Module Specification

C++ Development

Version: 2023-24, v2.0, 27 Jul 2023

Contents

Part 1: Information

Module title: C++ Development

Module code: UFCFBF-15-2

Level: Level 5

For implementation from: 2023-24

UWE credit rating: 15

ECTS credit rating: 7.5

Faculty: Faculty of Environment & Technology

Department: FET Dept of Computer Sci & Creative Tech

Partner institutions: None

Field: Computer Science and Creative Technologies

Module type: Module

Pre-requisites: Programming in C 2023-24

Excluded combinations: None

Co-requisites: None

Continuing professional development: No

Professional, statutory or regulatory body 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

Features: Not applicable

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

Part 3: Teaching and learning methods

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.

Module Learning outcomes: On successful completion of this module students will achieve the following 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

Hours to be allocated: 150

Contact hours:

Independent study/self-guided study = 114 hours

Face-to-face learning = 36 hours

Total = 150

Reading list: The reading list for this module can be accessed at readinglists.uwe.ac.uk via the following link https://uwe.rl.talis.com/modules/ufcfbf-15-2.html

Part 4: Assessment

Assessment strategy: 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.

Summative assessment:
At both first sit and resit, the summative assessment for this module consists of:
Portfolio.
The portfolio consists of the student logbook together with the
a small number of more challenging programming tasks. These tasks will be set in order of increasing complexity/weighting. The rationale for this strategy is to align assessed tasks with the topics being taught, and distribute workload for the module across the year.

Presentation:
The presentation/demonstration will require the students to demonstrate a detailed

understanding of language mechanisms that form part of several learning outcomes but cannot easily be assessed through practical tasks.

Assessment tasks:

Presentation (First Sit)
Description: Presentation / demonstration
Weighting: 25 %
Final assessment: Yes
Group work: No
Learning outcomes tested: MO2, MO3, MO4, MO5, MO6

Portfolio (First Sit)
Description: Portfolio of practical exercises
Weighting: 75 %
Final assessment: No
Group work: No
Learning outcomes tested: MO1, MO2, MO3, MO4, MO5

Presentation (Resit)
Description: Presentation / demonstration
Weighting: 25 %
Final assessment: Yes
Group work: No
Learning outcomes tested: MO2, MO3, MO4, MO5, MO6

Portfolio (Resit)
Description: Portfolio of practical exercises
Weighting: 75 %
Final assessment: No
Group work: No
Learning outcomes tested: MO1, MO2, MO3, MO4, MO5

Part 5: Contributes towards

This module contributes towards the following programmes of study: