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 курсовому проекту
на тему

ИГРА “СЛАБОЕ ЗВЕНО”

БГУИР КП 1-40 02 01 221 ПЗ

Студент: гр. 250502 Левошко А. Ю.

Руководитель: Богдан Е. В.

МИНСК 2023

Учреждение образования
«Белорусский государственный университет информатики
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой
_____
(подпись)
_____2023 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту *Левошко Артемию Юрьевичу*
Тема проекта *Игра "Слабое звено"*
2. Срок сдачи студентом законченного проекта *15 декабря 2023 г.*
3. Исходные данные к проекту *Язык программирования – C++, среда разработки – Qt-Creator*
4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)
1. Лист задания.
2. Введение.
3. Обзор литературы.
4. Функциональное проектирование.
4.1. Структура входных и выходных данных.
4.2. Разработка диаграммы классов.
4.3. Описание классов.
5. Разработка программных модулей.
5.1. Разработка схем алгоритмов (два наиболее важных метода).
5.2. Разработка алгоритмов (описание алгоритмов по шагам, для двух методов).
6. Результаты работы.
7. Заключение
8. Литература
9. Приложения

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)
1. Диаграмма классов.
2. Схема алгоритма метода answerButtonClicked()
3. Схема алгоритма метода passButtonClicked()
6. Консультант по проекту (с обозначением разделов проекта) Богдан Е. В.

7. Дата выдачи задания *15.09.2023г.*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

1. Выбор задания. Разработка содержания пояснительной записки. Перечень графического материала – 15 %;

разделы 2, 3 – 10 %;

разделы 4 к –20 %;

разделы 5 к – 35 %;

раздел 6,7,8 – 5 %;

раздел 9 к – 5%;

оформление пояснительной записки и графического материала к 15.12.23 – 10 %

Защита курсового проекта с 21.12 по 28.12.23г.

РУКОВОДИТЕЛЬ Богдан Е. В.

(подпись)

Задание принял к исполнению_____

(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном динамичном мире программной разработки выбор языка программирования и технологий становится критическим фактором, оказывающим существенное влияние на успех любого проекта. В контексте этого стремительно меняющегося ландшафта мой взгляд остановился на языке программирования C++ и фреймворке Qt, представляющих собой несравненно мощное и перспективное сочетание для разработки разнообразных приложений. Данное введение направлено на более детальное рассмотрение ключевых мотивов, лежащих в основе моих выборов.

Универсальность и эффективность языков программирования выступают двумя определяющими факторами, и C++ в этом отношении выделяется своей уникальной мощью и гибкостью. Этот язык предоставляет разработчикам возможность использовать различные парадигмы программирования, начиная от процедурного и заканчивая объектно-ориентированным и обобщенным программированием. Благодаря этим концепциям, C++ оказывается идеальным инструментом для решения многочисленных задач — от низкоуровневого программирования до создания высокоуровневых сложных систем.

Также стоит подчеркнуть, что C++ обладает обширной стандартной библиотекой, включающей множество инструментов и алгоритмов. Это не только упрощает процесс разработки, но и значительно повышает производительность программ, предоставляя разработчикам готовые решения для ряда типичных задач.

Важным аспектом, который подчеркивает привлекательность языка C++, является его активное участие в различных отраслях, таких как системное программирование, игровая индустрия, встраиваемые системы и научные исследования. Наличие богатого экосистемы инструментов и библиотек способствует созданию масштабных и инновационных проектов. Возможность эффективного использования ресурсов и оптимизации производительности делают C++ незаменимым инструментом для разработчиков, стремящихся к созданию высокопроизводительного и надежного программного обеспечения.

В свою очередь, Qt, как фреймворк, не только обеспечивает удобные средства для построения интерфейсов, но и активно участвует в развитии современных технологий, таких как Интернет вещей (IoT) и мобильная разработка. Использование Qt Creator, интегрированной среды разработки, упрощает процесс создания, отладки и тестирования приложений, что повышает эффективность и уровень удовлетворенности разработчиков.

В контексте создания графического интерфейса и разработки кроссплатформенных приложений выбор фреймворка Qt становится стратегически важным решением. Qt предоставляет обширный набор инструментов для построения современных и интуитивно-понятных пользовательских интерфейсов. Его кроссплатформенность дарит

возможность создавать приложения, которые могут без проблем функционировать на различных операционных системах, минимизируя необходимость в переписывании кода.

Система сигналов и слотов в Qt обеспечивает не только эффективную обработку событий, но и элегантное взаимодействие между компонентами приложения. Дополняя функциональность C++, богатая стандартная библиотека Qt предоставляет готовые решения для различных типичных задач, что делает этот фреймворк незаменимым инструментарием для разработки.

Мой выбор C++ и Qt в данной курсовой работе обусловлен не только их выдающейся функциональностью, универсальностью и эффективностью, но и способностью этих технологий эффективно взаимодействовать друг с другом. Это в совокупности делает их оптимальным инструментарием для успешной реализации поставленных задач в быстро развивающемся мире программной инженерии.

В итоге, выбор использовать C++ и Qt делает написание моей курсовой работы простым и понятным занятием. Эти технологии дают гибкость, повышают производительность и следят за последними трендами в разработке приложений. Они по-прежнему остаются в лидерах в своей области благодаря постоянному совершенствованию и умению адаптироваться к новым вызовам. Именно поэтому я выбрал именно данные программы для своей курсовой работы.

1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Анализ существующих аналогов

“Слабое Звено” является достаточно популярной телевизионной интеллектуальной игрой.

И даже на русском телевидении существуют программы, суть которых схожа со “Слабым Звеном”. Например, игра “Кто хочет стать миллионером?” или же “Один из ста”. Сейчас мы более подробно рассмотрим суть данных игр.

