Why do I get errors about some weird symbol called ?main@@YAHP$01E$AAV?$Array@PE$AAVString@Platform…, part 2

devblogs.microsoft.com/oldnewthing/20250626-00/?p=111314

June 26, 2025

We are investigating why a project is getting an error about a weird C++/CX symbol, and we thought we had figured it out, but our attempt to replicate the problem with a minimal example failed. So we must have removed something important from the example.

Since the problem occurred when the project involved C++/CX, let's add C++/CX to our minimal example. Maybe that will tell us something.

```
rem create a minimal fuzzer library
>lib.cpp echo void fuzzme(); int __cdecl main(int, char**) { fuzzme(); return 42;
}
cl /c lib.cpp
lib /out:lib.lib lib.obj

rem create our fuzzer plugin
>fuzzer.cpp echo void fuzzme() {}
cl /c fuzzer.cpp

rem new! Add a superfluous C++/CX component
>cx.cpp echo ref class Dummy {};
cl /c /EHsc /ZW cx.cpp

rem Try to link them all together
link /out:fuzzer.exe /subsystem:console fuzzer.obj cx.obj lib.lib
```

Output:

```
vccorlib.lib(climain.obj) : error LNK2019: unresolved external symbol "?
main@@YAHP$01E$AAV?$Array@PE$AAVString@Platform@@$00@Platform@@@Z" (?
main@@YAHP$01E$AAV?$Array@PE$AAVString@Platform@@$00@Platform@@@Z) referenced in
function "int __cdecl _main(void)" (?_main@@YAHXZ)
```

Okay, *now* we get the error.

So the presence of `cx.obj` introduces the problem.

Let's go back to the verbose log to see where `cx.obj` enters the picture.

Actually, something interesting jumps out right at the start.

```
Starting pass 1
Processed /DEFAULTLIB:LIBCMT
Processed /DEFAULTLIB:OLDNAMES
Processed /DEFAULTLIB:vccorlib.lib
Processed /DEFAULTLIB:MSVCRT
```

These two libraries got added as default libraries, and that's how `vccorlib.lib` became one of the libraries participating in the module.

If we dig into `cx.obj`, we can see where it requests those libraries.

```
link /dump /all cx.obj | findstr /i defaultlib
```

Output:

```
   /DEFAULTLIB:vccorlib.lib
   /DEFAULTLIB:MSVCRT
   /DEFAULTLIB:OLDNAMES
```

The compiler injects requests for three default libraries into `cx.obj`, so that's how `vccorlib.lib` joins the set of default libraries.

This explains why `cx.obj` is essential to the repro: It is `cx.obj` that pulls in `vccorlib.lib`, which means that a search for `main` finds the version in `vccorlib.lib` before it finds the one we want in `lib.lib`.

Now that we understand the source of the problem, we'll look at trying to fix it. Next time.