# __fastcall

Article • 09/16/2023

**Microsoft Specific**

The `__fastcall` calling convention specifies that arguments to functions are to be passed in registers, when possible. This calling convention only applies to the x86 architecture. The following list shows the implementation of this calling convention.

Expand table

| Element | Implementation |
|---|---|
| Argument-passing order | The first two `DWORD` or smaller arguments that are found in the argument list from left to right are passed in ECX and EDX registers; all other arguments are passed on the stack from right to left. |
| Stack-maintenance responsibility | Called function pops the arguments from the stack. |
| Name-decoration convention | At sign (@) is prefixed to names; an at sign followed by the number of bytes (in decimal) in the parameter list is suffixed to names. |
| Case-translation convention | No case translation performed. |
| Classes, structs, and unions | Treated as "multibyte" types (regardless of size) and passed on the stack. |
| Enums and enum classes | Passed by register if their underlying type is passed by register. For example, if the underlying type is `int` or `unsigned int` of size 8, 16, or 32 bits. |

> **Note**
>
> Future compiler versions may use different registers to store parameters.

Using the /Gr compiler option causes each function in the module to compile as `__fastcall` unless the function is declared by using a conflicting attribute, or the name of the function is `main`.

The `__fastcall` keyword is accepted and ignored by the compilers that target ARM and x64 architectures; on an x64 chip, by convention, the first four arguments are passed in registers when possible, and additional arguments are passed on the stack. For more information, see x64 Calling Convention. On an ARM chip, up to four integer arguments and eight floating-point arguments may be passed in registers, and additional arguments are passed on the stack.

For nonstatic class functions, if the function is defined out-of-line, the calling convention modifier doesn't have to be specified on the out-of-line definition. That is, for class non-static member methods, the calling convention specified during declaration is assumed at the point of definition. Given this class definition:

C++

```
struct CMyClass {
   void __fastcall mymethod();
};
```

this:

```
C++

void CMyClass::mymethod() { return; }
```

is equivalent to this:

```
C++

void __fastcall CMyClass::mymethod() { return; }
```

For compatibility with previous versions, `_fastcall` is a synonym for `__fastcall` unless compiler option /Za (Disable language extensions) is specified.

# Example

In the following example, the function `DeleteAggrWrapper` is passed arguments in registers:

```
C++

// Example of the __fastcall keyword
#define FASTCALL    __fastcall

void FASTCALL DeleteAggrWrapper(void* pWrapper);
// Example of the __ fastcall keyword on function pointer
typedef BOOL (__fastcall *funcname_ptr)(void * arg1, const char * arg2, DWORD flags, ...);
```

**END Microsoft Specific**

# See also

Argument Passing and Naming Conventions
Keywords

# Feedback

**Was this page helpful?** Yes No

Provide product feedback | Get help at Microsoft Q&A