Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 ПО ДИСЦИПЛИНЕ ТИПЫ И СТРУКТУРЫ ДАННЫХ

Студент **Простев Тимофей Александрович**

Группа **ИУ7-33Б**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____ **Простев Т. А.**

Преподаватель _____ **Никульшина Т. А.**

Преподаватель _____ **Барышникова М. Ю.**

Оценка _____

2022 г.

1. Описание условия задачи.

Создать таблицу записей, упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ – любое невариантное поле, используя: а) саму таблицу, б) массив ключей. Запись – машина, содержит: марка автомобиля, страна-производитель, цена, цвет, состояние (новая – гарантия (в годах), старая – год выпуска, пробег, количество ремонтов, количество собственников), иномарка – поддерживается обслуживание или нет. Вывести цены не новых машин указанной марки с одним предыдущим собственником, отсутствием ремонта в указанном диапазоне цен. Найти среди них иномарки с поддержанием обслуживания.

2. Техническое задание.

Исходные данные и правила их ввода.

- Пункт меню: числа от 1 до 13.
- Имя файла для чтения:
  - Не может превышать 1023 символа.
  - Может быть как абсолютным, так и относительным.
  - Файл должен существовать.
  - Файл должен быть доступен для чтения.
  - Содержимое файла должно подходить по формату записей.
- Имя файла для записи:
  - Не может превышать 1023 символа.
  - Может быть как абсолютным, так и относительным.
  - Файл не обязан существовать.
  - Файл должен быть доступен для записи или создания.
- Поиск записей:
  - Марка автомобиля не должна превышать 15 символов.
  - Нижняя граница стоимости не может быть отрицательной.
  - Верхняя граница стоимости не может быть отрицательной.
  - Верхняя граница не может быть меньше нижней границы.
- Добавление записи:
  - Марка автомобиля не должна превышать 15 символов.
  - Цвет не должен превышать 15 символов.
  - Стоимость не может быть отрицательной.
  - Название страны производства не должно превышать 31 символ.
  - Срок гарантии не может быть отрицательным.
  - Год выпуска не может быть отрицательным.
  - Пробег не может быть отрицательным.
  - Количество ремонтов не может быть отрицательным.
  - Количество владельцев не может быть отрицательным.
- Результат:
  - Таблица автомобилей.
  - Таблица ключей.
  - Таблица автомобилей, выведенная по таблице ключей.
  - Таблица автомобилей, подошедших под условие.
  - Таблица информации о сравнении сортировок.

Способ обращения к программе:

Запуск исполняемого файла app.exe в консоли. Выполнить команду ./app.exe.

Возможные аварийные ситуации:

- Попытка произвести действия над таблицей, которая не была загружена из файла.
- Попытка произвести действия над несформированной таблицей ключей.
- Ошибка при открытии файла на чтение/запись.
- Ошибка при чтении записей из файла.
- Ввод некорректных данных.
- Ошибки при выполнении исследования сортировок.
    - Отсутствие нужного файла с записями.
    - Ошибка при чтении данных из файла.
    - Недостаточное количество записей для исследования.

В аварийной ситуации программа выводит на экран сообщение об ошибке с указанием ошибки, далее, программа завершается с определённым кодом ошибки.

3. Описание внутренних структур данных.

Структура автомобиля:

```
typedef struct {
    char brand[CAR_BRAND_LENGTH + 1];
    char color[CAR_COLOR_LENGTH + 1];
    unsigned int. price;
    char country[CAR_COUNTRY_LENGTH + 1];

    maintenance_t maintenance;
    state_t state

} car_t;
```

В структуре автомобиля содержится 6 полей:

- `brand` – марка автомобиля – строка, содержащая не более 15 символов (+1 конечный символ);
- `color` – цвет автомобиля – строка, содержащая не более 15 символов (+1 конечный символ);
- `price` – стоимость автомобиля – беззнаковое целое типа `unsigned int`;
- `country` – страна производства автомобиля – строка, содержащая не более 31 символа (+1 конечный символ);
- `maintenance` – запись, содержащая в себе информацию о техническом обслуживании;
- `state` – запись, содержащая в себе информацию о состоянии автомобиля.

