**CLASSES**

* Classes bind data and function in the single entity.
* Difference between class and structure is that by default, members of class are private and that of structure are public.
* There is no general rule for when to use classes and when to use structure. Generally,c++ programmers use structure for holding data and classes for data and function.
* Class is basic mechanism to provide Encapsulation. Data Encapsulation is important concept of oops which bind data and function in a single entity in such away that that can be manipulated through functions only.
* Class has three access specifiers : public, protected, private.
* Data hiding is implemented through private access specifier of class.
* Function defined inside class is treated as inline func by default, provided they don’t fall into restricted category of inline fn;
* Function containing loop,switch,goto,static variable pr recursive code cant be INLINE.
* Initialising class data members inside class and no semi colon at end of class results in compile time error.
* Inline functions make the code faster, so functions defined outside class can be made by adding inline keyword in the function definition.
* Memory for member function in allocated only once when class is defined. Therefore single copy of function is shared among all objects.
* Memory for data members is allocated when object is declared of class.
* When data member is declared as static:

>>single copy of static member is created in memory irrespective of number of objects made.

>>all objects share same static member.

>>static data member is initialized to zero when object is created by default.

>>only visible in class i.e can be used by class member functions only.

>>although static data members are declared inside the class ,they are not considered as a part of the class. They are defined outside the class.

Class sample int main()

{ static int x; {sample obj,obj1,obj2;

Public: obj.getData();obj1.getData();

Void getData() } //value of x is now 2

{x++};

};

Int sample::x=1;

* Object of class with no member has size of 1-byte.
* **Static member Functions:**

>>allocated to memory only once just like static data member.

>>can only access static members within same class

>>cant access non-static members because non-static members are associated with objects and static have no object to work with.

>>called using class name and scope resolution operator.

>>since not attached with object, this pointer cant be used with static members.

>>static members fn cant be declared as virtual function.

>>fn can be declared with const,volatile type qualifiers.

>>static and non-static members with same names in same class is illegal.

* **Static object:** object of a class can be declared as static. By doing so, all the data members of class are initialized to zero;

Class sum

{ int a,b;

Public:

Void sum()

{ a+=10;

B+=10;}

Void display()

{cout<<a<<b;}

};

Static sum s;

s.dispaly() // a=0,b=0;

s.sum() //

s.display() //a=10 b=10;

* **FRIEND FUNCTION:**

A friend function of a class is a non-member function of a class that can access the private and protected members of the class. Friend function helps to implement data encapsulation in c++.

>> it is not the member function of the class

>>it is not called using . or -> operator as it is defined outside the class

>>friend function can be declared either in public or private section.

>>they don’t require ‘this’ pointer.

>> same function can be declared as a friend in any number of classes.

>> A friend function can access the members of class directly using object with dot (.) operator followed by member name.

>>used frequently to perform operations that are conducted on two different classes

#**Example:**

Class A

{Private:

Int x;

Public:

Friend void display(A&,B&);

}

Class B

{Private:

Int y;

Public:

Friend void display(A&,B&);

}

Void display(A &a, B &b)

{ cout<<a.x-b.y;

}

* **Empty class** is a class with no data member or member function and has a size of 1 byte.

Example: class student{};

* **Nested class** is a class defined inside the class.

>> scope of nested class is restricted by the scope of outer class.

>>while declaring the object of inner class , name of the inner class must be preceeded with the name of outer class.

Example: A::B b;

* **Friend class**

Friend class is one which can access private or protected members of another class. To declare all members of class A as friends of class B.

Class B;

Class A{

Friend class B;

Data member

Function;

};

Class B{

Data memebers

Member function;

}