

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1. ПОСТАНОВКА ЗАДАЧИ	4
2. РАЗРАБОТКА АЛГОРИТМОВ.....	7
3. РАЗРАБОТКА ПРОГРАММЫ.....	10
3.1. Выбор средств программирования.....	10
3.2. Разработка модулей.....	11
4. ТЕСТИРОВАНИЕ.....	21
4.1. Описание входных и выходных данных.....	21
4.2. Результаты тестирования.....	21
ЗАКЛЮЧЕНИЕ.....	31
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	32
ПРИЛОЖЕНИЕ А ТЕКСТ ПРОГРАММЫ	
ПРИЛОЖЕНИЕ Б ГРАФИЧЕСКИЙ МАТЕРИАЛ	

					КР.АС59.200054				
Изм	Лист	докум №	Подп.	Дата					
Разраб.		Качан Д.С.			База данных «Вокзал».	Лит	Лист	Листов	
Проверил		Аверина И.Н.				К	2	32	
						БрГТУ			
Н. контр.									
Утв.									

ВВЕДЕНИЕ

База данных «Вокзал» предназначена для ведения учета информации о поездах, которые отправляются с вокзала. Автоматизация хранения и обработки такого рода информации является актуальным и востребованным, так как позволяет регулярно мониторить состояние учитываемых объектов, оперативно получать необходимую информацию, своевременно отвечать на поставленные запросы, безошибочно и корректно формировать требуемые отчеты.

Ведение базы данных подразумевает стандартный набор действий над записями, а именно: добавление новых записей, редактирование записи, просмотр и сортировка всех записей, выборка информации по заданным критериям, статистическая обработка информации и многое другое.

Цель курсовой работы – систематизация, развитие и применение теоретических и практических знаний и умений по алгоритмизации и программированию на примере разработки программного комплекса для автоматизированной обработки массива записей базы данных «Вокзал», представленной в отдельном текстовом файле.

Для достижения поставленной цели в курсовой работе необходимо выполнить решение следующих задач, описанных в следующем пункте.

1. ПОСТАНОВКА ЗАДАЧИ

Для начала определим предметную область, чтобы знать, какие цели и задачи необходимо поставить. Предметная область, подлежащая изучению - «вокзал». В сферу этой предметной области попадают поезда, информация об их времени отправления с вокзала, время в пути, количество остановок, на которых они останавливаются.

Необходимо разработать приложение, которое будет обрабатывать массив структурированных данных «Вокзал» и которое будет, в том числе, вести учёт по самим поездом. Бинарный файл исходных данных должен содержать следующие поля: номер поезда, название конечной станции, дни следования поезда, время отправления, время в пути, количество остановок. Стоит отметить также, что бинарный файл будет содержать информацию о размерности массива, т.е. количество записей в файле.

Приложение необходимо разработать в консольном виде, а также создать консольное меню. Как отмечалось ранее, данные должны быть организованы с помощью структур, а записи должны храниться в отдельном бинарном файле.

В программе должно быть реализовано:

- дополнительное поле с рассчитываемым значением – «Время в пути в сутках»;
- ввод информации из бинарного файла в массив указателей на записи;
- добавление новых элементов в структуру в конец массива и после выбранной записи;
- просмотр всех элементов массива, вывод информации из массива в файл;
- изменение записи, удаление информации по любому полю структуры;
- сортировка по всем полям структуры с созданием индексных бинарных файлов;
- поиск информации в массиве структур по любому полю;
- вычисление количества записей по задаваемому интервалу.
- структуры, способ хранения записей – бинарный файл.

Алгоритмы функциональных процедур проектируемого приложения представлены в виде блок-схем, выполненных в соответствии с ГОСТ ЕСПД 19.701-90. Для создания программного комплекса был использован редактор исходного кода Visual Studio Code. Код скомпилирован и собран с помощью компилятора GCC. Код организован с помощью языка программирования C++. Стандарт языка C++20.

Автоматизированная информационная система (АИС) - это система, в которой информационный процесс управления автоматизирован за счет применения специальных методов обработки данных, использующих комплекс вычислительных, коммуникационных и других технических средств, в целях получения и доставки

результатной информации пользователю-специалисту для выполнения возложенных на него функций управления.

Решения в системе управления принимаются людьми на основе информации, являющейся продуктом ИС. На ее входе находится первичная информация обо всех изменениях, происходящих в объекте управления. Она фиксируется в результате выполнения функций оперативного учета. В ИС первичная информация преобразуется в результатную, пригодную для принятия решений. В автоматизированных ИС часть процедур формального преобразования первичной информации в результатную автоматически выполняются техническими средствами по заранее заданным алгоритмам, без непосредственного вмешательства человека.

Это не означает, что ИС может полностью функционировать в автоматическом режиме. Персонал системы управления определяет состав и структуру первичной и результатной информации, порядок сбора и регистрации первичной информации, контролирует ее полноту и достоверность, определяет порядок выполнения преобразований первичной информации в результатную, контролирует ход выполнения процесса преобразований. К тому же до сих пор слабо автоматизирована процедура сбора первичной информации, поэтому ее ввод в технические средства также осуществляется персоналом ИС.

Важнейшей частью технических средств преобразования информации являются компьютеры, осуществляющие автоматический процесс обработки данных на основе заранее заданных программ. В современных АИС процедуры информационного процесса децентрализованы и выполняются в диалоговом режиме работы пользователя с компьютером, что позволяет ему контролировать процесс преобразования данных, оперативно направляя его в нужное ему русло. Этим они отличаются от АИС, базирующихся на больших ЭВМ, в которых процесс обработки информации выполнялся централизованно и был отделен от управленческого персонала. Последний получал лишь конечные результаты обработки данных и, если они его по тем или иным причинам (например, вследствие поздно выявленных ошибок в исходных данных) не устраивали, вынужден был делать запрос соответствующим службам на повторение процесса решения интересующей его задачи.

Таким образом, в современных АИС автоматически выполняемые процедуры информационного процесса интегрированы с функциями управления. Наряду со своими основными функциями, их непосредственно выполняет управленческий персонал. Более того, используя инструментальные программные средства, ориентированные на пользователя, не имеющего профессиональной компьютерной подготовки, специалист-управленец часто сам может автоматизировать выполнение необходимых ему процедур обработки данных, выступая и в роли постановщика задачи и программиста.

Отметим, что в современном понятии термин «информационные системы» подразумевает автоматизацию информационных процессов. Поэтому оба термина используются как равноправные. Но следует помнить о том, что информационные системы могут использовать и неавтоматизированную технологию обработки информации.

Одно из важнейших мест в информационных системах предприятий занимает функция учёта. Для выполнения в полном объеме функций в управлении предприятием и для составления отчетности, предоставляемой внешним пользователям, необходимо осуществлять сбор, регистрацию, передачу, накопление, хранение и обработку учетных данных. Для реализации этого информационного процесса требуются соответствующие формы организации работы, технические средства, методы и способы преобразования данных, а также персонал определенной квалификации.