Суть игры “Кто хочет стать миллионером?” следующая:

Участники: В игре обычно участвует один игрок (кандидат), который соревнуется, чтобы выиграть денежные призы.

Цель игры: Главная цель игрока - ответить на серию вопросов разной сложности и выиграть как можно больше денег. Верное ответы позволяют игроку продолжать игру и зарабатывать деньги.

Структура игры: Игра имеет несколько уровней, обычно 15, причем каждый уровень имеет свою сложность и призовую сумму. Игра начинается с простых вопросов, а по мере продвижения игрока вперед вопросы становятся сложнее и приносят больше денег. Верный ответ позволяет продолжать игру и перейти к следующему уровню. Неправильный ответ может привести к потере части выигрыша.

Возможность использования “подсказок”: Игрок может использовать несколько видов “подсказок”, таких как “помощь зала” (аудитория в студии голосует за варианты ответов), “позвонить другу” (игрок звонит своему другу за советом), и “50:50” (две неверные опции удаляются).

Достижение миллиона: Главная цель игры - дойти до последнего вопроса и ответить на него верно, чтобы выиграть миллион долларов или эквивалент в местной валюте.

Риск и награда: Игра также предоставляет игроку возможность остановиться и забрать свой текущий выигрыш в любой момент.

Пример реализации игры “Кто хочет стать миллионером?” приведен на рисунке 1.1.

Рисунок 1.1. – Пример игры “Кто хочет стать миллионером?”

Суть игры “Один из ста” следующая:

Участники: В программе обычно есть один основной участник, который соревнуется против 100 других участников, которых называют “массой”.

Цель игры: Главная цель основного участника - ответить на серию вопросов и попытаться заработать денежные призы. Масса отвечает на те же вопросы и пытается помешать основному участнику выиграть.

Структура игры: Игра имеет несколько уровней, включая начальные вопросы, которые обычно легкие, и более сложные вопросы по мере продвижения вперед. Основной участник начинает игру с ограниченным количеством “подсказок” и имеет возможность использовать их для получения правильных ответов.

Соотношение с массой: Масса представлена 100 участниками, и их задачей является отвечать на вопросы вместе. Если хотя бы один участник из массы дает правильный ответ, основной участник теряет жизнь или долю выигрыша.

Достижение цели: Основной участник должен ответить правильно на определенное количество вопросов, чтобы выиграть денежный приз. Если основной участник считает, что не знает ответ на вопрос, он может выбрать “сняться с игры” и забрать свой текущий выигрыш.

Риск и награда: Игра предоставляет основному участнику возможность продолжать соревнование и увеличивать выигрыш, но он также рискует потерять его, если допускает ошибку.

Пример реализации игры “Один из ста” приведен на рисунке 1.2.

Рисунок 1.2. – Пример игры “Один из ста”

Сейчас мы более подробно рассмотрим русский аналог игры “Слабое звено”, на который я опирался при создании своей реализации.

Суть данной игры:

Команда участников: Как правило, в русской версии игры участвует восемь человек. Эти участники могут представлять разные социокультурные группы, например, работники различных профессий, представители разных возрастных категорий или жителей разных регионов.

Раунды вопросов: Раунды вопросов представляют собой основную часть игры. Участникам задаются вопросы по различным темам, таким как история, наука, искусство и текущие события.

Деньги и банк: За каждый правильный ответ команда зарабатывает деньги. Собранные деньги накапливаются в банке, и этот банк определяет выигрыш.

Выбор слабого звена: В конце каждого раунда участники голосуют и выбирают одного из своих участников как "самое слабое звено". Этот участник покидает игру. Голосование часто основывается на восприятии участников о производительности своих товарищей по команде. Игроки могут исключать как по-настоящему “самое слабое звено”, так и наоборот, сильных игроков, для повышения своих шансов на победу.

Ведущий: Ведущий играет важную роль в поддержании напряженности и динамики игры. Он задает вопросы, следит за временем и контролирует процесс голосования. Ведущий программы должен обладать четкой, поставленной речью, широким спектром знаний и умением заинтересовывать зрителей. Все эти качества полностью раскрываются в Марии Киселевой – ведущей “Слабого звена” в нашем регионе. Её образ, строгий вид, быстрый

голос и колкие фразочки быстро полюбились зрителям, и программа стала рейтинговой.

Пример телевизионной версии игры “Слабое Звено” приведен на рисунке 1.3.

Финальный раунд: В конце игры остаются два участника. В финальном раунде они поочередно отвечают на 5 дополнительных вопросов. Кто ответил на большее количество вопросов, тот и является победителем.

Выигрыш: Победитель забирает весь банк, накопленный за время всей игры.

Пример реализации игры “Слабое Звено” приведен на рисунке 1.4.

Рисунок 1.3. – Пример телевизионной версии игры “Слабое звено”

Рисунок 1.4. – Пример игры “Слабое звено”

Однако впервые телешоу “Слабое звено” появилось в 2000 году в Великобритании и правила в ней слегка отличались от привычной всем нам версии данной игры. Сейчас мы более подробно рассмотрим правила оригинальной игры “Слабое звено”.

Командные раунды: Игра начинается с группы игроков, которые образуют команду. Каждый раунд начинается с вопроса, и команда совместно обсуждает ответ. Правильный ответ приносит деньги для призового фонда.

Голосование: После каждого раунда игроки голосуют за того, кого считают наименее компетентным участником. Игроки могут выбирать стратегически или влиять на выбор других участников.

