Задание 3 для первого семестра 2 курса

Отображения

Требуется построить параметризованный класс, который реализует отображение где ключем является строка, а значением — некоторый другой класс. Интерфейс отображения должен поддерживать следующие операции:

— добавить пару (ключ,значение);
— искать значение по указанному ключу;
— удалить ключ и соответствующее значение;
— получить количество хранящихся ключей;
— итератор по множеству ключей и значений.

В качестве примеров и тестов можно рассмотреть отображение строк на целые числа и отображения строк на строки (как в словаре). Тесты должны предусматривать возможность загрузки множества из файла или автоматической генерации.

Варианты заданий различаются способом реализации отображения.

Задание 1.1. Обычное дерево поиска.

Задание 1.2. Сбалансированное дерево поиска.

Задание 1.3. Красно-черное дерево поиска.

Задание 1.4. B-дерево.

Задание 1.5. Хеш-множество по методу списков.

Задание 1.6. Хеш-множество по методу линейных проб.

Упорядоченное множество строк

Такое множество позволяет выбирать упорядоченные подмножества и определять следующий или предыдущий элементы для какого-либо элемента множества (строки). Реализация должна поддерживать следующий интерфейс

— добавить строку в множество;
— удалить строку;
— искать данную строку;
— получить количество элементов в множестве;
— итератор по части множества. Это означает, что после указания некоторой строки, мы можем перебирать последовательно упорядочнные строки множества от указанной в одну или другу сторону.

Для тестов следует предусмотреть загрузку из файла и автоматическую генерацию множества.

Варианты задания различаются основой реализации множества.

Задание 1.7. Обычное дерево для хранения указателей на строки.

Задание 1.8. Сбалансированное дерево для хранения указателей на строки.

Задание 1.9. Красно-черное дерево поиска для хранения указателей на строки.

Задание 1.10. B-дерево для хранения указателей на строки.

Множество точек R^2

Такое множество хранит координаты (x, y) точек плоскости и позволяет выбирать точки, лежащие в некоторой прямоугольной окрестности заданной точки. Предполагается, что все точки находятся внутри изначально заданного прямоугольника. Реализация должна поддерживать следующий интерфейс

— добавить точку в множество;
— удалить точку;
— есть ли точка в множестве;
— получить количество точек в множестве;
— получить список точек, лежащих в данной прямоугольной окрестности заданной точки.

Для тестов следует предусмотреть загрузку из файла и автоматическую генерацию множества.

Варианты задания различаются основой реализации множества.

Задание 1.11. Диапазон изменения по y делится на равные части (полосы), каждой такой части соответствует сбалансированное по значению x дерево точек, у которых y лежит в данной части.

Задание 1.12. Двумерный аналог дерева. Корень — это исходный прямоугольник. Поделив прямоугольник на 4 равные части, получим вершны первого уровня, и т.д. Деление вершин продолжается до тех пор, пока в прямоугольниках содержатся точки. Таким образом, концевые прямоугольники содержат ровно 1 точку.

Задание 1.13. Трехуровневое хеш-множество (дерево). Исходный прямоугольник покрывается прямоугольной сеткой. Ячейки сетки образуют элементы хеш-множества. Каждая точка попадает в некоторую ячейку. По указанным координатам можно вычислить номер нужной ячейки и получить точки, попадающие в эту ячейку. Если в ячейке слишком много точек, то ячейка делится своей прямоугольной сеткой второго уровня. Если в каких-то ячейках второго уровня оказывается много точек, то в них вводится разбиение третьего уровня.

Контейнер параметров

Требуется построить класс, который хранит параметры, необходимые для работы. Параметром может быть число, массив чисел, текстовая строка. Каждый параметр ассоциируется с некоторым именем — ключем поиска. Параметры с их ключами и значениями задаются в текстовом файле в определенном формате

<ключ> = <значение>

Класс должен загружать параметры из файла (из нескольких файлов), разрешать возможные некорректные ситуации, и предоставлять возможность получать значения параметров по их ключам с помощью процедур типа

```
int    GetInt( const char * key );
double GetDouble( const char * key );
const int * GetIntArray( const char * key );
const double * GetDoubleArray( const char * key );
const char * GetString( const char * key );
```

Нужно предусмотреть реакцию на запрос по несуществующему ключу. Детали задания обсуждаются в рабочем порядке.

Варианты задания различаются основой реализации множества.

Задание 1.14. Обычное дерево поиска.

Задание 1.15. Сбалансированное дерево поиска.

Задание 1.16. Красно-черное дерево поиска.

Задание 1.17. B-дерево.

Задание 1.18. Хеш-множество по методу списков.

Задание 1.19. Хеш-множество по методу линейных проб.