Министерство науки и высшего образования 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»

ОТЧЁТ ПО ЛАБОРАТРОНОЙ РАБОТЕ №2

«Записи с вариантами, обработка таблиц»

по курсу «Типы и структуры данных»

Студент: Шимшир Эмирджан Османович

Группа: ИУ7-33Б

Студент _____ Шимшир Э.О.

подпись, дата *фамилия, и.о.*

Преподаватель _____ Барышникова М.Ю

подпись, дата *фамилия, и.о.*

Оценка _____

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. Оценка эффективности должна быть относительной (в %).

Ввести список стран, в которых можно отдохнуть, содержащий название страны, количество жителей, столицу, материк, нужна ли прививка или ПЦР, основной вид туризма (экскурсионный - количество объектов, основной вид (природа, история, искусство); пляжный – основной сезон, температура воздуха и воды, время полета до страны; спортивный – вид спорта (горные лыжи, серфинг, восхождения), минимальная стоимость отдыха,). Вывести список стран на выбранном материке, где можно заняться указанным видом спорта.

Техническое задание

Входные данные

Файл с данными в формате .txt (важно представление данных). Значения полей структуры перечислены через пробел на одной строке.

Целое число, задающее номер пункта меню.

Строковые и числовые данные при добавлении удалении и поиске данных.

Выходные данные

Таблица, выгруженная из файла.

Текущее состояние таблицы до/после добавления, удаления из нее записей.

Исходная таблица или таблица ключей в отсортированном виде.

Результат поиска по таблице.

Количественная характеристика сравнения эффективности вариантов сортировки.

Задачи, реализуемые программой

1. Просмотр таблицы.

2. Добавление информации о новой стране.

3. Удаление информации о странах с заданным количеством жителей.

4. Просмотр всех стран на заданном континенте с заданным типом спортивной активности

5. Просмотр текущей таблицы ключей.

6. Просмотр упорядоченной по количеству жителей таблицы ключей (сортировка пузырёком $O(n^2)$).

7. Просмотр упорядоченной по количеству жителей таблицы ключей (сортировка qsort $O(n*\log(n))$).

8. Просмотр упорядоченной по количеству жителей таблицы (сортировка пузырёком $O(n^2)$).

9. Просмотр упорядоченной по количеству жителей таблицы (сортировка qsort $O(n*\log(n))$).

10. Просмотр таблицы в упорядоченном виде по количеству жителей по упорядоченной таблице ключей (сортировка пузырёком $O(n^2)$).

11. Просмотр таблицы в упорядоченном виде по количеству жителей по упорядоченной таблице ключей (сортировка qsort $O(n*\log(n))$).

12. Сравнение времени работы сортировок разной сложности ключей (сортировка пузырёком $O(n^2)$ и сортировка qsort $O(n*\log(n))$) и количества затраченной памяти.

Возможные аварийные ситуации и ошибки пользователя

Некорректный ввод пункта меню.

Некорректный ввод строковых данных (превышение длины строки).

Некорректный ввод целочисленных данных.

Попытка просмотра, сортировки, удаления и поиска данных в пустой таблице.

Описание внутренних структур данных

Для хранения каждой записи таблицы использована структура:

```
typedef struct
{
    char country[MAX_FIELD + 2];
    char capital[MAX_FIELD + 2];
    char mainland[MAX_FIELD + 2];
    int count_people;
    int need_PCR;
    tourism_t tourism;
    info_t more_info;
} country_t;
```

Поля структуры **country_t**:

country – название страны

capital – название столицы

mainland – название материка

count_people – количество жителей

need_PCR – нужен ли ПЦР тест

tourism – переменная типа **tourism_t**, содержит значение типа вариантой части

more_info – переменная типа **more_info_t**, вариантная часть записи

Объединение для хранения вариантной части записи:

```
typedef union
{
    excursion_t excursion;
    beach_t beach;
    sport_t sport;
} info_t;
```

Поля структуры **info_t**:

excursion – структура типа **excursion_t**, с полями параметрами

beach – структура типа **beach_t**:, с полями параметрами

sport- структура типа **sport_t**, с полями параметрами

```
typedef struct
{
    char excursion_type[MAX_FIELD + 2];
    int count_objects;
} excursion_t;
```

Поля структуры **excursion_t**:

excursion_type – тип эксукрсий

count_objects – количество объекстов

```
typedef struct
{
    char beach_type[MAX_FIELD + 2];
    int temperature_air;
    int temperature_water;
    int time_flight;
} beach_t;
```

Поля структуры **beach_t**:

beach_type – пляжный сезон

temperature_air – температура воздуха

temperature_water – температура воды

time_flight – времея полета

```
typedef struct
{
    char sport_type[MAX_FIELD + 2];
    int price_min;
} sport_t;
```

Поля структуры **sport_t**:

sport_type – вид спорта

price_min – минимальная цена

Взможные значения переменной *tourism*

```
typedef enum tourism
{
    EXCURSION,
    BEACH,
    SPORT
} tourism_t;
```

Для хранения ключей таблицы использована структура:

```
typedef struct
{
    int index;
    int count_people;
} country_key_t;
```

index – номер страны в исходной таблице

count_people – количество жителей

В программе создается массив структур типа **country_t** и **country_key_t** максимальный размер которых **MAX_TABLE** = 200.

Название файла с данными передается через аргументы командной строки

Описание кодов ошибок

Ниже описаны значения ошибок и их коды

```
#define EXIT_SUCCESS 0
#define EXIT_FAILURE 1
#define ERR_KEY 2
#define ERR_MAX_TABLE 3
#define ERR_READ_COUNTRY 4
#define ERR_READ_COUNT_PEOPLE 5
#define ERR_READ_CAPITAL 6
#define ERR_READ_MAINLAND 7
#define ERR_READ_NEED_PCR 8
#define ERR_READ_TOURISM 9
#define ERR_READ_COUNT_OBJECTS 10
#define ERR_READ_EXCURSION_TYPE 11
#define ERR_READ_BEACH_SEASON 12
#define ERR_READ_TEMP_AIR 13
#define ERR_READ_TEMP_WATER 14
#define ERR_READ_TIME_FLIGHT 15
#define ERR_READ_SPORT_TYPE 16
#define ERR_READ_PRICE_MIN 17
#define ERR_TOURISM_TYPE 18
```

Ограничения на входные данные

Название, столица, материк, тип экскурсии, пляжный сезон, вид спорта страны должны быть по одному слову и не больше 15 символов. Остальные параметры – целые положительные числа. При вводе переменных – флагов (нужен ли ПЦР) пользователю нужно на выбор ввести либо 0 либо 1.

В массиве максимально может хранится информации о 200 странах. Нельзя вызвать функцию добавления страны, если их уже 200 и нельзя вызвать функцию удаления, если массив пока пустой.

В случае неправильного ввода программа завершается.

Описание функций

```
/*
* Функция, выводящая таблицу на экран.
* Принимает указатель на таблицу (массив), которую необходимо вывести и
размер таблицы (массива)
*/
void print_table(country_t *country, size_t count);
```

```c
/*
* Функция, печатающая таблицу ключей.
* Принимает указатель на таблицу (массив), которую необходимо вывести и
размер таблицы (массива)
*/
;
void print_table_key(country_key_t *key_arr, size_t count);
```

```c
/*
* Функция, считывающая запись для добавления.
* Принимает указатель на таблицу (массив), в которую необходимо добавить
* запись и указатель на таблицу ключей (массив) и их размер.
*/
int add_country(country_t *country_arr, country_key_t *key_arr, size_t
*count);
```

```c
/*
* Функция, удаляющая запись по цене.
* Принимает указатель на таблицу (массив), в которую необходимо добавить
* запись и указатель на таблицу ключей (массив) и их размер.
*/
void del_country(country_t *country_arr, size_t *count, int count_people);
```

```c
/*
* Функция, реализующая сортировку qsort.
* Принимает указатель на первый элемент в массиве, размер массива,
*/
void qsort_table(country_t *country_arr, size_t count);
```

```c
/*
* Функция, реализующая сортировку пузырьком.
* Принимает указатель на первый элемент в массиве, размер массива,
*/
void sort_table(country_t *country_arr, size_t count);
```

```c
/*
* Функция, выводящая количественную характеристику времени
* работы и используемой памяти при двух сортировках
* исходной таблицы и таблицы ключей.
* Принимает указатель на исходную таблицу и таблицу ключей.
*/
void compare_sort(country_t *country_arr, size_t count, country_key_t
*key_arr);
```

Описание алгоритма и исследование полученных результатов

Для сортировки используется алгоритм **qsort** из стандартной библиотеки **stdlib.h** языка Си, сложность которого **O(n*log(n))**. Также используется алгоритм сортировки пузырьком, сложность которого **O(n^2)).**

Для нахождения информации использован алгоритм линейного поиска.

Результаты измерения времени сортировок при различном количестве записей. Количество экспериментов равно 1000 для каждой сортировки каждой таблицы каждого размера (значения считались как среднее арифметическое). При проведении экспериментов программа была скомпилирована без оптимизаций (**-O0**), внешние задачи отсутствовали.

Количество записей	Быстрая сортировка		Сортировка пузырьком	
	Исходная таблица, микросек	Таблица ключей, микросек	Исходная таблица, микросек	Таблица ключей, микросек
50	2 (100%)	1 (100%)	21 (1050%)	15 (1500%)
100	3 (100%)	1 (100%)	63 (2100%)	48 (4800%)
200	9 (100%)	4 (100%)	231 (2500%)	191 (4800%)

Таким образом, получается, что qsort намного быстрее, а особенно на большом количестве данных.

Также, можем сделать вывод о том, что намного быстрее сортировать таблицу ключей, чем исходную таблицу.

Расход памяти на максимальном количестве размера массива – 200 структур:

Для таблицы ключей: 19200 байт. (100%)

Для всей таблицы структур: 20800 байт (108%).

Тестирование

Позитивные тесты.

№	Входные данные	Выходные данные	Результат
1	Ключ = 0	Поток вывода пустой. Завершение программы	Код возврата - 0
2	Ключ = 1	На экран выводится таблица, которая ранее была считана из файла.	Ожидание следующего ключа
3	Ключ = 2 Ввод валидной страны.	На экран выводятся правила для ввода новой страны. Так как страна валидна, выводится сообщение об успешно записанной стране.	Ожидание следующего ключа
4	Ключ = 3 Ввод валидного количества жителей для удаления – целое положительное число.	На экран выводится правила для ввода количества жителей. Так как такие страны нашлись, то выведено сообщение об успешном удалении.	Ожидание следующего ключа
5	Ключ = 4 Ввод валидного материка и вида спорта	На экран выводятся правила для введения материка и вида спорта. Далее выводится таблица с найденными странами.	Ожидание следующего ключа
6	Ключ = 5	Текущее состояние таблицы ключей	Ожидание следующего ключа
7	Ключ = 6	Вывод отсортированной таблицы ключей (сортировка пузырьком)	Ожидание следующего ключа
8	Ключ = 7	Вывод отсортированной таблицы ключей (быстрая сортировка)	Ожидание следующего ключа

9	Ключ = 8	Вывод отсортированной полной таблицы (сортировка пузырьком)	Ожидание следующего ключа
10	Ключ = 9	Вывод отсортированной полной таблицы (быстрая сортировка)	Ожидание следующего ключа
11	Ключ = 10	Вывод отсортированной таблицы с помощью таблицы ключей (сортировка пузырьком)	Ожидание следующего ключа
12	Ключ = 11	Вывод отсортированной таблицы с помощью таблицы ключей (быстрая сортировка)	Ожидание следующего ключа
13	Ключ = 12	Вывод информации о времени для 4х видов сортировок и объема занимаемой памяти.	Ожидание следующего ключа
14	Ключ = 11 Ключ = 1	В начале программа выведет отсортированную таблицу, а далее – несортированную исходную таблицу.	Ожидание следующего ключа
15	Ключ = 5 Ключ = 6 Ключ = 5	В начале программа выводит несортированный массив ключей. После дважды выводит сортированный массив ключей.	Ожидание следующего ключа

Негативные тесты

№	Входные данные	Выходные данные	Результат
1	Ключ = 2 Неверный ввод структуры. gdfhjsdhjgfhsdfjhgjdsdf	Сообщение где именно пользователь ввёл невалидный результат. “Error country. Please try again according the rules”	Код возврата = 4
2	./app.exe fake_data.txt	Завершение программы	Код возврата = 1
3	Ключ = 2 Неверный ввод структуры. Russia gdjfgdshgfhdsgfjhdsgjf	Сообщение где именно пользователь ввёл невалидный результат. “Error count of people. Please try again”	Код возврата = 5
4	Ключ = 2, но в массиве уже есть 200 элементов	Сообщение, что массив полностью заполнен “It is impossible to add a country, since there are 200 of them at most”	Код возврата = 3
5	*Ключ = 3,* но массив пустой	Сообщение, что массив пустой “You can't delete country, because table is empty.”	Ожидание следующего ключа

Контрольные вопросы

Как выделяется память под вариантную часть записи?

В языке Си, вариативная часть структуры реализована с помощью объединений - “union”. Память выделяется в одном “куске” памяти, имеющий размер, который способен вместить наибольшее поле из указанных.

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

Результат будет системно-зависимым и трудно предсказуемым. Возможно, произойдет приведение типов.

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

Ответственность за правильность проведения операций целиком и полностью лежит на программисте.

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 выбранного программистом поля исходной таблицы для этой ячейки (в моем случае – количество жителей).

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

Обрабатывать данные в самой таблице эффективнее, когда время обработки не так важно, как задействованная память. А использование таблицы ключей, наоборот, эффективно, когда нужно быстрое время обработки и не так важна дополнительная задействованная память. Так же, использование таблицы ключей неэффективно, когда сама таблица состоит из маленького количества полей, например, таблица, имеющая два поля: “Страна” и “Столица”. В таком случае, таблица ключей будет лишь занимать дополнительное место в памяти и не даст никакой выгоды во времени.

Какие способы сортировки предпочтительнее для обработки таблиц и почему?

Для таблиц из большого количества записей предпочтительно использовать стандартные и устойчивые способы сортировки, со средним временем обработки O(n*log(n)), такие как qsort и т.д.

Если же в таблице не так много записей, то предпочтительнее использовать простые алгоритмы сортировки, например, сортировку пузырьком $O(n^2)$.

Вывод

Были применены различные варианты сортировки (быстрая и сортировка пузырьком), а также два подхода к сортировке структурных типов данных (с использованием дополнительного массива ключей и без).

Были проведены замеры на разных размерах массивов стран. Несмотря на то, что таблица ключей увеличивает размер использованной памяти на 8%, её сортировка значительно быстрее сортировки исходной таблицы (в моих экспериментах минимум на 20% при сортировки пузырьком, на 100% при сортировке qsort). Благодаря таблице ключей мы можем представить таблицу в упорядоченном виде, не сортируя её.

Также сортировка пузырьком, из-за сложности $O(n^2)$ значительно уступает быстрой сортировке $O(n*\log(n))$, особенно при больших размерах массива.