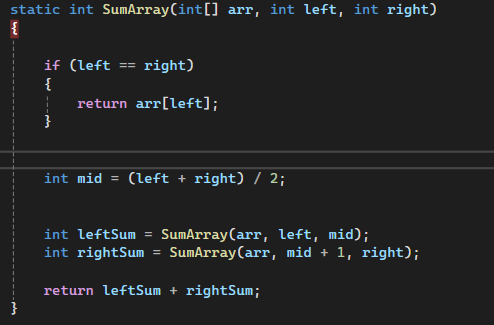
**Tính tổng bằng phương pháp đệ quy**

****

Hàm “SumArray” có:

* Nếu left == right -> mảng chỉ có 1 phần tử, trả về giá trị của phần tử đó.
* mid = (left + right) / 2 để chia mảng thành 2 nửa.
* Gọi đệ quy SumArray(arr, left, mid) để tính tổng nửa trái của mảng.
* Gọi đệ quy SumArray(arr, mid + 1, right) để tính tổng nửa phải của mảng.
* Trả về tổng của 2 nửa.

Quá trình đệ quy của hàm “SumArray”:

- Đệ quy cấp 1:

+ left = 0, right = 9, mid = 4.

+ Gọi SumArray(arr, 0, 4) và SumArray(arr, 5, 9).

- Đệ quy cấp 2:

+ Đối với SumArray(arr, 0, 4):

* left = 0, right = 4, mid = 2.
* Gọi SumArray(arr, 0, 2) và SumArray(arr, 3, 4).

+ Đối với SumArray(arr, 5, 9):

* left = 5, right = 9, mid = 7.
* Gọi SumArray(arr, 5, 7) và SumArray(arr, 8, 9).

- Đệ quy cấp 3:

+ Đối với SumArray(arr, 0, 2):

* left = 0, right = 2, mid = 1.
* Gọi SumArray(arr, 0, 1) và SumArray(arr, 2, 2).

+ Đối với SumArray(arr, 3, 4):

* left = 3, right = 4, mid = 3.
* Gọi SumArray(arr, 3, 3) và SumArray(arr, 4, 4).

+ Đối với SumArray(arr, 5, 7):

* left = 5, right = 7, mid = 6.
* Gọi SumArray(arr, 5, 6) và SumArray(arr, 7, 7).

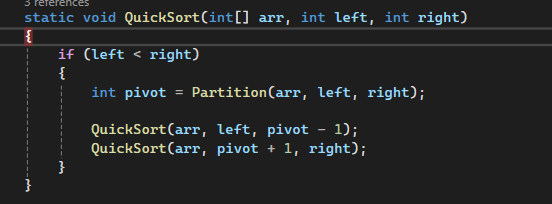
+ Đối với SumArray(arr, 8, 9):

* left = 8, right = 9, mid = 8.
* Gọi SumArray(arr, 8, 8) và SumArray(arr, 9, 9).

- Đệ quy cấp 4:

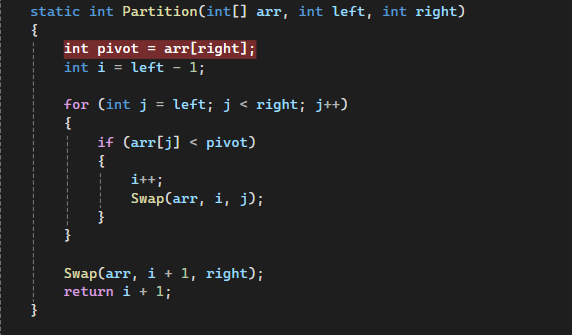
Nếu left == right sẽ trả về giá trị của arr[left].

**QuickSort bằng phương pháp đệ quy**



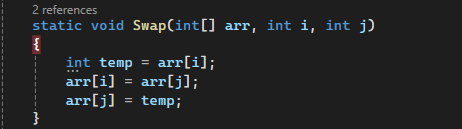
Trong hàm QuickSort có:

* Kiểm tra điều kiện left < right để đảm bảo có ít nhất 2 phần tử để sắp xếp.
* Gọi hàm Partition(arr, left, right) để lấy chỉ số của phần tử pivot.
* Gọi đệ quy QuickSort(arr, left, pivot - 1) để sắp xếp phần tử bên trái pivot.
* Gọi đệ quy QuickSort(arr, pivot + 1, right) để sắp xếp phần tử bên phải pivot.



Trong hàm Partition có:

* Chọn phần tử cuối cùng arr[right] làm pivot.
* Khởi tạo biến i để theo dõi vị trí phần tử cuối cùng nhỏ hơn pivot.
* Duyệt qua mảng, nếu arr[j] < pivot, tăng i và hoán đổi arr[i] và arr[j].
* Cuối cùng, hoán đổi arr[i + 1] và arr[right] để đưa pivot về đúng vị trí.
* Trả về chỉ số của pivot.



