МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

ОТЧЕТ

по учебной вычислительной (ознакомительной) практике

Усачёнка Артемия Дмитриевича

студента 1 курса, 7 группы

специальность «Прикладная Математика»

Руководитель практики:

старший преподаватель

Е.В. Пазюра

Минск, 2024

# ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc171081016)

[Введение 3](#_Toc171081017)

[Глава 1 Теоретические основы 5](#_Toc171081018)

[1.1 Знакомство с SoftClub 5](#_Toc171081019)

[1.2 Общая характеристика курса «Контроль версий с помощью Git» 5](#_Toc171081020)

[1.2.1 Установка платформы Git 5](#_Toc171081021)

[1.2.2 Изучение основных концепций системы контроля версий Git 5](#_Toc171081022)

[1.2.3 Работа с удаленными репозиториями 6](#_Toc171081026)

[1.3 Unit тестирование 6](#_Toc171081027)

[1.4 Разработка простейшего GUI-приложения в QT 7](#_Toc171081028)

[1.5 Разработка приложения отражающего табличные значения 7](#_Toc171081029)

[1.6 Разработка простейшего SDI-приложения 7](#_Toc171081030)

[1.7 Разработка аналога векторного графического редактора 8](#_Toc171081031)

[1.8 Разработка проекта с графической анимацией объектов 8](#_Toc171081032)

[1.9 Выводы к главе 1 9](#_Toc171081033)

[Глава 2 Практическая часть курса 12](#_Toc171081034)

[2.1 Unit-тестирование 12](#_Toc171081035)

[2.2 Разработка простейшего GUI-приложения в QT 12](#_Toc171081036)

[2.3 Разработка приложения, отражающего табличные данные 13](#_Toc171081037)

[2.4 Разработка простейшего SDI-приложения 14](#_Toc171081038)

[2.5 Разработка аналога векторного графического редактора 15](#_Toc171081039)

[2.6 Разработка проекта с графической анимацией объектов 16](#_Toc171081040)

[2.7 Выводы к главе 2 17](#_Toc171081041)

[Заключение 19](#_Toc171081042)

[Список использованных источников 20](#_Toc171081043)

# Введение

В период с 24 июня по 6 июля было предложено пройти учебную ознакомительную практику для студентов первого курса. За это время студенту необходимо было выполнить ознакомительную и практическую части.

В ознакомительной части был пройден курс по Git и GitHub, а также прослушана лекция SoftClub.

Практическая часть включает в себя Unit-тестирование, разработка простейшего GUI-приложения, разработка приложения, отражающего табличные данные, разработка простейшего SDI-приложения, разработка приложения, аналога векторного графического редактора, разработка проекта с графической анимацией объектов на экране.

Все части ставят основные цели и задачи практики, которые достигаются по мере прохождения.

Цели:

* Разработать векторный графический редактор, который позволит пользователям создавать, редактировать и управлять векторными графическими объектами.
* Обеспечить интуитивно понятный и функциональный графический интерфейс для работы с векторными изображениями.
* Реализовать возможность сохранения и загрузки векторных изображений в файлы.
* Спроектировать архитектуру приложения, которая будет обеспечивать модульность и возможность дальнейшего расширения функциональности.

Задачи:

* Спроектировать общую архитектуру приложения, включая структуру классов и взаимодействие между компонентами.
* Разработать функциональность для создания, редактирования и управления различными векторными графическими объектами (линии, фигуры, тексты и т.д.).
* Реализовать обработку событий от пользовательского ввода, таких как перемещение, масштабирование и вращение объектов с помощью мыши.
* Разработать механизм сериализации и десериализации набора графических объектов для сохранения и загрузки векторных изображений.
* Обеспечить модульность и расширяемость архитектуры приложения для возможности добавления новых типов графических объектов или инструментов редактирования.
* Протестировать и отладить разработанное приложение, обеспечив его стабильность и производительность.

Для прохождения практики было необходимо авторизоваться в ряде тестирующих систем:

* Learn.epam: imestor2@gmail.com
* Github.com: <https://github.com/1m3st3r>

# Глава 1 Теоретические основы

## Знакомство с SoftClub

SoftClub - это ИТ-компания, специализирующаяся на разработке программного обеспечения и предоставлении консалтинговых услуг. Была прослушана лекция компании про работу в данной сфере, а так же было рассказано про стажировку в SoftClub. Кроме того, на презентации вакансий участники узнали, что стажировка в SoftClub предоставляет отличные возможности для получения практического опыта в разработке программного обеспечения, ориентированного на финансовую отрасль.

Стажировка в SoftClub позволяет кандидатам приобрести ценные навыки и знания в области финансового ПО.

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

Во время учебной практики был пройден курс на Epam «Контроль версий с помощью Git». Данный курс включает в себя теоретическую и практическую части, а также промежуточные проверочные тесты для закрепления теоретических знаний. Была проделана практическая работа с целью углубления навыков использования системы контроля версий Git, а также изучены основы работы с платформой GitHub.