"Слабое звено": Игрок, получивший наибольшее количество голосов, объявляется "слабым звеном". Ведущий часто задает этому игроку вопросы, которые могут быть более сложными, чтобы проверить его знания. За правильные ответы "слабое звено" может заработать дополнительные деньги.

Повторяющийся процесс: Игра продолжается с новыми раундами, где команда отвечает на вопросы и голосует за "слабое звено". Процесс повторяется до тех пор, пока не останется только два игрока.

Финал: Оставшиеся два игрока играют в финале. Вопросы в финале могут быть более сложными и иметь большую стоимость. Победитель становится тем, кто наберет больше денег в финальном раунде.

Динамичность: Ведущий шоу, Энн Робинсон, известен своей строгостью и сарказмом. Ее комментарии касаются не только игрового процесса, но и личных характеристик участников, что придает шоу дополнительное напряжение

Моя реализация “Слабого Звена” имеет ряд недостатков по сравнению с оригинальной телевизионной игрой. Один из самых примечательных изъянов – это возможность игры только на одном устройстве. В моей игре опущены некоторые правила и возможности, например голосование. Однако оно убрано во избежание намеренного исключения из игры одного из участников (напоминаю, все игроки играют на одном устройстве). Также в своей игре я не посчитал нужным как-либо изменять количество времени в зависимости от раунда, поэтому время на любой раунд равно 60 секундам.

1.2 Обзор методов и алгоритмов решения поставленной задачи

Для создания “Слабого звена” был использован фреймворк Qt. Он содержит стандартные библиотеки C++, а также набор инструментов для создания удобного графического интерфейса.

Игрок может совершать действия, нажимая на различные кнопки. В моем проекте реализованы три окна: главное окно (`MainWindow`), где и происходит основная игра, на нем появляются вопросы, игроки вводят свои ответы, а также на нем указан банк и время; окно после раунда (`afterRoundWindow`), где указан игрок, который по истечению данного раунда покидает игру; завершающее окно (`finishWindow`), демонстрирубщее нам победителя игры.

В моей версии игры у игрока есть несколько возможностей, такие как: ответить на вопрос, положить деньги в банк, передать ход следующему игроку. После ввода ответа, игроку дают понять верным он был или нет.

Класс `Game` содержит данные об игре и оперирует ими, с помощью класса `Interface`.

Процесс игры продолжается пока не останется только один игрок. Он и является победителем.

2 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

2.1 Структура входных и выходных данных

В начале игры пользователь сразу попадает в первый раунд (класс `MainWindow`), где, начиная с первого игрока, участники поочередно отвечают на вопросы. Это окно показано на рисунке 2.1.

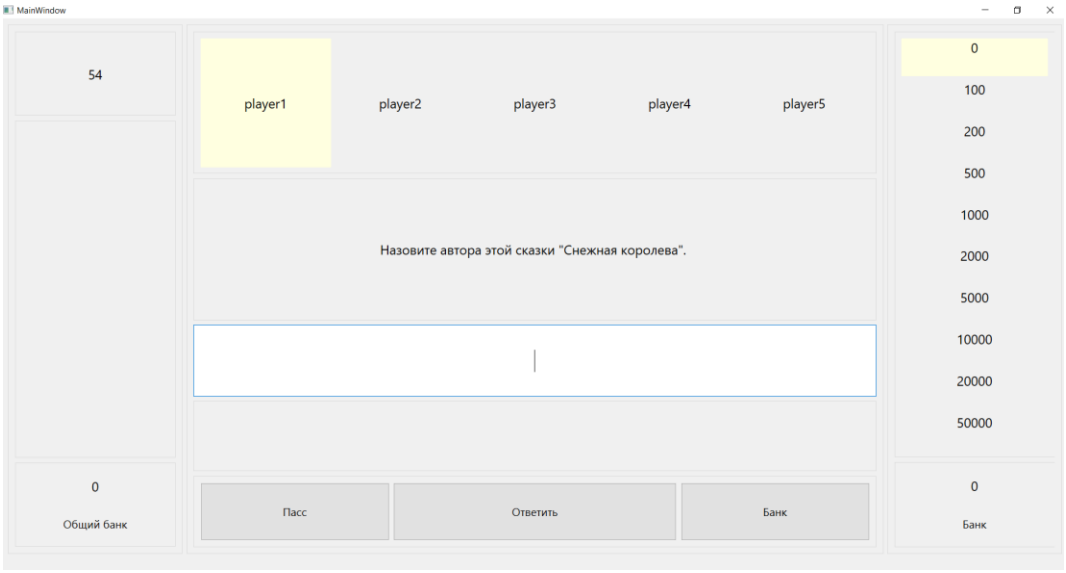

Рисунок 2.1 – Основное окно игры

Начальные данные у игроков состоят из имени и количества верных ответов за текущий раунд. В течение раунда, игроки посредством своих правильных ответов смогут повышать количество верных ответов за раунд. А игрок с наименьшим числом правильных ответов будет изгнан.

2.2 Разработка диаграммы классов

Диаграмма классов данной курсовой работы приведена в приложении А.

