SYLLABUS

COURSE	CSCI 272 | Object-Oriented Programming (OOP)
INSTRUCTOR	Avijit Roy
E-MAIL	ARoy@jjay.cuny.edu
OFFICE	TBA
COURSE WEBPAGE	TBA
OFFICE HOURS	I will be available for weekly virtual office hours by appointment. To schedule an appointment or to ask any questions about the course content, please email me.

LEARNING OBJECTIVES

At the end of this course, students will be able to:

1. Understand the concepts of functions, including function declaration, definition, and function call. Learn to work with arrays, including array declaration, initialization, and accessing elements using indexing. Explore the use of pointers for memory management and manipulation of data.
2. Discover vectors as a versatile alternative to arrays in C++, learning to declare, initialize, and access values in vectors. Understand the concept of generic functions and how function templates can be employed to write code that adapts to various data types.
3. Learn to work with strings, including creating, comparing, and concatenating strings. Explore methods to get the length of strings and convert between strings and other data types.
4. Understand the concept of classes and objects in object-oriented programming. Learn to define classes and instantiate objects, applying public and private access specifiers. Explore member functions and data members within classes.
5. Discover formatted and unformatted input and output using streams. Learn techniques to handle bad input and failed input states during stream operations.
6. Explore stringstreams for input and output manipulation, converting between strings and other data types. Learn to work with sequential text files and CSV files, including reading from and writing to them.
7. Separate class interface and implementation for better code organization. Understand constructors and destructors for proper object initialization and cleanup. Explore composition, dot and arrow operators, the "this" pointer, friend functions, and static members.
8. Learn to overload unary and binary operators to define custom behaviors for class objects. Understand dynamic memory allocation and its use in operator overloading.
9. Understand the concept of inheritance and its implementation with base and derived classes. Explore the protected access specifier for class members.
10. Learn about virtual functions, enabling runtime method selection in derived classes. Understand pure virtual functions and abstract classes.
11. Compare linear and binary search algorithms for efficient data retrieval. Explore various sorting algorithms to arrange data in a desired order.
12. Discover try, catch, and throw constructs for handling exceptional situations in code. Learn to create custom exceptions for specific error scenarios.

DESCRIPTION

The "OOP in C++" course provides students with a comprehensive understanding of OOP principles and C++ programming. Students will learn to design, implement, and manage efficient object-oriented programs. Key objectives include grasping OOP concepts, applying principles to build robust software, implementing classes and objects, understanding encapsulation and data abstraction, exploring inheritance and polymorphism, mastering operator overloading and exception handling, and utilizing the Standard Library. Practical application through hands-on projects reinforces real-world problem-solving skills, preparing students for success in computer science and software engineering.

The prerequisite for this course is ENG 101 and CSCI 271. This course builds upon the material covered in CSCI 271 and is the second Core Computer Science course in the Computer Science and Information Security major and minor.

TEXTBOOK

C++ How to Program (10th Edition), By Paul Deitel & Harvey Deitel
ISBN-13 978-0-13-444823-7 [Amazon]

This is the main book that we will use in this course.

References (Optional)

- C++ How to Program Late Objects Version (Ed. 7) by Deitel, Paul; Deitel, Harvey [Amazon]
- C++ Reference: http://en.cppreference.com/w/cpp
- C++ FAQ: https://isocpp.org/faq

IDE/Tools

- Code::Blocks: Main IDE for writing and compiling C++ programs. Lightweight and beginner-friendly.
- Visual Studio Community: An advanced IDE with IntelliSense and debugging support for C/C++.

METHOD OF ASSESSMENT

Discussion and Class Participation	15%
Quizzes (2 Announced & 2 Pop Quizzes)	15%
Assignments	20%
Mid Term	25%
Final	25%

Note: Grading of assignments and exams can be reviewed in person by the professor to ensure that the submitted work is the student's own code and to address any questions regarding the understanding of the submitted material.

QUIZZES

During the semester, students may be given quizzes from the topics covered during the previous lectures. This is done to encourage students to cultivate the habit of preparing for the classes weekly, and hence be ready to learn new concepts the following weeks. This also helps them to prepare for the examination. Generally, a quiz will be given at the beginning of a class and students who are absent or late for the class will not be allowed to take that quiz later.

POP QUIZZES

Unannounced quizzes may be given throughout the semester to reinforce weekly topics and encourage consistent preparation. These quizzes will be brief and administered at the beginning of class. Students arriving late or absent will not be permitted to make up missed pop quizzes, except under documented emergencies.

EXAMS

MIDTERM

The midterm will consist of two parts:

1. A written, in-class exam testing conceptual understanding and code analysis.
2. An individual take-home programming project that applies OOP concepts from Weeks 1–6.

