МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

ОТЧЕТ

по учебной вычислительной (ознакомительной) практике

Фоминой Марии Игоревны

студента 1 курса, 5 группы

специальность

«Прикладная математика»

Руководитель практики:

ассистент

В. Н. Романюк

Минск, 2024

# ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc171256213)

[Введение 4](#_Toc171256214)

[Глава 1 Теоретические основы 5](#_Toc171256215)

[1.1 Общая характеристика курса «Контроль версий с помощью Git» 5](#_Toc171256216)

[1.1.1 Настройка git и базовые операции 5](#_Toc171256217)

[1.1.2 Внутреннее устройство git 5](#_Toc171256218)

[1.1.3 Отмена действий в Git 7](#_Toc171256219)

[1.1.4 Ветки и merge 7](#_Toc171256220)

[1.1.5 Stash, tags, remotes 7](#_Toc171256221)

[1.1.6 Завершение курса 8](#_Toc171256222)

[1.2 Общая характеристика курса «Introduction to GitHub» 8](#_Toc171256223)

[1.2.1 Создать ветку 8](#_Toc171256224)

[1.2.2 Зафиксировать файл 9](#_Toc171256225)

[1.2.3 Открыть запрос на извлечение 9](#_Toc171256226)

[1.2.4 Объединить запрос на извлечение 10](#_Toc171256227)

[1.2.5 Результат 10](#_Toc171256228)

[1.3 Общая характеристика курса «Communicate using Markdown» 10](#_Toc171256229)

[1.3.1 Добавить заголовок 10](#_Toc171256230)

[1.3.2 Добавить изображение 11](#_Toc171256231)

[1.3.3 Добавить пример кода 11](#_Toc171256232)

[1.3.4 Составить список задач 11](#_Toc171256233)

[1.3.5 Окончание курса 12](#_Toc171256234)

[1.4 Выводы к главе 1 12](#_Toc171256235)

[Глава 2 Практическая часть курса 13](#_Toc171256236)

[2.1 Решение задач 13](#_Toc171256237)

[2.1.1 Лабораторная работа номер 1 13](#_Toc171256238)

[2.1.1.1 Работа с командной строкой в Linux 13](#_Toc171256239)

[2.1.1.2 Работа с GitHub 14](#_Toc171256240)

[2.1.1.3 Работа с SQLite 14](#_Toc171256241)

[2.1.2 Лабораторная работа номер 2 16](#_Toc171256242)

[2.1.2.1 Разработка необходимых таблиц, связи таблиц между собой и наполнения каждой таблицы. 17](#_Toc171256243)

[2.1.2.2 Создание базы данных с помощью DB Browser 18](#_Toc171256244)

[2.1.2.3Реализация работы с базой данных через QT. Создание интерфейса для простейших операций над базами данных. 18](#_Toc171256245)

[2.1.2.4 Написание валидатора 19](#_Toc171256246)

[2.1.2.5Написание теста к валидатору 20](#_Toc171256247)

[2.1.3 Лабораторная работа номер 3 21](#_Toc171256248)

[2.1.3.1 Условие задания 21](#_Toc171256249)

[2.1.3.2 Код программы 22](#_Toc171256250)

[2.1.3.3 Демонстрация работы 22](#_Toc171256251)

[2.2 Дополнительный материал 23](#_Toc171256252)

[2.3 Выводы к главе 2 23](#_Toc171256253)

[2.3.1Вывод по лабораторной работе номер 1 23](#_Toc171256254)

[2.3.2 Вывод по лабораторной работе номер 2 24](#_Toc171256255)

[2.3.3 Вывод по лабораторной работе номер 3 24](#_Toc171256256)

[Глава 3 Индивидуальный проект 25](#_Toc171256257)

[3.1 Функциональное назначение 25](#_Toc171256258)

[3.2 Структура проекта 25](#_Toc171256259)

[3.3 Интерфейс приложения 25](#_Toc171256260)

[3.4 Описание проекта 27](#_Toc171256261)

[3.5 Выводы к главе 3 27](#_Toc171256262)

[Заключение 28](#_Toc171256263)

[Список использованных источников 29](#_Toc171256264)

# Введение

Целью учебной вычислительной (ознакомительной) практики является закрепление полученных знаний за первый год обучения посредством выполнения специальных учебных заданий, участия в работе над командным проектом.

Задачи практики:

* Знакомство с IT компанией, возможностями обучения и повышения квалификации, знакомство с разными направлениями профессиональной подготовки IT специалистов.
* Изучение инструментов для совместной разработки проектов и контроля версий.
* Подготовка и настройка программных сред и средств тестирования для прохождения тренинга по выбранному направлению.
* Выполнение стандартных заданий.
* Выполнение заданий повышенного уровня.
* Разработка приложение «игра Арканоид»
* Оформление отчета.

За время практических занятий были изучены db Browser, ОС Linux, продолжилась работа с графикой в Qt. Они все связаны между собой и совместное использование выводит на новый уровень проектов и упрощает работу. Также большая часть практики, теоретической части в частности, была посвящена работе с git и GitHub. Знания и умения работать с ними очень актуально для современных разработчиков и точно пригодятся для дальнейшей работы.

# Глава 1 Теоретические основы

## Общая характеристика курса «Контроль версий с помощью Git»

Перед началом практической работы был пройден курс от Epam «Контроль версий с помощью Git». Основная цель курса — подготовить фундамент для изучения программирования, эффективного использования современных инструментов разработчика. Данный курс дает общее представление о работе с git и GitHub, а также учит простейшим командам для git.

## 1.1.1 Настройка git и базовые операции

В данном модуле была получена информация что такое git, зачем он нужен и где используется, как его установить и подключить к GitHub. Так же было рассказано какие типы и виды систем операционного контроля существуют.

**lock-modify-unlock strategy**(пока один правит файл, никто другой не может им пользоваться)

**copy-modify-merge strategy**(все могут править все файлы в любое время, разрешение конфликта)

Системы операционного контроля по механизму работы с сервером:

1)centralized

централизованная(на компьютерах людей только файлы)

2)distributed

распределенная(у каждого пользователя помимо своего файла есть история версий, версионная база=репозиторий)

Для соединения GitHub c git был необходим публичный ключ (id\_rsa).

Его нужно отправить владельцу репозитория для получения прав работы. Private-ключ никому не стоит показывать, т.к. он может быть использован для работы с проектами от вашего имени.

Также было рассказано о базовых командах для git Bash:

* git log — история коммитов
* git push — синхронизировать свой локальный репозиторий с удаленным
* git fetch — синхронизация двух репозиториев, но не устанавливает новый контент файла в файловой системе
* git pull — git fetch + git merge

## 1.1.2 Внутреннее устройство git

В данном модуле знакомят с понятиями git GUI &gitk и дают понмание, что хранится в папке .git.

Git GUI (Graphical User Interface) (Рисунок 1) является графической оболочкой для командной строки Git. Она предоставляет пользователям визуальный интерфейс для выполнения различных операций Git.

gitk — это встроенный в Git графический интерфейс пользователя, который позволяет визуально просматривать историю коммитов репозитория.

