COMP1511 - Programming Fundamentals

Term 2, 2019 - Lecture 1

What are we talking about today?

First Hour

- Introductions
- Welcome to UNSW and Programming
- How COMP1511 works
  - How we will be teaching you
  - What we expect from you
- How to get help and the best ways to approach learning Programming

Second Hour

- What are these amazing machines we call computers?
- A first look at C
- Working in Linux

Who's Teaching you?

- Course Convener/Lecturer **Marc Chee** ← This is me!
  - marc.chee@unsw.edu.au
- Tutors
  - Too many to mention in person!
  - You will meet your tutor this week in class
- Course webpage https://webcms3.cse.unsw.edu.au/COMP1511/19T2/
- Course Forum (you should have received an email invite to this)

Welcome to UNSW and Programming

What is studying at University like?

- You are now in control of your own education

What is programming?

- Talking to computers . . . in a language that we can both understand
- Learning to solve problems
- We give the computer a procedure, not an answer

How is this course going to run?

We will teach you how to code, with no assumptions of prior knowledge

- Programming Fundamentals
- The C programming language
- How to solve problems with code

Course Format

- Lectures
- Tutorials
- Lab Classes
- Weekly Tests and Assignments
- Help Sessions
- Live Streams

Lectures

Two hour sessions

- Here in Mathews A
- Tuesday 2pm - 4pm
- Wednesday 4pm - 6pm

If you have a question, feel free to signal me, I'll answer as best as I can

There's only one rule: No one disrupts anyone else's education

Lecture Content

- Theory - What are we trying to understand?
- Demonstrations - Some live coding to show you how some things work
- Problem Solving - How do we approach problems in programming

Lecture slides are available from the Course Website

Lecture recordings will be available on the Course Website

Tutorials

A one hour classroom environment

- Go further in depth into the topics we're teaching
- Actual practical working of tasks and problems we've given you
- Learning how to solve problems before you write the code!

Tutorial Questions will be available in advance of the tutorials

Tutorials are a good place for interactive learning. You'll have time to discuss and work through problems there

Lab Classes

Two hour laboratory sessions that follow immediately from tutorials

- Actual coding on CSE computers
- Practical coding including working in pairs
  - Pairs will be assigned by your tutor
- Lab exercises will be marked automatically and count towards your final marks (13%)
- There are challenge exercises for earning bonus marks (not necessary and some are hard enough that they’re not necessarily worth your time)

Weekly Programming Tests

Self-run practice for exam situations

- 1 hour tests under exam conditions with automated marking
- Self-run exam practice
- Running from week 3-10
- These are ways for you to gauge your current progress
- Tests count towards your final mark (7%)

Assignments

Larger scale projects

- Individual work
- These will take you a few weeks and will test how well you can apply the theory you've learnt
- There are two Assignments due in Weeks 6 and 10
  - Late penalties of 2% per hour apply (this reduces your maximum possible mark)
- Each is worth 13% of your total marks

Help Sessions

Optional Sessions scheduled during the week

- Some one on one consultation with tutors
- Held in some of the same labs you have your tutorials in
- Time for you to ask individual questions or get help with specific problems
- Schedule will be up on the Course Website soon

Live Streaming

Optional Extra Content

- Live streaming sessions using Youtube live
- Marc will do some live coding
- Examples will be based on what questions people have been asking
- And whatever questions we can get to during the session
- These will likely be starting on Monday of Week 3

Exam

Practical Programming held in our labs

- You'll be expected to be able to program using the CSE environment
- Some online documentation will be available
- You'll be given a series of problems to solve in C
- You will also be expected to read some C and show you understand it

Total Assessment

- 13% Labs
- 7% Weekly Tests
- 13% Assignment 1
- 13% Assignment 2
- 54% Exam

To pass the course you must:

- Score 50/100 overall
- Solve problems using arrays in the final exam
- Solve problems using linked lists in the final exam

Supplementary Assessment

A second exam that will be offered to students who are granted Special Consideration by the UNSW Student Service Centre

- Identical in format to the main exam
- Usually held around two weeks after the exam (depending on timetabling)

Course Textbook

This is an optional book if you wish to use one

- Programming, Problem Solving, and Abstraction with C
  Alistair Moffat, Pearson Educational, Australia, 2012,
  ISBN 1486010970

Code of Conduct

This course and this University allows all students to learn, regardless of background or situation

Remember the one rule . . . you will not hinder anyone else's learning!

Anything connected to COMP1511, including social media, will follow respectful behaviour

- No discrimination of any kind
- No inappropriate behaviour
  - No harassment, bullying, aggression or sexual harassment
- Full respect for the privacy of others

Plagiarism

Plagiarism is the presentation of someone else's work or ideas as if they were your own.

