What is a formal parameter?

Asked 11 years ago Modified 11 years ago Viewed 49k times

32

When compiling in C++ I often end up with error messages dealing with "formal parameters", such as

```
error C2719: 'b': formal parameter with __declspec(align('16')) won't be aligned
```

I do understand the error, and the fact that `b` is a parameter of a function I am defining.

However, what does it mean that a parameter is *formal*? Can there be *informal* parameters as well?

I do notice that the term "formal parameter" appears in other languages as well, so I presume it is a more generic term not necessarily specific to C-family of languages? Are informal parameters supported by some subset of languages?

Upon seeing the answers, one final question: Where those names *formal parameter* and *actual parameter* origin from? Does it origin from the C standard, or is it an effect of calling it as such in some abstract language calculus?

c++ c language-agnostic language-design

Share Improve this question Follow edited Sep 18, 2013 at 11:59 asked Sep 18, 2013 at 10:46

CygnusX1
21.7k 5 69 113

In this particular case it means you're trying to pass more than three SSE parameters to a function and you're stuck with Visual Studio and the WIN32 ABI. ;-) – Paul R Sep 18, 2013 at 10:53

@Paul R: In this particular case I was passing an aligned parameter by value and not `const&` . – CygnusX1 Sep 18, 2013 at 11:56

3 Regarding where the names come from: `formal` in mathematics is not the opposite of `informal` . `formal` comes from `form` , i.e. `formal` means well-defined. For example formal logic is not the opposite of informal logic, but a logic based on well-defined rules. Formal parameters therefore refer to the parameters that define the form of the function. Actual parameter is then obvious I think. – Shahbaz Sep 18, 2013 at 12:30

But then what is the opposite? "not well-defined"? I guess this shifts the discussion to the linguistic aspect so... never mind! – CygnusX1 Sep 18, 2013 at 14:32

3 Answers

Sorted by: Highest score (default)

48

There are *formal* and *actual* parameters:

```
void foo(int arg); //arg is a formal parameter

int main()
{
    int val = 1;
    foo(val);  //val is an actual parameter
}
```

From C++ Standard:

1.3.1 formal parameter (parameter)

> an object or reference declared as part of a function declaration or definition, or in the catch clause of an exception handler, that acquires a value on entry to the function or handler; an identifier from the comma-separated list bounded by the parentheses immediately following the macro name in a function-like macro definition; or a template-parameter. Parameters are also known as formal arguments or formal parameters.

1.3.10 actual parameter (argument)

> an expression in the comma-separated list bounded by the parentheses in a function call expression, a sequence of preprocessing tokens in the comma-separated list bounded by the parentheses in a function-like macro invocation, the operand of throw, or an expression, type-id or template-name in the comma-separated list bounded by the angle brackets in a template instantiation. Also known as an actual argument or actual parameter.

Share Improve this answer Follow

edited Sep 18, 2013 at 13:08

answered Sep 18, 2013 at 10:50

cpp
3,791 3 26 38

19 Is should be noted that these terms (formal and actual parameter) are deprecated by the C standard. The correct terms are "parameter" and "argument". – R.. GitHub STOP HELPING ICE Sep 18, 2013 at 10:56

@R.. But they still function in other languages I believe? What's the source? – CygnusX1 Sep 18, 2013 at 11:57

Source is the C standard, 3.3 and 3.15 in C99. – R.. GitHub STOP HELPING ICE Sep 18, 2013 at 12:40

I meant, what is the source, the reason of this naming used in the standard. It seems the authors of the C standard did not invent this naming, but it was present before. They just explicitly (re-)defined it for C.

– CygnusX1 Sep 18, 2013 at 14:34

17

Formal parameters are the parameters known at the function definition. The actual parameters are what you *actually* (hence the name) pass to the function when you call it.

```
void foo( int a ); // a is a formal parameter

foo(10); // 10 is the actual parameter
```

Share Improve this answer Follow

answered Sep 18, 2013 at 10:49
SingerOfTheFall
29.9k 8 71 105

3 I thought the thing that you pass to a function is an "argument", while the thing that a function has is a "parameter"? – CygnusX1 Sep 18, 2013 at 11:33

10

It's a matter of being a little pedantic over terminology, but quite useful: The formal parameters are what you just think of function parameters:

```
int foo(bool a, float b);
```

Here `a` and `b` are formal parameters. The point is that in the function body, you're referring to those parameters "formally" without actually knowing their value. It is only when you actual *evaluate a function call expression* that the formal function parameters are *bound* to the function call arguments:

```
int result = foo(false, 1.5);
```

In this call expression, the value `false` of the first argument is bound to the formal parameter `a`, and similarly for the second argument.

The distinction between parameters and arguments is maybe more important to language designers and comiler writers, but as an example in C++, it can be very helpful to get your head around this when you're trying to follow the rules for template argument deduction.

Share Improve this answer Follow

edited Sep 18, 2013 at 10:56

answered Sep 18, 2013 at 10:50
Kerrek SB
475k 93 892 1.1k