DEPARTMENT OF
COMPUTER
SCIENCE

FACULTY OF COMPUTING & ARTIFICIAL INTELLIGENCE

Course Guide

Programming Fundamentals (CS111)
2024 - 2025

Air University Islamabad

Course Description:

This course is designed as an introductory course on programming, using the imperative core of the C++ programming language. The students will learn basic problem solving and logic building skills and use them to implement basic C++ programs.

This course emphasis the basic concepts used in programming. The topics include Computer Programming; Basic Syntax & Semantics of a Higher-Level Language; Conditional & Iterative Control Structures; Functions & Parameter Passing; Recursion; Arrays; String Processing; Exception Handling; Refactoring; Debugging; Testing Fundamentals; File I/O; and User-defined data types such as Structure, Union, and Enum.

Pre-requisites

None

Relevant Graduate Attributes (GAs):

The course is designed so that students will achieve the GAs:

GA-01: Academic Education: To prepare graduates as computing professionals.

GA-02: Knowledge for solving computing problems: An ability to apply knowledge of mathematics, science, computing fundamentals and computing specialization to the solution of complex computing problems.

GA-03: Problem Analysis: An ability to identify, formulate, research literature, and analyze complex computing problems reaching substantiated conclusions using first principles of mathematics, natural sciences and computing sciences.

GA-04: Design/Development of Solutions: An ability to design solutions for complex computing problems and design systems, components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations.

Course Learning Outcomes:

Upon successful completion of the course, the student will be able to:

Sr. No	CLO	Domain	Taxonomy level	GA
CLO-01	**Understand** the fundamental concepts of programming	Cognitive	C2	GA-01
CLO-02	**Apply** basic programming concepts using a programming language	Cognitive	C3	GA-02
CLO-03	**Develop** program for data processing and file handling	Cognitive	C3	GA-03
CLO-04	**Use** user-defined data types to solve real-world computing problem	Cognitive	C3	GA-04

*Structure, Union, and Enum should be taught to achieve the CLO-04

Assessment of CLOs:

Performance related to each CLO is assessed as follows

CLOs	Assessments				Taxonomy level	GA
	Quizzes	Assignments	Midterm	Final		
CLO-01	✓ [Quiz 1]		✓ [Q1 20%]	✓ [Q1 20%]	C2	GA-01
CLO-02	✓ [Quiz 2]	✓ [Assg 1] ✓ [Assg 2]	✓ [Q2 50%]	✓ [Q2 40%]	C3	GA-02
CLO-03	✓ [Quiz 3]	✓ [Assg 3]	✓ [Q3 30%]	✓ [Q3 20%]	C3	GA-03
CLO-04	✓ [Quiz 4]	✓ [Assg 4]		✓ [Q4 20%]	C3	GA-04

Course Outline with Week Breakdown:

W#	Contents	Assessment	Activity	Reading Material
1	**Course Introduction** •Outline, grading **Introduction to Programming** •Why to Program? •Programs and Programming Languages •Intro to C++ language • What is a program made of? • Diagrams and Flowcharts **Logic building** • Understanding the development of a step-by-step process for performing a real-world task • Your first program • Source to machine code translation- the compilation process • Input, Processing, and Output The Programming Process • The Parts of a C++ Program • The cout/cin Object • The header files • The using Directive • The # define directive • Single line comments	*Reading Assignment*	*For Activities and case studies, please refer to Appendix A*	Slides + Code Examples Ch1 of text book

