Chapter 2 - Problem Solving

- Program Development Cycle
- Programming Tools

Terminology tip

- A computer program may also be called:
  - Project (Visual Studio 6 term)
  - Application (Generic term)
  - Solution (Visual Studio .NET term)
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 class

- How fast is a car traveling if it goes 50 miles in 2 hours?
- **Output**: a number giving the rate of speed in miles per hour
- **Input**: the distance and time the car has traveled
- **Process**: rate = distance/time

Pictorial representation of the problem solving process

Input → Processing → Output
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 Tips

• Always have a plan before trying to write a program
• The more complicated the problem, the more complex the plan must be
  • Start making a seared foie gras with lavender honey without knowing the recipe?
  • Lots of programmers try the same thing with their programs
• Planning and testing before coding saves time coding
Simplified 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.).

Program development cycle continued

4. **Code:** Translate the algorithm into a programming language.  
   Try to avoid jumping straight to this step, which can result in a sloppy solution!
5. **Test and debug:** Locate and remove any errors in the program.
6. **Complete the documentation:**  
   Organize all the material that describes the program.
2.2 Programming Tools

- Three tools used to convert algorithms into computer programs (there are more):
  - **Flowcharts** - Graphically depict the logical steps to carry out a task and show how the steps relate to each other.
  - **Pseudocode** - Uses English-like phrases with some VB.NET terms to outline the program.
  - **Hierarchy charts** - Show how the different parts of a program relate to each other.

Algorithms

- A step by step series of instructions for solving a problem (a recipe is an example of an algorithm)
- Algorithms are key to solving many problems efficiently
Problem solving example

- How many stamps do 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. *(input)*
2. Divide Sheets by 5. *(processing)*
3. Round the quotient up to the next highest whole number; call it Stamps. *(processing)*
4. Reply with the number Stamps. *(output)*
Flowcharts

- Graphically depict the logical steps to carry out a task and show how the steps relate to each other.

Flowchart symbols

<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>
<tr>
<td></td>
<td>Decision</td>
<td>Used for any logic or comparison operations. Unlithe 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 &quot;yes&quot; or &quot;no.&quot;</td>
</tr>
</tbody>
</table>
Flowchart symbols continued

- **Connector**: Used to join different flowlines.
- **Offpage Connector**: Used to indicate that the flowchart continues to a second page.
- **Predefined Process**: Used to represent a group of statements that perform one processing task.
- **Annotation**: Used to provide additional information about another flowchart symbol.

---

Flowchart example

1. **Start**
2. "Read sheet" (input)
3. "Set stamps - sheets ± 5" (processing)
4. "Round stamps up to next whole number" (processing)
5. "Display stamps" (output)
6. **End**
Pseudocode

- Uses English-like phrases with some VB.NET terms to outline the task.
- The idea is to represent the algorithm in a form that is in between pure English and actual running code.

Pseudocode example

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 charts

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

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

Hierarchy charts example

Each entry represents a module
**Divide-and-conquer method**

- Used in problem solving – take a large problem and break it into smaller problems solving the small ones first
- Breaks a problem down into modules

**Statement structure**

- Sequence – follow 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 over and over
Decision flow chart

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?
    No
    Exit Do
    Yes
      Process step(s)
Direction of Numbered NYC Streets Algorithm

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

• **Discussion:** in New York even numbered streets are Eastbound, odd numbered streets are Westbound
**Pseudocode**

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

Get street

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

More detail; How could we determine if a street number is even or odd?

---

**Hierarchy Chart**

Street direction program

<table>
<thead>
<tr>
<th>Get street number</th>
<th>Decide whether street number is even or odd</th>
<th>Display direction</th>
</tr>
</thead>
</table>
Class Average Algorithm

• **Problem:** Calculate and report the grade-point average for a class
• **Discussion:** The average grade equals the sum of all grades divided by the number of students

**Output:** Average grade

**Input:** Student grades

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

Flowchart
**Pseudocode**

**Program**: Determine the average grade of a class

1. Initialize Counter and Sum to 0
2. Do While there are more data
   - Get the next Grade
   - Add the Grade to the Sum
   - Increment the Counter
3. Loop
4. Compute Average = Sum / Counter
5. Display Average

**Hierarchy Chart**

- Class average problem
  - Get grade
  - Compute sum and number of grades
  - Calculate average
  - Display average
Algorithm Selection

- Our choice of algorithm can have a large impact on the performance of the program
- Consider the problem of searching for a name in a phone book
- The phone book is sorted by name

Algorithm 1 – Linear Search

- Search through the list of names for our target starting at the beginning and go through them one at a time until we reach the end

Apple, Bob
Atto, Tom
Attrick, Jerry
DeBanque, Robin
Fresco, Al
Guini, Lynn
Oki, Kerry
Wright, Eaton

Must check up to 8 names
What if millions of names?
Algorithm 2 – Binary Search

- Takes advantage of the fact that the list of names is sorted
- Start at the name in the middle of the list and compare to the target
  - If equal, there is a match!
  - If the target is alphabetically less, repeat the process on the first half of the list
  - If the target is alphabetically greater, repeat the process on the second half of the list
  - Stop and no match if the list is ever empty

Searching for Guini, Lynn

Apple, Bob
Atto, Tom
Attrick, Jerry
DeBanque, Robin
Fresco, Al
Guini, Lynn
Oki, Kerry
Wright, Eaton

Only checked 3 names!

What if millions of names?
Algorithm Analysis

- Algorithm 1 runs in time linear to the number of names;
  - 1 million names requires searching on average 500,000 names, all 1 million in the worst case
- Algorithm 2 runs in time $\log_2$ to the number of names
  - This is because we cut the size in half each time
  - For 1 million names, $\log_2 1000000$ is about 20.
  - Much better performance than algorithm 1!

Comments

- When tracing a flow chart, start at the start symbol and follow the flow lines to the end symbol
- Testing an algorithm at the flow chart stage is known as desk checking
- Flowcharts, pseudocode, and hierarchy charts are program planning tools that are not dependent on the programming language being used
Comments continued

- There are four primary logical programming constructs
  - sequence
  - decision
  - loop
  - unconditional branch

Unconditional branch

- Appear in some languages as Goto statements
  - Considered poor programming
- Involves jumping from one place in a program to another
- Structured programming uses the sequence, decision, and loop but forbids unconditional branch
**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 programming techniques, they are a valuable tool in the education of programmers

**Tips and tricks of pseudocode**

- There are many styles of pseudocode
- Some programmers use an outline form
- Some use a form that looks almost like a programming language
- The case studies of this text focuses on the primary tasks to be performed by the program and leaves many of the routine details to be completed during the coding process
Tips and tricks of hierarchy charts

- Many people draw rectangles around each item in a hierarchy chart
- In the text, rectangles are omitted to encourage the use of hierarchy charts by making them easier to draw

Formal Modeling and Design Tools

- The Unified Modeling Language (UML) is becoming a standard way to specify a design
- Baseline UML
  - Use Case Diagrams
  - Class Diagrams
  - Package Diagrams
  - Activity Diagrams
  - State Transition Diagrams
  - Interaction Diagrams
  - Deployment Diagrams