Programming Languages

• What is a Programming Language?
  – Syntax - words, symbols, and codes used to write computer programs
    • To write a program, appropriate software for the programming language being used is needed
  – Semantics – rules & logic
Programming Languages

- Low-Level Languages (earliest programming languages)
  - Machine language
    - Written at a very low level, just using 1s and 0s
    - First generation of programming languages
    - RISC vs CISC
  - Assembly language
    - Includes some names and other symbols to replace some of the 1s and 0s in machine language
    - Second generation of programming languages
    - Machine dependent
      » Written for one specific type of computer
Programming Languages

ASSEMBLY LANGUAGE
Assembly language instructions use mnemonic operating codes to make the instructions much easier to understand. Note that data must still be moved in and out through the registers (register 0 in this example).

MACHINE LANGUAGE
Machine language instructions are typically in binary form, and the memory address locations, as well as the instructions themselves, need to be specified. The highlighted machine language instructions shown to the left correspond to the highlighted assembly language statements below.

FIGURE 13-17
Assembly and machine language.
Programming Languages

- High-Level Languages
  - Closer to natural languages
  - Machine independent => Portable
  - Includes 3GLs (FORTRAN, BASIC, COBOL, C, etc.) and object-oriented languages (Visual Basic, C#, Python, Java, etc.)
  - Visual or graphical languages
    - Use graphical interface to create programs
    - Designed for educational purposes
— Fourth-Generation Languages (4GLs)
  • Even closer to natural languages and easier to work with than high-level
  • Declarative rather than procedural (Logical)
  • Includes structured query language (SQL) used with databases
• Common Programming Languages
Programming Paradigms

• Procedural/Structured (C)
• Object Oriented (Java)
• Logical (Prolog – also declarative)
• Functional (Lisp)
Approaches to Design and Development

• Procedural Programming
  – An approach to program design in which a program is separated into small modules that are called by the main program or another module when needed
  • Procedure call—locating specific tasks in procedures (modules or subprograms) that are called by the main program when needed
  • Allows each procedure to be performed as many times as needed; multiple copies of code not needed
  • Prior to procedural programming, programs were one large set of instructions (used GOTO statements)
Approaches to Design and Development

– Structured Programming
  – Top-down design breaks the program into small modules
  – Bottom up creates a library of utilities

– Variables
  • Named memory locations that are defined for a program
  • Used to store the current value of data items used in the program
Control Structures

– Control Structures
  • A pattern for controlling the flow of logic in a computer program, module, or method
  • The Sequence Control Structure
    – Series of statements that follow one another

FIGURE 13-7
Control Structures

• The Selection Control Structure
  – Multiple paths, direction depends on result of a certain condition
    » If-then-else
  – Case control structure
    » Allows for as many possible results of the specified condition as needed

• Repetition Control Structure (iteration control structure)
  – Repeat series of steps
    » Do-while
    » Do-until
Control Structures

FIGURE 13-7
The three fundamental control structures. Note that each structure has only one entry point and only one exit point.
Approaches to Design and Development

Modules are arranged hierarchically in a top-down fashion, as illustrated here for a payroll application.

The control program calls each module as needed, such as when it is time to compute the deductions.

Each module then calls further modules, as needed, such as to compute federal taxes.

FIGURE 13-1
Structured programming. A structured program is divided into individual modules; each module represents a very specific processing task.
Approaches to Design and Development

• Object-Oriented Programming (OOP)
  – Programs consist of a collection of objects that contain data and methods to be used with that data
    • Class
      – Group of objects that share some common properties
    • Instance
      – An individual object in a class
    • Attributes
      – Data about the state of an object
Approaches to Design and Development

- All execution performed by passing messages
  - Object.method()
- Methods
  - Perform actions on an object
- Objects can perform nontraditional actions and be easily used by more than one program
The Program Development Life Cycle (PDLC)

- Unified Modeling Language (UML) Models
  - Set of standard notations for creating business models
  - Widely used in object-oriented programs
  - Includes class diagrams and case diagrams
# UML Diagram for Rectangle class

<table>
<thead>
<tr>
<th>Rectangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>- length</td>
</tr>
<tr>
<td>- width</td>
</tr>
<tr>
<td>+setLength()</td>
</tr>
<tr>
<td>+setWidth()</td>
</tr>
<tr>
<td>+getLength()</td>
</tr>
<tr>
<td>+getWidth()</td>
</tr>
<tr>
<td>+getArea()</td>
</tr>
</tbody>
</table>
The Program Development Life Cycle (PDLC)

- Program Development (application software development)
  - The process of creating application programs
- Program Development Life Cycle (PDLC)
  - The five phases of program development
The Program Development Life Cycle (PDLC)

