Введение в тестирование

Разработка через тестирование

Кулаков Кирилл Александрович

Петрозаводск — 2017

Классическая схема разработки

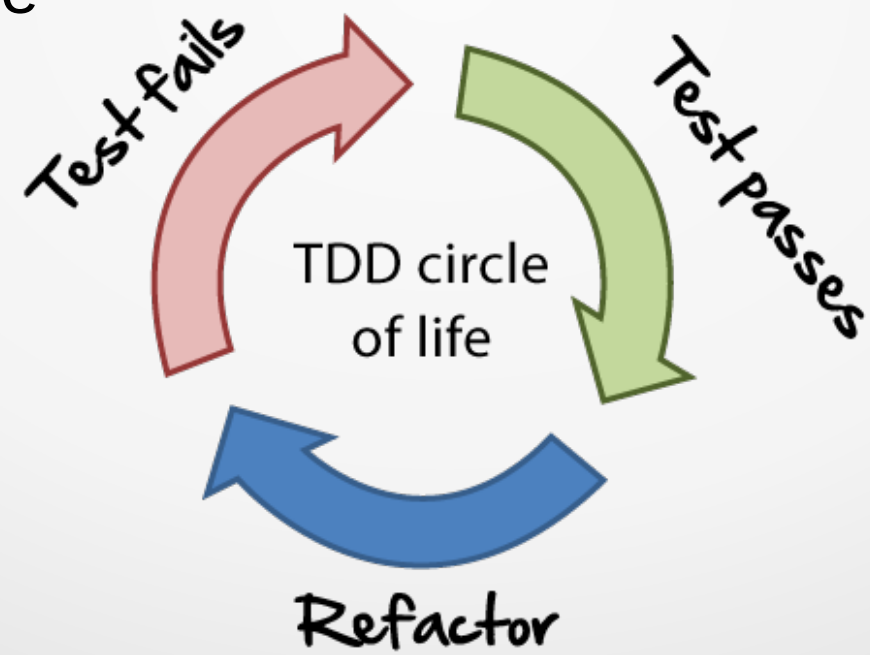
- Получем ТЗ от заказчика
- Немного думаем
- Пишем код
- Немного думаем
- Пишем еще код для предыдущего кода
- …
- Быстренько проверяем что что-то работает
- Сдаем заказчику

Недостатки традиционного подхода

- Слишком сложный код для решения простой задачи
- Что-то не работает
- Что-то забыли реализовать
- Монолитный проект: внесение небольшого изменения может привести к краху и требует осмысления
- Нет необходимости (времени и желания) писать тесты и проверки
- Демонстрация заказчику/разработчикам:
  - Что показывать?
  - Как показывать?
  - "Эффект демо"

Разработка через тестирование

- (Красный) Для требования пишем тесты
- (Зеленый) Пишем новый код только тогда, когда автоматический тест не сработал
- (Синий) Выполняем рефакторинг, устраняем дублирование



Преимущества

- Всегда работающий код
- Постоянный запуск кода → получаем представление о том, как он работает → помогает нам принимать правильные решения
- Мы самостоятельно пишем свои собственные тесты, так как не можем ждать, пока кто-нибудь напишет их для нас → 100% покрытие тестами
- Быстрое реагирование экосистемы разработки на небольшие модификации кода
- Слабо сцепленные компоненты → упрощение модификации и тестирования
- Меньшее время отладки

Алгоритм работы

- Написать тест
- Заставить компилироваться
- Запустить тест и убедиться, что он не работает
- Заставить тест работать
- Рефакторинг
- Проверить, что тест работает

Результаты внедрения

- Если количество дефектов в программе становится достаточно низким, то команда контроля качества может перейти от реактивной к превентивной работе.
- Если количество неприятных сюрпризов будет небольшим, то менеджеры проекта смогут с высокой точностью оценивать трудозатраты и привлекать заказчиков к процессу ежедневной разработки проекта.
- Если темы технических дискуссий будут четко определены, то программисты смогут взаимодействовать друг с другом постоянно, а не раз в день или раз в неделю.
- Если плотность дефектов будет достаточно небольшой мы сможем каждый день получать интегрированный рабочий продукт с добавленной в него новой функциональностью

Виды тестов

- Тест одного шага (One Step Test)
  - Тест должен добавить новую функциональность.
  - Вы должны реализовать тест за один шаг.
- Начальный тест (Starter Test)
  - Первый тест не выполняет каких либо сложных действий.
  - Тривиальные входные и выходные данные.
- Объясняющий тест (Explanation Test)
  - Объясняйте работу кода в виде тестов.
  - Просите у других объяснять работу их кода в виде тестов.
  - Преобразовывайте диаграммы последовательностей в тесты.

