《编译原理》实验指导书

适用实验课时：30

适用对象：计算机科学与软件学院

一、实验目的

编译原理实验的目的是使学生将编译理论运用到实践当中，通过实现一个简单语言集的词法分析程序、语法分析程序和简单语义处理程序，验证实际编译系统的实现方法，并加深对编译理论的认识。

课程目标 1：使学生能够综合运用计算机编译系统所涉及的基本理论、方法和技术，通过文献研究，深入分析一个简单语言集的前端程序，根据实验内容和要求进行问题的描述与表达，获得有效结论。

课程目标 2：使学生能够结合实践，根据学习到的几种主要编译方法，并结合数据结构和离散数学等基础理论知识，对实验内容和要求提出合理的设计与开发方案。采用高级编程语言工具，体验实际编译系统的实现方法。

课程目标 3：能够根据一遍扫描编译器的总体架构，基于词法分析程序、语法分析程序和语义分析程序的原理，综合分析、设计和开发各功能模块，围绕具体实现环节制定技术路线和方案，加深对编译技术的认识，增强设计、编写和调试程序的能力，并能按要求撰写实验报告文档。

二、实验内容

本实验分为三个部分，实验一词法分析器设计与实现：构造具有关键字、运算符、标识符、浮点常数等单词的词法分析程序。输入由符合及不符合规定单词类别结构的各类单词组成的源程序，输出单词串的二元式编码。实验二语法分析器设计与实现：针对简单算术表达式等语法成分，选择有代表性的语法分析方法，设计实现相应的语法分析程序。实验三语义分析程序设计与实现，对各语法单位增加语义子程序，将算术表达式语句翻译为四元式序列。实验内容包括基本实验和扩展实验两部分，其中基本实验部分要求每个同学完成，扩展实验部分供学生自愿选做。

三、实验要求

1、 每次实验前学生应做好实验的设计和准备工作。

2、 独立完成实验，程序书写应符合程序书写规范，积极配合实验进度检查和演示。

3、 针对所完成的实验内容撰写实验报告，不接受不完整的实验报告或者说明与程序、运行结果不符合的作业。

四、实验报告要求

实验报告主要包括三方面内容：

1）实验设计：实验采用的实现方法和依据，如描述语言的文法及其机内表示，词法分析的单词分类码表、状态转换图或状态矩阵等，语法分析中用到的分析表或优先矩阵等，语法制导翻译中文法的拆分和语义动作的设计编写等；具体的设计结果，应包括整体设计思想和实现算法，程序结构的描述，各部分主要功能的说明，以及所用数据结构的介绍等。

2）程序代码：实验实现的核心代码源程序清单，要求符合常规的程序书写风格，标识符命名合理，有必要的注释。

3）实验结果分析：自行编写若干源程序作为测试用例，对所生成的编译程序进行测试，记录运行结果（至少包括一个正确和一个错误单词或语句的运行结果）。

可参考本文附件：实验报告目录（参考）。

五、考核评价方法

成绩采用五级分制，根据实验完成情况、实验报告、实验过程表现、成果演示情况等方面综合评定。进行扩展实验的，按照完成质量加分。

实验一　词法分析程序设计与实现

一、实验目的

通过编写和调试一个词法分析程序，掌握在对程序设计语言的源程序进行扫描的过程中，将字符流形式的源程序转化为一个由各类单词构成的序列的词法分析方法。

二、基本实验内容与要求

假定一种高级程序设计语言中的单词主要包括关键字 begin、end、if、then、else、while、do；标识符；浮点常数；六种关系运算符；一个赋值符和四个算术运算符，试构造能识别这些单词的词法分析程序（各类单词的分类码可参见表 1）。

输入：由符合和不符合所规定的单词类别结构的各类单词组成的源程序文件。

输出：把所识别出的每一单词均按形如（CLASS，VALUE）的二元式形式输出，并将结果放到某个文件中。对于标识符和浮点常数，CLASS 字段为相应的类别码的助记符；VALUE 字段则是该标识符、常数的具体值；对于关键字和运算符，采用一词一类的编码形式，仅需在二元式的 CLASS 字段上放置相应单词的类别码的助记符，VALUE 字段则为“空”。

表 1　语言中的各类单词符号及其分类码表

单词符号	类别编码	类别码的助记符	单词值
begin	**1**	**BEGIN**	
end	**2**	**END**	
if	**3**	**IF**	
then	**4**	**THEN**	
else	**5**	**ELSE**	
while	**6**	**WHILE**	
do	**7**	**DO**	
标识符	**8**	**ID**	**字母打头的字母数字串**
浮点常数	**9**	**UCON**	**机内二进制表示**
<	**10**	**LT**	
<=	**11**	**LE**	
==	**12**	**EQ**	
<>	**13**	**NE**	

