SAKARYA ÜNİVERSİTESİ

Problem Solving In Software Engineering

LECTURE 4

Algorithm, Pseudocode & Flowchart

Assist. Prof. Dr. Gülüzar ÇİT

Outline

- Algorithm
  - Definition
  - What should be in a good algorithm?
  - Can Algorithms be Expressed in Different Ways?
- Basic Terms & Symbols in Algorithms
  - Identifier, variable, constant, counter, query, loop
  - Arithmetic, logical and comparison operations
- Flowcharts
- Symbols in Flowcharts
- Flowchart Examples
- References

Algorithm

- can be thought as the equivalent of the word **"plan"** in real life
- expressing the solution steps of a problem in order and finite
- a set of instructions that is used to perform a certain task
- in software engineering: "the logical and symbolic description of the predicted operations for the solution of a problem"
- Abdullah Muhammad bin Musa al-Khwarizmi, a Turkish-Islam mathematician and astronomer who lived in 9th century, did a study explaining some mathematical operations such as addition, subtraction, division by two, find twice an number. This work is called as "algorismus" in Latin from Westerners. So, todays "algorithm" was used firstly.
- free from any programming language, but should be adaptive to any of them.

Algorithm…

- **What Should be in a Good Algorithm?**
  - **Efficient**
    - should not include unnecessary repetitions and should be used in other algorithms.
  - **Finite**
    - Each algorithm must consist of a start, contain a specific action step, and have an end point. It should not go into a vicious circle.
  - **Certain**
    - The action result must be precise, producing the same result with each new run.
  - **Input/Output**
    - The algorithm must have input (values to be processed) and output (result values produced as a result of the operations performed) values.
  - **Performance**
    - The aim should be to write high-performance programs, taking into account performance criteria such as hardware requirement (memory usage, etc.), runtime, etc.

Algorithm…

- **Can Algorithms be Expressed in Different Ways?**
  - writing the algorithm as text
    - the problem to be solved is written in text step by step.
  - **Pseudo Code**
    - the algorithm can also be written in pseudo-codes.
    - between the spoken language and the programming language
    - also called as half code and half text
    - the solution steps of the problem are expressed in understandable text like commands.
    - not compiled and processed as programs
    - shold be clear and understandable by others.
  - **Flowcharts**
    - The solution steps of the problem are expressed with specific geometric shapes.
  - **UML**
  - **Design Patterns**

Algorithm…

- **EXAMPLE – 1**
  - Problem: A program which calculates the square of a number entered from the keyboard and prints the result on the screen.

  - Write down the **algorithm** of the problem
  - Write down the **pseudocode** of the problem
  - Draw the **flowchart** of the problem

Algorithm…

- **EXAMPLE – 1…**
  - Algorithm:

1. START
2. Input a number ⇒ a
3. Calculate the square of the number ⇒ square = a*a
4. Print result ⇒ square
5. END

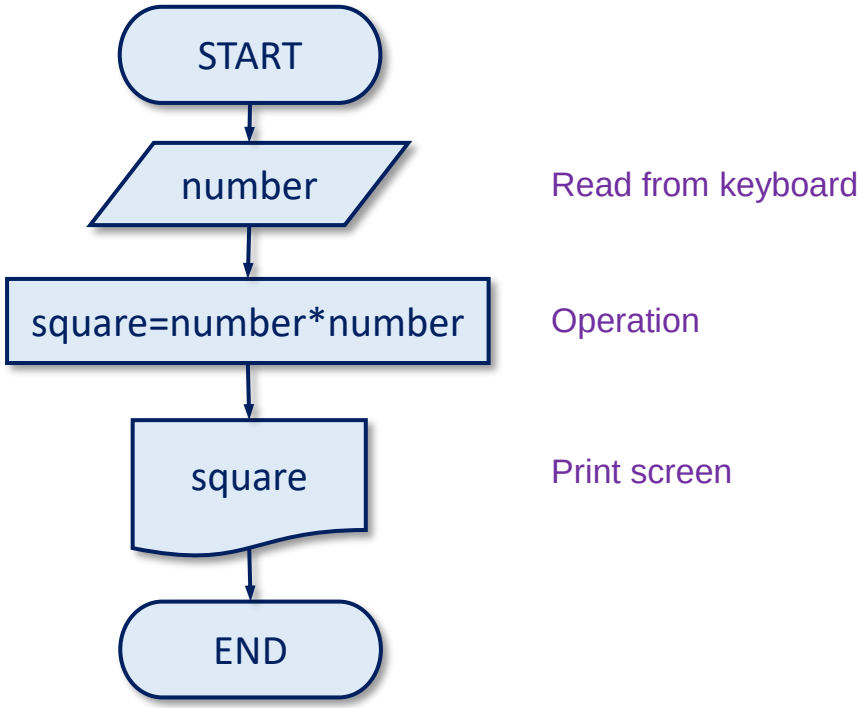
Algorithm…

- **EXAMPLE – 1…**
  - Pseudo Code:

```
1. /* Square Program */
2. START
3. cin >> a
4. square = a*a
5. cout << square
6. END
```

Algorithm…

- **EXAMPLE – 1…**
  - Flowchart:



Basic Terms & Symbols in Algorithms

- **Operator**
  - Symbols that indicate transactions, namely, have the ability to operate on data.

