Il Pensiero Computazionale

Computational Thinking

Algorithms and Flowcharts

Ai miei studenti e a tutti coloro che reputano il pensiero computazionale essere una grande palestra per la mente

To all my students and people who believe in computational thinking to be a wonderful gym for own mind

Roberto Atzori

“ L'informatica non riguarda i computer più di quanto l'astronomia riguardi i telescopi. “

“Computer Science is no more about computers than *astronomy* is about telescopes*”*

(Edsger Wybe Dijkstra)

I am very grateful to Devin Cook for Flowgorithm software

Diagrammi a blocchi – Flowcharts

Memorandum

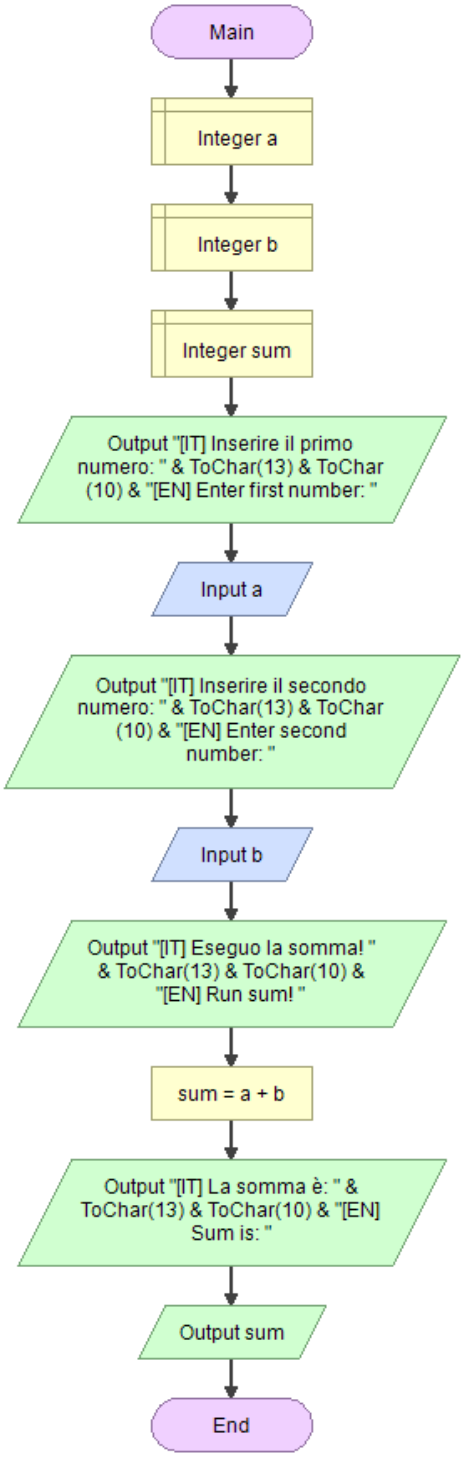
G S A F R

[IT] Elaborazioni Ricorsive	[EN] Recursive
[IT] Elaborazioni con Funzioni / Procedure	[EN] By Procedure / Function
[IT] Elaborazioni su Array	[EN] By Array
[IT] Elaborazioni su Stringhe	[EN] By String
[IT] Elaborazioni Generico / Matematico	[EN] Generic / Math

Sommario

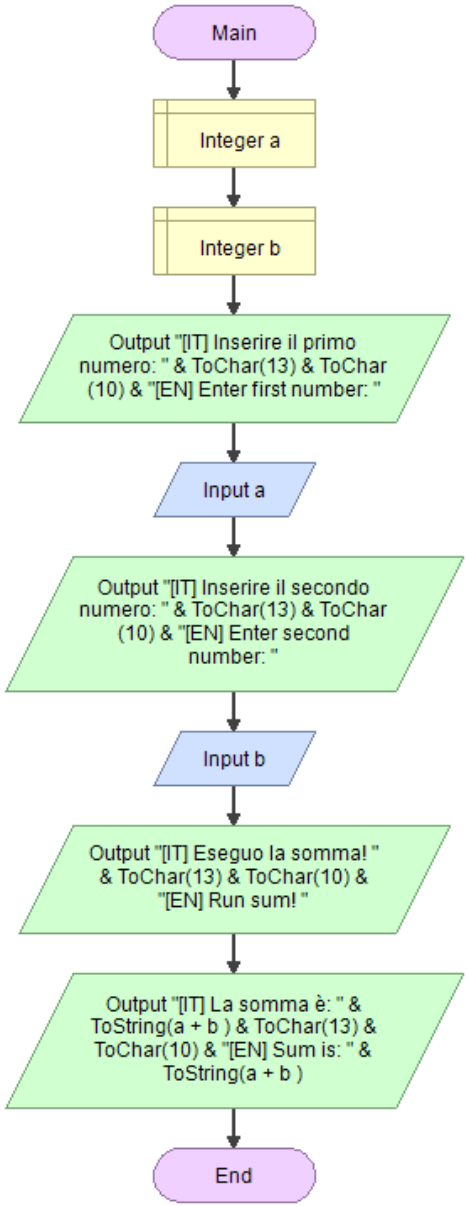
1. G ‖ Somma di due numeri (v1 - usando una terza variabile)

Sum of two no's (v1 - using third variable)



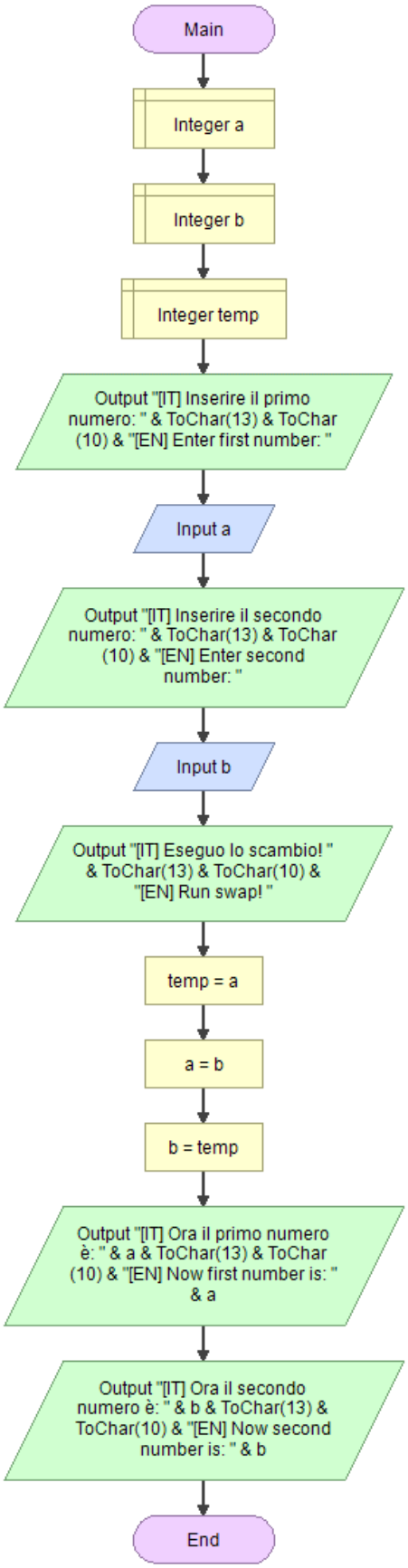
2. G ‖ Somma di due numeri (v2 - senza usare una terza variabile)

Sum of two no's (v2 - without using third variable)



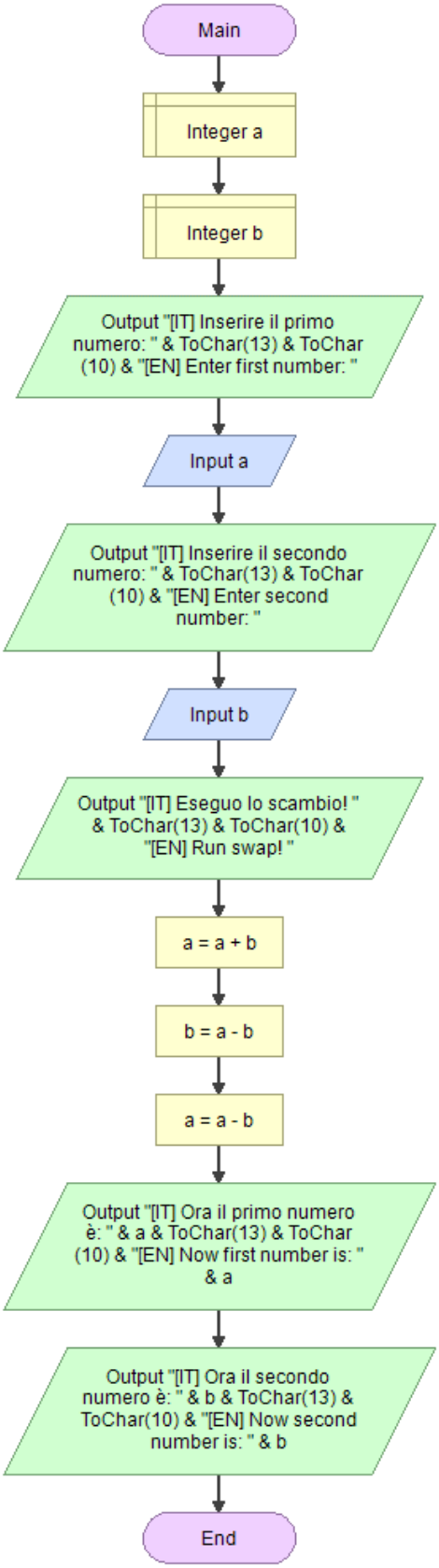
3. G ‖ Scambiare due numeri (v1 - usando una terza variabile)

Sum of two no's (v1 - using third variable)



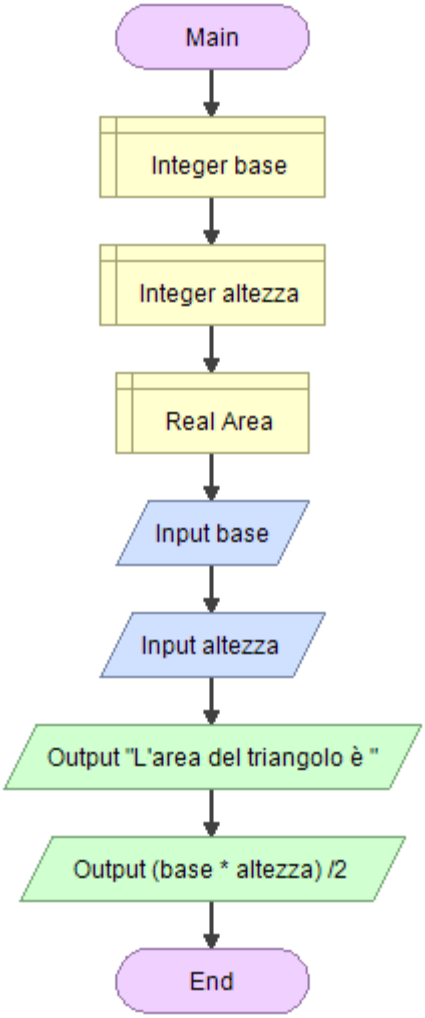
4. G ‖ Scambiare due numeri (v2 - senza usare una terza variabile)

