OOP 第六周作业文档

2019010175 孔瑞阳 土木 92

一、项目信息

1、功能说明

实现了一个学生管理系统 CP_StudentSystem。

支持以下几种操作：

1： 添加若干位学生(学号成绩)，以 0 结束。

2： 删除第 1 位指定学号的学生。

3： 删除所有指定成绩的学生。

4： 显示第 1 位指定学号的学生的成绩。

5： 显示所有指定成绩的学生的学号。

6： 显示所有学生信息。

-1：退出。

2、软件构件介绍

文件	功能介绍
CP_Integer.h/cpp	整形类，实现整数的操作
CP_Student.h	学生类，包含学生的信息（学号、成绩）
CP_StudentDoubleLink.h/cpp	学生类的双向链表，用来存储所有学生的信息
CP_StudentSystem.h/cpp	学生管理系统的类，用来实现以上操作
CP_studentSystemMain.cpp	主程序

3、测试环境

CPU	Intel(R) Core(TM)i7-9750H CPU @ 2.6Ghz 6 核 12 线程
GPU	NVIDIA GeForce RTX2070
RAM	DDR4 16G+16G
Operating System	Microsoft Windows 版本 1809 (OS 内部版本 17763.1098)
Compiler	MSVC++ 14.24

二、规定

1、在整个过程中，双向链表内的所有学生按学号升序排列。

2、由于第 1 个规定，和“一个学生（学号）多个成绩”以及操作 2/4 会产生 undefined behavior，也就是可能输出/删除的学生不一定，并且，这种情况会现实情况不符，所以，这里规定：当插入的学生的学号已经在双向链表中时，插入这个学生表示为修改学生的成绩，即把这个学号的学生的成绩改为新输入的成绩。

3、尽管没有要求，操作 5/6 中的所有学号按照升序排列。

三、测试

1、较为完备的测试思路

由于采用 CP_Integer 类进行输入，并且，这个类在前几次作业中已经进行了充分的测试，所以在这次测试的过程中，只会考虑输入的所有内容为正整数（可以不合法）的情况。

首先，我们将操作分为 2 类：

1、修改类：操作 123

2、输出类：操作 456

我们先在**保证链表内结构没有问题**的情况下，验证操作 456 的正确性。

如何保证？先只进行一次 1 操作（插入），再进行一次 6 操作（输出所有学生信息），来验证在**没有 23 干扰**的情况下 1 操作的正确性。如果没有出现问题，那么可以说明，当前链表内的结构是没有问题的，再进行 456 操作的测试。

步骤 1：进行操作 1、操作 6 的联合测试。

步骤 2：进行操作 4、5、6 的测试。

在保证操作 456 的正确性之后，我们就可以运用这三个操作，来验证链表结构的正确性。

接下来，我们不仅要保证修改类操作明面上的正确性，也要保证这些操作不会出现链表结构的错误。结构错误可能会有什么体现？

比如说，我先删除了一个结点，进行 6 操作之后的确消失了。

但是，可能再进行其他的修改操作之后，这个本应该被删除的结点又出现了，就说明在操作的过程中可能出现了链表结构的错误。

那么，如何验证这样的操作。

换一种说法，一个修改操作，**不会对之后的修改操作产生错误的影响**。

也就是说，可以进行如下的验证：

在进行操作 x 之后，立刻进行操作 y，验证操作 y 是否出现问题。

步骤 3：进行操作 1、2、3 的联合测试。

下面，给出操作 1、2、3 的等价类。

操作	等价类编号	描述
操作 1	1.1	插入已经存在的编号
	1.2	插入不存在，比所有存在的都大的学号
	1.3	插入不存在，比所有存在的都小的学号
	1.4	插入不存在，并处于中间的学号
操作 2	2.1	删除存在的学生
	2.2	删除不存在，但是以前存在过的学生
	2.3	删除不存在，并且从来不存在的学生
操作 3	3.1	删除存在的成绩
	3.2	删除不存在，但是以前存在过的成绩
	3.3	删除不存在，并且从来不存在的成绩

