

## 目录

2019 年 11 月 25 日

一、实验题目：带括号的算术表达式求值.....	2
二、实验的目的和要求:.....	2
三、实验的环境: .....	2
1、硬件环境: .....	2
2、软件环境: .....	2
四、算法描述: .....	2
1、算数表达式优先级 .....	2
2、输入规范化约定 .....	3
3、流程图 .....	3
4、具体步骤.....	4
5、函数及结构体说明 .....	6
五、源程序清单: .....	6
1、“MyStack.h” 文件 .....	6
2、“precheck.h”文件 .....	8
3、“calculate.h”文件 .....	10
4、main 函数.....	13
六、运行结果展示: .....	15
1、测试数据: .....	15
2、测试运行结果展示: .....	18
七、实验运行情况分析 .....	21
1.算法分析 .....	21
2.运行结果分析.....	22
3.运行环境 .....	23
八、参考文献: .....	23

# 带括号的算数表达式求值

计算机科学与技术 专业

学生 李涛      指导老师 朱宏

**【摘要】** 随着计算机技术的广泛应用，计算器已经成为人们日常数字计算必不可少的工具。本程序利用双栈算法，即分别用操作数栈和操作符栈来实现带括号表达式的求值。带括号表达式的求值是栈数据结构应用的一个典型例子。本文首先通过流程图以及具体操作步骤对双栈算法进行了描述，然后给出了四个可运行代码文件清单，其次编写了测试用例进行测试，并展示了运行结果。在实验运行分析部分，通过与逆波兰算法进行对比，解释了采用双栈算法的优点并对运行结果和运行环境作了简要的分析。

**关键词：**栈 带括号表达式的求值 C++语言

## 一、实验题目：带括号的算术表达式求值

## 二、实验的目的和要求：

- 1、采用算符优先数算法,能正确求值表达式;
- 2、熟练掌握栈的应用;
- 3、熟练掌握计算机系统的基本操作方法,了解如何编辑、编译、链接和运行一个 C++ 程序;
- 4、上机调试程序,掌握查错、排错使程序能正确运行。

