Creating a Function

In this lesson, we will learn how to create functions in C++ and use them in our program.

WE'LL COVER THE FOLLOWING ^

- Declaration
- Definition

Similar to variables, functions need to be defined before compilation. Every function has a name and a set of operations it needs to perform. The first part of creating a function **declaration**.

Declaration

The declaration of a function means defining its **name**, **type**, and **argument(s)**. This may sound confusing right now but we'll get the hang of it really soon. Here's the template for function declaration:

```
type functionName(argument(s));
```

Let's take a look at the three components one by one:

- type refers to the type of value the function produces. In formal terms, we say that a function **returns** something when we use it. The type of object it returns must be specified in the declaration. So, if a function returns a number, its type would be int or double etc. Any data type available in C++ can be used as the function's type. If the function does not return anything, its type will be void.
- functionName is simply the label we'll use for the function, just like we do for variables.
- arguments are the inputs of a function. We can give a function different

objects as arguments, and the function will then perform operations using them. For each argument, a data type and name must be defined. A function could also have no arguments at all. In that case, the arguments section remains empty.

Now that we've been through all the components of function declaration, let's see a few examples:

Note: It is a good practice to give meaningful names to the function and its arguments. That makes the code much more readable.

It's time to move on to the second part of creating a function: **definition**.

Definition

The definition (also known as implementation) of a function refers to the set of instructions which the function performs. Without the definition, a function will not know what to do. Hence, we need to make sure our implementation is flawless and doesn't produce any bugs. Here's the template for the definition of a function:

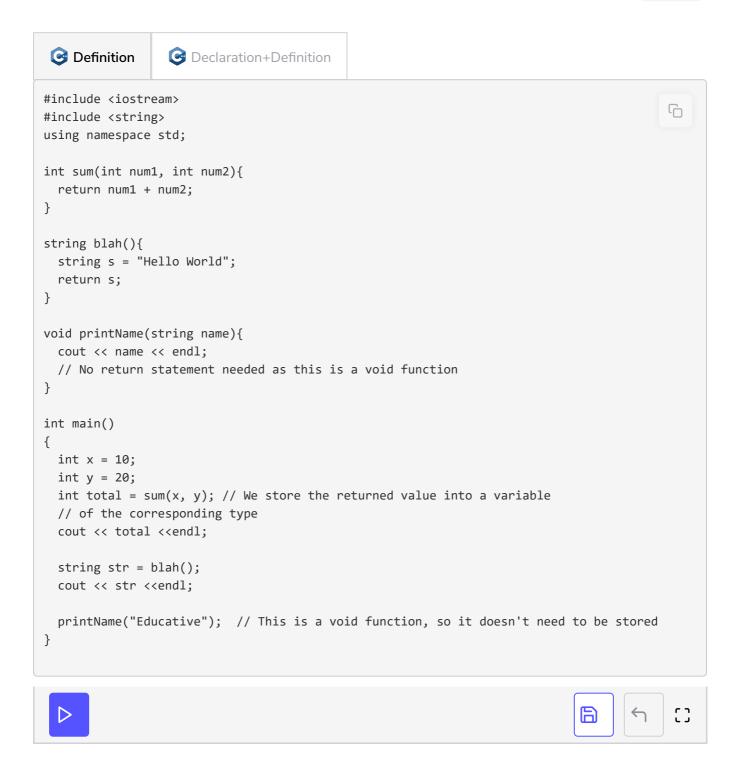
```
type functionName(arguments) {
  // Definition
}
```

The curly braces, {}, contain the definition of a function. This is known as the **scope** of the function (more on this later). Here, we can write a normal C++ code which will execute once every time the function is called.

You may have noticed by now that the int main(){} in a C++ program is also a function! It is the function that the compiler runs to execute our code.

the function.

Below, we've defined a few functions and called them in the main program. The first approach only uses the definition to create a function. The second approach first declares the functions and then defines them below the main().



At this point, we're equipped to write basic functions. The basic structure of the functions is clear. Remember, a function can call other functions as well. After all, the main function does that as well.

In the next lesson, we will learn some interesting details about the arguments of a function.