Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский государственный университет аэрокосмического приборостроения»

УЧЕБНАЯ ПРАКТИКА

(Первый курс. II семестр)

Методические указания

Автор: Гуков Сергей Юрьевич

Версия: 2024.01

Санкт-Петербург

2024

Учебная практика длится весь семестр, защита проекта происходит на зачетной неделе. Однако сдать работу можно в течение всего семестра, не обязательно дожидаться официальной даты сдачи. На защиту необходимо приходить с ***уже готовыми отчетом и программой****. Обратите внимание на дополнительную информацию, которая указана в описании задания в личном кабинете. Также можно обращаться со всеми возникшими вопросами и трудностями к преподавателю.*

Контакты для связи с преподавателем (Гуков Сергей Юрьевич):

- *ВКонтакте: vk.com/bruimafia*
- *Telegram: @bruimafia*
- *Email: sg_brui@mail.ru*

Контакты для связи с преподавателем (Шабунин Андрей Павлович):

- *Telegram: @andreyshabunin*
- *Email: andreyshabunin1999@mail.ru*

Учебная ознакомительная практика входит в состав обязательной части образовательной программы подготовки обучающихся по направлению подготовки/ специальности 09.03.02 «Информационные системы и технологии». Организацию и проведение практики осуществляет кафедра №42.

Цели проведения учебной практики: закрепление и углубление теоретических знаний, полученных при изучении естественно-научных и профессиональных дисциплин; приобретение практических навыков и компетенций в сфере профессиональной деятельности; освоение студентами перспективных информационных технологий; приобретение опыта применения современной вычислительной техники для решения практических задач.

Задачи проведения учебной практики: умение самостоятельно или в составе научно-производственного коллектива решать конкретные профессиональные задачи; формирование и развитие у студентов профессионально значимых качеств, устойчивого интереса к профессиональной деятельности, потребности в самообразовании; изучение специальной литературы и другой научно-технической информации, достижений отечественной и зарубежной науки и техники в области информационных технологий и систем; сбор, обработка, анализ и систематизация научно-технической информации по теме (заданию);

оформление результатов анализа информации по заданной теме и собственных исследований и разработок в виде отчета и технической документации.

Учебная ознакомительная практика обеспечивает формирование у обучающихся следующих компетенций:

– УК-6 «Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни»;

– ОПК-1 «Способен применять естественнонаучные и общеинженерные знания, методы математического анализа и моделирования, теоретического и экспериментального исследования в профессиональной деятельности»;

– ОПК-2 «Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности»;

– ОПК-3 «Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности»;

– ОПК-4 «Способен участвовать в разработке технической документации, связанной с профессиональной деятельностью с использованием стандартов, норм и правил»;

– ОПК-6 «Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий»;

– ПК-3 «Способен разрабатывать программное обеспечение, выполнять интеграцию программных модулей и компонентов».

Содержание практики охватывает круг вопросов, связанных с разработкой программного обеспечения и написания для него технической документации.

Аттестация по практике осуществляется путем защиты отчетов, составляемых обучающимися по итогам практики. Форма аттестации по практике – дифференцированный зачет.

ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ

В рамках учебной практики необходимо закрепить полученные знания по C++ на дисциплине «Основы программирования», написав цельное достаточно объемное приложение и оформив для него техническую документацию.

Необходимо реализовать какую-либо консольную игру (без графического интерфейса). Также можно разработать игру с наличием интерфейса и применением какого-либо движка. Необходимо, чтобы в игре было реализовано взаимодействие нескольких игроков и взаимодействие игрока с компьютером (если игра такое подразумевает), игра должна быть достаточно функциональной. Должно быть несколько уровней сложности, если так же игра это подразумевает. Выбор темы игры необходимо утвердить с преподавателем. Примеры игр: крестики-нолики (режимы игры двух игроков и игры с компьютером), сапер, покер и другие алгоритмически несложные карточные игры, симуляторы (жизни, студента, бездомного и подобные), упрощенный морской бой, кто хочет стать миллионером и прочие викторины, тамагочи, кроссворд, виселица и другие.

Готовый проект необходимо загрузить на GitHub или аналогичные системы контроля версий и оформить там файл «README.md».

