Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Кафедра информационных технологий и систем

**Сетевая игра «Покер»**

Курсовая работа

по дисциплине:

«Операционные системы»

Разработал:

Студент группы 0091

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_».\_\_\_\_\_.2022г.

Проверил:

Фуфин М.Н.\_\_\_\_\_\_

«\_\_».\_\_\_\_\_.2022г.

**Великий Новгород**

**2022**

Оглавление

[ВВЕДЕНИЕ 3](#_Toc103672176)

[РАЗРАБОТКА ПРОЕКТА. 4](#_Toc103672177)

[1. Постановка задачи. 4](#_Toc103672178)

[2. Выбор инструментальных средств. 4](#_Toc103672179)

[3. Выбор способа взаимодействия потоков. 4](#_Toc103672180)

[4. Выбор библиотеки для реализации многопоточной модели 4](#_Toc103672181)

[5. Алгоритм решения задачи. 4](#_Toc103672182)

[6. Вывод 5](#_Toc103672183)

# ВВЕДЕНИЕ

Целью данной работы является создание сетевой игры в покер от 2-х до 6 игроков. Разработанные программы под системы Windows.

# РАЗРАБОТКА ПРОЕКТА.

## **1. Постановка задачи.**

Задачей данной курсовой работы является разработка параллельного приложения сервера и приложения клиента, которые будут обмениваться информацией по ходу игры. Приложение сервера использует многопоточность для подключения игроков и синхронизации их действий, а остальные действия выполняются в одном главном потоке. Приложение клиента работает только на главном потоке. Игра должна работать под управлением операционной системы семейства Windows 10. Игра будет иметь графический интерфейс, нарисованный символами ascii. В приложении будет реализовано простое подключение игрока к серверу, анимированная заставка, игровой интерфейс. Приложение сервера будет лишь спрашивать кол-во игроков, участвующих в игре, и выводить строчки информации, которые отправляет клиентам.

## **2. Выбор инструментальных средств.**

Программировать будем в Visual studio 2022 на языке C++.

## **3. Выбор способа взаимодействия потоков.**

Я выбрал мьютексы, так как их поддерживает C++.

**4. Выбор библиотеки для реализации многопоточной модели.**  
Для реализации многопоточности использовалась стандартная библиотека “thread”.

## **5. Алгоритм решения задачи.**

Правила игры следующие. В одной игре может принимать участие от двух до шести игроков и человек, на компьютере которого запущен сервер(конечно, приложение сервера может быть запущенно на компьютере какого-нибудь игрока). Вначале запускается приложение сервера. При запуске оно спрашивает ip, на котором будет работать. Для этого запускаем командную строку(cmd), вводим команду ipconfig, находим "Адаптер Ethernet Ethernet", а в нём поле "IPv4-адрес". Это и есть наш ip, который нужно ввести. Можно конешно ввести символ '!' и нажать 'enter'. Тогда сервер будет использовать ip поумолчанию(но очень сомневаюсь, что он будет схожим с вашим, т.к. в таком случае сервер не заработает). После ip нас попросят ввести кол-во игроков. Вводим число игроков и сервер выдаёт нам сообщение "i am ready", что означает, что можно подключать игроков. При запуске приложения клиента, он попросит ввести ник, а после ip(ip должен быть такой же, как и тот, что мы ввели серверу). Если всё пройдёт успешно, то сервер выдаст сообщение "<ник> connected"(где ник это ник игрока), а у игрока начнётся анимированная заставка, по истечению которой выведется либо "Not enough players to start the game, please wait till more players will join", либо "Are you ready to begin?"(посмотрев код, вы можете подумать, что может вывесться ещё и сообщение "The game is going, please wait till it will over". Скажем так, ранее я планировал сделать так, что люди могут попробовать подключиться прямо во время игры и выведется это сообщение, но я отказался от этой идеи). Как только к серверу подключиться нужное кол-во игроков и они нажмут на любую кнопку после сообщения "Are you ready to begin?", начнётся игра. По началу самой игры у игроков прорисовывается интерфейс. Справа будет написанно кол-во денег, выйграшей и пройгрышей, максимальная ставка на данный момент(по-умолчанию 10$) и 2 карты. Посередине нарисован стол, слева от него крупъе, над ним число раундов, над столом распологаются игроки(их число зависит от кол-ва этих самых игроков). над игроками написан порядок их хода, ник, число побед и поражений, деньги. Слева сверху написанно то, каким ходит сам игрок и выводятся информационные сообщения(по типу: "<ник> made a call"). По мере хода игры, сервер будет выводить команды, которые отправляются игрокам(для упрощения отладки и чтоб держатель сервера видел, что там происходит). По правилам покера первые два игрока делают малый и большой блайд(короче первый игрок ставит 5$, а второй 10$). Потом каждый игрок выбирает, что он хочет сделать в свой ход: fold(скинуть карты и отказаться от игры, но можно посмотреть её до конца), check(согласиться с нынешней ставкой(типа если максимальная ставка 10$, а игрок поставил 5$, то он ставит ещё 5$)), bet(увеличить максимальную ставку). Так продолжается 4 раунда. За это время на стол выкладываются карты(3 карты после 1-го раунда, 1 карта после 2-го и 3-го). В конце приложение клиента ищет комбинации карт для своего игрока составляя его кол-во очков. Комбинации карт взял с этого сайта: https://academypoker.ru/combination.html. Все очки игроков отправляются на сервер и ищется их самое большое кол-во. У кого больше очков, тот и выйгрывает весь банк. Если у 2-х игроков одинаковое число очков, то банк делится между ними поровну. Клиент и сервер выводят ник(-и) победителя(-ей). После определения победителя клиент ставиться на паузу, а при нажатии любой кнопки, приложение клиента закрывается. Сервер же выводит "The end.", и после этого программу можно закрыть вручную. Сама она не закроется, так как(по-моему мнению) ждёт завершения работы потоков. Да, этот нюанс я не доделал. Вообще у меня ещё в планах было сделать сохранение информации о клиентах в отдельный файл, чтоб при запуске приложения клиента, пользователь вводил свой ник и пароль и ему давались данные о его деньгах, победах и пройгрышах. Сейчас у каждого пользователя по 200$ и по 0 побед и пройгрышей(Ну, будем считать, что это открытая бэта, а не я полинился доделать до конца).

