Project2 Report

這次的 project2 主要分為兩個部分，一是將 project1 的 lextemplate.l(這次 project2 我已將之改名為 scanner.l)做一些修改，另一是完成這次 project2 的重點：parser.y。另外，這次作業也要繳交 Makefile，所以此報告的最後面我也會描述我的 Makefile 的寫法。

---

scanner.l：

修改的地方有三個，分別是：

1. 在 Definitions 的部份加上#include “y.tab.h”
2. 在 Rules 的部份，對於每一個 scanner.l 所辨別出來的 token，都會在最後面加上 return token_name;，將這些辨別出來的 token 傳到 parser.y。
   例如以下截圖：

```
   /*  string  */

"\""  {LIST;BEGIN str;}
<str>.*\"  {pch=strtok(yytext,"\"");tokenString(string,pch);strcat(buf, "\"");BEGIN
0;return str_cons;}

   /*  Delimiter  */

","  {tokenChar(',');return COMMA;}
";"  {tokenChar(';');return SEMICOLON;}
":"  {tokenChar(':');return COLON;}
"("  {tokenChar('(');return LEFTS;}
")"  {tokenChar(')');return RIGHTS;}
"["  {tokenChar('[');return LEFTM;}
"]"  {tokenChar(']');return RIGHTM;}


   /*  Operator  */

"+"  {token(+);return ADD;}
"-"  {token(-);return SUB;}
"*"  {token(*);return MUL;}
"/"  {token(/);return DIV;}
"mod"  {token(mod);return MOD;}
":="  {token(:=);return ASSIGN;}
"<"  {token(<);return LT;}
"<="  {token(<=);return LE;}
"<>"  {token(<>);return NE;}
">="  {token(>=);return GE;}
">"  {token(>);return GT;}
```

```
   /*  Identifier  */

{letter}({letter}|{digit})*  {tokenString(id,yytext);return ID;}

   /*  Scientific Notations  */

-?{digit}+(\.{digit}+)?[Ee][-+]?{digit}+ {tokenString(scientific,yytext);return SCIEN;}

   /*  Numbers  */

-?0|[1-9][0-9]*  {tokenString(integer,yytext);return int_cons;}
0[1-7][0-7]*   {tokenString(oct_integer,yytext);return oct_cons;}
-?{digit}+\.{digit}+  {tokenString(float,yytext);return real_cons;}
```

```
   /*  Keywords  */

"array"    {token(KWarray);return KWarray;}
"begin"    {token(KWbegin);return KWbegin;}
"boolean"  {token(KWboolean);return KWboolean;}
"def"      {token(KWdef);return KWdef;}
"do"       {token(KWdo);return KWdo;}
"else"     {token(KWelse);return KWelse;}
"end"      {token(KWend);return KWend;}
"false"    {token(KWfalse);return KWfalse;}
"for"      {token(KWfor);return KWfor;}
```

這些 token_name 必須跟 parser.y 中使用的 token 相互對應，所以在 parser.y 的 Definitions 部分，就會定義以下%token。

```
%token KWarray KWbegin KWboolean KWdef KWdo KWelse KWend KWfalse KWfor KWinteger KWor
%token KWif KWof KWprint KWread KWreal KWstring KWthen KWto KWtrue KWreturn KWvar KWwhile
%token ADD SUB MUL DIV MOD ASSIGN LT LE NE GE GT EQ AND OR NOT
%token COMMA SEMICOLON COLON LEFTS RIGHTS LEFTM RIGHTM
%token ID int_cons real_cons str_cons oct_cons SCIEN
```

3. 在 Routines 的部份，我則使用 int yywrap(){return 1;}取代本來的 main function，成功解決編譯 scanner.l 與 parser.y 時兩個程式中 main function 重複定義的問題。

parser.y：

parser.y 要寫兩個部分，分別在 Definitions 和 Rules 兩處：

1. 在 Definitions 部分，除了前面所述要定義 %token 之外，還必須定義一部份 tokens 的 precedence 和 associativity。於是按照講義的方法，使用%left 來定義(因為 project2.pdf 有說以下這些 token 都是 left-associative)。如下：

```
%left OR
%left AND
%left NOT
%left LT LE NE GE GT EQ
%left ADD SUB
%left MUL DIV MOD
```

← 最下面 precedence 最高，反之。

2. 在 Rules 部分則是最重要的 P language 的 Context Free Grammar。基本上就是小心地按照 project2.pdf 中把 P language 的文法規則寫出來，然後以 yacc 規定的方式將該文法鍵入 parser.y，最後透過比對 example.tar.xz 中六個範例 code 的執行結果將 parser.y 中沒注意到的 BUG 都找出來。
   若文法定義如下：

Program

A program has the form:

identifier;
<zero or more variable and constant declaration>
<zero or more function declaration>
<one compound statement>
end *identifier*

則 parser.y 的寫法就是：

```
program    : ID SEMICOLON StarVCdec StarFdec Comp KWend ID
           ;
```

其中 StarVCdec、StarFdec 及 Comp 是 Nonterminal，其餘則皆為 Terminal。
而 StarVCdec 的衍生寫法如下(StarFdec 亦同)：

```
StarVCdec :
          | StarVCdec VCdec
          ;
```

如此便可達成 zero or mire variable and constant declaration 的要求了。

Makefile：

我使用的環境是 VMware Workstation 上的 ubuntu 作業系統，並使用 gedit 來編輯程式，再打開 Terminal 下指令來編譯及執行程式。

而我的 Mikefile 的內容如下：

```
parser: lex.yy.c y.tab.c
        gcc -o parser lex.yy.c y.tab.c -ly -ll
lex.yy.c: scanner.l
        lex scanner.l
y.tab.c: parser.y
        yacc -d -v parser.y
```

結構就是：

target_file: source_file1 source_file2 …

command1

command2

…

寫完 Makefile 後，存檔然後開啟 Terminal 輸入：

% rm parser (若第一次編譯不用輸入此行)

% make -f Makefile

% ./parser [input_file] (file_name.p)