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 Щербина Михаил Александрович

Группа ИУ7 – 35Б

Описание условия задачи

- Создать таблицу содержащую не менее 40-ка записей, (записи с вариантами). Упорядочить данные в ней по возрастанию ключей, где ключ - любое невариантное поле (по выбору программиста, мой выбор: название книги), используя 2 алгоритма сортировки, а так же:
    - Саму таблицу
    - Массив ключей
- Предусмотреть добавление и удаление записей в ручном режиме
- Произвести поиск информации по вариантному полю.
- Оценить эффективность этих алгоритмов (по времени и по объему памяти) при различной реализации программы. Обосновать выбор алгоритмов сортировки. Оценка эффективности должна быть относительной (в %)
- Оценить эффективность этих алгоритмов (по времени и памяти) при различной реализации программы. Обосновать выбор алгоритмов сортировки. Оценка должна быть относительной в %.
- Интерфейс программы должен быть понятным неподготовленному пользователю. При разработке следует предусмотреть
    - Указание формата и диапазона данных при вводе и добавлении
    - Указание операций
    - Наличие пояснений при выводе результата
    - Возможность добавления записей в конец таблицы
    - Возможность удаления по значению указанного поля
    - Просмотр отсортированной таблицы ключей при несортированной исходной
    - Вывод упорядоченной исходной таблицы
    - Вывод исходной таблицы в упорядоченном виде используя таблицу ключей
    - Вывод результатов сравнения эффективности работы программы при обработке данных в исходной таблице и 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 таблицы и таблицы ключей двумя алгоритмами сортировки (при этом не забыть оценить также время выборки данных из основной таблицы с использованием таблицы ключей), а также объем занимаемой оперативной памяти.
- При тестировании необходимо
    - Проверить правильность ввода и вывода данных (в том числе отследить попытки ввода неверных данных в вариантную часть записи)
    - Обеспечить вывод сообщений при отсутствии входных данных (пустой ввод)
    - Проверить правильность выполнения операций
    - Отследить переполнение таблицы.

Мой вариант

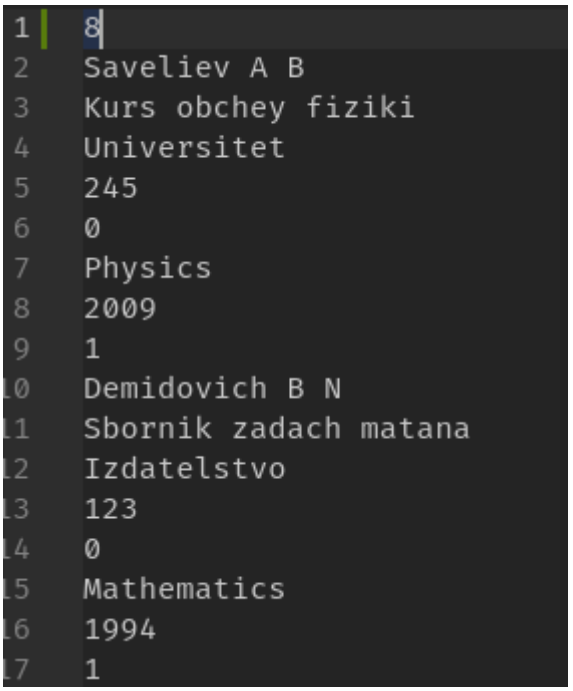
```
/*
 * Ввести список литературы, содержащий фамилию автора,
 * название книги, издательство, количество страниц, вид
 * литературы
 * (1: техническая – отрасль, отечественная, переводная, год
    издания
 *  2: художественная – роман, пьеса, стихи; 3: детская – сказки,
    стихи).
 *  Вывести список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

Описание технического задания

Входные данные

Файл с данными: текстовый файл. В первой строке содержится кол-во записей. Записи и их поля записаны подряд, разделены символом ‘\n’. Запись содержит название книги, автора,

издателя, кол-во страниц, тип книги, и соответствующую информацию для вариативной части.



- Название книги, до 64 символов
- Автора, до 64 символов
- Название издателя, до 64 символов
- Кол-во страниц (ограничено типом int), положительный
- Вариативная часть
    - Техническая
        - Отрасль, до 64 символов
        - Перечисление: Отечественная / переводная
        - Год издания, ограничен типом int, положительный
    - Художественная
        - Перечисление: Роман / пьеса / стихи
    - Детская
        - Перечисление: Сказки / стихи

Выходные данные

Полученная таблица
Сравнение сортировок

Функции программы

1. Прочитать таблицу из файла (путь к файлу не указывается)
2. Напечатать таблицу

3. Напечатать таблицу ключей
4. Напечатать таблицу используя таблицу ключей
5. Добавить запись в таблицу
6. Обновлять таблицу ключей (указать другое поле, после сортировки)
7. Отсортировать таблицу
8. Сортировать таблицу ключей
9. Вывести строки, поля которых соответствуют заданному ключу (фильтровать)
10. Удалить строки, поля которых соответствуют заданному ключу
11. Сравнить время сортировки таблицы со сложностями O(n^2) и O(n*log(n)) и сравнение времени сортировки главной таблицы и таблицы ключей.
12. Поиск технической литературы по заданному году
13. Завершить работу

Обращение к программе:

Запускается через терминал, собирается с помощью cmake (>=3.1) и make.

Аварийные ситуации

1. Некорректный ввод номера команды / пункта в меню
    a. На входе: число, большее чем ко-во пунктов меню, меньшее 0, не равное коду выхода (123).
    b. На выходе: ввод игнорируется, меню выводится заново
2. Пустой / некорректный / несуществующий файл с записями
    a. Пустой / некорректный / несуществующий файл
    b. Немедленное завершение работы программы, вывод ошибки
3. Попытка выполнить операцию над таблицей ключей без ее формирования / после изменения главной таблицы
    a. Добавление / удаление элемента, не успели обновить таблицу ключей

    b. Обновление при вызове соответствующей операции (печати ключей, т.д.)
4. Превышение кол-ва записей в таблице
    a. Добавление нового элемента при size + 1 == capacity
    b. Перевыделение таблицы с capacity * 2
5. Неверный ввод новой записи при добавлении
    a. Неверный формат одного из полей
    b. Игнорирование добавления, вывод сообщения об ошибке

Описание структуры данных

Главная таблица - table_t - это (вектор?) с двумя указателями:

```
typedef struct
{
   book_t *books;
   book_key_t *keys;
   int size;
   int capacity;
} table_t;
```

capacity это максимальное зарезервированное кол-во элементов. size - кол-во заполненных мест. books - указатель на массив книг. keys - массив ключей.

Структура - книга - все поля статические, хранятся на стеке

```
typedef struct
{
   char lastname[SSIZE];
   char title[SSIZE];
   char publisher[SSIZE];
   int pages;
   int type;
   book_kind_t kind;
} book_t;
```

Все поля были описаны выше.

```
typedef union
{
```

```
    tech_book_t tech;
    fiction_book_t fiction;
    kid_book_t kid;
} book_kind_t;
```

Это объединение с структурными типами соответствующих видов книг. Дам пояснение в виде комментариев:

```
enum translation_t
{
   native, // отечественная
   translated, // переведенная
};

typedef struct
{
   char field[SSIZE]; // отрасль, 63 символа
   int year; // год
   int language;  // язык
} tech_book_t;

enum genre_t // жанр художественной литературы
{
   genre_novel,
   genre_poem,
   genre_play,
};

typedef struct // художественная книга
{
   int genre;
} fiction_book_t;

enum kid_book_e // жанр детской книги
{
   kid_fairytail,
   kid_poem,
};

typedef struct // детская книга
{
   int genre;
} kid_book_t;
```

Структура - ключ.

```
enum book_key_base_type // на что будет указывать void* key
{
    key_string, // на строку
    key_int // на int
};

