Chapter 11: Programming Languages & Program Development
Objectives

- Explain what a programming language is.
- Contrast machine language and assembly language.
- Discuss the benefits and drawbacks of high-level programming languages.
- Explain how object-oriented languages attempt to remedy the shortcomings of earlier languages.
Objectives

- List several popular object-oriented languages and explain their advantage over older languages.
- List the six phases of the program development life cycle (PDLC) and explain why the PDLC is needed.
Objectives

- Explain why top-down program design makes programs easier to debug and maintain.
- List the three basic types of control structures and the advantages of each.
- Differentiate between syntax errors and logic errors in programs.
Programming Languages & Program Development

- **Programming** is the creation of software applications.
- **Programmers** are the people who create the software applications.
- A **programming language** is used by programmers to create software that the computer understands.
Programming Languages & Program Development

- **Syntax** is the vocabulary and rules of a programming language.
- **Code** is the programming instructions created by the programmers.
- **Source code**, the programming instructions in their original form, need to be translated into a form that the computer can understand.
Development of Programming Languages

Basic
10 REM Hello World in BASIC
20 PRINT "Hello World!"

Visual Basic .NET
'Hello World in Visual Basic .NET (VB.NET)

Imports System.Console

Class HelloWorld
    Public Shared Sub Main()
        WriteLine("Hello World!")
    End Sub
End Class
Development of Programming Languages

There are five distinct programming language generations, or levels:

- Machine language
- Assembly language
- Procedural languages
- Nonprocedural languages
- Natural languages
Development of Programming Languages

- **Machine language**
  - First-generation language
  - Based on binary numbers
  - The only programming language that a computer understands directly
  - Machine dependent
Development of Programming Languages

- **Assembly language**
  - Second-generation language
  - Machine dependent
  - Programs use:
    - **Mnemonics**, brief abbreviations for program instructions that make assembly language easier to use than machine language
    - Base-10 (decimal) numbers
  - Must be translated into machine language
Development of Programming Languages

- **Procedural languages**
  - Third-generation languages
  - Considered *high-level languages*, which do not require programmers to know details relating to the processing of data
  - Easier to read, write, and maintain than assembly and machine languages
  - Source code must be translated by a language translator
Development of Programming Languages

- Translation utility programs
  - **Compilers** translate source code into **object code**, the instructions in a specific computer’s machine language.
  - **Interpreters** translate source code and execute instructions without creating an object code.
Development of Programming Languages

- Compilers and Interpreters

![Diagram showing the process of compiling and interpreting Java and C++ programs.](image-url)
Development of Programming Languages

- Categories of Procedural Language
  - **Structured programming languages**
    - A set of standards to make programs more readable, reliable, and maintainable
    - A solution to *spaghetti code*, which uses too many GOTO statements that make code difficult to follow and prone to errors
Development of Programming Languages

- Categories of Procedural Language
  - **Modular programming languages**
    - Divide programs into self-contained modules
    - Use *information hiding*—the programmer does not need to know details of another module to write code in it
Development of Programming Languages

- **Nonprocedural languages**
  - Fourth-generation languages
  - Do not require step-by-step procedures to achieve the appropriate programming outcome
- **Examples**
  - Report generators (database reports)
  - Query languages
Development of Programming Languages

- **Natural language**
  - Fifth-generation languages
  - Still being perfected
  - Nonprocedural
  - Use everyday language to program
Development of Programming Languages

- **Object-oriented programming (OOP)**
  - Coding is attached to basic prebuilt items called **objects**, which include:
    - **Data**
    - **Attributes** that define the data
    - Procedures or operations called **methods**
    - An **interface** to exchange messages with other objects
Development of Programming Languages

- **Object-oriented programming (OOP)**
  - **Classes**: categories of objects
  - **Inheritance**: ability to pass on characteristics to subclasses
  - Makes information hiding (encapsulation) a reality
Development of Programming Languages

Program Development Methods

- **Rapid application development (RAD)**
  - Reuses prebuilt objects
  - Is possible because of OOP

- **Joint application development (JAD)**
  - Uses a team approach
  - **Agile** software development techniques follow a business approach to solution development.
Advantages of OOP

- Suitable for computer networks
- Uses **middleware** to make connections between applications on multiple networks
- Does not require changes in the code when data is modified
- Promotes thinking in a real-world environment, resulting in better software
A Guide to Programming Languages: One Size Doesn’t Fit All

- Programmers select the programming language that best meets users’ needs.
- Most popular languages today are:
  - Java
  - C
  - C++
  - Visual Basic
  - PHP
Early high-level languages

COBOL (Common Business-Oriented Language)
- Used for business applications
- Current focus is on editing aged code on mainframe computers

Fortran (formula translator)
- Used for scientific/math/engineering applications
- Being replaced by object-oriented and formula-solving programs
A Guide to Programming Languages: One Size Doesn’t Fit All

