Введение

Условие 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: а) исходную таблицу; б) 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 а) и б). Обосновать выбор алгоритмов сортировки.

Техническое задание

Входные данные: текстовый файл, номер команды из меню

При этом:

- Максимальная длина строки составляет 15 символов
- Поле «цена» вводится в рублях
- Поле «пробег» в вариантной части вводится в километрах
- Исследуемый файл не изменяется в результате работы программы
- Каждое поле записывается на отдельной строке

Выходные данные: таблица, полученная в соответствии с выбранным пунктом меню.

Задачи, реализуемые программой

Программа считывает данные о машинах, реализует поиск и удаление в таблице значений по заданному ключу; сортирует таблицу, используя либо таблицу ключей и последующее восстановление исходной таблицы, либо только исходную таблицу, выводит результат, соответствующий пункту меню, на экран.

Способ обращения к программе

Запуск из терминала командой

./app.exe

Пользователь взаимодействует с программой посредством меню со следующими функциями:

0. выход
1. вывод исходной таблицы
2. добавление новой записи в конец таблицы
3. удаление записей с заданными полями
4. вывод цен не новых машин указанной марки с одним предыдущим собственником, отсутствием ремонта в указанном диапазоне цен
5. вывод упорядоченной исходной таблицы
6. вывод упорядоченной таблицы ключей (цены автомобиля)
7. вывод упорядоченной исходной таблицы с использованием таблицы ключей (цены автомобиля)
8. сравнение эффективности использования разных алгоритмов сортировки и разных методов сортировки

При выборе каждого из пунктов меню программа подскажет пользователю, что нужно вводить, а в случае ошибки выдаст соответствующее сообщение.

Описание внутренних структур данных

Для хранения информации о машинах в программе используется следующая структураЖ

```
typedef struct
{
   char brand[MAX_LEN];
   char country[MAX_LEN];
   long price;
   char colour[MAX_LEN];
   int type;
   union
   {
      struct
      {
         int warranty;
      } is_new;
      struct
      {
         int year;
         int mileage;
         int repairs;
         long owners;
      } not_new;
   } state;
} car_t;
```

В которой:

- char brand[MAX_LEN] – марка машины
- char country[MAX_LEN] – страна производителя
- long price – цена машины (в рублях)
- char colour[MAX_LEN] – цвет машины
- int type – тип машины. Может принимать 2 значения: 0 – если машина новая, 1 – если машина подержана

Если машина новая:

- int warranty – гарантия на автомобиль (в годах)

Если машина подержанная:

- int year – год производства
- int mileage – пробег (в км)

- int repairs – количество ремонтов
- long owners – количество предыдущих владельцев

Дополнительная таблица ключей хранится в следующе структуре,

```
typedef struct
{
   int index;
   long price;
} key_price_t;
```

В которой

- int index – индекс записи
- long price – цена автомобиля

Описание алгоритма

Сортировка вставками ($O(n^2)$): на каждом шаге алгоритма мы берем один из элементов массива, находим позицию для вставки и вставляем. Стоит отметить что массив из 1-го элемента считается отсортированным.

Быстрая сортировка (O(nlogn)): рекурсивный метод сортировки, на каждом этапе рекурсии в массиве выбирается «опорный» элемент, который находится в середине массива. Элементы в массиве перераспределяются таким образом, что все элементы, меньше опорного, будут находиться в левой половине относительно опорного, а остальные - справа. Дальше сортировке подвергаются две полученные части по отдельности. База рекурсии – массив единичной длины.

Тесты

Действие	Результат
Выбор пункта 1	Вывод исходной таблицы
Выбор пункта 2 Volvo Russia 500000 Black 3	Добавление новой записи в конец таблицы
Выбор пункта 3 country USA	Удаление всех записей о машинах, произведенных в США
Выбор пункта 4	Вывод цен не новых машин указанной марки с одним предыдущим собственником, отсутствием ремонта в указанном диапазоне цен
Выбор пункта 5	Вывод упорядоченной исходной таблицы
Выбор пункта 6	Вывод упорядоченной таблицы ключей (цены автомобиля)
Выбор пункта 7	Вывод упорядоченной с использованием таблицы ключей таблицы
Выбор пункта 8	Вывод сравнения работы разных алгоритмов и методов сортировки

