МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт КНТ

Факультет ФИСП

Кафедра ПИ им. Л.П. Фельдмана

Лабораторная работа №7

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

Курс: Качество программного обеспечения и тестирование

Выполнил

ст. гр. ПИ-18Б

Моргунов А.Г.

Проверил

Федяев О. И.

Ищенко А. П.

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.

Код UserCheck.cpp

//

// Created by Arsenii on 22.12.2021.

//

#include "UserCheck.h"

USER\_RIGHTS CheckUserState(User user) {

switch(user.role){

case User::ROLE::USER:

if (user.rent > 0) {

return USER\_RIGHTS::USER\_RIGHTS;

} else if (user.rent == 0){

return USER\_RIGHTS::NO\_RENT\_RIGHTS;

} else {

return USER\_RIGHTS::NEGATIVE\_VALUE;

}

case User::ROLE::ADMIN:

return USER\_RIGHTS::ADMIN\_RIGHTS;

case User::ROLE::NONE:

return USER\_RIGHTS::NONE\_RIGHTS;

case User::ROLE::UNREGISTER:

return USER\_RIGHTS::UNREGISTER\_RIGHTS;

default:

return USER\_RIGHTS::UNEXPECTED;

}

}

Код UserCheck.hpp

//

// Created by Arsenii on 22.12.2021.

//

#ifndef MODULEFORTEST\_\_FUNCTION\_H\_

#define MODULEFORTEST\_\_FUNCTION\_H\_

#endif //MODULEFORTEST\_\_FUNCTION\_H\_

class User {

public:

enum ROLE{

UNREGISTER, // Sent registration request

NONE, // No in database

ADMIN,

USER,

UNEXPECTED\_ROLE

};

ROLE role;

int rent;

User(ROLE role, int rent = 0){

this->role = role;

this->rent = rent;

}

};

enum class USER\_RIGHTS{

USER\_RIGHTS,

ADMIN\_RIGHTS,

NONE\_RIGHTS,

NO\_RENT\_RIGHTS,

UNREGISTER\_RIGHTS,

UNEXPECTED,

NEGATIVE\_VALUE

};

USER\_RIGHTS CheckUserState(User user);

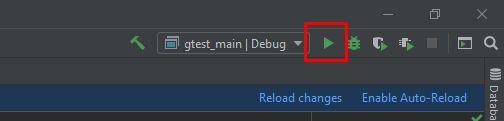
Метод получения GTest из репозитория (данный код в CMakeLists.txt).

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

add\_executable(ModuleForTest main.cpp UserCheck.cpp UserCheck.h)

target\_link\_libraries(${PROJECT\_NAME} PRIVATE gtest gtest\_main)

Запуск тестов производится по нажатию кнопки запуска тестов.



Тест-сценарии с тестовыми примерами на естественном языке

Для правильных классов эквивалентностей

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Входное значение | Выходное значения |
| 1 | Зарегистрированный пользователь с активной подпиской | Роль: USER  Дней до конца подписки: 23 | USER\_RIGHTS |
| 2 | Зарегистрированный пользователь с неактивной подпиской | Роль: USER  Дней до конца подписки: 0 | NO\_RENT\_RIGHTS |
| 3 | Администратор | Роль: ADMIN  Дней до конца подписки: 0 | ADMIN\_RIGHTS |
| 4 | Незарегистрированный пользователь | Роль: UNREGISTER  Дней до конца подписки: 0 | UNREGISTER\_RIGHTS |
| 5 | Незарегистрированный пользователь, ожидающий ответа на регистрационную заявку | Роль: NONE  Дней до конца подписки: 0 | NONE\_RIGHTS |

Тесты неправильных классов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Входное значение | Выходное значения |
| 6 | Несуществующая роль пользователя | Роль: UNEXPECTED\_ROLE  Дней до конца подписки: 0 | UNEXPECTED |
| 7 | Отрицательное значение дней | Роль: USER  Дней до конца подписки: -23 | NEGATIVE\_VALUE |

Метод анализа граничных условий

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Входное значение | Выходное значения |
| 8 | Зарегистрированный пользователь с активной подпиской | Роль: UNEXPECTED\_ROLE  Дней до конца подписки: 0 | UNEXPECTED |
| 9 | Зарегистрированный пользователь с неактивной подпиской | Покрывается тестом 2 |  |
| 10 | Отрицательное значение дней | Роль: USER  Дней до конца подписки: -1 | NEGATIVE\_VALUE |
| 11 | Несуществующая роль пользователя | Покрывается тестом 6 |  |

Метод покрытия операторов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Входное значение | Выходное значения |
| 12 | Покрытие пути a | Покрывается тестом 1 |  |
| 13 | Покрытие пути b | Покрывается тестом 2 |  |
| 14 | Покрытие пути c | Покрывается тестом 8 |  |
| 15 | Покрытие пути d | Покрывается тестом 3 |  |
| 16 | Покрытие пути e | Покрывается тестом 5 |  |
| 17 | Покрытие пути f | Покрывается тестом 4 |  |
| 18 | Покрытие пути g | Покрывается тестом 6 |  |

**Листинг тестов**

#include <iostream>

#include "gtest/gtest.h"

#include "function.h"

TEST(correct, test\_1) {

User user(User::ROLE::USER, 23);

auto result = CheckUserState(user);

ASSERT\_EQ(result, USER\_RIGHTS::USER\_RIGHTS);

}

TEST(correct, test\_2) {

User user(User::ROLE::USER, 0);

auto result = CheckUserState(user);

ASSERT\_EQ(result, USER\_RIGHTS::NO\_RENT\_RIGHTS);

}

TEST(correct, test\_3) {

User user(User::ROLE::ADMIN, 0);

auto result = CheckUserState(user);

ASSERT\_EQ(result, USER\_RIGHTS::ADMIN\_RIGHTS);

}

TEST(correct, test\_4) {

User user(User::ROLE::UNREGISTER, 0);

auto result = CheckUserState(user);

ASSERT\_EQ(result, USER\_RIGHTS::UNREGISTER\_RIGHTS);

}

TEST(correct, test\_5) {

User user(User::ROLE::NONE, 0);

auto result = CheckUserState(user);

ASSERT\_EQ(result, USER\_RIGHTS::NONE\_RIGHTS);

}

TEST(uncorrect, test\_6) {

User user(User::ROLE::UNEXPECTED\_ROLE, 0);

auto result = CheckUserState(user);

ASSERT\_EQ(result, USER\_RIGHTS::UNEXPECTED);

}

TEST(uncorrect, test\_7) {

User user(User::ROLE::USER, -23);

auto result = CheckUserState(user);

ASSERT\_EQ(result, USER\_RIGHTS::NEGATIVE\_VALUE);

}

TEST(borderline, test\_8) {

User user(User::ROLE::USER, 1);

auto result = CheckUserState(user);

ASSERT\_EQ(result, USER\_RIGHTS::USER\_RIGHTS);

}

TEST(borderline, test\_10) {

User user(User::ROLE::USER, -1);

auto result = CheckUserState(user);

ASSERT\_EQ(result, USER\_RIGHTS::NEGATIVE\_VALUE);

}

Результаты тестирования