На сегодняшний день страны очень тесно связаны друг с другом в информационном смысле этого слова. Существует множество различных баз данных между странами, причём совершенно в любой отрасли. Поэтому и эта база данных, которая описывается в этом курсовом проекте, ориентирована на использование единой базы данных, являющейся совокупностью структурированных данных, предназначенных для многоцелевого и многократного их применения. А также методов доступа к ним с целью уменьшения степени трудоёмкости, увеличения производительности, повышения точности и надёжности выполняемых операций и задач. Поэтому базу данных «вокзал» целесообразно автоматизировать. Заметим, что данная база данных носит глобальный характер, преимущественно о поездах, отправляющихся с вокзала. Под этим понимаются их характеристики, такие как номер поезда, название конечной станции, дни следования, время отправления, время в пути, количество остановок.

Автоматизированная система позволит приводить в порядок информацию по базам данных стран мира и их состояния, возможность обновления в связи с изменением некоторых экономических, политических, топографических и других причин. Изменения всё-таки не так редки, как может показаться на первый взгляд, поэтому можно сказать, что это актуально. А также использование баз данных — возможность надёжного хранения информации.

2. РАЗРАБОТКА АЛГОРИТМОВ

Этот этап один из самых важных. После того, как мы осуществили постановку задачи, для дальнейшей работы нам будет необходимо разработать алгоритмы. Разработать алгоритм - это означает, что на основе выбранного метода записывается последовательность действий, приводящих к решению задачи. Успешная разработка алгоритма позволяет избежать многих ошибок, поскольку именно на этом этапе определяется логика будущей программы. Для решения поставленной задачи потребуется создать простой и удобный алгоритм взаимодействия пользователя с базой данных «Вокзал». Для этого в основном модуле программы нужно реализовать меню. Алгоритм работы программы будет заключаться в следующем: в начале работы будет требоваться ввод имени файла, с которым будет проводиться дальнейшая работа программы, если файла с таким именем нет, то необходимо его создать и работать с ним как с пустым.

Алгоритм ввода информации из файла в массив записей:

- 1 Запрашиваем имя файла
- 2 Открываем файл для чтения.
- 3 Если файл не найден, то спросим пользователя о том, нужно ли его создать.
- 4 Считываем количество записей из файла, расширяем динамический массив на полученное значение количества записей.
- 5 В случае, если файл пустой:
 - 5.1 Чтение из файла прекращается
- 6 В случае, если файл не пустой:
 - 6.1 Читаем записи из файла пока процесс считывания данных не достигнет конца файла.
 - 6.2 Записываем прочитанные строки в массив данных
- 7 Закрываем файл.
- 8 Изменяем содержание индексных бинарных файлов.

Также сразу опишем следующие алгоритмы.

Алгоритм вывода информации в текстовый и бинарный файл:

- 1 Открываем два файла для записи (текстовый и бинарный).
- 2 Перезаписываем весь файл целиком, записывая записи из массива в файл бинарный, записываем шапку и записи в текстовом виде в текстовый.
- 3 Закрываем оба файла.

Стоит отметить, что рабочим файлом считается бинарный файл.

Алгоритм вывода информации в индексные бинарные файлы:

- 1 Открываем файлы для записи.
- 2 Сортируем записи по всем полям.
- 3 Записываем индексы в отсортированном порядке в файлы.
- 4 Закрываем файлы.

После успешного прохождения всех этих действий осуществляется вызов меню, с помощью которого пользователь взаимодействует со всеми остальными возможностями программы, алгоритмы которых, приведены ниже.

Алгоритм добавления новых элементов:

- 1 Запрашиваем индекс записи, после которой необходимо добавить запись.
- 2 Выделяем память для нового элемента структуры с увеличением размерности для добавления новой записи. Если запись вставляется в середину массива, то остальные элементы сдвигаются на 1 вперед.
- 3 Запрашивается ввод полей записи.
- 4 Непосредственный ввод полей записи проверкой ввода.
- 5 Изменяем содержание бинарного и текстового файлов по алгоритму ввода информации в текстовый и бинарный файл.
- 6 Изменяем содержание индексных бинарных файлов соответственно.

Алгоритм вывода элементов массива на консоль:

- 1 Вывод шапки.
- 2 Запускаем цикл для всех элементов массива.
- 3 Производим вывод полей структуры на экран в форматированном виде.

Алгоритм изменения полей записей в массиве:

- 1 Выбираем индекс элемента, для изменения полей.
- 2 Выбираем поле для изменения
- 3 Если не выбрана отмена изменения:
 - 3.1 На основании выбора вносим новое значение выбранного поля.
 - 3.2 Возвращаемся к пункту 2.
- 4 Изменяем содержание бинарного и текстового файлов по алгоритму ввода информации в текстовый и бинарный файл.
- 5 Изменяем содержание индексных бинарных файлов соответственно

Алгоритм поиска записи по значению ключевого поля:

- 1 Выбираем поле для поиска.

- 2 Вводим значение поля (или границы значений поля)
- 3 Запускаем цикл для всех элементов массива.
 - 3.1 Выводим записи с заданным значением (или промежутком) на экран.

Алгоритм удаления выбранного элемента:

- 1 Выбираем поле для удаления.
- 2 Вводим значение поля.
- 3 Запускаем цикл для всех элементов массива.
 - 3.1 Удаляем запись с введенным значением поля.
 - 3.2 Уменьшаем размерность массива и сдвигаем записи назад.
- 4 Изменяем содержание бинарного и текстового файла по алгоритму ввода информации в текстовый или бинарный файл.
- 5 Изменяем содержание индексных бинарных файлов соответственно.

Алгоритм сортировки записей по выбранному полю:

- 1 Выбор поля для сортировки.
- 2 Открываем соответствующий индексный бинарный файл.
- 3 Считываем информацию из индексного бинарного файлов.
- 4 Закрываем соответствующий индексный бинарный файл.
- 5 В соответствии со считанными индексами, выводим на экран отсортированные записи.

Алгоритм отмены последнего действия:

- 1 Если количество элементов буфера больше 0:
 - 1.1 Считываем из буфера количество записей.
 - 1.2 Начинаем цикл по записям из буфера:
 - 1.2.1 Копируем запись из буфера в массив.
 - 1.3 Удаляем элемент буфера.

3. РАЗРАБОТКА ПРОГРАММЫ

3.1. Выбор средств программирования

Для разработки программы была выбран редактор исходного кода Visual Studio Code, так как он является одним из самых современных, удобных и регулярно обновляемых редакторов исходного кода, имеет в своем распоряжении большое количество инструментов для разработки программного обеспечения. Помимо стандартного редактора, Visual Studio Code включает в себя интеграцию с системой контроля версий Git, средства подсказок для написания кода и многие другие функции для упрощения процесса разработки. Для компиляции и сборки готовой программы используется компилятор для различных языков программирования GCC. Так как он использует последний стандарт языка C++ и имеет возможность компилировать различные языки программирования. Операционная система, установленная на ПК, Linux Manjaro KDE Plasma 5.21.5, регулярно обновляется и поддерживается разработчиками, что также является большим плюсом для создания и тестирования программного обеспечения. Язык программирования – C++, стандарт языка – C++20.

