|  |  |
| --- | --- |
|  | **PIR MEHR ALI SHAH ARID AGRICULTURE UNIVERSITY**  **University Institute of Information Technology** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Object Oriented Programming (CS-423)** | | | | | |
| **Credit Hours:** | 4(3-2) | **Prerequisites:** | Programming Fundamentals (CS-323) | | |
| **Teacher Name:** | Dr. Tariq Ali | Office: Room 104 | Email tariq.ali@uaar.edu.pk | | |
| **Course Learning Outcomes (CLOs)** | | | | | |
| At the end of course the students will be able to: | | | | **Domain** | **BT Level\*** |
| 1. Understand principles of object oriented paradigm | | | | C | 2 |
| 1. Identify the objects & their relationships to build Object Oriented solution | | | | C | 3 |
| 1. Model a solution for a given problem using Object Oriented principles | | | | C | 3 |
| 1. Examine an Object Oriented solution. | | | | C | 4 |
| \*BT- Bloom’s Taxonomy, C=Cognitive domain, P=Psychomotor domain, A=Affective domain | | | | | |

|  |  |
| --- | --- |
| **Course Contents:** | |
| Introduction to object oriented design, history and advantages of object oriented design, introduction to object oriented programming concepts, classes, objects, data encapsulation, constructors, destructors, access modifiers, const vs non-const functions, static data members & functions, function overloading, operator overloading, identification of classes and their relationships, composition, aggregation, inheritance, multiple inheritance, polymorphism, abstract classes and interfaces, generic programming concepts, function & class templates, standard template library, object streams, data and object serialization using object streams, exception handling. | |
| **Course Objective:** | |
| This course covers the C++ language with a focus on its object-oriented features, and how these can be implemented as part of program designs and implementation. Student will also study and gain practical experience with the implementation issues related to object-oriented techniques, be able to build good quality software using object-oriented techniques and understand the role of patterns in object-oriented design. | |
| **Teaching Methodology:** | |
| Lectures, Written Assignments, Practical labs, Semester Project, Presentations | |
| **Courses Assessment:** | |
| Mid Exam, Home Assignments, Lab Assignments, Quizzes, Project, Presentations, Final Exam | |
| **Theory Total Marks= 60**  Mid Exam Marks: 18  QA (Including Labs): 12  Final Exam Marks: 30  **Practical Total Marks= 20**  Project Marks: 06  Final Practical Marks: 14  **Total Course Marks: 80** | **Grading Criteria:**  >=64 Grade A  >=52 Grade B  >=40 Grade C  >=32 Grade D  <32 Grade F  Min. 24 marks required in theory and Min. 08 marks required in practical to pass the course. |
| **Reference Materials:** | |
| 1. Object-Oriented Programming in C++, Robert Lafore; 4th Edition. 2. Problem Solving and Programming Concepts, M Sprankle, 9th Edition, Prentice Hall, 2012. 3. Programming and Problem Solving with C++, Nell Dale, Chip Weems, Comprehensive Edition, 2016. 4. C++ How to Program, 10th Edition, Deitel & Deitel, 2016. 5. Starting Out with C++ from Control Structures through Objects, 9th Edition, Tony Gaddis, 2017. 6. Java: How to Program, 11th Edition by Paul Deitel , 2017. 7. Beginning Java 2, 7th Edition by Ivor Horton 8. An Introduction to Object Oriented Programming with Java, 5th Edition by C. Thomas Wu. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Week/Lecture #** | | **Theory** | **Practical** |
| Week 1 | Lecture-I | Revision Structures:   * Defining Structures * Declaring Structure Variables * Initializing and Accessing Members of Structures Using Nested Structures |  |
| Lecture-II | Revision Functions:   * Declaration, Calling and Definition of Functions * Passing Arguments (constants & variables) to Functions * Returning Values from Functions |  |
| Lecture-III | Revision Functions:   * Passing Structure as Argument to Functions * Returning structure from functions * Passing Pointers as Arguments to Function |  |
| Practical-I |  | * Writing programs that input data into members of structure and then print data from the members of structure. * Writing programs that swap two structure type variables, Print the results before and after swapping. |
