7.1 — Function parameters and arguments

BY ALEX ON JULY 18TH, 2007 | LAST MODIFIED BY ALEX ON JANUARY 23RD, 2020

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

- **2.1 -- Introduction to functions**
- **2.3 -- Introduction to function parameters and arguments**
- **2.7 -- Forward declarations and definitions**
- **2.8 -- Programs with multiple code files**
- **2.11 -- 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:

```
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.

7.2 -- Passing arguments by value

Index

6.x -- Chapter P.6 comprehensive quiz

C++ TUTORIAL | PRINT THIS POST

51 comments to 7.1 — Function parameters and arguments

Wallace
November 15, 2019 at 2:33 pm · Reply

Minor typo: The opening sentence mentions Chapter 1 but all linked sections below are from Chapter 2. I suspect refactoring. :)

ryder
January 13, 2019 at 7:07 am · Reply

Why this happen?

```
#include "pch.h"
#include <iostream>

struct A {};
void foo(A const& a) {}
void callFoo() {
    foo(A());
}

```

```
int main()
{
    std::cout << "Hello World!\n";
}

```

The compile okay and programm works.

```
   #include "pch.h"
#include <iostream>

int main()
{
    struct A {};
    void foo(A const& a) {}
    void callFoo() {
        foo(A());
    }
    std::cout << "Hello World!\n";
}

```

Even I put ";" at the end of function:

```
#include "pch.h"
#include <iostream>

int main()
{
    struct A {};
    void foo(A const& a) {};
    void callFoo() {
        foo(A());
    };

    std::cout << "Hello World!\n";
}

```

The complier still complain about ";".

My question is what ";" really means? Why it works when the function is defined out of the main function, but not within the main function?

nascardriver
January 13, 2019 at 7:55 am · Reply

You cannot define functions inside functions.
If you want to define a function in-line, have a look at lambda-functions.

Samira Ferdi
November 7, 2018 at 9:11 pm · Reply

Hi Alex and Nascardriver! I don't understand the meaning of the term 'copied' in sentence 'the value of the arguments are copied into the parameters'. Could you explain to me? cause it's confuse me.

nascardriver
November 8, 2018 at 7:34 am · Reply

Think of the parameter as a variable. Someone the value for that variable has to come from the caller to the function.
Think of it like this:

```
int i{ 123 };
int j{ i };
```

@i's value was copied to @j. They have the same value, but they're not the same. When you call a function, the same thing happens.

Steven
January 20, 2018 at 10:16 am · Reply

Hi!
I'm preparing for an IT exam in University and I'm facing a little problem. In one exercise it ask to write the prototype of the function fz, giving you these details:

int x;
double y[100];

I have to write the prototype based on this last information:

1.5 + fz(y, y[4], x + y[x]);

in a way that it is possible.

My question is: is it possible?
'cause searching everywhere, I can't find anything that says that is correct to identify a var with (x+y[x]), so I wanted to ask you if it is correct.
In that case my prototype should be: int fz (int[], int[], int[]); is it correct?

Thanks for your answer!

nascardriver
January 20, 2018 at 10:37 am · Reply

Hi Steven!
y is a double array, so int[] is wrong.
Judging by the context fz returns a double (1.5 + fz()).
The first parameter to fz is y, we know y is a double[].
The second parameter is y[4], since y is a double array every element in y is a double.
The third parameter is (int + double) which is a double.

```
double fz(double, double[], double)
```

Alex
January 23, 2018 at 8:38 pm · Reply

The answer I'd go with would be:
double fz(double[], double, double)

y is defined as a double array, so the first parameter is clearly double[].
y[4] is the type of element 4 in the double array, so the second parameter is clearly double.
An int plus a double yields a double result, so the third parameter is also a double.

The return value isn't well defined (it could be anything), but since it's being added to a double, it's most likely a double.

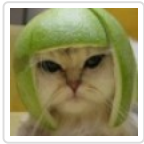
Hajra Farooq
December 6, 2017 at 12:32 pm · Reply

I had to enter a series of even number,but it is not working as it should.

