

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Дисциплина: Технология разработки программного обеспечения

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе на тему

**ПРОГРАММА АВТОМАТИЗАЦИИ ПОДБОРА ТУРИСТИЧЕСКИХ
ПУТЁВОК**

Студентка:
гр. 210901 Лойко А.И.

Руководитель:
Василькова А.Н.

Минск 2023

СОДЕРЖАНИЕ

Введение	4
1. Требования к программе	5
1.1 Исходные требования к курсовой работе	5
1.2 Функциональные требования	6
1.3 Требования к программной реализации	8
2. Конструирование программы	9
2.1 Разработка классов	9
2.2 Выбор способа организации данных	9
2.3 Разработка методов классов	10
3. Разработка программы	12
3.1 Описание класса «Main»	12
3.2 Описание класса «Window_Main»	12
3.3 Описание класса «Data»	12
4. Описание работы программы	13
4.1 Вход в программу	13
4.2 Модуль пользователя	13
4.3 Модуль администратора	16
4.4 Исключительные ситуации	20
Заключение	22
Список использованных источников	23
Приложение А (обязательное) Листинг кода с комментариями	24

ВВЕДЕНИЕ

Целью разработки курсовой работы «Программа автоматизации подбора туристических путёвок» является создание комфортной и функциональной программы для работы с туристическими путевками.

Пояснительная записка состоит из четырех разделов.

В первом разделе «Требования к программе» будет описано:

- Исходные требования к курсовой работе,
- Функциональные требования,
- Требования к программной реализации.

Во втором разделе «Конструирование программы» будет описано:

- Разработка классов,
- Выбор способа организации данных,
- Разработка методов классов.

В третьем разделе «Разработка программы» будет описано:

- Класс «Main»,
- Класс «Window_Main»,
- Класс «Data».

В четвёртом разделе «Описание работы программы» будет описано:

- Вход в программу,
- Модуль пользователя,
- Модуль администратора,
- Исключительные ситуации.

В приложении А будет представлен листинг кода с комментариями.

В графической части будут представлены следующие диаграммы: диаграмма классов, диаграмма вариантов использования, диаграмма деятельности.

1 ТРЕБОВАНИЯ К ПРОГРАММЕ

Программа автоматизации подбора туристических путевок осуществляет поиск путевочных билетов по параметрам: дата отправления; дата приезда; цена (от минимального до максимального заданного значения); город отправления; город приезда, вид транспорта. В программе должны быть понятные элементы интерфейса для сбора данных, вводимых пользователем, элементы обратной связи для понимания состояния программы в каждый момент времени.

В процессе создания программы необходимо реализовать хранение данных о имеющихся билетах в файле, функции программного добавления, изменения и удаления записи о той или иной путёвке, просмотр записей о путевках в удобном формате (учесть случаи, когда список путевок не помещается в окне программы, и случай, когда ни одной путевки по введенным параметрам поиска не найдено), практичное переключение между функциями и окнами, защиту от ввода некорректных данных, от попыток изменения или добавления путевок с пустыми или слишком длинными параметрами, от ввода несуществующих дат или названий городов и видов транспорта.

Реализовать авторизацию для входа в систему, функционал администратора и функционал пользователя, как минимум три вида поиска, как минимум три вида сортировки.

1.1 Исходные требования к курсовой работе

1. Язык программирования C++.
2. Среда разработки Microsoft Visual Studio Code версии 2020 и выше.
3. Вид приложения – GUI.
4. Парадигма программирования – объектно-ориентированная.
5. Способ организации данных – классы.
6. Способ хранения данных – файлы.
7. Каждая логически завершенная задача программы должна быть реализована в виде объектов определённого класса.
8. К защите курсовой работы предоставляются: GUI приложение и пояснительная записка.
9. Текст пояснительной записки оформляется в соответствии со стандартом предприятия «СТП 01–2017».

1.2 Функциональные требования к курсовой работе

В начале работы программы осуществляется чтение данных о путёвках из файла формата «txt» и помещение их в объект класса «Data» для дальнейшей работы. При каждом изменении информации о путёвках происходит изменение как объекта класса «Data», так и файла. Также при изменении данных о путёвках происходит своевременное изменение списка путевок, отображаемого на главном окне программы.

Сразу после запуска программа готова к работе. Пользователю без авторизации доступны базовые функции программы: просмотр путёвок, поиск по параметрам и сортировка. Для управления поиском и сортировкой предусмотрены поля ввода и выбора данных и кнопки «Подобрать» и «Сортировать».

Для поиска по интересующим параметрам пользователю необходимо ввести интересующие данные (для ввода доступны даты отправления и прибытия и диапазон цен) или выбрать из выпадающего списка (для выбора доступны города отправления и прибытия и вид транспорта). Если какие-то из параметров окажутся неважными для пользователя, он имеет возможность оставить их пустыми. Как только настройка параметров поиска завершена, необходимо нажать на кнопку «Подобрать», после чего в списке путевок отобразятся найденные по заданным параметрам результаты.

Для сортировки необходимо выбрать из выпадающего списка параметр сортировки (по умолчанию в поле выбора стоит пункт «По умолчанию», что означает сортировку по порядку записи в файле) и нажать на кнопку «Сортировать», после чего в области вывода путевок отобразятся все доступные путёвки в искомом порядке.

Для авторизации пользователю необходимо нажать на кнопку «Войти в аккаунт», после чего в отобразившемся окне ввести логин и пароль. После авторизации происходит переход в режим администратора и становятся доступны дополнительные функции: добавление записи, изменение записи и удаление записи о той или иной путёвке.

Основной модуль программы включает:

1. Работа с записями о путёвках:

- просмотр всех путёвок,
- поиск путёвок по параметрам,
 - а) поиск по дате отправления,
 - б) поиск по дате прибытия,
 - с) поиск по цене (от минимальной до максимальной),
 - д) поиск по городу отправления,
 - е) поиск по городу прибытия,

- f) поиск по виду транспорта,
- сортировка путёвок по параметрам,
 - a) сортировка по дате отправления,
 - b) сортировка по дате прибытия,
 - c) сортировка по цене,
 - d) сортировка по городу отправления,
 - e) сортировка по городу прибытия,
 - f) сортировка по виду транспорта,
 - g) сортировка по количеству билетов,
 - h) сортировка по умолчанию,
- 2. Вход в режим администратора;
Модуль администратора включает:
- 3. Работа с записями о путёвках:
 - просмотр всех путёвок,
 - поиск путёвок по параметрам,
 - a) поиск по дате отправления,
 - b) поиск по дате прибытия,
 - c) поиск по цене (от минимальной до максимальной),
 - d) поиск по городу отправления,
 - e) поиск по городу прибытия,
 - f) поиск по виду транспорта,
 - сортировка путёвок по параметрам,
 - a) сортировка по дате отправления,
 - b) сортировка по дате прибытия,
 - c) сортировка по цене,
 - d) сортировка по городу отправления,
 - e) сортировка по городу прибытия,
 - f) сортировка по виду транспорта,
 - g) сортировка по количеству билетов,
 - h) сортировка по умолчанию,
 - добавление записи,
 - изменение записи,
 - удаление записи;
- 2. Выход из режима администратора;

Для реализации перечисленных модулей/подмодулей необходимо создавать меню с соответствующими пунктами.

Предусмотреть:

1. Обработку исключительных ситуаций:
 - попытка ввести несоответствующие формату данные (символы в числовом поле и т.д.);

- ничего не найдено по результатам поиска;
 - попытка ввести номер удаляемой записи, выбор действия, номер редактирования записи и т.д., выходящий за пределы массива;
 - попытка ввести нелогичные данные (отрицательная цена, несуществующий возраст и т.д.);
2. Возможность возврата назад (навигация);
 3. Обратная связь с пользователем, вывод сообщения об успешности удаления/добавления/редактирования записи.

1.3 Требования к программной реализации

1. Все классы и объекты должны иметь осмысленные имена в рамках тематики варианта к курсовой работе. Объектам и классам рекомендуется присваивать имена, согласно C++ Code Convention.

2. Имена методов должны быть осмысленными и строиться по принципу «глагол + существительное». Если функция выполняет какую-либо проверку и возвращает результат типа `bool`, то ее название должно начинаться с глагола `is` (например, `isFileExist`, `isUnicLogin`).

3. Не допускается использование оператора прерывания `goto`.

4. Код не должен содержать неименованных числовых констант (так называемых «магических» чисел), неименованных строковых констант (например, имен файлов и др.). Подобного рода информацию следует выносить в глобальные переменные с атрибутом `const`. По правилам хорошего стиля программирования тексты всех информационных сообщений, выводимых пользователю в ответ на его действия, также оформляются как константы.

5. Код необходимо комментировать (как минимум в части нетривиальной логики).

6. Код не должен дублироваться – для этого существуют методы и функции.

7. Один метод решает только одну задачу (например, не допускается в одном методе считывать данные из файла и выводить их на экран – это два разных метода). При этом внутри метода возможен вызов других методов.

8. Следует избегать длинных функций и глубокой вложенности: текст метода должен уместиться на один экран, а вложенность блоков и операторов должна быть не более трёх.

9. Следует избегать длинных методов: текст метода должен уместиться на один экран (размер текста не должен превышать 25-50 строк).

10. Минимизировать область видимости объекта, сделав его как можно более локальным.

2 КОНСТРУИРОВАНИЕ ПРОГРАММЫ

Реализация программы будет осуществляться на языке C++ в IDE среде Microsoft Visual Studio Code. Программа будет компилироваться и использоваться в операционных системах семейства Microsoft Windows версии 7 и выше.

2.1 Разработка структуры программы

Согласно требованиям к программе, необходимо наличие исполняемой программы, которая работает с файлом «travel.txt». С точки зрения верхней архитектуры программы, можно выделить два основных модуля: модуль работы с данными, модуль авторизации.

Метод авторизации администратора заключается в проверке соответствия введенного логина и пароля логину и паролю, хранящемуся в векторе.

После успешной авторизации администратора становятся доступны функции работы с данными: добавления, удаления и редактирования записи.

2.2 Выбор способа организации данных

Для представления в программе аккаунтов администратора создается структура «admin», содержащая «login» и «password».

- «login» – логин администратора,
- «password» – пароль администратора.

Для представления в программе записи о каждой путевке создается структура «data_struct», содержащая «num_line», «num_file», «cityfirst», «monthfirst», «daysecond», «monthsecond», «citysecond», «price», «transport» и «number».

- «num_line» – порядковый номер записи,
- «num_file» – номер записи в файле,
- «cityfirst» – город отправления,
- «dayfirst» – день отправления,
- «monthfirst» – месяц отправления,
- «daysecond» – день прибытия,
- «monthsecond» – месяц прибытия,
- «citysecond» – город прибытия,
- «price» – цена путевки,
- «transport» – вид транспорта,
- «number» – количество билетов в наличии.

Программа обслуживает данные о записях, хранимые построчно

в файле «travel.txt». Пример реализации показан на рисунке 2.1.

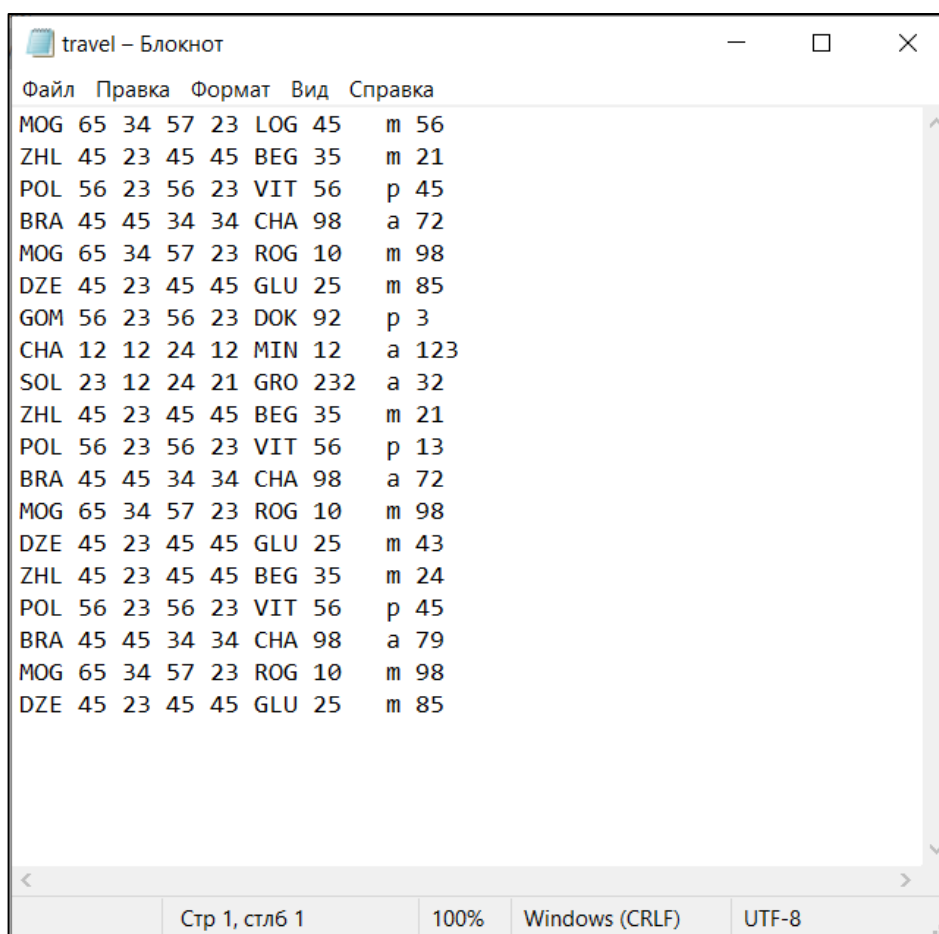


Рисунок 2.1 – хранение данных о записях файле travel.txt

Для хранения текстовых данных выбрана кодировка ANSI.