## 三、实验的环境：

### 1、硬件环境：

处理器：Intel® Core™ i5-8250U CPU @ 1.60GHz 1.80GHz  
RAM : 8.00GB

### 2、软件环境：

操作系统：Windows 10 （64 位，基于 x64 处理器）  
编译软件：Dev-C++ version 5.11

## 四、算法描述：

本算法主要用“栈”数据结构来实现带括号的表达式求值（包括加减乘除运算、模运算、乘方运算）。

### 1、算数表达式优先级

根据带括号的算术表达式运算法则，我们不难得到运算符的优先级如下：

运算符	(	+ -	x / %	^
优先级	0	1	2	3

## 2、输入规范化约定

为了规范表达式格式，我们做如下约定：

字符‘+’表示加法运算，如  $3+4=7$ ；

字符‘-’表示减法运算，如  $4-3=1$ ；

字符‘x’（小写字母 x）表示乘法运算，如  $3\times 5=15$ ；

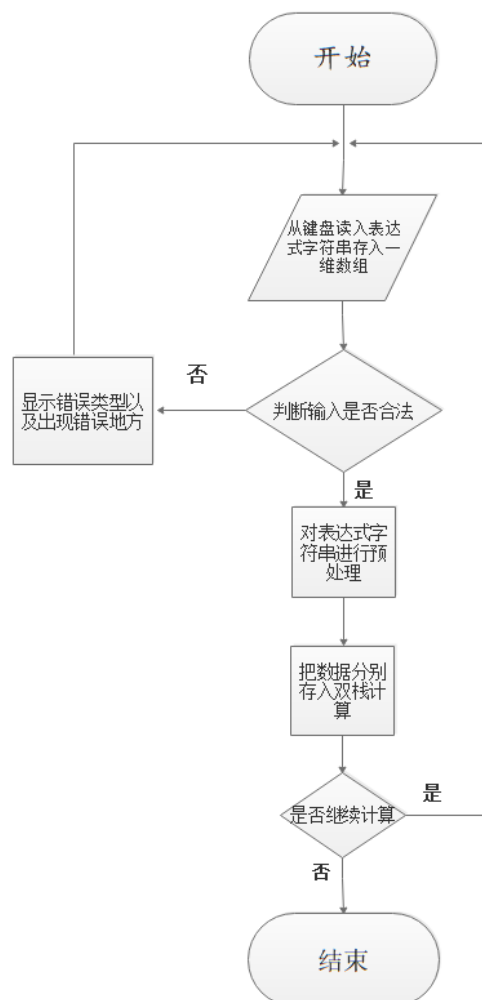
字符‘/’表示除法运算，如  $6/2=3$ ；

字符‘%’表示取模运算，如  $5\%2=1$ ；

字符‘^’表示乘方运算，如  $5^2=25$ 。

约定如有多个括号组合均用小括号()，而不用中括号[]或大括号{}  
最后是否输入=均可，如果用户没有输入=，我们会进行检测并补上=。

## 3、流程图



## 4、具体步骤

### (1) 表达式合法性检验：

对表达式合法性检验的目的是使程序运行时不会异常中断，提高程序鲁棒性，并为后续运算的正常进行提供保障。因此，我们先对用户输入的表达式的合法性进行检验，若表达式不合法，我们通过命令行窗口向用户输出错误类型，并指出错误之处（用符号<|指向错误字符）。在尽可能详尽的考虑几乎所有可能输入的情况后，按照出现频率高低对非法输入作出了如下的分类：

(a) 最常见的是输入字符不是事先约定好的字符，最明显的一个例子是将小写字母‘x’输成了‘\*’（因为我们约定乘法运算的符号是x而不是\*）。

(b) 次常见的是输入时没有转换输入法，误将中文字符输入，如中文括号（“（）”）、中文加减符号（“+”）。对于中文字符的检验有必要在此说明<sup>[1]</sup>：我们知道一个汉字字符占两个字节（8bits），因此，如果字符高四位为1且下一字符高四位也是1则说明有中文字符。

(c) 符号省略。例如  $2(3+4)$ 。因为在数学表达式规范中，数字与数字之间的乘法符号不可省略，因此我们认为该表达式不合法。

(d) 括号不匹配。 $(3 \times 2 + (2 - 3))$ ，表达式本身错误。

(e) 非法输入（表达式中间有括号），如  $3 \times 1 + 1 = 5$

(f) 小数点后面没有数字，如  $3. + 5$

(g) 符号冗余，如  $+3-4$

### (2) 字符串预处理：

若表达式合法，我们将存入数组中的字符串进行预处理。这里主要进行两个处理。

(a) 对于负号前面不是数字的负号字符进行加零处理：例如我们把  $-(-2)$  处理成  $-(0-2)$ 。需要注意的是，该表达式从数学角度不存在歧义，只是因为它不符合我们所使用的数据结构与所设计算法，所以我们要先对其进行预处理。

(b) 检验用户是否输入=：如果没有输入，则在字符串末尾补充=，如果输入，则不进行该操作。这一举措是为了提高程序的鲁棒性，也是为了适应程序所使用的算法。

### (3) 分别把数值和符号存入数值栈和符号栈

(a) 首先，建立数值栈、符号栈两个栈<sup>[2][3]</sup>分别存储操作符和操作数。

(b) 将数字字符转化成数值。得到经过预处理的字符串后，从头到尾依次读取单个字符。为了将数字字符转化成数值，我们建立了 `state_sign` 和 `state_fraction` 两个标志位。一开始，将 `state_sign` 标志位置 1，表示正在读取数字字符（数字字符还没读完），`state_fraction` 标志位置 0，表示即将读取的是一个数整数部分。如果遇到数字，并且 `state_fraction` 状态是 0，则将单个字符的数值（这里指的是当前数字字符的 ASCII 编码值减去字符 ‘0’ ASCII 编码值，下同）每次乘以 10 以后相加（最初  $d=0$ ,  $0*10=0$ ）；遇到小数点 ‘.’，则将 `state_fraction` 状态置为 1，表明即将读取的是一个数的小数部分；在此状态下读取数字字符，则依次将单个字符数值除以 10 后相加。

