Dmitriy Doronenko
V.A. Savel`ev, research supervisor
V.V. Tihonenko, language adviser
SHEI “National Mining University”, Dnipropetrovsk

Application of Abstract Classes in C++

This material studies application of an abstract class with a simple example of its use.

The abstract class is the class which copies cannot be created. The abstract class can be used only as the basic one for other class. If virtual function isn't defined in a derivative class, it remains like that, meaning that the derivative class is abstract as well. At such approach it is possible to implement classes step by step. Abstract classes are necessary for an interface task without specification of any concrete details of implementation. For example, it is possible to hide details of driver implementation in some operating system. The real drivers will be defined as derivative of the class «character device».

While maintaining an abstract class we have all basic means to write the complete program.

Interface differs from an abstract class in the fact that the interface contains the interface only while and an abstract class contains a part of implementation which can be in it.

Let us review a widespread example – two-dimensional geometrical figures. You cannot create a copy of "figure" type. It is too abstract, there is no specifics. However, some definite forms - a circle, a square, a triangle, etc. -can be inherited from a shape. Here their copies can already be created. We can find a lot of material about virtual functions because only the class containing at least one purely virtual function can be an abstract one. Class «Shape» is abstract only as purely virtual functional area is said to be there (after all there we cannot calculate the area of some abstract figure...) . But in classes which inherit from Shape - Circle and Square the area function already has some definition. Then, in the main function, the array of Shape indexes is developed. Then each index is initialized by the specific object – by a circle or a square. Further, without thinking, in which array cell the object is, we simply call in required area method, and it will execute definite the actions for each object. It is very convenient when it is not known in advance, which object next index will pointed. It is called late (or dynamic) binding.