2、测试过程

步骤 1：进行操作 1、操作 6 的联合测试。

对于操作 1，我们在测试的过程中，也验证是否一直按学号升序排列

(1) 等价类划分：

① 输入为负数

② 输入合法，且这个学号没有出现过

③ 输入合法，但是这个学号之前已经实现过了

④ 输入为 0

(2) 测试：

等价类	选取案例	输出结果（反馈 或者 操作 6 的输出）
等价类①	-10	错误:输入格式有误!
	-114514	错误:输入格式有误!
等价类②	1 1	[1]:学号(1)， 成绩(1)
	1 1 10 1	[1]:学号(1)， 成绩(1) [2]:学号(10)， 成绩(1)
	1 1 10 1 9 2	[1]:学号(1)， 成绩(1) [2]:学号(9)， 成绩(2) [3]:学号(10)， 成绩(1)
	1 1 10 1 9 2 11 3	[1]:学号(1)， 成绩(1) [2]:学号(9)， 成绩(2) [3]:学号(10)， 成绩(1) [4]:学号(11)， 成绩(3)
等价类③	1 1 10 10	[1]:学号(1)， 成绩(1) [2]:学号(10)， 成绩(10)
	1 1 10 10 1 2 10 9	[1]:学号(1)， 成绩(2) [2]:学号(10)， 成绩(9)
等价类④	0	（退出输入，程序继续运行）
无输入进行 6 操作		目前还没有学生。

步骤 2：进行操作 4、5、6 的测试。

信息案例	等价类	选取案例
1 1 2 2 3 3 7 1 8 1 9 2	操作 6	[1]:学号(1)， 成绩(1) [2]:学号(2)， 成绩(2) [3]:学号(3)， 成绩(3) [4]:学号(7)， 成绩(1) [5]:学号(8)， 成绩(1) [6]:学号(9)， 成绩(2)
	操作 4，学号不存在（4）	没有找到该学号的学生。
	操作 4，学号存在（3）	这位学生的成绩是:3
	（7）	这位学生的成绩是:1
	操作 5，学生不存在（4）	成绩为此的学生的学号是:

	操作 5，学生有一个（3）	成绩为此的学生的学号是: [1]：3
	操作 5，学生有多个（1）	成绩为此的学生的学号是: [1]：1 [2]：7 [3]：8

步骤 3：进行操作 1、2、3 的联合测试。

采用这个序列来验证操作 1、2、3 有没有相互影响：**1332112231**

可以注意到，这个序列已经完全包含了 11/12/13/21/22/23/31/32/33。

步骤	操作	测试	输出结果
1	1 1 1 2 2 3 3 7 1 8 1 9 2 (0)	操作 6	[1]:学号(1)， 成绩(1) [2]:学号(2)， 成绩(2) [3]:学号(3)， 成绩(3) [4]:学号(7)， 成绩(1) [5]:学号(8)， 成绩(1) [6]:学号(9)， 成绩(2)
2	3 4 3 2	操作 4 (2)	没有找到该学号的学生。
3	3 3	操作 5 (2)	成绩为此的学生的学号是:
4	2 2 2 1	操作 4 (1) 操作 6	没有找到该学号的学生。 没有找到该学号的学生。 [1]:学号(7)， 成绩(1) [2]:学号(8)， 成绩(1)
5	1 1 1 1 1 10 2	操作 6 **此处初次测试发现问题** (插入最小值时没有特判 在第一个位置插入的情况)	[1]:学号(1)， 成绩(1) [2]:学号(7)， 成绩(1) [3]:学号(8)， 成绩(1) [4]:学号(10)， 成绩(2)
6	1 1 2 7 3 4 1 (0)	操作 6	[1]:学号(1)， 成绩(2) [2]:学号(4)， 成绩(1) [3]:学号(7)， 成绩(3) [4]:学号(8)， 成绩(1) [5]:学号(10)， 成绩(2)
7	2 4	操作 4 (4)	没有找到该学号的学生。
8	2 4 2 11 2 1	操作 5 (1) 操作 6	没有找到该学号的学生。 没有找到该学号的学生。 成绩为此的学生的学号是: [1]:8 [1]:学号(7)， 成绩(3) [2]:学号(8)， 成绩(1) [3]:学号(10)， 成绩(2)

9	3 1 3 1	 操作 4 (8)	 没有找到该学号的学生。
10	1 8 2 1 1 4 5 1 4 (0)	操作 6 操作 5 (2) 操作 4 (9) 操作 4 (1)	[1]:学号(1), 成绩(4) [2]:学号(4), 成绩(5) [3]:学号(7), 成绩(3) [4]:学号(8), 成绩(2) [5]:学号(10), 成绩(2) 成绩为此的学生的学号是: [1]: 8 [2]: 10 没有找到该学号的学生。 这位学生的成绩是:4
非法	-10	错误:输入格式有误!	
	10	错误:输入格式有误!	
	-1	程序正常退出	

以及给出上述测试和等价类的对应关系

等价类	涉及的测试步骤
1.1	5、6、10
1.2	1、5
1.3	5、10
1.4	6、10
2.1	4、7、8
2.2	4、8
2.3	8
3.1	2、3、9
3.2	9
3.3	2

以上，是相对完备的测试。