Chapter 2: Algorithm Discovery and Design

Objectives

In this chapter, you will learn about

- Representing algorithms
- Examples of algorithmic problem solving
Initially, set the value of the variable carry to 0 and the value of the variable \( i \) to 0. When these initializations have been completed, begin looping as long as the value of the variable \( i \) is less than or equal to \((m - 1)\). First, add together the values of the two digits \( a_j \) and \( b_j \) and the current value of the carry digit to get the result called \( c_j \). Now check the value of \( c_j \) to see whether it is greater than or equal to 10. If \( c_j \) is greater than or equal to 10, then reset the value of carry to 1 and reduce the value of \( c_j \) by 10; otherwise, set the value of \( carry \) to zero. When you are done with that operation, add 1 to \( i \) and begin the loop all over again. When the loop has completed execution, set the leftmost digit of the result \( c_m \) to the value of carry and print out the final result, which consists of the digits \( c_m, c_{m-1}, \ldots, c_0 \). After printing the result, the algorithm is finished, and it terminates.

Figure 2.1
The Addition Algorithm of Figure 1.2 Expressed in Natural Language

---

**Representing Algorithms**

- **Natural language**
  - Language spoken and written in everyday life
  - Examples: English, Spanish, Arabic, and so on
  - Problems with using natural language for algorithms
    - Verbose
    - Imprecise
    - Relies on context and experiences to give precise meaning to a word or phrase
Representing Algorithms (continued)

- High-level programming language
  - Examples: C++, Java
  - Problem with using a high-level programming language for algorithms
    - During the initial phases of design, we are forced to deal with detailed language issues

```java
{ 
    int i, m, Carry;
    int[] a = new int[100];
    int[] b = new int[100];
    int[] c = new int[100];
    m = Console.readInt();
    for (int j = 0; j < m-1; j++) 
    { 
        a[j] = Console.readInt();
        b[j] = Console.readInt();
    }
    Carry = 0;
    i = 0;
    while (i < m) 
    { 
        c[i] = a[i] + b[i] + Carry;
        if (c[i] > -10)
            .
    }
}
```

Figure 2.2
The Beginning of the Addition Algorithm of Figure 1.2 Expressed in a High-Level Programming Language
Pseudocode

- English language constructs modeled to look like statements available in most programming languages

- Steps presented in a structured manner (numbered, indented, and so on)

- No fixed syntax for most operations is required

---

**Average Miles per Gallon Algorithm (Version 1)**

<table>
<thead>
<tr>
<th>STEP</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get values for gallons used, starting mileage, ending mileage</td>
</tr>
<tr>
<td>2</td>
<td>Set value of distance driven to (ending mileage – starting mileage)</td>
</tr>
<tr>
<td>3</td>
<td>Set value of average miles per gallon to (distance driven ÷ gallons used)</td>
</tr>
<tr>
<td>4</td>
<td>Print the value of average miles per gallon</td>
</tr>
<tr>
<td>5</td>
<td>Stop</td>
</tr>
</tbody>
</table>

---

Figure 2.3
Algorithm for Computing Average Miles per Gallon
Pseudocode (continued)

- Less ambiguous and more readable than natural language
- Emphasis is on process, not notation
- Well-understood forms allow logical reasoning about algorithm behavior
- Can be easily translated into a programming language

Sequential Operations

- Types of algorithmic operations
  - Sequential
  - Conditional
  - Iterative
Sequential Operations (continued)

- Computation operations
  - Example
    - Set the value of “variable” to “arithmetic expression”
    - Add var1 and constantA to get var2
  - Variable
    - Named entity (e.g. storage location) that can hold a data value

Sequential Operations (continued)

- Input operations
  - To receive data values from the outside world
  - Example
    - Get a value for \( r \), the radius of the circle
- Output operations
  - To send results to the outside world for display
  - Example
    - Print the value of \( Area \)
Conditional and Iterative Operations

- Sequential algorithm
  - Also called straight-line algorithm
  - Executes its instructions in a straight line from top to bottom and then stops

- Control operations
  - Conditional operations
  - Iterative operations
Conditional and Iterative Operations (continued)

- Conditional operations
  - Ask questions and choose alternative actions based on the answers
  - Example
    - If \( x \) is greater than 0 then
      \[
      \text{abs}_x = x;
      \]
    - else
      \[
      \text{abs}_x = -x;
      \]

Flow Chart of a Conditional Operation
Average Miles per Gallon Algorithm (Version 2)

