ЗМІСТ

ВСТУП…………………………………………………………………………….……3

РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ……………………………………………………….…4

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАММНОГО ЗАБЕЗПЕЧЕННЯ……………………………………………………………………...6

РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ……………………………………………………………………….11

ВИСНОВКИ……………………………………………………………….………….17

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……………………………………………18

ДОДАТОК…...………………………………………………………….……………19

ВСТУП

Метою данної навчальної практики є створення програмного додатку, що окремими функціями реалізує наступні функції:

* Введення даних з клавіатури;
* Дані мають зберігатися у вигляді файлів (бінарних, текстових).
* Виведення даних на екран як всіх так і за параметрами (кількість параметрів - 3).
* Пошук даних за параметрами (кількість параметрів - 3).
* Редагування даних. Перед редагуванням знайти потрібний запис.
* Видалення даних. Перед видаленням знайти потрібний запис.
* Впорядкування записів за зростанням або спаданням;
* Програма має містити за необхідності декілька рівнів меню та здійснювати перехід між різними рівнями меню.

В першому розділі було зроблено аналіз задачі. Ознайомився зі змістом завдання, визначив необхідність застосування знань та навичок отриманих на теоретичних та практичних заняттях з дисципліни «Основи програмування».

Мною було створено макет програми, який я пізніше вдосконалив. Далі почалась розробка програмного забезпечення, його тестування та виправлення помилок в його роботі.

На основі виконаної практичної частини зроблено опис роботи з програмним додатком та його тестування.

В результаті виконаної роботи я набув практичних навичок в роботі зі структурами даних та файлам (бінарними та текстовими). Такий програмний продукт дозволяє автоматизувати облік інформації по рейсам, її систематизації та пошук необхідних данних по рейсам.

РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ

Завданням моєї навчальної практики було створення такого програмного додатку – структури під назвою «Рейс» яка мала б такі поля: марка автомобіля; номер автомобіля; пункт призначення; вантажопідйомність (у тонах); вартість одиниці вантажу; загальна вартість вантажу; дата відправлення (рік, місяць, число); дата прибуття (рік, місяць, число).

Структура - це об'єднання одного або більше об'єктів (змінних, масивів, покажчиків, інших структур). Як і [масив](http://ua-referat.com/%D0%9C%D0%B0%D1%81%D0%B8%D0%B2), вона являє собою сукупність даних, але відрізняється від нього тим, що до її елементів необхідно звертатися на ім'я, а її різні елементи не обов'язково має належати до одного типу.

Структури зручно використовувати там, де різноманітні дані, що відносяться до одного і [того](http://ua-referat.com/%D0%A2%D0%BE%D0%B3%D0%BE) ж об'єкту, необхідно об'єднувати. Наприклад, учня середньої школи характеризують такі дані: прізвище, ім'я, дата народження, клас, вік.

Оголошення структури здійснюється за допомогою ключового слова struct, за яким необхідно вказати її тип, список елементів, укладених у фігурні дужки. Її можна представити в наступному загальному вигляді:

struct тип {тип елемента 1 ім'я елемента 1;

тип елемента n ім'я елемента n;};

Іменем елемента може бути будь-який ідентифікатор. В одному рядку можна записувати через кому кілька ідентифікаторів одного типу.

Наприклад:

struct Car

{

char brand[x];

int number;

char destination[x];

};

Таким чином мною було створено програму, в якій написано функії, що виконують такі функції:

* Введення інформації про рейс (марка автомобіля; номер автомобіля; пункт призначення; вантажопідйомність (у тонах); вартість одиниці вантажу; загальна вартість вантажу; дата відправлення (рік, місяць, число); дата прибуття (рік, місяць, число)) з клавіатури;
* Виведення всіх даних на екран (з метою ознайомлення з базою даних про рейси, перевірка повноти введеної інформації);
* Пошук даних за параметрами (за маркою автомобіля, номером автомобіля та загальною ціною вантажу);
* Редагування даних (за необхідності зміни будь-яких даних про рейс);
* Видалення даних (проводиться при прибутті рейсу на місце призначення або його відміні);
* Впорядкування записів за зростанням або спаданням (для зручності роботи з базою даних).

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАММНОГО ЗАБЕЗПЕЧЕННЯ

Програмне забезпечення являє собою сукупність програм, призначених для розв’язання завдань на комп’ютері. Програма – впорядкований набір команд.

Розробка програмного забезпечення включає в себе багато стадій: проектування, програмування, тестування, впровадження і підтримку.