	• Multi-line comment •Variables, Literals and Identifiers • The C++ string Class • int-Floating-Point Data Types • The bool Data Type • Determining the Size of a Data Type • Variable Assignments and Initialization •Scope •Arithmetic Operators •Named Constants •Programming Style **Expressions and Interactivity** •The cout/cin Object •Cascading << •Mathematical Expressions •Operator precedence			
2	**Introduction to C++** •Input and output with files •Opening, reading, and writing text files •fstream library •detecting and handling EOF •appending data to an existing file •Type Conversion o Overflow and Underflow o Type Casting o Converting Floating-Point Numbers into Integers o Avoiding Negative Remainders •Multiple Assignment and Combined Assignment •Formatting Output •Working with Characters and string Objects •Focus on Debugging: Hand Tracing a Program •Common program errors • Focus on Problem Solving: Case Study	Assignment#1	*For Activities and case studies, please refer to Appendix A*	Slides + Code Examples Ch1-Ch2 of text book
3	**Making Decisions** •Relational Operators •The if Statement •Expanding the if Statement •The if/else Statement •Nested if Statements •The if/else if Statement •Dangling else	Quiz1		Slides+ Code Examples Ch2 of text book

	• Flags • Logical Operators • Checking Numeric Ranges with Logical Operators • Menus ○ Validating User Input • Comparing Characters and Strings • The Conditional Operator • The switch Statement			
4	**Loops** • Introductions to Loops & Iterations • The Increment and Decrement Operators • Introduction to Loops: The while Loop • Using the while Loop for Input Validation • Counters • The do-while Loop • The for Loop • Sentinels Nested Loops		*For Activities and case studies, please refer to Appendix A*	Slides + Code Examples Ch2-Ch3 of text book
5	**Decision Making + Loops** • Deciding Which Loop to Use **Using Files for Data Storage** • Files: What is Needed • Opening Files • Testing for File Open Errors • Using Files	Assignment#2		Slides +Code Examples Ch3 of text book
6	**Functions** • Defining and Calling Functions • Function Prototypes • Sending Data into a Function • Passing Data by Value • Using Functions in a Menu-Driven Program • The return Statement • Functions for factorial of a number	Quiz2		Slides +Code Examples Ch4 of text book
7	**Functions** • Returning a Value from a Function • Returning a Boolean Value • Local and Global Variables ○ Static Local Variables • Default Arguments • Using Reference Variables as Parameters • Overloading Functions • The exit() Function			Ch4 of text book

	• Inline Function • Recursion and Recursive functions • Recursive vs. iterative solutions • Solving problems recursively (Fibonacci)			
Midterm exam				
9	**Arrays** • Introduction to Arrays • Declaring a 1-D array • Accessing Array Elements • No bounds checking in C++ • Buffer overflow • Arrays definition, initialization • The Range-Based for Loop • Processing Array Contents			Slides + Code Examples Ch4-Ch5 of text book
10	**Arrays** • Arrays as Function Arguments • 2-D Arrays • Processing Array Contents • Using Parallel Arrays • Search in arrays • Sorting arrays • Vector arrays • **3D array**	Assignment#3	*For Activities and case studies, please refer to Appendix A*	Slides +Code Example Ch5 of text book
11	**Pointers** • Introduction to Pointers • Getting the Address of a Variable • Pointer Variables • The Relationship Between Arrays and Pointers • Pointer Arithmetic • Initializing Pointers • Comparing Pointers	Quiz3		Slides +Code Example Ch6 of text book
12	**Pointers** • Pointers as Function Parameters • Dynamic Memory Allocation • Pointers as function parameters			
13	**Structures** • Structure Declaration, Initialization, usage • Arrays of Structures, Nested Structures • Enumerated data types	Assignment#4		Slides + Code Examples Ch6 of text book

14	**Structures** • Structures usage with Functions • Pointers to Structures, Usage • Unions	Quiz4	*For Activities and case studies, please refer to Appendix A*	Slides + Code Examples Ch7 of text book
15	Searching Techniques in C++ • Concept of linear search • Implementation of linear search • Concept of Binary search • Implementation of Binary search • Concept of Bubble sort • Implementation of Bubble sort • Concept of Insertion sort • Implementation of Insertion sort			Slides + Code Examples Ch9 of text book Ch11 of text book
Final exam				

Please note: This is a proposed schedule only and may be varied at the discretion of the instructor to give a greater or lesser degree of emphasis to particular topics.

