Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей Кафедра электронных вычислительных машин

Дисциплина: Программирование на языках высокого уровня

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

«Игра “Змейка”»

Студент В. В. Приловский

Руководитель Е. В. Богдан

МИНСК 2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc151838688)

[1 ОБЗОР ЛИТЕРАТУРЫ 4](#_Toc151838689)

[1.1 Обзор методов и алгоритмов решения поставленной задачи 4](#_Toc151838690)

[1.2 Разработка требований к функционалу 5](#_Toc151838691)

[2 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 7](#_Toc151838692)

[2.1 Структура входных и выходных данных 7](#_Toc151838693)

[2.2 Разработка диаграммы классов 7](#_Toc151838694)

[2.3 Описание классов 8](#_Toc151838695)

[3 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 12](#_Toc151838696)

[3.1 Разработка схем алгоритмов 12](#_Toc151838697)

[3.2 Разработка алгоритмов 12](#_Toc151838698)

[УСЛОВНЫЕ СОКРАЩЕНИЯ 14](#_Toc151838699)

[ЗАКЛЮЧЕНИЕ 15](#_Toc151838700)

[ПРИЛОЖЕНИЕ А 17](#_Toc151838701)

[ПРИЛОЖЕНИЕ Б 18](#_Toc151838702)

[ПРИЛОЖЕНИЕ В 19](#_Toc151838703)

[ПРИЛОЖЕНИЕ Г 20](#_Toc151838704)

# ВВЕДЕНИЕ

В течение всей истории информационных технологий, люди постоянно преобразуют свои идеи в код, создавая программы для различных сфер жизни, включая развлекательную индустрию. Одним из ярких примеров такого влияния является компьютерная игра “Змейка”. Эта игра, несмотря на свою простоту, завоевала мировую популярность и стала классикой жанра. Целью данного проекта является разработка игры “Змейка”.

Игра “Змейка” представляет собой увлекательную аркаду, где игрок управляет змейкой, цель которой - съесть как можно больше еды, избегая столкновения с собственным хвостом. Эффективность игры во многом зависит от качества ее реализации и интерфейса.

В рамках данного проекта были разработаны следующие функциональные возможности:

1. Регистрация новых пользователей;
2. Авторизация пользователя;
3. Возможность выйти из аккаунта;
4. Просмотр максимального результата всех зарегистрированных пользователей в порядке убывания;
5. Возможность выбора цвета “Змейки”;
6. Возможность регулировки громкости музыки, вплоть до её полного отключения;
7. Возможность включения и отключения звука кнопок;
8. Возможность подбирать еду для увеличения длины “Змейки” и увеличения счёта;
9. Возможность подбирать бонусы для ускорения и замедления “Змейки” и увеличении счёта;
10. Возможность поставить игру на паузу.

Проект выполнен с использованием языка программирования C++ и фреймворка Qt, что обеспечивает высокую производительность и кроссплатформенность. Для хранения данных о игроках и их результатах используются файлы с расширением .txt.

В дальнейшем планируется расширение функциональности игры, добавление препятствий и разных уровней.

В заключении хотелось бы отметить, что разработка данной игры является актуальной задачей, способной привнести новые идеи в классическую игру “Змейка” и предоставить пользователям интересное и увлекательное времяпрепровождение.

1. **ОБЗОР ЛИТЕРАТУРЫ**
   1. **Обзор методов и алгоритмов решения поставленной задачи**

**1.1.1 Объектно-ориентированное программирование**

Объектно-ориентированное программирование – это методология программирования, которая использует объекты для моделирования и организации кода. В C++ ООП реализуется с помощью классов и объектов [7].

Классы в C++ являются шаблонами для создания объектов. Они определяют, какие данные и функции будут содержать объекты этого класса.

Объекты в C++ – это экземпляры класса. Каждый объект имеет свой собственный набор данных (атрибутов) и функций (методов), определенных в его классе.

ООП в C++ поддерживает четыре основных принципа:

1. Инкапсуляция: Инкапсуляция означает сокрытие деталей реализации и объединение данных и методов в одном объекте. В приложении, например, данные пользователя инкапсулированы в объектах класса GameField;
2. Наследование: Наследование позволяет создавать новые классы на основе существующих, переиспользуя их код и добавляя новые функции. В приложении классы Snake, SnakeFood и SnakeBonus наследуются от класса ItemElements. Это позволяет им использовать переменные m\_x и m\_y из базового класса;
3. Полиморфизм: Полиморфизм позволяет использовать один и тот же интерфейс для различных типов данных;
4. Абстракция: Абстракция означает представление сложных систем через более простые интерфейсы.

