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	实验二：语法分析		
课程名称		编译原理		课程代码	0668056
实验时间		2018 年 5 月 30 日　第*、*节 2018 年 6 月 4 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：

可选择 LL1 分析法、算符优先分析法、LR 分析法之一，实现如下表达式文法的语法分析器：

（1）E→E+T | E-T | T

（2）T→T*F | T/F | F

（3）F→P^F | P

（4）P→(E) | i

实验目的：

1．掌握语法分析的基本概念和基本方法；

2．正确理解 LL1 分析法、算符优先分析法、LR 分析法的设计与使用方法。

实验要求：

1．按要求设计实现能识别上述文法所表示语言的语法分析器，并要求输出全部分析过程；

2．要求详细描述所选分析方法针对上述文法的分析表构造过程；

3．完成对所设计语法分析器的功能测试，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．按软件工程管理模式完成实验报告撰写工作，最后需要针对实验过程进行经验总结；

6．认真完成并按时提交实验报告。

****** 实验报告内容详述 ******

一个语法分析器从词法分析器获得一个词素序列，并验证这个序列是否可以由源语言的文法生成。语法分析器会构造一棵语法分析树，并把它传递给编译器的其他部分进一步处理，在构建语法分析树的过程中，就验证了这个词素序列是否符合源语言的文法

LR 语法分析表

一个 LR 语法分析器的语法分析表由一个语法分析动作函数 ACTION 和一个转换函数 GOTO 组成：

ACTION 函数：ACTION 函数有两个参数，一个是状态 i，一个是终结符号（包括输入结束标记$）a，ACTION[i, a]有四种取值：

移入。如果状态 i 上有一个转移 a 到达状态 j，那么 ACTION[i, a]=移入 j；

归约。如果状态 i 上没有一个转移 a，那么 ACTION[i, a]=按照状态 i 上的产生式进行归约；

接受。语法分析器接受输入并完成语法分析过程；

报错。语法分析器在它的输入中发现一个错误并执行某个纠正动作。

GOTO 函数：实质上和项集的 GOTO 函数一样，只不过把项集替换成了状态。即：如果对项集的 GOTO 函数，有 GOTO[Ii, A]=Ij，那么对 LR 语法分析表的 GOTO 函数，有 GOTO[i, A]=j。

根据一个 LR(0)自动机，我们可以立马得出 LR 语法分析表的 GOTO 函数，但是，对 ACTION 函数，应用下面的规则计算：

在 LR(0)自动机中，如果项 A→ α ·a β 在项集 Ii 中并且 GOTO[Ii, a]=Ij，那么将 ACTION[i, a]设为“移入 j”；

在 LR(0)自动机中，如果项 A→ α ·在项集 Ii 中，那么对于 FOLLOW(A)中的所有 a，将 ACTION[i, a]设为“按照 A→ α 归约”，这里 A 不等于 S’；

在 LR(0)自动机中，如果项 S’ →S·在项集 Ii 中，那么将 ACTION[i, $]设为“接受”。

除此之外，将所有空白的 ACTION 和 GOTO 设为“报错”。

LR 语法分析算法

一个 LR 语法分析器由一个输入缓冲区、一个状态栈、一个语法分析表和一个结果输出组成，

状态栈维护一个状态序列 s0s1…sn，其中 sn 位于栈顶，每个状态 si 对应于 LR(0)状态机中的某个状态，并且除了初始状态之外，每个状态都有一个唯一的相关联的文法符号。也就是说，在 LR(0)状态机中，如果从 Ii 经过符号 α 转移到 Ij，那么状态 j 的关联符号为 α

用状态栈和剩余的输入符号串可以完整的表示语法分析器在某一刻的状态，这个状态本质上是反向最右推导中的某个句型。我们用(s0s1…sm, a1a2…an$)表示语法分析器的状态，并称其为语法分析器的格局。其中，第一个分量是状态栈中的状态序列（sm 是栈顶），第二个分量是剩余的输入符号串，如果把第一个分量中的每个状态替换为其关联的文法符号，再与第二个分量连接，就能得到反向最右推导中的一个句型。

