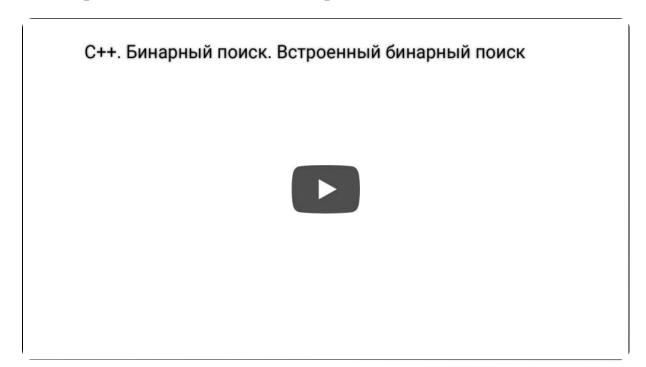
Встроенный бинарный поиск



Встроенный бинарный поиск

В языке C++ есть встроенные функции, которые реализуют алгоритм бинарного поиска в массиве. Чтобы их использовать, необходимо подключить библиотеку, в которой они лежат: #include <algorithm>

Есть две функции, которые реализуют поиск нижней и верхней границы для числа x — lower_bound и upper_bound, а также функция binary_search, которая проверяет наличие числа x. Эти функции можно применять на отсортированном векторе или отсортированном массиве.

Приведём примеры использования этих функций для вектора и массива.

Вектор ∨:

```
binary_search(v.begin(), v.end(), x)
lower_bound(v.begin(), v.end(), x) - v.begin()
Maccuв v размера n:
binary search(v, v + n, x)
```

```
lower bound (v, v + n, x) - v
```

Эти функции для массива и для вектора, а также для любого контейнера с Random-access Iterators (то есть для контейнеров с произвольным доступом по индексу) работают за $O(\log n)$. Иначе эти функции будут работать за O(n). Например, эти функции можно применить к контейнеру set или map, но работать они будут за линейное время.

При этом в контейнерах set и map есть встроенные методы $lower_bound$ и $upper_bound$. Эти методы позволяют искать нижнюю или верхнюю границу для заданного элемента за $O(\log n)$. Отличие от функций, которые мы разбирали выше, заключается в том, что эти методы можно применять только ко всему контейнеру целиком, тогда как функции можно применять на какомто заданном подотрезке массива или вектора.

Также стоит отметить, что методы $lower_bound$ и $upper_bound$ poдственны методу count. Все эти методы для контейнеров set и map основаны на структуре этих контейнеров, которая представляет собой двоичное дерево поиска, что позволяет им работать c асимптотикой $O(\log n)$.

Пример использования метода lower_bound для контейнера set:

```
set<int> se;
se.lower_bound(x);
```