ITI 1120
Labo #4

Loops

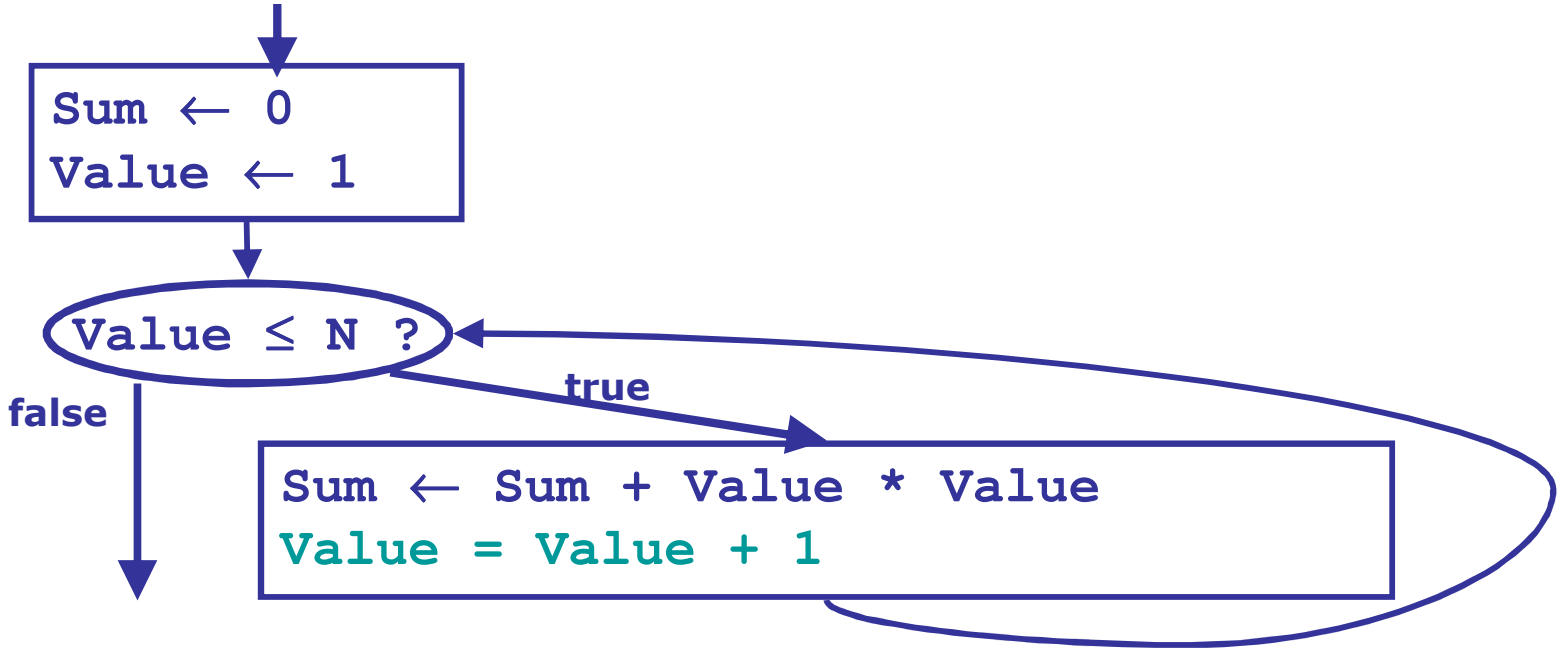
@2015 Diana Inkpen, University of Ottawa, All rights reserved

Laboratory objective:

- Instruction while
- Instruction for
- Exercises with loops

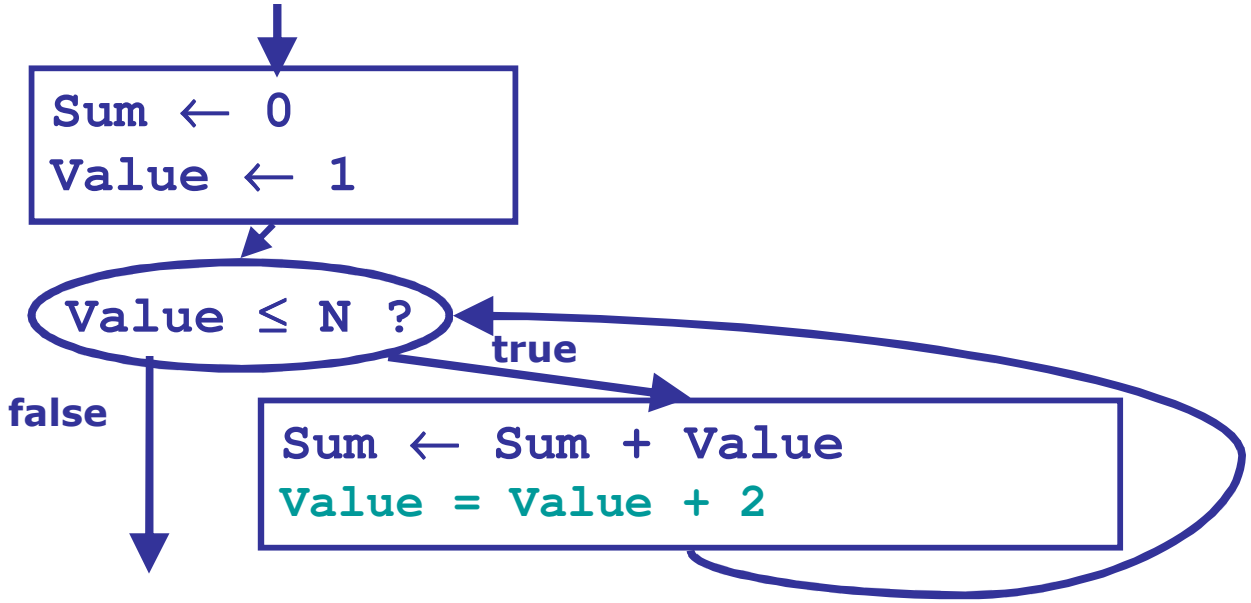
Sum of the squares

DATA: N (an integer ≥ 1)
RESULT: Sum (Sum of squares from 1^2 to N^2)
INTERMEDIARIES: Value (courant value to square)
HEADER: Sum ← SumOfSquares(N)
MODULE:

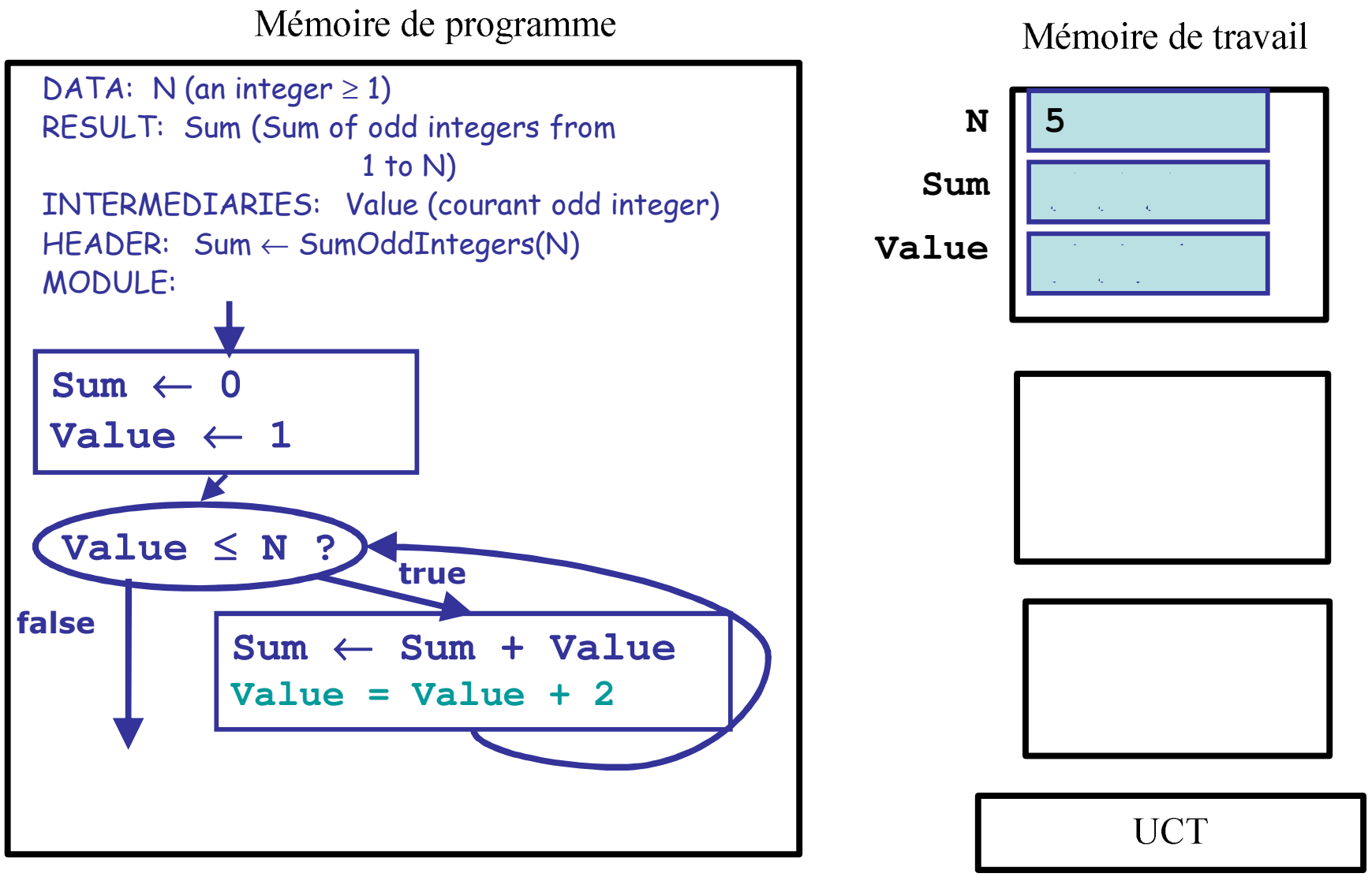


Sum of odd integers

DATA: N (an integer ≥ 1)
RESULT: Sum (Sum of odd integer from 1 to N)
INTERMEDIARIES: Value (courant odd integer)
HEADER: Sum ← SumOddIntegers(N)
MODULE:



Programming model for SumOddIntegers(5)



Trace of SumOddIntegers(5)

Instructions	N	Value	Sum
Initial values	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 integers from 10 to 1.
  - You should find to logical errors.
  - Correct them

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

Exersise 2

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

Exercise 3

- Develop a software to play a guessing game. The programm generate a number betweeen 1 and 10, and then ask the user to guess it. If the user does not get it, the program indicate if it is high or low and ask again the user to guess. One the user find the response, a successful message comes up with the number of tries.
  - Develop your algorithm. Convert in Python.
  - The program must generate a random number snd cll the l'algorithm/function guess(). It displays the successful message and the number of tries as the result.

Exercise 4

- The n *factorial* (n!) is the product of integers between 1 and 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 integer (≥0), calculate and display the factorial. Note that the computing of the factorial is similar to the of integers between 1 and N, but with the multiplication instead of the addition (and very similar to the product 1 to N). But do not forget to produce a value for 0.
  - Developp an algorithm for yourself.
  - Convert it in Python.
  - the program must ask the user a positive number, call computeFact() to obtain the factorial, and display the result.
  - The algorithm/function computeFact() calculatee the factorial.

Exercise 4 (suite)

- the program must respect the following:
  - Verify if the user has provideda positive number
  - If a negative number is given, display a message informing that the number should not be negative and ask for another number.
  - If the number is still negative, keep continue asking the user to enter a non negative number,
  - After receiving a positive number, calculate the factorial by calling the function computeFact() and display the result.
  - Test the above in Python.