浙江工业大学

C++程序设计课程设计
课设报告

2023/2024(2)

实验题目 校友录管理系统

学生姓名 郭谨志

学生学号 302024315377

学生班级 软件工程 03

任课教师 廖锋峰

提交日期 2025/6/8

计算机科学与技术学院

校友录管理系统 实验报告

一、 大型实验的内容

校友录管理系统（AMS：Alumni Management System）用于校友录的管理，主要面向人群包括访问者（在校学生、经过认证的社会人员）、毕业校友、管理人员。可以完成增添、删除、修改、精确或模糊查找校友信息、对校友信息进行排序和分类统计、设置信息隐私权限，管理认证账号等工作。要求使用学习过的 C/C++程序设计的知识完成校友录管理系统的设计与实现。

二、 运行环境

校友录管理系统（AMS）在 Visual Studio 2022 平台下开发，操作系统：Windows 11。

硬件环境：

处理器：Intel(R) Core(TM) Ultra 5 125H 1.20 GHz

内存：16.0 GB

系统类型：64 位操作系统，基于 x64 的处理器

三、 实验课题分析（主要的模块功能、流程图）

3.1 校友录管理系统的主要功能

校友录管理系统（AMS）主要功能为：增添、删除、修改、精确或模糊查找校友信息、对校友信息进行排序和分类统计、设置信息隐私权限，管理认证账号等工作。详细的系统功能结构为图 1 所示。

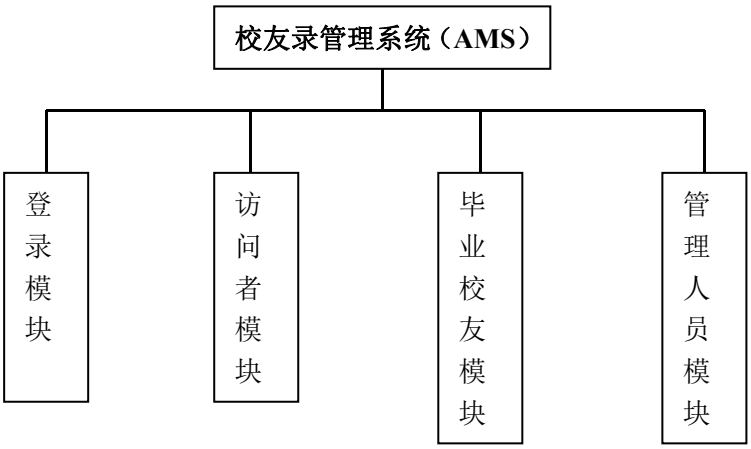


图 1 系统结构图

系统各模块的功能具体描述为：

1、登录模块

选择登录身份（在校学生、毕业校友、经过认证的社会人员、管理人员），输入用户名（学号或认证账号）和密码，成功后进行相应的功能模块。第一次登录要从文件读入所有的信息，即校友信息，个人信息（访问者/校友/管理人员）。