Практические занятия позволили закрепить теоретический материал и получить практический опыт работы с Git и GitHub. По итогам прохождения курса была предоставлена информация об этапах обучения, отражающая на каких шагах возникли сложности. Это дает возможность проанализировать успеваемость и сделать выводы для дальнейшего совершенствования навыков работы в Git.

## Установка платформы Git

Была установлена и настроена платформа Git, а также выполнена авторизация на GitHub.

## 1.2.2 Изучение основных концепций системы контроля версий Git

Были изучены: commit, push, branch, remotes, merge, stash, tags:

* Сохранение изменений в локальном репозитории с комментарием, описывающим произведенные модификации, осуществляется посредством операции commit.
* Для отправки локальных коммитов в удаленный репозиторий и синхронизации изменений с другими участниками проекта используется операция push.
* Ветвление (brunch) позволяет разделять разработку на независимые направления, что облегчает работу над проектом.
* Функционал remotes дает возможность синхронизировать локальный репозиторий с удаленными хранилищами, обеспечивая совместную работу над проектом.
* Для объединения изменений разных веток и интеграции наработок в единую кодовую базу применяется операция merge.
* Временное хранение незакоммиченных изменений с возможностью последующего восстановления для продолжения работы обеспечивается с помощью операции stash.
* Метки (tags) позволяют помечать ключевые моменты в истории проекта.



## Работа с удаленными репозиториями

Взаимодействие с удаленными репозиториями на платформе GitHub включает в себя связывание локального репозитория с его удаленным аналогом, размещенным на GitHub. Это позволяет эффективно организовывать совместную работу над проектом, обмениваться наработками и отслеживать историю развития кодовой базы. Для этого используются операции push и pull, обеспечивающие синхронизацию изменений между локальной и удаленной версиями проектов.

## Unit тестирование

Unit-тесты представляют собой небольшие, изолированные проверки, предназначенные для оценки корректности и соответствия ожидаемым результатам отдельных компонентов программного кода. Данный инструмент широко используется разработчиками для валидации правильности работы конкретных модулей или функций программы. Применение unit-тестов позволяет удостовериться, что каждый элемент кода функционирует корректно независимо от остальных частей программы, что в свою очередь способствует выявлению и устранению ошибок на ранних этапах разработки. В целом, unit-тестирование является важным компонентом в процессе разработки программного обеспечения, поскольку оно обеспечивает стабильность и надежность конечного кода.

## Разработка простейшего GUI-приложения в QT

Разработка простейшего GUI-приложения в Qt - это процесс создания графического пользовательского интерфейса (GUI) с использованием фреймворка Qt. Qt является кроссплатформенной библиотекой для разработки приложений с графическим интерфейсом, поддерживающей различные языки программирования.

Разработка простейшего GUI-приложения в Qt позволяет создавать интуитивно понятные и функциональные пользовательские интерфейсы, которые могут работать на различных платформах, включая Windows, macOS, Linux, Android и iOS. Это делает Qt популярным выбором для разработчиков, стремящихся создавать кроссплатформенные приложения с минимальными усилиями.

## Разработка приложения отражающего табличные значения

Разработка приложения, отражающего табличные данные, является важной задачей в создании современных пользовательских интерфейсов. Такие приложения позволяют эффективно представлять и взаимодействовать с большими объемами структурированной информации, что востребовано в различных областях, таких как бизнес-аналитика, научные исследования, обработка данных и многих других.

Разработка приложения, отражающего табличные данные, требует комплексного подхода, сочетающего в себе глубокое понимание пользовательских потребностей, проектирование эффективного интерфейса, надежную реализацию функциональности и тщательное тестирование. Применение передовых подходов и технологий в этой области позволяет создавать мощные и интуитивно понятные инструменты для работы с данными, повышая эффективность и производительность пользователей.

## Разработка простейшего SDI-приложения

SDI-приложения - это тип графических пользовательских интерфейсов (GUI), в котором каждый открытый документ отображается в отдельном окне.

Основными преимуществами SDI-приложений являются:

* Простота использования и интуитивно понятный интерфейс
* Возможность легко переключаться между открытыми документами
* Более гибкая организация рабочего пространства

## Разработка аналога векторного графического редактора

Использование фреймворка Qt и языка программирования C++ позволяет создать высокопроизводительный аналог векторного графического редактора. Основные компоненты такого проекта могут включать:

* Графический интерфейс пользователя (GUI): Разработка интуитивно понятного интерфейса с использованием виджетов.
* Работа с векторными объектами: Реализация классов для представления и управления различными геометрическими примитивами (линии, кривые, многоугольники и т.д.).
* Операции над объектами: Создание инструментов для выбора, масштабирования, поворота, дублирования и других действий над векторными объектами.
* Система слоев и группировка: Разработка механизма для организации объектов в слои и их группировки.
* Управление цветом и заливкой: Реализация подсистемы для применения сплошных цветов, градиентов и узоров к векторным объектам.
* Трансформации: Реализация различных геометрических преобразований, таких как перемещение, масштабирование, вращение, скос и отражение.
* Импорт и экспорт данных: Добавление функциональности для импорта и экспорта векторных изображений в популярных форматах.
* Сохранение и загрузка проектов: Разработка механизма для сохранения и загрузки состояния проекта.