Проектування починається із формулювання вимог до програмного забезпечення і створення специфікацій — документів, у яких описані функції, що їх повинна виконувати програма. На наступному етапі створюється загальний дизайн програми: розбиття її на окремі блоки і визначення взаємодії між ними. На етапі безпосереднього програмування створюється текстовий код програми на одній чи декількох мовах програмування. Після компіляції коду, програмний продукт обов'язково проходить тестування, у процесі якого визначається відповідність продукту специфікаціям, знаходяться і виправляються помилки.

Перед впровадженням програмний продукт потребує документації — опису можливостей, посібників користувача, системи допомоги. Після впровадження програмного забезпечення, що для програмних продуктів вимагає маркетингу, системи дистрибуції, реклами тощо, програмне забезпечення потребує підтримки. Необхідність у підтримці виникає внаслідок швидкого розвитку комп'ютерів, що зумовлює необхідність взаємодії програмного продукту з іншими, новішими програмами і новою матеріальною базою. Часто підтримка нових можливостей забезпечується випуском нових версій програмного продукту.

Тому спочатку мною було створено макет меню програми, зі врахуванням всіх функцій, які має виконувати моя програма.

Меню

Введення даних

Виведення даних

Сортування

За спаданням

За зростанням

Усі дані

Пошук

Видалення та редагування

Вихід

Рис 2.1. Схема меню програми

Далі я приступив до безпосереднього написання програмного коду. Спочатку я створив структуру «Car», яка містила такі данні: марка автомобіля; номер автомобіля; пункт призначення; вантажопідйомність (у тонах); вартість одиниці вантажу; загальна вартість вантажу; дата відправлення; дата прибуття; спеціальність та структури «date» Структура «date» містила в собі інформацію про рік, місяць та число відправлення та прибуття рейсу

Лістинг структур

struct date

{

int day;

int month;

int year;

};

struct Car

{

char brand[20];

int number;

char destination[20];

int massT;

int costOfTonn;

int costOfCargo;

date departure;

date arrive;

};

Наступним кроком було забезпечення введення інформації з клавіатури. Я створив функцію типу Student під назвою «AddReisData».

Функція - модуль, що містить деяку послідовність операцій. Її розробка та

реалізація у програмі може розглядатися як побудова операцій, що вирішують конкретну задачу (підзадачу). Однак взагалі функція може розглядатися окремо як єдина абстрактна операція, і, щоб її використовувати, користувачеві необхідно зрозуміти інтерфейс функції - її вхідні дані та результати виконання. Легко буде зрозуміти ту функцію, що відповідає абстрактним операціям, необхідним для рішення задачі. Функцію та її використання у програмі можна у такому разі представляти у термінах задачі, а не в деталях реалізації. Припустимо, необхідно розробити функціональний модуль, що розв'язує наступне завдання: існує вхідний список певних даних, який необхідно відсортувати, переставляючи його елементи у визначеному порядку. Ця функція може бути описана, як абстрактна операція сортування даних, що може бути частиною вирішення деякої підмножини задач. Функція, що реалізує цю операцію, може бути використана у багатьох програмах, якщо вона створена як абстракція, що не залежить від реалізації (контексту програми).

З функцією AddReisData, як і з усім програмним кодом, можна буде ознайомитися в додатках до звіту.

Також я забезпечив виведення на екран введеної інформаці, за допомогою функцій ReisPrint та ReisPrints. Функція ReisPrint забезпечує виведення інформації про одного студента на екран, а функція ReisPrints – виведення масиву студентів.

Наступним була створена функція виведення меню програми на монітор та зчитуваня інформації про обраний пункт меню.

Лістинг функції MeinMenu:

int MeinMenu()

{

int num;

printf("\t---Меню---\n");

printf("1. Додати запис \n");

printf("2. Виведення даних по рейсам \n");

printf("3. Зберегти \n");

printf("4. Завантажити\n");

printf("5. Сортування \n");

printf("6. Редагування даних \n");

printf("0. Вихід\n");

printf("-> ");

num = NumberCheckInn();

return num;

}

Головна функція містить в собі створення масиву рейсів, кейси меню, виклики функцій та інші функціональні частини програмного коду.

Далі я забезпечив роботу з файлами.

Файл — це впорядкована [сукупність](https://uk.wikipedia.org/w/index.php?title=%D0%A1%D1%83%D0%BA%D1%83%D0%BF%D0%BD%D1%96%D1%81%D1%82%D1%8C&action=edit&redlink=1) даних, що зберігається на диску і займає іменовану область зовнішньої пам'яті. Згідно з термінологією, прийнятою в програмуванні, файл — це довільний блок інформації, або пристрій вводу-виводу, асоційований із ним (як середовище, засіб передачі "довільних блоків інформації").

Мова Сі не має спеціальних операторів для роботи з файлами, цю роботу виконують функції. Всі вони описані в уже відомому нам заголовковому файлі stdio.h. Різні компілятори можуть мати різну кількість функцій вводу-виводу.

