计算机与信息工程学院实验报告

••••••••••••••••••••••••••••••••• 密 ••••••••••••••••••••••••••••••••• 封 ••••••••••••••••••••••••••••••••• 线 •••••••••••••••••••••••••••••••••

姓名：\_杨佳森 学号：2112080106\_ 专业：数据科学与大数据技术\_年级：\_2021\_

课程： 数据结构 主讲教师：\_刘成\_ 辅导教师：\_\_无\_\_\_\_\_

实验时间：2023\_年 3\_月 28\_日 下\_午\_17\_时至\_19\_时，实验地点\_计算机大楼213\_

实验题目： 顺序栈的基本操作

实验目的： 通过该实验，让学生掌握栈的相关基本概念，认识栈是插入和删除集中在一端进行的线性结构，掌握栈的“先入后出”操作特点。栈在进行各类操作时，栈底指针固定不动，掌握栈空、栈满的判断条件。

实验环境（硬件和软件） PC和vscode

实验内容：

用顺序存储结构，实现教材定义的栈的基本操作，提供数制转换功能，将输入的十进制整数转换成二进制。

实验步骤：

1、初始化

void initStack(stack &s) {

s.base = new int[MaxSize];

if(!s.base) exit(0);//内存分配失败，结束

s.top = s.base;//初始的时候栈的top与base相等

s.stackSize = MaxSize;//栈的最大容量

cout << "初始化成功！" << endl;

}

2、销毁栈

void DestoryStack(stack &s) {

if(s.base) {

delete s.base;

s.stackSize = 0;

s.base = NULL;

s.top = NULL;

cout << "成功销毁！" << endl;

}

else cout << "栈已经被销毁，无需销毁！" << endl;

}

3、将栈置空

void CleanStack(stack &s) {

if(s.base) {

s.top = s.base;

cout << "成功清空！" << endl;

}

else cout << "栈已经被销毁，无需清空！" << endl;

}

4、判断栈是否为空栈

int isEmpty(stack s) {

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

cout << "顺序栈为空！" << endl;

return 0;

}

cout << "顺序栈不为空！" << endl;

return 0;

}

5、返回栈的长度

int stackLength(stack s) {

cout << "栈的长度为：" << s.top - s.base << endl;

return s.top - s.base;

}

6、求栈顶元素

int top(stack s) {

if(isEmpty(s)) {

cout << "栈为空，没有栈顶元素！" << endl;

return 0;

}

else {

int e;

s.top--;

e = \*(s.top);

cout << "栈顶元素为：" << e << endl;

return 0;

}

}

7、插入元素，并使其成为栈顶元素

void push(stack &s,int e) {

if((s.top-s.base) == s.stackSize) cout << "栈满了，无法添加新元素" << endl;

else {

\*(s.top) = e;

s.top++;

cout << "添加成功" << endl;

}

}

8、删除栈顶元素，并返回其值

void pop(stack &s) {

int e;

if(isEmpty(s)) cout << "栈为空，无法弹出" << endl;

else {

s.top--;

e = \*(s.top);

cout << "成功弹出: " << e << endl;

}

}

9、输出栈内元素

void traverse(stack s) {

int length = stackLength(s);

if(length>0) {

cout << "顺序栈的元素为: ";

for(int i = 0; i < length; i++) cout << s.base[i] << " ";

cout << endl;

}

else cout << "顺序栈为空！" << endl;

}

10、创建并输入栈元素

void foundStack(stack s) {

initStack(s);

int n;

cout << "请输入元素个数：";

cin >> n;

cout << "请输入元素：";

for(int i = 0; i < n; i++) {

int e;

if((s.top-s.base) == s.stackSize) {

cout << "栈满了，无法添加新元素！" << endl;

return;

}

else {

\*(s.top) = e;

s.top++;

