**Отчёт лабораторной работы №2**

по курсу «Типы и структуры данных»

Отчёт выполнила:

Кондрашова Ольга

Группа ИУ7-35Б

Вариант 9

Задача 6

**Цель работы**:

Приобрести навыки работы с типом данных «запись», содержащим вариантную часть, и с данными, хранящимися в таблицах. Сравнить несколько различных алгоритмов сортировок массива при использовании таблиц записей с большим числом полей и таблиц ключей. Оценить относительную эффективность программы (в процентах) по времени и по используемому объему памяти.

**Задание (Задача 6)**:

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. Оценить эффективность этих алгоритмов (по времени и по используемому объему памяти) при различной реализации программы, то есть в случаях а) и б). Обосновать выбор алгоритма сортировки. Оценка эффективности должна быть относительной (в %).

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 2-х комнатное жилье в указанном ценовом диапазоне без животных.

**Входные данные:**

Программа получает на вход пустую таблицу и файл, из которого в таблицу будут добавлены записи.

На вход подается пункт меню (вводится пользователем), в зависимости от того, какое действие он хочет выполнить:

Если нажать 1 программа предложит выбрать: 1 – ввести данные из файла, 2 – ввести данные вручную, 0 – вернуться в меню.

Если нажать 2, нужно будет ввести номер записи, которые хотите удалить.

Если нажать 3, программа выведет таблицу с данными.

Если нажать 4, программа выведет таблицу ключей.

Если нажать 5, программа отсортирует исходную таблицу.

Если нажать 6, программа отсортирует таблицу ключей.

Если нажать 7, программа выполнит поиск записей по заданным параметрам. Для этого она предложит ввести минимальную и максимальную стоимость одного квадратного метра.

Если нажать 8, программа выведет информацию об оценке эффективности программы по времени и по памяти.

Если нажать 0, программа завершит работу.

**Выходные данные:**

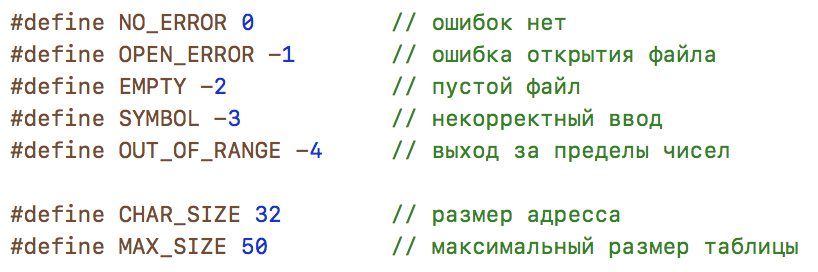
На выходе программа выводит таблицу с данными, по умолчанию отсортированную по порядку индекса. По запросу пользователя, программа выводит отсортированную таблицу (по цене или по площади (таблица ключей)). При добавлении или удалении записи программа выводит таблицу с выполненными изменениями. По команде пользователя программа выводит в таблице вторичное двухкомнатное жилье без животных в указанном пользователем ценовом диапазоне.

На выходе программа выводит таблицу с информацией об эффективности сортировок по времени и по памяти. Выводятся тесты с таблицами, состоящими из 1000, 5000 и 10000 записей. Для каждой измерено время сортировки и память, занимаемая исходной таблицей и таблицей ключей.

**Обращение к программе:**Через консоль

**Внутренняя структура данных:**

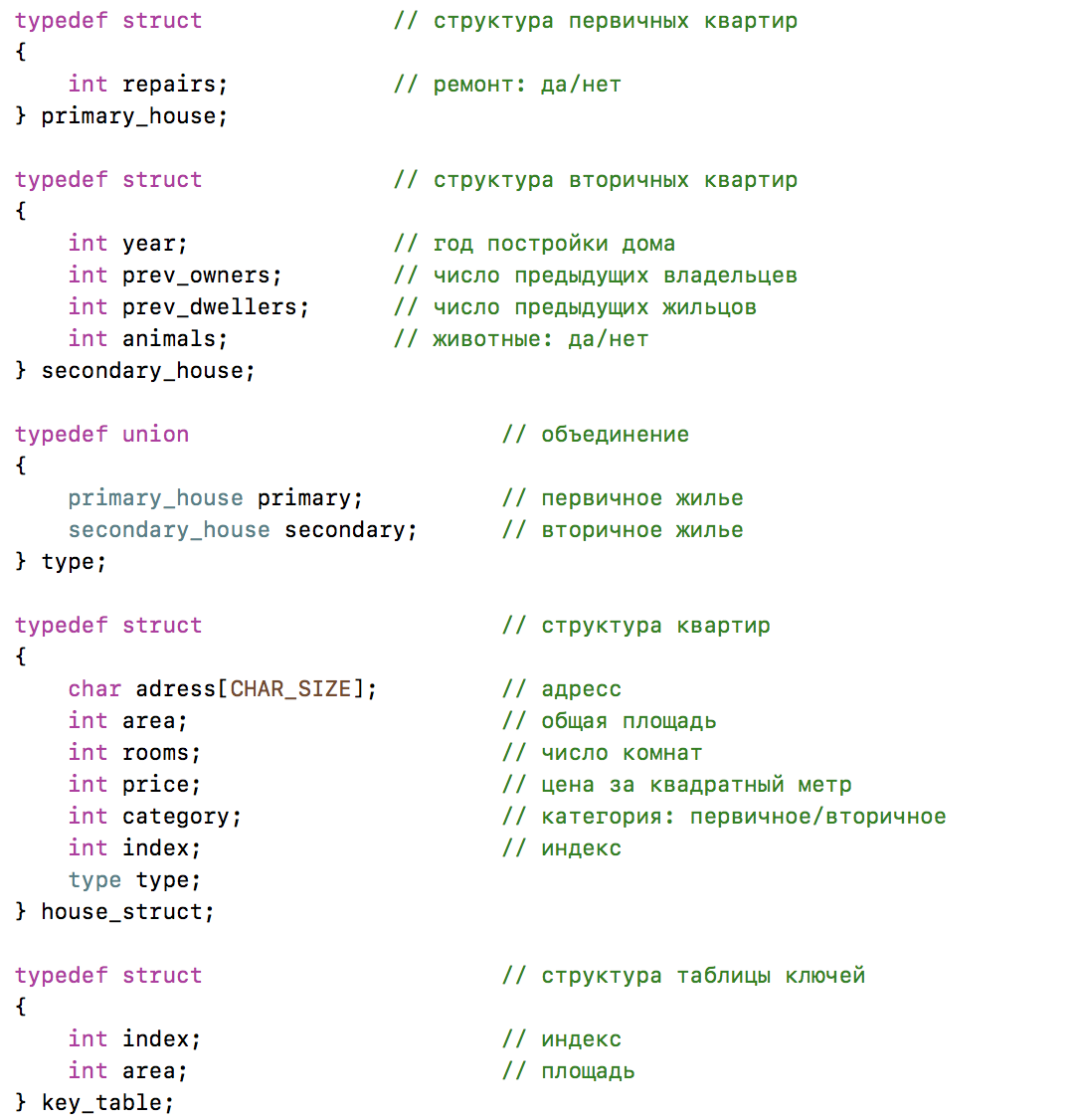
Константы:

****

Размеры массивов для сортировки:



Данные о квартирах представлены в виде структуры house\_struct.  
Делятся на первичные (primary\_house) и вторичные (secondary).



**Функции:**

int checkFile(FILE \*f)

Вход: файл.

Выход: код ошибки.

int readChar(char \*s, FILE \*f, int len)

Вход: строка char из файла либо из потока.

Выход: строка char либо код ошибки.

int readInt(int \*n, FILE \*f, int min\_range, int max\_range)

Вход: число int из файла либо из потока, допустимые границы числа.

Выход: число int либо код ошибки.

void recFromFile(house\_struct \*houses, key\_table \*key, int \*rec\_num, FILE \*f, int size)

Вход: пустые структуры квартир и ключей, число записей в таблице, файл, макисмальное число записей.

Выход: заполненные структуры квартир и ключей.

void recTable(house\_struct \*houses, key\_table \*key, int \*rec\_num)

Вход: пустые структуры квартир и ключей, число записей в таблице.

Выход: заполненные структуры квартир и ключей.

void printTable (house\_struct \*houses, key\_table \*key, int \*rec\_num, int \*flag\_sort)

Вход: пустые структуры квартир и ключей, число записей в таблице, флаг наличия сортировки.

Выход: выведенная на экран таблица квартир.

void printKeyTable (key\_table \*key, int \*rec\_num)

Вход: пустая структура ключей, число записей в таблице.

Выход: выведенная на экран таблица ключей.

void findApartment(house\_struct \*houses, key\_table \*key, int rec\_num)

Вход: массив структур квартир, массив структур ключей, число записей в таблице.

Выход: код ошибки.

void exitMenu()

Вход: функция void.

Выход: возвращение в меню.

void sort(house\_struct \*houses, int rec\_num)

Вход: массив структур квартир, число записей в таблице.

Выход: отсортированный массив структур квартир.

void quick\_sort(house\_struct \*houses, int first, int last))

Вход: массив структур квартир, индекс первого элемента, индекс последнего элемента.

