```c
/*
堆栈的链式存储：
    栈的链式存储结构实际上就是一个单链表，叫做链栈
    插入和删除操作只能在链的栈顶进行
    为了方便元素的插入和删除，栈顶指针Top指向链表的头
*/

#include<stdio.h>
#include<malloc.h>
typedef int ElementType;
typedef struct SNode *Stack;
struct SNode{
    ElementType Data;
    Stack Next;
};

Stack CreateStack(); // 初始化链栈
int IsEmpty(Stack S); // 判断链栈是否为空
void Push(Stack S, ElementType item); // 入栈
ElementType Pop(Stack S); // 出栈


// 初始化，头结点不保存任何元素
// S(top) -> node1 -> node2 -> node3 ...
Stack CreateStack()
{
    Stack S;
    S = (Stack)malloc(sizeof(struct SNode));
    S->Next = NULL; // 如果使用 S=NULL 初始化，表示头结点保存元素
    return S;
}

// 判断是否为空
int IsEmpty(Stack S)
{
    return (S->Next == NULL);
}

// 入栈
void Push(Stack S, ElementType item)
{
    Stack tmp;
    tmp = (Stack)malloc(sizeof(struct SNode));
    tmp->Data = item;
    tmp->Next = S->Next;
    S->Next = tmp;
}

// 出栈
ElementType Pop(Stack S)
{
    Stack popNode;
    ElementType popVal;
    if(IsEmpty(S))
    {
        printf("堆栈空");
        return;
    }
    else
    {
        popNode = S->Next; // 出栈第一个元素在栈顶元素后面
        S->Next = popNode->Next; // 把第一个元素从链栈删除
        popVal = popNode->Data; // 取出被删除结点的值
        free(popNode); // 释放空间
        return popVal;
    }
}

void print(Stack S)
{
    printf("打印当前的栈（从栈顶到栈底依次是）：");
    if(IsEmpty(S))
        printf("栈空");
```

```
    else
    {
        Stack p = S->Next;
        while(p)
        {
            int tmp = p->Data;
            printf("%d ", tmp);
            p = p->Next;
        }
    }
    printf("\n");
}

int main()
{
    Stack S;
    S = CreateStack();
    printf("5入栈\n");
    Push(S, 5);
    print(S);
    printf("7入栈\n");
    Push(S, 7);
    print(S);
    printf("66入栈\n");
    Push(S, 66);
    print(S);
    printf("%d出栈\n", Pop(S));
    print(S);
    printf("%d出栈\n", Pop(S));
    print(S);
    return 0;
}

```