Используемые библиотеки: `iostream` - заголовочный файл с классами, функциями и переменными для организации ввода-вывода в языке программирования C++. `fstream` - заголовочный файл из стандартной библиотеки C++, включающий набор классов, методов и функций, которые предоставляют интерфейс для чтения/записи данных из/в файл. `iomanip` - реализует инструменты для работы с форматированием вывода. `cstring` — содержит функции для работы с с-строками, представленные массивом символов, для копирования, вычисления длины строки, перевода из строки в число. `span` — реализует функции для работы с генерируемыми последовательностями. Используются в циклах `for`.

Сейчас опишем разные ключевые переменные, которые встречаются в программе. Ключевыми переменными являются массив `train_station[]` типа `Train` и его размерность `train_count` типа `int`. Крайне важен массив указателей на функции `menu_func_array`. Было решено организовать массив указателей на функции по той причине, что он крайне удобен для того, чтобы программу разбивать на модули и там определять основные функции. Так же важную роль для работы функции отмены является массив `trains_buffer[]` типа `TrainBuffer` и размерностью `train_buffer_count` типа `int`. Переменная `is_working` типа `bool` нужна для работы главного цикла программы и выхода из него. Переменная `action` типа `int` требуется для выбора нужного действия из главного меню: вывод записей, добавление, удаление, редактирование, сортировка, выборка записей, отмена последнего действия. Важной переменной для работы программы является имя файла `file_name` типа `char*`, требующаяся для открытия нужного бинарного файла с записями.

Теперь опишем некоторые частные переменные. Переменная `type` типа `int`, реализующая выбор поля для удаления, редактирования, сортировки. Массив `buffer[]` типа `Train` для копирования данных из основного массива. Используется для увеличения/уменьшения размера массива, сортировок (там такой массив называется `sort_buffer`). Массив индексов `index[]` типа `int` для хранения индексов записей. Используется в сортировках для последующей записи в индексные бинарные файлы. Переменные флагов `reverse`, `in_file` типа `bool` нужны для сортировок. Используются для обратной сортировки и записи в файл соответственно. Переменная `is_false` типа `bool` для проверки ввода практически во всех функциях. Используется наряду с проверкой через `cin.fail()` для проверки вводимого пользователем значения поля, переменной и др. Переменные `hours_str[]`, `munutes_str[]` типа `char` и `time_in_hours` типа `double` используются в функциях для того, чтобы переводить время, записанное в формате `XX:XX`, на время в часах. Используется в сортировках, удалении, выборке записей. Позиция `pos` типа `int` нужна для ввода позиции для добавления записи в массив. Переменные `is_stock` и `is_equal` типа `bool` используются как флаги для поиска нужного значения поля в массиве. Так же в программе использовались временные массивы для полей записей, требующиеся для их сортировки.

Ввод ответных переменных осуществляется через `cin`. Это обеспечивает удобное взаимодействие с программой. Плюсы: можно отследить некорректный ввод, можно изменить ввод до его подтверждения.

Для форматированного вывода использовалась функция `setw()`, позволяющая задавать ширину выводимых данных. Также была использована функция `stoi()` для преобразования строки в целое число.

Для сортировки использовался метод быстрой сортировки (`quick sort`). Плюс: работает быстро даже с большим количеством данных.

3.2. Разработка модулей

Программа разбита на 6 основных модулей: `proga.cpp`, `train_func.h` и `train_func.cpp`, `file.h` и `file.cpp`, `menu.h` и `menu.cpp`, `config_func.h` и `config_func.cpp`, `sort_func.h` и `sort_func.cpp`, `delete_func.h` и `delete_func.cpp`, `select_func.h` и `select_func.cpp`, `clear.h`, `clear.cpp` и `clear_w.cpp` – и библиотеку `train.h`. Это необходимо и рационально, т.к. весь код программы очень большой. Для того, чтобы лучше ориентироваться в коде, быстрее находить и исправлять ошибки, программа была разбита на модули.

Модуль `proga.cpp`

Является основным модулем программы, который содержит функцию `main()`, где содержится главное меню программы. Здесь происходит считывание записей из бинарного файла. После чего запрашивается ввод действия из пункта меню, вводится

соответственно само действие. Далее, происходит проверка вводимого значения. Если были введены некорректные данные, по типу символа, не относящегося к числу или несуществующего пункта меню, то выводится оповещение о некорректном вводе и предлагается ввести пункт заново. После уже корректного ввода управление потоком выполнения программы переходит к соответствующей функции из массива указателей на функции с передачей требуемых параметров. После выполнения работы функции, управление выполнением программы снова возвращается в основной цикл и выводит меню работы с файлом. Такой возврат будет продолжаться до тех пор, пока не будет избран пункт окончания работы программы.

Модуль **lender.h**

Модуль содержит в себе саму структуру данных. Этот модуль необходим и подключается во всех остальных файлах программы для того, чтобы доступ к информации о типе структуры и ее полей получали остальные файлы и соответственно могли работать с массивами таких структур.

Модуль **train_func.h** и **rain_func.cpp**

Содержит в себе реализацию всех функций, который есть в главном меню программы. Это функции вывода, добавления, редактирования, удаления, сортировки, выборки записей и отмены действий.

1 Функция **void print_train(Train*& trains, TrainBuffer* trains_buffer, int& count, int& buffer_count)**

Входные параметры: **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве.

TrainBuffer* trains_buffer - указатель на массив буфера для хранения действий, **int& buffer_count** - ссылка на количество элементов в буфере.

Назначение: Вывод записей в консоль в форматированном виде.

Возвращаемое значение: Значений не возвращает.

2 Функция **void add_train(Train*& trains, TrainBuffer* trains_buffer, int& count, int& buffer_count)**

Входные параметры: **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве.

TrainBuffer* trains_buffer - указатель на массив буфера для хранения действий, **int& buffer_count** - ссылка на количество элементов в буфере.

Назначение: Добавление новой записи после указанной.

Возвращаемое значение: Значений не возвращает.

3 **Функция void change_train(Train*& trains, TrainBuffer* trains_buffer, int& count, int& buffer_count)**

Входные параметры: **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве.

TrainBuffer* trains_buffer - указатель на массив буфера для хранения действий, **int& buffer_count** - ссылка на количество элементов в буфере.

Назначение: Редактирование полей записи по номеру.

Возвращаемое значение: Значений не возвращает.

4 **Функция void delete_train(Train*& trains, TrainBuffer* trains_buffer, int& count, int& buffer_count)**

Входные параметры: **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве.

TrainBuffer* trains_buffer - указатель на массив буфера для хранения действий, **int& buffer_count** - ссылка на количество элементов в буфере.

Назначение: Удаление записей по заданному значению поля.

Возвращаемое значение: Значений не возвращает.

5 **Функция void sort_train(Train*& trains, TrainBuffer* trains_buffer, int& count, int& buffer_count)**

Входные параметры: **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве.

TrainBuffer* trains_buffer - указатель на массив буфера для хранения действий, **int& buffer_count** - ссылка на количество элементов в буфере.

Назначение: Сортировка записей по выбранному полю.

Возвращаемое значение: Значений не возвращает.

6 **Функция void select_train(Train*& trains, TrainBuffer* trains_buffer, int& count, int& buffer_count)**

Входные параметры: **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве.

