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 Романов Семен Константинович

Группа ИУ7 – 35Б

Описание задачи

Создать таблицу, содержащую не менее 40 записей с вариантной частью. Произвести поиск информации по вариантному полю. Упорядочить таблицу, по возрастанию ключей (где ключ – любое невариантное поле по выбору программиста), используя:

1. Исходную таблицу;
2. Массив ключей, используя 2 разных алгоритма сортировки (простой, ускоренный).

Оценить эффективность этих алгоритмов (по времени и по используемому объему памяти) при различной реализации программы, то есть, в случаях 1) и 2). Обосновать выбор алгоритмов сортировки. Оценка эффективности должна быть относительной (в %).

Ввести список квартир, содержащий адрес, общую площадь, количество комнат, стоимость квадратного метра, первичное жилье или нет (первичное – с отделкой или без нее; вторичное – время постройки, количество предыдущих собственников, количество последних жильцов, были ли животные). Найти все вторичное 2-х комнатное жилье в указанном ценовом диапазоне без животных.

Техническое Задание

Входные данные:

1. ***Файлы с данными:*** несколько файлов, для каждого поля отдельный файл, записи в которых записываются каждый с новой строки

2. ***Целое число(номер команды):*** целое число от 0 до 9(см. Функции программы)
3. ***Данные таблицы:*** строковый или числовой тип, в зависимости от поля

Выходные данные:

1. Полученная таблица (основная или таблица ключей) в отсортированном или неотсортированном виде (в зависимости от выполненной команды).
2. Характеристика сравнения вариантов сортировки таблицы.

Способ обращения к программе:

Обращение происходит через консоль

Функции программы:

1. Добавление поля в конец таблицы
2. Удаление поля по его позиции в таблице
3. Отсортировать таблицу по таблице ключей и показать его
    a. Отсортировать таблицу обычным способом
    b. Использовать таблицу ключей, отсортированная на момент добавления полей в таблицу(сортировка при инициализации)
4. Отсортировать оригинальную таблицу(полная сортировка
5. Показать таблицу ключей
    a. Обычную
    b. Заранее отсортированную
6. Показать таблицу
7. Сравнить сортировку по ключам и полную сортировку
    a. Сравнить с обычной сортировкой

    b. Сравнить с сортировкой на момент инициализации
8. Сравнить два вида сортировки
9. Найти все вторичное 2-х комнатное жилье в указанном ценовом диапазоне без животных
0. Выход

Аварийные ситуации:

1. Некорректный ввод номера команды.
   На входе: число, большее чем 9 или меньшее, чем 0.
   На выходе: сообщение «Incorrect command»
2. Превышение количества записей в таблице.
   На входе: добавление новой записи при максимальном размере таблицы.
   На выходе: сообщение «Table overflow»
3. Неверный ввод строкового поля.
   На входе: строка, превышающая допустимый размер
   На выходе: сообщение «Invalid Input»
4. Ввод недопустимого признака поля.
   На входе: целое число, отличающееся от обусловленных допустимых значений для поля.
   На выходе: сообщение «Invalid Input»

Структуры данных:

1. Для хранения данных таблицы создается структура apartment, для хранения ключей создается структура apartment_key
2. Структура apartmnet:

```
typedef struct
{
    char address[ADDRESS_LEN];
    short square;
    short room_count;
    int square_price;
    short t;
    union type
    {
        struct secondary
        {
            short build_date;
            short owner_count;
            short last_owner_count;
            short was_animal;
        } sec;
        struct primary
        {
            short trim;
        } prim;
    } tp;
```

```
24.} apartment;
```

3. Структура apartment_key:

```
1. typedef struct
2. {
3.     int index_orig;
4.     int square_price;
5. } apartment_key;
```

Обозначение полей:

Apartment

1. char address[ADDRESS_LEN] – для хранения адреса
2. short square – количество квадратных метров
3. short room_count – количество комнат
4. int square_price – цена за квадратный метр
5. short t – тип квартиры(0 – первичная, 1 – вторичная)
6. union type – является вариативной частью структуры, содержащая две других структуры, для первичного жилья – prim, для вторичного – sec

Secondary:

1. short build_date – дата постройки
2. short owner_count – количество собственников
3. short last_owner_count – количество последних жильцов
4. short was_animal – были ли животные(0 – нет, 1 –да)

Primary:

1. short trim – Присутствие отделки (0 – нет, 1 – да)

Apartment_key

1. int index_orig – позиция поля в таблице
2. int square_price – цена за квадратный метр

Алгоритм:

1. На вход подается команда от 0 до 9
2. Пользователь выполняет действия с таблицей, пока не введет 0 (Выход)

Тесты

N	Test	Input	Result
1	Некорректная комманда	12	“Incorrect command”
2	Некорректная комманда	abc	“Incorrect command”
3	Превышение размера таблицы	1	“Table overflow”
4	Некорректный ввод строки	Abc…abc(101 символ)	“Invalid Input”
5	Некорректный ввод площади	5555555(переполнение типа)	“Invalid Input”
6	Некорректный ввод площади	-10	“Invalid Input”
7	Некорректный ввод логического	5	“Invalid Input”

	типа(для типа)		

Оценка Эффективности:

Для измерения времени будет использоваться библиотека time.h функцией clock() для того, чтобы зафиксировать начала и конец работы функции сортировки (все измерения будет измеряться в clock ticks)

Время сортировки:

N	Qsort	Bubble sort	Table bubble sort	Qsort Table sort
50	7	35	84	15
100	15	112	270	30
200	33	449	1152	61
300	73	970	2592	125

Занимаемая память(в байтах):

N	Занимаемый объем ключей от таблицы(в %)	Table	Table + Key
50	~4%	6160	6448
100	~4%	12160	12848
200	~4%	24160	25648
300	~5%	36160	38448

N	Рост скорости по сравнении массива ключей	Рост скорости по сравнении массива ключей

	и таблицы(bubble)	и таблицы(qsort)
50	В 2.4 раза	В 2 раза
100	В 2.4 раза	В 2 раза
200	В 2.6 раза	В 2 раза
300	В 2.7 раза	В 2 раза

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. 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?***

Для таблиц из большого количества записей предпочтительней использовать такие сортировки как qsort, MergeSort и т.д. со средним временем обработки равным O(n * log(n)). Данные методы сортировок не задействуют значительной памяти, но дают существенный выигрыш по времени по сравнению с тем же Bubble sort.

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. Тем не менее, для хранения массива ключей нужна дополнительная память. Для меня, на самом деле, понадобилось

относительно мало памяти, т. к. изначально в структуре таблицы выделено большой объем памяти под хранение строчного типа. При этом стоит отметить, что использование массива ключей неэффективно при небольших размерах самой таблицы, т.к. в таком случае эффективнее просто отсортировать таблицу, так как разница во времени несущественна. Также заметна динамика падения прироста скорости сортировки по ключам: у более простой сортировки скорость падения выше, чем у более сложной.