## **6. Вывод.**

Таким образом, для реализации сетевой игры “Покер” основной ОС выбрана Windows 10 и языком программирования C++. Для синхронизации потоков были выбраны мьютексы. Графический интерфейс строится из символов ascii. В качестве библиотеки, реализующей работу потоков был выбран thread.

# РЕАЛИЗАЦИЯ ПРОЕКТА.

## **1. Структура приложения и описание программной реализации приложения.**

Приложение сервера состоит из:

* Структуры clientInfo, которая хранит информацию о клиенте.
* Структуры gamers, которая хранит информацию о клиенте, как игроке (эти структуры почти одинаковы, но нужны обе).
* Функции getClientInfoFromFile, которая берёт информацию о клиентах из файла, в котором эта информация храниться (Как я уже писал в Алгоритме решения задачи, раньше я хотел сделать сохранение информации о клиентах, но потом отказался. Работоспособность функции я не проверял).
* Функции check, которая проверяет по нику клиента, есть ли о нём информация в файле или нет.
* Функции workingWithClient, которая принимает ник клиента, проверяет, есть ли о нём информация в файле, с помощью функции check и, если да, то просит ввести пароль от его “аккаунта”, чтоб получить эту информацию (но это часть функции никогда не срабатывает, так как информации в файле нет). Если функция не находит информацию о клиенте, то просит ввести новый пароль и даёт ему стартовый набор (200$ денег, 0 побед и 0 поражений). После информация о клиентах заносится в массив newGamers, в котором храниться информация о клиентах, которые будут учавствовать в игре, и отправляет эту информацию самим клиентам. Дальше функция ждёт, пока к серверу подключиться нужное количество клиентов
* Функции conectingClientsToServer, которая сигнализирует о начале работы сервера, устанавливает связь с клиентами и запускает функцию workingWithClient для работы с только что подключившимся клиентом в отдельном потоке.
* Функции isTheClientReady проверяет, готов ли клиент (принимает от него короткое сообщение и увеличивает счётчик готовых клиентов).
* Функции testTempNumb. Если честно, то она нигде не используется (да, да, я дурачок, который заметил это только сейчас, можете посмеяться).
* Функции checkFoldGamer, которая проверяет, сбросил ли карты во время игры клиент.
* Функции sendMsgToAllPlayers, которая комплектует и отправляет сообщение всем клиентам, кроме одного указанного нами (да, над названием нужно поработать).
* Функции PockerGame, в которой происходит всё действо игры в покер.
* Функции main, которая запускает функцию getClientInfoFromFile, узнаёт ip и кол-во игроков, подготавливает сервер к работе, запускает функцию PockerGame, пишет “The end.” в конце игры и ждёт окончания работы запущенных потоков (которое не происходит).

