Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ: Информатика и системы управления
КАФЕДРА: Программное обеспечение ЭВМ и информационные технологии
ДИСЦИПЛИНА: Типы и структуры данных
ТЕМА: Записи с вариантами. Обработка таблиц
ВАРИАНТ: 9

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

Студент: _____ *Княжев А. В.*
(подпись, дата)

Преподаватель: _____ *Силантьева А. В.*
(подпись, дата)

2021 г.

Содержание

1 Условие задачи

Ввести список квартир, содержащий адрес, общую площадь, количество комнат, стоимость квадратного метра, первичное жилье или нет (первичное — с отделкой или без нее; вторичное — время постройки, количество предыдущих собственников, количество последних жильцов, были ли животные). Найти все вторичное двухкомнатное комнатное жилье в указанном ценовом диапазоне без животных.

2 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.

Использовать два 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 процентах).

Программа должна выводить меню с возможностью выбирать варианты. При вводе нуля на запрос варианта меню, происходит выход из программы.

2.1 Входные данные

* имя файла, содержащего информацию о таблице;

* номер выбранного пункта меню.

2.2 Выходные данные

Выходные данные определяются выбранным пунктом меню.

2.3 Описание пунктов меню

2.3.1 Вывод таблицы с информацией обо всех квартирах

Выходные данные

* таблица, содержащая исходные записи.

2.3.2 Добавление записи в конец таблицы

Выводит диалог, позволяющий ввести запись. Диалог выглядит как набор запросов ввода одного поля записи.

Входные данные

* адрес квартиры — строка, длиной до 256 символов;

* площадь — вещественное число;

* количество комнат — целое число;
* цена за квадратный метр — целое число;
* статус — первичная или вторичная (строка Primary или Secondary);
* есть ли отделка (если первичная) — выбор из пунктов подменю (1 или 2);
* год постройки (если вторичная) — целое число;
* количество предыдущих владельцев (если вторичная) — целое число;
* количество жильцов (если вторичная) — целое число;
* жили ли животные (если вторичная) — выбор из пунктов подменю (1 или 2).

2.3.3 Удаление записи по полю

Выводит диалог, в котором пользователь указывает поле, по которому задается критерий удаления.

Входные данные

* номер поля, по которому выставляется значение критерия удаления;
* значение указанного поля (формат как в пункте про добавление записей).

2.3.4 Фильтрация

Выводит информацию о вторичных двухкомнатных квартирах в указанном ценовом диапазоне без животных.

Входные данные

* минимальное значение цены квартиры — целое число;
* максимальное значение цены квартиры — целое число.

Выходные данные

* таблица с информацией о записях, удовлетворяющих критериям фильтрации.

2.3.5 Сортировка таблицы ключей методом быстрой сортировки

Сортирует методом быстрой сортировки и выводит таблицу ключей.
Здесь и далее, **ключ сортировки — площадь квартиры**.

Выходные данные

* отсортированная таблица с информацией о ключах.

2.3.6 Сортировка таблицы ключей методом сортировки вставками

Сортирует методом сортировки вставками и выводит таблицу ключей.

Выходные данные

* отсортированная таблица с информацией о ключах.

2.3.7 Сортировка таблицы методом быстрой сортировки

Сортирует методом быстрой сортировки и выводит таблицу записей.

Выходные данные

* отсортированная таблица с информацией о записях.

2.3.8 Сортировка таблицы методом сортировки вставками

Сортирует методом сортировки вставками и выводит таблицу записей.

Выходные данные

* отсортированная таблица с информацией о записях.

2.3.9 Сортировка таблицы методом быстрой сортировки с помощью таблицы ключей

Сортирует таблицу ключей методом быстрой сортировки и выводит отсортированную таблицу записей с помощью таблицы ключей. При этом, таблица записей не изменяется.

Выходные данные

* отсортированная таблица с информацией о записях.

2.3.10 Сортировка таблицы методом сортировки вставками с помощью таблицы ключей

Сортирует таблицу ключей методом сортировки вставками и выводит отсортированную таблицу записей с помощью таблицы ключей. При этом, таблица записей не изменяется.

