



Suez University

Faculty of Petroleum and Mining Engineering

Petroleum Exploration and Production Engineering Program



Iterative Algorithms

Lecture 5 – Sunday November 13, 2016

Outline

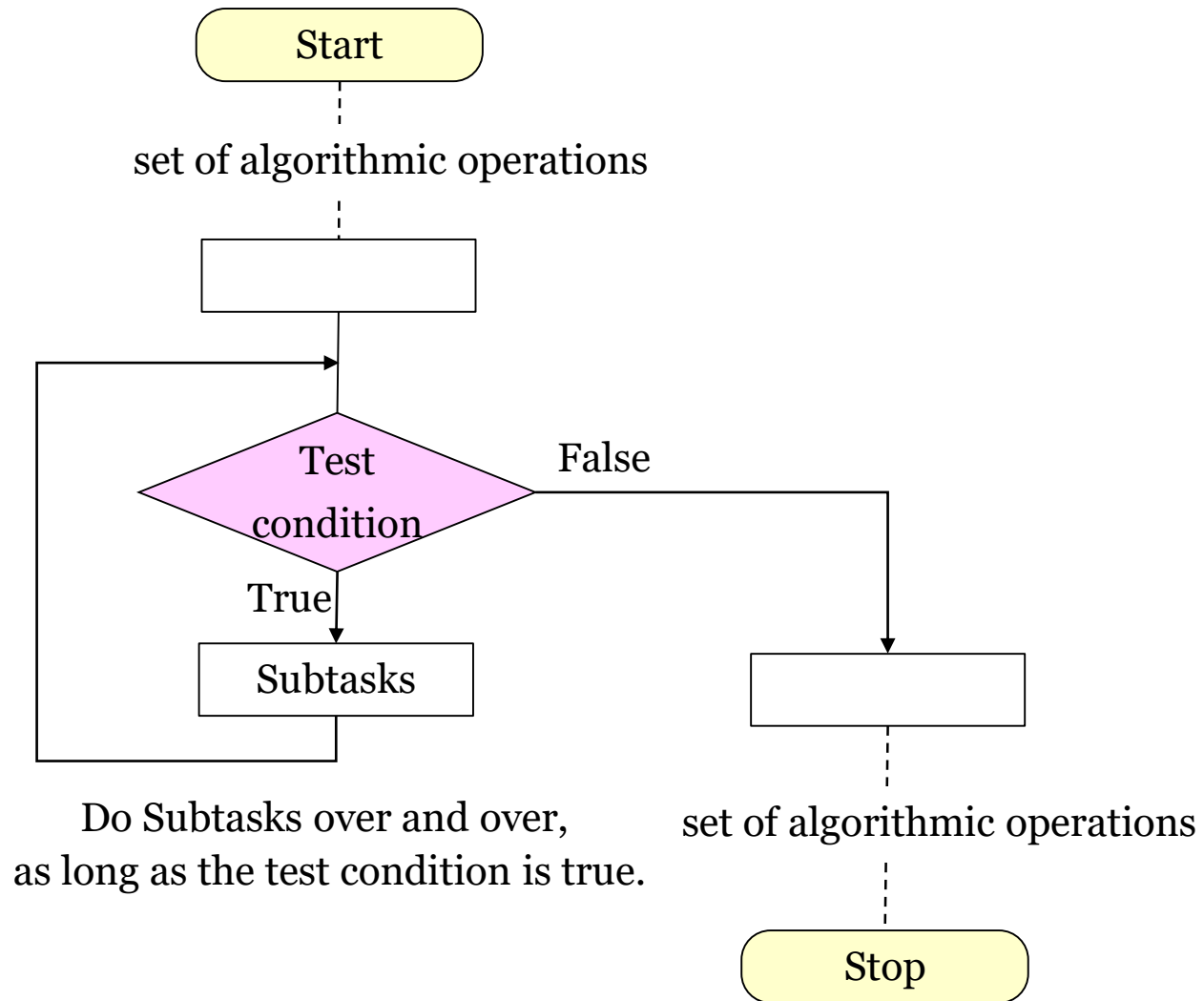
- Iterative Algorithms
- The while Loop
- The for Loop

