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 «ЗАПИСИ С ВАРИАНТАМИ. ОБРАБОТКА МАТРИЦ»

Студент Кузнецов Денис Евгеньевич

Группа ИУ7 – 33Б

Преподаватель

2022

Оглавление

Условие задачи. 2
Описание технического задания: 2
Выходные данные: 2
Действие программы: 2
Обращение к программе: 3
Аварийные ситуации: 3
Описание структуры данных: 3
Время сортировки таблиц: 5
Тесты: 9
Выводы: 10
Ответы на контрольные вопросы. 10
1.Как выделяется память под вариантную часть записи? 11
2.Что будет, если в вариантную часть ввести данные, несоответствующие описанным? 11
3.Кто должен следить за правильностью выполнения операций с вариантной частью записи? 11
4.Что представляет собой таблица ключей, зачем она нужна? 11
5.В каких случаях эффективнее обрабатывать данные в самой таблице, а когда – использовать таблицу ключей? 11
6.Какие способы сортировки предпочтительнее для обработки таблиц и почему? 12

Условие задачи.

Имеются описания: Type жилье = (дом, общежитие); Данные: Фамилия, имя, группа, пол (м, ж), возраст, средний балл за сессию, дата поступления адрес: дом: (улица, №дома, №кв); общежитие: (№общ., №комн.); Ввести общий список студентов. Вывести список студентов, указанного года поступления, живущих в общежитии.

Описание технического задания:

Входные данные: пользователь должен ввести целое число от 0 до 11, которое будет соответствовать определенной опции, в зависимости от выполняемой опции пользователь должен будет вводить дополнительные данные, которые будут иметь целочисленный, строковый, вещественный вид. Для каждого ввода пользователю будет выведено пояснение, в каких диапазонах должны лежать вводимые данные.

Выходные данные:

Для каждой опции вывод будет содержать сообщение об успешности или ошибке работы опции. Также выходные данные будут содержать выводы массивов структур, если это задача опции.

Действие программы:

Программа работает до тех пор, пока пользователь не введет 0, в случае, если пользователь будет вводить неверные данные, программа будет предлагать ему ввести номер опции заново, пока пользователь не введет корректные данные. При выполнении каждой из опций, если пользователю предлагается

ввести дополнительные данные, то при неверном вводе данных опция завершает свою работу и пользователю предлагается заново выбрать номер опции, которую должна выполнить программа.

Обращение к программе:

Программа запускается через терминал, затем следуя инструкциям программы пользователь должен вводить данные.

Аварийные ситуации:

```
101 - Неверно введен тип жилья
102 - Неверно введена фамилия
103 - Неверно введено имя
104 - Неверно введена группа
105 - Неверно введен пол
106 - Неверно введен возраст
107 - Неверно введен средний балл за сессию
108 - Неверно введена дата поступления
109 - Неверно введено называние улицы
110 - Неверно введен номер дома
111 - Неверно введен номер квартиры
112 - Неверно введен номер общежития
113 - Неверно введен номер комнаты
114 - Неверно введен год поступления
115 - В таблице содержится максимальное количество записей
200 - Таблица не заполнена, чтобы вывести таблицу, выгрузите данные из файла
1 - Введено неверное имя файла
```

Описание структуры данных:

```
typedef struct keys_t
{
    int age;
    int index;
} keys_t;
```

```
typedef struct house_t
{
    char street[MAX_SIZE_STREET];
    int number_house;
    int number_flat;
} house_t;

typedef struct dorm_t
{
    int number_dorm;
    int number_room;
} dorm_t;

typedef union choose_type_t
{
    house_t house;
    dorm_t dorm;
} choose_type_t;

typedef struct describtion_t
{
    char type_house[MAX_SIZE_TYPE_HOUSE];
    char surname[MAX_SIZE_SURNAME];
    char name[MAX_SIZE_NAME];
    int group;
    char gender[MAX_SIZE_GENDER];
    int age;
    double arithmetic_mean;
    char entry_day[MAX_SIZE_ENTRY_DAY];
    choose_type_t home;
} describtion_t;
```

Структура keys_t – структура, предназначенная для хранения ключа и соответствующего ему индекса в массиве. Поле index отвечает за позицию ключа в массиве, age – хранит значение ключа массива, в данном случае – целочисленный тип данных, описывающий возраст человека.

Структура dorm_t предназначена для описания типа данных, который хранит номер общежития и номер комнаты. Поля number_dorm и number_room хранят целочисленные типы данных.

