МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра математического моделирования и анализа данных

ОТЧЁТ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ №2

по дисциплине специализации «Тестирование случайных и псевдослучайных последовательностей»

Преподаватель**:**
Вечерко Егор Валентинович
канд. физ.-мат. наук, доцент

Выполнили студенты:
Болтач Антон Юрьевич
Тимошенко Станислав Вячеславович
Лагунов Антон Сергеевич
Лисай Вадим Олегович
3 курс, 9 группа

Минск 2019

Основная задача заключалась в том, чтобы создать оболочку для различных реализаций статистических тестов NIST, целью которых является определение меры случайности двоичных последовательностей. Также необходимо было создать пользовательский интерфейс непосредственно для работы с тестами.

Было это реализовано путём создания абстрактных классов для каждого теста, а также для уже конкретной его реализации отдельным человеком.

Содержание проекта

Проект содержит две основные папки: src, seq.

seq: содержит двоичные последовательности.

src:

- содержит именные папки, в каждой находится файлы .cpp с реализацией тестов;
- папка libraries, содержащая следующие заголовочные файлы:
  {test_name}.hpp – определяет абстрактный класс. Пример: serial_test.hpp
  {test_name}_{surname}.hpp – определяет дочерний класс с реализацией теста. Пример: serial_test_zakrevsky.hpp
  tests.hpp – включает заголовочные файлы реализаций теста.
  cephes.h – содержит специальные математические функции, которые используются в некоторых реализациях.
  interface.hpp – заголовочный файл для класса интерфейса.
- utils содержит cephes.cpp.
- interface.cpp – реализация интерфейса.

CMakeLists.txt: хранит правила и цели сборки.

Проект является кроссплатформенным. Для его сборки необходим CMake. CMake – кроссплатформенная система автоматизации сборки программного обеспечения из исходного кода.

Как собрать проект:

Windows:

Если CMake не установлен, сделать следующее:

- Скачать cmake-3.14.4-win64-x64.zip с сайта https://cmake.org/download/. И разархивировать его в удобное место.

- Добавить в системных переменных в Path следующее: {file_path}\cmake-3.14.4-win64-x64\bin

  Пример: C:\cmake-3.14.4-win64-x64\bin

Сборка:

1. Создать в папке с проектом папку build: **`mkdir build`**
2. Перейти в неё: **`cd build`**

Выполнить следующие команды:

- **`cmake -G "MinGW Makefiles" ..`**
- **`cmake –D "CMAKE_MAKE_PROGRAM:PATH=C:\MinGW\bin\make.exe" ..`**
- **`cmake --build . -- -j3`**
- **`testsSequence.exe`**

Linux

Если CMake не установлен, сделать следующее:

- Скачать cmake-3.14.4-Linux-x86_64.sh с сайта https://cmake.org/download/ .
- Переместить файл .sh в /opt.
  `mv cmake-3.14.4-Linux-x86_64.sh /opt/`
- С помощью команды chmod +x сделать файл .sh выполняемым:
  `chmod +x /opt/cmake-3.14.4-Linux-x86_64.sh`
- Запустить скрипт установки:
  `sudo bash /opt/cmake-314.4-Linux-x86_64.sh`
- В результате cmake устанавливается в дирректорию **`/opt/cmake-3.14.4-Linux-x86_64`**
- Добавляем символическую ссылку следующей командой:
  `sudo ln -s /opt/cmake-3.14.4-Linux-x86_64/bin/* /usr/local/bin`
- Для тестирования выполним команду:
  `cmake –version`

Сборка:

1. Создать в папке с проектом папку build: **`mkdir build`**
2. Перейти в неё: **`cd build`**

Выполнить следующие команды:

- **`cmake ..`**
- **`cmake --build . -- -j3`**
- **`./testsSequence.out`**

MacOS:

Аналогично, как и в Linux.

Добавление новой реализации

Чтобы добавить реализацию теста, необходимо:

- Создать заголовочные файлы .hpp:
  {test_name}.hpp с определением абстрактного класса. В случае, если он ещё не существует.
  {test_name}_{surname}.hpp с дочерним классом уже конкретной реализации.
- По аналогии создать исходный файл .cpp в соответствующей папке с именем и реализовать класс, который содержит два основных метода: read(std::string filename = "") для чтения последовательности из файла и run_test() для запуска теста. Подключить заголовочный файл {test_name}_{surname}.hpp.
- Добавить {test_name}_{surname}.hpp в tests.hpp
- Указать в CMakeLists.txt путь к добавленному исходному файлу.
- В interface.hpp определить метод с названием теста. И реализовать его в interface.cpp по аналогии с уже существующими.