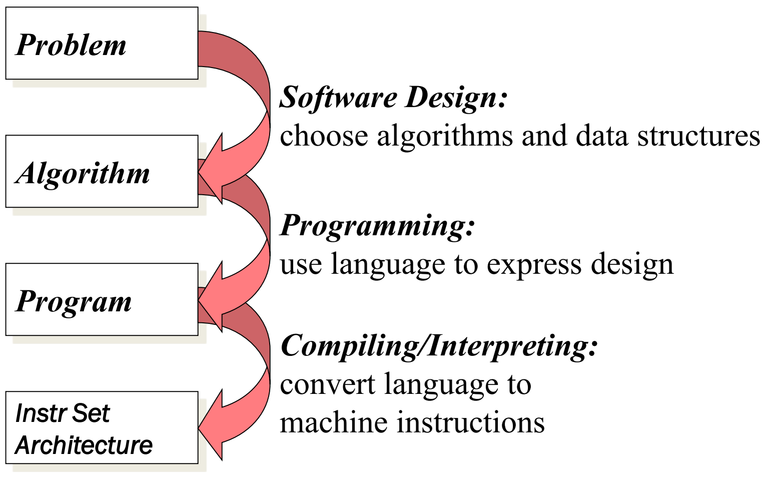
Algorithm- 1

Computer Science Department

How do we solve a problem using a computer?

A systematic sequence of transformations between layers of abstraction.



Software Design:
choose algorithms and data structures

Programming:
use language to express design

Compiling/Interpreting:
convert language to
machine instructions

Descriptions of Each Level

Problem Statement
- stated using "natural language"
- may be ambiguous, imprecise

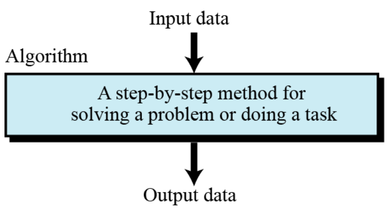
Algorithm
- step-by-step procedure, guaranteed to finish
- deterministic, definiteness, effective computability, finiteness

Program
- express the algorithm using a computer language
- high-level language, low-level language

Instruction Set Architecture (ISA)
- specifies the set of instructions the computer can perform
  data types, addressing mode

Algorithm & Pseudocode

- An **algorithm** is a *procedure or formula for solving a problem*.



- **Pseudocode** is a kind of structured English for describing algorithms. It allows the designer to focus on the logic of the algorithm without being distracted by details of language syntax.

Example

- Let's say that you have a friend arriving at the airport, and your friend needs to get from the airport to your house. Here are ***three*** different algorithms that you might give your friend for getting to your home:

Example Cont.

1. The taxi algorithm:
   - Go to the taxi stand.
   - Get in a taxi.
   - Give the driver my address.

Example Cont.

2. The call-me algorithm:

- When your plane arrives, call my cell phone.
- Meet me outside baggage claim.

Example Cont.

3. The bus algorithm:

- Outside baggage claim, catch bus number 70.
- Transfer to bus 14 on Rukab Street.
- Get off on Jerusalem street.
- Walk two blocks north to my house.

Common Action Keywords

- **Input**: READ , OBTAIN, GET
- **Output**: PRINT, DISPLAY, SHOW
- **Compute**: COMPUTE, CALCULATE
- **Initialize**: SET
- **Add one**: INCREMENT

Types of Algorithm Operations

1. Sequential

2. Conditional

3. Iterative

Sequential

❑ Computation operations

Example:

Set *the value* of “variable” to “value” or “arithmetic expression”

❑ Variable

Named storage location that can hold a data value

Sequential

❑Input operations

❖To receive data values from the user.

Example

Get a value for r, the radius of the circle

❑Output operations

❖To send results to the screen for display.

