**实 验 报 告**

|  |  |  |
| --- | --- | --- |
| **课程名称**：数据结构 | **班级**：软工21101 | **实验成绩**： |
| **实验名称**：单链表的实现 | **学号**：2105006207 | **批阅教师签字：** |
| **实验编号**：实验三 | **姓名**：方福涛 | **实验日期：**2022 年 9 月15日 |
| **指导教师**：董傲霜 | **组号**： | **实验时间**： 时 分－ 时 分 |

**一、实验设计思想**

1. 说明单链表存储结构结构的设计思想；

定义单链表结构体

初始化单链表操作

头插法创建单链表

尾插法创建单链表

计算单链表的长度

获取单链表中元素

新元素插入单链表

删除链表中某元素

单链表的整表删除

打印出整个单链表

1. 说明单链表中头结点设置的意义。

在开始结点之前的结点（可有可无），其值域不包含任何信息，也可以包含链表长度等信息。

**二、程序说明**

1. 给出单链表结构体定义及其属性含义的说明；

typedef struct Node //定义一个结构体

{

elemtype data; //数据域

struct Node \*next; //指针域

}Node;

typedef struct Node \*LinkList;

1. 给出你设计的函数的结构说明（函数，参数，执行的结果）；

int InitList(LinkList \*L) //带有头节点的单链表的初始化

void CreateListHead(LinkList \*L) //头插法创建一个单链表，n为要插入的元素个数

void CreateListTail(LinkList \*L) //尾插法创建一个单链表，n为要插入的元素个数

int LengthList(LinkList \*L) //计算链表长度

int GetElem(LinkList L, int i, elemtype \*e) //用e返回L中第i个元素值

int InsertList(LinkList \*L, int i, elemtype e) //在L中第i个位置插入元素e

int DeleteList(LinkList \*L, int i, elemtype \*e)//删除L中第i个元素，并用e返回其值

int ClearList(LinkList \*L) //单链表的整表删除

void ShowList(LinkList \*L) //打印整个链表

1. 说明核心操作插入、删除的实现步骤（处理步骤，可用文字也可画流程图）。

插入思想：

1.声明一指针p指向链表头节点，初始化j从1开始

2.当 j < i 时，就遍历链表，让p的指针向后移动，不断指向下一节点，j 累加1

3．若到链表末尾p为空，则说明第i个节点不存在

4.否则查找成功，在系统中生成一个空节点s

5.将数据元素e赋值给s->data

6.单链表的插入标准语句s->next=p->next;p->next = s

7.返回0

删除思想：

1.声明一指针p指向链表头节点，初始化j从1开始

2.当 j < 1 时，就遍历链表，让p的指针向后移动，不断指向下一个节点，j 累加1

3.若到链表末尾p为空，则说明第i个节点不存在

4.否则查找成功，将要删除的节点p->next赋值给q

5.单链表的删除标准语句p->next = q->next;

6.将q节点中的数据赋值给e，作为返回

7.释放q节点

8.返回0

**三、实验环境**

Windows10、DEV-C++

**四、实验过程分析**

实验调试过程中的问题及改正的事件举例说明（只要调程序就不可能没有错误，现象、原因、修正方法，可以截图但不能过多）。

1、 Node head;

head->next=NULL;这个没有分配地址，直接使用 head->next，会导致程序崩溃。同样的问题，在创建删除的函数是又同样遇到,例如while(p!=NULL&&p->id!=id)，若是先判断p->id!=id,则会导致程序崩溃。为什么？其原因是当p指向NULL时，p->id!=id会导致程序崩溃。解决方案：先判断是否为空，再判断p->id!=id。

2、free()仅仅是将指针所指地址回收，实质是指针所指的地址还在，只是其中内容不再保护，地址中的值会出现各种内容，为了规范和避免引起不必要的麻烦，则需要将指针置空。

3、二级指针,函数参数中使用二级指针，如 void deleteEmun(Node \* head,int id)，一般的如果在自定义的函数中没有使用malloc()函数，即形参指针指向malloc所定义的节点，而不是指向实参所传的地址，这时候就需要使用二级指针。当然，如果是返回值是指针的函数，则不需要二级指针，返回值返回函数内定义好的指针即可。