>	14	GT	
>=	15	GE	
=	16	IS	
+	17	PL	
-	18	MI	
*	19	MU	
/	20	DI	

要求：

1、上机前完成词法分析程序的程序流程设计，并选择好相应的数据结构。

2、用于测试扫描器的实例源文件中至少应包含两行以上的源代码。

3、对于输入的测试用例的源程序文件，词法正确的单词分析结果在输出文件中以二元式形式输出，错误的字符串给出错误提示信息。

例如，若输入文件中的内容为：“if myid>=1.5 then x=y”，则输出文件中的内容应为：

（IF，　）
（ID，’myid’）
（GE，　）
（UCON，1.5）
（THEN，　）
（ID，’x’）
（IS，　）
（ID，’y’）

三、参考实现方法

1、一般实现方法说明

词法分析是编译程序的第一个处理阶段，可以通过两种途径来构造词法分析程序。其一是根据对语言中各类单词的某种描述或定义（如 BNF），用手工的方式（例如可用 C 语言）构造词法分析程序。一般地，可以根据文法或状态转换图构造相应的状态矩阵，该状态矩阵连同控制程序一起便组成了编译器的词法分析程序；也可以根据文法或状态转换图直接编写词法分析程序。构造词法分析程序的另外一种途径是所谓的词法分析程序的自动生成，即首先用正规式对语言中的各类单词符号进行词型描述，并分别指出在识别单词

时，词法分析程序所应进行的语义处理工作，然后由一个所谓词法分析程序的构造程序对上述信息进行加工。如美国 BELL 实验室研制的 LEX 就是一个被广泛使用的词法分析程序的自动生成工具。总的来说，开发一种新语言时，由于它的单词符号在不停地修改，采用 LEX 等工具生成的词法分析程序比较易于修改和维护。一旦一种语言确定了，则采用手工编写词法分析程序效率更高。本实验建议使用手工编写的方法。

在一个程序设计语言中，一般都含有若干类单词符号，为此可首先为每类单词建立一张状态转换图，然后将这些状态转换图合并成一张统一的状态图，即得到了一个有限自动机，再进行必要的确定化和状态数最小化处理，最后添加当进行状态转移时所需执行的语义动作，就可以据此构造词法分析程序了。

2、单词分类与词法分析器的设计

为了使词法分析程序结构比较清晰，且尽量避免某些枝节问题的纠缠，我们假定要编译的语言中，全部关键字都是保留字，程序员不得将它们作为源程序中的标识符；在源程序的输入文本中，关键字、标识符、浮点常数之间，若未出现关系和算术运算符以及赋值符，则至少须用一个空白字符加以分隔。作了这些限制以后，就可以把关键字和标识符的识别统一进行处理。即每当开始识别一个单词时，若扫视到的第一个字符为字母，则把后续输入的字母或数字字符依次进行拼接，直至扫视到非字母、数字字符为止，以期获得一个尽可能长的字母数字字符串，然后以此字符串查所谓保留字表（此保留字表要事先造好），若查到此字符串，则取出相应的类别码；反之，则表明该字符串应为一标识符。