2.3 Описание классов

2.3.1 Класс игры

Описание полей класса Game

`Interface* interface` – объект типа `interface`

`qint32 roundTimeLeft` – оставшееся время раунда

`qint32 roundDuration` – длительность раунда

`qint32 currentRound = 0` – настоящий раунд

`qint32 currentQuestionIndex = 0` – настоящий вопрос

`qint32 newSize = 6` – количество игроков, увеличенное на единицу

`qint32 randomIndex` – номер случайного вопроса

`qint32 time` – время до конца раунда

`qint32 minPlayerAnswered` – индекс игрока с наименьшим количеством правильных ответов за раунд

`qint32 totalBankScoreInt` – итоговый банк

`std::vector<Player*> players` – вектор игроков

`Bank* bank = new Bank()` – объект типа `bank`

`std::vector<Question*> questions` – вектор вопросов

`std::vector<quint32> answeredQuestions` вектор кол-ва правильных ответов на вопрос, данными игроками

`QString filePath = "D:/qtFiles/sl_zv.txt"` – путь к файлу с вопросами

`QString filePathAnswers = "D:/qtFiles/sl_zv_otv.txt"` – путь к файлу с ответами

`QTimer *delayedTimer, *timer` – таймер до конца раунда

`int intScoreLabels[10]` – возможные значения банка

Описание методов класса Game

`Game(Ui::MainWindow* ui, qint32 roundDuration)`- конструктор

`void setRandomQuestion()` – задать случайный вопрос

`void addQuestionsFromFile()` – выбрать вопрос из файла

`void startNewRound(int newTime)` – запуск нового раунда

`void answerButtonClicked()`- обработать нажатие на кнопку ответа

`void BankButtonClicked()`- обработать нажатие на кнопку банка

`void toNextPlayer()` – переход хода к следующему игроку

`void passButtonClicked()` – обработать нажатие на кнопку пасс

`void timerSlot()` – проверка завершения раунда

`void checkAnswer(QString userAnswer)` – проверка ответа игрока

`void showDelayedWindow()` – вывод на экран окна завершения раунда
`void showFinishWindow()` – вывод на экран окна завершения игры
`void minPlayerDelete()` – удаление самого слабого игрока
`int findMinScorePlayer()` – поиск самого слабого игрока

2.3.2 Класс интерфейса

Описание полей:
`Ui::MainWindow* ui` – указатель на объект интерфейса окна
`QPushButton* answerButton` – кнопка ответа
`QPushButton* passButton` – кнопка пасс
`QPushButton* bankButton` – кнопка банка
`QLineEdit* userInput` – ответ игрока
`QHBoxLayout *playersLayout` – `groupbox` игроков
`QHBoxLayout *rightAnswerLayout` – `groupbox` правильного ответа
`QHBoxLayout *totalBankLayout` – `groupbox` общего банка
`QLabel* bankLabels[10]` – отображение банка на экране
`QLabel* questionLabel` – отображение вопроса на экране
`QLabel* rightAnswerLabel` – отображение правильного ответа на экарне
`QLabel* timeLabel` – отображения времени на экране
`QLabel* totalBankScore` – отображение значения общего банка на экране
`QLabel* totalBankLabel` – отображение общего банка на экране
`QVBoxLayout *questionLayout` – `groupbox` вопроса
`QVBoxLayout *bankLayout` – `groupbox` банка
`std::vector<QLabel*> playersLabels` – вектор отображения игроков на экране
Описание методов:
`Interface(Ui::MainWindow* ui)` - конструктор
`void setBank(QString newValue)` – установить значение банка
`void setBankZero()` – установить в значение банка ноль
`void setTime(QString newValue)` – установить время
`void addPlayerWidget(Player* player)` – вывод игроков на экран
`void totalBankUpdate(qint32 bank)` – изменение значения финального банка
`QString getUserInput()` – получить ответ, введенный игроком
`void SetQuestion(const QString& string)` – установить вопрос
`void DataOnScreen(std::vector<Player*>& players, qint32 newSize)` – вывод данных на экран
`void playersOnScreen(std::vector<Player*>& players, qint32 newSize)` – вывод игроков на экран
`void deletePlayersFromScreen()` – удаления игроков с экрана

`void rightAnswersBackground()` – изменение цвета заднего фона при ответе на вопрос

`void setRightAnswerOn(QString answer)` – вывод на экран правильного ответа

`void NextPlayerAnswerColor(int index, qint32 newSize)` – изменения цвета заднего фона у отвечающего игрока

`void bankColor(int index)` – изменения цвета заднего фона определенной суммы в банке

`void disableButtons()` – отключение кнопок

`void activateButtons()` – активация кнопок

`void disableBankButton()` – отключение кнопки банка

`void activateBankButton()` – включение кнопки банка

`void rightAnswerOn()` – закраска фона в случае правильного ответа

`void falseAnswerOn()` – закраска фона в случае неправильного ответа

`void clearUserInput()` – очистка строки для ввода ответа

2.3.3 Классы окон

Класс `MainWindow` – в нем происходит основной игровой процесс

Описание полей:

`Ui::MainWindow *ui` – указатель на объект класса `Ui::MainWindow`

`Game *g` – указатель на объект класса `game`

Описание методов:

`MainWindow(QWidget *parent = nullptr)` - конструктор

`~MainWindow()`- деструктор

`void keyPressEvent(QKeyEvent *event) override` – обработка нажатия клавиви `enter` на клавиатуре

`Game* getGame()` – возвращает указатель на объект типа `game`

`void on_button_answer_clicked()`- обработка нажатия на клавишу ответа

`void on_button_bank_clicked()` - обработка нажатия на клавишу банка

`void on_button_pass_clicked()` - обработка нажатия на клавишу пасс

Класс `afterRoundWindow` – служит для вывода на экран данных о самом слабом игроке. На экране имеется кнопка, которая закрывает данное окно и запускает следующий раунд, попутно исключая из игры участника, правильно ответившего на наименьшее число вопросов.

Это окно показано на рисунке 2.2.

