Задание 4-а. Сбалансированные деревья

Реализовать одну структур данных – «множество» или «мультимножество» (в зависимости от варианта задания):

0. Красно-чёрное дерево без повторов (аналог – set);
1. Красно-чёрное дерево с повторами (при вставке значения с существующем ключом добавляется новый узел дерева – аналог miltiset);
2. AVL-дерево без повторов (при повторной вставке элемент игнорируется – аналог set);
3. AVL-дерево с повторами (аналог multiset).

При любой неоднозначности в формулировке задания брать за образец соответствующие контейнеры библиотеки C++. Вариант задания выбирается следующим образом: количество символов в фамилии, имени и отчестве (своих собственных) по модулю 4 (посчитать количество буковок и взять остаток от деления на 4). Если студент уже выбрал вариант задания на занятии и обсудил это с преподавателем, то реализовывать уже выбранный.

Структуру данных оформить в виде контейнера, с поддержкой итераторов соответствующей категории. Любые аспекты реализации должны быть инкапсулированы (спрятаны), работа только посредством итераторов и соответствующих методов шаблона. Названия методов должны совпадать с таковыми у контейнеров *set* и *multiset* стандартной библиотеки. Должны быть реализованы методы *upper_bound* и *lower_bound*.

Замечание: указанные контейнеры в стандартной библиотеке поддерживают различные типы итераторов, относящиеся к категории двунаправленных итераторов – константные и неконстантные, прямые и обратные. Константные и неконстантные имеют одинаковую семантику и являются, по сути, константными итераторами: «**`iterator`** *of an associative container is of the bidirectional iterator category. For associative containers where the value type is the same as the key type, both* **`iterator`** *and* **`const_iterator`** *are constant iterators*» [C++ Standard, 23.2.4.6]. Соответственно, реализовывать константный итератор либо не нужно, либо сделать это с минимальными затратами как псевдоним неконстантного. Обратные итераторы реализовывать не требуется.

На основе выбранной структуры данных реализовать шаблон множества, поддерживающий следующие операции:

1. Добавление, извлечение элементов, поиск (проверка на вхождение);
2. Вывод содержимого структуры данных на экран «боком» (если явно указана возможность печати);

3. Для каждой операции добавления и удаления реализовать подсчет количества вращений для восстановления балансировки;
4. Сохранение в файл и чтение из файла (текстовый);
5. Сравнить эффективность реализации структуры данных со стандартным контейнером std::set – временные тесты на некоторых последовательностях значений.

Задание 4-b. Очередь с приоритетами на основе бинарной кучи

Реализовать шаблон структуры данных «Очередь с приоритетами», параметризованный по аналогии со стандартным priority_queue базовым контейнером:

```
template <class T, class Container = vector<T>, class Compare = less<typename Container::value_type> > class priority_queue;
```

В данном объявлении [C++ Standard, 23.6.4] базовым контейнером по умолчанию является **vector** – именно он представляет массив в реализации бинарной кучи, рассмотренной на лекции. Однако подходящим контейнером также является и **deque**, на основе которого можно конструировать очередь с приоритетами. Сравнить эффективность реализации для двух базовых структур данных.

Реализовать структуру данных «Очередь с приоритетами фиксированного размера» – по аналогии с **priority_queue**, однако очередь не может содержать более заданного числа элементов.

Задание 4-c. Использование структур данных

Реализовать указанные задания – наиболее эффективным способом, с обоснованием предложенного алгоритма.

Найти в Интернете файл с множеством слов либо взять какую-нибудь довольно объёмную книгу на английском языке. Построить множество слов указанного файла (какое именно множество – подумать, на усмотрение студентов), либо каких-нибудь структур на основе слов (например, **std::pair**). Возможно, решение задачи потребует

Задача 1. Найти 40 наиболее часто встречающихся слов (с указанием количества);

Задача 2. Найти 20 наиболее часто встречающихся собственных имён (для книг, например, Властелин Колец или аналогичных по объёму. Собственным именем считать слова, начинающиеся с прописной буквы. Для эстетов – аналогично, но анализировать книги на каком-либо языке на выбор – немецком, французском, итальянском и т.д.

Задача 3. Найти самую большую группу анаграмм – слов, состоящих из одинаковых символов (по сути, различные перестановки одного и того же множества символов, так, апельсин и спаниель относятся к одной группе анаграмм). Найти пару самых длинных анаграмм. В качестве текста можно использовать либо файл слов какого-либо алфавита, либо достаточно большое литературное произведение.

Задача 4. На основе приведённой базы компьютерных игр для каждого издателя указать суммарный доход по различным жанрам игр.

Задача 5. Вычислить наиболее доходный жанр игровой индустрии по каждому году.

Задача 6. Для каждого издателя выдать список из 5 наиболее доходных разработчиков (за всё время).