**项目说明文档**

**数据结构课程设计**

**——两个有序链表的交集**

### 作 者 姓 名 尹诚成

学 号 2351279

指 导 教 师 张 颖

学 院 专 业 计算机科学与技术学院 软件工程



二〇二三 年 十二 月 十三 日

**目录**

[1.项目分析 1](#_bookmark0)

* 1. .[项目背景分析 1](#_bookmark1)
  2. .[项目需求分析 1](#_bookmark2)
  3. .[项目功能分析 1](#_bookmark3)

1.3.1.链表输入功能 1

1.3.2.链表有序性检查功能 1

1.3.3.求两个有序链表交集功能 1

1.3.4.异常处理功能 2

[2.项目设计 2](#_bookmark12)

* 1. .[数据结构设计 2](#_bookmark13)
  2. .[结构体与类设计 2](#_bookmark14)

2.2.1 LinkNode 结构体的设计 2

2.2.2 List 类的设计 2

* 1. .[项目主体架构设计 3](#_bookmark32)

[3.项目功能实现 4](#_bookmark33)

3.1.项目主体架构的实现 4

3.2.链表输入功能的实现 4

3.3.链表有序性检查功能的实现 4

3.4.求两个有序链表交集功能的实现 5

3.5.异常处理功能的实现 5

4.项目测试 5

4.1. 链表输入功能测试 5

4.2. 链表有序性检查功能测试 6

4.3. 求两个有序链表交集功能测试 6

1. **项目分析**
   1. 项目背景分析

在数据处理和算法领域中，链表作为一种基础且重要的数据结构，广泛应用于各种场景。链表以其灵活的内存分配方式和高效的插入、删除操作而著称。在实际应用中，经常需要处理各种有序数据集合的交集问题，例如，在数据库查询优化、数据合并、信息检索等场景。传统的数组结构在处理这类问题时，可能会因为需要频繁的内存重新分配和元素移动而导致效率低下。因此，采用链表结构来处理非降序序列的交集问题，不仅可以提高处理效率，还能更好地适应动态数据变化的需求。

本项目旨在设计并实现一个函数，该函数能够接收两个非降序链表序列作为输入，并构造出这两个序列的交集新链表。这个问题不仅具有理论价值，也具有重要的实际意义，可以应用于多个领域的数据处理任务中。

* 1. 项目需求分析

基于以上背景分析，本项目需要实现需求如下：

(1)接收两个由若干个正整数构成的非降序链表序列，每个序列以-1作为结尾标识（注意，-1不属于序列本身）；

(2)利用链表的数据结构特性，设计算法高效计算两个非降序链表序列的交集，并构造一个新的非降序链表来存储结果。

(3)算法应能处理各种边界情况，如空链表、单个元素链表、完全重合链表等，确保在各种输入情况下都能正确输出交集序列。

(4)实现异常处理机制，确保系统稳定性和安全性，避免因用户输入错误导致系统崩溃或信息丢失；

* 1. 项目功能分析

本项目旨在通过模拟考试报名管理过程，实现对链表的输入、输入正确性检查以及有序性检查，并求出两个有序链表的交集，下面对项目的功能进行详细分析。

* + 1. 链表输入功能

允许用户输入一系列正整数，这些整数构成非降序链表序列，其中-1作为每个序列的结束标志（-1本身不作为链表的一部分）。该功能需要能够正确解析输入数据，将其转换为链表节点，并依次链接起来形成完整的链表结构。

* + 1. 链表有序性检查功能

检查输入元素格式的正确性，并判断输入的链表是否满足非降序序列，从而保证算法的正确性。

* + 1. 求两个有序链表交集功能

对于输入的两个有序链表，程序能求出这两个链表的交集并成功返回。

* + 1. 异常处理功能

实现异常处理机制，处理用户可能输入的非法信息，确保系统的稳定性和安全性。

1. **项目设计**

2.1.数据结构设计

基于项目分析，本项目选择使用链表作为数据结构而不是数组，主要基于以下几个考虑：

(1)动态大小需求：链表可以动态地分配内存，适应不同数量输入元素。

(2)插入和删除操作效率高：链表对于插入和删除操作效率较高，因为只需要调整节点的指针即可，无需移动元素；

(4)考虑系统维护的复杂度：考虑系统的维护和扩展性，链表可以更方便地进行扩展和修改。

链表适合在需要动态调整大小、频繁插入和删除操作的情况下使用，基于上述分析，在本项目中选择链表作为数据结构更合适。

2.2.结构体与类设计

* + 1. LinkNode结构体的设计

LinkNode 结构体是一个用于构建链表节点的模板结构体。该结构体用于表示链表中的每个节点，其中包括节点存储的数据以及指向下一个节点的指针。本项目希望链表结点类可以直接访问链表结点，所以使用 struct 而不是 class 描述链表结点类。其数据成员和构造函数定义及含义如下：

