

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

вычислительной техники

Кафедра

МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

БАКАЛАВРСКАЯ РАБОТА

на тему

Разработка программного имитатора системы централизованного оповещения в информационной системе метрополитена

Студент

(подпись, дата)

Афтаев Игорь Дмитриевич

(ФИО полностью)

Руководитель

(подпись, дата)

Попова Н.А.

(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись)

Козлов А.Ю.

(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.

(фамилия, инициалы)

Пенза, 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов «25»
декабря 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА**

1. Студент Афтаев Игорь Дмитриевич гр. 20ВИ1 факультета ФВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Разработка программного имитатора системы централизованного оповещения
в информационной системе метрополитена

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

2. Руководитель работы _____

3. Задание на работу (назначение разработки, исходные данные и т.п.) Разработка
программного имитатора системы централизованного оповещения системы централизованного
оповещения в информационной системе
метрополитена

4. Перечень подлежащих разработке вопросов

1. Анализ предметной области и постановка задачи; _____

2. Проектирование программных средств; _____

3. Реализация программных средств; _____

4. Тестирование программных средств; _____

5. Оформление ВКР. _____

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	12.04.2024– 16.04.2024	
2	Проектирование программных средств	20%	17.04.2024 – 26.04.2024	
3	Реализация программных средств	30%	27.04.2024 – 11.05.2024	
4	Тестирование программных средств	20%	12.05.2024 – 21.05.2024	
5	Оформление пояснительной записки	20%	22.05.2024 – 01.06.2024	

Дата выдачи задания «22» декабря 2023г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 135 листов, 16 рисунков, 23 таблицы, 2 листинга, 16 использованных источников и 2 приложений.

Visual Studio 2022, АВТОМАТИЗАЦИЯ, ИНФОРМАЦИОННАЯ СИСТЕМА, СИСТЕМА ЦЕНТРАЛИЗОВАННОГО ОПОВЕЩЕНИЯ.

Предмет работы: разработка программного имитатора системы централизованного оповещения.

Объект работы: автоматизация деятельности метрополитена.

Цель работы: разработка программного имитатора системы централизованного оповещения в информационной системе метрополитена.

Технология разработки: Visual Studio 2022.

Результаты работы: спроектирован и разработан программный имитатор системы централизованного оповещения в информационной системе метрополитена.

Область применения: метрополитен.

					ПГУ 09.03.02 – 08ВР201.02 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Афтаев И. Д.			Программный имитатор системы централизованного оповещения в информационной системе метрополитена Пояснительная записка	Лит.	Лист	Листов
Провер.		Попова Н.А.					4	135
Реценз.						группа 20ВИ1		
Н. Контр.		Такташкин Д.В.						
Утв.		Козлов А.Ю.						

Содержание

Введение	6
1 Постановка задачи и анализ разрабатываемой системы.....	9
1.1 Постановка задачи	9
1.2 Выбор технологий проектирования и разработки	12
2 Планирование и расчет бюджета разрабатываемой системы.....	14
3 Проектирование информационной системы	20
3.1 Анализ бизнес-процессов предприятия	20
3.2 Проектирование программного обеспечения	20
3.2.1 Диаграмма состояний	20
3.2.2 Диаграмма деятельности	22
3.2.3 Диаграмма вариантов использования	24
3.2.4 Диаграмма классов	25
4 Реализация информационной системы	36
5 Тестирование информационной системы	40
5.1 Функциональное тестирование	40
5.2 Модульное тестирование.....	42
5.3 Анализ кода	45
Заключение	51
Список использованных источников	52
Приложение А (справочное) Глоссарий	54
Приложение В (рекомендуемое) Листинг программы.....	56

Введение

Сегодня информационные технологии шире распространяются на абсолютно большинство сфер человеческой деятельности. С каждым годом по миру происходит повсеместное внедрение компьютеров абсолютно в каждой отрасли экономики, что кардинально изменяет жизнь людей [1].

Система оповещения является одной из самых распространенных видов систем звукообеспечения. Основной причиной является безопасность жизни людей. По требованиям вышестоящих органов громкое оповещение должно находиться, где возможно хоть какое-то скопление людей [2].

СЦО – современный программно-аппаратный комплекс, построенный на основе промышленных компьютеров [3].

Система оповещения представляет собой организационно-техническое объединение сил и средств связи и оповещения, сетей вещания, каналов сети связи общего пользования, обеспечивающих доведение информации и сигналов оповещения до органов управления, сил единой государственной системы предупреждения и ликвидации чрезвычайных ситуаций и населения.

Системы оповещения создаются:

- на федеральном уровне – федеральная система оповещения (на территории Российской Федерации);
- на межрегиональном уровне – межрегиональная система оповещения (на территории федерального округа);
- на региональном уровне – региональная система оповещения (на территории субъекта Российской Федерации);
- на муниципальном уровне – местная система оповещения (на территории муниципального образования);
- на объектовом уровне – локальная система оповещения (в районе размещения потенциально опасного объекта).

Оповещение сельских районов осуществляется по более сложной схеме. В региональные системы централизованного оповещения включены только

районные центры, а население в других населенных пунктах сельской местности оповещается в основном по сетям радио и телевидения, сельской телефонной сети, мобильными звукоусилительными средствами отделов внутренних дел и путем подворного обхода

Системы оповещения предназначены для обеспечения своевременного доведения информации и сигналов оповещения до органов управления, сил и средств гражданской обороны, РСЧС и населения об опасностях, возникающих при ведении военных действий или вследствие данных действий, а также угрозе возникновения или возникновения чрезвычайных ситуаций природного и техногенного характера.

Звуковому обеспечению станций метрополитена необходимо уделять должное внимание, что было отмечено и высшим руководством Российской Федерации. Аппаратура громкоговорящего оповещения должна обеспечивать:

- распространение рекламных сообщений;
- объявление о приближении состава;
- для объявления пропавших вещей или людей;
- воспроизведение фоновой музыки;
- режим работы метрополитена.

Основными компонентами системы централизованного оповещения являются:

- центр управления: это централизованный пункт управления, где собирается и анализируется информация о чрезвычайной ситуации, операторы в центре управления принимают решения о необходимости оповещения и управляют процессом передачи сообщений;

- различные средства связи: как телефоны, радиостанции, мобильные устройства и интернет, используемые для передачи оповещений различным группам людей или организациям;

- автоматизированные системы оповещения: специальные устройства, как сирены, громкоговорители, мобильные приложения и прочее,

которые могут автоматически воспроизводить предварительно записанные сообщения или тоновые сигналы для предупреждения людей о нештатной ситуации.

В данной работе будет рассматриваться трансляция информационных и чрезвычайных сообщений на станции метрополитенов с единого диспетчерского центра. Приложение не будет касаться моментов, как:

- передача полученной аудиотрансляции на громкоговорители;
- обработка полученных звуковых файлов.

Предмет работы: разработка программного имитатора системы централизованного оповещения в информационной системе метрополитена, представляющего клиент-серверное взаимодействие по передаче звуковых файлов.

Цель данной работы: разработка программного имитатора системы централизованного оповещения в информационной системе метрополитена.

1 Постановка задачи и анализ разрабатываемой системы

1.1 Постановка задачи

Сервер системы централизованного оповещения должен обеспечивать выполнение следующих основных функций:

- проверка наличия запросов от клиентов;
- выбор конечных точек для запуска передачи аудиопотока;
- остановка воспроизведения аудиопотока;
- обработка запросов состояния конечных точек.

Сервер СЦО должен использовать протоколы взаимодействия, реализованные на базе сетевых протоколов транспортного уровня TCP и RTP и должен обеспечивать возможность задания следующих основных параметров:

- локальный IP адрес интерфейса Ethernet клиента, подключенного к сети;
- локальный IP адрес интерфейса Ethernet сервера, подключенного к сети;
- выбор необходимых узлов станций для передачи аудиосигнала.

Пользователь вводит IP адреса сервера и клиента. После приложение должно проверять наличие подключения к клиенту, если подключение не установлено, на панели должно отобразиться сообщение об ошибке, если установлено, сообщение об успешном подключении, и создадутся TCP и RTP сокет для дальнейшей работы.

Далее пользователь должен нажать кнопку, запускающая поток для проверки запросов на подключение.

TCP запросы поступают в виде сообщений, в которых указываются номера конечных точек, готовых принять аудиопоток от сервера и должны выглядеть следующим образом.

```
<REQUEST client = "US-MUS.04.01.1F" command = "connection"
action="connect" state= "free" prefix = "US-MUS"/>
```

Запрос собирается из составных частей:

- ключевое слово REQUEST, непосредственно означающее, что поступает запрос;
- client = “US-MUS.04.01.1F” – станция и узел;
- command = “connection” – тип поступающего запроса, в данный момент подключения;
- action=“connect” – действие, которое нужно совершить;
- state= “free” – статус узла;
- prefix = “US-MUS” – условное обозначение узла.

После принятия запроса и его последующей обработки необходимо отправить ответ клиенту, с указанием, что запрос получен и обработан.

Запрос ответа клиенту должен выглядеть следующим образом.

```
<RESPONSE client = “US-MUS.04.01.1F” command = “connection”
action=“connect” state= “free” prefix = “US-MUS”/>
```

В запросе меняется ключевое слово REQUEST на RESPONSE, означающее «ответ».

После разблокировки на панели доступных станций, пользователь должен выбрать оконечные точки, на которые пойдет передача, файл, который нужно передать и после нажать на соответствующую кнопку, отправляющую запрос на передачу аудиопотока и после ответа о готовности, запускающую RTP поток для передачи.

RTP запрос должен выглядеть следующим образом:

```
<REQUEST client = “US-MUS.04.01.1F” command= “receive” action=
“start” session= “1501789000001” url= “rtp://192.168.1.12:55880” priority = “1”
source= “server_file:file137” streamer= “server”>
```

Запрос составляется из составных частей:

- ключевое слово REQUEST, означающее, что поступает запрос;
- client = “US-MUS.04.01.1F” – станция и узел;
- command = “receive” – тип поступающего запроса, в данный момент передачи;
- action=“start” – действие, которое нужно совершить;

- session= “1501789000001” – номер сессии;
- url= “rtp://192.168.1.12:55880” – так же состоит из составных частей: 192.168.1.12, отвечающий за IP адрес клиента и 55880, отвечающий за порт;
- priority = “1” – приоритет передачи;
- source= “server_file:file137” – название передающегося файла;
- streamer= “server” – объект, отвечающий за передачу;

Ответ должен выглядеть следующим образом:

```
<RESPONSE client = “US-MUS.04.01.1F” command= “receive” action=
“start” session= “1501789000001” url= “rtp://192.168.1.12:55880” priority = “1”
source= “server_file:file137” streamer= “server” result= “accepted”>
```

В запросе меняется ключевое слово REQUEST на RESPONSE, означающее «ответ», и добавляется result= “accepted”, означающая, что узел готов к передаче.

Аудиофайл должен отправляться сообщениями по частям. Сообщение формируется как показано на рисунке 1.

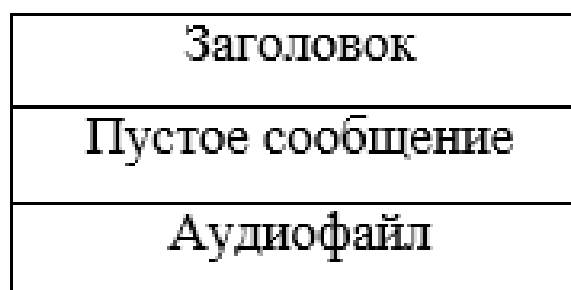


Рисунок 1 – Формировка сообщения

При более подробном разборе получаем:

- первый байт заполняется версией протокола RTP;
- второй байт аудиокодеком;
- байты с 3 по 15 пустые;
- аудиофайл занимает следующие 851 байт.

1.2 Выбор технологий проектирования и разработки

Перед разработкой приложения нужно определиться со средствами реализации.

Для проектирования информационной системы требуется построение UML-диаграмм: диаграмма вариантов использования, диаграмма деятельности, диаграмма последовательности, диаграмма классов. Для построения перечисленных диаграмм были выбраны такие CASE-средства, как Visual Paradigm.

Visual Paradigm – программа для создания графических схем и диаграмм с широким набором инструментов визуализации для наглядного представления сложных данных, таких как принципиальные схемы, комплексные системы многоуровневых процессов, блок-схемы или диаграммы. Visual Paradigm может использоваться для создания различных типов диаграмм, как блок-схемы, организационные диаграммы, диаграммы потоков работы, диаграммы Ганта, UML-диаграммы.

При выборе CASE-средств для проектирования программного имитатора мы учитываем следующие факторы: наличие необходимого функционала, удобный пользовательский интерфейс и легкость использования.

Для разработки программного имитатора системы централизованного оповещения информационной системы метрополитена были выбраны следующие технологии и инструменты: платформа Visual Studio 2022, язык программирования C++.

Visual Studio 2022 является лучшей средой разработки многофункциональных, кроссплатформенных приложений для таких платформ, как Windows, iOS, Mac, Linux и Android. Имеет в распоряжении множество языков программирования, удобный и отзывчивый интерфейс. Дает возможность пользователю интегрировать любые необходимые библиотеки, инструменты, платформы, такие как: .NET Framework, .Net Core, Xbox и другие [5].

C++ является высокотехнологичным языком программирования и широко используется в разработке:

- операционных систем;
- прикладных программ;
- драйверов;
- высокопроизводительных серверов;
- компьютерных игр.

Данный язык программирования был создан в 1979 году в «Bell Labs», как расширения языка программирования C. C++ отличается от своего предшественника большим набором возможностей:

- объектно-ориентированное программирование: C++ поддерживает принципы наследования, полиморфизм и инкапсуляцию, что позволяет создавать модульные и гибкие программы;
- мультипарадигменный язык: C++ объединяет различные парадигмы программирования, включая процедурное программирование, ООП и обобщенное программирование;
- высокая производительность: C++ предоставляет низкоуровневый доступ к аппаратному обеспечению компьютера, что делает данный язык отличным выбором для разработки производительных приложений;
- богатая стандартная библиотека: C++ имеет обширную стандартную библиотеку, которая включает контейнеры, алгоритмы, ввод-вывод и многие другие возможности;
- поддержка различных операционных систем: C++ поддерживает разные операционные системы, что позволяет языку быть популярным выбором для создания многоплатформенного программного обеспечения.

Выбранные технологии разработки и проектирования информационных систем позволяют с успехом выполнить необходимую работу, одновременно являясь при этом удобными и эффективными в использовании.

2 Планирование и расчет бюджета разрабатываемой системы

Для успешного управления проектами требуется определить конечные цели и обеспечить выполнение, путем планирования необходимых ресурсов, времени, качества и бюджета проекта, а также проведения оценки возможных рисков и трудозатрат. Наличие подробного плана заранее также играет ключевую роль в обеспечении эффективного проектного управления.

Для планирования и расчета бюджета разработки программного имитатора системы централизованного оповещения в информационной системе метрополитена необходимо пройти этапы, перечисленные ниже:

- определить объем и спецификации функциональных требований, позволяющих определить необходимые модули и функции системы;
- оценить трудозатраты и затраты на разработку системы. нужно учитывать затраты на анализ требований, проектирование, реализацию, тестирование и документацию. также нужно учитывать затраты на оборудование;
- составить план работ, расписание и бюджет. на основе оценки трудозатрат и затрат можно определить сроки выполнения работ и необходимый бюджет. расписание позволит управлять проектом и отслеживать выполнение работ в соответствии со сроками;
- оценить экономическую эффективность системы.

Существует несколько формализованных методов для планирования и управления выполнением проектных работ. Один из основных методов – построение линейного графика Ганта.

Диаграмма Ганта – инструмент визуализации временных интервалов выполнения задач и проектов, позволяющая представить графически этапы проекта, от начала до конца, и определить продолжительность каждой задачи [6].

Назначение диаграммы Ганта – помочь определить общую продолжительность проекта, сроки выполнения каждой задачи, количество

ресурсов, необходимых для выполнения проекта, а также выявить возможные риски и проблемы, которые могут возникнуть в процессе выполнения задач [7].

Построение диаграммы Ганта для осуществления планирования и расчета бюджета в программе для управления проектами GanttPRO.

В ходе планирования проекта было выделено 22 задачи, описание которых представлено в таблице 1.

Таблица 1 – Задачи проекта

№ п/п	Название задачи	Длительность	Начало	Окончание	Предшественники
1	Начало проекта	0 дней	12.04.2024	12.04.2024	
2	Анализ предметной области и требований к системе	3 дня	12.04.2024	16.04.2024	
3	Анализ бизнес-процессов предприятия	1 день	12.04.2024	12.04.2024	
4	Анализ требований и постановка задач	1 день	15.04.2024	15.04.2024	3
5	Выбор технологий проектирования и разработки	1 день	16.04.2024	16.04.2024	4
6	Завершение анализа предметной области и требований к системе	0 дней	17.04.2024	17.04.2024	2
7	Проектирование информационной системы	2 дня	17.04.2024	18.04.2024	
8	Проектирование UML диаграмм	1 день	17.04.2024	17.04.2024	
9	Проектирование пользовательского интерфейса	1 день	18.04.2024	18.04.2024	8
10	Завершение проектирования информационной системы	0 дней	19.04.2024	19.04.2024	7
11	Реализация информационной системы	16 дней	19.04.2024	10.05.2024	
12	Завершение реализации информационной системы	0 дней	13.05.2024	13.05.2024	11
13	Тестирование программных средств	6 дней	13.05.2024	20.05.2024	
14	Функциональное тестирование	2 дня	13.05.2024	14.05.2024	
15	Модульное тестирование	2 дня	15.05.2024	16.05.2024	14
16	Анализ метрик кода	2 дня	17.05.2024	20.05.2024	15
17	Завершение тестирования программных средств	0 дней	21.05.2024	21.05.2024	13
18	Оформление документации	9 дней	21.05.2024	31.05.2024	
19	Написание документации	7 дней	21.05.2024	29.05.2024	
20	Оформление и печать документации	2 дня	30.05.2024	31.06.2024	19
21	Завершение оформления документации	0 дней	03.06.2024	03.06.2024	18
22	Конец проекта	0 дней	03.06.2024	03.06.2024	21

Также были определены необходимые ресурсы. Список ресурсов представлен в таблице 2 [8].

Таблица 2 – Ресурсы и затраты проекта

№ п/п	Название ресурса	Тип ресурса	Стандартная ставка	Начисление
1	Аналитик	Трудовой	200,00 Р/ч	Пропорциональное
2	Проектировщик	Трудовой	300,00 Р/ч	Пропорциональное
3	Программист	Трудовой	500,00 Р/ч	Пропорциональное
4	Тестировщик	Трудовой	350,00 Р/ч	Пропорциональное
5	Компьютер	Затраты		В начале
6	Принтер	Затраты		В начале
7	Бумага	Материальный	300 Р	В начале
8	Интернет	Затраты		В начале
9	Электроэнергия	Затраты		По окончании

Выделенные задачи и ресурсы были занесены в проект в GanttPRO.

Результаты представлены на рисунке 2.

Задача	Начало	Длительность	Предшественник	Завершение	Название ресурсов
	12.04.2024	36д 2ч		03.06.2024	
1 Начало проекта	12.04.2024	1ч		12.04.2024	Бумага
2 ☒ Анализ предметной области и требований к системе	12.04.2024	3д		16.04.2024	
2.1 Анализ бизнес-процессов предприятия	12.04.2024	1д		12.04.2024	Аналитик
2.2 Анализ требований и постановка задачи	15.04.2024	1д	2.1	15.04.2024	Аналитик
2.3 Выбор технологий проектирования и разработки	16.04.2024	1д	2.2	16.04.2024	Аналитик
3 Завершение анализа предметной области и требований к системе	17.04.2024	1ч	2	17.04.2024	
4 ☒ Проектирование информационной системы	17.04.2024	2д		18.04.2024	
4.1 Проектирование UML диаграмм	17.04.2024	1д		17.04.2024	Проектировщик
4.2 Проектирование пользовательского интерфейса	18.04.2024	1д	4.1	18.04.2024	Проектировщик
5 Завершение проектирования информационной системы	19.04.2024	1ч	4	19.04.2024	
6 Реализация информационной системы	19.04.2024	16д		10.05.2024	Программист
7 Завершение реализации информационной системы	13.05.2024	1ч	6	13.05.2024	
8 ☒ Тестирования программных средств	13.05.2024	6д		20.05.2024	
8.1 Функциональное тестирование	13.05.2024	2д		14.05.2024	Тестировщик
8.2 Модульное тестирование	15.05.2024	2д	8.1	16.05.2024	Тестировщик
8.3 Анализ метрик кода	17.05.2024	2д	8.2	20.05.2024	Тестировщик
9 Завершение тестирования программных средств	21.05.2024	1ч	8	21.05.2024	
10 ☒ Оформление документации	21.05.2024	9д		31.05.2024	
10.1 Написание документации	21.05.2024	7д		29.05.2024	Аналитик
10.2 Оформление и печать документации	30.05.2024	2д	10.1	31.05.2024	Аналитик
11 Завершение оформления документации	03.06.2024	1ч	10	03.06.2024	
12 Конец проекта	03.06.2024	1ч	11	03.06.2024	

Рисунок 2 – Задачи проекта с распределенными по ним ресурсами в GanttPRO

В соответствии с занесенными задачами и ресурсами была построена диаграмма Ганта, представленная на рисунке 3.

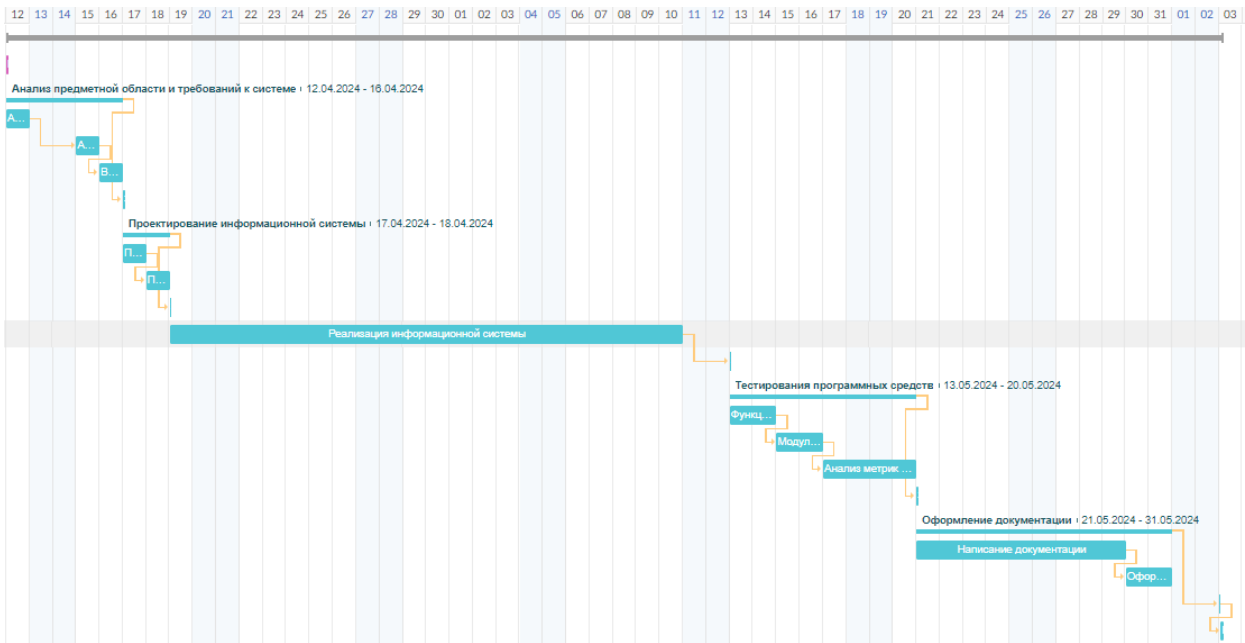


Рисунок 3 – Диаграмма Ганта

В соответствии с построенной диаграммой была сформулирована таблица по затратам на разработку, которая представлена на таблице 3.

