## 栈的相关操作 实验报告 2018204506 高恺元 18软工创新

实验内容：实现栈的创建及其初始化、判断栈空、栈的置空、返回栈顶元素、弹出栈顶元素、计算栈的长度、利用栈的相关操作实现进制转换等操作。

源码：

#include <stdio.h>

#include <stdlib.h>

#define MAXSIZE 1024

#define SIZEPLUS 20

#define OK 1

#define OVERFLOW 0

#define ERROR 0

//栈的表示

typedef struct{

int \*base;

int \*top;

int stacksize;//当前已分配

}Stack;

int createStack(Stack \*S){//创建一个空栈S

(\*S).base = (int \*)malloc(MAXSIZE \* sizeof(int));

if(!(\*S).base){

exit(OVERFLOW);//储存分配失败

}

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

(\*S).stacksize = MAXSIZE;

return OK;

}

int emptyStack(Stack S){//若栈空返回ok，反之返回error

if(S.top == S.base){

return OK;

}

else {

return ERROR;

}

}

int clearStack(Stack \*S){//把S置为空栈

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

return OK;

}

int getTop(Stack S){//若栈不空，用e返回S的栈顶元素，并返回ok；反之返回error

if(S.top == S.base){

return ERROR;

}

return \*(S.top - 1);

}

void Push(Stack \*S,int e){

if((\*S).stacksize<=MAXSIZE)

{

\*(S->top)=e;

(\*S).top++;}

else

printf("空间不足\n");

}

int Pop(Stack \*S, int \*e){//若栈非空，删除S栈顶元素，并用e返回其值

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

return ERROR;

}

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

S->top--;

return OK;

}

int StackTraverse(Stack S){

if(S.top == S.base){

return ERROR;

}

while(S.base < S.top){

printf("%d ",\*(S.base++));

}

printf("\n");

return OK;

}

int StackLength(Stack S){//栈的长度

if(S.top == S.base){

return 0;

}

return S.top - S.base;

}

//进制转换

int convertion(Stack \*S){

int e;

createStack(S);

int x,r;

printf("请输入十进制数及要转换为的进制：");

scanf("%d%d",&x,&r);

while(x){

Push(S ,x % r);

x=x/r;

}

while(!emptyStack(\*S)){

Pop(S,&e);

if(r==16){

if(e==10){

printf("A");

}else if(e==11){

printf("B");

}else if(e==12){

printf("C");

}else if(e==13){

printf("D");

}else if(e==14){

printf("E");

}else if(e==15){

printf("F");

}else{

printf("%d",e);

}

}

else{

printf("%d",e);

}

}

return OK;

}

int main(){

while(1){

system("cls");

printf("\t\t----------------------------------\n");

printf("\t\t 输入对应数字选择操作 \n");

printf("\t\t 1.初始化栈 2.元素入栈 \n");

printf("\t\t 3.检查栈空 4.把S置为空栈 \n");

printf("\t\t 5.弹出栈顶元素 6.取栈顶元素 \n");

printf("\t\t 7.输出栈中元素 8.求栈的长度 \n");

printf("\t\t 9.进制转换 0.退出 \n");

printf("\t\t----------------------------------\n");

Stack stackA;

int choose;

int e,q,n,i;

int popElem=NULL;

printf("您的选择：");

scanf("%d",&choose);

switch(choose){

case 1:

createStack(&stackA);

printf("初始化成功！\n");

system("pause");

break;

case 2:{

int ams,sds;

printf("请输入数列个数：");

scanf("%d",&ams);

if(ams>20){

printf("超限。");

break;

}

for(n=0;n<ams;n++){

scanf("%d",&sds);

Push(&stackA,sds);

}

printf("已入栈。\n");

system("pause");

}

break;

case 4:

clearStack(&stackA);

printf("已置空。\n");

system("pause");

break;

case 3:

i = emptyStack(stackA);

if(i==1){

printf("是空栈。\n");

}

else{

printf("不是空栈。\n");

}

system("pause");

break;

case 5:

Pop(&stackA,&popElem);

printf("%d",popElem);

system("pause");

break;

case 6:

e=getTop(stackA);

printf("栈顶元素是：%d\n",e);

system("pause");

break;

case 7:

StackTraverse(stackA);

system("pause");

break;

case 8:

q=StackLength(stackA);

printf("栈的长度为：%d\n",q);

system("pause");

break;

case 9:

convertion(&stackA);

system("pause");

break;

case 0:

exit(0);

break;

default:

printf("非法输入。");

system("pause");

}

}

return 0;

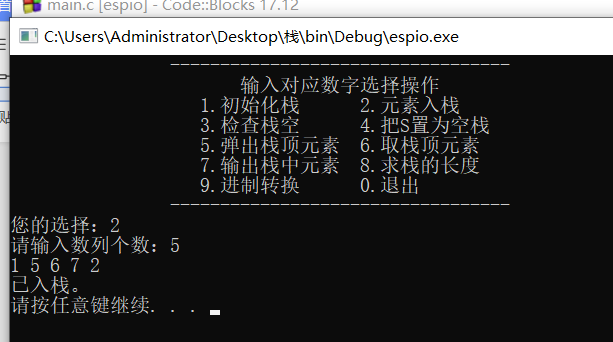
}

运行截图：

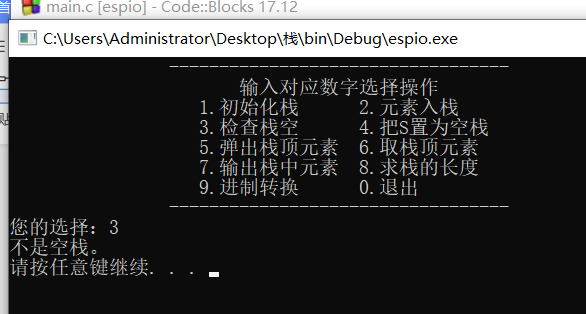
栈的初始化：



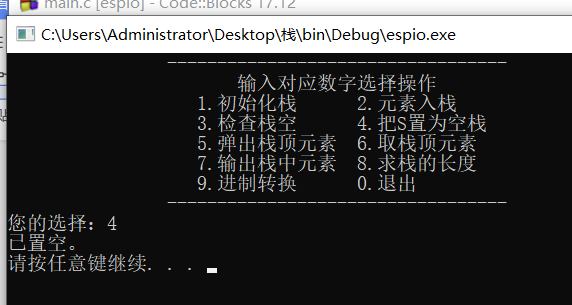
元素入栈：



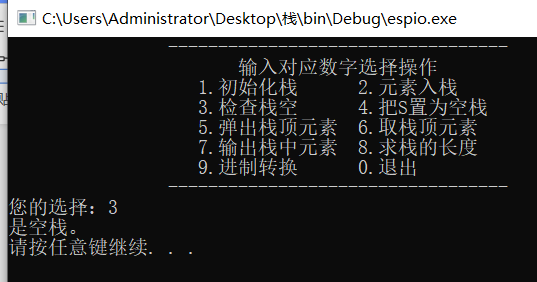
检查栈空：



