### **第三次实验报告**

### 本次实验亮点部分：

第三次实验分别用4种方式（递归子程序下降法、LL(1)分析法、LR(0)分析法和算符优先分析法）实现四元式语法制导翻译；

### **实验目标：**

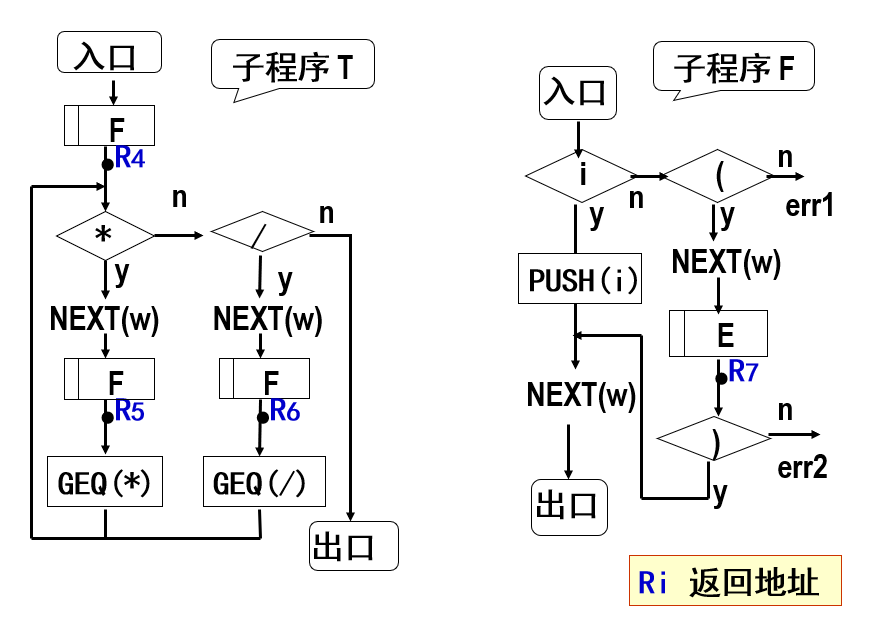
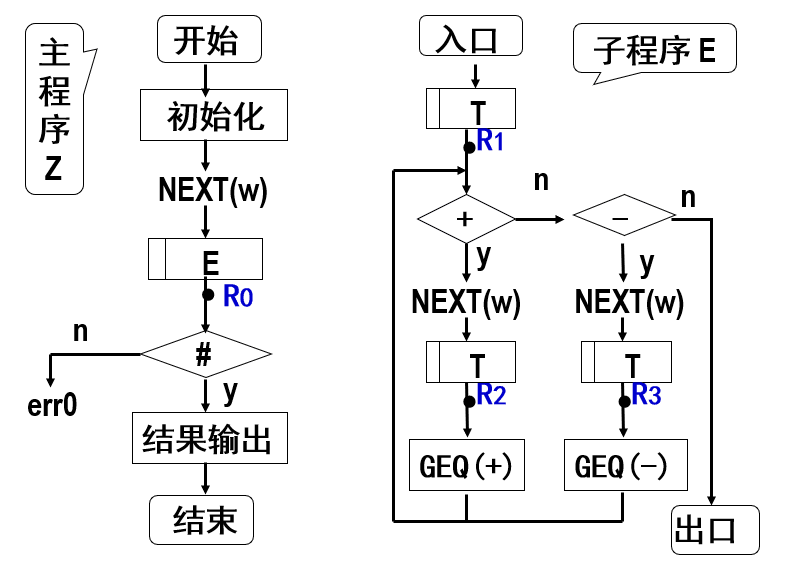
设计实现四元式语法制导翻译文法。

### **3.1 递归子程序下降法**

实验内容：

（包括：概要设计、数据结构、流程图、关键函数等有选择填写）

流程图如下所示：



在原来的实验二的递归子程序下降法代码基础上，在合适的地方加上四元式翻译动作。

源程序代码：（加入注释）

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FileName:GenerationRD.cpp

Function: Recursive subroutine descent method is used for

Quaternion grammar-guided translation.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <string>

#include <stack>

#include "WordFilter1.h"

using namespace std;

int tokens[1024], tokenNumNow = 0;

void functionE();

void functionT();

void functionF();

bool flag = 1;

stack<int> symbol, wm;

int temp = 1001;

int main()