- EXAMPLE:
  - + , = , > , >= , !=

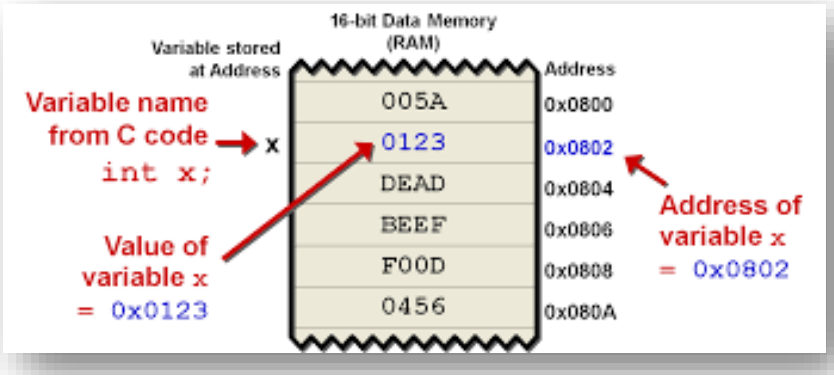
Basic Terms & Symbols in Algorithms…

- **Identifier**
  - Special words defined by the coder/programmer used to name variables, constants, classes, objects, special information types, subprograms, etc.
  - The identifier should associate the statement it will replace.

- **Rules to be followed when creating descriptive words:**
  - 26 letters of A-Z or a-z in the English alphabet can be used
  - numbers between 0-9 can be used
  - from symbols only underscores (_) can be used
  - can start with letters or underscores
  - cannot start with numbers or consist only of numbers
  - cannot be from the command or hidden words of the programming language used.

Basic Terms & Symbols in Algorithms…

- **Variable**
  - symbolic names given to the memory areas where data are stored.
  - information/memory areas that can receive/transfer different values during each program execution.
    - at any particular time, a variable will stand for one particular data, called the value of a variable
    - the value of a variable will change many times in time during a computing process.
  - naming variables is done by the programmer/coder.
  - it is important for the clarity of the program that the variable name evokes the expression it replaces.



- **Example:**
  - **name** to hold a person's name
  - **tel** to hold a person's phone

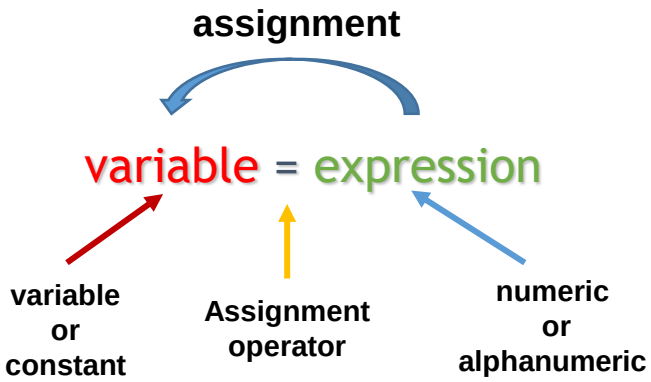
Basic Terms & Symbols in Algorithms…

- **Constant**
  - identifiers required to keep their values throughout the program
  - descriptive naming rules are also valid for constants
  - maybe a specific value or character string used explicitly in an
  - operation.

  - **Example:**
    - pi = 3.14
    - g = 9.8 m/s
    - first_letter_of_alphabet=‘A’
    - second_day_of_week = "Tuesday"

Basic Terms & Symbols in Algorithms…

- **Assignment Operation**
  - used to write data to an information field
  - used whenever you need to keep track of a value that will be needed later during the program execution



Example:

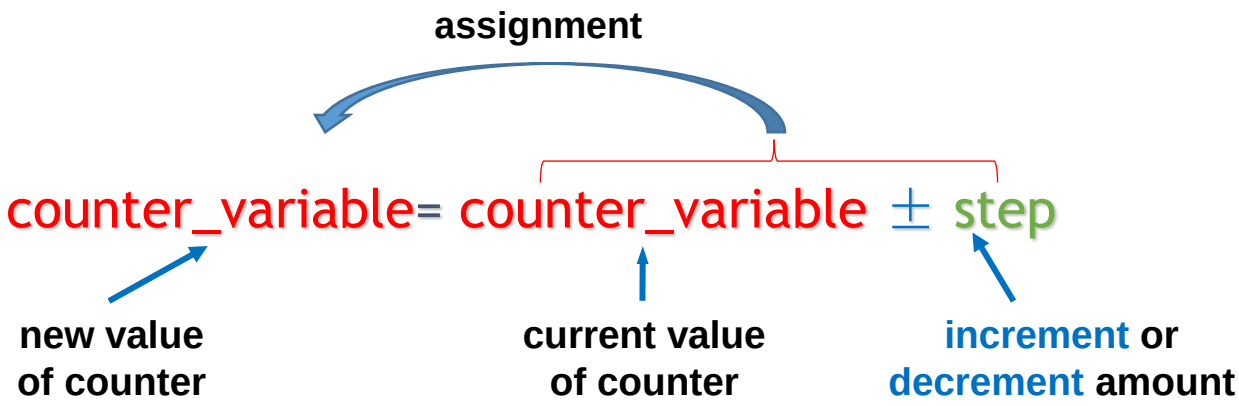
- as a numeric expressions
  - A= 2, B= 3
  - C = A + B ◄---------- C=5
- as an alphanumeric expression
  - A = “Sak” , B = “arya”
  - C = A + B ◄---------- C="Sakarya"

Basic Terms & Symbols in Algorithms…

- **Assignment Operation…**
  - Some example uses:
    - initializing a variable ⇒ **count = 0**
    - increment/decrement a **counter** variable ⇒ **count = count + 1**
    - accumulate values ⇒ **sum = sum + item**
    - capture the result of a computation ⇒ **y = 3*x + 4**
  - the assignment operation is not same of its reverse
    - i.e. **a = b** is not the same as **b = a**.

Basic Terms & Symbols in Algorithms…

- **Counter Variable**
  - to follow a certain number of requested operations
  - used when the processed/produced values need to be counted.
  - Example:
    - in determining the odd numbers in a randomly generated array, a counter is used to determine the number of odd numbers.



Example:

- count = count + 1
- i = i - 2

Basic Terms & Symbols in Algorithms…

Arithmetic Operations

Operation	Mathematics	Computer
Addition	a + b	a + b
Subtraction	a - b	a - b
Multiplication	a . b	a * b
Division	a ÷ b	a / b
Exponentiation	a^b	a ^ b

Basic Terms & Symbols in Algorithms…

Arithmetic Operations…

Mathematical Writing	Coding into Computer
a+b-c+2abc-7	*a+b-c+2*a*b*c-7*
$a+b^2-c^3$	*a+b^2-c^3*
$a-\frac{b}{c}+2ac-\frac{2}{a+b}$	*a-b/c+2*a*c-2/(a+b)*
$\sqrt{a+b}-\frac{2}{b^2-4ac}$	*(a+b)^(1/2)-2/(b^2-4*a*c)*
$\frac{a^2+b^2}{2ab}$	*(a^2+b^2)/(2*a*b)*

Basic Terms & Symbols in Algorithms…

Arithmetic Operations…

Order	Operation	Programming
1	Negativity of Numbers	-…
2	Paranthesis	(…..)
3	Arithmetic Functions	*cos*, *sin*, *log*, …
4	Exponentiation	a ^ b, *pow*, …
5	Multiplication and Division	a * b ve a/b
6	Addition and Subtraction	a + b ve a - b

- Example:
  - **Artihmetic Expression**

$$x = a.b/c + d.e^f - g$$

  - **Computer Language Expression**

$$x = a * b/c + d * e\hat{}f - g$$

7 2 3 5 4 1 6

Basic Terms & Symbols in Algorithms…

- **Comparison Operations**
  - checks which of the two data items (value, variable, etc.) is bigger or smaller
  - checks whether two data item (value, variable, etc.) are equal or not
  - Computer Language Equivalents of Comparison Operations

Symbol	Description
= or ==	equals
<> or !=	not equals
>	Bigger
<	Smaller
>= or =>	bigger or equals to
<= or =<	smaller or equals to

- Example:

```
IF A > B THEN write "A is bigger than B"
```

Basic Terms & Symbols in Algorithms…

- **Comparison Operations…**
  - values are compared directly while making numerical comparisons
    - Example: $25 > 5 \Rightarrow$ "25 is bigger than 5"
  - in alphanumeric comparisons, the comparison process is compared by starting from the first characters.
    - Example: $'a' > 'c' \Rightarrow$ "first character (a) is more ahead"

  - NOTE:
    - In character comparison operations, the comparison is made between the ASCII code counterparts of the characters, not between characters themselves.
    - For example, the ASCII equivalent of A is 65, but the ASCII equivalent of a is 97. The ASCII code difference between uppercase and lowercase letters is 32.

