

СОДЕРЖАНИЕ

Введение.....	5
1 Структуры и файлы.....	6
1.1 Файл City.h.....	6
1.2 Файл Conversation.h	6
1.3 Файл Menu.h	6
1.4 Файл main.cpp.....	6
1.5 Структура City.....	6
1.6 Структура Conversation	7
2 Алгоритм сортировки	8
3 Алгоритмы поиска	10
4 Пользовательские функции.....	11
4.1 Файл City.h.....	11
4.2 Файл Conversation.h	11
4.3 Файл Menu.h	12
5 Описание работы программы	14
Заключение	19
Список использованных источников	20
Приложение А. Листинг кода	21
Приложение Б. Схема алгоритма	31
Приложение В. Схема алгоритма	32

ВВЕДЕНИЕ

Целью данного проекта является разработка программы на языке программирования C++, которая обеспечивает обработку файла данных, содержащего информацию о разговорах на междугородной автоматической телефонной станции (АТС). Программа предоставляет пользователю возможность выполнять различные операции над данными, включая создание, просмотр, добавление, удаление, редактирование, линейный поиск и сортировку.

Структура данных в файле описывает информацию о разговорах и содержит следующие поля: дата разговора, код и название города, время разговора, тариф, номер телефона в этом городе и номер телефона абонента. Каждая запись представляет собой отдельную структуру.

Основными задачами, которые необходимо решить для достижения цели работы, являются:

1. Создание программы обработки файла данных, включающее чтение и запись информации из/в текстовый файл.
2. Реализация функционала просмотра данных о разговорах, позволяющего пользователю получить общую информацию о разговорах с каждым городом, включая общее время разговоров и сумму.
3. Возможность добавления, удаления и редактирования записей о разговорах, позволяющая пользователю вносить изменения в файл данных.
4. Реализация линейного поиска для быстрого нахождения информации о разговорах с заданным городом.
5. Реализация сортировки листа для удобства работы с данными.

Выбранная тема обработки данных разговоров на междугородной АТС является актуальной, так как обеспечивает эффективное управление информацией о разговорах и позволяет получать ценные статистические данные о времени разговоров с каждым городом. Это может быть полезно для оптимизации использования телефонных ресурсов и контроля затрат на связь.

В итоге, разработанная программа обеспечит удобный и эффективный способ работы с данными о разговорах на междугородной АТС, позволяя пользователю выполнять различные операции обработки и получать необходимую информацию о разговорах с каждым городом.

1 СТРУКТУРЫ И ФАЙЛЫ

1.1 Файл City.h

Файл City.h содержит реализацию функций, связанных с обработкой городов и статистики о разговорах. Он предоставляет функционал для сбора информации о разговорах в разных городах, вывода этой информации на экран, сортировки городов по общему времени разговоров и поиска городов по заданному значению времени разговора. Файл обеспечивает удобное управление статистикой и предоставляет возможность анализировать разговоры в разных городах на основе собранных данных.

1.2 Файл Conversation.h

Файл Conversation.h содержит реализацию функций, связанных с вводом, выводом и обработкой информации о разговорах. Он предоставляет функционал для считывания разговоров от пользователя, записи их в файл, чтения разговоров из файла, вывода информации о разговорах в удобном формате и удаления содержимого файла. Файл обеспечивает удобный интерфейс для работы с данными о разговорах и управления ими.

1.3 Файл Menu.h

Файл Menu.h отвечает за реализацию меню и обработку пользовательского ввода. Он содержит функции, которые выводят меню с доступными командами, добавляют, удаляют, изменяют и выводят разговоры, а также очищают список разговоров. Файл обеспечивает удобный интерфейс для взаимодействия с пользователем, управления разговорами и выполнения операций на основе выбора пользователя.

1.4 Файл main.cpp

Файл main.cpp является точкой входа в программу. Он подключает заголовочный файл Menu.h, устанавливает локализацию для поддержки русского языка, считывает разговоры из файла в список, запускает функцию Run() из файла Menu.h и возвращает 0 в качестве кода завершения программы. Основная цель файла main.cpp - инициализировать и запустить программу, используя функционал, предоставленный в файле Menu.h.

1.5 Структура City

```
struct City {  
    string name;  
    double total_talk_time = 0.0;  
    double sum_money = 0.0;};
```

Структура `City` состоит из трех членов:

1. `name` - строковая переменная, которая представляет название города.
2. `total_talk_time` - переменная типа `double`, которая хранит общее время разговоров, связанных с данным городом. Значение по умолчанию равно 0.0.
3. `sum_money` - переменная типа `double`, которая хранит сумму денег, связанных с данным городом (например, стоимость разговоров). Значение по умолчанию равно 0.0.

Структура `City` предназначена для хранения информации о городах, включая их название, общее время разговоров и сумму денег.

