**Упражнение 1**

**Unit tests**

**Какво представлява unit (компонентно) тестване?**

* Unit - най-малката част от кода, която подлежи на тестване (пример: функция)
* Изолира всяка част(unit) от програмата, проверявайки дали е независимо коректен.
* <https://www.guru99.com/unit-testing-guide.html>
* [Software Testing Training](https://www.youtube.com/watch?v=TDynSmrzpXw&list=PLDC2A0C8D2EC934C7)

**Какво наричаме unit?**

* В процедурното програмиране - програма, функция
* В обектно-ориентираното програмиране - методът.

**С какво ни помага unit тестването?**

* Благодарение на “компонентното” тестване се изграждат по - *сигурни* и *надеждни* програми. Когато първо пишем тестовете, знаем какви са случаите, които трябва да се покрият от нашата програма. (test-driven-development)
* За да може лесно да тестваме кода си, той става по-компактен(модулярен). По този начин, кода става по-лесен за *преизползване*.
* *Спестяваме време* - когато първо напишем тестовете, ние предварително сме помислили какви “конфликтни” ситуации, може да възникнат по време на изпълнение на нашата програма. Времето, което ще прекараме в дебъгване на нашата програма ще се намали.
* *Чист и качествен код.*

**Методи на тестване**

* White-box-testing
* Black-box-testing

<https://en.wikipedia.org/wiki/Mock_object>

<https://docs.microsoft.com/en-us/visualstudio/test/writing-unit-tests-for-c-cpp>

***Пример:*** Тестване на функция за сортиране.

* Какви възможни ситуации може да възникнат по време на изпълнение?
* Как да се справим с тях?

Advanced

**Integration testing**

* Интеграция - Обединяване в едно цяло
* Проверяваме как функциите си взаимодействат
* <http://softwaretestingfundamentals.com/integration-testing>

**Extreme programming**

* методология, подходяща да се използва при проекти, които имат често променящи се изисквания
* [**https://en.wikipedia.org/wiki/Extreme\_programming**](https://en.wikipedia.org/wiki/Extreme_programming)