## Разработка проекта с графической анимацией объектов

Разработка проекта с графической анимацией объектов является захватывающей и творческой задачей, которая требует интеграции различных навыков и подходов. Для создания успешной анимированной графики необходимо глубокое понимание основных принципов анимации, таких как тайминг, плавность движений и эмоциональная выразительность персонажей. Художники и аниматоры должны работать в тесном сотрудничестве, чтобы разработать визуальный стиль, который не только эстетически привлекателен, но и идеально передает настроение и личность анимированных объектов. Кроме того, разработчики должны владеть техническими навыками, необходимыми для создания анимированной графики, включая использование специализированного программного обеспечения и понимание принципов компьютерной графики. Этот всеобъемлющий процесс требует тщательного планирования, итеративного тестирования и постоянной оптимизации для достижения желаемого результата. В конечном итоге, успешная разработка проекта с графической анимацией объектов является результатом тесного сотрудничества, творческого мышления и технического мастерства.

## Выводы к главе 1

1. Знакомство с SoftClub:
   * SoftClub - ИТ-компания, специализирующаяся на разработке ПО и консалтинговых услугах.
   * Компания провела презентацию о работе в ИТ-сфере и возможностях стажировки.
   * Стажировка в SoftClub позволяет получить практический опыт в разработке финансового ПО.
2. Курс "Контроль версий с помощью Git":
   * Курс включает теоретическую и практическую части, а также промежуточные тесты.
   * Практические занятия позволили закрепить теоретический материал и получить опыт работы с Git и GitHub.
   * По итогам курса был предоставлен отчет об успеваемости, что позволяет проанализировать сложности и улучшить навыки.
3. Основные концепции системы контроля версий Git:
   * Commit - сохранение изменений в локальном репозитории с комментарием.
   * Push - отправка локальных коммитов в удаленный репозиторий.
   * Branch - ветвление для независимой разработки.
   * Remotes - синхронизация локального репозитория с удаленными.
   * Merge - объединение изменений разных веток.
   * Stash - временное хранение незакоммиченных изменений.
   * Tags - пометка ключевых моментов в истории проекта.
4. Работа с удаленными репозиториями:

* Взаимодействие с удаленными репозиториями на платформе GitHub включает в себя связывание локального репозитория с его удаленным аналогом, размещенным на GitHub.
* Это позволяет эффективно организовывать совместную работу над проектом, обмениваться наработками и отслеживать историю развития кодовой базы.
* Для этого используются операции push и pull, обеспечивающие синхронизацию изменений между локальной и удаленной версиями проектов.

1. Unit-тестирование:

* Unit-тесты представляют собой небольшие, изолированные проверки, предназначенные для оценки корректности и соответствия ожидаемым результатам отдельных компонентов программного кода.
* Данный инструмент широко используется разработчиками для валидации правильности работы конкретных модулей или функций программы.  
  Применение unit-тестов позволяет удостовериться, что каждый элемент кода функционирует корректно независимо от остальных частей программы, что в свою очередь способствует выявлению и устранению ошибок на ранних этапах разработки.
* Unit-тестирование является важным компонентом в процессе разработки программного обеспечения, поскольку оно обеспечивает стабильность и надежность конечного кода.

1. Разработка простейшего GUI-приложения в QT:

* Разработка простейшего GUI-приложения в Qt - это процесс создания графического пользовательского интерфейса (GUI) с использованием фреймворка Qt.
* Qt является кроссплатформенной библиотекой для разработки приложений с графическим интерфейсом, поддерживающей различные языки программирования, такие как C++, Python, Java и другие.
* Разработка простейшего GUI-приложения в Qt позволяет создавать интуитивно понятные и функциональные пользовательские интерфейсы, которые могут работать на различных платформах, включая Windows, macOS, Linux, Android и iOS.

1. Разработка приложения, отражающего табличные данные, является важной и комплексной задачей, требующей комплексного подхода. Ключевые аспекты при разработке такого приложения включают:

* Выбор подходящей технологической платформы и фреймворка.
* Тщательное проектирование пользовательского интерфейса.
* Эффективное взаимодействие с данными из различных источников.
* Использование эффективных средств визуализации данных.
* Реализация механизмов для обработки пользовательских взаимодействий.
* Обеспечение масштабируемости и высокой производительности.
* Проведение всесторонних тестов для обеспечения стабильности и надежности.
* Создание подробной документации для поддержки и дальнейшего развития приложения.

1. Разработка простейшего SDI-приложения:

* SDI-приложения - это тип графических пользовательских интерфейсов (GUI), в котором каждый открытый документ отображается в отдельном окне.
* Это противоположность MDI-приложениям (Multiple Document Interface), где документы открываются во вложенных окнах внутри главного окна приложения.
* Основными преимуществами SDI-приложений являются простота использования и интуитивно понятный интерфейс, возможность легко переключаться между открытыми документами, а также более гибкая организация рабочего пространства.

