

数据结构课程设计
项目说明文档

两个有序链表序列的交集

作者姓名:	<u>汪明杰</u>
学 号:	<u>1851055</u>
指导教师:	<u>张 颖</u>
学院专业:	<u>软件学院 软件工程</u>



同济大学
Tongji University

目录

1 项目分析	3
1.1 项目背景	3
1.2 项目需求分析	3
1.3 项目要求	4
1.3.1 功能要求	4
1.3.2 输入格式	4
1.3.3 输出格式	4
1.3.4 项目示例	4
2 项目设计	5
2.1 数据结构设计	5
2.2 类设计	5
2.2.1 结点类 (ListNode)	5
2.2.2 双向链表类 (List)	5
2.3 求链表交集算法	7
2.3.1 主要思想	7
2.3.3 代码	7
3 项目测试	9
3.1 一般情况	9
3.2 交集为空的情况	9
3.3 完全相交的情况	10
3.4 其中一个序列完全属于交集的情况	10
3.5 其中一个序列为空的情况	11
3.6 两个链表序列均为空的情况	11

1 项目分析

1.1 项目背景

链表是数据结构中极其重要的一种存储结构（也即物理结构），因此掌握对于链表的各种操作相当重要。

需要注意的是：链表是一种物理存储单元上非连续、非顺序的存储结构，数据元素的逻辑顺序是通过链表中的指针链接次序实现的。因此，对于链表的各种操作不像数组能够实现随机访问，而是要通过一个一个节点依次访问到下一个节点。

在工程中，常常需要应用到链表存储数据。也正是因为这一原因，有时候我们需要两个不同对象中的公共元素，这个时候往往就要求链表中各个元素的交集。

1.2 项目需求分析

针对于求两个有序链表交集的系统，应当满足以下需求：

- ✓ 功能完善

系统应当考虑到两个链表内元素的各种情况，如：链表 S1 长度大于链表 S2，链表 S1 长度等于 S2 等。

- ✓ 执行效率高

对于链表内节点数量比较多的情况，系统也应当具有比较快的处理速度，即时间复杂度应当较低。

- ✓ 健壮性

系统应当考虑当所输入的链表为空的情况。

1.3 项目要求

1.3.1 功能要求

已知两个非降序链表序列 S1 和 S2，设计函数构造出 S1 和 S2 的交集新链表 S3。

1.3.2 输入格式

输入分 2 行，分别在每行给出由若干个正整数构成的非降序序列，用 -1 表示序列的结尾（-1 不属于这个序列）。数字用空格间隔。

1.3.3 输出格式

在一行中输出两个输入序列的交集序列，数字间用空格分开，结尾不能有多余空格；若新链表为空，输出 NULL。

1.3.4 项目示例

序号	输入	输出	说明
1	1 2 5 -1	2 5	一般情况
	2 4 5 8 10 -1		
2	1 3 5 -1	NULL	交集为空的情况
	2 4 6 8 10 -1		
3	1 2 3 4 5 -1	1 2 3 4 5	完全相交的情况
	1 2 3 4 5 -1		
4	3 5 7 -1	3 5 7	其中一个序列完全属于交集的情况
	2 3 4 5 6 7 8 -1		
5	-1	NULL	其中一个序列为空的情况
	10 100 1000 -1		

2 项目设计

2.1 数据结构设计

如上述功能分析所述，该项目本质上是要实现一个链表数据结构，并在此结构上实现两个有序链表的交集算法。链表是计算机程序中常用的数据结构，根据节点之间连接方式和存储方式细分，有单链表、双向链表、循环链表、静态链表等种类。在本系统中，封装了一个双向链表，以实现所需要的功能。

2.2 类设计

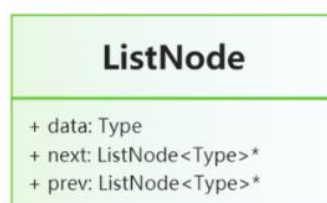
经典的链表一般包括两个抽象数据类型（ADT）——链表结点类（ListNode）与链表类（List），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。

为了实现代码的复用性，本系统实现了一个**链表**。采用 struct 描述链表结点类（ListNode），这样使得链表结点类（List）可以直接访问链表结点而不需要定义友元关系。本系统实现的链表结构各种操作的时间复杂度如下：

- ✓ 插入操作： $O(n)$
- ✓ 删除操作： $O(n)$
- ✓ 查询操作： $O(n)$
- ✓ 遍历操作： $O(n)$

2.2.1 结点类（ListNode）

链表的结点中存储的数据有链表数据、后继结点和前驱结点。其 UML 图如下所示：

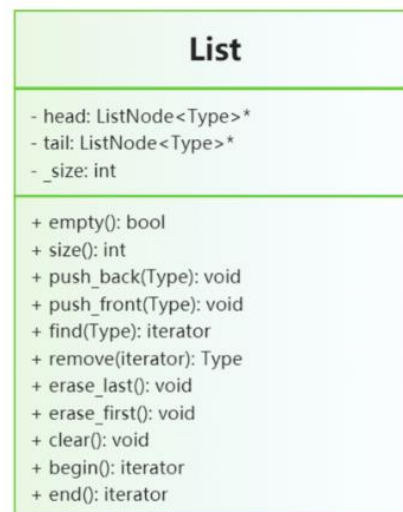


2.2.2 双向链表类（List）

链表的实现原理大同小异，不同之处在于：是否带头结点、是否带尾结点、每一个结点是否带前驱结点等。为了使得链表中各种操作的时间复杂度都尽可能

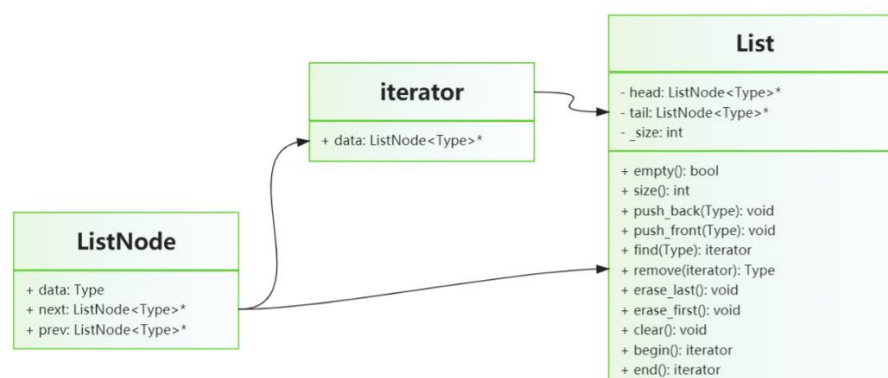
低，因此这里选择了带头结点和尾结点的双向链表来实现链表中的各种操作。也即：选择了牺牲空间来达到降低时间复杂度的效果。

链表的 UML 图如下所示：



为了便于对链表进行遍历、插入、删除、查找等操作，增加了一个 `iterator` 类。`iterator` 类内部存储一个链表节点指针，同时，通过运算符重载相应的自增、自减、判等等操作。

`iterator` 类、`ListNode` 类和 `List` 类的关系如下图所示：



其中，`List` 类中的主要函数如下所示：

- ◆ `inline int size()const`
返回链表中结点的个数，不包括头结点。
- ◆ `inline bool empty()const`
判断链表是否为空，也即链表中结点的个数是否为 0。
- ◆ `void push_back(Type data)`
在链表尾部插入新的数据，也即新增一个结点并且加入到链表末端。
- ◆ `void push_front(Type data)`
在链表头部插入新的数据，也即新增一个结点并且加入到链表的头部。
- ◆ `iterator find(const Type& data)const`

