# Course Outline

# COMP1921
# Computing 1B
## Teaching Period 1 2010

## Contents


- Course Staff
- Course Details
- Course Summary
- Course Aims
- Student Learning Outcomes
- Assumed Knowledge
- Resources for Students
- Course Schedule
- Teaching Rationale
- Teaching Strategies
- Assessment
- Academic Honesty and Plagiarism
- Course Evaluation and Development


## Course Staff

**Contact Details**

| Staff Name | Role | Email | Phone |
|---|---|---|---|
| John Shepherd | Course Convenor | jas at cse.unsw.edu.au | 93856494 |

**Consultations**

| Staff Name | Day/Time | Location |
|---|---|---|
| John Shepherd | Tuesday 1300-1400 | K17 410 |
| John Shepherd | Wednesday 1300-1500 | K17 410 |
| John Shepherd | Thursday 1700-1800 | K17 410 |

## Course Details

**Course Code:** COMP1921
**Course Title:** Computing 1B
**Units of Credit:** 6
**Course WebSite:** http://www.cse.unsw.edu.au/~cs1921/
**Handbook Entry:** http://www.handbook.unsw.edu.au/undergraduate/courses/2010/COMP1921.html

## Course Summary

This course explores the process of developing software solutions in the context of the C programming language. While the main goal is to understand general principles that are relevant to software development tasks in any language, using the C language has some useful advantages. C is very widely used in building scientific and industrial software applications. Systems such as Linux, Mac OSX, the PostgreSQL database system, etc. have a much/all of their core implemented in C. For our purposes, C has the added advantage of giving an insight into how programming languages interact with the hardware.

## Course Aims

This course aims to develop your proficiency in C programming to the point where you could take a description of a relatively straightforward problem and produce an elegant, efficient and robust software solution for it. This should be enough for you to able to use computers for useful tasks in the engineering workplace, or, if you *catch the bug*, to continue on to further study in computing.

## Student Learning Outcomes

After completing this course, you should be able to:

- devise algorithmic solutions for well-specified problems and express these solutions as C programs
- develop suites of test cases to ascertain the accuracy and robustness of your programs
- repair any errors discovered in testing, and analyse and improve the performance of your programs
- understand and explain the run-time behaviour of your programs at the machine level
- make effective use of a number of common, useful data structures

In short, you will be a confident and competent C programmer.

This course contributes to the development of the following graduate attributes:

| **Graduate Attribute** | Where Acquired |
|---|---|
| the skills involved in scholarly enquiry | reading documentation, lab work |
| the capacity for analytical and critical thinking and for creative problem solving | developing programs in tutes, labs and assignments |
| the ability to engage in independent and reflective learning | exploring the C language in labs |
| the skills to locate, evaluate and use relevant information (Information Literacy) | reading documentation |
| the capacity for enterprise, initiative and creativity | developing programs in tutes, labs and assignments |
| a respect for ethical practice and social responsibility | building correct programs |
| the skills of effective communication | discussions in tutorials |

## Assumed Knowledge

Before commencing this course, students should:

- have completed a first course in C programming
- be able to design, construct, document and test simple C programs that accurately comply with a given specification
- have acquired some familiarity with the Unix/Linux operating system and tools, the C compiler, and the C debugger

We assume that you achieved all of this in COMP1911.

Realising, however, that summer can take a toll of your hard-won COMP1911 knowledge, we will spend a little time at the start of the course revising the basics of the C language and elementary algorithmic techniques.

## Resources for Students

The recommended text for this course is:

- Alistair Moffat, *Programming, Problem Solving, and Abstraction with C*, Pearson Educational, Australia, 2003, ISBN 1-74103-080-3.

The following books might also serve as additional reference material:

- Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language*, 2nd edition, Prentice Hall, USA, 1988, ISBN 0-13-110370-9.
- Paul Davies, *The Indispensable Guide to C*, Addison Wesley, England, 1995, ISBN 0-201-62438-9.

## Course Schedule

*No schedule is available (yet)*

## Teaching Rationale

This course has a heavy emphasis on programming practice because

- **the best way to learn to program is by doing it.**
- a useful way to pick up ideas about programming is to watch someone else doing it

To reflect these two observations, COMP1921 has been designed so that

- we have 4 lecture hours per week which will include numerous demonstrations of programming
- we have 3 hours of tutorial and laboratory exercises per week; lab attendance is compulsory
- we have 2 assignments to help you practice and extend the skills you have been learning about

We are serious about producing programmers, and so

- 100% of the marks in this course are earnt for your programming efforts

We will try to help you through demonstrations during lectures and labs and by providing some assistance during tutorials, in labs, and at consultations.

However, the only way you will really learn to program is by practice, practice, practice, practice, practice, practice, .....

# Teaching Strategies

COMP1921 includes lectures, tutorials, labs and assignments. Tutorials and labs are combined into 3-hour tute-lab periods. All components are designed with the aim of improving your ability to program.