1.6 Структура Conversation

```
struct Conversation {  
    string date;  
    string city_code;  
    string city_name;  
    double duration = 0.0;  
    double tariff = 0.0;  
    string phone_number;  
    string subscriber_number;};
```

Структура `Conversation` представляет информацию о разговоре и включает следующие члены:

1. `date` - строковая переменная, которая содержит дату разговора.
2. `city_code` - строковая переменная, которая представляет код города, связанного с разговором.
3. `city_name` - строковая переменная, которая хранит название города, связанного с разговором.
4. `duration` - переменная типа `double`, которая содержит длительность разговора в минутах. Значение по умолчанию равно 0.0.
5. `tariff` - переменная типа `double`, которая представляет тарифную ставку для данного разговора. Значение по умолчанию равно 0.0.
6. `phone_number` - строковая переменная, которая хранит номер телефона, с которого был совершен разговор.
7. `subscriber_number` - строковая переменная, которая содержит номер абонента, с которым был проведен разговор.

Структура `Conversation` используется для хранения информации о конкретном разговоре, включая дату, город, длительность, тариф, номер телефона и номер абонента.

2 АЛГОРИТМ СОРТИРОВКИ

Этот алгоритм представляет собой функцию `'BubbleSort'`, которая выполняет сортировку городов в списке `'list_cities'` по возрастанию значения общего времени разговора с использованием алгоритма пузырьковой сортировки.

Вначале функция очищает экран с помощью команды `'system("cls")'`, чтобы обеспечить чистый вывод результатов.

Затем происходит проверка, пустой ли список `'list_cities'`. Если список пуст, выводится сообщение "Нет городов для сортировки".

Если список не пуст, то создается локальная переменная `'swapped'` типа `'bool'`, инициализированная значением `'true'`. Этот флаг указывает наличие обменов элементов в процессе сортировки.

Затем следует цикл `'while'`, который выполняется до тех пор, пока переменная `'swapped'` равна `'true'`. В начале каждой итерации цикла флаг `'swapped'` сбрасывается в `'false'`, чтобы отслеживать, происходили ли обмены элементов в текущей итерации.

Внутри цикла определены итераторы `'it'` и `'nextIt'`, указывающие на текущий элемент и следующий элемент списка соответственно.

Затем следует цикл `'while'`, который выполняется до тех пор, пока итератор `'nextIt'` не достигнет конца списка `'list_cities'`. Внутри этого цикла выполняются следующие действия:

- Сравнивается значение общего времени разговора текущего элемента `'(*it)'` с значением общего времени разговора следующего элемента `'(*nextIt)'`.
- Если значение общего времени разговора текущего элемента больше значения общего времени разговора следующего элемента, то выполняется обмен элементами с помощью функции `'iter_swap(it, nextIt)'`. Таким образом, более маленький элемент перемещается перед более большим элементом.
- Устанавливается флаг `'swapped'` в `'true'` для указания наличия обмена элементов.

После завершения внутреннего цикла `'while'`, происходит проверка флага `'swapped'`. Если флаг `'swapped'` равен `'false'`, это означает, что в текущей итерации не было произведено ни одного обмена элементов, и список уже отсортирован. В таком случае сортировка завершается.

После завершения сортировки вызывается функция `'print_cities'`, которая выводит отсортированный список городов.

Таким образом, данная функция выполняет пузырьковую сортировку городов в списке `'list_cities'` по возрастанию значения общего времени разговора.

3 АЛГОРИТМЫ ПОИСКА

Этот алгоритм представляет собой функцию ``SearchValue``, которая выполняет поиск городов в списке ``list_cities`` по заданному значению общего времени разговора.

Вначале функция очищает экран с помощью команды ``system("cls")``, чтобы обеспечить чистый вывод результатов.

Затем происходит проверка, пустой ли список ``list_cities``. Если список пуст, выводится сообщение "Нет городов для сортировки".

Если список не пуст, то создается локальная переменная ``flag`` типа ``bool``, инициализированная значением ``true``. Также объявляются переменные ``key`` типа ``double`` и ``i`` типа ``int``. Переменная ``key`` будет использоваться для хранения введенного пользователем значения общего времени разговора, а переменная ``i`` будет использоваться для нумерации найденных городов.

Затем пользователю предлагается ввести значение общего времени разговора с помощью команды ``cout``, а значение сохраняется в переменной ``key`` с помощью команды ``cin``.

Далее следует цикл ``for``, который проходит по каждому элементу в списке ``list_cities``. Внутри цикла проверяется, равно ли значение общего времени разговора элемента списка ``key``. Если значения равны, то выполняются следующие действия:

- Если переменная ``flag`` равна ``true``, то вызывается функция ``shapka_city()``, которая, вероятно, выводит заголовок таблицы с информацией о городах. Затем переменная ``flag`` устанавливается в ``false``, чтобы заголовок был выведен только один раз.

- Вызывается функция ``print_city``, которая, вероятно, выводит информацию о городе. Передается текущий элемент списка ``*it`` и значение ``i`` для нумерации.

- Значение ``i`` увеличивается на 1.

Если после выполнения цикла значение ``flag`` остается ``true``, это означает, что не было найдено ни одного города с заданным значением общего времени разговора. В таком случае выводится сообщение "Нет таких городов".

Таким образом, данная функция выполняет поиск городов с заданным значением общего времени разговора и выводит информацию о найденных городах, если они существуют.

