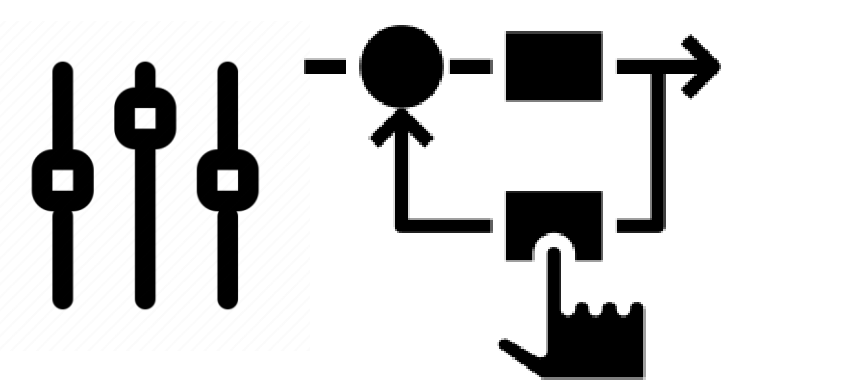
The National School of Artificial Intelligence
المدرسة الوطنية العليا للذكاء الاصطناعي



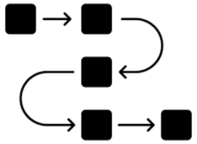
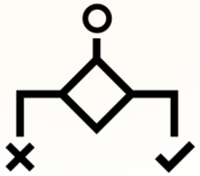
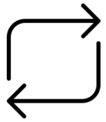
Data Structure & Algorithms 1

CHAPTER 2:
CONTROL STRUCTURE

Sep – Dec 2023

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

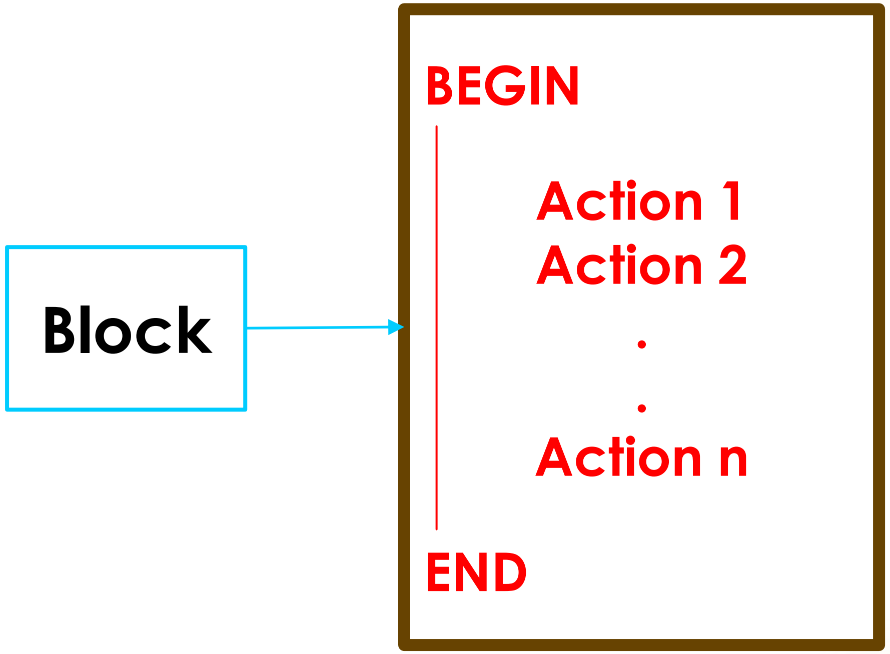
Algorithm: Control Structure

1. Sequencing 
2. Conditional and Alternative 
3. Repetitive 

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

1. **Sequencing:** Primitive actions are executed **in the order in which they appear** in the algorithm.



Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

Sequencing Example

- Create an algorithm that, based on an amount excluding taxes (MontHT), displays the amount to be paid including all taxes (MontTTC), considering that a tax (VAT) is applied to the amount excluding taxes.
- To calculate the total amount including taxes (what the customer will pay) based on an amount excluding taxes: **MontTTC = MontHT * (1 + VAT/100)**
- **Example** with a 20% VAT:
  - MontTTC = MontHT * (1 + 20/100)
  - MontTTC = MontHT * 1.2

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

```
Algorithm Calculate_Invoice

Constant VAT = 20
Variables Real NetAmount, GrossAmount

BEGIN
    WRITE ('Enter the Net Amount: ')
    READ (NetAmount)
    GrossAmount = NetAmount * (1 + VAT/100)
    WRITE ('The Gross Amount is: ', GrossAmount)
END
```

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

1. **Conditional:** If the condition is met, execute the block; otherwise, continue sequentially.

```
IF Logical_expression THEN

        Block

END IF
Next instruction
```

Evaluate the logical expression:

- **IF** the logical expression is **TRUE**, then the **Block** is executed.
- Otherwise, **IF** the logical expression is **FALSE**, we proceed sequentially (**Next Instruction**)

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

1. **Alternative:** If the condition is not met, execute the **else** block.

```
IF Logical_expression THEN
    Block1
ELSE
    Block2
ENIF
Next instruction
```

Evaluate the logical expression:

- **IF** the logical expression is **TRUE**, then the **Block1** is executed and then we proceed to **Next Instruction**
- **ELSE**, **i.e.**, logical expression is **FALSE**, so execute **block2** and then **Next Instruction**

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

Remarks:

1. The conditional statement is a specific case of the alternative statement;

2. A block is a coherent set composed of 1 or more actions. Intended under the **IF** statement.

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

Example1: Conditional and Alternative

```
……
Go straight until the next intersection.
IF the right street is open to traffic THEN
    Turn right
    Continue forward
    Take the second left
ELSE
    Keep going until the next right street
    Take that street
    Take the first right
ENIF
………
```

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

Example2: Conditional and Alternative

Write an algorithm that, given an amount, **displays** the amount to be paid, considering a 10% discount for any amount greater than or equal to 3500 DZD

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

Example2: First solution

```
Algorithm AmountToPay

Constant discount = 0.10
Variables Real InitAmount, AmountToPay

BEGIN
    READ (InitAmount)
    IF InitAmount >= 3500 THEN
        AmountToPay = InitAmount * (1 - discount)
    ELSE
        AmountToPay = InitAmount
    ENDIF
    WRITE (AmountToPay)
END
```

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

Example2: Second solution

```
Algorithm AmountToPay

Constant discount = 0.10
Variables Real InitAmount, AmountToPay

BEGIN
    READ (InitAmount)
    AmountToPay = InitAmount
    IF InitAmount >= 3500 THEN
        AmountToPay = InitAmount * (1 - discount)
    ENDIF
    WRITE (AmountToPay)
END
```

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

Example3:

IF... ELSE statement opens **two** paths, corresponding to **two** different processes.

However, there are many situations where **two** paths are **not sufficient**.

For example, a program that needs to determine the state of water based on its temperature should be able to choose from **three** possible responses (solid, liquid, or gaseous).

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

Example3:

ANALYSIS: First Solution

- Let Temp be the water temperature.
- Start by reading the temperature, Temp.
- Check if Temp is less than or equal to 0; if true, display that it is ice.
- Check if (Temp is greater than 0 and Temp is less than or equal to 100); if true, display that it is liquid.
- Check if Temp is greater than 100; if true, display that it is vapor.

```
Algorithm TempV1
Variables  Integer Temp
BEGIN
    WRITE (‘Provide the temperature: ’)
    READ (Temp)
    IF temp <= 0 THEN
        WRITE (‘Ice’)
    ENDIF
    IF (temp > 0) AND (Temp <=100) THEN
        WRITE (‘Liquid’)
    ENDIF
    IF temp > 100 THEN
        WRITE (‘Vapor’)
    ENDIF
END
```

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

Example3:

Remarks about the first solution

You will notice that this is a bit cumbersome. The conditions are somewhat similar, and most importantly, we force the machine to examine three successive tests, all of which concern the same thing: the water temperature (the value of the variable Temp).

- *It would be much more rational to **nest** the tests in this way,*

```
Algorithm TempV1
Variables  Integer Temp
BEGIN
    WRITE ('Provide the temperature: ')
    READ (Temp)
    IF temp <= 0 THEN
        WRITE ('Ice')
    ELSE
        IF Temp <=100 THEN
            WRITE ('Liquid')
        ELSE
            WRITE ('Vapor')
        ENDIF
    ENDIF
END
```

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

Example3:

Remarks about the second solution

We have made savings: instead of having to type three conditions, one of which is compound, we now have only two simple conditions. But more importantly, we have saved on the computer's ***execution time****.*

If the temperature is below zero, it writes "It's ice" and goes directly to the end without being slowed down by the examination of other possibilities (which are necessarily false).

