天津理工大学 实验报告

学院名称：计算机科学与工程学院

姓名	王路耀	学号	20152216	专业	计算机科学与技术
班级	15 级 2 班	实验项目	实验三：语义分析与中间代码生成		
课程名称		编译原理		课程代码	0668056
实验时间		2018 年 6 月 6 日　第*、*节 2018 年 6 月 11 日　第*、*节		实验地点	软件实验室 7-219 软件实验室 7-219

实验成绩考核评定分析

实验过程综合评价 30 分	实验目标结果评价 20 分	程序设计规范性评价 20 分	实验报告完整性评价 30 分	实验报告雷同分析分类标注	实验成绩
■实验过程认真专注，能独立完成设计与调试任务 30 分 ■实验过程认真，能较好完成设计与编成调试任务 25 分 ■实验过程较认真，能完成设计与编成调试任务 20 分 ■实验过程态度较好，基本完成设计与编成调试任务 15 分 ■实验过程态度欠端正，未完成设计与编成调试任务 10 分	■ 功能完善，且人机交互界面友好 20 分 ■ 满足功能要求，但人机交互界面一般 15 分 ■基本满足功能需求，人机交互界面欠缺 10 分 ■功能缺失 5 分	■程序易读性好 20 分 ■程序易读性较好 15 分 ■程序易读性欠缺 10 分 ■程序易读性较差 5 分 **注：易读性要求标识符命名见名知意，程序编制采用嵌套方式，层次结构清晰可读，关键部分具有简明注释。	■报告完整 30 分 ■报告较完整 25 分 ■报告内容一般 20 分 ■报告内容极少 10 分	凡雷同报告将不再重复评价前四项考核内容，实验成绩将按低学号雷同学生成绩除雷同人数计算而定。 标记为： **S 组号-人数(组分)**	前四项评价分数之总和 (** 雷同报告按第五项标准核算**)

实验内容：

已知 G[E]：　E→E+T | E-T | T

T→T*F | T/F | F

F→P^F | P

P→(E) | i

要求构造出符合语义分析要求的属性文法描述，并在完成实验二（语法分析）的基础上，进行语义分析程序设计，最终输出与测试用例等价的四元式中间代码序列。

实验目的：

1．掌握语法制导翻译的基本功能，巩固对语义分析的基本功能和原理的认识；

3．能够基于语法制导翻译的知识进行语义分析，掌握文法规则相应语义动作的设计方法；

5．理解并处理语义分析中的异常和错误。

实验要求：

1．在实验二的基础上，实现语法制导翻译功能，输出翻译后所得四元式序列；

2．要求详细描述所选分析方法进行制导翻译的设计过程；

3．完成对所设计分析器的功能测试，并给出测试数据和实验结果；

4．为增加程序可读性，请在程序中进行适当注释说明；

5．整理上机步骤，总结经验和体会，认真完成并按时提交实验报告。

算符优先文法

构造 FIRSTVT、LASTVT、算符优先关系表:

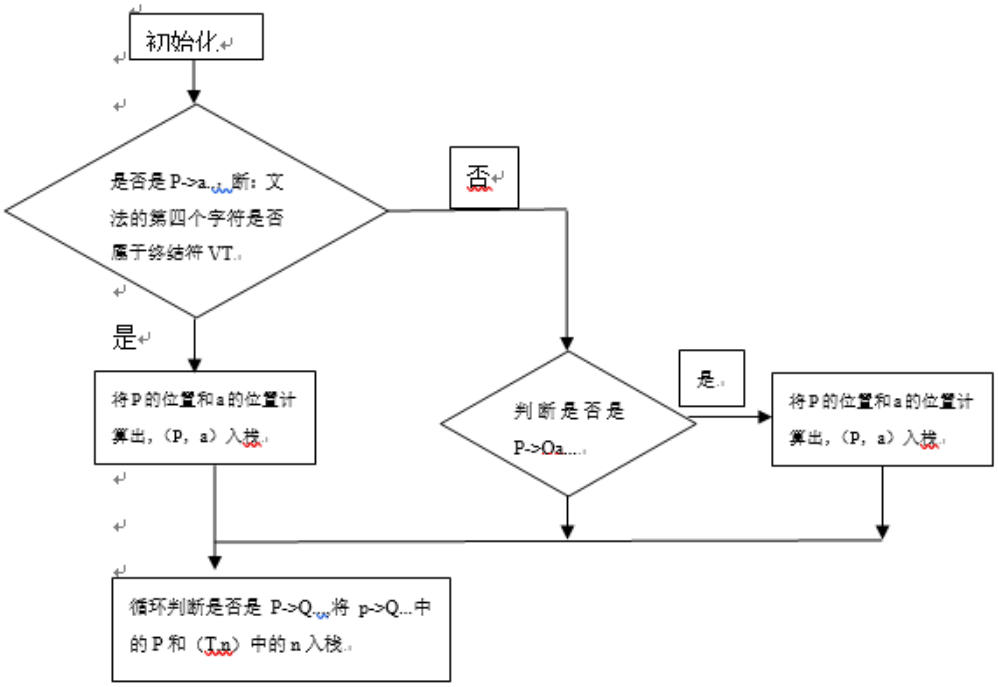
在构造 FIRSTVT 表时，通过循环找出每条产生式中的非终结符的 FIRSTVT 集，并把该非终结符和终结符压栈，设置标志位，标志一对非终结符和终结符具有对应关系。

