

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ
И ИНФОРМАТИКИ

Кафедра технологий программирования

ЛАБОРАТОРНАЯ РАБОТА 9
ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ МОБИЛЬНЫХ И
ВСТРАИВАЕМЫХ СИСТЕМ»

Разработка мобильного приложения для ОС Аврора с использованием QT

Методические указания по выполнению лабораторной работы

Минск 2025


СОДЕРЖАНИЕ

Введение	3
1 Методические рекомендации.....	4
1.1 Основной учебный материал	4
1.2 Установка Aurora SDK.....	5
1.3 Примеры проектов мобильных приложений для ОС Аврора....	6
1.4 Сборка проекта мобильного приложения и запуск в ОС Аврора	6
1.5 Тестирование мобильных приложений для ОС Аврора	8
2 Методические указания и задания.....	10
2.1 Методические указания.....	10
2.1.1 Критерии оценивания	10
2.1.2 Отчет по лабораторной работе	11
2.1.2.1 Требования к репозиторию и файлу README репозито-	
рия.....	11
2.1.2.2 Защита отчета по лабораторной работе	12
2.2 Задания.....	13
2.2.1 Задание 1.....	13
2.2.2 Задание 2.....	14
2.2.3 Задание 3.....	15
2.3 Контрольные вопросы	15



ВВЕДЕНИЕ

Целью работы является получение навыков разработки мобильных приложений на языке C++ в среде QT-Creator с использованием Aurora SDK для мобильной операционной системы Аврора. Для достижения поставленной цели необходимо решить следующие задачи:

- установить пакет Aurora SDK, включающий Aurora IDE, Aurora SDK, виртуальные машины со средой сборки и эмулятором под управлением ОС Аврора;
- изучить примеры и разработать мобильное приложение согласно заданию.

В случае недоступности  портала для разработчиков приложений для Аврора ОС рекомендуется реализовать мобильное приложение средствами QT Creator и QML для ОС Android или вместо Аврора SDK установить Sailfish SDK, которое включает ОС Sailfish и преднастроенную среду QT Creator для разработки мобильных приложений для ОС Sailfish.

Методические материалы для разработки мобильных приложения средствами QT Creator для ОС Android:

- Qt Creator Documentation >  Tutorial: Mobile application.
-  Modern mobile applications with Qt and QML.

Руководства и учебные материалы для разработки мобильных приложений для Sailfish OS:

- Бесплатный учебный курс  Введение в разработку приложений на Sailfish OS на stepik.org.
- Руководство по установке  Sailfish Platform SDK.



1 Методические рекомендации

В данном разделе представлены рекомендации по выполнению лабораторной работы.





1.1 Основной учебный материал

Учебный материал для выполнения лабораторной работы изложен в источниках:


а) Платформа ОС Аврора:


- 1)  Архитектура ОС Аврора;
- 2)  Разработка прикладного программного обеспечения

б) Учебные материалы по разработке приложений для ОС Аврора:



-  Разработка прикладного программного обеспечения
-  Учебный курс «Разработка приложений для ОС Аврора»
-  Вебинары по разработке приложений для ОС Аврора
-  Образовательная среда Аврора (учебные материалы и др.)

в)  QML Applications;

г)  ШлееМ. Qt 5.10. Профессиональное программирование на C++. — СПб:БХВ-Петербург, 2018. — 1072 с..

В случае *недоступности*  портала для разработчиков приложений для Аврора ОС рекомендуется реализовать мобильное приложение средствами QT Creator и QML для ОС Android или вместо Аврора SDK установить Sailfish SDK, которое включает ОС Sailfish и преднастроенную среду QT Creator для разработки мобильных приложений для ОС Sailfish.


Методические материалы для разработки мобильных приложения средствами QT Creator для ОС Android:

- Qt Creator Documentation >  Tutorial: Mobile application.
-  Modern mobile applications with Qt and QML.

Руководства и учебные материалы для разработки мобильных приложений для Sailfish OS:

- Бесплатный учебный курс  Введение в разработку приложений на Sailfish OS на stepik.org.
- Руководство по установке  Sailfish Platform SDK.

