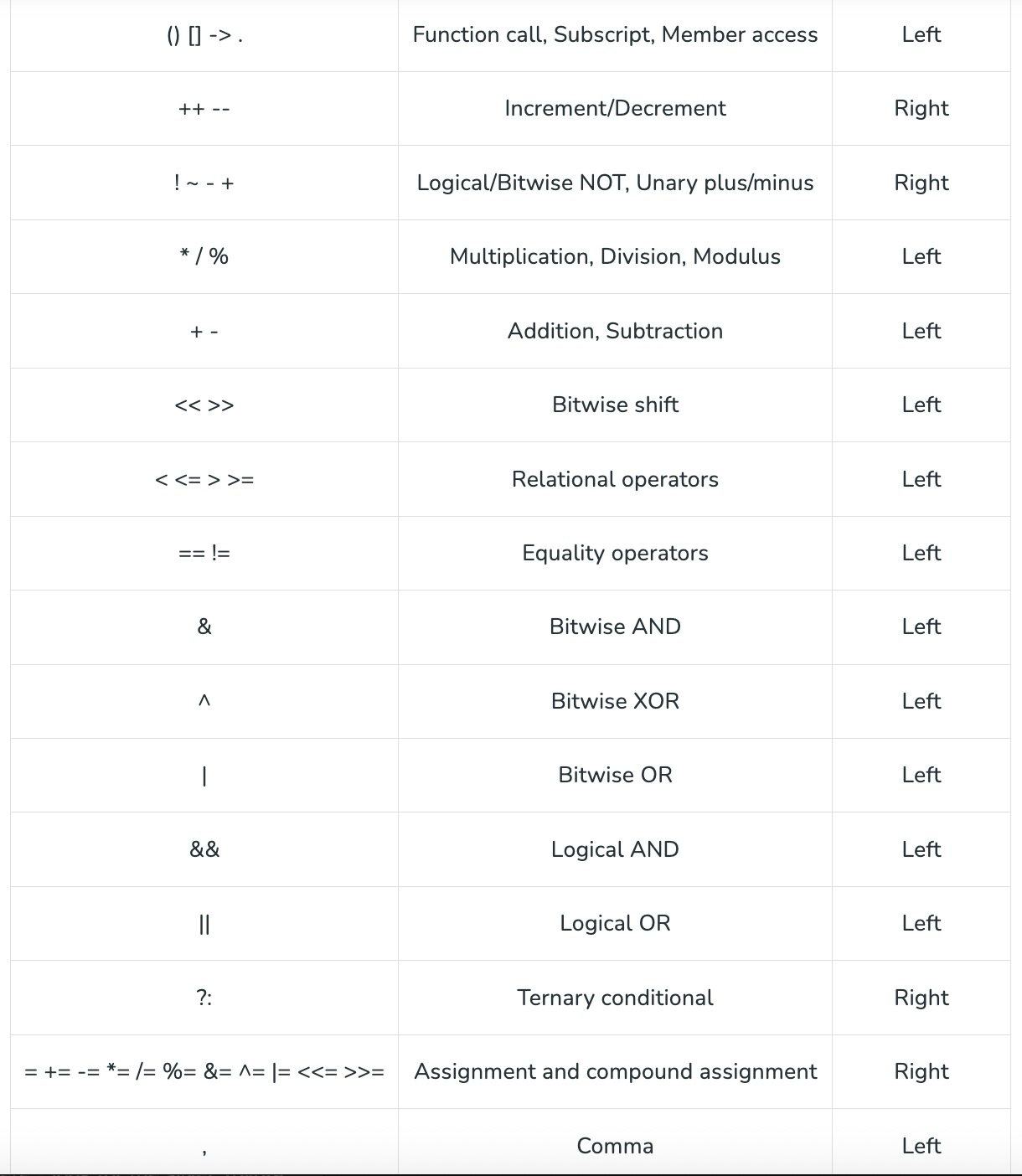
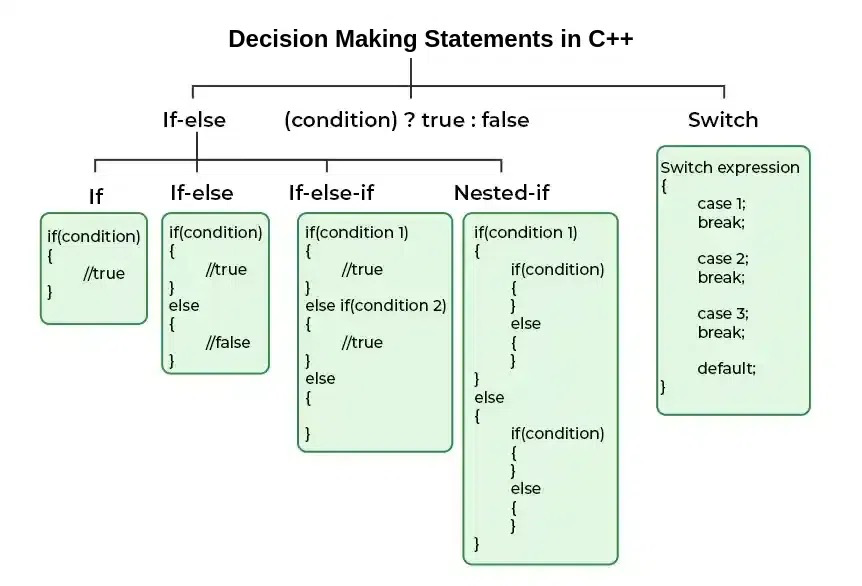


in descending precedence





Inline Functions vs Macros

| **Aspect** | **Inline Functions** | **Macros** |
| --- | --- | --- |
| **Definition** | Inline functions are functions defined with the inline keyword. | Macros are preprocessor directives defined using. #define. |
| **Scope** | Inline functions have scope and type checking, like regular functions. | Macros have no scope or type checking. They are replaced by the preprocessor. |
| **Evaluation of Arguments** | Arguments are evaluated once. | Arguments may be evaluated multiple times (e.g., in expressions). |
| **Handling** | Inline functions are handled by the compiler. | Macros are handled by the preprocessor. |
| **Private Members** | Can access private members of a class. | Cannot access private members of a class. |
| **Execution Overhead** | Compiler may ignore the inline request if the function is too large. | Macros are always substituted into code. |
| **Recursion** | Inline functions can call themselves recursively. | Macros cannot be recursive. |

Pointer vs Reference

| Aspect | Pointer | Reference |
| --- | --- | --- |
| Initialization | A pointer can be initialized after declaration. | A reference must be initialized at the time of declaration. |
| Nullability | A pointer can be assigned NULL or nullptr. | A reference cannot be null, it must always refer to a valid object. |
| Reassignment | A pointer can be reassigned to point to different objects. | A reference cannot be reassigned once it is bound to an object. |

Pointer vs Smart Pointer

| **Pointer** | **Smart Pointer** |
| --- | --- |
| A pointer is a variable that maintains a memory address as well as data type information about that memory location. A pointer is a variable that points to something in memory. | Smart pointers, in simple words, are classes that wrap a pointer, or scoped pointers. |
| It is not destroyed in any form when it goes out of its scope | It destroys itself when it goes out of its scope |
| Pointers are not so efficient as they don't support any other feature. | Smart pointers are more efficient as they have an additional feature of memory management. |
| They are very labor-centric/manual. | They are automatic/pre-programmed in nature. |