Виды тестов

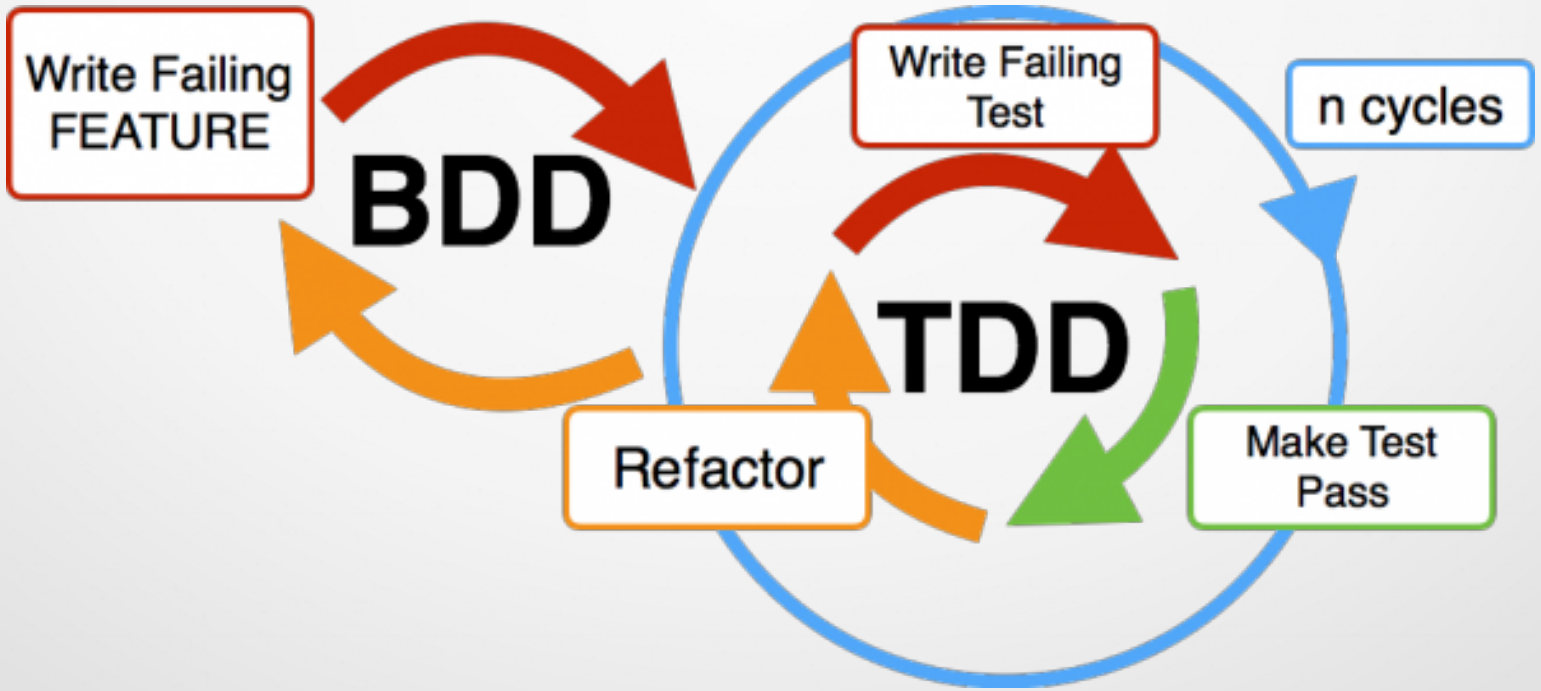
- Тест для изучения (Learning Test)
  - Перед использованием метода библиотеки, поведение которого не совсем ясно.
  - При проверки новой версии библиотеки, которую вы используете.
- Регрессионный тест (Regression Test)
  - Напишите тест прежде чем исправлять ошибку.
- Дочерний тест (Child Test)
  - Разделение большого теста на несколько маленьких

