# 编译原理程序设计实验报告

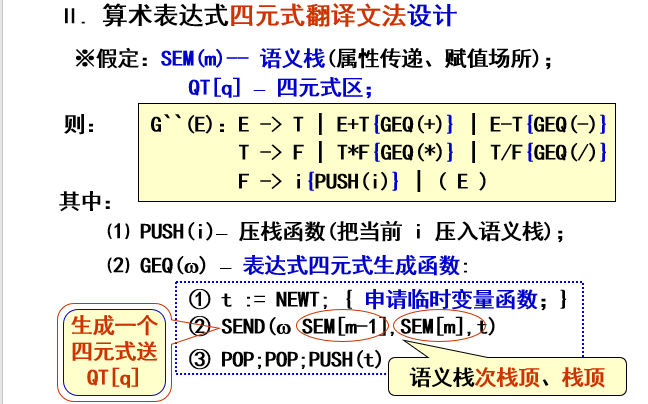
## ——算术表达式四元式翻译器的设计实现

班级：计算机1605班 姓名：张亚强 学号：20164382

实验目标：

用自顶向下LL（1）方法设计实现一个算术表达式四元式翻译器。

实验内容：

1. 概要设计：
2. 数据结构：

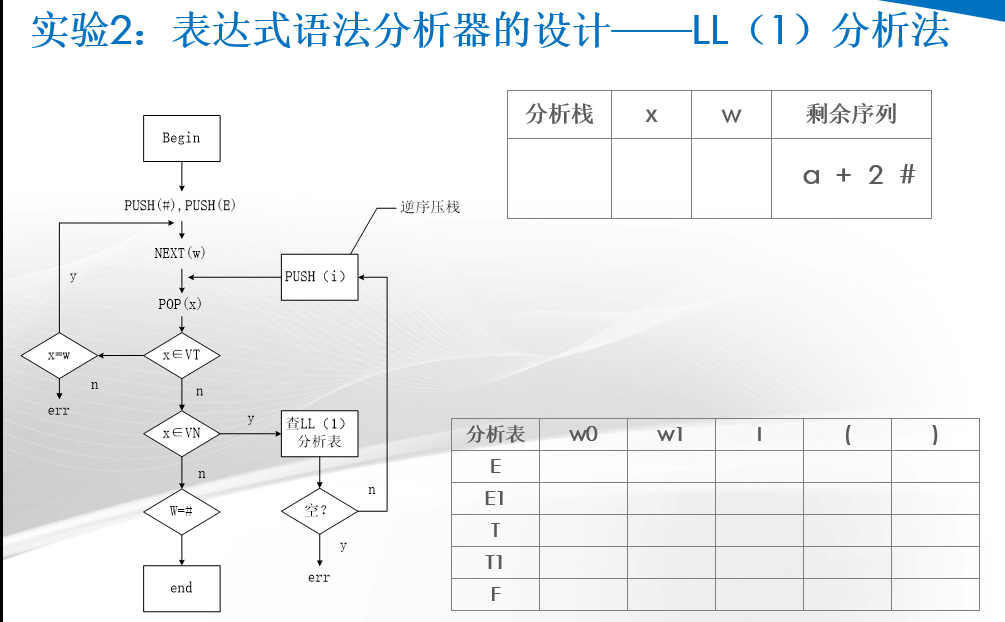
将输入串转换后的TOKEN串选择使用vector< pair<string,char>>表示，LL（1）分析过程中的栈结构选择用stack<string>表示，而四元式则用自定义的结构体表示，最终的四元式结果选择用vector<Qt>表示。

struct Qt //定义四元式结构体

{

string op,a,b,c;

};

1. 流程图：
2. 关键函数：

void To\_String**()** //将输入字符串转换为标准的TOKEN串

**{**

int i**=**0**;**

**while(**i**<(**int**)**input\_str**.**size**())**

**{**

**if(**input\_str**[**i**]** **==** ' '**)**i**++;**

**else** **if(** **(**input\_str**[**i**]>=**'a'**&&**input\_str**[**i**]<=**'z'**)** **||** **(**input\_str**[**i**]>=**'A'**&&**input\_str**[**i**]<=**'Z'**)** **||** **(**input\_str**[**i**]>=**'0'**&&**input\_str**[**i**]<=**'9'**)** **)** //当前是字符或者数字

**{**

string tmp**;**

**for(;**i**<(**int**)**input\_str**.**size**();**i**++)**

**{**

**if(!((**input\_str**[**i**]>=**'a'**&&**input\_str**[**i**]<=**'z'**)** **||** **(**input\_str**[**i**]>=**'A'**&&**input\_str**[**i**]<=**'Z'**)** **||** **(**input\_str**[**i**]>=**'0'**&&**input\_str**[**i**]<=**'9'**)** **||** **(**input\_str**[**i**]==**'.'**)))break;**