{

wordFilter(tokens);

functionE();

if(flag == 1 && tokens[tokenNumNow] == 405)

cout<<"Successful!"<<endl;

else

cout<<"Error!"<<endl;

return 0;

}

void functionE()

{

functionT();

while(tokens[tokenNumNow]==401)

{

//cout<<tokens[tokenNumNow]<<endl;

        wm.push(tokens[tokenNumNow]);

tokenNumNow++;

functionT();

//从E进入的T子程序跳出时，输出四元式

        cout<<"("<<wm.top()<<" "<<symbol.top();

        wm.pop();

        symbol.pop();

        cout<<" "<<symbol.top()<<" "<<temp<<")"<<endl;

        symbol.pop();

        symbol.push(temp);

        temp = temp+1;

}

}

void functionT()

{

functionF();

while(tokens[tokenNumNow] == 402)

{

//cout<<tokens[tokenNumNow]<<endl;

        wm.push(tokens[tokenNumNow]);

tokenNumNow++;

functionF();

//从T进入的F子程序跳出时，输出四元式

        cout<<"("<<wm.top()<<" "<<symbol.top();

        wm.pop();

        symbol.pop();

        cout<<" "<<symbol.top()<<" "<<temp<<")"<<endl;

        symbol.pop();

        symbol.push(temp);

        temp = temp+1;

}

}

void functionF()

{

int flagF1=0,flagF2=0;

if(tokens[tokenNumNow]>=200 && tokens[tokenNumNow]<=399)

{

flagF1=1;

//cout<<tokens[tokenNumNow]<<endl;

        symbol.push(tokens[tokenNumNow]);

tokenNumNow++;

}

else if(tokens[tokenNumNow]==403)

{

//cout<<tokens[tokenNumNow]<<endl;

tokenNumNow++;

functionE();

if(tokens[tokenNumNow]==404)

{

flagF2 = 1;

//cout<<tokens[tokenNumNow]<<endl;

tokenNumNow++;

}

}

else

{

flag = 0;

}

if(!flagF1 && !flagF2)

{

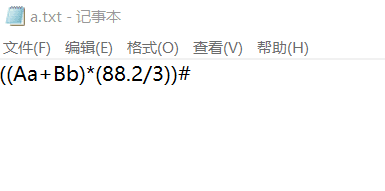
flag = 0;

}

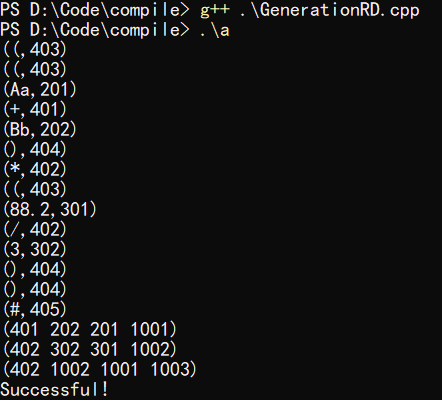
}

程序运行结果：（截屏）

输入：



输出：

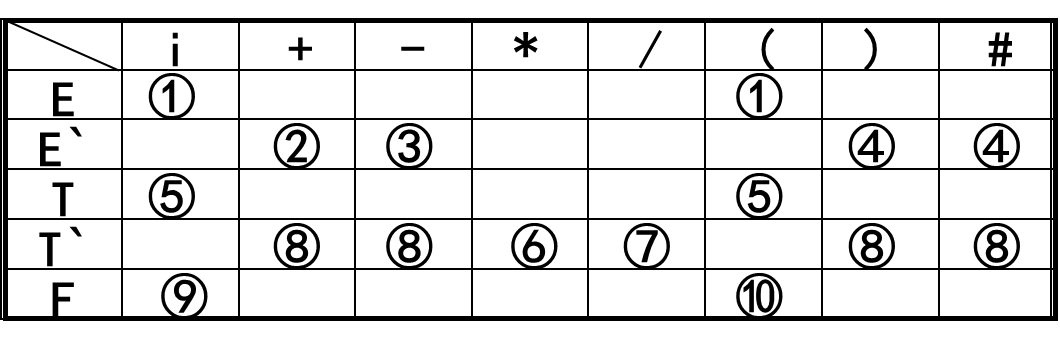


输出说明：每个四元式的前三项是使用token代替，与上方对照可知；四元式最后一项用四位数字XXXX代替t1、t2、t3...tn；因token中未对\*/做区分，因此后两项均为“402”。

