Министерство образования и науки 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 УЧЕБНОЙ ДИСЦИПЛИНЕ “ТИПЫ И СТРУКТУРЫ ДАННЫХ” ЛАБОРАТОРНАЯ РАБОТА №2

Студент Ляпина Наталья Викторовна

фамилия, имя, отчество

Группа ИУ7-32Б

Вариант 4

Студент _____ Ляпина Н.В.

подпись, дата *фамилия, и.о.*

Преподаватель _____ Силантьева А.В.

подпись, дата *фамилия, и.о.*

2021 г.

Оглавление

1) Условие задачи……...……3

2) Структура данных……...……...4

3) Интерфейс модулей……...……...5

4) Описание алгоритма……..…….……..6

5) Тесты…….……....……7

6) Вывод……..……...…….……....8

7) Ответы на контрольные вопросы……8

Задание

1. **Общее задание**

   Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами (объединениями)). Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ – любое невариантное поле (по выбору программиста), используя: а) саму таблицу, б) массив ключей. (Возможность добавления и удаления записей в ручном режиме обязательна). Осуществить поиск информации по варианту.

2. **Задание по варианту**

   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.

3. **Входные данные**

- Команда для выбора действия
    1. Загрузить список машин из файла в программу;
    2. Вывести таблицу машин
    3. Добавить машину
    4. Удалить машину по полю
    5. Вывести цены не новых машин указанной марки с одним предыдущим собственником, отсутствием ремонта в указанном диапазоне цен
    6. Вывести массив ключей (ключ - цена машины)
    7. Отсортировать массив ключей методом быстрой сортировки
    8. Отсортировать массив ключей методом пузырьковой сортировки
    9. Отсортировать таблицу методом быстрой сортировки
    10. Отсортировать таблицу методом пузырьковой сортировки
    11. Вывести отсортированную таблицу, используя массив ключей
    12. Вывести результаты сравнения эффективности программы при обработке таблицы и массива ключей
    0. Выйти из программы

- Запись, содержащая новую строку таблицы
- Файл с таблицей

- **Требования к входным данным**

1) Марка автомобиля содержит символы от A до Z (включая строчные буквы) и вводится без пробелов.
2) Страна содержит символы от A до Z (включая строчные буквы) и вводится без пробелов.
3) Цена автомобиля не может быть не натуральной величиной.
4) Цвет машины содержит символы от A до Z (включая строчные буквы) и вводится без пробелов.
5) Состояние машины может быть либо “new”, либо “old”.
6) Гарантия на машину не может быть отрицательной.
7) Год изготовления машины не может быть не натуральной величиной.
8) Пробег не может быть не натуральной величиной.
9) Количество ремонтов не может быть отрицательным.
10) Количество владельцев не может быть отрицательным.

4. Выходные данные
- Таблица машин
- Таблица ключей
- Результат сортировки
- Результат фильтрации
- Оценка относительной эффективности программы

5. Действие программы

Программа выполняет обработку таблицы машин используя структуру с вариантным полем.

5. Обращение к программе

Программа может быть вызвана через консоль компилятора с помощью команды “./app.exe”

6. Аварийные ситуации

В случае аварийной ситуации выводится сообщение об определенной ошибке.

Неверный ввод:

- Несуществующая команда;
- Ошибка при открытии файла;
- При загрузке таблицы произошла ошибка чтения размера таблицы;

- При загрузке таблицы произошла ошибка при чтении любого из полей на основе требований к входным данным;
- При добавлении новой записи в таблицу произошла ошибка при вводе полей структуры на основе требований к входным данным;
- При удалении структуры произошла ошибка чтения поля.

Структура данных

Для реализации данной задачи была создана структура *table_t,* которая используется для хранения всей таблицы. Эта структура содержит массив записей типа car_t, размер таблицы size и массив ключей типа key_t.
MAX_TABLE_SIZE = 6000 - максимальный размер таблицы.

Хранение таблицы записей

```
typedef struct
{
        size_t size;
        cat_t [MAX_TABLE_SIZE];
        keys_t keys[MAX_TABLE_SIZE];
} table_t;
```

Хранение одной записи таблицы осуществляется с помощью структуры car_t, которая содержит строку с маркой машины, строку со страной производства машины, ценой машины типа int, строку с состоянием машины и с вариантным полем типа type_of_car_t.
CAR_NAME_LEN = 20 - максимальная длина марки машины.
COUNTRY_NAME_LEN = 25 - максимальная длина страны машины.
CONDITION_LEN = 4 - длина поля состояния (“new” или “old”).
COLOR_LEN = 15 - максимальная длина цвета машины.