假定 LR 语法分析器当前的格局为(s0s1…sm, aiai+1…an$)，在根据当前格局决定下一个

动作时，首先读入下一个输入的符号 ai 和栈顶的状态 sm，然后查询 LR 语法分析表中的条目 ACTION[sm, ai]，执行相应的动作：

如果 ACTION[sm, ai]=移入 s，那么将状态 s 压入栈顶，格局变为(s0s1…sms, ai+1ai+2…an$)；
如果 ACTION[sm, ai]=按照 A→β归约，那么将 r（r 是β的长度）个状态从栈顶弹出，并将状态 s（s=GOTO[sm-r, A]）压入栈顶，格局变为(s0s1…sm-rs, aiai+1…an$)。注意，在执行归约动作时，当前的输入符号不会改变；
如果 ACTION[sm, ai]=接受，那么语法分析过程完成；
如果 ACTION[sm, ai]=报错，那么语法分析器发现了一个语法错误，并调用一个错误恢复例程。
综上所述，一个 LR 语法分析器和 LL 语法分析器一样，也是表驱动的。两个 LR 语法分析器的唯一不同之处在于它们的语法分析表不同。

规范 LR 方法

规范 LR 方法和 SLR 方法类似，它也会构造一个自动机，并从这个自动机得到语法分析表。但是，它们用于构造自动机的项集族不同，SLR 方法使用规范 LR(0)项集族构造一个自动机，而规范 LR 方法使用规范 LR(1)项集族构造一个自动机。
规范 LR(1)项
形如[A→α·β, a]的项是一个规范 LR(1)项，其中，第一个分量是一个规范 LR(0)项，第二个分量是一个终结符号或$，它表示这个规范 LR(1)项的向前看符号。在形如[A→α·β, a]且β≠ε的项中，向前看符号没有任何作用，但是在形如[A→α·, a]的项中，只有在下一个输入符号为 a 时才会按照 A→α进行规约。

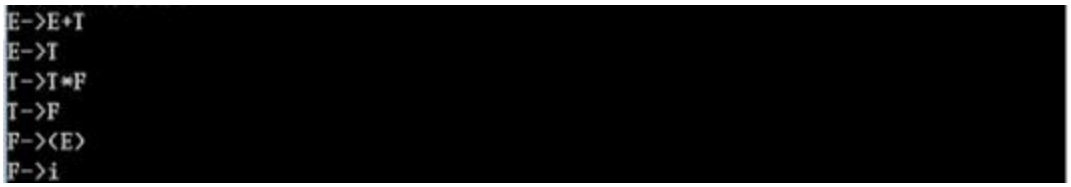
规范 LR(1)自动机
构造 LR(1)自动机也用到了 CLOSURE 和 GOTO 函数，只不过它们的规则有所改变。

计算一个规范 LR(1)项集 I 的 CLOSURE 集合，按照如下规则进行：

将 I 中的所有项加入 CLOSURE 集合中；
对 I 中每个形如[A→α·Bβ, a]的项，首先找到产生式头为 B 的每个产生式 B→γ，然后找到 FIRST(βa)中的每个终结符号 b，最后将所有[B→γ, b]加入 CLOSURE 集合中；
重复上面的步骤，直到没有新的项可以加入 CLOSURE 集合中为止。
计算一个规范 LR(1)项集 I 的 GOTO 集合，按照如下规则进行：

对 I 中每个形如[A→α·Xβ, a]的项，将[A→αX·β, a]加入 GOTO 集合中；
把 GOTO 集合作为 CLOSURE 函数的参数，计算 GOTO 集合的闭包。

