Module (Advanced Programming) Catalogue

2022-2023

College/ Institute	**Erbil Technology College**	
Department	**Information & Communication Technology Engineering (ICTE)**	
Module Name	**Advanced Programming**	
Module Code	**ADP603**	
Degree	**Technical Diploma** ☒ **Bachel** ☒ **High Diploma** ☐ **Maste** ☐ **Pl** ☐	
Semester	**Sixth Semester**	
Qualification	**Master's degree in computer science**	
Scientific Title	**Assistant Lecturer**	
ECTS (Credits)	**6**	
Module type	**Prerequisite** ☐ **Core** ☒ **Assist.** ☐	
Weekly hours	**4**	
Weekly hours (Theory)	**(2) hr Class**	**(67)Total hrs Workload**
Weekly hours (Practical)	**(3) hr Class**	**(108)Total hrs Workload**
Number of Weeks	**15**	
Lecturer (Theory)	**Chiman haidar salh**	
E-Mail & Mobile NO.	**Chiman.salh@epu.edu.iq**	
Lecturer (Practical)	**Naila faiq othman**	
E-Mail & Mobile NO.	**Naila.faiq.othman@gmail.com**	
Websites		

Course Book

Course Description	This course could be considered as the main course where basic concepts will be explained. These concepts are frequently repeated in other programs. Electronic devices are designed and operated by one of the operating systems which developing by one of the programming languages such as C, C++, C#, Visual Basic, and so on. Therefore any operating system or any application program that you want to create for the purpose of executing any specific job must be made by one of those programming languages classified as a high-level language. C++ is an important programming language to create an application program, Therefore you need to know or learn the basic concepts in the C++ language to create this kind of program. Then you have to have a good background in C++ and Object Oriented Programming because this will be your fundamental step and help you to learn easily. Finally, programming is considered the backbone of computer science.
Course objectives	• Write easier-to-read and easier-to-code using operator overloading • Perform file input-output and describe the concepts of persistent objects • Understand issues in using virtual base classes and multiple inheritances • Better utilize the features of inheritance and polymorphism in program design • Understand how templates can reduce code replication when used properly • To be comfortable using and learning different programming languages (C, C++, Java, Perl) and choosing the appropriate one for a given task and use tools and write programs to assist in developing programs. • Describe when/how to use run-time type identification, mutable keywords/other language features
Student's obligation	The Students should be attendant in class at less than 1:30 hours during lecturing and to pass this course should be fulfilled the following requirements: 1. The student has to submit almost all assignments, essays and reports and also. 2. The student must be passing the exams and quizzes which have been done during study year. 3. Students attending classes regularly. 4. Group work. 5.Doing assignments. 6.Class activities.
Required Learning Materials	The ways that we are using in our teaching for this course are: 1. Data show. 2. White Board. 3. Word Documents. 4. Notebook. 5. Group activity 6. Computer Lab

	Task		Weight (Marks)	Due Week	Relevant Learning Outcome
Evaluation	Paper Review		0		
	Assignments	Homework	5		
		Class Activity	2		
		Report	5		
		Seminar	5		
		Essay	0		
		Project	0		
	Quiz		8		
	Lab.		10		
	Midterm Exam		25		
	Final Exam		40		
	Total		100		
Specific learning outcome:	• This course will teach the student how to solve problems arising from subtleties of the C++ language as well as techniques for improving performance and efficiency. Students are invited to bring their current ideas and question to the classroom for discussion. • On successful completion of the module, students should be able to demonstrate a hands-on understanding of advanced programming topics				
Course References:	**1. Internet recourse**				

Course topics (Theory)	Week	Learning Outcome
Introduction & Refresher of OOP	1	Review of C++ and classes
C++ Pointer & Polymorphs	2	Pointer
C++ Pointer & Polymorphs	3	Polymorphs
C++ Overloading (Function and Operator)	4	Overloading
Error Handling	5	Error Handling
C++Working with files (IO Streams)	6	I/O Streams
C++Circular	7	C++Circular

Dynamic containers (STL Components)	8	Dynamic containers
C++ Lambda	9	C++ Lambda
Multitasking in C++	10	Multitasking in C++
C++ Header Files (Include file)	11	C++ Header Files
Domains(Namespaces)	12	Namespaces
Definition of C++ array list	13	Array list
Introduction to C++ Union& C++ Static	14	Union & Static
Introduction to Queue in C++& Introduction to C++ Queue	15	I Queue in C++& C++ Queue
Practical Topics	**Week**	**Learning Outcome**
Introduction & Refresher of OOP	1	Review of C++ and classes
C++ Pointer & Polymorphs	2	Pointer
C++ Pointer & Polymorphs	3	Polymorphs
C++ Overloading (Function and Operator)	4	Overloading
Error Handling	5	Error Handling
C++Working with files (IO Streams)	6	I/O Streams
C++Circular	7	C++Circular
Dynamic containers (STL Components)	8	Dynamic containers
C++ Lambda	9	C++ Lambda
Multitasking in C++	10	Multitasking in C++
C++ Header Files (Include file)	11	C++ Header Files
Domains(Namespaces)	12	Namespaces

Definition of C++ array list	13	Array list
Introduction to C++ Union & C++ Static	14	Union & Static
Introduction to Queue in C++& Introduction to C++ Queue	15	I Queue in C++& C++ Queue

Questions Example Design

Q1// The following example is an implementation of an ideal generic Function Where you will learn from it how to build a function to which you can pass a parameter of no specific type.