### **3.2 LL(1)方法**

实验内容：

LL（1）分析表如下：



跟递归子程序下降法相似，需要在合适的地方插入翻译语义动作函数。

源程序代码：（加入注释）

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FileName:GenerationLL.cpp

Function: LL (1) method is used for Quaternary grammar-guided translation.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <stack>

#include "WordFilter1.h"

using namespace std;

void reverseStack(int order);

int lookupTable(char x, int w);

bool endIsRight(char x, int w);

stack<char> s;

stack<int> symbol, wm;

int tokens[1024], tokenNumNow = 0, temp = 1001;

bool flag = 1;

//产生式逆序压栈

char E[2] = {'T','R'};

char R[3] = {'W','T','R'};

char T[2] = {'F','Y'};

char Y[3] = {'M','F','Y'};

char F[3] = {'(','E',')'};

//终结符表

char endSymbol[6] = {'I','W','M','(',')','#'};

//LL（1）分析表

int analysisTable[5][6] =

{

1, 0, 0, 1, 0, 0,

0, 2, 0, 0, 3, 3,

4, 0, 0, 4, 0, 0,

0, 6, 5, 0, 6, 6,

7, 0, 0, 8, 0, 0

};

int main()

{

char x;

int w;

wordFilter(tokens);

s.push('#');

s.push('E');

while(x != '#')

{

x = s.top();

s.pop();

w = tokens[tokenNumNow];

//cout<<"x="<<x<<","<<"w="<<w<<endl;

bool xIsEndSymbol = 0;

for(int i = 0; i < sizeof(endSymbol);i++)

{

if (x == endSymbol[i])

{

xIsEndSymbol = 1;

if(endIsRight(x,w))

{

if(w>200 && w<400)

{

symbol.push(w);

//cout<<"symbol.top:"<<symbol.top()<<endl;

}

else if(w==401||w==402)

wm.push(w);

tokenNumNow++;

break;

}

else {flag = 0;break;}

}

}

if(xIsEndSymbol == 0)

{

int orderhere = lookupTable(x,w);

if(orderhere == 0){flag = 0; break;}

else{reverseStack(orderhere);}

}

}

if(flag == 1 && x == '#' && w == 405)

cout<<"Successful!"<<endl;

else

cout<<"Error!"<<endl;

return 0;

}

int lookupTable(char x, int w)

{

int row, column;

switch(x)

{

case 'E':{row = 0;break;}

case 'R':{row = 1;break;}

case 'T':{row = 2;break;}

case 'Y':{row = 3;break;}

case 'F':{row = 4;break;}

default:{cout<<"ERROR::at LookupTable();"<<endl;break;}

}

switch(w)

{

case 401:{column = 1;break;}

case 402:{column = 2;break;}

case 403:{column = 3;break;}

case 404:{column = 4;break;}

case 405:{column = 5;break;}

default:{column = 0;break;}

}

//cout<<"choose:"<<analysisTable[row][column]<<endl;

return analysisTable[row][column];

}

void reverseStack(int order)

{

switch(order)

{

case 1:{for(int i = sizeof(E)-1; i>=0;s.push(E[i]), i--);break;}

case 2:{for(int i = sizeof(R)-1; i>=0;s.push(R[i]), i--);break;}

case 4:{for(int i = sizeof(T)-1; i>=0;s.push(T[i]), i--);break;}

case 5:{for(int i = sizeof(Y)-1; i>=0;s.push(Y[i]), i--);break;}

case 8:{for(int i = sizeof(F)-1; i>=0;s.push(F[i]), i--);break;}

case 3:{break;}

case 6:{break;}

case 7:{s.push('I');break;}

default:{cout<<"ERROR!"<<endl;break;}

}

}

bool endIsRight(char x, int w)

{

switch(x)

{

case 'I':{if(w >= 200 && w < 399) return 1;else return 0;break;}

case 'W':{if(w == 401) return 1;else return 0;break;}

case 'M':{if(w == 402) return 1;else return 0;break;}

case '(':{if(w == 403) return 1;else return 0;break;}

case ')':

