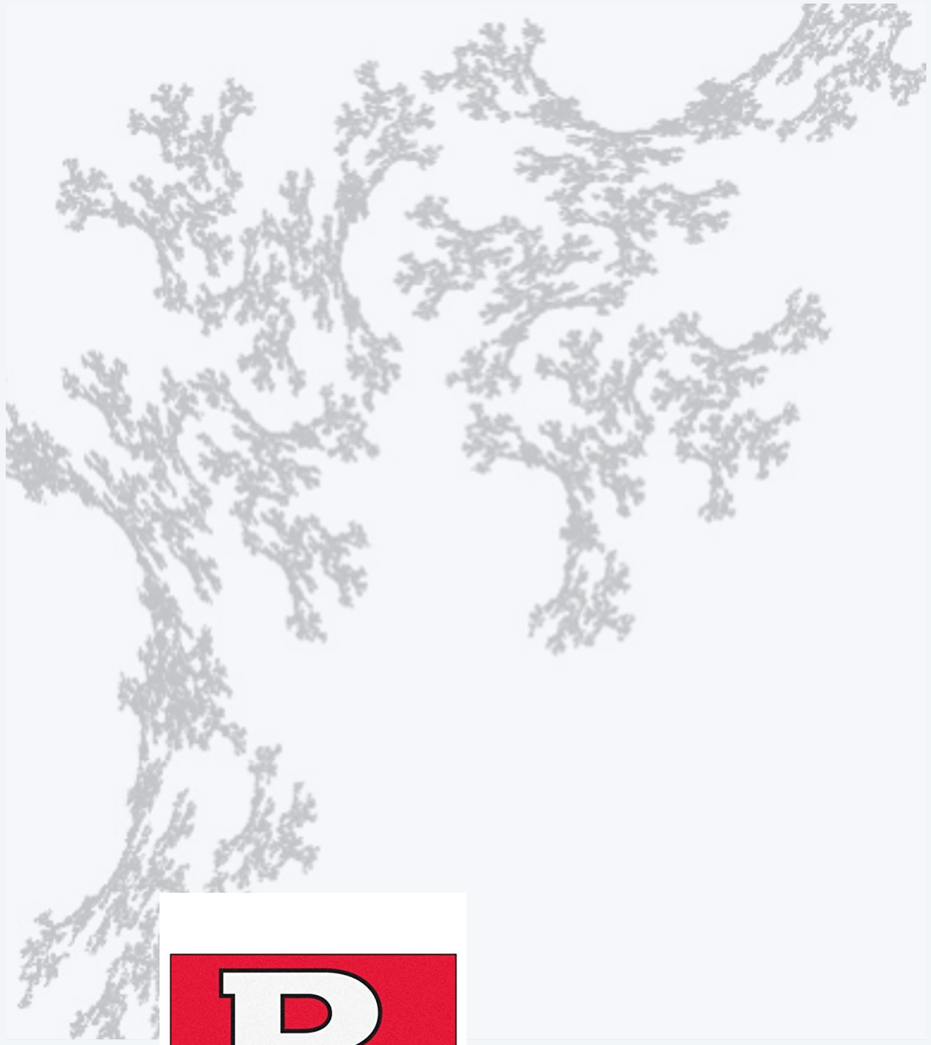


INTRODUCTION TO COMPUTER SCIENCE

Rutgers University

2. Algorithm Implementation and counting

http://introcs.cs.rutgers.edu

INTRODUCTION TO COMPUTER SCIENCE

Rutgers University

2. Algorithmic Implementation and counting

- From design to implementation
- Adding control flow
- Examples of programs
- Counting operations
- Measuring performance

