语法分析程序的设计与实现——YACC 实现

任飞 2021210724

目录

1 实验题目

1.1 内容

利用 YACC 自动生成语法分析程序，实现对算术表达式的语法分析。要求所分析算数表达式由如下的文法产生。

$$E \to E + T|E - T|T$$
$$T \to T * F|T/F|F$$
$$F \to (E)|\text{num}$$

1.2 要求

在对输入的算术表达式进行分析的过程中，依次输出所采用的产生式。

实现方法要求：

根据给定文法，编写 YACC 说明文件，调用 LEX 生成的词法分析程序。

2 程序设计说明

2.1 使用方法

程序运行后等待用户输入要解析的字符串，输入后程序会依次输出每一步的分析过程（即规范规约），如果输入的字符串符合文法，最后会输出 `Accept!`，否则会输出 `Reject!`。

程序将非负整数识别为 `num`，将 `+`、`-`、`*`、`/`、`(`、`)` 识别为对应的符号，遇到 `$` 或 `EOF` 表示输入结束，忽略空白字符，遇到其他字符则报错。

2.2 源程序解释

根据要求，程序分为 LEX 和 YACC 两部分。

LEX

LEX 部分为词法分析，负责识别输入的字符串，将其转换为 token 流，然后传递给 YACC 部分。题目要求的词法很简单，LEX 部分源码如下

```
%option noyywrap
%%
[ \f\n\r\t\v] {}
[\+\-\*\/\(\)] { return yytext[0]; }
[0-9]+ { return num; }
\$ { return 0; }
. { printf("unexpected character: %c\n", yytext[0]); exit(0); }
```

YACC

YACC 部分为语法分析，负责根据输入的 token 流，判断其是否符合文法，如果符合文法则输出分析过程，否则报错。YACC 部分源码如下

```
%{
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
int yyLEX();
void yyerror(const char *s);
int step;
%}
%start E
%token num
%%
E : E '+' T   { printf("(%d) E → E+T\n", ++step); }
  | E '-' T   { printf("(%d) E → E-T\n", ++step); }
  | T         { printf("(%d) E → T\n", ++step); }
  ;
```

```
16
17 T : T '*' F   { printf("(%d) T → T*F\n", ++step); }
18   | T '/' F   { printf("(%d) T → T/F\n", ++step); }
19   | F         { printf("(%d) T → F\n", ++step); }
20   ;
21
22 F : '(' E ')' { printf("(%d) F → (E)\n", ++step); }
23   | num       { printf("(%d) F → num\n", ++step); }
24   ;
25 %%
26
27 int main() {
28     if (yyparse()) {
29         printf("Reject!");
30     } else {
31         printf("Accept!");
32     }
33     return 0;
34 }
35
36 #include "lex.yy.c"
37
38 void yyerror(const char *s) {}
```

2.3 编译方法

首先使用 `lex lex.l` 生成 `lex.yy.c` 文件，该文件会被包含在 `yacc` 生成的文件中。然后使用 `yacc a.y` 生成 `y.tab.c` 文件，最后直接编译`y.tab.c` 文件即可。

3 测试

输入 1

```
1 (1+2)*(3+4)+5-(((6)/2))
```

输出 1

```
1 (1) F → num
2 (2) T → F
3 (3) E → T
4 (4) F → num
5 (5) T → F
```

```
(6) E → E+T
(7) F → (E)
(8) T → F
(9) F → num
(10) T → F
(11) E → T
(12) F → num
(13) T → F
(14) E → E+T
(15) F → (E)
(16) T → T*F
(17) E → T
(18) F → num
(19) T → F
(20) E → E+T
(21) F → num
(22) T → F
(23) E → T
(24) F → (E)
(25) T → F
(26) F → num
(27) T → T/F
(28) E → T
(29) F → (E)
(30) T → F
(31) E → T
(32) F → (E)
(33) T → F
(34) E → E-T
Accept!
```

程序给出了一个该输入的规范规约。输出与手工编写的 LR 分析程序相同，佐证了结果的正确性。

输入 2

```
1+2*(3)(4)
```

输出 2

```
(1) F → num
(2) T → F
(3) E → T
```

```
(4) F → num
(5) T → F
(6) F → num
(7) T → F
(8) E → T
(9) F → (E)
(10) T → T*F
(11) E → E+T
Reject!
```

虽然发现错误稍晚于手工编写的 LR 分析程序，但仍然能够正确地识别出错误。

4 总结

这次实验让我了解了 LEX 和 YACC 配合使用的方法，体会了语法分析器生成工具的便捷。