**Default values in parameters.**

In C++, functions can also have optional parameters, for which no arguments are required in the call, in such a way that, for example, a function with three parameters may be called with only two. For this, the function shall include a default value for its last parameter, which is used by the function when called with fewer arguments. For example:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 | // default values in functions  #include <iostream>  using namespace std;  int divide(double a, double b = 2)  {  double r = a / b;  cout << r << '\n';  return (r);  }  int main()  {  divide(16);  divide(81, 10);  return 0;  } | 8  8.1 | [Edit & Run](https://cplusplus.com/doc/tutorial/functions/) |

In this example, there are two calls to function *divide*. In the first one:

|  |  |  |
| --- | --- | --- |
|  | divide (16) |  |

The call only passes one argument to the function, even though the function has two parameters. In this case, the function assumes the second parameter to be 2 (notice the function definition, which declares its second parameter as *int b=2*). Therefore, the result is 8.

In the second call:

|  |  |
| --- | --- |
|  | divide (81, 10) |

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 | // default values in functions  #include <iostream>  using namespace std;  int divide(double a, double b = 2)  {  double r = a / b;  cout << r << '\n';  return (r);  }  int main()  {  divide(16);  divide(81, 10);  return 0;  } | // default values in functions  #include <iostream>  using namespace std;  double divide(double a, double b = 2)  {  double r;  r = a / b;  return (r);  }  int main()  {  cout << divide(16) << '\n';  cout << divide(81, 10) << '\n';  return 0;  } |

The call passes two arguments to the function. Therefore, the default value for *b* (*int b=2*) is ignored, and *b* takes the value passed as argument, that is 10, yielding a result of 8.1.

\*Compare above two the same programs which are not actually the same in coding manner!