程序设计基础二

C++ 大作业

2022 年 7 月 15 日

尤韦捷

目录

1 前言

这份报告是我在开学前完成的，这时候我已经上完了 python 小学期，有了更多的感悟和收获。事实上，如果时间允许，我很希望我能重写一边这个程序。卖油翁言：“无他，唯手熟尔”。这也是我所认为的让自己编程能力进步的最好办法。

由于程序代码过多，我在提交时将会把该报告和所有程序代码一并压缩打包。同时，我的所有源程序也上传到了我自己的github 仓库。

运行该程序时，请直接点击解压缩出的 “End_Of_Term_Hw.exe” 文件，并且保证该目录下包含‘ClassInfo.txt’和‘UserInfo.txt’两个文件。

2 系统需求分析

对于任何一个学校的课程系统，不同课程的成绩管理都是至关重要的部分。对学校的教务部门和老师来说，应该能为每个课程的每个学生进行成绩的录入和修改等管理。对学生而言，则应该能够了解自己每一门课程的具体成绩。这是一个成绩管理系统最基本的功能，在此功能的基础上，还可以有诸如计算班级平均成绩，计算某一个学生的均绩等功能。

在数据的底层存储上，成绩管理系统事实上是一个 “双变量” 的映射。也就是，如果课程为 A，学生为 a，那么，成绩管理系统最基础的功能就是提供一个 f，以获取学生 a 在课程 A 中的成绩 $grade$，即

$$grade = f(A, a)$$

剩下的功能，就是在该映射的基础上对获取的 $grade$ 进行操作，进而实现更加 fancy 的应用。

但是，在一开始设计系统时，我并没有很好地意识到这一点，从而建构了一些复杂且容易出错的数据存储结构。具体的思路会在下方进行阐述。

该系统主要面向教师和学生两类人群。由于用户身份不同，在系统中对不同类型的用户设定了不同类型的权限和功能。

对于教师，该系统提供以下功能

1. 查看自己所授课程的成绩情况，包括平均成绩、不及格人数等等
2. 上传、登记和修改自己所授课程的学生成绩
3. 登记一门自己教授的课程
4. 上传和注册一门新的，自己教授的课程

对于学生，该系统提供以下功能

1. 展示某一个特定课程或所有课程的成绩
2. 计算平均绩点
3. 选择加入某一门已有课程
4. 退课

同时，该系统提供简单的登录功能，并提供文字菜单以供用户进行操作。

3 总体设计

我所设置的成绩管理系统需要能完成几个基本功能——

a) 实现人员的登录和自动权限识别

b) 实现课程和人员信息的自动存储和读取

c) 实现对课程和人员信息的匹配

同时，在与用户的交互方式上，我采用文字菜单和文字提示用户输入的方式。系统将自动在窗口中打印出选项带有序号的菜单，用户通过输入选项对应序号进行选择。同时，在用户登录、登记成绩等需要输入其他信息的情景中，系统将通过文字提示（prompt），指引用户按顺序输入所需内容。每次输入通过 enter 键进行确认。

在数据的存储方面，该系统将人员数据和课程数据分别存储在两个文件中，每次启动程序时将读取这两个文件中的数据，对数据进行修改后则将改动存入文件中。为了方便测试和修改数据，这两个文件使用 txt 文本格式。如为了保护数据不受篡改，之后可改为二进制文件形式等。