- Structured and modular languages
  - Required for large-scale program development
- Languages in widespread use are:
  - Ada
  - BASIC
  - Visual Basic (replaced by Visual Basic .NET)
  - C
A Guide to Programming Languages: One Size Doesn’t Fit All

- Object-oriented languages
  - Provide easier programming techniques
  - Examples
    - Smalltalk
    - C++
    - Java
    - Ruby
    - Visual Basic .NET
    - Visual Studio .NET
A Guide to Programming Languages: One Size Doesn’t Fit All

- Web-based languages
  - Not considered programming languages
  - Tell the browser how to display text and objects
- Language types
  - Markup
  - Scripting
A Guide to Programming
Languages: One Size Doesn’t Fit All

- **Markup languages**
  - Use tags to define how text and objects display. **Tags** are markers that usually come in pairs.
  - Content lies between the opening and closing tags.
A Guide to Programming Languages: One Size Doesn’t Fit All

- Markup languages
  - Examples:
    - HTML (Hypertext Markup Language)
    - XML (Extensible Markup Language)
    - XHTML (Extensible Hypertext Markup Language)
A Guide to Programming Languages: One Size Doesn’t Fit All

- **Scripting languages**
  - Create *scripts*, programs that control Web page actions or responses
  - Examples:
    - VBScript
    - JavaScript
    - Visual Studio .NET
    - AJAX
    - JSON
    - PHP
The Program Development Life Cycle

Program development life cycle (PDLC)

- An organized plan for developing software
- Consists of six phases, from problem definition through program implementation and maintenance
The Program Development Life Cycle

PHASE 1: Defining the problem

PHASE 2: Designing the program

PHASE 3: Coding the program

PHASE 4: Testing and debugging the program

PHASE 5: Documenting the program

PHASE 6: Implementing and maintaining the program
Phase 1: Defining the problem

- Define the problem to be resolved before developing a program.
- Define the program specifications, including decisions regarding data input, required processing, output, and the user interface.
The Program Development Life Cycle

- Phase 2: Designing the program
  - **Program design** identifies components of the program.
    - **Top-down program design** breaks program into small, manageable, highly focused subroutines
    - **Structured design** uses **control structures**, which are logical elements assembled in blocks of code that determine how subroutines will be programmed.
The Program Development Life Cycle

- Phase 2: Designing the program
  - Basic control structures categories
    - **Sequence**: Code performed in line-by-line order
    - **Selection (conditional or branch)**: Portion of code that leads to a block of code based on conditions being met
    - **Case**: A portion of code that branches to extensive coding
    - **Repetition (looping or iteration)**: A portion of code that repeats
The Program Development Life Cycle

- Control module
  - Print report heading
  - Process each separation
  - Print total commissions
    - Calculate commission
    - Print salesperson, commission
The Program Development Life Cycle

- Phase 2: Designing the program

  - Algorithm
    - A combination of control structures
    - A step-by-step accounting of how to arrive at a solution
The Program Development Life Cycle

Phase 2: Designing the program

Program design tools

- **Structured charts** show top-down design of programs.
- **Flowcharts** show logic of program.
- **Pseudocode** uses a stylized form of writing to describe logic.
Phase 3: Coding the program

- Programmers convert algorithms into programming code.

**Syntax errors**

- Mistakes in the construction of the programming commands
- Must be corrected for the program to run appropriately
Phase 4: Testing and debugging the program

- All errors, not just syntax errors, must be removed.

**Logic errors**
- Relate to problems in the solution’s design
- Cause incorrect output
- Program still runs despite logic errors
The Program Development Life Cycle

- Phase 5: Documenting the program

  **Documentation** includes:
  - Overview of program functionality
  - Tutorials
  - Thorough explanation of main features
  - Reference documentation of program commands
  - Description of error messages
  - Program design work
The Program Development Life Cycle

Phase 6: Implementing and maintaining the program

- Test the program.
  - Have users work with the software.
  - Correct errors.
- Maintain and evaluate the program on a regular basis.
Summary

- A programming language, with its own vocabulary and syntax, creates instructions that a computer understands.
- Machine language is based on binary code. Assembly language is similar but a little easier to understand. Both are difficult and demanding compared with later languages.
Summary

- Third-generation (high-level) programming languages do not require programmers to have a thorough understanding of processor details.

- Fourth-generation languages are mostly restricted to accessing databases.

- Fifth-generation language is natural language.
Summary

- Object-oriented languages work with prebuilt objects for fast, even simpler programming when compared with earlier-generation languages.
- The PDLC minimizes errors and maintenance problems.
Summary

- The six phases of the PDLC are: (1) defining the problem, (2) designing the program, (3) coding the program, (4) testing and debugging the program, (5) documenting the program, and (6) implementing and maintaining the program.
Summary

- Debugging and maintaining programs using top-down program design are simpler because program functions are divided into separate modules.

- Algorithms to perform any processing task can be created using sequence, selection, and repetition control structures.
Summary

- There are two types of bugs: syntax errors, caused by faulty command structure, and logic errors, caused by faulty programming design.