

# Your First Program

Objectives:

Having read this section you should have an understanding of:

- 1.a preprocessor directive that must be present in all your C programs.
- 2.a simple C function used to write information to your screen.
- 3.how to add comments to your programs

Now that you've seen the compiler in action it's time for you to write your very own first C program. You can probably guess what it's going to be - the program that everyone writes just to check they understand the very, very, very basics of what is going on in a new language.

Yes - it's the ubiquitous "Hello World" program. All your first program is going to do is print the message "Hello World" on the screen.

The program is a short one, to say the least. Here it is:

```
#include <stdio.h>
int main()
{
    printf("Hello World\n");
    return 0;
}
```

The first line is the standard start for all C programs - main(). After this comes the program's only instruction enclosed in curly brackets {}. The curly brackets mark the start and end of the list of instructions that make up the program - in this case just one instruction.

Notice the semicolon marking the end of the instruction. You might as well get into the habit of ending every C instruction with a semicolon - it will save you a lot of trouble! Also notice that the semicolon marks the end of an instruction - it isn't a separator as is the custom in other languages.

If you're puzzled about why the curly brackets are on separate lines I'd better tell you that it's just a layout convention to help you spot matching brackets. C is very unfussy about the way you lay it out. For example, you could enter the Hello World program as:

```
main(){printf("Hello World\n");}
```

but this is unusual.

The printf function does what its name suggest it does: it prints, on the screen, whatever you tell it to. The "\n" is a special symbols that forces a new line on the screen.

OK, that's enough explanation of our first program! Type it in and save it as Hello.c. Then use the compiler to compile it, then the linker to link it and finally run it. The output is as follows:

```
Hello World
```

Add Comments to a Program:

A comment is a note to yourself (or others) that you put into your source code. All comments are ignored by the compiler. They exist solely for your benefit. Comments are used primarily to document the meaning and purpose of your source code, so that you can remember later how it functions and how to use it. You can also use a comment to temporarily remove a line of code. Simply surround the line(s) with the comment symbols.

In C, the start of a comment is signaled by the /\* character pair. A comment is ended by \*/. For example, this is a syntactically correct C comment:

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

Comments can extend over several lines and can go anywhere except in the middle of any C keyword, function name or variable name. In C you can't have one comment within another

comment. That is comments may not be nested. Lets now look at our first program one last time but this time with comments:

```
main() /* main function heading */  
{  
    printf("\n Hello, World! \n"); /* Display message on */  
    /* the screen */  
}
```

This program is not large enough to warrant comment statements but the principle is still the same.