

**Министр науки и высшего образования Российской Федерации Федеральное  
государственное автономное образовательное учреждение высшего  
образования «Национальный исследовательский университет ИТМО»**

**Факультет информационных технологий и программирования  
Цифровая культура**

Лабораторная работа №3(new)

Выполнил студент группы № М3111

Акберов Рустам Ханкишиевич

Подпись:

A handwritten signature in black ink, consisting of a stylized, cursive letter 'A' followed by a series of loops and a final flourish, all enclosed within a large, irregular oval shape.

Проверил:

## Требования к выполнению лабораторной работы

1. На основе лабораторных работ по программированию - необходимо дополнить их функциональными тестами.
2. Тесты в виде собственных функций не принимаются. Необходимо использовать сторонние библиотеки, например Google test.

### Отчет

**Google test** - библиотека для модульного тестирования на языках C/C++.

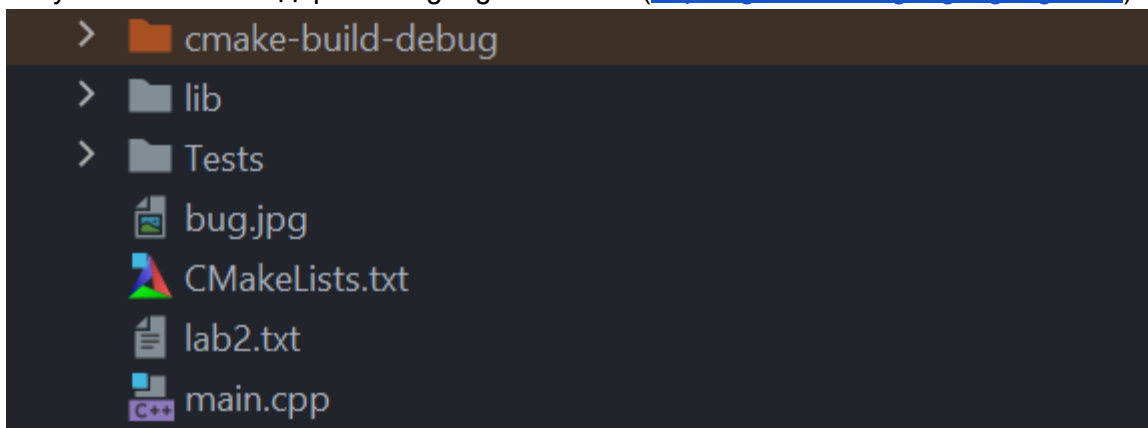
Документация частично переведена на русский язык. Google Test построена на методологии тестирования xUnit, то есть, когда отдельные части программы проверяются отдельно друг от друга, в изоляции.

Несколько принципов:

1. Минимальной единицей тестирования является одиночный тест.
2. Тесты не требуют отдельного регистрирования для запуска.
3. Каждый объявленный в программе тест автоматически будет запущен.
4. Тесты объединяются в группы (наборы).
5. Полное имя теста формируется из имени группы и собственного имени теста.
6. Тесты могут использовать тестовые классы, что позволяет создавать и повторно использовать одну и ту же конфигурацию объектов для нескольких различных тестов.
7. В состав библиотеки входит специальный скрипт, который упаковывает её исходные тексты всего в два файла: gtest-all.cc и gtest.h. Эти файлы могут быть включены в состав проекта без каких-либо дополнительных усилий по предварительной сборке библиотеки.

