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： 李子涵

学 号： 1851892

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](#_Toc2586)

[1.1 背景分析 1](#_Toc16604)

[1.2 功能分析 1](#_Toc14873)

[2 设计 1](#_Toc29576)

[2.1 数据结构设计 1](#_Toc7868)

[2.2 栈结构设计 1](#_Toc460)

[2.3 成员与操作设计 2](#_Toc10501)

[2.4 程序设计 3](#_Toc1743)

[3 实现 4](#_Toc25861)

[3.1 启动运算功能的实现 4](#_Toc19925)

[3.1.1 启动运算功能流程图 4](#_Toc2686)

[3.1.2 启动运算核心代码 4](#_Toc10979)

[3.2 运算功能的实现 6](#_Toc21386)

[3.2.1 运算功能流程图 6](#_Toc21830)

[3.2.2 运算功能核心代码 6](#_Toc20841)

[3.3 计算功能的实现 8](#_Toc15271)

[3.3.1 计算功能流程图 8](#_Toc30958)

[3.3.2 计算功能核心代码 8](#_Toc17734)

[3.4 单目运算功能的实现 10](#_Toc11041)

[3.4.1 单目运算功能流程图 10](#_Toc21146)

[3.4.2 单目运算功能核心代码 10](#_Toc25606)

[3.5 比较优先级功能的实现 11](#_Toc8040)

[3.5.1 比较优先级功能流程图 11](#_Toc18619)

[3.5.2 比较优先级功能核心代码 11](#_Toc31497)

[3.6 总体系统的实现 12](#_Toc22332)

[3.6.1 总体系统流程图 12](#_Toc12160)

[3.6.2 总体系统核心代码 12](#_Toc19855)

[4 测试 13](#_Toc30339)

[4.1 功能测试 13](#_Toc19249)

[4.1.1 测试1 13](#_Toc5044)

[4.1.2 测试2 14](#_Toc1779)

[4.2 出错测试 14](#_Toc9614)

[4.2.1 被除数为0 14](#_Toc1191)

[4.2.2 非单目运算符用于单个数字 15](#_Toc2774)

[4.2.3 输入非法运算符 15](#_Toc20810)

[4.2.4 询问是否继续时输入除y、n之外字符 15](#_Toc26776)

[4.3.5 查找考号不存在 15](#_Toc16797)

# 1 分析

## 1.1 背景分析

中缀表达式是大部分人最熟悉的算数表达式书写习惯。虽然对于计算机来说，中缀表达式的顺序并不易于计算，但为了方便更多人使用计算机解决算数问题，设计一款中缀表达式计算程序是必要的。

## 1.2 功能分析

作为一个计算器，程序首先需要能够读取表达式，然后按照要求的优先级分析运算的顺序并进行运算，最后输出结果。如果用户输入了错误的算术表达式，还应该有检测错误并发出提醒的功能。

因此，计算器至少应该具有输入、输出、分析、计算、报错的功能。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，计算器需要对输入的数据进行优先级分析，并且相邻的数字和计算符需要一起计算，因此先输入的数据应当先储存，比较后取出。而栈是先入后出的数据结构，和需求正相符，采用栈的数据结构进行程序设计。又因为中缀表达式需要比较运算符的优先级，运算符和数字需要分别储存，因此使用两个栈分别储存数字和运算符。

## 2.2 栈结构设计

栈通常为数组结构或链表结构。本程序使用链表结构的栈，以免数据溢出。使用struct设计结点，方便存取。同时将头结点保存在class的私有中，保证数据的稳定性。又因为需要储存数字和运算符两种类型，需要使用模板类。

## 2.3 成员与操作设计

**链表结点（**template<class T>**LinkNode）**

template<class T>

struct LinkNode{

T data;

LinkNode \*link;

};

**栈类（**template<class T>**Stack）**

**私有成员：**

LinkNode<T> \*top;//栈顶指针

**公有操作：**

Stack(){top=NULL;}//构造函数

~Stack(){}//析构函数

void push(const T&);//进栈

bool pop(T&);//退栈

bool get\_top(T&)const;//获取栈顶元素

bool is\_empty()const{return(top==NULL)?true:false;}//确定栈是否为空

template<class T>

void Stack<T>::push(const T& x){

LinkNode<T> \*p;

p=new LinkNode<T>;

p->data=x;

p->link=top;

top=p;