你定义的文法产生式如下：



算符优先分析表如下：

语法分析过程如下：

附录：源程序

```
#include<iostream.h>
#include<string.h>
#include<stdio.h
typedef struct
{
char R; char r; int flag;
}array;
typedef struct
{
char E; char e;
}charLode
typedef struct
{
charLode *base; int top;
}charstack;

char str[80][80],arr[80][80],brr[80][80]; array F[20];
int m,kk,p,ppp,FF=1; char r[10];
int crr[20][20],FLAG=0;
char ccrr1[1][20],ccrr2[20][1];
void Initstack(charstack &s)//定义栈
{
s.base=new charLode[20]; s.top=-1;
}

void push(charstack &s,charLode w)  //入栈
{
s.top++; s.base[s.top].E=w.E;
s.base[s.top].e=w.e;
}
void pop(charstack &s,charLode &w)//出栈
{
w.E=s.base[s.top].E;
w.e=s.base[s.top].e; s.top--;
}

int IsEmpty(charstack s)     //判断是否到栈顶
{
if(s.top==-1)
return 1;
```

```
else

}


return 0;


int IsLetter(char ch)    //判断是不是大写字母（非终结符）
{
if(ch>='A'&&ch<='Z') return 1;

else

}

return 0;

//judge1  判断是否是算符文法
int judge1(int n)
{
int j=3,flag=0; for(int i=0;i<=n;i++)
while(str[i][j]!='\0')
{
char a=str[i][j]; char b=str[i][j+1];
if(IsLetter(a)&&IsLetter(b))//两个非终结符相连，不是算符文法
flag=1; break;

j++;


if(flag==1)   //根据  flag  设定返回值
return 0;
else
return 1;

}

//judge2  是判断文法  G  是否为算符优先文法
void judge2(int n)
{
for(int       i=0;i<=n;i++) if(str[i][3]=='~'||FLAG==1)//'~'代表空
{
cout<<"文法  G  不是算符优先文法!"<<endl; FF=0;
```

```
break;
}

if(i>n)
cout<<"文法 G 是算符优先文法!"<<endl;
}

//search1 是查看存放终结符的数组 r 中是否含有重复的终结符
int search1(char r[],int kk,char a)
{
for(int i=0;i<kk;i++) if(r[i]==a)
break; if(i==kk)

//createF 函数是用 F 数组存放每个终结符与非终结符和组合
void createF(int n)
{
int k=0,i=1;char g;
char t[10];    //t 数组用来存放非终结符
t[0]=str[0][0];
while(i<=n)
{
if(t[k]!=str[i][0])
{
k++;
t[k]=str[i][0]; g=t[k];
i++;
}
else i++;

}
kk=0; char c;
for(i=0;i<=n;i++)
{
int j=3; while(str[i][j]!='\0')
{
c=str[i][j]; if(IsLetter(c)==0)
if(!search1(r,kk,c)) r[kk]=c;
kk++;   //r 数组用来存放终结符

m=0;
for(i=0;i<k;i++)
for(int j=0;j<kk-1;j++)
{
F[m].R=t[i];
```

```
F[m].r=r[j]; F[m].flag=0; m++;

//search 函数是将在 F 数组中寻找到的终结符与非终结符对的标志位值为 1 void search(charLode
w)
{
for(int i=0;i<m;i++) if(F[i].R==w.E&&F[i].r==w.e)
{
F[i].flag=1;break;
}
}

void FirstVT(int n)//求 FirstVT
{
charstack sta; charLode w; int i=0; Initstack(sta);

while(i<=n)
{
int k=3; w.E=str[i][0];
char a=str[i][k]; char b=str[i][k+1];
if(!IsLetter(a))     //产生式的后选式的第一个字符就是终结符的情况
{
w.e=a; push(sta,w); search(w); i++;
}
else if(IsLetter(a)&&b!='\0'&&!IsLetter(b))
//产生式的后选式的第一个字符是非终结符的情况
{
w.e=b; push(sta,w); search(w); i++;
}
else i++;
}
charLode ww; while(!IsEmpty(sta))
{
pop(sta,ww); for(i=0;i<=n;i++)
{