Swap two no's (v2 - without using third variable)



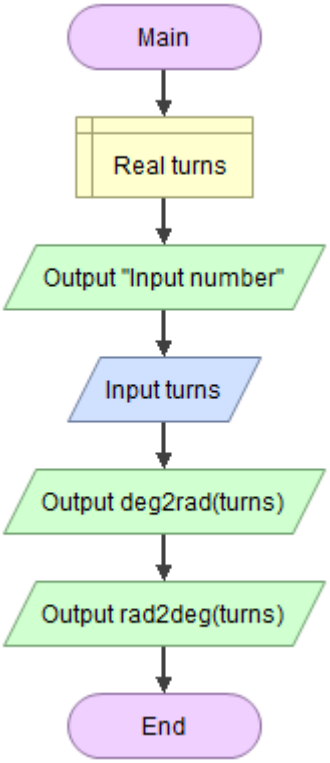
5. G ‖ Data la base e l'altezza di un triangolo, scrivere l'area

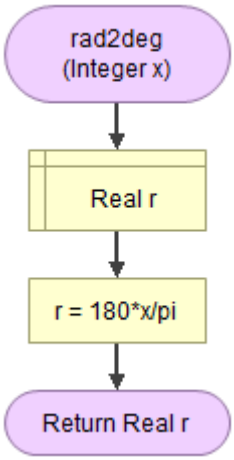
Calculate a triangle's area

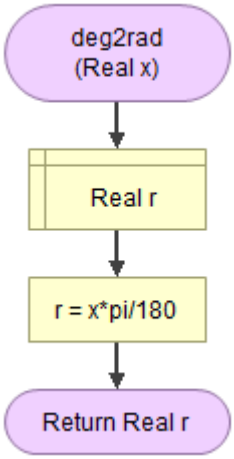


6. **G F** ‖ Conversione angolare (gradi ←→ radianti)

Angular conversion (grads ←→ radians)

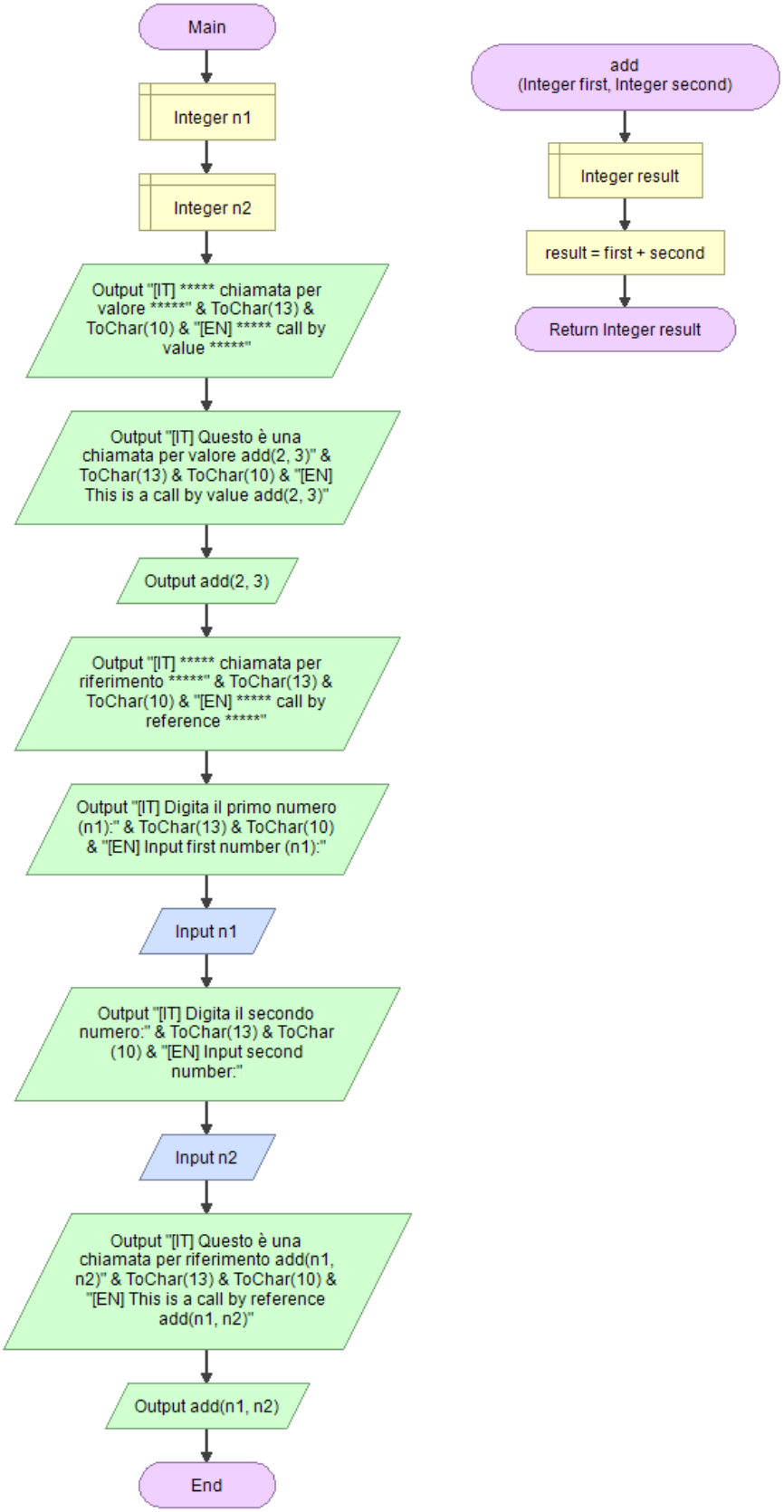






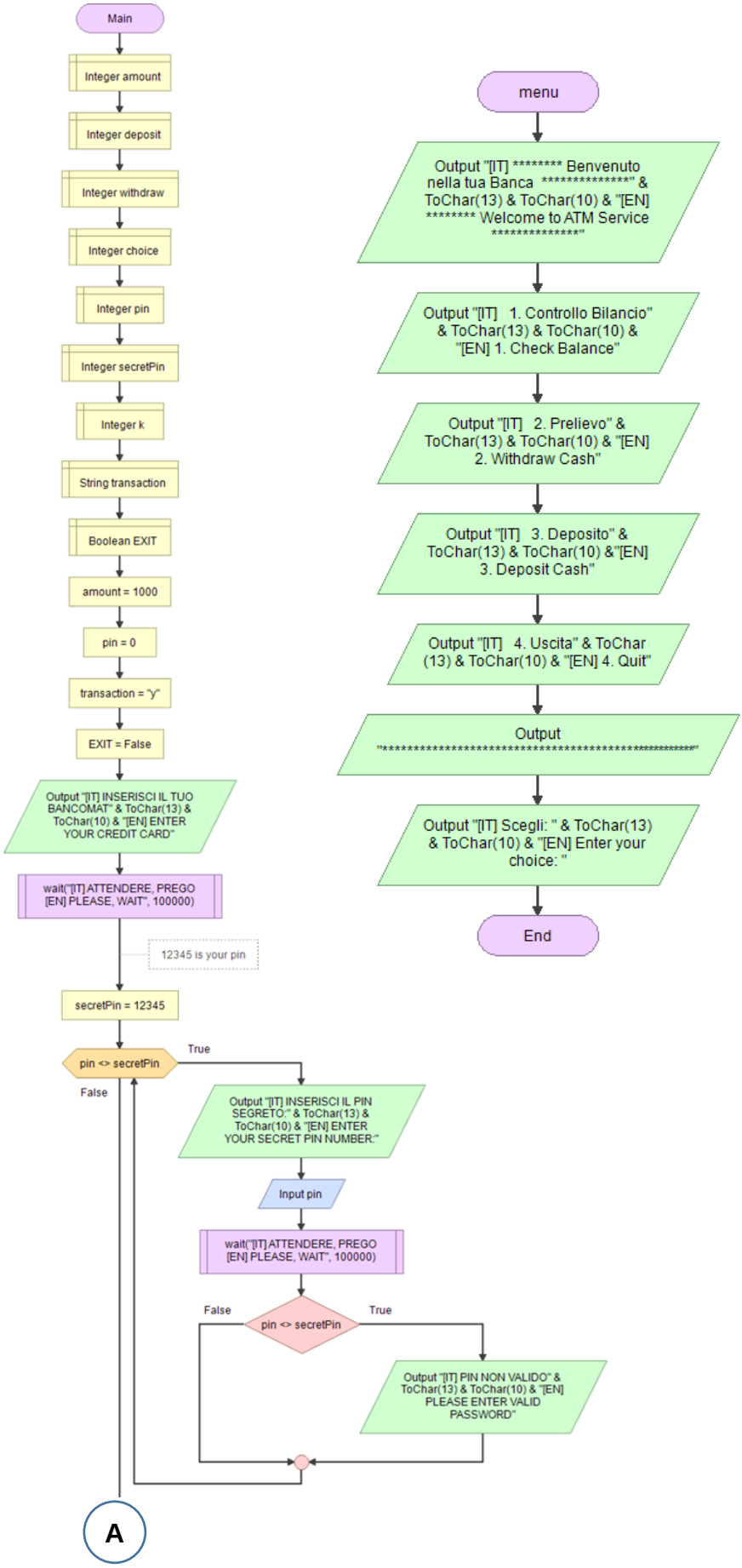
7. **G F** | Addizionare due numeri usando la chiamata per valore e la chiamata per riferimento

