Министерство образования и науки 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 (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 «ЗАПИСИ С ВАРИАНТАМИ, ОБРАБОТКА ТАБЛИЦ»

Студент, группа **Ивахненко Д., ИУ7-36Б**

2020 г.

Описание условия задачи

Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами). Упорядочить данные в ней по возрастанию ключей двумя алгоритмами сортировки, где ключ – любое невариантное поле (по выбору программиста), используя:

- саму таблицу
- массив ключей

(возможность добавления и удаления записей в ручном режиме обязательна).
Осуществить поиск информации по варианту.

Ввести список машин, имеющихся в автомагазине, содержащий:

- марку автомобиля
- страну-производитель
- цену
- цвет
- состояние:
  - новый
    - гарантия (в годах);
  - нет
    - год выпуска
    - пробег
    - количество ремонтов
    - количество собственников.

Вывести цены не новых машин указанной марки с одним предыдущим собственником, отсутствием ремонта в указанном диапазоне цен.

Техническое задание

Входные данные:

1. **Файл с данными:** текстовый файл формата **CSV.** Каждая новая запись таблицы в обязательном порядке должна находиться на новой строке. В конце строки «;» не ставится.
2. **Целое число, представляющее собой номер команды:** целое число в диапазоне от **0** до **14**.
3. **Дополнения к таблице:**
   - непустое строковое или беззнаковое целочисленное поле, в зависимости от вводимой информации.

Выходные данные:

1. **Полученная таблица** (основная или таблица ключей) в отсортированном или неотсортированном виде (в зависимости от выполненной команды).
2. **Храктеристика сравнения** вариантов сортировки таблицы.
3. **Файл с данными:** текстовый файл формата **CSV.**

Функции программы:

Программа выполняет ряд функций, указанных при её первом запуске. Она позволяет:

1. Выгрузить таблицу из файла.
2. Добавить запись в таблицу.
3. Удалить запись из таблицы по марке автомобиля.
4. Отсортировать таблицу ключей сортировкой $O(n^2)$.
5. Отсортировать таблицу ключей сортировкой O(nlogn).
6. Отсортировать основную таблицу сортировкой $O(n^2)$.
7. Отсортировать основную таблицу сортировкой O(nlogn).
8. Синхронизировать таблицу ключей с основной таблицей
9. Вывести на экран основную таблицу.
10. Вывести таблицу ключей.
11. Вывести исходную таблицу, используя таблицу ключей.
12. Вывести сравнение времени (и памяти) сортировки таблицы сортировками со сложностями $O(n^2)$ и O(nlogn) и сравнение времени (и памяти) сортировки основной таблицы и сортировки с использованием таблицы ключей.
13. Выполнять поиск по основной таблице по вариантной части согласно условию задачи.
14. Сохранить таблицу в файл

Замечания:

- В качестве сортировки с асимптотической сложностью $O(n^2)$ используется сортировка вставками, с O(nlogn) – алгоритм быстрой сортировки.
- Сортировка производится по полю marque.
- Программа работает с одним файлом data.csv, который находится в директории ./data. Его наличие на момент попытки считывания данных обеспечивается программой. Корректность данных файла не проверяется.
- При сохранении таблицы в файл происходит перезапись файла
- Всякая строка корректна, за исключением пустой и переполнения (64 символа).
- Ведущие нули в целых числах недопустимы.

- Поиск производится независимо от регистра.

Обращение к программе:

Запускается из терминала. Бинарный файл **cars, запускающий программу** распологается в директории ./bin. ***Пример запуска: ./bin/cars***

Аварийные ситуации:

1. Некорректный ввод номера команды.
   На входе: строка, не удовлетворяющая представлению номера команды, указанного в ТЗ.
   На выходе: сообщение о некорректном вводе номера команды.
2. Превышение количества записей в конечной таблице.
   На входе: добавление новой записи при максимальном размере таблицы.
   На выходе: сообщение о переполнении таблицы.
3. Неверный ввод строкового поля.
   На входе: пустая / переполненная строка
   На выходе: сообщение о некорректном вводе строкового поля.
4. Неверный ввод целочисленного поля.
   На входе: пустой ввод / переполнение / ввод некорректного числа.
   На выходе: сообщение о некорректном вводе целочисленного поля.
5. Ввод недопустимого признака поля.
   На входе: целое число, отличающееся от обусловленных допустимых значений для поля.
   На выходе: сообщение о некорректном вводе