TrainBuffer* trains_buffer - указатель на массив буфера для хранения действий, **int& buffer_count** - ссылка на количество элементов в буфере.

Назначение: Выборка записей по заданному значению (диапазону значений) поля.

Возвращаемое значение: Значений не возвращает.

7 **Функция void undo_train(Train*& trains, TrainBuffer* trains_buffer, int& count, int& buffer_count)**

Входные параметры: **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве.

TrainBuffer* trains_buffer - указатель на массив буфера для хранения действий, **int& buffer_count** - ссылка на количество элементов в буфере.

Назначение: Отмена последнего действия.

Возвращаемое значение: Значений не возвращает.

Модуль file.h и file.cpp

Реализация функций для работы с файлами. Описание функций представлено ниже.

1 **Функция int get_records_from_file(char file_name[], Train*& trains, int& count)**

Входные параметры: **char file_name[]** - имя файла для чтения записей, **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве.

Назначение: Получение записей из файла.

Возвращаемое значение: Целое число, означающее успешное или неудачное открытие файла.

2 **Функция int put_records_in_file(char file_name[], Train* trains, int count)**

Входные параметры: **char file_name[]** - имя файла для записи данных, **Train* trains** - указатель на массив структур (записей), **int count** - количество записей в массиве.

Назначение: Запись данных в бинарный файл и форматированный вывод полей записей в текстовый файл.

Возвращаемое значение: Целое число, означающее успешное или неудачное открытие файла.

3 Функция **void create_index_file(Train* trains, int count)**

Входные параметры: **Train* trains** - указатель на массив структур (записей), **int count** - количество записей в массиве.

Назначение: Сортировка записей и создание индексных бинарных файлов, содержащих индексы.

Возвращаемое значение: Значений не возвращает.

Модуль **menu.h** и **menu.cpp**

Функции для вывода различных меню для выбора варианта. Описание функций представлено ниже.

1 Функция **void print_menu()**

Входные параметры: Входных параметров нет.

Назначение: Вывод главного меню в консоль.

Возвращаемое значение: Значений не возвращает.

2 Функция **void print_sort_menu()**

Входные параметры: Входных параметров нет.

Назначение: Вывод меню вариантов сортировки в консоль.

Возвращаемое значение: Значений не возвращает.

3 Функция **void print_reverse_menu()**

Входные параметры: Входных параметров нет.

Назначение: Вывод меню выбора хода сортировки в консоль.

Возвращаемое значение: Значений не возвращает.

4 Функция **void print_file_menu()**

Входные параметры: Входных параметров нет.

Назначение: Вывод меню выбора записи в файл в консоль.

Возвращаемое значение: Значений не возвращает.

5 Функция **void print_delete_menu()**

Входные параметры: Входных параметров нет.

Назначение: Вывод меню вариантов удаления в консоль.

Возвращаемое значение: Значений не возвращает.

6 Функция void print_selection_menu()

Входные параметры: Входных параметров нет.

Назначение: Вывод меню вариантов выборки в консоль.

Возвращаемое значение: Значений не возвращает.

7 Функция void print_change_menu()

Входные параметры: Входных параметров нет.

Назначение: Вывод меню вариантов изменения в консоль.

Возвращаемое значение: Значений не возвращает.

Модуль config_func.h и config_func.cpp

Функции для работы с памятью. Описание функций представлено ниже.

1 Функция void add_memory_train(Train*& trains , int& count)

Входные параметры: **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве.

Назначение: Добавление дополнительной памяти в динамический массив структур.

Возвращаемое значение: Значений не возвращает.

2 Функция void sub_memory_train(Train*& trains , int& count)

Входные параметры: **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве.

Назначение: Удаление ячейки памяти из динамического массива структур.

Возвращаемое значение: Значений не возвращает.

3 Функция void save_trains_in_buffer(Train* trains , TrainBuffer* trains_buffer, int count, int& buffer_count)

Входные параметры: **Train* trains** - указатель на массив структур (записей), **int count** - количество записей в массиве. **TrainBuffer*& trains** - ссылка на указатель на массив буфера отмены, **int& buffer_count** - ссылка на количество записей в буфере.

Назначение: Сохранение текущего состояния в буфер для последующей возможности отменить действие.

Возвращаемое значение: Значений не возвращает.

Модуль `sort_func.h` и `sort_func.cpp`

Функции сортировки массива структур. Описание функций представлено ниже.

1 Функция `void number_sort(Train* trains , int* index, int count)`

Входные параметры: **Train* trains** - указатель на массив структур (записей), **int count** - количество записей в массиве, **int* index** — указатель на массив индексов.

Назначение: Сортировка по номеру поезда.

Возвращаемое значение: значений не возвращает.

2 Функция `void end_station_sort(Train* trains , int* index, int count)`

Входные параметры: **Train* trains** - указатель на массив структур (записей), **int count** - количество записей в массиве, **int* index** — указатель на массив индексов.

Назначение: Сортировка по названию конечной станции.

Возвращаемое значение: значений не возвращает.

3 Функция `void departure_time_sort(Train* trains , int* index, int count)`

Входные параметры: **Train* trains** - указатель на массив структур (записей), **int count** - количество записей в массиве, **int* index** — указатель на массив индексов.

Назначение: Сортировка по времени отправления поезда.

Возвращаемое значение: значений не возвращает.

4 Функция `void way_time_sort(Train* trains , int* index, int count)`

Входные параметры: **Train* trains** - указатель на массив структур (записей), **int count** - количество записей в массиве, **int* index** — указатель на массив индексов.

Назначение: Сортировка по времени в пути поезда.

Возвращаемое значение: значений не возвращает.

5 Функция `void stop_count_sort(Train* trains , int* index, int count)`

Входные параметры: **Train* trains** - указатель на массив структур (записей), **int count** - количество записей в массиве, **int* index** — указатель на массив индексов.

Назначение: Сортировка по количеству остановок.

Возвращаемое значение: значений не возвращает.

- 6 Перегруженная функция **void quick_sort(int* A , int from, int to, int* index)**
void quick_sort(double* A , int from, int to, int* index)
void quick_sort(char A[][256] , int from, int to, int* index)

Входные параметры: **int* A (double* A, char A[][256])** - указатель на массив данных для сортировки, **int from** — индекс начала сортировки, **int to** — индекс конца сортировки, **int* index** — указатель на массив индексов.

Назначение: Сортировка данных и массива индексов.

Возвращаемое значение: значений не возвращает.

Модуль **delete_func.h** и **delete_func.cpp**

Функции удаления записей. Описание функций представлено ниже.

- 1 Функция **void delete_train_by_number(Train*& trains, int& count, char number[5])**

Входные параметры: **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве, **char number[5]** — номер поезда для удаления.

Назначение: Удаление записи по номеру.

Возвращаемое значение: значений не возвращает.

- 2 Функция **void delete_train_by_end_station(Train*& trains, int& count, char end_station[256])**

Входные параметры: **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве, **char end_station[256]** — название станции для удаления.

Назначение: Удаление записей по названию станции.

Возвращаемое значение: значений не возвращает.

