**Friend function**

Data hiding is a fundamental concept of object-oriented programming. It restricts the access of private members from outside of the class.

Similarly, protected members can only be accessed by derived classes and are inaccessible from outside. For example,

class MyClass {

private:

int member1;

}

int main() {

MyClass obj;

// Error! Cannot access private members from here.

obj.member1 = 5;

}

However, there is a feature in C++ called **friend functions** that break this rule and allow us to access member functions from outside the class.

A **friend function** can access the **private** and **protected** data of a class. We declare a friend function using the friend keyword inside the body of the class.

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 function.
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’s 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

**Let's see a simple example when the function is friendly to two classes.**

1. #include <iostream>
2. **using** **namespace** std;
3. **class** B;          // forward declarartion.
4. **class** A
5. {
6. **int** x;
7. **public**:
8. **void** setdata(**int** i)
9. {
10. x=i;
11. }
12. **friend** **void** min(A,B);         // friend function.
13. };
14. **class** B
15. {
16. **int** y;
17. **public**:
18. **void** setdata(**int** i)
19. {
20. y=i;
21. }
22. **friend** **void** min(A,B);                    // friend function
23. };
24. **void** min(A a,B b)
25. {
26. **if**(a.x<=b.y)
27. cout << a.x << endl;
28. **else**
29. cout << b.y << endl;
30. }
31. **int** main()
32. {
33. A a;
34. B b;
35. a.setdata(10);
36. b.setdata(20);
37. min(a,b);
38. **return** 0;
39. }

**Output:**

10

In the above example, min() function is friendly to two classes, i.e., the min() function can access the private members of both the classes A and B.

**C++ Friend class**

A friend class can access both private and protected members of the class in which it has been declared as friend.

We can also use a friend Class in C++ using the friend keyword. For example,

class ClassB;

class ClassA {

// ClassB is a friend class of ClassA

friend class ClassB;

... .. ...

}

class ClassB {

... .. ...

}

When a class is declared a friend class, all the member functions of the friend class become friend functions.

Since ClassB is a friend class, we can access all members of ClassA from inside ClassB.

**Let's see a simple example of a friend class.**

1. #include <iostream>
3. **using** **namespace** std;
4. **class** A
5. {
6. **int** x =5;
7. **friend** **class** B;           // friend class.
8. };
9. **class** B
10. {
11. **public**:
12. **void** display(A &a)
13. {
14. cout<<"value of x is : "<<a.x;
15. }
16. };
17. **int** main()
18. {
19. A a;
20. B b;
21. b.display(a);
22. **return** 0;
23. }

**Output:**

value of x is : 5

In the above example, class B is declared as a friend inside the class A. Therefore, B is a friend of class A. Class B can access the private members of class A.

Example 2

// C++ program to demonstrate the working of friend class

#include <iostream>

using namespace std;

// forward declaration

class ClassB;

class ClassA {

private:

int numA;

// friend class declaration

friend class ClassB;

public:

// constructor to initialize numA to 12

ClassA() : numA(12) {}

};

class ClassB {

private:

int numB;

public:

// constructor to initialize numB to 1

ClassB() : numB(1) {}

// member function to add numA from ClassA and numB from ClassB

int add()

{

ClassA objectA;

return objectA.numA + numB;

}

};

int main() {

ClassB objectB;

cout << "Sum: " << objectB.add();

return 0;

}

**Output**

Sum: 13

Here, ClassB is a friend class of ClassA. So, ClassB has access to the members of classA.

In ClassB, we have created a function add() that returns the sum of numA and numB.

Since ClassB is a friend class, we can create objects of ClassA inside of ClassB.