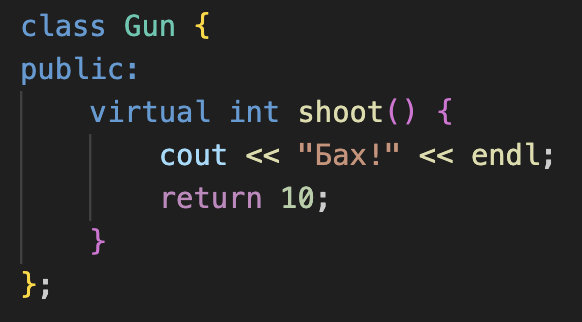
**Отчет курсанта Громова Григория Андреевича группы 22.Б05 о выполнении практического задания на тему «виртуальные функции» (девятое задание в третьем семестре)**

В самом начале изучения объектно-ориентированного программирования мы сформулировали три его главных принципа – наследование, инкапсуляция и полиморфизм. Если с двумя первыми мы уже сталкивались в предыдущих проектах, то последний до сих пор оставался почти незатронутым. В этом отчете я хочу изложить один из способов реализации полиморфизма – использование виртуальных функций.

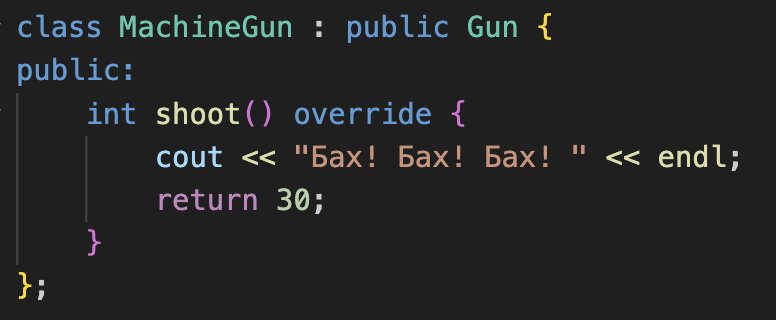
Рассмотрим проблему на примере. Допустим мы разрабатываем игру-стрелялку. Суть игры проста: есть несколько видов оружия, есть противник, у которого есть некоторое количество единиц здоровья и есть игрок, который может применять разное оружие и уменьшать количество единиц здоровья врага.

Среди оружия у нас есть:

1. Пистолет, у которого есть метод shoot(). Он пишет звук выстрела и возвращает урон. Функция помечена ключевым словом **virtual**, которое означает, что наследники класса могу переопределять эту функцию. Помимо этого, это означает, что если у нас есть указатель на класс Gun, и мы помести в него наследника класса Gun, то при вызове через указатель функции shoot, переопределенной в наследнике, то вызовется именно shoot наследника, а не родителя. Это замечание станет лучше ясно на пример, который бужет изложен ниже.



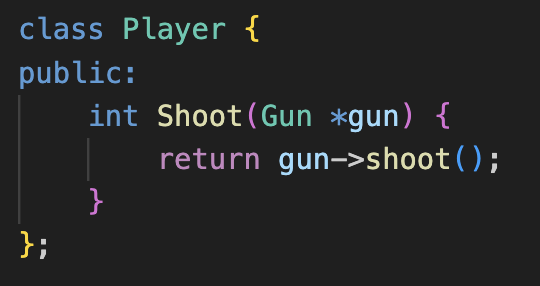
1. Автомат, он является наследником Gun и наносит урон в три раза больше его. Ключевое слово **override** в данном случае является необязательным. Оно нужно для того, чтобы компилятор проверял, что функция с таким названием и такой же сигнатурой есть в родительском классе и в противном случае выдавал ошибку.