- Problem Analysis
  - The problem is considered and the program specifications are developed
    - Specifications developed during the PDLC are reviewed by the systems analyst and the programmer (the person who will code the program)
    - Goal is to understand the functions the software must perform
  - Documentation: Program Specifications
    - Result of the first phase of the PDLC outlining what the program must do
The Program Development Life Cycle (PDLC)

• Program Design
  – The program specifications are expanded into a complete design of the new program
    • Algorithm for the program is developed
    • Careful planning and design of a computer program are extremely important
  – Program Design Tools
    • Planning tools that include diagrams, charts, tables, and models
    • Structure Charts (hierarchy charts)
      – Depict the overall organization of a program
The Program Development Life Cycle (PDLC) - Design

• Flowcharts
  – Show graphically, step-by-step, how a computer program will process data
  – Use special symbols and relational operators
  – Can be drawn by hand or with flowcharting software

• Pseudocode
  – Uses English-like statements to outline the logic of a program rather than the flowchart’s graphical symbols
The Program Development Life Cycle (PDLC)

**FLOWCHART OPERATORS**

- `<`  Less than
- `<=`  Less than or equal to
- `> `  Greater than
- `>=`  Greater than or equal to
- `==`  Equal to
- `<>`  Not equal to

**FLOWCHART SYMBOLS**

- Start/stop program
- Decision
- Processing
- Connector
- Input/output
- Flowline

**FLOWCHART SOFTWARE**

Can be used to create and modify flowcharts.

Figure 13-4
A flowchart example.

Understanding Computers: Today and Tomorrow, 14th Edition
Start
counter = 0
Read a record
DO WHILE there are records to process
    IF computer_experience
        IF company_service ≥ 5 years
            Print employee_name
            Increment counter
        ELSE
            Next statement
        END IF
    ELSE
        Next statement
    END IF
ELSE
    Next statement
END IF
Read another record
END DO
Print counter
Stop
The Program Development Life Cycle (PDLC)

- Good Program Design
  - Is essential
  - Saves time
  - Good Program Design Principles
    - Be Specific
      - All things the program must do or consider must be specified
    - Follow the One-Entry-Point/One-Exit-Point Rule
    - No Infinite Loops or Logic Errors
      - Infinite loop is a series of steps that repeat forever
The Program Development Life Cycle (PDLC)

- Program Design Testing
  - Design should be tested to ensure logic is correct
    - Desk check
    - Tracing tables
- Documentation: Design Specifications
  - Illustrates the program needed to fulfill the program requirements
  - Expressed using structure charts, flowcharts, pseudocode, and UML models
  - Include any test data and results from desk checking
The Program Development Life Cycle (PDLC)

• Program Coding
  – The program code is written using a programming language
  – Choosing a Programming Language
    • Suitability to the application
    • Integration with other programs
    • Standards for the company
    • Programmer availability
    • Portability if being run on multiple platforms
    • Development speed
The Program Development Life Cycle (PDLC)

– The Coding Process
  • Coding Standards
    – Rules designed to standardize programming
    – Makes programs more readable and easier to maintain
    – Includes the proper use of comments to:
      » Identify the programmer and last modification date
      » Explain variables used in the program
      » Identify the main parts of the program
The Program Development Life Cycle (PDLC)

**COMMENTS**
Comments are usually preceded by a specific symbol (such as *, C', #, or //); the symbol used depends on the programming language being used. Anything else in a comment line is ignored by the computer.

Comments at the top of a program should identify the name and author of the program, date written and last modified, purpose of the program, and variables used in the program.

Comments in the main part of a program should indicate what each section of the program is doing. Blank comment lines can also be used to space out the lines of code, as needed for readability.

```
**********************************************************************************************************************
* This program inputs two numbers, computes their sum, and displays the sum.
* Written by: Deborah Morley 3/12/12
**********************************************************************************************************************
* Variable list
* SUM: Running sum
* CNTR: Counter
* NUM: Number inputted
*
   REAL SUM, CNTR, NUM
**********************************************************************************************************************
* INITIALIZE VARIABLES
   SUM = 0
   CNTR= 0
*
* INPUT NUMBER, ADD IT TO THE SUM, INCREMENT COUNTER, AND THEN
* REPEAT UNTIL TWO NUMBERS HAVE BEEN ENTERED
   DO 10 CNTR = 1, 2
```
The Program Development Life Cycle (PDLC)

– Reusable code
  • Pretested, error-free code segments that can be used over and over again with minor modifications
  • Can greatly reduce development time

– Documentation: Documented Source Code
  • Program coding phase results in the program written in the desired programming language
  • Should include enough comments (internal documentation) so that the source code is easy to understand and update
The Program Development Life Cycle (PDLC)

