2024 / 25

School of Science and Computing
+353 (0)51 302037
Eleanor.Reade@setu.ie
www.wit.ie/schools/science_computing

Ollscoil Teicneolaíochta an Oirdheiscirt

South East Technological University

Module Descriptor

Functional Programming (Computing and Mathematics)

Functional Programming (A13433)

Short Title: Functional Programming
Department: Computing and Mathematics
Credits: 5 **Level:** Advanced

Description of Module / Aims

This module will introduce the student to the functional programming paradigm. The student will learn the mathematical basis underlying functional programming and then one of the popular languages (e.g. Haskell) and see how it can be applied to a variety of domains. The student will be able to apply functional techniques in non-functional paradigms when it is helpful to do so and is able to judge when this is the case.

Programmes

		stage/semester/status
PROG-0094	BSc (Hons) in Applied Computing (International) (WD_KACCM_BI)	4 / 8 / M
PROG-0094	BSc (Hons) in Applied Computing (WD_KACCM_B)	4 / 8 / M
PROG-0094	BSc (Hons) in Applied Computing (WD_KCOMP_B)	4 / 8 / M
PROG-0094	BSc (Hons) in Computer Forensics and Security (WD_KCOFO_B)	4 / 8 / E
PROG-0094	BSc (Hons) in Computer Science (WD_KCMSC_B)	4 / 8 / M

Indicative Content

- Introduction to functional programming – why and when should this paradigm be used
- Types and classes, functions
- Lambda expressions - untyped and typed
- Recursive functions, higher-order functions
- Use of a functional language (e.g Haskell)
- Effect-free programming
- Processing structured data (e.g. trees) via functions
- Use of functional programming techniques in other, non-functional programming languages (e.g. Java)
- Current trends in functional programming

Learning Outcomes

On successful completion of this module, a student will be able to:

1. Construct simple and more complex programs in a functional programming language (e.g. Haskell).
2. Construct basic constructs of a functional programming languageusing untyped lambda calculus.
3. Construct programs using typed lambda calculus.
4. Compare and contrast the procedural/functional approach and the object-oriented approach.
5. Evaluate and reason about variables and lexical scope in a program using function closures.
6. Construct basic algorithms that avoid assigning to mutable state or considering reference equality.
7. Construct and use iterators and other operations on aggregates, including operations that take functions as arguments, in multiple programming languages, selecting the most natural idioms for each language.
8. Write functional code in non-functional languages (e.g. Java)

Learning and Teaching Methods

- The material will be delivered using lectures and practicals. The lectures will introduce the theory and the practicals will be used to implement the theoretical concepts via programming examples.

Learning Modes

Learning Type	F/T Hours	P/T Hours
Lecture	24	
Practical	24	
Independent Learning	87	

Assessment Methods

	Weighting	Outcomes Assessed
Final Written Examination	50%	1,2,3,4,5,6
Continuous Assessment	50%	
Tutorial/Problem Sheets	10%	1,4,5,6,7
Project	40%	1,5,6,8

Assessment Criteria

<40%: Inability to write correct programs (small and medium) using the functional paradigm.

40%–49%: Able to write correct programs (small and medium) using the functional paradigm.

50%–59%: Above and able to reason correctly about the programs. Can write functional code in non-functional languages.

60%–69%: All of the above and writes elegant and clear functional code.

70%–100%: All of the above to an excellent level and demonstrates an enthusiasm for the functional approach.

Supplementary Material(s)

- "Java Software." https://www.oracle.com/java/index.html. https://www.oracle.com/java/index.html
- Hutton, G. *Programming in Haskell*. 1. England: Cambridge University Press, 2007.
- Khan, A. *Grokking Functional Programming*. 1. America: Manning Publications, 2016.

Requested Resources

- Computer Lab: BYOD Lab