Хранение записи таблицы

```
typedef struct
{
        char car_name [CAR_NAME_LEN];
        char country [COUNTRY_NAME_LEN];
        int price;
        char color[COLOR_LEN];
        char condition[CONDITION_LEN];
        type_of_car_t car_type;
} car_t;
```

Вариантные поля хранятся с помощью объединения записей union. Такое решение позволяет рационально использовать память, если одно из полей структуры не задействовано в записи машины.

Хранение вариантных полей записи

```
typedef union
{
        new_t new_car;
        old_t old_car;
} type_of_car_t;
```

Вариантные поля хранятся в структурах old_t и new_t.

```
typedef struct
{
        int year;
        int mileage;
        int repairs_num;
        int owners;
} old_t;

typedef struct
{
        int garanty_till;
} new_t;
```

Интерфейс модулей

Для обработки типа car_t используются функции:

int read_record(FILE *f, car_t *car)
// Функция проверки ввода записи таблицы
// f - файловая переменная, car - запись

int old_car_read(FILE *f, old_t *old)
// Функция проверки ввода вариантного поля для старой машины
// f - файловая переменная, old - вариантное поле записи

int new_car_read(FILE *f, new_t *new)
// Функция проверки ввода вариантного поля для новой машины
// f - файловая переменная, new - вариантное поле записи

int char_check(FILE *f, size_t max_len, char *str)
// Функция проверки ввода строки

// f - файловая переменная, max_len - максимальный размер строки, str - строка, в которую нужно записать строку

int compare(car_t *car, char *name, int price_min, int price_max)
// Функция сравнения записи с условиями
// car - запись, name - марка машины, price_min - минимальная цена, price_max - максимальная цена.

Для обработки типа table_t используются функции:
int load_table(char *file_name, size_t max_size, table_t *table)
// Функция загрузки таблицы записей из файла в программу
// file_name - имя файла, max_size - максимальный размер таблицы, table - структура для хранения таблицы.

int print_table(table_t *table)
// Функция печати таблицы записей
// table - структура для хранения таблицы.

int add_car(table_t *table)
// Функция добавления новой записи в таблицу
// table - структура для хранения таблицы.

int delite_car(table_t *table)
// Функция удаления записей, подходящих под условие, из таблицы
// table - структура для хранения таблицы.

int print_task(table_t *table)
// Функция вывода на экран таблицы записей, подходящих под условие
// table - структура для хранения таблицы.

void get_keys(table_t *table)
// Функция получения массива ключей
// table - структура для хранения таблицы.

int print_keys(table_t *table)
// Функция печати массива ключей
// table - структура для хранения таблицы.

int print_sorted_keys_table(table_t *table)

```
// Функция печати отсортированной таблицы по массиву ключей
// table - структура для хранения таблицы.

int sort_keys_qsort(table_t *table)
// Функция сортировки массива ключей первым способом
// table - структура для хранения таблицы.

int sort_keys_bubble(table_t *table)
// Функция сортировки массива ключей вторым способом
// table - структура для хранения таблицы.

int sort_table_qsort(table_t *table)
// Функция сортировки таблицы первым способом
// table - структура для хранения таблицы.

int sort_table_bubble(table_t *table)
// Функция сортировки таблицы вторым способом
// table - структура для хранения таблицы.
```

Описание алгоритма

1. Вводится команда для выполнения определенной функции программы.
2. Выполняется введённая функция
    - Загрузка таблицы из файла
    - Печать таблицы
    - Добавление записи в конец таблицы
    - Удаление записей из таблицы по вариантному полю (гарантия)
    - Вывод записей таблицы, удовлетворяющих условию задачи
    - Вывод массива ключей (ключ - цена машины)
    - Сортировка массива ключей двумя способами
    - Сортировка таблицы двумя способами
    - Вывод отсортированной таблицы на экран с помощью массива ключей
    - Сравнение методов сортировки массивом ключей и сортировки таблицы.
3. При ошибке выполнения функции выводится сообщение об ошибке и программа завершается с не нулевым кодом возврата.

Тесты

Негативные тесты

№	Описание	Вводимая структура	Результат
1	Недопустимый символ в марке машины	Mazda3	Ошибка
2	Недопустимый символ в названии страны	Ru33ia	Ошибка
3	Цена не натуральная величина	-102047	Ошибка
4	Состояние машины отличается от допустимых	oldest	Ошибка
5	Гарантия на машину отрицательная	-12	Ошибка
6	Год машины не натуральная величина	0	Ошибка
7	Отрицательный пробег машины	-1047	Ошибка
8	Отрицательное количество ремонтов	-8	Ошибка
9	Отрицательное количество владельцев	-6	Ошибка
10	Недопустимый символ в команде	5g	Ошибка

11	Недопустимая команда	103	Ошибка
12	Размер файла больше допустимого	7500	Ошибка
13	Размер файла отрицательный или равен 0	-3	Ошибка
14	Невозможный диапазон цены	max = 3 min = 5	Ошибка
15	Добавление в переполненную таблицу	comand = 3	Ошибка
16	Открытие не существующего файла	comand = 1	Ошибка

Позитивные тесты

№	Описание	Входные данные	Результат
1	Загрузка таблицы	comand = 1	Таблица заполнена
2	Чтение марки машины	BMW	Успешное чтение строки
3	Чтение страны машины	Russia	Успешное чтение строки
4	Чтение цены машины	12503	Успешное чтение цены
5	Чтение состояния машины	new	Успешное чтение состояния
6	Чтение гарантии	4	Успешное чтение гарантии
7	Чтение года	2021	Успешное чтение года
8	Чтение пробега	125000	Успешное чтение пробега
9	Чтение количества ремонтов	4	Успешное чтение количества ремонтов
10	Чтение количества владельцев	1	Успешное чтение количества владельцев
11	Чтение цвета машины	black	Успешное чтение цвета машины

Оценка эффективности

Объем занимаемой памяти

Кол-во элементов	Таблица	Массив ключей
1000	88000	8000
2000	176000	16000
4000	352000	32000
5000	440000	40000

Проигрыш по памяти 9%

Сравнение эффективности

Кол-во элементов	Процентное соотношение времени обработки таблицы пузырьковой и быстрой сортировкой	Процентное соотношение времени обработки массива ключей пузырьковой и быстрой сортировкой
1000	23.44 %	10.3 %
2000	46.87 %	34.12 %
4000	139.62 %	75.10 %
5000	156.52 %	80.85 %

Вывод

Для комбинации разных типов данных удобно использовать тип запись (структура), потому что параметры реальных объектов бывает сложно описать, используя только числовые, либо только символьные типы данных. Если же какой то тип данных содержит вариативные параметры, то удобно использовать запись с вариантами. Так, для этой задачи использовать запись с вариантами очень удобно: поля имеют разный тип данных (int и char[]), а поле, содержащее в себе информацию о статусе автомобиля, может быть вариативным.

Если же говорить об эффективности обработки таблиц, то можно сделать вывод, что чем больше становится размер таблицы, тем эффективнее становится обрабатывать таблицу ключей (при увеличении размера таблицы от 1000 до 5000 элементов скорость обработки таблицы ключей по сравнению с обработкой исходной таблицы увеличивается от 30% до 49%), нежели обрабатывать исходную таблицу.

С другой стороны, для хранения таблицы ключей требуется дополнительная память, поэтому при малых размерах таблицы эффективнее обрабатывать исходную таблицу, так как разница во времени будет незначительная, а затраты на память увеличатся.

Ответы на контрольные вопросы

1. Как выделяется память под вариантную часть записи? В языке СИ вариативная часть записей реализована с помощью union (объединений). Память выделяется в одном блоке памяти, имеющий размер, равный размеру наибольшего поля объединения.

2. Что будет, если в вариантную часть ввести данные, несоответствующие описанным? Такие действия влекут за собой неопределённое поведение.

3. Кто должен следить за правильностью выполнения операций с вариантной частью записи? Тип данных в вариантной части при компиляции не проверяется, поэтому, контроль за правильностью ее использования возлагается на программиста.
4. Что представляет собой таблица ключей, зачем она нужна? Таблица ключей представляет собой таблицу, в которой находится два столбца: номер ячейки в исходной таблице и значение невариативного, выбранного программистом поля исходной таблицы для этой ячейки (в данном случае - цена автомобиля).

5. В каких случаях эффективнее обрабатывать данные в самой таблице, а ко19 гда – использовать таблицу ключей? Обрабатывать данные в самой таблице эффективнее, когда время обработки таблицы не так важно, как объём занимаемой памяти. Наоборот, использование таблицы ключей эффективнее когда нужно быстрое время обработки исходной таблицы и не так важна дополнительная задействованная память. Так же, использование таблицы неэффективно, когда сама таблица состоит из небольшого количества столбцов, потому что таблица ключей будет лишь занимать дополнительное место в памяти и не даст почти никакого преимущества во времени.

6. Какие способы сортировки предпочтительнее для обработки таблиц и почему? Для таблиц из большого количества записей предпочитетльно использовать быстрые и устойчивые способы сортировки, например quicksort, mergesort. Если же в таблице не так много записей, то предпочтительнее использовать читаемые и лёгкие в понимании

алгоритмы сортировки, например, пузырьковую сортировку или сортировку простым выбором.