This lecture discusses two closely related data structures: sets and maps. A set is a collection of unique objects where duplicates are not stored. It is useful for checking the presence of an object and eliminating duplicates. Sets in C++ can be created using the `set` keyword followed by the data type, such as integers. There are two versions of sets in the C++ Standard Template Library: regular sets and unordered sets. Regular sets are implemented using red-black trees, allowing operations like insertion, deletion, and search to be performed in \(\mathcal{O}(\log n)\) time. Items in a regular set are ordered from smallest to largest.

Maps are an extension of sets, consisting of unique keys and associated values. In a map, keys are unique, and each key has a corresponding value. Maps are also implemented using red-black trees, providing \(\mathcal{O}(\log n)\) time complexity for operations. Items can be inserted into a map using the index operator, and the presence of a key can be checked using the `count` method. Iterating through a map yields pairs of keys and values, ordered by the keys.

Unordered sets use hash tables and hash functions to map values into buckets, resulting in average time complexity of \(\mathcal{O}(1)\) for insertion, deletion, and lookup operations. Unlike regular sets, unordered sets do not maintain any order for the stored items. They are faster than regular sets when ordering is not required. Unordered sets can be created and manipulated similarly to regular sets, but the order of items when iterating through them is unpredictable.

Unordered maps, like unordered sets, use hash tables internally, providing \(\mathcal{O}(1)\) average time complexity for operations. They consist of key-value pairs and do not maintain any order for the stored items. Items can be inserted and checked for presence using the index operator and `count` method, respectively. Iterating through an unordered map yields key-value pairs in an unpredictable order.

In summary, this lecture covers the implementation and usage of sets, maps, unordered sets, and unordered maps in C++. Regular sets and maps use red-black trees, providing ordered storage and \(\mathcal{O}(\log n)\) time complexity for operations. Unordered sets and maps use hash tables, offering faster \(\mathcal{O}(1)\) average time complexity but without maintaining order.