<table>
<thead>
<tr>
<th>STEP</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get values for gallons used, starting mileage, ending mileage</td>
</tr>
<tr>
<td>2</td>
<td>Set value of distance driven to (ending mileage − starting mileage)</td>
</tr>
<tr>
<td>3</td>
<td>Set value of average miles per gallon to (distance driven ÷ gallons used)</td>
</tr>
<tr>
<td>4</td>
<td>Print the value of average miles per gallon</td>
</tr>
<tr>
<td>5</td>
<td>If average miles per gallon is greater than 25.0 then</td>
</tr>
<tr>
<td>6</td>
<td>Print the message 'You are getting good gas mileage' Else</td>
</tr>
<tr>
<td>7</td>
<td>Print the message 'You are NOT getting good gas mileage'</td>
</tr>
<tr>
<td>8</td>
<td>Stop</td>
</tr>
</tbody>
</table>

Figure 2.5
Second Version of the Average Miles per Gallon Algorithm

Conditional and Iterative Operations (continued)

- Iterative operations
  - Perform “looping” behavior, repeating actions under some condition
  - Loop
    - The repetition of a block of instructions
Conditional and Iterative Operations (continued)

- Examples
  - while \( j > 0 \) do
    - set \( s \) to \( s + a_j \)
    - set \( j \) to \( j - 1 \)
  - repeat
    - print \( a_k \)
    - set \( k \) to \( k + 1 \)
    - until \( k > n \)

Conditional and Iterative Operations (continued)

- Components of a loop
  - Continuation condition
  - Loop body

- Infinite loop: An error
  - The continuation condition never becomes false
### Average Miles per Gallon Algorithm (Version 3)

<table>
<thead>
<tr>
<th>Step</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>response = Yes</td>
</tr>
<tr>
<td>2</td>
<td>While (response = Yes) do steps 3 through 11</td>
</tr>
<tr>
<td>3</td>
<td>Get values for gallons used, starting mileage, ending mileage</td>
</tr>
<tr>
<td>4</td>
<td>Set value of distance driven to (ending mileage − starting mileage)</td>
</tr>
<tr>
<td>5</td>
<td>Set value of average miles per gallon to (distance driven ÷ gallons used)</td>
</tr>
<tr>
<td>6</td>
<td>Print the value of average miles per gallon</td>
</tr>
<tr>
<td>7</td>
<td>If average miles per gallon &gt; 25.0 then</td>
</tr>
</tbody>
</table>
| 8    | Print the message ‘You are getting good gas mileage’  
| 9    | Else |
| 10   | Print the message ‘You are NOT getting good gas mileage’  
| 11   | Print the message ‘Do you want to do this again? Enter Yes or No’  
| 12   | Get a new value for response from the user  
|      | Stop |

#### Figure 2.7
Third Version of the Average Miles per Gallon Algorithm

---

### Conditional and Iterative Operations (continued)

- **Pretest loop**
  - Continuation condition tested at the beginning of each pass through the loop
  - It is possible for the loop body to never be executed

- **While loop**
Flowchart of a Pretest Loop

Conditional and Iterative Operations (continued)

- Posttest loop
  - Continuation condition tested at the end of loop body
  - Loop body must be executed at least once
  - Do … While loop
Flowchart of a Posttest Loop

**Summary of Pseudocode Language Instructions**

- **Computation:**
  - Set the value of "variable" to "arithmetic expression"

- **Input/Output:**
  - Get a value for "variable", "variable"...
  - Print the value of "variable", "variable", ...
  - Print the message 'message'

- **Conditional:**
  - If "a true/false condition" is true then
    - first set of algorithmic operations
  - Else
    - second set of algorithmic operations

- **Flowchart:**
  - True arrow from "statement" to "condition evaluated".
  - False arrow from "condition evaluated" to "false".
  - Loop arrow from "false" back to "true".

**Figure 2.9**
Examples of Algorithmic Problem Solving

- Go Forth and Multiply: Multiply two numbers using repeated addition
- Sequential search: Find a particular value in an unordered collection
- Find maximum: Find the largest value in a collection of data
- Pattern matching: Determine if and where a particular pattern occurs in a piece of text

(Algorithmic) Problem Solving

- Abstraction
  - Separating high-level view from low-level details
  - Key concept in computer science
  - Makes difficult problems intellectually manageable
  - Allows piece-by-piece development of algorithms
(Algorithmic) Problem Solving

- Top-down design
  - When solving a problem, especially a complex one
    - Create high-level operations in the first draft of an algorithm
    - After drafting the outline of the algorithm, return to the high-level operations and elaborate each one
    - Repeat until all operations are primitives
    - (use examples to help develop and test ideas)
  - **divide and conquer**

Once an algorithm has been developed, it may itself be used in the construction of other, more complex algorithms

- Library
  - A collection of useful algorithms
  - An important tool in algorithm design and development
Summary

- Algorithm design is a first step in developing an algorithm
- Algorithm design must
  - Ensure the algorithm is correct
  - Ensure the algorithm is sufficiently efficient
- Pseudocode is used to design and represent algorithms

Summary

- Pseudocode is readable, unambiguous, and able to be analyzed
- Algorithm design is a creative process; uses multiple drafts and top-down design to develop the best solution
- Abstraction is a key tool for good design