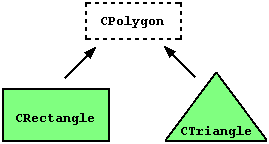
**Inheritance between classes**

Classes in C++ can be extended, creating new classes which retain characteristics of the base class. This process, known as inheritance, involves a *base class* and a *derived class*: The *derived class* inherits the members of the *base class*, on top of which it can add its own members.  
  
For example, let's imagine a series of classes to describe two kinds of polygons: rectangles and triangles. These two polygons have certain common properties, such as the values needed to calculate their areas: they both can be described simply with a height and a width (or base).  
  
This could be represented in the world of classes with a class Polygon from which we would derive the two other ones: Rectangle and Triangle:  
  
   
The Polygon class would contain members that are common for both types of polygon. In our case: width and height. And Rectangle and Triangle would be its derived classes, with specific features that are different from one type of polygon to the other.  
  
Classes that are derived from others inherit all the accessible members of the base class. That means that if a base class includes a member A and we derive a class from it with another member called B, the derived class will contain both member A and member B.  
  
The inheritance relationship of two classes is declared in the derived class. Derived classes definitions use the following syntax:  
  
class derived\_class\_name: public base\_class\_name  
{ /\*...\*/ };  
  
Where derived\_class\_name is the name of the derived class and base\_class\_name is the name of the class on which it is based. The public access specifier may be replaced by any one of the other access specifiers (protected or private). This access specifier limits the most accessible level for the members inherited from the base class: The members with a more accessible level are inherited with this level instead, while the members with an equal or more restrictive access level keep their restrictive level in the derived class.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 | // derived classes  #include <iostream>  using namespace std;  class Polygon {  protected:  int width, height;  public:  void set\_values (int a, int b)  { width=a; height=b;}  };  class Rectangle: public Polygon {  public:  int area ()  { return width \* height; }  };  class Triangle: public Polygon {  public:  int area ()  { return width \* height / 2; }  };    int main () {  Rectangle rect;  Triangle trgl;  rect.set\_values (4,5);  trgl.set\_values (4,5);  cout << rect.area() << '\n';  cout << trgl.area() << '\n';  return 0;  } | 20  10 |  |

The objects of the classes Rectangle and Triangle each contain members inherited from Polygon. These are: width, height and set\_values.  
  
The protected access specifier used in class Polygon is similar to private. Its only difference occurs in fact with inheritance: When a class inherits another one, the members of the derived class can access the protected members inherited from the base class, but not its private members.  
  
By declaring width and height as protected instead of private, these members are also accessible from the derived classes Rectangle and Triangle, instead of just from members of Polygon. If they were public, they could be accessed just from anywhere.