实验三 路凯 201930310034

#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 int 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 convention()

{

stack s;

unsigned n; /\* 非负整数 \*/

ElemType e;

InitStack(&s); /\* 初始化栈 \*/

printf("n(>=0)=");

scanf("%u", &n);

while (n)

{

Push(&s, n % 8);

n = n / 8;

}

while (!StackEmpty(s))

{

Pop(&s, &e);

printf("%d", e);

}

}

Status calculate(stack S)

{

ElemType e1 = 0,e2 = 0, i, j;

ElemType temp = 0;

char P[100];

InitStack(&S);

while ((temp = getchar()) != '\n')

{

i = 0;

P[i] = temp;

i++;

}

for(j=0;j<=i;j++)

{

if(P[j]>='0'&&P[j]<='9')

Push(&S,P[j]-47);

else

{

Pop(&S, &e1);

Pop(&S, &e1);

switch (P[j])

{

case '+': Push(&S, e2 + e1); break;

case '-': Push(&S, e2 - e1); break;

case '\*': Push(&S, e2 \* e1); break;

case '/': Push(&S, e2 / e1); break;

}

}

}

Pop(&S, &e1);

return e1;

}void main()

{

stack S1, S2;

ElemType ret;

printf("%d", calculate(S1, S2));

}

x 2020 / 10 / 28 20:33 : 32

Status calculate(stack S1, stack S2)

{

ElemType e, i, j;

ElemType temp = 0;

char P[100];

InitStack(&S1);

InitStack(&S2);

while ((temp = getchar()) != '\n')

{

i = 0;

P[i] = temp;

i++;

}

/\*for(j=0;j<=i;j++)

{

if(P[j]>='0'&&P[j]<='9')

Push(&S1,P[j]-47);

else

Push(&S2,P[j]);

}\*/

for (j = 0; j <= i; j++)

{

if (P[j] >= '0'&&P[j] <= '9')

Push(&S1, P[j] - 47);

}

}

把数组中的数据压入栈S中，如果是数字就压栈；如果是符号，就把栈里面的数字弹出来，然后进行相应的加减乘除，并把计算结果压栈

栈顶元素就是最后的计算结果，最后弹出来并返回。程序的实现关键在于压栈和弹栈的时间，弹数字的时候要减去47，因为数字是以ASCII码形式存放。