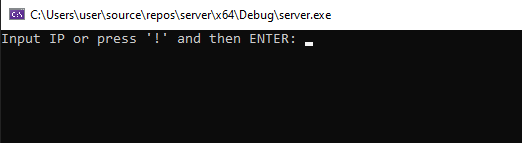
Приложение клиента состоит из:

* Структуры gamers, которая ранит информацию о врагах клиента.
* Функции gameStartScreen, которая выводит красивую анимацию при подключении к серверу.
* Функции whatCard, которая определяет, какое изображение карты нужно вывести на экран вовремя игры (они располагаются справа снизу во время игры, на скриншотах увидите).
* Функции whatToDraw, которая определяет, сколько врагов, сидящих за столом, нужно нарисовать (тоже на скриншотах увидите).
* Функции gamePlayScreen, которая рисует весь интерфейс, который видит игрок во время игры.
* Функции checkSuit, которая помогает определить комбинацию карт, под названием “ Роял-флеш”.
* Функции proverkaMsg, которая проверяет почти все сообщения, которые могут прийти от сервера и реагирует на них.
* Функции game, которая принимает информацию о картах клиента и о его врагах от сервера, постоянно запускает функции gamePlayScreen и proverkaMsg, а также принимает сообщения от сервера и передаёт их функции proverkaMsg.
* Функции game1, которая делает всё тоже самое, что и функция game, но запускается она у последнего подключившегося игрока (там есть небольшие различия).
* Функции main, которая узнаёт ник клиента и ip, подключается к серверу, принимает информацию о клиенте (те самые 200$ денег, 0 побед и 0 поражений), и запускает функцию gameStartScreen.

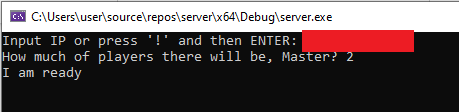
Также к приложению клиента подключена библиотека Header.h d, в которой описанны функции вывода на экран больших букв русского алфавита и знаков мастей карт (знак бубей, крестей, пик, червей, которые используются в функции gameStartScreen), а также функция gotoxy, которая позволяет перемещать каретку на заданные координаты на экране командной строки.

## **2. Пример работы приложения.**

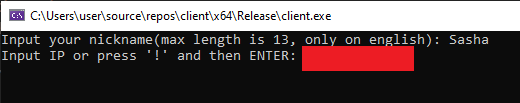
Вначале запускаем сервер



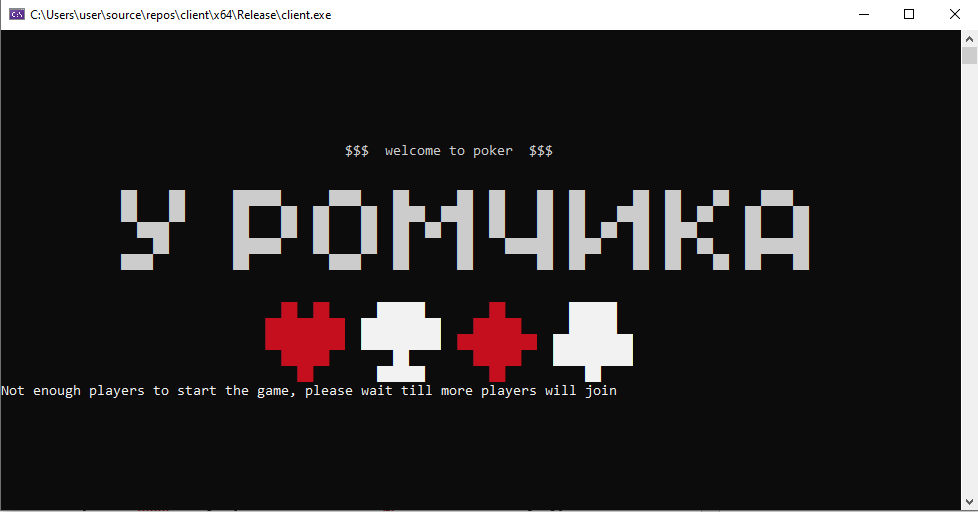
Он просит нас ввести ip и кол-во игроков что мы и делаем. Пусть у на будет 2 игрока:



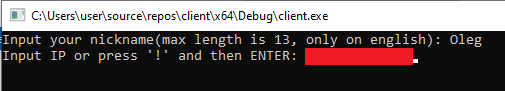
Теперь сервер ожидает подключения 2-х игроков. Начнём с 1-го. Его ник будет: Sasha. Ip будет тем же что и для сервера.



