# Zero-Based Indexing and Summation in Insertion Sort

## Key Points

- **Zero-Based Indexing**:
  - In languages like C or Rust, arrays start at index 0. The first element is `arr[0]`, the second element is `arr[1]`, and so on.
  - When implementing the insertion sort algorithm, the loops typically start from index 1 (the second element) when considering comparisons or shifts, because the first element at index 0 is already considered "sorted."
- **Summation in Zero-Based Context**:
  - If you are analyzing the loop starting from the second element (`arr[1]`), you would still sum over the indices, but the summation index would correspond to the position in the array starting from 1 rather than 2.
- **Impact on the Formula**:
  - The theoretical formula $\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$ assumes 1-based indexing where $j$ starts from 2.
  - In zero-based indexing, if you start the summation from $j = 1$ (which corresponds to the second element in a zero-based array), the formula would be adjusted accordingly to reflect that shift:

$$\sum_{j=1}^{n-1} j = \frac{(n-1)n}{2}$$

  - This reflects the sum of the first $n - 1$ natural numbers, which corresponds to the valid indices of an array with $n$ elements.

## Example in C or Rust

For an array of size $n = 5$:

- **Zero-Based Indexing**:
  - Indices: $0, 1, 2, 3, 4$
  - If you start summing from $j = 1$ (the second element): $1 + 2 + 3 + 4$
  - The sum is still:

$$\frac{(n-1)n}{2} = \frac{4 \times 5}{2} = 10$$

- **Theoretical Summation**:

  - This would correspond to the sum of the first 4 natural numbers in zero-based indexing, which aligns with summing from index 1 to $n-1$.

## Summary

- **Zero-Based Indexing**: The indexing affects the starting point of your loops and summations, but the core idea behind the summation formulas remains the same.

- **Adjusted Formula**: For zero-based arrays, if you're summing starting from the second element, the summation formula can be adjusted to reflect the zero-based index (starting from $j = 1$ instead of $j = 2$).