Task 1:

1. Create a pure abstract class *CFigure* which includes two methods: *double Area()*, *double Perimeter()*. Implement a pure virtual destructor. Try to create an instance of the class *CFigure*. What happens?

2. Construct classes *CCircle*, *CRectangle*, and *CSquare*. Decide how to organize the class hierarchy and which members the classes should include. Implement functions *Area* and *Perimeter*. For calculating the area and perimeter of a circle you will need the value of PI = 3.14159. This value is a constant which is shared by all the objects of the class *CCircle*. Arrange a way of knowing how many instances of classes *CCircle*, *CRectangle* and *CSquare* do exist (implement for such a purpose function *Count*, do not separate figures of different types when counting). Redefine the operator > to compare two figures (on the basis of their areas).

3. Implement two global functions (*DisplayAreas* and *DisplayPerimeters*) each of which receives an array of pointers to *CFigure* and an *unsigned* value indicating the total number of pointers in the array. The function *DisplayAreas* displays the area of each figure and the function *DisplayPerimeters* displays perimeters. Test your classes and functions with the following *main* function:

```
int main()
{
      {
            CCircle c1(5); CRectangle r1(2, 3); CSquare s1(4);
            CFigure* pContainer1[] = { &c1, &r1, &s1 } ;
            CFigure::Count();

            CCircle c2(15); CRectangle r2(3, 5);
            CFigure* pContainer2[] = { &c2, &r2 };
            CFigure::Count();

            DisplayAreas(pContainer1, sizeof (pContainer1) /
                         sizeof (*pContainer1));
            DisplayAreas(pContainer2, sizeof (pContainer2) /
                         sizeof (*pContainer2));

            cout << (*pContainer1[0] > *pContainer1[1]) << endl;
            cout << (*pContainer1[1] > *pContainer1[2]) << endl;
            DisplayPerimeters(pContainer1, sizeof (pContainer1) /
                              sizeof (*pContainer1));
            DisplayPerimeters(pContainer2, sizeof (pContainer2) /
                              sizeof (*pContainer2));

            CCircle new_c = c1;
            new_c = c2;
            CFigure::Count();
      }

      CFigure::Count();
      return 0;
}
```