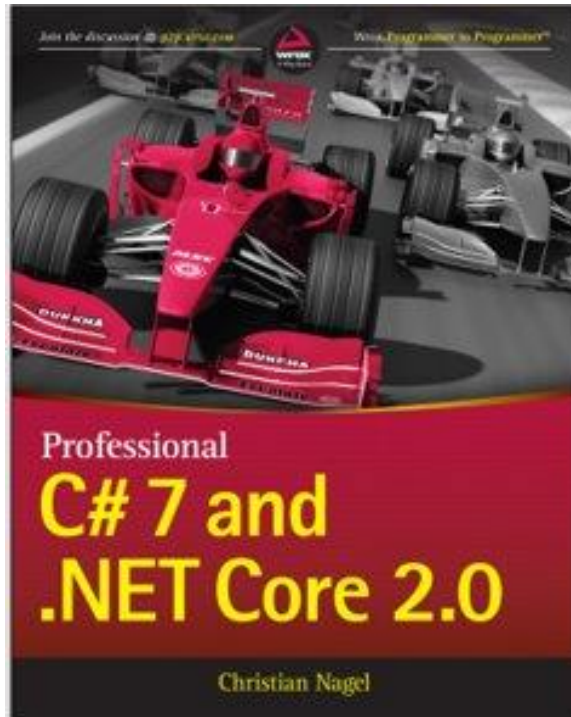


Programming

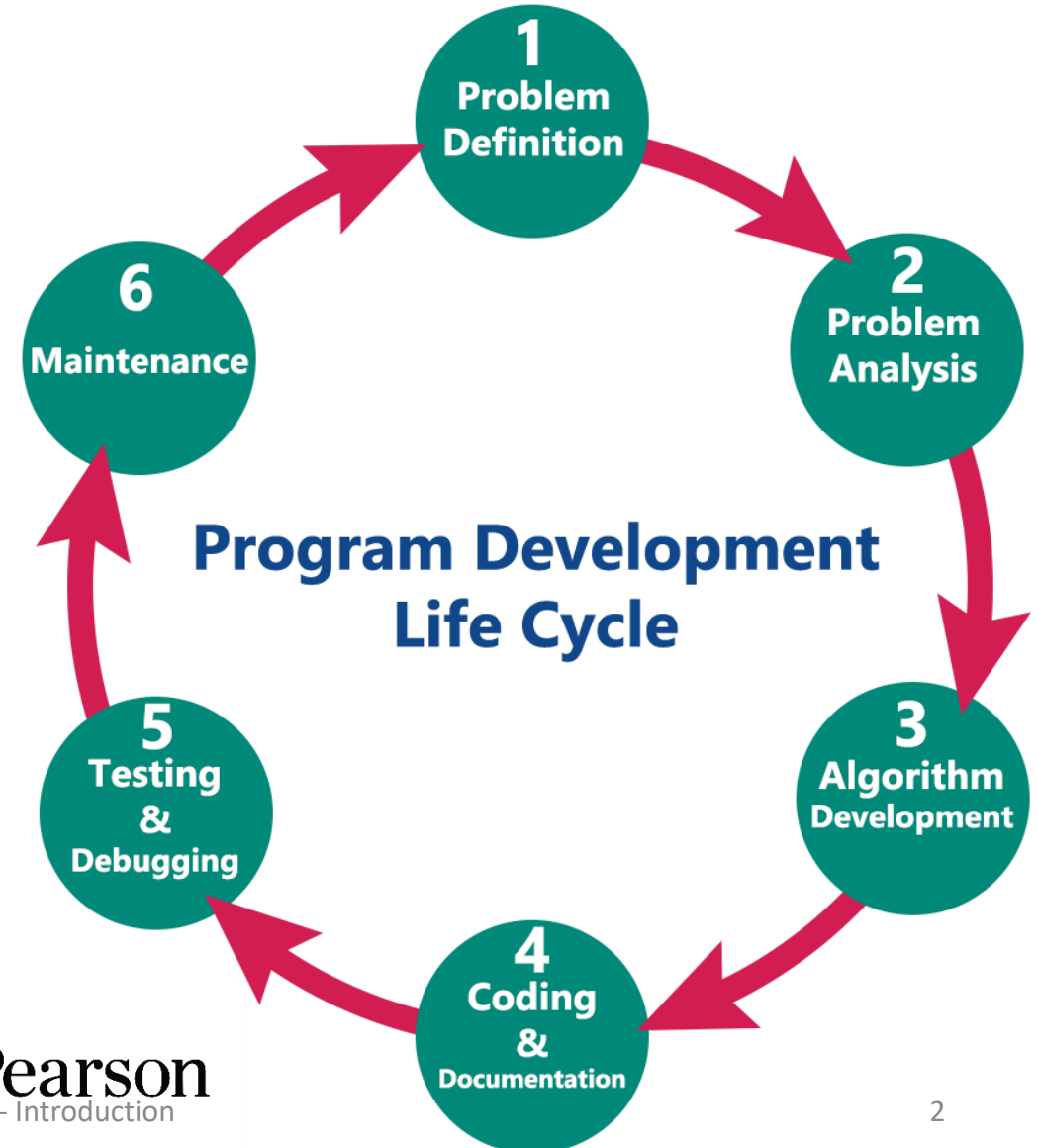


Chapter 2

Steps in program
development

Steps in Program Development

- The various steps involved are
 - Defining or Analyzing the problem
 - Design (Algorithm)
 - Coding
 - Documenting the program
 - Compiling and running the program
 - Testing and Debugging
 - Maintenance

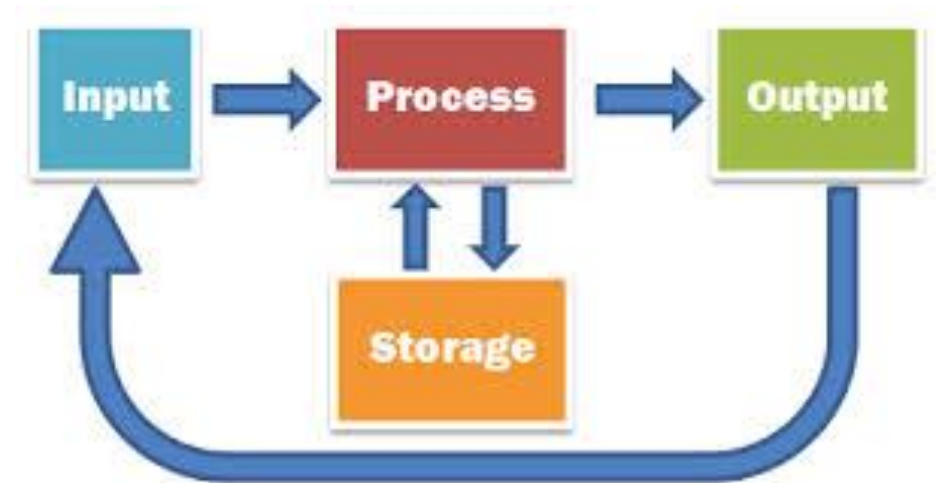


Steps in Program Development: Analyzing or Defining the Problem

- The problem is defined by doing a preliminary investigation
- Defining a problem helps us to understand problem clearly
- It is also known as Program Analysis

Tasks in defining a problem

- Followings are the tasks in order to define a problem
 - Specifying the input requirements
 - Specifying the output requirements
 - Specifying the processing requirements



Specifying the input requirements

- The input specification is obtained by answering following questions
 - What specific values will be provided as input to the program?
 - What format will the values be?
 - For each input item, what is the valid range of values that it may assume?
 - What restrictions are placed on the use of these values?

Specifying the output requirements

- The output specification is obtained by answering the following questions
 - What values will be produced?
 - What is the format of these values?
 - What specific annotation, headings, or titles are required in the report?
 - What is the amount of output that will be produced?