1.2 Установка Aurora SDK

В случае невозможности скачать и установить Aurora SDK рекомендуется создать мобильное приложение для ОС Android, используя QT Creator и QML, или же установить аналог Aurora SDK, такой как Sailfish SDK. Руководство по установке  Sailfish Platform SDK.

В состав Aurora SDK входят следующие компоненты:

- *Aurora IDE* (IDE) — интегрированная среда разработки, основанная на Qt Creator, для разработки приложений на языках C, C++ и QML для ОС Аврора с использованием компонентов Sailfish Silica. IDE предоставляет продвинутый редактор кода с интеграцией системы контроля версий, управления проектами и сборками.

- *Aurora OS Emulator* (эмулятор) — виртуальная машина, которая позволяет выполнять приложения в окружении ОС Аврора аналогично работе на мобильных устройствах.


- *Aurora OS Build Engine* (среда сборки) — окружение, которое обеспечивает среду для сборки приложений, не зависящую от домашней операционной системы (ОС).

Перед установкой Аврора SDK нужно убедиться, что компьютер разработчика удовлетворяет минимальным требованиям:

- процессор на архитектуре x86_64;
- не менее 10 Гб свободного дискового пространства;
- не менее 8 Гб оперативной памяти (рекомендуется);
- поддержка аппаратной виртуализации (рекомендуется для быстрой работы виртуальных машин).


Для установки ОС Аврора рекомендуется одна из следующих операционных систем:


- Ubuntu LTS не ниже версии 20.04;
- Альт Рабочая станция 10;
- Windows 10 (далее — Windows) не ниже версии сборки 17063 с архиватором tar, для корректной работы плагина примеров;
- macOS (версии для x86 процессора);


— macOS с архитектурой процессора ARM (M1/M2) только с организацией удаленного сервера Аврора ОС (подробнее см.  <https://github.com/keygenqt/aurora-m1>)


Требования к ПО и работа с мастером установки описаны в документации  «Установка и удаление Аврора SDK»


1.3 Примеры проектов мобильных приложений для ОС Аврора

Проекты программного обеспечения с открытым исходным кодом представлены компанией «Открытая мобильная платформа» в разделе  «Проекты с открытым исходным кодом» и включают следующие репозитории (см. рис. 1.1):

— *Demos*:  gitlab.com/omprussia/demos — небольшие приложения, которые простым образом демонстрируют отдельные технические решения, такие как правильное использование API.

— *Examples*:  Проекты с открытым исходным кодом — полнофункциональные приложения, которые можно использовать как лучшие практики для решения функциональных задач для ОС Аврора.

— *Examples Extra*:  Примеры проектов Examples Extra — полнофункциональные приложения с расширенными зависимостями, поэтому не поставляются через SDK.

— *Tools*:  gitlab.com/omprussia/tools — различные приложения, утилиты, скрипты и другие инструменты, которые могут помочь пользователям в разработке.

Офлайн версия установочного пакета Aurora SDK включает исходный код проектов из категорий *Demos* и *Demos*.

1.4 Сборка проекта мобильного приложения и запуск в ОС Аврора

Приложения для ОС Аврора пишутся на C++/Qt с использованием QML для описания интерфейса пользователя. Создание приложения осуществляется в IDE, основанной на Qt Creator, и практически совпадает с процессами создания приложений для множества настольных и мобильных

Проекты с открытым исходным кодом

Проекты программного обеспечения с открытым исходным кодом компании «Открытая мобильная платформа» располагаются в репозиториях:

- Demos: gitlab.com/omprussia/demos — небольшие приложения, которые простым образом демонстрируют отдельные технические решения, такие как правильное использование API.
- Examples: gitlab.com/omprussia/examples — полнофункциональные приложения, которые можно использовать как лучшие практики для решения функциональных задач для ОС Аврора.
- Examples Extra: gitlab.com/omprussia/examples-extra — полнофункциональные приложения с расширенными зависимостями, поэтому не поставляются через SDK.
- Tools: gitlab.com/omprussia/tools — различные приложения, утилиты, скрипты и другие инструменты, которые могут помочь пользователям в разработке.

