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 ЛАБРАТОРНОЙ РАБОТЕ №2

Название предмета: Типы и структуры данных

Студент : Кашима Ахмед

Группа : ИУ7-33Б

лаборант : РЫБКИН Ю.А.

2022 г

.

Условие задачи:

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:

Приобрести навыки работы с типом данных «запись» (структура), содержащим вариантную часть – объединение (union); произвести сравнительный анализ реализации алгоритмов сортировки с данными, хранящимися в таблице; произвести сравнительный анализ реализации алгоритмов сортировки и поиск информации в таблице, при использовании записей с большим числом полей, и тех же алгоритмов при использовании таблицы ключей; оценить эффективность программы по времени и по используемому объему памяти при использовании различных структур и эффективность использования различных алгоритмов сортировок.

Входные данные

- Имя файла с записями автомобилей
- Команда меню (число от 0 до 12 включительно)
- Параметра добавляемой/удаляемой записи

Выходные данные

- Печать таблицы записей (ключей)
- Результат поиска (удаления) записей

- Сравнение эффективности по времени алгоритмов сортировки

Возможные аварийные ситуации

- Неизвестная команда меню
- Файл с записями пуст или не существует
- Некорректный ввод информации о новой записи
- Попытка обработки (вывода) таблицы записей, не импортировав её из исходного файла

Структура даннных

```
typedef enum
{
new = 1,
used
} condition_t;

// Not new (alreary driven)
typedef struct
{
int year;
int mileage;
int repairs;
int owners;
} driven_t;

// Vehicle condition
typedef union
{
int guarantee; // new
driven_t used; // already owned
} variable_t;

// Car info type
typedef struct
{
char brand[MAX_FIELD_LEN];
char country[MAX_FIELD_LEN];
int price;
```

```
char color[MAX_FIELD_LEN];
condition_t car_type;
variable_t add_info;
} car_t;

// Keys (car price)
typedef struct
{
int car_table_index;
int car_price;
} car_key_t;


// All in one data
typedef struct
{
car_t *main_table;
car_key_t *key_table;
int size;
} data_t;
```

Оценки эффективности

Замеры памяти (в байтах)

Количество записей	***Размер таблицы записей***	***Размер таблицы ключей***
10	*1760*	*80*
50	*8800*	*400*
100	*17600*	*800*
500	*88000*	*4000*
1000	*176000*	*8000*

Замеры времени выполнения сортировки

(в миллисекундах)*

Количество записей	***Таблица записей (bubble sort)***	***Таблица записей (qsort)***	***Таблица ключей (bubble sort)***	***Таблица ключей (qsort)***

10	*1*	*1*	*1*	*1*
50	*17*	*6*	*14*	*3*
100	*61*	*15*	*42*	*4*
500	*1732*	*77*	*1480*	*19*
1000	*7825*	*224*	*6033*	*43*

(*) Время сортировки: среднее время выполнения сортировки на 1000 повторениях

Сравнение алгоритмов сортировки

(сортировки таблицы записей)

Количество записей	***Соотношение превосходства скорости алгоритма qsort над bubble sort***
10	*в ~ 1.0 раз*
50	*в ~ 2.8 раза*
100	*в ~ 4.1 раз*
500	*в ~ 22.5 раза*
1000	*в ~ 35.0 раз*

Сравнение времени сортировки

таблицы ключей и таблицы ключей

Количество записей	***Соотношение превосходства скорости сортировки таблицы ключей над таблицей записей***
10	*в ~ 1.0 раз*
50	*в ~ 2.0 раза*
100	*в ~ 3.8 раз*
500	*в ~ 4.1 раза*
1000	*в ~ 5.2 раза*

Чтобы понять, насколько оправдано использование дополнительной памяти, измерим время сортировки исходной таблицы и таблицы ключей. При

этом используем разные алгоритмы сортировки. Для 100 записей результаты сравнения эффективности работы программы получились следующие:

	Bubble sort	Minmax	Quicksort
с ключами	19.03	31.25	17.22
без ключей	209.40	59.40	66.85

Из полученных результатов очевидно, что выигрыш во времени составил:
• для простой сортировки (выбором) в ~2 раза;
• для qsort в ~ 4 раза;
• для bubble sort в ~11 раз;

При этом в исходной таблице запись состоит из фиксированной (по длине) части: 3 строковых поля, 1 вещественное и 1 целое; и вариантной: 4 целочисленных (берём по максимуму). Даже без учёта выравнивания (!) получим, что 1 запись на 64-битной машине займёт: 30 * 3 + 1 * 8 + 1 * 4 + 4 * 8 = 134 байта, а 100 записей 13400 байт (с учётом выравнивания на конкретной машине размер занимаемой памяти составил 14400). В то же время массив ключей займёт 100 * (8 + 8) = 1600 байт. То есть мы увеличим память примерно в 1.1 раз.Из полученных цифр очевидно, что использование массива ключей целесообразно всегда, когда объём занимаемой памяти не критичен (критичен он может быть в случае программирования, например, под какие-то микроконтроллеры, где память сильно ограничена). В частности, при решении нашей задачи использование массива ключей целесообразно. Примечание: сказанное верно при большой размерности таблицы.

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). Для хранения

моего массива ключей 1000 записей понадобилось ~4.5% памяти занимаемой таблицей записей, но это обеспечили выигрыш во времени более чем в 2 раз. Следовательно сортировка таблицы через массив ключей эффективнее всегда.

Ответы на контрольные вопросы

1 *Как выделяется память под вариантную часть записи?*

Ответ: в Си, вариативная часть структуры реализована с помощью “union”. Память выделяется в одном “куске” памяти, имеющий размер, который способен вместить наибольшее поле из указанных.

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

Ответ: результат будет зависеть от ОС. Предсказать результат будет очень трудно(но один из возможных вариантов – приведение типов).

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

Ответ: ответственность за правильность проведения операций целиком и полностью лежит на программисте.

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

Ответ: таблица ключей представляет собой таблицу, в которой находится два столбца: номер ячейки в исходной таблице и значение поля исходной таблицы для этой ячейки.

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

Ответ: обрабатывать данные в самой таблице записей эффективнее (по памяти), когда время обработки не так важно, как задействованная память. Использование же таблицы ключей, эффективно, когда нужно сократить время обработки и не так важна дополнительная задействованная память (эффективнее по времени). Стоит отметить, что использование таблицы ключей будет неэффективно, когда исходная таблица состоит из малого числа полей. Тогда таблица ключей будет лишь занимать дополнительное место в памяти и не даст значительной выгоды во времени.

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

Ответ: для таблиц из большого количества записей предпочтительно использовать алгоритмы сортировки с относительно низкой вычислительной сложностью (в моей лабораторной работе – qsort со сложностью O(n*log(n)). Если же в таблица имеет небольшое количество записей, то лучше использовать простые алгоритмы сортировки, например, сортировку пузырьком или сортировку вставками.