2.3 Разработка перечня пользовательских функций программы

Выход из программы – функция выхода из программы.

Авторизация – функция авторизации, администратор вводит логин и пароль и при корректно введенных данных производится переход в режим администратора.

Модуль пользователя (предоставляется меню возможного функционала для пользователя):

- Просмотр всех путевок – пользователь может просмотреть данные обо всех путевках (дата отправления, дата прибытия, цена, город отправления, город прибытия, вид транспорта)

- Поиск – предоставляется меню поиска (поиск по городу отправления, по городу прибытия, по дате отправления, по дате прибытия, по

цене (от и до)), вводятся данные, которые необходимо найти, в случае успешного поиска выводиться найденная информация.

- Сортировка – предоставляется сортировка по параметрам (сортировка по цене, по алфавиту (по городу отправления, по городу прибытия, по виду транспорта), по количеству билетов, по дате отправления, по дате прибытия), выбирается по какому признаку сортировать данные, в зависимости от выбора выводиться соответствующая информация.

Модуль администратора (предоставляется меню возможного функционала для администратора):

- Просмотр всех путевок – администратор может просмотреть данные обо всех путевках (дата отправления, дата прибытия, цена, город отправления, город прибытия, вид транспорта)

- Поиск – предоставляется меню поиска (поиск по городу отправления, по городу прибытия, по дате отправления, по дате прибытия, по цене (от и до)), вводятся данные, которые необходимо найти, в случае успешного поиска выводиться найденная информация.

- Сортировка – предоставляется сортировка по параметрам (сортировка по цене, по алфавиту (по городу отправления, по городу прибытия, по виду транспорта), по количеству билетов, по дате отправления, по дате прибытия), выбирается по какому признаку сортировать данные, в зависимости от выбора выводиться соответствующая информация.

- Добавление – администратор может добавлять новую запись в файл.

- Редактирование – администратор может редактировать запись по номеру в файле.

- Удаление – администратор может удалять записи по номеру из файла.

- Выход из режима администратора – после подтверждения о выходе можно выйти к главному модулю.

3 РАЗРАБОТКА ПРОГРАММЫ

3.1 Описание класса «Main»

Класс «Main» содержит функцию «main», которая является точкой входа в программу и содержит основную логику работы всех модулей программы, вызывая в нужные моменты выполнения нужные методы. Содержит цикл работы программы, таймер обновления экрана, обработку исключительных ситуаций.

3.2 Описание класса «Window_Main»

Класс «Window_Main» создает главное окно программы. Содержит методы для создания всех графических объектов главного окна, включает классы «TextBox» – для создания текстовых надписей, «RectangleBox» – для создания цветного прямоугольника заданных размеров и координат, «Status» – для вывода сообщения о состоянии программы для обратной связи с пользователем, «OutputBox» – для вывода информации о путевках, методы получения данных из объектов классов «WriteBox» и «ChooseBox», передает их классу «Data» для работы с данными. Получает состояния объектов класса «Button» для взаимодействия с пользователем. При каждом такте таймера обновляет состояния объектов классов «Button», «ChooseBox», «WriteBox», «ScrollBar» и «OutputBox» для своевременного реагирования на действия пользователя. Изменяет статус при каждом логическом действии пользователя. Передает управление программой другим окнам, а именно объектам класса «Window_Add», «Window_Change», «Window_Delete» и «Window_Login» для добавления записи, изменения записи, удаления записи или авторизации соответственно, при необходимости.

3.3 Описание класса «Data»

Класс «Data» включает объект класса «File» для работы с файлом и вектор структуры «data_struct», куда записываются все данные из файла. Содержит методы для работы с данными, а именно запись данных в вектор, получение данных их вектора для редактирования файла, преобразование данных для передачи в объекты для работы с графикой, методы для поиска записи и сортировки записей.

4 ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

4.1 Вход в программу

При запуске программа открывает файл «travel», затем открывается главное окно программы с выводом найденных путевок на экран. Задается статус «Программа открыта». В верхней части окна располагаются панели поиска и сортировки. В центральной части окна находится вывод записей о путевках в виде пролистываемой области. Пример реализации показан на рисунке 4.1.

№	Город отправления и прибытия	Даты отправления и прибытия	Вид транспорта	Цена, BYN	В наличии
1	Могилев Логойск	65.34.2023 57.23.2023	Маршрутка	45	56
2	Жлобин Бегомль	45.23.2023 45.45.2023	Маршрутка	35	21
3	Полоцк Витебск	56.23.2023 56.23.2023	Поезд	56	45
4	Браслав Чашники	45.45.2023 34.34.2023	Автобус	98	72
5	Могилев Рогачев	65.34.2023 57.23.2023	Маршрутка	10	98
6	Дзержинск Глубокое	45.23.2023 45.45.2023	Маршрутка	25	85
7	Гомель Докшицы	56.23.2023 56.23.2023	Поезд	92	3
8	Чашники Минск	12.12.2023 24.12.2023	Автобус	12	123
9	Солигорск Гродно	23.12.2023 24.21.2023	Автобус	232	32
10	Жлобин Бегомль	45.23.2023 45.45.2023	Маршрутка	35	21
11	Полоцк Витебск	56.23.2023 56.23.2023	Поезд	56	13
12	Браслав Чашники	45.45.2023 34.34.2023	Автобус	98	72
13	Могилев Рогачев	65.34.2023 57.23.2023	Маршрутка	10	98

Рисунок 4.1 – Главное окно программы

4.2 Модуль пользователя

– **Поиск** – пользователю предоставляется возможность подобрать путевки по интересующим параметрам (дата отправления, дата прибытия, диапазон цен, город отправления, город прибытия, вид транспорта). Если тот или иной параметр поиска не имеет значения, пользователь может оставить поле ввода пустым. Тогда при поиске данный параметр не будет учитываться. После выполнения поиска статус программы обновляется на «Найдено: X результатов», где X – количество найденных путевок. Пример реализации показан на рисунке 4.2.

Программа автоматизации подбора туристических путёвок

Программа «Туристпутёвподборавтоматизатор–4000» v1.7

Дата начала: 12.12.2023 Дата конца: 24.12.2023 Цена, BYN: от 2 до 46 Город отправления: Чашники Город прибытия: Минск Вид транспорта: Автобус **Подобрать**

Выбрать сортировку: По умолчанию **Сортировать**

№	Город отправления и прибытия	Даты отправления и прибытия	Вид транспорта	Цена, BYN	В наличии
8	Чашники Минск	12.12.2023 24.12.2023	Автобус	12	123

✓ Найдено: 1 результатов

Войти в аккаунт **Выйти**

Рисунок 4.2 – Поиск путевок по параметрам

– **Сортировка** – пользователю предоставляется возможность отсортировать путевки по интересующему параметру (по дате отправления, по дате прибытия, по цене, по городу отправления, по городу прибытия, по виду транспорта, по умолчанию). После выполнения сортировки статус программы обновляется на «Записи отсортированы». Пример реализации показан на рисунке 4.3.

Программа автоматизации подбора туристических путёвок

Программа «Туристпутёвподборавтоматизатор–4000» v1.7

Дата начала: Дата конца: Цена, BYN: от до Город отправления: Город прибытия: Вид транспорта: **Подобрать**

Выбрать сортировку: По цене **Сортировать**

№	Город отправления и прибытия	Даты отправления и прибытия	Вид транспорта	Цена, BYN	В наличии
5	Могилев Рогачев	65.34.2023 57.23.2023	Маршрутка	10	98
13	Могилев Рогачев	65.34.2023 57.23.2023	Маршрутка	10	98
18	Могилев Рогачев	65.34.2023 57.23.2023	Маршрутка	10	98
8	Чашники Минск	12.12.2023 24.12.2023	Автобус	12	123
9	Солигорск Гродно	23.12.2023 24.21.2023	Автобус	232	32
6	Дзержинск Глубокое	45.23.2023 45.45.2023	Маршрутка	25	85
14	Дзержинск Глубокое	45.23.2023 45.45.2023	Маршрутка	25	43
19	Дзержинск Глубокое	45.23.2023 45.45.2023	Маршрутка	25	85
2	Жлобин Бегомль	45.23.2023 45.45.2023	Маршрутка	35	21
10	Жлобин Бегомль	45.23.2023 45.45.2023	Маршрутка	35	21
15	Жлобин Бегомль	45.23.2023 45.45.2023	Маршрутка	35	24
1	Могилев Логойск	65.34.2023 57.23.2023	Маршрутка	45	56
3	Полоцк Витебск	56.23.2023 56.23.2023	Поезд	56	45

✓ Записи отсортированы

Войти в аккаунт **Выйти**

Рисунок 4.3 – Сортировка записей

– **Авторизация** – при нажатии кнопки «Войти в аккаунт» появляется окно авторизации, предлагается ввести логин и пароль. Пример реализации показан на рисунке 4.4.

Рисунок 4.4 – Окно авторизации

При вводе корректных логина и пароля происходит возврат к главному окну и переход в режим администратора, статус обновляется на «Вы вошли в аккаунт». Пример реализации показан на рисунке 4.5.

№	Город отправления и прибытия	Даты отправления и прибытия	Вид транспорта	Цена, BYN	В наличии
1	Могилев Логойск	65.34.2023 57.23.2023	Маршрутка	45	56
2	Жлобин Бегомль	45.23.2023 45.45.2023	Маршрутка	35	21
3	Полоцк Витебск	56.23.2023 56.23.2023	Поезд	56	45
4	Браслав Чашники	45.45.2023 34.34.2023	Автобус	98	72
5	Могилев Рогачев	65.34.2023 57.23.2023	Маршрутка	10	98
6	Дзержинск Глубокое	45.23.2023 45.45.2023	Маршрутка	25	85
7	Гомель Докшицы	56.23.2023 56.23.2023	Поезд	92	3
8	Чашники Минск	12.12.2023 24.12.2023	Автобус	12	123
9	Солигорск Гродно	23.12.2023 24.21.2023	Автобус	232	32
10	Жлобин Бегомль	45.23.2023 45.45.2023	Маршрутка	35	21
11	Полоцк Витебск	56.23.2023 56.23.2023	Поезд	56	13
12	Браслав Чашники	45.45.2023 34.34.2023	Автобус	98	72
13	Могилев Рогачев	65.34.2023 57.23.2023	Маршрутка	10	98

Рисунок 4.5 – Успешная авторизация

4.3 Функции администратора

– **Добавление записи** – администратору предоставляется возможность добавить запись в файл. Пример реализации показан на рисунке 4.6.

Добавление записи

Дата начала Дата конца Цена, BYN В наличии

Город отправления Город прибытия Вид транспорта

Минск Минск Автобус

Добавить

Рисунок 4.6 – Окно добавления записи

После добавления записи статус программы обновляется на «Запись добавлена». Пример реализации показан на рисунке 4.7.

Программа «Туристпутёвподборавтоматизатор–4000» v1.7

Дата начала Дата конца Цена, BYN Город отправления Город прибытия Вид транспорта

от до ----- ----- -----

Подобрать

Выбор сортировки

По умолчанию Сортировать

№	Город отправления и прибытия	Даты отправления и прибытия	Вид транспорта	Цена, BYN	В наличии
1	Минск	Молодечно	13.8.2023 21.8.2023	Поезд	12 47
2	Жлобин	Бегомль	45.23.2023 45.45.2023	Маршрутка	35 21
3	Полоцк	Витебск	56.23.2023 56.23.2023	Поезд	56 45
4	Браслав	Чашники	45.45.2023 34.34.2023	Автобус	98 72
5	Могилев	Рогачев	65.34.2023 57.23.2023	Маршрутка	10 98
6	Дзержинск	Глубокое	45.23.2023 45.45.2023	Маршрутка	25 85
7	Гомель	Докшицы	56.23.2023 56.23.2023	Поезд	92 3
8	Чашники	Минск	12.12.2023 24.12.2023	Автобус	12 123
9	Солигорск	Гродно	23.12.2023 24.21.2023	Автобус	232 32
10	Жлобин	Бегомль	45.23.2023 45.45.2023	Маршрутка	35 21
11	Полоцк	Витебск	56.23.2023 56.23.2023	Поезд	56 13
12	Браслав	Чашники	45.45.2023 34.34.2023	Автобус	98 72
13	Могилев	Рогачев	65.34.2023 57.23.2023	Маршрутка	10 98

✓ Запись добавлена Изменить Удалить Добавить Выйти из аккаунта Выйти

Рисунок 4.7 – Успешное добавление записи

– **Редактирование записи** – администратору предоставляется возможность редактировать любую запись в файле. Пример реализации показан на рисунке 4.8.

Рисунок 4.8 – Окно редактирования записи

После редактирования записи статус программы обновляется на «Запись изменена». Пример реализации показан на рисунке 4.9.

