Lingnan University

Course Syllabus

Course Title	:	C++ Programming
Course Code	:	CDS2101
Recommended Study Year	:	Year 2
No. of Credits	:	3
Mode of Tuition	:	Sectional Approach
Class Contact Hours	:	3 hours per week
Category	:	Major Required
Discipline	:	Nil
Prerequisite(s)	:	Nil
Co-requisite(s)	:	Nil
Exclusion(s)	:	Nil
Exemption Requirement(s)	:	Nil

Brief Course Description:

As software development continues to drive innovation across industries, mastering programming fundamentals is essential for building efficient and reliable applications. This course is designed to teach students the fundamentals of programming using the C++ language. The course begins with the fundamentals of language syntax, variables, data types, and control structures, then progresses to more advanced topics including functions, arrays, pointers, and dynamic memory management. Students will also learn about data structures and file structures, including techniques for organizing and manipulating data efficiently. The course covers input/output operations, interaction with devices, and file handling to enable students to build programs that manage data from various sources. In addition, the course introduces object-oriented programming principles such as classes, inheritance, polymorphism, and encapsulation, which are essential for creating modular and maintainable software. Throughout the course, emphasis is placed on problem-solving, algorithmic thinking, and writing clean, efficient code. Students will gain practical experience through assignments and projects that simulate real-world applications.

Aims:

This course aims to:

1. Develop students' foundational knowledge and technical proficiency in programming through comprehensive coverage of the C++ language.

2. Cultivate students' ability to design, implement, and troubleshoot reliable and efficient computer programs in both individual and collaborative settings.
3. Enable students to apply key programming concepts, including structured and object-oriented techniques, to solve real-world computational problems.
4. Prepare students for industry practices in software development, including project management, documentation, and teamwork.

Learning Outcomes (LOs)

Upon successful completion of this course, students will be able to:

1. Explain the fundamental syntax, programming constructs, and core concepts of the C++ language. (PILO1)
2. Develop and debug efficient C++ programs utilizing variables, control structures, functions, arrays, pointers, and file operations. (PILO2)
3. Apply object-oriented programming principles, including classes, encapsulation, inheritance, and polymorphism, to construct modular and maintainable software solutions. (PILO2)
4. Analyse and solve practical computational problems through algorithmic thinking and structured programming, demonstrating readiness for industry and further study. (PILO2)
5. Demonstrate effective project management and teamwork skills through collaborative software development and clear technical communication. (PILO6)

Indicative Contents:

Introduction to C++ Programming

Overview of the C++ language; comparison with other programming languages; compiling and executing C++ programs; preliminaries of Object-Oriented Programming.

Variables, Data Types, and Operators

Declaring and initializing variables; primitive and compound data types; arithmetic, logical, and relational operators.

Control Structures

Conditional statements (if, switch); loops (for, while, do-while); scope and block structure.

Functions and Modularization

Function definition and invocation; parameter passing (by value, by reference); recursion; scope rules.

Arrays, Strings, and Pointers

Defining and manipulating arrays and strings; pointer variables; dynamic memory allocation and deallocation.

Structures and File Handling

User-defined structures; organizing and processing structured data; file input/output operations for persistent data management.

Object-Oriented Programming

Concepts of classes and objects; encapsulation; constructors and destructors; inheritance and polymorphism.

Problem Solving and Algorithms

Algorithmic design and analysis; applying structured and object-oriented approaches to real-world tasks.

Best Practices and Industry Readiness

Code readability, documentation, debugging, introduction to version control, project teamwork, and presentation.

