include<iostream.h>
#include<conio.h>
int add(int,int);
int add(int,int,int);
void main()
{
    int a,b,p,q,r,a1,a2;
    cout<<"\nEnter two number";
    cin>>a>>b;
    a1=add(a,b);
    cout<<"\n Addition of two numbers="<<a1;
    cout<<"\nEnter three number";
    cin>>p>>q>>r;
    a2=add(p,q,r);
    cout<<"\n Addition of three numbers="<<a2;
    getch();
```

```
}
int add(int n1,int n2)
{
    return n1+n2;
}
int add(int m1,int m2,int m3)
{
    return m1+m2+m3;
}
```

Q. 169 Write a C++ program to accept an array of 10 integers and find smallest and largest element in array. July 2018

Ans. :

```
#include<iostream.h>
#include<conio.h>
void main()
{
    int a[10],i,j,temp;
    cout<<"Enter 10 integers";
    for(i=0;i<10;i++)
    {
        cin>>a[i];
    }
    for(i=0;i<10;i++)
    {
        for(j=0;j<=10-i;j++)
        {
            if(a[j]>a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
        }
    }
    cout<<"\nThe smallest number in array is="<<a[0];
    cout<<"\nThe smallest number in array is="<<a[9];
    getch();
}
```

Q. 170 Write a C++ program to accept a sentence (maximum 50 characters) and print sentence in reverse. **(March 2019)**

Ans. :

Method 1 :

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
void main()
{
    char S[50];
    int i, L;
    cout<<"Enter a setnece(50 chars)";
    cin.getline(S,50);
    L=strlen(S);
    cout<<"\n Sentence in reverse";
    for(i=L-1;i>=0;i--)
    {
        cout<<S[i];
    }
    getch();
}
```

Method : 2

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
void main()
{
    char S[50];
    int i, L;
    cout<<"Enter a sentence(50 chars)";
    cin.getline(S,50);
    cout<<"\n Sentence in reverse";
    strrev(S);
    cout<<S;
}
```

Method 3 : (Prefer This Method)

```
#include<iostream.h>
#include<conio.h>
{
    char S[50];
    int i, L;
    cout<<"Enter a sentence(50 chars)";
    cin.getline(S,50);
    for(i=1;S[i]!='\0';i++)
    {
        L++;
    }
    cout<<"\n Sentence in reverse";
    for(i=L;i>=0;i--)
    {
        cout<<S[i];
    }
}
```

Q. 171 Write a C++ program to find smallest in an array of 10 floats using pointer. (March 2019)

Ans. :

```
#include<iostream.h>
#include<conio.h>
void main()
{
    clrscr();
    float a[10],small,*p;
    int i;
    cout<<"\n Enter 10 numbers";
    for (i=0;i<10;i++)
    {
        cin>>a[i];
    }
    p=&a[0];
    small=*p;
    for(i=0;i<10;i++)
    {
        if(*p<small)
        {
            small=*p;
        }
        p++;
    }
    cout<<"\nsmallest element="<<small;
    getch();
}
```

Q. 172 Write a class based program in C++ to find area of a Triangle. (March 2019)

Ans. :

```
#include<iostream.h>
#include<conio.h>
class triangle
{
    private:
    float a,b,h;
    public:
    void area();
};
void triangle:: area()
```

```
{
    cout<<"Enter base and height"<<endl;
    cin>>b>>h;
    a=0.5*b*h;
    cout<<"\nArea of triangle="<<a;
}
void main()
{
    triangle t;
    t.area();
}
```

Q. 173 Write a class based C++ program to find the area of a sphere. (July 2019)

Ans. :

```
#include<iostream.h>
#include<conio.h>
class sphere
{
    float r, a;
    public:
    void area()
    {
            cout<<"Enter radius of sphere:";
            cin>>r;
            a=4*3.14*r*r;
            cout<<"area of sphere="<<a;
    }
};
void main()
{
    sphere s;
    s.area();
}
```

Q. 174 Write a C++ program to compute sum of even numbers stored in an array of 10 integers. (Dec. 2020)

Ans. :

```
#include<iostream.h>
#include<conio.h>
void main()
{
    clrscr();
    int a[10],i,sum=0;
```