LASTVT 表的构造则是将求 FIRSTVT 的过程翻转过来，可以仅仅将函数中的参数稍作修改就能够完成。

算符优先关系表是一个二维数组，用来存放任意两个终结符之间的优先关系。首先构造表头，通过扫描所有产生式将终结符不重复的存放在一个一维数组中并置为优先关系表的行和列，并将优先关系表其他内容全部初始化为空。接着遍历所有产生式，找出任意两个终结符之间存在的关系（可以没有关系），并判断任意两终结符是否至多存在一种优先关系，如发现优先关系表不为空，则证明该文法不是算符优先文法，否则，将相应的关系填入到相应的行列对应的单元中。

输入串分析过程的设计。首先将大于、小于、等于和无关系分别定义成一种类型的数据表示，通过查询符号栈栈顶以及当前符号之间的优先关系来判断移进和规约的操作。

计算 FirstVT 和 LastVT



规约过程

开始时，将"#"和文法开始符号放入栈底。总控程序在任何时候都是根据栈顶符号 X 和当前的输入符号进行工作的，它与文法无关

总控程序根据现行栈顶符号 X 和当前输入符号 a，执行下列三种动作之一:

1. 若 X＝a＝‘＃’，则宣布分析成功，停止分析。
2. 若 X＝a ¹ ‘＃’，则把 X 从 STACK 栈顶逐出，让指针指向下一个输入符号。

3.若 X 是一个非终结符，则查看分析表 M。

若 M[X，a]中存放着关于 X 的一个产生式，把 X 逐出 STACK 栈顶，把产生式的右部符号串按反序一一推进 STACK 栈(若右部符号为 e，则意味不推什么东西进栈)。在把产生式的右部符号推进栈的同时应做这个产生式相应的语义动作。