采用上述策略后，针对表 1 中的部分单词可以参考图 1 和程序 1，用 C 语言编写出符合以上几项要求的一个扫描器程序。

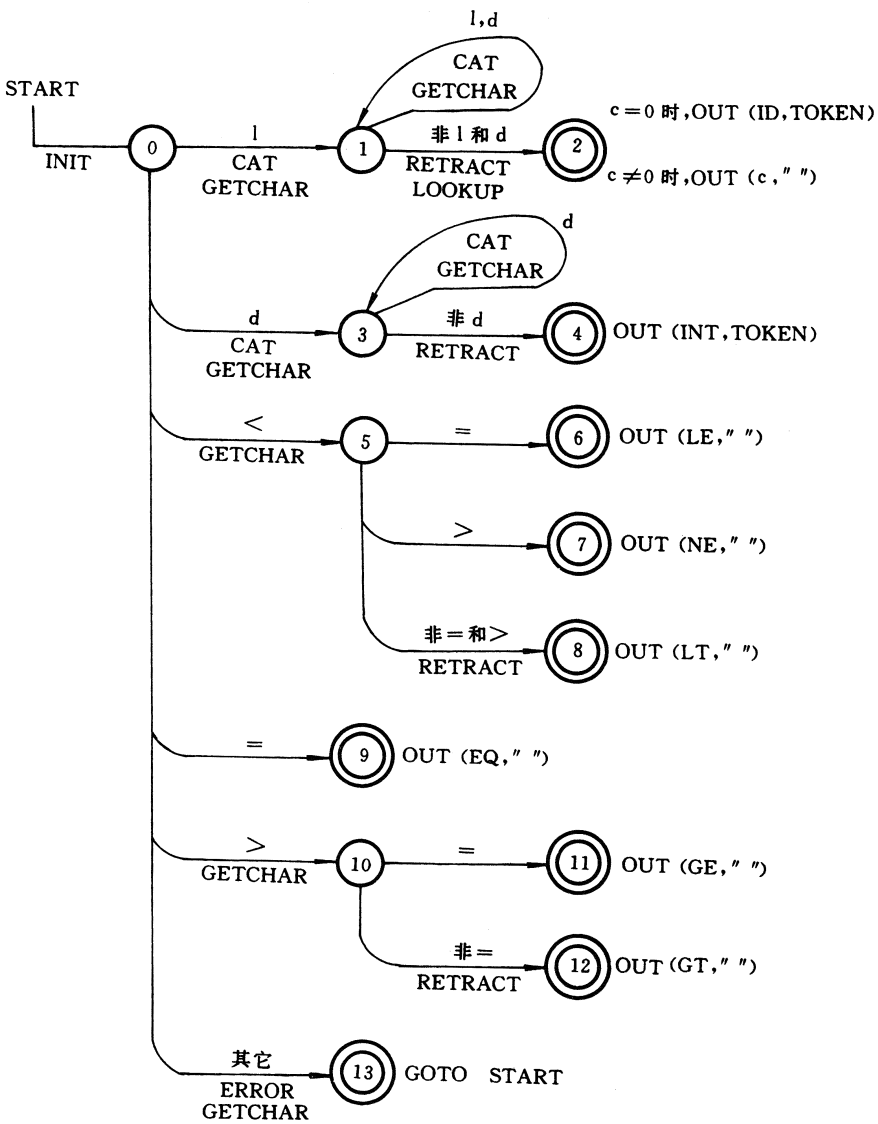

图 1　识别表 1 所列语言中的部分单词的 DFA 及相关的语义过程

图 1 中所出现的语义变量及语义函数的含义和功能说明如下：

函数 GETCHAR：每调用一次，就把扫描指示器当前所指示的源程序字符送入字符变量 ch，然后把扫描指示器前推一个字符位置。

字符数组 TOKEN：用来依次存放一个单词词文中的各个字符。

函数 CAT：每调用一次，就把当前 ch 中的字符拼接于 TOKEN 中所存字符串的右边。

函数 LOOKUP：每调用一次，就以 TOKEN 中的字符串查保留字表，若查到，

就将相应关键字的类别码赋给整型变量 c；否则将 c 置为零。

函数 RETRACT：每调用一次，就把扫描指示器回退一个字符位置（即退回多读的那个字符）。

函数 OUT：一般仅在进入终态时调用此函数，调用的形式为 OUT(c，VAL)。其中，实参 c 为相应单词的类别码助记符；实参 VAL 为 TOKEN（即词文）或为空串。函数 OUT 的功能是，在送出一个单词的内部表示之后，返回到调用该词法分析程序的那个程序。

3、词法分析程序的实现

程序 1　根据图 1 编写的扫描器

```
# include <stdio.h>
# include <ctype.h>
# include <string.h>
# define ID 6
# define INT 7
# define LT 8
# define LE 9
# define EQ 10
# define NE 11
# define GT 12
# define GE 13
char TOKEN[20];
extern int lookup (char*);
extern void out (int, char*);
extern report_error (void);
void scanner_example (FILE *fp)
{
   char ch; int i, c;
   ch=fgetc (fp);
   if (isalpha (ch))    /*it must be a identifer!*/
   {
      TOKEN[0]=ch; ch=fgetc (fp); i=1;
      while (isalnum (ch))
      {
         TOKEN[i]=ch; i++;
         ch=fgetc (fp);
      }
      TOKEN[i]='\0'
      fseek(fp,-1,1);    /* retract*/
      c=lookup (TOKEN);
      if (c==0) out (ID,TOKEN); else out (c," ");
   }
   else
      if(isdigit(ch))
      {
         TOKEN[0]=ch; ch=fgetc(fp); i=1;
         while(isdigit(ch))
         {
```

```
            TOKEN[i]=ch; i++;
            ch=fgetc(fp);
        }
        TOKEN[i]= ′＼0′;
        fseek(fp,-1,1);
        out(INT,TOKEN);
    }
    else
        switch(ch)
        {
            case ′＜′: ch=fgetc(fp);
                       if(ch==′=′)out(LE," ");
                       else if(ch==′＞′) out (NE," ");
                            else
                            {
                               fseek (fp,-1,1);
                               out (LT," ");
                            }
                       break;
            case ′=′: out(EQ, " "); break;
            case ′＞′: ch=fgetc(fp);
                       if(ch==′=′)out(GE," ");
                       else
                       {
                          fseek(fp,-1,1);
                          out(GT," ");
                       }
                       break;
            default: report_error( ); break;
        }
    return;
}
```

程序 1 中所用的若干函数以及主程序有待于具体编写，并需事先建立好保留字表，以备查询。例如：

```
/*  建立保留字表  */
#define MAX_KEY_NUMBER 20    /*关键字的数量*/
#define KEY_WORD_END “waiting for your expanding”    /*关键字结束标记*/
char *KeyWordTable[MAX_KEY_NUMBER]={“begin”,“end”, “if”, “then”, “else”, KEY_WORD_END};
/*  查保留字表，判断是否为关键字  */
int lookup (char *token)
{
   int n=0;
   while (strcmp(KeyWordTable[n], KEY_WORD_END)) /*strcmp 比较两串是否相同，若相同返回 0*/
   {
     if (!strcmp(KeyWordTable[n], token)) /*比较 token 所指向的关键字和保留字表中哪个关键字相符*/
       {
          return n+1; /*根据单词分类码表 I，设置正确的关键字类别码，并返回此类别码的值*/
          break;
       }
     n++;
   }
```

```
    return 0; /*单词不是关键字，而是标识符*/
}
```

四、扩展实验

1、扩充关键字的数目、增加逻辑运算符等单词类别、将常数再细分成字符串常量、整型常量和无符号数常量等；添加词法分析中单词出错的位置和错误类型，以及删除注释部分等。

2、对基本实验内容进行扩充，增加状态转换图显示、词法分析过程的显示等可视化展现功能。

3、研读 GCC，CLANG 等开源编译器的词法分析部分，分析其程序结构、实现方法、识别的单词分类等。

4、用 LEX 自动生成词法分析程序。

5、其它自选题目（注意自选扩展实验题目须经过实验指导教师同意并备案）。

实验二　语法分析程序设计与实现

一、实验目的

任选一种有代表性的语法分析方法，如算符优先法、递归下降法、LL(1)、SLR(1)、LR(1)等，通过设计、编制、调试实现一个典型的语法分析程序，对实验一所得扫描器提供的单词序列进行语法检查和结构分析，实现并进一步掌握常用的语法分析方法。

二、基本实验内容与要求

选择对各种常见高级程序设计语言都较为通用的语法结构——算术表达式的一个简化子集——作为分析对象，根据如下描述其语法结构的 BNF 定义 G2[<算术表达式>]，任选一种学过的语法分析方法，针对运算对象为无符号常数和变量的四则运算，设计并实现一个语法分析程序。

G2[<算术表达式>]：

<算术表达式> → <项> | <算术表达式>+<项> | <算术表达式>-<项>
<项> → <因式> | <项>*<因式> | <项>/<因式>
<因式> → <运算对象> | (<算术表达式>)

若将语法范畴<算术表达式>、<项>、<因式>和<运算对象>分别用 E、T、F 和 i 代表，则 G2 可写成：

G2[E]： E → T | E+T | E-T　　T → F | T*F | T/F　　F → i | (E)

输入：由实验一输出的单词串，例如：UCON，PL，UCON，MU，ID ……

输出：若输入源程序中的符号串是给定文法的句子，则输出“RIGHT”，并且给出每一步分析过程；若不是句子，即输入串有错误，则输出“ERROR”，并且显示分析至此所得的中间结果，如分析栈、符号栈中的信息等，以及必要的出错说明信息。

要求:

1、确定语法分析程序的流程图，同时考虑相应的数据结构，编写一个语法分析源程序。

2、将词法、语法分析合在一起构成一个完整的程序，并调试成功。

3、供测试的例子应包括符合语法规则的语句，及分析程序能判别的若干错例。对于所输入的字符串，不论对错，都应有明确的信息输出。

三、参考实现方法

1、一般实现方法说明

为了在对源程序的一遍扫描过程中，同时完成词法和语法分析任务，应注意首先修改实验一中的词法分析程序，将它编写为子程序的形式，以便供语法分析程序调用。另外，应加强错误检查，对输入符号串中的词法、语法错误，给出必要的说明信息，尽可能多地、确切地指出错误的位置、原因和性质。例如，在词法分析阶段，对当前正在处理的字符 ch，可进一步定义一些与该字符相关的信息 row 和 col，即定义：

```
char ch;   /*The current character*/
int row;   /*The line number position of the current character*/
int col;   /*The column number position of the current character*/
```

