项目说明文档

数据结构课程设计

——表达式转换

作 者 姓 名： 王星洲

学 号： 1652977

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 1](#_Toc533186144)

[1.1 背景分析 1](#_Toc533186145)

[1.2 功能分析 1](#_Toc533186146)

[2 设计 1](#_Toc533186147)

[2.1 数据结构设计 1](#_Toc533186148)

[2.2 算法设计 1](#_Toc533186149)

[2.3 流程设计 2](#_Toc533186150)

[3 实现 3](#_Toc533186151)

[3.1 算法对于正负号的判断 3](#_Toc533186152)

[3.1.1主要代码 3](#_Toc533186153)

[3.2 算法对于数字的操作 4](#_Toc533186154)

[3.2.1 数字操作流程 4](#_Toc533186155)

[3.2.2 数字操作核心代码 5](#_Toc533186156)

[3.2.3 数字操作说明 5](#_Toc533186157)

[3.3 对括号和符号的处理 6](#_Toc533186158)

[3.3.1 符号处理说明 6](#_Toc533186159)

[3.3.2 符号处理流程 7](#_Toc533186160)

[3.3.3 符号处理注意事项 7](#_Toc533186161)

[4 测试 8](#_Toc533186162)

[4.1 功能测试 8](#_Toc533186163)

[4.1.1 正常功能测试 8](#_Toc533186164)

[4.1.2 嵌套括号测试 8](#_Toc533186165)

[4.2 边界测试 8](#_Toc533186166)

[4.2.1 运算数超过1位整数且有非整数出现 8](#_Toc533186167)

[4.2.2 运算数有正或负号 9](#_Toc533186168)

[4.2.3 只有1个数字 9](#_Toc533186169)

[4.3 错误测试 9](#_Toc533186170)

[5. 亮点 10](#_Toc533186171)

[5.1 数据结构的选取 10](#_Toc533186172)

[5.2 算法的设计 10](#_Toc533186173)

# 1 分析

## 1.1 背景分析

算数表达式有前缀表示法，中缀表示法和后缀表示法等形式。日常使用的算术表达式是采用中缀表示法，即二元运算符位于两个运算数中间。请设计程序将中缀表达式转换成为后缀表达式。

## 1.2 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。

# 2 设计

## 2.1 数据结构设计

本题经过两种遍历顺序的转换，由中序变为后序。而用户的输入是不确定的，我们采用string类型数据来存储，并按字符分析，而改变遍历顺序需要一个后进先出的链，因此我们考虑使用栈来进行操作，将运算符，左括号压入栈中，遇到右括号或者式子结束时再按倒序输出出来，也就实现了后缀表达式的功能。

## 2.2 算法设计

将中缀表达式转换为后缀表达式的算法思想：

·开始扫描；

·数字时，加入后缀表达式；

·运算符：

a. 若为 '('，入栈；

b. 若为 ')'，则依次把栈中的的运算符加入后缀表达式中，直到出现'('，从栈中删除'(' ；

c. 若为 除括号外的其他运算符， 当其优先级高于除'('以外的栈顶运算符时，直接入栈。否则从栈顶开始，依次弹出比当前处理的运算符优先级高和优先级相等的运算符，直到一个比它优先级低的或者遇到了一个左括号为止，然后将其自身压入栈中（先出后入）。

