Sequential Algorithms

Lecture 3 – Sunday October 30, 2016

Outline

- What is an Algorithm?
- Representing Algorithms
- Pseudo-code
- Algorithm Implementation
- Types of Algorithms
- Sequential Algorithms
- Summary

Outline

- **What is an Algorithm?**
- Representing Algorithms
- Pseudo-code
- Algorithm Implementation
- Types of Algorithms
- Sequential Algorithms
- Summary

What is an Algorithm?

Computer science, or computing science, is the study of the theoretical foundations of information and computation and their implementation and application in computer systems.

Source: Wikipedia.org - the free encyclopedia

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 step-by-step specification of a method to solve a problem within a finite amount of time.

Muhammad ibn Musa al-Khwarizmi (780-850)
Mathematician, astronomer, astrologer and geographer

- An algorithm is a set of ordered steps for solving a problem, such as a mathematical formula or the instructions in a program.

Source: Computer Desktop Encyclopedia.

- An algorithm is a procedure (a finite set of well-defined instructions) for accomplishing some task which, given an initial state, will terminate in a defined end-state.

Source: Wikipedia

What is an Algorithm?

Data + Algorithms = Program

DATA
½ cup butter
1 cup sugar
2 eggs
…

Ingredients

+

ALGORITHM
cream butter
gradually, add sugar
break eggs
…

Recipe

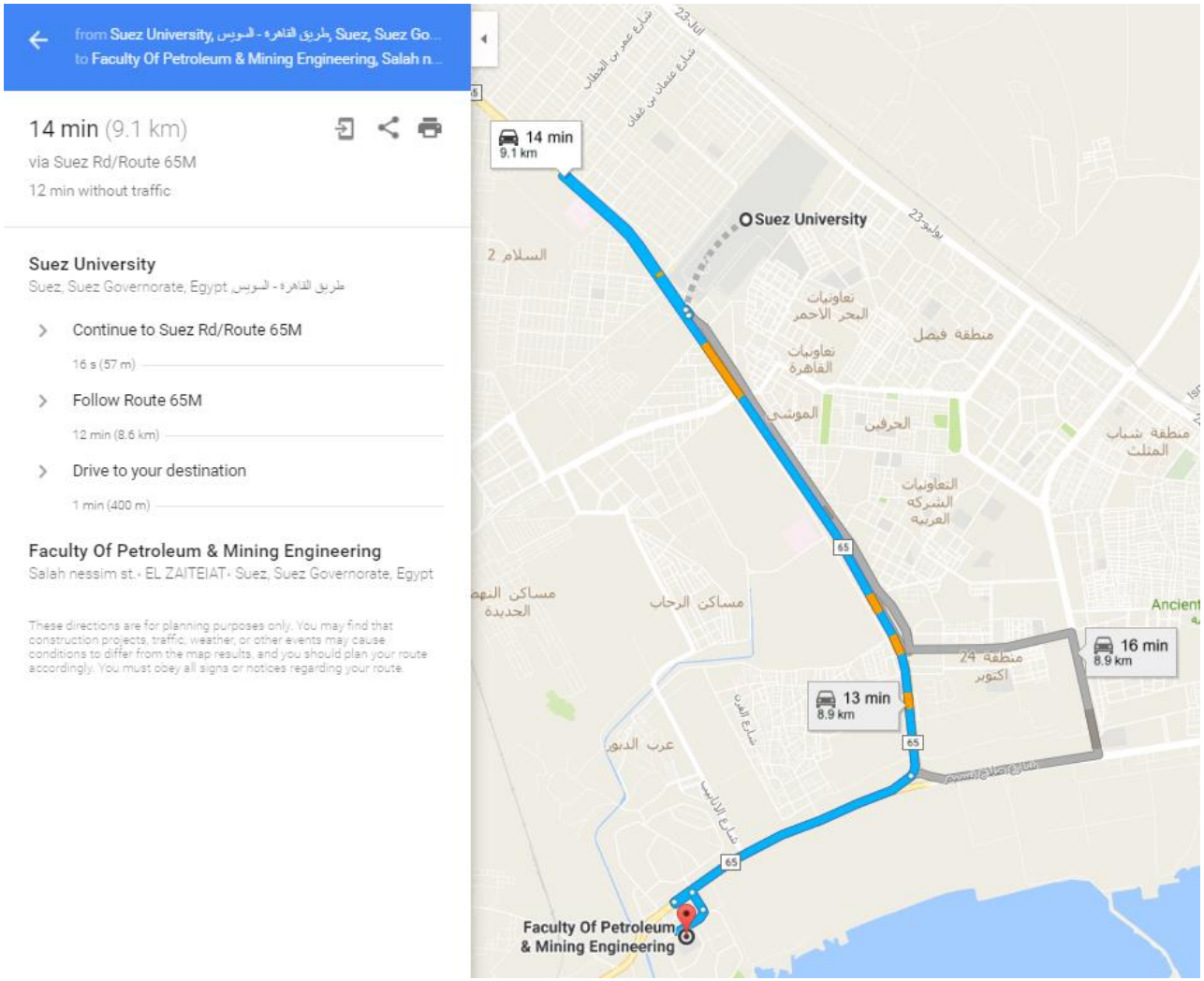
↓

PROGRAM
 Cake

What is an Algorithm?

Problem: Going from Suez University to Faculty of Petroleum

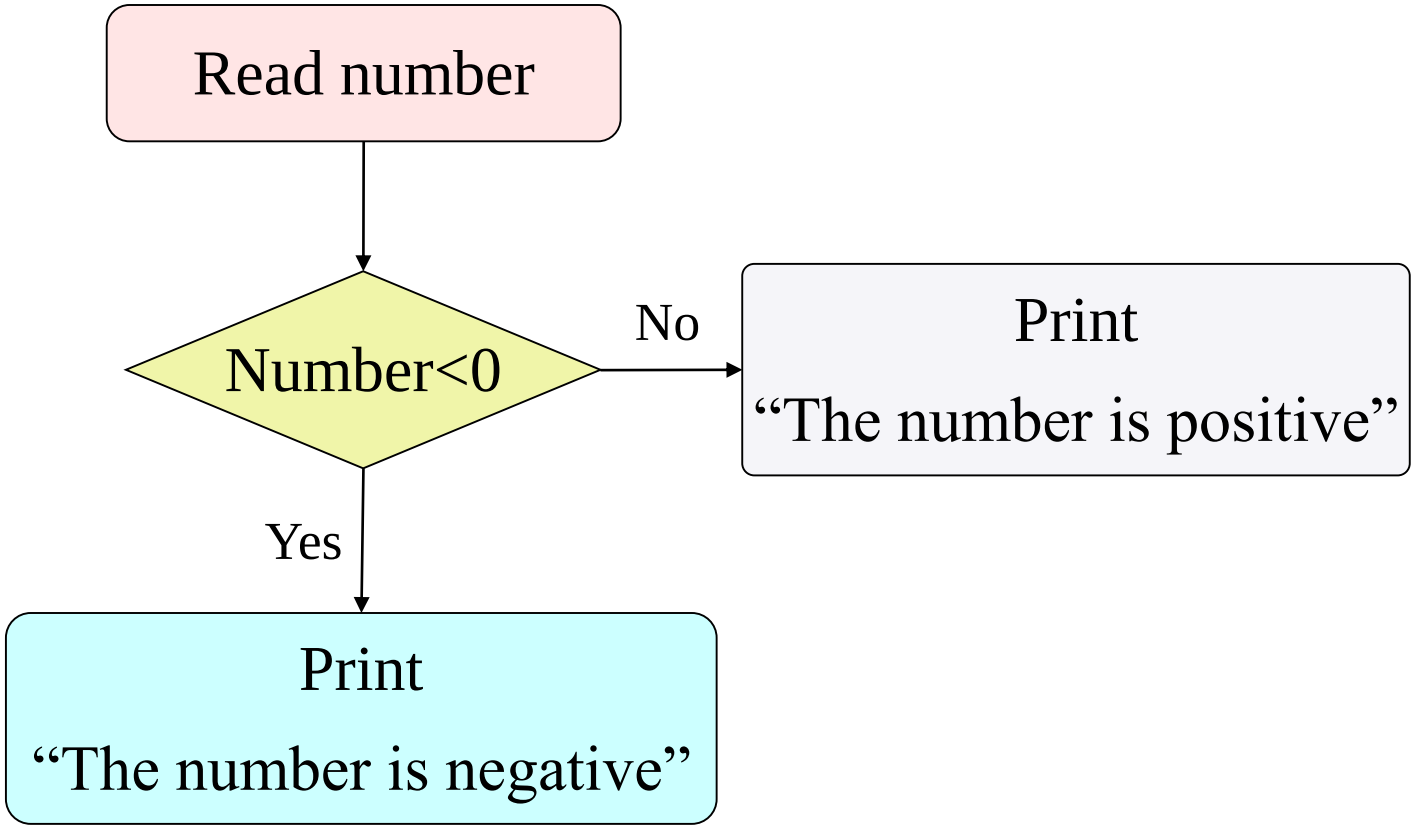
Solution:



What is an Algorithm?

Problem: Decide whether a number is positive or negative.

Solution:



What is an Algorithm?

Problem

Computer Scientist

Algorithm Description

- Natural Language
- Flow charts
- Formal Programming Language
- Pseudo-code

Computer Programmer

Algorithm Implementation

- Low-level Languages
  Assembly, machine code.
- High-level Languages
  Java, C/C++, Visual Basic, etc…

Solution

Outline

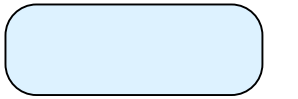
- What is an Algorithm?
- **Representing Algorithms**
- Pseudo-code
- Algorithm Implementation
- Types of Algorithms
- Sequential Algorithms
- Summary

Representing Algorithms

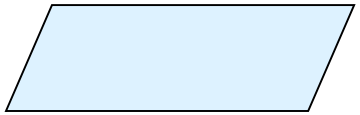
- Flow charts
- Natural Languages
- Formal Programming Language
- Pseudo-code

Representing Algorithms: Flow Charts

A flowchart is a graphical representation of an algorithm.

