Friend function

* A friend function is a non-member function, that can have an access to all private and protected members of the class.
* To declare an outside function as a friend of a class, precede the function prototype in class definition with the keyword friend.

Syntax:

```
Class classname
 {...
  friend returntype fn.name (args); //friend fn. declaration.
 }
returntype fn.name (args)   // Friend fn. definition
 {...
 }
```

Characteristics:

* A function can be declared as friend in any number of classes.
* A friend function definition does not use either the keyword 'friend' or the scope resolution (::) operator.
* A friend function is not in the scope of the class, to which it has been declared as friend.
   - can be invoked like normal function without using the object of that class.
* Cannot access member names directly. Has to use object name and dot membership operator with each member name.
* Can be declared either in public (or) private part of class.
* Usually has objects as arguments.

Categories of friend Function

(i) A normal function being friend with a single class
(ii) A normal fn, being friend with more than one class.
(iii) A member function, friend with another class.
(iv) A class, friend with another class.

(i) A normal fn, being friend with single class

(Eg:1)

```
# include <iostream.h>

class Distance
 {
 int meter;
 public:
   Distance()
    {
    meter=5;
    }
   friend int addfive (Distance); // can access private data 'meter'
 };
 int addfive (Distance d)
  {
   d.meter += 5;
   return d.meter;
  }
 void main()
  {
   Distance d1;
   cout << "Distance :" << addfive (d1); // objects as arguments.
  }
```

O/P:

```
Distance : 10
```

Eg 2 // Finding mean

```
#include <iostream.h>
class base
{
float a,b;
public:
void getdata()
{
cout << "Enter A and B\n";
cin >> a >> b;
}
friend void mean(base ob);
};
void mean(base ob)
{
float c = (ob.a + ob.b)/2;
cout << "Mean value : " << c;
}
void main()
{
base obj;
obj.getdata();
mean(obj);
}
```

ii) A normal function, being friend with more than one class

* A friend function can be used to operate on objects of two different classes - works as a bridge between classes.

(Eg) Calculation of mean of data members of two diff. classes using friend function.

(Eg) // Mean of data members of 2 classes

```
#include <iostream.h>
class base2; // forward declaration - to make compiler know the presence of base2.
class base1
{
float a;
public:
void getdata()
{
cout << "Enter A:\n";
cin>>a;
}
friend void mean(base1, base2);
};
class base2
{
float b;
public:
void getdata()
{
cout << "Enter B:\n";
cin>>b;
}
friend void mean(base1, base2);
};
void mean (base1 ob1, base2 ob2)
{
float c= (ob1.a+ ob2.b) / 2;
cout << "Mean value : " << c;
}
void main()
{
base1 obj1;
obj1.getdata();
base2 obj2;
obj2.getdata();
mean(obj1, obj2);
}
```

(iii) A member function, being friend with another class.

* Here member fns of one class can also be friend functions of another class
* Defined using scope resolution operator.

(Eg)
```
# include <iostream.h>
Class base2; // forward declaration.

Class base1
{
float a;
public:
void getdata ()
{
cout << "Enter A\n";
cin>> a;
}
void mean (base1, base2);
};

Class base2
{
float b;
public:
void getdata()
{
cout << "Enter B\n";
cin>> b;
}
friend void base1::mean(base1,base2);
};
void base1::mean( base1 ob1, base2 ob2)
{
float c = (ob1.a + ob2.b)/2;
cout << "Mean value : "<<c;
}
```

```
void main()
{
base1 obj1;
Obj1. getdata();
base2 obj2;
Obj2. getdata();
Obj1. mean (obj1, obj2);
}
```

(iv) A class being friend with another class (Friend class)

- A class can also be made a friend of another class
  - → Here all member functions of the class is the friend of another class

Syntax:

```
Class z
{
friend class x; // all member functions of x are friends to z.
};
```

(Eg)

```
Class accountant;
class employee
 {
 int income1;
 int income2;
 public:
 void setdata (int in1, int in2)
  {
  income1 = in1;
  income2 = in2;
  }
 friend class accountant;
};
class accountant
{
public:
int total (employee e1)
 {
 return e1.income1 + e1.income2;
 }
};
```

```
void main()
{
employee emp;
accountant act;
emp.setdata (1500, 9000);
cout << "Employee total income:"
            << act.total (emp);
};
```

↓ can access private data member of emp.

Note:

* Friendship is not mutual by default.
  - Here accountant is declared as friend of employee
  - But employee cannot access private members of accountant