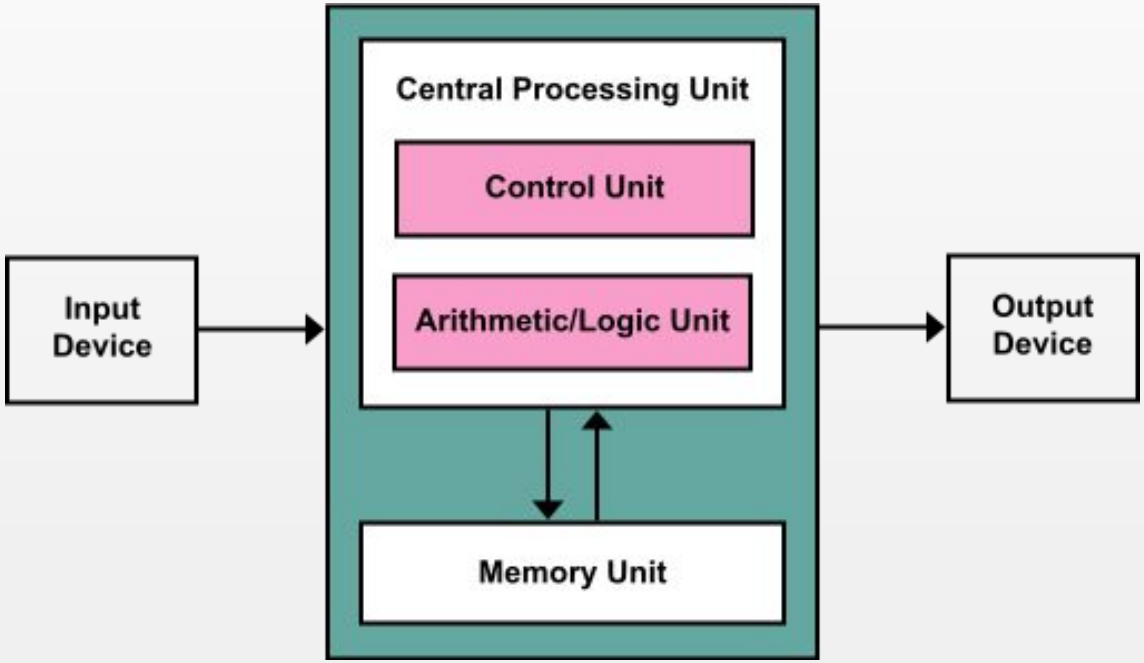
Introduction to Programming

Algorithm and Flowchart

What is in a Computer?

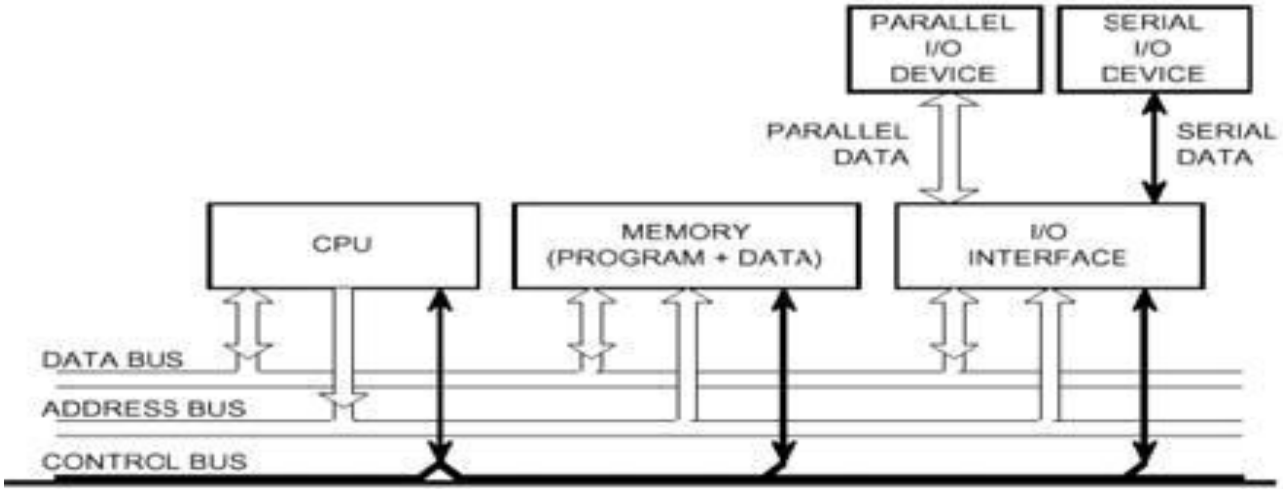
- **Computer: Hardware + Software**
- **Hardware: Input, Output, CPU (CU, ALU, Memory)**
- **Software: System Software and Application Software**