Таблица 3 – Затраты на разработку

Название	Фактические затраты	Оставшиеся затраты	Базовые затраты	Затраты	Отклонение по стоимости
Начало проекта	300,00	3064,00	0,00	3364,00	3364,00
Анализ предметной области и требований к системе	4800,00	0,00	0,00	4800,00	4800,00
Завершение анализа предметной области и требований к системе	0,00	0,00	0,00	0,00	0,00
Проектирование информационной системы	4800,00	0,00	0,00	4800,00	4800,00

Продолжение таблицы 3

Название	Фактические затраты	Оставшиеся затраты	Базовые затраты	Затраты	Отклонение по стоимости
Завершение проектирования информационной системы	0,00	0,00	0,00	0,00	0,00
Реализация информационной системы	64000,00	0,00	0,00	64000,00	64000,00
Завершение реализации информационной системы	0,00	0,00	0,00	0,00	0,00
Тестирование программных средств	16800,00	0,00	0,00	16800,00	16800,00
Завершение тестирования программных средств	0,00	0,00	0,00	0,00	0,00
Оформление документации	14400,00	0,00	0,00	14400,00	14400,00
Завершение оформления документации	0,00	0,00	0,00	0,00	0,00
Конец проекта	106,00	0,00	0,00	106,00	106,00

Таким образом, в результате функционально-стоимостного анализа, временные затраты для выполнения проекта составят 53 дня, включая выходные и праздничные дни, а стоимость выполнения, с учетом прочих затрат на ресурсы, составит 108270 рублей:

- начало проекта – 3364 рубля;
- анализ предметной области и требований к системе – 4800 рублей;
- проектирование информационной системы – 4800 рублей;
- реализация информационной системы – 64000 рублей;

- тестирование программных средств – 16800 рублей;
- оформление документации – 14400 рублей;
- конец проекта – 106 рублей.

Оценка стоимости является неотъемлемой частью планирования любого проекта, позволяющая установить возможный бюджет на разработку, определить оптимальный способ расходования средств и контролировать затраты на всех этапах работы над проектом. Оценка дает возможность руководству и исполнителям проекта понимать, какие ресурсы нужны, чтобы выполнить задачи в соответствии с ожиданиями, анализировать стоимостную эффективность различных решений и планировать нужные корректировки. Более того, правильная оценка стоимости создает базу для принятия качественных решений и снижения рисков в процессе выполнения проекта.

Разработка всего бюджета, полного или реального, практически никогда не удастся с первого раза. В него постоянно вносятся правки и корректировки в течении выполнения проекта.

3 Проектирование информационной системы

3.1 Анализ бизнес-процессов предприятия

Перед разработкой программного имитатора необходимо провести анализ бизнес-процессов. Основные бизнес-процессы представлены на таблице 4.

Таблица 4 – Бизнес-процессы

№	Бизнес-процесс	Входные данные		Выходные данные	
		Поставщик	Содержание	Потребитель	Содержание
1.	Проверка IP адреса	Пользователь	IP адреса клиента и сервера	Система	Подтверждение в правильности адресов
2.	Проверка наличия подключений	Пользователь	IP клиента	Система	Запрос на подключение
3.	Получение узлов	Система	Запрос	Система	Узел
4.	Отправка аудиофайла	Пользователь	Аудиофайл и узел	Система	Успешно отправлено
5.	Остановка отправки аудиофайла	Пользователь	Команда	Система	Остановка отправки

В дальнейшем данные бизнес-процессы будут использоваться при проектировании и реализации разрабатываемого имитатора СЦО.

3.2 Проектирование программного обеспечения

3.2.1 Диаграмма состояний

Диаграмма состояний – это графическое представление модели поведения системы, которое описывает возможные состояния объектов системы и переходы между ними. Диаграмма состояний состоит из состояний, переходов и событий.

Диаграмма состояний используется для моделирования поведения системы и определения требований. Диаграмма показывает, как система или отдельные объекты должны вести в различных ситуациях и как могут быть связаны друг с другом [12].

Диаграмма состояний, представленная на рисунке 4.

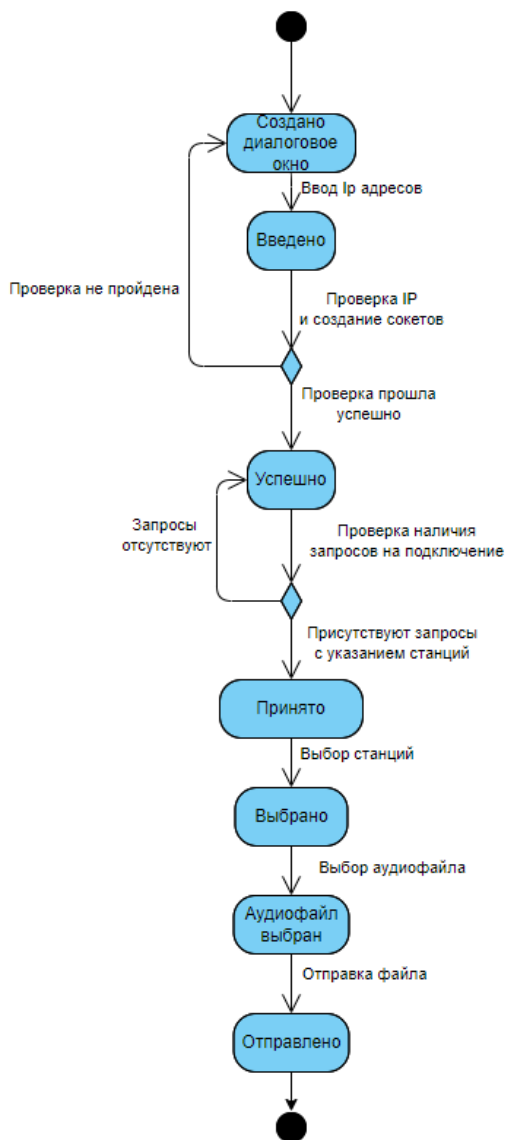


Рисунок 4 – Диаграмма состояний

Спецификации переходов системы из одного состояния в другое определены в таблице 5.

Таблица 5 – Спецификации переходов системы из состояния в состояние

Состояние 1	Событие	Состояние 2	Описание перехода
Начальное состояние	Запустить программу	Создано диалоговое окно	При запуске программы происходит переход из состояния «Начальное состояние» в состояние «Создано диалоговое окно»
Создано диалоговое окно	Ввод IP адресов	Введено	Происходит ввод IP адресов сервера и клиента

Продолжение таблицы 5

Состояние 1	Событие	Состояние 2	Описание перехода
Введено	Проверка IP и создание сокетов	Успешно	Происходит Пинг IP адресов клиента и сервера и создание сокетов. Если IP адреса введены верно, создаются TCP и RTP сокет и система переходит в состояние «Успешно», если IP адреса введены неверно, то система перейдет в состояние «Создано диалоговое окно»
		Создано диалоговое окно	
Успешно	Проверка наличия запросов на подключение	Успешно	При наличии запросов на подключение система переходит в состояние «Принято», если запросов нет, то система остается в режиме ожидания запросов на подключение
		Принято	
Принято	Выбор станций	Выбрано	Происходит выбор станций, на которые будет рассылаться аудиофайл
Выбрано	Выбор аудиофайла	Аудиофайл выбран	Выбирается аудиофайл для отправки
Аудиофайл выбран	Отправка файла	Отправлено	Файл отправляется непосредственно клиенту
Отправлено	Завершение программы	Конечное состояние	Программа завершает свою работу

Таким образом, была разработана диаграмма состояний, которая показала возможные состояния системы.

3.2.2 Диаграмма деятельности

Диаграмма деятельности, как и диаграмма состояний отражает динамические аспекты поведения системы.

Суть диаграммы заключается в графическом отображении последовательности действий и представляет собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой. Активности на диаграмме «разбросаны» по беговым дорожкам, каждая из которых соответствует поведению одного из объектов, например, клиента, менеджера, веб-сервера, сервера БД и так далее. Следовательно, легко определить, каким из объектов выполняется каждая из активностей. Для представления последовательности выполнения основных задач

автоматизированной информационной системы была составлена диаграмма деятельности, представленная на рисунке 5 [9].

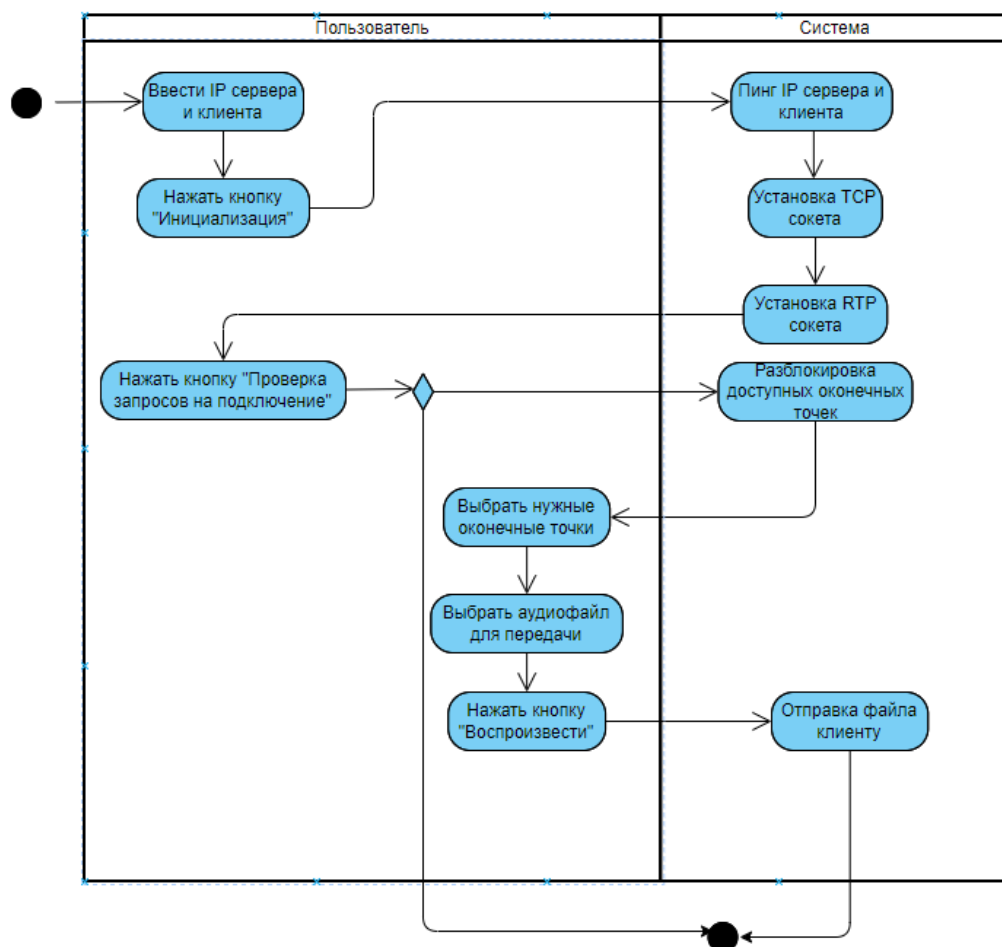


Рисунок 5 – Диаграмма деятельности

На представленной диаграмме фигурируют два участника: пользователь и система.

Входной точкой процесса является ввод пользователем IP адресов сервера и клиента. После нажатия кнопки «Инициализация» система проверит наличие подключения к клиенту и установит сокет TCP и RTP. Далее пользователь, нажав кнопку «Проверка запросов на подключение» либо получит список запросов в которых указывается готовые для получения файла оконечные точки, либо, если не будет подключений, система останется в режиме ожидания. Следующим шагом пользователю нужно выбрать станции, на

которые будет идти передача, а также аудиофайл. И после нажатия кнопки «Воспроизвести» начнется отправка выбранного файла.

3.2.3 Диаграмма вариантов использования

Диаграмма вариантов использования — диаграмма, описывающая функционал разрабатываемой программной системы доступен каждой группе пользователей [10].

На диаграмме вариантов использования, представляющими функции системы, и действующими лицами, представляющими людей или системы, получающие или передающие информацию в данную систему. Из диаграмм вариантов использования можно получить довольно много информации о системе. Данный тип диаграмм описывает общую функциональность системы. Пользователи, менеджеры проектов, аналитики, разработчики, специалисты по контролю качества и другие, заинтересованные системой в целом, могут, изучая диаграммы вариантов использования, понять, что система должна делать.

Обычно при проектировании разработчики изображают систему графически, поскольку человеку легко разобраться в таком представлении. Именно поэтому вместо написания громоздких текстов про каждую возможность будущей программы разработчики строят различные диаграммы для описания своих систем, помогающих не забывать, что нужно реализовать в программе, и быстро вводить в курс дела коллег.

Благодаря проведению анализа предметной области были выделены следующие функции, которые должны быть реализованы в разрабатываемой системе:

- ввод IP адресов клиента и сервера;
- отправка аудиопотока клиенту.

Для наглядности всех основных функций имитатора была составлена диаграмма вариантов использования, представленная на рисунке 6.

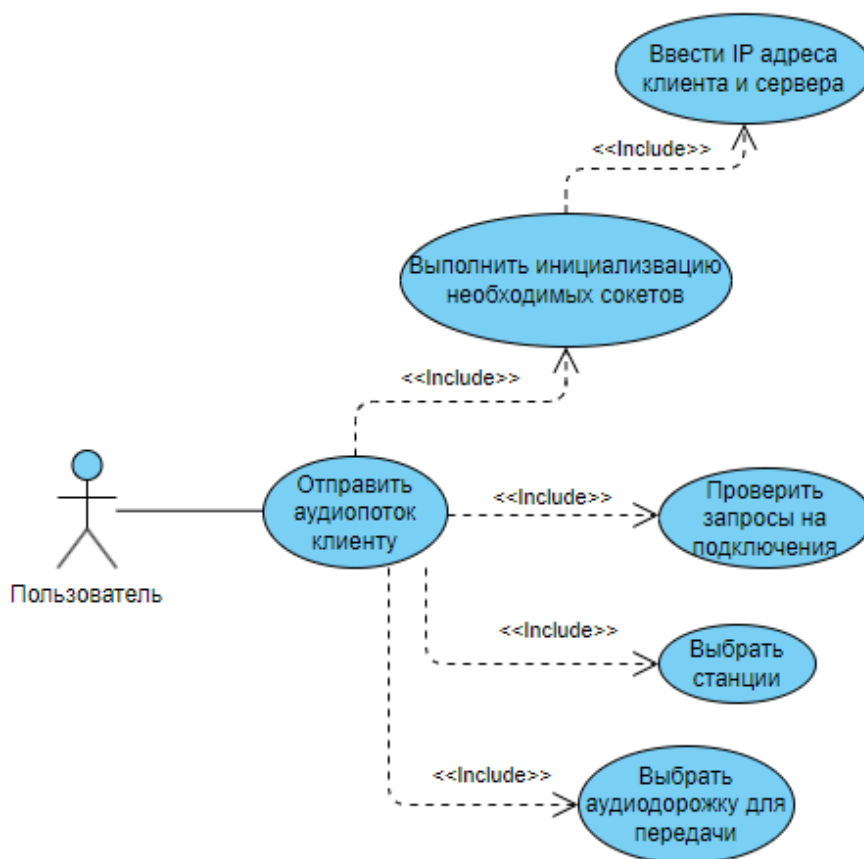


Рисунок 6 – Диаграмма вариантов использования

На диаграмме представлен пользователь, с возможностью передачи аудиопотока клиенту. Сначала необходимо установить подключение к клиенту посредством ввода IP адресов и инициализации необходимых сокетов, а именно TCP и RTP. Далее нужно проверить запросы на подключение. Если присутствуют входящие запросы на подключение, то необходимые узлы разблокируются и пользователю необходимо будет выбрать нужные станции и файл для передачи.

3.2.4 Диаграмма классов

Диаграмма классов – диаграмма языка моделирования UML, показывающая общую структуру и иерархию классов выбранной предметной области [11].

Диаграмма классов определяет типы классов системы и различного рода статические связи, существующие между друг другом. На диаграммах классов

изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами. Вид и интерпретация диаграммы классов существенно зависят от точки зрения:

- классы могут представлять сущности предметной области;
- элементы программной системы.

Виды связей между классами могут быть следующими:

- ассоциация, представляющая связь типа «один ко многим», либо «многие ко многим»;
- агрегация, представляющая типа «целое-часть»;
- композиция, представляющая собой связь, где объекты не могут существовать сами по себе и уничтожаются при уничтожении объекта агрегирующего класса;
- наследование, представляющее собой отношение «общее-частное».

Диаграмма классов представлена на рисунке 7.

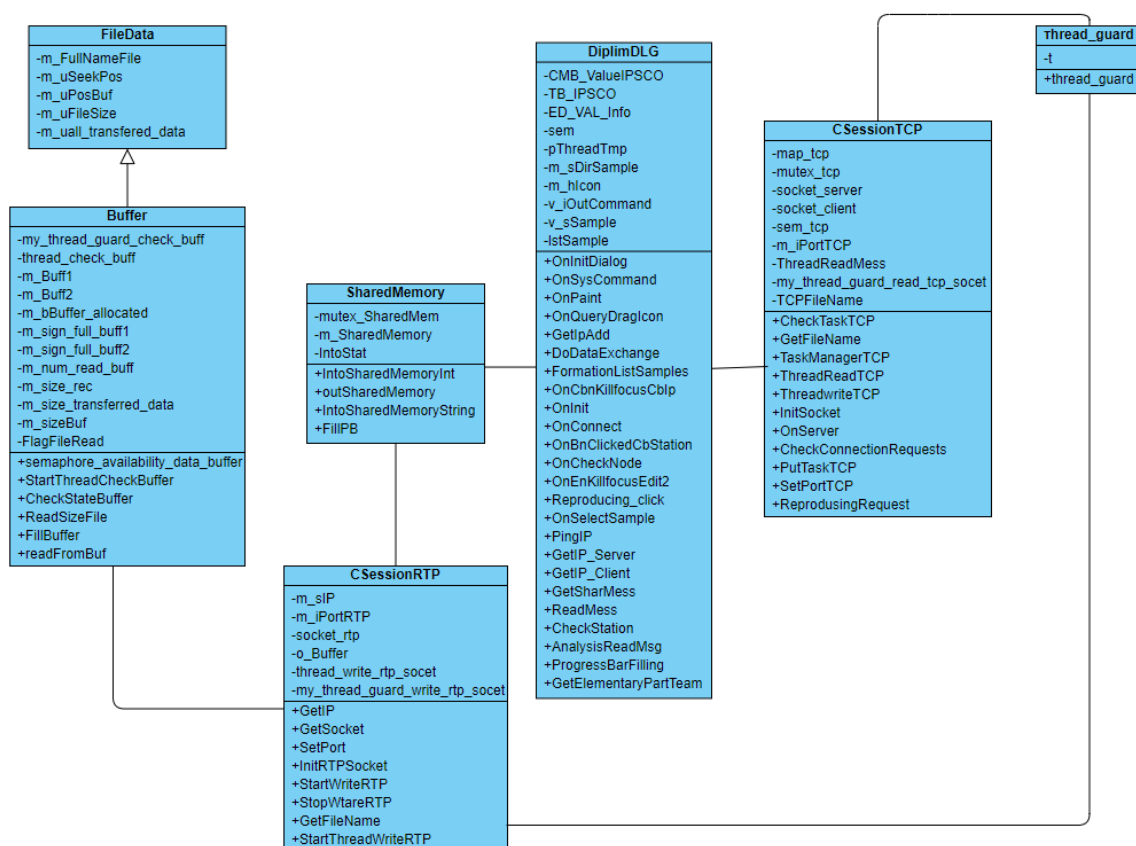


Рисунок 7 – Диаграмма классов

На диаграмме представлена структура классов программы.

Класс «DiplomDLG» используется для взаимодействия с пользовательским интерфейсом, поиска IP адресов, рассылки команд остальным классам и имеет доступ к «CSessinTCP» и «SharedMemory». Компонентами, которыми обладает класс, приведены в таблице 6.

Таблица 6 – Компоненты класса «DiplomDLG»

№	Компонент	Описание	Аргументы
1	OnInitDialog	Генерация диалогового окна с выводом IP адресов	MyIp – Вектор с IP адресами Schetchik – Содержит информацию о количестве IP адресов a – Содержит информацию об выбранном IP адресе IP – Содержит информацию об IP адресе, выделенном в документе Ip_SCO – Содержит информацию об IP адресе СЦО в документе HSC – Префикс перед IP адресом ШСЦ в документе SCO – Префикс перед IP адресом СЦО в документе CMB_ValueIPSCO Значение строки ввода СЦО IP адреса на форме IP_Change – Содержит информацию об IP адресе, выделенном в документе TB_IPHSHC Значение строки ввода ШСЦ IP адреса на форме
2	OnSysCommand	Системное меню	
3	OnPaint	Вызывается, когда окно подключаемого модуля должно закраситься.	
4	OnQueryDragicom	Возвращает идентификатор пиктограммы приложения.	
5	GetIpAdd	Получение IP адресов компьютера	
6	DoDataExchange	Содержит информацию об элементах на форме	
7	OnBnClickedCbStation	Позволяет выбрать необходимые узлы	bStation – Содержит информацию о станции iNumStation – Счетчик количества выбранных станций

Продолжение таблицы 6

№	Компонент	Описание	Аргументы
8	OnCbnKillFocusCblp	Автоматически выбирает из списка нужный IP адрес	IP_enter – Аргумент для выбора IP адреса Perenos – Хранит текст из документа с IP адресами In Открывает текстовый документ Out Закрывает текстовый документ HSC – Префикс перед IP адресом ШСЦ в документе SCO – Префикс перед IP адресом СЦО в документе
9	OnInit	Выполняет инициализацию сокетов	
10	OnConnect	Проверяет наличие подключений	iKey – Содержит уникальный ключ запроса
11	FormationListSamples	Ищет информацию об аудиофайлах для передачи	m_File – Содержит название папки с аудиофайлами strCurDir – Преобразует название папки в класс для работы в коде. dwAttr – Содержит информацию о наличие аудиофайлов в папке dwError – Содержит информацию об ошибках m_finder – Содержит название файла, выбранного на форме
12	OnCheckNode	Выполняет проверку узлов и на основе этого выделяет при необходимости станции	bNode – Информацию об узле
13	OnEnKillFocusEdit2	Автоматически заполняет из документа IP адрес ШСЦ	IP_enter – Аргумент для выбора IP адреса Perenos – Хранит текст из документа с IP адресами In Открывает текстовый документ Out Закрывает текстовый документ HSC – Префикс перед IP адресом ШСЦ в документе SCO – Префикс перед IP адресом СЦО в документе
14	Reproductin_click	Сообщает о начале запуска передачи аудиопотока клиенту	bNode – Информация о станциях FileName – Название аудиофайла
15	OnSelectSample	Функция, позволяющая менять выбор аудиофайла	Val2 – Индекс аудиофайла

Окончание таблицы 6

№	Компонент	Описание	Аргументы
16	PingIp	Проверяет наличия подключения	Ls_res – Результирующая строка для вывода на форму L_sIp – IP адрес iLenIp – Длина IP адреса buff_ip – буффер для хранения IP адресов RecvAddr – указывает адрес транспорта и порт семейства адресов AF_INET
17	GetIp_Server	Вывод IP адреса СЦО на форму	
18	GetIp_Client	Вывод IP адреса ШСЦ на форму	
19	GetSharMess	Выводит на форму сообщение о результате подключения	
20	ReadMess	Выводит на экран поступившие запросы на подключение	
21	CheckStation	Выполняет разблокировку станций	Unlock_CB – Содержит запрос StationFindBegin – Содержит начальный индекс позиции номера станции и узла StationUnlockCB – Содержит информацию об узле и станции для разблокировки StationFindEnd – Содержит конечный индекс позиции номера станции и узла
22	AnalysisReadMsg	Анализирует поступившие запросы для ответа	p_sMsg – Содержит информацию о запросе sCommand – содержит информацию о типе запроса
23	ProgressBarFilling	Имитирует прогресс загрузки	
24	GetElementaryPartTeam	Анализирует тип поступившего запроса	

Класс «CSessionTCP» служит для установки подключения с клиентом и имеет доступ к «CDiplomDLG» и «Thread_guard». Компоненты, которыми обладает класс, приведены в таблице 7.

