# 上机练习2

## 1

程序代码：

#include <stdio.h>

#include <malloc.h>

#define MaxLen 100

typedef struct node

{

char data;

struct node \*next;

}cnode;

cnode \*create(char s[])

{

int I=0;

cnode \*h,\*p,\*r;

while(s[I]!='\0')

{

p=(cnode\*)malloc(sizeof(cnode));

p->data=s[I];

p->next=NULL;

if(I==0)

{

h=p;

r=p;

}

else

{

r->next=p;

r=p;

}

I++;

}

return h;

}

int judge(cnode\*h)

{

char st[MaxLen];

int top=0;

cnode \*p=h;

while(p!=NULL)

{

st[top]=p->data;

top++;

p=p->next;

}

p=h;

while(p!=NULL)

{

top--;

if(p->data==st[top])

p=p->next;

else

break;

}

if(p==NULL)

return 1;

else

return 0;

}

void main()

{

char str[MaxLen];

cnode \*h;

printf("input a string:");

scanf("%s",str);

h=create(str);

if(judge(h)==1)

printf("%s is a right string\n",str);

else

printf("%s is a false string\n",str);

}

}

运行及展示：



## 2

程序代码：

#include <stdio.h>

#include <malloc.h>

#define MaxLen 100

typedef struct

{

char data[MaxLen];

int top;

}sqstacktp;

void InitStack(sqstacktp \*s)

{

s->top=0;

}

int StackEmpty(sqstacktp \*s)

{

if(s->top>0)

return 0;

else

return 1;

}

void Push(sqstacktp \*s,char c)

{

if(s->top==MaxLen)

printf("Overflow");

else

s->data[s->top++]=c;

}

char Pop(sqstacktp \*s)

{

if(s->top==0)

return '\0';

else

{

s->top--;

return s->data[s->top];

}

}

int Size(sqstacktp \*s)

{

return(s->top);

}

char Top(sqstacktp \*s)

{

if(s->top==0)

return '\0';

else

return (s->data[s->top-1]);

}

int judge(char \*str,sqstacktp \*s)

{

int i=0;

while(str[i]!='\0')

{

if(str[i]=='('||str[i]==')')

{

if(StackEmpty(s)||str[i]==Top(s))

Push(s,str[i]);

else

Pop(s);

}

i++;

}

i=0;

while(str[i]!='\0')

{

if(str[i]=='['||str[i]==']')

{

if(StackEmpty(s)||str[i]==Top(s))

Push(s,str[i]);

else

Pop(s);

}

i++;

}

i=0;

while(str[i]!='\0')

{

if(str[i]=='{'||str[i]=='}')

{

if(StackEmpty(s)||str[i]==Top(s))

Push(s,str[i]);

else

Pop(s);

}

i++;

}

if(StackEmpty(s))

return 1;

else

return 0;

}

void main()

{

char str[MaxLen];

sqstacktp \*s;

s=(sqstacktp\*)malloc(sizeof(sqstacktp));

InitStack(s);

printf("input a string:");

scanf("%s",str);

if(judge(str,s)==1)

printf("%s is a right match\n",str);

else

printf("%s is a false match\n",str);

}

运行及展示：



## 3

程序代码：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<math.h>

#define Stack\_Size 1000

char cmp[7][8]= {">><<<>>",">><<<>>",">>>><>>",">>>><>>","<<<<<=?",">>>>?>>","<<<<<?="};

typedef struct{//定义一个运算符栈

char Elem[Stack\_Size];

int top;

}Opter;

typedef struct{//定义一个操作数栈

double Elem[Stack\_Size];

int top;

}Opnd;

void InitOpter(Opter \*S){//初始化运算符栈

S->top=-1;

}

void InitOpnd(Opnd \*S){//初始化操作数栈

S->top=-1;

}

int PopOpter(Opter \*S)//弹出运算符栈

{

if(S->top==-1)

{

printf("运算符栈为空\n");

exit(10);

}

S->top--;

return 1;

}

int PopOpnd(Opnd \*S)

{

if(S->top==-1)

{

printf("运算符栈为空\n");

exit(11);

}

S->top--;

return 1;

}

int PushOpter(Opter\* S,char ch)

{

if(S->top==Stack\_Size-1)

{

printf("运算符栈满\n");

exit(12);

}

S->top++;

S->Elem[S->top]=ch;

return 1;

}

int PushOpnd(Opnd\* S,double ch)//入操作数栈

{

if(S->top==Stack\_Size-1)

{

printf("运算符栈满\n");

exit(13);