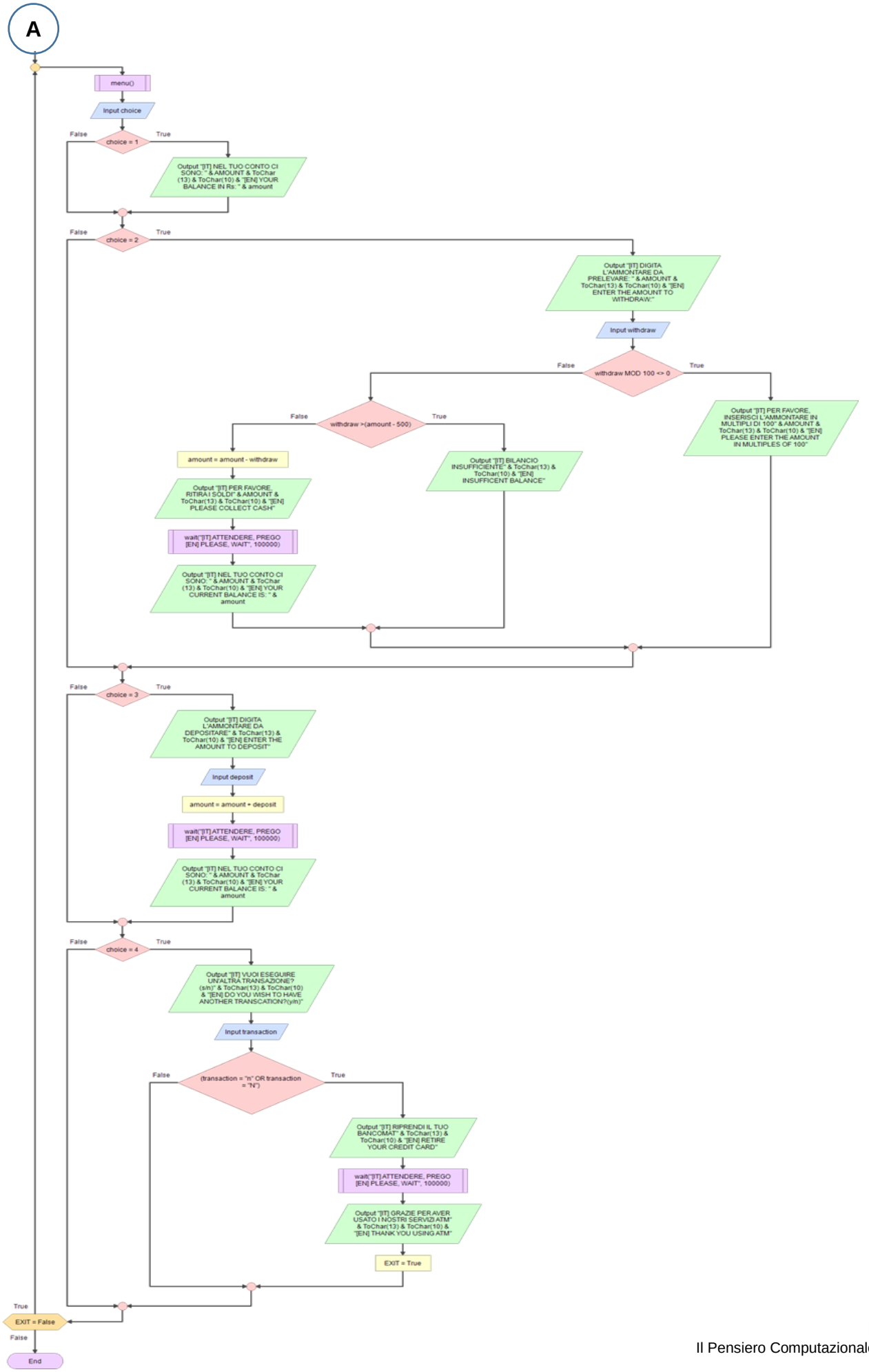
Add two numbers using call by value and by reference



8. G F | Simulazione sportello ATM

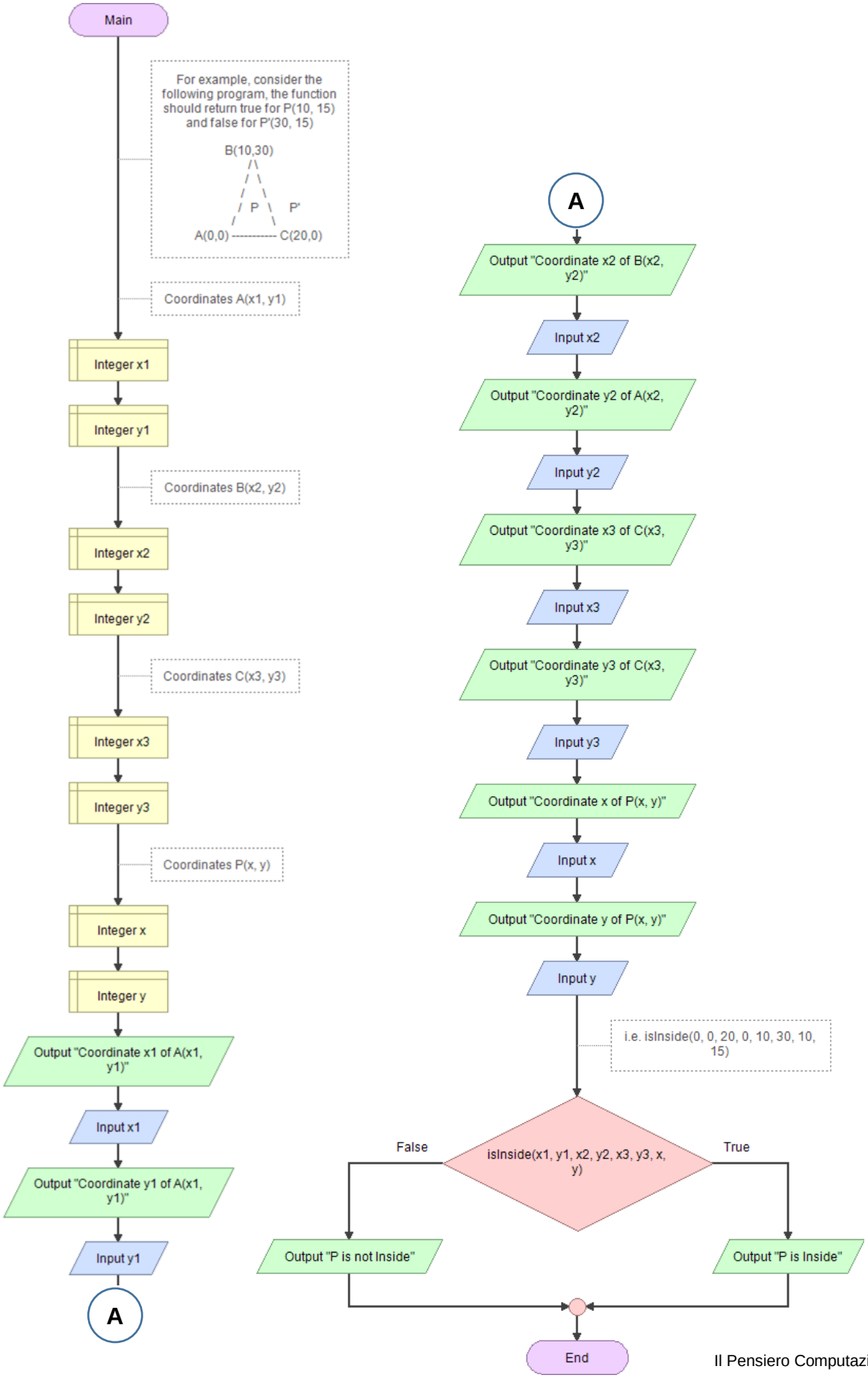
Program to display simplified ATM transaction