Структуры данных

Используемый именной тип данных **cars_t** представляет собой структуру, определяющую основную таблицу. Он содержит массив структур **car_t cars[MAX_NUMOF_CARS] а также беззнаковое** целочисленное поле **size_t amount**

```
#define MAX_NUMOF_CARS 1000

typedef struct
{
   car_t cars[MAX_NUMOF_CARS];
   size_t amount;
} cars_t;
```

Именной тип данных **keys_t** является представлением таблицы ключей и описывается как:

```
typedef struct
{
   short_car_info_t data[MAX_NUMOF_CARS];
   size_t amount;
} keys_t;
```

Соответствующие типы, описывающие массивы структур **car_t** и **short_car_info_t** описываются следующим образом:

```
#define LEN_MARQUE 64
#define LEN_COUNTRY 64
#define LEN_COLOR 64

typedef struct
{
   char marque[LEN_MARQUE ];
   char country[LEN_COUNTRY ];
   uint64_t price;
   char color[LEN_COLOR ];
   condition_t condition;
   condition_attributes_t cond_attr;
} car_t;

 typedef struct
{
   size_t index;
   char marque[LEN_MARQUE + 2];
} short_car_info_t;
```

Поля структур:

- **marque** – марка автомобиля (ключ, по которому производится сортировка)
- **country –** страна-производитель автомобиля
- **price —** цена автомобиля
- **color —** цвет автомобиля
- **condtition —** состояние автомобиля
- **cond_attr —** параметры автомобиля, зависящие от его состояния
- **index —** индекс данной записи в основной таблице **cars_t**

Состояние автомобиля описывается при помощи перечисляемого типа **condition_t** и объединения **condition_attributes_t,** что позволяет использовать одну и ту же область памяти для двух непересекающихся структур — характеристик нового и подержанного автомобиля:

```
typedef enum
{
   NEW,
   PRE_OWNED
} condition_t;

typedef union
{
   pre_owned_attr_t for_pre_owned;
   new_attr_t for_new;

} condition_attributes_t;
```

Поля структур:

- **NEW** и **PRE_OWNED –** определяют состояние автомобиля.
- **for_pre_owned и for_new –** определяют список характеристик которые имеет автомобиль в зависимости от состояния, соответственно нового и нет.

А каждый конкретный список характеристик также описывается при помощи именованных типов:

```
typedef struct
{
   uint8_t guarantee;
} new_attr_t;

typedef struct
{
   uint16_t product_year;
   uint32_t milage;
   uint16_t num_of_repairs;
   uint16_t num_of_owners;
} pre_owned_attr_t;
```

Поля структур:

- **guatantee –** гарантия, являющаяся характеристикой нового автомобиля.

- **product_year, milage, num_of_repairs, num_of_owners** — год производства, пробег, количество аварий и количество предыдущих владельцев соответственно.

Алгоритм

1. Пользователь вводит номер команды из меню.

```
/* Выбор пункта меню */
int choice_menu(cars_t *data, keys_t *key_data, int *choice);
```

2. Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять действия с таблицей.

Основные функции программы.

```
/* Ввести марку */
int input_marque(car_t *car);

/* Ввести страну */
int input_country(car_t *car);

/* Ввести цвет */
int input_color(car_t *car);

/* Ввести состояние */
int input_condition(car_t *car);

/* Ввести характеристики состояния */
int input_cond_attr(car_t *car);

/*Печать таблицы*/
void print_all(const cars_t *a);

/*Добавить автомобиль*/
int add(cars_t *a);

*Удалить автомобиль*/
void delete(cars_t *a, size_t index);
```

