表达式计算

程序目的

用堆栈实现将中缀表达式转为波兰表达式和逆波兰表达式

实现思想

建立两个堆栈，一个用于储存表达式中的运算符，另一个储存数字，逐个扫描表达式，将相应的字符存入相应的堆栈

变量含义

stackNum:储存数字以及最终表达式的堆栈

stackSymbol:储存运算符的堆栈

infixNotation:中缀表达式

changeExpression():将中缀表达式转为前缀表达式并存入stackNum栈中

printPolish():输出波兰表达式

printRePolish():输出逆波兰表达式

伪代码

(1) 初始化两个栈：运算符栈stackSymbol和储存数字的栈stackNum；

(2) 从右至左扫描中缀表达式；

(3) 遇到操作数时，将其压入stackNum；

(4) 遇到运算符时，比较其与stackSymbol栈顶运算符的优先级：

(4-1) 如果stackSymbol为空，或栈顶运算符为右括号“)”，则直接将此运算符入栈；

(4-2) 否则，若优先级比栈顶运算符的较高或相等，也将运算符压入stackSymbol；

(4-3) 否则，将stackSymbol栈顶的运算符弹出并压入到stackNum中，再次转到(4-1)与stackSymbol中新的栈顶运算符相比较；

(5) 遇到括号时：

(5-1) 如果是右括号“)”，则直接压入stackSymbol；

(5-2) 如果是左括号“(”，则依次弹出stackSymbol栈顶的运算符，并压入stackNum，直到遇到右括号为止，此时将这一对括号丢弃；

(6) 重复步骤(2)至(5)，直到表达式的最左边；

(7) 将S1中剩余的运算符依次弹出并压入stackNum；

(8) 依次弹出stackNum中的元素并输出，结果即为中缀表达式对应的前缀表达式。

核心代码

void Expression::changeExpression(){

int j=0,k=0;//定义变量j：stackNum堆栈下标，k：stackSymbol堆栈下标

for (long i=infixNotation.length()-1; i>=0; i--) {//循环扫描表达式

switch (infixNotation[i]) {

case '+':

if ('\*'==stackSymbol[k-1]||'/'==stackSymbol[k-1]) {//若stackSymbol堆栈最上层运算符优先级大于＋，将优先级大的转移到stackNum堆栈

do{

stackNum[j]=stackSymbol[k-1];

stackSymbol[k-1]='\0';

j++;

k--;

}while('\*'==stackSymbol[k-1]||'/'==stackSymbol[k-1]);

}else{//将运算符压入stackSymbol堆栈

stackSymbol[k]=infixNotation[i];

k++;

}

break;

case '-':

if ('\*'==stackSymbol[k-1]||'/'==stackSymbol[k-1]) {

do{

stackNum[j]=stackSymbol[k-1];

stackSymbol[k-1]='\0';

j++;

k--;

}while('\*'==stackSymbol[k-1]||'/'==stackSymbol[k-1]);

}else{

stackSymbol[k]=infixNotation[i];

k++;

}

break;

case '\*'://将运算符压入stackSymbol堆栈

stackSymbol[k]=infixNotation[i];

k++;

break;

case '/'://将运算符压入stackSymbol堆栈

stackSymbol[k]=infixNotation[i];

k++;

break;

case '('://若遇到（，将stackSymbol中运算符弹出压入stackNum，直到遇到）

while (')'!=stackSymbol[k-1]) {

stackNum[j]=stackSymbol[k-1];

stackSymbol[k-1]='\0';

j++;

k--;

}

stackSymbol[k-1]='\0';

k--;

break;

case ')'://将运算符压入stackSymbol堆栈

stackSymbol[k]=infixNotation[i];

k++;

break;

default://将数字压入stackNum堆栈

stackNum[j]=infixNotation[i];

j++;

break;