(c) 将符号字符与符号栈顶字符优先级比较后决定是否存入符号栈。如果遇到的不是数字字符，则将 `state_sign` 置为 0，表明数字字符已经转化成数值，可以将其放入数值栈中。如果读到的是 ‘(’，或者符号栈为空，则将当前字符入符号栈，如果遇到的是 ‘)’ 或者 ‘=’，则将符号栈全部出栈，（一次弹出两个，进行计算后存入数值栈）直到符号栈为空，或者栈顶元素为 ‘(’，如果遇到的不是上述符号并且符号栈不为空，则将当前字符与符号栈栈顶字符优先级进行比较（根据程序需要，我们将字符优先级定义为上面“四.1”所示），如果当前字符优先级小于等于栈顶字符（这里包含等于是为了保证从左到右的运算法则），则直接两次取出数值栈栈顶进行运算，并将运算结果存入栈顶；如果当前字符优先级大于栈顶字符，则把当前字符放入符号栈。

(d) 对整个字符串执行上述操作，最终栈顶元素就是最后的计算

结果。

## 5、函数及结构体说明

### (1) 链表结点

```
template<class N>
struct Node {
    N value;
    Node<N> *next;
    Node();
    Node(N item, Node<N> *add = NULL);
};
```

### (2) 数值栈与符号栈

```
MyStack<double> data;
MyStack<char> sign;
```

### (3) 表达式合法性检验函数与指出错误类型函数

```
bool precheck(char temp[]);
void showerror(char temp[],int e);
```

### (4) 计算功能函数

```
int priority(char );//比较运算符优先级
double aRb(char );//判断运算关系
int base(char );//判断是符号种类
double calculate(char temp[]);//计算
```

## 五、源程序清单：

### 1、“MyStack.h” 文件

```
#ifndef MY_STACK_H
#define MY_STACK_H
#include<iostream>
template<class N>
struct Node {
    N value;
```

```
        Node<N> *next;
        Node();
        Node(N item, Node<N> *add = NULL);
}; //创建链表结点
template<class N>
Node<N>::Node(){
    next = NULL;
}
template<class N>
Node<N>::Node(N item, Node<N> *add){
    value = item;
    next = add;
}

template<class S>
class MyStack {
public:
    MyStack();
    MyStack(MyStack<S> &item); //有参构造函数
    ~MyStack();
    void push(S &item); //入栈操作
    void pop();          //出栈操作
    S top();             //访问栈顶元素
    bool empty();        //判断栈是否为空
    void clear();         //清空栈（优化了 STL）
protected:
    Node<S> *topnode;
};
template<class S>
MyStack<S>::MyStack(){
    topnode=NULL;
}
template<class S>
MyStack<S>::~~MyStack(){
    clear();
}
template<class S>
bool MyStack<S>::empty(){
    if(topnode==NULL)
        return true;
    else
```



```
        return false;
    }
template<class S>
void MyStack<S>::push(S &item){
    Node<S> *newtop = new Node<S>(item, topnode);
    topnode = newtop;
}
template<class S>
void MyStack<S>::pop(){
    if (topnode != NULL){
        Node<S> *oldtop = topnode;
        topnode = oldtop->next;
        delete oldtop;
    }
}

template<class S>
S MyStack<S>::top(){
    if(!empty()){
        return topnode->value;
    }
}
template<class S>
void MyStack<S>::clear() {
    while (!empty()) pop();
}

#endif
```

