

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： C++ 面向对象编程实验	
<p>三、实验目的：</p> <p>掌握面向对象编程的三大特性：封装、继承、多态。</p> <p>理解如何通过面向对象技术提升代码的复用性和可扩展性。</p> <p>实现复杂指令逻辑的模块化，降低代码圈复杂度。</p> <p>使用 Google Test 进行单元测试验证程序的正确性。</p>	
<p>四、实验内容：</p> <p>重构现有指令逻辑（M、L、R），拆分到独立的类中。</p> <p>添加新的加速指令（F）功能，支持移动、转向操作的扩展行为。</p> <p>利用面向对象的多态特性，实现统一的指令处理机制。</p> <p>编写并运行完善的单元测试，验证各项功能的正确性。</p>	
<p>五、实验器材（设备、元器件）：</p> <p>实验平台： Windows 10 操作系统。</p> <p>编译器： g++ 11.2 或以上版本。</p> <p>集成开发环境（IDE）： Visual Studio Code。</p> <p>测试框架： Google Test。</p>	

## 六、实验步骤及操作：

### 初始化项目：

创建包含 `Executor.hpp`、`ExecutorImpl.hpp` 和 `ExecutorImpl.cpp` 的项目。

初始化 Git 仓库。

### 第一步：重构现有代码：

提取 `Move`、`TurnLeft`、`TurnRight` 方法，降低圈复杂度。

测试代码运行情况并提交至版本库。

### 第二步：封装指令类：

添加 `ICommand` 接口及其派生类 `MoveCommand`、`TurnLeftCommand`、`TurnRightCommand`。

修改 `Execute` 方法，实现通过多态处理指令。

验证代码正常运行并提交至版本库。

### 第三步：新增 F 指令：

添加 `FastCommand` 类及 `Fast` 状态管理函数。

修改 `MoveCommand`、`TurnLeftCommand` 和 `TurnRightCommand`，支持加速状态。

测试 F 指令功能并提交代码。

### 第四步：编写测试用例：

使用 Google Test 为所有指令逻辑添加测试用例。

运行测试确保功能正确，提交最终代码。

### 编译和验证：

使用 CMake 编译项目，确保无语法或运行时错误。

检查测试用例覆盖率。

## 八、实验数据及结果分析：

**测试结果：** 运行测试用例结果如下：

```
Running main() from D:\Desktop\cpp-training-start-1\cpp-training\tests\googletest\googletest\src\gtest_main.cc
[=====] Running 8 tests from 2 test suites.
[-----] Global test environment set-up.
[-----] 4 tests from ExecutorTest
[ RUN      ] ExecutorTest.should_return_initial_pose_when_initialized (0 ms)
[ OK       ] ExecutorTest.should_return_initial_pose_when_initialized (0 ms)
[ RUN      ] ExecutorTest.should_return_y_plus_1_when_command_is_M_and_facing_N (0 ms)
[ OK       ] ExecutorTest.should_return_y_plus_1_when_command_is_M_and_facing_N (0 ms)
[ RUN      ] ExecutorTest.should_return_y_plus_1_when_command_is_L_and_facing_N (0 ms)
[ OK       ] ExecutorTest.should_return_y_plus_1_when_command_is_L_and_facing_N (0 ms)
[ RUN      ] ExecutorTest.should_return_facing_W_when_command_is_L_and_facing_N (0 ms)
[ OK       ] ExecutorTest.should_return_facing_W_when_command_is_L_and_facing_N (0 ms)
[ RUN      ] ExecutorTest.should_return_facing_E_when_command_is_R_and_facing_N (0 ms)
[ OK       ] ExecutorTest.should_return_facing_E_when_command_is_R_and_facing_N (0 ms)
[-----] 4 tests from ExecutorTest (12 ms total)

[-----] 4 tests from ExecutorFastTest
[ RUN      ] ExecutorFastTest.should_return_x_plus_2_given_status_is_fast_command_is_M_and_facing_is_E (0 ms)
[ OK       ] ExecutorFastTest.should_return_x_plus_2_given_status_is_fast_command_is_M_and_facing_is_E (0 ms)
[ RUN      ] ExecutorFastTest.should_return_M_and_x_plus_1_given_status_is_fast_command_is_L_and_facing_is_E (0 ms)
[ OK       ] ExecutorFastTest.should_return_M_and_x_plus_1_given_status_is_fast_command_is_L_and_facing_is_E (0 ms)
[ RUN      ] ExecutorFastTest.should_return_S_and_x_plus_1_given_status_is_fast_command_is_R_and_facing_is_E (0 ms)
[ OK       ] ExecutorFastTest.should_return_S_and_x_plus_1_given_status_is_fast_command_is_R_and_facing_is_E (0 ms)
[ RUN      ] ExecutorFastTest.should_return_y_plus_1_given_command_is_FFM_and_facing_is_N (0 ms)
[ OK       ] ExecutorFastTest.should_return_y_plus_1_given_command_is_FFM_and_facing_is_N (0 ms)
[-----] 4 tests from ExecutorFastTest (13 ms total)

[-----] Global test environment tear-down
[=====] 8 tests from 2 test suites ran. (32 ms total)
[ PASSED ] 8 tests.
```

所有功能通过测试，包括基础功能（M、L、R）和新增 F 指令的行为。  
提高了代码的复用性和可扩展性。

九、实验结论：

通过本次实验，掌握了面向对象编程的基本特性及其在实际开发中的应用。通过指令类的封装和继承关系，程序变得更模块化，指令逻辑更清晰，扩展性也显著增强。

十、总结及心得体会：

本次实验增强了对面向对象编程思想的理解，尤其是如何运用多态性减少代码耦合。  
通过实现 F 指令的需求，深刻体会了封装和抽象的重要性。  
Google Test 的使用提高了开发效率，同时也验证了程序的鲁棒性。

十一、对本实验过程及方法、手段的改进建议：

实验步骤可以提供更详细的代码模板，以降低调试时间。  
增加对于测试框架的使用指导，例如如何处理特殊边界条件。  
建议补充更多实际案例来强化对多态性和继承关系的理解。

报告评分：

指导教师签字：

