**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**"ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"**

Лабораторная работа № 7

Тема: «Автоматизация модульного тестирования программ инструментальными средствами семейства xUnit»

Проверил: Выполнил:

проф. каф. ПИ ст. гр. ПИ-18а

Федяев О.И. Данильчук К. М.

\_\_\_\_.\_\_\_\_.2021г. \_\_\_\_.\_\_\_\_.2021г.

Донецк – 2021

**Цель работы:** изучить методику и освоить инструментальные средства семейства xUnit для программного управления условиями тестов и параметрами обработки их результатов; научиться использовать основные показатели и индикаторы качества кода для контроля правильности программ на более ранних этапах разработки.

**Тестовое окружение: Google C++ Testing Framework (Google Test)** — библиотека для модульного тестирования (англ. unit testing) на языке С++. Исходные тексты открыты с середины 2008 года[1] под лицензией BSD. Документация частично переведена на русский язык.

Google Test построена на методологии тестирования xUnit, то есть когда отдельные части программы (классы, функции, модули) проверяются отдельно друг от друга, в изоляции. Библиотека сама по себе разработана с активным применением тестирования, когда при добавлении каких-либо частей в официальную версию, кроме кода самих изменений необходимо написать набор тестов, подтверждающих их корректность.

**Способ запуска тестов в инструментальной среде.**

Как уже было сказано ранее, для запуска тестов использовался GTest, который позволяет тестировать модули в изоляции от основного приложения.

Модуль находится в проекте в следующей структуре (см. рис. 1).

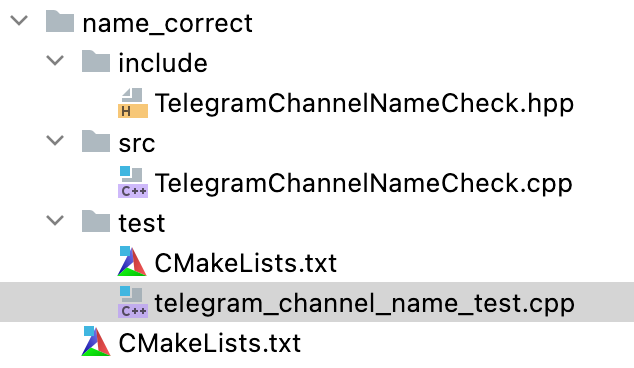


Рисунок 1 – Структура модуля

Код TelegramChannelNameCheck.hpp

*//  
// Created by Kirill Danilchuk <kirill.danilchuk01@gmail.com> on 11/12/2021.  
//*#pragma once  
  
#include <string\_view>

#include <string>  
  
constexpr bool IsCorrectChannelName(std::string\_view channel\_name) {  
 if (channel\_name.front() == '@') {  
 channel\_name.remove\_prefix(1);  
 } else {  
 return false;  
 }  
  
 if (!(1 < channel\_name.size() && channel\_name.size() < 25)) {  
 return false;  
 }  
  
 for (auto&& symbol : channel\_name) {  
 if (!(IsEnglishSymbol(symbol) || isdigit(symbol)  
 || IsUnderscore(symbol))) {  
 return false;  
 }  
 }  
  
 return true;  
}

Код TelegramChannelNameCheck.cpp

Пустой

test/CMakeLists.txt

create\_tests(name\_correct)

name\_correct/CMakeLists.txt

create\_static\_library(name\_correct)

cmake/create\_tests.cmake

macro(create\_tests MODULE)  
 cmake\_minimum\_required(VERSION 3.14)  
 project(${MODULE}\_tests)  
 file(GLOB\_RECURSE SOURCE\_FILES  
 ${CMAKE\_CURRENT\_SOURCE\_DIR}/\*.cpp)  
  
 add\_executable(${MODULE}\_tests ${SOURCE\_FILES})  
  
 *# Testing module* target\_link\_libraries(${MODULE}\_tests PRIVATE ${MODULE})  
  
 *# GTest* target\_link\_libraries(${MODULE}\_tests PRIVATE gtest gtest\_main)  
endmacro()

В коде выше можно увидеть, что main функция для запуска тестов не пишется пользователем, а подключается виде библиотеки gtest\_main.

cmake/create\_static\_library.cmake

macro(create\_static\_library MODULE)  
 cmake\_minimum\_required(VERSION 3.14)  
 project(${MODULE})  
 set(CMAKE\_CXX\_STANDARD 20)  
 file(GLOB\_RECURSE SOURCE\_FILES ${CMAKE\_CURRENT\_SOURCE\_DIR}/src/\*.cpp)  
 add\_library(${MODULE} STATIC ${SOURCE\_FILES})  
 target\_include\_directories(${MODULE} PUBLIC ${CMAKE\_CURRENT\_SOURCE\_DIR}/include)  
 add\_subdirectory(test)  
endmacro()

Метод получения GTest из репозитория (данный код в third-party/CMakeLists.txt).