校友信息：

姓名，性别，年龄，届级，系，专业，班级，（通讯地址，电话，qq，email【非必填项】）

个人信息：

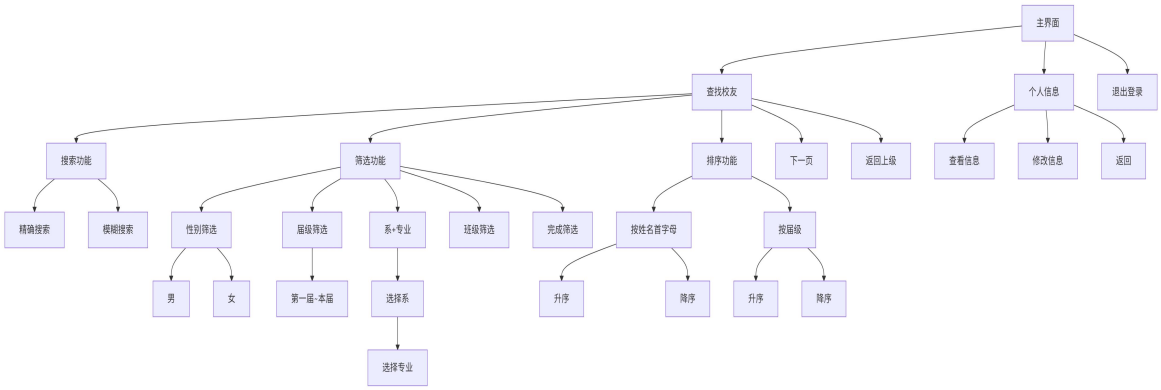
账号（学号/认证账号，密码，人员类别）

登录界面设计：

（1）选择登录身份

（2）输入账号密码，有返回键，输入错误弹出“用户名或密码不正确”（统一提示更安全），并重新输入。

退出登录的时候则要保存当前系统的所有状态，包括校友信息和当前所有有效人员的信息。



主操作模块（下面的模块在此模块基础上新增功能）界面设计：

第一级：查找校友/个人信息/退出登录

查找校友：

（1）搜索（精确+模糊）

（2）筛选

第一级：先选择（性别/届级/系+专业/班级/完成）

第二级：

（输入序号选择筛选条件，重复输入可切换选择状态，输入 0 结束输入）

性别：男/女

届级：第一届~本届（当前时间在毕业时间之前与否）

系+专业+班级：读取文件，专业在系的下一级

（3）排序（默认设置为姓名首字母升序）

第一级：按姓名首字母/按届级（筛选只选单个届级则不显示按届级，并回到默认排序方式）

第二级：升序/倒序

（4）返回上一级

个人信息：

展示个人信息

（1）修改密码

输入密码，再次输入密码

（2）返回上一级

2、访问者模块

在主操作模块上无新增功能。

3、校友模块

在主操作模块上新增新建，修改，删除自己的校友信息

输入序号选择要修改的信息，有保存键

4、管理员模块

在主操作模块上新增删除、修改校友信息，创建认证账号

可在查找校友信息时额外输入序号选择删除或修改校友信息

新建校友信息：

删除/修改校友信息：

查找校友信息时，可以输入序号选择修改信息

创建认证账号

进入新建账号模式，默认一直新建，输入账号和密码以及人员身份类型，有保存并退出键

3.2 系统分析及设计

系统涉及对象有以下类：工具函数类，筛选器类，链表模板类，校友信息类和人员类。其中人员类又可以细分为访问者类、校友类和管理员类。部分类涉及的功能操作归纳为如下表所示：

表 1 工具函数类涉及的操作

对象	成员函数
无	获取用户输入的选项：(1~n)的数字
	获取用户输入的选项：(0~n)的数字（用于动态选项的保存）
	修改密码（密码验证）
	只允许输入数字的输入验证
	拼音输入验证（自动转小写字母）

	在字符串中查找子串并返回权重值

（无对象，所有函数都是静态函数）

表 2 链表模板类涉及的操作

对象	成员函数
Node 结构体的声明 Node* head 迭代器 Iterator 类的定义	创建空链表（构造函数）
	清空链表（析构函数）
	判断链表是否为空
	插入元素
	查找元素（重载[]）
	删除元素
	打印链表
	清空链表
	拷贝构造函数
	判空
	查找元素是否存在
	赋值（重载=）
	Begin()返回链表头指针的 Iterator
	End()返回链表尾结点的 next 的 Iterator

表 3 筛选器类涉及的操作

对象	涉及的对象操作
五个由 vector 数据堆和 bool 是否筛选标识符组成的结构体，分别对于性别，届级，系，专业，班级。	构造函数
	拷贝构造函数
	针对单个对象筛选
	展示筛选条件
	修改筛选条件

以及一个含多重嵌套的系的 vector	重置筛选条件

表 3 校友类涉及的操作

3.3 系统的实现

（1）类的编写

系统工程名为：AMS。主要包含了 People 类（所有人员的基类），Department 类（往下派生出专业和班级，通过外部文件导入），List<T>（链表模板类），fileManager(文件管理），Utils(工具函数类）以及针对不同对象的操作模块类。其中 person 类成员为账号密码，而 alumni 类在继承 person 基础上增添一大批校友信息