Структура, содержащая в себе информацию о поддержке обслуживания иномарки.

```
typedef struct {
    int is_foreign;
    int is_available;
} maintenance_t;
```

В структуре содержится два поля:

- `is_foreign` - является ли автомобиль иномаркой;
- `is_available` - доступно ли обслуживание.

Структура, содержащая в себе информацию о состоянии автомобиля.

```
typedef struct {
    int type;
    union {
        new_car_state_t new_car_state;
        old_car_state_t old_car_state;
    } info;
} state_t;
```

В структуре содержится два поля:

- `type` - тип состояния (0 – старый автомобиль, 1 – новый автомобиль);
- `info` - вариативная часть (объединение):
    - `new_car_state` - информация о старом автомобиле,
    - `old_car_state` - информация о новом автомобиле.

Информация из объединения будет считываться в зависимости от указанного типа состояния.

Структура, содержащая в себе информацию о старом автомобиле:

```
typedef struct {
    unsigned int release_year;
    unsigned int mileage;
    unsigned int repairs_count;
    unsigned int owners_count;
} old_car_state_t;
```

В структуре содержится 4 поля:

- `release_year` - год выпуска автомобиля – беззнаковое целое типа `unsigned int`;
- `mileage` - пробег – беззнаковое целое типа unsigned int;
- `repairs_count` - количество ремонтов – беззнаковое целое типа `unsigned int`;
- `owners_count` - количество владельцев - беззнаковое целое типа `unsigned int`;

Структура, содержащая в себе информацию о новом автомобиле:

```
typedef struct {
    unsigned int warranty;
} new_car_state_t;
```

В структуре содержится 1 поле:

- `warranty` - срок действия гарантии - – беззнаковое целое типа `unsigned int`;

Структура таблицы записей:

```
typedef struct {
    size_t size;
    car_t *array;
}
car_array_t;
```

В структуре содержится 2 поля:

- `size` - размер динамического массива – беззнаковое целое число типа `size_t`;
- `array` - указатель на динамический массив записей типа `car_t`.

Структура таблицы ключей (ключ – стоимость автомобиля):

```
typedef struct {
    size_t size;
    price_key_t *keys;
} car_price_key_table_t;
```

В структуре содержится 2 поля:

- `size` - размер динамического массива – беззнаковое целое число типа `size_t`;
- `keys` - указатель на динамический массив ключей типа `price_key_t`.

Структура ключа:

```
typedef struct {
    size_t original_position;
    unsigned int price;
} price_key_t;
```

В структуре содержится 2 поля:

- `original_position` - позиция записи в таблице – беззнаковое целое число типа `size_t`.
- `price` - стоимость автомобиля - стоимость автомобиля – беззнаковое целое типа `unsigned int`.

Структура, хранящая в себе результаты исследования времени сортировок.

```
typedef struct {
    size_t size;

    double average_time_array_selection_sort;
    double average_time_array_merge_sort;

    double average_time_key_table_selection_sort;
    double average_time_key_table_merge_sort;
} research_t;
```

В структуре содержится 5 полей:

- size – размер исследуемого набора записей типа size_t;
- `average_time_array_selection_sort` – среднее время выполнения сортировки целой таблицы методом выбором. Тип - `double`;
- `average_time_array_merge_sort` - среднее время выполнения сортировки целой таблицы методом слияния. Тип – `double`;
- `average_time_key_table_selection_sort` – среднее время выполнения сортировки таблицы ключей методом выбором. Тип - `double`;
- `average_time_key_table_merge_sort` - среднее время выполнения сортировки таблицы ключей методом слияния. Тип - `double`.

Для работы со структурами используются следующие функции:

Структура `car_t`:

`int car_scan(car_t *, int);` - ввод автомобиля через консоль.

`int car_file_scan(car_t *, FILE *);-` чтение автомобиля из файла.

`void car_file_print(FILE *, car_t *);` - запись информации об автомобиле в файл.