**else**

tmp**.**push\_back**(**input\_str**[**i**]);**

**}**

str**.**push\_back**(**make\_pair**(**tmp**,**'I'**));**

**}**

**else** //其他情况，直接将字符放入

**{**

string tmp**;**

tmp**.**push\_back**(**input\_str**[**i**]);**

str**.**push\_back**(**make\_pair**(**tmp**,**input\_str**[**i**]));**

i**++;**

**}**

**}**

**}**

源程序代码：（加入注释）

#include<cstdio>

#include<vector>

#include<cstdlib>

#include<algorithm>

#include<string>

#include<stack>

#include<iostream>

#include<utility>

#define In() freopen("in.txt","r",stdin)

#define Out() freopen("out.txt","w",stdout)

**using** **namespace** std**;**

struct Qt //定义四元式结构体

**{**

string op**,**a**,**b**,**c**;**

**};**

string input\_str**;**

int i**=**0**,**num**=**1**;**

string ch**;**

bool flag**=**0**;** //flag=1表示当前算数表达式串无法识别

vector**<** pair**<**string**,**char**>** **>**str**;**

vector**<**Qt**>**ans**;**

stack**<**string**>**s**;**

void To\_String**()** //将输入字符串转换为标准的TOKEN串

**{**

int i**=**0**;**

**while(**i**<(**int**)**input\_str**.**size**())**

**{**

**if(**input\_str**[**i**]** **==** ' '**)**i**++;**

**else** **if(** **(**input\_str**[**i**]>=**'a'**&&**input\_str**[**i**]<=**'z'**)** **||** **(**input\_str**[**i**]>=**'A'**&&**input\_str**[**i**]<=**'Z'**)** **||** **(**input\_str**[**i**]>=**'0'**&&**input\_str**[**i**]<=**'9'**)** **)** //当前是字符或者数字

**{**

string tmp**;**

**for(;**i**<(**int**)**input\_str**.**size**();**i**++)**

**{**

**if(!((**input\_str**[**i**]>=**'a'**&&**input\_str**[**i**]<=**'z'**)** **||** **(**input\_str**[**i**]>=**'A'**&&**input\_str**[**i**]<=**'Z'**)** **||** **(**input\_str**[**i**]>=**'0'**&&**input\_str**[**i**]<=**'9'**)** **||** **(**input\_str**[**i**]==**'.'**)))break;**

**else**

tmp**.**push\_back**(**input\_str**[**i**]);**

**}**

str**.**push\_back**(**make\_pair**(**tmp**,**'I'**));**

**}**

**else** //其他情况，直接将字符放入

**{**

string tmp**;**

tmp**.**push\_back**(**input\_str**[**i**]);**

str**.**push\_back**(**make\_pair**(**tmp**,**input\_str**[**i**]));**

i**++;**

**}**

**}**

**}**

int main**()** //LL(1)方法

**{**

In**();**Out**();** //文件读写操作

input\_str**.**clear**();**

**while(**getline**(**cin**,**input\_str**))**

**{**

//stack<string>s;

stack**<**string**>**sem**;**

ans**.**clear**();**

str**.**clear**();**

i**=**0**;**flag**=**0**;**num**=**1**;**

cout **<<** input\_str **<<**endl**;**

To\_String**();**

//将输入的串转换为TOKEN串

s**.**push**(**"#"**);**

s**.**push**(**"E"**);** //初始化栈

i**=**0**;**

**while(!**s**.**empty**()** **&&** i**<(**int**)**str**.**size**())**

**{**

//print\_stack();

ch**=**s**.**top**();**

string tmp1**;**

tmp1**.**push\_back**(**str**[**i**].**second**);**

**if(**ch**==**"#"**)**

**{**

**if(**str**[**i**].**second**==**'#'**){**s**.**pop**();**flag**=**1**;break;}**

**else**

**break;**

**}**

**else** **if(**ch **==** tmp1**){**s**.**pop**();**i**++;}**

**else** **if(**ch**==**"E"**)** //当前栈顶元素为E

**{**

**if(**str**[**i**].**second**==**'I' **||** str**[**i**].**second**==**'('**){**s**.**pop**();**s**.**push**(**"A"**);**s**.**push**(**"T"**);}** //将1号产生式逆序压栈

**else**

**break;**

**}**

**else** **if(**ch**==**"A"**)** //当前栈顶元素为A

**{**

**if(**str**[**i**].**second**==**'+'**){**s**.**pop**();**s**.**push**(**"A"**);**s**.**push**(**"GEQ(+)"**);**s**.**push**(**"T"**);**s**.**push**(**"+"**);}** //将2号产生式逆序压栈

**else** **if(**str**[**i**].**second**==**'-'**){**s**.**pop**();**s**.**push**(**"A"**);**s**.**push**(**"GEQ(-)"**);**s**.**push**(**"T"**);**s**.**push**(**"-"**);}** //3号产生式