Выходные данные

* отсортированная таблица с информацией о записях.

2.3.11 Возврат таблицы к исходному состоянию

Перепрочитывает исходную таблицу из файла, тем самым отменяя все изменения, совершенные программой.

2.3.12 Сравнение эффективности сортировки таблицы с помощью таблицы ключей и без

Сравнивает эффективность сортировки таблицы с помощью таблицы ключей и без нее.

Выходные данные

* таблица с информацией о среднем времени выполнения сортировки без таблицы ключей для быстрой сортировки и сортировки вставками;
* таблица с информацией о среднем времени выполнения сортировки с таблицей ключей для быстрой сортировки и сортировки вставками;

2.3.13 Сравнение эффективности сортировки таблицы с помощью различных методов сортировки

Сравнивает эффективность сортировки таблицы записей и таблицы ключей с помощью быстрой сортировки и сортировки вставками.

Выходные данные

* таблица с информацией о среднем времени выполнения сортировки для быстрой сортировки и сортировки вставками;
* таблица с информацией о среднем времени выполнения сортировки таблицы ключей для быстрой сортировки и сортировки вставками;

2.4 Действие программы

Программа осуществляет работу с таблицей, содержащей информацию о квартирах.

2.5 Обращение к программе

Программа может быть запущена из командных оболочек `sh/bash/zsh/fish`, а также от IDE, способных работать с языком Си. Программа не принимает никаких аргументов.

2.6 Аварийные ситуации

2.6.1 Общие аварийные ситуации

1. строка имени файла с таблицей слишком длинная;
2. невалидный номер пункта меню (строка или пустая строка);
3. неверный номер пункта меню (такого пунта меню не существует);
4. ошибка считывания таблицы из файла: есть невалидные данные;
5. файл с таблицей не существует.

2.6.2 Пункт меню №1

1. таблица пуста.

2.6.3 Пункт меню №2

1. строка, содержащая адрес, слишком длинная;
2. площадь не является валидным вещественным числом;
3. площадь меньше или равна нулю;
4. количество комнат не является валидным целым числом;
5. количество комнат меньше или равно нулю;
6. цена за квадратный метр не является валидным целым числом;
7. цена за квадратный метр меньше или равна нулю;
8. номер пункта статуса не является валидным целым числом;
9. номер пункта статуса не существует;
10. номер пункта наличия отделки не является валидным целым числом;
11. номер пункта наличия отделки не существует;
12. год постройки не является валидным целым числом;
13. год постройки меньше или равен нулю;
14. количество владельцев не является валидным целым числом;
15. количество владельцев меньше или равно нулю;
16. количество жильцов не является валидным целым числом;

17. количество жильцов меньше или равно нулю;
18. номер пункта проживания животных не является валидным целым числом;
19. номер пункта проживания животных не существует;
20. таблица уже заполнена.

2.6.4 Пункт меню №3

1. номер поля, задающего критерий удаления не является валидным целым числом;
2. номер поля, задающего критерий удаления не существует;
3. для ввода значения поля ограничения такие же, как для пункта меню №2.

2.6.5 Пункт меню №4

1. нижняя граница цены не является валидным целым числом;
2. верхняя граница цены не ялвется валидным целым числом;
3. ни одна квартира не удовлетворяет критерию поиска.

2.6.6 Пункт меню №5

1. таблица пуста.

2.6.7 Пункт меню №6

1. таблица пуста.

2.6.8 Пункт меню №7

1. таблица пуста.

2.6.9 Пункт меню №8

1. таблица пуста.

2.6.10 Пункт меню №9

1. таблица пуста.

2.6.11 Пункт меню №10

1. таблица пуста.

3 Структуры данных

В данной работе используется запись со статическими полями с вариативной частью — с помощью нее хранятся записи о квартирах.

