Friend Function

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. For example, consider a case where two classes, manager and scientist, have been defined. We would like to use a function income_tax(()
to operate on the objects of both these classes. In such situations, 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 to be a member of any of these classes.

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 adc
{
        …..
        …..
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 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.

A friend function possesses certain special 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 the scope of the class, it cannot be called using the object of that Class. It can be invoked like a normal function without the help of any object,
3. 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.
4. 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. The friend functions are often used in operator overloading

Function friend to a class:

Example:

```
#include<iostream.h>
class complex
{
        float x;
        float y;
public:
```

```
        void input(float real, float imag)
        {
                x=real;
                y=imag;
        }
        friend complex sum( complex, complex);
        void show( complex);
};
complex sum ( complex c1, complex c2)
{
        complex c3;
        c3.x=c1.x+c2.x;
        c3.y=c1.y+c2.y;
        Return ( c3);
}
void complex :: show( complex)
{
        cout<<c.x<<"+j"<<c.y<<"\n";
}

int main( )
{
        complex a,b,c;
        a.input(3.1.5.65);
        b.input(2.75,1.2);
        c=sum(a,b);
        cout<<"a=";
        a.show(a)
        cout<<"b=";
        b.show(b)
        cout<<"c=";
        c.show(c)
}
```

Common function friend to two classes:

```
#include<iostream.h>
#include<conio.h>
class B;
class A
{
        int m;
public:
        void getdata()
        {
                cout<<"enter the value of m:\n";
                cin>>m;
        }
        void display(void)
        {
                cout<<m;
```

```
        }//end of display
        friend void swap (A &a1,B &b1);
}; //end of class a
class B
{
        int n;
public:
        void getdata()
        {
                cout<<"enter the value of n:\n";
                cin>>n;
        }
        void display(void)
        {

                cout<<n;
        } //end of display()
        friend void swap(A &a1,B &b1);
}; //end of class b
void swap(a&a1,b&b1)
{
        int temp=0;
        temp=a1.m;
        a1.m=b1.n;
        b1.n=temp;
        cout<<"m:\n"<<a1.m;
        cout<<n:\n"<<b1.n;
} //end of swap()


void main()
{
        A c;
        B d;
        c.getdata();
        d.getdata();
        cout<<"before swaping:\n";
        c.display();
        d.display();
        swap(c.d);
        cout<<"after swaping:\n";
        c.display();
        d.display();
} //end of main()
```

Output:

```
Enter value of m:
3
enter value of n
2
```

before swaping:
m:3

Friend member function

If a member function of class wants to access the private data members of another class then member function of class has to be declared as friend of that class. This type of friend functions are known as a friend member function.

Example: In this example function of class A is friend of class B. Member function of class A will be used to initialize the data member of class B

```
#include<iostream.h>
#include<conio.h>
class B; //Forward Declaration
class A
{
        public:
        void set(B &, int);

};
class B
{
        int x;
public:
        int get()
        {
                return x;
        }
        friend void A:: set(B &,int); //member function of class A declared as friend in class
                                        B
};
void A::set(B &j,int k)
{
        cout<<"Function of class A friend of class B"<<"\n";
        j.x=k;
}
void main()
{
        clrscr();
        A a;
        B b;
        a.set(b,5);
        cout<<"Value of class B data member:"<<b.get();
getch();
}
```

Output:

Function of class A friend of class B
Value of class B data member:5

Friend Class

A class can be made as a friend of another class. If class 'A' is a friend of class 'B' then, all the member function of class 'A' can access the private members of 'B'. But, the member functions of class 'B' is restricted in accessing the private members of class 'A'. To declare class A as friend of class, the following statement has to be included in either private or public section of a class B.

```
friend class A;
```

Example: Here class B is friend of class A therefore, class B is declared friend in class A. Now, the two members function of class B will be used to set the values of data members of class A

```
#include<iostream.h>
#include<conio.h>
class B;
class A
{
        int x;
        int y;
public:
        void show()
        {
                cout<<"x is :"<<x<<"\n"<<"y is:"<<y<<"\n";
        }
friend class B;

};
class B
{
        public:
        void setx(A &a, int b)
        {
                a.x=b;

        }
        void sety(A &a, int b)
        {
                a.y=b;
        }
};
void main()
{
        clrscr();
        A a;
        B b;
        b.setx(a,10);
        b.sety(a,30);
        cout<<"value of x and Y"<<"\n";
        a.show();
```

```
getch();
}
```

Output:

```
Value of x and y
x is :10
y is : 30
```