**1.1.2 Фреймворк QT**

Для создания графического интерфейса также используется Qt. Qt – это мощный инструмент для создания GUI в C++ [4]. Qt SQL. Qt – полностью объектно-ориентированная библиотека. Новая концепция ведения меж объектных коммуникаций, именуемая "сигналы и слоты", полностью заменяет былую, не вполне надежную модель обратных вызовов [6]. Этот модуль включает классы QWidget, QPainter и другие, которые обеспечивают удобный объектно-ориентированный интерфейс для работы с базами данных. Также имеется возможность обработки событий, например нажатия клавиш клавиатуры, нажатия элементов интерфейса [5].

**1.2****Разработка требований к функционалу**

“Игра «Змейка»” — это приложение, которое представляет собой интерактивную игру, позволяющую пользователям управлять змейкой на игровом поле. Функционал может зависеть от нужд пользователя:

1. Игра: Все пользователи могут играть в игру “Змейка”. Змейка движется по игровому полю, и цель игры - съесть как можно больше еды, избегая столкновения с собственным хвостом;

2. Использование бонусов: Пользователи могут подбирать бонусы во время игры, которые влияют на скорость змейки;

3. Просмотр счета: Пользователи могут просматривать свой текущий счет во время игры;

4. Пауза/продолжение игры: Пользователи могут ставить игру на паузу и продолжать игру после паузы;

5. Начать новую игру: Пользователи могут начать новую игру в любое время;

6. Регистрация и авторизация пользователей: Приложение поддерживает систему регистрации и авторизации пользователей. Пользователи должны ввести свои учетные данные для входа в систему;

7. Рекорды: Пользователи могут просматривать таблицу рекордов, которая отображает лучшие результаты всех зарегистрированных пользователей.

Дополнительные требования к приложению включают в себя:

1. Удобство использования: Интерфейс пользователя должен быть интуитивно понятным и легким в использовании;

2. Привлекательный дизайн: Дизайн игры должен быть привлекательным и интересным для пользователей;

3. Музыкальное сопровождение: Игра может включать в себя музыкальное сопровождение для улучшения игрового опыта;

4. Настройки: Пользователи могут настраивать различные параметры игры, такие как цвет змейки или громкость музыки.

# 2 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

**2.1 Структура входных и выходных данных**

Приложение работает с двумя основными типами входных данных:

1. Данные пользователя: Это данные, которые пользователь вводит через GUI. Они включают в себя информацию для регистрации нового пользователя, данные для входа в систему существующего пользователя, а также действия, которые пользователь хочет выполнить в игре, такие как управление змейкой перевод режима игры в паузу;

2. Данные игры: Это данные, которые приложение генерирует во время игры, такие как положение змейки, еды и бонусов на игровом поле.

Приложение генерирует следующие выходные данные:

1. Данные GUI: это данные, которые отображаются пользователю через GUI. Они могут включать в себя сообщения об ошибках, подтверждения успешных действий и информацию из файлов с расширением типа .txt;

2. Сигналы: приложение также генерирует сигналы как часть механизма сигналов и слотов Qt. Сигналы используются для обработки событий, таких как нажатие кнопок пользователем;

3. Текстовые файлы: приложение способно открывать текстовые файлы, содержащие всю информацию о призывниках.

**2.2 Разработка диаграммы классов**

Диаграмма классов – это структурная диаграмма языка моделирования UML, которая демонстрирует общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними. Она широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

На диаграмме классы представлены в рамках, содержащих три компонента:

1. В верхней части написано имя класса. Имя класса выравнивается по центру и пишется полужирным шрифтом;

2. В средней части перечислены атрибуты (поля) класса;

3. В нижней части перечислены методы класса.

Целью создания диаграммы классов является графическое представление статической структуры классов.

Диаграмма классов для игры «Змейка» приведена в приложении А.

**2.3 Описание классов**

**2.3.1 Класс MainWindow**

Класс MainWindow представляет собой главное окно приложения и является базовым классом для остальных классов. Вот более подробное описание его методов:

1. Конструктор (MainWindow::MainWindow(QWidget \*parent)): Это конструктор класса MainWindow. Он настраивает пользовательский интерфейс для окна, устанавливает размеры окна и стиль. Он также инициализирует медиаплеер для воспроизведения музыки во время игры.

2. Деструктор (MainWindow::~MainWindow()): Это деструктор класса MainWindow, который удаляет пользовательский интерфейс при уничтожении объекта.

