The Hong Kong University of Science and Technology

UG Course Syllabus (Fall 2025-26)

[Course Title] **Programming with C++**

[Course Code] **COMP 2011**

[No. of Credits] 4-credits

[Any pre-/co-requisites] COMP 1021 OR COMP 1022P OR ISOM 3230

Exclusion: COMP 2012H

Name: PAPADOPOULOS Dimitris, OUYANG Xiaomin, CHEUNG Tsz Him, CHAN Gary, LI Cindy

Email: dipapado@cse.ust.hk, xmouyang@cse.ust.hk, thjcheung@cse.ust.hk, gchan@cse.ust.hk, lixin@cse.ust.hk

Course Description

This course covers programming and data structures using C++. In addition to basic programming concepts such as variables and control statements, students will learn about arrays, pointers, dynamic data allocation, linked lists, stacks, queues, binary trees, recursion, and the basics of object oriented programming. Prerequisite(s): COMP 1021 OR COMP 1022P OR COMP 1022Q (prior to 2020-21) OR ISOM 3230. Exclusion(s): COMP 2012H.

List of Topics

1. Introduction to computer programming
2. C++ basics: basic syntax, data types, operators
3. Control flow
4. Functions
5. Array and structure
6. Recursion
7. Scope
8. Struct
9. Pointers
10. Dynamic Data
11. Class
12. Stack and Queue
13. File input / output

Intended Learning Outcomes (ILOs):

On successful completion of the course, students will be able to:

1. Use common software tools to develop and debug a program written in an OOP language.
2. Write a short program to solve a simple problem in an OOP language.
3. Demonstrate that recursive and non-recursive functions are abstractions of sub problems in a task.
4. Describe the concept and the use of pointers in indirect addressing and dynamic memory allocation.
5. Demonstrate the use of several data structures.
6. Implement an abstract data type by defining a class in an OOP language.

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
Lab exercises	12%	Weekly basis, 8 labs over the semester
Programming assignments	24% (8%, 8%, 8%)	20/10, 10/11, 29/11
Midterm	24%	25/10/2025
Final exam	40%	8-19/12/2025

* 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
Lab exercises	ILO 1-6	The lab exercises assess the students' capability to use software tools (ILO1), analyze and write code (ILO2), and exercise various specific techniques such as recursion, pointer manipulation, data structures, and abstract data types (ILO3-6).
Programming assignments	ILO 1-6	Same as above, but with an emphasis on developing larger programs requiring more

		advanced techniques and combining different tools (ILO1-6)
Midterm and Final Exam	ILO 2-6	The exams assess the students' understanding of the course concepts and capabilities to use them to model small problems and write code to provide solutions (ILO2-6). Since the exams are in written, we do not asses ILO1.

Grading Rubrics

Task Assessment for Learning Outcome	Exemplary	Competent	Needs Work	Unsatisfactory
Use common software tools to develop and debug a program written in C++.	Use an IDE such as VS Code proficiently to write, compile, run, and debug a C++ program consisting of one or many source files.	Use an IDE such as VS Code effectively to write, compile, run, and debug a C++ program consisting of one or many source files.	Use an IDE such as VS Code to write, compile, and run a C++ program consisting of one source file. Have difficulty in dealing with programs consisting of more than one source file as well as debugging.	Have difficulty in using an IDE such as VS Code to write, compile, run, and debug a C++ program consisting of one source file without guidance.
Write and analyze short programs that solve simple problems in C++.	Construct a solution to a written problem by writing a complete C++ program of no more than 500 lines of codes on one's own.	Construct a solution to a written problem by writing a complete C++ program of no more than 500 lines of codes on one's own if the	Require assistance to break down a written problem into sub-problems of sufficiently small sizes before being able to construct a	Is unable to construct a solution to a written problem by writing a complete C++ program even under guidance. Skeleton code need to be given which breaks down the solution into a set of small functions with clear interface

