

# 电子科技大学

## 计算机专业类课程

# 实验报告

课程名称: C++语言程序设计

学院专业: 计算机科学与工程学院

学生姓名: 何阳

学号: 2024080903004

指导教师: 郭磊

日期: 2025 年 11 月 21 日

# 电子科技大学

# 实验报告

## 实验四

### 一、实验室名称：

电子科技大学清水河校区主楼 A2-413-1

### 二、实验项目名称：

C++智能车项目组件

### 三、实验目的：

- 综合应用前面实验所学的高质量编程规范、面向对象编程、函数式编程、版本管理等知识。
- 掌握面向接口编程和泛型编程技术，提升代码的通用性和效率。
- 学会任务编排和目录结构优化，将复杂问题分解为原子操作。
- 实现多车辆类型支持，完成 Executor 组件的综合功能整合与优化。

### 四、实验器材（设备、元器件）：

操作系统：Windows11  
编译程序：Visual Studio Code  
编译器：MinGW64  
操作系统：Windows 10  
开发工具：Visual Studio Code、Git  
测试框架：Google Test (gtest)

## 五、实验主要内容：

### （一）面向接口编程

1. 定义 `IVehicle` 抽象接口，包含 `Init`、`ExecuteCommand`、`GetPose` 等通用方法。
2. 实现 `CarExecutor`、`TruckExecutor` 等子类，分别适配不同车辆的指令行为（如卡车倒车速度不同）。

### （二）任务编排

1. 将复杂指令序列（如"MLMRF"）分解为原子指令执行单元，通过队列管理执行顺序。
2. 实现指令预处理功能，支持指令合法性校验和优先级排序。

### （三）目录结构优化

1. 按照"功能模块"划分目录：`include`（接口定义）、`src`（实现代码）、`tests`（测试用例）、`script`（编译脚本）。
2. 优化 `CMakeLists.txt`，实现模块化编译和测试自动化。

### （四）综合功能测试与优化

1. 设计综合测试场景，覆盖多车辆类型、复杂指令序列、状态叠加等场景。
2. 性能优化：使用智能指针管理对象生命周期，通过 `STL` 容器提升数据操作效率。

## 六、实验结果

1. 成功实现多车辆类型支持，不同车辆可根据自身特性处理指令。
2. 复杂指令序列执行正常，任务编排逻辑高效可靠。
3. 目录结构清晰，代码模块化程度高，编译和测试自动化实现。
4. 所有综合测试用例（共 20 个）全部通过，性能满足要求。
4. Gtest 截图如下

```

PS C:\Users\Suuny\Desktop\repository\C++作业\build\debug> .\test_executor.exe
[=====] Running 20 tests from 1 test suite.
[-----] Global test environment set-up.
[-----] 20 tests from ExecutorTest
[ RUN      ] ExecutorTest.DefaultInitialization
[ OK       ] ExecutorTest.DefaultInitialization (0 ms)
[ RUN      ] ExecutorTest.CustomInitialization
[ OK       ] ExecutorTest.CustomInitialization (0 ms)
[ RUN      ] ExecutorTest.MoveForwardCommand
[ OK       ] ExecutorTest.MoveForwardCommand (0 ms)
[ RUN      ] ExecutorTest.TurnLeftCommand
[ OK       ] ExecutorTest.TurnLeftCommand (0 ms)
[ RUN      ] ExecutorTest.TurnRightCommand
[ OK       ] ExecutorTest.TurnRightCommand (0 ms)
[ RUN      ] ExecutorTest.BatchCommandsExecution
[ OK       ] ExecutorTest.BatchCommandsExecution (0 ms)
[ RUN      ] ExecutorTest.GetStatusMethod
[ OK       ] ExecutorTest.GetStatusMethod (0 ms)
[ RUN      ] ExecutorTest.AccelerationToggle
[ OK       ] ExecutorTest.AccelerationToggle (0 ms)
[ RUN      ] ExecutorTest.AccelerationMoveForward
[ OK       ] ExecutorTest.AccelerationMoveForward (0 ms)
[ RUN      ] ExecutorTest.AccelerationTurnLeft
[ OK       ] ExecutorTest.AccelerationTurnLeft (0 ms)
[ RUN      ] ExecutorTest.AccelerationTurnRight
[ OK       ] ExecutorTest.AccelerationTurnRight (0 ms)
[ RUN      ] ExecutorTest.AccelerationComplex
[ OK       ] ExecutorTest.AccelerationComplex (0 ms)
[ RUN      ] ExecutorTest.ReverseToggle
[ OK       ] ExecutorTest.ReverseToggle (0 ms)
[ RUN      ] ExecutorTest.ReverseModeCommands
[ OK       ] ExecutorTest.ReverseModeCommands (0 ms)
[ RUN      ] ExecutorTest.Combined_F_B_State_Commands
[ OK       ] ExecutorTest.Combined_F_B_State_Commands (0 ms)
[ RUN      ] ExecutorTest.CombinedStateComplex
[ OK       ] ExecutorTest.CombinedStateComplex (0 ms)
[ RUN      ] ExecutorTest.TurnRoundNormal
[ OK       ] ExecutorTest.TurnRoundNormal (0 ms)
[ RUN      ] ExecutorTest.TurnRoundAccelerated
[ OK       ] ExecutorTest.TurnRoundAccelerated (0 ms)
[ RUN      ] ExecutorTest.TurnRoundUnaffectedByReverse
[ OK       ] ExecutorTest.TurnRoundUnaffectedByReverse (0 ms)
[ RUN      ] ExecutorTest.TurnRoundComplex
[ OK       ] ExecutorTest.TurnRoundComplex (0 ms)
[-----] 20 tests from ExecutorTest (4 ms total)

[-----] Global test environment tear-down
[=====] 20 tests from 1 test suite ran. (5 ms total)
[ PASSED ] 20 tests.

```