在链表中查找值为 `data` 的元素是否存在，返回该位置的迭代器，若查找失败返回空指针对应的迭代器。

◆ `Type remove(iterator index)`

移除迭代器所处位置的元素，返回移除位置元素的值。

◆ `void erase_last()`

移除链表末端的元素，即最后的结点。

◆ `void erase_first()`

移除链表首端的元素，即第一个结点。

◆ `void clear()`

清空链表，即删除链表中所有的结点。

◆ `iterator begin()`

返回链表第一个元素所在位置的迭代器。

◆ `iterator end()`

返回链表尾结点后的空结点的迭代器。

2.3 求链表交集算法

2.3.1 主要思想

初始化结果链表为空，分别定义两个链表上的迭代器 1 和 2。

当两个迭代器均不指向链表末尾时进行循环：比较迭代器 1 和迭代器 2 所指向的元素，若两个元素相等，则将该元素插入到新链表末端，两个迭代器加 1；否则只将元素较小的迭代器加 1。

2.3.3 代码

```
template<class Type>
List<Type> intersection(const List<Type> &l1, const List<Type> &l2)
{
    auto it1 = l1.begin();
    auto it2 = l2.begin();

    // 保存交集结果的链表
    List<Type> result;

    while (it1 != l1.end() && it2 != l2.end())
    {
        if (*it1 < *it2)
```

```
        ++it1;
    else if (*it1 > *it2)
        ++it2;
    else
    {
        result.push_back(*it1);
        ++it1;
        ++it2;
    }
}

return result;
}
```

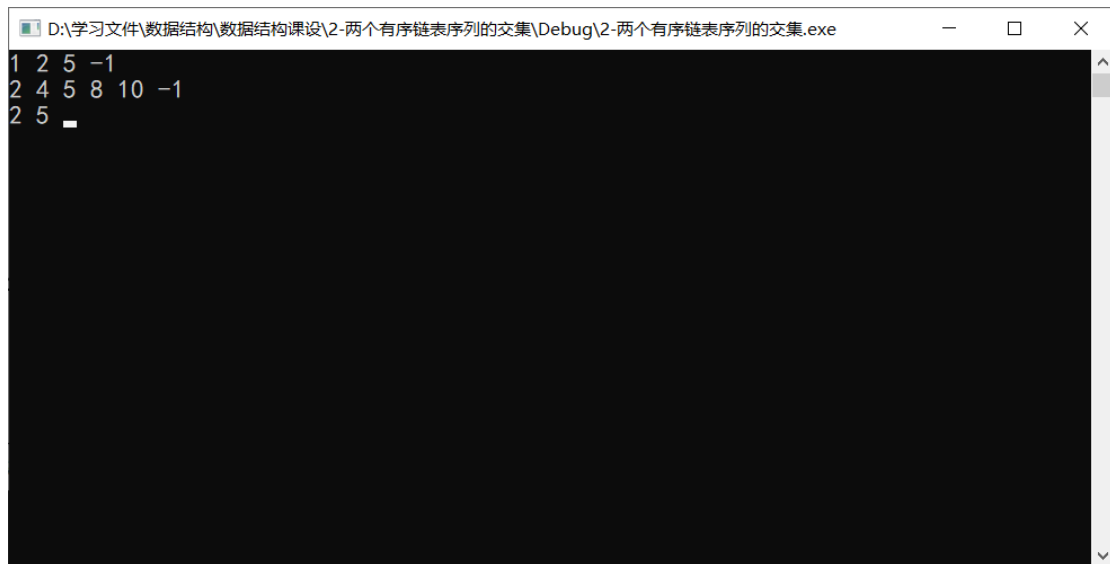

3 项目测试

3.1 一般情况

测试用例：1 2 5 -1 2 4 5 8 10 -1

预期结果：2 5

实验结果：



```
D:\学习文件\数据结构\数据结构课设\2-两个有序链表序列的交集\Debug\2-两个有序链表序列的交集.exe
1 2 5 -1
2 4 5 8 10 -1
2 5
```

