Dokumentacja projektu:

"System Zarządzania Samochodami w Komisie"

Michał Koba, 177099, 2EF-DI

Spis treści

1. Wprowadzenie

Projekt "System Zarządzania Samochodami" jest aplikacją konsolową stworzoną w języku C++ umożliwiającą zarządzanie bazą danych samochodów w komisie. Program pozwala na dodawanie, edytowanie, usuwanie, filtrowanie, sortowanie oraz zapisywanie i wczytywanie samochodów z plików tekstowych.

2. Cele projektu

- Ułatwienie zarządzania samochodami w bazie.
- Umożliwienie sortowania i filtrowania danych według wielu kryteriów.
- Zapewnienie pełnej walidacji wprowadzanych danych.
- Umożliwienie pracy na plikach tekstowych w systemie metrycznym (kilometry, złote).
- Zapewnienie czytelnej obsługi i prezentacji danych dla użytkownika.

3. Struktura projektu

Pliki projektu:

- main.cpp — funkcja główna i menu użytkownika.
- BazaSamochodow.h — deklaracje klasy BazaSamochodow.
- BazaSamochodow.cpp — definicje funkcji klasy BazaSamochodow.
- Samochod.h — deklaracja klasy Samochod.
- Samochod.cpp — definicje funkcji klasy Samochod.

4. Użyte biblioteki i ich przeznaczenie

Biblioteka	W pliku	Przeznaczenie
<iostream>	main.cpp, BazaSamochodow.cpp	Obsługa wejścia i wyjścia tekstowego (np. cin, cout)
<fstream>	BazaSamochodow.cpp	Operacje na plikach tekstowych (ifstream, ofstream) — zapis i odczyt bazy danych
<string>	Samochod.h, BazaSamochodow.h, BazaSamochodow.cpp	Obsługa łańcuchów znaków (std::string)
<vector>	BazaSamochodow.h, BazaSamochodow.cpp	Przechowywanie dynamicznej listy samochodów (std::vector<Samochod>)
<algorithm>	BazaSamochodow.cpp	Użycie funkcji sortowania (std::sort)
<numeric>	BazaSamochodow.cpp	Obliczenia statystyczne, np. średnia cena (std::accumulate)
<limits>	BazaSamochodow.cpp	Ustawienie maksymalnych wartości liczbowych (std::numeric_limits)
<cmath>	BazaSamochodow.cpp	Obliczenia matematyczne, np. pierwiastek (std::sqrt)

5. Opis klas i funkcji

5.1 Klasa Samochod

Opis:
Reprezentuje pojedynczy samochód.

Składowe:

- marka, model — nazwa samochodu.
- rocznik — rok produkcji.
- przebieg — przebieg w kilometrach.
- cena — cena w złotych.

Funkcje:

- Gettery i settery (pobierzMarke(), ustawMarke() itd.) — służą do odczytu i ustawiania wartości pól.
- Konstruktor domyślny i parametryczny.

5.2 Klasa BazaSamochodow

Opis:
Obsługuje całą bazę samochodów (wektor vector<Samochod>).

Funkcje publiczne:

- dodajSamochod() — Dodaje nowy samochód wprowadzony przez użytkownika.
- edytujSamochod(int indeks) — Umożliwia edycję konkretnego parametru istniejącego samochodu.
- usunSamochod(int indeks) — Usuwa samochód z bazy po numerze.
- wyswietlWszystkie() — Wyświetla wszystkie samochody w tabeli.
- wyswietlWszystkieSortowanie() — Wyświetla samochody posortowane według wybranego kryterium.
- wyswietlSamochodyFiltrowanie() — Wyszukuje samochody spełniające wiele jednoczesnych kryteriów (marka, przebieg, cena, rocznik).
- posortujBaze() — Trwale sortuje bazę według wybranego kryterium.
- zapiszDoPliku(const string&) — Zapisuje aktualną bazę do pliku tekstowego.
- wczytajZPliku(const string&, bool dodaj = false) — Wczytuje samochody z pliku, z opcją dodawania lub nadpisania.
- generujStatystyki() — Generuje statystyki dotyczące samochodów (średnia cena, najtańszy, najdroższy, itd.).
- iloscSamochodow() — Zwraca ilość samochodów w bazie.

- wyswietlNaglowek() — Wyświetla nagłówek tabeli.
- wyswietlSamochod(size_t, const Samochod&) — Wyświetla jeden samochód w tabeli.
- wczytajInt(), wczytajDouble(), wczytajZnak() — Funkcje do bezpiecznego wczytywania danych od użytkownika.

