Course Outline

CP1511

Introduction to Programming

Diploma Program

UNSW Global Education

Intake: September, 2018
1. Staff

<table>
<thead>
<tr>
<th>Position</th>
<th>Name</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Convenor &amp; Lecturer</td>
<td>Dr Angela Finlayson</td>
<td><a href="mailto:A.Finlayson@unswglobal.unsw.edu.au">A.Finlayson@unswglobal.unsw.edu.au</a></td>
</tr>
</tbody>
</table>

2. Course information

Units of credit (UOC): 6
Pre-requisite(s): none
Total course contact hours: 84

2.1 Course summary

This course introduces students to the basics of programming. Topics covered include:

- fundamental programming concepts
- the C programming language and use of a C compiler
- programming style
- program design and organisation concepts
- program testing and debugging

The course does not assume any previous programming experience.

2.2 Course aims

The course aims for students to become proficient in programming using a high level language, C. By the end of the course, students should be able to construct C programs to solve problems.
2.3 Course learning outcomes (CLO)

At the successful completion of this course you (the student) should be able to:

1. understand the core syntax & semantics of the C programming language including types, I/O, arrays, functions, pointers, structs, file manipulation and dynamic memory allocation
2. given a problem, solve it by proficiently constructing (designing, testing, debugging) a secure, reliable and correct C program
3. understand & employ fundamental data structures including stacks, queues and linked lists
4. use Linux and Unix-like operating systems to develop and test software

2.4 Relationship between course and program learning outcomes and assessments

<table>
<thead>
<tr>
<th>Course Learning Outcome (CLO)</th>
<th>Program Learning Outcome (PLO)</th>
<th>Related Tasks &amp; Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLO 1</td>
<td>Conceptual understanding of computer underpinnings (EA1.2)</td>
<td>Exam, Practical Exams, Labs, Assignments</td>
</tr>
<tr>
<td>CLO 2</td>
<td>Understanding of specialist bodies of engineering knowledge (EA1.3)</td>
<td>Exam, Practical Exams, Labs, Assignments</td>
</tr>
<tr>
<td>CLO 3</td>
<td>Understanding of underpinnings (EA1.1)</td>
<td>Exam, Practical Exams, Labs, Assignments</td>
</tr>
<tr>
<td>CLO 4</td>
<td>Understanding of specialist bodies of engineering knowledge (EA1.3)</td>
<td>Labs, Practical Exams, Assignments</td>
</tr>
</tbody>
</table>
3. Strategies and approaches to learning

3.1 Learning and teaching activities

This course involves a number of teaching activities:

Lectures – 4 hours per week

Lectures present theory and concepts, by way of case studies and practical examples. Lecture notes will be provided in advance of each class.

Tutorials – 1 hour per week

Tutorials allow students to collaboratively work through example problems to illustrate lecture idea, and have concepts from lectures clarified by the tutor.

Lab Classes – 3 hours per week

Lab classes involve small exercises where students build systems that illustrate the ideas covered in lectures. In the first 2 hours of labs per week, students work in pairs to write software, and show their work to the lab demonstrator for assessment and feedback. Optional “challenge exercises” will be provided for students who find the regular weekly exercises too easy. Each lab is worth a possible 1.2 marks and there are 12 labs. The best 10 out of 12 lab marks are added up and capped at 10.

In the third hour of labs per week, students work individually on programming. In some weeks this hour will hold assessable practical exams.

Practical Exams – 5 during the course

Practical Exams are assessed and help prepare students for the final exam. These are held in weeks 5, 7, 9, 11 and 12. They run for 50 minutes and are each worth 2 marks.

Assignments – 2 during the course

Assignments are take-home problems that are larger in scope than Lab exercises and require students to use creativity to solve a challenging realistic problem. Each assignment requires students to understand the problem, design a solution, and implement and test their solution. Students will be required to maintain a reflective diary (blog) of their experiences during the assignments.

Online Forum

An online forum allows students to ask and answer questions on the tutorial, lab and assignment exercises, and on lecture material.
Final Exam
There will also be a formal 3 hour final exam conducted in the labs, where students will need to solve both theory and practical problems.

3.2 Expectations of students

Students are expected to:

- attend all lectures, and ask questions, but otherwise not disturb other students
- attend all tutorials and actively participate in the discussions
- attend all lab classes and work diligently on the exercises
- do all of the assignment work themselves, asking only the forum or tutors for help

On the course forum, students should:

- use relevant/meaningful message titles on all posts
- ask questions clearly and provide sufficient background information that the question can be reasonably answered
- not post significant pieces of code, especially code for assignments
# 4. Course schedule and structure

This course consists of 8 hours of class contact hours per week. You are expected to take an additional 5 hours outside classes to complete assessments, readings, and exam preparation.

