|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **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 ЛАБОРАТОРНОЙ РАБОТЕ 2

По дисциплине «Типы и структуре данных»

### Название Записи с вариантами, обработка таблиц

### Студент Блохин Артем Олегович

*фамилия, имя, отчество*

### Группа ИУ7-34Б

Тип лабораторной работы Учебная

### Название

предприятия

|  |  |  |
| --- | --- | --- |
| Студент |  | Блохин А.О. |
| Преподаватель | *подпись, дата* | *фамилия, и.о.*  Силантьева А.В. |
|  | *подпись, дата* | *фамилия, и.о.* |

*2022 г.*

***Содержание***

### Условие задачи 3

### Техническое задание… 3

### Описание алгоритма 4

### Анализ алгоритмы сортировки… 9

### Контрольные вопросы… 11

### Заключение… 12

***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.*

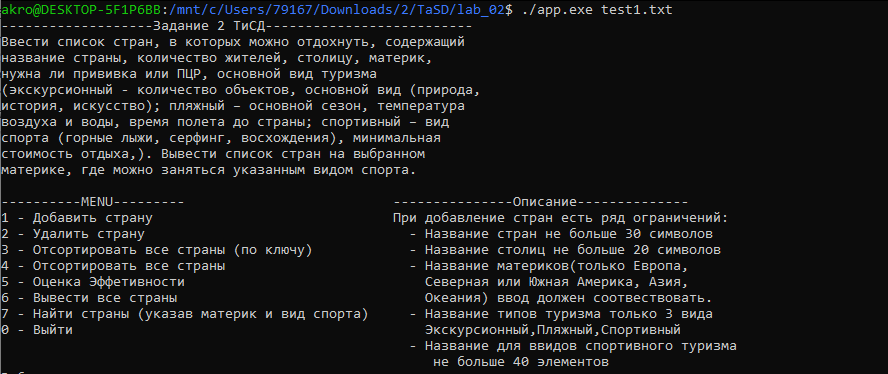
***Условия задачи***

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

*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.*

***Требование к работе с программой:***

- Взаимодействие с программой строго по меню.



***Описание алгоритма:***

#define STRING\_COUNTRY 61 // размер строки для названия страны  
#define STRING\_CAPITAL 41 // размер строки для названия столицы  
#define STRING\_MATERIC 37 // размер строки для названия материка

typedef struct  
{  
 struct main\_data\_country\_t  
 {  
 char name\_country[STRING\_COUNTRY]; // название страны  
 char capital[STRING\_CAPITAL]; // название столицы  
 char materic[STRING\_MATERIC]; // название материка  
 long int population; // число населения  
 } main\_data\_t;  
 bool covid\_tests; // значения для привики или ПЦР  
 tourism\_t tourism;   
  
} travel\_t;

*Главная структура для записи данных страны.*

typedef struct  
{  
 bool tourism\_sightseeing; // значение для экскурсионного туризма   
 bool tourism\_beach; // значение для пляжного туризма  
 bool tourism\_sport; // значение для спортивного туризма  
 union un\_kid\_toutism  
 {  
 sightseeing\_t sightseeing; // структура экскурсионного туризма  
 beach\_t beach; // структура пляжного туризма  
 sport\_t sport; // структура спортивного туризма  
 } kid\_tourism;  
} tourism\_t;

*Структура данных о туризме*

В данной структуре как раз и содержится объединение (по варианту), которое объединяет данные 3 разных видов туризма, так для одной страны вводится только основной вид туризма.

typedef struct  
{  
 unsigned int number\_objects; // количество экскурсионных объектов  
 char name\_object[STRING\_OBJECTS]; // тематика экскурсий  
} sightseeing\_t;  
typedef struct  
{  
 unsigned short main\_season; // значение для основного сезона  
 short air\_temperature; // температура воздуха  
 short water\_temperature; // температура воды   
 unsigned short time\_flying; // время полета  
} beach\_t;  
typedef struct  
{  
 char kind\_sport[STRING\_SPORT]; // вид спорта  
 unsigned long int min\_count\_price; // стоимость отдыха  
} sport\_t;