Any kind of cheating on your work for this course will incur penalties (see the course guide for details)

And really . . . if you don't spend the time to learn the basics, the only person who loses is you

Collaboration vs Plagiarism

- Discussion of work and algorithms is fine (and encouraged)
- The internet has a lot of resources you should learn to use, just make sure you credit your sources
- No collaboration on individual assignments
- Your submissions are entirely your own work
  - Don't use other people's code
  - Don't ask someone else to solve problems for you
  - Don't provide your code to other people
- At best, you'll lose the marks for the particular assignment
- At worst, you'll be asked to leave UNSW
- And even worse . . . you won't learn what you paid all this money and time to learn

How to succeed in COMP1511 (and life)

A simple idea:

> *The more time you spend practising something, the better you get at it*

We've set up the course to ease your learning . . .

If you just follow what we're teaching, that will be a good start

- Complete all tutorial and lab work
- Complete all assignment work

Try not to focus on how good you are (or are not), but the real focus is on whether you're improving or not

Unlearn what you have learnt

If you want more info …

- Reading
  - Course webpage
  - Course forum
- Recorded Lectures
- In Person
  - Help Sessions
  - Come and see me after the lecture!
  - Ask your tutor during tutorials
- Serious Issues
  - Email marc.chee@unsw.edu.au
  - Engineering Student Support Service J17 104
  - CSE Help Desk (http://www.cse.unsw.edu.au/~helpdesk/)

Speaking of Email

Make sure you have your UNSW email address set up

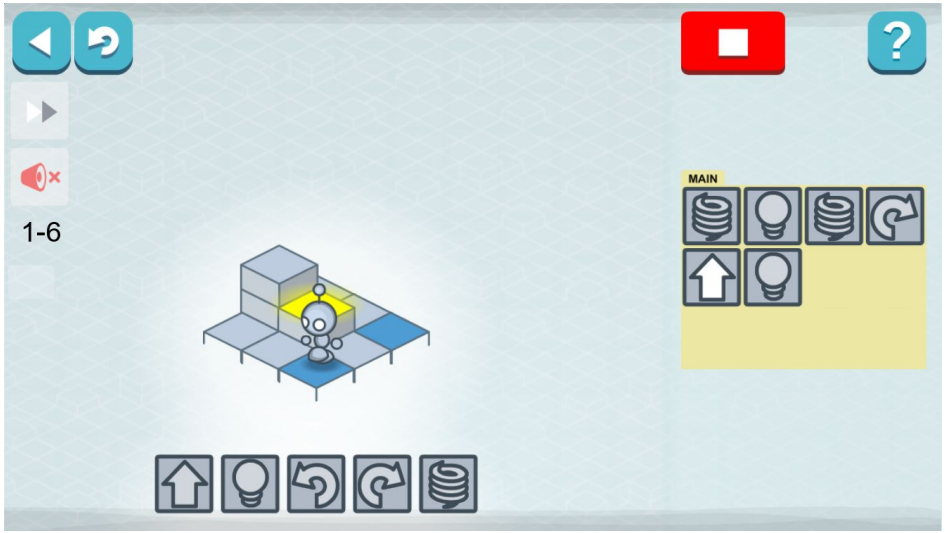
If you need to contact the course, use cs1511@cse.unsw.edu.au

Use your UNSW email address and make sure you include your zID so we know who we're helping

Break Time!

Let’s take five minutes in between lecture sections

- There's a fun little app called **Lightbot** (http://lightbot.com/flash.html)
- Also available on iOS and Android



What is a Computer?

A tool . . . a machine . . .

The ultimate tool in its ability to be reconfigured for different purposes.

The key elements:

- A processor to execute commands
- Memory to store information

History of Computers

Humans have been using calculation and data storage tools for millennia

The first concepts of a programmable computer were from around 1835

- Charles Babbage designed the first “Analytical Engine”
- Ada Lovelace was the first person to write a computer program

Computers this Century

In 1944, the computer Colossus is made in the UK to break German codes

Alan Turing was instrumental in the Allied effort in World War 2

He also developed the concept of the Turing Machine, which is the basis of all our modern computers

Modern Computing

We now have much more processing capability than in the past

So what can we do with it?

- Realtime solutions to very difficult problems
- Global simulations for climate and weather patterns
- Connecting individuals globally in communication networks
- Simulating highly complex virtual environments
- Deciding what you want to watch next on Netflix . . .

And maybe you will come up with something new to do with them in the future . . .