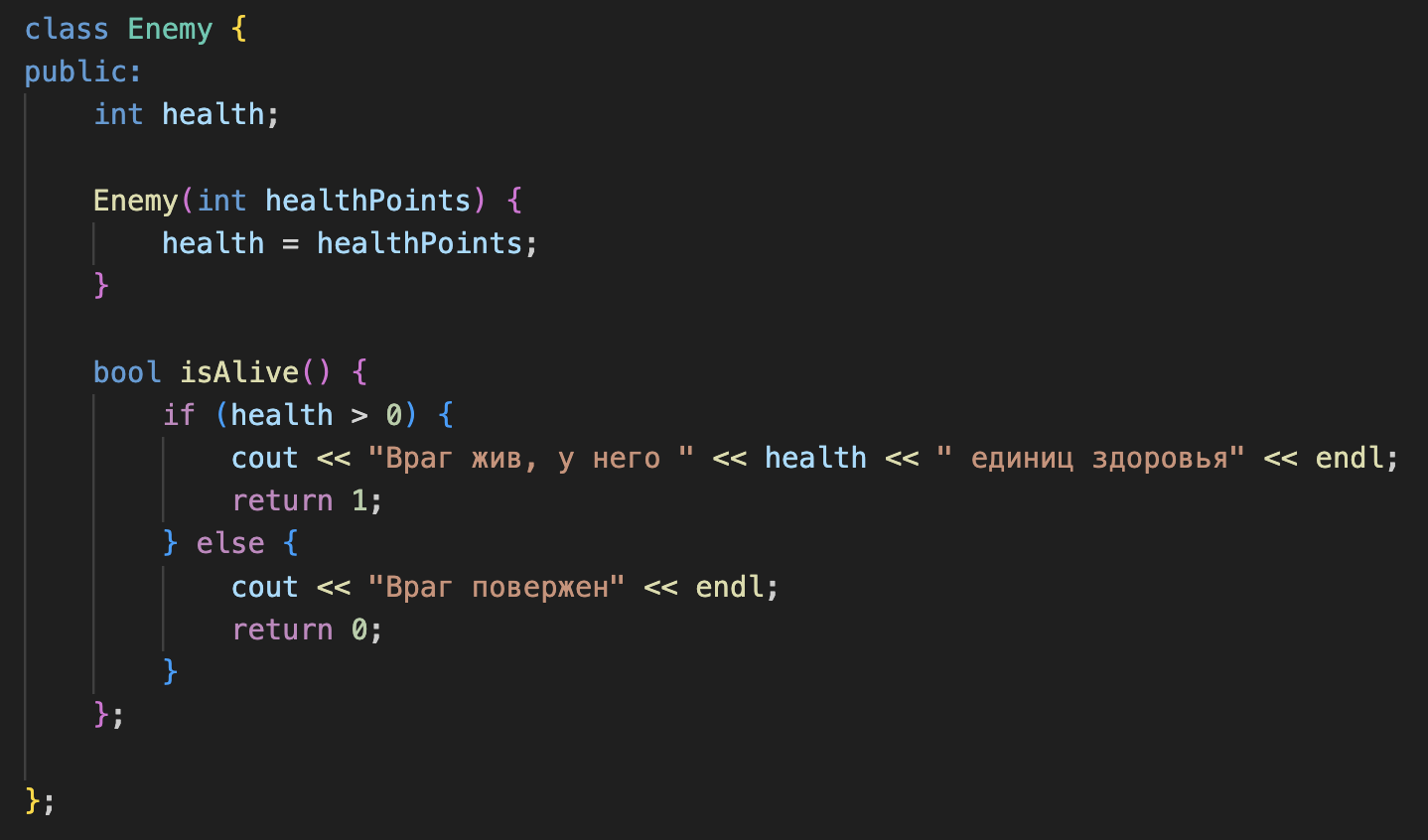


1. Аналогично добавляем оружие Ракета, у которого урон в 20 раз больше родителя.