若 M[X，a]中存放着“出错标志”，则调用出错诊察程序 ERROR。

文法表示：

S→v=E|E?|clear

E→E+T|E-T|T

T→T*F|T/F|F

F→ (E)|v|c

单词种别码设计：

= 1

? 2

+ 3

- 4

*** 5**

/ 6

(7

) 8

v 9

c 10

clear 11

 # 12

N 13

可归约串语义解释：

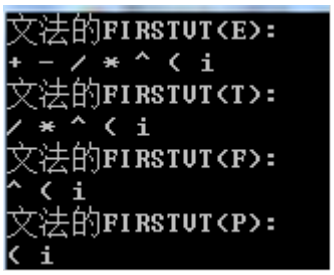
变量归约；常量归约；运算归约；括号归约；

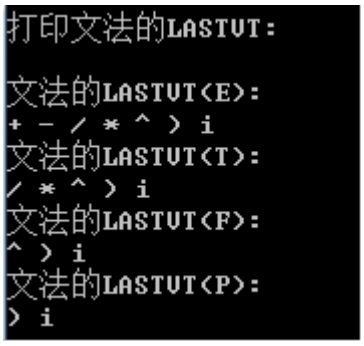
赋值语句；输出语句;清除语句。

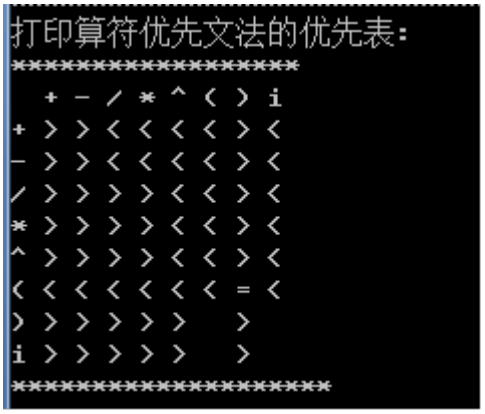
```
//要分析的文法
string
E[10]={"E->E+T","E->E-T","E->T","T->T*F","T->T/F","T->F","F->P^F","F->P","
P->(E)","P->i"};
char    upper[4]={'E','T','F','P'};
char    lower[8]={'+','-','/','*','^','(',')','i'};

void insert(char T,char n);      //入栈操作
bool SignArray[20][20];          //标记数组，标记每一个是 FIRSTVT、LastVT
void CreateFIRSTVT();      //构造 FIRSTVT 的函数
void PrintFirstvt();
bool IsT(char);              //判断是否为终结符
      bool IsN(char);             //判断是否为非终结符
```

实验截图：


文法的FIRSTVT(E):
+ - / * ^ (i
文法的FIRSTVT(T):
/ * ^ (i
文法的FIRSTVT(F):
^ (i
文法的FIRSTVT(P):
(i


打印文法的LASTVT:
文法的LASTVT(E):
+ - / * ^) i
文法的LASTVT(T):
/ * ^) i
文法的LASTVT(F):
^) i
文法的LASTVT(P):
) i


打印算符优先文法的优先表:
****
+ - / * ^ ( ) i
+ > > < < < < > <
- > > < < < < > <
/ > > > > < < > <
* > > > > < < > <
^ > > > > < < > <
(< < < < < < = <
) > > > > > > >
i > > > > > > >
****

```
**********************************
请输入一个句型,以#结束!
i*(i-i)#
读入 :i
#i
i 规约为 M
#M
读入 :*
#M*
读入 :(
#M*(
读入 :i
#M*(i
i 规约为 M
#M*(M
读入 :-
#M*(M-
读入 :i
#M*(M-i
i 规约为 M
#M*(M-M
M-M 规约为 M
#M*(M
读入 :)
#M*(M)
(M) 规约为 M
#M*M
M*M 规约为 M
#M
读入 :#
#M#
successed!
```

```
successed!
(0)    ( +, i, i, T1 )
(1)    ( *, T1, #, T2 )
(2)    ( :=, T2, _, i )
请按任意键继续. . .
```

