项目说明文档

数据结构课程设计

——算数表达式求解

作 者 姓 名： 李圣

学 号： 1952723

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji Universit

目录

[1 分析 1](#_Toc29237)

[1.1 背景分析 1](#_Toc1385)

[1.2 功能分析 1](#_Toc28831)

[2 设计 1](#_Toc4856)

[2.1 文件结构设计 1](#_Toc18607)

[2.2 类结构设计 1](#_Toc14134)

[2.3 成员与函数设计 1](#_Toc19144)

[3 实现 6](#_Toc2063)

[3.1 计算中缀运算式（calInfix）的实现 6](#_Toc26613)

[3.1.1 计算中缀运算式（calInfix）流程图 6](#_Toc16898)

[3.1.2 计算中缀运算式（calInfix）核心代码 7](#_Toc17361)

[3.1.3 计算中缀运算式（calInfix）截屏示例 9](#_Toc22254)

[3.2 计算后缀表达式（calSufix）的实现 10](#_Toc15017)

[3.2.1 计算后缀表达式（calSufix）流程图 10](#_Toc17459)

[3.2.2 计算后缀表达式（calSufix）核心代码 11](#_Toc9625)

[3.2.3 计算后缀表达式（calSufix）截屏示例 12](#_Toc29387)

[3.3 计算结果（calResult）的实现 13](#_Toc13497)

[3.3.1 计算结果（calResult）流程图 13](#_Toc14672)

[3.3.2 计算结果（calResult）核心代码 14](#_Toc14233)

[3.3.3 计算结果（calResult）截屏示例 14](#_Toc5894)

[4 测试 15](#_Toc11921)

[4.1 鲁棒性测试 15](#_Toc14437)

[4.1.1 正常案例测试 15](#_Toc5052)

[4.1.2 非法字符测试 16](#_Toc6322)

[4.1.3 括号不匹配测试 16](#_Toc144)

[4.1.4 等于号相关测试 16](#_Toc29097)

[4.1.5 表达式错误测试 16](#_Toc15729)

# 分析

## 背景分析

计算器在日常生活中拥有着较广泛的应用场景。因此，考虑使用Linux系统设计一个程序，使其能对一定格式的算数表达式求解。当输入不符合格式的时候，也能给出一定的提示。

## 功能分析

又题目以及分析可知，该程序至少应该具有的基本功能如下：

1.程序对所有输入的表达式作简单的判断，如表达式有错，能给出适当的提示。支持包括加减，乘除取余，乘方和括号等操作符，其中优先级是等于<括号<加减<乘除取余<乘方

2.能处理单目运算符：+或-。

# 设计

## 文件结构设计

表达式求解的过程中需要使用Vector，因此该项目包含两个文件，如下：

main.cpp

MyVector.h

其中main.cpp是主程序，且main.cpp调用了MyVector.h

## 类结构设计

考虑到功能需要，除了Vector相关的类以外，拟定设计以下类：

class Data

class Calculator

其中Data是数据类，Calculator是计算机类，用于计算相关的操作

## 成员与函数设计

**MyVectorIterator类：**

template<typename val\_type>

using iterator = MyVectorIterator;

using pointer = val\_type\*;

using reference = val\_type&;

**私有成员：**

pointer \_ptr;

int \_pos;

**公有成员：**

MyVectorIterator(pointer ptr = nullptr, int pos = -1)；

~MyVectorIterator()；

int getPos()；

bool operator==(const iterator& it)const；

bool operator!=(const iterator& it)const；

iterator& operator++()；

iterator& operator++(int)；

iterator& operator--()；

iterator& operator--(int)；

reference operator\*()；

pointer operator->()；

**MyVector类：**

template<typename val\_type>

using vector = MyVector<val\_type>;

using reference = val\_type&;

using iterator = MyVectorIterator<val\_type>;

**私有成员：**

int \_size;

int \_capacity;

val\_type\* \_data;