w.E=str[i][0]; if(str[i][3]==ww.E&&str[i][4]=='\0')
{
w.e=ww.e;
push(sta,w); search(w); break;
p=0;int k=1;i=1; while(i<m)
{
if(F[i-1].flag==1)
{
arr[p][0]=F[i-1].R;
```

```
arr[p][k]=F[i-1].r;
}
while(F[i].flag==0&&i<m) i++;
if(F[i].flag==1)
{
if(F[i].R==arr[p][0]) k++;
else {arr[p][k+1]='\0';p++;k=1;} i++;
void LastVT(int n)//求 LastVT
{
charstack sta; charLode w;
for(int i=0;i<m;i++) F[i].flag=0;
i=0;
Initstack(sta); while(i<=n)
{
int k=strlen(str[i]); w.E=str[i][0];
char a=str[i][k-1];
char b=str[i][k-2];
if(!IsLetter(a))
{
w.e=a; push(sta,w); search(w); i++;
}
else if(IsLetter(a)&&!IsLetter(b))
{
w.e=b; push(sta,w); search(w); i++;
}
else i++;
}
charLode ee;

while(!IsEmpty(sta))
{
pop(sta,ee); for(i=0;i<=n;i++)
{
w.E=str[i][0]; if(str[i][3]==ee.E&&str[i][4]=='\0')
{
w.e=ee.e;
push(sta,w); search(w);
int k=1;i=1; ppp=0; while(i<m)
{
if(F[i-1].flag==1)
{
brr[ppp][0]=F[i-1].R;
brr[ppp][k]=F[i-1].r;
}
```

```
while(F[i].flag==0&&i<m) i++;
if(F[i].flag==1)
{
if(F[i].R==arr[ppp][0]) k++;
else {brr[ppp][k+1]='\0';ppp++;k=1;} i++;

void createYXB(int n)//构造优先表
{
int i,j; for(j=1;j<=kk;j++)
ccrr1[0][j]=r[j-1]; for( i=1;i<=kk;i++)
ccrr2[i][0]=r[i-1]; for(i=1;i<=kk;i++)
for(j=1;j<=kk;j++) crr[i][j]=0;

int I=0,J=3;
while(I<=n)
{
if(str[I][J+1]=='\0')     //扫描右部

while(str[I][J+1]!='\0')

{
char aa=str[I][J]; char bb=str[I][J+1];
if(!IsLetter(aa)&&!IsLetter(bb))//优先及等于的情况，用 1 值表示等于
{
for(i=1;i<=kk;i++)
{
if(ccrr2[i][0]==aa) break;
}
for(j=1;j<=kk;j++)
{
if(ccrr1[0][j]==bb) break;
}
if(crr[i][j]==0)
crr[i][j]=1; else {
FLAG=1; I=n+1;
} J++;
}

if(!IsLetter(aa)&&IsLetter(bb)&&str[I][J+2]!='\0'&&!IsLetter(str[I][J+2]))
//优先及等于的情况
{
for(i=1;i<=kk;i++)
{
if(ccrr2[i][0]==aa) break;
```

```
}
for(int j=1;j<=kk;j++)
{

if(ccrr1[0][j]==str[I][J+2]) break;
}
if(crr[i][j]==0)
crr[i][j]=1;

FLAG=1; I=n+1;

if(!IsLetter(aa)&&IsLetter(bb))
//优先及小于的情况，用 2 值表示小于
{
for(i=1;i<=kk;i++)
{
if(aa==ccrr2[i][0]) break;
}
for(j=0;j<=p;j++)
{
if(bb==arr[j][0]) break;
}
for(int mm=1;arr[j][mm]!='\0';mm++)
{
for(int pp=1;pp<=kk;pp++)
{
if(ccrr1[0][pp]==arr[j][mm]) break;
}
if(crr[i][pp]==0)
crr[i][pp]=2; else {
FLAG=1;I=n+1;
}
} J++;
}
if(IsLetter(aa)&&!IsLetter(bb))
//优先及大于的情况，用 3 值表示大于
{
for(i=1;i<=kk;i++)
{
if(ccrr1[0][i]==bb)

break;
}
for(j=0;j<=ppp;j++)
```

```
{
if(aa==brr[j][0]) break;
}
for(int mm=1;brr[j][mm]!='\0';mm++)
{
for(int pp=1;pp<=kk;pp++)
{
if(ccrr2[pp][0]==brr[j][mm]) break;
}

if(crr[pp][i]==0)
crr[pp][i]=3;
else {FLAG=1;I=n+1;}

//judge3  是用来返回在归约过程中两个非终结符相比较的值
int judge3(char s,char a)
{