FINAL

The final assessment will consist of a team-based capstone project in which students collaboratively design, implement, and present a complete object-oriented C++ application. The project will integrate key concepts learned throughout the course, including class design, inheritance, polymorphism, operator overloading, exception handling, and file I/O.

Each team will be responsible for:

- Planning & design documentation (UML encouraged)
- Code implementation using modular and reusable components
- Code presentation to the class
- Individual code contribution explanation (Comments on codes)

Teams will be assigned or approved by Week 12. A detailed rubric and example project ideas will be provided in advance.

PROGRAMMING PROJECTS

In this course, students will develop proficiency in efficiently implementing object-oriented programs using the C++ programming language. Throughout the course, a series of challenging programming projects will provide students with practical experience in applying OOP principles, class and object implementation, inheritance, polymorphism, operator overloading, and exception handling. These hands-on projects will require students to design and construct software solutions for real-world scenarios, fostering their problem-solving skills and reinforcing their understanding of OOP concepts.

RESPONSIBILITIES

Students are expected to attend the class lectures and take the exams at the scheduled times. Assigned readings, exercises and projects must be completed on time.

ACADEMIC INTEGRITY

Cheating on exams or copying projects will not be tolerated. Please review the College's policies on Plagiarism and Cheating. Moreover, if you copy projects or assignments, you will not be able to answer questions on the exams.

ACCESSIBILITY SERVICES

If you are a student with a disability, either temporary or permanent, please reach out to the Office of Accessibility Services (OAS) at accessibilityservices@jjay.cuny.edu. We are here to assist you in achieving your academic goals.

OAS works in partnership with the entire John Jay community to ensure access to all areas of campus life and arrange for appropriate academic adjustments, programs and services. We provide referrals to both on-campus offices and outside agencies to support students' success.

USE OF AI TOOLS (E.G., CHATGPT, COPILOT, ETC.)

You may use AI tools (e.g., ChatGPT, Copilot) to learn and explore concepts, but not as a substitute for your own work.

- **Cite all use:** If AI helps you solve a problem or write code, clearly cite the tool and include a short summary or screenshot of your interaction or a sharing link of the conversation.
- **No copy-paste code:** Submitting AI-generated code without citation is considered plagiarism.
- **Understand your work:** You must be able to explain your code in person if asked. Inability to do so may result in loss of credit.
- **Purpose:** AI can support learning, not replace it.

Citation Example: *Used ChatGPT on March 12, 2025, to understand operator overloading syntax. Prompt: "How to overload + operator in C++?"*

IMPORTANT WEBSITES

CUNY Brightspace: All course related assignments and material will be available on Brightspace. Announcements will be posted on Brightspace.

Students must have a working CUNYFirst account in order to receive course related information.

PROPOSED SCHEDULE OF LECTURES

Week	Topics	Chapters or sections	Assignments
1	**Review** Functions, Arrays, Pointers	6.4, 6.5, 6.10, 6.11, 6.13 - 6.16, 7.1 - 7.2, 8.1 - 8.9	**Assignment 1** **Due in 7 days**
2	**Function Templates, Vectors** Vectors as an alternative to arrays, declare, initialize, access values, get size, add values	6.6, 6.17, 7.10	
3	**Strings** Create, compare, concatenate, get the length of, modify strings, and convert between string and other data types	21.1 - 21.10, 21.13	**Assignment 2** **Due in 7 days**
4	**Introduction to classes** Class definitions and instantiation, public and private access specifiers, member functions, data members	3.1 - 3.7	**QUIZ 1**
5	**Streams – Input/Output** Formatted and unformatted input and output, recovering from bad input and failed input state	13.1 - 13.8	**Assignment 3** **Due in 7 days**
6	**Streams – Stringstreams** Stringstreams for input and output manipulation	21.12	-
7	**Midterm Exam -** A written exam on theories and an Individual take-home Project	Week 1 to 6	**Midterm Exam and Project**
8	**Streams – Files** Sequential text and CSV files	14.1 - 14.4	**Assignment 4** **Due in 7 days**
9	**More on Classes** Separate interface and implementation, constructors, destructors, composition, dot and arrow operators, this pointer, friend, static	9.1 – 9.15	
10	**Operator Overloading** Overload unary and binary operators, dynamically allocated memory	10.1 – 10.11	**Assignment 5** **Due in 7 days**
11	**Inheritance** Base and derived classes, protected access specifier	11.1 – 11.6	**QUIZ 2**
12	**Polymorphism** Virtual functions, pure virtual functions, abstract classes	12.1 – 12.7	**Assignment 6** **Due in 7 days**
13	**Class Templates** Reusable Generic Classes	20.1 – 20.3	
14	**Exception Handling** Try, catch and throw, create custom exceptions	17.1 – 17.5	**Assignment 7** **Due in 7 days**
15	**Final Exam** Team-based capstone project		**Final Project**