**公有成员：**

MyVector(int capacity = DEFAULT\_CAPACITY)；

MyVector(const vector& vec)；

~MyVector()；

bool empty()；

int size()；

int length()；

int capacity()；

void resize(int newcapacity)；

void insert(iterator pos, const val\_type& val)；

val\_type erase(iterator pos)；

void push\_back(val\_type val)；

val\_type pop\_back()；

vector& operator=(vector& vec)；

vector& operator=(vector&& vec)；

iterator find(val\_type val)；

reference at(int pos)；

reference operator[](int pos)；

reference front()；

reference back()；

iterator begin()；

iterator end()；

void clear()；

**Data类：**

**公有成员：**

Type \_type;

int \_data;

Data(Type type = WRONGTYPE, int data = 0)；

bool operator==(const Data& data)const；

bool operator!=(Data& data)const；

**CalCulator类：**

**私有成员：**

bool \_wrong = false;

string \_input;

MyVector<Data> \_infix;

MyVector<Data>\_suffix;

double \_result;

**公有成员：**

Calculator(string str = "")；

bool wrong()；

void clear()；

double cal(int t1, int t2, char ope)；

void input()；

void include(MyVector<Data>& res, MyVector<char>& tmp, char input, char begin, char end)；

void calInfix()；

void calSuffix()；

void showInfix()；

void showSuffix()；

void calResult()；

int result()；

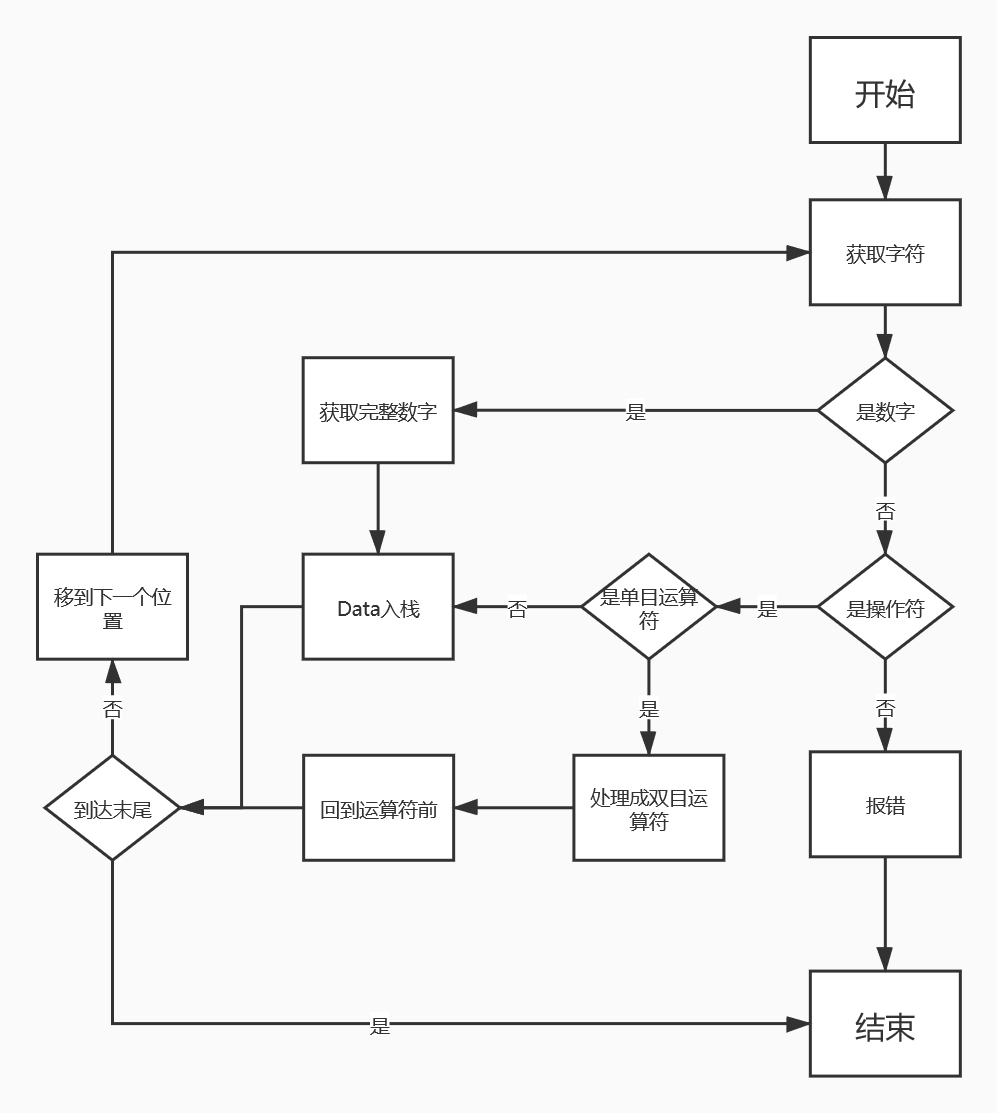
bool calAll()；

void loop()；

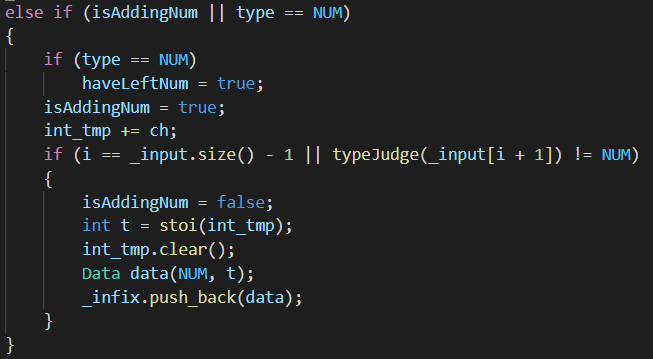
# 实现

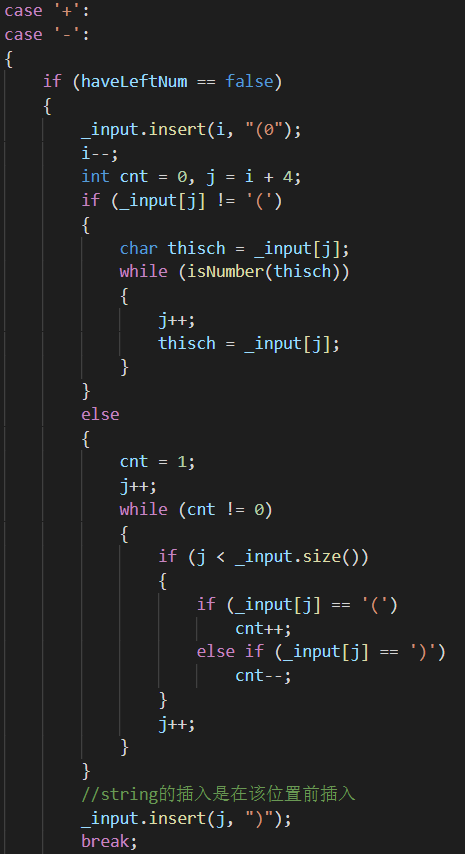
## 计算中缀运算式（calInfix）的实现

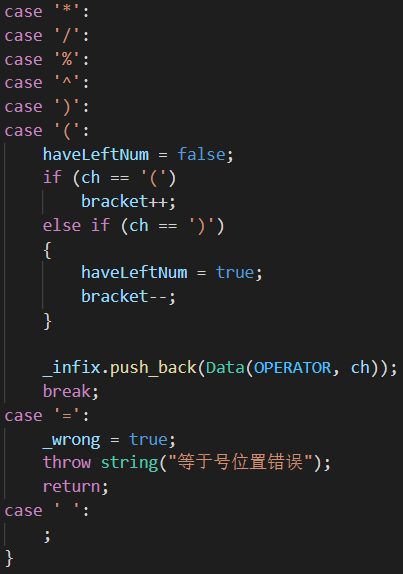
### 计算中缀运算式（calInfix）流程图



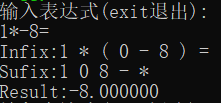
### 计算中缀运算式（calInfix）核心代码





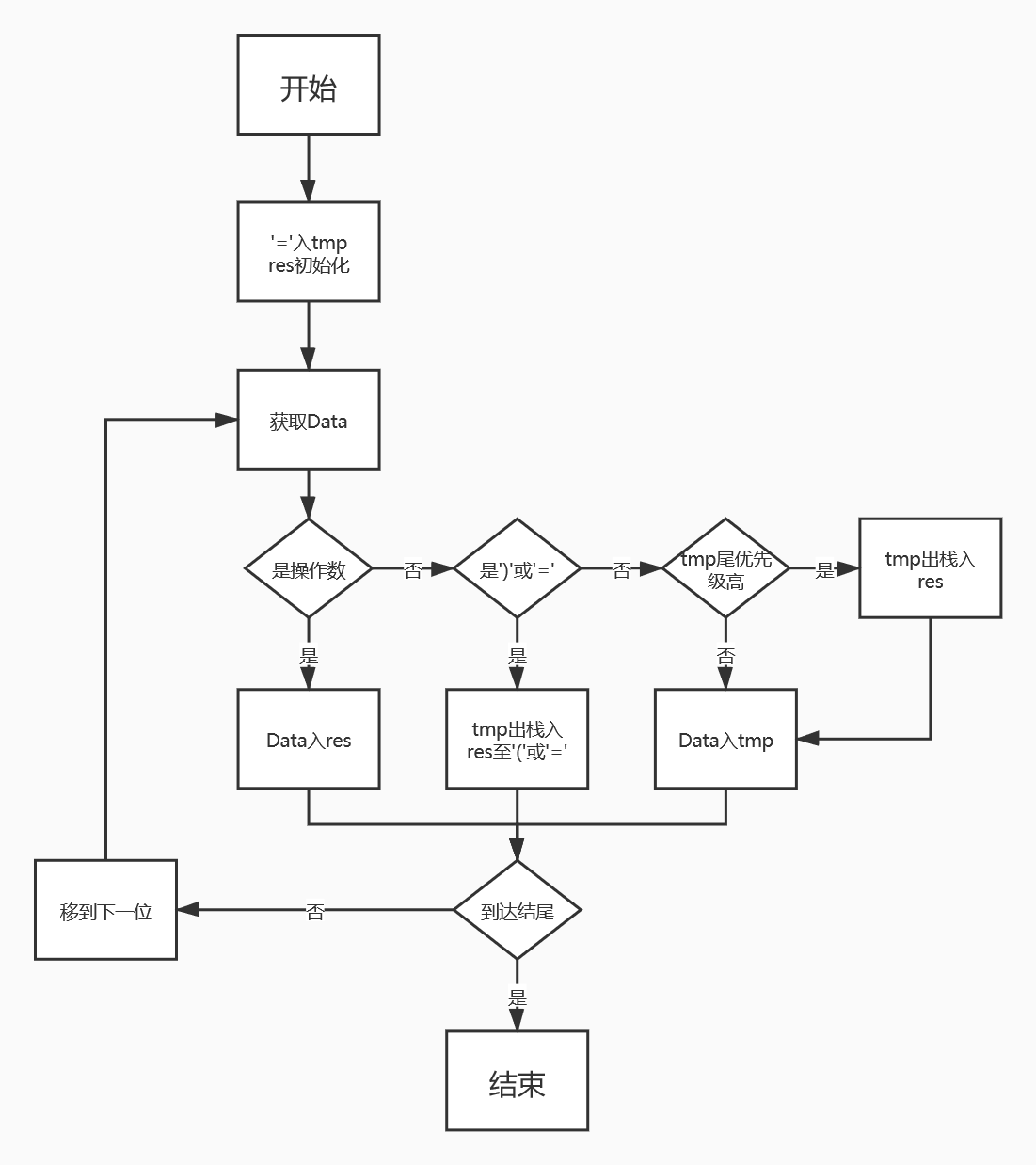