·当扫描的中缀表达式结束时，栈中的的所有运算符出栈。

## 2.3 流程设计

系统首先获取用户的输入，并储存在字符串中，之后通过算法分析字符串中的每个字符，进行有关栈的相应操作，操作中包含输出，最终得到完整的后缀表达式。

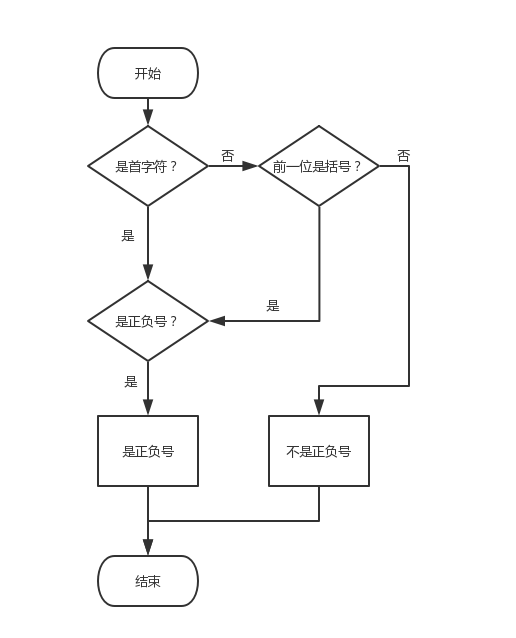
# 3 实现

## 3.1 算法对于正负号的判断

### 3.1.1主要代码

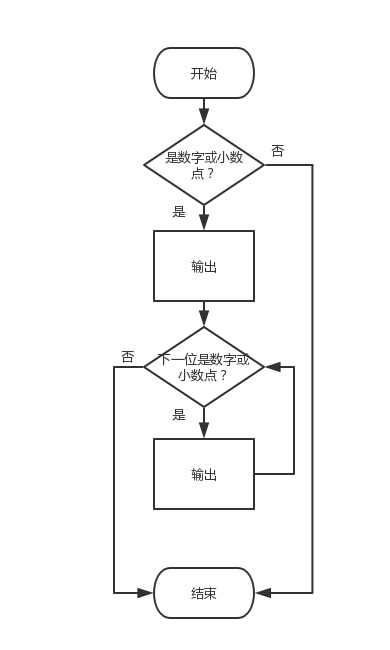
(i<1 ||a[i - 2] == '(' && i>=2 && a[i - 1] == ' ' ||a[i - 1] == '(') && (a[i] == '+' || a[i] == '-')

其中i为循环控制变量，即该字符的位置，i<1表示初始字符，也就是说正负号的判断判断过程如下：



## 3.2 算法对于数字的操作

### 3.2.1 数字操作流程



### 3.2.2 数字操作核心代码

else if ((i<1 || a[i - 1] == '(' || (i>=2 && a[i – 2] == '(' && a[i - 1] == ' ')) && (a[i] == '+' || a[i] == '-') || a[i] == '.' || a[i] >= '0'&&a[i] <= '9')

{

//当该字符是正负号或为小数点或为数字时

if (!firstPrint) //如果之前有过输出

{

cout << " "; //输出空格

}

firstPrint = false; //将首次输出置为假

if (a[i] != '+') //如果该符号不是正号

{

cout << a[i]; //将其输出

}

while (a[i + 1] == '.' || a[i + 1] >= '0'&&a[i + 1] <= '9') //如果该字符下一位是小数点或数字

{

i++; //计数增加

cout << a[i]; //输出a[i]

}

}

### 3.2.3 数字操作说明

由于是中缀表达式变为后缀表达式，我们选择对运算符进行调序操作，而对于数字，只要遇见数字就进行输出。然而数字有几个容易出错的点。

1.小数

2.正数负数

3.空格的判断

所以如何获得整个数字成为了关键，我采用的方法是，判断正负号（3.1说明），与加减号区别开，正号不需要输出，负号需要连同数字输出。对于正负号开头或正常以数字开头的数字，我们依次向后遍历，知道遇见不为小数点或数字的字符时停止。这样就输出了我们预期的操作数了。

