Technology in Action

Alan Evans • Kendall Martin
Mary Anne Poatsy

Eleventh Edition
Topics

• Understanding Software Programming
  – The Importance of Programming
  – The Life Cycle of an Information System
  – The Life Cycle of a Program

• Programming Languages
  – Many Languages for Many Projects
  – Exploring Programming Languages
Understanding Software Programming

• Importance of Programming
• Life Cycle of an Information System
• Life Cycle of a Program
• Some tasks are complex
  – Requires creative thought
  – Requires human touch

• Some tasks are candidates for automation
  – Repetitive
  – Works with electronic information
  – Follows a series of clear steps
The Importance of Programming

• A career in programming offers
  – Plentiful jobs
  – Strong salaries
  – Telecommuting is often easy to arrange
• Computer programs exist for many tasks
• Programming necessary when no existing software for task
The Importance of Programming

• Basic knowledge of programming
  – Add features that support personal needs
  – Create miniprograms (macros)
  – Add custom commands
  – Create custom applications
  – Successfully complete projects
Life Cycle of an Information System

• System
  – Collection of pieces working to achieve common goal

• An information system includes:
  – Data
  – People
  – Procedures
  – Hardware
  – Software
Life Cycle of an Information System

SDLC

1. Problem/Opportunity Identification
2. Analysis
3. Design
4. Development
5. Testing & Installation
6. Maintenance & Evaluation
• Corporations form a development committee
  – Evaluates systems-development proposals
  – Decide which projects to take forward based on resources
Life Cycle of an Information System

Analysis

• Analysts explore problem/need in depth
• Develop program specifications
• Feasibility assessment is performed
• User requirements are defined
• Plan of action is recommended
Life Cycle of an Information System

Design

- Generates detailed plan for programmers
- Flowcharts and data-flow diagrams
  - Flowcharts are visual diagrams of a process
  - Data-flow diagrams trace data from point it enters system to final place
Life Cycle of an Information System

Design

Data-flow diagram

Concert Attendee

Ticket Request

Customer Ticket Information

Ticket Reservation System

Ticketing Information

Customer Preferences

Concert Hall Ticket Agent

Concert Hall

Customer Ticket Information

Copyright © 2015 Pearson Education, Inc.
Life Cycle of an Information System

Development

- Actual programming takes place
- First phase of program development life cycle (PDLC)
Life Cycle of an Information System
Testing and Installation

- Program tested to ensure it works properly
- Program is installed for official use
Life Cycle of an Information System

Maintenance and Evaluation

- Program performance monitored
- Corrections and modifications made
- Additional enhancements evaluated
- Appropriate modifications made
- **Scope creep** is set of requests for additional features

Copyright © 2015 Pearson Education, Inc.
• Programming
  – Translating task into commands used to perform task
  – Identifying parts of performable tasks
  – Describing tasks in specific, complete manner
  – Translating description into language understood by CPU
Program Development Life Cycle

STEP 1: Describing the Problem (The Problem Statement)

STEP 2: Making a Plan (Algorithm Development)

STEP 3: Coding (Speaking the Language of the Computer)

STEP 4: Debugging (Getting Rid of Errors)

STEP 5: Testing and Documentation (Finishing the Project)
Life Cycle of a Program

Problem Statement

• Necessary to solve problems
• Interact with users
• Programmers handle bad inputs through error handling
• Problem statement includes testing plan
  – Does not cover every possible input but patterns of inputs
Life Cycle of a Program

Algorithm Development

• Developing detailed algorithm
  – Set of specific sequential steps
  – Describe in natural language what program must do

• Algorithms are limited
Life Cycle of a Program

Algorithm Development

• Algorithms represented through flowcharts
  – Provides a visual representation of patterns
Life Cycle of a Program

Algorithm Development

• Algorithms also represented through Pseudocode
  – Text-based approach
  – Words describe actions
  – Organized like an outline
  – Combination of common and special words
Life Cycle of a Program

Decision Making and Design

• Programmers handle complex algorithms
  – List of choices
  – Decision points
    • Binary decisions
  • Loops