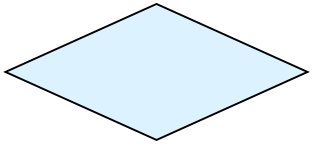


Start and End of an algorithm

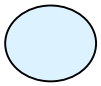


Input and Output

Process – Mathematical Operation



Decision – Logical Operation



Connector

Representing Algorithms: Flow Charts

Example-1: Calculate the area of a rectangle.

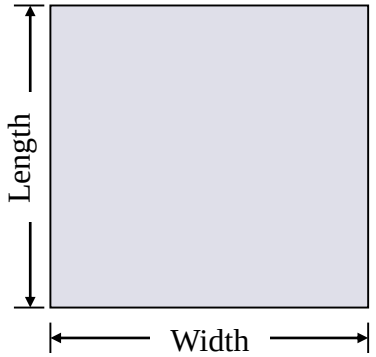
Solution:

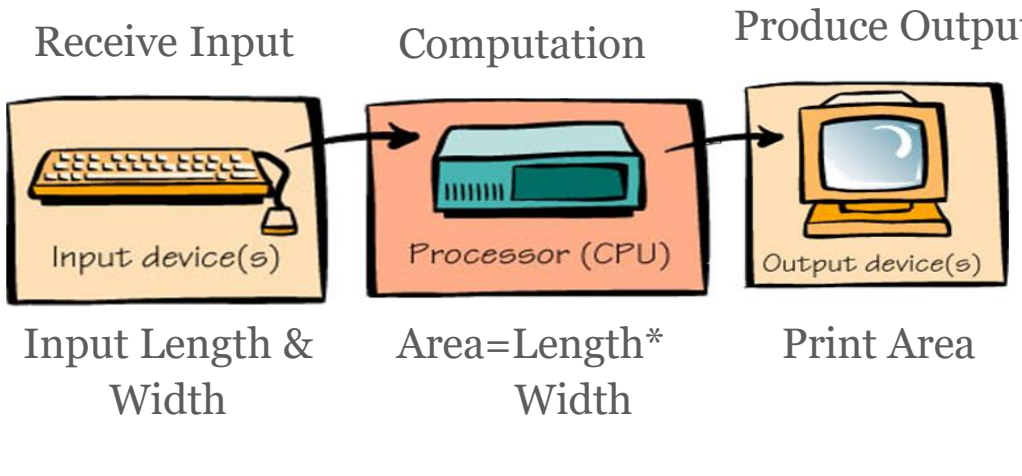
Inputs: Length & Width

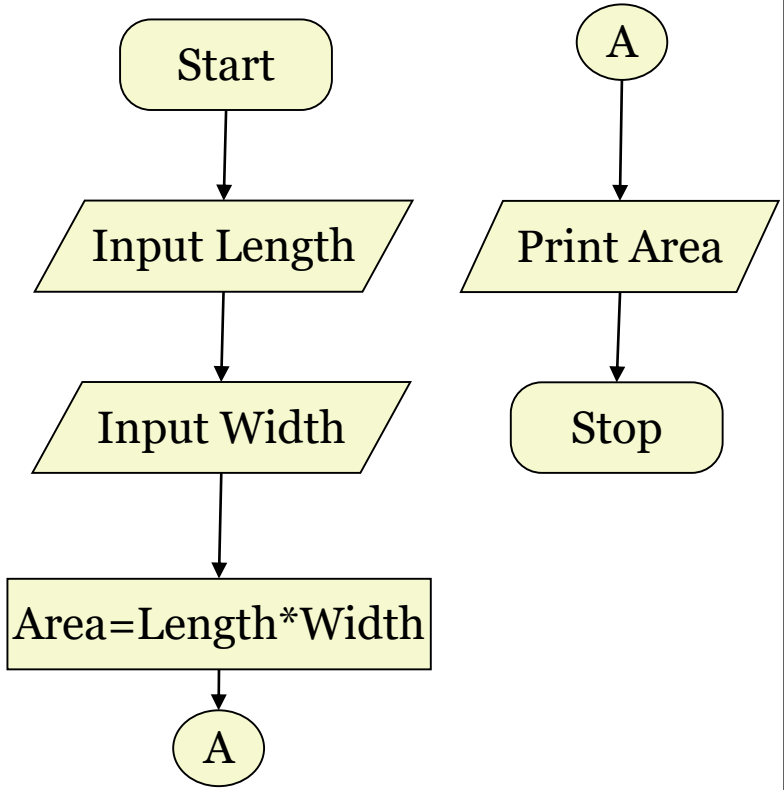
Outputs: Area

Expression:

Area=Length*Width







Representing Algorithms: Flow Charts

Example-2: Given a list of corporate income taxes paid by some Canadian industries in 2005. Calculate the average tax paid.

ID	Industry	M$
1	finance and insurance industries	11,028
2	Agriculture, fishing, hunting, trapping and support activities	495
3	Oil and gas extraction and support activities	3,749
4	Food and soft drink manufacturing	912
5	Clothing, textile and leather manufacturing	160

Source: Statistics Canada, http://www.statcan.ca/

Solution:

Inputs: 5 paid taxes $\Rightarrow T_1, T_2, T_3, T_4, T_5$

Outputs: Average tax

Expression: $Average=(T_1+T_2+T_3+T_4+T_5)/5$

Representing Algorithms: Flow Charts

Example-2:

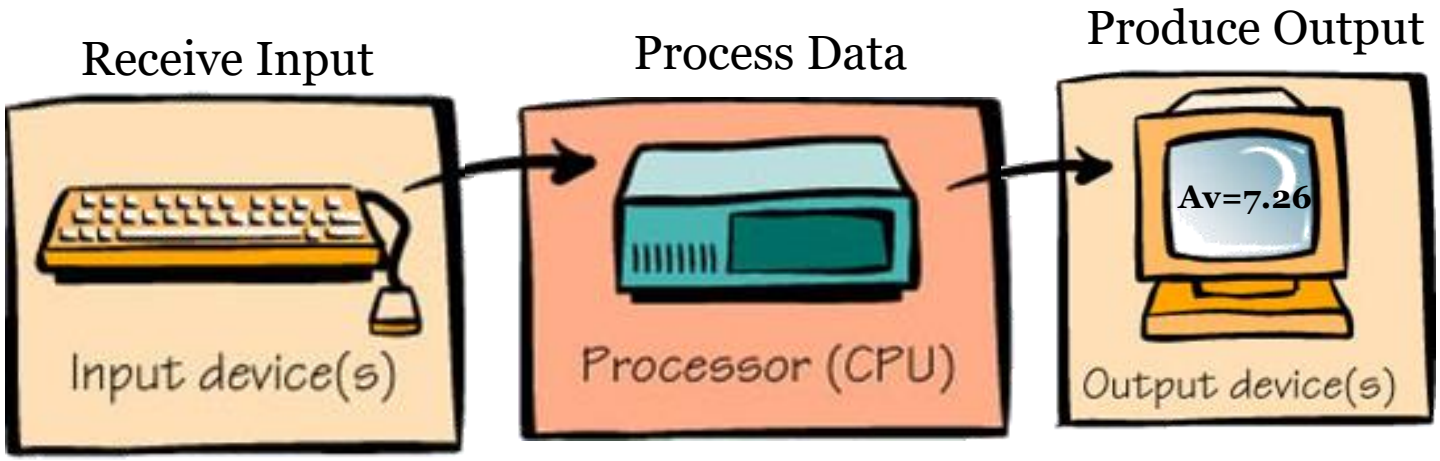
ID	Industry	M$
1	finance and insurance industries	11,028
2	Agriculture, fishing, hunting, trapping and support activities	495
3	Oil and gas extraction and support activities	3,749
4	Food and soft drink manufacturing	912
5	Clothing, textile and leather manufacturing	160

Inputs: 5 taxes $\Rightarrow T_1, T_2, T_3, T_4, T_5$

Outputs: Average paid tax

Expression:

$Average=(T_1+T_2+T_3+T_4+T_5)/5$



11028,495,3749,912,160

Av=(11028+495+3749+912+160)/5

Av=3268.8

Representing Algorithms: Flow Charts

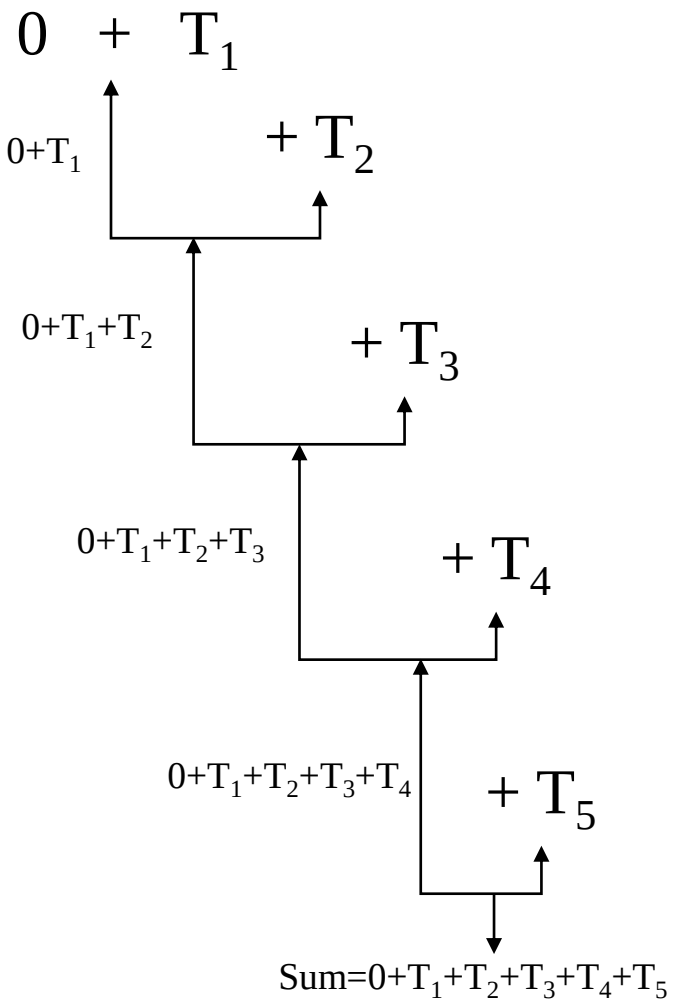
Example-2:

Inputs: 5 taxes $\Rightarrow T_1, T_2, T_3, T_4, T_5$

Expression: $Sum=T_1+T_2+T_3+T_4+T_5$,

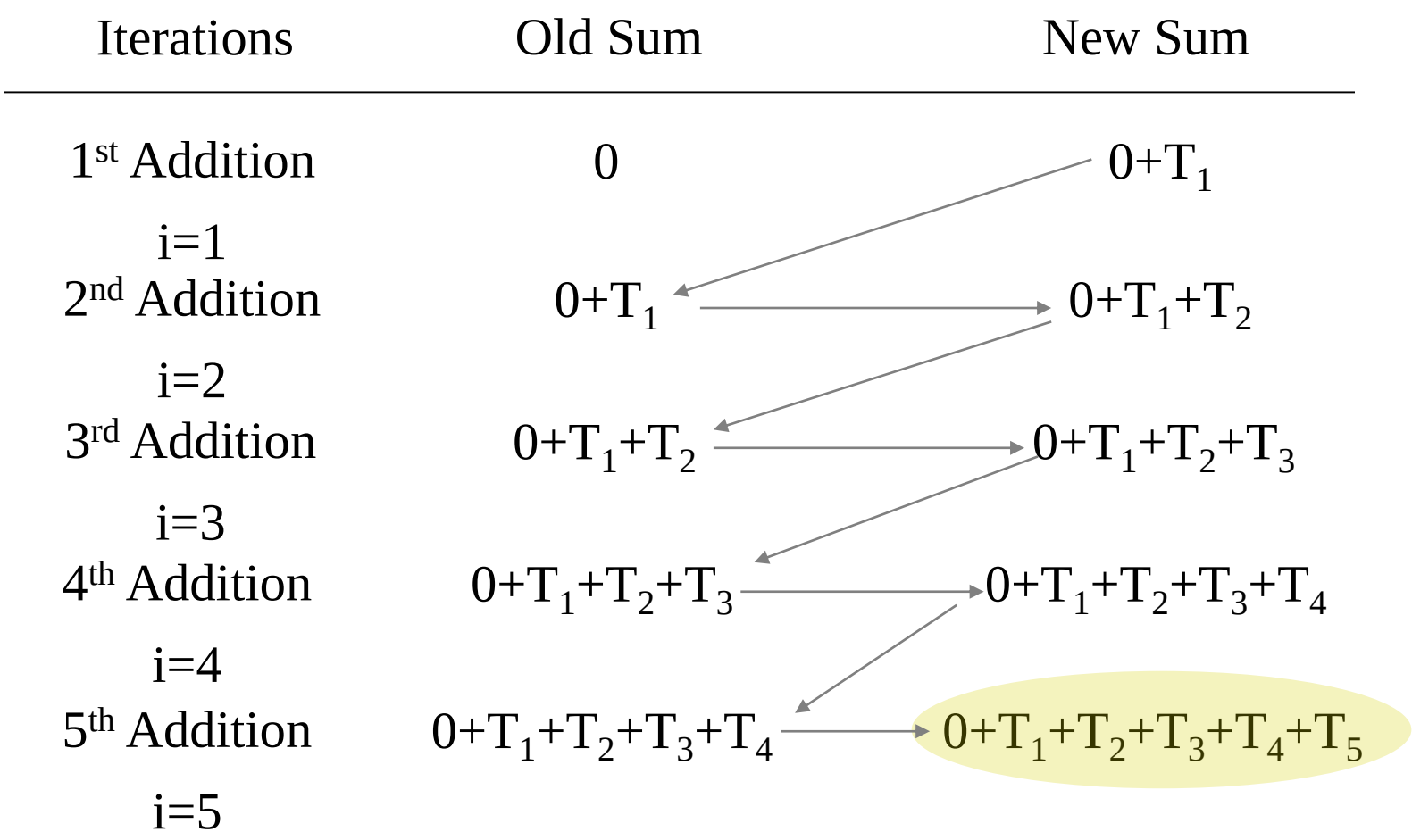
$$Average=Sum/5$$

Computer is just a **fast stupid machine** so to calculate the average, this machine must be provided by both data and algorithm. A computer with a single processor can perform one operation at a certain time. To calculate the sum of 5 taxes, computer will carry out five additions:

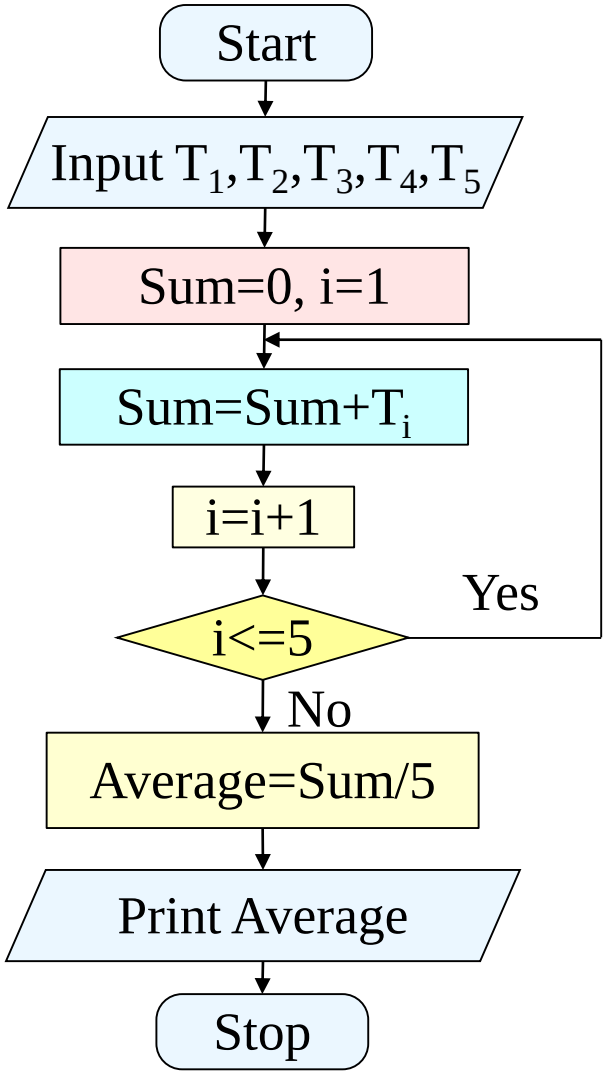


Representing Algorithms: Flow Charts

Example-2:



Representing Algorithms: Flow Charts



Iterations	Old Sum	New Sum
1st Addition i=1	0	$0+T_1$
2nd Addition i=2	$0+T_1$	$0+T_1+T_2$
3rd Addition i=3	$0+T_1+T_2$	$0+T_1+T_2+T_3$
4th Addition i=4	$0+T_1+T_2+T_3$	$0+T_1+T_2+T_3+T_4$
5th Addition i=5	$0+T_1+T_2+T_3+T_4$	$0+T_1+T_2+T_3+T_4+T_5$

Representing Algorithms: Natural Languages

Get the 5 paid taxes. Initially, set the value of the sum to 0 and the value of the counter i to 1. When these initializations have been completed, begin looping until the value of the variable i becomes greater than 5. First, add T_i to sum. Then add 1 to i and begin the loop all over again. Divide the sum by 5 to compute the average price.

Disadvantages

- too verbose
- unstructured
- too rich in interpretation and meaning (ambiguous)
- imprecise.

Representing Algorithms: Programming Language

```
import java.text.DecimalFormat;
import java.util.Scanner;

public class Average
{

   public static void main (String[] args)
   {
      int sum = 0, Tax, count = 0;
      double average;

      Scanner scan = new Scanner (System.in);

      System.out.print ("Enter paid tax (0 to quit): ");
      Tax = scan.nextInt();

      while (Tax != 0)  // 0 to terminate loop
      {
         count++;
         sum += Tax;

         System.out.print ("Enter paid tax (0 to quit): ");
         Tax = scan.nextInt();
      }

      System.out.println ();

      if (count == 0)
         System.out.println ("No values were entered.");
      else
      {
         average = (double)sum / count;

         DecimalFormat fmt = new DecimalFormat ("0.###");
         System.out.println ("The average paid tax is " + fmt.format(average) + "$");
      }
   }
}
```

Disadvantages

- Too many implementation details to worry about such as language syntax, grammar, punctuation, etc.
- Too rigid syntax.

Representing Algorithms: Pseudo-code

- Pseudo is a prefix of Greek origin. It means "false" or fake code.
- Not actually executed on computers.
- Allows us to think out a program before writing the code for it.

BEGIN

get values for paid taxes, T_1,T_2,T_3,T_4,T_5

set Sum to 0

set i to 1

while (i<=5){

set Sum to Sum+T_i

set i to i+1

}

set Average to Sum/5

print Average

END

Representing Algorithms: Pseudo-code

- English like constructs (or other natural language) but
- Modeled to look like statements in typical programming languages.

BEGIN

get values for paid taxes, T_1,T_2,T_3,T_4,T_5

set Sum to 0

set i to 1

while (i<=5){

set Sum to Sum+T_i

set i to i+1

}

set Average to Sum/5

print Average

END

Outline

- What is an Algorithm?
- Representing Algorithms
- **Pseudo-code**
- Algorithm Implementation
- Types of Algorithms
- Sequential Algorithms
- Summary

Pseudo-code

Pseudo-code (pseudo is a prefix of Greek origin means "**false**" or fake) is a description of a computer programming algorithm that uses the structural conventions of programming languages, but omits language-specific syntax.

It can also refer to a **high level "language"** whose aim is to generalize the logic and program flow of a computer program.

Pseudo-code

- **Input**

  get values for paid taxes, T_1, T_2, T_3, T_4, T_5

  or

  get T_1, T_2, T_3, T_4, T_5

- **Output**

  print Average

Pseudo-code

- **Variables**

  A variable is a named storage that holds the data assigned to it until a new value is assigned or the program is finished

  Examples:

  set the value of Tax to 3.5 or **set Tax to 3.5**

  The variable Tax holds the value 3.5 after its execution.

  set the value of Tax to Tax+1

  Same as: add 1 to the value of Tax or increment Tax (Tax is now 4.5)

Pseudo-code

- **Conditional Statements**

if statement:

```
if <condition> then
operations for the then-part
else
operations for the else-part
endif
```

```
if (my_income>=10,000) then
        set Tax to 0.10
else
        set Tax to 0.05
endif
```

Conditional operation is a control operation that allow us to alter the normal sequential flow of control in an algorithm. Conditional statements are the “question-asking” operations of an algorithm.