Таблица 7 – Компоненты класса «CSessionTCP»

№	Компонент	Описание	Аргументы
1	CheckTaskTCP	Проверка на наличие задач	uKey – ИндексЗадачи
2	GetFileName	Передаёт в RTP название файла	p_FileName – Название файла

Продолжение таблицы 7

№	Компонент	Описание	Аргументы
3	ThreadwriteTCP	Поток для отправки запросов ответа	OutCommand – Содержит информацию о типе запроса
4	ThreadReadTCP	Поток для принятия и обработки запросов	Mess – Содержит обработанный запрос sReceivedMessage – Содержит один запрос из всей кучи TimeLess – Конвертирует запрос в CString Buff_prm – буффер со всеми запросами iStartingPos – Позиция начала запроса iEndPos – Позиция конца запроса status – Статус приема запросов
5	TaskManagerTCP	Передаёт в память поступившие запросы	iKey – Уникальный ключ запроса ConnRequest – Индекс наличия запросов svBufMsg – Содержит текст запроса
6	InitSocket	Выполняет инициализацию TCP сокета	vVersionRequested – Версия сокета wsData – Сведения о реализации сокета Windows
7	OnServer	Выполняет подключение к клиенту	Socket_server – Сокет сервера Socket_client – Сокет клиента Local_addr – Указывает адрес транспорта и порт для семейства адресов AF_INET
8	CheckConnectionRequest	Проверяет на наличие подключений	client_addr – Указывает адрес транспорта и порт для семейства адресов AF_INET s_set – Используется для размещения сокетов в наборе. numMsg – Уникальный идентификатор подключения
9	SetPortTCP	Определяет порт для TCP	Sem_tcp – Выполняет синхронизацию работы процессов и потоков syncTCP – Определяет механизм контроля доступа к ресурсу
10	ReproducingRequest	Отправляет запрос о готовности отправки аудиофайла	sClient – Содержит информацию об узле Num_session – Содержит информацию о номере сессии FileSize – Содержит информацию о размере файла cNameFile – Содержит информацию о названии файла str – Содержит информацию о конечной форме запроса

Класс «Thread_guard» служит для контроля программы в случае возникновения непредвиденных ошибок, останавливая работающие потоки во избежание утечки памяти и имеет доступ к моделям, имеющим потоки, а именно «CSessionTCP» и «CSessionRTP».

Класс «CSessionRTP» служит для непосредственной передачи аудиопотока клиенту и имеет доступ к «Buffer» и «SharedMemory». Компоненты, которыми обладает класс, приведены в таблице 8.

Таблица 8 – Компоненты класса «CSessionRTP»

№	Компонент	Описание	Аргументы
1	GetIp	Получает IP адрес клиента	sIP – IP адрес
2	GetSocket	Получает сокет RTP	Tmp_socket – RTPсокет
3	SetPort	Получает порт	M_iPortRTP – порт
4	InitRTPSocket	Инициализация RTP сокета	srv_address – Указывает адрес транспорта и порт для семейства адресов AF_INET v_Ip_SCO – Содержит IP адрес СЦО buff_ip – Буффер с IP адресами iLenIP – Содержит длину IP адреса
5	StartWrateRTP	Производит отправку аудиофайла	buff_ip – Буффер с IP адресами s_address – Указывает адрес транспорта и порт для семейства адресов AF_INET rtp_hdr – Заголовок RTP Send_buf – Буфер с данными аудиофайла FileSize – Размер аудиофайла Progress – Прогресс передачи Sell_Progress – Прогресс передачи за одну отправку PercentProgress – Прогресс передачи в процентах dat – Количество отправок
6	StopWriteRTP	Производит остановку отправки аудиофайла	
7	GetFileName	Получает имя файла	P_FileName – Содержит название файла
8	StartThreadWriteRTP	Запускает поток отправки аудиофайла	

Класс «SharedMemory» служит для обработки полученных запросов и имеет доступ к «CSessionTCP» и «CDiplomDLG». Компоненты, которыми обладает класс, приведены в таблице 9.

Таблица 9 – Компоненты класса «SharedMemory»

№	Компонент	Описание	Аргументы
1	FillPB	Передаёт прогресс отправки форме	Progress – Прогресс
2	IntoSharedMemoryInt	Кладёт в память значения типа int	MessageKey – Необходимые данные iKey – Уникальный ключ
3	IntoSharedMemoryString	Кладёт в память значения типа string	MessageKey – Необходимые данные iKey – Уникальный ключ
4	OutSharedMemory	Убирает из памяти выбранные значения	OutMess – Необходимые данные iKey – Уникальный ключ
5	FillPB	Передаёт прогресс отправки форме	Progress – Прогресс

Класс «Buffer» служит для обработки аудиофайла в побитовую форму для передачи и имеет доступ к «CSessionRTP» и «FileData». Компоненты, которыми обладает класс, приведены в таблице 10.

Таблица 10 – Компоненты класса «Buffer»

№	Компонент	Описание	Аргументы
1	Semaphore_availability_data_buffer	Проверка на заполненность и смену буфера	m_sign_full_buff1 – Флаг заполненности первого буфера m_sign_full_buff2 – Флаг заполненности второго буфера
2	StartThreadCheckBuffer	Запускает поток функции CheckStateBuffer	
3	CheckStateBuffer	Проверка на заполненность буфферов	m_sign_full_buff1 – Флаг заполненности первого буфера m_sign_full_buff2 – Флаг заполненности второго буфера
4	ReadSizeFile	Считывает размер файла	m_uFileSize – Содержит информацию о размере файла
5	FillBuffer	Заполняет буфер данными из файла	cNameFile – Название файла myFile – Файл p_c_buff – Буффер, хранящий данные из файла
6	ReadFromBuff	Считывает информацию в буфферы	m_sign_full_buff1 – Флаг заполненности первого буфера m_sign_full_buff2 – Флаг заполненности второго буфера m_Buff1 – Первый буффер m_Buff2 – Второй буффер

Класс «FileData» служит для получения содержимого файла и связан с «Buffer». Данный класс не обладает собственными функциями, так как все его компоненты использует класс «Buffer». Компонентами, которыми обладает класс, приведены в таблице 11.

Таблица 11 – Компоненты класса «thread_guard»

№	Компонент	Описание	Аргументы
1	thread_guard	Поток для экстренного завершения остальных потоков	

Кроме операций следует рассмотреть и атрибуты, объявленные в классах.

Атрибуты класса «CDiplomDLG» представлены в таблице 12.

Таблица 12 – Атрибуты класса «CDiplomDLG»

№	Атрибут	Описание
1	CMB_ValueIPSCO	В этом атрибуте хранятся данные об IP адресе сервера, введенным на форме
2	TB_IPSCO	В этом атрибуте хранятся данные об IP адресе клиента, введенным на форме
3	ED_VAL_Info	В этом атрибуте хранятся данные, выводимые в окно с информацией
4	sem	Семафор для контроля за потоками
5	pThreadTmp	Хранит указатель на диалоговое окно
6	m_sDirSample	Хранит путь к корневой папке аудиофайла
7	m_hicon	Дескриптор для значка GDI
8	V_iOutCommand	Хранит команды запросов
9	V_sSample	Служит для хранения информации состояния узлов
10	lstSample	Хранит информацию об индексе выбранного аудиофайла

Атрибуты класса «CSessionTCP» представлены в таблице 13.

Таблица 13 – Атрибуты класса «CSessionTCP»

№	Атрибут	Описание
1	map_tcp	Буффер для tcp
2	mutex_tcp	Мьютекс для работы с потоками
3	socket_server	Сокет сервера
4	socket_client	Сокет клиента
5	sem_tcp	Семафор для контроля за потоками
6	m_iPortTCP	Хранит порт tcp
7	threadReadMess	Поток для чтения запросов

Продолжение таблицы 13

№	Атрибут	Описание
8	my_thread_guard_read_tcp_socket	Атрибут для экстренного завершения потоков
9	TCPFileName	Хранит путь к корневой папке аудиофайла

Атрибуты класса «SharedMemory» представлены в таблице 14.

Таблица 14 – Атрибуты класса «SharedMemory»

№	Атрибут	Описание
1	mutex_SharedMem	Мьютекс для работы с потоками
2	m_SharedMemory	Буфер для временной памяти
3	IntoStat	Структура с буффером

Атрибуты класса «CSessionRTP» представлены в таблице 15.

Таблица 15 – Атрибуты класса «CSessionRTP»

№	Атрибут	Описание
1	m_sIP	Хранит информацию IP адреса клиента
2	m_IPortRTP	Хранит порт
3	socket_rtp	Хранит сокет сервера
4	o_Buffer	Буфер для rtp
5	thread_write_rtp_socket	Поток для отправки данных с аудиофайла
6	my_thread_guard_write_rtp_socket	Атрибут для экстренного завершения потока

Атрибуты класса «Buffer» представлены в таблице 16.

Таблица 16 – Атрибуты класса «Buffer»

№	Атрибут	Описание
1	my_thread_guard_check_buff	Атрибут для экстренного завершения потока
2	m_Buff1	Первый буфер для данных из аудиофайла
3	m_Buff2	Второй буфер для данных из аудиофайла
4	m_bBuffer_allocated	Флаг, что буфер выделен
5	m_sign_full_buff1	Хранит информацию о заполненности первого буфера
6	m_sign_full_buff2	Хранит информацию о заполненности второго буфера
7	m_num_read_buff	Хранит номер буфера, в который происходит чтение данных
8	m_size_rec	Хранит текущий объем данных к передаче в тракт
9	m_size_transferred_data	Хранит объем переданных данных
10	m_sizeBuf	Хранит размер буфера

Продолжение таблицы 16

№	Атрибут	Описание
11	FlagFileRead	Флаг чтения в буфферы

Атрибуты класса «FileData» приведены в таблице 17.

Таблица 17 – Атрибуты класса «FileData»

№	Атрибут	Описание
1	m_fullNameFile	Хранит путь к корневой папке аудиофайла
2	m_uSeekPos	Хранит сколько данных считали
3	m_uPosBuf	Хранит размер буфера при передаче
4	m_uFileSize	Хранит размер файла
5	m_uall_transferred_data	Хранит размер буфера относительно файла

Атрибуты класса «thread_guard» приведены в таблице 18.

Таблица 18 – Атрибуты класса «thread_guard»

№	Атрибут	Описание
1	t	Поток для завершения остальных потоков

Таким образом были выделены основные операции, атрибуты классов и их взаимодействие.

4 Реализация информационной системы

Разработка информационной системы производилась с помощью Visual Studio 2022 с поддержкой библиотеки MFC. При открытии приложения на экране возникает интуитивно понятный интерфейс, представленный на рисунке 8 с доступными полями ввода IP адресов и кнопок «Инициализация», «Проверить наличие запросов».

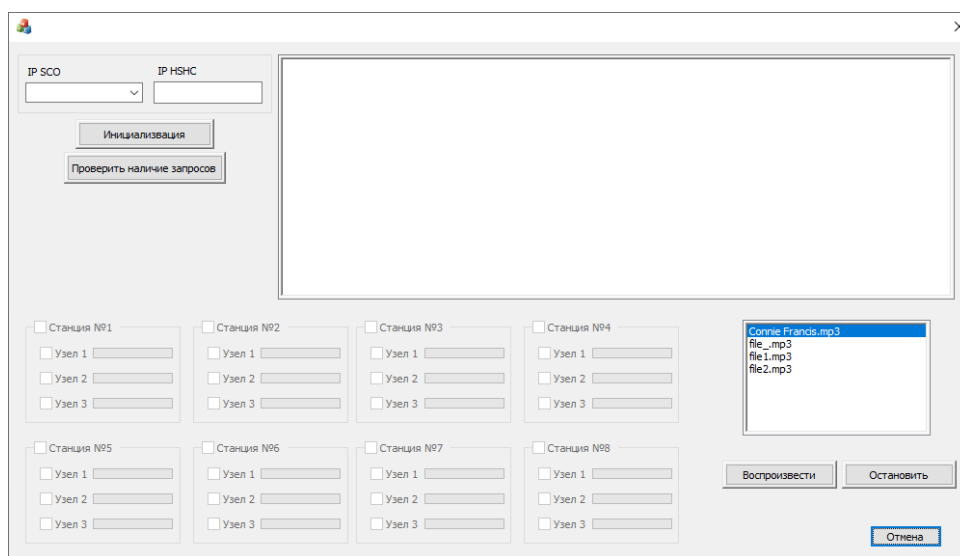


Рисунок 8 – Первоначальный интерфейс

Сначала пользователю необходимо ввести IP адреса клиента и сервера. IP сервера ищутся автоматически и записываются в списке, представленным на рисунке 9.

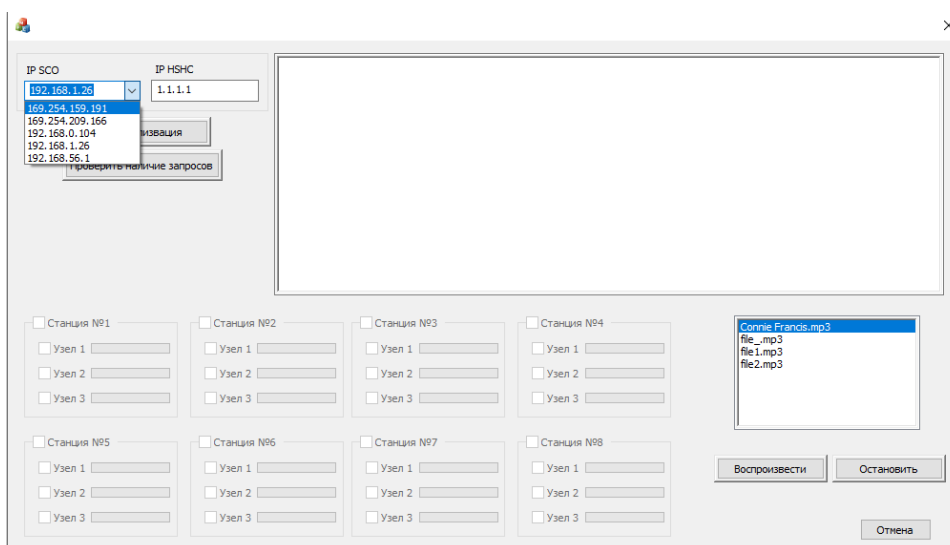


Рисунок 9 – Список IP адресов

После ввода адресов, при нажатии на кнопку «Инициализация» приложение проверит правильность введенных данных и создаст сокеты RTP и TCP. Если IP адреса введены неверно, выведется ошибка, если адреса введены корректно, проверка пройдет успешно, как показано на рисунке 10 и 11.

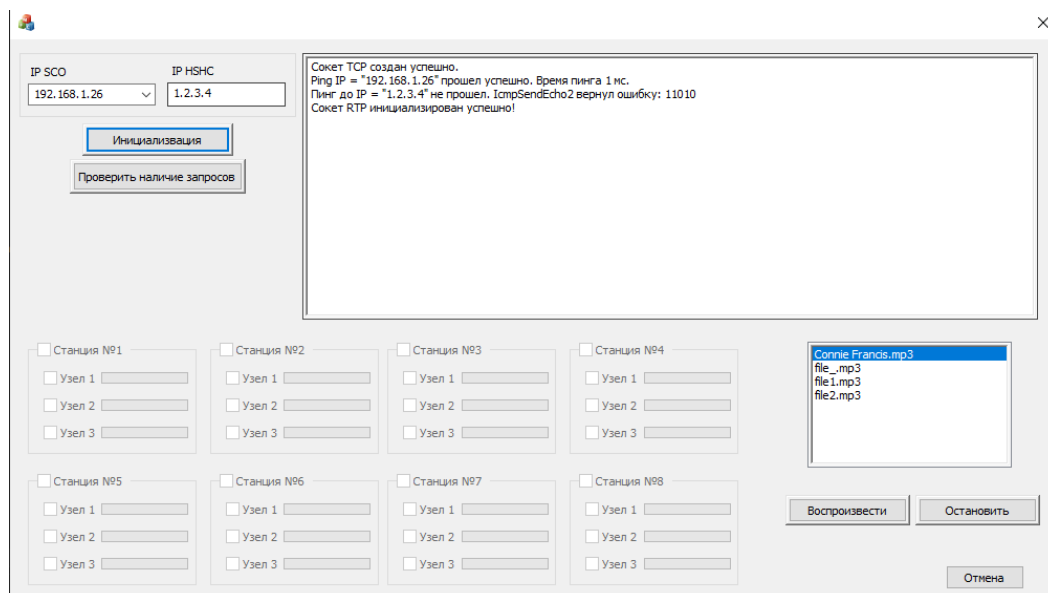


Рисунок 10 – Некорректный ввод IP адресов

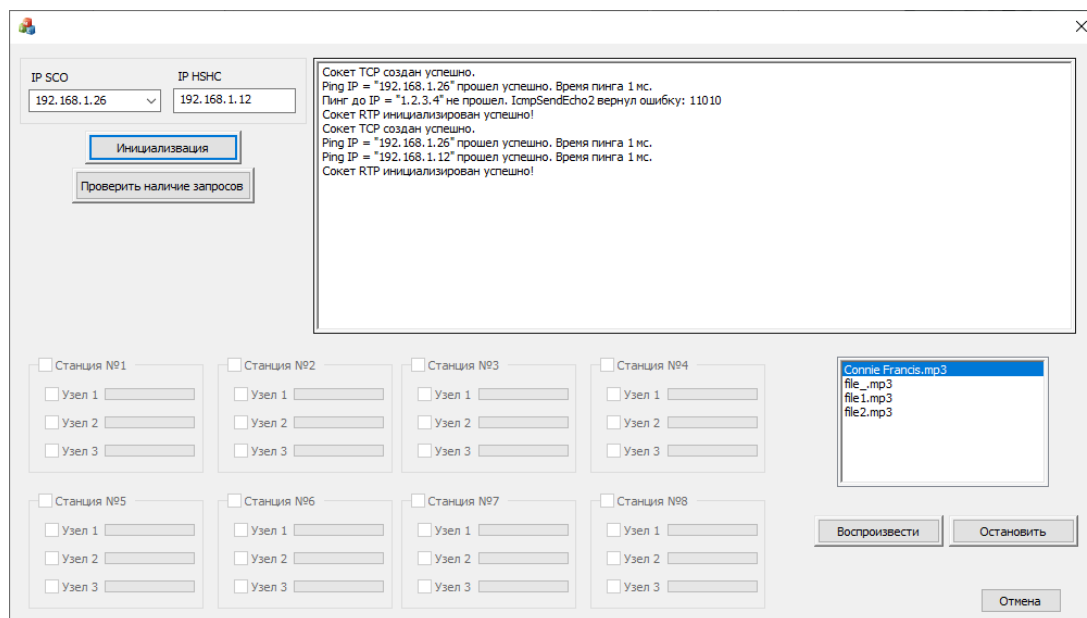


Рисунок 11 – Корректный ввод IP адресов

Следующим шагом пользователю необходимо проверить наличие запросов на подключение нажатием кнопки «Проверить наличие запросов». Если запросы отсутствуют, программа останется в режиме ожидания. Если

присутствуют, узлы, готовые к принятию аудиопотока, станут доступные для выбора, как показано на рисунке 12.

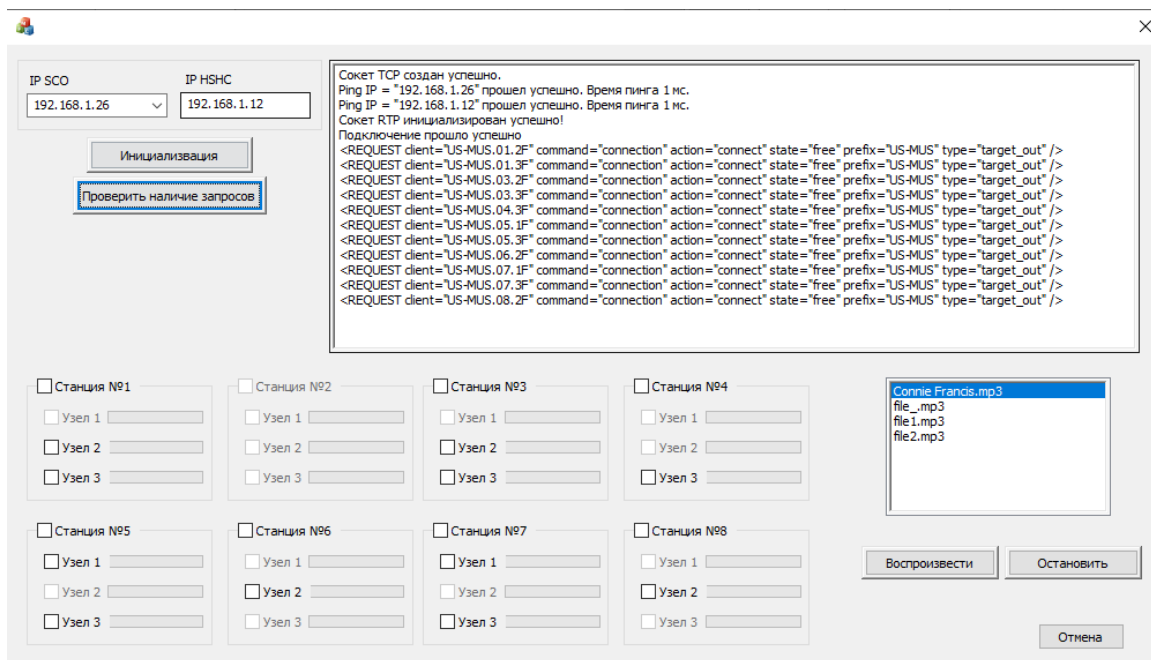


Рисунок 12 – Разблокировка узлов

Пользователю необходимо выбрать узлы, на которые пойдет передача потока и аудиофайл из списка, представленного справа. Процесс изображен на рисунке 13.

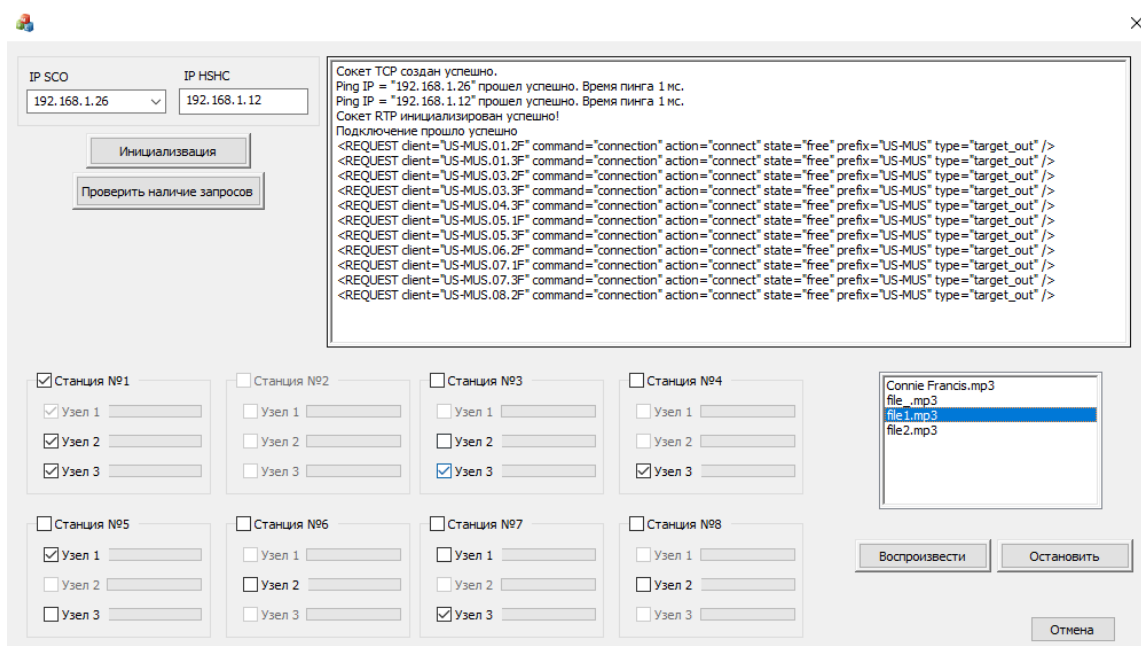


Рисунок 13 – Выбор узлов и аудиофайла

Конечным результатом будет нажатие кнопки «Воспроизвести», после чего RTP поток начнет передачу файла на выбранные узлы. Процесс будет отображаться в виде шкалы загрузки, находящейся справа от названия узла, как показано на рисунке 14.