1. Разработка аналога векторного графического редактора:

* Использование фреймворка Qt и языка программирования C++ позволяет создать высокопроизводительный аналог векторного графического редактора с широкими возможностями. Такое приложение может включать в себя интуитивно понятный графический интерфейс, набор инструментов для работы с векторными объектами, систему слоев и группировки, управление цветом и заливкой, трансформации, а также функции импорта, экспорта и сохранения проектов. Данный подход обеспечивает создание кроссплатформенного приложения с высокой производительностью и удобным пользовательским опытом.

1. Разработка проекта с графической анимацией объектов:

* Разработка проекта с графической анимацией объектов требует сочетания художественных, технических и коммуникативных навыков. Это творческий процесс, который включает в себя понимание принципов анимации, создание визуальной концепции, использование специализированных инструментов и тесную совместную работу команды. Ключом к успеху является тщательное планирование, итеративное тестирование и постоянная оптимизация, что в конечном итоге позволяет создать анимированную графику, которая гармонично сочетает эстетическую привлекательность, эмоциональную выразительность и технические характеристики.

# Глава 2 Практическая часть курса

## 2.1 Unit-тестирование

Задача: составить UNIT-тест в рамках модульного тестирования.

QFETCH - это макрос, предоставляемый фреймворком Qt Test, который используется для извлечения данных, необходимых для выполнения тестов в Qt. Основная цель QFETCH - отделить данные тестов от самих тестовых функций. Это позволяет легко управлять наборами тестовых данных и повторно использовать их в различных тестах.

QVERIFY является макросом, предоставляемым фреймворком Qt для модульного тестирования (Qt Test Framework). Он используется для проверки условий в юнит-тестах.

QTEST\_APPLESS\_MAIN - это макрос, предоставляемый фреймворком Qt Test, который позволяет создавать автономные (без приложения) модульные тесты в Qt.

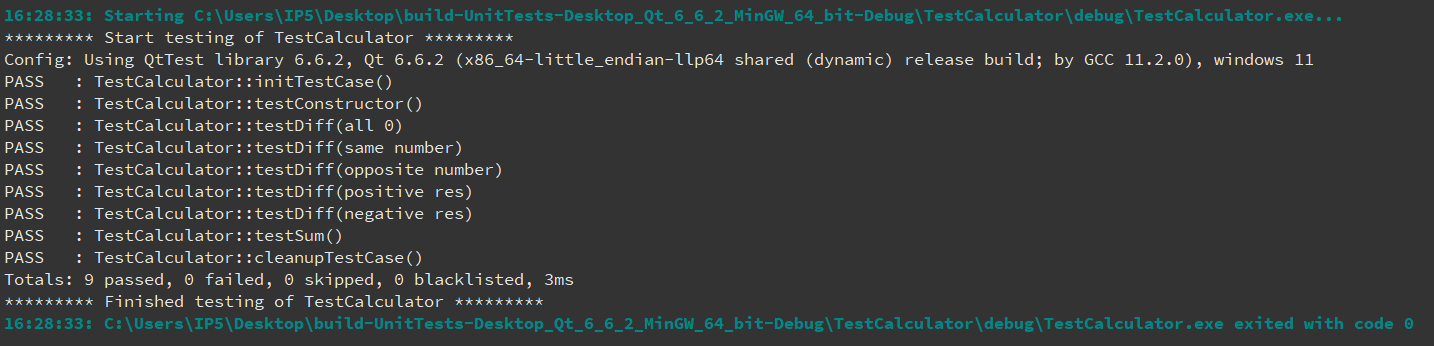
⠀

Рисунок 1. Результаты тестов

## 2.2 Разработка простейшего GUI-приложения в QT

Задача: разработать простейшее GUI-приложения в QT

Реализация: реализует визуализацию биологических ритмов человека в зависимости от количества прожитых дней. Пользователь может видеть графическое отображение физиологического, эмоционального и интеллектуального состояния на заданную дату, а также количество дней между датами.

Графики биоритмов - это визуальное представление циклических изменений в физическом, эмоциональном и интеллектуальном состоянии человека. Биоритмы - это периодические колебания различных физиологических и психологических процессов в организме человека.

Для построения графиков биоритмов используется следующая формула:

Значение биоритма = Амплитуда \* sin(2π \* t / Период).

