



东南大学国家示范性软件学院

College of Software Engineering  
Southeast University

# 编译原理

# 实验报告

实验名称： Syntax Parser

学生姓名： 孟越

学生学号： 71Y15102

东南大学 软件学院 制



## 一、实验目的

1. 理解语法原理及相关理论
2. 巩固课上知识即语法分析器工作的整个过程，加深对语法分析的理解
3. 编写一个简单的人工 yacc，提升编程代码水平

## 二、实验说明

**1.编程环境：** windows7 & vs2010 & C++

- 2.设计说明：**
- 1.设计一套文法可以处理上次实验的程序代码
  - 2.对这套文法进行处理，消除左递归，编序号
  - 3.画出 LL（1）表
  - 4.最后使用代码实现 top-down 的分析。

**3.数据结构算法说明：**主要是要用到双栈和表。对于上一个实验中得到的 token 按序存在一个栈中，另一个栈中放 S 进行 top-down 的分析。在具体解析时，使用了 LL(1)表，帮助我们在两个栈读到不同数据时所要进行的操作。在数据输出时使用了队列记录了语法解析的顺序，



### 三、实验过程

#### 1. 设计一段 CFG

$S \rightarrow D;E$   
 $E \rightarrow \text{for}(T)\{E\}$   
 $E \rightarrow X;$   
 $T \rightarrow >;X;$   
 $D \rightarrow \text{int } X$   
 $X \rightarrow \text{variable} < \text{integer}$   
 $X \rightarrow \text{variable} + \text{integer}$   
 $X \rightarrow \text{variable} = \text{integer}$   
 $X \rightarrow \text{variable} = \text{variable} + \text{integer}$

#### 2. 对这个 CFG 消除左递归并标上序号

1.  $S \rightarrow D;E$
2.  $E \rightarrow \text{for}(T)\{E\}$
3.  $E \rightarrow X$
4.  $T \rightarrow >;X;$
5.  $D \rightarrow \text{int } X$
6.  $X \rightarrow \text{variable } X'$
7.  $X' \rightarrow =X''$
8.  $X' \rightarrow <X''$
9.  $X' \rightarrow +X''$
10.  $X'' \rightarrow \text{integer}$
11.  $X'' \rightarrow X$

#### 3. 算出非终结符的 first 和 follow 集合，画出 LL (1) 表

FIRST 和 FOLLOW 集合

	FIRST	FOLLOW
$X''$	{integer,variable}	{ ; }
$X'$	{=,<,+}	{ ; }
$X$	{variable}	{ ; }
$D$	{int}	{ ; }
$T$	{ ; }	{ > }
$E$	{for,variable}	{ }, \$ }
$S$	{int}	{ \$ }



LL(1)表

	=	<	;	+	int	for	variable	integer
X''							11	10
X'	7	8		9				
X							6	
D					5			
T			4					
E						2	3	
S					1			