Basic Terms & Symbols in Algorithms…

Logical Operations

Basic Logical Operation Equivalents

Operation	Arithmetic Symbol	Description
AND	&	The result is TRUE, if all of the conditions are TRUE
OR	|	The result is TRUE, if at least one of the conditions are TRUE
NOT	!	The result is reverse of the condition (TRUE if FALSE)

Priority in Logical Operations

Order	Operation	Command
1	Operations in paranthesis	(…..)
2	NOT	!
3	AND	&
4	OR	|

Basic Terms & Symbols in Algorithms…

- **Logical Operations…**
  - **EXAMPLE:**
    - Only those who are over the age of 23 and earn the minimum salary are asked among the workers
      - There are two conditions here, and both conditions must be true.

```
IF age > 23 AND salary = basepay THEN PRINT…
```

1.CONDITION *2.CONDITION*

- *PRINT… command is processed if both conditions are met.*

Basic Terms & Symbols in Algorithms…

- **Logical Operations…**
  - **EXAMPLE:**
    - The names of the students in a class who get more than 65 points from the COMPUTER course and who get a score above 65 in any of the TURKISH LANGUAGE or ENGLISH courses are requested.
      - There are two requirements and three conditions here:
        - The first requirement is to get a grade above 65 from the computer course is a basic requirement.
        - The second requirement is to get a grade from any of the other two courses must be above 65.

```
IF computer_grade>65 AND
   (turkish_grade>65 OR english_grade>65)
THEN PRINT…
```

Basic Terms & Symbols in Algorithms…

- **Decision/Comparison Structure**
  - In algorithms, operations usually consist of sequential steps.
  - In some conditions, it may be necessary to change the order of operations and to select another operation order or to continue the program from a new operation sequence.
  - The **IF** query statement is used to change transaction sequences or query some conditions whether true or not.

  - Example:

```
IF average > 50 THEN GO…
```

CONDITION *PROCESS NUMBER*

Basic Terms & Symbols in Algorithms…

- **Decision/Comparison Structure…**
  - Example:
    - Write down the algorithm of the program that compares two numbers entered from keyboard.

```
1 START
2 READ x,y
3 IF x>y THEN PRINT "x is bigger than y", GO STEP 6
4 IF x<y THEN PRINT "y is bigger than x", GO STEP 6
5 PRINT "x equals to y"
6 END
```

Basic Terms & Symbols in Algorithms…

- **Decision/Comparison Structure…**
  - Example:
    - Write down the algorithm of the program that continues until a number greater than 10 and less than 20 is entered.

```
1 START
2 READ x
3 IF (x>10)&(x<20) THEN GO STEP 6
4 PRINT "number is not between 10 and 20"
5 GO STEP 2
6 PRINT "number is between 10 and 20"
7 END
```

Basic Terms & Symbols in Algorithms…

- **Loop Structure**
  - repeating some operations a certain number of times,
  - used to deal with consecutive values within a certain range.
  - transaction flow cycles that perform certain transaction blocks in the program for a given number of times

  - Example:
    - Sum of numbers from 1 to 5
    - Printing SAKARYA 4 times on the screen consecutively
    - Sum of even numbers or odd numbers between 1 and 100

Basic Terms & Symbols in Algorithms…

- **Loop Structure…**
  - Rules for creating a loop
    - the loop variable is assigned an initial value
    - the increase or decrease of the loop is determined.
    - the end value of the loop is determined
    - if the loop is created with decision statements; the loop variable must be increased / decreased by the amount of steps in the loop.
  - Example:
    - Write down the algorithm of the program that calculates the sum of odd numbers between 1-10.

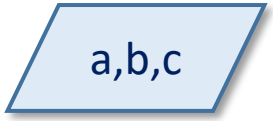
```
1 START
2 T=0
3 J=1
4 IF J>10 THEN GO STEP 8
5 T=T+J
6 J=J+2
7 GO STEP 4
8 PRINT T
9 END
```

Flowchart Symbols

- **Start**
  - indicates where the program will start.
  - each algorithm has one.

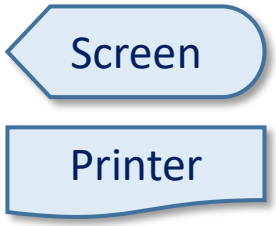
- **End**
  - indicates where the program will end.
  - can be more than one.
  - If possible, only one stop symbol should be used.