Pseudo-code

- **Conditional Statements**

Nested if statement:

```
if <first condition> then
        // do first thing
   else if <second condition> then
        // do second thing
   else
        // do something else
   endif
endif
```

```
if (my_income > 100,000) then
  set Tax to 0.17
  else if (my_income =100,000) then
        set Tax to 0.12
   else   set Tax to 0.08
   endif
endif
```

Pseudo-code

- **Looping (Iterations)**

while statement:

```
while <condition>
{
   // do some operations
} // end while
```

```
while (years <= 5)
{
    set balance to initial_balance(1+0.05*)^years
    set years to years+1
}
```

This means repeat the operations as long as the condition is true

Pseudo-code

- **Examples**

Exmple-1: Write an algorithm in pseudo-code to determine the flying time between two cities given the distance between them and the average speed of the airplane.

Time=Distance/Speed

Exmple-2: Write an algorithm in pseudo-code to convert the temperature from Celsius to Fahrenheit.

$$T_f=(9/5)*T_c+32$$

Pseudo-code

- **Examples**

Exmple-3: Write an algorithm in pseudo-code that gets the radius of a circle as input and calculates the area and circumference.

$$\mathbf{Area=3.14*r^2}$$

$$\mathbf{Circumference=2*3.14*r}$$

Exmple-4: If you invested 10000 pounds with 5% compounded annually. Write an algorithm in pseudo-code that prints out your balance after one year.

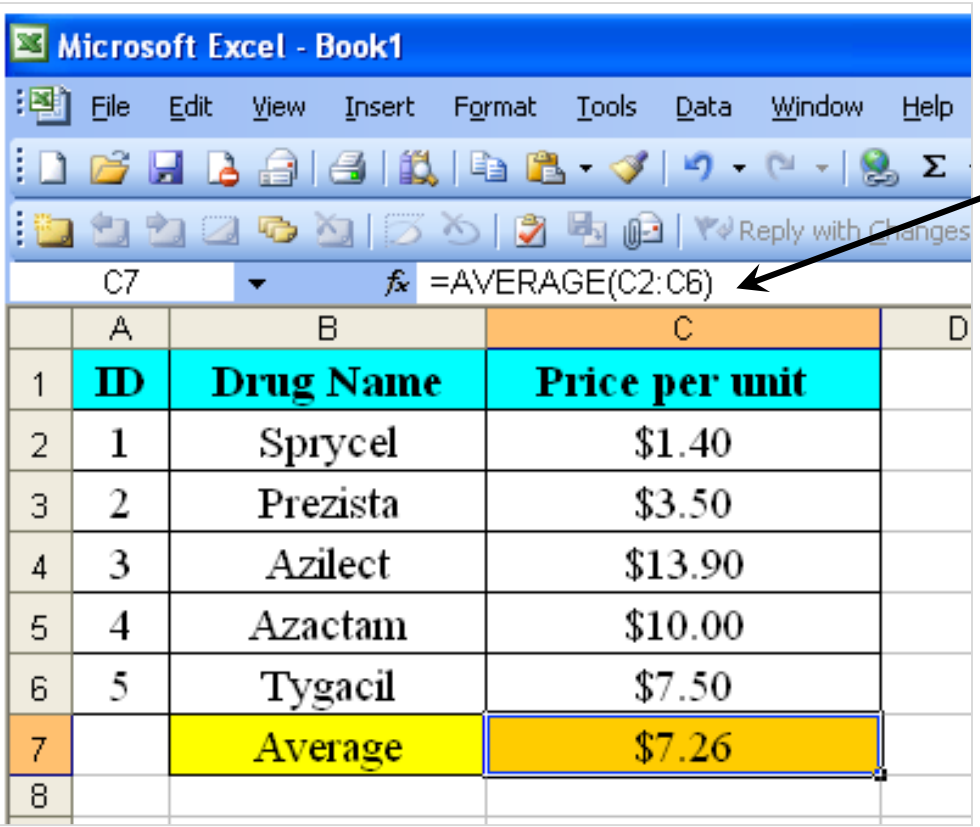
Exmple-5: Write an algorithm in pseudo-code that calculates the net salary of an employee after deducting the 8% of taxes from a gross salary of 2000 L.E.

Outline

- What is an Algorithm?
- Representing Algorithms
- Pseudo-code
- **Algorithm Implementation**
- Types of Algorithms
- Sequential Algorithms
- Summary

Algorithm Implementation

Problem: Given a list of medical drugs and their prices. Calculate the average price.

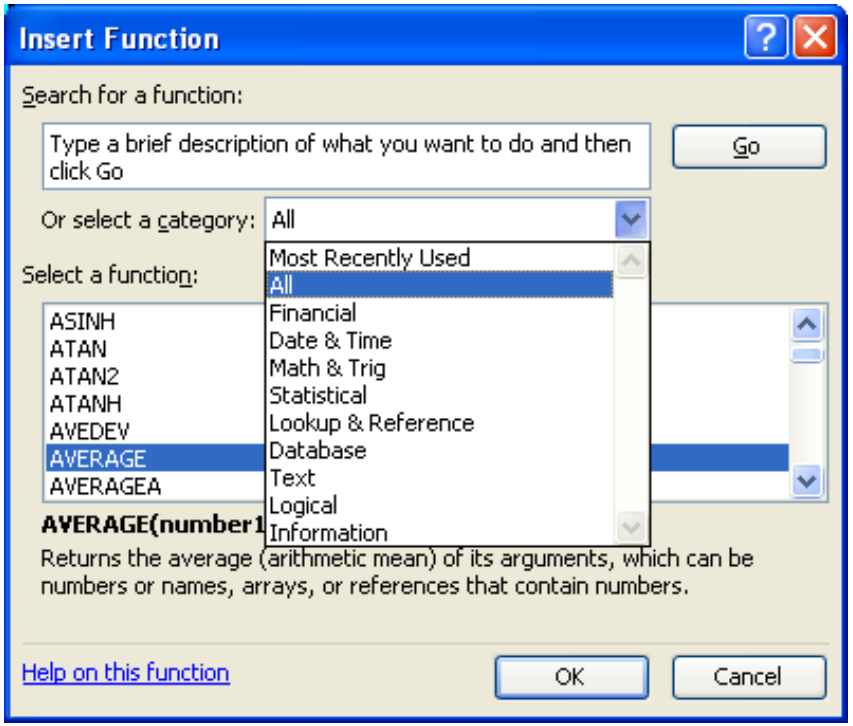


	A	B	C	D
1	**ID**	**Drug Name**	**Price per unit**	
2	1	Sprycel	$1.40	
3	2	Prezista	$3.50	
4	3	Azilect	$13.90	
5	4	Azactam	$10.00	
6	5	Tygacil	$7.50	
7		Average	$7.26	
8				

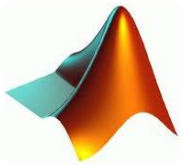
MS Excel has a built-in function called **AVERAGE**, which can be used to calculate the average price in very abstracted way.

Algorithm Implementation

Problem: Sometimes we need to implement an algorithm with mathematical and/or logical operations not supported by Excel or other software packages.



Solution: start to create your own program using a high-level programming language.



MATLAB

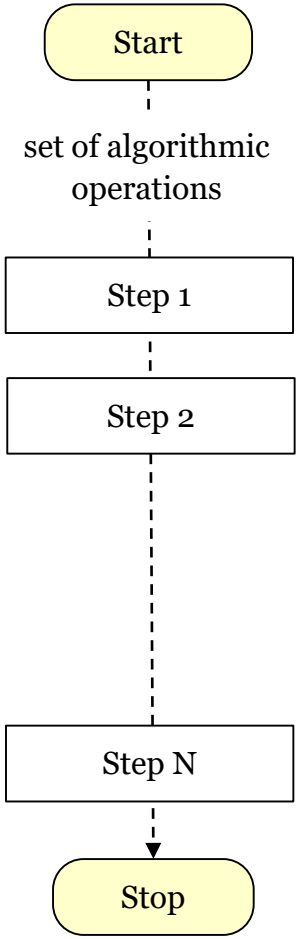
GNU Octave

Outline

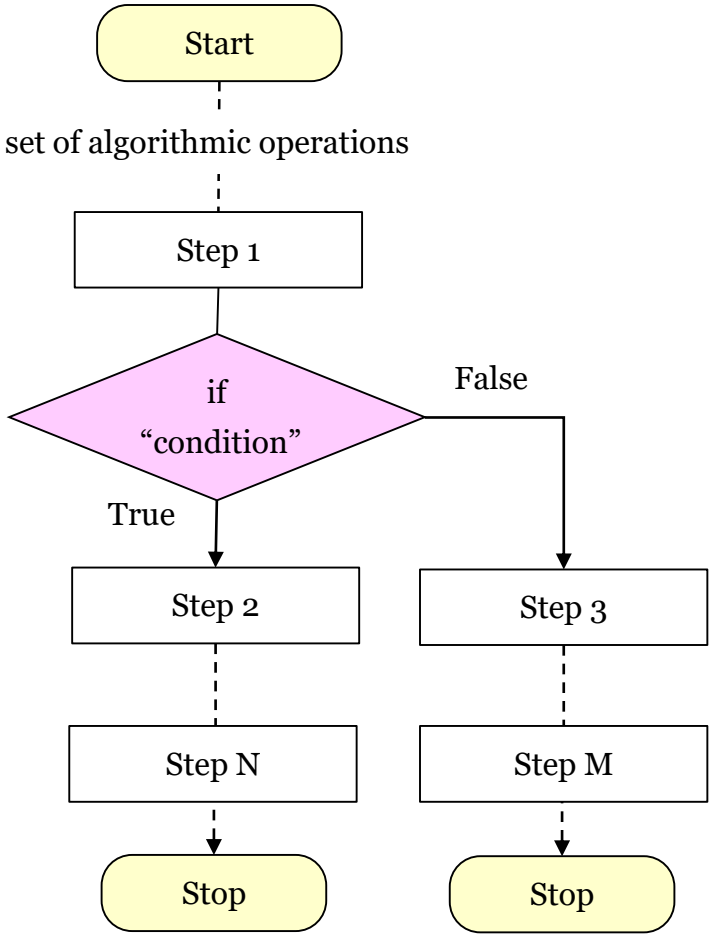
- What is an Algorithm?
- Representing Algorithms
- Pseudo-code
- Algorithm Implementation
- **Types of Algorithms**
- Sequential Algorithms
- Summary