Структура house_t нужна для описания данных дома, эти данные включают в себя называние улицы, номер дома и номер квартиры. Называние улицы

хранится в массиве типа char, номер дома и квартиры – целочисленные типы данных.

Объединение choose_t предназначено для описания нескольких структур данных, для того чтобы в памяти сэкономить место описание дома или общежития хранится в одной области памяти, так как мы храним данные либо о доме, либо об общежитии, то нет смысла выделять для этого разные участки памяти.

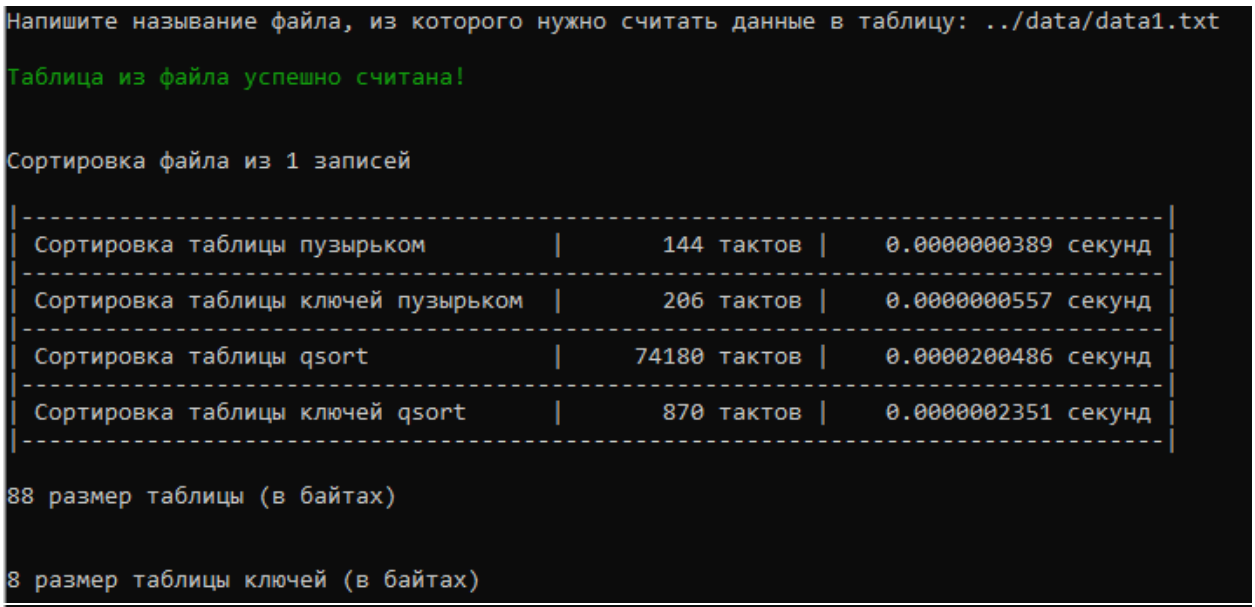
Структура describtion_t описывает тип данных, описанный в задаче, данный тип данных содержит в себе тип жилья, фамилию, имя, группу, пол, возраст, средний балл за сессию, дату поступления и объединение, которое описывает один из типов жилья.

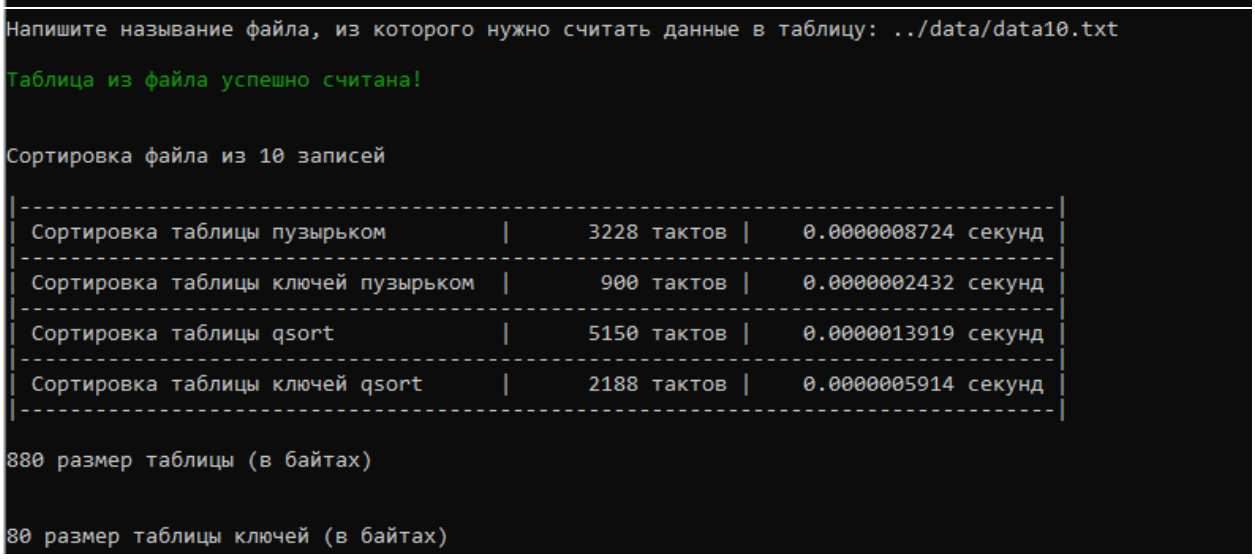
```
#define MAX_SIZE_ARRAY              500
#define MAX_SIZE_SURNAME            15
#define MAX_SIZE_NAME               15
#define MAX_SIZE_TYPE_HOUSE         6
#define MAX_SIZE_GENDER             1
#define MAX_SIZE_ENTRY_DAY          10
#define MAX_SIZE_STREET             12
#define MAX_SIZE_FILENAME           30
```

