

```

1  /* LinkStack.c */
2  #include <string.h>
3  #include <malloc.h>
4  #include <stdlib.h>
5  #include <assert.h>
6  #include "LinkStack.h"
7
8  //链栈的初始化
9  void StackNew(STACK *s, int keySize, StackFree *freeFn)
10 {
11     assert(keySize > 0);
12     s->head.next = NULL;
13     s->keySize = keySize;
14     s->size = 0;
15     s->freeFn = freeFn;
16 }
17
18 //栈判空
19 int StackEmpty(STACK *s)
20 {
21     return (0 == s->size);
22 }
23
24 //栈中节点数量
25 int StackSize(STACK *s)
26 {
27     return s->size;
28 }
29
30 //链栈的销毁
31 void StackDispose(STACK *s)
32 {
33     STACKNODE *cur = s->head.next, *post;
34     for (; NULL != cur; cur = post)
35     {
36         post = cur->next;
37         if (NULL != s->freeFn)
38         {
39             s->freeFn(cur->key);
40         }
41         free(cur);
42     }
43     s->head.next = NULL;
44     s->size = 0;
45 }
46
47 //入栈
48 int StackPush(STACK *s, const void *e)
49 {
50     STACKNODE *newNode = malloc(sizeof(STACKNODE) + s->keySize);
51     if(NULL == newNode)
52     {
53         return -1;
54     }
55     newNode->next = s->head.next;
56     s->head.next = newNode;
57     memcpy(newNode->key, e, s->keySize);
58     s->size ++;
59     return 0;
60 }
61
62 //出栈
63 int StackPop(STACK *s, void *e)
64 {
65     if (StackEmpty(s))
66     {
67         return -1;
68     }
69     STACKNODE *node = s->head.next;
70     if (NULL != e)
71     {
72         memcpy(e, node->key, s->keySize);
73     }

```

```
74     s->head.next = node->next;
75     free(node);
76     s->size --;
77     return 0;
78 }
79
80 //获取栈顶元素
81 int StackTop(STACK *s, void *e)
82 {
83     if (StackEmpty(s) && (NULL != e))
84     {
85         return -1;
86     }
87     STACKNODE *node = s->head.next;
88     memcpy(e, node->key, s->keySize);
89     return 0;
90 }
```