该系统的功能模块和运行逻辑如下图：

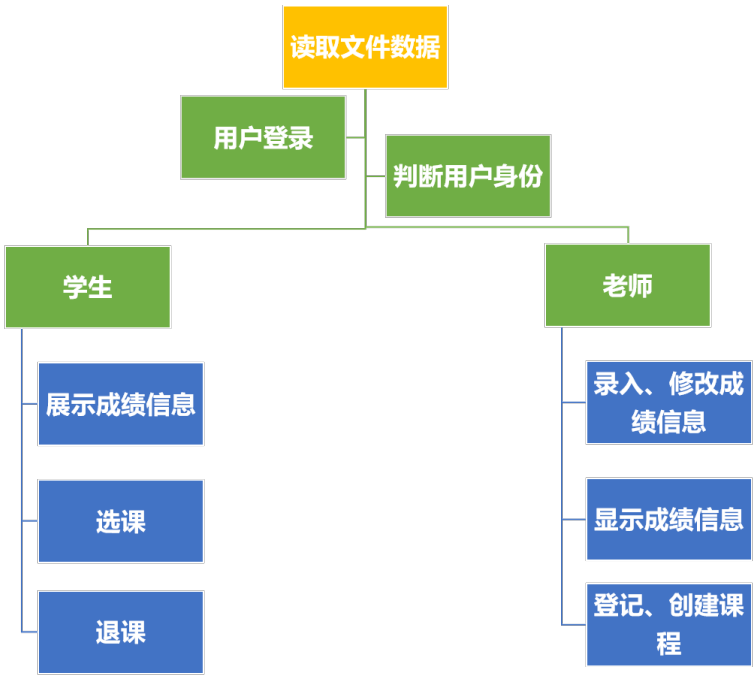

图 1 系统运行流程示意图

4 详细设计

为了实现该系统的功能，我设计了两种类别，一种是数据类型的类，包括用户类（User），课程类（Class）。在用户类（User）的基础上，又派生出两种子类——学生类（Student）和教师类（Teacher）。这些类的主要功能是将姓名、ID、密码等属于同一个对象的不同数据进行封装和打包，并且提供操纵这些数据的接口。

除此之外,我还设计了两个方法类,分别为人员操控类(Interface),课程操控类(ClassInterface)。这些类提供了系统所需的其他功能，如读取存储的数据并构建用户对象动态数组，提供对数组的操作接口和函数等（如修改成绩的函数)。

该系统的基本数据结构如图3所示，其数据的骨架是两个数组，用户类对象数组和课程类对象数组。其中用户类数组（本质是用户类对象指针数组，即 User** users) 数组的初始化在构造对应的方法类时一并通过读取文件中的信息实现，从而达到将硬盘中的数据读取到内存中的目的。也就是说，这两个类提供了对系统中所存储的所有人员信息和课程信息进行操作的函数和方法。其中 Ingerface 类的定义代码如下。

```
class Interface          //Interface of the Users
{
public:
        Interface();
        ~Interface();

        //to initiate the interface using user's info file
        void init();

        //INPUT: usr, can be any User* ptr, will be modified to
        //fit the login user
        //OUTPUT: return the arthority of the user if login successes.
        int login(User* &usr);

        //save all user info into one file
        void save();

        //add one user to the system
        void addUser();

        //OUTPUT: return the subindex of User who had the same id
        //in the User array,
        //      or return -1 if no match exists.
        int locOfId(int id);

        //make a list of all the classes the User attended to
        //or lectured in and transfer it into a array of Class ptrs
        //and the total num of this array
        void list_all_classes(Class**& clsptr, int& clsNum);

        User** getUsers();
        int getUserNum();
protected:
```

```
        //----------------data----------------
        //all users in the system
        User** users;
        //the logined user
        User* lgUsr;
        //total number of the users
        int uNum;
};
```

关于课程和授课教师以及参与课程学习的学生之间的对应关系，我采用了一种愚笨臃肿，但暂时还没有出错的对应方式。具体来说，我创建了一些链表类，用于存储课程的 id，在每一个用户类的对象中，包含有一串其所教授/参与的课程 ID 链表，以供搜索和修改。在每一个课程类的对象中，也打包有一串其参课学生对应的对象的指针和其成绩的链表。在进行学生成绩的修改时，可以直接通过遍历该链表，通过指针所指向的位置信息进行对应学生的定位和成绩的修改。在进行“选课”和“退课”操作时，则需要修改两个对象中的链表，以避免出错。

课程类对象所含链表的定义如下：

```
class ListOfStdts {
public:
        User* stdt;

        //pointing to the next student;
        ListOfStdts* nxstdt;

        //the grade of this course;
        double grade;

        ListOfStdts(User* std,
                    double g = -1,
                    ListOfStdts* next = NULL) {
                stdt = std;
                grade = g;
                nxstdt = next;
        }
};
```