}

template<class T>

bool Stack<T>::pop(T& x){

if(is\_empty())

return 0;

LinkNode<T>\*p=top;

top=top->link;

x=p->data;

delete p;

return 1;

}

template<class T>

bool Stack<T>::get\_top(T& x)const{

if(is\_empty())

return 0;

x=top->data;

return 1;

}

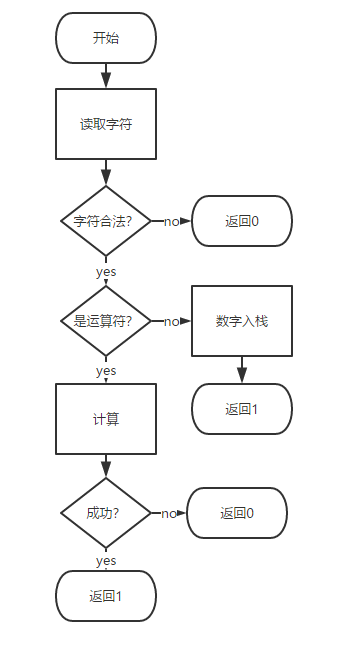
## 2.4 程序设计

首先由用户输入表达式，然后按优先级完成计算，返回计算结果或出错提示，询问用户是否继续计算并依照指令再次计算或退出。

# 3 实现

## 3.1 启动运算功能的实现

### 3.1.1 启动运算功能流程图



### 3.1.2 启动运算核心代码

Stack<char> Ope;//运算符栈

Stack<double> Num;//数字栈

char ch,fore='0';//ch记录新读入字符,fore记录前一字符

double n;

while(cin>>ch)//输入字符

{

if(priority(ch)==6)//非数字非运算符

{

cout<<"无法识别运算符！";

return 0;

}

if(priority(ch))//合法运算符

{

if (!calculate<bool>(ch, Ope, Num, fore))

return 0;

if(ch=='=')//遇到等号不计算

break;

fore = ch;//更新fore

continue;

}

cin.putback(ch);//输入为数字,退回输入流

cin>>n;//重新输出double

Num.push(n);//数字入栈

fore = ch;//更新fore

}

if(Ope.is\_empty()){//计算结束时运算符栈空

if(Num.pop(n)){//得数出栈

if(Num.is\_empty()){//数字栈空

cout<<n<<endl;//输出结果

return 1;

}

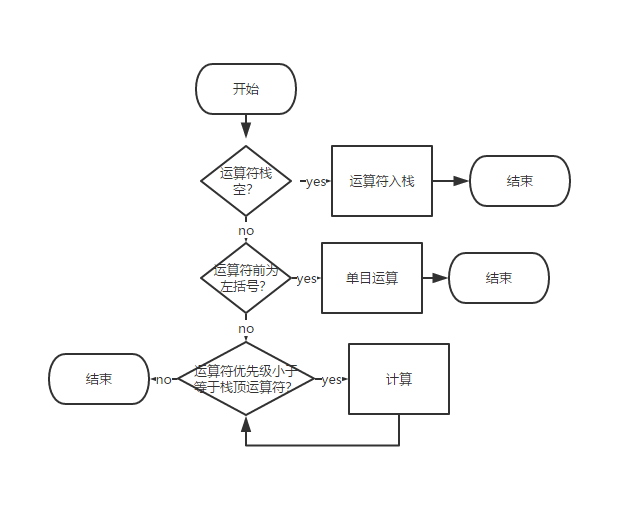
}

}

return 0;

## 3.2 运算功能的实现

### 3.2.1 运算功能流程图



### 3.2.2 运算功能核心代码

template<class T>

bool calculate(char op,Stack<char> &Ope,Stack<double> &Num,char &fore){

if (Ope.is\_empty()||op=='(')//(直接入栈,若运算符栈为空当前运算符直接入栈

{

Ope.push(op);

return 1;

}

if (fore == '(')//若前一字符为(,检测是否为单目运算

{

if (priority(op)==3)//若当前运算符为+-,进行单目运算

{

double x;

cin >> x;

unary(op, x);

Num.push(x);//结果入栈

fore = char(x);//更新fore

return 1;