## 3.3 对括号和符号的处理

### 3.3.1 符号处理说明

a. 若为 '('，入栈；

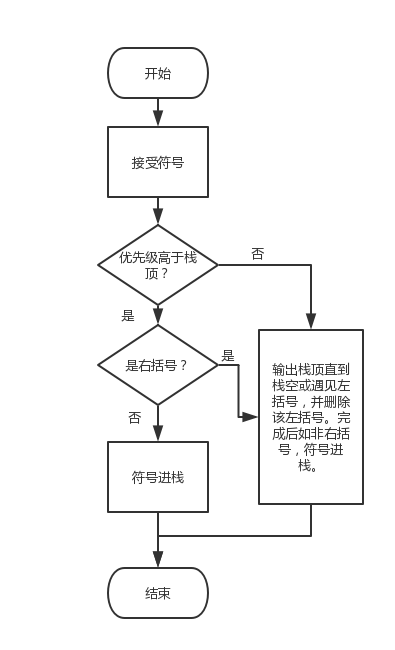
b. 若为 ')'，则依次把栈中的的运算符加入后缀表达式中，直到出现'('，从栈中删除'(' ；

c. 若为除括号外的其他运算符，当其优先级高于除'('以外的栈顶运算符时，直接入栈。否则从栈顶开始，依次弹出比当前处理的运算符优先级高和优先级相等的运算符，直到一个比它优先级低的或者遇到了一个左括号为止，然后将其自身压入栈中（先出后入）。

·当扫描的中缀表达式结束时，栈中的的所有运算符出栈。

注意：为了实现便捷，我把括号的优先级定义为2，\* /号定义为1，加减无优先级

### 3.3.2 符号处理流程



### 3.3.3 符号处理注意事项

1.右括号不需要进栈，当左括号前的符号出栈后，也要将左括号删除。

2.使用优先级可以处理更多符号的问题，更有扩展性。

# 4 测试

## 4.1 功能测试

### 4.1.1 正常功能测试

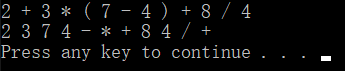
**测试用例**：

输入2 + 3 \* ( 7 – 4 ) + 8 / 4

**预期结果**：

输出 2 3 7 4 - \* + 8 4 / +

**实验结果**



### 4.1.2 嵌套括号测试

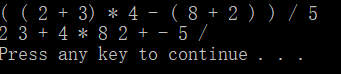
**测试用例：**

输入 ( ( 2 + 3) \* 4 – ( 8 + 2 ) ) / 5

**预期结果：**

输出 2 3 + 4 \* 8 2 + - 5 /

**实验结果：**



## 4.2 边界测试

### 4.2.1 运算数超过1位整数且有非整数出现

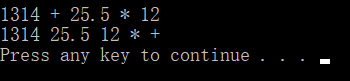
**测试用例：**

输入1314 + 25.5 \* 12

**预期结果：**

输出1314 25.5 \* +

**实验结果：**



### 4.2.2 运算数有正或负号

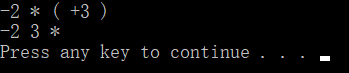
**测试用例：**

输入-2 \* ( +3 )

**预期结果：**

输出-2 3 \*

**实验结果：**



### 4.2.3 只有1个数字

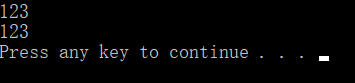
**测试用例：**

输入 123

**预期结果：**

输出 123

**实验结果：**



## 4.3 错误测试

本实验的代码中，我可以应对的错误是空格的有无，如上述4.1.1的题例中，我将空格全部删去后的测试。

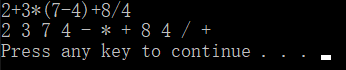
**测试用例：**

输入2+3\*(7–4)+8/4

**预期结果：**

输出 2 3 7 4 - \* + 8 4 / +

**实验结果：**



这样，程序就不再要求相邻元素间必须添加空格了，我的程序也无需根据空格判断符号是否为正负号。

# 5. 亮点

## 5.1 数据结构的选取

本实验中，我考查了许多数据结构，而决定使用栈作为主要存储结构，详见2.1

## 5.2 算法的设计

本次实验的算法设计很精妙，首先为了应对正负数和可有可无的空格，我使用了大量逻辑并与交实现对于正负号的判断。使得程序中无论空格有或没有都可以正常运行。见3.1。其次是算法对于符号及栈的操控，我采用优先级符号解决问题，增加了程序的可扩展性。同时对栈的使用也给予了说明，详见3.3。