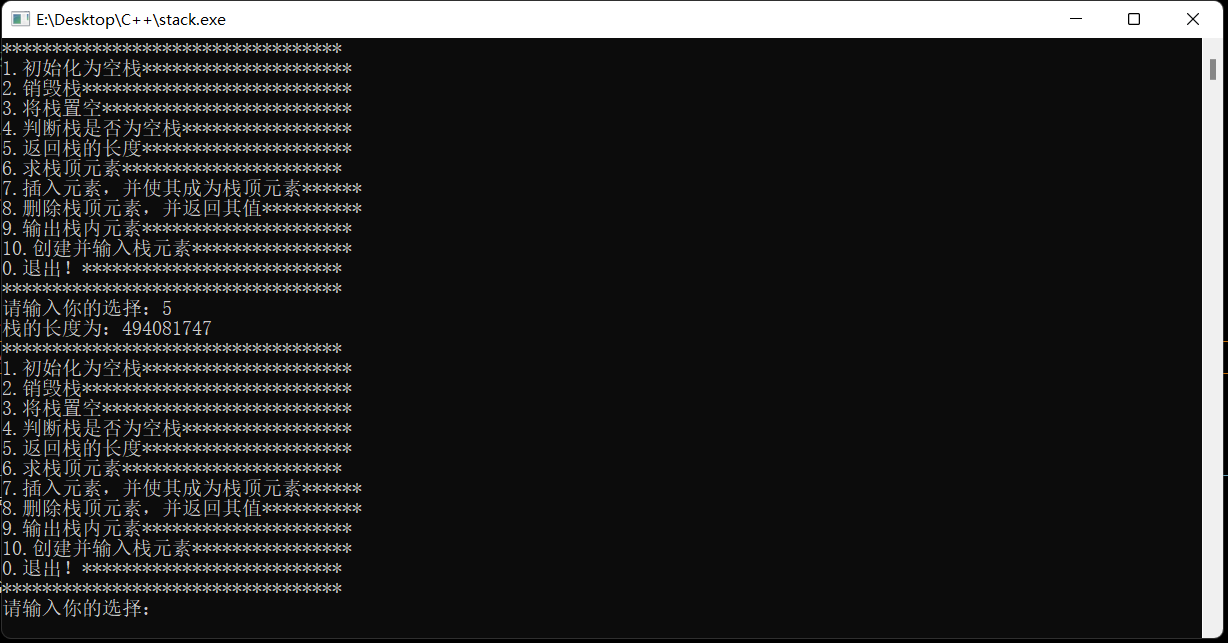
}

}

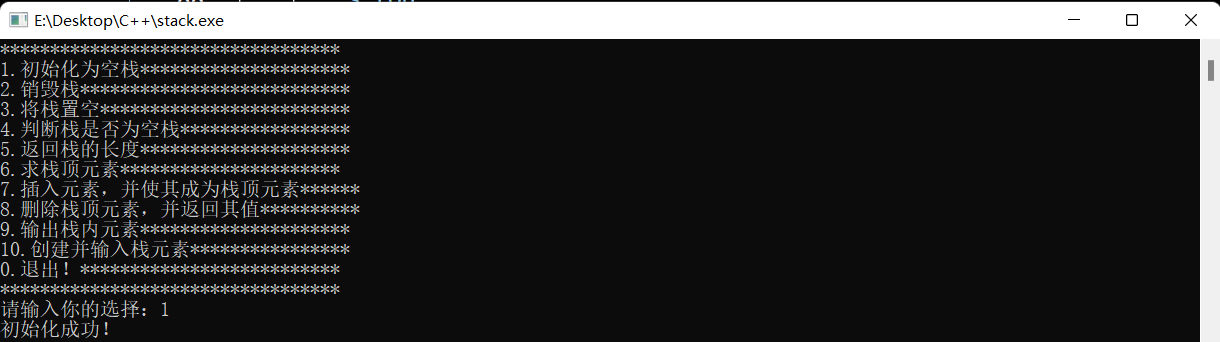
}

实验数据记录：

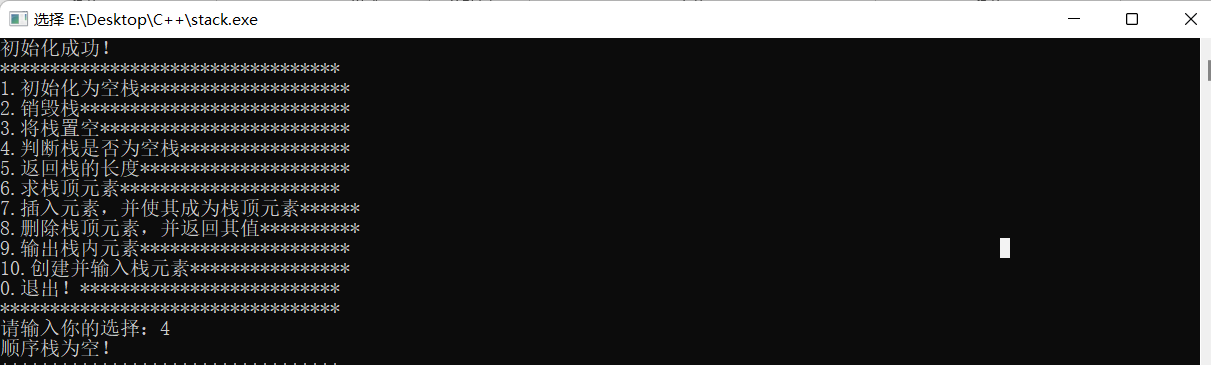
1. 没有进行初始化前进行其他操作，程序是否能控制住：



