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 **Рядинский К.В., ИУ7-33Б**

2020 г.

Описание условия задачи

Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами). Упорядочить данные в ней по возрастанию ключей, где ключ – любое невариантное поле (по выбору программиста), используя:
• саму таблицу
• массив ключей
(возможность добавления и удаления записей в ручном режиме обязательна).

Ввести список машин, имеющихся в автомагазине, содержащий: марку автомобиля, страну-производитель, цену, цвет и 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, содержащий искомую таблицу. Формат файла: каждое поле структуры в отдельной строке
2. *Команда* — строка, содержащая название команды
3. Дополнение к таблице: строковое или целочисленное поле, в зависимости от вводимой информации

Выходные данные

1. Таблица

2. Время сортировки

3. Цена автомобиля

Функции программы

1. Загрузка таблицы из файла
2. Вывод таблицы на экран
3. Добавление записи в таблицу

4. Удаление записи из таблицы
5. Сортировка таблицы по цене методом «Пузырек»
6. Сортировка таблицы по цене методом «Быстрая сортировка»
7. Вывод цен поддержанных автомобилей с одним бывшим хозяином, отсутствием ремонта, указанной марки и в определенном ценовом диапазоне.

Обращение к программе

Запускается из терминала

Аварийные ситуации

1. Неправильно введенный бренд автомобиля: длина больше 20 символов.
2. Неправильно введенная страна производства: длина больше 20 символов.
3. Неправильно введенная цена: отрицательная или вовсе не целое число.
4. Неправильно введеный цвет автомобиля: длина более 10 символов
5. Неправильный ответ на вопрос «да», «нет»
6. Неправильно введенная гарантия: нецелое число или отрицательное
7. Неправильно введенный год производства: нецелое число или отрицательное
8. Неправильно введенный пробег автомобиля: нецелое число или отрицательное
9. Неправильно введенное количество починок автомобиля: нецелое число или отрицательное
10. Неправильно введенное количество бывших хозяев: нецелое число или отрицательне
11. Неправильно введенный номер записи при удалении: число, выходящее за пределы таблицы или вовсе не число
12. Неправильное имя файла: длина более 10 символов
13. Несуществующий файл
14. Неправильный файл: файл не соответствует правилам, описанным в пунке «Входные данные»
15. Переполнение таблицы: длина более 500 записей

Структуры данных

Используется структура *static_vector_t*, представляющая собой тип данных, имеющий два поля: массив из структур типа *auto_t* и текущей длины (массив статический).

```
static_vector_t
{
        auto_t vec[VECTOR_LEN];
        int len;
}
```

auto_t представляет собой структуры со следующими полями: бренд, страна производства, цена, цвет, новая или старая машина, состояние — смесь, в которой хранится или гарантия (если машина новая), или год производства, пробег, кол-во ремонтов, кол-во бывших владельцев

```
Enum is_new
{
        NO,
        YES
}

Struct auto_t
{
        char brand[BRAND_LEN + 1];
        char country[COUNTRY_LEN + 1];
        int cost;
        char color[COLOR_LEN + 1];
        is_new is_new;
        condition_t condition;
}
BRAND_LEN = 20
COUNTRY_LEN = 20

Union condition_t
{
        new_auto_t new_auto;
        old_auto_t   old_auto;
}
```

```
Struct new_auto_t
{
	int warranty;
}



Struct old_auto_t
{
	int prod_year;
	int mileage;
	int repair_num
	int owner_num;
}
```

*Int ask_y_n(const char *msg)*

Спрашивает у пользователя вопрос msg, на который надо дать ответ y/n

Возвращает 1, если ответ да, иначе 0

Void print_help(void)

Печатает сообщение со всеми командами

Void print_hello(void)

Начальное сообщение при запуске программы

Void print_table(const static_vector_t table)

Печатает таблицу на экран

*Int add_record(static vector_t *table)*

Добавляет запись в таблицу. Возвращает код ошибки, если ввод был сделан неправильно

*Int delete_record(static_vector_t *table*)

Удаляет запись из таблицы. Возвращает код ошибки, если ввод был сделан неправильно

*Int write_int_file(static_vector_t *table)*

Записывает таблицу в файл. Возвращает код ошибки, если ввод был сделан неправильно

*Int read_from_file(static_vector_t *table)*

Считывает таблицу из файла. Возвращает код ошибки, если ввод был сделан неправильно

*Int init_sort(static_vector_t *table, int type, int is_key)*

Сортирует таблицу table сортировкой type по массиву ключей, если is_key > 0, иначе сортирует саму таблицу.

*Void bubsort(void *base, size_t __nel, size_t __width, cmp_fun cmp)*

Сортировка пузырьком

*Int cost_cmp(const void *a, const void *b)*

Вспомогательная функция для сортировки таблицы: сравнивает поля цен. Возвращает 1, если a > b; 0, если a == b; -1 иначе

*Int find_task(const static_vector_t *table)*

Поиск в таблице table по спецификации, указанной в техническом задании.

*Char *mystrerr(int errnum)*

Возвращает сообщение об ошибке, соответствующее номеру errnum

Алгоритм

1. Пользователь вводит команду из меню
2. Пока пользователь не введет команду *exit*, программа не заканчивается

Тесты

Тест	Ввод	Результат
Некорректная команда	Get	Неправильная команда. Печать сообщения с помощью
Несуществующий файл	Load test.txt	Несуществующий файл
Переполнение таблицы	Add	Переполнение таблицы
Некорректный ввод строкового поля	\n	Вывод соответствующего сообщения об ошибке
Некорректный ввод положительного целого числа	A	Вывод соответствующего сообщения об ошибке
Удаление несуществующей записи	Del	Номер записи был введен неправильно
Ввод записи в таблицу	Add	Готово
Удаление записи из таблицы	Del	Готово
Поиск по таблице	Task	Вывод цен
Сортировка	Tsort (T — b or q)	Сортировка таблице по цене и вывод времени

Оценка эффективности

Измерение эффективности сортировок будет производится в микросекундах.

Количество записей	**Пузырек**		**Быстрая сортировка**	
	Исходная таблица	Таблица ключей	Исходная таблица	Таблица ключей
50	260	48	10	7
100	1140	133	34	16
200	3570	634	45	28
500	17961	3330	106	96
1000	62550	12507	258	193
2000	231299	38150	482	345

Объем занимаемой памяти в байтах:

Кол-во записей	Таблица	Таблица ключей
50	5300	400
100	8004	800
200	16004	1600
500	40004	4000
1000	80004	8000
2000	160004	16000

Кол-во записей	Отношение времени сортировки массива ключей и таблицы для метода пузырька	Отношение времени сортировки массива ключей и таблицы для быстрой сортировки
50	5,4	1,4
100	8,6	2,1
200	5,6	1,6
500	5,4	1,1
1000	5,0	1,3
2000	6,1	1,4

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. Выбор данных из основной таблицы в порядке, определенном таблицей ключей, замедляет вывод. 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

Для сортировки больших таблиц уместно использовать дополнительный массив из ключей, чтобы ускорить сортировку. Этот метод позволит значительно повысить итоговое время сортировки, но потребует дополнительной памяти. Также уместно использовать подходящий и быстрый алгоритм сортировки — как, например, быстрая сортировка Хоара. Однако же, если таблица будет небольшая, то использованием дополнительным массивом можно пренебречь, так как не так существенна разница во времени — можно просмотреть таблицу выше, где есть данные по сортировке таблица размером 50.