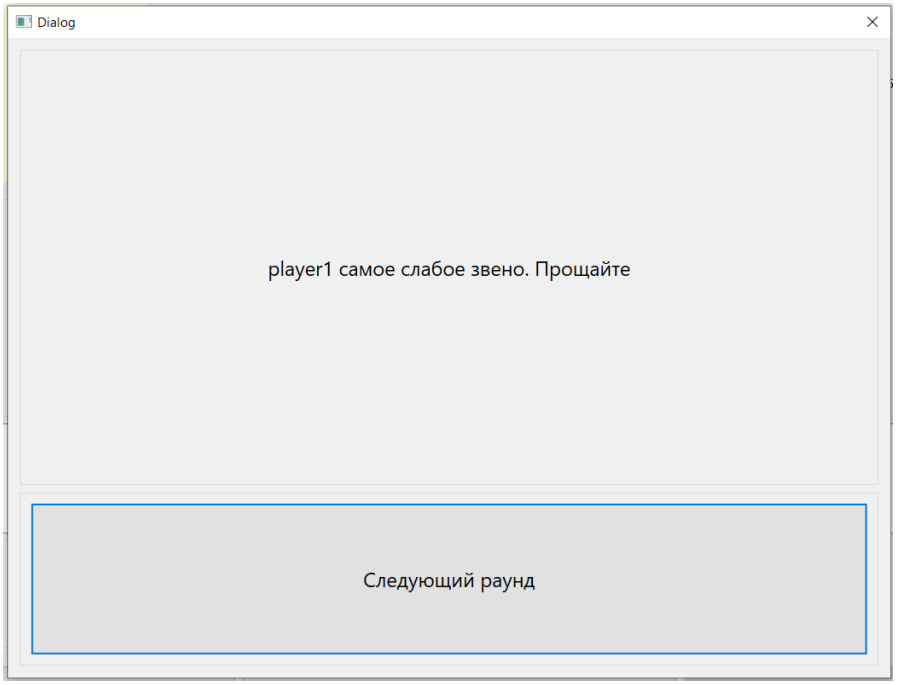

Рисунок 2.2 – Окно “afterRoundWindow”

Описание полей:

`Ui::afterRoundWindow *ui` – указатель на объект класса `Ui::afterRoundWindow`

`Game *g` – указатель на объект класса `game`

`QString weakestLink` – самый слабый игрок по истечению раунда

Описание методов:

`afterRoundWindow(Game* g, QString weakestLink)` - конструктор

`~afterRoundWindow()` - деструктор

`void playerKick()` – вывод на экран информации об удалении игрока

`void on_pushButton_clicked()` – переход к следующему раунду

Класс `finishWindow` – служит для вывода на экран информации о победителе. На экране также имеется кнопка, которая завершает игру и закрывает все окна.

Это окно показано на рисунке 2.3.

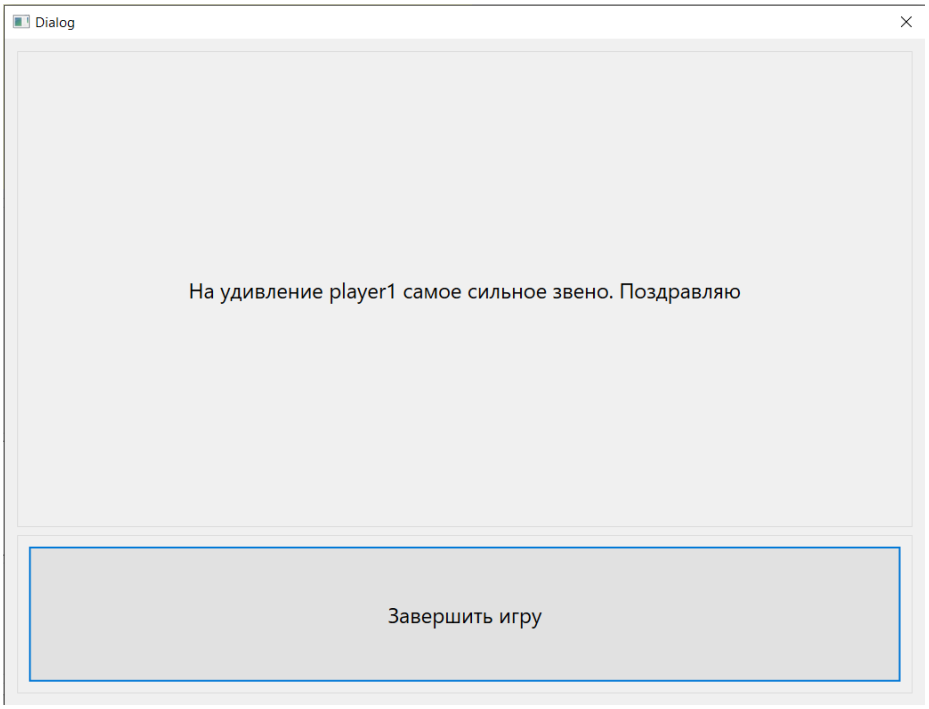

Рисунок 2.3 – Окно “finishWindow”

Описание полей:

`Ui::finishWindow *ui` – указатель на объект класса `Ui::finishWindow`

`Game *g` – указатель на объект класса `game`

`QString weakestLink` – самый сильный игрок по истечению всей игры

Описание методов:

`finishWindow(Game* g, QString weakestLink)` - конструктор

`~finishWindow()` - деструктор

`void playerWin()` – вывод на экран информации о победе игрока

`void on_finishButton_clicked()` – завершение игры

2.3.4 Дополнительные классы

Класс `Bank`

Описание полей:

`qint32 rightAnswersInARow` – кол-во правильных ответов подряд

`qint32 currentRound` - значение текущего раунда

`qint32 total` – итоговый банк

`static const std::vector<qint32> scores` – вектор возможных значений цен ответа на вопрос

Описание методов:

`void newRound()` – новый раунд (очистка банка за раунд, установка нового значения в итоговый банк)