- **Input**
  - represents the variables assigned from external input of information into the program.

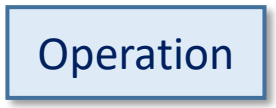


Flowchart Symbols…

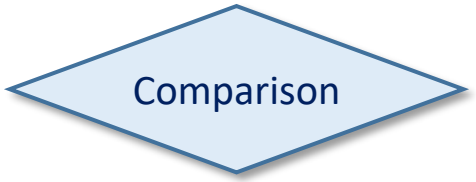
- **Output**
  - represents sending information to the screen or printer



- **Operation**
  - used to express the actions to be taken during the processing of the program.

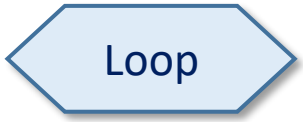


- **Decision/Comparison**
  - used to express that different action will be taken according to the result of the given condition.



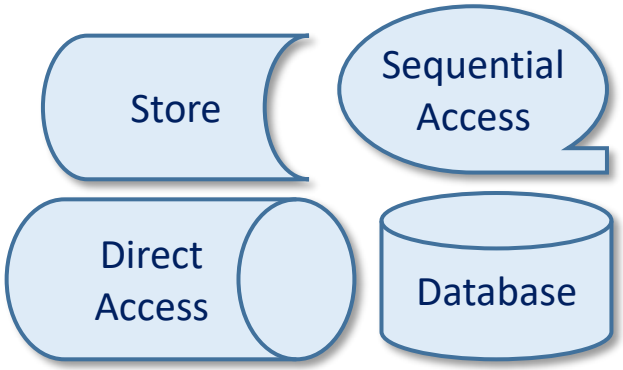
Flowchart Symbols…

- **Loop**
  - Used when a specific job or group of jobs needs to be repeated more than once.
  - The number of cycles in the loop, the loop counter and the counter increment/decrement are clearly written.



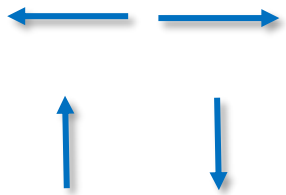
- **Calling a Function**
  - refers to the use of a previously created algorithm without putting it into the written algorithm.

- **Store into a File**
  - represents the storage or reading of the information/data into a file.

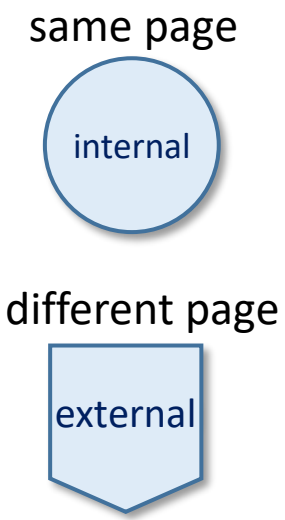


Flowchart Symbols…

- **Flow Direction**
  - Determines where the flow will be directed after a process is over.

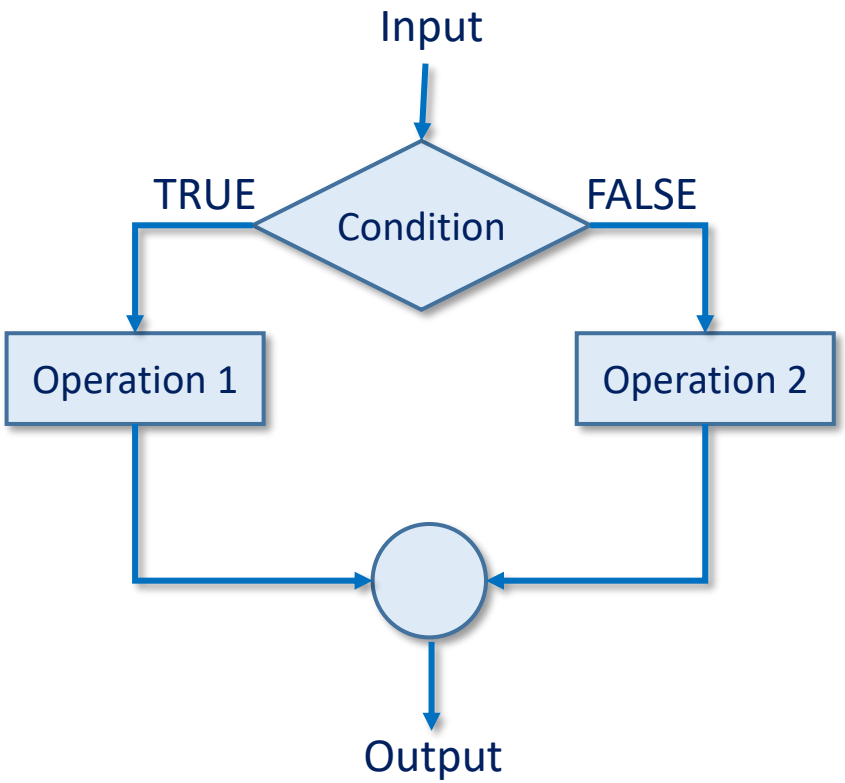


- **Connection**
  - used to continue the drawing from another place in case it does not fit on the page while drawing the flowchart.