重要代码

```
void zh(int  n)
{   if(n>=300)
        cout<<"T"<<n-300;
    else      cout<<char(n);
}
int p(char  n)
{
    if(n=='=')
        return  0;
    else if(n==')')
        return  10;
    else if(n=='+'||n=='-')
        return  20;

    else if(n=='*'||n=='/')
        return  30;
    else if(n=='(')
        return  40;
    else return -1;

}
class CreateFirstvt                    //构造 FIRSTVT 的类
{
public:
      CreateFirstvt();
    ~CreateFirstvt()
    {}
    void insert(char  T,char  n);
    bool SignArray[20][20];       //标记数组，标记每一个是 FIRSTVT
    void CreateFIRSTVT();       //构造 FIRSTVT 的函数
      void PrintFirstvt();
    bool IsT(char);              //判断是否为终结符
    bool IsN(char);              //判断是否为非终结符
private:
    string stack[50];      //栈
    int top;
};

CreateFirstvt::CreateFirstvt()//初始化 SignArray
{
    top=0;
```

```
    for(int i=0;i<20;i++)
    {
        for(int j=0;j<20;j++)
        {
            SignArray[i][j]=false;
        }
    }
}
void CreateFirstvt::insert(char T,char n)
{
    for(int j=0;j<4;j++)//判断是否为 VT，并标记位置
    {
        if(T==upper[j])
            break;
    }
    for(int k=0;k<8;k++)//判断是否为 VN，并标记位置
    {
        if(n==lower[k])
            break;
    }
    if(SignArray[j][k]==false)
    {
        SignArray[j][k]=true;
        stack[top]="(";
        stack[top].append(1,T);
        stack[top]+=",";
        stack[top].append(1,n);
        stack[top]+=")";
        top++;                              //不在栈中，则将“(P,a)”入栈
    }
}
void CreateFirstvt::CreateFIRSTVT()
{
    for(int i=0;i<10;i++)
    {
        if(IsT(E[i].at(3)))                 //处理 P->a.....这种情况
        {
            char pos1,pos2;
            pos1=E[i].at(0);                //P->a... 将 P 给 pos1
            pos2=E[i].at(3);                //p->a... 将 a 给 pos2
            insert(pos1,pos2);                  //a 属于 P 的首符集
        }
        if(IsN(E[i].at(3)) && E[i].length()>4 && IsT(E[i].at(4)))       //处理 P->Qa.....这
种情况
```

```
        {
            char pos1,pos2;
            pos1=E[i].at(0);
            pos2=E[i].at(4);
            insert(pos1,pos2);
        }
    }
    while(top!=0)                                   //处理 P->Q.....这种情况
    {
        string PopElement=stack[--top];
        stack[top]="";
        char pos1,pos2;
        for(i=0;i<10;i++)
        {
            if(PopElement.at(1)==E[i].at(3))            //(T,n)中的 T  和  p->Q...中的 Q  相
等
            {
                pos1=E[i].at(0);                        //p->Q...的 P 和(T,n)中的 n 入栈
                pos2=PopElement.at(3);
                insert(pos1,pos2);
            }
        }
    }
}
void CreateFirstvt::PrintFirstvt()
{
    cout<<"*********************************************"<<endl;
    cout<<"打印文法的 FIRSTVT:"<<endl<<endl;
    cout<<"文法的 FIRSTVT(E):"<<endl;
    for(int i=0;i<4;i++)
    {
        for(int j=0;j<8;j++)
        {
            if(SignArray[i][j]==true  && upper[i]=='E')
            {
                cout<<lower[j]<<"  ";
            }
        }
        if(upper[i]=='E')
            break;
    }
    cout<<endl;
    cout<<"文法的 FIRSTVT(T):"<<endl;
    for(i=0;i<4;i++)
```

```
    {
        for(int j=0;j<8;j++)
        {
            if(SignArray[i][j]==true  && upper[i]=='T')
            {
                cout<<lower[j]<<"  ";
            }
        }
        if(upper[i]=='T')                              //如果找到了，就退出本次循环
            break;
    }
    cout<<endl;
    cout<<"文法的 FIRSTVT(F):"<<endl;
    for(i=0;i<4;i++)
    {
        for(int j=0;j<8;j++)
        {
            if(SignArray[i][j]==true  && upper[i]=='F')
            {
                cout<<lower[j]<<"  ";
            }
        }
        if(upper[i]=='F')
            break;
    }
    cout<<endl;
    cout<<"文法的 FIRSTVT(P):"<<endl;
    for(i=0;i<4;i++)
    {
        for(int j=0;j<8;j++)
        {
            if(SignArray[i][j]==true  && upper[i]=='P')
