**Course Description/Objectives/Goals:**

The core objectives of this course are to introduce,

* Object oriented programming with data abstraction and encapsulation.
* The classes, objects and relationship among different objects and classes in C++?
* Generic programming using templates, and template specializations.

**Course Learning Outcomes (CLOs):**

|  |  |  |
| --- | --- | --- |
| **At the end of the course students will be able to:** | **Domain** | **BT\* Level** |
| Understand dynamic memory management with pointers. | C | 2 |
| Understand principles of object oriented program | C | 2 |
| Identify the objects & their relationships to build object oriented solution | C | 3 |
| Model a solution for a given problem using object oriented principles | C | 3 |
| Examine an object oriented solution | C | 4 |
| \* BT= Bloom’s Taxonomy, C=Cognitive domain, P=Psychomotor domain, A= Affective domain | | |

**Course Textbooks:**

* C++ Programming: Program Design Including Data Structures, by D. S. Malik (8th Edition)
* C++: How to Program? by Deitle & Deitle (9th Edition)

**Additional references and books related to the course:**

* Problem Solving with C++, by Walter Savitch
* [https://www.learncpp.com](https://www.learncpp.com/)

**Course Contents Weekly and Lecture-wise Breakdown**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week** | **Topic** | **Lecture-1** | **Lecture-2** |
| 1 | **Pointers** | Pointers Introduction, Pointer variables and Initialization, Address of Operator, Dereferencing Operator. Pointer Operations (Relational, Arithmetic) | Use of Constant with Pointers.  Difference between a Pointer and a Reference.  Passing pointers to functions by value and by reference. |
| 2 | Dynamic memory allocation using pointers and accessing dynamic memory. Dynamic Variables new and delete operators. | Dynamic 1- dimensional arrays, Create, Delete, Grow and Shrink.  Example of programs using 1D dynamic allocation: e.g., mathematical sets union and intersection. |
| 3 | Memory Leak and Dangling Pointers, Dynamic 1- dimensional char arrays for strings, string operations like search, concatenation etc. | Pointers Indirection. Dynamic 2D, allocation, matrices, CStrings etc. |
| 4 | **Object-oriented basics** | Structured Programming vs Object-oriented Programming, Principles of modularization, abstraction and encapsulation. | Objects vs Class, state vs behavior, access specifiers (Public, Private), Member functions (accessors, utilities, mutators etc) |
| 5 | Constructors (default, overloaded), Function overloading. | Dynamic memory allocation and Object assignment, Parameter passing, Shallow vs Deep copy, |
| **6** | **Mid Term 1** | | |
| 7 | **Object-oriented basics** | Copy constructor, Destructors, this pointer, | Cascaded function calls, static members, inline functions and other miscellaneous issues |
| 8 | **Operator overloading** | Unary operators using member functions | Binary operators using member functions |
| 9 | Binary operators using non-member functions, concept of friendship, | Unary operators, Pre and post increment, subscript operator. |
| 10 | **Object and Class relationships** | Part-whole relationships, Association/Aggregation | Composition  Implementation issues (constructor call sequence, initializer list, etc) |
| 11 | Inheritance basics, Type of Inheritance, public, protected, private. | Function Overriding and sub-typing details |
| **12** | **Mid Term 2** | | |
| 13 | **Object and Class relationships** | Polymorphism introduction Static vs dynamic binding details, virtual tables and virtual pointers, | Polymorphism vs down casting, run-time type identification, dynamic cast |
| 14 | Pure-virtual functions, Abstract classes, Interfaces (optional) | Multiple Inheritance and Diamond Problem Multiplicity, Memory Management  Bi-directional relationships, Forward-class declarations issues |
| 15 | **Generic Programming**  **&**  **Exception Handling.** | Template functions | Template classes  Template Specializations, |
| 16 | Exception Handling. | Introduction to STL, Iterators and Collections |