При використаннi файлу його обов’язково необхідно оголосити, це робиться так:

FILE \*f1,\*f2, ... ,\*fn;

Тут \*f1,...,\*fn – вказівники на файли, одночасно можна оголосити n вказівників. Кожному файлу вiдповiдає свiй вказівник, ця вiдповiднiсть встановлюється в момент вiдкриття файлiв. Він являє собою адресу пам’яті, де знаходяться відомості про буфер: адреса буфера, ступінь його заповнення та ім’я фізичного файлу, для якого він призначений.

Блок-схема — Представлення алгоритму розв'язування або аналізу задачі за допомогою геометричних елементів (блоків), які позначають операції, потік, дані тощо.

Блок вхідних та вихідних даних прийнято позначати [паралелограмом](https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%B5%D0%BB%D0%BE%D0%B3%D1%80%D0%B0%D0%BC), [блок обчислень](https://uk.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) (обробки) даних — [прямокутником](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D1%8F%D0%BC%D0%BE%D0%BA%D1%83%D1%82%D0%BD%D0%B8%D0%BA), [блок прийняття рішень](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%B0_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D1%96_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F) — [ромбом](https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%BC%D0%B1), [еліпсом](https://uk.wikipedia.org/wiki/%D0%95%D0%BB%D1%96%D0%BF%D1%81) — початок та кінець алгоритму.

У [автоматиці](https://uk.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) функціональна схема, або блок-схема [САР](https://uk.wikipedia.org/wiki/%D0%A1%D0%90%D0%A0), складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками,

які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема зображаються більш детально, що полегшує її читання та ілюструє принцип роботи.

Нижче зображено блок-схему роботи програмного додатку.

Блок-схема роботи програмного додатку

Початок

Кінець

Меню

Обраний пункт

Введення даних

Виведення даних

Робота з бінарним та текстовим файлами

Редагування/

Видалення даних

Сортування

Вихід

РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ

3.1. Опис роботи

При запуску програмного додатку буде попередньо завантажено файл, якщо він існує, після чого на екрані з’явиться повідомлення що просигналізує упішне завантаження “Файл успішно завантажено”, в іншому випадку з’явиться повідомлення “Файл не існує, створіть нові дані”. Після чого на екрані з’являється вікно консолі з пунктами меню. З клавіатури потрібно ввести номер пункту меню, який ви обираєте.

Наприклад, якщо обрати перший пункт меню, тобто додавання інформації, з’явиться рядок, який буде викликано функцію яка дозволить вводити дані.

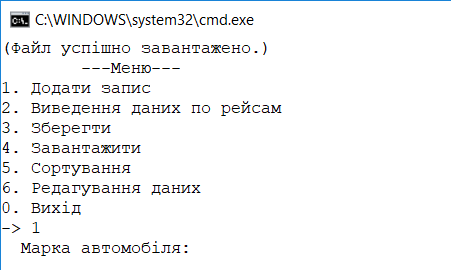


Рис 3.1.1. Головне меню програми та приклад роботи першого пункту

Після введення всіх даних програма поверне вас до головного меню. Обравши другий пункт плану відбудеться виведення підменю.

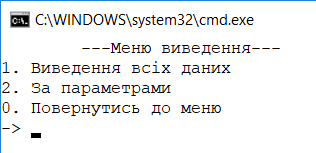


Рис 3.1.2. Приклад роботи підменю другого пункту меню

Перший пункт підменю виводить всі наявні дані на екран.



Рис 3.1.3. Приклад роботи функції виведення

Другий пункт підменю викликає виведення за параметрами.

