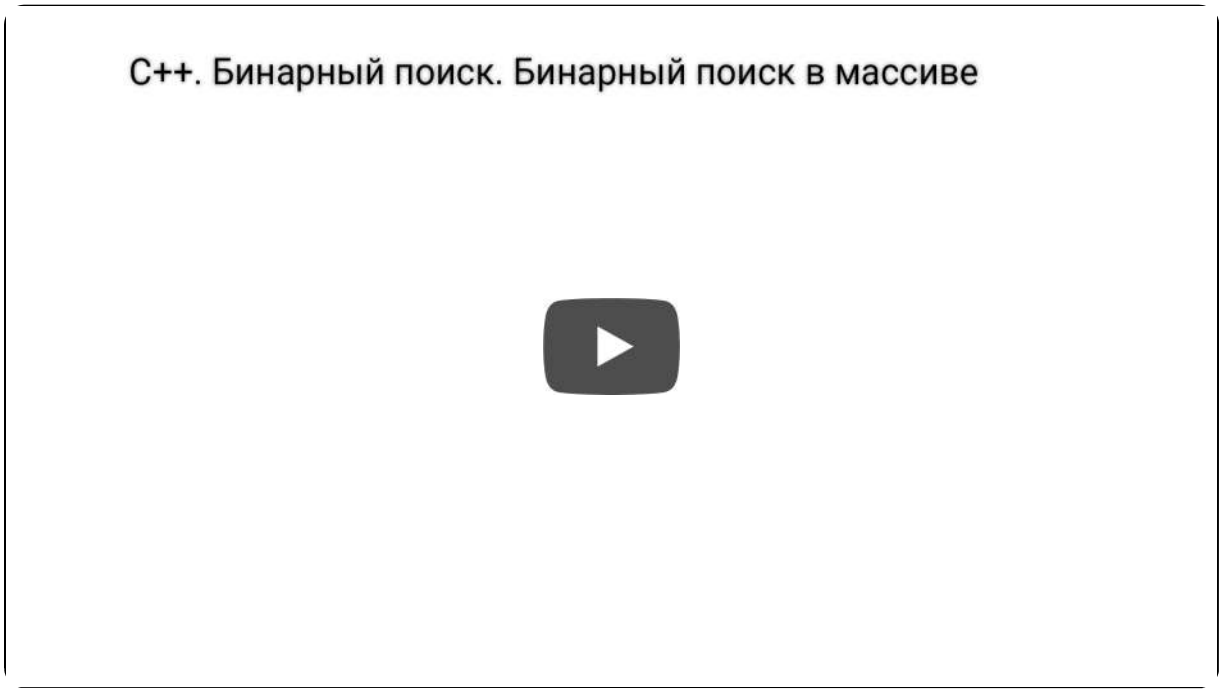
Бинарный поиск в массиве



Бинарный поиск в массиве

Рассмотрим задачу о поиске заданного числа в отсортированном массиве. Пусть заданы число x и отсортированный массив $a_0 \leq a_1 \leq \ldots \leq a_{n-1}$. Требуется найти, на какой позиции в массиве стоит число x, либо определить, что такого числа в массиве нет. Сначала добавим в наш массив два фиктивных элемента – элемент $-\infty$ с индексом -1 и элемент $+\infty$ с индексом n. Получим массив: $-\infty < a_0 \leq a_1 \leq \ldots \leq a_{n-1} < +\infty$.

Тогда мы можем выбрать следующие границы бинпоиска: $L = -1, R = n$, потому что на позиции -1 стоит число заведомо меньшее, чем число x, а на позиции n стоит число заведомо большее, чем число x.

Далее мы можем выбрать один из двух инвариантов для решения нашей задачи.

Инвариант 1.

- $a[L] < x$
- $a[R] \geq x$

После того как алгоритм бинарного поиска с таким инвариантом закончит

свою работу и индексы L и R окажутся соседними числами, то число x надо искать на позиции с индексом R. Если $R = n$, то это будет означать, что числа x нет в массиве. Если $R < n$ и $a[R] \neq x$, то числа x нет в массиве. Если же $R < n$ и $a[R] = x$, то число x найдено на позиции с индексом R. Причём, если в массиве есть несколько элементов равных числу x, то индекс R задаёт позицию первого вхождения числа x в массив. Таким образом, при данном инварианте индекс R будет задавать нижнюю границу вхождений числа x, и поэтому такой индекс R принято называть *LowerBound*.

Инвариант 2.

- $a[L] \leq x$
- $a[R] > x$

После того как алгоритм бинарного поиска с таким инвариантом закончит свою работу и индексы L и R окажутся соседними числами, то число x надо искать на позиции с индексом L. Если $L = -1$, то это будет означать, что числа x нет в массиве. Если $L \geq 0$ и $a[L] \neq x$, то числа x нет в массиве. Если же $L \geq 0$ и $a[L] = x$, то число x найдено на позиции с индексом L. Причём, если в массиве есть несколько элементов равных числу x, то индекс L задаёт позицию последнего вхождения числа x в массив. Таким образом, при данном инварианте индекс R будет задавать позицию, которая следует после последнего вхождения числа x, то есть верхнюю границу вхождений числа x, и поэтому такой индекс R принято называть *UpperBound*.

Для решения задач достаточно одного из двух предложенных инвариантов. Но бывает удобно использовать эти инварианты в паре. Например, если нам необходимо найти количество чисел в нашем массиве, для которых выполнено условие $l \leq a_i \leq r$, то их количество можно найти по формуле *UpperBound*(r)−*LowerBound*(l). Удобство этой формулы в том, что она работает верно во всех случаях.

Приведём код на языке C++, который реализует бинарный поиск числа x в отсортированном массиве a с использованием инварианта 1.

```
int L = -1;
int R = n;
while (R - L > 1) {
    int M = (R + L) / 2;
```

```
    (a[M] < x ? L : R) = M;
}
if (R < n && a[R] == x)
    cout << R << endl;
else
    cout << -1 << endl;
```