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 «ЗАПИСИ С ВАРИАНТАМИ, ОБРАБОТКА ТАБЛИЦ»

Студент Гаврилюк Владислав Анатольевич

Группа ИУ7 – 31Б

Преподаватель Барышникова Марина Юрьевна
Проверил

2023 г.

Оглавление

Описание условия задачи

Создать таблицу машин, содержащую не менее 40-ка записей (тип – запись с вариантами (объединениями)). Запись содержит марку автомобиля, страну-производитель, цену, цвет и состояние: новый – гарантия (в годах); нет - год выпуска, пробег, количество ремонтов, количество собственников; иномарка – поддерживается обслуживание или нет. Упорядочить данные в таблице по возрастанию ключей, двумя алгоритмами сортировки, где ключ – цена автомобиля, используя: а) саму таблицу, б) массив ключей. Реализовать добавление и удаление записей в ручном режиме. Осуществить поиск не новых машин указанной марки с одним предыдущим собственником, отсутствием ремонта в указанном диапазоне цен. Найти среди них иномарки с поддержанием обслуживания.

Описание технического задания

Входные данные:

в аргументах командной строки указывается путь к текстовому файлу, который содержит список машин, имеющихся в автомагазине, где каждое поле находится на отдельной строке.

Выходные данные:

В зависимости от задачи, программа либо добавляет данные в текстовый файл, либо выводит таблицу с данными в терминал (за исключением удаления), где таблица с данными — исходная таблица с автомобилями или таблица ключей.

Задачи программы:

1. Добавление нового автомобиля в базу данных.

2. Удаление автомобилей с указанной ценовой категорией из базы данных.

3. Сортировка таблицы ключей.

4. Сортировка исходной таблицу.

5. Анализи эффективности сортировки исходной таблицы и таблицы ключей с помощью quick_sort и selection_sort.

6. Поиск старых иномарок с 1 владельцем, без ремонта, с поддержанием обслуживания (марка и диапазон цен задаются пользователем).

7. Вывод исходной таблицы.

8. Вывод таблицы ключей.

Обращение к программе:

Запускается через терминал: в параметрах запуска указывается путь к файлу (базе данных). Далее по инструкции пользователь указывает команду и программа выполняет ее.

Аварийные ситуации:

1. Некорректные аргументы

2. Файл не существует / пустой

4. Неполная структура в файле.

6. Недостаточно места для добавления нового автомобиля.

7. Ввод некорректных данных из терминала.

8. Пустой результат поиска

Ошибочные ситуации сопровождаются поясняющим сообщением об ошибке.

Описание структур данных

Для хранения данных о ключах и автомобилях были реализованы следующие структуры:

```
typedef struct
{
    int index;
    int value;
} key_t;
```

Структура **key_t** содержит в себе пару ключ-значение, где ключом является поле **value** (цена автомобиля), а значением – поле **index** (индекс автомобиля с с ценой **value** в исходной таблице).

```
typedef struct
{
    int year;
    int mileage;
    int repairs;
    int owners;
} old_t;

typedef struct
{
    char brand[MAX_STR_LEN + 1];
    char country[MAX_STR_LEN + 1];
    char color[MAX_STR_LEN + 1];
    int price;
    bool service;
    bool condition;
    union
    {
        int guarantee;
        old_t old;
    };
```

} car_t;

Поля структуры **car_t**:

1. **char brand[MAX_STR_LEN + 1];** - марка автомобиля.

2. **char country[MAX_STR_LEN + 1];** - страна-производитель.

3. **char color[MAX_STR_LEN + 1];** - цвет автомобиля.

4. **int price;** - цена автомобиля.

5. **bool service;** - Обслуживание (поддерживается / не поддерживается)

6. **bool condition;** - Состояние (новый / старый)

7. Далее идет безымянное объединение, которое является вариантным полем и зависит от состояние автомобиля: если новый — то вариантное поле состоит только из гарантии, в противном случае будет состоять из года выпуска, пробега, количества ремонтов и количества предыдущих собственников. (в стандарте c11 поддерживаются безымянные объединения внутри структур и к полям объединения можно обращаться как к полям структуры). Параметры старых автомобилей хранятся в структуре **old_t**.

(MAX_STR_LEN = 15)

Описание алгоритмов сортировки

1. Сортировка выбором

Находится максимальный элемент и перемещается в конец массива. Указатель на конец массива уменьшается на единицу. Процедура повторяется, пока не останется обработанных элементов.

2. Быстрая сортировка

Устанавливается опорный элемент, элементы, которые меньше опорного, помещаются слева от него, которые больше – справа. Рекурсивно выполняем

те же операции для 2 подмассивов: от начала массива до опорного элемента и от опорного до конца.