Teaching Methods:

This course employs practice-oriented, sectional approach that integrates lectures, laboratory sessions, and group project-based learning. Lectures systematically introduce the core concepts, principles, and technical foundations of C++ programming, enabling students to gain a strong theoretical background in language syntax, control structures, data types, and object-oriented design. Laboratory sessions provide structured hands-on experience, where students apply theoretical knowledge to solve practical programming problems, develop algorithms, and implement C++ solutions under guided supervision. Throughout the course, in-class assessments are embedded within both lecture and lab environments, reinforcing immediate understanding of key topics and fostering active learning. Students are encouraged to engage in peer discussion and critique, both during scheduled class sessions and as part of their collaborative group project work. The group project simulates real-world software development practices, emphasizing teamwork, project planning, technical communication,

and the integration of object-oriented and structured programming techniques. Presentation and reporting activities further support the development of students' abilities to communicate technical ideas effectively and reflect critically on their programming process and outcomes. The final examination provides a comprehensive assessment of students' mastery of C++ programming concepts, their ability to solve novel problems, and their readiness for further study or professional application.

Measurement of Learning Outcomes:

Assessment / LOs	A1 (10%)	A2 (15%)	A3 (15%)	A4 (20%)	A5 (10%)	A6 (30%)
LO1	✓	✓				✓
LO2		✓	✓			✓
LO3			✓	✓		✓
LO4			✓	✓	✓	✓
LO5				✓	✓	

1. **Class Attendance and Participation** assess students' engagement in learning activities, their ongoing understanding of fundamental C++ concepts, programming constructs, and teamwork (LO1).
2. **In-class Lecture Assessment** evaluates students' immediate comprehension of lecture content, including theoretical foundations, syntax, and problem analysis (LO1, LO2).
3. **In-class Lab Assessment** measures students' practical proficiency in programming, their ability to apply C++ constructs, debug code, and solve problems in real-time settings (LO2, LO3, LO4).
4. **Group Project Presentation** assesses students' abilities in communicating technical ideas, demonstrating software solutions, and articulating their approach to problem-solving and project management (LO3, LO4, LO5).
5. **Group Project Report** evaluates the quality and completeness of project documentation, clarity of technical descriptions, and students' reflection on both programming process and teamwork (LO4, LO5).
6. **Final Examination** comprehensively measures students' mastery of course concepts, their ability to apply programming knowledge to novel problems, and their readiness for further study or industry engagement (LO1, LO2, LO3, LO4).

Assessment:

A1: Class Attendance and Participation	10%

A2: In-class Lecture Assessment	15%
A3: In-class Lab Assessment	15%
A4: Group Project Presentation	20%
A5: Group Project Report	10%
A6: Final Examination	30%
Total	100%

Required/Essential Readings:

1. Deitel, P., & Deitel, H. (2024). C++ How to Program: An Objects-Natural Approach (11th Edition). Pearson.
2. Stroustrup, B. (2014). Programming: Principles and Practice Using C++ (2nd Edition). Addison-Wesley.

Recommended/Supplementary Readings:

1. Schildt, H. (2015). C++: The Complete Reference (5th Edition). McGraw-Hill Education.
2. Meyers, S. (2014). Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14. O'Reilly Media.
3. Lippman, S., Lajoie, J., & Moo, B. (2012). C++ Primer (5th Edition). Addison-Wesley.

Important Notes:

(1) Students are expected to spend a total of 9 hours (i.e. 3 hours of class contact and 6 hours of personal study) per week to achieve the course learning outcomes.

(2) Students shall be aware of the University regulations about dishonest practice in course work, tests and examinations, and the possible consequences as stipulated in the Regulations Governing University Examinations and Course Work. In particular, plagiarism, being a kind of dishonest practice, is "the presentation of another person's work without proper acknowledgement of the source, including exact phrases, or summarised ideas, or even footnotes/citations, whether protected by copyright or not, as the student's own work". Students are required to strictly follow university regulations governing academic integrity and honesty.

(3) Students are required to submit writing assignment(s) using Turnitin.

(4) To enhance students' understanding of plagiarism, a mini-course "Online Tutorial on Plagiarism Awareness" is available on http://pla.ln.edu.hk/.
Students must adhere to the University's guidelines and practices when using Generative Artificial Intelligence (GAI) tools. The official documents "Guidelines for Using GAI Tools at Lingnan University" and "Best Practices for Ethical and Responsible Use of GAI Tools in Course Assessments" can be found here: https://www.ln.edu.hk/tlc/generative-artificial-intelligence/gai-guidelines-best-practices.