3.1 Основные константы

```
#define MAX_NUM_OF_FLATS 6000
#define MYSTRING_SIZE 257
```

3.2 Модуль для работы с ошибками

3.2.1 Типы данных

```
/**
 * text - текст ошибки;
 * code - код ошибки;
 * func - функция, в которой случилась ошибка.
 */
typedef struct
{
    const char *text;
    int code;
    const char *func;
} error_t;
```

3.2.2 Функции

```
/**
 * Создание новой ошибки.
 * text - текст ошибки;
 * code - код ошибки;
 * func - функция, в которой случилась ошибка.
 */
error_t new_error(const char *text, int code, const char *func);
```

```
/**
 * Создание ошибки-маркера успеха.
 * func - функция, в которой был создан "успех".
 */
error_t new_success(const char *func);
```

```
/**
 * Проверка, отображает ли ошибка ошибочную ситуацию.
 * err - исходная ошибка.
 */
bool is_failure(error_t err);
```

3.3 Модуль для работы с записью о квартире

3.3.1 Типы данных

```
/**
 * Перечисление для определения типа квартиры:
 * первичная или вторичная.
 * primary - первичная;
 * secondary - вторичная.
 */
typedef enum
{
    primary,
    secondary,
} flat_class_t;
```

```
/**
 * Информация о первичной квартире.
 * have_finishing - есть ли отделка.
 */
typedef struct
{
    bool have_finishing;
} primary_flat_info_t;
```

```
/**
 * Информация о вторичной квартире.
 * year_of_construction - год постройки;
 * number_of_owners - количество предыдущих собственников;
 * number_of_tenants - количество предыдущих жильцов;
 * were_animals - были ли животные.
 */
typedef struct
{
    int year_of_construction;
    int number_of_owners;
    int number_of_tenants;
    bool were_animals;
} secondary_flat_info_t;
```

```
/**
 * Дополнительная информация о квартире (вариативная часть).
 * primary - информация о первичной недвижимости;
 * secondary - информация о вторичной недвижимости.
 */
typedef union
{
    primary_flat_info_t primary;
    secondary_flat_info_t secondary;
} additional_info_t;
```

```
/**
 * Данные о кваритире.
 * address - адрес квартиры;
 * area - площадь;
 * number_of_rooms - количество комнат;
 * cost_per_square_meter - цена за квадратный метр;
 * class - класс квартиры: вторичная или первичная;
 * info - дополнительная информация (вариативная часть).
 */
typedef struct
{
    mystring_t address;
    double area;
    int number_of_rooms;
    int cost_per_square_meter;
    flat_class_t class;
    additional_info_t info;
} flat_t;
```

```
/**
 * Функция преобразования полей и строк.
 */
typedef error_t (*field_converter_t)(mystring_t, flat_t *);
```

```
/**
 * Набор функций-преобразователей из строковых значений
 * в поля структуры.
 * Названия функций соответствуют названиям полей,
 * которые они преобразуют.
 */
typedef struct
{
    field_converter_t address;
    field_converter_t area;
    field_converter_t number_of_rooms;
    field_converter_t cost_per_square_meter;
    field_converter_t class;
    field_converter_t have_finishing;
    field_converter_t year_of_construction;
    field_converter_t number_of_owners;
    field_converter_t number_of_tenants;
    field_converter_t were_animals;
} flat_converter_t;
```

3.3.2 Функции

```
/*
 * Преобразование массива строк в запись с
 * информацией о квартире.
 * strs - исходный массив строк;
 * n - количество элементов массива строк;
 * flat - запись с информацией о квартире;
 * c - преобразователь.
 */
error_t convert_to_flat(mystring_t *strs, size_t n,
flat_t *flat, flat_converter_t c);
```

```
/*
 * Преобразование записи с информацией о квартире
 * в массив строк, используя все варианты
 * вариативной части.
 * strs - массив строк;
 * n - количество элементов массива строк;
 * flat - исходная запись с информацией о квартире;
 * c - преобразователь.
 */
error_t convert_all_from_flat(mystring_t *strs, size_t n,
flat_t *flat, flat_converter_t c);
```

3.4 Модуль работы с «базой данных» квартир

3.4.1 Типы данных

```
/*
 * Ключ по площади квартиры.
 * area - площадь квартиры (значение ключа);
 * pos - позиция записи с этим ключом в исходной
 *       таблице.
 */
typedef struct
{
    double area;
    size_t pos;
} rooms_key_t;
```

```
/*
 * База данных с информацией о квартирах.
 * content - записи о квартирах;
 * keys - таблица ключей;
 * n - размер базы данных;
 * f - файл, в котором изначально записана база данных.
 */
typedef struct
{
    flat_t content[MAX_NUM_OF_FLATS];
    rooms_key_t keys[MAX_NUM_OF_FLATS];
    size_t n;
    FILE *f;
} flats_db_t;
```

3.4.2 Функции

```
/*
 * Считывание базы данных из файла.
 * db - база данных.
 */
error_t db_read(flats_db_t *db);
```

```
/*
 * Добавление записи о квартире в базу данных.
 * db - база данных;
 * flat - запись с информацией о квартире.
 */
error_t db_add_flat(flats_db_t *db, flat_t flat);
```

```
/*
 * Удаление записи из базы данных.
 * db - база данных;
 * key - значение ключа, по которому удаляем;
 * del - функция, определяющая, соответствует ли данная запись
 *       критерию удаления, заданному в key.
 */
error_t db_delete(flats_db_t *db, void *key,
bool (*del) (flat_t, void *));
```

```
/*
 * Критерии удаления для каждого поля.
 * flat - запись, которая проверяется;
 * key - значение ключа.
 */
bool db_del_by_address(flat_t flat, void *key);
bool db_del_by_area(flat_t flat, void *key);
bool db_del_by_cost_per_square_meter(flat_t flat, void *key);
bool db_del_by_number_of_rooms(flat_t flat, void *key);
bool db_del_by_class(flat_t flat, void *key);
bool db_del_by_have_finishing(flat_t flat, void *key);
bool db_del_by_year_of_construction(flat_t flat, void *key);
bool db_del_by_number_of_owners(flat_t flat, void *key);
bool db_del_by_number_of_tenants(flat_t flat, void *key);
bool db_del_by_were_animals(flat_t flat, void *key);
```

```
/*
 * Сортировка таблицы ключей методом сортировки вставками.
 * db - база данных.
 */
void db_insertion_sort_keys(flats_db_t *db);
```

```
/*
 * Сортировка таблицы методом сортировки вставками.
 * db - база данных.
 */
void db_insertion_sort_flats(flats_db_t *db);
```

```
/*
 * Сортировка таблицы ключей методом быстрой сортировки.
 * db - база данных.
 */
void db_quick_sort_keys(flats_db_t *db);
```

```
/*
 * Сортировка таблицы методом быстрой сортировки.
 * db - база данных.
 */
void db_quick_sort_flats(flats_db_t *db);
```

```
/*
 * Фильтрация таблицы по критерию из условия.
 * db - база данных;
 * min_cost - минимальная стоимость квартиры;
 * max_cost - максимальная стоимость квартиры.
 */
error_t db_filter(flats_db_t *db, int min_cost, int max_cost);
```

3.5 Модуль для работы со строками

3.5.1 Типы данных

```
/*
 * Тип строки длиной 256 символов + терминальный нуль.
 */
typedef char mystring_t[MYSTRING_SIZE];
```

