|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 №1**

**«ОБРАБОТКА БОЛЬШИХ ДАННЫХ»**

**по курсу «Типы и структуры данных»**

Студент: Чепиго Дарья Станиславовна

Группа: ИУ7-34Б

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Чепиго Д.С

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Барышникова М.Ю.

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Условие задачи**

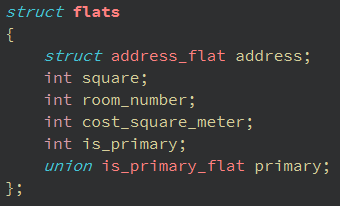
Создать таблицу, содержащую не менее 40 записей с вариантной частью. Произвести поиск информации по вариантному полю. Упорядочить таблицу, по возрастанию ключей (где ключ – любое невариантное поле по выбору программиста), используя: а) исходную таблицу; б) массив ключей, используя 2 разных алгоритма 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 %).

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.

**Техническое задание**

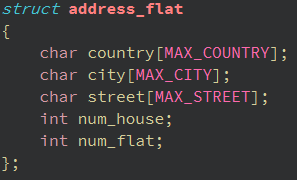
*Входные данные*

Массив структур типа “struct flats” максимальный размер которого MAX\_FLAT = 200.



В свою очередь структура “flats” содержит:

- Структуру “address\_flat”



Где char country[MAX\_COUNTRY] – страна, MAX\_COUNTRY = 15

char city[MAX\_CITY] – город, MAX\_CITY = 15

char street[MAX\_STREET] – улица, MAX\_STREET = 15

int num\_house – номер дома

int num\_flat – номер квартиры

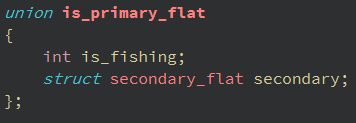
- int square – целое число, площадь квартира

- int room – целое число, количество комнат в квартире

- int cost\_square\_meter – целое число, стоимость одного квадратного метра

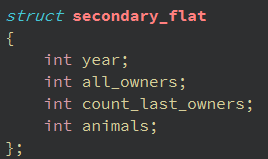
- int is\_primary – флаг новая ли квартира

- *union* is\_primary\_flat primary



Int is\_fishing – в случае, если квартира новая, то объединение состоит из информации об отделке квартиры.

Если же квартира не новая, то используется структура secondary\_flat



int year – целое число, год постройки квартиры

int all\_owners – целое число, общее количество жильцов

int count\_last\_owners – целое число, количество последних жильцов

int animals – флаг, отвечающий были ли у прошлых жильцов питомцы

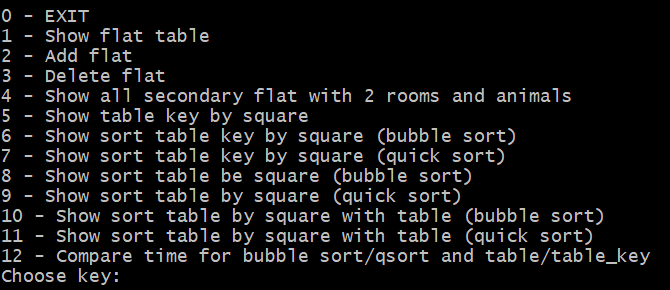
*Ограничения на входные данные:*

Страна, город и улица, где находится квартира должны быть по- одному слову и не больше 15 символов. Номер дома, номер квартиры, площадь квартиры, количество комнат, цена за один квадратный метр, год, количество жильцов – целые положительные числа. При вводе переменных – флагов (новая ли квартира или жили ли ранее питомцы) пользователю нужно на выбор ввести либо 0 либо 1.

В массиве максимально может хранится информации о 200 квартирах. Нельзя вызвать функцию добавления новой квартиры, если их уже 200 и нельзя вызвать функцию удаления, если массив пока пустой.

В случае неправильного ввода программа завершается. В случае, если массив пустой или наоборот в нем уже 200 элементов программа не завершается, но предлагает пользователю выбрать новый пункт меню.

*Описание меню*

**

0 – завершение программы с кодом возврата 0

1 – вывод таблицы с квартирами, в настоящее время

2 – возможность добавления квартиры (и в массив, и в файл)

3 - возможность удаления квартир с заданной площадью (и в массиве, и в файле)

4 – вывод на экран всех не новых 2х комнатных квартир, у крайних жильцов которой были питомцы. Диапазон цены за квадратный метр вводится пользователем

5 – вывод на экран текущей таблицы ключей

6 – вывод на экран отсортированной с помощью сортировкой пузырьком таблицы ключей

7 - вывод на экран отсортированной с помощью быстрой сортировки таблицы ключей

8 – вывод на экран отсортированной с помощью сортировкой пузырьком всей таблицы

9 - вывод на экран отсортированной с помощью быстрой сортировки всей таблицы

10 – вывод на экран отсортированной с помощью таблицы ключей (сортировка пузырьком) всей таблицы

11 – вывод на экран отсортированной с помощью таблицы ключей (быстрая сортировка) всей таблицы

12 – вывод на экран результаты выполнения четырех сортировок

Результаты сортировки разными способами

Всего 4 варианта сортировки:

-Сортировка таблицы ключей

-Сортировка исходной таблицы

-Сортировка пузырьком

-Быстрая сортировка (функция qsort из библиотеки stdlib.h)

В таблице указаны результаты 1 тыс. сортировок

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество записей | Быстрая сортировка | | Сортировка пузырьком | |
| Исходная таблица,  мс | Таблица ключей,  мс | Исходная таблица, мс | Таблица ключей,  мс |
| 50 | 16 | 1 | 22 | 4 |
| 100 | 21 | 16 | 100 | 23 |
| 200 | 46 | 17 | 536 | 32 |

Таким образом, получается, что qsort намного быстрее, а особенно на большом количестве данных.

