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

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 «ЗАПИСИ С ВАРИАНТАМИ, ОБРАБОТКА ТАБЛИЦ»

Студент Козлова Ирина Васильевна

Группа ИУ7 – 32Б

2020 г.

Оглавление

Описание условия задачи

- Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами). Упорядочить данные в ней по возрастанию ключей, где ключ – любое невариантное поле (по выбору программиста), используя:
  - саму таблицу
  - массив ключей

  (возможность добавления и удаления записей в ручном режиме обязательна).

- Произвести поиск информации по вариантному полю

- Упорядочить таблицу, по возрастанию ключей (где ключ – любое невариантное поле по выбору программиста (в моем случае — количество команат), 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. Обосновать выбор алгоритмов сортировки. Оценка эффективности должна быть относительной (в %).

- Интерфейс программы должен быть понятен неподготовленному пользователю. При разработке интерфейса программы следует предусмотреть:
  - указание формата и диапазона данных при вводе и (или) добавлении записей;
  - указание операций, производимых программой;
  - наличие пояснений при выводе результата;
  - возможность добавления записей в конец таблицы и удаления записи по значению указанного поля;
  - просмотр отсортированной таблицы ключей при несортированной исходной таблице;
  - вывод упорядоченной исходной таблицы;

- вывод исходной таблицы в упорядоченном виде, используя упорядоченную таблицу ключей
- вывод результатов сравнения эффективности работы программы при обработке данных в исходной таблице и в таблице ключей;
- вывод результатов использования различных алгоритмов сортировок.

- Одним из результатов работы программы должна быть количественная информация (лучше представить в виде таблицы) 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 также - объем занимаемой при этом оперативной памяти.
- При тестировании программы необходимо:
  - проверить правильность ввода и вывода данных (в том числе, отследить попытки ввода неверных по типу данных в вариантную часть записи);
  - обеспечить вывод сообщений при отсутствии входных данных («пустой ввод»);
  - проверить правильность выполнения операций;
  - отследить переполнение таблицы. При хранении исходных данных в файлах необходимо также проверить наличие файла и изменения информации в нем при удалении и добавлении данных в таблицу.

Мой вариант данной лабораторной работы:
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.

Описание технического задания

Входные данные:

1. **Файл с данными:** текстовый файл формата TXT. Разделителем в файле является символ пробел “ ”. Каждая новая запись таблицы в

обязательном порядке должна находиться на новой строке. Запись содержит :

- Город (до 50 символов)
- Улицу (до 50 символов)
- Номер дома (от 1 до 999)
- Номер квартиры (от 1 до 999)
- Признак первичности жилья (0 или 1)
- Вид жилья
  - первичное
    - с отделкой (0 или 1)
  - вторичное
    - год постройки (от 1930 до 2020)
    - количество прежних жильцов (от 1 до 15)
    - количество прежних собственников (от 1 до 15)
    - признак наличия животных (0 или 1)

2. **Целое число, представляющее собой номер команды**: целое число в диапазоне от 0 до 10.
3. **Дополнения к таблице**:
   - строковое или целочисленное поле, в зависимости от вводимой информации.

Выходные данные:

1. Полученная таблица (основная или таблица ключей) в отсортированном или неотсортированном виде (в зависимости от выполненной команды).
2. Характеристика сравнения вариантов сортировки таблицы.

Функции программы:

1. Загрузить таблицу из файла.
2. Добавить запись в конец таблицы.
3. Удалить запись из таблицы по количеству комнат.
4. Отсортировать таблицу ключей (выбор сортировки на выбор) и вывести ее на экран.
5. Отсортировать таблицу (выбор сортировки на выбор) и вывести таблицу на экран.
6. Отсортировать таблицу ключей и вывести исходную таблицу по отсортированной таблице ключей.

7. Сравнить время сортировки таблицы сортировками со сложностями O(n^2) и O(n*log(n)) и сравнение времени обычной сортировки и сортировки таблицы ключей.
8. Вывести все вторичное 2-х комнатное жилье в указанном ценовом диапазоне без животных.
9. Вывести таблицу на экран.
10. Очистка таблицы.
11. Вызов помощи (меню).

