Лабораторная работа №7

«Продвинутое программирование BST. Bitset»

Задание 1 и 2. Необходимо реализовать 2 вида BST.

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 (тот самый) 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. тип итераторов для итератора над вершиной дерева Node с ключом key находит следующий по, например, методу Next - должны были в midterm реализовать;
2. тип итераторов: 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

Задание 3. Реализовать класс BitSet и продемонстрировать его работу при помощи визуальных компонентов.

Кроме геттеров, сеттеров, &operator[], operator[] (index) const, необходимы следующие функции:

all	Проверяет все биты в этом параметре, bitset чтобы определить, все ли они имеют значение true .
any	Функция-член проверяет, равен ли какой-либо бит в последовательности 1.
count	Эта функция-член возвращает количество бит, заданных в последовательности бит.
flip	Инвертирует все биты в bitset или инвертирует один бит в указанной позиции.
none	Проверяет, присвоено ли хотя бы одному биту в объекте bitset значение 1.
reset	Сбрасывает все биты в bitset в значение 0 или сбрасывает бит в указанной позиции в 0.
set	Присваивает всем битам в bitset значение 1 или присваивает биту в указанной позиции значение 1.
size	Возвращает количество бит в объекте bitset.
test	Проверяет, присвоено ли биту в указанной позиции в bitset значение 1.
to_string	Преобразует объект bitset в строковое представление.
to_ullong	Возвращает сумму значений бит в bitset как unsigned long long.

to_ulong	Преобразует объект bitset в unsigned long, чтобы получить последовательность его битов, если используется для инициализации bitset.

Операции ~,&,| должны работать за O(N/16) (или 32 или 64), а другие за O(1)