在数据的读取方面，我设定了固定的数据输出和读取格式，通过‘\n’和‘ ’等方式完成数据的分隔和切换。人员数据和课程数据在不同的文件中分开存储，每一行代表一名用户或者一门课程的数据。具体示例如图2，其中前四项分别为 ID、权限、姓名和密码，最后的数字为所参与或教授的课程 ID（如第一行，ywj 同学所参加的课程为课程 id 为 3、2、1 的三门课程）。

具体的类图如图3所示

其中，Menu 类不包含数据，仅为打包的显示菜单函数。Menu 本身为虚基类，仅提供标准接口。由这一个类派生出 StudentMenu 和 TeacherMenu 两个类分别重载实现不同用户的菜单显示。

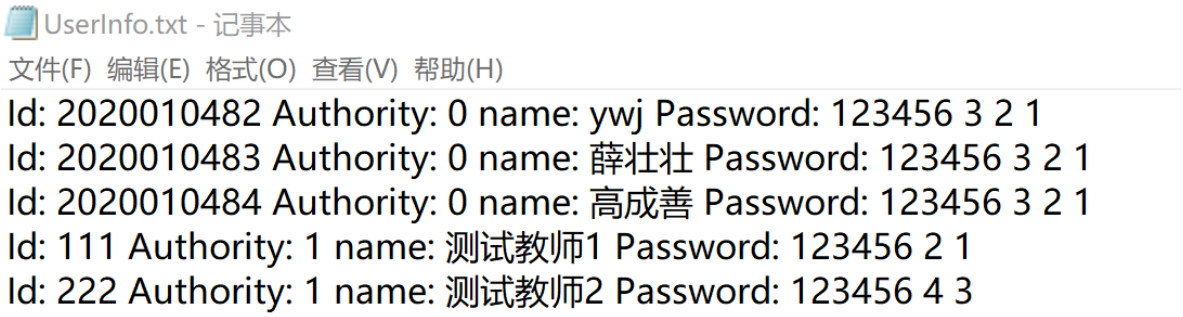

图 2 存储结果示例

同时，我在 main.cpp 文件中重新写了一个标准接口的菜单显示函数 showMenu()，以通过虚基类指针和函数重载实现标准化的，不需要根据用户类型作太多调整的菜单显示。

当然，过度打包和追求标准化也出现了“双刃剑”的效果。这使得许多函数和结构仅为了标准化和统一接口而创建，且为了实现统一接口，还必须附上复杂的判断逻辑。而这些标准化的效果不过是使得 main() 函数显得非常简洁罢了。事后来看有些得不偿失。

5 运行示意

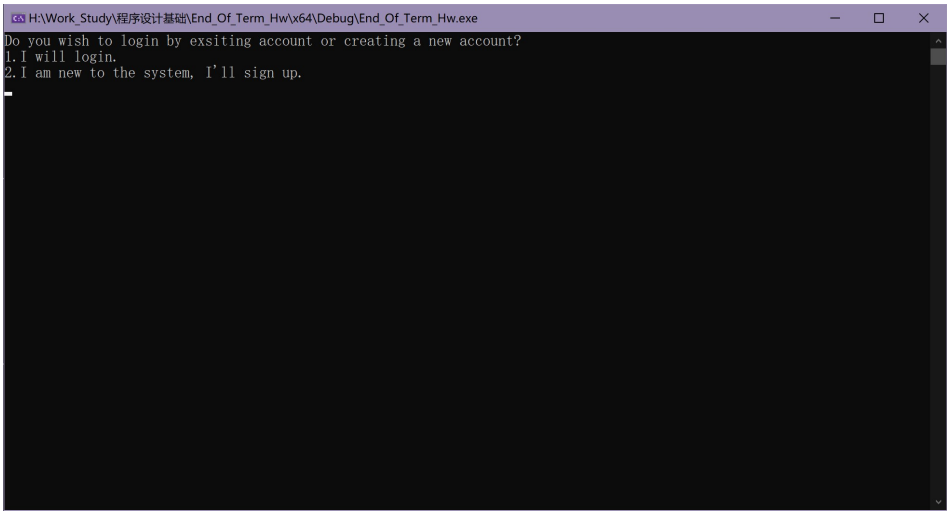

图 4 系统初始面板

该程序可直接通过点击‘End_Of_Term_Hw.exe’图标进行运行，但需要确保目录下同时包含‘ClassInfo.txt’和‘UserInfo.txt’两个文件。