}

else

{

cout<<'op'<<"不适于单目运算！";//运算符输入错误提示

return 0;

}

}

char top\_op;

Ope.get\_top(top\_op);

if ((op == ')') && (top\_op == '('))//数字前后分别对应(),(出栈,)不入栈

{

Ope.pop(top\_op);

return 1;

}

if (!(op == ')') && (top\_op == '('))//数字前为(,后非),运算符入栈不计算

{

Ope.push(op);

return 1;

}

while(!Ope.is\_empty()&&priority(op)<=priority(top\_op))//当运算符栈非空,当前运算符优先级小于等于栈顶运算符,对栈顶运算符做运算

{

if ((op == ')') && (top\_op == '('))//数字前后分别对应(),(出栈,)不入栈

{

Ope.pop(top\_op);

return 1;

}

if (!(op == ')') && (top\_op == '('))//数字前为(,后非),运算符入栈不计算

{

Ope.push(op);

return 1;

}

if(!result<bool>(Ope,Num))

return 0;

Ope.get\_top(top\_op);//运算完后更新栈顶运算符

}

if(!(op==')')&&op!='=')

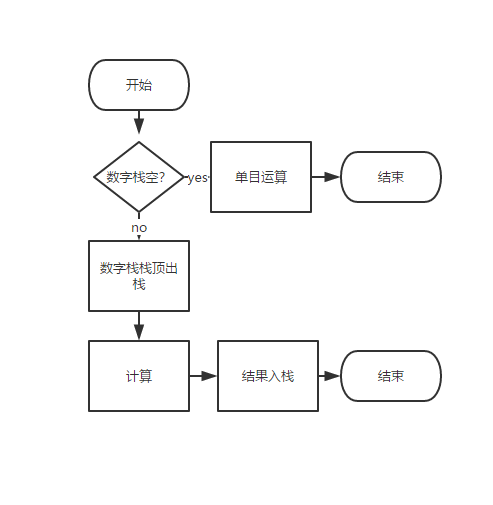
Ope.push(op);//所有运算完成,当前运算符入栈

return 1;

}

## 3.3 计算功能的实现

### 3.3.1 计算功能流程图



### 3.3.2 计算功能核心代码

template<class T>

bool result(Stack<char> &Ope,Stack<double> &Num){

char op;

Ope.pop(op);//取出运算符

double x,y;

Num.pop(y);//数字出栈

if(Num.is\_empty())//判断数字栈是否为空

{

if(priority(op)==3)//数字栈空且运算符为+-,进行单目运算

{

Num.push(unary(op,y));

return 1;

}

cout<<"运算符"<<op<<"不适用于单目运算！";//输入错误

return 0;

}

Num.pop(x);//数字栈不空,栈顶出栈

switch (op)//进行运算

{

case'+':Num.push(x+y);return 1;

case'-':Num.push(x-y);return 1;

case'\*':Num.push(x\*y);return 1;

case'/':

if(y==0)

{

cout<<"0不能作为被除数！";//被除数为0提示

return 0;

}

Num.push(x/y);return 1;

case'%':

Num.push(double(int(x)%int(y)));return 1;

case'^':

int i,re=1;

for(i=0;i<y;i++)

re\*=x;

Num.push(double(re));

return 1;

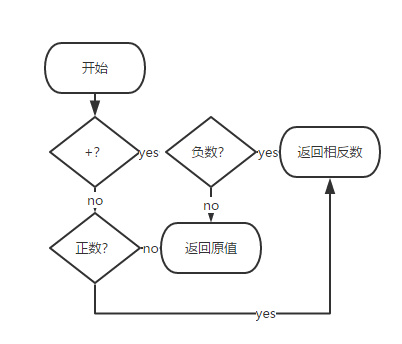
}

return 0;

}

## 3.4 单目运算功能的实现

### 3.4.1 单目运算功能流程图



### 3.4.2 单目运算功能核心代码

double unary(char op,double num){

if(op=='+')//+将负数变相反数

if(num<0)num=-num;

else;

else//-将正数变相反数

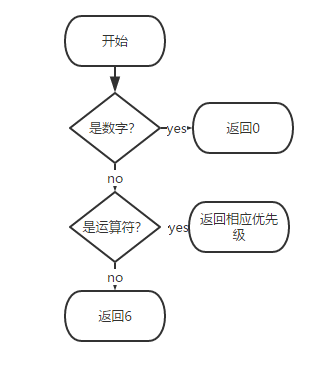
if(num>0)num=-num;

return num;

