External C functions are allowed to be variadic. However, a variadic function takes a minimum number of arguments. For example, consider C's variadic `printf` function:

```
use std::os::raw::{c_char, c_int};

extern "C" {
    fn printf(_: *const c_char, ...) -> c_int;
}

unsafe { printf(); } // error!
```

Using this declaration, it must be called with at least one argument, so simply calling `printf()` is invalid. But the following uses are allowed:

```
# #![feature(static_nobundle)]
# use std::os::raw::{c_char, c_int};
# #[cfg_attr(all(windows, target_env = "msvc"),
#            link(name = "legacy_stdio_definitions", kind = "static-nobundle"))]
# extern "C" { fn printf(_: *const c_char, ...) -> c_int; }
# fn main() {
unsafe {
    use std::ffi::CString;

    let fmt = CString::new("test\n").unwrap();
    printf(fmt.as_ptr());

    let fmt = CString::new("number = %d\n").unwrap();
    printf(fmt.as_ptr(), 3);

    let fmt = CString::new("%d, %d\n").unwrap();
    printf(fmt.as_ptr(), 10, 5);
}
# }
```