Аварийные ситуации

Действие	Результат
Выбор неверного пункта меню	Сообщение: «Такого пункта в меню нет. Пожалуйста, выберите существующий пункт»
Открытие файла, содержащего некорректные данные	Сообщение: «Неверные входные данные»
Открытие несуществующего файла	Сообщение: «Неверное имя файла»
Запись не числа в поле, которое должно содержать числовое значение	Сообщение: «Было введено не число»
Запись слишком длинной строки в поле	Сообщение: «Неверная длина строки»
Ввод отрицательного числа в числовое поле, которое не может содержать отрицательное значение, если придерживаться здравого смысла	Сообщение: «Число не может быть отрицательным»

Оценка эффективности

Замеры, выполненные программой:

Количество элементов	Медленная сортировка исходной таблицы		Медленная сортировка таблицы ключей		Быстрая сортировка исходной таблицы		Быстрая сортировка таблицы ключей	
	время, μs	память, b	время, μs	память, b	время, μs	память, b	время, μs	память, b
100	50	14400	40	15200	10	14400	0	15200
500	1370	72000	708	76000	115	72000	32	76000
1000	5576	144000	3121	152000	183	144000	41	152000
2000	22418	288000	11887	304000	404	288000	85	304000
3000	50872	432000	26165	456000	584	432000	129	456000
4000	88384	576000	46796	608000	761	576000	196	608000
5000	138974	720000	72973	760000	951	720000	188	760000
6000	200544	864000	105616	912000	1111	864000	254	912000
7000	273004	1008000	142651	1064000	1356	1008000	326	1064000
8000	355666	1152000	186545	1216000	1479	1152000	344	1216000
9000	451685	1296000	236815	1368000	1652	1296000	408	1368000
10000	558329	1440000	291586	1520000	1795	1440000	453	1520000

Во время каждого из замеров взято среднее время, т.к. получается весомая погрешность во время вычисления времени выполнения программы из-за того, что другие приложения тоже занимают систему.

Из таблицы видно, что при создании дополнительной таблицы затрачиваемая память увеличивается примерно на 5,6%.

Учитывая эту информацию можно подсчитать, во сколько раз использование таблицы ключей эффективнее использования исходной таблицы:

Количество записей	**Таблица ключей / исходная таблица (сортировка вставками)**	**Таблица ключей / исходная таблица (быстрая сортировка)**
100	~экв	~ экв
500	~2	~3.5
1000	~2	~4.4
2000	~2	~4.7
3000	~2	~4.5
4000	~2	~3.8
5000	~2	~5

6000	~2	~4.4
7000	~2	~4.2
8000	~2	~4.2
9000	~2	~4
10000	~2	~4

Вывод

Чем больше размер сортируемой таблицы, тем более выгодно использовать сортировку с использованием таблицы ключей. Однако для этого требуется выделять дополнительную память. Таблицы ключей выгоднее применять до тех пор, пока затраты на память дают существенное увеличение скорости сортировки. При этом следует учитывать, что если выбранное поле является строкой, то время сортировки таблицы ключей возрастет, т.к. обработка строк занимает существенно больше времени, чем, например, обработка целых чисел.

Контрольные вопросы

1.Как выделяется память под вариантную часть записи?

Выделяется один блок памяти, который будут разделять вариантные переменные. Одновременно занимать память, т.е. быть «активным» может лишь одно из доступных вариантных полей. Размер области памяти, выделяемый под вариантную часть, равен максимальному из размеров вариантных полей.

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

Неопределенное поведение

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

Программисту необходимо следить за правильностью хранения и обработки данных, содержащихся в вариантной части.

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

Таблица ключей - таблица, содержащая индекс элемента в исходной таблице и выбранный ключ, по которому происходит поиск или сортировка. Она нужна для оптимизации сортировки.

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

Если затрачиваемая память на выделение таблицы ключей значительно меньше по сравнению с выигрышем по времени работы программы, то эффективнее использовать таблицу ключей. Этот выигрыш достигается за счет того, что снижается количество перестановок в исходной таблице и программа производит перестановку всего лишь одного ключа (в отличие от целой записи в случае сортировки исходной таблицы).

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

В случае сортировки таблицы ключей наиболее выгодно выбирать алгоритмы с наименьшей сложностью(~O(n*logn)).

В случае сортировки исходной таблицы наиболее выгодно использовать такие способы сортировки с наименьшим количеством перестановок, т.к. перестановка местами записей целиком занимает довольно продолжительное время.