CS-1201 Object Oriented Programming

Introduction

Arbish Akram

Department of Computer Science
Government College University

About Course

Code	1201
Title	Object Oriented Programming (OOP)
Credit Hours	3+1
Semester	Fall 2024
Prerequisite	Programming Fundamental (PF)

Recommended Books

1. **Object-Oriented Programming in C++** by Robert Lafore, 4th Edition
2. **C++ How to Program**, 8th Edition, by Paul J. Deitel and Harvey Deitel
3. **C++ Programming**, by D.S. Malik, 5th Edition

Quizzes

- Quizzes can be announced or unannounced.
- There is no retake for any quiz.
- If you miss a quiz, do not send an email to request a makeup. Sending such a request will result in a deduction of 5 marks.
- If you are found discussing or cheating during a quiz, 20 marks will be deducted.

Assignments

- Instructions in each assignment must be followed carefully.
- Failure to follow instructions will result in a score of 0.
- Requests for changes or resubmissions will lead to a deduction of **5 marks**.
- Deadlines for assignments will not be extended.
- ChatGPT or other solutions should not be used.
- The goal is to learn programming through practice; retry problems if you fail to solve them.
- A penalty of 20 marks will be imposed if solutions are found to be copied from ChatGPT or any other source.

Labs

- Discussions, online solutions, and ChatGPT are not allowed.
- If caught using such resources, you will receive a score of 0 for that lab and will be unable to attend the next two labs.

Attendance

- Maintaining at least 80% attendance is required to be eligible for the final exam.
- No favors regarding attendance will be granted.
- If you miss attendance during roll call, it will not be marked again.
- Do not request to mark attendance after roll call, whether in person or via email.

