实验五 LR 分析器

一、实验目的与任务

设计一个 LR 分析器，实现对表达式语言的分析，加深对 LR 语法分析方法的基本思想的理解，掌握 LR 分析器设计与实现的基本方法。

二、实验要求

建立文法及其 LR 分析表表示的数据结构，设计并实现一个 LALR(1)的分析器，对源程序经词法分析后生成的二元式代码流进行分析，如果输入串是文法定义的句子则输出“是”，否则输出“否”。

三、实验原理

3.1 文法描述及其 LALR(1)分析表

描述表达式语言的文法 G 如下：

S → E
E → E+T
E → T
T → T*F
T → F
F → (E)
F → ID

该文法的 LALR(1)分析表如下：

分析表	动作 Action 表(Yy_action)						转移 Goto 表(Yy_goto)			
状态	#	ID	+	*	(	)	S	E	T	F
0	-	S1	-	-	S2	-	-	3	4	5
1	R6	-	R6	R6	-	R6	-	-	-	-
2	-	S1	-	-	S2	-	-	6	4	5
3	A	-	S7	-	-	-	-	-	-	-
4	R2	-	R2	S8	-	R2	-	-	-	-
5	R4	-	R4	R4	-	R4	-	-	-	-
6	-	-	S7	-	-	S9	-	-	-	-
7	-	S1	-	-	S2	-	-	-	10	5
8	-	S1	-	-	S2	-	-	-	-	11
9	R5	-	R5	R5	-	R5	-	-	-	-
10	R1	-	R1	S8	-	R1	-	-	-	-
11	R3	-	R3	R3	-	R3	-	-	-	-
SN = 移进并转移到状态 N　　A = accept 接受 RN = 按第 N 条产生式进行规约　　- = error 转移										

3.2 LR 分析器总控程序框架

```
push(0);
```

```
advance();
while(Action[tos][sym]!=accept)
   if(Action[tos][sym]==’-’) error();   // (=255)
   else if (Action[tos][sym]==SN){     //（>0)
      push(N);
      advance();
   }
   else if(Action[tos][sym]==RN{     //   (<0)
      act(N);
      pop(产生式 N 的右部的符号个数);
      push(Goto[新 tos][产生式 N 的左部符号]);
   }
accept();
```

上述算法中的有关函数与符号的意义如下：

accept()：返回成功状态，LR 分析器停止工作；

act(N)：执行利用产生式 N 的归约的动作，通常为产生代码；

advance()：从输入流读下一单词到 sym；

error()：出错处理；

pop(N)：从栈顶弹出 N 个符号(状态) ；

push(N)：把状态 N 压入状态栈；

sym：当前输入的单词符号；

tos：栈顶状态号。

四、存放 LR 分析表的数据结构

4.1 用一个二维整数数组表示

数组元素为表示动作的整数。数组的行下标为状态号，列下标用来表示终结符与非终结符的整数表示。数组元素可作如下约定：

正整数：表示移进动作，如 S6 用数 6 表示；

负整数：表示归约动作，如 R5 用数-5 表示；

0：表示接受，通常为按产生式 0 归约；

状态号也用整数表示；

用不可能是状态号的较大的整数表示错误转移。

请将上述 LALR(1)分析表用这种表示方法，完成 LR 分析器的程序设计，并添加输出状态栈内容的功能。用上述表达式文法 G 的一个句子作为输入，进行测试。

4.2 采用压缩表示法

动作 Action 表的每一行用一个数组表示，数组的第一个元素是本数组中存放的数偶个数，第二个元素到最后一个元素都以[终结符，动作]的数偶的形式存放。再用一个以状态号为下标的下标数组，每个元素含一个指向数偶数组的指针。若数组元素的值为 NULL，则表示从此状态无转移弧发出。若分析表有几行相同，则只需保存一行，其它元素则都指向存放这一行表的数组即可。转移 Goto 表也按同样方式组织，只是这个行数组的数偶为[非终结符，下一状态号]。

每个行数组 Yyan 表示动作表 Yy_action 的一行，每个行数组 Yygn 表示转移表 Yy_goto 的一行。假定上述表达式文法 G 中终结符及非终结符的整数值为：