Computer Architecture



von Neumann Machine Architecture

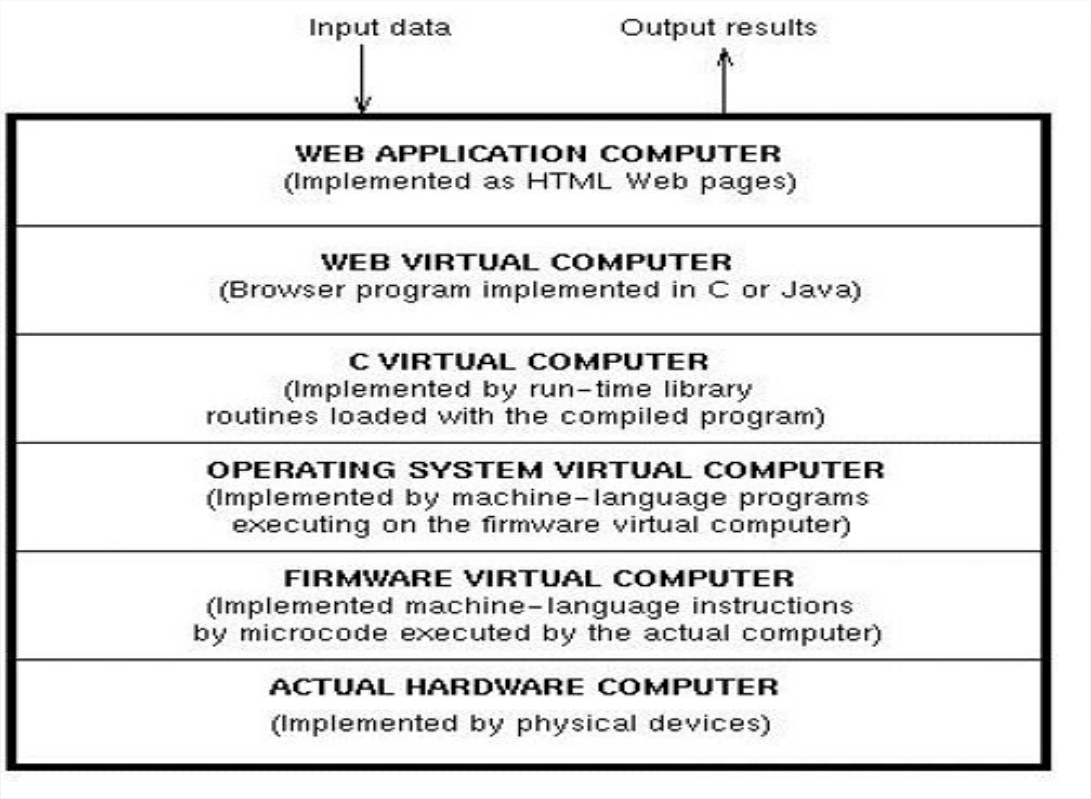
John von Neumann, 1903-1957



Important Features

- Stored program.
- Separation of processing from storage.
- Predefined instruction set.
- Shared buses.
- Sequential architecture.
- Control flow vs. data flow.

Software Hierarchy



Basic Functions of a Computer

- Input Data
- Process Data
- Output Data
- Store Data

Input Data

- **Feeding information into a computer**
  - Symbols – letters, words, numbers, etc.
  - Pictures (using a camera)
  - Sounds (using a microphone)
- **Common forms of *input***
  - Keyboard
  - Mouse
- **Often involves converting analog to digital**

Process Data

- **Analyze data, or use it in a computation.**
- **Perform some action based on data.**
- **Generate new data, or change existing data.**
- **Processing is done in the CPU.**
- **Processing is managed by a computer program.**
- **Basic Terms:**
  - *Program* – a sequence of instructions (with data) to accomplish a certain task
  - *Process* – a program in execution
  - *Processor* (CPU) - device where a program gets executed

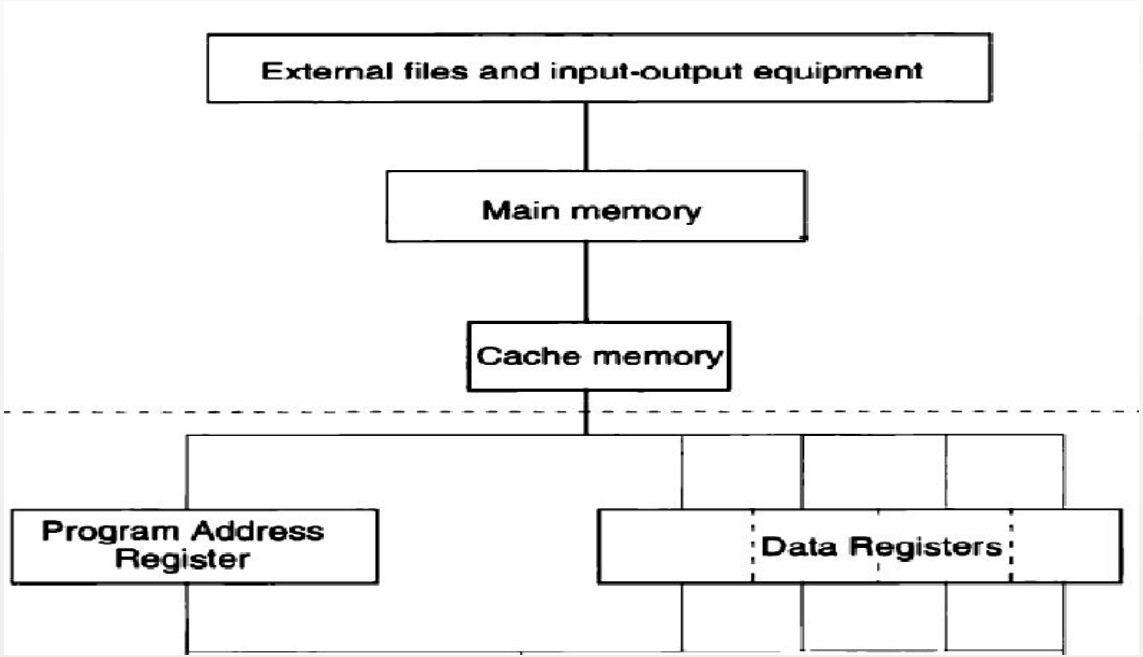
Output Data

- **Common forms of *output***
  - Monitor
  - Printers
  - Speakers (music and Sound)
  - File
- **Often involves converting digital signals to analog signals**