Books / Reference Materials:

Text Book:

- "Starting Out with C++ from Control Structures through Objects" by Tony Gaddis, Pearson
- "C++ How to Program" by Paul Deitel and Harvey Deitel, Pearson Education

Reference Books:

- "Object-Oriented Programming in C++" (4th Edition) by Robert Lafore, Sams publishing 2002

Online Sources:

- Coursera Programming Fundamentals Offered By Duke University, https://www.coursera.org/learn/programming-fundamentals#about

Note: (soft form of the books are uploaded on GCR)

Software Tools/ IDEs:

MS Windows OS:

- Visual C++, Visual Code, Visual Studio
- Eclipse IDE for C/C++ Developers

Linux & MacOS

- Eclipse IDE for C/C++ Developers

Note: Dev C++ is not encouraged to use.

General Grading Policy:

tentative assessment's weightage is as under:

Item	Assessment Task	Weightage
1.	Quizzes [cnt: 4]	15%

2.	Assignments [cnt: 4]	15%
3.	Midterm	25%
4.	Final	45%

Grading and General Course Policies:

- Assignments and/or grade percentages are subject to change.
- There will be no makeup quizzes / assignments
- No late assignments will be accepted.
- All assignments and projects submitted should be the outcome of individual work only. Group work is explicitly prohibited (severe penalties for violation).
- **An 'F' grade will be allotted if projects/Assignments are found copied from internet or any other sources**

APPENDIX – A

Week 01

Case study 01:

Programming Languages are like different cuisines (e.g., French, Italian, Chinese). Each cuisine has its rules and ingredients, just like programming languages have their syntax and libraries.

C++ is like a versatile kitchen with lots of tools (libraries) and the ability to cook different types of dishes (applications). The "Hello World" program is equivalent to a basic recipe, like making a sandwich. It's simple but essential for understanding the process.

Case study 02:

Imagine you're a tour guide leading a group through a maze-like forest. Your goal is to get everyone safely to the exit. To do this, you must think step by step, considering each path's consequences. If you choose the wrong path, the group might get lost.

Similarly, in programming, logic is like being a tour guide for the computer. You provide clear, step-by-step instructions (algorithm) to reach a specific outcome (goal). Just like in the forest, a wrong decision (flawed logic) can lead to errors or unexpected results.

Case study 03:

To communicate the app's structure and logic effectively, you use UML and flowcharts:

Diagram: You create a diagram that represents the app's core components, such as user interface (UI), data storage, and communication modules. This visual representation helps your client understand the app's structure and relationships between different parts.

Flowchart: You design a flowchart to illustrate the app's user journey, from registration to using various features. The flowchart maps out the sequence of screens, user interactions, and decision points. This helps your client visualize the app's workflow and user experience.

By using UML and flowcharts in your project, you ensure that both you and your client have a clear understanding of the app's design, which aids in efficient development and minimizes misunderstandings during the project's execution.

Week 02

Case Study 01:

Imagine you are building a simple calculator app in C++. Users can enter two numbers and choose an operation (addition, subtraction, multiplication, or division). Your program should display the result.

Case study 02:

Imagine you are a cashier at a grocery store. You need to keep track of the total cost of items in a shopping cart. You can use variables to do this:

- **Variables**: You introduce variables like itemPrice and totalCost to represent the price of each item and the running total.
- **Literals**: You explain that literals are like the price tags on items, such as $2.50 for a can of soup.
- **Identifiers**: Identifiers are like product names; they help you distinguish items. For example, itemPrice could represent the price of "Soup."

Case study 03:

You are a software developer working on a payroll system. You illustrate the practical applications of multiple assignment and formatting output. Imagine you need to calculate and print paychecks for employees. Each paycheck contains the employee's name, hours worked, hourly rate, and total pay.