分别表示该字符所在的行和列，这些内容在出错处理时用来提供和源程序位置相关的信息。

2、采用具有递归功能的高级语言编制递归下降法的语法分析程序

运算对象 i 可以进一步定义为变量、常数等，例如，此处将 i 定义为：

i→<变量> <变量>→<标识符> <标识符>→<字母> <字母>→A|B|C|…|X|Y|Z

利用扩充的 BNF 消除 G2[E]中 E 和 T 的直接左递归后，采用递归下降法分析上述算术表达式的框图见图 3。其中 ZC 过程为总控程序，被设计成可以分析无穷多个算术表达式，主要完成：①通知外界键入算术表达式。②控制 E 过程分析算术表达式。③根据分析结果之正误，分别输出不同的信息。E、T 和 F 三个过程分别对应<算术表达式>、<项>和<因式>三个产生式的处理。它们都利用到一个公共过程 ADVANCE。ADVANCE 过程负责从输入字符串 ST 中取出下一个字符，并存入 SYM 中等待分析；然后再剔除 ST 中的首字符，这可通过挪动字符串指针的办法来实现（而实际上是通过调用词法分析程序来实现 ADVANCE 功能的）。变量 TZ 之值标志分析的结果，表明表达式是否有错。

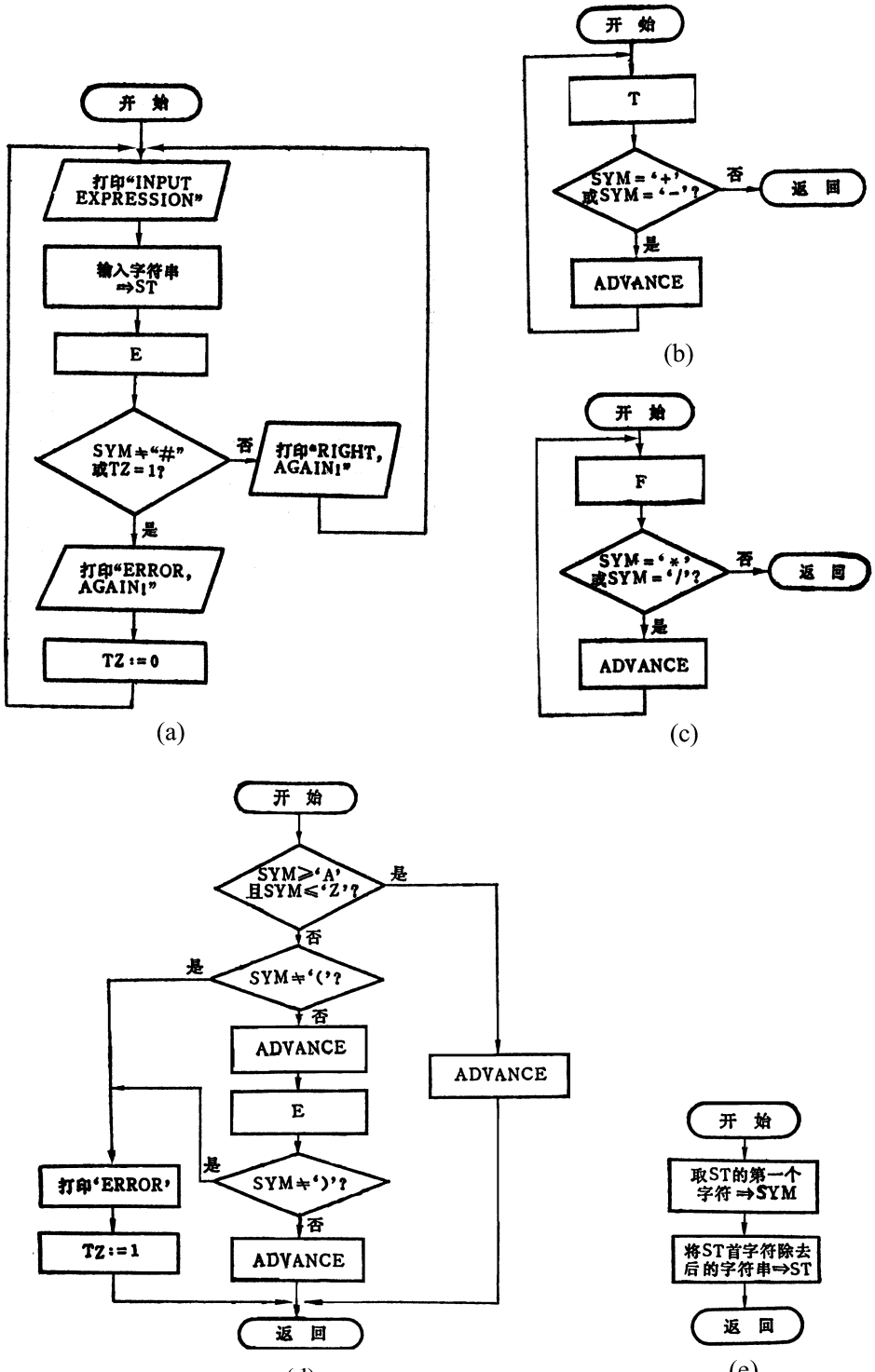

图 3　递归下降法分析算术表达式的框图

(a) ZC 过程　(b) E 过程　(c) T 过程　(d) F 过程　(e) ADVANCE 过程

3、基于预测分析法的语法分析程序

参考教材 P121 例 4.3，首先编写无二义性的简单算术表达式的文法(4.1)，并通过消除左递归对其进行改写得到文法(4.1)’。然后求出如表 4-1 所示的各个 FIRST 集和 FOLLOW 集。再检查是不是 LL(1)文法，并按照 LL(1)文法构造如表 4-2 所示的预测分析表。最后根据预测分析器的工作流程，实现预测分析器的控制程序。

4、算符优先分析法的语法分析程序

如程序 3 所示。其中使用了分析栈 stack，用来在分析过程中存放当前句型的某一前缀，一旦句型的最左素短语在栈顶形成，便立即进行归约。

程序 3　算符优先分析算法

```
#define RIGHT 1
#define ERROR 0
#define MAXINPUT 300
#define MAXSTACK 100
#define STARTSYMBOL 'S'
int stack[MAXSTACK]，a[MAXINPUT];   /* a[ ] is input line */
int IsHigherThan (int, int);    /* priority detection */
int IsLowerThan (int, int);    /* priority detection */
int IsEqualTo (int, int);       /* priority detection */
int Reduce (int begin, int end, int len);
int IsVt (int);    /* determine if stack symbol is in Vt */
int parser (void)
{
   int i, k, r, NewVn;    /* NewVn holds left side of a production */
   i=0; k=0;    /* i, k is index of a[ ] and stack[ ] separately */
   stack[0]= '#';
   do
   {
      int j;
