C PROGRAMMING: HELLO WORLD AND ADD!

Let's go through our "hello world" program and discuss its elements.

```
/* Print 'Hello, world' on the screen */
#include <stdio.h>
int main()
{
    printf("Hello, world\n");
    return 0;
}
```

The first line is a comment line. The compiler gcc ignores text between /* and */. Including comments when you write code is very useful, especially as programs get longer. The second line tells the compiler to include information about commands like "printf" which are part of "standard input and output". The line int main() together with the braces { and } tell the compiler where the main module of the program begins and ends. More complex codes may have additional pieces. The printf command is part of standard input and output and tells the computer to write what is within the quotation marks to the screen. (The slash-n tells the computer to go to the next line after writing.) Finally the return 0; is an instruction to return the value 0 when the program successfully executes. One can use these returned values from different pieces of a program (if more than one exists) to decide how to proceed further in a code's execution.

The following lines will appear in all the programs we will be writing.

```
#include <stdio.h>
#include <math.h>
int main()
{
    return 0;
}
```

Notice that we have included another header file, math.h. This file instructs the compiler to include information about what the primitive mathematical operations + - * / mean.

The next program reads in two numbers and adds them. Use gedit to type it in

```
/* This program reads in two numbers and adds them */
#include <stdio.h>
#include <math.h>
int main(void)
{
double x,y,z;
printf("\nEnter x");
printf("\n");
scanf("%lf",&x);
printf("\nEnter y");
printf("\n");
scanf("%lf",&y);
z=x+y;
printf("\nx+y=%12.8lf",z);
printf("\n");
return 0;
}
```

There are several new features to explain here:

`double x,y,z;`

You need to declare all variables used in a C program. Variables may be of different types, e.g. integer, float (a real number of 32 bit precision) or double (a real number of 64 bit precision). Unless you really have a very good reason, it is better always to use double instead of float.

`scanf("%lf",&x);`

scanf is the partner to printf. It instructs the computer to expect the user to enter (input) information and also where to put that information. In this case the lf (shorthand for 'long float') tells the computer to expect a double precision number, and the &x tells the computer to assign the number which is input to the variable x. A similar explanation applies to the line where a value for y is entered. scanf("%f",&x) would be used to read in a 32-bit float and scanf("%i",&x) would be used to read in an integer.

Finally in the printf command

```
printf("\nx+y=%lf",z);
```

the %lf tells the computer that it is to print out the value of the variable z. (Again the computer needs to be reminded what type of variable z is.) Later it will be useful to format the output of numbers. The following line

```
printf("\nx+y=%12.8lf",z);
```

tells the computer to use a total of 12 spaces when printing z, with 8 of them following the decimal point.

After using gedit to type this new program in, save it and compile it as you did with hello.c. For example, if you save the program wth the name "add.c" then you could compile:

```
gcc -o add.e add.c
```

After compiling, run your program

```
./add.e
```

and see that it works.

Closing Comments: Notice that most lines of C programs end in a semicolon. Forgetting the semicolon is one of the most common errors made in writing code. (Inserting a semicolon on those occasions when one is not needed is also a fairly common bug.) Another very common mistake is not to match the type of variable which is declared at the top of the program with the type designated in scanf. If you tell the computer that x is a double precision variable when you declare it initially but then use scanf("%i",&x) your program will do very weird things.