

**Отчет по дисциплине:**

**“Типы и Структуры Данных”**

*Лабораторная работа*

*«Записи с вариантами. Обработка таблиц»*

*Выполнил студент*: Сироткина П.Ю.

*Группа*: ИУ7-36Б

*Преподаватель*: Силантьева А.В.

**Москва, 2020 г.**

1. ***Условие задачи***:

* Создать таблицу, содержащую не менее 40 записей с вариантной частью, имеющих следующую структуру: марка автомобиля, страна-производитель, цена, цвет,

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.

* Оценить относительную эффективность программы (в процентах) по времени и по используемому объему памяти в зависимости от используемого алгоритма и от объема сортируемой информации

1. ***Описание ТЗ:***

***А) Входные и выходные данные***

*Входные данные:* текстовый файл, номер команды из меню. В зависимости от команды может потребоваться ввод дополнительных параметров (например, марка автомобиля или диапазон цен).

При этом:

* + Максимальная длина строкового поля составляет 15 символов
  + Поле «цена» вводится в рублях
  + Поле «пробег» в вариантной части вводится в километрах
  + Поле «гарантия» в вариантной части вводится в годах
  + Каждое поле записывается на отдельной строке
  + После записи каждой структуры в файле идет пустая строка

*Выходные данные:* таблица с расчетными данными или выполнение определенного действия с последующим сообщением о результате выполнения в соответствии с выбранным пунктом меню.

***Б) Описание задачи, реализуемой программой:***

Задачи, которые способна решать программа:

*1) Напечатать данные –* выбор файла ипечать данных из файла на экран в виде таблицы.

1. *Отсортировать данные –* выбор файла, ввод данных для сортировки(марка автомобиля и ценовой диапазон), вывод цен подходящих под условие машин либо сообщение о отсутствии таких машин в таблице.
2. *Добавить информацию о машине в конец таблицы* - выбор файла, ввод информации о записи, вывод сообщения о результате операции на экран.
3. *Удалить данные по полю –* выбор файла, ввод поля для удаления, удаление записей из файла, если такие записи существуют, и вывод сообщения о результате операции на экран.
4. *Справка –* вызов справочного меню(не предназначено для стандартных пользователей).
5. *Выход* *–* завершение работы программы.

При выборе пункта 5 (не предназначен для стандартных пользователей) предусмотрены следующие команды:

1. *Просмотр отсортированной таблицы ключей при неотсортированной исходной таблице –* вывод информации на экран.
2. *Вывод упорядоченной исходной таблицы –* вывод информации на экран.
3. *Вывод исходной таблицы в упорядоченном виде, используя упорядоченную таблицу ключей –* вывод информации на экран.

*10) Вывод результатов сравнения эффективности работы программы при обработке данных в исходной таблице и таблице ключей –* вывод информации на экран.

***С) Способы обращения к программе:***

Пользователь взаимодействует с программой посредством меню со следующими функциями:

*1) Напечатать данные*

1. *Отсортировать данные*
2. *Добавить информацию о машине в конец таблицы*
3. *Удалить данные по полю*
4. *Справка*
5. *Выход*

*При выборе пункта 5(не предназначен для стандартных пользователей):*

1. *Просмотр отсортированной таблицы ключей при неотсортированной исходной таблице*
2. *Вывод упорядоченной исходной таблицы*
3. *Вывод исходной таблицы в упорядоченном виде, используя упорядоченную таблицу ключей*
4. *Вывод результатов сравнения эффективности работы программы при обработке данных в исходной таблице и таблице ключей*

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

**3. Описание внутренних структур данных:**

В программе для хранения информации об автомобилях используется следующая структура данных:

typedef struct {

char brand[MAX\_STR\_LEN]; char country[MAX\_STR\_LEN]; int cost; char color[MAX\_STR\_LEN]; int new; union { int warranty; used\_car\_t used;

} condition;

} car\_t; , в которой:

* поле **brand** – строка, содержащая название марки машины,
* поле **country** – строка, содержащая название страны-производителя
* поле **cost -** целое число, стоимость автомобиля
* поле **color** – строка, содержащая название цвета автомобиля
* поле **condition** – числовой индикатор новизны автомобиля(равен 0, когда машина является новой, и 1 иначе)

В случае *новой* машины: вариативное поле **warranty** – целое число, гарантия автомобиля в годах

В случае *не новой* машины: вариативное поле **used** – структура для хранения информации о не новых машинах со следующими полями:

typedef struct { int year; int mileage; int repairs; int owners; } used\_car\_t;, в которой:

* 1. поле **year** – целое число, год выпуска автомобиля в годах
  2. поле **mileage** – целое число, пробег автомобиля в километрах
  3. поле **repairs** – целое число, количество ремонтов
  4. поле **owners** – целое число, количество собственников Также существует структура для хранения информации о ключах, выбранных из исходной таблицы:

typedef struct { int field; int index;

} key\_t; в которой:

* поле **field** – значение выбранного ключа
* поле **index** – индекс соответствующей записи в исходной таблице

**4. Описание алгоритма:** как и в сортировке исходной таблицы, так и в сортировке таблицы ключей, применяются следующие алгоритмы:

*Сортировка пузырьком*: Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N – 1 раз, где N - длина массива.

