public class BubbleSortRecursivo {

// Método para realizar a ordenação Bubble Sort de forma recursiva

public static void bubbleSortRecursivo(int[] arr, int n) {

// Caso base: se o tamanho do array é 1 ou menor, não há nada a ordenar

if (n == 1) {

return;

}

// Passa pelo array e realiza as trocas necessárias

for (int i = 0; i < n - 1; i++) {

if (arr[i] > arr[i + 1]) {

// Troca arr[i] e arr[i + 1]

int temp = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = temp;

}

}

// Chama recursivamente o método para o restante do array

bubbleSortRecursivo(arr, n - 1);

}

public static void main(String[] args) {

int[] array = {64, 34, 25, 12, 22, 11, 90};

System.out.println("Array original:");

for (int i = 0; i < array.length; i++) {

System.out.print(array[i] + " "); // Usando índice para acessar os elementos

}

System.out.println();

bubbleSortRecursivo(array, array.length);

System.out.println("Array ordenado:");

for (int i = 0; i < array.length; i++) {

System.out.print(array[i] + " "); // Usando índice para acessar os elementos

}

System.out.println();

}

}

**Explicação função recursiva**

1. **for (int i = 0; i < n - 1; i++) {**:
   * Este é um loop for que itera sobre o array.
   * int i = 0: Inicializa a variável i em 0, que representa o índice do array.
   * i < n - 1: O loop continua enquanto i é menor que n - 1. O motivo de usar n - 1 é que estamos comparando o elemento arr[i] com o próximo elemento arr[i + 1]. Se i for igual a n - 1, arr[i + 1] estaria fora dos limites do array, resultando em um erro.
   * i++: Incrementa i em 1 após cada iteração, passando para o próximo elemento.
2. **if (arr[i] > arr[i + 1]) {**:
   * Esta linha verifica se o elemento atual (arr[i]) é maior do que o próximo elemento (arr[i + 1]).
   * Se essa condição for verdadeira, significa que os dois elementos estão fora de ordem e precisam ser trocados para ordenar o array.
3. **int temp = arr[i];**:
   * Se a condição do if for verdadeira, armazenamos o valor do elemento atual (arr[i]) na variável temporária temp. Isso é necessário para realizar a troca sem perder o valor do elemento.
4. **arr[i] = arr[i + 1];**:
   * Aqui, atribuímos o valor do próximo elemento (arr[i + 1]) ao elemento atual (arr[i]). Isso efetivamente "movimenta" o menor elemento para a posição anterior.
5. **arr[i + 1] = temp;**:
   * Finalmente, atribuímos o valor armazenado em temp (que é o valor original de arr[i]) ao próximo elemento (arr[i + 1]). Isso completa a troca entre os dois elementos.

**Resumo**

O loop percorre o array, comparando cada elemento com o próximo e realizando trocas quando necessário. Isso é repetido até que o array esteja ordenado, com o maior elemento "subindo" para a posição correta a cada iteração. O processo é repetido para cada elemento até que não haja mais trocas necessárias.

**Explicação da função de encerramento**

1. **Conceito de Caso Base**:
   * Em algoritmos recursivos, o **caso base** é uma condição que encerra a recursão. Ele evita que a função se chame indefinidamente, resultando em um loop infinito ou estouro de pilha (stack overflow).
   * No contexto do Bubble Sort, o caso base verifica se o array tem 1 ou 0 elementos, já que um array com 1 ou menos elementos está, por definição, ordenado.
2. **if (n == 1)**:
   * Aqui, verificamos se o tamanho do array (n) é igual a 1.
   * Se n for igual a 1, significa que o array contém apenas um elemento. Como não há outros elementos para comparar ou trocar, ele já está ordenado.
3. **return;**:
   * A instrução return; encerra a execução da função no caso base.
   * Assim, quando a função atinge essa condição, ela sai, e a recursão para naquele ponto.

**Resumo do Caso Base**

* O caso base é fundamental para garantir que o algoritmo funcione corretamente. Sem ele, a função continuaria a chamar a si mesma indefinidamente, causando um erro.
* Ao retornar quando n é 1, garantimos que a função só continua sua lógica de ordenação se houver elementos a serem ordenados (ou seja, quando n é maior que 1).

**Exemplo Prático**

Por exemplo, se o array for [5] (ou seja, um único elemento), ao chamar bubbleSortRecursivo(arr, 1), o código verá que n é 1, executará a verificação

if (n == 1), e a função retornará imediatamente, sem fazer mais nada, pois não há necessidade de ordenar.