Friend Function/Class

- It has been emphasized that the private members cannot be accessed from outside the class.
- That is, a non-member function cannot have an access to the private data of a class.
- However, there could be a situation where we would like two classes to share a particular function.
- Ex: Consider a case where two classes, **manager** and **scientist,** would like to use a common function **income_tax()** to operate on the objects of both the classes.
- C++ allows the common function to be made friendly with both the classes, thereby allowing the function to have access to the private data of these classes.
- Such a function need not be a member of any of these classes.

Friend Non-member Function

- A friend function is a non-member function that has special rights to access private data members of any object of the class of whom it is a friend.
- In this section, we will study only those friend functions that are not member functions of some other class(i,e, outside function).
- To make an outside function “friendly” to a class, we have to simply declare this function as a friend of the class as shown below:

```
class abc
{
   ....
   ....
public:
   ....
   ....
   friend void xyz(void);  // declaration
};
```

- The function declaration should be preceded by the keyword **friend**.
- The function is defined elsewhere in the program like a normal C++ function.
- The function definition does not use either the keyword **friend** or the scope resolution operator (::).
- The functions that are declared with the keyword **friend** are known as ***friend functions***.
- A function can be declared as a friend in any number of classes.
- A friend function, although not a member function, has full access rights to the private members of the class.
- Friend functions are not called with respect to an object.

Characteristics of Friend Functions

- It is not in the scope of the class to which it has been declared as **friend**.
- Since it is not in the scope of the class, it cannot be called using object of that class.
- It can be invoked like a normal function without the help of any object.
- Unlike member functions, it cannot access the member names directly and has to use an object name and dot membership operator with each member name. (Ex: A.x)
- It can be declared either in the public or the private part of a class without affecting its meaning.
- Usually it has the objects as arguments.

Ex: Friend1 program

Friend Member Functions

- Member functions of one class can be **friend** functions of another class.
- In such cases they are defined using the scope resolution operator.

```
class X                          class Y
{                                {
    - - - - -                        - - - - -
    - - - - -                        - - - - -
    int fun1();                      friend int X :: fun1();      //fun1() of X is friend of Y
    - - - - -                        - - - - -
};                               };
```

The function fun1() is a member of **class X** and a **friend** of class Y.

```cpp
// C++ program to demonstrate the working of friend function
#include <iostream>
using namespace std;
class Distance {
    private:
        int meter;
        // friend function
        friend int addFive(Distance);
    public:
        Distance() : meter(0) {}

};
// friend function definition
int addFive(Distance d) {
    //accessing private members from the friend function
    d.meter += 5;
    return d.meter;
}
int main() {
    Distance D;
    cout << "Distance: " << addFive(D);
    return 0;
}
```

```
//Friend function as bridge between two class
#include<iostream>
class second; //pre-declaration of class
class first
{
 private:
      int data1;
     public:
      void setdata(int x)
  {
 data1=x;
 }
friend int sum(first a, second b);//friend function
};
class second
{
 private:
      int data2;
    public:
     void setdata(int x)
       { data2=x;}
       friend int sum(first a, second b);//friend function
};
```

```
int sum(first a, second b)
{
 return (a.data1 + b.data2);
}

int main()
{
 first a;
    second b;
    a.setdata(15);
    b.setdata(10);
    cout<<"sum of first and second is:"<<sum(a, b);//displays 25
}
```

Friend Classes

- We can also declare all the member functions of one class as the friend functions of another class.
- In such cases, a class is called as a **friend class**.
- That is, a class can be a friend of another class.
- Member functions of a friend class can access private data members of objects of the class of which it is a friend.

Ex: If class B is to be made a friend of class A, then the statement

```
friend class B;
```

Should be written within the definition of class A.

Ex:

```
class B;          //Forward declaration.
class A
{
    friend class B;
      //rest of the class A
};
```

- It does not matter whether the statement declaring class B as a friend is mentioned within the private or the public section of class A.
- Now, member functions of class B can access the private data members of objects of class A.
- Forward declaration: Necessary because definition of class B is after the statement that declares the class B a friend of class A.

NOTE:

- Friendship is not transitive.
- That is, If class A is friend with class B, and class B is friend with class C. This doesn’t mean that class A is friend with class C.

```cpp
// C++ program to demonstrate the working of friend class
#include <iostream>
using namespace std;
class ClassB; // forward declaration
class ClassA {
    private:
        int numA;
        friend class ClassB; // friend class declaration
    public:
        ClassA() : numA(12) {} // constructor to initialize numA to 12
};
class ClassB {
    private:
        int numB;
    public:
        ClassB() : numB(1) {}  // constructor to initialize numB to 1
    // member function to add numA from ClassA and numB from ClassB
    int add() {
        ClassA objectA;
        return objectA.numA + numB;
    }
};
```

```
int main() {
    ClassB objectB;
    cout << "Sum: " << objectB.add();
    return 0;
}
```