CPSC 370B2 C++ Programming

Fall 2024

Professor	Email	Course Location
Jennifer A. Polack	jenniferpolack@gmail.com	Farmer 054 Wednesday 8-8:50 am
Office Hours		
M 8-9am, M 11am-12pm, W 11-12, R 9:30 – 11am, 12:15 – 12:45 PM		

Description

C++ is a powerful and versatile programming language widely used in various industries, such as software development, gaming, and system programming. This course is designed to introduce you to the fundamentals of C++ programming and help you understand its key concepts and features.

You will learn about data types, variables, operators, control structures, functions, classes, and objects in C++ throughout the course. You will also dive into more advanced topics such as memory management, pointers, and file handling. By the end of the course, you will have the knowledge and skills to create your own C++ programs and applications.

Expectations and Goals

This C++ course requires several coding style expectations to ensure clean and readable code. Some common expectations include:

1. Consistent indentation: Use tabs or spaces to indent code blocks uniformly for improved readability.

2. Descriptive variable names: To make the code more understandable, use meaningful and descriptive names for variables, functions, and classes.

3. Proper use of comments: Include comments to explain complex logic, algorithms, or functionalities within the code.

4. Limit line length: To prevent horizontal scrolling, keep lines of code to a reasonable length (usually around 80-100 characters).

5. Proper formatting of code blocks: Use braces to consistently define code blocks (if statements, loops, functions) and align them appropriately for easy readability.

6. Avoid unnecessary complexity: Avoid overly complicated logic or unnecessary features to keep the code simple.

7. Consistent naming conventions: Follow a consistent naming convention for variables, functions, and classes (e.g., CamelCase, snake_case) throughout the codebase.

8. Use of white space: Use white space effectively to improve the overall readability of the code.

Documentation is an essential aspect of software development in C++ to help developers understand the code, its functionality, and how to use it. Some common types of documentation typically required in C++ projects include:

1. Code comments: Inline comments that explain the purpose of code blocks, functions, variables, and complex logic. Comments should clarify the code's behavior and reasoning behind confident implementation choices.

2. Function/method documentation: Documentation that describes the purpose, input parameters, return values, and potential side effects of functions or methods. This documentation is often in the form of comments above the function definition.
3. Class documentation: Documentation that describes the purpose, data members, member functions, and class usage. This helps to explain the class's role and how it can be used in the codebase.
4. File/module documentation: Overview of the contents and purpose of a particular file or module, including its dependencies, responsibilities, and usage within the project.

Required materials

ZYBOOKS
ISBN 13: 9798203352149

ZyBooks: Log into Canvas and go to homework one. It will connect you with the book for purchase.

1. Click on your zyBooks link in your learning management system (Do not go to the zyBooks website and create a new account)
2. Subscribe

Course Schedule

Week	Topic	Reading	Exercises
Week 1	Introduction Fundamental Data Types	Chapters 1 & 2	Lab 1: 1.9, 1.10, 2.21, 2.22, 2.23, 2.24 Challenge Activities
Week 2	Decision, Loops NO CLASS (I am in and you can come get help but content is same as java. If people come I will help and/or lecture)	Chapters 3 & 4	Lab 2: 3.25, 4.24, 4.26, 4.27, 4.28, 4.29, 4.30 Challenge Activities
Week 3	Functions (5.1- 5.13) NO CLASS (I am in and you can come get help but content is same as java, except functions are not in classes. If people come I will help and/or lecture))	Chapter 5	Lab 3: 5.3, 5.24 and 5.25 Challenge Activities
Week 4	Functions (5.14 – 5.19)	Chapter 5	Lab 4: 5.14, 5.15 and 5.26 Challenge Activities
Week 5	Arrays (6.1 – 6.12) NO CLASS (I am in and you can come get help but content is same as c, which you should have learned in 305. If people come I will help and/or lecture))	Chapter 6	Lab 5: 6.1, 6.3,6.7,6.22,6.23, 6.24 Challenge Activities
Week 6	Arrays (6.13 – 6.18)	Chapter 6	Lab 6: 6.13 and 6.15 Challenges Activities 6.25 and 6.26 Challenge and Participation Activities

