Chapter 1 - An Introduction to Computers and Problem Solving

- 1.1 An Introduction to Computers
- 1.2 Windows, Folders, and Files
- 1.3 Program Development Cycle
- 1.4 Programming Tools
1.1 An Introduction to Computers

• Miscellaneous Questions
Communicating with the Computer

- Machine language – low level, hard for humans to understand
- Visual Basic – high level, understood by humans, consists of instructions such as Click, If, and Do
A compiler translates a high-level language into machine language.

The Visual Basic compiler points out certain types of errors during the translation process.
Programming and Complicated Tasks

- Tasks are broken down into instructions that can be expressed by a programming language
- A *program* is a sequence of instructions
- Programs can be only a few instructions or millions of lines of instructions
All Programs Have in Common:

- Take data and manipulate it to produce a result
- Input – Process – Output
  - Input – from files, the keyboard, or other input device
  - Output – usually to the monitor, a printer, or a file
Hardware and Software

• Hardware – the physical components of the computer
  • Central processing unit
  • Disk drive
  • Monitor

• Software – The instructions that tell the computer what to do
Programmer and User

- Programmer – the person who solves the problem and writes the instructions for the computer
- User – any person who uses the program written by the programmer
Problem Solving

- Developing the solution to a problem
- Algorithm – a step by step series of instructions to solve a problem
Visual Basic 2010

• BASIC originally developed at Dartmouth in the early 1960s
• Visual Basic created by Microsoft in 1991
• Visual Basic 2010 is similar to original Visual Basic, but more powerful
XP vs Vista vs Windows 7
1.2 Windows, Folders, and Files

- Windows and Its Little Windows
- Mouse Actions
- Files and Folders
Windows and Its Little Windows

- Difference between Windows and windows.
- Title bar indicates if window is active.
Mouse Actions:

- Hover
- Drag and drop
- Click
- Right-click
- Double-Click
Files and Folders

File: holds programs or data. Its name usually consists of letters, digits, and spaces.

Folder: contains files and other folders (called subfolders).
### Key Terms in using Folders and Files

<table>
<thead>
<tr>
<th>Term</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk</td>
<td>Hard disk, flash drive, DVD</td>
</tr>
<tr>
<td>File name</td>
<td>Payroll</td>
</tr>
<tr>
<td>Extension</td>
<td>.txt</td>
</tr>
<tr>
<td>Filename</td>
<td>Payroll.txt</td>
</tr>
<tr>
<td>Path</td>
<td>TextFiles\Payroll.txt</td>
</tr>
<tr>
<td>Filespec</td>
<td>C:\TextFiles\Payroll.txt</td>
</tr>
</tbody>
</table>
Windows Explorer

- Used to view, organize, and manage folders and files.
- Manage: copy, move, delete
Invoking Windows Explorer

- Right-click on Windows Start button
- Click on Explore (or Open Windows Explorer) in context menu
Display File Extensions (Vista & Windows 7)

• Click on Windows Start button.
• Type Folder Options into Search box.
• Press Enter key.
• Click on View tab in dialog box.
• Uncheck “Hide extensions for known file types”.
• Click on OK.
Display File Extensions (Vista & Windows 7 cont.)

Folder Options

General View Search

Folder views
You can apply the view (such as Details or Icons) that
you are using for this folder to all folders of this type.

Apply to Folders Reset Folders

Advanced settings:

Files and Folders
- Always show icons, never thumbnails
- Always show menus
- Display file icon on thumbnails
- Display file size information in folder tips
- Display the full path in the title bar (Classic theme only)
- Hidden files and folders
  - Don't show hidden files, folders, or drives
  - Show hidden files, folders, and drives
  - Hide empty drives in the Computer folder
- Hide extensions for known file types
- Hide protected operating system files (Recommended)

Restore Defaults OK Cancel Apply
Display File Extensions (XP)

- Alt/Tools/Folder Options
- Click the View tab.
- Uncheck "Hide extensions for known file types".
- Click on OK.
1.3 Program Development Cycle

- Performing a Task on the Computer
- Program Planning
**Terminology**

A computer program may also be called:

- Project
- Application
- Solution
Program Development Cycle

- Software refers to a collection of instructions for the computer.
- The computer only knows how to do what the programmer tells it to do.
- Therefore, the programmer has to know how to solve problems.
Performing a Task on the Computer

- Determine *Output*
- Identify *Input*
- Determine *process* necessary to turn given *Input* into desired *Output*
Problem-Solving: Approach Like Algebra Problem

• How fast is a car traveling if it goes 50 miles in 2 hours?
• **Output**: a number giving the speed in miles per hour
• **Input**: the distance and time the car has traveled
• **Process**: speed = distance / time
Pictorial representation of the Problem Solving Process
Program Planning

• A recipe is a good example of a plan
• Ingredients and amounts are determined by what you want to bake
• Ingredients are input
• The way you combine them is the processing
• What is baked is the output
Program Planning (continued)

• Always have a plan before trying to write a program
• The more complicated the problem, the more complex the plan must be
• Planning and testing before coding saves time
Program Development Cycle

1. Analyze: Define the problem.
2. Design: Plan the solution to the problem.
3. Choose the interface: Select the objects (text boxes, buttons, etc.).
4. Code: Translate the algorithm into a programming language.
5. Test and debug: Locate and remove any errors in the program.
6. Complete the documentation: Organize all the materials that describe the program.
1.4 Programming Tools

- Flowcharts
- Pseudocode
- Hierarchy Chart
- Direction of Numbered NYC Streets Algorithm
- Class Average Algorithm
Three tools are used to convert *algorithms* into computer programs:

- **Flowchart** - Graphically depicts the logical steps to carry out a task and shows how the steps relate to each other.
- **Pseudocode** - Uses English-like phrases with some Visual Basic terms to outline the program.
- **Hierarchy chart** - Shows how the different parts of a program relate to each other.
Algorithm

A step-by-step series of instructions for solving a problem (a recipe is an example of an algorithm).
Problem Solving Example

• How many stamps should you use when mailing a letter?
• One rule of thumb is to use one stamp for every five sheets of paper or fraction thereof.
Algorithm

1. Request the number of sheets of paper; call it Sheets. \((\text{input})\)
2. Divide Sheets by 5. \((\text{processing})\)
3. Round the quotient up to the next highest whole number; call it Stamps. \((\text{processing})\)
4. Reply with the number Stamps. \((\text{output})\)
Flowchart

Graphically depict the logical steps to carry out a task and show how the steps relate to each other.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Flowline</td>
<td>Used to connect symbols and indicate the flow of logic.</td>
</tr>
<tr>
<td></td>
<td>Terminal</td>
<td>Used to represent the beginning (Start) or the end (End) of a task.</td>
</tr>
<tr>
<td></td>
<td>Input/Output</td>
<td>Used for input and output operations, such as reading and displaying. The data to be read or displayed are described inside.</td>
</tr>
<tr>
<td></td>
<td>Processing</td>
<td>Used for arithmetic and data-manipulation operations. The instructions are listed inside the symbol.</td>
</tr>
</tbody>
</table>
Flowchart Symbols (continued)

- **Decision**: Used for any logic or comparison operations. Unlike the input/output and processing symbols, which have one entry and one exit flowline, the decision symbol has one entry and two exit paths. The path chosen depends on whether the answer to a question is “yes” or “no.”

- **Connector**: Used to join different flowlines.

- **Annotation**: Used to provide additional information about another flowchart symbol.
Flowchart Example

Start

Read sheets

Set stamps = sheets / 5

Round stamps up to next whole number

Display stamps

End
Pseudocode

Uses English-like phrases with some Visual Basic terms to outline the task.
Determine the proper number of stamps for a letter

Read Sheets (input)

Set the number of stamps to Sheets / 5 (processing)

Round the number of stamps up to the next whole number (processing)

Display the number of stamps (output)
Hierarchy Chart

- Shows how the different parts of a program relate to each other

Hierarchy charts are also called
- structure charts
- HIPO (Hierarchy plus Input-Process-Output) charts
- top-down charts
- VTOC (Visual Table of Contents) charts
Hierarchy Charts Example

Postage stamp program

Read sheets

Calculate stamps

Set stamps = sheets / 5

Display stamps

Round stamps up to next whole number
Divide-and-Conquer Method

- Used in problem solving – take a large problem and break it into smaller problems
- Solve the small problems first
Statement Structures

- **Sequence** – execute instructions from one line to the next without skipping over any lines
- **Decision** - if the answer to a question is “Yes” then one group of instructions is executed. If the answer is “No,” then another is executed.
- **Looping** – a series of instructions are executed repeatedly
Start

Read sheets

Set stamps = sheets / 5

Round stamps up to next whole number

Display stamps

End

input

processing

processing

output
If condition is true Then
  Process step(s) 1
Else
  Process step(s) 2
End If
Looping
Flow Chart

Do While condition is true
Process step(s)
Loop

Is condition true?

Yes

Process step(s)

No
Direction of Numbered NYC Streets Algorithm

- **Problem:** Given a street number of a one-way street in New York City, decide the direction of the street, either eastbound or westbound

- **Discussion:** in New York City even numbered streets are Eastbound, odd numbered streets are Westbound
Start

Get street

Is street even?

No

Display westbound

Yes

Display eastbound

End
Pseudocode

**Program**: Determine the direction of a numbered NYC street

Get street

If street is even Then
    Display Eastbound
Else
    Display Westbound
End If
Hierarchy Chart

Street direction program

Get street number

Decide whether street number is even or odd

Display direction
Class Average Algorithm

**Problem:** Calculate and report the average grade for a class

**Discussion:** The average grade equals the sum of all grades divided by the number of students

**Input:** Student grades

**Processing:** Find sum of the grades; count number of students; calculate average

**Output:** Average grade
Flowchart

Start

Initialize counter and sum to 0

Are there more data?

No

Yes

Get next grade

Increment counter

Add grade to sum

Set average to sum/counter

Display average

counter and sum start at 0

read next grade

add 1 to counter

accumulate sum of grades

find the average

display the answer

End
**Pseudocode**

**Program**: Determine average grade of a class  
Initialize Counter and Sum to 0  
Do While there are more data  
   Get the next Grade  
   Add the Grade to the Sum  
   Increment the Counter  
Loop  
Compute Average = Sum / Counter  
Display Average
Hierarchy Chart

Class-average problem

- Get grade
- Compute sum and number of grades
- Calculate average
- Display average
Comments

• When tracing a flowchart, begin at the start symbol and follow the flow lines to the end symbol.
• Testing an algorithm at the flowchart stage is known as desk checking.
• Flowcharts, pseudocode, and hierarchy charts are program planning tools that are independent of the language being used.
Tips and Tricks of Flowcharts

• Flowcharts are time-consuming to write and difficult to update

• For this reason, professional programmers are more likely to favor pseudocode and hierarchy charts

• Because flowcharts so clearly illustrate the logical flow of programs, they are a valuable tool in the education of programmers