Styles Adopted from Slides by Sedgewick and Wayne

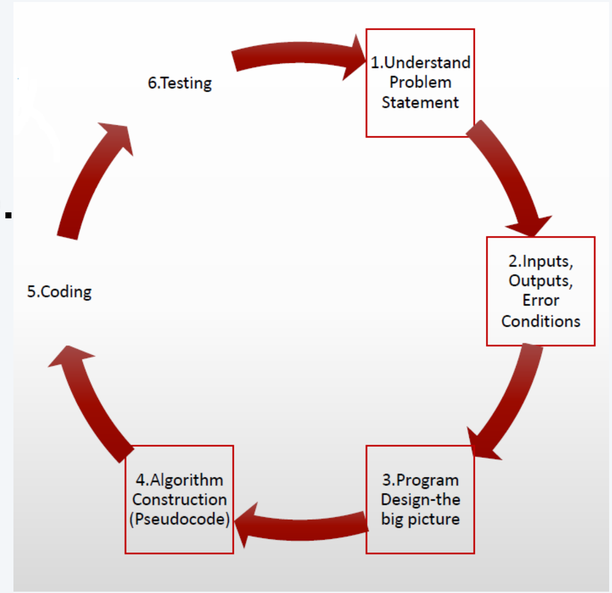
From Design to Implementation

Learn how to convert the algorithm from design to pseudocode.

Pseudocode can be a good way to express your algorithm as a first step.

from design to code →

Programming is about implementing algorithms in a programming language (Java-in this course).

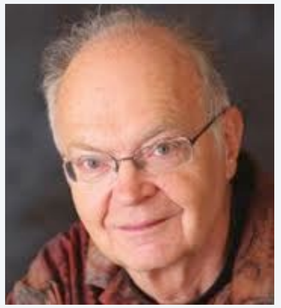


Algorithms, in general can be expressed as

- Flow charts
- Pseudocode
- Program code

" my general working style is to write everything first with pencil and paper, sitting beside a big waste basket"

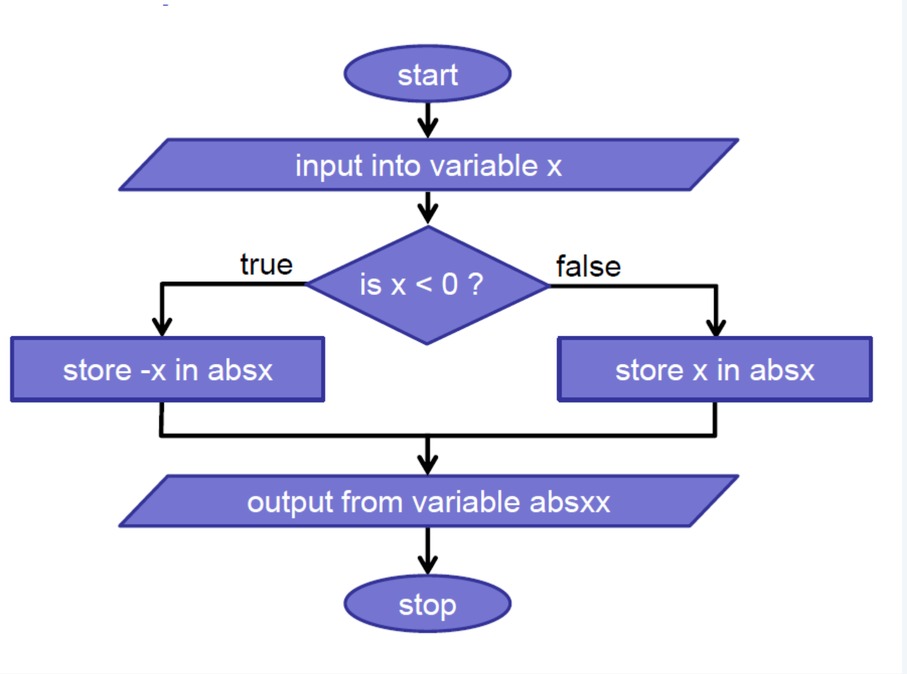
– Don Knuth



Flow Charts

Flowcharts:

- Allow organizing control flow more visually.
- *Check the path of the control based on input*.
- Change the path based on input.