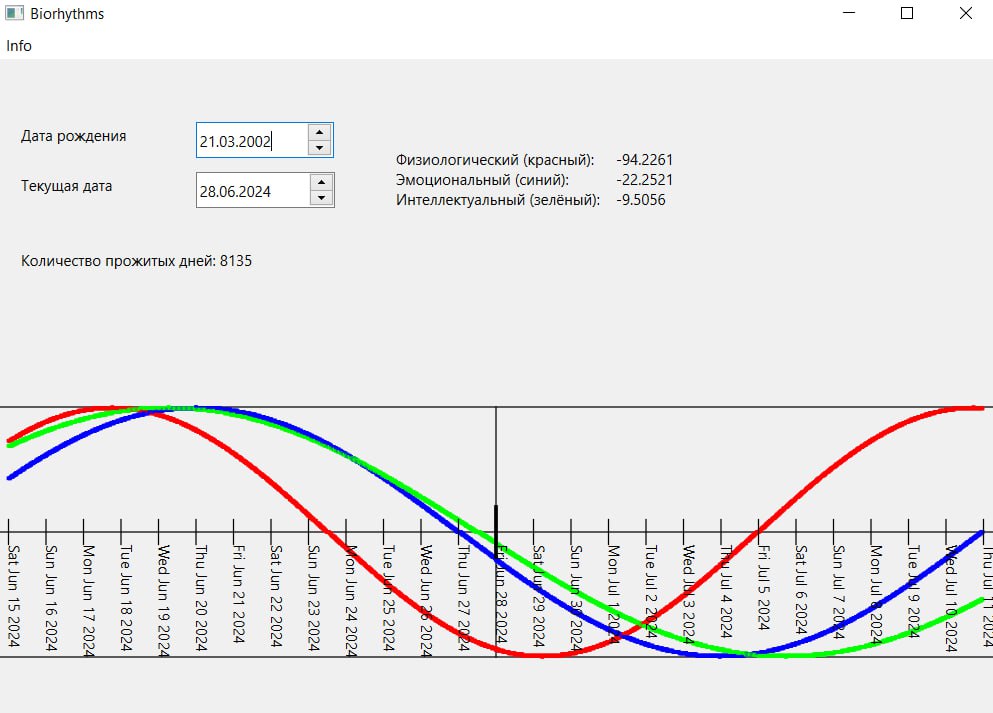


Рисунок 2. Графики биоритмов

## 2.3 Разработка приложения, отражающего табличные данные

Задача: Разработать приложение, отражающее табличные данные.

Реализация: реализует простое приложение, в котором можно открывать CSV-файл, отображать его содержимое в таблице, и изменять размеры таблицы. Данные из CSV-файла также записываются в новый файл.

В конструкторе происходят следующие действия:

* Создается экземпляр QStandardItemModel для хранения данных таблицы.
* Устанавливается количество столбцов модели и задаются их заголовки.
* Связывается QTableView с моделью данных.
* Устанавливается режим редактирования ячеек таблицы - по двойному клику.

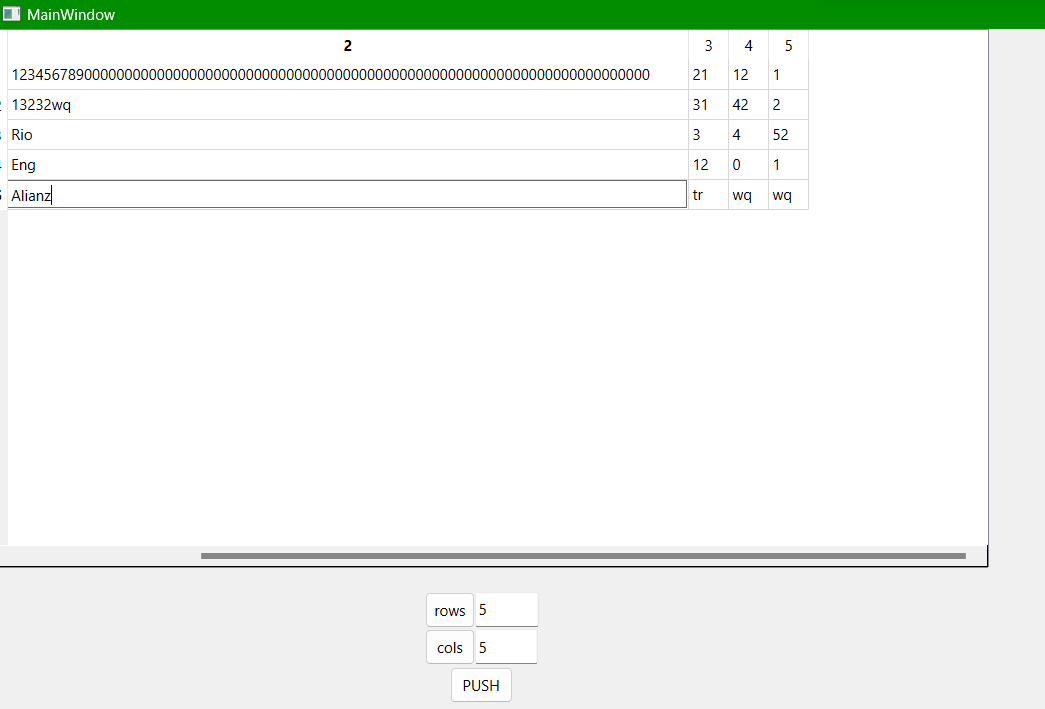


Рисунок 3. Таблица

## 2.4 Разработка простейшего SDI-приложения

Задача: Разработать простейшее SDI-приложение в QT

QMyForm::QMyForm(QWidget \*parent)

: QMainWindow(parent)

{

QMenu\* file = new QMenu("&File");

QMenu\* info = new QMenu("&Info");

QWorkForm\* form = new QWorkForm;

file->addAction("&Open", form, SLOT(slotLoad()));

info->addAction("&Help", this, SLOT(slotInfo()));

file->addAction("&Save", form, SLOT(slotSave()));

file->addAction("&Save as", form, SLOT(slotSaveAs()));

file->addSeparator();

file->addAction("&Quit", qApp, SLOT(quit()));

menuBar()->addMenu(file);

menuBar()->addMenu(info);

statusBar()->setVisible(false);

setCentralWidget(form);

connect(form, SIGNAL(ChangeTitle(const QString&)), SLOT(slotChangeTitle(const QString&)));

}

Этот конструктор настраивает главное окно приложения, добавляя в него меню "File" и "Info", а также устанавливая центральный виджет QWorkForm. Также устанавливается обработчик сигнала ChangeTitle(const QString&), испускаемого объектом QWorkForm.



Рисунок 4. Простейшее SDI-приложение

## 2.5 Разработка аналога векторного графического редактора

Задача: разработать аналог векторного графического редактора.

Реализация: реализованы сохранение и загрузка файлов, линии, прямоугольники с границами.

Выполняет следующие основные функции:

* Инициализация пользовательского интерфейса:

Конструктор MainWindow создает и настраивает пользовательский интерфейс, используя класс Ui::MainWindow, сгенерированный Qt Designer.

Он создает экземпляр QStandardItemModel для хранения и отображения данных в QTableView.

* Изменение размера таблицы:

Обработчик события on\_pushButton\_5\_clicked() реагирует на нажатие кнопки с идентификатором pushButton\_5. Он проверяет, что значения, введенные в поля lineEdit и lineEdit\_2, являются положительными числами. Он также добавляет новые ячейки в модель, чтобы соответствовать новому размеру.

В конце он обновляет размер ячеек в QTableView, чтобы они соответствовали новому размеру модели.

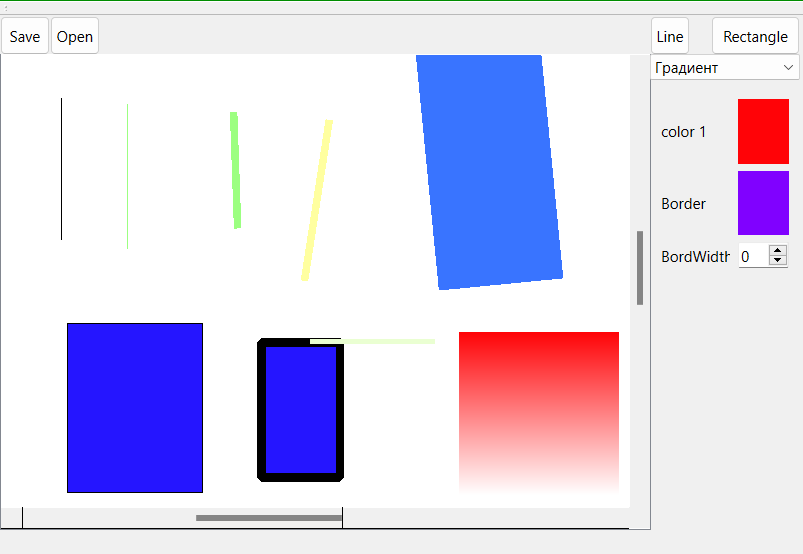


Рисунок 5. Аналог графического редактора

## 2.6 Разработка проекта с графической анимацией объектов

Задача: разработать приложение с графической анимацией объектов на экране.

На экране появляется фотография, а также на экране появляется вместо курсора бейсбольная бита. Управляя мышкой пытаться догнать фото, нажимая ЛКМ. Фото в это время убегает. В том случае, когда удается попасть по фотографии – она меняется на другую. При четвертом (4) нажатии по фотографии на экране появляется гроб и движется в сторону фотографии.

Также реализован выбор персонажей, у каждого определенное количество жизней.

