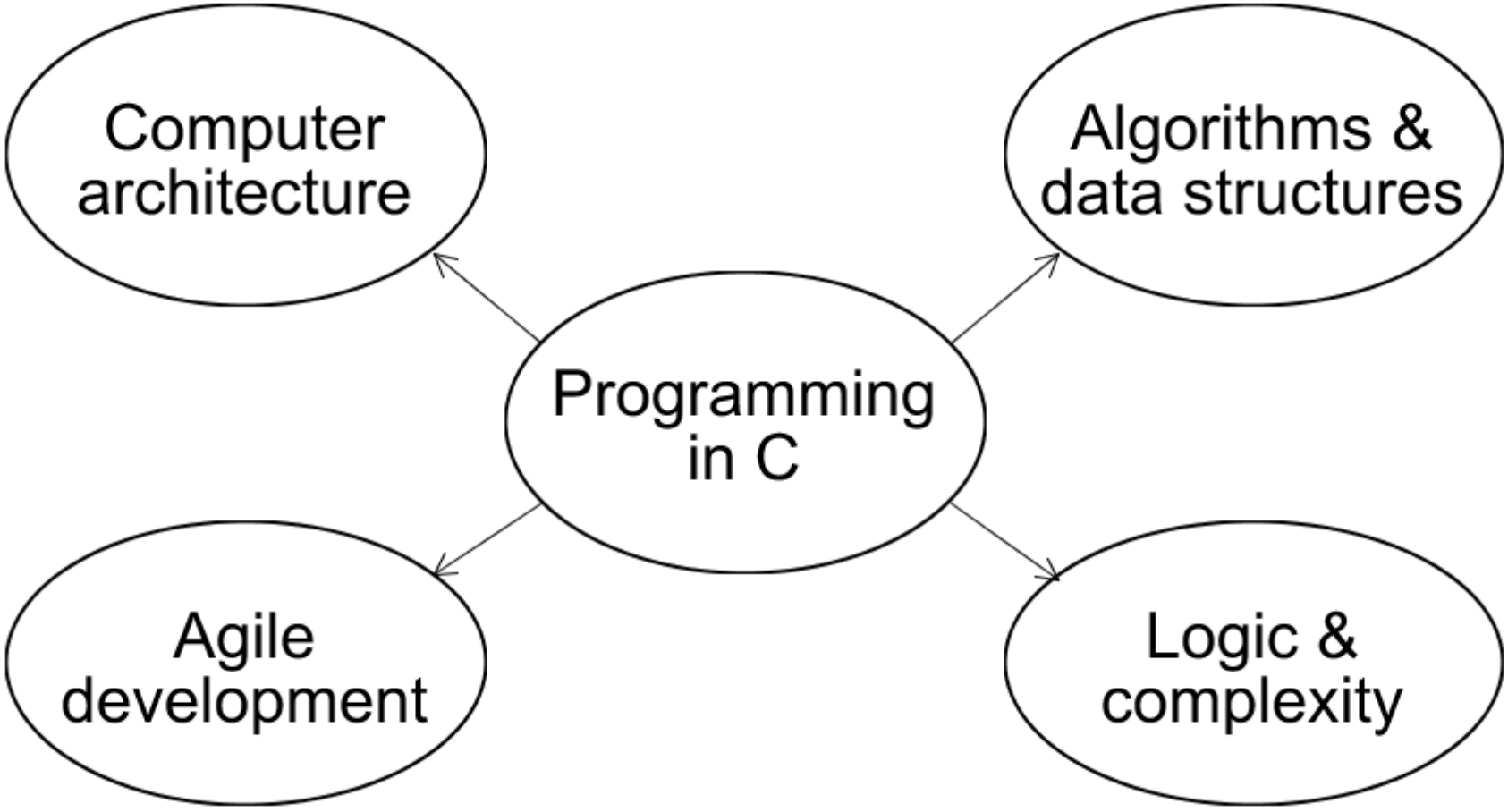
Imperative Programming

COMS10008

Ian Holyer

Links


Computer architecture
Algorithms & data structures
Programming in C
Agile development
Logic & complexity

Lecture and lab slots

- 2:00 Mon (QB Pugsley)
- 4:00 Fri (QB Pugsley)

- 2-4 Tue (MVB 2.11, Group 1)
- 4-6 Tue (MVB 2.11, Group 2)

Old lab, plus atrium overflow

Coursework

Coursework is supposed to take roughly 5 hours per week (based on a 40h week) to be done in:

- timetabled lab sessions, with help
- the (old or new) lab at other times, if not full
- your own time, on your own computer or Codio

There is an assignment every week, one in three will count towards your credit for the unit (no exam)

Materials

The unit web site is
https://csijh.gitlab.io/COMS10008/

- lecture notes
- coursework assignments
- asides (self-study notes)

It also has a link to SAFE (faculty server) for coursework submission and marks:
https://wwwa.fen.bris.ac.uk/COMS10008/

Asides

- computers
- compilers
- make
- book
- editors
- unix commands
- working together
- submission and marks
- version control

How hard?

Learning to program is like learning to drive:

- everyone can do it
- you have to put the hours in
- different people take different amounts of time
- the slowest learners can be the best programmers

Don't get stressed by other people racing ahead of you

Getting help

Get help from the students near you in the lab

Or the lab helpers

Or your tutor

Or Ian in MVB 3.27, see calendar (csijh)

If you get behind ***for any reason***, then talk to Ian, get some personal tuition

I prefer to see you in person - don't rely on email

Don't get stressed, get help

Low stress assignments

Each assignment typically comes in two parts

The first part is closed - it is fully specified, it is about precision, it is marked automatically, there is a skeleton program for you to fill in, which includes all the tests, and the number of tests passed when you submit is almost certainly the mark you'll get for it

The second part is open, "do what you like", marked by hand

Strategies

If you only want to pass, or you lack confidence, aim for a 50% mark by doing the first parts of the assignments (40% is a pass)

If you want a high mark, submit the first part for 50% then work out how much time you have left and spend it on the second part

Beware diminishing returns

Programming languages

There are various different kinds of programming languages:

- imperative (aka procedural, e.g. C)
- object oriented (OO, e.g. Java)
- functional (aka declarative, e.g. Haskell)

Imperative languages

Imperative means telling the computer ***how*** to do things

C is imperative, and it is the lowest level of language you are likely to want to use

Only assembly languages are lower

It is a good early language because it's cross-platform, most common languages including OO languages are based on its ideas, and it helps you to learn how computers work

Functional languages

Functional means telling the computer ***what*** needs doing

Haskell is functional, and it is the highest level of language you are likely to want to use

It is a good early language because it forces you to design instead of hack, it helps you to learn how to solve problems, and it illustrates the other end of the language spectrum

The C language

The C language is old (1970s) but not the oldest (Fortran, Lisp, 1950s)

It is important because

- it is still very popular
- it is available on every computer
- it matches what computer hardware provides
- it is used for operating systems and device drivers
- it is used for efficiency, e.g. in cryptography
- other languages use its libraries and conventions

Code

The text of a program is often called *code*

This is ok for 'machine code', but a ***rubbish*** name for higher level languages, because it encourages unreadable programs

A program should be extremely clear and readable (if you are fluent in the language, of course)

Coders

A *coder* is the lowest order of program writer

Our aim is to lift you to the level of *programmer* by the end of the year

And to lift you to the level of *developer* by the end of your degree

Of course, there are other things in your degree as well