*** Include ***

Include is a C preprocessor directive. I serves to incorporate the contents of another file into the current one. Usually, it is used to include header files from the C standard library or user defined ones. Include is usually placed at the start of a C file, before any of the other code. Included files can either be in the same directory as the file that includes them or can be added to the preprocessor using different flags depending on the compiler.

The syntax of include is

```
#include <(C standard library file)>
#include "(user defined file)"
```

Do you want to see examples (type y for yes)?
y

*** Include Examples ***

```
#include <stdio.h>
```

This instance of include serves to add all the functions located in the standard library stdio.h file to the current source code. It will allow the code to use functions such as printf, and types such as FILE.

```
#include "myheader.h"
```

In this instance, include takes all the contents from myheader.h, which is completely user defined.

```
#include "module1.h"
```

Here, include is used to incorporate all the contents from a user defined module. This module can be part of a bigger project, where

code separation and modularity become essential. Include allows developers to make their code as modular as they want.
*** Main Paramters ***

Main functions in C can accept two different parameters. These are argc and argv. These are meant to be used for command-line arguments that are passed to the program when it is executed. The integer argc represents the amount of arguments (including the executable itself), while argv is an array of strings.
Again, the name of the executable is always included, so argc will always be at least one, and argv at index 0 will be the executable's name.

The syntax of main parameters is
```
int main(int argc, char *argv[]) {}
```

Do you want to see examples (type y for yes)?
y

*** Main Parameters Examples ***

```
./test first second third
```
This is an example of executing a C program from the command-line (bash), while passing three arguments. In this case, argc would be equal to 4, while argv[0] would be ./test, argv[1] would be first, argv[2] would be second, and so on.

```
printf("%s\n", argv[0]);
```
This code is printing the first argument in the argv parameter. This is going to be the executable name, so if the file is test.c and the executable is test.exe, this code will print ./test.exe.

```
    printf("%d, %s\n", argc, argv[argc - 1]);
```
In this code, n and the nth argument are being printed, where n is the amount of arguments (argc). Looking at the previous examples, the output would be '4, third'.

```
    for(int i = 0; i < argc; i++) {
      printf("%d, %s\n", i, argv[i]);
    }
```
Here, each index and equivalent argument is being printed. Looking at the previous examples, the output would be '0, ./test', then '1, first', then '2, second', and then '3, third'.