# Boost::Algorithmn

* [boost::algorithm::all\_of\_equal() in C++ library](https://www.geeksforgeeks.org/boostalgorithmall_of_equal-in-c-library/) : It takes a sequence and a value, and returns true if the all the elements are same in the sequence.
* [boost::algorithm::one\_of() in C++ library](https://www.geeksforgeeks.org/boostalgorithmone_of-in-c-library/) : returns true if the predicate returns true for exactly one given element in the sequence.
* [boost::algorithm::is\_sorted() in C++ library](https://www.geeksforgeeks.org/boostalgorithmis_sorted-in-c-library/) : tests if the given sequence is sorted or not
* [boost::algorithm::all\_of() in C++ library](https://www.geeksforgeeks.org/boostalgorithmall_of-in-c-library/)
* [boost::algorithm::one\_of\_equal() in C++ library](https://www.geeksforgeeks.org/boostalgorithmone_of_equal-in-c-library/): returns true if the exactly one of the elements are same in the sequence to the value passed.
* [boost::algorithm::none\_of\_equal() in C++ library](https://www.geeksforgeeks.org/boostalgorithmnone_of_equal-in-c-library/): returns true if none of the elements are equal to the value in the sequence.
* [C++ Boost String Algorithms Library](https://www.geeksforgeeks.org/c-boost-string-algorithms-library/)

1. **Converting a string to uppercase and lowercase.**

* **to\_upper()** and **to\_lower()** convert the case of characters in a string using a specified locale.
* **to\_upper\_copy()** and **to\_lower\_copy()** returns the copy of the converted string.

1. **To remove characters from a string**

* **erase\_first\_copy()** will remove the first occurrence in the source string.
* **erase\_nth\_copy()** will remove the nth occurrence in the source string.
* **erase\_all\_copy()** will remove all occurrences of a particular character from a

1. **To replace characters from a string**

* **replace\_first\_copy()** will replace the first occurrence in the source string.
* **replace\_nth\_copy()** will replace the nth occurrence in the source string.
* **replace\_all\_copy()** will replace all occurrences of a particular character from a
* [boost::algorithm::clamp() in C++ library](https://www.geeksforgeeks.org/boostalgorithmclamp-in-c-library/): It returns low if value is less than low; It returns high if high is greater than value; In all other cases, it returns value.
* [boost::algorithm::is\_partitioned() in C++ library](https://www.geeksforgeeks.org/boostalgorithmis_partitioned-in-c-library/) : which tests if the given sequence is partioned according to the given predicate or not. Partition here means that all the items in the sequence that satisfy the predicate are at the beginning of the sequence.
* [boost::algorithm::none\_of() in C++ library](https://www.geeksforgeeks.org/boostalgorithmnone_of-in-c-library/): returns true if the predicate returns false when applied to every element in the sequence.
* [boost::algorithm::any\_of() in C++ library](https://www.geeksforgeeks.org/boostalgorithmany_of-in-c-library/) : returns true if the predicate returns true for any given element in the sequence.
* [boost::algorithm::any\_of\_equal() in C++ library](https://www.geeksforgeeks.org/boostalgorithmany_of_equal-in-c-library/): returns true if the any of the elements are same in the sequence to the value passed.