| Practical-II |  | * Writing programs that find out the maximum/minimum value in an array through pointer notation * Writing program to swap two values by passing pointers to function. |
| Practical-III |  | * Writing program to copy one string to another string using pointers. |
| Week 2 | Lecture-I | **Introduction to Object Oriented Design:**   * Object oriented design |  |
| Lecture-II | **History and advantages of object oriented design:**   * Object Oriented Approach |  |
| Lecture-III | * Characteristics/Advantages of OOP Languages (Encapsulation, Abstraction, Inheritance, Polymorphism). |  |
| Practical-I |  | * Practical exercise that how to convert a structural program into object oriented program using C++ |
| Practical-II |  | * Practical exercise that how to convert a structural program into object oriented program using C++ |
| Practical-III |  |  |
| Week 3 | Lecture-I | **Introduction to Object Oriented Programming Concepts:**  **Classes and Objects:**   * Objects and Classes. |  |
| Lecture-II | **Data Encapsulation:**   * Data encapsulation * Data Members and Member Function |  |
| Lecture-III | **Access Modifiers:**   * Public and private Access * Functions defining outside the class * Inline functions VS macros |  |
| Practical-I |  | * Programming Exercise for Classes and Objects |
| Practical-II |  | * Practical exercise that how to access private and public members of the class |
| Practical-III |  | * Practical exercise that how functions defined outside the class |
| Week 4 | Lecture-I | **Classes and Objects:**   * Array of Objects * Passing/Returning objects * Object as function argument * Function returning objects |  |
| Lecture-II | **Constructors & Destructors:**   * Constructors and destructors. * Default copy constructors |  |
| Lecture-III | * Constructor Overloading |  |
| Practical-I |  | * Programming Exercise for constructor |
| Practical-II |  | * Programming Exercise for destructor |
| Practical-III |  | * Program for Constructor overloading |
| Week 5 | Lecture-I | **Constant vs Non-Constant functions:**   * Constant and Classes * Constant Member Function * Constant Objects |  |
| Lecture-II | **Static data Members & functions:**   * Static Class Data (variables and functions) |  |
| Lecture-III | **Identification of classes and their relationships:**  **Composition:**  **Aggregation:**   * Association * Inheritance * Composition vs Inheritance |  |
| Practical-I |  | * Programming Exercise for constant variables and function access. |
| Practical-II |  | * Programming Exercise for Static variables and function access. |
| Practical-III |  | * Programming Exercise for concept of composition vs inheritance. |
| Week 6 | Lecture-I | * Friend Functions |  |
| Lecture-II | * Friend classes |  |
| Lecture-III | **Inheritance:**   * Derived and Base Classes. * Derived Class Constructors * Protected Specifier |  |
| Practical-I |  | * Programming Exercise for friend functions. |
| Practical-II |  | * Programming Exercise for friend classes. |
| Practical-III |  | * Programming Exercise for inheritance, constructors call sequence. |
| Week 7 | Lecture-I | **Inheritance:**   * Function Overriding * Scope Resolution with overridden function |  |
| Lecture-II | **Multi Inheritance:**   * Single Inheritance * Multilevel Inheritance |  |
| Lecture-III | * Multiple Inheritance and Constructor calling |  |
| Practical-I |  | * Programming Exercise for function overriding. |
| Practical-II |  | * Programming Exercise for multilevel inheritance. |
| Practical-III |  | * Programming Exercise for multiple inheritance and constructor calling. |
| Week 8 | Lecture-I | **String Classes:**   * String class * String class functions |  |
| Lecture-II | **String Classes:**   * String class * String class functions |  |
| Lecture-III | **Mid Course Revision:** |  |
| Practical-I |  | * Programming Exercise for string class built-in functions. |
| Practical-II |  | * Programming Exercise for string class built-in functions. |
| Practical-III |  | * Revision of upto mid practical’s. |
| **Midterm Exam** | | | |
| Week 9 | Lecture-I | **Function Overloading:**  **Operator Overloading:**   * Operators which cannot be overloaded * Unary operator overloading |  |
