ITM-604 Program Concepts

School of Informatics

Management of Information Technology, 1/2015

Chapter 1
Program Design

Lecturer:
Nichnan
Kittiphattanabawon

Students:
MIT Year 1

August 31, 2015

Outline

Contents

Objectives

After completing this lesson, students will be able to:

- Describe the steps in the program development process.
- Introduce algorithms, pseudocode and flowchart
- Define the three basic control structures
- Illustrate the three basic control structure using pseudocode and flowchart
- Develop an algorithm using sequence control structure

Problems, Algorithms, and Programs

Problems

- A task to be performed
- A function of inputs to outputs

Algorithms

- A method/process followed to solve a problem
- A recipe for solving a problem whose steps are concrete and unambiguous

Programs

- A computer program of an algorithm in some programming language
- An instantiation of an algorithm in a computer programming language

The situation of problems, algorithms, and programs

- **Any problem there are many possible algorithms**
- **Any algorithm there are many possible programs**

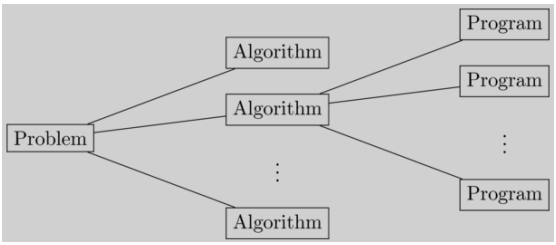

Figure 1: The Situation among Problems, Algorithms, and Programs. (Roche, 2014)

Steps in Program Development

- Seven basic stpes in the development of a program
  1. Define the problem
  2. Outline the solution
  3. Develop the outline into an algorithm
  4. Test the algorithm for correctness
  5. Code the algorithm into a specific programming language
  6. Run the program on the computer
  7. Document and maintain the program

Introduction to Algorithms

Introduction to Algorithms
Algorithm
A program must be systematically and properly designed before coding begins.

- An algorithm is like a recipe.
  - Lists of steps involved in accomplishing a task.
    - unambiguous instructions
    - ordered instructions

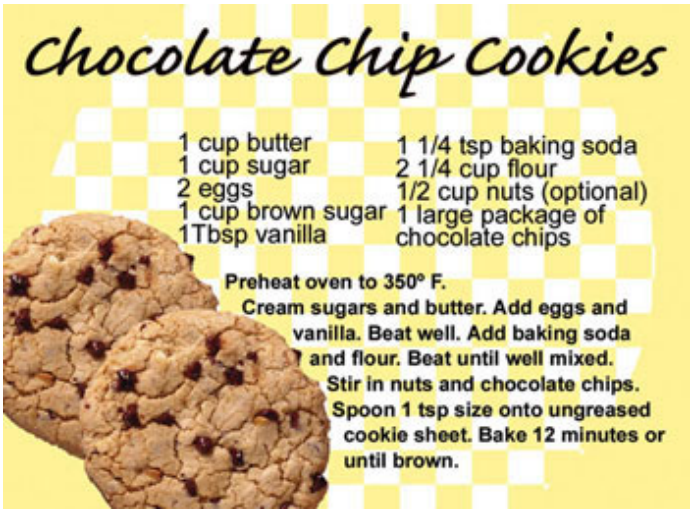

Figure 2: Recipe. (www.gone-ta-pott.com)

Definition of an algorithm in programming terms

- A set of detailed and ordered instructions developed to describe the processes necessary to produce the desired *output* from a given *input*

e.g., Algorithm of adding up a list of prices on a pocket calculator

1. Turn on calculator
2. Clear calculator
3. Repeat the following instructions
    - Key in baht amount
    - Key in decimal point (.)
    - Key in satangs amount
    - Press addition (+) key
4. Until all prices have been entered
5. Write down total price
6. Turn off calculator

Popular ways of representing algorithms

- Pseudocode
- Flowchart
- Nassi-Schneiderman diagrams

Pseudocode

What Is Pseudocode?