typedef struct
{
    void *key; // указатель на поле записи
    int type; // имя поля записи (название, издательство,
     n_страниц…)
    int base_type; // тип типа поля записи (int / string)
    int pos_actual; // позиция в главной таблице
    int pos_fake; // условная позиция в таблице ключей
} book_key_t;
```

Описание Алгоритма

1. Вывод пунктов меню
2. Обработка ввода пользователя
3. Вывод результатов обработки
4. Выход по необходимости

Набор Тестов

Index	Описание	Ввод	Результат
1	Некорректный выбор пункта в меню (пустой		Игнорирование, вывод меню

	ввод)		заново
2	Некорректный выбор пункта в меню (несуществующий пункт)		Игнорирование, вывод меню заново
3	Некорректный выбор пункта в меню (ввод плохой строчки)		Игнорирование, вывод меню заново
4	Некорректный файл с записями (пустой)	файл	Немедленное завершение программы с выводом ошибки
5	Некорректный файл с записями (сломанная запись)	файл	Немедленное завершение программы с выводом ошибки
6	Некорректный файл с записями (кол-во записей = 0 или записей больше чем нужно)	файл	Немедленное завершение программы с выводом ошибки
7	Добавление записи при достижении макс размера capacity	Комманда, запись	Перевыделение таблицы
8	Некорректный ввод строкового поля записи при добавлении записи (превышение лимита 63 символа)	A x 64	Игнор ввода, вывод сообщения об ошибке
9	Некорректный ввод численного поля записи при добавлении записи	asd	Игнор ввода, вывод сообщения об ошибке
10	Некорректный ввод поля перечисления записи при добавлении записи	234 или asd	Игнор ввода, вывод сообщения об ошибке
11	Ввод записей из корректного файла	комманда	Отсутствие ошибки

12	Добавление записи в таблицу	комманда	Запись таблицы в файл
13	Некорректный файл с записями (не существует / нет прав) при чтении / записи	путь	Вывод сообщения об ошибке, таблица не загружается / пишется
14	Добавление записи	Запись	Запись добавляется
15	Удаление записей, которые есть	Ключ	Записи удаляются
16	Удаление записи, которых нет	Ключ	Выводится “Nothing to delete no matches found”
17	Фильтр записей которые есть	Ключ	Записи выводятся
18	Фильтр записей которых нет	Ключ	Вывод “No matches found”
19	Сортировка таблицы	Индекс ключа	Таблица сортируется
20	Сортировка таблицы, неверный ввод выбор	Неверный ввод выбор	Ввод игнорируется, пользователю предлагается заново выбрать
21	Вывод таблицы, таблица заполнена	Команда	Таблица выводится
22	Вывод таблицы, таблица пуста	Команда	Выводится *empty table*
23	Вывод таблицы ключей, заполнена	Команда	Выводится таблица

24	Вывод таблицы ключей, пуста	Команда	Выводится *empty_table*

25	Поиск тех. Литературы, введен сущ. год	Комманда	Выводятся записи в сериализованном формате
26	Поиск тех. Литературы, введен несущ. год	Комманда	Выводится “Nothing found”

Оценка Эффективности

CPU: AMD Ryzen 7 3800X (16) @ 3.900GHz

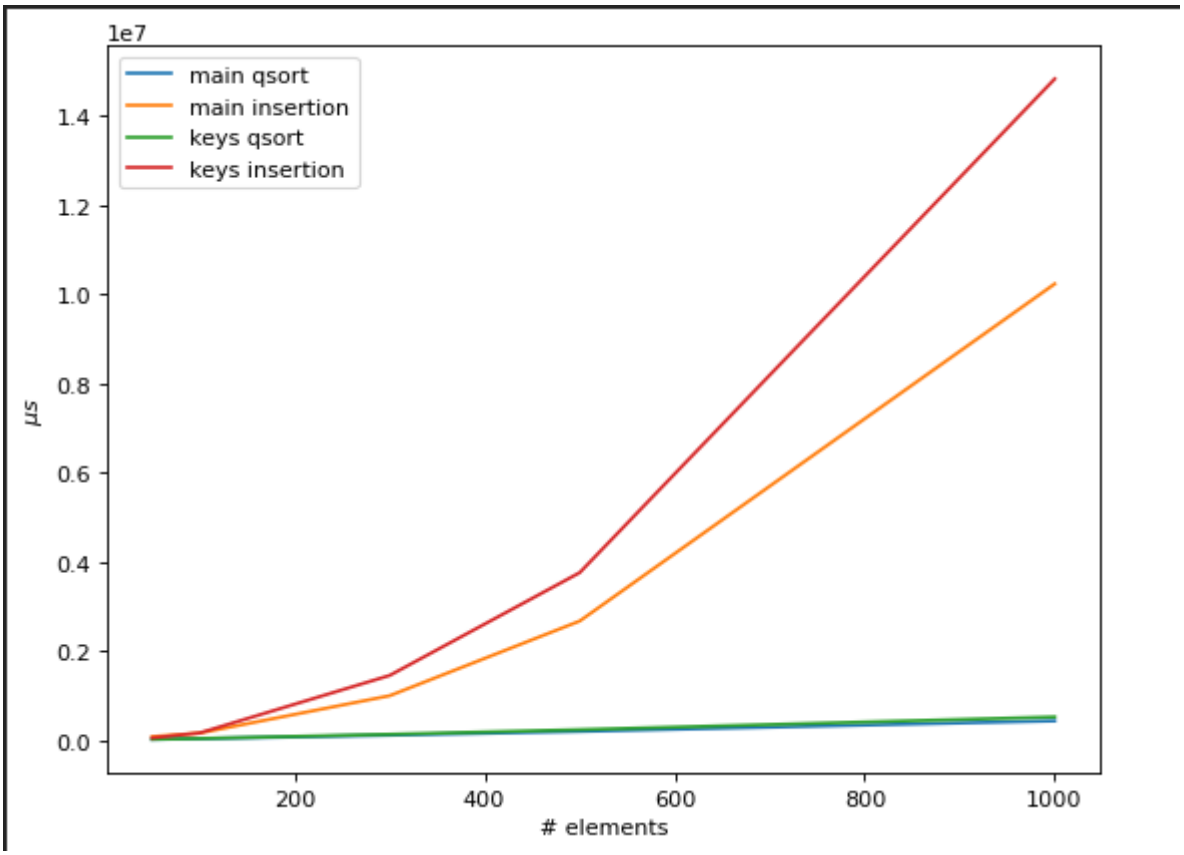
Измерения производятся в тактах процессора с помощью специальной функции. Частота процессора 3.9 ГГц.

Из-за того, что я сделал поддержку сортировки ключей с любым полю, таблица ключей стала неэффективна по времени.

Время сортировки:

Idx	Size (elems)	Таблица, insertion sort	Таблица, quick sort	Таблица ключей, insertion sort	Таблица ключей, quick sort
0	50	61095	13694	40506	13029
1	100	214858	28290	149469	28894