Выход: отсортированный массив структур квартир.

void key\_sort(key\_table \*key, int rec\_num)

Вход: массив структур ключей, число записей в таблице.

Выход: отсортированный массив структур ключей.

void key\_quick\_sort(key\_table \*key, int first, int last)

Вход: массив структур ключей, индекс первого элемента, индекс последнего элемента.

Выход: отсортированный массив структур ключей.

void inputTestTable(house\_struct \*houses, key\_table \*key, int \*rec\_num, int size)

Вход: пустые структуры квартир и ключей, число записей в таблице, макисмальное число записей.

Выход: сгенерированные массивы структур квартир заданного размера.

void sorting\_test()

Вход: функция void.

Выход: вывод на экран таблицы с замером времени сортировок, размера структур, анализ эффективности по времени и памяти.

**Описание задачи, реализуемой программой**

1. Работа с данными: добавление записей в таблицу (из файла или из стандартного потока ввода) и удаление записей.
2. Вывод на экран основной таблицы и таблицы ключей.
3. Сортировка данных в таблице с использованием таблицы ключей.
4. Сравнение эффективности различных методов сортировки таблиц по времени и по памяти.

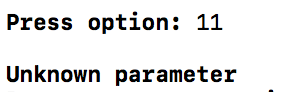
**Алгоритм**

1. Пустая таблица.
2. Добавление 40 записей из файла.
3. Добавление записей вручную.
4. Удаление записи по индексу.
5. Вывод основной таблицы.
6. Вывод таблицы ключей.
7. Сортировка по таблице ключей.
8. Поиск квартир, удовлетворяющих заданным параметрам (вторичные, двухкомнатные, без животных, цена задается пользователем).
9. Вывод информации о сортировке (время и память)

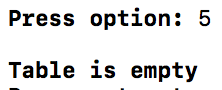
* Bubble sort
* Bubble key sort
* Quick sort
* Quick key sort
* Main table memory
* Key table memory

**Аварийные ситуации и ошибки пользователя**

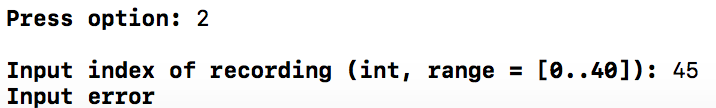
Ввод несуществующего параметра меню:



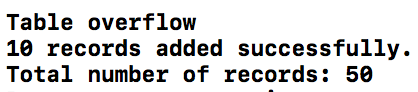
Сортировка, фильтрация или удаление из пустой таблицы:



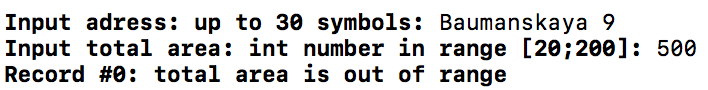
Удаление несуществующего элемента



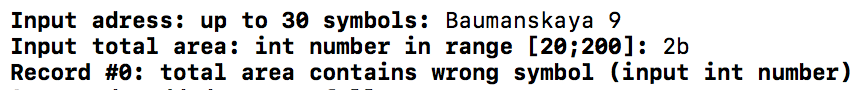
Переполнение таблицы



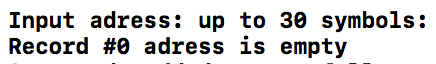
Выход за рамки допустимых значений



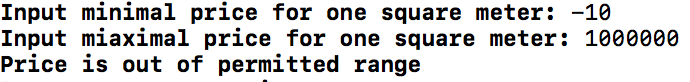
Ввод недопустимого символа в значение int

