电子科技大学

计算机科学与工程学院

标 准 实 验 报 告

（实验）课程名称 C++语言程序设计

电子科技大学教务处制表

电 子 科 技 大 学

实 验 报 告

学生姓名： 杨敬	学 号：2023080903022

一、 实验室名称：计算机学院实验中心

二、实验项目名称： 面向对象与函数式编程综合实验

三、实验目的：

1. 掌握 C++面向对象编程的基本思想，包括继承、多态、封装及解耦循环依赖。
2. 学习 C++函数式编程的特点和实现方式，包括使用 lambda 表达式、std::function 等工具实现代码的简洁性。
3. 提升代码的可扩展性与可维护性，结合 STL 容器与表驱动模式优化复杂逻辑。
4. 实现功能需求，包括倒车（Reverse）指令和加速（Fast）指令，支持组合指令操作。

四、实验内容：

实验内容由两部分组成：

1. **面向对象编程：**
   - 使用继承与多态设计基本指令（如 MoveCommand、TurnLeftCommand）。
   - 通过表驱动方式将指令与处理逻辑解耦，减少循环复杂度。
   - 优化小车的状态管理，抽象出 PoseHandler 类，消除循环依赖。
2. **函数式编程：**
   - 删除 ICommand 接口，改用 lambda 表达式和 std::function 实现指令的功能操作。
   - 改写 ExecutorImpl::Execute 方法，使用表驱动管理指令。
   - 实现组合功能需求（倒车与加速模式），并支持复杂指令序列的处理。

五、实验器材（设备、元器件）：

开发环境：Visual Studio 2022

语言版本：C++20

单元测试框架：Google Test

六、实验步骤及操作：

1. 面向对象部分

1.实现基本指令类：

设计 MoveCommand、TurnLeftCommand、TurnRightCommand 等指令类，继承 ICommand 接口。

在指令类中实现具体操作（如移动、转向）。

2.优化状态管理：

抽象出 PoseHandler 类，管理小车的状态（位置、方向、加速与倒车模式）。

PoseHandler 提供状态查询接口，支持基本状态操作（Move、TurnLeft 等）。

3.引入表驱动：

在 ExecutorImpl::Execute 方法中，使用 std::unordered_map 将指令字符与指令类绑定，优化指令解析逻辑。

替代之前的多层 if-else 语句。

4.支持倒车指令：

增加 ReverseCommand 类，支持切换倒车模式。

修改 PoseHandler::Move 方法，加入倒车逻辑。

2. 函数式编程部分

1.删除 ICommand 接口：

移除所有指令类对 ICommand 的继承，使用 std::function 代替多态实现。

每个指令类中定义 lambda 表达式，实现相应的操作逻辑。

2.重写指令类：

每个指令类实现一个 const std::function<void(PoseHandler&)> operate，用于直接操作 PoseHandler。

3.修改指令解析逻辑：

在 ExecutorImpl::Execute 中，构造指令映射表 commandMap，将字符与操作绑定为键值对。

使用表驱动解析并执行指令。

4.支持组合功能：

修改 PoseHandler::Move，支持倒车与加速模式的叠加逻辑。

在测试用例中验证组合指令（如 BFM）。

八、实验数据及结果分析：

代码结构：

- PoseHandler.hpp / PoseHandler.cpp：管理小车状态，提供 Move、TurnLeft 等操作。
- Command.hpp：定义指令类，每个指令类实现 std::function 的操作。
- ExecutorImpl.hpp / ExecutorImpl.cpp：实现指令解析与执行逻辑。
- ExecutorTest.cpp：单元测试代码，验证指令的正确性。

测试用例：

使用 Google Test 进行单元测试，测试结果如下：

1. **基本功能测试：**

   测试初始化状态是否正确。

   测试 M、L、R 指令的功能。

2. **倒车模式测试：**

   验证倒车模式下的 M、L、R 是否正确执行。

   测试切换倒车模式的边界情况。

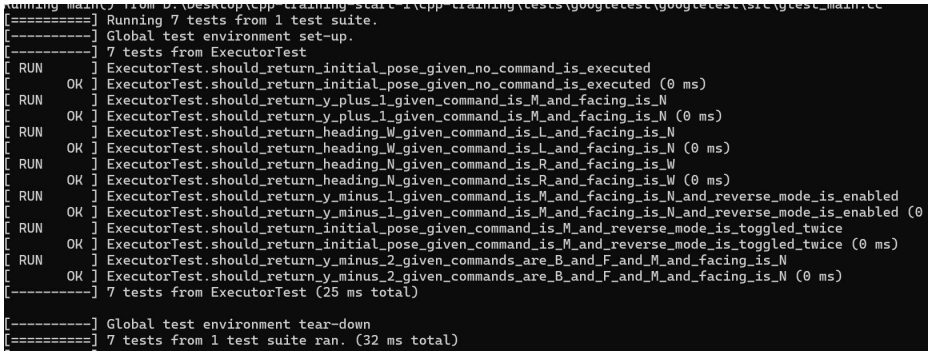
3. **加速模式测试：**

   验证加速模式是否使移动距离增加两倍。

4. **组合功能测试：**

   测试复杂指令序列，如 BFM、FFR。

结果：



九、实验结论：

1. **目标达成：**
   实验成功完成了从面向对象编程到函数式编程的转换，提升了代码简洁性和可扩展性。
   表驱动有效地降低了循环复杂度，指令解析逻辑更加直观。
   成功实现倒车与加速模式的功能，并验证了组合指令的正确性。
2. **关键改进：**
   使用 lambda 表达式和 std::function 取代多态继承，显著简化了代码结构。
   将状态管理封装到 PoseHandler 中，减少了模块间的耦合。

十、总结及心得体会：

本次实验让我深入理解了面向对象和函数式编程的优劣对比。
表驱动模式的引入，使得复杂逻辑得以简化，并易于扩展。
通过状态抽象与数据封装，代码的可维护性显著提升。
在实现组合功能时，倒车与加速模式的叠加逻辑尤为复杂，但通过精细测试得到了验证。

十一、对本实验过程及方法、手段的改进建议：

引入更多复杂指令序列的测试用例，确保功能的全面性。
优化 PoseHandler 的状态管理，使用数学运算简化方向切换逻辑。
在指令解析中加入异常处理机制，防止无效指令的执行。

报告评分：

指导教师签字：