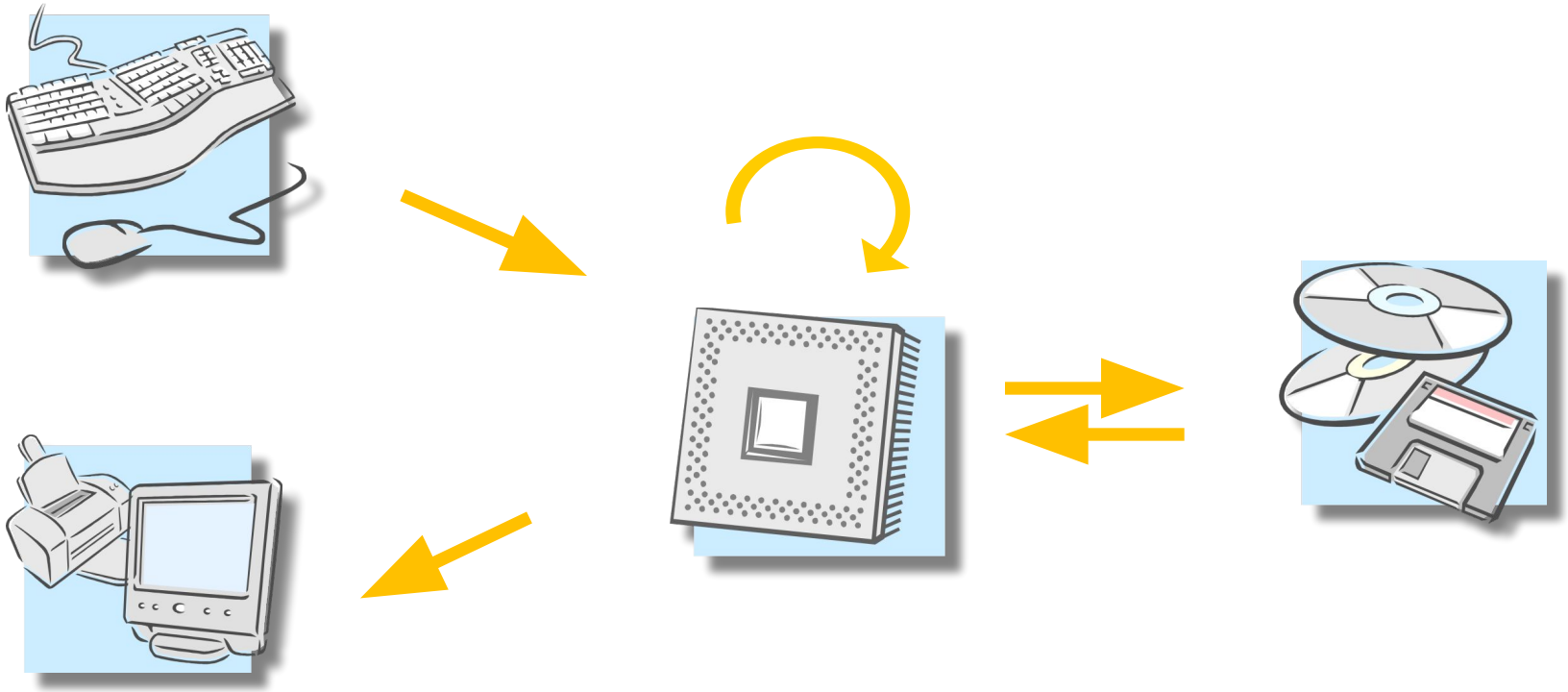
Store Data

- **Save data on a device for later use.**
- **Data is always stored in digital form.**
- **Common Forms:**
  - Memory
  - CD ROM
  - Hard Disk
  - Flash Disk
  - Cloud

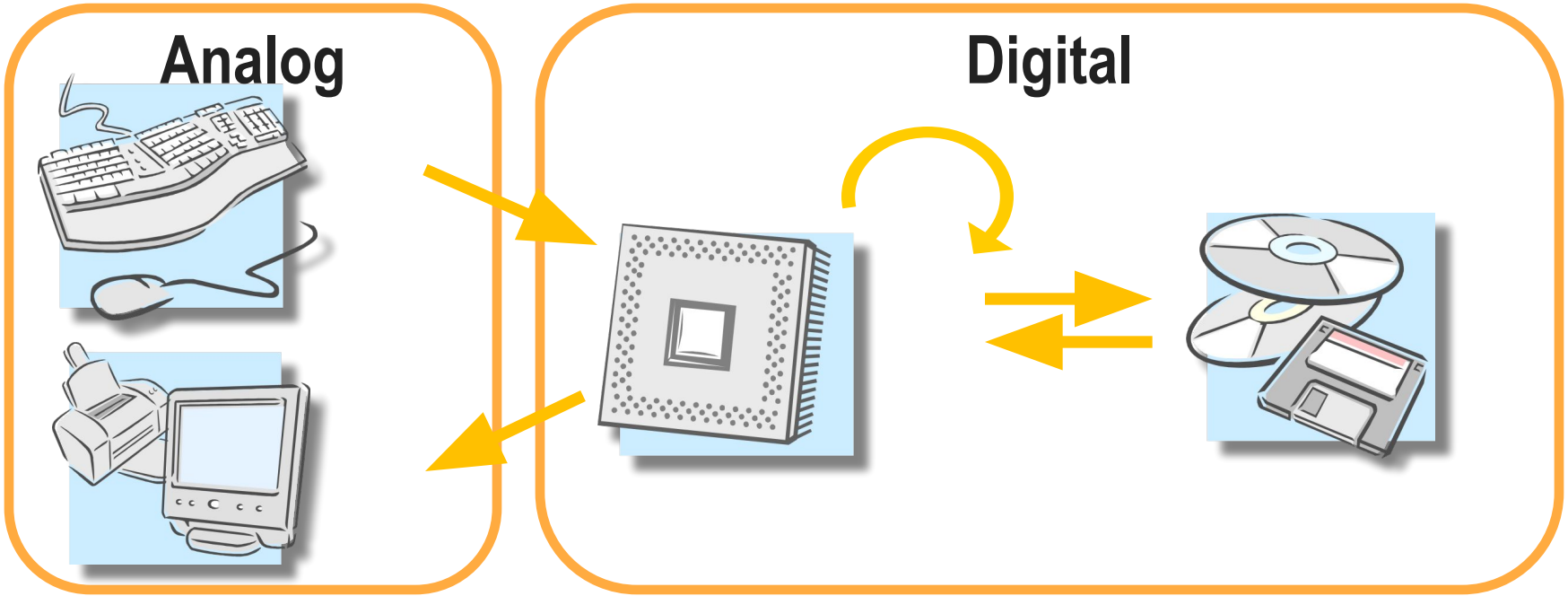
Memory Hierarchy

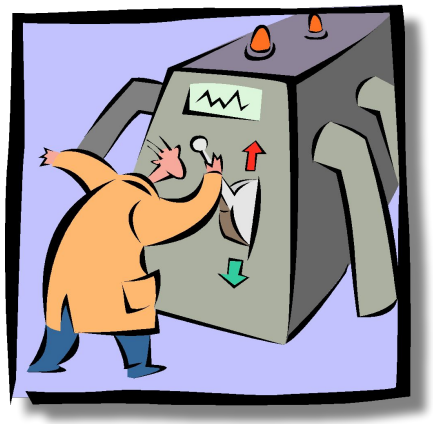


Functions of a Computer



Functions of a Computer





Computer Categories

Different Types of Computers

Computer Categories

- **Four categories**
  - Personal computers (Notebooks & PCs)
  - Workstation computers
  - Server computers
  - Supercomputers
- **Computers can be categorized by**
  - function
  - size
  - performance
  - cost

Programming Basics

Programs, Coding, Engineers & More

Programming Languages

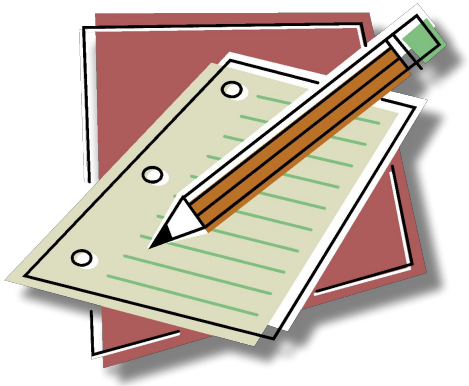
- **Language**
  - series of symbols & words that form a meaningful pattern
  - This is true of natural languages such as English, Turkish, Spanish, Hindi, Arabic, etc...
- **Programming**
  - language used to write programs to be executed in computers
  - there are many different programming languages

Popular Programming Languages

- C, C++, C#, Objective-C
- Java
- JavaScript
- Python
- Basic
- HTML
- PHP
- GO
- SQL
- ……

Algorithms

- **A sequence of steps for carrying out a task**
- **Examples:**
  - Attending to class
  - Making Tea
  - Withdraw money at ATM
- **Capture “the essence” in solving a problem**
  - can be implemented into a computer program using some programming language



Pseudocode

Describing an Algorithm with words

What is Pseudocode?