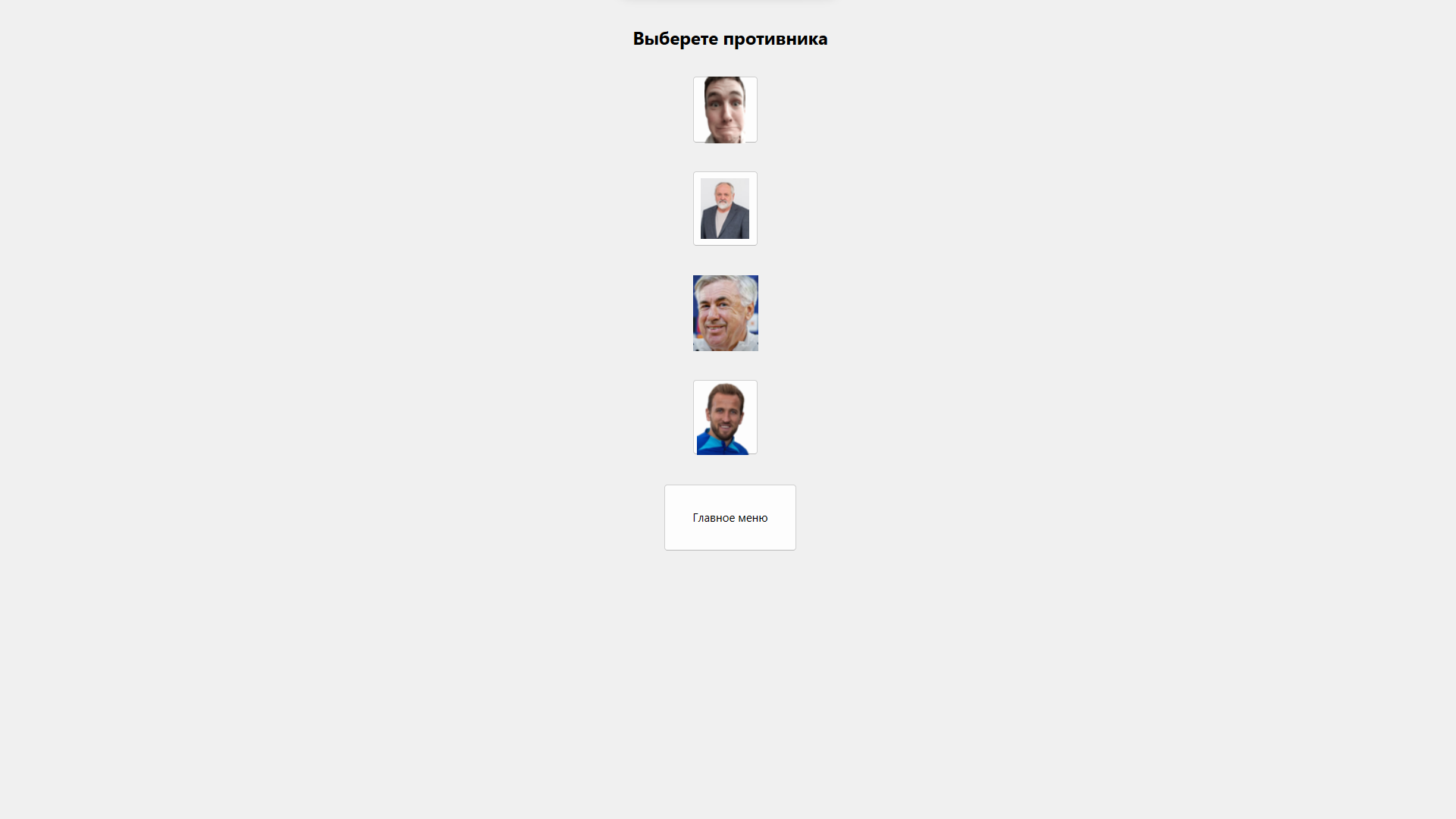


Рисунок 6. Главное меню

В данном окне происходят действия игры.

