Lecture 7- Friend Functions

19CSE201 Advanced Programming

Friend function

- If two classes want to use a function to operate on the objects of both the classes, then a common function can be made friendly with both the classes .
- Function need not be a member of both the classes.

Declare friend function

```
class ABC
{
---------
---------
public:
--------
----------
friend void xyz(void);
};
```

- Function can be defined anywhere in the program
- **Since it is not a member of the class, no need to use :: to define it outside the class.**
- It is not in the scope of the class to which it has been declared as friend.
- Since it is not in the scope of the class, it cannot be called using the object of that class.
- **It can be invoked like a normal function without the help of any object.**
- **Inside friend function, to access the data members, it has to use objects.**
- Usually this function has objects as arguments.

Function friend of one class

```
class sample
{
    int a;
    int b;
    public:
    void setvalue( ) { a=25;b=40;}
    friend float mean( sample s);
} ;
float mean (sample s)
{
return (float(s.a+s.b)/2.0);
}
```

```
int main ( )
{
sample x;
x . setvalue( );
cout<<"meanvalue="<<mean(x)<<endl;
return(0);
}
```

- If the function is friend of one class, then there is no problem in defining the friend function inside the class itself.
- But if the function is friend of two classes, it can not be defined inside any of those classes.
- It has to be defined outside the class
- So always try to define friend function outside the class, because it is just a friend function not a member function.

Function friend of two classes

```
class sample; //forward declaration
class item
{
int x;
public:
void setvalue(int a) { x=a; }
friend void max (item, sample);
};
class sample
{
int a;
public:
void setvalue( int b) {a=b; }
friend void max(item, sample);
};
```

```
void max(item m, sample s)
{
if(m . x >= s.a)
cout<<m.x;
else
cout<< s.a;
}
int main( )
{
item I;
I . setvalue( 10);
sample S;
S.setvalue(20);
max( I, S );
}
```

Swapping private data of classes

```
class class2;   // forward declaration
class class1
{
int value1;
public:
void setdata( int a) { value1=a; }
void display(void) { cout<<value1<<endl;  }
friend void exchange ( class1 &, class2 &);
};
class class2
{
int value2;
public:
void setdata( int a) { value2=a; }
void display(void) { cout<<value2<<endl;  }
friend void exchange(class1 & , class2 &);
};
void exchange ( class1 &x, class2 &y)
{
int temp=x. value1;
x. value1=y.value2;
y.value2=temp;
}
int main( )
{
class1 c1;
class2 c2;
c1.setdata(100);
c2.setdata(200);
cout<<"values before exchange:"<<endl;
c1.display( );
c2.display( );
exchange (c1,c2);
cout<<"values after exchange :"<< endl;
c1. display ( );
c2. display ( );
}
```

- If we are trying to update the private data member using friend function, then it should be using reference.

Questions

1. Create 2 classes complex1, complex2 with members real and imaginary. Include member function to read the values for data members. Include friend function to add object of both the classes.(add real nos of both the class and add imaginary nos of both the class)
2. Create two classes account1 and account2 with data member balance, member functions read() and disp() which read and display the values. Include friend function transfer which transfers the given amount from account1 to account2.
3. Create an employee class with empname, age, salary as data member and student class with studname, age and cgpa as data member. Include sufficient fns to read and display the values. Include a friend function which checks whether the name of employee is same as the name of student.
4. Create an employee class with empname, age, salary as data member and student class with studname, age and cgpa as data member. Include sufficient fns to read and display the values. Include a friend function which finds the age difference between employee and student.

Public member function of one class as a friend of another class

```
class X
{
public: int fun1(){----}
};
class Y
{-----
 friend int X::fun1();
-----
};
```

1. Can we make private function of one class as friend of another class?

   No

2. Can we access friend function from a member function?

Yes

3. If one function is friend of two classes, then can we give that friend function's definition inside any of those class?

No

Restrictions of friend function

- Derived class does not inherit friend function, because its not a member function
- Friend function cant be static
- ‘this’ pointer cant be used inside friend function, because its not a member function

‘this’ pointer

- ‘this’ pointer points to the current object ie to the invoking object
- A. max(); - ‘this’ pointer points to the object A
- ‘this pointer has the address of invoking object
- This pointer is automatically passed to the member function when it is called
- The pointer acts as an implicit argument to all the member function, for e.g.

```
class ABC
{
int a ;
-----
};
```

- The private variable ‘a’ can be used directly inside a member function, like a=123;
- But actually it will be assigned like: this → a = 123

- Used explicitly in the case of returning objects
- Used explicitly in operator overloading

```
class Distance
{ int feet, inches;
public:
 Distance add(Distance a, distance b)
{
inches=a.inches+b.inches;
feet= a.feet+b.feet;
return *this;
}
void disp(){cout<<feet<<inches;}
};
main()
{
Distance d1, d2, d3,d4;
d4=d3.add(d1,d2);
d4.disp();
}
```

- return *this – to return the value of that object
- return this- to return the address of the object

- What is the difference between static function and normal function?
- Static function can not access nor mal data members of the class
- Static function wont have this pointer

Restrictions

- This pointer cant be used in friend function
- Static member function does not have this pointer