Министерство образования и науки 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 (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

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

Студент, группа **Кононенко С., ИУ7-33Б**

2019 г.

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).

Ввести список литературы, содержащий фамилию автора, название книги, издательство, количество страниц, вид литературы (1: техническая – отрасль, отечественная, переводная, год издания; 2: художественная – роман, пьеса, стихи; 3: детская – сказки, стихи). Вывести список отечественной технической литературы по указанной отрасли.

Техническое задание

Входные данные:

1. **Файл с данными:** текстовый файл формата **CSV** (comma separated values). Разделителем в файле является символ точки с запятой “**;**”. Каждая новая запись таблицы в обязательном порядке должна находиться на новой строке.
2. **Целое число, представляющее собой номер команды:** целое число в диапазоне от **0** до **12**.
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. Количественная характеристика сравнения вариантов сортировки таблицы.

Функция программы: программа выполняет ряд функций, указанных при её первом запуске. Она позволяет:

1. Выгрузить таблицу из файла.
2. Вывести на экран выгруженную таблицу.
3. Добавить запись в таблицу.
4. Удалить запись из таблицы по количеству страниц книги.
5. Отсортировать таблицу ключей сортировкой $O(n^2)$.
6. Отсортировать таблицу ключей сортировкой O(nlogn).
7. Вывести таблицу ключей.
8. Вывести отсортированную исходную таблицу по количеству страниц в книге сортировкой $O(n^2)$.

9. Вывести отсортированную исходную таблицу по количеству страниц в книге сортировкой O(nlogn).
10. Вывести исходную таблицу, используя упорядоченную таблицу ключей
11. Вывести сравнение времени сортировки таблицы сортировками со сложностями $O(n^2)$ и O(nlogn) и сравнение времени обычной сортировки и сортировки с использованием
    массива ключей.
12. Вывести список отечетвенной технической литературы по указанной области.

В качестве сортировки с асимптотической сложностью $O(n^2)$ используется сортировка пузырьком, с O(nlogn) – алгоритм быстрой сортировки.

Обращение к программе: запускается из терминала.

Аварийные ситуации:

1. Некорректный ввод номера команды.
   На входе: число, большее чем 12 или меньшее, чем 0.
   На выходе: сообщение «Введена недопустимая команда! Повторите попытку.»
2. Некорректный ввод строки с именем файла или файл пуст.
   На входе: строка с полным путем к файлу, относительно текущей директории.
   На выходе: сообщение «Ошибка файла! Файл не существует или пустой!»
3. Выполнение какой-либо команды до выгрузки файла.
   На входе: целое число в диапазоне от 2 до 12 (номер команды).
   На выходе: сообщение «Файл не выгружен! Таблица пуста!»
4. Превышение количества записей в конечной таблице.
   На входе: добавление новой записи при максимальном размере таблицы.
   На выходе: сообщение «Достигнут максимальный размер таблицы!»
5. Неверный ввод строкового поля.
   На входе: строка, содержащая некорректные строковые литералы.
   На выходе: сообщение «Введено неверное строковое поле!»
6. Ввод недопустимого признака поля.
   На входе: целое число, отличающееся от обусловленных допустимых значений для поля.
   На выходе: сообщение «Введене число, выходящее за допустимый интервал!»
7. Ввод в вариантной части более чем одного истинного уникального поля.
   На входе: пара одновременно истинных признаков или тройка одновременно истинных (ложных) признаков.
   На выходе: сообщение «Логическое поле может быть истинно только в одном случае!»

Структуры данных

Используемый именной тип данных **aio_table_t** представляет собой структуру, определяющую таблицу целиком и содержащую одновременно массивы структур ***book_t* *main_table**, ***book_key_t* *key_table** а также целочисленное поле ***int* size_of_table**. Данный тип описывается как:

```
typedef struct
{
   book_t *main_table;
   book_key_t *key_table;
   int size_of_table;
} aio_table_t;
```

Именной тип данных **book_key_t** является представлением таблицы ключей и описывается как:

```
typedef struct
{
   int book_table_index;
   int page_count;
} book_key_t;
```

Поля структуры:

- **book_table_index –** индекс записи в основной таблице записей.
- **page_count –** количество страниц – ключ, по которому будет сортироваться таблица.

