МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

ОТЧЕТ ПО учебной

вычислительной (ознакомительной)

практике

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

студента 1 курса 7 группы,

специальности «Прикладная математика»

Руководитель практики:

Зубович Константин Антонович,

доцент кафедры ТП

Минск, 2025

ОГЛАВЛЕНИЕ

[Введение 4](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095773)

[ГЛАВА 1 ТЕОРИТИЧЕСКИЕ ОСНОВЫ ВЫБРАННОГО КУРСА 5](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095774)

[1.1 Общая характеристика курса 5](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095775)

[1.2 Модули курса 5](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095776)

[1.2.1 VCS Concept 5](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095777)

[1.2.2 Version Control Types 5](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095778)

[1.2.3 Why Git 6](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095779)

[1.2.4 Download, Install, Configure 6](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095780)

[1.2.5 Основные моменты по настройке Git 6](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095781)

[1.2.6 Тест: настройка Git 6](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095782)

[1.2.7 Create a GitHub Repo and Clone It 6](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095783)

[1.2.8 Pull from Remote 6](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095784)

[1.2.9 Тест: базовые операции 6](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095785)

[1.2.10 Git GUI & gitk 7](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095786)

[1.2.11 Inside .git Folder 7](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095787)

[1.2.12 Тест: внутреннее устройство Git 7](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095788)

[1.2.19 Conflict Solving 7](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095789)

[1.2.20 Rebase 7](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095790)

[1.2.21 Cherry-pick 7](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095791)

[1.2.22 Тест: ветки и merge 7](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095792)

[1.2.23 Tags 7](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095793)

[1.2.24 Stashing 8](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095794)

[1.2.25 Remotes 8](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095795)

[1.2.26 Branching Strategies 8](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095796)

[1.2.27 Extras 8](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095797)

[1.2.28 Тест: stash, tags, remotes 8](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095798)

[1.3 Выводы к главе 1 8](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095799)

[Глава 2 Практическая часть курса 9](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095800)

[2.1 Решение задач 9](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095801)

[Задание 2: Первая программа для 64-разрядной ЭВМ 10](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095802)

[Задание 3: Создание меню и реализация функционала 10](file:///C:\Users\user\Downloads\Отчет.docx#_Toc171095803)

Задание 4: Разработка приложения с клиент-серверной архитектурой 16

ссылка на проект в GitHub…………………………………………………..23

Использованные ресурсы………………………………………….24

ЗАВЕРШЕНИЕ………………………………………………………………24

Введение

Целью учебной вычислительной (ознакомительной) практики является закрепление полученных знаний за первый год обучения посредством выполнения специальных учебных заданий, участия в работе над командным проектом.   
 Впервые за наше обучение программированию нам предстоит работать не в одиночку, а командой. Нужно было развить сплоченность, командный дух и способность понимать друг друга.

Научились пользоваться основами командной строки.

Вспомнили язык программирования “ассемблер”.

Научились делать клиент-серверные системы.

И еще многое, что вам ещё предстоит увидеть в отчете…

С 30/06/2025 по 2.07.2025 знакомились с учебными курсами, доступные на сайте EPAM learn/com.  
Основной целью этой практики было применение и закрепление теоретических знаний, полученных в ходе обучения, а также приобретение навыков самостоятельной работы и взаимодействия в команде. Практика предоставила уникальную возможность познакомиться с профессиональной средой и методами работы одной из ведущих IT-компаний.

Цели практики:

* Ознакомиться с порталом компании, возможностями обучения и повышения квалификации.
* Изучить инструменты для совместной разработки проектов и контроля версий.
* Ознакомление с различными направлениями профессиональной подготовки в IT-сфере.

Практическая часть:

* Выполнить стандартные задания тренинга.
* Выполнить задания тренинга повышенного уровня.
* Ознакомление с порталом компании
* Изучение инструментов для разработки и контроля версий
* Закрепление навыков работы с Git

Курс "Version Control with Git", пройденный в рамках практики, предоставляет понимание системы контроля версий Git, которая является одной из наиболее популярных и широко используемых в индустрии IT. В процессе обучения участники изучают основы работы с Git, включая создание репозиториев, управление ветками, внесение изменений и работу с удаленным репозиториями. Git используется для отслеживания изменений в коде, что позволяет нескольким разработчикам эффективно сотрудничать над проектами, отслеживать версии и при необходимости откатываться к предыдущим версиям. Это ключевой инструмент для управления проектами любого масштаба, от небольших команд до крупных корпораций. В рамках курса "Version Control with Git" были изучены следующие модули: Введение в системы контроля версий, основы работы с Git, основные команды Git, управление ветками, работа с удаленными репозиториями, разрешение конфликтов и т.д.

ГЛАВА 1 ТЕОРИТИЧЕСКИЕ ОСНОВЫ ВЫБРАННОГО КУРСА

1.1 Общая характеристика курса

Программы Epam помогают мотивированным новичкам получить все необходимые навыки для успешного старта в IT. Выпускники их программ приобретают как теоретические знания, так и практический опыт, что позволяет им чувствовать себя уверенно на проектах.

Их программа по .NET является отличной входной точкой в мир .NET-разработки. В процессе обучения участники получают глубокое понимание языка C#, баз данных и .NET Framework.

Обучение можно пройти в комфортном темпе по собственному расписанию, возвращаться к изученным материалам по мере необходимости. Сочетание теории различных форматов, практических заданий и тестов для самопроверки поможет лучше усвоить и закрепить пройденный материал.

Система оценки заданий:

* этап компиляции (Compile) в случае успеха дает 5 очков;
* этап запуска юнит-тестов (Test) в случае 100% успеха дает 65 очков;
* этап проверки Sonar (Quality) в случае отсутствия существенных замечаний дает 30 очков (если присутствуют замечания, то из этих очков вычитаются штрафные очки в зависимости от типа замечания).

Для ознакомления с IT, мной были просмотрены все образовательные лекции на <https://training.by/>. Каждый видеоурок был познавательный и открыл мне много нового. Далее можно прочесть краткое описание каждой лекции.

1. **Краткий обзор IT-профессия**В этом видео рассказывается о различных ролях в IT-индустрии и какие проекты можно выполнять. Спикер делится своим опытом работы и рассказывает о своей позиции в компании.
2. **SAP Development & Consulting**

Стрим предназначен для знакомства с компанией Цепи и ее деятельностью в области SAP. Предоставляются тренинги по Java-разработке и консалтингу.

1. **Data Analytics Engineering**

В этом видео сотрудники EPAM рассказывают о карьере в области анализа данных. Они объясняют, что такое аналитика данных, как она используется бизнесом и какие ценности она приносит. Гости обсуждают роль аналитика данных в превращении технических данных в ценную информацию для бизнеса и приводят примеры того, как аналитика данных влияет на принятие бизнес-решений.

1. **Functional Testing**

Беседа о профессии QA-инженера и основных аспектах тестирования программного обеспечения. Тестирование программного обеспечения – это вселенная с собственными правилами, типами и методами.

QA-инженеры выполняют подготовительные работы, создают документацию, проводят тестирование различных типов и составляют отчеты.

1. **Robotic Process Automation**

На этом онлайн-конференции "Robotic Process Automation Junior" были представлены темы, связанные с робототехникой и автоматизацией процессов. Рассматривалась технология Orpheus Robotics и ее применение в автоматизированных процессах. Была продемонстрирована работа небольшого робота и его возможностей.

1. **Performance Optimization**

На прямом эфире "Junior's Performance Optimization Conference" рассматривается роль и обязанности аналитика производительности, а также предоставляются секреты и советы для начинающих в этой области.