（2）链表的使用

系统实现采用文件的输入输出流对文本数据进行读取与写入，以链表为操作载体，文件用于数据持久化，对数据的存储组织使用了单向链表。

因为校友录管理系统在登录、查找、修改、添加的时候都需要处理大量的数据，所以使用链表十分必要。

校友录管理系统的信息的管理就具体表现为链表的操作。校友信息的查找、修改、添加和删除与链表的查找、修改、添加、删除对应。

- **校友的查找：**

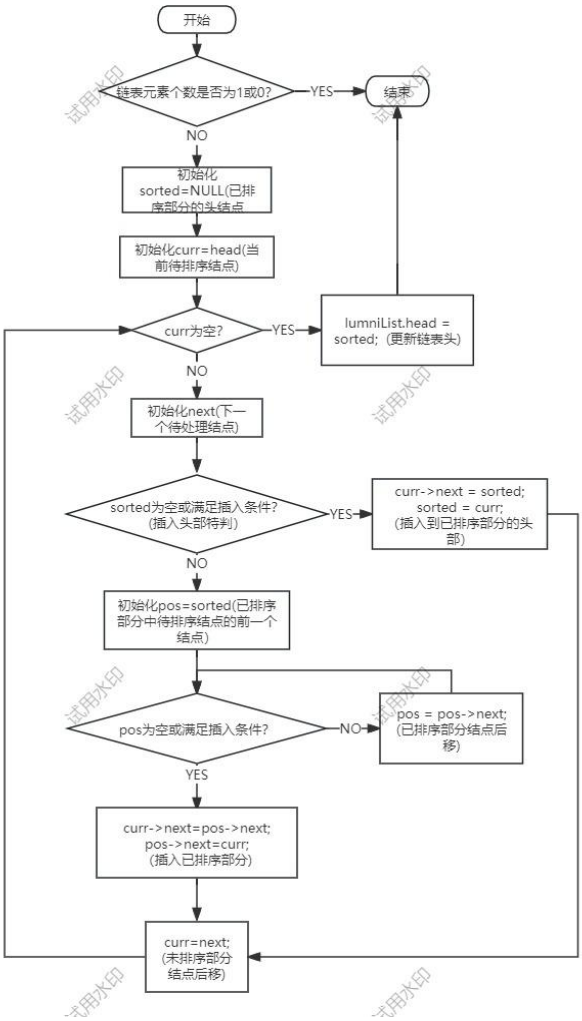
在用户在查询校友时，有以下方式：

搜索查找

筛选查找

排序查找

其中排序查找需要对链表排序，采用原链表内插入排序的方式，函数接受一个函数指针参数作为比较函数。函数流程图如下：



四、 实验调试、测试、运行记录及分析

遇到的问题及解决方法如下：

- **问题 1（最大问题）：**

无法解析的外部符号 "public: void __cdecl alumni::show(void)" (?show@alumni@@QEAAXXZ)，函数 "public: void __cdecl alumniModle::run(void)" (?run@alumniModle@@QEAAXXZ) 中引用了该符号	course design_AMS	alumniModle.obj	1
无法解析的外部符号 "public: void __cdecl alumni::show(void)" (?show@alumni@@QEAAXXZ)	course design_AMS	alumni_list.obj	1
无法解析的外部符号 "public: static bool __cdecl alumni::Compare_by_year_up(class alumni const &,class alumni const &)" (?Compare_by_year_up@alumni@@SA_NAEBV1@0@Z)，函数 "public: void __cdecl managerModle::run(void)" (?run@managerModle@@QEAAXXZ) 中引用了该符号	course design_AMS	managerModle.obj	1
无法解析的外部符号 "public: static bool __cdecl alumni::Compare_by_year_up(class alumni const &,class alumni const &)" (?Compare_by_year_up@alumni@@SA_NAEBV1@0@Z)	course design_AMS	VistorModle.obj	1
无法解析的外部符号 "public: static bool __cdecl alumni::Compare_by_year_down(class alumni const &,class alumni const &)" (?Compare_by_year_down@alumni@@SA_NAEBV1@0@Z)，函数 "public: void __cdecl managerModle::run(void)" (?run@managerModle@@QEAAXXZ) 中引用了该符号	course design_AMS	managerModle.obj	1
无法解析的外部符号 "public: static bool __cdecl alumni::Compare_by_year_down(class alumni const &,class alumni const &)" (?Compare_by_year_down@alumni@@SA_NAEBV1@0@Z)	course design_AMS	VistorModle.obj	1
无法解析的外部符号 "public: static bool __cdecl alumni::Compare_by_name_up(class alumni const &,class alumni const &)" (?Compare_by_name_up@alumni@@SA_NAEBV1@0@Z)，函数 "public: void __cdecl managerModle::run(void)" (?run@managerModle@@QEAAXXZ) 中引用了该符号	course design_AMS	managerModle.obj	1
无法解析的外部符号 "public: static bool __cdecl alumni::Compare_by_name_up(class alumni const &,class alumni const &)" (?Compare_by_name_up@alumni@@SA_NAEBV1@0@Z)	course design_AMS	VistorModle.obj	1
无法解析的外部符号 "public: static bool __cdecl alumni::Compare_by_name_down(class alumni const &,class alumni const &)" (?Compare_by_name_down@alumni@@SA_NAEBV1@0@Z)，函数 "public: void __cdecl managerModle::run(void)" (?run@managerModle@@QEAAXXZ) 中引用了该符号	course design_AMS	managerModle.obj	1
无法解析的外部符号 "public: static bool __cdecl alumni::Compare_by_name_down(class alumni const &,class alumni const &)" (?Compare_by_name_down@alumni@@SA_NAEBV1@0@Z)	course design_AMS	VistorModle.obj	1