3.5.2 Функции

```
/*
 * Чтение строки из файла.
 * f - файл;
 * str - строка, в которую считываем.
 */
error_t f_read_line(FILE *f, mystring_t str);
```

```
/*
 * Преобразование строки в число с плавающей точкой.
 * str - исходная строка;
 * res - результат.
 */
error_t to_double(char *str, double *res);
```

```
/*
 * Преобразование строки в целое число.
 * str - исходная строка;
 * res - результат.
 */
error_t to_integer(char *str, int *res);
```

```
/*
 * Удаление пробельных символов с двух сторон строки.
 * str - исходная строка.
 */
void trim_spaces(char *str);
```

```
/*
 * Разбиение строки по разделителю.
 * str - исходная строка;
 * seps - разделители;
 * words - количество максимальных слов разбиения;
 * res - результат;
 * n - количество слов в результате разбиения.
 */
error_t splitn(char *str, char *seps, size_t words,
mystring_t *res, size_t *n);
```

```
/*
 * Выравнивание строки.
 * str - исходная строка;
 * sym - символ, которым заполняется свободное пространство;
 * len - до какой длины выравнивать;
 * center - центрировать ли строку.
 */
error_t align(mystring_t str, char sym, size_t len, bool center);
```

3.6 Модуль для работы с тектовыми таблицами

3.6.1 Типы данных

```
/*
 * Данные о поле таблицы.
 * width - ширина поля;
 * name - название поля.
 */
typedef struct
{
    int width;
    mystring_t name;
} table_field_t;
```

```
/*
 * Информация о всех полях таблицы.
 * fields - данные о полях таблицы;
 * n - количество полей таблицы.
 */
typedef struct
{
    table_field_t fields[MAX_NUM_OF_FLATS];
    size_t n;
} table_t;
```

3.6.2 Функции

```
/*
 * Печать таблицы квартир.
 * db - база данных о квартирах.
 */
error_t print_flats_table(flats_db_t *db);
```

```
/*
 * Печать таблицы квартир по таблице ключей.
 * db - база данных о квартирах.
 */
error_t print_flats_table_by_keys(flats_db_t *db);
```

```
/*
 * Печать таблицы ключей.
 * db - база данных о квартирах.
 */
error_t print_keys_table(flats_db_t *db);
```

4 Описания алгоритмов

1. ввод с клавиатуры имени файла с информацией о квартирах;
2. считывание записей из файла;
3. вывод главного меню;
4. получение у пользователя номера пункта меню:

4.1 Пункт меню №1

1. проверка таблицы на пустоту;
2. вывод таблицы на экран.

4.2 Пункт меню №2

1. предложение пользователю пользователю ввести все параметры квартиры (адрес, площадь, количество комнат, цену за квадратный метр, тип квартиры: первичная или вторичная. Для первичной: наличие отделки. Для вторичной: год постройки, количество предыдущих владельцев, количество жильцов, проживание животных);
2. обработка ошибок для каждого поля;
3. добавление записи в «базу данных».

4.3 Пункт меню №3

1. предложение пользователю выбрать поле, по которому будет выставлен критерий удаления из «базы данных»;
2. ввод пользователем значения этого поля, обработка ошибок;
3. поиск и удаление соответствующих полей в «базу данных».

4.4 Пункт меню №4

1. создание копии «базы данных»;
2. фильтрация ее по данному признаку;
3. вывод таблицы на экран.

4.5 Пункт меню №5

1. сортировка таблицы ключей методом быстрой сортировки;
2. вывод таблицы ключей на экран.

4.6 Пункт меню №6

1. сортировка таблицы ключей методом сортировки вставками;
2. вывод таблицы ключей на экран.

4.7 Пункт меню №7

1. сортировка таблицы квартир методом быстрой сортировки;
2. вывод таблицы квартир на экран.

4.8 Пункт меню №8

1. сортировка таблицы квартир методомсортировки вставками;
2. вывод таблицы квартир на экран.

4.9 Пункт меню №9

1. сортировка таблицы ключей методом быстрой сортировки;
2. вывод таблицы квартир на экран с порядком строк, указанным в таблице ключей.

4.10 Пункт меню №10

1. сортировка таблицы ключей методом сортировки вставками;
2. вывод таблицы квартир на экран с порядком строк, указанным в таблице ключей.

4.11 Пункт меню №11

1. переставновка указателя в начало файла;
2. считывание записи из файла в таблицу.

