C++ Friend function

If a function is defined as a friend function in C++, then the protected and private data of a class can be accessed using the function.

By using the keyword friend compiler knows the given function is a friend function.

For accessing the data, the declaration of a friend function should be done inside the body of a class starting with the keyword friend.

Declaration of friend function in C++

```
1. class class_name
2. {
3.     friend data_type function_name(argument/s);          // syntax of friend func
   tion.
4. };
```

In the above declaration, the friend function is preceded by the keyword friend. The function can be defined anywhere in the program like a normal C++ function. The function definition does not use either the keyword **friend or scope resolution operator**.

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.

C++ friend function Example

Let's see the simple example of C++ friend function used to print the length of a box.

```
1. #include <iostream>
2. using namespace std;
3. class Box
```

```
4.  {
5.      private:
6.          int length;
7.      public:
8.          Box(): length(0) { }
9.          friend int printLength(Box); //friend function
10. };
11. int printLength(Box b)
12. {
13.    b.length += 10;
14.     return b.length;
15. }
16. int main()
17. {
18.     Box b;
19.     cout<<"Length of box: "<< printLength(b)<<endl;
20.     return 0;
21. }
```

Output:

```
Length of box: 10
```