`__thiscall`

Article • 08/03/2021

The **Microsoft-specific** `__thiscall` calling convention is used on C++ class member functions on the x86 architecture. It's the default calling convention used by member functions that don't use variable arguments (`vararg` functions).

Under `__thiscall`, the callee cleans the stack, which is impossible for `vararg` functions. Arguments are pushed on the stack from right to left. The `this` pointer is passed via register ECX, and not on the stack.

On ARM, ARM64, and x64 machines, `__thiscall` is accepted and ignored by the compiler. That's because they use a register-based calling convention by default.

One reason to use `__thiscall` is in classes whose member functions use `__clrcall` by default. In that case, you can use `__thiscall` to make individual member functions callable from native code.

When compiling with /clr:pure, all functions and function pointers are `__clrcall` unless specified otherwise. The `/clr:pure` and `/clr:safe` compiler options are deprecated in Visual Studio 2015 and unsupported in Visual Studio 2017.

`vararg` member functions use the `__cdecl` calling convention. All function arguments are pushed on the stack, with the `this` pointer placed on the stack last.

Because this calling convention applies only to C++, it doesn't have a C name decoration scheme.

When you define a non-static class member function out-of-line, specify the calling convention modifier only in the declaration. You don't have to specify it again on the out-of-line definition. The compiler uses the calling convention specified during declaration at the point of definition.

See also

Argument passing and naming conventions

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A