4.12 Пункт меню №12

1. измерение времени быстрой сортировки исходной таблицы;
2. измерение времени быстрой сортировки по таблице ключей;
3. измерение времени сортировки вставками исходной таблицы;
4. измерение времени сортировки вставками по таблице ключей;
5. вывод таблицы для сортировки без таблицы ключей;
6. вывод таблицы для сортировки по таблице ключей;
7. расчет преимущества сортировки с таблицей ключей перед сортировкой без нее в процентах и вывод в виде таблицы;
8. расчет размера таблицы, вывод на экран;
9. расчет размера таблицы ключей, вывод на экран;
10. расчет суммарного размер «базы данных», вывод на экран;
11. расчет разницы размеров «базы данных» с таблицей ключей и без нее в процентах, вывод на экран.

4.13 Пункт меню №13

1. измерение времени быстрой сортировки исходной таблицы;
2. измерение времени быстрой сортировки таблицы ключей;
3. измерение времени сортировки вставками исходной таблицы;
4. измерение времени сортировки вставками таблицы ключей;
5. вывод таблицы для сортировки без таблицы ключей;
6. вывод таблицы для сортировки таблицы ключей.

5 Тестирование

Для проверко корректности работы программы было проведено функциональное тестирование. Таблица с тестовыми данными для «позитивных» и «негативных» случаев приведена ниже.

Так как входных/выходных данных может быть очень много, то в соответствующих полях таблицы могут быть указаны наиболее значимые части.

5.1 «Негативные» тесты

Название	**Входные данные**	**Выходные данные**
пустая строка в имени файла		101: Считанная строка пуста (f_read_line).
файл не существует	q.txt	2: Ошибка открытия файла (main).
некорректный формат записей в файле	data/broken.txt	201: Ошибка чтения из файла (db_read).
невалидный вариант пункта меню	abc	104: Строка не является корректным числом (to_integer).
несуществующий вариант пункта меню	100	101: Неверный пункт меню (process_main).
пустая строка в качестве варианта пункта меню		Выберите пункт меню: 101: Считанная строка пуста (f_read_line).
слишком длинная строка в поле адреса	*очень длинная строка*	103: Считанная строка слишком длинная (f_read_line).
невалидное значение площади	test	104: Строка не является корректным числом (to_double).
неположительное значение площади	-2.0	104: Некорректная площадь (area_from_input).
невалидное значение количества комнат	blabla	104: Строка не является корректным числом (to_integer).
неположительное значение количества комнат	0	104: Некорректная площадь (area_from_input).

Название	**Входные данные**	**Выходные данные**
невалидное значение цены за квадратный метр	ustal	104: Строка не является корректным числом (to_integer).
неположительное значение цены за квадратный метр	-10000	104: Некорректная площадь (area_from_input).
невалидное значение класса	class	401: Заданного варианта не существует (process_add_record).
некорректное значение класса	3	401: Заданного варианта не существует (process_add_record).
невалидное значение наличия отделки	eeeee	104: Строка не является корректным числом (to_integer).
некорректное значение наличия отделки	-1	401: Выбран несуществующий вариант (have_finishing_from _input).
невалидное значение года постройки	hod	104: Строка не является корректным числом (to_integer).
неположительное значение года постройки	-1991	104: Некорректный год постройки (year_of_construction _from_input).
невалидное значение количества владельцев	ekeck	104: Строка не является корректным числом (to_integer).
неположительное значение количества владельцев	-0	104: Некорректное количество владельцев (number_of_owners _from_input).
невалидное значение количества жильцов	ekeck	104: Строка не является корректным числом (to_integer).
неположительное значение количества жильцов	-10	104: Некорректное количество жильцов (number_of_tenants _from_input).

Название	Входные данные	Выходные данные
невалидное значение проживания животных	yes	104: Строка не является корректным числом (to_integer).
некорректное значение проживания животных	4	401: Выбран несуществующий вариант (were_animals_from_input).
невалидное значение минимальной цены в фильтрации	abcabc	104: Строка не является корректным числом (to_integer).
невалидное значение максимальной цены в фильтрации	abcabc	104: Строка не является корректным числом (to_integer).
пустой результат фильтрации	0 1	203: Пустой результат фильтрации (db_filter).
вывод пустой таблицы		401: Таблица пуста (print_flats_table).
ввод невалидного номера поля при удалении	bruh	104: Строка не является корректным числом (to_integer).
ввод не подходящего номера поля при удалении	100	401: Введен неправильный вариант поля (process_deletion).

