

Set Solutions

Set

- What are the two main characteristics of `std::set`?
 - Each element is unique
 - The elements are stored in order, using the `<` operator of the key by default
- Which data structure is used to implement it?
 - Tree

Adding and removing elements

- Which member function is called to add elements to an `std::set`?
 - `insert()`
- What happens if the element we want to add is already present in the set?
 - The insert will fail, because the elements must be unique
- How can we find out whether this has happened?
 - By inspecting the return value, which is an `std::pair`
 - The second member is a boolean (true if insert succeeded)

Set example

- Write a program that creates a set containing the elements 6, 7, 4, 5 and 3
- Add an element with key 3 to the set. Does this succeed? If not, why not?
 - This fails, because the set already contains an element with key 3
- Erase the element with key 3 and try again. Does this succeed? If not, why not?
 - This succeeds, because the set does not now contain an element with key 3

Finding Elements

- Write a simple program that populates an `std::set` object
- Use the `find()` and `count()` member functions to check whether an element with a given key is in the set
- Make sure your program works correctly if the element is not present

std::set and Algorithms

- What type of algorithms from the Standard Library can be used with a set?
 - Read-only algorithms which take a predicate
- Why can we not use other algorithms?
 - The elements in a set are const (to maintain the ordering constraint)
 - Algorithms which modify elements cannot be used
 - Algorithms which change the order of elements cannot be used

std::set Pros and Cons

- Give an example of a programming problem where std::set could be useful
 - Creating a collection of data that does not contain duplicates