Friend Function

By – Mohammad Imran

Friend Function

Introduction

In C++, global functions cannot access the private members of a class. However, sometimes we need a global function to access private members to perform certain operations, without defining it inside the class. This is where friend functions come into play.

A friend function in C++ is a non-member function that is granted access to the private, protected, and public members of a class when declared as a friend. This allows the function to perform operations that require access to private members, without having to define the function inside the class.

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.

Friend Function

Characteristics or Features

- ✓ A global function or a member function of another class, both can be declared as a friend function.
- ✓ A friend function in C++ should not be in the scope of the class of which it is supposed to be the friend. This means that the function which is declared as a friend should not be a member of the same class.
- ✓ A friend function in C++ can be declared anywhere in the class, that is, in the public section or the private section of the class.

Friend Function

Characteristics or Features

- ✓ The friend function in C++ can be called (invoked) just like a normal function using any instance of any class (object).
- ✓ A friend function in C++ cannot directly access the protected or private data members of the class. It is required to use an object (instance of that class) and then use the dot operator (.) to access the data members.
- ✓ Friend functionality in C++ is not restricted to only one class. That is, it can be a friend to many classes.
- ✓ Friend functions in C++ can use objects (instance of a class) as arguments.

Friend Function

Benefits

- ✓ Enhanced encapsulation control by selectively allowing external access to private members
- ✓ Improved code organization by grouping related functions with a class
- ✓ Efficient access to private data without getter/setter methods
- ✓ Simplified and more readable code for certain operations
- ✓ Flexible design for custom operators or type conversions
- ✓ Easier integration with third-party libraries or external code
- ✓ Reduced overhead compared to making all members public
- ✓ Enhanced maintainability by centralizing access permissions

By – Mohammad Imran

Friend Function

Application

Operator Overloading: Friend functions are commonly used for overloading operators, such as `+`, `-`, `==`, etc., to provide custom behaviors for user-defined types, making complex operations intuitive and efficient.

Accessing Private Members: Friend functions allow controlled access to private and protected members of a class, making them useful for encapsulation while still permitting specific external functions or classes to work closely with an object's internals.

Friend Function

Application

Type Conversion: Friend functions can define custom type conversion operators (typecasting) between user-defined types, enabling seamless and intuitive type conversions when working with objects of different classes.

Mathematical Classes: In mathematical libraries or classes representing mathematical concepts (e.g., vectors, matrices), friend functions can facilitate advanced mathematical operations that require direct access to private data, ensuring efficiency and code clarity.

Friend Function

Application

Serialization: Friend functions are valuable for serializing and deserializing objects, especially in applications that involve saving or loading complex object structures to/from files or across networks, where direct access to object internals is necessary for efficient data handling.

Friend Function

Advantages

- ✓ The friend function allows the programmer to generate more efficient codes.
- ✓ It allows the sharing of private class information by a non-member function.
- ✓ It accesses the non-public members of a class easily.
- ✓ It is widely used in cases when two or more classes contain the interrelated members relative to other parts of the program.
- ✓ It allows additional functionality that is not used by the class commonly.

Friend Function

Declaration

```
class class_Name
{
  private:
   int x;
   friend return_type function_Name(Argument_1,...,Argument_5);
}
```

By – Mohammad Imran

Friend Function

Types

A friend function can be:

- ✓ **A Global Function**
- ✓ **A Member Function of Another Class**

Global Friend Function

We can declare any global function as a friend function. The following example demonstrates how to declare a global function as a friend function in C++:

Global Friend Function

```
class FriendFunction
{
  private:
   int x;
  protected:
   int y;
  public:
   void display()
   {
       cout << x << " " << y << endl;
   }
};
```

```
//Global Function
void access()
{
   cout << x << " " << y;
}
```

Note: x and y is not accessible because of private member.

By – Mohammad Imran

Global Friend Function

Example - 1

```
#include<bits/stdc++.h>
using namespace std;
class MyClass
{
   private:
    int member1;
   public:

    friend void show(MyClass);

    void display()
    {
       Cout << "Hello!" << endl;
    }
};
```

```
void show(MyClass mc)
{
   mc.member1 = 100;
   cout<<mc.member1<<endl;
}

int main()
{
   MyClass obj;
   obj.display();
   show(obj);
   return 0;
}
```

By – Mohammad Imran

Friend Function

Another Class Function as Friend

We can also declare a member function of another class as a friend function in C++. The following example demonstrates how to use a member function of another class as a friend function in C++:

Friend Function

Another Class Function as Friend

Example - 2

```cpp
#include <bits/stdc++.h>
using namespace std;

class B;
class A
{
    int x;
   public:
    void setdata(int i)
    {
        x = i;
    }
    friend void min(A, B);
};
```

```cpp
class B
{
    int y;
   public:
    void setdata(int i)
    {
        y = i;
    }
    friend void min(A, B);
};
void min(A a, B b)
{
    if(a.x <= b.y)
      cout << a.x << endl;
    else
      cout << b.y << endl;
}
```

By – Mohammad Imran

Friend Function

Another Class Function as Friend

Example - 2

```
int main()
{
   A a;
   B b;
   a.setdata(10);
   b.setdata(20);
   min(a,b);
   return 0;
}
```

OUTPUT

```
10
```

By – Mohammad Imran

Friend Class

Just like a friend function, a particular class can also have a friend class. A friend class shares the same privilege, i.e., it can access the private and protected members of the class whose friend it has been declared. This means that all the functions declared inside the friend class will also be able to access the private and protected members of the class. Before learning more about friend classes, we will first look at how to declare a class as a friend class for another class.

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

Friend Class

Introduction

Friend Class is a class that can access both private and protected variables of the class in which it is declared as a friend, just like a friend function. Classes declared as friends to any other class will have all the member functions as friend functions to the friend class. Friend functions are used to link both these classes.

Like a friend function **friend** keyword is used to make a class as friend of another class.

By – Mohammad Imran

Friend Class

Syntax

To declare a class as a friend class in C++, it needs to be preceded by the keyword "**friend**" inside the body of the class.

```
class One
{
   Statements;
   friend class Two;
};

class Two
{
   <few lines of code>
};
```

By – Mohammad Imran

Friend Class

Key Points

To make you understand in detail:

- ✓ If class A is a friend of class B, then class B is not a friend of class A.
- ✓ Also, if class A is a friend of class B, and then class B is a friend of class C, class A is not a friend of class C.
- ✓ If Base class is a friend of class X, subclass Derived is not a friend of class X; and if class X is a friend of class Base, class X is not a friend of subclass Derived.

Friend Class

Example - 3

```
#include <iostream>
using namespace std;
class A
{
    int x = 5;
    friend class B;
};
class B
{
   public:
    void display(A &a)
    {
        cout << "X = " << a.x;
    }
};
```

OUTPUT

```
X = 5
```

```
int main()
{
    A a;
    B b;
    b.display(a);
    return 0;
}
```

By – Mohammad Imran

Friend Class

Example - 4

```
#include <iostream>
using namespace std;
class Shape;
class Square
{
  private:
   int side;
  public:
   void set_values (int s)
   {
      side = s;
   }
   friend class Shape;
};
```

```
class Shape
{
  public:
   void print_area (Square& s)
   {
      int area = s.side*s.side;
      cout<<"Area of Square = " <<
      area << endl;
   }
};
```

By – Mohammad Imran

Friend Class

Example - 4

```
int main ()
{
    Square s;
    s.set_values(5);
    Shape sh;
    sh.print_area(s);
    return 0;
}
```

OUTPUT

```
Area of Square = 25
```

By – Mohammad Imran

Friend Class

Two Class as Friend

We can make two class a friend to access private and protected member of those classes in each other. Suppose class A has two member private and protected so class B can access those members and if class B has private and protected member then class A can access those member. To do this we can make both class as friend.

Friend Class

Two Class as Friend | **Syntax**

Declaring two class as friend:

```
class A
{
   friend class B;
};

class B
{
   friend class A;
};
```

Friend Class | Two Class as Friend | Example - 5

```
#include <iostream>
using namespace std;
class B;
class A
{
    int data;
   public:

    friend class B;

    void acceptA(int d)
    {
        data = d;
    }
    void get_data_B(B objb);
};
```

```
class B
{
    int dataB;
   public:
    friend class A;
    void acceptB(int d)
    {
        dataB = d;
    }
    void get_data_A(A obja)
    {
        cout << "Data of A is: "<<
        obja.data << endl;
    }
};
```

By – Mohammad Imran

Friend Class | Two Class as Friend | Example - 5

```
void A::get_data_B(B objb)
{
   cout << "Data of B is: "<< objb.dataB;
}

int main()
{
   A a1;
   a1.acceptA(100);
   B b1;
   b1.acceptB(200);
   b1.get_data_A(a1);
   a1.get_data_B(b1);
   return 0;
}
```

OUTPUT

```
Data of A = 100
Data of B = 200
```

By – Mohammad Imran

Passing Object as an Argument

In C++ programming language, we can also pass an object as an argument within the member function of class.

This is useful, when we want to initialize all data members of an object with another object, we can pass objects and assign the values of supplied object to the current object. For complex or large projects, we need to use objects as an argument or parameter.