Обращение к программе:

Запускается через терминал. Так же можно собрать программу используя makefile и запустить ее с помощью команды run.

Аварийные ситуации:

1. Некорректный ввод номера команды.
   На входе: число, большее чем 12 или меньшее, чем 0.
   На выходе: сообщение «Invalid command entered, please re-enter!!!»
2. Некорректный ввод номера команды.
   На входе: пустой ввод.
   На выходе: сообщение «Invalid command entered, please re-enter!!!»
3. Файл пуст.
   На входе: пустой файл.
   На выходе: сообщение «File empty, please check your file!!!»
4. Выполнение какой-либо команды до выгрузки файла.
   На входе: целое число в диапазоне от 2 до 9 (номер команды).
   На выходе: сообщение «The table is empty.»
5. Превышение количества записей в конечной таблице.
   На входе: добавление новой записи при максимальном размере таблицы.
   На выходе: сообщение «The table is completely filled in!»
6. Неверный ввод строкового поля.
   На входе: строка, содержащая некорректные строковые литералы.
   На выходе: сообщение «Invalid city/street entered!»
7. Неверный ввод строкового поля.
   На входе: пустой ввод.
   На выходе: сообщение «Invalid city/street entered!»
8. Ввод недопустимого признака поля.

На входе: целое число, отличающееся от обусловленных допустимых значений для поля.
На выходе: сообщение «Invalid attribute entered.»

9. При поиске записей по указанному полю, нет подходящих записей.
На входе: таблица и параметры поиска (необходимые).
На выходе: «There are no apartments with this number.»

Описание структуры данных

Используемый именной тип данных table_r представляет собой структуру, определяющую таблицу целиком и содержащую одновременно массивы стуктр appartment_r appar[100}, key_r key[100], а так же размер таблицы, который находится в целочисленном поле func_var size.

```
typedef struct table
{
    appartment_r appar[100];
    key_r key[100];
    func_var size;
} table_r;
```

Именной тип appartment_r представляет записи в таблицу, содержащей все сведения о книге:

```
typedef struct appartment
{
    char address[101];
    func_var arrr;
    func_var room;
    int square;
    boolean is_prim_sec;
    prim_sec_r flat;
} appartment_r;
```

Поля структуры:

- **char address[110] –** адрес квартиры, содержащий город, улицу, номер дома, и номер квартиры;
- **func_var arrr –** площадь, которую занимает квартира;

- **func_var room –** количество комнат в квартире;
- **int square –** цена за метр квадратный;
- **boolean is_prim_sec -** признак первичности жилья (true — первичное, false — вторичное);
- **prim_sec_r flat –** вариативная часть, зависит от типа жилья;

Именной тип key_r представляет таблицу ключей:

```
typedef struct keys
{
    func_var room;
    func_var id;
} key_r;
```

Поля структуры:

- **func_var room –** количество комнат
- **func_var id;**

Тип prim_sec_r является вариантной частью:

```
typedef union prim_sec
{
    primary_r primary;
    secondary_r second;
} prim_sec_r;
```

В свою очередь primary_r и secondary_r являются именованными типами и отвечают за поля по типу жилья (первичное или вторичное):

```
typedef struct secondary
{
       func_var build_time;
       func_var previous_own_count;
       func_var last_tenants_count;
       boolean animals;
} secondary_r;
```

Поля структуры:

- **func_var build_time –** время постройки;
- **func_var previous_own_count –** количество последних жильцов;
- **func_var last_tenants_count –** количество предыдущих собственников;
- **boolean animals –** признак были ли животные (true – были, false – не было);

```
typedef struct primary
{
    boolean decoration;
}primary_r;
```

Поля структуры:

- **boolean decoration –** признак наличия отделки (true – есть отделка, false – нет отделки)

func_var именованный типа для работы с функциями и для того, чтобы занимать меньше памяти (удобно будет заменить его в случае смены ТЗ):

```
typedef short int func_var;
```

boolean является перечисляемым типом:

```
typedef enum BOOLEAN
{
    false = 0,
    true = 1
} boolean;
```

MAX_LEN_STRING = 100 максимально допустимая длина для текстового поля.