{

if(w == 404)

{

//插入语义动作

cout<<"("<<wm.top()<<" "<<symbol.top();

wm.pop();

symbol.pop();

cout<<" "<<symbol.top()<<" "<<temp<<")"<<endl;

temp++;

return 1;

}

else

return 0;

break;

}

case '#':{if(w == 405) return 1;else return 0;break;}

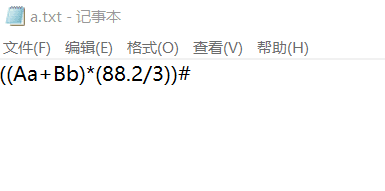
default:{cout<<"ERROR::at endIsRight();"<<endl;return 0;break;}

}

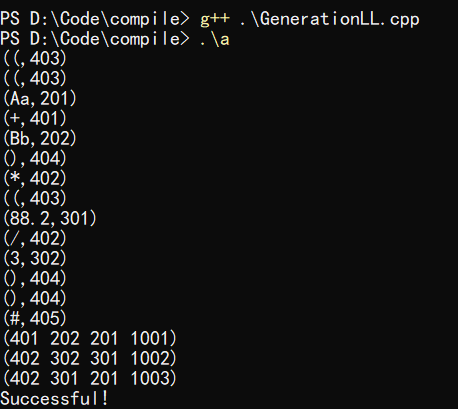
}

程序运行结果：（截屏）

输入：



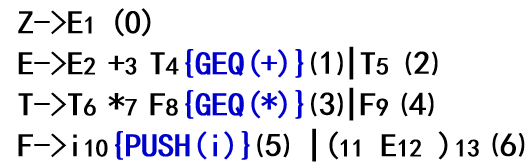
输出：



### **3.3 LR(0)方法**

实验内容：

翻译文法如下所示：



源程序代码：（加入注释）

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FileName:GenerationLR0.cpp

Function: LR(0) method is used for Quaternary grammar-guided translation.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <stack>

#include "WordFilter1.h"

using namespace std;

int analysisTable[12][9] =

{

// I W0 W1 ( ) # E T F

105,400,400,104,400,400, 1, 2, 3,

400,106,400,400,400,300,400,400,400,

400,202,107,400,202,202,400,400,400,

400,204,204,400,204,204,400,400,400,

105,400,400,104,400,400, 8, 2, 3,

400,206,206,400,206,206,400,400,400,

105,400,400,104,400,400,400, 9, 3,

105,400,400,104,400,400,400,400, 10,

400,106,400,400,111,400,400,400,400,

400,201,107,400,201,201,400,400,400,

400,203,203,400,203,203,400,400,400,

400,205,205,400,205,205,400,400,400

};

int getPlusCode(int x, int w);

int getReduceCode(int x, char c);

char getSymbol(int w);

void reduceSymbolStack(int order);

int tokens[1024], tokenNumNow = 0, flag = 2, temp = 1001;

stack<int> statusStack, symbol, wm;

stack<char> symbolStack;

int main()

{

wordFilter(tokens);

statusStack.push(0);

symbolStack.push('#');

int x, w, code1, code2;

char c;

while(flag!=0 && flag!=1)

{

x = statusStack.top();

w = tokens[tokenNumNow];

c = symbolStack.top();

code1 = getPlusCode(x,w);

//cout<<"code1="<<code1<<endl;

if(code1 == 400)

flag = 0;

else if(code1 == 300)

flag = 1;

else if(code1>100 && code1<200)

{

//移进

statusStack.push(code1 - 100);

symbolStack.push(getSymbol(w));

if(w>200 && w<400)

symbol.push(w);

else if(w==401||w==402)

wm.push(w);

tokenNumNow++;

}

else if(code1>200 && code1<300)

{

//归约

reduceSymbolStack(code1 - 200);

//四元式输出

if(code1-200==1||code1-200==3)

{

cout<<"("<<wm.top()<<" "<<symbol.top();

wm.pop();

symbol.pop();

cout<<" "<<symbol.top()<<" "<<temp<<")"<<endl;

temp++;

