Jonathan Tan

HW3 Report
Builder V0.2

Motivation and Objective

The motivation and objective behind my new builder is to add in the possibility of multiple inheritance. The previous builder was good but it was restricted because it could only handle single class inheritance. My new builder is capable of creating working C++ code with classes inheriting from an infinite amount of other classes. My new AddMethod shell script also takes care of multiple inheritance by adding in the virtual function to the base class files if the passed in class inherits from other classes. As long as the inheritance is defined in “InheritanceTree.txt,” my builder will create the proper working code. In order to define classes and their parent classes, I created the shell script “InsertInheritance” which takes in 3+ parameters: first the InheritanceTree file, then the derived class and then the derived class's base classes. InsertInheritance inserts the class family tree into InheritanceTree.txt. After this, my InitClassFiles shell script creates the class and takes care of the rest. As far as my C++ knowledge goes, my current builder is probably the best a basic builder can get. The next step would be to add much more complex functions to the code using the AddMethod function. To do this however, the only thing I can think of is to add a new parameter to the AddMethod function where you input a file containing your function's code to add to your AddMethod function. So the new call to that shell script would look something like : './AddMethod.sh Class Method Const/Non-Const Type File Param1:Type Param2:Type Param3:Type....' The file would contain the code to append into the middle of the function created by the AddMethod script.

Changes and Additions

I created the following shell scripts:

getParents.sh – Takes in two parameters: a) InheritanceTree.txt and b) a class name. If the class is a derived class in InheritanceTree.txt, then getParents.sh echoes out the name of the passed-in class’s parent classes. If the class is not a derived class in InheritanceTree.txt, getParents.sh echoes out -1.

InsertInheritance.sh – Takes in at least three parameters: a) InheritanceTree.txt, b) the derived class’s name, c) the base class, d) another base class, e) another base class… This script appends the derived class and its base classes into InheritanceTree.txt in the following format:
DerivedClass BaseClass1:BaseClass2:BaseClass3:BaseClass4(…)
Note: There is no colon following the last base class.

isParent.sh – Takes in two parameters: a) InheritanceTree.txt and b) a class name. Returns 1 if the passed-in class is a parent class in InheritanceTree.txt. Otherwise it returns -1.

I edited the following shell scripts:
AddConsDes.sh – Now takes care of multiple inheritance when creating the class and its constructor and destructor. If the passed-in class is a derived class then the base classes are automatically appended to the class definition. For example, Pegasis is a derived class from Horse and Bird. So if Pegasis was passed into AddConsDes.sh as the class parameter, its class definition would look like this: class Pegasis: Public Horse, Public Bird. However, if the class Bird is passed in (which is a base class and not derived from any class stated in InheritanceTree.txt), then it creates

the class as normal except it makes the destructor a virtual function to account for the warning: “Warning: Class has virtual functions but non-virtual destructor.”

AddMethod.sh – Acts as usual if the class passed-in is not a derived class. However, if the class passed-in is a derived class then it not only creates the functions in the regular class as normal, but it also adds virtual functions to the passed-in class’s base classes if the class is defined as having parent classes in InheritanceTree.txt.

Design and Implementation Strategies

My design and implementation strategies for this homework assignment is to take care of multiple inheritance with minimal changes to the original builder so as to not mess up my original builder's perfect functionality. To do this, I created the scripts stated above to limit the complexity of my changes to my old builder. Thanks to those scripts, most of the changes I made to my old scripts were only a couple lines. The most complex change I made was to my AddMethod shell script because it had to create the virtual functions in the passed in class's base classes as well as do its original function. However, thanks to the three shell scripts I made, this ended up being a very simple process as well. Thanks to my getParents shell script, all I needed to do to get the parent classes in my AddMethod shell script is do a for loop and for each word in what is returned by the getParents shell script, I create a virtual function in those classes.

Difficulties Encountered

The only difficulty I encountered creating this script was learning how to change the passed in parameters so that my AddMethod.sh did not need to be changed too much. Reason being that the way my AddMethod shell script works is that it originally had 2 functions within my shell script, one that takes care of creating void return types and one that takes care of the other data types (double, string, integer, etc). Changing my AddMethod to create virtual functions in the parent classes was easy once I discovered I could use “shift” to ignore the first class parameter and pass in a new class name to my function parameters. Other than this, the rest of the homework assignment was relatively easily since there was no new problem that I had not encountered before.

Detailed Results/Conclusion

The result of my new builder is a builder that takes care of Multiple Inheritance with absolutely no errors or warnings even when compiled with the warning flags (-W –Werror –Wall). My builder creates the base class with a virtual destructor if it is a base class (Although, it seemed that this step was unnecessary as new compilers often removed the no virtual destructor warning). My builder includes libraries as needed, creates the proper constructor with calls to the base classes, and creates proper functions with the AddMethod script. In conclusion, I am very happy with my new builder because it really is an upgrade from the old now that it can take care of multiple inheritance.