Types of Algorithms

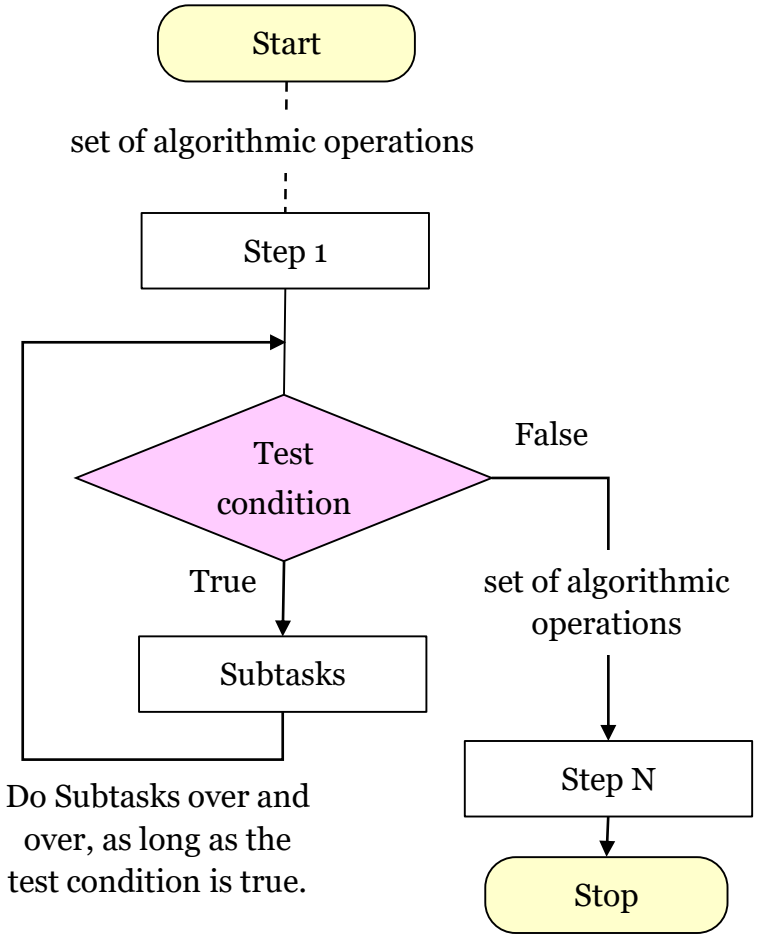
Sequential Algorithm



Conditional Algorithm



Iterative Algorithm



Types of Algorithms

Example: Convert temperature from Celsius to Fahrenheit using the following formula in the following cases:

$$T_f = 9/5 * T_c + 32$$

a. Convert only one value from Celsius to Fahrenheit,

b. Convert the Celsius degree to Fahrenheit if and only if the Celsius degree is less than certain value, let's say (90°C)

c. Convert range of Celsius degrees from (1°C) to (100°C)

Types of Algorithms

Example: Convert temperature from Celsius to Fahrenheit using the following formula in the following cases:

$$T_f=9/5*T_c+32$$

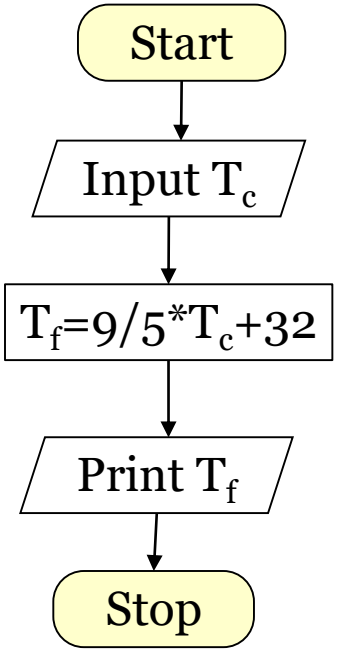
a. Convert only one value from Celsius to Fahrenheit.

Input: T_c

Output: T_f

Expression: $T_f=9/5*T_c+32$

Sequential Algorithm



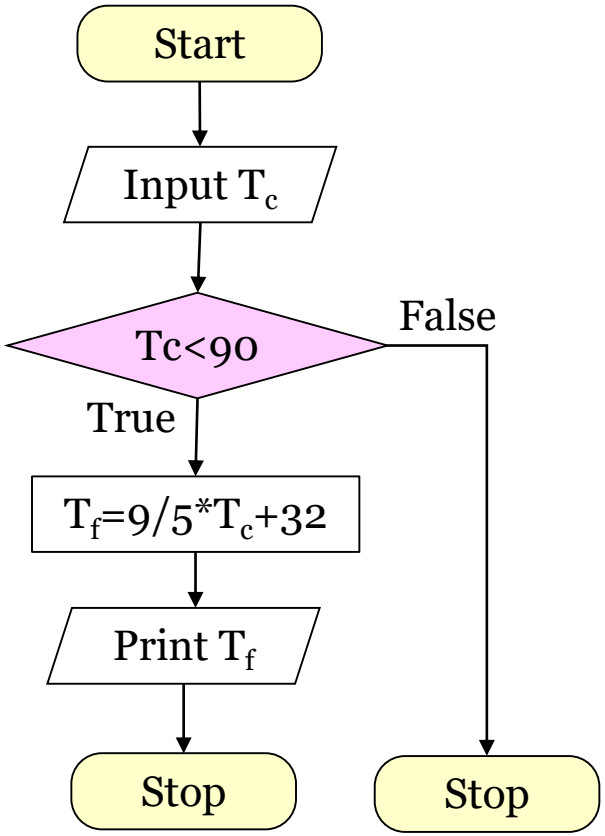
Types of Algorithms

Example: Convert temperature from Celsius to Fahrenheit using the following formula in the following cases:

$$T_f = 9/5 * T_c + 32$$

b. Convert the Celsius degree to Fahrenheit **if and only if** the Celsius degree is less than certain value, let's say (90°C)

Conditional Algorithm



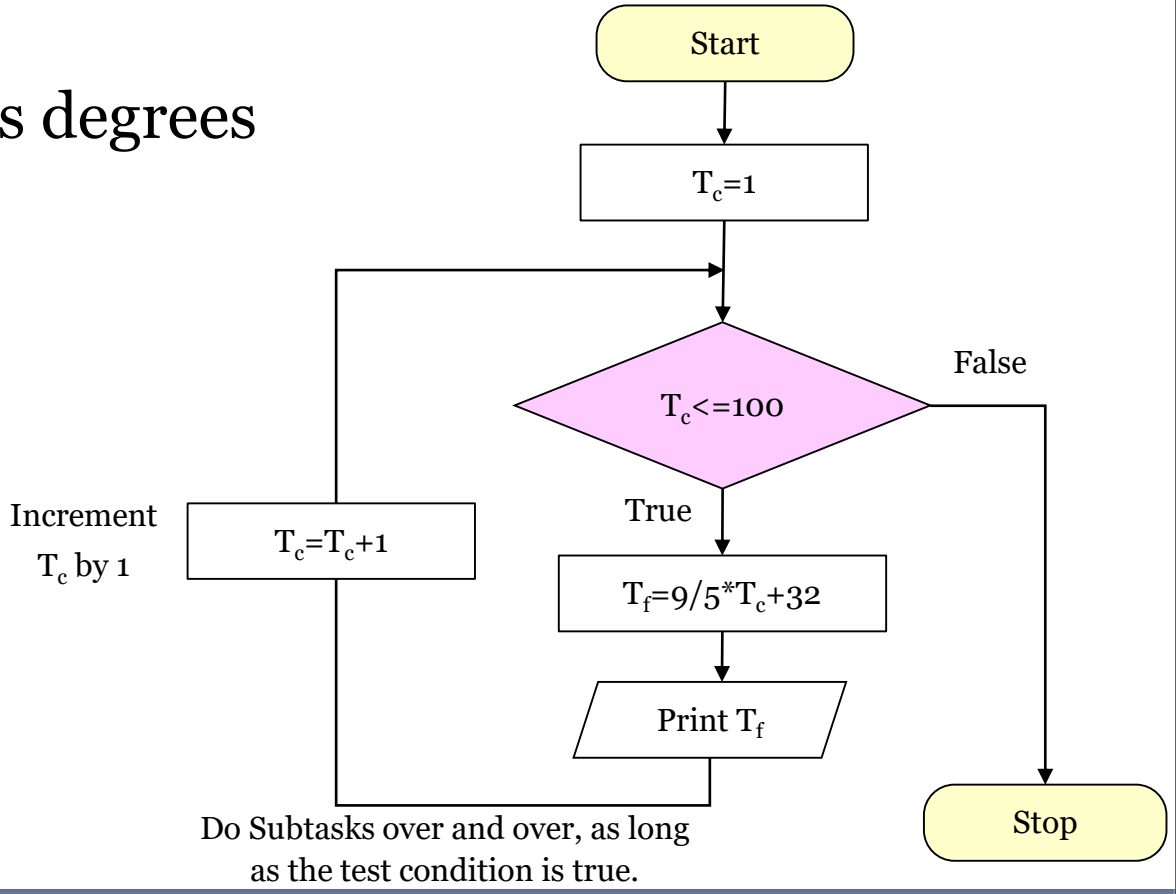
Types of Algorithms

Example: Convert temperature from Celsius to Fahrenheit using the following formula in the following cases:

$$T_f = 9/5 * T_c + 32$$

c. Convert range of Celsius degrees from (1^oC) to (100^oC)

