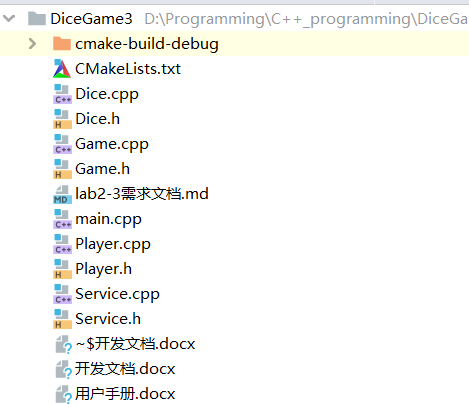
开发文档

——骰子游戏

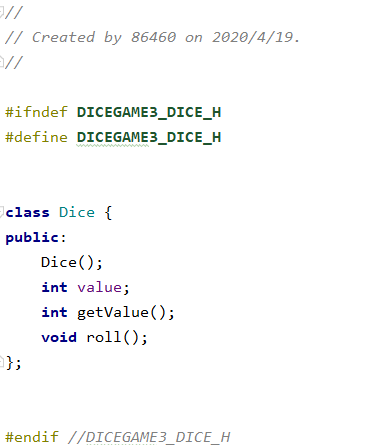
本次lab我采用了**面向对象**的设计方法，进行了抽象和分装

首先是目录结构

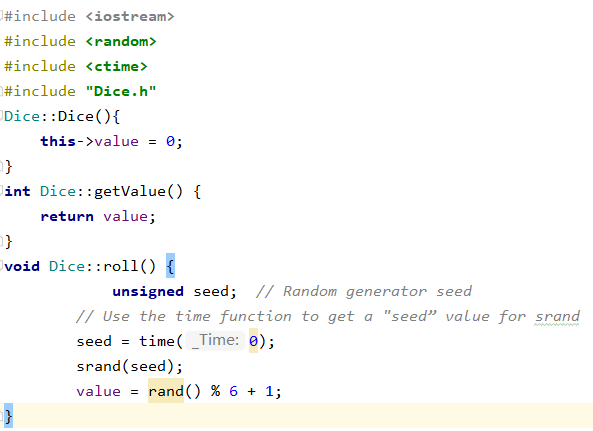


# 骰子类

## 头文件



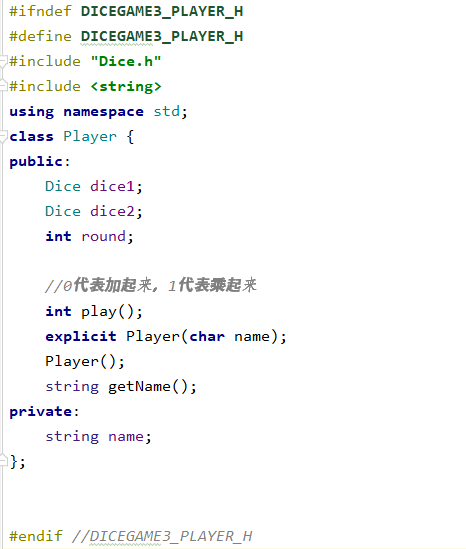
