数据结构实验报告

**【算法说明】**

用到的数据结构：string类 stack堆栈

一、中缀表达式转后缀表达式：

1. 使用for循环遍历字符串s，并将s【i】转换成数字进行判断，并判断输入的字符串是否合法（是否有范围外的字符出现、左右括号是否匹配、运算符是否重复如1++2+3）。
2. 用字符串str储存数字，将运算符压栈到StoreOperator中，并在遇到运算符时在数字后加上空格。
3. 用函数priority（）判断运算符的优先级，如果前一个运算符优先级高于该运算符或者遇到右括号，则将栈中存储的运算符到第一个左括号为止全部弹出；当数字是最后一位时，输出数字并将栈中的运算符全部弹出。
4. 将弹出的运算符加到str中，得到后缀表达式。

中缀表达式转后缀表达式步骤图 1

二、后缀表达式的计算

1. 使用for循环遍历字符串str，用stemp来暂时储存数字。

2. 遇到空格将stemp中的字符串转成数字并压栈到StoreNumber中，并清空stemp。

3. 遇到运算符将StoreNumber中顶端的两个数字弹出，进行计算后将得数压入StoreNumber，在计算过程中判断运算是否合法。

4. 循环至字符串结束，将得数输入到文件”output.txt”中。

后缀表达式计算步骤示意图 １

**【测试结果】**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测 试 用 例 表 1** | | | | | |
|  | 用例1 | 用例2 | 用例3 | 用例4 | 用例5 |
| 测试输入 | （11.1+0.9）^2+1 | 1+2\*（4-3） | （5-1）--3 | 12---3 | 1+2+3+ |
| 测试目的 | 多位数、小数运算 | 优先级运算 | 负数运算 | 负数运算符重复 | 最后一位非数字无法计算 |
| 正确输出 | 145 | 3 | 7 | ERROR IN INFIX NOTATION | ERROR IN INFIX NOTATION |
| 实际输出 | 145 | 3 | 7 | ERROR IN INFIX NOTATION | ERROR IN INFIX NOTATION |
| 错误原因 | 无 | 无 | 无 | 无 | 无 |
| 当前状态 | 良好 | 良好 | 良好 | 良好 | 良好 |
| **测 试 用 例 表 2** | | | | | |
|  | 用例6 | 用例7 | 用例8 | 用例9 | 用例10 |
| 测试输入 | cos5+1 | 12.345 | 1++2\*3 | )(1+3(4+2) | 12/(3-3) |
| 测试目的 | 非法字符 | 纯数字 | 运算符非法 | 左右括号不匹配 | 运算非法 |
| 正确输出 | ERROR IN INFIX NOTATION | ERROR IN INFIX NOTATION | ERROR IN INFIX NOTATION | ERROR IN INFIX NOTATION | ERROR IN INFIX NOTATION |
| 实际输出 | ERROR IN INFIX NOTATION | ERROR IN INFIX NOTATION | ERROR IN INFIX NOTATION | ERROR IN INFIX NOTATION | ERROR IN INFIX NOTATION |
| 错误原因 | 无 | 无 | 无 | 无 | 无 |
| 当前状态 | 良好 | 良好 | 良好 | 良好 | 良好 |

**【分析与报告】**

一、测试结果分析

1．均运行良好没有问题，但是计算非常大的数字时会出现错误如111111111111111111111+111111111111会得出如2.22222e+16这样的结果。

2. 加减乘除、取余、乘方、带括号运算得数均正常，但是不能计算函数表达式。

3. 为便于负数计算，本例中将减去一个数换成了加上一个负数，并且将减去负数换成了加号，也就是说在后缀表达式的计算过程中并不会真正使用减法。

4. 在中缀表达式转后缀表达式过程中，因为多位数和负数计算，在后缀表达式输出上并不标准，若需标准后缀表达式，可以另加一段代码去掉空格。

二、探讨更多解决问题的途径

1. 可以尝试使用指向字符的指针来遍历字符串，但是好像和for循环差别不大。

2．可以将所有多位运算简化成a+b\*c 也就是把所有带加减运算的看作是一个整体 在按下一个“+”或“-”后，前面部分就成为一个整体，遍历s，用operator和lastoperator储存两个运算符，用op储存多位连乘或连除前的符号然后判断后一个字符的优先级是否高于前一个字符，不高于就直接计算，这样不需要堆栈。

【附录】

#include<iostream>

#include<fstream>

#include<string>

#include<cstdlib>

#include<stack>

#include<math.h>

using namespace std;

//优先级函数

int priority(int temp)

{

int priority;

if (temp == 37 || temp == 42 || temp == 47 || temp == 94)

priority = 2;

else

{

switch (temp) {

case 40:

priority = 0;

break;

case 41:

priority = 3;

break;

case 43:

priority = 1;

break;

case 45:

priority = 1;

break;

}

}

return priority;

}

int main()

{

stack<char> storeOperator;

stack<float> storeNumber;

string s; //原始字符串

string str; //储存字符串

string stemp;//储存两位以上数字

//读写文件

ifstream infile;

ofstream outfile;

infile.open("input.txt");

outfile.open("output.txt", ofstream::out);

//打不开报错

if (!infile)

{

cerr << "unable to open";

}

if (!outfile)

{

cerr << "unable to write";

}

else

{

//逐行读入并处理

while (!infile.eof())