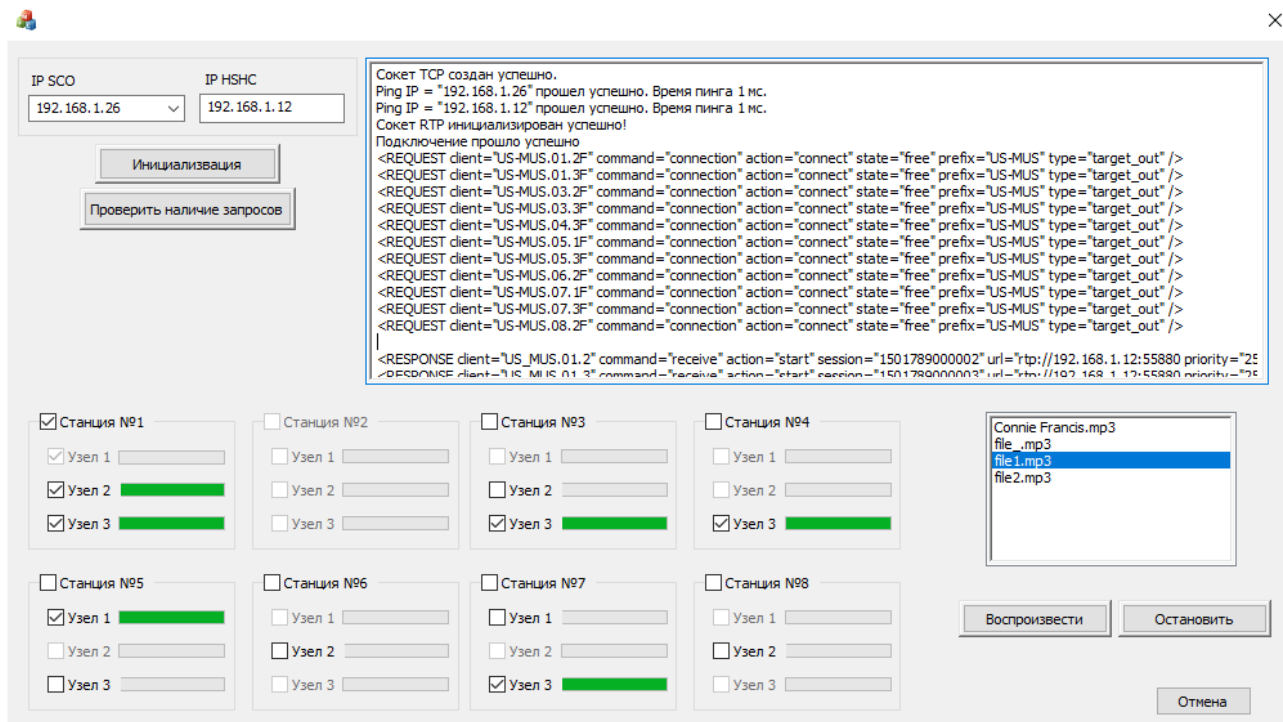


Рисунок 14 – Передача аудиофайла

Таким образом была продемонстрирована работа программного имитатора системы централизованного оповещения.

5 Тестирование информационной системы

Для тестирования работоспособности разрабатываемого приложения будут применены следующие методики тестирования программного обеспечения: функциональное тестирование черным ящиком, модульное тестирование и анализ метрик кода [12].

5.1 Функциональное тестирование

После анализа требований и вариантов использования были выделены самые часто возникающие сценарии использования системы управления разработкой программного обеспечения [13]. Сценарии необходимо покрыть тестами для обеспечения максимальной надежности работы и минимизации количества ошибок на этапе кодирования:

- загрузка страницы;
- создание TCP сокета с корректными IP адресами;
- создание TCP сокета с некорректными IP адресами;
- пинг корректных IP адресов;
- пинг некорректных IP адресов;
- создание RTP сокета с корректными IP адресами;
- создание RTP сокета с некорректными IP адресами;
- установка соединения с другим компьютером;
- отправка файла по RTP потоку.

На основе приведенных выше сценариев был составлен план тестирования, представленный в таблице 19.

Таблица 19 – Функциональное тестирование работы системы

№	Названия тестового случая	Тест	Результат
1	Загрузка страницы	Будет отображена запрашиваемая страница	Страница успешно загружена

Продолжение таблицы 19

№	Названия тестового случая	Тест	Результат
2	Создание TCP сокета с корректными IP адресами	Создастся сокет и выведется сообщение об успешном создании	Сокет создан
3	Создание TCP сокета с некорректными IP адресами	Выведется сообщение об ошибке создания сокета	Вывод ошибки
4	Пинг корректных IP адресов	Вывод сообщения об успешной операции	Пинг прошел успешно
5	Пинг некорректных IP адресов	Вывод сообщения об ошибке	Вывод ошибки
6	Создание RTP сокета с корректными IP адресами	Вывод сообщения об успешной операции	Сокет создан
7	Создание RTP сокета с некорректными IP адресами	Вывод сообщения об ошибке	Вывод ошибки
8	Установка соединения с другим компьютером	После отправки запроса, получение номеров станций, готовых к принятию потока	Станции разблокированы
9	Отправка файла по RTP потоку	Отправка файла другому компьютеру	Сообщение об успешной отправке файла

Для введения даты проведения пары для тестового случая № 5 было проведено тестирование методом выделения классов эквивалентности

Выделенные классы приведены в таблице 20.

Таблица 20 – Выделенные классы

	Правильные КЭ	Неправильные КЭ
Длина поля	7–15(1)	0–6(2), 15+(3)
Структура поля	XXX.XXX.XXX.XXX	другие (5)
Содержание	[1–224].[1–224].[1–224].[1–224] (6), (7), (8),(9)	Другие (10)

Результаты приведены в таблице 21.

Таблица 21 – Результаты тестирования

№	КЭ	Входные данные	Выходные данные
1	1, 4, 6, 7, 8	193.168.1.12	+
2	2, 5, 6, 7, 9,10	2.1.1.	-
3	3, 5, 6, 7, 8,10	1.23.12.19999211212	-
4	1, 5, 6, 7, 8,9	2211219991...	-
5	1, 4, 7, 8, 9,19	0.1.1.9	-
6	1, 4, 6, 8, 9,10	1.0.13.1	-
7	1, 4, 6, 7, 9,10	1.1.0.1	-
8	1,4,6,7,8,10	1.1.1.0	-

По результатам тестирования программа показала корректную работоспособность системы.

5.2 Модульное тестирование

Метрика Холстеда – это метод измерения сложности программного кода. Так как приложение состоит из клиентской части и серверной части, то и тестирование каждой части нужно проводить, учитывая особенности каждой технологии. Серверная часть разрабатывается на языке C++ с использованием фреймворка для приложений .NET Core и фреймворка для работы с базами данных Entity Framework.

Рассмотрим подробнее инструменты тестирования, которые будут применяться для серверной части.

Модульное тестирование, иногда блочное тестирование или юнит-тестирование – процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки [14].

Для тестирования серверной части приложения применяются средства тестирования, встроенные в интегрированную среду разработки Visual Studio.

На основе вариантов использования для модуля работы с расписанием были составлены Unit-тесты. Пример теста PingIP представлен в листинге 1.

Листинг 1 – Тесты модуля PingIp

```
CString CDiplomDlg::PingIP(CString sIP)
{
    HANDLE hIcmpFile;
    DWORD dwRetVal = 0;
    char SendData[32] = "Data Buffer";
    LPVOID ReplyBuffer = NULL;
    DWORD ReplySize = 0;
    // unsigned long ipaddr = INADDR_NONE;
    CString ls_Res = L"", l_sIP = sIP;
    int iLenIP = 0;
    char buff_ip[16];
    iLenIP = l_sIP.GetLength();
    memset(buff_ip, 0, sizeof(buff_ip));
    StrToChar(&l_sIP, buff_ip, iLenIP);
    SOCKADDR_IN RecvAddr;
    InetPton(AF_INET, l_sIP, &RecvAddr.sin_addr.s_addr);
    hIcmpFile = IcmpCreateFile();

    if (hIcmpFile == INVALID_HANDLE_VALUE)
        ls_Res = L"Ошибка открытия дескриптора эха IPv4.\r\n";

    else
    {
        ReplySize = sizeof(ICMP_ECHO_REPLY) + sizeof(SendData);
        ReplyBuffer = (VOID*)malloc(ReplySize);
        dwRetVal = IcmpSendEcho2(hIcmpFile, NULL, NULL, NULL,
            RecvAddr.sin_addr.s_addr, SendData, sizeof(SendData), NULL, ReplyBuffer,
            ReplySize, 1000);

        if (dwRetVal != 0)
            ls_Res.Format(L"Ping IP = \"%s\" прошел успешно.
                Время пинга %ld мс. \r\n", l_sIP, dwRetVal,
                    ((PICMP_ECHO_REPLY)ReplyBuffer)-
                    >RoundTripTime);
    }
}
```

```

else
    ls_Res.Format(L"Пинг до IP = \"%s\" не прошел.
IcmpSendEcho2 вернул ошибку: %ld \r\n", l_sIP, GetLastError());

    if (ReplyBuffer != NULL)
    {
        free(ReplyBuffer);
        ReplyBuffer = NULL;
    }

    IcmpCloseHandle(hIcmpFile);
}
return ls_Res;
}

```

Таким же образом было запущено автоматизированное модульное тестирование для всех модулей, результаты которого представлены на рисунке 15.

Test run finished: 189 Tests (189 Passed, 0 Failed, 0 Skipped) run in 3.1 sec

Test	Duration	Traits	Error Message
✔ UnitTestOfTimetableOfClasses (189)	1.9 sec		
▶ ✔ UnitTestOfTimetableOfClasses (2)	390 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CAcadem...	346 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CAcadem...	20 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CAcadem...	17 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CAcadem...	510 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CAuditor....	8 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CDirectio...	41 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CDiscipli...	6 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CEnclosu...	13 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CEnclosu...	110 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CEnclosu...	10 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CGroup....	25 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CGroup....	103 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CGroup....	72 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CInstitute...	14 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CInstitute...	45 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CInstitute...	5 ms		
▶ ✔ UnitTestOfTimetableOfClasses.UT_CTeacher...	2 ms		

Рисунок 15 – Результат выполнения юнит-тестов

Таким образом все методы класса оказались покрыты тестами. Функциональные требования выполнены, путем подтверждения отсутствия ошибок после проведения тестов.

Тестирование клиентской части полностью проводится вручную, поэтому использование встроенных средств тестирования не требуется.

5.3 Анализ кода

Метрика SLOC

Source Lines of Code (SLOC) или количество строк кода, – метрика оценки ПО, подсчитывающая физические и логические строки кода, а так же комментарии. Проведем подсчет объема функции PingIP, используя данную метрику. Функция приведена в листинге 2.

Листинг 2 – Функция PingIP

```
CString CDiplomDlg::PingIP(CString sIP)
{
    HANDLE hIcmpFile;
    DWORD dwRetVal = 0;
    char SendData[32] = "Data Buffer";
    LPVOID ReplyBuffer = NULL;
    DWORD ReplySize = 0;
    // unsigned long ipaddr = INADDR_NONE;
    CString ls_Res = L"", l_sIP = sIP;
    int iLenIP = 0;
    char buff_ip[16];
    iLenIP = l_sIP.GetLength();
    memset(buff_ip, 0, sizeof(buff_ip));
    StrToChar(&l_sIP, buff_ip, iLenIP);
    SOCKADDR_IN RecvAddr;
    InetPton(AF_INET, l_sIP, &RecvAddr.sin_addr.s_addr);
    hIcmpFile = IcmpCreateFile();
    if (hIcmpFile == INVALID_HANDLE_VALUE)
        ls_Res = L"Ошибка открытия дескриптора эха IPv4.\r\n";
    else
    {
```

```

        ReplySize = sizeof(ICMP_ECHO_REPLY) + sizeof(SendData);
        ReplyBuffer = (VOID*)malloc(ReplySize);
        dwRetVal = IcmpSendEcho2(hIcmpFile, NULL, NULL, NULL,
RecvAddr.sin_addr.s_addr, SendData, sizeof(SendData), NULL, ReplyBuffer,
ReplySize, 1000);

        if (dwRetVal != 0)
            ls_Res.Format(L"Пинг IP = \"%s\" прошел успешно.
Время пинга %ld мс. \r\n", l_sIP, dwRetVal,
((PICMP_ECHO_REPLY)ReplyBuffer)-
>RoundTripTime);
        else
            ls_Res.Format(L"Пинг до IP = \"%s\" не прошел.
IcmpSendEcho2 вернул ошибку: %ld \r\n", l_sIP, GetLastError());
            if (ReplyBuffer != NULL)
            {
                free(ReplyBuffer);
                ReplyBuffer = NULL;
            }
            IcmpCloseHandle(hIcmpFile);
        }
        return ls_Res;
    }

```

Физические строки кода (PLOC) – 58 строк:

– функция PingIP: 58 строк.

Логические строки кода (LLOC) – 53 строки:

– функция PingIP: 53 строки.

Комментарии – 2 строки:

– функция PingIP: 2 строки.

Итого, получаем, что SLOC составляет 71.

Метрики Холстеда

Метрика Холстеда относится к метрикам, вычисляемым на основании анализа числа строк и синтаксических элементов исходного кода программы. В основе лежит следующая концепция – программы состоят из операторов и

операндов. Операнды – переменные или константы. Операторы – комбинации символов, влияющие на значение и порядок операндов [15].

Для расчета необходимо просчитать следующие значения:

- число различных операторов данной реализации – $n1$;
- число различных операндов модуля – $n2$;
- общее число всех операторов – $N1$;
- общее число всех операндов – $N2$;
- число различных входных и выходных операндов – $n2^*$;
- величина словаря – $n = n1 + n2$;
- длина модуля – $N = N1 + N2$.

Операторы и операнды приведены в таблицах 21 и 22 соответственно.

Таблица 22 – Операторы

Операторы	Число вхождений
=	20
!	2
->	1
if	3
else	2
sizeof	4
inetpton	1
IcmpCreateFile	1
memset	1
GetLength	1
IcmpCloseHandle	1

Продолжение таблицы 22

Операторы	Число вхождений
malloc	1
;	24
,	21
()	21
+	1

Таблица 23 – Операнды

Операнды	Число вхождений
hIcmpFile	5
dwRetVal	4
dwRetVal[32]	4
ReplyBuffer	7
ReplySize	4
ReplySize	5
l_sIP	6
iLenIP	3
buff_ip[16]	4
RecvAddr	3
HANDLE	1
DWORD	2

Продолжение таблицы 23

Операнды	Число вхождений
SOCKADDR_IN	1
n2 = 16	N2 = 55
char	2
cstring	3
int	1

Длина программы рассчитывается по формуле:

$$N = N1 + N2 = 105 + 55 = 160$$

Словарь программы рассчитывается по формуле:

$$n = n1 + n2 = 16 + 16 = 32$$

Объем программы рассчитывается по формуле:

$$NV = N * \log_2(n) = 160 * 5 = 800$$

Сложность программы рассчитывается по формуле:

$$ND = (n1 / 2) * (N2 / n2) = (16 / 2) * (55 / 16) = 27.5$$

Уровень программы имеет значение ниже среднего по данным измерений из книги Холстеда.

Цикломатическая сложность

Граф потока управления (Control Flow Graph, CFG) – это граф, где узлы представляют базовые блоки кода, а ребра представляют переходы между ними. В статическом анализе кода, CFG может быть использован для обнаружения потенциальных проблем в коде. Граф потока управления программы для рассматриваемой функции приведен на рисунке 16.

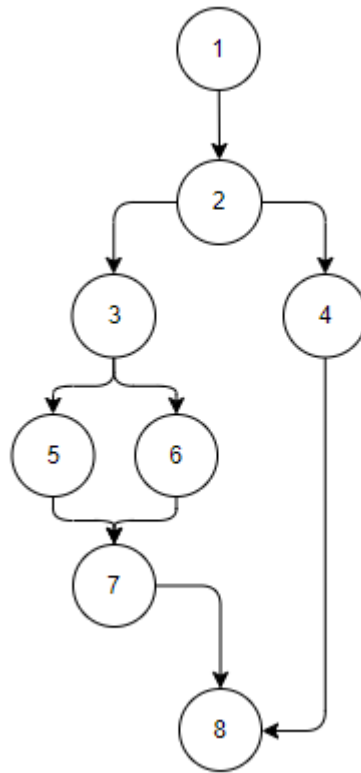


Рисунок 16 – Граф потока управления программы

На основе этого графа вычисляется цикломатическая сложность программы по формуле.

$$V(G) = e - n + 2p$$

где e – количество дуг;

n – количество вершин;

p – число компонент связности.

Граф содержит 8 вершин, 9 дуг и один компонент связности.

$$V(G) = 9 - 8 + 2 = 3$$

Таким образом значение цикломатической сложности равно 3.

Это значение не превышает 10, следовательно, функция не является структурно сложной.

Заключение

В результате выполнения выпускной квалификационной работы был разработан программный имитатор системы централизованного оповещения в информационной системе метрополитена в среде разработки Visual Studio 2022.

В процессе выполнения работы было проведено детальное исследование предметной области. Путем анализа были выявлены функциональные возможности, помогавшие определить основные требования к разрабатываемой системе. На основании требований были выбраны средства для проектирования и разработки, представляющие наилучшую комбинацию функциональности и производительности.

Кроме функциональных возможностей, было проведено детальное планирование разработки, учитывающее этапы работы, и составлен бюджет на разработку информационной системы.

Проведено проектирование информационной системы с использованием Visual Paradigm, которое включало построение диаграммы бизнес-процессов и диаграмм UML: диаграмма состояний, диаграмма деятельности, диаграмма вариантов использования, диаграмма классов.

Была проведена разработка системы с использованием средств среды разработки Visual Studio 2022.

После завершения разработки системы было проведено комплексное тестирование, включающее функциональное тестирование, модульное тестирование и тестирование и анализ метрик кода, как метрика SLOC, метрика Холстеда и цикломатическая сложность. Результаты тестирования показали, что разработанный программный имитатор работает корректно и соответствует заявленным требованиям.

В результате разработки программного имитатора системы централизованного оповещения в информационной системе метрополитена были успешно достигнуты поставленные цели, включая проектирование, реализацию и тестирование программной информационной системы.

Список использованных источников

1. Автоматизированная система централизованного оповещения [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Автоматизированная_система_централизованного_оповещения (Дата обращения 29.05.2024)
2. Системы оповещения в метрополитене. Практическое решение [Электронный ресурс]. URL: https://www.escortpro.ru/page/article/article114_metro_roxton.htm (Дата обращения 29.05.2024)
3. 1С: Автоматизированная система централизованного оповещения [Электронный ресурс]. URL: <https://fireman.club/inseklodepia/avtomatizirovannaya-sistema-tsentralizovannogo-opoveshheniya> (Дата обращения 29.05.2024)
4. Система оповещения в метро [Электронный ресурс]. URL: <https://alfatechnologies.ru/complex-solutions/gorodskaya-infrastruktura/transportnaya-infrastruktura/sistema-opoveshcheniya-v-metro> (Дата обращения 29.05.2024)
5. Что такое Visual Studio? [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2022> (дата обращения 29.05.2024).
6. Что такое диаграмма Ганта [Электронный ресурс]. URL: <https://www.atlassian.com/ru/agile/project-management/gantt-chart> (дата обращения 29.05.2024).
7. Диаграмма Ганта и способы ее использования // GeekBrains [Электронный ресурс]. URL: <https://gb.ru/blog/diagramma-ganta/> (дата обращения 29.05.2024).
8. Сколько зарабатывают программисты в России [Электронный ресурс]. URL: <https://www.sravni.ru/kursy/info/skolko-zarabatyvaet-programmist-v-rossii-i-mire/> (дата обращения 29.05.2024).

9. Простое руководство по диаграммам активности UML [Электронный ресурс]. URL: <https://creatly.com/blog/ru/uncategorized-ru/> (дата обращения 29.05.2024).

10. Использование диаграммы вариантов использования UML при проектировании программного обеспечения // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/566218/> (дата обращения 29.05.2024).

11. Построение диаграммы классов [Электронный ресурс]. URL: https://flexberry.github.io/ru/gpg_class-diagram.html (дата обращения 29.05.2024).

12. Какие бывают этапы и виды тестирования: подробный разбор [Электронный ресурс]. URL: <https://ru.hexlet.io/blog/posts/vidy-testirovaniya> (дата обращения 29.05.2024).

13. Функциональное тестирование ПО: задачи, виды, методы проведения [Электронный ресурс]. URL: <https://practicum.yandex.ru/blog/funkcionalnoe-testirovanie-po/> (дата обращения 29.05.2024).

14. Модульное тестирование: что это? Типы, инструменты [Электронный ресурс]. URL: https://logrocon.ru/news/unit_testing (дата обращения 29.05.2024).

15. Статистический анализ кода [Электронный ресурс]. URL: <https://pvs-studio.ru/ru/blog/terms/0046/> (дата обращения 29.05.2024).

Приложение А
(справочное)
Глоссарий

СЦО – система централизованного оповещения.

RTP – протокол, определяющий стандарт пакетов для передачи мультимедиа-данных через интернет (Real-Time Transport Protocol).

TCP – один из основных протоколов передачи данных интернета (Transmission Control Protocol).

UML (Unified Modeling Language) – это язык графического моделирования, используемый для создания, визуализации, документирования и анализа системных проектов.

Информационная система (ИС) – это совокупность взаимосвязанных компонентов (аппаратных, программных, информационных и организационных), которые используются для получения, обработки, хранения, передачи и использования информации.