## 2、“precheck.h”文件

```
#ifndef PRECHECK_H
#define PRECHECK_H
#include <iostream>
#include<string.h>
using namespace std;
void showerror(char temp[],int e);
bool precheck(char temp[]); //函数声明

void showerror(char temp[],int e){ //指出第一次出现错误的位置
    for(int i=0;i<strlen(temp);i++){
        if(i==e) cout<<temp[e]<<"<| "<";
```

```
        else cout<<temp[i];
    }
    cout<<endl;
}
bool precheck(char temp[]){
    MyStack<char> t;
    for(int i=0;i<strlen(temp);i++){
        if((temp[i]&0x80) && (temp[i+1]&0x80)){
//如果字符高位为 1 且下一字符高位也是 1 则有中文字符
            cout<<"请不要输入中文字符"<<endl;
            showerror(temp,i+1);
            return false;
        }
        if(temp[i] == '='&& i!=strlen(temp)-1){
//如果等号出现在表达式中间则报错
            cout<<"Invalid input"<<endl;
            showerror(temp,i);
            return false;
        }
        if(!base(temp[i])){
            cout<<"Please input the sign appointed ! "<<endl;
//出现未知符号
            showerror(temp,i);
            return false;
        }

        if(((temp[i]=='('||temp[i]==')')&&i!=0&&i!=strlen(temp)-1&&base(temp[i-1])==1&&base(temp[i+1])==1)){
            cout<<"Both sides of parenthesis are numbers!"<<endl;
//符号省略
            showerror(temp,i);
            return false;
        }

        if(((base(temp[0])==2)&&temp[0]!='-')||(i==strlen(temp)-1&&base(temp[i])==2)||((base(temp[i])==2&&base(temp[i+1])==2))){
            cout<<"Signs redundancy!"<<endl;//符号冗余（重复输入）
            showerror(temp,i);
            return false;
        }
        if(temp[i]=='.'&&base(temp[i+1])!=1){
```

```
        cout<<"The fraction part you put is wrong!"<<endl;//小数
部分输入错误
        showerror(temp,i);
        return false;
    }
    if(temp[i]=='(') t.push(temp[i]);
    if(temp[i]==')') t.pop();
}
if(t.empty()) return true;
else {
    cout<<"Parenthesis is not matching!"<<endl;//括号不匹配
    return false;
}
}
#endif
```

### 3、“calculate.h”文件

```
#ifndef CALCULATE_H
#define CALCULATE_H
#include <iostream>
#include<string.h>
#include<math.h>
using namespace std;
int priority(char );
double aRb(char );
int base(char );
double calculate(char temp[]);
MyStack<double> data;
MyStack<char> sign; //函数声明

int priority(char ch){ //运算符优先级判断
    if(ch == '+'||ch == '-') return 1;
    if(ch == 'x' || ch == '/' || ch == '%') return 2;
    if(ch == '^') return 3;
    if(ch == '(') return 0;
}

double aRb(char ch){ //判断运算类型
    double a,b;
    a = data.top();
    data.pop();
```

```
b = data.top();
data.pop(); //依次取出栈顶两个元素
switch(ch){
    case '+':
        b += a;
        break;
    case '-':
        b -= a;
        break;
    case 'x':
        b *= a;
        break;
    case '/':
        if(a==0){
            throw "Divider can not be zero!";
            //除数为 0，抛出异常
        }else {
            b /= a;
            break;
        }
    case '%':
        if(a==0){
            throw "divider can not be zero!"; //除数为 0，抛出异常
        }else {
            b = (double)((int)b%(int)a);
            break;
        }
    case '^':
        b = pow(b,a); //乘方、开方运算
        break;
}
return b;
}

int base(char temp){ //判断字符类型
    char base_digital[] = {"1234567890"};
    char base_sign[] = {"+-x/%^"};
    char base_others[] = {"().="};
    if(strchr(base_digital,temp)) return 1;
    else if(strchr(base_sign,temp)) return 2;
    else if(strchr(base_others,temp)) return 3;
```

```
        else return 0;
    }

double calculate(char temp[]){
    double d = 0;
    bool state_fraction = 0; //判断当前数字字符在小数点前面还是后面
    bool state_sign = 1; //判断一个数是否读完
    int count = 1;
    for(int i=0; i<strlen(temp); i++){
        //把数字字符处理成数值
        if(!state_fraction && (base(temp[i]) == 1)){
            d *= 10;
            d += (double)(temp[i] - '0');
            state_sign = 0;
        } else if(temp[i] == '.'){
            state_fraction = 1;

        } else if(state_fraction && (base(temp[i]) == 1)){
            d += (double)(temp[i] - '0') / pow(10, count);
            count++;
            state_sign = 0;
        } else {
            if(!state_sign) {
                data.push(d);
            }
            d = 0;
            state_fraction = 0;
            count = 1;
            if(temp[i] == '(' || sign.empty()) { //如果是(或栈为空直接入栈
                sign.push(temp[i]);
            } else if(temp[i] == ')' || temp[i] == '=') {
                //如果遇到)或者=, 则符号栈全部出栈, 同时从数值栈取出两个数进行运算
                while(!sign.empty() && sign.top() != '('){
                    //根据短路原则, 注意顺序
                    double res = aRb(sign.top());
                    data.push(res);
                    sign.pop();
                }
                if(!sign.empty()) sign.pop(); //弹出(
            } else if(priority(temp[i]) <= priority(sign.top())) {
                //如果优先级小于栈顶, 则直接进行运算, 把运算结果存
```

