# 线性表 第 22 题

## 问题描述:

将节点插入有序动态单链表，使其仍然有序。

## 1：函数结构设计

| | |
|---|---|
| 函数名 | Insert |
| 函数正常输入 | LinkList 型的单链表头指针，<br>ElemType 型(宏定义为 int 型)的要插入的节点数据 |
| 函数正常输出 | 插入成功返回 TRUE=1，插入失败返回 FALSE=-1 |

## 2：测试样例设计

| | 输入 | 预测结果 |
|---|---|---|
| 一般正常情况 | 单链表头指针，要插入的节点数据 | 正常插入，返回 TRUE=1 |
| 异常情况 | 输入的链表头指针为空 | 返回 FALSE=-1 |

## 3：伪代码描述

| |
|---|
| 如果输入的链表为空，返回 FALSE=-1 |
| 指针 prev 指向头指针 lst，lst 指向 prev 的下一个节点，即 lst = prev->next |
| prev 和 lst 向后移动，循环查找，直到参数 dat 小于 lst->dat，或者到达链表结尾(lst 为空) |
| 产生新的节点，并插入到 prev 和 lst 节点之间 |
| 返回 TRUE=1 |

## 4：程序描述

```
// 线性表 22 题

#include "stdafx.h"
#include <assert.h>
#include <iostream>
using namespace std;

#define TRUE              1
#define FALSE            -1

typedef int ElemType;

typedef struct LinkList
{
	ElemType dat;
	LinkList *next;
} LinkList;

// 创建空链表
LinkList *List_InitEmpty(void)
{
	LinkList *head = new LinkList;
	assert(head != NULL);
	head->next = NULL;
	return head;
}

// 尾插法创建链表
LinkList *List_InitRear(ElemType dat[], int n)
{
	int i;
	if(dat == NULL) {
		return List_InitEmpty();
	}
	LinkList *head = new LinkList, *New, *Last;
	assert(head != NULL);
	Last = head;
	for(i = 0; i < n; ++i) {
		Last->next = New = new LinkList;
		assert(New != NULL);
		New->dat = dat[i];
```

```
            Last = New;
        }
        Last->next = NULL;
        return head;
}

// 递增有序单链表 插入节点
int Insert(LinkList *lst, ElemType dat)
{
        if(lst == NULL) {
            return FALSE;
        }
        LinkList *prev = lst;
        lst = lst->next;
        while(lst != NULL) { // 如果找到最后一个节点仍没有比 dat 大的，则插入最后一个节点
prev 之后
            if(dat < lst->dat) {
                break;                  // 找到比参数 dat 值大的节点，跳出循环
            }
            prev = lst;
            lst = lst->next;
        }
        // 此时将新的节点插入到 prev 和 lst 之间即可
        LinkList *New = new LinkList;
        New->dat = dat;
        New->next = lst;
        prev->next = New;
        return TRUE;
}

// 输出所有节点
void PrintAll(LinkList *lst)
{
        if(lst == NULL) {
            return;
        }
        lst = lst->next;
        while(lst != NULL) {
            cout << lst->dat << " ";
            lst = lst->next;
        }
        cout << endl;
}
```

```
int main(void)
{
	int num[] = {1,3,5,6,7,8,9,15,20,21};
	LinkList *L = List_InitRear(num, 10);	// 用数组 num 初始化一个链表
	PrintAll(L);					// 打印链表所有元素
	// 插入一些元素，并输出返回值
	cout << Insert(NULL, 1) << endl;	// 异常情况测试
	cout << Insert(L, 0) << endl;		// 正常情况测试
	cout << Insert(L, 2) << endl;
	cout << Insert(L, 5) << endl;
	cout << Insert(L, 13) << endl;
	cout << Insert(L, 21) << endl;
	cout << Insert(L, 25) << endl;
	PrintAll(L);					// 打印链表所有元素
	system("pause");
	return 0;
}
```

## 5：结果展示

```
1 3 5 6 7 8 9 15 20 21	// 最初初始化的链表元素
-1			// 出错返回 FALSE=-1
1			// 执行正常返回 TRUE=1
1
1
1
1
1
0 1 2 3 5 5 6 7 8 9 13 15 20 21 21 25	// 插入元素 0,2,5,13,21,25 后的链表仍然为有序链表
```