Functions

Terms

Debugging
Functions
Function arguments
Function declaration
Function definition
Function parameters
Function prototype
Function signature
Global variables
Invoking a function
Header files
Local variables
Namespaces
Overloading functions

Summary

- A *function* is a group of one or more statements that perform a task. Each function should have a clear responsibility. It should do one and only one thing.

- A function can have zero or more *parameters*

- *Arguments* are the values passed to a function.

- To *call* (or *invoke*) a function, we type its name, followed by parenthesis, and the arguments (if any).

- Function parameters can have a default value. This way, we don't have to provide arguments for them.

- The *signature* of a function includes the function name, and the number, order, and type of parameters.

- *Overloading* a function means creating another variation with a different signature. By overloading functions, we can call our functions in different ways.
- Arguments of a function can be passed by value or reference. When passed by value, they get copied to the parameters of the function.
- To pass an argument by a reference, we should add an & after the parameter type.
- *Local variables* are only accessible within the function in which they are defined. *Global variables* are accessible to all functions.
- Global variables can lead to hard-to-detect bugs and should be avoided as much as possible.
- A *function declaration* (also called a *function prototype*) tells the compiler about the existence of a function with a given signature. A *function definition* (or implementation) provides the actual body (or code) for the function.
- As our programs grow in more complexity, it becomes critical to split our code into separate files.
- A *header file* ends with "**.h**" or "**.hpp**" extension and consists of function declarations and constants. We can import header files using the #**include** directive.
- An *implementation file* ends with "**.cpp**" extension and consists of function definitions.
- Using *namespaces* we can prevent name collisions in our programs.
- *Debugging* is a technique for executing a program line by line and identifying potential errors.

```
// Defining functions
void greet(string name) {
    cout << "Hello " << name;
}

string fullName(string firstName, string lastName) {
    return firstName + " " + lastName;
}


// Parameters with a default value
double calculateTax(double income, double taxRate = .3) {
    return income * taxRate;
}


// Overloading functions
void greet(string name) {
}

void greet(string title, string name) {
}


// Reference parameters
void increase(double& number) {
    number++;
}
```

```cpp
// Function declaration
void greet(string name);


// Defining a namespace
namespace messaging {
    void greet(string name) {}
}


// Using a namespace
using namespace messaging;
// or
using messaging::greet;
```