## 数据结构课程设计之项目说明文档

# 算数表达式求解

A picture containing drawing

Description automatically generated

作 者 姓 名： 张佰一

学 号： 1952060

指 导 教 师： 张 颖

学院、 专业： 软件学院、软件工程

*本程序版本控制及远程仓库网址：[点击访问本项目Github仓库](https://github.com/zhangbaiyi/ARITH-EXP-CALC-P4-DS-FALL-2020)*

目录

[数据结构课程设计之项目说明文档 1](#_Toc58441364)

[算数表达式求解 1](#_Toc58441365)

[1分析 3](#_Toc58441366)

[1.1背景分析 3](#_Toc58441367)

[1.2功能分析 3](#_Toc58441368)

[2设计 3](#_Toc58441369)

[2.1设计概览 3](#_Toc58441370)

[2.2数据结构类： 3](#_Toc58441371)

[2.3 预处理函数类： 4](#_Toc58441372)

[2.4 calcRes()： 4](#_Toc58441373)

[2.5 main() 4](#_Toc58441374)

[3实现 4](#_Toc58441375)

[3.1 类和其他数据结构的定义 4](#_Toc58441376)

[3.2界面与功能选择的实现 6](#_Toc58441377)

[3.3 具体实现 6](#_Toc58441378)

[3.3.1 字符串输入函数expInput() 6](#_Toc58441379)

[3.3.2 预处理函数preProcess(string&exp, vector<ITEM>&result) 6](#_Toc58441380)

[3.3.3 中缀表达式转换为后缀表达式 14](#_Toc58441381)

[3.3.4 计算：逆波兰表达式 16](#_Toc58441382)

[4测试 18](#_Toc58441383)

[4.1 测试实例 18](#_Toc58441384)

# 1分析

## 1.1背景分析

计算一个算数表达式是人类在学习数学中的基本功。为了进行不同种类的运算，我们设计了许多不同的运算符。在这些运算符同时出现时，按照优先级功能决定了初级的运算步骤。但有时为了完成一些比较复杂的运算，往往需要先计算优先级比较低的运算符。这时候，我们就引入了括号帮助计算。但同时，也让线性读取数据的计算机更加头疼。为此，设计新的数据结构来帮助我们做到想做的事情就十分有意义。

## 1.2功能分析

题目要求输入中缀算术表达式，输出计算的结果。

这就需要我们来处理字符串中的符号和数字。计算机更希望能接收到后缀表达式。这样可以知道两个运算对象和它们之间做什么运算，在线性读取后可以得到结果。我们要做的就是将中缀表达式转化为这样的后缀表达式。出现优先级问题时，通过后缀表达式的运算符在后，我们也可以很方便地得到优先计算的对象为何。但在处理实际问题时，往往需要考虑更多问题。这将在后续的文档中进行介绍。

# 2设计

## 2.1设计概览

本程序的基本思想是通过类和类中的函数实现信息表中的功能，通过主函数和一些辅助函数来调用类函数。具体的实现主要分为三大类。第一类是栈的实现，第二类是字符串的处理，最后一类是用户友好界面。

## 2.2数据结构类：

栈类：arithStack<T>类。包括压栈出栈等等基本功能。

字符数据结构体：typedef struct ITEM。考虑到字符串很难对两位数进行高效判断和处理，我用一个item类来表示一个算术表达式中可能出现的项目。可能是数字也可能是符号。

