Contents

1 Printing variables & conversion specifications

As seen in the hello world program, `printf` can be used to output text to the screen. It can also be used to print the values of variables using formatted output (which is what the f in `printf` stands for!). Here are some examples:

```
// This will print 4
printf("%i\n", 4);

int number = 7;

printf("The value of number is %i\n", number);

unsigned int other_number = 55;
float dog_weight = 3.221;

printf("The other number is %u. Dog weighs %f pounds.\n", other_number, dog_weight);
```

As you may have gathered, the `%` character in the first string argument of `printf` (also known as the **format string**) has a special use. `%` is used to denote a **conversion specification**. This is a location in the string sent to `printf` that will be substituted by a value of some kind when the output is finalized. The character(s) that come after `%` indicate what type of value will be printed.

Here is a table matching all conversion specification to the various data types:

Characters after %	Data type
c	char
i or d	int
li or ld	long int
u	unsigned int
f or F	float
lf or lF	double
p	void *
%	none, this inserts a literal % character

See `https://en.cppreference.com/w/c/io/fprintf` for a full conversion specification reference.

To match an argument value to a conversion specification, the order of arguments is used. The second argument of `printf` is matched with the first conversion specification in the format string. The third argument of `printf` is matched with the second conversion specification in the format string, and so on.