Lingnan University

Course Rubrics

Rubric for Attendance and Participation (10%)

Criterion	Excellent (90-100)	Good (70-89)	Pass (50-69)	Fail (0-49)
Class Attendance (50%)	Attends over 90% of all scheduled classes, including lectures and other sessions.	Attends at least 70% but less than 90% of all scheduled classes.	Attends at least 50% but less than 70% of all scheduled classes.	Attends less than 50% of all scheduled classes.
Class Participation (50%)	Actively and consistently participates class activities, discussions, and in-class exercises; demonstrates strong engagement, preparedness, and constructive contribution.	Participates regularly in class activities and discussions; demonstrates adequate preparation and engagement.	Participates occasionally in class activities with limited engagement or preparation.	Rarely or never participates in class activities; shows minimal engagement or preparation.

Rubric for In-class Lecture Assessment (15%)

Criterion	Excellent (90-100)	Good (70-89)	Pass (50-69)	Fail (0-49)
Theoretical Understanding (60%)	Demonstrates comprehensive and accurate understanding of all key programming concepts, syntax, and principles covered in lectures; answers are precise and well-justified.	Demonstrates good understanding of most programming concepts; minor gaps or inaccuracies in answers.	Demonstrates basic understanding with some gaps; several inaccuracies or unclear explanations.	Lacks basic understanding; answers are largely incorrect or missing.
Problem Analysis & Application (40%)	Correctly analyses and solves given programming problems using appropriate C++ constructs and logic; solutions are clear and efficient.	Analyses and solves most problems; solutions are generally appropriate though may lack clarity or efficiency.	Attempts most problems with partial solutions; logic is sometimes flawed or incomplete.	Fails to analyse or solve problems; solutions are absent or incorrect.

Rubric for In-class Lab Assessment (15%)

Criterion	Excellent (90-100)	Good (70-89)	Pass (50-69)	Fail (0-49)
Lab Task Completion and Programming Skills (50%)	Correctly implements all required programming tasks using C++; code runs as intended and meets specifications.	Most tasks completed correctly; minor errors.	Some tasks completed; frequent errors or incomplete features.	Fails to complete tasks; code incomplete or does not run.
Understanding of Concepts and Procedures (30%)	Clearly explains lab procedures, programming constructs, and logic used; demonstrates understanding of task requirements and C++ syntax.	Mostly clear explanations; minor misunderstandings.	Basic or incomplete explanations; some confusion.	Lacks understanding; explanations mostly incorrect or missing.
Code Design and Structure (20%)	Code is well-organized, demonstrates good modularity, uses functions/classes appropriately, and is easy to read.	Generally organized code with minor issues.	Code is minimally organized; modularity limited.	Code is disorganized, hard to follow, or not modular.

Rubric for Group Project Presentation (20%)

Criterion	Excellent (90-100)	Good (70-89)	Pass (50-69)	Fail (0-49)
Project Implementation and Functionality (40%)	Demonstrates a fully functional C++ program that meets all project requirements, integrates features such as input/output, data structures, error handling, and object-oriented modules; program runs reliably in demonstration.	Program implements most required features; minor errors or missing components, but generally runs as intended.	Meets basic project requirements, but program is incomplete, contains frequent errors, or limited functionality.	Fails to meet key project requirements; program is largely non-functional or fails to run.
Modularity and Code Design (20%)	Clear modular structure with effective use of classes, functions, and encapsulation; code organization facilitates understanding, extension, and debugging.	Modules/classes used appropriately in most parts; some inconsistencies or limited encapsulation.	Minimal modularization; code organization is weak or difficult to follow.	No meaningful modularization; code is monolithic, tangled, or disorganized.