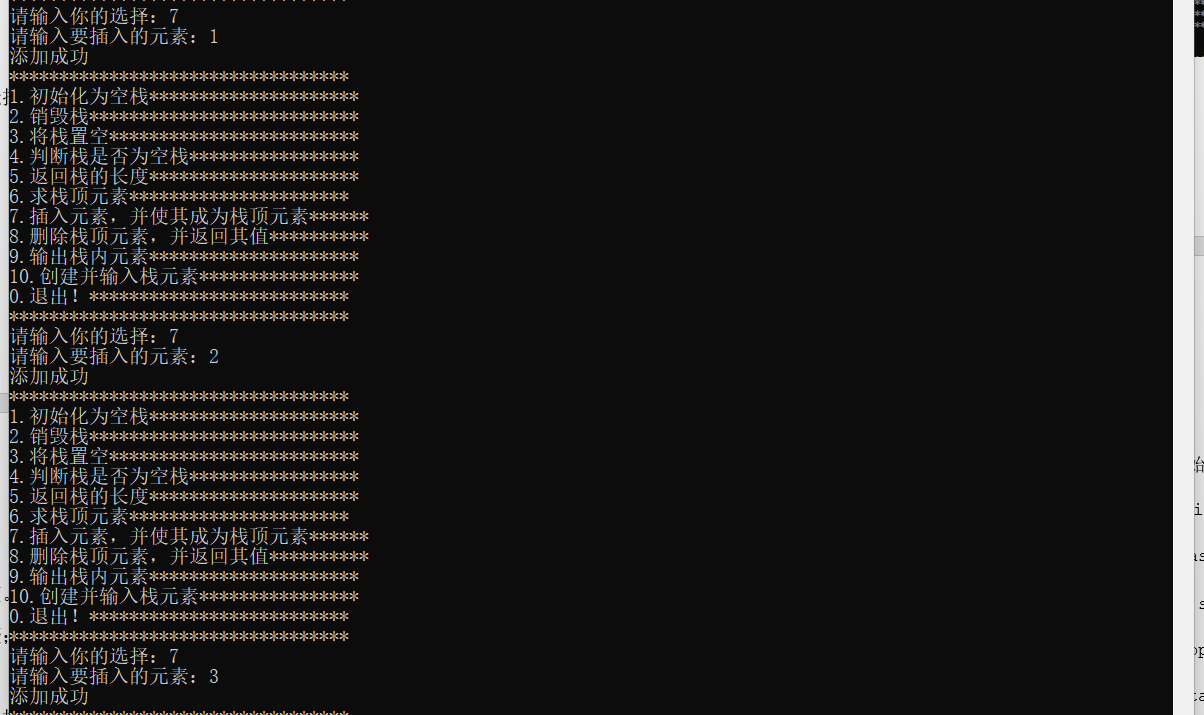
1. 初始化一个栈：



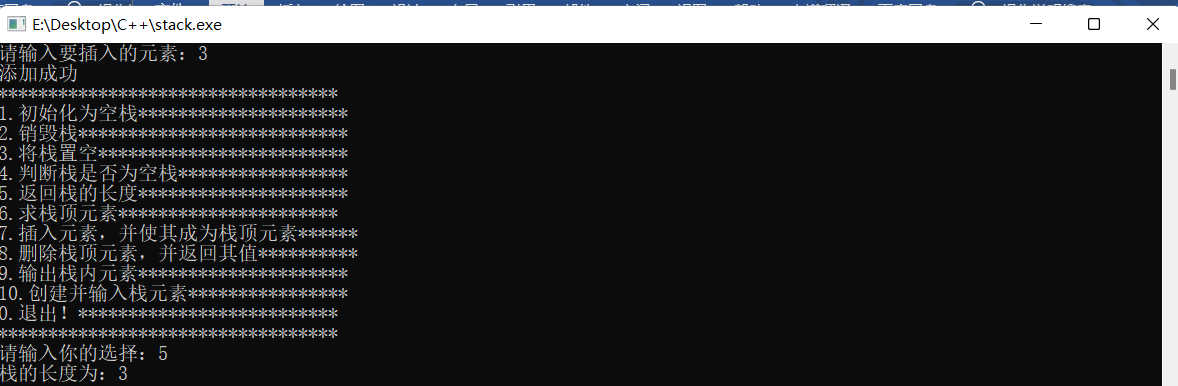
1. 判断栈是否为空：



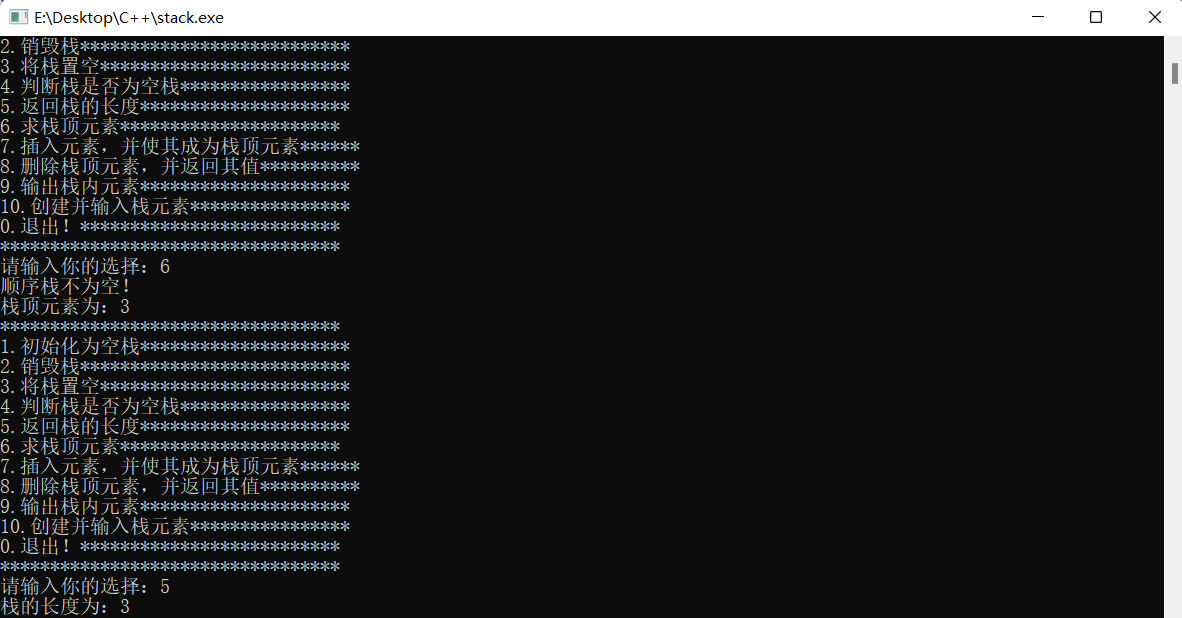
1. 3数入栈，1、2、3：