```
        cout<<"Enter 10 elements of array :";
        for(i=0;i<10;i++)
        {
                cin>>a[i];
        }
        cout<<"\n Addition of even numbers: ";
        for(i=0;i<10;i++)
        {
                if(a[i]%2==0)
                {
                        cout<<a[i]<<"+";
                        sum=sum+a[i];
                }
        }
        cout<<"= "<<sum;
        getch();
}
```

Q. 175 Write a C++ program to interchange two entered number without using third variable. (Dec. 2020)

Ans. :

```
#include<iostream.h>
#include<conio.h>
void main()
{
        clrscr();
        int a,b;
        cout<<"Enter 2 numbers:";
        cin>>a>>b;
        cout<<"Values before swapping:a="<<a<<" b="<<b;
        b=b+a;
        a=b-a;
        b=b-a;
        cout<<"Values after swapping:a="<<a<<" b="<<b;
        getch();
}
```

Q. 176 Write a C++ program that accepts ten integer numbers. It count and display positive integer numbers. (Dec. 2020)

Ans. :

```
#include<iostream.h>
#include<conio.h>
```

```
void main()
{
    clrscr();
    int a[10],i;
    cout<<"Enter 10 elements of array :";
    for(i=0;i<10;i++)
    {
        cin>>a[i];
    }
    cout<<"\n positive numbers: ";
    for(i=0;i<10;i++)
    {
        if(a[i]>0)
        {
            cout<<a[i]<<" ";
        }
    }
    getch();
}
```

Q. 177 Write a C++ program to count and print occurrence of the character 'M' in a given string of maximum 79 characters. (Ch. 3/Q.136/Pg. No. 3-108)
(Hint : Here in the given program , array should be char arr[79]; and modify for loop accordingly) (March 2022)

Ans. :

```
# include <iostream. h>
class search
{
private :
char arr [80] ;
public :
void getdata( ) ;
void find ( ) ;
} ;
void search :: getdata ()
{
cout << " Enter string / sentence upto 79 characters";
for (int i =0 ; i < 80 ; i ++)
cin >> arr [i];

}
void search : : find ()
```

```
{
int count = 0 ;
for ( i = 0 ; i < = 79 ; i ++)
{
if ( arr [i] = = 'M')
count ++ ;
}
cout << " The character M is present in array " << count << " times" ;
}
void main ( )
{
search s ;
s. getdata ( ) ;
s. find ( ) ;
}
```

Q. 178 Select the correct alternative and rewrite the following.

1. float *ptr :

In above declaration, data type of ptr is —— and data type of variable pointed by ptr is ——

(i) float, float (ii) float, int (iii) int, float (iv) pointer, float

Ans. : (iv) Pointer, float

2. When a function is called by reference, it can work an —— variables in the calling program.

(i) Original (ii) Virtual (iii) Copies of (iv) None of these

Ans. : (i) Original

3. *ptr ++ means ——

(i) Increment the content of ptr by size of data type to which ptr is pointer.
(ii) Increment the content of ptr by 1.
(iii) Increment the content of memory location pointed by ptr by 1.
(iv) None of these.

Ans. : (iii) Increment the content of memory location pointed by ptr by 1.

4. Last character of a string is ——

(i) 0 (ii) \0 (iii) \n (iv) end

Ans. : (ii) 0

5. When we use string functions such as strlen(), strrev() etc. then it must include file ——

(i) #include<string.h> (ii) #include<iostream.h>
(iii) #include<fstream.h> (iv) #include<iostring.h>

Ans. : (i) #include<string.h>

6. Objects are basic —— in object oriented programming. (March 2018)

(i) Run time entities (ii) Compile time entities
(iii) Data types (iv) None of these.

Ans. : (i) Run time entities

7. Object is a variable, whose data type is ——
(i) integer (ii) class (iii) structure (iv) float
Ans. : (ii) class

8. —— is not a visibility label. (Oct. 2011)
(i) Public (ii) Private (iii) Separate (iv) Protected
Ans. : (iii) Separate

9. The members declared under —— visibility label are hidden from external use.
(i) Public (ii) Private (iii) Both (i) and (ii) (iv) None of (i) and (ii)
Ans. : (ii) Private.

10. If all visibility labels are missing, then by default members of class are ——
(i) Public (ii) Protected (iii) Private (iv) Any of these (March 2017)
Ans. : (iii) Private

11. When a member function is defined inside the class, then it is treated as a —— function.
(i) inline (ii) outline (iii) external (iv) virtual
Ans. : (i) inline

12. A class is defined as follows,

