PAGE NO.:

- Difference between C and C++

C	C++
1] This is called procedure oriented Language	1] This is called object oriented Language.
2] Large programs can be divided into smaller parts using function	2] programs have classes and object only the functions in a class can access the data in a given object.
3] Data can be access by any function and Hence there is a scope of unexpected change of in a Data.	3] Data can be access only by the functions of that class & hence it can not be change unexpectedly
4] Global data is allow hence data is not hidden	4] no global data, Data is encapsulated.
5] if changes are done in some data the corresponding changes are to be implemented in all the functions using the Data.	5] Adding of new Data or changes in the data requires to change only the functions of the class.

6] It follows top-down approch in program design.	6] It follows bottom-top approch in program design.

⇒ Basic structure of C++

```
#include <iostream.h>
#include <conio.h>
Void main()
{
  Cout<<"Hello Word";
}
```

Q Enumerated Data

```
#include <iostream.h>
Void main()
{
enum weekday ({Sun, mon, tue, coed, thar, fri, sat};
 workday today;
 if (today == Sun || today == Sat)
 {
 Cout << "Holiday";
 }
 else
 {
  Cout << "working day";
 }
getch();
}
```

Q.1 Write a program to accept a number and display its square?

→

```
#include<iostream.h>
#include<conio.h>
Void main()
{
 int n, Square;
 Cout<<" Enter the number";
 cin>> n;
 Square = n*n;
 Cout << " Square is :" << Square;
 getch();
}
```

Q.2 Write a program to calculate simple interest taking principal, Rate of interest, year as input from User.

→

```
#include<iostream.h>
#include<conio.h>
Void main()
 {
int P, R, y, mul;
Float SI;
Cout << " Enter principal Amount, Rate of interest
        and year";
cin >> P >> R >> y;
mul = P*R*y;
SI = mul/100;
Cout << " Simple interest is" << SI;
}
```

Q3: Write a program to swap numbers ?

→

```
#include <iostream.h>
# include <conio.h>
Void main()
{
 int a, b, swap;
 Cout << " Enter Values";
 Cin >> a >> b;
  swap = a;
   a = b;
   b = swap;
 Cout << " After Swapping" <<a<<b;
 getch();
}
```

Q.4 Write a program to accept a Float number & Display the integer part using type casting operator.

→

```
# include <iostream.h>
# include <iostream.h>
Void main()
{
 Float a;
 int b;
 Cout << " Enter a";
 Cin >> a;
 b = (int) a;
 Cout << " Value is :" << b;
 getch();
}
```

Q.5 Write a program to accept a number from user and check whether it is greater than 50.

→

```
#include <iostream.h>
#include <conio.h>
Void main()
{
int n;
Cout << "Enter the number";
cin >> n;
if (n > 50)
{
 Cout << "number is greater";
}
getch();
}
```

Q.6 Write a program to check given number is even or odd ?

→

```
#include <iostream.h>
#include <conio.h>
Void main()
{
int n;
Cout << "Enter the number";
cin >> n;
if (n % 2 == 0)
{
 Cout << "Number is even";
}
```

```
else
{
 cout << " Number is odd";
}
getch();
}
```

Q.7 Write a program to check given number is Positive, negative, zero.

→

```
#include <iostream.h>
#include <conio.h>
Void main()
{
int n;
cout << " Enter the number";
cin >> n;
if (n>0)
{
 cout << " Number is positive";
}
else if (n<0)
{
 cout<< " Number is Negative";
}
else
{
 cout<< " Number is zero";
}
getch();
}
```

Q8. Write a program to accept a number from user & check whether that number is Divisible by 4 & 8?

→

```
#include <iostream.h>
#include <conio.h>
Void main()
{
int n;
Cout << "Enter the number";
cin >> n;
if (n % 4 == 0 && n % 8 == 0)
{
  Cout << "No. is Divisible by 4 & 8";
}
else
{
  Cout << " number is not Divisible by 4 & 8";
}
getch();
}
```

PAGE NO.:

Loop

1] For loop

```
for (initialisation ; condition ; updation)
{
    statements;
}
```

2] while loop

```
initialisation
while (condition)
{
  statements;
  updation
}
```

3] do - while loop

```
initialisation;
do
{
 statements;
 updation;
}while (condition);
```

2 Write a program to print 10 to 50 Numbers using any loop.

```
#include <iostream.h>
#include <conio.h>
Void main()
{
 int i;
 for(i=10 ; i<=50 ; i++)
 {
  cout << i << endl;
 }
 getch();
}
```

2 Write a program to print 10 to 0 numbers using for loop.

```
# include <iostream. h>
# include <conio.h>
Void main()
{
 int i;
 for (i=10; i>=0 ; i--)
 {
   cout << i << "endl;
 }
getch();
}
```

Q. Write a program to find out even no between 1 to 50 ?

```
#include <iostream.h>
# include <conio.h>
Void main()
{
 int i ;
for (i=1 ; i<=50 ; i++)
 {
  if ( i%2 == 0)
 {
  cout<<i<<endl;
 }
}
getch();
}
```

Q. Write a program to access a number from user and find out table of that number.

```
# include <iostream.h>
# include <conio.h>
Void main()
{
  int i,j;
  cout<<" Enter the number";
  cin >> I ;
  for (i=I ; i<10 ; i++)
  {
   cout << i*I ;
  }
}
```

PAGE NO.:

Operators in C++

1] Arithmatic operator
+ , - , * , / , %

2] Relational operator
> , < , >= , <= , == , !=

3] Assignment operator
= , += , -= , *= , /=

4] Logical operator
&& , || , !

