



PRIFYSGOL  
**BANGOR**  
UNIVERSITY

# Imperative Programming in C

---

Laboratory 2: Input and Output

*Dr. Llyr ap Cenydd*

# Overview

In this lab we are going to be looking at creating and modifying variables of different data types, and simple input and output commands. This will include using `scanf` and `printf`, which allow us to take data in from the user via keyboard, and print formatted data out to the console.

Topics covered:

- Creating `int`, `float`, `char`, string variables
- Modifying variables, arithmetic
- Output formatted data to console using `printf`
- Input from keyboard using `scanf`
- Importing libraries using `#include<header file>`
- Using `#define` to create constants
- Using `#define` to create macros

For each lab, there will be link in the appendix to numerous resources you might find useful.

**You will also find a “Code Snippets” file on Blackboard (Labs->Extra Material) that contains examples of small programs as a reference.** Alternatively all the commands required to complete these exercises can be found in the lecture notes. The full C language specification, and detailed reference on all the main library headers, can also be found [here](#).

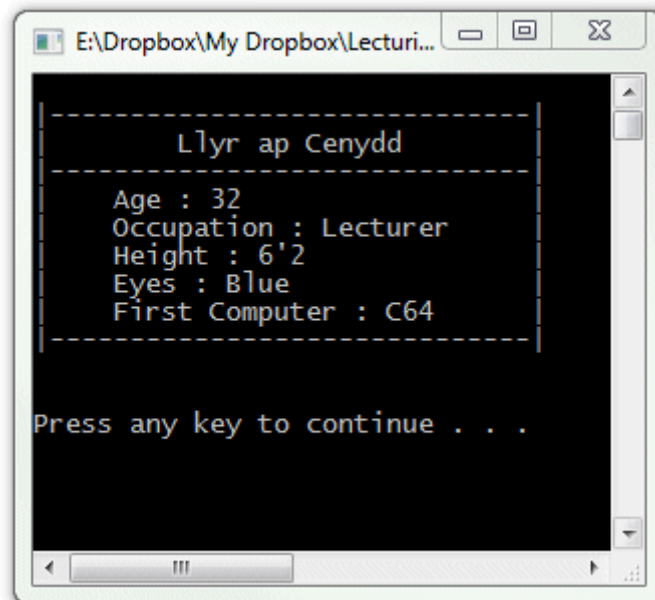
## Setup

Recall how we setup a project and source files in last week's lab. For this lab we advise you to follow exactly the same process.

Create a new folder called Lab 2, and then create a new project in Dev-C++ called Lab 2. The main function should be automatically created for you, in the file `main.c`. Then, for each Exercise create a new source file (e.g. `Lab2Exercise1.c`) which contains a function invoked from `main.c`. If you are unsure how to do any of these steps please ask a lab demonstrator.

## Exercise 1 - Say Hello

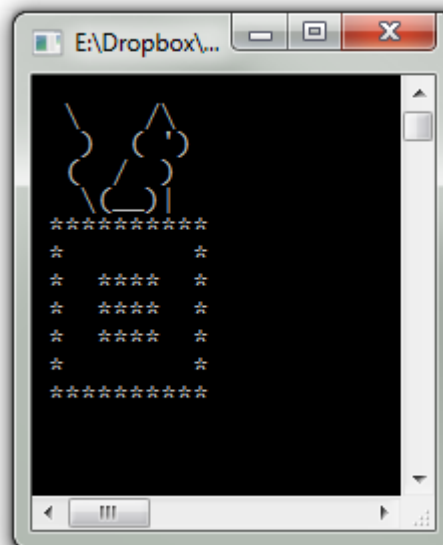
Write a program that uses printf to create a virtual business card that displays a minimum of 5 facts about yourself. Your output should appear something like this:



## Exercise 2 - Cat on a Box

For this exercise, write some code that reproduces the output below.

```
  \      /
   )    (
  (  /  )
   \__/
*****
*               *
*      *      *
*      *      *
*      *      *
*      *      *
*               *
*****
```



## Hints

- The `\n` character sequence can be used to skip to a new line
- To print a single backslash `\` we need to use the character sequence `\\`
- Try creating the box first!