№	Город отправления и прибытия	Даты отправления и прибытия	Вид транспорта	Цена, BYN	В наличии
1	Минск	Молодечно	13.8.2023 21.8.2023	Поезд	12 47
2	Жлобин	Бегомль	45.23.2023 45.45.2023	Маршрутка	35 21
3	Полоцк	Витебск	56.23.2023 56.23.2023	Поезд	56 45
4	Браслав	Чашники	45.45.2023 34.34.2023	Автобус	98 72
5	Могилев	Рогачев	65.34.2023 57.23.2023	Маршрутка	10 98
6	Дзержинск	Глубокое	45.23.2023 45.45.2023	Маршрутка	25 85
7	Гомель	Докшицы	56.23.2023 56.23.2023	Поезд	92 3
8	Чашники	Минск	12.12.2023 24.12.2023	Автобус	12 123
9	Солигорск	Гродно	23.12.2023 24.21.2023	Автобус	232 32
10	Жлобин	Бегомль	45.23.2023 45.45.2023	Маршрутка	35 21
11	Полоцк	Витебск	56.23.2023 56.23.2023	Поезд	56 13
12	Браслав	Чашники	45.45.2023 34.34.2023	Автобус	98 72
13	Могилев	Рогачев	65.34.2023 57.23.2023	Маршрутка	10 98

Рисунок 4.9 – Успешное редактирование записи

– **Удаление записи** – администратору предоставляется возможность удалить любую запись из файла (рисунок 4.10).

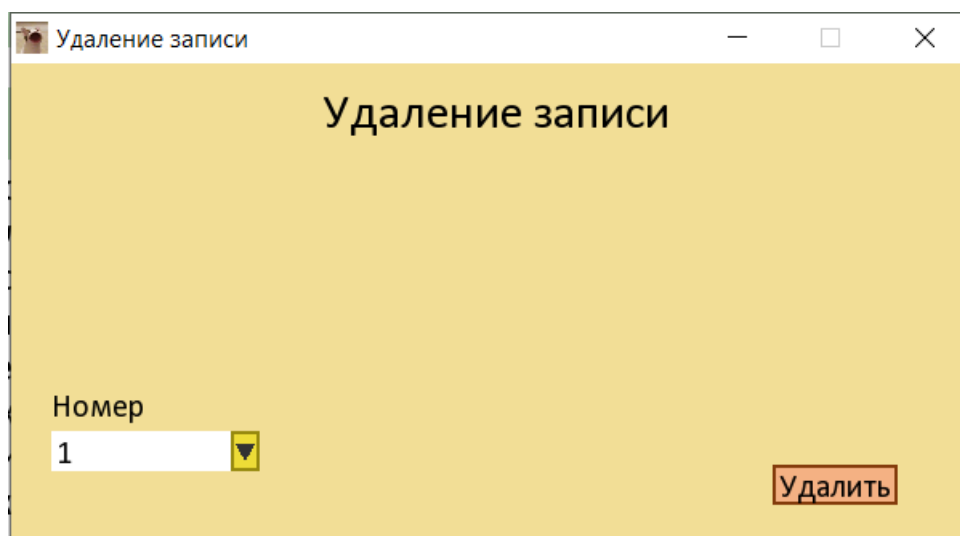


Рисунок 4.10 – Окно удаления записи

После удаления записи статус программы обновляется на «Запись удалена». Пример реализации показан на рисунке 4.11.

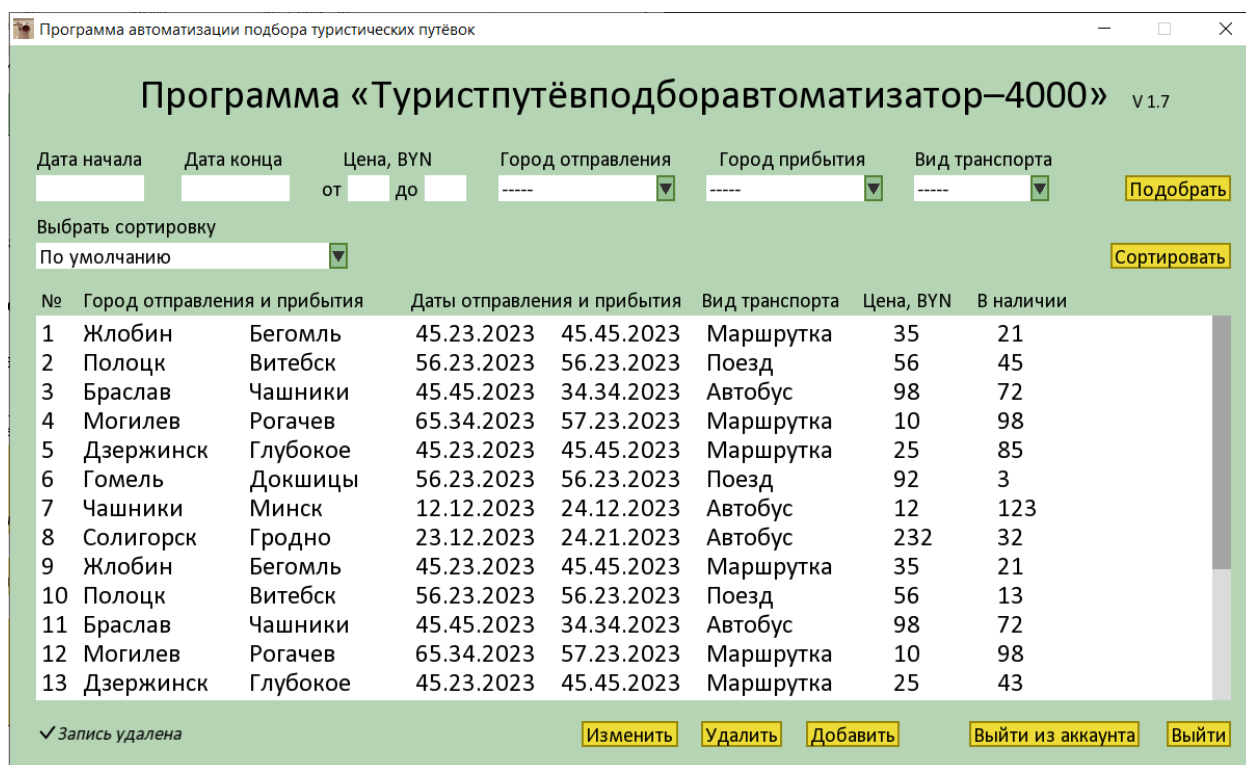


Рисунок 4.11 – Успешное удаление записи

– **Выход из аккаунта** – При нажатии на кнопку «Выйти из аккаунта» появляется окно выхода «Вы точно хотите выйти из аккаунта?» с кнопками «Да» и «Нет» (рисунок 4.12).

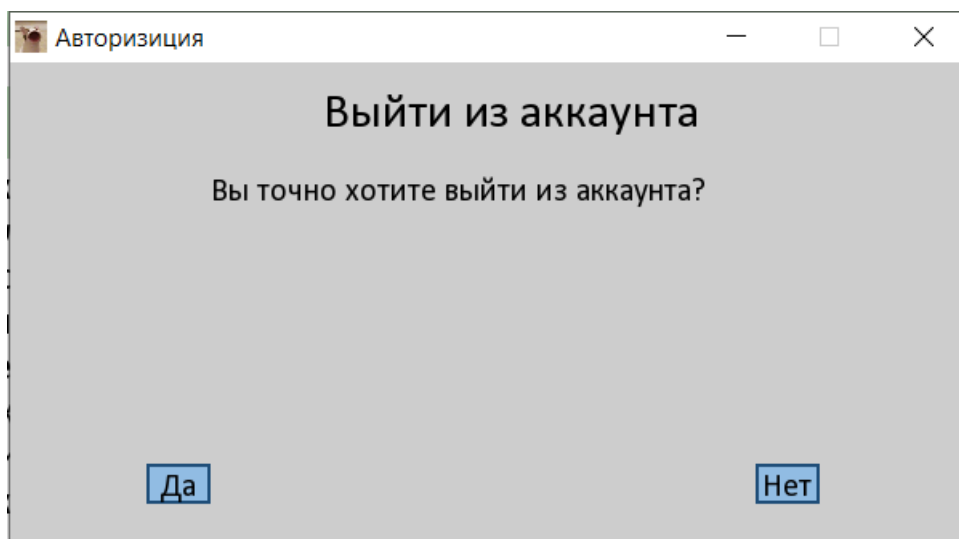


Рисунок 4.12 – Окно выхода из аккаунта

При нажатии на кнопку «Да» происходит переход в режим пользователя, статус программы изменяется на «Вы вышли из аккаунта». Пример реализации показан на рисунке 4.13.

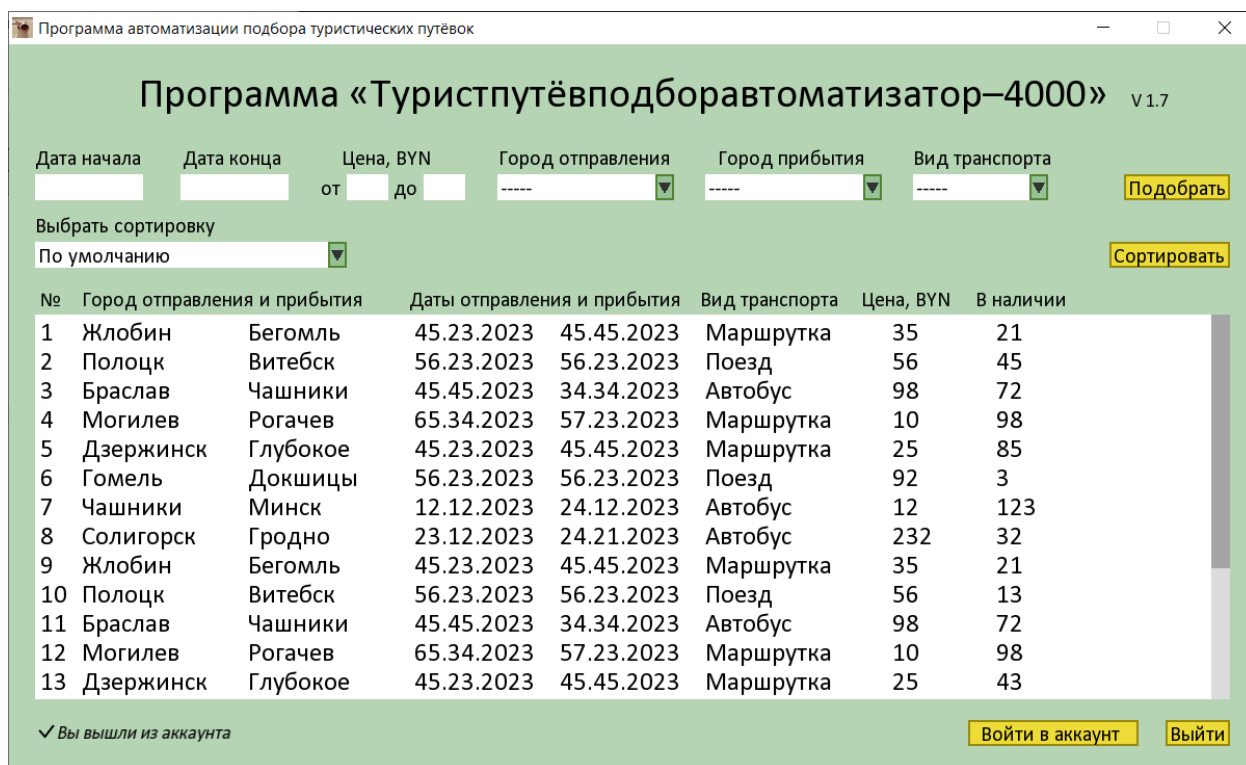


Рисунок 4.13 – Успешный выход из аккаунта

При нажатии на кнопку «Нет» происходит возврат на главное окно программы.

– **Выход из программы** – при нажатии кнопки «Выйти» закрывается главное окно и происходит завершение программы.

4.4 Исключительные ситуации

В данной программе предусмотрены все основные исключительные ситуации, которые могут возникнуть в опыте пользователя.

В текстовых полях ввода введено ограничение на число и тип вводимых символов. Например, на ввод цены отведено 3 символа, на ввод даты – 4 символа, на ввод логина и пароля – 20 символов. Пример реализации показан на рисунке 4.14.

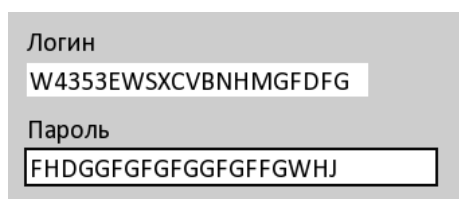
The image shows a login form with two input fields. The first field is labeled 'Логин' (Login) and contains the text 'W4353EWSXCVBNHMGDFG'. The second field is labeled 'Пароль' (Password) and contains the text 'FHDGGFGFGGGFGFFGWHJ'. Both fields have a maximum length of 20 characters.

Рисунок 4.14 – Ввод логина и пароля

Причем при вводе даты присутствует автоматическая вставка точек между числом и месяцем, месяцем и годом, автоматическая вставка текущего года – 2023-го. В полях ввода даты и цены для ввода доступны только цифры. В полях ввода логина и пароля для ввода доступны цифры и латинские буквы.

Для предупреждения вводом пользователем несуществующего города, несуществующего транспорта, несуществующего номера редактируемой либо удаляемой записи, получение данных производится с помощью полей выбора. Пример реализации показан на рисунке 4.15.