Specifying the processing requirements

- The processing requirement is obtained by answering following questions
 - What is the method (technique) required in producing the desired output?
 - What are the validation checks that need to be applied to the input data?
 - What calculations are needed?

Activity 1: Find Factorial Number

- Input?
- Output?
- Process?

Activity 2: Sum of numbers from 1 to n

- Input?
- Output?
- Process?

Activity 1: Find Factorial Number

- **Input:** Positive integer number
- **Output:** Factorial of that number
- **Process:** Solution technique which transforms input to output.
Factorial of a number can be calculated by the formula
$$n! = 1 * 2 * 3 * \dots * n$$








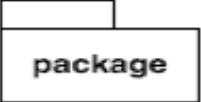
Activity 2: Sum of numbers from 1 to n

- **Input:** Positive integer number
- **Output:** Sum of numbers from 1 to n
- **Process:** 2 methods
 - ✓ **Method A:** You can add them up. It might take a while, depending on what N is, but let's practice with $N = 7$. $1+2+3+4+5+6+7 = 28$.
 - ✓ **Method B:** You can look into a general case to try to find a formula: $\frac{N(N+1)}{2}$

Use-case diagram

- Use case diagrams are used to gather the requirements of a system
- So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.
- The purposes of use case diagrams can be as follows:
 - Used to gather requirements of a system
 - Used to get an outside view of a system
 - Identify external and internal factors influencing the system
 - Show the interacting among the requirements are actors

