**利用栈求解中缀表达式**

妥明翔 哈尔滨工业大学计算机科学与技术学院，1403104班；150090

**【摘要】：研究如何通过栈来实现计算一个合理的中缀表达式的运算，方法：首先利用栈将中缀表达式转换为后缀表达式，再通过栈将后缀表达式计算出来。**

1. **实现运算功能综述**

本算法目前实现的运算功能为：加法运算、减法运算、乘法运算、除法运算、求余运算、以及括号等，数据要求为整数或者小数，对算术表达式的要求为必须是合法的算术表达式

1. **将中缀表达式转化为后缀表达式**
   1. 核心算法思想

输入一个合理的中缀表达式，从左向右依次扫描，伪代码如下：

while：扫描未结束

chr = 当前扫描元素

if chr为数：

加入后缀表达式

else chr为操作符：

if 栈为空，或栈顶元素为“(”,或栈顶元素优先级低于chr：

chr入栈

else if 栈顶元素优先级高于或等同于chr：

while栈顶元素优先级高于或等同于chr，且栈顶元素不为“(”,且栈不为空：

栈顶元素出栈并加入到后缀表达式中

新的栈顶元素与chr的优先级作比较

chr入栈

else if chr为“)”:

栈顶元素不断出栈并依次加入到后缀表达式中，直到遇到“(”，但“(”直接出栈而不加入到后缀表达式中

else 优先级高于栈顶元素

chr直接入栈

chr = 下一个扫描字符

while 栈不为空

栈内元素出栈并添加到后缀表达式中

* 1. 操作符优先级的判断

优先级为第一级的操作符为：（、），优先级为第二级的操作符有：\*、/、%，第三级的操作符有：+、—，一级操作符优先级最高，其余各级依次递减。同级运算符中优先级相同。本算法使用一个cmp\_op的函数，原型为：int cmp\_op(char a, char b)，这个过程比较简单，大致算法思想是：将同级的操作符储存到同一个字符数组中，再将致谢字符数组按运算优先级存储到一个叫op的二维字符数组中，像这样：char op[][4] = { "()", "\*/%", "+-", };这样只需要在op中检索操作符a和b，并储存他们在op中的位置。再通过位置的比较就可以得到二者的优先级。其中a>b,a<b,a=b分别返回 ：1,0，-1.

* 1. 对输入表达式的处理并获得后缀表达式

2.3.1数据存储

操作数和操作符全部都是利用C++中的string类来储存的，使用string类的优点有二，其一为string类可以作为函数的返回值直接进行返回，在出栈和读取栈顶元素会比较方便；其二为string类含有很多内置的字符串处理方法，更利于对操作数进行处理，比如字符串操作数转化为浮点型操作数，以及将负号用其他符号替换以防止与减号混淆而发生未知的错误。后缀表达式是有一个string类型的数组储存的。栈中的数据也是string类型。

2.3.2算术表达式的预处理

预处理主要处理两种情况，其一为将输入表达是中的负号用‘！’代替，以区分负号和减号。另外，对于形如-（2+3）\*5这样一开始就是一个负号加左括号的式子，直接使用上述核心算法会出错，因此将其处理成为-1\*（2+3）\*5，这样就适应上述核心算法了。预处理伪代码如下：

chr=表达式第一个字符

if chr== ‘-’并且chr下一个为‘（’

在chr后面插入‘1\*’

while 表达式扫描未结束：

chr=当前扫描字符

if chr=‘-’

if chr不是第一个字符，或者chr前后字符都是数

chr为减号

else

chr是负号

将后缀表达式中chr换成‘！’

chr=下一个字符

2.3.3提取操作数和操作符

由于操作数会有不止一个字符，甚至还包含小数点和符号，故从算术表达式中分离出操作数需要特殊处理，而操作符的难点在于减号和负号的区分，而在预处理阶段已经解决了这样问题，故提取操作符还是相对简单的，伪代码如下：

while 扫描未结束

chr=当前扫描字符

num初始化为空string

if chr是一个数，或者是‘！’

while chr是一个数，或者chr==‘！’，或者chr==‘.’