4 ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ

4.1 Файл City.h

Функция `collect_city(Conversation conversation)` отвечает за сбор статистики о разговорах в городах. Она принимает объект типа `Conversation`, который содержит информацию о разговоре, включая название города, длительность разговора и тариф. Функция проверяет, существует ли уже город в списке `list_cities`. Если город уже присутствует, то обновляются суммарное время разговоров и общая сумма денег в этом городе. Если город отсутствует в списке, то создается новый объект `City` и добавляется в список `list_cities`, с учетом информации о разговоре.

Функция `collect_cities(list<Conversation>& List)` используется для сбора статистики о разговорах во всех городах из переданного списка `List`. Она очищает список `list_cities` и затем вызывает функцию `collect_city` для каждого элемента списка `List`.

Функция `print_cities()` выводит информацию о городах на экран. Если список `list_cities` пуст, то выводится сообщение "Нет городов для вывода". В противном случае, функция выводит заголовок таблицы с названиями столбцов ("Номер", "Название города", "Общее время разговоров", "Общая сумма") и затем выводит информацию о каждом городе в виде строки таблицы.

Функция `print_city(City city, int number)` используется для вывода информации о конкретном городе. Она принимает объект `City` и номер города в порядке вывода. Функция выводит информацию о городе, включая его номер, название, общее время разговоров и общую сумму денег.

Функция `shapka_city()` выводит заголовок таблицы с названиями столбцов для функции `print_cities()`. Она выводит строку разделителей и названия столбцов ("Номер", "Название города", "Общее время разговоров", "Общая сумма").

4.2 Файл Conversation.h

Функция ``read_conversation_from_user()`` считывает информацию о разговоре с пользователем. Она запрашивает у пользователя дату, код города, название города, продолжительность разговора, тариф, номер телефона и номер абонента. Затем создает объект ``Conversation`` и возвращает его.

Функция ``write_conversation_to_file(const Conversation& conversation)`` записывает информацию о разговоре в файл. Она открывает файл в режиме

добавления ('ios::app') и записывает каждое поле разговора в отдельную строку, разделяя их символом табуляции. Затем файл закрывается.

Функция `read_conversation_from_file(list<Conversation>& List)` считывает информацию о разговорах из файла и сохраняет ее в список `List`. Она открывает файл и читает строки по одной. Каждая строка разделяется на поля, которые затем присваиваются соответствующим полям объекта `Conversation`. Объект добавляется в список. Функция завершается, когда все строки файла прочитаны.

Функция `delete_file_conversation()` удаляет содержимое файла с информацией о разговорах. Она открывает файл в режиме перезаписи ('ios::trunc') и затем закрывает его.

Функция `print_conversation(const Conversation& conversation, int number)` выводит информацию о конкретном разговоре. Она принимает объект `Conversation` и номер разговора в порядке вывода. Функция форматирует вывод в виде таблицы с помощью функции `setw()` и выводит каждое поле разговора в отдельной колонке.

Функция `shapka_conversation()` выводит заголовок таблицы с названиями столбцов для функции `print_conversation()`. Она выводит строку разделителей и названия столбцов, соответствующие полям объекта `Conversation`.

4.3 Файл Menu.h

Функция `Run()` является основной функцией, которая запускает меню и организует цикл обработки пользовательских команд. Внутри цикла она выводит меню, считывает выбор пользователя и вызывает соответствующую функцию в зависимости от выбора.

Функция `print_menu()` выводит на экран меню с доступными командами. Каждая команда имеет свой номер и описание.

Функция `add_conversation()` добавляет новый разговор. Она считывает информацию о разговоре от пользователя, записывает его в файл и добавляет в список разговоров.

Функция `delete_conversation()` удаляет разговор по выбранному пользователем номеру. Она выводит список разговоров, считывает номер для удаления, проверяет его валидность и удаляет соответствующий разговор из списка и файла.

Функция `change_conversation()` изменяет разговор по выбранному пользователем номеру. Она выводит список разговоров, считывает номер для изменения, проверяет его валидность, считывает новую информацию о разговоре от пользователя, обновляет разговор в списке и файле.

Функция `print_conversations()` выводит список всех разговоров. Если список пуст, выводится сообщение об отсутствии разговоров. Иначе, она выводит заголовок таблицы и каждый разговор в форматированном виде.

Функция `clear_conversations()` очищает список разговоров. Если список пуст, выводится сообщение об отсутствии разговоров. Иначе, она удаляет содержимое файла и очищает список разговоров.

Функция `write_list_conversation_to_file()` записывает все разговоры из списка в файл. Она сначала очищает файл, а затем записывает каждый разговор из списка в отдельной строке.

Функция `read_conversation_from_file_to_list()` считывает разговоры из файла и сохраняет их в список разговоров.

5 ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

В результате проделанной работы была создана программа для обработки файла данных, содержащих информацию о разговорах на междугородной автоматической телефонной станции (АТС).

Далее приведен пример работы программы.

