__stdcall

Article • 08/03/2021

The `__stdcall` calling convention is used to call Win32 API functions. The callee cleans the stack, so the compiler makes `vararg` functions `__cdecl`. Functions that use this calling convention require a function prototype. The `__stdcall` modifier is Microsoft-specific.

Syntax

> *return-type* `__stdcall` *function-name*[`(` *argument-list* `)`]

Remarks

The following list shows the implementation of this calling convention.

Expand table

Element	Implementation
Argument-passing order	Right to left.
Argument-passing convention	By value, unless a pointer or reference type is passed.
Stack-maintenance responsibility	Called function pops its own arguments from the stack.
Name-decoration convention	An underscore (`_`) is prefixed to the name. The name is followed by the at sign (`@`) followed by the number of bytes (in decimal) in the argument list. Therefore, the function declared as `int func( int a, double b )` is decorated as follows: `_func@12`
Case-translation convention	None

The /Gz compiler option specifies `__stdcall` for all functions not explicitly declared with a different calling convention.

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

Functions declared using the `__stdcall` modifier return values the same way as functions declared using __cdecl.

On ARM and x64 processors, `__stdcall` is accepted and ignored by the compiler; on ARM and x64 architectures, by convention, arguments are passed in registers when possible, and subsequent arguments are passed on the stack.

For non-static class functions, if the function is defined out-of-line, the calling convention modifier does not 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 __stdcall mymethod();
};
```

this

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

is equivalent to this

```C++
void __stdcall CMyClass::mymethod() { return; }
```

Example

In the following example, use of **`__stdcall`** results in all `WINAPI` function types being handled as a standard call:

```C++
// Example of the __stdcall keyword
#define WINAPI __stdcall
// Example of the __stdcall keyword on function pointer
typedef BOOL (__stdcall *funcname_ptr)(void * arg1, const char * arg2, DWORD flags, ...);
```

See also

Argument Passing and Naming Conventions
Keywords

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A