项目说明文档

数据结构课程设计

——表达式转换

作 者 姓 名： 李佳诺

学 号： 1751188

指 导 教 师： 张 颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

# 目录

1 分析 2

1.1 背景分析 2

1.2 功能分析 3

2 设计 3

2.1 算法设计 3

2.2 数据结构设计 4

2.2 成员与操作设计 4

3 设计 5

3.1 转换函数change(string \_inString\_sp) 5

3.2 返回优先级函数generate\_priority() 9

3.3 调整格式函数trim() 9

4 测试 10

4.1 正常测试6种运算符 10

4.2 嵌套括号 10

4.3 运算数超过1位整数且有非整数出现 11

4.4 运算数有正或负号 11

4.5 只有1个数字 12

# 

# 1 分析

## 背景分析

在日常生活中，我们所遇到的数学表达式一般都是中缀表达式，中缀表达式是一种通用的算术或逻辑公式表示方法，操作符以中缀形式处于操作数的中间。中缀表达式是人们常用的算术表示方法。

虽然人的大脑很容易理解与分析中缀表达式，但对计算机来说中缀表达式却是很复杂的，因此计算表达式的值时，通常需要先将中缀表达式转换为前缀或后缀表达式，然后再进行求值。对计算机来说，计算前缀或后缀表达式的值非常简单。

所以，为了提升计算机性能，本系统通过中缀表达式转换成后缀表达式，以此来提升计算的性能。

## 功能分析

1. 输入说明：输入在一行中给出以空格分隔不同对象的中缀表达式，可包含+, -, \*, /, -, \*, /以及左右括号，表达式不超过20个字符（不包括空格）。
2. 输出说明：在一行中输出转换后的后缀表达式，要求不同对象（运算数，运算符号）之间以空格分隔，但是结尾不得有多余空格。

# 设计

## 算法设计

中缀表达式转换成后缀表达式的基本算法如下：

(1) 初始化两个栈：运算符栈S1和储存中间结果的栈S2；

(2) 从左至右扫描中缀表达式；

(3) 遇到操作数时，将其压入S2；

(4) 遇到运算符时，比较其与S1栈顶运算符的优先级：

(4-1) 如果S1为空，或栈顶运算符为左括号“(”，则直接将此运算符入栈；

(4-2) 否则，若优先级比栈顶运算符的高，也将运算符压入S1（注意转换为前缀表达式时是优先级较高或相同，而这里则不包括相同的情况）；

(4-3) 否则，将S1栈顶的运算符弹出并压入到S2中，再次转到(4-1)与S1中新的栈顶运算符相比较；

(5) 遇到括号时：

(5-1) 如果是左括号“(”，则直接压入S1；

(5-2) 如果是右括号“)”，则依次弹出S1栈顶的运算符，并压入S2，直到遇到左括号为止，此时将这一对括号丢弃；

(6) 重复步骤(2)至(5)，直到表达式的最右边；

(7) 将S1中剩余的运算符依次弹出并压入S2；

(8) 依次弹出S2中的元素并输出，结果的逆序即为中缀表达式对应的后缀表达式（转换为前缀表达式时不用逆序）。

## 2.2 数据结构设计

为了实现后进先出（LIFO）的特性，我们利用了栈这个存储结构来实现。

## 成员与操作设计

string resolve(string \_inString); // 用来将中缀表达式\_inString转化成相应的后缀表达式，返回值即为相应的后缀表达式

void generate\_priority(map<char, int> & \_prior\_list); // map结构，数字或者符号为key,其value即为所对应的优先级

bool is\_num(char c); //判断是否为数字

bool is\_min\_plus(char c); //判断是否是加号或者减号

bool is\_multi\_div(char c); //判断是否是乘号或者除号

string& trim(string &s); //调整格式

# 设计

## 转换函数change(string \_inString\_sp)

**转换函数流程图**

开始

退栈

输出

结束

当前ch为操作数？

表达式没检索完&&栈不为空？

判断ch的优先级和位于栈顶的op优先级

退栈

ch进栈

否

是

<

=

>

否

是

**代码如下：**

// 用来将中缀表达式\_inString转化成相应的后缀表达式，返回值即为相应的后缀表达式

string change(string \_inString\_sp)

{

//用于存放运算符号的栈

map<char, int> prior\_list;

generate\_priority(prior\_list);

stack<char> signals;

string \_inString(\_inString\_sp.length(), ' ');

int ck = 0;

for (int i = 0; i < \_inString\_sp.length(); i++) {

if (\_inString\_sp[i] == ' ') {

continue;

}

else {

\_inString[ck++] = \_inString\_sp[i];

