**СОДЕРЖАНИЕ**

ВВЕДЕНИE……………………………………………………………….……….2

1 ПОСТАНОВКА ЗАДАЧИ……………………………………………..……….3

2 ОБЗОР ЛИТЕРАТУРЫ…………………………………………………...…….4

2.1 Обзор методов и алгоритмов решения поставленной задачи…….…….4

2.2 Обзор аналогов приложения……………………………………………...4

2.3 Вывод по аналогам………………………………………………………...6

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ……………….……………….7

3.1 Структура входных и выходных данных………………………………...7

3.2 Разработка диаграммы классов………...…………………………………7

3.3 Описание классов……….…………………………………………………5

3.3.1 Основные классы………….………………………………………….5

3.3.2 Класс контейнера...…………………………………………………...9

3.3.3 Другие классы……………………………………………………….10

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ………………………………10

4.1 Разработка схем алгоритмов………………………...……………….......10

4.2 Разработка алгоритмов…………………………...………………………11

5. РЕЗУЛЬТАТЫ РАБОТЫ……………………………………………………...12

ЗАКЛЮЧЕНИЕ……………………………………………………………….....14

СПИСОК ЛИТЕРАТУРЫ……………………………………………………….15

ПРИЛОЖЕНИЕ А………………………………………………………………..16

ПРИЛОЖЕНИЕ Б………………………………………………………………..17

ПРИЛОЖЕНИЕ В………………………………………………………………..18

ПРИЛОЖЕНИЕ Г………………………………………………………………..19

**ВВЕДЕНИЕ**

Язык C++ является довольно мощным инструментом для умелого разработчика. Сам язык имеет 3 парадигмы, за счёт этого можно по-разному выстраивать структуру программы. Он сочетает свойства как высокоуровневых, так и низкоуровневых языков программирования.

Если сравнивать данный язык с C, то можно выделить разные парадигмы: C++ в первую очередь был создан для обретения других парадигм, речь идёт про ООП и обобщённое программирование.

Данный язык широко используется для разработки ПО, написания игр, создание операционных систем, написания драйверов и решения множества других задач.

Язык имеет богатую стандартную библиотеку, включающую в себя множество алгоритмов, контейнеров, ввод-вывод, строковые, языковую поддержку, регулярные выражения, поддержку многопоточности и многое другое.

Один из фреймворков под названием Qt позволяет реализовать красивый и удобный графический интерфейс для разработки ПО.

Таким образом для разработки небольшой компьютерной версии игры “*Покер*” Техасской вариации язык C++ подходит идеально.

**1 ПОСТАНОВКА ЗАДАЧИ**

В первую очередь игра должна иметь понятный интерфейс, реализованный на фреймворке QT и имеющий все необходимые функции для игрового процесса. В программе должна быть реализована сама игровая логика и возможность игры с ботами. Количество ботов необходимо выбирать перед началом игры.

Для взаимодействия необходимо реализовать все особенности покера, такие как ставки, банк, большие и малые блайнды. Также должна быть реализована колода карт, ее перетасовка, раздача игрокам и на стол. Необходимо также дать игроку возможность совершать ставку, пасовать, выжидать или сравнивать ставку с поставленной ранее в раунде.

Для отслеживания победившего в раздаче необходимо написать функцию для определения покерной комбинации конкретного игрока и сравнения ее стоимости с комбинациями других участников, учитывая при равенстве старшинство карты в руке.

**2 ОБЗОР ЛИТЕРАТУРЫ**

**2.1 Обзор методов и алгоритмов решения поставленной задачи**

Для хранения данных в процессе игры использованы шаблонные контейнеры vector<T>, которые в основном хранят значения карт и всех игроков за столом.

Для метода определения карточной комбинации была использована библиотека algorithm, в частности сортировка для шаблонного контейнера vector<T> содержащего карты по значениям и мастям sort(T firstcard, T secontcard, bool Comparator), использующая самописные компараторы.

Непосредственно для организации самого игрового процесса использованы возможности фреймворка QT, а именно сигналы и слоты, позволяющие осуществлять событийно-ориентированное программирование, а так же простые циклические алгоритмы раздачи и ожидания действий игрока или ботов.

