

《面向对象程序设计》课程设计报告

****

题 目 战棋类小游戏

姓 名 倪俊锋

学 号 202183290391

学 院 计算机学院

专 业 计算机科学与技术

二Ｏ二二 年 十二 月 日

**目 录**

[一、 项目目标 3](#_Toc122957171)

[二、 需求分析 3](#_Toc122957172)

[三、 功能描述 4](#_Toc122957173)

[四、设计 8](#_Toc122957174)

[五、开发及测试 11](#_Toc122957175)

[参考文献 16](#_Toc122957176)

# 一、 项目目标

本项目预期在两周内使用 C++编程语言和面向对象的软件开发方法实现一款通过控制台操作的“战棋”小游戏。该游戏应当具备基本的商店购买单位组建队伍、显示回合对局、单位自走战斗以及判断胜负等功能。

# 二、 需求分析

* 1. 功能性需求

本项目有以下必须实现的功能：

* + 1. 游戏开始菜单：供玩家选择开始游戏或退出游戏；
    2. 回合开始商店界面：供玩家选择商店内的单位，通过购买、出售组建队伍
    3. 显示回合对局：实现基于控制台的队伍单位可视化；
    4. 回合中战斗阶段：实现单位的自行移动战斗，俗称自走棋；
    5. 判断胜负：提示玩家胜负信息。
  1. 非功能性需求

本项目有以下非功能性需求：

1. 界面需求：游戏界面应简洁明了，有良好观感；
2. 易用性需求：游戏的操控方式、按键布局应尽量符合玩家习惯；
3. 安全性需求：开发过程中应避免产生安全漏洞。
   1. 设计约束

本项目有以下设计约束条件：

1. 成员函数必须在类中定义，在类外实现；
2. 尽可能使用避免不必要的类、函数、变量；
3. 类、函数、变量的命名必须具有可读性。

# 

# 三、 功能描述

* 1. 开始和退出游戏

开始界面提供“start game”和“exit game”以及“game rule”(如图3.1.1)，供玩家选择。若选择“start game”则进入回合开始商店界面（如图3.1.2），若选择“exit game”则关闭控制台（如图3.1.3），若选择“game rule”则在控制台中打印游戏规则供玩家查阅，之后可通过“exit? 1.yes 2.no”选择回到主菜单或者退出游戏。（如图3.1.4）

图形用户界面

描述已自动生成

图3.1.1

文本

描述已自动生成

图3.1.2

图片包含 日程表

描述已自动生成

图3.1.3

* 1. 回合开始商店界面

本程序将商店的单位和玩家的队伍打印在控制台上，实现可视化操作。(如图3.1.2打印了商店界面和玩家队伍信息)。

玩家进行购买和出售操作同样会打印玩家的当前队伍信息，实现可视化操作。（如图3.1.3）当组建队伍完成，玩家可选择“Battle!”结束商店阶段，进入战斗阶段。

文本

描述已自动生成

图3.1.2-图3.1.3

* 1. 回合中战斗界面

在战斗阶段开始时，程序会打印显示玩家的队伍信息以及AI的队伍信息（如图3.3.1）本阶段会用玩家队伍与AI队伍进行对战，采取一对一、角色死亡则下一

个角色继续战斗的方式，每个角色会在攻击完毕后触发自身的特殊攻击。

玩家的队伍不需要依位置顺序组建队伍，允许位置有空缺，在此基础上战斗回合仍能实现自动战斗，一个单位死亡则下一个单位继续战斗，不会出现位置空缺而战斗的一方无单位可进行战斗的情况。最后谁的队伍还有角色在场则获得回合胜利。（自走棋模式）(如图3.3.2)

文本

描述已自动生成

图3.3.1

* 1. 回合终结算

战斗阶段结束后，进行回合结算，在控制台上打印输出玩家该回合是胜还是负。根据规则，一局游戏玩家的生命值为4，输掉一个回合扣除1点，直到生命值为零游戏结束。AI队伍的生命同样为4，使AI生命削减到0则游戏胜利。(如图3.4.1)