}

}

string \_outString(2\*\_inString.length(),' ');

int index = 0;

for (int i = 0; i < \_inString.length(); i++) {

if (\_inString[i] == ' ') {

continue;

}

else if (is\_num(\_inString[i])||\_inString[i]=='.') {

\_outString[index++] = \_inString[i];

}

else if (is\_min\_plus(\_inString[i])) {

//处理正负号

if (i - 1 < 0 || (!is\_num(\_inString[i - 1])&&\_inString[i-1]!=')')) {

\_outString[index++] = \_inString[i];

}

//否则的话

else {

while (!signals.empty()

&& prior\_list[signals.top()] >= prior\_list[\_inString[i]]

&& signals.top() != '(') {

\_outString[index++] = ' ';

\_outString[index++] = signals.top();

signals.pop();

}

\_outString[index++] = ' ';

signals.push(\_inString[i]);

}

}

else if (is\_multi\_div(\_inString[i])) {

while (!signals.empty()

&& prior\_list[signals.top()] >= prior\_list[\_inString[i]]

&& signals.top()!='(') {

\_outString[index++] = ' ';

\_outString[index++] = signals.top();

signals.pop();

}

\_outString[index++] = ' ';

signals.push(\_inString[i]);

}

else if (\_inString[i] == '(') {

signals.push(\_inString[i]);

}

else if (\_inString[i] == ')') {

while (signals.top() != '(') {

\_outString[index++] = ' ';

\_outString[index++] = signals.top();

signals.pop();

}

signals.pop();

}

}

while (!signals.empty()) {

\_outString[index++] = ' ';

\_outString[index++] = signals.top();

signals.pop();

}

\_outString = trim(\_outString);

return \_outString;

}

## 返回优先级函数generate\_priority()

通过map类型来表示，其中key为操作符，value为其所对应的优先级。具体代码如下：

void generate\_priority(map<char, int> & \_prior\_list)

{

//数字优先级

for (int i = 0; i <= 9; i++)

{

\_prior\_list['0' + i] = 1;

}

//加减优先级

\_prior\_list['+'] = 2;

\_prior\_list['-'] = 2;

//乘除优先级

\_prior\_list['\*'] = 3;

\_prior\_list['/'] = 3;

//括号优先级

\_prior\_list['('] = 4;

\_prior\_list[')'] = 4;

}

## 调整格式函数trim()

本函数用来对所输入的表达式进行处理，使得处理后符合题目的格式要求，具体代码如下：

string& trim(string &s)

{

if (s.empty())

{

return s;

}

s.erase(0, s.find\_first\_not\_of(" "));

s.erase(s.find\_last\_not\_of(" ") + 1);

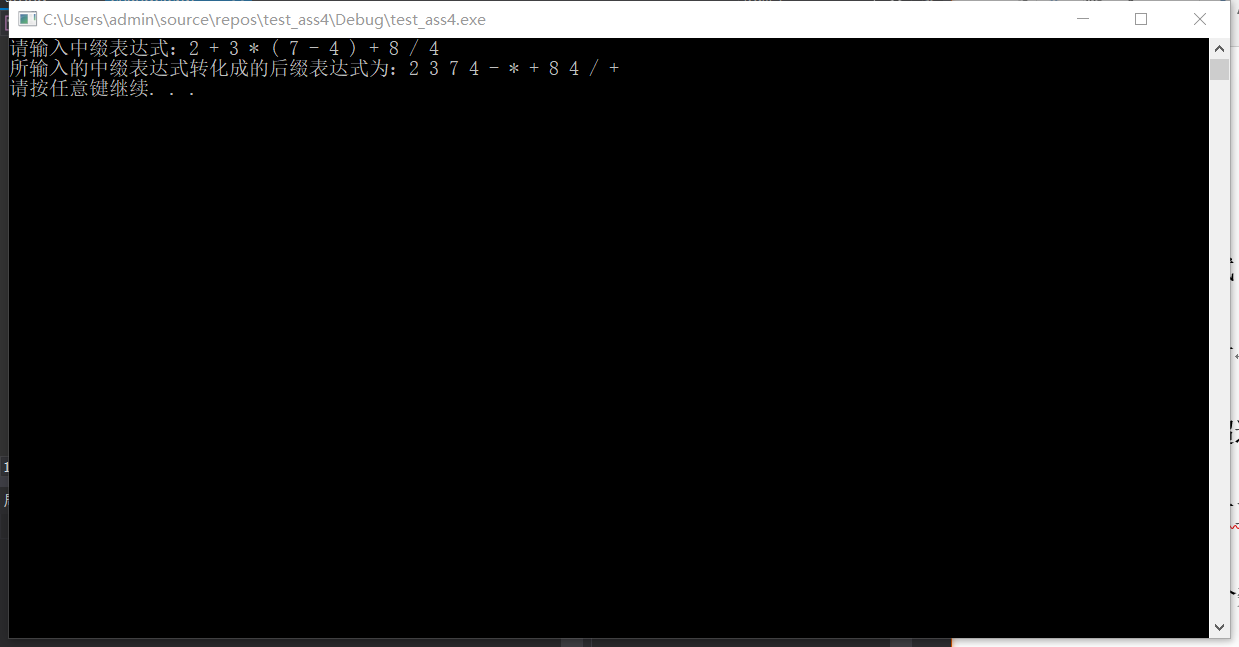
return s;