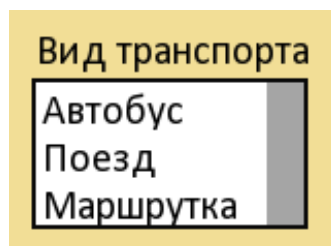
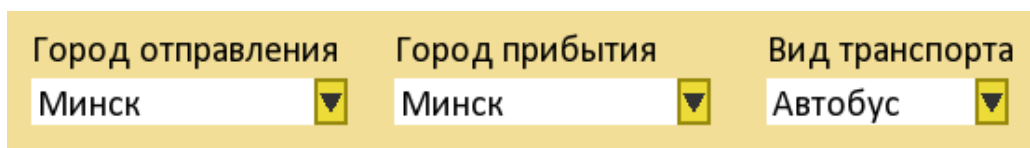
The image shows a dropdown menu titled 'Вид транспорта' (Type of transport). The menu is open, showing three options: 'Автобус' (Bus), 'Поезд' (Train), and 'Маршрутка' (Minibus). The 'Автобус' option is currently selected.

Рисунок 4.15 – Поле выбора транспорта при добавлении и изменении записи

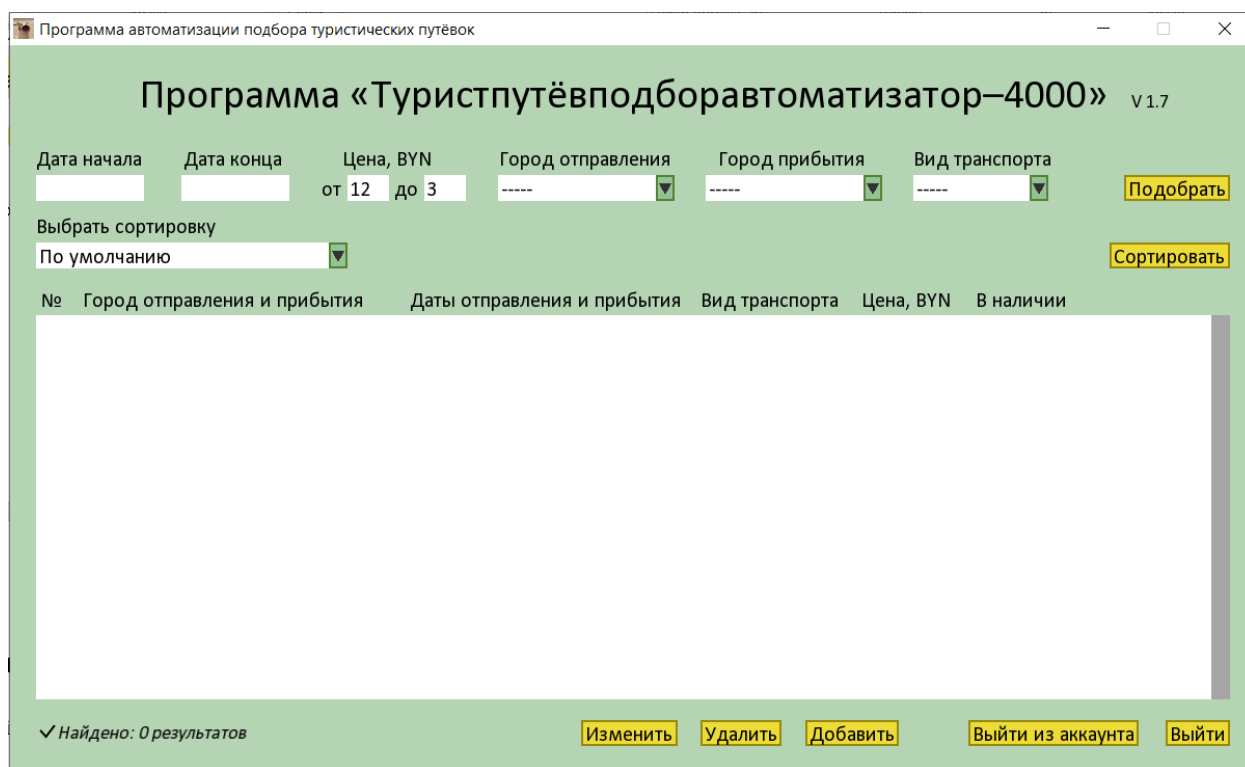
При этом в окнах редактирования и добавления записей присутствует автоматический выбор города и транспорта по умолчанию, чтобы исключить возможность оставить данные поля пустыми при записи в файл. Пример реализации показан на рисунке 4.16.



The image shows a yellow rectangular box containing three dropdown menus. The first menu is labeled 'Город отправления' (City of departure) and has 'Минск' (Minsk) selected. The second menu is labeled 'Город прибытия' (City of arrival) and also has 'Минск' selected. The third menu is labeled 'Вид транспорта' (Type of transport) and has 'Автобус' (Bus) selected. Each dropdown menu has a small yellow arrow icon on its right side.

Рисунок 4.16 – Автозаполнение полей выбора городов и транспорта при добавлении/редактировании записи

При вводе некорректного диапазона цен, при котором минимальная цена больше максимальной, ошибок в программе не происходит. Высвечивается пустой блок вывода записей, статус изменяется на «Найдено: 0 результатов». Пример реализации показан на рисунке 4.17.



The image shows a screenshot of a software application window titled 'Программа автоматизации подбора туристических путёвок' (Program for automating the selection of tourist packages). The main title bar says 'Программа «Туристпутёвподборавтоматизатор-4000» v1.7'. The interface includes several input fields: 'Дата начала' (Start date), 'Дата конца' (End date), 'Цена, BYN' (Price, BYN) with a range from 'от 12' to 'до 3', 'Город отправления' (City of departure), 'Город прибытия' (City of arrival), and 'Вид транспорта' (Type of transport). There are 'Подобрать' (Select) and 'Сортировать' (Sort) buttons. Below these is a 'Выбор сортировки' (Sorting) dropdown menu set to 'По умолчанию' (By default). A table header is visible with columns: '№', 'Город отправления и прибытия', 'Даты отправления и прибытия', 'Вид транспорта', 'Цена, BYN', and 'В наличии'. The table body is empty. At the bottom, there is a status bar showing '✓ Найдено: 0 результатов' (Found: 0 results) and several action buttons: 'Изменить' (Edit), 'Удалить' (Delete), 'Добавить' (Add), 'Выйти из аккаунта' (Log out), and 'Выйти' (Exit).

Рисунок 4.17 – Некорректный ввод диапазона цен

ЗАКЛЮЧЕНИЕ

В ходе данной курсовой работы после анализа уже существующих программных решений поставленной задачи была разработана программа автоматизации подбора туристических путевок, с учетом всех особенностей данной темы. Была достигнута цель данной курсовой работы – создание приложения, осуществляющего функционал для подбора путевок по заданным параметрам.

Разработанное приложение имеет функционал авторизации, редактирования списка записей о путевках, поиска записей по параметрам и сортировки записей. Приложение имеет простой и удобный функционал, благодаря которому пользователь легко сможет разобраться в его работе.

Особенностью полученного приложения можно назвать доступность для каждого пользователя, своевременное реагирование на действия пользователя и легкий переход в режим администратора, в котором, вдобавок ко всем пользовательским функциям, доступно добавление, удаление и редактирование записей в файле.

При написании приложения были учтены основные исключительные ситуации, которые могут возникнуть в процессе работы программы. Были добавлены вспомогательные функции, такие как рамка вокруг активного поля ввода, подсветка выбираемого значения в полях выбора, статус, отображающий актуальное состояние программы после каждого действия пользователя, выделение активной и доступной для нажатия кнопки, подписи каждого поля ввода и выбора.

Данное приложение впоследствии может быть улучшено, например, возможностью добавлением отметок на записи о путевках (добавление в «избранные», в «отложенные» и т.д.), расширением списка городов, функционалом непосредственного заказа и оформления путешествия в сотрудничестве с той или иной туристической компанией и другими усовершенствованиями.

Таким образом, в результате курсовой работы на тему «программа автоматизации подбора туристических путевок» было разработано GUI приложение, предоставляющее пользователю функционал для работы с записями туристических путевок, в котором реализованы основные функции для комфортного и продуктивного пользовательского опыта.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

[1] Меженная М.М. Основы конструирования программ. Курсовое проектирование: пособие / М. М. Меженная – Минск: БГУИР, 2019. – 80 с.: ил [Электронный ресурс]. – Режим доступа: ..\трпо\ТРПО – Меженная ОКП Курсовое проектирование – Пособие. – Дата доступа: 04.03.2023.

[2] Лабораторная работа №6.1 «Особенности рабочего интерфейса Rational Rose» [Электронный ресурс]. – Режим доступа: ..\трпо\ТРПО – ЛР 6 Моделирование Банкомата. – Дата доступа: 29.04.2023
делирование Банкомата. – Дата доступа: 29.04.2023

[3] Инструкция по выполнению лабораторной работы «Построение диаграммы классов. Построение диаграммы состояний. Построение диаграммы последовательности» [Электронный ресурс]. – Режим доступа: ..\трпо\ТРПО – Лабораторная работа №5.docx. – Дата доступа: 02.05.2023

[4] Инструкция по выполнению лабораторной работы «Построение диаграммы вариантов использования. Построение диаграммы деятельности» [Электронный ресурс]. – Режим доступа: ..\трпо\ТРПО – Лабораторная работа №4.docx. – Дата доступа: 06.05.2023

[5] ТРПО – Курсовое проектирование (работа) – Пособие [Электронный ресурс]. – Режим доступа: [https://cloud/mail.ru/public/oaoc/asWuomL8y](https://cloud.mail.ru/public/oaoc/asWuomL8y). – Дата доступа: 25.03.2023

[6] Доманов А.Т. Стандарт предприятия СТП 01-2017 / А.Т. Доманов, Н.И. Сорока. – Минск : БГУИР, 2017. – 169 с. – Дата доступа: 13.04.2023

[7] Программа для автоматизации деятельности турагентств [Электронный ресурс]. – Режим доступа: <https://biznesplan-primer.ru/programma/turagentstvo/master-tur>. – Дата доступа: 22.04.2023

[8] Система автоматизации туристической компании [Электронный ресурс]. – Режим доступа: <http://www.erp.travel/>. – Дата доступа: 25.04.2023

[9] Автоматизация турагентства: выбираем облачную CRM-систему [Электронный ресурс]. – Режим доступа: <https://www.tourbc.ru/tehnologii/obzorg/537-crm-dlya-turagentstva.html>. – Дата доступа: 21.04.2023

[10] Мастер-Тур - комплекс программ для автоматизации туристических агентств [Электронный ресурс]. – Режим доступа: <https://www.megatec.ru/mastertour15>. – Дата доступа: 21.04.2023

[11] Перечень компьютерных систем для туристских агентств [Электронный ресурс]. – Режим доступа: <https://docs.google.com/document/d/1mq003SyssZhX3jj090YZYnUr2mVj9VSRto14PRUgmmY/preview?hgd=1>. – Дата доступа: 22.03.2023

ПРИЛОЖЕНИЕ А

1. Файл main.cpp

```
#include <SFML/Graphics.hpp>
#include <SFML/Audio.hpp>
#include <fstream>
#include <string>
#include <vector>
#include <ctime>
#include <iostream>
#include <cmath> // для round (округления в scrolllist)
#include <windows.h> // чтобы скрыть консоль при запуске

using namespace std;
bool admin_or_user = false;
sf::Vector2f mouse_position;
sf::Cursor cursor_arrow;
sf::Cursor cursor_hand;
sf::Cursor cursor_text;
struct codetocode {
    string code = "";
    wstring name = L"";
};

// для преобразования из wstring в int
struct wstringtoint_struct {
    int code = 0;
    wchar_t name = L'0';
};
```



```

};

struct codetonamechar {
    char code = ' ';
    wstring name = L"";
};

vector<codetonamechar> transportnameschar = {
    {'a', L"Автобус"},
    {'p', L"Поезд"},
    {'m', L"Маршрутка"}
};

struct admin {
    wstring login;
    wstring password;
};

struct textkey {
    bool pressed = false;
    wstring text = L"A";
};

struct numkey {
    bool pressed = false;
    int text = 0;
};

struct events_struct {
    bool R_mouse = false;
    bool L_mouse = false;
    int mouse_scroll = 0;
    bool backspace_pressed = false;
}; events_struct events_mouse;

vector<data_struct> data_file;

```

```

vector<data_wstring_struct> data_wstring;
vector<wstring> citynames_for_chooseboxes;
vector<wstring> transportnames_for_chooseboxes;
vector<wstring> add_change_citynames_for_chooseboxes;
vector<wstring> add_change_transportnames_for_chooseboxes;
// чтобы загрузить font
sf::Font TextBox::font;
TextBox::TextBox_font TextBox::textbox_font_initialize;

int main() {
    ShowWindow(GetConsoleWindow(), SW_HIDE); // чтобы скрыть
    консоль при запуске

    TextBox::font.loadFromFile("materials/fonts/calibri.ttf");

    cursor_arrow.loadFromSystem(sf::Cursor::Arrow);
    cursor_hand.loadFromSystem(sf::Cursor::Hand);
    cursor_text.loadFromSystem(sf::Cursor::Text);

    data.create_it();
    while(program_main.is_open()) {

        float time = clock.getElapsedTime().asSeconds();
        clock.restart();
        timer += time;

        if (timer > delay){
            program_main.run_it();
            if(program_main.is_button_exit_clicked()){

