专题6 语法制导翻译程序设计原理与实现技术

13281193 田容先

# 程序功能

此程序为以SLR(1)语法分析程序为基础的语义分析程序，输入是一个字符串，输出是一个四元式文件。

# 程序结构

## 主要数据结构

typedef int Status;

typedef int state;

typedef struct{

char c;//char字符

state s;//state状态

char\* o;//归约出当前符号的终结符号，用来最后输出

}SElemType;//char字符 state状态

typedef struct{

char type[5];//s,r,a,E

}TableType;

typedef struct{//文法，->左边是head，右边是tail

char head;

char tail[10];

}grammar;

int NextSym;//当前输入指针

char input[30]="i=(i+i)\*i+i\*i#";//输入字符串

grammar sentence[11] = { { 'S', "A" }, { 'A', "V=E" }, //1 2

{ 'E', "E+T" }, //{ 'E', "E-T" },//3

{ 'E', "T" }, { 'T', "T\*F" }, //4 5

//{ 'T', "T/F" },

{ 'T', "F" }, //6

{ 'F', "(E)" }, { 'F', "i" }, { 'V', "i" }};//7 8 9

TableType ACTION[16][7] = {

// i = + \* ( ) #

{ "s03", "E", "E", "E", "E", "E", "acc" },//0

{ "E", "E", "E", "E", "E", "E", "r01" },//1

{ "E", "s04", "E", "E", "E", "E", "E" }, //2

{ "E", "r09", "E", "E", "E", "E", "E" }, //3

{ "s09", "E", "E", "E", "s08", "E", "E" }, //4

{ "E", "E", "s10", "E", "E", "E", "r02" }, //5

{ "E", "E", "r04", "s11", "E", "r04", "r04" }, //6

{ "E", "E", "r06", "r06", "E", "r06", "r06" }, //7

{ "s09", "E", "E", "E", "s08", "E", "E" },//8

{ "E", "E", "r08", "r08", "E", "r08", "r08" },//9

{ "s09", "E", "E", "E", "s08", "E", "E" },//10

{ "s09", "E", "E", "E", "s08", "E", "E" },//11

{ "E", "E", "s10", "E", "E", "s15", "E" },//12

{ "E", "E", "r03", "s11", "E", "r03", "r03" }, //13

{ "E", "E", "r05", "r05", "E", "r05", "r05" },//14

{ "E", "E", "r07", "r07", "E", "r07", "r07" }, //15

};

// i = + - \* / ( ) #

int GOTO[16][6] = {

// S A V E T F i = + - \* / ( ) #

0, 1, 2, 0, 0, 0, //0

0, 0, 0, 0, 0, 0, //1

0, 0, 0, 0, 0, 0, //2

0, 0, 0, 0, 0, 0, //3

0, 0, 0, 5, 6, 7, //4

0, 0, 0, 0, 0, 0, //5

0, 0, 0, 0, 0, 0, //6

0, 0, 0, 0, 0, 0, //7

0, 0, 0, 12, 6, 7, //8

0, 0, 0, 0, 0, 0, //9

0, 0, 0, 0, 13, 7, //10

0, 0, 0, 0, 0, 14, //11

0, 0, 0, 0, 0, 0, //12

0, 0, 0, 0, 0, 0, //13

0, 0, 0, 0, 0, 0, //14

0, 0, 0, 0, 0, 0, //15

};

typedef struct{

char Type;

int PLACE;

}SymbolTable;

typedef struct{

int Op;

int Arg1;

int Arg2;

int Result;

}FourTable;

char\* TT[6] = {"T1","T2","T3","T4","T5","T6"};

int Tp = 0;//临时变量表指针

## 函数定义

void analyze(SqStack S);//SLR(1)分析函数

SElemType readS(SqStack);//用来读入当前字符和下一个状态

int stateof(TableType);//求这个表项中的数字（如s10，结果就是10）

int mapA(char c);//根据c求c在ACTION表中的位置

int mapG(char c);//根据c求c在GOTO表中的位置

//////////////////////////////////

char\* NewTemp();//生成新的变量

int GEN(int Op, int Arg1, int Arg2, int Result);//将四元式输出到文件里

char\* work(SqStack S, int num);//根据num决定要进行哪步语义分析

////////

## 设计方法

这个实验主要分为两个部分，第一个是slr文法的构造，第二个是如何将语义分析的程序加入到语法分析中。这两个部分都不简单。

首先slr(1)分析中，因为时间关系，分析表没有求出来，所以使用的文法是ppt中的，也只是差了个减号和除号，只要有足够时间，是完全可求的，而且这个不是本实验的最重点部分，所以我也没有过度改进这部分。slr分析的代码不是很多，只是要理清关系，知道应该怎么做。

将语法分析的部分实现之后，就要将语义部分插进去。由于不需要考虑符号表，所以程序难度大大减少。在这里，为了能够输出终结符号，我在对偶栈的结构中加了一个字符串，用来表示，归约成现在这个非终结符号的终结符号（本实验中都是i）或者临时变量。，之后在输出时，找的都是这些终结符号。这个思想是第五章ppt中的图示给我的启发。

