

Re: Pointers, DLL Header files and lots of exceptions?

Source: <http://coding.derkeiler.com/Archive/Delphi/comp.lang.pascal.delphi.misc/2004-08/1549.html>

From: Rob Kennedy (*me3_at_privacy.net*)

Date: 08/25/04

Date: Wed, 25 Aug 2004 10:38:24 -0500

Kevin Oleniczak wrote:

```
> DECLSPEC_CLASS int get_values(HANDLE hCom, int* error_count, int*
> total_count);
```

You need to look through the header file (and everything it #includes, recursively) to figure out what DECLSPEC_CLASS means.

> Here's a snippet of the Delphi unit that wraps it:

```
>
> var get_values: function (hCom: THandle; var error_count: integer; var
> total_count: integer): integer cdecl {$IFDEF WIN32} stdcall {$ENDIF};
```

When choosing a calling convention, Delphi uses the last one specified. In a Win32 program, cdecl is mentioned before stdcall, so Delphi uses stdcall. That's only correct if the DECLSPEC_CLASS macro resolves to stdcall, though.

```
> .... this works and shows the correct values. BUT after I click ok on the
> message dialog... The click procedure ends and throws a similar
> EAccessViolation as before, but at a different address. !?!?
```

Sounds like a problem of calling conventions.

> So my questions are:

```
> 1) is the c data type "int*", really convert to "var someparam integer"? The
> option I would have thought would work would be a "pInteger".
```

Then go ahead and use that. Whether that's important depends on what the function really uses those values for. A var parameter in Delphi is passed as a pointer, so on the surface, they are the same things.

```
> 2) If the function really is working with pointers, then shouldn't my button
> code use a pointer type to pass to the function?
```

The answer to your question is yes, but since the function is *not* really working with pointers — the parameters are declared var — your

comp.lang.pascal.delphi.misc: Re: Pointers, DLL Header files and lots of exceptions?

button code should not be using them either.

- > 3) *why does memo1.lines.append throw an exception? Something even stranger*
- > *is that i can even change the showmessage call to this...*
- >
- > *showmessage('Hello world with no reference to the values returned by the*
- > *function!');*
- >
- > *this will throw an exception on "end;" as well but at address 00000000.*

When you change the code in a procedure, the compiler changes how it uses the stack, including how it allocates hidden, temporary variables. An incorrect calling convention on a function will cause the stack to get corrupted. Since the stack will get used different ways, that corruption will manifest itself in different ways, too.

--
Rob