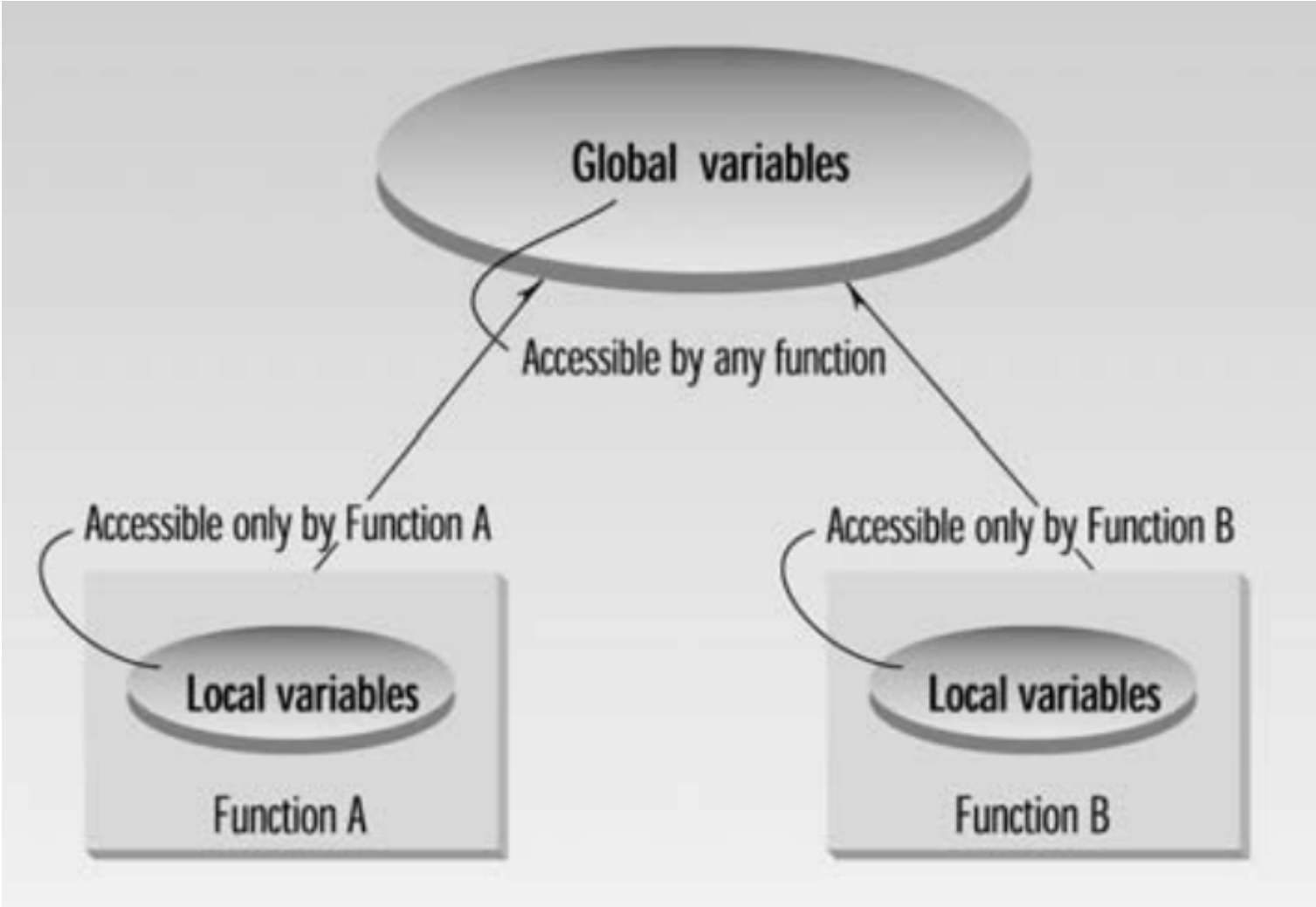
Procedural Programming

- Utilizes a sequence of instructions where each statement directs specific tasks (e.g., input, calculations, output).
- For small programs, a simple sequence of instructions is sufficient; no additional organizing principles are needed.
- Large programs become difficult to manage with a single list of instructions.
- *Functions* are used to break down programs into smaller, manageable units.
- In other languages, this concept may be called subroutines, subprograms, or procedures.
- Each function should have a clear purpose and a well-defined interface with other functions.

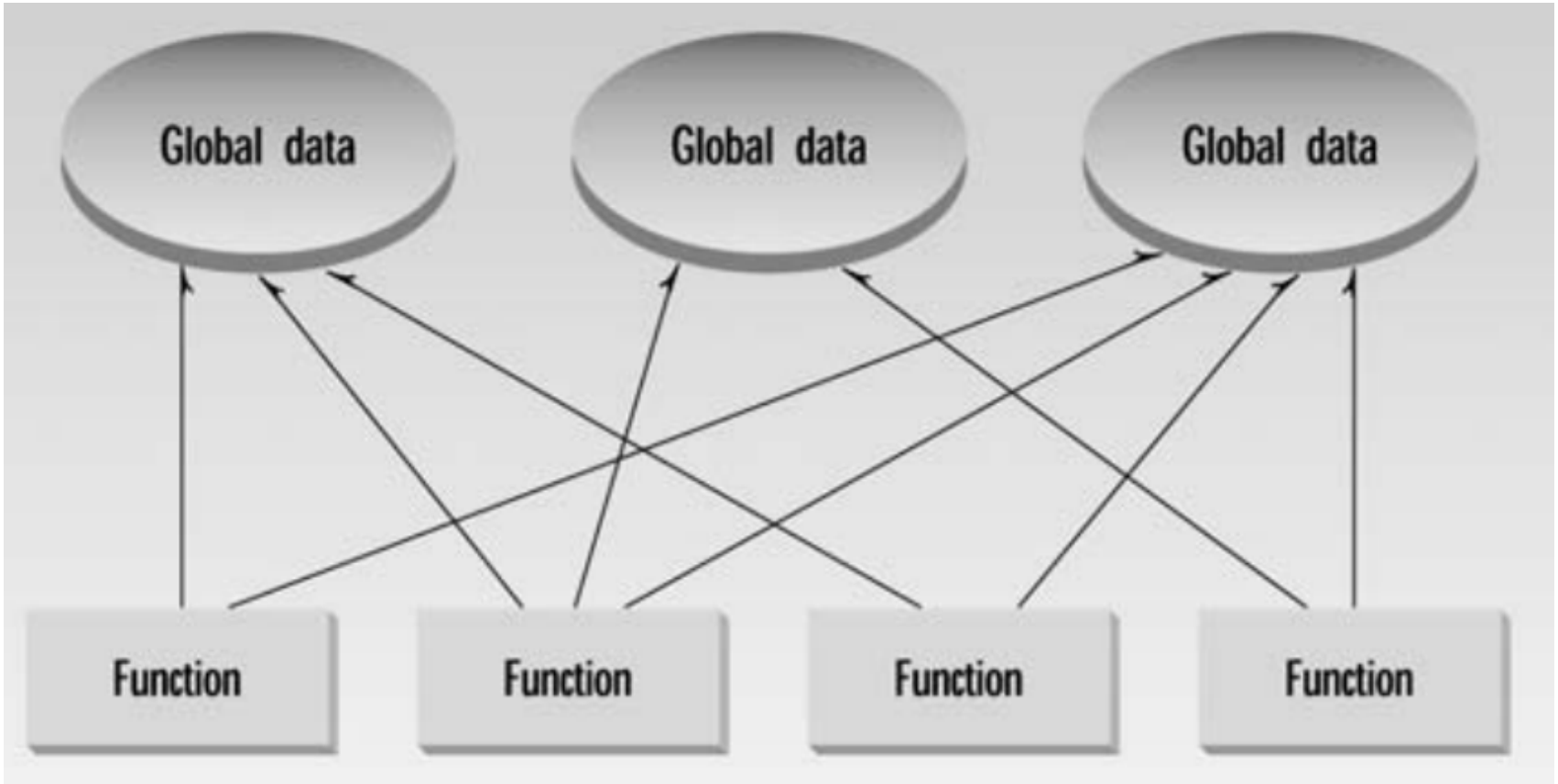
Unrestricted Access in Procedural Programming