5] Conditional operator
> , < , ≥ , ≤

6] Bit-wise operator
>> , <<

7] Scope Resolution
::

8] memory management
new, delete , malloc()

9] manupulator
endl , setw() , setfill()

1] Scope Resolution : (::)
The Scope Resolution operator in c++ enables us to access Variables, functions, or classes Defined in different Scope.

2] memory management operator :
The "new" operator is used in c++ for allocating memory Dynamically. In the c programming language malloc() for the same.

New operator	Malloc()
I] It is an operator	I] It is a function
II] It returns the exact datatype	2] It returns the void type data which is to be typecasted to the required datatype.
III] It calles the constructor and can also be used to define the primitive datatype.	3] It does not call the constructor.

⇒ The "Delete" operator is used in c++ for Deallocating memory Dynamically. which was allocated by new operator.

3] Manupulator

manupulator are used to perform certain special operation, like moving the carsor to new line, fixing the width of a data value etc

The Different manupulator used in C++ are endl, setw(), setFill(), setprecision().

Difference between Class & Structure

Class	Structure
1] Class is the reference type and its object is created on the hip memory.	1] Structure is a Value type that is why its variable is created on the Stack memory
2] Class can inherite the another class	2] Structure does not support the inheritance.
3] Class can have all types of constructor and Destructor	3] Structure doesn't have Constructor and destructor.
4] The member variable of class can be initialize directly	4] The member of Variable of structure can not initialize directly

Class and object

- Class :
It is a type or category of this which is similar to a structure with the difference that it can also have functions beside data items.

Class in c++ is a user defined type

Syntax:

```
Class class_name
{
Access specifier;
Data member;
Access specifier:
member function;
}
  class object;
```

- Access Specifier
There are three Access Specifier, Public, protected, Private.

This access specifier are used to specify the access of the data and function members of a class

A Public Access Specifier Indiate that everg Fanction can Access this member

2] A protected Access specifier indicates that only class derived can used this member.

3] A Private Access specifier indicate that only the functions of the same class can Access the member.

Defining member functions of a class.

Threre are three methods to defining member functions of class

I] Internally defined function
II] Externally defined function
III] Inline defined function.

I] Internally defined function [Inside class]
When the member functions of the class are define inside the class itself they are called as internally defined member function.

Q.1 Write a C++ Program to find out area of circle such that the class circle must have three member function namely
1] read function : to accept radius from user.
2] Compute () : To calculate area of circle.
3] display () : For displaying the area of circle.

```
#include <iostream.h>
#include <conio.h>
Class circle
{
Float a, r;
Public:
Void read()
{
Cout << " Enter radius";
cin >> r;
}
Void Compute()
{
a = 3.14 * r * r;
}
Void display()
{
Cout << " area :" << a;
}
};
Void main()
{
Circle c;
c.read();
c.Compute();
c.display();
}
```

Q2. Write a program to calculate value of the following series using internal member function.

→
```
#include <iostream.h>
#include <conio.h>
Class Series
{
 int sum, n, i;
 Public:
 Void read()
 {
  Cout << "enter n";
  Cin >> n;
 }
 Void Compute()
 {
  Sum = 0;
  for (i = 1 ; i <= n; i++)
  {
   sum = sum + (i*i);
  }
 }

 Void display()
 {
  Cout << "Sum of Series = " << sum;
 }
};

Void main()
{
 Series S;
 S.read();
```

```
S.compute();
S.display();
}
```

Q3. Write a program to define a class Student having data members name & Roll_no. Accept and Display data for one object.

→

```
#include <iostream.h>
#include <conio.h>
Class Student
{ Public:  // Private:
  int Roll_no;
  Char name[100];
  Public:
Void Accept ()
{
 Cout << " Enter your Roll_no";
 cin >> Roll_no;
 Cout << " Enter your name";
 cin >> name ;
}

Void display()
{
Cout << " Student name:" << name;
Cout << " Student Roll no :" << Roll_no ;
};
Void main()
{
```

```
Student s;
s.Accept();
s.display();
}
```

Q4. Devlop a C++ program to create Structure Student with Data members name, Roll_no, Percentage, Accept and display data for 10 Student using structure.

→
```
#include<iostream.h>
#include<conio.h>
Struct student
{
  int roll_no;
  Char name[100];
  Float percent;
};

Void main()
{
 Struct student s[10];
 int i;
 Cout<<" Student Information"<<endl;
 for(i=0; i<10; i++)
 {
 Cout<< "Enter name:"<<endl;
 Cin>>s[i].name;
 Cout<<"Enter Roll no:"<<endl;
 cin>>s[i].roll_no;
```

```
cout<<" Enter percentage :" <<endl;
cin >> s[i].percent;
}

cout << " Student Details are" <<endl;
for(i=0; i<10; i++)
{
cout<<" Student Roll no :" << s[i].roll_no<<endl;
cout<<" Student Name:"<<s[i].name<<endl;
cout<<" Student percentage :" << s[i].percent<<endl;
}

}
```

Q.5 Write a program to declare class student data member name and percentage. Write a constructor to initialize data & data memember to accept & display data for 1 student.