## 2.3 预处理函数类：

preProcess()：将字符串中的字符转换为item，其中就包括对于负数和连续出现相同优先级的运算符情况等等的判断和处理。

infix\_to\_postfix()：将情况比较单一的中缀item串转化为后缀表达式。

## 2.4 calcRes()：

对后缀表达式进行以栈作为基础的求值操作。

## 2.5 main()

实现一个循环输入与计算的功能，实现一个输出计算输入规则的函数menu()，并且显示计算结果。

# 3实现

## 3.1 类和其他数据结构的定义

-> string oriInput

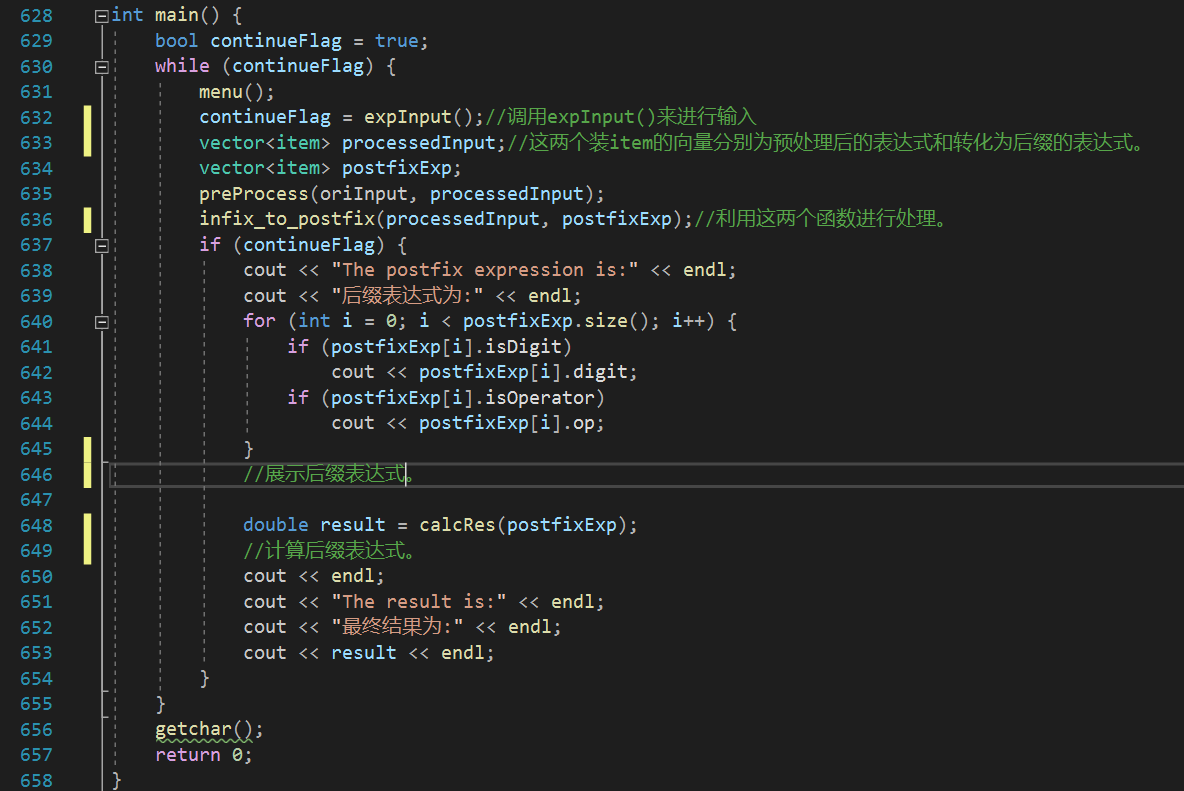
-> struct stackNode

-> class arithStack

-> struct ITEM



## 3.2界面与功能选择的实现



## 3.3 具体实现

### 3.3.1 字符串输入函数expInput()

* 3.3.1.1 核心代码



### 3.3.2 预处理函数preProcess(string&exp, vector<ITEM>&result)

将输入的字符串转化为符号数字分离的数组（向量）。并且剔除出现的小数、未合拢的括号等错误表达式。遇到时报错。

* 3.3.2.1 字符串合法性判断
  + 3.3.2.1.1流程图

Timeline

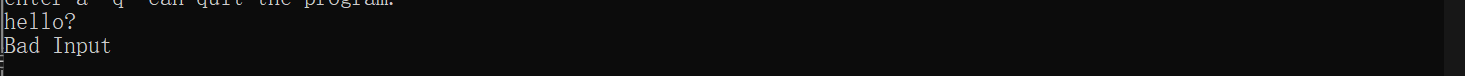
Description automatically generated

* + 3.3.2.1.2 核心代码