3. playGame(): Этот метод вызывается при нажатии кнопки “Play”. Он создает новое окно игры и скрывает главное окно.

4. enterName(): Этот метод вызывается при нажатии кнопки “Enter nickname”. Он создает диалоговое окно для ввода имени пользователя.

5. openSettings(QString&): Этот метод вызывается при нажатии кнопки “Settings”. Он создает диалоговое окно для настройки параметров игры, таких как цвет змейки и громкость музыки.

6. createRegistrationDialog(): Этот метод создает диалоговое окно для регистрации нового пользователя. Он проверяет валидность введенных данных и сохраняет их в файле, если они допустимы.

7. createAuthorizationDialog(): Этот метод создает диалоговое окно для авторизации пользователя. Он проверяет валидность введенных данных и загружает данные пользователя из файла, если они допустимы.

8. openImageWindow(): Этот метод вызывается при нажатии на изображение в главном окне. Он создает новое окно с информацией об авторизованном пользователе и таблицей рекордов.

9. showMainWindow(): Этот метод вызывается при нажатии кнопки “Back” в окне игры. Он закрывает окно игры и отображает главное окно.

* + 1. **Класс CustomTitleBar**

Класс CustomTitleBar представляет собой пользовательскую панель заголовка окна:

1. Конструктор (CustomTitleBar::CustomTitleBar(QWidget \*parent)): Это конструктор класса CustomTitleBar. Он настраивает пользовательский интерфейс для панели заголовка и устанавливает фиксированную высоту для панели.

2. mousePressEvent(QMouseEvent \*event): Этот метод вызывается при нажатии кнопки мыши на панели заголовка. Он сохраняет текущую позицию курсора.

3. mouseMoveEvent(QMouseEvent \*event): Этот метод вызывается при перемещении мыши по панели заголовка. Если кнопка мыши зажата, он перемещает окно в новую позицию курсора.

* + 1. **Класс GameWindow**

Класс GameWindow представляет собой окно игры:

1. Конструктор (GameWindow::GameWindow(MainWindow \*mainWindow, QWidget \*parent)): Это конструктор класса GameWindow. Он принимает указатель на MainWindow и настраивает пользовательский интерфейс для окна игры.

2. handleAltPressed(): Этот метод вызывается при нажатии клавиши Alt. Он закрывает окно игры и открывает главное окно.

**2.3. Класс SoundButton**

Класс SoundButton представляет собой настраиваемую кнопку QPushButton, которая может воспроизводить звук при наведении курсора:

1. Конструктор (SoundButton::SoundButton(const QString &text, QWidget \*parent)): Это конструктор класса SoundButton. Он принимает текст кнопки и указатель на родительский виджет в качестве параметров.

2. enterEvent(QEvent \*event): Этот метод вызывается при наведении курсора на кнопку. Если звук включен, он воспроизводит звук.

* + 1. **Класс HelpField**

Класс HelpField представляет собой виджет, который отображает текстовую информацию:

1. Конструктор (HelpField::HelpField()): Это конструктор класса HelpField. Он инициализирует текстовое поле.

2. paintEvent(QPaintEvent \*e): Этот метод вызывается при обновлении виджета. Он перерисовывает текстовое поле.

3. ChangeTextSlot(QString text): Этот слот вызывается при получении сигнала ChangeTextSignal от другого объекта. Он изменяет текст в текстовом поле.

* + 1. **Класс ItemElements**

Класс ItemElements представляет собой базовый класс для элементов игры:

Конструктор (ItemElements::ItemElements(int x, int y)): Это конструктор класса ItemElements. Он принимает координаты x и y в качестве параметров и инициализирует положение элемента.

* + 1. **Класс Snake**

1. Класс Snake представляет собой змейку в игре:

Конструктор (Snake::Snake()): Это конструктор класса Snake. Он инициализирует змейку.

2. getBody(): Этот метод возвращает тело змейки в виде списка элементов.

**2.3.8 Класс SnakeFood**

Класс SnakeFood представляет собой еду для змейки:

1. Конструктор (SnakeFood::SnakeFood(int x, int y)): Это конструктор класса SnakeFood. Он принимает координаты x и y в качестве параметров и инициализирует положение еды.

2. CreateFood(Snake\* m\_snake, int m\_fieldSize): Этот метод создает новую еду для змейки в случайном месте на игровом поле, которое не занято змейкой.

