**《编译系统设计实践》**

实验二：语法分析实验

学号： 031502248

姓名： 左文航

年级： 15级

学院： 数计学院

专业： 计算机

实验时间：2016－2017学年第一学期

任课教师：刘秉瀚

目录

[一、实验目的 1](#_Toc503541176)

[二、实验内容 1](#_Toc503541177)

[三、设计思路 1](#_Toc503541178)

[First集的构造 1](#_Toc503541179)

[分析表的构造 2](#_Toc503541180)

[分析的实现 2](#_Toc503541181)

[四、代码设计 2](#_Toc503541182)

[函数设计 2](#_Toc503541183)

[五、实验结果 2](#_Toc503541184)

[测试样例一 2](#_Toc503541185)

[测试样例二 5](#_Toc503541186)

[测试样例三 7](#_Toc503541187)

[六、实验心得 9](#_Toc503541188)

[七、附 10](#_Toc503541189)

[1. void get\_first() 10](#_Toc503541190)

[2. void get\_search(project tmp) 11](#_Toc503541191)

[3. void e\_closure(int T) 12](#_Toc503541192)

[4. void make\_set() 13](#_Toc503541193)

[5. void get\_action() 14](#_Toc503541194)

[6. void cal() 15](#_Toc503541195)

## 一、实验目的

根据给出的文法编制LR（1）分析程序，以便对任意输入的符号串进行分析。本次实验的目的主要是加深对LR（1）分析法的理解。

## 二、实验内容

1、LR（1）分析法的功能   
LR（1）分析法的功能是利用LR（1）分析表，对输入符号串自下而上的分析过程。   
   
2、LR（1）分析表的构造及分析过程。

分析表的构造参考课本167页图4-40，

分析过程参考课本160页图4-36，

程序输出参考课本160页图4-38。

## 三、设计思路

**本次实验完全采用编程的方法实现**

### First集的构造

* 计算FIRST(X)的方法
  + 如果X是终结符号，那么FIRS(X)=X
  + 如果X是非终结符号，且X🡪Y1Y2…Yk是产生式，
    - 如果a在FIRST(Yi)中，且ε在FIRST(Y1), FIRST(Y2),…, FIRST(Yi-1)中，那么a也在FIRST(X)中。
    - 如果ε在FIRST(Y1), FIRST(Y2),…, FIRST(Yk)中，那么ε在FIRST(X)中。
  + 如果X是非终结符号，且X🡪ε是一个产生式，那么ε在FIRST(X)中
* 计算FIRST(X1X2…Xn)的方法
  + 向集合中加入FIRST(X1)中所有非ε的符号；
  + 如果ε在FIRST(X1)中，再加入FIRST(X2)中的所有非ε的符号；
  + …
  + 如果ε在所有的FIRST(X1)中，将ε加入FIRST(X1X2…Xn)中。

### 分析表的构造

将给定的pascal读入到程序中去,然后利用课本167页图4-40，先构造出first集,然后构造clouse,给出分析表的I0项,然后通过编程的方法实现。

* 算法：
  + 首先将I中的各个项加入到CLOSURE(I)中；
  + 如果A🡪α.Bβ在CLOSURE(I)中，那么对B的任意产生式B🡪γ，将B🡪.γ加到CLOSURE(I)中；
  + 不断重复第二步，直到收敛。
* 第二步的分析含义：
  + 项A🡪α.Bβ表示期望在接下来的输入中归约到B。
  + 显然，要归约到B，首先要扫描归约到B的某个产生式的右部；
  + 因此对每个产生式B🡪γ，加入B🡪.γ；
    - 表示它期望能够扫描归约到γ。
* 注意和 ε-CLOSURE的对应关系

### 分析的实现

两个部分：动作ACTION，转换GOTOACTION有两个参数：状态i、终结符号a移入j：j是一个状态。把j压入栈；归约Aβ：把栈顶的β归约为A。接受：接受输入、完成分析。报错：在输入中发现语法错误。状态集上的GOTO函数

如果GOTO[Ii,A]=Ij，那么GOTO[i,A] = j。

## 四、代码设计

### 函数设计

void get\_grammar()

读入文法

void get\_first()

获取非终结符的first集

bool is\_in(project tmp,int T)

判断新增加的e\_closure 是否有重复

void get\_search(project tmp)

获取向前看符号

void e\_closure(int T)

求项目集

int is\_in\_project(int k)

判断新得到的项目集是否重复,如果重复,输出存在的项目集,否则输出0

void make\_set()

构建项目集

void get\_action()

构造分析表

Void Print\_first()

输出非总结符的first表

void Print\_action\_table()

输出分析表

void Print\_set()

输出项目集

void Print\_state()

输出当前分析表中的状态

void Print\_sign()

输出当前分析表中的符号

void Print\_ACTION(int q)

输出分析表中的当前动作

void Print\_anslysis()

输出分析表

void cal()

