**Лабораторная работа №2 «Записи с вариантами. Обработка таблиц»**

**Выполнил: Чалый Андрей Александрович**

**Группа: ИУ7-32Б**

**Цель работы:** приобрести навыки работы с типом данных «запись» (структура), содержащим вариантную часть (объединение, смесь), и с данными, хранящимися в таблицах, произвести сравнительный анализ реализации алгоритмов сортировки и поиска информации в таблицах, 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.

**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.

**Исходные данные:**

На вход программы подаются данные трех типов:

1) Выбор поля, указанного в подсказке. Необходимо ввести одно из указанных значений, никакие другие числа недопустимы.

2) Слова на английском языке, длина слов не должна превышать 30 символов.

3) Неотрицательные целые числа

**Возможные ошибки пользователя:**

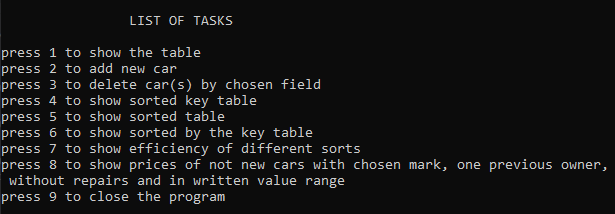
1) При некорректном вводе или выборе меню пользователь получит следующее сообщение об ошибке "Ошибочный ввод"

2) При превышении количества элементов таблицы: "Переполнение массива записей"

3) При недоступности файла с данными: "Файл не может быть открыт!”