Example (Absolute Value)

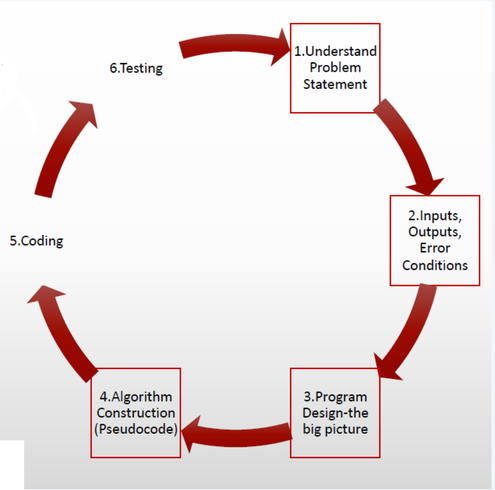
- Read a number.
- If the number is positive, then store the number as is.
- If the number is negative, store the negative of the number

Flow charts, Pseudocode and Program code

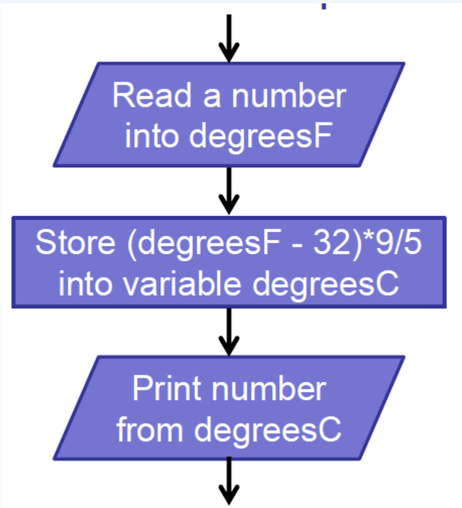
flow chart can be a good way to express the algorithm visually

pseudo code is a way to express your algorithm using pseudo instructions

program code is a way to implement your algorithm using Java



from design to code



Flow chart

READ degF

COMPUTE degC **AS** (degF – 32)*5/9

DISPLAY degC

Pseudocode

```
double degF, degC;
degF = StdIn.readDouble();
degC = (degF – 32)*5/9;
StdOut.println(degC);
```

Java code

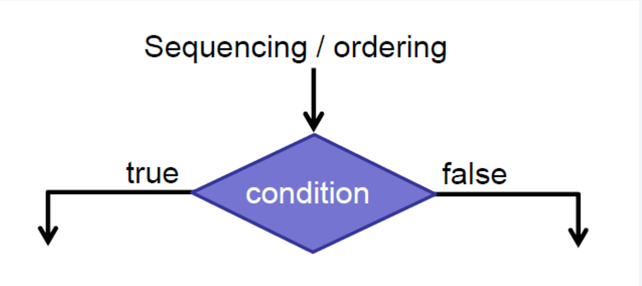
2. Algorithm Implementation and counting

- From design to implementation
- Adding control flow
- Examples of programs
- Counting operations
- Measuring performance

Adding control flow

Programming

- Writing logical instructions to perform computation.
- *Programs execute instructions sequentially by default*.
- Control flow enables programs to take different paths.



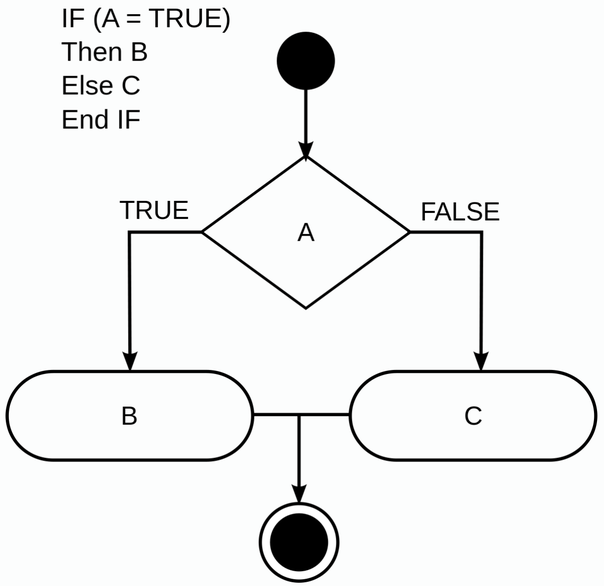
Challenges

- Learn how to think of control flow.
- Understand the semantics of various control flows.
- Apply them to program instructions.

