

Programming Languages and Compilers

Lectured by Prof. Chung Yung

Programming Assignment 1

410821312 張宸瑋 資工三

410921217 鄭宇婕 資工二

1. Problem description

Use lex (or flex) and yacc (or bison) to implement a front end (including a lexical analyzer and a syntax recognizer) of the compiler for the MiniJ programming language, which is a simplified version of Java especially designed for a compiler project by Professor Chung Yung.

2. Highlight of the way you write the program

In this assignment, with the given reference file package by professor. There are two main file need to be modify, which is `minij_lex.l` and `minij_parse.y`.

In the `minij_lex.l` file, we need to write out own MiniJ Lexical Rules according to the description.

Following picture are the modify part of the file:

```
45  System.Out.println      {return PRINT;}
46  "!"                     {return NOT;}
47  "||"                    {return OR;}
48  "=="                    {return EQ;}
49  "["                     {return LSP;}
50  "]"                     {return RSP;}
51  ";"                     {return SEMI;}
52  "="                     {return ASSIGN;}
53
54  "//"{NONNL}*            {/* DO NOTHING */}
55  {ID}    {scanf("%s",name); return (ID);}
56  {LIT}   {return (LIT);}
```

In the `minij_lex.l` file, we basically just have to understand the rules and write the corresponding token and the return words. Note that `System.Out.println` is a reserved word, so it doesn't need a double quotes. Also for the `COMMENT`, `ID` and `LIT`, since they are sets, so we need a curly brackets. The special thing about comment is that we don't want to compile that line, so everything after `//`, we don't have to do anything about it, that's why instead of returning comments, we simply just `/* DO NOTHING */`.

For the minij_parse.y, we need to declare our tokens and change our expect number based on our code. Also understand the MiniJ grammar. After understanding the grammar rules, the next step is to write our grammar rules based on the EBNF rules in the E-learning pdf file of this assignment.

There are some mistake we made during modifying the minij_parse.y file

1. The statements constant we named it as states, but professor named it as stmts, So we need to find out all the stmts and change it into states.
2. If the grammar rule has a *, we need to add a new rules for it. For example, if we need a ABC*, we must have the ABC rule and also add a ABCS, which implies ABC ABCS(based on regular expression)
ex: ABCS : ABC ABCS {printf("ABCS -> ABC ABCS"); }
;
3. Need to check if the expect value is correct. If the expect value is not correct, while using bison to compile the minij_parse.y, there might be warring in the cmd making the compilation failed.

Codes are in the listing part!

3. Program listing

Link to the code file on HackMD

- [minij_parse.y and minij_lex.l](#)

4. Test and run result

Original folder

名稱	修改日期	類型	大小
.DS_Store	2022/3/24 上午 09:45	DS_STORE 檔案	7 KB
Makefile	2022/3/24 上午 09:45	檔案	1 KB
minh.txt	2022/3/25 上午 12:53	文字文件	1 KB
minij.c	2022/3/24 上午 09:45	C 來源檔案	1 KB
minij.h	2022/3/24 上午 09:45	C Header 來源檔案	1 KB
minij_lex.l	2022/3/26 下午 03:40	L 檔案	2 KB
minij_parse.y	2022/3/26 下午 03:44	Y 檔案	4 KB
README.txt	2022/3/24 上午 09:45	文字文件	1 KB
test1.mj	2022/3/24 下午 08:16	MJ 檔案	1 KB
test2.mj	2022/3/24 上午 09:45	MJ 檔案	1 KB
test3.mj	2022/3/24 上午 09:45	MJ 檔案	1 KB

1. Use bison to compile minij_parse.y into minij_parse.c.
2. Use flex to compile minij_lex.l into minij_lex.c.
3. Use gcc to compile minij_lex.c into minij_lex.o, minij_parse.c into minij_parse.o, and minij.c into minij.o.
4. Use gcc to link minij.o, minij_lex.o, and minij_parse.o into minijparse