*# GTest*FetchContent\_Declare(  
 googletest  
 GIT\_REPOSITORY https://github.com/google/googletest.git  
 GIT\_TAG release-1.10.0  
)  
FetchContent\_MakeAvailable(googletest)

Запуск тестов производится по нажатию кнопки запуска тестов (см. рис. 2).

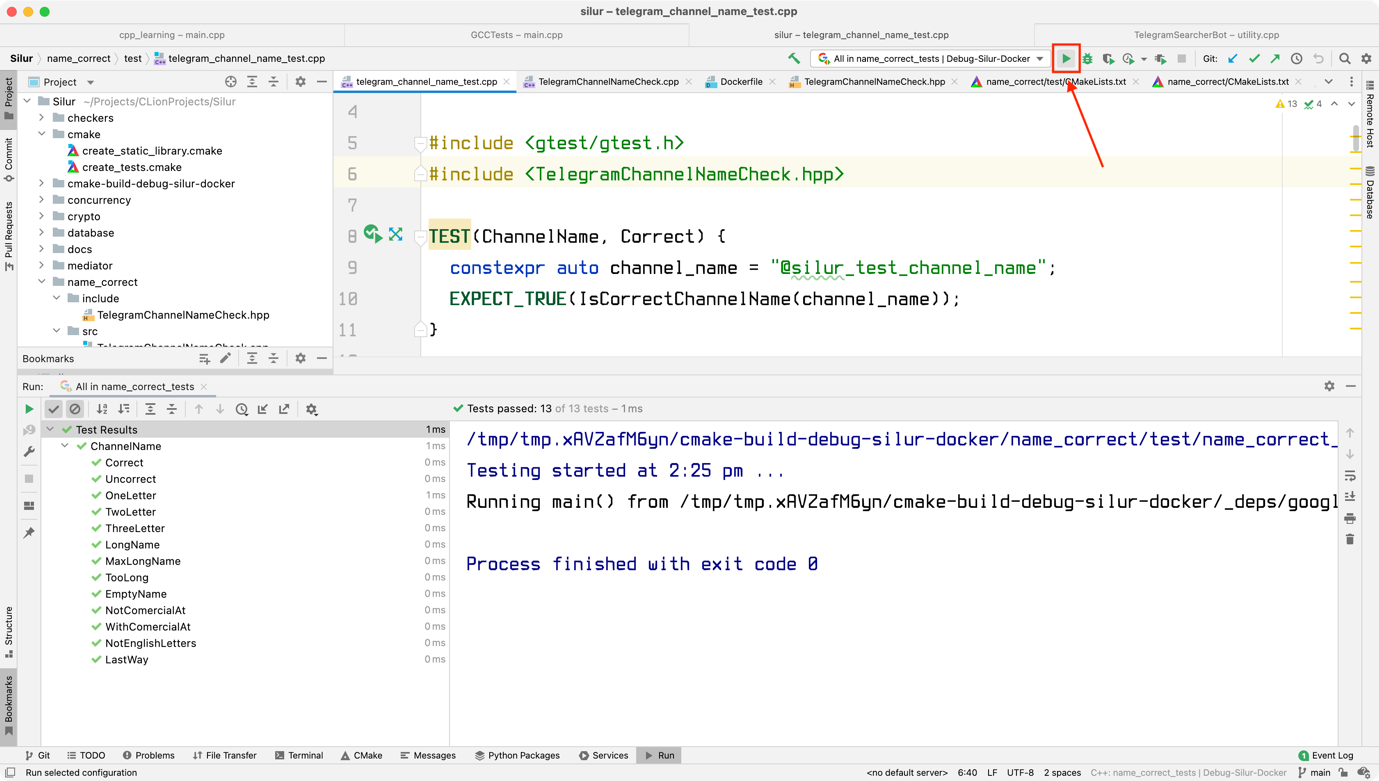


Рисунок 2 – Запуск тестов

Инструкция выше является полной для запуска тестов для модуля.

Данный модуль оказывает огромную ценность в обработки ошибок пользователя (см. рис. 3).

