面向对象大作业报告

题目:　基于类模版的仓库管理系统

计科 2202 翟浩天 2022030148

项目简介

在本次面向对象编程课程的大作业中，我设计并实现了一个复杂且功能全面的仓库管理系统（Warehouse Management System）。该系统采用了 C++11 标准，结合了类模板技术，做到了高效的代码复用，并且具备较高的扩展性，并集成了简洁高效的图形用户界面（GUI），旨在为用户提供一个强大且易用的仓库管理解决方案。Github 链接：

https://github.com/zhaihaotian/warehouse_management_system

项目背景

仓库管理系统在现代企业运营中扮演着至关重要的角色。传统的仓库管理系统往往面临着界面复杂、操作繁琐、用户体验不佳等问题。为了提升用户体验和系统效率，我决定设计并实现仓库管理系统，通过面向对象编程技术和现代 C++标准来完成。

系统功能

用户管理

用户创建：系统允许管理员创建新用户，输入用户 ID、名称和电子邮件地址。

用户登录：已有用户可以通过输入用户 ID 进行登录，系统会自动加载用户的订阅信息和历史通知。

用户列表：管理员可以查看所有用户的列表，包括他们的 ID、名称和电子邮件地址。

商品管理

添加商品：仓库管理员可以添加新商品，输入商品名称、库存和价格。

修改价格：仓库管理员可以修改现有商品的价格，通过输入商品名称、活动名

称和新价格。

调整库存：仓库管理员可以调整现有商品的库存，通过输入商品名称和新库存数量。

事件管理与通知系统

订阅通知：用户可以订阅特定商品的价格或库存变动通知，系统会在相应事件发生时发送通知。

查看通知：用户可以查看历史通知，了解商品的最新动态。

类模板设计与实现

类模板设计思路:

为了提高系统的通用性和灵活性，我设计了一个类模板 Item，用于表示仓库中的商品。该类模板支持处理不同类型的数据，例如整数类型的库存数量、浮点数类型的价格等。类模版中的库存可以是一个任意类型的，整形与浮点只是在我们的 demo 可视化中出现，但实际上，倘若与实际情形中的货架结合起来（我们可以把每个货架的库存多少作为一个矩阵），才是真正的做到支持任意输入的类模版，同时与实际相结合。

此外，我还设计了一个事件类模板 Event 和一个事件管理器类模板 EventManager，用于处理商品的各种事件（如创建新的商品、库存更新、价格更新）及其通知功能。总而言之，事件类模版使得我们可以支持不同事件的传递，在本程序中有三种，然而我们可以随时定义新的 struct 来表示我们想传递的消息，然后重载输出，最终就可以做到类模版的扩展。

类模板实现过程

商品类模板 Item

```
template <typename T>
class Item {
private:
    std::string name;
    T stock;
    double price;

public:
    Item(std::string name, T stock, double price) : name(std::move(name)), stock(stock),
price(price) {}

    void updateStock(T amount) {
        stock += amount;
    }

    void updatePrice(double newPrice) {
        price = newPrice;
    }

    std::string getName() const { return name; }
    T getStock() const { return stock; }
    double getPrice() const { return price; }
```

```
    void display() const {
        std::cout << "Item: " << name << ", Stock: " << stock << ", Price: $" << price <<
std::endl;
    }
};
```

事件类模版

```
// 事件类模板
template <typename T>
class Event {
public:
    enum Type { ITEM_CREATED, STOCK_UPDATED, PRICE_UPDATED };

private:
    Type type;
    T relatedData;
    std::string activityName;
    time_t triggerTime;

public:
    Event(Type type, T relatedData, std::string activityName = "")
```

```
        : type(type), relatedData(std::move(relatedData)),
activityName(std::move(activityName)), triggerTime(time(0)) {}

    Type getType() const { return type; }
    const T& getData() const { return relatedData; }
    const std::string& getActivityName() const { return activityName; }
    time_t getTime() const { return triggerTime; }

    void display() const {
        std::ostringstream oss;
        oss << "Event Type: " << type << ", Data: " << relatedData << ", Activity: " <<
activityName << ", Time: " << triggerTime;
        std::cout << oss.str() << std::endl;
    }
};
```

可视化程序设计与实现

可视化程序设计思路

为了使系统更具交互性和易用性，我决定使用 wxWidgets 库来开发一个可视化程序。由于可视化主要是为了使我们理解类模版的通用型，所以我们的可视化程序主要是可视化了我们所使用的两个类模版，也就是商品和事件，通过可视化程序可以创建类模板的实例，并调用其成员函数。程序界面简洁明了，用户

