Introduction

Basic Structure

```
#include<stdio.h>
int main()
{
	printf("Hello World!\n");
}
```

- `#include<stdio.h>`
  Header importing: contains code necessary to run programs. In this case, `stdio.h` contains basic input and output functions.

- `int main()`
  Every C program (and many other languages too) starts execution from the `main()` function. `int` specifies the return type of the function.

- `{}` (curly braces)
  Curly braces indicate the start and end of a block of code. In this case, they indicate the start and end of the `main()` function.

- `printf("Hello World\n");`
  This is an example of a statement. Every statement in C should end with a semicolon (;). In this case, `printf()` prints something on the screen.

Important Header files

- `stdio.h`
  Contains functions for basic input and output. stdio is short for standard input and output

- `stdlib.h`
  Contains general utilities for memory management, algorithms, random generator, program utilities etc.

- `limits.h`
  Contains constants that define the upper and lower limits of datatypes.

- `float.h`
  Contains limits and constants related to the float datatype.

- `math.h`
  Contains many mathematical functions and constants used in math operations.

Comments

Single-line comment

```
	// This is a comment
```

Multi-line comment

```
	/*
	This is a comment
	*/
```

Documentation comment

```
	/**
	* This is a doc comment
	*/
```

The `printf()` statement

- Used to print stuff on the output screen (also called the Terminal).

Syntax:

```
 printf(string);
```

`string`: just a normal string

```
 printf(formatString, variables);
```

`formatString`: a string with the usual characters and format specifiers

`variables`: variables to be printed

Format Specifiers

- %d -> used for integers (decimal - base 2)
- %o -> used for octal numbers (base 8)
- %x -> used for hexadecimal numbers (base 16)
- %f -> used for floating point numbers

Commonly used terminal commands

Compile a program

```
	gcc filename.c
```

Compile a program and generate an output file

```
	gcc filename.c -o outputFile
```

Run program

```
	outputFile.exe
```

Create assembly code out of program

```
	gcc -s filename.c
```

Show all warnings in program

```
	gcc -Wall filename.c
```