语法错误: 标识符"List"	course design_AMS	alumni_list.h	13
"alumni_list::alumni_list(void)": 已经定义或声明成员函数	course design_AMS	alumni_list.h	13
语法错误: 缺少";"(在"<"的前面)	course design_AMS	alumni_list.h	23
缺少类型说明符 - 假定为 int。注意: C++ 不支持默认 int	course design_AMS	alumni_list.h	23
意外的标记位于";"之前	course design_AMS	alumni_list.h	23
"alumniList": 未声明的标识符	course design_AMS	alumni_list.h	14
"alumniList_": 未声明的标识符	course design_AMS	alumni_list.h	14

"List": 未定义基类	course design_AMS	alumni_list.h
"List": 未定义基类	course design_AMS	alumni_list.h

缺少类型说明符 - 假定为 int。注意: C++ 不支持默认 int	course design_AMS	alumni_list.h	22
缺少类型说明符 - 假定为 int。注意: C++ 不支持默认 int	course design_AMS	alumni_list.h	22

使用了未定义类型"alumni_list"	course design_AMS	alumni_list.h	66
使用了未定义类型"alumni_list"	course design_AMS	alumni_list.h	75
"it": 初始化之前无法使用	course design_AMS	alumni_list.h	76
使用了未定义类型"alumni_list"	course design_AMS	alumni_list.h	76
无法取消引用类型为"int"的操作数	course design_AMS	alumni_list.h	77
无法取消引用类型为"int"的操作数	course design_AMS	alumni_list.h	78
无法取消引用类型为"int"的操作数	course design_AMS	alumni_list.h	79
无法取消引用类型为"int"的操作数	course design_AMS	alumni_list.h	80
无法取消引用类型为"int"的操作数	course design_AMS	alumni_list.h	81
无法取消引用类型为"int"的操作数	course design_AMS	alumni_list.h	82
无法取消引用类型为"int"的操作数	course design_AMS	alumni_list.h	83
无法取消引用类型为"int"的操作数	course design_AMS	alumni_list.h	84
无法取消引用类型为"int"的操作数	course design_AMS	alumni_list.h	85
无法取消引用类型为"int"的操作数	course design_AMS	alumni_list.h	86
无法取消引用类型为"int"的操作数	course design_AMS	alumni_list.h	87
无法取消引用类型为"int"的操作数	course design_AMS	alumni_list.h	88
无法取消引用类型为"int"的操作数	course design_AMS	alumni_list.h	89
无法取消引用类型为"int"的操作数	course design_AMS	alumni_list.h	90
使用了未定义类型"alumni_list"	course design_AMS	alumni_list.h	95
使用了未定义类型"alumni_list"	course design_AMS	alumni_list.h	99

不允许使用不完整的类型 "alumniFilter"	course design_AMS	alumni_list.h	250
不允许使用不完整的类型 "alumniFilter"	course design_AMS	alumni_list.h	311

问题描述：最开始遇到这种问题，还是我只写了函数声明没有写定义，于是我以为把定义补上就好，没想到后面补充完整后依旧大量报错。编译器不会给这些错误标红色下划线，并且编译器自带的检索可以检索到这些函数的存在。这意味着这些函数声明正常，但找不到定义

解决方法：经过询问 ai，上图这些报错，大致有两种可能，第一种是函数有声明没有写定义，第二种是找不到完整函数定义。其中模板类的使用和头文件互相包含是 ai 提到最多的关键词。首先了解到模板类一定要在类内定义，因为编译器需要在每次实例化时看到完整的模板定义。若将实现放在源文件（.cpp），其他文件包含该头文件时无法访问模板的具体实现，导致链接错误，但改后问题依然存在。于是我和其他同学尝试后发现将类的实现放到头文件中，或者将两个头文件合并，或者调整头文件引用顺序可以减少部分报错。经过多次把类的实现放到头文件中和合并两个头文件的操作，问题减少了许多，但依然存在。我发现即使在同一个文件中，即使加上前置声明，后面使用的类型，也必需在前方有完整的实现，这可能是这些类与一个模版类相关导致的。后面我开始着重寻找文件之间的依赖关系，发现存在一个相互依赖的三角关系（filemanager 需 alumnlist,listT

