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»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 ПО ДИСЦИПЛИНЕ ТИПЫ И СТРУКТУРЫ ДАННЫХ

Студент **Паншин Сергей Константинович**

Группа **ИУ7-33Б**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____ **Паншин С. К.**

Преподаватель _____ **Никульшина Т. А.**

Преподаватель _____ **Барышникова М. Ю.**

Оценка _____

2023 г.

Условие задачи

Ввести список стран, содержащий название страны, столицу, материк, необходимость наличия визы, время полета до страны, минимальную стоимость отдыха, основной вид туризма (экскурсионный – количество объектов, их основной вид (природа, история, искусство); пляжный – основной сезон, температура воздуха и воды; спортивный – вид спорта (горные лыжи, серфинг, восхождения, сплав)). Вывести список стран на выбранном материке, со стоимостью ниже указанной, где можно заняться указанным видом спорта.

Техническое задание

Разработать программу для 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.

Входные данные:

Пункт меню (число от 0 до 12):

```
Menu:
    1) Read table from fil
    2) Print keys table
    3) Print table
    4) Add new row
    5) Delete row by id (warming: you need id from table
     not from keys table and you lose keys sort order!)
```

```
    6) Print rows that meet conditions:
      on the mainland that you choose,
      with a cost below the specified one,
      where you can do the specified sport\n\
    7) Print sorted keys table
    8) Print sorted table
    9) Print table by sorted keys table
    10) Print results of comparing efficiency by work with keys table and
table
    11) Print results of using various sorting algorithms
    12) Update data in file
    0) Quit program
```

Также путь к файлу, файл, параметры добавляемой/удаляемой страны.

Выходные данные:

Текущее состояние всей таблицы/таблицы ключей, результат поиска по заданным полям, сравнение эффективности сортировок.

Возможные аварийные ситуации:

Некорректный ввод: пункта меню, пути к файлу, параметры страны.
Некорректный файл.

Способ обращения к программе

В папке с программой запустить команду *make run*.

Структуры данных

```
// Время полета
typedef struct
{
    int hours;
    int minute;
} flight_time_t;

// Перечисление видов отдыха
typedef enum
{
    EXCURSION = 1,
    BEACH,
    SPORTS
} tourism_enum;

// Перечисление типов объектов
typedef enum
{
    NATURE = 1,
    HISTORY,
    ART
} obj_main_type_enum;

// Параметры экскурсионного отдыха
typedef struct
{
    int obj_count;
    obj_main_type_enum obj_main_type;
} excursion_t;

// Перечисление сезонов
typedef enum
{
    WINTER = 1,
```

```
    SPRING,
    SUMMER,
    AUTUMN
} main_season_enum;

// параметры отдыха на пляже
typedef struct
{
    main_season_enum main_season;
    double air_temp;
    double water_temp;
} beach_t;

// Перечисление типов спорта
typedef enum
{
    DOWNHILL_SKIING = 1,
    SURFING,
    CLIMBING,
    RAFTING
} sports_enum;

// Перечисление типов туризмов
typedef union
{
    excursion_t excursion;
    beach_t beach;
    sports_enum sport_type;
} tourism_union;

// Перечисление существующих континентов
typedef enum
{
    EURASIA = 1,
    AFRICA,
    NORTH_AMERICA,
```

```
    SOUTH_AMERICA,
    AUSTRALIA,
    ANTARCTICA
} continent_enum;

// Страна являющаяся строкой таблицы
typedef struct
{
    char name[MAX_STR_LEN + 1];
    size_t name_len;
    char capital[MAX_STR_LEN + 1];
    size_t capital_len;
    continent_enum continent;
    size_t continent_len;
    int is_need_visa;
    flight_time_t flight_time;
    int min_rest_cost;
    tourism_enum tourism_type;
    tourism_union tourism;
} country_t;

// таблица ключей
typedef struct
{
    size_t country_id;
    int min_rest_cost;
} keys_table_t;

// Структура для хранения ключей и строк таблицы и количество этих строк
typedef struct
{
    keys_table_t keys[MAX_ROW_COUNT];
    country_t countries[MAX_ROW_COUNT];
    size_t rows_count;
} table_t;
```

