沈阳航空航天大学

人工智能学院

**实验报告**

|  |  |
| --- | --- |
| 课程名称 | 算法导论 |
| 专 业 | 物联网工程 |
| 班 级 | 物联网2202 |
| 学 号 | 223428010210 |
| 学生姓名 | 陈梓欣 |
| 指导教师 | 孙恩岩 |
| 实验时间 | 2024年5月26日 1、2节 |
| 实验地点 | 机械馆410-3 |

# 一、实验名称

有序链表的合并

# 二、实验目的

理解单链表的存储结构，熟练掌握单链表的基本操作。

# 三、实验内容和要求

熟悉C++集成开发环境Visual Studio，掌握在Visual Studio环境下开发C++程序的方法，两个有序表合并为一个有序表。

# 四、实验环境

Microsoft Visual Studio 2022

# 五、实验设计

本次实验主要涉及算法和数据结构设计，重点在于链表这一数据结构的设计和实现。以下是实验中涉及的关键点：

算法和数据结构设计

1.合并两个有序链表：

2.创建一个新的链表作为合并结果的头节点和尾节点。

3.依次比较两个链表的节点，将较小的节点添加到合并链表的尾部。

4.循环直到其中一个链表为空。

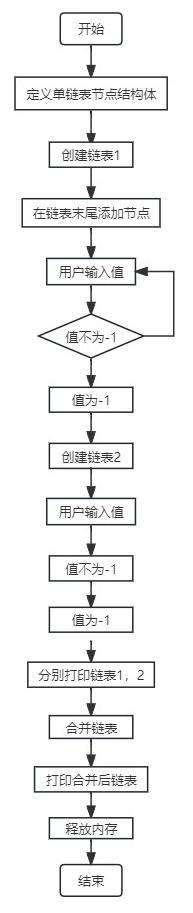
5.将非空链表的剩余部分直接连接到合并链表的尾部。

数据结构：

1.链表节点结构体：

2.包含一个整数值和指向下一个节点的指针。

3.方便存储和操作链表中的元素。



**图1 主模块流程图**

程序代码要加入必要注释，程序示例如下：

#include<iostream>

using namespace std;

//定义单链表节点结构体

struct ListNode

{

int value;

ListNode\* next;

ListNode(int x) : value(x), next(nullptr) {}

};

//在链表末尾添加节点

void appendNode(ListNode\*& head, ListNode\*& tail, int value) {

ListNode\* newNode = new ListNode(value); // 声明 newNode

if (!head)

{

head = newNode;

}

else

{

tail->next = newNode;

}

tail = newNode;

}

// 打印链表

void printList(ListNode\* head)

{

while (head)

{

cout << head->value << " ";

head = head->next;

}

cout << endl;

}

//合并两个有序链表

ListNode\* mergeSortedLists(ListNode\* list1, ListNode\* list2)

{

if (!list1) return list2;

if (!list2) return list1;

ListNode\* head = nullptr, \* tail = nullptr;

if (list1->value < list2->value) {

head = tail = list1;

list1 = list1->next;

}

else {

head = tail = list2;

list2 = list2->next;

}

while (list1 && list2) {

if (list1->value < list2->value) {

tail->next = list1;

list1 = list1->next;

}

else {

tail->next = list2;

list2 = list2->next;

}

tail = tail->next;

}

if (list1) tail->next = list1;

if (list2) tail->next = list2;

return head;

}

int main()

{

ListNode\* list1 = nullptr, \* list1Tail = nullptr;

ListNode\* list2 = nullptr, \* list2Tail = nullptr;

cout << "Enter the first sorted list(end with -1): ";

int value;

while (cin >> value, value != -1)

{

appendNode(list1, list1Tail, value);

}

cout << "Enter the second sorted list(end with -1): ";

while (cin >> value, value != -1)

{

appendNode(list2, list2Tail, value);

}

// 打印原始链表

cout << "List 1: ";

printList(list1);

cout << "List 2: ";

printList(list2);

// 合并链表

ListNode\* mergedList = mergeSortedLists(list1, list2);

// 打印合并后的链表

cout << "Merged List: ";

printList(mergedList);

// 释放链表内存

ListNode\* current = mergedList;

while (current) {

ListNode\* toDelete = current;

current = current->next;

delete toDelete;