Приложение В
(рекомендуемое)
Листинг программы

Листинг В.1 – Модуль Buffer.h

```
#pragma once

#include "thread_guard.h"
#include "Common.h"

using namespace SCO_COMMON;

class FileData
{
protected:

    std::string m_FullNameFile;

    u_long m_uSeekPos;

    u_long m_uPosBuf;
    //Размер буфера при передаче

    u_long m_uFileSize;
    //Размер файла

    u_long m_uall_transfered_data;
    //Размер буфера относительно файла

public:

    FileData() :m_FullNameFile(""), m_uSeekPos(0){}

    ~FileData() {}

    char readFromFile();

    void SetNameFile(std::string p_FullNameFile) { m_FullNameFile =
p_FullNameFile; }

};

class Buffer : public FileData
{
private:

    std::thread* thread_check_buff; // Поток
    проверки заполненности буферов

    thread_guard* my_thread_guard_check_buff;

private:

    char* m_Buff1;

    char* m_Buff2;
```

```

        bool                m_bBuffer_allocated;                ///< флаг того,
что буфер выделен

        bool                m_sign_full_buff1;                ///< признак
заполненности первого буфера

        bool                m_sign_full_buff2;                ///< признак
заполненности второго буфера

        u_short            m_num_read_buff;                ///< номер буфера,
из которого производится чтение данных

        u_int               m_size_rec;                ///< текущий
объем данных к передаче в тракт

        u_long              m_size_transferred_data;                ///< объем
переданных данных

        size_t              m_sizeBuf;                ///< размер буфера

private:

        bool semaphore_availability_data_buffer()                // Проверяет
есть ли данные хотя бы в одном из буферов

        {

                if (m_sign_full_buff1 || m_sign_full_buff2)

                        return true;

                else

                        return false;

        }

public:

        Buffer();

        ~Buffer();

        bool FlagFileRead;

        BOOL StartThreadCheckBuffer();

        void CheckStateBuffer();

        u_long ReadSizeFile() { return m_uFileSize; };

        int FillBuffer(char* p_cBuff);

        u_int readFromBuf(char* p_data, size_t p_size_receive_data);

};

```

Листинг В.2 – Модуль Common.h

```
#pragma once
```

[illegible]

```

        RTP_SEND
    = 1000,                                     ///
Указатель на отправляемые rtp данные
        DATA_TRANSFER_REQUIRED,
        CHECK_REQUEST_CLIENT,
        TCP_BUF_SIZE
    = 8192,
};
enum
{
    TCP_SOCKET_CREATE_SUCCESFULLY                = 0,
    SOCKET_INITIALIZATION_ERROR,
    ERROR_TRANSFER,
    ERROR_CONNECTONG_SOCKET_TO_SERVER,
};
enum
{
    CONNECTION_REQUEST
    = 10,
    TCP_SOCKET_CONNECT
};
}
u_short StrToChar(CString* str, char* cBuf, int iSizeBuf);
std::string to_std_string(CString const& src);

```

Листинг В.3 – Модуль defines.h

```

#ifndef DEFINES_H
#define DEFINES_H

#define CONNECTION_WAITING_TIME                20                // Время
ожидания установления соединения

#define WM_MSGTHREAD_ALL                      100
#define INPUT_FORM                            101

/*****
**/
/*****      Константы направления RTP потока (передача/прием)
***/
/*****      TRANSFER_MODE и RECEIVE_MODE являются индексами массива
***/
/*****      и тем самым определяют тип RTP объекта (прием/передача)
***/
/*****
**/
#define TRANSFER_MODE                        0
#define RECEIVE_MODE                        1

#endif

```

Листинг В.4 – Модуль Diplom.h

```

// Diplom.h: главный файл заголовка для приложения PROJECT_NAME

```

```

//

#pragma once

#ifndef __AFXWIN_H__
    #error "включить pch.h до включения этого файла в PCH"
#endif

#include "resource.h"           // основные символы


// CDiplomApp:
// Сведения о реализации этого класса: Diplom.cpp
//

class CDiplomApp : public CWinApp
{
public:
    CDiplomApp();

    // Переопределение
public:
    virtual BOOL InitInstance();

    // Реализация

    DECLARE_MESSAGE_MAP()
};

extern CDiplomApp theApp;

```

Листинг В.6 – Модуль DiplomDLG.h

```

// DiplomDlg.h: файл заголовка
//
#include <vector>
#include <string>

#include "Common.h"

using namespace std;
using namespace SCO_COMMON;

#pragma once

// Диалоговое окно CDiplomDlg
class CDiplomDlg : public CDialogEx
{
private:
    CWinThread* pThreadTmp;

    CString      m_sDirSample;

```

```

        // Создание
public:
    CDiplomDlg(CWnd* pParent = nullptr);        // стандартный конструктор
    void CDiplomDlg::GetIpAdd(vector<string>& MyIp);
    // Данные диалогового окна
#ifdef AFX_DESIGN_TIME
    enum { IDD = IDD_DIPLOM_DIALOG };
#endif

protected:
    virtual void DoDataExchange(CDataExchange* pDX);        // поддержка DDX/DDV

// Реализация
protected:
    HICON m_hIcon;

    BOOL v_iOutCommand;

    // Созданные функции схемы сообщений
    virtual BOOL OnInitDialog();
    afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
    afx_msg void OnPaint();
    afx_msg HCURSOR OnQueryDragIcon();

    DECLARE_MESSAGE_MAP()
public:
    void Init(CString sIP_SCO, CString sIP_SHSC);

    CEdit TB_IPSCO_Element;
    CEdit ED_EL_Info;

    CComboBox CB_EnterIp;

    CString CMB_ValueIPSCO;
    CString TB_IPSCO;
    CString ED_VAL_Info;

    void FormationListSamples();

    CSemaphore sem;

    BOOL *CB_Station_Value[NUMBER_STATION];
    BOOL *CB_Station_Node[NUMBER_STATION][NUMBER_NODE_STATION];

    CProgressCtrl* PB_Station[NUMBER_STATION][NUMBER_NODE_STATION];

    afx_msg void OnCbnKillfocusCbIp();
    afx_msg void OnInit();
    afx_msg void OnConnect();
    afx_msg void OnBnClickedCbStation();
    afx_msg void OnCheckNode();

```

```

afx_msg void OnEnKillfocusEdit2();
afx_msg void Reproducing_click();
afx_msg void OnSelectSample();

CString PingIP(CString sIP);

CString GetIP_Server() { return CMB_ValueIPSCO; }
CString GetIP_Client() { return TB_IPSCO; }

LRESULT GetSharMess(WPARAM wParam, LPARAM lParam);

LRESULT ReadMess(WPARAM wParam, LPARAM lParam);

void CheckStation(string Mess);
void AnalysisReadMsg(string& p_sMsg);
void ProgressBarFilling(int Progress);

string GetElementaryPartTeam(string& p_sMsg, string p_sElement);

CButton CB_Station1;
CButton CB_Station2;
CButton CB_Station3;
CButton CB_Station4;
CButton CB_Station5;
CButton CB_Station6;
CButton CB_Station7;
CButton CB_Station8;

CButton CB_Station1_1;
CButton CB_Station1_2;
CButton CB_Station1_3;

CButton CB_Station2_1;
CButton CB_Station2_2;
CButton CB_Station2_3;

CButton CB_Station3_1;
CButton CB_Station3_2;
CButton CB_Station3_3;

CButton CB_Station4_1;
CButton CB_Station4_2;
CButton CB_Station4_3;

CButton CB_Station5_1;
CButton CB_Station5_2;
CButton CB_Station5_3;

CButton CB_Station6_1;
CButton CB_Station6_2;
CButton CB_Station6_3;

CButton CB_Station7_1;

```

```

CButton CB_Station7_2;
CButton CB_Station7_3;

CButton CB_Station8_1;
CButton CB_Station8_2;
CButton CB_Station8_3;

BOOL CB_Station1_Value, CB_Station2_Value, CB_Station3_Value,
CB_Station4_Value, CB_Station5_Value, CB_Station6_Value, CB_Station7_Value,
CB_Station8_Value;

BOOL CB_Station1_1_Value, CB_Station1_2_Value, CB_Station1_3_Value;
BOOL CB_Station2_1_Value, CB_Station2_2_Value, CB_Station2_3_Value;
BOOL CB_Station3_1_Value, CB_Station3_2_Value, CB_Station3_3_Value;
BOOL CB_Station4_1_Value, CB_Station4_2_Value, CB_Station4_3_Value;
BOOL CB_Station5_1_Value, CB_Station5_2_Value, CB_Station5_3_Value;
BOOL CB_Station6_1_Value, CB_Station6_2_Value, CB_Station6_3_Value;
BOOL CB_Station7_1_Value, CB_Station7_2_Value, CB_Station7_3_Value;
BOOL CB_Station8_1_Value, CB_Station8_2_Value, CB_Station8_3_Value;

CProgressCtrl PB_Station1_1_Value, PB_Station1_2_Value,
PB_Station1_3_Value;
CProgressCtrl PB_Station2_1_Value, PB_Station2_2_Value,
PB_Station2_3_Value;
CProgressCtrl PB_Station3_1_Value, PB_Station3_2_Value,
PB_Station3_3_Value;
CProgressCtrl PB_Station4_1_Value, PB_Station4_2_Value,
PB_Station4_3_Value;
CProgressCtrl PB_Station5_1_Value, PB_Station5_2_Value,
PB_Station5_3_Value;
CProgressCtrl PB_Station6_1_Value, PB_Station6_2_Value,
PB_Station6_3_Value;
CProgressCtrl PB_Station7_1_Value, PB_Station7_2_Value,
PB_Station7_3_Value;
CProgressCtrl PB_Station8_1_Value, PB_Station8_2_Value,
PB_Station8_3_Value;

CString v_sSample;
CListBox lstSample;
};

```

Листинг В.7 – Модуль framework.h

```

#pragma once

#ifdef VC_EXTRALEAN
#define VC_EXTRALEAN // Исключите редко используемые компоненты из
заголовков Windows
#endif

#include "targetver.h"

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS // некоторые конструкторы
CString будут явными

```

```

// отключает функцию скрывтия некоторых общих и часто пропускаемых предупреждений
MFC
#define _AFX_ALL_WARNINGS

#include <afxwin.h>           // основные и стандартные компоненты MFC
#include <afxext.h>           // Расширения MFC

#include <afxdisp.h>         // классы автоматизации MFC

#ifdef _AFX_NO_OLE_SUPPORT
#include <afxdtctl.h>         // поддержка MFC для типовых элементов
управления Internet Explorer 4
#endif
#ifdef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>           // поддержка MFC для типовых элементов
управления Windows
#endif // _AFX_NO_AFXCMN_SUPPORT

#include <afxcontrolbars.h>   // поддержка MFC для лент и панелей управления

#ifdef _UNICODE
#ifdef _M_IX86
#pragma comment(linker, "/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
processorArchitecture='x86' publicKeyToken='6595b64144ccf1df' language='*'\")
#elif defined _M_X64
#pragma comment(linker, "/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
processorArchitecture='amd64' publicKeyToken='6595b64144ccf1df' language='*'\")
#else
#pragma comment(linker, "/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
processorArchitecture='*' publicKeyToken='6595b64144ccf1df' language='*'\")
#endif
#endif

```

Листинг В.8 – Модуль pch.h

```

// pch.h: это предварительно скомпилированный заголовочный файл.
// Перечисленные ниже файлы компилируются только один раз, что ускоряет
последующие сборки.

```

```
// Это также влияет на работу IntelliSense, включая многие функции просмотра и
// завершения кода.
// Однако изменение любого из приведенных здесь файлов между операциями сборки
// приведет к повторной компиляции всех(!) этих файлов.
// Не добавляйте сюда файлы, которые планируете часто изменять, так как в этом
// случае выигрыша в производительности не будет.
```

```
#ifndef PCH_H
#define PCH_H
```

```
// Добавьте сюда заголовочные файлы для предварительной компиляции
#include "framework.h"
```

```
#endif //PCH_H
```

Листинг В.9 – Модуль Resource.h

```
//{{NO_DEPENDENCIES}}
// Включаемый файл, созданный в Microsoft Visual C++.
// Используется Diplom.rc
//
#define IDM_ABOUTBOX 0x0010
#define IDD_ABOUTBOX 100
#define IDS_ABOUTBOX 101
#define IDD_DIPLOM_DIALOG 102
#define IDR_MAINFRAME 128
#define IDC_BUTTON1 1000
#define IDC_EDIT2 1002
#define IDC_BUTTON2 1003
#define IDC_IP_Find 1006
#define IDC_CB_IP 1009
#define ED_Info 1012
#define IDC_CHECK1 1013
#define IDC_CB_Station1 1013
#define IDC_CB_Node1_1 1014
#define IDC_CB_Node1_2 1015
#define IDC_CB_Node1_3 1016
#define IDC_PROGRESS1 1017
#define IDC_PB_Station1 1017
#define IDC_PB_Station1_1 1017
#define IDC_PROGRESS2 1018
#define IDC_PB_Station1_2 1018
#define IDC_PROGRESS3 1019
#define IDC_PB_Station1_3 1019
#define IDC_CB_Station2 1020
#define IDC_CB_Node2_1 1021
#define IDC_CB_Node2_2 1022
#define IDC_CB_Node2_3 1023
#define IDC_PROGRESS4 1024
#define IDC_PB_Station2_1 1024
#define IDC_PROGRESS5 1025
#define IDC_PB_Station2_2 1025
#define IDC_PROGRESS6 1026
#define IDC_PB_Station2_3 1026
```

```

#define IDC_CB_Station3 1027
#define IDC_CB_Node3_1 1028
#define IDC_CB_Node3_2 1029
#define IDC_CB_Node3_3 1030
#define IDC_PROGRESS7 1031
#define IDC_PB_Station3_1 1031
#define IDC_PROGRESS8 1032
#define IDC_PB_Station3_2 1032
#define IDC_PROGRESS9 1033
#define IDC_PB_Station3_3 1033
#define IDC_CB_Station4 1034
#define IDC_CB_Node4_1 1035
#define IDC_CB_Node4_2 1036
#define IDC_CB_Node4_3 1037
#define IDC_PROGRESS10 1038
#define IDC_PB_Station4_1 1038
#define IDC_PROGRESS11 1039
#define IDC_PB_Station4_2 1039
#define IDC_PROGRESS12 1040
#define IDC_PB_Station4_3 1040
#define IDC_CB_Station5 1041
#define IDC_CB_Node5_1 1042
#define IDC_CB_Node5_2 1043
#define IDC_CB_Node5_3 1044
#define IDC_PROGRESS13 1045
#define IDC_PB_Station5_1 1045
#define IDC_PROGRESS14 1046
#define IDC_PB_Station5_2 1046
#define IDC_PROGRESS15 1047
#define IDC_PB_Station5_3 1047
#define IDC_CB_Station6 1048
#define IDC_CB_Node6_1 1049
#define IDC_CB_Node6_2 1050
#define IDC_CB_Node6_3 1051
#define IDC_PROGRESS16 1052
#define IDC_PB_Station6_1 1052
#define IDC_PROGRESS17 1053
#define IDC_PB_Station6_2 1053
#define IDC_PROGRESS18 1054
#define IDC_PB_Station6_3 1054
#define IDC_CB_Station7 1055
#define IDC_CB_Node7_1 1056
#define IDC_CB_Node7_2 1057
#define IDC_CB_Node7_3 1058
#define IDC_PROGRESS19 1059
#define IDC_PB_Station7_1 1059
#define IDC_PROGRESS20 1060
#define IDC_PB_Station7_2 1060
#define IDC_PROGRESS21 1061
#define IDC_PB_Station7_3 1061
#define IDC_CB_Station8 1062
#define IDC_CB_Node8_1 1063
#define IDC_CB_Node8_2 1064

```

```

#define IDC_CB_Node8_3                1065
#define IDC_PROGRESS22                1066
#define IDC_PB_Station8_1            1066
#define IDC_PROGRESS23                1067
#define IDC_PB_Station8_2            1067
#define IDC_PROGRESS24                1068
#define IDC_PB_Station8_3            1068
#define IDC_LIST1                     1069
#define IDC_LST_SAMPLE                1069
#define IDC_BT_Reproduce                1070
#define IDC_BT_Stop                   1071

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifdef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE      130
#define _APS_NEXT_COMMAND_VALUE      32771
#define _APS_NEXT_CONTROL_VALUE      1071
#define _APS_NEXT_SYMED_VALUE        101
#endif
#endif

```

Листинг В.10 – Модуль SessionRTP.h

```

#pragma once

#include <thread>

#include "Buffer.h"
#include "thread_guard.h"

extern BOOL g_bFlagWorkThread;

class CSessionRTP
{
private:
    CString m_sIP;
                                     // IP-адрес куда будет передаваться аудиопоток

    u_int m_iPortRTP;
                                     // Номер локального порта
    SOCKET socket_rtp;
                                     // Дескриптор сокета

    Buffer o_Buffer;

    std::thread* thread_write_rtp_socet = NULL;
                                     // Поток чтения из сокета (прослушки)

    thread_guard* my_thread_guard_write_rtp_socet = NULL;

public:

```

```

CSessionRTP() {};
~CSessionRTP() {};

CString GetIP();
SOCKET* GetSocket();
//u_int GetPort(u_short p_iMode);

void SetPort(u_int iPort) { m_iPortRTP = iPort; };

BOOL InitRTPSocket(BOOL bMultiCast = true);

void StartWriteRTP();
void StopWtareRTP() { g_bFlagWorkThread = false; }

void GetFileName(std::string p_FileName) {
o_Buffer.SetNameFile(p_FileName); }

    BOOL StartThreadWriteRTP();
};

```

Листинг В.11 – Модуль SessionTCP.h

```

#pragma once

#include <winsock2.h>
#include <Ws2tcpip.h>

#include <map>
#include <mutex>

#include "Common.h"
#include "thread_guard.h"

// #pragma comment(lib, "Ws2_32.lib")

class CSessionTCP
{
private:

    std::map<clock_t, UINT> map_tcp;

    std::mutex mutex_tcp;

    SOCKET socket_server = NULL;
    SOCKET socket_client = NULL;

    CSemaphore sem_tcp;

    u_short m_iPortTCP;

    std::thread ThreadReadMess;
    thread_guard my_thread_guard_read_tcp_socet;

```

```

        std::string TCPFileName;

private:

        UINT CheckTaskTCP(clock_t iKey);

public:

        CSessionTCP();
        ~CSessionTCP();

        std::string GetFileName(std::string p_FileName) { return TCPFileName =
p_FileName; };

        void TaskManagerTCP();

        void ThreadReadTCP();
        void ThreadwriteTCP(clock_t numMsg, UINT iTypeCommand);

        int InitSocket();

        int OnServer();
        int CheckConnectionRequests();

        void PutTaskTCP(clock_t p_KeyTask, UINT p_idTask);

        void SetPortTCP(u_short pPortTCP);

        void ReproducingRequest(std::string i, std::string j);

};

```

Листинг В.12 – Модуль SharedMemory.h

```

#pragma once

#include <map>
#include <time.h>
#include <vector>
#include <string>
#include <mutex>

struct STMass;

typedef std::map<clock_t, STMass> Shared_Mem_map;

struct STMass
{
public:
        std::vector < std::string > Message;
        int status;

        UINT m_Kei;

```

```
};

class SharedMemory
{
private:

    std::mutex mutex_SharedMem;

    Shared_Mem_map m_SharedMemory;

    STMass IntoStat;

public:
    SharedMemory();
    ~SharedMemory();

    clock_t IntoSharedMemoryInt(int status);
    std::string outSharedMemory(clock_t MessKey, UINT m_key);
    clock_t IntoSharedMemoryString(std::string Mess, UINT m_key);

    void FillPB(int Progress);

protected:

};
```

Листинг В.13 – Модуль StructDef.h

```
#pragma once

struct rtp_header
{
    /* первый байт */
    unsigned short int version : 2; // Версия
    RTP-протокола.
    unsigned short int padding : 1; // Поле
    заполнения (признак наличия октетов-заполнителей в конце поля данных).
    unsigned short int extension : 1; //
    Поле расширения заголовка.
    unsigned short int csrc_len : 4; //
    Количество идентификаторов (CSRC) отправителей.
    /* второй байт */
    unsigned short int marker : 1; // Поле
    маркера (интерпритация маркера определяется профайлом).
    unsigned short int payload_type : 7; //
    Поле типа полезной нагрузки (идентифицирует тип полезной нагрузки и
    //формат
    данных RTP-пакета).
    /* третий-четвертый байты */
```

```

    unsigned short int seq_no;
        // Номер последовательности (порядковый номер пакета).
Занимает 16 бит.
    /* пятый-восьмой байты */
    unsigned int timestamp;
        // Временная метка. Занимает 32 бита.
    /* девятый-двенадцатый байт */
    unsigned int ssrc;
        // Список полей идентификаторов источников, "подмешанных" в
основной поток,

//например, с помощью смесителя.
};

```

Листинг В.14 – Модуль thread_guard.h

```

#pragma once

#include <iostream>
#include <thread>

extern BOOL g_bFlagTCPManager;

// Вариант класса с перемещением потока
class thread_guard
{
    std::thread t;
public:
    explicit thread_guard(std::thread t_) : t(std::move(t_))
    {
        if (!t.joinable())
            throw std::logic_error("No thread!");
    }

    ~thread_guard()
    {
        if (t.joinable())
        {
            // Завершаем диспетчер TCP обмена
            g_bFlagTCPManager = false;

            t.join();
        }
    }
}

```

```

        // пока не разобрался для чего эти две строчки операторов
//          thread_guard( thread_guard const& ) = delete;

//          thread_guard& operator=( thread_guard const& ) = delete;
};

```

Листинг В.15 – Модуль Buffer.cpp

```

#include "pch.h"
#include "SessionRTP.h"
#include "Buffer.h"

#include <fstream>

extern BOOL g_bFlagWorkThread;

char FileData::readFromFile()
{
    std::ifstream out;
    out.open(m_FullNameFile);

    return 0;
}

Buffer::Buffer() : m_Buff1(NULL), m_Buff2(NULL),
                  m_bBuffer_allocated(false),
                  m_sign_full_buff1(false), m_sign_full_buff2(false),

                  thread_check_buff(NULL),
                  my_thread_guard_check_buff(NULL),

                  m_sizeBuf(RTP_BUFFER_SIZE),
                  FlagFileRead(false)
{
    m_Buff1 = new char[m_sizeBuf];
    m_Buff2 = new char[m_sizeBuf];

    m_uPosBuf = 0;
    m_uall_transferred_data = 0;
    m_size_transferred_data = 0;

    m_uFileSize = 0;

```

```

    m_num_read_buff = 1;

    if (m_Buff1)
    {
        if (m_Buff2)
        {
            m_bBuffer_allocated = true;

        }
        else
        {
            delete[] m_Buff1;
            m_Buff1 = NULL;
            m_Buff2 = NULL;
        }
    }
    else if (m_Buff2)
    {
        delete[] m_Buff2;
        m_Buff2 = NULL;
        m_Buff1 = NULL;
    }
    else
    {
        m_Buff2 = NULL;
        m_Buff1 = NULL;
    }
}

Buffer::~~Buffer()
{
    if (m_bBuffer_allocated)
    {
        delete[] m_Buff1;
        delete[] m_Buff2;
    }
}

BOOL Buffer::StartThreadCheckBuffer()
{

```

```

    BOOL bRez = false;

    if (thread_check_buff == NULL && my_thread_guard_check_buff == NULL)
    {
        thread_check_buff = new std::thread(&Buffer::CheckStateBuffer,
this);

        my_thread_guard_check_buff = new
thread_guard(std::move(*thread_check_buff));

        bRez = true;
    }
    return bRez;
}

void Buffer::CheckStateBuffer()
{
    std::chrono::milliseconds timespan(SCO_COMMON::TIME_OUT_CHECK_BUFF);

    while (g_bFlagWorkThread)
    {
        if (!m_sign_full_buff1)
        {
            TRACE("Enter First buffer\n\r");
            switch (FillBuffer(m_Buff1))
            {
                case ERROR_WORK_FILE:
                    break;

                case FILE_READ_SUCCES:

                    m_sign_full_buff1 = true;
                    break;
            }
        }
        if (!m_sign_full_buff2 && m_sign_full_buff1)
        {
            TRACE("Ebter Second Buffer\n\r");
            switch (FillBuffer(m_Buff2))
            {
                case ERROR_WORK_FILE:
                    break;

```

```

        case FILE_READ_SUCCES:

            m_sign_full_buff2 = true;
            break;

        }

    }

    std::this_thread::sleep_for(timespan);
}

}

int Buffer::FillBuffer(char* p_cBuff)
{
    /*
    * Если файл меньше нашего буфера
    *
    * повторный заход в метод при полном прочтении файла
    */
    /*
    std::ifstream out;
    out.open(m_FullNameFile);

    if (out.is_open())
    {
        out.seekg(m_uSeekPos);

        memset(p_cBuff, 0, RTP_BUFFER_SIZE);

        out.read(p_cBuff, RTP_BUFFER_SIZE);

        m_uSeekPos += out.gcount();

        if (feof(out))
        {
            m_uSeekPos = 0;

        }
    }

    */

    CFile myFile;
    CFileException fileException;

```

```

        UINT nAmountReadData;

//    char data_sample[RTP_BUFFER_SIZE];

    CString cNameFile = L"";
    cNameFile = m_FullNameFile.c_str();

/*    std::fstream out(cNameFile);
    if (out.is_open())
    {
        m_uFileSize = out.tellg();
        out.close();
    }*/

    if (myFile.Open(cNameFile, CFile::modeRead, &fileException) == 0)
    {
        return ERROR_WORK_FILE;
    }
    m_uFileSize = myFile.GetLength();

    if (m_uSeekPos != m_uFileSize)
    {
        myFile.Seek(m_uSeekPos, CFile::begin);

        nAmountReadData = myFile.Read(p_cBuff, RTP_BUFFER_SIZE);

        m_uSeekPos += nAmountReadData;

        myFile.Close();
    }

    return FILE_READ_SUCCES;
}

u_int Buffer::readFromBuf(char* data, size_t p_size_receive_data)
{
    /*
    1)проверить какой буффер активный
    2)позиционирование в буффере для считывания следующей порции данных