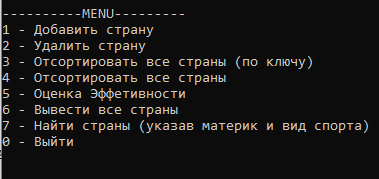
*Структуры видов туризма*

#define STRING\_COUNTRY 61 // размер строки для названия страны

typedef struct {  
 int index; // индекс из общей таблицы  
 char name\_country[STRING\_COUNTRY]; // название страны  
} key\_travel\_t;

*Структура ключей*

Программа считывает данные таблицы из файла, чье название передано через командную строку, при ошибке чтение или некорректности данных в файле не выполняет никаких действий и выводится сообщение об ошибке.



*Меню программы.*

После чего программа по выбору (то есть по использовании меню) можно выбирать различные варианты действий. Также при неправильных воздействиях на данные файла (то есть удаление или добавление), она выдает сообщение об ошибке.

Или же при неправильном вводе пункта меню сообщает пользователю, что нет такого пункта и дает возможность ввести его еще раз (пока не будет введен правильный либо пункт выхода)

При добавлении записи в таблицу, значение каждого поля новой записи должно вводиться отдельно.

Поля названий ограничиваются: СТРАНА – 30 элементов, СТОЛИЦА – 20 элементов, МАТЕРИК – 18 элементов, ВИД СПОРТА ТУРИЗМА – 100 элементов.

После ввода новая запись добавляется в конец массива и затем уже весь массив перезаписывается в файл.

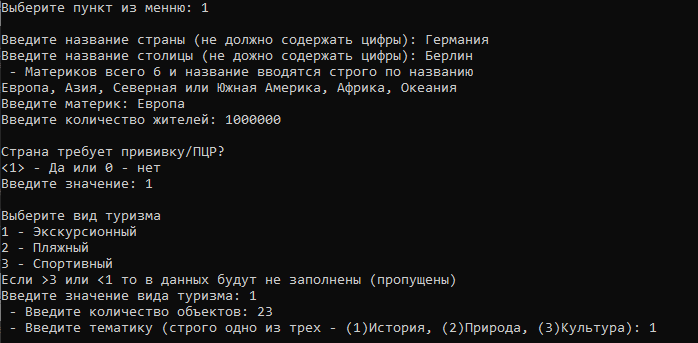
При удалении, которое осуществляется по выбору страны, в случае нахождение удаляется из массива, а затем уже файл перезаписывается. При не нахождение такой страны выдает сообщение “Такой страны нет”. При успешности «Страна была удалена»

При поиске задается МАТЕРИК и ВИД СПОРТА ТУРИЗМА, и выводит найденные страны.

При сортировке таблица выводится в удобочитаемом виде в порядке возрастания значений названий страны. Также реализована сортировка по ключу, где сортировка производится по названию страны.

Реализован вывод двух таблиц – общей и ключей

Пункт добавления



Функции добавления:

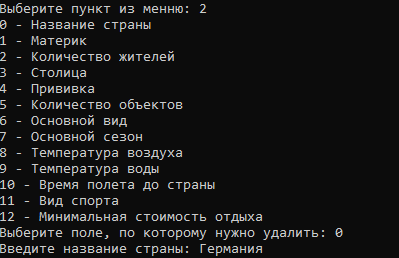
// добавления по названию спорта и материка в таблицу ключей

void fill\_key\_arr (key\_travel keys[], travel\_t countries[], size\_t len); // заполнения таблицу ключей

// добавления страны

int add\_country(travel\_t \*countries, size\_t \*n\_countries);

