Friendship

Friendship

- In principle, private and protected members of a class cannot be accessed from outside the same class in which they are declared. However, this rule does not affect *friends*.

- If 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;
}
```

```
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;
}
```

- **The duplicate function is a friend of CRectangle. From within that function we have been able to access the**
- **members width and height of different objects of type CRectangle, which are private members**.

- **Notice 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.

- **The duplicate function is a friend of CRectangle. From within that function we have been able to access the members width and height of different objects of type CRectangle, which are private members.**

Friend classes

- **Just as we have the possibility to define a friend function, 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;
}
```

```
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 couldhave access to the protected and private members of CSquare, more concretely to CSquare::side, which describes the side width of the square.**