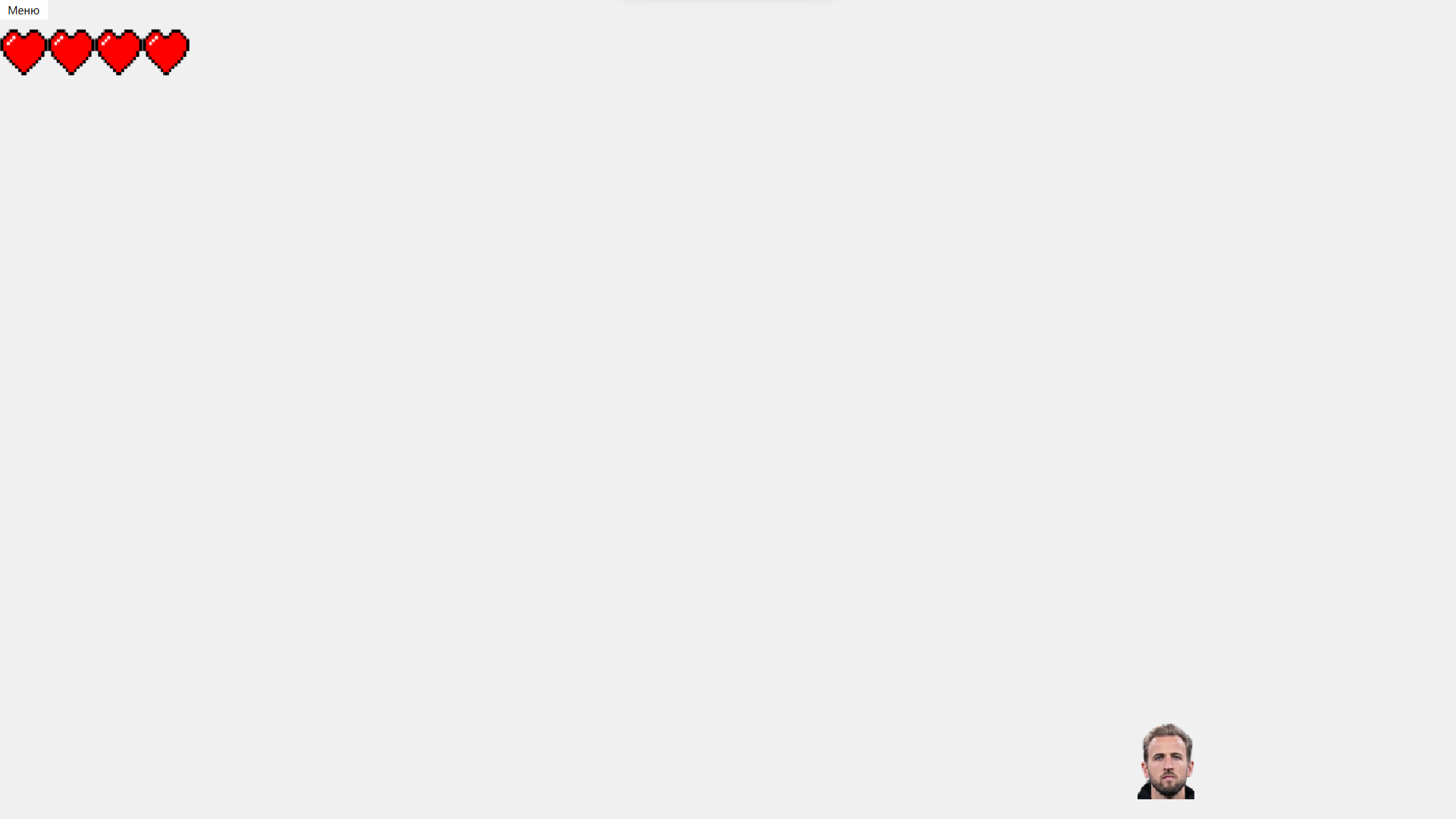


Рисунок 7. Игра

## 2.7 Выводы к главе 2

Были созданы программы (приложения):

Unit-тестирование:

* QFETCH - макрос для извлечения данных, необходимых для выполнения тестов в Qt. Позволяет отделить данные тестов от самих тестовых функций.
* QVERIFY - макрос для проверки условий в юнит-тестах в Qt Test Framework.
* QTEST\_APPLESS\_MAIN - макрос для создания автономных (без приложения) модульных тестов в Qt.

Разработка GUI-приложения:

* Для построения графиков биоритмов используется формула: Значение биоритма = Амплитуда \* sin(2π \* t / Период).

Разработка приложения, отражающего табличные данные:

* Приложение должно позволять открывать CSV-файл, отображать его содержимое в таблице, и изменять размеры таблицы.
* Данные из CSV-файла также должны записываться в новый файл.

Разработка простейшего SDI-приложения:

* Представленный код конструктора класса QMyForm настраивает главное окно приложения, добавляя в него меню "File" и "Info", а также устанавливая центральный виджет QWorkForm.
* Меню "File" содержит пункты для открытия, сохранения и выхода из приложения, а меню "Info" - пункт для вызова справки.
* Устанавливается обработчик сигнала ChangeTitle(const QString&), испускаемого объектом QWorkForm, который, вероятно, используется для изменения заголовка главного окна.

Разработка аналога векторного графического редактора:

* Продемонстрированы основные принципы создания графического интерфейса пользователя с помощью фреймворка Qt, включая работу с моделями данных, отображение данных в таблице, загрузку данных из файла и обработку пользовательских действий.

Разработка проекта с графической анимацией объектов:

* Игра представляет собой своеобразное преследование убегающей фотографии при помощи бейсбольной биты, где при определенном количестве попаданий появляется гроб, а также присутствует возможность выбора персонажей с ограниченным количеством жизней.

# Заключение

В рамках практики были выполнены следующие задачи:

Прослушана вводная лекция от SoftClub, получены базовые знания по Git. Пройдены все модули основного курса, выполнены все задачи. Общая успешность прохождения составила 96%, при этом теоретические тесты были пройдены на 100%.

Кроме того, были выполнены задачи различного уровня сложности:

Задачи на темы Unit-тестирования, простейшего GUI-приложения и SDI-приложения, приложение с графической анимацией объектов (игра) были выполнены полностью.

Задача по созданию аналога векторного графического редактора была выполнена с незначительными недочетами.

Дополнительно было разработано приложение "Хэш-таблица", демонстрирующее понимание более сложных алгоритмических концепций.

Результаты проделанной работы можно оценить как положительные. Практические задания разной сложности были выполнены достаточно успешно. Вместе с тем, есть небольшие недочеты в решении некоторых задач, которые можно использовать как основу для дальнейшего развития навыков.

# Список использованных источников

1. Chacon Scott, Straub Ben. Pro Git. Нидерланды: Apress, 2014. – 419 c.
2. GitHub. Режим доступа: [https://github.com/](https://github.com/%20). – Дата доступа – 28.06.2024.

<https://elearn.epam.com/assets/courseware/v1/d8b3970d06d567a94e0bb179b8d84acb/asset-v1:EPAM+VCG+RU+type@asset+block/DevTestOps-Version_Control_with_Git.pdf> Дата доступа – 24.06.2024.

<https://learn.epam.com/detailsPage?id=601f195a-d408-4439-a16d-0630ed2a412e> Дата доступа – 24.06.2024.

1. [METANIT.COM](https://metanit.com/) - Сайт о программировании. Режим доступа: [https://metanit.com/](https://metanit.com/%20) - Дата доступа: 01.07.2024.
2. Qt Forum. Режим доступа: <https://forum.qt.io/>. Дата доступа – 25.06.2024.
3. Курс на Epam. Режим доступа:
4. Презентация по Git и GitHub. Режим доступа:
5. Сайт Git. Режим доступа: <https://git-scm.com/>. Дата доступа: 24.06.2024.