Проблемы TDD

- Высокий порог вхождения
- Ошибочный тест = ошибочный код
- Поддержка большого количества тестов
- Сложно произвести автоматическое тестирование GUI. Особенно в случае нестандартного интерфейса.
- Сложно тестировать распределенные объекты и много-поточные приложения.
- TDD нельзя использовать для разработки схемы базы данных.
- TDD нельзя использовать для разработки языка программирования.

Behavior-driven development

- Ответвление TDD
  - Связь кода с требованиями
  - Фокус на поведении а не на тестах
  - Запись требований с помощью обычных фраз



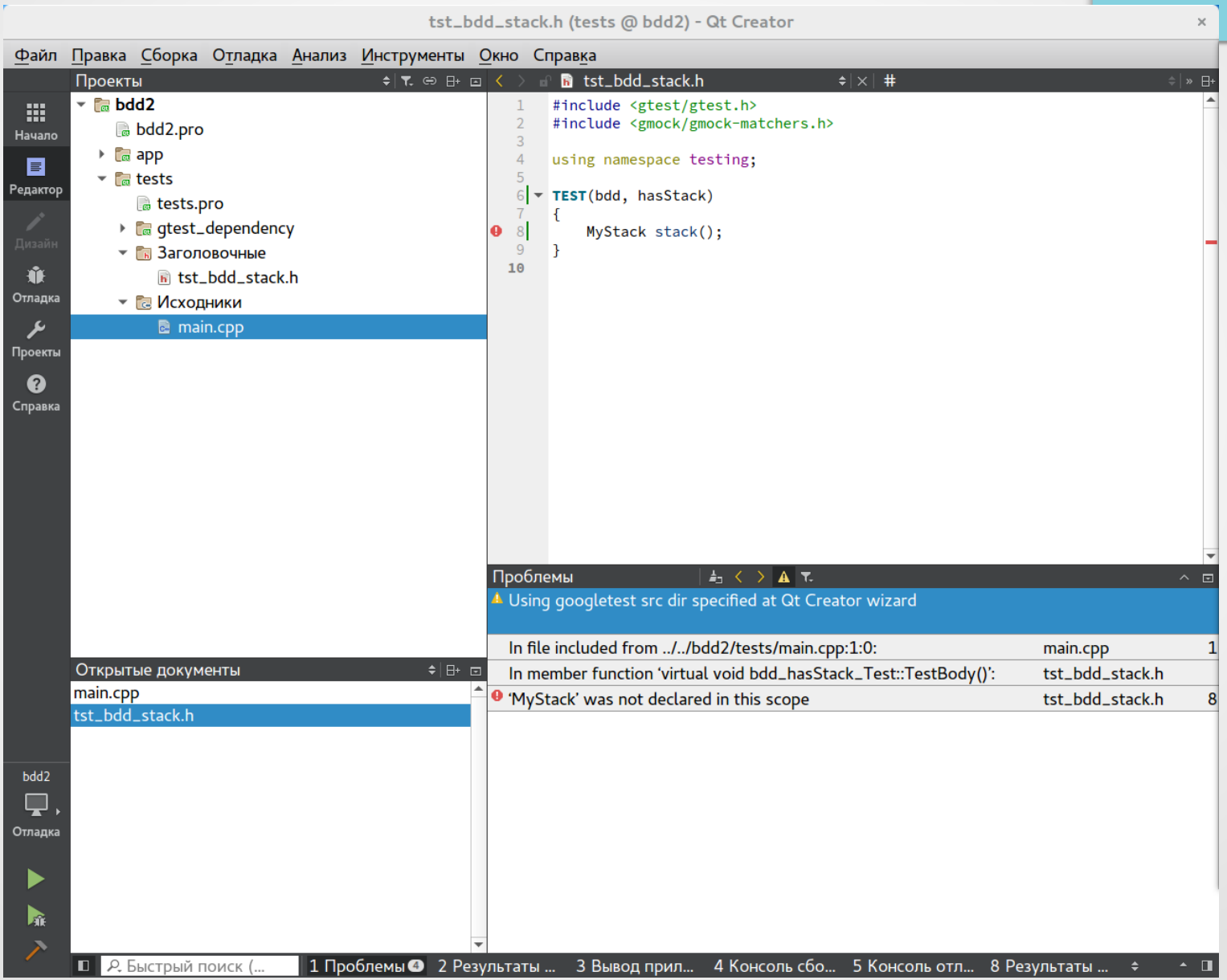
Пример

- Необходимо реализовать работу со стеком
  - Что такое стек? → структура данных, с порядком объектов «первым вошел, последним вышел» или "последним вошел, первым вышел", есть методы push(), pop(), peek().
  - Что делает метод push()? → push() принимает входной объект и помещает во внутренний контейнер, ничего не возвращает
  - Что будет, если передать методу push() два объекта, например, сначала foo, а потом bar? → второй объект при вызове pop() должен быть извлечен первым