4. 根据以上双栈和 LL(1)表编写代码

Main 函数:

```
void main()
{
    ifstream inputFile("SyntaxInput.txt");
    ofstream outputFile("ResultOutput.txt");
    string stemp=" ";
    string s=" ";
    if(!inputFile.is_open()){
        cout<<"未!ä成 ``| 功|打ä `` ° 开a文?件t"<<endl;
    }
    while(getline(inputFile, stemp)) {
        s+=stemp;
    }
    s+=" ";
    stack <string>s1;
    stack <string>s2;
    s1.push("$");
    s2.push("$");
    s2.push("S");
    int i=0;
    vector <string>v;
    vector <vector<string>>cfgV;
    vector <string>cfgTemp;
    cfgTemp.push_back("D");
    cfgTemp.push_back(";");
    cfgTemp.push_back("E");
    cfgV.push_back(cfgTemp);
    cfgTemp.clear();
    cfgTemp.push_back("for");
    cfgTemp.push_back("(");
    cfgTemp.push_back("T");
```



```
cfgTemp.push_back("");  
cfgTemp.push_back("{");  
cfgTemp.push_back("E");  
cfgTemp.push_back("}");  
cfgV.push_back(cfgTemp);  
cfgTemp.clear();  
cfgTemp.push_back("X");  
cfgTemp.push_back(";");  
cfgV.push_back(cfgTemp);  
cfgTemp.clear();  
cfgTemp.push_back(";");  
cfgTemp.push_back("X");  
cfgTemp.push_back(";");  
cfgV.push_back(cfgTemp);  
cfgTemp.clear();  
cfgTemp.push_back("int");  
cfgTemp.push_back("X");  
cfgV.push_back(cfgTemp);  
cfgTemp.clear();  
cfgTemp.push_back("variable");  
cfgTemp.push_back("X");  
cfgV.push_back(cfgTemp);  
cfgTemp.clear();  
cfgTemp.push_back("=");  
cfgTemp.push_back("X\\");  
cfgV.push_back(cfgTemp);  
cfgTemp.clear();  
cfgTemp.push_back("<");  
cfgTemp.push_back("X\\");  
cfgV.push_back(cfgTemp);  
cfgTemp.clear();  
cfgTemp.push_back("+");  
cfgTemp.push_back("X\\");  
cfgV.push_back(cfgTemp);  
cfgTemp.clear();  
cfgTemp.push_back("integer");  
cfgV.push_back(cfgTemp);  
cfgTemp.clear();  
cfgTemp.push_back("X");  
cfgV.push_back(cfgTemp);  
cfgTemp.clear();  
string temp;  
while(i<s.length()){  
    if(i==0)
```



```
{
    if(s[i]=='('){
        temp="";
        while(s[++i]!='(',')'){
            temp+=s[i];
        }
        v.push_back(temp);
    }else{
        i++;
    }
} else{
    if(s[i]=='(' && s[i-1]!='(' && s[i-1]!='(',')'){
        temp="";
        while(s[++i]!='(',')'){
            temp+=s[i];
        }
        v.push_back(temp);
    }
    else{
        i++;
    }
}
}
i=v.size()-1;
for(;i>=0;i--){
    s1.push(v[i]);
}
//按ä;ä照?表Ä ``a扫|i § 描 `` 栈?
outputFile<<LL1(s1,s2,cfgV);

system("pause");
inputFile.close();
outputFile.close();
}
```

辅助函数:

```
string output(queue <int>order){
    string result=" ";
    while(!order.empty())
    {
        int temp=order.front();
        if(temp==1){
            result+="S->D;E\n";
        }else if(temp==2){
```



```
        result+="E->for(T) {E} \n";
    }else if(temp==3) {
        result+="E->X\n";
    }else if(temp==4) {
        result+="T->;X;\n";
    }else if(temp==5) {
        result+="D->int X\n";
    }else if(temp==6) {
        result+="X->variable X' \n";
    }else if(temp==7) {
        result+="X' ->X' \n";
    }else if(temp==8) {
        result+="X' -><X' \n";
    }else if(temp==9) {
        result+="X' ->+X' \n";
    }else if(temp==10) {
        result+="X' ->integer\n";
    }else if(temp==11) {
        result+="X' ->X";
    }
    order.pop();
}
return result;
}
```

