CN208
Introductory Computer Programming

Week 4:- Algorithm and Flowchart

By

Dr. Piya Techateerawat

Algorithm and Flowchart

- **Computer Program Design**
- Algorithm Development
- Flowchart

Computer Program Design

- As a summary,
  - Problem Consideration.
    - Understand, Brainstorm, Design
  - Coding.
    - Making it up or writing code.
  - Debugging.
    - Test, review, documentation.

Algorithm and Flowchart

- Computer Program Design
- **Algorithm Development**
- Flowchart

Algorithm Development

- ***<u>Algorithm</u>*** describes the step-by-step process that using **objects** in data structure in order to **accomplish the task** which is described by the **problem definition**.

Algorithm Development

- What is the problem ?
- Can we separate to small problem?
- Each small problem can by solve by more than one method ?
- Which one is most suitable for this problem?
- What objects, procedure, function do we need ?
- How this small problems link to each others?
- How do you know this will solve the problem correct?
- How do you know this will solve the correct problem?

Algorithm and Flowchart

- Computer Program Design
- Algorithm Development
- **Flowchart**

Flow Chart

- ***Flowchart*** represents the **steps in the algorithm** in a **graphical** manner.

Flow Chart

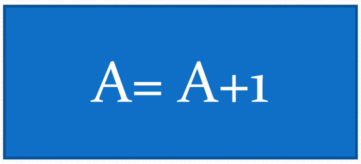
- **Symbols**
- Start and End Symbol

Flow Chart

- **Symbols**
- Flow Control

Flow Chart

- **Symbols**
- Processing Steps

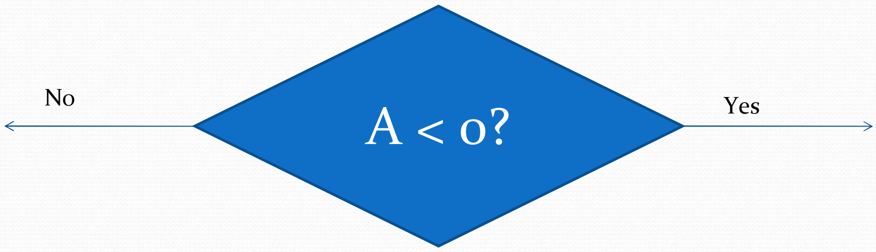


Flow Chart

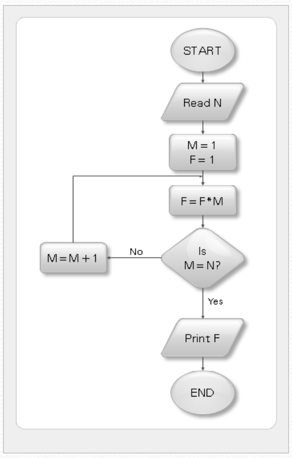
- **Symbols**
- Input/Output

Flow Chart

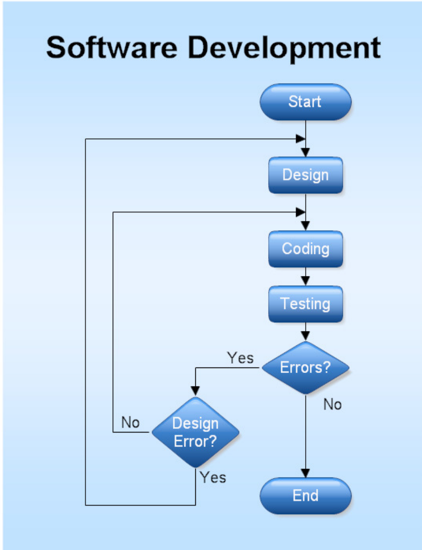
- **Symbols**
- Conditional or decision



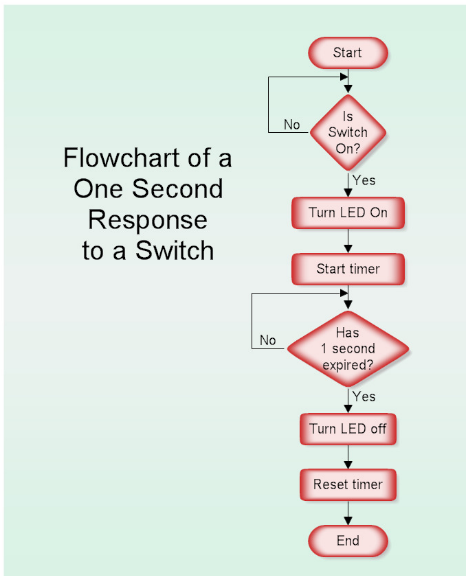
Flow Chart



Flow Chart



Flow Chart



MATLAB Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (3) Simple operations on vectors and matrices

