数据结构实验三

张笑辰

机1901

201930310107

#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,&e2);

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;

}int main()

{

stack S1, S2;

ElemType ret;

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

return 0;

}

实验总结：刚开始没有按照乘除顺序进行运算，然后用了一个栈进行储存运算，避免了这个问题。每个数乘一次，两数相乘的结果又储存起来，依次进行计算。