1. **Cloud & DevOps**

На этом выпуске Junior's DevOps Online Conference рассматривается тема DevOps и его роль на проектах. Ведущий беседует с профессиональными DevOps-инженерами, которые объясняют суть и обязанности этой роли. Также обсуждаются различные аббревиатуры и технические аспекты DevOps.

Больше всего меня вдохновило видео про DevOps, так как спикеры (Виктор и Максим) довольно захватывающе рассказывали про свою профессию. Особенно интересными были а также проведение практических занятий по разворачиванию контейнеров. Эта профессия прекрасна тем, что благодаря взаимодействию разработчиков, тестировщиков и других IT-специалистов команда работает намного эффективней. Здесь нужны обширные знания и умение коммуницировать.

1.2 Модули курса

Здесь перечислены модули, которые были мной изучены на <https://learn.epam.com/>.

Getting Started with Autocode

После изучения этого раздела я научилась работать с Autocode и разобралась с кодстайлом языка C#.

Integers

После изучения этого раздела я научилась использовать литералы и константы целочисленных числовых типов в выражениях языка C#.

Static Methods

После изучения этого раздела я могу использовать статические методы для организации алгоритма в структурную единицу.

Bools & Logic Operators

После изучения этого раздела я в силах использовать логические операторы для создания логических выражений на языке C#.

Сonditional Statements (if)

После завершения этого урока я могу использовать условный оператор if для построения простых алгоритмов.

Real numbers

После завершения этого урока я научилась использовать литералы и константы типов значений с плавающей запятой в выражениях языка C#.

Conditional Statements (If-Else)

После завершения этого урока я поняла, как создавать простые алгоритмы, используя условный оператор if-else и тернарный оператор ?:.

Exceptions

После завершения этого урока я обучилась, как обрабатывать непредвиденные или исключительные ситуации в своем коде или вызывать исключение, чтобы сигнализировать об исключительной ситуации.

Iteration Statements (While)

После завершения этого урока я могу разрабатывать простые алгоритмы с циклами, используя оператор while.

Iteration Statements (For, Do)

После завершения этого урока я умею разрабатывать простые алгоритмы с циклами, используя операторы for и do.

Formatting And Parsing Strings

После завершения этого урока я умею разрабатывать алгоритмы средней сложности для работы со строками.

Working With Arrays

После завершения этого урока я могу разрабатывать алгоритмы средней сложности для работы с массивами.

