SLR 语法分析程序实验报告

一、 文法

原文法：

E->E+T|E-T|T

T->T*F|T/F|F

F->id|(E)|num

其中： id: a-f, A-F, num:0-9

拓广文法：

(0)S->E

(1)E->E+T (2)E->E-T （3）E->T

(4)T->T*F (5)T->T/F (6)T->F

(7)F->i (8)F->(E) (9)F->n

其中：i:id, n:num

二、 SLR 分析表

状态	action									goto		
	+	-	*	/	(	)	i	n	$	E	T	F
0					s5		s4	s6		1	2	3
1	s7	s8							acc			
2	r3	r3	s9	s10		r3			r3			
3	r6	r6	r6	r6		r6			r6			
4	r7	r7	r7	r7		r7			r7			

5					s5		s4	s6		11	2	3
6	r9	r9	r9	r9		r9			r9			
7					s5		s4	s6			12	3
8					s5		s4	s6			12	3
9					s5		s4	s6				14
10					s5		s4	s6				15
11	s7	s8				s16						
12	r1	r1	s9	s10		r1			r1			
13	r2	r2	s9	s10		r2			r2			
14	r4	r4	r4	r4		r4			r4			
15	r5	r5	r5	r5		r5			r5			
16	r8	r8	r8	r8		r8			r8			

三、 运行环境

CodeBlocks-13.12 with GCC compiler from TDM-GCC (4.7.1, 32 bit)

四、 输入输出设计

输入：文件“fin.txt”输入待分析串

输出：SLR 分析过程输出至“fout.txt”

五、 主要数据结构

vector<string> G　　//拓广文法的产生式

```
map<char, int> index     //文法符号到下标的转换字典
vector<vector<int> > action     //SLR action 表
vector<vector<int> > goTo       //SLR goto 表
```

六、 核心算法

```
int main()
{
    从文件 fin.txt 读取待分析串到 s;
    s 末尾加‘$’;
    状态栈 vector<int> statusStack;
    符号栈 vector<char> symbolStack;
    状态栈 0; 符号栈压‘$’;
    ip 指向 s 的第一个字符;
    do{
        top 是栈顶符号;
        cur 是 ip 所指向的输入符号;
        if（cur 是字母） cur = ‘i’;
        if（cur 是数字） cur = ‘n’;
        x = top 对应下标;   y = cur 对应下标;
        动作 val = action[x][y];
        if（val == acc）{
            输出 acc;
```

```
            break;
        }
        else if（val 为 shift）{
            输出 shift;
            当前输入符号 cur 压入符号栈;
            动作 val 压入状态栈;
        }
        else if（val 为 reduce）{
            len = reduce 产生式右部长度;
            状态栈和符号栈各弹出 len 个;
            topS = 当前状态栈栈顶;
            curA = 产生式左部非终结符号;
            x = topS 对应下标;  y = curA 对应下标;
            curA 压入符号栈;
            goto[x][y]压入状态栈;
            输出 reduce 产生式;
        }
        else{
            error; break;
        }
    }while（true）;
}
```