Today’s Lecture

- Structure of the CP course (and CPMT)
- What is programming?
  - Imperative Programming?
  - C programming?
- Hello World.
- Any questions?

CP Lect 1 – slide 2 – 19 September 2016

Structure of CP

- 2 lectures per week:
  - Monday 14:10–15:00: LT3, Appleton Tower
  - Tuesday 11:10–12:00 LHA, DHT Lecture Theatres.
- One 2-hour lab starting **Thursday of week 1** in the Forrest Hill labs:
  - Thu 10:00–11:50 or Mon 15:10–17:00 or Tue 12:10–14:00
  - You will be **assigned** to a specific lab, though we will respect your preferences if possible (see form on Learn).
- One 1-hour tutorial per week, starting **week 3**.
  - You will be **assigned** to a specific tutorial group. If it doesn’t work for you, then ask for a change using the ITO contact form.
- If enrolment is higher than expected, we may need to change things at short notice: **check your Uni email regularly**

CPMT has the same structure except that **CPMT students will do an extra piece of practical work with the School of Music in week 11.**

CP Lect 1 – slide 4 – 19 September 2016

People on CP

- Julian Bradfield (lecturer and course organizer)
- Kenneth Heafield (lecturer)
- Mahesh Marina (lecturer)
- Paul Anderson (Informatics UG1 organizer)
  - Only contact with Paul will be if you have a (non-trivial) illness or other special circumstance, and wish to apply for a coursework extension
- Informatics Teaching Organisation (ITO)
  - They take care of the admin for assigning students to Tutorials and Labs. We have a link to their contact form on the course website.

CP Lect 1 – slide 3 – 19 September 2016
Resources

Course webpage:
http://www.inf.ed.ac.uk/teaching/courses/cp

(we will put all slides, lab sheets, tutorial sheets, lab/tutorial allocations etc on this webpage throughout the semester)

We do not use Learn, apart from the initial Lab signup poll.

Course textbooks:
- A Book on C: programming in C by Al Kelley and Ira Pohl (4th edition). A standard text, comprehensive – also quite large and expensive, but many second-hand copies.
- There are dozens of other books – any of them should be fine, so browse and see what you like the look of.

What is Programming?

Writing a program to solve a problem involves several steps:
- understand the problem (a very important step!)
- represent the problem in a precise way, using numbers, symbols, and other data types we will introduce you to
- work out how to compute the answer to the problem: design the algorithm
- express the algorithm in the programming language by writing instructions that conform to the syntax (permitted expressions) of the language
- This creates a program.

When the program is run according to the semantics (meaning of the expressions), this will perform the computational task the programmer intended.
- If, that is, you’re perfect. Otherwise:
  - find the first mistake in your algorithm or your program, fix it
  - rinse, lather, repeat.

Assessment

Your overall grade for CP will be based on two things:
- 90% from a 3-hour computer-based programming exam, which will take place in the exam period after semester 1 and before Christmas. We will run a mock exam on Thurs of week 8, and Mon, Tues of week 9 for practice.
- 10% from the coursework for CP, which will be released on Friday of week 5 and will be due on Friday of week 9. We aim to return feedback within two weeks.

For CPMT students the weighting is 70% exam, 10% coursework (due week 8) and 20% for the extra practical work (with School of Music) in week 11.

The practical labs are very important, if you want to become proficient enough to pass the computer-based exam, and more importantly if you actually want to learn to program!

What is Imperative Programming?

- The original style of programming-languages, and closest to what actually is done by computers at the ‘machine-level’.
- In Imperative programming, the expressions of the language are instructions or commands to the machine to perform some action:
  - Often the actions involve a change of state in the environment of the program (more in lecture 2);
  - Sometimes these are input or output actions.
- Writing an imperative program is like creating a “recipe” to solve a problem:
  - The instructions will be carried out one at a time;
  - The order of the instructions is important.
C programming

- C is an imperative programming language.
- Originally developed by Dennis Ritchie between 1969–1973 when he was working at Bell Labs.
- Our version of C (used on all modern platforms) is ANSI C (ANSI means “American National Standards Institute”), standardized in 1989.
- C is a small programming language, in terms of the number of operations and programming constructs which are supported:
  - There are many libraries of functions used to “fill-in the gaps”.
- There are several later ANSI standards for C (1995, 1999, 2011), and we’ll silently use one or two features of C99; C11 is not yet widely available with full support.

hello.c

```c
#include <stdlib.h> /* Include Standard Library */
#include <stdio.h> /* Include Input/Output Library */

int main() { /* Exactly one "main" function */
    printf("Hello, World.\n"); /* Output function printf */
    return EXIT_SUCCESS; /* "All ok" signal returned */
}
```

- Even basic Input/Output functionality is missing from the kernel of the C-programming Language.
- We needed to include the I/O library stdio.h in order to print to standard output.

Making hello.c active

- First need to translate our program into machine-level code (“executable”) that the computer can understand - this is called compiling the code:
  - The name of the compiler we use is gcc.
  - To compile hello.c, we type `gcc -Wall hello.c` at the command line.
  - The compiler creates the executable in the file a.out.

You will get experience with command-line, files, compiling in the first labsheet.

Output from running hello.c

- The process of executing a program on a computer is also called “running” a program.
- It is not really our original .c program that gets executed, but the executable code created by the compiler.
- To execute the program, we just type `./a.out`

On my computer:

```
zagreb: ./a.out
Hello, World.
```
What will you get from CP?

▶ Will learn how to convert simple computational problems (eg, solving quadratic equations) into a series of steps, convert these to C code, and run them;
▶ Will learn to apply the structured programming constructs of branching, iteration, functions and recursion towards solving more complex problems.
▶ Will learn the basics of working within a Linux/Unix environment, when working on our Informatics network.
▶ Experience with debugging, a necessary part of a programmer’s life.
▶ Maybe you will find a new career.

Wrapping Up

▶ If you aren’t already enrolled, and want to take this course, please email your Personal Tutor immediately and ask them to enrol you, if you plan to take this course.
  ▶ We cannot allocate your Informatics (DICE) username/password unless you are formally registered.

Any Questions/Comments??