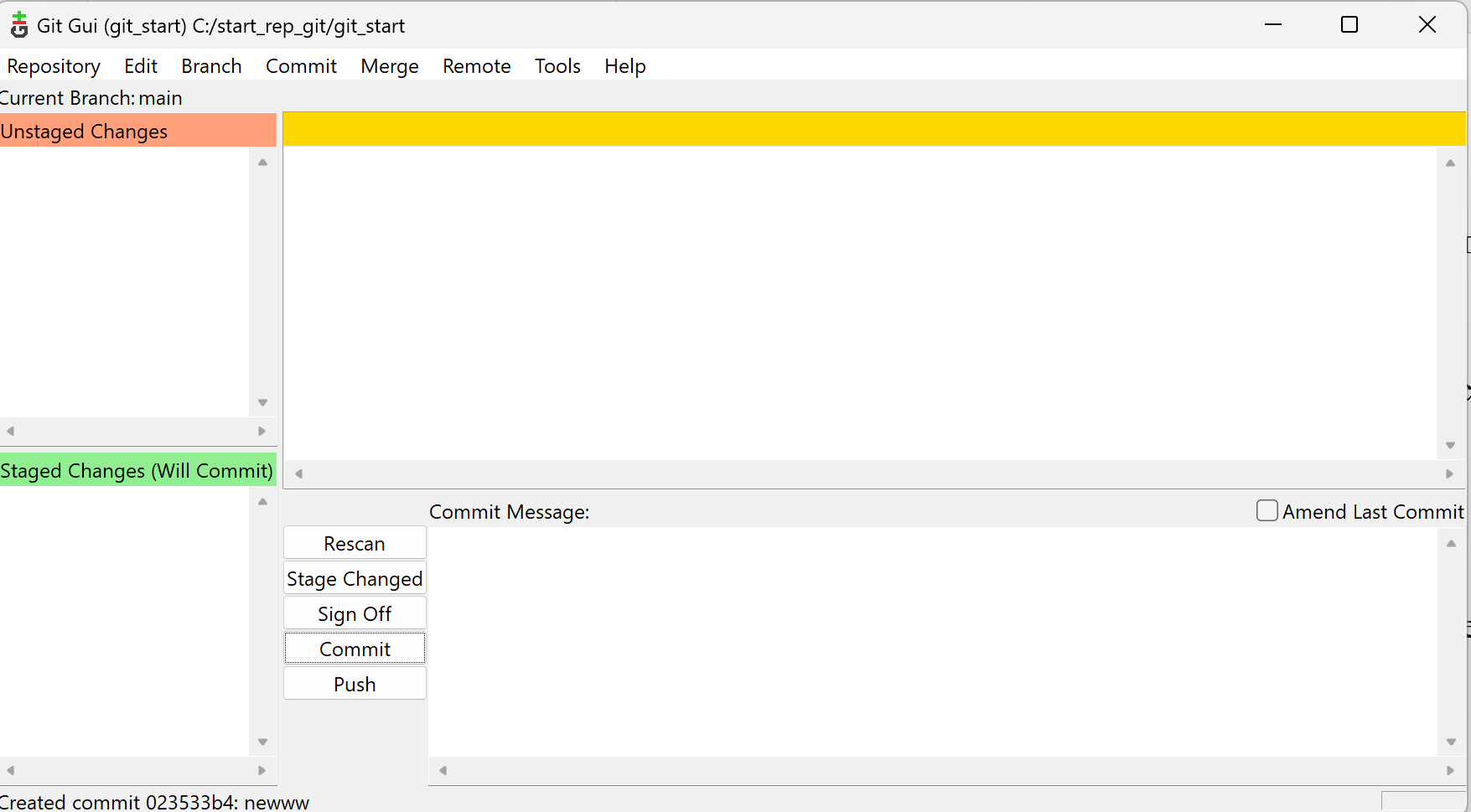
****

Рисунок 1—git GUI&

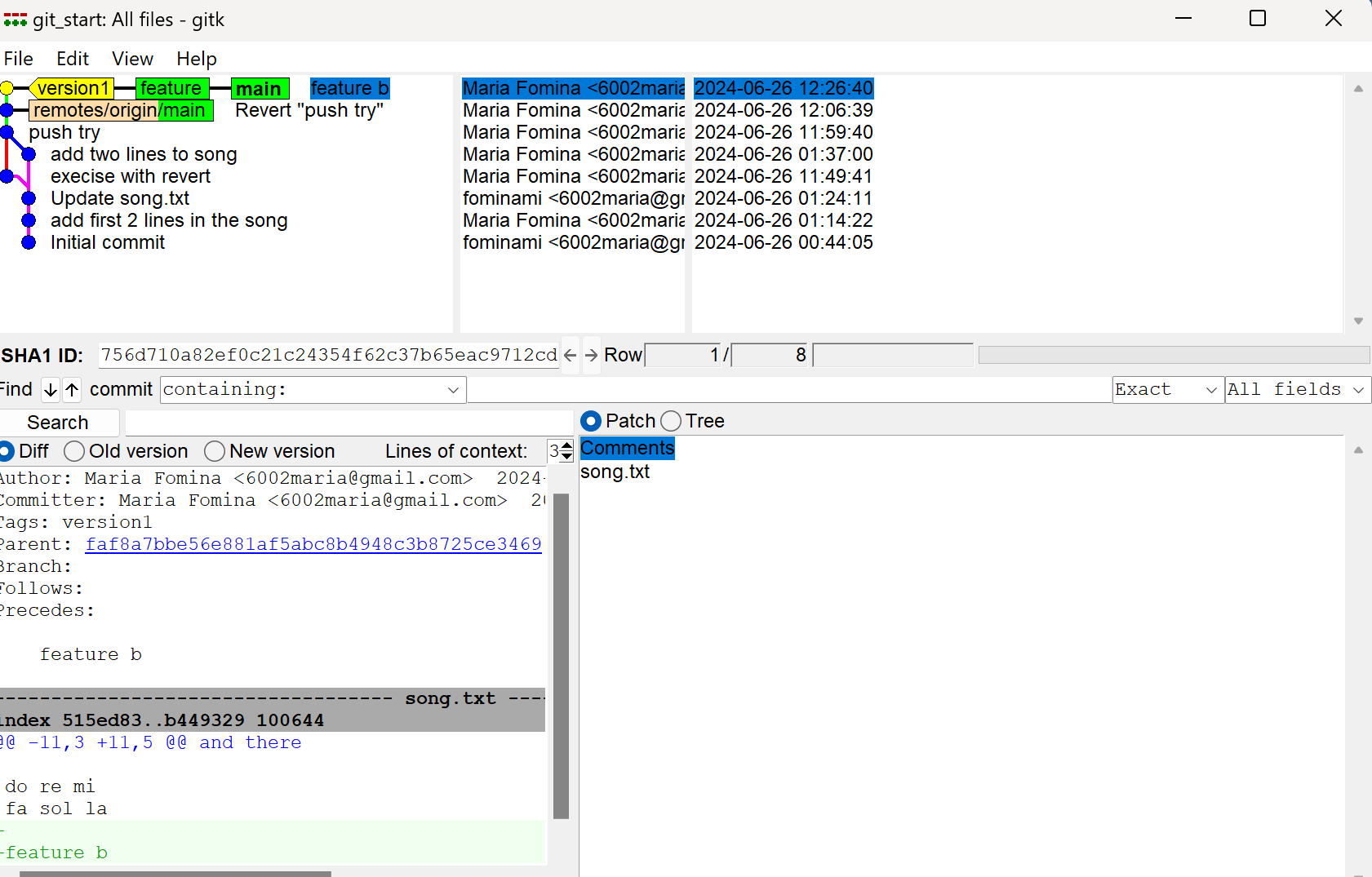
****

Рисунок 2—gitk

Касательно того, что хранится в папке .git:

remote — те адреса, те состояния коммитов, которые находятся на remote репозиториях(удаленных репозиториях)

index — здесь находится промежуточная информация, которую мы собираемся коммитать

В Git все построено на графах.

3 фундаментальных вида объекта в Git:

1. коммит
2. дерево
3. блоб (binary large object) - минимальная атомарная ячейка хранения информации в Git

Каждый коммит может содержать в себе как измененные файлы, так и директории, в которых эти файлы хранятся.

Коммиты ссылаются на деревья, а уже в деревьях указываются имена файлов, которые мы коммитаем, и названия директориев, в которых они хранятся(т.е. пути).

## 1.1.3 Отмена действий в Git

git checkout — отменить изменения в одном файле

git checkout — отменить изменения во всех файлах от текущей директории и ниже

git clean — удалить новые файлы(о которых у Git нет данных)

git reset — удаление коммита. Может быть два варианта git resert –soft(до состояния add), git resert –mixed( до начального состояния)

git revert — используется для отмены (отката) одного или нескольких ранее совершенных коммитов. Создается новый коммит, который отменяет предыдущий

.gitignore — папка, которая делает файлы «невидимыми» для git

## 1.1.4 Ветки и merge

Ветвление необходимо для параллельной работы и безопасного хранения финальной готовой версии.

Merge — слияние, это процесс объединения двух или более ветвей разработки в одну.

Conflict solving — cитуация, где 2 бранча имеют разный контент

Rebase — изменение базы, разработчик берет последние коммиты, не коммитя своей работы. Другими словами он изменяет базу своей ветки на более свежую.

Cherry-pick — копирование коммита в другое место.

## 1.1.5 Stash, tags, remotes

Tags — текстовая метка при коммите

stashing — временное хранилище для файла

git stash pop(возвращает данные из стеша в файловую систему, но удаляет сам стеш)

git stash apply(возвращает данные из стеша в файловую систему, но при этом не удаляет сам стеш)

Remotes:

Ctrl insert копировать

remote -v просмотреть существующий репозиторий

remote remove - удалить существующий

remote add - добавить новый

Branching strategies

1. centralized
2. feature-branch workflow
3. gitflow
4. integration manager workflow
5. dictator and lieutenants workflow
6. forking workflow

## 1.1.6 Завершение курса

После прохождения курса были выполнены практические задания и изучены полезные ссылки

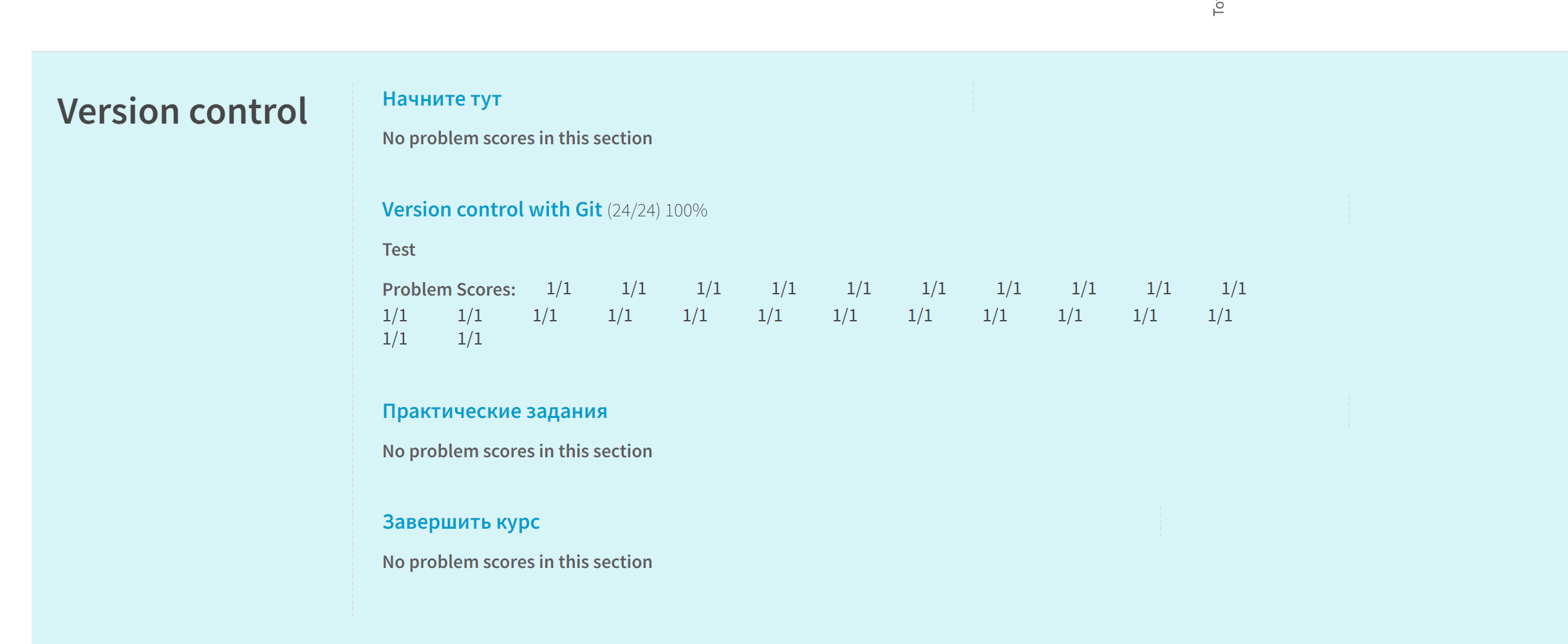


Рисунок 3— завершение курса

Сверху прикреплено подтверждение об окончании курса.

## Общая характеристика курса «Introduction to GitHub»

Цель курса: изучение основ работы с GitHub

Данный курс находится на GitHub и контролирует правильность выполнения каждого шага и дает доступ к следующему заданию только после верного результата текущего.

GitHub — это платформа для совместной работы, которая использует git для управления версиями. GitHub — популярное место для обмена и внесения вклада в программное обеспечение с открытым исходным кодом .

## Создать ветку

Алгоритм:

Перейти на вкладку **< > Код** в верхнем меню репозитория.

Нажать на раскрывающийся список **основной ветви.**

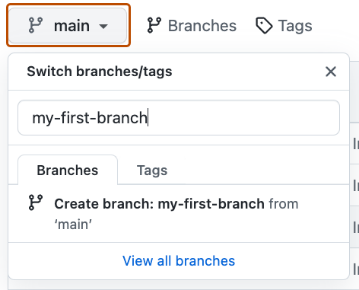
[](https://github.com/fominami/skills-introduction-to-github/blob/main/images/main-branch-dropdown.png)

Рисунок 4 —демонстрация шага

Далее назвать ветку

Нажать **Создать ветку**, чтобы создать свою ветку.

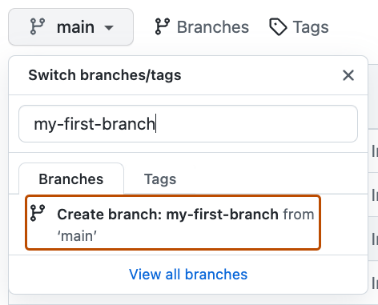
[](https://github.com/fominami/skills-introduction-to-github/blob/main/images/create-branch-button.png)

Рисунок 5 — демострация

Ветка автоматически переключится на ту, которую только что создали. **Основная** выпадающая панель ветки отразит новую ветку и отобразит ее имя.

## Зафиксировать файл

Алгоритм:

* На вкладке **< > Код** в меню заголовка репозитория убедиться, что вы находитесь на новой ветке
* Выбрать раскрывающийся список **Добавить файл и нажать Создать новый файл**.
* Нажать **Commit changes...** в правом верхнем углу над полем содержимого. Для коммитов можно ввести короткое сообщение о коммите, описывающее внесенные изменения. Это сообщение поможет другим узнать, что включено в коммит.
* Нажать **«Сохранить изменения»**

## Открыть запрос на извлечение

Алгоритм:

* Нажать на вкладку **«Запросы на извлечение»** в верхнем меню репозитория.
* Нажать **«Новый запрос на извлечение»**.
* В раскрывающемся списке **base:** убедиться, что выбран **параметр main.**
* Выбрать **раскрывающийся список «**Сравнить» и нажать на ветку
* Нажать **Создать запрос на извлечение**.
* Ввести заголовок для запроса на извлечение.
* Нажать **Создать запрос на извлечение**.