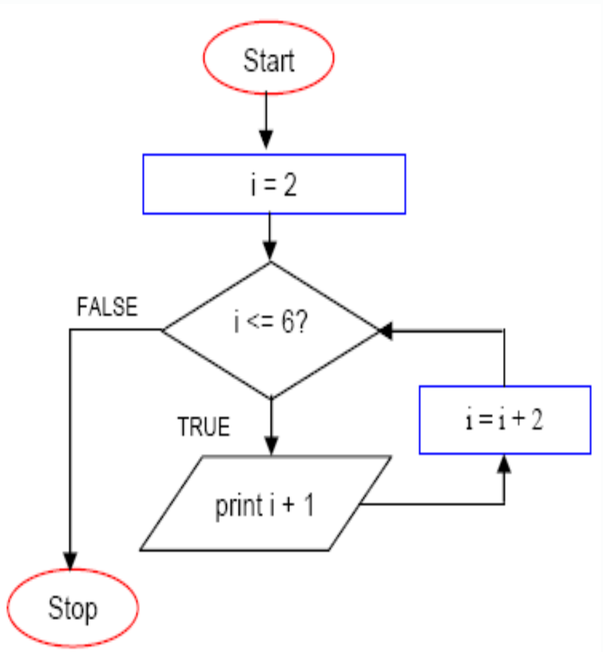
Basic control flow

Branching and Iteration

- if-else-statement: Branching based on a decision.
- while loop: Iteration while a condition is true.



If-else-statement



while loop

What prints?

2. Algorithm Implementation and counting

- From design to implementation
- Adding control flow
- Examples of programs
- Counting operations
- Measuring performance

Example Program 1

Problem. Write a program to find the larger of two numbers.

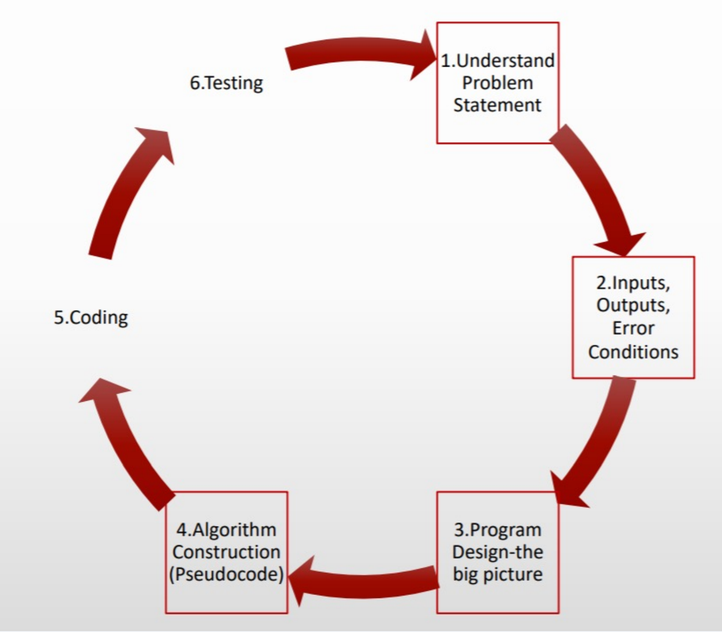
Solution Design. Input: two numbers

Output: larger of the two

Preconditions: input values can be compared

Error Conditions: none

Test cases: provide [input][output] pairs.



Algorithm.

```
READ firstNumber
READ secondNumber
IF firstNumber > secondNumber THEN
    DISPLAY firstNumber
ELSE
    DISPLAY secondNumber
ENDIF
```

TEST CASES:

[input][output]

[-2, 3][3]

[0, -1][0]

[2, 2][2]

[3, -3][3]

Example Program 2 (with error handling)

Problem. An algorithm to determine the pay, given the hours worked and rate per hour.

Solution Design. Input, output, error conditions

Algorithm.

READ hoursWorked

READ ratePerHour

IF hoursWorked < 0 **OR** ratePerHour < 0 **THEN**

    DISPLAY error_message

ELSE

    COMPUTE totalPay **AS** hoursWorked*ratePerHour

    DISPLAY totalPay

ENDIF

Input: hoursWorked, ratePerHour

Output: totalPay

Error Conditions: hoursWorked or ratePerHour < 0

Test Cases [input][output] pairs

- [hoursWorked, ratePerHour] [totalPay]
- [3, 4.5][13.5], [3.5, 4][14.0] ← Good input
- [-3, 8][ERROR], [8, -2][ERROR] ← Bad input

Example Program 3

Problem. Convert Fahrenheit to Celsius.