После запуска программы появляется главное меню (рис 5.1). В нем пользователь получает возможность выполнить одну из 9-и функций.

```
Чтение из файла успешно выполнено.

----- Menu -----
1. добавить разговор
2. удалить разговор
3. вывести разговоры
4. изменить разговор
5. очистить разговоры
6. вывести города
7. поиск города по ключу
8. отсортированные города по ключу
9. выйти из программы
-----
*ключ – общее время разговора городов
|
```

Рисунок 5.1 – Меню программы

Рассмотрим подробнее функционал программы, предоставленный пользователю:

- 1) При нажатии «1» нам будет предложено заполнить объект структуры звонка, который добавиться в лист.

```

Введите дату: 2021-05-12
Введите код города: 456
Введите название города: Минск
Введите продолжительность разговора: 15.5
Введите тариф: 0.35
Введите номер телефона: 123456789
Введите номер абонента: 987654321
разговор добавлен успешно

```

Рисунок 5.2 – ввод объекта структуры звонка

- 2) При нажатии «2» вы сможете воспользоваться удалением объекта структуры звонка из листа, выбрав его номер (рис. 5.3).

13	2021-05-17	987	Челябинск	9.8	0.98	987654321	123456789	
14	2021-05-18	234	Омск	7.1	0.71	123456789	987654321	
15	2021-05-19	567	Самара	3.9	0.39	987654321	123456789	
16	2021-05-20	890	Ростов-на-Дону	2.6	0.26	123456789	987654321	
17	2021-05-21	432	Уфа	1.8	0.18	987654321	123456789	
Введите номер звонка для удаления: 17								
Удаление прошло успешно								

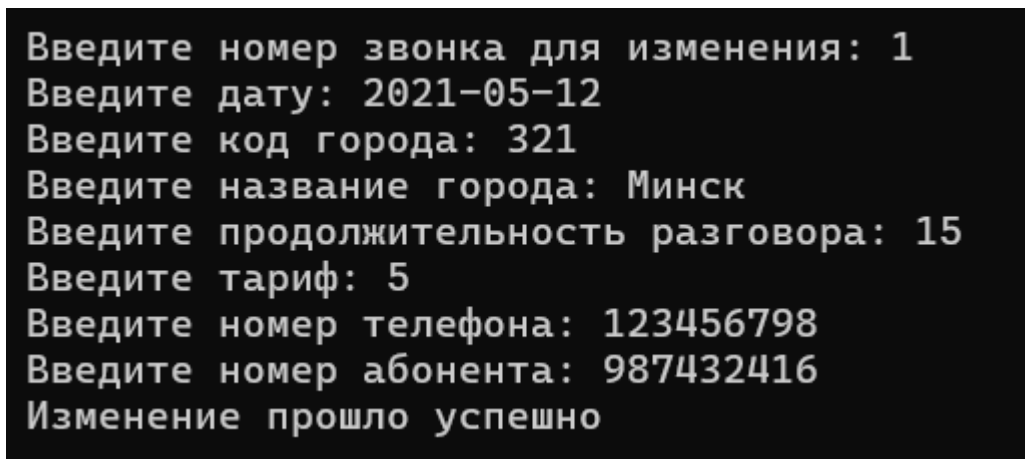
Рисунок 5.3 – Пример выполнения удаления

- 3) При нажатии «3» вы сможете вывести все объекты структуры звонков(рис.5.4.).

Номер	Дата	Код города	Название города	Время разговора	Тариф	Н/Т в городе	Н/Т абонента	
1	2021-05-12	123	Москва	10.5	1.05	123456789	987654321	
2	2021-05-12	123	Минск	15.5	0.35	123456789	987654321	
3	2021-05-12	123	Минск	15.5	4.56	123456789	987654321	
4	2021-05-12	123	Москва	10.5	0.25	123456789	987654321	
5	2021-05-12	123	Москва	10.5	7.89	123456789	987654321	
6	2021-05-12	123	Москва	10.5	5.33	123456789	987654321	
7	2021-05-12	123	Москва	10.5	0.05	123456789	987654321	
8	2021-05-12	123	Москва	10.5	1.05	123456789	987654321	
9	2021-05-13	456	Санкт-Петербург	8.2	0.82	987654321	123456789	
10	2021-05-14	789	Новосибирск	5.5	0.55	123456789	987654321	
11	2021-05-15	321	Екатеринбург	6.7	0.67	987654321	123456789	
12	2021-05-16	654	Казань	4.3	0.43	123456789	987654321	
13	2021-05-17	987	Челябинск	9.8	0.98	987654321	123456789	
14	2021-05-18	234	Омск	7.1	0.71	123456789	987654321	
15	2021-05-19	567	Самара	3.9	0.39	987654321	123456789	
16	2021-05-20	890	Ростов-на-Дону	2.6	0.26	123456789	987654321	

Рисунок 5.4 – Пример выполнения вывода объектов структур звонков

4) При нажатии «4» вы сможете изменить структуру в листе(рис.5.5).



```
Введите номер звонка для изменения: 1
Введите дату: 2021-05-12
Введите код города: 321
Введите название города: Минск
Введите продолжительность разговора: 15
Введите тариф: 5
Введите номер телефона: 123456798
Введите номер абонента: 987432416
Изменение прошло успешно
```

Рисунок 5.5 – изменение поля выбранной структуры

5) При нажатии «5» вы сможете очистить лист структур звонков. (рис. 5.6).



```
Звонки очищены
```

Рисунок 5.6 –очистка листа звонков.