Technical Communication and Explanation (20%)	Presentation clearly and confidently explains program logic, design decisions, key algorithms, and testing; uses relevant code samples and diagrams; answers questions accurately.	Presentation covers main aspects of logic and design; explanation is generally clear with minor gaps or limited detail.	Basic explanation of program structure and design; some aspects unclear or omitted.	Presentation is unclear, disorganized, or fails to communicate technical content.
Teamwork and Collaboration (10%)	Team members demonstrate clear role allocation, smooth collaboration (e.g. use of version control, merge coordination); all members contribute and communicate effectively.	Most team members contribute; some roles overlap or minor coordination issues.	Contributions are uneven, with unclear roles or limited coordination.	Little or no evidence of teamwork; major collaboration breakdown.
Responsiveness and Q&A (10%)	Accurately and confidently answers technical and project-related questions, demonstrating deep understanding of the code and process.	Responds to most questions appropriately; minor uncertainty or gaps.	Limited or vague responses to questions; struggles with technical detail.	Unable to respond meaningfully to questions; lacks project understanding.

Rubric for Group Project Report (10%)

Criterion	Excellent (90-100)	Good (70-89)	Pass (50-69)	Fail (0-49)
Technical Documentation and Code Explanation (40%)	Provides comprehensive, clear documentation detailing program architecture, module/class responsibilities, algorithm choices, and key code snippets; comments and diagrams support understanding.	Documentation is generally clear and complete, covers architecture and code explanations with minor omissions.	Basic documentation provided; covers some key aspects but lacks depth or clarity; few or no diagrams.	Documentation missing, incomplete, or unclear; code is not explained.
Testing, Debugging, and Quality Assurance (20%)	Thorough description of program testing strategies, debugging process, bug tracking, and evidence of efforts to improve reliability and robustness.	Describes main testing and debugging approaches; some evidence of systematic quality checks.	Limited or basic testing/debugging described; ad-hoc or incomplete.	No meaningful evidence of testing/debugging or quality assurance.
Reflection on Design Decisions and Improvement (20%)	Insightful analysis of design trade-offs, challenges faced, and potential improvements;	Some analysis and reflection on design and process;	Limited or superficial reflection; little discussion of	No meaningful reflection; report is only descriptive or incomplete.

	team reflects critically on their process and outcomes.	improvement areas mentioned.	improvements or lessons learned.	
Code Quality and Organization (10%)	Code in report and appendix is consistently well-organized, properly indented, follows naming conventions, and includes helpful comments.	Code is generally well-presented with some minor organizational issues or inconsistencies.	Code is difficult to follow due to inconsistent style or poor organization.	Code is messy, unstructured, or largely unreadable.
Referencing and Attribution (10%)	All external sources, libraries, and code are properly cited and attributed; clear delineation of team/individual work.	Most sources and contributions are cited; minor inconsistencies.	Some citations missing or unclear attribution.	No citation/attribution, or evidence of plagiarism.

Rubric for Final Examination (30%)

Criterion	Excellent (90-100)	Good (70-89)	Pass (50-69)	Fail (0-49)
Conceptual Understanding (40%)	Demonstrates thorough and accurate understanding of all core C++ programming concepts and principles; explanations are precise and well-justified.	Demonstrates good understanding of most concepts; minor gaps or inaccuracies.	Basic understanding shown, with gaps and some inaccuracies.	Lacks basic understanding; major errors or missing answers.
Programming Application (40%)	Correctly designs and writes code to solve given problems; solutions are efficient, logically structured, and well-documented.	Most problems solved correctly; code is functional with minor issues.	Some problems solved; code is incomplete or contains errors.	Cannot solve problems; code is incorrect or missing.
Problem Solving and Analysis (20%)	Effectively analyses novel programming problems, chooses appropriate solutions, and justifies approaches with clarity.	Analyses and solves most problems; justification generally clear.	Limited analysis or incomplete solutions; justification weak.	No meaningful analysis or problem solving demonstrated.