Описание алгоритма

1. Выводится меню данной программы.
2. Пользователь вводит номер команды из предложенного меню.

3. Пока пользователь не введет 0 (выход из программы), ему будет предложено вводить номера команд и выполнять действия по выбору.

НАБОР ТЕСТОВ

	Название теста	**Пользовательский ввод**	**Результат**
1	Некорректный ввод команды	45	Invalid command entered, please re-enter!!!
2	Пустой ввод	Пустой ввод	Invalid command entered, please re-enter!!! / Invalid house number entered / Invalid appartment number entered / Incorrect area size entered / Invalid number of rooms entered / Incorrect price per square meter entered
3	Некорректный ввод из файла (файл пустой)	In_02.txt	File empty, please check your file!!!
4	Некорректный ввод из файла (файл не существует)	In_03.txt	Some file error!!!
5	Добавление записи при максимальном размере таблицы	Ввод команды на добавление записи	The table is completely filled in.
6	Некорректный ввод для строкового поля	Qqqqqqq ….(51 буква q)	Invalid city/street entered.

	(город или улица)		
7	Некорректный ввод численного поля (буква)	q	Invalid house number entered / Invalid appartment number entered / Incorrect area size entered / Invalid number of rooms entered / Incorrect price per square meter entered
8	Некорректный ввод численного поля (дробь)	2,25	Invalid house number entered / Invalid appartment number entered / Incorrect area size entered / Invalid number of rooms entered / Incorrect price per square meter entered
9	Некорректный ввод признака (можно 0 или 1)	45 (допускается 0 или 1)	Invalid attribute entered
10	Ввод записей из верного файла (команда 1)	Ввод команды 1	Записи введены, таблица заполнена
11	Добавление верной записи в конец файла (команда 2)	Введена команда 2 и Запись введена верно	Запись успешно добавлена в коней файла
12	Удаление из таблицы (команда 3)	Введена команда 3 и указано не существующее значение в поле удаления	There are no apartments with this number.

13	Удаление из таблицы (команда 3)	Введена команда 3 и указано существующее значение в поле удаление (4).	All (7) apartments with the entered number have been deleted.
14	Сортировка таблицы ключей (команда 4)	Команда введена до заполнения таблицы.	The table is empty.
15	Сортировка таблицы ключей (команда 4)	Команда введена верно, при выборе сортировки введена неверная цифра (не 1 или 2)	Error choice sort.
16	Сортировка таблицы ключей (команда 4)	Команда введена верно.	Таблица отсортированных ключей выведена а экран.
17	Сортировка таблицы(команда 5)	Команда введена до заполнения таблицы.	The table is empty.
18	Сортировка таблицы(команда 5)	Команда введена верно, но при выборе сортировки введена неверная цифра (не 1 или 2)	Error choice sort.
19	Сортировка таблицы(команда 5)	Команда введена верно.	Отсортированная таблица выведена на экран.
20	Вывод таблицы по отсортированной таблицы ключей. (команда 6)	Команда введена до заполнения таблицы.	The table is empty.
21	Вывод таблицы по отсортированной таблицы ключей. (команда 6)	Команда введена верно.	Таблица выводится в отсортированном виде, причем сама

			не сортируется
22	Сравнение сортировок (команда 7)	Команда введена до заполнения таблицы.	The table is empty.
23	Сравнение сортировок (команда 7)	Команда введена верно.	Выводятся замеры времени различным видов сортировок.
24	Поиск по заданным параметрам (команда 8)	Команда введена до заполнения таблицы.	The table is empty.
25	Поиск по заданным параметрам (команда 8)	Введена неверная нижняя граница (буква или дробное число)	Invalid minimum or maximum borders are entered.
26	Поиск по заданным параметрам (команда 8)	Введена неверная верхняя граница (буква или дробное число)	Invalid minimum or maximum borders are entered.
27	Поиск по заданным параметрам (команда 8)	Нижняя граница больше верхней.	The lower border is larger than the upper one.
28	Поиск по заданным параметрам (команда 8)	Нет подходящих записей	No apartments found for your search.
29	Поиск по заданным параметрам (команда 8)	Команда введена верно и записи нашлись (хотя бы одна)	Выведена (ы) найденные записи на экран.