6) При нажатии «6» вы сможете вывести собранную информацию о городах. (рис. 5.7).

Номер	Название города	Общее время разговоров	Общая сумма
1	Москва	63	164.01
2	Минск	31	76.105
3	Санкт-Петербург	8.2	6.724
4	Новосибирск	5.5	3.025
5	Екатеринбург	6.7	4.489
6	Казань	4.3	1.849
7	Челябинск	9.8	9.604
8	Омск	7.1	5.041
9	Самара	3.9	1.521
10	Ростов-на-Дону	2.6	0.676
11	Уфа	1.8	0.324

Рисунок 5.7 – вывод собранную информацию о городах.

- 7) При нажатии «7» вы сможете найти город по ключу (рис. 5.8).

Введите общее время разговора: 31			
Номер	Название города	Общее время разговоров	Общая сумма
1	Минск	31	76.105

Рисунок 5.7 – поиск объекта по ключу.

- 8) При нажатии «8» вы выведете отсортированную информацию о городах (рис. 5.9).

Номер	Название города	Общее время разговоров	Общая сумма
1	Уфа	1.8	0.324
2	Ростов-на-Дону	2.6	0.676
3	Самара	3.9	1.521
4	Казань	4.3	1.849
5	Новосибирск	5.5	3.025
6	Екатеринбург	6.7	4.489
7	Омск	7.1	5.041
8	Санкт-Петербург	8.2	6.724
9	Челябинск	9.8	9.604
10	Минск	31	76.105
11	Москва	63	164.01

Рисунок 5.9 –вывод отсортированных городов.

9) При нажатии «9» вы выйдете из программы (рис. 5.10).

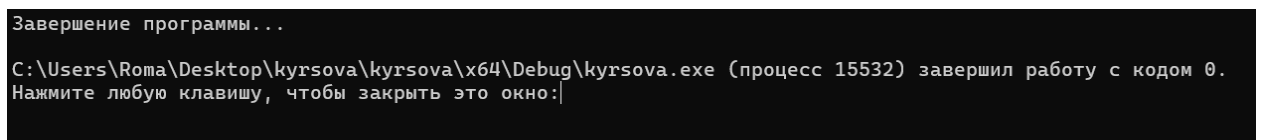


Рисунок 5.10 – выход из программы.

ЗАКЛЮЧЕНИЕ

В ходе данного проекта была разработана программа на языке программирования C++, предназначенная для обработки файла данных, содержащих информацию о разговорах на междугородной автоматической телефонной станции (АТС). Программа успешно реализует основные задачи, такие как создание, просмотр, добавление, удаление, редактирование, линейный поиск и сортировка данных.

В результате создания программы были достигнуты следующие результаты:

Разработана функциональная программа, позволяющая пользователю удобно и эффективно работать с данными о разговорах на междугородной АТС. Программа обладает интуитивно понятным интерфейсом, который обеспечивает выполнение всех необходимых операций над данными.

Реализован функционал просмотра данных о разговорах, позволяющий получать общую информацию о разговорах с каждым городом, включая общее время разговоров и сумму. Это позволяет пользователям получать ценные статистические данные и контролировать затраты на связь.

Реализованы операции добавления, удаления и редактирования записей о разговорах, что позволяет пользователям вносить изменения в файл данных и актуализировать информацию.

Реализован линейный поиск, который обеспечивает быстрое нахождение информации о разговорах с заданным городом. Это позволяет пользователям удобно находить необходимые данные по запросу.

Реализована сортировка массива (или файла), что облегчает работу с данными и обеспечивает их удобное представление.

В процессе разработки программы были проведены тесты, которые подтвердили корректность функционирования всех реализованных операций. Программа успешно обрабатывает файл данных и выполняет требуемые операции над ними.

В заключение, разработанная программа представляет собой эффективное решение для обработки данных разговоров на междугородной АТС. Она обладает широким спектром функциональности и обеспечивает удобный интерфейс для работы с данными. Программа успешно решает поставленные задачи и может быть использована в реальных условиях для управления информацией о разговорах и анализа статистики связи.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Герберт Шилдт. С++ для начинающих. Шаг за шагом;
- [2] CyberForum [Электронный ресурс] – Форум программистов С++. – Электронные данные. – Режим доступа: <http://www.cyberforum.ru/cpp/>;
- [3] CPP-REFERENCE [Электронный ресурс]. – Режим доступа: <https://en.cppreference.com/w/>.