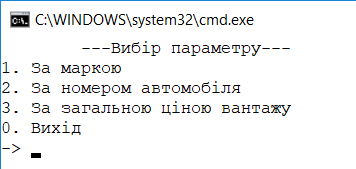


Рис 3.1.4. Приклад підменю виведення за параметрами

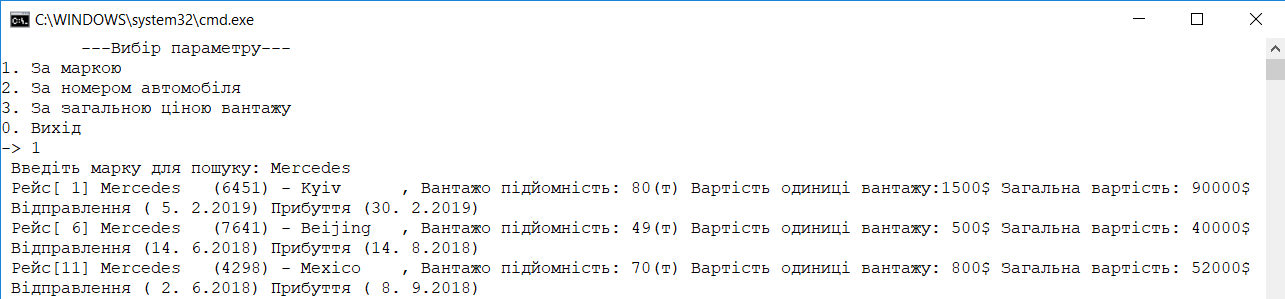


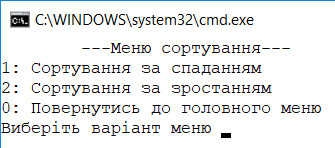
Рис 3.1.5. Приклад роботи підменю виведення за параметрами

Виведення за параметрами має три параметрами за якими можна здійснити пошук потрібних даних і наступним виведенням знайденої інформації на екран.

Третій пункт меню(можна побачити на Рис 3.1.1) викликає функцію що зберігає дані у бінарний та текстовий файли, після чого повертає користувача до головного меню. Дозволяє попередити втратити дані, якщо користувач необережно завершить роботу програми.

Четвертий пункт меню(можна побачити на Рис 3.1.1) викликає функцію що завантажує дані з бінарного файло до динамічно виділеного масиву програми, після чого повертає користувача до головного меню. Завантаження даних дозволяє спростити роботу користувачу з наступною передачою інформації за потрібності.

У п’ятому пункті меню(можна побачити на Рис 3.1.1) знаходиться функція сортування. Після вибору цього пункту з’явиться ще одне вкладене меню. В ньому знаходяться параметри для сортування за параметром номера за зростанням або спаданням.



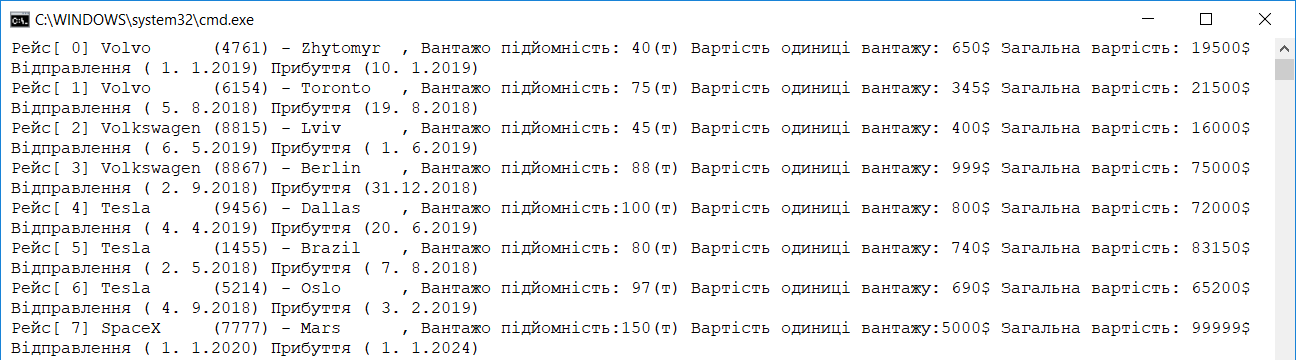
Рис 3.1.6. Приклад підменю п’ятого пункту.

Рис 3.1.7. Приклад виконання п’ятого пункту.

Після вибору сортування(за спаданням або зростанням) результат можна побачити після виведення даних на екран другим пунктом меню.

Шостий пункт меню виводить на екран всі дані, після чого пропонує користувачу ввести номер рейсу, який буде змінено або видалено.

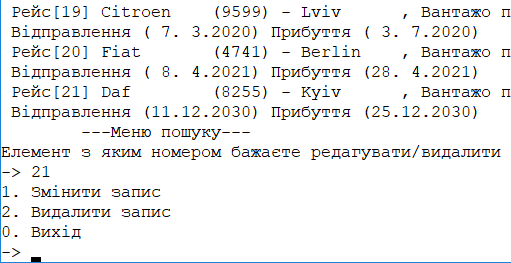


Рис 3.1.8. Приклад роботи з підменю редагування даних.

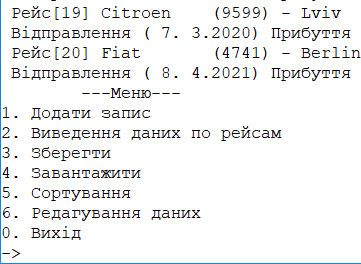


Рис 3.1.9. Приклад видалення даних.

Після вибору викликаються відповідні функції які дозволяють маніпулювати доступною інформацією.