Hàm Swap để hoán đổi giá trị của arr[i] và arr[j]

**Bất đầu quá trình đệ quy:**

* QuickSort(numbers, 0, 5) được gọi đầu tiên (với left = 0 và right = 5).
* Gọi hàm Partition(numbers, 0, 5).

- Quá trình Partition

+ pivot = arr[5] = 3.

+ i = 0 - 1 = -1.

+ Vòng lặp for bắt đầu với j = 0 đến j = 4:

* j = 0: arr[0] = 5, không nhỏ hơn pivot (3), i không thay đổi.
* j = 1: arr[1] = 2, nhỏ hơn pivot, tăng i lên 0, gọi Swap(arr, 0, 1) (mảng thành { 2, 5, 8, 1, 9, 3 }).
* j = 2: arr[2] = 8, không nhỏ hơn pivot (3), i không thay đổi.
* j = 3: arr[3] = 1, nhỏ hơn pivot, tăng i lên 1, gọi Swap(arr, 1, 3) (mảng thành { 2, 1, 8, 5, 9, 3 }).
* j = 4: arr[4] = 9, không nhỏ hơn pivot (3), i không thay đổi.

+ Kết thúc vòng lặp, gọi Swap(arr, 2, 5) (mảng thành { 2, 1, 3, 5, 9, 8 }).

+ Trả về pivot index là 2.

- Tiếp tục QuickSort:

Quay lại hàm Quicksort:

* QuickSort(numbers, 0, 1) (với left = 0 và right = 1).
* Gọi hàm Partition(numbers, 0, 1).

- Tiếp tục quá trình Partititon:

* pivot = arr[1] = 1.
* i = 0 - 1 = -1.
* Vòng lặp for bắt đầu với j = 0 đến j = 0:
* j = 0: arr[0] = 2, không nhỏ hơn pivot (1), i không thay đổi.
* Kết thúc vòng lặp, gọi Swap(arr, 0, 1) (mảng thành { 1, 2, 3, 5, 9, 8 }).
* Trả về pivot index là 0.

- Tiếp tục QuickSort:

+ Quay lại hàm QuickSort:

* QuickSort(numbers, 0, -1) (với left = 0 và right = -1) không làm gì vì left >= right.
* QuickSort(numbers, 1, 1) (với left = 1 và right = 1) không làm gì vì left >= right.

+ Tiếp tục với phần phải của mảng:

* QuickSort(numbers, 3, 5) (với left = 3 và right = 5).
* Gọi hàm Partition(numbers, 3, 5).

- Tiếp tục Partititon:

+ pivot = arr[5] = 8.

+ i = 3 - 1 = 2.

+ Vòng lặp for bắt đầu với j = 3 đến j = 4:

* j = 3: arr[3] = 5, nhỏ hơn pivot, tăng i lên 3, gọi Swap(arr, 3, 3) (không đổi).
* j = 4: arr[4] = 9, không nhỏ hơn pivot (8), i không thay đổi.

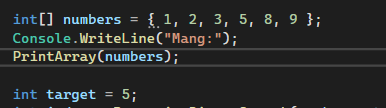
+ Kết thúc vòng lặp, gọi Swap(arr, 4, 5) (mảng thành { 1, 2, 3, 5, 8, 9 }).

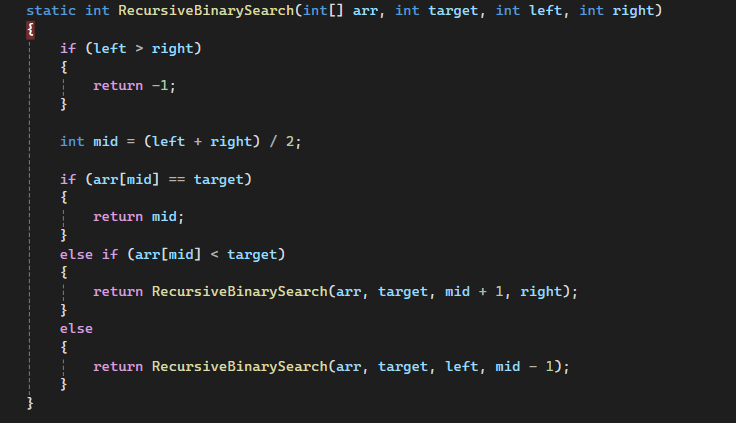
+ Trả về pivot index là 4.

