

# 数据结构与算法课程设计报告

题目 利用后缀表达式计算中缀表达式的值

学号： 2017141051019

姓名： 王崇智

教师点评：

成绩：

# 四川大学计算机学院、软件学院

## 实验报告

学号: 2017141051019 姓名: 王崇智 专业: 计算机科学与技术 班级: 173040104 第 10 周

课程名称	数据结构课程设计	实验课时	2
实验项目	利用后缀表达式计算中缀表达式的值	实验时间	11.5
实验目的	<ol style="list-style-type: none"><li>1. 采用算符优先数算法, 能正确求值表达式, 并熟练掌握栈的应用;</li><li>2. 熟练掌握如何编辑、编译、链接和运行一个 C++ 程序;</li><li>3. 上机调试程序, 掌握查错、排错使程序能正确运行。</li></ol>		
实验环境	WIN10 DEV C++ GCC 8.1.0		

## 一、问题描述

中缀表达式是方便人类思维计算的一种计算多项式结果的算法，同样的形式在计算机中需要经过额外的处理才能完成工作。而后缀表达式的特点就是计算机运算非常方便，需要用到栈；计算机处理过程只需要顺序读入，如果遇到数字，则放入栈中，如果是运算符，则将两个栈中数字取出进行运算；

## 二、基本要求

从键盘输入一个字符串表达式，形如“ $a+b*(c-d)^e=$ ”，利用中缀表达式转化为后缀表达式的思想计算求解。

## 三、工具/准备工作

DevC++ 与 gcc

## 四、分析与实现

本实验需要利用堆栈的数据结构，将输入字符串扫描后分别利用操作符堆栈存储运算符和括号，并利用数据堆栈存储数字。其中字符串中含有加减乘除与幂运算的运算符。

有几点需要注意：

1. 表达式可能以‘+’，或‘-’号开头，此时应先向数据栈中压入一个 0，再对一个预先设置好的首位符号位进行操作，实际处理中相当于将其保存为相应的正数或者负数。
2. 在对字符串处理之前应先检查格式是否正确，如果检测到末尾忘记输入‘=’，应能够自动将其加上，以方便最后的操作。
3. 考虑到中缀转后缀的关键在于各个运算符本身的优先级以及与括号出现的相对位置，预先设计好各个算符的优先级，如{‘=’: 0, ‘+’: 1, ‘-’: 1, ‘\*’: 2, ‘/’: 2, ‘^’: 3}
4. 本实验利用的堆栈采用 C++自带的库<stack>定义实验需要的两个库 nums 与 ops。

