lab1 实验报告

学号 PB20081590 姓名 吕凯盛

实验要求

完成一个完整的 Cminus-f 解析器，包括基于 `flex` 的词法分析器和基于 `bison` 的语法分析器。

实验难点

1. flex，bison的使用说明没看懂。。。

   实验文档上的教程感觉有点简略，在CSDN上查看了别人用这两个软件实现的小词法分析器才读懂

2. comment的识别

   推了挺久，依照comment的模式，得到以下结果

   ```
   \/\*[^*]*[*]+([^*/][^*]*[*]+)*\/
   ```

   测试过没有问题

3. Windows换行符和Unix换行符区别导致比对失败

实验设计

1. 词法分析

   补全模式和动作即可，能够输出识别出的`token`， `text` ,`line(刚出现的行数)`， `pos_start(该行开始位置)`， `pos_end(结束的位置,不包含)`

   - 无须过滤掉的字符处理模式为(识别，start=end，移动end，传结点，返回token)

   ```
   [a-zA-Z]+ {pos_start=pos_end;pos_end+=strlen(yytext);pass_node(yytext);return ID;}
   ```

   - 过滤掉的字符处理模式为(识别，start=end，移动end,start，维护lines)

   ```
   \r {pos_start = pos_end; pos_end = 1;pos_start=1;lines+=1;}
   ```

2. 语法分析

   分三步，

   1. 补全union结构

   ```
   %union {
      struct _syntax_tree_node * node;
   }
   ```

   2. 将1中的结点定义为token即可，将文法中的结点定义为type

   ```
   %type <node> params param-list param args arg-list
   %token <node> CLOSEBRACKET
   ```

   3. 按提供的例子，将文法翻译一遍即可，特别地，empty直接用空即可，即node(0)，否则导致段错误。

   ```
   declaration-list: declaration-list declaration{$$ = node("declaration-list",2,$1,$2);}
                   | declaration {$$ = node("declaration-list",1,$1);}
   statement-list:statement-list statement {$$ = node("statement-list",2,$1,$2);}
                   |{$$ = node("statement-list",0);}
   ```

实验结果验证

测试代码为一个小程序

```
int i;
int j;
int t;
int key;
void swap(int a,int b){
    t=a;
    a=b;
    b=t;
    /*****test\\**))////*/
}
```