语法分析函数:

```
string LL1(stack <string>s1,stack <string>s2,vector <vector<string>>cfgV)
{
    queue <int>order;
    vector<string> temp;
    while(s1.top()!="$") {
        if(s1.top()==s2.top()) {
            s1.pop();
            s2.pop();
            continue;
        }
        else if(s1.top()=="variable") {
            if(s2.top()=="X'") {
                s2.pop();
                temp=cfgV[10];
                cout<<11<<endl;
                order.push(11);
                for(int i=temp.size()-1;i>=0;i--){
```



```
        s2.push(temp[i]);
    }
    temp.clear();
    continue;
} else if(s2.top()=="X") {
    s2.pop();
    temp=cfgV[5];
    cout<<6<<endl;
    order.push(6);
    for(int i=temp.size()-1;i>=0;i--){
        s2.push(temp[i]);
    }
    temp.clear();
    continue;
} else if(s2.top()=="E") {
    s2.pop();
    temp=cfgV[2];
    cout<<3<<endl;
    order.push(3);
    for(int i=temp.size()-1;i>=0;i--){
        s2.push(temp[i]);
    }
    temp.clear();
    continue;
}
}

else if(s1.top()=="integer") {
    if(s2.top()=="X\\") {
        s2.pop();
        temp=cfgV[9];
        cout<<10<<endl;
        order.push(10);
        for(int i=temp.size()-1;i>=0;i--){
            s2.push(temp[i]);
        }
        temp.clear();
        continue;
    }
}

else if(s1.top()=="for") {
    if(s2.top()=="E") {
        s2.pop();
        temp=cfgV[1];
        cout<<2<<endl;
```





```
        order.push(2);
        for(int i=temp.size()-1;i>=0;i--){
            s2.push(temp[i]);
        }
        temp.clear();
        continue;
    }
}

else if(s1.top()=="int"){
    if(s2.top()=="D"){
        s2.pop();
        temp=cfgV[4];
        cout<<5<<endl;
        order.push(5);
        for(int i=temp.size()-1;i>=0;i--){
            s2.push(temp[i]);
        }
        temp.clear();
        continue;
    }else if(s2.top()=="S"){
        s2.pop();
        temp=cfgV[0];
        cout<<1<<endl;
        order.push(1);
        for(int i=temp.size()-1;i>=0;i--){
            s2.push(temp[i]);
        }
        temp.clear();
        continue;
    }
}

else if(s1.top()=="+"){
    if(s2.top()=="X'"){
        s2.pop();
        temp=cfgV[8];
        cout<<9<<endl;
        order.push(9);
        for(int i=temp.size()-1;i>=0;i--){
            s2.push(temp[i]);
        }
        temp.clear();
        continue;
    }
}
```



```
else if(s1.top()==";"){
    if(s2.top()=="T"){
        s2.pop();
        temp=cfgV[3];
        cout<<4<<endl;
        order.push(4);
        for(int i=temp.size()-1;i>=0;i--){
            s2.push(temp[i]);
        }
        temp.clear();
        continue;
    }
}
else if(s1.top()=="<"){
    if(s2.top()=="X' "){
        s2.pop();
        temp=cfgV[7];
        cout<<8<<endl;
        order.push(8);
        for(int i=temp.size()-1;i>=0;i--){
            s2.push(temp[i]);
        }
        temp.clear();
        continue;
    }
}
else if(s1.top()=="="){
    if(s2.top()=="X' "){
        s2.pop();
        temp=cfgV[6];
        cout<<7<<endl;
        order.push(7);
        for(int i=temp.size()-1;i>=0;i--){
            s2.push(temp[i]);
        }
        temp.clear();
        continue;
    }
}
else{
    cout<<"don't match this grammar"<<endl;
    return "don't match this grammar";
    break;
}
```

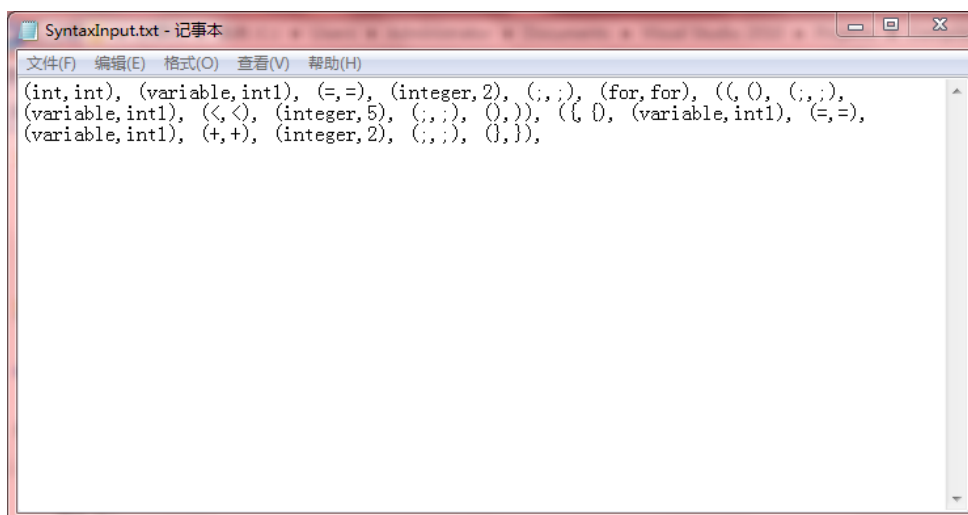


```
    }  
    if(s2.top()=="$"){  
        cout<<"match"<<endl;  
    }  
    return output(order);  
}
```



