```
    // == => complete equality check for value and type
    // != => complete inequality check for value and type
    // > < >= <= are also comparsion operators
```

Control structures

```
// ---------- CONTROL STRUCTURE ----------

// CONDITIONALS

// IF ELSE IF ELSE

int x = 10;
if (x < 10) {
    writeln("x is smaller than 10");
} else if (x > 10) {
    writeln("x is bigger than 10");
} else {
    writeln("x is equals to 10");
}

// SWITCH CASE DEFAULT
    // break => remember to include break below each case, otherwise logic will flow through after each case statement
    // default => represents the default case

int y = 5;
switch (y) {
    case 1:
        println("y is equals to 1");
        break;
    case 2:
        println("y is equals to 2");
        break;
    case 3:
        println("y is equals to 3");
        break;
    case 4:
        println("y is equals to 4");
        break;
    case 5:
        println("y is equals to 5");
        break
    default:
        println("welcome to the default case where y isn't 1 to 5");
}

// LOOPS

// FOR LOOPS

for (int i=0; i<1000; i++) { // allows creation of basic for loops
    writeln(i);
}

// FOREACH LOOPS
    // .. => creates a continuous range that is first-value inclusive and last-value exclusive
    // foreach => allows for iteration over a created loop or iterable data structure
    // foreach_reverse => allows for iteration over a created loop or iterable data structure in reverse order

foreach (n;1..1000) {
    if (n % 2 == 0) {
        writeln(n);
    }
}

foreach_reverse (n;1..1000) {
    if (n % 2 == 1) {
        writeln(n); // odd
    } else {
        writeln("even!");
    }
}

// WHILE LOOPS

int n = 1;
while (n < 10000) {
    n += n;
}

// DO WHILE LOOPS

do {
    n -= (n/2);
} while (n>0);
```