

```

1  /*
2  堆栈的链式存储:
3      栈的链式存储结构实际上就是一个单链表, 叫做链栈
4      插入和删除操作只能在链的栈顶进行
5      为了方便元素的插入和删除, 栈顶指针Top指向链表的头
6  */
7
8  #include<stdio.h>
9  #include<malloc.h>
10 typedef int ElementType;
11 typedef struct SNode *Stack;
12 struct SNode{
13     ElementType Data;
14     Stack Next;
15 };
16
17 Stack CreateStack(); // 初始化链栈
18 int IsEmpty(Stack S); // 判断链栈是否为空
19 void Push(Stack S, ElementType item); // 入栈
20 ElementType Pop(Stack S); // 出栈
21
22
23 // 初始化, 头结点不保存任何元素
24 // S(top) -> node1 -> node2 -> node3 ...
25 Stack CreateStack()
26 {
27     Stack S;
28     S = (Stack)malloc(sizeof(struct SNode));
29     S->Next = NULL; // 如果使用 S=NULL 初始化, 表示头结点保存元素
30     return S;
31 }
32
33 // 判断是否为空
34 int IsEmpty(Stack S)
35 {
36     return (S->Next == NULL);
37 }
38
39 // 入栈
40 void Push(Stack S, ElementType item)
41 {
42     Stack tmp;
43     tmp = (Stack)malloc(sizeof(struct SNode));
44     tmp->Data = item;
45     tmp->Next = S->Next;
46     S->Next = tmp;
47 }
48
49 // 出栈
50 ElementType Pop(Stack S)
51 {
52     Stack popNode;
53     ElementType popVal;
54     if(IsEmpty(S))
55     {
56         printf("堆栈空");
57         return;
58     }
59     else
60     {
61         popNode = S->Next; // 出栈第一个元素在栈顶元素后面
62         S->Next = popNode->Next; // 把第一个元素从链栈删除
63         popVal = popNode->Data; // 取出被删除结点的值
64         free(popNode); // 释放空间
65         return popVal;
66     }
67 }
68
69 void print(Stack S)
70 {
71     printf("打印当前的栈 (从栈顶到栈底依次是): ");
72     if(IsEmpty(S))
73         printf("栈空");

```

```
74     else
75     {
76         Stack p = S->Next;
77         while(p)
78         {
79             int tmp = p->Data;
80             printf("%d ", tmp);
81             p = p->Next;
82         }
83     }
84     printf("\n");
85 }
86
87 int main()
88 {
89     Stack S;
90     S = CreateStack();
91     printf("5入栈\n");
92     Push(S, 5);
93     print(S);
94     printf("7入栈\n");
95     Push(S, 7);
96     print(S);
97     printf("66入栈\n");
98     Push(S, 66);
99     print(S);
100    printf("%d出栈\n", Pop(S));
101    print(S);
102    printf("%d出栈\n", Pop(S));
103    print(S);
104    return 0;
105 }
106
```