```

```

        window_main.close();
    }

    if(program_main.is_button_find_clicked()) {
        wstring cityfirstwstring = program_main.get_choose-
box_cityfirst_wstring();

        wstring citysecondwstring = program_main.get_choose-
box_citysecond_wstring();

        wstring transportwstring = program_main.get_choose-
box_transporttype_wstring();

        wstring datefirstwstring = pro-
gram_main.get_writebox_datefirst_wstring();

        wstring datesecsecondwstring = pro-
gram_main.get_writebox_datesecsecond_wstring();

        wstring pricefirstwstring = pro-
gram_main.get_writebox_pricefirst_wstring();

        wstring pricesecsecondwstring = pro-
gram_main.get_writebox_pricesecsecond_wstring();

        data.find_by_parameters(cityfirstwstring, citysecondw-
string, transportwstring, datefirstwstring, datesecsecondwstring, pricefirstwstring,
pricesecsecondwstring);

        program_main.change_output();

        wstring resultswstring = L"Найдено: " +
to_wstring(data_wstring.size()) + L" результатов";

        program_main.change_status(resultswstring);

        //      // cout << "dayfirst: " << data_find_min.dayfirst << "
monthfirst: " << data_find_min.monthfirst << " daysecond: " <<
data_find_min.daysecond << " monthsecond: " << data_find_min.monthsecond <<
" cityfirst: " << data_find_min.cityfirst << " citysecond: " << data_find_min.cit-
ysecond << " transport: " << data_find_min.transport << " min.price: " <<
data_find_min.price << " max.price: " << data_find_max.price << "\n";

```

```
}
```

2. Файл `borderbox.h`

```
#ifndef BORDERBOX_H
#define BORDERBOX_H

class BorderBox{
private:
    int window_n;
    int x = 0;
    int y = 0;
    int w = 0;
    int h = 0;
    int thickness = 0;
    sf::Color color = sf::Color(255,255,255);
    sf::RectangleShape rectangle_left;
    sf::RectangleShape rectangle_right;
    sf::RectangleShape rectangle_top;
    sf::RectangleShape rectangle_bottom;

public:

    void set_window_n(int WINDOW_N) { window_n = WINDOW_N; }

    int get_x(){ return x; }
    int get_y(){ return y; }
    int get_w(){ return w; }
    int get_h(){ return h; }
```

```
int get_thickness(){ return thickness; }
```

```
sf::Color get_color(){ return color; }
```

```
void change_color(sf::Color COLOR){  
    color = COLOR;  
    rectangle_left.setFillColor(color);  
    rectangle_right.setFillColor(color);  
    rectangle_top.setFillColor(color);  
    rectangle_bottom.setFillColor(color);  
}
```

```
void change_position(int X, int Y){  
    x = X;  
    y = Y;  
    rectangle_left.setPosition(x-thickness, y-thickness);  
    rectangle_right.setPosition(x+w, y);  
    rectangle_top.setPosition(x, y-thickness);  
    rectangle_bottom.setPosition(x-thickness, y+h);  
}
```

```
void change_size(int W, int H){  
    w = W;  
    h = H;  
    rectangle_left.setPosition(x-thickness, y-thickness);  
    rectangle_right.setPosition(x+w, y);  
    rectangle_top.setPosition(x, y-thickness);  
    rectangle_bottom.setPosition(x-thickness, y+h);  
    rectangle_left.setSize(sf::Vector2f(thickness, h+thickness));  
    rectangle_right.setSize(sf::Vector2f(thickness, h+thickness));  
}
```

```

rectangle_top.setSize(sf::Vector2f(w+thickness, thickness));
rectangle_bottom.setSize(sf::Vector2f(w+thickness, thickness));
}

```

```

void change_thickness(int THICKNESS){
    thickness = THICKNESS;
    rectangle_left.setPosition(x-thickness, y-thickness);
    rectangle_right.setPosition(x+w, y);
    rectangle_top.setPosition(x, y-thickness);
    rectangle_bottom.setPosition(x-thickness, y+h);
    rectangle_left.setSize(sf::Vector2f(thickness, h+thickness));
    rectangle_right.setSize(sf::Vector2f(thickness, h+thickness));
    rectangle_top.setSize(sf::Vector2f(w+thickness, thickness));
    rectangle_bottom.setSize(sf::Vector2f(w+thickness, thickness));
}

```

```

void draw_it(){
    switch (window_n) {
    case 1:
        window_add.draw(rectangle_left);
        window_add.draw(rectangle_right);
        window_add.draw(rectangle_top);
        window_add.draw(rectangle_bottom);
        break;

    case 2:
        window_change.draw(rectangle_left);
        window_change.draw(rectangle_right);
        window_change.draw(rectangle_top);

```

```
    window_change.draw(rectangle_bottom);  
    break;
```

case 3:

```
    window_delete.draw(rectangle_left);  
    window_delete.draw(rectangle_right);  
    window_delete.draw(rectangle_top);  
    window_delete.draw(rectangle_bottom);  
    break;
```

case 4:

```
    window_login.draw(rectangle_left);  
    window_login.draw(rectangle_right);  
    window_login.draw(rectangle_top);  
    window_login.draw(rectangle_bottom);  
    break;
```

default:

```
    window_main.draw(rectangle_left);  
    window_main.draw(rectangle_right);  
    window_main.draw(rectangle_top);  
    window_main.draw(rectangle_bottom);  
    break;
```

```
}
```

```
}
```

```
void create_it(int WINDOW_N, int X, int Y, int W, int H, int THICKNESS,  
sf::Color COLOR){
```

```
    window_n = WINDOW_N;
```

```

x = X;
y = Y;
w = W;
h = H;
thickness = THICKNESS;
color = COLOR;
rectangle_left.setFillColor(color);
rectangle_right.setFillColor(color);
rectangle_top.setFillColor(color);
rectangle_bottom.setFillColor(color);
rectangle_left.setSize(sf::Vector2f(thickness, h+thickness));
rectangle_right.setSize(sf::Vector2f(thickness, h+thickness));
rectangle_top.setSize(sf::Vector2f(w+thickness, thickness));
rectangle_bottom.setSize(sf::Vector2f(w+thickness, thickness));
rectangle_left.setPosition(x-thickness, y-thickness);
rectangle_right.setPosition(x+w, y);
rectangle_top.setPosition(x, y-thickness);
rectangle_bottom.setPosition(x-thickness, y+h);
}

BorderBox(){ }
~BorderBox(){ }

};

#endif

```

3. Файл button.h

```

#ifndef BUTTON_H
#define BUTTON_H

```



```

class Button{
private:
    int window_n;
    int x = 0;
    int y = 0;
    int w = 0;
    int h = 0;
    int thickness = 0;
    int textsize = 20;
    int otstup_left = 0;
    int otstup_top = 0;
    wstring string = L"";
    sf::Color color_border = sf::Color(150,137,14);
    sf::Color color_fill = sf::Color(237,220,55);
    sf::Color color_fill_pressing = sf::Color(250,243,158);
    RectangleBox rectanglebox;
    BorderBox borderbox;
    TextBox textbox;
    ClickedArea clickedarea;

public:

    void set_window_n(int WINDOW_N) { window_n = WINDOW_N; }

    int get_x(){ return x; }
    int get_y(){ return y; }
    int get_w(){ return w; }
    int get_h(){ return h; }
    int get_thickness(){ return thickness; }
    int get_textsize(){ return textsize; }
    int get_otstup_left(){ return otstup_left; }
    int get_otstup_top(){ return otstup_top; }
    wstring get_string(){ return string; }
    sf::Color get_color_border(){ return color_border; }
    sf::Color get_color_fill(){ return color_fill; }
    sf::Color get_color_fill_pressing(){ return color_fill_pressing; }

    bool is_active(sf::Vector2f MOUSE_POSITION = mouse_position){
        return clickedarea.is_active(MOUSE_POSITION);
    }

    bool is_clicked(sf::Vector2f MOUSE_POSITION = mouse_position){
        return clickedarea.is_clicked(MOUSE_POSITION);
    }
}

```

```

void update_it(sf::Vector2f MOUSE_POSITION = mouse_position){
    if(clickedarea.is_active(MOUSE_POSITION)){
        rectanglebox.change_color(color_fill_pressing);

        switch (window_n) {
        case 1:
            window_add.setMouseCursor(cursor_hand);
            break;

        case 2:
            window_change.setMouseCursor(cursor_hand);
            break;

        case 3:
            window_delete.setMouseCursor(cursor_hand);
            break;

        case 4:
            window_login.setMouseCursor(cursor_hand);
            break;

        default:
            window_main.setMouseCursor(cursor_hand);
            break;
        }

    } else {
        rectanglebox.change_color(color_fill);
    }
}

void update_without_cursor(sf::Vector2f MOUSE_POSITION =
mouse_position){
    if(clickedarea.is_active(MOUSE_POSITION)){
        rectanglebox.change_color(color_fill_pressing);
    } else {
        rectanglebox.change_color(color_fill);
    }
}

void draw_it(){
    rectanglebox.draw_it();
    borderbox.draw_it();
    textbox.draw_it();
}

```

```

void create_it(int WINDOW_N, int X, int Y, int W, int H, int THICKNESS,
int TEXTSIZE, sf::Color COLOR_BORDER, sf::Color COLOR_FILL, sf::Color
COLOR_FILL_PRESSING, int OTSTUP_LEFT, int OTSTUP_TOP, wstring
STRING){
    window_n = WINDOW_N;
    x = X;
    y = Y;
    w = W;
    h = H;
    textsize = TEXTSIZE;
    thickness = THICKNESS;
    color_border = COLOR_BORDER;
    color_fill = COLOR_FILL;
    color_fill_pressing = COLOR_FILL_PRESSING;
    otstup_left = OTSTUP_LEFT;
    otstup_top = OTSTUP_TOP;
    string = STRING;
    rectanglebox.create_it(window_n, x, y, w, h, color_fill);
    borderbox.create_it(window_n, x+thickness, y+thickness, w-
2*thickness, h-2*thickness, thickness, color_border);
    textbox.create_it(window_n, x + thickness + otstup_left, y + thickness
+ otstup_top, textsize, string);
    clickedarea.create_it(window_n, x, y, w, h);
}

Button(){ }
~Button(){ }
};

#endif

```

4. Файл choosebox.h

```

#ifndef CHOOSEBOX_H
#define CHOOSEBOX_H

class ChooseBox{
private:
    int window_n;
    int x = 0;
    int y = 0;
    int w = 0;
    int h_button = 0;

```

```

int h = 0;
int h_area = 0;
int borderthickness = 2;
int scrollbarthickness = 18;
bool choosing = false;
bool active = false;
int choosenum = 0;
sf::Color color_border = sf::Color(10,10,10);
sf::Color color_decor = sf::Color(149,194,148);
sf::Color color_decor_active = sf::Color(182,213,181);
sf::Color color_decor_border = sf::Color(84,130,53);
BorderBox borderbox;
RectangleBox rectanglebox;
RectangleBox rectanglebox_decor;
BorderBox borderbox_decor;
TextBox textbox;
ClickedArea clickedarea;
ScrollList scrolllist;
vector<Button> button_list;
vector<wstring> button_texts;
int number;
int h_text;
sf::CircleShape triangle;

sf::View view;

```

public:

```

void set_window_n(int WINDOW_N) { window_n = WINDOW_N; }

int get_x(){ return x; }
int get_y(){ return y; }
int get_w(){ return w; }
int get_h(){ return h; }
int get_h_button(){ return h_button; }
int get_h_area(){ return h_area; }
int get_h_text(){ return h_text; }
int get_borderthickness(){ return borderthickness; }
int get_scrollbarthickness(){ return scrollbarthickness; }
bool get_choosing(){ return choosing; }
int get_choosenum(){ return choosenum; }
sf::Color get_color_border(){ return color_border; }
sf::Color get_color_decor(){ return color_decor; }
sf::Color get_color_decor_active(){ return color_decor_active; }
sf::Color get_color_decor_border(){ return color_decor_border; }

```

```

int get_number(){ return number; }

wstring get_choosestring(){
    return(button_list[choosenum].get_string());
}

bool is_active(){
    check_it();
    return active;
}

bool is_choosing(){
    check_it();
    return choosing;
}

    if(events_mouse.L_mouse && !(mouse_position.x >= x &&
mouse_position.x <= x+w && mouse_position.y >= y && mouse_position.y <=
y+h)){ choosing = false; }

        if(!choosing){
            if(rectanglebox.is_active()){ active = true; }
            else { active = false; }
        } else {
            if(mouse_position.x >= x && mouse_position.x <= x+w-
scrollbarthickness && mouse_position.y >= y && mouse_position.y <= y+h){
active = true; }
            else { active = false; }
        }
    }

void draw_it(){
    if(!choosing){
        rectanglebox.draw_it();
        rectanglebox_decor.draw_it();
        borderbox_decor.draw_it();
        textbox.draw_it();
        switch (window_n) {
        case 1:
            window_add.draw(triangle);
            break;

        case 2:
            window_change.draw(triangle);

```

```

        break;

    case 3:
        window_delete.draw(triangle);
        break;

    case 4:
        window_login.draw(triangle);
        break;

    default:
        window_main.draw(triangle);
        break;
    }
} else {
    borderbox.draw_it();
    scrolllist.draw_it();
    for(int i = 0; i < button_list.size(); i++){
        button_list[i].draw_it();
    }
    switch (window_n) {
    case 1:
        window_add.setView(view_add);
        break;

    case 2:
        window_change.setView(view_change);
        break;

    case 3:
        window_delete.setView(view_delete);
        break;

    case 4:
        window_login.setView(view_delete);
        break;

    default:
        window_main.setView(view_main);
        break;
    }
}
}

```

```

void create_it(int WINDOW_N, int X, int Y, int W, int H_BUTTON, int H,
int SCROLLBARTHICKNESS, int H_TEXT, vector<wstring>
BUTTON_TEXTS){
    window_n = WINDOW_N;
    x = X;
    y = Y;
    w = W;
    h_button = H_BUTTON;
    h = H;
    h_text = H_TEXT;
    scrollbarthickness = SCROLLBARTHICKNESS;
    button_texts = BUTTON_TEXTS;
    number = button_texts.size();
    button_list.resize(number);
    if(h_text*(number) > h){
        h_area = h_text*(number);
    } else {
        h_area = h;
    }

    if(window_n == 1 || window_n == 2 || window_n == 3){
        color_decor = sf::Color(237,220,55);
        color_decor_active = sf::Color(250,243,158);
        color_decor_border = sf::Color(150,137,14);
    }

    borderbox.create_it(window_n, x, y, w, h, borderthickness,
color_border);
    rectanglebox.create_it(window_n, x, y, w, h_button,
sf::Color(255,255,255));
    rectanglebox_decor.create_it(window_n, x+w-scrollbarthickness, y,
scrollbarthickness, h_button, color_decor);
    borderbox_decor.create_it(window_n, x+w-scrollbarthickness+2,
y+2, scrollbarthickness-4, h_button-4, 2, color_decor_border);
    clickedarea.create_it(window_n, x, y, w, h_button);
    scrolllist.create_it(window_n, x, y, w, h, h_area, scrollbarthickness);
    textbox.create_it(window_n, x+3, y, 20, L"");

    for(int i = 0; i < number; i++){
        button_list[i].create_it(window_n, 0, h_text*i, w-
scrollbarthickness, h_text, 0, 20, sf::Color(255,255,255), sf::Color(255,255,255),
sf::Color(220,220,220), 3, 0, button_texts[i]);
    }
}

```

```

        triangle.setRadius(7);
        triangle.setPointCount(3);
        triangle.setRotation(180);
        triangle.setFillColor(sf::Color(50,50,50));
        triangle.setPosition(x+w-2, y+h_button-6);
    }

    ChooseBox(){ }
    ~ChooseBox(){ }
};

#endif

```

5. Файл clickedarea.h

```

#ifndef CLICKEDAREA_H
#define CLICKEDAREA_H

class ClickedArea{
private:
    int window_n;
    bool active = true; // активный ли для нажатия
    bool pressing = false; // нажат ли
    bool grabbing = false; // зажат ли
    bool clicked = false; // срабатывает 1 раз после нажатия
    bool grabbed = false; // срабатывает 1 раз при зажатии
    int x = 0;
    int y = 0;
    int w = 0;
    int h = 0;

public:

    void set_window_n(int WINDOW_N) { window_n = WINDOW_N; }

    int get_x(){ return x; }
    int get_y(){ return y; }
    int get_w(){ return w; }
    int get_h(){ return h; }

    bool is_active(sf::Vector2f MOUSE_POSITION = mouse_position){
        check_it(MOUSE_POSITION);
        return active;
    }

```



```

    }

    bool is_pressing(sf::Vector2f MOUSE_POSITION = mouse_position){
        check_it(MOUSE_POSITION);
        return pressing;
    }

    bool is_grabbing(sf::Vector2f MOUSE_POSITION = mouse_position){
        check_it(MOUSE_POSITION);
        return grabbing;
    }

    bool is_clicked(sf::Vector2f MOUSE_POSITION = mouse_position){
        check_it(MOUSE_POSITION);
        return clicked;
    }

    bool is_grabbed(sf::Vector2f MOUSE_POSITION = mouse_position){
        check_it(MOUSE_POSITION);
        return grabbed;
    }

    void change_position(int X, int Y){
        x = X;
        y = Y;
    }

    void change_size(int W, int H){
        w = W;
        h = H;
    }

    void check_it(sf::Vector2f MOUSE_POSITION = mouse_position){
        if (MOUSE_POSITION.x >= x && MOUSE_POSITION.x <= x + w
        && MOUSE_POSITION.y >= y && MOUSE_POSITION.y <= y + h){
            active = true;
        } else {
            active = false;
        }

        if (events_mouse.L_mouse){
            if (active){
                pressing = true;
                if(!grabbing){
                    grabbed = true;

```

```

        } else {
            grabbed = false;
        }
        grabbing = true;
    } else {
        pressing = false;
    }
} else {
    if (pressing){
        clicked = true;
        pressing = false;
    } else {
        clicked = false;
    }
    grabbing = false;
    grabbed = false;
}
}

void create_it(int WINDOW_N, int X, int Y, int W, int H){
    window_n = WINDOW_N;
    x = X;
    y = Y;
    w = W;
    h = H;
}

ClickedArea(){ }
~ClickedArea(){ }
};

#endif

```

6. Файл data.h

```

#ifndef DATA_H
#define DATA_H

class Data{
public:

```

File file;

```
void read_file(){
    data_file.resize(file.length_it());
    data_file = file.read_it();
    data_wstring.resize(file.resize_wstring_data(data_file));
    data_wstring = file.read_wstring_data(data_file);
}
```

```
void find_by_parameters(wstring cityfirstwstring, wstring citysecondwstring,
wstring transportwstring, wstring datefirstwstring, wstring dateseccondwstring,
wstring pricefirstwstring, wstring priceseccondwstring){
```

```
    data_struct data_find_min = {-1, -1, " ", 0, 0, 0, 0, " ", 0, " ", 0};
```

```
    data_struct data_find_max = {-1, -1, " ", 0, 0, 0, 0, " ", 0, " ", 0};
```

```
    if(cityfirstwstring == L"-----") {
```

```
        data_find_min.cityfirst = "-----";
```

```
        data_find_max.cityfirst = "-----";
```

```
    } else {
```

```
        for (int i = 0; i < citynames.size(); i++){
```

```
            if(cityfirstwstring == citynames[i].name){
```

```
                data_find_min.cityfirst = citynames[i].code;
```

```
                data_find_max.cityfirst = citynames[i].code;
```

```
            }
```

```
        }
```

```
    }
```

```
    if(citysecondwstring == L"-----") {
```

```
        data_find_min.citysecond = "-----";
```

```

        data_find_max.citysecond = "-----";
    } else {
        for (int i = 0; i < citynames.size(); i++){
            if(citysecondwstring == citynames[i].name){
                data_find_min.citysecond = citynames[i].code;
                data_find_max.citysecond = citynames[i].code;
            }
        }
    }
    if(transportwstring == L"-----") {
        data_find_min.transport = "-----";
        data_find_max.transport = "-----";
    } else {
        for (int i = 0; i < transportnames.size(); i++){
            if(transportwstring == transportnames[i].name){
                data_find_min.transport = transportnames[i].code;
                data_find_max.transport = transportnames[i].code;
            }
        }
    }
    if(datefirstwstring == L"") {
        data_find_min.dayfirst = -1;
        data_find_max.dayfirst = -1;
        data_find_min.monthfirst = -1;
        data_find_max.monthfirst = -1;
    } else {
        for(int i = 0; i < wstringtoint.size(); i++){
            if(datefirstwstring[0] == wstringtoint[i].name){
                data_find_min.dayfirst += 10*wstringtoint[i].code;

```

```

        data_find_max.dayfirst +=
10*wstringtoint[i].code;
    }
    if(datefirstwstring[1] == wstringtoint[i].name){
        data_find_min.dayfirst += wstringtoint[i].code;
        data_find_max.dayfirst += wstringtoint[i].code;
    }
    if(datefirstwstring[3] == wstringtoint[i].name){
        data_find_min.monthfirst +=
10*wstringtoint[i].code;
        data_find_max.monthfirst +=
10*wstringtoint[i].code;
    }
    if(datefirstwstring[4] == wstringtoint[i].name){
        data_find_min.monthfirst += wstringtoint[i].code;
        data_find_max.monthfirst += wstringtoint[i].code;
    }
}

if(datesecondwstring == L"") {
    data_find_min.daysecond = -1;
    data_find_max.daysecond = -1;
    data_find_min.monthsecond = -1;
    data_find_max.monthsecond = -1;
} else {
    for(int i = 0; i < wstringtoint.size(); i++){
        if(datesecondwstring[0] == wstringtoint[i].name){
            data_find_min.daysecond +=
10*wstringtoint[i].code;

```

```

        data_find_max.daysecond +=
10*wstringtoint[i].code;
    }
    if(datesecondwstring[1] == wstringtoint[i].name){
        data_find_min.daysecond += wstringtoint[i].code;
        data_find_max.daysecond += wstringtoint[i].code;
    }
    if(datesecondwstring[3] == wstringtoint[i].name){
        data_find_min.monthsecond +=
10*wstringtoint[i].code;
        data_find_max.monthsecond +=
10*wstringtoint[i].code;
    }
    if(datesecondwstring[4] == wstringtoint[i].name){
        data_find_min.monthsecond +=
wstringtoint[i].code;
        data_find_max.monthsecond +=
wstringtoint[i].code;
    }
}

if(pricefirstwstring == L"") {
    data_find_min.price = -1;
} else {
    if(pricefirstwstring.size() == 1){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(pricefirstwstring[0] == wstringtoint[i].name){
                data_find_min.price += wstringtoint[i].code;
            }
        }
    }
}

```

```

        }
    }
    if(pricefirstwstring.size() == 2){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(pricefirstwstring[0] == wstringtoint[i].name){
                data_find_min.price +=
10*wstringtoint[i].code;
            }
            if(pricefirstwstring[1] == wstringtoint[i].name){
                data_find_min.price += wstringtoint[i].code;
            }
        }
    }
    if(pricefirstwstring.size() == 3){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(pricefirstwstring[0] == wstringtoint[i].name){
                data_find_min.price +=
100*wstringtoint[i].code;
            }
            if(pricefirstwstring[1] == wstringtoint[i].name){
                data_find_min.price +=
10*wstringtoint[i].code;
            }
            if(pricefirstwstring[2] == wstringtoint[i].name){
                data_find_min.price += wstringtoint[i].code;
            }
        }
    }
}

```

```

        if(pricesecondwstring == L"") {
            data_find_max.price = -1;
        } else {
            if(pricesecondwstring.size() == 1){
                for(int i = 0; i < wstringtoint.size(); i++){
                    if(pricesecondwstring[0] == wstringtoint[i].name){
                        data_find_max.price +=
wstringtoint[i].code;
                    }
                }
            }
            if(pricesecondwstring.size() == 2){
                for(int i = 0; i < wstringtoint.size(); i++){
                    if(pricesecondwstring[0] == wstringtoint[i].name){
                        data_find_max.price +=
10*wstringtoint[i].code;
                    }
                    if(pricesecondwstring[1] == wstringtoint[i].name){
                        data_find_max.price +=
wstringtoint[i].code;
                    }
                }
            }
            if(pricesecondwstring.size() == 3){
                for(int i = 0; i < wstringtoint.size(); i++){
                    if(pricesecondwstring[0] == wstringtoint[i].name){
                        data_find_max.price +=
100*wstringtoint[i].code;
                    }
                    if(pricesecondwstring[1] == wstringtoint[i].name){

```



```

                                data_find_max.price +=
10*wstringtoint[i].code;
                                }
                                if(pricesecsecondwstring[2] == wstringtoint[i].name){
                                    data_find_max.price +=
wstringtoint[i].code;
                                }
                            }
                        }
                    }

                    data_find_min.number = -1;
                    data_find_max.number = -1;

                    data_wstring.resize(file.resize_wstring_data_find(data_file,
data_find_min, data_find_max));

                    data_wstring = file.read_wstring_data_find(data_file, data_find_min,
data_find_max);
                }

void sort_it(wstring sorttype){
    if(sorttype == L"По умолчанию"){
        data_file.resize(file.length_it());
        data_file = file.read_it();
        data_wstring.resize(file.resize_wstring_data(data_file));
        data_wstring = file.read_wstring_data(data_file);
    }
    if(sorttype == L"По цене"){
        data_wstring = file.sort_by_price(data_wstring);
    }
}

```

```

if(sorttype == L"По алфавиту (город 1)"){
    data_wstring = file.sort_by_alphabet_1(data_wstring);
}
if(sorttype == L"По алфавиту (город 2)"){
    data_wstring = file.sort_by_alphabet_2(data_wstring);
}
if(sorttype == L"По алфавиту (вид транспорта)"){
    data_wstring = file.sort_by_alphabet_3(data_wstring);
}
if(sorttype == L"По количеству билетов"){
    data_wstring = file.sort_by_tickets (data_wstring);
}
if(sorttype == L"По дате отправления"){
    data_wstring = file.sort_by_date_1 (data_wstring);
}
if(sorttype == L"По дате прибытия"){
    data_wstring = file.sort_by_date_2 (data_wstring);
}
}