      r=a[i++];
      if (IsVt(stack[k])) j=k; else j=k-1;
      while (IsHigherThan(stack[j],r))
      {
         int q;
         do {
               q=stack[j];
               if (IsVt(stack[j-1])) j--; else j-=2;
             } while(!IsLowerThan(stack[j],q);
         NewVn=Reduce(stack[j+1],stack[k],k-j);
         k=j+1;                    /* reduce the leftmost prime phrase */
         stack[k]=NewVn;    /* it is stack[j+1] stack[j+2] … stack[k] */
      } /*end of while*/
      if (IsLowerThan(stack[j],r)) || IsEqualTo(stack[j],r))
         stack[++k]=r;
      else return ERROR;
   } while (r!='#');
   return RIGHT;
```

}

程序 3 给出的仅是采用算符优先分析方法的示意性识别算法，还有许多工作要做。如：首先要确定一种合适的数据结构，以便构造所给文法 G2[<算术表达式>]的机内表示。然后，构造该文法的算符优先关系矩阵，在此可根据算术表达式中各算符的优先级和结合性，直接手工构造算符优先关系。最后，编写程序 3 中所用到的各个函数，完成整个算符优先语法分析器的开发。

5、SLR(1)分析器的开发

首先，对于分析对象，即算术表达式的文法 G2[E]，引入一个新的开始符号 E’，得到 G2[E]的拓广文法 G2’[E’]：

0. E’→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→(E)　　8. F→i

求出文法中各个非终结符号的 FOLLOW 集如下：

Follow(E)={#,), +, -}

Follow(T)={#,), +, -, *, /}

Follow(F)={#,), +, -, *, /}

然后，根据文法 G2’[E’]构造识别其全部活前缀的 DFA，以便据此构造 SLR(1)分析表，参见表 3。

表 3 G2’[E’]的 SLR(1)分析表

状态	ACTION								GOTO		
	(	)	+	-	*	/	i	#	E	T	F
0	S4						S5		1	2	3
1			S6	S7				Acc			
2		R3	R3	R3	S8	S9		R3			
3		R6	R6	R6	R6	R6		R6			
4	S4						S5		10	2	3
5		R8	R8	R8	R8	R8		R8			
6	S4						S5			11	3
7	S4						S5			12	3
8	S4						S5				13
9	S4						S5				14
10		S15	S6	S7							
11		R1	R1	R1	S8	S9		R1			
12		R2	R2	R2	S8	S9		R2			
13		R4	R4	R4	R4	R4		R4			
14		R5	R5	R5	R5	R5		R5			
15		R7	R7	R7	R7	R7		R7			

