**C++ Friend Functions**

A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions.

A friend can be a function, function template, or member function, or a class or class template, in which case the entire class and all of its members are friends.

The function is not in the scope of the class to which it has been declared as a friend. It cannot be called using the object as it is not in the scope of that class. It can be invoked like a normal function without using the object.

**Characteristics of a Friend function:**

* The function is not in the scope of the class to which it has been declared as a friend.
* It cannot be called using the object as it is not in the scope of that class.
* It can be invoked like a normal function without using the object.
* It cannot access the member names directly and has to use an object name and dot membership operator with the member name.
* It can be declared either in the private or the public part.
* Friend function can access data from different classes.
* The parameters of Friend function are object parameters.

To declare a function as a friend of a class, precede the function prototype in the class definition with keyword **friend** as follows −

class Box {

double width;

public:

double length;

friend void printWidth( Box box );

void setWidth( double wid );

};

To declare all member functions of class ClassTwo as friends of class ClassOne, place a following declaration in the definition of class ClassOne −

friend class ClassTwo;

Consider the following program −

#include <iostream>

using namespace std;

class Box {

double width;

public:

friend void printWidth( Box box );

void setWidth( double wid );

};

// Member function definition

void Box::setWidth( double wid ) {

width = wid;

}

// Note: printWidth() is not a member function of any class.

void printWidth( Box box ) {

/\* Because printWidth() is a friend of Box, it can

directly access any member of this class \*/

cout << "Width of box : " << box.width <<endl;

}

// Main function for the program

int main() {

Box box;

// set box width without member function

box.setWidth(10.0);

// Use friend function to print the wdith.

printWidth( box );

return 0;

}

When the above code is compiled and executed, it produces the following result −

Width of box : 10

# C++ friend Function and friend Classes

One of the important concepts of OOP is data hiding, i.e., a [nonmember function](https://www.programiz.com/cpp-programming/object-class) cannot access an object's private or protected data.

But, sometimes this restriction may force programmer to write long and complex codes. So, there is mechanism built in C++ programming to access private or protected data from non-member functions.

This is done using a friend function or/and a friend class.

## friend Function in C++

If a function is defined as a friend function then, the private and protected data of a class can be accessed using the [function](https://www.programiz.com/cpp-programming/function).

The complier knows a given function is a friend function by the use of the keyword **friend**.

For accessing the data, the declaration of a friend function should be made inside the body of the class (can be anywhere inside class either in private or public section) starting with keyword friend.

### Declaration of friend function in C++

class class\_name

{

... .. ...

friend return\_type function\_name(argument/s);

... .. ...

}

Now, you can define the friend function as a normal function to access the data of the class. No friend keyword is used in the definition.

class className

{

... .. ...

friend return\_type functionName(argument/s);

... .. ...

}

return\_type functionName(argument/s)

{

... .. ...

// Private and protected data of className can be accessed from

// this function because it is a friend function of className.

... .. ...

}

### Example 1: Working of friend Function

/\* C++ program to demonstrate the working of friend function.\*/

#include <iostream>

using namespace std;

class Distance

{

private:

int meter;

public:

Distance(): meter(0) { }

//friend function

friend int addFive(Distance);

};

// friend function definition

int addFive(Distance d)

{

//accessing private data from non-member function

d.meter += 5;

return d.meter;

}

int main()

{

Distance D;

cout<<"Distance: "<< addFive(D);

return 0;

}

**Output**

Distance: 5

Here, friend function addFive() is declared inside Distance class. So, the private data meter can be accessed from this function.

Though this example gives you an idea about the concept of a friend function, it doesn't give show you any meaningful use.

A more meaningful use would to when you need to operate on objects of two different classes. That's when the friend function can be very helpful.

You can definitely operate on two objects of different classes without using the friend function but the program will be long, complex and hard to understand.

### Example 2: Addition of members of two different classes using friend Function

#include <iostream>

using namespace std;

// forward declaration

class B;

class A {

private:

int numA;

public:

A(): numA(12) { }

// friend function declaration

friend int add(A, B);

};

class B {

private:

int numB;

public:

B(): numB(1) { }

// friend function declaration

friend int add(A , B);

};

// Function add() is the friend function of classes A and B

// that accesses the member variables numA and numB

int add(A objectA, B objectB)

{

return (objectA.numA + objectB.numB);

}

int main()

{

A objectA;

B objectB;

cout<<"Sum: "<< add(objectA, objectB);

return 0;

}

**Output**

Sum: 13

In this program, classes A and B have declared add() as a friend function. Thus, this function can access private data of both class.

Here, add() function adds the private data numA and numB of two objects objectA and objectB, and returns it to the main function.

To make this program work properly, a forward declaration of a class class B should be made as shown in the above example.

This is because class B is referenced within the class A using code: friend int add(A , B);.

## friend Class in C++ Programming

Similarly, like a friend function, a class can also be made a friend of another class using keyword friend. For example:

... .. ...

class B;

class A

{

// class B is a friend class of class A

friend class B;

... .. ...

}

class B

{

... .. ...

}

When a class is made a friend class, all the member functions of that class becomes friend functions.

In this program, all member functions of class B will be friend functions of class A. Thus, any member function of class B can access the private and protected data of class A. But, member functions of class A cannot access the data of class B.

Remember, friend relation in C++ is only granted, not taken.