

# 天津大学

## 数据结构上机实验报告

题目:实现 Locate 函数

学生姓名	<u>王雨朦</u>
学生学号	<u>2016229082</u>
学院名称	<u>国际工程师学院</u>
专业	<u>计算机</u>
时间	<u>2016/9/23</u>

# 目 录

第一章 需求分析 .....	1
1.1 原题表述 .....	1
1.2 解决方案 .....	1
第二章 概要设计 .....	2
2.1 抽象数据类型 .....	2
第三章 详细设计 .....	3
3.1 程序代码 .....	3
第四章 调试分析 .....	5
4.1 调试过程 .....	5
第五章 测试结果 .....	6
5.1 测试过程 .....	6

# 第一章 需求分析

## 1.1 原题表述

设有一个带表头结点的双向链表 L，每个结点有 4 个数据成员：指向前驱结点的指针 prior、指向后继结点的指针 next、存放数据的成员 data 和访问频度 freq。所有结点的 freq 初始时都为 0。每当在链表上进行一次 Locate (L, x)操作时，令元素值为 x 的结点的访问频度 freq 加 1，并将该结点前移，链接到与它的访问频度相等的结点后面（如果该结点没有找到与它访问频度相等的结点，链接到表头后面结点），使得链表中所有结点，保持按访问频度递减的顺序排列，以使频繁访问的结点总是靠近表头。

## 1.2 解决方案

- 1) 建立一个结点数为 n 的带表头的双向链表，初始化每个结点的参数。初始化时给头结点的频度赋一个较大的值。
- 2) 遍历每个结点，对于每个结点输出结点序号和频度。
- 3) 对于输入的每个 Locate 的值，找到该结点，频度值+1，头结点频度+1。
- 4) 从该结点开始，往前遍历，如果遍历到的结点频度值小于该结点的频度，则继续往前；否则将该结点插入到遍历的结点的后面。
- 5) 结束 locate 后，再次遍历每个结点，对于每个结点输出结点序号和频度。

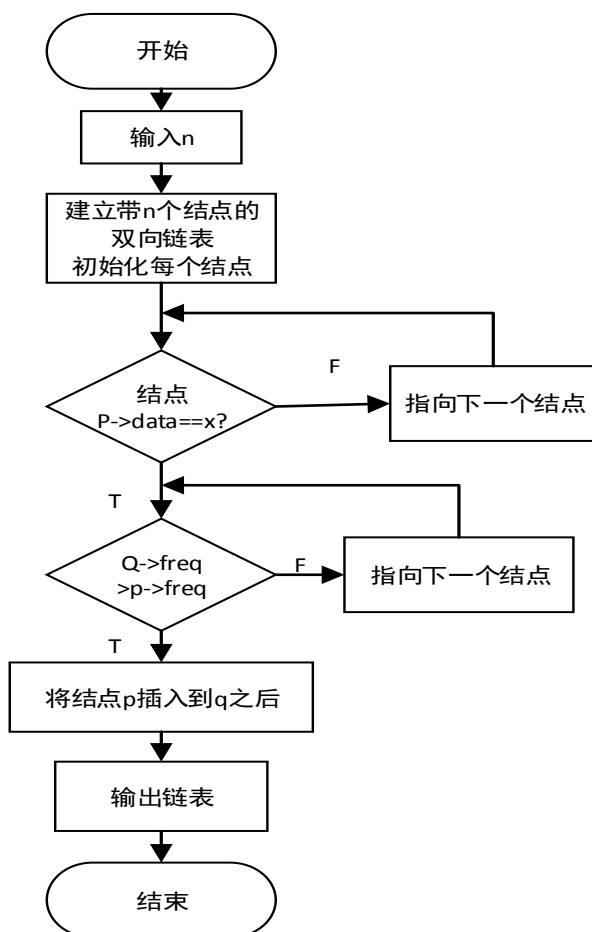
## 第二章 概要设计

### 2.1 抽象数据类型

带头结点的双向链表:

```
typedef struct node{
    struct node *prior;
    int data;
    int freq;
    struct node *next;
}node,*dulinklist;
dulinklist ha;
void initl(int n)//初始化链表
void display(dulinklist &ha)//输出链表
dulinklist findnode(dulinklist &ha,int x)//找到值为 x 的结点
void locate(dulinklist &ha,int x)//locate 函数
```

### 2.2 算法



## 第三章 详细设计

### 3.1 程序代码

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
typedef struct node{
    struct node *prior;
    int data;
    int freq;
    struct node *next;
}node,*dulinklist;

dulinklist ha;

void initl(int n){
    ha=(dulinklist)malloc(sizeof(node));
    ha->freq=100; ha->data=-1;
    node *p,*q=ha;
    for(int i=1;i<=n;i++){
        p=(node *)malloc(sizeof(node));
        p->prior=q; p->data=i; p->freq=0;
        q->next=p;
        q=p;
    }
    q->next=NULL;
}

void display(dulinklist &ha){
    printf("The link list data are:\n");
    node *p=ha;
    while(p->next !=NULL){
        p=p->next;
        printf("The node is %d, its frequency is %d.\n",p->data,p->freq);
    }
}

dulinklist findnode(dulinklist &ha,int x){
    node *p=ha;
    while(p!= NULL && p->data!=x)
        p=p->next;
    return p;
}
```