`int car_cmp_price(const void *, const void *);` сравнение стоимости двух автомобилей.

Структура `car_array_t`:

`void car_array_create(car_array_t *);` - заполнение полей нулевыми значениями.

`int car_array_file_scan(car_array_t *, FILE *);` - считывание таблицы записей из файла.

`int car_array_push_back(car_array_t *, car_t *);` - добавление записи в конец массива.

`int car_array_remove(car_array_t *, size_t);` - удаление записи из таблицы по индексу.

`int car_array_table_print(car_array_t *);` - вывод форматированной таблицы записей на экран.

`int car_array_file_print(FILE *, car_array_t *);` - вывод таблицы записей в файл.

`void car_array_destructor(car_array_t *);` - освобождение памяти, выделенной под массив записей, задание полей начальными значениями.

`void car_array_selection_sort_by_price(car_array_t *);` - сортировка таблицы записей по стоимости автомобиля методом выбора.

`void car_array_merge_sort_by_price(car_array_t *);` - сортировка таблицы записей по стоимости автомобиля методом слияния.

`int car_array_table_print_filtered_by_task(car_array_t *cars, char *brand, unsigned int floor, unsigned int ceil);` - поиск записей по заданию. Вывод найденных записей как форматированная таблицы на экран.

Структура `car_price_key_table_t`:

`void car_price_key_table_create(car_price_key_table_t *);` - заполнение полей нулевыми значениями.

`int car_price_key_table_init(car_price_key_table_t *, car_array_t *);` - создание таблицы ключей по существующей таблице записей.

`int car_price_key_table_print(car_price_key_table_t *);` - вывод таблицы ключей.

`int car_array_print_with_key_table(car_array_t *, car_price_key_table_t *);` - вывод таблицы записей по таблице ключей.

`void car_price_key_table_destructor(car_price_key_table_t *);` - освобождение памяти, выделенной под массив ключей, задание полей начальными значениями.

`void car_price_key_table_selection_sort(car_price_key_table_t *);` - сортировка таблицы ключей методом выбора.

`void car_price_key_table_merge_sort(car_price_key_table_t *);` - сортировка таблицы ключей методом выбора.

4. Описание алгоритма.

При запуске программы пользователю выводится сообщение с пунктами меню:

Возможные действия:

1 - Загрузить таблицу из файла.

2 - Выгрузить таблицу в файл.

3 - Сформировать таблицу ключей (ключ - стоимость).

4 - Вывести таблицу.

5 - Вывести таблицу ключей.

6 - Вывести таблицу по таблице ключей.

7 - Произвести поиск.

8 - Отсортировать таблицу по стоимости.

9 - Отсортировать таблицу ключей.

10 - Добавить запись в таблицу.

11 - Удалить запись из таблицы.

12 - Произвести исследование времени сортировок.

13 - Выход из программы.

Если таблица записей не загружена в память, перед меню будет выведено сообщение.

Если таблица записей не сформирована, перед меню будет выведено сообщение.

Выбор действия производится путём ввода в консоль номера этого действия. При некорректном вводе номера действия программа выведет на экран сообщение об ошибке и вернётся к выбору. При выборе действия выхода из программы, программа завершается.

Для начала работы с таблицей нужно загрузить её из файла, или добавить в пустую таблицу запись. После создания таблицы, пользователь сможет вывести форматированную таблицу на экран, записать таблицу в

файл, сформировать таблицу ключей, отсортировать таблицу по стоимости автомобиля, произвести поиск, добавить запись в таблицу, удалить запись из таблицы.

После формирования таблицы ключей пользователю становятся доступны следующие действия: вывести таблицу ключей, вывести таблицу записей по таблице ключей, отсортировать таблицу ключей.

При выборе действия, которое в данный момент недоступно пользователю, программа выведет сообщение об ошибке и вернется к выбору номера действия.

Пользователю всегда доступен пункт 12.

5. Оценка эффективности.