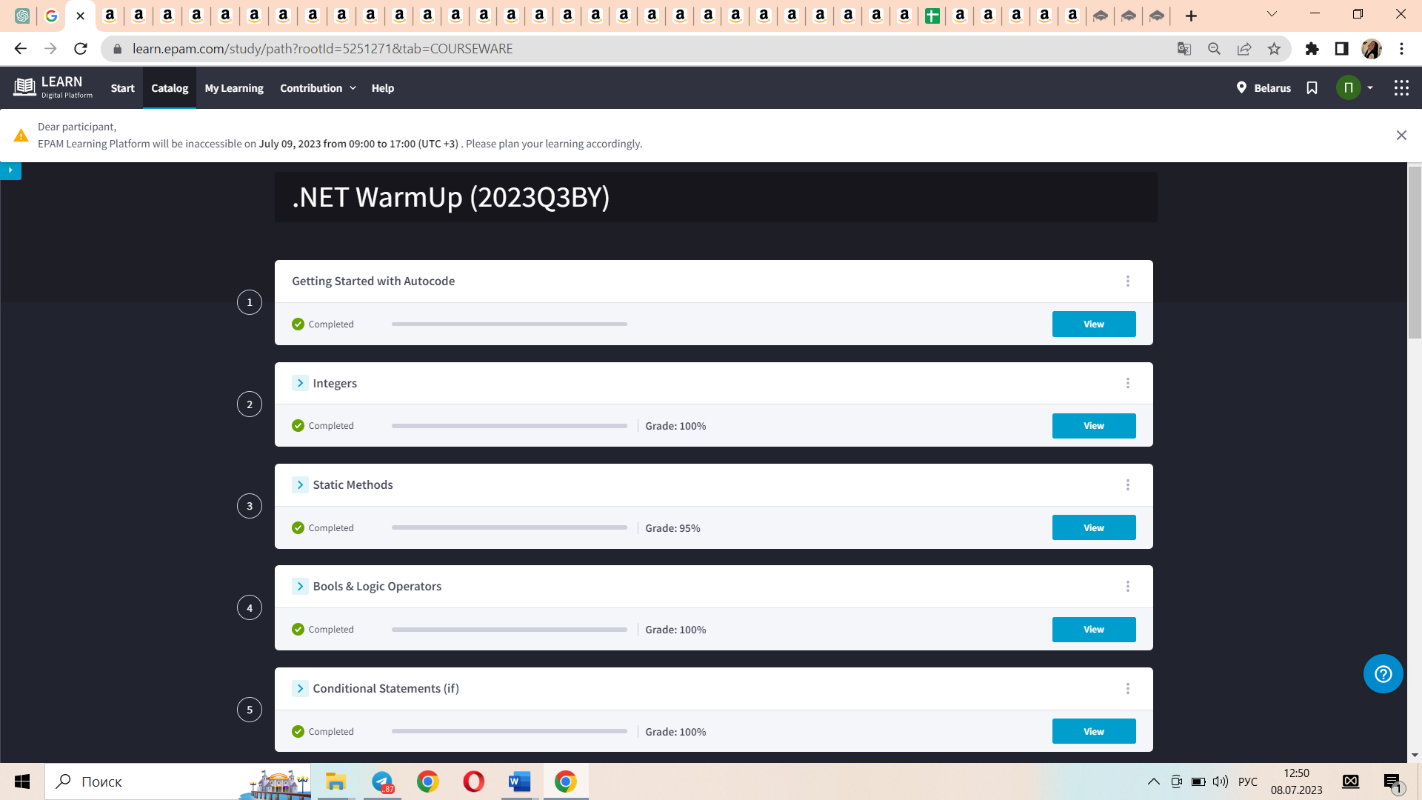
Implementing Search Algorithms

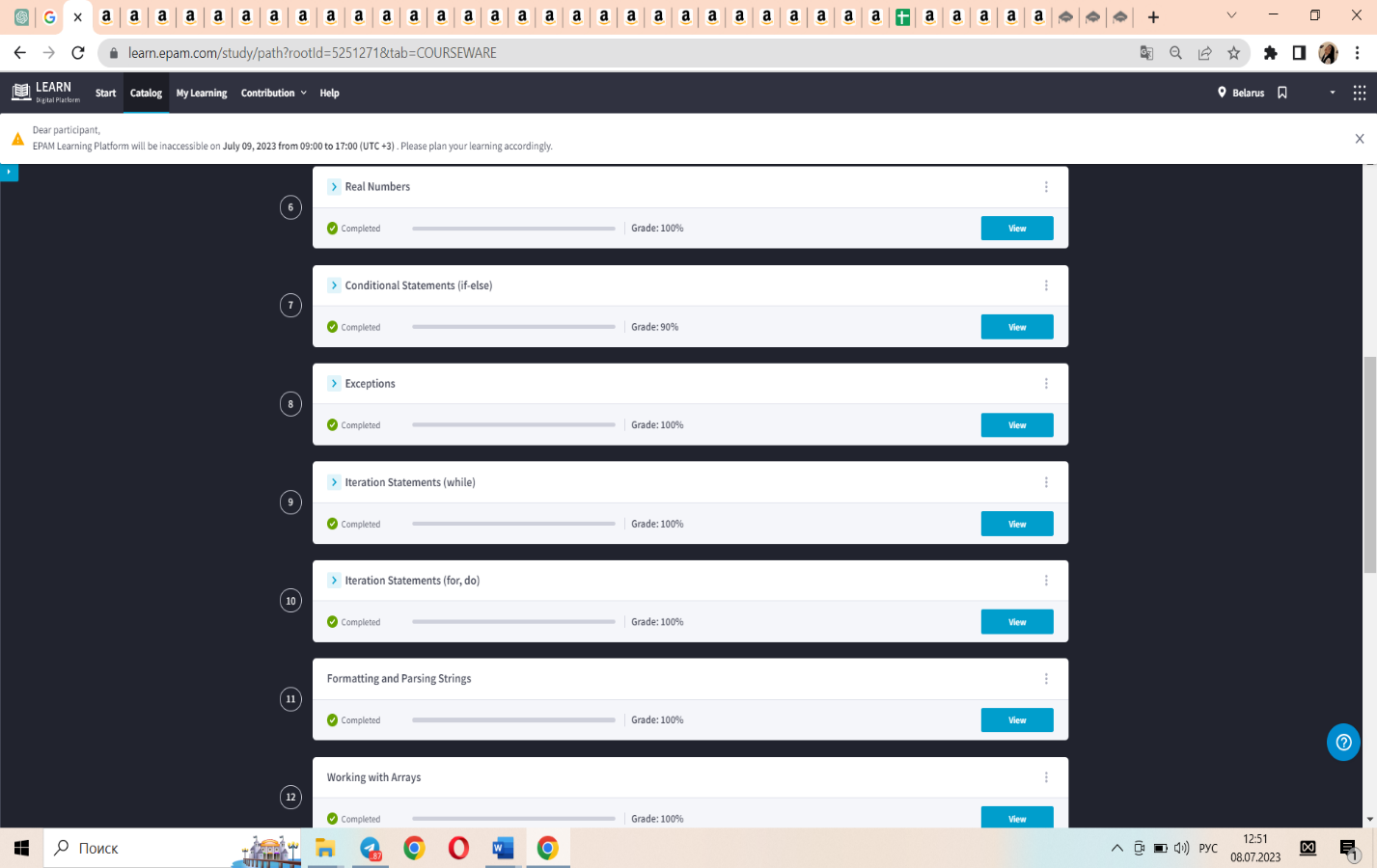
Implementing Sorting Algorithms

Searching in Strings

Solving Problems with Recursion

Bit Operations

Streams in .NET



Глава 2 Практическая часть курса

2.1 Решение задач

**Задание 1: Создание аккаунта для совместной работы**

Для выполнения первого задания каждой группе необходимо было создать общий аккаунт для совместной работы всех членов группы. Это обеспечивало централизованное управление проектом и доступ к репозиторию для всех участников. В каждой группе была выбрана одна задача, которая решалась кем-либо из этой группы в рамках лабораторной работы №4. Это позволило нам работать с уже известной задачей и сосредоточиться на изучении инструментов контроля версий. После выбора задачи и её решения, мы должны были разместить полученное решение в созданном репозитории на GitHub. Это обеспечивало доступность кода для всех членов группы и возможность отслеживания изменений и версий.

Цели выполнения практических заданий:

**Освоение совместной работы с Git**: Научатся работать в команде, используя систему контроля версий Git.

**Реальное применение теоретических знаний**: Закрепление знаний, полученных в теоретической части курса, на практике.

**Развитие навыков управления проектами**: Научатся создавать и управлять репозиториями, вносить изменения и отслеживать их историю.

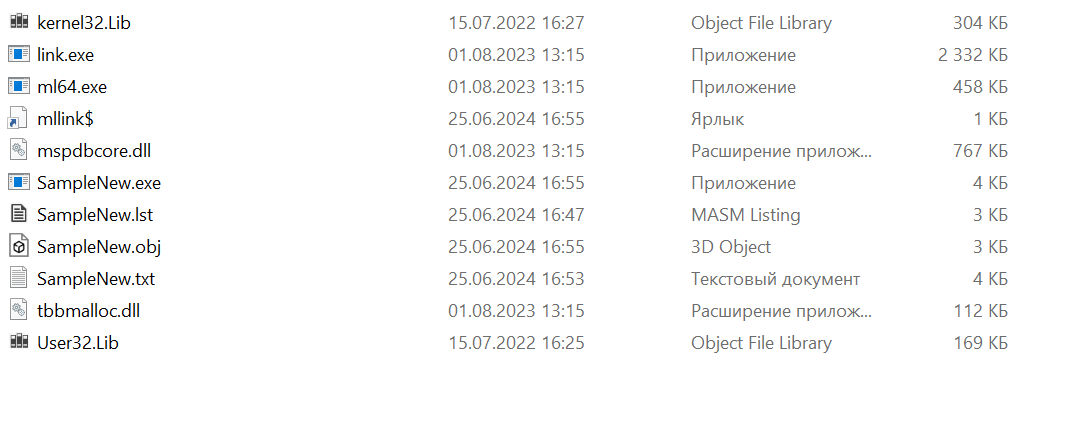
**Практика решения задач**: Углубление понимания решения реальных задач в команде с использованием Git.

### **Задание 2: Первая программа для 64-разрядной ЭВМ**

**Цель задания**: Создание программы на языке ассемблера, которая выводит стандартное окно MessageBox с сообщением "Hello world", а затем изменяет сообщение на "Hello hell!" после нажатия кнопки.

**Пошаговая инструкция**:

1. **Прочитать Лекцию 5**: Основное внимание уделяется пониманию процесса перевода исходного текста на ассемблере в машинный код и получения исполнимого файла.
2. **Набор исходного текста**: Написание программы на ассемблере в текстовом редакторе (например, Блокнот) и сохранение файла в рабочей папке.
3. **Трансляция**: Использование транслятора ml64.exe для преобразования исходного текста в объектный файл (.obj).
4. **Линковка**: Использование линковщика (например, LINK.exe) для создания исполнимого файла (.exe) из объектного файла.
5. **Запуск программы**: Загрузка и выполнение исполнимого файла в операционной системе Windows 10.



### **Задание 3: Создание меню и реализация функционала. Совместная проектная работа “Часы”**

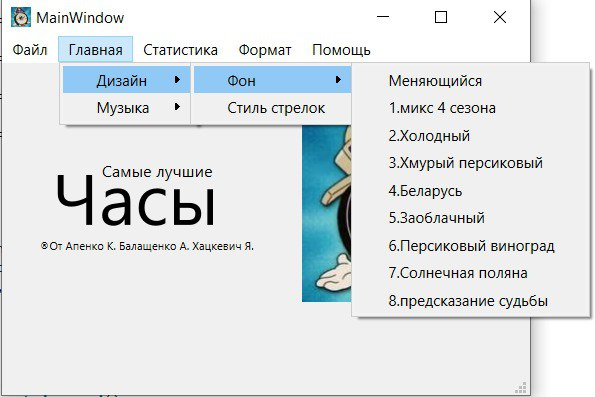
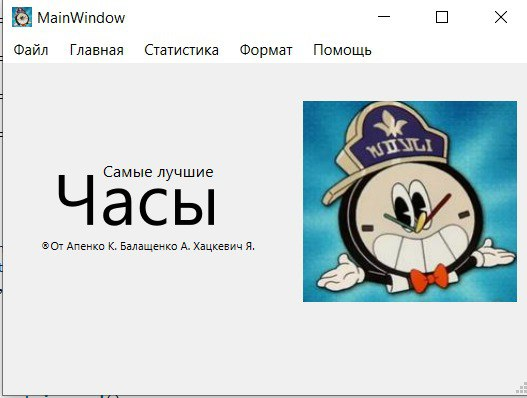
**Цель задания**: Реализация функционала меню в программе, а также добавление функций для ввода и проверки даты и времени.