This second version is not only simpler to write and more readable but also more efficient in execution.

Nested testing structures are therefore an indispensable tool for simplifying and optimizing algorithms.

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

Example4:

It is about reading 3 integers **A, B, C** and **sorting** them in **ascending** order, which means putting the smallest value in A and the largest value in C.

- **Two** different solutions with two different analyses.

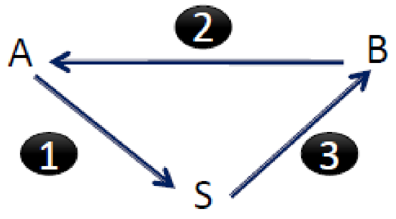
Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

Example4:

ANALYSIS: First Solution

- Let A, B, and C be three integers.
- We compare A and B, and if they are out of order, we swap A and B.
- Next, we compare B and C, and if they are out of order, we swap B and C.
- Finally, we compare A and B again, and if they are out of order, we swap A and B.
- To swap two variables, we use a third variable S to temporarily store the value of one and perform the swap.




```
Algorithm Sorting
Variables  Integer A, B, C, S
BEGIN
    READ (A, B, C)
    IF A > B THEN
        S = A
        A = B
        B = S
    ENDIF
    IF B > C THEN
        S = C
        B = C
        C = S
    ENDIF
    IF A > B THEN
        S = A
        A = B
        B = S
    ENDIF
END
```

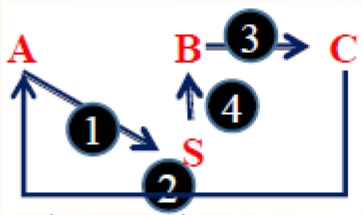
Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

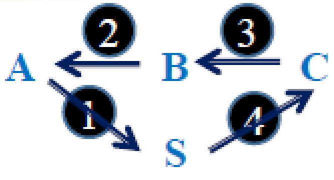
Example4:

ANALYSIS: Second Solution

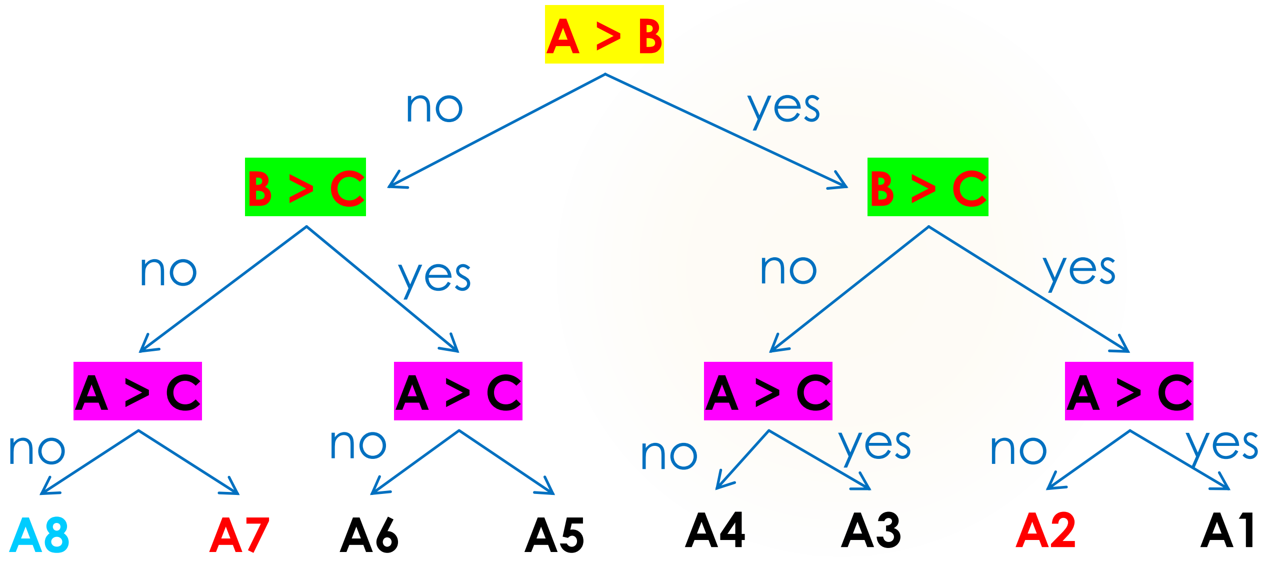
- Let A, B, and C be three integers.
- We compare A and B, then B and C, then A and C.
- Next, we decide about the action to take once **all** the comparisons are done.