文本

描述已自动生成

图3.4.1

文本

描述已自动生成

图3.4.2

# 四、设计

* 1. UML 类图

图示

描述已自动生成

对以上UML类图进行说明：

本程序主要包含四个类，分别是role（单位类）、team(队伍类)、shop(商店类)、Gamemanager类（控制游戏过程主体）。

其中，team类和role类是聚合关系，role是team的一部分，但也可脱离team独立存在。shop类和team类是依赖关系，shop中的成员函数需要使用team类型的参数，并需要调用team中的addrole成员函数和deleterole成员函数。

shop类与role类是依赖关系，shop类中的成员函数Buy使用了role类型对象作为局部变量，且调用了role中的私有成员函数setJob。

Gamemanager与team类和shop类是组合关系，在Gamemanager类中直接定义了team类型的对象和shop类型的对象，并且调用两个类中的成员函数。Gamemanager与role类是依赖关系，在成员函数中使用了role类的局部变量。

* 1. 枚举类型Job

Job枚举类用于存放单位的所有职业，所有类的实现都需要调用该枚举类型中的值，因此在role类型中给出了一个将枚举类型Job转换为字符串类型的接口。

如下图：



图片包含 应用程序

描述已自动生成

* 1. role类

由于本程序的代码中有大量与单位相关的操作，故将与单位有关的操作单独抽象出一个role类，用于实现队伍增加单位、删除单位，商店购买单位等操作。

具体的类成员及各个成员函数功能如下图所示：图形用户界面, 文本

描述已自动生成

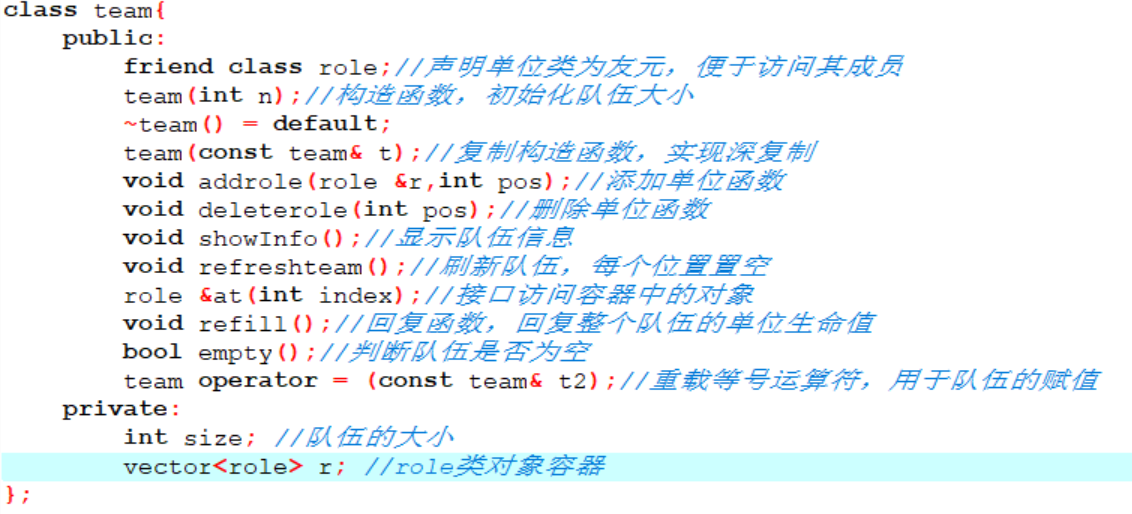
* 1. shop类

一个回合中开始阶段就是商店界面的相关操作，玩家通过购买单位、出售单位来组建队伍。将这些操作单独抽象出一个类shop,同时创建有关函数接口，方便外部类的调用。具体的类成员及各个成员函数功能如下图所示： 文本, 信件

描述已自动生成

* 1. team类

玩家购买出售单位组成的队伍单独封装为一个类，同时创建有关函数接口，方便外部调用。shop类依赖于team类的addrole与deleterole函数进行购买出售，并且team类也主要封装了role类对象的容器。具体的类成员及各个成员函数功能如下图所示：



* 1. Gamemanager类

GameManager 类是本程序能够正常运行的最关键部分。它包含几个重要的回合过程函数以及一个判断游戏是否结束的函数，这些函数要通过调用role类、team类及shop类的成员实现。具体的类成员及各个成员函数功能如下图所示：