- **Description of an algorithm's logic**
  - simpler than spoken English
  - looks like a program, but easier to read

- **It is often useful**
  - to express some piece of the solution
  - without having to worry about every detail

Area of a Circle Pseudocode

Program: area of a circle

1. Get the value for the radius
2. Calculate the area (pi x radius^2)
3. Output radius and area

When we ask an algorithm, you will write a pseudocode!

Procedural/Structural Programming

- **Traditional approach to programming**
- **Programs**
  - a sequence of instructions (statements)
  - statements run in order – from first to last
  - repetition is performed with “looping”
  - decisions are described as “if – then – else”

Example Algorithms

Example 1: Write an algorithm to find the average of two numbers.

Description of the alg.

1. **Start**
2. Read two numbers
3. Add the two numbers
4. Divide the result by 2
5. Print the result
6. **End**

Pseudocode of the alg.

1. **Start**
2. Read two numbers in x and y
3. Set w = x + y
4. Set z = w / 2
5. Print z
6. **End**

Analyzing the Problem

- To write an algorithm, we first analyze the problem:
  - What is the input(s)
  - What is the output(s)
  - What is the relation between the input and output
    - How we can reach from the input to the output

Algorithms

Example 2: Write an algorithm to take your birth year and print your age

1. **Start**
2. Read birth year in x
3. Set a = 2018 - x
4. Print a
5. **End**

Analyze the Problem:

Input: year

Output: age

Relation: 2018-year

Algorithms

Example 3: Write an algorithm to take your age and print it in seconds.

1. **Start**
2. Read age in a
3. Set s = a × 365 × 24 × 60 × 60
4. Print s
5. **End**

Algorithms

Example 4: Write an algorithm to take the sides (length and width) of a rectangle and calculate its perimeter and area

1. **Start**
2. Read width and length in w and l
3. Set a = w × l
4. Set p = 2 × (w + l)
5. Print “perimeter is p and area is a”
6. **End**

Algorithms

Example 5: A company pays 7500 TL for each of his developers. Take the salary increment percentage and calculate how much money yearly extra cost is needed for the promotion.

1. **Start**
2. Read increment percentage in p
3. Set c = 2 × 12 × 7500 × p / 100
4. Print c
5. **End**

Alg: Try Yourself!

Example: Write an algorithm to takes two numbers and swap their values (w/o temp var)

Conditions in Algorithms

Most of the times we need to do different actions in different cases.

If it is raining then take an umbrella

otherwise take sunglasses

Conditions in Algorithms

Example 6: Write an algorithm to take a number and prints if it is odd or even.

1. **Start**
2. Read number in p
3. If p modulo 2 = 0 then
4. 	print even
5. Else
6. 	print odd
7. **End**

We write the commands inside the if or else body with some indent

Conditions in Algorithms

Example 7: Write an algorithm to take two numbers and print the minimum and maximum number.

1. **Start**
2. Read numbers in x and y
3. If x > y then
4. print “maximum is x”
5. print “minimum is y”
6. Else
7. print “maximum is y”
8. print “minimum is x”
9. **End**

Conditions in Algorithms

Example 8: Write an algorithm to take the value for x and calculates 1/(x^2+x+3). Check the divide by zero.

1. **Start**
2. Read a number in x
3. Set divider = x^2+x+3
3. If divider = 0 then
4. print “Division is not possible”
5. Else
4. Set Calc = 1 / divider
4. print “The result is Calc”
7. **End**

Conditions in Algorithms

Example 9: Write an algorithm to take the required input of an employee and calculates the net salary based on the following rules: (Let min salary = 2000 TL)

1) Tax: for minimum salaries take 5% tax, for salaries upto three times minimum salary take 7% tax and for salaries more than that take 10% tax.

2) Insurance: take 14%

3) Additional: for each child add 100 TL

Analyze the problem!

Conditions in Algorithms

1. **Start**
2. Read GrossSal and NumChild
3. If GrossSal = 2000
4. Set Tax = GrossSal * 5 / 100
5. If 2000 < GrossSal < 6000
6. Set Tax = GrossSal * 7 / 100
7. If GrossSal > 6000
8. Set Tax = GrossSal * 10 / 100
9. Set Ensur = GrossSal * 14 / 100
10. Set ChildExtra = NumChild * 100
11. Set Total = GrossSal - Tax - Ensur + ChildExtra
12. print “Net Salary is:” Total
13. **End**

Conditions in Algorithms

Example 10: Write an algorithm to take three numbers and print the maximum of them

1. **Start**
2. Read x, y, z
3. If x >= y then
4. Set Max = x
5. Else
6. Set Max = y
7. If z >= Max
8. Set Max = z
9. print “The maximum number is Max”
10. **End**

Think about more numbers - need for an iteration!!!

Iteration or Loops in Algorithms

In many cases we need to repeat some actions for a specific number of times or until a specific condition

- Conditions are important part of loops
- Loops may need a counter to control the number of iterations

Loops in Algorithms

Example 11: Write an algorithm to print integers 1 to N

1. **Start**
2. Read N
3. Count = 1
4. Print Count
3. If Count < N then
4. Count = Count + 1
5. Goto Line 4
10. **End**

Tracing an Algorithm

- To find possible bugs in the algorithm we debug it
- One way to debug the algorithm is tracing it with some input
- Trace:
  - Follow the steps from the start to the end
  - Execute the steps with the given input
  - Keep the track of variables values
  - Check the results in each step until the end of the algorithm

Loops in Algorithms

Example 12: Write an algorithm to take 15 integer numbers and print the maximum of them (extended version of Example 10)

1. **Start**
2. Set Max = 0, Count = 1
3. Read N
3. If N > Max then
4. Set Max = N
5. If Count < 15 then
6. Count = Count + 1
7. Goto Line 3
9. print “The max number is Max”
10. **End**

Trace the algorithm for:

14, 3, 17, 4, 20, …

Max	Count	N (input)	output
0	1	14	
14	2	3	
	3	17	
17	4	4	
	5	20	
20	6	...	

