编译器专题实验报告

**实验四：语法分析（独立模式）**

**一、实验目的**

根据某文法写出SLR(1)分析表。

构造SLR(1)分析表的步骤：

1.写出拓广文法

2.画出项目集规范族

3.求该非终结符的FOLLOW集

4.判断是否是SLR(1)文法

5.构造SLR(1)分析表

**二、实验一（必做）**

**1.实验要求**

参考给出的SLR(1)分析代码，实现SLR(1)分析器的构建。

参考文法：

S->E E->E+T|T T->T\*F|F F->(E)|i

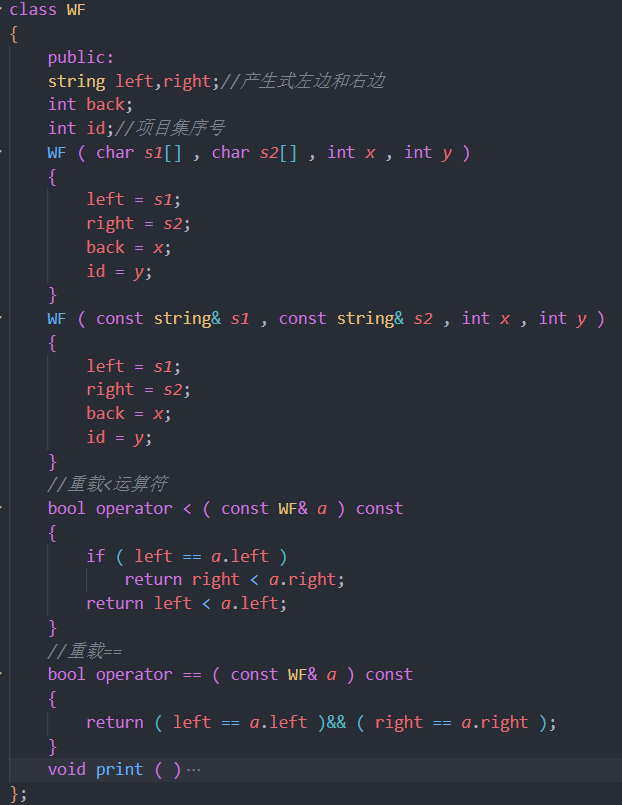
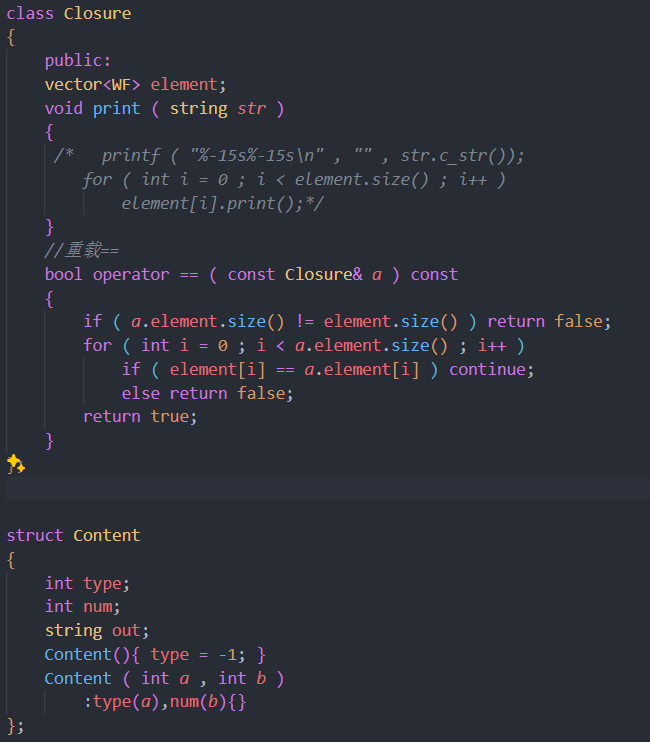
输入字符串或句子i+i\*i。

观察输出分析过程，查看是否可以规约成功。

**2. SLR(1)分析器的构建**

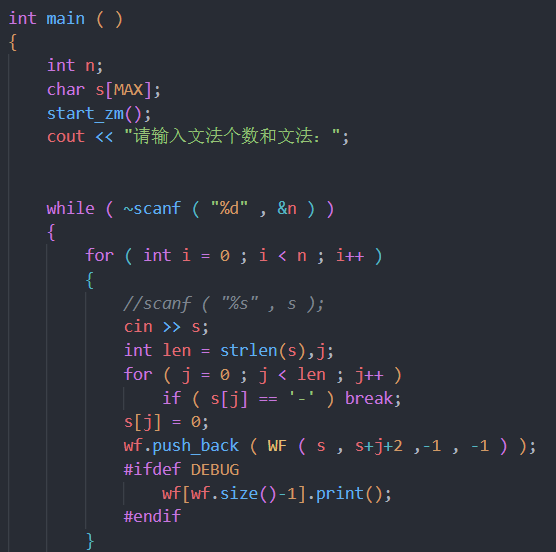
**（1）基本数据结构**

输入的文法是形如A->B的形式，在程序读取文法之后，需要对文法进行保存，程序中使用如下的WF类进行文法的存储。同时，由于SLR(1)文法规范族的构建中出现了闭包，每个闭包中包含多条文法，所以还需要定义一个闭包类。在使用SLR(1)进行文法分析时，分析器需要根据不同的表项进行不同的操作，因此需要一个Content类进行表项的表示。

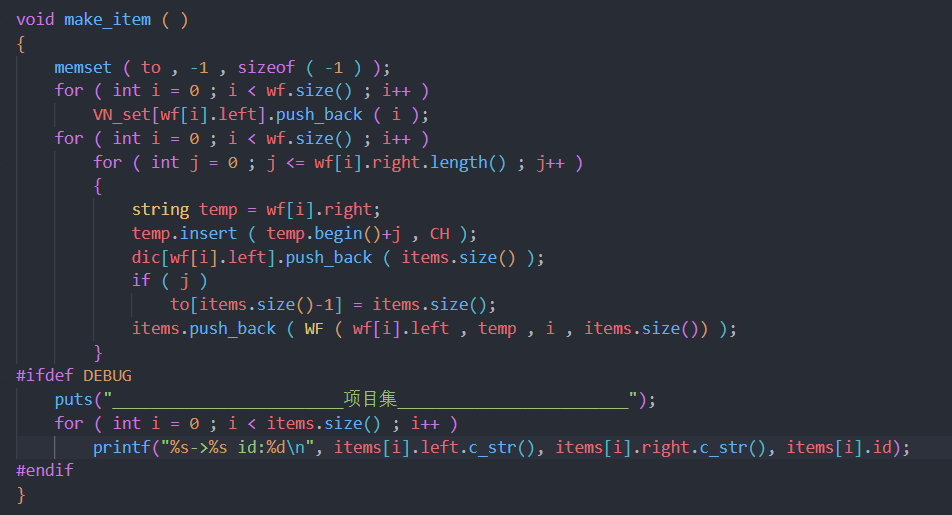
**（2）总体分析过程**

在main函数中，我们首先读取文法的开始字符（start\_zm函数），而后根据文法个数循环读入需要的文法并存在WF类中，随后程序将进行文法项目的构建（make\_item），即在文法中加· ，而后程序将进行FIRST集和FOLLOW集的构建（make\_first和make\_follow），并使用之前得到的文法项目进行规范族的构建（make\_set），make\_V函数生成一个字符集合，包含了所有在文法中出现过的字符。利用此集合，结合之前得到规范族，即可生成需要的SLR分析表。此时，程序将读入一个需要分析的字符串并进行分析（analyse）。