- 3 Функция **void delete_train_by_depatrure_time(Train*& trains, int& count, char departure_time[7])**

Входные параметры: **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве, **char departure_time[7]** — время отправления для удаления.

Назначение: Удаление записей по времени отправления.

Возвращаемое значение: значений не возвращает.

4 Функция void delete_train_by_way_time(Train*& trains, int& count, i way_time[7])

Входные параметры: **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве, **char way_time[5]** — время в пути для удаления.

Назначение: Удаление записей по времени в пути.

Возвращаемое значение: значений не возвращает.

5 Функция void delete_train_by_stop_count(Train*& trains, int& count, int stop_count)

Входные параметры: **Train*& trains** - ссылка на указатель на массив структур (записей), **int& count** - ссылка на количество записей в массиве, **int stop_count** — количество остановок для удаления.

Назначение: Удаление записей по количеству остановок.

Возвращаемое значение: значений не возвращает.

Модуль select_func.h и select_func.cpp

Функции выюорки записей. Описание функций представлено ниже.

1. Функция void select_by_number(Train* trains, TrainBuffer* trains_buffer1, int count, int buffer_count, int down_number, int up_number)

Входные параметры: **Train*& trains** - суказатель на массив структур (записей), **int count** - количество записей в массиве, **TrainBuffer* trains_buffer1** — указатель на буфер отмены, **int buffer_count** – количество элементов в буфере, **int down_number** – нижняя граница номера поезда, **int up_number** – верхняя граница номера поезда.

Назначение: Выборка записей по номерам, входящим в промежуток.

Возвращаемое значение: значений не возвращает.

2 Функция void select_by_end_station(Train* trains, TrainBuffer* trains_buffer1, int count, int buffer_count, char end_station[256])

Входные параметры: **Train*& trains** - суказатель на массив структур (записей), **int count** - количество записей в массиве, **TrainBuffer* trains_buffer1** — указатель на буфер отмены, **int buffer_count** – количество элементов в буфере, **char end_station[256]** – название станции.

Назначение: Выборка записей по названию станции, начинающихся на заданное.

Возвращаемое значение: значений не возвращает.

3 Функция void select_by_departure_time(Train* trains, TrainBuffer* trains_buffer1, int count, int buffer_count, char down_departure_time[7], char up_departure_time[7])

Входные параметры: **Train*& trains** - суказатель на массив структур (записей), **int count** - количество записей в массиве, **TrainBuffer* trains_buffer1** — указатель на буфер отмены, **int buffer_count** – количество элементов в буфере, **char down_departure_time[7]** – нижняя граница времени отправления, **char up_departure_time[7]** – верхняя граница времени отправления.

Назначение: Выборка записей по времени отправления, входящим в промежуток.

Возвращаемое значение: значений не возвращает.

4 Функция void select_by_way_time(Train* trains, TrainBuffer* trains_buffer1, int count, int buffer_count, char down_way_time[7], char up_way_time[7])

Входные параметры: **Train*& trains** - суказатель на массив структур (записей), **int count** - количество записей в массиве, **TrainBuffer* trains_buffer1** — указатель на буфер отмены, **int buffer_count** – количество элементов в буфере, **char down_way_time[7]** – нижняя граница времени в пути, **char up_way_time[7]** – верхняя граница времени в пути.

Назначение: Выборка записей по времени в пути, входящим в промежуток.

Возвращаемое значение: значений не возвращает.

5 Функция void select_by_stop_count(Train* trains, TrainBuffer* trains_buffer1, int count, int buffer_count, int down_stop_count, int up_stop_count)

Входные параметры: **Train*& trains** - суказатель на массив структур (записей), **int count** - количество записей в массиве, **TrainBuffer* trains_buffer1** — указатель на буфер отмены, **int buffer_count** – количество элементов в буфере, **int down_stop_count** – нижняя граница количества остановок, **int up_stop_count** – верхняя граница количества остановок.

Назначение: Выборка записей по количеству остановок, входящих в промежуток.

Возвращаемое значение: значений не возвращает.

4. ТЕСТИРОВАНИЕ

4.1. Описание входных и выходных данных

При запуске программы выполняется чтение данных из бинарного файла и запись их в массив. Затем на экране появляется меню для работы с массивом структур. В ходе выполнения программы происходит изменение информации, хранимой в массиве и в файле. Немного об ограничениях. Ограничение по длине для названия станции 256 символов, для времени отправления и времени в пути — 7 символов, для номера поезда — 5 символов. Количество остановок не может быть отрицательным. Однако пользователю не нужно будет держать это в голове, т.к. в программе реализована защита от некорректного ввода.

Файл с информацией test.bin (для наглядности используется output.txt, т. к. в нем информация представлена в текстовом виде) (см.рисунок 4.1).



The screenshot shows a text editor window titled 'output_file.txt - Kate'. The editor displays a table with 8 columns: 'Номер' (Number), 'Конечная станция' (Final station), 'Дни следования' (Days of travel), 'Отправление' (Departure), 'Время в пути' (Travel time), 'Кол-во остановок' (Number of stops), 'Время в пути в сутках' (Travel time in days), and 'Время в пути в сутках' (Travel time in days). The table contains 13 rows of data, with the last row being empty. The data is as follows:

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках	Время в пути в сутках
2133	Minsk	Every	6:00	3:00	9	0.125	
4513	Smolensk	Mon Fri Sat	9:00	14:00	12	0.583333	
3815	Moskva	Every	12:45	15:35	19	0.649306	
7618	Grodno	Sat Sun	13:25	6:10	9	0.256944	
6354	Luninets	Every	6:40	2:30	3	0.104167	
3548	Vilno	Mon Sun	13:20	6:45	8	0.28125	
9703	Baranovichi	Mon Fri Sat	18:32	1:30	5	0.0625	
2591	Warszawa	Every	14:18	4:50	7	0.201389	
5924	Praga	Tue Sun	7:40	9:35	13	0.399306	
3951	Mogilev	Mon Tue Fri	14:55	7:30	8	0.3125	
5625	Kobrin	Mon Sat	4:00	0:50	1	0.0347222	
7688	Berlin	Every	5:20	21:40	17	0.902778	

Рисунок 4.1 - Файл с информацией

4.2. Результаты тестирования

При запуске программа автоматически запрашивает имя файла для чтения записей из него (см. рисунок 4.2). В случае, если файл с таким именем нет, то выводится оповещение о том, что файл не найден, и запрос на создание нового файла с таким именем (см.рисунок 4.3).

```
Введите имя файла: data.bin|
```

Рисунок 4.2 – Запрос на ввод имени файла

```
Введите имя файла: data.bin
Файл не открыт!!!
Создать новый файл? (0 - не создавать, 1 - создать): 0|
```

Рисунок 4.3 — Запрос на создание нового файла

```
Введите имя файла: data.bin
Файл не открыт!!!
Создать новый файл? (0 - не создавать, 1 - создать): 0
Введите имя файла: test.bin|
```

Рисунок 4.4 — Ввод действительного имени файла

Введем имя test.bin (см. рисунок 4.4). После открытия и считывания данных с файла выводится меню (см. рисунок 4.5) для непосредственной работы со считанными данными. В меню предлагается выбрать один из 8 пунктов, считая нулевой пункт – выход из программы. В программе осуществляется проверка ввода, поэтому программа устойчива к некорректному вводу (см. рисунок 4.6).

```
=====MENU=====

1      вывод всех поездов
2      добавление нового поезда
3      изменение поезда
4      удаление поезда
5      сортировка поездов
6      выборка записей
7      отмена последнего действия
0      выход из программы

Выберите действие: |
```

Рисунок 4.5 – Основное меню

```

Выберите действие: gfdg
Неправильный ввод!!!
Выберите действие: fy64
Неправильный ввод!!!
Выберите действие: 2642|

```

```

Такого действия нет

=====MENU=====

1      вывод всех поездов
2      добавление нового поезда
3      изменение поезда
4      удаление поезда
5      сортировка поездов
6      выборка записей
7      отмена последнего действия
0      выход из программы

Выберите действие: |

```

Рисунок 4.6 — Проверка ввода

В пункте 1 осуществляется вывод всех записей в консоль (см.рисунок 4.7).