2	300	1825958	101855	1362153	110756
3	500	3111775	183375	15028898	202682

4	1000	12428137	408130	15028898	446528



Использование памяти (в байтах)

Idx	Таблица	Таблица Ключей
50	14808	1204
100	29608	2404
300	88808	7204
500	148008	12004
1000	296008	24004

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	~8%	~2 раз	~2 раз
100	~8%	~1.6 раз	~1.5 раз
300	~8%	~1,1 раз	~1.2 раз
500	~8%	~1.2 раз	~1.2 раз
1000	~8%	~1.08 раз	~1.2 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.

Память, выделяемая под вар. Часть равна максимальному по длине полю, из которой вариативная часть состоит. Память является общей для всех полей вар. части

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

Неопределенное поведение, т.к. при компиляции тип данных в вариантной части не проверяется.

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?

Это дополнительный массив структур, содержащих следующие поля: индекс элемента в исходной таблице и выбранный ключ.

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

Сортируя таблицу ключей, мы экономим время. Перестановка записей в основной таблице отсутствует. Минус данного подхода: для размещения таблицы ключей требуется дополнительная память. 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 (N*Log(N)). Если же сортировка производится по таблице ключей, то эффективнее использовать сортировки с наименьшей сложностью работы.

Вывод

Преимуществом использования вариативной части заключается в уменьшении потребления оперативной памяти по сравнении с хранением нескольких полей структуры для разных типов. Недостатком является сложность контролирования правильности операций. Делать это придется программисту.

Чем больше размер таблицы, которую мы сортируем, тем эффективнее использовать сортировку массива ключей, также 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).

Стоит отметить: сортировку массива ключей неэффективно использовать при малых размерах таблицы. В данном случае лучше воспользоваться сортировкой самой таблицы, так как разница во времени не столь существенна, а использование дополнительной памяти сократится (зависит от реализации, у меня не сократится).

До использования таблицы ключей следует проанализировать, будет ли выигрыш по памяти и его размер. Если время выполнение одинаковое, не следует использовать таблицу ключей в виду выделения лишней памяти (зависит от реализации).

Использованные функции:

Сравнение книг

cmp_func_t book_cmp_f(int type);

Геттер ключа из книги

book_key_t book_get_key(book_t *self, int type);

Сравнение ключей

int key_cmp(const void *a, const void *b);

Тип ключа (перечисление)

int get_base_type(int key_type);

Вставить в таблицу

void table_insert(table_t *self, book_t book);

3

table_t table_read_file(FILE *fin, int n, int *ec);

void table_serialize_file(table_t *self, FILE *fout);

void table_update_keys(table_t *self, int type);

void table_sort_keys(table_t *self, sort_func_t sort);

void table_sort(table_t *self, sort_func_t sort, int type);

int *table_filter(table_t *self, book_key_t *key, int *n);

void table_remove(table_t *self, int *indexes, int n);

int *table_filter_tech_year(table_t *self, int year, char *field, int *n);

size_t table_size(table_t *self);

size_t table_keys_size(table_t *self);

int64_t ticks(void);