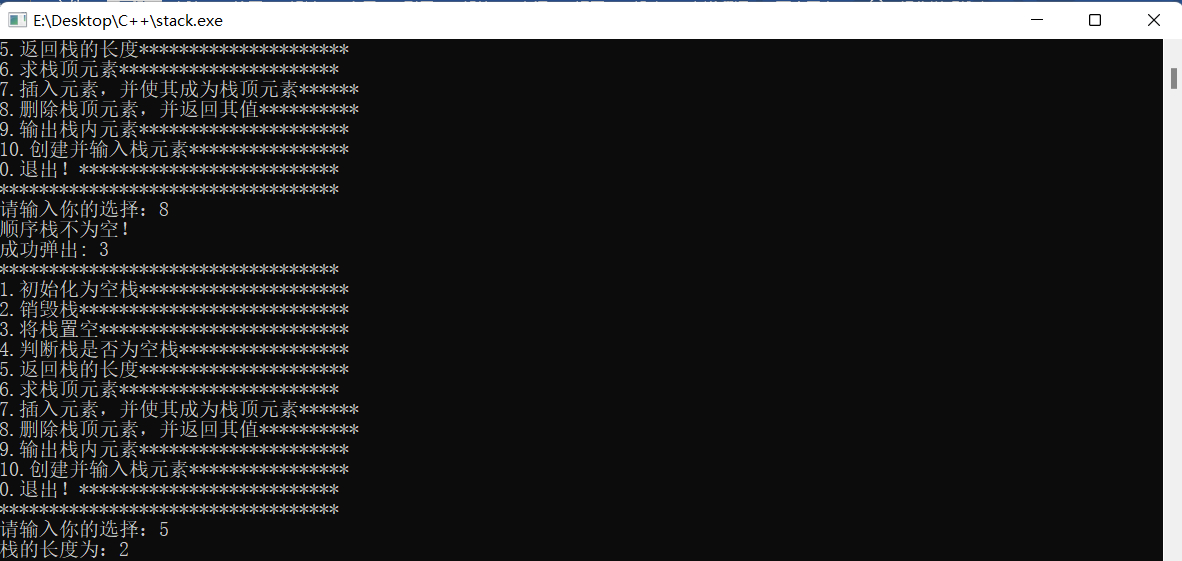
1. 输出栈长度：



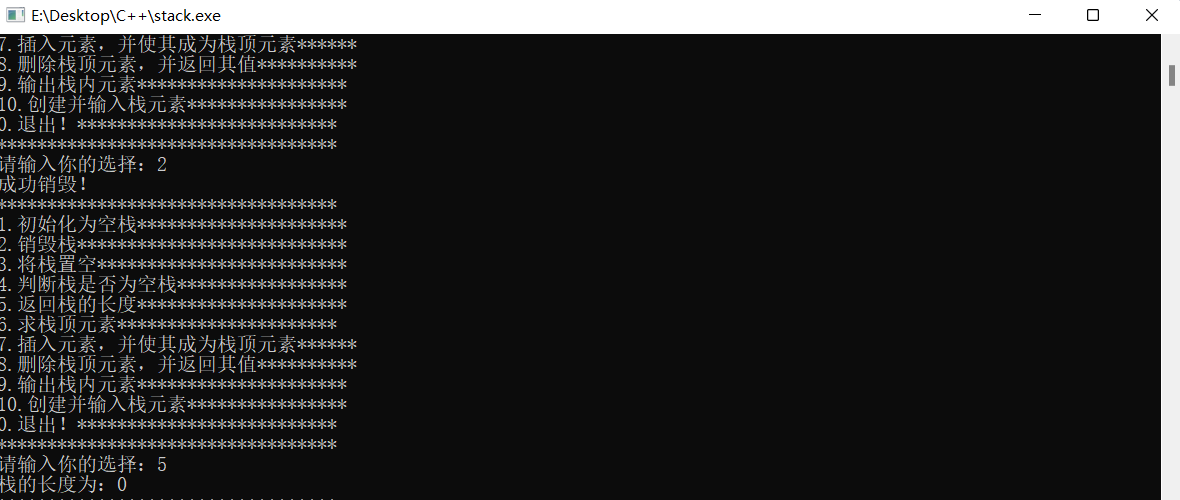
6、取栈顶元素，在判断栈是否为空，然后在判断栈的长度



7、出栈，在判断栈的长度



8、销毁栈，在做其他操作，判断程序是否能控制住：



问题讨论：

1、本次实验报告代码采用了Markdown风格。

源代码：

#include <iostream>

using namespace std;

const int MaxSize = 100;

typedef struct sqstack {

int \*top;//栈顶指针

int \*base;//栈底指针

int stackSize;//栈的最大容量

}stack;