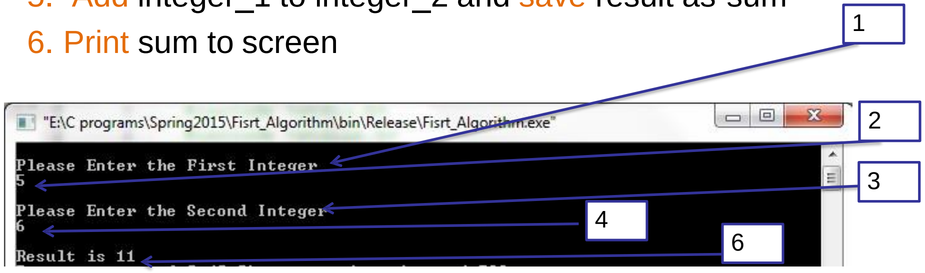
Example

Print the value of Area

Sequential – Example 1

- **Write an algorithm to find and print the sum of two integers ?**

1. Ask user to enter first integer
2. Read the integer and save as integer_1
3. Ask user to enter the second integer
4. Read second integer and save as integer_2
5. Add integer_1 to integer_2 and save result as sum
6. Print sum to screen


"E:\C programs\Spring2015\Fisrt_Algorithm\bin\Release\Fisrt_Algorithm.exe"
1
2
3
4
6

```
Please Enter the First Integer
5

Please Enter the Second Integer
6

Result is 11
```

Sequential – Example 2

- **Write an algorithm to find and print the area of rectangle.**

1. Ask user to enter the height of rectangle.
2. Read height and save as rectangle_height.
3. Ask user to enter the width of rectangle.
4. Read width and save as rectangle_width.
5. Multiply rectangle_heigh by rectangle_width and save the result as area.
6. Display area.

Sequential – Example 3

- **Write an algorithm to reverse any "*two digits number*".**

1. Ask user to enter two digits number.
2. Read number and save as num.
3. Divide num by ten and save result as tens.
4. Divide num by ten and save remainder as rem.
5. Multiply rem by ten and save the result as rev.
6. Add tens to rev.
7. Print rev.

Suppose **num**=12
tens=num /10 =12/10→tens=1
rem=num%10=12%10→rem=2
rev=rem*10=2*10→rev=20
rev=rev+tens=20+1→rev=21

Sequential – Example 3 – cont.

Suppose num=**12**

tens=num /10 =12/10 →tens=1
rem=num %10 =12%10 →rem=2

rev=rem*10 =2*10 →rev=20
rev=rev+tens =20+1

→rev=**21**

Sequential – Example 4 (num = 4562)

Write an algorithm to reverse any "*four digits number*".

Inititalization num = 4562, rev = 0

rev = rev *10 + num%10 = **2**
num = num/10 = **456**

rev = rev *10 + num%10 = 20 + 6 = **26**
num = num/10 = **45**

rev = rev *10 + num%10 = 260 + 5 = **265**
num = num/10 = **4**

rev = rev *10 + num%10 = 265 + 4 = **2654**
num = num/10 = **0**

Return rev

Conditional

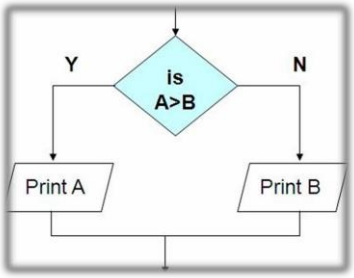
- Selection logic
- Case

Conditional

- Ask questions and ***choose alternative actions*** *based on the answers*.

Example

```
if A is greater than B then
        print A
else
        print B
end if
```



Conditional – More Choices

ELSE keyword is optional

IF condition THEN
    Sequence
END IF
	IF condition1 THEN
	    Sequence 1
	 ELSE IF condition2 THEN
	    Sequence 2
	 ELSE IF condition3 THEN
	    Sequence 3
	 ELSE
	    Sequence 4
	END IF

Conditional – Operators

Logical Operators :	Relational Operators :
• AND • OR	• Greater than • Greater than or equal • Smaller than • Smaller than or equal • Equal • Not Equal

Conditional - Example 1

Write an algorithm to print passed or failed based on the student grade.

```
1. Ask user to enter student grade.
2. Read grade and save as student_grade.
3. If student_grade greater than or equal sixty then
   print "passed"
else
   print "failed"
end if
```

Conditional - Example 2

Write an algorithm to find and print the maximum element of a set of 3 integers.