`qint32 getTotal()` – получить значение итогового банка

`void checkAnswer(bool isRight)` – проверка ответа на правильность

`qint32 getRightAnswerInARow()` – получить значение количества правильных ответов подряд

`void setRightAnswerInARow(qint32 value)` – обнуление количества правильных ответов подряд

`qint32 getScore(int index)` – получить интовый эквивалент значения цены, соответствующей кол-ву правильных ответов подряд

Класс `Player`

Описание полей:

`int rightAnswersInThisRound` – количество правильных ответов за текущий раунд у конкретного игрока

`QString name` – имя игрока

Описание методов:

`Player(QString name, int rightAnswersInThisRound = 0)` - конструктор

`void incRightAnswers()` – увеличение правильных ответов в текущем раунде у конкретного игрока

`QString getName()` – вернуть имя у конкретного игрока

`int getRightAnswersInThisRound()` – вернуть значение кол-ва правильных ответов в текущем раунде у конкретного игрока

Класс `Question`

Описание полей:

`QString question` - вопрос

`QString answer` - ответ

Описание методов:

`Question(QString question, QString answer)` - конструктор

`bool checkUserAnswer(QString userAnswer)` – проверка ответа игрока на правильность

`const QString& getQuestion()` вернуть вопрос

`const QString& getAnswer()` – вернуть ответ

3 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

3.1 Разработка схем алгоритмов

Метод `answerButtonClicked()` класса `Game` вызывается после нажатия игроком на кнопку ответа. Данный метод проверяет ответ на правильность и передает ход следующему игроку.

Метод `passButtonClicked ()` класса `Game` вызывается после нажатия игроком на кнопку Пасс. Данный метод обнуляет стрик из правильных ответов и передает ход следующему игроку.

3.2 Разработка алгоритмов

3.2.1 Разработка алгоритма метода startNewRound()

Метод `startNewRound()` класса `Game` начинает новый раунд и выполняет все необходимые для этого операции.

Шаг 1. Начало

Шаг 2. Вызов метода `interface->clearUserInput()` - очистка строки, введенной пользователем

Шаг 3. Уменьшение количества игроков на единицу

Шаг 4. Если количество оставшихся игроков равно единице, то перейти к Шагу 5, в противном случае к Шагу 6

Шаг 5. Вызов метода `showFinishWindow()` - вывести на экран `finishWindow`

Шаг 6. Вызов метода `interface->deletePlayersFromScreen()` - удалить игроков с экрана

Шаг 7. Вызов метода `interface->playersOnScreen()` - вывести на экран обновленное количество игроков (уменьшенное на единицу)

Шаг 8. Вызов метода `interface->setBankZero()` - установить 0 в значение банка текущего раунда

Шаг 9. Установка нового времени раунда

Шаг 10. Переход хода к первому игроку

Шаг 11. Вызов метода `bank->newRound()` - обнуление количества правильных ответов подряд

Шаг 12. Перенос денег из банка за текущий раунд в итоговый банк

Шаг 13. Вызов метода `interface->totalBankUpdate ()` - изменение итогового банка на экране

Шаг 14. Увеличение значения текущего раунда на единицу

Шаг 15. Вызов метода `setRandomQuestion()` - выбор случайного вопроса

Шаг 16. Раскраска игрока, чья очередь отвечать на вопрос

Шаг 17. Раскраска количества правильных ответов подряд, то есть нуля

Шаг 18. Установка таймера на 60 секунд, по истечению данного времени будет удален игрок с наименьшим количеством правильных ответов за раунд

Шаг 19. установка в переменную `time` значения `newTime`

Шаг 20. Создание новой переменной `timer`

Шаг 21. Соединение переменной `timer` с методом `timerSlot`

Шаг 22. Запуск `timer` с частотой обновления равной одной секунде

Шаг 23. Конец

3.2.2 Разработка алгоритма метода checkAnswer()

Метод `checkAnswer()` используется для проверки, введенного игроком, ответа на вопрос на правильность. В случае правильного ответа, данный метод начисляет балл ответившему игроку и увеличивает число денег “На кону”, а в противном случае обнуляет число денег “На кону”.

Опишем данный алгоритм по шагам:

Шаг 1. Начало

Шаг 2. Если пользователь ввел правильный ответ, то перейти к Шагу 2, в противном случае к Шагу 5

Шаг 3. Увеличиваем кол-во правильных ответов у ответившего пользователя с помощью метода `incRightAnswers()`

Шаг 4. Вызываем метод `bank->checkAnswer(true)` - увеличение числа правильных ответов подряд

Шаг 5. Вызываем метод `interface->rightAnswerOn()` - установка зеленого фона на месте для правильного ответа. Переходим к Шагу 7

Шаг 6. Вызываем метод `bank->checkAnswer(true)` - обнуление числа правильных ответов подряд

Шаг 7. Вызываем метод `interface->falseAnswerOn()` - установка красного фона на месте для правильного ответа

Шаг 8. Вызываем метод `interface->setRightAnswerOn()` - вывод на экран правильного ответа

Шаг 9. Конец

4 РЕЗУЛЬТАТЫ РАБОТЫ

В самом начале, пользователь сразу попадает в главное окно, где и происходит основная игра. Пример этого окна показан на рисунке 4.1

Вопросы начинают задаваться от первого игрока, и передаются слева-направо. На первый раунд, как и на все остальные, дается 60 секунд.