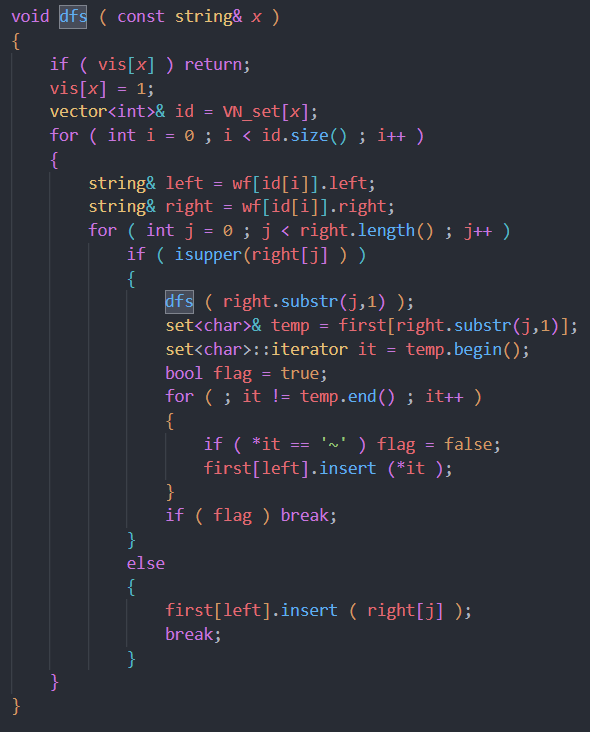
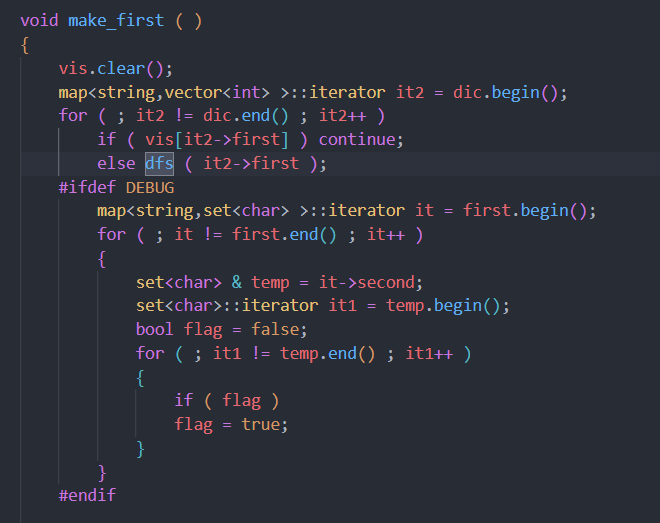
**（3）生成项目集**

通过对于每个产生式调整·的位置，可以得到需要的项目集，具体代码如下所示。

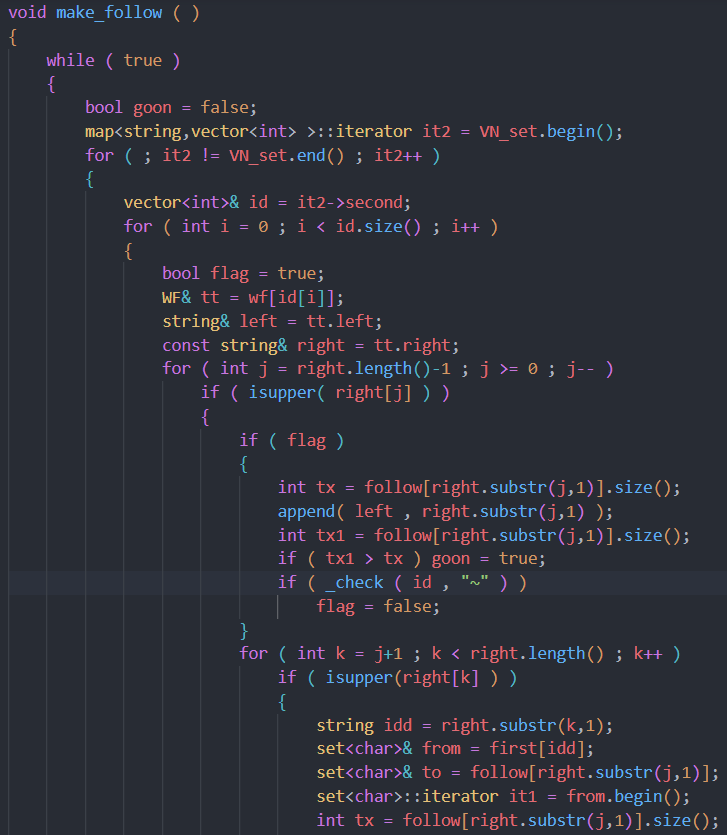
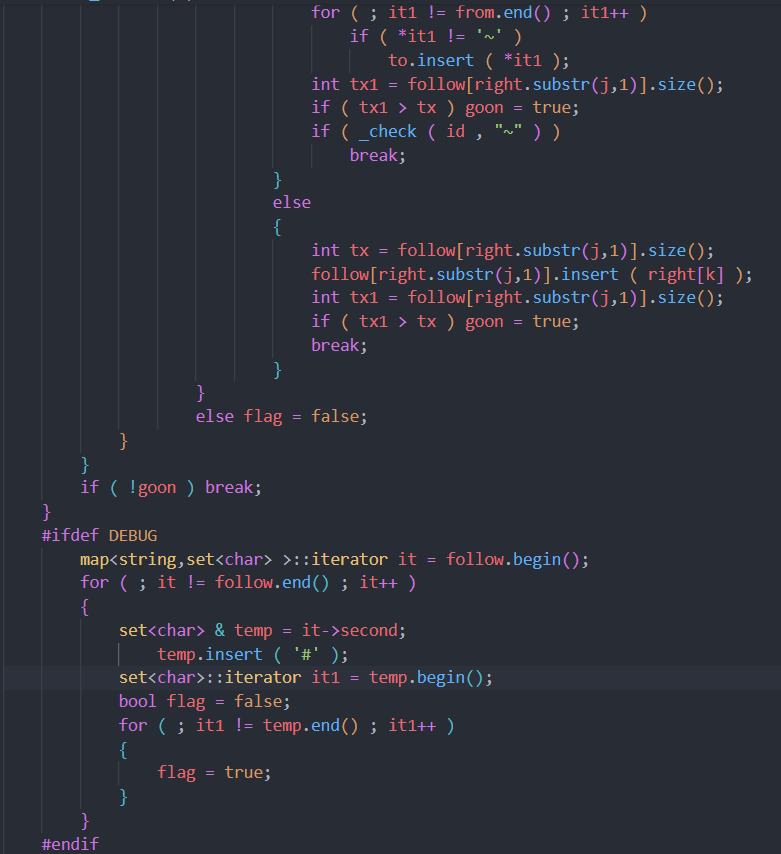
****