入数值栈

//等号的位置决定从左到右进行运算

```
while(!sign.empty() && priority(temp[i]) <= priority(sign.top())){
    double res = aRb(sign.top());
    data.push(res);
    sign.pop();
}
sign.push(temp[i]);
}else if(priority(temp[i]) > priority(sign.top())){
//如果优先级大于栈顶，则入符号栈
sign.push(temp[i]);
}
state_sign = 1;    //遇到非数字字符表示数字部分已读完
}
}
return data.top(); //栈顶元素即为最终运算结果
}

#endif
```

#### 4、main 函数

```
#include <iostream>
#include "MyStack.h"
#include "calculate.h"
#include "precheck.h"
using namespace std;
int main() {
    cout<<"-----Welcome!-----"<<endl;
    cout<<"----Input exit to exit----"<<endl;
    double ans; //最终计算结果
    while(1){
        char temp[300]; //存储读入的字符串，用于预处理等操作
        memset(temp,0,300);
        data.clear();
        sign.clear(); //初始化
        cin.getline(temp,300);
        if(strcmp(temp,"exit")==0) break; //程序出口
        //展示更精确的结果
        if(strcmp(temp,"show more")==0){
            printf("ans = %.9lf\n",ans);
```

```
        continue;
    }
    //对空格进行处理
    for(int i=0;i<strlen(temp);i++){
        if(temp[i]==' ')
            for(int k=i;k<strlen(temp);k++)
                temp[k] = temp[k+1];
    }
    if(!precheck(temp)) continue; //预处理结果判断
    //对'-'前没有数字的情况进行加0处理
    if(temp[0]=='-'){
        int k = strlen(temp);
        for(k;k>0;k--) temp[k] = temp[k-1];
        temp[0] = '0';
    }
    for(int i=1;i<strlen(temp);i++){
        if(temp[i]=='-'&& base(temp[i-1])!=1&&temp[i-1]!='-'){
            int k = strlen(temp);
            for(k;k>i;k--) temp[k] = temp[k-1];
            temp[i] = '0';
        }
    }
    //如果用户没有输入'=', 在字符串末尾添加 '='
    int k = strlen(temp);
    temp[k] = '\0';
    if(temp[k-1]!='=') {
        temp[k] = '=';
        temp[k+1] = '\0';
    }
    //运行异常检测
    try{
        ans = calculate(temp);
        cout<<"ans = "<<ans<<endl;
    }catch(const char*message){
        cout<<message<<endl;
    }
}
return 0;
}
```

## 六、运行结果展示：

### 1、测试数据：

测试目的		输入	预期结果	实验结果	备注
测试一位整数运算	加减乘	$-(-2+3-5)\times(-2)$	-8	-8	正常
	除取模	$-8/(3-5)/5\%2$	0	0	正常
	括号测试	$(-2-(-(-(-(-2)\times3)\times1)\times2)/3)-2)/2$	0	0	正常
	乘方开方	$3^2+4^{(1/2)}-3^0$	10	10	正常
	综合测试	$-(-3+5/2-(3^2\%5)\times2)/3$	2.83333	2.83333	正常
测试多位整数运算	加减乘	$-(-23+534-235)\times(-21)$	5796	5796	正常
	除取模	$-(35-50)/5\%2$	1	1	正常
	乘方开方	$12^2+49^{(12/24)}-3^4$	70	70	正常
	综合测试	$-(-32+532/2-(3^5\%6)\times2)/12$	-19	-19	正常
测试带小数运算	加减乘	$-(-2.3+5.4-2.5)\times(-2.1)$	1.26	1.26	正常
	除取模	$-(3.1-5.1)/0.5\%2$	1	1	正常
	乘方开方	$0.3^2+0.09^{(1/2)}-2^{0.5}$	-1.02421	-1.02421	正常
	综合测试	$-(-3.2+5.32/2-(3^5\%6)\times0.2)/1.2$	0.95	0.95	正常
精确度测试	小数精确度（小数点后 9 位）	$-3.4356\times3.5432+1.3245/3.563$	-11.8013	-11.8013	正常
		show more	-11.80128062	-11.8012806	正常
			0	20	
		$3.356456458+3.474475675$	6.83093	6.83093	正常
		show more	6.830932133	6.830932133	正常
		$3.3564564581+3.4744756756$	6.83093	6.83093	正常
		show more	6.830932134	6.830932134	四舍五入
	大整数运算范围（16 位）	$11000123+32452401-43533245$	-80721	-80721	正常
		$1111111111111111\times2$	$2.22222e+015$	$2.22222e+015$	正常
		show more	222222222222	2222222222	正常