### 计算中缀运算式（calInfix）截屏示例

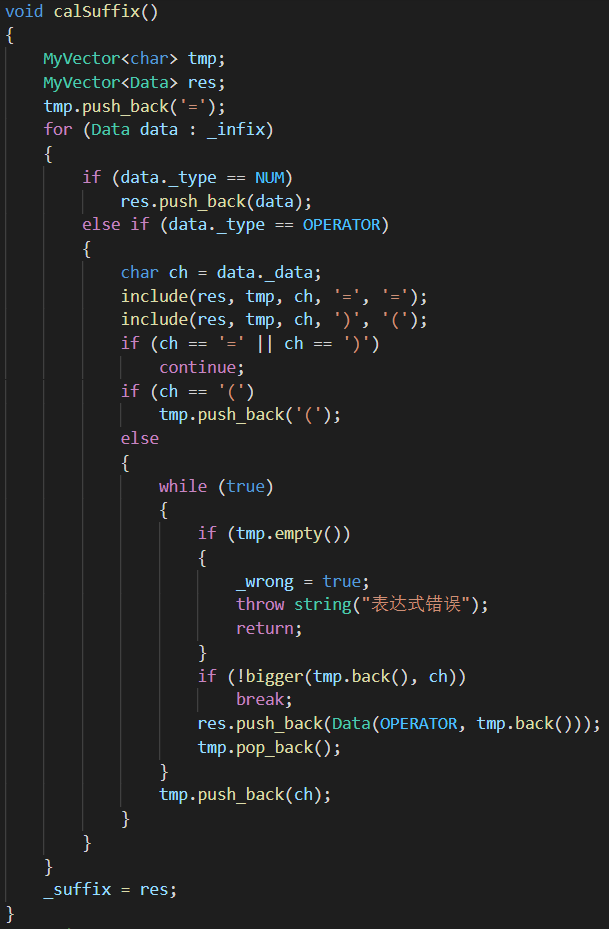


## 计算后缀表达式（calSufix）的实现

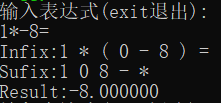
### 计算后缀表达式（calSufix）流程图



### 计算后缀表达式（calSufix）核心代码

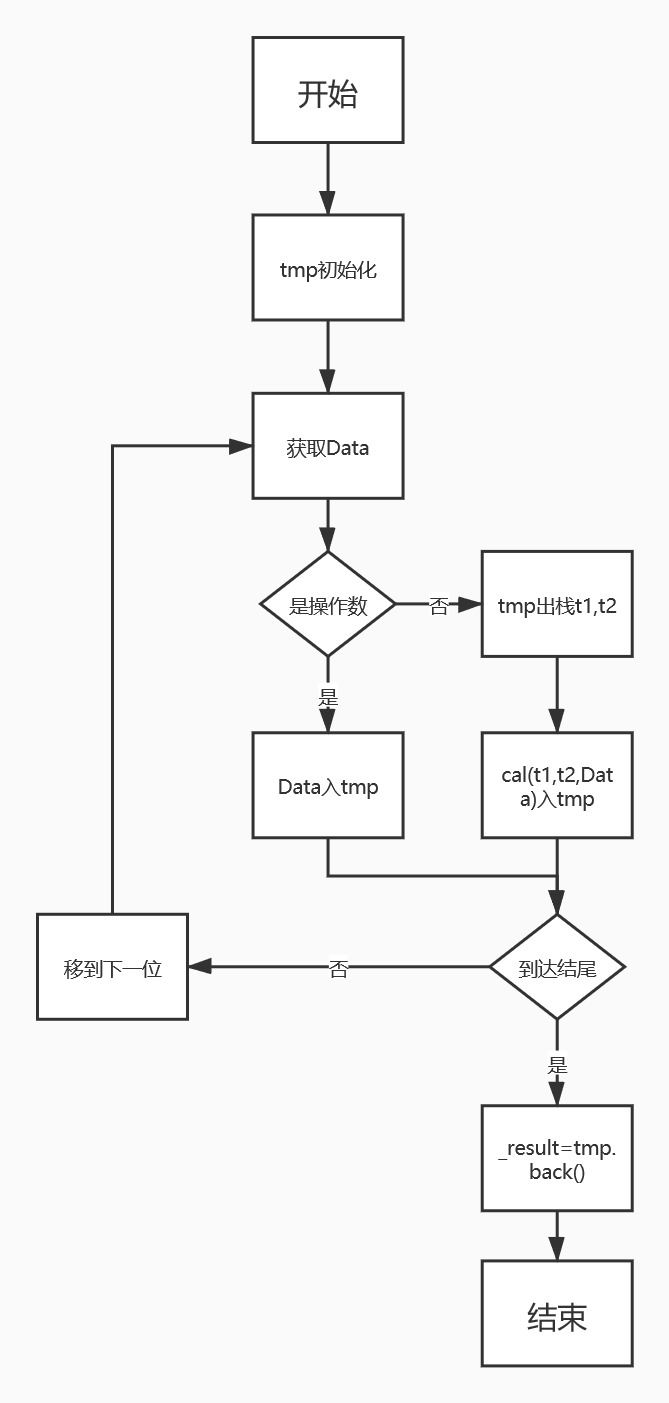


### 计算后缀表达式（calSufix）截屏示例

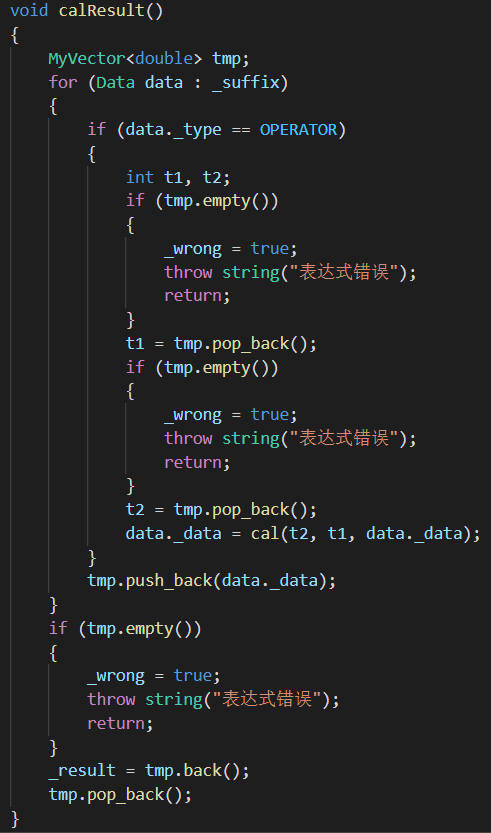


## 计算结果（calResult）的实现

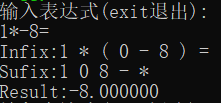
### 计算结果（calResult）流程图



### 计算结果（calResult）核心代码



### 计算结果（calResult）截屏示例



# 测试

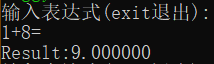
## 鲁棒性测试

### 正常案例测试

**测试用例：**一般情况

**预期结果：**对应的正确结果。

**实验结果：**



**测试用例：**含有括号的情况

**预期结果：**对应的正确结果。

**实验结果：**



**测试用例：**含有单目运算符的情况

**预期结果：**对应的正确结果。

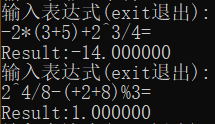
**实验结果：**



**测试用例：**复杂的情况

**预期结果：**对应的正确结果。

**实验结果：**



### 非法字符测试

**测试用例：**输入含有非法字符

**预期结果：**报错并给出提示

**实验结果：**



### 括号不匹配测试

**测试用例：**括号数量不匹配

**预期结果：**报错并给出提示

**实验结果：**



### 等于号相关测试

**测试用例：**未输入等于号

**预期结果：**报错并给出提示

**实验结果：**



**测试用例：**等于号位置错误

**预期结果：**报错并给出提示

**实验结果：**



### 表达式错误测试

**测试用例：**表达式错误

**预期结果：**报错并给出提示

**实验结果：**

