**Returning from the function call**

**When do you want to?**

* **Return by Value: Stack objects**
* **Return by Address (kind of return by value): *new* keyword**
* **Return by Reference: static objects, fun() = value;**

1. Return by value, always possible (as long as the copy constructor is not deleted)
2. Return by pointer or by reference should only be done when the object being returned has a life beyond the local scope.

For example, the static objects and heap-allocated objects can be returned by pointer or by reference but the stack objects created inside the local scope should always be returned by value.

A function returning by reference can also be used on the left side of the assignment operator to assign a value to the variable returned.

Remarks:

1. ReturnByReference: Stack objects should never be returned by reference or address. Though objects of vector, map and other containers etc allocate memory on heap but the vector variable itself is on the stack, and it is completely managed by the vector class itself. In order to return it by reference or address, create the vector object on the heap and return the address of the memory (return the pointer).
2. *new* keyword - return by Value (Address): For the memory allocated by using *new,* return by address instead of reference to avoid the memory leaks. If the object is returned by reference, then you can not free the memory by calling delete.
3. Tuples and pairs: They allocate the memory on the stack. They are just structs with additional functionality. However, prefer Structs over Tuples or Pairs.

**Pass by value and pass by reference**

There are only two ways to pass data to the functions -

1. Pass by value
2. Pass by reference

Pass by address is just another way of achieving ‘pass by reference’ using pass by value. When we pass by value, the following events occur.

* The value contained in the variable being passed is copied
* And used to initialize a new variable being created in the function parameters.

So, no matter we pass a variable or a pointer, the value contained in them will be copied and passed to the function to initialize a new variable/pointer in the function.

But when we pass by reference, no new variable is created but the variable being passed is referred to.

[CodeSnippet](https://drive.google.com/file/d/1jEpmpfv3ndWMdvamHAJiIV4bFq6Y2OhR/view?usp=sharing)