```

}

void locate(dulinklist &ha, int x) {
    node *p= findnode(ha, x);
    //printf("aaa  %d  aaa", findnode(ha, x)->data);
    p->freq++; ha->freq++;
    node *q= p->prior;
    //find the freq equal with p
    while(q->freq < p->freq)
        q=q->prior;
    //change position
    if(p->next !=NULL) { //not the last node
        p->prior->next = p->next;
        p->next->prior = p->prior;
    }
    else{
        p->prior->next=NULL;
    }
    p->next=q->next;
    q->next->prior= p;
    q->next= p;
    p->prior=q;
}

int main() {
    int n, x;
    printf("Please input the link list length:\n");
    scanf("%d", &n);
    initl(n);
    display(ha);
    printf("Let's start to test Locate function <-1 means stopping input\n");
    number>:\n");

    do{
        printf("Please input number: ");
        scanf("%d", &x);
        locate(ha, x);
    }
    while(x!=-1);
    printf("After test\n");
    display(ha);
}
}

```

## 第四章 调试分析

### 4.1 调试过程

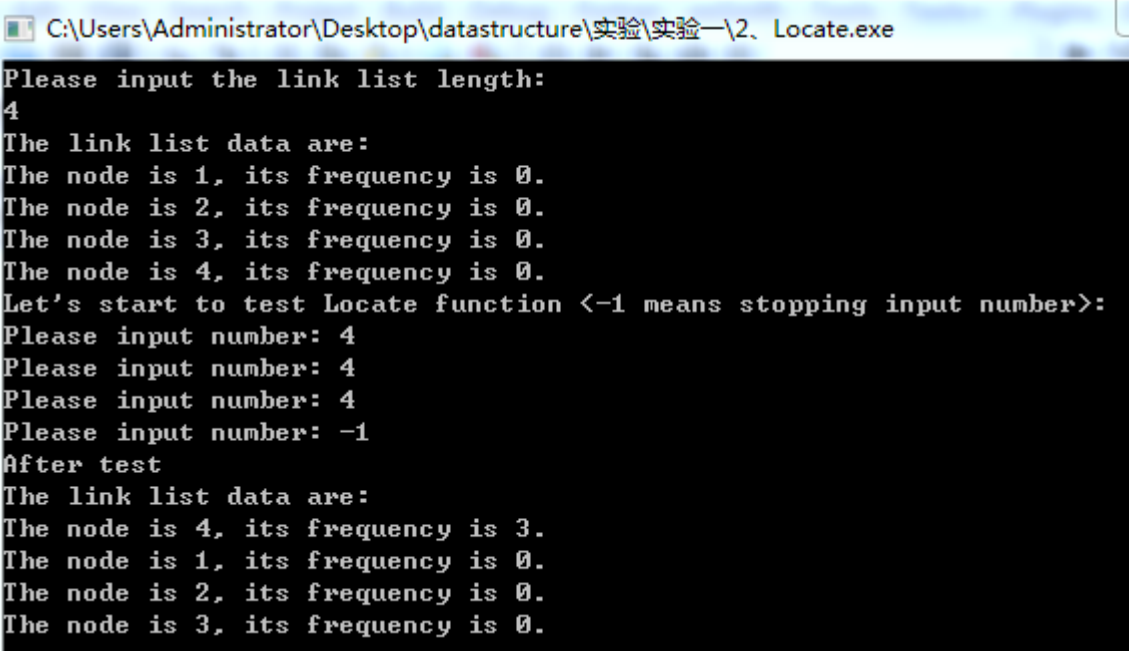
<b>Bug 名称</b>	边界特殊处理
<b>Bug 描述</b>	当 locate 最后一个结点值时，程序无限循环
<b>Bug 原因</b>	Locate 时最后一个结点的 next 为 NULL，此时 p->next->prior 这句会没有意义
<b>Bug 解决方案</b>	单独处理尾结点
<b>Bug 总结</b>	考虑不周

<b>Bug 名称</b>	数据声明错误
<b>Bug 描述</b>	头结点报错
<b>Bug 原因</b>	头结点声明出错
<b>Bug 解决方案</b>	将头结点的声明由 node 改为 linklist
<b>Bug 总结</b>	链表的应用不熟练

## 第五章 测试结果

### 5.1 测试过程

测试编号	1
测试对象	尾结点
测试输入参数	N=4,locate:4 重复三次
测试步骤	改变尾结点的频度值，观察输出
测试预期结果	4 的频度为 3，且排在队首
测试输出结果	4 的频度为 3，且排在队首
测试分析	程序正确



```

C:\Users\Administrator\Desktop\datastructure\实验\实验一\2、Locate.exe
Please input the link list length:
4
The link list data are:
The node is 1, its frequency is 0.
The node is 2, its frequency is 0.
The node is 3, its frequency is 0.
The node is 4, its frequency is 0.
Let's start to test Locate function <-1 means stopping input number>:
Please input number: 4
Please input number: 4
Please input number: 4
Please input number: -1
After test
The link list data are:
The node is 4, its frequency is 3.
The node is 1, its frequency is 0.
The node is 2, its frequency is 0.
The node is 3, its frequency is 0.
  
```



测试编号	2
测试对象	第一个结点
测试输入参数	N=4,locate:1 重复五次
测试步骤	改变第一个结点的频度值，观察输出
测试预期结果	1 的频度为 5，且排在队首
测试输出结果	1 的频度为 5，且排在队首
测试分析	程序正确

```

C:\Users\Administrator\Desktop\datastructure\实验\实验一\2、Locate.exe

Please input the link list length:
4
The link list data are:
The node is 1, its frequency is 0.
The node is 2, its frequency is 0.
The node is 3, its frequency is 0.
The node is 4, its frequency is 0.
Let's start to test Locate function <-1 means stopping input number>:
Please input number: 1
Please input number: 1
Please input number: 1
Please input number: 1
Please input number: 1
Please input number: -1
After test
The link list data are:
The node is 1, its frequency is 5.
The node is 2, its frequency is 0.
The node is 3, its frequency is 0.
The node is 4, its frequency is 0.
    
```