Pseudocode
What Is Pseudocode?
Pseudocode is easy to read and write

- Structured English
    - Formalised and abbreviated to look like high-level computer language
- No standard psedudocode
    - Depend on author styles

      e.g.,
        * Simple English

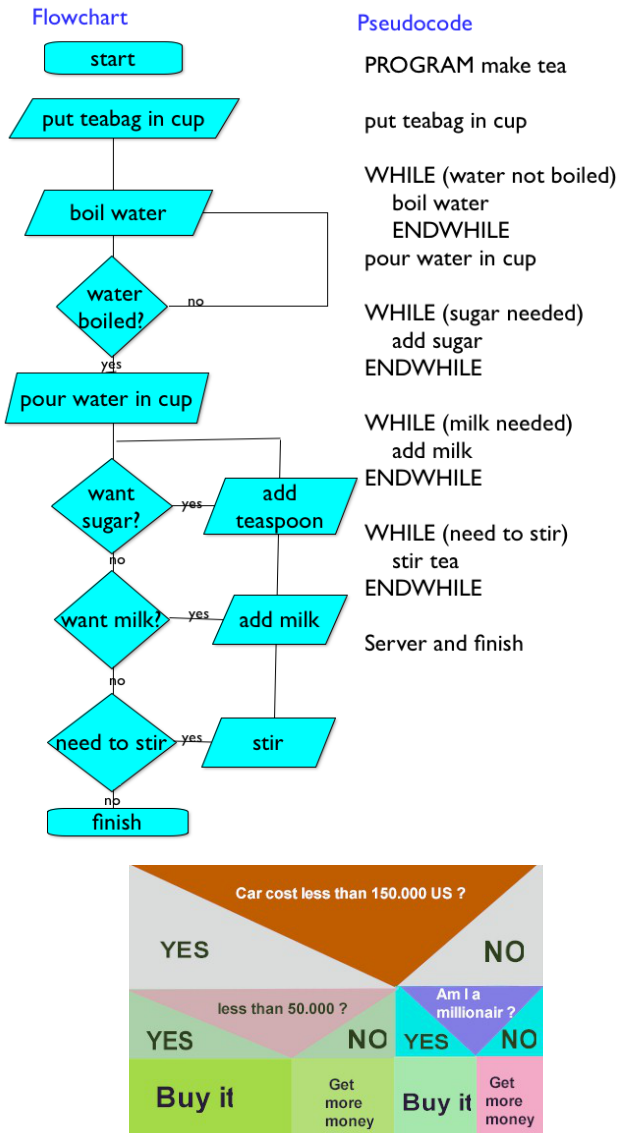

Figure 3: Flowchart, Pseudocode, and Nassi Schneider Diagrams. (www.csgcse.co.uk, www.thocp.net)

* One line per each instruction
* Top to bottom with one entry and one exit
* Indentation
* Keywords
* Groups of statements

How to Write Pseudocode?

Pseudocode
How to Write Pseudocode?
Six basic computer operations

1. A computer can receive information
2. A computer can put out information
3. A computer can perform arithmetic
4. A computer can assign a value to a variable or memory location
5. A computer can compare two variables and select one of two alternate actions
6. A computer can repeat a group of actions

---

1. A computer can receive information

The computer is required to receive information

- **Get**
  - When the algorithm is to receive input from the keyboard.

    e.g.,
    - **Get** student_id
    - **Get** height, weight
- **Read**
  - When the algorithm is to receive input from a record on a file

    e.g.,
    - **Read** student_name
    - **Read** subject1, subject2, subject3

2. A computer can put out information

The computer is required to supply information or output to a device

- **Print**
  - When the output is to be sent to a printer

    e.g.,
    - **Print** "Happy Birthday to You"