При исследовании времени сортировки целой таблицы записей и таблицы ключей двумя методами сортировки, каждая сортировка запускается на размерах таблиц от 10’000 до 100’000 по 50 раз, после чего берется среднее значение времени её выполнения.

Таблица времен сортировок:

	Таблица записей		Таблица ключей	
Размер таблицы, тыс шт.	Сортировка выбором, тики	Сортировка слиянием, тики	Сортировка выбором, тики	Сортировка слиянием, тики
10	6.7e+08	6.3e+08	7.9e+06	6.3e+06
20	2.7e+09	2.5e+09	1.7e+07	1.3e+07
30	6.4e+09	5.7e+09	2.7e+07	2.1e+07
40	1.2e+10	1.0e+10	3.7e+07	2.8e+07
50	2.1e+10	1.6e+10	4.7e+07	3.6e+07
60	3.2e+10	2.3e+10	5.8e+07	4.4e+07
70	4.5e+10	3.1e+10	6.8e+07	5.2e+07
80	6.0e+10	4.0e+10	7.9e+07	6.0e+07
90	7.7e+10	5.1e+10	9.1e+07	6.7e+07
100	9.7e+10	6.3e+10	1.0e+08	7.5e+07

Таблица размеров:

Размер таблицы, тыс шт.	Таблица записей, байт	Таблица записей + Таблица ключей, байт
10	960000	1120000
20	1920000	2240000
30	2880000	3360000
40	3840000	4480000
50	4800000	5600000

Таблица сравнения эффективности:

Размер таблицы, тыс шт.	Выигрыш сортировки таблицы ключей по времени (сортировка выбором), %	Выигрыш сортировки таблицы ключей по времени (сортировка слиянием), %	Проигрыш по памяти сортировки с таблицей ключей, %
10	5	25	17
20	6	28	17
30	12	29	17
40	23	31	17
50	32	33	17
60	39	32	17
70	44	32	17
80	48	33	17
90	51	35	17
100	53	35	17

6. Вывод.

При присутствии вариативности в записях, объединения экономят память, так как позволяют хранить несколько полей разных типов в одной области.

Использование таблицы ключей при сокращает время сортировки за счет меньшего объема памяти, которую нужно обменивать или записывать; однако таблица ключей занимает дополнительную память, которая линейно зависит от размера таблицы записей.

Из-за дополнительной занимаемой памяти, сортировка при помощи таблицы ключей не имеет смысла при малом количестве записей, так как прирост в скорости незначителен. При большем количестве записей, сортировка таблицы ключей выигрывает в скорости в среднем в 1.4 раза, по сравнению с сортировкой целой таблицы, при этом проигрывая в памяти на 17 %

Использование более эффективного метода сортировки ускоряет процесс как для целой таблицы, так и для таблицы ключей.

7. Контрольные вопросы.

1. Как выделяется память под вариантную часть?

   Объединение содержит несколько полей разного типа, которые, в отличие от структур, занимают одну область памяти. Размер выделяемой под объединение области равен размеру максимального по длине поля.

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

   Неопределённое поведение.

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

   За правильностью выполнения таких операций отвечает только программист.

4. Что представляет собой таблица ключей, зачем она нужна?
   Таблица ключей – массив структур, поля которых – сам ключ и индекс элемента в таблице записей. Таблица ключей нужна для ускорения времени сортировки за счет потерь в памяти.

5. В каких случаях эффективнее обрабатывать данные в самой таблице, а когда - использовать таблицу ключей?
   Таблица ключей наиболее эффективна, когда запись сортируемой таблицы занимает большую область памяти и таблица хранит большое количество записей. Если же таблица ключей небольшого размера, или размер одной записи невелик, эффективнее сортировать саму таблицу.

6. Какие способы сортировок предпочтительнее для таблиц и почему?
   Для таблиц наиболее эффективно использовать способы сортировок, при которых происходит наименьшее количество обменов записей, так как запись может занимать большой объём памяти, из-за чего возрастает время, требуемое на обмен двух записей.
   Если при сортировке используется таблица ключей, эффективнее использовать наиболее выигрышные по скорости сортировки.