Outline

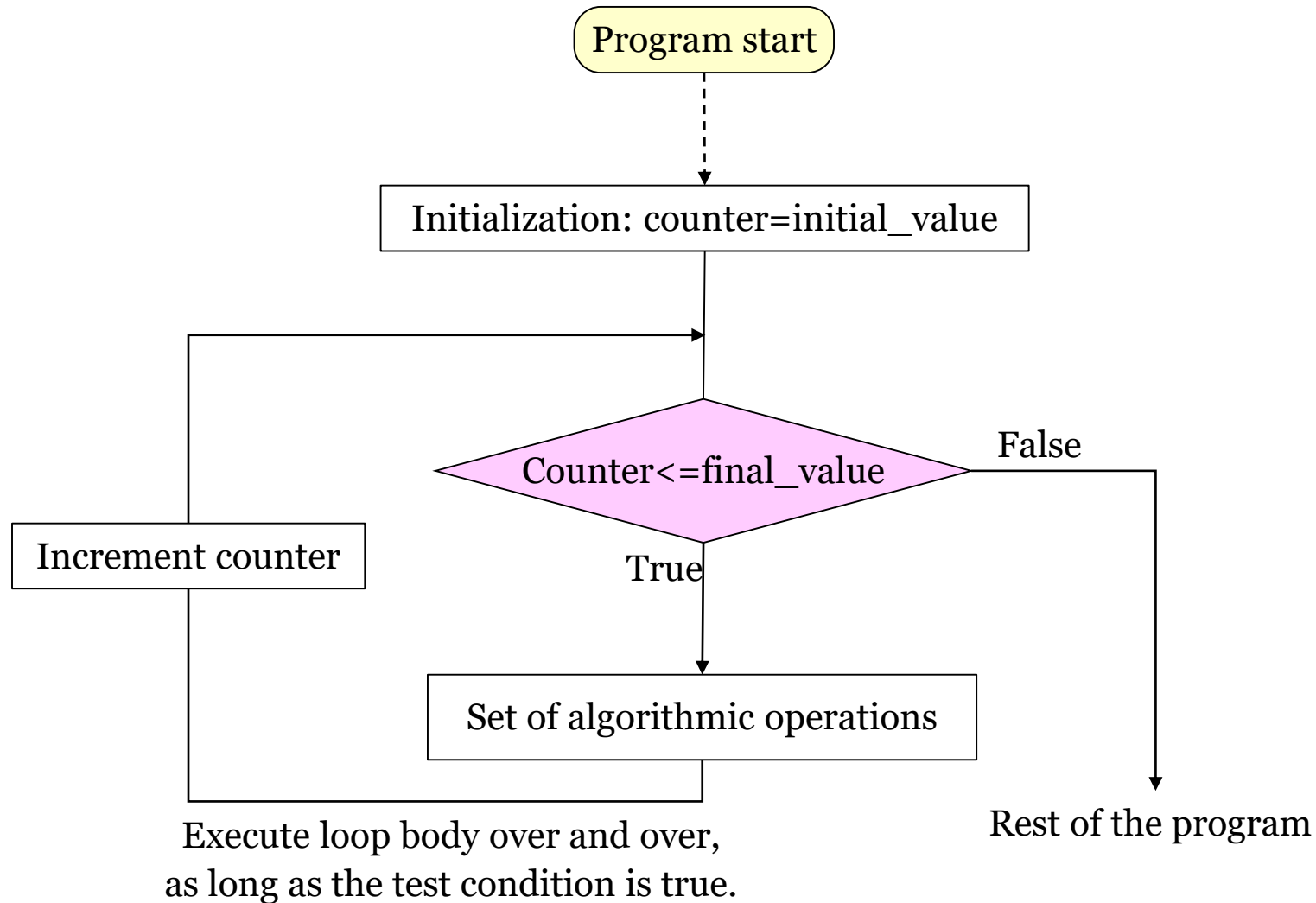
- **Iterative Algorithms**
- The while Loop
- The for Loop

Iterative Algorithms

An Iterative operation allows the repetition of a block of statements according to a condition. Iteration is sometimes called **looping**.

