МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники (ФИТР)

Кафедра программного обеспечения информационных систем и технологий

**КУРСОВАЯ РАБОТА**

по дисциплине: ”Основы алгоритмизации и программирования”

на тему: ***”Расписание движения автобусов”***

Выполнил**:** ст. гр. 10701124 Ефименко А.В.

Приняла**:** ст**.**преподаватель И.М.Борисова

Минск 2025

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра программного обеспечения информационных систем и технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе**

**по дисциплине ”Основы алгоритмизации и программирования”**

Тема ***”Расписание движения автобусов”***

**Исполнитель**:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ефименко А. В.

(подпись)

**Студент 1 курса 10701124 группы**

**Руководитель** : \_\_\_\_\_\_\_\_\_\_\_\_\_Борисова И.М.

(подпись)

Минск 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире транспортная сфера остаётся одной из ключевых отраслей, обеспечивающих мобильность населения и функционирование экономики. С увеличением числа пассажиров и маршрутов возрастает потребность в удобных и эффективных системах управления расписанием движения автобусов, которые могли бы удовлетворить запросы как пассажиров, так и транспортных компаний. В этом контексте разработка программного обеспечения для управления расписанием движения автобусов становится важным элементом оптимизации работы транспортных организаций.

Целью курсовой работы является создание программного обеспечения для управления расписанием движения автобусов с использованием языка программирования C++.

Для достижения поставленной цели в процессе выполнения работы необходимо решить следующие задачи:

1. Изучить принципы организации расписания движения автобусов.
2. Провести анализ требований к программному обеспечению.
3. Разработать спецификацию и алгоритмы работы программы.
4. Реализовать программный код.
5. Выполнить тестирование и отладку созданного приложения.
6. Составить руководство пользователя и описание программы.

Результатом данной работы станет упрощение процесса управления расписанием движения автобусов в транспортной организации или компании.

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1 Постановка задачи

В справочной автовокзала хранится расписание движения автобусов. Для каждого рейса указаны: номер рейса; тип автобуса; пункт назначения; время отправления; время прибытия на конечный пункт. Индивидуальное задание: вывести информацию о всех рейсах, которыми можно воспользоваться для прибытия в пункт назначения не позднее чем за 12 часов до заданного времени (интересующее время прибытия вводится с клавиатуры

1.2. Исходные данные для курсовой работы

* Язык программирования С++.
* Среда разработки MicrosoftVisualStudio.
* Вид приложения – консольное.
* Парадигма программирования – процедурная .
* Способ организации данных – структуры (struct)
* Способ хранения данных – файлы

1.3. Требования к программной реализации:

* Каждая логически завершенная подзадача программы должна быть реализована в виде отдельной функции
* Допускается использовать классы string и vector, библиотеку algorithm.
* Возможность возврата назад (навигация).
* Запрос на выполнение необратимых действий
* Обратную связь с пользователем

2.ПРОЕКТИРОВАНИЕ СТРУКТУРЫ ПРОГРАММЫ

Работа с данными:

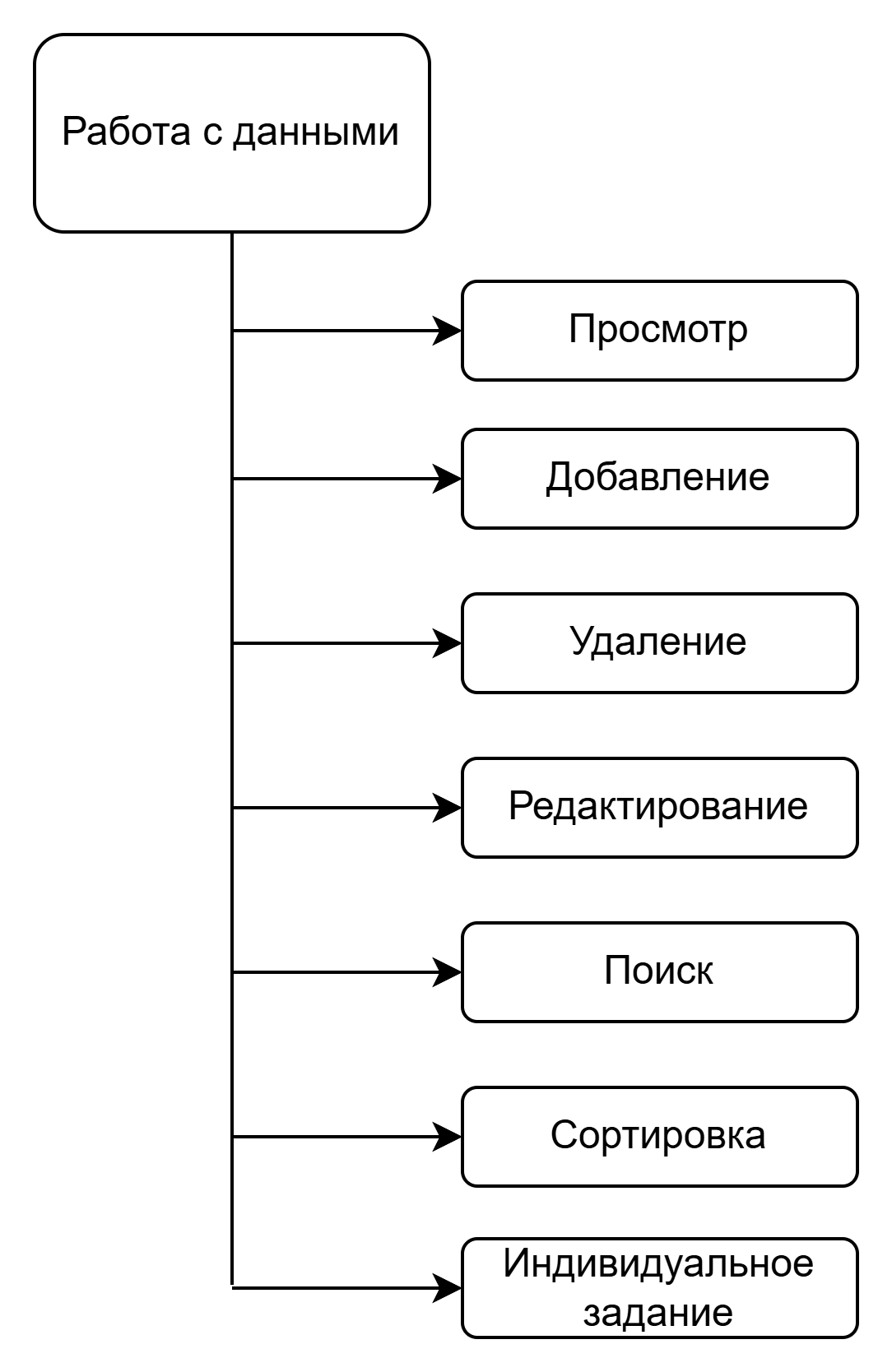
* режим редактирования:
* просмотр всех данных;
* добавление новой записи;
* удаление записи;
* редактирование записи;

режим обработки данных:

* выполнение индивидуального задания;
* поиск данных;
* сортировка.

2.1 Разработка модульной структуры программы

Графическое представление структуры программы с указанием модулей и их функциональных возможностей (рисунок 2.1).



2.2. Выбор способов организации данных:

В качестве выбора способа описания входных данных приводится описание типов struct: для данных с указанием конкретных полей

В качестве способа объединения входных данных указывается:

 использование массивов (статически или динамически создаваемых) или векторов,

 их выбранная область видимости (локальные или глобальные).

2.3 Разработка перечня пользовательских функций программы

Сам перечень:

inline bool isDigit(char& chr);

void inputDataFromFile(struct Trip\* trips, int& tripCount);

time\_t toTimeT(const char\* timeStr);

void inputDataFromFile(struct Trip\* trips, int& tripCount);

void inputTime(char\* inputTime);

void outputBusesBefore12(const struct Trip\* trips, const int& tripCount, const char\* inputTime);

void displayAll(struct Trip\* trips, const int tripCount);

void addTrip(struct Trip\* trips, int& tripCount);

void deleteTrip(struct Trip\* trips, int& tripCount);

void editTrip(struct Trip\* trips, int tripCount);

void searchTripByTripNum(struct Trip\* trips, int tripCount);

void searchTripByBusType(struct Trip\* trips, int tripCount);

void searchTripByDestinationPunct(struct Trip\* trips, int tripCount);

void searchTripByDepartureTime(struct Trip\* trips, int tripCount);

void searchTripByArrivalTime(struct Trip\* trips, int tripCount);

void searchTrip(struct Trip\* trips, int tripCount);

inline bool compareTripsByTripNumber(const struct Trip& a, const struct Trip& b);

inline bool compareTripsByBusType(const struct Trip& a, const struct Trip& b);

inline bool compareTripsByCityName(const struct Trip& a, const struct Trip& b);

inline bool compareTripsByDepartureTime(const struct Trip& a, const struct Trip& b);

inline bool compareTripsByArrivalTime(const struct Trip& a, const struct Trip& b);

void searchTrip(struct Trip\* trips, int tripCount);

