1、采用书上第 46 页定义的栈的顺序存储表示，编程实现栈的下列基本操作。

（1）初始化顺序栈　　（2）创建顺序栈　　（3）判断栈空　　（4）输出顺序栈

（5）取栈顶元素　　（6）入栈　　（7）出栈

注：书上第 47 页函数代码未列出。

```
#include <stdio.h>
#include <stdlib.h>
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define OVERFLOW -2
#define STACK_INIT_SIZE 100          //存储空间初始分配量
#define STACKINCREMENT 10            //存储空间分配增量
typedef int Status;
typedef int SElemType;
typedef struct
{
    SElemType *base;      //在栈构造之前和销毁之后，base 的值为 NULL
    SElemType *top;       //栈顶指针
    int stacksize;        //当前已分配的存储空间，以元素为单位
}SqStack;

Status StackEmpty(SqStack &S){
    if (S.top ==S.base)
        return TRUE;
    else
        return FALSE;
}//StackEmpty

Status Stackoutput(SqStack &S){
    //若栈不空，则从栈顶到栈底依次输出数据元素，并返回 OK；否则返回 ERROR
    SElemType *p;
    if (S.top ==S.base) return ERROR;
    p = S.top;
    while(p!=S.base)
        printf("%d ",* --p);
    printf("\n");
    return OK;
}//StackTraverse

Status StackTraverse(SqStack &S){
    //若栈不空，则从栈底到栈顶依次输出数据元素，并返回 OK；否则返回 ERROR
    SElemType *p;
    if (S.top ==S.base) return ERROR;
    p = S.base;
    while(p!=S.top)
```

```
        printf("%d ",*p++);
    printf("\n");
    return OK;
}//StackTraverse

void main()
{
    int i,n,k,h,a,b;
    SqStack S;
    printf("创建一个空栈！\n");
    InitStack(S);
    printf("判断栈是否为空！\n");
    printf("StackEmpty(S)=%d\n",StackEmpty(S));
    printf("创建栈的元素个数：\n");
    scanf("%d", &n);
    printf("输入%d 个入栈元素的值：\n",n);
    for(i=0;i<n;i++)
    {
        scanf("%d", &k);
        Push(S, k);
    }
    printf("逆序输出顺序栈元素值：\n");
    Stackoutput(S);
    printf("输出顺序栈元素值：\n");
    StackTraverse(S);
    printf("输入入栈元素值：");
    scanf("%d", &h);
    Push(S, h);
    printf("输出入栈后的顺序栈元素值：\n");
    StackTraverse(S);
    Pop(S, a);
    printf("输出第 1 个出栈元素值：%d\n", a);
    Pop(S, a);
    printf("输出第 2 个出栈元素值：%d\n", a);
    printf("输出两次出栈后顺序栈元素值：");
    StackTraverse(S);
    GetTop(S, b);
    printf("输出栈顶元素值：%d\n",b);
}
```