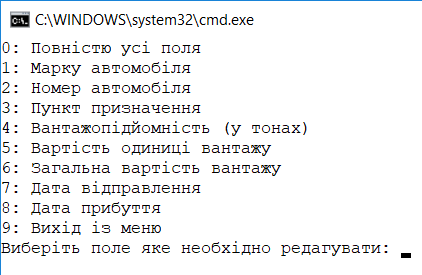


Рис 3.1.9. Приклад підменю зміни запису.

Сьомий(нульовий) пункт меню здійснює вихід з програми.

3.2. Тестування

Тестування програмного забезпечення — це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку [помилок](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D2%91) або інших дефектів, так і випробування програмних складових з метою оцінки. Може оцінюватись:

* відповідність вимогам, якими керувалися проектувальники та розробники
* правильна відповідь для усіх можливих вхідних даних
* виконання функцій за прийнятний час
* практичність
* сумісність з програмним забезпеченням та операційними системами
* відповідність задачам замовника.

Оскільки число можливих тестів навіть для нескладних програмних компонент практично нескінченне, тому стратегія тестування полягає в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як результат програмне забезпечення (ПЗ) тестується стандартним виконанням програми з метою виявлення [баґів](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D2%91) (помилок або інших дефектів).

Тестування ПЗ може надавати об'єктивну, незалежну інформацію про якість ПЗ, ризики відмови, як для користувачів так і для замовників.

Тестування може проводитись, як тільки створено виконуваний код (навіть частково завершено). [Процес розробки](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) зазвичай передбачає коли та як буде відбуватися тестування. Наприклад, при поетапному процесі, більшість тестів відбувається після визначення системних вимог і тоді вони реалізуються в тестових програмах. На противагу цьому, відповідно до вимог [гнучкої розробки ПЗ](https://uk.wikipedia.org/wiki/%D0%93%D0%BD%D1%83%D1%87%D0%BA%D0%B0_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F), програмування і тестування часто відбувається одночасно.

Я проведу тестування методом «Чорної скриньки».

Тестування «чорної скриньки»:

Відомі: функції програми.

Досліджується: робота кожної функції на всій області визначення.

Основне місце програми тестів «чорної скриньки» — інтерфейс ПЗ.

Ці тести демонструють:

* Як виконуються функції програми.
* Як приймаються вихідні дані.
* Як виробляються результати.
* Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе. Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10 . Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми. Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

* Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його IT);

Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Пункт меню | Дія | Результат | Помилки |
| 1. | Введення інформації | Введення інформації по рейсам | Дані коректно прийняті | Відсутні |
| 2. | Виведення інформації | Виведення даних на екран | Дані виведені коректно та в повному обсязі | Відсутні |
| 3. | Збереження | Дані зберігаються до текстового та бінарного файлу | Дані були коректно збережено в файли | Відсутні |
| 4. | Завантаження | Завантаження даних із файлу | Інформація була завантажена у повному обсязі | Відсутні |
| 5. | Сортування | Сортування за зростанням та спаданням | Інформацію було відсортовано коректно | Відсутні |
| 6. | Редагування даних | Редагування даних про рейс та їх видалення | Дані було відредаговано та збережено. Далі процес видалення пройшов успішно | Відсутні |
| 7. | Вихід | Вихід з програмного додатку | Програма завершила свою роботу та консольне вікно закрилось. | Відсутні |

При тестуванні програми помилок не було виявлено. Програмний додаток працює коректно.

ВИСНОВКИ

У ході виконання практичних завдань навчальної практики мною було отримано нові знання, уміння і навички у роботі зі структурами та файлами, у результаті чого створено програмний продукт – структуру під назвою «Рейс» яка має такі поля: марка автомобіля; номер автомобіля; пункт призначення; вантажопідйомність (у тонах); вартість одиниці вантажу; загальна вартість вантажу; дата відправлення (рік, місяць, число); дата прибуття (рік, місяць, число). Цей програмний додаток може виконувати такі функції: введення та виведення інформації, її збереження, пошук та зміна, сортування і видалення. Також забезпечено роботу з файлами – бінарними та текстовими. Було створено схему його меню, а також блок-схему алгоритму.

В третьому розділі є опис роботи з цією програмою. Також було проведено тести роботи програмного додатку, було виявлено вразливі місця коду, такі як вічний цикл, виведення не всіх потрібних даних, можливість введення чисел замість літер та навпаки. Далі було проведено їх виправлення. Під час наступних тестів помилок в роботі програми виявлено не було.