可以轻松地进行商品添加、库存调整、价格修改以及事件订阅等操作。

商品管理界面

商品管理界面旨在提供一个直观的方式，让用户可以添加新商品并查看已添加的商品信息。用户需要能够输入商品的名称、库存类型（如整数或浮点数）、库存数量和价格，并在界面中显示这些信息。

事件管理界面

事件管理界面的目的是让用户能够触发和查看商品相关的事件。用户可以选择事件类型（如商品创建、库存更新或价格更新），输入相关商品的信息，并在界面中显示触发的事件

实现过程

商品管理界面实现

界面布局：使用 wxWidgets 创建主窗口（MainFrame），并添加必要的控件，包括文本框、下拉框和按钮。

输入控件：添加用于输入商品名称、库存数量和价格的文本框。

选择控件：添加一个下拉框，用于选择库存的类型（整数或浮点数）。

列表控件：添加一个列表框，用于显示已添加的商品信息。

按钮事件处理：为“创建商品”按钮添加事件处理函数，读取用户输入的数据，创建相应的商品对象，并将商品信息显示在列表框中。

事件管理界面实现

界面布局：使用 wxWidgets 创建主窗口（MainFrame），并添加必要的控件，包括文本框、下拉框、列表框和按钮。

输入控件：添加用于输入事件相关信息的文本框，如商品名称、库存变动和价

格变动。

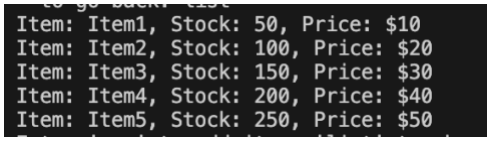
选择控件：添加一个下拉框，用于选择事件类型（商品创建、库存更新或价格更新）。

列表控件：添加一个列表框，用于显示已触发的事件信息。

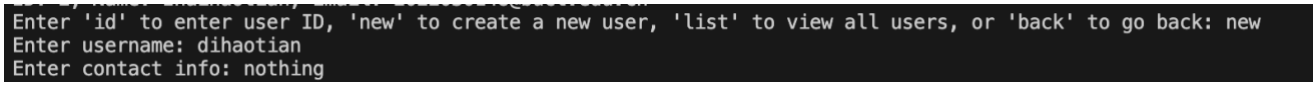
按钮事件处理：为“触发事件”按钮添加事件处理函数，读取用户输入的数据，创建相应的事件对象，并将事件信息显示在列表框中。

结合的实际问题

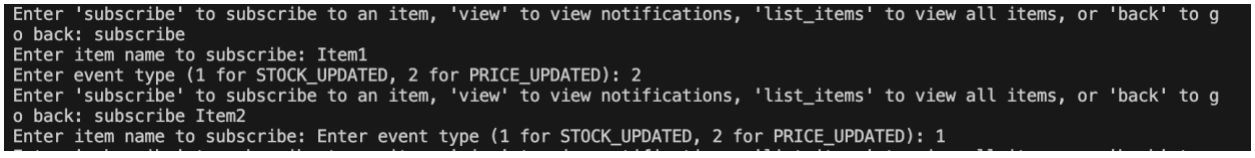
通过上述开发的类模版，我们再加入一个事件管理器的类模版，我们就开发出了一个库房管理系统，下面则带来一个这个库房管理系统的演示，源代码在代码库中的 main.cpp 里，该管理系统主要通过命令行来做交互。初始时有两个用户，五个商品。