```

3) анализ конца буфера RTP и допустим в случае чего переход к следующему буферу

```

4)заполнение data
*/
/*
    for (int index = 0; index < p_size_receive_data; index++)
    {
        if (m_sign_full_buff1)
            data[index] = m_Buff1[index];
        if (m_sign_full_buff2)
            data[index] = m_Buff2[index];

        m_uPosBuf++;
        if (m_uPosBuf == RTP_BUFFER_SIZE)
        {
            m_uPosBuf = 0;
            m_sign_full_buff1 = false;
        }
    }*/
short iRez = 1;

    if (!semaphore_availability_data_buffer() || m_uall_transferred_data >=
m_uFileSize) return 0;

    m_size_rec = 0;

    if (m_num_read_buff == 2 && m_size_transferred_data + p_size_receive_data
== m_sizeBuf)
    {
        m_size_rec = 0;
    }

    if (m_sizeBuf - m_size_transferred_data >= p_size_receive_data) //
Если в текущем буфере еще остались данные
    {

        if (m_uFileSize - m_uall_transferred_data <=
p_size_receive_data)
        {
            m_size_transferred_data = 0;

            m_sign_full_buff1 = false;
            m_sign_full_buff2 = false;

```

```

        m_size_rec = m_uFileSize - m_uall_transferred_data;
    }
    else
        m_size_rec = p_size_receive_data;
        // то этот объем данных мы копируем в выходной буфер и
увеличиваем счетчик объема переданных данных.

        if (m_num_read_buff == 1) memcpy(data,
&m_Buff1[m_size_transferred_data], m_size_rec);
        if (m_num_read_buff == 2) memcpy(data,
&m_Buff2[m_size_transferred_data], m_size_rec);

        if (m_sizeBuf - m_size_transferred_data ==
p_size_receive_data) //Если количество данных ровно столько же, сколько в
запросили
        {
            if (m_num_read_buff == 1)
            {
                TRACE("m_num_read_buff -> 2\n\r");
                m_num_read_buff = 2;
                m_size_transferred_data = 0;
                memset(m_Buff1, 0, m_sizeBuf);
            }
            if (m_num_read_buff == 2)
            {
                TRACE("m_num_read_buff -> 1\n\r");
                m_num_read_buff = 1;
                m_size_transferred_data = 0;
                memset(m_Buff2, 0, m_sizeBuf);
            }
        }

        m_size_transferred_data += m_size_rec;
        m_uall_transferred_data += m_size_rec;
        return m_size_rec;
    }
    else
        if (m_sizeBuf - m_size_transferred_data > 0) // Если в
текущем буфере еще остались данные, но объемом меньше запрашиваемого

```

```

    {
        m_size_rec = m_sizeBuf - m_size_transferred_data; //
получаем остаток данных в бффере

        char* tmp_buf = new char[m_sizeBuf];

        switch (m_num_read_buff)
        {
            case 1:
                memcpy(tmp_buf, &m_Buff1[m_size_transferred_data],
m_size_rec); // Заносим во временный буфер остатки из первого буфера
                memset(m_Buff1, 0, m_sizeBuf); //
Обнуляем первый буфер

                m_size_transferred_data = 0;
                break;

            case 2:
                memcpy(tmp_buf, &m_Buff2[m_size_transferred_data],
m_size_rec);
                memset(m_Buff2, 0, m_sizeBuf); //
Обнуляем второй буфер

                m_size_transferred_data = 0;
                break;
        }
        // Проверем есть ли данные во втором буфере
        if (m_num_read_buff == 1 && m_sign_full_buff2)
// Текущий буфер m_buff1Sample (первый) и есть данные в втором буфере
        {
            m_sign_full_buff1 = false; //
Сбрасываем флаг наличия данных в первом буфере.
            // m_size_data_buff1 = 0;

            m_num_read_buff = 2; //
Устанавливаем признак активности второго буфера.

            //Тут необходимо к остатку данных из первого
буфера добавить кусок из второго
            m_size_transferred_data = p_size_receive_data -
m_size_rec; // Высчитываем объем данных к передаче из второго буфера

```

```

        memcpy(&tmp_buf[m_size_rec], &m_Buff2,
m_size_transferred_data); // Дополняем временный буфер недостающим объемом из
второго буфера

        m_size_rec = m_size_rec + m_size_transferred_data;
    }
    else
        if (m_sign_full_buff1 && m_num_read_buff == 2)
        {
            m_sign_full_buff2 = false;
            //          m_size_data_buff2 = 0;

            m_num_read_buff = 1;

            memcpy(tmp_buf,
&m_Buff2[m_size_transferred_data], m_size_rec);

            m_size_transferred_data =
p_size_receive_data - m_size_rec;

            memcpy(&tmp_buf[m_size_rec], &m_Buff1,
m_size_transferred_data);

            m_size_rec = m_size_rec +
m_size_transferred_data;
        }
        memcpy(data, tmp_buf, m_size_rec);

        delete[] tmp_buf;
    }
    else
        if (m_size_transferred_data == m_uFileSize)
        {
            m_size_transferred_data = 0;
            m_sign_full_buff1 = false;
            memset(m_Buff1, 0, m_sizeBuf);

            m_sign_full_buff2 = false;
            memset(m_Buff2, 0, m_sizeBuf);
        }

    m_uall_transferred_data += m_size_rec;
    return m_size_rec;

```

```
}
```

Листинг В.16 – Модуль Common.cpp

```
#include "pch.h"
#include "SessionTCP.h"
#include "DiplomDlg.h"
#include "Common.h"

#define STR_COPIED_COMPLETELY      1           // Строка скопирована в буффер
                                         полностью
#define STR_COPIED_PARTIALLY      2           // В буффер перенесена часть
                                         строки
#define STR_COPIED_ERROR          3

extern CSessionTCP* tcp_session;

u_short StrToChar(CString* str, char* cBuf, int iSizeBuf)
{
    CString sString = *str;
    CString sSymbol;

    u_short ret_val = 0;

    u_short len_str = sString.GetLength();

    if (iSizeBuf == 0)          return STR_COPIED_ERROR;
    if (len_str > iSizeBuf) ret_val = STR_COPIED_PARTIALLY;
    else
    {
        iSizeBuf = len_str;
        ret_val = STR_COPIED_COMPLETELY;
    }
    for (int i = 0; i <= iSizeBuf; i++)
        cBuf[i] = (char)sString.GetAt(i);

    return ret_val;
}

std::string to_std_string(CString const& src)
{
    auto p = src.GetString();
    auto l = src.GetLength();
    size_t dstSz = ::WideCharToMultiByte(CP_UTF8, 0, p, l, 0, 0, 0, 0);
```

```

        std::vector<char> dst(dstSz, '\\0');
        ::WideCharToMultiByte(CP_UTF8, 0, p, 1, dst.data(), dst.size(), 0, 0);

        return { dst.begin(), dst.end() };
    }

```

Листинг В.17 – Модуль Diplom.cpp

```

// Diplom.cpp: определяет поведение классов для приложения.
//

#include "pch.h"
#include "framework.h"

#include "Diplom.h"
#include "DiplomDlg.h"
#include "SessionTCP.h"
#include "SharedMemory.h"
#include "thread_guard.h"
#include "SessionRTP.h"

#include <thread>

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

CSessionRTP* rtp_session;
CSessionTCP* tcp_session;
CDiplomDlg* o_DiplomDLG;
SharedMemory* o_SharedMemory;

BOOL g_bFlagTCPManager = true;
BOOL g_bFlagWorkThread = true;

// CDiplomApp

BEGIN_MESSAGE_MAP(CDiplomApp, CWinApp)
    ON_COMMAND(ID_HELP, &CWinApp::OnHelp)
END_MESSAGE_MAP()

```

```

// Создание CDiplomApp

CDiplomApp::CDiplomApp()
{
    // поддержка диспетчера перезагрузки
    m_dwRestartManagerSupportFlags = AFX_RESTART_MANAGER_SUPPORT_RESTART;

    // TODO: добавьте код создания,
    // Размещает весь важный код инициализации в InitInstance
}

// Единственный объект CDiplomApp

CDiplomApp theApp;

// Инициализация CDiplomApp

BOOL CDiplomApp::InitInstance()
{
    // InitCommonControlsEx() требуется для Windows XP, если манифест
    // приложения использует ComCtl32.dll версии 6 или более поздней версии
    для включения
    // стилей отображения. В противном случае будет возникать сбой при
    создании любого окна.
    INITCOMMONCONTROLSEX InitCtrls;
    InitCtrls.dwSize = sizeof(InitCtrls);
    // Выберите этот параметр для включения всех общих классов управления,
    которые необходимо использовать
    // в вашем приложении.
    InitCtrls.dwICC = ICC_WIN95_CLASSES;
    InitCommonControlsEx(&InitCtrls);

    CWinApp::InitInstance();

    AfxEnableControlContainer();

    // Создать диспетчер оболочки, в случае, если диалоговое окно содержит
    // представление дерева оболочки или какие-либо его элементы управления.
    CShellManager *pShellManager = new CShellManager;

```

```

        // Активация визуального диспетчера "Классический Windows" для включения
элементов управления MFC
        CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS(CMFCVisualManagerWindow
s));

        // Стандартная инициализация
        // Если эти возможности не используются и необходимо уменьшить размер
        // конечного исполняемого файла, необходимо удалить из следующих
        // конкретных процедур инициализации, которые не требуются
        // Измените раздел реестра, в котором хранятся параметры
        // TODO: следует изменить эту строку на что-нибудь подходящее,
        // например на название организации
        SetRegistryKey(_T("Локальные приложения, созданные с помощью мастера
приложений"));

        rtp_session      = new CSessionRTP;
        o_SharedMemory    = new SharedMemory;
        tcp_session       = new CSessionTCP;
        o_DiplomDLG       = new CDiplomDlg;

        m_pMainWnd = o_DiplomDLG;

        // Запускается в отдельном потоке диспетчер задач
        std::thread thread_SessionTCP(&CSessionTCP::TaskManagerTCP, tcp_session);

        // Конструктору thread_guard передается поток диспетчера задач для
отслеживания правильного завершения этого потока.
        thread_guard my_thread_guard(std::move(thread_SessionTCP));

        INT_PTR nResponse = o_DiplomDLG->DoModal();
        if (nResponse == IDOK)
        {
            // TODO: Введите код для обработки закрытия диалогового окна
            // с помощью кнопки "ОК"
        }
        else if (nResponse == IDCANCEL)
        {
            // TODO: Введите код для обработки закрытия диалогового окна
            // с помощью кнопки "Отмена"
        }

```

```

        else if (nResponse == -1)
        {
            TRACE(traceAppMsg, 0, "Предупреждение. Не удалось создать диалоговое
окно, поэтому работа приложения неожиданно завершена.\n");
            TRACE(traceAppMsg, 0, "Предупреждение. При использовании элементов
управления MFC для диалогового окна невозможно #define
_AFX_NO_MFC_CONTROLS_IN_DIALOGS.\n");
        }

        // Удалить диспетчер оболочки, созданный выше.
        if (pShellManager != nullptr)
        {
            delete pShellManager;
        }

#ifdef !defined(_AFXDLL) && !defined(_AFX_NO_MFC_CONTROLS_IN_DIALOGS)
        ControlBarCleanUp();
#endif

        // Поскольку диалоговое окно закрыто, возвратите значение FALSE, чтобы
        можно было выйти из
        // приложения вместо запуска генератора сообщений приложения.
        return FALSE;
    }

```

Листинг В.18 – Модуль DiplomDLG.cpp

```

// DiplomDlg.cpp: файл реализации
//
#include "pch.h"
#include "framework.h"
#include "Diplom.h"
#include "DiplomDlg.h"
#include "Common.h"
#include "SessionTCP.h"
#include "SharedMemory.h"
#include "defines.h"
#include "SessionRTP.h"

#include <afx.h>
#include <afxwin.h>
#include <winsock2.h>
#include <fstream>

```

```

#include "afxdialogex.h"
#include <sys/types.h>
#include "ws2tcpip.h"
#include <iphlpapi.h>
#include <icmpapi.h>
#include <string>

#include <direct.h>

using namespace SCO_COMMON;

extern CSessionRTP* rtp_session;
extern CSessionTCP* tcp_session;
extern SharedMemory* o_SharedMemory;
extern CDiplomDlg* o_DiplimDLG;

// Диалоговое окно CAboutDlg используется для описания сведений о приложении

class CAboutDlg : public CDialogEx
{
public:
    CAboutDlg();

    // Данные диалогового окна
#ifdef AFX_DESIGN_TIME
    enum { IDD = IDD_ABOUTBOX };
#endif

protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // поддержка DDX/DDV

    // Реализация
protected:
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialogEx(IDD_ABOUTBOX)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)

```

```

{
    CDialogEx::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialogEx)
END_MESSAGE_MAP()

// Диалоговое окно CDiplomDlg

CDiplomDlg::CDiplomDlg(CWnd* pParent /*=nullptr*/)
    : CDialogEx(IDD_DIPLOM_DIALOG, pParent)
    , CMB_ValueIPSCO(_T(""))
    , TB_IPSCO(_T(""))
    , ED_VAL_Info(_T(""))

    , CB_Station1_Value(FALSE)
    , CB_Station1_1_Value(FALSE)
    , CB_Station1_2_Value(FALSE)
    , CB_Station1_3_Value(FALSE)

    , CB_Station2_Value(FALSE)
    , CB_Station2_1_Value(FALSE)
    , CB_Station2_2_Value(FALSE)
    , CB_Station2_3_Value(FALSE)

    , CB_Station3_Value(FALSE)
    , CB_Station3_1_Value(FALSE)
    , CB_Station3_2_Value(FALSE)
    , CB_Station3_3_Value(FALSE)

    , CB_Station4_Value(FALSE)
    , CB_Station4_1_Value(FALSE)
    , CB_Station4_2_Value(FALSE)
    , CB_Station4_3_Value(FALSE)

    , CB_Station5_Value(FALSE)
    , CB_Station5_1_Value(FALSE)
    , CB_Station5_2_Value(FALSE)
    , CB_Station5_3_Value(FALSE)

```

```

        , CB_Station6_1_Value(FALSE)
        , CB_Station6_2_Value(FALSE)
        , CB_Station6_3_Value(FALSE)

        , CB_Station7_1_Value(FALSE)
        , CB_Station7_2_Value(FALSE)
        , CB_Station7_3_Value(FALSE)

        , CB_Station8_1_Value(FALSE)
        , CB_Station8_2_Value(FALSE)
        , CB_Station8_3_Value(FALSE)
        , v_sSample(_T(""))
    {
        m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
    }

void CDiplomDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialogEx::DoDataExchange(pDX);
    DDX_Control(pDX, IDC_CB_IP, CB_EnterIp);
    DDX_Control(pDX, IDC_EDIT2, TB_IPSCO_Element);

    DDX_CBString(pDX, IDC_CB_IP, CMB_ValueIPSCO);

    DDX_Text(pDX, IDC_EDIT2, TB_IPSCO);
    DDX_Text(pDX, IDC_EDIT2, TB_IPSCO);
    DDX_Control(pDX, ED_Info, ED_EL_Info);
    DDX_Text(pDX, ED_Info, ED_VAL_Info);

    DDX_Control(pDX, IDC_CB_Station1, CB_Station1);
    DDX_Control(pDX, IDC_CB_Station2, CB_Station2);
    DDX_Control(pDX, IDC_CB_Station3, CB_Station3);
    DDX_Control(pDX, IDC_CB_Station4, CB_Station4);
    DDX_Control(pDX, IDC_CB_Station5, CB_Station5);
    DDX_Control(pDX, IDC_CB_Station6, CB_Station6);
    DDX_Control(pDX, IDC_CB_Station7, CB_Station7);
    DDX_Control(pDX, IDC_CB_Station8, CB_Station8);

    DDX_Control(pDX, IDC_CB_Node1_1, CB_Station1_1);
    DDX_Control(pDX, IDC_CB_Node1_2, CB_Station1_2);
    DDX_Control(pDX, IDC_CB_Node1_3, CB_Station1_3);

```

```

DDX_Control(pDX, IDC_CB_Node2_1, CB_Station2_1);
DDX_Control(pDX, IDC_CB_Node2_2, CB_Station2_2);
DDX_Control(pDX, IDC_CB_Node2_3, CB_Station2_3);

DDX_Control(pDX, IDC_CB_Node3_1, CB_Station3_1);
DDX_Control(pDX, IDC_CB_Node3_2, CB_Station3_2);
DDX_Control(pDX, IDC_CB_Node3_3, CB_Station3_3);

DDX_Control(pDX, IDC_CB_Node4_1, CB_Station4_1);
DDX_Control(pDX, IDC_CB_Node4_2, CB_Station4_2);
DDX_Control(pDX, IDC_CB_Node4_3, CB_Station4_3);

DDX_Control(pDX, IDC_CB_Node5_1, CB_Station5_1);
DDX_Control(pDX, IDC_CB_Node5_2, CB_Station5_2);
DDX_Control(pDX, IDC_CB_Node5_3, CB_Station5_3);

DDX_Control(pDX, IDC_CB_Node6_1, CB_Station6_1);
DDX_Control(pDX, IDC_CB_Node6_2, CB_Station6_2);
DDX_Control(pDX, IDC_CB_Node6_3, CB_Station6_3);

DDX_Control(pDX, IDC_CB_Node7_1, CB_Station7_1);
DDX_Control(pDX, IDC_CB_Node7_2, CB_Station7_2);
DDX_Control(pDX, IDC_CB_Node7_3, CB_Station7_3);

DDX_Control(pDX, IDC_CB_Node8_1, CB_Station8_1);
DDX_Control(pDX, IDC_CB_Node8_2, CB_Station8_2);
DDX_Control(pDX, IDC_CB_Node8_3, CB_Station8_3);

DDX_Check(pDX, IDC_CB_Station1, CB_Station1_Value);
DDX_Check(pDX, IDC_CB_Node1_1, CB_Station1_1_Value);
DDX_Check(pDX, IDC_CB_Node1_2, CB_Station1_2_Value);
DDX_Check(pDX, IDC_CB_Node1_3, CB_Station1_3_Value);

DDX_Check(pDX, IDC_CB_Station2, CB_Station2_Value);
DDX_Check(pDX, IDC_CB_Node2_1, CB_Station2_1_Value);
DDX_Check(pDX, IDC_CB_Node2_2, CB_Station2_2_Value);
DDX_Check(pDX, IDC_CB_Node2_3, CB_Station2_3_Value);

DDX_Check(pDX, IDC_CB_Station3, CB_Station3_Value);
DDX_Check(pDX, IDC_CB_Node3_1, CB_Station3_1_Value);
DDX_Check(pDX, IDC_CB_Node3_2, CB_Station3_2_Value);

```

```

DDX_Check(pDX, IDC_CB_Node3_3, CB_Station3_3_Value);

DDX_Check(pDX, IDC_CB_Station4, CB_Station4_Value);
DDX_Check(pDX, IDC_CB_Node4_1, CB_Station4_1_Value);
DDX_Check(pDX, IDC_CB_Node4_2, CB_Station4_2_Value);
DDX_Check(pDX, IDC_CB_Node4_3, CB_Station4_3_Value);

DDX_Check(pDX, IDC_CB_Station5, CB_Station5_Value);
DDX_Check(pDX, IDC_CB_Node5_1, CB_Station5_1_Value);
DDX_Check(pDX, IDC_CB_Node5_2, CB_Station5_2_Value);
DDX_Check(pDX, IDC_CB_Node5_3, CB_Station5_3_Value);

DDX_Check(pDX, IDC_CB_Station6, CB_Station6_Value);
DDX_Check(pDX, IDC_CB_Node6_1, CB_Station6_1_Value);
DDX_Check(pDX, IDC_CB_Node6_2, CB_Station6_2_Value);
DDX_Check(pDX, IDC_CB_Node6_3, CB_Station6_3_Value);

DDX_Check(pDX, IDC_CB_Station7, CB_Station7_Value);
DDX_Check(pDX, IDC_CB_Node7_1, CB_Station7_1_Value);
DDX_Check(pDX, IDC_CB_Node7_2, CB_Station7_2_Value);
DDX_Check(pDX, IDC_CB_Node7_3, CB_Station7_3_Value);

DDX_Check(pDX, IDC_CB_Station8, CB_Station8_Value);
DDX_Check(pDX, IDC_CB_Node8_1, CB_Station8_1_Value);
DDX_Check(pDX, IDC_CB_Node8_2, CB_Station8_2_Value);
DDX_Check(pDX, IDC_CB_Node8_3, CB_Station8_3_Value);

DDX_LBString(pDX, IDC_LST_SAMPLE, v_sSample);
DDX_Control(pDX, IDC_LST_SAMPLE, lstSample);

DDX_Control(pDX, IDC_PB_Station1_1, PB_Station1_1_Value);
DDX_Control(pDX, IDC_PB_Station1_2, PB_Station1_2_Value);
DDX_Control(pDX, IDC_PB_Station1_3, PB_Station1_3_Value);

DDX_Control(pDX, IDC_PB_Station2_1, PB_Station2_1_Value);
DDX_Control(pDX, IDC_PB_Station2_2, PB_Station2_2_Value);
DDX_Control(pDX, IDC_PB_Station2_3, PB_Station2_3_Value);

DDX_Control(pDX, IDC_PB_Station3_1, PB_Station3_1_Value);
DDX_Control(pDX, IDC_PB_Station3_2, PB_Station3_2_Value);
DDX_Control(pDX, IDC_PB_Station3_3, PB_Station3_3_Value);

```

```

DDX_Control(pDX, IDC_PB_Station4_1, PB_Station4_1_Value);
DDX_Control(pDX, IDC_PB_Station4_2, PB_Station4_2_Value);
DDX_Control(pDX, IDC_PB_Station4_3, PB_Station4_3_Value);

DDX_Control(pDX, IDC_PB_Station5_1, PB_Station5_1_Value);
DDX_Control(pDX, IDC_PB_Station5_2, PB_Station5_2_Value);
DDX_Control(pDX, IDC_PB_Station5_3, PB_Station5_3_Value);

DDX_Control(pDX, IDC_PB_Station6_1, PB_Station6_1_Value);
DDX_Control(pDX, IDC_PB_Station6_2, PB_Station6_2_Value);
DDX_Control(pDX, IDC_PB_Station6_3, PB_Station6_3_Value);

DDX_Control(pDX, IDC_PB_Station7_1, PB_Station7_1_Value);
DDX_Control(pDX, IDC_PB_Station7_2, PB_Station7_2_Value);
DDX_Control(pDX, IDC_PB_Station7_3, PB_Station7_3_Value);

DDX_Control(pDX, IDC_PB_Station8_1, PB_Station8_1_Value);
DDX_Control(pDX, IDC_PB_Station8_2, PB_Station8_2_Value);
DDX_Control(pDX, IDC_PB_Station8_3, PB_Station8_3_Value);

}

BEGIN_MESSAGE_MAP(CDiplomDlg, CDialogEx)
    ON_WM_SYSCOMMAND()

    ON_WM_PAINT()

    ON_WM_QUERYDRAGICON()

    ON_EN_KILLFOCUS(IDC_EDIT2, &CDiplomDlg::OnEnKillfocusEdit2)

    ON_CBN_KILLFOCUS(IDC_CB_IP, &CDiplomDlg::OnCbnKillfocusCbIp)

    ON_BN_CLICKED(IDC_BUTTON1, &CDiplomDlg::OnInit)
    ON_BN_CLICKED(IDC_BUTTON2, &CDiplomDlg::OnConnect)

    ON_MESSAGE(WM_MSGTHREAD_ALL, &CDiplomDlg::GetSharMess)
    ON_MESSAGE(INPUT_FORM, &CDiplomDlg::ReadMess)

    ON_BN_CLICKED(IDC_CB_Station1, &CDiplomDlg::OnBnClickedCbStation)

```

```

ON_BN_CLICKED(IDC_CB_Station2, &CDiplomDlg::OnBnClickedCbStation)
ON_BN_CLICKED(IDC_CB_Station3, &CDiplomDlg::OnBnClickedCbStation)
ON_BN_CLICKED(IDC_CB_Station4, &CDiplomDlg::OnBnClickedCbStation)
ON_BN_CLICKED(IDC_CB_Station5, &CDiplomDlg::OnBnClickedCbStation)
ON_BN_CLICKED(IDC_CB_Station6, &CDiplomDlg::OnBnClickedCbStation)
ON_BN_CLICKED(IDC_CB_Station7, &CDiplomDlg::OnBnClickedCbStation)
ON_BN_CLICKED(IDC_CB_Station8, &CDiplomDlg::OnBnClickedCbStation)

ON_BN_CLICKED(IDC_CB_Node1_1, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node1_2, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node1_3, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node2_1, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node2_2, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node2_3, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node3_1, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node3_2, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node3_3, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node4_1, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node4_2, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node4_3, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node5_1, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node5_2, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node5_3, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node6_1, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node6_2, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node6_3, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node7_1, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node7_2, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node7_3, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node8_1, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node8_2, &CDiplomDlg::OnCheckNode)
ON_BN_CLICKED(IDC_CB_Node8_3, &CDiplomDlg::OnCheckNode)

ON_BN_CLICKED(IDC_BT_Reproduce, &CDiplomDlg::Reproducing_click)

ON_LBN_SELCHANGE(IDC_LST_SAMPLE, &CDiplomDlg::OnSelectSample)
END_MESSAGE_MAP()

// Обработчики сообщений CDiplomDlg

