Лабораторная работа №3 «Работа с таблицами»

Введение

Необходимо спроектировать и разработать на языке C две программы, осуществляющие работу с таблицами.

Структура таблицы

Таблица задается структурой:

```
struct Table {
    // указатель на первое пространство ключей
    KeySpace1 *ks1;
    // указатель на второе пространство ключей
    KeySpace2 *ks2;

    // опциональные поля, ограничивающие размер пространства ключей,
    // их наличие определяется типом организации соответствующего пространства,
    // в соответствии с условиями индивидуального задания

    // размер области первого пространства ключей
    IndexType1 msize1;
    // размер области второго пространства ключей
    IndexType2 msize2;

    // опциональные поля с текущим количеством элементов
    // в пространстве ключей,
    // их наличие определяется типом организации соответствующего пространства,
    // в соответствии с условиями индивидуального задания

    // количество элементов в области первого пространства ключей
    IndexType1 csize1;
    // количество элементов в области второго пространства ключей
    IndexType2 csize2;
};
```

Операции, поддерживаемые таблицей

Должны быть предусмотрены следующие операции:

1. включение нового элемента в таблицу с соблюдением ограничений на уникальность ключей в соответствующих ключевых пространствах (см. предыдущие пункты задания) и уникальность составного ключа `(key1, key2)`;
2. поиск в таблице элемента, заданного составным ключом;
3. удаление из таблицы элемента, заданного составным ключом;

4. поиск в таблице элемента по любому заданному ключу; результатом поиска должна быть копии всех найденных элементов со значениями ключей;
5. удаление из таблицы всех элементов, заданного ключом в одном из ключевых пространств;
6. вывод содержимого таблицы на экран (или текстовый файл), при этом формат вывода должен соответствовать элемента таблицы;
7. особые операции, определяемые в соответствующем пространстве ключей.

Задачи

Основные задачи

Необходимо разработать два следующих варианта программы (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.

За задачу, отмеченную «*», можно получить до 10% дополнительных баллов (до 1 балла по десятибалльной шкале), за задание, отмеченное «**» — до 20% дополнительных баллов (до 2 баллов по десятибалльной шкале), а за задание, отмеченное «***» — до 30% дополнительных баллов (до 3 баллов по десятибалльной шкале).

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 кеширования записей (тип кэша и стратегии управления кэшем выбираются по согласованию с преподавателем).

Упрощение задания

Возможна реализация программы в упрощенным виде только с одним пространством ключей (выбирается из основного задания по согласованию с преподавателем) и с упрощением структур данных, задающих таблицу. В данном случае за задачу возможно получение не более 60% от возможных баллов (до 6 баллов по десятибалльной шкале).

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. Программы, реализующие работу с таблицами, размещенными во внешней памяти, должны использовать модифицированную структуру, определяющую элемент таблицы, в которую включена длина информации и её смещение в файле.

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

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

Вариант №049

Написать программу для работы с таблицей, использующей два пространства ключей, по запросам оператора.

Каждый элемент таблицы имеет следующую структуру:

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

    // опциональные поля, для оптимизации выполнения операций,
    // состав и наличие которых должны быть обоснованы:

    // ключ элемента из 1-го пространства ключей
    KeyType1 key1;
    // ключ элемента из 2-го пространства ключей
    KetType2 key2;
    // связь с элементом 1-го пространства ключей по индексу
    IndexType1 ind1;
    // связь с элементом 2-го пространства ключей по индексу
    IndexType2 ind2;
    // связь с элементом 2-го пространства ключей по указателю
    PointerType1 *p1;
    // связь с элементом 2-го пространства ключей по указателю
    PointerType2 *p2;
};
```

В таблице не могут присутствовать два элемента с одинаковыми составными ключами `(key1, key2)`.

Первое пространство ключей организовано как просматриваемая таблица, организованная списком; каждый элемент таблицы имеет следующую структуру:

```
struct KeySpace1 {
    // ключ элемента
    KeyType1 key;
    // указатель на информацию
    Item *info;
    // указатель на следующий элемент
    KeySpace1 *next;
};
```

В пространстве не может быть двух элементов с одинаковыми ключами.

В данном пространстве ключей предусмотрены следующие особые операции:

- поиск элементов, заданных диапазоном ключей; в таблице могут отсутствовать элементы с ключами, задающими диапазон; результатом поиска должна быть новая таблица, содержащая найденные элементы.

Второе пространство ключей организовано как перемешанная таблица, использующая перемешивание сцеплением. Перемешанная таблица представлена массивом указателей на элементы таблицы, имеющие следующую структуру:

```
struct KeySpace2 {
    // ключ элемента
    KeyType2 key;
    // указатель на информацию
    Node2 *node;
    // указатель на следующий элемент
    KeySpace2 *next;
};
```

Указатель на информацию определяет список элементов с одинаковыми значениями ключей. Элемент списка имеет следующую структуру:

```
struct Node2 {
    // номер версии элемента
    RelType2 release;
    // указатель на информацию
    InfoType *info;
    // указатель на следующий элемент
    Node2 *next;
};
```

Максимальный размер массива указателей ограничен величиной `msize2`, значение которой определяется при инициализации таблицы.

В пространстве могут находиться несколько элементов с одинаковыми ключами и разными номерами версий (номер версии элемента формируется как порядковый номер элемента в последовательности элементов с одинаковыми ключами, определяемый при включении элемента в таблицу).

В данном пространстве ключей предусмотрены следующие особые операции:

- поиск в таблице всех версий элемента, заданного ключом, или конкретной (заданной) версии элемента, также заданного своим ключом; результатом поиска должна быть новая таблица, содержащая найденные элементы;
- «чистка таблицы» (или реорганизация таблицы) — удаление из таблицы всех версий элементов, кроме последних.