实验三

#include<string.h>

#include<ctype.h>

#include<malloc.h> /\* malloc()等 \*/

#include<limits.h> /\* INT\_MAX等 \*/

#include<stdio.h> /\* EOF(=^Z或F6),NULL \*/

#include<stdlib.h> /\* atoi() \*/

#include<io.h> /\* eof() \*/

#include<math.h> /\* floor(),ceil(),abs() \*/

#include<process.h> /\* exit() \*/

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASIBLE -1

typedef int Status;

#define LIST\_INIT\_SIZE 100

#define LISTINCREMRNT 10

typedef double ElemType;

typedef struct

{

ElemType \*base; //储存空间基地址

ElemType \*top; // 记录当前链表长度

int stacksize; //链表规模

} stack;

Status InitStack(stack \*S)

{

(\*S).base = (ElemType\*)malloc(LIST\_INIT\_SIZE\*sizeof(ElemType));

if(!(\*S).base)

{

exit(OVERFLOW);

}

(\*S).top = (\*S).base;

(\*S).stacksize = LIST\_INIT\_SIZE;

return OK;

}

Status DestroyStack(stack \*S)

{

free((\*S).base);

(\*S).base=NULL;

(\*S).top=NULL;

(\*S).stacksize=0;

return OK;

}

Status ClearStack(stack \*S)

{ /\* 把S置为空栈 \*/

(\*S).top=(\*S).base;

return OK;

}

Status StackEmpty(stack S)

{ /\* 若栈S为空栈，则返回TRUE，否则返回FALSE \*/

if(S.top==S.base)

return TRUE;

else

return FALSE;

}

int StackLength(stack S)

{

return S.top-S.base;

}

Status GetTop(stack S,ElemType \*e)

{ /\* 若栈不空，则用e返回S的栈顶元素，并返回OK；否则返回ERROR \*/

if(S.top>S.base)

{

\*e=\*(S.top-1);

return OK;

}

else

return ERROR;

}

Status Push(stack \*S, ElemType e)

{

if((\*S).top - (\*S).base >= (\*S).stacksize)

{

(\*S).base = (ElemType\*)realloc((\*S).base, ((\*S).stacksize + LISTINCREMRNT) \* sizeof(ElemType));

if(!(\*S).base)

exit(OVERFLOW);

(\*S).top = (\*S).base + (\*S).stacksize;

(\*S).stacksize += LISTINCREMRNT;

}

\*(S->top) = e;

(S->top)++;

}

Status Pop(stack \*S,ElemType \*e)

{

if((\*S).top==(\*S).base)

return ERROR;

(\*S).top--;

\*e = \*((\*S).top);

}

Status ListTraverse(stack S)

{

ElemType \*e = S.base;

while (e < S.top)

{

printf("%d \n", \*e++);

}

return OK;

}

void f(stack \*S,char str[])

{

int i=0;

double number,e,d;

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

{

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

Push(S,str[i]-48);

switch(str[i])

{

case '+':

Pop(S,&e);

Pop(S,&d);

Push(S,d+e);

break;

case '-':

Pop(S,&e);

Pop(S,&d);

Push(S,d-e);

break;

case '\*':

Pop(S,&e);

Pop(S,&d);

Push(S,d\*e);

break;

case '/':

Pop(S,&e);

Pop(S,&d);

if(e == 0)

{

printf("输入出错，分母为零！\n");

break;

}

Push(S,d/e);

break;

}

i++;

}

Pop(S,&e);

printf("计算结果为：%lf",e);

}

int main()

{

char str[100];

stack S;

InitStack(&S);

printf("请输入数据,每个数据之间用空格隔开:");

gets(str);

f(&S,str);

return 0;

}

计算后缀表达式的值，需要建立两个栈对数进行压栈出栈的操作，同时加以运算符的运算，当读到数字是，将数字压栈，读到运算符时，存放数字的栈弹出两个数字进行运算，之后将结果在进行压栈，以此类推直至输入数据读完。实验操作时，由于不懂后缀表达式怎么化为正常表达式，思考了半天，同时运算符的优先级也尤其需要注意。