```c
/*
堆栈：
    具有一定操作约束的线性表
    只在一端（栈顶，Top）做插入、删除
    插入数据：入栈（Push）
    删除数据：出栈（Pop）
    后入先出

栈的顺序存储实现：
    通常由一个一维数组和一个记录栈顶元素位置的变量组成
*/

#include<stdio.h>
#include<malloc.h>
#define MaxSize 100 // 堆栈元素的最大个数

typedef int ElementType;
typedef struct SNode *Stack;
struct SNode{
    ElementType Data[MaxSize]; // 存储堆栈元素
    int Top; // 记录栈顶元素下标
};

Stack CreateStack(); // 初始化堆栈
int IsFull(Stack S); // 判断堆栈是否已满
int IsEmpty(Stack S); // 判断堆栈是否为空
void Push(Stack S, ElementType item); // 入栈
ElementType Pop(Stack S); // 出栈

// 初始化堆栈
Stack CreateStack()
{
    Stack S;
    S = (Stack)malloc(sizeof(struct SNode));
    S->Top = -1;
    return S;
}

// 是否已满
int IsFull(Stack S)
{
    return (S->Top == MaxSize-1);
}

// 是否为空
int IsEmpty(Stack S)
{
    return (S->Top == -1);
}

// 入栈，栈顶指针先+1，然后元素入栈
void Push(Stack S, ElementType item)
{
    if(IsFull(S))
    {
        printf("堆栈满");
        return;
    }
    else
    {
        S->Top++; // 栈顶元素加一
        S->Data[S->Top] = item;
        return;
    }
}

// 出栈，得到栈顶元素的值，栈顶指针-1
ElementType Pop(Stack S)
{
    if(IsEmpty(S))
    {
        printf("堆栈空");
        return;
```

```
74      }
75      else
76      {
77          ElementType val = S->Data[S->Top]; // 取出最上
78          S->Top--; // 栈顶元素减一
79          return val;
80      }
81  }
82
83  void print(Stack S)
84  {
85      printf("打印当前的栈: ");
86      if(IsEmpty(S))
87          printf("栈空");
88      else
89      {
90          int i;
91          for(i=0; i<=S->Top; i++)
92              printf("%d ", S->Data[i]);
93      }
94      printf("\n");
95  }
96
97  int main()
98  {
99      Stack S;
100     S = CreateStack();
101     printf("5入栈\n");
102     Push(S, 5);
103     printf("7入栈\n");
104     Push(S, 7);
105     printf("66入栈\n");
106     Push(S, 66);
107     print(S);
108     printf("%d出栈\n", Pop(S));
109     print(S);
110     printf("%d出栈\n", Pop(S));
111     print(S);
112     return 0;
113 }
114
```