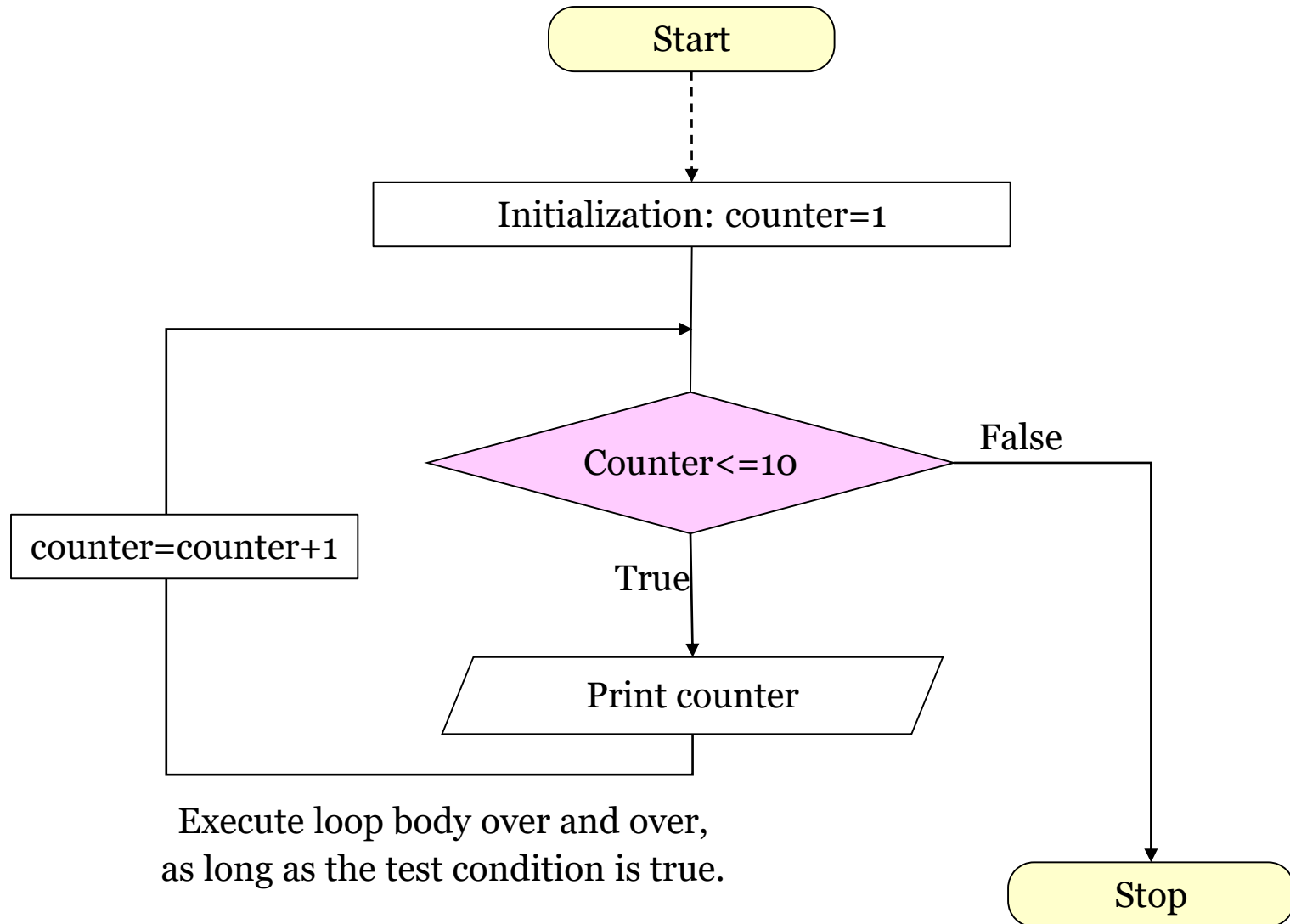


Iterative Algorithms



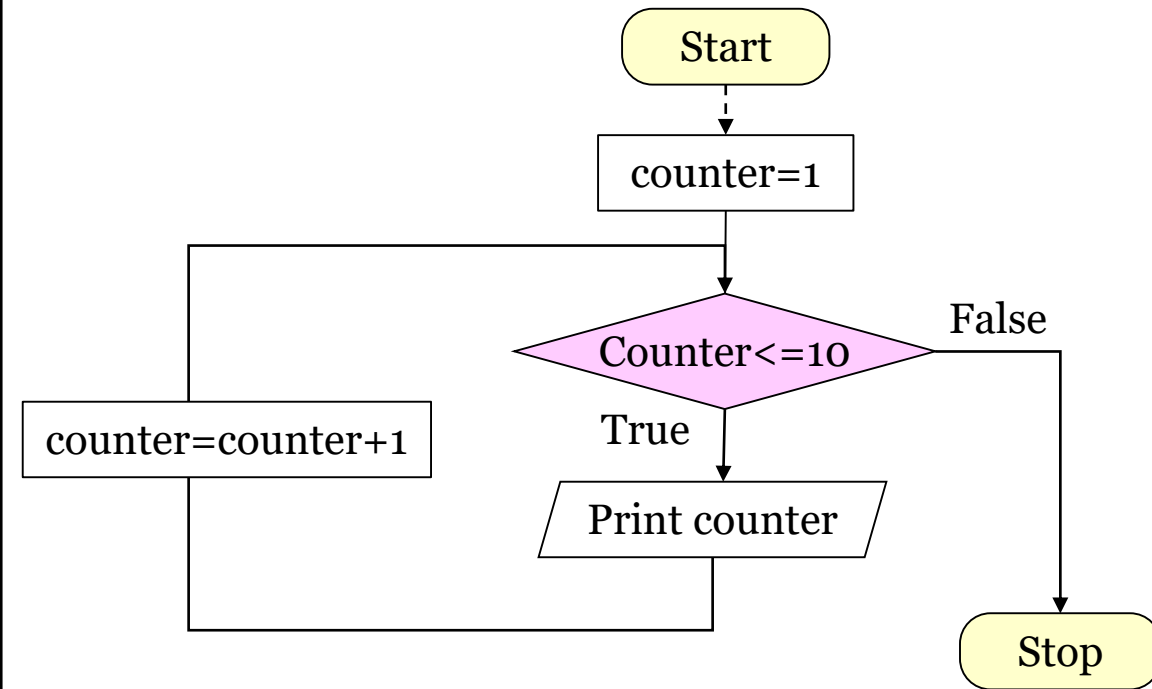
Iterative Algorithms

Example: printing numbers from one to ten.

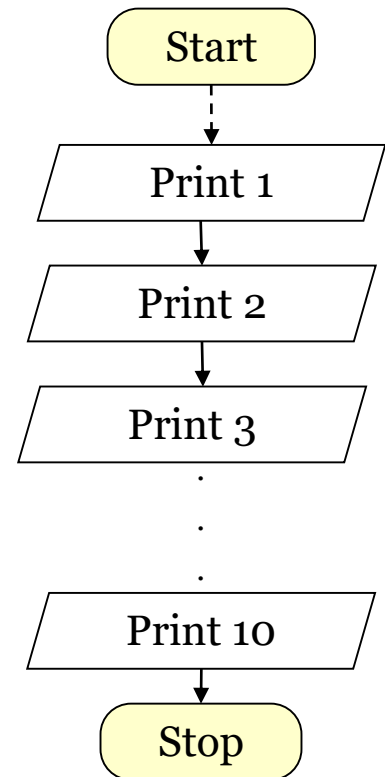


Iterative Algorithms

Example: printing numbers from one to ten.



Iterative Algorithm



Sequential Algorithm

These two programs give the same output but instead of writing an instruction 10 times, it is far better-especially if the number of instructions is huge- to write it only once and indicate when it is to be repetitively executed 10 times.

Iterative Algorithms

Example: printing numbers from one to ten.

Pseudo-code

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

```
set counter to 1
while (counter <= 10)
{
    print counter
    set counter to counter+1
}
print done
```


Iterative Algorithms

- Loops are MATLAB constructs that permit us to execute a sequence of statements more than once.

MATLAB Looping Constructs

The **while** Loop

The **for** loop

- The major difference between these two types of loop is how the repetition is controlled. The code in a **while loop** is repeated an infinite number of times until some user-defined condition is satisfied.
- By contrast, the code of a **for loop** is repeated a specific number of times, and the number of repetitions is known before the loops starts.

Outline

- Iterative Algorithms
- **The while Loop**
- The for Loop

The while Loop

```
while expression
```

```
Statement 1 }  
Statement 2 } Code block  
.... }
```

```
end
```

- The controlling *expression* produces a logical value.
- As long as *expression* is true, the code block will be executed, and then control will return to the while statement.
- This process will be repeated until the *expression* becomes false.

The while Loop

- **Example-0: Drill Exercises**

How many times will this loop print 'Hello World'?

```
n = 10;  
while n > 0  
    disp('Hello World')  
    n = n - 1;  
end
```

Solution

10 times.

The while Loop

- **Example-0: Drill Exercises**

How many times will this loop print 'Hello World'?

```
n = 1;
while n > 0
    disp('Hello World')
    n = n + 1;
end
```

Solution

This loop will continue to print 'Hello World' until the user stops the program. You can stop a program by holding down the 'Ctrl' key and simultaneously pressing the 'c' key.

The while Loop

- **Example-0: Drill Exercises**

What values will the following code print?

```
a = 1
while a < 100
    a = a*2
end
```

Solution

```
a = 1
a = 2
a = 4
a = 8
a = 16
a = 32
a = 64
a = 128
```

The while Loop

- **Example-0: Drill Exercises**

What values will the following code print?

```
a = 1;
n = 1;
while a < 100
    a = a*n
    n = n + 1;
end
```

Solution

```
a = 1
a = 2
a = 6
a = 24
a = 120
```

The while Loop

- **Example-1: Statistical Analysis**

- ◇ It is very common in science and engineering to work with large sets of numbers, each of which is a measurement of some particular property that we are interested in.
- ◇ A simple example would be the well logging data that reflects the geologic formations. For example, resistivity logging measures the subsurface electrical resistivity, which is the ability to impede the flow of electric current.
- ◇ Much of the time, we are not interested in looking closely at every single measurement that we make. Instead we want to summarize the results of a set of measurements with a few numbers that tell us a lot about the overall data set.

The while Loop

- **Example-1: Statistical Analysis**

- ◇ Two such numbers are the average (or arithmetic mean) and the standard deviation of the set of measurements.

- ◇ The average or arithmetic mean of a set of numbers is defined as:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$$

where x_i is sample i out of N samples.

- ◇ If all of the input values are available in an array, the average of a set of number may be calculated by the MATLAB function **mean**.

The while Loop

- **Example-1: Statistical Analysis**

- ◇ The standard deviation of a set of numbers is defined as:

$$s = \sqrt{\frac{N \sum_{i=1}^N x_i^2 - \left(\frac{1}{N} \sum_{i=1}^N x_i \right)^2}{N(N-1)}}$$

- ◇ Standard deviation is a measure of the **amount of scatter/dispersion** on the measurements; the greater the standard deviation, the more scattered the points in the data set are.
- ◇ Implement an algorithm that reads in a set of measurements and calculates the **mean** and the **standard deviation** of the input data set.

The while Loop

- **Example-1: Statistical Analysis**

- ◇ Inputs: data set (x)

- ◇ Output: average ($xbar$) and standard deviation (std_dev)

- ◇ Expressions:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$$

$$s = \sqrt{\frac{N \sum_{i=1}^N x_i^2 - \left(\frac{1}{N} \sum_{i=1}^N x_i \right)^2}{N(N-1)}}$$

The while Loop

- **Example-1: Statistical Analysis**

- ◇ Pseudocode:

```
BEGIN
Initialize n, sum_x, and sum_x2 to 0
Prompt user for first number
Read in first x
while x >= 0
Set n to n+1
Set sum_x to sum_x+x
Set sum_x2 to sum_x2+x^2
Prompt user for next number
Read in next x
end
```

```
Set x_bar to sum_x/n
Set std_dev to sqrt((n*sum_x2-
sum_x^2)/(n*(n-1)))
Print x_bar
Print std_dev
Print n
End
```

The while Loop

• Example-1: Statistical Analysis

◇ MATLAB Code:

```
%Script file: stats_1.m
%
% Purpose:
% This program calculates mean and standard deviation of
% an input data set containig an arbitrary number of input values:

%Record of revisions:
%Date      Programmer      Description of change
%=====
%24/01/2004 S. Chapman      Original Code
%
%Define variables:
% x:        An input data value
% n:        the number of input samples
% sum_x:    the sum of the inout values
% sum_x2:   the sum of the squares of the input values
% xbar:     the average of the input samples
% std_dev:  the standard deviation of the input samples

% Initialize sums.
n=0; sum_x=0; sum_x2=0;

% Read in first value
x=input('Enter first value:');

%while Loop to read input values
while x>=0

    % Accumulate sums
    n=n+1;
    sum_x=sum_x+x;
    sum_x2=sum_x2+x^2;

    % Read in next value
    x=input('Enter next value:');
end
```

```
% Calculate the mean and the standard deviation
x_bar=sum_x/n;
std_dev=sqrt((n*sum_x2-sum_x^2)/(n*(n-1)));

% Tell the user
fprintf('The mean of this data set is: %f\n', x_bar);
fprintf('The standard deviation of this data set is: %f\n', std_dev);
fprintf('The number of data points is: %f\n', n);
```

```
Enter first value:3
Enter next value:4
Enter next value:5
Enter next value:-1
The mean of this data set is: 4.000000
The standard deviation of this data set is: 1.000000
The number of data points is: 3.000000
>> |
```

The while Loop

- **Example-1: Statistical Analysis**

- ◇ In the preceding example, we failed to follow the design process completely.
- ◇ This failure has left the program with a **fatal flaw!** Did you spot it?
- ◇ We have failed because we did not completely test the program for all possible types of input.
- ◇ Look at the example once again. If we enter either **no numbers or only one number**, then we will be **dividing by zero** in the foregoing equations!

The while Loop

- **Example-1: Statistical Analysis**

- ◇ The **division-by-zero** error will cause divide-by-zero warnings to be printed, and the output values will be **NaN**.
- ◇ We need to modify the program to detect this problem, tell the user what the problem is' and stop gracefully'.
- ◇ A modified version of the program called **stats-2** follows.
- ◇ Here, we check to see if there are enough input values before performing the calculations. If not, the program will print out an intelligent error message and quit.
- ◇ Test the modified program for yourself.

The while Loop

- **Example-1:**
Statistical Analysis

- ◊ **Modified Code:**

```
Enter first value:1
Enter next value:
At least 2 values must be entered!
>>
```

```
% Initialize sums.
n=0; sum_x=0; sum_x2=0;

% Read in first value
x=input('Enter first value:');

%while Loop to read input values
while x>=0

    % Accumulate sums
    n=n+1;
    sum_x=sum_x+x;
    sum_x2=sum_x2+x^2;

    % Read in next value
    x=input('Enter next value:');
end

% Check to see if we have enough input data
if n<2    % Insufficient information
    display('At least 2 values must be entered!');
else % There are enough information so
    % Calculate the mean and the standard deviation
    x_bar=sum_x/n;
    std_dev=sqrt((n*sum_x2-sum_x^2)/(n*(n-1)));

% Tell the user
fprintf('The mean of this data set is: %f\n', x_bar);
fprintf('The standard deviation of this data set is: %f\n', std_dev);
fprintf('The number of data points is: %f\n', n);

end
```


The while Loop

- **Example-2: Euclidean Algorithm**

The Euclidean algorithm finds the greatest common divisor of two positive numbers X and Y by repeatedly replacing the larger number with the result of subtracting the smaller one from it until the two numbers become equal.

Express this algorithm in pseudo-code and write a MATLAB program to implement this algorithm.

The while Loop

- **Example-2: Euclidean Algorithm**

Input: X, Y

Output: greatest common divisor (GCD)

```
BEGIN
get X
get Y

while (X!=Y)
{
    if(X>Y) then
        set X=X-Y
    else
        set Y=Y-X
    endif
}
print X
END
```

The while Loop

- **Example-2: Euclidean Algorithm**

Greatest common divisor (GCD)

Assume $x=13$ & $y=4$

$$x=13-4=9$$

$$x=9-4=5$$

$$x=5-4=1$$

$$y=4-1=3$$

$$y=3-1=2$$

$$y=2-1=1$$

GCD=1

```
BEGIN
get X
get Y

while (X!=Y)
{
    if(X>Y) then
        set X=X-Y
    else
        set Y=Y-X
    endif
}
print X
END
```

The while Loop

- **Example-2: Euclidean Algorithm**

Greatest common divisor (GCD)

Assume $x=12$ and $y=4$

$$x=12-4=8$$

$$x=8-4=4$$

$$\text{GCD}=4$$

```
BEGIN
get X
get Y

while (X!=Y)
{
    if(X>Y) then
        set X=X-Y
    else
        set Y=Y-X
    endif
}
print X
END
```

The while Loop

• Example-2: Euclidean Algorithm

```
%Script file: GCD.m
%
% Purpose:
% This program calculates the greatest common divisor (GCD)
% of two positive numbers

%Record of revisions:
%Date      Programmer      Description of change
%=====
%22/05/2011 A. Khamis      Original Code
%
%Define variables:
% x:  first positive number
% y:  second positive number

% Get two positive numebers.
disp('This program calculates the Greatest Common Divisor (GCD) of two positive numbers. ');
x=input('Enter the first positive number: ');
y=input('Enter the second positive number: ');

% Calculate the Greatest Common Divisor (GCD)
while x~=y
    if x>y
        x=x-y
    else
        y=y-x
    end
end

% Tell user
fprintf('The Greatest Common Divisor (GCD) is %6.2f.\n', x);
```

