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»

Студент: Лысцев Никита Дмитриевич

Группа: ИУ7-33Б

Студент _____ Лысцев Н.Д

подпись, дата *фамилия,и.о.*

Преподаватель _____ Барышникова М. Ю

подпись, дата *фамилия,и.о.*

Оценка _____

2022 г

Оглавление

1. Описание условия 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 %).

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.

2. Описание технического задания

2.1. Входные данные

Массив структур. Каждый элемент массива содержит информацию о студенте.

2.2. Выходные данные

В зависимости от выбора пользователя либо выход из программы, либо выполнение одной из предложенной по меню взаимодействия возможности.

2.3. Описание задачи, реализуемой программой

Задача программы – с помощью меню взаимодействия осуществить обработку таблицы, содержащей записи о студентах.

2.4. Способ обращение к программе

Ввод и вывод всех данных осуществляется через консоль.

Также существует возможность получить большое количество данных из текстового файла (порядка 10 тыс. записей).

2.5. Описание возможных аварийных ситуаций и ошибок пользователя

Аварийные ситуации:

- Пустое поле ввода – ничего не произойдет, программа будет ждать ввода пользователя;
- Переполнение таблицы при попытке добавления новых данных.

Ошибки пользователя:

- Некорректный ввод: превышение допустимой длины названия улицы, фамилии или имени студента, выход за поставленный диапазон числовых данных, ввод не валидных данных.

3. Описание внутренних структур данных

Информация об одном студенте хранится в структуре типа «student_t».

```
typedef struct student_t
{
    int flag;
    char surname[SURNAME_SIZE];
    char name[NAME_SIZE];
    int study_group;
    int gender;
    int age;
    int mark;
    int year_adm;
    house_t house;
} student_t;
```

Листинг 1. Структура типа «student_t»

Поля структуры «student_t»:

1. flag – целочисленная переменная, определяющая место проживания студента (0 – общежитие, 1 – дом).
2. surname – массив символов, хранящий в себе фамилию студента. Длина массива SURNAME_SIZE = 31 символ.
3. name – массив символов, хранящий в себе имя студента. Длина массива NAME_SIZE = 31 символ.
4. study_group – целочисленная переменная, определяющая номер учебной группы студента (принимает значения от 1 до 6).
5. gender -- целочисленная переменная, определяющая пол студента (0 – женский, 1 – мужской).
6. age -- целочисленная переменная, хранящая в себе возраст студента (принимает значения от 18 до 26).
7. mark -- целочисленная переменная, хранящая в себе средний балл за сессию у студента (принимает значения от 60 до 100).
8. year_adm -- целочисленная переменная, хранящая в себе год поступления студента (принимает значения от 2015 до 2022).
9. house_t house – объединение, вариантное поле. Содержит в себе информацию о месте проживания студента (либо общежитие, либо дом).

```
typedef union
{
    home_t home;
    dorm_t dorm;
} house_t;
```

Листинг 2. Объединение «house_t»

Объединение содержит в себе одно из двух полей «home_t» и «dorm_t» (дом и общежитие соответственно):

1. home – структура, содержащая в себе информацию о доме:
   - street – массив символов, содержащий в себе название улицы, на которой находится дом. Длина массива STREET_SIZE = 31 символ.

- num_home – целочисленная переменная, хранящая в себе номер дома, в котором проживает студент (значение > 0).
- num_flat -- целочисленная переменная, хранящая в себе номер квартиры, в которой проживает студент (значение > 0).

```
typedef struct
{
    char street[STREET_SIZE];
    int num_home;
    int num_flat;
} home_t;
```

Листинг 3. Структура типа «home_t»

2. «dorm_t» -- структура, содержащая в себе информацию об общежитии:
   - num_dorm -- целочисленная переменная, хранящая в себе номер общежития, в котором проживает студент (значение > 0).
   - num_room – целочисленная переменная, хранящая в себе номер комнаты, в которой проживает студент (значение > 0).

```
typedef struct
{
    int num_dorm;
    int num_room;
} dorm_t;
```

Листинг 4. Структура типа «dorm_t»

Структура «key_t», где содержится ключ (значение среднего балла за сессию):

```
typedef struct
{
    int mark;
    int index_mark;
} key_t;
```

Листинг 5. Структура «key_t»

Поля структуры «key_t»:

1. mark – целочисленная переменная, хранящая в себе средний балл за сессию у студента (принимает значения от 60 до 100).

