COMP1511 Fundamentals of Programming

Dr John Shepherd

(C stream lecturer)

Welcome to COMP1511!

In this course, you will …

- learn to "think like a programmer"
- become part of the CSE community

Welcome (cont)

At the end of the course, you'll be able to …

- take a description of a problem
- design a step-by-step method of solving it
- implement your method in the C programming language

You will also …

- know you way around the Linux operating system
- be able to use Linux command-line tools
- and understand what on earth the above two lines mean

About You

We assume that you …

- have some mathematical background
- can speak fluent English
- have (maybe) touched a computer before

We do not assume

- that you have ever programmed before
- that you are familiar with the Linux OS

About Me

- Deputy Head of School (Education) in CSE
- taught many many CSE courses (UG and PG)
- likes: craft beer, AFL, programming, …
- dislikes: dishonesty, Donald Trump, …

How COMP1511 Runs

- lectures: explain concepts, give demos
- tutorials: clarify concepts, practice analysis
- lab classes: practice building small software
- assignments: build "large" software systems
- exam: show that you've worked out the above

COMP1511 Admin

Lecturer-in-charge & A stream: Andrew Taylor

Lecturer for B stream: Andrew Bennett

Lecturer for C stream: John (Andrew) Shepherd

Website: https://cgi.cse.unsw.edu.au/~cs1511/

• COMP1511 uses Webcms3 *not* Moodle

Email: andrewt@unsw.edu.au

COMP1511 Admin (cont)

Getting Help …

• read Course Outline (on web site)

• help sessions (listed on web site)

• ask lecturer after the lecture

• talk to your tutor

• ask on the course Forum (linked from webpage)

• Student Office (K17 ground floor)

COMP1511 Admin (cont)

• Assessment

• 12% lab exercises

• 8% weekly programming tests

• 6% assignment 1 … due week 5

• 12% assignment 2 … due week 8

• 12% assignment 3 … due week 12

• 50% final exam

Student Conduct

COMP1511 is a learning environment

• do not plagiarise, contract out work

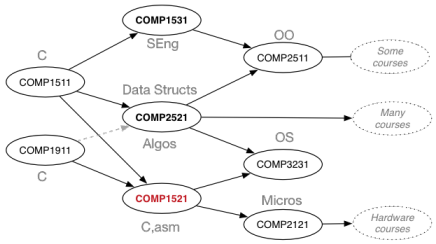
COMP1511 should be a safe environment

• do not troll, harass other course members

Breaches of above result in

• referral for UNSW academic misconduct

COMP1511 in Context



COMP1511 Alternatives

In 18s1, COMP1511 has very very high demand

COMP1511 is required for students studying

• Computer Science, Bioinformatics

• Computer Engineering, Software Engineering

• Mechatronics, Data Science & Decisions

For other degrees, it's optional …

Consider whether you need to take it in 18s1
(CSE majors do … others?)

Computers

"Computers" have existed for 1000's of years

E.g.

But, until 20th century, were specialised/simple devices

Computers (cont)

Modern computers are

- electronic, digital, stored-program
- able to realise any computable function
  - demonstrated by Alan Turing in the 1940's

Wednesday Admin

- make sure your UNSW email is set up ok
  - most reliable version **z1234567@unsw.edu.au**
  - if redirecting, check that redirection works
- lecture videos available on Echo360
  - only accessible via Moodle
  - link to Moodle under Course Work > Lectures
- some lecturers also put videos on YouTube

Wednesday Admin (cont)

Something to think about ..

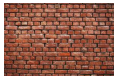
- COMP1511 is harder than COMP1911
  - despite having a lower number (1511 < 1911)
- if COMP1511 is core in your program
  - then you *must* do it … no choice
- if not core, consider COMP1911
  - can do more COMP via COMP1911 + bridging course

Programs

A program is

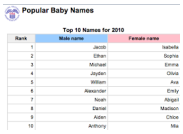
- a set of instructions
- that gives a procedure
- to accomplish a goal

Programs (cont)



Example goals:

- make a cake
- build a wall
- sort a list of names



Programs (cont)

A program needs to be

- sufficiently detailed
- unambiguous
- eventually leading to goal

So we don't use English for programming

Programs (cont)

So far, we're really describing *algorithms*

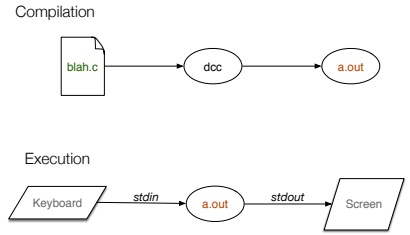
A program is a text document, containing

- a description of an algorithm
- expressed in a programming language

It cannot be directly executed on a computer

- need to translate to executable machine code

Programs (cont)



Programs (cont)

Typical program structure

- get input values
- process input to compute result
- display result

The C Programming Language

C is an important programming language

- relatively simple
- widely used

- venerable (developed in late 60's by Thompson & Ritchie)
- forms the basis for many other languages

C (the language)

A C program

- consists of a collection of functions
- one of which must be called main()
- may have functions in multiple files

By convention, C programs are

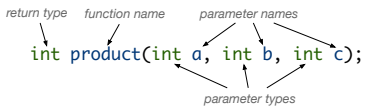
- text files with a .c suffix (e.g. myProgram.c)

C (the language) (cont)

Each C function has an interface, defined by

- the function name
- the type of its return value
- the names and types of its parameters

E.g.



C (the language) (cont)

A C program also contains variables

- variables are data objects
- each variable has a type
- each variable has a current value
- each variable has a location in memory

C (the language) (cont)

A C program also contains constants

- constants are fixed values
- each constant has a type, e.g.
  - 123 42 0 -1 are integer constants, type **int**
  - 3.14 1.0 are floating point constants, type **double**
  - 'a' '!' '5' are character constants, type **char**
  - "john" "123" are string constants, type **char ***

C (the language) (cont)

```
// Say hello                          <-- Comments
// Written Feb 2018

#include <stdio.h>                    <-- Directive

int main()                            <-- Function heading
{
    printf("Hello\n");                <-- Code
    return 0;
}
```

Writing C Programs

John's "patented" programming strategy …

- read and understand the requirements
- work out a computational method
- create a **.c** file containing **main()** function
- add your method as comments in **main()**
- treat each comment as a smaller problem
- for each smaller problem, write C code to solve it

Writing C Programs (cont)

Write C programs to do the following:

- do absolutely nothing (except succeed)
- print the word "Hello" on a line by itself
- show the meaning of life, the universe, everything

Writing C Programs (cont)

Another problem to solve in C:

- say hello to a person by name
  - print a message asking for the name
  - read the name
  - print "Hello, *name*" on a line by itself

Writing C Programs (cont)

Another problem to solve in C:

- add two numbers
  - print a message asking for the first number
  - read the first number
  - print a message asking for the second number
  - read the second number
  - add the numbers and print the sum on a line by itself

Writing C Programs (cont)

Another problem to solve in C:

- convert temperature in fahrenheit to celsius
  - print a message asking for the temperature in F
  - read in the temperature
  - convert to C $= \frac{5}{9} \times (F - 32)$
  - print value of C

Writing C Programs (cont)

Another problem to solve in C:

- ask for the meaning of life, universe, …
  - print a message asking for The Answer
  - read in the answer
  - if 42, then print "Ahhhh! … so that's it"
  - if non-zero, then print "Are you sure"?
  - if zero, then print "What's that supposed to mean?"