3.2 交集为空的情况

测试用例：1 3 5 -1 2 4 6 8 10 -1

预期结果：NULL

实验结果：



```
D:\学习文件\数据结构\数据结构课设\2-两个有序链表序列的交集\Debug\2-两个有序链表序列的交集.exe
1 3 5 -1
2 4 6 8 10 -1
NULL
```

3.3 完全相交的情况

测试用例：1 2 3 4 5 -1 1 2 3 4 5 -1

预期结果：1 2 3 4 5

实验结果：

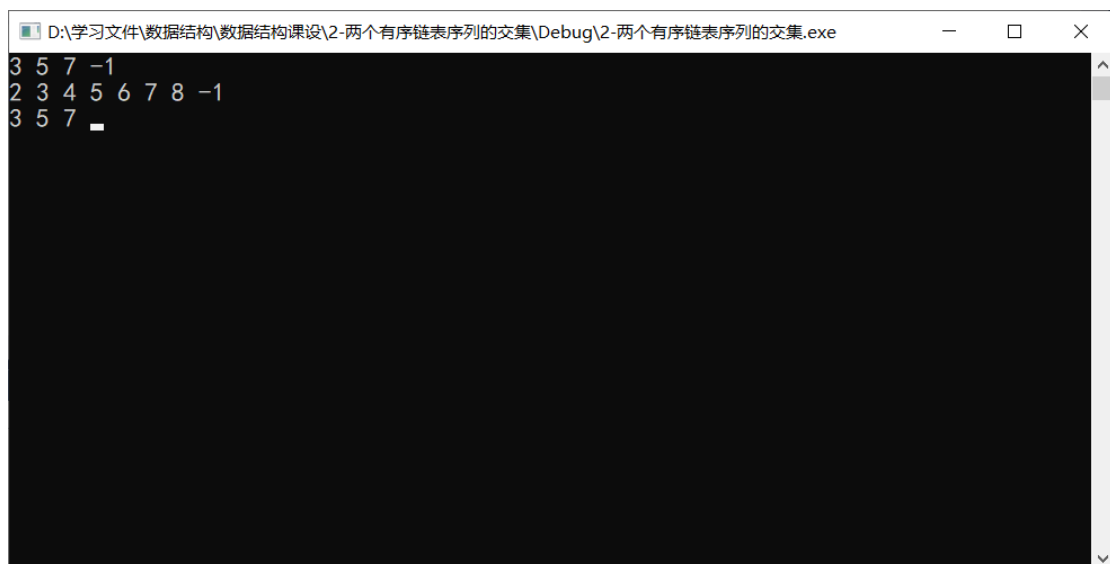


3.4 其中一个序列完全属于交集的情况

测试用例：3 5 7 -1 2 3 4 5 6 7 8 -1

预期结果：3 5 7

实验结果：

