B103040040 莊晏碩

1. Lex 版本：flex 2.6.4
   Yacc 版本：bison (GNU Bison) 3.8.2
2. 作業平台：Cygwin
3. 執行方式：輸入 make 或 make all 即可產生 a.exe 執行檔，再輸入 ./a < 1.pas(或其他要測試的 pascal 檔)便可執行。
4. 你/妳如何處理這份規格書上所提到可能會出現的 error：

(1) 變數(或函數)未定義就使用，或是賦值型態與宣告型態不同(宣告 integer，賦值 string)：

在成功宣告變數時，將變數名稱和形態存在 symbol table 裡，每當要使用變數時，確認該變數是否存在於 symbol table 裡，若不存在則輸出錯誤訊息。當賦值給變數時，若該值不符合該變數之型態，則輸出錯誤訊息，如 real 型態變數可存 integer、double、real 的值，所以存其他型態的值會輸出錯誤訊息。

(2) 結構有缺失，如遺漏左括號、then 前面沒有 if 等等：

在程式中 define parse.error verbose，輸出其所偵測到的錯誤。

(3) 缺少必要的符號(分號、句號等等)或使用錯誤的符號：

部分符號問題有特別於程式中抓出來判斷，如 program 最後面的 end 沒有加句點或宣告變數時將 : 誤打成 := ，其餘則輸出 parse.error verbose 所跳出的錯誤訊息。

(4) 不同類別的變數相加：

在加、減、乘、除、取餘數時檢查左右兩邊的型態，若是變數則到 symbol table 中確認該型態，若是數字則檢查有沒有小數點或 E，有則判定為實數，沒有則判定為整數，再根據各運算方式制定不同的規則，不符合規則的情況輸出錯誤訊息。規則例如：整數、浮點數、實數可互相加減乘除；字元和字串可相加(做字串的串接(concatenation))；取餘數時兩邊都只能是整數。

5. 你/妳寫這個作業所遇到的問題：

從 lex 傳 token 的方法不太熟悉，輸出錯誤訊息時需要標明錯誤在第幾個字的部分也研究了很久，直到後來每次 lex 判斷一個 token 時都記錄該 token 的起始位置時才解決。

6. 所有測試檔執行出來的結果：(見下頁)

```
$ ./a < yacc_testfile/testfile/correct.pas
Line 1: program test;
Line 2: var
Line 4:   i, j: integer;
Line 5:   ans: array[0 .. 81] of integer;
Line 6: begin
Line 7:     i := -1+3;
Line 8:     j := +7*8;
Line 9:     ans[0] := 7;
Line 14:     for i:=1 to 9 do
Line 15:     begin
Line 16:         for j:=1 to i do
Line 17:             ans[i*9+j] := i*j;
Line 18:     end;
Line 20:     for i:=1 to 9 do
Line 21:     begin
Line 22:         for j:=1 to i do
Line 23:             if ( ans[i*9+j] mod 2 = 0) then
Line 24:                 write(i, '*', j, '=', ans[i*9+j], ' ');
Line 25:         writeln;
Line 26:     end;
Line 27: end.
```

```
$ ./a < yacc_testfile/testfile/error1.pas
Line 1: program test;
Line 2: var
Line 3:   i: integer;
Line 4: begin
Line 5, at char 5: ":=" expected but "=" found
Line 6, at char 3: Identifier not found "j"
Line 7, at char 11: Identifier not found "j"
Line 8:     Write('ok');
Line 9: end.
```

```
$ ./a < yacc_testfile/testfile/error2.pas
Line 1: program test;
Line 2: var
Line 3:   i, j : integer;
Line 4: begin
Line 5:   i := 5*2;
Line 6:   j := 9;
Line 7:   if (i > j) then
Line 8:     Write('ok');
Line 9: end.
```

```
$ ./a < yacc_testfile/testfile/error3.pas
Line 1: program test;
Line 2: var
Line 3, at char 8: ":" expected but ":=" found
Line 4: begin
Line 5, at char 3: Identifier not found "i"
Line 6, at char 4: Syntax error, "." expected but "end of file" found
```

```
$ ./a < yacc_testfile/testfile/error4.pas
Line 1: program test;
Line 2: var
Line 3:   i, j : integer;
Line 4:   c : string;
Line 5: begin
Line 6:   i := 5;
Line 7:   c := 'aa';
Line 8, at char 5: ":=" expected but "=" found
Line 9: end.
```