MODULE SPECIFICATION

Part 1: Information			
Module Title	C++ Development		
Module Code	UFCFBF-15-2	Level	Level 5
For implementation from	2018-19		
UWE Credit Rating	15	ECTS Credit Rating	7.5
Faculty	Faculty of Environment & Technology	Field	Computer Science and Creative Technologies
Department	FET Dept of Computer Sci & Creative Tech		
Contributes towards			
Module type:	Standard		
Pre-requisites	Entertainment Software Development 2018-19, Programming in C 2018-19		
Excluded Combinations	None		
Co- requisites	None		
Module Entry requirements	None		

Part 2: Description

Overview: Pre-requisites: students must take one out of UFCFWA-30-1 Entertainment Software Development or UFCFF6-30-1 Programming in C

Educational Aims: See Learning Outcomes

Outline Syllabus: Rationale for using C++ in Software Development

C++ language features:

Memory allocation / deallocation

Object orientation: inheritance and polymorphism

Exception handling

Templates

Operator overloading

Delegate functions

Compiler directives

Unmanaged code:

Automatic vs dynamic memory handling

Standard Template Library

Measuring and analysing performance

Memory alignment, bit manipulation, packing, pooling

Teaching and Learning Methods: Contact time: 36 hours

Assimilation and development of knowledge: 74 hours

Exam preparation: 10 hours

Coursework preparation: 30 hours

Total study time: 150 hours

Lectures will be used to introduce relevant programming concepts whilst being practically explored within supervised studio sessions guided by tutorial tasks.

A set number of the tutorial tasks are to be completed to form individual lab logbooks.

Aside from the tutorial tasks, students will be set a small number of more challenging tasks to implement taught concepts, using supplied designs / code / libraries / SDKs where appropriate. It is expected that the majority of this work will be carried out independently, outside of taught sessions, though specific sessions will be organised to provide targeted help with these tasks prior to hand-in.

Part 3: Assessment

Formative assessment:
The tutorial tasks set for the module will be peer and tutor reviewed regularly in studio/practical sessions. Completed tasks will contribute to a logbook, which forms part of the students' portfolios. While this logbook contributes to the summative assessment, it is assessed on a pass/fail basis only, and is designed to encourage student engagement.

Summative assessment:
In addition to the tutorial tasks, a small number of more challenging tasks will be set. These tasks form the summative part of the portfolio for the module, and will be set in order of increasing complexity/weighting. The reason behind this strategy is to align assessed tasks with the topics being taught, and distribute workload for the module across the year.

A final assessment for the module will ensure detailed understanding of language mechanisms that form part of several learning outcomes but cannot easily be assessed through practical tasks.

First Sit Components	Final Assessment	Element weighting	Description
Portfolio - Component B		75 %	Portfolio of practical exercises and lab logbook
Presentation - Component A	✓	25 %	Presentation / demonstration
Resit Components	**Final Assessment**	**Element weighting**	**Description**
Portfolio - Component B		75 %	Portfolio of practical exercises
Presentation - Component A	✓	25 %	Presentation / demonstration

Part 4: Teaching and Learning Methods

Learning Outcomes

On successful completion of this module students will be able to:

	Module Learning Outcomes
MO1	Analyse the impact of using various C++ language features on the compilation process for non-trivial software projects
MO2	Demonstrate an in-depth understanding of the run-time behaviour of a C++ application, and the significance of the call-stack
MO3	Design and implement object orientated applications that make appropriate use of mechanisms such as polymorphism, templates and delegate functions
MO4	Apply their understanding of issues surrounding memory management within C++, to develop object oriented applications which avoid issues such as memory leaks, pointer errors and undefined behaviour
MO5	Recognise issues related to efficiency and organisation of memory resources within unmanaged code and apply strategies to reduce their impact on run-time performance
MO6	Discuss the role and significance of external libraries and Software Development Kits (SDKs), their relationship to C++ and their role in crossplatform development

Contact Hours

Contact Hours	
Independent Study Hours:	
Independent study/self-guided study	114
Total Independent Study Hours:	114
Scheduled Learning and Teaching Hours:	
Face-to-face learning	36

	Total Scheduled Learning and Teaching Hours:	36
	Hours to be allocated	150
	Allocated Hours	150
Reading List	*The reading list for this module can be accessed via the following link:* https://uwe.rl.talis.com/modules/ufcfbf-15-2.html	