FAST NATIONAL UNIVERSTIY OF COMPUTER AND EMERGING SCIENCES, PESHAWAR

DEPARTMENT OF COMPUTER SCIENCE

CL217 – OBJECT ORIENTED PROGRAMMING LAB

Friend Function and Friend class in C++

LAB MANUAL # 11

Instructor: Fariba Laiq

SEMESTER SPRING 2023

Friend Functions in C++

Friend functions of the class are granted permission to access private and protected members of the class in C++. They are defined globally outside the class scope. Friend functions are not member functions of the class. So, what exactly is the friend function?

A friend function in C++ is a function that is declared outside a class but is capable of accessing the private and protected members of the class. There could be situations in programming wherein we want two classes to share their members. These members may be data members, class functions. In such cases, we make the desired function, a friend to both these classes which will allow accessing private and protected data of members of the class.

Generally, non-member functions cannot access the private members of a particular class. Once declared as a friend function, the function is able to access the private and the protected members of these classes.

```
class class_name
{
  friend data_type function_name(arguments/s); //syntax of friend function.
};
```

Characteristics of Friend Function in C++

- The function is not in the 'scope' of the class to which it has been declared a friend.
- Friend functionality is not restricted to only one class
- Friend functions can be a member of a class or a function that is declared outside the scope of class.
- It cannot be invoked using the object as it is not in the scope of that class.
- We can invoke it like any normal function of the class.
- Friend functions have objects as arguments.
- It cannot access the member names directly and has to use dot membership operator and use an object name with the member name.
- We can declare it either in the 'public' or the 'private' section.

C++ program to demonstrate the working of friend function

```
#include <iostream>
using namespace std;
class Distance {
   private:
      int meter;
      // friend function
      friend int addFive(Distance);
   public:
      Distance() : meter(0) {}
};

// friend function definition
int addFive(Distance d) {

   //accessing private members from the friend function
   d.meter += 5;
   return d.meter;
}

int main() {
   Distance D;
   cout << "Distance: " << addFive(D);
   return 0;
}
```

Output:

Distance: 5

Accessing private data members of a class from the member function of another class using friend function

```
#include <iostream>
using namespace std;
//forward declaration
class Distance;
class Add {
	public:
		int addFive(Distance d);
};
class Distance {
	private:
		int meter;
		// friend function
	public:
		int getDistance()
		{
			return meter;
		}
		Distance(int meter) {
			this->meter=meter;
		}
		friend int Add::addFive(Distance d);
};

int Add::addFive(Distance d) {
	//accessing private members from the friend function
	d.meter += 5;
	return d.meter;
}

int main() {
	Distance d(5);
	cout<<"Distance before: "<<d.getDistance()<<endl;
	Add a;
	cout<<"Distance now: "<<a.addFive(d);
	return 0;
}
```

Access members of two different classes using friend functions

```
#include <iostream>
using namespace std;
// forward declaration
class ClassB;
class ClassA {
        private:
                int numA;
        public:
                ClassA(int numA)
                {
                        this->numA=numA;
                }
                // friend function declaration
                friend int add(ClassA, ClassB);
};
class ClassB {
        private:
                int numB;
        public:
                ClassB(int numB)
                {
                        this->numB=numb;
                }

                // friend function declaration
                friend int add(ClassA, ClassB);
};
// access members of both classes
int add(ClassA objectA, ClassB objectB) {
        return (objectA.numA + objectB.numB);
}
int main() {
        ClassA objectA(4);
        ClassB object(6);
        cout << "Sum: " << add(objectA, objectB);
        return 0;
}
```

Output:

Sum: 10

Friend class

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

When a class is declared a friend class, all the member functions of the friend class become friend functions. If ClassB is a friend class of ClassA. So, ClassB has access to the members of classA, but not the other way around. Now in the following code, if we want the class Operations to access all the member functions of the Distance class, instead explicitly declaring all the functions of class Distance as friends, we just make the whole class as a friend.

Syntax:

```
friend class class_name;
```

It includes two classes, "Distance" and "Operations". The "Distance" class has a private member "meter" representing distance in meters and a public function "getDistance()" to return the distance. The "Operations" class has two friend functions, "addFive()" and "subFive()", which can access the private member of the "Distance" class. The main function creates an object of the "Distance" class and initializes it with a value of 5. It then creates an object of the "Operations" class and calls the "addFive()" and "subFive()" functions to increment and decrement the distance by 5, respectively.

Forward declaration is a technique in C++ that allows declaring the name of a class, function, or object before defining it. It informs the compiler that the name is a valid entity, and the definition will be provided later in the code.

```
#include <iostream>
using namespace std;
//forward declaration
class Distance;
class Operations {
	public:
		int addFive(Distance &d);
		int subFive(Distance &d);
};
class Distance {
	private:
		int meter;
		// friend function
	public:
		int getDistance()
		{
			return meter;
		}
		Distance(int meter) {
			this->meter=meter;
		}
		friend class Operations;
};

int Operations::addFive(Distance &d) {
	//accessing private members from the friend function
	d.meter += 5;
	return d.meter;
}

int Operations::subFive(Distance &d) {
	//accessing private members from the friend function
	d.meter -= 5;
	return d.meter;
}
int main() {
	Distance d(5);
	cout<<"Distance before: "<<d.getDistance()<<endl;
	Operations a;
	cout<<"Distance now by adding 5: "<<a.addFive(d)<<endl;
	cout<<"Distance now by subtracting 5: "<<a.subFive(d);
}
```

Output:

Distance before: 5

Distance now by adding 5: 10

Distance now by subtracting 5: 5