            {
                cout<<lower[j]<<"  ";
            }
        }
        if(upper[i]=='P')
            break;
    }
      cout<<endl;
}
bool CreateFirstvt::IsT(char  ch)
{
    for(int i=0;i<8;i++)
```

```
    {
        if(ch==lower[i])
        {
            return true;
        }
    }
    return false;
}
bool CreateFirstvt::IsN(char ch)
{
    for(int i=0;i<4;i++)
    {
        if(ch==upper[i])
        {
            return true;
        }
    }
    return false;
}

class CreateLastvt                          //构造 LASTVT 的类
{
public:
      CreateLastvt();
    ~CreateLastvt()
    {}
    void insert(char T,char n);
    bool SignArray[20][20];
    void CreateLASTVT();        //构造 LASTVT 的函数
      void PrintLastvt();
    bool IsT(char);              //判断是否为终结符
    bool IsN(char);              //判断是否为非终结符
private:
    string stack[50];
    int top;
};
CreateLastvt::CreateLastvt()
{
    top=0;
    for(int i=0;i<20;i++)
    {
        for(int j=0;j<20;j++)
        {
            SignArray[i][j]=false;
```

```
        }
    }
}
void CreateLastvt::insert(char  T,char n)
{
    for(int  j=0;j<4;j++)
    {
        if(T==upper[j])
            break;                  //查出非终结符的位置
    }
    for(int  k=0;k<8;k++)
    {
        if(n==lower[k])
            break;                  //查出终结符的位置
    }
    if(SignArray[j][k]==false)    //如果再 SignArray 中没有，则将其入栈
    {
        SignArray[j][k]=true;
        stack[top]="(";
        stack[top].append(1,T);
        stack[top]+=",";
        stack[top].append(1,n);
        stack[top]+=")";
        top++;                      //将"(P,a)"入栈
    }
}
void CreateLastvt::CreateLASTVT()
{
    for(int  i=0;i<10;i++)
    {
        if(IsT(E[i].at(E[i].length()-1))==true)                    //处理 P->.....a 这种情况
        {
            char pos1,pos2;
            pos1=E[i].at(0);
            pos2=E[i].at(E[i].length()-1);
            insert(pos1,pos2);
        }
        if(IsT(E[i].at(E[i].length()-2))==true  && IsN(E[i].at(E[i].length()-1))==true)      //
处理 P->.......aQ 这种情况
        {
            char pos1,pos2;
            pos1=E[i].at(0);
            pos2=E[i].at(E[i].length()-2);
            insert(pos1,pos2);
```

```
        }
    }
    while(top!=0)                                   //处理 P->.....Q 这种情况
    {
        string PopElement=stack[--top];
        stack[top]="";
        char pos1,pos2;
        for(i=0;i<10;i++)
        {
            if(PopElement.at(1)==E[i].at(E[i].length()-1))
            {
                pos1=E[i].at(0);
                pos2=PopElement.at(3);
                insert(pos1,pos2);
            }
        }
    }
}
void CreateLastvt::PrintLastvt()
{
    cout<<"****************************************"<<endl;
    cout<<"打印文法的 LASTVT:"<<endl<<endl;
    cout<<"文法的 LASTVT(E):"<<endl;
    for(int i=0;i<4;i++)
    {
        for(int j=0;j<8;j++)
        {
            if(SignArray[i][j]==true  && upper[i]=='E')
            {
                cout<<lower[j]<<"  ";
            }
        }
        if(upper[i]=='E')
            break;
    }
    cout<<endl;
    cout<<"文法的 LASTVT(T):"<<endl;
    for(i=0;i<4;i++)
    {
        for(int j=0;j<8;j++)
        {
            if(SignArray[i][j]==true  && upper[i]=='T')
            {
                cout<<lower[j]<<"  ";
```

```
            }
        }
        if(upper[i]=='T')
            break;
    }
    cout<<endl;
    cout<<"文法的 LASTVT(F):"<<endl;
    for(i=0;i<4;i++)
    {
        for(int j=0;j<8;j++)
        {
            if(SignArray[i][j]==true  && upper[i]=='F')
            {
                cout<<lower[j]<<"  ";
            }
        }
        if(upper[i]=='F')
            break;
    }
    cout<<endl;
    cout<<"文法的 LASTVT(P):"<<endl;
    for(i=0;i<4;i++)
    {
        for(int j=0;j<8;j++)
        {
            if(SignArray[i][j]==true  && upper[i]=='P')
            {
                cout<<lower[j]<<"  ";
            }
