实习报告：题目三

顺序表及其应用——栈和队列及应用&表达式求值

班级：17计算机一班 姓名：陈新朋 学号：1725111006 完成日期：2018.10.26

1. 需求分析

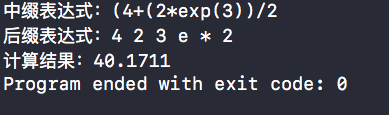
（1）以链栈进行中缀表达式到后缀表达式的转换。链栈中数据类型为char。

（2）输入表达式数据。

（3）实现链栈的各种基本操作，包括初始化链栈、插入节点和删除节点、读取栈顶元素、链栈的输出、判断栈空、读取栈长度、销毁栈等等。

（4）将中缀表达式转换成后缀表达式；再计算后缀表达式。

（5）测试数据及结果见图



（6）程序执行的命令为：

1）得到运算表达式；2）将运算表达式转换成后缀表达式；3）计算后缀表达式，输出结果

二、 概要设计

1. 设定栈的抽象数据类型定义：

ADT stack{

数据对象：D={a i |ai∈CharSet,i=1,2,…,n,n>=0}

数据关系：R1={<a i-1 ,a i >|a i-1 ,a i ∈D,i=2,…,n}

基本操作：

/\*\*

\* @brief 初始化一个栈

\* 由于有头结点，故将头结点的指针域置为零

\* -top为栈底的地址，\*top为现在栈顶的位置

\*/

void InitStack(Node \*top)

/\*\*

\* @brief 打印一个栈

\* -

\*/

void Display(Node \*top)

/\*\*

\* @brief 判断栈是否为空

\* 判断栈顶指针的next是否为空

\* @condition 已有一个栈

\* -top为栈顶指针

\* @return 栈空或非空

\*/

int StackEmpty(Node \*top)

/\*\*

\* @brief 进栈

\* 建立一个新节点，给它的数据赋值，并把它压入栈

\* @condition 已有一个栈

\* -top为栈顶指针

\* -p为新节点的指针

8 -e为入栈元素的值

\* -必须先执行p->next=top->next;再执行top->next=p;顺序不可颠倒

\* @return 进栈是否成功

\*/

int PushStack(Node \*top, char e)

/\*\*

\* @brief 出栈

\* @condition 已有一个栈，且栈不为空

\* -top为栈顶指针，e为出站元素的值

\* @return 出栈是否成功

\*/

int PopStack(Node \*top, char \*e)

/\*\*

\* @brief 取栈顶元素

\* 返回栈顶指针指向的元素数据值

\* @condition 已有一个栈，且栈不为空

\* -top为栈顶指针

\* -p为新节点的指针

\* @return 是否成功返回元素数据值

\*/

int GetTop(Node \*top, char \*e)

/\*\*

\* @brief 求表长

\* 返回栈中元素个数

\* @condition 已有一个栈

\* -top为栈顶指针

\* -p为节点指针

\* -i为计数变量

\* @return 返回栈的长度

\*/

int StackLen(Node \*top)

/\*\*

\* @brief 销毁栈

\* 释放栈空间

\* @condition 已有一个栈

\* -top为栈顶指针

\* -p、q均为节点指针，q用于释放空间

\*/

void DestroyStack(Node \*top)

}}ADT list

三、 详细设计

#include <iostream>

#include "Stack.h"

#include <string>

#include <math.h>

using namespace std;

typedef struct{

float data[50];

int top;

}sta;

string TranExp(char \*str, string exp);

float ComExp(string s);

int main() {

char str[]="(4+(2\*exp(3))/2";

//cout<<"输入一个运算表达式：";

char \*p=&str[0];

string e;

e=TranExp(p, e);

cout<<"中缀表达式："<<p<<endl;

cout<<"后缀表达式："<<e<<endl;

cout<<"计算结果："<<ComExp(e)<<endl;

return 0;

}

/\*\*

\* @brief 计算后缀表达式并得出结果

\* @condition 已有一个后缀表达式

\* -输出：后缀表达式值

\*/

float ComExp(string s)

{

sta rand;

rand.top=-1;

int i=0,value;

float x1,x2,result;

for(;i<s.length();i++)

{

if(s[i]!=' '&&s[i]>='0'&&s[i]<='9')

{

value=0;

while(s[i]!=' ')

{

value=10\*value+s[i]-'0';

i++;

}

rand.top+=1;

rand.data[rand.top]=value;

