数据结构课程设计文档

简易计算器

小组人员：史文翰舒星源刘子欣

1. **问题描述**

输入表达式字符串，以“=”表示结束，计算并输出表达式值。操作数可以是整数或实数，操作符有“+”、“-”、“\*”、“/”、“^”（乘方）和“sin( )”（正弦）、“cos( )”（余弦）、“log( )”（对数）、“ln( )”（自然对数）等函数。

1. **算法思路**

由键盘输入算式串，逐字段处理，遇到数字则把数字压入数字栈中，遇到运算符考虑此运算符与运算符栈顶优先级问题，依据优先级不同，要不把运算符压入栈中，要不把栈顶运算符和数字栈顶元素（依运算符为单目还是双目弹出一个或者两个）弹出进行运算，把运算结果压入栈中，再次考虑优先级问题。

若运算到最后，数字栈中只有一个元素了，那么即为运算结果。

1. **核心算法描述**

bool priority(intop\_in\_stk, intop\_to\_be\_pushed)

//运算优先级比较，返回false为直接压栈，返回true为取出栈顶进行运算

{

if (栈顶为左括号)

return false;

//遇到左括号或者栈外运算符比站内运算符优先级高，则直接入栈

if (两者都是双目运算符)

{

if (待入栈优先级比栈顶高)

return false;

else

return true;

}

else if (待入栈元素为单目运算符，栈顶为双目运算符)

return false;

else if (待入栈元素为双目运算符，栈顶元素为单目运算符)

return true;

else

//两者都是单目运算符，则直接压栈

return false;

}

bool calculate(char input[])

//计算主函数

{

stack<double> number;

stack<int> operation;

把等号压入栈中

for (从第一个字段到最后一个字段)//每个字段为一个数字或一个运算符

{

If(字段为常数)

把常数压入栈中

Else if(字段为数字)

把数字压入栈中

else//字段为运算符

{

While(栈非空且priority一直为true)

{

取出运算符栈顶元素

If(栈顶元素为单目运算符)

{

弹出一个数字栈元素

对元素进行单目运算

把运算结果压入栈中

}

Else //栈顶元素为双目运算符

{

弹出两个个数字栈元素

对元素进行单目运算

把运算结果压入栈中

}

}

if (待压栈运算符为右括号)

把栈顶的左括号弹出

else

把待压栈运算符压栈

}

}

if (数字栈大小不为一)

return false;

输出数字栈栈顶元素。

return true;

}

1. **源程序**

#include<iostream>

#include<string>

#include<cstring>

#include<cmath>

#include<stack>

#include<cstdio>

#include<cstdlib>

using namespace std;

const double PI = M\_PI;

const double e = 2.718281828;

voidinput\_equation(char input[])

//输入去除空格

{

chartmp;

intlen = 0;

while ((tmp = getchar()) != '\n')

if (tmp != ' ')

input[len++] = tmp;

input[len] = '\0';

return;

}

boolcheck\_bracket(char input[])

//检查括号是否配对

{

stack<char>stk;

stk.push('\0');

for (inti = 0; i<strlen(input); i++)

{

if (input[i] == '(')

stk.push('(');

else if (input[i] == ')')

{

if (stk.top() == '(')

stk.pop();

else

return false;

}

}

if (stk.top() == '\0')

return true;

return false;

}

bool check(char input[])

//初步检查输入是否合法

{

if (input[strlen(input) - 1] != '=')

return false;

if (!check\_bracket(input))

return false;

return true;

}

boolunary\_operation(int op, double& x)

//单目运算符

{

switch (op)

{

case 0: return true;

case 8: x=sin(x); return true;

case 9: x=cos(x); return true;

case 10: x=tan(x); return true;

case 11: if (x <= 0.0) return false; x = log10(x); return true;

case 12: if (x <= 0.0) return false; x = log(x); return true;

case 13: x = -x; return true;

}

}

bool binocular\_operation(int op, double& x, double y)//前缀单目运算符的运算定义

//双目运算符

{

switch (op)

{

case 2: x = x + y; return true;

case 3: x = x - y; return true;

case 4: x = x \* y; return true;

case 5: if (fabs(y - 0.0) < 1e-10) return false; else x = x / y; return true;

case 6: x = pow(x, y); return true;

}

}

bool priority(intop\_in\_stk, intop\_to\_be\_pushed)

//运算优先级

{

if (op\_in\_stk == 14)

//遇到左括号或者栈外运算符比站内运算符优先级高，则直接入栈

return false;

if (op\_in\_stk<= 6 &&op\_to\_be\_pushed<= 6)

//两者都是双目运算符

{

if (op\_in\_stk / 2 <op\_to\_be\_pushed / 2)

