University of Southern California

Viterbi School of Engineering

Software Design

OOP: Friendship

Reference: Online Resources

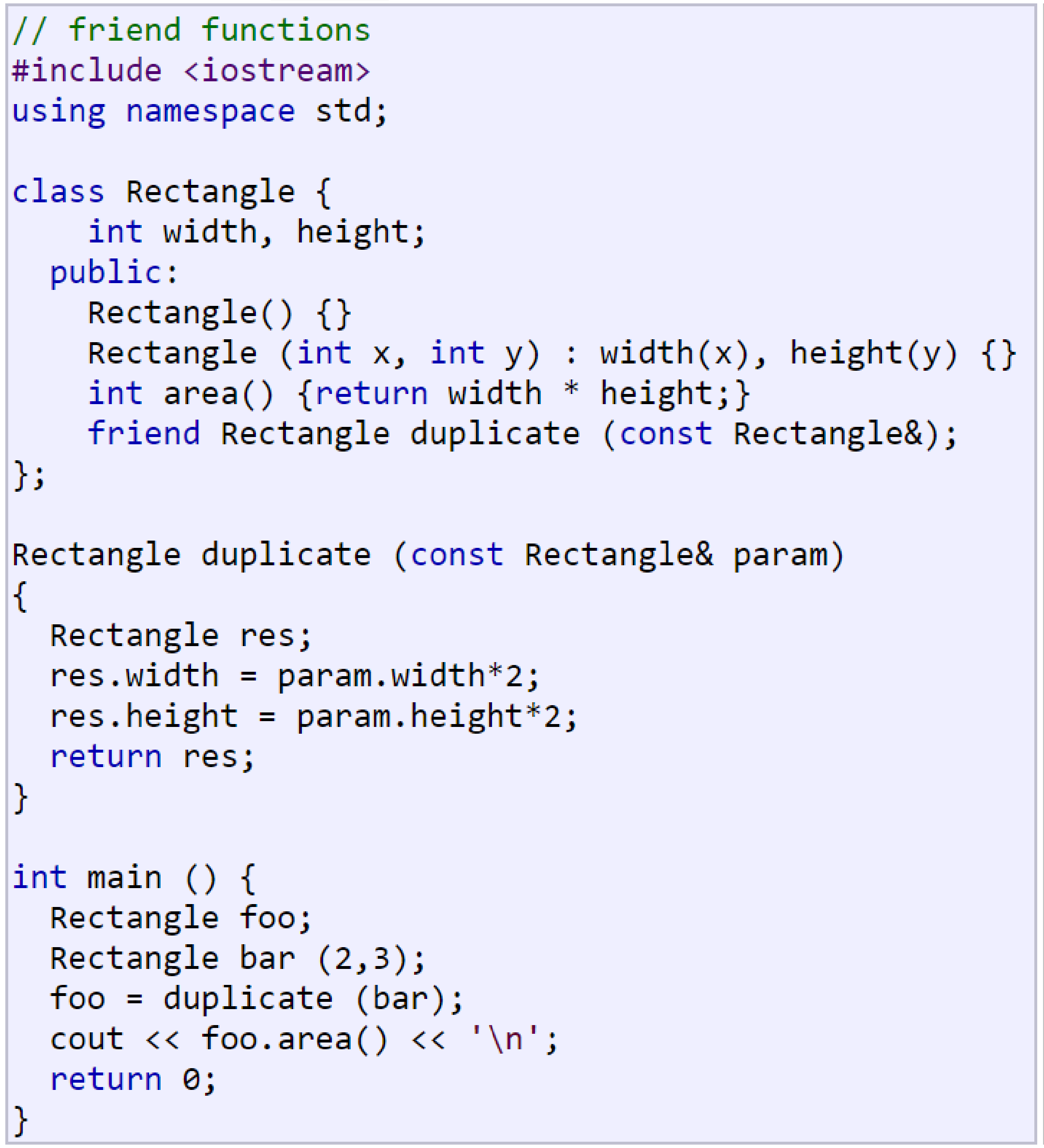
Friend Function

- A non-member function can access the private and protected members of a class if it is declared a *friend* of that class
- That is done by including a declaration of this external function within the class, and preceding it with the keyword **friend**

```cpp
/* C++ program to demonstrate the working of friend function.*/
#include <iostream>
using namespace std;

class Distance
{
    private:
        int meter;
    public:
        Distance(): meter(0) { }
        //friend function
        friend int addFive(Distance);
};

// friend function definition
int addFive(Distance d)
{
    //accessing private data from non-member function
    d.meter += 5;
    return d.meter;
}

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

Friend Function (cont.)

- **Example:**
  - **The function duplicate is a *friend* of class Rectangle**
  - **Therefore duplicate is able to access the members width and height (which are private) of any object of type Rectangle**



```
// friend functions
#include <iostream>
using namespace std;

class Rectangle {
    int width, height;
  public:
    Rectangle() {}
    Rectangle (int x, int y) : width(x), height(y) {}
    int area() {return width * height;}
    friend Rectangle duplicate (const Rectangle&);
};

Rectangle duplicate (const Rectangle& param)
{
  Rectangle res;
  res.width = param.width*2;
  res.height = param.height*2;
  return res;
}

int main () {
  Rectangle foo;
  Rectangle bar (2,3);
  foo = duplicate (bar);
  cout << foo.area() << '\n';
  return 0;
}
```

Friend Class

- **Similar to a friend function, a friend class is a class whose members have access to the private or protected members of another class**

- Example: class Rectangle is a friend of class Square allowing Rectangle's member functions to access private and protected members of Square. More precisely, Rectangle accesses the member variable side (which is private)
- Note that the declaration of Square at line 4 is necessary as Rectangle needs to access it in its function member

```
// friend class
#include <iostream>
using namespace std;

class Square;

class Rectangle {
    int width, height;
  public:
    int area ()
      {return (width * height);}
    void convert (Square a);
};

class Square {
  friend class Rectangle;
  private:
    int side;
  public:
    Square (int a) : side(a) {}
};

void Rectangle::convert (Square a) {
  width = a.side;
  height = a.side;
}

int main () {
  Rectangle rect;
  Square sqr (4);
  rect.convert(sqr);
  cout << rect.area();
  return 0;
}
```

Observations

- **Question: Why friendship when we have the option of the inheritance?**
  - **Just because you are a friend, it does not mean you inherit from me :D**
- **Friendship cannot be inherited, i.e., if a base has a friend function, that function does not become a friend of the derived classes**
- **Friendship is not transitive**

Not Commutative

Example:

Class DT
friend class VP

/* VP has access to private parts of DT but DT does not have access to private parts of VP */

```
class A {};
class B {};
class C : public A, public B {};

class C {
        //private
        friend class A;
        friend class B;
}
```