Лабораторная работа № 3 «Работа с просматриваемыми и упорядоченными таблицами»

Введение

Необходимо спроектировать и разработать на языке C:

1. Прикладную программу, позволяющую пользователю в диалоговом режиме работать с таблицей.
2. Библиотеку, предоставляющую функциональность по работе с таблицей, размещенной в основной памяти (лабораторная работа № 3а).
3. Библиотеку, предоставляющую фунциональность по работе с таблицей, размещенной во внешней памяти (лабораторная работа № 3б).

Структура таблицы

Таблица задаётся структурой:

```
struct Table {
    // указатель на пространство ключей
    KeySpace *ks;

    // опциональное поле, ограничивающее размер пространства ключей,
    // его наличие определяется типом организации соответствующего пространства,
    // в соответствии с условиями индивидуального задания

    // размер области пространства ключей
    IndexType msize;

    // опциональное поле с текущим количеством элементов
    // в пространстве ключей,
    // его наличие определяется типом организации соответствующего пространства,
    // в соответствии с условиями индивидуального задания

    // количество элементов в области пространства ключей
    IndexType csize;
};
```

Структура элемента таблицы

Элементы таблицы задаются следующей структурой:

```
struct Item {
    // указатель на информацию
    InfoType *info;

    // опциональные поля, для оптимизации выполнения операций,
```

```
    // состав и наличие которых должны быть обоснованы:

    // ключ элемента
    KetType key;
    // связь с элементом пространства ключей по индексу
    IndexType ind;
    // связь с элементом пространства ключей по указателю
    PointerType *p1;
};
```

Операции, поддерживаемые таблицей

Должны быть предусмотрены следующие операции:

1. включение нового элемента в таблицу с соблюдением ограничений на уникальность значений ключевой информации;
2. удаление из таблицы элемента по заданному значению ключа;
3. поиск в таблице элемента по заданному значению ключа, результатом поиска должны быть копии всех найденных элементов со значениями ключей;
4. вывод содержимого таблицы в стандартный поток;
5. импорт данных из текстового файла;
6. особые операции, в соответствии с индивидуальным заданием.

Задачи

Основные задачи

Необходимо разработать два варианта решения задачи (лабораторные работы 3а и 3б):

1. Сама таблица и информация, относящаяся к элементам таблицы, хранятся в основной памяти.
2. Сама таблица и информация, относящаяся к элементам таблицы, хранятся во внешней памяти (используется двоичный файл произвольного доступа). Описатель таблицы и описатели пространств ключей считывается из файла (или создаются в первый раз) в начале сеанса работы и записывается в файл в конце сеанса работы. Информация, относящаяся к элементам таблицы, записывается в файл сразу же при выполнении операции включения в таблицу и в основной памяти не хранится (возможно за исключением элемента, с которым производится текущая операция). Все операции выполняются с описателем таблицы и пространств ключей, размещенными в основной памяти. Все структуры данных модифицируются соответствующим образом (замена указателей на смещение в файле и т.п.). Имя файла вводится по запросу из программы и хранится в описателе таблицы.

Дополнительные задачи

Существует ряд дополнительных задач, не обязательных к выполнению, но позволяющих получить дополнительные баллы.

ИНФОРМАЦИЯ О ДОПОЛНИТЕЛЬНЫХ БАЛЛАХ БУДЕТ УТОЧНЕНА ПОЗДНЕЕ

Дополнительные задачи:

1. * Реализация поиска как итератора одним из возможных способов (например, в виде функции, которая при каждом вызове возвращает очередной из найденных элементов).

2. ** Аналогично п. 2, но все операции выполняются с пространствами ключей, размещенными во внешней памяти, в основной памяти может храниться только описатель таблицы.

3. *** Аналогично предыдущему заданию, но с реализацией буферизации файловых операций (можно считывать и записывать по несколько записей) и кэширования записей (тип кэша и стратегии управления кэшем выбираются по согласованию с преподавателем).

Примечания

1. Логически законченные части алгоритма решения задачи должны быть оформлены в виде отдельных функций с параметрами. Использование глобальных переменных не допускается.

2. Функции для работы с таблицами не должны быть диалоговыми, т. е. они должны принимать все необходимые данные в качестве параметров и возвращать результат работы в виде соответствующих структур данных и кодов ошибок (исключение: функции вывода таблицы в стандартный поток вывода или записи файл).

3. Диалоговые функции должны использовать описанные выше функции, т. е. должен быть реализован принцип Model-View-Controller (MVC).

4. Программа должна осуществлять проверку корректности вводимых данных и, в случае ошибок, выдавать соответствующие сообщения, после чего продолжать работу.

5. В случае возникновения ошибочных ситуаций при выполнении операций с таблицами программа должна выводить соответствующие сообщения, после чего продолжать работу.

6. Для работы с таблицами, размещенными во внешней памяти, должна использоваться модифицированная структура, определяющую элемент таблицы, в которую включена длина информации и её смещение в файле.

7. Для работы с таблицами, размещенными во внешней памяти, должны использовать функции `fread()` и `fwrite()`, которым в качестве аргумента должна передаваться реальная длина информации.

8. Для сборки программы и библиотек должна использоваться система сборки (например: Make или CMake).

9. Библиотеки и прикладная программа должны собираться независимо друг от друга.

10. Программа должна корректным образом работать с памятью, для проверки необходимо использовать соответствующие программные средства, например: valgrind, санитайзеры, встроенные в IDE средства и т.д.