Following picture is the screen shot while compiling on cmd

Now we type in the command line provide by professor

Bison

- bison -d -o minij_parse.c minij_parse.y l

Flex

- flex -ominij_lex.c minij_lex.l l

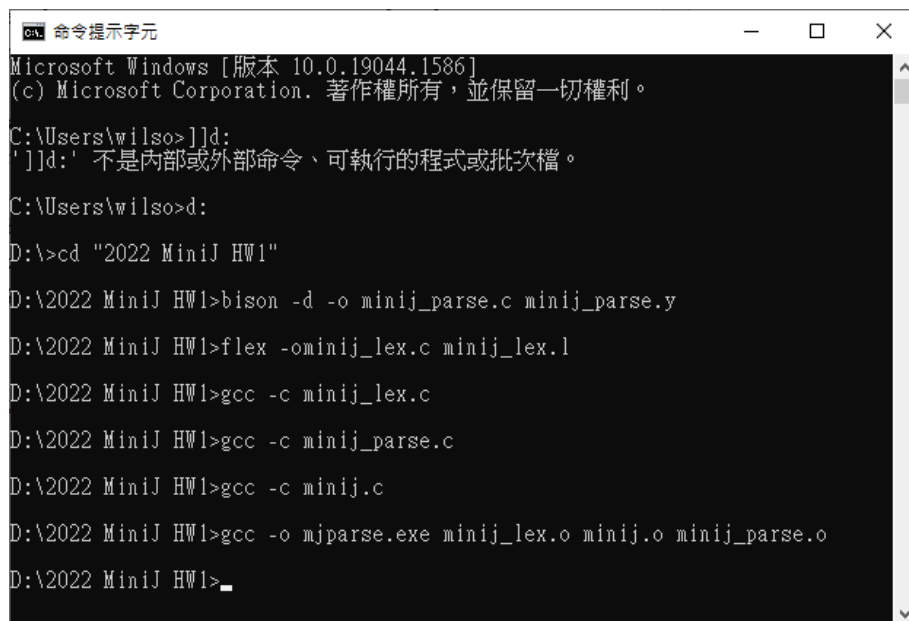
GCC

- gcc -c minij_lex.c

- gcc -c minij_yacc.c

- gcc -c minij.c gcc -o mjparse minij_lex.o minij.o minij_parse.o

-



```
Microsoft Windows [版本 10.0.19044.1586]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\wilso>]]d:
']]d:' 不是內部或外部命令、可執行的程式或批次檔。

C:\Users\wilso>d:
D:\>cd "2022 MiniJ HW1"
D:\2022 MiniJ HW1>bison -d -o minij_parse.c minij_parse.y
D:\2022 MiniJ HW1>flex -ominij_lex.c minij_lex.l
D:\2022 MiniJ HW1>gcc -c minij_lex.c
D:\2022 MiniJ HW1>gcc -c minij_parse.c
D:\2022 MiniJ HW1>gcc -c minij.c
D:\2022 MiniJ HW1>gcc -o mjparse.exe minij_lex.o minij.o minij_parse.o
D:\2022 MiniJ HW1>_
```