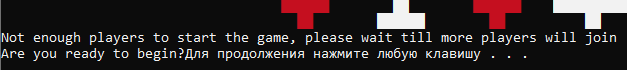
После ввода этих данных мы наблюдаем анимированную заставку, по окончании которой выводится сообщение о недостаточном числе игроков.



Теперь займёмся 2-м игроком. Его ник будет: Oleg. Ip будет тем же что и для сервера.



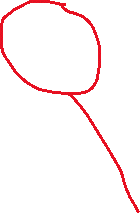
Дожидаемся окончания заставки у клиента Oleg и видим следующее сообщение у обоих пользователей.





Оба пользователя нажимают на любую клавишу и начинается игра. Вначале первый пользователь, то есть Sasha, делает маленькую ставку (little blaind).





Это отображается и у Oleg. Работа функции whatToDraw Работа функции whatCard



После Oleg делает большую ставку (big blaind). Теперь ход Sasha. Пусть он выберет check, то есть поддержит ставку (сейчас максимальная ставка 10$, а Sasha сделал маленькую на 5$, то есть Sasha поставит ещё 5$ и ставки всех игроков сравняются). Вводим 2 и нажимаем enter.





Причём решение Sasha видит и Oleg.

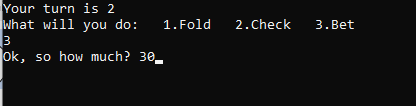


Oleg тоже сделает check и начнётся новый раунд. При начале 2-го раунда на стол выкладывается три карты. При начале 3-го и 4-го ещё по одной. В сумме 5.





Пусть Sasha сделает check, а Oleg сделает Bet (повышение ставки). И повысит он её на 20$, значит ввести он должен цифру 30.

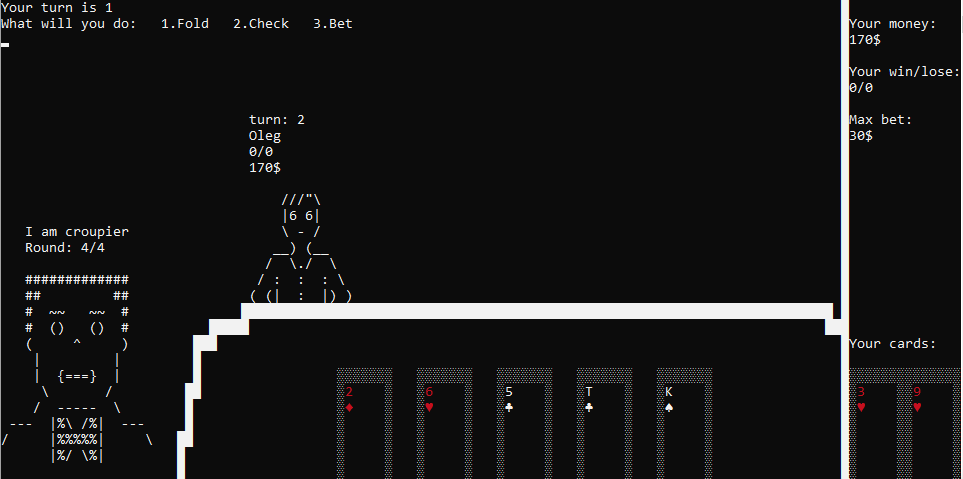




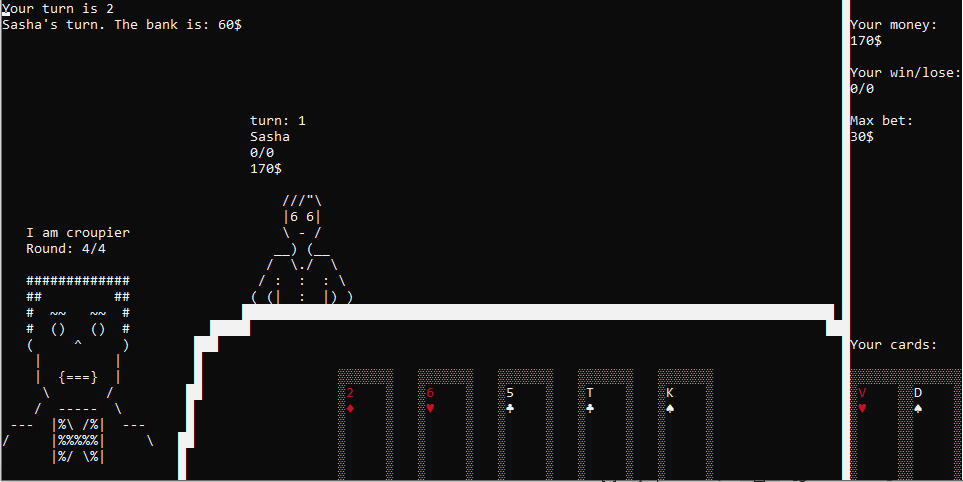


Теперь снова наступает ход Sasha (по правилам покера), а потом ход Oleg. Пусть они оба выберут check. В 3-м и 4-м раундах они тоже выберут check. В конце 4-го раунда расклад сил такой:

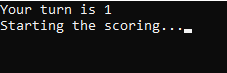
Sasha:



Oleg:

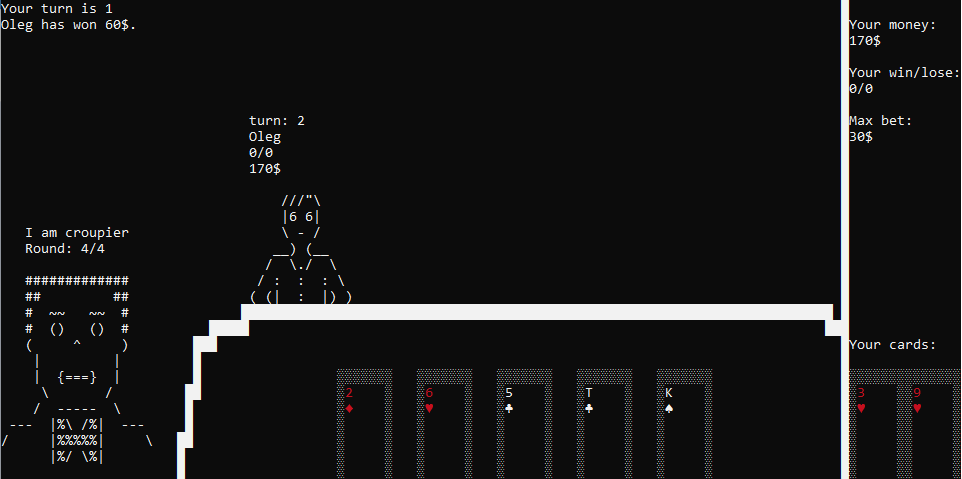


У Oleg есть комбинация из 5 карт, идущие по порядку V (валет), D (дама), K (король), T (туз), 2. Он должен выиграть. В конце 4-го раунда выводится сообщение о подсчёте очков.

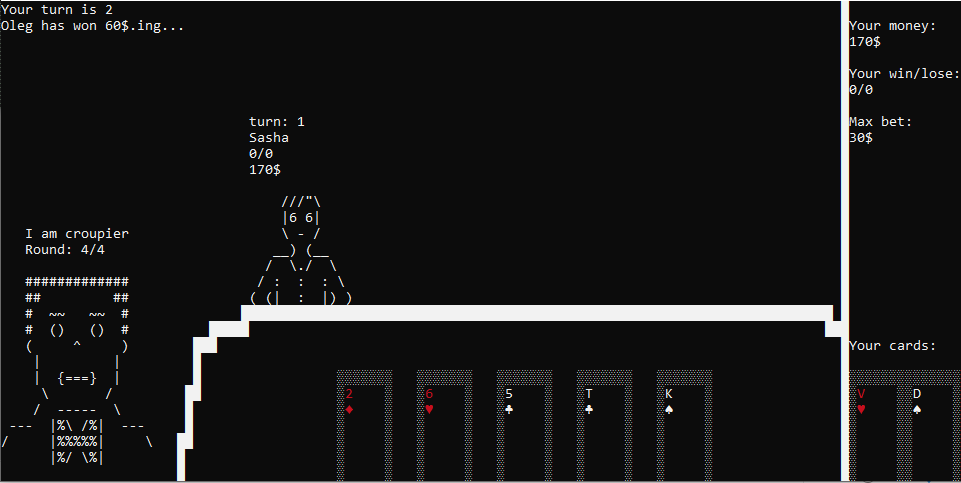


И выиграл Oleg.

Что видит Sasha:



Что видит Oleg:



После этого игроки могут нажать на любую кнопку и их программы закроются. А вот сервер надо закрыть вручную.

**ЗАКЛЮЧЕНИЕ**  
  
Таким образом, результат курсовой работы соответствует требованиям к реализации. Приложение работает на операционной системах Windows 10. Для создания приложения был использован язык C++ и компилятор Visual studio 2022. Многопоточность была реализована с помощью библиотеки thread.