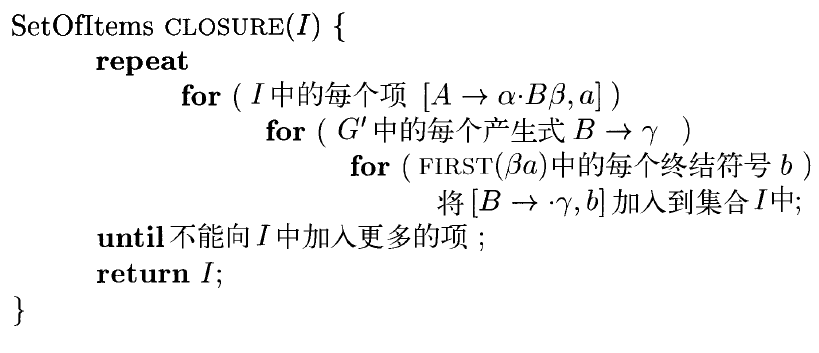
执行分析过程

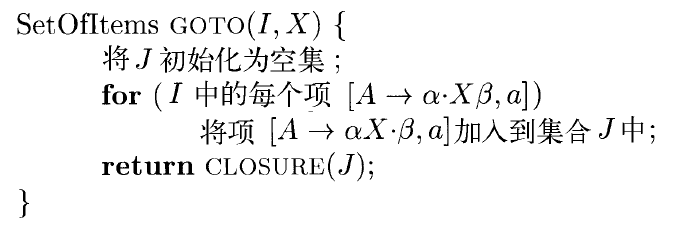
void open\_file()

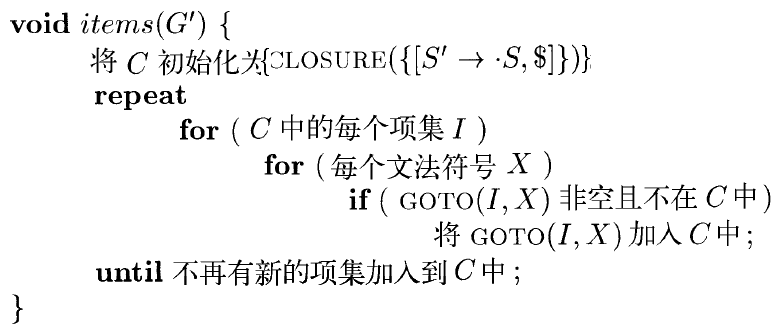
打开本次实验需要用到的文件

void close\_file()