## Lectures

Lectures are scheduled into a 2-hour block on Mondays and a 2-hour block on Tuesdays. Roughly half of the time per lecture will be spent explaining theory and programming concepts. The rest of the time will be spent on demonstrating the process of programming solutions to problems.

The material presented during lectures will not necessarily be comprehensive. Students are advised to read the course textbook to augment the ideas being presented - the depth and breadth of information presented in the text and references of a similar standard should be used as a guide for your learning.

Should you have any questions on the content or presentation, please don't hesitate to ask during lectures. Usually other people will have similar questions.

## Tutorials

Tutorials commence in Week 2.

Tutorials offer a vital chance for you to discuss problems and their solution in more depth with talented programmers. Please make the most of the opportunity by preparing for them and engaging in the discussion.

We will make tutorial questions available by 5pm on the Friday of the week before they are to be discussed. In general, tutors will not be providing answers to the tutorial questions directly. Their role is to facilitate the tutorial by steering the direction of the solution development and presentation. To get the most from tutorials **you must participate** in finding and presenting solutions to tutorial questions.

## Labs

Labs will commence in Week 2 as well. Lab exercises will be released by 5pm on Friday of the week before they are to be attempted.

It is important that you gain sufficient programming proficiency during the labs to be able to pass the final exam. Some lab exercises will have a flavour of the practical exercises in the final exam (although none of the lab exercises will be repeated on the final exam).

As much as possible, you should attempt to finish lab exercises within the 2-hour lab class. You can submit your lab exercises as soon as they are completed and assessed by your tutor, and *must* submit them by 5pm Friday each week. Late labs will not be accepted.

All lab exercises are assessable and will be marked by your tutor during lab sessions. A total of 15% of the marks in the course will be awarded for completing the selected exercises satisfactorily. The mark for an exercise will be determined on the basis of correctness, programming style, quality of comments, and your ability to explain good/bad points of your solution. The available lab marks sum to more than 15; you can score bonus marks by completing all labs satisfactorily.

## Assignments

This course involves two programming assignments worth a total of 25% of the marks in the course. These are to be completed individually. Please pay particular attention to the Plagiarism rules. It is likely that some aspects of the assignments will be tested during the final exam.

Assignments will be released at least 3 weeks before they are due. Assignment deadlines are noted in the course website.

**Late Submission:** The maximum mark achievable for an assignment will be reduced by 0.625% for each hour it is late (i.e. 15% per whole day).

## Exam

There will be a final exam worth 65% of the marks in this course. The exam will be a 3 hour on-line exam involving answers to both practical and theory questions. Your ability to understand and debug programs as well as your ability to create solutions to programming problems will be tested.

If you have developed the appropriate C skills during the semester, the practical questions on the exam will be relatively straightfoward. Since you'll also be familiar with debugging by then, even if you encounter problems with bugs you should have plenty of time to recover the situation.

## Assessment

Your mark in this course is computed as follows:

```
labs   = mark for 10 best labs     (out of 15)
ass1   = mark for assignment 1     (out of 11)
ass2   = mark for assignment 2     (out of 14)
exam   = mark for final exam       (out of 60)

okExam = (exam >= 25/60)           (after scaling)

mark   = labs + ass1 + ass2 + exam

grade  = HD|DN|CR|PS  if mark >= 50 && okExam
       = FL           if mark < 50 && okExam
       = UF           if !okExam
```

## Academic Honesty and Plagiarism

**Plagiarism** is defined as "using the words or ideas of others and presenting them as your own". UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Learning Centre: Plagiarism and Academic Integrity
- MyUNSW: Plagiarism and Academic Misconduct
- CSE: Addendum to UNSW Plagiarism Guidelines
- CSE: Yellow Form (whose terms you have agreed to)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism.

## Course Evaluation and Development

This course was developed in response to the difficulties meeting the differing needs, aspirations and capabilities of Computing and non-Computing students within our first programming courses. We have targetted this course at students who require a more practical and less theoretical background to programming for their careers.

At the end of the course, we would appreciate your feedback on your experiences via the University's CATEI system. The School uses the feedback you provide to improve the course for future sessions. The University uses the results of your evaluation to gauge its performance.

Previous suggestions that have been addressed include: clarifying the course objectives following discussions within the Schools of CSE, EET & SISTM, more extensive use of examples, less emphasis on computer organization, the need to improve course pass rates, the need to better integrate COMP1911 and COMP1921. Based on these changes, the CATEI evaluation for COMP1921 09s2 revealed no major problems, and so we will run this semester's offering similar to the 09s2 offering.

Some suggestions from 09s2 were: less revision of material from COMP1911 and not having 2 x 2-hour lectures. On the first point, I was planning to do a little revision, but you should let me know if you think I'm regurgitating too much COMP1911 material. One the second point, unfortunately, we have no control over lecture times these days; it's all done centrally to "optimise usage of space resources".

If you have any suggestions or concerns about the course during session, please do not hesitate to contact the lecturer, your tutor or the course administrator to let us know.

Enjoy the course! ... *jas*