Рисунок 1.1 — Репозитории проектов с открытым исходным кодом для ОС Аврора



Рисунок 1.2 — Примеры полнофункциональных проектов в среде разработки Aurora IDE

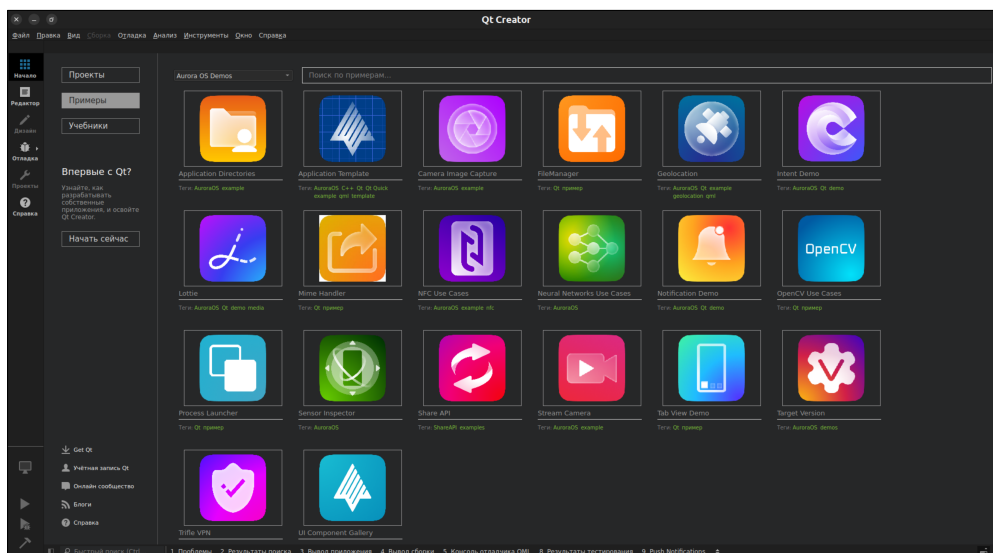




Рисунок 1.3 — Примеры простых проектов в среде разработки Aurora IDE

платформ. Отличия связаны с тем, что сборка происходит в среде сборки, а запуск — в эмуляторе или на внешнем устройстве с ОС Аврора.

Для того чтобы получить приложение, работающее в эмуляторе или на устройстве, необходимо выполнить три последовательных шага:

- Создать или открыть проект.
- Собрать проект.
- Запустить приложение.

Подробнее в документации  «Запуск и отладка проекта» и в статье  Сборка первого приложения для ОС Аврора для реального устройства.

1.5 Тестирование мобильных приложений для ОС Аврора





Аврора IDE поддерживает тестирование приложений с помощью:

- Qt-тестов;
- Google-тестов.

Тесты можно запускать из консоли с помощью машины сборки и эмулятора. Тесты можно запустить тремя способами:

- Отдельной командой после сборки на машине сборки.
- Запуском подпроекта на устройстве или эмуляторе.
- Одновременно со сборкой.

Подробнее о создании тестовом в

-  «Инструкция по запуску тестов Qt»
-  «Тестирование приложений на машине сборки»
-  «Использование Google-тестов»
-  «Инструкция по сбору тестового покрытия»

2 Методические указания и задания

2.1 Методические указания

Общие рекомендации по выполнению заданий лабораторной работы:

- а) Проект для любого из заданий может быть реализован в виде мобильного приложения в среде Aurora IDE и с использованием Aurora SDK.
- б) Проект для заданий 2 и 3 должен содержать модульные тесты.
- в) Проект для каждого из заданий должен предусматривать обработку исключительных ситуаций.

2.1.1 Критерии оценивания

Оценка 4-5

Выполнены задания 1-2. Структура проекта в задании 2 соответствует архитектурному шаблону *MVC* или *MVVM*. Модульные тесты не требуются. Для сборки используется `qmake` или `cmake` (по желанию). Представлен отчет с ответами на контрольные вопросы, исходный код проекта в `git`-репозитории. Лабораторная работа сдана с задержкой в 1-2 недели.

Оценка 6-7

Выполнены задания 1-3. Структура проекта в заданиях 2-3 соответствует архитектурному шаблону *MVC* или *MVVM*. Модульные тесты соответствуют минимальным требованиям (см. задание) и добавлены в один из проектов. Для сборки используется `qmake` или `cmake` (по желанию). Представлен отчет с ответами на контрольные вопросы, исходный код проектов в `git`-репозитории. Лабораторная работа сдана с задержкой в 1 неделю.

