

1.

Класс Map (Отображение), реализация классическая ссылочная через бинарное сбалансированное дерево поиска. Построить параметризованный класс, который реализует отображение, где уникальным ключом является строка в стиле Си, а значением – некоторый другой класс.

Необходимые методы: инициализация дерева (конструктор), удаление дерева (деструктор), конструктор копирования и оператор присваивания, добавить пару (ключ, значение), искать значение по указанному ключу, искать следующую (предыдущую) пару для найденного по ключу значения, удалить ключ и соответствующее значение, получить количество хранящихся ключей, итератор по множеству ключей и значений.

В качестве тестов рассмотреть отображение строк на целые числа, отображение строк на строки, взять в качестве другого класса один из реализованных в задачах 2-3 добавив, при необходимости, в этот класс некоторые методы.

2.

Класс Map (Отображение), реализация классическая ссылочная через бинарное красно-черное дерево поиска. Построить параметризованный класс, который реализует отображение, где уникальным ключом является строка в стиле Си, а значением – некоторый другой класс.

Необходимые методы: инициализация дерева (конструктор), удаление дерева (деструктор), конструктор копирования и оператор присваивания, добавить пару (ключ, значение), искать значение по указанному ключу, удалить ключ и соответствующее значение, получить количество хранящихся ключей, искать следующую (предыдущую) пару для найденного по ключу значения, итератор по множеству ключей и значений.

В качестве тестов рассмотреть отображение строк на целые числа, отображение строк на строки, взять в качестве другого класса один из реализованных в задачах 2-3 добавив, при необходимости, в этот класс некоторые методы.

3.

Реализовать на C++ AVL-дерево строк std::string. Необходимые методы: инициализация дерева (конструктор), удаление дерева (деструктор), добавление строки в дерево, удаление строки из дерева, поиск строки в дереве. Также необходимо реализовать итератор по дереву и вывод в консоль элементов дерева нужного «этажа». Реализация дерева – с использованием «умных» указателей.

Для тестов реализовать загрузку строк из файла и генерацию случайного множества строк.

4.

Класс MultiMap (Мультиотображение), реализация через хеш-множество по методу многих списков. Построить параметризованный класс, который реализует мультиотображение, где ключом является строка в стиле Си, а значением – некоторый другой класс. Каждый ключ может быть связан с одним или более значениями.

Необходимые методы: инициализация MultiMap (конструктор), удаление (деструктор), конструктор копирования и оператор присваивания, добавить пару (ключ, значение), искать значение по указанному ключу, удалить ключ и соответствующее значение, получить количество хранящихся ключей, искать следующую (предыдущую) пару для найденного по ключу значения, итератор по множеству ключей и значений.

В качестве тестов рассмотреть отображение строк на целые числа, отображение строк на строки, взять в качестве другого класса один из реализованных в задачах 2-3 добавив, при необходимости, в этот класс некоторые методы.

5.

Класс MultiMap (Мультиотображение), реализация через хеш-множество по методу линейных проб. Построить параметризованный класс, который реализует мультиотображение, где ключом является целое число, а значением – некоторый другой класс. Каждый ключ может быть связан с одним или более значениями.

Необходимые методы: инициализация MultiMap (конструктор), удаление (деструктор), конструктор копирования и оператор присваивания, добавить пару (ключ, значение), искать значение по указанному ключу, удалить ключ и соответствующее значение, получить количество хранящихся ключей; искать следующую (предыдущую) пару для найденного по ключу значения, итератор по множеству ключей и значений.

В качестве тестов рассмотреть отображение целых чисел на целые числа, отображение целых чисел на строки, взять в качестве другого класса один из реализованных в задачах 2-3 добавив, при необходимости, в этот класс некоторые методы.

6.

Класс `Rectangle` (множество точек в  $R^2$ ), Такое множество хранит координаты  $(x, y)$  точек плоскости, и позволяет выбирать точки, лежащие в некоторой окрестности заданной. Предполагается, что все точки находятся внутри изначально заданного прямоугольника. Диапазон изменения по  $y$  делится на равные части, каждой такой части соответствует сбалансированное по  $x$  дерево точек, у которых  $y$  лежит в данной части. Реализация дерева – классическая ссылочная.

Необходимые методы: инициализация `Rectangle` (конструктор), удаление (деструктор), конструктор копирования и оператор присваивания, добавить точку в множество, удалить точку из множества, искать в множестве данную точку, получить количество элементов в множестве, итератор по части множества. Это означает, что после указания некоторой точки мы можем перебирать последовательно точки от указанной в одну или другую сторону, получить список точек, лежащих в данной прямоугольной окрестности заданной точки.

Для тестов реализовать загрузку точек из файла и генерацию случайного множества.

7.

Реализовать В-дерево строк `std::string`. Необходимые методы: инициализация В-дерева (конструктор), удаление В-дерева (деструктор), добавление строки в В-дерево, удаление строки из В-дерева, поиск строки в В-дереве. Также необходимо реализовать итератор по В-дереву. Реализация В-дерева – с использованием «умных» указателей.

Для тестов реализовать загрузку строк из файла и генерацию случайного множества строк.

8.

Реализовать k-d дерево для хранения двумерного облака точек. Корень – это исходный прямоугольник, охватывающий всё облако. Деление прямоугольника пополам (линией, проходящей через середину более длинной стороны) даёт две вершины дерева первого уровня (потомков корня) и т.д. Деление вершин дерева продолжается, пока в прямоугольниках есть точки. Таким образом, концевые прямоугольники содержат по одной точке. Необходимые методы: инициализация k-d дерева (конструктор), удаление k-d дерева (деструктор), добавление точки в k-d дерево, удаление точки из k-d дерева, поиск точки в k-d дереве. Также необходимо реализовать метод, позволяющий получить список точек, лежащих в данной прямоугольной окрестности данной точки. Реализация k-d дерева – с использованием «умных» указателей.

Для тестов реализовать загрузку точек из файла и генерацию случайного множества.