Case study 04:

You are a web developer creating a registration system that validates user inputs. You emphasize the practicality of working with characters and string objects for input validation. Imagine you are building a user registration system. Users must provide valid email addresses, usernames, and passwords.

Week 03

Case study 01:

You are a traffic engineer designing a smart traffic light system to manage traffic flow. Imagine you are designing a smart traffic light system that detects the number of vehicles waiting at an intersection. Based on this count, the system decides whether to change the traffic light.

Case study 02:

You are a restaurant manager developing a menu ordering system. Imagine you are managing a restaurant, and you need to implement a menu ordering system. Customers can choose from various menu items, and you need to handle special requests and discounts.

Case study 03:

You are a game developer creating a text-based adventure game. Imagine you are developing a text-based adventure game where players make choices by typing commands. You need to compare user input to predefined commands and navigate the game accordingly.

Week 04

Case study 01:

Imagine you are developing a simulation program for a robotics competition. Robots need to perform specific actions, such as navigating a maze or picking up objects, repeatedly until they complete their tasks.

Case study 02:

You are a cashier at a retail store implementing a sales system. Imagine you are working as a cashier, and you need to ensure that customers enter valid coupon codes for discounts during checkout. You want to repeatedly prompt customers for the code until they provide a valid one.

Case study 03:

You are a game developer creating a chess-playing program. Imagine you are developing a chess-playing program where you need to iterate through the chessboard's rows and columns to make valid moves and check for win conditions.

Week 05

Case study 01:

You are a software engineer designing a weather monitoring system. Imagine you are designing a weather monitoring system to collect temperature data at regular intervals. Depending on the data collection requirements, you may need to choose between different loops to efficiently gather the data.

Case study 02:

You are a data analyst developing a program to process and analyze a large dataset. Imagine you are tasked with analyzing a massive dataset containing sales records for a retail company. To efficiently manage and process this data, you need to read and write it to files.

Case study 03:

You are a financial analyst developing a program to manage investment portfolios. Imagine you are developing a program that allows users to manage their investment portfolios. To provide a seamless user experience, you need to save and load portfolio data to and from files.

Case study 04:

You are a software developer working on a financial application. Imagine you are developing a financial application that processes transactions. However, sometimes these transactions might result in errors, such as division by zero or invalid input. To handle these errors gracefully, you need to use try-catch blocks.

Case study 05:

You are a game developer working on a multiplayer online game. Imagine you are developing a multiplayer online game where players can interact and trade items. To ensure fair gameplay, you need to handle exceptions when trading items, such as checking for item availability and invalid trades.

Case study 06:

You are a software engineer developing a file management application. Imagine you are building a file management application where users can create, edit, and delete files. To ensure that resources, such as file handles, are managed correctly and released when exceptions occur, you need to use exception-safe resource management techniques.

Week 06

Case study 01:

You are a chef managing a restaurant kitchen. Imagine you are a chef in a restaurant, and you have various recipes to prepare. Each recipe consists of a set of instructions. To keep your kitchen organized and efficient, you use functions to encapsulate each recipe.

Case study 02:

You are a librarian managing a library system. Imagine you are managing a library, and you have a catalog of books that you need to organize and search. To efficiently handle various library operations, you use functions.

Case study 03:

Imagine you are managing a library, and you have a catalog of books that you need to organize and search. To efficiently handle various library operations, you use functions. Imagine you are managing a library, and you have a catalog of books that you need to organize and search. To efficiently handle various library operations, you use functions.

Week 07

Case study 01:

Imagine you are developing a calculator application that can perform various operations such as addition, subtraction, multiplication, and division. To provide a user-friendly experience, you want to use function overloading to create different versions of the calculate function for each operation.

Case study 02:

Imagine you are developing a text-based RPG game with various characters, quests, and battles. To keep your code organized and maintainable, you use functions to modularize different aspects of the game.