Solution Design. Find out what inputs are needed.

Determine error conditions (if any)

(Absolute zero is the lowest possible temperature where nothing could be colder, and no heat energy remains in a substance. Input should not be less than -457.67F)

Determine the output.

Test the program.

INPUT: degrees Fahrenheit

OUTPUT: degrees Celsius

Error Conditions: degreesF < -457.67 (absolute 0)

TEST CASES [input][output] pairs:

[32][0]

[-457.68][ERROR]

[-457.67][-272.038]

[65][18.33]

ALGORITHM:

```
READ degreesF
IF degreesF < -457.67 THEN
   DISPLAY ERROR: Temperature must be greater than or equal to -457.67
ELSE
  COMPUTE degreesC AS (degreesF – 32) * 5/9
  DISPLAY degreesC
ENDIF
```

Example Program 4

Problem. Determine if an integer is even or odd.

Solution Design. **Input**: an integer

Output: number is even or odd

Error Conditions: none

ALGORITHM:

```
READ number
IF number % 2 IS 0  THEN
   DISPLAY even
ELSE
  DISPLAY odd
ENDIF
```

TEST CASES:

[input][output]

[-2][even]

[0][even]

[2][even]

[3][odd]

Example Program 5

Problem. A cake of diameter less than 6” is $10.00. A cake of diameter between 6” and 12” (inclusive) is $15.00. A cake of diameter larger than 12” is $20.00. Find the cost of the cake given the diameter.

ALGORITHM:

```
READ cakeSize
IF cakeSize <= 0 THEN
   DISPLAY error
ELSE
    IF cakeSize < 6 THEN
        SET price TO 10.00
    ELSE
        IF cakeSize >= 6 AND cakeSize <= 12  THEN
              SET price TO 15.00
        ELSE
             SET price TO 20.00
        ENDIF
    ENDIF
    DISPLAY price
ENDIF
```

INPUT: cake diameter size

OUTPUT: cost of cake

ERROR CONDITIONS:
diameter size <=0

TEST CASES:

[input][output] pairs

[-3][error]

[0][error] -*boundary*

[5][10]

[6][15] -*boundary*

[8][15]

[12][15] -*boundary*

[13][20]

[15][20]

Example Program 6

Problem. Given three inputs(num, num2, num3), display TRUE if they are in strictly increasing order, such as 2 3 8, or 15 16 17, but not 16 15 17 or 25 25 27. Display FALSE otherwise.

ALGORITHM:

```
READ num1
READ num2
READ num3

IF num1 < num2 AND num2 < num3 THEN
   DISPLAY TRUE
ELSE
   DISPLAY FALSE
ENDIF
```

INPUT: 3 numbers

OUTPUT: TRUE if the numbers are in strictly increasing order, FALSE otherwise

ERROR CONDITIONS: none

TEST CASES:

[input][output] pairs

[-3, 0 3][TRUE]

[0, 2, 4][TRUE]

[5, 5, 7][FALSE]

[5, 4, 6][FALSE]

2. Algorithm Implementation and counting

- From design to implementation
- Adding control flow
- Examples of programs
- Counting operations
- Measuring performance

Rules of Counting operations

READ One operation

COMPUTE One operation

ADD One operation

SUBTRACT One operation

DISPLAY One operation

SET One operation

IF Total number of compares

WHILE Depends on the number of the iterations (the number of times the loop is executed)

Do not count as operations:

ELSE, ENDIF, ENDWHILE, *ENDFOR*, HALT

Examples of counting operations


```
READ firstNumber                          ← 1 operation
READ secondtNumber                        ← 1 operation
IF firstNumber > secondNumber THEN        ← 1 operation
    DISPLAY  firstNumber                  ← 1 operation
ELSE                                            OR
    DISPLAY secondNumber                  ←1 operation
ENDIF                                     Total: 4 operations
```

Example 1

```
ALGORITHM:
    READ hoursWorked                                          ←1 operation
    READ ratePerHour                                          ←1 operation
    IF hoursWorked < 0 OR ratePerHour < 0 THEN                ←2 operations
        DISPLAY ERROR                                         ←1 operation
    ELSE
        COMPUTE totalPay AS hoursWorked * ratePerHour         ←1 operation
        DISPLAY totalPay                                      ←1 operation
    ENDIF                                                     Total: 5 or 6 operations
```