## Объединить запрос на извлечение

Алгоритм:

* Нажать **«Объединить запрос на извлечение»** .
* Нажать « **Подтвердить объединение »** .
* После того, как ветка была объединена, она не нужна. Для удаления ветки, нажать **Удалить ветку**.

## Результат

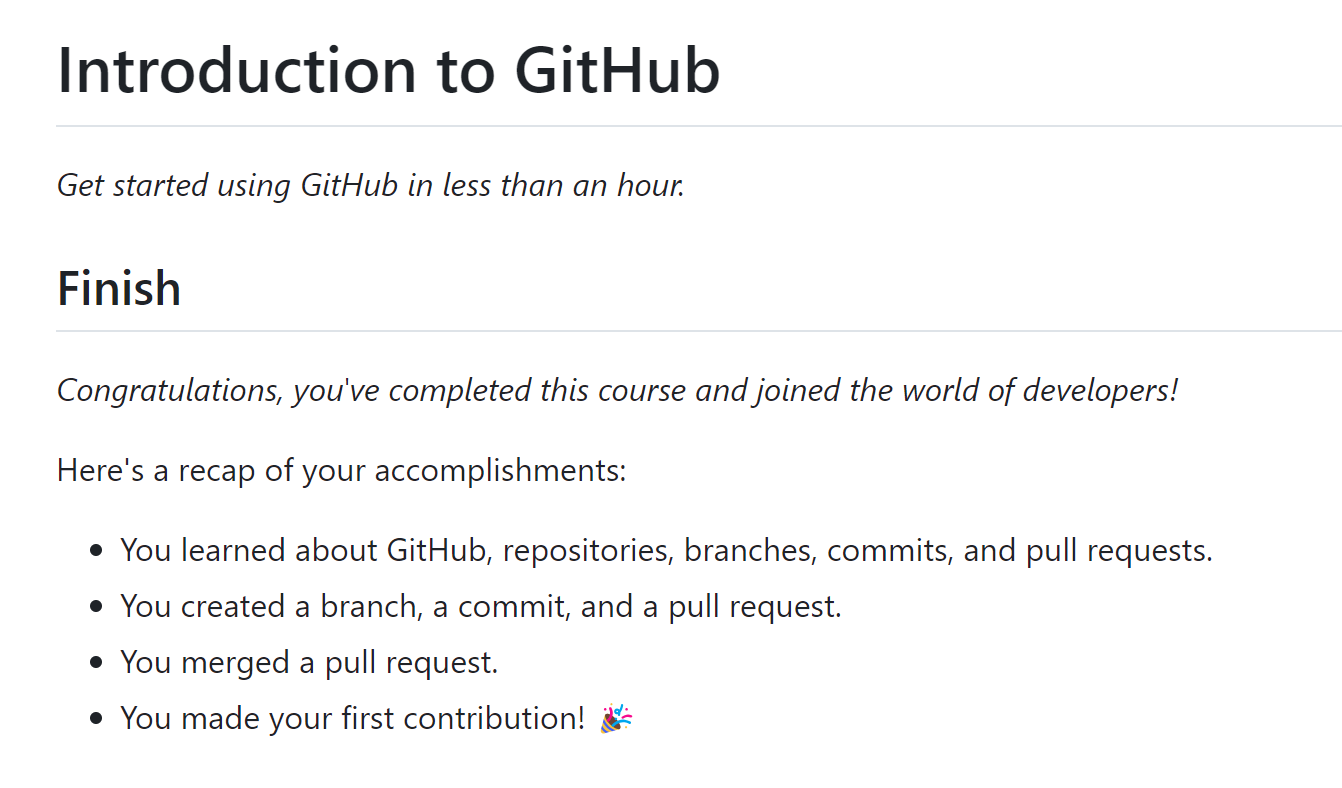


Рисунок 6 — подтверждение

Подтверждение об успешном окончании

## Общая характеристика курса «Communicate using Markdown»

Markdown — это [облегченный синтаксис](https://docs.github.com/github/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax) для общения на GitHub. Вы можете форматировать текст, добавляя заголовки, списки, **полужирный шрифт** , курсив , таблицы и многие другие стили. Вы можете использовать Markdown в большинстве мест на GitHub:

* Комментарии по [проблемам](https://docs.github.com/issues/tracking-your-work-with-issues/about-issues) , [запросы на включение изменений](https://docs.github.com/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests) и [обсуждения](https://docs.github.com/discussions/collaborating-with-your-community-using-discussions/about-discussions)
* Файлы с расширением .md или .markdown
* Обмен фрагментами текста в [Gists](https://docs.github.com/github/writing-on-github/editing-and-sharing-content-with-gists/creating-gists)

## Добавить заголовок

Алгоритм:

* В pull request перейти к файлу .md .
* Выбрать **«Изменить файл»** в меню с тремя точками **...** в правом верхнем углу окна просмотра файлов.
* На вкладке **«Изменить файл»** добавить , #а затем **пробел** , перед любым содержимым. Получается заголовок H1. Можно добавить больше заголовков, используя от одного до шести #символов с **пробелом** .
* Над новым контентом нажать **«Предварительный просмотр»** .
* В нижней части страницы ввести короткое, содержательное сообщение о фиксации, описывающее изменение, которое внесены в файл.
* Нажать  **«Принять изменения»** .

## Добавить изображение

Изображение добавляется следующим образом:

![Image of Yaktocat](<https://octodex.github.com/images/yaktocat.png>)

Описательный текст в квадратных скобках читается вслух для людей, использующих программы чтения с экрана. Он также показывается в тех случаях, когда изображение не отображается, например, при плохом соединении.

## Добавить пример кода

При добавлении кода он берется в ‘‘‘. А отображается после сохранения в привычном виде.

## Составить список задач

Список задач создает флажки для отметки. Они очень полезны для отслеживания проблем и запросов на извлечение.

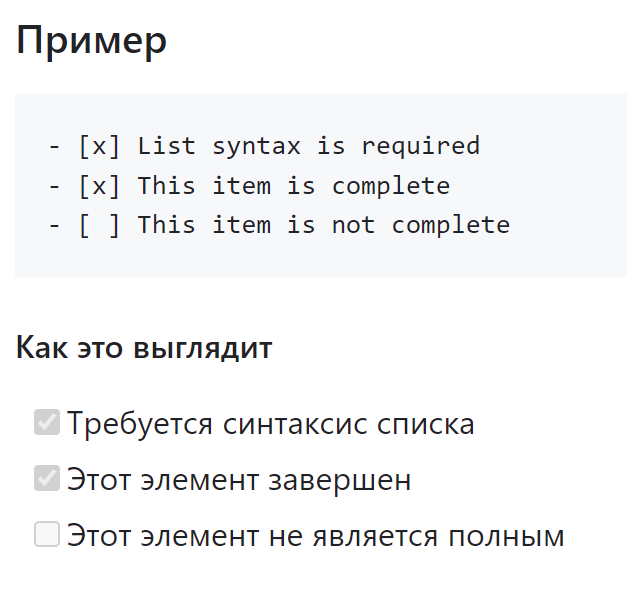


Рисунок 7 —пример создания таблицы

Из рисунка видно, что необходимо перед каждой строкой ставить –[]

## Окончание курса

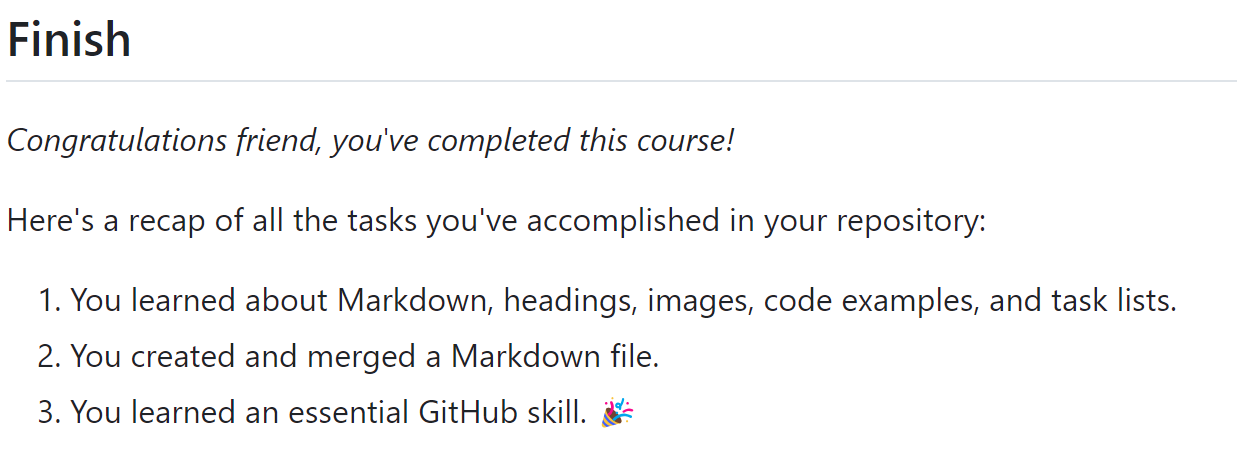


Рисунок 8 —подтверждение об окончании курса

Последняя страница курса, которая свидетельствует об окончании курса.

## Выводы к главе 1

Вывод разделен на три части, каждая из которых соответствует одному из трех пройденных курсов

Вывод по курсу «Контроль версий с помощью Git»:

Данный курс помог в установке git, помог разобраться в работе с git, создать свой первый репозиторий и освоить простейшие команды необходимые для качественной работы.

В ходе курса была создана страница на GitHub.Мой логин: fominami, почта: [6002maria@gmail.com](mailto:6002maria@gmail.com).

Вывод по курсу «Introduction to GitHub»:

Курс был направлен на обучение простейшим действиям в самом GitHub уже после создания репозитория и добавления необходимых файлов. Умение создавать ветку, фиксировать файл и т.д. необходимы для грамотной работы в команде и просто корректном редактировании репозитория.

Вывод по курсу «Communicate using Markdown»:

Данный курс дает понимание как работать с файлом .md и о сути этого файла в общем. Действительно, после прохождения курса становится ясно, что markdown это способ для взаимодействия и пояснения информации содержащейся в репозитории. Данный файл имеет множество функции, который помогают структурировать информацию.

# Глава 2 Практическая часть курса

## 2.1 Решение задач

## 2.1.1 Лабораторная работа номер 1

## 2.1.1.1 Работа с командной строкой в Linux

Цель работы: изучить основные команды работы с файлами и папками в командной строке ОС Linux

Основные команды

* **cd** — возврат в домашнюю директорию текущего пользователя;
* **ls** — вывод списка файлов и директорий в текущем каталоге
* **sudo** — запустить команду с правами администратора.
* **mkdir** — создание директории
* **cp** *—* копирование файлов и директорий
* **mv** — перемещение и переименование файлов и директорий
* **rm** — удаление файлов или директорий
* **find** —поиск в указанной папке файлов, имя которых содержит заданные символы.
* **clear** — очищение окна терминала от всех сообщений.
* **history** — отображение всех недавно введённых команды. Кроме того, вы можете переключаться между недавними с помощью клавиш «Вверх» и «Вниз»
* **сat** — вывод содержимого файла
* **more** — программа просмотра текстовых файлов в терминале. Она позволяет постранично прокручивать содержимое файла.
* **less** —это более расширенная программа просмотра текстовых файлов в терминале.Она позволяет как постранично прокручивать содержимое файла, так и переходить к любому месту в файле.
* **nano** и **vim —**текстовые редакторы для редактирования файлов.
* **!!** — повторить последнюю программу

Изучение структуры команд на примере команды echo 'Hello World'

Команда **echo** выводит сообщение на консоль.