Iterative Algorithm

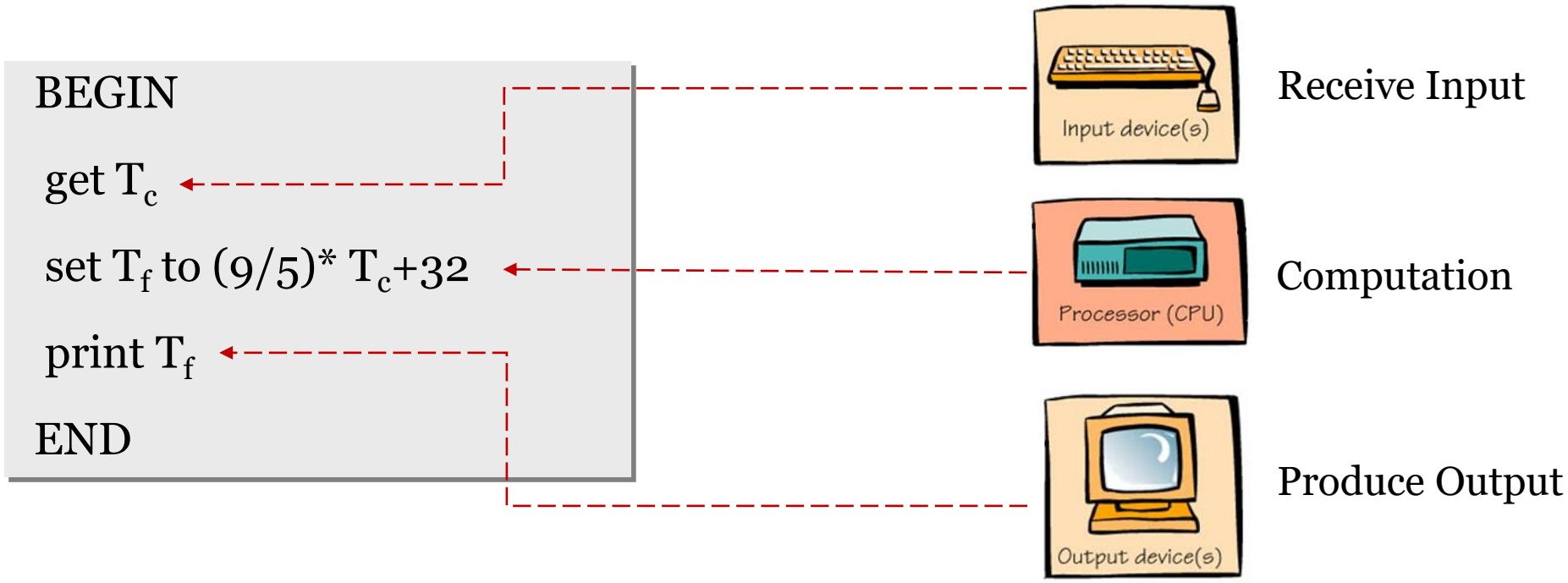


Outline

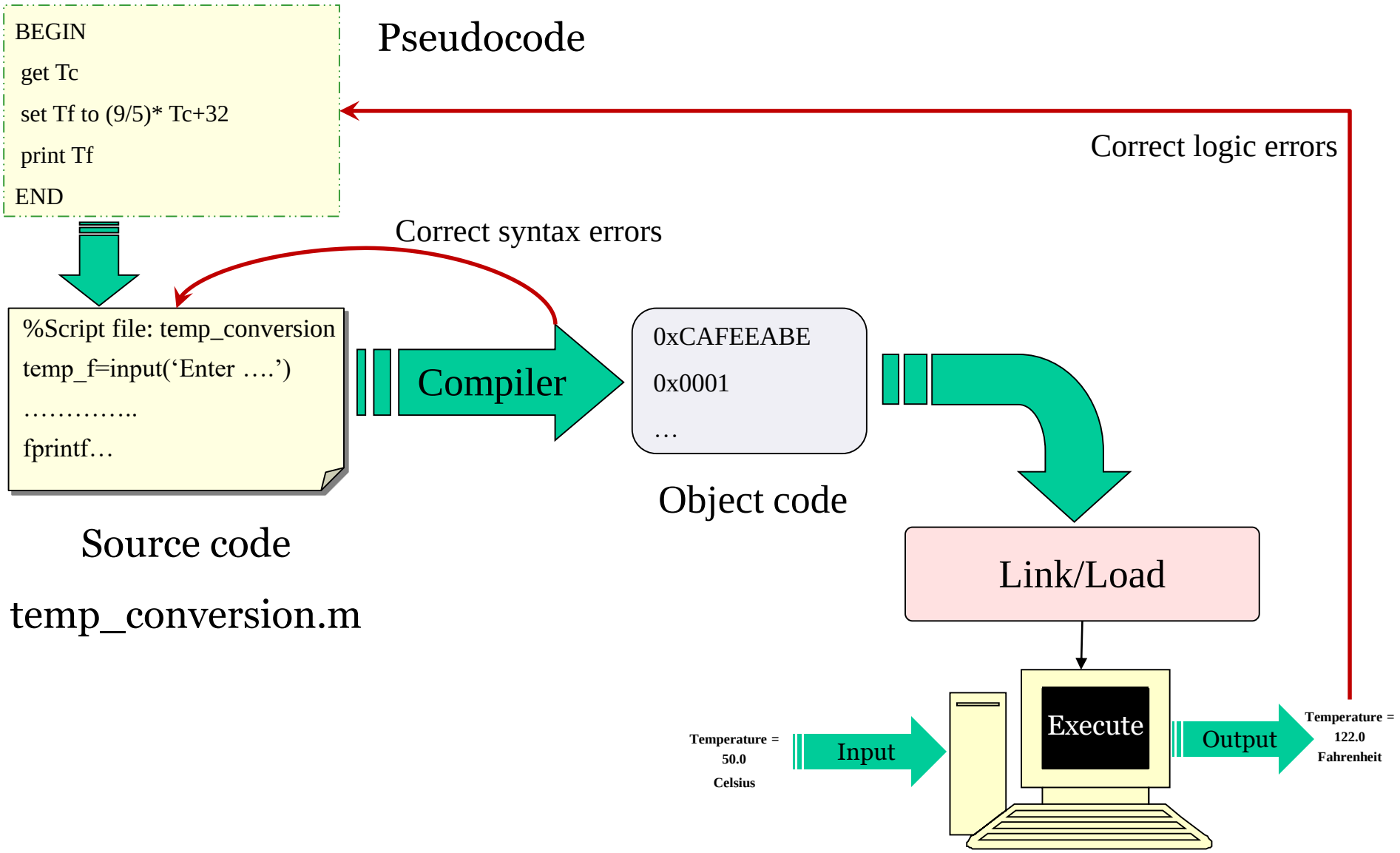
- What is an Algorithm?
- Representing Algorithms
- Pseudo-code
- Algorithm Implementation
- Types of Algorithms
- **Sequential Algorithms**
- Summary

Sequential Algorithms

Sequential algorithm is formed by a list of operations (or steps) arranged in a "linear" fashion, such that the order of these steps is well defined and significant.



Sequential Algorithms



Sequential Algorithms

- **Interactive Programming**

```
>>
>> temp_c=50

temp_c =

     50

>> temp_f=(9/5)*(temp_c)+32

temp_f =

    122
```

Sequential Algorithms

- **Saving and Restoring Matlab Information**

It is good engineering practice to keep records of calculations.

These records can be used for several purposes, including:

◊To **revise the calculations** at a later time.

◊To **prepare a report** on the project.

Matlab provides the following methods for saving information from a workspace session:

◊Saving the session output with the **diary command** and

◊Saving and loading the variables with the **save and load command**.

Sequential Algorithms

- **Diary Command**

  The diary commands allows you to record all of the input and displayed output from a Matlab interactive workspace session.

  The commands include:

  - **diary file**: Saves all text from the Matlab session, except for the prompts (>>), as text in file, written to the present working directory. If file is not specified, the information is written to the file named **diary**.
  - **diary off**: Suspends diary operation.
  - **diary on**: Turns diary operation back on.
  - **diary**: Toggles diary state.

Sequential Algorithms

- **Diary Command**

```
>> diary t_conversion
>> temp_c=50

temp_c =

     50

>> temp_f=(9/5)*(temp_c)+32

temp_f =

    122

>> diary off
```

```
>> type t_conversion

temp_c=50

temp_c =

     50

temp_f=(9/5)*(temp_c)+32

temp_f =

    122

diary off
```

Note that this is nearly the same as the display in the command window, with the exception that the Matlab prompt (>>) is not included.

Sequential Algorithms

- **Saving and Retrieving Matlab Variables**
  - ◇ Storing and Loading Workspace Values

Command	Description
save	Stores workspace values (variable names, sizes, and values), in the binary file matlab.mat in the present working directory **save data** stores all workspace values in the file **data.mat**
save **data_1** x y	Stores only the variables x and y in the file **data_1.mat**
load **data_1**	Loads the values of the workspace values previously stored in the file **data_1.mat**

Sequential Algorithms

- **Saving and Retrieving Matlab Variables**
  - ◇ Exporting and Importing Data

    There are also situations in which you wish to export Matlab data to be operated upon with other programs, or to import data created by other programs. This must be done with text files written with save or read with load.

    To write a text file **data1.dat** in the current working directory containing values of Matlab variables in long e format:

```
save data1.dat -ascii
```

Sequential Algorithms

- **Script M-files**

Editor - C:\MATLAB7\work\temp_conversion.m

File Edit Text Cell Tools Debug Desktop Window Help

Stack: Base

```
1   %Script file: temp_conversion
2   %
3   % Purpose:
4   % To convert an input temperature from degrees Celsius
5   % to an output temperature in Fahrenheit
6   %
7   %Record of revisions:
8   %Date       Programmer      Description of change
9   %=====      ==========      =====================
10  %02/05/2011 Alaa Khamis     Orignial Code
11  %
12  %Define variables:
13  % temp_c: Temperature in Celsius
14  % temp_f: Temperature in Fahrenheit
15
16  %Input value in Celsius
17  temp_c=50;
18
19  %Convert to Fahrenheit
20  temp_f=(9/5)*(temp_c)+32;
21
22  %Write the results
23  fprintf('%6.2f degrees Celsius %6.2f Fahrenheit.\n',temp_c,temp_f);
24
```

script Ln 1 Col 1 OVR

```
>>
 50.00 degrees Celsius 122.00 Fahrenheit.
```

Sequential Algorithms

- **Getting the inputs from the user**

The **input command** is used for user input of data in a script and it is used in the form of an assignment statement.

For example:

```
temp_c=input('Enter the temperature in degrees Celsius:');
```

- ◇ When this command is executed, the text string **Enter the temperature in degrees Celsius:** is displayed as a user prompt in the command window.
- ◇ The user then types in data value, which is assigned to the variable **temp_c**.