30	Вывод таблицы на экран (команда 9)	Команда введена до заполнения таблицы.	The table is empty.
31	Вывод таблицы на экран (команда 9)	Команда введена верно.	Таблица выведена на экран.
32	Вызов помощи (меню) (команда 11)	Команда введена верно.	Выведено меню на экран.
33	Выход (команда 0)	Команда введена верно.	Произведен выход.
32	Очистка таблицы (команда 10)	Команда введена верно.	Таблица очищена.

Оценка эффективности

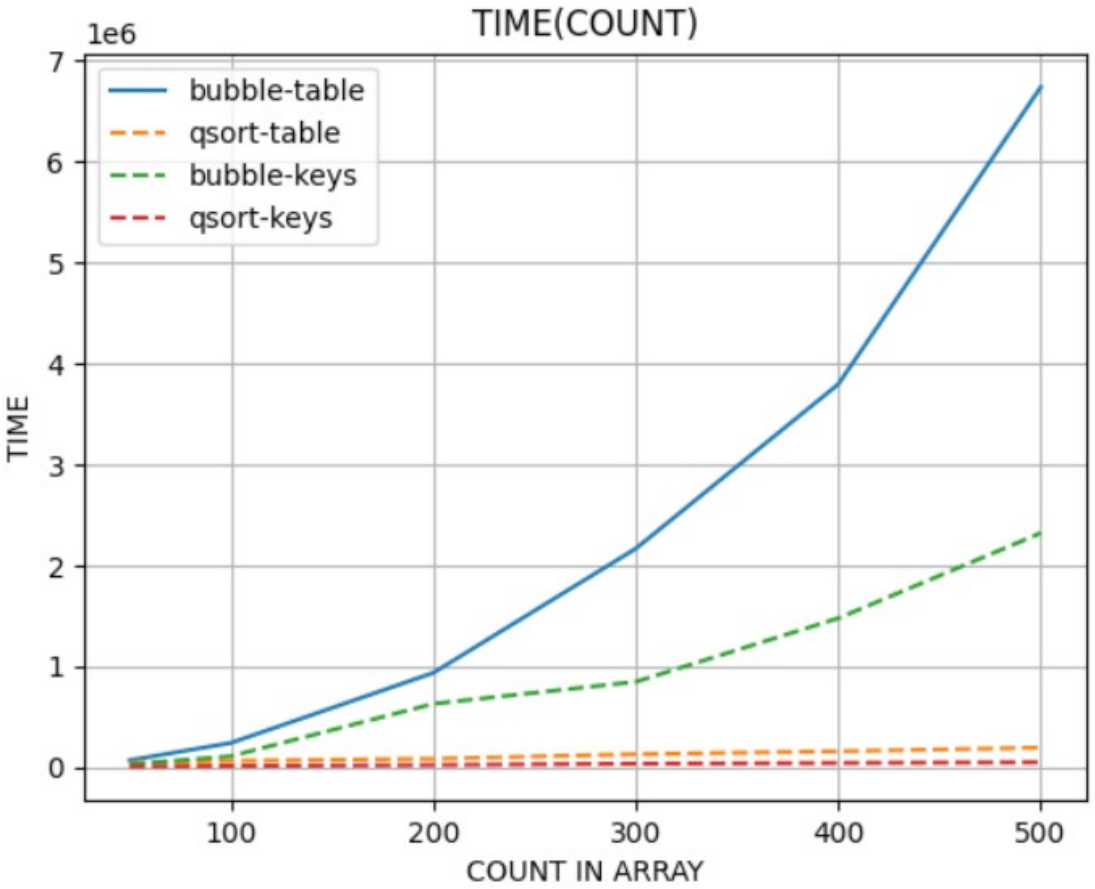
Измерения эффективности сортировок будут производиться в тактах процессора, с помощью специальной функции будут делаться замеры количества тактов, поэтому погрешность данного измерения минимальна. Частота процессора 1990000000 Гц.