Пример выполнения:

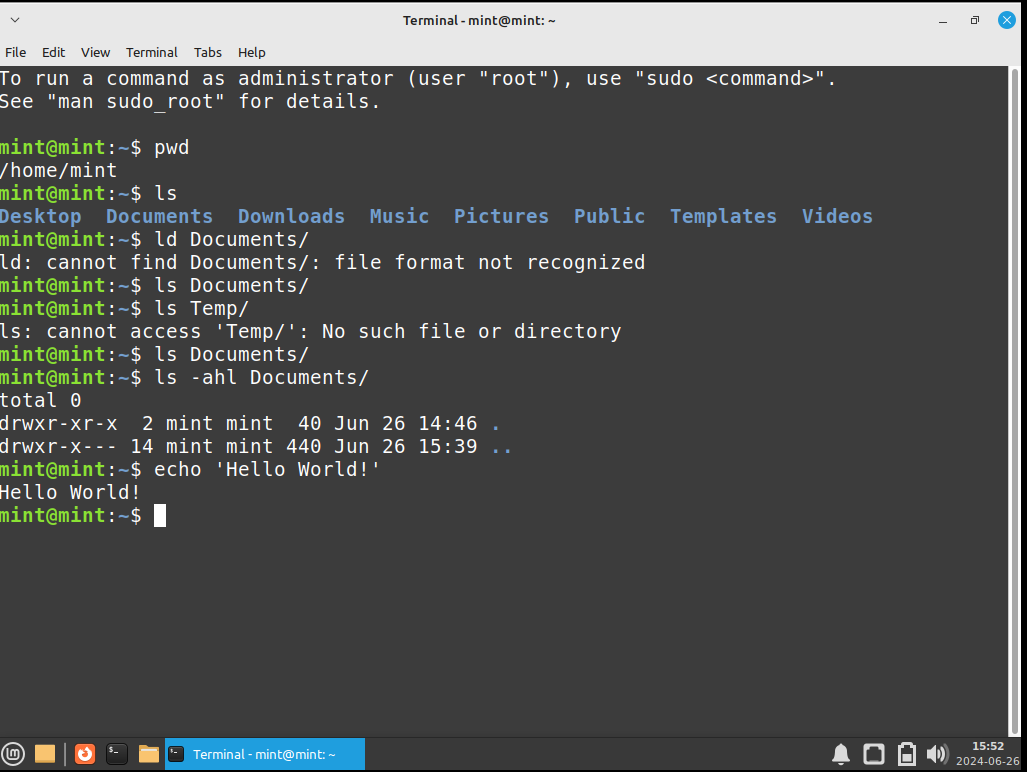


Рисунок 9 —Пример выполнения

На фото видна команда и результат выполнения.

Создание простейшего приложения на языке С++в ОС Linux

Для создания приложения необходимо выполнить следующие действия:

1. Установить компилятор g++
   1. sudo apt upgrate
   2. sudo apt install g++
2. Создать файл(в моем выполнении start.cpp). Работа с файлом производится через текстовый редактор vim или nano
   1. nano start.ccp
3. Скомпилировать файл
   1. g++ -o start start.cpp
4. Запустить
   1. ./start

Результат:

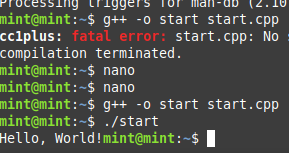
****

Рисунок 10 —результат выполнения

На рисунке видно, что файл был скомпилирован и успешно запустился.

## 2.1.1.2 Работа с GitHub

**Цель работы:** изучение основ работы с GitHub и особенностей работы Markdown

Отчет об этом пункте расположен в главе 1, пунктах 1.2 и 1.3

## 2.1.1.3 Работа с SQLite

**Цель работы:** изучить простейшие команды для работы с SQLite

Работать с SQLite можно как в консоли, так и в DB Browser. Далее представлена работа через DB Browser.

Выполнение команд и результаты:

* Select, insert, drop:

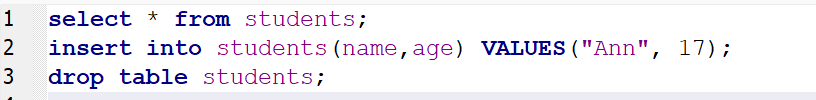


Рисунок 11 —Select, insert, drop

* Выборка данных с фильтрацией (условие where), если id=5:



Рисунок 12 —пример ввода

Результат:

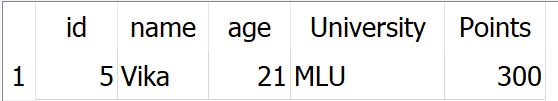


Рисунок 13— результат выполнения

* Выборка данных с фильтрацией (условие where) и с совпадением по маске,  
  например все записи, где имя объекта (согласно варианту) начинается на  
  первую букву вашего имени (инструкция like); 

Рисунок 14 — пример ввода

Результат:

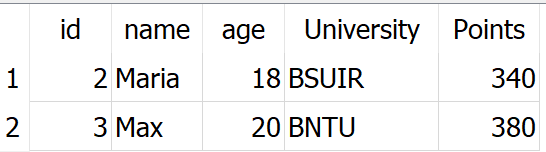


Рисунок 15 — результат выполнения

* Переименование таблицы (alter):



Рисунок 16 — пример ввода

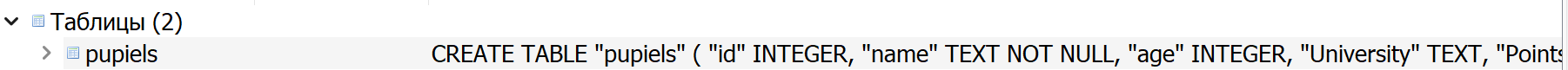
Результат:  


Рисунок 17— результат

* Обновление данных с использованием update

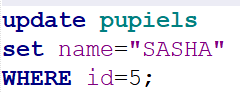


Рисунок 18 — пример ввода

Результат:

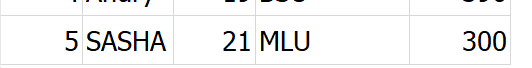


Рисунок 19 — результат

* Удаление строк по id и по названию объекта:



Рисунок 20 — пример ввода

Результат:

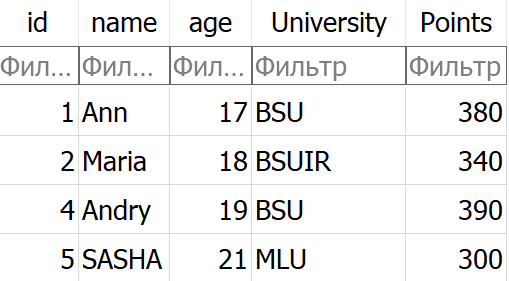


Рисунок 21 — результат

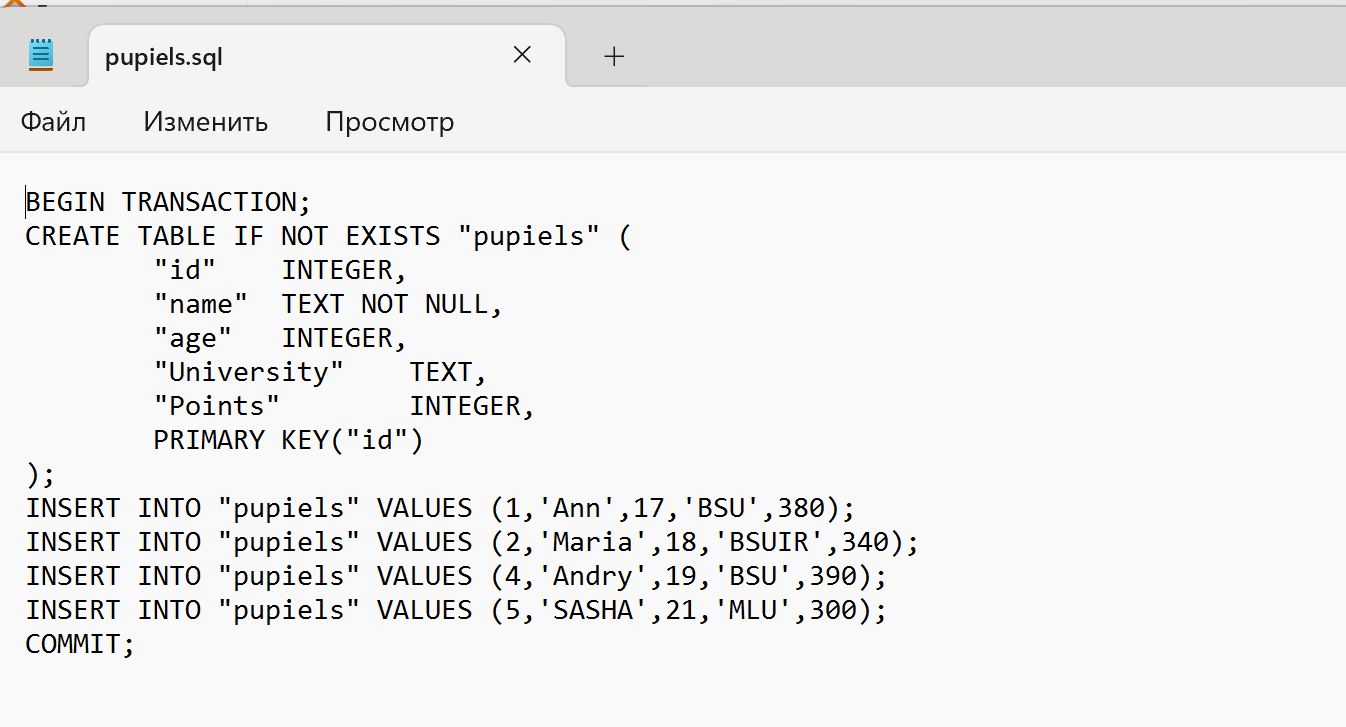


Рисунок 22 — пример .sql

.csv

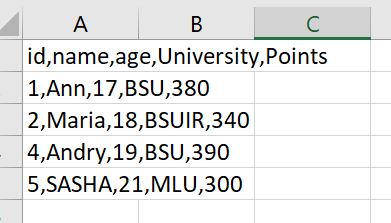


Рисунок 23 —пример .csv

Важной возможность является экспорт в файлы .sql .csv

DB Browser также имеет удобный интерфейс для сортировки. Каждый столбец имеет справа стрелочку, позволяющую менять порядок сортировки и обозначать приоритетный столбец для сортировки:

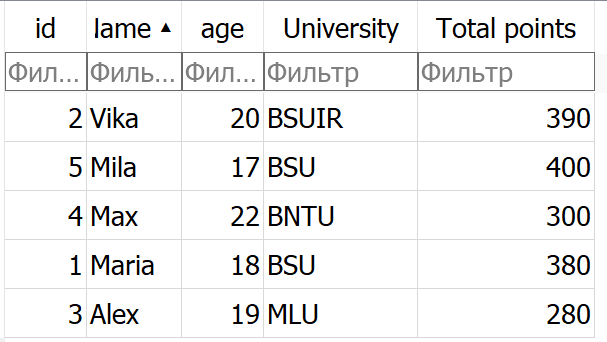


Рисунок 24 — демонстрация сортировки

## 2.1.2 Лабораторная работа номер 2

**Лабораторная работа №2**

**Цель работы:** создать приложение в Qt для работы с базой данных. Реализовать валидатор и написать к нему Unitest.

**Задача:**

Цветочная оранжерея выращивает различные виды цветов и продает на заказ  
составленные из них композиции. Каждая композиция имеет свое название и может состоять  
как из цветов одного вида, так и из цветов разного вида.