**（4）生成FIRST集和FOLLOW集**

使用如下代码可以实现FIRST集和FOLLOW集的构建，其中FIRST集的构建使用到了递归的方法根据深度优先搜索进行深入的求解，FOLLOW集将在后面被用于SLR分析表的生成。

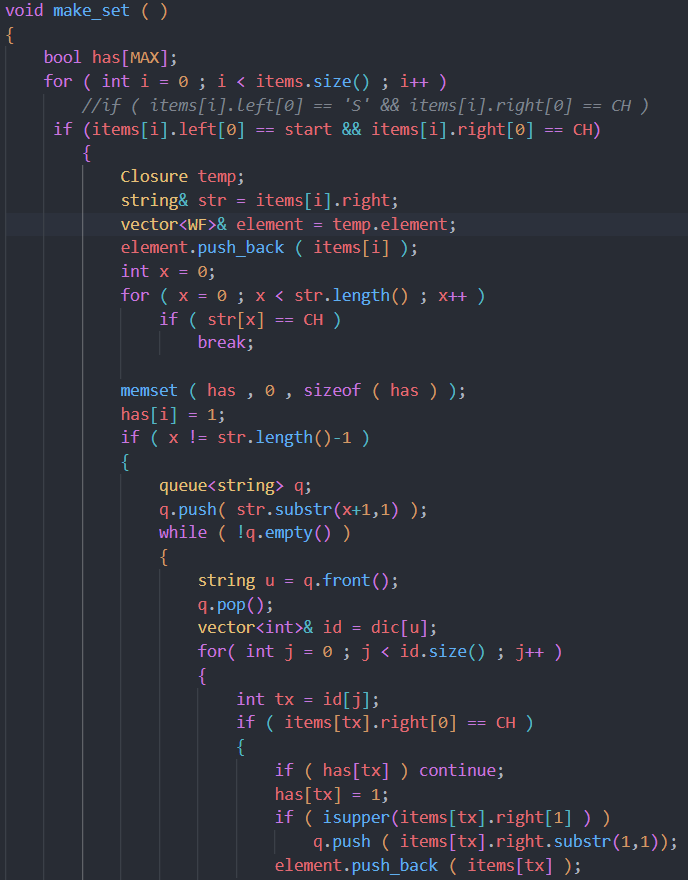
 

构建FOLLOW集时，首先创建了一个无限循环，直到没有新的元素可以添加到任何FOLLOW集中为止。然后，代码遍历所有的非终结符。对于每个非终结符，代码获取其所有的产生式，并对每个产生式的右部进行处理。对于产生式的右部，代码从后向前遍历每个字符。如果字符是大写的（非终结符），那么代码将执行一些操作来更新FOLLOW集。在计算FOLLOW集时，如果出现B->aAb且空串属于FIRST(b)，则需要将FOLLOW(B)中的所有元素放入FOLLOW(A)。这里体现了一开始需要计算FIRST集。以下为计算FOLLOW集的代码。

**（5）生成项目集规范族**

随后，程序将根据得到的项目和进行项目集规范族的构建，其中需要进行闭包的计算，同时，对于每一个闭包都要存储其可以转移到的下一个状态，代码如下所示。

**（6）生成SLR(1)分析表**

根据上述得到的规范族和FOLLOW集，即可进行SLR分析表的构建，分析表的构建按照如下原则：

① 若项目A→ 属于Ik且GO(Ik,a)＝ Ij，a为终结符，且置ACTION[k，a]为“把状态j和符号a移进栈”，简记为“sj”；

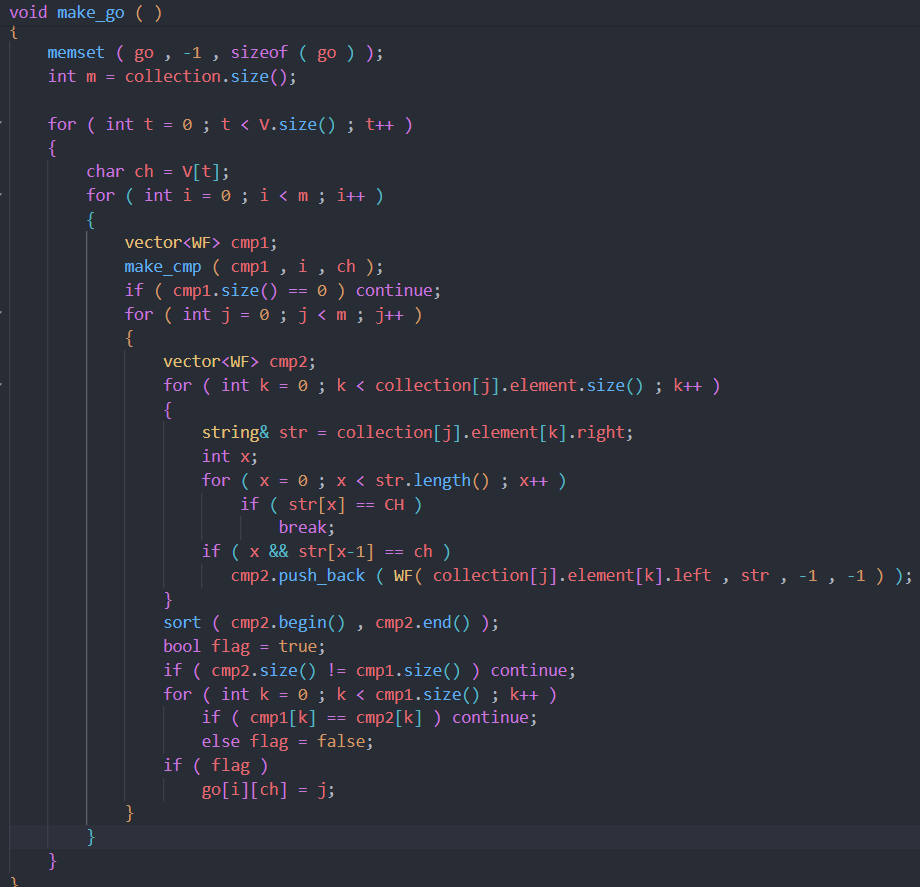
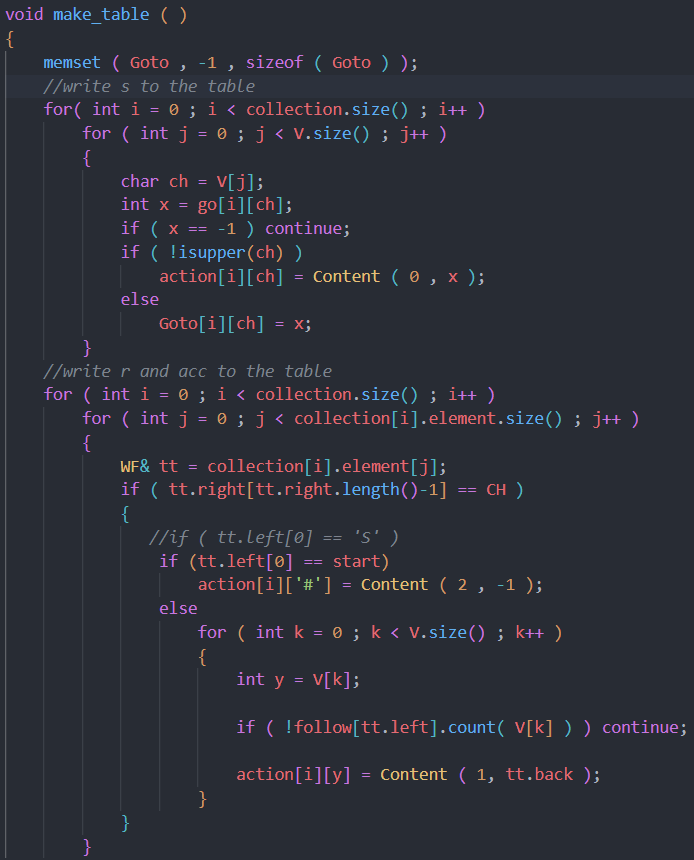
② 若项目A→属于Ik，那么，对任何输入符号a，a∈FOLLOW(A)，置ACTION[k，a]为“用产生式A→进行归约”，简记为“rj”；其中，假定A→为文法G’的第j个产生式；

