

Лабораторная работа №7
«Продвинутое программирование деревьев»

Задание 1.

Бинарная куча (*binary heap*) представляет собой бинарное дерево, для которого выполняется основное свойство кучи: приоритет каждой вершины больше приоритетов её потомков. В простейшем случае приоритет можно считать равным значению.

Реализуйте бинарную кучу на основе массива и на основе списка. Используйте умные указатели `weak_ptr` на основе своего указателя `shared_ptr` из предыдущих лабораторных.

Задание 2.

Необходимо реализовать 2 вида BST используя шаблоны, итераторы и умные указатели.

Аналоги `std::map` и `std::set`. Продемонстрировать работу при помощи визуальных компонентов.

`Map<KeyType, ValueType>` принимает два шаблонных типа: тип ключа (`KeyType`), тип значения (`ValueType`).

В дереве данные должны лежать в парах `pair<const KeyType, ValueType>`, все операции над деревом выполнять исключительно над `KeyType`

`Set<KeyType> ~ Map<KeyType, char>` (просто фиктивное `Value`, которое не надо использовать).

`Set`, `Map` должны поддерживать два типа итераторов:

1. Для итератора над вершиной дерева `Node` с ключом `key` находит следующий элемент, например, методу `Next` - должны были реализовать в `midterm`;
2. Каждая вершина дерева является еще и вершиной двусвязного списка (такого, что все ключи вершин списка упорядочены по возрастанию). Найти следующий элемент можно просто обратившись к правому соседу в списке.

Необходимо учесть, что метод `Insert`, который вставляет элемент в дерево, предполагает, что это дерево **без итераторов вообще**.

После этого другой виртуальный метод, для разных типов деревьев изменяет некоторые метаданные в них для работы итераторов.

Необходимо реализовать полноценный функционал хеш-таблицы (к примеру аналог `std::unordered_map`), а именно:

- метод `contains` который возвращает `true` если ключ `X` содержится в таблице
- `template` обязательно
- индексация аналогичная `std::map` (при отсутствии элемента по заданному ключу создавать его, используя конструктор по-умолчанию для `ValueType`). нужна версия `ValueType& operator[]....., ValueType operator[](...) const`
- вставка (`insert`), удаление (`erase`), `clear`, `rehash`
- хеш-таблица в качестве шаблонного аргумента обязана принимать функтор хеширования
- для самих цепочек надо использовать `std::forward_list<std::pair<const KeyType, ValueType>>`
- при вставке по необходимости делайте `rehash`

<https://neerc.ifmo.ru/wiki/index.php?title=Хеш-таблица>

https://neerc.ifmo.ru/wiki/index.php?title=Разрешение_коллизий

Следуя принципам ООП избавиться от дублирования кода при написании `Set` и `Map`.

Темы для защиты лабораторной работы: Обходы в глубину и ширину, большие и малые повороты, Большая О Нотация для деревьев различных видов и весь пройденный материал за полный курс дисциплины «Основы алгоритмизации и программирования».