

Subject Description Form

Subject Code	ENG236
Subject Title	Computer Programming
Credit Value	3
Level	2
Pre-requisite / Co-requisite/ Exclusion	Nil/Nil/Nil
Objectives	<ul style="list-style-type: none"> (i) To introduce the fundamental concepts of computer programming (ii) To equip students with sound skills in C/C++ programming language (iii) To equip students with techniques for developing structured computer programs (iv) To demonstrate the techniques for implementing engineering applications using computer programs.
Intended Learning Outcomes	<p>Upon completion of the subject, students will be able to:</p> <p>Develop a good computer program using C/C++ programming language. To be specific, the students should be able to achieve the following:</p> <ul style="list-style-type: none"> a) Familiarize themselves with at least one C/C++ programming environment. b) Be proficient in using the basic constructs of C/C++ to develop a computer program. c) Be able to develop a structured and documented computer program. d) Understand the fundamentals of object-oriented programming and be able to apply it in computer program development. e) Be able to apply the computer programming techniques to solve practical engineering problems. f) Be able to solve problems by using systematic approaches in a team.
Subject Synopsis/ Indicative Syllabus	<ol style="list-style-type: none"> 1. Introduction to programming - Components of a computer; Programming environment; Process of application development. 2. Bolts and Nuts of C/C++ - Preprocessor; Program code; Functions; Comments; Variables and constants; Expressions and statements; Operators. 3. Program Flow Control - Branching and looping; Function parameters passing; Return values; Local and global variables; Scope of variables. 4. Program Design and Debugging - Structured program design; Modular programming; Exceptions and debugging. Case study: Using the Visual C++ debugger. 5. Basic Object Oriented Programming - Objects and classes; Private versus public; Implementing class methods; Constructors and destructors.

	<p>6. Pointer and Array - The stack and the free store; Create and delete objects in the free store; Pointer arithmetic; Passing function arguments by pointer; Returning values by pointer; Array of objects; Array and pointer; Array of pointers; Pointer of array; Character array; Command-line processing.</p> <p>7. Stream I/O - Input and output as streams; File I/O using streams.</p> <p>8. Using C/C++ in Engineering Applications - Solving practical problems using C/C++; Developing graphical user interfaces for engineering applications.</p>																																																														
<p>Teaching/Learning Methodology</p>	<p>The subject is delivered through weekly lectures. Tutorials in terms of exercises related to the lecturing materials follow in the same week. Tutors will aid the lecturers in helping the students finishing the exercises, and interactive Q&A will take place. The lectures and tutorials aim at achieving the learning outcomes a, b, c, d and e.</p> <p>To assure students' understanding of fundamental concepts, short-quizzes and closed-book tests are arranged regularly. The learning outcomes b, c and d can be evaluated at different check-points.</p> <p>To enhance the students' problem solving skill in a given programming environment, open-book programming tests are arranged regularly. The learning outcomes a, b, c, d and e can be evaluated at different check-points.</p> <p>After all the subject materials have been delivered, students are asked to finish a mini-project in a team. The project involves a practical engineering problem of some stated specifications. Apart from meeting the learning outcomes a-e, the students have to practice solving problems using systematic approaches in a team. The learning outcome f should be reflected from the mini-project result.</p>																																																														
<p>Assessment Methods in Alignment with Intended Learning Outcomes</p>	<table border="1" data-bbox="443 1205 1471 1787"> <thead> <tr> <th rowspan="2">Specific assessment methods/tasks</th> <th rowspan="2">% weighting</th> <th colspan="6">Intended subject learning outcomes to be assessed (Please tick as appropriate)</th> </tr> <tr> <th>a</th> <th>b</th> <th>c</th> <th>d</th> <th>e</th> <th>f</th> </tr> </thead> <tbody> <tr> <td>1. In-class exercises</td> <td>10</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td></td> </tr> <tr> <td>2. Short-quizzes</td> <td>10</td> <td></td> <td>✓</td> <td>✓</td> <td>✓</td> <td></td> <td></td> </tr> <tr> <td>3. Closed-book tests</td> <td>20</td> <td></td> <td>✓</td> <td>✓</td> <td>✓</td> <td></td> <td></td> </tr> <tr> <td>4. Programming tests</td> <td>30</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td></td> </tr> <tr> <td>5. Mini-project</td> <td>30</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> <td>✓</td> </tr> <tr> <td>Total</td> <td>100 %</td> <td colspan="6"></td> </tr> </tbody> </table> <p>Explanation of the appropriateness of the assessment methods in assessing the intended learning outcomes:</p> <p>The short-quizzes and closed-book tests are for assessing the understanding of fundamental concepts. The in-class exercises and programming tests are conducted within the programming environment to help students familiarized with it. The problems to be solved by the students are typically presented as practical engineering problems. Through conducting the mini-project that lasts for several weeks, students</p>	Specific assessment methods/tasks	% weighting	Intended subject learning outcomes to be assessed (Please tick as appropriate)						a	b	c	d	e	f	1. In-class exercises	10	✓	✓	✓	✓	✓		2. Short-quizzes	10		✓	✓	✓			3. Closed-book tests	20		✓	✓	✓			4. Programming tests	30	✓	✓	✓	✓	✓		5. Mini-project	30	✓	✓	✓	✓	✓	✓	Total	100 %						
Specific assessment methods/tasks	% weighting			Intended subject learning outcomes to be assessed (Please tick as appropriate)																																																											
		a	b	c	d	e	f																																																								
1. In-class exercises	10	✓	✓	✓	✓	✓																																																									
2. Short-quizzes	10		✓	✓	✓																																																										
3. Closed-book tests	20		✓	✓	✓																																																										
4. Programming tests	30	✓	✓	✓	✓	✓																																																									
5. Mini-project	30	✓	✓	✓	✓	✓	✓																																																								
Total	100 %																																																														

	would be able to experience how to solve problems by using systematic approaches in a team.	
Student Study Effort Required	Class contact:	65 Hrs.
	▪ Lecture	27 Hrs.
	▪ Tutorial	26 Hrs.
	▪ Test/Quiz	11 Hrs.
	▪ Mini-project presentation	1 Hrs.
	Other student study effort:	81 Hrs
	▪ Self-studying	52 Hrs.
	▪ Homework	17 Hrs.
	▪ Mini-project/Report	12 Hrs.
	Total student study effort	146 Hrs.
Reading List and References	<p>Textbook:</p> <ol style="list-style-type: none"> 1. J. Liberty, S. Rao, and B. Jones, <i>Sams Teach Yourself C++ in One Hour a Day</i>. Sams, 2009. <p>Reference Book:</p> <ol style="list-style-type: none"> 1. H.M. Deitel and P.J. Deitel, <i>C++ How To Program</i>, 5th ed., Prentice-Hall, 2005. 2. I. Horton, <i>Ivor Horton's Beginning Visual C++ 2005</i>, Wiley Publishing, 2006. 	

July 2010