```

```

void add_it(wstring cityfirstwstring, wstring datefirstwstring, wstring
datesecondwstring, wstring citysecondwstring, wstring pricewstring, wstring
transportwstring, wstring ticketwstring){

```

```

    string cityfirst;
    for (int i = 0; i < citynames.size(); i++){
        if(cityfirstwstring == citynames[i].name){
            cityfirst = citynames[i].code;
        }
    }
}

```

```

        int dayfirst = 0;
        int monthfirst = 0;
        for(int i = 0; i < wstringtoint.size(); i++){
            if(datefirstwstring[0] == wstringtoint[i].name){
                dayfirst += 10*wstringtoint[i].code;
            }
            if(datefirstwstring[1] == wstringtoint[i].name){
                dayfirst += wstringtoint[i].code;
            }
            if(datefirstwstring[3] == wstringtoint[i].name){
                monthfirst += 10*wstringtoint[i].code;
            }
            if(datefirstwstring[4] == wstringtoint[i].name){
                monthfirst += wstringtoint[i].code;
            }
        }
        int daysecond = 0;
        int monthsecond = 0;
        for(int i = 0; i < wstringtoint.size(); i++){
            if(datesecondwstring[0] == wstringtoint[i].name){
                daysecond += 10*wstringtoint[i].code;
            }
            if(datesecondwstring[1] == wstringtoint[i].name){
                daysecond += wstringtoint[i].code;
            }
            if(datesecondwstring[3] == wstringtoint[i].name){
                monthsecond += 10*wstringtoint[i].code;
            }
            if(datesecondwstring[4] == wstringtoint[i].name){

```

```

        monthsecond += wstringtoint[i].code;
    }
}

string citysecond;
for (int i = 0; i < citynames.size(); i++){
    if(citysecondwstring == citynames[i].name){
        citysecond = citynames[i].code;
    }
}

    int price;
    if(pricewstring.size() == 1){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(pricewstring[0] == wstringtoint[i].name){
                price += wstringtoint[i].code;
            }
        }
    }
    if(pricewstring.size() == 2){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(pricewstring[0] == wstringtoint[i].name){
                price += 10*wstringtoint[i].code;
            }
        }
    }
    if(pricewstring[1] == wstringtoint[i].name){
        price += wstringtoint[i].code;
    }
}

}

```

```

if(pricewstring.size() == 3){
    for(int i = 0; i < wstringtoint.size(); i++){
        if(pricewstring[0] == wstringtoint[i].name){
            price += 100*wstringtoint[i].code;
        }
        if(pricewstring[1] == wstringtoint[i].name){
            price += 10*wstringtoint[i].code;
        }
        if(pricewstring[2] == wstringtoint[i].name){
            price += wstringtoint[i].code;
        }
    }
}

char transport;
for (int i = 0; i < transportnameschar.size(); i++){
    if(transportwstring == transportnameschar[i].name){
        transport = transportnameschar[i].code;
        transport = transportnameschar[i].code;
    }
}

int tickets;
if(ticketswstring.size() == 1){
    for(int i = 0; i < wstringtoint.size(); i++){
        if(ticketswstring[0] == wstringtoint[i].name){
            tickets += wstringtoint[i].code;
        }
    }
}

```

```

    }
    if(ticketswstring.size() == 2){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(ticketswstring[0] == wstringtoint[i].name){
                tickets += 10*wstringtoint[i].code;
            }
            if(ticketswstring[1] == wstringtoint[i].name){
                tickets += wstringtoint[i].code;
            }
        }
    }
    if(ticketswstring.size() == 3){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(ticketswstring[0] == wstringtoint[i].name){
                tickets += 100*wstringtoint[i].code;
            }
            if(ticketswstring[1] == wstringtoint[i].name){
                tickets += 10*wstringtoint[i].code;
            }
            if(ticketswstring[2] == wstringtoint[i].name){
                tickets += wstringtoint[i].code;
            }
        }
    }

    file.add_it(cityfirst, dayfirst, monthfirst, daysecond, monthsecond,
citysecond, price, transport, tickets);
    read_file();
}

```

```
void change_it(wstring numwstring, wstring cityfirstwstring, wstring
datefirstwstring, wstring datesecndwstring, wstring citysecondwstring, wstring
pricewstring, wstring transportwstring, wstring ticketwstring){
```

```
    int line_num = 0;
    if(numwstring.size() == 1){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(numwstring[0] == wstringtoint[i].name){
                line_num += wstringtoint[i].code;
            }
        }
    }
    if(numwstring.size() == 2){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(numwstring[0] == wstringtoint[i].name){
                line_num += 10*wstringtoint[i].code;
            }
        }
        if(numwstring[1] == wstringtoint[i].name){
            line_num += wstringtoint[i].code;
        }
    }
    if(numwstring.size() == 3){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(numwstring[0] == wstringtoint[i].name){
                line_num += 100*wstringtoint[i].code;
            }
            if(numwstring[1] == wstringtoint[i].name){
                line_num += 10*wstringtoint[i].code;
            }
        }
    }
}
```

```

        }
        if(numwstring[2] == wstringtoint[i].name){
            line_num += wstringtoint[i].code;
        }
    }
}

int file_num = 0;
for (int i = 0; i < data_file.size(); i++) {
    if (data_file[i].num_line == line_num) {
        file_num = data_file[i].num_file;
    }
}

string cityfirst;
for (int i = 0; i < citynames.size(); i++){
    if(cityfirstwstring == citynames[i].name){
        cityfirst = citynames[i].code;
    }
}

int dayfirst = 0;
int monthfirst = 0;
for(int i = 0; i < wstringtoint.size(); i++){
    if(datefirstwstring[0] == wstringtoint[i].name){
        dayfirst += 10*wstringtoint[i].code;
    }
    if(datefirstwstring[1] == wstringtoint[i].name){

```



```

        dayfirst += wstringtoint[i].code;
    }
    if(datefirstwstring[3] == wstringtoint[i].name){
        monthfirst += 10*wstringtoint[i].code;
    }
    if(datefirstwstring[4] == wstringtoint[i].name){
        monthfirst += wstringtoint[i].code;
    }
}

int daysecond = 0;
int monthsecond = 0;
for(int i = 0; i < wstringtoint.size(); i++){
    if(datesecondwstring[0] == wstringtoint[i].name){
        daysecond += 10*wstringtoint[i].code;
    }
    if(datesecondwstring[1] == wstringtoint[i].name){
        daysecond += wstringtoint[i].code;
    }
    if(datesecondwstring[3] == wstringtoint[i].name){
        monthsecond += 10*wstringtoint[i].code;
    }
    if(datesecondwstring[4] == wstringtoint[i].name){
        monthsecond += wstringtoint[i].code;
    }
}

string citysecond;
for (int i = 0; i < citynames.size(); i++){

```

```

    if(citysecondwstring == citynames[i].name){
        citysecond = citynames[i].code;
    }
}

    int price;
    if(pricewstring.size() == 1){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(pricewstring[0] == wstringtoint[i].name){
                price += wstringtoint[i].code;
            }
        }
    }
    if(pricewstring.size() == 2){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(pricewstring[0] == wstringtoint[i].name){
                price += 10*wstringtoint[i].code;
            }
        }
    }
    if(pricewstring[1] == wstringtoint[i].name){
        price += wstringtoint[i].code;
    }
}

    if(pricewstring.size() == 3){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(pricewstring[0] == wstringtoint[i].name){
                price += 100*wstringtoint[i].code;
            }
            if(pricewstring[1] == wstringtoint[i].name){
                price += 10*wstringtoint[i].code;
            }
        }
    }
}

```

```

        }
        if(pricewstring[2] == wstringtoint[i].name){
            price += wstringtoint[i].code;
        }
    }
}

char transport;
for (int i = 0; i < transportnameschar.size(); i++){
    if(transportwstring == transportnameschar[i].name){
        transport = transportnameschar[i].code;
        transport = transportnameschar[i].code;
    }
}

int tickets;
if(ticketswstring.size() == 1){
    for(int i = 0; i < wstringtoint.size(); i++){
        if(ticketswstring[0] == wstringtoint[i].name){
            tickets += wstringtoint[i].code;
        }
    }
}

if(ticketswstring.size() == 2){
    for(int i = 0; i < wstringtoint.size(); i++){
        if(ticketswstring[0] == wstringtoint[i].name){
            tickets += 10*wstringtoint[i].code;
        }
        if(ticketswstring[1] == wstringtoint[i].name){
            tickets += wstringtoint[i].code;
        }
    }
}

```

```

    }
}
if(ticketswstring.size() == 3){
    for(int i = 0; i < wstringtoint.size(); i++){
        if(ticketswstring[0] == wstringtoint[i].name){
            tickets += 100*wstringtoint[i].code;
        }
        if(ticketswstring[1] == wstringtoint[i].name){
            tickets += 10*wstringtoint[i].code;
        }
        if(ticketswstring[2] == wstringtoint[i].name){
            tickets += wstringtoint[i].code;
        }
    }
}

```

```

    file.change_it(file_num, cityfirst, dayfirst, monthfirst, daysecond,
monthsecond, citysecond, price, transport, tickets);
    read_file();
}

```

```

void delete_it(wstring numwstring){
    int line_num = 0;
    if(numwstring.size() == 1){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(numwstring[0] == wstringtoint[i].name){
                line_num += wstringtoint[i].code;
            }
        }
    }
}

```

```

    }

    if(numwstring.size() == 2){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(numwstring[0] == wstringtoint[i].name){
                line_num += 10*wstringtoint[i].code;
            }
            if(numwstring[1] == wstringtoint[i].name){
                line_num += wstringtoint[i].code;
            }
        }
    }

    if(numwstring.size() == 3){
        for(int i = 0; i < wstringtoint.size(); i++){
            if(numwstring[0] == wstringtoint[i].name){
                line_num += 100*wstringtoint[i].code;
            }
            if(numwstring[1] == wstringtoint[i].name){
                line_num += 10*wstringtoint[i].code;
            }
            if(numwstring[2] == wstringtoint[i].name){
                line_num += wstringtoint[i].code;
            }
        }
    }

    int file_num = 0;
    for (int i = 0; i < data_file.size(); i++) {
        if (data_file[i].num_line == line_num) {
            file_num = data_file[i].num_file;
        }
    }

```

```

    }
}

    file.delete_it(file_num);
    read_file();
}

void create_it(){
    data_file.resize(file.length_it());
    data_file = file.read_it();
    data_wstring.resize(file.resize_wstring_data(data_file));
    data_wstring = file.read_wstring_data(data_file);

    citynames_for_chooseboxes.resize(citynames.size()+1);
    citynames_for_chooseboxes[0] = L"-----";
    for(int i = 1; i < citynames_for_chooseboxes.size(); i++){
        citynames_for_chooseboxes[i] = citynames[i-1].name;
    }
    transportnames_for_chooseboxes.resize(transportnames.size()+1);
    transportnames_for_chooseboxes[0] = L"-----";
    for(int i = 1; i < transportnames_for_chooseboxes.size(); i++){
        transportnames_for_chooseboxes[i] = transportnames[i-1].name;
    }

    add_change_citynames_for_chooseboxes.resize(citynames.size());
    for(int i = 0; i < add_change_citynames_for_chooseboxes.size(); i++){
        add_change_citynames_for_chooseboxes[i] = citynames[i].name;
    }
    add_change_transportnames_for_chooseboxes.resize(transportnames.size());

```

```

        for(int i = 0; i < add_change_transportnames_for_chooseboxes.size(); i++){
            add_change_transportnames_for_chooseboxes[i] =
transportnames[i].name;
        }
    }

    Data(){
        file.create_it("materials/travel.txt");
    }
    ~Data(){}
};

#endif

```

7. Файл file.h

```

#ifndef FILE_H
#define FILE_H

class File{
private:
    std::string filename = "materials\\travel.txt";
    std::fstream file;
    int filelength = 0;

public:

    int length_it() {
        file.open(filename);
    }

```

```

filelength = -1;

file.seekg(0, std::ios::end);
if (file.tellg() == 0) {
    filelength = 0;
    return 0;
} else {
    file.seekg(0, std::ios::beg);
}

while (file) {
    filelength++;
    std::string a;
    std::getline(file, a);
}
file.close();
return filelength;
}

std::vector<data_struct> read_it(){
    std::vector<data_struct> vector;
    vector.resize(filelength);
    file.open(filename);
    int num_lines = 1;
    for (unsigned int i = 0; i < vector.size(); i++) {
        file >> vector[i].cityfirst;
        file >> vector[i].dayfirst;
        file >> vector[i].monthfirst;
    }
}

```



```

    file >> vector[i].daysecond;
    file >> vector[i].monthsecond;
    file >> vector[i].citysecond;
    file >> vector[i].price;
    file >> vector[i].transport;
    file >> vector[i].number;
    vector[i].num_file = i;
    if (vector[i].cityfirst == "#") {
        vector[i].num_line = -1;
    } else {
        vector[i].num_line = num_lines;
        num_lines++;
    }
}
file.close();
return vector;
}

```

```

void add_it(std::string cityfirst, int dayfirst, int monthfirst, int daysecond, int
monthsecond, std::string citysecond, int price, char transport, int number) {
    file.open(filename);
    file.seekp(0, std::ios::end);
    if (filelength > 0) {
        file << "\n";
    }
    file << cityfirst;
    for (int i = cityfirst.length(); i < 4; i++) { file << " "; }
    file << dayfirst;
    for (int i = std::to_string(dayfirst).length(); i < 3; i++) { file << " "; }
}

