Лабораторная работа №2. "Таблицы"

Студент Ларин Владимир - ИУ7-34Б

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 (в км)
        - количество ремонтов
        - количество собственников.

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. **Целые числа, представляющее собой команды пользовотеля** .

Выходные данные:

1. **Полученная таблица** (основная или таблица ключей) в отсортированном или неотсортированном виде (в зависимости от выполненной команды).
2. **Результаты 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. Вывод данных из оперативной памяти
    1. Вывести автомобили
    2. Вывести автомобили (с помощью таблицы ключей)
    3. Вывести таблицу ключей
2. Редактирование таблицы
    1. Добавить автомобиль в таблицу
    2. Удалить автомобиль из таблицы (по бренду)
3. Сортировка
    1. Быстрая сортировка (основная таблица)
    2. Быстрая сортировка (таблица ключей)
    3. ортировка выбором (основная таблица)
    4. Сортировка выбором (таблица ключей)
4. Синхронизировать таблицу ключей с основной таблицей
5. Поиск по условию
6. Работа с файловой системой
    1. Считать таблицу из файла
    2. Сохранить таблицу в файл
7. Оценка сортировок
8. Перемешать таблицы

Замечания:

- Сортировка производится по полю Бренд (`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
- Ведущие нули в целых числах недопустимы.
- Поиск производится независимо от регистра.

Обращение к программе:

Запускается из терминала.

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

Для хранения таблицы автомобилей используется структура.

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

typedef struct
{
    char marque[LEN_MARQUE]; // Бренд автомобиля
    char country[LEN_COUNTRY]; // Страна производства
    uint64_t price; // Стоимость автомобиля
    char color[LEN_COLOR]; // Цвет автомобиля
    enum
    {
        NEW, // Новое
        PRE_OWNED // Поддержаное
    } condition; // Состояние автомобиля
    union
    {
        pre_owned_attr_t for_pre_owned;
        new_attr_t for_new;

    } cond_attr; // Смесь
} car_t;
```

Для разных состояний автомобиля используются типы `pre_owned_attr_t` и `new_attr_t`

```
typedef struct
{
    uint8_t guarantee; // Гарантия в годах

} new_attr_t;

typedef struct
{
    uint16_t product_year; // Год выпуска
    uint32_t milage; // Пробег в киллометрах
```

```
    uint16_t num_of_repairs; // Количество ремонтов
    uint16_t num_of_owners; // Количество владельцев

} pre_owned_attr_t;
```

Для упрощения взаимодействия с таблицей автомобилей создал структуру.

```
typedef struct
{
    car_t cars[MAX_NUMOF_CARS]; // Массив автомобилей
    size_t amount; // Количество автомобилей
} cars_t;
```

Чтобы хранить ключ индексирования (элемент доп. таблицу) создал структуру.

```
typedef struct
{
    size_t index; // Номер в начальной таблице
    char marque[LEN_MARQUE]; // Бренд автомобиля

} short_car_info_t;
```

Для хранения доп. таблицы использовал структуру.

```
typedef struct
{
    short_car_info_t data[MAX_NUMOF_CARS]; // Ключи доп. таблицы
    size_t amount; // Количество автомобилей
} keys_t;
```

Алгоритм

1. Пользователь вводит номер команды из меню.
2. Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять действия с таблицей.

Тесты

Меню:

```
Доступные команды:
1)  Вывести автомобили
2)  Вывести автомобили (с помощью таблицы ключей)
3)  Вывести таблицу ключей
```

```
4)  Добавить автомобиль в таблицу
5)  Удалить автомобиль из таблицы (по бренду)

6)  Быстрая сортировка (основная таблица)
7)  Быстрая сортировка (таблица ключей)
8)  Сортировка выбором (основная таблица)
9)  Сортировка выбором (таблица ключей)

