ITI 1120
Labo #3

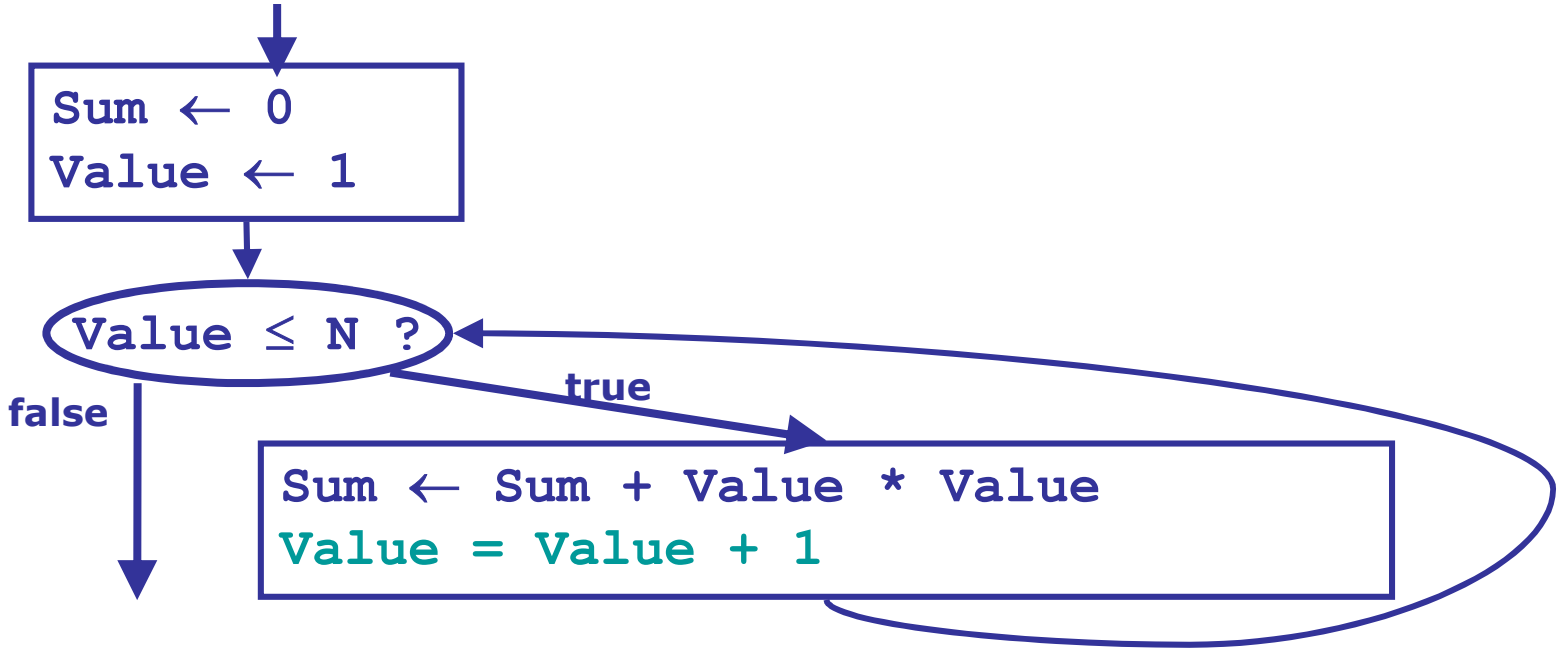
Loops

Laboratory objective:

- Instruction while
- Instruction for
- Exercises with loops

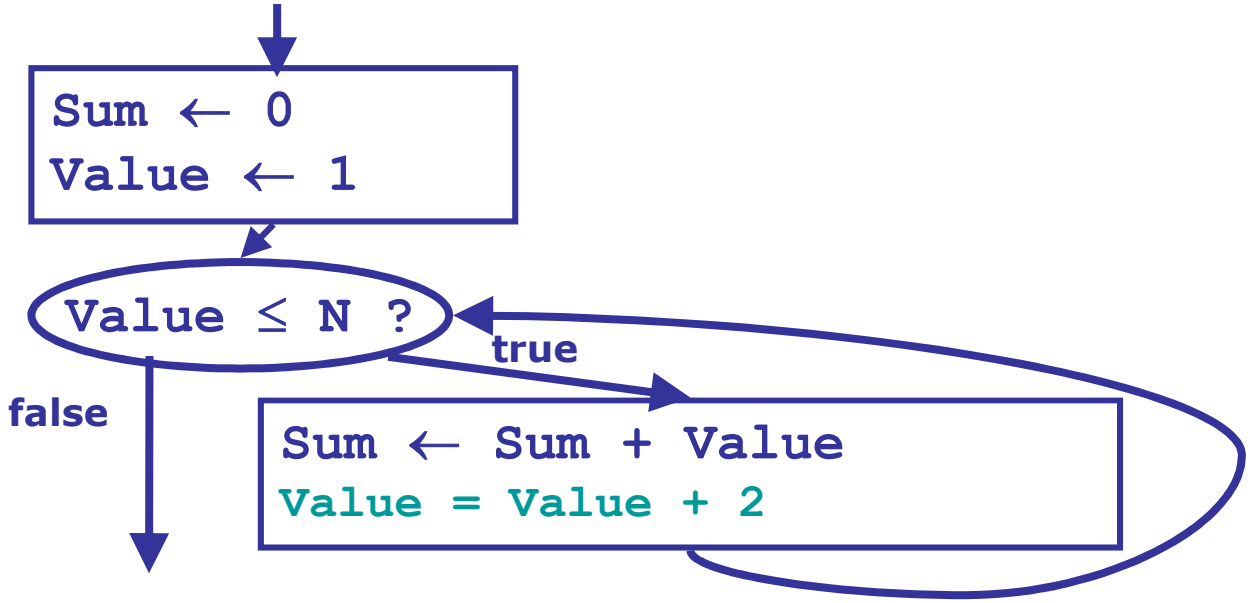
Sum of squares

DATA: N (integer ≥ 1)
RESULTS: Sum (Sum of squares from 1^2 to N^2)
INTERMEDIAIRIES: Value (current value to square)
HEADER: Sum ← SumOfSquares(N)
MODULE:

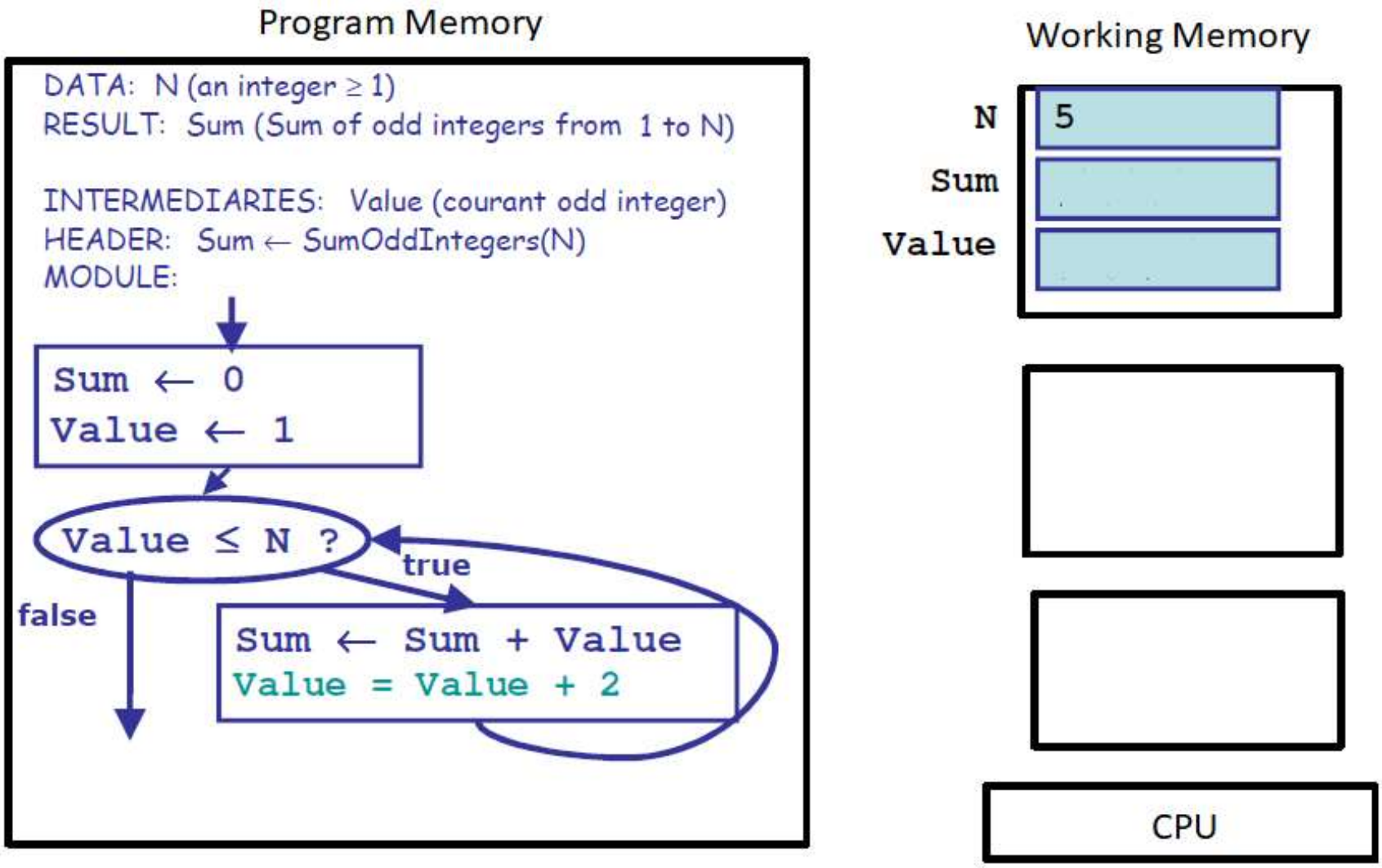


Sum of odd integers

DATA: N (integer ≥ 1)
RESULTS: Sum (Sum of odd integers from 1 to N)
INTERMEDIAIRIES: Value (current odd integer)
HEADER: Sum ← SumOddIntegers(N)
MODULE:



Programming model for SumOddIntegers(5)



Trace for SumOddIntegers(5)

Instructions	N	Value	Summ
Initial valeurs	5	?	?
Sum ← 0			0
Value ← 1		1	
Value <=N ? (1 <= 5) true			
Sum ← Sum + Value			1
Value = Value + 2		3	
Value <=N ? (3 <= 5) true			
Sum ← Sum + Value			4
Value = Value + 2		5	
Value <=N ? (5 <= 5) true			
Sum ← Sum + Value			9
Value = Value + 2		7	
Value <=N ? (5 <= 5) false			

Exercise 1

- The code was written to print integer numbers from 10 à 1.
  - You should find two logical errors.
  - Correct them

```
counter = 10
while(counter >= 0):
  print(counter)
  counter = counter + 1
```

Exercise 2

- Develop a program that prints all the numbers from 1 to N inclusively.
  - Think about an algorithm (draw it yourself).
  - Convert the algorithm in Python.
  - Ask the user lto provide a value for N and then call the algorithm/problem resolution function.
  - The algorithme must display the numbers using a loop. (try with both while and for).

Exercise 3

- Develop a guessing software. the program generate a number between 1 and 10, and then ask the user to guess that number. If the user does not get it, the program indicates if it is too high or low and asker the user for another guess. Once the user gets the correct response, a successful message comes up with the number of tries used.

  - Develop your algorithm. Convert it in Python.

  - The program must generate a random number and call the algorithm/function guess(). It will display the successfull message and the number of tries as the result.

Exercise 4

- The *n factorial* de n (n!) is the product of integers from 1 to n. Thus 4! = 1*2*3*4 = 24. By definition 0! = 1. The factorial is not defined for negative numbers.

- Develop a program that asks for a non negative number (≥0), calculates and display its factorial. Note that the computing of the factorial is similar to the sum of integers between 1 and N, but with the multiplication instead of the addition (and very similar to the product from 1 to N). But do not forget to produce a value for 0.
  - Develop an algorithm yourself.
  - Convert it into Python.
  - The program must ask the user for a positive number, call computeFact() to compute the factorial, and display the result.

Exercice 4 (suite)

- The program must respect the following:
  - Verify if the user has provided a negative number.
  - If a negative number is given, display a message indicating that the number should not be negative and request another number.
  - If the number is still negative, keep continue requesting a non negative number,
  - After receving a positive number, compute its factorial by calling the function Computefact() and display the result.
  - Test the above in Python.