Object Oriented Programming Specialisation Syllabus

Document version 1.0

Specialisation description

There are several programming paradigms in computer science. One of the most important, especially when creating large software projects, is object-oriented programming, or OOP. OOP allows the integration of data and processes on that data into discrete software modules. Learning about OOP will allow you to develop more modularised, more complex software designs. It will also let you understand better how many existing software libraries and systems have been designed.

There are also several programming languages such as Javascript, Python, Java and so on. Many languages support OOP. In this specialisation we have chosen the C++ language as our object-oriented language. C++ is a popular language which enables the development of a range of software, from low level operating system components through to high performance graphical programs. With C++, you can build any program you want and it will run fast!

This specialisation consists of five courses aiming to help you to become confident working in the object-oriented paradigm in the C++ language. During the five courses, you will work with the instructor on a single project: a crypto-currency exchange platform. Whilst building the project, you will learn about a range of programming techniques from basic control flow and input/output through to file parsing, object-oriented techniques and user interaction in the console.

Specialisation learning outcomes

Upon successful completion of this Specialisation, you will be able to:

1. Understand and explain the key principles of object-oriented programming
2. Choose appropriate basic data types to represent different data
3. Write classes with data and functions that model discrete elements of programs
4. Describe the components of an integrated development environment (IDE) and use an IDE to write, build and run programs in the C++ language
5. Use control flow, classes and input/output to construct interactive programs and algorithms of moderate complexity

Specialisation outline

The Specialisation consists of 5 courses, each of which spans two weeks.

Course 1 In this first course, you will get started with C++ by writing, building and running your first program. You will then learn about text input/output, if statements and loops by building an interactive menu system for the crypto-currency exchange platform.	**Key concepts:** • C++ edit, compile and run cycle • Text I/O • Functions **Learning outcomes:** • Write, compile and run a C++ program that prints messages to the console • Use the standard library to do text I/O in the console • Use a while loop to repeatedly receive and respond to user input • Write and call simple functions
Course 2- This course Using classes and variables to model data: the OrderBookEntry class, part 1	**Key concepts:** • Basic data types: numbers and string • Classes and data • Classes and functions **Learning outcomes:** • Select appropriate data types to represent a dataset in a C++ program • Describe how a class can be used to combine multiple pieces of data into one unit • Write a class with functions
Course 3	**Key concepts:** • Translating pseudocode to C++ • Exception handling • File I/O

File I/O, exception handling and algorithms: the CSVReader class, part 1	**Learning outcomes:** • Convert pseudocode algorithms involving iteration, logic and string processing into working C++ code • Use exception handling to gracefully recover when processing unreliable data Read text data from a file using the getline function
Course 4 Writing and testing an algorithm: taking orders and the order matching engine, part 1	**Key concepts:** • Iterating over vectors • Exception handling **Learning outcomes:** • Write functions that calculate basic statistics by iterating over vectors of objects • Use test data to evaluate the correctness of an algorithm • Use exception handling to write robust user input processing code
Course 5 Object interactions: the wallet class	**Key concepts:** • Object interactions • Modelling real world items with classes • Static and non-static functions **Learning outcomes:** • Use object interactions to achieve complex functionality through a simple command sequence • Explain how to model a familiar real-world entity as a class with data and functions Decide when it is appropriate to use static or non-static functions

Activities in the specialisation

The course is comprised of the following elements:

- Lecture videos. In each course, you will find a sequence of videos in which the example programs for the course are coded up. Further videos review the key programming techniques seen in the coding videos.
- Readings. Each course may include several suggested readings. These are a core part of your learning, and, together with the videos, will cover all of the concepts you need for this module.
- Practice Quizzes. Each course will include practice quizzes, intended for you to assess your understanding of the topics. You will be allowed unlimited attempts at each practice quiz. There is no time limit on how long you take to complete each attempt at the quiz. These quizzes do not contribute toward your final score in the class.
- Programming Activities. Each course includes programming activity worksheets. These take you through the steps you have seen in the videos, and provide code excerpts. They also contain challenges activities which challenge you develop the program beyond the functionality seen in the lecture videos.
- Code. Each course includes the C++ code written in the videos. You can use this in combination with the worksheets to ensure you have the correct code.
- Discussion Prompts. Each course includes discussion prompts. You will see the discussion prompt alongside other items in the lesson. Each prompt provides a space for you to respond. After responding, you can see and comment on your peers' responses. All prompts and responses are also accessible from the general discussion forum and the module discussion forum.
- Assessed coursework. There are two assessed activities for each course, at the end of week 1 and at the end of the course. They consist of a peer review assignment and a final summative quiz

How to pass this speciailisation

Each course has two major assessments each worth 50% of your grade:

- Peer review. This consists of a programming task that you need to submit for other peers to evaluate. You will also need to provide feedback for other 3 peers.
- End of course summative quiz. This consists of a summative quiz.

Activity	Required ?	Deadline week	Estimated time per course	% of final grade
Peer review	Yes	End of week 1	2 hours	50%
End of course summative quiz	Yes	End of week 2	1 hours	50%

Specialisation Readings

There are no specific textbooks for this Specialisation that you need to read to successfully obtain your certification. Instead, there are reading activities written by the course author, some of which involve coding exercises.
The specific URL links for the reading activities will be given in the platform, and there is no need to read beyond to recommended pages.

There will also be discussion prompts asking you to do some independent research using online sources.