Friend Functions & Classes

- Note: Private members cannot be accessed from outside the class. i.e a non-member function cannot have an access to private data of a class.

- **If any situation where we would like two or more classes to share a particular function, what to do??**

- **C++ allows a 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 these classes.**

Friend Functions- Declaration

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

- To make an outside function friendly to a class, we have to declare this function as friend of the class.
- Function is defined anywhere like normal C++ function.
- Function definition **does not use either keyword friend or scope operator::**
- **Although not a member function, has full access rights to private member of the class.**

Characteristics of Friend Functions

- A friend function is created by placing the keyword friend in the **function declaration** but not in function definition. Exception is if you declare and define at the same place.
- A friend function is a friend of the class in which it is declared.
- A friend function is not a member function of the class and cannot be called from any object of the class using dot operator.
- A friend function can have full access to the public, private and protected data member of the class to which it is a friend.
- The **arguments of friend functions are usually objects of the class to which it is a friend**.
- A friend function not being a member function of class is called as a normal function.

Characteristics of Friend Functions

- A friend function can be friend of more than one class.
- A function of one class can be a friend of another class.
- We can have whole class as a friend of another class
- We use friend function usually with multiple classes but can used with single class also.
- A friend function can be declared in the public or private visibility mode without affecting its meaning.

Syntax- Friend Functions

```
class demo
{
        data members :
        public :
        members functions;
        // friend function declaration
        friend data_type function_name (parameters);
};
data_type function_name (parameters) //definition
{
        function definition;
}
```

Note : we are not using like this

```
Void product :: getdata(int a, float b)
{
        number=a;
        cost=b;
}
```

Example-1 Friend Functions

```
class demo
{
        int y;
        public :
        void input(int x)
        {
            y=x;
        }
friend int findsqr(demo);
};
```

```
int findsqr(demo d)
{
        return d.y * d.y;
}
void main( )
{
        demo F;
        F.input(30);
        cout<<"Square
        is="<<findsqr(F);
}
```

OUTPUT:
Square is = 900

Friend function with single class

Example-2 Friend Functions

```
class sample
{
        int a,b;
        public :
        void setvalue()
        {
            a=10;
            b=20;
        }
friend float avg(sample s);
};
```

```
float avg(sample s)
{
return float(s.a+s.b)/2;
}
int main( )
{
        sample x;
        x.setvalue();
        cout<<"Average is =”<<avg(x);
        return 0;
}
```

OUTPUT:
Average is = 15

Friend function with single class

Example-3 Max of two data of two different classes

```
#include <iostream.h>
class second;// declaration
class first
{
        int fx;
public :
void inputf(int x)
{
        fx=x;
}
friend void findmax(first,second);
};
```

```
class second
{
        int sx;
public :
void inputs(int x)
{
        sx = x;
}
friend void findmax(first,second);
};
```

friend of both the class first and second

Friend function with two different class

Example-3 Max of two data of two different classes

```
void findmax(first A, second B)
{
if(A.fx>B.sx)
        cout<<A.fx<<"of class first is greater than
        "<<B.sx<<"of class second<<endl;
else
        cout<<B.sx<<"of class second is greater
        than"<<A.fx<<"of class first<<endl;
}
 void main( )
 {
        first F;
        second S;
        F.inputf(40);
        S.inputs(70);
        findmax(F,S);
 }
```

Output:
70 of class second is greater than 40 of class first

Example-4: Function of one class friend of another class

```
#include <iostream.h>
class second;
class first
{
        int num;
public :
void input_first( )
{
        num=20;
}
void show(second);
```

Created function in class first which will be friend of class second

```
class second
{
int num;
        public :
void input_second(int x)
{
        num==x;
}
friend void first : :show(second);
};
```

*we can have object of **class second** in the function of the first class*

Example-4: Function of one class friend of another class

```
void first : :show(second s)
{
cout<<"NUM OF CLASS FIRST :="<<num<<endl;
s.input_second(num*num);
cout<<"NUM OF CLASS SECOND :="<<s.num<<endl;
}

void main( )
{
        first f;
        f.input_first( );
        second s;
        f.show(s);
}
```

Output:
NUM OF CLASS FIRST :=20
NUM OF CLASS SECOND :=400

Example-5:Whole class as a friend of another class

```
#include <iostream.h>
class second;
class first
{
public :
void output( )
{
cout<<"FIRSTCLASS "<<endl;
}
friend class second;
};
```

```
class second
{
public :
void show(first s)
{
s.output( );
}
};

void main( )
{
        second s;
        first f;
        s.show(f);
}
```

Output:First class

Thank You