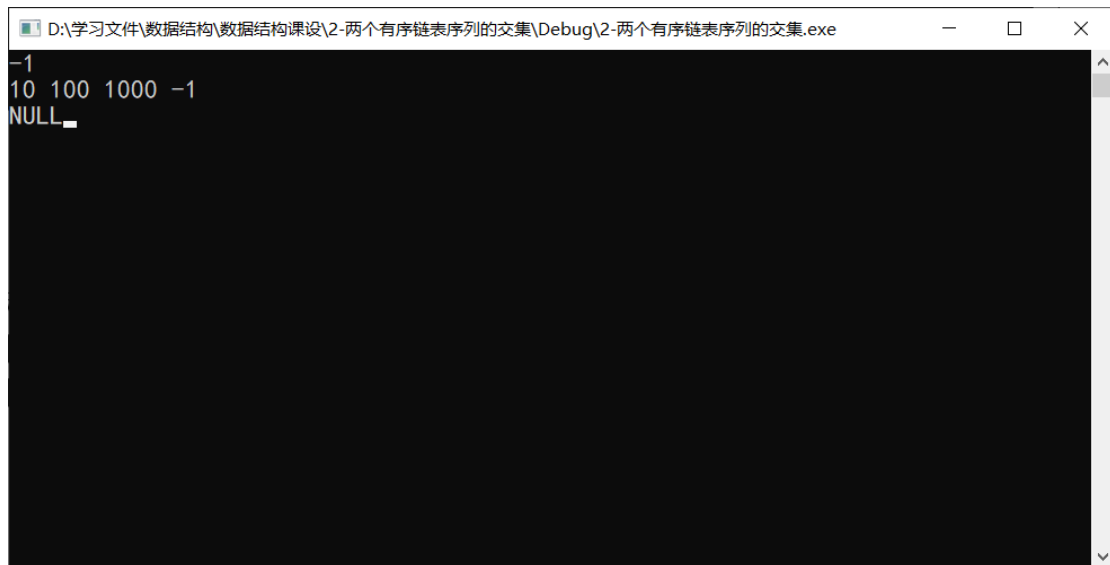


3.5 其中一个序列为空的情况

测试用例：-1 10 100 1000 -1

预期结果：NULL

实验结果：



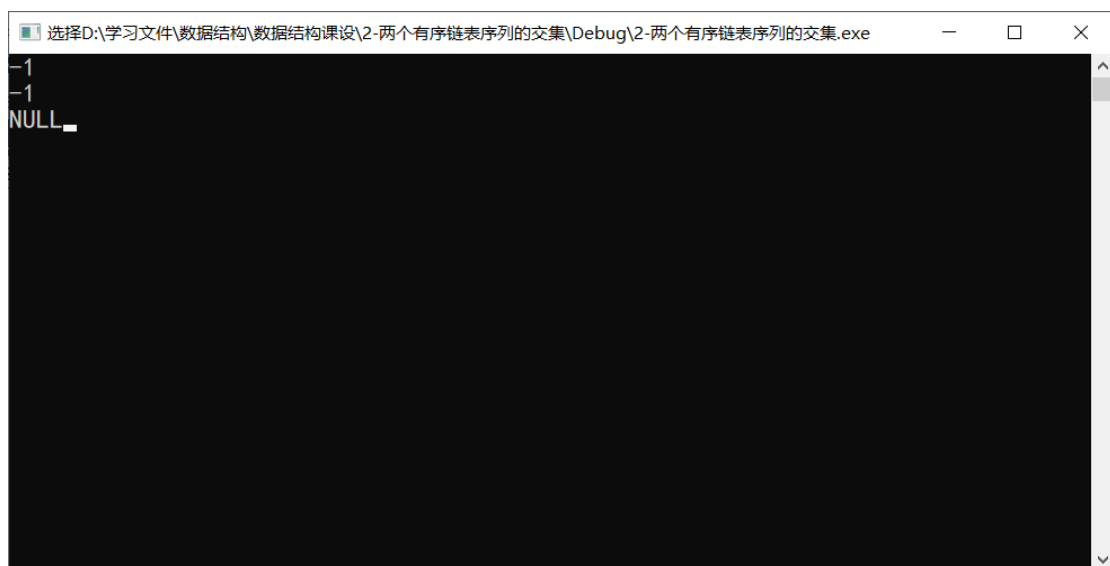
```
D:\学习文件\数据结构\数据结构课设\2-两个有序链表序列的交集\Debug\2-两个有序链表序列的交集.exe
-1
10 100 1000 -1
NULL_
```

3.6 两个链表序列均为空的情况

测试用例：-1 -1

预期结果：NULL

实验结果：



```
选择D:\学习文件\数据结构\数据结构课设\2-两个有序链表序列的交集\Debug\2-两个有序链表序列的交集.exe
-1
-1
NULL_
```