Время сортировки (при сортировке таблица не выводится на экран, то есть замеряется время исключительно сортировки):

Количество записей	Таблица целиком		Таблица ключей	
	Сортировка «bubble»	Сортировка «qsort»	Сортировка «bubble»	Сортировка «qsort»
50	73825	45863	33002	11096
100	245172	67726	113675	17952
200	939153	88322	632705	27056
300	2169413	133282	850278	39350

400	3795797	162663	1479267	46417
500	6736887	198281	2322925	53395

График зависимости время от количества записей в таблице:



Объем занимаемой памяти:

Количество записей	Таблица целиком	Таблица ключей
50	6800	200
100	13600	400

200	27200	800
300	40800	1200
400	54400	1600
500	68000	2000

Таблица соотношений памяти и времени:

Количество записей	% памяти, занимаемый таблицей ключей от всей таблицы	Во сколько раз сортировка таблицы ключей быстрее сортировки всей таблицы (“bubble”)	Во сколько раз сортировка таблицы ключей быстрее сортировки всей таблицы (“qsort”)
50	~3%	~2 раз	~4 раз
100	~3%	~2 раз	~3,7 раз
200	~3%	~1,5 раз	~3,2 раз
300	~3%	~2,5 раз	~3,4 раз
400	~3%	~2,5 раз	~3,5 раз
500	~3%	~3 раз	~3,7 раз

Ответы на контрольные вопросы

1.Как выделяется память под вариантную часть записи?

Размер памяти, который выделяется под вариантную часть, равен максимальному по длине полю вариантной части. Эта память является общей для всех полей вариантной части записи.

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

При компиляции тип данных в вариантной части не проверяется. Поведение будет неопределенным из-за того, что невозможно корректно считать данные.

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

За правильностью выполнения операций с вариантной частью должен следить программист.

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

Таблица ключей представляет собой дополнительный массив (структура), содержащий индекс исходного элемента в исходной таблице и выбранный ключ.

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

Когда мы сортируем таблицу ключей, мы экономим время, так как перестановка записей в основной таблице (которая может содержать большой количество полей) отсутствует. Минус данного подхода в том, что для размещения таблицы ключей требуется дополнительная память. Кроме того, если использовать в качестве ключа символьное поле, то необходимо будет дополнительно обрабатывать данное поле в цикле, что увеличивает время выполнения, так же выбор данных из основной таблицы в порядке, определенном таблицей ключей, замедляет вывод. Если исходная таблица содержит небольшие число полей, то выгоднее обрабатывать данные в самой таблице.

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

Если будет производится сортировка самой таблицы, то необходимо использовать алгоритмы, требующие наименьшее количество операций перестановки. Если же сортировка производится по таблице ключей, то эффективнее использовать сортировки с наименьшей сложностью работы.

Вывод

Одно из преимуществ использования вариантной части записи состоит в том, что тратится меньше памяти, так как один участок памяти (размер равен наибольшему полю в вариантной части) используется сразу для всех значений вариантной части, что экономит память. Один из недостатков — контролировать правильность заполнения такой части должен сам программист, так как компилятор не может отследить ошибку.

Чем больше размер таблицы, которую мы сортируем, тем эффективнее использовать сортировку массива ключей, так же на маленьких размерах сортировка массива ключей значительно быстрее, чем сортировка самой таблицы. Но для хранения массива ключей необходимо использовать дополнительную память (в моем случае понадобилось относительно памяти, около 3% от исходной таблицы).

Можно отметить, что сортировку массива ключей неэффективно использовать при малых размерах таблицы. В данном случае лучше воспользоваться сортировкой самой таблицы, так как разница во времени не столь ссущественна, а использование дополнительной памяти сократится (но стоит отметить, что необходимо базироваться на начальном соотношении памяти).

Таким образом, прежде чем использовать таблицу ключей, необходимо проанализировать будет ли выигрыш по памяти или по времени, и в каком размере. Так например, если время выполнения примерно одинаковое, то не следует использовать таблицу ключей, так как будет тратится лишняя память.