## Exercise 3 – Input

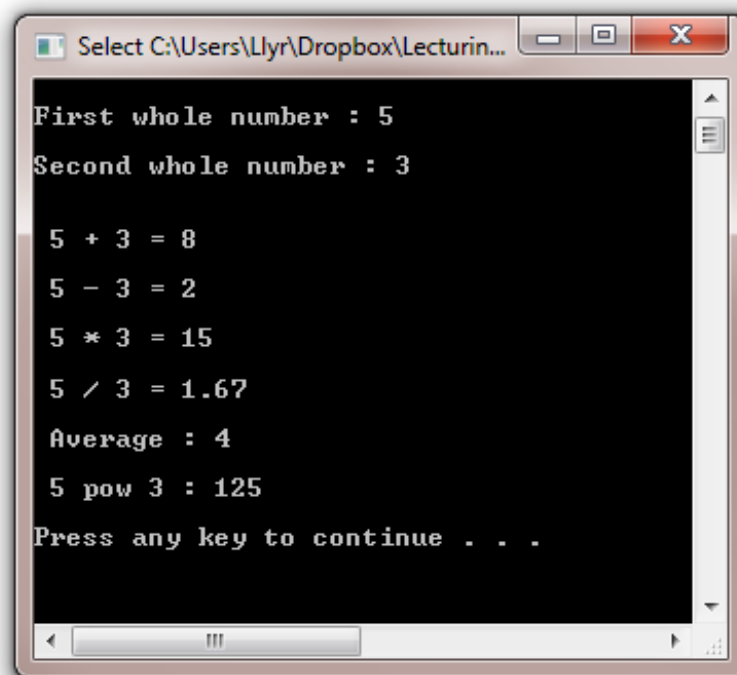
Scanf can be used to take input from the keyboard. For example, to store an integer from the user we can type:

```
int value;  
printf( "\nPlease enter a value : " );  
scanf( "%d" , &value);  
printf( "\nVariable value contains : %d" , value);
```

For this exercise write a program that asks the user for two numbers, adds them together, and then prints out the result.

## Exercise 4 – Simple Calculator

Create another program that asks the user for two whole numbers, and then calculates their addition, subtraction, multiplication, division, average and power (first value to the power of second). Your output should resemble the following:



## Hints

- Consider first writing a program that adds two numbers together, as in exercise 3
- Take a look at the [reference](#) for printf for information on how to format your output
- Be careful about data precision when dividing and finding the average
- The library Math.h contains a function to calculate powers - pow(x,y).

## Exercise 5 – Broken Circle

Create a new source file and copy->paste the code below.

The code is supposed to calculate the diameter, circumference and area of a circle given a radius, but it doesn't work! Your task for this exercise is to fix this program.

```
#include <math.h>
#define PI 3.14159265359

void myCircleCalculator()
{
    float radius;

    printf("\n-----");
    printf("\n ~~~ Circle Calculator ~~~ ");
    printf("\n-----");

    printf("\nEnter the Radius : ")

    scanf("%d", &radius);

    printf("\nDiameter is : %g", 2 * radius);
    printf("\nCircumference is : %.2f", 3 * PI * radius);
    printf("\nArea is : %.2f", PIE * pow(radius,radius));
}
```

## Hints

- Fixing compiler errors is a very important part of being a programmer
- Take a look at the compiler error window. It will almost always point to the problem
- There are six errors in the code
  - Some are syntax errors
  - Others are logical errors
- Try going line by line – just get it to compile first!
- Use the [internet](#) to find a circle calculator to test if the program is working properly

## Exercise 6 – Macro Calculator

The C pre-processor modifies a source code file before handing it over to the compiler. You will most likely use the pre-processor to include files directly into others, or to define constants like the constant number PI in a previous exercise. However the C pre-processor can also be used to create “inlined” code using macros which are expanded at compile time.

For example, at the top of our program we could define a macro called PRINT that will cause the pre-processor to insert the appropriate printf code:

```
#define PRINT(a) printf("%d\n", a);
```

Which we can then use by typing the following:

```
int myInteger = 10;  
PRINT(myInteger);
```

Other example macros can be seen in the appendix.

For this exercise, create macros for the calculator program you created in Exercise 4. You should define 6 macros – ADDITION, SUBTRACTION, MULTIPLICATION, DIVISION, AVERAGE and POWER that take two variables as input and perform the appropriate calculation inside the print statement.

## A note about scanf

If your program is asking for multiple inputs from the keyboard, you can run into a situation where pressing return at the end of a scanf will also trigger the next scanf to take that as input. The reason for this is pretty simple, let's say we are asking for two float values:

```
float myValue1;  
float myValue2;  
scanf("%f", &myValue1);  
scanf("%f", &myValue2);
```

When we press return after the first scanf what is passed into the input stream might be "5.223\n". The important part here is the "\n", i.e. the new line character created when we hit RETURN - because that can trigger the next scanf to take it as input. So myValue1 would contain "5.223", and myValue2 "\n".

The solution to this is very simple - place a getchar() after every scanf statement, like this:

```
scanf("%f", &myValue1); getchar();
```

This will ensure that the input stream is clear for the next scanf statement.

# Assessed Work – Chat Bot v0.1

Your task for this assessed exercise is to create a primitive [chat bot](#)!