最后，编程实现 SLR(1)分析表的驱动程序，即开发 LR 分析器的总控程序，完成对算术表达式的识别。

四、扩展实验

1、对所给算术表达式的文法 G2，完成两种以上不同的语法分析程序的设计与实现。

2、在 G2 的基础上，适当增加功能，如进一步选择高级语言中的赋值语句、复合语句、流程控制语句等其它类型的语法结构作为分析对象。

3、对基本实验内容进行扩充，增加语法分析过程的可视化展现功能。

4、研读 GCC，CLANG 等开源编译器的语法分析部分，分析其程序结构、实现方法、识别的语法结构等。

5、其它自选题目。注意自选扩展实验题目须经过实验指导教师同意并备案。

实验三　语义分析程序设计与实现

一、实验目的

在实现词法、语法分析程序的基础上，编写相应的语义子程序，进行语义处理，加深对语法制导翻译原理的理解，进一步掌握将语法分析所识别的语法范畴变换为某种中间代码（四元式）的语义分析方法，并完成相关语义分析器的代码开发。

二、基本实验内容及要求

对文法 G2[<算术表达式>]中的产生式添加语义处理子程序，完成运算对象是简单变量（标识符）和无符号数的四则运算的计值处理，将输入的四则运算转换为四元式形式的中间代码。

输入：包含测试用例（由标识符、无符号数和+、－、*、/、(、)构成的算术表达式）的源程序文件。

输出：将源程序转换为中间代码形式表示，并将中间代码序列输出到文件中。若源程序中有错误，应指出错误信息。

要求：

1、结合实验一和实验二中的相关内容，完成语法制导翻译的程序设计。

2、对完成的编译系统进行全面测试，供测试的例子应包括符合语义规则的语句，以及分析程序能够判别的若干错例，并给出执行结果。

三、参考实现方法

1、语法制导翻译一般实现方法说明

语法制导翻译模式是在语法分析的基础上，增加语义操作来实现的，实际上是对前后文无关文法的一种扩展。一般而言，首先需要根据进行的语义分析工作，完成对给定文法的必要拆分和语义动作的编写，从而为每一个产生式都配备相应的语义子程序，以便在进行语法分析的同时进行语义解释。即在语法分析过程中，每当用一个产生式进行推导或归约时，语法分析程序除执行相应的语法分析动作之外，还要调用相应的语义子程序，以便完成生

成中间代码、查填有关表格、检查并报告源程序中的语义错误等工作。每个语义子程序需指明相应产生式中各个符号的具体含义，并规定使用该产生式进行分析时所应采取的语义动作。这样，语法制导翻译程序在对源程序从左到右进行的一遍扫描中，既完成语法分析任务，又完成语义分析和中间代码生成方面的工作。本实验要求从编译器的整体设计出发，重点通过对实验二中语法分析程序的扩展，完成一个编译器前端程序的编写、调试和测试工作，形成一个将源程序翻译为中间代码序列的编译系统。

2、基于递归下降法的语法制导翻译

对文法 G2[<算术表达式>]在利用递归下降法进行语法分析的同时，生成四元式形式的中间代码序列。其语法制导翻译程序的核心部分（指表达式 E、项 T 和因式 F 的处理）的算法思想，可用程序 4 所示的框架描述。

程序 4　利用递归下降法生成简单算术表达式的四元式序列

```
E( )   /*  识别<算术表达式> */
{
   E1_PLACE=T( );   /*调用 T( )分析产生算术表达式计算的第一项 E1*/
   while (SYM='+' || SYM='-')
   {
      ADVANCE;   /*使输入串指针指向下一个输入符号，即调用扫描器读入下一个单词符号*/
      E2_PLACE=T( );   /*调用 T( )分析产生算术表达式计算的第二项*/
      T1=NewTemp( );   /*分配一个新的临时变量，以便存储计算结果*/
      GEN(±, E1_PLACE, E2_PLACE, T1);   /*根据所给实参产生一个四元式，送入四元式表*/
      E1_PLACE=T1;   /*将计算结果作为下一次表达式计算的第一项*/
   }
   return E1_PLACE;
}

T( )   /*  识别<项>*/
{
   T1_PLACE=F( );
   while (SYM='*' || SYM='/')
   {
      ADVANCE;
      T2_PLACE=F( );
      T1=NewTemp( );
      GEN(*/, T1_PLACE, T2_PLACE, T1);
      T1_PLACE=T1;
   }
   return T1_PLACE;
}

F( )   /*  识别<因式>*/
{
   if (SYM='标识符')   /*在此设运算对象 i 为标识符，即简单变量*/
```

```
  {
    ADVANCE;
    return Entry(i.词文);   /*以标识符的词文为名字查、填符号表，可理解为返回标识符的值*/
  }
  else
       if ( SYM='(' )
         {
           ADVANCE;
           PLACE=E( );
           if ( SYM=')' )
              {
                ADVANCE;
                return PLACE;
              }
           else ERROR;   /*例如：输出“缺少‘)’错误”*/
         }
       else ERROR;   /*例如：输出“算术表达式应以‘标识符’或‘(’开头”*/
}
```

