Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

	***«Московский государственный технический университет имени Н.Э. Баумана» (МГТУ им. Н.Э. Баумана)***

ФАКУЛЬТЕТ _____ИУ_____

КАФЕДРА _____ИУ7_____

Отчет

по лабораторной работе № 2

Дисциплина: Типы и Структуры Данных

Название лабораторной работы: Обработка больших чисел

Студент гр. **ИУ7-34Б Малышев Илья Николаевич**

(Подпись, дата) (И.О. Фамилия)

Преподаватель **Силантьева Александра Васильевна**

(Подпись, дата) (И.О. Фамилия)

Москва, 2021

Условия задачи

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

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.

Техническое задание

Входные данные:

- Текстовый файл со списком объектов (“студентов”), каждое поле структуры находится на отдельной строке;
- Целое число, обозначающее действие, которое должна выполнить программа (смотри “Задача программы”);
- Запрашиваемые данные, необходимые для выполнения некоторых действий.

Выходные данные:

- Полученная таблица (выводится путем прямого обращения или с помощью таблицы ключей);
- Сравнение разных типов сортировок;
- Результаты поиска по таблице.

Задача программы:

Программа позволяет выполнить операции, которые перечислены ниже:

1. Загрузить список студентов из файла;
2. Добавить студента в конец списка;

3. Удалить студентов по году поступления;
4. Вывести список студентов указанного года поступления, живущих в общежитии;
5. Отсортировать таблицу методом простых вставок (по году поступления);
6. Отсортировать таблицу ключей методом простых вставок (по году поступления);
7. Отсортировать таблицу с помощью qsort (по году поступления);
8. Отсортировать таблицу ключей с помощью qsort (по году поступления);
9. Вывести сравнение сортировок;
10. Вывести таблицу;
11. Вывести таблицу по таблице ключей;
12. Выход.

Способ обращения к программе:

Программа является консольной. После запуска исполняемого файла из консоли, программа выведет справку и приглашение на ввод.

Аварийные ситуации:

- Некорректный ввод номера команды;
- Некорректный ввод строки с именем файла или файл пуст;
- Превышение количества записей в конечной таблице;
- Неверный ввод строкового поля;
- Ввод недопустимого признака поля;
- Некорректный ввод вариантного поля структуры.

Структуры данных:

house_t - абстракция квартиры, в данном случае, адреса квартиры. Содержит структуру строка (хранит название улицы) и два

целочисленных поля (хранят номер дома и номер квартиры соответственно)

```c
#define MAX_LEN 128

typedef struct {
    char street[MAX_LEN];
    unsigned int house_number;
    unsigned int apartment_number;
} house_t;
```

hostel_t - абстракция комнаты общежития, точнее адреса комнаты. Содержит два беззнаковых целочисленных поля: номер общежития и номер комнаты в общежитии.

```c
typedef struct {
    unsigned int hostel_number;
    unsigned int room_number;
} hostel_t;
```

Строка, как массив символов, заканчивающийся нулем, представляет собой абстракцию текстового поля фиксированной длины. Содержит MAX_LEN элементов типа char.

```c
char street[MAX_LEN];
```

```c
#define MAX_LEN 128
```

student_t - является абстракцией студента, хранящей разносторонние данные о нем: Фамилию, имя, название группы в виде структуры строки, пол и тип жилья, как “логические” флаги, возраст и год поступления, как беззнаковое целое, средний балл, как число с плавающей точкой и объединение двух полей типа **hostel_t** и **house_t** зависящие от типа жилья.

```
typedef struct {
    char surname[MAX_LEN];
    char name[MAX_LEN];
    char group[MAX_LEN];
    int sex;
    unsigned int age;
    double gpa;
    unsigned int entrance_year;
    int residence_type;
    union {
        house_t house;
        hostel_t hostel;
    } residence;
} student_t;
```

key_student_t - это структура, которая хранит в себе данные о годе поступления студента и индекс студента в исходной таблице.

```
typedef struct
{
    size_t index;
    unsigned int entrance_year;
} key_student_t;
```

Алгоритм:

1. Пользователь вводит номер команды из меню.

2. Пока пользователь не введет 0 (выход из программы) или не произойдет аварийная остановка из-за ошибки, ему будет предложено выполнять действия с таблицей.

Тесты:

№	Ошибка	Пользовательский ввод
1	Некорректный ввод команды	999
2	Некорректный ввод имени файла	joke.txt
3	Добавление записи при максимальном размере таблицы.	Ввод записи
4	Некорректный ввод строкового поля	Символ не из кодировки ANCII
5	Некорректный ввод целочисленного поля	asd
6	Ввод недопустимого значения	При указании пола введена “3”
7	Ввод записи в таблицу (успешно)	Корректный ввод
8	Удаление из таблицы записи по ключу (успешно)	Корректный ввод даты
9	Корректный ввод № действия	Введено целочисленное число в пределах [0,11]

Оценка эффективности:

Измерения эффективности сортировок будут производиться в единицах измерения – секундах.

Количество записей	Сортировка вставками		Быстрая сортировка	
	Исходная таблица	Таблица ключей	Исходная таблица	Таблица ключей
50	0	0	0	0
200	0.12	0	0	0
800	0.258	0.0064	0	0
3200	3.089	0.073	0.004	0
12800	58.912	1.25	0.004	0.002

Время сортировки*

Объём занимаемой памяти (в байтах):

Количество записей	Исходная таблица	Таблица ключей
50	27200	800
200	108800	3200
800	435200	12800
3200	1740800	51200
12800	6963200	204800

Контрольные вопросы

1. *Как выделяется память под вариантную часть записи?*

Размер памяти, выделяемый под вариантную часть, равен максимальному по длине полю вариантной части. Эта память является общей для всех полей вариантной части записи.

2. *Что будет, если в вариантную часть ввести данные, не соответсвтующие описанным?*

Тип данных в вариантной части при компиляции не проверяется. Из-за того, что невозможно корректно прочитать данные, поведение будет неопределенным.

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

Контроль за правильностью выполнения операций с вариантной частью записи возлагается на программиста.

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

Дополнительный массив (структура), содержащий индекс элемента в исходной таблице и выбранный ключ. Она нужна для оптимизации сортировки.

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

В случае, если мы сортируем таблицу ключей, мы экономим время, так как перестановка записей в исходной таблице, которая может содержать большое количество полей, отсутствует. С другой стороны, для размещения таблицы ключей требуется дополнительная память. Кроме того, если в качестве ключа используется символьное поле записи, то для сортировки таблицы ключей необходимо дополнительно обрабатывать данное поле в

цикле, следовательно, увеличивается время выполнения. Выбор данных из основной таблицы в порядке, определенном таблицей ключей, замедляет вывод. Если исходная таблица содержит небольшое число полей, то выгоднее обрабатывать данные в самой таблице.

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

Если обработка данных производится в таблице, то необходимо использовать алгоритмы сортировки, требующие наименьшее количество операций перестановки. Если сортировка производится по таблице ключей, эффективнее использовать сортировки с наименьшей сложностью работы.

Вывод

В моей реализации программы для хранения таблицы ключей необходимо около 3% от количества памяти, необходимой для хранения всей таблицы. Как видно из таблицы замеров времени, различие в скорости сортировки для метода вставок можно оценить при объеме таблицы 800 элементов и более. При этом время сортировки всей таблицы превышает время сортировки таблицы ключей в 40 раз при 800 элементах в таблице (при увеличении количества элементов в таблице в 4 раза, это отношение увеличивается примерно в 1.05). Из этого можно сделать вывод, что при сортировке со сложностью O(n*n) имеет смысл генерировать таблицу ключей уже при объеме исходной таблицы в 800 элементов, так скорость выполнения увеличится в 40 раз, если не учитывать время генерации таблицы ключей, а объем памяти, выделенной для хранения информации увеличится всего в 1.03 раза. Относительно “быстрой сортировки” (o(n*ln(n)) можно сделать вывод, что генерация вспомогательной таблицы ключей имеет смысл при объеме таблицы 12800 элементов и более. При этом размер выделенной памяти при генерации таблицы ключей увеличится всего на 3%, а скорость выполнения сортировки увеличится в 2 раза.