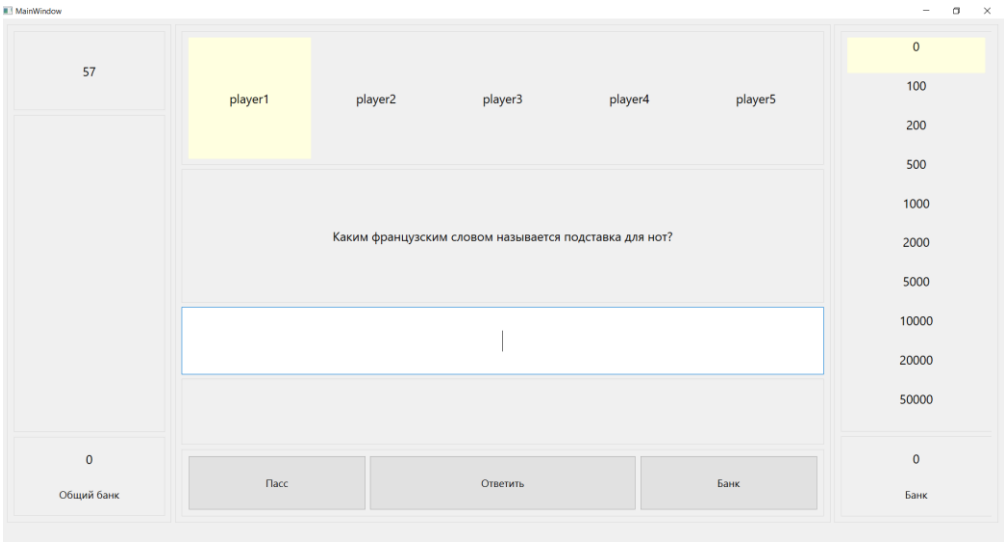

Рисунок 4.1 – Главное окно игры

Когда игрок отвечает верно, на экран выводится правильный ответ на зеленом фоне, подтверждающий правильность ответа. Значение суммы “На кону” увеличивается, а ход передается следующему игроку. Пример правильного ответа представлен на рисунке 4.2.

Рисунок 4.2 – Правильный ответ на вопрос

В случае неправильного ответа на вопрос, ответ, выводимый на экран, появляется на ярко-красном фоне, давая игроку понять, что он допустил ошибку. Сумма “На кону” сбрасывается до нулевого значения, а ход переходит следующему игроку. Пример неправильного ответа представлен на рисунке 4.3.

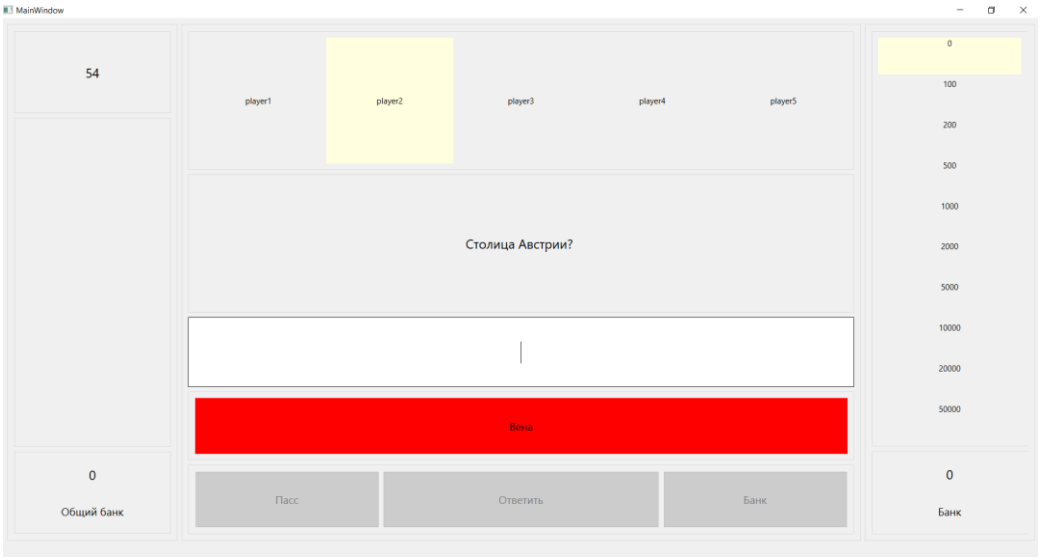

Рисунок 4.3 – Неправильный ответ на вопрос

По истечению времени, на экране появится окно `afterRoundWindow` на котором будут изображены имя игрока, который покидает игру, и кнопка, переносящая нас в следующий раунд. При этом весь банк, накопленный за предыдущий раунд, будет перенесен в Итоговый банк, сумма которого не сгорает. Пример данного окна приведен на рисунке 4.4.