			2222.00000000 00	222222.0000 00000	
		1111111111111111x2	2.22222e+016	2.22222e+01 6	正常
		show more	222222222222 22224.000000 000	2222222222 2222224.000 000000	出现错误 位
运 算 异 常 处 理 测 试	除数为 0	5/0	Divider can not be zero!	Divider can not be zero!	正常
		8/(3-3)	Divider can not be zero!	Divider can not be zero!	正常
	模数为 0	5%0	Divider can not be zero!	Divider can not be zero!	正常
		8%(3-3)	Divider can not be zero!	Divider can not be zero!	正常
容 错 性 测 试	小 数 点 前 为 0	$-(3.2+.32/2-(3^{5\%6})x.2)/1.2$	3.03333	3.03333	正常
	用 户 是 否 输入=	$-(3.2+.32/2-(3^{5\%6})x.2)/1.2=$	3.03333	3.03333	正常
	用 户 误 加 空格	110 011 + 203 939 – 321 343	-7393	-7393	正常
非 法 输 入 测 试 ( 检 验 程 序 是 否 会 中 断 )	输 入 中 文 字 符 ( 中 文 括 号 、 中 文 加 减 符 号 )	3x (4+3-7/2)	请不要输入中 文字符	请不要输入 中文字符	正常，程 序无异常 退出情况
		3+2/4-6	请不要输入中 文字符	请不要输入 中文字符	加减符号 是中文字 符
	括 号 不 匹 配	(3x2+(2-3)	Parenthesis is not matching!	Parenthesis is not	正常

				matching!	
非法输入 (表达式中间有括号)	$3 \times 1 + 1 = 5$	Invalid input	Invalid input	Invalid input	正常
出现非中文未约定字符	$3 * 4 + 1$	Please input the sign appointed!	Please input the sign appointed!	Please input the sign appointed!	* 应为小写字母 x
小数点后没有数字	$3. + 5$	The fraction part you put is wrong!	The fraction part you put is wrong!	The fraction part you put is wrong!	表达式无意义
符号冗余	$+ 3 - 4$	Signs redundancy!	Signs redundancy!	Signs redundancy!	+ 应省略
符号省略	$2(3 + 4)$	Both sides of parenthesis are numbers!	Both sides of parenthesis are numbers!	Both sides of parenthesis are numbers!	乘号不能省略
结束运行测试	exit	Return 0	Return 0	Return 0	正常

此外，此代码程序完成后，自己平时都在使用自己编写的这个计算器，使用一个月的过程中，未发生结果运算错误或程序异常中断现象。

## 2、测试运行结果展示：

### (1) 测试一位整数、多位整数、小数运算

 D:\大二上 (2019.7-2020.1) \数据结构课程设计\代码实现\Calculator\Calculator.exe

```
-----Welcome!-----  
----Input exit to exit----  
-(-2+3-5)x(-2)  
ans = -8  
-8/(3-5)/5%2  
ans = 0  
(-2-(-(-(-(-2)x3)x1)x2)/3)-2)/2  
ans = 0  
3^2+4^(1/2)-3^0  
ans = 10  
-(-3+5/2-(3^2%5)x2)/3  
ans = 2.83333  
-(-23+534-235)x(-21)  
ans = 5796  
-(35-50)/5%2  
ans = 1  
12^2+49^(12/24)-3^4  
ans = 70  
-(-32+532/2-(3^5%6)x2)/12  
ans = -19  
-(-2.3+5.4-2.5)x(-2.1)  
ans = 1.26  
-(3.1-5.1)/0.5%2  
ans = 1  
0.3^2+0.09^(1/2)-2^0.5  
ans = -1.02421  
-(-3.2+5.32/2-(3^5%6)x0.2)/1.2  
ans = 0.95
```

