C++/Arduino cheat sheet

Variables

In many high level languages you don’t explicitly define the type of you variables. For example in javascript you might write

```
var x = 1;
```

Or in python just

```
x = 1
```

In C++ variables have definite types. This is important when introducing the variable for the first time, which is also known as declaring. In the following we’ll shortly introduce all the types you need to know in this workshop.

After declaring the variables you can use the variables “as usual”.

For example

```
int i = 1;
i = i + 3;
```

Would result in i being 4.

int

int is short for integer, an even number. For example you might write

```
int x = 0;
```

float

Float is short for floating point, and technicalities aside is just a type for a decimal number.

```
float pi = 3.14;
```

bool

Bool, or boolean, contains a true/false value.

```
bool isThisHard = false;
```

void

You don’t really need to know about void, but it can been seen in every project. The Arduino setup and loop functions return a “void” type variable - which means they return nothing.

String

String isn’t an actual C++ type, but a class of the Arduino development environment. This is nice, because C++ strings are pretty much a pain. You can declare a String just by writing

```
String hello = “World”;
```

You can combine Strings with the + operator

```
String hello = “Hello”;
String world = “ world!”;
String helloWorld = hello + world;
```

Sometimes the code interpreter (aka compiler) might confuse a string as a C++ string. This is likely if you try to combine several strings which are not explicitly Strings.

For example

```
String hello = “Hello” + “ world!”;
```

Is asking for trouble. Instead you can use the String constructor method

```
String hello = String(“Hello”) + String(“ world!”);
```

Classes and objects

You might encounter some classes. If you need to make a variable out of a class you just use the class name as the type.

```
WifiClient newClient = getNewClient();
```

#define

#Define isn’t exactly a variable. However, in Arduino applications it is often used as a way to define some settings that are constant in the program. In addition to some technical benefits it also makes the code easier to read as well as eases modifications.

In the following example we define the keyword redLed to be 2 and later use it to turn the led on.

```
#define redLed 2
.
.
.
digitalWrite(redLed, HIGH);
```

If we ever decide to switch the led to a different GPIO pin it would be really easy to just change the definition.

Brackets and conditions

If, else, else if, loops etc. use brackets. For example

```
if (a == b) {
  goHome();
}
else if (b == c) {
  doSomethingElse();
}
else {
  somethingDifferent();
}
```

To combine conditions use && for ‘and’, ll for ‘or’ and ! for negation. For example

```
if ((a && b) ll (c != a)) {
  something();
}
```

The semicolon

Lines end in semicolons. Just remember this and your code will compile.

Printing to console

What about console.log or print? We’ll since your code isn’t running on your PC you have to debug through the serial port (through the USB cable).

This has been made quite easy. Refer to https://www.arduino.cc/en/Serial/Print

C++ dynamic memory allocation

You don’t know what this is? Good. If you do, don’t use it. It’s not worth it.