Right circular swapping



Left circular swapping



A1 : 8 5 4 - Swap A and C
A2 : impossible
A3 : 8 5 6 – Left circular swapping
A4 : 8 5 9 - Swap A and B
A5 : 5 8 4 – Right circular swapping
A6 : 5 8 6 – Swap B and C
A7 : impossible
A8 : 5 6 8 Do nothing

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

Example4:

A1 : 8 5 4 - Swap A and C
A2 : impossible
A3 : 8 5 6 – Left circular swapping
A4 : 8 5 9 - Swap A and B
A5 : 5 8 4 – Right circular swapping
A6 : 5 8 6 – Swap B and C
A7 : impossible
A8 : 5 6 8 Do nothing

```
Algorithm Sorting
Variables Integer A, B, C, S
BEGIN
    READ (A, B, C)
    IF A > B THEN
        IF B > C THEN
            A1
        ELSE
            IF A > C THEN
                A3
            ELSE
                A4
            ENDIF
        ENDIF
    ELSE
        IF B > C THEN
            IF A > C THEN
                A5
            ELSE
                A6
            ENDIF
        ENDIF
    ENDIF
END
```

A1:
```
S = A
A = C
C = S
```

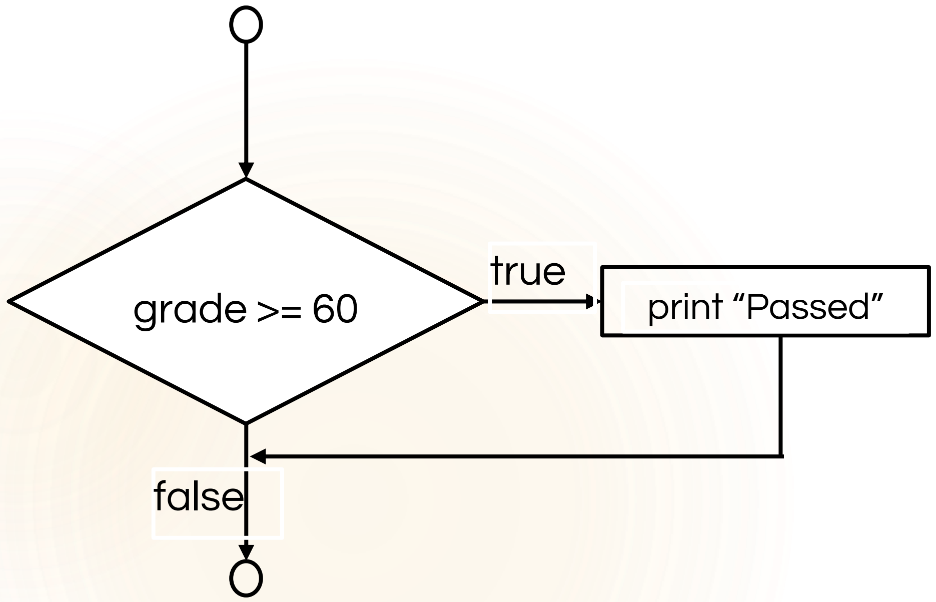
A3:
```
S = A
A = B
B = C
C = S
```

A5:
```
S = A
A = C
C = B
B = S
```

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

- Flowchart of pseudocode statement



A decision can be made on any expression.