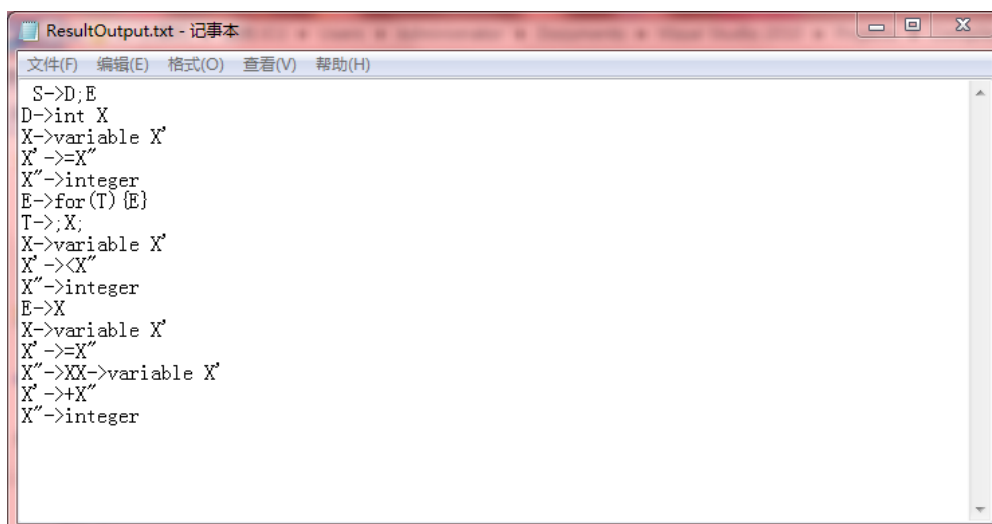
## 四、用户手册

1. **程序规格说明：**输入一段程序代码，扫描识别出各种 token，语法分析后输出解析过程，可以得到语法分析树。
2. **使用说明：**在项目文件夹下放了 SyntaxInput.txt 文件放了上个实验输出的词法单元和 ResultOutput.txt 文件作为最终结果的输出，直接运行代码程序会对 SyntaxInput.txt 文件进行扫描，对其中的 token 逐一扫描和提取存放到栈中，再通过双栈使用 LL(1)表逐一对比，可根据设计的文法输出 top-down 分析的过程从而可以构造出语法分析树。
3. **例子：**  
输入：



```
SyntaxInput.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
(int,int), (variable,int1), (=,=), (integer,2), (,;,), (for,for), ((,(), (,;,),
(variable,int1), (<,<), (integer,5), (,;,), (,;,), ((,(), (variable,int1), (=,=),
(variable,int1), (+,+), (integer,2), (,;,), (,;,),
```

输出：



```
ResultOutput.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
S->D;E
D->int X
X->variable X'
X' ->=X''
X''->integer
E->for(T) {E}
T->;X;
X->variable X'
X' -><X''
X''->integer
E->X
X->variable X'
X' ->=X''
X''->XX->variable X'
X' ->+X''
X''->integer
```



## 五、实验总结

### ■ 本次实验分析：

这次实验做法很简单，因为最重要的构建表的过程是人工实现的，代码实现的只是使用 LL(1)表的过程。代码结构也比较简单，没有使用什么复杂的数据结构，当然，top-down 分析方法中最基础的双栈还是得用到，根据原理，其实就是每次从栈顶取元素，根据人工构造的表进行解析。输入的是 token 的序列，输出是语法序列（语法分析的整个过程）。在 main 函数中实现了对初始值（CFG 和 token 序列）饿入栈操作，ll（1）函数中，便是实现了对 ll（1）表对使用，最后把结果即分析过程存下来即可。

### ■ 实验的未来展望：

1. 期望使用 LR（1）的方法进行语法分析
2. 期望把构建表的过程自动化,可以根据设计的 CFG 自动帮我们构造出相应的 LL(1)和 LR(1)表

注：源程序在同文件夹中