ПРИЛОЖЕНИЕ А. Листинг кода

```
1.  //запуск программы
2.  //main.cpp
3.  #include "Menu.h"
4.
5.  int main() {
6.      setlocale(LC_ALL, "Russian");
7.      read_conversation_from_file_to_list();
8.      Run();
9.      return 0;
10. }
11. //City.h
12. #pragma once
13. #include "Conversation.h"
14.
15. struct City {
16.     string name;
17.     double total_talk_time = 0.0;
18.     double sum_money = 0.0;
19. };
20.
21. void collect_city(Conversation conversation);
22. void collect_cities(list<Conversation>& List);
23. void print_cities();
24. void print_city(City city, int number);
25. void shapka_city();
26. void BubbleSort();
27. void SearchValue();
28. //Conversation.h
29. #pragma once
30. #include <iostream>
31. #include <fstream>
32. #include <sstream>
33. #include <string>
34. #include <locale>
35. #include <iomanip>
36. #include <list>
37. #include <algorithm>
38.
39.
40. using namespace std;
41.
42. struct Conversation {
43.     string date;
44.     string city_code;
45.     string city_name;
46.     double duration = 0.0;
47.     double tariff = 0.0;
48.     string phone_number;
49.     string subscriber_number;
50. };
51.
52.
53. Conversation read_conversation_from_user();
```

Продолжение приложения А

```
54. void write_conversation_to_file(const Conversation&
    conversation);
55. void read_conversation_from_file(list<Conversation>& List);
56. void delete_file_conversation();
57. void print_conversation(const Conversation& conversation,
    int number);
58. void shapka_conversation();
59. //Menu.h
60. #pragma once
61. #include "City.h"
62.
63.     void Run();
64.     void add_conversation();
65.     void delete_conversation();
66.     void change_conversation();
67.     void print_menu();
68.     void print_conversations();
69.     void clear_conversations();
70.     void read_conversation_from_file_to_list();
71.     void write_list_conversation_to_file();
72. //City.cpp
73. #include "City.h"
74.
75. list<City> list_cities;
76.
77. void collect_city(Conversation conversation)
78. {
79.     bool flag = true;
80.     for (list<City>::iterator it = list_cities.begin(); it
        != list_cities.end(); ++it)
81.     {
82.         if ((*it).name == conversation.city_name)
83.         {
84.             flag = false;
85.             (*it).total_talk_time += conversation.duration;
86.             (*it).sum_money += conversation.duration *
conversation.tariff;
87.             break;
88.         }
89.     }
90.     if (flag)
91.     {
92.         City city;
93.         city.name = conversation.city_name;
94.         city.total_talk_time = conversation.duration;
95.         city.sum_money = conversation.tariff *
conversation.duration;
96.         list_cities.push_back(city);
97.     }
98. }
99.
100. void collect_cities(list<Conversation>& List)
101. {
102.     list_cities.clear();
```

Продолжение приложения А

```
103.     for (list<Conversation>::iterator it = List.begin(); it
      != List.end(); ++it)
104.     {
105.         collect_city(*it);
106.     }
107. }
108.
109. void print_cities()
110. {
111.     system("cls");
112.     if (list_cities.empty())
113.     {
114.         cout << "Нет городов для вывода" << endl;
115.     }
116.     else {
117.         int i = 1;
118.         shapka_city();
119.
120.         for (list<City>::const_iterator it =
list_cities.cbegin(); it != list_cities.cend(); ++it)
121.         {
122.             print_city(*it, i);
123.             i++;
124.         }
125.     }
126.
127. }
128.
129. void print_city(City city, int number)
130. {
131.     cout << '|' << left << setw(5) << number << '|';
132.     cout << left << setw(15) << city.name << '|';
133.     cout << left << setw(25) << city.total_talk_time << '|';
134.     cout << left << setw(15) << city.sum_money << '|' <<
endl;
135.     cout << string(65, '-') << endl;
136. }
137.
138. void shapka_city()
139. {
140.     cout << string(65, '-') << endl;
141.     cout << '|' << left << setw(5) << "Номер" << '|';
142.     cout << left << setw(15) << "Название города" << '|';
143.     cout << left << setw(25) << "Общее время разговоров" <<
'|';
144.     cout << left << setw(15) << "Общая сумма" << '|' <<
endl;
145.     cout << string(65, '-') << endl;
146. }
147.
148. void BubbleSort()
149. {
150.     system("cls");
151.     if (list_cities.empty())
152.     {
```

Продолжение приложения А

```
153.         cout << "Нет городов для сортировки" << endl;
154.     }
155.     else
156.     {
157.         bool swapped = true; // Флаг, указывающий наличие
обменов
158.         while (swapped)
159.         {
160.             swapped = false; // Сбрасываем флаг перед каждой
итерацией
161.
162.             // Проходим по элементам списка
163.             list<City>::iterator it = list_cities.begin();
164.             list<City>::iterator nextIt = next(it);
165.
166.             while (nextIt != list_cities.end())
167.             {
168.                 // Если текущий элемент больше следующего
элемента
169.                 if ((*it).total_talk_time >
(*nextIt).total_talk_time)
170.                 {
171.                     // Меняем местами элементы
172.                     iter_swap(it, nextIt);
173.                     swapped = true; // Устанавливаем флаг
обмена
174.                 }
175.
176.                 // Переходим к следующим элементам
177.                 ++it;
178.                 ++nextIt;
179.             }
180.         }
181.     }
182.     print_cities();
183. }
184.
185. void SearchValue()
186. {
187.     system("cls");
188.     if (list_cities.empty())
189.     {
190.         cout << "Нет городов для сортировки" << endl;
191.     }
192.     else
193.     {
194.         bool flag = true;
195.         double key;
196.         int i = 1;
197.         cout << "Введите общее время разговора: ";
198.         cin >> key;
199.         for (typename list<City>::iterator it =
list_cities.begin(); it != list_cities.end(); ++it)
200.         {
```