`zero - false`

`nonzero - true`

C++ code

```
if ( grade >= 60 )
    {
    cout << "Passed";
    }
else
    {
    cout << "Failed";
    }
```

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The repetitive structure: It often happens that we need to repeat an action or a set of actions (block) multiple times. To achieve this, we use a repetitive structure. There are three forms of repetitive structures:

1. The **FOR** form
2. The **WHILE** form
3. The **REPEAT** form

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **FOR** form:

FORMAT:

```
FOR Control_variable FROM Start_value TO End_value DO

    Block

END FOR
```

Remark: This structure is used when the number of repetitions is <u>known in advance</u>

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **FOR** form:

OPERATION:

a) Initialization of the control variable
b) Incrementation at each iteration
c) End test

For each value of the control variable, which ranges from the initial value to the final value with a step of **1**, the block will be executed. Each execution of the block is called an iteration (or loop).

- The block is repeated (final value - initial value + 1) times

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **FOR** form:

Note: The FOR form is well-suited for cases where you **can predict** the number of block executions, i.e., the number of iterations, and where it is possible to determine the **initial** and **final** values of the control variable.

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **FOR** form: EXAMPLE

```
A = 5
FOR I FROM 1 TO 6 DO
   K = A * I
   A = K
END FOR
```

A	I	K
5	1	5
5	2	10
10	3	30
30	4	120
120	5	600
600	6	3600
3600	7	

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **WHILE** form:

FORMAT:

```
WHILE Condition (logical expression) DO

        Block

END WHILE
```

Remark: The number of times to repeat is **not known** in advance. The number of iterations depends on the value of the **logical expression**.

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **WHILE** form:

OPERATION:

The block is repeated as long as the logical expression is **TRUE**.

At the beginning of each iteration, the logical expression is evaluated. **IF** the value is **TRUE**, then the block is executed; **ELSE**, it proceeds sequentially.

The repetition stops when the condition is no longer satisfied.

Remark: The block may not be executed at all (logical expression is FALSE from the start). In this case, the WHILE structure is a 0, N structure.

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **WHILE** form: EXAMPLE

```
A = 5
I = 1
WHILE I <= 6 DO
   K = A * I
   A = K
   I = I + 1
END WHILE
```

Initialization

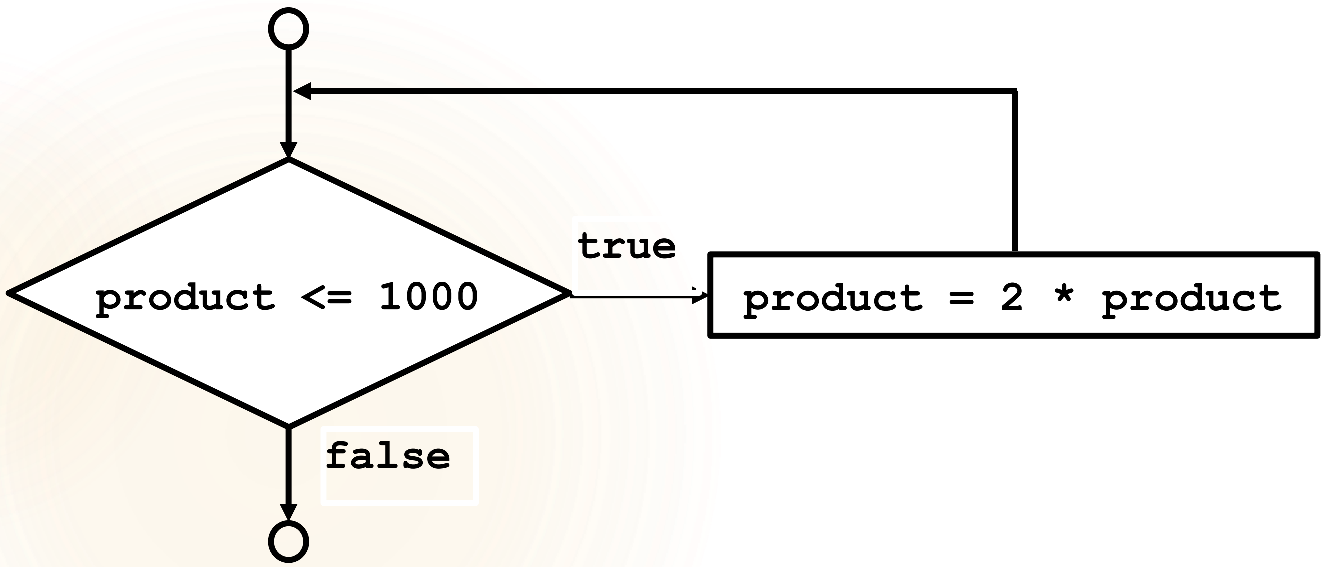
End of repetition test

Incrementation

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **WHILE** form: Flowchart



C++ code

```
while (product <= 1000)
{
    product = 2 * product
}
```

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **WHILE** form: EXAMPLE

Write the algorithm that allows you to determine if an integer N, read from the keyboard, is prime or not.