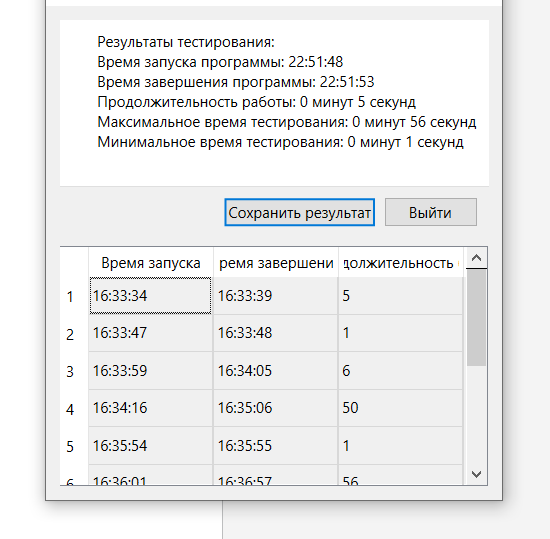
#### Шаги выполнения задания:

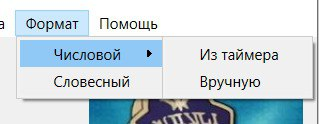
1. **Создание меню**:
   * Меню состоит из опций: Файл, Главная, Формат задания исходных данных, Статистика, Помощь.
   * Формирование и обработка меню выполняются всеми членами группы.
   * Меню было спроектировано на листочках, сфотографировано и отправлено преподавателю на почту.
2. **Распределение задач**:
   * Задачи 2, 3, 4 решаются отдельно членами группы.
   * Все версии и изменения хранятся в репозитории на GitHub.
3. **Формат задания даты и времени**:
   * Реализация ввода даты и времени в формате ДД.ММ.ГГГГ ЧЧ:ММ:СС.
   * Одновременный вывод словесного указания даты и времени (например, 26.06.2024 12:16:45 -> Двадцать шестого июня две тысячи двадцать четвертого года Двенадцать часов шестнадцать минут сорок пять секунд).
   * Возможность автоматического ввода даты и времени из таймера.
   * Проверка правильности ввода даты и времени с записью неверных данных в файл статистики.
4. **Обратный ввод даты и времени**:
   * Ввод даты и времени в словесном формате.
   * Одновременный вывод в стандартном формате (например, Двадцать шестого июня две тысячи двадцать четвертого года Двенадцать часов шестнадцать минут сорок пять секунд -> 26.06.2024 12:16:45).
   * Проверка правильности ввода.
5. **Функция завершения тестирования**:
   * Кнопка "ЗАКОНЧИТЬ ТЕСТИРОВАНИЕ", при нажатии на которую тестирование заканчивается и формируется статистика.
   * Запись в файл информации о начале тестирования, времени окончания, наименовании задачи.
6. **Статистика**:
   * Подменю в разделе "Статистика" должно включать данные о самом раннем и позднем начале тестирования, самом длительном тестировании и другие параметры.
   * Таблица с параметрами каждого тестирования и опциями (например, неверные даты и время).

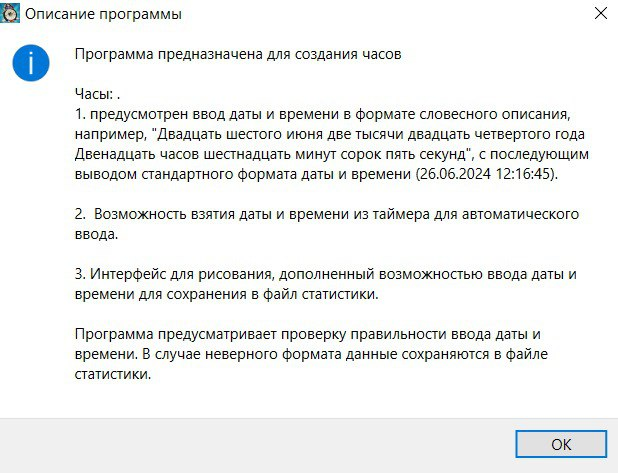
### Реализованные файлы и их описание:

1. **formatslovestny.h**:
   * Заголовочный файл для модуля, реализующего формат словесного указания даты и времени.
2. **formatslovestny.ui**:
   * Файл пользовательского интерфейса для модуля формата словесного указания даты и времени.
3. **LABA\_C\_CHASAMI\_POLNAYA.pro**:
   * Файл проекта для сборки и компиляции приложения.
4. **main.cpp**:
   * Основной файл программы, содержащий точку входа и основной цикл приложения.
5. **mainwindow.cpp**:
   * Реализация основного окна приложения, обработка событий и взаимодействие с пользователем.
6. **mainwindow.h**:
   * Заголовочный файл для основного окна приложения.
7. **mainwindow.ui**:
   * Файл пользовательского интерфейса для основного окна приложения.
8. **Media.qrc**:
   * Ресурсный файл, содержащий медиа-ресурсы приложения.
9. **clocky.cpp, clocky.h, clocky.ui**:
   * Файлы для реализации функционала часов, используемых для автоматического ввода даты и времени.
10. **formatchislovoy.cpp, formatchislovoy.h, formatchislovoy.ui**:
    * Файлы для реализации ввода и вывода даты и времени в числовом формате.
11. **formatslovestny.cpp**:
    * Реализация функций для работы с форматом словесного указания даты и времени.



