MODULE DESCRIPTION FORM

Module Information					
Module Title	**Programming Basics**			**Module Delivery**	
Module Type	**C**			☒ **Theory** ☒ **Lecture** ☒ **Lab** ☒ **Tutorial** ☒ **Practical** ☐ **Seminar**	
Module Code	**AI1102**				
ECTS Credits	**9**				
SWL (hr/sem)	**225**				
Module Level	1		**Semester of Delivery**	1	
Administering Department	Artificial Intelligence		**College**	Computer Science and Information Technology	
Module Leader	Abdulkareem Zwaen		**e-mail**	abdulkareem.zouine@uowa.edu.iq	
Module Leader's Acad. Title	Lecturer		**Module Leader's Qualification**	Ph.D.	
Module Tutor	Abdulkareem Zwaen		**e-mail**	abdulkareem.zouine@uowa.edu.iq	
Peer Reviewer Name	Ali Mahmoud Ali		**e-mail**	ali.mahmoud@uowa.edu.iq	
Scientific Committee Approval Date	01/11/2025		**Version Number**	1.0	

Relation with other Modules			
Prerequisite module	None	**Semester**	-
Co-requisites module	None	**Semester**	-

Department Head Approval **Dean of the College Approval**

Module Aims, Learning Outcomes and Indicative Contents	
Module Objectives	The objectives of this module are to: 1. Introduce students to fundamental programming concepts and problem-solving techniques using the C++ programming language. 2. Develop students' understanding of programming paradigms, with emphasis on the imperative approach and foundational functional concepts. 3. Enable students to understand and apply core C++ language features, including variables, data types, operators, control structures, and loops. 4. Build a foundational understanding of object-oriented programming concepts and their implementation in C++. 5. Strengthen students' ability to analyze computational problems and design efficient, structured C++ solutions. 6. Prepare students to develop, test, and debug basic C++ applications using standard libraries and best programming practices.
Module Learning Outcomes	By the end of this module, students will be able to: 1. Explain the fundamental concepts and terminology associated with programming languages, with emphasis on imperative and functional programming paradigms. 2. Apply core C++ programming constructs, including variables, data types, operators, control structures, and loops, to develop correct programs. 3. Use standard C++ libraries to perform input/output operations and basic mathematical computations. 4. Design and implement object-oriented programs using key OOP principles such as encapsulation and modularity. 5. Analyze and solve computational problems by applying structured and functional programming techniques. 6. Develop, test, and debug C++ programs using appropriate programming practices and logical reasoning.
Indicative Contents	• Introduction to algorithms and problem-solving techniques • Overview of programming languages and programming paradigms • Introduction to C++ programming environment and syntax • Variables, data types, and user input/output in C++ • Operators, expressions, and logical conditions • Control structures: if, switch, and decision making

	• Looping structures: while, do-while, and for loops • Strings, Boolean values, and mathematical operations in C++ • C++ standard libraries and basic program organization • Object-oriented programming concepts in C++ • Program testing, debugging, and code optimization • Development of simple C++ applications

Learning and Teaching Strategies	
Strategies	1. **Lectures** Core programming concepts, syntax, and paradigms are introduced through structured lectures, providing students with a solid theoretical foundation in C++ programming and problem-solving techniques. 2. **Guided Practical Sessions** Hands-on programming exercises are used to reinforce lecture material, allowing students to practice coding, experiment with language constructs, and gain confidence in writing C++ programs. 3. **Problem-Based Learning** Students are engaged in solving progressively complex programming problems that promote analytical thinking, algorithmic reasoning, and the application of imperative and functional programming concepts. 4. **Demonstrations and Code Walkthroughs** Live demonstrations and step-by-step code walkthroughs are employed to illustrate program logic, control flow, and debugging techniques. 5. **Independent Learning** Students are encouraged to explore additional programming examples, documentation, and online resources to strengthen self-learning skills and technical competence. 6. **Continuous Feedback and Assessment** Regular quizzes, assignments, and practical tasks provide formative feedback, enabling students to identify strengths and areas for improvement throughout the semester. 7. **Revision and Exam Preparation Sessions** Dedicated sessions are conducted to review key concepts, clarify common misconceptions, and prepare students for midterm and final examinations.

Student Workload (SWL)			
Structured SWL (h/sem)	90	**Structured SWL (h/w)**	6
Unstructured SWL (h/sem)	135	**Unstructured SWL (h/w)**	9
Total SWL (h/sem)	**225**		

Module Evaluation					
		Time/Number	**Weight (Marks)**	**Week Due**	**Relevant Learning Outcome**
Formative assessment	**Quizzes**	5	10% (10)	2,4,6,8,11	LO #1, LO #2, LO #10, LO #11
	Assignments	5	10% (10)	2,3,5,9,12	LO #3, LO #4, LO #6, LO #7
	Projects / Lab.	10	10% (10)	Continuous	All Learning Outcomes
	Report	1	10% (10)	13	LO #5, LO #8, LO #10
Summative assessment	**Midterm Exam**	2hr	10% (10)	7	LO #1 – LO #7
	Final Exam	3hr	50% (50)	16	All Learning Outcomes
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus)	
	Material Covered
Week 1	**Algorithms**
Week 2	**Introduction to** programming **languages and C++**
Week 3	**Variables**
Week 4	**C++ Libraries**
Week 5	**C++ User Input**
Week 6	**C++ Operators**
Week 7	Mid-term Exam
Week 8	**C++ Strings & C++ Math**
Week 9	**C++ Booleans**
Week 10	**If condition**
Week 11	**Switch condition**
Week 12	**While loop**
Week 13	**Do-while loop**
Week 14	**For loop**
Week 15	**C++ Break and Continue**
Week 16	**Preparatory week before the final Exam**

Delivery Plan (Weekly Lab. Syllabus)	
	Material Covered
Week 1	**C++ Libraries**
Week 2	**C++ User Input**
Week 3	**C++ Operators**
Week 4	**If condition**
Week 5	**Switch condition**
Week 6	**While loop**
Week 7	**Do-while loop**
Week 8	**For loop**
Week 9	**C++ Break and Continue**

Learning and Teaching Resources		
	Text	**Available in the Library?**
Required Texts	The C++ Programming Language (4th Edition) by Bjarne Stroustrup	No
Recommended Texts	**Stroustrup, B.** (2013). *The C++ Programming Language* (4th ed.). **Deitel, P., & Deitel, H.** (2020). *C++ How to Program* (10th ed.).	
Websites	https://www.learncpp.com https://www.w3schools.com/CPP/default.asp	

Grading Scheme				
Group	**Grade**	Mark	**Marks %**	**Definition**
Success Group (50 - 100)	**A -** Excellent	Excellent	90 - 100	Outstanding Performance
	B - Very Good	Very Good	80 - 89	Above average with some errors
	C - Good	Good	70 - 79	Sound work with notable errors
	D - Satisfactory	Fair / Average	60 - 69	Fair but with major shortcomings
	E - Sufficient	Pass / Acceptable	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX –** Fail	Fail (Pending)	(45-49)	More work required but credit awarded
	F – Fail	Fail	(0-44)	Considerable amount of work required
Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.				