**2.2 Обзор аналогов приложения**

Перед началом работы необходимо было изучить представленные в общем доступе аналоги игры в покер для определения необходимых функций и особенностей выбранной темы

* + 1. **Приложение “Покер” на мобильные устройства**

****

Рис. 2.1 Приложение “Покер”

В данном приложении(рис. 2.1) реализована игра по сети с другими людьми. Ставки делаются не настоящими деньгами, а внутриигровой валютой. Простой интерфейс, небольшие иконки, но играть приятно.

Из минусов можно отметить отсутствие автоподбора игр и игры с ботами без интернета. Так как я собираюсь создать исключительно офлайн игру в покер, для меня имеет значение только второй минус.

* + 1. **Приложение “World Poker Club”**



Рис. 2.2 Главное меню приложения “World Poker Club”

****

Рис. 2.3 Игровое поле приложения “World Poker Club”

Приложение явно имеет более коммерческую структуру и большой охват аудитории. В приложении опять же существуют онлайн игры и турниры, различные виды покера для игры онлайн. Система прогрессии и автоподбор стола(рис. 2.2). Имеется приятный и удобный интерфейс игрового поля, на котором размещена вся нужная игроку информация (рис 2.3).

Это пример хорошего или даже почти идеального приложения на тему покера. Единственная проблема снова заключается в отсутствии возможности играть в офлайн режиме с ботами.

**2.3 Выводы по аналогам**

Делая вывод на основе анализа приведенных приложений можно сказать, что написанное мной приложение в итоге не станет конкурентом таким приложениям, так как действительна важна возможность игры в онлайне с другими людьми, однако же явным достоинством станет возможность игры в офлайне с ботами при условии отсутствия интернета.

С помощью QT Creator возможно создать подходящий интерфейс и функционал, оформить приложение и подарить пользователю приятный игровой опыт. Будет создано окно с игровым столом, на котором, как и в аналогичных приложениях, будут разворачиваться партии в покер. На нем обязательно должны быть карты крупье, баланс игроков, банк. Должны показываться ставки, статусы игроков на каждом шагу игры. Непосредственно сами объекты взаимодействия с игрой, такие как кнопки и спинбоксы легко реализовать в QT Creator, используя встроенный редактор форм, сигналы и слоты.

**3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

В данном разделе описываются входные и выходные данные программы, диаграмма классов, а также приводится описание используемых классов и их методов.

**3.1 Разработка диаграммы классов**

Диаграмма классов для курсового проекта приведена в Приложении А.

**3.2 Описание классов**

**3.2.1 Классы игры**

Класс Card – класс карты

Поля:

string suit – масть карты.

int value – ценность карты.

Методы и операторы:

int GetValue() – получить ценность

string GetSuit() – получить масть

void SetCard(int value, string suit) – установить карту

Класс Deck – класс колоды

Поля:

vector<Card> \*deck – указатель на контейнер колоды

Методы и операторы:

Deck() – конструктор по умолчанию

~Deck() – деструктор

vector<Card>\* GetDeck() – получение указателя на контейнер колоды

void SetDeck() – создание колоды

void reshuffle() – перемешивание колоды

Класс Entity – класс участника игры

Поля:

Card card1 – первая карта

Card card2 – вторая карта

int balance – стек(баланс)

Методы и операторы:

void SetBalance(int balance) – установка баланса

int GetBalance() – получение баланса

Card GetCard1() – получение первой карты

Card GetCard2() – получение второй карты

void SetCards(Card card1, Card card2) – установка значений карт

virtual void Raise() – повышение ставки

virtual void Check() – ожидание ставок

virtual void Call() - уравнивание ставки

virtual void Pass() – пас

void ShowCards() – вскрытие карт на столе

Класс Bot – класс наследуемый от Entity, класс бота

Методы и операторы:

void Raise() – повышение ставки

void Check() – ожидание ставок

void Call() – уравнивание ставки

void Pass() – пас

Класс Player – класс наследуемый от Entity, класс игрока

Методы и операторы:

void Raise() – повышение ставки

void Check() – ожидание ставок

void Call() – уравнивание ставки

void Pass() - пас

Класс Table – класс стола

Поля:

std::vector<Card> \*DeskCards – указатель на контейнер карт на столе

int balance – общий баланс банка в текущем раунде

Методы и операторы:

Table() – конструктор по умолчанию

~Table() – деструктор

std::vector<Card>\* GetDeskCard() – получение карт на столе

void SetDeskCards(std::vector<Card>\* deskcards) – раздача карт на столе

void ShowFlopp() – вскрытие первых трех карт на столе

void ShowTern() – вскрытие 4 карты на столе

void ShowRiver() – вскрытие 5 карты на столе

Класс Interface – класс интерфейса и взаимодействий

Поля:

Deck deck – колода карт

Table table – стол

std::vector<Entity\*> \*Players – указатель на контейнер указателей на участников игры

Методы и операторы:

Interface() – конструктор по умолчанию

~Interface() – деструктор

std::vector<Entity\*>\* GetPlayers() – получение контейнера игроков

int BotsNumber – количество ботов

Deck GetDeck() – получение колоды

Table GetTable() получение стола

void SetDeck(Deck deck) – установка колоды

void SetPlayers() – установка игроков

void Distribution() – метод раздачи карт

void Blinds() – ставка большого и алого блайнда

void CombinationCheck() – проверка покерной комбинации

**3.2.2 Класс контейнера**

В проекте используется шаблонный список двунаправленное кольцо Ring<T> для хранения игрового поля и игроков. Стоит отметить что Node<T> - узел, хранящий данные, а также указатель на предыдущий и следующий элемент.

Описание полей класса Ring<T>:

Node<T>\* first – указатель на первый элемент кольца

int size – размер кольца

Описание методов класса Ring<T>:

Ring() – конструктор по умолчанию

Ring(const Ring& ring) – конструктор копирования

~Ring() – деструктор

void push(T value) – добавление элемента в кольцо

void popValue(T value) – удалить элемент в соответствии с заданным аргументом

Node<T>\* getValue(T value) – возвращает элемент в соответствии с заданным аргументом

Node<T>\* getFirst() – возвращает первый элемент кольца

Node<T>\* getNext(Node<T>\* current) – возвращает элемент после current

Node<T>\* getNextN(Node<T>\* current, int N) – возвращает через N элемент после current

Node<T>\* getPrev(Node<T>\* current) – возвращает элемент перед current

Node<T>\* getPrevN(Node<T>\* current, int N) – возвращает через N элемент перед current

int&getSize() – возвращает размер кольца

**3.2.3 Другие классы**

Класс MainWindow – класс наследуемый от QMainWindow, класс главного окна

Поля:

Ui::MainWindow \*ui – указатель на графический интерфейс

Interface interface – объект интерфейса окна

StartMenu startmenu – объект окна стартового меню

Методы и операторы:

MainWindow(QWidget \*parent = nullptr) – конструктор по умолчанию

~MainWindow() - деструктор

void SetBalance(int balance, Entity player) – установка баланса игрока

void SetCard(Entity player) – установка карт игрока

void StartingTasks() – выполнение стартовых задач(установка игроков, создание колоды и т. д.)

Слоты:

void ExecWindow() – вывод главного окна

void GetNumber(int aboba) – получение количества игроков из стартового меню

Класс StartMenu – класс наследуемы от QDialog, класс диалогового окна

Поля:

Ui::StartMenu \*ui – указатель на графический интерфейс окна

Методы и операторы:

explicit StartMenu(QWidget \*parent = nullptr) – конструктор по умолчанию

~StartMenu() – деструктор

int GetSpin() – получить введенное количество игроков из спинбокса

**4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

**4.1 Разработка схем алгоритмов**

Алгоритм по шагам функции PokerCombination

result PokerCombination(std::vector<Card> cards) – объявление функции

Функция возвращает структуру, содержащую стоимость комбинации и старшие карты

Алгоритм по шагам:

Шаг 1.Начало

Шаг 2. Входные данные:

std::vector<Card> cards – шаблонный контейнер вектор, содержащий ровно 7 карт, включающие в себя карты со стола и карты в руке у конкретного игрока

Шаг 3. Сортировка функцией std::sort() библиотеки algorithm вектора карт по масти

Шаг 4. Прохождение циклом по отсортированному векторы и подсчет количества карт каждой масти

Шаг 5. Если количество карт какой-либо масти больше или равно 5, значит есть флеш, еще одним циклом находим в нем старшую карту

Шаг 6. Сортируем вектор по возрастанию стоимости карт функцией std::sort() и проходим циклом, проверяя возрастание стоимости карт на 1 единицу, то есть проверяем наличие стрита

Шаг 7. Если найден стрит и флеш одновременно, значит проверим старшинство карты стрита на флеш рояль, если нет, то комбинацией является стрит флеш. Если найден стрит и не найден флеш, значит комбинация стрит. Также определим старшую карту в процессе.