        }
        if(upper[i]=='P')
            break;
    }
      cout<<endl;
}
bool CreateLastvt::IsT(char ch)
{
    for(int i=0;i<8;i++)
    {
        if(ch==lower[i])
        {
            return true;
        }
    }
```

```
    return false;
}
bool CreateLastvt::IsN(char ch)
{
    for(int i=0;i<4;i++)
    {
        if(ch==upper[i])
        {
            return true;
        }
    }
    return false;
}


class CreatePreTable
{
public:
    CreatePreTable();
    ~CreatePreTable()
    {}
    void CreatePreCedentTable();
    void PrintPreTable();
    bool IsT(char);
    bool IsN(char);
    int GetLocation(char);
private:
    char PreTable[8][8];       //算符优先关系表
};

CreatePreTable::CreatePreTable()//初始化
{
    for(int i=0;i<8;i++)
        for(int j=0;j<8;j++)
        {
            PreTable[i][j]='  ';
        }
}
bool CreatePreTable::IsT(char ch)
{
    for(int i=0;i<8;i++)
    {
        if(ch==lower[i])
            return true;
```

```
    }
}
bool CreatePreTable::IsN(char ch)
{
    for(int i=0;i<4;i++)
    {
        if(ch==upper[i])
            return true;
    }
}

void CreatePreTable::CreatePreCedentTable()//
{
    CreateFirstvt firstvt;
    firstvt.CreateFIRSTVT();
    CreateLastvt lastvt;
    lastvt.CreateLASTVT();

    for(int i=0;i<10;i++)
        for(int j=1;j<(E[i].length()-3);j++)
        {
            if(IsT(E[i].at(j+2))==true)
            {
                if(IsT(E[i].at(j+3))==true)
                {
                    PreTable[GetLocation(E[i].at(j+2))][GetLocation(E[i].at(j+3))]='=';//
判定为等于关系
                }
                if(IsN(E[i].at(j+3))==true  && j<=(E[i].length()-5)  &&
IsT(E[i].at(j+4))==true)
                {
                    PreTable[GetLocation(E[i].at(j+2))][GetLocation(E[i].at(j+4))]='=';//
判定为等于关系
                }
                if(IsN(E[i].at(j+3))==true)
                {
                    int loc=GetLocation(E[i].at(j+2));
                    for(int m=0;m<4;m++)
                    {
                        for(int k=0;k<8;k++)
                        {
                            if(firstvt.SignArray[m][k]==true  &&
upper[m]==E[i].at(j+3))
                            {
```

```
                                PreTable[loc][k]='<';
                            }

                        }
                        if(upper[m]==E[i].at(j+3))
                            break;
                    }
                }
            }
            else
            {
                if(IsT(E[i].at(j+3))==true)
                {
                    int loc=GetLocation(E[i].at(j+3));
                    for(int m=0;m<4;m++)
                    {
                        for(int k=0;k<8;k++)
                        {
                            if(lastvt.SignArray[m][k]==true &&
upper[m]==E[i].at(j+2))
                            {
                                PreTable[k][loc]='>';
                            }
                        }
                        if(upper[m]==E[i].at(j+2))
                            break;
                    }
                }
            }
        }
}
void CreatePreTable::PrintPreTable()
{
    cout<<"****************************************"<<endl;
    cout<<"打印算符优先文法的优先表:"<<endl;
    cout<<"******************"<<endl;
    cout<<"    "<<"+ - / * ^ ( ) i"<<endl;
    cout<<"+ ";
    for(int j=0;j<8;j++)
    {
        cout<<PreTable[0][j]<<"  ";
    }
    cout<<endl;
    cout<<"- ";
```

```
    for(j=0;j<8;j++)
    {
        cout<<PreTable[1][j]<<"  ";
    }
    cout<<endl;
    cout<<"/ ";
    for(j=0;j<8;j++)
    {
        cout<<PreTable[2][j]<<"  ";
    }
    cout<<endl;
    cout<<"* ";
    for(j=0;j<8;j++)
    {
        cout<<PreTable[3][j]<<"  ";
    }
    cout<<endl;
    cout<<"^ ";
    for(j=0;j<8;j++)
    {
        cout<<PreTable[4][j]<<"  ";
    }
    cout<<endl;
    cout<<"( ";
    for(j=0;j<8;j++)
    {
