ICST252 - Compiler Design and Implementation (3 units)

Computer Science Department, College of Computer Studies
Second Semester, School Year 2014-2015

Ateneo de Naga University
Allan A. Sioson, PhD
email: allan@adnu.edu.ph

Programming Assignment No. 2 - Parser

Objective

Your task is to develop `pparse`, a syntax analyzer of the Pmin Programming Language. The Pmin Programming Language grammar is provided in a section below. The description of the lexical units of Pmin Programming Language is provided in the previous programming assignment.

Synopsis

The `pparse` program should be executed as follows:

```
pparse file.pmn
```

A message about correct usage of your program should be shown if no filename is provided or more than one filename is provided. File opening error message should also be shown whenever the filename provided does not exist or cannot be opened.

Program output

The program output must show the generated parse tree using the format demonstrated in class. At the end of the printed parse tree, the *total* number of parse and lexical errors encountered should be reported. The format of the message should be like this:

x `error(s) found.`

where $x > 0$.

Example.

```
5 errors found.
```

Parse Error handling

A parse error message should be outputted to the standard output stream (`cout`) whenever an unexpected symbol is encountered. The format of a parse error message follows:

`file.pmn:row:col:unexpected 'lexeme' encountered (`x `tokens skipped).`

Example.

```
prog1.pmn:3:8:unexpected '+' encountered (5 tokens skipped).
```

An Error message like this should be outputted immediately as soon as a parse error is encountered.

Lexical Error handling

A lexical error message should be outputted to the standard output stream (`cout`) whenever invalid symbols or patterns are encountered. The required formats of lexical error messages follow:

```
file.pmn:row:col:invalid character 'symbol'.
file.pmn:row:col:invalid pattern 'pattern'.
file.pmn:row:col:unterminated string literal.
file.pmn:row:col:unterminated comment.
```

A lexical error message like this should be outputted immediately as soon as a lexical error is encountered.

Comments in Pmin Language

Any sequence of characters that start with `{` and ends with `}` is a Pmin comment. Alternatively, any sequence of characters that start with `(*` and ends with `*)` is a Pmin comment. No token should be generated for any Pmin comment.

Deliverables

The following are the project deliverables:

1. Source files that constitute the `pparse` source code.

   (a) `Makefile` : Makefile to build the `pparse` executable file.
   (b) `main.cpp` : Driver program of the parser.
   (c) `parser.cpp` : File containing the parser.
   (d) `lexer.cpp` : File containing the lexical analyzer.
   (e) `global.h` : Header file containing the global definitions, preprocessor commands, and function prototypes.

   The `FIRST()` and `FOLLOW()` sets for each non-terminal should be computed and documented in the source code. Each source file should have the honor code shown below. You have to affix your signature.

```
/**

  ICST252 Syntax Analyzer
  lastname_p2_pparse.cpp
  Purpose: Prints parse tree from input

  Synopsis:

      pparse file.pmn

  @author Juan A. Dela Cruz (ID 9999-9-9999)
  @version 1.0 11/26/2014

  In my honor, I assure that I have not given nor
  received any unauthorized help in this work.

                            Juan A. Dela Cruz

*/
```

2. Output file for each input file that I will provide. If the input file is named `prog01.pmn`, the output file should be named `prog01.out`.

Important Notes

1. Deadline of printouts is 5:00pm on December 16, 2014 (Tuesday).
2. Deadline of electronic copies is 11:59 pm on December 16, 2010 (Tuesday). You have to submit all source files and program outputs in one zipped file. The format of the filename should be `pparse_<lastname>.zip`.
3. There should be no discrepancy between the electronic copies and the printed copies. There is a deduction of 1 point for each character discrepancy.
4. If the honor code is missing and/or unsigned, you get a zero score.

Grading Considerations

1. Completeness of the Source Code Documentation
2. Efficiency and Correctness of implementation of Algorithms
3. Correctness of output

Syntax of Pmin Language

program $\longrightarrow$ *programHeading declarations* `BEGIN` *optionalStatements* `END` '`.`'

programHeading $\longrightarrow$ `PROGRAM ID` '`;`'

declarations $\longrightarrow$ `VAR` *declarationList*
| ϵ

declarationList $\longrightarrow$ [*identifierList* '`:`' *type* '`;`']$^+$

identifierList $\longrightarrow$ `ID` ['`,`' `ID`]*

size $\longrightarrow$ '`[`' `NUM DOTDOT NUM` '`]`'

type $\longrightarrow$ `INTEGER`
| `REAL`
| `ARRAY` *size* `OF` (`INTEGER` | `REAL`)

optionalStatements $\longrightarrow$ *statementList*
| ϵ

statementList $\longrightarrow$ *statement* ['`;`' *statement*]*

statement $\longrightarrow$ `BEGIN` *optionalStatements* `END`
| `IF` *expression* `THEN` *statement* [`ELSE` *statement*]
| `WHILE` *expression* `DO` *statement*
| `FOR ID ASSIGN` *expression* `TO` *expression* `DO` *statement*
| `ID` *restStatement*

restStatement $\longrightarrow$ ['`[`' *expression* '`]`'] `ASSIGN` *expression*
| '`(`' *expression* '`)`'

expressionList $\longrightarrow$ *expression* [`,` *expression*]*

expression $\longrightarrow$ *simpleExpression* [('`<`' | '`>`' | `EQ` | `LEQ` | `GEQ` | `NEQ`) *simpleExpression*]

simpleExpression $\longrightarrow$ *term* [('+' | '-' | OR) *term*]*

term $\longrightarrow$ *factor* [('*' | '/' | DIV | MOD | AND) *factor*]*

factor $\longrightarrow$ NUM
| FLOAT
| STRING
| '(' *expression* ')'
| NOT *factor*
| '-' *factor*
| '+' *factor*
| ID *restFactor*

restFactor $\longrightarrow$ [*expression*]
| '(' *expressionList* ')'
| ϵ

Sample input

```
(* sample.pmn *)
program test;
var
  x :+ integer;% { extra colon here, invalid symbol '%' }
begin
  x := 5         { missing semicolon }
  writeln(x);
  writeln('hello')
end.
```

Sample output

```
sample.pmn:4:6:unexpected '+' encountered (1 token skipped).
sample.pmn:4:16:invalid symbol '%' encountered.
sample.pmn:7:3:unexpected 'writeln' encountered (4 tokens skipped).

program
|---programHeading
|---|---PROGRAM
|---|---ID 'test'
|---|---';'
|---declarations
|---|---VAR
|---|---declarationList
|---|---|---identifierList
|---|---|---|---ID 'x'
|---|---|---':'
|---|---|---type
|---|---|---|---INTEGER
|---|---|---';'
|---BEGIN
|---optionalStatements
|---|---statementList
|---|---|---statement
|---|---|---|---ID 'x'
```

```
|---|---|---|---restStatement
|---|---|---|---|---ASSIGN
|---|---|---|---|---expression
|---|---|---|---|---|---simpleExpression
|---|---|---|---|---|---|---term
|---|---|---|---|---|---|---|---factor
|---|---|---|---|---|---|---|---|---NUM '5'
|---|---|---';'
|---|---|---statement
|---|---|---|---ID 'writeln'
|---|---|---|---restStatement
|---|---|---|---|---'('
|---|---|---|---|---expression
|---|---|---|---|---|---simpleExpression
|---|---|---|---|---|---|---term
|---|---|---|---|---|---|---|---factor
|---|---|---|---|---|---|---|---|---ID 'hello'
|---|---|---|---|---|---|---|---|---restFactor
|---|---|---|---|---')'
|---END
|---'.'

3 errors found
```