关闭本次实验需要用到的文件

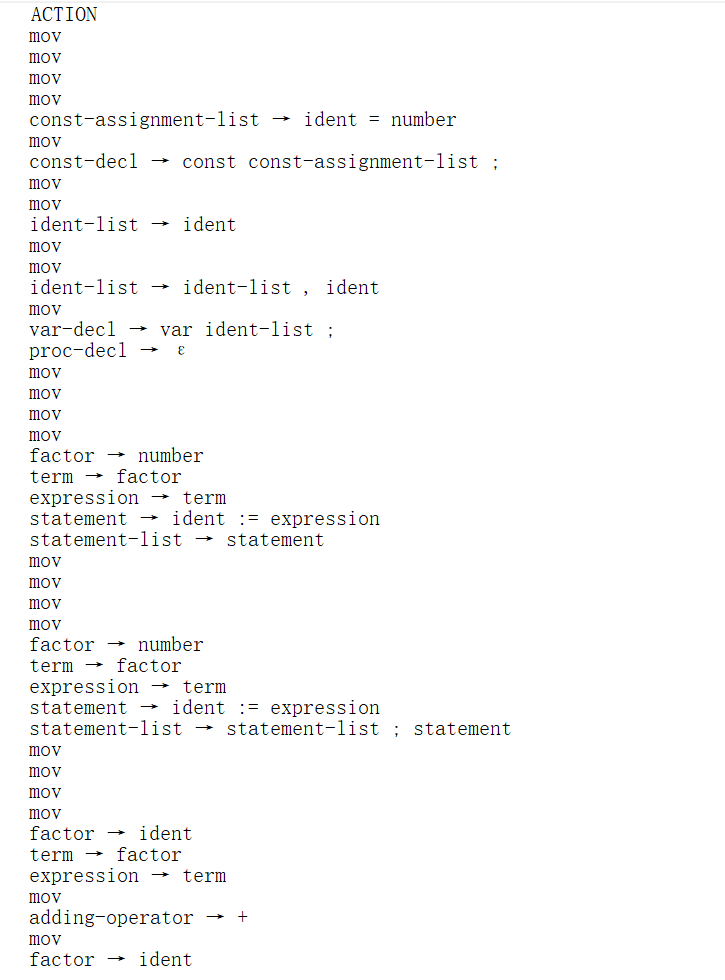
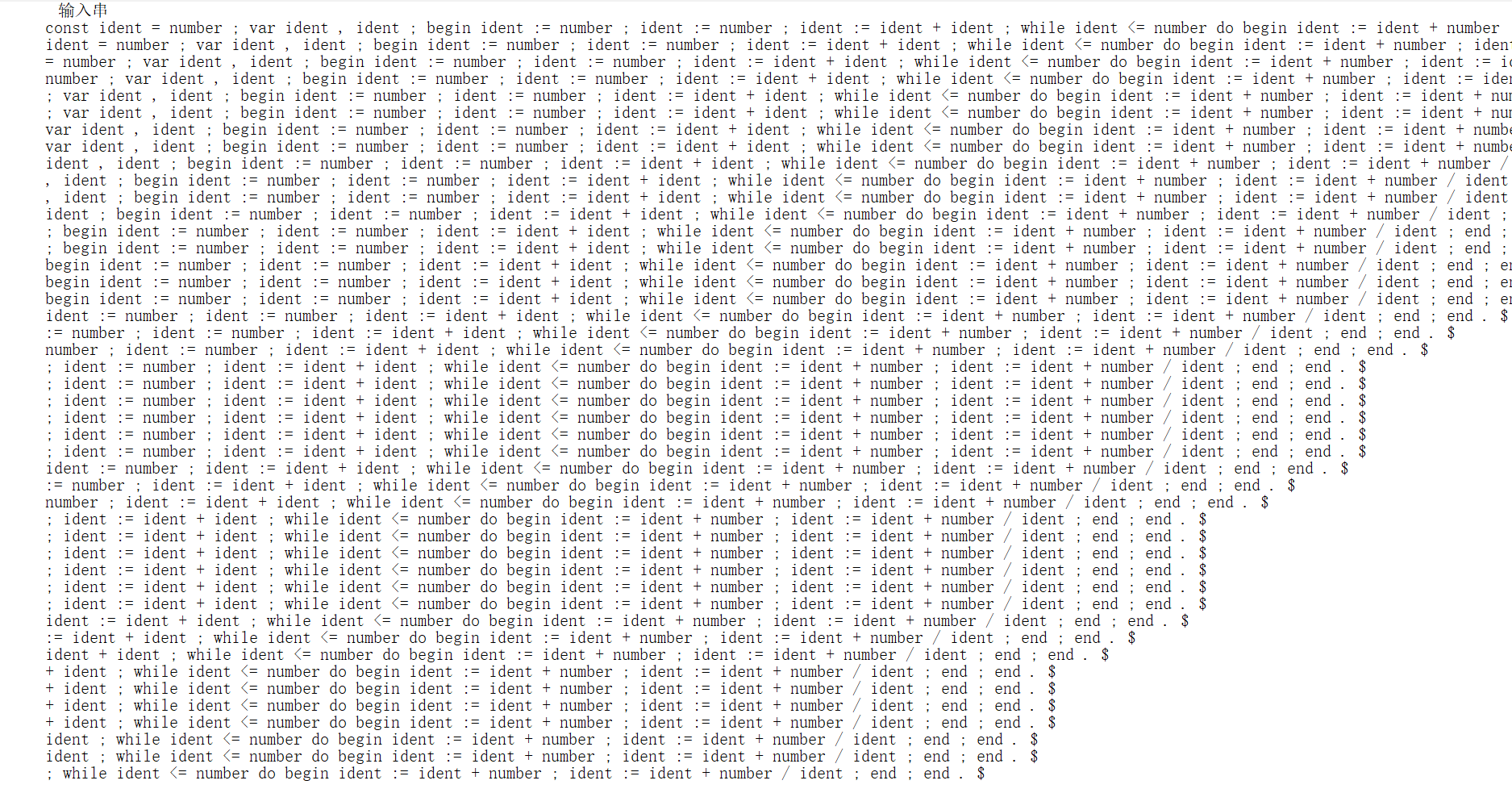