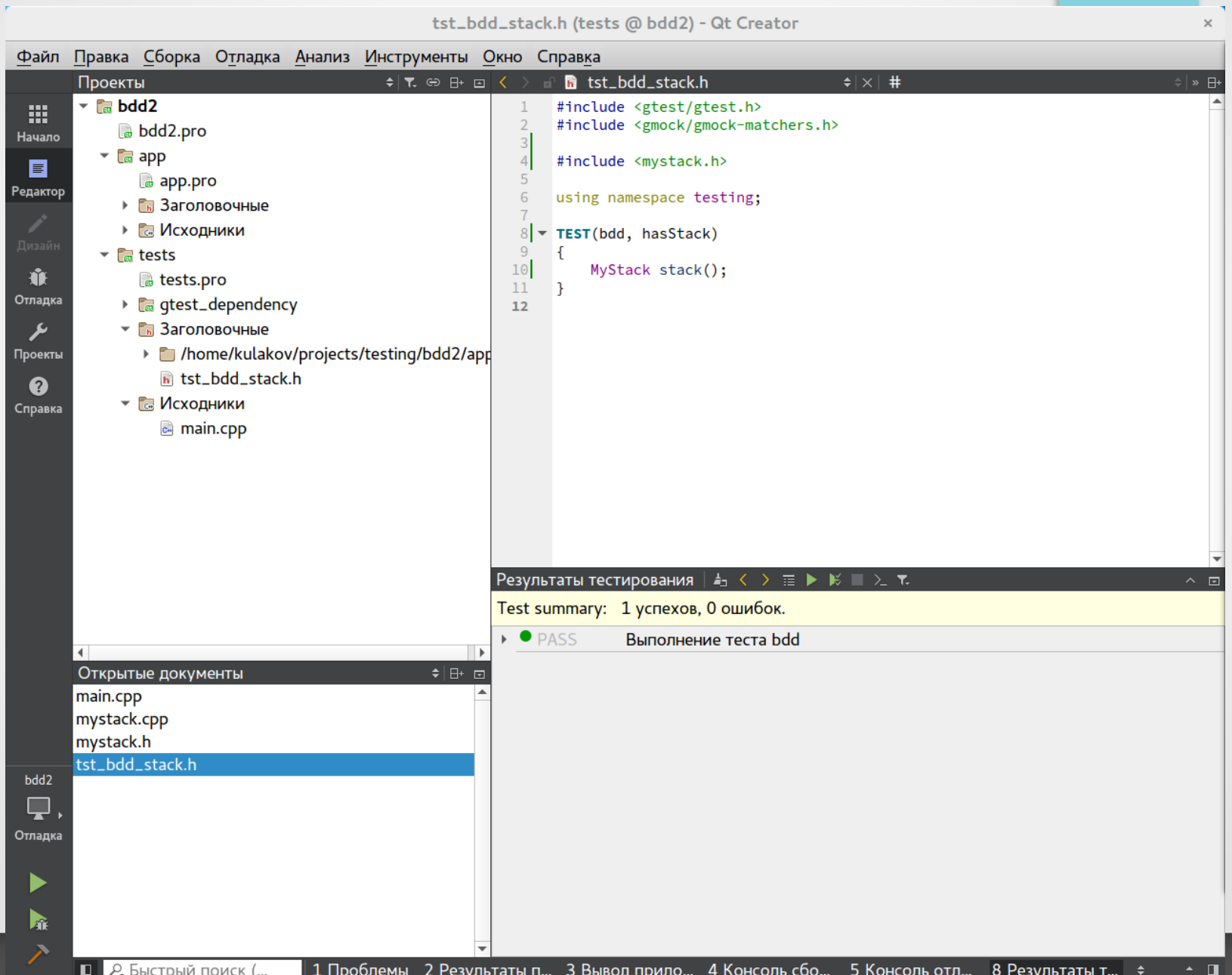
Пример

- Что делает метод pop()? → Извлекает последний объект из стрека если он там есть и возвращает его
- Что будет делать pop() если в стек еще ничего небыло добавлено? → должна выдаваться ошибка
- Что будет, если выполнить команду push() null? → должна выдаваться ошибка