Сравнение эффективности

Время сортировки (Замеры времени производились в микросекундах)

Количество записей	Исходная таблица		Таблица ключей	
	Quick sort	Selection sort	Quick sort	Selection sort
10	7	6	3	1
50	70	32	18	14
100	121	88	35	45
1000	930	1308	227	1231
2000	3434	4872	391	4638
3000	7443	10733	630	10090
4000	13297	20257	862	18597
5000	17358	34310	1065	28753
10000	121737	141706	2293	115629

Как можно заменить, сортировка исходной таблицы менее эффективная по сравнению с сортировкой таблицы ключей. Также стоит обратить внимание на то, что selection sort быстрее quick_sort при малых размерах массива данных. При больших размерах в скорости будет выигрывать уже quick_sort.

Мы получили необычные результаты замеров при 100 элементах: на исходной таблице quick_sort медленее selection_sort, но быстрее на таблице ключей. Скорее всего, это связано с тем, что количество обменов в быстрой сортировке больше, чем в сортировке выбором, поэтому на данном этапе, с учетом того, что обмены в исходной таблице более затратные, количество обменов сильнее сказывается на итоговом времени, чем асимтотическая сложность.

Затраты по памяти (Данные указаны в байтах)

Количество записей	Исходная таблица	Таблица ключей (с учетом исходной таблицы)
10	792	880
50	3600	4000
100	7200	8000
1000	72000	80000
2000	144000	160000
3000	216000	240000
4000	288000	320000
5000	360000	400000
10000	720000	800000

Сортировка данных с помощью таблицы ключей в среднем использует на 11.111112% больше памяти.

НАБОР ТЕСТОВ

№1	Название теста	Результат
1	Некорректные параметры	[ERR]: Некорректные параметры запуска.
2	Файл не существует	[ERR]: Ошибка при открытии файла.
3	Файл пуст	[ERR]: В файле нет данных.
4	В файле базы данных присутствуют некорректные данные	[ERR]: Ошибка при попытке чтения данных.

5	Слишком много элементов в файле	[ERR]: Слишком много объектов в файле.
6	Добавление нового авто. Корректные данные	Запись добавлена успешно.
7	Добавление нового авто. Некорректные данные.	[ERR]: Введены некорректные данные.
8	Добавление нового авто. База данных заполнена	[ERR]: База данных заполнена. Освободите место, чтобы добавить новый автомобиль.
9	Удаление. Корректные данные	Записи с указанными параметрами удалены успешно.
10	Удаление. Некорректные данные	[ERR]: Введены некорректные данные.
11	Удаление. Нет элементов с указанной ценой.	[ERR]: Нет автомобилей в указанной ценовой категории. Удаление невозможно.

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как выделяется память под вариантную часть записи?

Выделяется объем памяти, необходимый для хранения максимального по длине поля вариантной части.

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

Если объединение состоит из двух полей, где второе поле длинее первого, то при вводе в первое поле значения, длина которого превышает длину первого поля, то остаток данных запишется в часть второго поля. Если

длина вводимого объекта будет превышать и длину второго поля, то программа экстренно завершится с ошибкой (segmentation fault).

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

Тип данных в вариантной части при компиляции не проверяется, поэтому контроль за правильностью ее использования возлагается на программиста.

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

Таблица ключей — таблица пар ключ-значение, где ключом является какое-либо поле, а значением — индекс объекта с данным полем в исходной таблице. Таблица ключей позволяет быстрее обрабатывать большие массивы данных, т. к. пара ключ-значение является более легковесной структурой данных.

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

Таблицу ключей эффективнее использовать при больших объемах данных. Тогда прирост в скорости в процентном соотношении будет во много раз больше затрат памяти. При маленьких массивах данных использовать таблицу ключей не всегда целесообразно (не всегда будет большой прирост эффективности).

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

Сортировка таблицы ключей предпочтительнее при больших объемах данных (причина описана в вопросе №5). Если выбирать алгоритмы сортировки, то при больших размерах входных массивов лучше использовать quick_sort, а при малых размерах использовать selection_sort.

(На малых размерах сортировки с асимтотической сложностью O(n^2) эффективнее, чем сортировки с O(n * log(n)).

Вывод

Результатом данной лабораторной работы является готовая программа, позволяющая добавлять, удалять и сортировать данные, производить поиск по определенным критериям.

Мы экспериментально подтвердили, что при больших объемах данных эффективнее использовать таблицу ключей.

При малых объемах данных эффективнее использовать сортировку выбором, нежели быструю сортировку.