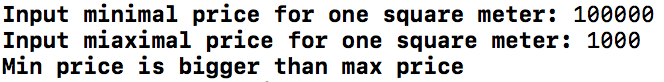


Пустой ввод



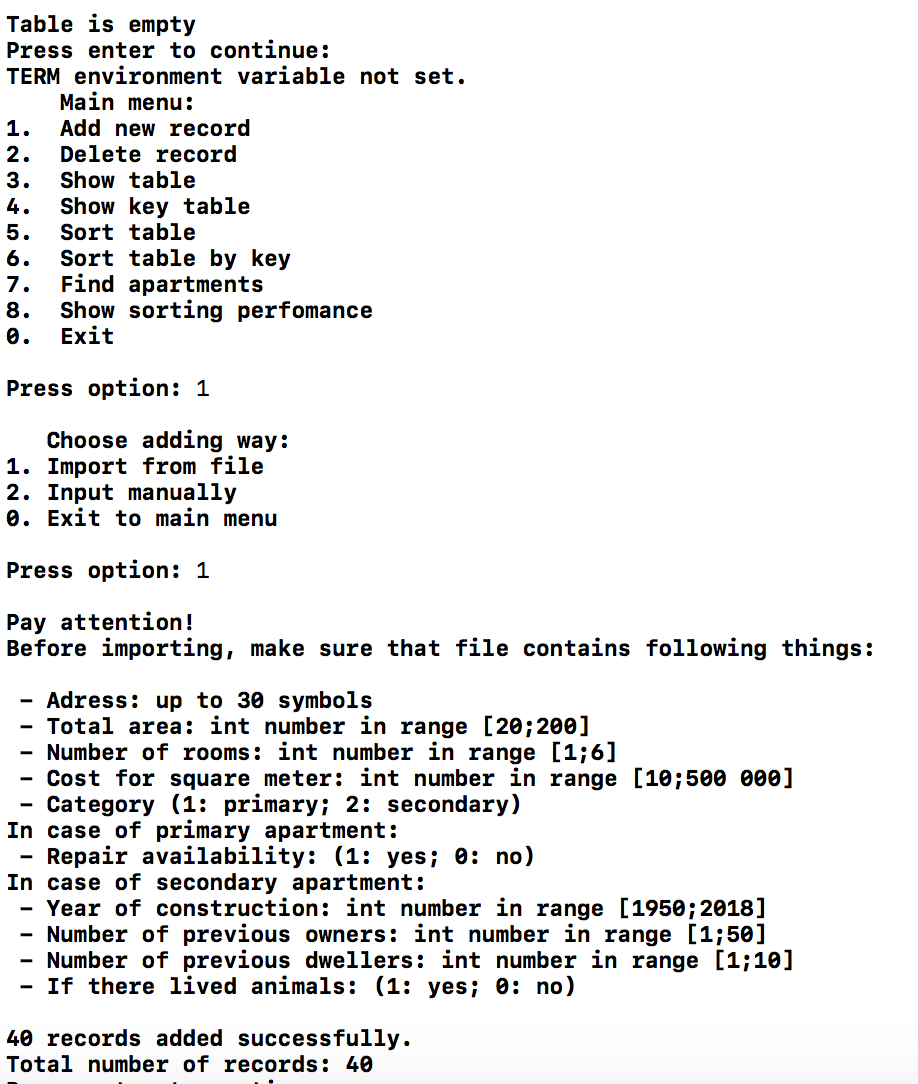
Ошибки фильтрации



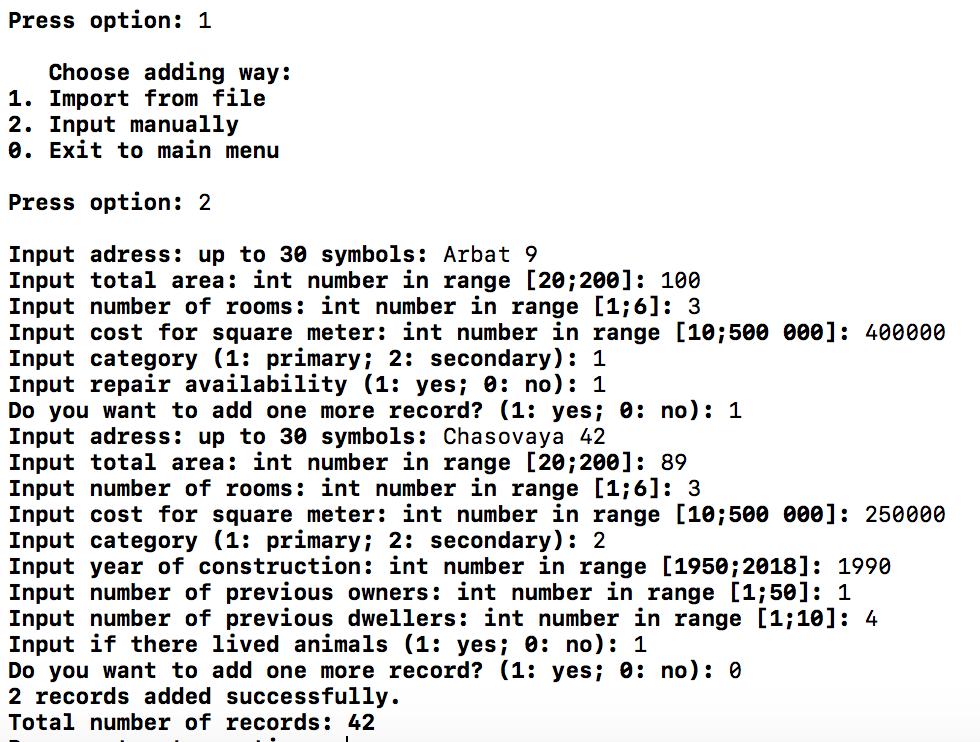


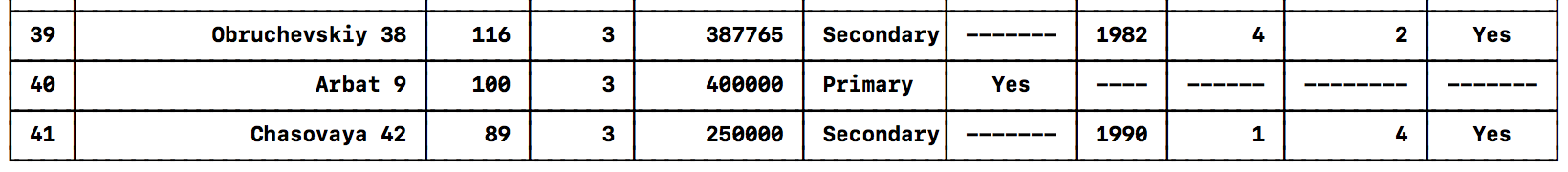
**Тесты**

Добавление записей из файла

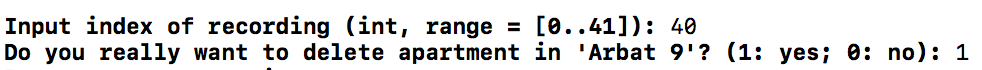


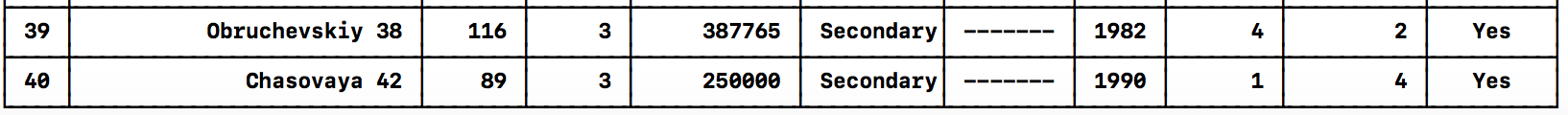
Добавление записей вручную



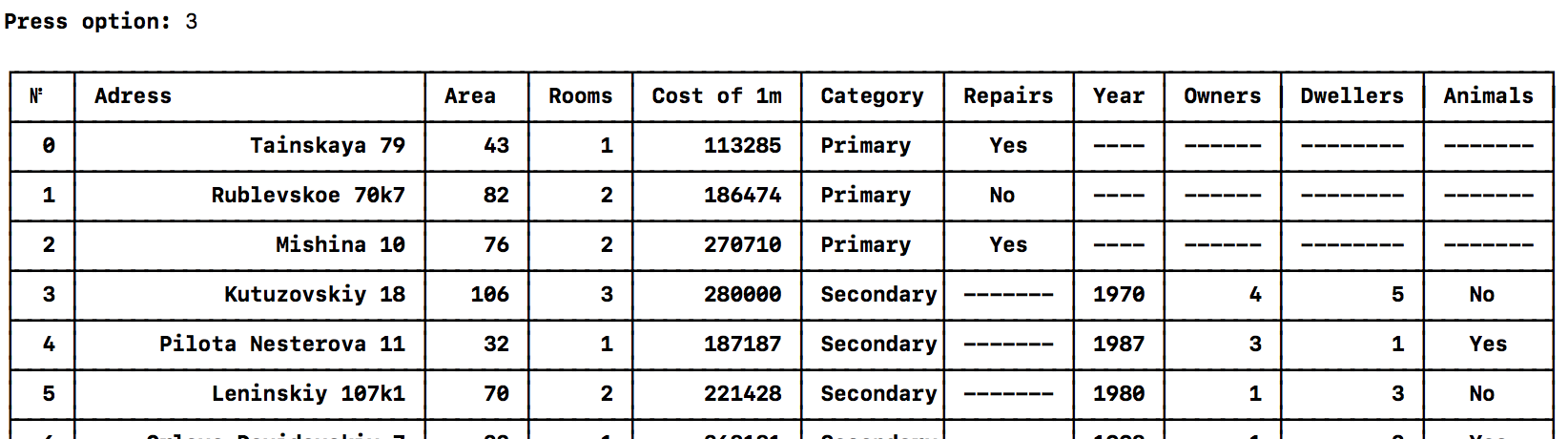


Удалений записей

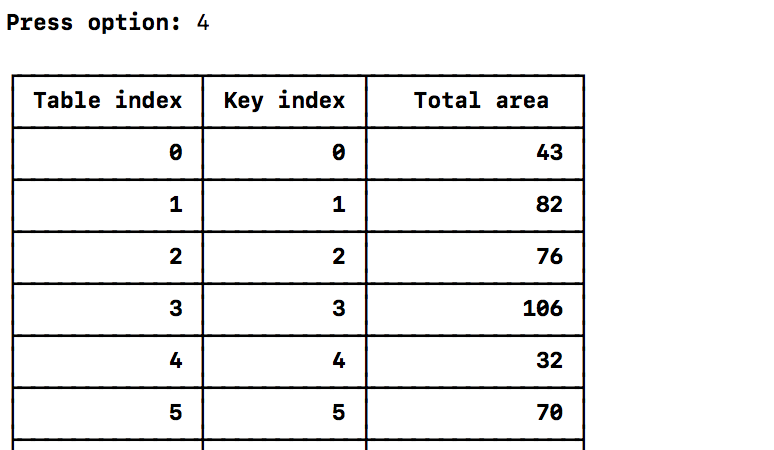




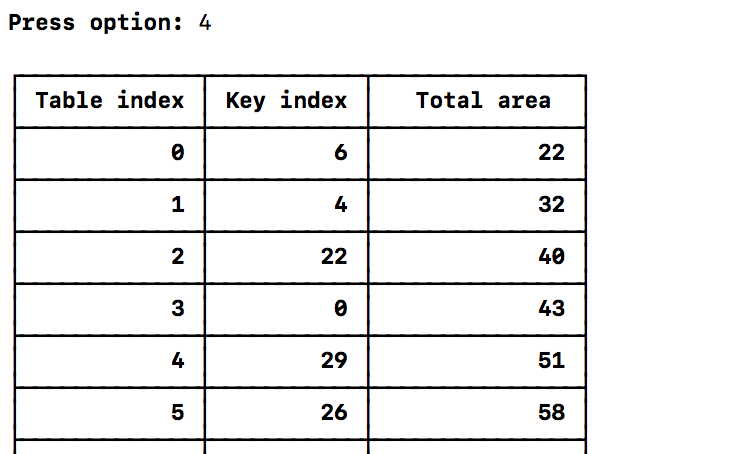
Вывод таблицы на экран

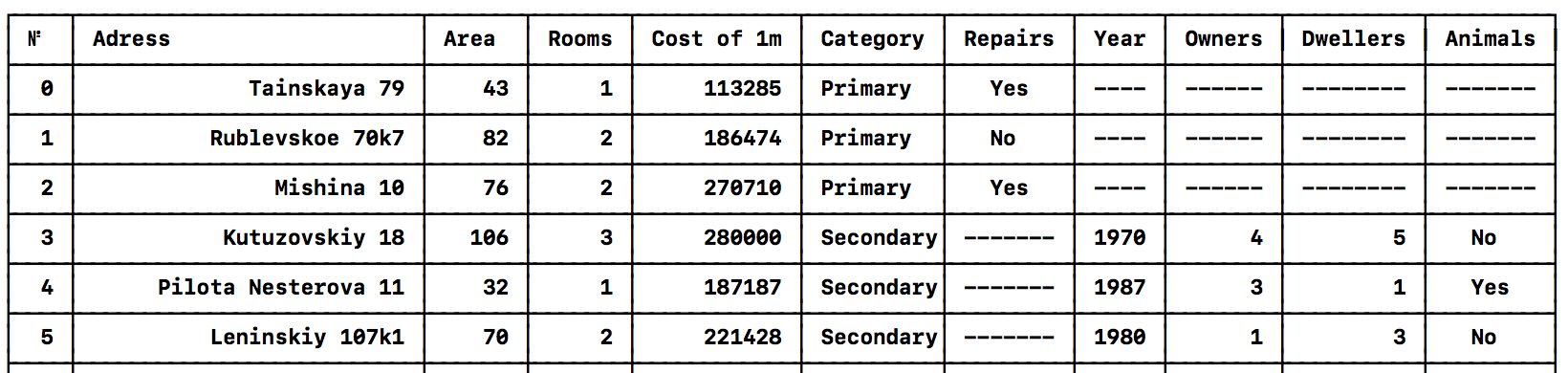


Вывод таблицы ключей на экран

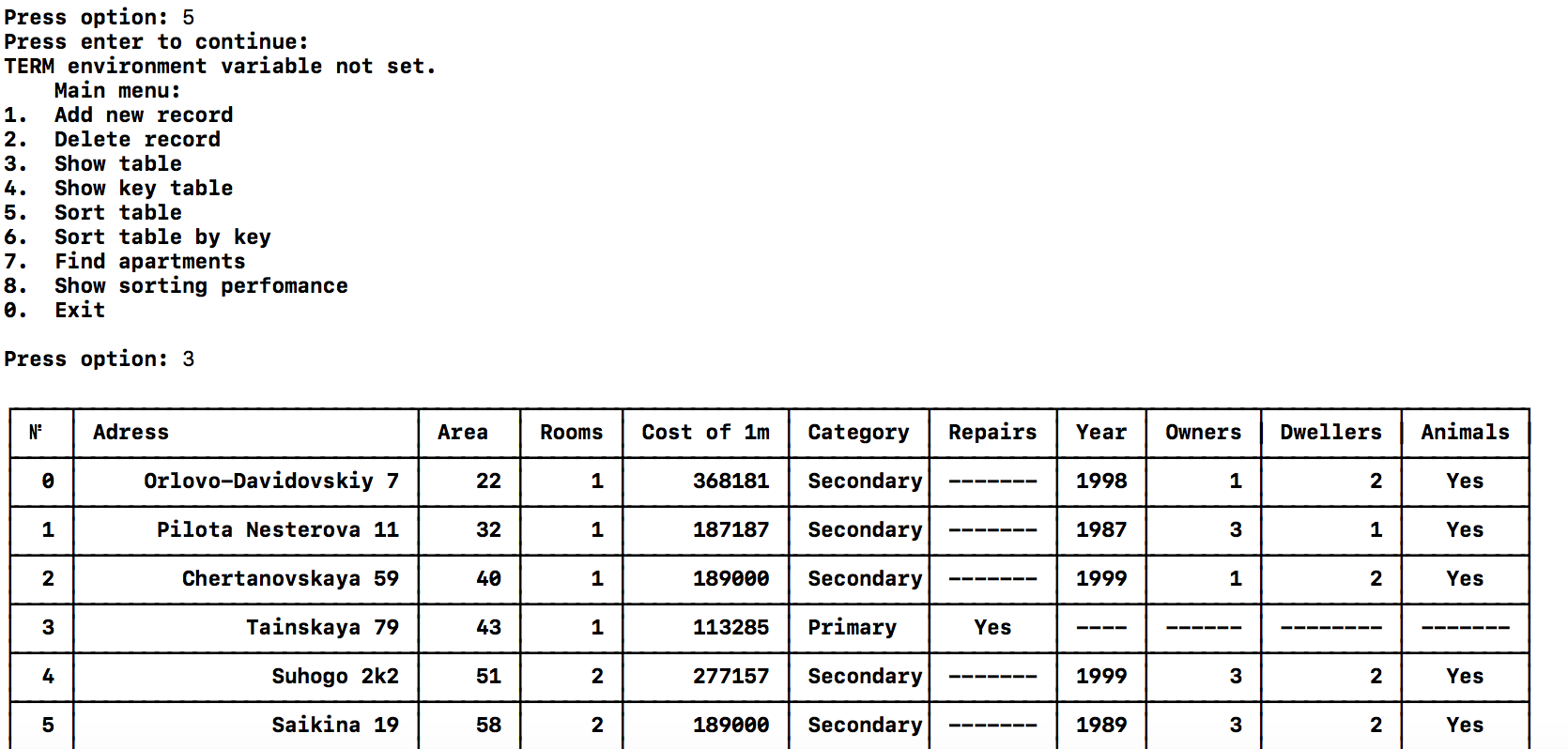