int i=1,j=1; while(ccrr2[i][0]!=s)
i++;
while(ccrr1[0][j]!=a) j++;
if(crr[i][j]==3)
return 3;

else


if(crr[i][j]==2)
return 2;

else


if(crr[i][j]==1)
return 1;

else
return 0;

}

void print(char s[],char STR[][20],int q,int u,int ii,int  k)//打印归约的过程
{
cout<<u<<"  "; for(int i=0;i<=k;i++)
cout<<s[i]; cout<<"     ";
```

```
for(i=q;i<=ii;i++) cout<<STR[0][i];
cout<<" ";
}

void process(char STR[][20],int ii)//对输入的字符串进行归约的过程
{
// cout<<"步骤"<<"  "<<"符号栈"<<"  "<<"输入串"<<"  "<<"动作"<<endl; int k=0,q=0,u=0,b,i,j;
char s[40],a;
s[k]='#';
print(s,STR,q,u,ii,k); cout<<"预备"<<endl; k++;
u++; s[k]=STR[0][q]; q++;
print(s,STR,q,u,ii,k); cout<<"移进"<<endl; while(q<=ii)
{
a=STR[0][q];
if(!IsLetter(s[k])) j=k; else j=k-1; b=judge3(s[j],a);
if(b==3)//大于的情况进行归约
{
while(IsLetter(s[j-1])) j--;
for(i=j;i<=k;i++) s[i]='\0';
k=j; s[k]='N'; u++;
print(s,STR,q,u,ii,k); cout<<"归约"<<endl;
}

else if(b==2||b==1)//小于或等于的情况移进

k++;
s[k]=a; u++; q++;
print(s,STR,q,u,ii,k); if(s[0]=='#'&&s[1]=='N'&&s[2]=='#')
cout<<"接受"<<endl;
else cout<<"移进"<<endl;

cout<<"出错"<<endl; break;

if(s[0]=='#'&&s[1]=='N'&&s[2]=='#')
cout<<"归约成功"<<endl; else cout<<"归约失败"<<endl;
}

void main()
{
int n,i,j;
cout<<"输入定义的文法 G 的产生式的个数 n:"; cin>>n;
cout<<"输入文法产生式："<<endl;
for(i=0;i<n;i++)
{
```

```
gets(str[i]); j=strlen(str[i]); str[i][j]='\0';
}
str[i][0]='Q'; //最末行添加扩展
str[i][1]='-';
str[i][2]='>';
str[i][3]='#';
str[i][4]=str[0][0];
str[i][5]='#';
cout<<"你定义的产生式如下:"<<endl;  str[i][6]='\0';
for(i=0;i<=n;i++) cout<<str[i]<<endl;
if(judge1(n)==0)  //判断文法 G 是否为算符文法

cout<<"文法 G 不是算符文法!"<<endl;

if(judge1(n)==1)
{
cout<<"文法 G 是算符文法!"<<endl;

}
createF(n); FirstVT(n); LastVT(n); createYXB(n);

for(i=0;i<=p;i++)//打印 FirstVT
{
cout<<"FirstVT("<<arr[i][0]<<")={";
for(int l=1;arr[i][l+1]!='\0';l++)
cout<<arr[i][l]<<","; cout<<arr[i][l]<<"}"<<endl;
}
cout<<"FirstVT(Q)={#}"<<endl; for(i=0;i<=ppp;i++)//打印 LastVT
{
cout<<"LastVT("<<arr[i][0]<<")={";
for(int l=1;brr[i][l+1]!='\0';l++)
cout<<brr[i][l]<<","; cout<<brr[i][l]<<"}"<<endl;
}
cout<<"LastVT(Q)={#}"<<endl; cout<<"算符优先表如下："<<endl; for(i=1;i<kk;i++)//打印优先关系表
{
cout<<" "; cout<<ccrr1[0][i];
}
cout<<endl; for(i=1;i<kk;i++)
{
cout<<ccrr2[i][0]<<" "; for(j=1;j<kk;j++)
{
if(crr[i][j]==0)
cout<<" "; else if(crr[i][j]==1)
cout<<"="; else if(crr[i][j]==2)
```

```
cout<<"<"; else if(crr[i][j]==3)
cout<<">"; cout<<"     ";
}
cout<<endl;
}

judge2(n);//判断文法 G 是否为算符优先文法
if(FF==1)
{
char STR[1][20];
cout<<"请输入要规约的字符串:"<<endl; gets(STR[0]);
int ii=strlen(STR[0]); STR[0][ii]='#';
cout<<"语法分析过程如下："<<endl;
process(STR,ii);
```