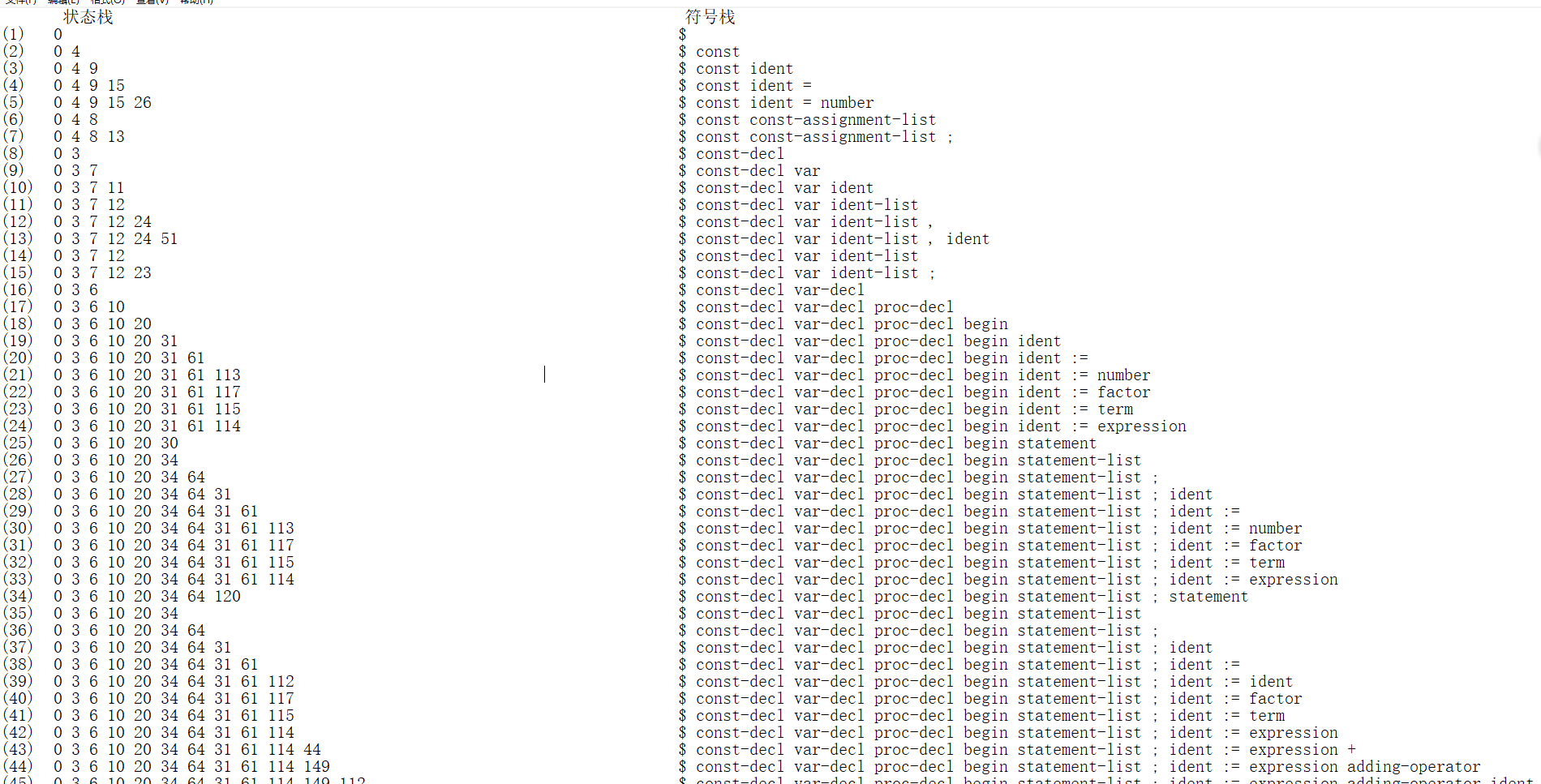
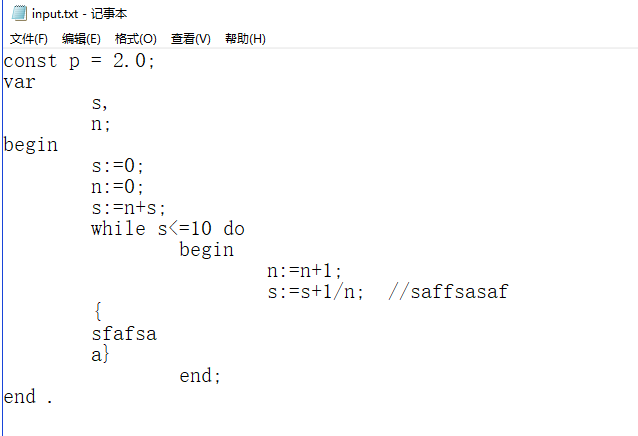