Loops in Algorithms

Example 13: Write an algorithm to take integer N print N!

1. **Start**
2. Read N
3. Set Fact = 1, Count = 1
4. Set Fact = Fact * Count
5. Set Count = Count + 1
6. If Count <= N then
7. Goto Line 4
8. print “The Factorial of N is Fact”
9. **End**

Trace the algorithm for: 5

Loops: Try Yourself!

Example: Write an algorithm to takes an integer number and print its reverse number and number of its digits

(e.g.: 263 => 362 and 3)

Nested Loops in Algorithms

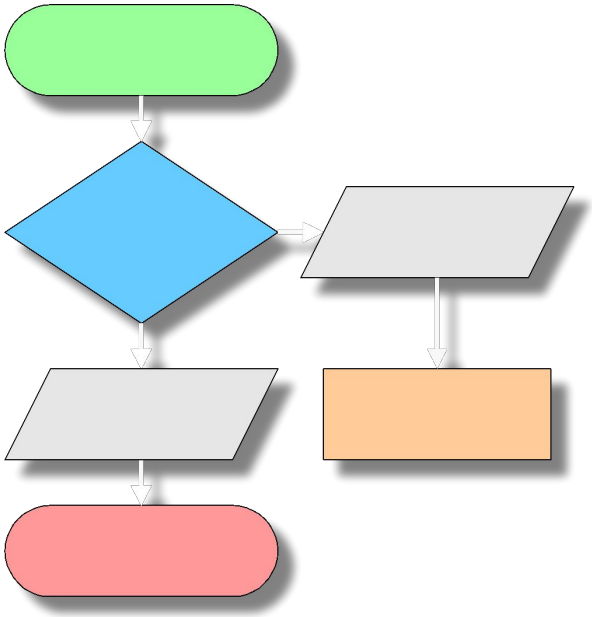
Example 14: Write an algorithm to print the multiplication table

1. **Start**
2. Set row = 1, col = 1
3. Print row * col
4. If col < 10 then
5. Set col = col + 1
6. Goto Line 3
7. Else
8. Print newline
09. If row < 10 then
10. row = row + 1
11. col = 1
12. Goto Line 3
13. **End**

Nested Loops: Try Yourself!

Example 14: Write an algorithm to take N and print e value using the following series:

$$e = \sum_{i=0}^{N} 1/i! = 1/0! + 1/1! + 1/2! + 1/3! + \ldots$$

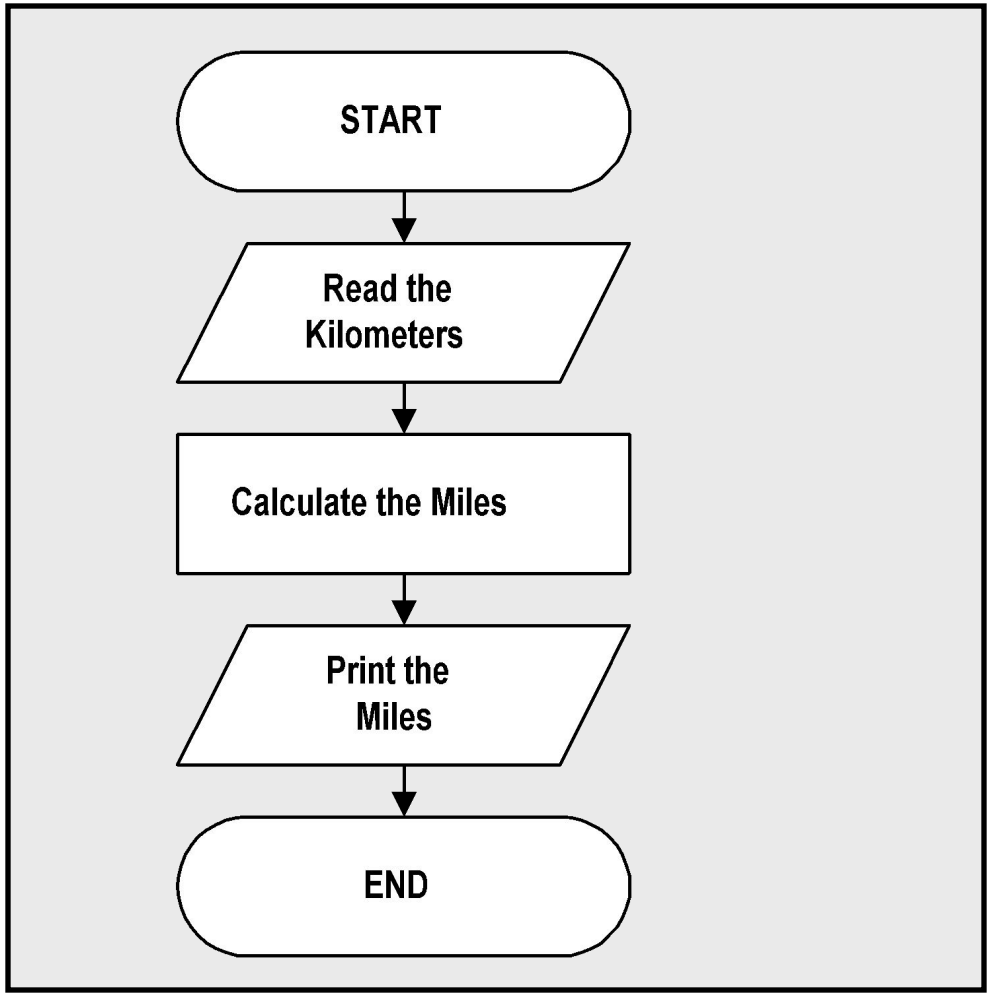


Flow Charts

Graphically Representing Algorithms

Flow Chart Overview

- **Graphical representation**
  - each step is a *shape* (box, circle, …)
  - useful for conceptualizing an *algorithm*
  - easy to understand and visualize
- **Used to document how an algorithm was designed**



Input / Output

Process Data

Conditions

Start / End

- **Indicates the start and end of an algorithm**
- **Represented by a rectangle with rounded sides**
- **There are typically two:**
  - one to start the flowchart
  - one to end the flowchart

Input / Output

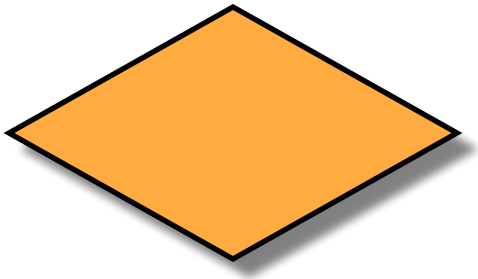
- **Indicates data being:**
  - inputted into the computer
  - outputted to the user
- **Represented by a parallelogram**
- **Flowcharts can have many of this shape**