Week	Topic	Reading	Exercises
Week 7	Pointers NO CLASS (I am in and you can come get help but content is same as c, which you should have learned in 305. If people come I will help and/or lecture))	Chapter 7.1 – 7.8	Lab 7: 7.1, 7.3, 7.4, 7.6, 7.8 Participation Activities
Week 8	Pointers and Structures	Chapter 7.8 – 7.21	Lab 8: 7.8, 7.9, 7.10, 7.14, 7.15, 7.21 Participation and Challenge Activities
Week 9	Streams	Chapter 8	Lab 9: 8.6, 8.17, 8.18, 8.19 Challenge Activities
Week 10	Classes	Chapter 9	9.2, 9.5, 9.6, 9.23, and 9.25 Challenge Activities 9.15 and 9.16 Participation Activities
Week 11	Inheritance	Chapter 10	10.2, 10.4, 10.5, 10.14, 10.15, 10.16 Challenge Activities
Week 12	Advance C++	Chapter 11	11.1, 11.12 Participation Activities 11.18 Participation & Challenge Activities
Week 13	Thanksgiving No Class	FINAL PROJECT (Must have Documentation)	Final Project Submission 1 All Class Files should be declared and documented. Function stubs should be present. It should compile
Week 14	No Class	FINAL PROJECT (Must have Documentation)	Final Project Submission 2: Your main menu/driver should introduce your program. At least two features of your program should compile and run.
Week 15	Final Exam	Mon, December 9, 8:30 - 11:00 a.m. FINAL PROJECT DEMO: Don't present do Pass (Must have Documentation)	

Grades

50% Labs
50% Final Project (If your project is not up to par, you will fail the course. Must Compile and do 80% of all features)

You can always get an up-to-date picture of your course grade by checking Canvas.

Final letter grades will be determined according to the scale and course policy below.

A 93-100% B+ 87-89% C+ 77-79% D+ 67-69%

A- 90-92%	B 83-86%	C 73-76%	D 60-66%
	B- 80-82%	C- 70-72%	F below 60%

Mid-semester Grades

The University provides the opportunity to provide grading feedback midway through the semester. A student will receive a mid-semester unsatisfactory (U) grade if they have a 65% or below overall average, 65% or below on exams or quizzes, or 65% or below on programming projects. Any student receiving a U in this course should meet with the instructor to develop a performance improvement plan.

The Honor Code

Plagiarism of code in a program is an honor code violation, just like plagiarism of words in a paper.

For any assignment you submit, be prepared to explain how the program works or how you arrived at your answer if asked. This ensures you only turn in the items you fully understand in your projects.

All CPSC programs submitted for class assignments may be subjected to automated plagiarism tools such as Moss.

You may collaborate freely with other students on labs but you are not allowed to share all your code with another student

You may not collaborate on the final project.

Use of AI in this Course

AI is not permitted on any assignments or deliverables in this course. Use of AI on any submitted work will be considered a violation of course policy and as such, the student may be referred to the UMW Honor Council for a violation of academic integrity.

Recording

Classroom activities in this course may be recorded by students enrolled in the course for the personal, educational use of that student or for all students presently enrolled in the class only, and may not be further copied, distributed, published or otherwise used for any other purpose without the express written consent of the course instructor. All students are advised that classroom activities may be taped by students for this purpose. Distribution or sale of class recordings is prohibited without the written permission of the instructor and other students who are recorded. Distribution without permission is a violation of copyright law. This policy is consistent with UMW's *Policy on Recording Class and Distribution of Course Materials.*

Disabilities

"The Office of Disability Resources has been designated by the university as the primary office to guide, counsel, and assist students with disabilities. If you receive services through the Office of Disability Resources and require accommodations for this class, please provide me a copy of your accommodation letter via email or during a meeting. I encourage you to follow-up with me about your accommodations and needs within this class. I will hold any information you share with me in the strictest confidence unless you give me permission to do otherwise.

If you have not made contact with the Office of Disability Resources and have reasonable accommodation needs, their office is located in Seacobeck 005, phone number is (540) 654-1266 and email is odr@umw.edu. The office will require appropriate documentation of disability."

Title IX Statement

University of Mary Washington faculty are committed to supporting students and upholding the University's Policy on Sexual and Gender Based Harassment and Other Forms of Interpersonal Violence. Under Title IX and this Policy, discrimination based upon sex or gender is prohibited. If you experience an incident of sex or gender based discrimination, we encourage you to report it. While you may talk to me, understand that as a "Responsible Employee" of the University, I MUST report to UMW's Title IX Coordinator what you share. If you wish to speak to someone confidentially, please contact the confidential resources found below. They can connect
you with support services and help you explore your options. You may also seek assistance from UMW's Title IX Coordinator, their contact information can be found below. Please visit http://diversity.umw.edu/title-ix/ to view UMW's Policy on Sexual and Gender Based Harassment and Other Forms of Interpersonal Violence and to find further information on support and resources.

Ruth Davison, Ph.D.
Title IX Coordinator
Lee Hall, Room 401
1301 College Avenue
Fredericksburg, VA 22401
Phone: 540-654-5656
E-mail: rdavison@umw.edu
Website:
http://diversity.umw.edu/title-ix/

Confidential Resources
On-Campus
Talley Center for Counseling Services
Lee Hall, Room 106, 540-654-1053
Student Health Center
Lee Hall, Room 112, 540-654-1040

Off-Campus
Empowerhouse
24-hr hotline: 540-373-9373
Rappahannock Council Against Sexual Assault (RCASA)
24-hr hotline: 540-371-166

Github for Final Project

I am implementing a submission policy using GitHub for the final project to uphold its integrity and facilitate a more transparent workflow.