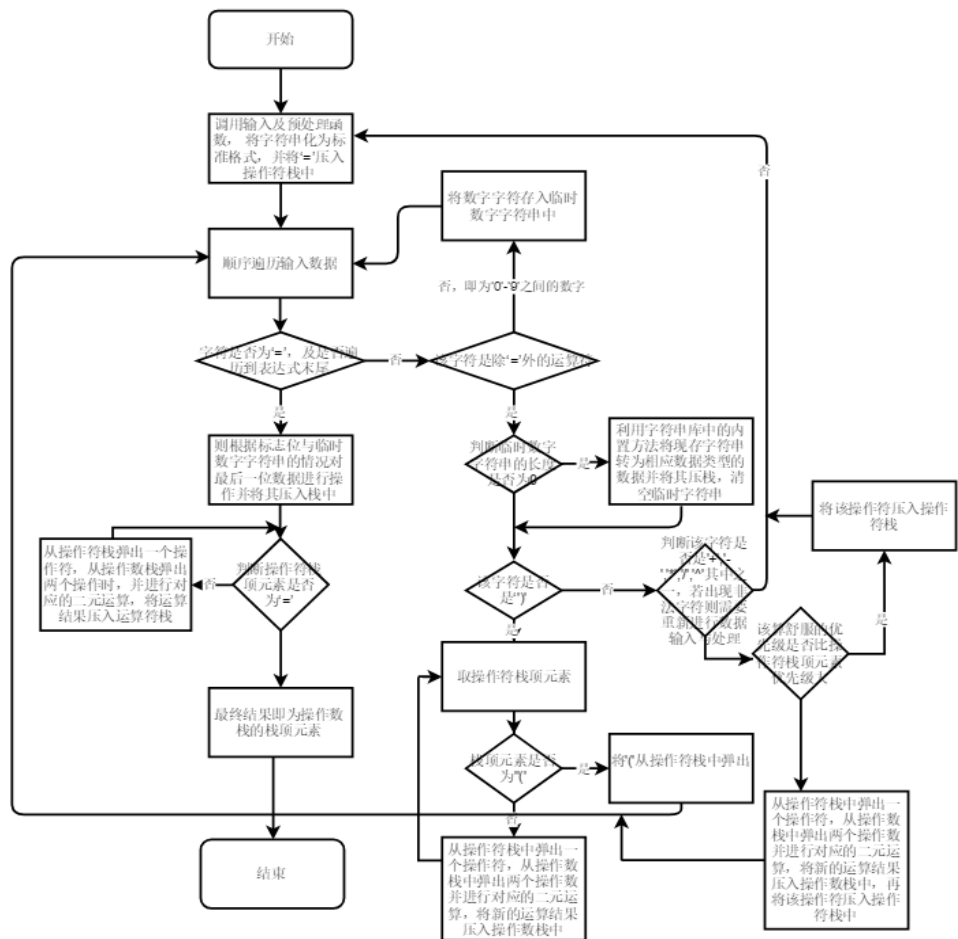
算法思想：

1. 先将表达式化为理想模式，即如果结尾不是‘=’，则将字符串末尾加上‘=’，并将‘=’压入操作符栈中。
2. 顺序扫描字符串，若未遇到‘=’，则执行步骤 3，否则执行步骤 7。
3. 如果遇到的是字符，则进行判断。如果字符为‘(’或‘)’,则执行步骤 4，其他则执行步骤 5。如果是非字符则进行步骤 6。如果除括号外的第一个出现的操作符为‘+’或‘-’，则先将 0 压入操作数栈
4. 如果字符为‘(’，则将‘(’压入操作符栈中,然后继续对下一个字符进行判断。如果字符为‘)’,则不将其入栈，开始对栈中元素进行操作，遍

历操作符栈顶元素，每次遍历直到遇到‘(’，同时弹出操作数栈中的两个操作数进行二元的算术运算对应操作，并将每一次遍历的操作数重新进行压栈，遍历结束后将‘(’弹出操作符栈

5. 如果是除了‘(,)’之外的操作符，则先将该操作符的优先级与操作符栈顶元素的优先级进行判断，如果优先级比栈顶元素高，则将该操作符压入操作符栈。如果优先级比栈顶操作符低，则先弹出一个操作符和两个操作数进行二元运算并将运算结果压入操作数栈后再将该操作符压入栈中
6. 用临时字符串存储数字字符，进行操作 2.
7. 先判断临时字符串是否存有数字，如果有则将其运算，并判断正负存入栈中。否则直接对操作符栈进行判断，如果栈顶元素不是‘=’，则每次遍历弹出一个操作符和两个操作数，并进行二元算术运算，并将运算结果重新压入栈中。
8. 操作数栈中留下的最后一个元素就是最终的运算结果。

算法流程图如下：



源程序如下：

```

#include<stack>
#include<iostream>
#include<cmath>
#include<cstring>

using namespace std;

class EV{
    stack<double> nums;
    stack<char> ops;
    string test;
public:
    EV(string &input)
    {
        test=input;
    }
    int ischar(char op)
    {
        switch(op){

            case '+':
            case '-':
            case '*':
            case '/':
            case '^':
            case '(':
            case ')':return 1;break;
            default:return 4;break;
        }
    }
    int priroity(char op)
    {
        switch(op){

            case ';':
            case '=':return 0;break;
            case '+':
            case '-':return 1;break;
            case '*':
            case '/':return 2;break;

```

```

        case '^':return 3;break;
        default:return 4;break;
    }
}
double tosolve(double numl,double numr,char op)
{
    if(op=='+')
    {
        return numl+numr;
    }
    else if(op=='-')
    {
        return numl-numr;
    }
    else if(op=='*')
    {
        return numl*numr;
    }
    else if(op=='/')
    {
        return numl/numr;
    }
    else
    {
        return pow(numl,numr);
    }
}
void Run();
};
void EV::Run()
{
    int len=test.size();
    if(test[len-1]!='=')
    {
        test+="=";
    } //确保获得一位终止标志
    string strnum;

    double tempnum;

```

```

int flag=0;//用来记录处于第一个数字的正负性
len=test.size();
cout<<test;
ops.push('=');
int error=0;
for(int i=0;i<len;i++)
{
    if(error==0)
    {
        cout<<"数据异常，请重新输入"<<endl;
        break;
    }
    if(test[i]=='=')
    {
        if(strnum.size()!=0)
        {
            tempnum=stod(strnum);
            if(flag== -1)
            {
                tempnum=pow(-1,flag)*stod(strnum);
                flag=0;
            }
            nums.push(tempnum);
            strnum.clear();
        }
        while(ops.top()!='=')
        {
            char op=ops.top();
            ops.pop();
            double numr=nums.top();
            nums.pop();
            double numl=nums.top();
            nums.pop();
            double result=tosolve(numl,numr,op);
            nums.push(result);
        }
    }
    else
    {
        If(ischar(test[i])==4)