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (A) Element-wise operations:

% These operations are done "element by element".  If two
% vectors/matrices are to be added, subtracted, or element-wise
% multiplied or divided, they must have the same size.

a = [1 2 3 4]';                  % A column vector
2 * a                            % Scalar multiplication
a / 4                            % Scalar division
b = [5 6 7 8]';                  % Another column vector
a + b                            % Vector addition
a - b                            % Vector subtraction
a .^ 2                           % Element-wise squaring (note the ".")
a .* b                           % Element-wise multiplication (note the ".")
a ./ b                           % Element-wise division (note the ".")

log([1 2 3 4])                   % Element-wise logarithm
round([1.5 2; 2.2 3.1])          % Element-wise rounding to nearest integer

% Other element-wise arithmetic operations include e.g. :
%   floor, ceil, ...
```

MATLAB Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (B) Vector Operations
% Built-in Matlab functions that operate on vectors

a = [1 4 6 3]                 % A row vector
sum(a)                        % Sum of vector elements
mean(a)                       % Mean of vector elements
var(a)                        % Variance of elements
std(a)                        % Standard deviation
max(a)                        % Maximum
min(a)                        % Minimum

% If a matrix is given, then these functions will operate on each column
%   of the matrix and return a row vector as result
a = [1 2 3; 4 5 6]            % A matrix
mean(a)                       % Mean of each column
max(a)                        % Max of each column
max(max(a))                   % Obtaining the max of a matrix
mean(a, 2)                    % Mean of each row (second argument specifies
                              %   dimension along which operation is taken)


[1 2 3] * [4 5 6]'            % 1x3 row vector times a 3x1 column vector
                              %   results in a scalar.  Known as dot product
                              %   or inner product.  Note the absence of "."

[1 2 3]' * [4 5 6]            % 3x1 column vector times a 1x3 row vector
                              %   results in a 3x3 matrix.  Known as outer
                              %   product.  Note the absence of "."
```

MATLAB Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (C) Matrix Operations:

a = rand(3,2)                % A 3x2 matrix
b = rand(2,4)                % A 2x4 matrix
c = a * b                    % Matrix product results in a 3x4 matrix

a = [1 2; 3 4; 5 6];         % A 3x2 matrix
b = [5 6 7];                 % A 1x3 row vector
b * a                        % Vector-matrix product results in
                             %   a 1x2 row vector
c = [8; 9];                  % A 2x1 column vector
a * c                        % Matrix-vector product results in
                             %   a 3x1 column vector

a = [1 3 2; 6 5 4; 7 8 9];   % A 3x3 matrix
inv(a)                       % Matrix inverse of a
eig(a)                       % Vector of eigenvalues of a
[V, D] = eig(a)              % D matrix with eigenvalues on diagonal;
                             %   V matrix of eigenvectors
                             %   Example for multiple return values!
[U, S, V] = svd(a)           % Singular value decomposition of a.
                             %   a = U * S * V', singular values are
                             %   stored in S

% Other matrix operations: det, norm, rank, ...
```

MATLAB Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (D) Reshaping and assembling matrices:

a = [1 2; 3 4; 5 6];         % A 3x2 matrix
b = a(:)                     % Make 6x1 column vector by stacking
                             %   up columns of a
sum(a(:))                    % Useful:  sum of all elements

a = reshape(b, 2, 3)         % Make 2x3 matrix out of vector
                             %   elements (column-wise)

a = [1 2]; b = [3 4];        % Two row vectors
c = [a b]                    % Horizontal concatenation (see horzcat)

a = [1; 2; 3];               % Column vector
c = [a; 4]                   % Vertical concatenation (see vertcat)

a = [eye(3) rand(3)]         % Concatenation for matrices
b = [eye(3); ones(1, 3)]

b = repmat(5, 3, 2)          % Create a 3x2 matrix of fives
b = repmat([1 2; 3 4], 1, 2) % Replicate the 2x2 matrix twice in
                             %   column direction; makes 2x4 matrix
b = diag([1 2 3])            % Create 3x3 diagonal matrix with given
                             %   diagonal elements
```

Reference

http://www.conceptdraw.com/products/img/cd5/article/top_down_flow_chart.gif@ 20 OCT 2008

http://www.rff.com/flowchart_one_second_response.htm@ 20 OCT 2008

http://www.rff.com/software_development.htm@ 20 OCT 2008

http://www.conceptdraw.com/products/img/cd5/article/top_down_flow_chart.gif@ 20 OCT 2008

Q & A