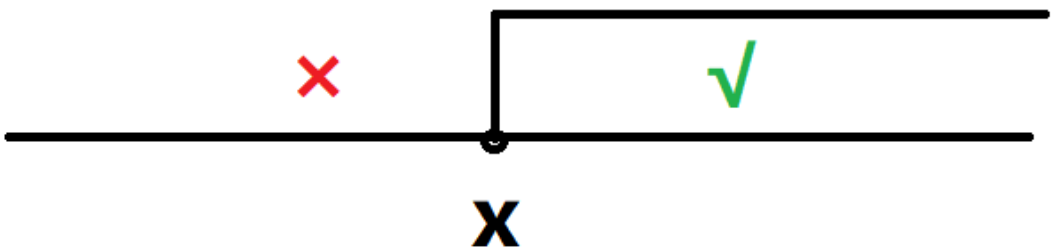
二分模板一共有两个，分别适用于不同情况。
算法思路：假设目标值在闭区间 `[l, r]` 中， 每次将区间长度缩小一半，当 `l = r` 时，我们就找到了目标值。

版本1

在单调递增序列 `a` 中查找 `>= x` 的数中最小的一个：

当我们将区间 `[l, r]` 划分成 `[l, mid]` 和 `[mid + 1, r]` 时，其更新操作是 `r = mid` 或者 `l = mid + 1`;，计算 `mid` 时不需要加1。



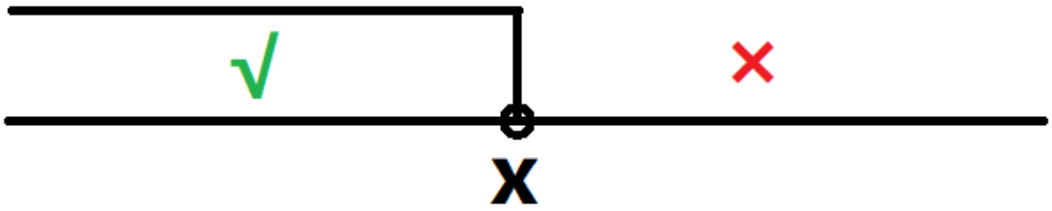
C++ 代码模板：

```
int bsearch_1(int l, int r)
{
    while (l < r)
    {
        int mid = l + r >> 1;
        if (a[mid] >= x) r = mid;
        else l = mid + 1;
    }
    return a[l];
}
```

版本2

在单调递增序列 `a` 中查找 `<=x` 的数中最大的一个:

当我们将区间 `[l, r]` 划分成 `[l, mid - 1]` 和 `[mid, r]` 时，其更新操作是 `r = mid - 1` 或者 `l = mid;`，此时为了防止死循环，计算 `mid` 时需要加1。



C++ 代码模板：

```
int bsearch_2(int l, int r)
{
    while (l < r)
    {
        int mid = l + r + 1 >> 1;
        if (a[mid] <= x) l = mid;
        else r = mid - 1;
    }
    return a[l];
}
```