| Lecture-II | **Operator Overloading:**   * Binary operators overloading * Multiple Overloading |  |
| Lecture-III | **Operator Overloading:**   * Subscript operator overloading * Function call operator overloading * Assignment Operator overloading * Stream insertion and extraction operators overloading |  |
| Practical-I |  | * **Group formation and Course Project Allocation** |
| Practical-II |  | * Programming Exercises for unary, binary and multiple operator overloading. |
| Practical-III |  | * Programming Exercises for (), [] and = operator overloading. |
| Week 10 | Lecture-I | Data Conversion:   * Conversion between basic types * Conversion between user-defined and basic types * Conversion between objects of different classes |  |
| Lecture-II | **Polymorphism:**   * Virtual functions * Base class pointer * Inline Function * Late and Early binding |  |
| Lecture-III | **Abstract Classes and Interfaces:**   * Abstract Classes * Pure virtual functions |  |
| Practical-I |  | * Programming Exercise for conversion between objects of different classes. |
| Practical-II |  | * Programming Exercise for virtual functions. |
| Practical-III |  | * Programming Exercise for abstract class. |
| Week 11 | Lecture-I | **Memory Management:**   * Pointer arithmetic * Pointer vs. array * Use of New and Delete Keyword. * Stack vs. heap |  |
| Lecture-II | **Memory Management:**   * Pointer to Object * Pointer to pointer |  |
| Lecture-III | **Memory Management:**   * Array of Pointer to string |  |
| Practical-I |  | * Programming Exercise for dynamic memory. |
| Practical-II |  | * Programming Exercise for dynamic memory. |
| Practical-III |  | * Programming Exercise for dynamic memory. |
| Week 12 | Lecture-I | **Function Templates:**   * Template functions |  |
| Lecture-II | **Class Templates:**   * Template Classes |  |
| Lecture-III | **Class Templates:**   * Multi Template Classes |  |
| Practical-I |  | * Programming Exercise for template functions. |
| Practical-II |  | * Programming Exercise for template classes. |
| Practical-III |  | * Programming Exercise for template classes with multi template variables. |
| Week 13 | Lecture-I | **Exception handling:**   * Exceptions syntax * Simple Exceptions |  |
| Lecture-II | **Exception handling:**   * Multiple exceptions with Arguments. |  |
| Lecture-III | **Standard template library:**   * Introduction to STL * Stack as an STL example |  |
| Practical-I |  | * Programming Exercise for concept of exception |
| Practical-II |  | * Programming Exercise for exception with multiple error classes with arguments. |
| Practical-III |  | * Programming Exercise for STL (Stack library) |
| Week 14 | Lecture-I | **Standard template library:**   * Vector, List and Deque * Iterator * Function Objects |  |
| Lecture-II | **Object Streams:**   * Object Filing * Read, Write functions in object filing. |  |
| Lecture-III | **Object Streams:**   * Update and delete operations in object filing * File pointers for read and write |  |
| Practical-I |  | * Programming Exercise for Vector and List. |
| Practical-II |  | * Programming Exercise to read and write data using object filing in C++. |
| Practical-III |  | * Programming Exercise to update and delete data using object filing in C++. |
| Week 15 | Lecture-I | Introduction to Java:   * Object and Classes in Java * How to compile and execute java programs |  |
| Lecture-II | **Data and Object serialization using object streams:**   * File handling using Java |  |
| Lecture-III | * Update and delete in File using Java |  |
| Practical-I |  | * Practical exercise that how to convert a structural program into object-oriented program using Java |
| Practical-II |  | * Programming Exercise to read and write data using object streams in Java. |
| Practical-III |  | * Programming Exercise to update and delete data using object streams in Java. |
| Week 16 | Lecture-I | **Project Demos:**  GroupWise Project Demos and Presentations |  |
| Lecture-II | **Project Demos:**  GroupWise Project Demos and Presentations |  |
| Lecture-III | **Final Course Revision:** |  |
| Practical-I |  | * Revision of programming exercises |
| Practical-II |  | * Revision of programming exercises |
| Practical-III |  | * Revision of programming exercises |
| **Final term Exam** | | | |