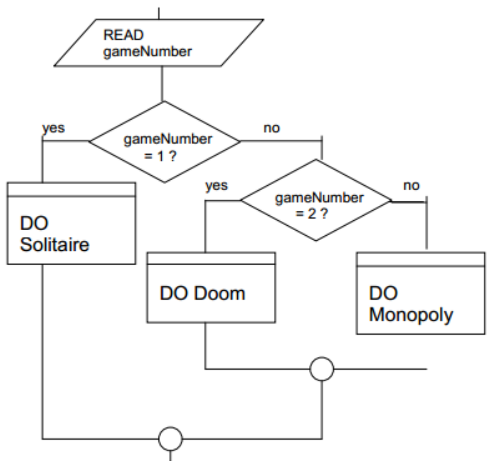
```
1. Ask user to enter the first integer.
2. Read number and save as first_integer.
3. Ask user to enter the second integer.
4. Read number and save as second_integer.
5. Ask user to enter third integer.
6. Read number and save as third_integer.
```

```
7. Let max equal to the first_integer.
8. If max less than second_integer then
       set max to second_integer
   end if
9. If max less than third_integer then
       set max to third_integer
   end if
10. Print "the maximum integer is" max
```

Nested If – Example 1

We wanted to put a little menu up on the screen: 1. Solitaire 2. Doom 3. Monopoly
The user selects which game to play. How would we activate the correct game?

```
READ gameNumber
IF gameNumber = 1
        DO Solitaire
ELSE
        IF gameNumber = 2
                DO Doom
        ELSE
                DO Monopoly
        ENDIF
ENDIF
```



Nested If – Example 2

Write an algorithm to find and print the smallest of three given numbers (assume all numbers are different).

1. Ask user to enter first number
2. Read number and save as num1
3. Ask user to enter second number
4. Read number and save as num2
5. Ask user to enter third number
6. Read number and save as num3

Rules for logical And operations		
T	T	T
T	F	F
F	T	F
F	F	F

```
7. If (num1 smaller than num2) and (num1 smaller than num3) then
        print num1 "is the smallest"
else
        If (num2 smaller than num1) and (num2 smaller than num3) then
            print num2 "is the smallest "
        else
            print num3 "is the smallest "
  end if
```

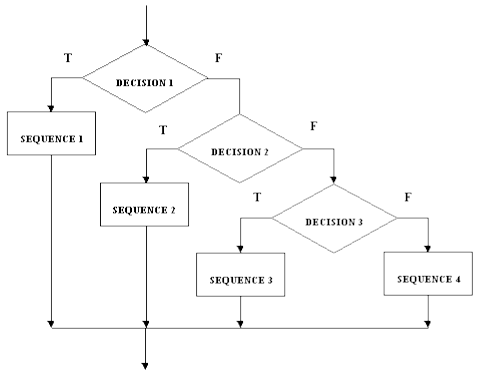
Nested If – Example 3

Write an algorithm to *read a number x and display its **sign***.

```
Ask user to enter a number
Read number and save as X
If (x is greater than zero)  then
        print x  "is positive"
Else
        if  (x is equal zero)  then
            print x "is zero"
        else
            print x "is negative"
end if
```

Nested If – Example 4

Write an algorithm that will input student **average**. If the average is greater than or equal to **60** and less than or equal to **70**, the algorithm should display “Passed”. If it is greater than 70 and less than or equal to **80**, print “Good”. If it is greater than 80 and less than **90**, print “Very good”. If it is greater than or equal 90 , print “Excellent”. If it is less than 60 the prints “Fail”.



Nested If – Example 4- Cont.

```
1. Ask user to enter  student average
2. Read average and save as ag
3. If ag is greater than or equal to sixty and ag is less than or equal to seventy then
       print “Pass”
   else
       if ag is greater than seventy and ag is less than or equal to eighty then
           print “Good”
       else
           if ag is greater than eighty and ag is less than ninety
               then print “Very good”
           else
               if ag is greater than or equal ninety then
                   print “Excellent”
               else
                   print “Fail”
   end if
```

Conditional – Case Statement

- A *multiway branch* based on conditions that are mutually exclusive
- **Three keywords**: CASE OF, OTHERS, and ENDCASE
- Conditions are used to indicate the various alternatives

General Form:

```
CASE expression OF

        condition 1 : sequence 1
        condition 2 : sequence 2
        ...
        condition n : sequence n
        OTHERS:
        default sequence

ENDCASE
```

Notice:

- The **OTHERS** clause with its default sequence is optional.
- **Conditions** are normally numbers or characters

Case Statement - Examples

```
CASE  Title  OF
        Mr     : Print "Mister"
        Mrs    : Print "Missus"
        Miss   : Print "Miss"
        Ms     : Print "Mizz"
        Dr     : Print "Doctor"
ENDCASE
```

```
CASE  grade  OF
        A      : points = 4
        B      : points = 3
        C      : points = 2
        D      : points = 1
        F      : points = 0
ENDCASE
```