Реализация. Шаг 1. Подготовка



Реализация. Шаг 1. Подготовка

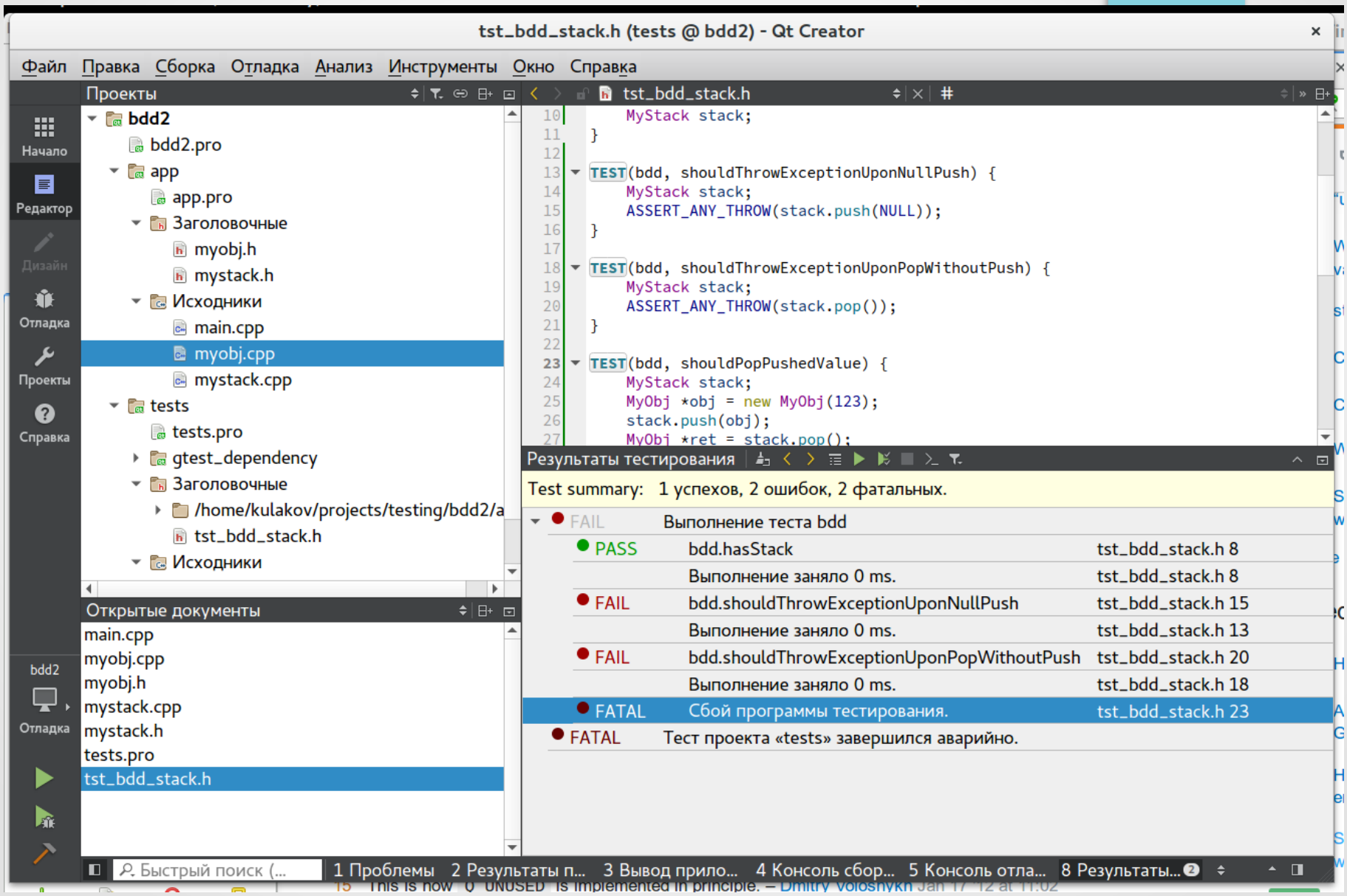


Реализация. Шаг 2. BDD

- Добавление тестов BDD

```
TEST(bdd, hasStack) { … }
TEST(bdd, shouldThrowExceptionUponNullPush) { ... }
TEST(bdd, shouldThrowExceptionUponPopWithoutPush) { … }
TEST(bdd, shouldPopPushedValue) { … }
TEST(bdd, shouldPopSecondPushedValueFirst) { … }
TEST(bdd, shouldLeaveValueOnStackAfterPeek) {
    MyStack stack;
    MyObj *obj1 = new MyObj(123);
    MyObj *obj2 = new MyObj(321);
    stack.push(obj1);
    stack.push(obj2);
    MyObj *ret = stack.pop();
    ASSERT_TRUE(obj2->equals(stack.peek()));
    ASSERT_TRUE(obj2->equals(ret));
    ASSERT_TRUE(obj1->equals(stack.peek()));
}
```

Реализация. Шаг 3. Запуск



Реализация. Шаг 4. Исправление

- Сценарии с исключениями (3 успеха, 3 ошибки, 1 фатальная)

```
void MyStack::push(MyObj *val) {
    if (val == NULL) {
        throw std::invalid_argument("Argument val is null");
    }
}

MyObj *MyStack::pop() {
    if (startItem == NULL) {
        throw std::invalid_argument("Stack is empty");
    }
    return NULL;
}
```

Реализация. Шаг 4. Исправление