T data：数据域，存储节点的数据。

LinkNode<T>\* link：指针域，指向下一个节点的指针。

LinkNode(LinkNode\* ptr = NULL)：构造函数，初始化指针域。

LinkNode(const T& item, LinkNode\* ptr = NULL)：构造函数，初始化数据域和指针域。

* + 1. List类的设计

该通用模板类 MyList 用于表示单链表。此链表以附加的头节点作为起点。链表节点由 MyLinkNode 结构体表示，其中包含数据和指向下一个节点的指针。该链表提供了一系列基本操作函数，包括节点的插入、删除、查找、访问等，以及链表的构造和析构，满足了链表的常见操作需求。其数据成员、构造函数、析构函数、公有成员函数，运算符重载定义及含义如下：

LinkNode\* first：指向链表的第一个节点（头节点）的指针。

LinkNode\* last：指向链表的最后一个节点的指针。

List()：默认构造函数，创建一个空链表。

List(const T& item)：转换构造函数，创建一个只包含一个元素的链表。 List(List& L)：复制构造函数，通过复制另一个链表创建新链表。

~List()：析构函数，释放链表的内存资源，包括所有节点的内存。

void makeEmpty()：清空链表，释放所有结点的内存。

int getLength()const：获取链表中节点的个数。

LinkNode<T>\* getHead()const：获取链表头节点的指针。

LinkNode<T>\* getTail()const：获取链表尾节点的指针。

LinkNode<T>\* search(T item)const：搜索链表中值为 item 的节点，返回指向该节点的指针，若不存在则返回 NULL。

LinkNode<T>\* locate(int i)const：返回链表中第 i 个节点的指针，若 i 超出链表长度或小于 0，则返回 NULL。

int findData(T& item)const：获取链表中值为 item 的节点所在的位置，并通过返回值返回。

bool getData(int i, T& item)：获取链表中第i个节点的数据，并通过引用返回。返回值为操作是否成功。

void setData(int i, T& item)：设置链表中第i个节点的数据。返回值为操作是否成功。

bool insert(int i, T& item)：在链表中第i个节点后插入新节点。返回值为操作是否成功。

bool remove(int i, T& item)：删除链表中第i个节点，并通过引用返回其数据。返回值为操作是否成功。

bool isEmpty()const：检查链表是否为空。

void output()const：输出链表中所有节点的数据。

* 1. 项目主体架构设计

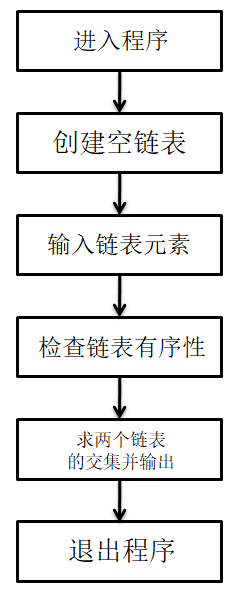


图 2.3 项目主体架构设计流程图

项目主体架构设计为：

(1)进入求取两个有序链表交集的程序；

(2)创建三个空链表，分别存放用户输入的两个链表和链表交集；

(3)依次输入两个链表中的元素，以-1结束输入(-1不属于链表元素)；

(4)检查输入链表是否满足有序性，如果不满足，程序给出提示信息并让用户重新输入；

(5)求取用户输入的两个有序链表的交集并输出其元素

(6)退出程序

1. **项目功能实现**

3.1.项目主体架构的实现

项目主体架构实现思路为：

(1)在main 函数中调用two\_ordered\_lists\_intersection函数进入求取两个有序链表交集的程序；

(2)创建三个存放整型元素的空链表list1, list2, intersection，分别代表用户输入的两个链表和链表交集；

(3) 依次输入两个链表中的元素，每个元素中间用空格隔开，以-1结束输入(-1不属于链表元素)；

(4)通过检查元素是否为整数以及调用check\_list\_is\_ordered函数来检查链表是否满足有序性，如果满足，程序继续进行，如果不满足，程序会清空当前输入的链表，并输出提示信息让用户重新输入；

(5)求取两个有序链表的交集，并按照指定格式输出：每个元素之间用空格隔开，结尾不能又空格，若交集为空，则输出NULL。

(6)退出程序。

3.2.链表输入功能的实现

输入链表功能的函数名为input\_list，其实现思路为：

(1)输出提示信息说明链表的输入元素范围和结束标志；

(2)利用无限循环输入链表元素直到输入结束标志位置；

(3)每输入一个元素，都会检查其是否为输入范围内的整数，若是，则将其插入链表内，程序继续运行；若不是，则清空当前链表并输出提示信息让用户重新输入；

(4)每向链表插入一个元素，都会调用check\_list\_is\_ordered函数检查其是否满足有序性，若满足，程序继续运行；若不满足，则清空当前链表并输出提示信息让用户重新输入；

3.3.链表有序性检查功能的实现

检查链表有序性的函数名为check\_list\_is\_ordered其实现思路为：

(1)判断链表是否为空，若为空，则返回ture；

(2)获取链表的第一个有效节点，并初始化前一个节点值，用于后续比较；

(3)遍历链表，每次遍历都会检查当前节点数据是否小于前一个数据的值，若小于，说明链表不是非降序序列，函数返回false；若不小于，则更新两个指针所指向的节点的值；