③ 若项目S’→S·属于Ik，则置ACTION[k，＃]为“接受”，简记为“acc”；

④ 若GO(Ik，A)=Ij，A为非终结符，则置GOTO[k，A]= j；

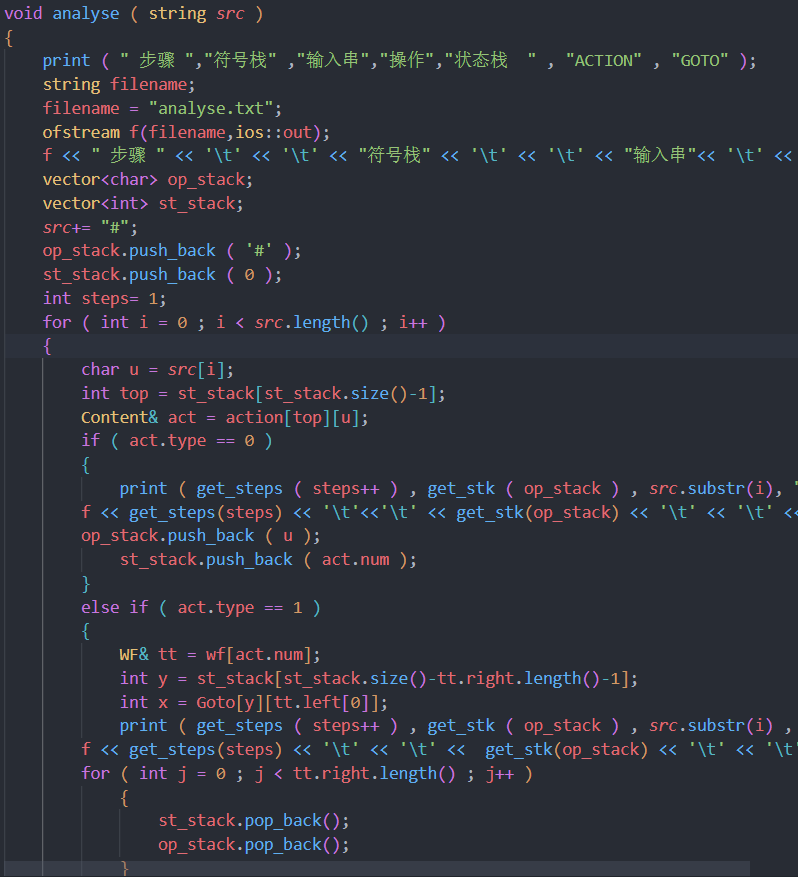
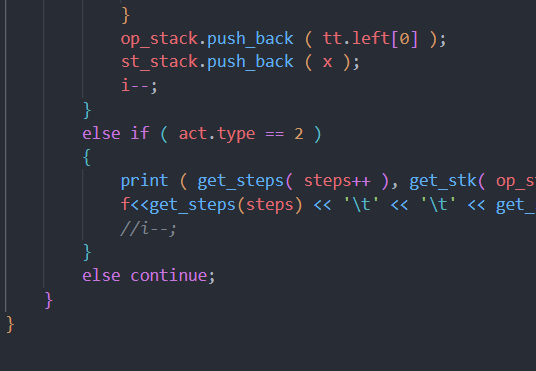
⑤ 分析表中凡不能使用规则1至4填入信息的空白格均置上“出错标志”。

代码如下所示，其中make\_go进行GOTO表的构建，make\_table将生成完整的SLR分析表。

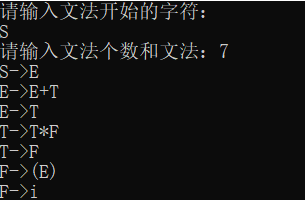
**（7）SLR(1)分析过程**

analyse是SLR分析的主要过程，其使用了双栈进行实现，根据读入的字符和当前的状态，分析器会进行状态转移、规约、接受和报错等处理。具体代码如下所示。

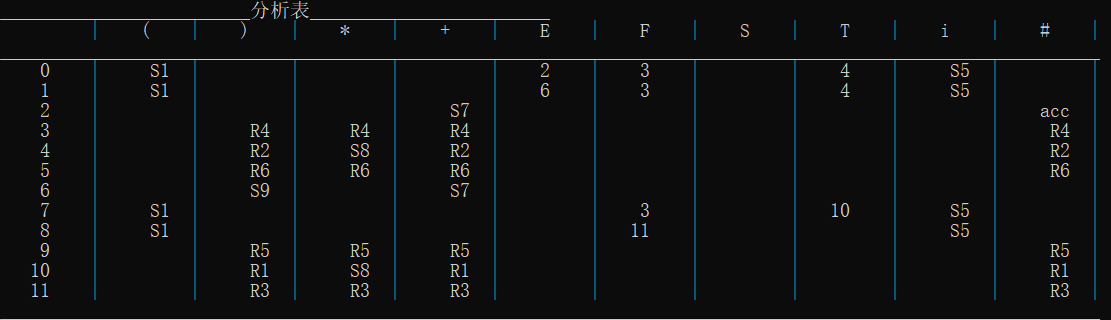
 

**（8）实验验证**

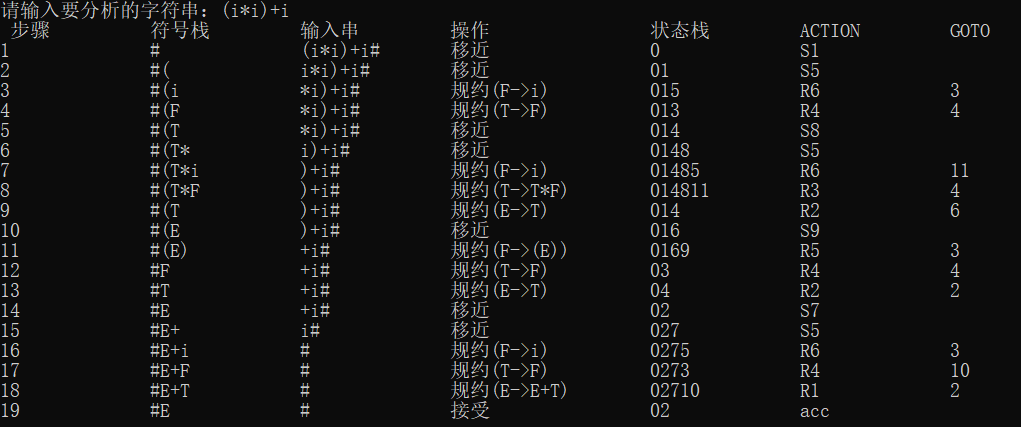
按照上述参考文法，输入文法的开始字符和具体文法，如下所示：



可以得到文法的分析表如下：



输入要分析的字符串(i\*i)+i可以得到如下的分析过程，发现其满足上述分析表的移进规约要求，同时这个字符串是可以被分析器所接受的，所以我们发现在分析文法的最后出现了acc，表示这个串最终被SLR(1)分析器接受。