```

```

BOOL CDiplomDlg::OnInitDialog()
{
    CDialogEx::OnInitDialog();

    CB_Station_Value[0] = &CB_Station1_Value;
    CB_Station_Value[1] = &CB_Station2_Value;
    CB_Station_Value[2] = &CB_Station3_Value;
    CB_Station_Value[3] = &CB_Station4_Value;
    CB_Station_Value[4] = &CB_Station5_Value;
    CB_Station_Value[5] = &CB_Station6_Value;
    CB_Station_Value[6] = &CB_Station7_Value;
    CB_Station_Value[7] = &CB_Station8_Value;

    CB_Station_Node[0][0] = &CB_Station1_1_Value; CB_Station_Node[0][1] =
&CB_Station1_2_Value; CB_Station_Node[0][2] = &CB_Station1_3_Value;
    CB_Station_Node[1][0] = &CB_Station2_1_Value; CB_Station_Node[1][1] =
&CB_Station2_2_Value; CB_Station_Node[1][2] = &CB_Station2_3_Value;
    CB_Station_Node[2][0] = &CB_Station3_1_Value; CB_Station_Node[2][1] =
&CB_Station3_2_Value; CB_Station_Node[2][2] = &CB_Station3_3_Value;
    CB_Station_Node[3][0] = &CB_Station4_1_Value; CB_Station_Node[3][1] =
&CB_Station4_2_Value; CB_Station_Node[3][2] = &CB_Station4_3_Value;
    CB_Station_Node[4][0] = &CB_Station5_1_Value; CB_Station_Node[4][1] =
&CB_Station5_2_Value; CB_Station_Node[4][2] = &CB_Station5_3_Value;
    CB_Station_Node[5][0] = &CB_Station6_1_Value; CB_Station_Node[5][1] =
&CB_Station6_2_Value; CB_Station_Node[5][2] = &CB_Station6_3_Value;
    CB_Station_Node[6][0] = &CB_Station7_1_Value; CB_Station_Node[6][1] =
&CB_Station7_2_Value; CB_Station_Node[6][2] = &CB_Station7_3_Value;
    CB_Station_Node[7][0] = &CB_Station8_1_Value; CB_Station_Node[7][1] =
&CB_Station8_2_Value; CB_Station_Node[7][2] = &CB_Station8_3_Value;

    PB_Station[0][0] = &PB_Station1_1_Value; PB_Station[0][1] =
&PB_Station1_2_Value; PB_Station[0][2] = &PB_Station1_3_Value;
    PB_Station[1][0] = &PB_Station2_1_Value; PB_Station[1][1] =
&PB_Station2_2_Value; PB_Station[1][2] = &PB_Station2_3_Value;
    PB_Station[2][0] = &PB_Station3_1_Value; PB_Station[2][1] =
&PB_Station3_2_Value; PB_Station[2][2] = &PB_Station3_3_Value;
    PB_Station[3][0] = &PB_Station4_1_Value; PB_Station[3][1] =
&PB_Station4_2_Value; PB_Station[3][2] = &PB_Station4_3_Value;
    PB_Station[4][0] = &PB_Station5_1_Value; PB_Station[4][1] =
&PB_Station5_2_Value; PB_Station[4][2] = &PB_Station5_3_Value;
    PB_Station[5][0] = &PB_Station6_1_Value; PB_Station[5][1] =
&PB_Station6_2_Value; PB_Station[5][2] = &PB_Station6_3_Value;

```

```

        PB_Station[6][0] = &PB_Station7_1_Value; PB_Station[6][1] =
        &PB_Station7_2_Value; PB_Station[6][2] = &PB_Station7_3_Value;
        PB_Station[7][0] = &PB_Station8_1_Value; PB_Station[7][1] =
        &PB_Station8_2_Value; PB_Station[7][2] = &PB_Station8_3_Value;

        std::vector<std::string> MyIp;
        /***** Получаем IP адреса компа, на котором запущена данная
программа*****/
        GetIpAdd(MyIp);
        /*****
*****/

        UpdateData(true);

        int Schetchik = 0;

        CString a = L"";

        for (int i = 0; i < MyIp.size(); i++)           //Ввод в ComboBox значений
        {
            a = MyIp[i].c_str();

            CB_EnterIp.AddString(a);
        }

        ifstream in("D:\\IP.txt");

        if (in.is_open())
        {
            string Ip = "";
            string Ip_SCO = "";
            string HSC = "SHSC", SCO = "SCO";

            getline(in, Ip);
            in.close();
            string Ip_Change = Ip;
            Ip_SCO = Ip;

            if (Ip_Change.find(HSC) != string::npos && Ip_Change.find(SCO) !=
string::npos)
            {

```

```

        if (Ip_Change.find(HSC) < Ip_Change.find(SCO))
//Правильное построение документа
        {
            Ip.erase(0, Ip.find(HSC) + HSC.length() + 1);
            Ip.erase(Ip.find(SCO) - 1, Ip.length());

            for (int i = 0; i < MyIp.size(); i++)
            {
                a = MyIp[i].c_str();
                if (Schetchik == 0)
                {
                    if (Ip.c_str() == a)
                    {
                        CMB_ValueIPSCO = a;
                        Schetchik++;
                    }
                    else CMB_ValueIPSCO = a;
                }
            }
            Ip_SCO.erase(0, Ip_SCO.find(SCO) + SCO.length() + 1);
            a = Ip_SCO.c_str();
            TB_IPSCO = a;
        }
        else if (Ip_Change.find(HSC) > Ip_Change.find(SCO))
//SCO стоит спереди
        {
            Ip.erase(0, Ip.find(SCO) + SCO.length() + 1);
            Ip.erase(Ip.find(HSC) - 1, Ip.length());

            for (int i = 0; i < MyIp.size(); i++)
            {
                a = MyIp[i].c_str();
                if (Schetchik == 0)
                {
                    if (Ip_SCO.c_str() == a)
                    {
                        CMB_ValueIPSCO = a;
                        Schetchik++;
                    }
                    else CMB_ValueIPSCO = a;
                }
            }
        }
    }
}

```

```

        CMB_ValueIPSCO = Ip.c_str();
        Ip_SCO.erase(0, Ip_SCO.find(HSC) + HSC.length() + 1);
        TB_IPSCO = Ip_SCO.c_str();

    }

}

else if (Ip_Change.find(HSC) != string::npos && Ip_Change.find(SCO)
== string::npos) //Есть только HSC
{
    Ip.erase(0, Ip.find(HSC) + HSC.length() + 1);

    for (int i = 0; i < MyIp.size(); i++)
    {
        a = MyIp[i].c_str();
        if (Schetchik == 0)
        {
            if (Ip.c_str() == a)
            {
                CMB_ValueIPSCO = a;
                Schetchik++;
            }
            else CMB_ValueIPSCO = a;
        }
    }

}

else if (Ip_Change.find(HSC) == string::npos && Ip_Change.find(SCO)
!= string::npos) //Есть только SCO
{
    Ip.erase(0, Ip.find(SCO) + SCO.length() + 1);
    a = Ip_SCO.c_str();
    TB_IPSCO = a;
}

if (Ip_Change.find(HSC) == string::npos && Ip_Change.find(SCO) ==
string::npos) //Нет ничего
{
    CMB_ValueIPSCO = a;
    TB_IPSCO = L"192.168.1.12";
}

/*          Ip_SCO = Ip;

        Ip.erase(0, Ip.find(start) + start.length() + 1);
        Ip.erase(Ip.find(finish) - 1, Ip.length());

```

```

        for (int i = 0; i < MyIp.size(); i++)
        {
            a = MyIp[i].c_str();
            if (Schetchik == 0)
            {

                if (Ip.c_str() == a)
                {
                    CMB_ValueIPSCO = a;
                    Schetchik++;
                }
                else CMB_ValueIPSCO = a;
            }
        }
        Ip_SCO.erase(0, Ip_SCO.find(finish) +
finish.length() + 1);

        a = Ip_SCO.c_str();
        TB_IPSCO = a;*/

    }
    else
        CMB_ValueIPSCO = a;

    FormationListSamples();

    UpdateData(false);

    // Добавление пункта "О программе..." в системное меню.

    // IDM_ABOUTBOX должен быть в пределах системной команды.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != nullptr)
    {
        BOOL bNameValid;
        CString strAboutMenu;
        bNameValid = strAboutMenu.LoadString(IDS_ABOUTBOX);
        ASSERT(bNameValid);
        if (!strAboutMenu.IsEmpty())
        {

```

```

        pSysMenu->AppendMenu(MF_SEPARATOR);
        pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
    }
}

// Задаёт значок для этого диалогового окна. Среда делает это
автоматически,
// если главное окно приложения не является диалоговым
SetIcon(m_hIcon, TRUE);           // Крупный значок
SetIcon(m_hIcon, FALSE);          // Мелкий значок

// TODO: добавьте дополнительную инициализацию

return TRUE; // возврат значения TRUE, если фокус не передан элементу
управления
}

void CDiplomDlg::FormationListSamples()
{
    char m_File[_MAX_PATH];

    _getcwd(m_File, sizeof(m_File));

    strcat_s(m_File, (char*)"\\Sample");

    CString strCurDir(m_File, strlen(m_File));

    m_sDirSample = strCurDir;

    DWORD dwAttr = GetFileAttributes(m_sDirSample);

    switch (dwAttr)
    {
    case FILE_ATTRIBUTE_ARCHIVE:
        break;
    case FILE_ATTRIBUTE_COMPRESSED:
        break;
    case FILE_ATTRIBUTE_DIRECTORY:
        break;
    case FILE_ATTRIBUTE_HIDDEN:

```

```

        break;
    case FILE_ATTRIBUTE_NORMAL:
        break;
    case FILE_ATTRIBUTE_OFFLINE:
        break;
    case FILE_ATTRIBUTE_READONLY:
        break;
    case 0xffffffff:
        DWORD dwError = GetLastError();

        if (dwError == ERROR_FILE_NOT_FOUND)
        {
            MessageBox(L"Not Found", L"Error", MB_OK);
        }
        return;
    }
    CFileFind          m_finder;

    BOOL               bWorking;

    bWorking = m_finder.FindFile(m_sDirSample + L"\\*.mp3");

    CString m_FileName;

    while (bWorking)
    {
        bWorking = m_finder.FindNextFile();

        if (!m_finder.IsDirectory() && !m_finder.IsDots())
        {
            m_FileName = m_finder.GetFileName();

            lstSample.AddString(m_FileName);
        }
    }

    lstSample.SetCurSel(0);

    OnSelectSample();
}

void CDiplomDlg::OnSysCommand(UINT nID, LPARAM lParam)

```

```

{
    if ((nID & 0xFFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialogEx::OnSysCommand(nID, lParam);
    }
}

// При добавлении кнопки свертывания в диалоговое окно нужно воспользоваться
// приведенным ниже кодом,
// чтобы нарисовать значок. Для приложений MFC, использующих модель документов
// или представлений,
// это автоматически выполняется рабочей областью.

void CDiplomDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // контекст устройства для рисования

        SendMessage(WM_ICONERASEBKGND,
reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

        // Выравнивание значка по центру клиентского прямоугольника
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);

        CRect rect;

        GetClientRect(&rect);

        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Нарисуйте значок
        dc.DrawIcon(x, y, m_hIcon);
    }
    else

```

```

    {
        CDialogEx::OnPaint();
    }
}

// Система вызывает эту функцию для получения отображения курсора при
// перемещении
// свернутого окна.
HCURSOR CDiplomDlg::OnQueryDragIcon()
{
    return static_cast<HCURSOR>(m_hIcon);
}

void CDiplomDlg::Init(CString sIP_SCO, CString sIP_SHSC)
{
    CMB_ValueIPSCO = sIP_SCO;
    TB_IPSCO = sIP_SHSC;

    return;
}

#pragma comment(lib, "IPHLPAPI.lib")
// #pragma comment(lib, "ws2_32.lib")

void CDiplomDlg::GetIpAdd(std::vector<std::string> &MyIp)
{
    IP_ADAPTER_ADDRESSES* adapter_addresses(NULL);
    IP_ADAPTER_ADDRESSES* adapter(NULL);

    // Start with a 16 KB buffer and resize if needed -
    // multiple attempts in case interfaces change while
    // we are in the middle of querying them.
    DWORD adapter_addresses_buffer_size = 16 * 1000;

    for (int attempts = 0; attempts != 3; ++attempts)
    {
        adapter_addresses =
(IP_ADAPTER_ADDRESSES*)malloc(adapter_addresses_buffer_size);
        assert(adapter_addresses);

        DWORD error = ::GetAdaptersAddresses(

```

```

        AF_UNSPEC,
        GAA_FLAG_SKIP_ANYCAST |
        GAA_FLAG_SKIP_MULTICAST |
        GAA_FLAG_SKIP_DNS_SERVER |
        GAA_FLAG_SKIP_FRIENDLY_NAME,
        NULL,
        adapter_addresses,
        &adapter_addresses_buffer_size);

if (ERROR_SUCCESS == error)
{
    // We're done here, people!
    break;
}
else if (ERROR_BUFFER_OVERFLOW == error)
{
    // Try again with the new size
    free(adapter_addresses);
    adapter_addresses = NULL;

    continue;
}
else
{
    // Unexpected error code - log and throw
    free(adapter_addresses);
    adapter_addresses = NULL;

    // @todo
    //OG_AND_THROW_HERE();
}
}

// Iterate through all of the adapters
for (adapter = adapter_addresses; NULL != adapter; adapter = adapter-
>Next)
{
    // Skip loopback adapters
    if (IF_TYPE_SOFTWARE_LOOPBACK == adapter->IfType)
    {
        continue;
    }
}

```

```

        // Parse all IPv4 and IPv6 addresses
        for (IP_ADAPTER_UNICAST_ADDRESS* address = adapter-
>FirstUnicastAddress; NULL != address; address = address->Next)
        {
            auto family = address->Address.lpSockaddr->sa_family;
            if (AF_INET == family)
            {
                // IPv4
                SOCKADDR_IN* ipv4 =
reinterpret_cast<SOCKADDR_IN*>(address->Address.lpSockaddr);

                char str_buffer[INET_ADDRSTRLEN] = { 0 };
                inet_ntop(AF_INET, &(ipv4->sin_addr), str_buffer,
INET_ADDRSTRLEN);

                MyIp.push_back(str_buffer);
                //ipAddrs.mIpv4.push_back(str_buffer);
            }
        }
    }
    return;
}

```

```

void CDiplomDlg::OnEnKillfocusEdit2()
{
    UpdateData(true);

    wstring Ip_enter = L"";

    string Perenos;
    ifstream in("D:\\IP.txt");
    if (in.is_open())
    {
        getline(in, Perenos);
        in.close();
    }

    ofstream out("D:\\IP.txt");
    if (out.is_open())

```

```

{
    string HSC = "SHSC", SCO = "SCO";

    Ip_enter = TB_IPSCO;

    Perenos.erase(0, Perenos.find(HSC) + HSC.length() + 1);

    string s_Ip_enter(Ip_enter.begin(), Ip_enter.end());
    s_Ip_enter = Perenos + s_Ip_enter;

    out.write(s_Ip_enter.c_str(), s_Ip_enter.size());
    out.close();
}
return;
}

```

```

void CDiplomDlg::OnCbnKillfocusCbIp()
{
    UpdateData(true);

    wstring Ip_enter = L"";

    string Perenos;
    ifstream in("D:\\IP.txt");
    if (in.is_open())
    {
        getline(in, Perenos);
        in.close();
    }

    ofstream out("D:\\IP.txt");
    if (out.is_open())
    {
        string HSC = "HSC", SCO = "SCO";

        Ip_enter = CMB_ValueIPSCO;

        Perenos.erase(0, Perenos.find(SCO) - 1);

        string s_Ip_enter(Ip_enter.begin(), Ip_enter.end());

```

```

        s_Ip_enter = "HSC:" + s_Ip_enter + Perenos;

        out.write(s_Ip_enter.c_str(), s_Ip_enter.size());
        out.close();
    }
    return;
}

void CDiplomDlg::OnInit()
{
    switch (tcp_session->InitSocket())
    {
    case TCP_SOCKET_CREATE_SUCCESFULLY:
    {
        ED_VAL_Info += L"Сокет TCP создан успешно.\r\n";

        ED_VAL_Info += PingIP(CMB_ValueIPSCO);
        ED_VAL_Info += PingIP(TB_IPSCO);
        break;
    }
    case SOCKET_INITIALIZATION_ERROR:
    {
        ED_VAL_Info += L"Ошибка инициализации сокета!\r\n";
        WSACleanup();
        break;
    }
    case ERROR_TRANSFER:
    {
        ED_VAL_Info += L"Ошибка переноса Ip!\r\n";
        WSACleanup();
        break;
    }
    case ERROR_CONNECTONG_SOCKET_TO_SERVER:
    {
        ED_VAL_Info += L"Ошибка присоединения сокета к серверу!\r\n";
        WSACleanup();
        break;
    }
    }
}

```

```

    if (rtp_session->InitRTPSocket())
        ED_VAL_Info += L"Сокет RTP инициализирован успешно!\r\n";
    else
        ED_VAL_Info += L"Ошибка инициализации RTP сокета!\r\n";

    UpdateData(false);
}

CString CDiplomDlg::PingIP(CString sIP)
{
    HANDLE hIcmpFile;

    DWORD dwRetVal = 0;

    char SendData[32] = "Data Buffer";

    LPVOID ReplyBuffer = NULL;

    DWORD ReplySize = 0;

    // unsigned long ipaddr = INADDR_NONE;

    CString ls_Res = L"", l_sIP = sIP;

    int iLenIP = 0;

    char buff_ip[16];

    iLenIP = l_sIP.GetLength();

    memset(buff_ip, 0, sizeof(buff_ip));

    StrToChar(&l_sIP, buff_ip, iLenIP);

    SOCKADDR_IN RecvAddr;

    InetPton(AF_INET, l_sIP, &RecvAddr.sin_addr.s_addr);

    hIcmpFile = IcmpCreateFile();

    if (hIcmpFile == INVALID_HANDLE_VALUE)
        ls_Res = L"Ошибка открытия дескриптора эха IPv4.\r\n";
}

```

```

else
{
    ReplySize = sizeof(ICMP_ECHO_REPLY) + sizeof(SendData);

    ReplyBuffer = (VOID*)malloc(ReplySize);

    dwRetVal = IcmpSendEcho2(hIcmpFile, NULL, NULL, NULL,
RecvAddr.sin_addr.s_addr, SendData, sizeof(SendData), NULL, ReplyBuffer,
ReplySize, 1000);

    if (dwRetVal != 0)
        ls_Res.Format(L"Пинг IP = \"%s\" прошел успешно. Время пинга
%ld мс. \r\n", l_sIP, dwRetVal,
((PICMP_ECHO_REPLY)ReplyBuffer)->RoundTripTime);
    else
        ls_Res.Format(L"Пинг до IP = \"%s\" не прошел. IcmpSendEcho2
вернул ошибку: %ld \r\n", l_sIP, GetLastError());

    if (ReplyBuffer != NULL)
    {
        free(ReplyBuffer);
        ReplyBuffer = NULL;
    }
    IcmpCloseHandle(hIcmpFile);
}
return ls_Res;
}

void CDiplomDlg::OnConnect()
{
    clock_t iKey = clock();

    tcp_session->PutTaskTCP(iKey, CHECK_REQUEST_CLIENT);
}

LRESULT CDiplomDlg::GetSharMess(WPARAM wParam, LPARAM lParam)
{
    //o_SharedMemory->outSharedMemory(wParam);

    ED_VAL_Info += "Подключение прошло успешно \r\n";
}

```

```

        UpdateData(false);
        return 0;
    }

void CDiplomDlg::CheckStation(string UnlockCB)
{
    int StationFindBegin = UnlockCB.find(US_MUS);
    StationFindBegin += 4;

    string StationUnlockCB = UnlockCB;
    StationUnlockCB.erase(0, StationFindBegin);

    int StationFindEnd = StationUnlockCB.find("command");
    StationFindEnd -= 5;

    StationUnlockCB.erase(StationFindEnd, StationUnlockCB.length());

    StationFindBegin = UnlockCB.find(US_MUS);
    StationFindBegin += 7;

    string Station = UnlockCB;
    Station.erase(0, StationFindBegin);

    StationFindEnd = Station.find("command");
    StationFindEnd -= 3;

    Station = Station.erase(StationFindEnd, Station.length());

    if (StationUnlockCB == "01")
    {
        CB_Station1.EnableWindow(true);

        if (Station == "1")
            CB_Station1_1.EnableWindow(true);
        else if (Station == "2")
            CB_Station1_2.EnableWindow(true);
        else if (Station == "3")
            CB_Station1_3.EnableWindow(true);
    }
}

```

```

else if (StationUnlockCB == "02")
{
    CB_Station2.EnableWindow(true);

    if (Station == "1")
        CB_Station2_1.EnableWindow(true);
    else if (Station == "2")
        CB_Station2_2.EnableWindow(true);
    else if (Station == "3")
        CB_Station2_3.EnableWindow(true);
}
else if (StationUnlockCB == "03")
{
    CB_Station3.EnableWindow(true);

    if (Station == "1")
        CB_Station3_1.EnableWindow(true);
    else if (Station == "2")
        CB_Station3_2.EnableWindow(true);
    else if (Station == "3")
        CB_Station3_3.EnableWindow(true);
}
else if (StationUnlockCB == "04")
{
    CB_Station4.EnableWindow(true);

    if (Station == "1")
        CB_Station4_1.EnableWindow(true);
    else if (Station == "2")
        CB_Station4_2.EnableWindow(true);
    else if (Station == "3")
        CB_Station4_3.EnableWindow(true);
}
else if (StationUnlockCB == "05")
{
    CB_Station5.EnableWindow(true);

    if (Station == "1")
        CB_Station5_1.EnableWindow(true);
    else if (Station == "2")
        CB_Station5_2.EnableWindow(true);
    else if (Station == "3")

```

```

        CB_Station5_3.EnableWindow(true);
    }
    else if (StationUnlockCB == "06")
    {
        CB_Station6.EnableWindow(true);

        if (Station == "1")
            CB_Station6_1.EnableWindow(true);
        else if (Station == "2")
            CB_Station6_2.EnableWindow(true);
        else if (Station == "3")
            CB_Station6_3.EnableWindow(true);
    }
    else if (StationUnlockCB == "07")
    {
        CB_Station7.EnableWindow(true);
        if (Station == "1")
            CB_Station7_1.EnableWindow(true);
        else if (Station == "2")
            CB_Station7_2.EnableWindow(true);
        else if (Station == "3")
            CB_Station7_3.EnableWindow(true);
    }
    else if (StationUnlockCB == "08")
    {
        CB_Station8.EnableWindow(true);
        if (Station == "1")
            CB_Station8_1.EnableWindow(true);
        else if (Station == "2")
            CB_Station8_2.EnableWindow(true);
        else if (Station == "3")
            CB_Station8_3.EnableWindow(true);
    }

    UpdateWindow();
}

LRESULT CDiplomDlg::ReadMess(WPARAM wParam, LPARAM lParam)
{
    CString IntoInForm = L"";
    string UnlockCB = "";
    string sCommand = "";

