%D OR %P WHEN DISPLAYING POINTER VALUES?

Some students have noticed that I occasionally use the decimal `%d` format specifier when displaying pointer values in strings, like this:

```
printf("%d", ptr);
```

Whereas it is usually the convention to use the `%p` format specifier, like this:

```
printf("%p", ptr);
```

So which should you use - `%p` or `%d`?

The simple answer is that `%p` is considered to be the standard format specifier for pointers and therefore, unless you have some good reason for choosing `%d`, you should use `%p`.

In the course, however, I *do* have a good reason for (sometimes) using `%d`. Let me explain. A pointer value is a number. It is a number that represents an address. That number can be displayed in any numeric format. The `%p` specifier displays it in hexadecimal format. Now, some experienced C programmers are so familiar with hexadecimal that they can count in hex numbers as easily as in decimal numbers. So they are happy to display values like this (here `%x`, like `%p` displays as hex):

```
printf( "%x, %x, %x", 105, 110, 115);
```

This shows these values:

```
69, 6e, 73
```

In fact, you could even use the `%p` specifier in this case:

```
printf( "%p, %p, %p", 105, 110, 115);
```

That would display the same hex values though formatted differently:

```
00000069, 0000006E, 00000073
```

Now, if you are sufficiently comfortable with hexadecimal to understand that 00000069 is the same value as 105, there is no problem at all here. Just show values in hex. However many (*most*!) people cannot count easily in hexadecimal. But in many of my lessons we need to be able to count pointer values. This is especially true later in the course when I explain data-type alignment and pointer arithmetic.

For example, look at this program:

```
#include <stdio.h>

int main(int argc, char **argv)
{
    int *p;
    int a[] = {100,200,300,400};
     p = a;
     for (int i = 0; i < 4; i++){
         printf("Value = %d, Address = %p\n", *p, p);
         p = p + 1;
     }
}
```

This iterates over an array of integers (if you don't understand this at present, don't worry – it is explained in the section on pointer arithmetic later in this course). At each iteration, the value of the pointer is incremented by 4 bytes (the size of an integer). This is the output in hexadecimal (using the `%p` specifier):

```
Value = 100, Address = 0028FF08
Value = 200, Address = 0028FF0C
Value = 300, Address = 0028FF10
Value = 400, Address = 0028FF14
```

For those people who aren't fluent in hexadecimal, the difference between 0028FF00 and 0028FF0C may not be immediately obvious. But if I change the code to use the `%d` specifier, like this:

```
printf("Value = %d, Address = %d\n", *p, p);
```

Now, this is the output:

```
Value = 100, Address = 2686728
Value = 200, Address = 2686732
Value = 300, Address = 2686736
Value = 400, Address = 2686740
```

This time, anyone can easily see that each address has been incremented by 4. Just look at the last two digits in each case: 28, 32, 36, 40.

So, in short: *in most cases*, when you need to display a pointer value in a formatted string, `%p` is the standard specifier to use. But as `%p` shows hexadecimal values it may, in some cases, be easier to understand pointer values if shown in decimal format. In this course, I occasionally use `%d` instead of `%p` for the sake of clarity. This subject is also discussed in the course FAQ.