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 Дубов Андрей Игоревич
фамилия, имя, отчество
Группа ИУ7-33Б

Тип лабораторной работы Учебная

Студент _____ Дубов А. И.
подпись, дата *фамилия, и.о.*

Преподаватель _____ Барышникова М. Ю.
подпись, дата *фамилия, и.о.*

2022 г.

Оглавление

Условия задачи

Имеются описания: 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

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 %).

Входные данные

Пункт меню (число от 0 до 14), файл с записями квартир.

Выходные данные

Текущее состояние таблицы, результаты поиска по заданным полям, результаты сортировок

Возможные аварийные ситуации

Некорректный ввод данных

```
struct date_t
{
    int day;
    int month;
    int year;
};

struct house_t
{
    char street[MAX_LEN_OF_STR * 2 + 2];
    int num_house;
    int num_flat;
};

struct hostel_t
{
    int num_hostel;
    int num_room;
```

```
};

union residence_t
{
    house_t house;
    hostel_t hostel;
};

struct student_t
{
    char surname[MAX_LEN_OF_STR+2];
    char name[MAX_LEN_OF_STR+2];
    int sex;
    int group;
    int age;
    double average;
    date_t date;
    int where;
    residence_t residence;
};

struct keystudent_t
{
    int id;
    int age;
};
```

Замеры времени и памяти

В таблице 40 записей. Измерения проводились 10 000 раз. Взяты средние значения. Массив ключей занимает 320 байт, основная таблица занимает 10240 байт

Сортировка пузырьком, миллисек.		Сортировка QuickSort, миллисек.	
Таблица	Таблица ключей	Таблица	Таблица ключей
9.6984	3.5055	1.9102	1.5254

	Таблица ключей в памяти	Время сортировки таблицы (пузырек)	Время сортировки таблицы (QuickSort)
За 100% принят	Массив таблицы в памяти	Время сортировки массива ключей (пузырек)	Время сортировки таблицы ключей (QuickSort)
	~3%	~277%	~125%

	Время сортировки таблицы (пузырек)	Время сортировки таблицы (пузырек)	Время сортировки таблицы ключей (пузырек)
За 100% принят	Время сортировки таблицы ключей (QuickSort)	Время сортировки таблицы (QuickSort)	Время сортировки таблицы ключей (QuickSort)
	~636%	~508%	~230%

Выводы по проделанной работе

QuickSort быстрее в случаях когда сортируешь один и тот же массив, но сортировка массива ключей намного быстрее засчёт меньшего размера массива в памяти. Массив ключей занимает всего на 3% больше, поэтому его можно спокойно использовать для сортировки, даже большого количества строк в таблице.

Контрольные вопросы

1. Как выделяется память под вариантную часть записи?

   Для вариативной части используется объединение. В память для каждой строки выделяется одинаковое количество памяти, но вписываются только нужные данные. Так же добавлено дополнительное поле отвечающее за знание какой вариант вписан.

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

   Никто не знает, что произойдёт в этом случае, поэтому таким неподребством лучше не заниматься.

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

   Тот кто написал программу и тестировал её, если пользователю не запретили вводить что попало в программу, то виноват программист, который ничего не запрещал, и тестировщик, который этого не заметил.

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

   Это дополнительная таблица, в которой хранятся данные какого-то одного столбца, по которому будет сортироваться таблица, и индексы основной таблицы. Как выяснилось в ходе лабораторной работы, таблица нужна, чтобы оптимизировать время сортировки.

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

   Таблица ключей весьма ограничена в выборе столбцов, по которым можно сортировать, поэтому следует использовать таблицу ключей для столбца наиболее используемого для сортировки, или же использовать таблицу ключей для нескольких столбцов, но, например, без сохранения строк.

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

   Те способы, которые работают быстрее, потому что в них меньше перестановок. Используем QuickSort с наименьшим количеством перестановок (n*log(n)), а если имеет смысл, то и таблицу ключей.