Computer Lab #8.

Inheritance and Polymorphism in C++

Inheritance, abstraction, encapsulation, and polymorphism features are the cornerstones of object-oriented programming. In general, inheritance allows one data type to acquire properties of other data types and polymorphism enables one common interface for many implementations (for objects to act differently under different circumstances). C++ supports several kinds of static (compile-time) and dynamic (run-time) polymorphisms.

Static polymorphism is a function overloading, which allows programs to declare multiple functions having the same name (but with different arguments). The functions are distinguished by the number or types of their formal parameters. Dynamic polymorphism means that variable pointers (and references) to a base class type can refer to objects of any derived classes of that type in addition to objects exactly matching the variable type. This allows arrays and other kinds of containers to hold pointers to objects of differing types. Because assignment of values to variables usually occurs at run-time, this is necessarily a run-time phenomenon.

Task 1

Write a program for registering and storing computer components. There are two types of available components: printers and displays.

a) For a display, you should store its identification number, price and year of appropriation. For printer, you should store its identification number, price and printing speed.

b) Write the *Equipment* class that should be the base class for all the components. Inherit the components from the common base class.

c) Your task is to keep the different components in a single array; the maximum number of components is *maxEquipment*. Download ***CompSys.cpp*** file from the website and use it as the main program. Iterating the components in a loop, print the components' attributes. Make sure that the attributes are printed with respect to component's type. Check the output.

d) What happens if the method *Equipment::print()* is not pure virtual? Check it using the debugger.

<u>Decomposition of the program:</u>

- *Equipment.h, Printer.h, Display.h*: class declaration headers;
- *Display.cpp, Printer.cpp*: implementation of class members declared in the headers;
- *CompSys.cpp*: the main program with testing code.

Task 2 (optional)

Combine *Printer* class and *Vector* dynamic array. Modify the data type stored by *Vector* class (see Computer Lab #7): change *int* to *Printer* type. As a result, you should get a dynamic array that can store printers. The header and implementation for *Vector* can be downloaded from the website (***Vector.h*** and ***Vector.cpp*** respectively).