<table>
<thead>
<tr>
<th>Week</th>
<th>Lectures</th>
<th>Tutorial and Labs</th>
<th>Assessment</th>
<th>Related CLO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1</td>
<td>Introduction to course/Linux/C; data types; variables, simple I/O, expressions, It Statements</td>
<td>Create/run first C programs on Linux</td>
<td></td>
<td>1,4</td>
</tr>
<tr>
<td>Week 2</td>
<td>Loops</td>
<td>Numeric I/O, arithmetic and simple control</td>
<td></td>
<td>1,2,4</td>
</tr>
<tr>
<td>Week 3</td>
<td>Memory and Functions</td>
<td>complex expressions, iteration</td>
<td></td>
<td>1,2,4</td>
</tr>
<tr>
<td>Week 4</td>
<td>Arrays</td>
<td>Functions, exploring memory</td>
<td></td>
<td>1,2,4</td>
</tr>
<tr>
<td>Week 5</td>
<td>Chars, strings, files</td>
<td>Array manipulation</td>
<td>Assignment 1 Released Practical Exam 1</td>
<td>1,2,4</td>
</tr>
<tr>
<td>Week 6</td>
<td>Pointers and extra C (for loops, pre/post increment, multi-file compilation)</td>
<td>Character-based I/O,</td>
<td></td>
<td>1,2,4</td>
</tr>
<tr>
<td>Week</td>
<td>Topic</td>
<td>Additional Details</td>
<td>Assignment</td>
<td>Notes</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------------------------------</td>
<td>--------------------------------------------------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>7</td>
<td>Dynamic memory allocation, C implementation memory models, structs</td>
<td>Command Line Arguments, File Manipulation</td>
<td>Practical Exam 2</td>
<td>1,2,4</td>
</tr>
<tr>
<td>8</td>
<td>Linked Lists</td>
<td>malloc, pointers, structs</td>
<td>Assignment 1 Due</td>
<td>1,2,3,4</td>
</tr>
<tr>
<td>9</td>
<td>Abstract Data Types, including stacks and queues</td>
<td>Programming with linked-lists</td>
<td>Practical Exam 3 Assignment 2 released</td>
<td>1, 2, 3, 4</td>
</tr>
<tr>
<td>10</td>
<td>Recursion including recursion with linked lists</td>
<td>Programming and testing ADTs</td>
<td></td>
<td>1, 2, 3, 4</td>
</tr>
<tr>
<td>11</td>
<td>Introduction to searching, sorting and complexity</td>
<td>Programming with linked lists using recursion</td>
<td>Practical Exam 4</td>
<td>1, 2, 3, 4</td>
</tr>
<tr>
<td>12</td>
<td>Exam Information and Revision</td>
<td>Sorting and a practice exam</td>
<td>Practical Exam 5 Assignment 2 due</td>
<td>1, 2, 3, 4</td>
</tr>
</tbody>
</table>
5. Assessment

5.1 Assessment tasks

<table>
<thead>
<tr>
<th>Assessment task</th>
<th>Length</th>
<th>Weight</th>
<th>Due</th>
<th>CLOs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assessment 1: Programming assignment (arrays)</td>
<td>3 weeks</td>
<td>15%</td>
<td>Monday week 8</td>
<td>1, 2, 4</td>
</tr>
<tr>
<td>Assessment 2: Programming assignment (linked data structures)</td>
<td>3 weeks</td>
<td>15%</td>
<td>Friday week 12</td>
<td>1, 2, 3, 4</td>
</tr>
<tr>
<td>Assessment 3: Lab exercises</td>
<td>Throughout semester</td>
<td>10%</td>
<td>Weekly, on Sunday</td>
<td>1, 2, 3, 4</td>
</tr>
<tr>
<td>Assessment 4: Practical Exams</td>
<td>Throughout semester</td>
<td>10%</td>
<td>Weeks 5, 7, 9, 11, 12</td>
<td>1, 2, 3, 4</td>
</tr>
<tr>
<td>Assessment 5: Final Exam</td>
<td></td>
<td>50%</td>
<td></td>
<td>1, 2, 3, 4</td>
</tr>
</tbody>
</table>

There are 2 hurdle requirements on the final exam.

Hurdle requirement #1: in the final exam you must solve a task by writing a program that uses an **array**. The final exam will contain multiple questions (clearly marked) which if answered successfully will meet this hurdle requirement. Answering any one of these questions will meet this hurdle requirements.

Hurdle Requirement #2: on the final exam you must solve a task by writing a program that uses a **linked list**. The final exam will contain multiple questions (clearly marked) which if answered successfully meet this hurdle requirement. Answering any one of these questions will meet this hurdle requirements.

You **can not** pass CP1511 unless you achieve both the above hurdles. However you will be offered an additional chance to pass the hurdles in the supplementary exam, if you achieve a mark of 50+ but do not pass both hurdles.
5.2 Assessment criteria and standards

In all programming work, the primary assessment criterion is correctness (i.e. does the code produce the expected output/behaviour according to the exercise specification). This will be tested by executing code against a variety of test cases, some of which are available to students, and others of which are used after submission for assessment purposes. Code is also expected to be expressed clearly, with consistent formatting and using relevant variable names.

5.3 Submission of assessment tasks

All assignments will be submitted online via CSE’s submission system. Late penalties accrue on an hourly basis. Marks are capped according to how late the submission is, but it would typically be the case that marks are capped at 50% after 36 hours.

If you are unable to submit an assignment by the due date, due to medical reasons or other reasons which significantly affect your ability to carry out your work, you should contact the lecturer as soon as possible, preferably well before the assignment deadline. If the lecturer considers that your ability to complete the assignment on time has been adversely affected, an extension may be granted to make up for the time you were unable to work on the assignment.

Lab exercises must be submitted by the end of the Sunday after the lab class. Demonstrators will then look at the exercise, and assess it, possibly asking you to explain what you did in the following lab class. Failure to complete and submit a lab exercise results in a mark of zero for that lab.

5.4. Feedback on assessment

Assignments will be marked after the submission deadline and annotated with comments by the tutor. You can discuss the tutor’s comments in a lab class after you have received the feedback.

Lab demonstrators will discuss your lab submission with you during the lab class in the week following the submission.

6. Readings and resources

The course website contains a reading list, along with all course notes.
We gratefully acknowledge the contributions of the School of Computer Science and Engineering (UNSW) towards the creation of this Course Outline.