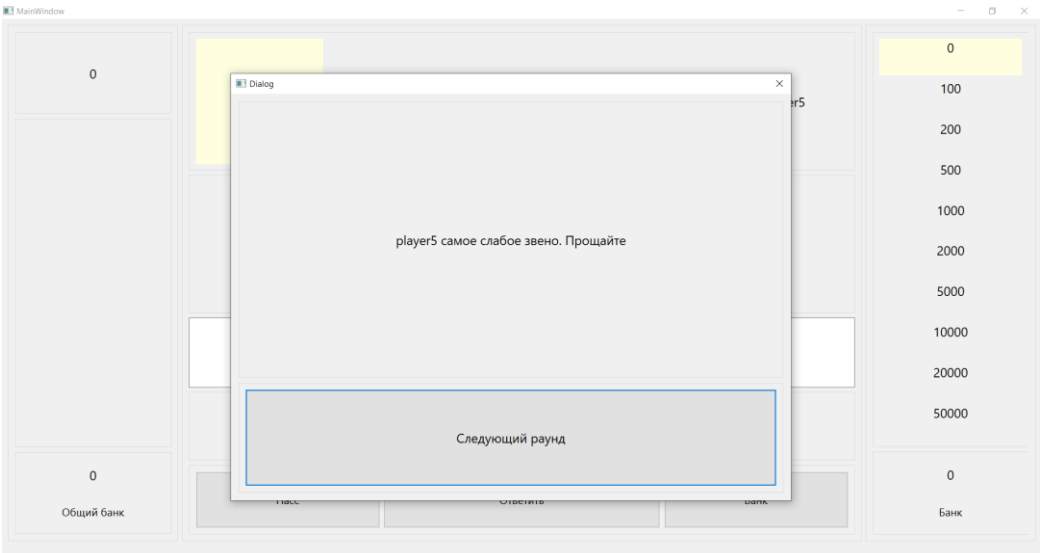

Рисунок 4.4 – Окно после раунда

По истечению всех раундов остается только один игрок, он и является победителем. На экране появляется окно `finisfWindow` на котором изображены имя победителя и кнопка, при нажатии на которую – игра завершится. Пример данного окна приведен на рисунке 4.5.

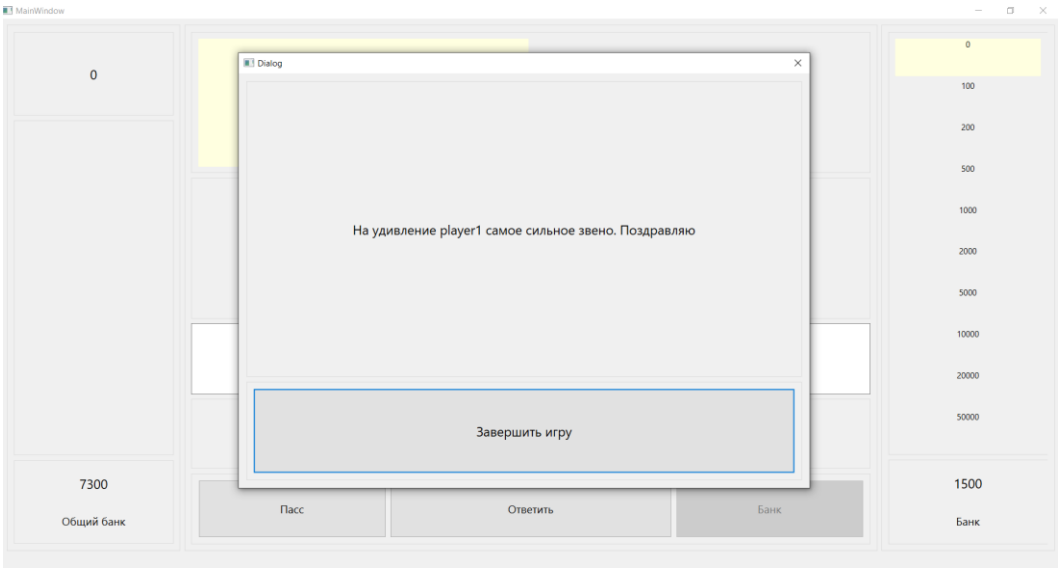

Рисунок 4.5 – Финальное окно

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы были закреплены знания основ программирования и алгоритмизации. Были приобретены знания в объектно-ориентированном программировании. Также я познакомился с созданием полноценных проектов на фреймворке Qt и убедился, что он идеально подходит для создания несложных проектов с удобным интерфейсом.

В процессе разработки игры “Слабое звено” были реализованы алгоритмы передачи хода, ответа на вопрос, увеличение банка, изгнания игрока и т. д. Как упоминалось раннее, моя игра является неполной версией оригинальной игры “Слабое звено”. Однако она открыта для модернизации. Например, можно добавить механику голосования, изменить время на конкретный раунд, добавить больше игроков и вопросов и много чего еще.

Игра оказалась нетрудной в реализации, имеет удобный интерфейс. В нее можно со своими друзьями с одного компьютера.

На основе полученных знаний при создании игры можно будет создавать более проработанные и интересные игры, например стратегии или викторины.

Создание игры было выполнено на ОС Windows 10, используя среду разработки QtCreator.

Код программы приведен в приложении Г.

СПИСОК ЛИТЕРАТУРЫ

[1] Стивен Прата. Язык программирования C++. Лекции и упражнения: учеб. пособие / С. Прата – СПб.: ООО «ДиаСофтЮП», 2003. – 1104с.
[2] Конструирование программ и языки программирования: метод. указания по курсовому проектированию – А. В. Бушкевич, А. М. Ковальчук, И. В. Лукьянова. – Минск : БГУИР, 2009.
[3] Qt Documentation [Электронный ресурс]. -Электронные данные. -Режим доступа: https://doc.qt.io/ Дата доступа: 24.10.2023
[4] UML-диаграммы классов [Электронный ресурс]. -Электронные данные. -Режим доступа: https://habr.com/ru/articles/150041/ Дата доступа 27.10.2023
[5] СТП 01–2017. Дипломные проекты (работы): общие требования. – Введ. 2017–01–01. – [Электронный ресурс].– 2017– Режим доступа: http://library.bsuir.by/online/showpage.jsp?PageID=86151 – Дата доступа: 23.11.2023

ПРИЛОЖЕНИЕ А

(обязательное)

Диаграмма классов

ПРИЛОЖЕНИЕ Б

(обязательное)

Схема метода answerButtonClicked ()

ПРИЛОЖЕНИЕ В

(обязательное)

Схема метода passButtonClicked ()

ПРИЛОЖЕНИЕ Г

(обязательное)

Код программы

ПРИЛОЖЕНИЕ Д

(обязательное)

Ведомость документов