На защите необходимо иметь само приложение с исходным кодом и готовый оформленный отчет. Также быть готовым ответить на ряд возможных теоретических вопросов по своей работе и своему коду программы. При несогласии с предлагаемой преподавателем оценкой, можно прямо на защите попробовать сделать небольшое практическое задание на дополнительные баллы.

ЧАСТЬ 1. РЕАЛИЗАЦИЯ ЗАДАНИЯ

Что такое GitHub и как с ним работать?

Git – распределённая система контроля версий, которая даёт возможность разработчикам отслеживать изменения в файлах и работать над одним проектом совместно с коллегами.

Чтобы лучше понимать, что такое Git и как он работает, нужно знать, что такое система контроля версий. Системы контроля версий (СКВ, VCS, Version Control Systems) позволяют разработчикам сохранять все изменения, внесённые в код. При возникновении проблем разработчики могут просто откатить код до рабочего состояния и не тратить часы на поиски ошибок. СКВ также позволяют нескольким разработчикам работать над одним проектом и

сохранять внесённые изменения независимо друг от друга. При этом каждый участник команды видит, над чем работают коллеги.

GitHub – наиболее популярный сервис онлайн-хостинга репозиториев, обладающий всеми функциями распределённого контроля версий и функциональностью управления исходным кодом – всё, что поддерживает Git и даже больше. Также GitHub может похвастаться контролем доступа, багтрекингом, управлением задачами и вики для каждого проекта. Кроме GitHub есть другие сервисы, которые используют Git, – например, Bitbucket и GitLab. Вы можете разместить свой проект на любом из них.

Рассмотрим работу с репозиториями на примере GitHub. Для работы с GitHub необходимо зайти на сайт https://github.com/ и создать аккаунт. В целом есть несколько вариантов, как залить/управлять свой репозиторий (проект) на сайт. Рассмотрим один из них.

В программе Visual Studio есть вкладка «Git», в которой необходимо «Создать репозиторий» и ввести данные от своего аккаунта (рисунок 11). Далее, выбрав имя репозитория, отправить его на GitHub.