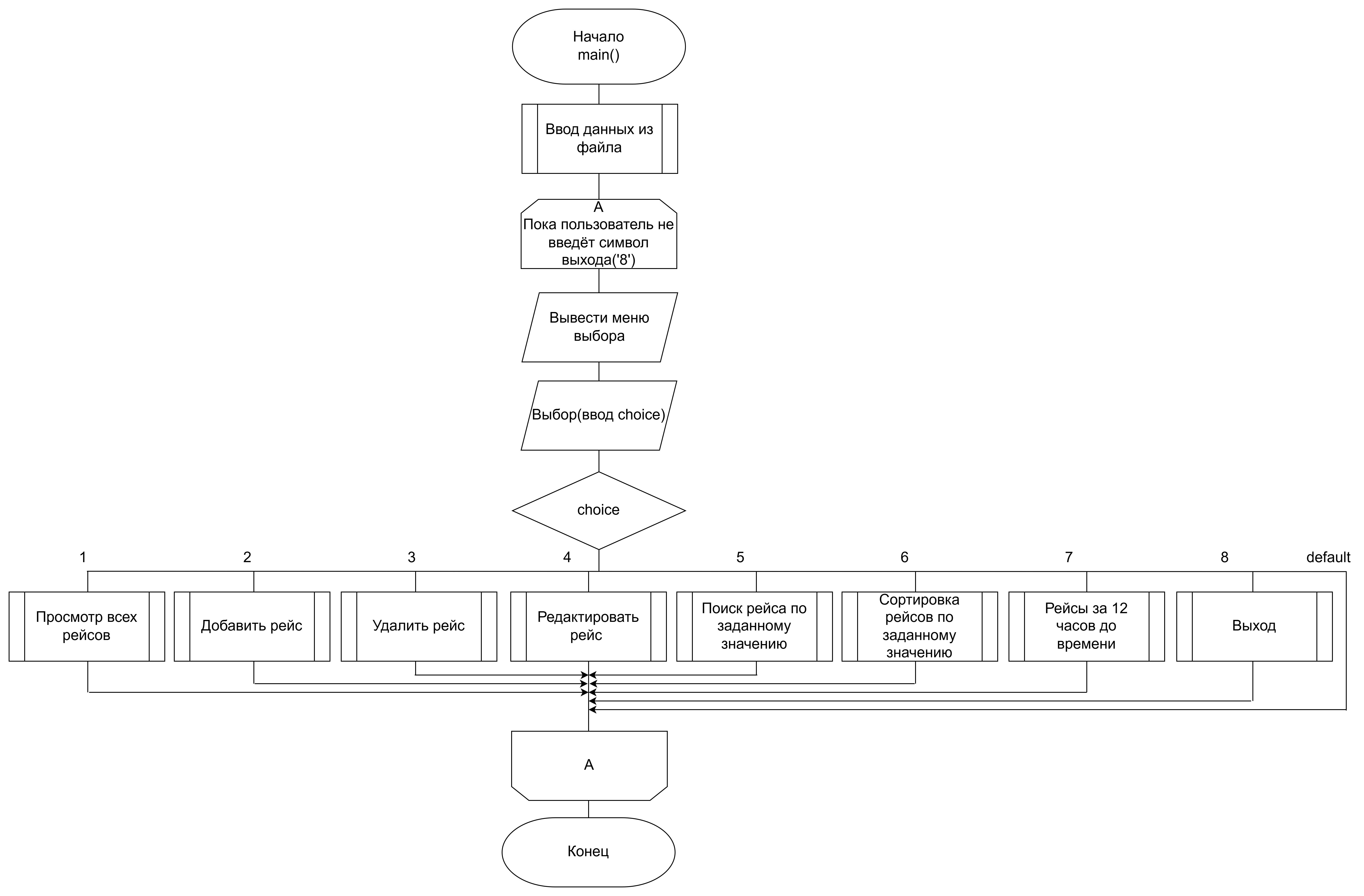
void sort(Trip\* first, Trip\* last, bool (\*comparator)(const struct Trip&, const struct Trip&));

void saveToFile(struct Trip\* trips, int tripCount);

3. РАЗРАБОТКА АЛГОРИТМОВ РАБОТЫ ПРОГРАММЫ

3.1. Алгоритм функции main()

**Блок-схема алгоритма:**



****Описание работы алгоритма:****

**Инициализация данных**:

* + Создаётся массив структур Trip для хранения информации о поездках.
  + Инициализируются переменные: tripCount (количество поездок), time (время), choice (выбор пользователя), error (флаг ошибки).
  + Загружаются данные из файла с помощью функции inputDataFromFile.