Шаг 8. Проверим циклом вхождение карт с одинаковой стоимостью 2, 3, 4 раза при условии, что до этого не была обнаружена более крупная комбинация. Найдем соответственно комбинацию каре, сет или пара, запомним значение карты.

Шаг 9. Определим наличие фулл хауса, если не найдена более дорогая комбинация до этого. Если было найдено вхождение сета, то пройдем циклом и найдем, есть ли помимо сета еще и пара. Если да, то это фулл хаус, если нет, то это сет. Запомним стоимость обеих карт в комбинации

Шаг 10. Таким же образом при вхождении пары попробуем найти вторую пару циклом. Если найдена, то комбинация две пары, если нет, то пара. Также запомним значение обеих карт

Шаг 11. Если не найдена ни одна комбинация, то циклом найдем просто максимальный элемент, то есть старшую карту и запомним ее.

**4.2 Разработка алгоритмов**

Функция сравнения стоимости комбинации и выявления победителя

int Comparison(std::shared\_ptr<std::vector<Entity\*>> ActivePlayers, Table table)

Блок-схема приведена в приложении Б

Метод класса Interface, совершающий раздачу карт из колоды игроками

void Interface::Distribution()

Блок-схема приведена в приложении В

**5. РЕЗУЛЬТАТЫ РАБОТЫ**

При запуске игры в первую очередь открывается стартовое окно с выбором количества оппонентов. В моем приложении оппонентами являются боты, поэтому пользователю необходимо выбрать количество ботов, с которыми он будет сидеть за столом. По правилам покера количество игроков может изменяться от 2 до 9, соответственно максимальное число ботов – 8.