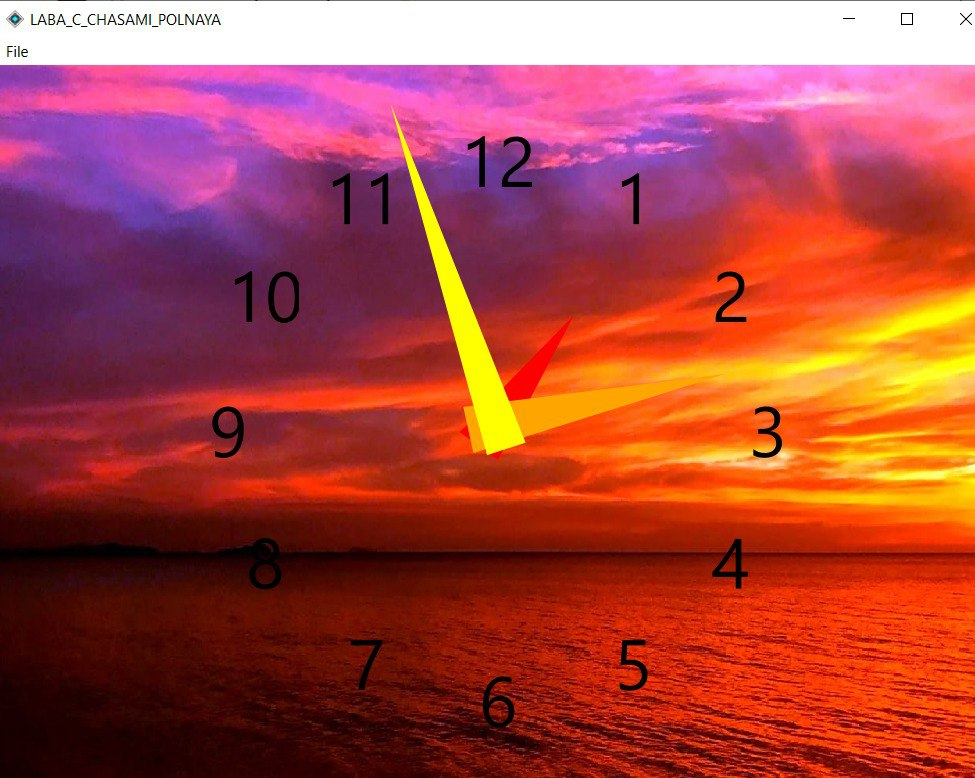


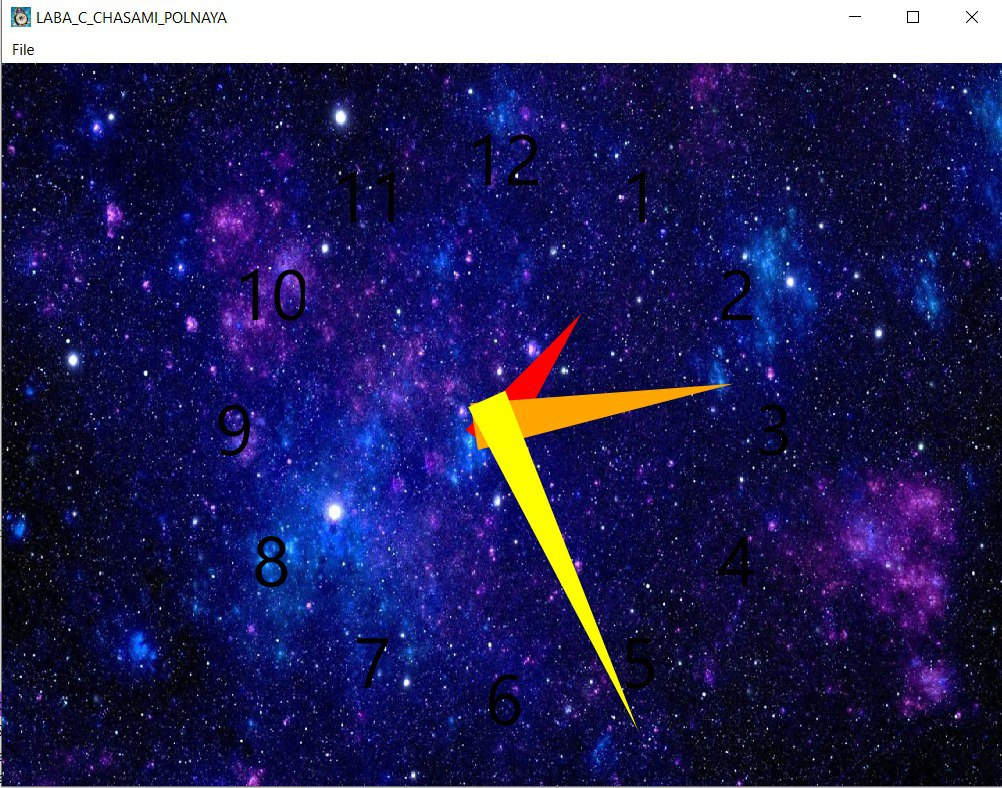


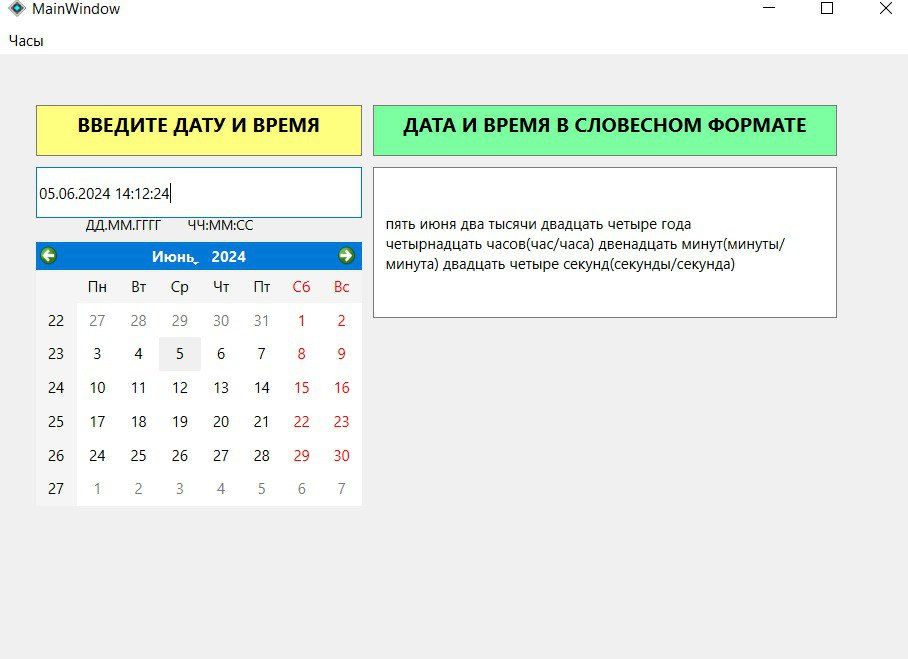


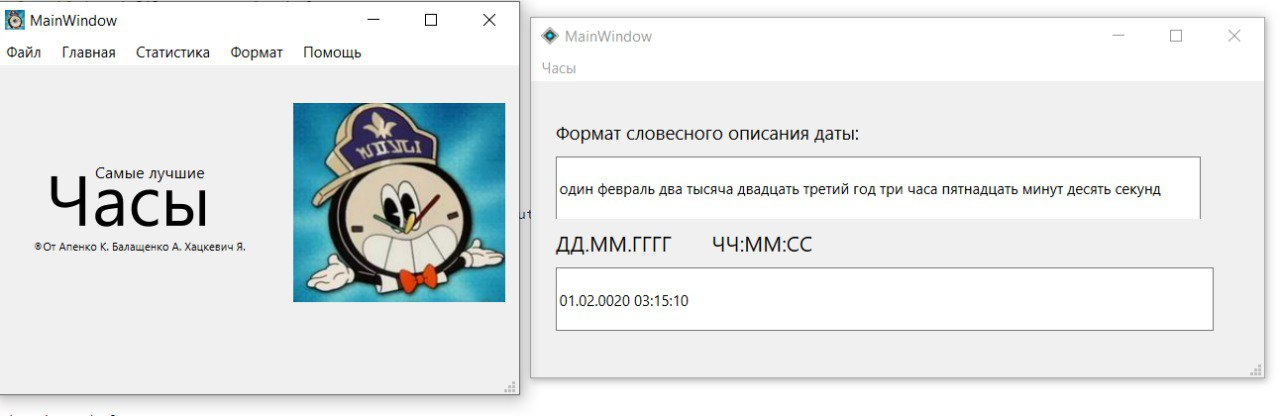
Итоговые часы:











### **Задание 4: Разработка приложения с клиент-серверной архитектурой**

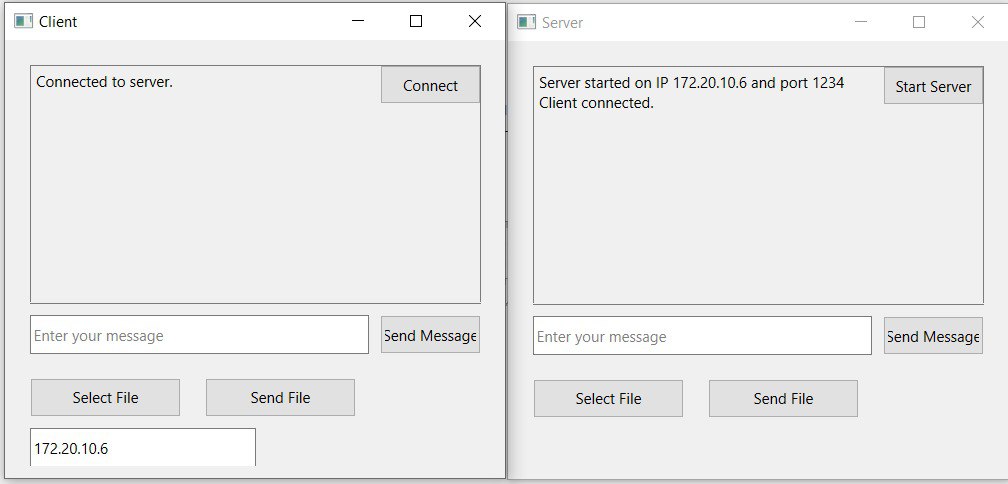
Используя всё что угодно, добиться того, чтобы клиенты (два клиента— задний и передний концы — два человека) могли со своего компьютера  отправить сообщение на сервер «Привет, Garson от Заднего конца (Переднего конца)!», сервер (Полный стакан) получает это сообщение, меняет его на «Я НЕ GARSON! Я просто сервер» и отправляет его тому клиенту, от которого получено сообщение.Это всё делается с использованием С++, учя что такое сетевая карта, что такое socket, что такое host, что такое IP-адрес и какие средства есть в языке записи алгоритмов С++ для их изображения. 3. Клиенты сообщения вводят в окне редактирования, Сервер отображает его в messagebox, переносит его в окно редактирования, меняет его и отправляет сообщение клиенту.

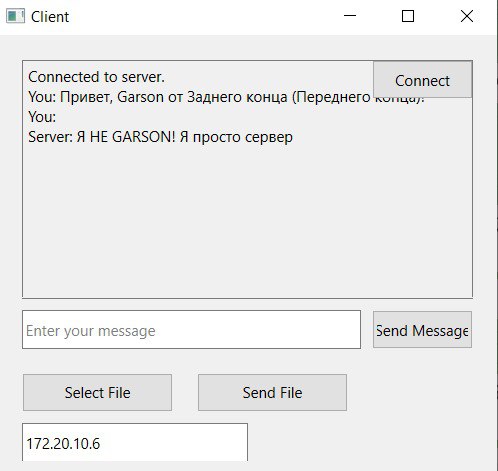
Для выполнения данного задания была разработана клиент-серверная архитектура на языке программирования C++. Основные компоненты решения включают:

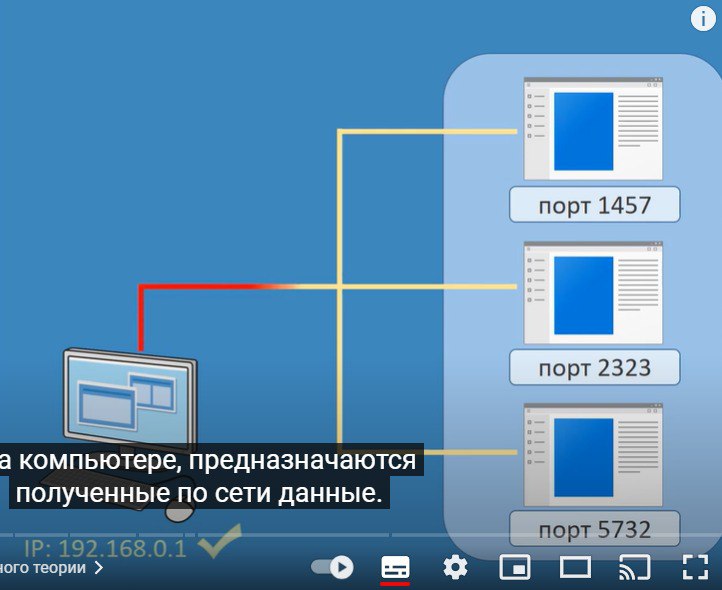
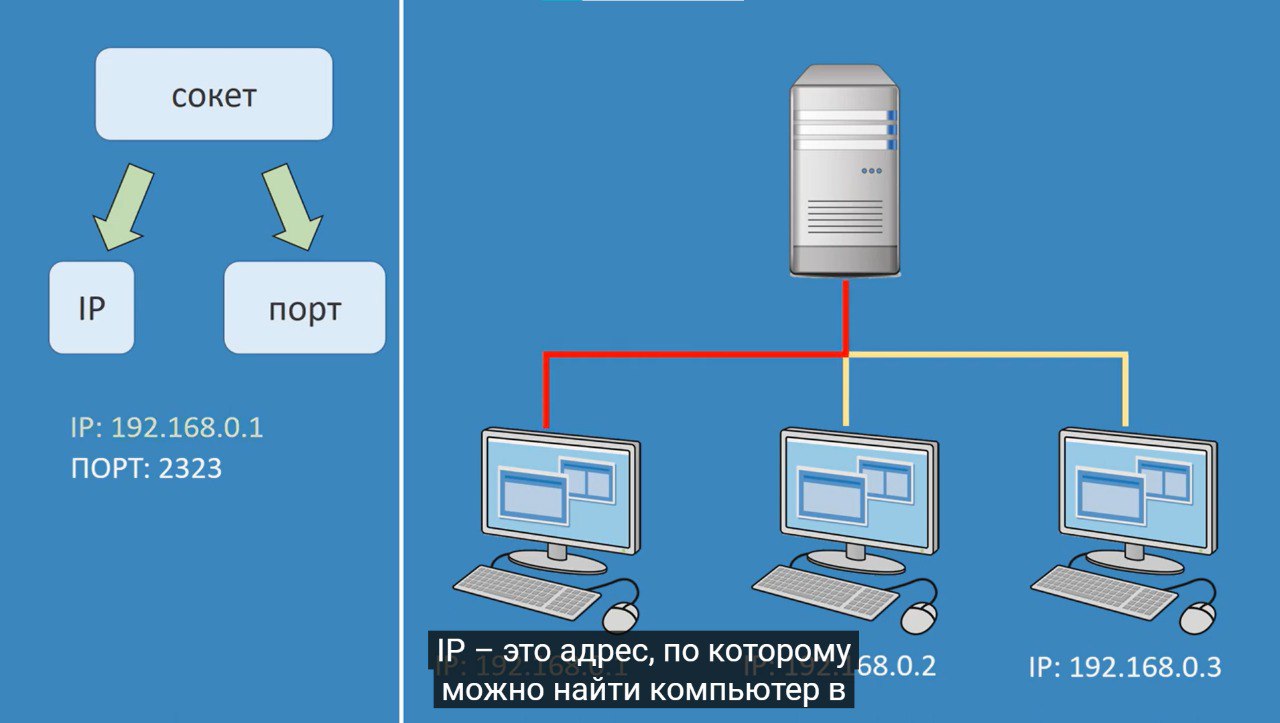
1. Сетевая карта — аппаратное обеспечение, позволяющее компьютеру взаимодействовать с сетью.
2. Socket — программный интерфейс для обмена данными между узлами сети.
3. Host — узел в сети, идентифицируемый уникальным IP-адресом.
4. IP-адрес — уникальный идентификатор узла в сети.

