__cdecl

Article • 08/03/2021

`__cdecl` is the default calling convention for C and C++ programs. Because the stack is cleaned up by the caller, it can do `vararg` functions. The `__cdecl` calling convention creates larger executables than __stdcall, because it requires each function call to include stack cleanup code. The following list shows the implementation of this calling convention. The `__cdecl` modifier is Microsoft-specific.

Expand table

Element	Implementation
Argument-passing order	Right to left.
Stack-maintenance responsibility	Calling function pops the arguments from the stack.
Name-decoration convention	Underscore character (_) is prefixed to names, except when __cdecl functions that use C linkage are exported.
Case-translation convention	No case translation performed.

> **Note**
>
> For related information, see Decorated Names.

Place the `__cdecl` modifier before a variable or a function name. Because the C naming and calling conventions are the default, the only time you must use `__cdecl` in x86 code is when you have specified the `/Gv` (vectorcall), `/Gz` (stdcall), or `/Gr` (fastcall) compiler option. The /Gd compiler option forces the `__cdecl` calling convention.

On ARM and x64 processors, `__cdecl` is accepted but typically ignored by the compiler. By convention on ARM and x64, arguments are passed in registers when possible, and subsequent arguments are passed on the stack. In x64 code, use `__cdecl` to override the **/Gv** compiler option and use the default x64 calling convention.

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 __cdecl mymethod();
};
```

this:

```
C++

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

is equivalent to this:

```
C++

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

For compatibility with previous versions, **cdecl** and **_cdecl** are a synonym for `__cdecl` unless compiler option /Za (Disable language extensions) is specified.

Example

In the following example, the compiler is instructed to use C naming and calling conventions for the `system` function.

```
C++

// Example of the __cdecl keyword on function
int __cdecl system(const char *);
// Example of the __cdecl keyword on function pointer
typedef BOOL (__cdecl *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