MODULE DESCRIPTION FORM

Module Information			
Module Title	**Object-oriented programming I**	**Module Delivery**	
Module Type	**Core**	☒ **Lecture** ☒ **Practical**	
Module Code	**IT2112**		
ECTS Credits	**6**		
SWL (hr/sem)	**150**		
Module Level	UG2	**Semester of Delivery**	1
Administering Department	Information Technology	**College**	College of Science
Module Leader	Mohsin Hassan Hussein Abbas	**e-mail**	mohsin.ha@uowa.edu.iq
Module Leader's Acad. Title	Asst. Professor	**Module Leader's Qualification**	Ph.D.
Module Tutor	Mohsen Hassan Hussein Abbas	**e-mail**	mohsin.ha@uowa.edu.iq
Peer Reviewer Name	Asst. Prof Haider Mohammed Ali	**e-mail**	hayder.alghanami@uowa.edu.iq
Scientific Committee Approval Date	2024-09-17	**Version Number**	V1.0

Relation with other Modules			
Pre-requisite module	Programming Fundamentals 2	**Semester**	2
Co-requisites module	Programming Fundamentals 2	**Semester**	2

أ.م.د شيماء حسين نوفل
٢٠٢٥/٢٠٢٤

جامعة وارث الانبياء
عمادة كلية العلوم

جامعة وارث الانبياء
كلية العلوم
قسم تكنولوجيا المعلومات

أ.م.د شيماء حسين نوفل
٢٠٢٥/٢٠٢٤

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

Module Aims, Learning Outcomes and Indicative Contents

Module Aims

1. Provide a sound knowledge of the underlying principles and experience in the practical application of this course is essential for any information technology specialist.
2. extend students with procedural programming knowledge and skills in the object-oriented paradigm and builds experience with interpreted languages to introduce compiled languages.
3. In addition to further shaping a solid development methodology, the course prepares students for continued investigation into advanced programming topics.
4. develop a wide range of software solutions for real-world scenarios.

Module Learning Outcomes

On completion of this course students will be able to:

1. identify and demonstrate an understanding of the hardware of a computer;
2. comprehend what programming is and what a programming language does;
3. know about the evolution of C++;
4. identify and design suitable classes and class hierarchies and code class implementations in C++;
5. design and develop C++ programs using classes and class libraries;
6. apply the principles of information hiding using C++ facilities for private and protected class attributes;
7. employ C++ facilities for dynamic storage;
8. employ C++ facilities such as operator overloading, pointers, and references;
9. develop programs using the C++ Standard for real-world.

Indicative Contents

Topics

	l scription	Weighting (75%)
1.	Overview of Object Oriented Programming, C++ or Python Basics	5.00
2.	Control flow	5.00
3.	Function Basics	5.00
4.	Parameters and Overloading	10.00
5.	Arrays and Structures	10.00
6.	Objects and Classes	10.00
7.	Constructors and Destructors	5.00
8.	Operator Overloading	5.00
9.	Friends and References	10.00
10.	Strings and Pointer	5.00
11.	Separate Compilation and Namespace	5.00

Learning and Teaching Strategies	
Strategies	**Overview Strategies** Object-oriented software development has become a standard methodology throughout the software engineering discipline. Therefore, a solid grasp of object-oriented programming is essential for any information technology specialist. While there are a variety of object-oriented programming languages available, C++ or Python are the most widely used in this course. This course extends the student's basic procedural design and programming knowledge and skills into the object-oriented paradigm and builds on previous experience with interpreted languages to introduce compiled languages. In addition to further shaping a solid development methodology, the course prepares students for continued investigation into advanced programming topics. The students will be expected to learn and apply the basic concepts of object oriented design and programming through giving lectures, practical exercises within the laboratories, assignments about some specific topics, and small projects. Key software engineering principles such as decomposition and component re-use will also be emphasized.