*Быстрая сортировка*: 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.

Таблица ключей формируется на основе исходной таблицы следующим образом:

Заранее определяется, по какому полю будет производиться поиск информации. В соответствии с этим каждая из записей в таблице ключей содержит 2 поля – одно содержит значение выбранного поля для поиска информации, другое – индекс соответствующей записи в исходной таблице.

Для каждого типа таблиц (ключей и исходной) записаны функции быстрой и медленной сортировки на базе алгоритмов, приведенных выше. Если информацию о поиске необходимо зафиксировать, предусмотрены специальные функции с префиксом *match*.

**5. Тестирование программы**

**Набор тестов:**

|  |  |
| --- | --- |
| **Действие** | **Результат** |
| Выбор пункта 3 в меню  Выбор корректного файла  Добавление следующей структуры:  BMW  USA  500  10 | Добавление структуры в конец файла.  Сообщение: “Информация о машине успешно добавлена.” |
| Выбор пункта 4 в меню  Выбор корректного файла  Удаление структуры по полю brand Volvo | Удаление из файла всех структур с введенной маркой. Сообщение: “Данные удалены.” |
| Выбор пункта 1 в меню  Выбор корректного файла | Печать таблицы |

|  |  |
| --- | --- |
| Ввод недопустимого номера команды в меню(например, -1 или 56) | Сообщение: «Ошибка ввода: необходимо ввести номер одного из предложенных вариантов.» |
| Открытие файла, содержащего некорректные данные | Сообщение: «Ошибка: файл содержит некорректные данные.» |
| Открытие несуществующего файла | Сообщение: «Ошибка во время открытия файла.» |
| Добавление записи при максимальном размере таблицы | Сообщение: «Невозможно добавить запись – достигнут максимальный размер файла.» |
| Ошибка при вводе целочисленного поля во время  добавления/удаления(например, ввод 12ук4) | Сообщение: «Ошибка: должно быть введено целое неотрицательное число.» |
| Ошибка при вводе строкового поля во время  добавления/удаления(превышена максимальная длина) | Сообщение: «Ошибка: превышена максимальная длина строки.» |
| Ошибка при вводе строкового поля во время добавления/удаления(пустая строка) | Сообщение: «Ошибка: пустая строка.» |

**6. Оценка эффективности. Выводы о проделанной работе**

Замеры, выполненные программой:



Во время каждого из замеров взято среднее время, т.к. получается большая погрешность во время вычисления времени выполнения программы в виду того, что другие приложения так же занимают систему. Для каждого показателя было проведено 1000 опытов.

Из таблицы видно, что в целом для оптимизации сортировки посредством создания таблицы затрачиваемая память увеличивается на 10%.

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

|  |  |  |
| --- | --- | --- |
| **Количество записей** | **Таблица ключей / исходная таблица (Сортировка пузырьком, сложность O(n2)** | **Таблица ключей / исходная таблица**  **(Быстрая сортировка, сложность O(n\*logn))** |
| 100 | ~5% | ~0% |
| 250 | ~18% | ~18% |
| 500 | ~14% | ~19% |
| 1000 | ~13% | ~20% |
| 2000 | ~11% | ~22% |
| 5000 | ~13% | ~20% |
| 7500 | ~12% | ~18% |
| 10000 | ~11% | ~16% |

**Выводы:**

Чем больше размер сортируемой таблицы, тем более выгодно использовать сортировку с использованием таблицы ключей. Однако за скорость приходится «платить» памятью, требуемой для создания этой таблицы. Для конкретной задачи размер таблицы ключей составил 10% от исходной. Таблицы ключей наиболее выгодно применять до тех пор, пока затраты на память дают существенное увеличение скорости сортировки. Наиболее эффективно применять таблицу ключей, когда выбранное поле является целочисленным типом данных, однако если в качестве выбранного ключа выступает строковый тип данных, выигрыш по времени значительно снижается, т.к. обработка строк занимает больше ресурсов.

**7. Ответы на 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?***

Поведение программы не определено.

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

При обработке записей с вариантами ответственность возлагается на программиста.

***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 посредством сокращения количества перестановок в исходной таблице, которые является ресурсозатратными.

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

Эффективно использовать таблицу ключей, когда затрачиваемая память на выделение таблицы ключей значительно меньше по сравнению с выигрышем по времени работы программы. 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). Наиболее эффективно применять таблицу ключей, когда выбранное поле является целочисленным типом данных, однако если в качестве выбранного ключа выступает строковый тип данных, выигрыш по времени значительно снижается, т.к. обработка строк занимает больше ресурсов.

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

В случае сортировки исходной таблицы наиболее выгодно использовать такие способы сортировки, в которых количество перестановок минимально.

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