5.2 «Позитивные» тесты

Название	Входные данные	Выходные данные
открытие корректного файла из 50 записями с информацией о квартирах	data/50.txt	*успешное открытие таблицы*
вывод непустой таблицы	data/50.txt \n 1	*выведенная таблица*
добавление корректной записи	data/50.txt \n 2 \n Tverskaya, 5 \n 100 \n 3 \n 400000 \n 1 \n 1	Запись успешно добавлена.
удаление по полю «адрес»	data/50.txt \n 3 \n 1 \n Injenerov, 32	Удалено 1 записей.

Название	**Входные данные**	**Выходные данные**
удаление по полю «площадь»	data/50.txt \n 3 \n 2 \n 106.30	Удалено 1 записей.
удаление по полю «количество комнат»	data/50.txt \n 3 \n 3 \n 1	Удалено 23 записей.
удаление по полю «цена за квадратный метр»	data/50.txt \n 4 \n 4 \n 94917	Удалено 1 записей.
удаление по полю «класс»	data/50.txt \n 3 \n 5 \n 2	Удалено 28 записей.
удаление по полю «наличие отделки»	data/50.txt \n 3 \n 6 \n 2	Удалено 7 записей.
удаление по полю «год постройки»	data/50.txt \n 3 \n 7 \n 2002	Удалено 1 записей.
удаление по полю «количество владельцев»	data/50.txt \n 3 \n 8 \n 3	Удалено 4 записей.
удаление по полю «количество жильцов»	data/50.txt \n 3 \n 9 \n 2	Удалено 11 записей.
удаление по полю «проживание животных»	data/50.txt \n 3 \n 10 \n 1	Удалено 15 записей.
фильтрация с ненулевым количеством подходящих записей	data/50.txt \n 4 \n 30000000 \n 45000000	*2 записи*
сортировка непустой таблицы ключей методом быстрой сортировки	data/50.txt \n 5	*отсортированная таблица ключей*
сортировка непустой таблицы ключей методом сортировки вставками	data/50.txt \n 6	*отсортированная таблица ключей*
сортировка непустой таблицы методом быстрой сортировки	data/50.txt \n 7	*отсортированная таблица квартир*

Название	**Входные данные**	**Выходные данные**
сортировка непустой таблицы методом сортировки вставками	data/50.txt \n 8	*отсортированная таблица квартир*
сортировка непустой таблицы методом быстрой сортировки с помощью таблицы ключей	data/50.txt \n 9	*отсортированная таблица квартир*
сортировка непустой таблицы методом сортировки вставками с помощью таблицы ключей	data/50.txt \n 10	*отсортированная таблица квартир*

6 Оценка эффективности

6.1 Объем занимаемой памяти

N	**Размер ключей, Б**	**Размер таблицы, Б**
500	8000	160016
1000	16000	320016
2000	32000	640016
3000	48000	960016
4000	64000	1280016
5000	80000	1600016

6.2 Время обработки

6.2.1 Сортировка вставками

N	**Время с ключами,** μs	**Время без ключей,** μs
500	493	4055
1000	1518	14346
2000	11006	56269
3000	14153	132472
4000	27185	229533
5000	37798	358023

6.2.2 Быстрая сортировка

N	**Время с ключами,** μs	**Время без ключей,** μs
500	44	195
1000	88	394
2000	264	957
3000	336	1642
4000	552	2724
5000	608	3028

6.3 Сравнение эффективности

6.3.1 Сортировка вставками

N	**Увеличение размера при использовании ключей, %**	**Преимущество по скорости при использовании ключей, %**
500	5	723
1000	5	845
2000	5	411
3000	5	836
4000	5	744
5000	5	847

6.3.2 Быстрая сортировка

N	**Увеличение размера при использовании ключей, %**	**Преимущество по скорости при использовании ключей, %**
500	5	343
1000	5	348
2000	5	262
3000	5	389
4000	5	393
5000	5	398

6.4 Выводы

Быстрая сортировка работает в данном случае заметно быстрее сортировки вставками.