* + 1. **Класс SnakeBonus**

Класс SnakeBonus представляет собой бонус для змейки:

1. Конструктор (SnakeBonus::SnakeBonus(int x, int y, bool isSpeedUp)): Это конструктор класса SnakeBonus. Он принимает координаты x и y и булево значение isSpeedUp в качестве параметров и инициализирует положение и тип бонуса.

2. CreateBonus(Snake\* m\_snake, int m\_fieldSize, SnakeFood\* m\_food, SnakeBonus \*&m\_bonus, QTimer \*m\_bonusTimer): Этот метод создает новый бонус для змейки в случайном месте на игровом поле, которое не занято змейкой или едой. Бонус может быть либо ускорением, либо замедлением. Бонус исчезает после определенного времени, если его не съели.

* + 1. **Класс GameField**

Класс GameField представляет собой игровое поле:

1. Конструктор (GameField::GameField()): Это конструктор класса GameField. Он инициализирует игровое поле и создает змейку, еду и бонусы.

2. paintEvent(QPaintEvent \*e): Этот метод вызывается при обновлении виджета. Он перерисовывает игровое поле.

3. keyPressEvent(QKeyEvent \*e): Этот метод вызывается при нажатии клавиши на клавиатуре. Он обрабатывает управление змейкой.

4. MoveSnakeSlot(): Этот слот вызывается при каждом шаге таймера. Он перемещает змейку на одну клетку в текущем направлении.

5. RemoveBonus(): Этот слот вызывается при истечении времени действия бонуса. Он удаляет бонус с игрового поля.

6. ExitToMenu(): Этот метод вызывается при завершении игры. Он закрывает окно игры и открывает главное меню.

7. SetGameStatus(): Этот метод вызывается для установки статуса игры (пауза, игра, конец игры).

8. GameOver(): Этот метод вызывается при проигрыше в игре. Он отображает сообщение о проигрыше и предлагает начать новую игру или выйти в меню.

9. StartNewGame(): Этот метод вызывается для начала новой игры. Он сбрасывает игровое поле и создает новую змейку, еду и бонусы.

10. GameOverAndGoInMenu(): Этот метод вызывается при проигрыше в игре. Он отображает сообщение о проигрыше и возвращает пользователя в главное меню.

11. updateScore(int gameScore): Этот метод вызывается для обновления счета игры. Он принимает новый счет в качестве параметра и обновляет отображаемый счет.

12. changeSnakeColor(int color): Этот слот вызывается для изменения цвета змейки. Он принимает новый цвет в качестве параметра и изменяет цвет змейки.

# 3 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

**3.1 Разработка схем алгоритмов**

Метод CreateBonus(Snake\* m\_snake, int m\_fieldSize, SnakeFood\* m\_food, SnakeBonus \*&m\_bonus, QTimer \*m\_bonusTimer) отвечает за перемещение создание бонусов для змейки. Схема метода CreateBonus(Snake\* m\_snake, int m\_fieldSize, SnakeFood\* m\_food, SnakeBonus \*&m\_bonus, QTimer \*m\_bonusTimer) показана в приложении Б.

Метод keyPressEvent(QKeyEvent \*e) обрабатывает нажатия клавиш для управления змейкой и паузы в игре. Схема метода keyPressEvent(QKeyEvent \*e) показана в приложении В.

**3.2 Разработка алгоритмов**

**3.2.1****Алгоритм для начала игры “Змейка”**

1. Запуск программы: При запуске программы отображается главное окно приложения MainWindow, которое содержит кнопки “Play”, “Enter nickname”, “Settings” и “Back”.

2. Ввод имени пользователя: Пользователь может ввести свое имя пользователя, нажав на кнопку “Enter nickname”. Открывается диалоговое окно для ввода имени пользователя. После ввода имени пользователя и нажатия кнопки “OK”, имя пользователя сохраняется.

3. Настройка игры: Пользователь может выбрать цвет змейки, нажав на кнопку “Settings”. Открывается диалоговое окно, где можно выбрать цвет змейки из предложенных вариантов.

4. Начало игры: Пользователь нажимает кнопку “Play”, чтобы начать игру. Создается новое окно игры GameWindow, главное окно скрывается. На игровом поле появляется змейка и еда.

5. Игровой процесс: Змейка автоматически движется в текущем направлении. Пользователь может изменить направление змейки, нажав клавиши со стрелками на клавиатуре. Если змейка сталкивается со стеной или собственным телом, игра заканчивается, и отображается сообщение о проигрыше.

6. Еда: На игровом поле случайным образом появляется еда. Если змейка съедает еду, она увеличивается в размере, и пользователь получает очки. После того как еда съедена, на игровом поле появляется новая еда.