Рассмотрим основные этапы работы с Google Test:

1. Будем подключать к собранному проекту в CLion. ([Google Test | CLion](#))
2. Создадим все необходимые файлы для работы.
3. В папку lib поместим содержимое googletest-main (<https://github.com/google/googletest>)



4. Зайдем в CMakeLists.txt и изменим содержимое, добавив подключение директории lib к основному проекту.

```

cmake_minimum_required(VERSION 3.20)
project(LAB2 LANGUAGES C CXX)

set(CMAKE_C_STANDARD 11)

add_subdirectory(lib/googletest-main)
include_directories(lib/googletest-main/googletest/include)
include_directories(lib/googletest-main/googlemock/include)

add_executable(LAB2 Tests/ClassName.h Tests/test.cpp main.cpp)target_link_libraries(LAB2
gtest gtest_main)

```

5. В папке Tests будем создавать тесты в соответствии с синтаксисом gtest. Заголовочный файл ClassName.h и test.cpp - все, что нужно для правильной работы тестов.

```

#ifndef C_LAB2_CLASSNAME_H
#define C_LAB2_CLASSNAME_H

class ClassName{
public:
    typedef struct {
        uint64_t *values;
    }uint1024_t;
    int value;
    uint1024_t string_uint;
    static int get_length_uint64(int value){int cnt=0; while(value>0){value/=10; cnt+=1;}}
return cnt;
    int get_value() const {
        return value;
    }
    void set_value(int value) {
        ClassName::value = get_length_uint64(value);
    }
    uint1024_t from_string(const char * string, int base) {
        uint1024_t num;
        num.values = static_cast<uint64_t *>(malloc(base * sizeof(uint64_t)));
        for (int i = 0; i < base; i++) {num.values[i] = 0;}
        // if strlen/9 > base -> warning
        unsigned long str_size = strlen(string);
        int power = 0, lim = 3;
        for (unsigned int i = str_size; i > 0; i--){
            num.values[(str_size - i)/lim]+=(string[i-1]-'0')*pow(10,++power -1);
            if(power == lim) {power = 0;}
        }
        return num;
    }
}

```

```

uint1024_t get_value_uint_string() const {
    return string_uint;
}
void set_value_uint_string(const char * value, int base) {
    ClassName::string_uint = from_string(value, base);
}
};

#endif //C_LAB2_CLASSNAME_H

```

## 6. И файл с тестами test.cpp

```

//test.cpp

#include <gtest/gtest.h>
#include <gmock/gmock.h>
#include "ClassName.h"

using testing::Eq;
namespace {
    class ClassDeclaration : public testing::Test {
    public:
        ClassName obj;
        ClassDeclaration(){
            obj;
        }
    };
}
TEST_F(ClassDeclaration, Test_lab21){
    obj.set_value(373777283);
    ASSERT_EQ(9, obj.get_value());
    obj.set_value(31111211);
    ASSERT_THAT(8, testing::Eq(obj.get_value()));
    ASSERT_EQ("", "");
}
TEST_F(ClassDeclaration, Test_lab23){
    obj.set_value(373777283);
    ASSERT_EQ(9, obj.get_value());
    obj.set_value(31111211);
    ASSERT_THAT(8, testing::Eq(obj.get_value()));
    ASSERT_EQ("", "");
}
TEST_F(ClassDeclaration, Test_lab24){
    obj.set_value(0);
    ASSERT_EQ(0, obj.get_value());
    obj.set_value(1131);
}

```

```

    ASSERT_THAT(4, testing::Eq(obj.get_value()));
    ASSERT_EQ("", "");
}
TEST_F(ClassDeclaration, Test_lab25){
    obj.set_value(589954);
    ASSERT_EQ(6, obj.get_value());
    obj.set_value(1);
    ASSERT_THAT(1, testing::Eq(obj.get_value()));
    ASSERT_EQ("", "");
}
TEST_F(ClassDeclaration, Test_lab26){
    obj.set_value(372);
    ASSERT_EQ(3, obj.get_value());
    obj.set_value(47845);
    ASSERT_THAT(5, testing::Eq(obj.get_value()));
    ASSERT_EQ("", "");
}

```

7. И файл main.cpp, который нужен для связки тестов в проекте

```

#include <iostream>
#include <gtest/gtest.h>
#include <gmock/gmock-actions.h>
int main(int argc, char* argv[]) {
    testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
    return 0;
}

```

8. Снова переходим в файл с тестами, дописываем функциональную проверку и дебажим:

