QA Automation Engineer

[Задания для самостоятельного выполнения 2](#_Toc66717308)

[Задача 1 3](#_Toc66717309)

[Пример 3](#_Toc66717310)

[Задача 2 4](#_Toc66717311)

[Пример 4](#_Toc66717312)

[Задача 3 5](#_Toc66717313)

[Тест-кейс 1: Список файлов 5](#_Toc66717314)

[Тест-кейс 2: Случайный файл 5](#_Toc66717315)

# Задания для самостоятельного выполнения

Выберите одну любую задачу из представленных ниже. Решите задачу, написав код на одном из следующих языков:

* Python
* C/C++
* C#

Рекомендуется там, где это уместно, использовать стандартные библиотеки, а не выполнять реализацию общеизвестных алгоритмов с нуля.

## Задача 1

Реализовать программу, осуществляющую копирование файлов в соответствии с конфигурационным файлом. Конфигурационный файл должен иметь формат xml. Для каждого файла в конфигурационном файле должно быть указано его имя, исходный путь и путь, по которому файл требуется скопировать.

### Пример

Конфигурационный файл:

<config>  
 <file  
 source\_path="C:\Windows\system32"  
 destination\_path="C:\Program files"  
 file\_name="kernel32.dll"  
 />  
 <file  
 source\_path="/var/log"  
 destination\_path="/etc"  
 file\_name="server.log"  
 />  
</config>

## Задача 2

Дан файл, содержащий имена файлов, алгоритм хэширования (один из MD5/SHA1/SHA256) и соответствующие им хэш-суммы, вычисленные по соответствующему алгоритму и указанные в файле через пробел. Напишите программу, читающую данный файл и проверяющую целостность файлов.

### Пример

Файл сумм:

file\_01.bin md5 aaeab83fcc93cd3ab003fa8bfd8d8906

file\_02.bin md5 6dc2d05c8374293fe20bc4c22c236e2e

file\_03.bin md5 6dc2d05c8374293fe20bc4c22c236e2e

file\_04.txt sha1 da39a3ee5e6b4b0d3255bfef95601890afd80709

Пример вызова:

<your program> <path to the input file> <path to the directory containing the files to check>

Формат вывода:

file\_01.bin OK

file\_02.bin FAIL

file\_03.bin NOT FOUND

file\_04.txt OK

## Задача 3

Напишите прототип тестовой системы, состоящей из двух тест-кейсов. В данной задаче использование стороннего модуля для автоматизации тестирования не приветствуется.

Тестовая система представляет собой иерархию классов, описывающую тест-кейсы.

У каждого тест-кейса есть:

* Номер (tc\_id) и название (name)
* Методы для подготовки (prep), выполнения (run) и завершения (clean\_up) тестов.
* Метод execute, который задаёт общий порядок выполнения тест-кейса и обрабатывает исключительные ситуации.

Все этапы выполнения тест-кейса, а также исключительные ситуации должны быть задокументированы в лог-файле или в стандартном выводе.

### Тест-кейс 1: Список файлов

* **[prep]** Если текущее системное время, заданное как целое количество секунд от начала эпохи Unix, не кратно двум, то необходимо прервать выполнение тест-кейса.
* **[run]** Вывести список файлов из домашней директории текущего пользователя.
* **[clean\_up]** Действий не требуется.

### Тест-кейс 2: Случайный файл

* **[prep]** Если объем оперативной памяти машины, на которой исполняется тест, меньше одного гигабайта, то необходимо прервать выполнение тест-кейса.
* **[run]** Создать файл test размером 1024 КБ со случайным содержимым.
* **[clean\_up]** Удалить файл test.