PROGRAMMING IN C - 2024/5

Module code: EEE1035

Module Overview

Module purpose: Programming is a key part of electronic engineering and the C programming language is at the heart of many embedded software systems. This module will provide the students with a solid practical knowledge of the C programming language, its relationship to the underlying hardware and aspects of both high level programming and low level manipulation of memory.

Module provider

Computer Science and Electronic Eng

Module Leader

PORTO BUARQUE DE GUSMAO Pedro (CS & EE)

Number of Credits: 15

ECTS Credits: 7.5

Framework: FHEQ Level 4

Module cap (Maximum number of students): N/A

Overall student workload

Independent Learning Hours: 62

Lecture Hours: 18

Tutorial Hours: 6

Laboratory Hours: 44

Guided Learning: 10

Captured Content: 10

Module Availability

Semester 2

Prerequisites / Co-requisites

None.

Module content

Programming in C Spring

Introduction, origins of C, Advantages and Disadvantages of C, general course overview, introduction to labs etc.

Binary Representation, decimal, binary, octal and hexadecimal number system. Boolean operations, Binary arithmetic.

Hardware and Software, Computer Hardware, Operating Systems, Memory Structures, Machine Code, High-low level languages, Compilers and linkers, Program Layout, Pre-processor, ANSI standard.

Simple Data Types, Variables, Variable Definitions, Identifiers (data types), Constants, Characters and String Constants, Using #define.

Standard IO, Data Types, printf and formatting, format sequences, scanf, problems with scanf, Reading Characters - getchar, Detecting Keystrokes, Keyboard Buffering, Detecting Invalid Input.

Operators, Expressions and Statements, Integer Expressions, Floating Point Expressions, Assignment Statements, Arithmetic Operators, unary Operators, Precedence, Math Functions, Overflow and Underflow, Mixed Type Expressions, Type Casting, Increment and Decrement Operators.

Making Decisions, Conditional statements, if…else, case statements.

Looping, the for statement, while loops, do…while loops.

Arrays, Array Concepts, Array Definitions, Array Subscripts, Passing Arrays.

Basics of Pointers, Pointer concepts, Defining Pointers, The Operator, Assigning Pointers, Pointer Indirection, null Pointers, Pointers as Parameters, Arrays as Parameters, Pointers as Return Values, Absolute Addressing.

Strings, String variables, I/O of Strings, String Assignments, String Comparison, Case Conversion, Library Functions, Value Conversions.

Basics of Functions, Simple Procedural Functions, Parameterised Function, Function with Return Values, Local & Global Variables, the   main function, Scope.

More functions, pass by reference vs pass by value

Files, File Structure, Opening and Closing Files, Writing to Files, Reading Files,W User Specified Files.

Data Structures, Concept of Structure, Declaring New Data Types, Declaring a Structure, Defining Structured Variables, Accessing Structured Variables, Array of Structures, Passing Records to Functions, Returning Structures.

Assignment lectures

Introduction to programming Assignment 1

Introduction to programming Assignment 2

Assessment pattern

Assessment type	Unit of assessment	Weighting
Coursework	In lab Assessment	20
Coursework	Programming assignment 1	40
Coursework	Programming assignment 2	40

Alternative Assessment

N/A

Assessment Strategy

The **assessment strategy** for this module is designed to provide students with the opportunity to demonstrate the following.

- Assessment of performance/understanding of laboratories. This assesses the students general understanding of the material as the course progresses. 2 programming assignments will be performed by the students. Each of these assignments will be presented to the students in lectures allowing design guidance to be provided and any questions answered. Assessment allows the students to demonstrate being able to break down and tackle more complex programming problems.

Thus, the **summative assessment** for this module consists of the following.

- Four assessments during laboratories
- Two programming assignments

Any deadlines given here are indicative. For confirmation of exact date and time, please check the Departmental assessment calendar issued to you.

Formative assessment and feedback

For the module, students will receive formative assessment/feedback in the following ways.

- During lectures, by question and answer sessions
- During tutorials/tutorial classes
- By means of unassessed tutorial problem sheets (with answers/model solutions)
- During supervised computer laboratory sessions
- Via assessed coursework

Module aims

- To familiarise students with the fundamentals of Computer Architecture
- To discuss the relationship between the underlying computer and the C language.
- To develop these concepts and show how the language can be used as a true high-level language
- To demonste the concepts of both procedural programming and data abstraction.
- To provide each student with practical experience of programming in a UNIX workstation environment to help build understanding and confidence in programming.
- The module also aims to provide opportunities for students to learn about the Surrey Pillars listed below.

Learning outcomes

Ref		Attributes Developed	
001	Demonstrate competency with a modern networked computing facility (the UNIX operating system, X windows and the world wide web)	KCPT	C3
002	Describe the relationship between computer architecture and programming	KC	C1
003	Demonstrate understanding of the principles behind procedural and data abstraction	KCT	C2
004	Apply abstraction to tackle large scale programming problems	PT	C3
005	Independently design and code relatively complex programs in C	KCPT	C6

Attributes Developed

C - Cognitive/analytical

K - Subject knowledge

T - Transferable skills

P - Professional/Practical skills

Methods of Teaching / Learning

The learning and teaching strategy is designed to achieve the following aims.

The course consists of lectures and supervised hours per week in a terminal or workstation room.

The fundamentals of the C programming language will be covered within lectures. These will be supported with a concurrent programme of laboratories where each aspect of the language covered in lectures is explored in a practical setting.

Programming assignments will be performed by the students ,each of these assignments will be presented to the students in lectures allowing design guidance to be provided and any questions answered. For each assignment, supported labs will be provided per assignment.

Learning and teaching methods include the following.

Lectures

Labs

Tutorials

Indicated Lecture Hours (which may also include seminars, tutorials, workshops and other contact time) are approximate and may include in-class tests where one or more of these are an assessment on the module. In-class tests are scheduled/organised separately to taught content and will be published on to student personal timetables, where they apply to taken modules, as soon as they are finalised by central administration. This will usually be after the initial publication of the teaching timetable for the relevant semester.

Reading list

https://readinglists.surrey.ac.uk
Upon accessing the reading list, please search for the module using the module code: **EEE1035**

Other information

This module will enhance **digital capabilities** by providing a fundamental grounding in computer programming and the underlying architecture of modern-day computers. By developing skills using a real-world programming language, the module will contribute to the improving the **employability** of students as the C programming language is not only a relevant programming language for embedded systems but is indicative of many procedural programming languages. In fact, many languages borrow their syntax and structure directly from C. As such the skill developed in this course are transferable to a wider skillset of problem solving. This in turn will help develop **resourcefulness and resilience**, especially as the course is taught and assessed through a series of independent programming assignments that require students to manage their time, use the resources around them (both staff, demonstrators, and fellow students) to develop effective approaches to problem solving and decision making. Complexity of the material and problems increases throughout the course to provide scaffolded learning for the students.

Programmes this module appears in

Programme	Semester	Classification	Qualifying conditions
Astronautics and Space Engineering BEng (Hons)	2	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Astronautics and Space Engineering MEng	2	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Computer and Internet Engineering BEng (Hons)	2	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Computer and Internet Engineering MEng	2	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electrical and Electronic Engineering BEng (Hons)	2	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electrical and Electronic Engineering MEng	2	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electronic Engineering BEng (Hons)	2	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electronic Engineering MEng	2	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electronic Engineering with Computer Systems BEng (Hons)	2	Compulsory	A weighted aggregate mark of 40% is required to pass the module
Electronic Engineering with Computer Systems MEng	2	Compulsory	A weighted aggregate mark of 40% is required to pass the module