Fundamentals of Computing

Week 1

Fundamentals of Computing

Module Leader:

Dr. Cristina Luca

Senior Lecturer

Course Leader MSc Computer Science

Cristina.Luca@anglia.ac.uk

Office: Com307

Office hours:

Mondays 11:00–13:00

Wednesdays 11:00–12:00

Fundamentals of Computing

- ✓ Canvas
- ✓ Assignment
- ✓ Reading List

Fundamentals of Computing

- ✓ Good Academic Practice (G.A.P.)

- ✓ Ruskin Card
  - Tap in at the start of every class;
  - The attendance system will be open for them to 'tap' in their attendance ***10 minutes*** before the start of the class and ***10 minutes*** after the class has started.
  - International / Tier 4 students are required to tap in **PLUS** sign weekly at the Faculty Help Desk / Office;

Fundamentals of Computing

Textbooks

McGrath, M, 2012, C Programming in Easy Steps, 4th edition, Leamington Spa. ISBN 978-1840785449

Software

Orwell Dev-C++ (free)

Assessment

Coursework

Fundamentals of Computing

Assessment Hand-in

- Every solution requires a description and thorough comments.
- All need to be printed and copied to a CD / USB Disk.
- A hard copy and the CD / USB Disk need to be handed in as one to the iCentre by **Friday 4th May 14:00**

Why C?

It is considered to be the most commonly used programming language.

The original C since the 70s, and C++ since the early 80s.

C is the most versatile high level language. C permits just about anything. Its code runs very fast.

Your first program

First program written using any language typically starts out with a simple Hello World program.

```
/* hello world program */
#include "stdio.h”

int main() {
      // print to screen
      printf("\nHello World\n");
      return 0; // success!
}
```

The result is that the characters “Hello World” are printed out, preceded by an empty line.

Your first program

A typical C program is made out of numerous lines of code AKA statements. They are executed line by line unless certain special keywords are utilised that might cause some statements to be skipped.

The statements are grouped in functions.

Your first program

A C program contains **functions** and **variables.** The functions specify the tasks to be performed by the program, while the variables store data relevant to the program.

They can be identified by the two brackets () that immediately follow their name.

Your first program

The **Hello World program** has one function called **main**. This function is the starting point of your program.

main functions are normally kept short and make use of other different functions to perform the necessary sub-tasks.

Your first program

All C programs must have a main function.

C is case sensitive. Using “Number” for a variable or “NUMBER” or “number” is **not the same**.

Your first program

C denotes the end of statement with a **semi-colon**.

```
printf("\nHello World\n");
```

Your first program

Curly brackets either "{" begin a group of statements, or "}" end a group of statements.

```
int main() {
  printf("\nHello World\n"); // print to screen
  return 0;                  // success!
}
```

If you also observed the two brackets (), then you have found a function! You may, as necessary, add extra information between them, but more on this later on.

Your first program

Every function has a return type. It is specified by the keyword just before the function name. It represents the type of value a function will output once it has finished running successfully. More on this later.

```
int main() {
      printf("\nHello World\n"); // print to screen
      return 0; // success!
}
```

main is expected to return a number AKA an integer indicating the success of it's operation. 0 is associated with an error free operation.

Your first program

The // or /* some comment here */ designates a comment.

The compiler ignores anything after two slashes // or between /* and */.

The comments make the program more readable for the programmers not for the compiler.

```
printf("\nHello World\n"); // print to screen

/* hello world program */
```

Your first program

#include tells the compiler to include a group of functions from the filename specified between the quotes "...", effectively copy-pasting the file contents in place of #include.

```
#include "stdio.h”
```

stdio.h contains basic functionality generally utilised by most C programs. It contains functions dealing with a range of input and output functionality.

Tasks 1 & 2

Task (1)

Write a program that prints your name.

Task (2)

Change the program above to print your address under your name.

Data types

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

int main(){
  int price = 2;               // declare a variable number
  double cost;                 /* declare a variable that
can store decimals */
  cost = 0.4 * price;          // do some math
  printf("\nPlease pay %f to the cashier!\n", cost);
  getch();                     // wait for any key press,
requires <conio.h> to be included
  return 0;                    // success!
}
```

This program has two variables price, and cost. These are the names of the variables. **Before a variable can be used it must be declared.**

Also, by default a C program closes down when the last statement has been executed. getch() can be used to effectively pause it until any key has been pressed before it goes away.

Data types

A type specifies the kind of data.

0.41 (double)   2 (integer)   ‘a’ (char)   “apple” (string)

Declaration - specify the type and the name of a variable

```
    int price;
```

Assignment - assign a value to a variable

```
    cost = 20.5;
```

Data is stored as bits in binary format (1s and 0s). 8 bits make up a byte, as in MegaByte, GigaByte, etc.

Each type is associated with a specified amount of bit-storage. e.g.: int = 32 bits = 4 bytes

Data types

Type	Meaning
int	integer variable AKA whole number -32,768 to 32,767 OR -2,147,483,648 to 2,147,483,647
short	short integer -32,768 to 32,767
long	long integer -2,147,483,648 to 2,147,483,647
float	single precision real (floating point) variable 1.2E-38 to 3.4E+38, 6 decimal places
double	double precision real (floating point) variable 2.3E-308 to 1.7E+308, 15 decimal places
char	character variable (single byte) -128 to 127 OR 0 to 255

Arithmetical operators

Operator	**Meaning**
()	brackets, group together a set of instructions
*	multiply
/	divide
%	modulus (returns the remainder of integer division)
+	add
-	subtract

Example

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

int main(){
    int a=5, b=3;   // declare two variables integer
    int c, d;       // declare two variables integer
    c = a + b;      // do some math
    d = a - b;      // do some math
    printf("\n%d = %d + %d\n", c, a, b);
    printf("\n%d = %d - %d\n", d, a, b);
    getch();
    return 0;       // success!
}
```

This program has four variables – four integers `a, b, c` and `d`.

`a, b, c,` and `d` are the names of the variables.

Before a variable can be used it must be declared.

Task 3

Write the following program and try to **compile** it – there are a number of **errors** that you have to **fix**. If multiple error messages are displayed it is best to edit/correct and re-compile **one error at a time** since often one mistake can generate several error messages.

Task 3

```
#include "stdio.h"
#include "conio.h"

int main()
       float result1, result3; // Results of a division
statement
       int   result2, input1, Divisor; // Some int variables
       // Divide a floating point number by another
       result1 = 7.0 / 22.0;
       printF("The 7 divided by 22 is  %f", result1);
       // Add two times two to ten and print to screen
       input1 = 10;
       result2 = input1 + 2 * 2
       Printf(" %d plus two times two is %d ", input1,
result2);
       // Division
       divisor = 7;
       result3 = 1/divisor;
       printf("1/%d is ", divisor, result3);
       return 0;
}
```