• Programmers create algorithms using:
  – Top-down design
  – Object-oriented analysis
Developing the Algorithm
Decision Making

Flowchart Symbols

- Diamond
  - **BINARY DECISION**
  - A yes/no question will be asked here

- Rectangle
  - **PROCESS**
  - Series of steps of work will be done here

- Parallelogram
  - **INPUT/OUTPUT**
  - Data will be read (input) or printed (output)

- Oval
  - **TERMINATE**
  - Seen at the beginning or end of the program

- Directed Line
  - **DIRECTION OF FLOW**
  - Shows the path to follow to the next executable instruction

Decision Points

1. **Ask for # of Hours Worked**
2. **Read # of Hours Worked**
   - **Decision point**
   - **Is Number of Hours Worked <= 8?**
     - **Yes**
       - Program executes one set of steps if answer is Yes
       - **Total Pay = $7.50 * Number of Hours Worked**
       - **Done**
     - **No**
       - Program executes different set of steps if answer is No
       - **Total Pay = $7.50 * 8 + $11.25 * (Number of Hours Worked - 8)**
       - **Done**
Developing the Algorithm
Decision Making

Start on Monday
Start with Total Pay for Week = 0.00

Are we still in the same week?

Calculate Pay for the Day

Calculate Total Pay for Week So Far

Bump to Next Day

If test condition passes, loop continues

Initial value

Test condition

No

If test condition fails, break out of loop and move to next step

Update

Print Weekly Paycheck
Top-Down Design

- Problem is broken into series of high-level tasks
- Detailed subtasks created from high-level tasks
- Continue until steps are close to programming language commands
Top-Down Design

- How top-down design is used in programming

```
if (NumberHoursWorkedToday <= 8)
    Pay = $7.50 * NumberHoursWorkedToday
else
    Pay = $7.50 * 8 + $11.25 * (NumberHoursWorkedToday - 8)
```
Object-Oriented Analysis

• Programmers identify categories of inputs
  – Classes (categories of inputs) identified
  – Classes defined by information (data) and actions (methods)
  – Algorithm enables objects to interact

• Object-oriented often chosen over top-down design
  – Takes advantage of reusability
Object-Oriented Analysis

**Data (Information)**
- Employee Class:
  - NAME
  - ADDRESS
  - SOCIAL SECURITY #
  - PAY GRADE
  - PAY RATE
  - GoToWork()
  - LeaveWork()
  - CollectPay()

**Methods (Actions)**
- John Doe
  - 1313 Mockingbird
  - 011-11-0000
  - 5
  - $43.02
  - Employee1

- Jane Doe
  - 1060 West Addison
  - 999-09-0909
  - 10
  - $57.33
  - Employee2

- Bill McGillicutty
  - 7 Freedom Square
  - 123-45-6789
  - 4
  - $39.80
  - Employee3
Coding

• Coding: Translating an algorithm into CPU instructions
• Highly precise format
  – few keywords
  – consistent structure
Coding

• Programmers move from algorithm to code by:
  
  – Identifying key pieces of information
  – Identifying flow of each step
  – Converting algorithm into specific programming language
Coding

• Programming language
  – A “code” for instructions CPU knows how to perform
  – Languages use special words and strict rules
  – Allows control of CPU without knowing hardware details
### Sample Code for Different Language Generations

<table>
<thead>
<tr>
<th>GENERATION</th>
<th>EXAMPLE</th>
<th>SAMPLE CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1GL</td>
<td>Machine</td>
<td><strong>Bits</strong> describe the commands to the CPU.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1110 0101 1001 1111 0000 1011 1110 0110</td>
</tr>
<tr>
<td>2GL</td>
<td>Assembly</td>
<td><strong>Words</strong> describe the commands to the CPU.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADD Register 3, Register 4, Register 5</td>
</tr>
<tr>
<td>3GL</td>
<td>FORTRAN, BASIC, C, Java</td>
<td><strong>Symbols</strong> describe the commands to the CPU.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total Pay = Pay + Overtime Pay</td>
</tr>
<tr>
<td>4GL</td>
<td>SQL</td>
<td><strong>More powerful commands</strong> allow complex work to be done in a single sentence.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SELECT isbn, title, price, price*0.06 AS sales_tax FROM books WHERE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>price&gt;100.00 ORDER BY title;</td>
</tr>
<tr>
<td>5GL</td>
<td>PROLOG</td>
<td>Programmers can build applications <strong>without specifying an algorithm</strong>. Find all the people who are Mike’s cousins as. ?-cousin (Mike, family)</td>
</tr>
</tbody>
</table>
Coding