Mandated: Updating your local repository with your online repository in Visual Studio is an important habit that must be done once an hour each time you work on it. FAILING TO DO THIS WILL LEAD TO A 0 on your program grade.

This platform will enable me to monitor the progress of your code development and verify the individuality of your work.

**Instructions for Using GitHub with Visual Studio: **

1. **Create a GitHub Account: **
If you don't already have a GitHub account, please sign up at [github. com](https://github.com).

2. **Create a New Repository (Repo) on GitHub:**
- Log in to your GitHub account.
- Click the "+" icon in the upper-right corner and select "New repository".
- Name your repository (e.g., program-4-your username).
- Keep the repository Public (private repositories are not allowed for submissions).
- Initialize the repository with a README.
- Click "Create repository".

3. **Clone the Repository to Your Local Machine:**
- Open Visual Studio and go to Team Explorer.

Mandated: Updating your local repository with your online repository in Visual Studio is an important habit

that must be done once an hour each time you work on it. FAILING TO DO THIS WILL LEAD TO A 0 on your program grade.

Keeping your local repository up-to-date with your online repository in Visual Studio is simple, especially when you're the sole contributor.

Here's how to do it:

1. **Open Visual Studio**:
Launch Visual Studio and open your project linked to the GitHub repository.

2. **Navigate to Team Explorer**:
Go to "View" on the main menu and click "Team Explorer." This pane manages all your source control features.

3. **Fetch the Latest Changes**:
- In the "Team Explorer," navigate to the "Synchronization" section.
- Click "Fetch" to retrieve any changes from the online repository without merging them into your local repository. This is a safe operation to see what changes are available.
- If you're the only one using the repository, fetching might not be necessary, as there shouldn't be any new changes unless you've used a different device or interface to update the repo.

4. **Pull the Latest Changes**:
- If you've made any updates to the repository outside of Visual Studio (e.g., from the GitHub website, another instance of Visual Studio, or a different computer), you'll want to integrate these changes into your local repository.
- Click "Pull" to merge the online repository changes into your local branch.
Pulling will fetch the changes and merge them simultaneously, updating your local repository to the state of the online repository.

5. **Resolve Any Merge Conflicts**:
- If there are any conflicts between your local and remote changes, Visual Studio will highlight these.
- Manually resolve each conflict by editing the files. Visual Studio offers merge conflict resolution tools to help with this process.

6. **Push Changes (If Any) **:
- If you've made new commits locally that you need to update to the remote repository, click the "Push" button to upload your local changes to the online repo.

Final Project
All projects must be hosted on GitHub. I will use your GitHub to track your programming progress. Pull and push to the repo once an hour to see the changes over time. If you have no history, you will not pass the final project.

Option 1: Stock Market Simulation System

Description: You are tasked with creating a stock market simulation program in C++ that simulates traders' buying and selling of stocks. The program should model the real-world stock market scenario by allowing traders to buy and sell stocks, view stock prices, and track their portfolio performance.

Requirements:

1. Implement a Stock class that represents a stock with symbol, price, and quantity attributes.
2. Implement a Trader class representing a trader with attributes such as name, account balance, and stock portfolio.
3. Allow traders to buy stocks by specifying the stock symbol and quantity, deducting the cost from their account balance.
4. Allow traders to sell stocks from their portfolio by specifying the stock symbol and quantity, adding the sale proceeds to their account balance.
5. Display a list of available stocks with their current prices.
6. Display the trader's stock portfolio with details such as stock symbol, quantity, and total value.
7. Calculate and display the total value of the trader's portfolio based on the current stock prices.
8. Implement a simple stock price simulation that randomly fluctuates the prices of stocks over time.
9. Allow traders to view their account balance and make deposits or withdrawals.
10. Implement error handling for insufficient funds, invalid stock symbols, or negative quantities.
11. You must have separate files for each class and main.

Your program should provide a user-friendly interface for traders to interact with the stock market simulation system. Consider implementing features like interactive menus, error messages, and clear instructions for navigating through the different functions.

Option 2: Text-Based RPG Game

Description: You must create a text-based RPG (role-playing game) program in C++ that simulates a dungeon adventure scenario. The program should allow players to explore the dungeon, battle monsters, gain experience points, and collect treasure.

