浙江工业大学

C++ 课程设计实验报告

(2021级)

实验题目 教职工信息管理系统

学生姓名 葛骏

学　　号 202105130406

专业班级 计科 2103

所在学院 计算机科学与技术学院

提交日期 2023 年 5 月 9 日

目　录

1　课程设计题目和内容 1
2　开发测试环境 1
3　实验课题分析 1
4　实验主要模块 2
4.1　链表 2
4.1.1　链表模板类源代码组织 2
4.1.2　链表的实现 2
4.2　教职工类 4
5　实验调试和运行 4
5.1　实验调试说明 4
6　实验总结 4

1 课程设计题目和内容

我抽到的题目是第六题，即教职工信息管理系统。 要求如下：

基本要求： 定义教职工（employee ）类，其中至少包括姓名、性别、工号、联系电话、所在学院、系和学历。

1. 设计菜单实现功能选择；
2. 输入功能：输入职工信息，并保存到文件中；
3. 查询功能：
   - 能够根据工号精确查询职工信息；
   - 能够根据姓名、学院、系、学历各项信息查询职工信息；
   - 系进行学历统计，计算各学历的人数；
4. 根据教职工的学历排序输出；
5. 根据工号修改职工信息；
6. 根据工号删除职工信息；
7. 所有的增加、修改、删除能同步到文件；也从文件读取数据到程序。
8. 考虑各项数据的合法性检查

2 开发测试环境

开发测试环境:

- 操作系统： Arch Linux x86_64
- Kernel: 6.3.1-arch1-1
- Shell: fish 3.6.1
- CPU: AMD Ryzen 7 5800H with Radeon Graphics (16) @ 3.200GHz
- GPU: NVIDIA GeForce RTX 3060 Laptop GPU
- Memory: 16GiB
- C++ 编译器: Clang version 15.0.7
- 编辑器: Neovim v0.9.0
- 编码: UTF-8
- 版本控制: Git 2.40.1
- 构建工具: Bazel 6.2
- 其他:
  - C++ 标准: C++14
  - libprotoc: 3.21.12
  - flutter: 3.7.12

3 实验课题分析

本题要求实现一个教职工信息管理系统，要求底层的数据结构一定要是链表。 那么我将先实现一个双向链表的模板类。

再实现一个教职工类，其中包含需要的信息。

为了实现多种功能，需要专门实现一个管理类。如果用现代的设计模式来说，在 MVC 架构中，这个类就是Controller。

考虑各项数据的合法性检查，需要另外实现一个简单的cpp文件。

4 实验主要模块

4.1 链表

我实现了一个双向链表作为底层数据结构。

4.1.1 链表模板类源代码组织

链表源文件存放在 `lib` 文件夹中。 由于链表是一个模板类，所以所有实现都在头文件中。 但是为了定义和实现的分离，使用了 `LinkedList.hpp`引用 `LinkedListDetail.hpp`。 后者再引用 `LinkedListTemplate.hpp`

头文件的引用关系大致如代码 4-1 所示。

```
// in LinkedList.hpp
@pragma once
#include "LinkedListDetail.hpp"

// in LinkedListDetail.hpp
#include "LinkedListTemplate.hpp"
// detailed implementation ...

// in LinkedListTemplate.hpp
@pragma once
// template implementation ...
```

代码 4-1 链表头文件引用关系

接下来我将说明链表的实现。

4.1.2 链表的实现

```
#pragma once
// just a simple double-direction linked list
template <typename T> class LinkedList {
public:
  struct Node { // the node of the linked list
    T data;
    Node *next;
    Node *prev;
    Node() : next(nullptr), prev(nullptr){}; // default constructor
```

```
    Node(T data) : data(data), next(nullptr), prev(nullptr){};
  };
  LinkedList();
  LinkedList(const LinkedList &);
  LinkedList &operator=(const LinkedList &);
  ~LinkedList();
  void pushBack(T);
  void remove(T);
  Node *find(T) const;
  Node *begin() const;
  Node *end() const;
  int getSize() const;

private:
  Node *head = new Node(); // the head of the linked list.
  Node *tail = new Node(); // the tail of the linked list.
  int size = 0;            // the size of the linked list.
};
```

代码 4-2　链表类的定义

如 代码 4-3 所示， 我实现的是一个双向链表。 每个节点都有一个指向前一个节点和后一个节点的指针。

并且对于每一个链表，含有两个特殊的节点，即头节点和尾节点。 这样做的好处是可以方便获取头尾节点，并且在遍历的过程中不会出现空指针。 （如果出现空指针则很容易产生 segmentation fault）

链表只给出六个方法：插入方法pushBack，删除方法remove， 获取大小方法getSize，查找方法find，头节点begin，尾节点end。

插入方法的实现如代码 4-3 所示。

```
template <typename T> void LinkedList<T>::pushBack(T data) {
  Node *p = new Node(data);
  p->next = tail;
  p->prev = tail->prev;
  tail->prev->next = p;
  tail->prev = p;
  size++;
}
```

代码 4-3　插入方法的实现

插入的位置是在尾节点的前面，即tail->prev的位置。

删除方法的实现如代码 4-4 所示。

```
template <typename T> void LinkedList<T>::remove(T data) {
  Node *p = find(data);
  if (p == nullptr) {
    return;
  }
  p->prev->next = p->next;
  p->next->prev = p->prev;
  delete p;
  size--;
}
```

代码 4-4　删除方法的实现

删除函数特判了p是否为空指针，如果为空则直接返回。删除的实现是比较简单的，只需要将前后的两节点连接起来即可。不过需要注意的是要手动释放内存，否则会造成内存泄漏。

查找方法的实现如代码 4-5 所示。

```
template <typename T>
typename LinkedList<T>::Node *LinkedList<T>::find(T data) const {
  Node *p = head->next;
  while (p != tail) {
    if (p->data == data) {
      return p;
    }
    p = p->next;
  }
  return nullptr;
}
```

代码 4-5　查找方法的实现

查找方法的实现则需要遍历整个链表，直到找到目标节点或者遍历到尾节点。因此查找的时间复杂度为$O(n)$，在后续的实验中，需要注意尽量减少查找的次数。

剩下的三个方法略去。

4.2　教职工类

教职工类的定义在`src/Employee.hpp`中。

教职工类定义了教职工的各种基本信息

5　实验调试和运行

5.1　实验调试说明

需要安装 `bazel` 构建工具。

笔者使用的是 `Arch Linux`，可以直接使用 `yay -S bazel` 安装。

6　实验总结