**五、实验结果总结**

1、单链表中查找、插入和删除操作的性能分析。

查找：

我们要查找的元素 e 所在位置有关系，如果单链表中第一个元素，或者说首元结点就是 e ，那么仅执行一次即可。如果不是，则顺着指针链，依次向后查找。因线性链表只能顺序存取，即在查找时要从头指针找起，查找的时间福再度为 O(n)。

插入和删除：

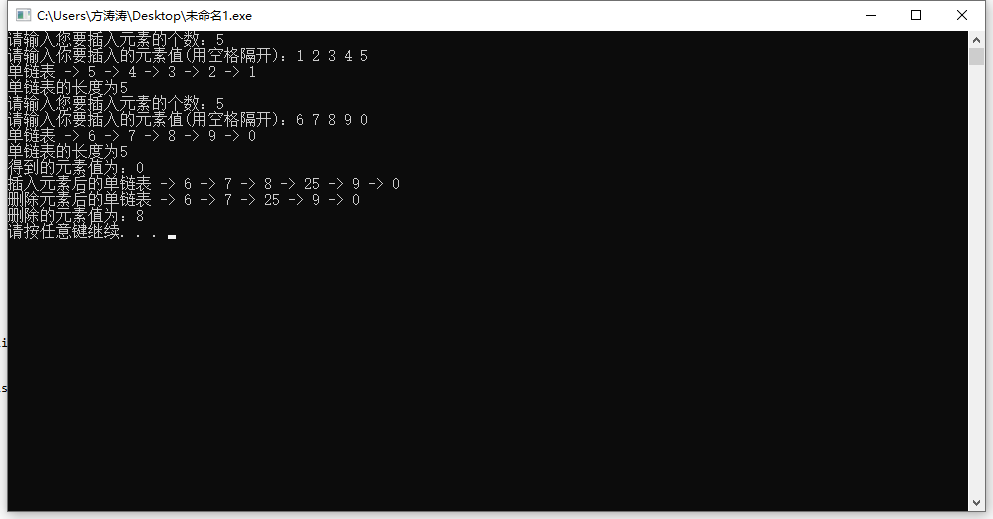
因线性链表在插入或删除时，不需要移动元素，只要修改指针，一般情况下时间复杂度为O(1)。

但是，如果要在单链表中进行前插或删除操作，由于要从头查找前驱结点，所耗时间复杂度为O(n)。

进行插入和删除的操作是常数即便，但是寻找前驱才要O(n)。

2、给出你的测试方法和测试截图（不能超过三个截图）；

分别插入元素1 2 3 4 5跟元素 6 7 8 9 0，分别进行插入删除。



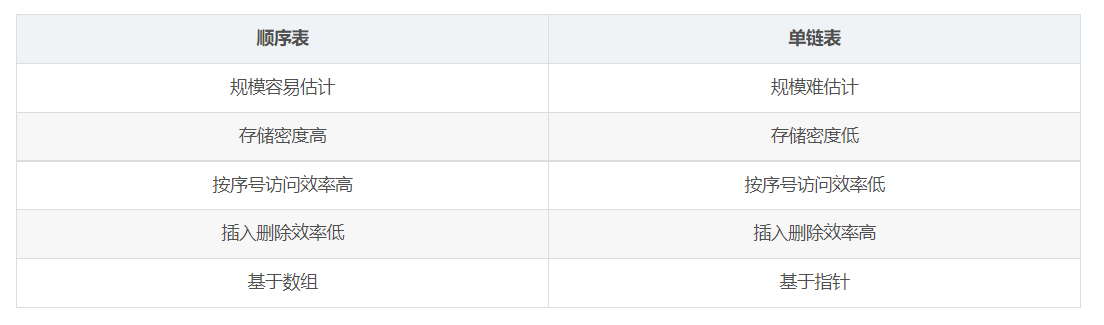
**六、附录**

1. 意见和建议（没有可不写）。

2. 思考题：

回答以下问题：

线性表的单链表对比顺序表实现方式具有什么优点、缺点？



**七、打分表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 项目 | 考核点 | 分数 | **得分** | 备注 |
| 程序 | 逻辑是否正确  程序可读性  创新点  是否符合OOD设计原则 | 50 | **0** |  |
| 报告完整性 | 实验过程阐述是否完整  测试数据设计是否合理  运行结果是否正确 | 40 | **0** |  |
| 调试问题及解决方法 | 是否对调试过程问题进行阐述 | 5 | **0** |  |
| 思考题目 | 回答是否正确 | 5 | **0** |  |
| 合计 |  | 100 | **0** |  |