```c
/* LinkStack.c */
#include <string.h>
#include <malloc.h>
#include <stdlib.h>
#include <assert.h>
#include "LinkStack.h"

//链栈的初始化
void StackNew(STACK *s, int keySize, StackFree *freeFn)
{
    assert(keySize > 0);
    s->head.next = NULL;
    s->keySize = keySize;
    s->size = 0;
    s->freeFn = freeFn;
}

//栈判空
int StackEmpty(STACK *s)
{
    return (0 == s->size);
}

//栈中节点数量
int StackSize(STACK *s)
{
    return s->size;
}

//链栈的销毁
void StackDispose(STACK *s)
{
    STACKNODE *cur = s->head.next, *post;
    for (; NULL != cur; cur = post)
    {
        post = cur->next;
        if (NULL != s->freeFn)
        {
            s->freeFn(cur->key);
        }
        free(cur);
    }
    s->head.next = NULL;
    s->size = 0;
}

//入栈
int StackPush(STACK *s, const void *e)
{
    STACKNODE *newNode = malloc(sizeof(STACKNODE) + s->keySize);
    if(NULL == newNode)
    {
        return -1;
    }
    newNode->next = s->head.next;
    s->head.next = newNode;
    memcpy(newNode->key, e, s->keySize);
    s->size ++;
    return 0;
}

//出栈
int StackPop(STACK *s, void *e)
{
    if (StackEmpty(s))
    {
        return -1;
    }
    STACKNODE *node = s->head.next;
    if (NULL != e)
    {
        memcpy(e, node->key, s->keySize);
    }
```

```
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      }
```