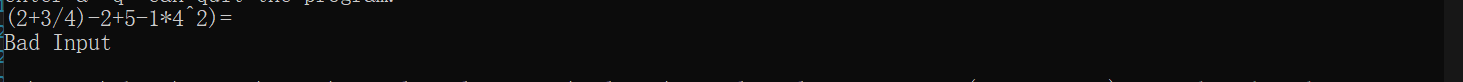
Text

Description automatically generated

* + 3.3.2.1.3 截屏示意
    - 3.3.2.1.3.1 非法字符



* + - 3.3.2.1.3.2 未关闭括号



* + - 3.3.2.1.3.3 输入非整数



* + - 3.3.2.1.3.4 未输入等号



* + - 3.3.2.1.3.5 连续出现运算符号



* + 3.3.2.2 转化为ITEM类型
    - 3.3.2.2.1流程图

Timeline

Description automatically generated

* + - 3.3.2.2.2 核心代码

Text

Description automatically generated

* + 3.3.2.3 特殊情况监测
    - 3.3.2.3.3 流程图

Timeline

Description automatically generated

* + 3.3.2.3.2 核心代码
    - 3.3.2.3.2.1 处理正负数

Text

Description automatically generated

Text

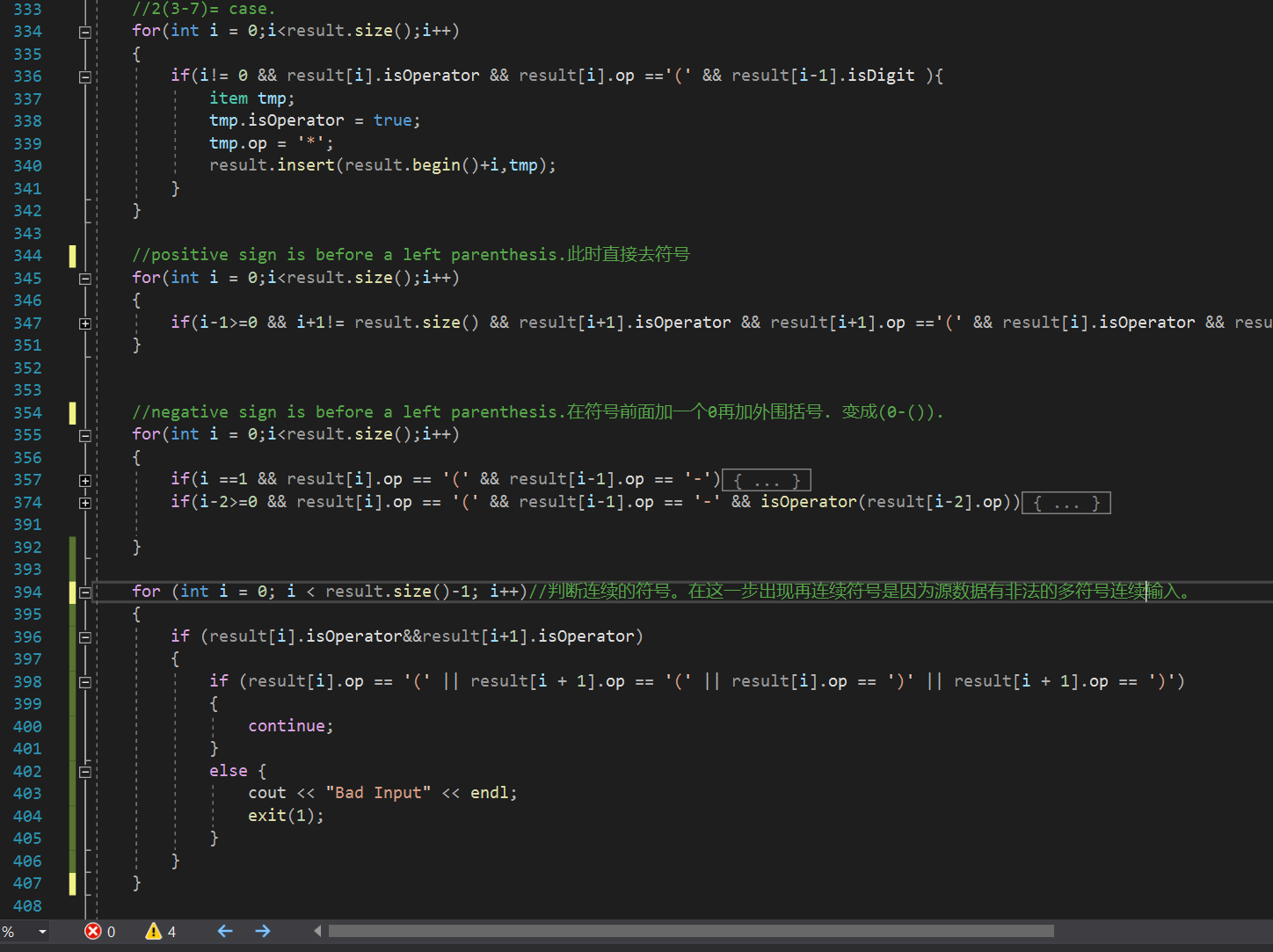
Description automatically generated

* + - 3.3.2.3.2.2 连续出现同级符号

Text

Description automatically generated

* + - 3.3.2.3.2.3 省略乘号/括号外正负号