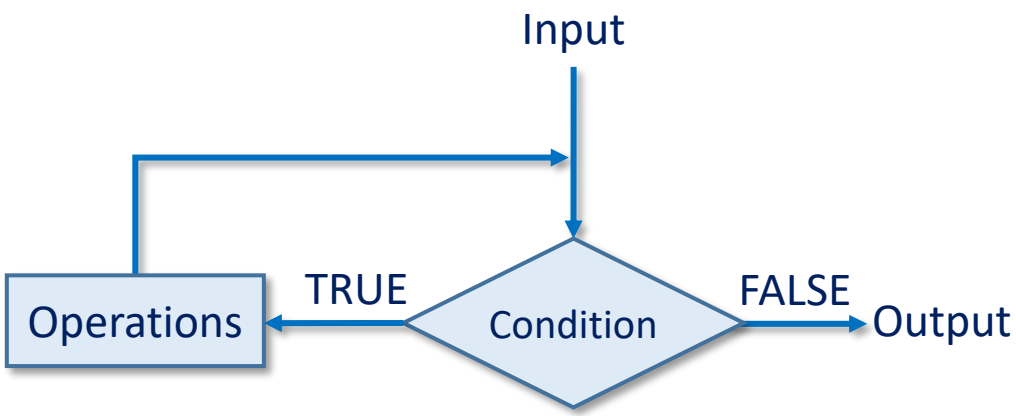
Decision/Comparison Structure in Flowcharts

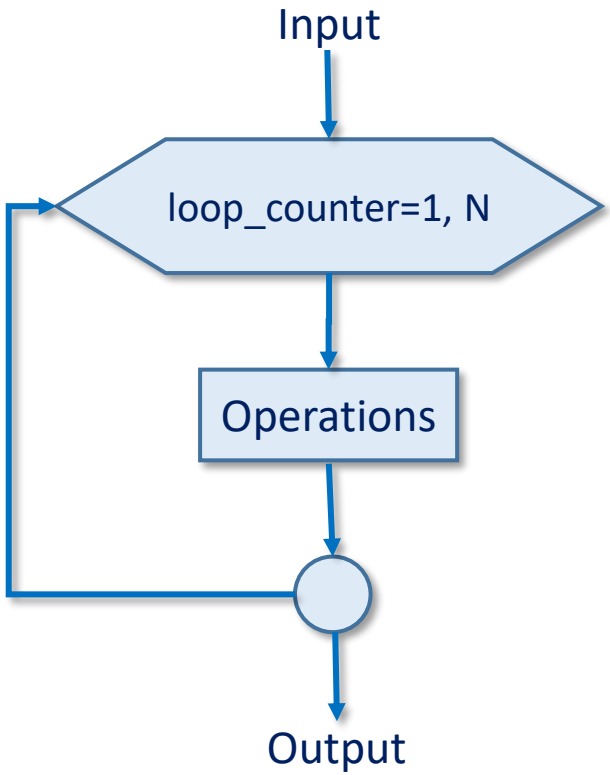
- Implementation of **DECISION** structure in algorithm design



Repeat(Loop) Structure in Flowcharts

- Implementation of **LOOP** structure in algorithm design





Examples

- **EXAMPLE – 1**
  - Problem: Write down the pseudocode and draw the flowchart of the program which calculates the total of numbers from 1 to N where N is entered from the keyboard.

  - Solution:
    - **N** ⇒ count of numbers from 1 to N.
    - We can add numbers in an incremental loop.
    - **i** ⇒ counter variable that we will control the loop increment.
    - **T** ⇒ variable to store the the total value
    - the loop variable **i** will start from 1 and increase by one to reach N.
    - the initial value of T will be 0
    - the current value of i will be added to T in the loop.