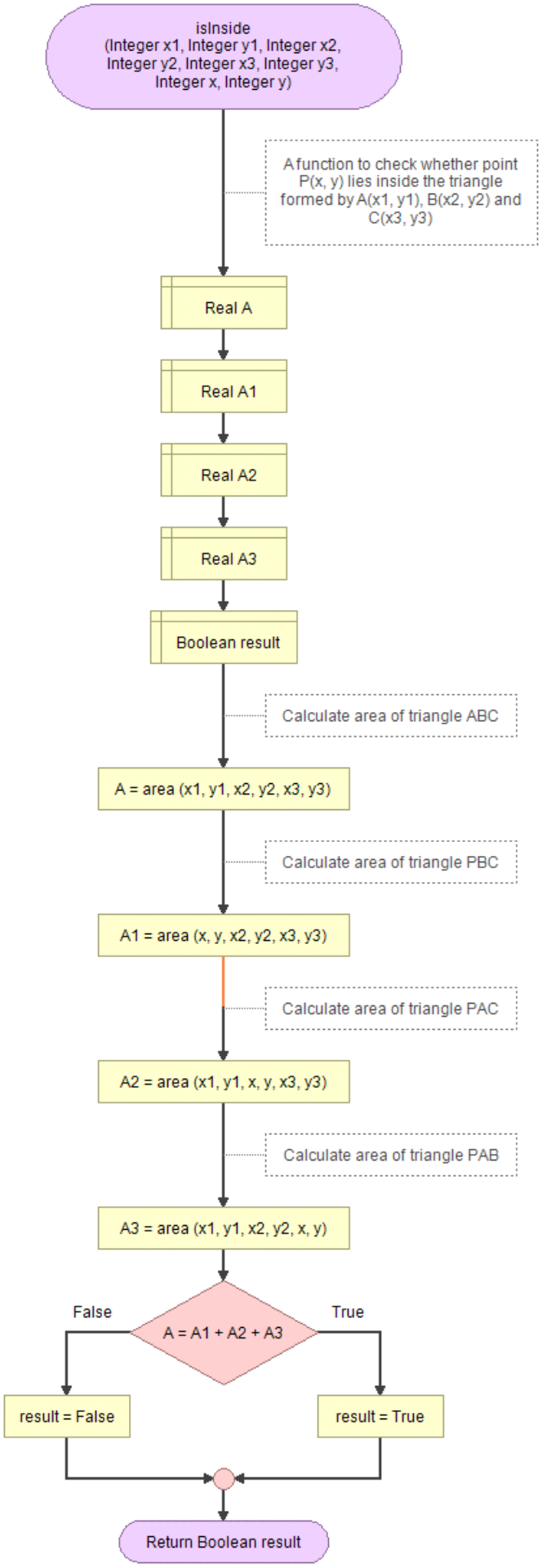


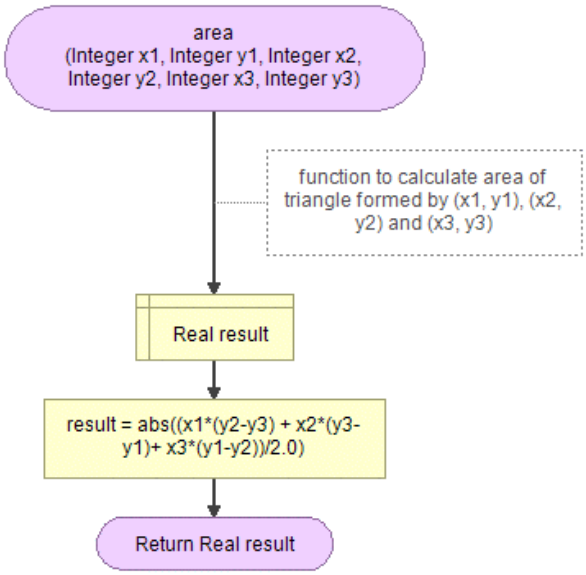

A
menu()
Input choice
choice = 1
False
True
Output "[IT] NEL TUO CONTO CI SONO: " & AMOUNT & ToChar (13) & ToChar(10) & "[EN] YOUR BALANCE IN Rs: " & amount
choice = 2
False
True
Output "[IT] DIGITA L'AMMONTARE DA PRELEVARE: " & AMOUNT & ToChar(13) & ToChar(10) & "[EN] ENTER THE AMOUNT TO WITHDRAW:"
Input withdraw
withdraw MOD 100 <> 0
False
True
Output "[IT] PER FAVORE, INSERISCI L'AMMONTARE IN MULTIPLI DI 100" & AMOUNT & ToChar(13) & ToChar(10) & "[EN] PLEASE ENTER THE AMOUNT IN MULTIPLES OF 100"
withdraw >(amount - 500)
False
True
amount = amount - withdraw
Output "[IT] PER FAVORE RITIRA I SOLDI" & AMOUNT & ToChar(13) & ToChar(10) & "[EN] PLEASE COLLECT CASH"
wait("[IT] ATTENDERE, PREGO [EN] PLEASE, WAIT", 100000)
Output "[IT] NEL TUO CONTO CI SONO: " & AMOUNT & ToChar (13) & ToChar(10) & "[EN] YOUR CURRENT BALANCE IS: " & amount
Output "[IT] BILANCIO INSUFFICIENTE" & ToChar(13) & ToChar(10) & "[EN] INSUFFICENT BALANCE"
choice = 3
False
True
Output "[IT] DIGITA L'AMMONTARE DA DEPOSITARE" & ToChar(13) & ToChar(10) & "[EN] ENTER THE AMOUNT TO DEPOSIT"
Input deposit
amount = amount + deposit
wait("[IT] ATTENDERE, PREGO [EN] PLEASE, WAIT", 100000)
Output "[IT] NEL TUO CONTO CI SONO: " & AMOUNT & ToChar (13) & ToChar(10) & "[EN] YOUR CURRENT BALANCE IS: " & amount
choice = 4
False
True
Output "[IT] VUOI ESEGUIRE UN'ALTRA TRANSAZIONE? (s/n)" & ToChar(13) & ToChar(10) & "[EN] DO YOU WISH TO HAVE ANOTHER TRANSCATION?(y/n)"
Input transaction
(transaction = "n" OR transaction = "N")
False
True
Output "[IT] RIPRENDI IL TUO BANCOMAT" & ToChar(13) & ToChar(10) & "[EN] RETIRE YOUR CREDIT CARD"
wait("[IT] ATTENDERE, PREGO [EN] PLEASE, WAIT", 100000)
Output "[IT] GRAZIE PER AVER USATO I NOSTRI SERVIZI ATM" & ToChar(13) & ToChar(10) & "[EN] THANK YOU USING ATM"
EXIT = True
True
EXIT = False
False
End

9. **G F** ‖ Verificare se un punto P è all'interno di un triangolo

Check whether a given point lies inside a triangle or not

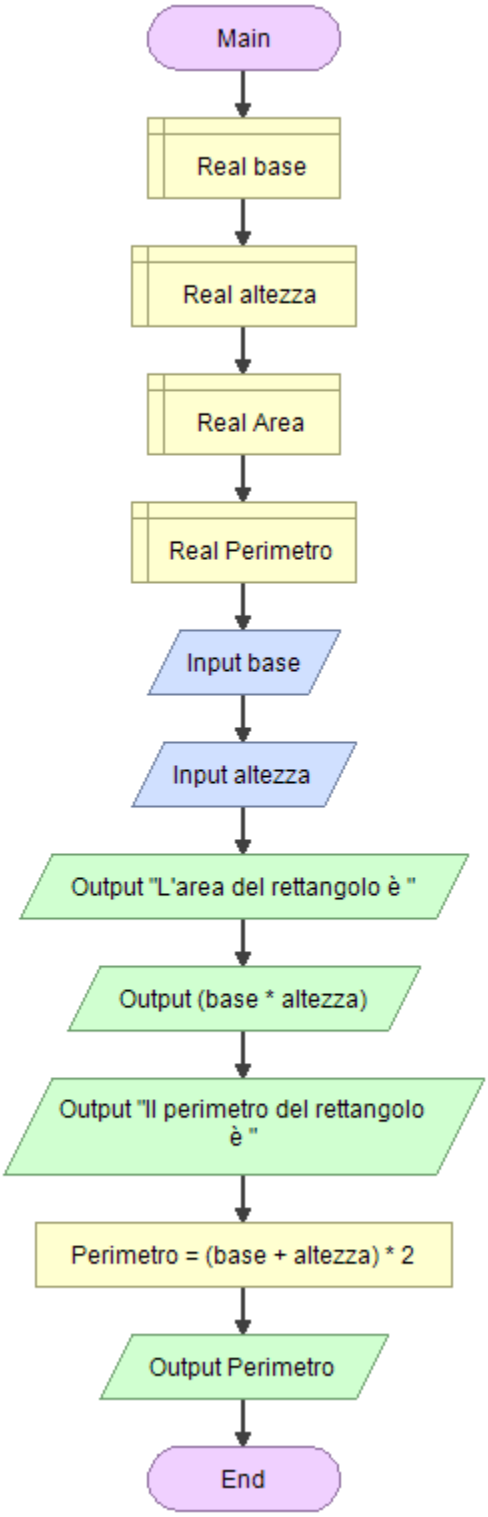



isInside
(Integer x1, Integer y1, Integer x2, Integer y2, Integer x3, Integer y3, Integer x, Integer y)
A function to check whether point P(x, y) lies inside the triangle formed by A(x1, y1), B(x2, y2) and C(x3, y3)
Real A
Real A1
Real A2
Real A3
Boolean result
Calculate area of triangle ABC
A = area (x1, y1, x2, y2, x3, y3)
Calculate area of triangle PBC
A1 = area (x, y, x2, y2, x3, y3)
Calculate area of triangle PAC
A2 = area (x1, y1, x, y, x3, y3)
Calculate area of triangle PAB
A3 = area (x1, y1, x2, y2, x, y)
False
A = A1 + A2 + A3
True
result = False
result = True
Return Boolean result


area
(Integer x1, Integer y1, Integer x2, Integer y2, Integer x3, Integer y3)
function to calculate area of triangle formed by (x1, y1), (x2, y2) and (x3, y3)
Real result
result = abs((x1*(y2-y3) + x2*(y3-y1)+ x3*(y1-y2))/2.0)
Return Real result

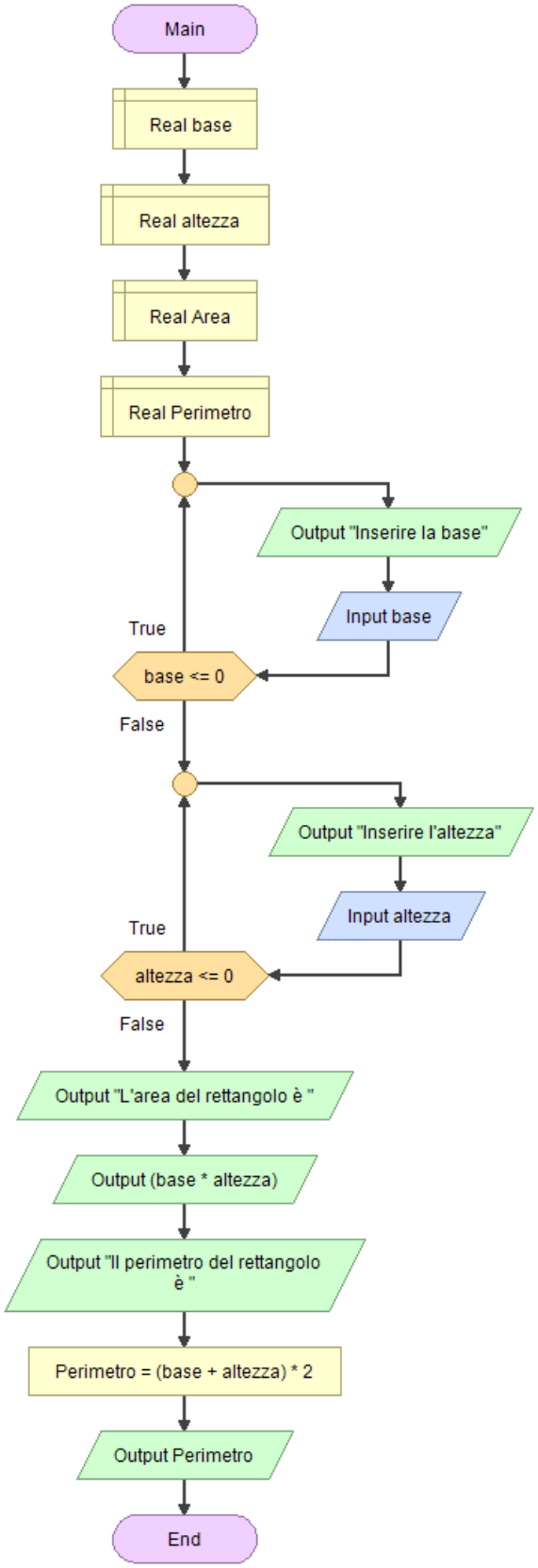
10. G ‖ Data la base e l'altezza di un rettangolo scrivere l'area e il perimetro

Calculate a rectangle's area and perimeter



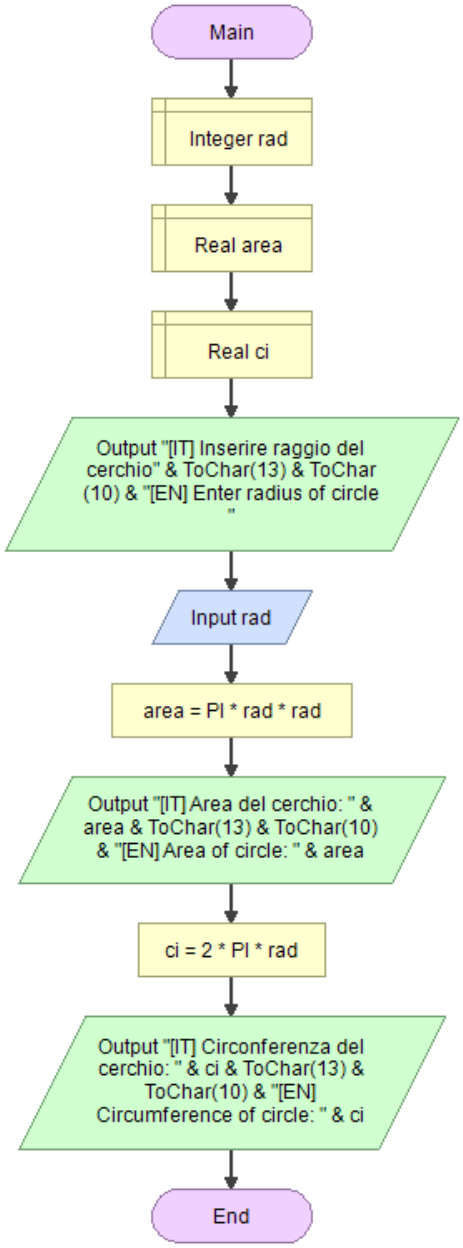
11. G ‖ Calcolare l'area e il perimetro di un rettangolo conoscendo la base e l'altezza (con controllo misure positive)

Calculate a rectangle's area and perimeter checking positives measures



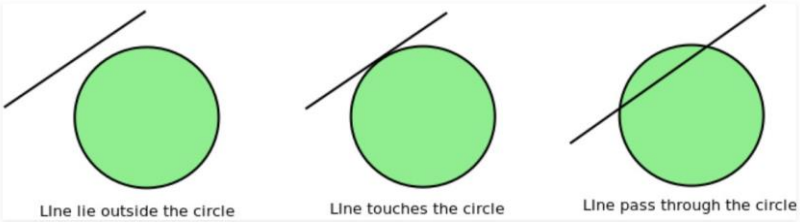
12. G ‖ Calcolo area e circonferenza di un cerchio