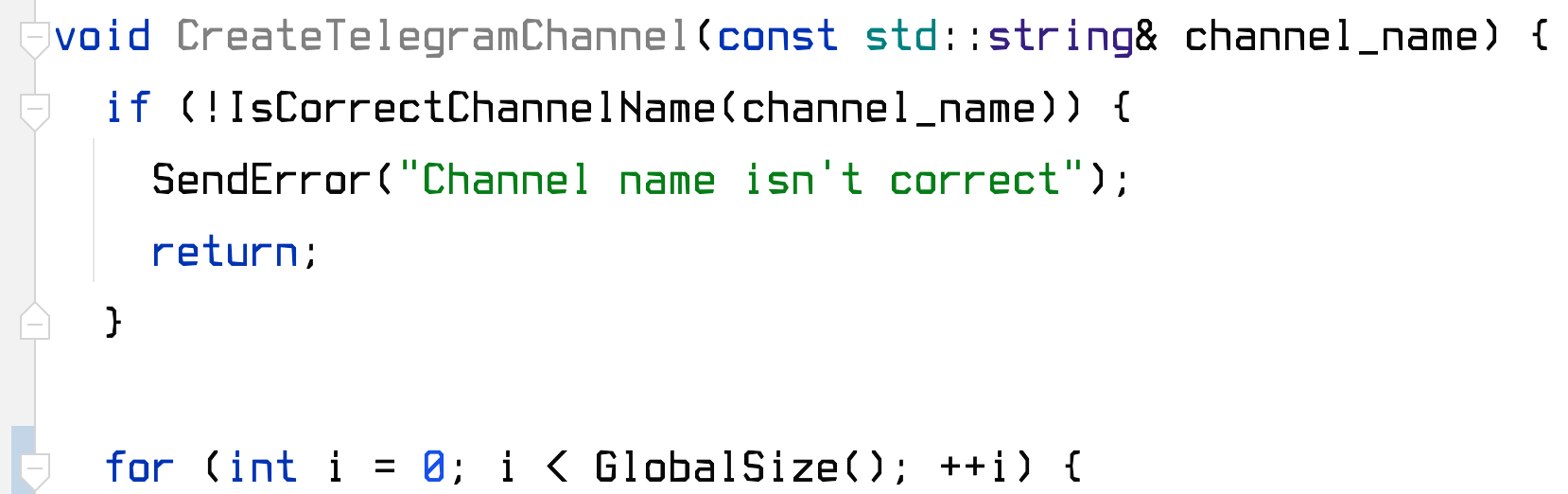


Рисунок 3 – Использование функции модуля в реальном проекте

При получении названия сервер проверяет его корректность и в случае возникновения ошибки сообщает об этом клиенту.

Тест-сценарии с тестовыми примерами на естественном языке

Для правильных классов эквивалентностей

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Входное значение | Выходное значения |
| 1 | Название в диапазоне от 2 до 25 | «@silur\_test\_channel\_name» | true |
| 2 | Первый символ @ | «@silur\_test» | true |
| 3 | Содержит только английские буквы | «@english» | true |
| 4 | Содержит только цифры | «@1231234» | true |
| 5 | Содержит только нижние подчеркивание | «@\_\_» | true |
| 6 | Содержит английские буквы и цифры | «@number123» | true |
| 7 | Содержит английские буквы и нижние подчеркивания | «@line\_\_x» | true |
| 8 | Содержит цифры и нижние подчеркивания | «@123123\_\_123123» | true |
| 9 | Содержит все допустимые символы (английские буквы, цифры и нижние подчеркивания) | «@test\_123» | true |

Тесты неправильных классов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Входное значение | Выходное значения |
| 10 | Содержит один символ в названии | «@s» | false |
| 11 | Содержит больше 25 символов в названии | «@abcdefghijklmnopqrstuvwxyz» | false |
| 12 | Не содержит первым символом @ | «no» | false |
| 13 | Содержит недопустимые символы | «@ыфыва» | false |

Метод анализа граничных условий

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Входное значение | Выходное значения |
| 14 | Строка состоит из символа @ и английской буквы | Покрывается тестом 10 |  |
| 15 | Строка состоит из символа @ и 2х английских букв | «@ab» | true |
| 16 | Строка состоит из символа @ и 3х английских букв | «@abc» | true |
| 17 | Строка состоит из символа @ и 24х английских букв | «@abcdefghijklmnopqrstuvwx» | true |
| 18 | Строка состоит из символа @ и 25 английских букв | «@abcdefghijklmnopqrstuvwxy». | true |
| 19 | Строка состоит из символа @ и 26х английских букв | Покрывается тестом 3 |  |
| 20 | Первый символ строки @. | Покрывается тестом 2 |  |
| 21 | Первый символ строки не @. | Покрывается тестом 12 |  |
| 22 | На вход подается пустая строка | «» | false |

