Bonus Assignment: Building your Own Compiler in C++

Due: Friday, April 28, 2017 by midnight

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

- Design, implement and test classes in C++
- Apply and implement *overloaded* operators and functions
- Maintain and expand on a previous C++ solution

II. Prerequisites:

Before starting this programming assignment, participants should be able to:

- Analyze a basic set of requirements for a problem
- Compose basic C++ language programs
- Create basic test cases for a program
- Apply arrays, strings, and pointers
- Declare and define *constructors*
- Declare and define *destructors*
- Compare and contrast *public* and *private* access specifiers in C++
- Describe what is an *attribute* or data member of a class
- Describe what is a *method* of a class
- Apply and implement *overloaded* functions
- Distinguish between pass-by-*value* and pass-by-*reference*
- Discuss *classes* versus *objects*

III. Overview & Requirements:

NOTE: Parts of this assignment are courtesy of Jack Hagemeister.

Preview the description for "Building Your Own Compiler" at the following site: http://deitel.com/bookresources/cpphtp10/CompilerExercise.pdf. You should first complete the Simpletron Computer simulator before starting this project (problems 8.15 – 8.17). Some of you may have experimented with the traditional Basic programming language. This project involves building a compiler / assembler for a restricted version of Basic called Simple. You are required to design and implement the compiler and assembler that is described in the listed .pdf. You are required to complete exercises 19.29 – 19.33. You will have to change the design from that described to have a multiple steps to the compilation process. These are described in the Phases below.

At the completion of this project, you will have developed a complete environment for developing and running Simpletron programs. You may even have an interface that will allow a user to write and edit simple programs or simple assembly programs, compile/assemble them, and then run them in a Simpletron Simulator.

Phase I:

For this first phase, you will want to design value or container type objects that will provide the functionality to convert Simpletron assembly programs to Simpletron machine programs. You will need to lay out the ending machine code so that the executable statements start at location 0000 in memory. Any variables or constant values that are then used by the program should be laid out in high memory, e.g. starting from 9999 and working down.

See Chapter 8 for examples of Simpletron assembly programs and the resulting machine code. Your assembly should support all of the operations that are available in the Simpletron simulator that you have completed.

Phase II:

The second phase is building a compiler/translator from Simple to Simpletron assembly. You should end up with a description of a Simpletron assembly program just like those in Chapter 19. By this point you should be tired of writing them by hand. Your goal is to again design objects that can be dynamically allocated to let you use them if you so choose.

Phase III:

Putting it all together! When you have a working translator and simulation programs, you will want to integrate this all into one program. You will want to be able to load and save programs, as well as edit and compile them. The final part is the integration of the Simpletron simulator that lets you run the programs and step through them to see what is generated by the "SIMPLE" microcomputer.

IV. Submitting Assignments:

1. Send your solution to aofallon@eecs.wsu.edu.
2. Your .zip file should contain a project workspace. Your project folder must have at least one header file (a .h file), two C++ source files (which must be .cpp files), and project workspace. Delete the debug folder before you zip your project folders.
3. Your project must build properly. You could earn up to 3% of your overall grade.

V. Grading Guidelines:

This assignment is worth 200 points. Your assignment will be evaluated based on a successful compilation and adherence to the program requirements. We will grade according to the following criteria:

- 10 pts – Appropriate design, style, and commenting according to class standards
- 40 pts – Design and implement the Simple assembler class (Phase I)
- 50 pts – Design and implement the Simple compiler tool (Phase II)
- 50 pts – Design and implement text-based wrapper for the compiler to assembly (Phase II) part. You should also run the assembly programs through your Phase I tool to check functionality.
- 50 pts – Develop an application to illustrate a running compiler and Simpletron (Phase III).