将chr添加到num的尾部

chr指向下一个字符

num为一个完整的擦作数

else

chr为一个操作符

（执行核心算法）

chr指向下一个字符

上述伪代码中，每次循环可以保证提取出操作数或操作符，甚至可以同时得到操作数和操作符，然后利用2.1中核心算法对他们进行处理。在扫描完一次后，就可以直接得到后缀表达式。

**3.对后缀表达式进行计算**

3.1 主要计算过程

在得到后缀表达式后，整个计算过程就相当于完成了3/4，因为后缀表达式计算过程相对简单，后缀表达式是一个string类的数组，从头到尾进行遍历并按如下伪代码执行即可得出结果：

for (遍历后缀表达式)：

str为当前扫描到的后缀表达式中的元素

if (str是一个数)

str入栈;

else str是一个操作符

栈内元素出栈并赋值给a1;

栈内元素出栈并赋值给a2；

将a1、a2转化为浮点型（利用函数float c\_to\_n（string a），具体实现见3.2.1）

switch (str[0])//(貌似源代码更简单易懂，大部分应用了源代码)

{

case '+':

reslt = a2 + a1;

break;

case '-':

reslt = a2 - a1;

break;

case '\*':

reslt = a2\*a1;

break;

case '/':

if (fabs(a1) <= 1e-6)//判断除数是否为0

输出“ERROR！除另错误”

返回

else

reslt = a2 / a1;

break;

case '%':

if (!is\_int(a1) || !is\_int(a2))//a1、a2是不是整数（is\_int详情见3.2.3）

输出错误"ERROR!求余运算操作数中出现小数！"

返回

else

reslt = (int)a2 % (int)a1;

将计算结果转换为string类型（利用string n\_to\_c(float a)，详见3.2.2）

计算结果入栈

计算结果出栈

输出计算结果

}

3.2 计算过程中用用到的函数的解析

在计算过程中，涉及string转float和float转string的过程，主要通过函数：

string n\_to\_c(float a)、float c\_to\_n(string s)来实现的，另外在求余运算中需要判断两个操作数是不是都是整数，通过int is\_int(float n)实现的，下面为三个函数具体实现过程

3.2.1 float c\_to\_n（string a）的实现

该函数主要讲一个string类型的数据转换为float类型的数据并返回。主要利用检索小数点的位置，然后通过10进制数的特征进行转换，例如“123.45”中，小数点位置为3，则123.45=1\*10^2+2\*10^1+3\*10^0+4\*10^-1+5\*10^-2,伪代码如下：

if s是一个负数

sym = -1;

将负号从string中删去

查找”.”的位置并赋值给pos;

if 没找到

pos = s的长度;

else

将”.”从string中删去 for (int i = 0; i < s.length(); i++)

for (遍历s)

ans += (s[i] - '0')\*pow(10, pos - 1 - i);

返回 ans \* sym;

3.2.2 string n\_to\_c(float a)的实现

该函数将一个浮点型数转换为string型字符串并返回，可以直接利用c++中现有的库函数\_gcvt\_s函数来实现，\_gcvt\_s(str, 20, a, 4)可以将浮点数a转化为字符数组并赋值给str，然后利用string类中的assign方法将str转化为一个string类型。

3.2.3int is\_int(float n)的实现

该函数用于判断float类型的n是不是一个整数，是则返回1，否则返回0；仅仅简单的利用if ((int)a - a == 0)就可以实现，如果n为整数，(int)a - a 等于0，否则(int)a - a 不等于0。

**参考文献**

[1] 在线文献 作者. mplus 文题 [字符串与数值相互转换的函数]． <http://blog.csdn.net/mplus/article/details/6069657> 日期. 2010-12-11

[2] 在线文献 作者. liuhuiyi 文题 [利用堆栈求解算术表达式]． <http://blog.csdn.net/liuhuiyi/article/details/8433203> 日期. 2012-12-25

[3] 在线文献 作者. 洗尘斋 文题 [C++string类常用函数]． <http://www.cppblog.com/lmlf001/archive/2006/04/19/5883.html> 日期. 2012-12-25

[4] 专著 作者. [爱尔兰]Paul Kelly 苏小红 书名[双语版 C++程序设计].电子工业出版社 2010年6月第1次印刷