Student Workload (SWL)			
Structured SWL (h/sem)	75	**Structured SWL (h/w)**	5
Unstructured SWL (h/sem)	72	**Unstructured SWL (h/w)**	5
Student workload expectations (SWL &USWL) To do well in this subject, students are expected to commit approximately 10 hours per week including class contact hours, independent study, and all assessment tasks. If you are undertaking additional activities, the weekly workload hours may vary.			
Total SWL (h/sem)	147 + 3 final = 150		

Module Evaluation

		Time/ Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	**Quizzes**	5	10% (8)	3, ,6,9,11, 13	1,2,3,4
	OnSite Assignments	5	10% (5)	3,5,8,10,11	All
	HomeWork	5	10% (7)	2,5,8,10,12	All
	Project	1	10% (10)	12	All
	Labs	5	10% (15)	3,5,7,9,11	All
Summative assessment	Midterm Exam	2hr	10% (10)	7	
	Final Exam	3hr	50% (50)	16	
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus)

	Material Covered	Weighting (30+5=35%)
Week 1	The fundamental concepts of programming, including procedural and object-oriented programming will be introduced. Also, consider the basic principles behind object-oriented programming techniques, including objects, classes, inheritance, and polymorphism. Then you will get started in programming environment by applying what you have learned.	2
Week 2	Introduction about the basic logic components used in programs that called control structures. It includes sequence structure, a selection structure, and loop structure, with examples.	2
Week 3	Learn about function features, including passing arguments, returning values, prototypes, and recursion, with examples.	2
Week 4	Present specific features of functions, such as function overloading and reference parameters, with examples.	2
Week 5	Introduce arrays concept with a specific element in an array, index, memory locations, the lowest address, highest address, arrays dimensions, arrays and pointers, with examples	2
Week 6	Overview about structures, structure declaration forms, and structure members, with examples.	2
Week 7	**Mid Term Exam Revision**	2
Week 8	Introduction about objects and classes, class declaration, Object declaration, with examples.	2
Week 9	Understanding constructors and destructors, constructors and destructors declaration with examples.	2
Week 10	Learn about overloading operators, operator declaration, unary operators, binary operators, and operator arguments.	2
Week 11	Learn what a friend is, Declare a friend function, and Examine the benefits of Use a friend function to access data from two classes, with examples.	2
Week 12	Understanding the three ways that a reference can be used: as a function parameter, as a function return value, or as a stand-alone reference, with examples.	2
Week 13	Learn about the string class , Learn about pointers, string and pointers declaration, with examples.	2

Week 14	Describes namespaces and several other advanced features, including conversion functions, explicit constructors, const and volatile member functions, the asm keyword, and linkage specifications, with examples.	2
Week 15	Students course workload evaluation.	2
Week 16	**Prepare to the final Exam**	3

Delivery Plan (Weekly Lab. Syllabus)

	Material Covered	**Weighting (45%)**
Week 1 - Lab 1	- Prepare OOP environment, overview about unified modeling language (UML) diagram. - Access to a standard C++ or Python compiler - Linux g++ compiler and its equivalent MinGW running under windows.	3
Week 2 - Lab 2	- learn how to create a main () function, work with variables and constants, and create comments. - learn how to produce output and process input with Python or C++, and how to create first objects.	3
Week 3 - Lab 3	- Basic Functions and Pointers, - Implement recursion function, - Understand the manipulation on pointers.	3
Week 4 – Lab 4	- Understand function call by value method of parameter passing - Understand Pass parameters by reference method	3
Week 5 – Lab 5	- Study the use of structures - Understand array processing in C++ or Python - Understand heterogeneous data types	3
Week 6 – Lab 6	- Introduction to Classes and Objects	3
Week 7 – Lab 7	- Labs exam1 with evaluation	3
Week 8 – Lab 8	- Access Specifiers, Constructors and Destructors	3
Week 9 – Lab 9	- Constructor Overloading and Copy Constructors	3
Week 10 – Lab 10	- Introduction to Operator Overloading	3
Week 11 – Lab 11	- Friend Functions and Friend Classes	3
Week 12 – Lab 12	- Study string class and pointer concepts - Understand reference to an object concept	3

Week 13 – Lab 13	- Labs exam2 with evaluation	3
Week 14 – Lab 14	- Study the use of storage specifiers - Familiarise with global and static variables - Understanding separate Compilation and Namespace	3
Week 15 – Lab 15	- OOP project Implementation with discussion for each student	3

Learning and Teaching Resources

	Text	**Available in the Library?**
Required Texts	1. Malik, D.S 2018, *C++ Programming: Program Design Including Data Structures*, 8th edn, Cengage. (ISBN 978-1-337-11756-2.) 2. OOP – Learn Object Oriented Thinking and Programming, ISBN-10: 8090466184, Tomas Bruckner, 2013. 3. The student must have access to a standard C++ compiler. The only supported compilers are the Linux g++ compiler and its equivalent MinGW running under Windows.	No
Recommended Texts	4. Object-Oriented Programming Using C++ Fourth Edition by Joyce Farrell	No
Websites		

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.