Example 2

Example Program 3

Problem. Convert Fahrenheit to Celsius.

```
READ degreesF                                          ← 1 operation
IF degreesF < -457.67 THEN                             ← 1 operation
   DISPLAY ERROR                                       ← 1 operation
ELSE
  COMPUTE degreesC AS (degreesF – 32) * 5/9            ← 1 operation
  DISPLAY degreesC                                     ←1 operation
ENDIF

                                                       Total: 3 or 4 operations
```

Example Program 4

Problem. Determine if an integer is even or odd.

Solution Design. **Input**: an integer

Output: number is even or odd

Error Conditions: none

ALGORITHM:

```
READ number
IF number % 2 IS 0  THEN
   DISPLAY even
ELSE
  DISPLAY odd
ENDIF
```

Number of operations???

Example Program 5

LO 2.1

Problem. A cake of diameter less than 6 is $10.00. A cake of diameter between 6” and 12” (inclusive) is $15.00. A cake of diameter larger than 12” is $20.00. Find the cost of the cake given the diameter.

ALGORITHM:

```
READ cakeSize                                          ← 1 operation
IF cakeSize <= 0 THEN                                  ← 1 operation
   DISPLAY error                                       ←1 operation
ELSE
     IF cakeSize < 6 THEN                              ← 1 operation
          SET price TO 10.00                           ←1 operation
     ELSE
          IF cakeSize >= 6 AND cakeSize <= 12  THEN    ←2 operations
                SET price TO 15.00                     ←1 operation
          ELSE
               SET price TO 20.00                      ←1 operation
          ENDIF
     ENDIF
     DISPLAY price                                     ←1 operation
ENDIF
```

Total operations? (Minimum and Maximum?

Minimum = 1 + 1 + 1 = 3

Maximum = 1 + 1 + 1 + 2 + 1 + 1 = 7

Example Program 6

LO 2.1

Problem. Given three inputs(num, num2, num3), display TRUE if they are in strictly increasing order, such as 2 3 8, or 15 16 17, but not 16 15 17 or 25 25 27. Display FALSE otherwise.

ALGORITHM:

```
READ num1
READ num2
READ num3

IF num1 < num2 AND num2 < num3 THEN
   DISPLAY TRUE
ELSE
   DISPLAY FALSE
ENDIF
```

Number of operations???

Example 7

Find (and display) the largest of three negative numbers

```
READ num1
READ num2
READ num3
IF (num1 >= 0) OR (num2 >= 0) OR (num3 >= 0) THEN
        DISPLAY ERROR
        HALT
ELSE
        SET largest TO num1
        IF (num2 > largest) THEN
            SET largest TO num2
        ENDIF
        IF (num3 > largest) THEN
            SET largest TO num3
        ENDIF
        DISPLAY largest
ENDIF
```

Number of operations???

MINIMUM?

MAXIMUM?

Example 8 – WHILE LOOPS

LO 2.2., 2.3

Problem. Display the positive integers less than or equal to 5.

Solution Design. Input. none

Output. integers from 1 to 5

Precondition. None

Algorithm.

```
SET num TO 1
WHILE num <= 5
    DISPLAY num
    ADD 1 TO num
ENDWHILE
```

1 operation

1 comparison operation done 5+1 times

1 operation done 5 times

1 operation done 5 times

Total operations?

1 + 6 + 5 + 5 = 17

Example 9 WHILE LOOPS

Problem. Given an integer n, display the integers from 0,1,2…n. Count the total operations.

Solution Design. Input: the value n. **SUPPOSE n = 7**

Output: integers from 0,1,2,...,n

PRECONDITION: n >= 0. **SUPPOSE n = 7**

ALGORITHM:

READ n **(7)**	← 1 operation **(remember n = 7)**
SET count **TO** 0	← 1 operation
WHILE count <= **7**	← 1 comparison operation done **8** + 1 times
DISPLAY count	← 1 operation done 8 times
ADD 1 **TO** count	← 1 operation done 8 times
ENDWHILE	

When the condition is false and the loop ends

Total operations.