Шаги выполнения:

1. Создание сервера:
   * Сервер создает сокет и привязывает его к IP-адресу и порту.
   * Слушает входящие соединения.
   * Принимает сообщение от клиента.
   * Меняет сообщение на «Я НЕ GARSON! Я просто сервер».
   * Отправляет измененное сообщение обратно клиенту.
2. Создание клиента:
   * Клиент создает сокет и подключается к серверу.
   * Отправляет сообщение «Привет, Garson от Заднего конца (Переднего конца)!» на сервер.
   * Получает ответ от сервера и отображает его в окне редактирования.





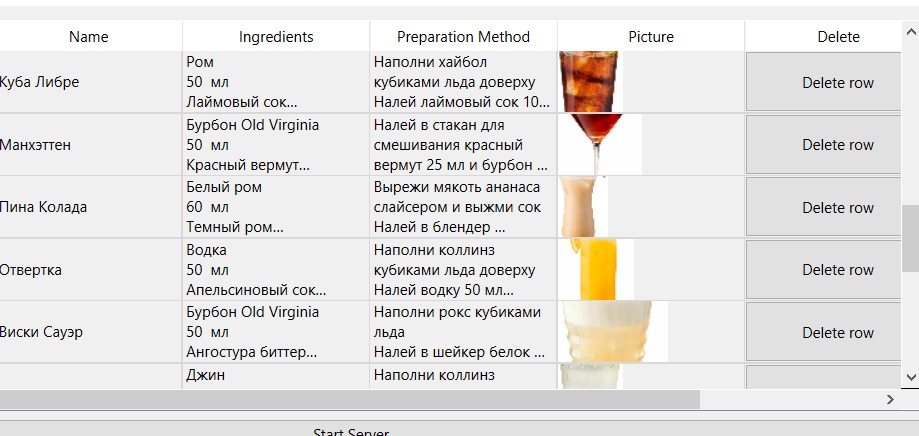
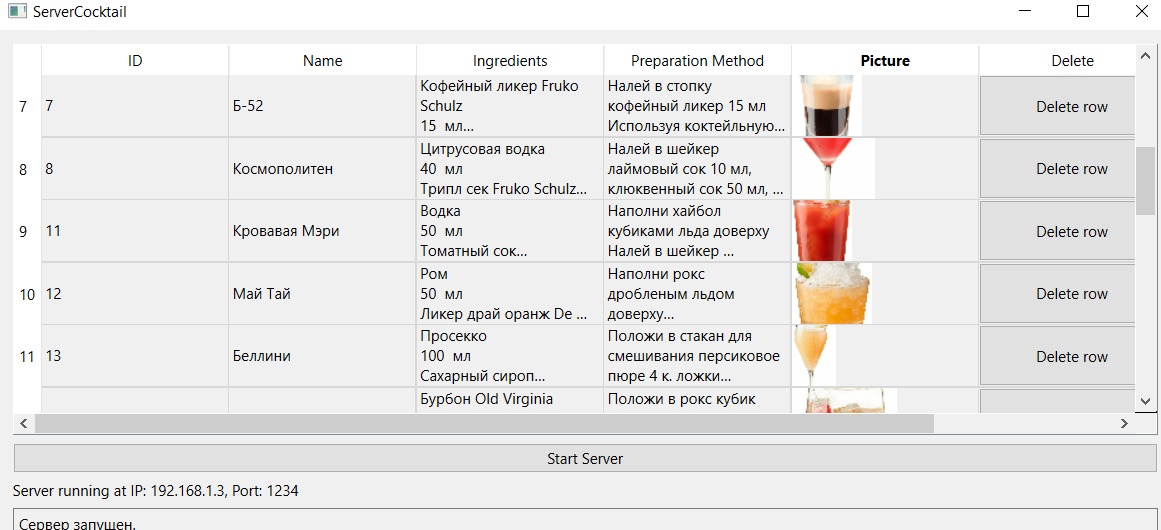
****

### **Задание 4.1: Разработка приложения с клиент-серверной архитектурой, базой данных коктейлей и напитков. Получение коктейля по названию или по ингридиентам**

(Фул стек занимается организацией и хранением информации о том, какое блюда(напиток) может быть приготовлен из тех или иных ингредиентов с указанием  
веса или обьема . Фул стек наполняет базу данных. Заполняет какой продукт, из каких компонент состоит и описанием процесса приготовления  
Тот, кто будет заниматься клиентами 2 вопроса  
передний конец организует общение с пользователем, задний конец- общение с сервером(т.е правильный запрос на сервер, чтобы сервер выдал по определенной клиентной информации)  
Именно клиент может спросить рецепт по названию или несколько рецептов что можно приготовить из того или иного ингредиента)

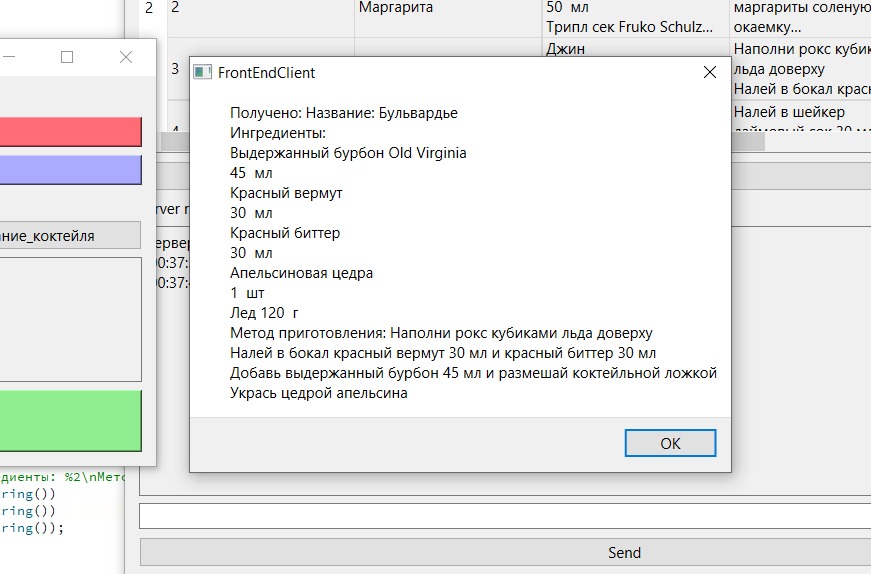
**Прилагаются фото программы :**

**Сервер**

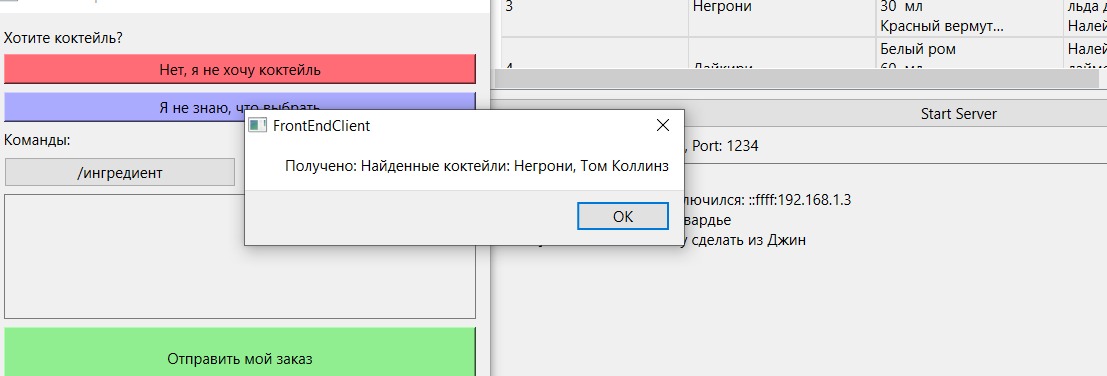
****

****

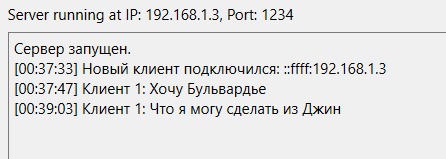
**База данных, отображенная в QTableWidget, который пригодиться при дальнейшем поске и сортировке .  
  