//1、初始化为空栈

void initStack(stack &s) {

s.base = new int[MaxSize];

if(!s.base) exit(0);//内存分配失败，结束

s.top = s.base;//初始的时候栈的top与base相等

s.stackSize = MaxSize;//栈的最大容量

cout << "初始化成功！" << endl;

}

//2、销毁栈

void DestoryStack(stack &s) {

if(s.base) {

delete s.base;

s.stackSize = 0;

s.base = NULL;

s.top = NULL;

cout << "成功销毁！" << endl;

}

else cout << "栈已经被销毁，无需销毁！" << endl;

}

//3、将栈置空

void CleanStack(stack &s) {

if(s.base) {

s.top = s.base;

cout << "成功清空！" << endl;

}

else cout << "栈已经被销毁，无需清空！" << endl;

}

//4、判断栈是否为空栈

int isEmpty(stack s) {

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

cout << "顺序栈为空！" << endl;

return 0;

}

cout << "顺序栈不为空！" << endl;

return 0;

}

//5、返回栈的长度

int stackLength(stack s) {

cout << "栈的长度为：" << s.top - s.base << endl;

return s.top - s.base;

}

//6、求栈顶元素

int top(stack s) {

if(isEmpty(s)) {

cout << "栈为空，没有栈顶元素！" << endl;

return 0;

}

else {

int e;

s.top--;

e = \*(s.top);

cout << "栈顶元素为：" << e << endl;

return 0;

}

}

//7、插入元素，并使其成为栈顶元素

void push(stack &s,int e) {

if((s.top-s.base) == s.stackSize) cout << "栈满了，无法添加新元素" << endl;

else {

\*(s.top) = e;

s.top++;

cout << "添加成功" << endl;

}

}

//8、删除栈顶元素，并返回其值

void pop(stack &s) {

int e;

if(isEmpty(s)) cout << "栈为空，无法弹出" << endl;

else {

s.top--;

e = \*(s.top);

cout << "成功弹出: " << e << endl;

}

}

//9、输出栈内元素

void traverse(stack s) {

int length = stackLength(s);

if(length>0) {

cout << "顺序栈的元素为: ";

for(int i = 0; i < length; i++) cout << s.base[i] << " ";

cout << endl;

}

else cout << "顺序栈为空！" << endl;

}

//10、创建并输入栈元素

void foundStack(stack s) {

initStack(s);

int n;

cout << "请输入元素个数：";

cin >> n;

cout << "请输入元素：";

for(int i = 0; i < n; i++) {

int e;

if((s.top-s.base) == s.stackSize) {

cout << "栈满了，无法添加新元素！" << endl;

return;

}

else {

\*(s.top) = e;

s.top++;

}

}

}

void show() {

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "1.初始化为空栈\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "2.销毁栈\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "3.将栈置空\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "4.判断栈是否为空栈\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "5.返回栈的长度\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "6.求栈顶元素\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "7.插入元素，并使其成为栈顶元素\*\*\*\*\*\*" << endl;

cout << "8.删除栈顶元素，并返回其值\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "9.输出栈内元素\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "10.创建并输入栈元素\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "0.退出！\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

int main()

{

int choice;

stack s;

while(1)

{

show();

cout << "请输入你的选择：";

cin >> choice;

switch(choice)

{

case 1:

initStack(s);

break;

case 2:

DestoryStack(s);

break;

case 3:

CleanStack(s);

break;

case 4:

isEmpty(s);

break;

case 5:

stackLength(s);

break;

case 6:

top(s);

break;

case 7:

int num;

cout << "请输入要插入的元素：";

cin >> num;

push(s,num);

break;

case 8:

pop(s);

break;

case 9:

traverse(s);

break;

case 10:

foundStack(s);

break;

case 0:

exit(0);

break;

default:

cout << "输入有误，请重新输入！" << endl;

break;

}

}

}