Main elements of use-case diagram

Element	Description	Symbol
Actor	An actor is a person, organization, or external system that plays a role in one or more interactions with your system	
Use-cases	A use case describes a sequence of actions that provide something of measurable value to an actor	
Associations	Associations between actors and use cases: An association exists whenever an actor is involved with an interaction described by a use case	
Some other relationships	The relationships among actors, or use-cases like inheritance, extends, includes, etc.	  
System boundary boxes (optional)	A rectangle around the use cases, called the system boundary box, to indicates the scope of your system.	
Packages (optional)	UML constructs that enable you to organize model elements (such as use cases) into groups	

Activity: Draw use-case diagram

You are hired to develop FAI's library system with following description

- Admin who could
 - Manage books, readers (staff, lecturers, and students), etc.
 - Manage borrow/return books
- Users (staff, lecturers, students) who could
 - View/search books
 - Reserve books
 - Borrow/return books

Please draw Use-case diagram for this scenario

STEPS IN Program Development: Design

- A design is the path from the problem to a solution in code
- The well designed program is likely to be:
 - Easier to read and understand later
 - Less of bugs and errors
 - Easier to extend to add new features
 - Easier to program in the first place

Modular Design

- Once the problem is defined clearly, several design methodologies can be applied
- An important approach is Top-Down program design
- It is structured design technique
 - It breaks up the problem into a set of sub-problems called Modules
 - It creates a hierarchical structure of the modules

STEPS IN Program Development: Coding

- An algorithm expressed in programming languages is called Program
- Writing a program is called Coding
- The logic that has been developed in the algorithm is used to write program

STEPS IN Program Development: Documenting the Program

- Document explains
 - How the program works and how to use the program (user manual)
 - How to maintain the program (developer manual)
- Details of particular programs, or particular pieces of programs, are easily forgotten or confused without suitable documentation

Forms of documentation

- Documentation comes in two forms
 - External documentation, which includes things such as reference manuals, algorithm descriptions, flowcharts, and project workbooks
 - Internal documentation, which is part of the source code itself (essentially, the declarations, statements, and comments)

Compiling and Executing the Program

- Compilation is a process of translating a source program into machine understandable form
- The compiler is system software
 - It examines each instruction for its correctness
 - It does the translation
- During the execution
 - Program is loaded into the computer's memory
 - The program instructions are executed

STEPS IN Program Development: Testing

- Testing is the process of executing a program with the deliberate intent of finding errors
- Testing is needed to check whether the expected output matches the actual output
- Testing is done during every phase of program development
- Initially, requirements can be tested for its correctness
- Then, the design (algorithm, flow charts) can be tested for its exactness and efficiency

Test criteria

- Programs are tested with several test criteria and the important ones are given below
 - Test whether *each and every statement* in the program *is executed* at least one (*Basic path testing*)
 - Test whether *every branch* in the program *is traversed* at least once (control flow)
 - Test whether the *input* data flows through the program and is *converted* to an *output* (*data flow*)

STEPS IN Program Development: Debugging

- Debugging is a process of correcting the errors
 - Programs may have logical errors which cannot be caught during compilation
 - Debugging is the process of identifying their root causes
 - One of the ways is to print out the intermediate results at strategic points of computation
 - Another way is to use support from the IDE
- Testing vs Debugging
 - Testing means detecting errors
 - Debugging means diagnosing and correcting the root causes

STEPS IN Program Development: Maintenance

- Program maintenance
 - Continuing process of maintenance and modification
 - To keep pace with changing requirements and technologies
- Maintainability of the program is achieved by
 - Modularizing it
 - Providing proper documentation for it
 - Following standards and conventions (naming conventions, using symbolic constants, etc.)