- **Write**
  - When the output is to be written to a file

    e.g.,
    - **Write** student record to master file
- **Put, Output** or **Display**
  - When the output is to be written to the screen

    e.g.,
    - **Put** student_id, student_name
    - **Output** GPA
    - **Display** "You got A"
- **Prompt**
  - When the algorithm is to send a message to the screen, which requires the user to respond
  - Usually used before **Get**

    e.g.,
    - **Prompt** for student_mark
    - **Get** student_mark

3. A computer can perform arithmetic

The computer is required to perform some sort of mathematical calculation, or formula

- **Compute** or **Calculate**
  - When the algorithm is to perform a calculation
- **+, -, *, /, ()** (actual mathematical symbols) or **Add, Subtract, Multiply, Divide** (the words)

  e.g.,
  - total = total + quiz1 (or **Add** quiz1 to total)
  - **Compute** C = (F-32)*5/9
  - **Calculate** triangle_area = 1/2*base*height

4. A computer can assign a value to a variable or memory location

- **Initialize** or **Set**
  - When giving data an initial value

    e.g.,
    - **Initialize** total to zero
    - **Set** student_count to 0
- = or ←
  - When assigning a value as a result of some processing

    e.g.,
    - total = cost+tax
    - score ← midterm+final
- **Save** or **Store**
  - When keeping a variable for later use

    e.g.,
    - **Save** customer_id in last_customer_id
    - **Store** student_id in last_student_id

5. A computer can compare two variables and select one of two alternate actions

The computer is required to compare two variables

Then select one of two alternate actions

- **IF**
  - When establishing the comparison of data
- **THEN**
  - When determining the first choice of alternatives
- **ELSE**
  - When determining the second choice of alternatives

e.g.,

IF score > 49 **THEN**
    Display "PASS"
ELSE
    Display "FAIL"

6. A computer can repeat a group of actions

The computer is required to repeat a sequence of processing steps

- **DOWHILE**
  - When establishing the condition for the repetition of a group of actions
- **ENDDO**
  - A delimiter of **DOWHILE**
  - As soon as the condition for the repetition is found false, control passes to the next statement after the **ENDDO**

e.g.,

DOWHILE student_total ¡ 30
    Read student record
    Print student_id, student_name, GPA to report
    student_total = student_total+1
ENDDO

Flowchart

What is Flowchart?

Flowchart
What is Flowchart?
Flowcharts are an alternative method of representation algorithms

- Flowcharts are popular
  - Graphically represent the program logic
  - Easy to learn

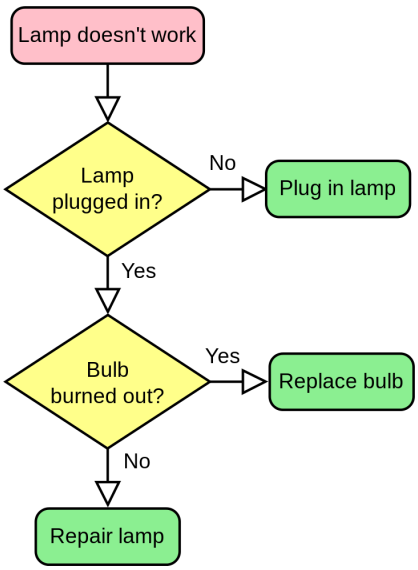

Figure 4: A simple flowchart representing a process for dealing with a non-functioning lamp. (en.wikipedia.org)

How to Write Flowchart?

Flowchart
How to Write Flowchart?
Six standard flowchart symbols

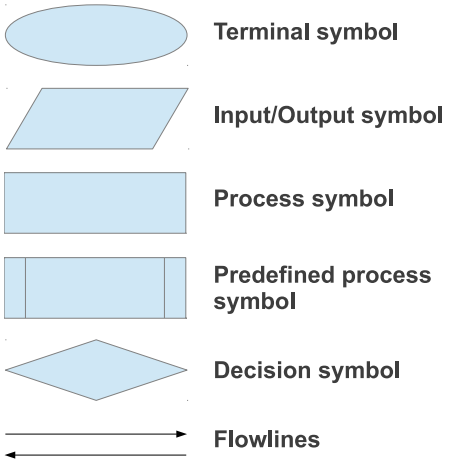

Figure 5: Six standard flowchart symbols.

