STL → binary Search

↳ lowerbound

↳ upperbound

always sorted array

1	1	2	2	2	4	5
0	1	2	3	4	5	6

what to search

lower_bound (v.begin() , v.end() , 2)

↳ Index → 2

what to search

upper_bound (v.begin() , v.end() , 2)

↳ Index → 5

↳ last occurrence ka next pointer

104	108	112	116	120	124	128
1	1	2	2	2	3	4
0	1	2	3	4	5	6

↑ it

auto it = lower_bound(v.begin(), vend(), 2)

↗

we dont know what they return Then we use auto

cout << *it ; → to get address

for print index

Cout << " index : " << $\underbrace{\text{it - v.begin()}}_{\text{return index}}$;

it − v.begin()

⇓

112 − 104 = 8 bytes

1 int ⇒ 4 byte   8/4 = 2 index