1. Условие

Ввести список машин, имеющихся в автомагазине, содержащий следующие сведения: марка автомобиля, страна-производитель (если иномарка – поддерживается обслуживание или нет), цена, цвет, состояние:

1. Новый:
    - Гарантия (в годах)
2. Не новый:
    - Год выпуска
    - Пробег
    - Количество собственников
    - Количество произведенных ремонтов

Вывести список не новых машин указанной марки с одним предыдущим собственником, отсутствием ремонтов в указанном диапазоне цен. Найти среди них иномарки с поддержкой обслуживания

2. Техническое задание

2.1 Исходные данные и результаты

Входные данные

1. Файл с данными: бинарный файл формата .bin;
2. Целое число — номер команды (от 0 до 10; См. Описание задачи);
3. Дополнительные данные (строка / число), в зависимости от вводимой информации и контекста.
4. **Выходные данные**

1. Таблица автомобилей
2. Отсортированная при помощи таблицы-ключей таблица автомобилей
3. Таблица ключей
4. Таблица с результатом выполнения задания
5. Характеристика сравнения способов сортировки таблицы.

2.2 Описание задачи, реализуемой программой

Программа реализует интерфейс управления таблицей машин, и позволяет выполнить следующие действия:

> Номер перечисления соответствует номеру команды в программе

0. Выход
1. Добавить машину в таблицу
2. Удалить машину из таблицы
3. Вывести таблицу машин на экран
4. Вывести таблицу машин в отсортированном виде (Сортировка происходит при помощи таблицы ключей)
5. Вывести таблицу ключей (ключ - цена)
6. Вывести отсортированную таблицу ключей
7. Отфильтровать машины согласно заданию (См. условие)
8. Вывести сравнительную таблицу, с данными о затраченных ресурсах на сортировку таблицы при помощи:
    - Быстрой сортировки
    - Таблицы ключей (Быстрая сортировка)
    - Сортировки вставками
    - Таблицы ключей (Сортировка вставками)
9. Загрузить таблицу автомобилей из файла data.bin
10. Сохранить текущее состояние таблицы автомобилей в файл data.bin

2.3 Способ обращения к программе

Взаимодействие с программой происходит через консольный интерфейс, входные данные вводятся пользователем с клавиатуры Запуск программы из рабочей директории:

```
./app.exe
```

2.4 Описание возможных аварийных ситуаций и ошибок пользователя

- Отсутствие файла из которого происходит попытка чтения / записи
- Неверная структура файла, из которого происходит чтение
- Переполнение таблицы

> В программе предусмотренна защита от неправильного ввода, поэтому она не завершится аварийно в этом случае.

3. Внутренние структуры данных

Структура записи (машины)

```
#define VARCHAR_LENGTH 30 // Максимальная длинна текстового поля

typedef struct
{
    char brand[VARCHAR_LENGTH + 1];
    char country[VARCHAR_LENGTH + 1];
    bool is_foreign;
    bool supports_maintain;
    unsigned long long price;
    char color[VARCHAR_LENGTH + 1];
    bool is_new;
    car_spec spec;
} car_t;
```

Структура вариантного поля

```
typedef struct
{
    unsigned warranty;
} car_spec_new;

typedef struct
{
    unsigned release_year;
    size_t mileage_km;
    size_t owners_amount;
    size_t repairs_amount;
} car_spec_secondhand;

typedef union
{
    car_spec_new new;
    car_spec_secondhand secondhand;
} car_spec;
```

Структура записи для таблицы ключей

```
// element of key-table
typedef struct
{
    size_t src_idx;
    unsigned long long price;
} key_price;
```

4. Тесты

Описание теста	Ввод	Вывод
Чтение из несуществующего файла	-	Error: can't access file
Запись в несуществующий файл	-	Error: can't access file
Чтение из файла с неверной структурой	-	Error: invalid file data structure
Добавление в таблицу, при условии, что таблицы достигла	[Корректные	

Описание теста	Ввод	Вывод
максимального количества записей	данные]	Error: table overflow
Ввод некорректной опции выбора	-23	Невалидное значение. Попробуйте еще раз
Ввод некорректного диапазона цен		[!] High price must be higher than low
Вывод таблицы на экран	3	[Таблица автомобилей]
Вывод таблицы ключей	5	[Таблицы ключей]
Добавление строки в таблицу	[Корректные данные]	[+] Successfully added 1 record
Удаление строки из таблицы	[Корректные данные]	[+] Successfully removed 1 record

5. Основные функции

`return_code` - тип, описывающий код возврата

`car_t` - тип, описывающий запись в таблице (См. структуры данных)

```
typedef enum
{
    OK,
    DIALOG_EXIT,

    ERR_CARS_LIMIT_REACHED,

    ERR_DIALOG_INPUT_OVERFLOW,
    ERR_DIALOG_INVALID_INPUT,
    ERR_IO,

} return_code;
```

Функция добавления записи

```
return_code process_add_car(car_t *cars, size_t *n);
```

Функция удаления записи

```
return_code process_remove_car(car_t *cars, size_t *n);
```

Функция вывода таблицы ключей

```
void process_show_key_table(const car_t *cars, const size_t n);
```

Функция вывода отсортированной таблицы ключей

```
void process_show_sorted_key_table(const car_t *cars, const size_t n);
```

Функция вывода таблицы автомобилей

```
void process_show_cars_table(const car_t *cars, const size_t n);
```

Функция вывода отсортированной таблицы автомобилей

```
void process_show_sorted_cars_table(const car_t *cars, const size_t n);
```

Функция получения и вывода статистики

```
void process_show_stat(car_t *cars, const size_t n);
```

Функция выборки из таблицы по заданию

```
void process_super_filter(const car_t *cars, const size_t n);
```

Функция загрузки таблицы из файла

```
return_code process_load_cars(car_t *cars, size_t *n);
```

Функция сохранения таблицы в файл

```
return_code process_save_cars(const car_t *cars, const size_t n);
```

5. Оценка эффективности

Для каждой контрольной точки мною было проведено 10 замеров, а затем взят средний результат. В рамках одной итерации перед каждой очередной сортировки массив перемешивался при помощи одного и того же сида.