2 + 9 + 8 + 8 = 27

SUPPOSE n = 10

Example 10 WHILE LOOPS

Problem. Given an integer n, display the integers from 0,1,2…n. Count the total operations.

Solution Design. Input: the value n

Output: integers from 0,1,2,...,n

PRECONDITION: n >= 0

ALGORITHM:

READ n	← 1 operation
SET count **TO** 0	← 1 operation
WHILE count <= n	← 1 comparison operation done n + 1 + 1 times
DISPLAY count	← 1 operation done n + 1 times
ADD 1 **TO** count	← 1 operation done n + 1 times
ENDWHILE	

When the condition is false and the loop ends

Total operations.

2 + (n + 1) + 1 + 2(n+1)

Total operations.

3 + 3(n+1)=3n+6

Example 11 – WHILE LOOPS

Display all even numbers from 1 to 100

Algorithm 1

```
SET num TO 2
WHILE (num <= 100)
    DISPLAY num
    ADD 2 TO num
END WHILE
```

Algorithm 2

```
SET num TO 1
WHILE (num <= 100)
    IF num % 2 IS 0 THEN
        DISPLAY num
     ENDIF
     ADD 1 TO num
END WHILE
```

Number of operations???

Algorithm 1?

Algorithm 2?

Example 12 WHILE LOOPS

Problem. Find the **sum** of the first 50 counting numbers (1 + 2 + 3 + … + 50).

Solution Design. Input: none

Output: the sum 1 + 2 + 3 + … + 50

Precondition: none

Algorithm.

SET num **TO** 1 ← 1 operation

SET sum **TO** 0 ← 1 operation

WHILE num <= 50 ← 1 comparison done 50 + 1 times

  COMPUTE sum **AS** sum + num ← 1 operation done 50 times

  ADD 1 **TO** num ← 1 operation done 50 times

ENDWHILE

DISPLAY sum ← 1 operation

Total operations?

2 + 3 * 50 + 1 + 1 = 154

Example 13

Another way?
Sum of first n counting numbers

```
READ n
COMPUTE sum AS n/2 * (1 + n)
DISPLAY sum
```

I love the story of Carl Friedrich Gauss—who, as an elementary student in the late 1700s, amazed his teacher with how quickly he found the sum of the integers from 1 to 100 to be 5,050. Gauss recognized he had fifty pairs of numbers when he added the first and last number in the series, the second and second-last number in the series, and so on. For example: (1 + 100), (2 + 99), (3 + 98), . . . , and each pair has a sum of 101.

2. Algorithm Implementation and counting

- From design to implementation
- Adding control flow
- Examples of programs
- Counting operations
- Measuring performance (optional)

Measuring Performance

Counting Operations

is a way to assess the performance of an algorithm

Each Operation

takes some amount of computer time (in CPU cycles)

Performance of an Algorithm

depends on many variables such as processor speed

Performance of an Algorithm

is typically expressed based on some variable (e.g. **data size n**) that is only known to the algorithm at the run time.

RECALL: Example Program 10

Problem. Given an integer **n**, display the integers from 0,1,2…n . Count the total operations.

Solution Design. Input: the value n

Output: integers from 0,1,2,..n

Precondition: number is positive

Algorithm.

```
READ n                          <-- 1 operation
SET count TO 0                  <-- 1 operation
WHILE count <= n                <-- 1 comparison operation done n +1+1 times
    DISPLAY count               <-- 1 operation done n+1 times
    ADD 1 TO count              <-- 1 operation done n+1 times
ENDWHILE
```

Total operations.

3 + 3(n+1)

Another Example:


```
SET count TO 0                          ← 1 operation
WHILE count <= n                        ← 1 operation done n + 1 + 1 times
   DISPLAY count                        ← 1 operation done n + 1 times
   COMPUTE count AS count + 1           ← 1 operation done n + 1 times
ENDWHILE
```

Q. What is the performance of this algorithm in terms of n?

A. 1 + (n+1+1) + 2(n+1) = 3n + 5

ONE FINAL EXAMPLE!

Problem. Given an integer n, print all proper factors of n.

