Title:	C Programming APPROVED
Long Title:	C Programming
Module Code:	SOFT7019
Credits:	5
NFQ Level:	Intermediate
Field of Study:	Computer Software
Valid From:	Semester 1 - 2017/18 (September 2017)
Module Delivered in	4 programme(s)
Module Coordinator:	Donna OShea
Module Author:	Ignacio Castineiras
Module Description:	C is a general-purpose, procedural, imperative computer programming language that is used to develop applications in many domains including embedded systems. In this module the learner will effectively design and implement C applications, with a special emphasis for the memory representation of a program execution. Basic assembly commands (for low-level access to the machine architecture) and their embedding within a C program are also discussed.

Learning Outcomes	
On successful completion of this module the learner will be able to:	
LO1	Assess the memory representation in the execution of a program and evaluate the evolution of the stack and heap areas.
LO2	Apply pointers in programming and use pointers to achieve efficient handling of storage.
LO3	Model complex applications via the use of arrays and user-defined data types (structs).
LO4	Apply well-known String and Input/Output (I/O) APIs to efficiently read from/write to text files.
LO5	Embed assembler instructions within a C program to control the use of registers and system calls.

Pre-requisite learning

Module Recommendations
This is prior learning (or a practical skill) that is strongly recommended before enrolment in this module. You may enrol in this module if you have not acquired the recommended learning but you will have considerable difficulty in passing (i.e. achieving the learning outcomes of) the module. While the prior learning is expressed as named CIT module(s) it also allows for learning (in another module or modules) which is equivalent to the learning specified in the named module(s).

12689	SOFT7019	C Programming

Incompatible Modules
These are modules which have learning outcomes that are too similar to the learning outcomes of this module. You may not earn additional credit for the same learning and therefore you may not enrol in this module if you have successfully completed any modules in the incompatible list.

No incompatible modules listed

Co-requisite Modules

No Co-requisite modules listed

Requirements
This is prior learning (or a practical skill) that is mandatory before enrolment in this module is allowed. You may not enrol on this module if you have not acquired the learning specified in this section.

No requirements listed

Co-requisites

No Co Requisites listed

Module Content & Assessment

Indicative Content
Introduction. Basic syntax. Compiled vs. interpreted languages. The C-compiler application. Stages: Pre-processing, compilation and linking. Circular dependencies. Makefiles.
Memory Perspective. Hexadecimal representation. Virtual memory: Static-based data and code areas vs. dynamic-based stack and heap. Heap memory management: Malloc, free and realloc. Memory debugging tools.
Pointers. Definitions and operatiors. Void, non-initialised and null pointers. Maths with pointers. Pointer as function arguments. Pointer to pointers. Pointer to functions.
Arrays. Declaration and initialisation. Equivalence of arrays and pointers. Array limitations. Functions and arrays. Multi-dimensional arrays.
Strings and Input/Output (I/O) APIs Main API. Variable-length argument programs. Reading and writing characters and lines. Parsing text and formatted files.
User-defined Data Types. Structs. Fields access and manipulation. Compound structs. Structs and dynamic data structures. Struct pointers for isolating data usage from data internal representation.
Language Spectrum High, mid and low-level languages. Positioning C vs. a low-level language as assembly and a high-level language as Java or Python. Assembly and machine architecture: Registers, Memory Allocation, System Calls, Data Transfer, Memory Stack, Labels and Branches. Use 'asm' to embed assembler instructions within a C program to control the use of registers and system calls.

Assessment Breakdown	%
Course Work	100.00%

Course Work

Assessment Type	*Assessment Description*	*Outcome addressed*	*% of total*	*Assessment Date*
Project	Design and implement a purely stack-based C application involving the use of pointers and arrays. Use memory debugging tools to assess the memory representation in the execution of the application.	1,2	35.0	Week 5
Project	Design and implement a C application involving the use of pointers, arrays, strings and user-defined data types. Embed assembler instructions within a component of the C application (to control the use of registers and system calls).	3,4,5	35.0	Week 11
Short Answer Questions	An examination of the students understanding of the key concepts in C and their competencies in writing C code.	1,2,3,4	30.0	Week 12

No End of Module Formal Examination

Reassessment Requirement
Coursework Only *This module is reassessed solely on the basis of re-submitted coursework. There is no repeat written examination.*

The institute reserves the right to alter the nature and timings of assessment

Module Workload

Workload: Full Time				
Workload Type	*Workload Description*	*Hours*	*Frequency*	*Average Weekly Learner Workload*
Lecture	Lecture delivering theory underpinning learning outcomes.	2.0	Every Week	2.00
Lab	Practical computer-based lab supporting learning outcomes.	2.0	Every Week	2.00
Independent & Directed Learning (Non-contact)	Independent Study.	3.0	Every Week	3.00
			Total Hours	7.00
			Total Weekly Learner Workload	7.00
			Total Weekly Contact Hours	4.00

Workload: Part Time				
Workload Type	*Workload Description*	*Hours*	*Frequency*	*Average Weekly Learner Workload*
Lecture	Lecture delivering theory underpinning learning outcomes.	2.0	Every Week	2.00
Lab	Practical computer-based lab supporting learning outcomes.	2.0	Every Week	2.00
Independent & Directed Learning (Non-contact)	Independent Study.	3.0	Every Week	3.00
			Total Hours	7.00
			Total Weekly Learner Workload	7.00
			Total Weekly Contact Hours	4.00

Module Resources

Recommended Book Resources

- **Zed A. Shaw 2015, *Learn C the Hard Way: Practical Exercises on the Computational Subjects You Keep Avoiding (Like C)*, Addison-Wesley Professional [ISBN: 9780133124385]**
- **Richard Reese 2013, *Understanding and Using C Pointers*, O'Reilly Media [ISBN: 9781449344184]**

Supplementary Book Resources

- **Brian W. Kernighan, Dennis M. Ritchie 1988, *The C programming language*, Prentice Hall [ISBN: 9780131103627]**
- **Peter Prinz and Ulla Kirch-Prinz 2003, *C Pocket Reference - C Syntax and Fundamentals*, O'Reilly Media [ISBN: 9780596004361]**
- **Daniel Kusswurm 2014, *Modern X86 Assembly Language Programming: 32-bit, 64-bit, SSE, and AVX*, Apress [ISBN: 9781484200650]**

This module does not have any article/paper resources

Other Resources

- **Website: *C Tutor - Visualize C code execution to learn C online***
  http://www.pythontutor.com/c.html
- **Website: *C reference - cppreference.com - C/C++ Reference***
  http://en.cppreference.com/w/c
- **Website: *Simple 8-bit Assembler Simulator***
  https://schweigi.github.io/assembler-sim ulator/

Module Delivered in

Programme Code	Programme	Semester	Delivery
CR_KSDEV_8	Bachelor of Science (Honours) in Software Development	4	Mandatory
CR_KDNET_8	Bachelor of Science (Honours) in Computer Systems	4	Mandatory
CR_KCOMP_7	Bachelor of Science in Software Development	4	Mandatory
CR_KCOME_6	Higher Certificate in Science in Software Development	4	Mandatory