Outline

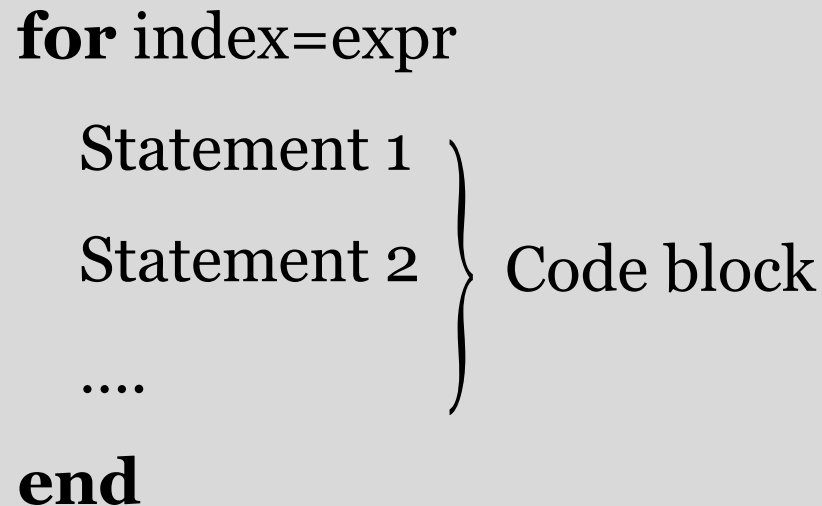
- Iterative Algorithms
- The while Loop
- **The for Loop**

The for Loop

- The for loop is a loop that executes a block of statements a specified number of times.
- The for loop has the form:

Loop variable or loop index

Expression \Rightarrow **first:incr:last**



The diagram shows a code block for a for loop. The text is as follows:

```
for index=expr  
    Statement 1  
    Statement 2  
    ....  
end
```

Two red arrows point from the text above to the code. The first arrow points from "Loop variable or loop index" to "index" in "index=expr". The second arrow points from "Expression \Rightarrow first:incr:last" to "expr" in "index=expr". A right-facing curly brace groups "Statement 1", "Statement 2", and "...." and is labeled "Code block".

The while Loop

- **Example-0: Drill Exercises**

How many times will this loop print 'Hello World'?

```
for a=0:50  
disp('Hello World')  
end
```

Solution

The code `0:50` creates a vector of integers starting at 0 and going to 50; this vector has 51 elements. "Hello World" will be printed once for each element in the vector (51 times).

The while Loop

- **Example-0: Drill Exercises**

What sequence of numbers will the following for loop print?

```
n = 10;  
for j = 1:n  
    n = n-1;  
    j;  
end
```

Solution

In the first line, the value of n is set to 10. The code `1:n` creates a vector of integers from 1 to 10. Each iteration through the loop sets j to the next element of this vector, so j will be sent to each value 1 through 10 in succession, and this sequence of values will be printed. Note that each time through the loop, the value of n is decreased by 1; the final value of n will be 0. Even though the value of n is changed in the loop, the number of iterations through the loop is not affected, because the vector of integers is computed once before the loop is executed and does not depend on subsequent values of n .

The while Loop

- **Example-0: Drill Exercises**

What value will the following program print? What is a simpler way to achieve the same results?

Solution:

The **d-loop** will be executed **7 times**. In each iteration of the d loop, the **h-loop** will be executed **24 times**. In each iteration of the h loop, the **m-loop** will be executed **60 times**. In each iteration of the m loop, the **s-loop** will be executed **60 times**. So the variable count will be incremented **$7 \times 24 \times 60 \times 60 = 604800$ times**.

```
count = 0;
for d = 1:7
    for h = 1:24
        for m = 1:60
            for s = 1:60
                count = count + 1;
            end
        end
    end
end
count
```

The while Loop

- **Example-0: Drill Exercises**

What value will the following program print? What is a simpler way to achieve the same results?

Solution:

A simpler way to achieve the same results is the command

7*24*60*60

```
count = 0;
for d = 1:7
    for h = 1:24
        for m = 1:60
            for s = 1:60
                count = count + 1;
            end
        end
    end
end
end
count
```

The for Loop

• Example-1: Factorial Function

◇ Write a MATLAB program to calculate the factorial function for numbers from 1 to 10.

Inputs: Numbers from 1 to 10

Outputs: print factorials

Expression:

$$N! = 1 \quad N = 0$$

$$N! = N * (N - 1) * \dots * 3 * 2 * 1 \quad N > 0$$

```
BEGIN
set f to 1 ← Initial value
For n=1 to 10
{
set f to f*n
print f
set n to n + 1 ← Step
}
print Done
END
```

Final value

The for Loop

- **Example-1: Factorial Function**

- ◊ **MATLAB Code:**

```
%Script file: factorial.m
%
% Purpose:
% This program calculates factorials for numebr from 1 to 10

%Record of revisions:
%Date          Programmer      Description of change
%=====      =====
%22/05/2011 A. Khamis      Orignial Code
%
%Define variables:
% ii:          for loop index
% n_factorial: factorial of the number

% Initialize factorial.
n_factorial=1;

for ii=1:10
    n_factorial=n_factorial*ii;
    fprintf('%6.2f factorial is %6.2f.\n',ii,n_factorial);
end
```

```
>> factorial
1.00 factorial is 1.00.
2.00 factorial is 2.00.
3.00 factorial is 6.00.
4.00 factorial is 24.00.
5.00 factorial is 120.00.
6.00 factorial is 720.00.
7.00 factorial is 5040.00.
8.00 factorial is 40320.00.
9.00 factorial is 362880.00.
10.00 factorial is 3628800.00.
>> |
```

The for Loop

- **Example-2: Calculating the Day of Year**
 - ◇ The day of year is the number of days (including the current day) that have elapsed since the beginning of a given year.
 - ◇ It is a number in the range 1 to 365 for ordinary years, and 1 to 366 for leap years.
 - ◇ Write a MATLAB program that accepts a day, month, and year, and calculates the day of year corresponding to that date.

The for Loop

- **Example-2: Calculating the Day of Year**

- ◇ Inputs: day, month and year

- ◇ Output: day of year

- ◇ Expression:

- A number in the range 1 to 365 for ordinary years, and
- 1 to 366 for leap years.

The for Loop

- **Example-2: Calculating the Day of Year**

- ◇ In the **Gregorian calendar**, leap years are determined by the following rules:

1. Years evenly divisible by 400 are leap years.
2. Years evenly divisible by 100 but not by 400 are not leap years.
3. All years divisible by 4 but not by 100 are leap years.
4. All other years are not leap years.

The for Loop

- **Example-2: Calculating the Day of Year**

- ◇ In the **Gregorian calendar**, leap years are determined by the following rules:

1. Years evenly divisible by 400 are leap years.

Ex. 2000

1. Years evenly divisible by 100 but not by 400 are not leap years.

Ex. 1900

1. All years divisible by 4 but not by 100 are leap years.

Ex. 1984

1. All other years are not leap years.

The for Loop

- **Example-2: Calculating the Day of Year**

- ◇ Pseudocode:

```
BEGIN
```

```
Get day, month and year
```

```
if (year%400=0) then
```

```
    leap_day=1
```

```
else
```

```
    leap_day=0
```

```
endif
```

```
Set day_of_year=day
```

```
for ii=1 to month-1
```

```
    Switch (ii)
```

```
        Case 1,3,5,7,8,10,12
```

```
            Set day_of_year=day_of_year+31
```

```
        Case 4,6,9,11
```

```
            Set day_of_year=day_of_year+30
```

```
        Case 2
```

```
            Set day_of_year=day_of_year+28+leap_day
```

```
    Print day_of_year
```

```
End
```

The for Loop

• Example-2: Calculating the Day of Year

◇ MATLAB Code:

```
%Script file: doy.m
%
% Purpose:
% This program calculates the day of year corresponding
% to a speicied date. It illustrates the use swtich and
% for constructs

%Record of revisions:
%Date      Programmer      Description of change
%=====  =====
%27/01/2007 S. Chapman      Orignial Code
%
%Define variables:
% day:      day (dd)
% month:    month (mm)
% year:     year (yyyy)
% ii:       loop index
% leap_day: extra day for leap year
% day_of_year: day of year

% Get day, month, and year to conert.
disp('This program calculates the dya of year given the speicied date. ');
day=input('Enter specified day (1-31): ');
month=input('Enter specified month (1-12): ');
year=input('Enter specified year (yyyy): ');

% Check for leap year, and add extra day if necessary
if mod(year,400)==0
    leap_day=1;    % Years divisible by 400 are leap years
elseif mod(year,100)==0
    leap_day=0;    % Other centurries are not leap years
elseif mod(year,4)==0
    leap_day=1;    % Otherwise every 4th year is a leap years
else
    leap_day=0;    % Other years are not leap years
end
```

```
% Calculate day of year by adding current day to the other day in previous months.
day_of_year=day;

for ii=1:month-1
    % Add days in months from January to last month
    switch (ii)
        case {1,3,5,7,8,10,12}
            day_of_year=day_of_year+31;
        case {4,6,9,11}
            day_of_year=day_of_year+30;
        case 2
            day_of_year=day_of_year+28+leap_day;
    end
end

% Tell user
fprintf('The date %2d/%2d/%4d is day of year %d.\n', day, month, year, day_of_year);
```

```
>> doy
This program calculates the dya of year given the speicied date.
Enter specified day (1-31):5
Enter specified month (1-12):11
Enter specified year (yyyy):2007
The date 5/11/2007 is day of year 309.
>> |
```