• Program Structure
  – Each variables is announced early in program
  – Memory space is set aside for inputs and outputs
  – Programmers can leave notes inside a program (comments)
```cpp
#include <iostream>
using namespace std;

void main()
{
    // Begin by asking for some variables to be stored in RAM
    float NumberHoursWorkedToday = 0.0;
    float Pay = 0.0, TotalPay = 0.0;
    int Day;

    // Set up a loop to ask the user how many hours they worked each day
    for (Day = 1; Day <= 7; Day++)
    {
        // Read the input data from the screen
        cout << "Enter the number of hours you worked on day " << Day << " : ";
        cin >> NumberHoursWorkedToday;

        // Check the input makes sense
        if (NumberHoursWorkedToday < 0)
        {
            // Wait a minute! We need to handle possible errors here.
            // Print a warning message to the user if they enter a negative value.
            cout << "You can't work negative hours! Try again: ";
            cin >> NumberHoursWorkedToday;
        }

        if (NumberHoursWorkedToday <= 8)
        {
            // Compute pay earned today at normal base rate
            Pay = NumberHoursWorkedToday * 7.50;
        }
        else
        {
            // Compute the pay earned using the overtime rule
            Pay = (8 * 7.50) + (11.25 * (NumberHoursWorkedToday - 8));
        }

        // Update the total pay you have earned so far this week
        TotalPay += Pay;
    }

    // When we hit this brace, we bounce back up to the beginning of the loop
    // Now we're free from the loop! Let's print what we've earned so far
    cout << "Your total pay for this week comes to: " << TotalPay;
}
// The program is done.
```
• Modularity in code
  – "Containers": sections that can be used repeatedly
  – Referred to as functions, methods, procedures, subroutines, modules, or packages
Compilation

- Code must be converted to 1s and 0s
- Compiler creates an executable program
- Some languages use an interpreter
Coding Tools

Integrated Development Environments

- IDE makes coding process easier
- Provides editor
- debugger
Debugging

- Finding and correcting errors
Debugging

• Testing plan helps programmers know program has solved the problem
• Compilation errors
  – Improper syntax
• Logical errors
  – Program runs but executes incorrectly
Testing and Documentation

- Internal testing
  - Group in company uses program every way possible
- External testing
  - Eventual users work with program
  - Beta version – release to public
Testing and Documentation

• Solving problems after beta testing
  – Manufacturer makes changes before final release
  – Release to manufacturers (RTM)
  – After RTM, product is in general availability (GA)
  – Problems addressed in updates or service packs
Testing and Documentation

• Finishing the project
  – Technical writers produce internal documentation
  – User documentation is produced
  – Software trainers teach others how to use program
Programming Languages

- Many Languages for Many Projects
- Exploring Programming Languages
Many Languages for Many Projects

- Programming languages have been developed to balance conflicting goals
Visual Basic
Visual Basic

• Advantages of Visual Basic
  – Prototyping is form of rapid application development (RAD)
  – Developers create prototype then generate system documents
  – RAD is alternative to waterfall approach
  – Used to build Windows applications

• MS .NET Framework
C and C++

• Provides higher-level programming features
• Allows direct manipulation of system memory and CPU registers
C and C++

• Code runs fast and uses small amount of memory
• Basic components are common to many languages
Java

• Java is object-oriented language
• Java applications work on many types of computers
Java

Compile once and run on many platforms

```java
for (i = 1; i < 20; i++)
{
    System.out.println(i);
}
```
Objective C

