



# Chapter 1 (Schneider)

## Lecture #1: Intro to Computer Science

Dr. Alfred R. Watkins

[alwatkins@spelman.edu](mailto:alwatkins@spelman.edu)

404.270.5768

Copyright © 2007 Spelman College

Department of Computer and  
Information Sciences (CIS)

CIS121



## Objectives

In this chapter, you will learn about:

- The definition of a computer
- The definition of computer science
- The definition of an algorithm

Copyright © 2007 Spelman College

Department of Computer and  
Information Sciences (CIS)

CIS121  
(Lecture #2.1 Slide #2)



## What is a Computer?

---

- A computer is a device that under the direction and control of a program performs four basic functions:
  - **input**
  - **output**
  - **processing**
  - **storage.**



## What is computer science?

---

- Computer science is the study of algorithms, including
  - **Their formal and mathematical properties**
  - **Their hardware realizations**
  - **Their linguistic realizations**
  - **Their applications**



## What is an algorithm?

- An algorithm is a well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time.



## Addition Algorithm in English

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*, and *b*, and the current value of the carry digit to get the result called  $c_i$ . Now check the value of  $c_i$  to see whether it is greater than or equal to 10. If  $c_i$  is greater than or equal to 10, then reset the value of *carry* to 1 and reduce the value of  $c_i$  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  $c_m c_{m-1} \dots c_0$ . After printing the result, the algorithm is finished, and it terminates.



## Representing Algorithms

- 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



## A C++ Program for Addition

```
{
  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)
      .
      .
      .
  }
}
```



## Pseudocode

---

- English language constructs modeled to look like statements available in most programming languages
- Steps presented in a structured manner (numbered, indented, etc.)
- No fixed syntax for most operations is required



## 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”**
  - **Variable**
    - **Named 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



## Algorithm: Average Miles Per Gallon

---

1. **Get values** for *gallons used*, *starting mileage*, and *ending mileage*
2. **Set value** of *distance driven* to (*ending mileage* - *starting mileage*)
3. **Set value** of *average miles per gallon* to (*distance driven* / *gallons used*)
4. **Print the value** of *average miles per gallon*
5. **Stop**



## 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 25 then  
    print  $x$
    - else  
    print  $x$  times 100



## Algorithm: Average Miles Per Gallon (Version 2)

---

1. **Get values** for *gallons used*, *starting mileage*, and *ending mileage*
2. **Set value** of *distance driven* to (*ending mileage* - *starting mileage*)
3. **Set value** of *average miles per gallon* to (*distance driven* / *gallons used*)
4. **Print the value** of *average miles per gallon*
5. **if** *average miles per gallon* is greater than 25.0 **then**
6.     **Print** the message "You are getting good gas mileage"
7. **Else**
8.     **Print** the message "You are NOT getting good gas mileage"
9. **STOP**



## Conditional and Iterative Operations (continued)

---

- Iterative operations
  - Perform "looping" behavior; repeating actions until a continuation condition becomes false
  - 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
  - **The continuation condition never becomes false**
  - **An error**



## Algorithm: Average Miles Per Gallon (Version 3)

---

1. **Set** the value of response to Yes
2. **While** response is equal to Yes do steps 3 - 12
3.     **Get values** for *gallons used*, *starting mileage*, and *ending mileage*
4.     **Set value** of *distance driven* to (*ending mileage* - *starting mileage*)
5.     **Set value** of *average miles per gallon* to (*distance driven* / *gallons used*)
6.     **Print the value** of *average miles per gallon*
7.     **if** *average miles per gallon* is greater than 25.0 **then**
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** value of response from the user
13. **STOP**



## Conditional and Iterative Operations

---

- 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



## 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



## Pseudocode Summary

### 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

### ITERATIVE:

While ("a true/false condition") do step *i* through step *j*

Step *i*: operation

.

.

.

Step *j*: operation

While ("a true/false condition") do

operation

.

.

.

operation

End of the loop

Do

operation

operation

.

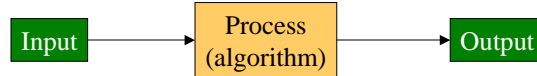
.

.

While ("a true/false condition")



## Basic Steps to Writing an Algorithm



1. Identify the input data, and create variable names for the input data
2. Identify the output data, and create variable names for the output data
  - If there are some variables that are the same, then that is okay
3. Solve the problem by hand (not as an algorithm) using the variables you have identified
  - Make note of the steps and formulas you are using as you solve it by hand
    - If you repeat steps over and over then that is a loop
4. Begin writing your algorithm
  - Input statements first
  - The main process of your algorithm
  - Output statements
5. Check your algorithm with a table trace



## An Example

- Write an algorithm that gets N numbers, computes the average of those N numbers, and prints out the average.
  - Analysis
    - What are the inputs?
    - What are the outputs?
    - What are the formulas/processes you have to do to solve this by hand?
    - Are there any special conditions?
  - Design
    - Write the algorithm
  - Test
    - Perform table trace



## Class Exercises

---

- Algorithm Practice Problems, each student should do the following:
  - Choose 1 of the 4 algorithms on page 47 for the class to write and present in class
  - Choose 1 of the 4 algorithms on page 54 for the class to write and present in class
  - Check your answers with a table trace