MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية					
Module Title	**Programming Fundamentals**			**Module Delivery**	
Module Type	Core			☒ **Theory** ☒ **Lecture** ☒ **Lab** ☒ **Tutorial** ☐ **Practical** ☐ **Seminar**	
Module Code	TUCS110				
ECTS Credits	8				
SWL (hr/sem)	**200**				
Module Level	1		**Semester of Delivery**	1st	
Administering Department	Computer Science		**College**	CCSM	
Module Leader	Mohanad Hatem Ramadhan		**e-mail**	Mohanad.H.Ramadhan@tu.edu.iq	
Module Leader's Acad. Title	Assistant Lecturer		**Module Leader's Qualification**	master	
Module Tutor	Yahya Laith Khalil		**e-mail**		
Peer Reviewer Name	Mahammed Aktham		**e-mail**		
Scientific Committee Approval Date	07/06/2023		**Version Number**	1.0	

Relation with other Modules العلاقة مع المواد الدراسية الأخرى			
Prerequisite module	None	**Semester**	
Co-requisites module	Advanced Programming	**Semester**	2

Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Aims أهداف المادة الدراسية	1. To introduce students to the fundamental principles and concepts of programming. 2. To familiarize students with the syntax and structure of the C++ programming language. 3. To develop students' problem-solving skills and algorithmic thinking. 4. To enable students to design, implement, and test programs using C++ to solve computational problems. 5. To provide students with hands-on experience in programming through practical exercises, assignments, and projects. 6. To promote the use of modular programming techniques for creating reusable and maintainable code. 7. To enhance students' ability to debug and troubleshoot programs effectively. 8. To develop students' communication skills in expressing programming concepts and solutions clearly and effectively. 9. To prepare students for advanced programming courses and real-world software development scenarios.
Module Learning Outcomes مخرجات التعلم للمادة الدراسية	Upon successful completion of this module, students should be able to: 1. Demonstrate a solid understanding of the fundamental principles and concepts of programming. 2. Utilize the syntax and structure of the C++ programming language to write well-structured and efficient code. 3. Apply problem-solving skills and algorithmic thinking to develop solutions for a variety of computational problems. 4. Design, implement, and test programs using C++ to solve specific tasks and challenges. 5. Utilize modular programming techniques to create reusable and maintainable code. 6. Debug and troubleshoot programs effectively using appropriate debugging techniques and tools. 7. Collaborate and work effectively in teams to complete programming projects. 8. Communicate programming concepts, solutions, and ideas clearly and effectively, both orally and in written form. 9. Demonstrate a readiness to progress to more advanced programming courses or pursue a career in software development.
Indicative Contents المحتويات الإرشادية	

1. Introduction to Computer Science:
   - Overview of computer science as a discipline
   - Key concepts and principles in computer science
   - Role of programming in computer science

2. Introduction to Computers, Binary System, and Information Representation:
   - Basics of computer architecture and components
   - Understanding the binary system and its significance in computing
   - Conversion between binary and decimal.
   - Representation of different data types in computers
   - ASCII and Unicode for character encoding

3. Algorithm Design and Problem Solving:
   - Understanding algorithms and problem-solving strategies
   - Analyzing problem requirements and designing algorithmic solutions
   - Time and space complexity analysis
   - Representing algorithms with Pseudocode and Flowcharts:
      - Using pseudocode as a high-level representation of algorithms
      - Writing pseudocode to describe the logic and steps of an algorithm
      - Understanding flowcharts as visual representations of algorithms
      - Basic flowchart symbols and their meanings
      - Creating flowcharts to represent the flow of control in algorithms

4. Introduction to C++:
   - History and features of the C++ programming language
   - Setting up a C++ development environment
   - Basic syntax and structure of C++ programs

5. Variables and Data Types:
   - Declaring and initializing variables
   - Fundamental data types (integers, floating-point numbers, characters)
   - Working with constants and literals

6. Operators and Expressions:
   - Arithmetic operators
   - Assignment operators
   - Comparison operators
   - Logical operators

7. Control Structures:
   - Decision-making with if-else statements
   - Switch statements for multiple choices
   - Repetition with loops (while, do-while, for)
   - Handling user input and validation

	8. Functions: - Function declaration and definition - Parameters and argument passing - Return values and function overloading - Scope and lifetime of variables

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	1. Lectures: The instructor will deliver lectures to introduce and explain programming concepts, C++ syntax, and problem-solving techniques. This will provide students with a solid theoretical foundation. 2. Interactive Discussions: Engaging students in interactive discussions allows them to ask questions, seek clarifications, and participate actively in the learning process. Discussions can include reviewing code examples, discussing programming best practices, and exploring real-world applications of programming concepts. 3. Laboratory Sessions: Laboratory sessions are dedicated practical sessions where students apply the concepts learned in lectures to hands-on programming exercises. Key strategies for the laboratory sessions include: a. Programming Exercises: Students will work on programming exercises and projects in the laboratory, providing them with practical experience in coding and problem-solving. b. Guided Practice: Lab instructors or teaching assistants will be available to provide guidance, assistance, and immediate feedback on students' code. They can help students debug their programs, identify errors, and improve their coding skills. c. Collaboration and Peer Learning: Students can collaborate with their peers in the laboratory, fostering teamwork and enabling knowledge sharing. Working together on programming tasks promotes discussions, problem-solving, and peer learning.