**else** **if(**str**[**i**].**second**==**')' **||** str**[**i**].**second**==**'#'**){**s**.**pop**();}** //4号产生式

**else**

**break;**

**}**

**else** **if(**ch**==**"T"**)**

**{**

**if(**str**[**i**].**second**==**'I' **||** str**[**i**].**second**==**'('**){**s**.**pop**();**s**.**push**(**"B"**);**s**.**push**(**"F"**);}** //5号产生式

**else**

**break;**

**}**

**else** **if(**ch**==**"B"**)**

**{**

**if(**str**[**i**].**second**==**'\*'**){**s**.**pop**();**s**.**push**(**"B"**);**s**.**push**(**"GEQ(\*)"**);**s**.**push**(**"F"**);**s**.**push**(**"\*"**);}** //6号产生式

**else** **if(**str**[**i**].**second**==**'/'**){**s**.**pop**();**s**.**push**(**"B"**);**s**.**push**(**"GEQ(/)"**);**s**.**push**(**"F"**);**s**.**push**(**"/"**);}** //7号产生式

**else** **if(**str**[**i**].**second**==**'+' **||** str**[**i**].**second**==**'-' **||** str**[**i**].**second**==**')' **||** str**[**i**].**second**==**'#'**){**s**.**pop**();}** //8号产生式

**else**

**break;**

**}**

**else** **if(**ch**==**"F"**)**

**{**

**if(**str**[**i**].**second**==**'I'**)**

**{**

string tmp **=** "PUSH-"**+**str**[**i**].**first**;**

s**.**pop**();**s**.**push**(**tmp**);**s**.**push**(**"I"**);**

**}** //9号产生式

**else** **if(**str**[**i**].**second**==**'('**){**s**.**pop**();**s**.**push**(**")"**);**s**.**push**(**"E"**);**s**.**push**(**"("**);}** //10号产生式

**else**

**break;**

**}**

**else** **if(**ch**[**0**]==**'P'**)** //PUSH操作

**{**

string tmp**;**

**for(**int k**=**5**;**k**<(**int**)**ch**.**size**();**k**++){**tmp**.**push\_back**(**ch**[**k**]);}**

sem**.**push**(**tmp**);**

s**.**pop**();**

**}**

**else** **if(**ch**[**0**]==**'G'**)** //生成四元式操作

**{**

string tmp**;**

tmp**.**push\_back**(**'t'**);**

tmp**.**push\_back**(**48**+**num**);**

//cout << tmp <<endl;

string res1**=**sem**.**top**();**

sem**.**pop**();**

string res2**=**sem**.**top**();**

sem**.**pop**();**

Qt qt**;**

qt**.**op **=** ch**[**4**];**

qt**.**a **=** res2**;**

qt**.**b **=** res1**;**

qt**.**c **=** tmp**;**

ans**.**push\_back**(**qt**);**

sem**.**push**(**tmp**);**

num**++;**

s**.**pop**();**

**}**

**}**

**if(**flag **==**1**)**

**{**

printf**(**"Yes\n"**);**

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

**{**

cout **<<** '('**;**

cout **<<** ans**[**k**].**op**;**

cout **<<** ','**;**

cout **<<** ans**[**k**].**a**;**

cout **<<** ','**;**

cout **<<** ans**[**k**].**b**;**

cout **<<** ','**;**

cout **<<** ans**[**k**].**c**;**

cout **<<** ')'**<<**endl**;**

**}**

**}**

**else**

printf**(**"No\n"**);**

**}**

**return** 0**;**

**}**

/\*

E -> T E1

E1 -> w0 T E1 | ϵ

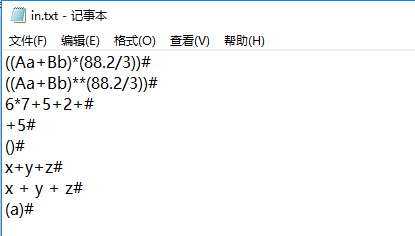
T -> F T1

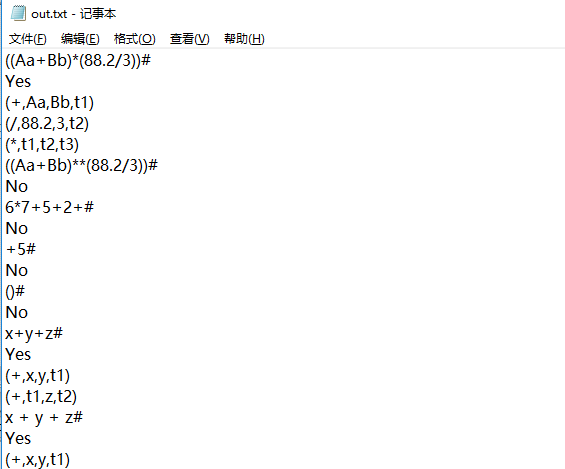
T1 -> w1 F T1 | ϵ

F -> I | (E)

\*/

程序运行结果：（截屏）

输入：

输出：