# Introduction to Object Oriented Programming (OOP) II Syllabus

|  |  |
| --- | --- |
| **Week** | **Sunday/Tuesday/Thursday**  **Plan** |
| **Week 1** | Introduction + Outline |
| Revision (Function overloading + Arrays) |
| Revision ( Text File + Matrix ) |
| **Week 2** | **Pointers:**   * RAM addresses * Pointer concept * cin + cout tracing examples using & and \* operators. |
| **Pointers:**  pass by value, pass by address, pass by reference (swap example, passing a result to the calling function without using the return instruction example). |
| **Pointers:**   * Pointer operations: **+, -, =, ++, --, <, >, ==** * Pointer to pointer concept (\*\*) * Introduction to relationship between pointers and one-dimensional arrays. |
| **Week 3** | **Pointers:**  Continue pointers and arrays   * Various cin/cout/assignment statements using pointer notation (& and \*) on arrays |
| **Pointers:**  Pointers and strings   * Code examples (functions that use pointers on arrays in various ways): Sum array, string length. * Difference in ***cout*** between pointers to characters and other pointers (detailed tracing example). |
| **Pointers:**  Pointers and strings + constant qualifier   * Strings and pointers (errors’ example) * Constant qualifier * Code examples (functions) on using constant qualifier * Assignment operator with const qualifier * getchar, putchar, gets, puts (We must use **gets** to read strings with spaces) |
| **Week4** | **Pointers:**  Pointers and strings   * String handling library explanation + examples (strcpy, strlen, strcat, strcmp, stricmp). * String handling library codes (writing strcpy + strlen functions) using const. qualifier whenever suitable. * Void pointer (just mention it). |
| **Pointers:**  **Advanced Pointers:**   * Relationship between pointers and matrices * Matrix of integers (cin/cout examples using pointers) * Matrix of characters or Array of strings (cin/cout examples using pointers) |
| **Pointers:**  Dynamic memory allocation. |
| **Week5** | **Pointers:**  Dynamic memory allocation. |
| **Pointers:**  Dynamic memory allocation. |
| **First Exam** |
| **Week6** | **Function Overloading**  Various examples |
| **Object Oriented Programming (OOP):**   * Programming Paradigms * OOP design goals and principles |
| **Structures:**   * Structure definition, typedef * Defining objects * Input/output structure elements * array of structures (input/output/compute average of students). |
| **Week7** | **Structures:**   * Assign one structure to another one part at a time * Assign one structure to another using (=) * Comparing structures (==) * Pointer to structure |
| **Structures:**   * Dynamic structure / dynamic array of structures * Passing structures by value, address and reference. * Return structure, return address of structure. |
| **Structures:**   * Nested Structures * Nested Structure I/O * Class vs Structure   **Class**   * Introduction to classes and objects (class rectangle) * Access Control: public, private * Defining objects * Information hiding |
| **Week8** | **Class**   * Member data and member functions (Setters, Getters, ComputeArea) * Constructor specifications * Constructor importance and use * default constructor, constructors with parameters. * Compiler’s constructor * Destructor: Specifications, compiler’s destructor |
| * Class lab |
| **Second Exam** |
| **Week9** | Copy constructor (Shallow copy vs deep copy) |
| * Operator overloading * Assignment operator vs copy constructor |
| **String Class**   * Definition, member data and member functions * Default constructor, constructor with parameter * Default initializers * Destructor (importance of destructor when class has dynamically allocated data) * Shallow copy problem overview   + String Copy constructor   + String Operator = overloading |
| **Week10** | **String Class continue**   * Operator overloading * iostream operators overloading (>>, <<) |
| * **Lab: String Class** |
| **Static keyword**   * Static variables within function or loops * Static member function and static member data. * Examples (Shape class , Circle class, …etc) |
| **Week11** | **Inheritance**   * **Types of Relationship in C++ (Is a , uses a , has a)** * What is Inheritance ? * Inheritance is Hierarchal * Advantages of Inheritance * Class Inheritance Definition * Public, Protected and Private |
| * Inheritance and Accessibility * Public and Private Inheritance * Constructors, Destructors, and Inheritance * Rules for Building a Class Hierarchy |
| **Inheritance and function overriding**   * Overriding vs Overloading * 2 Ways to Prevent Method Overriding * Introduction to polymorphism |
| **Week12** | **Polymorphism**   * Polymorphism Types * Virtual Functions and Runtime Polymorphism in C++ * Late binding vs early binding |
| **Polymorphism**   * Polymorphism examples * What are pure virtual functions? * Abstraction + interfaces |
| * Polymorphism lab |
| **Week13** | **Multiple** **inheritance** |
| **Exception Handling statements**   * Exception Handling Examples |
| * Exception Handling - more examples |
| **Week14** | **Template**  Function Template |
| **Template**  Class Template |
| **Template**  More examples |
| **Week15** | **Binary files:**   * Introduction * Difference between binary files and text files   fwrite, fread, rewind (example)  **-------------------------------------------------------------------------------------------------**  **Binary files:**   * fseek with examples * Write 100 students, rewind, read and display * find student with max score * modify 3rd student’s grade * modify student’s grade given his name * modify all students’ grades (using array) * delete student using array * delete student using another file |
| **Binary files:**   * Binary File exercise: Display student given his name until the user enters (end)   function to compute size of binary file in bytes |
| **Lab: Binary Files’ Exercise** |
| **Week16** | Final Exam |