The Hong Kong University of Science and Technology

UG Course Syllabus (Spring 2025-26)

[Course Title] Object-Oriented Programming and Data Structures

[Course Code] COMP 2012

[No. of Credits] 4 credits

[Any pre-/co-requisites] COMP 2011

Name: CHAN Cecia, MAK Brian, SANDER Pedro

Email: kccecia@ust.hk, bmak@ust.hk, psander@ust.hk

Course Description

To learn the fundamental concepts and techniques behind object-oriented programming. They include abstract data types; creation, initialization, and destruction of objects; class hierarchies; polymorphism, inheritance, and dynamic binding; generic programming using templates. To learn the object-oriented view of data structures: linked lists, stacks, queues, binary trees, and algorithms such as searching and hashing.

List of Topics

1. Revision of dynamic data structures
2. C++ class basics
3. Separation compilation and makefile
4. Constructors, destructor, initialization
5. Inheritance, polymorphism, and dynamic binding
6. Generic programming
7. Namespace
8. Static member functions/data
9. rvalue reference and move semantics
10. Hashing
11. Binary search trees
12. STL (optional)

Intended Learning Outcomes (ILOs)

By the end of this course, students should be able to:

1. Write and analyze object-oriented programs in C++ with object creation, destruction, member variables and functions, inheritance, polymorphism, and templates.

2. Analyze simple problems and provide solutions with OOP.

3. Understand the basic operations of data structures such as stacks, queues, lists, binary search trees, and hashes, and their implementations.

4. Demonstrate the ability to use the learned data structures to solve problems in C++.

5. Develop large programs using separate compilation, good OOP design, and code reuse through the use of inheritance, and generic programming.

Assessment and Grading

This course will be assessed using criterion-referencing and grades will not be assigned using a curve. Detailed rubrics for each assignment are provided below, outlining the criteria used for evaluation.

Assessments:

Assessment Task	Contribution to Overall Course grade (%)	Due date
Mid-Term	26%	TBD
Laboratory exercises	10%	Weekly Basis (6-8 weeks during term)
Programming assignments	24%	Three Pas: 15/3, 31/3, 10/5
Final examination	40%	TBD

* Assessment marks for individual assessed tasks will be released within two weeks of the due date.

Mapping of Course ILOs to Assessment Tasks

Assessed Task	**Mapped ILOs**	**Explanation**
Laboratory exercises	ILO1-4	This task assesses students' ability to write and analyze object-oriented programs in C++ (ILO 1), analyze simple problems and solve them in C++ (ILO 2), understand the basic operations of data structures and use them to solve problems (ILO 3 & 4).
Programming assignments	ILO1-5	As with the above task, this task assesses students' ability to write and analyze object-oriented programs in C++ (ILO 1), analyze simple problems and solve them in C++ (ILO 2), understand the basic operations of data structures and use them to solve problems (ILO 3 & 4). Moreover, it assesses students' abilities to apply these skills to revelop large programs (ILO 5).
Midterm and Final Exam	ILO1-4	These exams are designed in order to assesses students' ability to write and analyze object-oriented programs in C++ (ILO 1), analyze simple problems and solve them in C++ (ILO 2),

		understand the basic operations of data structures and use them to solve problems (ILO 3 & 4).

Grading Rubrics

The following rubrics are posted on the course website at the beginning of the semester.

Task Learning Outcome	**Exemplary**	**Competent**	**Needs Work**	**Unsatisfactory**
1. Write and analyze object-oriented programs in C++ with object creation, destruction, member variables and functions, inheritance, polymorphism, and templates.	Demonstrate a comprehensive grasp of object-oriented concepts and fully demonstrates how to write object-oriented programs in C++.	Demonstrate a thorough grasp of object-oriented concepts and mostly demonstrates how to write object-oriented programs in C++.	Demonstrate a basic grasp of object-oriented concepts and barely able to demonstrate how to write object-oriented programs in C++.	Demonstrate a lack of grasp of object-oriented concepts and fail to demonstrate how to write object-oriented programs in C++.
2. Analyze simple problems and provide solutions with OOP.	Demonstrate an exemplary ability to analyze problems and solve them using OOP.	Demonstrate a proficient ability to analyze problems and solve them using OOP.	Demonstrate a developing ability to analyze problems and solve them using OOP.	Demonstrate deficiencies in their ability to analyze problems and solve them using OOP.
3. Understand the basic operations of data structures such as stacks, queues, lists, binary search trees, and hashes, and their implementations.	Demonstrate excellent understanding of the basic operations of the learned data structures and ability to implement them.	Demonstrate sufficient understanding of the basic operations of the learned data structures and ability to implement them.	Demonstrate limited understanding of the basic operations of the learned data structures and limited ability to implement them.	Demonstrate a lack of understanding of the basic operations of the learned data structures and inability to implement them.
4. Demonstrate the ability to use the learned data structures to solve problems in C++	Demonstrate an exemplary ability to use the learned data structures to solve problems in C++.	Demonstrate a proficient ability to use the learned data structures to solve problems in C++.	Demonstrate a developing ability to use the learned data structures to solve problems in C++.	Demonstrate deficiencies in the ability to use the learned data structures to solve problems in C++.