After typing those commands, we can see the new generate file in the folder

名稱	修改日期	類型	大小
.DS_Store	2022/3/24 上午 09:45	DS_STORE 檔案	7 KB
Makefile	2022/3/24 上午 09:45	檔案	1 KB
minh.txt	2022/3/25 上午 12:53	文字文件	1 KB
minij.c	2022/3/24 上午 09:45	C 來源檔案	1 KB
minij.h	2022/3/24 上午 09:45	C Header 來源檔案	1 KB
minij.o	2022/3/26 下午 04:28	O 檔案	2 KB
minij_lex.c	2022/3/26 下午 04:27	C 來源檔案	46 KB
minij_lex.l	2022/3/26 下午 03:40	L 檔案	2 KB
minij_lex.o	2022/3/26 下午 04:28	O 檔案	14 KB
minij_parse.c	2022/3/26 下午 04:27	C 來源檔案	59 KB
minij_parse.h	2022/3/26 下午 04:27	C Header 來源檔案	3 KB
minij_parse.o	2022/3/26 下午 04:28	O 檔案	10 KB
minij_parse.y	2022/3/26 下午 03:44	Y 檔案	4 KB
mjparse.exe	2022/3/26 下午 04:28	應用程式	58 KB
README.txt	2022/3/24 上午 09:45	文字文件	1 KB
test1.mj	2022/3/24 下午 08:16	MJ 檔案	1 KB
test2.mj	2022/3/24 上午 09:45	MJ 檔案	1 KB
test3.mj	2022/3/24 上午 09:45	MJ 檔案	1 KB

Red dots are the new generate file

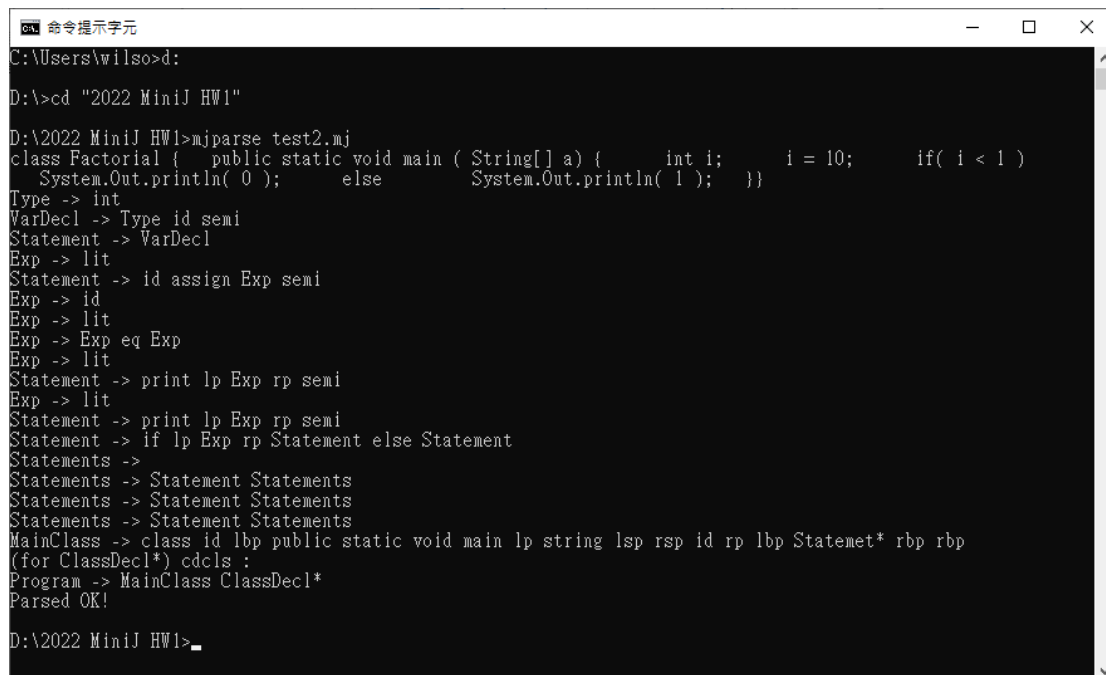
After we get the mjparse.exe, we are able to test or file

Test1



```
命令提示字元
Microsoft Windows [版本 10.0.19044.1586]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。
C:\Users\wilso>d:
D:\>cd "2022 MiniJ HW1"
D:\2022 MiniJ HW1>mjparse test1.mj
class Factorial {   public static void main ( String[] a) {      System.Out.println( 10 );   }}
Exp -> lit
Statement -> print lp Exp rp semi
Statements ->
Statements -> Statement Statements
MainClass -> class id lbp public static void main lp string lsp rsp id rp lbp Statemet* rbp rbp
(for ClassDecl*) cdcls :
Program -> MainClass ClassDecl*
Parsed OK!
D:\2022 MiniJ HW1>
```