## 源码



自带属性值，还有获取随机值和外部获取值的方法。

# 玩家对象

## 头文件



## 源代码

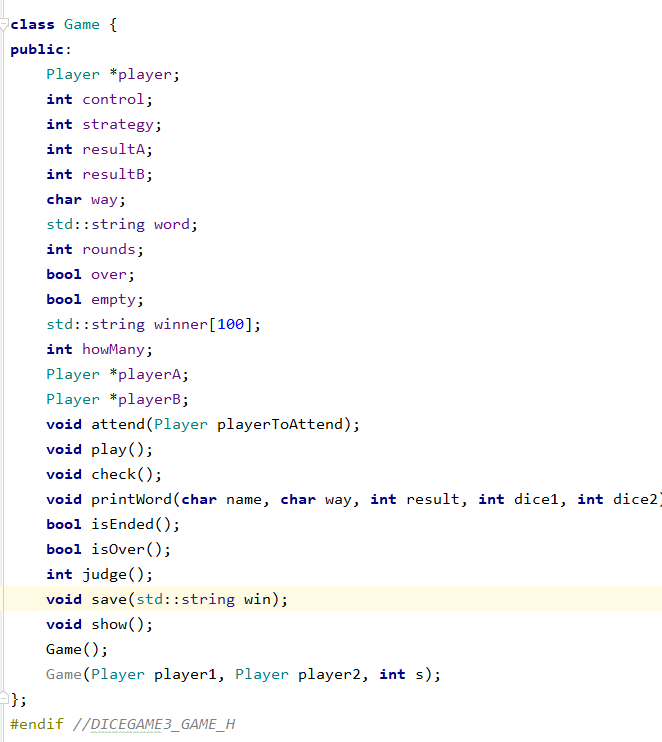


自带属性：两个骰子，游戏轮数，昵称

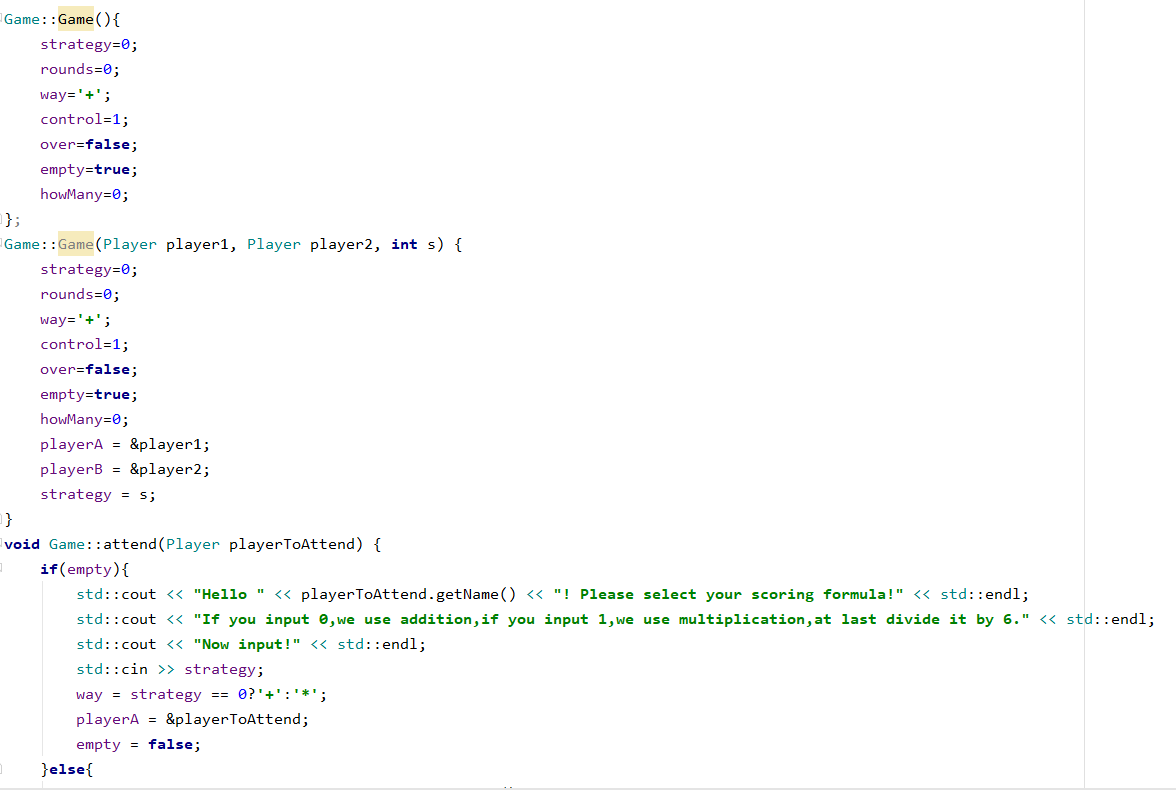
每轮扔一次骰子，一共扔两轮，外部可以获取他的点数，不用关心其实现方式

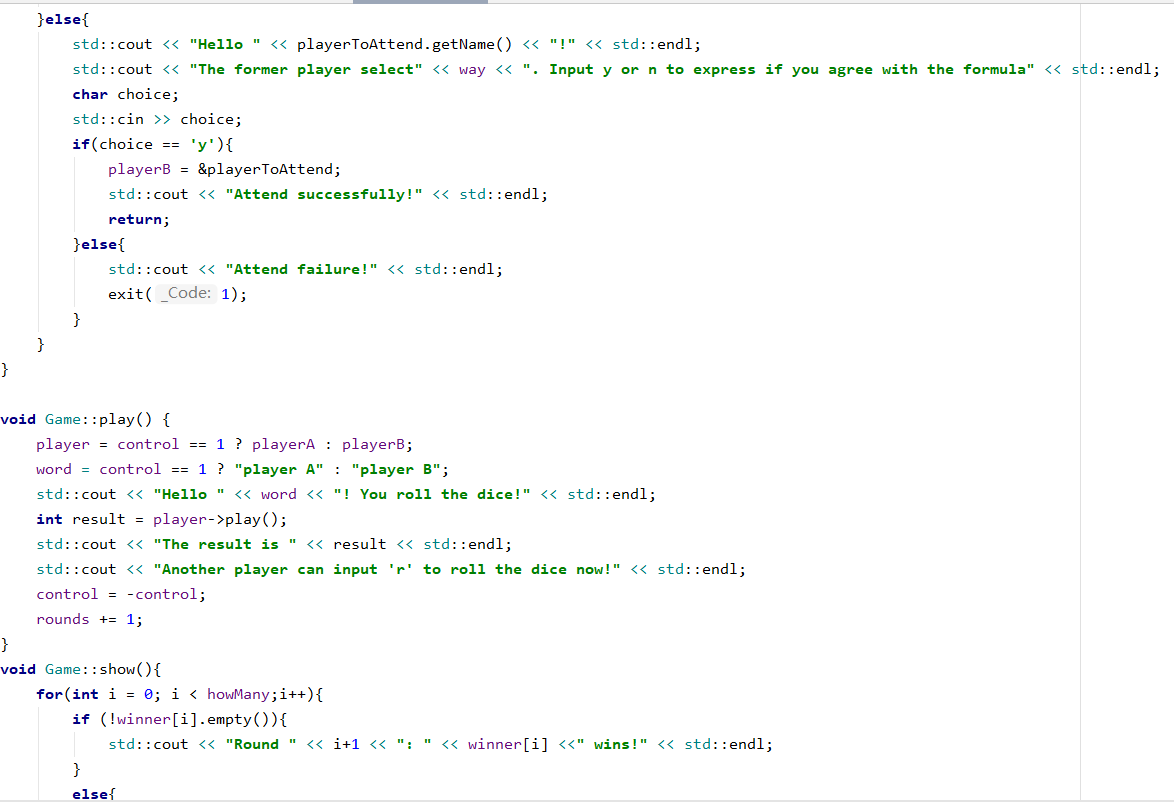