进入系统后，将会要求用户进行判断——如果已经使用过该系统，则进行登录，如果没有，则进行注册（即添加用户）操作。

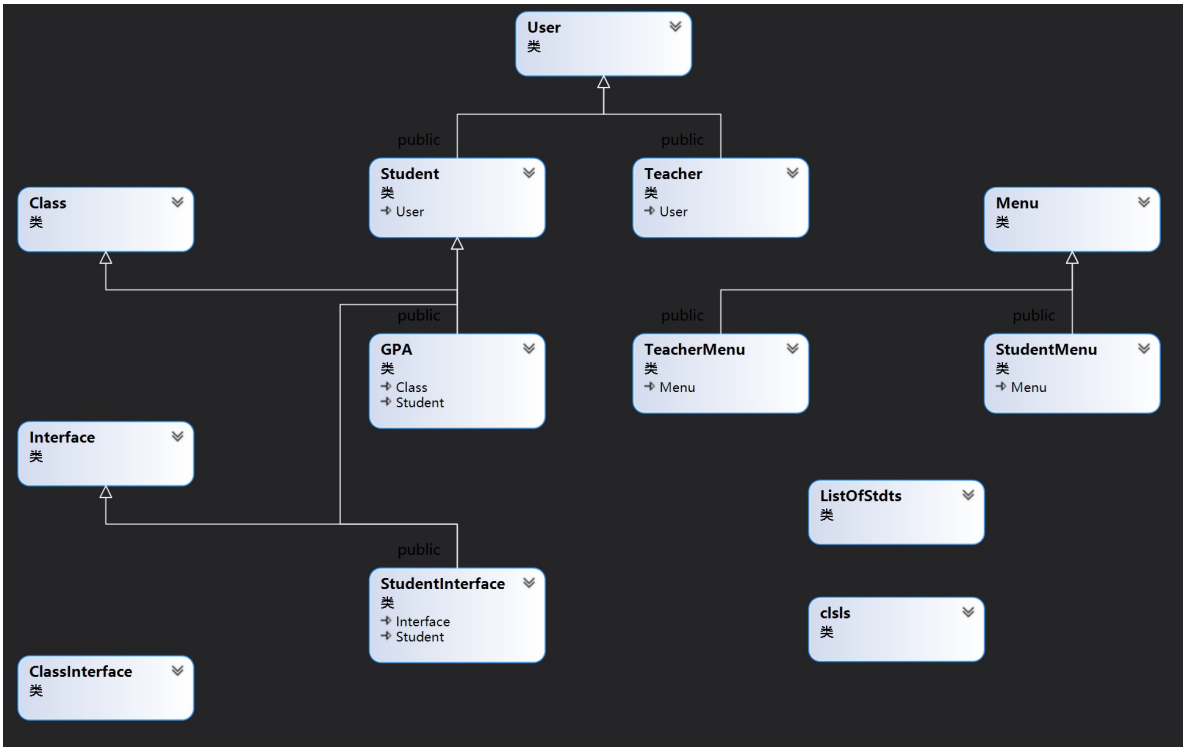

图 3 程序类图

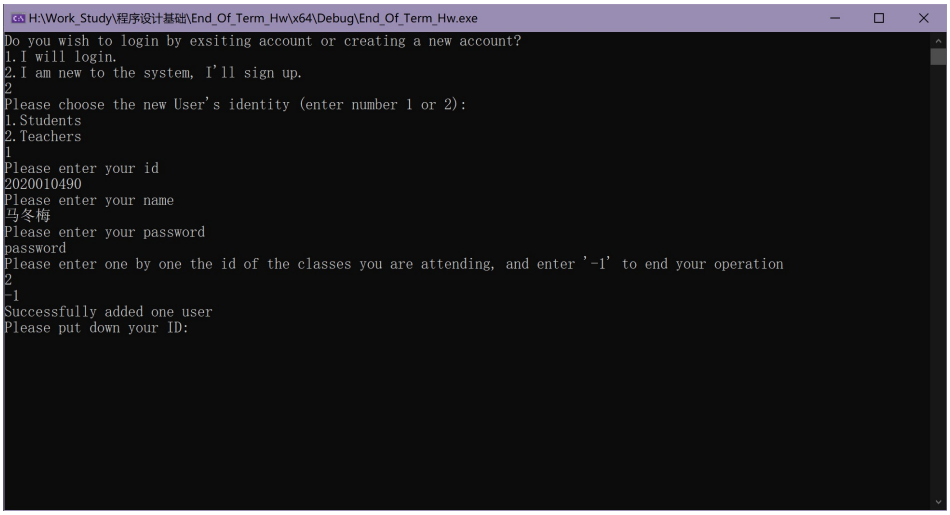

图 5 初始化用户示例

添加用户的操作如图5所示，用户将在提示下输入自己的学号、姓名、所选课程等信息，程序将在进行比对后将用户录入系统中。之后将直接进行登录操作。

如图6，当成功进行登录操作后，系统将欢迎用户，并且弹出开始菜单。之后，系统将给出初始的选择菜单，以提示用户选择自己想使用的功能。

如图7所示的为学生身份登录后所展示的第一级菜单。

通过输入对应选项前的序号即可进行功能选择。示例中，我们进行了学生用户的成绩查询和均绩计算演示，具体演示结果如图8所示