В рамках данной реализации выделяются вышеописанные размеры массивов. Максимальное количество записей, которое может обрабатывать программа – 500, максимальная длина фамилии и имени – 15 символов, для хранения типа жилья отведено 6 символов, пол хранится в виде 1 символа. Дата поступления хранится в виде строки записанной виде dd.mm.yyyy, для этого выделено 10 символов, максимальная длина улицы может составлять 12 символов и максимальная длина имени файла – 30 символов.

Время сортировки таблиц:

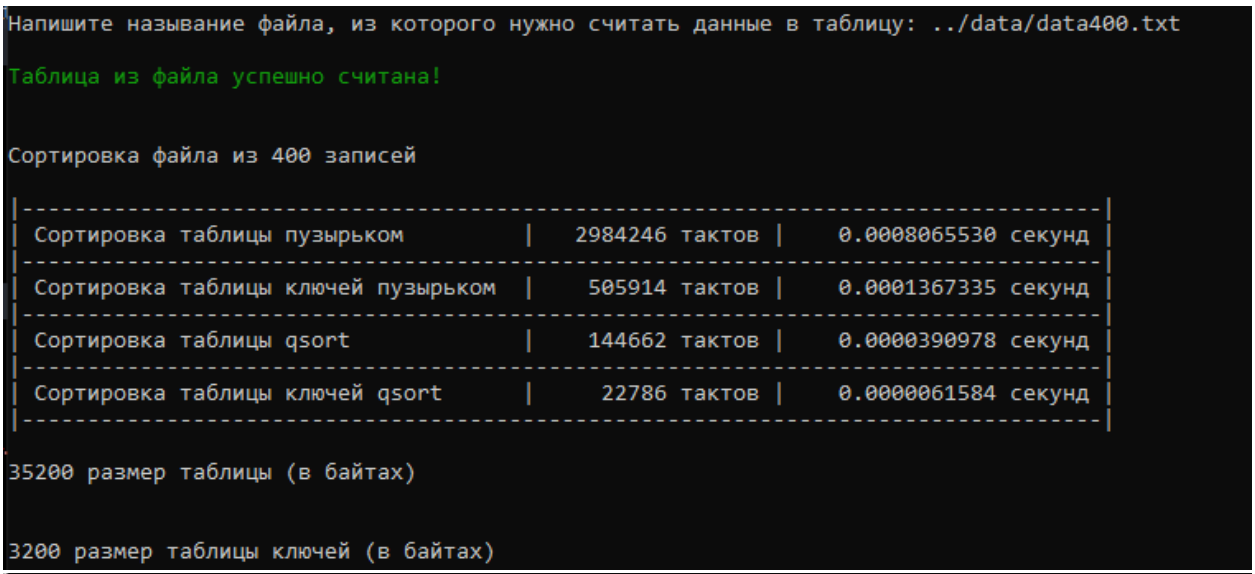
Ниже представлены скрины с зависимостью времени сортировки от выбранной сортировки, данная программа была протестирована на таблицах с количеством записей: 1, 10, 100, 250, 400, 500.