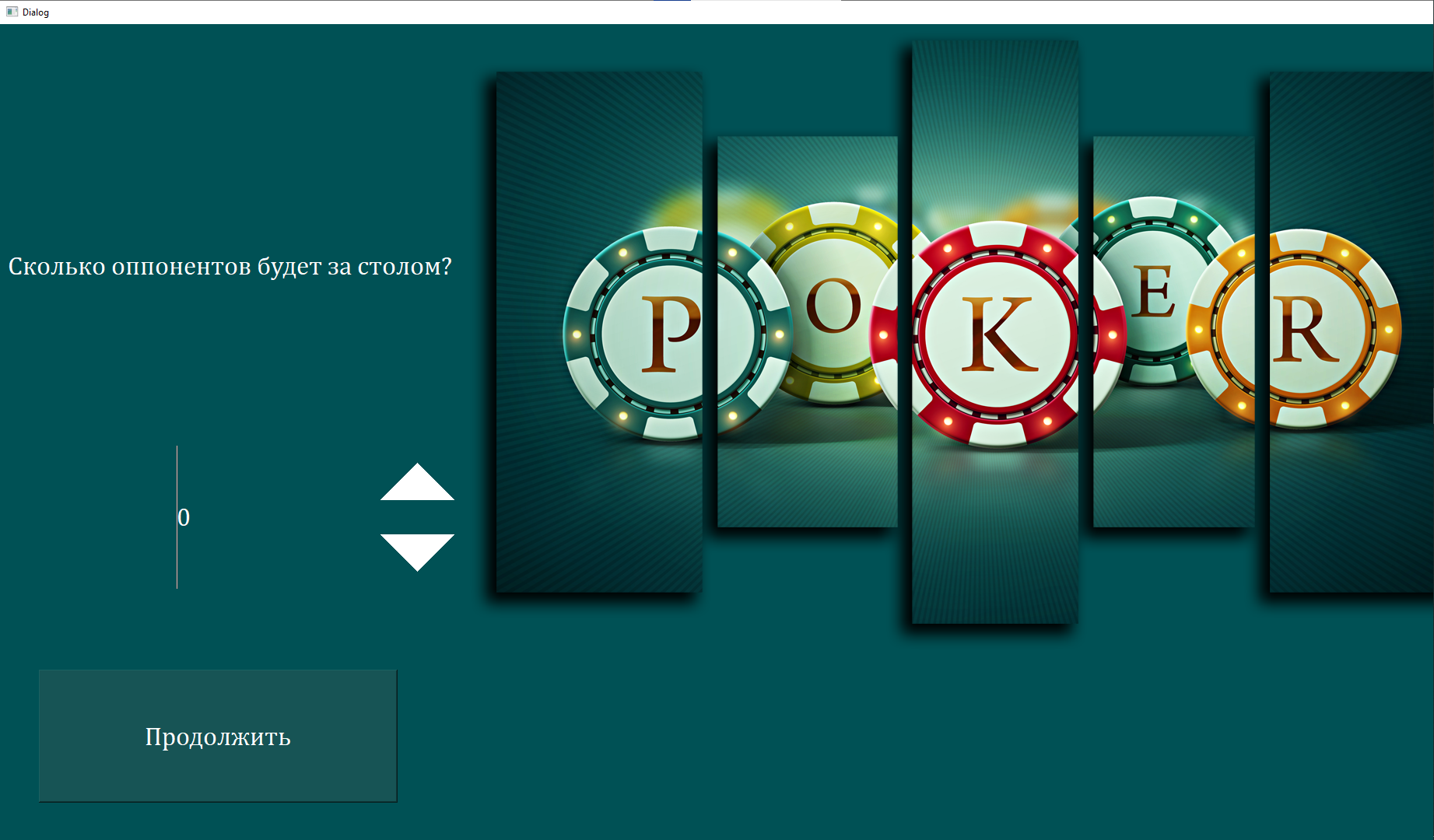
****

Рис 5.1 Стартовое меню

Как только пользователь выбрал количество игроков, он видит перед собой непосредственно игровой стол с оппонентами (рис. 5.2). На поле отображаются скрытые карты игроков, участвующих в раздаче, банк, карты крупье, карты в руке игрока и кнопки взаимодействия.

По нажатию кнопки начать раздачу, начинают приниматься ставки, а значит начинается и сама игра.



Рис. 5.2 Игровое поле

Как только начинается раздача, в руке игрока появляются карты и выставляются малый и большой блайнды (рис. 5.3). Вследствие действий игрока и ботов раздача завершается чьей-то победой и ля продолжения необходимо снова нажать на кнопку “Начать раздачу”. Таким образом игра ведется либо до выбивания всех оппонентов, либо до поражения игрока.



Рис. 5.3 Игровой процесс

**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. Для этого фреймворка создана полная и понятная документация, что позволяет быстро научится пользоваться его возможностями.

В процессе разработки игры “Покер” были реализованы достаточно сложные логические функции проверки комбинации и сравнения стоимости комбинаций всех игроков, были написаны методы для раздачи карт, перетасовки колоды, а также сам цикл игрового процесса. Было написано односвязное кольцо как динамическая структура данных для хранения игроков, участвующих в игре. Также был создан приятный минималистичный интерфейс, дающий всю необходимую для пользователя информацию и возможности для взаимодействия с игрой. Моя версия покера открыта для модификации, например добавление мультиплеера, улучшения игрового интеллекта ботов на основе математических расчетов и так далее.

На основе полученных знаний при создании игры можно несложно будет создать приложения, где от пользователя будет требоваться ввод каких-либо данных и нажимание кнопок на экране. Такими примерами могут быть приложение для ведения заметок, ежедневник и т.д.

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

Код программы приведен в приложении Г.

**СПИСОК ЛИТЕРАТУРЫ**

[1] Шилдт Г. С++ базовый курс / Г. Шилдт - 3-е изд. М. : Вильямс , 2007 – 624с.

[2] Страуструп Б. Язык программирования С++ : специальное издание / Б. Страуструп : пер. с англ. – Москва : Бином , 2006 – 1104с.

[3] Конструирование программ и языки программирования: метод. Указания по К65 курсовому проектированию для студ. I-40 02 01 “Вычислительные машины, системы и сети” для всех форм обуч. / сост. А. В. Бушкевич , А. М. Ковальчук , И. В. Лукьянова. - Минск: БГУИР , 2009.

[4] Объектно-ориентированное программирование на языке С++:учеб. Пособие /Ю. А. Луцик , В. Н. Комличенко. – Минск: БГУИР , 2008.

[5] Стивен Прата. Язык программирования С++. Лекции и упражнения: учеб. пособие / С. Прата – СПб.: ООО «ДиаСофтЮП», 2003. – 1104с.

**ПРИЛОЖЕНИЕ А**

(обязательное)

(Диаграмма классов)

**ПРИЛОЖЕНИЕ Б**

(обязательное)

(блок-схема)

**ПРИЛОЖЕНИЕ В**

(обязательное)

(блок-схема)

**ПРИЛОЖЕНИЕ Г**

(обязательное)

(Полный код программы)