Пункт удаления



Функции удаления

// удаление страны

int delete\_country\_by\_name(travel\_t countries[], char str[], size\_t \*len);

int delete\_country\_by\_materic(travel\_t countries[], char str[], size\_t \*len);

int delete\_country\_by\_population(travel\_t countries[], long int population, size\_t \*len);

int delete\_country\_by\_capital(travel\_t countries[], char str[], size\_t \*len);

int delete\_country\_by\_test(travel\_t countries[], bool test, size\_t \*len);

int delete\_country\_by\_number\_objects(travel\_t countries[], unsigned int number\_objects, size\_t \*len);

int delete\_country\_by\_name\_object(travel\_t countries[], char str[], size\_t \*len);

int delete\_country\_by\_main\_season(travel\_t countries[], unsigned short main\_season, size\_t \*len);

int delete\_country\_by\_air\_temperature(travel\_t countries[], short air\_temperature, size\_t \*len);

int delete\_country\_by\_water\_temperature(travel\_t countries[], short water\_temperature, size\_t \*len);

int delete\_country\_by\_time\_flying(travel\_t countries[], unsigned short time\_flying, size\_t \*len);

int delete\_country\_by\_kind\_sport(travel\_t countries[], char str[], size\_t \*len);

int delete\_country\_by\_min\_count\_price(travel\_t countries[], unsigned long int min\_count\_price, size\_t \*len);

Функция вывода

void print\_task(void); // вывод задания  
void print\_menu(void); // вывод меню  
void print\_main\_table(); // вывод шапки таблицы  
  
void print\_travel\_sightseeing(sightseeing\_t \*country);

void print\_travel\_beach(beach\_t \*country);

void print\_travel\_sport(sport\_t \*country);  
void print\_basic\_info(travel\_t \*country);

// вывод всех стран в виде таблицы  
void print\_countries(st\_travel \*countries, size\_t n);  
// вывод таблицы ключей  
void print\_table\_by\_key(travel\_t countries[], key\_travel\_t keys[], size\_t len);

***Исходные структуры данных***.

Таблица, содержащая информацию об странах:

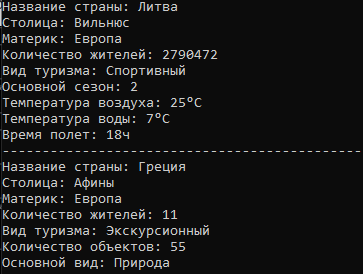
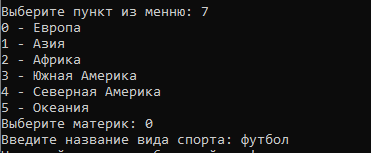


Таблица ключей

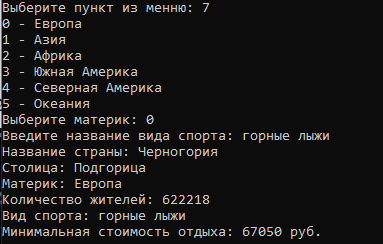
***Реализация поиска (по варианту)***

В данной работе мне требовалось реализовать поиск стран по материкам и вид спортивного туризма