}

c = symbolStack.top();

x = statusStack.top();

code2 = getReduceCode(x,c);

cout<<"status:"<<code2<<endl;

if(code2 > 10 || code2 < 0)

flag = 0;

else

{

statusStack.push(code2);

}

}

else

{

flag = 0;

}

}

if(flag == 1)

cout<<"Successful"<<endl;

else

cout<<"Error"<<endl;

return 0;

}

int getPlusCode(int x, int w)

{

int row, column;

row = x;

switch(w)

{

case 401:{column = 1;break;}

case 402:{column = 2;break;}

case 403:{column = 3;break;}

case 404:{column = 4;break;}

case 405:{column = 5;break;}

default:{column = 0;break;}

}

return analysisTable[row][column];

}

int getReduceCode(int x, char c)

{

int row, column;

row = x;

switch(c)

{

case 'E':{column = 6;break;}

case 'T':{column = 7;break;}

case 'F':{column = 8;break;}

default:{cout<<"ERROR::at getReduceCode();"<<endl;flag = 0;break;}

}

return analysisTable[row][column];

}

char getSymbol(int w)

{

char res;

switch(w)

{

case 401:{res = 'w';break;}

case 402:{res = 'm';break;}

case 403:{res = '(';break;}

case 404:{res = ')';break;}

case 405:{res = '#';break;}

default:{res = 'i';break;}

}

return res;

}

void reduceSymbolStack(int order)

{

switch(order)

{

case 1:{symbolStack.pop();symbolStack.pop();statusStack.pop();statusStack.pop();statusStack.pop();break;}

case 2:{symbolStack.pop();symbolStack.push('E');statusStack.pop();break;}

case 3:{symbolStack.pop();symbolStack.pop();statusStack.pop();statusStack.pop();statusStack.pop();break;}

case 4:{symbolStack.pop();symbolStack.push('T');statusStack.pop();break;}

case 5:{symbolStack.pop();symbolStack.pop();symbolStack.pop();symbolStack.push('F');statusStack.pop();statusStack.pop();statusStack.pop();break;}

case 6:{symbolStack.pop();symbolStack.push('F');statusStack.pop();break;}

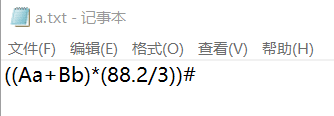
default:{cout<<"ERROR::at reduceSymbolStack()"<<endl;flag = 0;break;}

}

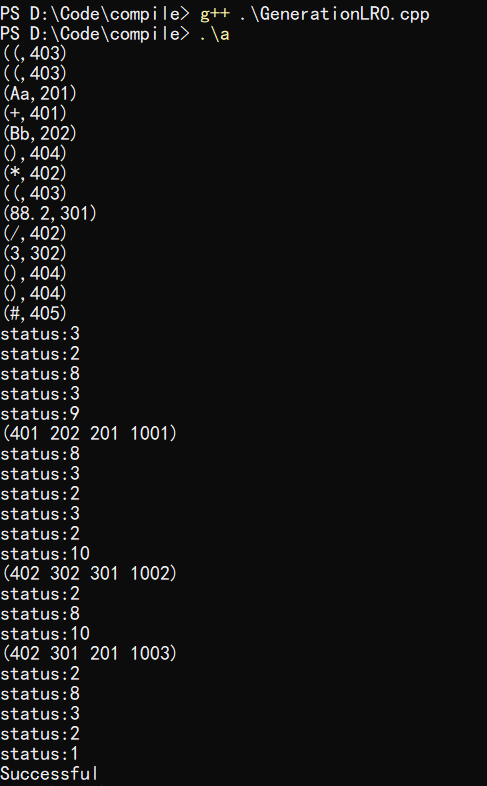
}

程序运行结果：（截屏）

输入：



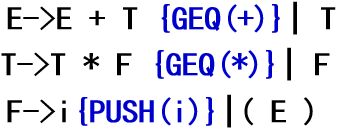
输出：



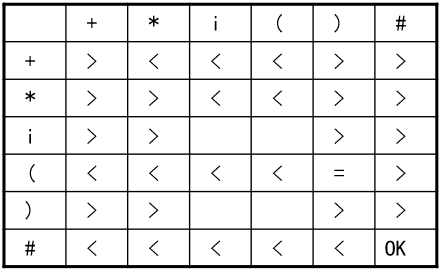
### **3.4 算符优先方法**

实验内容：

翻译文法：



算符优先分析表：



源程序代码：（加入注释）

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FileName:GenerationSF.cpp

Function: Quaternion grammar-guided translation using operator-first method.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <stack>