}

# 测试

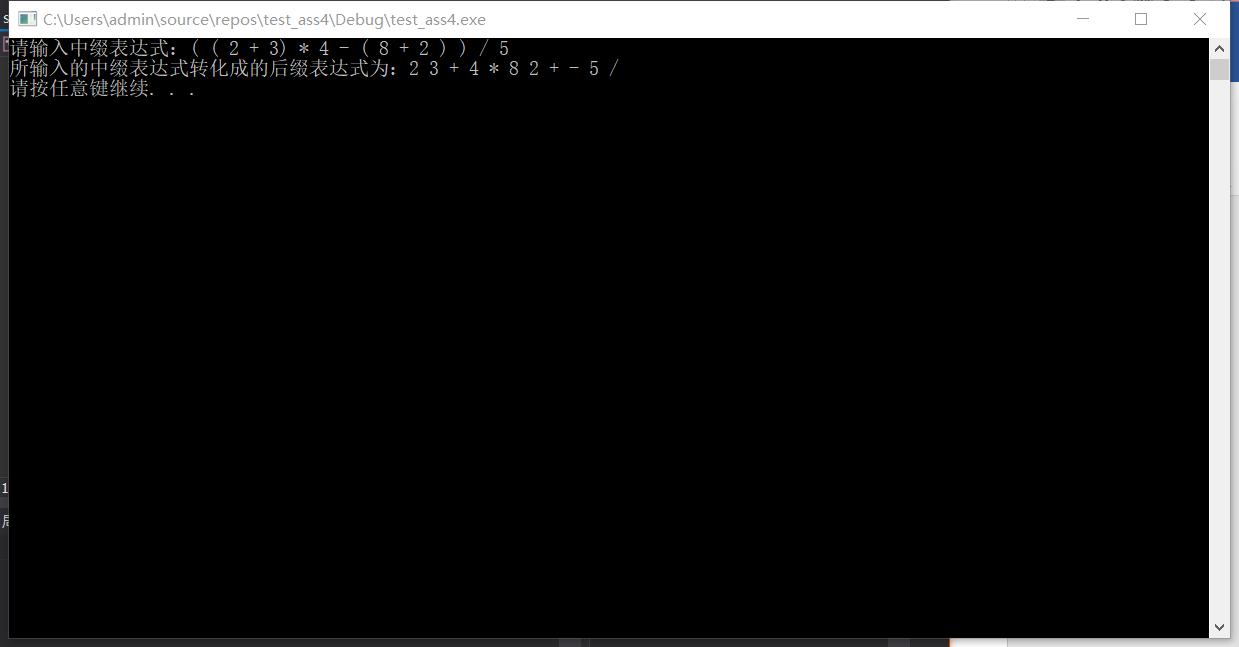
## 正常测试6种运算符

实验结果如下：



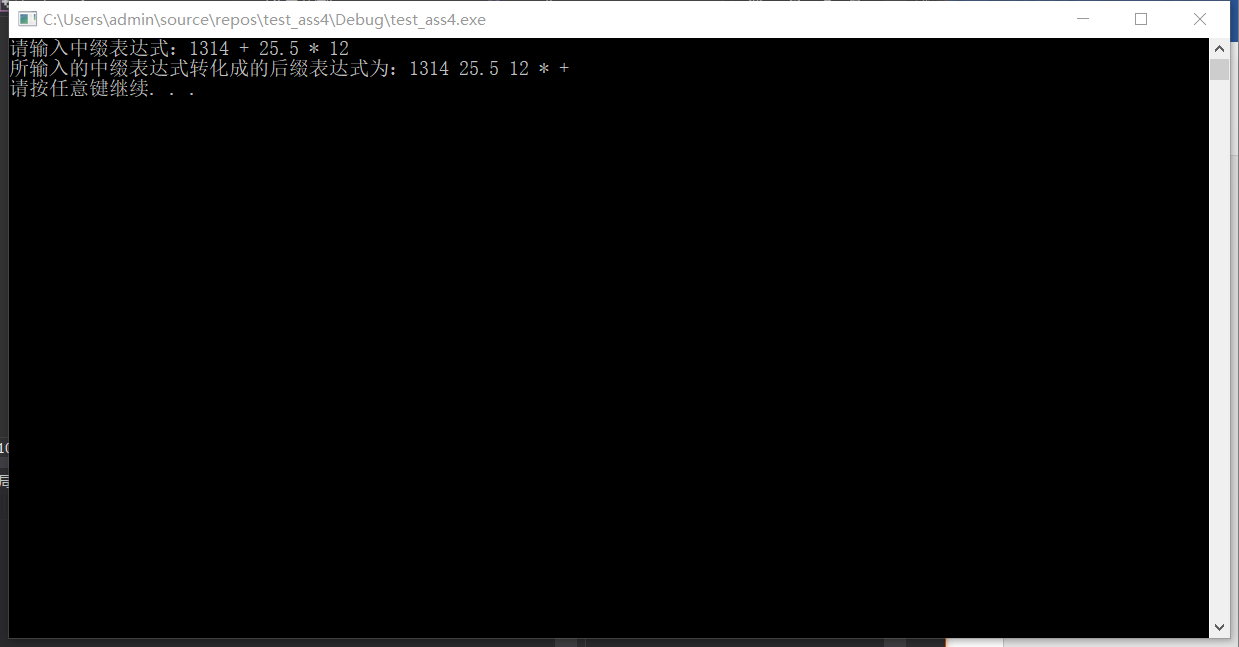
## 嵌套括号

实验结果如下：