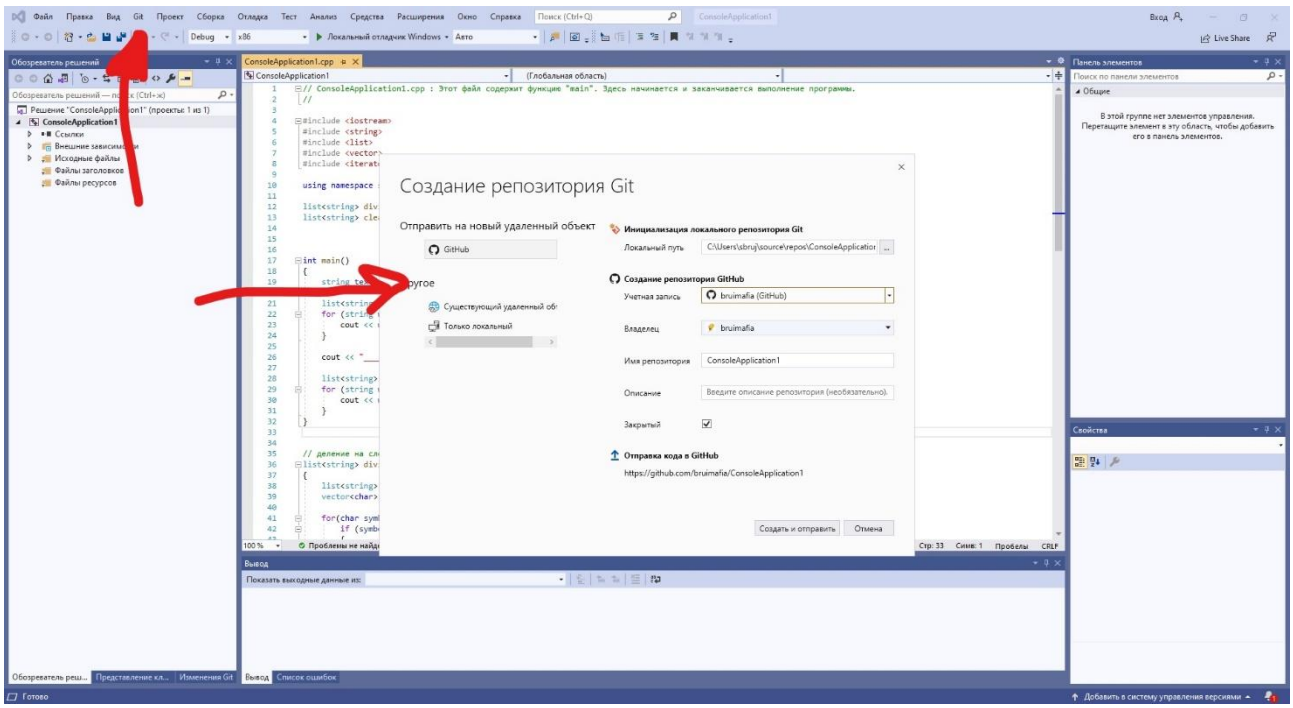

Рисунок 11 – Создание репозитория

Теперь во вкладке «Git» будет доступен примерно следующий функционал (рисунок 12). А в нижней вкладке «Изменения Git» появится возможность отправлять коммиты.

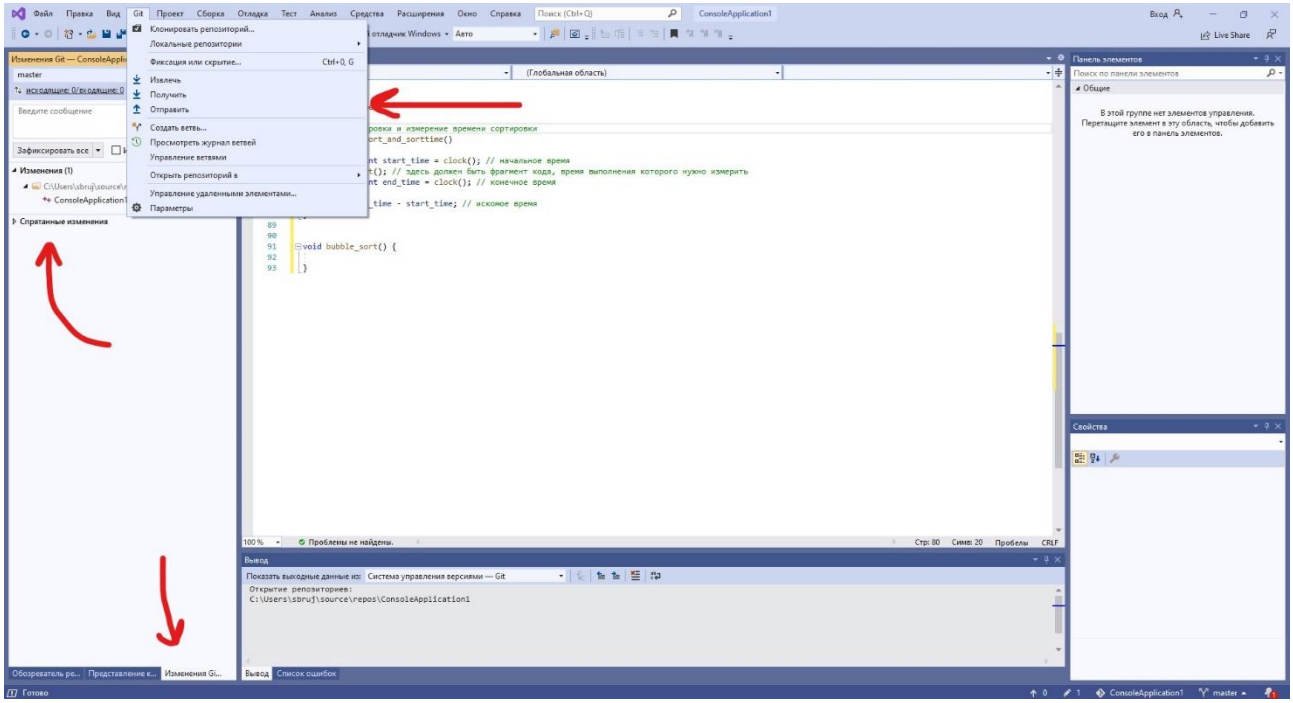

Рисунок 12 – Управление репозиторием

Git-коммит – это операция, которая берет все подготовленные изменения и отправляет их в репозиторий как единое целое. Каждый коммит в git репозитории хранит снимок всех файлов в репозитории. Почти как огромная копия, только эффективнее. Git пытается быть лёгким и быстрым, так что он не просто слепо копирует весь проект каждый раз, а ужимает коммит в набор изменений или «дельту» между текущей версией и предыдущей. Это позволяет занимать меньше места. Также Git хранит всю историю о том, когда какой коммит был сделан и кем. Это очень важно, так как можно посмотреть, из-за чего и из-за кого сломался весь код и «надавать ему по башке».