Passing Object as an Argument

By Value

The objects of a class can be passed as arguments to member functions as well as non-members functions either by value or by reference.

Passing Object as an Argument

By Value

Program - 6

```
#include <iostream>
using namespace std;
class Demo
{
   private:
    int a;
   public:
    void set(int x)
    {
        a = x;
    }
    void sum(Demo ob1, Demo ob2)
    {
        a = ob1.a + ob2.a;
    }
```

```
    void print()
    {
        cout << "A = " << a <<
        endl;
    }
};
```

By – Mohammad Imran

Passing Object as an Argument

Program - 6

```
int main()
{
   Demo d1;
   Demo d2;
   Demo d3;
   d1.set(10);
   d2.set(20);
   d3.sum(d1,d2);
   d1.print();
   d2.print();
   d3.print();
   return 0;
}
```

OUTPUT

```
A = 10
A = 20
A = 30
```

By – Mohammad Imran

Passing Object as an Argument

By Reference

Like pass by reference variable in function we can also pass reference to an object to a function. Using pass by reference object to a function we can change the value of variable in both section using single object.

In C++ programming, many times we need to reflect the changes in actual parameters. So, here we need to pass an object by reference.

Passing Object as an Argument

By Reference

Program - 7

```
#include <iostream>
using namespace std;
class ByRef
{
    float val;
   public:
    void get_input(void);
    friend void modify(ByRef &,float);
    void display(void);
};
```

By – Mohammad Imran

Passing Object as an Argument

By Reference

Program - 7

```
void ByRef::get_input()
{
   cout<<"Enter Value : ";
   cin>>val;
}
void ByRef::display()
{
   cout<<"\nValue = "<<val<<endl;
}

void modify(ByRef &t,float new_val)
{
   t.val = new_val;
   cout<<"\tNew Val = "<<t.val<<endl;
}
```

By – Mohammad Imran

Passing Object as an Argument

By Reference

Program - 7

```
int main()
{
   float value;
   ByRef t1;
   t1.get_input();
   cout <<"\n\n--BEFORE MODIFICATION--";
   t1.display();
   cout << "\n\n\tEnter new value :  ";
   cin >> value;
   modify(t1,value);
   cout<<"\n\n\n--AFTER MODIFICATION--";
   t1.display();
}
```

OUTPUT

```
Enter value = 100
Before Modification
Value = 100
Enter new value = 50
After Modification
Value = 50
```

By – Mohammad Imran

Returning Object to a Function

As we can return a value to function in return function we can also return an object to the return function of type class and can use the variable of the class using that object.

Returning value will be stored in the function of type class.

By – Mohammad Imran

Returning Object to a Function

Program - 8

```
#include <iostream>
using namespace std;
class Student
{
  public:
   int val1,val2,res;
   Student input()
   {
      Student obj;
      obj.val1 = 20;
      return obj;
   }
   void display(Student obj)
   {
      cout << "Value 1 = " << obj.val1v<<vendl;
   }
};
```

```
int main()
{
   Student s;
   s = s.input ();
   s.display(s);
   return 0;
}
```

OUTPUT

```
Value 1 = 20
```

By – Mohammad Imran

Returning Object to a Function

Program - 9

```
#include <iostream>
using namespace std;
class Student
{
   int roll;
   string name;
  public:
   Student input()
   {
      Student st;
      st.roll = 111;
      st.name = "Smith";
      return st;
   }
   void display(Student obj)
   {
      cout << "Roll = " << obj.roll << endl;
      cout << "Name = " << obj.name << endl;
   }
};

int main()
{
      Student s;
      s = s.input();
      s.display(s);
      return 0;
}
```

OUTPUT

```
Roll = 111
Name = Smith
```

By – Mohammad Imran

Coding Questions

By – Mohammad Imran

Question - 1

Write a C++ program to demonstrate to creating array of object and provide different value of same variable using different object and display the value accordingly.

Click here to see code

By – Mohammad Imran

Question - 2

Write a C++ program to demonstrate to creating a pointer to an object and also create two function a simple function and an argument function. Call both function using the pointer object.

Click here to see code

By – Mohammad Imran

Question - 3

Write a C++ program to implement a class called **BankAccount** that has private member variables for account number and balance. Include member functions to deposit and withdraw money from the account.

Sample Output

```
Account Number: SB1001
Balance: 1000.00
Deposit: 2000.00
Balance: 3000.00
Withdraw: 500.00
Balance: 2500.00
```

Click here to see code

By – Mohammad Imran

Question - 4

21 – 22 - CHO Practice Problem – P2

Write a C++ program to find the palindrome words and their number of occurrence in a sentence.

Click here to see code

By – Mohammad Imran