        cout<<PreTable[5][j]<<"  ";
    }
    cout<<endl;
    cout<<") ";
    for(j=0;j<8;j++)
    {
        cout<<PreTable[6][j]<<"  ";
    }
    cout<<endl;
    cout<<"i ";
    for(j=0;j<8;j++)
    {
        cout<<PreTable[7][j]<<"  ";
    }
    cout<<endl;
    cout<<"**********"<<endl;
    cout<<"****************************************"<<endl;
}
```

```
int CreatePreTable::GetLocation(char ch)//获得在 VN 的位置
{
    for(int i=0;i<8;i++)
    {
        if(lower[i]==ch)
        {
            return i;
        }
    }
}

class   stack
{
private:
    int size;
    char   array[length];
public:
    stack()
    {
        size=0;
    }
    void push(char ch)
    {
        if(size<length)
        {
            array[size]=ch;
             size++;
        }
        else
            cout<<"overflow!"<<endl;
    }
    int   pop(char   ch[],int len)
    {
        if(size-len>=0)
        {
            for(int i=0;i<len;i++)
                ch[i]=array[size-len+i];
                   size-=len;
              return len;
        }
        else
        {
            cout<<"参数错误!"<<endl;
```

```
                return 0;
            }
        }
char peek(int pos)
{
    if(pos>=0&&pos<size)
        return array[pos];
     return '\0';
}
void peekall()
{
    for(int i=0;i<getsize();i++)
        cout<<peek(i);
    cout<<endl;
}
int getsize()
{
    return size;
}
};
char    guiyue(char ch[])
{
   return 'E';
}
int isnumch(char  ch)
{
    return (ch>='0'&&ch<='9'||ch>='a'&&ch<='z');
}


int getrank(char ch1,char ch2)
{
    if(isnumch(ch1))
        ch1='i';
    if(isnumch(ch2))
        ch2='i';
    if(ch1=='+')
    {
        if(ch2=='+'||ch2=='-'||ch2=='#'||ch2==')')
            return 1;
        else
            return -1;
    }
    if(ch1=='-')
```

```
    {
        if (ch2=='+'||ch2=='-'||ch2=='#'||ch2==')')
            return 1;
        else
            return -1;
    }
    if(ch1=='*'||ch1=='^')
    {
        if(ch2=='^'||ch2=='('||ch2=='i')
            return -1;
        else
            return 1;
    }
    if(ch1=='i'||ch1==')')
    {
        if(ch2=='i'||ch2=='(')
            return 2;
        else
            return 1;
    }
    if(ch1=='(')
    {
        if(ch2=='#')
            return 2;//error
        else
            if(ch2==')')
                return 0;
            return -1;
    }
    if(ch1=='#')
    {
        if(ch2=='#')
            return 0;
        else
            if(ch2==')')
                return 2;
            else
                return -1;
    }
    return 2;
    //0 表示等于，1 表示大于，-1 表示小于，2 表示没有优先关系
}
int isvt(char ch)//区别参加规约的资格
{
```

```
    if(ch>='a'&&ch<='z')
        return 1;
    if(ch>='('&&ch<='-')
         return 1;
    if(ch=='#'||ch=='^')
        return 1;
    if(ch>='0'&&ch<='9')
        return 1;
     return 0;

}


void main()
{
    CreateFirstvt firstvt;
    firstvt.CreateFIRSTVT();
    firstvt.PrintFirstvt();
      CreateLastvt lastvt;
    lastvt.CreateLASTVT();
    lastvt.PrintLastvt();
    CreatePreTable pretable;
    pretable.CreatePreCedentTable();
    pretable.PrintPreTable();


    char a1[100];
    int b[100];
    int atop=0,btop=0;
    string str;
    //char    str[maxsize];
    stack   s;
    //int    len;
    cout<<"请输入一个句型,以#结束!"<<endl;
    cin>>str;
    //len=int(strlen(str));
    s.push('#');
    int k=s.getsize()-1,t=0,j;
    char a=str[0];
    while(a!='#')
    {
        a=str[t];
        if(isvt(s.peek(k)))
            j=k;
```

```
        else
            j=k-1;
        while(isvt(a)&&getrank(s.peek(j),a)==1)
        {
            int h=j,low=j-1;
            if(!isvt(s.peek(low)))
                low--;
            while(getrank(s.peek(low),s.peek(h))!=-1)
            {
                h=low;
                low--;
                if(!isvt(s.peek(low)))
                    low--;
            }
            h=s.getsize()-1;
            low++;