# 游戏对象

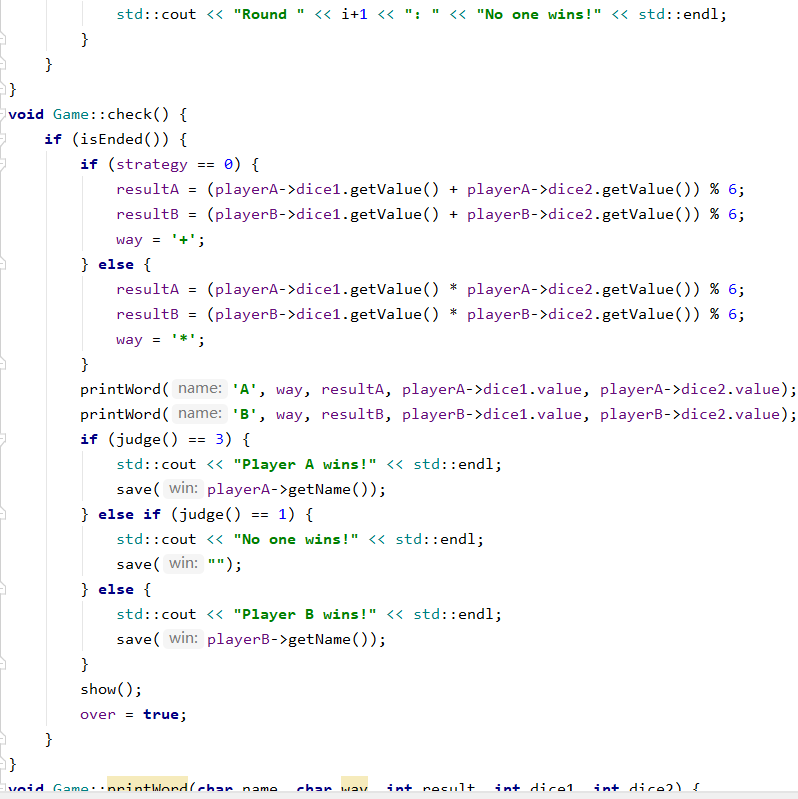
## 头文件

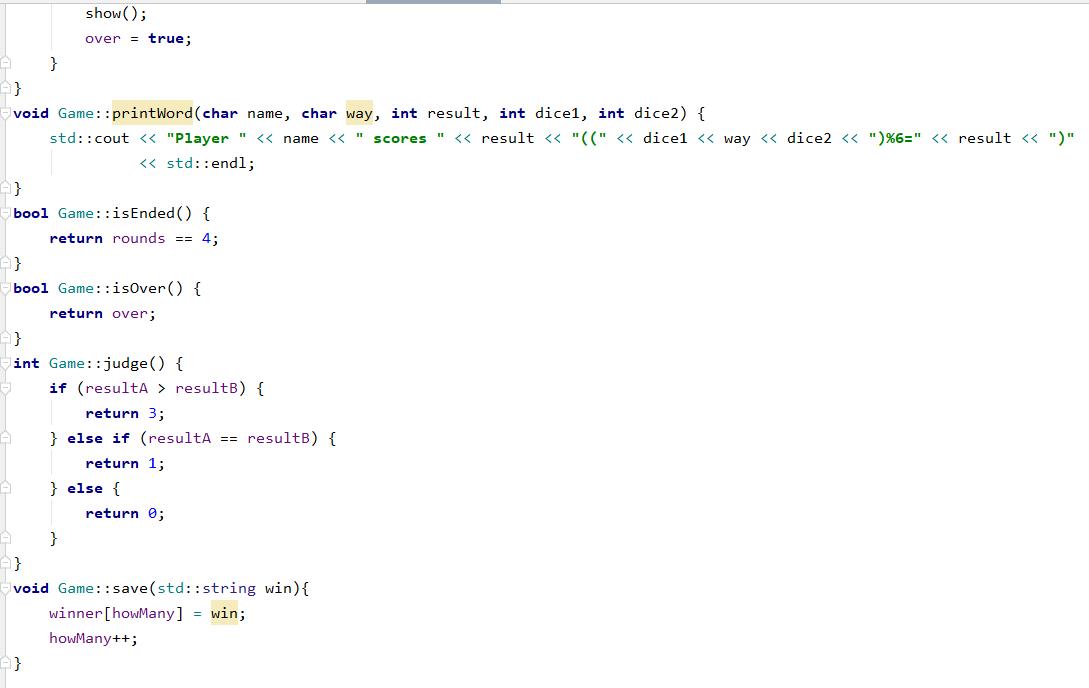


## 源代码









这个类是设计最为复杂的类

PlayerA,PlayerB每个游戏维护两个玩家，

\*player-维护一个当前玩家

strategy-维护一个计分策略

resultA,resultB-维护两个玩家的得分结果

empty-维护是否有玩家加入

over-维护游戏是否结束

