Lab1

1 实验内容

本次实验基于 Cminus-f 语言，构建简单的 Cminus-f 解析器，完善 `src/parser` 中 **Flex** 词法解析器和 **Bison** 语法分析器

Flex

- Flex 是一个生成词法分析器的工具，可以将输入按照正则表达式识别为定义的Token，当Flex词法分析器与Bison连用时，Flex通过在 .l 文件中通过对 **yylval** 进行修改，以传递Token的语义值，并返回Token。Bison就可以将对应的值压入栈中，进行规约操作。
- ```
  [0-9]+ {yylval.num=yytext;return INTEGER;}
  ```

  以上面代码为例，通过**Regular Expression**匹配到的数字后，将匹配到的值传递给 **INTEGER** 的语义，并返回 **INTERGER** Token
- 在 Flex 中按照正则表达式定义的顺序，从长到短，从上到下的优先级来决定Token的匹配顺序

Bison

- Bison 可以生成语法分析器，解析输入流中的Token，采用自底向上LALR的分析方法。当Bison读入一个终结符（Token），它会将该终结符及其语义值一起压入堆栈，这个堆栈叫做分析器堆栈。把一个Token压入堆栈通常叫做移进。在Bison分析文件的过程中，我们能够通过编写自己的代码并在action部分调用函数实现分析语法树的构建。
- ```
  expr: term ADDOP term
  {
      switch ($2) {
      case '+': $$ = $1 + $3; break;
      case '-': $$ = $1 - $3; break;
      }
  }
  ```

  以上面代码为例，在自底向上规约过程中，通过 $$,$1,$2...来实现语义的规约结果，对于终结符，$*的值就来自Flex所修改的yylval的值

2 思考题

2.1 Regular Experssion

- `\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*` 正则表达式匹配的字符串的含义是什么?

  答: 可以分开来观察 `\w+([-+.]\w+)*` 匹配电子邮电的开头部分，`@` 将本地部分和公共域分开，后面则是表达公共域名部分。

  总之，这串正则表达式是验证电子邮件地址的
- 匹配 HTML 注释：编写一个正则表达式，可以匹配 HTML 中的注释，例如 `<!-- This is a comment -->`。

  答: `<!--[\s\S]*?-->`

2.2 Flex

- 如果存在同时以下规则和动作，对于字符串 `+=`，哪条规则会被触发，并尝试解释理由

```
%%
\+ { return ADD; }
= { return ASSIGN; }
\+= { return ASSIGNADD; }
%%
```

  答: `\+=` 会优先被触发，因为 `\+=` 更长，正则表达式将会优先匹配较多的字符

- 如果存在同时以下规则和动作，对于字符串 `ABC`，哪条规则会被触发，并尝试解释理由。

```
%%
ABC { return 1; }
[a-zA-Z]+ {return 2; }
%%
```

  答: `ABC` 会优先匹配第一条，因为在匹配字符等长的情况下，先定义表达式的先匹配

- 如果存在同时以下规则和动作，对于字符串 `ABC`，哪条规则会被触发，并尝试解释理由。

```
%%
[a-zA-Z]+ {return 2; }
ABC { return 1; }
%%
```

  答: `ABC` 会优先匹配第一条，因为在匹配字符等长的情况下，先定义表达式的先匹配，另外在命令行中也会提示相关的 **Warning**

2.3 Bison

- 上述计算器例子的文法中存在左递归，为什么 `bison` 可以处理?

  答: 由于 `bison` 为自底向上的规约化处理，只要设计出的 Cminus-f 语言符合 SLR（或者更高级、限制更小的 LR 语言）规则，可以通过查表的方式进行唯一的状态跳转。

- 能否修改计算器例子的文法，使得它支持除数 0 规避功能?

  答: 可以添加对除数是否为零的判断 `factor_zero`，如果为零，报错。修改如下

```
/* calc.y */
term
: factor { $$ = $1;}
| term MULOP factor {
    switch ($2) {
    case '*': $$ = $1 * $3; break;
    case '/':
        if($3 == 0){
            yyerror("ERROR: divide by zero"); YYABORT; // 终止解析
        }
        else $$ = $1 / $3;break;
    }
}
```