```

```

{
    error=1;
}
else if(ischar(test[i])==1)
{ //判断是否为符号
    if(test[i]=='+' && (i==0 || (i>=1 && test[i-1]!='(')))
    {
        nums.push(0);
        continue;
    }
    if(test[i]=='-' && (i==0 || (i>=1 && test[i-1]!='(')))
    {
        nums.push(0);
        //flag=-1;
        continue;
    }

    if(strnum.size()!=0)
    { //现在碰到了符号，之前字符串有长度，即有数字，我要记
录下来

        double tempnum=stod(strnum);
        tempnum=pow(-1,flag)*tempnum;
        flag=0;
        nums.push(tempnum);
        strnum.clear();
    }

    if(test[i]=='(')
    {
        ops.push('(');
        continue;
    }
    if(test[i]==')')
    {
        while(ops.top()!='(')
        {
            char op=ops.top();

            ops.pop();
            double numr=nums.top();

```

录下来






```

        nums.pop();
        double numl=nums.top();
        nums.pop();
        double tempres=tosolve(numl,numr,op);
        nums.push(tempres);
    }
    ops.pop();//把左括号退出来
}
else
{
    if(priroity(test[i])>priroity(ops.top()))
    {
        ops.push(test[i]);
    }
    else
    {
        char op=ops.top();
        ops.pop();
        double numr=nums.top();
        nums.pop();
        double numl=nums.top();
        nums.pop();
        ops.push(test[i]);
        double tempres=tosolve(numl,numr,op);
        nums.push(tempres);
    }
}
}
else
{
    strnum+=test[i];
}
}
}
cout<<nums.top()<<endl;
}

int main()

```

	<pre> {     string ab;     cin&gt;&gt;ab;     EV a(ab);     a.Run();      return 0; } </pre>
<p>数据 记录 和计 算</p>	<h3>五、测试与结论</h3> <p>例 1: <math>(4+9.5) - 3*(2)^2</math></p> <p> D:\大二秋季学期\数据结构\BIG2.exe</p> <pre> (4+9.5)-3*(2)^2 (4+9.5)-3*(2)^2=1.5 </pre> <p>例 2: <math>(1+(2-(5*6))^2) / 3</math></p> <p> D:\大二秋季学期\数据结构\BIG2.exe</p> <pre> (1+(2-(5*6))^2)/3 (1+(2-(5*6))^2)/3=261.667 </pre> <p>例 3: <math>(-1-1-1+3) + (1.23)^3</math></p> <p> D:\大二秋季学期\数据结构\BIG2.exe</p> <pre> (-1-1-1+3)+(1.23)^3 (-1-1-1+3)+(1.23)^3=3.86087 </pre>

	<p>通过计算器计算上述前三个例子，得出的结果完全一致。以此我们基本验证了算法能够处理加法、减法、乘法、除法、乘方等基本运算，同时有能力处理负数和浮点数，也能够准确地理解处理多级括号的运算优先级</p>
小结	<p><b>六、课程设计总结</b></p> <p>本次实验充分了解了堆栈的使用，特别是顺序栈的使用方法，也理解了中缀表达式变后缀表达式的流程。但该表达式计算还不够非常完善，<math>- -1</math> 和 <math>---1</math>，还没能解决这样的问题，会在后续过程中继续尝试解决。这个过程中也充分了解了分支的实现过程对之后遇到复杂的问题有一定的借鉴意义。在编写一个复杂的程序时，需要提前考虑好各部分之间的逻辑关系，否则就要在主体写完之后重新添加模块，会产生很多麻烦。</p>
指导老师评议	<p>成绩评定：_____ 指导教师签名：_____</p>

## 实验报告说明

### 专业实验中心

**实验名称** 要用最简练的语言反映实验的内容。如验证某程序、定律、算法，可写成“验证 $\times\times\times$ ”；分析 $\times\times\times$ 。

**实验目的** 目的要明确，要抓住重点，可以从理论和实践两个方面考虑。在理论上，验证定理、公式、算法，并使实验者获得深刻和系统的理解，在实践上，掌握使用实验设备的

技能技巧和程序的调试方法。一般需说明是验证型实验还是设计型实验，是创新型实验还是综合型实验。

**实验环境** 实验用的软硬件环境（配置）。

**实验内容（算法、程序、步骤和方法）** 这是实验报告极其重要的内容。这部分要写明依据何种原理、定律算法、或操作方法进行实验，要写明经过哪几个步骤。还应该画出流程图（实验装置的结构示意图），再配以相应的文字说明，这样既可以节省许多文字说明，又能使实验报告简明扼要，清楚明白。

**数据记录和计算** 指从实验中测出的数据以及计算结果。

**结论（结果）** 即根据实验过程中所见到的现象和测得的数据，作出结论。

**小结** 对本次实验的体会、思考和建议。

**备注或说明** 可写上实验成功或失败的原因，实验后的心得体会、建议等。

**注意：**

- 实验报告将记入实验成绩；
- 每次实验开始时，交上一次的实验报告，否则将扣除此次实验成绩。