Оценка 8-9


Выполнены задания 1-3 на отличном уровне. Структура проекта в заданиях 2-3 соответствует архитектурному шаблону *MVC* или *MVVM*. Модульные тесты добавлены в оба проекта заданий 2-3. Обеспечивается максимальное покрытие тестами.

Представлен отчет, ответы на контрольные вопросы, исходные коды скриптов в `git`-репозитории. Отчет, исходный код не содержат ошибок. Лабораторная работа сдана в срок.

2.1.2 Отчет по лабораторной работе

Отчет по лабораторной работе должен быть опубликован в репозитории и отвечать требованиям:

1. Отчет по лабораторной работе состоит из письменного отчета, кода приложений и тестов к ним, опубликованных в репозитории
2. Письменный отчет содержит цель работы.
3. Письменный отчет включает вариант задания.
4. Письменный отчет добавит описание ключевых моментов реализации и тестов.
5. Исходный код программ для каждого задания опубликовать в подкаталоге `/src` соответствующего каталога проекта (задания) и в соответствующей ветке репозитория.



Ссылка на репозиторий для лабораторной работы 9 доступна в курсе  «Программирование мобильных и встраиваемых систем».


В файле `README` в корневом каталоге проекта на *github* должна быть ссылка на отчет и краткие сведения о выполненных заданиях (проектах).

Отчет опубликовать во внешнем хранилище или в репозитории в каталоге `/docs`. **! Отчёт должен результаты тестов по каждому приложению и ответы на контрольные вопросы.**

2.1.2.1 Требования к репозиторию и файлу `README` репозитория

Корневой каталог репозитория должен включать:

1. файл `README`, содержащий ссылку на отчет;
2. при публикации отчета в репозитории, разместить его в папке `/docs`;
3. каждое задание находится в каталоге проекта, структура которого соответствует требованиям задания;
4. в корневом каталоге репозитория и папках проектов должен быть добавлен файл `.gitignore`, в котором определены правила для исключения временных, исполняемых, объектных файлов и т.д. В качестве примера для проекта мобильного приложения рекомендуется использовать  шаблон `.gitignore` для среды QT из библиотеки шаблонов  A collection

of .gitignore templates на  *GitHub*. При необходимости шаблонный файл может быть дополнен инструкциями.

Пример оформления файла README может быть таким:

```
1  # Overview
2
3  Report on LabRabota9.
4
5  # Usage
6
7  \\ Paste link on repository instead of "link" and "folders/branches"
8
9  To check, please, preview report by <<link>> and projects in
   <<folders/branches>>.
10
11 # Author
12
13 Your name and group number.
14
15 # Additional Notes
16
17 \\ Copy and paste link on your repository, as example:
18
19 https://github.com/maryiad/lab9-gr12a-david
```

2.1.2.2 Защита отчета по лабораторной работе

Каждая лабораторная работа содержит тексты задач и контрольные вопросы, ответы на которые проверяются преподавателем при приёме работы у студента.

Выполнение студентом лабораторной работы и сдача её результатов преподавателю происходит следующим образом:

1. Студент выполняет разработку программ.

В ходе разработки студент обязан следовать указаниям к данной задаче (в случае их наличия). Исходные тексты программ следует разрабатывать в соответствии с требованиями к оформлению для программ на языке C++.

2. Студент выполняет самостоятельную проверку исходного текста каждой разработанной программы и правильности её работы, а также свои знания по теме лабораторной работы.

Исходные тексты программ должны соответствовать требованиям к оформлению, приведённым в приложении. Недопустимо отсутствие в тексте программы следующих важных элементов оформления: спецификации про-

граммного файла и подпрограмм, а также отступов, показывающих структурную вложенность языковых конструкций.

Для проверки правильности работы программы студенту необходимо разработать набор тестов и на их основе провести тестирование программы. Тестовый набор должен включать примеры входных и выходных данных, приведённые в тексте задачи, а также тесты, разработанные студентом самостоятельно.

Самостоятельная проверка знаний по теме лабораторной работы выполняется с помощью контрольных вопросов и заданий, приведённых в конце текста лабораторной работы.