我将语义分析部分插在归约之前，在出栈之前，就要查明之前所携带的终结符号是哪个，归约后，要把它传给新的量。

新变量部分，我设立一个数组，内容是T1,T2,T3….每次调用NewTemp函数都会指针加一，将下一个变量名返回。这样做比较省事。

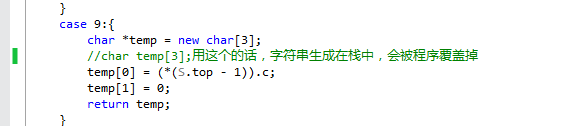
ACTION和GOTO矩阵，因为都是字符串，所以要将其中的数字提取出来，用到了ascii码的转换。

## 设计难点

**1. char \*temp = new char[3];**

**char temp[3]**

我在设计的时候，有一个部分一开始函数将临时生成的temp数组返回给外部，但是这个外部字符串会在几个步骤之后，值自动消失。经过请教学长，应该是因为这个临时变量是在栈中生成，拿到外面的时候，有可能会被程序的进行而挤掉。所以改为new来申请空间，这样会在堆空间申请，不会自动释放。

这个地方是非常重要的难点之一。

**2. 理解如何将语义分析加入**

在完成slr分析之后，不知道如何下手。后来和朱玥同学探讨，她将她的思路告诉了我，虽然我没按照她的方法，但我得知了不用考虑符号表，那么之前许多问题就不用再思考了，也不用被已知的函数、方法局限，所以我就想出了自己的方法。

**3. slr(1)分析表的建立**

因为最近没做这方面的复习，所以slr的分析表一开始建错了。这个分析表错误导致调试时有很大不便，不知道是程序错了，还是表错了。可以说“工欲善其事必先利其器”。后来发现表不对，只能重新写表，又一次认真复习了分析表写法，但因为时间关系，没能写完。但方法已经会了，只要时间够，是一定能写出来的。

**4.数据结构的确定**

做这个实验时，我的数据结构修改了许多次，当然这是实验本身的难度所在，使本不是很长的实验写了很长时间。

# 程序测试

## 测试用例

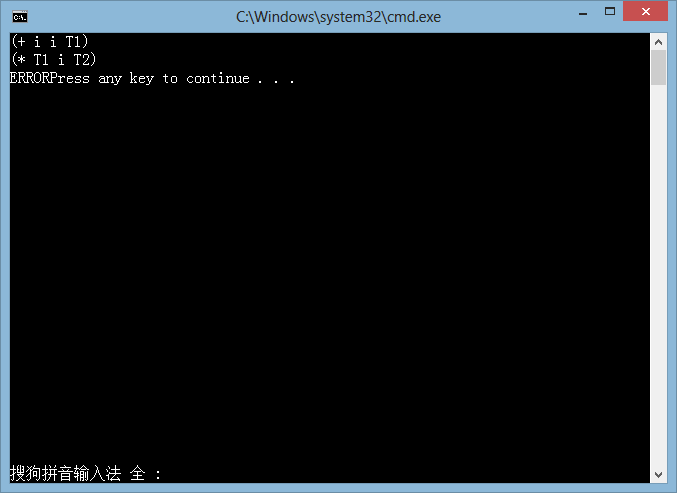
输入："i=(i+i)\*i+i\*i#"

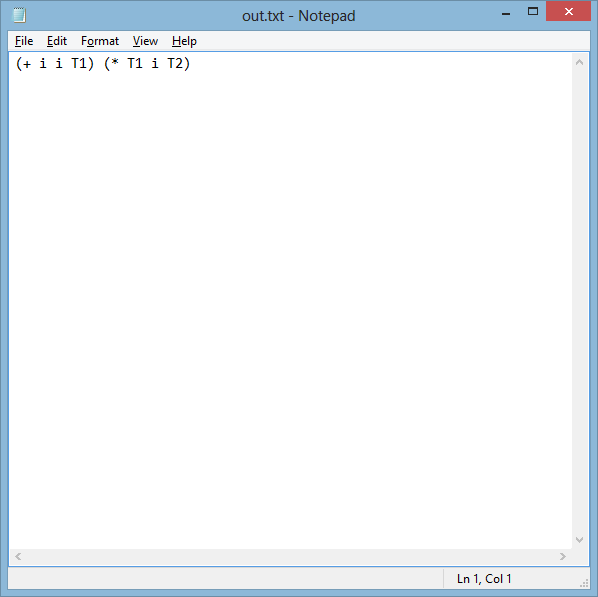
## 测试结果

## 错误用例

"i=(i+i)\*i+i\*\*#"

## 结果





# 实验总结

这次实验室最后一个编译原理实验，代码不难，但是很难思考，让人无从下手，的确要认真思考才能做。在这个实验中，除了对编译知识的理解之外，我认为一个最大的收获是对于new来分配字符串与直接生成的区别，这是对c语言底层的理解，是非常重要的知识。当然我的程序还有大大小小的不足，我会在以后的学习中不断完善自己。

这几天一直在写这几个实验，可以说当真正写完的时候，心情很轻松，虽然还要复习，但因为没有其他任务，所以感到目标很明确。感谢老师的教导。