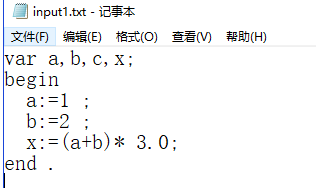


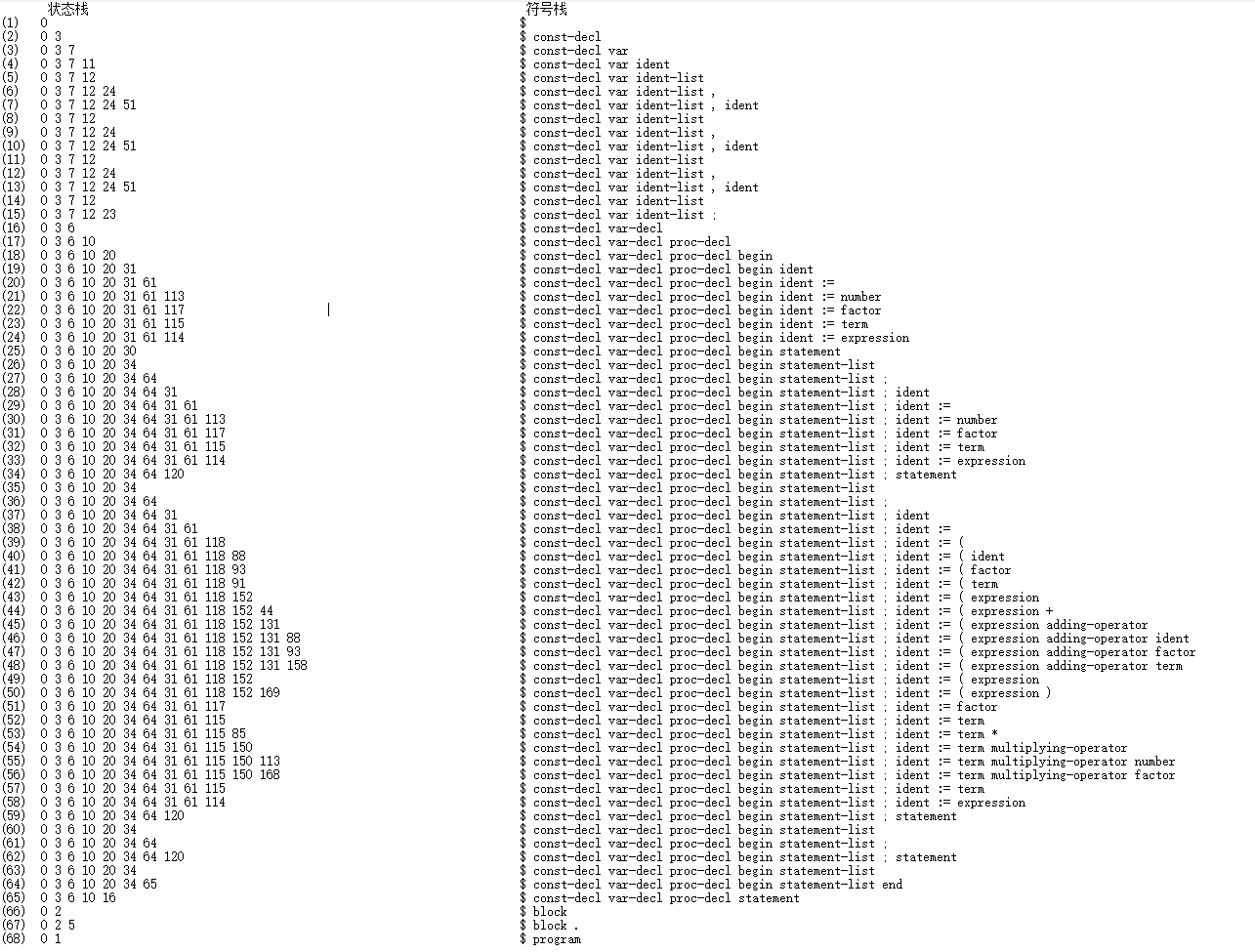
## 五、实验结果

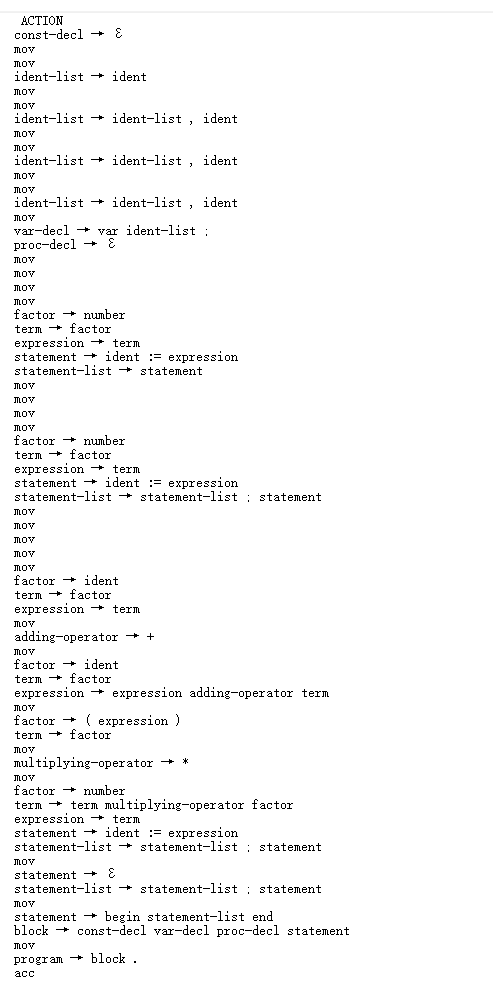
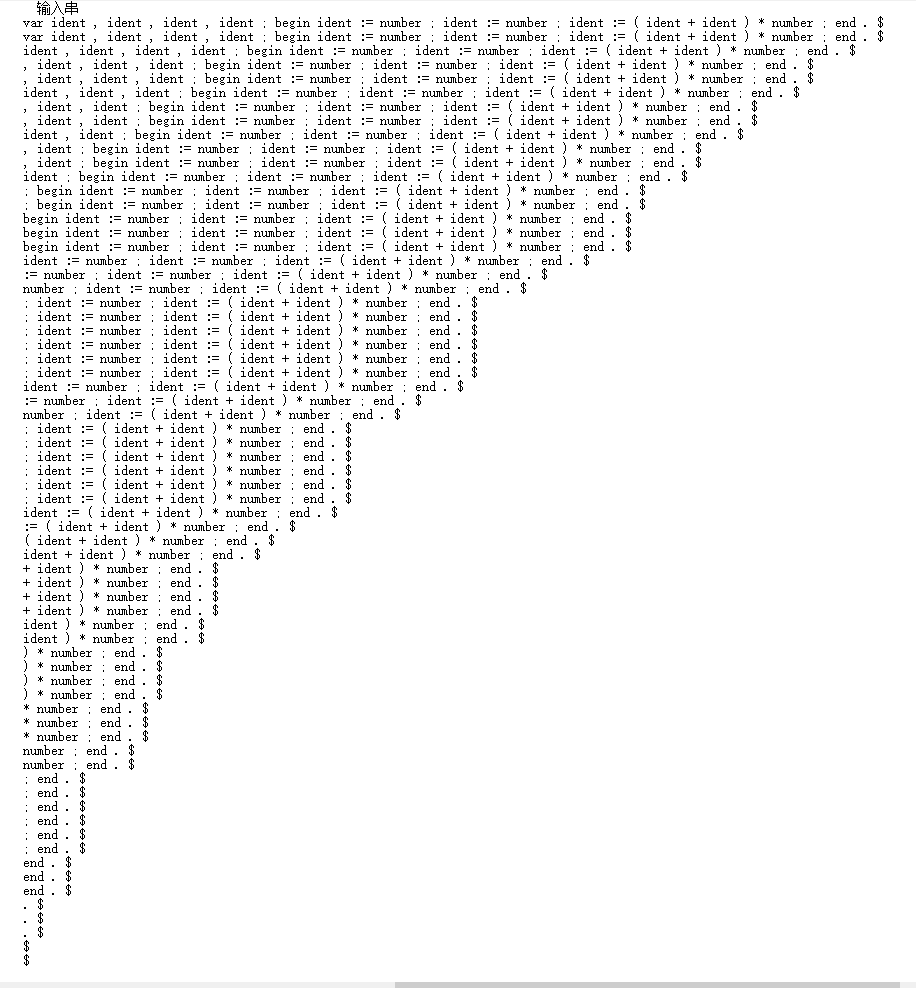
### 测试样例一