```

```

file << monthfirst;
for (int i = std::to_string(monthfirst).length(); i < 3; i++) { file << " "; }
file << daysecond;
for (int i = std::to_string(daysecond).length(); i < 3; i++) { file << " "; }
file << monthsecond;
for (int i = std::to_string(monthsecond).length(); i < 3; i++) { file << " "; }
file << citysecond;
for (int i = citysecond.length(); i < 4; i++) { file << " "; }
file << price;
for (int i = std::to_string(price).length(); i < 5; i++) { file << " "; }
file << transport << " ";
file << number;
for (int i = std::to_string(number).length(); i < 8; i++) { file << " "; }
file.close();
}

```

```

void change_it(int num_file, std::string cityfirst, int dayfirst, int monthfirst, int
daysecond, int monthsecond, std::string citysecond, int price, char transport, int
number) {

```

```

    file.open(filename);
    file.seekp((num_file)*37, std::ios::beg);
    file << cityfirst;
    for (int i = cityfirst.length(); i < 4; i++) { file << " "; }
    file << dayfirst;
    for (int i = std::to_string(dayfirst).length(); i < 3; i++) { file << " "; }
    file << monthfirst;
    for (int i = std::to_string(monthfirst).length(); i < 3; i++) { file << " "; }
    file << daysecond;
    for (int i = std::to_string(daysecond).length(); i < 3; i++) { file << " "; }

```

```

file << monthsecond;
for (int i = std::to_string(monthsecond).length(); i < 3; i++) { file << " "; }
file << citysecond;
for (int i = citysecond.length(); i < 4; i++) { file << " "; }
file << price;
for (int i = std::to_string(price).length(); i < 5; i++) { file << " "; }
file << transport << " ";
file << number;
for (int i = std::to_string(number).length(); i < 8; i++) { file << " "; }
file.close();
}

```

// ----- удалить строчку -----

```

void delete_it(int num_file) {
    file.open(filename);
    file.seekp((num_file)*37, std::ios::beg);
    file << "# 0 0 0 0 # 0 # 0 ";
    file.close();
}

if(
    (data_find_min.cityfirst == "-----" || data[i].cityfirst ==
data_find_min.cityfirst) &&
    (data_find_min.citysecond == "-----" || data[i].citysecond ==
data_find_min.citysecond) &&
    (data_find_min.dayfirst == -1 || data[i].dayfirst >=
data_find_min.dayfirst) &&
    (data_find_max.dayfirst == -1 || data[i].dayfirst <=
data_find_max.dayfirst) &&

```

```

        (data_find_min.monthfirst == -1 || data[i].monthfirst >=
data_find_min.monthfirst) &&

        (data_find_max.monthfirst == -1 || data[i].monthfirst <=
data_find_max.monthfirst) &&

        (data_find_min.daysecond == -1 || data[i].daysecond >=
data_find_min.daysecond) &&

        (data_find_max.daysecond == -1 || data[i].daysecond <=
data_find_max.daysecond) &&

        (data_find_min.monthsecond == -1 || data[i].monthsecond >=
data_find_min.monthsecond) &&

        (data_find_max.monthsecond == -1 || data[i].monthsecond <=
data_find_max.monthsecond) &&

        (data_find_min.transport == "-----" || data[i].transport ==
data_find_min.transport) &&

        (data_find_min.price == -1 || data[i].price >= data_find_min.price)
&&

        (data_find_max.price == -1 || data[i].price <= data_find_max.price)
&&

        (data_find_min.number == -1 || data[i].number >=
data_find_min.number) &&

        (data_find_max.number == -1 || data[i].number <=
data_find_max.number)

    ){

        howmany++;

    }

    return howmany;

}

vector<data_wstring_struct> read_wstring_data_find(vector<data_struct> data,
data_struct data_find_min, data_struct data_find_max){

    vector<data_wstring_struct> data_wstring;

    if(

```

```

        (data_find_min.cityfirst == "-----" || data[i].cityfirst ==
data_find_min.cityfirst) &&

        (data_find_min.citysecond == "-----" || data[i].citysecond ==
data_find_min.citysecond) &&

        (data_find_min.dayfirst == -1 || data[i].dayfirst >=
data_find_min.dayfirst) &&

        (data_find_max.dayfirst == -1 || data[i].dayfirst <=
data_find_max.dayfirst) &&

        (data_find_min.monthfirst == -1 || data[i].monthfirst >=
data_find_min.monthfirst) &&

        (data_find_max.monthfirst == -1 || data[i].monthfirst <=
data_find_max.monthfirst) &&

        (data_find_min.daysecond == -1 || data[i].daysecond >=
data_find_min.daysecond) &&

        (data_find_max.daysecond == -1 || data[i].daysecond <=
data_find_max.daysecond) &&

        (data_find_min.monthsecond == -1 || data[i].monthsecond >=
data_find_min.monthsecond) &&

        (data_find_max.monthsecond == -1 || data[i].monthsecond <=
data_find_max.monthsecond) &&

        (data_find_min.transport == "-----" || data[i].transport ==
data_find_min.transport) &&

        (data_find_min.price == -1 || data[i].price >= data_find_min.price)
&&

        (data_find_max.price == -1 || data[i].price <= data_find_max.price)
&&

        (data_find_min.number == -1 || data[i].number >=
data_find_min.number) &&

        (data_find_max.number == -1 || data[i].number <=
data_find_max.number)

    ){

        data_wstring[k].num_line = to_wstring(data[i].num_line);
        for(int j = 0; j < citynames.size(); j++){

```

```

        if (data[i].cityfirst == citynames[j].code) {
            data_wstring[k].cityfirst = citynames[j].name;
        }
    }
    for(int j = 0; j < citynames.size(); j++){
        if (data[i].citysecond == citynames[j].code) {
            data_wstring[k].citysecond = citynames[j].name;
        }
    }

    data_wstring[k].datefirst = to_wstring(data[i].dayfirst) + L"." +
to_wstring(data[i].monthfirst) + L".2023";

    data_wstring[k].datesecond = to_wstring(data[i].daysecond) + L"." +
to_wstring(data[i].monthsecond) + L".2023";

    data_wstring[k].number = to_wstring(data[i].number);
    data_wstring[k].price = to_wstring(data[i].price);
    for(int j = 0; j < transportnames.size(); j++){
        if (data[i].transport == transportnames[j].code) {
            data_wstring[k].transport = transportnames[j].name;
        }
    }
    k++;
}

return data_wstring;
}

```

```

vector<data_wstring_struct> sort_by_price(vector<data_wstring_struct>
DATA_WSTRING){

```

```

    vector<data_wstring_struct> data_wstring = DATA_WSTRING;

```

```

    data_wstring_struct a;

```

```

for (int i = 0; i < data_wstring.size() - 1; i++) {
    for (int j = 0; j < data_wstring.size() - i - 1; j++) {
        if (data_wstring[j].price > data_wstring[j + 1].price) {
            a = data_wstring[j];
            data_wstring[j] = data_wstring[j + 1];
            data_wstring[j + 1] = a;
        }
    }
}

return data_wstring;
}

vector<data_wstring_struct> sort_by_alphabet_1(vector<data_wstring_struct>
DATA_WSTRING){
    vector<data_wstring_struct> data_wstring = DATA_WSTRING;

    data_wstring_struct a;
    for (int i = 0; i < data_wstring.size() - 1; i++) {
        for (int j = 0; j < data_wstring.size() - i - 1; j++) {
            if (data_wstring[j].cityfirst > data_wstring[j + 1].cityfirst) {
                a = data_wstring[j];
                data_wstring[j] = data_wstring[j + 1];
                data_wstring[j + 1] = a;
            }
        }
    }

    return data_wstring;
}

```

```

vector<data_wstring_struct> sort_by_alphabet_2(vector<data_wstring_struct>
DATA_WSTRING){
    vector<data_wstring_struct> data_wstring = DATA_WSTRING;

    data_wstring_struct a;
    for (int i = 0; i < data_wstring.size() - 1; i++) {
        for (int j = 0; j < data_wstring.size() - i - 1; j++) {
            if (data_wstring[j].citysecond > data_wstring[j + 1].citysecond) {
                a = data_wstring[j];
                data_wstring[j] = data_wstring[j + 1];
                data_wstring[j + 1] = a;
            }
        }
    }

    return data_wstring;
}

```

```

vector<data_wstring_struct> sort_by_alphabet_3(vector<data_wstring_struct>
DATA_WSTRING){
    vector<data_wstring_struct> data_wstring = DATA_WSTRING;

    data_wstring_struct a;
    for (int i = 0; i < data_wstring.size() - 1; i++) {
        for (int j = 0; j < data_wstring.size() - i - 1; j++) {
            if (data_wstring[j].transport > data_wstring[j + 1].transport) {
                a = data_wstring[j];
                data_wstring[j] = data_wstring[j + 1];
            }
        }
    }

    return data_wstring;
}

```



```

        data_wstring[j + 1] = a;
    }
}
}

```

```

    return data_wstring;
}

```

```

vector<data_wstring_struct> sort_by_tickets(vector<data_wstring_struct>
DATA_WSTRING){

```

```

    vector<data_wstring_struct> data_wstring = DATA_WSTRING;

```

```

    data_wstring_struct a;

```

```

    for (int i = 0; i < data_wstring.size() - 1; i++) {

```

```

        for (int j = 0; j < data_wstring.size() - i - 1; j++) {

```

```

            if (data_wstring[j].number > data_wstring[j + 1].number) {

```

```

                a = data_wstring[j];

```

```

                data_wstring[j] = data_wstring[j + 1];

```

```

                data_wstring[j + 1] = a;

```

```

            }

```

```

        }

```

```

    }

```

```

    return data_wstring;

```

```

}

```

```

vector<data_wstring_struct> sort_by_date_1(vector<data_wstring_struct>
DATA_WSTRING){

```

```

    vector<data_wstring_struct> data_wstring = DATA_WSTRING;

```

```

data_wstring_struct a;
for (int i = 0; i < data_wstring.size() - 1; i++) {
    for (int j = 0; j < data_wstring.size() - i - 1; j++) {
        if (data_wstring[j].datefirst > data_wstring[j + 1].datefirst) {
            a = data_wstring[j];
            data_wstring[j] = data_wstring[j + 1];
            data_wstring[j + 1] = a;
        }
    }
}

return data_wstring;
}

```

```

vector<data_wstring_struct> sort_by_date_2(vector<data_wstring_struct>
DATA_WSTRING){

```

```

    vector<data_wstring_struct> data_wstring = DATA_WSTRING;

```

```

data_wstring_struct a;
for (int i = 0; i < data_wstring.size() - 1; i++) {
    for (int j = 0; j < data_wstring.size() - i - 1; j++) {
        if (data_wstring[j].datesecond > data_wstring[j + 1].datesecond) {
            a = data_wstring[j];
            data_wstring[j] = data_wstring[j + 1];
            data_wstring[j + 1] = a;
        }
    }
}
}

```

```

        return data_wstring;
    }

    void create_it(std::string FILENAME){
        filename = FILENAME;
    }

    File(std::string FILENAME){
        filename = FILENAME;
    }
    File(){}
};

#endif

```

8. Файл imagebox.h

```

#ifndef IMAGEBOX_H
#define IMAGEBOX_H

class ImageBox{
private:
    int window_n;
    int x = 0;
    int y = 0;
    int w = 0;
    int h = 0;
    sf::Sprite sprite;
    sf::Texture texture;

public:

    void set_window_n(int WINDOW_N) { window_n = WINDOW_N; }

```

```

int get_x(){ return x; }
int get_y(){ return y; }
int get_w(){ return w; }
int get_h(){ return h; }

void change_image(string FILENAME){
    texture.loadFromFile(FILENAME);
    sprite.setTexture(texture);
}

void change_position(int X, int Y){
    x = X; y = Y;
    sprite.setPosition(x + w/2, y + h/2);
}

void change_size(int W, int H){
    w = W; h = H;
    sprite.setTextureRect(sf::IntRect(0, 0, w, h));
    sprite.setPosition(x + w/2, y + h/2);
    sprite.setOrigin(w/2, h/2);
}

bool is_active(sf::Vector2f MOUSE_POSITION = mouse_position){
    if(MOUSE_POSITION.x >= x && MOUSE_POSITION.x <= x + w
    && MOUSE_POSITION.y >= y && MOUSE_POSITION.y <= y + h){
        return true;
    } else {
        return false;
    }
}

void draw_it(){
    switch (window_n) {
        case 1:
            window_add.draw(sprite);
            break;

        case 2:
            window_change.draw(sprite);
            break;

        case 3:
            window_delete.draw(sprite);
            break;
    }
}

```

```

        case 4:
            window_login.draw(sprite);
            break;

        default:
            window_main.draw(sprite);
            break;
    }
}

void create_it(int WINDOW_N, int X, int Y, int W, int H, string
FILENAME){
    window_n = WINDOW_N;
    x = X;
    y = Y;
    w = W;
    h = H;
    texture.loadFromFile(FILENAME);
    sprite.setTextureRect(sf::IntRect(0, 0, w, h));
    sprite.setTexture(texture);
    sprite.setPosition(x + w/2, y + h/2);
    sprite.setOrigin(w/2, h/2);
}

ImageBox(){ }
~ImageBox(){ }
};

#endif

```