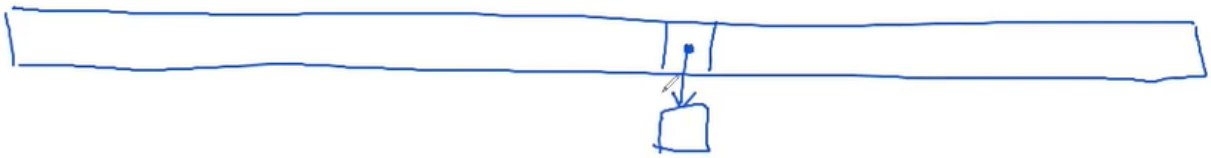
5.14 Контейнеры unordered_map и unordered_set, их внутреннее устройство с алгоритмической точки зрения: что хранится внутри, какие алгоритмы и структуры данных используются для реализации методов. Что алгоритмически происходит при вставке, при удалении, при rehash? Пример использования нестандартного компаратора, нестандартной хэш-функции в unordered_map. Асимптотика обхода unordered_map итератором с объяснением, как он (обход) работает. Правила инвалидации итераторов и ссылок в unordered_map.

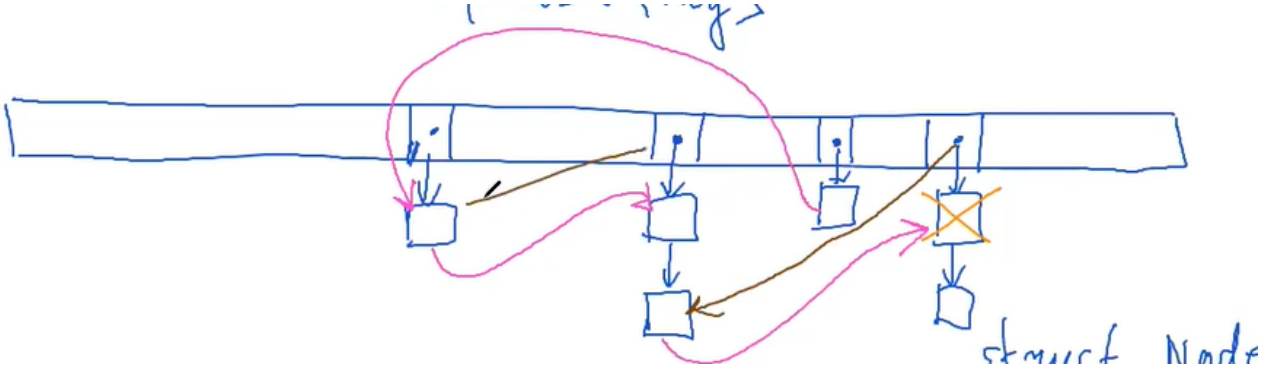
unordered_map https://en.cppreference.com/w/cpp/container/unordered_map

```
template<
    class Key,
    class T,
    class Hash = std::hash<Key>,
    class KeyEqual = std::equal_to<Key>,
    class Allocator = std::allocator< std::pair<const Key, T> >
> class unordered_map;
```

unordered_map - это хеш-таблица, которая позволяет выполнять операции за O(1) в среднем за счет потери упорядоченности. Изначально эта таблица пустая (заполнена nullptr), в конкретный bucket попадают элементы, чей хеш равен номеру bucket



Unordered_map представляет собой некоторое количество односвязных списков, конец каждого из которых указывает на начало следующего, а начало каждого списка - на конец предыдущего



insert

Если нужно положить элемент в ячейку, в ячейке заводится односвязный список, этот односвязный список вставляется в начало таблицы и в него вставляется элемент, после чего добавляем

указатель из конца нового списка в начало следующего. Если там уже был не пустой список при вставке, то нужно пройтись по всему списку и с помощью переданного компаратора equal_to сравнить ключи. Так как в unordered_map не может быть двух элементов с одинаковым ключом, мы либо обновляем элемент в таблице (если ключ нашелся), либо добавляем новый в конец списка, обновляем связи.

erase

Чтобы удалить элемент, мы аналогично insert его ищем, и в случае, если элемент действительно лежит в контейнере, нам нужно удалить его из списка и обновить указатели на конец предыдущего списка/начало следующего списка в случае, если был удален соответственно первый или последний элементы односвязного списка в bucket

rehash

При rehash мы создаем таблицу большего размера, перекладываем все элементы старой таблицы в новую и удаляем старую

Обходя unordered_map по итератору, мы просто итерируемся по одному длинному списку, поэтому асимптотика O(n)

В качестве параметра Hash и KeyEqual можно передать свои нестандартные хеш-функцию и компаратор

```cpp
template<typename T>
struct MyHash
{
    std::size_t operator()(T const& value) const
    {
        size_t h1 = std::hash<T>{}(value);
        size_t h2 = std::hash<T>{}(value);
        return h1 ^ (h2 << 1);
    }
};
```

```cpp
template<typename T>
struct MyEqual
{
    std::size_t operator()(T const& value1, T const& value2) const
    {
        return (value1 > (value2 + value2));
    }
};
```

Правила инвалидации итераторов и ссылок в unordered_map

unordered_map **не инвалидирует ссылки**, а **итераторы** инвалидируются **только если произошел rehash**

unordered_set

std::unordered_set — ассоциативный контейнер, который содержит набор уникальных объектов. Далее можно написать всё то же, что и про unordered_map, ведя разговор только о ключах.
https://en.cppreference.com/w/cpp/container/unordered_set