Тесты

	Тест	Пользовательский ввод	Результат
1	Некорректный ввод комманды	15	:::Ошибка::: :::Такого пункта нет:::
2	Добавление записи при максимальном размере таблицы	4. Добавить автомобиль в таблицу	:::Ошибка::: :::Произошло переполнение таблицы:::
3	Некорректный ввод строкового поля	«»	:::Ошибка::: :::Поле не может быть пустым:::
4	Некорректный ввод строкового поля	Строка, длиной более 60 символов	:::Ошибка::: :::Некорректный ввод строки:::
6	Некорректный ввод целочисленного поля	q	:::Ошибка::: :::Некорректный ввод числа:::
7	Ввод недопустимого признака поля	3 (при допустимых значениях от 0 до 1)	:::Ошибка::: :::Некорректный ввод диапазона::: :::Такого пункта нет:::
8	Некорректный ввод беззнакового целочисленного поля	-5	:::Ошибка::: :::Некорректный ввод числа:::
9	Корректное добавление новой записи	4. Добавить автомобиль в таблицу Lada, Russia, 50000 Red 0 50	::Выполнено::
10	Корректное удаление новой записи	5. Удалить автомобиль из таблицы (по названию) Lada	[Было удалено автомобилей: 1] :::Выполнено:::
11	Поиск	11. Поиск Lada 0 1000000	1. Марка: Lada Страна производства: Russia Цена: 20000 Цвет: Blue Состояние: подержанный

			Год производства: 2019 Пробег (км.): 30500 Количество ремонтов: 0 Количество предыдущих владельцев: 1 :::Выполнено:::

Оценка эффективности

Измерения эффективности сортировок будут производиться в единицах измерения – тактах процессорв. При записи результатов использовалось среднее количество тактов, полученное по результатам 10 измерений.

Время(такты процессора).

Количество записей	Сортировка вставками		Быстрая сортировка	
	Исходная таблица	Таблица ключей	Исходная таблица	Таблица ключей
50	15	10	13	7
100	41	27	33	17
250	234	155	76	47
500	983	608	160	95
750	2710	2038	228	150
1000	4226	2645	307	186

Память (байт).

Количество записей	Сортировка вставками		Быстрая сортировка	
	Исходная таблица	Таблица ключей	Исходная таблица	Таблица ключей*
50	11832	15360	12000	16000
100	23432	31280	24000	32000
250	58232	78080	60000	80000
500	116232	156080	120000	160000
750	174232	234080	180000	240000
1000	232232	312080	240000	320000

* при подсчете учитывались затраты на хранение исходной таблицы

Контрольные вопросы

1. Как выделяется память под вариантную часть записи?

Размер памяти, выделяемый под вариантную часть, равен максимальному по длине полю вариантной части. Эта память является общей для всех полей вариантной части записи.

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

Тип данных в вариантной части при компиляции не проверяется. Из-за того, что невозможно корректно прочитать данные, поведение будет неопределенным.

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

Контроль за правильностью выполнения операций с вариантной частью записи возлагается на программиста.

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

Дополнительный массив (структура), содержащий индекс элемента в исходной таблице и выбранный ключ. Она нужна для оптимизации сортировки.

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

В случае, если мы сортируем таблицу ключей, мы экономим время, так как перестановка записей в исходной таблице, которая может содержать большое количество полей, отсутствует. С другой стороны, для размещения таблицы ключей требуется дополнительная память. Кроме того, если в качестве ключа используется символьное поле записи, то для сортировки таблицы ключей необходимо дополнительно обрабатывать данное поле в цикле, следовательно, увеличивается время выполнения. Если исходная таблица содержит небольшое число полей, то выгоднее обрабатывать данные в самой таблице.

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

Если обработка данных производится в таблице, то необходимо использовать алгоритмы сортировки, требующие наименьшее количество операций перестановки. Если сортировка производится по таблице ключей, эффективнее использовать сортировки с наименьшей сложностью работы.

Вывод

Чем больше размер таблицы, тем эффективнее сортировка массива ключей, но даже на маленьких размерах таблицы, сортировка массива ключей происходит быстрее, чем сортировка самой таблицы (В моем случае, сортировка по ключам проиходит в среднем на **51%** быстрее при сортировке вставками и на **71%** быстрее при быстрой сортировке. Однако, для хранения массива ключей нужна дополнительная память. В моем случае, для сортировки таблицы ключей понадобилось примерно на **34%** больше памяти для сортировки вставками и на **33% -** для быстрой сортировки, чем для реализации сортировки основной матрицы. Стоит отметить, что использование массива ключей неэффективно при небольших размерах самой таблицы. В таком случае эффективнее просто отсортировать таблицу, так как разница во времени будет несущественна, а затраты на память сократятся.

	Сортировка вставками $O(n^2)$.	**Быстрая сортировка** O(nlogn).
Время	+ 51%	+71%
Память	+34%	+33%