Дирекция оранжереи владеет информацией:  
**о цветах:** название цветка, сорт, стоимость одного цветка;  
**о композициях:** название композиции, название входящего в композицию цветка, количество единиц;  
**о выполнении заказов:** дата принятия заказа, название композиции,  
количество единиц, покупатель. **о покупателе:** логин, пароль

Необходимо выполнить:

1. Создать таблицы БД
2. Обеспечить с помощью операторов Select, Insert, Update, Delete обновление и выборку информации в указанных таблицах
3. Создать оконное приложение, которое позволяет манипулировать данными с использованием вышеупомянутых операторов.
4. Написать тесты для валидатора

**Свою работу я разделила на 5 частей:**

1. Разработка необходимых таблиц, связи таблиц между собой и наполнения каждой таблицы.
2. Создание базы данных с помощью DB Browser
3. Реализация работы с базой данных через QT. Создание интерфейса для простейших операций над базами данных
4. Написание валидатора
5. Написание теста к валидатору

## 2.1.2.1 Разработка необходимых таблиц, связи таблиц между собой и наполнения каждой таблицы.

Предварительная заготовка для таблиц выглядела следующим образом:

Flower(название text, сорт text, cost integer, id PK integer)

Compositions(id integer, name text, cost integer)

Flower\_Comp(id integer, id\_flower integer, id\_comp integer, count integer)

Order(id integer, date\_v text, id\_comp integer, count integer, user\_id integer, summa integer)

User(id integer, login text, password text)

Из заготовки видно, что таблицы Order и Flower\_Comp связаны с остальными таблицами путем указания id нужных строк.

Типы данных выбраны соответственно доступным в db Browser, так как дальнейшая работа будет проводится в нем.

## 2.1.2.2 Создание базы данных с помощью DB Browser

В db Browser создается 5 таблиц:

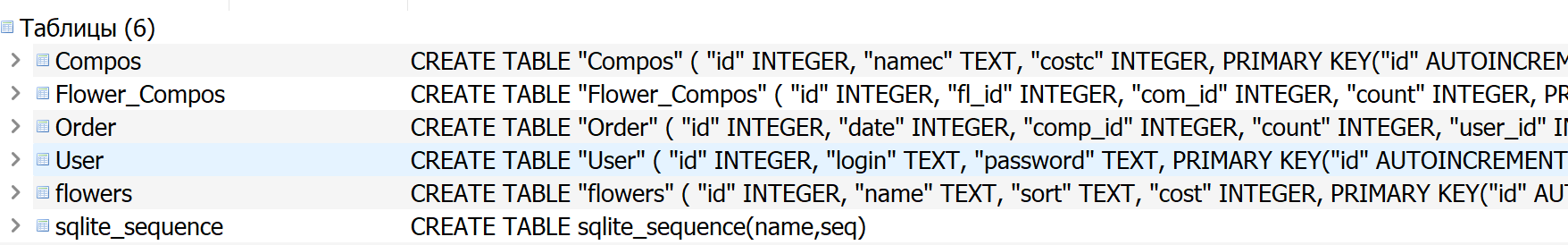


Рисунок 25 —-Таблицы SQLite

Далее таблицы были заполнены вручную. Пример:

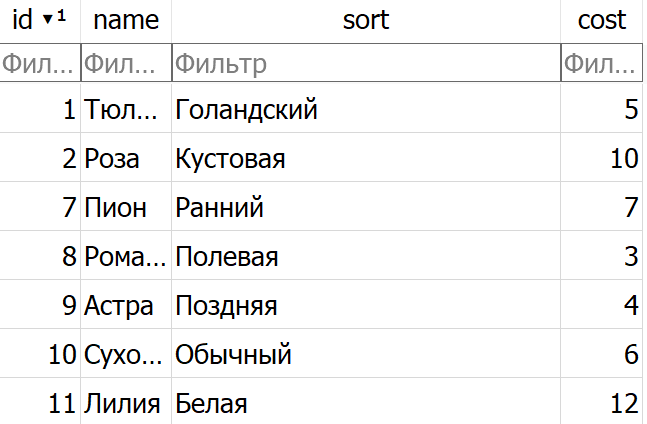


Рисунок 26 —пример заполнения таблицы

Далее файл был сохранен с расширение .db

## 2.1.2.3Реализация работы с базой данных через QT. Создание интерфейса для простейших операций над базами данных.

Основной задачей было создать удобный интерфейс для пользователя, который бы выполнял те же операции что и db Browser, однако пользователю не нужно было бы писать команды вручную.

Для того, чтобы подать информацию структурировано, использовались такие инструменты QT как: QSqlTableModel(вывод таблиц в текущем состоянии на выбор), GroupBox(для объединения элементов, которые решают общую задачу), TextBrowser и Label( для вывода указаний пользователю).

Вот финальный вид приложения:

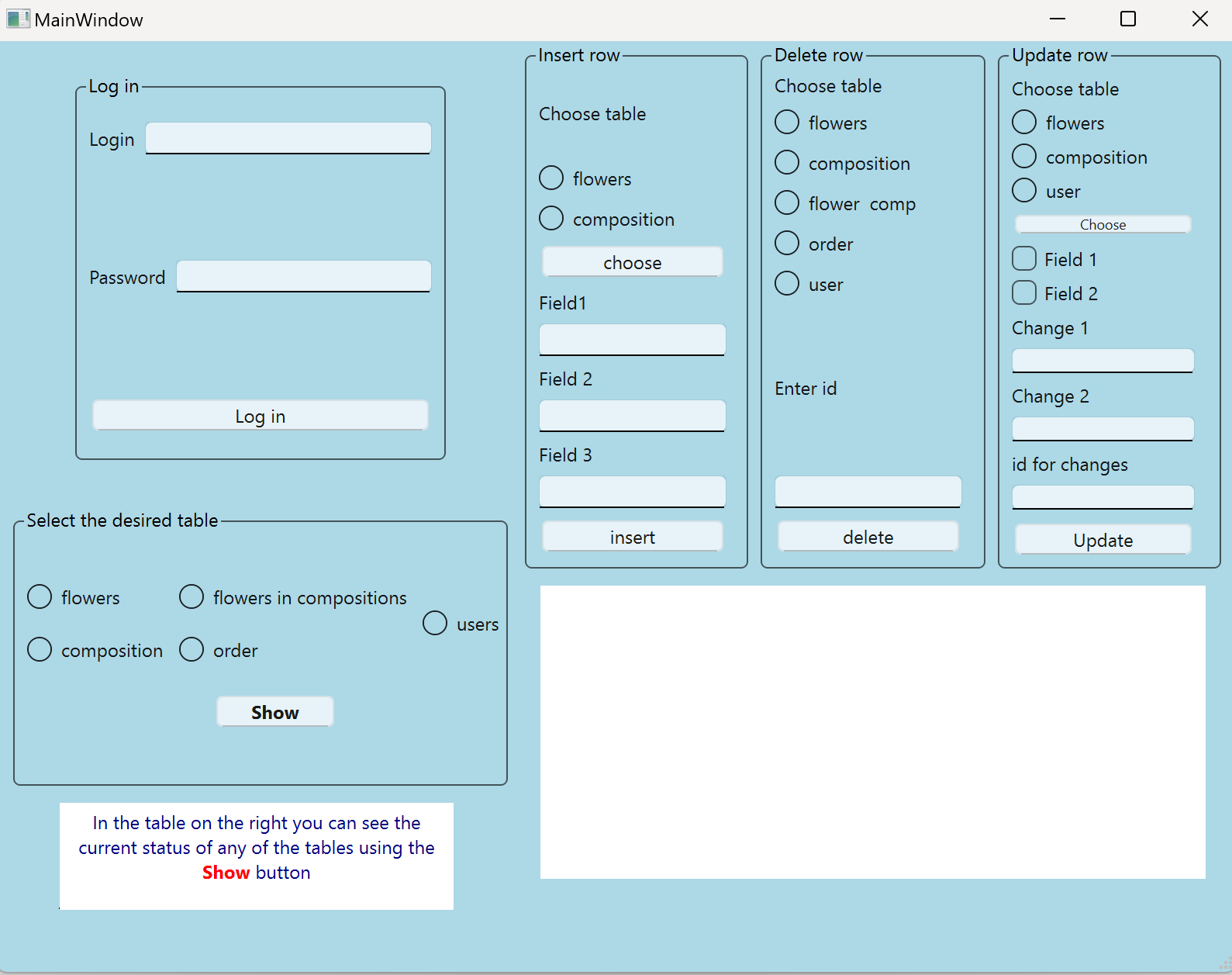


Рисунок 27 — финальный вид

Пользователь в каждом действии может выбрать таблицу, с которой он работает, и в соответствии с этим изменятся Label для упрощения ввода, т. к. каждая таблица имеет специфичные столбцы.

Пример такого изменения на операции Insert:

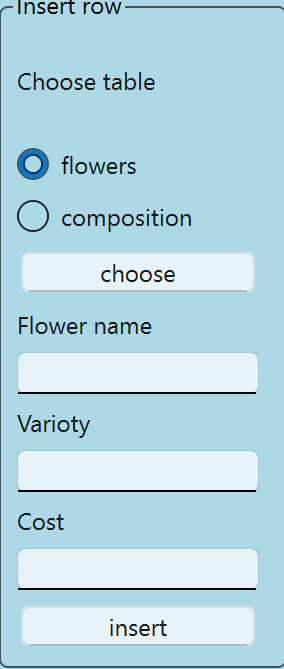


Рисунок 28 — пример операции insert

(изменение происходит после нажатия кнопки choose)

Для более глубокого разбора кода оставляю ссылку на GitHub:

<https://github.com/fominami/lab_2_flowers>

## 2.1.2.4 Написание валидатора

Валидатор был создан для операции добавления пользователя в БД. Чтобы добавить пользователя было необходимо ввести login и password. С помощью валидатора было ограничено количество символов для ввода и поставлены правила для создания надежного пароля. Для наглядности в случае подходящего логина и пароля lineEdit, куда вводились строки, подсвечивался зеленым, а в случае неудачи красным. Также при неверном вводе всегда показывались условия для правильного ввода.

Вот пример двух вариантов:

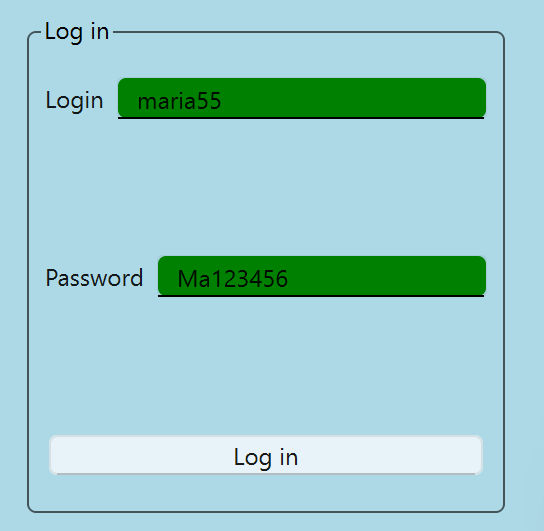


Рисунок 29 — правильный пароль

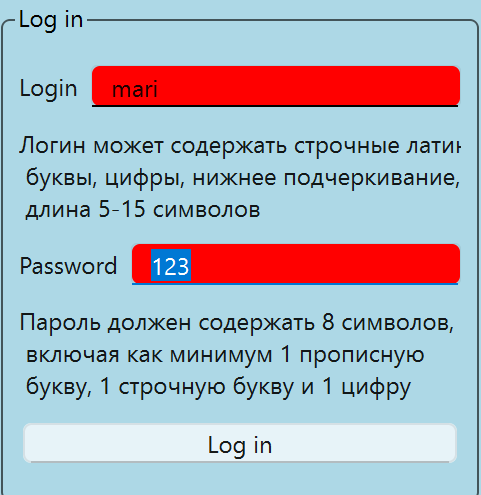


Рисунок 30 — неправильный пароль

Рисунки показывают, что условия ввода появляются только при неправильном вводе.