(база данных сохраняется и загружается из** cocktails.db), где db – data base

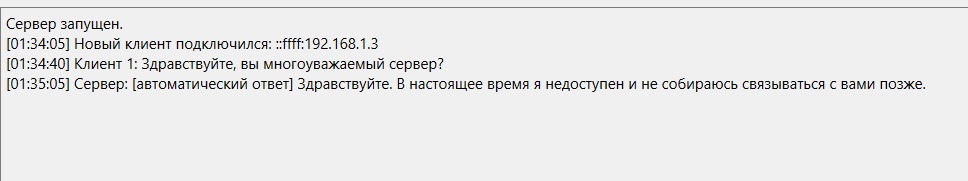
 **поиск по названию**

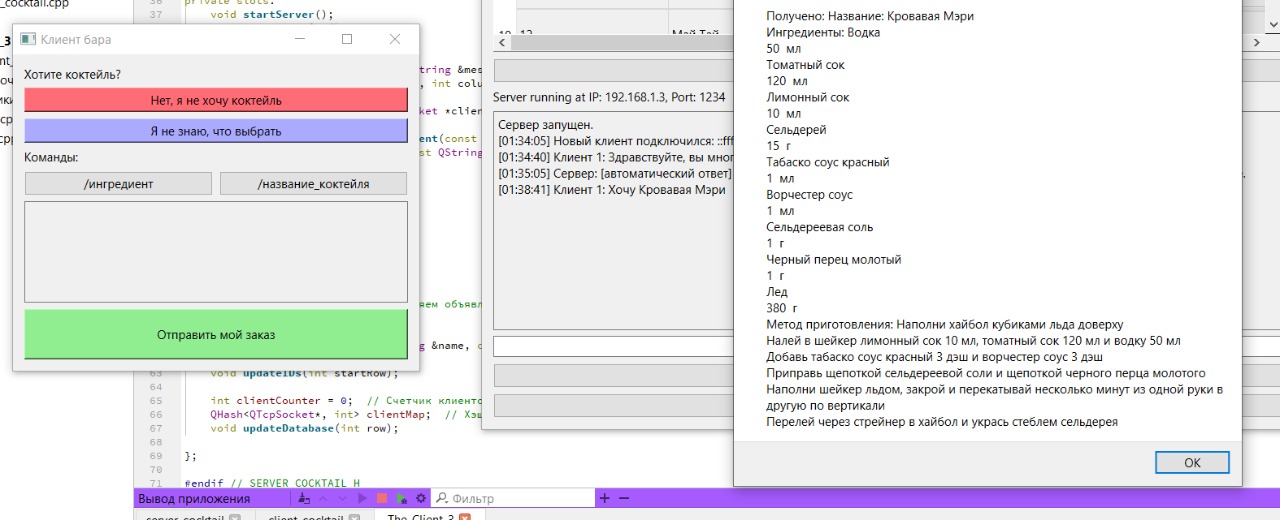
**фото коктейля, присланное клиенту.**

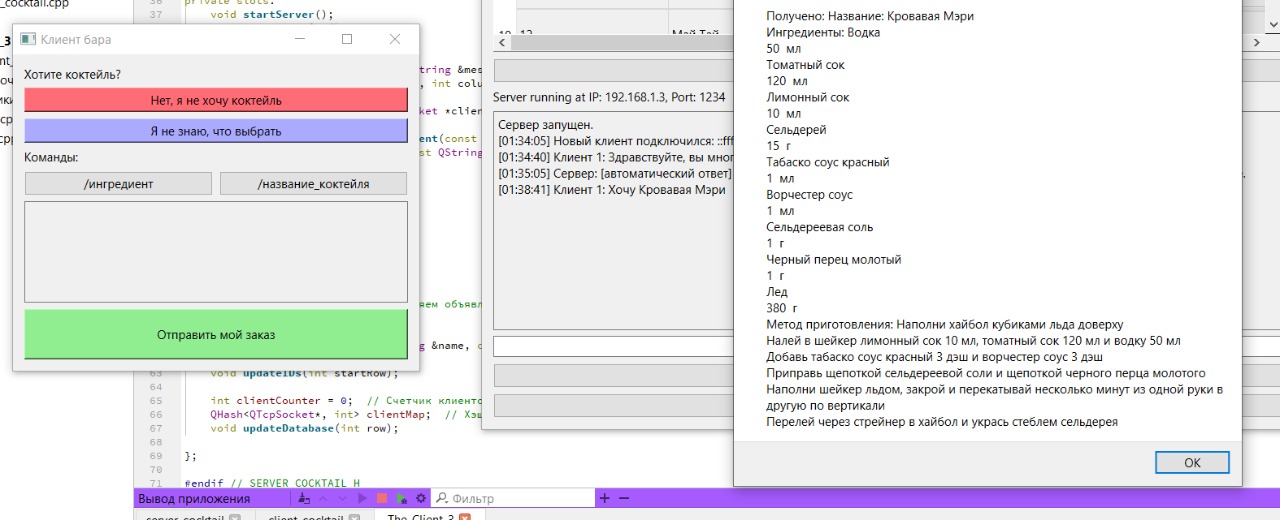
****

**Поиск по ингредиентам.**

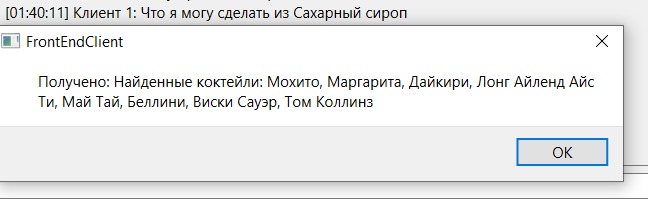
**  
запросы на сервер**

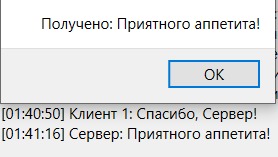
****

****

****

**Окно клиента, где при выборе кнопок клиент получает шикарную помощь в создании запроса для сервера.**

****

****

**Ссылки на этот проект в GitHub:**

[**https://github.com/Fresh-timeey/Comedy-Club/tree/%D0%90%D0%BF%D0%B5%D0%BD%D0%BA%D0%BE-%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D0%B0%D0%BD%D1%82%D0%B8%D0%BD1**](https://github.com/Fresh-timeey/Comedy-Club/tree/%D0%90%D0%BF%D0%B5%D0%BD%D0%BA%D0%BE-%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D0%B0%D0%BD%D1%82%D0%B8%D0%BD1)

**Пользуйтесь!**

**Использованные ресурсы:**

* **Сайты с полезными материалами для программирования.**
* **Материалы с учебного портала edu.fpmi .**
* **Видео про клиент-серверные системы и онлайн в Qt на ютубе.**
* **Материалы, присланные руководителем практики по почте.**
* **…**

**Завершение**

Летняя учебная практика закрепила все знания по программированию, полученные за 1 курс обучения, и дала много новых!  
Научила руководить маленькой командой разработчиков с желанием создавать большие проекты.