



# 数据结构课程设计文档

## 题目：两个有序链表的交集

姓名： 赵卓冰

学号： 2252750

专业： 软件工程

年级： 2023 级

指导教师： 张颖

2024 年 10 月 9 日

## 目录

### 运行环境与开发工具

#### 项目要求

##### 1 功能要求

#### 项目简介

#### UI界面展示

#### 代码架构

##### 1 链表节点类 `Node`

##### 2 链表类 `List`

#### 功能模块介绍

##### 1 链表输入

##### 2 交集计算

##### 3 打印输出

#### 用户交互

#### 性能考虑

#### 错误处理和输入验证

#### 未来改进

#### 心得体会

## 1. 运行环境与开发工具

---

本项目支持在以下开发环境和编译运行环境中运行：

- **Windows 操作系统：**

- 版本：Windows 10 x64
- IDE：Visual Studio 2022 (Debug 模式)
- 编译器：MSVC 14.39.33519

- **Linux 操作系统：**

- 版本：Ubuntu 20.04.6 LTS
- IDE：VS Code
- 编译器：gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04.2)

## 2. 项目要求

---

该项目的目标是实现两个非降序链表序列的交集计算。输入为两个由若干个正整数构成的非降序链表，用 `-1` 表示序列的结尾（`-1` 不属于链表的一部分）。程序需要构造出这两个链表的交集序列，并输出结果。如果交集链表为空，则输出 `NULL`。

## 2.1. 功能要求

1. **输入说明**：输入分两行，每行包含一个非降序序列，序列以 `-1` 作为结束标志，数字间用空格分隔。
2. **输出说明**：在一行中输出两个序列的交集，数字之间用空格分开，结尾不能有多余空格；若交集为空，则输出 `NULL`。

## 3. 项目简介

本项目的目的是通过链表操作实现两个非降序链表的交集计算。用户可以通过输入两个有序链表序列，程序将自动构造交集链表并输出结果。整个过程采用链表数据结构，能够灵活处理动态数据。

项目主要功能包括：

- 使用链表数据结构存储两个有序序列。
- 计算两个有序链表的交集，并输出结果。
- 处理交集中重复元素的情况，确保输出唯一的交集序列。
- 对输入进行验证和边界情况处理（例如，空链表或无交集的情况）。

## 4. UI界面展示

五个测试用例结果如下

test1

```
请输入两个链表(各占一行):
1 2 5 -1
2 4 5 8 10 -1
交集链表为:
2 5
```

test2

```
请输入两个链表(各占一行):
1 3 5 -1
2 4 6 8 10 -1
NULL
```

test3

```
请输入两个链表(各占一行):  
1 2 3 4 5 -1  
1 2 3 4 5 -1  
交集链表为:  
1 2 3 4 5
```

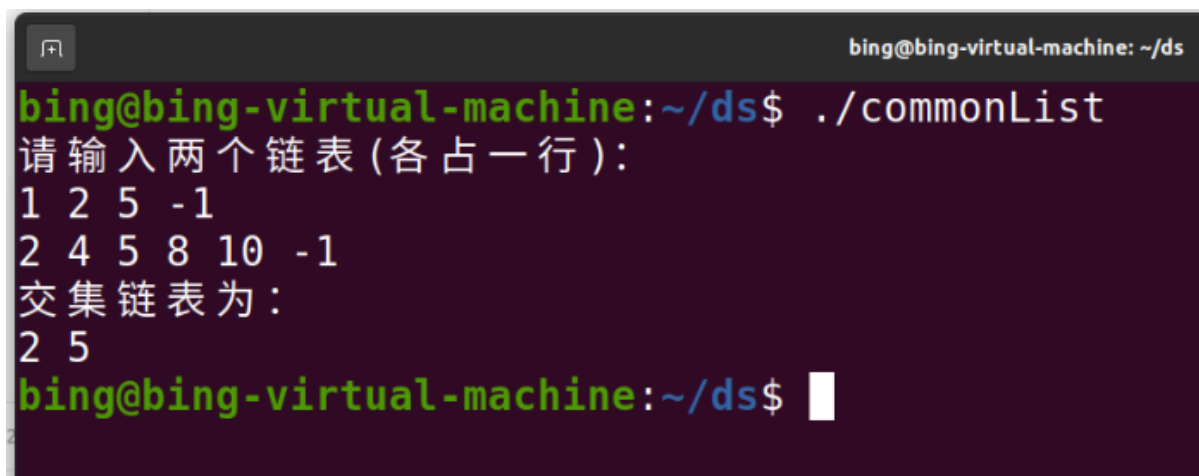
test4

```
请输入两个链表(各占一行):  
3 5 7 -1  
2 3 4 5 6 7 8 -1  
交集链表为:  
3 5 7
```

test5

```
请输入两个链表(各占一行):  
-1  
10 100 1000 -1  
NULL
```

Linux界面展示



```
bing@bing-virtual-machine: ~/ds  
bing@bing-virtual-machine:~/ds$ ./commonList  
请输入两个链表(各占一行):  
1 2 5 -1  
2 4 5 8 10 -1  
交集链表为:  
2 5  
bing@bing-virtual-machine:~/ds$
```

