Контрольные вопросы. Задание 15.

27 февраля 2023 г.

1 В каких ситуациях используются контейнеры типа множество-отображение?

Контейнеры типа множество-отображение (ассоциативные контейнеры) могут использоваться в ситуациях, когда необходимо:

- Довольно быстрый поиск элемента с заданным значением. Дело в том, что обычно ассоциативные контейнеры реализуются в виде бинарных деревьев (или других структур с возможностью автоматической сортировки по ключу), что обеспечивает логарифмическую сложность. В то же время в силу предварительной сортировки невозможно изменять значения элементов.

- Лишь знать, принадлежит ли конкретный элемент рассматриваемой выборке, т.е. находится ли где-то в контейнере. В этом случае следует использовать неупорядоченные ассоциативные контейнеры, в которых элементы не имеют определённого порядка. При этом поиск элемента с конкретным значением выполняется за амортизированную единицу (при "достаточно хорошей"хеш-функции), т.е. ещё быстрее, чем в ассоциативном массиве (т.е. в массиве, имеющем индекс произвольного типа).

- Чтобы все элементы рассматриваемой коллекции должны быть уникальными, т.е. встречаться единственный раз. В этом случае можно использовать множество или неупорядоченное множество. В свою очередь, соответсвтующие мультимножества используются, когда дубликаты возможны.

- Построить коллекцию из пар "ключ-значение"с уникальными ключами. Такое может потребоваться для реализации ассоциативного массива, для чего могут быть использованы множетсво или неупорядоченнное множество. В свою очередь, соответствующие мультиотображения могут содержать несколько элементов, имеющих одинаковые значения. Мультиотображения могут использоваться как словари.

2 Каким требованиям должна удовлетворять качественная хеш-функция?

Качественная хеш-функция (функция, отображающая значение в хеш-код) должна удовлетворять требованиям:

- детерминированность, т.е. хеш от одинаковых значений должен оставаться одним и тем же;

- довольна высокая скорость вычисления, которая не должна зависеть от размера хеш-таблицы, но может зависеть от размера отображаемого значения;

- равномерность, т.е. значения хеша распределены равномерно;
- односторонность, т.е. сложность обратного вычисления (по известному хешу);
- устойчивость к коллизиям.

Последние два требования особенно важны для криптографических хеш-функций.

3 Из-за чего в хеш-таблицах возникают коллизии и как их можно разрешать?

Коллизии в хеш-таблицах возникают в связи с тем, что хеш-функции от разных значений может давать одинаковый результат (т.к. хеш-функция отображает множество большей мощности во множетсво меньшей). Коллизии можно разрешать, например, методом цепочек или с помощью открытой адресации.

В методе цепочек при обнаружении коллизии по данному ключу в хеш-таблице добавляется новый элемент, т.е. теперь в таблице будет храниться на одну пару "ключ-значение"с данным ключом больше. При этом на каждое следующее значение ссылается предыдущее. Т.о., формируется последовательный список.

Открытая адресация при обнаружении коллизии, двигать элемент с некоторым шагом по хеш-таблице (размер шага может быть различным в зависимости от реализации), пока не найдётся свободная ячейка, куда и записывается значение. Если все места заняты, то происходит рехеширование. Более того, оно может иметь место и в методе цепочек (чтобы не создавались очень длинные цепочки, затягивающие поиск элементов).

4 Почему сложность основных операций хеш-таблиц в худшем случае O(N)?

Сложность основных операций хеш-таблиц в худшем случае O(N), поскольку неизбежны коллизии, требующие разрешения и при этом в худшем случае:

- В методе цепочек может образоваться довольно большой список значений (скорее даже, несколько списков) с одинаковым ключом (каждый список – со своим), а поиск в списках выполняется за O(N) (отсутствует возможность произвольного доступа к элементам). С этим можно бороться посредством рехеширования, однако оно тоже требует времени не меньше O(N) (хоть и вызывается редко).
- При открытой адресации произойдёт рехеширование, требующее времени не меньше, чем O(N).

5 В чём заключается преимущество интерфейсов Boost.Multiind

Boost.Multiindex предоставляет возможность использования различных интерфейсов для одного и того же набора данных. В свою очередь, это может помочь сделать код более быстрым и простым, т.к. появится возможность использовать преимущества различных структур для одной и той же коллекции данных.

Литература.

[1] Конспект семинара. Макаров И.С.

[2] Standard library. Dzhosattis N.

[3] http://david-grs.github.io/why_boost_multi_index_container-part1/