Funkcje prywatne:

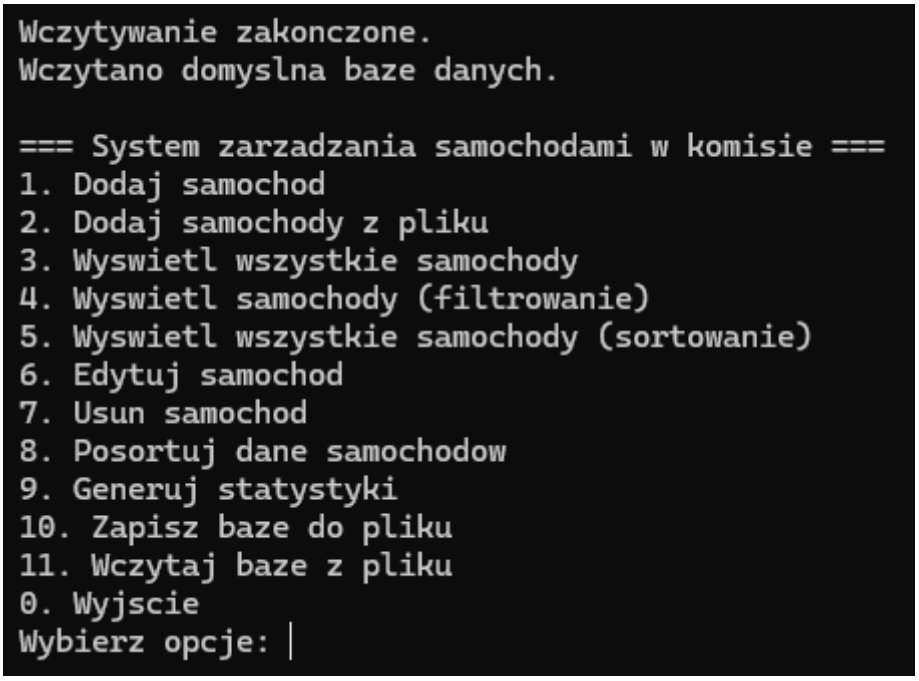
Funkcje generujące szczegółowe statystyki (średnia, odchylenie, mediany).

- void generujSredniaCene() const;
- void generujNajtanszySamochod() const;
- void generujNajdrozszySamochod() const;
- void generujMedianePrzebiegu() const;
- void generujOdchylenieStandardoweCen() const;
- void generujWszystkieStatystyki() const;
- void sortujPoWybranymKryterium(vector<Samochod>& bazaDoSortowania) const;

Gdzie funkcje są używane:

- dodajSamochod(), edytujSamochod(), usunSamochod(), wyswietlWszystkie() — bezpośrednio w głównym menu (main.cpp).
- wyswietlSamochodyFiltrowanie() — przy filtrowaniu.
- posortujBaze(), wyswietlWszystkieSortowanie() — przy sortowaniu.
- zapiszDoPliku(), wczytajZPliku() — przy zapisie i wczytywaniu bazy.
- generujStatystyki() — generowanie statystyk przy wyborze w menu.

6. Instrukcja użytkownika



Rys. 1 „Menu główne”

1. **Po uruchomieniu programu**:
    - Program automatycznie wczytuje bazę z pliku samochody.txt.
    - Wyświetlany jest komunikat: "Wczytano domyslna baze danych."
2. **Menu główne**:
    - Wybierz numer odpowiadający operacji:
        - 1 ➔ Dodaj nowy samochód.
        - 2 ➔ Dodaj samochody z innego pliku.
        - 3 ➔ Wyświetl wszystkie samochody.
        - 4 ➔ Wyswietl samochody (filtrowanie wielokryterialne).
        - 5 ➔ Wyswietl wszystkie samochody (sortowanie).
        - 6 ➔ Edytuj samochod (wybierz co konkretnie zmieniasz: markę, model, przebieg, cenę lub rocznik).
        - 7 ➔ Usun samochod.
        - 8 ➔ Posortuj baze według wybranego kryterium.
        - 9 ➔ Generuj statystyki.
        - 10 ➔ Zapisz aktualna baze do pliku.
        - 11 ➔ Wczytaj baze z pliku.

        - 0 ➔ Wyjście z programu.

3. **Podczas edycji / dodawania**:
    - Program oczekuje poprawnych danych: liczby dla rocznika, przebiegu, ceny.
    - Każda niepoprawna wartość jest odrzucana i użytkownik proszony jest o ponowne podanie poprawnej.
4. **Filtrowanie wielokryterialne**:
    - Wybierz, po których kryteriach chcesz filtrować (marka, cena, rocznik, przebieg).
    - Podaj zakresy, jeśli wybrałeś dane kryterium.
    - Wyniki można zapisać do osobnego pliku tekstowego.
5. **Zapisywanie do pliku**:
    - Wpisz nazwę pliku (np. baza_kopia.txt).
    - Dane zostaną zapisane w formacie tekstowym, rozdzielane średnikami.
6. **Wczytywanie z pliku**:
    - Wpisz nazwę pliku (np. baza_kopia.txt).
    - Można dodać nowe samochody lub nadpisać aktualną bazę.

7. Pliki źródłowe projektu

https://github.com/fen0x1k/projektcpppub