→
```
#include<iostream.h>
#include<conio.h>
Class Student
{
Private:
 Char name[50];
 Float percentage;
 Public:
Void accept ()
{
Cout<<" Enter the name :" << endl;
Cin >> name;
Cout<<" Enter the percentage :" << endl;
Cin>> percentage;
}
Void display ()
{
Cout << " Student Name :" << name << endl;
Cout << " Student Percentage :" << percentage << endl;
}
};

Void main()
{
   Student s;
   s.accept();
   s.display();
}
```

PAGE NO.:

Nested member function or nesting of member function:

A member function of A class calling another member function of the same class is called as nesting of member function.

Q. Write a program to find out area of square using nested member function.

→
```
#include <iostream.h>
#include <conio.h>
Class Square
{
  Public:
  int s, area;
  Void read()
{
cout << " enter size of square";
cin >> s;
}
int Compute()
{
area = s * s;
return area;
}
Void display()
{
    Compute();
    Cout<< " area of Square : " << area;
};
```

```
Void main()
{
 Square sq;
 sq.read();
 sq.Compute(); //
 sq.display();
}
```

Q. Write a program to find out Mul Sum of following Series using nested member function.

Sum = 1 * 2 * 3 * 4 .... * n

→

```
#include <iostream>
#include <conio.h>
Class Series
{
 Public :
 int i, n, Mul;
 Void read()
{
 Cout << "enter the n";
 Cin >> n;
}

 int Compute()
{
 Mul = 1;
for (i=1; i<n; i++)
 {
    Mul = Mul * i;
 }
    return Mul;
```

```
Void display()
{
  Compute();
cout << " multiplication of Series :" << mul;
}

Void main()
{
 Series s;
  s.read();
   s.display();
}
```

2] Externally Defined function or outside the class.

When the member function of the class are defined out side the class they are called as externally defined member function.

The functions when defined outside must use scope Resolution operater to define the scope of the function to be within the class

Syntax :

```
return type :: f^n_name (argument list)
  class name
  {
     statements;
  }
```

Q1. Write a Program to find area of circle such that the class circle must have Three extrnally defined member functions
read function : Accept radius from user.
Compute (): For calculating area.
display (): Displaying result.

→
```
# include <iostream.h>
# include <conio.h>
Class circle
{
Public :
int r;
Float a;
Void read();
Void Compute();
Void display();
};

Void circle :: read()
{
Cout<< "Enter radius";
Cin >> r;
}

Void circle :: Compute()
{
a = 3.14 * r * r;
}
```

```
Void circle :: display()
{
Cout << " area : " << a;
}

Void main()
{
Circle c;
C.read();
C. compute();
C. display();
}
```

Q2. Write a Program to calculate the Value of the following series using external member function.

→

```
#include <iostream>
#include <conio.h
Class Series
{ Public:
  int sum, n, i;
};

Void Series :: read()
{
Cout << " Enter n";
cin >> n;
}
```

```
Void Series :: compute()
{
 Sum = 0;
for (i=0 ; i<=n ; i++)
{
   Sum = Sum + (i*i*i);
}
}

Void Series :: display()
{
  Cout<< " Sum of Series is:" << Sum;
}

Void main()
{
 Series S;
 S.read();
 S.compute();
 S.display();
}
```

Q.3 Write a Program to calculate Factorial of a number using External member function.

→

```
#include <iostream.h>
#include <conio.h>
class Fact
{
 Public :
  int i, fact, n;
};

Void Fact :: read()
{
 Cout << " Enter n";
 cin >> n;
}

Void Series :: Compute()
{
 fact = 1;
 for (i = 1 ; i <= n; i++)
 {
 fact = fact * i;
 }
}

Void Fact :: display()
{
 Cout << " Sum of Series :" << sum;

}
```

```
Void main()
{
  Fact F;
  F.read();
  F.Compute();
  F.display();
}
```

⟹ features of characterstics : of object oriented programming.

1. Class :
A class in c++ is user defined type. It is a group of similar object It is similar to a structure with the difference that it can also have functions beside data items.

2. object :
It is an instance of a class that encapsulate data and functionality together

3. Data Abstraction :
Data Abstraction is like defining or abstracting the object according to the required parameters

Data Abstraction provides only the essential details to the outside world and hiding the internal details.

4. Data Encapsulation:
Data Encapsulation involves combining similar data and functions into a single unit called as class.

The Data of an object is hidden from other object this is called as encapsulation.

5. Inheritance:

The mechanism of deriving the properties of one class into another class known is inheritance.

6. Polymorphism:

Polymorphism means multiple forms of the samething and it occurs when we have many classes that are related to each other By Inheritance.