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：**高质量编程基础

三、**实验目的：**

掌握开发者测试方法，确保代码正确性和需求完整性。

学习代码命名规范，提高代码可读性。

掌握 Git 的版本管理规范，建立良好的代码管理习惯。

四、**实验内容：**

需求澄清：明确开发目标。

接口设计：设计清晰的接口规范。

命名实践：应用命名原则命名变量、类和方法。

开发者测试：利用单元测试框架构建测试用例。

代码版本管理：应用 Git 管理代码提交与分支。

五、**实验器材（设备、元器件）：**

开发环境：VS Code、C++编译器。

版本控制工具：Git。

测试工具：Google Test 框架

六、实验步骤及操作：

需求明确：

- 设计 Executor 类，负责初始化车辆位置 (x, y, heading) 和支持以下操作：
  - **移动指令**：M 前进一步；
  - **左转指令**：L 左转 90 度，保持位置不变；
  - **右转指令**：R 右转 90 度，保持位置不变；
  - **查询指令**：获取车辆当前位置和朝向。

代码编写与实现：

- 开发 Executor 类，并定义基本接口，具体功能包括：

```
void initialize(int startX, int startY, char startHeading);  // 初始化位置
void executeCommand(char command);                           // 执行指令
std::tuple<int, int, char> getCurrentPosition();             // 获取当前位置
```

测试功能：

- 编写 Google Test 单元测试，包括以下场景：
  - 默认位置和朝向 (0, 0, N)；
  - 执行移动指令 M；
  - 执行左转指令 L 和右转指令 R。

代码提交：

- 使用 Git 进行版本控制，分支管理包括以下操作：
  - 提交初始代码；
  - 增加测试用例；
  - 通过小步提交实现功能代码；
  - 最终提交通过的完整功能。

八、实验数据及结果分析：

测试代码：

```
TEST(ExecutorTest, should_return_initial_pose_given_no_command_is_executed) {
    Pose initialPose = {2, 3, 'E'};
    Executor* executor = Executor::NewExecutor(initialPose);
    Pose pose = executor->Query();
    EXPECT_EQ(pose.x, 2);
    EXPECT_EQ(pose.y, 3);
    EXPECT_EQ(pose.heading, 'E');
    delete executor;
}

// 测试基础指令
TEST(ExecutorTest, should_return_y_plus_1_given_command_is_M_and_facing_is_N) {
    Pose initialPose = {0, 0, 'N'};
    Executor* executor = Executor::NewExecutor(initialPose);

    executor->Execute("M");
    Pose pose = executor->Query();
    EXPECT_EQ(pose.x, 0);
    EXPECT_EQ(pose.y, 1);
    EXPECT_EQ(pose.heading, 'N');

    delete executor;
}
```

测试结果：所有测试用例通过，功能正确。

```
[ RUN      ] ExecutorTest.should_return_initial_pose_given_no_command_is_executed
[       OK ] ExecutorTest.should_return_initial_pose_given_no_command_is_executed (0 ms)
[ RUN      ] ExecutorTest.should_return_y_plus_1_given_command_is_M_and_facing_is_N
[       OK ] ExecutorTest.should_return_y_plus_1_given_command_is_M_and_facing_is_N (0 ms)
[ RUN      ] ExecutorTest.should_return_heading_W_given_command_is_L_and_facing_is_N
[       OK ] ExecutorTest.should_return_heading_W_given_command_is_L_and_facing_is_N (0 ms)
[ RUN      ] ExecutorTest.should_return_heading_N_given_command_is_R_and_facing_is_W
[       OK ] ExecutorTest.should_return_heading_N_given_command_is_R_and_facing_is_W (0 ms)
```

Git 提交记录：包括创建分支、提交日志等。

九、实验结论：

通过本次实验，掌握了高质量代码开发的基本流程，包括需求分析、接口设计、测试驱动开发和版本管理。验证了 Executor 类的功能实现，能够正确执行移动、转向和状态查询指令。实验达到了预期目标。

十、总结及心得体会：

1. **代码质量**：本次实验强化了代码规范性和测试驱动开发的意识，提高了代码的可读性和可维护性。
2. **版本管理**：使用 Git 进行分支管理和小步提交，能够清晰记录代码变更历史，为代码审查和问题回溯提供便利。
3. **问题解决能力**：在代码开发和测试过程中，通过查阅资料和不断调试，解决了函数接口设计、测试用例设计等问题，提升了实践能力。

十一、对本实验过程及方法、手段的改进建议：

1. **增加复杂场景测试**：如支持连续指令的执行和状态切换。
2. **扩展团队协作内容**：模拟多人协作开发，增强团队合作能力。
3. **优化测试覆盖率**：增加边界值和异常输入的测试用例，确保代码健壮性。

报告评分：

指导教师签字：