## 2.1.2.5Написание теста к валидатору

Тестирование проводилось с помощью Visual Studio, где был выбран шаблон google test. Валидатор, который нужно было протестировать, был написан как отдельный класс, поэтому мы подключали файл .h для дальнейшего подключения к тестам. Т. к. класс писался в Qt, он содержал библиотеки Qt, поэтому для качественного тестирования нужно было провести замены на эквивалентные элементы из Visual Studio: (QRegularExpression-> regex, QRegularExpressionMatch->regex\_match, QValidator::State->bool, QString->string)

Далее представлены тесты и их результаты для login и password:

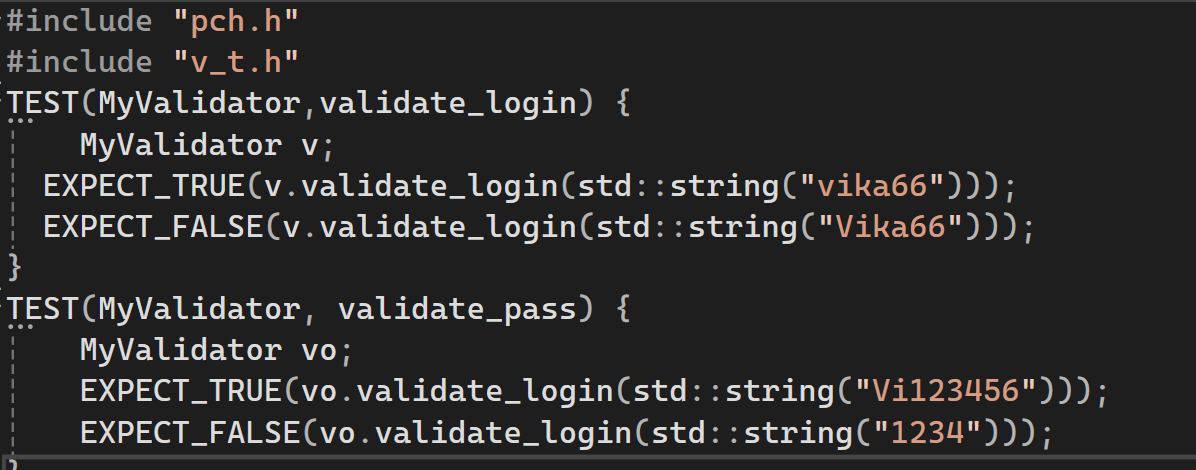


Рисунок 31 — код тестирования

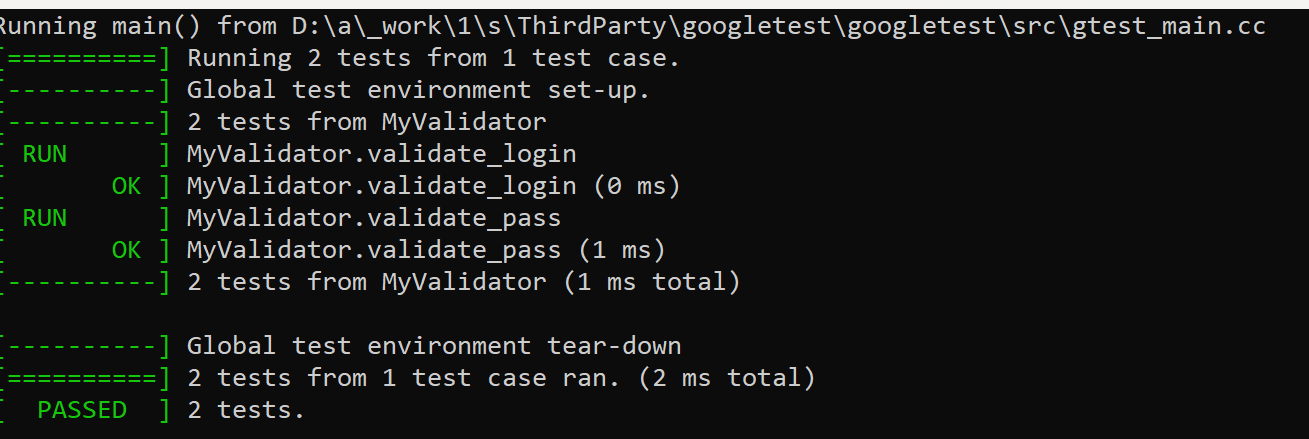


Рисунок 32 — результат тестирования

Данные два рисунка демонстрируют, что валидатор прошел тестирование.

## 2.1.3 Лабораторная работа номер 3

Цель лабораторной работы: создать аналог Paint на QT

## 2.1.3.1 Условие задания

• Прямоугольник, Треугольник, Эллипс - при активации любой из этих

кнопок следующее нажатие левой кнопки мыши в рабочей области

приводит к началу рисования соответствующей фигуры. Левой верхней

точкой фигуры является точка нажатия мыши, правой нижней - текущие

координаты мыши (эллипс и треугольник вписываются в прямоугольник с данными координатами). При отпускании левой кнопки мыши

рисование фигуры завершается, она добавляется в рабочую область. Если

до отпускания левой кнопки была нажата правая кнопка мыши или

кнопка Esc - рисование прерывается, и фигура не добавляется в рабочую

область.

• Связь - при активации данной кнопки клик на любой фигуре в рабочей

области начинает рисование связи (линии) от центра данной фигуры к

текущим координатам курсора. При повторном клике на другой фигуре в

рабочую область добавляется связь между первой и второй фигурами.

Если повторный клик был произведен не на фигуре, либо на начальной

фигуре - рисование связи прерывается. Рисование также прерывается при

нажатии правой кнопки мыши или нажатии Esc.

• Переместить - при активации данной кнопки нажатие левой кнопки на

любой фигуре в рабочей области приводит к изменению курсора на

«руку». При перемещении указателя мыши фигура также перемещается.

Если фигура имеет связи с другими фигурами, эти связи также должны

перемещаться. При отпускании левой кнопки мыши перемещение

фигуры завершается. Перемещение прерывается при нажатии правой

кнопки мыши или нажатии Esc.

• Удалить - при активации данной кнопки, клик на любой фигуре в рабочей

области приводит к удалению, вместе со всеми её связями.

• Загрузить, Сохранить - загрузка и сохранение рисунка в файл. Формат

файла не важен. После загрузки должна быть возможность продолжить

рисование (добавление, перемещение, удаление и т.д.) фигур.

!!!Не использовать QGraphicsltem!!!

## 2.1.3.2 Код программы

С кодом программы можно ознакомится перейдя по ссылке на GitHub: <https://github.com/fominami/paint_project_lab_3>

## 2.1.3.3 Демонстрация работы

В данном разделе показана работа программы при нажатии разных кнопок.

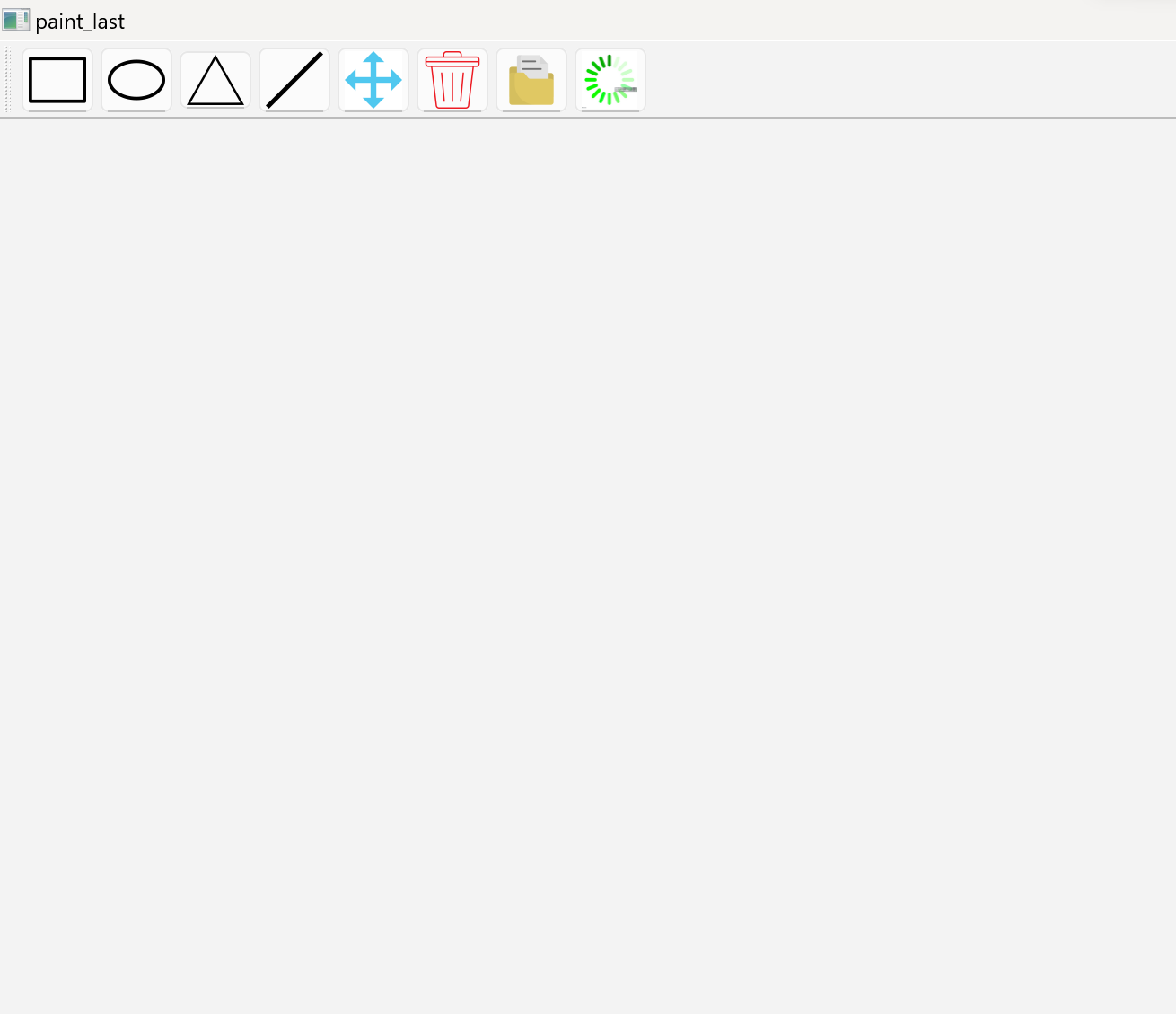


Рисунок 33 — открытие приложения

Так выгладит приложение при открытии.

