The **reference of the array (vector)** is passed in the function for **efficiency** and **avoiding unnecessary copies**. Let's break this down step by step.

**1. Why Pass by Reference?**

In C++, function arguments are passed in two ways:

* **Pass by Value** (default)
* **Pass by Reference** (& operator)

If we **pass by value**, a **copy** of the vector will be created inside the function, leading to:

* **More memory usage**
* **More execution time** (since copying large vectors is expensive)
* **Unnecessary duplication of data**

To avoid these issues, we pass the **vector by reference** (vector<int> &arr), which means:

* The function directly works with the **original vector** instead of a copy.
* No extra memory is used.
* It improves performance, especially for **large vectors**.

**2. Effect of Pass by Value vs Pass by Reference**

**(a) Pass by Value (Inefficient)**

int largestElement(vector<int> arr, int n) { // Pass by value (Bad practice)

int largest = arr[0];

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

if(arr[i] > largest) {

largest = arr[i];

}

}

return largest;

}

* A **new copy** of arr is created.
* If arr has **1 million elements**, it will be copied **entirely**, consuming extra memory and slowing down execution.

**(b) Pass by Reference (Efficient)**

int largestElement(vector<int> &arr, int n) { // Pass by reference (Good practice)

int largest = arr[0];

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

if(arr[i] > largest) {

largest = arr[i];

}

}

return largest;

}

* arr is **not copied**, instead, the function works with the **original vector**.
* Saves memory and improves performance.

**3. Should We Use const?**

If we do **not** want the function to modify the vector, we can use const:

int largestElement(const vector<int> &arr, int n) { // `const` prevents modification

int largest = arr[0];

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

if(arr[i] > largest) {

largest = arr[i];

}

}

return largest;

}

* This ensures that arr **cannot be changed** inside the function.
* Recommended for **read-only operations** like finding the largest element.

**Conclusion**

| **Method** | **Memory Usage** | **Performance** | **Can Modify Original Data?** |
| --- | --- | --- | --- |
| Pass by Value (vector<int> arr) | High (copies entire vector) | Slow | ❌ No |
| Pass by Reference (vector<int> &arr) | Low (no copying) | Fast | ✅ Yes |
| Pass by const Reference (const vector<int> &arr) | Low (no copying) | Fast | ❌ No |

Since we **only need to read** the vector and not modify it, **const vector<int> &arr is the best choice**.

Would you like an example demonstrating this in action? 🚀