说明：

① 程序四中出现的主要语义变量和辅助函数的功能为：

E1_PLACE：文法符号 E1 的一个语义属性，用于描述变量在符号表中登记项的序号（>0 时）或临时变量的编号（<0 时），可理解为代表其值。

void GEN(char *Op, char *Arg1, char *Arg2, char *Result)：用来生成一个四元式，将其送到四元式表中。参考代码如下：

```
void GEN(char *Op, char *Arg1, char *Arg2, char *Result)
{
  strcpy (pQuad[NXQ].op, Op);  /*pQuad 为全局变量，是用于存放四元式的数组*/
  strcpy (pQuad[NXQ].arg1, Arg1);
  strcpy (pQuad[NXQ].arg2, Arg2);
  strcpy (pQuad[NXQ].result, Result);
  NXQ++;  /*全局变量 NXQ 用于指示所要产生的下一个四元式的编号*/
}
```

char *NewTemp(void)：产生临时变量的函数，每调用一次都产生一个新的临时变量，并返回这个新的临时变量名，临时变量名产生的顺序依次为 T1、T2、T3、……。例如：

```
char *NewTemp(void)  /*产生一个临时变量*/
{
  char *TempID=(char *)malloc(MAXLENGTH);
  sprintf (TempID, “T%d”, NXTemp++);  /*整型变量 NXTemp 指示临时变量的编号*/
  return TempID;
}
```

② 注意修改已在前面两个实验中完成的词法、语法分析器，以便在此基础上进行语义分析。如有必要，从总体设计的角度出发，重新定义整个系统所需要的一些数据结构、宏和全局变量等，如程序 5。

程序 5 定义与词法分析器的接口

```
union WORDCONTENT   /*存放单词词文内容的联合*/
{
   char Val1[MAXLENGTH];   /*对于标识符、关键字或由一个以上字符组成的复合单词结构的词文（如:=、<=、<>等），采用字符串作为其值的内部表示*/
   int Val2;   /*对于数值常量采用其二进制值作为它们的内部表示*/
   float Val3;
   char Val4;   /*记录由一个字符组成的单词的词文(如+、-、*、/ 等)*/
};

struct WORD   /*单词的二元式形式表示*/
{
   int Sym;   /*单词的类别编码*/
   union WORDCONTENT value;   /*单词的值*/
}word;   /*word 存放由词法分析程序扫描得到的二元式形式的单词*/

/*定义语法（语义）分析器的接口*/
struct QUATERNION   /*四元式表的结构*/
{
   char op[MAXLENGTH];   /*操作符*/
   char arg1[MAXLENGTH];   /*第一个操作数*/
   char arg2[MAXLENGTH];   /*第二个操作数*/
   char result[MAXLENGTH];   /*运算结果*/
}*pQuad;   /*存放四元式的数组*/
```

这样用于处理表达式、项和因式的函数原型可分别写为 char *E(void)、char *T(void)和 char *F(void)。在此仅给出处理表达式的函数的参考代码见程序 6。

程序 6 处理表达式的函数的参考代码

```
char *E(void)
{
   char opp[2], *E1_place, *E2_place, *Temp_place;
   E1_place=T( );
   while (word.Sym==PL || word.Sym==MI)   /*单词为'+'或'－'*/
   {
      sprintf(opp, "%c", word.value.Val4);
      scanner( );   /*调扫描器读下一个单词，并将其二元式形式的表示送入全局变量 word 中*/
      E2_place=T( );
      Temp_place=NewTemp( );
      GEN(opp, E1_place, E2_place, Temp_place);
      E1_place=Temp_place;
   }
   return E1_place;
}
```

附件：实验报告目录（参考）

河北工业大学

《编译原理实验》报告

班级：_____

学号：_____

姓名：_____

完成日期：_____

目　录