## (2) 精确度测试

D:\大二上 (2019.7-2020.1) \数据结构课程设计\代码实现\Calculator\Calculator.exe


```
-----Welcome!-----
----Input exit to exit----
-3.4356x3.5432+1.3245/3.563
ans = -11.8013
show more
ans = -11.801280620
3.356456458+3.474475675
ans = 6.83093
show more
ans = 6.830932133
3.3564564581+3.4744756756
ans = 6.83093
show more
ans = 6.830932134
11000123+32452401-43533245
ans = -80721
1111111111111111x2
ans = 2.22222e+015
show more
ans = 222222222222222.000000000
1111111111111111x2
ans = 2.22222e+016
show more
ans = 2222222222222224.000000000
```

## (3) 运算异常处理测试

D:\大二上 (2019.7-2020.1) \数据结构课程设计\代码实现\Calculator\Calculator.exe


```
-----Welcome!-----
----Input exit to exit----
5/0
Divider can not be zero!
8/(3-3)
Divider can not be zero!
5%0
divider can not be zero!
8%(3-3)
divider can not be zero!
-(-3.2+.32/2-(3^5%6)x.2)/1.2
ans = 3.03333
-(-3.2+.32/2-(3^5%6)x.2)/1.2=
ans = 3.03333
110 011 + 203 939 - 321 343
ans = -7393
```

## (4) 容错性测试与非法输入测试（检验程序是否会中断）

 D:\大二上 (2019.7-2020.1) \数据结构课程设计\代码实现\Calculator\Calculator.exe

```
-----Welcome!-----
----Input exit to exit----
-(-3.2+.32/2-(3^5%6)x.2)/1.2
ans = 3.03333
-(-3.2+.32/2-(3^5%6)x.2)/1.2=
ans = 3.03333
110 011 + 203 939 - 321 343
ans = -7393
3x (4+3-7/2)
请不要输入中文字符
3x (<| 4+3-7/2)
3+2/4-6
请不要输入中文字符
3+<| 2/4-6
(3x2+(2-3)
Parenthesis is not matching!
3x1+1=5
Invalid input
3x1+1=<| 5
3*4+1
Please input the sign appointed!
3*<| 4+1
3.+5
The fraction part you put is wrong!
3.<| +5
+3-4
Signs redundancy!
+<| 3-4
2(3+4)
Both sides of parenthesis are numbers!
2(<| 3+4)
```

## (5) 结束运行测试

 D:\大二上 (2019.7-2020.1) \数据结构课程设计\代码实现\Calculator\Calculator.exe

```
-----Welcome!-----
----Input exit to exit----
-(-2.3+5.4-2.5)x(-2.1)
ans = 1.26
-(3.1-5.1)/0.5%2
ans = 1
0.3^2+0.09^(1/2)-2^0.5
ans = -1.02421
-(-3.2+5.32/2-(3^5%6)x0.2)/1.2
ans = 0.95
exit

-----
Process exited after 3.126 seconds with return value 0
请按任意键继续. . .
```

## 七、实验运行情况分析(包括算法、运行结果、运行环境等问题的讨论)

### 1.算法分析

#### (1) 本程序所采用算法（双栈算法）概述

本程序采用双栈算法实现带括号表达式的求值，即利用两个“栈”数据结构，分别存放操作符（符号栈）和操作数（数值栈）。利用栈的后进先出（last in first out）的特点实现带括号表达式的求值，具有效率高、代码结构清晰易实现的特点。

#### (2) 双栈算法与逆波兰算法比较

实现带括号表达式求值有两种算法，一种是本程序所采用的双栈算法，另一种是逆波兰算法。

本程序一开始也考虑过采用逆波兰算法的方式实现表达式的求值。它的核心思想是将普通的中缀表达式转换为后缀表达式，然后对后缀表达式利用栈结构进行求值。并且除了完成本程序所采用的双栈算法代码编写之外，本人也用逆波兰算法完成了代码编写，通过对比，我选择了双栈算法作为最后程序所运行的代码。