(4)若遍历结束，说明该链表是非降序的，函数返回true。

3.4.求两个有序链表交集功能的实现

求两个有序链表交集的函数名为find\_the\_intersection\_of\_two\_ordered\_lists其实现思路为：

(1)初始化两个链表的节点，使用LinkNode<int>\* current1 = list1.getHead()->link和LinkNode<int>\* current2 = list2.getHead()->link获取两个链表的第一个有效节点；

(2)遍历两个链表,直到一个链表到达末尾；

(3)每次循环都会对当前节点的数据进行比较。如果current1->data == current2->data，说明两个链表中有相同的值，将该值插入到链表intersection中，然后同时移动两个链表的指针；如果两个指针指向的值不相等，则移动指向值较小的指针；

(4)当一个链表的当前节点为NULL时，说明检查完毕，结束循环。

3.5.异常处理功能的实现

在进行List类中链表节点的动态内存申请时，程序使用new(std::nothrow)来尝试分配内存。new(std::nothrow)在分配内存失败时不会引发异常，而是返回一个空指针（NULL或nullptr），代码检查指针是否为空指针，如果为空指针，意味着内存分配失败，这时程序将执行以下操作：

(1)向标准错误流std::cerr输出一条错误消息"Error: Memory allocation failed."，指出内存分配失败；

(2)调用exit函数，返回错误码MEMORY\_ALLOCATION\_ERROR（通过宏定义方式定义为-1），用于指示内存分配错误，并导致程序退出。

1. **项目测试**

4.1.链表输入功能测试

分别输入超过上下限的整数、浮点数、字符、字符串，可以验证程序对输入非法的情况进行了处理。

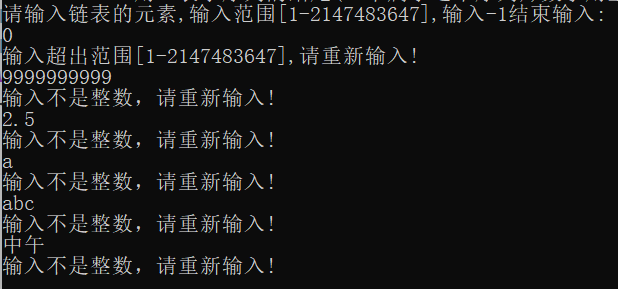


图 4.1.1 链表输入功能测试（输入非法）

当输入合法时，程序继续运行

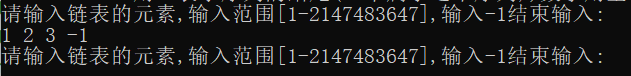


图 4.1.2 链表输入功能测试（输入合法）

4.2.判断链表有序性功能测试

通过输入不满足非降序的链表序列，可以验证程序对链表的有序性进行了处理。

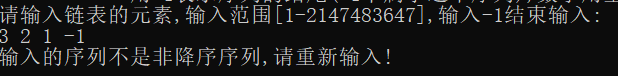


图 4.2 判断链表有序性功能测试

4.3.求两个有序链表交集功能测试

通过输入两个非降序的链表，可以验证程序能正常求出并输出两个链表的交集。

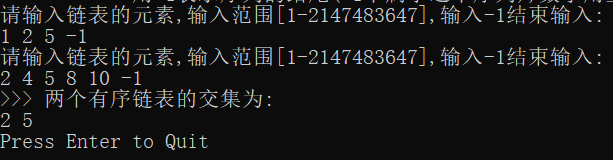


图 4.3.1 求两个有序链表交集功能测试（实验数据1）

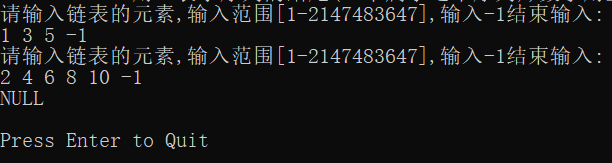


图 4.3.2 求两个有序链表交集功能测试（实验数据2）

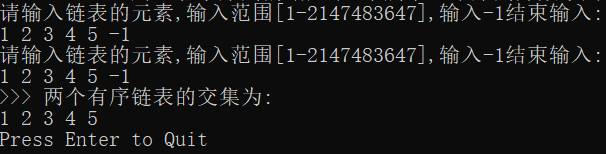


图 4.3.3 求两个有序链表交集功能测试（实验数据3）

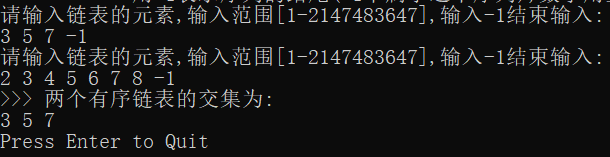


图 4.3.4 求两个有序链表交集功能测试（实验数据4）

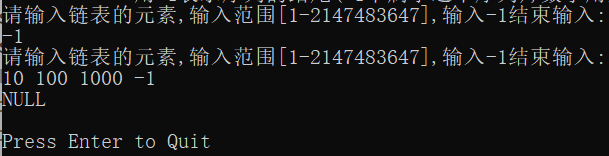


图 4.3.5 求两个有序链表交集功能测试（实验数据5）