way-维护运算符

howMany-维护当前进行盘数

rounds-维护当前游戏进行回合数

attend()方法-让玩家能加入游戏

check()方法：检查现在是否应该结束

isOver()方法：判断游戏是否结束

judge()方法：判定游戏结果

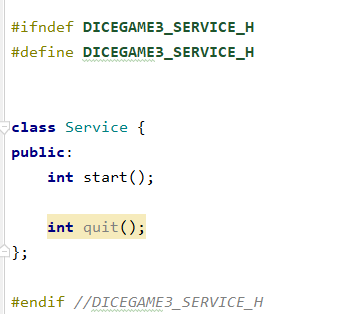
save()方法：保存历局游戏结果

show()方法：展示结果

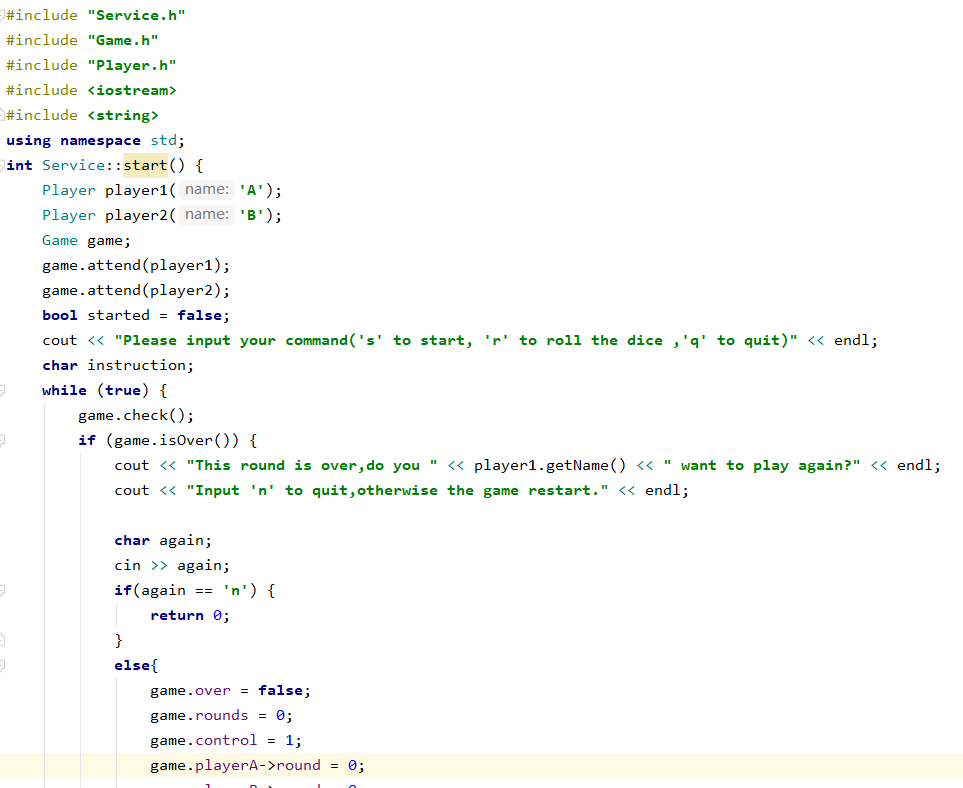