Возможность отправить коммит появляется после любого мельчайшего изменения кода. Однако делать это желательно после каждого идеологического/алгоритмического изменения кода программы. Перед нажатием «Зафиксировать все и отправить» следует ввести имя/описание коммита, отражающее причину изменения либо нововведения кода (рисунок 13).

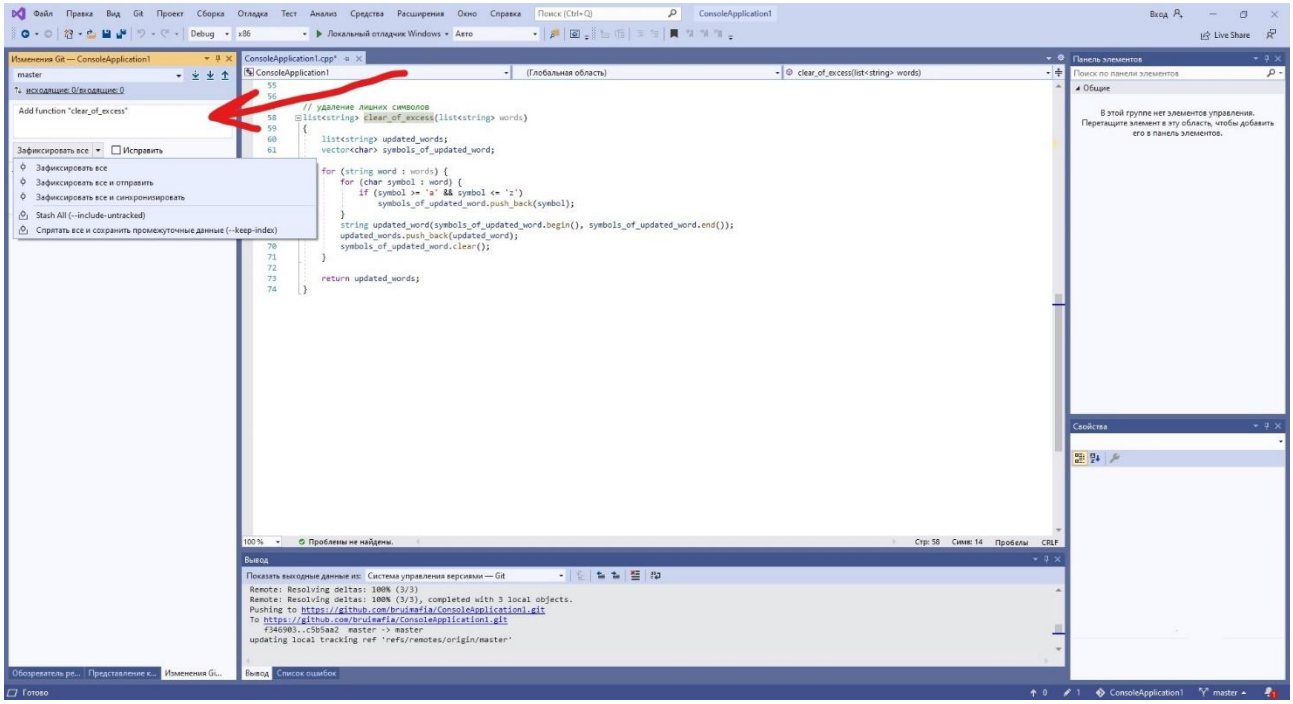

Рисунок 13 – Создание коммита

Для описания проектов на GitHub используется файл «README.md», который пишется на языке разметки markdown. «README.md» («прочти меня») – это первый файл, который нужно читать, получив доступ к проекту на GitHub или любой Git-хостинговой площадке. Этот файл в первую очередь и предлагается вниманию пользователя, когда он открывает здесь репозиторий того или иного проекта. Такой файл содержит кучу полезной информации, так что его вполне можно рассматривать как справочное руководство по проекту. Файл «README.md» находится в корневой папке репозитория и автоматически отображается в каталоге проекта на GitHub. Примеры файлов «README.md» приведены на рисунках 14-16.