7. Бонусы: На игровом поле случайным образом появляются бонусы. Если змейка съедает бонус, она может временно ускориться или замедлиться. После определенного времени бонус исчезает.

8. Конец игры: Если змейка сталкивается со собственным телом, игра заканчивается. Отображается сообщение о проигрыше и предлагается начать новую игру или выйти в меню.

9. Возврат в меню: Пользователь может вернуться в главное меню в любое время, нажав клавишу Alt. Текущая игра прерывается, и отображается главное окно.

**3.2.2 Алгоритм регистрации пользователя**

1. Пользователь нажимает кнопку “Enter nickname”, чтобы ввести свое имя пользователя. Открывается диалоговое окно для ввода имени пользователя.

2. Пользователь вводит свое имя пользователя и его пароль в текстовые поля после чего нажимает кнопку “Save”.

3. Если имя пользователя допустимо (не пустое и не содержит недопустимых символов), оно сохраняется в файле.

4. Если имя пользователя недопустимо, отображается сообщение об ошибке, и пользователь может попробовать ввести имя пользователя снова.

5. После успешной регистрации имя пользователя отображается в главном окне. Пользователь теперь может начать игру с этим именем пользователя.

6. Если пользователь нажал кнопку “Back” в диалоговом окне регистрации, диалоговое окно закрывается, и имя пользователя не сохраняется. Пользователь возвращается в главное меню.

7. Если пользователь закрыл диалоговое окно регистрации, не вводя имя пользователя, отображается сообщение об ошибке, и пользователь может попробовать ввести имя пользователя снова.

8. Имя пользователя используется для отображения в таблице рекордов и для сохранения прогресса игры.

**ЗАКЛЮЧЕНИЕ**

В рамках данной работы было успешно разработано приложение "Змейка" с использованием фреймворка Qt. Это приложение предлагает пользователям возможность играть в классическую игру "Змейка" с дополнительными функциями, такими как выбор цвета змейки и регистрация пользователя.

Для обеспечения удобства взаимодействия с пользователем были использованы различные виджеты Qt для отображения игрового поля и интерфейса. Была также реализована функциональность по работе с файлами для сохранения данных пользователя.

В ходе работы были применены различные методы для достижения поставленных целей. Вначале был проведен анализ требований к функционалу и характеристикам приложения. Затем были разработаны алгоритмы на языке программирования C++, которые управляют игровым процессом и обрабатывают данные.

В итоге разработанное приложение успешно реализует задуманный функционал и предоставляет интересный и увлекательный игровой процесс. Благодаря использованию фреймворка Qt, приложение обладает приятным и интуитивно понятным интерфейсом, что делает его доступным для широкого круга пользователей.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Рожнова, Н. Г. Вычислительные машины, системы и сети. Дипломное проектирование : учебно-метод.пособие / Н. Г. Рожнова, Н. А. Искра, И. И. Глецевич. – Минск : БГУИР, 2014. – 96 с. : ил.

[2] СТП 01–2013. Дипломные проекты (работы): общие требования. – Введ. [Электронный ресурс]. – 2013. – Режим доступа: http://www.bsuir.by/online/showpage.jsp.

[3] Единая система конструкторской документации (ЕСКД): справ. пособие / С. С. Борушек [и др.]. – М. : Изд-во стандартов, 1989. – 352 с

[4] Шлее М. - Qt4. Профессиональное программирование на С+/ Шлее М. - Л.:Наука, 2013. - 770 с.

[5] Бланшет Ж., Саммерфилд М. - QT 4: программирование GUI на С+/ Бланшет Ж., Саммерфилд М. М. — M. : ALT Linux, 2015. — 948 с. : ил.

[6] А. В.Чеботарев Библиотека Qt 4. Программирование прикладных приложений в среде Linux / А. В.Чеботарев - Л.: Наука, 2013. - 821 с.

[7] Страуструп, Б. Язык программирования С++ / Б. Страуструп. - М. : БИНОМ, 2004.- 1098 с.

[8] Дейтел, Х. Как программировать на С++ / Х. Дейтел, П. Дейтел. - М. : БИНОМ, 2001. - 1152 с.

**ПРИЛОЖЕНИЕ А**

*(обязательное)*

Диаграмма классов

**ПРИЛОЖЕНИЕ Б**

*(обязательное)*

Блок-схема метода

**ПРИЛОЖЕНИЕ В**

*(обязательное)*

Блок-схема метода

**ПРИЛОЖЕНИЕ Г**

*(обязательное)*

Код программы