Solution:

```
template < typename E >

void printArray ( E arr [ ] , int length )

{

   for ( int i=0; i < length; i++ )

   {

      cout << arr [ i ] << " " ;

   }

      cout << endl;

}

int main ()  {

   int arr1 [] = { 1, 2, 3, 4, 5 } ;

   char arr2 [] = { 'a' , 'b' , 'c' , 'd' , 'e' } ;

   string arr3 [] = { "I'm" , "learning" , "C++" , "in" , "harmash.com" } ;

   printArray < int >( arr1, 5 ) ;

   printArray < char >( arr2, 5 ) ;

   printArray < string >( arr3, 5 ) ;

   return 0; }
```

Q2// How to define a function do **throw** more than one of their kind **int** in addition to how to summon it.

Solution: In the following example, we define a function called compare Ages ( )When calling her, we pass two numbers to her, the first number is the age of the son and the second is the age of his mother.

The function will compare the age of the son with the age of his mother and return the difference between them, provided that the numbers we pass to it are logically acceptable numbers, otherwise it will throw an exception.

```
int compareAges ( int sonAge, int momAge )

{

   // If the son's age is greater than or equal to the mother's age, exception number 1 will be thrown

   if ( sonAge > = momAge )

      throw 1;

   // If the child's age is less than or equal to zero, an exception number 2 will be thrown

   else if ( sonAge < = 0 )

      throw 2;

   // If the mother's age is less than or equal to zero, an exception number 3 will be thrown

   else if ( momAge < = 0 )

      throw 3;

   // In case the difference between the age of the mother and the son is not at least 12 years old, exception number 4 will be thrown

   else if ( momAge - sonAge < 12 )

      throw 4;

   // If no exception is thrown, the age difference will be returned

   return momAge - sonAge;

}

// main() Here we define the function

int main ()

{
```

```
    // And pass two numbers to it representing the age of a command and the age of its child to see if the entered ages are acceptable or not compare Ages() Here we called the function

    try {

        compareAges ( 26, 24 ) ;  }

    // e The exception to be thrown will be a text (a string of characters) and these characters will be passed as the value of the variable

    catch ( int e )  {

        switch ( e )  {

            case 1:

                cout << "Error: Son's age can't be less than his mom! \n" ;

                break ;

            case 2:

                cout << "Error: Son's age can't be less than or equal zero \n" ;

                break ;

            case 3:

                cout << "Error: Mom's age can't be less than or equal zero \n" ;

                break ;

            case 4:

                cout << "Error: Mom's age should be 12 years older than son's age \n" ;

                break ;   } }

        cout << "The program is still working properly :)" ;

    return 0;  }
```

Q3// In the following example, you will learn how to operate more than one porridge at the same time

Solution: In the following example, we define a function called foo ( )When called it prints the sentence "foo is executed.."5 times.

Then we define another function called bar ( )When called it prints the sentence "bar is executed.."5 times too.

Finally, we create two class object threads, The first executes the function foo ( )The second one executes the function bar ( ).

```
#include <iostream>

#include <thread>

#include <chrono>

using namespace std;

void foo ( )  {

    for ( int i=1; i < =5; i++ )    {

        cout << "foo is executed.. \n" ;

        this_thread:: sleep_for ( chrono:: seconds ( 1 ) ) ;  } }

voidbar ( ) {

    for ( int i=0; i < 5; i++ )  {

        cout << "bar is executed.. \n" ;

        this_thread:: sleep_for ( chrono:: seconds ( 1 ) ) ;    } }

int main ( )  {

    thread t1 ( foo ) ;

    thread t2 ( bar ) ;

    t1. join () ;

    t2. join () ;

  cout << "All threads are stopped!" ;

    return 0; }
```

Extra notes:

Please take this note into consideration:
Making the topics covered in the semester compatible with the reality of the educational process due to the national and religious holidays or other reasons. For instance, covering more than 80% of the prescribed subjects to improve the scientific level of students and preserve the standardization of diploma programs.

External Evaluator

I have reviewed this course book, and it's perfect and fit for this subject at the level of institute student, so I have no suggestion.

Zanyar Shwan Ahmed
Assistant Lecture (Erbil Technology College) Information technology department