Area Calculation and circumference of a circle



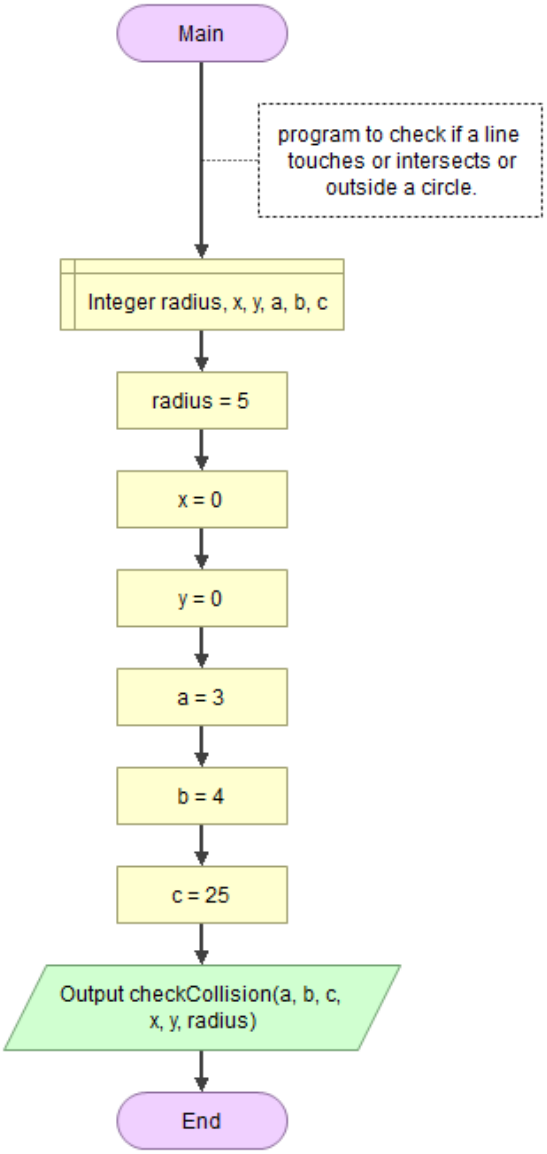
13. **G F S** ‖ Verificare se una retta è tangente, secante o è esterna ad un cerchio

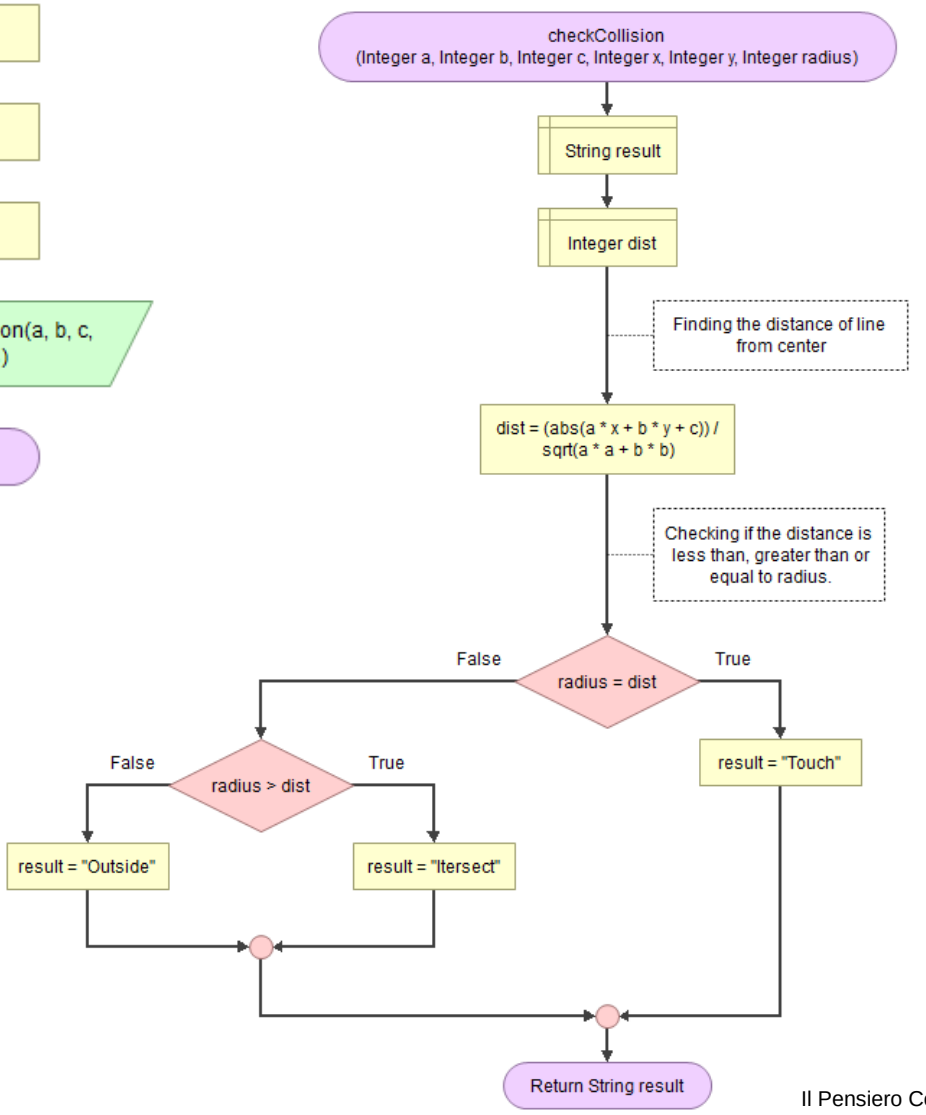
Check if a line touches or intersects a circle



$$\text{Distance} = \frac{|ax0 + by0 + c|}{\sqrt{a^2 + b^2}}$$

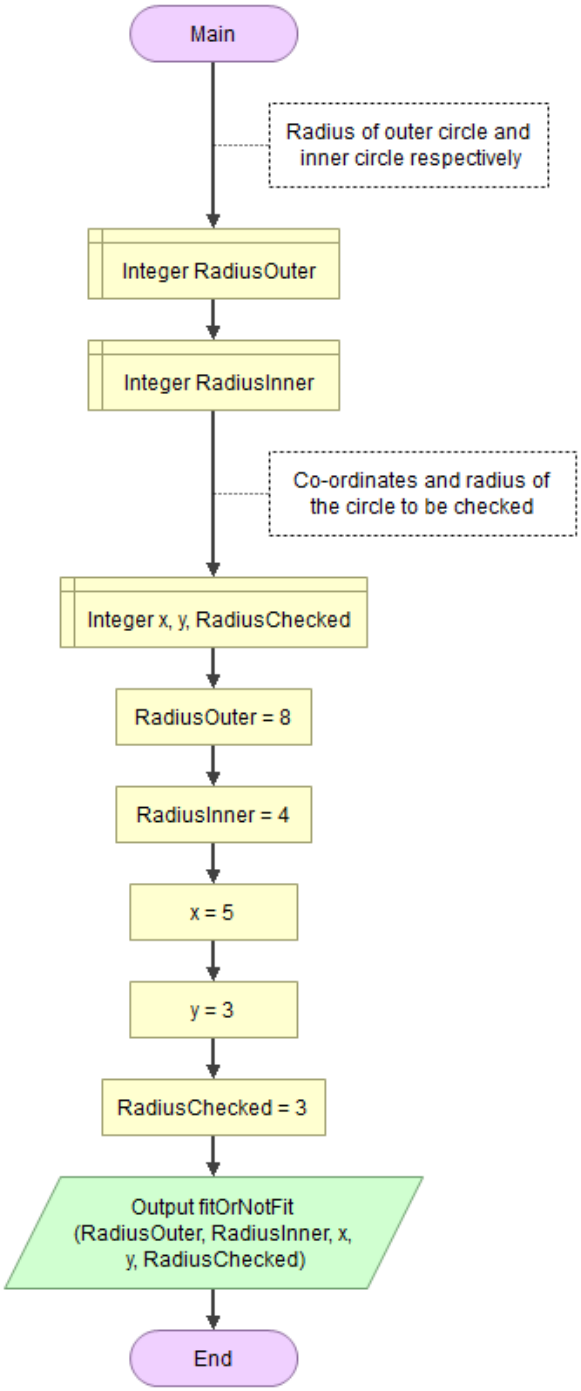
```
Input : radius = 5, center = (0, 0),
        a = 1, b = -1, c = 0.
Output : Intersect

Input :  radius = 5, center = (0, 0),
         a = 5, b = 0, c = 0.
Output : Touch

Input : radius = 5, center = (0, 0),
         a = 1, b = 1, c = -16.
Output : Outside
```

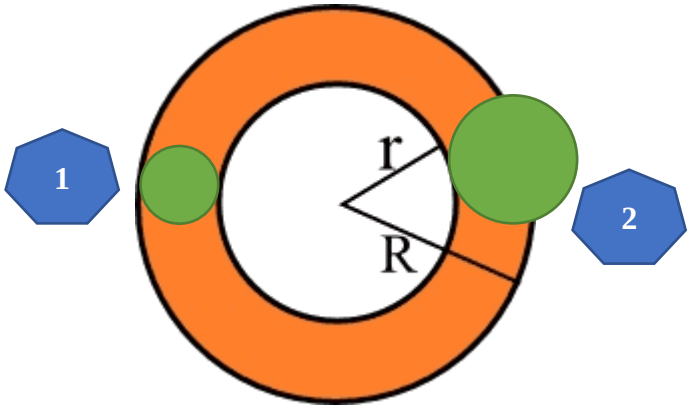




14. **G F S** ‖ Verificare se un determinato cerchio risiede nella corona di un altro cerchio (o due cerchi concentrici)