Examples…

EXAMPLE – 1…

- The pseudocode of the program

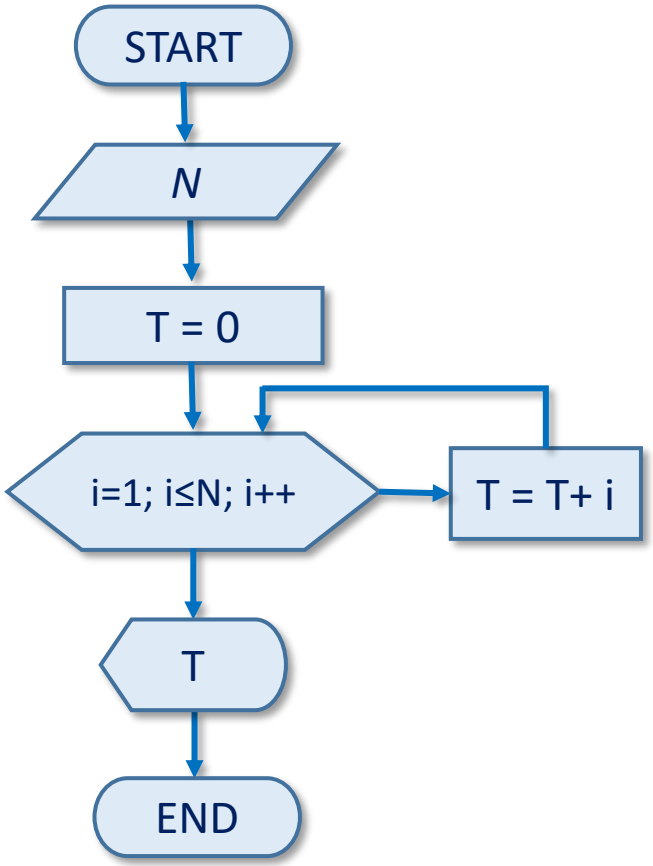
```
Step 1: START
Step 2: READ N
Step 3: T=0, i=0
Step 4: T=T+i
Step 5: i=i+1
Step 6: IF i<=N THEN GO STEP 4
Step 7: PRINT T
Step 8: END
```

LOOP

Examples…

EXAMPLE – 1…

- The flowchart of the program



i	N	Operation	T
-	10	-	-
-	10	T=0	0
1	10	T=0+1	1
2	10	T=1+2	3
3	10	T=3+3	6
4	10	T=6+4	10
5	10	T=10+5	15
6	10	T=15+6	21
7	10	T=21+7	28
8	10	T=28+8	36
9	10	T=36+9	45
10	10	T=45+10	55

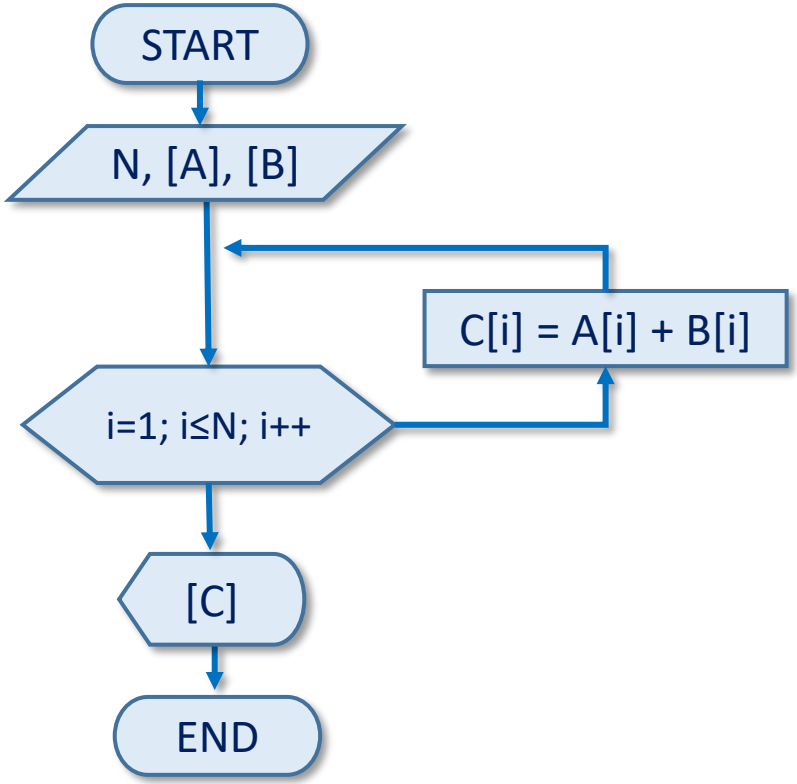
Examples…

- **EXAMPLE – 2**
  - Problem: Draw the flowchart of the program which calculates the sum of two N sized column vectors.
  - Solution:
    - i represents the column indices from 1 to N representing the elements of the vector
    - each element of the vector A can be expressed mathematically as A_i or programming as A [i].
    - let the vectors whose elements are going to be added be A and B
    - let the vector to be formed as a result of the sum be C.
    - in the vector addition operation , the element with index [i] of the first vector and the element with index [i] of the second vector are added together to obtain the element with index [i] of the total/result vector.