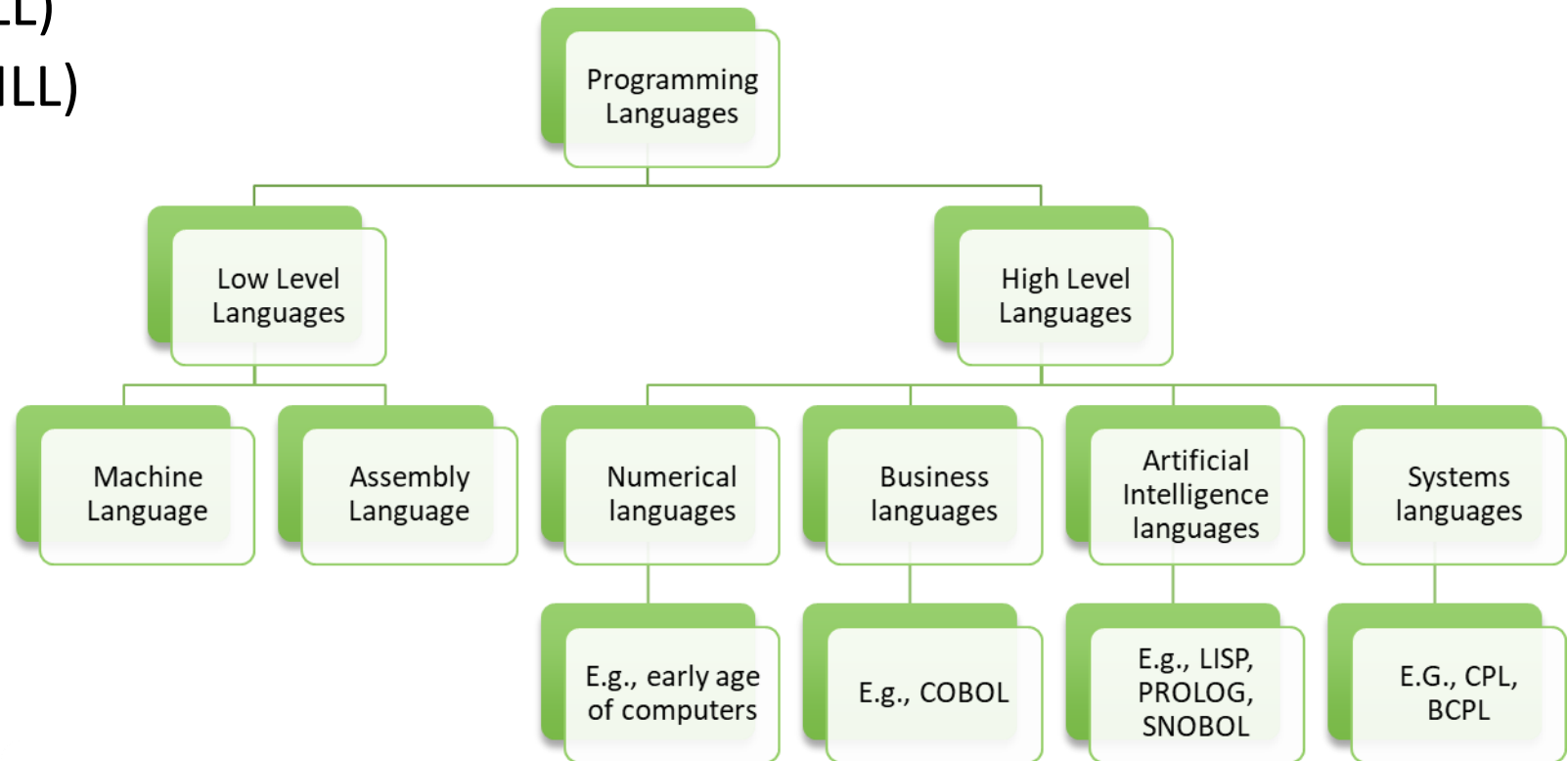
Introduction to programming language

What is a Programming Language?

- It is an art of making a computer to do the required operations
 - By means of issuing sequence of commands to it
- It can be defined as
 - A vocabulary (unique set of characters/keywords)
 - A set of grammatical rules (syntax)
- The term programming languages usually refers to high-level languages
 - E.g., BASIC, C, C++, COBOL, FORTRAN, Ada, and Pascal

Types of programming languages

- There are two major types of programming languages
 - Low level languages (LLL)
 - High level languages (HLL)

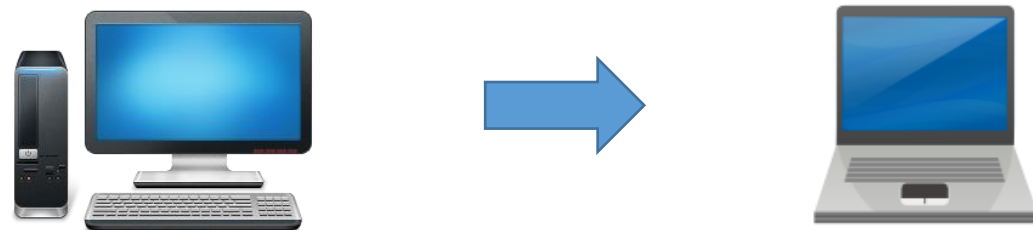


What makes a good language?

- Every language has its strengths and weaknesses
- FORTRAN is good for numeric data but not good to organize large program
- PASCAL is good for structured and readable programs, but it is not as flexible as C
- C++ has powerful object-oriented features, but it is complex and difficult to learn
- The choice of PL depends on type of the computer used, type of program, and the expertise of the programmer

Development Environments

- Programming on Host Environment:
 - The environment under which a program is designed, coded, tested & debugged
- Operating on Target Environment
 - The external environment which supports the execution of a program Target



Hello.c

Hello.exe



Pearson

Hello.exe

Summary

- Draw a mind-map to summarize the content of this lecture