Используемые константы

```
// общие константы
#define MAX_STR_LEN 30
#define MAX_CELL_LINE "──────────────────────────────────"
#define MAX_ROW_COUNT 10000
#define ITER_COUNT_TIME 100
// ошибки полезных функций
#define ERROR_EMPTY_INPUT 1
#define ERROR_STR_LEN 2
#define ERROR_WRONG_NUM 3
// ошибки страны
#define ERROR_WRONG_VISA_FLAG 11
#define ERROR_WRONG_HOURS 12
#define ERROR_WRONG_MINUTES 13
#define ERROR_WRONG_COST 14
#define ERROR_WRONG_TOURISM_TYPE 15
#define ERROR_WRONG_OBJ_COUNT 16
#define ERROR_WRONG_OBJ_TYPE 17
#define ERROR_WRONG_SEASON 18
#define ERROR_WRONG_AIR_TEMP 19
#define ERROR_WRONG_WATER_TEMP 20
#define ERROR_WRONG_SPORTS_TYPE 21
#define ERROR_WRONG_CONTINENT 22
// ошибки таблицы
#define ERROR_EMPRY_FILE 30
#define ERROR_ROW_COUNT 31
#define ERROR_WRONG_ID 32

// ошибки меню
#define ERROR_WRONG_MENU_ITEM 40
#define ERROR_WRONG_FILEPATH 41
```

Замеры

Производиться 100 итераций при замерах.

Время сортировки в наносекундах:

Кол-во записей	Пузырек		Шейкер	
	Таблиц	Ключи	Таблицы	Ключи
10	19480	350	12350	380
50	250190	5540	257290	4840
100	1265740	18320	1187890	18300
500	62575140	440810	66717610	431420

Объем занимаемой памяти:

Кол-во записей	Таблица	Ключи
10	1440	160
50	7200	800
100	14400	1600
500	72000	8000

Анализ полученных результатов:

Кол-во записей	Занимаемый объем памяти таблицей ключей от таблицы	Рост скорости сортировки ключей относительно таблицы (пузырек)	Рост скорость сортировки ключей относительно таблицы (шейкер)
10	~11%	В ~55 раз	В ~33 раз
50	~11%	В ~45 раз	В ~53 раз
100	~11%	В ~69 раз	В ~65 раз
500	~11%	В ~142 раз	В ~155 раз

Выводы по проделанной работе

Чем больше размер самой таблицы, тем эффективнее становится сортировка массива ключей, на малых таблицах разница не сильно заметна между сортировкой самой таблицы и таблицы ключей. Однако, для хранения таблицы ключей нужна дополнительная память. В моем случае понадобилось не так много дополнительной памяти под таблицу ключей, так как я выбрал в качестве поля сортировки минимальную стоимость (тип int), а, если, например, выбрал бы строку, длиной хотя бы 10 символов, то затраты на память сильно бы возросли. Также мы используем «запись» с вариантной частью для того, чтобы сэкономить память, так как мы можем хранить разные данные в одном участке памяти.

Контрольные вопросы

1.Как выделяется память под вариантную часть записи?

> В языке си, вариантная часть структуры реализована с помощью union. Память выделяется в одном “куске” памяти, имеющий размер, который способен вместить наибольшее поле из указанных.

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

> Поведение в данном случае не определенно (undefined behavior).

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

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

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 выбранного программистом поля исходной таблицы (в моем случае – минимальная

стоимость отдыха). Нужна она для того, чтобы не работать со всей основной таблицей при сортировках и т.д.

5.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 используемая память. А использование таблицы ключей, наоборот, эффективнее когда нужно быстрое время обработки и не так важна дополнительная используемая память. Так же, использование таблицы ключей неэффективно, когда сама таблица маленькая или имеет малое количество полей, например, таблица, имеющая два поля: “Имя” и “Почта”. В данном случае, таблица ключей будет лишь занимать дополнительное место в памяти и не даст никакой выгоды во времени.

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

Для таблиц с большим количеством строк предпочтительней использовать сортировки с наименьшим количеством перестановок, например, qsort, но так как он сам задействует дополнительную память, то можно использовать, например, шейкер или другие улучшения сортировок, не использующих много дополнительной памяти.