```
class abc
{
private:
        int a;
public:
        friend void getdata (void);
};
```

The correct header for defining getdata() function outside the class is ——
(i) friend void abc::getdata (void)
(ii) void abc::getdata (void)
(iii) friend void getdata (void)
(iv) void getdata (void)
Ans. : (iv) void getdata (void)

13. A constructor can never return a value. Hence it has ....
(i) no return type. (ii) void return type
(iii) int return type (iv) any of these
Ans. : (i) no return type

14. A destructor is invoked implicitly by the compiler upon —— the program.
(i) entry in (ii) exit from
(iii) Mid point of (iv) none of these.
Ans. : (ii) exit from

15. A special function, which is used to define an additional task to an operator is called as ——
(i) operator overloading (ii) operator function
(iii) friend function (iv) constructor
Ans. : (ii) operator function

16. Operator function as a member function will have only one argument for ___ operators.
(i) unary (ii) binary (iii) sizeof (iv) none of these

Ans. : (ii) binary

17. ——— is the operator which cannot be overloaded. (Oct. 2002)
(i) scope resolution (ii) binary (iii) unary (iv) none of these

Ans. : (i) Scope resolution

18. The derivation of one class from another derived class is called as ———
(i) multiple inheritance (ii) single inheritance.
(iii) multilevel inheritance (iv) hybrid inheritance

Ans. : (iii) multilevel inheritance

19. When a class is made ——— the compiler takes necessary care to see that only one copy of that class is inherited in derived classes.
(i) virtual base class (ii) base class
(iii) derived class (iv) single class

Ans. : (i) virtual base class

20. In early binding, the correct function to be invoked is selected at ———
(i) runtime (ii) polymorphism time
(iii) compile time (iv) none of these

Ans. : (iii) compile time

21. To achieve run-time polymorphism, C++ supports mechanism of ———
(i) function overloading (ii) operator overloading
(iii) virtual functions (iv) both (i) and (ii)

Ans. : (iii) virtual functions

22. ——— stream extracts data from the file.
(i) input (ii) output (iii) input/output (iv) none of these

Ans. : (i) input

23. The class ——— is not derived from fstream base class.
(i) filebuf (ii) fstream (iii) ifstream (iv) ofstream.

Ans. : (i) filebuf

24. ios::in means ———
(i) open a file for writing only. (ii) open a file for reading only.
(iii) open fails if file already exists. (iv) open a file in binary mode.

Ans. : (ii) open a file for reading only.

25. C++ was developed by Bjarne stroustrup at ———
(i) Xerox corporation
(ii) AT and T Bell laboratories.
(iii) Polo Alu Research Centre (PARC)
(iv) None of these.

Ans. : (ii) AT and T Bell laboratories

26. ——— is not a built-in datatype in C++.
(i) void (ii) float (iii) char (iv) class

Ans. : (iv) class

27. The do-while statement is ——
(i) an entry control loop (ii) conditional statement
(iii) an exit control loop (iv) none of these.

Ans. : (iii) an exit control loop

28. Function overloading is an example of ——
(i) Inheritance (ii) Run-time polymorphism.
(iii) Compile time polymorphism (iv) All of these.

Ans. : (iii) Compile time polymorphism

29. The range of signed short int is ——
(i) - 128 to 127 (ii) - 32768 to 32767 (iii) 0 to 255 (iv) 0 to 65535

Ans. : (ii) - 32768 to 32767

30. If the value of a = 4 and b = 7, then the value of p after execution of the statement p = - - b + a + + is —— **(March 2002)**
(i) 10 (ii) 11 (iii) 9 (iv) 12

Ans. : (i) 10

31. To read data from a file, the file should be opened in ——— mode. **(March 2003, Oct. 2006)**
(i) input (ii) output (iii) append (iv) none

Ans. : (i) input

32. The ability to take more than one form is called ——— in object-oriented programming. **(Oct. 2003; March 2011, July 2017)**
(i) inheritance (ii) encapsulation (iii) polymorphism (iv) data abstraction

Ans. : (iii) polymorphism

33. Which of the following is not a feature of object-oriented programming ? **(March 2004)**
(i) Follows bottom-up approach in program design.
(ii) Objects may communicate with each other through functions.
(iii) Follows top-down approach in program design.
(iv) Programs are divided into what are known as objects.

Ans. : (iii) Follows top-down approach in program design.

34. Programming in C++ using classes is called ———programming. **(Oct. 2004)**
(i) procedure oriented (ii) event drivern
(iii) object oriented (iv) database

Ans. : (iii) object oriented

35. ——— is not a derived data type in C++. **(March 2005; Oct. 2010)**
(i) Class (ii) Array (iii) Function (iv) Pointer

Ans. : (i) Class

36. ——— is not the feature of OOPs. **(Oct. 2005, 09)**
(i) Polymorphism
(ii) Inheritance
(iii) Data Abstraction
(iv) Top-down Approach

Ans. : (iv) Top-down Approach

37. A derived class with several base classes is ——— inheriance. (March 2006, 09, 22, Oct. 2021)

(i) single (ii) multiple (iii) multilevel (iv) hierarchical

Ans. : (ii) multiple

38. Which of the following allows to access the private data of other class is ____

(i) In -line function (ii) Friend function
(iii) Main function (iv) All of the above (March 2007)

Ans. : (ii) Friend function

39. While accessing the number in a float array using pointer, the pointers value every time increases by _____.

i) 2 ii) 4 iii) 8 iv) 16 (Oct. 2007)

Ans : (ii) 4

40. Following operator. cannot be overloaded. (March 2008)

i) ++ ii) :: iii) – iv) *

Ans. : (ii) ::

41. _____ type of member function of class never takes any argument. (Oct. 2008)

(i) Constructor (ii) Destructor
(iii) Operator (iv) None of these

Ans : (ii) Destructor

42. For a = 23 and b = 3, the value of c after execution of the statement c = (a/b) * (a%b) will be _____. (March 2010)

(i) 14 (ii) 49 (iii) 21 (iv) 69

Ans. : (i) 14

43. Out of the following C++ operators, _____ operators can be overloaded. (March 2012)

(i) Sizeof (ii) : : (iii) ?: (iv) *

Ans. : (iv) *

44. A pointer is a. variable that holds _____ of another variable. (Oct. 2012)

(i) Value (ii) Memory Address
(iii Data-type (iv) None of these

Ans. : (ii) Memory Address

45. Object Oriented Programming follows _____ approach in program design. (March 2013)

(i) top-down (ii) Non-hieranchical (iii) random (iv) Bottom-up

Ans. : (iv) Bottom-up

46. When a base class is privately inherited by a derived class, public member of the class becomes _____ of the derived class.

(i) Public Member (ii) Protected Member
(iii) Private Member (iv) Non-Member (Oct. 2013; July 2016)

Ans. : (iii) Private Member

47. In $C^{++}$ _____ is an extraction operator. (Oct. 2014)

(i) << (ii) >> (iii) && (iv) !

Ans. : (ii) >>

48. Object oriented programming uses_____ approach of programming. **(March 2015)**

(i) Linear (ii) Non-linear (iii) Top down (iv) Bottom up

Ans. : (iv) Bottom up

49. We can not define more than one _____ in a class. **(Oct. 2015)**

(i) Constructor (ii) Destructor (iii) Member Function (iv) Data Member

Ans. : (ii) Destructor

50 In C++, double type data consumes _____ bytes in memory.. **(March 2016)**

(i) 2 (ii) 4 (iii) 6 (iv) 8

Ans. : (iv) 8

51. _____ symbol is used to declare destructor function in C++. **(July 2018)**

(i) @ (ii) # (iii) ~ (iv) !

Ans. : (iii) ~

52. What will be the value of x after execution of following expression in C++?

X = + + m + n + + ; where m = 10 and n = 15. **(March 2019)**

(i) 25 (ii) 27 (iii) 26 (iv) 28

Ans. : (iii) 26

53. _____ operator can be overloaded. **(July 2019)**

(i) + (Plus) (ii) | | (Logical OR)

(iii) % (Modulus) (iv) All i, ii and iii

Ans. : (iv) All i, ii and iii

54. _____ operator cannot be overloaded. **(March 2020)**

(i) ++ (ii) +

(iii) : : (iv) >>

Ans. : (iii) : :

55. If any access specifier is not specified for member function or data number in class then by default it is _____. **(Dec. 2020)**

(i) public (ii) private

(iii) protected (iv) void

Ans. : (ii) private

❑❑❑