Solution Design. Input: the value n

Output: all integers that divide n evenly (excluding 1 and n)

Precondition: n is positive

Algorithm.

READ n ← 1 operation

SET divisor **TO** 2 ← 1 operation

WHILE divisor < n ← 1 comparison operation done n – 2 + 1 times

  IF n % divisor IS 0 **THEN** ← 1 operation done n - 2 times

    DISPLAY divisor ← 1 operation done ?? Times (?? = number of divisors of n)

  ENDIF

  ADD 1 **TO** divisor ← 1 comparison operation done n - 2 times

ENDWHILE

Total operations?

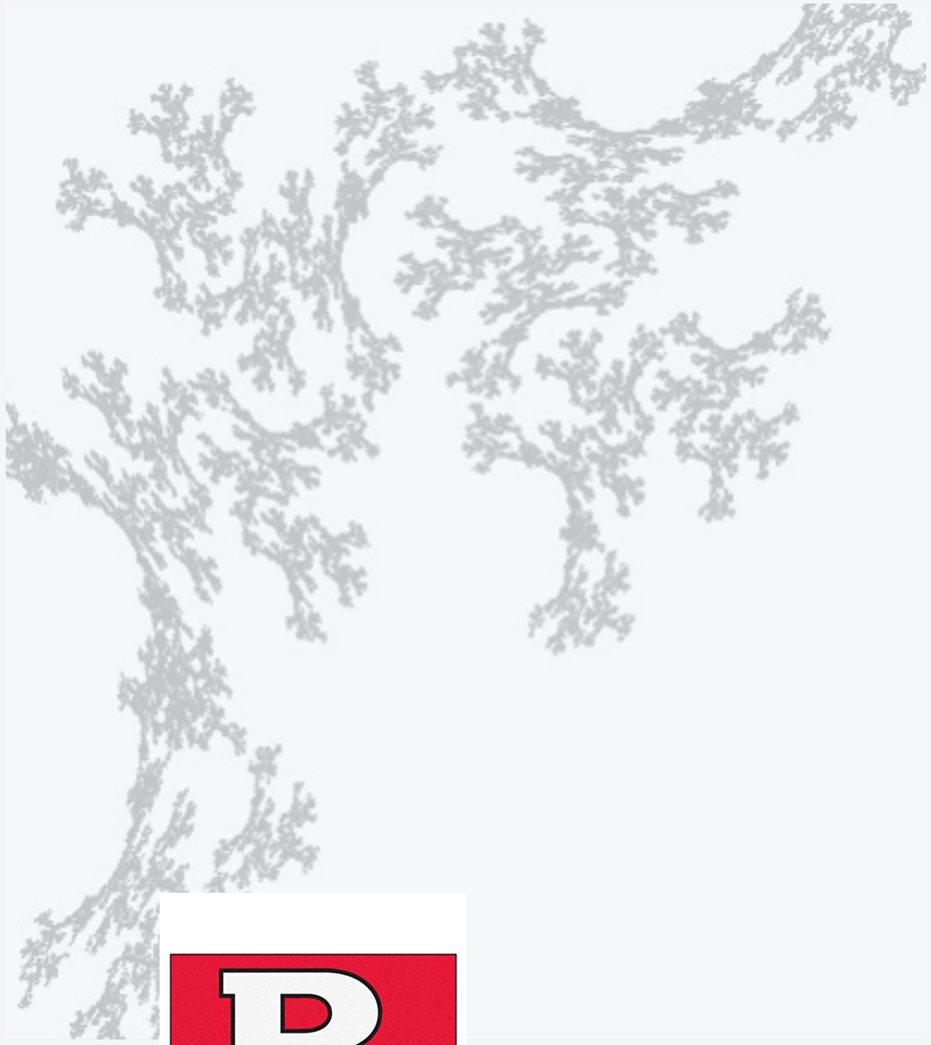
Let d be the number of divisors of n

Total number of operations =

$$2 + (n-1) + 2(n-2) + d$$

2. Algorithm Implementation and counting

- From design to implementation
- Adding control flow
- Examples of programs
- Rules of counting
- Measuring performance



INTRODUCTION TO COMPUTER SCIENCE

Rutgers University

2. Algorithm Implementation and counting

http://introcs.cs.rutgers.edu