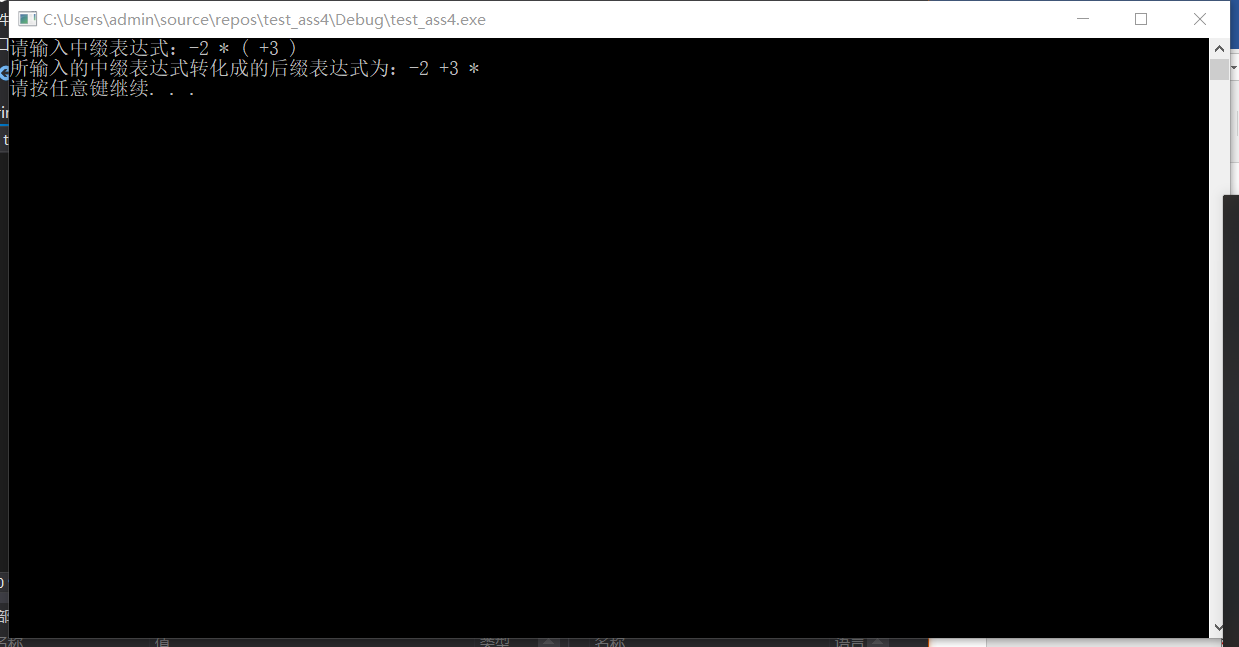
## 运算数超过1位整数且有非整数出现

实验结果如下：



## 运算数有正或负号

实验结果如下：



## 只有1个数字

实验结果如下：