Reminder: the twenty-five prime numbers less than 100 are:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, and 97.

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **WHILE** form: EXAMPLE

Idea:

Simply look for a divisor of N in the interval [2, N Div 2].

As soon as such a divisor is found, the number N is not prime.

Therefore, we will use the WHILE structure, which allows us to stop the search as soon as <u>a divisor is found</u>.

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **WHILE** form: EXAMPLE

Analysis:

- Let N be an integer.
- Create an intermediate variable `Ndiv2`, which corresponds to (N DIV 2).
- We divide `N` by divisors `d` generated from `2` up to `Ndiv2` at most.
- This process stops either when `d` divides `N` or when `d` exceeds `Ndiv2`.
- At the end of the process, to determine if N is prime or not, we examine how the previous process concluded. `IF (N MOD d > (N Div 2)),` then N is prime; OTHERWISE, it is not.

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **WHILE** form: EXAMPLE

```
Algorithm PrimeNumber
Variables Integer N, d, Ndiv2
BEGIN
    READ (N)
    d = 2
    Ndiv2 = N DIV 2
    WHILE (N MOD d <> 0) AND (d <= Ndiv2)  DO
        d = d + 1
    END WHILE
    IF (d > Ndiv2) THEN
        WRITE (N, ‘ is a Prime number’)
    ELSE
        WRITE (N, ‘ is NOT a Prime number’)
    ENDIF
END
```

```
include <iostream>
using namespace std;
int main() {
    int N, d, Ndiv2;
    cout << "Enter an integer N: ";
    cin >> N;
    d = 2;
    Ndiv2 = N / 2;
    while (N % d != 0 && d <= Ndiv2) {
        d = d + 1;
    }
    if (d > Ndiv2) {
        cout << N << " is a Prime #" << endl;
    } else {
        cout << N << " is not a Prime #" <<endl;
    }
return 0
}
```

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **REPEAT** form:

FORMAT:

```
REPEAT

    Block

UNTIL Condition (Logical expression)
```

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **REPEAT** form:

OPERATION:

The actions located between the keywords **REPEAT** and **UNTIL** are repeated until the logical expression becomes TRUE. At the end of each iteration, the logical expression is evaluated. If the value is FALSE, the actions are executed; otherwise, the program continues in sequence.

Remark: The actions are executed **at least once** because the logical expression is evaluated only when we reach the UNTIL keyword. This is why the REPEAT structure is referred to as a 1, N structure.

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

The **REPEAT** form: EXAMPLE

```
A = 5
I = 1
REPEAT
  K = A * I
  A = K
  I = I + 1
UNTIL (I > 6)
```

Initialization

Incrementation

End of repetition test

Total number of iteration is **6**

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

Additional Notes:

- The WHILE and REPEAT forms are used when the number of loops (iterations) is **unknown**. However, it is always at least 1 for REPEAT, whereas it can be equal to 0 in the WHILE form. This difference often influences the choice between these two forms.
- Be cautious of design errors that can lead to an "infinite loop," where the condition always remains true.

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure

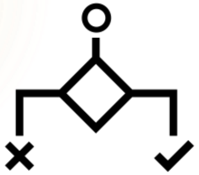
Infinite Loop Example

```
A = 5
REPEAT
   X = A * 2
   Y = 100 - X
   A = A - 2
UNTIL (A == 0)
WRITE (X, Y)
```

Indeed, if we unfold this portion of the algorithm, we can see that A will never be equal to zero. This would result in an infinite loop.

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024

Algorithm: Control Structure



Choice of the loop (FOR, WHILE, REPEAT)

The rule is simple:

- If the number of iterations is known in advance, then we use a FOR loop.
- Otherwise: we use a REPEAT loop (when there is always at least one iteration),
- or a WHILE loop (when the number of iterations can be zero).

Copyright © Dr. Noureddine Lasla/ Dr. Fouzia Anguel, 2023/2024