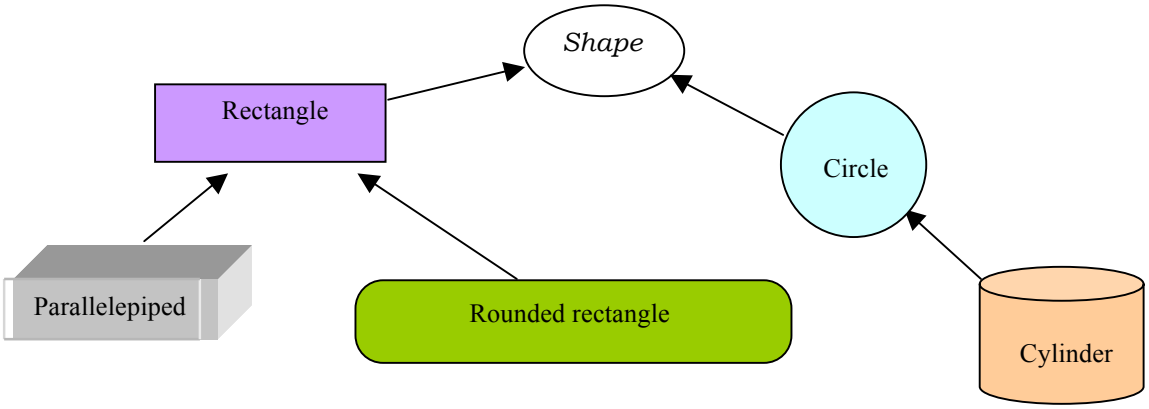
Assignment for the laboratory work № 2.

Develop a class hierarchy of different coloured geometric shapes/figures, starting with the abstract base class *Shape*. This class should contain data members that might be common in derived classes (such as c*olour*). Moreover, the class *Shape* must contain at least two dummy functions declared as pure virtual functions and being intended both for shapes' area calculations and for getting information about colour(e.g., *GetArea( )* and *GetColour( )*). These functions must be overriden in the derived classes that depict 2D and 3D shapes, e.g., *Rectangle*, *Circle*, *RoundedRectangle*, *Cylinder*, *Parallelepiped*, etc. It should be a single indirect inheritance hierarchy such as:



Variable c*olour* should be of **char*** type allocated dynamically, or you may use strings of the class *basic_string* from the header <string>.
In the *main* function, the array of base-class pointers (so-called indirect container) must be created in order to keep pointers to the derived class objects created here statically or dynamically. Initially, all elements of this fixed-size array should be set into NULL.There should also be developed a global function, say *GetData*, that takes this array and its size as parameters and provides polymorphic area calculation of the total area of all shapes in the array, as well as displaying information about their colours. Needless to say that the function *GetData* must not be changed in case of adding some new shape to the array or removing an old one.
Because a number of objects created can exceed the size of the array, it is necessary to provide some kind of bound checking, for example, by declaring in the base class a static member *count* to compare it with array's size in the function *GetData*.
Explain (in a suitable manner) by example how this hierarchy can be extended and what benefits are gained by the inheritance mechanism.
Provide a brief description of a way to tag a Shape subclass as 2D or 3D using multiple inheritance?

References:

Bjarne Stroustrup, The C++ Programming Language, 3rd ed., 1997:

12.2. Derived classes.
12.2.1. Member functions.
12.2.2. Constructors and destructors.
12.2.4. Class hierarchies.
12.2.6. Virtual functions.
12.3. Abstract classes.

Jan Skansholm, C++ direct, Studentlitteratur, 1996:

9.1. Härledda klasser.
9.2. Konstruktorer och destructorer vid arv.

9.3. Tillgänglighet.
9.5. Polymorfism och dynamisk binding.
9.7. Virtuella destruktorer.
9.8. Abstrakta klasser.