设计思路：两个玩家先后加入游戏，统一策略，游戏才能开始，玩家轮流投骰子，完毕后计算结果并记录，如果要再玩则重置暂时的游戏数据。

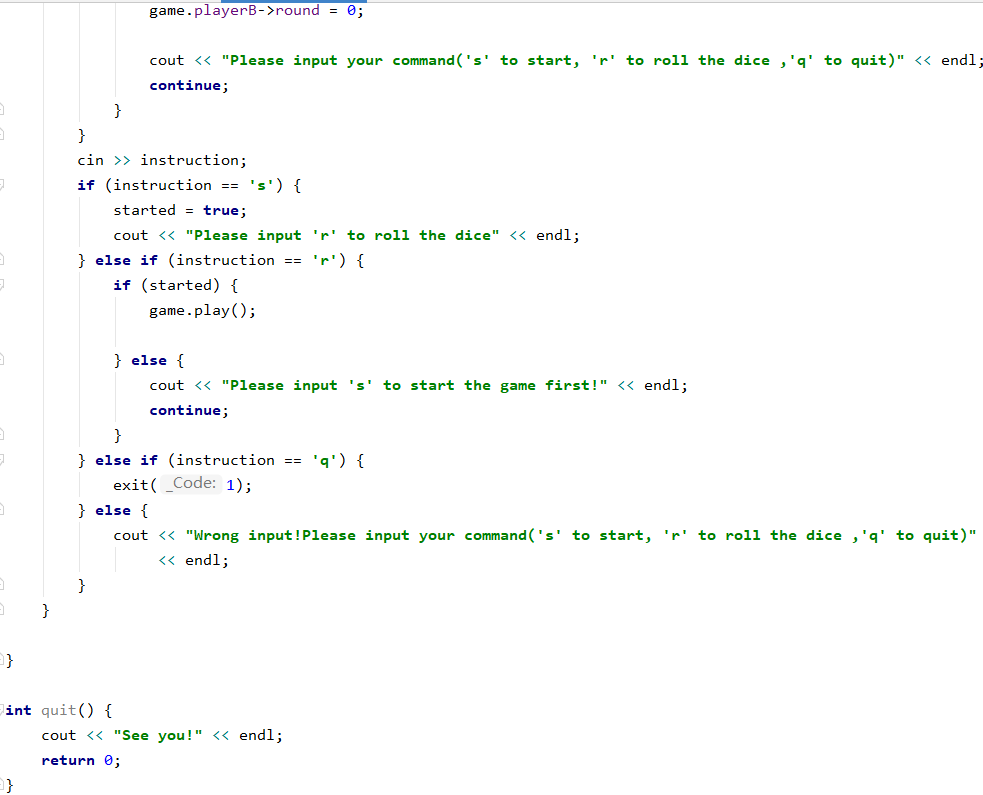
# Service类

## 头文件



## 源代码





主要处理游戏开始先后的逻辑

游戏、玩家、骰子维护在这个对象上，负责控制流程

接收指令，然后判断

本次lab我将各个类放到各个文件，更加抽象，同时认识到了面向对象程序设计的方便和松耦合