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках
2133	Minsk	Every	6:00	3:00	9	0.125
4513	Smolensk	Mon Fri Sat	9:00	14:00	12	0.583333
3815	Moskva	Every	12:45	15:35	19	0.649306
7618	Grodno	Sat Sun	13:25	6:10	9	0.256944
6354	Luninets	Every	6:40	2:30	3	0.104167
3548	Vilno	Mon Sun	13:20	6:45	8	0.28125
9703	Baranovichi	Mon Fri Sat	18:32	1:30	5	0.0625
2591	Warszawa	Every	14:18	4:50	7	0.201389
5924	Praga	Tue Sun	7:40	9:35	13	0.399306
3951	Mogilev	Mon Tue Fri	14:55	7:30	8	0.3125
5625	Kobrin	Mon Sat	4:00	0:50	1	0.0347222
7688	Berlin	Every	5:20	21:40	17	0.902778

Рисунок 4.7 – Вывод содержания файла

В пункте 2 осуществляется ввод новых данных. При выборе этого пункта запрашивается позиция для добавления (см. рисунок 4.8). Выберем позицию -1, т. е. добавление в конец. Для ввода позиции так же осуществляется проверка ввода. После выбора позиции программа запрашивает ввод полей записи, так же проверяя ввод (см. рисунок 4.9). Результат добавления выведем в консоль (см. рисунок 4.10).

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках
2133	Minsk	Every	6:00	3:00	9	0.125
4513	Smolensk	Mon Fri Sat	9:00	14:00	12	0.583333
3815	Moskva	Every	12:45	15:35	19	0.649306
7618	Grodno	Sat Sun	13:25	6:10	9	0.256944
6354	Luninets	Every	6:40	2:30	3	0.104167
3548	Vilno	Mon Sun	13:20	6:45	8	0.28125
9703	Baranovichi	Mon Fri Sat	18:32	1:30	5	0.0625
2591	Warszawa	Every	14:18	4:50	7	0.201389
5924	Praga	Tue Sun	7:40	9:35	13	0.399306
3951	Mogilev	Mon Tue Fri	14:55	7:30	8	0.3125
5625	Kobrin	Mon Sat	4:00	0:50	1	0.0347222
7688	Berlin	Every	5:20	21:40	17	0.902778

Введите позицию для добавления записи после указанной записи (-1 - в конец): |

Введите позицию для добавления записи после указанной записи (-1 - в конец): -1|

Рисунок 4.8 – Запрос ввода позиции

```

Номер поезда: 12345
Номер поезда должен состоять из менее, чем 5 символов!!!
Номер поезда: 2133
Такой номер уже есть в базе данных!!!
Номер поезда: 7632
Конечная станция: Orsha
Дни следования: Mon Sat
Время отправления: 24:00
Неправильно задано время!!!
Время отправления: 12:65
Неправильно задано время!!!
Время отправления: 7:30
Время в пути: 1:70
Неправильно задано время!!!
Время в пути: 6:42
Количество остановок: -7
Количество остановок должно быть больше нуля!!!
Количество остановок: 0
Количество остановок должно быть больше нуля!!!
Количество остановок: 5

```

Рисунок 4.9 — Ввод полей записи для добавления

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках
2133	Minsk	Every	6:00	3:00	9	0.125
4513	Smolensk	Mon Fri Sat	9:00	14:00	12	0.583333
3815	Moskva	Every	12:45	15:35	19	0.649306
7618	Grodno	Sat Sun	13:25	6:10	9	0.256944
6354	Luninets	Every	6:40	2:30	3	0.104167
3548	Vilno	Mon Sun	13:20	6:45	8	0.28125
9703	Baranovichi	Mon Fri Sat	18:32	1:30	5	0.0625
2591	Warszawa	Every	14:18	4:50	7	0.201389
5924	Praga	Tue Sun	7:40	9:35	13	0.399306
3951	Mogilev	Mon Tue Fri	14:55	7:30	8	0.3125
5625	Kobrin	Mon Sat	4:00	0:50	1	0.0347222
7688	Berlin	Every	5:20	21:40	17	0.902778
7632	Orsha	Mon Sat	7:30	6:42	5	0.279167

Рисунок 4.10 — Вывод после добавления

В пункте 3 осуществляется изменение полей записи. Для изменения требуется ввод номера поезда для его изменения (см. рисунок 4.11). Далее появляется меню изменения полей. Здесь можно выбрать поле для изменения и ввести его новое значение (см. рисунок 4.12).

Введите номер поезда для изменения: 7632|

Рисунок 4.11 – Ввод номера для изменения

Выберите действие: 1|

Номер поезда:1573|

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках
1573	Orsha	Mon Sat	7:30	6:42	5	0.279167

Рисунок 4.12 — Ввод нового значения поля

В пункте номер 4 осуществляется удаление записей. Первично выводится меню удаления (см.рисунок 4.13), где предлагается выбрать поле для удаления.

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках
2133	Minsk	Every	6:00	3:00	9	0.125
4513	Smolensk	Mon Fri Sat	9:00	14:00	12	0.583333
3815	Moskva	Every	12:45	15:35	19	0.649306
7618	Grodno	Sat Sun	13:25	6:10	9	0.256944
6354	Luninets	Every	6:40	2:30	3	0.104167
3548	Vilno	Mon Sun	13:20	6:45	8	0.28125
9703	Baranovichi	Mon Fri Sat	18:32	1:30	5	0.0625
2591	Warszawa	Every	14:18	4:50	7	0.201389
5924	Praga	Tue Sun	7:40	9:35	13	0.399306
3951	Mogilev	Mon Tue Fri	14:55	7:30	8	0.3125
5625	Kobrin	Mon Sat	4:00	0:50	1	0.0347222
7688	Berlin	Every	5:20	21:40	17	0.902778
1573	Orsha	Mon Sat	7:30	6:42	5	0.279167

=====DELETE=====

1 удаление записи по номеру
2 удаление записей по названию станции
3 удаление записей по времени отправления
4 удаление записей по времени в пути
5 удаление записей по количеству остановок
0 отмена

Выберите тип удаления: |

Рисунок 4.13 – Меню удаления

Выберем удаление по номеру. В этом случае удаляется одна запись с этим номером (см. рисунок 4.14).