При составлении описания проекта, то есть файла «README.md», обычно придерживаются такого плана:

- название проекта;
- описание проекта (с использованием слов и изображений);
- демо (изображения, ссылки на видео, интерактивные демо-ссылки);
- технологии/фишки, использованные в проекте;
- что-то характерное для проекта (проблемы, с которыми пришлось столкнуться, уникальные составляющие проекта);
- техническое описание проекта (установка, настройка, как помочь проекту).

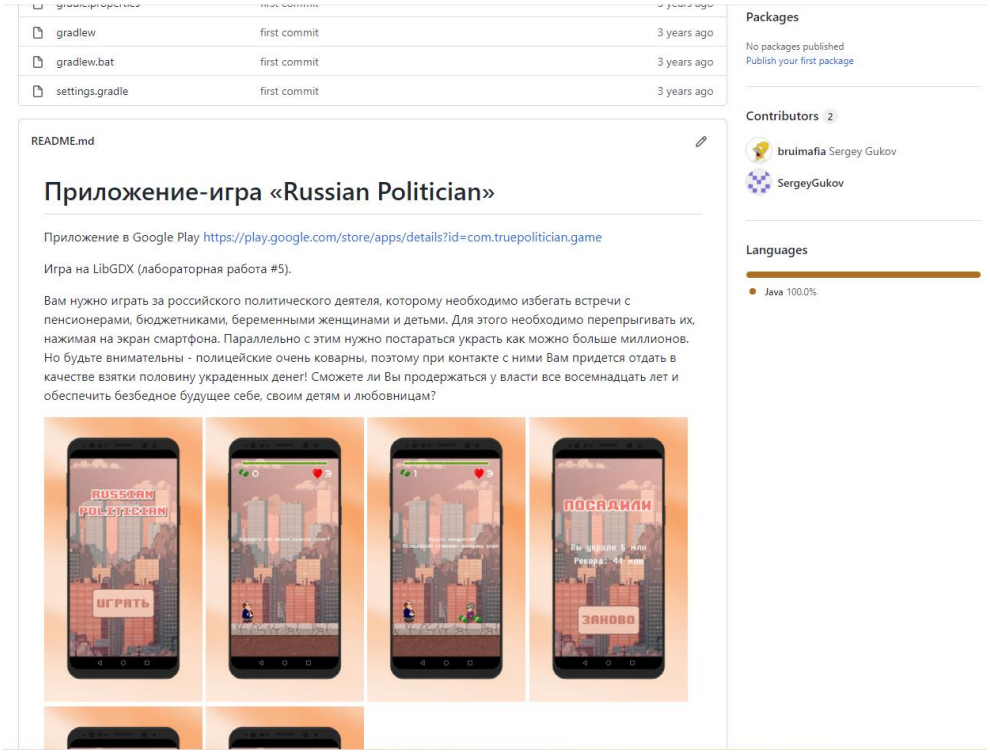

Рисунок 14 – Пример файла «README.md»