**三、实验二（选做）**

**1.对比两种文法之间的区别和联系**

上一实验中使用的YACC使用的是LALR文法，而此次实验中使用的是SLR(1)文法。LALR（Look-Ahead LR）和 SLR（Simple LR）都是LR解析器的一种，都是用于构建语法分析器的技术。不过它们之间也有一些不同之处。

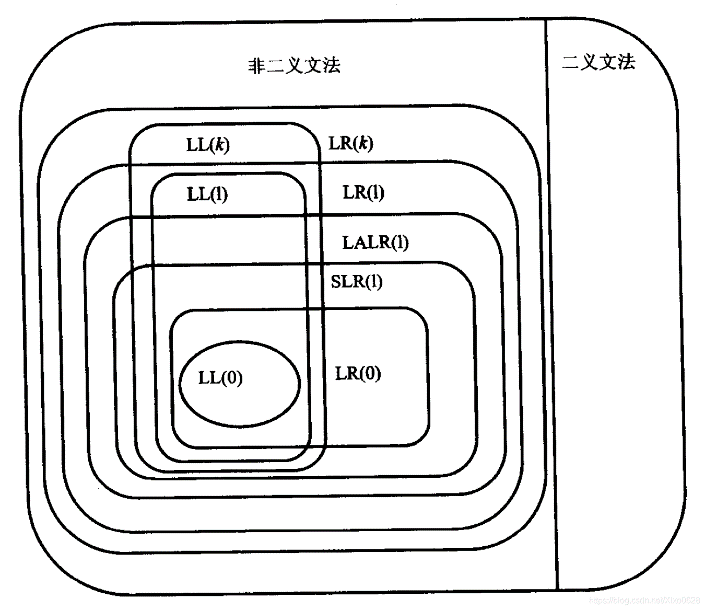
① 项目集的构建：

SLR(1)使用 LR(0)项目集。在构建分析表时，SLR(1) 只关心项目集中项目的核心部分，而不考虑展望符号（lookahead）。 规约操作的选择基于项目集的 FOLLOW 集。LALR(1)使用合并的 LR(1)项目集。LR(1)项目集包含展望符号（lookahead），更精细地处理语法规则，避免某些 SLR(1) 可能遇到的冲突。LALR(1) 项目集通过合并具有相同核心的 LR(1) 项目集而得，这样在保留展望符号的细粒度信息的同时，节省了状态数量。

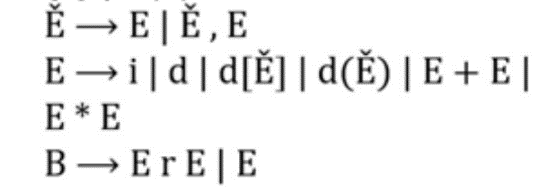
②处理能力：

SLR(1)分析表可能会出现更多的冲突（移入-规约冲突和规约-规约冲突），因为它只考虑 FOLLOW 集，不考虑具体的展望符号。LALR(1)分析表相对更强大，因为它考虑了展望符号，能够处理更多种类的文法（即一些 SLR(1) 无法处理的文法，LALR(1) 可以处理）。

这两类文法和其他文法的关系可以使用下面的图进行表示。



**2. 对本文法表达式部分，分析是否可以实现SLR：**

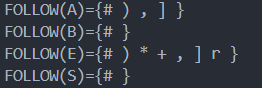


我们将其拓广并将以下的产生式输入到程序中（用A代替）：

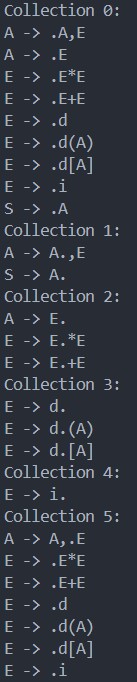
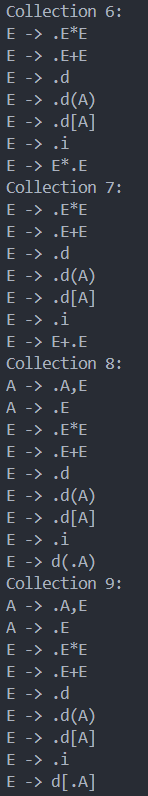
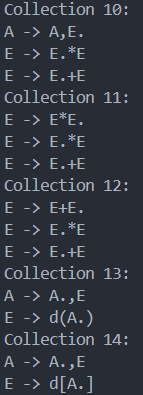
S->A A->E A->A,E E->i

E->d E->d[A] E->d(A) E->E+E E->E\*E B->ErE B->E

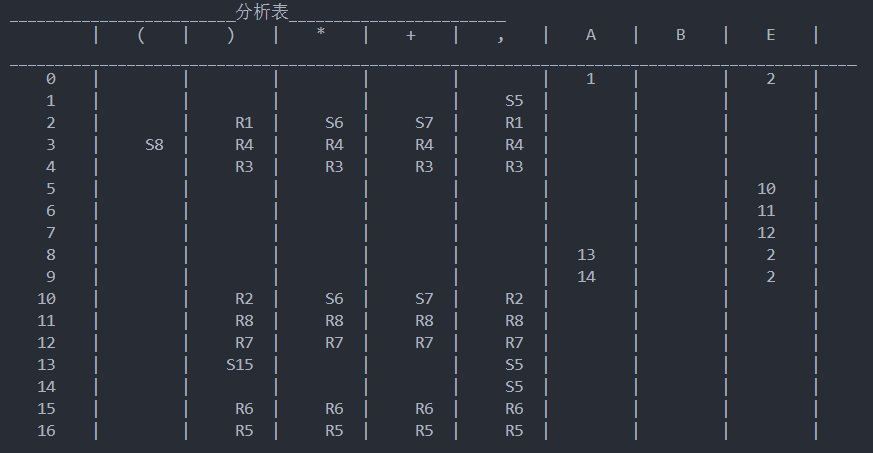
可以得到其FOLLOW集：



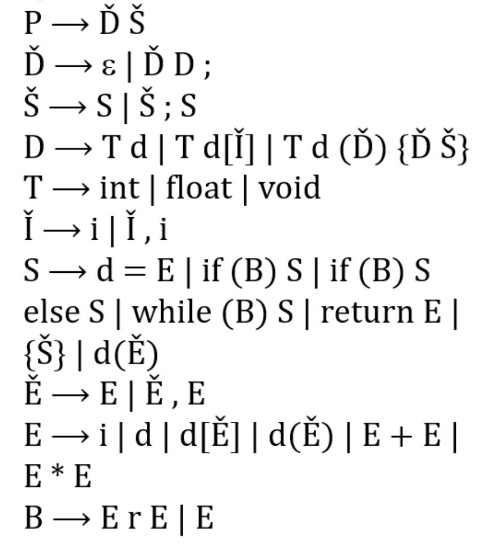
可以得到规范族：

通过分析FOLLOW集和规范族，可以发现该文法满足SLR(1)，得到分析表如下：



同理，对于以下文法，我们也可以判断其不是SLR(1)文法



将上述文法换成以下文法内容：

Z->P P->FG F-> F->FD; G->S G->G;S D->Td D->Td[H]

D->Td(F){FG} T->a T->b T->c H->i H->H,I S->d=E

S->e(B)S S->e(B)SfS S->g(B)S S->hE S->{G} S->d(A)

A->E A->A,E E->i E->d E->d[A] E->d(A) E->E+E

E->E\*E B->ErE B->E

其中int表示为a float表示为b void表示为c if表示为e else表示为f

while表示为g return表示为h

将上述内容输入代码，发现在计算FOLLOW集时发现错误，即判断其不是SLR(1)文法。

**四、遇到的问题**

编译原理——理解LL/LR/SLR/LALR

https://blog.csdn.net/Xixo0628/article/details/105658030

**五、附件：代码**

#include <iostream>

#include <cstdio>

#include <algorithm>

#include <cstring>

#include <cctype>

#include <vector>

#include <string>

#include <queue>

#include <map>

#include <set>

#include <sstream>

#include<fstream>

#define MAX 507

#define DEBUG

//#ifdef \_DEBUG的意思是如果预定义了名字\_DEBUG，则函数执行该语句里的代码, 并且抛弃下面的代码

//否则抛弃上面的代码而执行#else里的，

//#endif表示该预处理结束。

using namespace std;

//文法类

class WF

{

public:

string left,right;//产生式左边和右边

int back;

int id;//项目集序号

WF ( char s1[] , char s2[] , int x , int y )

{

left = s1;

right = s2;

back = x;

id = y;

}

WF ( const string& s1 , const string& s2 , int x , int y )

{

left = s1;

right = s2;

back = x;

id = y;

}

//重载<运算符

bool operator < ( const WF& a ) const

{

if ( left == a.left )

return right < a.right;

return left < a.left;

}

//重载==

bool operator == ( const WF& a ) const

{

return ( left == a.left )&& ( right == a.right );

}

void print ( )

{

///\*printf ( "%s->%s\n" , left.c\_str() , right.c\_str() );\*/

}

};

//闭包类

class Closure

{

public:

vector<WF> element;

void print ( string str )

{

/\* printf ( "%-15s%-15s\n" , "" , str.c\_str());

for ( int i = 0 ; i < element.size() ; i++ )

element[i].print();\*/

}

//重载==

bool operator == ( const Closure& a ) const

{

if ( a.element.size() != element.size() ) return false;

for ( int i = 0 ; i < a.element.size() ; i++ )

if ( element[i] == a.element[i] ) continue;

else return false;

return true;

}

};

struct Content

{

int type;

int num;

string out;

Content(){ type = -1; }

Content ( int a , int b )

:type(a),num(b){}

};

vector<WF> wf;

map<string,vector<int> > dic;

map<string,vector<int> > VN\_set;

map<string,bool> vis;

//定义文法的开始字符

char start;

void start\_zm()

{

char start\_s;

cout << "请输入文法开始的字符：" << endl;

cin >> start\_s;

start = start\_s;

}

//string start = "S";

vector<Closure> collection;

vector<WF> items;

char CH = '.';

int go[MAX][MAX];

int to[MAX];

vector<char> V;

bool used[MAX];

Content action[MAX][MAX];

int Goto[MAX][MAX];

map<string,set<char> > first;

map<string,set<char> > follow;

void make\_item ( )

{

memset ( to , -1 , sizeof ( -1 ) );

for ( int i = 0 ; i < wf.size() ; i++ )

VN\_set[wf[i].left].push\_back ( i );

for ( int i = 0 ; i < wf.size() ; i++ )

for ( int j = 0 ; j <= wf[i].right.length() ; j++ )

{

string temp = wf[i].right;

temp.insert ( temp.begin()+j , CH );

dic[wf[i].left].push\_back ( items.size() );

if ( j )

to[items.size()-1] = items.size();

items.push\_back ( WF ( wf[i].left , temp , i , items.size()) );

}

#ifdef DEBUG

puts("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_项目集\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

for ( int i = 0 ; i < items.size() ; i++ )

printf("%s->%s id:%d\n", items[i].left.c\_str(), items[i].right.c\_str(), items[i].id);

#endif

}

void dfs ( const string& x )

{

if ( vis[x] ) return;

vis[x] = 1;

vector<int>& id = VN\_set[x];

for ( int i = 0 ; i < id.size() ; i++ )

{

string& left = wf[id[i]].left;

string& right = wf[id[i]].right;

for ( int j = 0 ; j < right.length() ; j++ )

if ( isupper(right[j] ) )

{

dfs ( right.substr(j,1) );

set<char>& temp = first[right.substr(j,1)];

set<char>::iterator it = temp.begin();

bool flag = true;

for ( ; it != temp.end() ; it++ )

{

if ( \*it == '~' ) flag = false;

first[left].insert (\*it );

}

if ( flag ) break;

}

else

{

first[left].insert ( right[j] );

break;

}

}

}

void make\_first ( )

{

vis.clear();

map<string,vector<int> >::iterator it2 = dic.begin();

for ( ; it2 != dic.end() ; it2++ )

if ( vis[it2->first] ) continue;

else dfs ( it2->first );

#ifdef DEBUG

map<string,set<char> >::iterator it = first.begin();

for ( ; it != first.end() ; it++ )

{

set<char> & temp = it->second;

set<char>::iterator it1 = temp.begin();

bool flag = false;

for ( ; it1 != temp.end() ; it1++ )

{

if ( flag )

flag = true;

}

}

#endif

// 输出first集

/\*cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_FIRST集\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

for (auto it = first.begin(); it != first.end(); ++it)

{

cout << "FIRST(" << it->first << ")={";

for (auto it1 = it->second.begin(); it1 != it->second.end(); ++it1)

{

cout << \*it1 << " ";

}

cout << "}" << endl;

}\*/

}

void append ( const string& str1 , const string& str2 )

{

set<char>& from = follow[str1];

set<char>& to = follow[str2];

set<char>::iterator it = from.begin();

for ( ; it != from.end() ; it++ )

to.insert ( \*it );

}

bool \_check ( const vector<int>& id, const string str )

{

for ( int i = 0 ; i < id.size() ; i++ )

{

int x = id[i];

if ( wf[x].right == str ) return true;

}

return false;

}

void make\_follow ( )

{

while ( true )

{

bool goon = false;

map<string,vector<int> >::iterator it2 = VN\_set.begin();

for ( ; it2 != VN\_set.end() ; it2++ )

{

vector<int>& id = it2->second;

for ( int i = 0 ; i < id.size() ; i++ )

{

bool flag = true;

WF& tt = wf[id[i]];

string& left = tt.left;

const string& right = tt.right;

for ( int j = right.length()-1 ; j >= 0 ; j-- )

if ( isupper( right[j] ) )

{

if ( flag )

{

int tx = follow[right.substr(j,1)].size();

append( left , right.substr(j,1) );

int tx1 = follow[right.substr(j,1)].size();

if ( tx1 > tx ) goon = true;

if ( \_check ( id , "~" ) )

flag = false;