Просмотр отсортированной таблицы ключей при неотсортированной исходной таблице

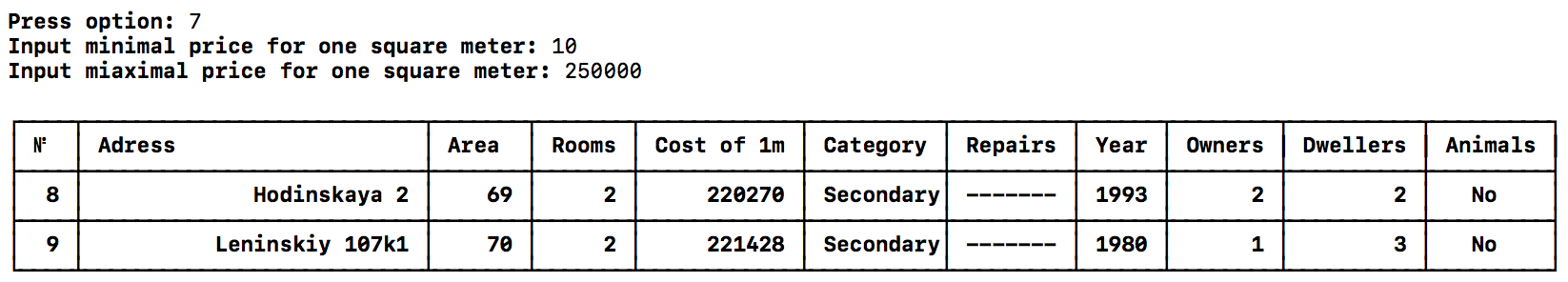




Отсортированная по ключу таблица

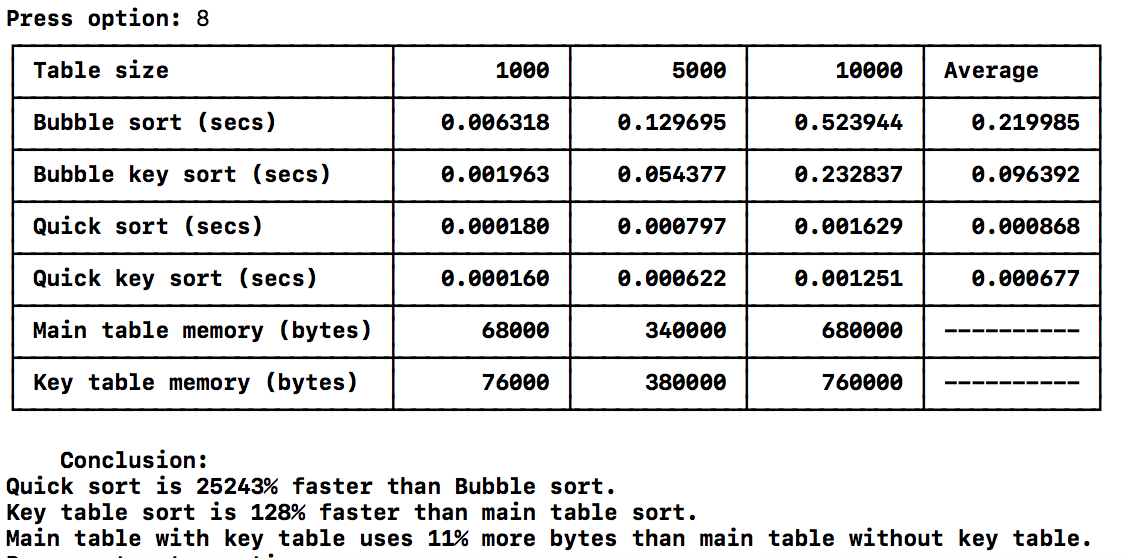


Поиск по фильтру



**Оценка эффективности**

Были проведены тесты с таблицами, состоящими из 1000, 5000 и 10000 записей. Под каждую таблицу память выделялась перед сортировкой и освобождалась после. Каждая таблица разной размерности заполнялась заново во избежание сортировки уже отсортированной таблицы.



В ходе тестирования были получены следующие результаты:

Эффективность по времени:

Quick sort быстрее bubble sort в среднем на 25243%.