- **Local Data**: Used exclusively within a function; protected from modification by other functions.
- **Global Data**: Accessible by any function; necessary for shared data but introduces risks.
- **Issues with Global Data**:
  - **Complexity**: Large number of connections between functions and data.
  - **Difficulty in Modification**: Changes in global data require updates to all related functions.
- **Example**: Changing data types (e.g., from short to long) affects all functions accessing that data.

Global and local data



Procedural programming



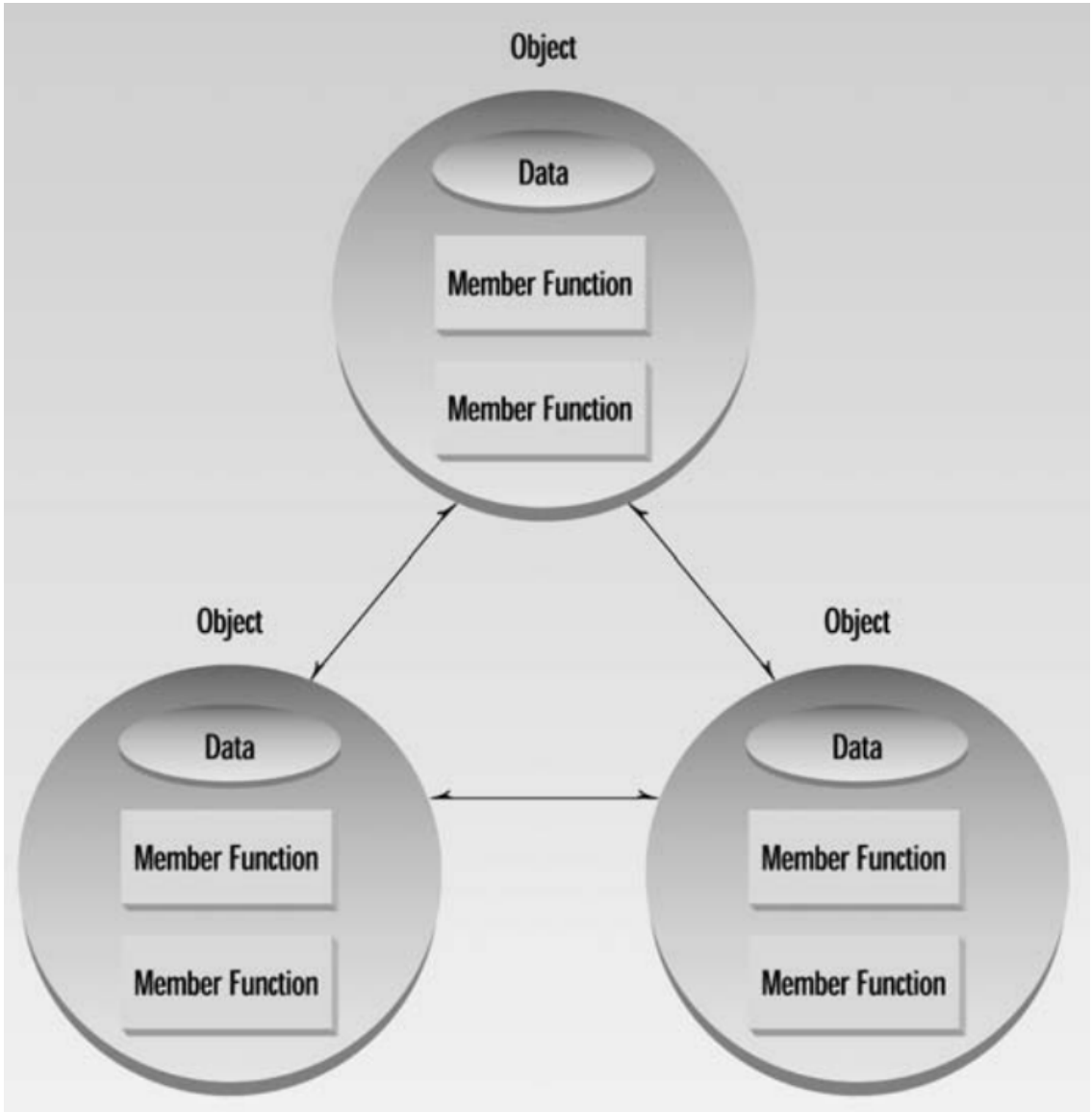
Real-World Modeling and Procedural Programming