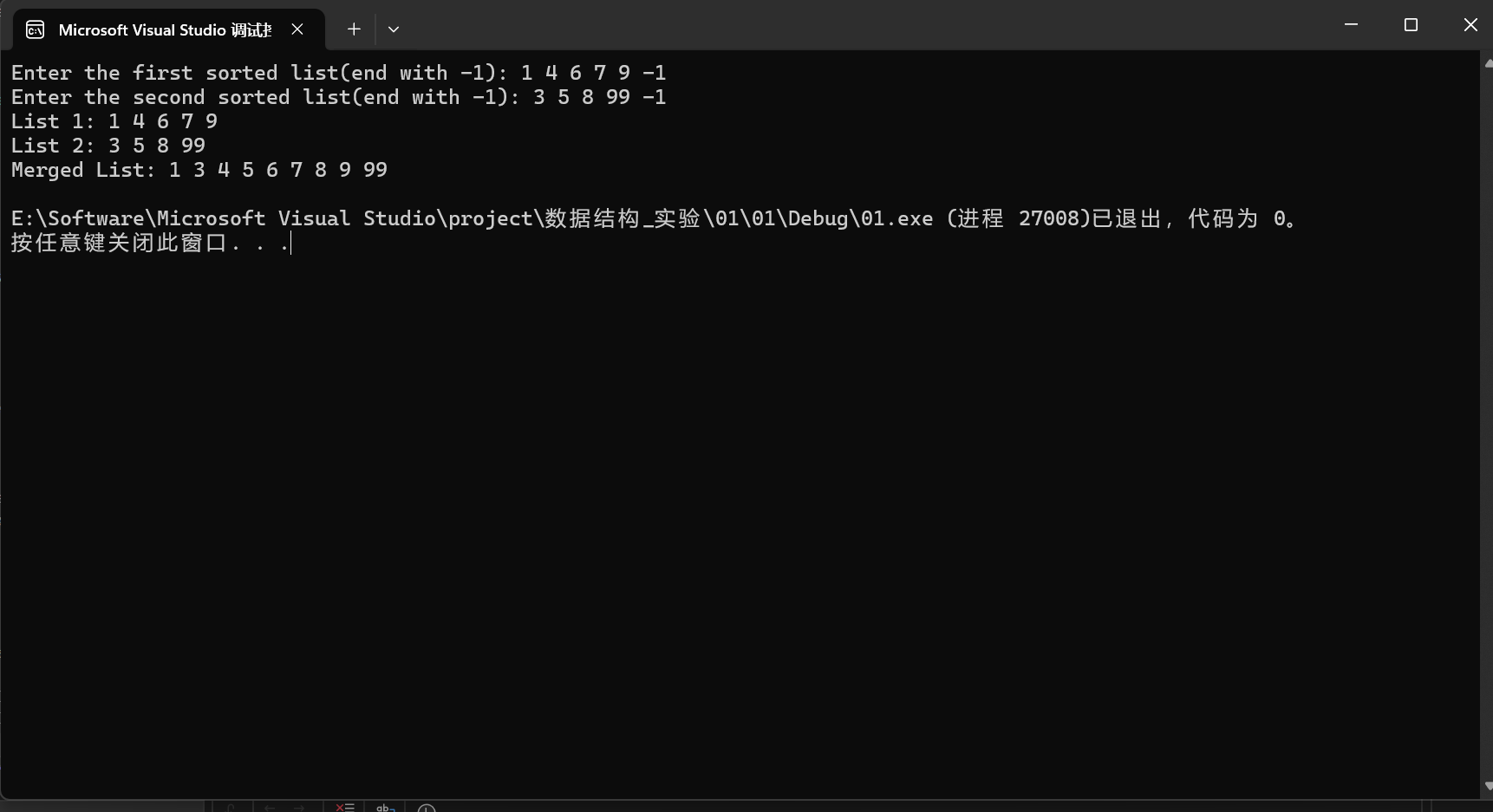
}

return 0;

}

# 六、实验步骤及实验结果

本节用文字和屏幕截图详实记录实验的设计结果、完成过程（步骤）和每一步的测试结果



**图2 打印信息**

# 七、结论

本次实验旨在验证链表数据结构的灵活性和合并算法的有效性。首先，通过实现合并有序链表的算法，我们利用了链表的动态特性，使得插入、删除等操作更为高效。合并算法通过比较节点值实现有序合并，从而确保合并后的链表仍然保持有序性，并包含了两个原始链表的所有元素。

在实验过程中，我们成功验证了算法的正确性。合并后的链表经过检查，确保节点顺序正确，并且所有原始链表中的元素都正确地包含在了合并后的链表中。

综合以上结果，实验验证了链表数据结构的灵活性和合并算法的可行性。链表的动态特性使得它在插入和删除操作时更为高效，而合并算法的正确性和有效性则为实际应用提供了可靠的支持。这项实验不仅在理论上证明了链表的优势，也在实践中展示了其在解决实际问题时的价值和可靠性。