Именной тип данных **book_t** является представлением записи в таблице, содержащей все сведения о книге и описывается как:

```
typedef struct
{
   char author_last_name[MAX_STRING_FIELD_SIZE];
   char book_name[MAX_STRING_FIELD_SIZE];
   char publisher[MAX_STRING_FIELD_SIZE];
   int page_count;
   literature_t book_type;
   variative_t variative_part;
} book_t;
```

Поля структуры:

- **author_last_name** – фамилия автора книги.
- **book_name** – название книги.
- **publisher** – издатель книги.
- **page_count** – количество страниц в книге.
- **book_type** – тип литературы.
- **variative_part –** вариативная часть, зависящая от типа литературы.

В свою очередь **literature_t** является перечисляемым типом и описывается как:

```
typedef enum literature
{
   technical=1,
   fiction,
   kids
} literature_t;
```

Объединение **variative_t** является вариантной частью и описывается как:

```
typedef union
{
   technical_t technical_book;
   fiction_t fiction_book;
   kids_t kids_book;
} variative_t;
```

Отсюда **techincal_t**, **fiction_t** и **kids_t** являются именованными типами, отвечающими за поля в одноименном типе литературы:

```
typedef struct
{
   char field[MAX_STRING_FIELD_SIZE];
   boolean_t is_national;
   boolean_t is_translated;
   int release_year;
} technical_t;
```

Поля структуры:

- **field –** отрасль данной книги.
- **is_national –** признак, является ли данная книга отечественной.
- **is_translated –** признак, является ли данная книга переведенной.
- **release_year –** год издания данной книги.

typedef struct
{
   ***boolean_t* is_novel;**
   ***boolean_t* is_play;**
   ***boolean_t* is_poetry;**
} *fiction_t*;

Поля структуры:

- **is_novel –** признак, является ли данная книга романом.
- **is_play –** признак, является ли данная книга пьесой.
- **is_poetry –** признак, является ли данная книга сборником стихотворений.

typedef struct
{
   ***boolean_t* is_fairytale;**
   ***boolean_t* is_poetry;**
} *kids_t*;

Поля структуры:

- **is_fairytale –** признак, является ли данная книга сборником сказок.
- **is_poetry** – признак, является ли данная книга сборником стихотворений.

boolean_t является перечисляемым типом и описывается как:

typedef enum boolean
{
   FALSE,
   TRUE
} *boolean_t*;

MAX_STRING_FIELD_SIZE = 256 – максимальная допустимая длина для текстового поля.

Алгоритм

1. Пользователь вводит номер команды из меню.
2. Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять действия с таблицей.

Тесты

	Тест	Пользовательский ввод	Результат
1	Некорректный ввод комманды	13	Введена недопустимая команда! Повторите попытку.
2	Некорректный ввод имени файла	data/data.txxt	Ошибка файла! Файл не существует или пустой!
3	Выбор команды до выгрузки файла	2 (до этого не была нажата 1)	Файл не выгружен! Таблица пуста!
4	Добавление записи при максимальном размере таблицы	Ввод записи	Достигнут максимальный размер таблицы!
5	Некорректный ввод строкового поля	NonASCIIliteral	Введено неверное строковое поле!
6	Некорректный ввод целочисленного поля	a	Введене число, выходящее за допустимый интервал!
7	Некорректный ввод целочисленного поля	4 (при допустимых значениях от 1 до 3)	Введене число, выходящее за допустимый интервал!
8	Ввод в вариантной части более чем 1 уникального поля	1 1 0	Логическое поле может быть истинно только в одном случае!
9	Ввод в вариантной части всех ложных полей	0 0 0	Логическое поле может быть истинно только в одном случае!
10	Ввод записи в таблицу	Запись введена корректно	Запись добавлена в таблицу
11	Удаление из таблицы записей по ключу	513 (при наличии 3 подходящих полей)	Удалено(-а) 3 записей(-ь)
12	Удаление из таблицы записи по ключу	999 (при отсуствии подходящих полей)	Удалено(-а) 0 записей(-ь)
13	Корректный ввод команды	3	Добавление записи в таблицу

Оценка эффективности

Измерения эффективности сортировок будут производиться в единицах измерения – тактах процессорв. Для измерения была специально написана ассемблерная функция, поэтому погрешность измерений минимальна. При записи результатов использовалось среднее количество тактов, полученное по результатам 10 измерений.