- Kết thúc QuickSort:

* QuickSort(numbers, 3, 3) (với left = 3 và right = 3) không làm gì vì left >= right.
* QuickSort(numbers, 5, 5) (với left = 5 và right = 5) không làm gì vì left >= right.

**Tìm kiếm nhị phân bằng phương pháp đệ quy**

****

****

* Bước 1: Kiểm tra xem left có lớn hơn right hay không. Nếu có, trả về -1 vì không tìm thấy giá trị target.
* Bước 2: Tính chỉ số mid bằng cách lấy trung bình của left và right.
* Bước 3: So sánh giá trị tại mid với target:
* Nếu bằng nhau, trả về chỉ số mid.
* Nếu giá trị tại mid nhỏ hơn target, gọi đệ quy với left = mid + 1 và right = right.
* Nếu giá trị tại mid lớn hơn target, gọi đệ quy với left = left và right = mid - 1.

**Quá trình đệ quy:**

- Lần gọi đầu tiên:

* arr = {1, 2, 3, 5, 8, 9}
* target = 5
* left = 0
* right = 5
* mid = (0 + 5) / 2 = 2
* arr[mid] = arr[2] = 3
* arr[mid] < target (3 < 5), gọi lại hàm với left = mid + 1 = 3 và right = 5.

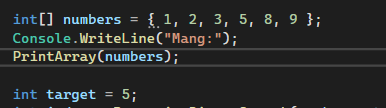
- Lần gọi thứ hai:

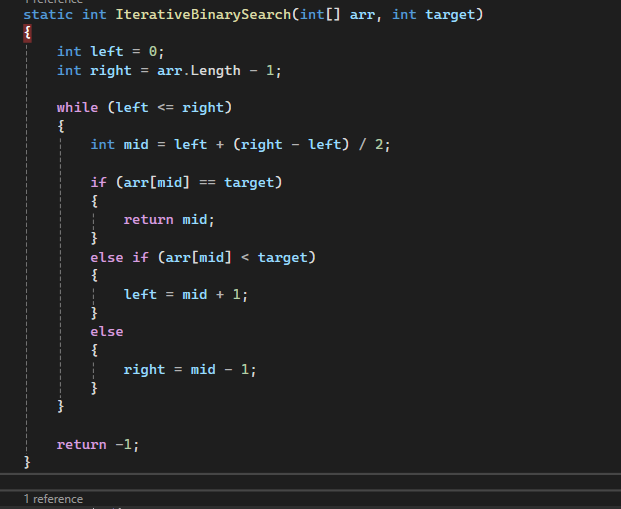
* arr = {1, 2, 3, 5, 8, 9}
* target = 5
* left = 3
* right = 5
* mid = (3 + 5) / 2 = 4
* arr[mid] = arr[4] = 8
* arr[mid] > target (8 > 5), gọi lại hàm với left = 3 và right = mid - 1 = 3.

- Lần gọi thứ ba:

* arr = {1, 2, 3, 5, 8, 9}
* target = 5
* left = 3
* right = 3
* mid = (3 + 3) / 2 = 3
* arr[mid] = arr[3] = 5
* arr[mid] == target (5 == 5), trả về mid = 3.

**Tìm kiếm nhị phân bằng phương pháp không đệ quy**

****



* Bước 1: Khởi tạo biến left bằng 0 và biến right bằng arr.Length - 1.
* Bước 2: Kiểm tra xem left có nhỏ hơn hoặc bằng right không. Nếu không, thoát khỏi vòng lặp và trả về -1 vì không tìm thấy giá trị target.
* Bước 3: Tính chỉ số mid bằng cách left + (right - left) / 2. Điều này giúp tránh tràn số khi left và right quá lớn.
* Bước 4: So sánh giá trị tại mid với target:
* Nếu bằng nhau, trả về chỉ số mid.
* Nếu giá trị tại mid nhỏ hơn target, cập nhật left = mid + 1.
* Nếu giá trị tại mid lớn hơn target, cập nhật right = mid - 1.
* Bước 5: Lặp lại từ bước 2.

**Quá trình thực hiện:**

- Lần lặp đầu tiên:

* mid = left + (right - left) / 2 = 0 + (5 - 0) / 2 = 2
* arr[mid] = arr[2] = 3
* arr[mid] < target (3 < 5), cập nhật left = mid + 1 = 3

- Lần lặp thứ hai:

* mid = left + (right - left) / 2 = 3 + (5 - 3) / 2 = 4
* arr[mid] = arr[4] = 8
* arr[mid] > target (8 > 5), cập nhật right = mid - 1 = 3

- Lần lặp thứ ba:

* mid = left + (right - left) / 2 = 3 + (3 - 3) / 2 = 3
* arr[mid] = arr[3] = 5
* arr[mid] == target (5 == 5), trả về mid = 3