- Terminal symbol
  - The starting or stopping point in the logic
- Input/Output symbol
  - An input or output process
    - Reading input
    - Writing output
- Process symbol
  - A single process
    - Assigning a value
    - Performing a calculatin
- Predefined process symbol
  - A module
    - A predefined process that has its own flowchart
- Decision symbol
  - A decision in the logic
    - Comparison of two values
    - Alternative paths (true or false)
- Flowlines
  - Connection of symbols
    - Top to bottom
    - Left to Right

The Structure Theorem

A structured framework for representing a solution algorithm

- Three basic control structures
  1. Sequence
  2. Selection
  3. Repetition

Sequence

- Straightforward execution of one processing step after another

  statement a
  statement b
  statement c

- Represents the first four basic computer operations
  - Receive information
  - Put out information
  - Perform arithmetic
  - Assign values

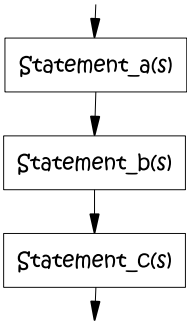

Figure 6: Sequence structure.

Selection

- Presentation of a condition and the choice between two actions
  - The choice depending on whether the condition is true or false
- Represents the decision-making abilities of the computer
- Illustrates the fifth basic computer operation
  - Compare two variables and select one of two alternate actions

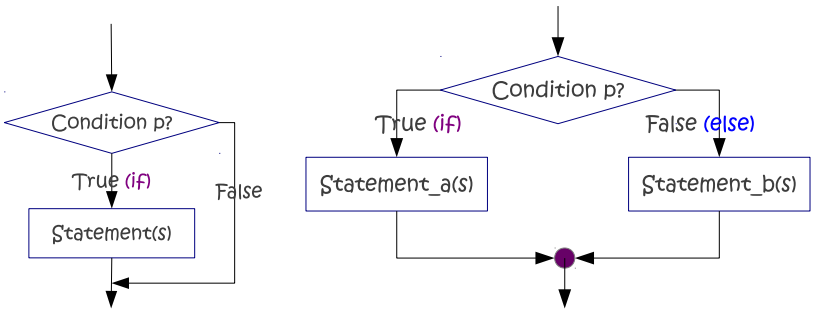

Figure 7: Selection control structure.

Repetition

- Presentation of a set of instruction to be performed repeatedly
  - As long as the condition is true
- Block statement is executed again and again until a terminating condition occurs
- Illustrates the sixth basic computer operation to repeat a group of actions.
  - Repeat a group of actions.

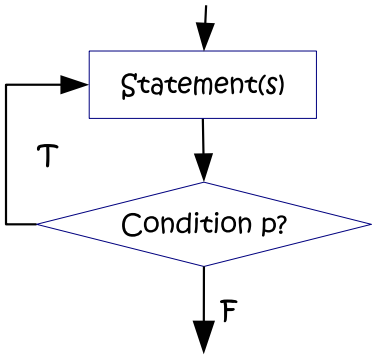

Figure 8: Repetition control structure.