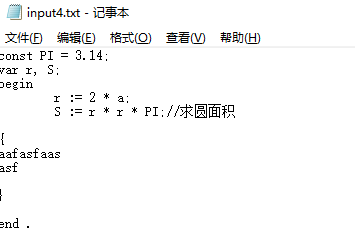
### 测试样例二

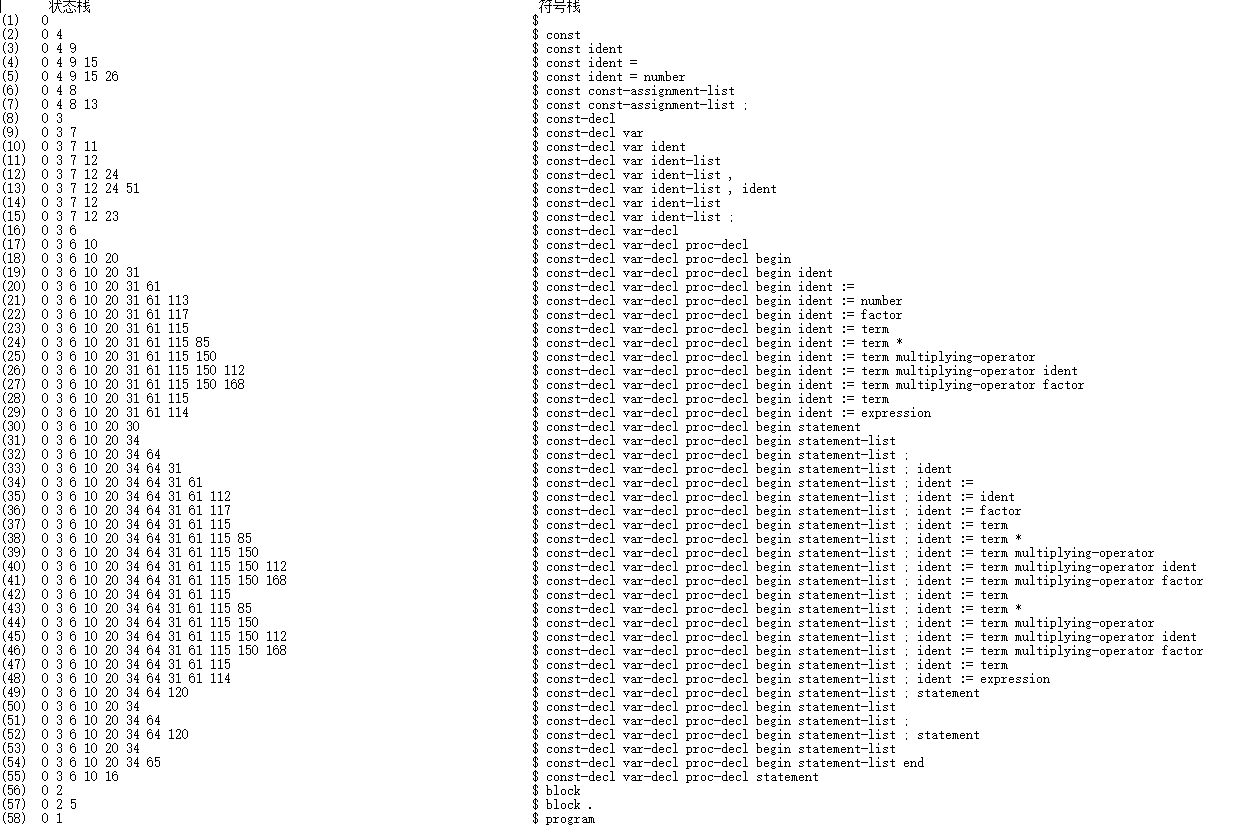


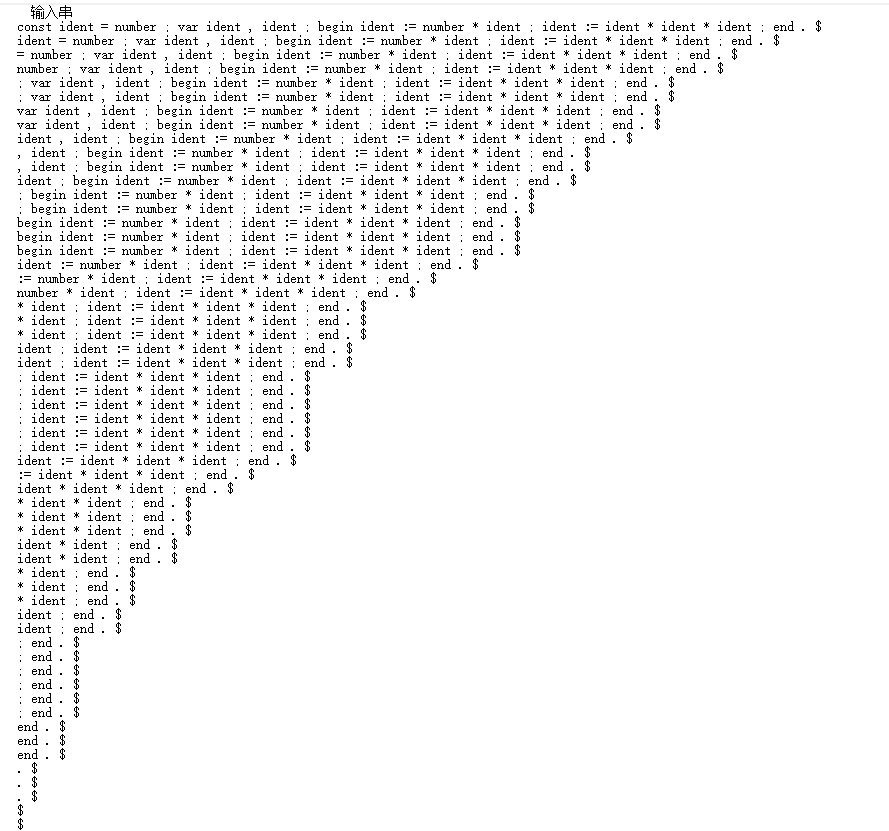


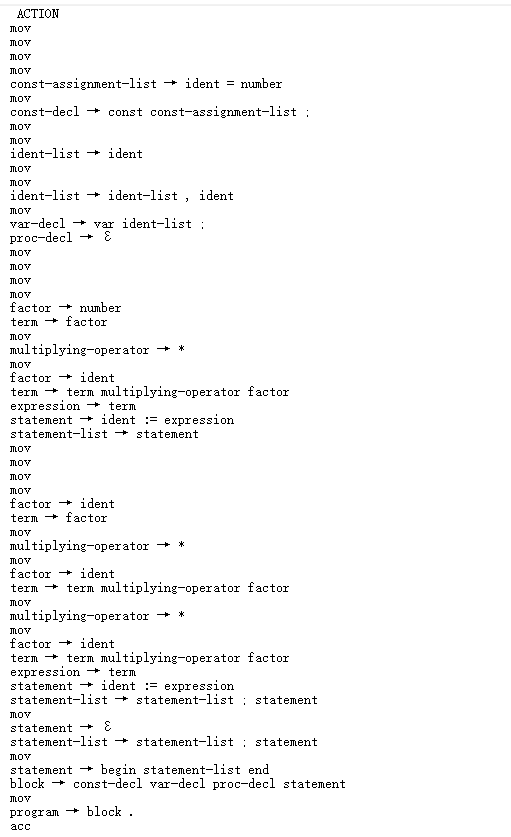


### 测试样例三

****







## 六、实验心得

通过这次实验,让我充分掌握了LR(1)的实现过程,也了解了LR(1)与SLR(1),LALR这几个文法的区别,因为实验的文法过于庞大,导致手动构造分析表难以实现,所以花了大量的时间,去了解分析表的编程构造,也因此对LR（1）有了更深入的了解,在完成分析表构造的同时,也掌握了first集的构造,并完成了项集的构造,在这些的基础上,实现了LR(1)语法分析移进和规约如何对程序进行分析充分掌握并实现了程序的LR(1)自底向上的分析,。

这次实验真的费了我很多的心血,因为是一个人做,为了做这个实验,把课本看了很多遍,也到处去找其他资料,然后独立写了600多行的代码,并且与实验一合并,在实现上诉功能的同时,又花了很多时间来调整输出格式,如分析表等，最终写了1100+行代码,才实现了这个功能。

相对的,这次实验的收获很大,充分掌握了LR(1)文法的实现流程。

## 七、附

核心代码如下:

### 1. void get\_first()

void get\_first()

{

bool flag = true;

while (flag) //只要有更新就一直循环

{

flag = false;

for (int i = 0; i < number; ++i)

{

int p = G[i][0];

int j = 1;

for (; j < length[i]; ++j)

{

int q = G[i][j];

int t = mp["ε"];

if (is\_end[q] == false)//非总结符

{

for (int k = 1; k <= mp\_cnt; ++k)

{

if (k == t) continue;//不能为ε

if (first[q][k] == true && first[p][k] == false)

{

first[p][k] = true;

flag = true;

}

}

}

else//总结符

{

if (first[p][q] == false)

{

first[p][q] = true;

flag = true;

}

break;

}

if (first[q][t] == false) break;//Xi不含有 ε

}

if (j == length[i])

{

int t = mp["ε"];

if (first[p][t] == false)

{

first[p][t] = true;

flag = true;

}

}

}

}

}

### 2. void get\_search(project tmp)

//获取搜索符

void get\_search(project tmp)

{

bool vis[Max\_N] = { false };//去重

buf\_size = 0;

int t = mp["ε"];

int i = tmp.now + 1;

for (; i < length[tmp.num]; ++i)

{

int p = G[tmp.num][i];

if (is\_end[p] == false)//非终结符

{

//p的first集合

for (int j = 0; j <= mp\_cnt; ++j)

{

if (first[p][j] && j != t)

{

if (!vis[j])

buf[buf\_size++] = j, vis[j] = true;

}

}

if (first[p][t] == false) return;

}

else //终结符

{

if (!vis[p])

buf[buf\_size++] = p, vis[p] = true;

return;

}

}

if (!vis[tmp.search])

buf[buf\_size++] = tmp.search, vis[tmp.search] = true;

}

### 3. void e\_closure(int T)

void e\_closure(int T) //求项目集

{

bool flag = true;

while (flag)

{

flag = false;

for (int i = 0; i < items\_size[T]; ++i)

{

int p = G[items[T][i].num][items[T][i].now];//·后面的单词

if (is\_end[p]) continue;//当前是总结符

for (int j = 1; j <= mp\_cnt; ++j)

{

if (G[j][0] == p)

{

get\_search(items[T][i]);

for (int k = 0; k < buf\_size; ++k)

{

project tmp = project(j, 1, buf[k]);

if (is\_in(tmp, T) == false) items[T][items\_size[T]++] = tmp, flag = true;

}

}

}

}

}

}

### 4. void make\_set()

//构建项目集

void make\_set()

{

//初始化第一个项目集

if (!mp["$"])

{

mp["$"] = ++mp\_cnt;

strcpy(String[mp\_cnt], "$");

}

items[0][0] = project(0, 1, mp["$"]);

items\_size[0] = 1;

e\_closure(0);

items\_count = 1;

for (int i = 0; i < items\_count; ++i)//从第一个项目开始扩展

{

for (int j = 1; j <= mp\_cnt; ++j)//暴力每个符号

{

if (j == mp["ε"]) continue;

project tmp[200];

int tmp\_size = 0;

// cout << String[j] << endl;

for (int k = 0; k < items\_size[i]; ++k)//第i个项集中的第k个句子

{

int p = items[i][k].num;

int q = items[i][k].now;

// cout << String[G[p][q]] << endl;

if (q >= length[p]) continue;//到头了

if (G[p][q] == j)

{

tmp[tmp\_size] = items[i][k];

++tmp[tmp\_size].now;

tmp\_size++;

}

}

if (tmp\_size)

{

for (int k = 0; k < tmp\_size; ++k)

items[items\_count][k] = tmp[k];

items\_size[items\_count] = tmp\_size;

e\_closure(items\_count);

int p = is\_in\_project(items\_count);

if (p == 0)//新的项目集

{

trains[i][items\_count] = j;

++items\_count;

}

else//已经存在

{

trains[i][p] = j;

}

}

}

}

}

### 5. void get\_action()

//构造分析表

void get\_action()

{

int t = mp["ε"];

for (int i = 0; i < items\_count; ++i)

{

for (int j = 0; j < items\_size[i]; ++j)

{

int p = items[i][j].num;

int q = items[i][j].now;

//到了a -> bcd· 的情况

if (q == length[p] || (length[p] == 2 && q == 1 && G[p][q] == t))

{

action[i][items[i][j].search].next = items[i][j].num;//归约

action[i][items[i][j].search].op = guiyue;//归约

if (q == length[p] && length[p] == 2 && items[i][j].search == mp["$"]

&& G[p][0] == mp[begin1] && G[p][1] == mp[begin2])

{

action[i][items[i][j].search].op = acc;//归约

}

}

}

}

for (int i = 0; i <= items\_count; ++i)

{

for (int j = 0; j <= items\_count; ++j)

{

if (trains[i][j] != 0)

{

action[i][trains[i][j]].next = j;

action[i][trains[i][j]].op = mov;

}

}

}

}

### 6. void cal()

//分析代码

void cal()

{

read1();

//setO("output");

bool ERROR = false;

state.push(0);

sign.push(mp["$"]);

char s[1500] = " 状态栈 符号栈 输入串 ACTION ";

strcpy(anslysis[anslysis\_len++], s);

int i = 0;

bool end = false;

while (ERROR == false && end == false)

{

int now\_state = state.top();//当前是Ii or Si

int now\_input = read[i];//输入

Print\_state();

Print\_sign();

Print\_input(i);

int op = action[now\_state][now\_input].op;

int next\_state = action[now\_state][now\_input].next;

if (op == error)

{

char p[] = "error!";

strcpy(anslysis[anslysis\_len++], p);

ERROR = true;

continue;

}

else if (op == mov)

{

state.push(next\_state);//状态压栈

sign.push(now\_input);//移入到输入栈

Print\_ACTION(-1);

++i;

}

else if (op == acc || guiyue)

{

int l = length[next\_state] - 1;//A->B

if (empty[next\_state] == true) l = 0;

//弹出l个 状态

for (int i = 0; i < l; ++i) state.pop(), sign.pop();

int t = state.top();

int A = G[next\_state][0];

//GOTO[t,A] 压栈

int tmp = next\_state;

next\_state = action[t][A].next;

state.push(next\_state);

sign.push(A);

if (op == acc)

{

end = true;

Print\_ACTION(-2);//输出A->B

}

else Print\_ACTION(tmp);//输出A->B

}

strcpy(anslysis[anslysis\_len++], pp);

}

}