Processes

- **Indicates data:**
  - being processed
  - also called "calculations"
- **Represented by a rectangle**
- **The most common shape in a flowchart**

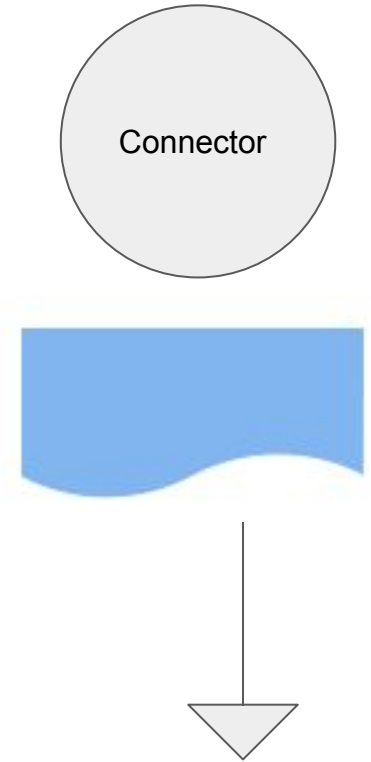
Decisions

- **Indicates a conditional branch**
  - describe a condition or a question
  - Has more than one outgoing branch, depending on the outcomes to the condition/question
- **Represented by a diamond**



Additional Symbols

- **Connectors**
  - (to extend large flowcharts in different pages
- **Print document**
  - Kind of output
- **Links**
  - To link the symbols and show the control flow



Even More Symbols

Terminator
Indicates the beginning or end of a program flow in your diagram.

Process
Indicates any processing function.

Decision
Indicates a decision point between two or more paths in a flowchart.

Delay
Indicates a delay in the process.

Data
Can represents any type of data in a flowchart.

Document
Indicates data that can be read by people, such as printed output.

Multiple documents
Indicates multiple documents.

Subroutine
Indicates a predefined (named) process, such as a subroutine or a module.

Preparation
Indicates a modification to a process, such as setting a switch or initializing a routine.

Display
Indicates data that is displayed for people to read, such as data on a monitor or projector screen.

Manual input
Indicates any operation that is performed manually (by a person).

Manual loop
Indicates a sequence of commands that will continue to repeat until stopped manually.

Loop limit
Indicates the start of a loop. Flip the shape vertically to indicate the end of a loop.

Stored data
Indicates any type of stored data.

Connector
Indicates an inspection point.

Off-page connector
Use this shape to create a cross-reference and hyperlink from a process on one page to a process on another page.

Off-page connector

Off-page connector

Off-page connector

Or
Logical OR

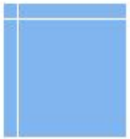
Summing junction
Logical AND

Collate
Indicates a step that organizes data into a standard format.

Sort
Indicates a step that organizes items list sequentially.

Merge
Indicates a step that combines multiple sets into one.

Database
Indicates a list of information with a standard structure that allows for searching and sorting.

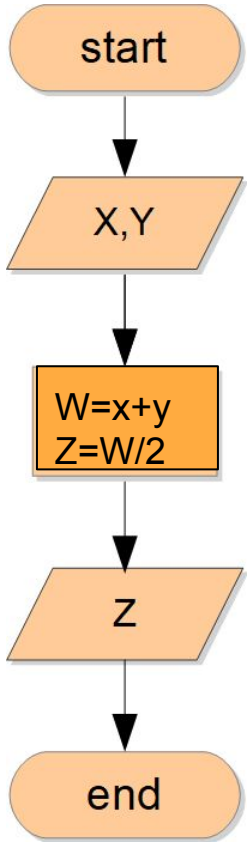


Internal storage
Indicates an internal storage device.

Flow Chart Example

Example: Draw a flowchart to find the average of two numbers.

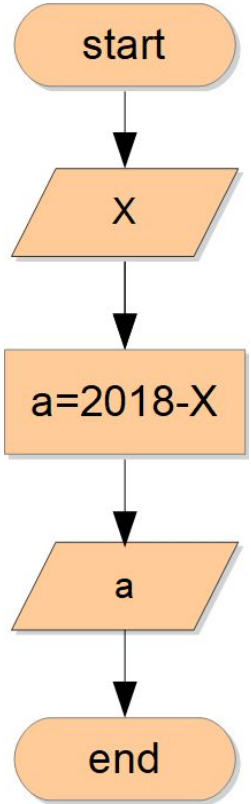
1. **Start**
2. Read two numbers in x and y
3. Set w = x + y
4. Set z = w / 2
5. Print w
6. **End**



Flow Charts

Example: Draw a flowchart to take your birth year and print your age

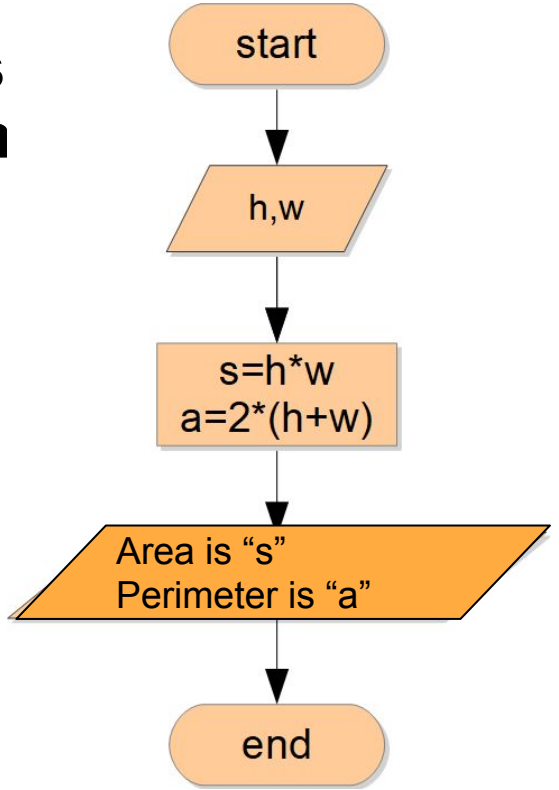
1. **Start**
2. Read birth year in x
3. Set a = 2018 - x
4. Print a
5. **End**



Flow Charts

Example: Draw a flowchart to take the sides a rectangle and calculate its perimeter an area.