创建一个新用户：

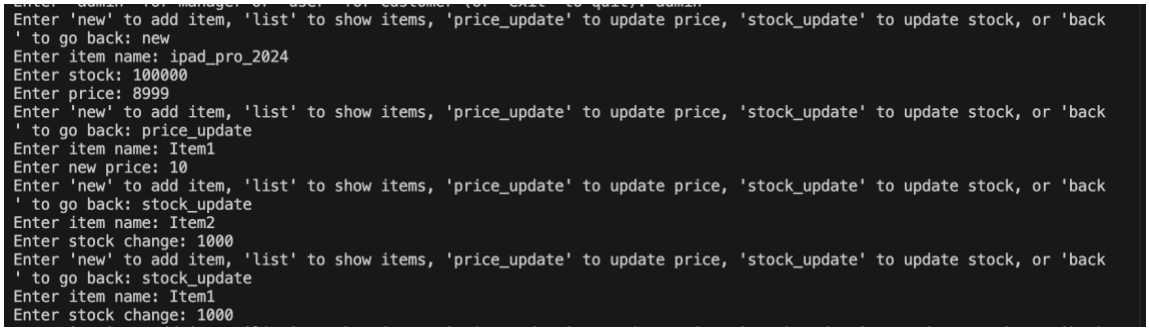


订阅两个商品：对商品 1 订阅价格 对商品 2 订阅库存变化



然后我们切换到用户 1，再订阅一个商品 1 的价格

现在我们切换回库房管理者，先上新一个物品，然后我们来改改价格与库存：



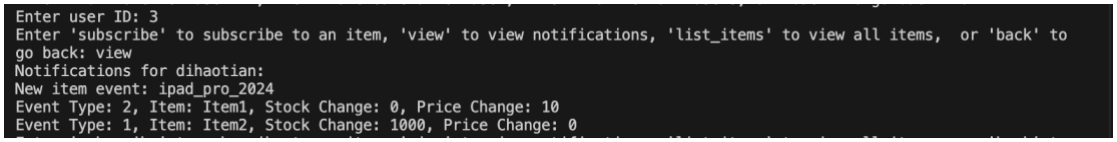
我们上新了新款的 ipadpro，同时对商品 1 的价格和库存做了修改，对商品 2 的库存做了修改

现在让我们切换回各个 user 看看他们会收到哪些信息：

首先我们从用户 1 开始看：

可以观察到，用户 1 由于只订阅了商品 1 的价格变化，我们只会推送价格变化，至于为什么推送新品？因为我设置了所有用户默认接受新品推送。

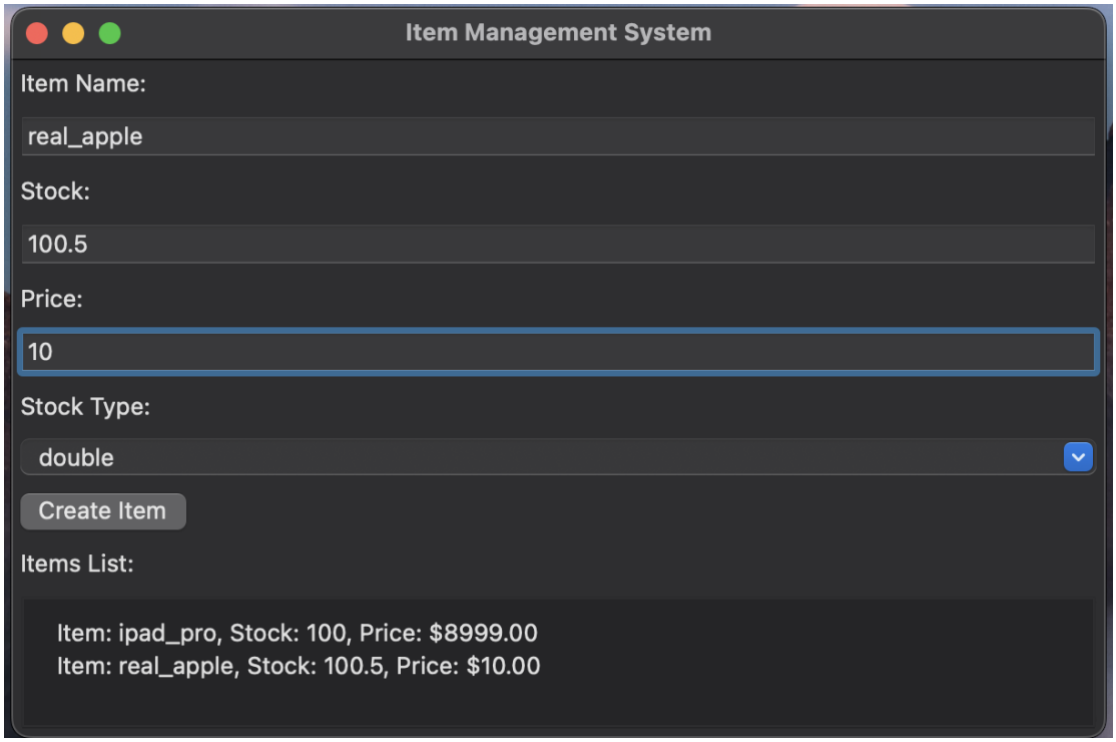
下面我们观察用户 3:



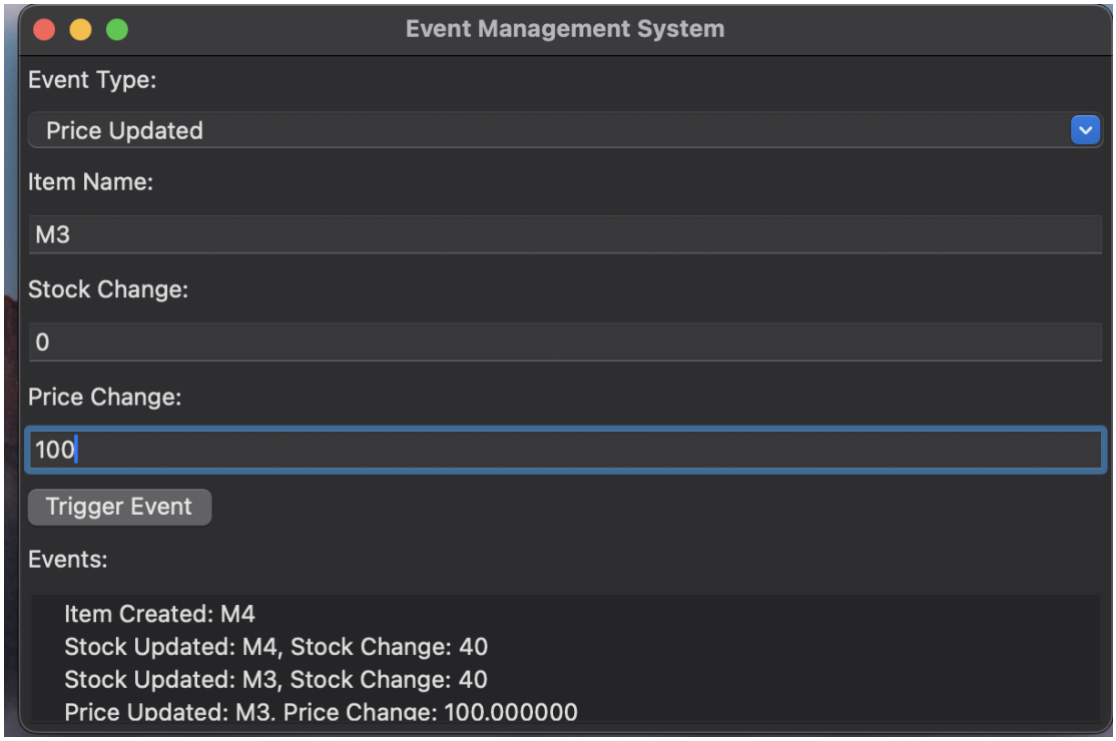
我们可以观察到，虽然商品 1 的价格和库存都发生了变化，但是由于用户 1 只订阅了商品 1 的价格变化，所以只会收到价格变化，然后毫不意外的收到了商品二的价格变化，同时这些消息也是按顺序接受到的。

如果我们再 view 一下，会发现看过的消息已经消失了。

可视化的结果运行结果展示：



考虑真实的苹果的库存为精准到小数点，所以设置成了 double，一般的商品的库存数量是 int。



定义了三种不同的事件类，以及他们的可视化。

遇到的问题：

1. 事件管理系统的设计与实现：我使用了一个嵌套的 unordered_map，第一级映射事件类型，第二级映射商品名称，从而快速定位到订阅该事件的用户列表。此外，通过对事件数据的封装和统一管理，简化了事件的处理流程。我觉得让事件去找用户通知的方式会比说商品更新了，然后用户把所有的商品扫一遍要高效的多，但其实用树的类模版然后建立一个图也不失为一种选择，但是用 unordered_map 即好写又高效。
2. 数据输入和验证：用户在输入商品信息和事件数据时，可能会输入无效或错误的数据，如不存在的商品。这会导致程序运行时发生异常或错误。为了避免这些问题，我在处理函数中添加了数据验证步骤。在添加商品时会检验该商品是否已经出现，在用户订阅的时候会查询该商品是否存在。

总结与反思：

通过本次大作业的设计与实现，我深刻理解了面向对象编程的核心思想和实践方法：抽象。面向对象就是不断的抽象与实例，也就是 object。继承，派生，多态是面向对象的核心思想。通过类模板的设计，我学会了如何编写具有通用性和可重用性的代码；通过事件管理系统的实现，我掌握了如何设计高效的事件处理机制；通过可视化程序的开发，我提升了界面设计和用户交互的能力。本次大作业不仅使我巩固了所学知识，更让我在实践中提升了编程能力和解决问题的能力，学会了有效地将一个较复杂的任务分解成许多易于控制和处理的子任务，便于开发和维护。