Під час практики я набув досвіду в роботі зі структурами даних, багаторівневим меню, функціями, а також у роботі з файлами, який знадобиться мені в професійній діяльності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Войтенко В. В., Морозов А. В. С\С++ Практика програмування. Навчально методичний посібник - Житомир: ЖДТУ, 2003. – 324 с.
2. Зубенко В. В. Програмування: навчальний посібник (гриф МОН України) / В. В. Зубенко, Л. Л. Омельчук. — К. : ВПЦ «Київський університет», 2011. — 623 c.
3. Нікітченко М. С. Теоретичні основи програмування: навчальний посібник / М.С Нікітченко — Ніжин: Видавництво НДУ імені Миколи Гоголя, 2010. — 121с.
4. Павловская Т.А. С/С++. Программирование на языке высокого уровня. – СПб.: Питер, 2002. – 464 с.
5. <https://uk.wikipedia.org/wiki/Блок-схема>
6. <https://uk.wikipedia.org/wiki/Програмне_забезпечення>

ДОДАТКИ

Додаток А.

Лістинг програми

#include "stdafx.h"

#include <stdio.h>

#include <tchar.h>

#include <stdlib.h>

#include <windows.h>

#include <conio.h>

#include <string.h>

#include <math.h>

int N, X;

struct date

{

int day;

int month;

int year;

};

struct Car

{

char brand[20];

int number;

char destination[20];

int massT;

int costOfTonn;

int costOfCargo;

date departure;

date arrive;

};

Car\* buf = (Car\*)calloc(N, sizeof(Car));

int NumberCheckInn() {

int number;

do

{

if (scanf("%d", &number)) return number;

else printf(" Не є числом, спробуйте знову: ");

while (getchar() != '\n');

} while (1);

getchar();

}

void SaveFile()

{

system("cls");

FILE \*file = fopen("d:\\File\_Reis.bin", "w+b");

fwrite(&N, sizeof(int), 1, file);

fwrite(buf, sizeof(Car), N, file);

fclose(file);

FILE \*ft;

ft = fopen("D:\\Text\_File\_Reis.txt", "wt");

for (int i = 0; i < N; i++)

{

fprintf(ft," Рейс[%2d] %-10s (%4d) - %-10s, Вантажо підйомність:%3d(т) Вартість одиниці вантажу:%4d$ Загальна вартість:%6d$\n Відправлення (%2d.%2d.%4d) Прибуття (%2d.%2d.%2d)\n",

i, buf[i].brand, buf[i].number, buf[i].destination, buf[i].massT, buf[i].costOfTonn, buf[i].costOfCargo, buf[i].departure.day,

buf[i].departure.month, buf[i].departure.year, buf[i].arrive.day,

buf[i].arrive.month, buf[i].arrive.year);

}

fclose(ft);

}

void LoadFile()

{

system("cls");

FILE \*file;

file = fopen("d:\\File\_Reis.bin", "r+b");

fread(&N, sizeof(int), 1, file);

Car \*buf1 = (Car\*)malloc(N \* sizeof(Car));

fread(buf1, sizeof(Car), N, file);

fclose(file);

free(buf);

buf = buf1;

}

void PreLoadFile()

{

FILE \*file;

file = fopen("d:\\File\_Reis.bin", "r+b");

if (file)

{

LoadFile();

printf("(Файл успішно завантажено.)\n");

}

else

{

printf("(Файл не існує, створіть нові дані.)\n");

}

}

void ReadReis(int Num)

{

getchar();

do {

printf(" Марка автомобіля: ");

gets\_s(buf[Num].brand);

X = true;

X = isupper(\*buf[Num].brand);

} while (X != true);

do {

printf(" Номер автомобіля: ");

buf[Num].number = NumberCheckInn();

} while (buf[Num].number < 1000 || buf[Num].number > 9999);

getchar();

do {

printf(" Пункт призначення: ");

gets\_s(buf[Num].destination);

X = isupper(\*buf[Num].destination);

} while (X != true);

printf(" Вантажопідйомність (у тонах): ");

buf[Num].massT = NumberCheckInn();

printf(" Вартість одиниці вантажу: ");

buf[Num].costOfTonn = NumberCheckInn();

printf(" Загальна вартість вантажу: ");

buf[Num].costOfCargo = NumberCheckInn();

printf(" Дата відправлення\n");

do {

printf(" День: ");

buf[Num].departure.day = NumberCheckInn();

} while (buf[Num].departure.day < 0 || buf[Num].departure.day > 32);

do {

printf(" Місяць: ");

buf[Num].departure.month = NumberCheckInn();

} while (buf[Num].departure.month < 0 || buf[Num].departure.month > 13);

do {

printf(" Рік: ");

buf[Num].departure.year = NumberCheckInn();

} while (buf[Num].departure.year < 2018 || buf[Num].departure.year > 2100);

printf(" Дата прибуття\n");

do {

printf(" День: ");

buf[Num].arrive.day = NumberCheckInn();