```

```

UnlockCB = o_SharedMemory->outSharedMemory(wParam, lParam);

sCommand = GetElementaryPartTeam(UnlockCB, "command=");

IntoInForm = UnlockCB.c_str();

ED_VAL_Info += IntoInForm += "\r\n";

UpdateData(false);
return 0;
}

void CDiplomDlg::AnalysisReadMsg(string& p_sMsg)
{
    string sClient;
    string sCommand;

    clock_t numMsg = clock();

    sCommand = GetElementaryPartTeam(p_sMsg, "command=");

    //Определяем станцию и узел
    sClient = GetElementaryPartTeam(p_sMsg, "client=");

    if (sCommand == "connection")
    {
        CheckStation(p_sMsg);

        string UnlockCB = "";

        sCommand = p_sMsg;

        sCommand.replace(sCommand.find("REQUEST"), 7, "RESPONSE");

        sCommand.replace(sCommand.find("/>"), 2, " result=\"accepted\"/>");

        numMsg = o_SharedMemory->IntoSharedMemoryString(sCommand,
TCP_SOCKET_CONNECT);
        tcp_session->ThreadwriteTCP(numMsg, TCP_SOCKET_CONNECT);
    }
}

```

```

        p_sMsg = sCommand;
    }
    //<RESPONSE client="kievskaya" command="receive" action="start"
    session="15" url="rtp://192.168.222.139:2797/session_15" priority="24"
    result="accepted" />

    /*if (lv_sCommand == "transmit")
    {
        lv_sCommand = GetElementaryPartTeam(p_sMsg, L"action=");

        if (lv_sCommand == "start" || lv_sCommand == "circular")
        {
            numMsg = dlgMain->SetMsgStack(L"", TAKE_RTP, dataClient);
            dlgMain->PostMessage(WM_MSGTHREAD_R_RTP, numMsg,
iThreadNumber);
        }

        if (lv_sCommand == "finish")
        {
            if (p_iModeWork == STANDART_MODE)
            {
                // numMsg = dlgMain ->
SetMsgStack(L"", STOP_TAKE, dataClient);
                // dlgMain ->
PostMessage(WM_MSGTHREAD_R_RTP, numMsg, iThreadNumber);
            }
        }

    }*/

}

string CDiplomDlg::GetElementaryPartTeam(string& p_sMsg, string p_sElement)
{
    std::string v_sCommand = "",
        v_sSymbol = "";

    short i, v_iLenMsg, v_iLenElement, v_iVal = 0;

    v_iLenElement = p_sElement.size();

    if (p_sMsg != "")

```

```

{
    v_iVal = p_sMsg.find(p_sElement);
    if (v_iVal > 0)
    {
        v_iLenMsg = (short)p_sMsg.size();

        for (i = (v_iVal + v_iLenElement + 1); i < v_iLenMsg; i++)
        {

            v_sSymbol = p_sMsg[i];

            if (v_sSymbol != "" && v_sSymbol != "\\") v_sCommand +=
v_sSymbol;

            else break;

        }

    }

    return v_sCommand;
}

```

```

void CDiplomDlg::OnBnClickedCbStation()
{
    BOOL bStation[NUMBER_STATION], bFl = false;

    u_short iNumStation, i = 0, j = 0;

    for (i = 0; i < NUMBER_STATION; i++)        bStation[i] =
*CB_Station_Value[i];

    UpdateData(true);

    for (i = 0; i < NUMBER_STATION; i++)
    {
        if (bStation[i] != *CB_Station_Value[i])
        {
            iNumStation = i;
            bFl = true;
            *CB_Station_Value[i] = false;

            for (j = 0; j < NUMBER_NODE_STATION; j++)
            {

```

```

        *CB_Station_Value[iNumStation] = true;
        break;
    }
    break;
}

}

if (!bFl) return;
for (u_short j = 0; j < NUMBER_NODE_STATION; j++)
{
    *CB_Station_Node[iNumStation][j] = *CB_Station_Value[iNumStation];
}

UpdateData(false);
UpdateWindow();

}

void CDiplomDlg::OnCheckNode()
{
    BOOL bNode[NUMBER_STATION][NUMBER_NODE_STATION];

    u_short iStation, iNode;

    // Считываем все текущие значения чек-боксов зон
    for (u_short i = 0; i < NUMBER_STATION; i++)
        for (u_short j = 0; j < NUMBER_NODE_STATION; j++)
            bNode[i][j] = *CB_Station_Node[i][j];

    UpdateData(true);

    // Тут их сравнить с текущими, тем самым определяется зона с изменениями
    for (u_short i = 0; i < NUMBER_STATION; i++)
        for (u_short j = 0; j < NUMBER_NODE_STATION; j++)
        {
            if (bNode[i][j] != *CB_Station_Node[i][j])
            {
                iStation = i;
                iNode = j;
            }
        }
}

```

```

// Непосредственно изменение самих чекбоксов
for (u_short j = 0; j < NUMBER_NODE_STATION; j++)
{
    if (!*CB_Station_Node[iStation][j])
    {
        *CB_Station_Value[iStation] = false;
        break;
    }
    else *CB_Station_Value[iStation] = true;
}

UpdateData(false);
UpdateWindow();
}

void CDiplomDlg::Reproducing_click()
{
    BOOL bNode[NUMBER_STATION][NUMBER_NODE_STATION];

    m_sDirSample = m_sDirSample + L"\\\" + v_sSample;
    string FileName = to_std_string(m_sDirSample);

    tcp_session->GetFileName(FileName);

    for (u_short i = 0; i < NUMBER_STATION; i++)
        for (u_short j = 0; j < NUMBER_NODE_STATION; j++)
        {
            if (*CB_Station_Node[i][j] == true)
            {
                string I = to_string(i + 1);
                string J = to_string(j + 1);

                tcp_session->ReproducingRequest(I, J);
            }
        }

    // В этом месте передать имя файла в буффер
    rtp_session->GetFileName(FileName);
}

```

```

        return;
    }

void CDiplomDlg::OnSelectSample()
{
    int val2 = lstSample.GetCurSel();

    lstSample.SetCurSel(val2);

    UpdateData(true);
}

void CDiplomDlg::ProgressBarFilling(int Progress)
{
    for (int i = 0; i < NUMBER_STATION; i++)
        for (int j = 0; j < NUMBER_NODE_STATION; j++)
            if (*CB_Station_Node[i][j] == true)
                PB_Station[i][j]->SetPos(Progress);
}

```

Листинг В.19 – Модуль pch.cpp

// pch.cpp: файл исходного кода, соответствующий предварительно скомпилированному заголовочному файлу

```
#include "pch.h"
```

// При использовании предварительно скомпилированных заголовочных файлов необходим следующий файл исходного кода для выполнения сборки.

Листинг В.20 – Модуль SessionRTP.cpp

```
#include "pch.h"
```

```
#include "SessionRTP.h"
```

```
#include "Common.h"
```

```
#include "StructDef.h"
```

```
#include "defines.h"
```

```
#include "SharedMemory.h"
```

```
#include <ws2tcpip.h>
```

```
using namespace SCO_COMMON;
```

```

extern SharedMemory* o_SharedMemory;

CString CSessionRTP::GetIP()
{
    CString sIP;

    sIP = m_sIP;

    return sIP;
}

SOCKET* CSessionRTP::GetSocket()
{
    SOCKET* tmp_socket;

    tmp_socket = &socket_rtp;

    return tmp_socket;
}
/*
u_int CSessionRTP::GetPort(u_short p_iMode)
{
    u_int iPortRTP;

    switch (p_iMode)
    {
        case TRANSFER_MODE:
            iPortRTP = m_iTransferPortRTP;
            break;
        case RECEIVE_MODE:
            iPortRTP = CSessionRTP::GetPort();
            break;
        default:
            iPortRTP = 0;
            break;
    }

    return iPortRTP;
}*/

```



```

        if (!inet_pton(AF_INET, buff_ip,
&srv_address.sin_addr.s_addr))
            return false;
    }
    srv_address.sin_port = htons(P);

                                                                    // Номер порта

    if (bind(socket_rtp, (SOCKADDR*)&srv_address,
sizeof(SOCKADDR_IN)) == 0)
    {
        SetPort(P);
        break;
    }
    else
    {
        closesocket(socket_rtp);

        socket_rtp = 0;
        SetPort(0);
    }

}
if (socket_rtp == 0)    return false;

if (!bMultiCast) return true;

struct in_addr if_ip;

if (v_IP_SCO == "")
    if_ip.s_addr = INADDR_ANY;
                                                                    // Все адреса локального компьютера
else
{
    //if_ip.s_addr = inet_addr(buff_ip);
                                                                    // Все адреса локального компьютера
    if (!inet_pton(AF_INET, buff_ip, &if_ip.s_addr))
        return false;
}

```

```

        ret = setsockopt(socket_rtp, IPPROTO_IP, IP_MULTICAST_IF, (const
char*)(&if_ip), sizeof(if_ip));
        if (ret == -1)
        {
            TRACE("\r\nHandle error.");
        }

    return true;
}

void CSessionRTP::StartWriteRTP()
{

    char buff_ip[16], SendBuf[PACKET_RTP_SIZE_MAX];
    UINT nAmountReadData;

    SOCKADDR_IN s_address;

    time_t osBinaryTime;  // C run-time time (defined in <time.h>)

    // * заголовок RTP * //
    rtp_header* rtp_hdr;

    memset(SendBuf, 0, sizeof(SendBuf));
    memset(buff_ip, 0, sizeof(buff_ip));

    rtp_hdr = (rtp_header*)&SendBuf[0];

    // u_int iPortTr = GetPort(TRANSFER_MODE);
    m_sIP = L"192.168.1.12";
    StrToChar(&m_sIP, buff_ip, m_sIP.GetLength());

    time(&osBinaryTime);

    SendBuf[8] = (char)GetTickCount64();
    SendBuf[9] = (char)GetTickCount64();
    SendBuf[10] = (char)osBinaryTime;
    SendBuf[11] = (char)osBinaryTime;

    s_address.sin_family = AF_INET;
    s_address.sin_port = htons(55880);

```

```

    if(!inet_pton(AF_INET, buff_ip, &s_address.sin_addr.s_addr))
    {
        return;
    }

    // Заполняем поля заголовка
    SendBuf[0] = (char)0x80; // 128; (1000 0000 - в 2-й системе) //
Первый байт заполняется только версией протокола RTP
    SendBuf[1] = (char)0x8e; // 142; (1000 1110 - в 2-й системе) //
Указываем аудиокодек (маска в 0)

    SendBuf[12] = (char)0x00;
    SendBuf[13] = (char)0x00;
    SendBuf[14] = (char)0x00;
    SendBuf[15] = (char)0x00;

    std::chrono::milliseconds timespan(SCO_COMMON::TIME_OUT_THREAD_READ_TCP);

    u_long FileSize = 0;

    int Progress = 0;
    int SellProgress = 5;

    int PercentProgress = 0;

    int dat = 0;
    FileSize = o_Buffer.ReadSizeFile();

    while (g_bFlagWorkThread)
    {
        nAmountReadData =
o_Buffer.readFromBuf(&SendBuf[SIZE_RTP_HIADER], PACKET_RTP_SIZE_MAX -
SIZE_RTP_HIADER);

        Progress += nAmountReadData;

        if (nAmountReadData <= 0 || (nAmountReadData <
PACKET_RTP_SIZE_MAX - SIZE_RTP_HIADER && nAmountReadData == FileSize))
        {
            g_bFlagWorkThread = false;

```

```

    }

    if (sendto(*GetSocket(), SendBuf, PACKET_RTP_SIZE_MAX, 0,
(struct sockaddr*)&s_address, sizeof(s_address)) <= 0)
    {
        std::this_thread::sleep_for(timespan);
        continue;
    }
    else
    {
        PercentProgress = (Progress * 100) / FileSize;
        if ((PercentProgress > SellProgress) || (PercentProgress
== 100))
        {
            o_SharedMemory->FillPB(PercentProgress);
            SellProgress += 5;
        }
    }
}
return;
}

BOOL CSessionRTP::StartThreadWriteRTP()
{
    BOOL bRez = false;

    o_Buffer.StartThreadCheckBuffer();

    std::chrono::milliseconds timespan(500);
    std::this_thread::sleep_for(timespan);

    if (thread_write_rtp_socet == NULL && my_thread_guard_write_rtp_socet ==
NULL)
    {
        thread_write_rtp_socet = new
std::thread(&CSessionRTP::StartWriteRTP, this);
        my_thread_guard_write_rtp_socet = new
thread_guard(std::move(*thread_write_rtp_socet));

        bRez = true;
    }
    return bRez;
}

```

```
}
```

Листинг В.21 – Модуль SessionTCP.cpp

```
#include "pch.h"
//#include "afxdialogex.h"

#include "DiplomDlg.h"
#include "Common.h"
#include "SessionTCP.h"
#include "defines.h"
#include "SharedMemory.h"
#include "SessionRTP.h"

#include <chrono>
#include <stack>
#include <fstream>

using namespace SCO_COMMON;
using namespace std;

extern BOOL g_bFlagTCPManager;

u_short      g_iTimeWait = CONNECTION_WAITING_TIME;

extern CSessionRTP* rtp_session;
extern CDiplomDlg* o_DiplomDLG;
extern SharedMemory* o_SharedMemory;

static u_long iGlNumSession = 789000000;

//SOCKET CSessionTCP::socket_server, CSessionTCP::socket_client;

CSessionTCP::CSessionTCP(void) :
    ThreadReadMess(&CSessionTCP::ThreadReadTCP, this),
    my_thread_guard_read_tcp_socet(std::move(ThreadReadMess))
{
    TCPFileName = "";
}

CSessionTCP::~CSessionTCP()
{
}
```

```

void CSessionTCP::TaskManagerTCP()
{
    std::chrono::milliseconds timespan(SCO_COMMON::TIME_OUT_THREAD_READ_TCP); //
    or whatever

    clock_t iKey = 0;
    int ConnRequest = 0;

    UINT iexecutionStatus = 1;

    std::vector<std::string> svBufMsg;

    while (g_bFlagTCPManager)
    {
        iKey = 0;

        iexecutionStatus = CheckTaskTCP(iKey);

        switch (iexecutionStatus)
        {
            case SCO_COMMON::DATA_TRANSFER_REQUIRED:
                /*
                    {

                        g_oSM->ReadSharedMemory(iKey, svBufMsg);

                        SendCommand(svBufMsg);

                    }*/
                break;

            case CHECK_REQUEST_CLIENT:

                ConnRequest = CheckConnectionRequests();

                if (ConnRequest == 1)
                {
                    iKey = o_SharedMemory->IntoSharedMemoryInt(ConnRequest);

                    // o_SharedMemory->outSharedMemory(iKey);

                    o_DiplomDLG->PostMessage(WM_MSGTHREAD_ALL, iKey, 0);
                }
                break;

```

```

        default:
            break;
    }
    // Если удастся реализовать через лямбда функцию (см. конструктор) то
    эта функция будет не нужна.
    // Не, ее придется заменить на обработчик сообщений для прохода по map с
    лямбда-функциями для их вызова.
    // Помоему слишком замороченно.

    std::this_thread::sleep_for(timespan);
}
}

void CSessionTCP::ThreadReadTCP()
{
    std::chrono::milliseconds timespan(100);

    short int status = 0;

    char buff_prm[TCP_BUF_SIZE];

    fd_set s_get;
    timeval timeout;

    std::string sReadData;
    std::string sCommand;

    clock_t iKey = clock();

    for (short int i = 0; i < TCP_BUF_SIZE; i++) buff_prm[i] = 0;

    while (g_bFlagTCPManager)
    {
        if (socket_client != NULL)
        {
            FD_ZERO(&s_get);
            FD_SET(socket_client, &s_get);

            timeout.tv_sec = 10;
            timeout.tv_usec = 0;

```

```

status = select(socket_client, &s_get, NULL, &s_get, &timeout);

if (!FD_ISSET(socket_client, &s_get))
{
    std::this_thread::sleep_for(timespan);
    continue;
}
status = recv(socket_client, buff_prm, sizeof(buff_prm), 0);

string      RTP_Message = "";

if (status != SOCKET_ERROR)
{
    string Mess = "";

    CString sReceivedMessage = L"";
    CString TimeMess = L"";

    sReceivedMessage = buff_prm;

    CString a = sReceivedMessage;

    short iStartingPos = 0;
    short iEndPos = 0;

    if (sReceivedMessage != L"")
        while (iStartingPos >= 0)
        {

            iEndPos = sReceivedMessage.Find(L">") + 1;

            TimeMess = sReceivedMessage.Mid(iStartingPos, (iEndPos -
iStartingPos));

            if (TimeMess == L"") sReceivedMessage = "";

            sReceivedMessage.Delete(iStartingPos, iEndPos);

            Mess = to_std_string(TimeMess);
            RTP_Message = Mess;

```

```

        clock_t b = o_SharedMemory->IntoSharedMemoryString(Mess,
CONNECTION_REQUEST);

        o_DiplomDLG->PostMessage(INPUT_FORM, b,
CONNECTION_REQUEST);

        o_DiplomDLG->AnalysisReadMsg(Mess);

        iStartingPos = sReceivedMessage.Find(L"<");
        Mess = "";
    }
    sReceivedMessage = L"";

    sCommand = o_DiplomDLG->GetElementaryPartTeam(RTP_Message,
"command=");

    if (sCommand == "receive")
    {
        string Mess = "";

        rtp_session->StartThreadWriteRTP();
    }

    memset(buff_prm, 0x00, TCP_BUF_SIZE);
}
}
}

void CSessionTCP::ReproducingRequest(string i, string j)
{
    string sClient = "US_MUS.0" + i + "." + j;
    string str = "", Num_session = "";
    CString num_session = L"", num_port = L"";

    if ((iGlNumSession + 1) > 999999999) iGlNumSession = 789000000;

    ++iGlNumSession;
    num_session.Format(L"1501%d", iGlNumSession);

    Num_session = to_std_string(num_session);

    u_long FileSize = 0;

```

```

CFile myFile;
CFileException fileException;

CString cNameFile = L"";
cNameFile = TCPFileName.c_str();

myFile.Open(cNameFile, CFile::modeRead, &fileException);
FileSize = myFile.GetLength();

    str = "<REQUEST client=\"" + sClient + "\" command=\"receive\"
action=\"start\" session=\"" + Num_session + "\" url=\"rtp://192.168.1.12:55880
priority=\"255\" owner=\"server\" hash=\"9338\" result=\"accepted\" size=\"" +
to_string(FileSize) + "\"/>";

    send(socket_client, str.c_str(), (int)str.size(), 0);

}

void CSessionTCP::ThreadwriteTCP(clock_t numMsg, UINT iTypeCommand)
{
    string OutCommand = "";

    switch (iTypeCommand)
    {
    case TCP_SOCKET_CONNECT:
        OutCommand = o_SharedMemory->outSharedMemory(numMsg, iTypeCommand);

        send(socket_client, OutCommand.c_str(), (int)OutCommand.size(), 0);

        break;
    default:
        break;
    }
}

int CSessionTCP::InitSocket()
{
    int result = TCP_SOCKET_CREATE_SUCCESFULLY;

    WORD wVersionRequested = 0x0202;

```

```

WSADATA wsData;

int erStat = WSASStartup(wVersionRequested, &wsData);

if (erStat == 0)
    result = OnServer();

return result;
}

void CSessionTCP::SetPortTCP(u_short pPortTCP)
{
    CSingleLock syncTCP(&sem_tcp);
    syncTCP.Lock();

    m_iPortTCP = pPortTCP;

    syncTCP.Unlock();

    return;
}

int CSessionTCP::OnServer()
{
    int status = 0;

    for (u_short P = START_PORT_TCP; P < 0xFFFE; P += 2)
    {

        socket_server = socket(AF_INET, SOCK_STREAM, 0);

        if (socket_server == INVALID_SOCKET)
        {
            WSACleanup();

            socket_client = 0;

            return SOCKET_INITIALIZATION_ERROR;
        }

        /*          in_addr ip_to_num;

```

```

        int erStat_add = inet_pton(AF_INET, &ip_to_num);

        if (erStat_add <= 0)
            return ERROR_TRANSFER;
        */
    SOCKADDR_IN local_addr;

    local_addr.sin_family = AF_INET;                                //
Семейство адресов IPv4
    local_addr.sin_addr.s_addr = htonl(INADDR_ANY);                // Все
адреса локального компьютера /***** 15.03.2023 закоментил из-за кода ниже
    local_addr.sin_port = htons(P);                                //
Номер порта

    if (bind(socket_server, (sockaddr*)&local_addr, sizeof(local_addr)) ==
0)
    {
        SetPortTCP(P);
        break;
    }
    else
    {
        closesocket(socket_server);
        socket_server = 0;
    }
}

status = listen(socket_server, SOMAXCONN);
if (status != 0)
{
    closesocket(socket_server);
    socket_server = 0;

    WSACleanup();
    return ERROR_CONNECTONG_SOCKET_TO_SERVER;
}
return TCP_SOCKET_CREATE_SUCCESFULLY;
}

int CSessionTCP::CheckConnectionRequests()

```

```

{
    timeval t_timeval;

    fd_set s_set;

    FD_ZERO(&s_set);
    FD_SET(socket_server, &s_set);

    SOCKADDR_IN client_addr;

    int status, client_addr_size;

    clock_t numMsg = clock();

    t_timeval.tv_sec = g_iTimeWait;
    t_timeval.tv_usec = 0;

    if (g_iTimeWait != NULL)
        status = select(socket_server, &s_set, NULL, NULL, &t_timeval);
    else
        status = select(socket_server, &s_set, NULL, NULL, NULL);

    switch (status)
    {
    case SOCKET_ERROR:
        break;

    case 0:
        break;

    default:
        client_addr_size = sizeof(client_addr);

        if (status == 1)
        {
            socket_client = accept(socket_server, (sockaddr*)&client_addr,
&client_addr_size);

            CString l_sIPClient = o_DiplomDLG->GetIP_Client();

            std::string lstr;

```

```

        lstr = to_std_string(l_sIPClient);

        const char* cstr = lstr.c_str();
    }
}

return status;
}

void CSessionTCP::PutTaskTCP(clock_t p_KeyTask, UINT p_idTask)
{
    std::lock_guard<std::mutex> l_gMutexTCP(mutex_tcp);

    //Далее заполнение стека задач для tcp

    map_tcp.emplace(p_KeyTask, p_idTask);

};

UINT CSessionTCP::CheckTaskTCP(clock_t iKey)
{
    std::lock_guard<std::mutex> l_gMutexTCP(mutex_tcp);

    // Проверка на наличие задачи для tcp
    // Первый пришел, первый ушел (LIFO)

    UINT uKey = 0;

    if (!map_tcp.empty())
    {
        auto Task = map_tcp.rbegin();

        iKey = Task->first;

        uKey = Task->second;

        map_tcp.erase(iKey);
    }

    return uKey;
}

```

```
}
```

Листинг В.22 – Модуль SharedMemory.cpp

```
#include "pch.h"

#include "DiplomDlg.h"
#include "SharedMemory.h"
#include "Common.h"

using namespace SCO_COMMON;

extern CDiplomDlg* o_DiplomDLG;

SharedMemory::SharedMemory()
{}

SharedMemory::~~SharedMemory()
{}

clock_t SharedMemory::IntoSharedMemoryInt(int status)
{
    std::lock_guard<std::mutex> l_gMutexSM(mutex_SharedMem);

    static unsigned long counter = 0;
    clock_t MessageKey = clock() + counter;

    IntoStat.status = status;

    m_SharedMemory.emplace(MessageKey, IntoStat);

    counter++;

    return MessageKey;
}

std::string SharedMemory::outSharedMemory(clock_t MassKey, UINT iKey)
{
    std::lock_guard<std::mutex> l_gMutexSM(mutex_SharedMem);

    std::string OutMess = "";

    auto it = m_SharedMemory.find(MassKey);
```

```

    if (it != m_SharedMemory.end())
    {
        std::vector<std::string> VecPeren;

        VecPeren = it->second.Message;

        OutMess = VecPeren[0];

        m_SharedMemory.erase(MassKey);
    }

    return OutMess;
}

clock_t SharedMemory::IntoSharedMemoryString(std::string Mess, UINT m_key)
{
    std::lock_guard<std::mutex> l_gMutexSM(mutex_SharedMem);

    static unsigned long counter = 0;
    clock_t MessageKey = clock() + counter;

    STMass LocalIntoStat;

    LocalIntoStat.Message.push_back(Mess);
    LocalIntoStat.m_Kei = m_key;

    m_SharedMemory.emplace(MessageKey, LocalIntoStat);

    counter++;

    return MessageKey;
}

void SharedMemory::FillPB(int Progress)
{
    o_DiplomDLG->ProgressBarFilling(Progress);
}

```