- **Problem**: Procedural programming struggles to model real-world objects effectively.
- **Real-World Objects**: Have both *attributes* and *behavior*.
  - Attributes: Characteristics such as eye color (people) or horsepower (cars). Equivalent to data in programs.
  - Behavior: Actions in response to stimuli, like a boss's response to a raise request or a car stopping when brakes are applied. Equivalent to functions in programs.
- **Limitation**: Data and functions alone dont fully capture the complexity of real-world objects, which combine attributes and behavior.

Object Oriented Programming

- The real world consists of objects.
- Computer programs may contain computer world representations of the things (objects) that constitute the solutions of real world problems.
- Real-world objects have two parts:
  - Properties or state: characteristics that can change
  - Behavior or abilities: things they can do
- To solve a programming problem in an object-oriented language, the programmer no longer asks how the problem will be divided into functions, but how it will be divided into objects.
- The emphasis is on data.

Object Oriented Paradigm



Object Oriented Paradigm

- A program typically consists of a number of objects.
- These objects communicate by calling each others member functions.
- Member functions are equivalent to methods in other object oriented (OO) languages (e.g., Smalltalk).
- Data items are referred to as attributes or instance variables.
- Calling a member function of an object is often referred to as sending a message to the object.

Four pillars of OOP

1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

Abstraction

- Abstraction simplifies complex systems by *exposing only essential details and hiding unnecessary ones*.
- Reduces complexity by focusing on the critical aspects relevant to the user.
- Examples:
  - Online Shopping: When you shop online, you simply add items to your cart and make a payment. Behind the scenes, various complex processes such as inventory management, order processing, and payment verification are happening, but they are hidden from the customer.
  - Using a Smartphone: When you use a smartphone, you interact with apps to make a call, send messages, or browse the internet. However, internal processes such as network communication, data encryption, and memory management are abstracted from the user.

Encapsulation

- Encapsulation is the practice of *wrapping data* (attributes) and *methods* that operate on that data i*nto a single unit* (object).
- Protects the data by *restricting direct access* and provides *controlled interfaces for interaction*.
- Examples:
  - Banking system: In a banking system, your account balance is private, and you can only interact with it through defined methods like deposit() or withdraw(). You cant directly modify the balance variable.
  - Smartphone: You use apps like the camera or messages, but how the phone processes these tasks (e.g., image processing or data handling) is hidden from the user. The only access you have is through the user-friendly interface.

Inheritance

- Inheritance allows a new class (child) to inherit properties and behaviors from an existing class (parent).
- Enables code reusability and creates a hierarchical relationship among classes.
- Examples:
  - Vehicles: A Vehicle class may define common characteristics like wheels, engine, and speed. A Car class inherits these attributes but adds unique behaviors like having air conditioning or a stereo system.
  - Electronic Devices: A Device class could include shared attributes such as power source and manufacturer. A Smartphone class could inherit these features and add specific properties like a touchscreen, camera, and apps.

Polymorphism

- Polymorphism allows objects of different types to be treated as objects of a common parent class.
- Types:
  1. Compile-time Polymorphism: The same method name can be used with different parameters (method overloading).
  2. Run-time Polymorphism: A method in a subclass can provide a specific implementation of a method already defined in its parent class (method overriding).
- Remote Control: A remote can operate multiple devices (TV, AC, or sound system). Pressing the "power" button has a different effect depending on whether its the TV or AC, but its the same button.
- Payment System: In an online payment system, you can pay via credit card, PayPal, or cryptocurrency. Regardless of the payment method, the system processes the payment, but each method has its own underlying process.

Why OOP?

- Reducing the effort, complexity, and cost of development and maintenance of software systems.
- Reducing the time to adapt an existing system (quicker reaction to changes in the business environment).
- Flexibility, reusability.
- Increasing the reliability of the system.