Сортировка таблицы ключей быстрее сортировки основной таблицы в среднем на 128%.

Эффективность по памяти:

Работа с таблицей ключей занимает на 11% (8 байт) больше памяти, чем работа исключительно с исходной таблицей.

**Вывод:**

Для хранения исходных данных с ветвлением были выбраны структуры с объединением.

Было выявлено, что таблицу записей быстрее сортировать с помощью таблицы ключей (на 128%), хоть это и занимает на 11% больше памяти.

**Ответы на вопросы.**

1. Как выделяется память под вариантную часть записи?

Размер памяти равен памяти, занимаемой максимальным по длине полем. Эта память является общей для всех полей вариантной части записи.

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

Невозможно корректно прочитать данные.

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

Разработчик.

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

При больших размерах таблиц поиск данных и их сортировка может потребовать больших затрат времени. В этом случае можно уменьшить время обработки за счет создания дополнительного массива – таблицы ключей, содержащей индекс элемента в исходной таблице и выбранный ключ.

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

Если мы сортируем таблицу ключей, то экономится время, поскольку перестановка

записей в исходной таблице, которая иногда может содержать достаточно большое число полей, отсутствует. Однако, для размещения таблицы ключей требуется дополнительная память. Кроме того, если в качестве ключа используется символьное поле записи, то это влечет за собой необходимость посимвольной обработки данного поля в цикле, и, следовательно, приводит к увеличению времени выполнения любых операций. Выбор данных из основной таблицы в порядке, определенном таблицей ключей, так же замедляет вывод этих данных.

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

Сортировка эффективнее выполняется с помощью таблицы ключей, т.к. сортировка с использованием таблицы ключей значительно выигрывает по времени.