**Основной цикл программы**:

* + Выводится меню с доступными опциями.
  + Если флаг error активен, выводится сообщение об ошибке и цикл программы прокручивается вновь.
  + Пользователь вводит свой выбор с помощью функции inputChoice.

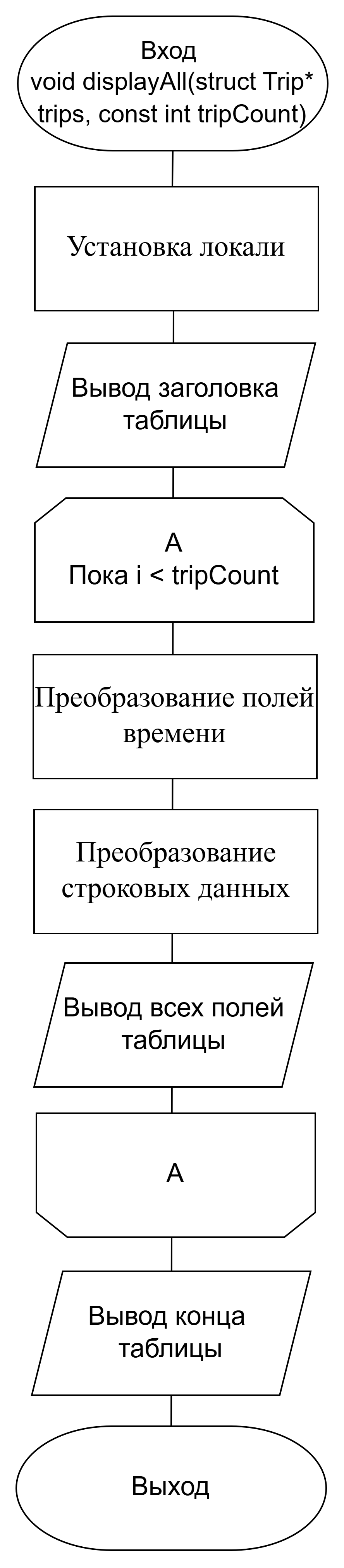
**Обработка выбора пользователя**:

* + В зависимости от выбора (choice), выполняются соответствующие действия:

1. Вывод всех поездок с помощью функции displayAll.
2. Добавление новой поездки с помощью функции addTrip.
3. Удаление поездки с помощью функции deleteTrip.
4. Редактирование поездки с помощью функции editTrip.
5. Поиск поездки с помощью функции searchTrip.
6. Сортировка поездок с помощью функции sortTrips.
7. Ввод времени и вывод автобусов, отправляющихся до 12:00.
8. Сохранение данных в файл и завершение программы.
   * Если выбор некорректен, устанавливается флаг error, тем самым говоря пользователю, что требуется повторить попытку.

3.2. Алгоритм функции displayAll(struct Trip\* trips, const int tripCount)

**Блок-схема алгоритма:**



****Описание работы алгоритма:****

**Настройка локали**:

* + Устанавливается локаль для корректного отображения кириллицы. В случае её использования место прибытия или тип автобуса будут отображаться корректно.