5. Develop large programs using separate compilation, good OOP design, and code reuse through the use of inheritance, and generic programming.	Demonstrate complete comprehension of OOP concepts and techniques for the development of large programs.	Demonstrate basic comprehension of OOP concepts and techniques for the development of large programs.	Demonstrate minimal comprehension of OOP concepts and techniques for the development of large programs.	Demonstrate no comprehension of OOP concepts and techniques for the development of large programs.

Final Grade Descriptors:

Grades	Short Description	Elaboration on subject grading description
A	Excellent Performance	This student excels in object-oriented programming (OOP) with C++, demonstrating a strong ability to design, analyze, and solve problems using OOP principles. They have a deep understanding of data structures, skillfully implementing and applying them to optimize solutions. Their proficiency in OOP techniques also enables them to develop scalable and well-structured programs for complex projects.
B	Good Performance	This student shows a solid grasp of object-oriented programming (OOP) in C++, capably designing and implementing object-oriented solutions for most problems. They understand core data structures and can effectively implement and apply them to solve programming challenges. While their OOP skills are proficient, they are still developing techniques for scaling programs to more complex, large-scale applications.
C	Satisfactory Performance	This student has a foundational understanding of object-oriented programming (OOP) in C++ and can apply basic OOP concepts with some guidance. They are capable of analyzing simple problems and implementing basic data structures, though they would benefit from further practice to strengthen their skills. With continued effort, they can develop a more robust ability to apply OOP principles and data structures to larger, more complex programs.
D	Marginal Pass	This student demonstrates a threshold understanding of object-oriented programming (OOP) in C++, sometimes struggling to apply OOP concepts effectively in their code. While they show some ability to analyze problems and use data structures their comprehension of C++ and OOP techniques for larger programs is minimal, but can potentially be improved based on knowledge from this course.
F	Fail	This student has failed to demonstrate a functional understanding of object-oriented programming (OOP) concepts in C++ and cannot write proper object-oriented programs. Their ability to analyze problems or implement solutions using OOP and data structures is severely lacking, showing no viable application of these concepts in practice. Additionally, they exhibit no comprehension of how to use OOP techniques for program development, falling short of even basic course expectations.

Course AI Policy

Generative artificial intelligence tools like ChatGPT or similar software are not allowed for labs and programming assignments. This has been clearly communicated to the students. The rational behind this is that this course teaches fundamental coding techniques in C++ and students need to learn and apply correct programming techniques, syntax, and development; relying on generative AI tools can hamper this, at their current level of programming understanding.

Communication and Feedback

Assessment marks for individual assessed tasks will be communicated via Canvas within two weeks of submission. Feedback on assignments will include detailed grading, and itemized marks deduction. We also provide template solutions to offer students the chance to improve their skills and correct their errors. Students who have further questions about the feedback including marks should consult the grader and instructor within a few working days after the feedback is received.

Resubmission Policy

No resubmission of assessment tasks allowed.

Required Texts and Materials

Textbooks

Paul Deitel, Deitel & Associates (2017). **C++ How to Program**.

M.A. Weiss (2014). **Data Structures and Algorithm Analysis in C++**.

Clifford A Shaffer. **Data Structures and Algorithm Analysis** Ed. 3.2 (C++ Version; electronic version: http://people.cs.vt.edu/~shaffer/Book/C++3elatest.pdf.)

Reference books

B. Eckel (2000). **Thinking in C++**.

L. Nyhoff (2005). **ADTs, Data Structures and Problem Solving with C++**.

Stanley Lippman (2013). **C++ Primer**.

Academic Integrity

Students are expected to adhere to the university's academic integrity policy. Students are expected to uphold HKUST's Academic Honor Code and to maintain the highest standards of academic integrity. The University has zero tolerance of academic misconduct. Please refer to Academic Integrity | HKUST – Academic Registry for the University's definition of plagiarism and ways to avoid cheating and plagiarism.

Additional Resources

N/A