10) Синхронизировать таблицу ключей с основной таблицей

11) Поиск по условию

12) Считать таблицу из файла
13) Сохранить таблицу в файл

14) Оценка сортировок

15) Перемешать таблицы

0)  Выход
```

Описание теста	**Ввод**	**Результат**
Напечатать таблицу	12 enter enter 1	Вывод таблицы
Напечатать таблицу ключей	12 enter enter 2	Вывод таблицы
Отсортировать основной массив	12 enter enter 6 enter 1	Вывод отсортированной таблицы
Отсортировать доп массив	12 enter enter 7 enter 2	Вывод отсортированной таблицы
Добавить автомобиль	Lada enter Russia 900000 enter black enter 0 enter 5 enter 1	Вывод таблицы с новым элементом
Добавить автомобиль	Lada enter Russia 900000 enter black enter 1 enter 2010 enter 100000 enter 0 enter 1 enter 1	
Удалить автомобиль	5 enter Lada enter enter 1	Вывод таблицы без удаленных элементов
Оценить сортировки	12 enter enter 14 enter	Вывод результатов тестирования

Описание теста	Ввод	Результат
Ввод некорректного пункта меню	16 enter	Вывод сообщения об ошибке
Ввод некорректного пункта меню	-1 enter	Вывод сообщения об ошибке

Оценка эффективности

Измерения эффективности сортировок будут производиться в единицах измерения – тактах процессорв. При записи результатов использовалось среднее количество тактов, полученное по результатам 1000 измерений.

Быстрая сортировка (stdlib `qsort`)

Количество элементов	Время (`clock_t`) (осн)	Память (байт) (осн)	Время (`clock_t`) (доп)	Память (байт) (доп + осн)
2000	486	456000	357	616000
1500	365	342000	261	462000
1000	246	228000	169	308000
500	118	114000	79	154000
250	55	57000	31	77000
100	22	22800	11	30800
50	11	11400	5	15400
25	6	5700	2	7700
10	4	2280	1	3080

Сортировка выбором

Количество элементов	Время (`clock_t`) (осн)	Память (байт) (осн)	Время (`clock_t`) (доп)	Память (байт) (доп + осн)
2000	25942	440220	24918	600080
1500	14614	330220	14452	450080
1000	6409	220220	6292	300080
500	1600	110220	1548	150080
250	408	55220	386	75080
100	67	22200	60	30080

Количество элементов	Время (`clock_t`) (осн)	Память (байт) (осн)	Время (`clock_t`) (доп)	Память (байт) (доп + осн)
50	21	11200	16	15080
25	7	5720	4	7580
10	2	2420	1	3080

Делаем ввывод,что быстрая сортировка эффеективнее, так как ее сложность $O(n\ln(n))$ меньше сложности сортировки выбором $O(n^2)$, но при этом выделяется больше памяти. (память под стек вызова, при делении массива на два)

Контрольные вопросы

1. Как выделяется память под вариантную часть записи?
    - Выделяется столько байт, сколько нужно для хранения максимальной вариантивной части. (без учета выравнивания)
2. Что будет, если в вариантную часть ввести данные, не соответсвтующие описанным?
    - Будет неопределенное поведение.
3. Кто должен следить за правильностью выполнения операций с вариантной частью записи?
    - Программист должен следить за правильностью выполнения операций с вариантной частью записи
4. Что представляет собой таблица ключей, зачем она нужна?
    - Таблица ключей содержит индекс элемента в исходной таблице и ключ, по которому производится поиск/ сортировка. Данная таблица позволяет сократить кол-во операций работы с памятью, так как строка таблицы ключей меньше исходной строки.
5. В каких случаях эффективнее обрабатывать данные в самой таблице, а когда – использовать таблицу ключей?
    - При нехватке памяти стоит не использовать таблицу ключей, в остальных случаях для оптимизации скорости сортировок / поиска строк (структура значительно больше ключа, по которому ведется сортировка) стоит применять таблицы ключей
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

Мы видем, что сортировка выбором медленне быстрой сортировки примерно в 5.5 раз. Но при этом выделяется больше памяти примерно на $3%$. Следовательно быструю сортировку эффективнее использовать, почти во всех случаях

Также заметим, что при сортировке доп таблице мы выигрывает в скорости работы примерно $36%$, но выделяем больше памяти на $35%$

Чем больше размер таблицы, тем эффективнее сортировка массива ключей, но даже на маленьких размерах таблицы, сортировка массива ключей происходит быстрее, чем сортировка самой

таблицы/ Однако, для хранения массива ключей нужна дополнительная память.

Стоит отметить, что использование массива ключей неэффективно при небольших размерах самой таблицы.