Module Specification

C++ Development

Version: 2021-22, v5.0, 12 Jul 2021

Contents

Part 1: Information

Module title: C++ Development

Module code: UFCFK4-30-2

Level: Level 5

For implementation from: 2021-22

UWE credit rating: 30

ECTS credit rating: 15

Faculty: Faculty of Environment & Technology

Department: FET Dept of Computer Sci & Creative Tech

Partner institutions: None

Delivery locations: Frenchay Campus, Taylors University

Field: Computer Science and Creative Technologies

Module type: Standard

Pre-requisites: Programming in C 2021-22

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 UFCFF6-30-1 Programming in C.

Features: Not applicable

Educational aims: See Learning Outcomes

Outline syllabus: Software development process:

Unified Modelling Language diagrams: Use-case, Class and Sequence.

Source/Version Control.
Software Testing procedures.
Test-Driven Design

Software Design Principles:
Cohesion and Coupling
Polymorphism and Inheritance
Encapsulation
Validation and Verification

Rationale for using C++ in Software Development

C++ language features:
Memory allocation / deallocation.
Object orientation
Exception handling.
Templates.
Operator overloading.
Compiler directives.

Software development using C++:
Compilation process (trivial vs non-trivial projects).
Compiler representation of language features.
Runtime behaviour / call-stack behaviour.
Using basic and advanced debugging facilities (dump files, expressions, exception handling, memory examination and tracing).
Practical considerations – IDEs, libraries and SDKs.
Automatic Documentation Generation and Code Commenting

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: 72 hours
Assimilation and development of knowledge: 148 hours
Exam preparation: 20 hours
Coursework preparation: 60 hours
Total study time: 300 hours

Lectorials 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:

MO1 Analyse the impact of using various C++ language features on the compilation process for non-trivial software projects and make appropriate choices during development

MO2 Deconstruct a programming problem into a set of functional requirements and create basic object-orientated design solutions using an appropriate diagramming technique

MO3 Implement and test C++ applications based on an object-orientated design solution that make appropriate use of mechanisms such as polymorphism and templates

MO4 Apply 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 Evaluate object-oriented solutions using principles of good software design and suggest alternative approaches to improve them (where appropriate)

Hours to be allocated: 300

Contact hours:

Independent study/self-guided study = 228 hours

Face-to-face learning = 72 hours

Total = 300

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/ufcfk4-30-2.html

Part 4: Assessment

Assessment strategy: Formative assessment:
The tutorial tasks set for the module will be tutor reviewed regularly in studio/practical sessions. Some of these tutorial tasks will contribute to a logbook (these tasks will be clearly marked).

Summative assessment:
The logbook represents the first summative assessment for this module and is designed to encourage student engagement. Completed task must be demonstrated to a tutor before they can be signed-off. The number of tasks that qualify for the logbook is subject to change but is likely to be 4 or 5 tasks.

A small number of more challenging tasks will be set in the form of a small programming project. These tasks form the second assessment, and will be set in order of increasing complexity. The project will bring together many aspects of the module materials and assess many learning objectives.

The reason for this assessment strategy is to align assessed tasks with the topics being taught, and distribute workload for the module across the year.

Resit Assessment strategy:
This will focus on a single programming project which assesses all of the module learning outcomes. The project may be based on the original project or may be a new programming project. An additional reflective element will be added to the coursework to provide a way for students to reflect on any summative feedback they were given in the first attempt at their coursework and on how they have used this in their approach the resit assessment.

Assessment components:

Practical Skills Assessment - Component A (First Sit)
Description: Practical tutorial exercises (Logbook)
Weighting: 40 %
Final assessment: No
Group work: No
Learning outcomes tested: MO3, MO4, MO5, MO6

Practical Skills Assessment - Component A (First Sit)
Description: Small Programming Project
Weighting: 60 %
Final assessment: No
Group work: No
Learning outcomes tested: MO1, MO2, MO3, MO4

Practical Skills Assessment - Component A (Resit)
Description: Small Programming Project

Weighting: 100 %

Final assessment: No

Group work: No

Learning outcomes tested: MO1, MO2, MO3, MO4, MO5, MO6

Part 5: Contributes towards

This module contributes towards the following programmes of study:

Computing [Sep][FT][Frenchay][3yrs] BSc (Hons) 2020-21

Computing [Sep][SW][Frenchay][4yrs] BSc (Hons) 2020-21

Computing {Dual} [Aug][FT][Taylors][3yrs] BSc (Hons) 2020-21

Computing {Dual} [Aug][SW][Taylors][4yrs] BSc (Hons) 2020-21

Computing {Dual} [Mar][FT][Taylors][3yrs] BSc (Hons) 2020-21

Computing {Dual} [Mar][SW][Taylors][4yrs] BSc (Hons) 2020-21

Computing {Foundation} [Sep][SW][Frenchay][5yrs] BSc (Hons) 2019-20

Computing {Foundation} [Sep][FT][Frenchay][4yrs] BSc (Hons) 2019-20