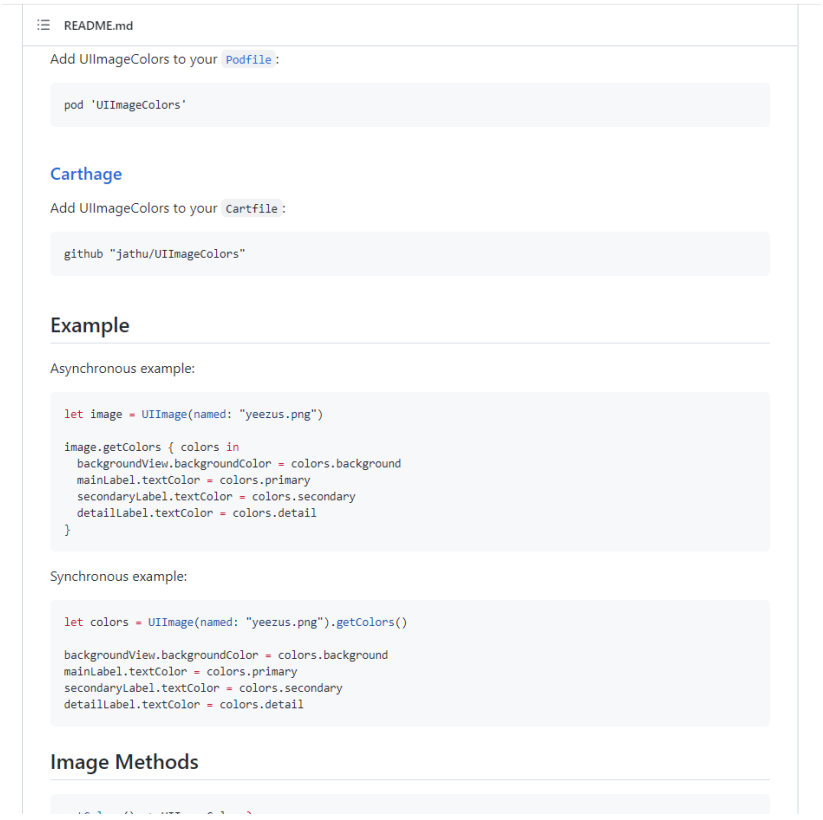

Рисунок 15 – Пример файла «README.md»

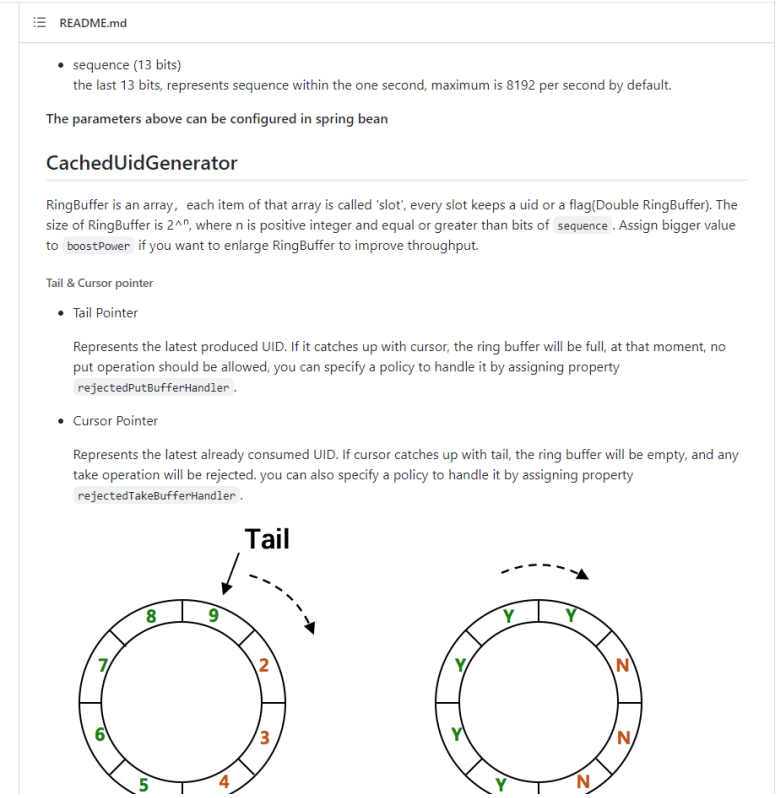

Рисунок 16 – Пример файла «README.md»

В общем, настоятельно рекомендуется познакомиться/поиграться с GitHub или другими подобными сервисами, так как практически в любой компании при разработке проекта используются системы контроля версий.

ЧАСТЬ 2. НАПИСАНИЕ ДОКУМЕНТАЦИИ

Рано или поздно разработчик сталкивается с ситуацией, когда не может понять написанный код программы. Такое случается и когда он приходит в новый для себя проект, и когда коллеги проверяют его код, и даже, когда он смотрит результат своей же работы. Чтобы избежать подобных проблем и упросить поддержку проекта, необходимо писать и читать документацию.

Для разработчика есть два основных вида документации:

– *Пользовательская документация.* Это руководство по эксплуатации программ. Обычно оно нужно для сложных профессиональных инструментов и является избыточным для небольших проектов. Если пользователи не могут сами разобраться в приложении заказа такси, то скорее всего проблемы в самом интерфейсе и лучше его доработать и упростить.

– *Техническая документация.* Это пояснения для программистов, которые будут использовать или дорабатывать существующий код. Они помогут быстро вникнуть в проект и начать работать. Или же продолжить самому разработчику писать программу после долгого перерыва.

Основные правила написания технической документации:

– *Документация нужна не всегда.* Если программа выглядит как небольшой скрипт, не стоит тратить время на написание пояснений.

– *Документация нужна не везде.* Если код написан хорошо, то по названиям уже будет понятно, что это такое и зачем оно используется. Исключение можно сделать, если речь идет об API или фреймворке, которыми пользуются многие разработчики: они не всегда видят исходный код, но могут использовать классы и методы.

– *Документация должна быть точной.* Очень важно уметь ясно выражать свои мысли. Нужно предельно точно описывать, что делает тот или иной фрагмент кода.

– *Документация должна быть сухой.* Нужно писать надо максимально сухо и по делу. Избегать стихов, метафор, аналогий, шуток и т.п.

– *В документации не должно быть старого кода.* Старайтесь не хранить в коде старые методы и операторы, даже если они закомментированы. Если что-то не используется в текущий момент, то от которого нужно избавиться. Если есть сомнения пригодится ли еще этот код, его лучше сохранить в системе контроля версий.

Ниже на рисунках 17-20 представлен один из вариантов хорошего описания функций (методов) и переменных (полей). Более подробную информацию об этих примерах можно посмотреть по следующим ссылкам:

- https://yandex.ru/dev/mobile-ads/doc/android/ref/com/yandex/mobile/ads/banner/AdSize.html
- https://yandex.ru/dev/mobile-ads/doc/android/ref/com/yandex/mobile/ads/nativeads/Rating.html
- https://yandex.ru/dev/mobile-ads/doc/android/ref/com/yandex/mobile/ads/rewarded/RewardedAd.html

Modifier and Type	Method and Description
`static AdSize`	`flexibleSize(int width, int height)` Задает размеры баннера с учетом размеров контейнера.
`int`	`getHeight()` Возвращает высоту баннера в dp (density-independent pixels).
`int`	`getHeight(android.content.Context context)` Возвращает высоту баннера в dp (density-independent pixels).
`int`	`getHeightInPixels(android.content.Context context)` Возвращает высоту баннера в пикселях.
`int`	`getWidth()` Возвращает ширину баннера в dp (density-independent pixels).
`int`	`getWidth(android.content.Context context)` Возвращает ширину баннера в dp (density-independent pixels).
`int`	`getWidthInPixels(android.content.Context context)` Возвращает ширину баннера в пикселях.
`static AdSize`	`stickySize(int width)` Задает ширину sticky баннера.

Рисунок 17 – Описание функций (слева – тип возвращаемого значения, справа – имя и назначение)

Modifier and Type	Method and Description
`float`	`getRating()` Возвращает значение рейтинга.
`void`	`setRating(float rating)` Задает значение рейтинга.

Рисунок 18 – Описание функций (слева – тип возвращаемого значения, справа – имя и назначение)

Modifier and Type	Method and Description
`void`	`destroy()` Удаляет объект класса `RewardedAd` и очищает все занятые ресурсы.
`boolean`	`isLoaded()` Возвращает результат загрузки рекламы.
`void`	`loadAd(AdRequest adRequest)` Начинает загрузку рекламы в фоновом режиме.
`void`	`setAdUnitId(java.lang.String adUnitId)` Задает уникальный идентификатор рекламного места.
`void`	`setRewardedAdEventListener(RewardedAdEventListener rewardedAdEventListener)` Задает объект класса `RewardedAdEventListener` для получения оповещений о событиях, происходящих во время жизненного цикла рекламы с вознаграждением.
`void`	`show()` Показывает рекламу, если она была загружена.