Examples…

EXAMPLE – 2…

- The flowchart of the program



$$[A] = \begin{bmatrix} 1 & 4 & 2 \end{bmatrix}$$

$$[B] = \begin{bmatrix} 2 & 0 & 3 \end{bmatrix}$$

for vectors A,B and N=3

i	Operation	C[i]
1	C[1]=1+2	3
2	C[2]=4+0	4
3	C[3]=2+3	5

$$[C] = \begin{bmatrix} 3 & 4 & 5 \end{bmatrix}$$

Examples…

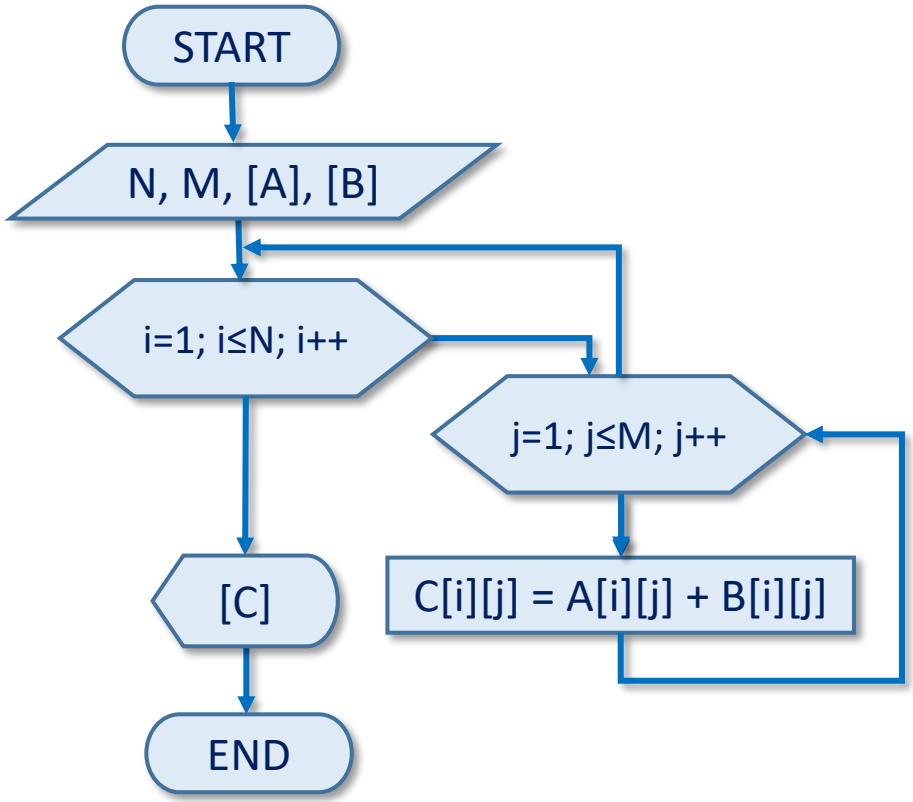
- **EXAMPLE – 3**
  - Problem: Draw the flowchart of the program which calculates the sum of two NxM sized matrices.

  - Solution:
    - **i** and **j** ⇒ indices representing the elements of the matrices
    - i represents row indices from 1 to M
    - j represents column indices from 1 to N
    - each element of the matrix A can be expressed mathematically as $A_{i,j}$ or programming as A [i] [j].
    - Let the matrices whose elements are going to be added be matrices A and B
    - Let the matrix to be formed as a result of the sum be the matrix C.
    - In the matrix addition operation , the element with index [i][j] of the first matrix and the element with index [i][j] of the second matrix are added together to obtain the element with index [i] [j] of the total/result matrix.

Examples…

EXAMPLE – 3…

- The flowchart of the program



$$[A]=\begin{bmatrix}1 & 3\\ 1 & 2\end{bmatrix} \quad [B]=\begin{bmatrix}2 & 1\\ 3 & 0\end{bmatrix}$$

for matrices A,B and N=M=2

i	j	İşlem	C[i][j]
1	1	C[1][1]=1+2	3
1	2	C[1][2]=3+1	4
2	1	C[2][1]=1+3	4
2	2	C[2][2]=2+0	2

$$[C]=\begin{bmatrix}3 & 4\\ 4 & 2\end{bmatrix}$$

STUDY QUESTIONS…

- Draw the flowchart of the program which calculates the multiplication of two matrices (first matrice is MxN sized and the second is NxK sized).

References

- Deitel, C++ How To Program, Prentice Hall
- Prof. Dr. Cemil ÖZ, Programlamaya Giriş Ders Notları