}

## 3.5 比较优先级功能的实现

### 3.5.1 比较优先级功能流程图



### 3.5.2 比较优先级功能核心代码

int priority(char op){

if(op>='0'&&op<='9')

return 0;//数字返回0

switch (op)//按优先级排列

{

case '=':return 1;break;

case '(':case')':return 2;break;

case '+':case '-':return 3;break;

case '\*':case '/':case'%':return 4;break;

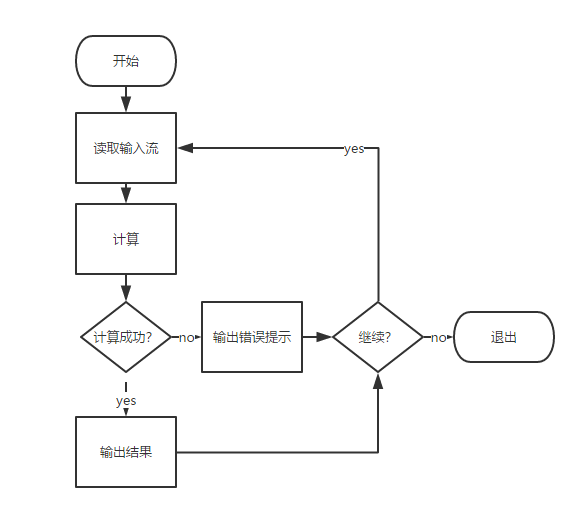
case'^':return 5;break;

}

return 6;//不合法运算符返回6

## 3.6 总体系统的实现

### 3.6.1 总体系统流程图



### 3.6.2 总体系统核心代码

bool flag;//标记运算是否成功

while(1)

{

cout<<"输入表达式："<<endl;

fflush(stdin);

flag=run<bool>();

if(!flag)

{

cout<<"无法计算！";//若flag为0,输出运算出错提示

}

fflush(stdin);

cout<<"是否继续？（y/n）";//询问是否继续

string fb="a";

cin>>fb;//用户输入

while(fb!="y"&&fb!="n")//若输入不是y或n

{

cout<<"（y/n）";//提示正确输入

fflush(stdin);

cin>>fb;//重新输入

}

if(fb=="n")//输入n退出循环

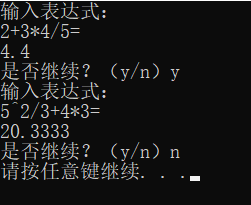
break;

}

system("pause");

return 0;

3.6.3 总体系统截图示例



# 4 测试

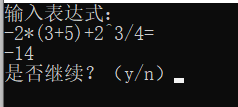
## 4.1 功能测试

### 4.1.1 测试1

**测试用例**：-2\*(3+5)+2^3/4=

**预期结果**：-14

**实验结果**

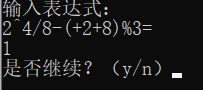


### 4.1.2 测试2

**测试用例：**2^4/8-(+2+8)%3=

**预期结果：1**

**实验结果：**



## 

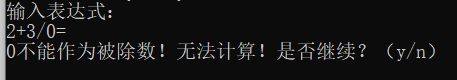
## 4.2 出错测试

### 4.2.1 被除数为0

**测试用例：**2+3/0=

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

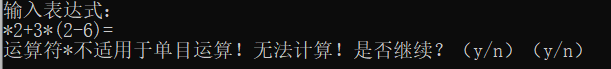


### 4.2.2 非单目运算符用于单个数字

**测试用例：\*2+3\*(2-6)=**

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

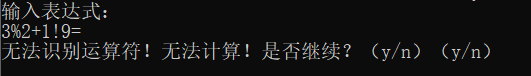


### 4.2.3 输入非法运算符

**测试用例：3%2+1!9=**

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**



### 4.2.4 询问是否继续时输入除y、n之外字符

**测试用例：**a

**预期结果：**程序给出错误信息，提醒重新输入。

**实验结果：**

### 4.3.5 查找考号不存在

**测试用例：**要查找的考号不存在

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

