11.1 — Function parameters and arguments

ALEX JUNE 17, 2021

In Chapter 2, we covered function basics in the following sections:

- 2.1 -- Introduction to functions
- 2.3 -- Introduction to function parameters and arguments
- 2.6 -- Forward declarations and definitions
- 2.7 -- Programs with multiple code files
- 2.10 -- Header files

You should be familiar with the concepts discussed in those lessons before proceeding.

Parameters vs Arguments

In the next three lessons, we'll talk quite a bit about parameters and arguments, so let's revisit those definitions before proceeding.

In common usage, the terms parameter and argument are often interchanged. However, for the purposes of further discussion, we will make a distinction between the two:

A **function parameter** (sometimes called a **formal parameter**) is a variable declared in the function declaration:

```
void foo(int x); // declaration (function prototype) -- x is a
parameter

void foo(int x) // definition (also a declaration) -- x is a
parameter
{
}
```

An **argument** (sometimes called an **actual parameter**) is the value that is passed to the function by the caller:

```
foo(6); // 6 is the argument passed to parameter x
foo(y+1); // the value of y+1 is the argument passed to
parameter x
```

When a function is called, all of the parameters of the function are created as variables, and the value of the arguments are copied into the parameters. For example:

```
void foo(int x, int
y)
{
}

foo(6, 7);
```

When foo() is called with arguments 6 and 7, foo's parameter x is created and assigned the value of 6, and foo's parameter y is created and

assigned the value of 7.

Even though parameters are not declared inside the function block, function parameters have local scope. This means that they are created when the function is invoked, and are destroyed when the function block terminates:

```
1 void foo(int x, int y) // x and y are created
  here
  {
  } // x and y are destroyed here
```

There are 3 primary methods of passing arguments to functions: pass by value, pass by reference, and pass by address. We'll look at each of those in the next set of lessons.

Next lesson
11.2 Passing arguments by value

Back to table of contents

Previous lesson
10.x Chapter 10 comprehensive quiz

Leave a comment... Put C++ code between triple-backticks (markdown style):```Your C++ coc

Name*

Email*

Avatars from https://gravatar.com/ are connected to your provided email address.

Notify me about replies: POST COMMENT

DP N N FOUT

Newest