图片包含 文本

描述已自动生成

五、开发及测试

5.1 类的构建

考虑战棋类游戏都有组建队伍、自走战斗的过程，所以构建组建队伍的商店类，队伍类和基础的单位类。最后发现游戏主体代码过于复杂不利于维护，于是构建Gamemanager类,每个回合模块作为其成员函数，最后结合胜负判断的结算成员函数构建游戏主体。由于角色的职业固定且经常要被调用，故设定枚举类型Job,并设定枚举转换字符串接口便于访问。

5.2 枚举类Job

枚举类主要存放单位的职业类型，同时none类型用于标识队伍空缺的位置，便于实现自走的战斗模式及删除角色操作，角色死亡后则被赋予该职业，便于判断。



文本

描述已自动生成

5.3 role类的创建

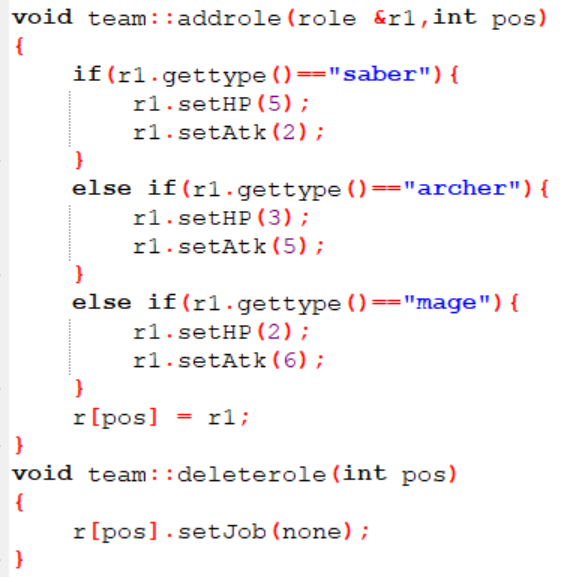
role类主要包含战斗阶段的attack函数(包含普通攻击和特殊攻击)，属于类对象之间的交互

图片包含 文本

描述已自动生成

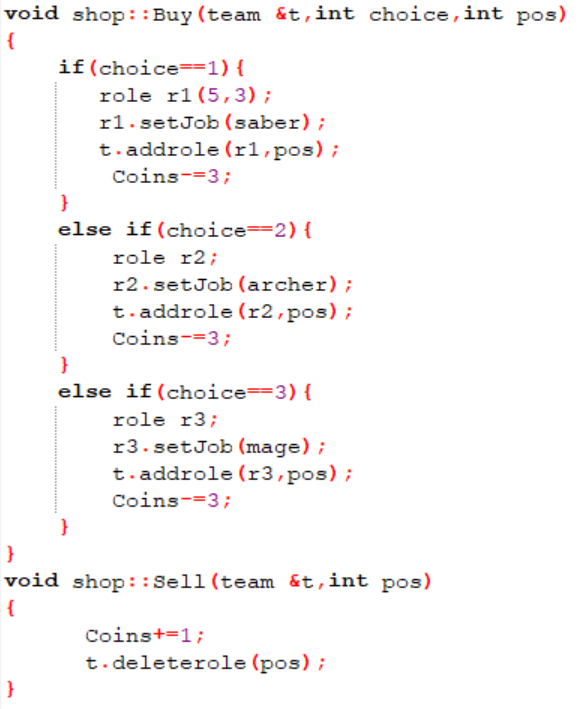
5.4 team类的创建

team类主要包含addrole和deleterole函数，用于添加或删除队伍中的单位。



5.5 shop类的创建

shop类主要调用team类中的addrole和deleterole进行购买和出售。



5.6 Gamemanager类的创建

该类主要战斗阶段的成员函数可实现功能：玩家的队伍不需要依位置顺序组建队伍，允许位置有空缺，在此基础上战斗回合仍能实现自动战斗，一个单位死亡则下一个单位继续战斗，不会出现位置空缺而战斗的一方无单位可进行战斗。（自走棋战斗模式）

