Map

Maps are associative containers that store elements in a mapped fashion. Each element has a **key** value and a **mapped** value. The keys are **unique**.

1. To use a map, you have to include the 'map' header file

```
#include <map>
```

2. The syntax to declare a map is:

```
std::map<key_type, value_type> name;
```

3. The elements of a map are sorted according to their keys.

Functions

- **begin()** - Returns an iterator to the first element in the map.
- **end()** - Returns an iterator to the theoretical element that follows the last element in the map.
- **size()** - Returns the number of elements in the map.
- **empty()** - Returns whether the map is empty.
- **insert(key, value)** - Insert elements with a particular key in the map container –> *O(log n)*
- **count(g)** - Returns the number of matches to element with key 'g' in the map. –> *O(log n)*
- **erase()** - Used to erase elements from the container –> *O(log n)*
- **find(g)** - Returns an iterator to the element with key-value ‘g’ in the map if found, else returns the iterator to end.
- **upper_bound(g)** - Returns an iterator to the first element that is equivalent to mapped value with key-value ‘g’ or definitely will go after the element with key-value ‘g’ in the map.
- **lower_bound(g)** - Returns an iterator to the first element that is equivalent to the mapped value with key-value ‘g’ or definitely will not go before the element with key-value ‘g’ in the map –> *O(log n)*

The **.first** and **.second** properties are used so get key and value respectively from a map element.