Tutorial 16 - Call by Value & Call by Reference in C++

1. Call by Value

- **Definition:** Copies of actual parameters are passed to the function. Changes made in the function do **not** affect the original values.
- **Example:**

```
void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}
int main() {
    int x = 4, y = 5;
    cout << "Before Swap: x = " << x << ", y = " << y << endl;
    swap(x, y);
    cout << "After Swap: x = " << x << ", y = " << y << endl;
    return 0;
}

```

- **Key Points:**
  - Values of `x` and `y` remain unchanged outside the function.
  - Only copies of `x` and `y` are swapped.

---

2. Call by Pointer

- **Definition:** Addresses of actual parameters are passed to the function. Changes made in the function **affect** the original values.
- **Example:**

```
void swapPointer(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
int main() {
    int x = 4, y = 5;
    cout << "Before Swap: x = " << x << ", y = " << y << endl;
    swapPointer(&x, &y);
    cout << "After Swap: x = " << x << ", y = " << y << endl;
    return 0;
}

```

- **Key Points:**
  - The `&` operator passes addresses of variables `x` and `y`.
  - The values of `x` and `y` are swapped inside the function.

---

3. Call by Reference

- **Definition:** References of actual parameters are passed to the function. Changes made in the function **affect** the original values.
- **Example:**

```
void swapReferenceVar(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}
int main() {
    int x = 4, y = 5;
    cout << "Before Swap: x = " << x << ", y = " << y << endl;
    swapReferenceVar(x, y);
    cout << "After Swap: x = " << x << ", y = " << y << endl;
    return 0;
}

```

- **Key Points:**
  - The `&` operator allows direct reference to `x` and `y`.
  - The values of `x` and `y` are swapped inside the function.

Comparison Table

Method	Parameter Passed	Affects Original Values?	Use Case
Call by Value	Copy of variables	No	When original values must not change.
Call by Pointer	Address of variables	Yes	When you need to modify the original values using pointers.
Call by Reference	Reference variables	Yes	When you need to modify the original values without using pointers.

Complete Example Code

```
#include<iostream>
using namespace std;
void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}
void swapPointer(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
```

```
}
void swapReferenceVar(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}
int main() {
    int x = 4, y = 5;
    cout << "Call by Value:" << endl;
    cout << "Before: x = " << x << ", y = " << y << endl;
    swap(x, y);
    cout << "After: x = " << x << ", y = " << y << endl;
    cout << "\nCall by Pointer:" << endl;
    cout << "Before: x = " << x << ", y = " << y << endl;
    swapPointer(&x, &y);
    cout << "After: x = " << x << ", y = " << y << endl;
    cout << "\nCall by Reference:" << endl;
    cout << "Before: x = " << x << ", y = " << y << endl;
    swapReferenceVar(x, y);
    cout << "After: x = " << x << ", y = " << y << endl;
    return 0;
}
```