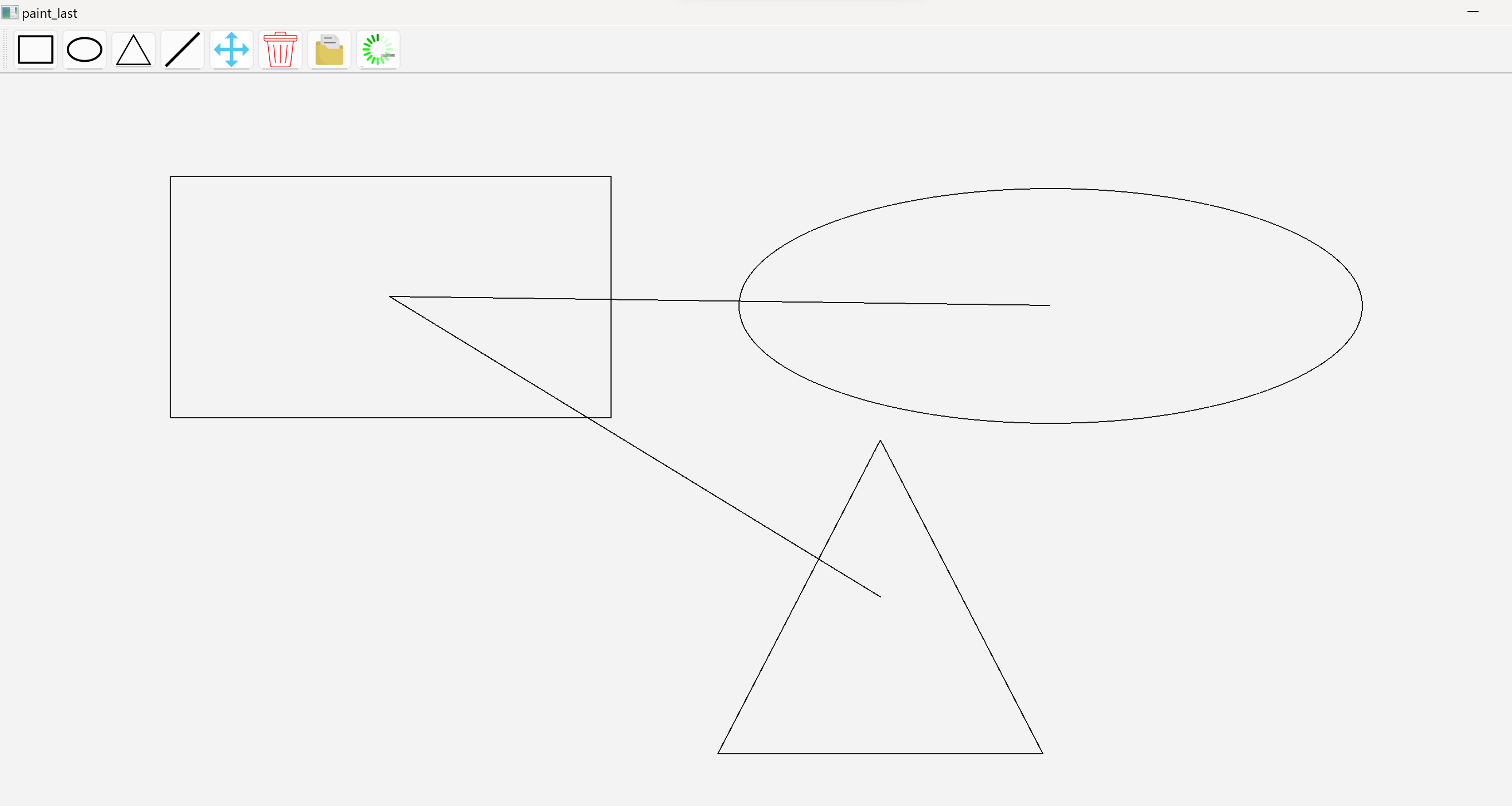


Рисунок 34 —нарисованы прямоугольник, эллипс, треугольник и связи

На рисунке уже нарисовали фигуры и соединили , нажав кнопку связь, квадрат с эллипсом и квадрат с треугольником. При перемещении фигур связь переместится вместе с фигурами.

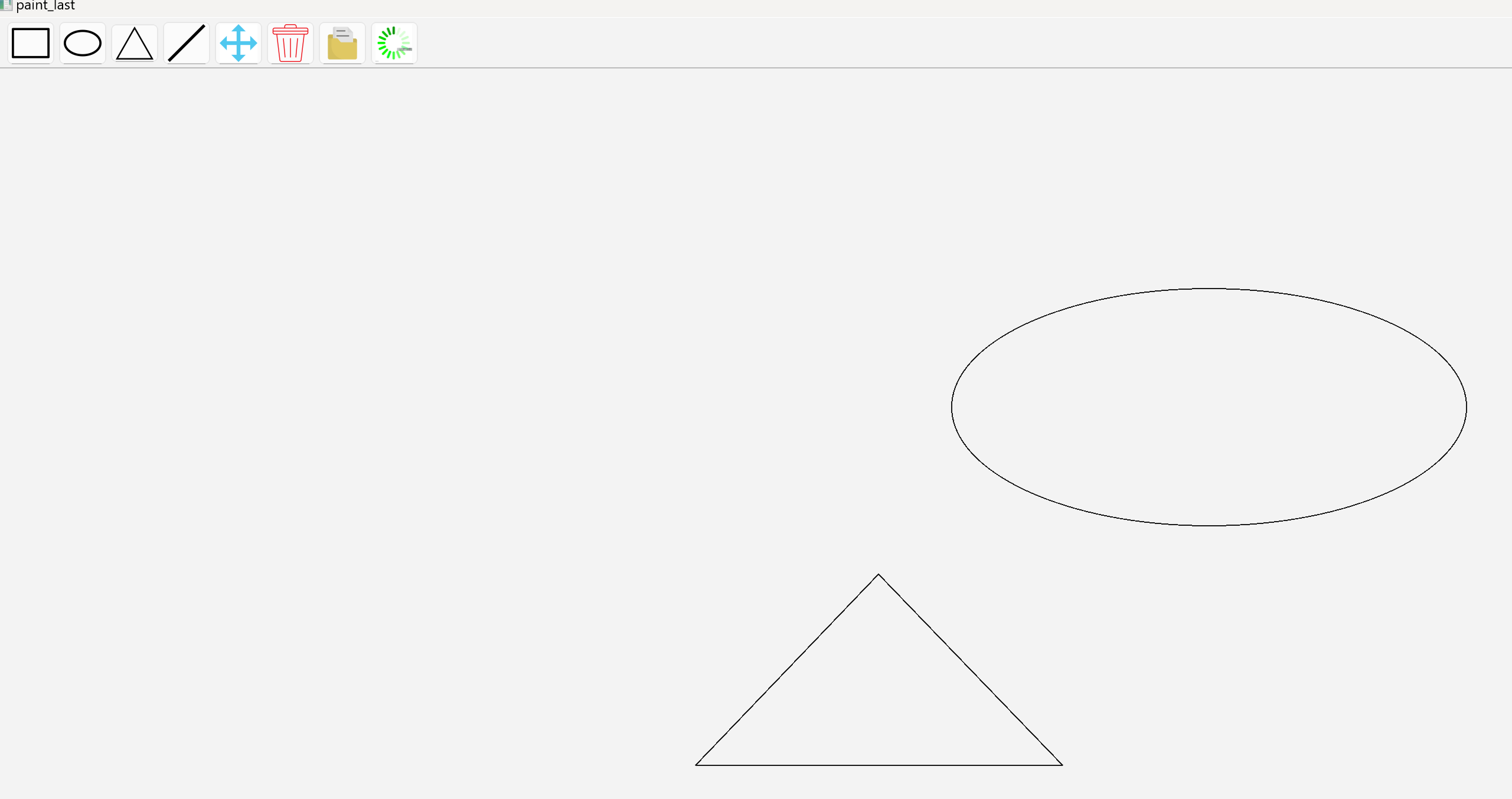


Рисунок 35 — результат удаления квадрата

Вместе с объектом удаляются и связи, которые он образовал, что и продемонстрировано на рисунке.

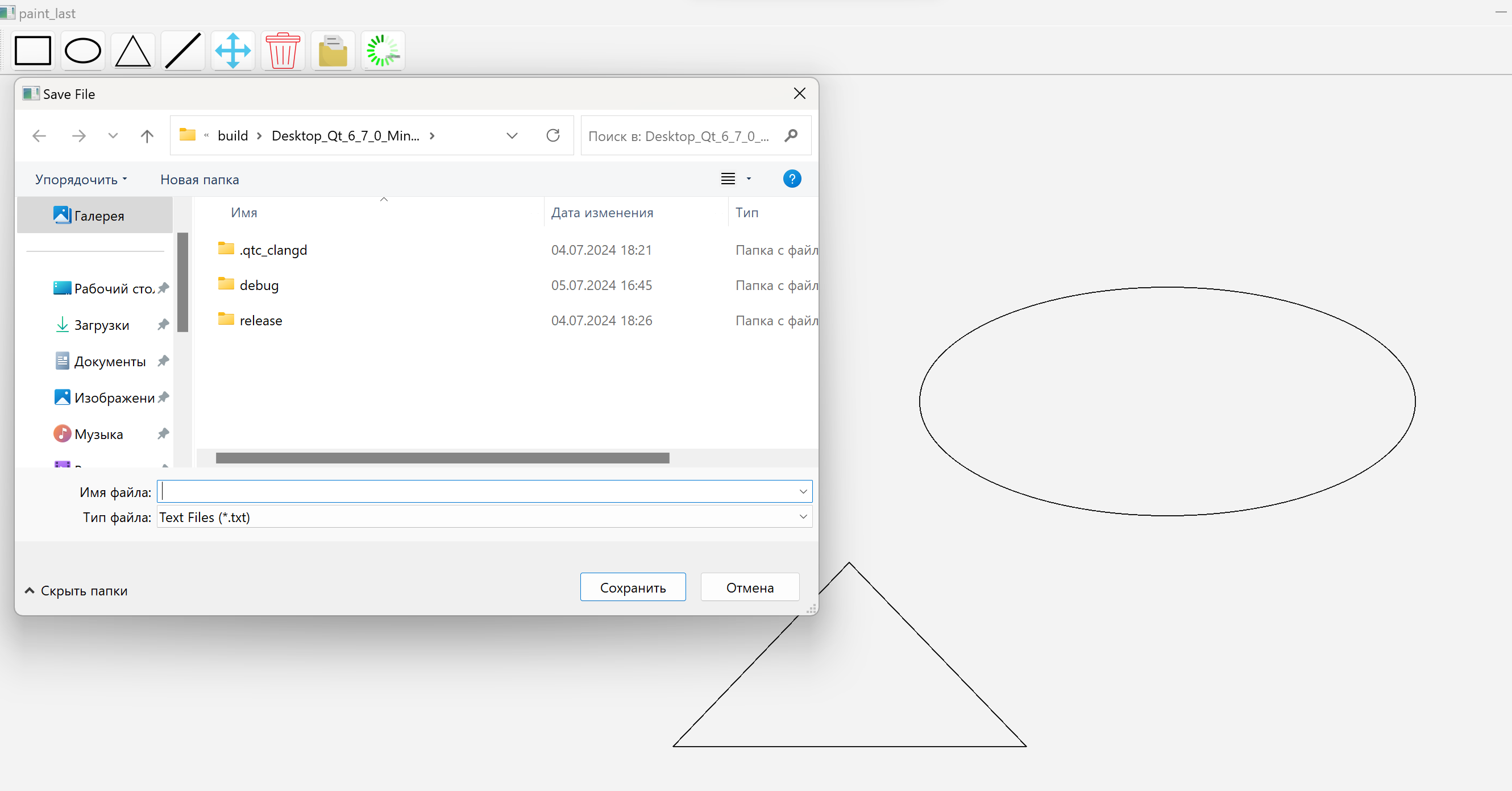


Рисунок 36 — результат нажатия кнопки "сохранить"

Как видно на рисунке, после нажатия на кнопку «сохранить» можно выбрать куда данный файл будет сохранен.

После загрузки файла его можно будет редактировать также как и до сохранения.

## 2.2 Дополнительный материал

На практических занятиях не вводилось понятие дополнительных работ. В предыдущем пункте были предоставлены отчеты по всем работам.

## 2.3 Выводы к главе 2

Выводы разбиты по работам предоставленным в пункте 2.1.