50 элементов

	Time (nanoseconds)	Memory (bytes)
Table (quick sort)	4600	6800
Key-Table (quick sort)	3300 (+28.26%)	7600 (-11.76%)
Table (selection sort)	43900	6800
Key-Table (selection sort)	12900 (+70.62%)	7600 (-11.76%)

100 элементов

	Time (nanoseconds)	Memory (bytes)

	Time (nanoseconds)	Memory (bytes)
Table (quick sort)	8900	13600
Key-Table (quick sort)	5100 (+42.70%)	15200 (-11.76%)
Table (selection sort)	88900	13600
Key-Table (selection sort)	33700 (+62.09%)	15200 (-11.76%)

500 элементов

	Time (nanoseconds)	Memory (bytes)
Table (quick sort)	67000	68000
Key-Table (quick sort)	47400 (+29.25%)	76000 (-11.76%)
Table (selection sort)	918300	68000
Key-Table (selection sort)	596600 (+35.03%)	76000 (-11.76%)

1000 элементов

	Time (nanoseconds)	Memory (bytes)
Table (quick sort)	126100	136000
Key-Table (quick sort)	90600 (+28.15%)	152000 (-11.76%)
Table (selection sort)	2458600	136000
Key-Table (selection sort)	1853800 (+24.60%)	152000 (-11.76%)

2500 элементов

	Time (nanoseconds)	Memory (bytes)
Table (quick sort)	276700	340000
Key-Table (quick sort)	202100 (+26.96%)	380000 (-11.76%)
Table (selection sort)	10586900	340000
Key-Table (selection sort)	8762700 (+17.23%)	380000 (-11.76%)

5000 элементов

	Time (nanoseconds)	Memory (bytes)
Table (quick sort)	597800	680000
Key-Table (quick sort)	413200 (+30.88%)	760000 (-11.76%)
Table (selection sort)	41876400	680000
Key-Table (selection sort)	32478100 (+22.44%)	760000 (-11.76%)

Из данных можно заметить, что применение таблицы ключей в среднем дает прирост по скорости в **~31%** для быстрой сортировки, и **~39%** для сортировки выбором при дополнительных затратах памяти в **~12%**.

> Но при этом крайне важно отметить, что такой прирост в первую очередь обусловлен достаточно большим объемом памяти, которую занимает каждая запись в таблице

6. Контрольные вопросы

1. Как выделяется память под вариантную часть записи?

В языке С под вариантную часть (объединение) выделяется столько памяти, сколько требует наибольший из возможных вариантов (типов) объединения

2. Что будет, если в вариантную часть ввести данные, несоответствующие описанным?

В языке С если вписать в вариантную часть несоответствующие данные, то при прочтении они будут неявно преобразованы согласно описанию, что может привести к искажению. Поэтому программист должен следить за тем, что хранится в объединении

3. Кто должен следить за правильностью выполнения операций с вариантной частью записи?

Программист

4. Что представляет собой таблица ключей, зачем она нужна?

Таблица ключей представляет собой таблицу записей, которые состоят из значения поля (ключа) и индекса (Id) в исходной таблице. Таблица ключей используется для ускорения операций, над большими таблицами, когда работа происходит, с конкретными полями. Преимуществом использования таблицы ключей является то, что для того, чтобы получить доступ к интересующему полю записи не приходится считывать ее целиком. Минусом является дополнительные затраты памяти на таблицу ключей

5. В каких случаях эффективнее обрабатывать данные в самой таблице, а когда – использовать таблицу ключей?

Использование таблицы ключей для оптимизации целесообразно, когда мы работаем с таблицами, записи в которых имеют большое количество полей или имеют поля занимающие много памяти. Если же у нас мало записей или чтение записи не требует много ресурсов, то нет смысла затрачивать дополнительные силы на реализацию таблиц ключей.

6. Какие способы сортировки предпочтительнее для обработки таблиц и почему?

Так как записи таблиц зачастую занимают намного больше памяти, чем стандартные несоставные типы, для сортировки таблиц стоит использовать алгоритмы выполняющие как можно меньше сравнений и прочтений. Отлично подходят быстрая сортировка

7. Выводы

Использование таблицы ключей может давать приросты в скорости обработки таблиц данных, при дополнительных затратах по памяти: В рамках моей работы это **+31%** для оптимизированной сортировки и **+39%** для неоптимизированной, при дополнительных затратах памяти в **12%**. Наиболее эффективно этот метод показывает себя при работе с большими таблицами, которые занимают много памяти, когда дополнительные затраты памяти будут сравнительно малы. Однако, при этом не стоит использовать таблицы ключей при работе с маленькими таблицами, так как это повлечет дополнительные затраты памяти и усложнит структуру программы, дав взамен относительно небольшой прирост скорости.