```
#include <iostream>
using namespace std;

bool number(int x);
int main()
{
   int x = 0,i=0;
   while (i!=-1)
   {
      cout << number(x);
      if (number(x) == true)
         cout << "Its even"<<endl;
      else
         cout << "its odd"<<endl;
   }
   return 0;
}
bool number(int x = 0)
{
   int t = 0;

   cout << "Enter Number";
   cin >> x;
      t = x % 2;
      if (t = 0)
      {
         return true;
      }
      else
      {
         return false;
      }
}
```



Alex
December 7, 2017 at 9:59 pm · Reply

There are several errors here. First, in function number(), you're using assignment (=) instead of comparison (==). Second, you call number(x) from main() twice per iteration. You should only call it once (stick the result in a variable if need be).

Michael
November 20, 2017 at 10:35 pm · Reply

Hi Alex,
Just write this down to thank you for putting such a comprehensive and yet easy-to-pick-up

tutorial for c++. I have been slacking off for several days since finishing coding the blackjack at the end of Chapter 6. It's super rewarding experience. Now ready to move on!

Thank you!

b1
February 19, 2017 at 5:06 am · Reply

excellent job Alex, please continue your great work
i have a questions:

1. i heard a lot about (Passing a functions AS parameters to another function)
but i have never seen an example of it in web
i imaging it like this (also so you know what i mean)

```
int foo()
{

}

int foo2( foo() )
{

}
```

is it even valid ?
if yes, is it safe ?

2. i want to use a variable from another function

```
int foo()
{
int x1 = 55;
}

int foo2()
{
 // how to use the variable here in normal way like this
x1 = 100;
}
```

Alex
February 19, 2017 at 1:13 pm · Reply

1) I talk about passing functions as parameters to other functions in the section on function pointers.
2) There is no way to directly access the x1 from function foo from foo2. This is intentional, as variable x1 is destroyed as soon as function foo() is done. So when foo2() is executing, x1 doesn't even exist at that point.

B1
February 20, 2017 at 10:34 pm · Reply

thank you Alex

Emmanuel Muniko
December 5, 2016 at 1:23 am · Reply

function parameter =variable declared in function declaration
function argument =value passed to the declared variable
simple as that

vineet singh
November 11, 2016 at 5:59 am · Reply

nice

yekyawtun
September 2, 2016 at 8:38 am · Reply

Dear Admin,

I would like to learn about multithreading in visual studio on window 64 bits. Can you give me any suggestion about that topic? Which resources are suitable for me?

Thank you very much for your support and help,
Best Regards,
Ye Kyaw Tun

Esraa Alkhalaila
June 14, 2016 at 10:16 pm · Reply

how I can treat the inputs passed by arguments???

Alex
June 16, 2016 at 12:27 pm · Reply

I'm sorry, I don't understand the question.

Adib Khazaee
May 29, 2016 at 3:26 am · Reply

Hi Alex. Can I declare an enum Variable as a Function parameter? For instance a function with a signature like this:
int printenumString(enum AAAA)

ps. So grateful for the amazing tutorial!

Alex
May 29, 2016 at 8:08 am · Reply

Yes, absolutely.

Adib Khazaee
May 29, 2016 at 10:43 pm · Reply

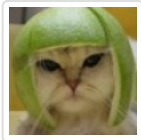
Then, how come doesn't this compile?

```
void enumInt(enum A)
{
   std::cout << A << "\n";
```

}
The compiler ends with this error:

error C2275: 'A' : illegal use of this type as an expression

ps. Tnx man, I appreciate it.



Alex
May 30, 2016 at 9:17 am · Reply

Because "enum" isn't a type, it's a keyword used to make enumerated types.

Try something like this:

```
#include <iostream>
enum Foo
{
    ONE,
    TWO
};