结果为

```
>--+ program
|  >--+ declaration-list
|  |  >--+ declaration-list
|  |  |  >--+ declaration-list
|  |  |  |  >--+ declaration-list
|  |  |  |  |  >--+ declaration-list
|  |  |  |  |  |  >--+ declaration
|  |  |  |  |  |  |  >--+ var-declaration
|  |  |  |  |  |  |  |  >--+ type-specifier
|  |  |  |  |  |  |  |  |  >--* int
|  |  |  |  |  |  |  |  >--* i
|  |  |  |  |  |  |  |  >--* ;
|  |  |  |  |  |  >--+ declaration
|  |  |  |  |  |  |  >--+ var-declaration
|  |  |  |  |  |  |  |  >--+ type-specifier
|  |  |  |  |  |  |  |  |  >--* int
|  |  |  |  |  |  |  |  >--* j
|  |  |  |  |  |  |  |  >--* ;
|  |  |  |  |  >--+ declaration
|  |  |  |  |  |  >--+ var-declaration
|  |  |  |  |  |  |  >--+ type-specifier
|  |  |  |  |  |  |  |  >--* int
|  |  |  |  |  |  |  >--* t
|  |  |  |  |  |  |  >--* ;
|  |  |  |  >--+ declaration
|  |  |  |  |  >--+ var-declaration
|  |  |  |  |  |  >--+ type-specifier
|  |  |  |  |  |  |  >--* int
|  |  |  |  |  |  >--* key
|  |  |  |  |  |  >--* ;
|  |  >--+ declaration
|  |  |  >--+ fun-declaration
|  |  |  |  >--+ type-specifier
|  |  |  |  |  >--* void
|  |  |  |  >--* swap
|  |  |  |  >--* (
|  |  |  |  >--+ params
|  |  |  |  |  >--+ param-list
|  |  |  |  |  |  >--+ param-list
|  |  |  |  |  |  |  >--+ param
|  |  |  |  |  |  |  |  >--+ type-specifier
|  |  |  |  |  |  |  |  |  >--* int
|  |  |  |  |  |  |  |  >--* a
|  |  |  |  |  |  |  >--* ,
|  |  |  |  |  |  |  >--+ param
|  |  |  |  |  |  |  |  >--+ type-specifier
|  |  |  |  |  |  |  |  |  >--* int
|  |  |  |  |  |  |  |  >--* b
|  |  |  |  >--* )
|  |  |  |  >--+ compound-stmt
```

```
|  |  |  |  |  >--* {
|  |  |  |  |  >--+ local-declarations
|  |  |  |  |  |  >--* epsilon
|  |  |  |  |  >--+ statement-list
|  |  |  |  |  |  >--+ statement-list
|  |  |  |  |  |  |  >--+ statement-list
|  |  |  |  |  |  |  |  >--+ statement-list
|  |  |  |  |  |  |  |  |  >--* epsilon
|  |  |  |  |  |  |  |  >--+ statement
|  |  |  |  |  |  |  |  |  >--+ expression-stmt
|  |  |  |  |  |  |  |  |  |  >--+ expression
|  |  |  |  |  |  |  |  |  |  |  >--+ var
|  |  |  |  |  |  |  |  |  |  |  |  >--* t
|  |  |  |  |  |  |  |  |  |  |  >--* =
|  |  |  |  |  |  |  |  |  |  |  >--+ expression
|  |  |  |  |  |  |  |  |  |  |  |  >--+ simple-expression
|  |  |  |  |  |  |  |  |  |  |  |  |  >--+ additive-expression
|  |  |  |  |  |  |  |  |  |  |  |  |  |  >--+ term
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  >--+ factor
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  >--+ var
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  >--* a
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  >--* ;
|  |  |  |  |  |  |  |  >--+ statement
|  |  |  |  |  |  |  |  |  >--+ expression-stmt
|  |  |  |  |  |  |  |  |  |  >--+ expression
|  |  |  |  |  |  |  |  |  |  |  >--+ var
|  |  |  |  |  |  |  |  |  |  |  |  >--* a
|  |  |  |  |  |  |  |  |  |  |  >--* =
|  |  |  |  |  |  |  |  |  |  |  >--+ expression
|  |  |  |  |  |  |  |  |  |  |  |  >--+ simple-expression
|  |  |  |  |  |  |  |  |  |  |  |  |  >--+ additive-expression
|  |  |  |  |  |  |  |  |  |  |  |  |  |  >--+ term
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  >--+ factor
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  >--+ var
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  >--* b
|  |  |  |  |  |  |  |  |  |  >--* ;
|  |  |  |  |  |  |  >--+ statement
|  |  |  |  |  |  |  |  >--+ expression-stmt
|  |  |  |  |  |  |  |  |  >--+ expression
|  |  |  |  |  |  |  |  |  |  >--+ var
|  |  |  |  |  |  |  |  |  |  |  >--* b
|  |  |  |  |  |  |  |  |  |  >--* =
|  |  |  |  |  |  |  |  |  |  >--+ expression
|  |  |  |  |  |  |  |  |  |  |  >--+ simple-expression
|  |  |  |  |  |  |  |  |  |  |  |  >--+ additive-expression
|  |  |  |  |  |  |  |  |  |  |  |  |  >--+ term
|  |  |  |  |  |  |  |  |  |  |  |  |  |  >--+ factor
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  >--+ var
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  >--* t
|  |  |  |  |  |  |  |  |  >--* ;
|  |  |  |  |  >--* }
```

实验反馈

这次实验的主要困难是第一次接触flex和bison这两种新的程序，看了几遍文档也没看懂，后面找了一些别人的小项目看了，对这两个软件的理解也深刻了许多，写起实验来也感觉没什么困难。按照文档给的文法和例子来，大概一个下午加晚上就做完了(还有一天是在琢磨文档的意思)😭，总的来说收获还是很大的，对这两种自动化程序有了一定的理解。