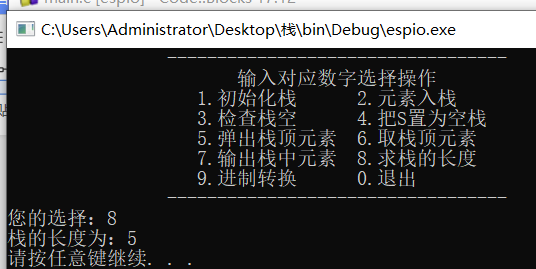
栈的置空：



置空后再次检查栈空：



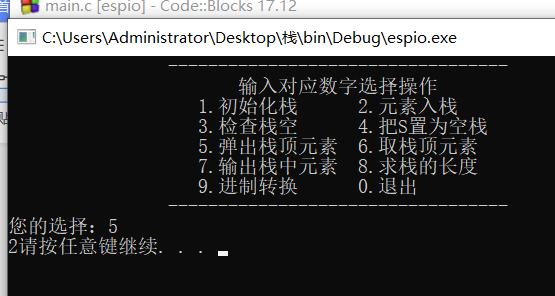
重新进行元素入栈后，计算栈的长度：



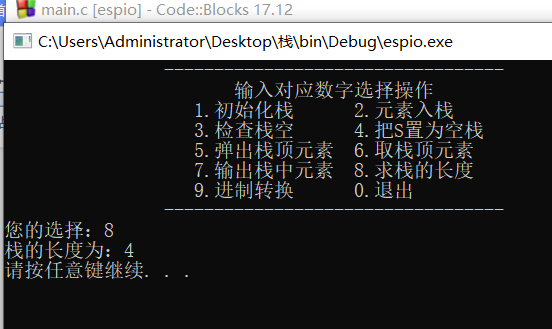
返回栈顶元素（不把它从此栈中删除）：



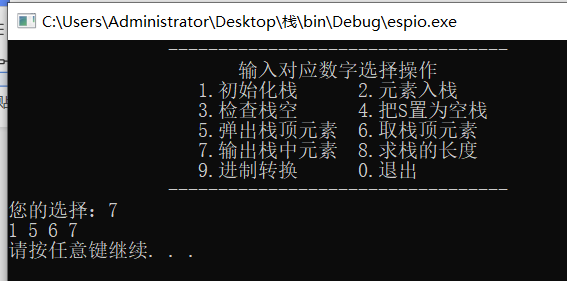
弹出栈顶元素（把它从此栈中删除）：



弹出栈顶元素后，再次计算栈的长度：

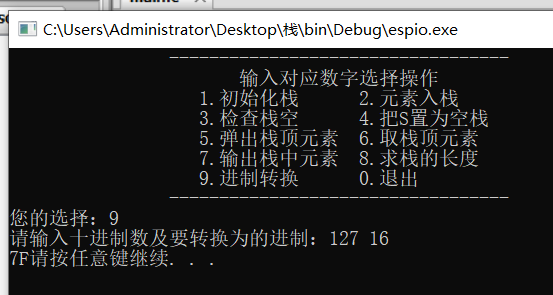


输出栈中元素：

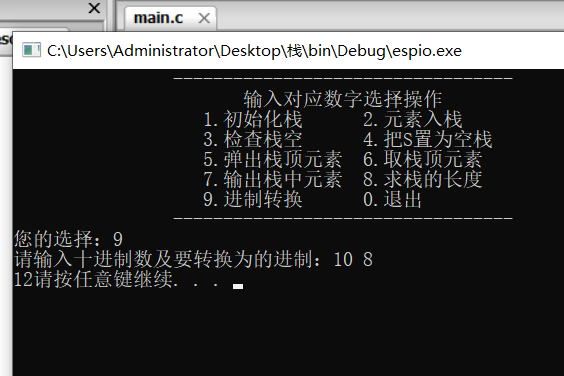


## 进制转换：

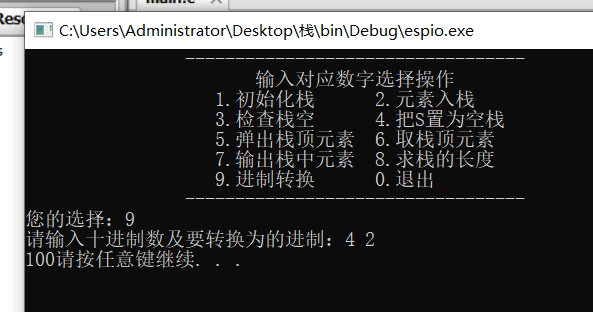
十进制转十六进制：



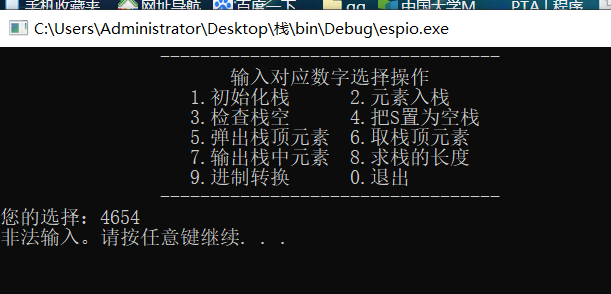
十进制转八进制：



十进制转二进制：



对于非法输入的反馈：



退出程序：

