Don’t pass lambdas (or other multi-line entities) as parameters to macros

devblogs.microsoft.com/oldnewthing/20190115-00

January 15, 2019

Raymond Chen

Consider this macro:

```
#ifdef DEBUG
#define LOG(value) LogValue(value)
#else
// In production, evaluate but don't log.
#define LOG(value) (value)
#endif
```

This seems not entirely unreasonable, but bad things happen if you pass a multi-line entity as the macro parameter.

```
// Suppose this is line 12
LOG(zap_if([&](auto&& item)
{
 ... decide whether to zap this item ...
});
```

You will never to be able to debug this lambda, because the preprocessor slurps up the entire macro parameter list, and then spits the text back out when the macro is expanded.

But when it spits the text out, *it spits it all into one giant line of code*. The result is as if you had written

```
LogValue(zap_if([&](auto&& item) { ... decide whether to zap this item ... });
```

Therefore, all compiler errors in the `... decide whether to zap this item ...` are reported on line 12. Your IDE will just send you to that line and tell you, “Good luck!”

But wait, the pain doesn’t stop there.

Even it you manage to fix the compiler errors in the code, you won’t be able to debug it. You’ll put the cursor inside the lambda on line 14 and tell the debugger, “Set a breakpoint here,” and the debugger will say, “There is no code here for me to set a breakpoint on.” Because there really is no code there. All the code you wrote on line 14 got mashed into the huge line of regurgitated text back on line 12.

To make your life sane, don’t pass lambdas or other multi-line entities as macro parameters. Break the lambda out into a place where it won’t be part of a macro.

```
auto result = zap_if([&](auto&& item)
{
 ... decide whether to zap this item ...
});

LOG(result);
```

Bonus chatter: The C++ Core Guidelines recommends

> Scream when you see a macro that isn’t just used for source control (e.g., `#ifdef`)

On the other hand, you may be forced into using macros that you didn’t write, such as the `SUCCEEDED` macro provided by COM.

Raymond Chen

Follow