}

}

for ( ; k>=0; k--) {//将stackSymbol的运算符全部弹出压入stackNum

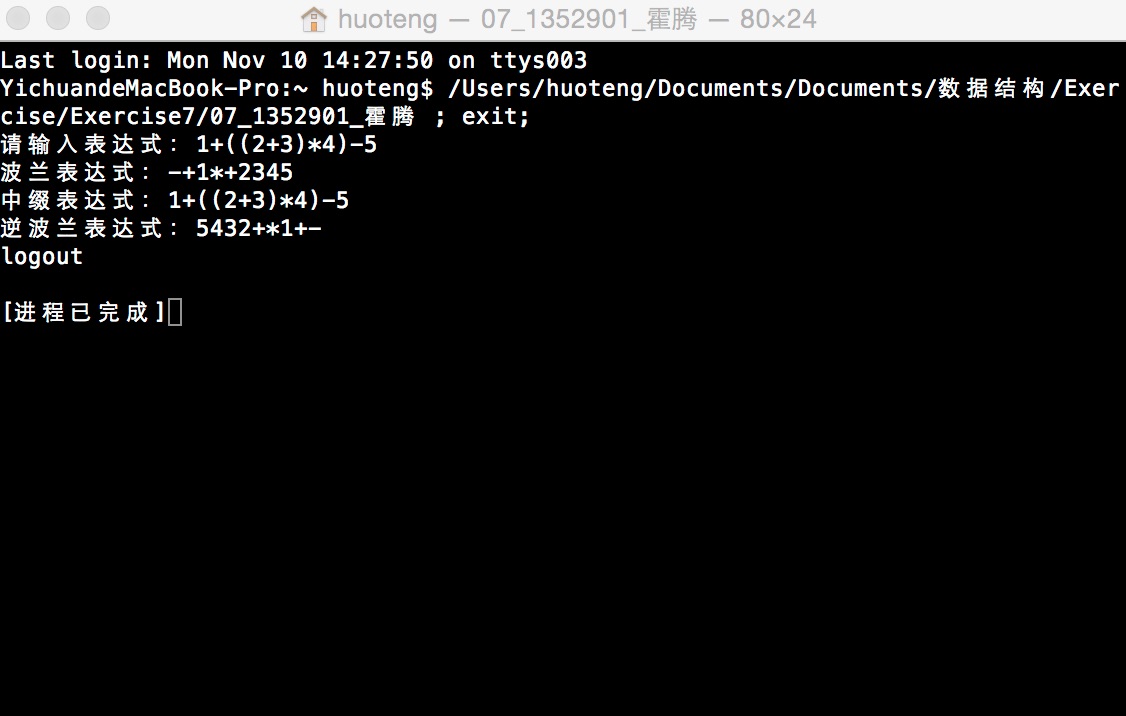
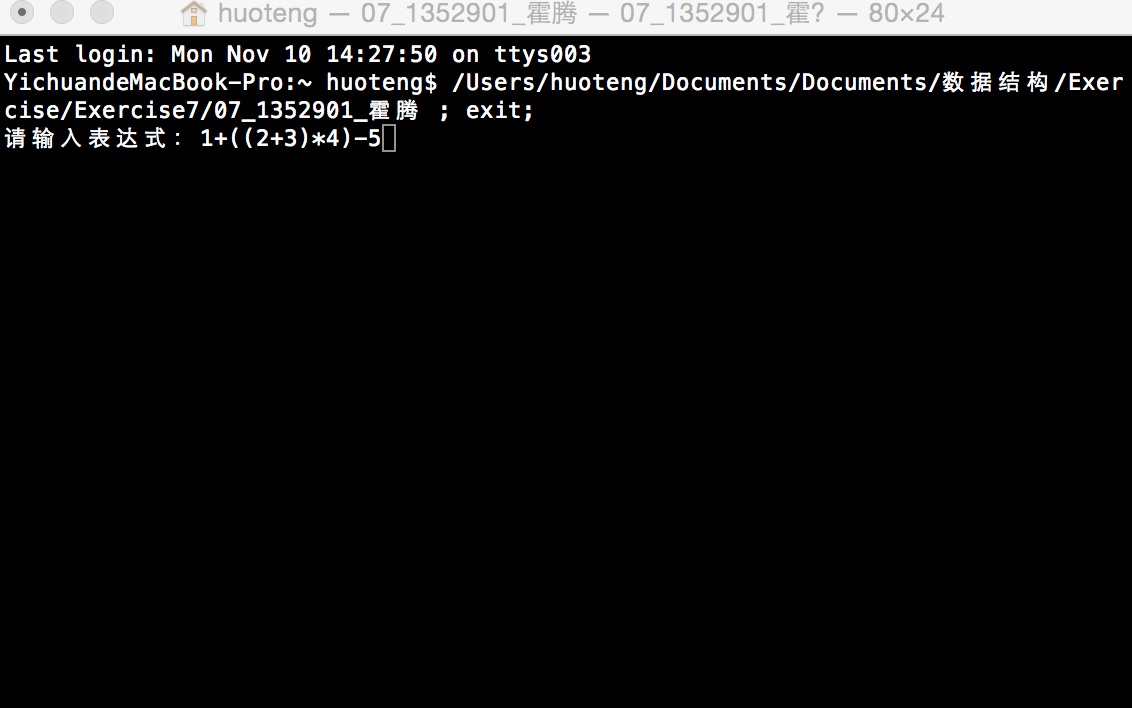
stackNum[j]=stackSymbol[k-1];

stackSymbol[k]='\0';

j++;

}

}

运行截图