**Вывод заголовка таблицы**:

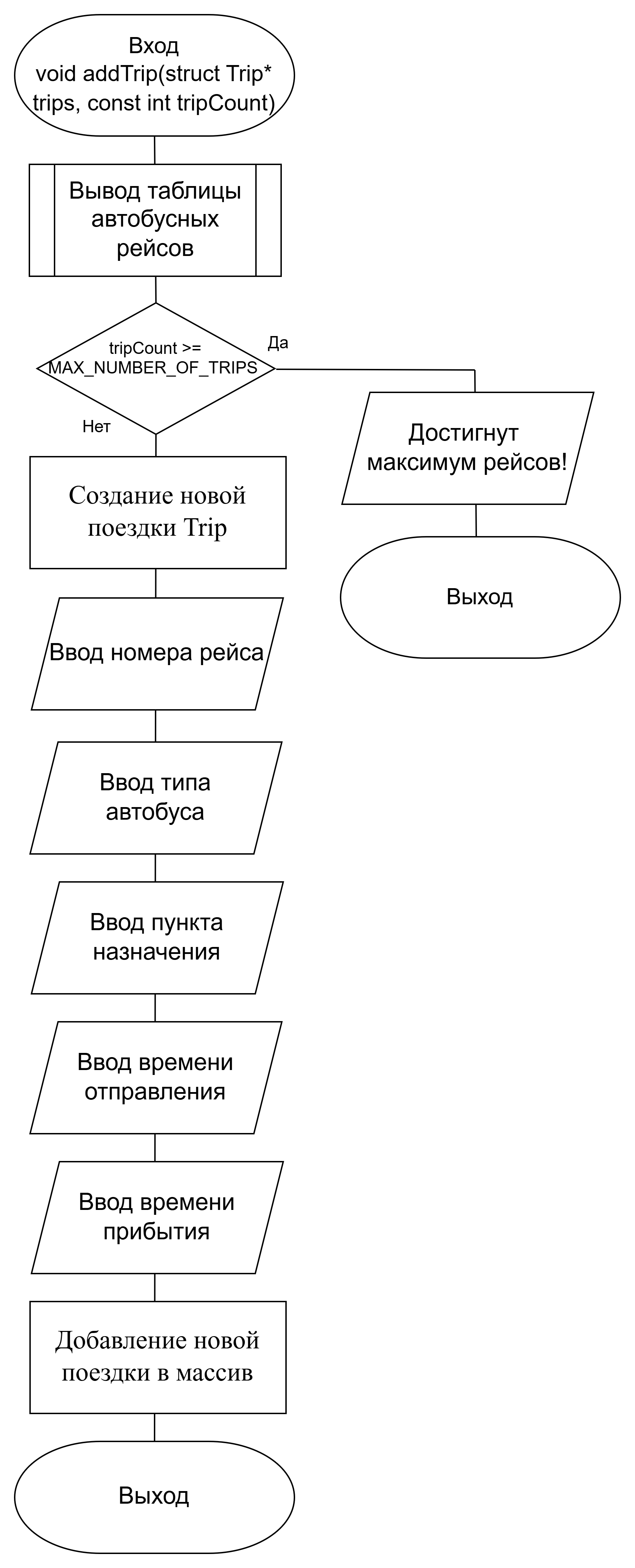
* + Формируется и выводится шапка таблицы с колонками: "Номер рейса", "Тип автобуса", "Место прибытия", "Время отправления", "Время прибытия".

**Вывод данных о поездках**:

* + Для каждой поездки в массиве trips:
    - Преобразуются время отправления и время прибытия time\_t в формат tm.
    - Преобразуются строковые данные (тип автобуса и пункт назначения) в широкие строки для корректного вывода.
    - Данные выводятся в табличном формате.

3.3. Алгоритм функции addTrip(struct Trip\* trips, int& tripCount)

**Блок-схема алгоритма:**



****Описание работы алгоритма:****

**Проверка на переполнение**:

* + Если количество поездок (tripCount) достигло максимального значения, выводится сообщение об ошибке, и функция завершается.

**Создание новой поездки**:

* + Пользователю предлагается ввести данные для новой поездки:
    - Номер рейса (целое число).
    - Тип автобуса (строка, не более 12 символов).
    - Пункт назначения (строка, не более 12 символов).
    - Время отправления и время прибытия (ввод в формате строки и преобразование в тип time\_t).

**Добавление поездки в массив**:

* + Новая поездка добавляется в массив trips, и счётчик tripCount увеличивается на 1.

**3.3. Этап. Разработка программных текстов**

**Основные требования**

* Код необходимо комментировать (как минимум в части объявления структур, массивов/векторов, прототипов функций, нетривиальной логики).
* Код не должен дублироваться – для этого существуют функции!
* Одна функция решает только одну задачу (например, не допускается в одной функции считывать данные из файла и выводить их на консоль – это две разные функции). При этом внутри функции возможен вызов других функций.
* Выполнение операций чтения/записи в файл должно быть сведено к минимуму (т.е. после однократной выгрузки данных из файла в массив/вектор дальнейшая работа ведется с этим массивом/вектором, а не происходит многократное считывание данных из файла в каждой функции).
* Следует избегать глубокой вложенности условных и циклических конструкций: вложенность блоков должна быть не более трѐх.
* Следует избегать длинных функций: текст функции должен умещаться на один экран (в эквиваленте строк – не превышать 25-50 строчек).
* Следует выносить код логически независимых модулей в отдельные .cpp файлы и подключать их с помощью заголовочных .h файлов.

**3.4. Этап. Отладка программного кода.**

Провести несколько тестов разработанной программы. Привести исходные данные и результаты тестирования.

* Обработку исключительных ситуаций:
  + - введенные пользователем данные не соответствуют формату поля (например, символы в числовом поле);
    - введенные пользователем данные нелогичны (например, отрицательное время);
    - файл с данными для чтения не существует;
    - номер удаляемой записи выходит за пределы массива/вектора.

**3.5. Описание работы программы**

подразумевает краткое словесное описание работы программы со скриншотами консоли.