**实验二：顺序栈实现**

**学号：B21080526 班级：B210416 姓名：单家俊**

1. **实验目的**
2. 理解队列、栈、队列的实现原理，并能用C语言实现。
3. 掌握链队、循环队列的实现方法，并能用C语言实现。
4. **实验任务**

利用顺序表或链表设计一个栈或队列，实现其初始化、加入元素、取出元素、判断非空操作。

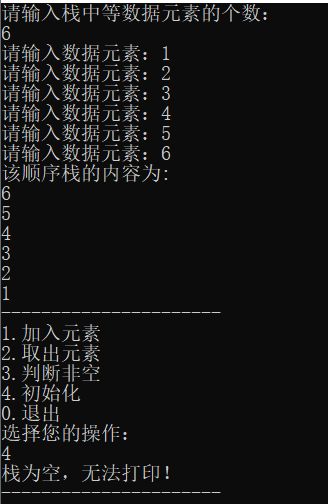
1. **实验设备**

Visual Studio

1. **实验内容（步骤）**
2. 利用C++设计一个顺序栈。
3. 编写函数分别实现顺序栈的初始化、加入元素、取出元素、判断非空操作。
4. 编写主函数引导用户输入顺序栈的元素，并选择功能。
5. **实验结果（总结、分析）**

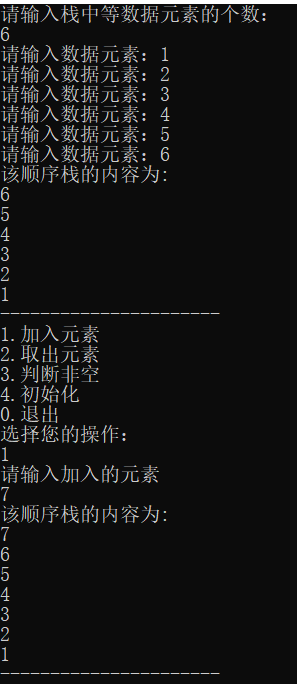
**实验结果分析：**

1. 初始化



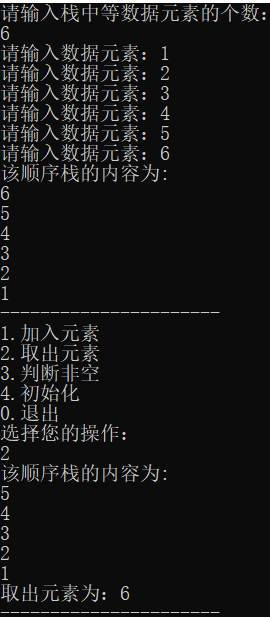
创建顺序栈之后，选择初始化操作，栈变为空，无法打印，结果正确。

1. 加入元素



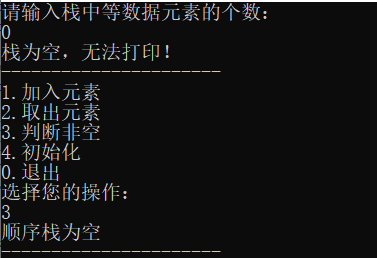
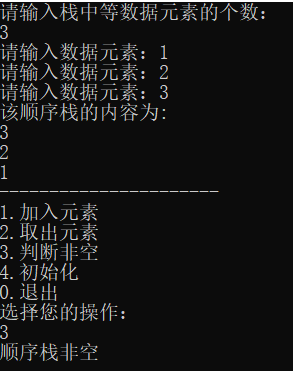
创建顺序栈之后加入元素7，结果正确。

