Обща информация за проекта

Проектите се оценяват по редица от критерии, част от които са описани по-долу.

Тъй като курсът се фокусира върху обектно-ориентираното програмиране и неговата реализация в езика C++, най-важното изискване за проектите е те да са изградени съгласно добрите принципи на ООП. Решението, в което кодът е процедурен, има лоша ООП архитектура и т.н. се оценява с нула точки. Други важни критерии за оценка на проектите са:

* Дали решението работи коректно, съгласно спецификацията. Решение, което не работи и/или не се компилира носи минимален брой (или нула) точки.
* Дали решението отговаря на заданието на проекта.
* Каква част от необходимата функционалност е била реализирана.
* Дали решението е изградено съгласно добрите практики на обектно-ориентирания стил. Тъй като курсът се фокусира върху ООП, решения, които не са обектно-ориентирани се оценяват с нула или минимален брой точки.
* Оформление на решението. Проверява се дали кодът е добре оформен, дали е спазена конвенция за именуване на променливите, дали е добре коментиран и т.н.
* Дали решението е било добре тествано. Проверява се какви тестове са били проведени върху приложението, за да се провери дали то работи коректно. Очаква се по време на защитата да можете да посочите как сте тествали приложението, за да проверите дали то работи коректно и как се държи в различни ситуации.

По време на защитата се очаква да можете да отговорите на различни въпроси, като например: (1) каква архитектура сте избрали, (2) защо сте избрали именно нея, (3) дали сте обмислили други варианти и ако да — кои, (4) как точно работят различните части от вашия код и какво се случва на по-ниско ниво и др.

Оценката на всеки от проектите се формира от онази негова част, която е била самостоятелно разработена от вас. Допустимо е да използвате код написан от някой друг (напр. готова библиотека или помощ от ваш приятел/колега), но (1) той не носи точки към проекта и (2) това трябва да бъде ясно обявено както при предаването, така и при защитата на проекта, като ясно обозначите коя част от проекта сте разработили самостоятелно. Това означава, че:

1. Използваният наготово код трябва да се маркира ясно, като поставите коментари на подходящи места в кода си.
2. По време на защитата трябва да посочите кои части сте разработили самостоятелно и кои са взети от други източници.

Както е написано по-горе, когато в проекта си използвате чужд код, сам по себе си той не ви носи точки. Допълнителни точки могат да се дадат или отнемат, според (1) способността ви за внедряване на кода във вашето решение (напр. в случаите, когато се използва външна библиотека) и за това (2) дали добре разбирате какво прави той.

Проект “Чудовища - Бойци”

Постройте подходяща йерархия от класове, чрез която да може да се имплементира компютърна симулация на гладиаторски двубои между “Чудовища - Бойци”. Двубоите са организирани в турнири, а те се провеждат на подходящи за целта места наречени “Арени”.  Организацията на самите турнири може да е различна - директна елиминация, всеки-срещу-всеки и т.н. Някои турнири позволяват възстановителни сесии между двубоите, в които бойците да възвърнат силите си - в някой пък оцеляват само най-издръжливите. По време на двубоите на някои от бойците е възможно да бъдат раздадени щитове и оръжия от организаторите. Бойците умеят да боравят различно добре с тях и не винаги успяват да ги използват по предназначение. Самите щитове и оръжия също имат лимит на своята здравина, след което стават негодни за използване.

Изисквания към вашата програма

* Имплементирайте всички нужни класове с подходящи релации между тях, както и техните методи
* Трябва да имате най-малко три различни вида “Бойци”, които се държат различно по-време на битка. Примерни идеи:
  + Вампир - когато удря живи същества възстановява част от кръвта си
  + Дракон - умее да лети, което понякога му дава възможност да избегне удар. Умее да бълва огън нанасяйки допълнителни щети. Не умее да борави с оръжия и щитове
  + Зомби - остава боеспособно, дори когато кръвта му е под 0
  + Рицар - умее да борави безотказно с оръжия и щитове и винаги ги използва по предназначение.
* Турнирите трябва да поддържат най-малко три различни вида организация. Примерни идеи:
  + Директна елиминация
  + Всеки срещу всеки
  + Смесен (групи, в които всеки играе срещу всеки, победителите от групите играят директни елиминации)
* Турнирите трябва да поддържат пълна история на двубоите състояли се в тях
* Във всеки един момент от изпълнението на турнира трябва да е възможно то да бъде прекратено, нужната за него информация да може бъде сериализирана във файлов формат по ваш избор и в последствие да може да бъде десериализирана, така че турнира да може да продължи.
* Класовете ви трябва адекватно да управляват паметта, която използват и да не я разхищават.
* Имплементирайте основна функция, която показва/тества всички реализирани функционалности