3. Студент защищает разработанные программы. Защита заключается в том, что студент должен ответить на вопросы преподавателя, касающиеся разработанной программы, и контрольные вопросы.

К защите необходимо представить исходные тексты программ, оформленных в соответствии с требованиями, исходные коды модульных тестов (unit-тестов).

При защите лабораторной работы студент демонстрирует исходный код теста (описание входных данных и соответствующих им выходных данных), описание выходных данных, полученных при запуске программы на данном тесте, и отметку о прохождении теста. Тест считается пройденным, если действительные результаты работы программы совпали с ожидаемыми.

Программы, не прошедшие тестирование, к защите не принимаются. В случае неверной работы программы хотя бы на одном тесте студент обязан выполнить отладку программы для поиска и устранения ошибки.


2.2 Задания

2.2.1 Задание 1


Цель — установить Aurora SDK, включая Aurora IDE, Build Engine и эмулятор Аврора ОС, познакомиться с примерами проектов и запустить приложение из примера проекта в эмуляторе Аврора ОС.

1. Изучить рекомендации по установке Aurora SDK (см. 1.2. «Установка Aurora SDK») и на сайте портала разработчиков Аврора ☞ «Аврора SDK».

2. Скачать онлайн или офлайн версию Aurora SDK с сайта портала разработчиков Аврора ☞ «Загрузка Аврора SDK». При наличии проблем

с загрузкой установочного пакета скачать версию Aurora SDK 5.0.1.27 с Яндекса.диска  <https://disk.yandex.ru/d/b0QptN2IQPTjoA>.

3. Изучить примеры предустановленных проектов мобильных приложений, например **Geolocation** и **Camera Image Capture**, в среде Aurora IDE (см. 1.4. «Сборка проекта мобильного приложения и запуск в ОС Аврора») или загрузить из репозитория, которые опубликованы на странице  Проекты с открытым исходным кодом (см. 1.3. «Примеры проектов мобильных приложений для ОС Аврора»).

4. Изучить пример проектирования приложения Telegram  «Telegram UI»

5. Изучить материалы по сборке проектов (см. 1.4. «Сборка проекта мобильного приложения и запуск в ОС Аврора»). Изменить конфигурацию любого проекта из рассмотренных, указав в настройках выпуска виртуальную машину, развернутой в Вашем окружении, выполнить сборку и запуск приложения в виртуальной машине.

2.2.2 Задание 2

Реализовать на языке C++ мобильное приложение «Угадай число» для ОС Аврора по аналогии с приложением для ОС Android. Требования к проекту:

- Структура приложения должна соответствовать одному из архитектурных шаблонов **MVC** или **MVVM**.
- Приложение должно быть локализовано на три языка, например русский, белорусский и английский.
- Включает обработку исключительных ситуаций с выводом сообщений об ошибках (в консоль).
- Приложение хранит настройки:
 - счетчик — количество запусков приложения;
 - уникальный ID приложения, генерируемый при первом запуске;
 - настройки пользователя (количество попыток, набранных баллов лучший результат и др.);
 - данные учетной записи пользователя.
- В диалоговом окне выводит результаты текущей попытки.

- Содержит анимацию элементов пользовательского интерфейса (на выбор).
- При сборке проекта запускаются модульные тесты (не менее трех тестов).

2.2.3 Задание 3

Реализовать на языке C++ мобильное приложение для ОС Аврора с хранением данных в БД SQLite согласно заданию 4.4 из лабораторной работы 6 и Вашему варианту. Добавить в проект модульные тесты (не менее трех).

2.3 Контрольные вопросы

1. Опишите основные этапы установки и развертывания Aurora SDK?
2. Какую роль выполняет Build Engine в Aurora SDK?
3. Приведите пример создания форматированного текста на языке QML?
4. Приведите примеры `TextField` и обработки ввода.
5. Приведите пример анимации элементов на языке QML.
6. Приведите пример создания меню в приложении для ОС Аврора.
7. Приведите пример создания диалогового окна в приложении для ОС Аврора.
8. Охарактеризуйте `QObject`.
9. Перечислите особенности сигналов и слотов.
10. Какой метод используется для выполнения SQL-запросов в мобильном приложении для ОС Аврора?