Using CSE's Computing Resources

Our labs are running **Linux** with the basic tools necessary to get started

You might also want to get your own computer ready to code with

Some options:

- VLAB allows you to remotely use CSE's resources
- You can set up a programming environment on your own computer (check the course website for links to guides)

For COMP1511 we need:

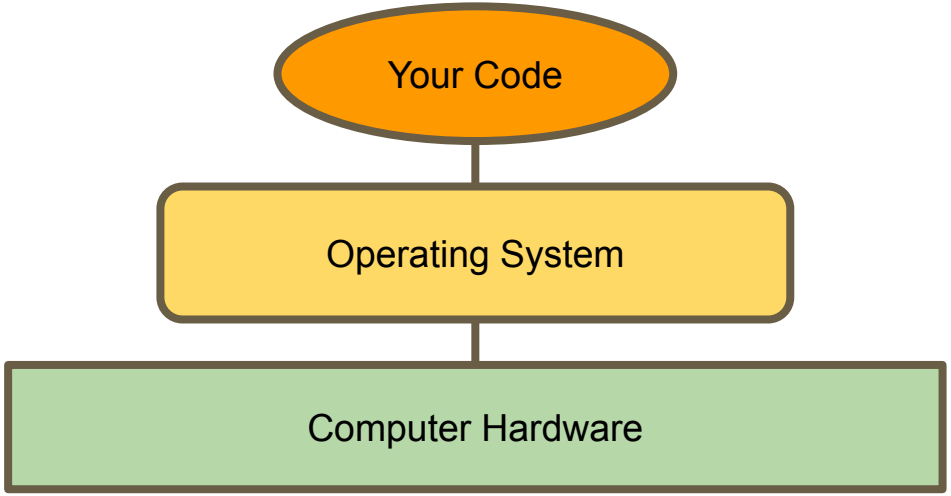
- A text editor like gedit
- A compiler

Working in Linux

The first thing is to get setup with a simple programming environment

Here at CSE we use the Linux Operating System

An Operating System sits between our code and the computer, providing essential services



Using a Terminal

The main interface to Linux is a terminal

This means all our interaction is in text

Some commands:

- ls
  - Lists all the files in the current directory
- mkdir *directoryName*
  - Makes a new directory called directoryName
- cd
  - Changes the current directory
- pwd
  - Tells you where you are in the directory structure at the moment

What the basics look like

gedit

- A basic text editor
- Helps out a little by highlighting C in different colours

dcc/gcc

- A compiler - A translator that takes our formal human readable C and turns it into the actual machine readable program
- The result of the compiler is something we can “run”

You can use VLAB to access CSE’s editor and compiler or you can get setup on your own computer

Programming in C

Programming is like talking to your computer

We need a shared language to be able to have this conversation

We'll be looking at one particular language, **C** and learning how to write it

C is:

- A clear language with defined rules so that nothing we write in it is ambiguous
- Many current programming languages are based on C
- A good starting point for learning how to control a computer from its roots

Let's see some C

```c
// Demo Program showing output
// Marc Chee, June 2019

#include <stdio.h>

int main (void) {
    printf("Hello World.\n");
    return 0;
}
```

Comments

```
// Demo Program showing output
// Marc Chee, June 2019
```

Words for humans

- Half our code is for the machine, the other half is for humans! (roughly)
- We put **“comments”** in to describe to our future selves or our colleagues what we intended for this code
- **//** in front of a line makes it a comment
- If we use **/*** and ***/** everything between them will be comments
- The compiler will ignore comments, so they don’t have to be proper code

#include

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

#include is a special tag for our compiler

It asks the compiler to grab another file of code and add it to ours

In this case, it's the Standard Input Output Library, allowing us to make text appear on the screen (as well as other things)

The main Function

```
int main (void) {
    printf("Hello World.\n");
    return 0;
}
```

A function is a block of code that is a set of instructions

Our computer will run this code line by line, executing our instructions

The first line tells us:

- int is the output - this stands for integer, which is a whole number
- main is the name of the function
- (void) means that this function doesn't take any input

The Body of the Function

```
int main (void) {
    printf("Hello World.\n");
    return 0;
}
```

Between the { and } are a set of program instructions

printf() is actually another function from **stdio.h** which we included. The brackets ( ) are like the brackets in our main, signifying input to a function

return is a C keyword that says we are now delivering the output of the function. A main that returns **0** is signifying a correct outcome of the program

Editing and Compilation

We can open a terminal now and try the code we've just looked at

In the linux terminal we will open the file to edit by typing:

```
gedit helloWorld.c &
```

Once we're happy with the code we've written, we'll compile it by typing:

```
dcc helloWorld.c -o helloWorld
```

The **-o** part tells our compiler to write out a file called "helloWorld" that we can then run by typing:

```
./helloWorld
```

The ./ lets us run the program "helloWorld" that is in our current directory

One working Program!

That's one program working!

What to do next?

- Try this yourself!
- Try it using a CSE lab computer
- Try it using VLAB via your own computer
- Try setting up a programming environment on your own computer (differing levels of difficulty depending on your operating system)

What did we learn today?

- COMP1511 and administration
- Where to find resources (course webpage and forum!)
- What your responsibilities are and how to succeed in programming

- A brief look at the history of computers
- An overview of how they work
- Your very first C program
- Using the basics of Linux