} while (buf[Num].arrive.day < 0 || buf[Num].arrive.day > 32);

do {

printf(" Місяць: ");

buf[Num].arrive.month = NumberCheckInn();

} while (buf[Num].arrive.month < 0 || buf[Num].arrive.month > 13);

do {

printf(" Рік: ");

buf[Num].arrive.year = NumberCheckInn();

} while (buf[Num].arrive.year < 2018 || buf[Num].arrive.year > 2100);

getchar();

system("cls");

}

void ReisPrint(int i)

{

printf(" Рейс[%2d] %-10s (%4d) - %-10s, Вантажо підйомність:%3d(т) Вартість одиниці вантажу:%4d$ Загальна вартість:%6d$\n Відправлення (%2d.%2d.%4d) Прибуття (%2d.%2d.%2d)\n",

i, buf[i].brand, buf[i].number, buf[i].destination, buf[i].massT, buf[i].costOfTonn, buf[i].costOfCargo, buf[i].departure.day,

buf[i].departure.month, buf[i].departure.year, buf[i].arrive.day, buf[i].arrive.month, buf[i].arrive.year);

}

void ReisPrints()

{

system("cls");

for (int i = 0; i < N; i++)

{

ReisPrint(i);

}

}

void AddReisData() {

Car \*buf1 = (Car\*)malloc((N + 1) \* sizeof(Car));

for (int i = 0; i < N; i++)

buf1[i] = buf[i];

free(buf);

buf = buf1;

ReadReis(N);

N++;

}

void DeletReisData(int z)

{

if (z >= 0 && z <= N - 1) {

for (int i = z; i < N - 1; i++)

{

buf[i] = buf[i + 1];

}

Car \*buf1 = (Car\*)malloc((N - 1) \* sizeof(Car));

N--;

for (int i = 0; i < N; i++)

buf1[i] = buf[i];

free(buf);

buf = buf1;

}

}

int EditData(int z)

{

int choise;

system("cls");

printf("0: Повністю усі поля\n");

printf("1: Марку автомобіля\n");

printf("2: Номер автомобіля\n");

printf("3: Пункт призначення\n");

printf("4: Вантажопідйомність (у тонах)\n");

printf("5: Вартість одиниці вантажу\n");

printf("6: Загальна вартість вантажу\n");

printf("7: Дата відправлення\n");

printf("8: Дата прибуття\n");

printf("9: Вихід із меню\n");

printf("Виберіть поле яке необхідно редагувати: ");

scanf("%d", &choise);

system("cls");

switch (choise)

{

case 0: ReadReis(z);

break;

case 1:

getchar();

do {

printf(" Марка автомобіля: ");

gets\_s(buf[z].brand);

X = true;

X = isupper(\*buf[z].brand);

} while (X != true);

break;

case 2:

do {

printf(" Номер автомобіля: ");

buf[z].number = NumberCheckInn();

} while (buf[z].number < 1000 || buf[z].number > 9999);

break;

case 3:

getchar();

do {

printf(" Пункт призначення: ");

gets\_s(buf[z].destination);

X = isupper(\*buf[z].destination);

} while (X != true);

break;

case 4:

printf(" Вантажопідйомність (у тонах): ");

buf[z].massT = NumberCheckInn();

break;

case 5:

printf(" Вартість одиниці вантажу: ");

buf[z].costOfTonn = NumberCheckInn();

break;

case 6:

printf(" Загальна вартість вантажу: ");

buf[z].costOfCargo = NumberCheckInn();

break;

case 7:printf(" Дата відправлення\n");

do {

printf(" День: ");

buf[z].departure.day = NumberCheckInn();

} while (buf[z].departure.day < 0 || buf[z].departure.day > 32);

do {

printf(" Місяць: ");

buf[z].departure.month = NumberCheckInn();

} while (buf[z].departure.month < 0 || buf[z].departure.month > 13);

do {

printf(" Рік: ");

buf[z].departure.year = NumberCheckInn();

} while (buf[z].departure.year < 2018 || buf[z].departure.year > 2100);

break;

case 8:

printf(" Дата прибуття\n");

do {

printf(" День: ");

buf[z].arrive.day = NumberCheckInn();

} while (buf[z].arrive.day < 0 || buf[z].arrive.day > 32);

do {

printf(" Місяць: ");

buf[z].arrive.month = NumberCheckInn();

} while (buf[z].arrive.month < 0 || buf[z].arrive.month > 13);

do {

printf(" Рік: ");

buf[z].arrive.year = NumberCheckInn();

} while (buf[z].arrive.year < 2018 || buf[z].arrive.year > 2100);

break;

case 9:

return 0;

break;