* + 3.3.2.3.3 截屏示意
    - 3.3.2.3.3.1 负数判断



* + - 3.3.2.3.3.2 带有正号的正数



* + - 3.3.2.3.3.3 连续出现同级符号





* + - 3.3.2.3.3.4 省略乘号的括号乘法



* + - 3.3.2.3.3.5 括号外带有正负号



### 3.3.3 中缀表达式转换为后缀表达式

* + 3.3.3.1 流程图

Diagram

Description automatically generated

* + 3.3.3.2 核心代码

Text

Description automatically generatedText

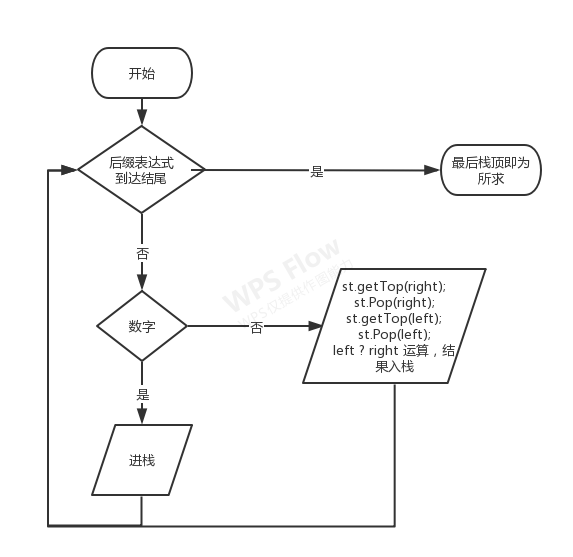
Description automatically generated

* + 3.3.3.3 截屏示意



### 3.3.4 计算：逆波兰表达式

* + 3.3.4.1 流程图



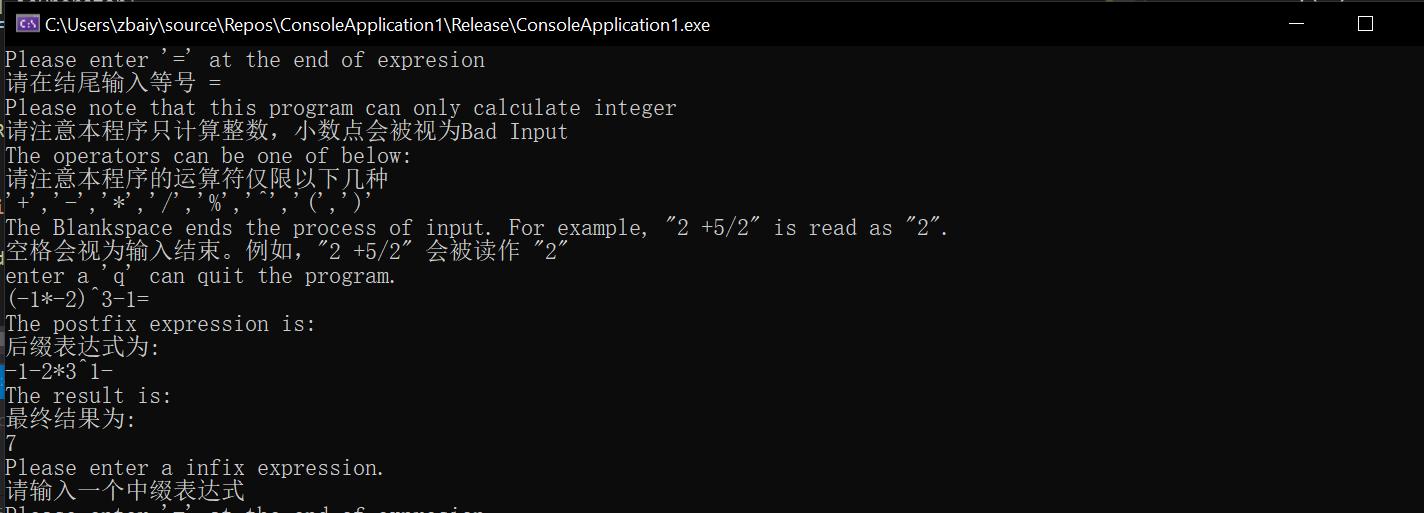
* + 3.3.4.2 核心代码

Text

Description automatically generated

# 4测试

## 4.1 测试实例



Text

Description automatically generated

经验算结果正确（浮点数计算方法不同会导致误差）

Graphical user interface, text, application

Description automatically generated