冲突 filemanager 需 alumnilist 需 alumnifilter 需 filemanager

），使得即使在一个文件中，无论如何调整顺序也总有看不见定义的情况存在 通过将 alumnifilter 类内函数使用 filemanager 导入 department 向量，改为类外创造 department 向量，直接将 department 向量作为 alumnifilter 成员，现在，这个文件用到的每个类型都在上面有完整的实现，问题就解决了。

图 6 调试测试问题 1

- **问题 2：**
- **问题描述：**进行搜索查找输入关键词，软工 01，发现按照权值排序有个软工 02 班的排在软工 01 的前面。后来发现软工 02 班的毕业年份是 2021 年而 01 的是 2023 年这个 2021 的 1 使权值相等
- **解决方法：**加入额外权值判断，首先为不同的特征设置不同的权重，同时如果关键字完全等于输入的搜索句权值大大增加。如果关键词连续权值也翻倍；

五、 实验总结（优点、不足、收获及体会）

优点：

1.分不同类型人员操作，不同人员有不同的权限，可以在登录时选择登录身份

2.筛选功能强大，可以用多类型的条件筛选，每个类型支持多选条件。采用选择选项的方式，避免用户输入不规范的条件。另外能根据外部导入的系专业班级文件动态调整系专业班级选项，根据当前时间动态调整毕业年份选项

3.搜索功能强大，直接采用搜索栏搜索的方式，无需用户逐一对每个类型输入搜索词。搜索结果加权排序

4.管理员权限宽，不仅可以修改，增添，删除校友信息，还能创建认证账号，用于给外部人员访问

5.部分校友信息可以调整隐私权限，选择公开或私密

6.有输入合法性检测，通过工具函数类成员函数实现分类又统一的输入检测

不足：

1.未实现预想的分页展示功能，即展示校友信息时每次展示 10 条，用户可以翻页

2.界面设计不够精美，风格不统一，没有颜色区分

3.部分地方使用额外容器来实现如去重之类的功能，内存占用大

4.如果能重来，尽量不再使用 switch,改用 ifelse，因为 switch 中定义对象会报错，只能在外部定义，大大降低可读性

5.未能良好解决“无法解析的外部函数”“未定义类型”等问题，发现将类的实现放在头文件和将不同头文件合并能减少报错，最终发现可能的问题在于三个类的三角关系包含，但迫于时间压力和改动过大没能按照正常的分文件形式完美编译通过

6.没有设置的规范报错，部分地方缺少特殊判断和异常处理

7.没有将排序整合到筛选与搜索中

收获：

1.加头文件守卫防止重复包含：#pragma once。Cpp 文件不用因为不会被重复引用

2.注意头文件调用顺序，遵循后调用依赖先调用的原则

3.（*int)(int,int)才是指向函数的指针，*int(int,int)是返回指针的函数

4.静态函数类内声明加 static，类外实现不需要加 static

5.若已在定义中初始化成员变量，不需要在构造函数中再次初始化，否则报错已有主体定义

6.vector 类型成员不用在析构函数中释放，由系统自己释放

7.注意对象一定不要和类型同名了，否则会在某些情况下无法识别类型

8.vs 创建 txt 文件要将属性里内容一项改为是

9.工具类成员函数都设计为静态函数，不需要对象即可调用

10.学习了 git 提交到 github 的方法，学会了创建分支合并分支，运用了一次克隆储存库回档

11.类优先写构造，默认构造，拷贝构造，重载赋值运算符，不然极易出错，也写上获取成员的函数以备不时之需

12.注意何时使用 cin.ignore(numeric_limits<streamsize>::max(), '\n') ,作用时清空输入缓冲区中当前所有的剩余字符，直到遇到换行符或达到最大忽略数量。一般用于 cin>>后，不要直接位于 getline 后，因为 getline 先吃了一个换行符导致需要多输入一个换行符，建议放在 if(!getline(...))后

13.const string& a=...相当于宏字符替换

14.学会了用 csv 格式写文本文件：CSV 格式,通用性强，可以用 Excel 打开(分隔符推荐，或\t）

15.for 的冒号用法

体会：

也许分开编译测试会更好，所有模块加在一起最后整合时极其痛苦，问题会互相牵连。可以先从基类开始逐渐测试扩展。有时代码的更改并未应用，需要清除解决方案。保持每完成一个功能就上传代码到 github 的习惯，当要进行一项比较大的更改时，可以把原代码注释掉，这样出现问题方便回档。

六、 附录：源代码