• Program Debugging and Testing
  – The process of ensuring a program is free of errors (bugs) and works as it is supposed to
  – Translating Coded Programs into Executable Code
    • Coded programs need to be translated from source code written by the programmer to object code the computer can execute
    • Converted using a language translator
      – Program that converts source code to machine language
The Program Development Life Cycle (PDLC)

– Compilers
  » Language translator that converts an entire program into machine language before executing it
  » Designed for specific programming languages such as Java or Python

– Interpreters
  » Translates one line of code at one time

– Assemblers
  » Convert assembly language programs into machine language
The Program Development Life Cycle (PDLC)

**FIGURE 13-11**
Compiler and linkage editor. A compiler and a linkage editor convert source code into executable code.
The Program Development Life Cycle (PDLC)

– Preliminary Debugging
  • Compiler and Syntax Errors
    – As programs are compiled or interpreted, errors occur which prevent the program from running properly
    – Syntax errors occur when the programmer has not followed the rules of the programming language
  • Run Time and Logic Errors
    – Run time errors occur when the program is running
The Program Development Life Cycle (PDLC)

– Logic errors are errors in the logic of the program
  » Program will run but produces incorrect results
  » Dummy print statements can help locate logic errors and other run time errors
The Program Development Life Cycle (PDLC)

– Testing

• Occurs after the program appears to be correct to find any additional errors

• Uses good test data—data that is very similar to the actual data that will be used in the program

• Tests conditions that will occur when the program is implemented

• Checks for coding omissions (i.e., product quantity allowed to be < 0)
The Program Development Life Cycle (PDLC)

- Two stages
  - Alpha test—internal on-site test
  - Beta test—outside test
- Documentation: Completed Program Package
  - Copy of the test data, test results, finished program code, and other documentation generated during the testing phase should be added to the program package
    - Developer documentation
    - User documentation
The Program Development Life Cycle (PDLC)

• Program Implementation and Maintenance
  – Once the system containing the program is up and running, the implementation process is complete
  – Program maintenance
    • Process of updating software so it continues to be useful
    • Very costly
  – Documentation: Amended program package
    • Program package should be updated to reflect new problems or issues that occur and what changes to the program were necessary
Approaches to Design and Development

• Agile software development
  – Goal is to create software rapidly
    » many short steps or iterations (spirals)
  – Focuses on building small functional program pieces during the project
  – Includes earlier adaptive software approaches such as RAD (rapid application development) and extreme programming (XP)

• Extreme Programming – integrates a test first methodology
Recurrent Design

![Diagram of Recurrent Design Process]

- Determine objectives, alternatives, constraints
- Progress through steps
- Evaluate alternatives, identify, resolve risks
- Cumulative Cost

Steps:
- Commitment review
- Requirements plan
- Concept of operation
- Development plan
- Requirements validation
- Integration and test plan
- Design validation and verification
- Implementation
- Acceptance test
- Integration and test
- Unit test
- Code
- Detailed design
- Software requirements
- Software product design
- Operational prototype
- Prototype 3
- Prototype 2
- Prototype 1
- Simulations, models, benchmarks

Next phases:
- Develop, verify next level product
- Plan next phases

Understanding Computers: Today and Tomorrow, 14th Edition
Tools for Facilitating Program Development

• Application Lifecycle Management (ALM) Tools
  – Creating and managing an application during its entire lifecycle, from design through retirement
  – Tools include:
    • Requirements management
      – Keeping track of and managing the program requirements as they are defined and then modified
    • Configuration management
      – Keeping track of the progress of a program development project
Tools for Facilitating Program Development

• Application Generators
  – Software program that helps programmers develop software

• Macros
  • Record and play back a series of keystrokes
  • Programmers write them in a macro programming language such as Visual Basic for Applications

• Report and Form Generators
  • Tools that enable individuals to prepare reports and forms quickly
Tools for Facilitating Program Development

• Device Software Development Tools
  – Assist with developing embedded software to be used on devices, such as cars, ATM machines, and consumer devices

• Software Development Kits (SDKs) and Application Program Interfaces (APIs)
  – Designed for a particular platform
  – Enables programmers to develop applications more quickly and easily
    • Often released by hardware or software companies
      – iOS SDK—allows third party developers to create new applications for iPhone, iPad, iPod Touch
Tools for Facilitating Program Development

• Application Program Interfaces (APIs)
  – Help applications interface with a particular operating system
• Often used in conjunction with Web sites
• Rich Internet Application (RIA) Tools
  – Web-based applications that work like installed software programs
  – Desktop RIA can access local files and used without an Internet connection
  – Web-based RIAs are common
  – Tools to develop RIAs
    • Adobe AIR
Summary

• Approaches to Program Design and Development
• The Program Development Life Cycle (PDLC)
• Tools for Facilitating Program Development
• Programming Languages