**Интерфейс программы:**

1. Главное меню

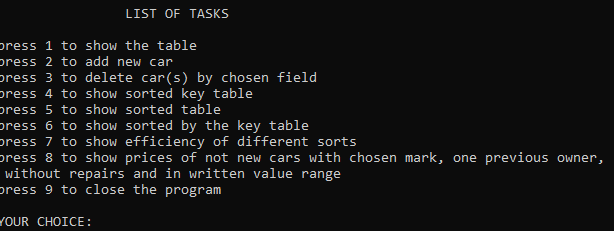


2) Фрагмент напечатанной таблицы

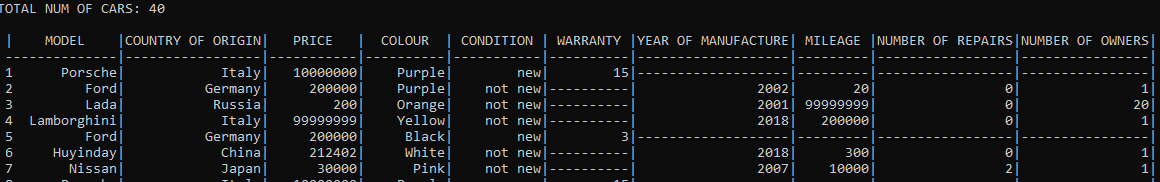
**Формат выходных данных:**

1. Главное меню

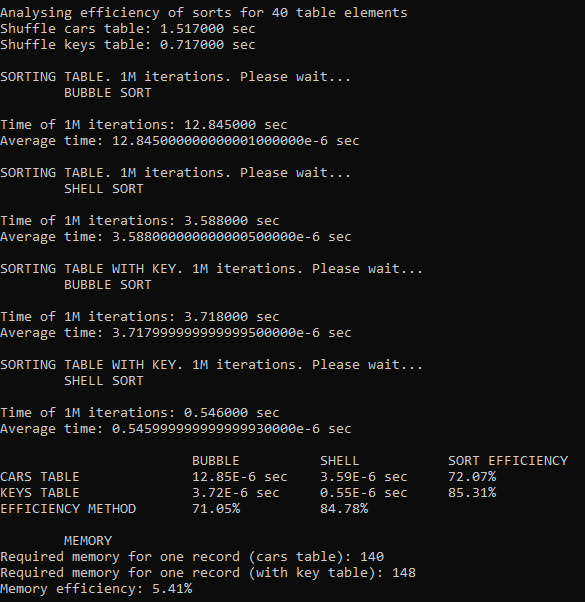
\



2) Фрагмент напечатанной таблицы



3) Анализ эффективности



**Сравнение времени работы сортировок:**

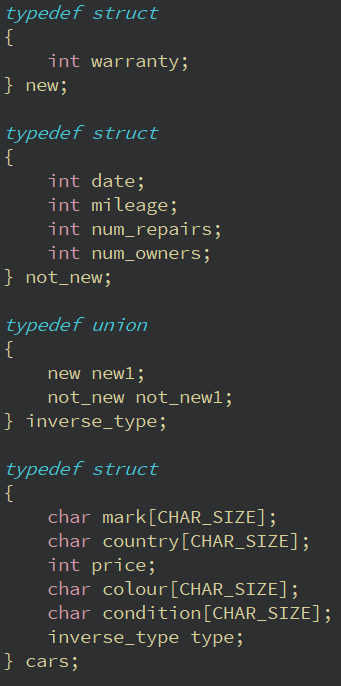
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Кол-во элементов** | **Bubble\_sort** | **Bubble\_sort\_key** | **Shell\_sort** | **Shell\_sort\_key** |
| **20** | **3.28E-6 sec** | **0.99E-6 sec** | **1.19E-6 sec** | **0.19E-6 sec** |
| **40** | **13.29E-6 sec** | **3.94E-6 sec** | **3.81E-6 sec** | **0.49E-6 sec** |
| **100** | **88.25E-6 sec** | **25.66E-6 sec** | **12.00E-6 sec** | **2.01E-6 sec** |

Сортировка Шелла заметно быстрее пузырьковой сортировки. Это становится особенно заметно с ростом количества элементов таблицы. Процентное отношение для 40 элементов: Использование сортировки Шелла по сравнению с пузырьковой эффективнее на 70%. Использование таблицы ключей эффективнее на 85% но при этом мы теряем в памяти 5.41%.

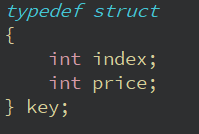
**Вывод:**

Во время работы было проведено сравнение времени работы Шелла и пузырьковой сортировок. Сортировка пузырьком имеет более высокую алгоритмическую сложность, что приводит к ее медлительности (пузырек - O(n^2), Шелла — O(NlogN)). Использование сортировки Шелла эффективнее на 70%.Для хранения информации о машинах была использована вариативная запись. Хранение дополнительной таблицы ключей значительно уменьшает время сортировки (на 85%) , хоть и требует незначительного дополнительного объема памяти для ее хранения на 5.41%. Таким образом, сравнивая обычную таблицу и неэффективную сортировку с таблицей ключей и эффективной сортировкой, мы повышаем эффективность программы на 70-90%, хоть и теряем немного в памяти и используем сложный ключ.

**Структуры данных для хранения записей таблицы:**



**Структура данных для хранения ключей:**



**Контрольные вопросы:**

**1. Как выделяется память под вариантную часть записи?**

Объем памяти, необходимый для записи с вариантами складывается из длин полей фиксированной части и максимального по длине поля вариантной части.

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

Тип данных в вариантной части при компиляции не проверяется, поэтому, контроль за правильностью ее использования возлагается на программиста. Поэтому может произойти что угодно.

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

Контроль за правильностью ее использования возлагается на программиста .

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

При больших размерах таблиц поиск данных, имеющих указанный ключ, может потребовать больших затрат времени. Если же помимо поиска требуется произвести сортировку данных, то временные затраты многократно возрастут, так как потребуется осуществлять их перестановку (перемещение). В этом случае можно уменьшить время обработки за счет создания дополнительного массива – таблицы ключей, содержащей индекс элемента в исходной таблице и выбранный ключ.

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

Если каждая запись таблицы состоит из небольшого кол-ва полей, то полезно обрабатывать данные в таблице. В других случаях выгоднее использовать таблицу ключей.

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

Если обработка данных производится в таблице, то необходимо использовать алгоритмы сортировки, предполагающие наименьшее кол-во операций перестановки. Если сортировка производится по таблице ключей, эффективнее использовать сортировки с наименьшей сложностью работы.