1. Поиск не дал результат



2)Поиск в результате нашел искомые данные



***Анализ алгоритмов сортировки***

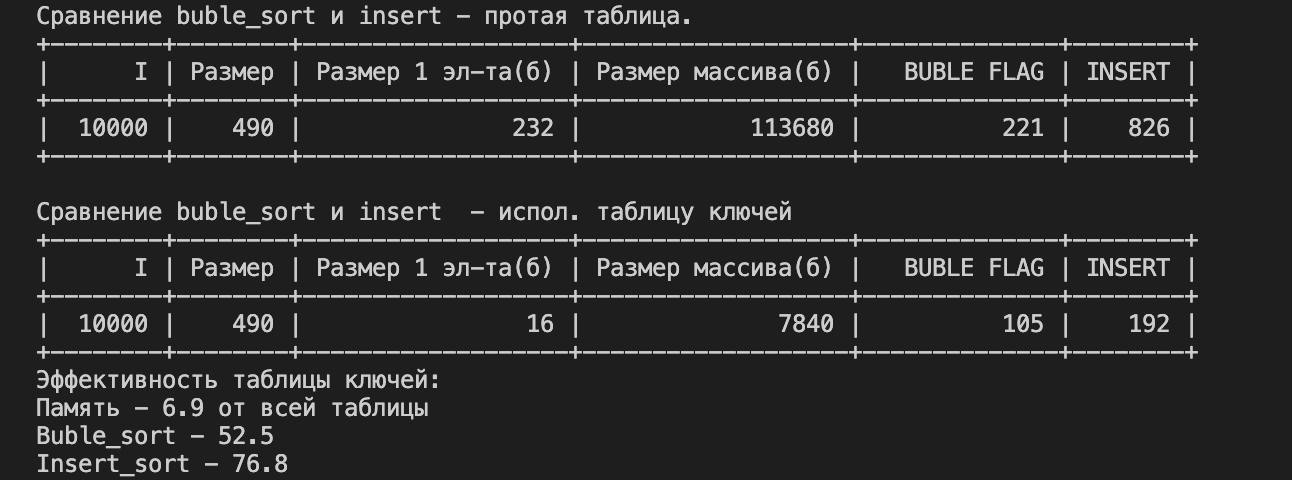
Замеры времени:

(Замеры были произведены на списках, содержащие 574 элементов, при 10000 повторении – в таблице приведены усредненные результаты)

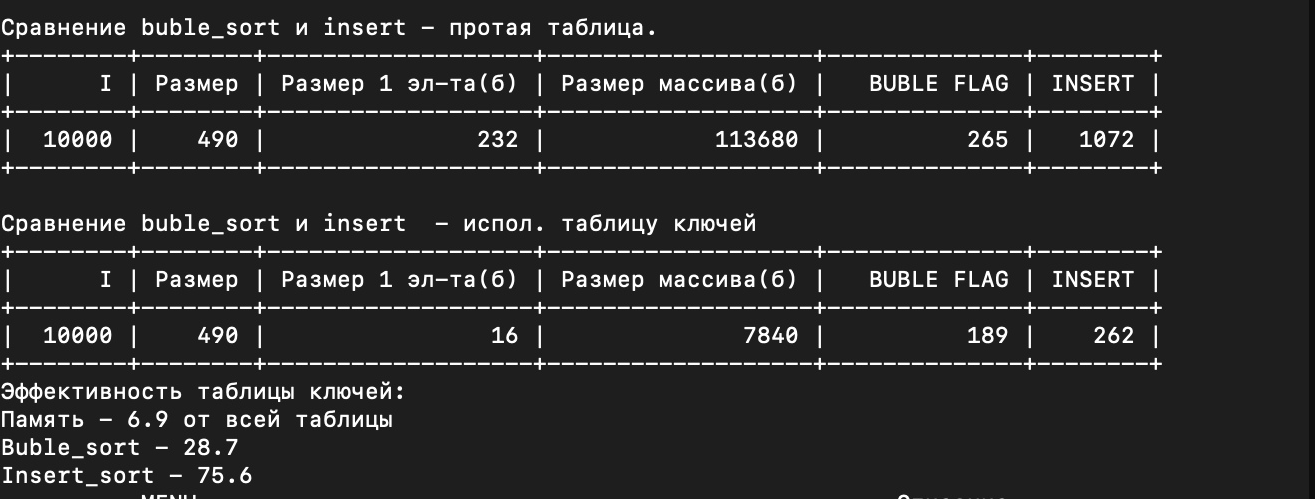
Сортировка ключей оказалось более эффективнее сортировки всей таблицы примерно на 20 - 30% по времени. (28-53% при сортировке пузырька с флажком и 65-77% при сортировки вставками)

Сама таблица ключей занимает 7% от всей таблицы, то есть по хранению двух таблиц память занимает больше места.

При использовании ключей у нас память больше занимает места, но по скорости обработки она наиболее лучше работает.