Рисунок 19 – Описание функций (слева – тип возвращаемого значения, справа – имя и назначение)

Modifier and Type	Field and Description
`static AdSize`	`BANNER_240x400` Баннер с размерами 240x400 в dp (density-independent pixels).
`static AdSize`	`BANNER_300x250` Баннер с размерами 300x250 в dp (density-independent pixels).
`static AdSize`	`BANNER_300x300` Баннер с размерами 300x300 в dp (density-independent pixels).
`static AdSize`	`BANNER_320x100` Баннер с размерами 320x100 в dp (density-independent pixels).
`static AdSize`	`BANNER_320x50` Баннер с размерами 320x50 в dp (density-independent pixels).
`static AdSize`	`BANNER_400x240` Баннер с размерами 400x240 в dp (density-independent pixels).
`static AdSize`	`BANNER_728x90` Баннер с размерами 728x90 в dp (density-independent pixels).
`static AdSize`	`FULL_SCREEN` Баннер на весь размер экрана.

Рисунок 20 – Описание переменных (слева – тип, справа – имя и назначение)

Существуют различные варианты и стандарты по написанию документации. В случае выполнения задания по учебной практике на языке C++ необходимо использовать:

- блок-схему программы;
- краткие комментарии в коде (однострочные, многострочные);
- описание функций и переменных (как на примерах выше);
- описание алгоритмов программы;
- описание взаимодействия пользователя с программой.

ЧАСТЬ 3. ТРЕБОВАНИЯ К ОТЧЕТУ

Весь программный код должен быть разбит на отдельные функции/методы, писать весь код внутри функции «void main()» не допускается. Имена функций и переменных должны отражать их назначение. Основные части кода должны иметь комментарии. Желательно придерживаться главных пунктов конвенции по оформлению кода C++ (https://habr.com/ru/post/172091/ и https://tproger.ru/translations/stanford-cpp-style-guide/) либо языка, на котором пишется проект.

Объем отчета не менее 17 страниц без учета приложения (в приложении весь исходный код).

Образцы титульного листа по практике и бланка индивидуального задания по практике размещены на сайте ГУАП в разделе Нормативная документация в строке «Документация для учебного процесса».

Отчет оформляется в соответствии с требованиями, описанными на сайте ГУАП в разделе Нормативная документация (URL: https://guap.ru/standart/doc).

В отчете по учебной практике должно быть следующее:

- титульный лист;
- индивидуальное задание;
- содержание;
- введение;
- тема и описание игры;
- пользовательская документация (описание взаимодействия пользователя с программой);
- техническая документация (блок-схема, комментарии в коде, описание алгоритмов, описание функций и переменных);
- тестирование программы;
- ссылка на короткое видео с демонстрацией механики игры;
- описание назначения и процесса загрузки проекта на Github (реализовать 2-4 коммита с описанием изменений), скриншоты файла документации «README.md» (в этом файле тоже указать ссылку на видео с демонстрацией механики игры);
- заключение;
- список использованных источников;
- приложение с исходным кодом.

СПИСОК ЛИТЕРАТУРЫ

1. Дополнительные материалы в личном кабинете
2. Конспекты лекций
3. Алгоритмы и структуры данных, Н. Вирт

4. Программирование. Принципы и практика с использованием C++, Страуструп Бьярне
5. Язык программирования C++. Лекции и упражнения, Прата Стивен
6. Язык программирования C++. Базовый курс, Липпман Стенли Б., Лажойе Жози
7. 90 рекомендаций по стилю написания программ на C++, https://habr.com/ru/post/172091/ (дата посещения: 06.02.2023)
8. Гайд по оформлению кода на C++ от Стэнфордского университета, https://tproger.ru/translations/stanford-cpp-style-guide/ (дата посещения: 06.02.2023)
9. Основы C++. Программирование для начинающих (канал «#SimpleCode»), https://www.youtube.com/watch?v=kRcbYLK3OnQ&list=PLQOaTSbfxUtCrKs0nicOg2npJQYSPGO9r (дата посещения: 06.02.2023)
10. C++ программирование / Уроки C++ (канал «Гоша Дударь»), https://www.youtube.com/watch?v=qSHP98i9mDU&list=PL0lO_mIqDDFXNfqIL9PHQM7Wg_kOtDZsW (дата посещения: 06.02.2023)
11. Краткое введение в Git и его использование с GitHub - GitHub - ashtanyuk/Git-intro: Краткое введение в Git и его использование с GitHub, https://github.com/ashtanyuk/Git-intro (дата посещения: 06.02.2023)