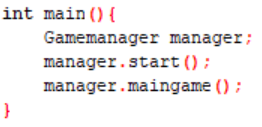
函数实现如下图：

图片包含 日程表

描述已自动生成

5.7 主函数的实现

主函数主要有打印开始界面和调用Gamemanager类的对象进行游戏循环的两个部分，因此较为简单，调用函数接口即可。



5.8 其他函数实现

其他函数实现在源码中，由于不属于重点故不在此处过多赘述。

六、总结

6.1 项目总结

本项目是结合 C++编程语言和面向对象的方法实现的一款战棋类小游戏。

优点：

开发过程中，开发者将单位、队伍、商店视为对象进行封装，通过类与类之间的函数接口调用实现了战棋类游戏运转的基本功能，结构较清晰，较利于维护。异常处理中处理了商店阶段组建队伍的溢出异常或是出售空位置的常见异常。

缺点：

由于开发者水平有限且开发周期短，本项目只实现了基本的人机对战回合流程，还缺少商店强化道具和商店随机刷新单位的功能，单位职业较少，平衡性较差。设计上，只是能形成一个大概的抽象设计实现相关功能，成员函数访问权限等设计思路并不太清晰。异常处理也仅处理了部分，如果游戏中途突然崩溃，那玩家组建的队伍也会随之流失，因为队伍只保存在内存中。

最后，希望以后在深度学习后能够进一步提升个人能力，对类的设计思想有进一步的提高。本项目还有很大的改进空间。

6.2 开发过程中遇到的问题及解决办法

在开发过程中遇到了许多问题，先列举如下：

* + 1. 代码设计思路上，开发者本来想将单位作为一个基类再派生出所有的职业，但是在构建队伍类时发现C++的对象数组不能存放派生类，因为每个位置的内存大小不好确定，但可以存放派生类的对象指针。奈何开发者代码功底实在有限，从构建智能指针数组开始就出现了大量的报错。一方面，开发者对智能指针容器这方面没有语法经验，涉及内存管理的操作不熟悉（尽管查阅了相关书籍，但是这方面能让开发者看懂的优秀代码也较少），另一方面，派生类也仅仅是多出一个特殊攻击的属性，因此最终还是采用对象数组这样的设计思路来实现队伍类。
    2. 由于设计思想功底较差，开发到最后脑海中类与类的关系变得混乱，最后收尾研究了UML图才差不多理顺了关系。
    3. 在战斗回合中，经常出现位置操控不好，有一方队伍单位无法与另一方的单位战斗，会与空位置战斗，并出现数组越界的情况。针对这一问题开发者花了很长的时间修改，首先是将队伍类中的对象数组改成用标准库的vector封装的role类型数组，然后关键是将队伍空缺的位置设置为none职业这一枚举类型，同时单位阵亡也会被设置为none类型。这样才算解决了自走战斗的问题。
    4. 在战棋游戏规则中，一回合结束的下一回合玩家的队伍状态应该保留，上一回合 购买的单位仍应该在相应位置上不变。同时保证单位的职业被更改的情况下仍能实现该功能，开发者使用了 重载赋值运算符函数的方式，在每一回合战斗阶段开始前将队伍信息保存下来，下一回合开始前再赋值给玩家队伍，实现了该功能。

6.3 新学到的知识/技能

(1) 补充了很多关于智能指针的相关知识，虽然没有用上，但有了一些内存管理的理论基础

(2) 熟悉了游戏主循环设计

(3) 熟悉了类与类之间函数接口调用的语法逻辑

(4) 常见和不常见的报错处理/语法规则，如

linker error:连接出错，可能是函数接口没写好/主体缺失

Process exit with return value 3221225477：内存泄漏/数组越界

等号运算符重载只能声明为成员函数

(5) UML类图类与类之间关系的箭头表示

(6) 补充了格式化输入输出的相关操作

# 参考文献

1. 郑莉, 董渊, 何江舟. C++语言程序设计[M]. 北京: 清华大学出版社, 2010.
2. [美] Stanley B. Lippman\_ [美] Josée Lajoie\_ [美] Barbara E. Moo - C++ Primer 中文版（第 5 版） 2013, 电子工业出版社
3. 刘伟 《深入浅出UML类图》 CSDN博客链接：[深入浅出UML类图 -C++ 技术中心-C++博客(cppblog.com)](http://www.cppblog.com/API/archive/2013/07/04/201506.html)

附录