Sequential Algorithms

- **Getting the inputs from the user**

```
%Script file: temp_conversion2
%
% Purpose:
% To convert an input temperature from degrees Celsius
% to an output temperature in Fahrenheit
%
%Record of revisions:
%Date        Programmer       Description of change
%=====       ==========       =====================
%02/05/2011 Alaa Khamis      Orignial Code
%
%Define variables:
% temp_c: Temperature in Celsius
% temp_f: Temperature in Fahrenheit

%Prompt the user for the input temperature
temp_c=input('Enter the temperature in degrees Celsius:');

%Convert to Fahrenheit
temp_f=(9/5)*(temp_c)+32;

%Write the results
fprintf('%6.2f degrees Celsius %6.2f Fahrenheit.\n',temp_c,temp_f);
```

```
>> temp_conversion2
Enter the temperature in degrees Celsius:50.0
 50.00 degrees Celsius 122.00 Fahrenheit.
```

Sequential Algorithms

- **Example-2: Flying Time**

Write an algorithm in pseudo-code to determine the flying time between two cities given the distance between them and the average speed of the airplane. Convert this pseudo-code into a Matlab program to calculate the flying time between Cairo and Aswan (distance is 850 Km) if the average speed of the airplane is 550 km/hr.

Inputs: Distance "distance"
and Average Speed
"speed"

Outputs: Flying Time
"time"

Pseudo-code:

```
BEGIN
get distance, speed
set time to distance/speed
print time
END
```

Sequential Algorithms

- **Example-2: Flying Time**

Matlab Program:

```
%Script file: flying_time.m
% Purpose:
% To determine the flying time between two cities given the distance
% between them and the average speed of the airplane
%Record of revisions:
%Date       Programmer      Description of change
%=====      ==========      =====================
%03/05/2011 Alaa Khamis     Orignial Code

%Define variables:
% distance: Distance between two cities (Km)
% speed:    Average speed of the airplane (Km/hr)
% time:     Flying time betwee the two cities

%Prompt the user for the distance
distance=input('Enter the distance between the two cities (Km):');

%Prompt the user for the distance
speed=input('Enter the average speed of the airplane (Km/hr):');

%Calculate the flying time
time=distance/speed;

%Write the results
fprintf('The flying time between the two cities is  %6.2f Hours.\n',time);
```

Sequential Algorithms

- **Example-2: Flying Time**

Program run:

```
Enter the distance between the two cities (Km):850
Enter the average speed of the airplane (Km/hr):550
The flying time between the two cities is    1.55 Hours.
```

Sequential Algorithms

- **Example-3: Quadratic Roots**

  Write an algorithm in pseudo-code that computes the roots of the quadratic equation.

$$as^2 + bs + c = 0$$

  Implement this algorithm using MATLAB.

  1. Prompting for input of coefficients a, b, and c;
  2. Format the display of the computed roots s1 and s2;

Sequential Algorithms

- **Example-3: Quadratic Roots**

$$as^2 + bs + c = 0$$

Inputs: a, b and c

Outputs: roots s

Expression:

$$s = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Algorithm in Pseudo-code:

BEGIN

get a,b and c

set $s = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

print s

END

Sequential Algorithms

- **Example-3: Quadratic Roots**

Implementation in Matlab:

```
%Script rqroots.m

% rqroots: quadratic root finding script
format compact;

% prompt for coefficient input
a = input('Enter quadratic coefficient a: ');
b = input('Enter quadratic coefficient b: ');
c = input('Enter quadratic coefficient c: ');
disp('')

% compute intermediate values x & y
x = -b/(2*a);
y = sqrt(b^2-4*a*c)/(2*a);

% compute roots
s1 = x+y;

% display roots
disp('Value of first quadratic root: '),disp(s1);
s2 = x-y;
disp('Value of second quadratic root: '),disp(s2);
```

```
Enter quadratic coefficient a: 1
Enter quadratic coefficient b: -5
Enter quadratic coefficient c: 3
Value of first quadratic root:
    4.3028
Value of second quadratic root:
    0.6972
```

Sequential Algorithms

- **Example-4: Carbon 14 Dating**

  A radioactive isotope of an element is a form of the element that is not stable. Instead, it spontaneously decays into another element over a period of time. Radioactive decay is an exponential process. If Q_o is the initial quantity of a radioactive substance at time t=0, then the amount of that substance which will be present at any time t in the future is given by:

$$Q(t) = Q_o e^{-\lambda t}$$

  Where λ is the radioactive decay constant.

Sequential Algorithms

- **Example-4: Carbon 14 Dating**

  Because radioactive decay occurs at a known rate, it can be used as a clock to measure the time that has elapsed since the decay started. If we know the initial amount of radioactive material Q_o present in a sample an the amount of material Q left at the current time t, we can solve for t in the previous equation to determine how long the decay has been going on. The resulting equation is:

$$t_{decay} = -\frac{1}{\lambda} \log_e \frac{Q}{Q_o}$$

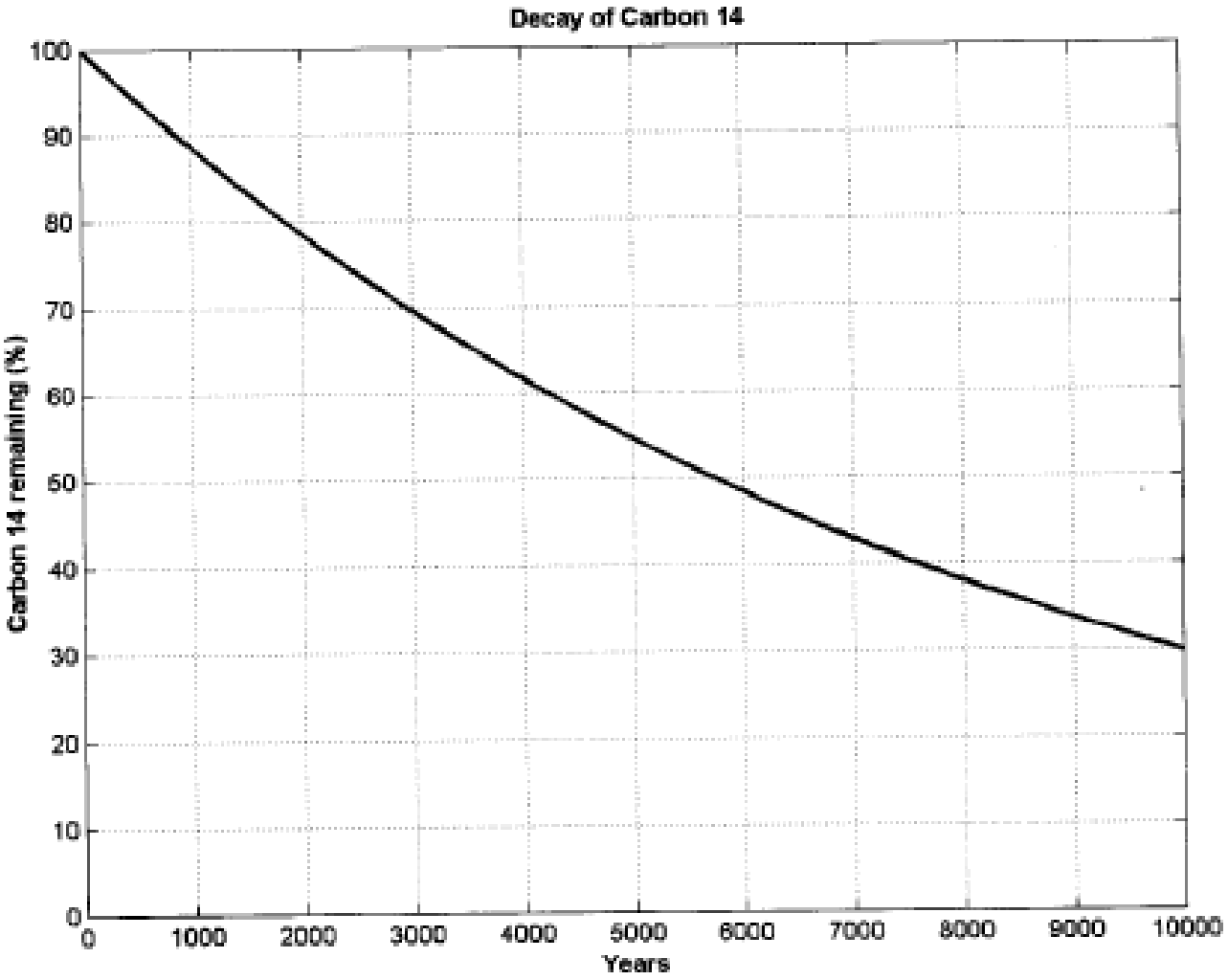
Sequential Algorithms

- **Example-4: Carbon 14 Dating**

  Archaeologists use a radioactive clock based on carbon 14 to determine the time that has passed since a once-living thing died. Carbon 14 is continually taken into the body while a plant or animal is living, so the amount of it present in the body at the time of death is assumed to be known. The decay rate of carbon 14 is well known to be 0.00012097/year. Therefore, the amount of carbon 14 remaining now can be accurately measured, and the previous equation can be used to determine how long ago the living thing died.

Sequential Algorithms

- **Example-4: Carbon 14 Dating**



Sequential Algorithms

- **Example-4: Carbon 14 Dating**

  Write an algorithm in pseudo-code that reads the percentage of carbon 14 remaining in a sample, calculates the age of the sample from it, and prints out the result with proper units.

