

# Argument Passing and Naming Conventions

Article • 08/03/2021

## Microsoft Specific

The Microsoft C++ compilers allow you to specify conventions for passing arguments and return values between functions and callers. Not all conventions are available on all supported platforms, and some conventions use platform-specific implementations. In most cases, keywords or compiler switches that specify an unsupported convention on a particular platform are ignored, and the platform default convention is used.

On x86 platforms, all arguments are widened to 32 bits when they are passed. Return values are also widened to 32 bits and returned in the EAX register, except for 8-byte structures, which are returned in the EDX:EAX register pair. Larger structures are returned in the EAX register as pointers to hidden return structures. Parameters are pushed onto the stack from right to left. Structures that are not PODs will not be returned in registers.

The compiler generates prolog and epilog code to save and restore the ESI, EDI, EBX, and EBP registers, if they are used in the function.


### Note

When a struct, union, or class is returned from a function by value, all definitions of the type need to be the same, else the program may fail at runtime.

For information on how to define your own function prolog and epilog code, see [Naked Function Calls](#).

For information about the default calling conventions in code that targets x64 platforms, see [x64 Calling Convention](#). For information about calling convention issues in code that targets ARM platforms, see [Common Visual C++ ARM Migration Issues](#).

The following calling conventions are supported by the Visual C/C++ compiler.

 Expand table

Keyword	Stack cleanup	Parameter passing
<a href="#">__cdecl</a>	Caller	Pushes parameters on the stack, in reverse order (right to left)
<a href="#">__clrcall</a>	n/a	Load parameters onto CLR expression stack in order (left to right).
<a href="#">__stdcall</a>	Callee	Pushes parameters on the stack, in reverse order (right to left)
<a href="#">__fastcall</a>	Callee	Stored in registers, then pushed on stack
<a href="#">__thiscall</a>	Callee	Pushed on stack; <code>this</code> pointer stored in ECX
<a href="#">__vectorcall</a>	Callee	Stored in registers, then pushed on stack in reverse order (right to left)

For related information, see [Obsolete Calling Conventions](#).

END Microsoft Specific

## See also

[Calling Conventions](#)

---

## Feedback

Was this page helpful?



[Provide product feedback](#)  | [Get help at Microsoft Q&A](#)