Call By Pointer

When working with functions, you can simulate call by reference by using explicit pointer parameters. Many programmers prefer to do this because it is obvious that you are passing by pointer instead of by value at the function call site.

Let's look at a `swap()` function that **exchanges the values** contained in two integers. The algorithm is simple, assuming the two parameters are `a` and `b`. Using reference parameters, we write it like this:

```
void swap(int& a, int& b) {
    int temp{a};
    a = b;
    b = temp;
}
```

When you **call the function**, there is **no indication** that `x` and `y` will be changed.

```
swap(x, y);     // does this change x and y? Can't tell here!
```

To use **explicit pointer parameters**, change the implementation of `swap()` like this:

```
void swap(int *a, int *b) {
    int temp{*a};
    *a = *b;
    *b = temp;
}
```

Then, **call the function** like this. The effect is the same as pass-by-reference, but you can tell at the call-site that `x` and `y` may be changed in the function.

```
int x{3}, y{4};
swap(&x, &y);     // explicit address passing
cout << "x->" << x << ", y->" << y << endl;
// x->4, y->3
```