Метод покрытия операторов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Входное значение | Выходное значения |
| 23 | Покрытие пути a | «test\_channel» | false |
| 24 | Покрытие пути b, c | «@test\_channel» | true |
| 25 | Покрытие пути b, e | «@ыъ» | false |

Метод покрытия переходов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Входное значение | Выходное значения |
| 26 | Покрытие пути b, d | «@ab» | true |

**Листинг тестов**

*//  
// Created by Kirill Danilchuk <kirill.danilchuk01@gmail.com> on 06/12/2021.  
//*#include <gtest/gtest.h>  
#include <TelegramChannelNameCheck.hpp>  
  
*// 1***TEST**(ChannelName, Correct1) {  
 constexpr auto channel\_name = "@silur\_test\_channel\_name";  
 **EXPECT\_TRUE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 2***TEST**(ChannelName, WithComercialAt) {  
 constexpr auto channel\_name = "@silur\_test";  
 **EXPECT\_TRUE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 3***TEST**(ChannelName, OnlyEnglishLetters) {  
 constexpr auto channel\_name = "@english";  
 **EXPECT\_TRUE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 4***TEST**(ChannelName, OnlyNumbers) {  
 constexpr auto channel\_name = "@1231234";  
 **EXPECT\_TRUE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 5***TEST**(ChannelName, OnlyLines) {  
 constexpr auto channel\_name = "@\_\_";  
 **EXPECT\_TRUE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 6***TEST**(ChannelName, EnglishLettersAndNumbers) {  
 constexpr auto channel\_name = "@number123";  
 **EXPECT\_TRUE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 7***TEST**(ChannelName, EnglishLettersAndLines) {  
 constexpr auto channel\_name = "@line\_\_x";  
 **EXPECT\_TRUE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 8***TEST**(ChannelName, NumbersAndLines) {  
 constexpr auto channel\_name = "@123123\_\_123123";  
 **EXPECT\_TRUE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 9***TEST**(ChannelName, AllSymbols) {  
 constexpr auto channel\_name = "@test\_123";  
 **EXPECT\_TRUE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 10***TEST**(ChannelName, OneLetter) {  
 constexpr auto channel\_name = "@ы";  
 **EXPECT\_FALSE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 11***TEST**(ChannelName, TooLong) {  
 constexpr auto channel\_name = "@abcdefghijklmnopqrstuvwxyz";  
 **EXPECT\_FALSE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 12***TEST**(ChannelName, NotComercialAt) {  
 constexpr auto channel\_name = "test\_channel";  
 **EXPECT\_FALSE**(IsCorrectChannelName(channel\_name));  
}  
  
  
*// 13***TEST**(ChannelName, Uncorrect) {  
 constexpr auto channel\_name = "@ыфыва";  
 **EXPECT\_FALSE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 15***TEST**(ChannelName, TwoLetter) {  
 constexpr auto channel\_name = "@ab";  
 **EXPECT\_TRUE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 16***TEST**(ChannelName, ThreeLetter) {  
 constexpr auto channel\_name = "@abc";  
 **EXPECT\_TRUE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 17***TEST**(ChannelName, LongName) {  
 constexpr auto channel\_name = "@abcdefghijklmnopqrstuvwx";  
 **EXPECT\_TRUE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 18***TEST**(ChannelName, MaxLongName) {  
 constexpr auto channel\_name = "@abcdefghijklmnopqrstuvwxy";  
 **EXPECT\_FALSE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 22***TEST**(ChannelName, EmptyName) {  
 constexpr auto channel\_name = "";  
 **EXPECT\_FALSE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 23***TEST**(ChannelName, WayA) {  
 constexpr auto channel\_name = "test\_channel";  
 **EXPECT\_FALSE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 24***TEST**(ChannelName, WayBc) {  
 constexpr auto channel\_name = "@test\_channel";  
 **EXPECT\_TRUE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 25***TEST**(ChannelName, WayBE) {  
 constexpr auto channel\_name = "@ыъ";  
 **EXPECT\_FALSE**(IsCorrectChannelName(channel\_name));  
}  
  
*// 26***TEST**(ChannelName, WayBD) {  
 constexpr auto channel\_name = "@ab";  
 **EXPECT\_TRUE**(IsCorrectChannelName(channel\_name));  
}

Результаты тестирования