//0和1同优先级，2和3同优先级，4和5同优先级，6优先级最高

return false;

else

return true;

}

else if (op\_in\_stk<= 6 &&op\_to\_be\_pushed> 6)

return false;

else if (op\_in\_stk> 6 &&op\_to\_be\_pushed<= 6)

return true;

else

//两者都是单目运算符，则直接压栈

return false;

}

bool calculate(char input[])

{

stack<double> number;

stack<int> operation;

operation.push(0);//把等号压入栈中

for (inti = 0; i<strlen(input); i++)

{

if ((input[i] == 'p' || input[i] == 'P') && (input[i + 1] == 'i' || input[i + 1] == 'I'))

//允许用户不分大小写地输入pi

{

number.push(PI);

i++;

}

else if (input[i] == 'e')

//常数e

number.push(e);

else if (input[i] >= '0' && input[i] <= '9' || input[i] == '.')

//数字

{

doublenum = 0.0;

while (input[i] >= '0' && input[i] <= '9')

{

num = num \* 10 + input[i] - '0';

i++;

}

if (input[i] == '.')

{

double power = 1;

i++;

while (input[i] >= '0' && input[i] <= '9')

{

power /= 10;

num += power \* (input[i] - '0');

i++;

}

}

number.push(num);

i--;

}

else

//运算符处理

{

intop\_to\_be\_pushed = -1;

chars\_opration[15][5] = { "=", ")", "+", "-", "\*", "/", "^", "\0", "sin", "cos", "tan", "log", "ln", "~", "(" };

for (int j = 0; j <= 14; j++)

if (j == 7) continue;

else if (strncmp(&input[i], s\_opration[j], strlen(s\_opration[j])) == 0)

{

op\_to\_be\_pushed = j;

i += strlen(s\_opration[j]) - 1;

break;

}

if (op\_to\_be\_pushed == -1)

//输入没有匹配

return false;

while (!operation.empty() && priority(operation.top(), op\_to\_be\_pushed))

{

intop\_in\_stk = operation.top();

operation.pop();

if (op\_in\_stk>= 8 &&op\_in\_stk<= 13 || op\_in\_stk == 0)

//单目运算符的处理

{

doublenum;

if (!number.empty())

{

num = number.top();

number.pop();

}

else

return false;

if (!unary\_operation(op\_in\_stk, num))

return false;

number.push(num);

}

else

//双目运算符的处理

{

double num1, num2;

if (!number.empty())

{

num1 = number.top();

number.pop();

}

else

return false;

if (!number.empty())

{

num2 = number.top();

number.pop();

}

else

return false;

if (!binocular\_operation(op\_in\_stk, num2, num1))

return false;

number.push(num2);

}

}

if (op\_to\_be\_pushed == 1)

//待入栈运算符为右括号则把栈顶的左括号给弹出

operation.pop();

else

//否则压栈

operation.push(op\_to\_be\_pushed);

}

}

if (number.size() != 1)

return false;

cout<< input;

if (fabs(ceil(number.top())-number.top()) < 1e-8)

cout<< ceil(number.top()) <<endl;

else if (fabs(floor(number.top())-number.top()) < 1e-8)

cout<< floor(number.top()) <<endl;

//如果和最近整数差值小于10^-8则输出整数

else

cout<<number.top() <<endl;

return true;

}

int main(void)

{

char input[1000];

bool flag;

cout<< "简单计算器ver.yjbbrqw\n"

<< "简易使用说明：\n"

<< "1.本计算器可以实现基本的加减乘除乘方开方，以及sin, cos, tan, ln, log运算。\n"

<< "2.输入算式时，所有空格将被忽略，且请保证算式末尾为'='，左右括号匹配（请勿省略算式末尾的')'。\n"

<< "3.提供基本常数的pi与e可以直接输入。\n"

<< "4.若要输入负号非减号，请输入~来代替负号。\n"

<< "5.若要开方，请保证被开方数大于0。\n"

<< "6.若浮点数整数部分为零，则可以省略这个零，比如.1234\n"

<< "7.输入exit回车即可退出程序。\n";

system("pause");

while (1)

{

cout<< "请输入算式：\n";

input\_equation(input);

if (strcmp(input, "exit") == 0)

return 0;

flag = check(input);

if (flag)

flag = calculate(input);

if (!flag)

cout<< "输入有误，请重新输入。\n";

}

return 0;

}

1. **测试数据及说明**

简单计算器ver.yjbbrqw

简易使用说明：

1.本计算器可以实现基本的加减乘除乘方开方，以及sin, cos, tan, ln, log运算。

2.输入算式时，所有空格将被忽略，且请保证算式末尾为'='，左右括号匹配（请勿省略算

式末尾的')'。

3.提供基本常数的pi与e可以直接输入。

4.若要输入负号非减号，请输入~来代替负号。

5.若要开方，请保证被开方数大于0。

6.若浮点数整数部分为零，则可以省略这个零，比如.1234

7.输入exit回车即可退出程序。

请按任意键继续. . .