}

else{

switch(s[i])

{

case '+':

x1=rand.data[rand.top];

rand.top--;

x2=rand.data[rand.top];

rand.top--;

result=x1+x2;

rand.top++;

rand.data[rand.top]=result;

break;

case '-':

x1=rand.data[rand.top];

rand.top--;

x2=rand.data[rand.top];

rand.top--;

result=x2-x1;

rand.top+=1;

rand.data[rand.top]=result;

break;

case '\*':

x1=rand.data[rand.top];

rand.top--;

x2=rand.data[rand.top];

rand.top--;

result=x1\*x2;

rand.top++;

rand.data[rand.top]=result;

break;

case '/':

x1=rand.data[rand.top];

rand.top--;

x2=rand.data[rand.top];

rand.top--;

result=x2/x1;

rand.top++;

rand.data[rand.top]=result;

break;

case 'e':

x1=rand.data[rand.top];

result=exp(x1);

rand.data[rand.top]=result;

break;

case 'l':

x1=rand.data[rand.top];

result=log10(x1);

rand.data[rand.top]=result;

break;

case 's':

x1=rand.data[rand.top];

result=sqrt(x1);

rand.data[rand.top]=result;

break;

}

i++;

}

}

if(!(rand.top!=-1))

{

result=rand.data[rand.top];

rand.top--;

if(rand.top==-1)

return result;

else

{

cout<<"表达式错误"<<endl;

return result;

}

}

else

return result;

}

/\*\*

\* @brief 将中缀表达式转换成后缀表达式

\* -

\* @condition 已有一个运算表达式

\* -栈S用于存放运算符

\*/

string TranExp(char \*str, string exp)

{

Node S;

char ch,e = '\0';

int i=0,j=0;

InitStack(&S);

PushStack(&S, '#');

ch=str[i];

i++;

while(ch!='\0')

{

if(ch=='(')

PushStack(&S, ch);

else if(ch==')')

{

GetTop(&S,&e);

while(GetTop(&S,&e)&&e!='('&&e!='e'&&e!='l'&&e!='s')

{

PopStack(&S, &e);

exp+=e;

exp=exp+' ';

j++;

}

PopStack(&S, &e);

if(e=='e')

exp=exp+"e ";

else if(e=='l')

exp=exp+"l ";

else if(e=='s')

exp=exp+"s ";

}

else if(ch=='+'||ch=='-')

{

while(!StackEmpty(&S)&&GetTop(&S, &e)&&e!='('&&e!='e'&&e!='l'&&e!='s'&&e!='#')

{

PopStack(&S, &e);

exp+=e;

exp=exp+' ';

j++;

}

PushStack(&S, ch);

}

else if(ch=='\*'||ch=='/')

{

while(!StackEmpty(&S)&&GetTop(&S, &e)&&e=='/'&&e=='\*')

{

PopStack(&S, &e);

exp+=e;

exp=exp+' ';

j++;

}

PushStack(&S, ch);

}

else if(ch=='e')

{

PushStack(&S, ch);

while(ch!='(')

{

i++;

ch=str[i];

}

i++;

}

else if(ch=='l')

{

PushStack(&S, ch);

while(ch!='(')

{

i++;

ch=str[i];

}

i++;

}

else if(ch=='s')

{

PushStack(&S, ch);

while(ch!='(')

{

i++;

ch=str[i];

}

i++;

}

else if(ch==' '){}

else if(ch=='#')

{

while(!(GetTop(&S, &e)&&e=='#'))

{

PopStack(&S, &e);

exp+=e;

exp=exp+' ';

}

}

else

{

while(ch>='0'&&ch<='9')

{

exp+=ch;

j++;

ch=str[i];

i++;

}

i--;

exp=exp+' ';

j++;

}

ch=str[i];

i++;

}

return exp;

}

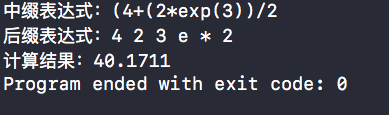
四、 调试分析

1．调试时遇到了各种bug，关键在于先做最简单的功能，再往上加功能。

2．本题中主要算法：TranExp和ComExp的时间复杂度均为 0（n\*n），

3．经验体会：使用断点进行debug，可以快速找到错误。

五、 测试结果



六、 附录

源程序文件名清单：

main.cpp//主程序

stack.cpp//源程序

stack.h//头文件