

Dark corners of C/C++: The typedef keyword doesn't need to be the first word on the line

 devblogs.microsoft.com/oldnewthing/20130424-00

April 24, 2013



Raymond Chen

Here are some strange but legal declarations in C/C++:

```
int typedef a;  
short unsigned typedef b;
```

By convention, the `typedef` keyword comes at the beginning of the line, but this is not actually required by the language. The above declarations are equivalent to

```
typedef int a;  
typedef short unsigned b;
```

The C language (but not C++) also permits you to say `typedef` without actually defining a type!

```
typedef enum { c }; // legal in C, not C++
```

In the above case, the `typedef` is ignored, and it's the same as just declaring the `enum` the plain boring way.

```
enum { c };
```

Other weird things you can do with `typedef` in C:

```
typedef;  
typedef int;  
typedef int short;
```

None of the above statements do anything, but they are technically legal in pre-C89 versions of the C language. They are just alternate manifestations of the quirk in the grammar that permits you to say `typedef` without actually defining a type. (In C89, this loophole was closed: Clause 6.7 Constraint 2 requires that “A declaration shall declare at least a declarator, a tag, or the members of an enumeration.”)

That last example of `typedef int short;` is particularly misleading, since at first glance it sounds like it's redefining the `short` data type. But then you realize that `int short` and `short int` are equivalent, and this is just an empty declaration of the `short int` data type. It doesn't actually widen your shorts. If you need to widen your shorts, go see a tailor.¹

Note that just because it's legal doesn't mean it's recommended. You should probably stick to using `typedef` the way most people use it, unless you're looking to enter the IOCCC.

¹ The primary purpose of this article was to tell that one stupid joke. And it's not even my joke!

Raymond Chen

Follow