	d. Equipment and Resource Access: The laboratory should provide access to computers, necessary software tools, programming references, and relevant online resources. This ensures that students have the necessary resources to complete their lab exercises and assignments effectively. 4. Programming Assignments: Assignments will be given to students to reinforce their understanding of programming concepts and encourage independent problem-solving. These assignments may involve implementing algorithms, designing software systems, or developing small-scale projects using C++. 5. Code Reviews and Feedback: The instructor will provide feedback on students' code, reviewing their solutions, and offering suggestions for improvement. This feedback will help students enhance their coding skills and adhere to best practices. 6. Office Hours and Individual Support: The instructor should be available for individual consultations and provide support to students who need additional help or guidance in understanding programming concepts or completing assignments.

Student Workload (SWL)

الحمل الدراسي للطالب محسوب لـ ١٥ أسبوعا

Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	92	**Structured SWL (h/w)** الحمل الدراسي المنتظم للطالب أسبوعيا	6.13
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	108	**Unstructured SWL (h/w)** الحمل الدراسي غير المنتظم للطالب أسبوعيا	7.2
Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	200		

Module Evaluation

تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	**Quizzes**	2	10% (10)	5, 11	#LO 1-3, #LO 5-8
	Assignments	2	10% (10)	7, 12	#LO 3-5, #LO 5-8
	Projects	1	10% (10)	continuous	
	Report	1	10% (10)	14	#LO 1-8
Summative assessment	**Midterm Exam**	2 hr	10% (10)	11	#LO 1-7
	Final Exam	2 hr	50% (50)	16	All

Total assessment	100% (100 Marks)		

Delivery Plan (Weekly Syllabus) المنهاج الاسبوعي النظري	
Week No.	**Material Covered**
Week 1	Introduction to Computer Science, Computers Components, Binary and Info Representation
Week 2	Algorithms Design and Writing pseudocode
Week 3	Algorithms design and Drawing Flowchart
Week 4	Introduction to Programming Languages (History, Categories, Main Differences) and, Introduction to Programming in C++ (Program Structure and Coding Environment)
Week 5	Variables, Datatypes, Output, and Input
Week 6	Operations (Arithmetic and Assignment) and Math Functions
Week 7	Operations (Comparison and Logical)
Week 8	Flow Control (if – else)
Week 9	Flow Control (switch – case)
Week 10	Loops (counter and cumulative variables)
Week 11	Uncountable Loops
Week 12	Nested Loops
Week 13	Functions
Week 14	building a TikTacToe Game
Week 15	Reviewing Students' Projects

Delivery Plan (Weekly Lab. Syllabus): المنهاج الاسبوعي للمختبر:	
Week No.	**Material Covered**
Week 1	**Using Operating System, Creating Files and Folders, Writing Text)**
Week 2	**Difference among (Text Editor, Word Processor, Code Editor and IDE)**
Week 3	**Drawing (Darg and drop) Flowcharts**
Week 4	**Installing C++ coding environment and running Hello World program**
Week 5	**Running Examples on Variables, Datatypes, Output, and Input**
Week 6	**Running Examples on Operations (Arithmetic and Assignment) and Math Functions**
Week 7	**Running Examples on Operations (Comparison and Logical)**

Week 8	**Running Examples on Flow Control (if – else)**
Week 9	**Running Examples on Flow Control (switch – case)**
Week 10	**Running Examples on Loops (counter and cumulative variables)**
Week 11	**Running Examples on Uncountable Loops**
Week 12	**Running Examples on Nested Loops**
Week 13	**Running Examples on Functions**
Week 14	**Fixing problems in students' projects**
Week 15	**Applying instructor's feedback on students' projects**

Learning and Teaching Resources
مصادر التعلم والتدريس

	Text	Available in the Library?
Required Texts	Stroustrup, Bjarne - Programming_ principles and practice using C++-Addison-Wesley (2015)	Yes
Recommended Texts	Olsson, Mikael - C++20 Quick syntax reference: a pocket guide to the language, apis, and library	No
Websites		

Grading Scheme
مخطط الدرجات

Group	Grade	التقدير	Marks (%)	Definition
Success Group (50 - 100)	**A** - Excellent	امتياز	90 - 100	Outstanding Performance
	B - Very Good	جيد جدا	80 - 89	Above average with some errors
	C - Good	جيد	70 - 79	Sound work with notable errors
	D - Satisfactory	متوسط	60 - 69	Fair but with major shortcomings
	E - Sufficient	مقبول	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX** – Fail	راسب (قيد المعالجة)	(45-49)	More work required but credit awarded
	F – Fail	راسب	(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.