July 11th, 2013

1. Friendship in C++:

Private and protected members of a class cannot be accessed from outside the same class in which they are declared. BUT, this rule doesn’t affect *friends*.

**Friend function:**

When we want to declare an external function as friend of a class, thus allowing this function to have access to the private and protected members of this class, we do it by declaring a prototype of this external function within the class, and preceding it with the keyword friend.

// friend functions

#include <iostream>

using namespace std;

class CRectangle {

int width, height;

public:

void set\_values (int, int);

int area () {return (width \* height);}

friend CRectangle duplicate (CRectangle);

};

void CRectangle::set\_values (int a, int b) {

width = a;

height = b;

}

CRectangle duplicate (CRectangle rectparam)

{

CRectangle rectres;

rectres.width = rectparam.width\*2;

rectres.height = rectparam.height\*2;

return (rectres);

}

int main () {

CRectangle rect, rectb;

rect.set\_values (2,3);

rectb = duplicate (rect);

cout << rectb.area();

return 0;

}

Note that neither in the declaration of duplicate() nor in its later use in main() have we considered duplicate a member of class CRectangle. It isn’t! It simply has access to its private and protected members without being a member.

**Friend classes:**

We can also define a class as friend of another one, granting that first class access to the protected and private members of the second one.

// friend class

#include <iostream>

using namespace std;

class CSquare;

class CRectangle {

int width, height;

public:

int area ()

{return (width \* height);}

void convert (CSquare a);

};

class CSquare {

private:

int side;

public:

void set\_side (int a)

{side=a;}

friend class CRectangle;

};

void CRectangle::convert (CSquare a) {

width = a.side;

height = a.side;

}

int main () {

CSquare sqr;

CRectangle rect;

sqr.set\_side(4);

rect.convert(sqr);

cout << rect.area();

return 0;

}

We have declared CRectangle as a friend of CSquare so that CRectangle member functions could have access to the protected and private members of CSquare, more concretely to CSquare::side, which describes the side width of the square.

An empty declaration of class CSquare is necessary because within the declaration of CRectangle we refer to CSquare (as a parameter in convert()). The definition of CSquare is included later, so if we didn’t include a previous empty declaration for CSquare this class would not be visible from within the definition of CRectangle.