逆波兰算法的具体步骤是：

首先，将中缀表达式转换成后缀式（逆波兰表达式）

- 1) 从左到右读进中缀表达式的每个字符。
- 2) 如果读到的字符为操作数，则直接输出到后缀表达式中。
- 3) 如果遇到“)”，则弹出栈内的运算符，直到弹出一个“(”
- 4) “(”的优先级在栈内比任何运算符都小，但可直接入栈
- 5) 当运算符准备进入栈内时，和栈顶的运算符比较，如果外面的运算符优先级高于栈顶的运算符的优先级，则压栈；如果优先级低于或等于栈顶的运算符的优先级，则弹栈。直到栈顶的运算符的优先级低于外面的运算符优先级或者栈为空时，再把外面的运算符压栈。
- 6) 中缀表达式读完后，如果运算符栈不为空，则将其内的运算符逐一弹出，输出到后缀表达式中。

然后对后缀表达式进行求值

1) 直接读取表达式，如果遇到数字就压栈。

2) 如果遇到运算符，弹出两个数进行运算，将运算结果压栈。

它的优点有：使得运算顺序有规律可寻，计算机能编写出代码完成计算。虽然后缀表达式不利于人阅读，但利于计算机处理。

(3) 为什么不采用逆波兰算法

逆波兰表达式进行一位数运算时，具有很大的优点，但涉及到多位数与小数的运算时，它使用起来却十分不便。因为转化为后缀表达式时，将数值和运算符存入同一个容器中，而如果是多位数与小数，则必须将数值转化为 `double` 型，而不能利用 `char` 型，而运算符是 `char` 型，这样就导致了同一个容器中要存入两种类型的元素。唯一的解决办法是把运算符 `double` 化，即把运算符转为 `double` 型类型的数据与数值一道，存入同一个容器中，为了避免运算符的值与数值的值发生冲突，我们必须对数值的范围作出限制（如若我们规定+为 99999999，那么所有进行计算的数值必须小于 99999999），这样使得运算范围缩小，程序性能下降。

## 2.运行结果分析

(1) 程序能够进行小数和多位数的运算。

(2) 程序能进行加、减、乘、除、取模、乘方、开方及带括号表达式运算。

(3) 程序计算结果可保留 9 位小数，最大运算量级为 10 的 16 次方。

(4) 程序显示结果默认为保留 6 位有效数字，如果想获取更多有效数字，可输入指令“show more”进行查看。

(5) 程序可对除数和模数为 0 的情况抛出异常，并且不会让程序异常中断（即给出错误提示后程序仍可继续运行，进行后面操作）

(6) 程序容错性良好。用户是否输入等号、用户是否省略小数点前的 0、用户是否在表达式中间输入空格，最后都能得出正确的结果。

(7) 程序非法输入识别完善，并且可以对非法输入的字符进行显

示（将整个表达式输出，错误字符后面用<|指出）。程序识别了非法输入中文字符、输入括号不匹配、等号出现在表达式中间、出现未约定字符、小数点后面没有数字、符号冗余、符号省略七大情况进行了识别并提示错误。并且识别这些错误并给出错误提示后，程序不会异常中断，仍可继续运行。程序唯一的出口是输入”exit”指令。

### 3.运行环境

本程序在 Windows 10 操作系统下运行，使用的集成开发环境（IDE）是 Dev-C++ version 5.11，该编译器轻便、操作简单，适合小型项目的编写。本程序的代码可在 Dev-C++ version 5.11 及以上版本、Visual C++ 2013 及以上、VS code 等编译器中运行。

## 八、参考文献：

[1]中文字符的编码.

[http://www.360doc.com/content/18/1009/22/36367108\\_793389185.shtml](http://www.360doc.com/content/18/1009/22/36367108_793389185.shtml)

[2] Shaffer,C.A.. 数据结构与算法分析：C++版：第3版 [M]. 北京：电子工业出版社，2013年10月：60-81.

[3]