# 二分查找

## 前置知识

### 函数

是非常重要的函数。2 的几次方是 ，就是几。例如： 。当然，实际使用时只需稍微估计一下，等于号实际上是约等于。

## 实现

### 思路

* 区间有左指针l和右指针r。只要l<r，就不断地循环。
  + 在所有数字代表的区间，尝试中间下标（mid），也就是左指针和右指针的平均值。
    - 如果发现a[mid]的值就是要找的，不急着返回答案，而是记录一下找到的答案，然后继续二分左区间。（这样可以查找第一个出现的位置）
    - 如果数字太小，则继续处理右区间。
    - 如果数字太大，则继续处理左区间。

### 代码

#include <bits/stdc++.h>  
int a[1000010], ans = -1;  
using namespace std;  
int main() {  
 int n, x;  
 cin >> n;  
 for(int i = 1; i <= n; i++)  
 cin >> a[i];  
 cin >> x;  
 int l = 1, r = n, ans = -1;  
 while (l <= r) {  
 int mid = (l + r) / 2;  
 if (a[mid] == x) { // 如果中间的数字等于要找的  
 ans = mid; // 记录答案位置  
 r = mid - 1; // 局限在左区间  
 } else if (a[mid] > x) // 如果中间数字大于要找的  
 r = mid - 1; // 局限在左区间  
 else // 如果中间数字小于于要找的  
 l = mid + 1; // 局限在右区间  
 }  
 cout << ans; // 输出答案  
}

## 例题：A - B 数对

重复元素的二分查找：查找第一个 or 最后一个元素的位置。

#include <bits/stdc++.h>  
int a[200010], n, c;  
long long ans = 0;  
using namespace std;  
  
int findx(int k) { // 找到第一次出现的位置  
 int l = 1, r = n, ans = -1;  
 while (l <= r) {  
 int mid = (l + r) / 2;  
 if (a[mid] == k) { // 如果中间的数字等于要找的  
 ans = mid; // 记录答案位置  
 r = mid - 1; // 局限在左区间  
 } else if (a[mid] > k) // 如果中间数字大于要找的  
 r = mid - 1; // 局限在左区间  
 else // 如果中间数字小于于要找的  
 l = mid + 1; // 局限在右区间  
 }  
 return ans;  
}  
int findy(int k) { // 找到最后一次出现的位置  
 int l = 1, r = n, ans = -1;  
 while (l <= r) {  
 int mid = (l + r) / 2;  
 if (a[mid] == k) { // 如果中间的数字等于要找的  
 ans = mid; // 记录答案位置  
 l = mid + 1; // 局限在右区间  
 } else if (a[mid] > k) // 如果中间数字大于要找的  
 r = mid - 1; // 局限在左区间  
 else // 如果中间数字小于于要找的  
 l = mid + 1; // 局限在右区间  
 }  
 return ans;  
}  
int main() {  
 cin >> n >> c;  
 for(int i = 1; i <= n; i++)  
 cin >> a[i];  
 sort(a + 1, a + n + 1);  
 for(int i = 1; i <= n; i++) {  
 int x = findx(a[i] + c);  
 int y = findy(a[i] + c);  
 if(x == -1) continue;  
 ans += y - x + 1;   
 }  
 cout << ans; // 输出答案  
}