终结符： # ID + * ( ) 非终结符： S E T F

整数值： 0 1 2 3 4 5 整数值： 0 1 2 3

Yy_action 数组是以状态号为下标的下标数组，每个元素含有指向数组 Yyan 的指针；下标数组 Yy_goto 的每个元素含有指向数组 Yygn 的指针。

表达式文法 G 的 LALR(1)分析表的具体压缩表示如下：

Yy_action

.	0	2	4, 2	1, 1			Yya000
.	1	4	5, -6	3, -6	2, -6	0, -6	Yya001
.	2						
.	3	2	0, 0	2, 7			Yya003
.	4	4	5, -2	2, -2	0, -2	3, 8	Yya004
.	5	4	5, -4	3, -4	2, -4	0, -4	Yya005
.	6	2	5, 9	2, 7			Yya006
.	7						
.	8						
.	9	4	5, -5	3, -5	2, -5	0, -5	Yya009
.	10	4	5, -1	2, -1	0, -1	3, 8	Yya010
.	11	4	5, -3	5, -3	2, -3	0, -3	Yya011

Yyg000

.	0	3	3, 5	2, 4	1, 3		Yyg000
NULL	1						
.	2	3	3, 5	2, 4	1, 6		Yyg002
NULL	3						
NULL	4						
NULL	5						
NULL	6						
.	7	2	3, 5	2, 10			Yyg007
.	8	1	3, 11				Yyg008
NULL	9						
NULL	10						
NULL	11						

以上分析表用 C++语言程序描述如下：

```
/*
*       Yy_action 表
```

```
*/
int Yya000[]={2,4,2,1,1};
int Yya001[]={4,5,-6,3,-6,2,-6,0,-6};
int Yya003[]={2,0,0,2,7};
int Yya004[]={4,5,-2,2,-2,0,-2,3,8};
int Yya005[]={4,5,-4,3,-4,2,-4,0,-4};
int Yya006[]={2,5,9,2,7};
int Yya009[]={4,5,-53,-5,2,-5,0,-5};
int Yya010[]={4,5,-1,2,-1,0,-1,3,8};
int Yya011[]={4,5,-3,3,-3,2,-3,0,-3};
int Yy_action[]=
    {
      Yya000, Yya001, Yya000, Yya003, Yya004, Yya005,
      Yya006, Yya000, Yya000, Yya009, Yya010, Yya011
    };
/*
*       Yy_goto 表
*/
int Yyg000[]={3,3,5,2,4,1,3};
int Yyg002[]={3,3,5,2,4,1,6};
int Yyg007[]={2,3,5,2,10};
int Yyg008[]={1,3,11};
int Yy_goto[]=
{
Yyg000, NULL, Yyg002, NULL, NULL, NULL,
NULL, Yyg007, Yyg008, NULL, NULL, NULL
};
/*
*       为了进行归约，使用一个 Yy_lhs[]数组，其值为代表产生式左部符号
的
*       整数，数组的下标为产生式号
*/
    int Yy_lhs[7]=
    {
    /*   0   */     0,
    /*   1   */     1,
    /*   2   */     1,
    /*   3   */     2,
    /*   4   */     2,
    /*   5   */     3,
    /*   6   */     3
    };
/*
*       Yy_reduce[]数组元素的值为产生式右部符号的个数，
```

```
*        以产生式号为数组的下标索引
*/
int Yy_reduce[]=
{
/*   0   */     1,
/*   1   */     3,
/*   2   */     1,
/*   3   */     3,
/*   4   */     1,
/*   5   */     3,
/*   6   */     1
};
```

根据以上数组结构，构造函数 Yy_next()，其功能是在给定状态和输入符号下，求出应采取的动作或转向的下一状态。

```
int Yy_next(table,cur_state,symbol)
int **table;         /*    要查的表    */
int cur_state;      /*      行号（状态）          */
int symbol;          /*      列号  （VN/VT)         */
{
int    *p=table[cur_state];
int i;
if(p)
for(i=(int)*p++ ; --i>0 ; p+=2)
if(symbol = = p[0])
return(p[1]);
return YYF;          /*      出错指示        */
};
```

指针 p 指向 Yyan 数组或 Yygn 数组，由参数 table 的值而定。如果 table 指向 Yy_action，则 p 指向 Yyan 数组；若 table 指向 Yy_goto，则 p 指向 Yygn 数组据。

五、程序设计

根据上述 LALR(1)分析表压缩表示方法，完成 LR 分析器的程序设计，并添加输出状态栈内容的功能。用上述表达式文法 G 的一个句子作为输入，进行测试。