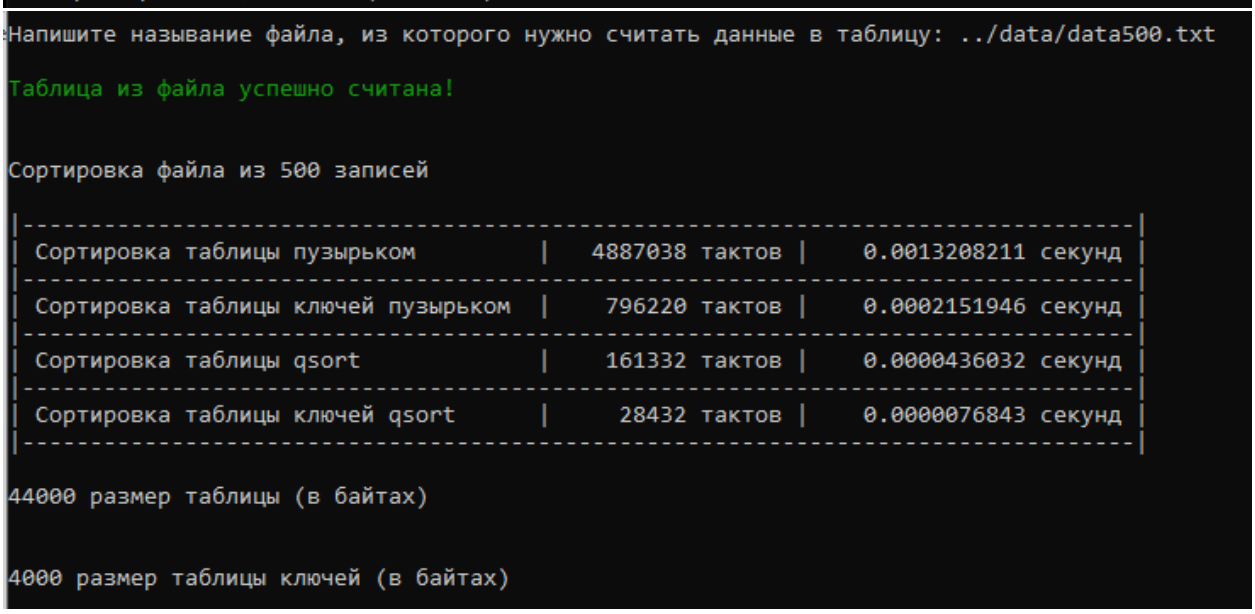




```
Напишите называние файла, из которого нужно считать данные в таблицу: ../data/data100.txt

Таблица из файла успешно считана!


Сортировка файла из 100 записей

|-----------------------------------------------------------------------------|
| Сортировка таблицы пузырьком         |    194922 тактов |    0.0000526816 секунд |
|-----------------------------------------------------------------------------|
| Сортировка таблицы ключей пузырьком  |     34332 тактов |    0.0000092789 секунд |
|-----------------------------------------------------------------------------|
| Сортировка таблицы qsort             |    114884 тактов |    0.0000310497 секунд |
|-----------------------------------------------------------------------------|
| Сортировка таблицы ключей qsort      |      7378 тактов |    0.0000019941 секунд |
|-----------------------------------------------------------------------------|

8800 размер таблицы (в байтах)


800 размер таблицы ключей (в байтах)
```

```
Напишите называние файла, из которого нужно считать данные в таблицу: ../data/data250.txt

Таблица из файла успешно считана!


Сортировка файла из 250 записей

|-----------------------------------------------------------------------------|
| Сортировка таблицы пузырьком         |   1287946 тактов |    0.0003480935 секунд |
|-----------------------------------------------------------------------------|
| Сортировка таблицы ключей пузырьком  |    355096 тактов |    0.0000959719 секунд |
|-----------------------------------------------------------------------------|
| Сортировка таблицы qsort             |     86476 тактов |    0.0000233719 секунд |
|-----------------------------------------------------------------------------|
| Сортировка таблицы ключей qsort      |     15716 тактов |    0.0000042476 секунд |
|-----------------------------------------------------------------------------|

22000 размер таблицы (в байтах)


2000 размер таблицы ключей (в байтах)
```





Из выходных данных получаем, что размер массива ключей равен 9% от размера исходного массива. Т.е. используя массив ключей в данном случае, нагрузка на память возрастает несущественно.

