Functions

```
/*

        . First hand on functions
        . Declaration and definition
        . Multiple files
        . Declarations and definitions in the main.cpp file
            . minimum
            . maximum
            . inc_mult
        . Multiple files:
            . declarations and definition separated in module file: math.ixx
            . Putting definitions in math_impl.cpp
            . Separating the declarations and definitions across operation and compare modules.
*/
```

Functions: in the main file

```
//Declarations and definitions in one place
int maximum(int a, int b) {
    if (a > b)
        return a;
    else
        return b;
}


int inc_mult(int a, int b) {
    int result = ((++a) * (++b));
    return result;
}

int main(){

    int a{ 10 };
    int b{ 20 };
    int result{};

    result = maximum(a, b);
    print_number(result);

    result = inc_mult(a, b);
    print_number(result);
}
```

Functions: declarations and definitions separated

```
//Declarations
int maximum(int a, int b);
int inc_mult(int a, int b);

int main(){

    int a{ 10 };
    int b{ 20 };
    int result{};

    result = maximum(a, b);
    print_number(result);

    result = inc_mult(a, b);
    print_number(result);
}

//Definitions
int maximum(int a, int b) {
    if (a > b)
        return a;
    else
        return b;
}


int inc_mult(int a, int b) {
    int result = ((++a) * (++b));
    return result;
}
```

Functions shipped into an external module: utilities

```
//The utilities module purview
//Declarations in an export block
export {
    int maximum(int a, int b);
    int minimum(int a, int b);
    int inc_mult(int a, int b);
}

//Implementations
int maximum(int a, int b) {
    if (a > b)
        return a;
    else
        return b;
}

int inc_mult(int a, int b) {
    int result = ((++a) * (++b));
    return result;
}

//If the utilities module is imported in main, we'll have access to these
//You can also export each declaration separately.
//Notice that we just need to export the declaration.
//We could have just exported the definition alltogether.
```

Shipping the definitions into a module implementation file

```
module;

module utilities;
//The declarations still live in the module interface file
//Implementations
int maximum(int a, int b) {
    if (a > b)
        return a;
    else
        return b;
}

int inc_mult(int a, int b) {
    int result = ((++a) * (++b));
    return result;
}
```

```
//The module implementation file is accounted for in the CMakeLists.txt file
add_executable(${PROJECT_NAME} main.cpp math_impl.cpp compare.cpp operations.cpp)

target_sources(${PROJECT_NAME}
        PUBLIC
        FILE_SET CXX_MODULES FILES
        utilities.ixx
        compare.ixx
        operations.ixx
)
```

Shipping the definitions into a module implementation file

```
module;

module utilities;
//The declarations still live in the module interface file
//Implementations
int maximum(int a, int b) {
    if (a > b)
        return a;
    else
        return b;
}

int inc_mult(int a, int b) {
    int result = ((++a) * (++b));
    return result;
}
```

```
//The module implementation file is accounted for in the CMakeLists.txt file
add_executable(${PROJECT_NAME} main.cpp math_impl.cpp compare.cpp operations.cpp)

target_sources(${PROJECT_NAME}
        PUBLIC
        FILE_SET CXX_MODULES FILES
        utilities.ixx
        compare.ixx
        operations.ixx
)
```

Breaking the functions across different modules

```
//The module implementation file is accounted for in the CMakeLists.txt file
add_executable(${PROJECT_NAME} main.cpp math_impl.cpp compare.cpp operations.cpp)

target_sources(${PROJECT_NAME}
        PUBLIC
        FILE_SET CXX_MODULES FILES
        utilities.ixx
        compare.ixx
        operations.ixx
)
```

Functions: A few more secrets

```
//Parameters: one or more
//Parameters are copied by default
//Have fun!
```