{

getline(infile, s);

int length = s.length();

int count = 0;//存入栈中的操作符数

int lasttemp = 43;//将其初始值预存为除了“（”以外优先级最低的

int test = 0;//用于判断括号是否成对

int repeat = 0;//用于避免运算符重复导致的无效运算

int isrp = 0;//判断前一位是不是右括号

int minus = 0;//计算负数，将减号看作是加上一个负数

bool done = 0;//用于判断字符串是否是纯数字，并且在外部设置使其只执行一次

str = "";//初始化str

//后缀转中缀部分

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

{

int temp = (int)s[i];//用temp判断这是哪个操作符或者数字

//存入数字(将小数点看作与数字等效)

if (temp >= 48 && temp <= 57 || temp == 46)

{

if (minus == 2)

{

str = str + "-";

minus = 0;

}

//每个数字与它之前运算符相配

if (i == 0 || (i == 1 && lasttemp == 40))

repeat = 0;

else

repeat--;

//若数字在最末一位将栈里储存的运算符全部弹出

if (i == length - 1)

{

if (test != 0 || temp == 46)

str = "error";

else

{

str = str + s[i];

str = str + " ";

while (count != 0)

{

str = str + storeOperator.top();

storeOperator.pop();

count--;

}

}

}

//将数字存储在str中

else

str = str + s[i];

}

//存入操作符（入栈、出栈）

else if (temp == 37 || temp == 45 || temp == 47 || temp == 94 || (temp >= 40 && temp <= 43))

{

int plt = priority(lasttemp);

int pt = priority(temp);

if (temp == 45)

{

if (repeat != 0 || i == 0)

{

if (minus != 2)

str = str + "-";

minus = 1;

}

else

{

minus = 2;

temp = 43;

}

}

//str = str + " ";//十位计算

//检测运算是否合法（运算符重复以及最后一位不是数字）

if (temp == 37 || temp == 42 || temp == 43 || temp == 47 || temp == 94)

{

if (isrp == 0)//如果前面有右括号遇到运算符就不空格了

{

str = str + " ";

if ((i == 0 && done == 0) || (i == 1 && lasttemp == 40))

repeat = 0;

else

repeat--;

}

else

isrp = 0;

repeat++;

if (done==0)

done = 1;

}

if (repeat >1 || (i == length - 1 && pt != 3))//运算符在最后一位或者运算符重复

{

str = "error";

break;

}

//出栈

if (((pt < plt) && (plt != 3)) && (pt != 0))

{

//str = str + " ";

while (count != 0)

{

str = str + storeOperator.top();

storeOperator.pop();

count--;

}

storeOperator.push((char)temp);

count++;

}

//出现右括号

else if (pt == 3)

{

test--;

if (test < 0)//若出现“)(”情况报错

{

//outfile << "ERROR IN INFIX NOTATION" << endl;

//goto end;

str = "error";

break;

}

str = str + " ";

isrp++;

if ( i == 1 && lasttemp == 40)

repeat = 0;

else

repeat--;

while ((int)storeOperator.top() != 40)

{

str = str + storeOperator.top();

storeOperator.pop();

count--;

}

storeOperator.pop();

count--;

//test--;

//当右括号是最后一位

if (i == length - 1)

{

if (test != 0)//检测左右括号个数是否匹配

str = "error";

else

{

while (count != 0)

{

str = str + storeOperator.top();

storeOperator.pop();

count--;

}

}

}

else

{

if (storeOperator.empty())

temp = 43;

else

temp = (int)storeOperator.top();

}

}

//将操作符压入栈

else if(temp!=45)

{

storeOperator.push((char)temp);

count++;

if (temp == 40)

test++;

}

lasttemp = temp;

}

//出现范围外符号报错

else

{

str = "error";

break;

}

}

if (done != 0)

str = "error";

//计算后缀表达式

int length2 = str.length();

float a; float b;//用于计算

if (str == "error")

{

outfile << "ERROR IN INFIX NOTATION"<<endl;

goto end;

}

for (int i = 0; i < length2; i++)

{

int temp = (int)str[i];

if (temp >= 48 && temp <= 57 || temp == 46||temp==45)

stemp = stemp + str[i];

//storeNumber.push(temp- 48);

else if (temp == 32)//读取过程中遇到空格则stemp整体表示一个数字

{

storeNumber.push(atof(stemp.c\_str()));

stemp = "";

}

else

{

//storeNumber.push(atoi(stemp.c\_str()));

a = storeNumber.top();

storeNumber.pop();

b = storeNumber.top();

storeNumber.pop();

switch (temp)

{

case 42:

a = a \* b;

break;

case 43:

a = a + b;

break;

//case 45:

//a = b - a;

//break;

case 94:

a = pow(b, a);

break;

case 37:

{

if (a == 0)

{

outfile << "ERROR IN INFIX NOTATION" << endl;

goto end;

}

int c; int d;//%操作只支持int型变量进行强制类型转换

c = (int)a; d = (int)b;

a = d%c;

}

break;

case 47:

if (a == 0)//运算非法

{

outfile << "ERROR IN INFIX NOTATION" << endl;

goto end;

}

a = b / a;

break;

}

storeNumber.push(a);

if (i == length2 - 1)

{

//outfile << str<<endl;

outfile << a << endl;

}

}

}

end:;

}

}

infile.close();

outfile.close();

}