3、取出元素



栈后进先出，所以取出元素时先将栈顶元素取出。结果正确。

4、判断非空操作



结果正确。

**过程中遇到的问题：**

1、对于C++的动态内存管理机制不太熟练。

2、栈和队列的基本概念、操作(如入栈、出栈、入队、出队)有所遗忘。

**体会：**

1. 理解概念的重要性：在学习任何数据结构之前，首先要理解其基本概念。对于栈和队列来说，理解它们的定义、操作以及应用场景非常重要。只有深入理解了这些概念，才能更好地掌握栈和队列的实现和应用。
2. 实践是检验真理的唯一标准：通过编写代码来实现栈和队列的数据结构，可以更深入地理解它们的工作原理。在实践中，我发现了解如何正确处理边界条件、避免内存泄漏等问题非常重要。
3. 多角度思考问题：在学习数据结构时，要从多个角度思考问题。例如，在实现栈和队列时，需要考虑时间复杂度、空间复杂度等因素。此外，还需要考虑如何与其他数据结构结合使用，以便解决特定的问题。
4. 持续学习和实践的重要性：数据结构是计算机科学的核心基础知识之一，因此需要不断地学习和实践。在学习过程中，我意识到只有不断练习和思考，才能真正掌握数据结构的知识和技能。
5. **附录（代码）**

//栈的顺序表示

#include<iostream>

using namespace std;

#define ERROR 0

#define OVERFLOW 0

#define OK 1

typedef int SElemType;

typedef int Status;

#define MAXSIZE 100

typedef struct

{

SElemType\* base; //栈底指针

SElemType\* top; //栈顶指针

int stacksize; //栈的容量

}SqStack;

//顺序栈的初始化

Status InitStack(SqStack &S)

{

S.base = new SElemType[MAXSIZE];

if (!S.base) return OVERFLOW;

S.top = S.base;

S.stacksize = MAXSIZE;

return OK;

}

//判断是否为空

bool StackEmpty(SqStack S)

{

if (S.top == S.base) return true;

else return false;

}

//进栈

Status Push(SqStack &S, SElemType e)

{

if (S.top - S.base == S.stacksize) // 栈满

return ERROR;

\*S.top++ = e;

return OK;

}

//出栈

Status Pop(SqStack &S, SElemType &e)

{

if (S.top == S.base) // 栈空

return ERROR;

e=\*(--S.top); //栈顶指针减1，将栈顶元素赋给e

return OK;

}

//取栈顶元素

Status GetTop(SqStack S, SElemType &e)

{

if (S.top == S.base) return ERROR; // 栈空

e = \*(S.top-1); //返回栈顶元素的值，栈顶指针不变

return OK;

}

//打印顺序栈

void PrintStack(SqStack S)

{

if (S.top == S.base)

cout << "栈为空，无法打印！" << endl;

else

{

int count;

cout << "该顺序栈的内容为:" << endl;

SElemType\* p = new SElemType;

p = S.top;

for(count=0;count<S.top-S.base;count++)

{

p = p - 1;

cout << \*p << endl;

}

}

}

int main()

{

SqStack S;

int choice;

int x,e;

InitStack(S);

cout << "请输入栈中等数据元素的个数：" << endl;

int n,elem;

cin >> n;

int i;

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

{

cout << "请输入数据元素：";

cin >> elem;

Push(S, elem);

}

PrintStack(S);

do {

cout << "----------------------" << endl;

cout << "1.加入元素" << endl;

cout << "2.取出元素" << endl;

cout << "3.判断非空" << endl;

cout << "4.初始化" << endl;

cout << "0.退出" << endl;

cout << "选择您的操作：" << endl;

cin >> choice;

switch(choice)

{

case 1:

cout << "请输入加入的元素" << endl;

cin >> x;

Push(S, x);

PrintStack(S);

continue;

case 2:

Pop(S, e);

PrintStack(S);

cout << "取出元素为：" << e << endl;

continue;

case 3:

if (StackEmpty(S) == true)

cout << "顺序栈为空" << endl;

else

cout << "顺序栈非空" << endl;

continue;

case 4:

InitStack(S);

PrintStack(S);

continue;

case 0:

break;

}

} while (choice != 0);

return 0;

}