Check whether given circle resides in boundary maintained by two other circles



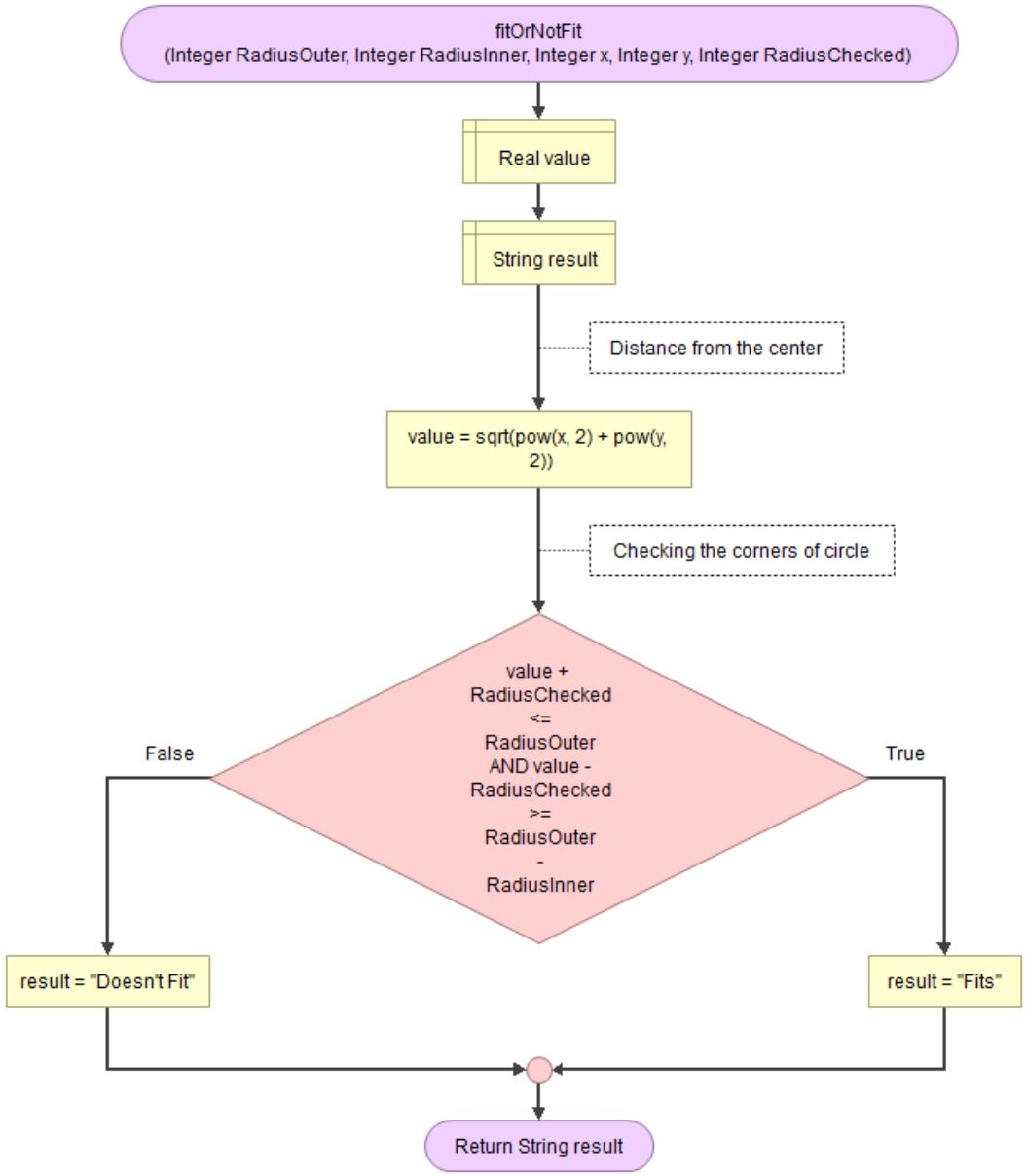


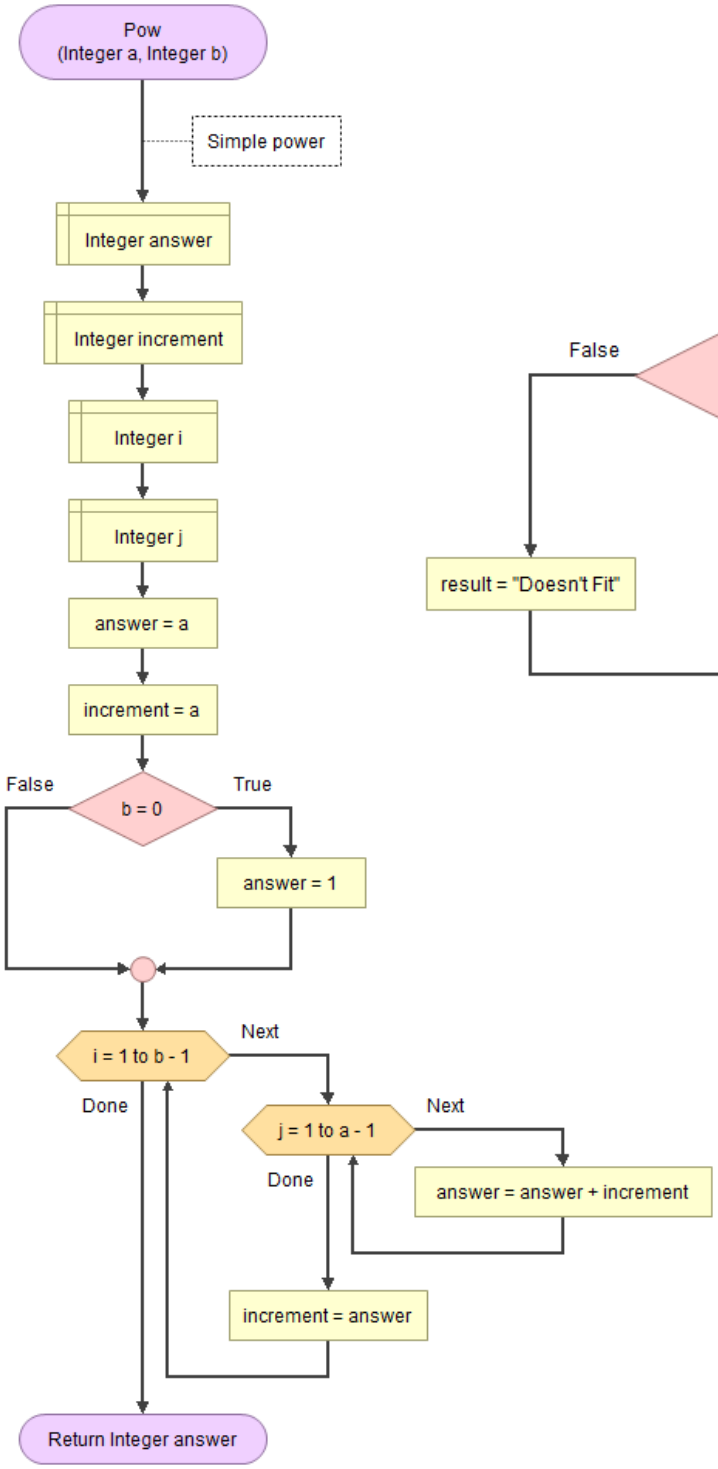
1

```
Input : R = 8, r = 4
x = 5, y = 3, rad = 1
Output : Fits
```

2

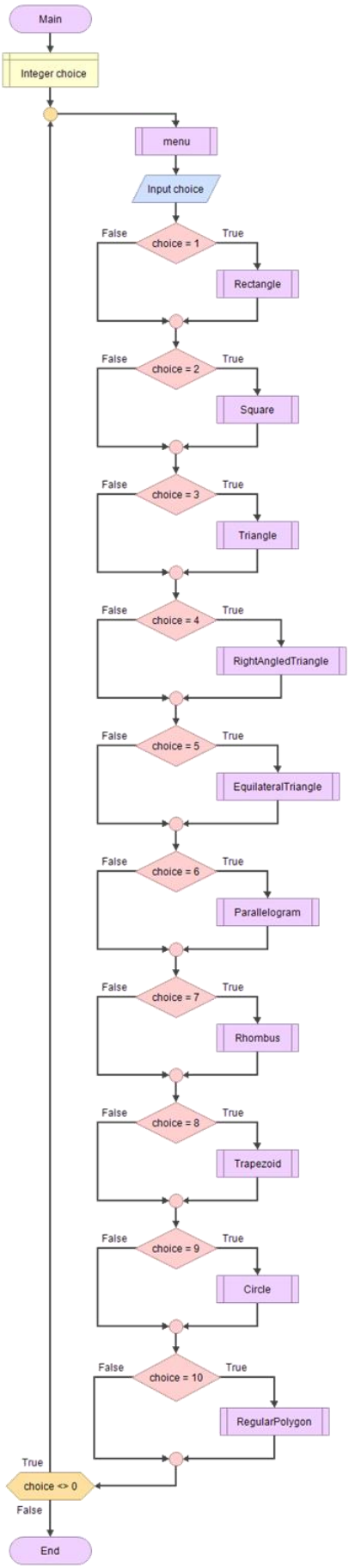
```
Input :  R = 8, r = 4
x = 5, y = 3,  rad = 3.
Output : Doesn't Fit
```

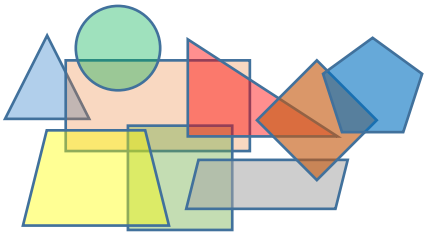

fitOrNotFit
(Integer RadiusOuter, Integer RadiusInner, Integer x, Integer y, Integer RadiusChecked)
Real value
String result
Distance from the center
value = sqrt(pow(x, 2) + pow(y, 2))
Checking the corners of circle
value + RadiusChecked <= RadiusOuter AND value - RadiusChecked >= RadiusOuter - RadiusInner
False
True
result = "Doesn't Fit"
result = "Fits"
Return String result

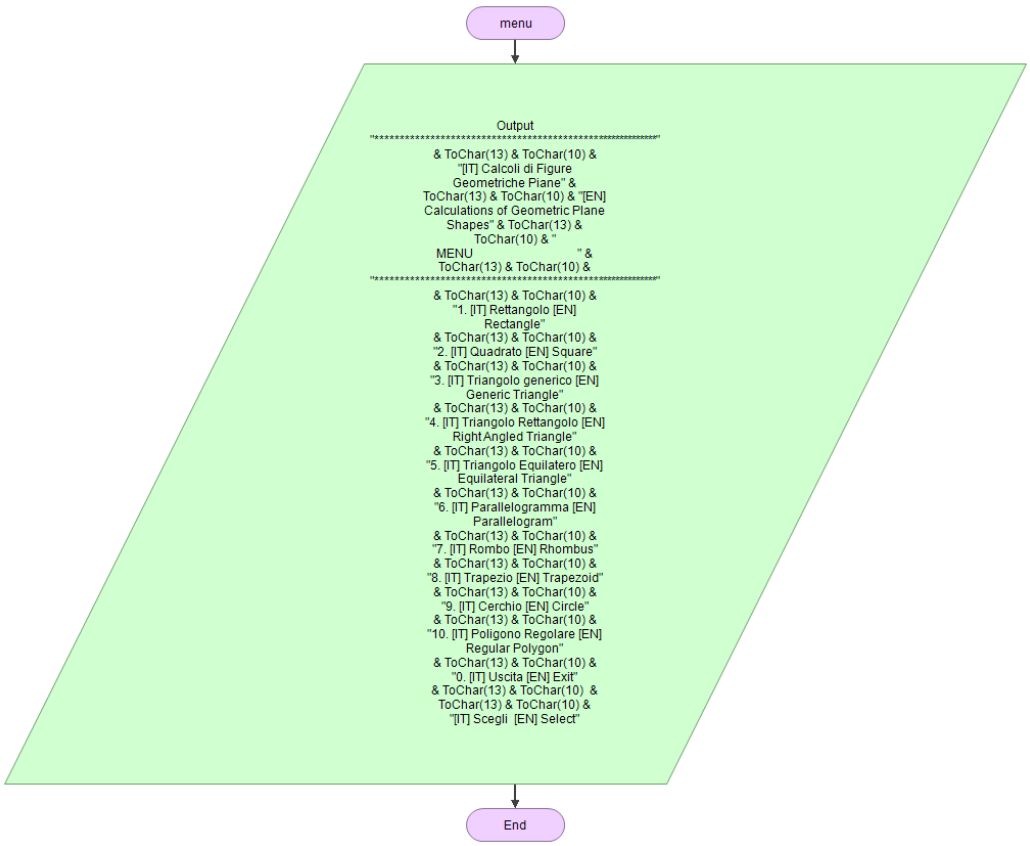

Pow
(Integer a, Integer b)
Simple power
Integer answer
Integer increment
Integer i
Integer j
answer = a
increment = a
False
b = 0
True
answer = 1
i = 1 to b - 1
Next
Done
j = 1 to a - 1
Next
Done
answer = answer + increment
increment = answer
Return Integer answer