Выберите тип удаления: 1
Введите номер: 1573|

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках
2133	Minsk	Every	6:00	3:00	9	0.125
4513	Smolensk	Mon Fri Sat	9:00	14:00	12	0.583333
3815	Moskva	Every	12:45	15:35	19	0.649306
7618	Grodno	Sat Sun	13:25	6:10	9	0.256944
6354	Luninets	Every	6:40	2:30	3	0.104167
3548	Vilno	Mon Sun	13:20	6:45	8	0.28125
9703	Baranovich	Mon Fri Sat	18:32	1:30	5	0.0625
2591	Warszawa	Every	14:18	4:50	7	0.201389
5924	Praga	Tue Sun	7:40	9:35	13	0.399306
3951	Mogilev	Mon Tue Fri	14:55	7:30	8	0.3125
5625	Kobrin	Mon Sat	4:00	0:50	1	0.0347222
7688	Berlin	Every	5:20	21:40	17	0.902778

Рисунок 4.14 – Удаление по номеру

При выборе удаления по другому полю удаляются несколько записей с таким значением (см. рисунок 4.15).

Выберите тип удаления: 2
Введите название станции: Мо

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках
2133	Minsk	Every	6:00	3:00	9	0.125
4513	Smolensk	Mon Fri Sat	9:00	14:00	12	0.583333
7618	Grodno	Sat Sun	13:25	6:10	9	0.256944
6354	Luninets	Every	6:40	2:30	3	0.104167
3548	Vilno	Mon Sun	13:20	6:45	8	0.28125
9703	Baranovich	Mon Fri Sat	18:32	1:30	5	0.0625
2591	Warszawa	Every	14:18	4:50	7	0.201389
5924	Praga	Tue Sun	7:40	9:35	13	0.399306
5625	Kobrin	Mon Sat	4:00	0:50	1	0.0347222
7688	Berlin	Every	5:20	21:40	17	0.902778

Рисунок 4.15 – Удаление нескольких записей по названию станций

В пункте номер 5 осуществляется сортировка записей. Для начала выводится меню сортировки. Далее выбирается поле для сортировки (см. рисунок 4.16).

```

=====SORT MENU=====

1      сортировка по номеру
2      сортировка по конечной станции
3      сортировка по времени отправления
4      сортировка по времени в пути
5      сортировка по количеству остановок
0      отмена

Выберите тип сортировки: |

```

Рисунок 4.16 – Меню сортировки записей

В качестве примера отсортируем записи по номерам в порядке возрастания (см. рисунок 4.17) и по названиям станций в обратном порядке (см. рисунок 4.18).

```

=====SORT MENU=====

1      сортировка по номеру
2      сортировка по конечной станции
3      сортировка по времени отправления
4      сортировка по времени в пути
5      сортировка по количеству остановок
0      отмена

Выберите тип сортировки: 1

=====REVERSE=====

0      прямая сортировка (от меньшего к большему)
1      обратная сортировка (от большего к меньшему)

Выберите порядок сортировки: 0

=====FILE=====

0      без записи в файл
1      с записью в файл

Выберите тип записи в файл: 0

```

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках
2133	Minsk	Every	6:00	3:00	9	0.125
2591	Warszawa	Every	14:18	4:50	7	0.201389
3548	Vilno	Mon Sun	13:20	6:45	8	0.28125
4513	Smolensk	Mon Fri Sat	9:00	14:00	12	0.583333
5625	Kobrin	Mon Sat	4:00	0:50	1	0.0347222
5924	Praga	Tue Sun	7:40	9:35	13	0.399306
6354	Luninets	Every	6:40	2:30	3	0.104167
7618	Grodno	Sat Sun	13:25	6:10	9	0.256944
7688	Berlin	Every	5:20	21:40	17	0.902778
9703	Baranovichi	Mon Fri Sat	18:32	1:30	5	0.0625

Рисунок 4.17 – Сортировка записей по номерам в порядке возрастания

```

=====SORT MENU=====

1      сортировка по номеру
2      сортировка по конечной станции
3      сортировка по времени отправления
4      сортировка по времени в пути
5      сортировка по количеству остановок
0      отмена

Выберите тип сортировки: 2

=====REVERSE=====

0      прямая сортировка (от меньшего к большему)
1      обратная сортировка (от большего к меньшему)

Выберите порядок сортировки: 1

=====FILE=====

0      без записи в файл
1      с записью в файл

Выберите тип записи в файл: 0

```

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках
2591	Warszawa	Every	14:18	4:50	7	0.201389
3548	Vilno	Mon Sun	13:20	6:45	8	0.28125
4513	Smolensk	Mon Fri Sat	9:00	14:00	12	0.583333
5924	Praga	Tue Sun	7:40	9:35	13	0.399306
2133	Minsk	Every	6:00	3:00	9	0.125
6354	Luninets	Every	6:40	2:30	3	0.104167
5625	Kobrin	Mon Sat	4:00	0:50	1	0.0347222
7618	Grodno	Sat Sun	13:25	6:10	9	0.256944
7688	Berlin	Every	5:20	21:40	17	0.902778
9703	Baranovichi	Mon Fri Sat	18:32	1:30	5	0.0625

Рисунок 4.18 – Сортировка записей по названиям станции в обратном порядке

В пункте номер 6 производится выборка по указываемому значения (промежутку значений) поля. В начале выводится меню выборки (см. рисунок 4.19).

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках
2133	Minsk	Every	6:00	3:00	9	0.125
4513	Smolensk	Mon Fri Sat	9:00	14:00	12	0.583333
7618	Grodno	Sat Sun	13:25	6:10	9	0.256944
6354	Luninets	Every	6:40	2:30	3	0.104167
3548	Vilno	Mon Sun	13:20	6:45	8	0.28125
9703	Baranovichi	Mon Fri Sat	18:32	1:30	5	0.0625
2591	Warszawa	Every	14:18	4:50	7	0.201389
5924	Praga	Tue Sun	7:40	9:35	13	0.399306
5625	Kobrin	Mon Sat	4:00	0:50	1	0.0347222
7688	Berlin	Every	5:20	21:40	17	0.902778


```
=====SELECT=====
```

- 1 выборка записей по номеру
- 2 выборка записей по названию станции
- 3 выборка записей по времени отправления
- 4 выборка записей по времени в пути
- 5 выборка записей по количеству остановок
- 0 отмена

Рисунок 4.19 – Меню выборки

Для тестирования выполним выборку по количеству остановок в диапазоне от 3 до 12 (см. рисунок 4.20) и по времени отправления в диапазоне от 8:00 до 10:30 (см. рисунок 4.21).

```
=====SELECT=====
```

- 1 выборка записей по номеру
- 2 выборка записей по названию станции
- 3 выборка записей по времени отправления
- 4 выборка записей по времени в пути
- 5 выборка записей по количеству остановок
- 0 отмена

Выберите тип выборки: 5
Введите диапазон значений
Нижняя граница: 3
Верхняя граница: 12