Requirements:

1. Implement a Character class that represents the player character with attributes such as name, level, health points, attack points, and experience points.
2. Create a Dungeon class that generates a random layout for the dungeon with rooms and encounters.
3. Implement a Monster class that represents the enemies in the dungeon with attributes such as type, health points, and attack points.
4. The player can navigate the dungeon by moving between rooms and triggering encounters.
5. Implement a battle system that simulates turn-based combat between the player character and monsters.
6. Display battle outcomes, including damage dealt, health points remaining, and experience points gained.
7. Include a leveling system that allows the player character to gain experience points and level up with increased stats.
8. Include treasure chests in the dungeon that players can open to find loot, such as gold coins, weapons, or healing items.
9. Implement a game-over condition when the player character's health points reach zero.
10. Provide a scoring system based on the player's performance in the dungeon adventure.
11. You must have separate files for each class and the main.

Your program should provide players an immersive and interactive experience as they navigate the dungeon, battle monsters, and collect treasures. To make the gameplay more dynamic and enjoyable, consider implementing descriptive text, random events, and engaging combat mechanics.

Option 3: Client-Server Chat Application

Description: You are tasked with creating a client-server chat application in C++ that allows multiple clients to communicate with each other through a central server. The program should simulate a simple chat room environment where users can send and receive real-time messages.

Requirements:

1. .Implement a Server class as the central chat server, managing client connections and message routing.
2. Implement a Client class representing a user connected to the chat server, handling message sending and receiving.
3. Allow multiple clients to connect to the server and join a chat room.
4. Enable clients to send messages to the server, which then forwards the messages to all connected clients.
5. Implement a simple chat protocol for message formatting and routing between clients.
6. Include features for clients to specify their usernames when joining the chat room.
7. Display messages chronologically with timestamps to track the message history.
8. Implement error handling for disconnected clients, invalid messages, or server timeouts.
9. To ensure secure communication, provide basic security features like client authentication or message encryption.
10. You must have separate files for each class and the main.

Your program should provide a real-time chat experience for users to interact with each other through the client-server architecture. To enhance the user experience, consider implementing features like user prompts, message notifications, and chat room management functionalities.

Note: You can use socket programming in C++ to establish network connections between the server and clients. Implement server-side and client-side event handling to manage incoming and outgoing messages efficiently.

Option 4: Missile Defense System Simulation

Description: You are tasked with creating a program in C++ that simulates a missile defense system for detecting and intercepting incoming missiles. The program should model a real-world defense scenario where missiles are launched and intercepted by defense systems to minimize the threat.

Requirements:

1. Implement a Missile class representing an incoming missile with attributes such as speed, trajectory, and target coordinates.
2. Create a Defense System class representing a missile defense system with attributes such as radar range, interceptor missiles, and interception capabilities.
3. Simulate the detection of incoming missiles within the radar range of the defense system.
4. Implement a targeting algorithm to calculate the optimal interception trajectory for incoming missiles.
5. Trigger interceptor missiles to intercept incoming missiles based on the calculated trajectories.
6. Display simulation results, including successful interceptions, missile trajectories, and defense system performance.
7. Include system response time calculations for detecting and intercepting missiles.
8. Implement a scoring system based on the number of successful interceptions and defense system efficiency.
9. Include features for adjusting defense system parameters, such as radar sensitivity and interceptor missile accuracy.
10. Provide visualizations or graphics to represent the missile defense system simulation in real time.
11. You must have separate files for each class and main.

Your program should provide a realistic simulation of a missile defense system's operation and effectiveness in countering incoming threats. Consider implementing real-time data updates, graphical representations of missile trajectories, and interactive controls for adjusting defense system settings during the simulation.

Note: You can use physics calculations to simulate missile trajectories, collision detection algorithms for missile interceptions, and event-driven programming to handle real-time simulation updates and visualizations. You can also include user input mechanisms to trigger missile launches and interceptions during the simulation.

Option 5: File System Management Utility

Description: You are required to create a file system management utility program in C++ that simulates the basic functionalities of a file system. The program should allow users to create, delete, read, and write files, navigate through directories, and manage file permissions.
Requirements:

1. Implement a File class that represents a file with attributes such as name, size, permissions, and content.
2. Create a Directory class that represents a directory or folder with attributes such as name, contents (files and subdirectories), and parent directory.
3. Allow users to create new files in a specified directory, specifying the file name, contents, and permissions.
4. Allow users to create new directories within a specified directory, specifying the name and parent directory.
5. Implement file reading and writing functions to display a file's contents and write additional content to it.
6. Allow users to delete files and directories from the file system, including handling cases of nested directories and file dependencies.
7. Implement a navigation system to move between directories, listing the contents of the current directory.
8. Include file permission management features, such as setting read, write, and execute permissions for files and directories.
9. Implement error handling for cases such as file not found, permission denied, or disk full.

Your program should provide a user-friendly interface for users to interact with the file system management utility. Consider implementing command-line prompts, menu options, and clear instructions for users to navigate the different file system functions.

Note: You can use data structures like maps, vectors, or linked lists to represent the file system hierarchy and manage file operations. You can also utilize file-handling functions to read and write data to files on the system.