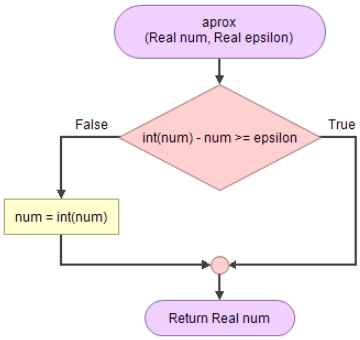
15. G F | Figure Geometriche Piane

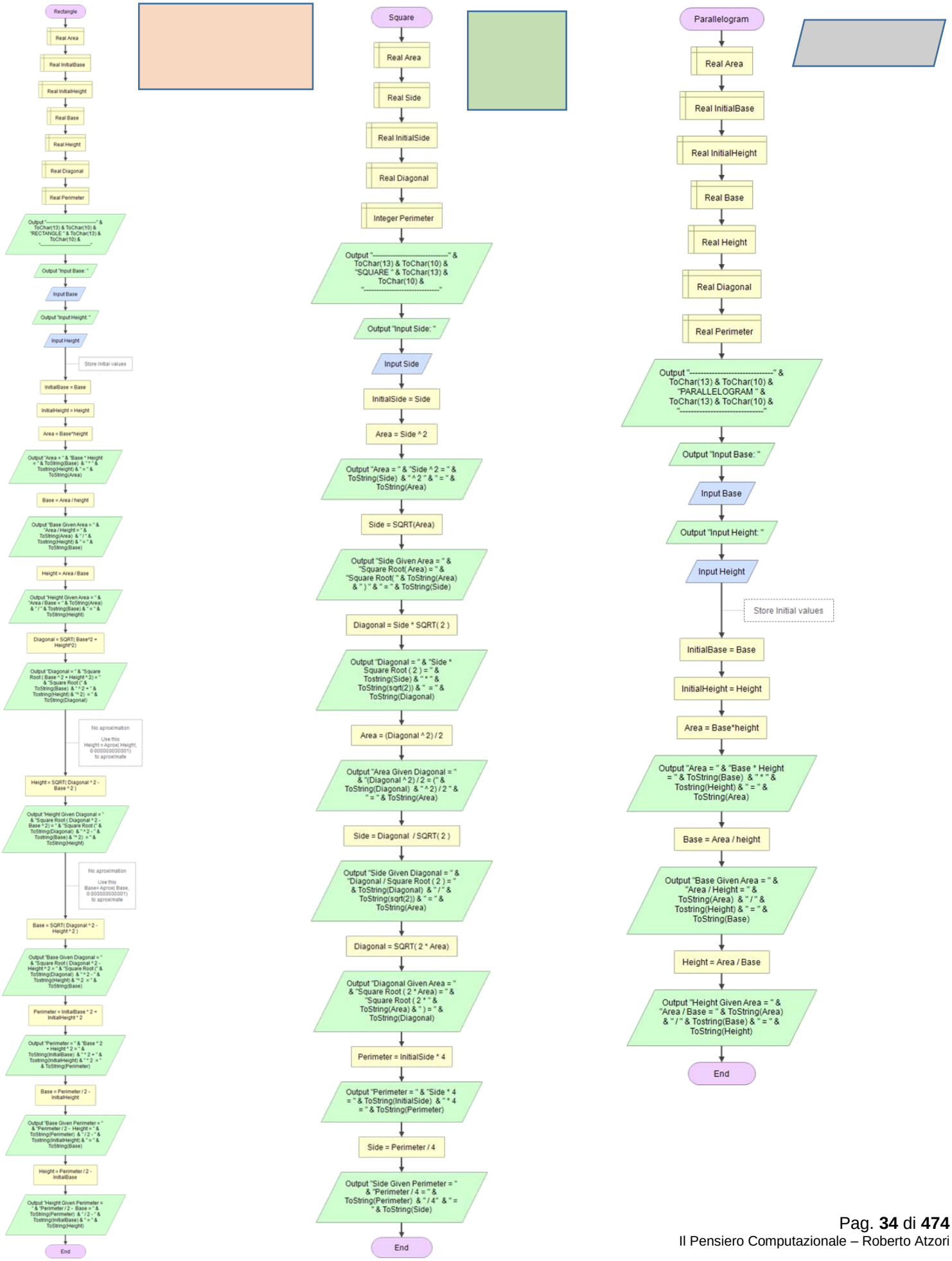
Calculations of Geometric Plane Shapes

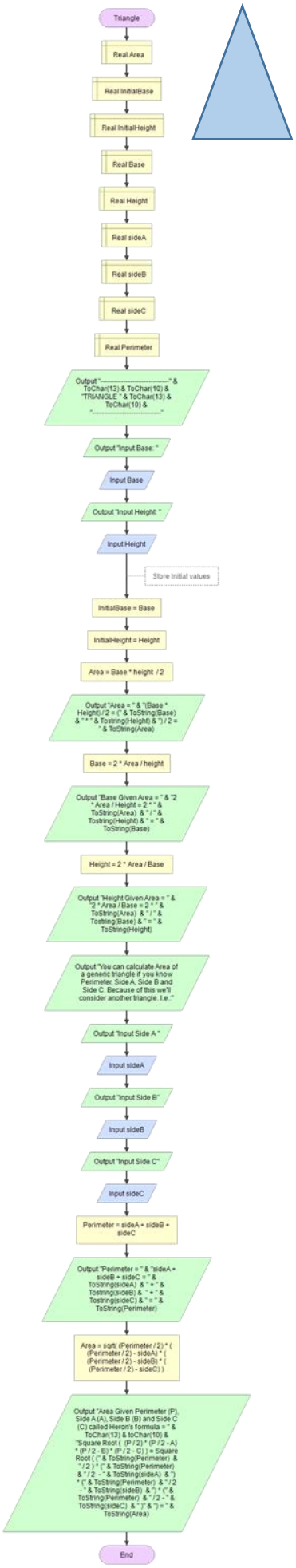


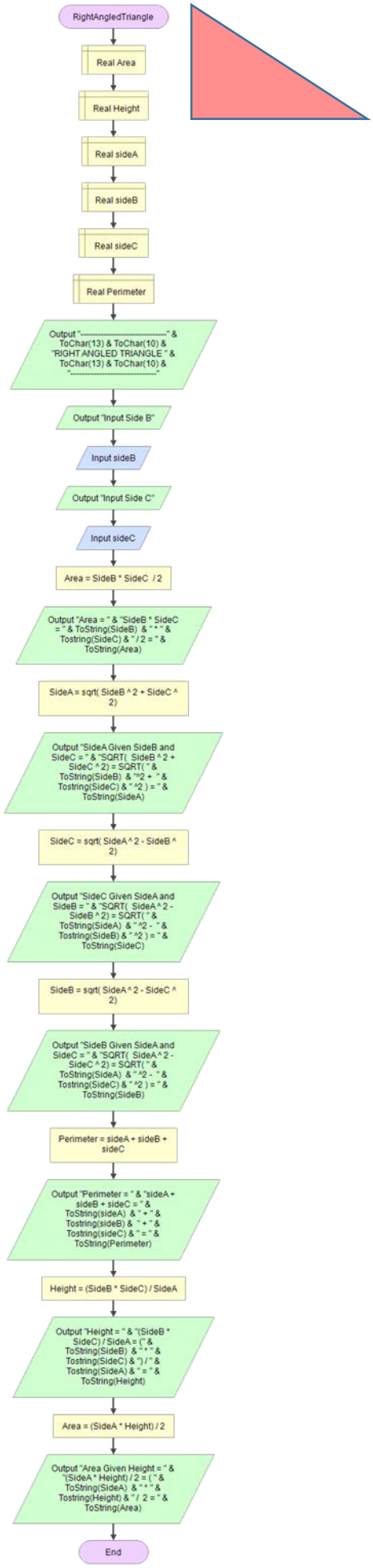


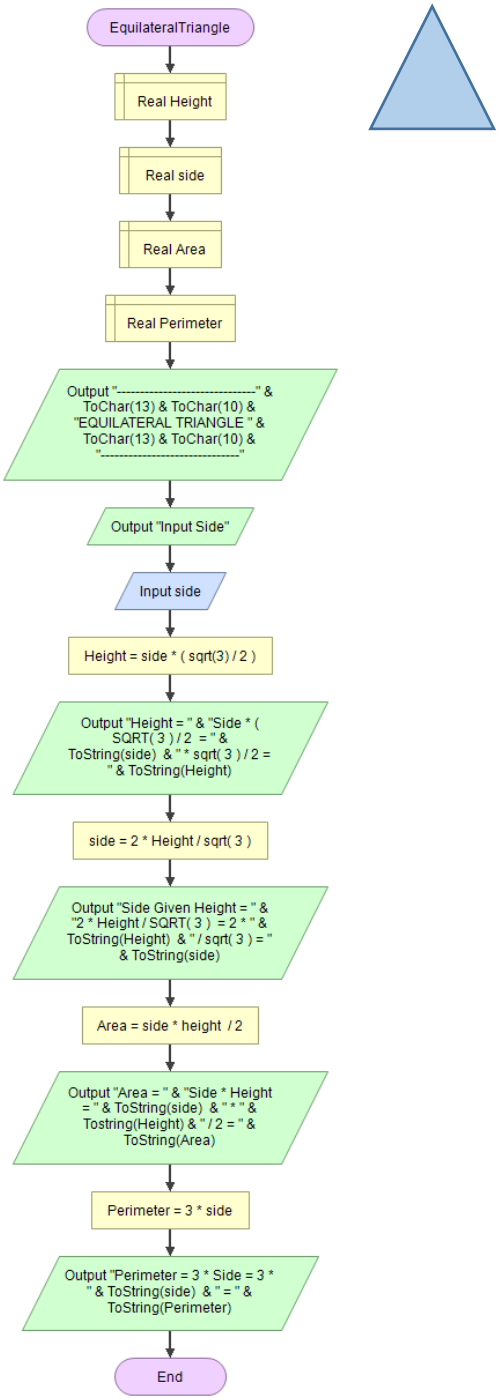


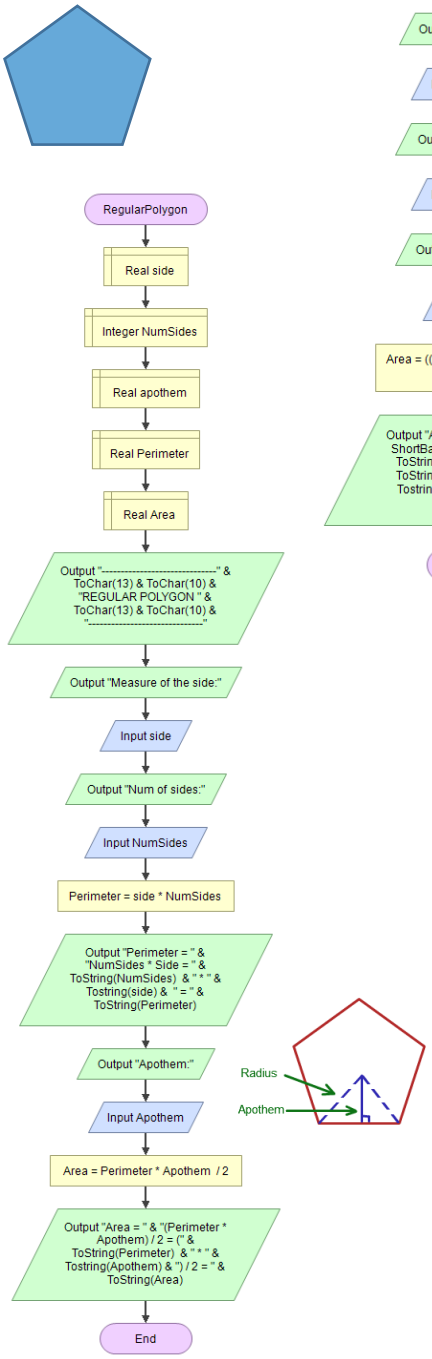


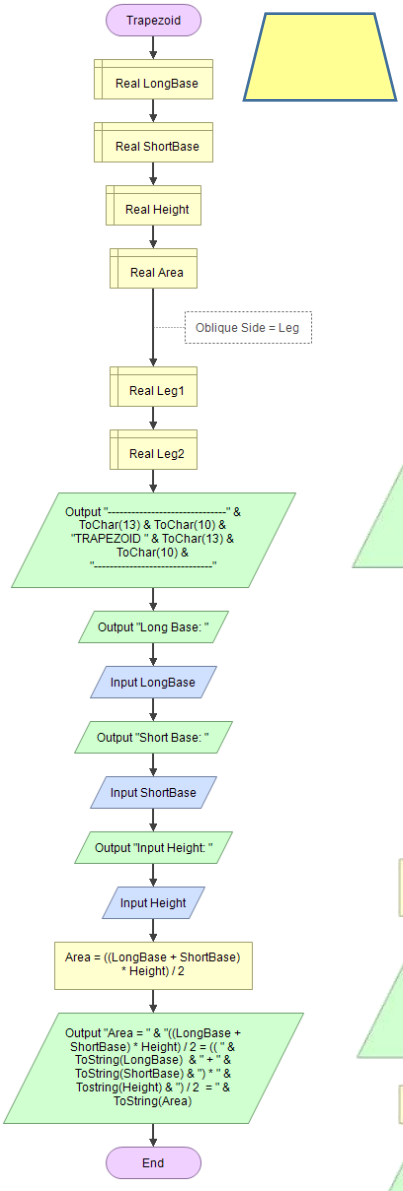

Rectangle
Real Area
Real InitialBase
Real InitialHeight
Real Base
Real Height
Real Diagonal
Real Perimeter
Output "----------" & ToChar(13) & ToChar(10) & "RECTANGLE " & ToChar(13) & ToChar(10) & "----------"
Output "Input Base: "
Input Base
Output "Input Height: "
Input Height
Store initial values
InitialBase = Base
InitialHeight = Height
Area = Base*height
Output "Area = " & "Base * Height = " & ToString(Base) & " * " & Tostring(Height) & " = " & ToString(Area)
Base = Area / height
Output "Base Given Area = " & "Area / Height = " & ToString(Area) & " / " & Tostring(Height) & " = " & ToString(Base)
Height = Area / Base
Output "Height Given Area = " & "Area / Base = " & ToString(Area) & " / " & Tostring(Base) & " = " & ToString(Height)
Diagonal = SQRT(Base^2 + Height^2)
Output "Diagonal = " & "Square Root (Base ^ 2 + Height ^ 2) = " & "Square Root (" & ToString(Base) & " ^ 2 + " & Tostring(Height) & "^2) = " & ToString(Diagonal)
No aproximation
Use this Height = Aprox(Height, 0.000000000001) to aproximate
Height = SQRT(Diagonal ^ 2 - Base ^ 2)
Output "Height Given Diagonal = " & "Square Root (Diagonal ^ 2 - Base ^ 2) = " & "Square Root (" & ToString(Diagonal) & " ^ 2 - " & Tostring(Base) & "^2) = " & ToString(Height)
No aproximation
Use this Base= Aprox(Base, 0.000000000001) to aproximate
Base = SQRT(Diagonal ^ 2 - Height ^ 2)
Output "Base Given Diagonal = " & "Square Root (Diagonal ^ 2 - Height ^ 2) = " & "Square Root (" & ToString(Diagonal) & " ^ 2 - " & Tostring(Height) & "^ 2 = " & ToString(Base)
Perimeter = InitialBase * 2 + InitialHeight * 2
Output "Perimeter = " & "Base * 2 + Height * 2 = " & ToString(InitialBase) & " * 2 + " & Tostring(InitialHeight) & " * 2 = " & ToString(Perimeter)
Base = Perimeter / 2 - InitialHeight
Output "Base Given Perimeter = " & "Perimeter / 2 - Height = " & ToString(Perimeter) & " / 2 - " & Tostring(InitialHeight) & " = " & ToString(Base)
Height = Perimeter / 2 - InitialBase
Output "Height Given Perimeter = " & "Perimeter / 2 - Base = " & ToString(Perimeter) & " / 2 - " & Tostring(InitialBase) & " = " & ToString(Height)
End
Square
Real Area
Real Side
Real InitialSide