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках
2133	Minsk	Every	6:00	3:00	9	0.125
4513	Smolensk	Mon Fri Sat	9:00	14:00	12	0.583333
7618	Grodno	Sat Sun	13:25	6:10	9	0.256944
6354	Luninets	Every	6:40	2:30	3	0.104167
3548	Vilno	Mon Sun	13:20	6:45	8	0.28125
9703	Baranovichi	Mon Fri Sat	18:32	1:30	5	0.0625
2591	Warszawa	Every	14:18	4:50	7	0.201389

Количество записей найдено: 7

Рисунок 4.20 – Выборка по количеству остановок в диапазоне от 3 до 12

```

=====SELECT=====
1      выборка записей по номеру
2      выборка записей по названию станции
3      выборка записей по времени отправления
4      выборка записей по времени в пути
5      выборка записей по количеству остановок
0      отмена

Выберите тип выборки: 3
Введите диапазон значений
Нижняя граница: 8:00
Верхняя граница: 10:30
Номер|      Конечная станция|      Дни следования| Отправление|Время в пути| Кол-во остановок|Время в пути в сутках
4513|      Smolensk|      Mon Fri Sat|      9:00|      14:00|      12|      0.583333
Количество записей найдено: 1

```

Рисунок 4.21 – Выборка по времени отправления в диапазоне от 8:00 до 10:30

В пункте номер 7 выполняется отмена последнего действия. Для теста удалим одну запись (см. рисунок 4.22). Далее выберем пункт 7 и выведем записи в консоль (см. рисунок 4.23).

```

=====DELETE=====
1      удаление записи по номеру
2      удаление записей по названию станции
3      удаление записей по времени отправления
4      удаление записей по времени в пути
5      удаление записей по количеству остановок
0      отмена

Выберите тип удаления: 1
Введите номер: 7688

```

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках
2133	Minsk	Every	6:00	3:00	9	0.125
4513	Smolensk	Mon Fri Sat	9:00	14:00	12	0.583333
7618	Grodno	Sat Sun	13:25	6:10	9	0.256944
6354	Luninets	Every	6:40	2:30	3	0.104167
3548	Vilno	Mon Sun	13:20	6:45	8	0.28125
9703	Baranovich	Mon Fri Sat	18:32	1:30	5	0.0625
2591	Warszawa	Every	14:18	4:50	7	0.201389
5924	Praga	Tue Sun	7:40	9:35	13	0.399306
5625	Kobrin	Mon Sat	4:00	0:50	1	0.0347222

Рисунок 4.22 – Удаление записи

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках
2133	Minsk	Every	6:00	3:00	9	0.125
4513	Smolensk	Mon Fri Sat	9:00	14:00	12	0.583333
7618	Grodno	Sat Sun	13:25	6:10	9	0.256944
6354	Luninets	Every	6:40	2:30	3	0.104167
3548	Vilno	Mon Sun	13:20	6:45	8	0.28125
9703	Baranovich	Mon Fri Sat	18:32	1:30	5	0.0625
2591	Warszawa	Every	14:18	4:50	7	0.201389
5924	Praga	Tue Sun	7:40	9:35	13	0.399306
5625	Kobrin	Mon Sat	4:00	0:50	1	0.0347222
7688	Berlin	Every	5:20	21:40	17	0.902778

Рисунок 4.23 – Вывод записей в консоль

The screenshot shows a code editor window titled "output_file.txt — Kate". The editor displays a table of train schedule data. The table has 11 columns: "Номер" (Number), "Конечная станция" (Final station), "Дни следования" (Days of travel), "Отправление" (Departure), "Время в пути" (Travel time), "Кол-во остановок" (Number of stops), "Время в пути в сутках" (Travel time in days), and "Время в пути в сутках" (Travel time in days). The data is organized into 12 rows, with the first row being the header and the subsequent rows representing individual train routes. The routes are numbered 1 through 12. The stations listed are Minsk, Smolensk, Grodno, Luninets, Vilno, Baranovichi, Warshawa, Praga, Kobrin, and Berlin. The days of travel are listed as "Every", "Mon Fri Sat", "Sat Sun", "Mon Sun", "Mon Fri Sat", "Every", "Tue Sun", and "Mon Sat". The departure times are listed as 6:00, 9:00, 13:25, 6:40, 13:20, 18:32, 14:18, 7:40, 4:00, and 5:20. The travel times are listed as 3:00, 14:00, 6:10, 2:30, 6:45, 1:30, 4:50, 9:35, 0:50, and 21:40. The number of stops are listed as 9, 12, 9, 3, 8, 5, 7, 13, 1, and 17. The travel time in days is listed as 0.125, 0.583333, 0.256944, 0.104167, 0.28125, 0.0625, 0.201389, 0.399306, 0.0347222, and 0.902778.

Номер	Конечная станция	Дни следования	Отправление	Время в пути	Кол-во остановок	Время в пути в сутках	Время в пути в сутках
1	2133	Minsk	Every	6:00	3:00	9	0.125
2	4513	Smolensk	Mon Fri Sat	9:00	14:00	12	0.583333
3	7618	Grodno	Sat Sun	13:25	6:10	9	0.256944
4	6354	Luninets	Every	6:40	2:30	3	0.104167
5	3548	Vilno	Mon Sun	13:20	6:45	8	0.28125
6	9703	Baranovichi	Mon Fri Sat	18:32	1:30	5	0.0625
7	2591	Warshawa	Every	14:18	4:50	7	0.201389
8	5924	Praga	Tue Sun	7:40	9:35	13	0.399306
9	5625	Kobrin	Mon Sat	4:00	0:50	1	0.0347222
10	7688	Berlin	Every	5:20	21:40	17	0.902778
11							
12							

[illegible]

ЗАКЛЮЧЕНИЕ

Результатом является работающая программа, содержащая в себе базу данных «Вокзал». Удобный интерфейс, возможность записи данных в файл упрощает работу с большим количеством записей, что является основной целью данной курсовой работы.

Все условия и задачи курсовой работы были успешно решены, программа функционирует исправно. Весь требуемый функционал присутствует в программе, она в полной мере обеспечивает требования пользователя. Возникновение ошибок и исключений сведено к минимуму. Программа простая и удобная в эксплуатации и готова для использования.

В ходе выполнения работы были практически закреплены знания о работе с массивами структур данных, инструментами файлового ввода и вывода, основными языковыми средствами языка C++. Опыт в разработке программы, обрабатывающей структуры данных, важен для каждого программиста и поможет в разработке более сложных программ в будущем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Объектно-ориентированное программирование в С++ 4-е издание, Роберт Лафоре.
2. 2020 Алгоритмы и структуры данных (С++) Тимофей Хирьянов — Режим доступа: https://www.youtube.com/playlist?list=PLRDzFCPr95fL_5Xvnufpwj2uYZnZBBnsr
3. Уроки программирования на языке С++ – Режим доступа: <https://ravesli.com/uroki-cpp/> - Дата доступа: 04.05.2021.
4. ГОСТ 19.504 – 79. ЕСПД. Руководство программиста. Требования к содержанию и оформлению.
5. ГОСТ 2.105-95. ЕСКД. Общие требования к текстовым документам