*Неотсортированные данные*



*Отсортированные данные*

***Сортировка пузырька с флажком***

void buble\_sort\_flag\_table(travel\_t \*countries, size\_t n\_countries)  
{  
 travel\_t buff;  
 bool flag = true;  
 for (size\_t i = 0; i < n\_countries; i++)  
 {  
 for (size\_t j = i; j < n\_countries - 1; j++)  
 if (countries[j].main\_data\_t. population > countries[j + 1].main\_data\_t.population)  
 {  
 buff = countries[j];  
 countries[j] = countries[j + 1];  
 countries[j + 1] = buff;  
 flag = false;  
 }  
 if (flag)  
 break;  
 }  
}

*Сортировка всей таблицы*

void buble\_sort\_flag\_key(key\_travel\_t \*key, size\_t n\_key)  
{  
 bool flag = true;  
 key\_travel\_t buff;  
 for (size\_t i = 0; i < n\_key; i++)  
 {  
 for (size\_t j = i; j < n\_key - 1; j++)  
 if (key[j].population > key[j + 1].population) {  
 buff = key[j];  
 key[j] = key[j + 1];  
 key[j + 1] = buff;  
 flag = false;  
 }  
 if (flag)  
 break;  
 }  
}

*Сортировка по ключу*

***Сортировка вставками***

void insert\_sort\_table(travel\_t \*countries, size\_t n\_countries)  
{  
 for (size\_t i = 1; i < n\_countries; i++)  
 {  
 travel\_t buff = countries[i];  
 int j = i - 1;  
 for (; j >= 0 && countries[j].main\_data\_t.population > buff.main\_data\_t.population; j--)  
 countries[j + 1] = countries[j];  
 countries[j + 1] = buff;  
 }  
}

*Сортировка всей таблицы*

void insert\_sort\_key(key\_travel\_t \*key, size\_t n\_key)  
{  
 for (size\_t i = 1; i < n\_key; i++)  
 {  
 key\_travel\_t buff = key[i];  
 int j = i - 1;  
 for (; j >= 0 && key[j].population > buff.population; j--)  
 key[j + 1] =key[j];  
 key[j + 1] = buff;  
 }  
}

*Сортировка по ключу*

***Контрольные вопросы***

*Как выделяется память под вариантную часть записи?*

В языке си вариантная часть может быть реализована с помощью объединений. Объединения могут иметь несколько полей, однако количество памяти, выделяемое под объединение, равно размеру максимального по длине полю.

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

Транслятор не контролирует тип данных, который заносится в вариантную часть, поэтому данные будут записаны, что повлечёт ошибки в дальнейшей работе программы.

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

Исключительно программист.

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

Таблица ключей - это дополнительная таблица, содержащая индекс элемента в исходной таблице и значения выбранного ключевого поля (по которому, например, будет производиться сортировка всей таблицы). Таблица ключей экономит время обработки основной таблицы.

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

Если исходная таблица небольшая и содержит малое количество записей, то эффективнее сортировать её сразу всю. Однако, если же массив данных большой, то эффективнее использовать таблицу ключей.

*Какие способы сортировки предпочтительнее для обработки таблиц и*

*почему?*

Предпочтительнее те способы сортировки, которые не переставляют уже отсортированные элементы, так как перестановка элементов таблицы занимает много времени. Например, сортировка пузырька с флагом или метод сортировки вставками. (то есть являются устойчивыми)

***Вывод***

В процессе выполнения данной лабораторной работы я ознакомилась с принципами реализации записей с вариантами и способами обработки таблиц.

В заключение можно сделать вывод, что для работы с большим количеством данных, имеющие разные вариантности лучше использовать для экономии памяти объединения. Также для обработки данных следует использовать определенные методы, например, как с сортировкой, если объем большой, то лучше использовать сортировку по ключу, что менее затратное по времени и производительности, также учитывать метод сортировки и ее устойчивость.

Сортировка ключей оказалось более эффективнее сортировки всей таблицы примерно на 30 - 60% по времени. А память при этом занимает на 7% больше обычной таблицы.