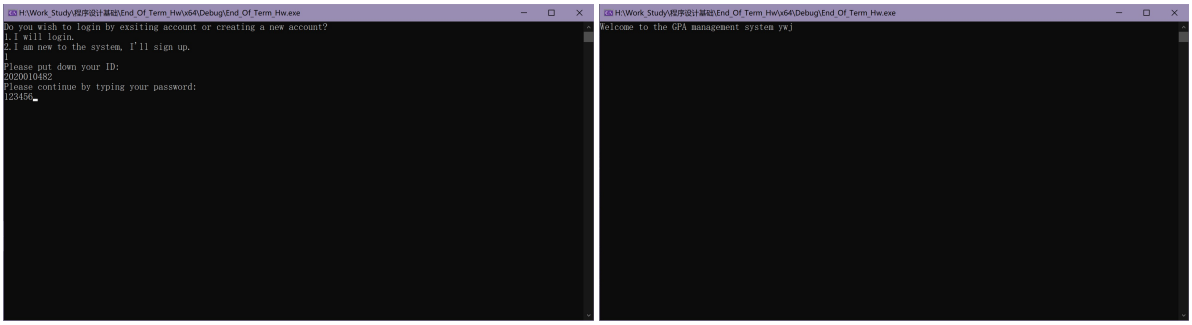

图 6 系统登录示意

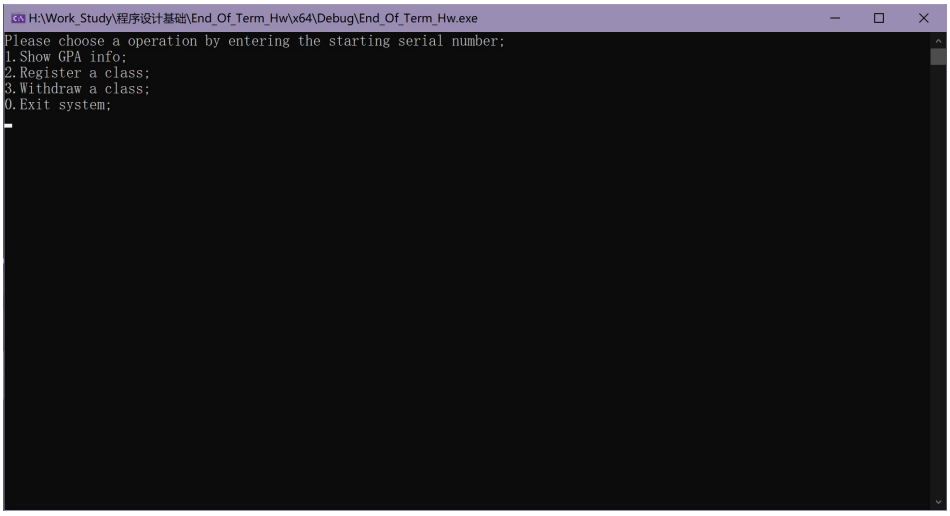

图 7 学生用户第一级菜单

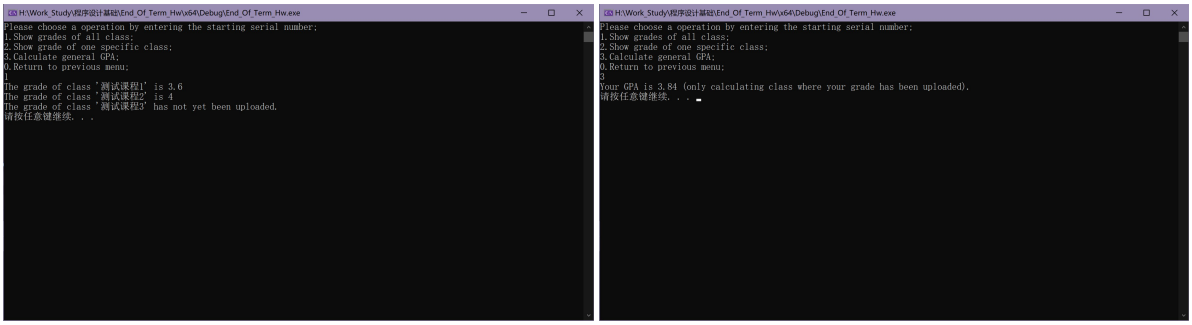

图 8 学生查看成绩及均绩示例

选课和退课功能的演示如下。选课时，系统将检测课程是否存在，以及该学生是否已经在课程参课学生中。退课时，除了检测以上两点，系统还将额外检测该课程是否已登记成绩，登记成绩后的课程无法退课。

关于程序的设计时间和编写进度，可参见我的 github 仓库中的README 文档。

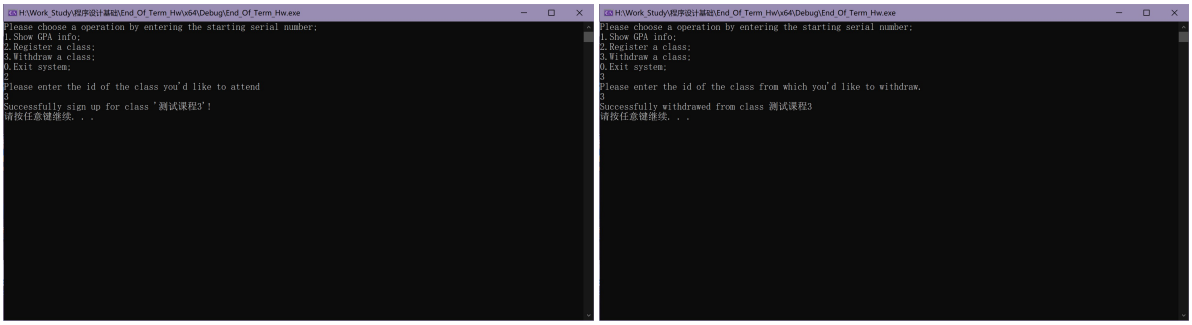

图 9 学生选课及退课示例

6 系统调试

我的代码编写主要在 7 月份完成，花费时间将近一周。之后，由于其他小学期事务，我仅在写完代码后做了最简单的测试以验证程序的可行性。更大规模的测试在八月份陆续进行。然而，在进行 python 小学期时，我发现我犯了一个错误——我没有在完成每一个方法或函数之后就进行测试，而是在写完整个程序之后在一起进行测试。这使得我的程序在一开始有许多错误而完全无法编译。这些错误包括但不限于：1. 最简单的语法错误，比如忘记了 “;”；2. 内存读取错误，因为我大量使用了指针和指向指针的指针，以及动态数组，使得我常常在不经意间将指针指向未分配的空间，发生越界；3. 逻辑错误，尤其是在类和函数的数量大大增加的情况下，由于并不规范的注释和自身的疏忽，我常常出现各式各样的逻辑错误，印象最深的就是我在计算均绩时，忘记将加起来的绩点乘以对应的学分，甚至我用于存储绩点的变量类型还是整型而非双精度浮点数。