## 5. 代码架构

项目代码由链表类 `List` 和链表节点类 `Node` 组成，主要包含链表操作和交集计算的功能模块。

### 5.1. 链表节点类 `Node`

`Node` 类用于表示链表的节点，包含数据域和指向下一个节点的指针。

- 成员变量：
  - `data`: 存储节点的值。

- `next`: 指向下一个节点。
- 构造函数:
  - `Node(const T& value)`: 初始化节点的数据和指针。

## 5.2. 链表类 `List`

---

`List` 类实现了一个单链表，支持链表的基本操作和交集计算。

- 成员变量:
  - `head`: 指向链表的头节点。
  - `tail`: 指向链表的尾节点。
- 主要方法:
  - `Append()`: 在链表末尾添加新节点。
  - `Display()`: 打印链表中的元素。
  - `Clear()`: 释放链表的内存。
  - `GetHead()` 和 `GetBack()`: 分别返回链表的头指针和尾指针。

# 6. 功能模块介绍

---

## 6.1. 链表输入

---

`ListInput()` 函数用于从标准输入读取链表数据，并将数字添加到链表中，遇到 `-1` 停止。

```
1 void ListInput(List<int>& list) {
2     while (1) {
3         int num;
4         cin >> num;
5         if (num != -1) {
6             list.Append(num);
7         }
8         else {
9             break;
10        }
11    }
12 }
```

## 6.2. 交集计算

`GetCommonList()` 函数接受两个有序链表作为参数，返回它们的交集链表。函数通过比较两个链表的元素，找到它们的公共部分，并将其加入新的链表中。

- **逻辑描述：**
  - 初始化两个指针，分别指向两个链表的头节点。
  - 依次比较两个链表的当前节点，如果相等且未被添加过，则将该值添加到交集链表。
  - 如果第一个链表节点的值较小，则移动第一个指针；否则，移动第二个指针。
- **边界情况处理：**
  - 当任一链表为空时，返回空的交集链表。
  - 输出结果时，如果交集链表为空，则显示 `NULL`。

## 6.3. 打印输出

程序在主函数中调用 `Display()` 函数来打印交集链表的结果。如果交集链表为空，则输出 `NULL`。

## 7. 用户交互

用户通过命令行输入链表序列，使用 `-1` 作为结束标志。输入后的操作如下：

1. 输入两个链表序列，使用空格分隔数字，每行以 `-1` 结束。
2. 程序计算交集，并在标准输出中显示交集链表。
3. 如果没有交集，输出 `NULL`。

示例交互过程：

- **输入：**

```
1 | 1 2 5 -1
2 | 2 4 5 8 10 -1
```

- **输出：**

```
1 | 2 5
```

## 8. 性能考虑

由于两个链表是有序的，交集计算的时间复杂度为  $O(m + n)$ ，其中  $m$  和  $n$  分别为两个链表的长度。这种方法利用了有序性，无需额外的排序或复杂的查找操作。

链表数据结构在小规模数据时具有较好的性能，尤其适用于动态数据的插入和删除操作。

## 9. 错误处理和输入验证

为了保证程序的稳定性，对用户输入进行了严格的验证：

- 结束标志**：输入序列必须以 `-1` 结束。
- 数据类型**：所有输入值均需为正整数。
- 空序列**：如果输入链表为空，交集结果应输出 `NULL`。

程序在输入时对这些情况进行检查，并在不合法时给出提示。

## 10. 未来改进

未来可以对项目进行如下改进：

- 增加数据持久化**：将链表数据保存到文件中，以便后续加载和分析。
- 引入图形化界面**：通过可视化工具展示链表结构和交集计算过程，提升用户体验。
- 优化性能**：在处理大规模链表时，可以尝试更高效的数据结构（如哈希表）来提升性能。
- 扩展功能**：增加对差集、并集等操作的支持，进一步增强程序的功能。

## 11. 心得体会

本项目让我深入理解了链表数据结构的基本操作及其在实际问题中的应用。通过交集计算的实现，我学会了如何利用链表的有序性优化算法，如何处理输入验证和边界情况，以及如何设计易于扩展的代码架构。