CSS 448 Translation project Spring 2011

This is a quarter long project, a translation from one language to another using the bottom-up parsing technique. The phases take you completely through the translation process. The project is to translate a program written in modified Pascal into C++. The only difference between your project and writing a compiler is the target language; compilers translate into assembly language or machine language.

While the concepts covered in this course are not unrealistically difficult to grasp, one of the characteristics of this course is that you are asked to write a program (a translator) when you don't feel very comfortable (yet) with what that requires. While you're writing one phase, you have no idea how this fits in the whole project. This can be different from other classes and frustrating, but it is quite normal to feel like you don't know how to write a translator until after you've written one.

There is quite a bit of programming required and each phase must be correct to work with the other phases. It must be correct to all work together. In other words, you can't blow off any portion.

Source language: **modified Pascal**
Target language: **C++**

Here is a brief overview of each component:

Phase I - Lexical Analyzer Due Mon, April 11
Overall, you read in characters and clump them to create a token. A token is a meaningful unit in the language. You will strip out white space and recognize and return in your yylex routine (what the routine must be called):
-- reserved words
-- identifiers
-- constants
-- special characters or groups of characters (e.g., operators, one left parenthesis, left bracket, etc.)
-- etc.

Phase II - Parser Due Wed, April 20
You create (it is started, you finish), a Backus-Normal (Naur) Form (BNF) grammar for the language so that yacc (yet another compiler compiler) can generate a parse table for the language. The parsing phase groups together tokens into meaningful units to determine if they have the right form for the language.

Phase III - Symbol Table Due Wed, May 11
The symbol table (ST) is a data structure with an entry for each identifier in the language. Associated with each identifier you will store all relevant information (identifier attributes). For example:
-- for a simple variable, store type
-- for an array, store type, the number of dimensions and range of each dimension
-- for a record (struct), store each field (member)
-- for a function, store return type as well as a parameter list
-- etc.

Phase IV - Code Generation Due Wed, June 1
As the parser recognizes proper groups of tokens, you have it perform actions. Those actions generate the proper code at the appropriate time.