}

for ( int k = j+1 ; k < right.length() ; k++ )

if ( isupper(right[k] ) )

{

string idd = right.substr(k,1);

set<char>& from = first[idd];

set<char>& to = follow[right.substr(j,1)];

set<char>::iterator it1 = from.begin();

int tx = follow[right.substr(j,1)].size();

for ( ; it1 != from.end() ; it1++ )

if ( \*it1 != '~' )

to.insert ( \*it1 );

int tx1 = follow[right.substr(j,1)].size();

if ( tx1 > tx ) goon = true;

if ( \_check ( id , "~" ) )

break;

}

else

{

int tx = follow[right.substr(j,1)].size();

follow[right.substr(j,1)].insert ( right[k] );

int tx1 = follow[right.substr(j,1)].size();

if ( tx1 > tx ) goon = true;

break;

}

}

else flag = false;

}

}

if ( !goon ) break;

}

#ifdef DEBUG

map<string,set<char> >::iterator it = follow.begin();

for ( ; it != follow.end() ; it++ )

{

set<char> & temp = it->second;

temp.insert ( '#' );

set<char>::iterator it1 = temp.begin();

bool flag = false;

for ( ; it1 != temp.end() ; it1++ )

{

flag = true;

}

}

#endif

// 输出follow集

/\*cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_FOLLOW集\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

for (auto it = follow.begin(); it != follow.end(); ++it)

{

cout << "FOLLOW(" << it->first << ")={";

for (auto it1 = it->second.begin(); it1 != it->second.end(); ++it1)

{

cout << \*it1 << " ";

}

cout << "}" << endl;

}\*/

}

void make\_set ( )

{

bool has[MAX];

for ( int i = 0 ; i < items.size() ; i++ )

//if ( items[i].left[0] == 'S' && items[i].right[0] == CH )

if (items[i].left[0] == start && items[i].right[0] == CH)

{

Closure temp;

string& str = items[i].right;

vector<WF>& element = temp.element;

element.push\_back ( items[i] );

int x = 0;

for ( x = 0 ; x < str.length() ; x++ )

if ( str[x] == CH )

break;

memset ( has , 0 , sizeof ( has ) );

has[i] = 1;

if ( x != str.length()-1 )

{

queue<string> q;

q.push( str.substr(x+1,1) );

while ( !q.empty() )

{

string u = q.front();

q.pop();

vector<int>& id = dic[u];

for( int j = 0 ; j < id.size() ; j++ )

{

int tx = id[j];

if ( items[tx].right[0] == CH )

{

if ( has[tx] ) continue;

has[tx] = 1;

if ( isupper(items[tx].right[1] ) )

q.push ( items[tx].right.substr(1,1));

element.push\_back ( items[tx] );

}

}

}

}

collection.push\_back ( temp );

}

for ( int i = 0 ; i < collection.size() ; i++ )

{

map<int,Closure> temp;

for ( int j = 0 ; j < collection[i].element.size() ; j++ )

{

string str = collection[i].element[j].right;

int x = 0;

for ( ; x < str.length() ; x++ )

if ( str[x] == CH ) break;

if ( x == str.length()-1 )

continue;

int y = str[x+1];

int ii;

//cout << i << "previous: " << str << endl;

str.erase ( str.begin()+x);

str.insert ( str.begin()+x+1 , CH );

//cout << i <<"after: " << str << endl;

WF cmp = WF ( collection[i].element[j].left , str , -1 , -1 );

for ( int k = 0 ; k< items.size() ; k++ )

if ( items[k] == cmp )

{

ii = k;

break;

}

//string& str1 = items[ii].right;

memset ( has , 0 , sizeof ( has ) );

vector<WF>& element = temp[y].element;

element.push\_back ( items[ii] );

has[ii] = 1;

x++;

if ( x != str.length()-1 )

{

queue<string> q;

q.push( str.substr(x+1,1) );

while ( !q.empty() )

{

string u = q.front();

q.pop();

vector<int>& id = dic[u];

for( int j = 0 ; j < id.size() ; j++ )

{

int tx = id[j];

if ( items[tx].right[0] == CH )

{

if ( has[tx] ) continue;

has[tx] = 1;

if ( isupper(items[tx].right[1] ) )

q.push ( items[tx].right.substr(1,1));

element.push\_back ( items[tx] );

}

}

}

}

}

map<int,Closure>::iterator it = temp.begin();

for ( ; it != temp.end() ; it++ )

collection.push\_back ( it->second );

for ( int i = 0 ; i < collection.size() ; i++ )

sort ( collection[i].element.begin() , collection[i].element.end() );

for ( int i = 0 ; i < collection.size() ; i++ )

for ( int j = i+1 ; j < collection.size() ; j++ )

if ( collection[i] == collection[j] )

collection.erase ( collection.begin()+j );

}

/\*\*/

for ( int i = 0 ; i < collection.size() ; i++ )

{

std::cout << "Collection " << i << ":\n";

for (const auto& item : collection[i].element)

{

std::cout << item.left << " -> ";

for (const auto& ch : item.right)

{

std::cout << ch;

}

std::cout << "\n";

}

}

}

void make\_V ( )

{

memset ( used , 0 , sizeof ( used ) );

for ( int i = 0 ; i < wf.size() ; i++ )

{

string& str = wf[i].left;

for ( int j = 0 ; j < str.length() ; j++ )

{

if ( used[str[j]] ) continue;

used[str[j]] = 1;

V.push\_back ( str[j] );

}

string& str1 = wf[i].right;

for ( int j = 0 ; j < str1.length() ; j++ )

{

if ( used[str1[j]] ) continue;

used[str1[j]] = 1;

V.push\_back ( str1[j] );

}

}

sort ( V.begin() , V.end() );

V.push\_back ( '#' );

}

void make\_cmp ( vector<WF>& cmp1 , int i , char ch )

{

for ( int j = 0 ; j < collection[i].element.size() ; j++ )

{

string str = collection[i].element[j].right;

int k;

for ( k = 0 ; k < str.length() ; k++ )

if ( str[k] == CH )

break;

if ( k != str.length() - 1 && str[k+1] == ch )

{

str.erase ( str.begin()+k);

str.insert ( str.begin()+k+1 , CH );

cmp1.push\_back ( WF ( collection[i].element[j].left , str , -1 , -1 ) );

}

}

sort ( cmp1.begin() , cmp1.end() );

}

void make\_go ( )

{

memset ( go , -1 , sizeof ( go ) );

int m = collection.size();

for ( int t = 0 ; t < V.size() ; t++ )

{

char ch = V[t];

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

{

vector<WF> cmp1;

make\_cmp ( cmp1 , i , ch );

if ( cmp1.size() == 0 ) continue;

for ( int j = 0 ; j < m ; j++ )

{

vector<WF> cmp2;

for ( int k = 0 ; k < collection[j].element.size() ; k++ )

{

string& str = collection[j].element[k].right;

int x;

for ( x = 0 ; x < str.length() ; x++ )

if ( str[x] == CH )

break;

if ( x && str[x-1] == ch )

cmp2.push\_back ( WF( collection[j].element[k].left , str , -1 , -1 ) );