因此，经过 7 月份一整天痛苦的 debug 过程之后，我才勉强将程序跑动起来。但实际上，这只是能跑，还是有许多地方存在我没有发现的错误。幸而，这些错误大多只要改动一两个函数，三四行代码，而不需要对我的程序结构作大的改动。7 月份写完代码之后，python 小学期刚好开始。小学期关于测试代码的部分让我触动很大。我相信之后我再也不会干把所有代码都写好之后再进行测试的蠢事。

同样的，在 8 月份重新回到这个任务继续进行 bug 的修复时，我注意到，虽然只过去了一个月，这个程序也并不是很复杂，但是大量同名或者命名相似的函数和不同类型的继承已经让我有些晕头转向。这个时候我在 debug 的同时开始重写程序注释，希望能让程序注释规范化。注释的格式方面，我参考了 mit 课程 6.0001 中的 python 注释格式。

同样的，在学习 python 过程中，我发现 “字典”（Dictionary）这一变量类型十分有助于我的程序数据结构的构建。这个时候，我才忽然意识到，C++ 的标准库中很可能内置了类似的 “字典” 容器。在谷歌的帮助下，我很快就发现，C++ 中的 “map” 容器事实上完成了和 python 中 Dictionary 类型 object 一样的功能。因此，被复杂的程序所涌现出的多种多样的 bug 和复杂的数据存储结构搞得心力憔悴的我很快开始思考能否使用 map 容器简化程序的数据存储和读取，同时利用 map 容器已有的方法和函数，从而简化程序结构。因此，我构造了一个新的程序分支 “simpified”。但是由于所参加的其他课程小学期、暑期实践和一些科创比赛的多重压力，现在这个分支还仍处于仅有一个构想和几个基本类的进度。

7 总结

我非常清楚地明白，我自己的代码能力并不算很强。究其原因，是我使用代码和练习代码的机会仍然偏少。因此，在编写这一个大作业时，我希望能够在这个时间还算充裕的暑假为自己提供一些练习代码的机会。于是，我特地，甚至说是有意地使用了指针动态数组，自己编写的链表类和多种重载函数。某种程度上，我希冀能通过这个大作业让我熟悉一下这学期所讲过的几乎所有 C++ 操作和功能。

当然，也是由于我的代码水平和熟练度不够，在进行这个程序的编写时，我遇到了很多困难，既有思路上的，也有程序错误上的。在这里我要感激互联网时代和 CSDN、StackOverflow 等网站为我提供的帮助。

经过这次大作业的编写，我认为我最主要的收获有二：其一是对 C++ 的多个特性和功能有了更深的了解，加深了自己的熟练度。与此同时，也减弱了我对指针、数组等较为“复杂”和“困难”的知识的畏惧之心。我发现我也能够写出一个还算是能用的指针动态数组（虽然常常出现指针越界等错误）。其二是我充分地认识到，一个良好的程序编写和测试习惯的重要性。如果我能遵守注释规范，虽然在一开始会多花一些时间，但是在后续重新回顾程序和进行 debug 时就会省去大量的麻烦。如果我在写好每一个函数和方法之后就进行单独的测试，也不会光让程序跑动起来就花了我一整天的时间，更不会在程序中埋下许多我至今可能都未发现的隐患和错误。

我个人接触较多的编程语言有 3 种，C++，python 和 matlab 的 m 语言。C++ 像是老牌而稳重的助手，你可能觉得它有些笨，需要更复杂的语法规范，且在对张量和第三方库的支持上并不是非常友好。但你写下的每一行代码都让你安心——因为它清楚、明确，没有歧义。这是我一开始接触 python 时所令我难以理解的——一个函数的形参怎么可以没有数据类型呢？虽然我也使用过 m 语言这类相似的，不需要显式说明变量类型的解释型语言，但毕竟 m 语言更多地用于科学计算，而 python 是能编写更加多样和复杂的程序的。我已经被稳重清楚的 C++ 养成了习惯，面对 python 总是担心传入非法参数发生错误。可以说，C++ 和这门课为我奠定了一些编程的固有概念和习惯，这是我认为我在上完这门课后最大的变化。