		requirements are clearly explained and the required programming constructs are told.	solution for each sub-problem with no more than 100 lines of codes. The required programming constructs also need to be told.	so that the student may be able to implement them.
Demonstrate that recursive and non-recursive functions are abstractions of sub-problems in a task.	Demonstrate thorough understanding of how recursion works. Be able to develop a recursive solution to a written problem on one's own, and sometimes contrast it with the corresponding non-recursive solution.	Demonstrate sufficient understanding of how recursion works. Be able to develop a recursive solution to a written problem if given the recursive algorithm, and sometimes contrast it with the corresponding non-recursive solution.	Demonstrate insufficient understanding of how recursion works. Be able to develop recursive solutions only to some simple problems, and only if the recursive algorithm is given. Cannot contrast the recursive solution with the corresponding non-recursive solution.	Is unable to understand how recursion works. Is unable to develop recursive solutions to problems even if the recursive algorithm is given.
Understand and demonstrate the use of pointers in indirect addressing and dynamic memory allocation.	Demonstrate strong understanding of the concept of pointers. Is able to use pointers effectively in indirect addressing and dynamic memory allocation in a great variety of scenarios.	Demonstrate sufficient understanding of the concept of pointers. Is able use pointers effectively in indirect addressing and dynamic memory allocation in standard scenarios.	Demonstrate marginal understanding of the concept of pointers. Is able to use pointers in indirect addressing and dynamic memory allocation only in simple scenarios.	Demonstrate little understanding of the concept of pointers. Have great difficulty inusing pointers in indirect addressing and dynamic memory allocation even in simple scenarios.

Understand and demonstrate the use of various data structures.	Is able to choose the appropriate data structures such as linked lists, stacks and queues to solve problems, and implement the solution in C++ on one's own.	Is able to use the required data structures such as linked lists, stacks and queues to solve problems, and implement the solution in C++ on one's own.	Is able to use the required data structures such as linked lists, stacks and queues to solve simple problems, and implement the solution in C++ with guidance.	Demonstrate little understanding of data structures such as linked lists, stacks and queues, and is unable to use them to solve problems even with guidance.
Implement an abstract data type (ADT) by defining a class in C++.	Given the description of a simple ADT, is able to implement it with a complete C++ class definition that includes appropriate class members and class member functions. Is able to write programs that create and manipulate ADT objects.	Given the description of a simple ADT, able to implement it with a complete C++ class definition when its class members and class member functions are also hinted. Is able to write programs that create and manipulate ADT objects.	Given the description of the C++ class definition of a simple ADT, including its class members and class member functions, is able to implement the member functions. Sometimes is able to write programs that create and manipulate ADT objects.	Have great difficulty in understanding the link between a simple ADT and its C++ definition. Is unable to complete C++ definition for simple ADTs and to program with C++ classes.

Final Grade Descriptors:

Grades	Short Description	Elaboration on subject grading description
A	Excellent Performance	This student excels in C++ programming, expertly using IDEs to write, debug, and manage code while independently solving problems with well-structured programs under 500 lines. They demonstrate mastery of recursion, pointers, and dynamic memory allocation, applying them effectively in diverse scenarios, and can skillfully implement data structures like linked lists, stacks, and queues. Additionally, they can design robust C++ classes for abstract data types (ADTs) and manipulate objects with precision.
B	Good Performance	This student demonstrates solid C++ skills, competently using IDEs to develop and debug programs while solving

		well-defined problems (under 500 lines) when given clear requirements. They understand recursion, pointers, and dynamic memory allocation well enough to apply them in standard scenarios, and can implement basic data structures like linked lists, stacks, and queues independently. With some guidance, they can also design simple ADTs as C++ classes and manipulate objects effectively.
C	Satisfactory Performance	This student has achieved satisfactory proficiency in foundational C++ programming, capably writing and running single-file programs in VS Code while beginning to explore multi-file projects with some guidance. They can break down and solve well-defined problems (under 100 lines) when given structural support, demonstrating growing competence with core programming constructs. While their understanding of recursion, pointers, and data structures is not fully developed, they can apply these notions. With continued effort, they are well-positioned to solidify these skills for more complex tasks.
D	Marginal Pass	This student shows minimal C++ proficiency, (e.g., handling only basic single-file programs and requiring extensive help with multi-file projects or debugging). They solve small problems (under 100 lines) when given step-by-step guidance and pre-divided tasks. Their grasp of advanced notion is weak, often misapplying concepts, and they depend heavily on examples. With significant effort and using this course as a basis, the student can strengthen their skills.
F	Fail	This student has failed to demonstrate basic C++ programming competency. They cannot independently write, compile, or debug simple programs, requiring extensive step-by-step guidance for every task. Fundamental programming concepts are not understood, and they are unable to solve problems or implement solutions even with significant support. Their performance shows no evidence of meeting the course's minimum learning objectives.

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.

Textbooks

Big C++: Late Objects, 3rd Edition
Cay S. Horstmann eBook ISBN:
9781119402978

Reference books
N/A

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