2. Variables

2.1 Intro to variables

Variables are what the name suggests: bits of memory stored in RAM (random access memory) that can vary and can be modified. There are many type of variables.

- **Integers**: these are counting numbers that go into positives and negative.
- **Characters**: these are ASCII characters that you can type out on a keyboard.
- **Floats**: these are 'floating decimal numbers', which represents a number that can have **decimal points**.

To create a variable, put in **the type** followed by **the name**. The convention for the name is **camelCase** – words are stringed together and all except the first word start with a capital.

```
int numberOfChickens;
char myFirstNameInitial;
float pi;
```

When created, you must assign it a value. Note you can both create and assign it a value – see pi.

```
int numberOfChickens;
numberOfChickens = 20;
float pi = 3.14159265;
```

2.2 Printing variables

To print out variables, we need to append it into the printf statement.

```
#include <stdio.h>

int main () {
	int numberOfChickens = 20;
	printf("I have %d chickens.\n", numberOfChickens);
	return 0;
}
```

```
gcc chickens.c -Wall -o chickens
./chickens
```

Note the syntax.

- **numberOfChickens** and **the comma** are **not** within the quotation marks.

2.3 Errors

> The worst errors are those that don't show up as errors.

When you code, you are bound to get errors. Errors you receive from the compiler is **like a seatbelt** – they help you a lot but the worst errors you get are those that don't show up as errors.

Let's try compile this code:

```
#include <stdio.h>

int main () {
      int numberOfChickens = 20;
      printf("I have %d chickens.\n, numberOfChickens");
}
```

```
gcc chickens_error.c -Wall -o chickens_error.c
```

In the console, you should get something like this:

```
chickens.c: In function 'main':
chickens.c:5:2: warning: format '%d' expects a matching 'int' argument [-
Wformat=]
  printf("I have %d chickens.\n, numberOfChickens");
  ^
...
```

- chickens.c:5:2 tells you the error is on line 5
- It says that **%d expects a matching 'int' argument** – meaning that it can't find an integer to replace the %d with.

You may have also gotten this error:

```
chickens.c:6:1: warning: control reaches end of non-void function [-
Wreturn-type]
 }
 ^
```

- What line is the error on?
- Are we missing something at the end of our **main** function?

If we leave out -Wall, you'll notice **it compiles**! This is because -Wall shows all warnings and errors before it compiles.

2.4 Integers round down

Integers cannot have decimal points. If you do, **the compiler will always round down for you**.

```
#include <stdio.h>

int main () {
	int numberOfChickens = 3.5;
	printf("I have %d chickens.\n", numberOfChickens);
	return 0;
}
```

```
gcc chickens.c -Wall -o chickens
./chickens
I have 3 chickens.
```

If you want to use decimal points, **make sure you use a float**, and instead of %d, you can use **%f**.

```
#include <stdio.h>

int main () {
	float heightOfBuilding = 3.513;
	printf("The building is %f metres high.\n", heightOfBuilding);
	return 0;
}
```

```
gcc building.c -Wall -o building
./building
The building is 3.513000 metres high.
```

2.5 Ariane the Rocket

In 1996, the European Space Agency sent a rocket named Ariane. It cost $7 billion dollars, and it exploded on its first voyage. They used C to code their trajectory. It turned out that a number that had decimal points was stored into an integer, and the C compiler simply said "Hah, don't worry, we're good!"

So, the lesson is – make sure you know what is going into your variables! Integers will always round down.