Быстрая сортировка почти в 11 раз быстрее сортировки пузырьком на больших данных и в 1.5 раза на маленьких данных.

Также, можем сделать вывод о том, что намного быстрее сортировать таблицу ключей, чем исходную таблицу.

При быстрой сортировке время сортировки меньше в 16 раз в лучшем случае и в 1.5 в худшем случае, при сортировке пузырьком - в 17 раз в лучшем случае и в 4.5 в худшем случае.

Расход памяти на максимальном количестве размера массива – 200 структур:

Для таблицы ключей: 1600 байт.

Для всей таблицы структур: 17600 байт.

**Тестирование**

*Позитивные тесты.*

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Выходные данные | Результат |
| 1 | Ключ = 0 | Поток вывода пустой. Завершение программы | Код возврата - 0 |
| 2 | Ключ = 1 | На экран выводится таблица, которая ранее была считана из файла. | Ожидание следующего ключа |
| 3 | Ключ = 2  Ввод валидной квартиры. | На экран выводятся правила для ввода новой квартиры. Так как квартира валидна, выводится сообщение об успешно записанной квартире. | Ожидание следующего ключа |
| 4 | Ключ = 3  Ввод валидной площади для удаления – целое положительное число. | На экран выводится правила для ввода площади. Так как такие квартиры нашлись, то выведено сообщение об успешном удалении. | Ожидание следующего ключа |
| 5 | Ключ = 4  Ввод валидного диапазона для поиска квартир. | На экран выводятся правила для введения диапазона цен. Далее выводится таблица с найденными квартирами. | Ожидание следующего ключа |
| 6 | Ключ = 5 | Текущее состояние таблицы ключей | Ожидание следующего ключа |
| 7 | Ключ = 6 | Вывод отсортированной таблицы ключей (сортировка пузырьком) | Ожидание следующего ключа |
| 8 | Ключ = 7 | Вывод отсортированной таблицы ключей (быстрая сортировка) | Ожидание следующего ключа |
| 9 | Ключ = 8 | Вывод отсортированной полной таблицы (сортировка пузырьком) | Ожидание следующего ключа |
| 10 | Ключ = 9 | Вывод отсортированной полной таблицы(быстрая сортировка) | Ожидание следующего ключа |
| 11 | Ключ = 10 | Вывод отсортированной таблицы с помощью таблицы ключей (сортировка пузырьком) | Ожидание следующего ключа |
| 12 | Ключ = 11 | Вывод отсортированной таблицы с помощью таблицы ключей(быстрая сортировка) | Ожидание следующего ключа |
| 13 | Ключ = 12 | Вывод информации о времени для 4х видов сортировок и объема занимаемой памяти. | Ожидание следующего ключа |
| 14 | Ключ = 11  Ключ = 1 | В начале программа выведет отсортированную таблицу, а далее – несортированную исходную таблицу. | Ожидание следующего ключа |
| 15 | Ключ = 5  Ключ = 6  Ключ = 5 | В начале программа выводит несортированный массив ключей. После дважды выводит сортированный массив ключей. | Ожидание следующего ключа |

*Негативные тесты*

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Выходные данные | Результат |
| 1 | Ключ = 2  Неверный ввод структуры.  Italy Rome Colisis -123 | Сообщение где именно пользователь ввёл невалидный результат.  “Error number of house. Please try again” | Код возврата = 10 |
| 2 | Ключ = 1  Файл не найден | Сообщение об ошибке, что файл не найден | Код возврата = 2 |
| 3 | Ключ = 2  Неверный ввод параметров квартиры  Italy Rome Colisis 123 12 -1234 | Сообщение где именно пользователь ввёл невалидный результат.  "Error square of flat. Please try again.” | Код возврата = 11 |
| 4 | Ключ = 2, но в массиве уже есть 200 элементов | Сообщение, что массив полностью заполнен | Ожидание следующего ключа |
| *5* | *Ключ = 3, но массив пустой* | *Сообщение, что массив пустой и в нем нечего удалять* | Ожидание следующего ключа |

**Контрольные вопросы**

*Как выделяется память под вариантную часть записи?*

В языке си, вариативная часть структуры реализована с помощью объединений - “union”. Память выделяется в одном “куске” памяти, имеющий размер, который способен вместить наибольшее поле из указанных.

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

Результат будет системно-зависимым и трудно предсказуемым. Возможно, произойдет приведение типов.

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

Ответственность за правильность проведения операций целиком и полностью лежит на программисте.

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

Таблица ключей представляет собой таблицу, в которой находится два столбца: номер ячейки в исходной таблице и значение выбранного программистом поля исходной таблицы для этой ячейки (в моем случае – цена машины).

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

Обрабатывать данные в самой таблице эффективнее использовать, когда время обработки не так важно, как задействованная память. А использование таблицы ключей, наоборот, эффективно, когда нужно быстрое время обработки и не так важна дополнительная задействованная память. Так же, использование таблицы неэффективно, когда сама таблица состоит из маленького количества полей, например, таблица, имеющая два поля: “Ученик” и “Оценка”. В таком случае, таблица ключей будет лишь занимать дополнительное место в памяти и не даст никакой выгоды во времени.

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

Для таблиц из большого количества записей предпочтительно использовать стандартные и устойчивые способы сортировки, со средним временем обработки O(n\*log(n)), такие как QuickSort, MergeSort и т.д.   
Если же в таблице не так много записей, то предпочтительнее использовать простые алгоритмы сортировки, например, сортировку пузырьком.