Продолжение приложения А

```
201.     if ((*it).total_talk_time == key)
202.     {
203.         if (flag) { shapka_city(); flag = false; }
204.         print_city(*it, i);
205.         i++;
206.     }
207.     }
208.     if (flag) cout << "There are no elements" << endl;
209. }
210. }
211. //Conversation.cpp
212. #include "Conversation.h"
213.
214. string filename = "conversations.txt";
215.
216. Conversation read_conversation_from_user() {
217.     Conversation conversation;
218.
219.     cin.ignore(); // Очистка символа новой строки из буфера
    ввода
220.     cout << "Введите дату: ";
221.     getline(cin, conversation.date);
222.
223.     cout << "Введите код города: ";
224.     getline(cin, conversation.city_code);
225.
226.     cout << "Введите название города: ";
227.     getline(cin, conversation.city_name);
228.
229.     cout << "Введите продолжительность разговора: ";
230.     cin >> conversation.duration;
231.     cin.ignore(); // Очистка символа новой строки из буфера
    ввода
232.
233.     cout << "Введите тариф: ";
234.     cin >> conversation.tariff;
235.     cin.ignore(); // Очистка символа новой строки из буфера
    ввода
236.
237.     cout << "Введите номер телефона: ";
238.     getline(cin, conversation.phone_number);
239.
240.     cout << "Введите номер абонента: ";
241.     getline(cin, conversation.subscriber_number);
242.
243.     return conversation;
244. }
245.
246.
247. void write_conversation_to_file(const Conversation&
    conversation) {
248.     ofstream file(filename, ios::app); // Открываем файл в
    режиме добавления (append)
249.     if (file.is_open()) {
250.         file << conversation.date << "\t";
```

Продолжение приложения А

```
251.         file << conversation.city_code << "\t";
252.         file << conversation.city_name << "\t";
253.         file << conversation.duration << "\t";
254.         file << conversation.tariff << "\t";
255.         file << conversation.phone_number << "\t";
256.         file << conversation.subscriber_number << "\n";
257.         file.close();
258.     }
259.     else {
260.         cout << "Не удалось открыть файл.\n";
261.     }
262. }
263.
264. void read_conversation_from_file(list<Conversation>& List) {
265.     Conversation conversation;
266.     ifstream file(filename);
267.     if (file.is_open()) {
268.         string line;
269.         while (true)
270.         {
271.             if (getline(file, line)) {
272.                 // Создаем строковый поток из прочитанной
строки
273.                 istringstream iss(line);
274.
275.                 getline(iss, conversation.date, '\t');
276.                 getline(iss, conversation.city_code, '\t');
277.                 getline(iss, conversation.city_name, '\t');
278.                 iss >> conversation.duration;
279.                 iss.ignore(); // Пропускаем символ табуляции
280.                 iss >> conversation.tariff;
281.                 iss.ignore(); // Пропускаем символ табуляции
282.                 getline(iss, conversation.phone_number,
'\t');
283.                 getline(iss, conversation.subscriber_number,
'\t');
284.                 List.push_back(conversation);
285.             }
286.             else { break; }
287.         }
288.         file.close();
289.         cout << "Чтение из файла успешно выполнено.\n";
290.     }
291.     else {
292.         cout << "Не удалось открыть файл.\n";
293.     }
294. }
295. }
296. void delete_file_conversation() {
297.     ofstream file(filename, ios::trunc);
298.     if (!file.is_open()) {
299.         cout << "Error with rewrite File";
300.     }
301.     file.close();
302. }
```

Продолжение приложения А

```
303.
304. void print_conversation(const Conversation& conversation,
    int number) {
305.     cout << '|' << left << setw(5) << number << '|';
306.     cout << left << setw(15) << conversation.date << '|';
307.     cout << left << setw(15) << conversation.city_code <<
    '|';
308.     cout << left << setw(15) << conversation.city_name <<
    '|';
309.     cout << left << setw(15) << conversation.duration <<
    '|';
310.     cout << left << setw(15) << conversation.tariff << '|';
311.     cout << left << setw(15) << conversation.phone_number <<
    '|';
312.     cout << left << setw(15) <<
    conversation.subscriber_number << '|' << endl;
313.     cout << string(119, '-') << endl;
314. }
315.
316. void shapka_conversation() {
317.     cout << string(119, '-') << endl;
318.     cout << '|' << left << setw(5) << "Номер" << '|';
319.     cout << left << setw(15) << "Дата" << '|';
320.     cout << left << setw(15) << "Код города" << '|';
321.     cout << left << setw(15) << "Название города" << '|';
322.     cout << left << setw(15) << "Время разговора" << '|';
323.     cout << left << setw(15) << "Тариф" << '|';
324.     cout << left << setw(15) << "Н/Т в городе" <<
    '|'; //Номер телефона в городе
325.     cout << left << setw(15) << "Н/Т абонента" << '|' <<
    endl; //Номер телефона абонента
326.     cout << string(119, '-') << endl;
327. }
328. //Menu.cpp
329. #include "Menu.h"
330.
331. list<Conversation> list_conversations;
332.
333. void Run() {
334.     int choice;
335.     while (true)
336.     {
337.         print_menu();
338.         cin >> choice;
339.         switch (choice)
340.         {
341.             case 1:
342.                 system("cls");
343.                 add_conversation();
344.                 break;
345.             case 2:
346.                 system("cls");
347.                 delete_conversation();
348.                 break;
349.             case 3:
```