Developing an Algorithm

Three steps in the development of an algorithm

1. Defining the problem
2. Designing a solution algorithm
3. Checking the solution algorithm

Defining the Problem

Developing an Algorithm
Defining the Problem
Defining the program is the first step in the development of a computer program

- Carefully reading and rereading the problem
- Seeking additional information
  - data
  - formula

Dividing a problem into three components to help the initial analysis

1. Input
   - a list of the source data provided to the problem
2. Output
   - a list of the output required
3. Processing
   - a list of actions needed to produce the required outputs

$\Longrightarrow$ **Write down each components into a defining diagram**

A simple diagram, called a defining diagram

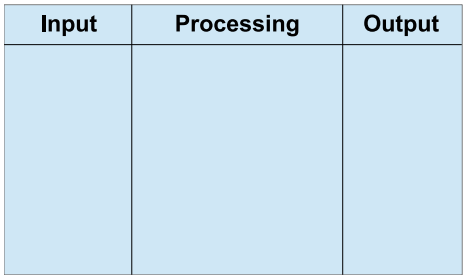

Figure 9: Defining Diagram.

Designing a Solution Algorithm

Developing an Algorithm
Designing a Solution Algorithm
Write the steps required to solve the problem

- Begin with a rough sketch of the steps
- Alter or delete later
  - trail-and-error process

$\Longrightarrow$ **Write down a solution algorithm**

- **Pseudocode**
- **Flowchart**
- **Nassi Schneiderman diagram**

Checking the Solution Algorithm

Developing an Algorithm
Checking the Solution Algorithm
Testing for correctness

- If not detected, the errors can be passed on to the program
- Much easier to detect errors in an algorithm than in the corresponding program code

$\Longrightarrow$ **Spend a few minutes desk checking the solution algorithm**

Steps in desk checking an algorithm

1. Choose two or three simple input test cases
2. Establish what the expected result should be for each test case
3. Make a table on a piece of paper of the relevant variable names within the algorithm
4. Walk the first test case through the algorithm, line by line, until the algorithm has reached its logical end
5. Repeat the walk-through process using the other test data cases
6. Check that the expected result established in Step 2 matches the actual result developed in Step

Summary

- *Seven steps* in program development
    1. Define the problem
    2. Outline the solution
    3. Develop an algorithm
    4. Test the algorithm
    5. Code
    6. Run the program
    7. Document and maintain
- An *algorithm* is a set of detailed, unambiguous and ordered instructions developed to describe the process necessary to produce the desired output from the given input
- *Pseudocode* is an English-like way of representing the algorithm
- A *flowchart* is a graphical representation of program logics, using a series of standard geometric symbols and lines.

- Six basic *computer operations*
    1. Receive information
    2. Put out information
    3. Perform arithmetic
    4. Assign a value to a variable

5. Decide between two alternate actions
6. Repeat a group of actions

- The Structure Theorem: Three basic *control structures*
  1. Sequence
  2. Selection
  3. Repetition
- Each control structure *associates* with each of the six basic computer operations

- Three steps for *developing an algorithm*
  1. Defining a problem (*what*)
     - Understand a problem before attempting to find a solution
  2. Designing a solution algorithm (*how*)
     - Find the solution and express it as an algorithm
  3. Checking the solution algorithm
     - Trace through the algorithm step by step

Outlook

- Selection control structures

References

[1] Lesley Anne Robertson. *Simple Program Design: A Step-by-Step Approach.* Fourth Edition, Thomson Course Technology, 2004.

[2] K. N. King. *C programming: A Modern Approach.* Second Edition, W.W. Norton & Company Inc., New York, 2008.

Sources of Pictures

[3] GCSE Computer Science. *Flowcharts & Pseudocode*, 2014. www.csgcse.co.uk [Online; accessed August 19, 2014].

[4] The time line. *[Nassi and Schneider develop a diagramming technique that produces "program structure diagrams" (PSD)]*, January, 20 2014. www.thocp.net [Online; accessed August 19. 2014].

[5] Ajune Wanis. *[Problem Solving Skills)]*, October 27, 2010. ajune-programming.blogspot.com [Online; accessed August 19, 2014].

[6] Wikipedia, the free encyclopedia. *[Flowchart)]*, August 16, 2014. en.wikipedia.org [Online; accessed August 20, 2014].