Inputs: Q/Q_o

Outputs: age in years

Expression:

$$t_{decay} = -\frac{1}{\lambda} \log_e \frac{Q}{Q_o}$$

Algorithm in Pseudo-code:

BEGIN

get Q/Q_o

set $t_{decay} = -\frac{1}{\lambda} \log_e \frac{Q}{Q_o}$

print t_{decay}

END

Sequential Algorithms

- **Example-4: Carbon 14 Dating**

Implementation in Matlab:

```
%Script file: c14_date.m

%Purpose:
% To calculate teh age of an organic sample from the
% percentaeg of the orginial carbon 14 remaining in the sample

%Define variables
% age: the age of the sample in years
% lambda: the radioactive decay constant for carbon 14, in units 1/years.
% percent: the percentage of the carbon 14 remaining at time of the measurement
% ratio: the ratio of the carbon 14 remaining at time of the measurement
%        to the time of the orignal amount of carbon 14.

%Set the decay constant for Carbon-14
lambda=0.00012097;

%Prompt the user for the percntage of C-14 remaining.
percent=input('Enter the percntage of Carbon-14 remaining:\n');

%Perform calculations
ratio=percent/100;    % Convert to fractional ratio
age=(-1.0/lambda)*log(ratio);  % Get age in years

%tell the user about the age of the sample.
string=['The age of the sample is '   num2str(age)   ' years'];
disp(string);
```

Sequential Algorithms

- **Example-4: Carbon 14 Dating**

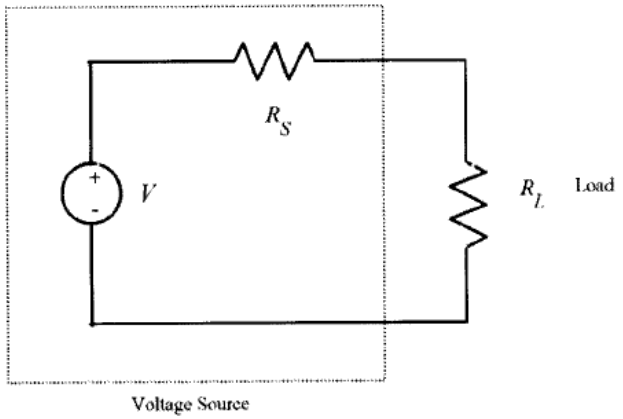
  Implementation in Matlab:

```
Enter the percntage of Carbon-14 remaining:
50
The age of the sample is 5729.9097 years
```

Sequential Algorithms

- **Example-5: Maximum Power Transfer**

The figure shows a voltage source V=120 V with an internal resistance R_s of 50 Ω supplying a load of resistance R_L.



Write an algorithm in pseudo-code that calculates how much power be supplied by the source to the load?

Implement this algorithm using Matlab and also, plot the power supplied to the load as a function of the load resistance R_L. What is the value of R_L that results in the maximum power being supplied by the source to the load.

Sequential Algorithms

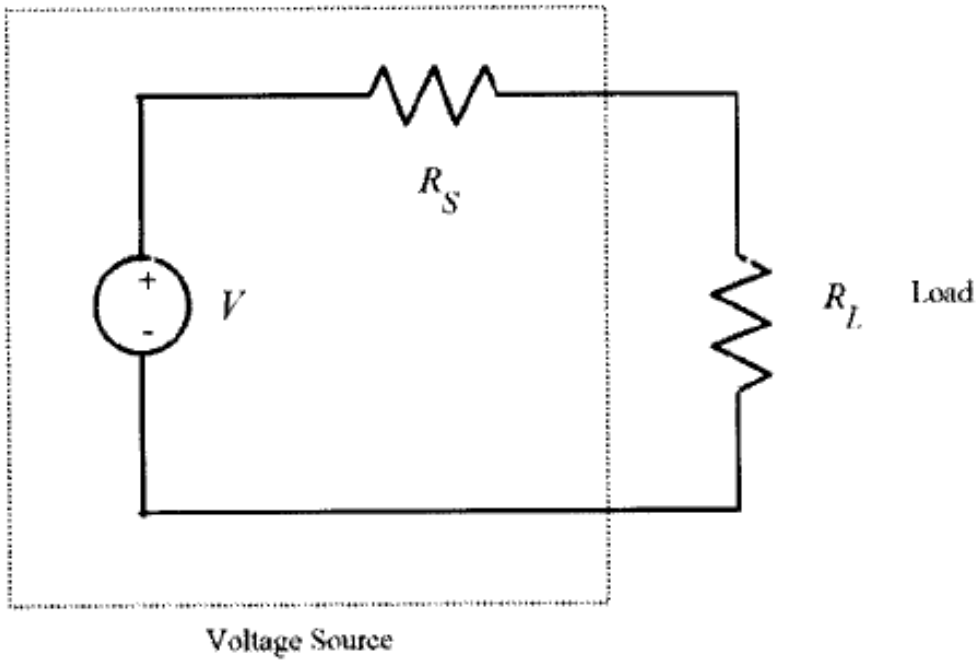
- **Example-5: Maximum Power Transfer**

In this problem, we need to vary the load resistance R_L and compute the power supplied to the load at each value of R_L. The power supplied to the load resistance is given by:

$$P_L = I^2 R_L$$

where I is the current supplied to the load:

$$I = \frac{V}{R_{tot}} = \frac{V}{R_S + R_L}$$



Sequential Algorithms

- **Example-5: Maximum Power Transfer**

Inputs: R_L

Outputs: P_L

Expression:

$$I = \frac{V}{R_{tot}} = \frac{V}{R_S + R_L}$$

$$P_L = I^2 R_L$$

Algorithm in Pseudo-code:

BEGIN

get R_L

set $I = \frac{V}{R_{tot}} = \frac{V}{R_S + R_L}$

set $P_L = I^2 R_L$

print P_L

END

Sequential Algorithms

- **Example-5: Maximum Power Transfer**

```
%Script file: calc_power
% Purpose:
% To calculate and plot the power supplied to
% a load as a function of the load resistance
%Record of revisions:
%Date       Programmer      Description of change
%=====      ==========      =====================
%03/05/2011 Alaa Khamis     Orignial Code
%Define variables:
% amps: Current flow to load (amps)
% pl:   Power supplied to load (watts)
% rl:   Resistance of the load (ohms)
% rs:   Internal resistance of the power source (ohms)
% volts:Voltage of the power source (volts0

%Set the values of source voltage and internal resistance
volts=120;
rs=50;
%Create an array of load resistance
rl=1:1:100;

%Calculate the current flow for each resistance
amps=volts./(rs+rl);
```

Sequential Algorithms

- **Example-5: Maximum Power Transfer**

```
%Calculate the power supplied to the load
pl=(amps.^2).*rl;

%plot the power versus load resistance
plot(rl,pl);
title('Plot of power versus load resistance');
xlabel('Load resistance (ohms)');
ylabel('Power (watts)');
grid on;
```

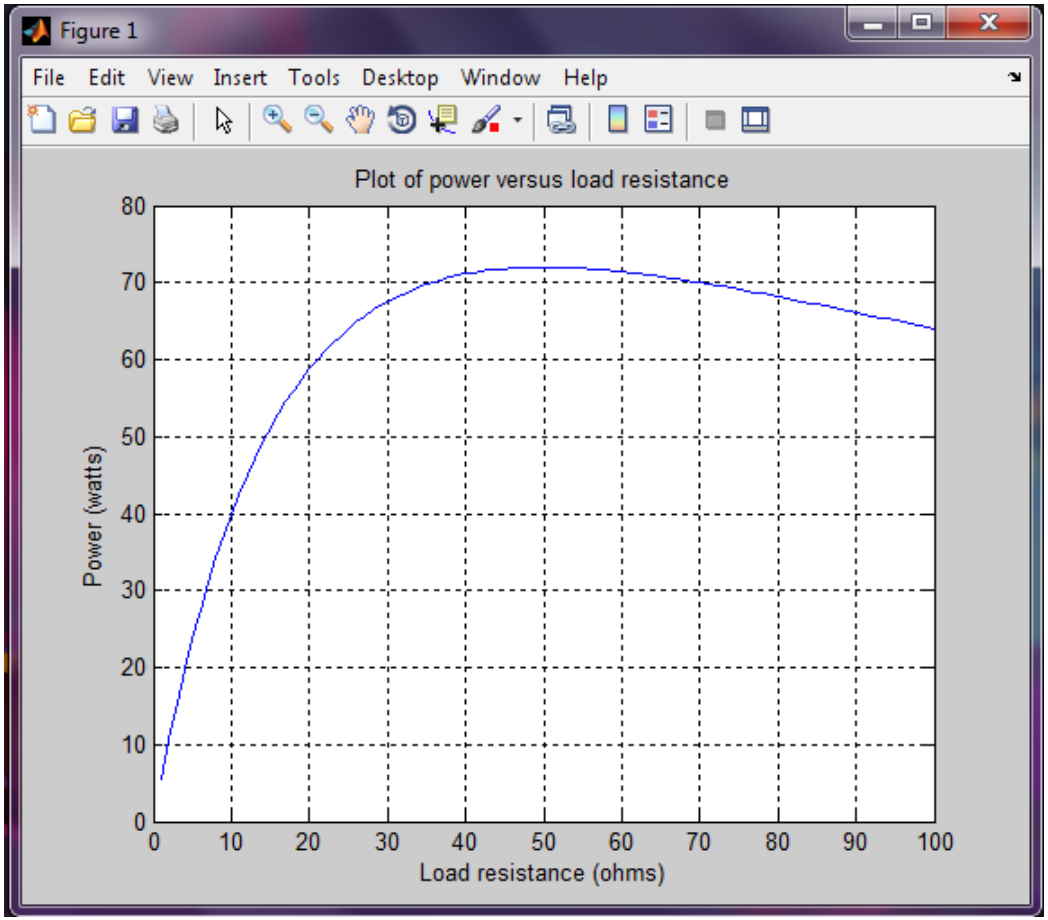
Sequential Algorithms

- **Example-5: Maximum Power Transfer**

Program Run

From the plot, we can see that the maximum power is supplied to the load when the load's resistance is 50Ω.

The power supplied to the load at this resistance is 72 watts.



Outline

- What is an Algorithm?
- Representing Algorithms
- Pseudo-code
- Algorithm Implementation
- Types of Algorithms
- Sequential Algorithms
- **Summary**

Summary

- An algorithm is a step-by-step specification of a method to solve a problem within a finite amount of time.
- Usually we use “pseudo-code” to describe algorithms.
- Pseudo-code is a notation resembling a programming language but not intended for actual compilation.
- Flowcharts can be thought of as a graphical form of pseudo-code.
- Algorithms can be implemented in any programming language.

Summary

- Sequential algorithm is formed by a list of operations (or steps) arranged in a “linear” fashion, such that the order of these steps is well defined and significant.
- Conditional operation is a control operation that allow us to alter the normal sequential flow of control in an algorithm. Conditional statements are the “question-asking” operations of an algorithm.
- An Iterative operation allows the repetition of a block of statements according to a condition. Iteration is sometimes called looping.