Алгоритмы определены в заголовочном файле **<algorithm.h>** .

Ниже приведены имена некоторых наиболее часто используемых алгоритмов **STL**.

**Немодифицирующие алгоритмы**

1. **adjasent\_find()** – находит пары соседних значений
2. **count() –** подсчитывает количество вхождений значения в последовательность
3. **count\_if()** – подсчитывает количество выполнений предиката в последовательности
4. **equal()** – определяет попарно равенство элементов двух последовательностей
5. **find()** – находит первое вхождение значения в последовательность
6. **find\_end()** – находит последнее вхождение одной последовательности в другую
7. **find\_first\_of()** – находит первое вхождение значения из одной последовательности в другой
8. **find\_if()** – находит первое соответствие предикату в последовательности
9. **for\_earch()** – выполняет операции для каждого элемента последовательности
10. **mismatch()** – находит первый несовпавший элемент в двух последовательностях
11. **search()** – находит первое вхождение последовательности как подпоследовательности
12. **search\_n()** – находит n-е вхождение значения в последовательность

**Модифицирующие алгоритмы**

1. **copy()** – копирует последовательность, начиная с первого элемента
2. **copy\_backward()** – копирует последовательность, начиная с последнего элемента
3. **fill()** – заменяет все элементы заданным значением
4. **fill\_n()** – заменяет первые n элементов заданным значением
5. **generate()** – заменяет все элементы результатом операции
6. **generate \_n()** – заменяет первые n элементов результатом операции
7. **iter\_swap()** – меняет местами два элемента, заданные итераторами
8. **random\_shuffle()** – перемещает элементы согласно случайному равномерному распределению ("тасует" последовательность)
9. **remove()** – удаляет элементы с данным значением
10. **remove\_if()** – удаляет элементы при выполнении предиката
11. **remove\_copy()** – копирует последовательность, удаляя элементы с указанным значением
12. **remove\_copy\_if()** – копирует последовательность, удаляя элементы при выполнении предиката
13. **replace()** – заменяет элементы с указанным значением
14. **replace\_if()** – заменяет элементы при выполнении предиката
15. **replace\_copy()** – копирует последовательность, заменяя элементы с указанным значением
16. **replace\_copy\_if()** – копирует последовательность, заменяя элементы при выполнении предиката
17. **reverse()** – меняет порядок следования элементов на обратный
18. **reverse\_copy()** – копирует последовательность в обратном порядке
19. **rotate()** – циклически перемещает элементы последовательности
20. **rotate\_copy()** – циклически копирует элементы
21. **swap()** – меняет местами два элемента
22. **swap\_range()** – меняет местами элементы двух последовательностей
23. **transform()** – выполняет заданную операцию над каждым элементом последовательности
24. **unique()** – удаляет равные соседние элементы
25. **unique\_copy()** – копирует последовательность, удаляя равные соседние элементы

**Сортировка**

1. **sort()** – сортирует последовательность с хорошей средней эффективностью
2. **partial\_sort()** – сортирует часть последовательности
3. **partial\_sort\_copy()** – сортирует часть последовательности с копированием
4. **stable\_sort()** – сортирует последовательность, сохраняя порядок следования равных элементов

**Поиск**

1. **binary\_search()** – определяет, есть ли данный элемент в отсортированной последовательности
2. **equal\_range()** – находит последовательность элементов с заданным значением
3. **lower\_bound()** – находит первое вхождение значения в отсортированной последовательности
4. **upper\_bound()** – находит первый элемент, больший чем заданное значение

**Слияние**

1. **inplace\_merge()** – сливает две отсортированные последовательности одного диапазона
2. **merge()** – сливает две отсортированные последовательности
3. **partition()** – перемещает вперед элементы, удовлетворяющие условию
4. **stable\_partition()** – перемещает вперед элементы, удовлетворяющие условию, с сохранением их относительного порядка

**Работа с множествами**

1. **includes()** – проверка на вхождение
2. **set\_union()** – объединение множеств
3. **set\_intersection()** – пересечение множеств
4. **set\_difference()** – разность множеств

**Минимумы и максимумы**

1. **min()** – меньшее из двух
2. **max()** – большее из двух
3. **min\_element()** – наименьшее значение в последовательности
4. **max\_element()** – наибольшее значение в последовательности

**Перестановки**

1. **nth\_element()** – помещает n-й элемент на заданное место
2. **next\_permutation()** – следующая перестановка в лексикографическом порядке
3. **prev\_permutation()** – предыдущая перестановка в лексикографическом порядке