Для таблицы, которая содержит одну запись, выходные данные отличаются от остальных таблиц. В связи с тем, что в сортировке qsort происходит больше вызовов функции, поэтому время на выполнения алгоритма увеличивается. Для таблицы состоящих из массива ключей и содержащей 1 запись, время работы qsort составляет 422% относительно времени работы bubble_sort, в случае, если сортируется обычный массив, то время работы qsort составляет 48736%. Т.е. при работе с небольшим количеством данных сортировка qsort менее эффективная, чем bubble_sort.

На больших данных сортировка qsort показывает лучшее время работы, относительно bubble_sort

Таблица 1

Количество записей	Время работы bubble_sort для массива ключей Относительно qsort для массива ключей	Время работы bubble_sort для обычного массива относительно qsort для обычного массива	Время работы qsort для массива ключей	Время работы qsort для обычного массива
10	43%	63%	100%	100%
100	466%	170%	100%	100%
250	2366%	1489%	100%	100%
400	2200%	2072%	100%	100%
500	2804%	3029%	100%	100%

Тесты:

Таблица 2

Описание теста	Входные данные	Вывод
Ввод пункта меню	-1	Вы ввели неверный номер действия, введите заново опцию, которую нужно выполнить, программа завершится аварийно!
Ввод пункта меню	13	Вы ввели неверный номер действия, введите заново опцию, которую нужно выполнить, программа завершится аварийно!
Ввод пункта меню	abc	Вы ввели неверный номер действия, введите заново опцию, которую нужно выполнить, программа завершится аварийно!
Загрузка таблицы	Кол-во студентов больше 500 или меньше 1	При выполнении опции произошла ошибка!

Добавление записи в таблицу	В случае, если одно из полей не соответствует требованиям ввода	При выполнении опции произошла ошибка!
Вывод таблицы	В случае, если таблица еще не была считана из файла	При выполнении опции произошла ошибка!
Вывод студентов из общежития, указанного года поступления	В случае, если студентов указанного года поступления из общежития нет	В таблице нет студентов, живущих в общежитии … года поступления!
Сортировка таблиц	В случае, если таблица не была считана из файла	Таблица не заполнена, чтобы вывести таблицу, выгрузите данные из файла!!!
Загрузка таблицы из файла	Количество студентов от 1 до 500	Таблица из файла успешно считана!
Вывод времени сортировок таблиц	В случае, если таблица не была считана из файла	Введено неверное имя файла! При выполнении опции произошла ошибка!
Вывод времени сортировок таблиц	В случае, если таблица была считана из файла	Выводится таблица, в которой указана сортировка и время ее работы.
Удаление записи из таблицы по номеру группы	Если во время считывания номера группы ошибок не произошло	Записи успешно удалены из таблицы!

Выводы:

Из таблицы 1 видно, что массив ключей сортируется в разы быстрее, чем обычный массив, хотя количество дополнительных данных не сильно увеличивает затраты по памяти в относительных единицах. Поэтому алгоритм qsort для массива ключей является наиболее эффективным, за исключением тех случаев, когда сортируются совсем маленькие таблицы.

Ответы на контрольные вопросы.

1.Как выделяется память под вариантную часть записи?

Вариативная часть в си образована с помощью union. Чтобы узнать размер union нужно узнать, какое количество памяти выделяется под поле, которое весит больше всего, именно столько памяти будет выделено под вариативную часть.

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

В случае, если программист не проверяется правильность входных данных, то поведение программы будет непредсказуемым. Но программист должен сам проверять, что вводятся именно те данные, которые ему нужны, поэтому проверка данных лежит полностью на программисте.

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

За правильностью выполнения операции с вариантной частью должен следить программист, так как программист задает типы данных для вариантной части, поэтому перед выполнением определенных операции над переменным union, он должен следить за тем, что в union лежат нужные программисту данные.

4.Что представляет собой таблица ключей, зачем она нужна?

Таблица ключей – таблица, в которой содержаться только значения одного из полей и соответствующий индекс этого поля в массиве.

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

Таблица ключей эффективна в том случае, если имеется большая таблица, и чтобы в памяти не перемещать такие большие объемы памяти используются таблицы ключей, которые позволяют программе задействовать меньше ресурсов памяти, а значит работать быстрее. Но в случае, если нам важно

количество выделенной под программу памяти, а не время ее работы, то в этом случае таблица ключей не будет эффективна.

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

Для таблиц, состоящих из большого количества полей эффективнее использовать таблицу ключей, потому что таким образом программа будет обрабатывать меньшее количество памяти. Если в таблице большое количество записей, то эффективнее будет использовать быстрее сортировки, например, qsort, а если в таблице значений не очень много, то предпочтительнее использовать простые алгоритмы сортировки, например пузырьковую сортировку.