#include "WordFilter1.h"

using namespace std;

int relationTable[6][6] =

{

// +,\*,i,(,),#

2,3,3,3,2,0,

2,2,3,3,2,0,

2,2,0,0,2,2,

3,3,3,3,4,2,

2,2,0,0,2,2,

0,0,3,3,3,1

};//Error:0,OK:1,>:2,<:3,=:4

int getRelation(char x, int w);

char getSymbol(int w);

int tokens[1024], tokenNumNow = 0;

stack<char> symbolStack;

stack<int> symbol, wm;

int flag = 2;

int temp = 1001;

int main()

{

char x;

int w;

wordFilter(tokens);

symbolStack.push('#');

x = symbolStack.top();

w = tokens[tokenNumNow];

while(flag!=0 && flag!=1)

{

x = symbolStack.top();

w = tokens[tokenNumNow];

if(x!='V')

{

int relation = getRelation(x,w);

cout<<"x="<<x<<",w="<<w<<",r="<<relation<<endl;

if(relation==3 || relation==4) //移进

{

symbolStack.push(getSymbol(w));

if(w>200 && w<400)

symbol.push(w);

else if(w==401||w==402)

wm.push(w);

tokenNumNow++;

}

else if(relation==0 || relation==1)

{

flag = relation;

}

else if(relation == 2) //归约

{

switch(x)

{

case 'i':{symbolStack.pop();symbolStack.push('V');break;}

case ')':{symbolStack.pop();symbolStack.pop();symbolStack.pop();symbolStack.push('V');break;}

default:{flag = 0;cout<<"War::at Guiyue;"<<endl;break;}

}

}

}

else

{

symbolStack.pop();

x = symbolStack.top();

if(x == 'V') {flag = 0;cout<<"x2=V"<<endl;break;}

int relation = getRelation(x,w);

if(relation==3 || relation==4) //移进

{

symbolStack.push('V');

symbolStack.push(getSymbol(w));

if(w>200 && w<400)

symbol.push(w);

else if(w==401||w==402)

wm.push(w);

tokenNumNow++;

}

else if(relation==0 || relation==1)

{

flag = relation;

}

else if(relation == 2) //归约

{

if(x=='w' ||x=='m')

{

symbolStack.pop();

cout<<"("<<wm.top()<<" "<<symbol.top();

wm.pop();

symbol.pop();

cout<<" "<<symbol.top()<<" "<<temp<<")"<<endl;

temp++;

}

else

{

cout<<"?V,?!=+-\*/"<<endl;

flag = 0;

}

}

}

}

x = symbolStack.top();

if(flag == 1 && w == 405 && x == '#')

cout<<"Successful!"<<endl;

else

cout<<"Error!"<<endl;

return 0;

}

int getRelation(char x, int w)

{

int row, column;

switch(x)

{

case 'w':{row = 0;break;}

case 'm':{row = 1;break;}

case 'i':{row = 2;break;}

case '(':{row = 3;break;}

case ')':{row = 4;break;}

case '#':{row = 5;break;}

default:{cout<<"Err::at getRelation();"<<endl;break;}

}

switch(w)

{

case 401:{column = 0;break;}

case 402:{column = 1;break;}

case 403:{column = 3;break;}

case 404:{column = 4;break;}

case 405:{column = 5;break;}

default:{column = 2;break;}

}

return relationTable[row][column];

}

void intoStack(int relation)

{

switch(relation)

{

case 3:{}

}

}

char getSymbol(int w)

{

char symbol;

switch(w)

{

case 401:{symbol = 'w';break;}

case 402:{symbol = 'm';break;}

case 403:{symbol = '(';break;}

case 404:{symbol = ')';break;}

case 405:{symbol = '#';break;}

default:{symbol = 'i';break;}

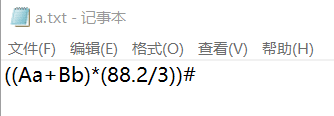
}

return symbol;

}

程序运行结果：（截屏）

输入：



输出：