}

}

int SerchForElement()

{

system("cls");

ReisPrints();

int num, z;

printf("\t---Меню пошуку---\n");

do {

printf("Елемент з яким номером бажаєте редагувати/видалити\n");

printf("-> ");

scanf\_s("%d", &z);

} while (z > N || z < 0);

printf("1. Змінити запис\n");

printf("2. Видалити запис\n");

printf("0. Вихід\n");

printf("-> ");

scanf\_s("%d", &num);

switch (num)

{

case 1: EditData(z); break;

case 2: DeletReisData(z); break;

case 0: break;

default: SerchForElement(); break;

}

return 0;

}

int MeinMenu()

{

int num;

printf("\t---Меню---\n");

printf("1. Додати запис \n");

printf("2. Виведення даних по рейсам \n");

printf("3. Зберегти \n");

printf("4. Завантажити \n");

printf("5. Сортування \n");

printf("6. Редагування даних \n");

printf("0. Вихід\n");

printf("-> ");

num = NumberCheckInn();

return num;

}

void ChoiseMenu()

{

system("cls");

int choise, resCount = 0;

printf("\t---Вибір параметру---\n");

printf("1. За маркою \n");

printf("2. За номером автомобіля \n");

printf("3. За загальною ціною вантажу \n");

printf("0. Вихід\n");

printf("-> ");

choise = NumberCheckInn();

switch (choise)

do {

case 1:

char searchBrand[20];

char searchBrand2[20];

int point;

printf(" Введіть марку для пошуку: "); getchar(); gets\_s(searchBrand);

for (int i = 0; i < N; i++)

{

strcpy(searchBrand2, buf[i].brand);

point = strcmp(searchBrand, searchBrand2);

if (point == NULL)

{

ReisPrint(i);

resCount++;

}

}

if (resCount == 0)

printf(" Рейсів з таким параметром не знайдено\n");

break;

case 2:

int cNum;

resCount = 0;

printf(" Номер автомобіля: ");

cNum = NumberCheckInn();

for (int i = 0; i < N; i++)

if (buf[i].number == cNum)

{

ReisPrint(i);

resCount++;

}

if (resCount == 0)

printf(" Рейсів з таким параметром не знайдено\n");

break;

case 3:

int fCost;

printf(" Загальна вартість вантажу: ");

fCost = NumberCheckInn();

for (int i = 0; i < N; i++)

if (buf[i].costOfCargo == fCost)

{

ReisPrint(i);

resCount++;

}

if (resCount == 0)

printf(" Рейсів з таким параметром не знайдено\n"); break;

case 0: break;

default: printf("Помилка вибору\n");

} while (choise != 0);

}

void PrintChoiseMenu()

{

system("cls");

int num;

printf("\t---Меню виведення---\n");

printf("1. Виведення всіх даних \n");

printf("2. За параметрами \n");

printf("0. Повернутись до меню \n");

printf("-> ");

num = NumberCheckInn();

switch (num)

do{

case 1: ReisPrints(); break;

case 2: ChoiseMenu(); break;

} while (num != 0);

}

int Sort()

{

system("cls");

int choise; int index = 1;

Car tmp;

printf("\t---Меню сортування---\n");

printf("1: Сортування за спаданням\n");

printf("2: Сортування за зростанням\n");

printf("0: Повернутись до головного меню\n");

do

{

printf("Виберіть варіант меню ");

scanf("%d", &choise);

if (choise == 3)

return 0;

if (choise == 1 || choise == 2)

break;

printf("Вибраний невідповідний пункт меню\n");

} while (true);

switch (choise)

{

case 1: {

while (index)

{

index = 0;

for (int i = 0; i + 1 < N; i++)

{

if (strcmp(buf[i].brand, buf[i + 1].brand) < 0)

{

tmp = buf[i];

buf[i] = buf[i + 1];

buf[i + 1] = tmp;

index = 1;

}

}

}

} break;

case 2: {

while (index)

{

index = 0;

for (int i = 0; i + 1 < N; i++)

{

if (strcmp(buf[i].brand, buf[i + 1].brand) > 0)

{

tmp = buf[i];

buf[i] = buf[i + 1];

buf[i + 1] = tmp;

index = 1;

}

}

}

}break;

}

return 0;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int num;

PreLoadFile();

do

{

num = MeinMenu();

switch (num)

{

case 1: AddReisData();break;

case 2: PrintChoiseMenu(); break;

case 3: SaveFile(); break;

case 4: LoadFile(); break;

case 5: Sort(); break;

case 6: SerchForElement(); break;

case 0: break;

default: printf("Помилка вибору\n");

}

} while (num != 0);

return 0;

}