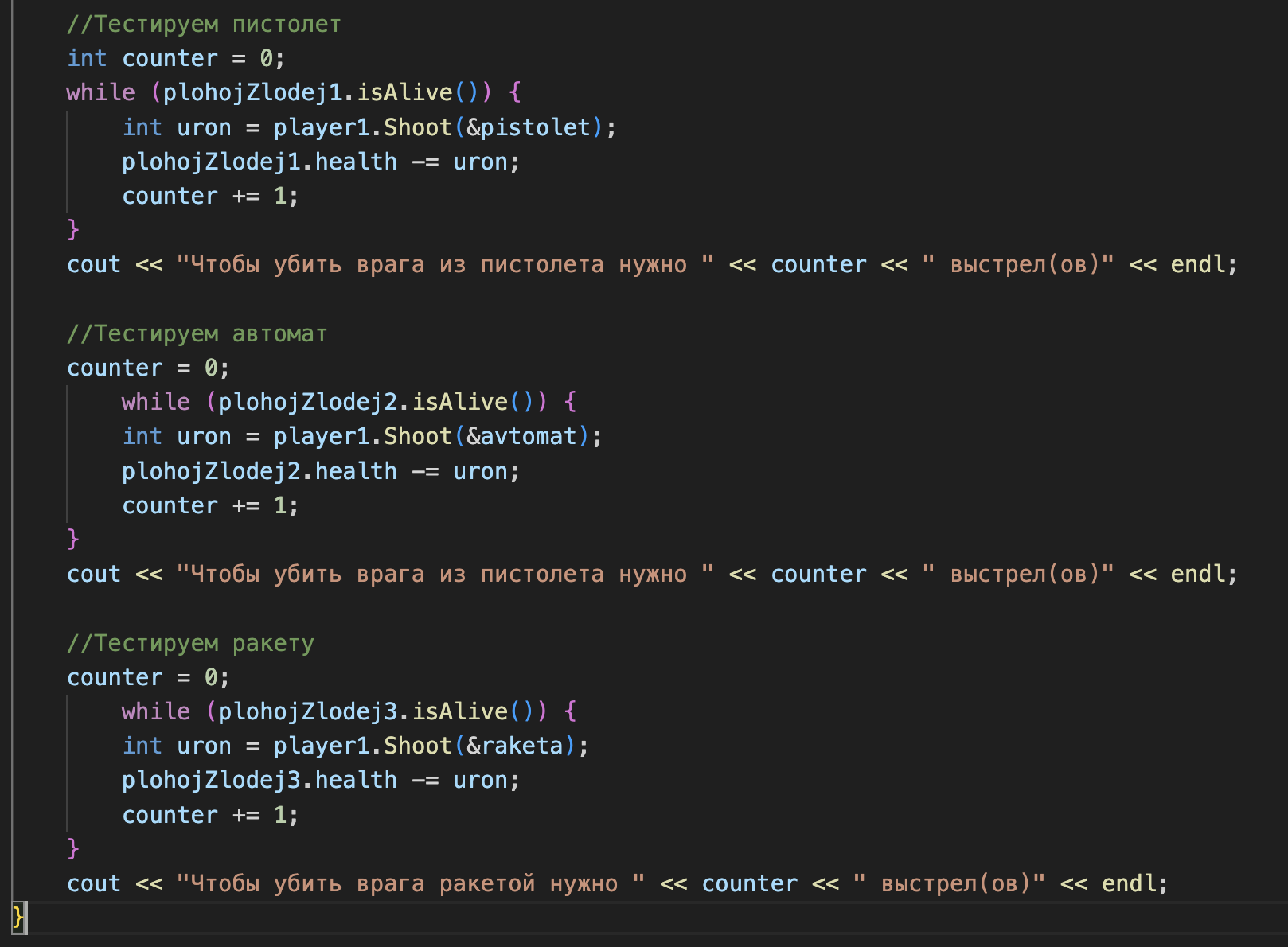
Теперь создадим классы игрока и врага:



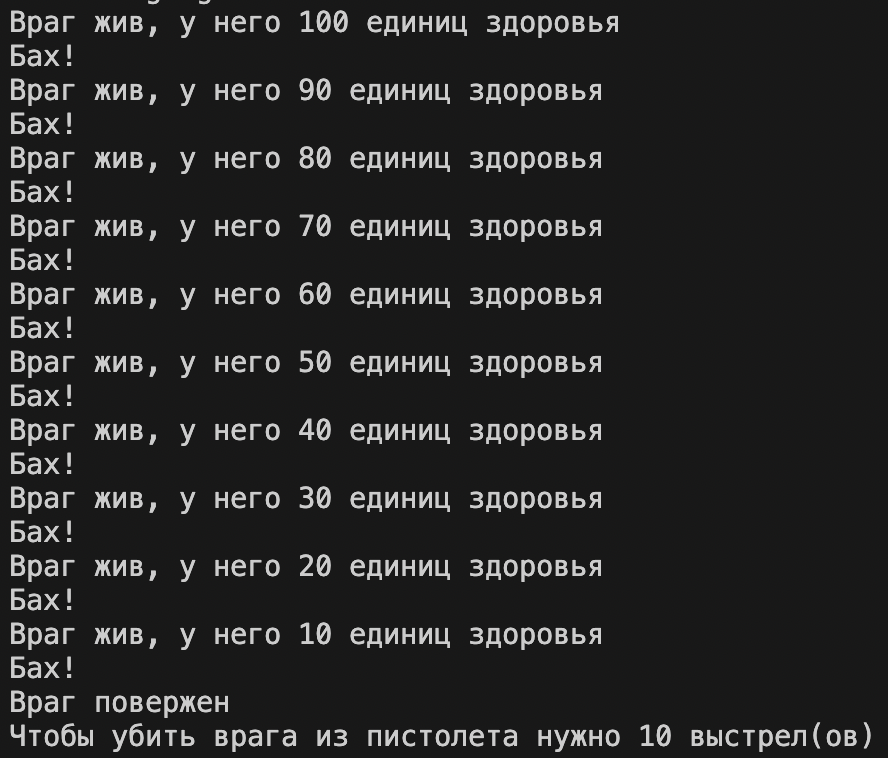


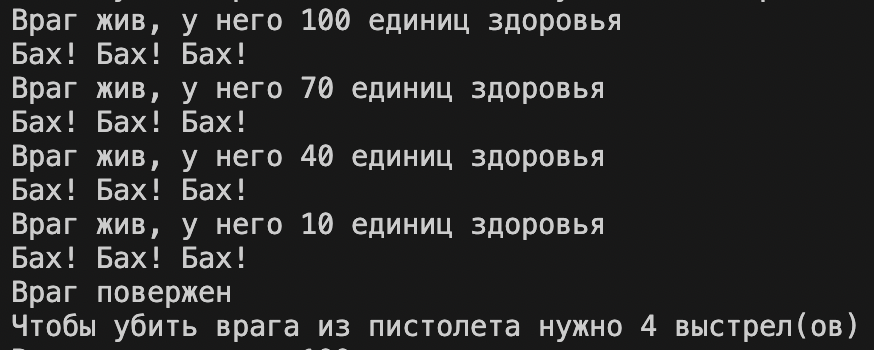
Теперь в main мы можем «протестировать» наше оружие:

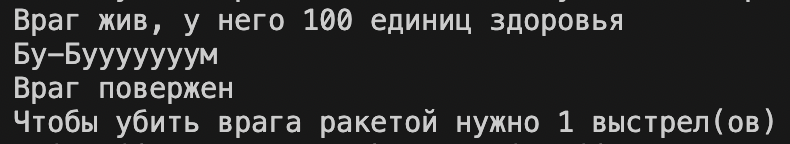




И в терминале видим:







Обращу внимание на то, что при определении класса Player, мы указали, что в функцию Shoot мы будем передавать указатель на класс типа Gun, однако так мы использовали принцип полиморфизма, мы можем передавать ему и другие типы.

Таким образом Virtual и override нужны для того, чтобы мы могли, *используя указатель на родительский класс, менять поведение его классов-наследников.*