PHYS 2020

Basic C

An introduction to writing simple but useful programs in C

In these lectures I will take you through the basics of C, but you will need to read the C programming notes (on the PHYS2020 website) as it is not possible to cover everything in lectures.

All C programs must have…

```
main()
{

}
```

This program doesn't actually do anything, but it does compile, and so is a valid C program.
Most C programs have at least..

```c
#include <stdio.h>
/* the include statement includes various header files
(libraries) you may need. The libraries are actually bits of
code to perform functions such as printing to the keyboard
(stdio.h) or using maths functions (math.h) */

int main()
{
    printf("Here is some output.\n");
    return(0);
}
```

#include

Some Useful Header files:
- `#include <stdio.h>`
- `#include <stdlib.h>`
- `#include <math.h>`
- `#include <string.h>`

Note, `#include` statements do NOT end in a semicolon.

Note:
1. Including `math.h` is not enough to get you access to the
   maths libraries (such as sin, cos). You also need to link in the
   maths libraries when you compile, with the `-lm` (link maths)
   option of gcc. So, if you want to compile a file called `myfile.c`
   you would use the command `gcc -lm myfile.c`
The C programming Language consists of

• Keywords: 32 words that are reserved as commands etc. by the C language. See page 11 of the C programming notes. (e.g. if, int, char)

• Functions: Such as main(). Small routines that do specific tasks. There are hundreds available. See gcc.gnu.org to find out all the functions available.

• Operators: Such as =, +, -. See pp. 8/9 of the C programming notes.

Displaying Output to the Screen

printf("Print verbatim text by enclosing it in quotes");

printf("Get a newline like this\n");

printf("To get a quotation mark to print out on screen you need to use the \" escape\" character before the quotation mark.");
#include <stdio.h>
int main()
{
    char answer; // variable
    printf("Would you like the computer to tell your fortune?\n\n\n\n");
    printf("Please enter "y" in lowercase if the answer is yes.\n\n\n");
    printf("or enter "n" if the answer is no.\n\n\n");
    answer = getchar();
    if(answer == 'y')
    {
        printf("Tomorrow you will break your leg by tripping over a black cat.\n\n\n");
    }
    else if(answer == 'Y')
    {
        printf("Tomorrow you will break your leg by tripping over a black cat.\n\n\n");
    }
    else if(answer == 'n')
    {
        printf("Don't you believe that a computer knows the future?\n\n\n");
    }
    else if(answer == 'N')
    {
        printf("Don't you believe that a computer knows the future?\n\n\n");
    }
    else
    {
        printf("Look moron, I asked for "y\" or "n\". If you want your fortune told you will need to run the program again!!!\n\n\n");
    }
    return(0);
}

• Variables in C are of four basic data types:

  • int: integer
  • char: single ascii (text) character
  • float: real number
  • double: double precision whole number
  • (there are more than these, but this will get you started)

• You must declare variable that you want to use in your program. e.g.

  • int my_integer
  • char my_char
Functions

- There are five basic types of function:
  - void: doesn't return anything
  - int: returns integer
  - char: returns text
  - float: returns real number
  - double: returns double precision whole number
- so
  int main()
  {
    Return(0)
  } // returns an integer value to the operating system if it runs successfully.

Printing out variables

To use printf() to print variables we need to 1) use a placeholder for the variable in the string to be printed out, and 2) Give the name of variable at the end of the printf statement, after a comma.

So, to print an integer use
printf("print this integer %d \n", val_of_int);
where val_of_int is an integer variable. (You could also just use an integer - an "immediate" value, such as 12.)

For integers use %d
For real numbers use %f
For char variables use %c
For text strings use %s

scanf() reads from the keyboard in a similar way:
scanf("%s", &name);  reads a string variable typed at the keyboard into the variable name. It is important to use the ampersand in front of the variable. It tells the compiler that the input from the keyboard is the actual value to go into the variable name.
The ampersand (&) and variables

- When you declare a variable in C you use for example:
  - int my_variable

- This statement sets up a place in memory with an address and a value, called my_variable.

- The & is a unary operator which returns the address of the thing to its right.

- So scanf(%c, &my_variable); takes the word you read in at the keyboard and stores it at the address of my_variable.