请输入算式：

1+1=

1+1=2 //简单加法测试，与理论相符

请输入算式：

1-1=

1-1=0 //简单减法测试，与理论相符

请输入算式：

3\*2=

3\*2=6 //简单乘法测试，与理论相符

请输入算式：

1/3=

1/3=0.333333 //简单除法测试，与理论相符

请输入算式：

1/0=

输入有误，请重新输入。 //除数为0测试，能够提示错误

请输入算式：

2^3=

2^3=8 //简单乘方运算，与理论相符

请输入算式：

4^(1/2)=

4^(1/2)=2 //简单开方测试，与理论相符

请输入算式：

~4^(1/2)=

~4^(1/2)=nan//负数开偶次方根测试，输出无意义字符

请输入算式：

sinpi=

sinpi=0 //简单sin运算，与理论相符

请输入算式：

sin(pi)=

sin(pi)=0 //带括号sin运算，与理论相符

请输入算式：

cospi=

cospi=-1 //简单cos运算，与理论相符

请输入算式：

tan0=

tan0=0 //简单tan运算，与理论相符

请输入算式：

lne=

lne=1 //简单ln运算，与理论相符

请输入算式：

log10=

log10=1 //简单lg运算，与理论相符

请输入算式：

ln0=

输入有误，请重新输入。//ln无效运算，能够提示错误

请输入算式：

log0=

输入有误，请重新输入。//lg无效运算，能够提示错误

请输入算式：

sincos(pi/2)=

sincos(pi/2)=0 //单目运算符的复合，与理论相符

请输入算式：

~4\*~5=

~4\*~5=20 //负号测试，与理论相符

请输入算式：

.1\*pi=

.1\*pi=0.314159 //省略0测试，与理论相符

请输入算式：

1+~2=

1+~2=-1 //优先级测试，与理论相符

请输入算式：

~(2+2)=

~(2+2)=-4 //优先级测试，与理论相符

请输入算式：

exit=

输入有误，请重新输入。//无效测试，能够提示错误

请输入算式：

(1+(2)=

输入有误，请重新输入。//括号不匹配测试，能够提示错误

请输入算式：

1+2

输入有误，请重新输入。//无等号测试，能够提示错误

请输入算式：

(sin pi - tan(pi / 4) )\* (lne + log10)=

(sinpi-tan(pi/4))\*(lne+log10)=-2 //综合测试，与理论相符

请输入算式：

exit

//能够成功退出

Process returned 0 (0x0) execution time : 386.338 s

Press any key to continue.

1. **结果分析和结论**

这个简易计算器对各种正确的表达式都可以正确输出结果，对于错误表达式可以提示出错，功能完善，简单易用，容易上手。

但是对偶次方根被开方数为负的情况无法判别，这点仍需改进。

1. **心得体会**

史文翰：这次是我负责的文档工作，由于计算器工作量繁重，其文档内容也就相对而言丰富起来。对于核心算法描述，我们将重要的概念和步骤用红色字体作为突出标注，这一点在测试文档方面也是相同的。其余方面如分析、概述要做到尽可能的简单明晰，让人一看便懂。计算器让我们学到了很多，这次没有参与到编码，实感遗憾。

舒星源：作为测试，我一开始也没有检查出什么问题。然后我想到了在数学上本来就不在定义域内的一些输入，这对在程序中规避反馈错误输入尤为重要，因为处理不当程序会陷入死循环或者输出无意义的字符。这次我认识到计算器看似简单常用其实在背后有这么多需要思考解决的问题。

刘子欣：这次是我来编码的，编程过程中遇到的最大的问题就是单目运算符和双目运算符优先级的处理的不同，一开始我采用的是相同的处理方式，即优先级相同的话就弹出，但是这样的话遇到sincos1=之类的单目运算符复合便会提示输入错误，原因在于单目运算符对这种情况正确做法应该是优先级相同直接压栈。通过这次编码，我了解了测试的重要性，如果没有舒星源的测试，我不可能发现我的程序原来有这么严重的逻辑错误。我还对计算器的原理有了更深入的理解，并且希望在以后继续为这个计算器添加各种功能。

1. **分工与签名**

编码：刘子欣

文档：史文翰

测试：舒星源