void whatever(Foo foo)
{
    std::cout << foo;
}
```

Adib khazaee
May 30, 2016 at 7:53 pm · Reply

Got it...thank u very much for the enlightenment

Adib khazaee
May 30, 2016 at 8:22 pm · Reply

Now that I think about it, it was stupid of me. Sorry for wasting ur time!

Darren
June 10, 2016 at 5:08 am · Reply

Somebody once said, I don't know who, there are no stupid questions only stupid answers. In other words ask questions and learn, the only stupid thing you could do is not to ask questions; don't beat yourself up.

Adib Khazaee
June 10, 2016 at 10:56 pm · Reply

tnx man, I appreciate it.

asad
December 7, 2016 at 10:56 am · Reply

mam why we use two time variable in finction for example we write in start (int x ,int y) and then in between we use a and b instead of x and y i dont understand

Alex
December 7, 2016 at 12:28 pm · Reply

I don't understand what you're asking. What are a and b in this context?

Shiva
April 16, 2016 at 2:42 am · Reply

Declaration or Definition?

In the first example and the statement just above it you used the word *declaration*, where you really talk about *definition*. Typo?

Alex
April 17, 2016 at 2:43 pm · Reply

Yup, typo. Fixed.

kenny7891
December 12, 2015 at 11:35 pm · Reply

Hi Alex! I have extremely enjoyed your guide and have learned a ton. In terms of learning the language, is it common to have to refer back to previous lessons? I ask this because I find myself looking back to remember how to do things. For example, I completely understand the concepts (pointers, for each loops, etc.) but when I want to implement them I don't remember the exact syntax. I guess I want to make sure I am okay to continue into future lessons even if I need to look back to remember how to implement certain concepts. Thanks a bunch!!

Alex
December 15, 2015 at 4:22 pm · Reply

Yes! It's totally expected. I have to look up the specific syntax for things all the time.

What's important is that you understand the concepts themselves, and that when you see a code example, you can appropriately match the syntax to a concept (e.g. if you see an &, you know that this can mean address-of or reference depending on context).

Muhammad Tanzeel Khalid
October 31, 2015 at 5:16 am · Reply

8. When arguments are passed by value, the function works with the original arguments in the calling program. T / F

please reply??

Alex
November 5, 2015 at 8:29 pm · Reply

What do you think, and why?

Tony

August 26, 2015 at 5:42 am · Reply

Hey Alex, I'm reading your guide, but I'm a little worried; Are you going to be updating the rest of the guide anytime soon? I'd like to be caught up with the latest design patterns! Thank you!

Alex
August 26, 2015 at 11:41 am · Reply

I'm working on it, but I can only write so fast! :)

I've been putting dates on all the articles as they get updated, so check back occasionally and re-read the articles again as they get updated.

Priscilla
May 31, 2015 at 4:41 am · Reply

thanks...this helped alot

Mihai
April 10, 2015 at 10:45 pm · Reply

Hello,
Do you have any ideea how could I make the function foo with a variable number of parameters?

Let's say that I would like to make it do a sum of n numbers. Those parameters will be insterted from keyboard and can be 2, 3, 4, 5..

Alex
April 11, 2015 at 11:02 am · Reply

If the user will be entering the numbers, then you'll probably want to write a function that takes an array of numbers the user entered. The function can step through the array and process each element in whatever fashion it desires (e.g. add them up). In this way, you don't need a variable number of parameters; Instead, the array is variable length to accommodate the number of user inputs.

Diana
February 21, 2015 at 11:46 am · Reply

You should specify here that the call of function should be done in the main function :D

Alex
October 12, 2015 at 4:25 pm · Reply

It doesn't need to be -- non-main functions can call other functions.

chesslover
September 21, 2011 at 9:34 pm · Reply

Thanks for the lessons!
They are amazingly well explained!
Congratulations!

ender
October 28, 2010 at 4:26 pm · Reply

This helped thanks

koorosh
February 7, 2010 at 5:37 am · Reply

what are actual argument(actual parameter) and formal argument(formal parameter)?!
they use in some books!
what difference between argument and parameter?
please explain

THE TERMINATOR
August 23, 2010 at 12:37 pm · Reply

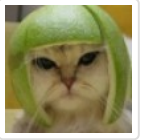
Think of it like this: an argument is the giver and the paremeter is the reciever.

Ankit Mishra
September 30, 2015 at 3:31 am · Reply

Agreed with this

Devashish
September 30, 2015 at 7:19 am · Reply

As far as I know, argument is not the giver. Argument has to be "given".
Parameters are variables that receive the argument as input, with which the called function would work.



Alex
September 30, 2015 at 11:05 am · Reply

It would probably be most accurate to say the caller is the giver, and the arguments are what is given.

But the overall sentiment is directionally correct -- arguments provide data, parameters receive it.

Shriram M
September 4, 2010 at 11:37 am · Reply

Actual parameters are nothing but the actual values/arguments you are passing during function call[ex: foo(6,7);].
Formal parameters are the parameters into function definition [ex: int foo(int i, int j)]

Meghnadh
October 19, 2015 at 11:57 pm · Reply

Actual parameter is the value or any other input passed to the function by user when function is invoked .....
Formal parameter is the variable that is declared when function is declared.....