## 2.3.1Вывод по лабораторной работе номер 1

Командная строка Linux удобна в использовании для разных задач. В ходе лабораторной работы так же производилась работа с git и SQLite через консоль. Как отражено в отчете, через командную строку также можно компилировать и запускать файлы. Все эти возможности отражают многофункциональность консоли Linux. Отдельная часть лабораторной работы была посвящена изучению работы с базами данных.

Была предоставлена возможность выполнять основные операции в db Browser или в консоли. Я выбрала db Browser, т.к. он имеет более привычный интерфейс.

## 2.3.2 Вывод по лабораторной работе номер 2

Данная работа была посвящена созданию интерфейса для операций выполняемых над базами данных. Интерфейс создавался на Qt, поэтому использовались теоретические знания, полученные в ходе семестра. Дизайн может быть в последующем доработан, на данном этапе был оставлена минималистическая версия. Также проект был собран на GitHub в отдельном репозитории и промежуточным отчетом.

## 2.3.3 Вывод по лабораторной работе номер 3

В данной лабораторной работе мы продолжили совершенствовать навыков работы с QT.Главный полученный навык: умение работать с обработчиком мыши.

# Глава 3 Индивидуальный проект

## 3.1 Функциональное назначение

**Арканоид** — это классическая аркадная игра, которая была популярна в 1980-х годах. Основная цель игры заключается в разрушении всех кирпичей на экране, управляя платформой и отбивая шарик.

## 3.2 Структура проекта

Основные файлы и папки

Файлы заголовков (.h):

ball.h

paddle.h

brick.h

breakout.h

Файлы реализации (.cpp):

ball.cpp

paddle.cpp

brick.cpp

breakout.cpp

main.cpp

Ресурсы (изображения):

ball.png

paddle.png

brick.png

background.png

Проектные файлы:

arconoid.pro

arconoid.pro.user

Классы:

Class Paddle

Class Ball

Class Brick

Class Breakout

## 3.3 Интерфейс приложения

Начальный вид приложения:

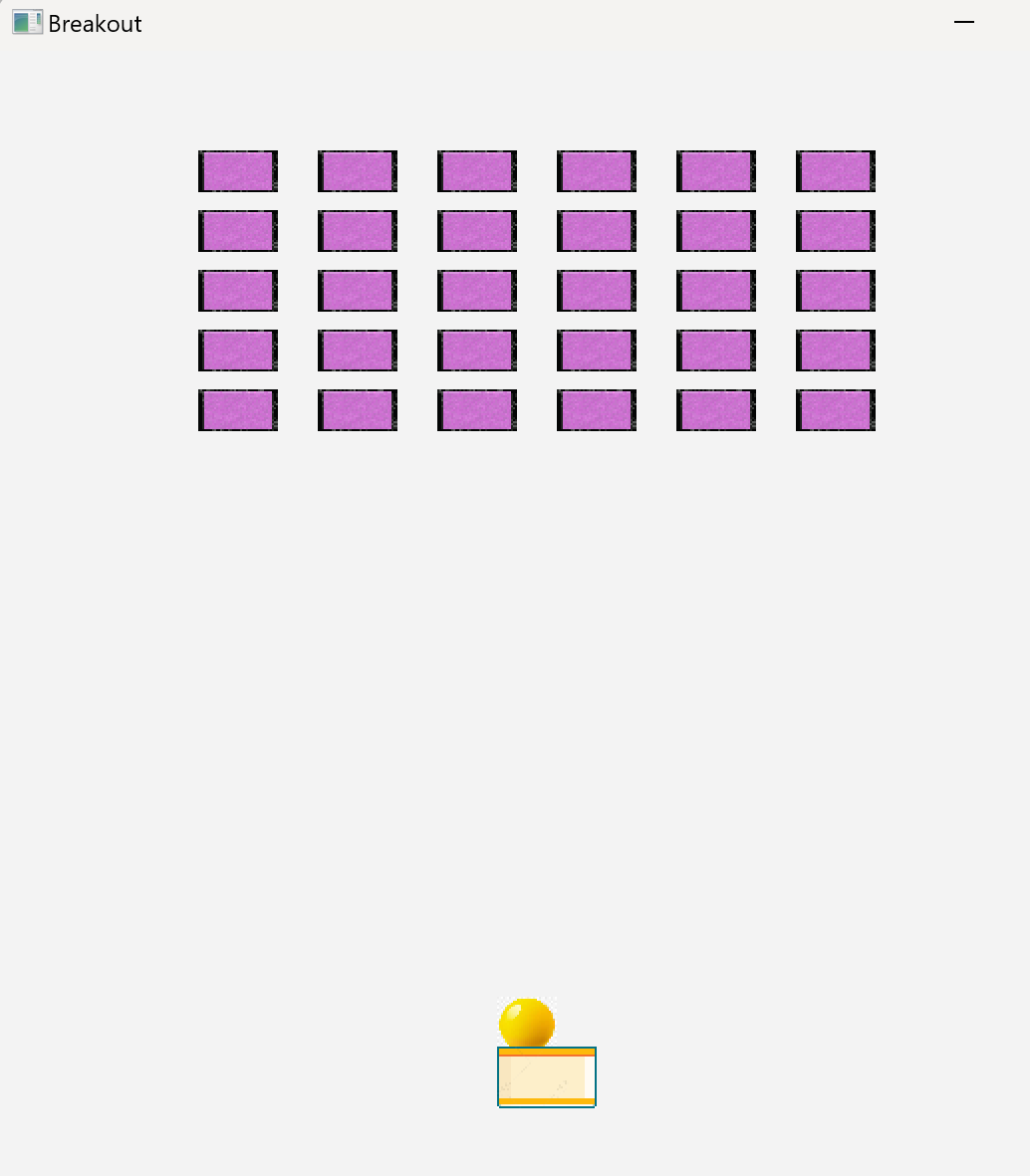


Рисунок 37 — вид приложения при запуске

После нажатия Enter мяч начинает летать и при столкновении с кубиком, кубик удаляется. Каретка(платформа) двигается с помощью клавиш <- и ->

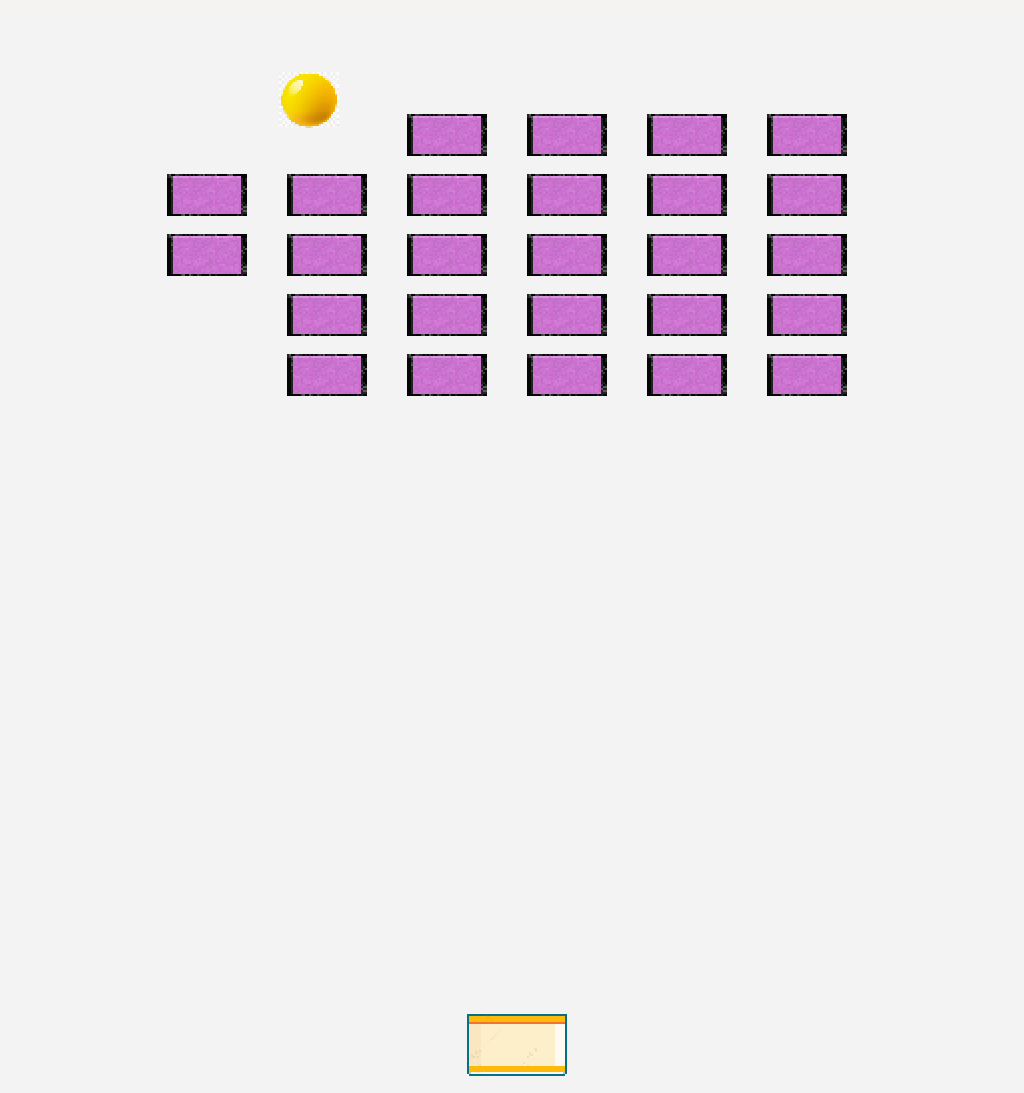


Рисунок 38 — демонстрация основной функции программы

Игра заканчивается в двух случаях: шарик не попал на каретку, тогда выведется надпись Game Lost



Рисунок 39— завершение программы



Рисунок 40 — завершение при выигрыше

Или пользователь выиграл, то есть сбил все кирпичики

## 3.4 Описание проекта

Код игры можно найти на GitHub: <https://github.com/fominami/game_lab4>

Для сборки игры нужно создать новый проект в QT и добавить все файлы.

## 3.5 Выводы к главе 3

Для создания игры были использованы интернет ресурсы, которые указаны в источниках. Во время работы трудностей не возникало. Большое внимание уделялось интерфейсу и проверке точности работы приложения.

# Заключение

Проделав все практические задания был получен опыт работы с GitHub, db Browser, git, с командной строкой Linux и Qt. Полученные в ходе практики навыки и знания упростили выполнение индивидуальной работы. Перед каждым задание проводилось тщательное изучение теоретических материалов по теме, что предотвратило возникновение сложностей. Отдельное внимание уделялось составлению плана работы для достижения полноты выполнения задания. В качестве дальнейших целей работы ставлю изучение иных фреймворков и совершенствование знаний по пользованию GitHub.

# Список использованных источников

1.METANIT.COM Сайт о программировании [Электронный ресурс]. – Режим доступа: https://metanit.com/cpp/tutorial/1.3.php – Дата доступа: 27.06.2024.

2. METANIT.COM Сайт о программировании [Электронный ресурс]. – Режим доступа: https://metanit.com/sql/sqlite/1.1.php – Дата доступа: 28.06.2024.

3. git-scm.com Сайт со всей информацией о Git [Электронный ресурс]. – Режим доступа: https://git-scm.com/ – Дата доступа: 29.06.2024.

4. ravesli.com Сайт о программирования [Электронный ресурс] Режим доступа: <https://ravesli.com/urok-15-sozdayom-igru-arkanoid-v-qt5-s/> —Дата доступа: 04.07.2024