|  |  |
| --- | --- |
| **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**

// пункты меню, ввод с клавиатуры

**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.

**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]

**результат**

Одним из результатов работы программы является количественная информация (представленная в виде графика) с указанием времени, затраченного на обработку исходной таблицы и таблицы ключей двумя алгоритмами сортировки (при этом, не забыть оценить так же время выборки данных из основной таблицы с использованием таблицы ключей), а так же - объем занимаемой при этом оперативной памяти.

// результат – сортировка таблицы, поиск квартиры, удаление, добавление

// выходные данные – таблица ключей, отсортированная таблица, таблица по ключам

**описание задачи, реализуемой программой**

Программа при запуске запрашивает имя файла с данными, после чего считывает данные из файла и запускает цикл, в котором пользователь может выбрать одну из доступных опций (за одну итерацию цикла). По окончании работы программа записывает результирующие данные в выходной файл, название которого программой запрашивается в начале работы программы.

**способы обращения к программе**

Вызов программы происходит любым возможным способом запуска **app.ехе** программ. (Например, через консоль). Для корректной работы программы при её вызове не требуются дополнительные аргументы командной строки.

**возможные аварийные ситуации и ошибки пользователя**

Наиболее возможными в работе программы будут ошибки, связанные с некорректным вводом пользователя или с содержанием файла. Хотя не исключены возможные сбои из-за невозможности выделения динамической памяти при считывании данных из файла.

// неправильный ввод с клавиатуры

**Структуры данных**

Для хранения информации об одной квартире мной используется структура **flat\_t**, которая содержит в себе в качестве полей: указатель на строку с адресом квартиры, общая площадь комнат, количество комнат, стоимость квадратного метра, тип (первичное жильё или вторичное), далее для первичного типа — с отделкой или без неё, а для вторичного — были ли животные; время постройки, количество предыдущих собственников, количество последних жильцов.

|  |  |
| --- | --- |
|  |  |
|  |

**// скопипастить, не скринить структуры**

**// использовать объединения полностью**

Вычислим объём памяти необходимый для хранения одной записи:

**sizeof(flat\_t)** == 4Б + 32Б + … + 4Б == **72Б**

**Набор функций**

// Добавить имена функциям (этот блок перенести в обращение с програмой)

Над структурой, представляющей собой запись в таблице, допустимо совершать следующие действия:

* Инициализация структуры квартиры нулевыми значениями
* Освобождение памяти, выделенное для данной структуры
* Преобразование информации из структуры в строку и наоборот
* Печать данных структуры в файл или на экран
* Считывание данных структуры из файла или консоли

Над таблицей данных разрешено выполнять следующие действия:

1. Инициализация таблицы нулевым количеством квартир
2. Освобождение памяти, выделенной для данной таблицы
3. Сортировка таблицы по одному из предложенных невариантных полей: адрес, площадь комнат, количество комнат
4. Добавление новой записи в конец таблицы
5. Считывание данных таблицы из файла
6. Вывод данных таблицы на экран консоли
7. Создание копии таблицы
8. Инициализация массива ключей для сортировки

**Описание алгоритма обработки данных**

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

Не смотря на то, что обработка таблицы ключей с использованием указателей является более удобным способом, чем хранение некоторых полей из исходной таблицы, **использование указателей снижает эффективность алгоритмов сортировки** т. к. сравнение происходит не «*на месте*» в таблице ключей, а «*где-то там*» по указателям.

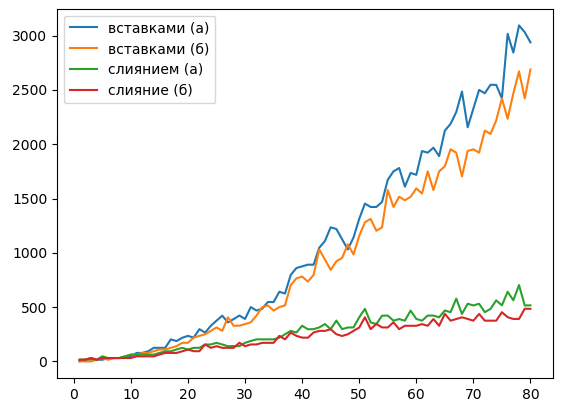
Поэтому для повышения эффективности по времени для данных алгоритмов **следует использовать таблицу значений полей, используемых при сортировке**.

Приведём сложность упомянутых алгоритмов:

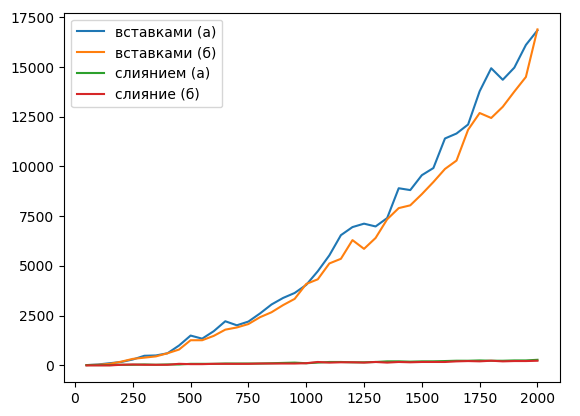
Сортировка вставками: **O(N2)**

Сортировка слиянием: **O(N\*log(N))**

Результаты тестирования в виде графика (с учетом времени на инициализацию новой таблицы для сортировки):



Для большего числа элементов:



Расчеты, проведённые в программе MathCAD для наиболее точного расчёта относительной эффективности двух алгоритмов **при использовании дополнительной таблицы ключей**:

Отностительная эффективность по времени:

**двух медленных сортировок: 3.1%**

**двух быстрых сортировок: 14.5%**

**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 **14.5%** для алгоритма сортировки слиянием). Однако уменьшение скорости оборачивается увеличением необходимого для работы объема памяти (приблизительно на **14.28%**).

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.

Также следует подчеркнуть, что использование указателей должно быть сведено к разумному минимуму, так как **разыменование указателей требует дополнительного процессорного времени**. И выбирая между таблицей указателей, и талицей значений полей для сортировки, **следует использовать таблицу значений полей для сортировки**.

**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 // чего-то // 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, вставка 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.