Test2



```
命令提示字元
C:\Users\wilso>d:
D:\>cd "2022 MiniJ HW1"
D:\2022 MiniJ HW1>mjparse test2.mj
class Factorial {   public static void main ( String[] a) {      int i;      i = 10;      if( i < 1 )
    System.Out.println( 0 );      else      System.Out.println( 1 );   }}
Type -> int
VarDecl -> Type id semi
Statement -> VarDecl
Exp -> lit
Statement -> id assign Exp semi
Exp -> id
Exp -> lit
Exp -> Exp eq Exp
Exp -> lit
Statement -> print lp Exp rp semi
Exp -> lit
Statement -> print lp Exp rp semi
Statement -> if lp Exp rp Statement else Statement
Statements ->
Statements -> Statement Statements
Statements -> Statement Statements
Statements -> Statement Statements
MainClass -> class id lbp public static void main lp string lsp rsp id rp lbp Statemet* rbp rbp
(for ClassDecl*) cdcls :
Program -> MainClass ClassDecl*
Parsed OK!
D:\2022 MiniJ HW1>_
```

Test3

```
命令提示字元
D:\>cd "2022 MiniJ HW1"

D:\2022 MiniJ HW1>mjparse test3.mj
class Factorial { public static void main ( String[] a ) {      System.Out.println(new Fac().ComputeF
ac(10));  }}// Facclass Fac { public int ComputeFac ( int num ) {      int num_aux;      if (num <
1)      num_aux = 1;      else      num_aux = num * (this.ComputeFac( num-1 ));      return num_
aux;  }}
Exp -> new id lp rp
Exp -> lit
ExpRest ->
ExpList -> exp exprests
Exp -> id lp ExpList rp
Exp -> Exp dot Exp
Statement -> print lp Exp rp semi
Statements ->
Statements -> Statement Statements
MainClass -> class id lbp public static void main lp string lsp rsp id rp lbp Statemet* rbp rbp
(for VarDecl*) vdcls :
Type -> int
Type -> int
FormalRest ->
FormalList -> Type id FormalRest*
Type -> int
VarDecl -> Type id semi
(for VarDecl*) vdcls :
(for VarDecl*) vdcls : vdcl vdcls
Exp -> id
Exp -> lit
Exp -> Exp eq Exp
Exp -> lit
Statement -> id assign Exp semi
Exp -> id
Exp -> this
Exp -> id
Exp -> lit
Exp -> Exp minus Exp
ExpRest ->
ExpList -> exp exprests
Exp -> id lp ExpList rp
Exp -> Exp dot Exp
Exp -> lp Exp rp
Exp -> Exp times Exp
Statement -> id assign Exp semi
Statement -> if lp Exp rp Statement else Statement
Statements ->
Statements -> Statement Statements
Exp -> id
MethodDecl -> public Type id lp FormalList rp lbp Statements* return Exp semi rbp
(for MethodDecl*) mdcls :
(for MethodDecl*) mdcls : mdcl mdcls
ClassDecl -> class id lbp VarDecl* MethodDecl* rbp
(for ClassDecl*) cdcls :
(for ClassDecl*) cdcls : cdcl cdcls
Program -> MainClass ClassDecl*
Parsed OK!

D:\2022 MiniJ HW1>
```

5. Discussion

We think there are still some room for improvement, assume that we want to run more complicate test code in the future, we need to declare more variable in the `minij_lex.l` file, such as float, double, char, long, extends, etc., we can also modify our grammar rules in `minij_parse.y` to make it more flexible and stricter, by adding more %token and priority rules using %left or %right.