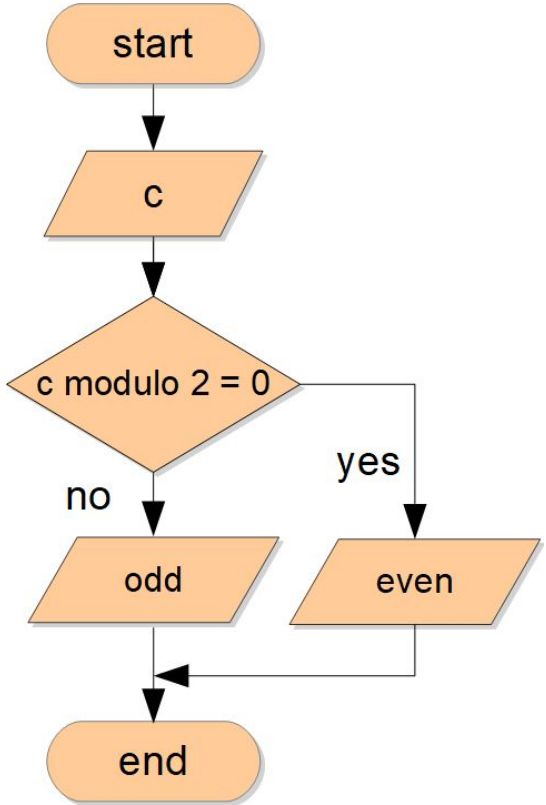
1. **Start**
2. Read width and height in w and h
3. Set s = h × w
4. Set a = 2 × (h + w)
5. Print “surface is s and area is a”
6. **End**



Flow Charts

Example: Draw a flowchart to take a number and prints if it is odd or even.

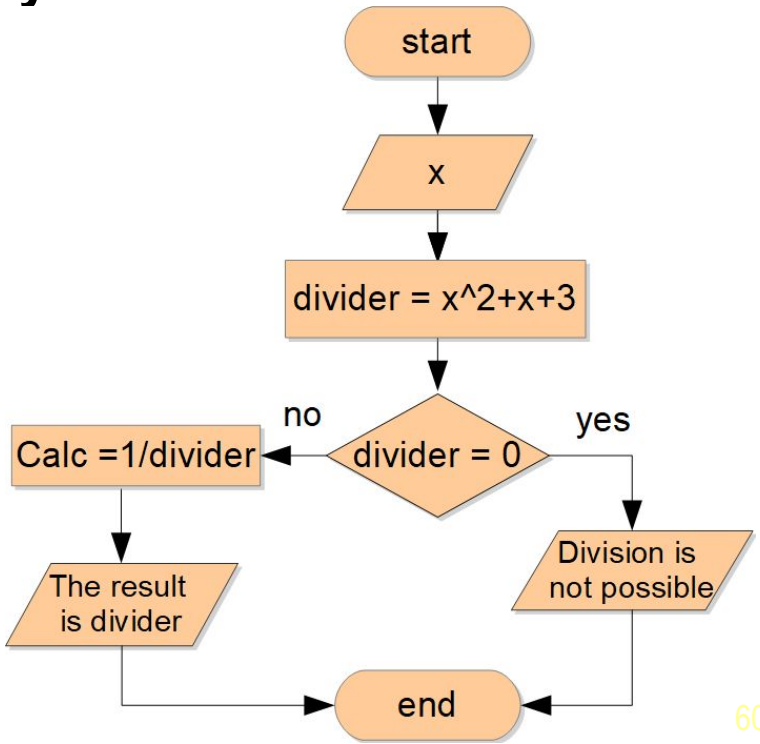
1. **Start**
2. Read number in c
3. If c modulo 2 = 0 then
4. print even
5. Else
6. print odd
7. **End**



Flow Charts

Example: Draw a flowchart to take the value for x and calculates 1/(x^2+x+3). Check the divide by zero.

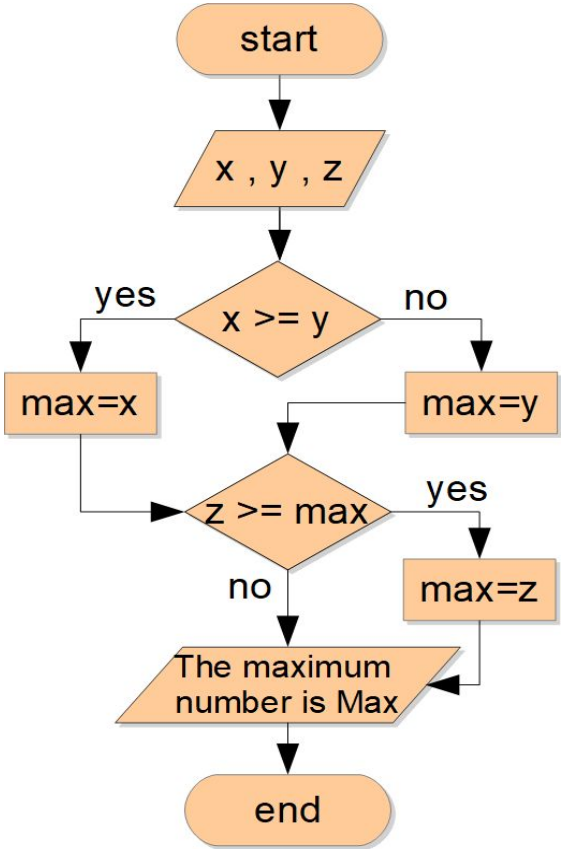
1. **Start**
2. Read a number in x
3. Set divider = x^2+x+3
3. If divider = 0 then
4. print “Division is not possible”
5. Else
4. Set Calc = 1 / divider
4. print “The result is divider”
7. **End**



