# 线性表的实现与应用

# 蒲毅1140310323

实验的内容与要求：

内容： 算术表达式求值，能够完成整数、实数等的四则运算及求模运算。

要求：输入中缀表达式，先转换成后缀表达式，输出后缀表达式，然后对后缀表达式求值。

要写出实验报告，按科研论文形式写，具体格式自行查询。注意表达式的特殊情况的处理。

线性表的实现与应

中缀表达式转后缀表达式并求值

将问题分解为从中缀表达式转到后缀表达式和求后缀表达式的值两个问题分别解决

1. 中缀表达式的算法设计思路：

为了简化不必要的操作，我们将表达式的符号，数字，括号之间都以空格加以分隔，这样可以简化输入（如何判断输入是否合法不是本文讨论的问题），使得每一个有效的符号，数字和括号成为单独的一个字符串，我们将它称为元素。这样我们就可以从表达式的最左端开始逐个元素的向表达式的右边进行遍历。我们还需要一个栈来保存未被加入后缀表达式的运算符，称为S。最后的后缀表达式保存在线性表 E 中。

1. 开始扫描字符串。
2. 如果当前元素是数字，则把数字加入当前的后缀表达式的末尾，转至步骤 (6) 。
3. 如果当前元素是 “(” ， 那么将当前元素压入S。转至步骤 (6) 。
4. 如果当前元素是 “)” ， 那么一次吧S中的运算符pop出来并且加到E中，直到pop 出的元素为 “(“ ， “(” 不需要加入E中。转至步骤 (6) 。
5. 如果当前元素为运算符，当其优先级高于 “(“ ，以外的栈顶运算符是，直接压入S，否则从栈顶开始，依次弹出比当前处理的运算符优先级高和优先级相等的运算符，直到一个栈顶元素为 “（” 或者优先级小于当前元素时，然后将其加入S。转至步骤 (6) 。
6. 是否扫描到表达式的末尾，是则 pop 出S 中的元素依次加入E。否则读入下一个元素，转至步骤(2) 。
7. 后缀表达式的求值：

建立一个栈S 。从左到右读表达式，如果读到操作数就将它压入栈S中，如果读到n元运算符(即需要参数个数为n的运算符)则取出由栈顶向下的n项按操作符运算，再将运算的结果代替原栈顶的n项，压入栈S中 。如果后缀表达式未读完，则重复上面过程，最后输出栈顶的数值则为结束。

经过分析，中缀表达式的转化和求值是可以合并在一起的。

C++ code， compiled with G++ 4.8.1

#include <cstdio>

#include <iostream>

#include <cstring>

#include <algorithm>

#include <cmath>

#include <vector>

#include <map>

#include <string>

using namespace std;

struct Node

{

int tp;

double di;

char dc;

Node \*nex;

Node():nex(NULL){}

Node(int tp, double di, int dc):tp(tp),di(di),dc(dc),nex(NULL){}

void show()

{

if(tp == 1) printf("%.2f ", di);

else printf("%c ", dc);

}

};

Node \*head;

char stk[1000];

int stkk[1000];

double stkf[1000];

int top = 0;

void inop(char ch, Node \*&now)

{

int nowrank;

if(ch == '+' || ch == '-') nowrank = 1;

else nowrank = 2;

while(top && stkk[top] >= nowrank && stkk[top] < 10)

{

now->nex = new Node(0,0,stk[top]);

now = now->nex;

top--;

}

stk[++top] = ch;

stkk[top] = nowrank;

}

void inbr(char ch, Node \*&now)

{

int nowrank = 10;

if(ch == '(')

{

stk[++top] = ch;

stkk[top] = nowrank;

}

else

{

while(top && stkk[top] < nowrank)

{

now->nex = new Node(0,0,stk[top]);

now = now->nex;

top--;

}

top--;

}

}

void stkout(Node \*&now)

{

cout << top << endl;

while(top)

{

now->nex = new Node(0,0,stk[top--]);

now = now->nex;

}

}

double cal(double a, double b, char ch)

{

if(ch == '+') return a + b;

else if(ch == '-') return a - b;

else if(ch == '\*') return a \* b;

else if(ch == '/') return a / b;

else return (int)a % (int)b;

}

int main()

{

printf("input the nifix expression , seperate integers , brackets and operators by a blank space,\ninput shold be end with a # in a single line\n");

char s[20];

int len;

head = new Node();

Node \*now = head;

//输入的括号， 数字， 符号用 =空格隔开， 所以以字符串的方式读入数据

while(scanf("%s", s))

{

// for(int i = 0; i <= top; i++)

// {

// printf("%c ", stk[i]);

// }

// printf("\n");

if(s[0] == '#') break; // 输入以单独的一行 # 结束

len = strlen(s);

if(len == 1)// 如果字符串的长度为1， 那么可能是符号 也可能是数字

{

//分别判断是否是括号 或者 运算符

if(s[0] == '+' || s[0] == '-' || s[0] == '\*' || s[0] == '/' || s[0] == '%') {inop(s[0], now);continue;}

else if(s[0] == '(' || s[0] == ')') {inbr(s[0], now);continue;}

}

//如果是数字， 就在保存的后缀表达式最后加上这个数字

now->nex = new Node(1,atof(s),0);

now = now->nex;

}

// cout << "!!!!!!" <<endl;

stkout(now);// 处理完整个表达式， 将未出栈的符号 依次 出栈

now = head->nex;

top = 0;

while(now != NULL)// 打印后缀表达式， 顺便求出 后缀表达式的值

{

now->show();

// 求后缀表达式的值 用栈来实现

if(now->tp == 1) stkf[++top] = now->di;

else

{

double tmp = cal(stkf[top-1], stkf[top], now->dc);

top -= 2;

stkf[++top] = tmp;

}

now = now->nex;

}

printf("\nans = %.4f\n", stkf[1]); //最后的栈顶元素就是后缀表达式的值

return 0;

}

测试样例：

|  |  |  |
| --- | --- | --- |
| 中缀表达式 | 后缀表达式 | 值 |
| ( 1 + 2 ) \* 3 / 4 | 1.00 2.00 + 3.00 \* 4.00 | 2.2500 |
| 1 + 2 \* 3 / 4 | 1.00 2.00 3.00 \* 4.00 / + | 2.5000 |
| 1 + 2 \* ( 3 / 4 ) | 1.00 2.00 3.00 4.00 / \* + | 2.5000 |