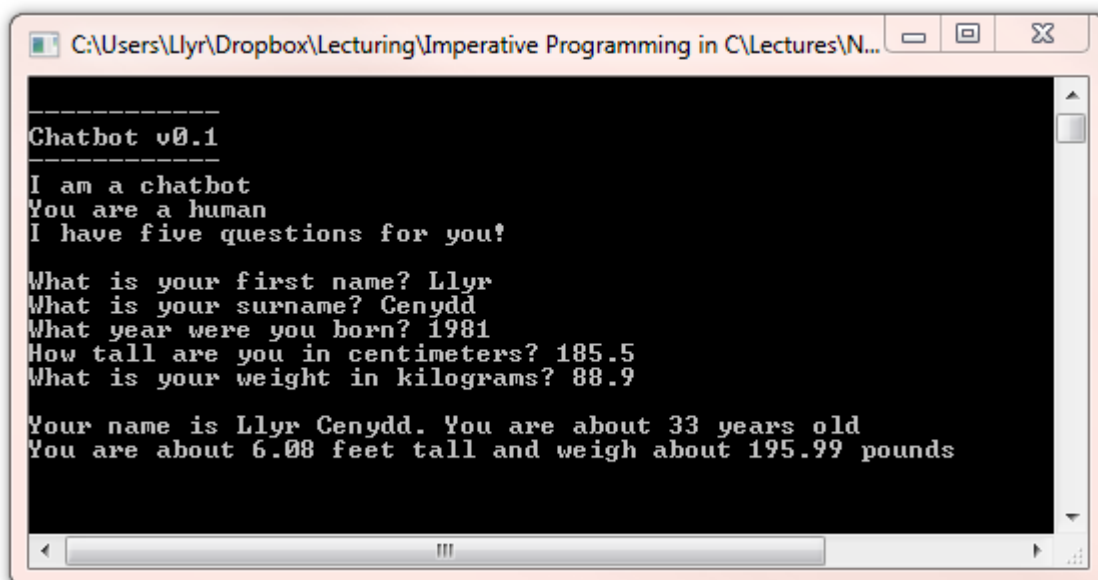
Write code that asks the user for five pieces of information. This information should be stored in separate variables and should be a mix of integers, strings and floating point numbers.

Once all the information has been read the chat bot should print out a paragraph showing it has knowledge of the five pieces of information, which should include **at least three** deductions or conversions\*. Feel free to use if-statements to personalize the message, although this is not part of the assessment.

\*Examples:

- If the user says they were born in 1994 the chatbot should reference the person's age as around 20.
- Convert user's weight from kilos to pounds
- Convert user's height from centimeters to feet
- Convert years to days, hours or minutes

Your output should be similar to the following screenshot:



```
-----
Chatbot v0.1
-----
I am a chatbot
You are a human
I have five questions for you!

What is your first name? Llyr
What is your surname? Cenydd
What year were you born? 1981
How tall are you in centimeters? 185.5
What is your weight in kilograms? 88.9

Your name is Llyr Cenydd. You are about 33 years old
You are about 6.08 feet tall and weigh about 195.99 pounds
```

## Detailed Specification

- Print a message welcoming the user
- Ask five questions and store the results
- Print out a paragraph containing at least three deductions or conversions

- Floats should be printed to two decimal places. If the float is a round number, it should be displayed without a decimal point i.e. show 4 instead of 4.00

As always when developing software test your program with a range of values. Check the results your program gives against some hand calculations. Do not simply assume that your program is giving the correct results. Failure to test properly will result in a marks penalty.

**Finally, make sure to read the submission notes.**

## Submission

Use **Blackboard** to submit your source code. **You only need to submit the .c file(s) that contains the code for the assessed work.**

In general, each source code file must:

- Contain a program header
- An appropriate level of comments
- Follow a consistent style of indentation
- Follow the usual C programming conventions

The deadline for submission will be published on Blackboard. Late submissions will be penalised in line with School policy.

Marks for this laboratory exercise are awarded for

- Managing input and output
- Meaningful variable names
- Program correctness (i.e. do you print the correct information)
- Program testing
- Layout and structure
- Conceptual understanding

When submitting work it is your responsibility to ensure that all work submitted is

- Consistent with stated requirements
- Entirely your own work
- Submitted through Blackboard on time

Please note that there are severe penalties for submitting work which is not your own. If you have used code which you have found on the Internet or from any other source then you must signal that fact with appropriate program comments.



Note also that to obtain a mark you must attend a laboratory session and be prepared to demonstrate your program and answer questions about the coding. Non-attendance at labs will result in your work not being marked.

## Appendix

[Online C Programming Resources](#)

[Complete C Reference Library](#)

## C Programming IDE's

[Dev-C++](#) (Windows)

[Code::Blocks](#) (Windows, Mac, Linux)

[Visual Studio/C++ Express](#) (Windows)

[Netbeans C/C++](#) (Windows, Mac, Linux)

[Codelite](#) (Windows, Mac, Linux)