Week 09

Case study 01:

Imagine you are a teacher responsible for recording and analyzing exam scores for a class of students. To efficiently manage and process the scores, you use arrays to store and manipulate the data.

Case study 02:

Imagine you are managing inventory for a retail store with various products and quantities. To keep track of the available items and their stock levels, you use arrays to represent the store's inventory.

Case study 03:

Imagine you are analyzing survey responses from participants, and you need to store their answers in an array. However, you must also ensure that the input data is managed safely to prevent buffer overflow.

Week 10

Case study 01:

Imagine you are responsible for maintaining student records, including names, IDs, and grades. To efficiently manage this data, you use arrays as function arguments to perform operations such as calculating averages and displaying student information.

Case study 02:

Imagine you are developing a calendar application that displays events and appointments for each day of the month. To represent the calendar's structure and efficiently store event data, you use a 2-D array.

Case study 03:

Imagine you are responsible for keeping track of employee information, such as names, IDs, salaries, and departments. To efficiently organize and process this data, you use parallel arrays, where each array corresponds to a specific attribute.

Week 11

Case study 01:

Imagine you are developing an optimization algorithm that involves swapping the values of two variables. To optimize memory usage and speed, you use pointers to perform the variable swap operation.

Case study 02:

Imagine you are developing a text editor that allows users to create and edit documents of varying lengths. To handle text of unknown size, you use dynamic memory allocation with pointers to allocate memory as needed.

Case study 03:

Imagine you are conducting a scientific simulation that involves complex data structures, such as multidimensional arrays. To efficiently manipulate and process data, you use pointers to access elements within these arrays.

Week 12

Case study 01:

Imagine you are responsible for maintaining comprehensive student records, including names, IDs, grades, and contact information. To efficiently manage this data, you use structures to represent each student's information as a single entity.

Case study 02:

Imagine you are managing inventory for a retail store with various products, each having multiple attributes like name, ID, price, and stock level. To efficiently organize and track this data, you use structures and arrays of structures.

Case study 03:

Imagine you are developing a calendar application that displays events and appointments for each day of the week. To represent the days of the week in a structured and understandable manner, you use enumerated data types.

Week 13

Case study 01:

Imagine you are responsible for managing employee information, including names, IDs, salaries, and departments. To efficiently organize and process this data, you use structures with functions to perform operations such as adding new employees, updating information, and generating reports.

Case study 02:

Imagine you are responsible for maintaining detailed student records, including names, IDs, grades, and contact information. To optimize memory usage and data manipulation, you use pointers to structures to represent and access student information.

Case study 03:

Imagine you are conducting experiments and simulations that involve various data types, such as integers, floating-point numbers, and text descriptions. To efficiently store and represent this diverse data, you use unions to conserve memory and facilitate data conversion.

Week 14

Case study 01:

Imagine you are working on a text processing application that needs to perform tasks such as identifying and converting characters to uppercase or lowercase. To achieve this, you use character testing and case conversion functions.

Case study 02:

Imagine you are analyzing textual data that requires tasks like counting words, finding substrings, or extracting specific information. To efficiently perform text analysis, you use C-Strings and library functions.

Case study 03:

Imagine you are creating a file processing utility that reads and modifies text files. To efficiently handle file operations, including reading and writing text data, you use C-Strings to process file content.

Week 15

Case study 01:

Imagine you are creating a phonebook application that allows users to add, search for, and display contact information. To efficiently manage contacts, you use functions for sorting and searching.

Case study 02:

Imagine you have a database of student records, and you need to search for a specific student's information based on their ID. To find the student efficiently, you implement a linear search algorithm.

Case study 03:

Imagine you are developing a contact management application that stores contacts in sorted order. To efficiently search for a contact by name, you implement a binary search algorithm.

Case study 04:

Imagine you have a list of student scores that need to be sorted in ascending order. To accomplish this, you implement both bubble sort and insertion sort algorithms for educational purposes.