|  |  |
| --- | --- |
| **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)** |

**Дисциплина “Типы и структуры данных”**

**Лабораторный практикум №2**

**по теме: «Записи с вариантами, обработка таблиц»**

Выполнил студент: \_\_*Клименко Алексей Константинович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*фамилия, имя, отчество*

Группа: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*ИУ7-35Б*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил, к.п.н.: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*подпись, дата*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2020 г.*

**цель работы**

Приобрести навыки работы с типом данных «запись» (структура), содержащим вариантную часть (объединение, смесь), и с данными, хранящимися в таблицах, произвести сравнительный анализ реализации алгоритмов сортировки и поиска информации в таблицах, при использовании записей с большим числом полей, и тех же алгоритмов, при использовании таблицы ключей; оценить эффективность программы по времени и по используемому объему памяти при использовании различных структур и эффективность использования различных алгоритмов сортировок.

**описание условия задачи**

Необходимо cоздать 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, двумя алгоритмами сортировки

Оценить эффективность выбранных 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 %).

**техническое задание**

**исходные данные**

Исходными данными является таблица квартир, считываемая из файла, а также добавляемые в процессе работы программы квартиры.

**Программное меню**

При запуске, пользователю предлагается ввести имя файла с данными таблицы. После успешного считывания данных из файла, перед пользователем отображается меню, в котором он может выбирать опции по своему желанию.

В главном меню пользователь может:

* Добавить квартиру в таблицу
* Посмотреть таблицу
* Удалить квартиру по ID
* Найти квартиру по параметрам
* Перейти в меню сортировки

В меню сортировки пользователь может:

* Посмотреть исходную таблицу
* Посмотреть таблицу по ключам
* Посмотреть таблицу ключей
* Отсортировать таблицу ключей

**Множетво допустимых значений**

Для данной СД для хранения информации о квартирах допустимые значения для полей структуры таковы: // не нужно здесь

Адрес (address) – строка до 30 символов (без символа «;»)

Площадь комнат (area) – (0.0f, 830.0f]

Количество комнат (rooms\_amount) – [1, 10]

Цена за квадратный метр (price\_per\_m2) – (0.0f, 1.0e+6]

Тип жилья (type) – { PRIMARY, SECONDARY }

Наличие отделки (has\_trim) – { 0, 1 }

Были ли домашние животные (was\_pets) – { 0, 1 }

Время постройки (build\_time) – [1880, 2020]

Кол-во предыдущих владельцев – [0, 100]

Кол-во предыдущих жильцов – [0, 20]

**выходные данные**

Выходными данными является таблица ключей, отсортированная таблица и таблица по ключам.

**результат**

Результатом работы программы является сортировка таблицы; поиск, добавление и удаление квартиры.

**описание задачи, реализуемой программой**

Программа при запуске запрашивает имя файла с данными, после чего считывает данные из файла и запускает пользовательское меню.

**способы обращения к программе**

Вызов программы происходит любым возможным способом запуска программы **app.ехе**. Для корректной работы программы при её вызове не требуются дополнительные аргументы командной строки.

**возможные аварийные ситуации и ошибки пользователя**

При неправильном вводе пользователя программа возвращается в меню и предлагает повторить ввод еще раз.

**Структуры данных**

Для хранения информации об одной квартире мной используется структура **flat\_t**:

**typedef enum**

**{**

**PRIMARY,** // Первичная квартира

**SECONDARY** // Вторичная квартира

**} flat\_type\_t;**

**typedef struct**

**{**

**unsigned int id;** // ID квартиры (в таблице)

**char address[MAX\_ADDRESS\_SIZE];** // адрес квартиры (30 символов)

**float area;** // общая площадь

**uint8\_t rooms\_amount;** // кол-во комнат

**float price\_per\_m2;** // цена за квадратный метр

**uint8\_t type;** // PRIMARY или SECONDARY

**union**

**{**

**uint8\_t has\_trim;** // Отделка. 0 или 1

**uint8\_t was\_pets;** // Наличие животных. 0 или 1

**} type\_data;**

**time\_t build\_time;** // время постройки

**uint8\_t prev\_owners\_amount;** // предыдущие владельцы

**uint8\_t prev\_lodgers\_amount;** // предыдущие жильцы

**} flat\_t;**

Объём памяти необходимый для хранения одной записи:

**sizeof(flat\_t) == 72**

Структуры для хранения и обработки таблицы с квартирами:

**typedef struct**

**{**

**uint32\_t id;**

**uint8\_t rooms\_amount;**

**} flat\_key\_t; // Структура для хранения одного ключа**

**typedef struct**

**{**

**size\_t size;**

**flat\_t \*flats\_array;**

**} flat\_table\_t; // Таблица квартир**

**typedef struct**

**{**

**size\_t size;**

**flat\_key\_t \*keys;**

**} keys\_table\_t; // Таблица ключей**

**Набор функций**

**// Инициализация структуры квартиры нулевыми значениями**

**flat\_t create\_flat(void);**

**// Создание копии структуры**

**flat\_t clone\_flat(flat\_t \*original);**

**// Преобразование информации из строки в структуру**

**int sread\_flat(const char \*str, flat\_t \*flat);**

**// Печать данных структуры на экран**

**void printf\_flat(flat\_t \*flat);**

**// Создание пустой таблицы квартир**

**flat\_table\_t ft\_create(void);**

**// Копирование таблицы квартир**

**flat\_table\_t ft\_clone(flat\_table\_t \*original);**

**// Очищение памяти и обнуление таблицы**

**void ft\_free(flat\_table\_t \*flat\_table);**

**// Чтение таблицы из файла**

**int ft\_read(FILE \*file, flat\_table\_t \*flat\_table);**

**// Добавление квартиры в конец таблицы**

**int ft\_append\_flat(flat\_table\_t \*flat\_table, flat\_t \*flat);**

**// Удаление квартиры по ID**

**int ft\_delete\_flat(flat\_table\_t \*table,**

**unsigned int id, flat\_t \*deleted\_flat);**

**// Заполнение таблицы ключей по исходной таблице**

**void ft\_gen\_keys(flat\_table\_t \*table, flat\_t \*keys);**

**// 4 метода сортировки.**

**// a = с таблицей ключей, b = без ключей**

**// slow = обменом, fast = вставками**

**size\_t ft\_sort\_a\_fast(flat\_table\_t \*table,**

**key\_table\_t \*keys, sort\_params\_t params);**

**size\_t ft\_sort\_a\_slow(flat\_table\_t \*table,**

**key\_table\_t \*keys, sort\_params\_t params);**

**size\_t ft\_sort\_b\_fast(flat\_table\_t \*table,**

**key\_table\_t \*keys, sort\_params\_t params);**

**size\_t ft\_sort\_b\_slow(flat\_table\_t \*table,**

**key\_table\_t \*keys, sort\_params\_t params);**

**Описание алгоритмов обработки данных**

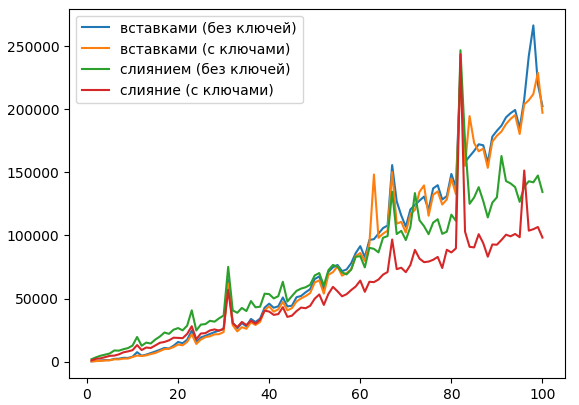
Для реализации сортировки мною будут рассмотрены такие методы как сортировка обменом и сортировка слиянием как представители более медленной и более быстрой сортировок.

Приведём сложность упомянутых алгоритмов:

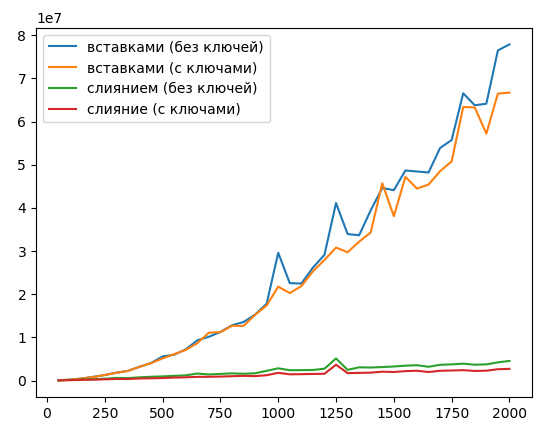
Сортировка вставками: **O(N2)**

Сортировка слиянием: **O(N\*log(N))**

Результаты тестирования в виде графиков:



Для большего числа элементов:



Отностительная эффективность по времени:

вставками/слиянием (без ключей): **33.55**%

вставками/слиянием (с ключами): **50.12**%

с/без ключей (слияние): **26.86**%

**набор тестов**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Описание теста** | **Входные данные** | **Выходные данные** |
| 1 | Вывод таблицы | in.txt  3 | Форматированная таблица с данными |
| 2 | Ошибка при чтении файла | Null | Сообщение об ошибке. Завершение работы |
| 3 | Неверный выбор опции меню | in.txt  a | Отображение сообщения об ошибке и повторный запуск меню |
| 4 | Добавление новой записи | in.txt  1  *<flat data>*  3 | Добавление записи в конец таблицы |
| 5 | Ошибки при добавлении новой записи | in.txt  1  *<invalid data>* | Прерывание оперции, возврат в меню |
| 6 | Удаление записи | in.txt  5  *<flat id>*  3 | Удаление записи с указанным ID из таблицы |
| 7 | Ошибки при удалении записи | in.txt  5  *<not flat id>* | Отображение сообщения об ошибке и переход в меню |
| 8 | Сортировка таблицы по ключу | in.txt  2  *<1,2,3,4>*  *<1,2,3>*  *<0,1>* | Сортировка таблицы с последующим её отображением |
| 9 | Сортировка пустой таблицы | in.txt  2  *<1,2,3,4>*  *<1,2,3>*  *<0,1>* | Сообщение о пустой таблице и переход в главное меню |
| 10 | Ошибки при сортировке таблицы | in.txt  2  *<invalid vals>* | Отображение сообщения об ошибке и переход в меню |
| 11 | Поиск записей по условию | in.txt  4  *<min price>*  *<max price>* | Вывод на экран всех подходящих под условия поиска записей или сообщения, что подходящих записей нет |
| 12 | Неверное условие при поиске записей | in.txt  4  *<wrong prices>* | Вывод на экран сообщения об ошибке и переход в меню |

**Выводы по проделанной работе**

По окончании работы мне удалось на практике сравнить эффективность двух различных алгоритмов сортировки, а также подтвердить практически их асимптотическую сложность, рассчитанную теоритически.

Судя по полученным результатам, сортировать таблицу с применением дополнительных массивов, оказывается немного более эффективным решением с точки зрения времени выполнения (приблизительно на **26.9%** для алгоритма сортировки слиянием).

Таким образом, при разработке программ необходимо выбирать структуры данных и алгоритмы по их обработке основываясь на имеющихся ресурсах для того, чтобы сделать разрабатываемый продукт наиболее эффективным.

**Контрольные вопросы**

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

Память под вариантную часть записи выделяется единым блоком, который по своему объему может уместить максимальный тип из используемых. При этом остальные типы используют ту же область памяти, из-за чего могут быть логические ошибки при неверном интерпретировании имеющихся в вариантой части данных.

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

В лучшем случае произойдет ошибка компиляции. В худшем — введённые данные будут неправильно интерпретироваться в дальнейшем и в какой-то момент приведут к более серьёзным последствиям.

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

За правильностью выполнения операций с вариантной частью должен следить сам программист.

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

Таблица ключей представляет собой массив из упрощенных моделей обычных записей, которые включают в себя минимально возможную информацию для однозначного сопоставления их с исходными записями.

Таблица ключей нужна для сокращения времени работы с исходной таблицей при необходимости частой модификации структуры таблицы, но не самих записей в ней. Например, такой модификацией можно считать сортировку записей, вставку новой записи с сохранением упорядоченности таблицы.

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

В случаях, когда память является более весомым критерием эффективности, следует обрабатывать данные непосредственно на месте, а когда на первом месте стоит время, то конечно стоит использовать таблицу ключей.

Также, если в самой таблице не очень много данных, и они не часто обрабатываются, то перебарщивать с оптимизацией не нужно — в большинстве случаев прирост производительности будет неоправданным (если вообще будет).

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

Для обработки таблиц предпочтительнее использовать способы сортировки не требующие большого количества проходов по всему объему данных, так как таблицы зачастую хранят довольно большие объемы информации и такие «обходы» могут очень дорого обойтись, когда речь зайдёт об эффективности алгоритмов сортировки.