Время сортировки*

Количество записей	Сортировка пузырьком		Быстрая сортировка	
	Исходная таблица	Таблица ключей	Исходная таблица	Таблица ключей
50	2780268	73884	1576203	25864
100	5174736	349872	2768808	62832
150	8476456	539338	4037900	85500
200	13485184	799512	6036096	116712
250	18818800	1198512	9117288	144192
300	22595916	1474588	10000881	183108

*при сортировке исходной таблица таблица так же выводилась на экран, поэтому показатели не гораздо убедительнее превосходят более асимптотически сложную сортировку.

Объём занимаемой памяти (в байтах):

Количество записей	Исходная таблица*	Таблица ключей
50	52200	400
100	104400	800
150	156600	1200
200	208800	1600
250	261000	2000
300	313200	2400

*большой размер обусловлен большим количеством строковых полей, каждое из которых занимает по 256 байт.

Количество записей	Занимаемый % массива ключей всей таблицы	Рост скорости сортировки массива ключей по сравнению с таблицей (сортировка пузырьком)	Рост скорости сортировки массива ключей по сравнению с таблицей (быстрая сортировка)
50	~1%	~38 раз	~61 раз
100	~1%	~14 раз	~44 раза
150	~1%	~15 раз	~47 раз
200	~1%	~17 раз	~52 раза
250	~1%	~16 раз	~63 раза
300	~1%	~15 раз	~55 раз

Контрольные вопросы

1. Как выделяется память под вариантную часть записи?

Размер памяти, выделяемый 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, не соответсвтующие описанным?

Тип данных в вариантной части при компиляции не проверяется. Из-за того, что невозможно корректно прочитать данные, поведение будет неопределенным.

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

Контроль за правильностью выполнения операций с вариантной частью записи возлагается на программиста.

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

Дополнительный массив (структура), содержащий индекс элемента в исходной таблице и выбранный ключ. Она нужна для оптимизации сортировки.

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

В случае, если мы сортируем таблицу ключей, мы экономим время, так как перестановка записей в исходной таблице, которая может содержать большое количество полей, отсутствует. С другой стороны, для размещения таблицы ключей требуется дополнительная память. Кроме того, если в качестве ключа используется символьное поле записи, то для сортировки таблицы ключей необходимо дополнительно обрабатывать данное поле в цикле, следовательно, увеличивается время выполнения. Выбор данных из основной таблицы в порядке, определенном таблицей ключей, замедляет вывод. Если исходная таблица содержит небольшое число полей, то выгоднее обрабатывать данные в самой таблице.

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

Если обработка данных производится в таблице, то необходимо использовать алгоритмы сортировки, требующие наименьшее количество операций перестановки. Если сортировка производится по таблице ключей, эффективнее использовать сортировки с наименьшей сложностью работы.

Вывод

Чем больше размер таблицы, тем эффективнее сортировка массива ключей, но даже на маленьких размерах таблицы, сортировка массива ключей происходит быстрее, чем сортировка самой таблицы. Однако, для хранения массива ключей нужна дополнительная память. В моем случае понадобилось очень мало памяти (относительно исходной таблицы всего 1%), т. к. изначально в структуре таблицы много места выеделено под хранение символьных полей. Стоит отметить, что использование массива ключей неэффективно при небольших размерах самой таблицы. В таком случае эффективнее просто отсортировать таблицу, так как разница во времени будет несущественна, а затраты на память сократятся (но опять же, следует базироваться на стартовом соотношении памяти).

Также заметна динамика падения прироста скорости сортировки по ключам: у более простой асимптотически сортировки скорость падения выше, чем у более сложной.