Flow Charts

Example: Draw a flowchart to take three numbers and print the maximum of them

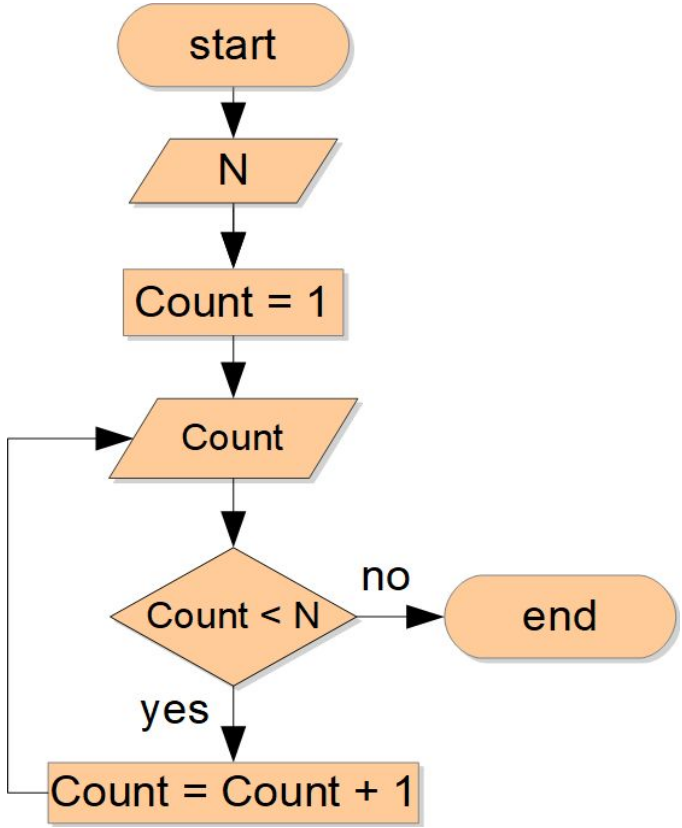
1. **Start**
2. Read x, y, z
3. If x >= y then
4. Set Max = x
5. Else
6. Set Max = y
7. If z >= Max
8. Set Max = z
9. print “The maximum number is Max”
10. **End**



Flow Charts

Example: Draw a flowchart to print integers 1 to N

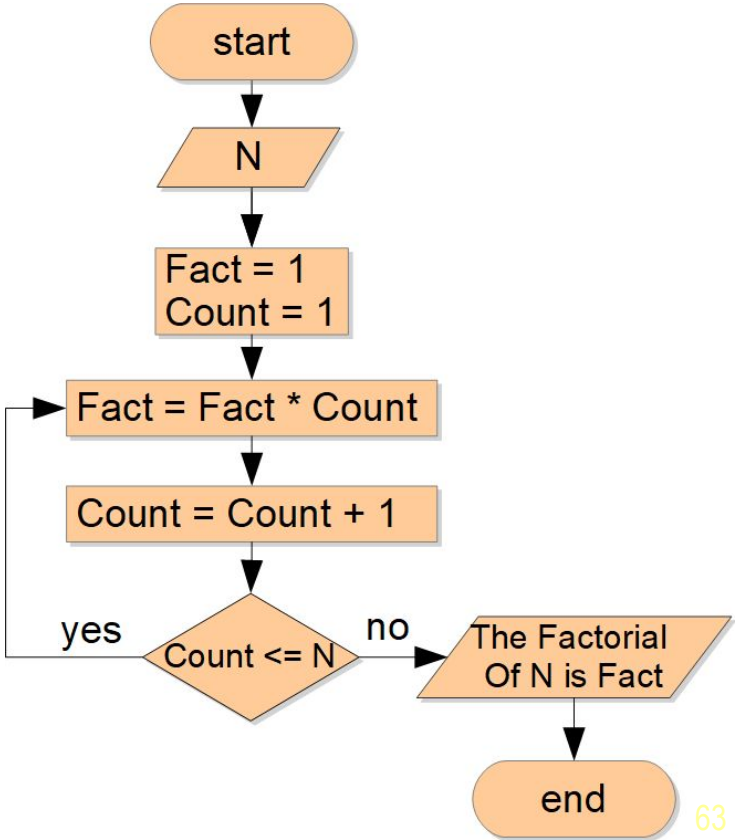
1. **Start**
2. Read N
3. Count = 1
4. Print Count
3. If Count < N then
4. Count = Count + 1
5. Goto Line 4
10. **End**



Flow Charts

Example: Draw a flowchart to take integer N print N!

1. **Start**
2. Read N
3. Set Fact = 1, Count = 1
4. Set Fact = Fact * Count
5. Set Count = Count + 1
6. If Count <= N then
7. Goto Line 4
8. print “The Factorial of N is Fact”
9. **End**

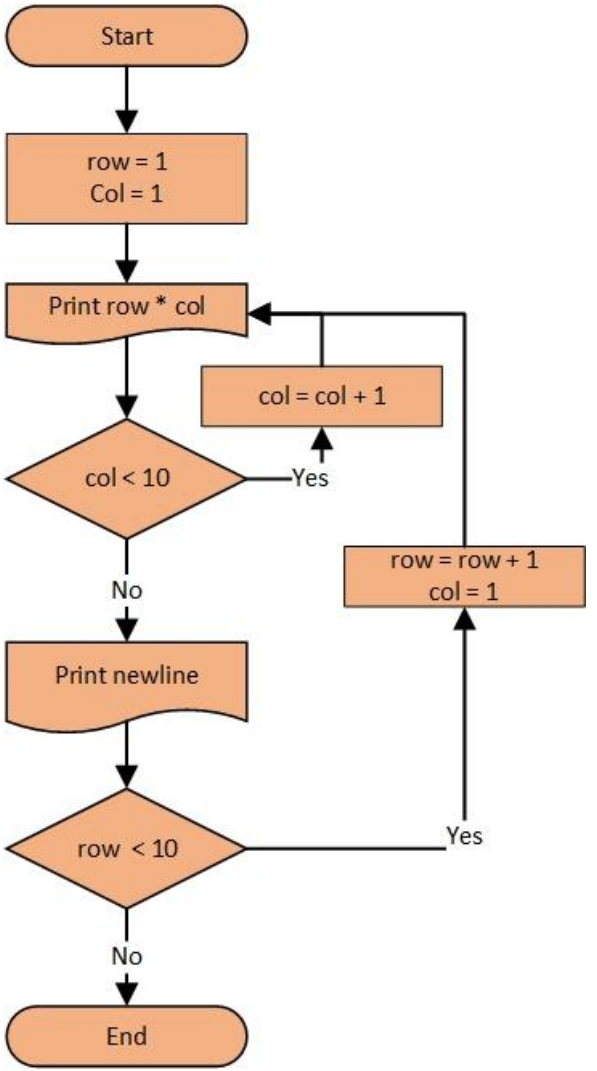


Flow Charts

Example:

Draw a flowchart to print the multiplication table

1. **Start**
2. Set row = 1, col = 1
3. Print row * col
4. If col < 10 then
5. Set col = col + 1
6. Goto Line 3
7. Else
8. Print newline
9. If row < 10 then
10. row = row + 1
11. col = 1
12. Goto Line 3
13. **End**



Try Yourself!

Example: Draw a flowchart to take N and print e value using the following series:

$$e = \sum_{i=0}^{N} 1/i! = 1/0! + 1/1! + 1/2! + 1/3! + \ldots$$

- **Analysis**
- **Trace for 5**