• Most popular language for writing Mac OS X applications
HTML

- HTML - markup language for web page design
JavaScript and VBScript

Scripting languages

– Allows decisions and calculations

– Adds interactivity to web pages
• Interactive web pages
Flash, AJAX, and XML

• Creating web page that includes sophisticated animation
  – Adobe Flash: Web-based multimedia
  – Microsoft Silverlight: Supports rich multimedia and interactive applications
  – Can update information without requiring page refresh
Flash, AJAX, and XML

- XML helps websites gather information from other sites
  - eXtensible Markup Language (XML) enables designers to define data-based tags
  - Easier for website to transfer key information
  - Formatting controls is important
  - Groups agree on standard system of tags
Mobile Applications
Mobile Applications

• Tools for building apps for Android devices
  – Android software development kit (SDK) is required
  – Uses well-known IDEs with special plug-ins

• Build iOS apps (iPhone, iPad)
  • Objective C language
Mobile Applications

Corona and Magmito support several different devices and save time for simple applications.

For specific features and ultimate performance, custom programming is still required.
The Next Great Language

Learn “how to learn” new languages

• always changing
Check Your Understanding

1. Why do I need to understand how to create software?

• Programming skills allow you to customize existing software products to accomplish required tasks.

• A beginning-level knowledge of programming will let you create macros, customized mini-programs that speed up redundant tasks.
Check Your Understanding

2. What is a system development life cycle, and what are the phases in the cycle?

• System Development Life Cycle (SDLC) has six phases:
  1. A **problem** or opportunity is identified.
  2. The problem is **analyzed**,
  3. A detailed plan is **designed** using flowcharts and data-flow diagrams.
  4. Programmers **develop** the program, and then document.
  5. The program is **tested**.
  6. Ongoing **maintenance** and evaluation ensure a working product.
Check Your Understanding

3. What is the life cycle of a program?

- The **problem statement** identifies the task to be computerized.
- An **algorithm** specifies the sequence of steps that the program must take to complete the work.
- The algorithm is then translated into programming code.
- The code goes through **debugging**, finding and repairing errors.
- **Testing** is performed.
- The results of the entire project are **documented**.
Check Your Understanding

4. What role does a problem statement play in programming?

- The problem statement is a description of what tasks the computer program must accomplish and how the program will execute these tasks.
- It describes the input data that users will have at the start of the job, the output that the program will produce, and the exact processing that converts these inputs to outputs.
- The problem statement identifies potential errors and plans to address these errors.
Check Your Understanding

5. How do programmers create algorithms and move from algorithm to code?

- Programmers create an algorithm by converting a problem statement into a list of steps.
- Yes/no **binary decisions** are common
- Algorithms are documented in the form of a **flowchart** or in **pseudocode**.
- Programmers use either **top-down** or **object-oriented** analysis
- Programming languages are classified in several major groupings, sometimes referred to as **generations**
- **Compilation** is the process by which code is converted into machine language
- Each programming language has its own compiler.
Check Your Understanding

6. What steps are involved in completing the program?

- Debugging detects all the code errors
- Before its commercial release, software is often provided at a reduced cost or at no cost in a beta version
- Technical writers create internal documentation for the program and external documentation that will be provided to users of the program.
- User training, which begins once the software is distributed, teaches the user community how to use the software efficiently.
Check Your Understanding

7. How do programmers select the right programming language for a task?

- Certain languages are best used for certain problems.
- The target language should be well matched to the amount of space available for the final program.
- Some projects require the selection of a language that can produce code that executes the fastest.
- Selecting a language with which the programmers are familiar is also helpful.
8. What are the most popular programming languages?

- **Visual Basic, C/C++, and Java** are popular languages
- **Objective C** is a language used in programming applications for mobile devices using iOS
- Programmers use HTML tags to structure web pages. **HTML5** includes more advanced tags like `<video>`.
- Scripting programs such as **JavaScript, PHP, and VBScript** are popular
- **AJAX** is a programming solution that uses a combination of technologies to create websites that can update without the user refreshing the page.
- **XML and JSON** allow web pages to exchange information, not just formatting details.