}

S->top++;

S->Elem[S->top]=ch;

return 1;

}

char GetOpter(Opter \*S)//获取运算符栈的栈顶元素

{

if(S->top==-1)

{

printf("运算符栈为空\n");

exit(17);

}

return S->Elem[S->top];

}

double GetOpnd(Opnd \*S)

{

if(S->top==-1)

{

printf("操作数栈为空\n");

exit(18);

}

return S->Elem[S->top];

}

double Calc(double a,double b,char opt)//计算函数，传入两个数以及一个运算符

{

double T; //T用于存放计算得出的结果

if(opt=='+') T=a+b;

if(opt=='-') T=a-b;

if(opt=='\*') T=a\*b;

if(opt=='/') //要防止发生除0错误

{

if(fabs(b)<0.00001)

{

printf("发生除0错误\n");

exit(15);

}

T=a/b;

}

printf("中间过程输出： %.2lf %c %.2lf = %.2lf\n",a,opt,b,T);

return T; //返回得到的结果

}

int change(char ch)

{

switch(ch)

{

case '+':

return 0;

case '-':

return 1;

case '\*':

return 2;

case '/':

return 3;

case '(':

return 4;

case ')':

return 5;

case '#':

return 6;

}

}

int Compare(char ch1,char ch2)

{

if(cmp[change(ch1)][change(ch2)]=='?'){

printf("输入表达式错误");

exit(16);

}

return cmp[change(ch1)][change(ch2)];

}

int Check(char \*S,int len)//检查函数，记得考虑输入带小数点的数字的情况

{

int i;

for(i=0;i<len;i++){

if(S[i]>='0'&&S[i]<='9')continue;

if(S[i]=='('||S[i]==')'||S[i]=='\*'||S[i]=='/'||S[i]=='+'||S[i]=='-'||S[i]=='.')continue;

return 0;

}

return 1;

}

int main()

{

char a[1000],b[1000]; //创建两个数组，a是用来存输入的表达式的，b是用来存操作数的

int i,j=0,k=0;

double x,y; //x,y是从操作数中取出的两个即将用于计算的数

double answer;

int len; //len为输入表达式的长度，通过strlen求得

Opter S; //创建一个运算符栈

Opnd N; //创建一个操作数栈

InitOpnd(&N); //初始化操作数栈

InitOpter(&S); //初始化运算符栈

PushOpter(&S,'#');

printf("输入表达式：\n");

scanf("%s",a);

len=strlen(a); //求输入的表达式的长度，并打印出来

printf("字符长度为%d\n",len);

if(Check(a,len)==0)

{

printf("输入中存在多余字符\n");

exit(19);

}

a[len]='#';

for(i=0;i<=len;i++) //遍历输入的表达式

{

if((a[i]>='0'&&a[i]<='9')||a[i]=='.')//如果为数字

{

b[k++]=a[i];//将数字存入数组b中，注意此时数字仍为字符

j=1;

continue; //在该循环下其余部分都不做了，直接进入下一次xunh

}

if(j)//条件成功即遇到了运算字符，将操作数压入操作数栈中

{

//此时数组b已经有了一个或者几个数字在里面，需要加一个'\0'使其成为字符串

//再通过atof函数使其由字符型变为双精度型，然后加入操作数栈中进行相应运算

b[k]='\0';

PushOpnd(&N,atof(b));//atof函数可以使char变为double

j=0;

k=0; //k置零为下一次计数做准备

}

switch(Compare(GetOpter(&S),a[i]))//比较运算符栈的栈顶运算符top和运算符a[i]的优先级

{ //底下的部分我们按照之前给的规则来写

case '<'://即top<a[i]，则将a[i]直接入栈Opter

PushOpter(&S,a[i]);

break;

case'=':

PopOpter(&S);

break;

case'>':

//当为‘>'的情况，即需要进行运算，先取操作数栈中最上面的两个元素

y=GetOpnd(&N),PopOpnd(&N);

x=GetOpnd(&N),PopOpnd(&N);

//然后将计算结果压入操作数栈中

PushOpnd(&N,Calc(x,y,GetOpter(&S)));

//已经用过的运算符就废掉了，弹出！！！

PopOpter(&S);

i--;//这句是为了重新把反括号入栈，使之与左括号配对，否则会造成左括号多余出来

break;

}

}

answer=GetOpnd(&N);//最终操作数栈中的数就是我们想要的结果

printf("最终结果为%.2lf",answer);

return 0;

}

运行结果及展示：