}

sort ( cmp2.begin() , cmp2.end() );

bool flag = true;

if ( cmp2.size() != cmp1.size() ) continue;

for ( int k = 0 ; k < cmp1.size() ; k++ )

if ( cmp1[k] == cmp2[k] ) continue;

else flag = false;

if ( flag )

go[i][ch] = j;

}

}

}

}

void make\_table ( )

{

memset ( Goto , -1 , sizeof ( Goto ) );

//write s to the table

for( int i = 0 ; i < collection.size() ; i++ )

for ( int j = 0 ; j < V.size() ; j++ )

{

char ch = V[j];

int x = go[i][ch];

if ( x == -1 ) continue;

if ( !isupper(ch) )

action[i][ch] = Content ( 0 , x );

else

Goto[i][ch] = x;

}

//write r and acc to the table

for ( int i = 0 ; i < collection.size() ; i++ )

for ( int j = 0 ; j < collection[i].element.size() ; j++ )

{

WF& tt = collection[i].element[j];

if ( tt.right[tt.right.length()-1] == CH )

{

//if ( tt.left[0] == 'S' )

if (tt.left[0] == start)

action[i]['#'] = Content ( 2 , -1 );

else

for ( int k = 0 ; k < V.size() ; k++ )

{

int y = V[k];

if ( !follow[tt.left].count( V[k] ) ) continue;

action[i][y] = Content ( 1, tt.back );

}

}

}

#ifdef DEBUG

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_分析表\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

printf ( "%10s%5c%5s" , "|" , V[0] , "|");

for ( int i = 1 ; i < V.size() ; i++ )

printf ( "%5c%5s" , V[i] , "|" );

puts ("");

for ( int i = 0 ; i < (V.size()+1)\*10 ; i++ )

printf ( "\_" );

puts("");

stringstream sin;

for ( int i = 0 ; i < collection.size() ; i++ )

{

printf ( "%5d%5s" , i , "|" );

for ( int j = 0 ; j < V.size() ; j++ )

{

char ch = V[j];

if ( isupper(ch) )

{

if ( Goto[i][ch] == -1 )

printf ( "%10s" , "|" );

else

printf ( "%5d%5s" , Goto[i][ch] , "|" );

}

else

{

sin.clear();

if ( action[i][ch].type == -1 )

printf ( "%10s" , "|" );

else

{

Content& temp = action[i][ch];

if ( temp.type == 0 )

sin << "S";

if ( temp.type == 1 )

sin << "R";

if ( temp.type == 2 )

sin << "acc";

if ( temp.num != -1 )

sin << temp.num;

sin >> temp.out;

printf ( "%7s%3s" , temp.out.c\_str() , "|" );

}

}

}

puts ("");

}

for ( int i = 0 ; i < (V.size()+1)\*10 ; i++ )

printf ( "\_" );

puts("");

#endif

}

void print ( string s1 , string s2 , string s3 , string s4 , string s5 , string s6 , string s7 )

{

printf("%-15s%-15s%-15s%-20s%-15s%-15s%-15s\n", s1.c\_str(), s2.c\_str(), s3.c\_str(), s4.c\_str(), s5.c\_str(),

s6.c\_str(), s7.c\_str());

}

string get\_steps ( int x )

{

stringstream sin;

sin << x;

string ret;

sin >> ret;

return ret;

}

template<class T>

string get\_stk ( vector<T> stk )

{

stringstream sin;

for ( int i = 0 ; i < stk.size() ; i++ )

sin << stk[i];

string ret;

sin >> ret;

return ret;

}

string get\_shift ( WF& temp )

{

stringstream sin;

sin << "规约(" << temp.left << "->" << temp.right <<")";

string out;

sin >> out;

return out;

}

void analyse ( string src )

{

print ( " 步骤 ","符号栈" ,"输入串","操作","状态栈 " , "ACTION" , "GOTO" );

string filename;

filename = "analyse.txt";

ofstream f(filename,ios::out);

f << " 步骤 " << '\t' << '\t' << "符号栈" << '\t' << '\t' << "输入串"<< '\t' << '\t' << "操作" << '\t' << '\t' << "状态栈 " << '\t' << '\t' << "ACTION" << '\t' << '\t' << "GOTO" << endl;

vector<char> op\_stack;

vector<int> st\_stack;

src+= "#";

op\_stack.push\_back ( '#' );

st\_stack.push\_back ( 0 );

int steps= 1;

for ( int i = 0 ; i < src.length() ; i++ )

{

char u = src[i];

int top = st\_stack[st\_stack.size()-1];

Content& act = action[top][u];

if ( act.type == 0 )

{

print ( get\_steps ( steps++ ) , get\_stk ( op\_stack ) , src.substr(i), "移近", get\_stk( st\_stack ) , act.out , "" );

f << get\_steps(steps) << '\t'<<'\t' << get\_stk(op\_stack) << '\t' << '\t' << src.substr(i) << '\t' << '\t' << "移近" << '\t' << '\t' << get\_stk(st\_stack) << '\t' << '\t' << act.out << '\t' << '\t' << "" << endl;

op\_stack.push\_back ( u );

st\_stack.push\_back ( act.num );

}

else if ( act.type == 1 )

{

WF& tt = wf[act.num];

int y = st\_stack[st\_stack.size()-tt.right.length()-1];

int x = Goto[y][tt.left[0]];

print ( get\_steps ( steps++ ) , get\_stk ( op\_stack ) , src.substr(i) , get\_shift(tt) ,get\_stk( st\_stack),act.out,get\_steps(x));

f << get\_steps(steps) << '\t' << '\t' << get\_stk(op\_stack) << '\t' << '\t' << src.substr(i) << '\t' << '\t' << get\_shift(tt) << '\t' << '\t' << get\_stk(st\_stack) << '\t' << '\t' << act.out << '\t' << '\t' << get\_steps(x) << endl;

for ( int j = 0 ; j < tt.right.length() ; j++ )

{

st\_stack.pop\_back();

op\_stack.pop\_back();

}

op\_stack.push\_back ( tt.left[0] );

st\_stack.push\_back ( x );

i--;

}

else if ( act.type == 2 )

{

print ( get\_steps( steps++ ), get\_stk( op\_stack ) , src.substr(i) , "接受" , get\_stk(st\_stack) , act.out , "" );

f<<get\_steps(steps) << '\t' << '\t' << get\_stk(op\_stack) << '\t' << '\t' << src.substr(i) << '\t' << '\t' << "接受" << '\t' << '\t' << get\_stk(st\_stack) << '\t' << '\t' << act.out << '\t' << '\t' << ""<<endl;

//i--;

}

else continue;

}

}

int main ( )

{

int n;

char s[MAX];

start\_zm();

cout << "请输入文法个数和文法：";

while ( ~scanf ( "%d" , &n ) )

{

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

{

//scanf ( "%s" , s );

cin >> s;

int len = strlen(s),j;

for ( j = 0 ; j < len ; j++ )

if ( s[j] == '-' ) break;

s[j] = 0;

wf.push\_back ( WF ( s , s+j+2 ,-1 , -1 ) );

#ifdef DEBUG

wf[wf.size()-1].print();

#endif

}

make\_item();

make\_first();

make\_follow();

make\_set();

make\_V();

make\_go();

make\_table();

string s1;

cout << "请输入要分析的字符串：";

cin >> s1;

analyse ( s1);

}

}