- Сценарии с добавлением/удалением элементов (5 успехов, 1 ошибка, 1 фатальная)

```
void MyStack::push(MyObj *val) {
    if (val == NULL) {
        throw std::invalid_argument("Argument val is null");
    }

    StackItem *new_item = (StackItem *)malloc(sizeof(StackItem));
    new_item->next = startItem;
    new_item->val = val;
    startItem = new_item;
}
```

Реализация. Шаг 4. Исправление

```
MyObj *MyStack::peek() {
    if (startItem == NULL) {
        throw std::invalid_argument("Stack is empty");
    }
    return startItem->val;
}

MyObj *MyStack::pop() {
    if (startItem == NULL) {
        throw std::invalid_argument("Stack is empty");
    }
    StackItem *ret_item = startItem;
    startItem = startItem->next;
    MyObj *ret = ret_item->val;
    free(ret_item);
    return ret;
}
```

Реализация. Шаг 5. Завершение

- Ошибка в тесте

```
TEST(bdd, shouldLeaveValueOnStackAfterPeep) {
    MyStack stack;
    MyObj *obj1 = new MyObj(123);
    MyObj *obj2 = new MyObj(321);
    stack.push(obj1);
    stack.push(obj2);
    ASSERT_TRUE(obj2->equals(stack.peek()));
    MyObj *ret = stack.pop();
    ASSERT_TRUE(obj2->equals(ret));
    ASSERT_TRUE(obj1->equals(stack.peek()));
}
```