Как видно из таблиц, сортировка с помощью таблицы ключей может быть заметно быстрее, чем сортировка по таблице исходных записей, так как размер ключа сильно меньше размера записи, и его копирование занимает меньшее время. Для большей показательности, я расчитал процент, на который увеличился размер таблицы при добавлении в нее таблицы ключей, а также то, насколько быстрее работает сортировка по таблице ключей по сравнению с сортировкой по таблице записей.

По составленной таблице нетрудно заметить, что чем больше элементов содержит исходная таблица, тем больший прирост производительности дает использование таблицы ключей для сортировки: с 343% для быстрой сортировки на 500 элементах, он вырастает до 398% на 5000 элементах, с 723% для сортировки вставками на 500 элементах, он вырастает до 847% на 5000 элементах.

При этом, в данном случае, таблица ключей дает прирост всего в 5% объема используемой памяти. Относительно такого прироста производительности, такой прирост по потреблению памяти можно считать оправданным.

7 Контрольные вопросы

7.1 Как выделяется память под вариантную часть записи?

Память выделяется единым блоком размером с максимальное поле вариантной части. Вариантная часть должна находиться в конце записи, таким образом, размер записи с вариантной частью будет вычисляться, как сумма длин полей фиксированной части и максимального по длине поля вариантной части.

7.2 Что будет, если в вариантную часть ввести данные, не соответствующие описанным?

Поведение не определено.

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

При обработке записей с вариантами программисту необходимо следить за правильностью хранения и обработки данных, содержащихся в вариантной части.

7.4 Что представляет собой таблица ключей, зачем она нужна?

Таблица ключей — дополнительный массив, содержащий индекс элемента в исходном массиве и ключ. Таблица нужна для ускорения операций, связанных с перестановками элементов (из-за меньшего размера ключа), например, сортировок. В таком случае, сортируется таблица ключей, а исходная таблица остается неизменной.

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

Таблицу ключей эффективнее использовать, когда нужно производить множество обменов на основании каких-либо ключей.

Однако, если в данных условиях важнее оптимальное использование памяти, а не скорость, то эффективнее обрабатывать данные в самой таблице, так как таблица ключей использует дополнительный по сравнению с таблицей объем памяти.

7.6 Какие способы сортировки предпочтительнее для обработки таблиц и почему?

Для обработки больших таблиц предпочтительнее использовать более оптимальные алгоритмы сортировки, например, быструю сортировку или сортировку слиянием.

Однако, если нам не так важна скорость, или происходит обработка небольших таблиц, можно использовать такие методы сортировок, как пузырьковая сортировка или сортировка вставками. Эти сортировки наиболее просты в понимании.

8 Вывод

Параметры реальных объектов бывает сложно описать, используя только числовые, либо только символьные типы данных. Часто требуется комбинация разных типов. Для этого используется тип «запись» (структура). Запись – это структурированный тип, состоящий из фиксированного числа компонентов различного типа. Иногда, когда надо использовать несколько вариантов параметров, можно использовать запись с вариативной частью — это запись, которая содержит часть, которую можно определить несколькими вариантами.

Для обработки больших таблиц записей (под обработкой подразумевается, например, сортировка), целесообразно использовать таблицу ключей. Ее использование может значительно (до 1000% преимущества в измерениях выше) увеличить скорость сортировки. Но стоит помнить, что таблица ключей занимает память, так что ее использование может быть нецелесообразно при обработке маленьких таблиц и в тех случаях, когда мы жестко ограничены по памяти.

Список литературы

[1] Методические рекомендации по лабораторной работе №2 (*http://wwwcdl.bmstu.ru/*)