Продолжение приложения А

```
350.         system("cls");
351.         print_conversations();
352.         break;
353.     case 4:
354.         system("cls");
355.         change_conversation();
356.         break;
357.     case 5:
358.         system("cls");
359.         clear_conversations();
360.         break;
361.     case 6:
362.         system("cls");
363.         collect_cities(list_conversations);
364.         print_cities();
365.         break;
366.     case 7:
367.         system("cls");
368.         collect_cities(list_conversations);
369.         SearchValue();
370.         break;
371.     case 8:
372.         system("cls");
373.         collect_cities(list_conversations);
374.         BubbleSort();
375.         break;
376.     case 9:
377.         system("cls");
378.         cout << "Завершение программы..." << endl;
379.         break;
380.     default:
381.         cout << "неправильный ввод" << endl;
382.         break;
383.     }
384.     if (choice == 9) break;
385. }
386. }
387.
388. void print_menu() {
389.     cout << "\n----- Menu -----" << endl;
390.     cout << "1. добавить разговор" << endl;
391.     cout << "2. удалить разговор" << endl;
392.     cout << "3. вывести разговоры" << endl;
393.     cout << "4. изменить разговор" << endl;
394.     cout << "5. очистить разговоры" << endl;
395.     cout << "6. вывести города" << endl;
396.     cout << "7. поиск города по ключу" << endl;
397.     cout << "8. отсортированные города по ключу" << endl;
398.     cout << "9. выйти из программы" << endl;
399.     cout << "-----" << endl;
400.     cout << "*ключ - общее время разговора городов" << endl;
401. }
402.
403. void add_conversation() {
404.     Conversation conversation;
```

Продолжение приложения А

```
405.     conversation = read_conversation_from_user();
406.     write_conversation_to_file(conversation);
407.     list_conversations.push_back(conversation);
408.     cout << "разговор добавлен успешно" << endl;
409. }
410.
411. void delete_conversation() {
412.     system("cls");
413.     print_conversations();
414.     if (list_conversations.size() > 0)
415.     {
416.         cout << "Введите номер звонка для удаления: ";
417.         int num;
418.         cin >> num;
419.         num--;
420.         if (num >= list_conversations.size())
421.         {
422.             cout << "Нет такого элемента" << endl;
423.             return;
424.         }
425.         list<Conversation>::iterator it =
list_conversations.begin();
426.         advance(it, num);
427.         list_conversations.erase(it);
428.         write_list_conversation_to_file();
429.         cout << "Удаление прошло успешно" << endl;
430.     }
431.     else
432.     {
433.         cout << "Нет элементов для удаления" << endl;
434.     }
435. }
436.
437. void change_conversation() {
438.     system("cls");
439.     print_conversations();
440.     if (list_conversations.size() > 0)
441.     {
442.         cout << "Введите номер звонка для изменения: ";
443.         int num;
444.         cin >> num;
445.         num--;
446.         if (num >= list_conversations.size())
447.         {
448.             cout << "Нет такого элемента" << endl;
449.             return;
450.         }
451.         list<Conversation>::iterator it =
list_conversations.begin();
452.         advance(it, num);
453.         *it = read_conversation_from_user();
454.         write_list_conversation_to_file();
455.         cout << "Изменение прошло успешно" << endl;
456.     }
457.     else
```

Продолжение приложения А

```
458.     {
459.         cout << "Нет звонков для изменения" << endl;
460.     }
461. }
462.
463. void print_conversations() {
464.     system("cls");
465.     if (list_conversations.empty())
466.     {
467.         cout << "Нет звонков для вывода" << endl;
468.     }
469.     else {
470.         int i = 1;
471.         shapka_conversation();
472.
473.         for (list<Conversation>::const_iterator it =
list_conversations.cbegin(); it !=
list_conversations.cend(); ++it)
474.         {
475.             print_conversation(*it, i);
476.             i++;
477.         }
478.     }
479. }
480.
481. void clear_conversations() {
482.     if (list_conversations.empty())
483.     {
484.         cout << "Звонки очищены" << endl;
485.     }
486.     else {
487.         delete_file_conversation();
488.         list_conversations.clear();
489.         cout << "Звонки очищены" << endl;
490.     }
491. }
492.
493. void write_list_conversation_to_file() {
494.     delete_file_conversation();
495.     for (list<Conversation>::const_iterator it =
list_conversations.cbegin(); it !=
list_conversations.cend(); ++it)
496.     {
497.         write_conversation_to_file(*it);
498.     }
499. }
500.
501. void read_conversation_from_file_to_list()
502. {
503.     read_conversation_from_file(list_conversations);
504. }
```

ПРИЛОЖЕНИЕ Б. Схема алгоритма

ПРИЛОЖЕНИЕ В. Схема алгоритма