            int len=h-low+1;
            char    ch[10];
            for(int p=0;p<10;p++)
                ch[p]='\0';
            s.pop(ch,len);
            char c=guiyue(ch);
            s.push(c);
            cout<<ch<<"  规约为 "<<guiyue(ch)<<endl;
            s.peekall();
            j=s.getsize()-1;
            if(!isvt(s.peek(j)))
                j--;
        }
        if(!(a>='A'&&a<='Z')&&getrank(s.peek(j),a)==2)
        {
            cout<<"你的输入有错误!"<<endl;
            cout<<"错误为第 "<<t+1<<"个字符 :"<<str[t]<<endl;
            exit(0);
        }
        else
        {
            cout<<"读入 :"<<a<<endl;
            s.push(a);s.peekall();
            t++;
            k=s.getsize()-1;
        }
    }
    char temp[10];
```

```
      s.pop(temp,3);
    if(s.getsize()==0)
         cout<<"successed!"<<endl;
        else
             cout<<"fail!"<<endl;
        //cin>>k;
//////////////////求四元式////////////////////////////////

    for(int i=0;i<str.length()-1;i++)
    {
        if(p(str[i])==-1)
        b[btop++]=str[i];
        else if(atop>1)
        {
            if(str[i]=='('&&str[i+1]=='-')
            {

                cout<<"("<<ss<<")     ( "<<"uminus, ";zh(str[i+2]);cout<<", _,
"<<"T"<<n-300<<" )"<<endl;
                i=i+3;

                b[btop++]=n;
                n++;
                ss++;
            }
            else if(str[i]=='('&&str[i+1]!='-')
            {
                a1[atop++]=str[i];

            }
            else if(str[i]==')'&&p(str[i])<=p(a1[atop-1]))
            {
                while(a1[atop-1]!='(')
                {

                    cout<<"("<<ss<<")     ( "<<a1[--atop]<<", ";zh(b[btop-2]);cout<<",
";zh(b[btop-1]);cout<<", "<<"T"<<n-300<<" )"<<endl;
                    btop=btop-2;
                    b[btop++]=n;
                        n++;
                        ss++;

                }
                if(a1[atop-1]=='(')
```

```
                --atop;
            }
            else if(str[i]!=')'&&p(str[i])<=p(a1[atop-1]))
            {

                while(p(str[i])<=p(a1[atop-1])&&a1[atop-1]!='='&&a1[atop-1]!='(')
                {

                    cout<<"("<<ss<<")     ( "<<a1[--atop]<<",  ";zh(b[btop-2]);cout<<",
";zh(b[btop-1]);cout<<",  "<<"T"<<n-300<<" )"<<endl;
                    btop=btop-2;
                    b[btop++]=n;
                        n++;
                        ss++;
                }

                a1[atop++]=str[i];
            }

            else
            {
                a1[atop++]=str[i];


            }
        }
        else
        {
            a1[atop++]=str[i];
        }
    }

    if(str[str.length()-1]==')')
    {
        while(p(str[str.length()-1])<=p(a1[atop-1])&&a1[atop-1]!='(')
                {

                    cout<<"("<<ss<<")     ( "<<a1[--atop]<<",
";zh(b[btop-2]);cout<<",  ";zh(b[btop-1]);cout<<",  "<<"T"<<n-300<<" )"<<endl;
                    btop=btop-2;
                    b[btop++]=n;
                        n++;
                        ss++;
                }
```

```
                if(a1[atop-1]=='(')
                --atop;
            while(a1[atop-1]!='=')
            {
                {
                    cout<<"("<<ss<<")     ( "<<a1[--atop]<<", ";zh(b[btop-2]);cout<<",
";zh(b[btop-1]);cout<<",  "<<"T"<<n-300<<" )"<<endl;
                    btop=btop-2;
                    b[btop++]=n;
                        n++;
                        ss++;
                }
            }
    }

    else
    {
        b[btop++]=str[str.length()-1];
        cout<<"("<<ss<<")     ( "<<a1[--atop]<<", ";zh(b[btop-2]);cout<<",
";zh(b[btop-1]);cout<<",  "<<"T"<<n-300<<" )"<<endl;
        btop=btop-2;
        b[btop++]=n;
        n++;
        ss++;

    }
    cout<<"("<<ss<<") "<<"   ( :=, ";zh(b[--btop]);cout<<",  _,
";zh(b[--btop]);cout<<"  )"<<endl;
    system("pause");

}
```