2. index_matrix – целочисленная переменная, хранящая в себе индекс элемента из исходной таблицы с данными о студентах.

3.1. Описание алгоритма

Взаимодействие пользователя с программой осуществляется через консоль с помощью специального меню, в котором пользователю предлагается выполнить то, или иной действие.

В начале меню в одном предложении кратко изложена суть программы. Далее в пунктах с 1 по 9 кратко изложены возможности программы. Пункт с номером 0 служит для принудительного завершения работы с программой.

```
Программа для обработки данных о студентах.
1 - Загрузить данные из файла
2 - Вывести на экран общий список студентов
3 - Вывести на экран отсортированную таблицу ключей
4 - Вывети на экран список студентов указанного года поступления,
    живущих в общежитии
5 - Упорядочить и вывести данные по среднему баллу за сессию,
    используя саму таблицу
6 - Упорядочить и вывести данные по среднему баллу за сессию,
    используя массив ключей
7 - Добавить данные в таблицу
8 - Удалить записи по среднему баллу за сессию
9 - Вывести результаты сравнения эффективности программы
    при обработке таблицы и массива ключей
0 - Выйти из программы
Выберите пункт меню:
```

Листинг 6. Меню взаимодействия

3.2. Ограничения на входные данные

Имя, фамилия, улица – все эти значения состоят максимум из одного слова, длина каждого элемента не больше 30 символов.

Номер дома, номер квартиры, номер комнаты в общежитии, номер общежития – это целые положительные значения.

Флаг, определяющий место проживания студента, пол студента – целочисленные переменные, принимающие всего два значения – либо 0, либо 1.

Возраст студента – целое число в диапазоне от 18 до 26 лет. Средний балл за сессию – целое число в диапазоне от 60 до 100 баллов.

Максимальный размер массива структур, т.е. таблицы – 10100 элементов.

Невозможно удалить данные из пустого массива, или же добавить данные в уже полностью заполненную таблицу.

В случае ввода некорректных данных программа завершает свою работу. Пустой массив структур или же полностью заполненная таблица не являются ошибкой.

4. Вывод результатов использования различных алгоритмов сортировок

Время сортировки:

Количество записей	**Сортировка пузырьком**		**Быстрая сортировка**	
	Исходная таблица, мс	Таблица ключей, мс	Исходная таблица, мс	Таблица ключей, мс
1000	28	2	0	0
5000	420	68	1	0
10000	1689	288	2	0

Объем занимаемой памяти:

Количество записей	**Исходная таблица, байты**	**Таблица ключей, байты**
1000	128000	8000
5000	640000	40000

10000	1280000	80000

Сравнение эффективности алгоритмов сортировки:

Количество записей	**% занимаемой памяти таблицей ключей от исходной таблицы**	**Эффективность сортировки пузырьком по отношению к быстрой сортировке в %**		**На сколько % быстрая сортировка эффективнее сортировки пузырьком**	
		Исходная таблица	Таблица ключей	Исходная таблица	Таблица ключей
1000	6.25	0	0	100	100
5000	6.25	0.23	0	99.77	100
10000	6.25	0.06	0.36	99.94	99.64

Из полученных данных видно, что при любом количестве записей выгоднее по времени сортировать массив ключей, но при не слишком большом количестве данных разница во времени при сортировке исходной таблицы и таблицы ключей не столь существенна, но при этом дополнительно выделяется 6.25 процентов памяти от исходной таблицы для хранения таблицы ключей.

Также видно, что быстрая сортировка «qsort» намного быстрее, чем сортировка пузырьком на любом объеме данных.

5. Выводы

При работе с любым количеством данных выгоднее использовать сортировку дополнительного массива ключей, так как такой массив сортируется в разы быстрее, чем таблица с исходными данными. Но при этом выделяется дополнительная память для хранения этого массива ключей. Это будет несущественно при большом количестве записей, но будет не эффективно при малом их количестве, поскольку разница во времени сортировки исходной таблицы и сортировки таблицы ключей при

малом количестве не столь существенна, чем при сортировке таблиц с множеством данных.

6. Набор тестов

№ Теста	Входные данные	Выходные данные	Результат
01	Пункт меню - 0	Завершение работы программы	Нулевой код возврата
02	Пункт меню - 1	Загрузка данных из файла	Сообщение: Данные были успешно загружены в таблицу. Ожидание следующего действия.
03	Пункт меню - 2	Вывод считанной таблицы на экран	Ожидание следующего действия.
04	Пункт меню - 3	Вывод отсортированной таблицы ключей на экран	Ожидание следующего действия.
05	Пункт меню – 4 Ввод валидного года поступления	Вывод на экран студентов с введенным годом поступления, живущих в общежитии. Либо сообщение о том, что таких студентов в таблице нет	Ожидание следующего действия.
06	Пункт меню – 5	Вывод упорядоченной таблицы по среднему баллу за сессию на экран, сортируя саму таблицу	Ожидание следующего действия.
07	Пункт меню – 6	Вывод упорядоченной таблицы по среднему баллу за сессию на экран, используя массив ключей	Ожидание следующего действия.
08	Пункт меню – 7 Ввод валидных данных о студенте	Добавление новых данных в таблицу. Если таблица переполнена, то вывод соответствующего сообщения.	Ожидание следующего действия.
09	Пункт меню – 8 Ввод валидного среднего балла за сессию	Удаление студентов с введенным баллом за сессию из таблицы. Если таких студентов нет, то выводится соответствующее сообщение	Ожидание следующего действия.
10	Пункт меню – 9	Вывод результатов эффективности программы при различных алгоритмах сортировки	Ожидание следующего действия.
11	Неверный пункт меню	Сообщение о том, что пункт меню некорректный	Завершение программы с ненулевым кодом возврата.
12	Пункт меню – 7 Ввод любых не валидных данных о студенте	Сообщение о том, что были введены некорректные данные	Завершение программы с ненулевым кодом возврата.
13	Пункт меню – 8 Ввод не валидного среднего балла за сессию	Сообщение о том, что введенный балл некорректный	Завершение программы с ненулевым кодом возврата.
14	Пункт меню – 4 Ввод не валидного года поступления	Сообщение о том, что введенный год поступления некорректный	Завершение программы с ненулевым кодом возврата.
15	Пункт меню – 1 Неверный файл с данными	Сообщение об ошибке чтения данных из файла	Завершение программы с ненулевым кодом возврата.
16	Пустая таблица. Все пункты, кроме 7 и 0	Сообщение о том, что таблица пустая	Ожидание следующего действия.

17	Пункт меню – 7 Таблица изначально заполнена максимально возможным количеством записей.	Сообщение о том, что таблица переполнена.	Завершение программы с ненулевым кодом возврата.

7. Ответы на контрольные вопросы

7.1. Как выделяется память под вариантную часть записи?

Выделяется область памяти, равная размеру максимального по длине поля вариантной части.

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

Тип данных в вариантной части при компиляции не проверяется, возможно неопределенное поведение. Ввиду этого все проверки необходимо осуществлять самостоятельно.

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

За правильностью выполнения операций с вариантной частью должен следить сам разработчик.

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

Таблица ключей содержит индекс элемента в исходной таблице и выбранный ключ. За счет небольших дополнительных затрат памяти позволяет ускорить процесс поиска и сортировки элементов исходной таблицы.

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

Использовать таблицу ключей эффективнее в случае большого количества записей или большого размера памяти, необходимой для хранения каждой записи.

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

Для сортировки небольших таблиц можно использовать простые алгоритмы сортировки, такие как сортировка пузырьком, вставками. Если же в таблице много различных полей, количество элементов в таблице также велико, то предпочтительнее было бы использование устойчивых алгоритмов сортировки с квадратичной сложностью или O(n*logn).