Real Diagonal
Integer Perimeter
Output "----------" & ToChar(13) & ToChar(10) & "SQUARE " & ToChar(13) & ToChar(10) & "----------"
Output "Input Side: "
Input Side
InitialSide = Side
Area = Side ^ 2
Output "Area = " & "Side ^ 2 = " & ToString(Side) & " ^ 2 " & " = " & ToString(Area)
Side = SQRT(Area)
Output "Side Given Area = " & "Square Root(Area) = " & "Square Root(" & ToString(Area) & ") " & " = " & ToString(Side)
Diagonal = Side * SQRT(2)
Output "Diagonal = " & "Side * Square Root (2) = " & Tostring(Side) & " * " & ToString(sqrt(2)) & " = " & ToString(Diagonal)
Area = (Diagonal ^ 2) / 2
Output "Area Given Diagonal = " & "(Diagonal ^ 2) / 2 = (" & ToString(Diagonal) & " ^ 2) / 2 " & " = " & ToString(Area)
Side = Diagonal / SQRT(2)
Output "Side Given Diagonal = " & "Diagonal / Square Root (2) = " & ToString(Diagonal) & " / " & ToString(sqrt(2)) & " = " & ToString(Area)
Diagonal = SQRT(2 * Area)
Output "Diagonal Given Area = " & "Square Root (2 * Area) = " & "Square Root (2 * " & ToString(Area) & ") = " & ToString(Diagonal)
Perimeter = InitialSide * 4
Output "Perimeter = " & "Side * 4 = " & ToString(InitialSide) & " * 4 = " & ToString(Perimeter)
Side = Perimeter / 4
Output "Side Given Perimeter = " & "Perimeter / 4 = " & ToString(Perimeter) & " / 4" & " = " & ToString(Side)
End
Parallelogram
Real Area
Real InitialBase
Real InitialHeight
Real Base
Real Height
Real Diagonal
Real Perimeter
Output "----------" & ToChar(13) & ToChar(10) & "PARALLELOGRAM " & ToChar(13) & ToChar(10) & "----------"
Output "Input Base: "
Input Base
Output "Input Height: "
Input Height
Store Initial values
InitialBase = Base
InitialHeight = Height
Area = Base*height
Output "Area = " & "Base * Height = " & ToString(Base) & " * " & Tostring(Height) & " = " & ToString(Area)
Base = Area / height
Output "Base Given Area = " & "Area / Height = " & ToString(Area) & " / " & Tostring(Height) & " = " & ToString(Base)
Height = Area / Base
Output "Height Given Area = " & "Area / Base = " & ToString(Area) & " / " & Tostring(Base) & " = " & ToString(Height)
End

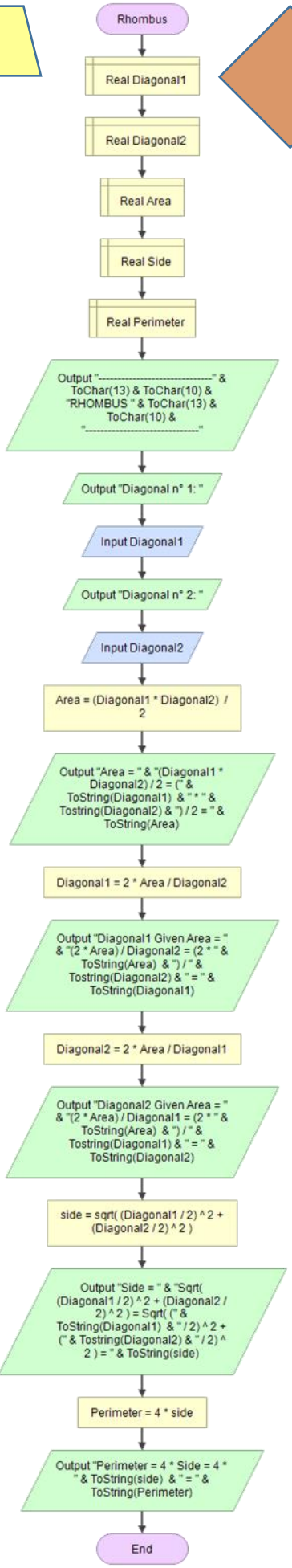

Triangle
Real Area
Real InitialBase
Real InitialHeight
Real Base
Real Height
Real sideA
Real sideB
Real sideC
Real Perimeter
Output "----------" & ToChar(13) & ToChar(10) & "TRIANGLE " & ToChar(13) & ToChar(10) & "----------"
Output "Input Base: "
Input Base
Output "Input Height: "
Input Height
Store Initial values
InitialBase = Base
InitialHeight = Height
Area = Base * height / 2
Output "Area = " & "(Base * Height) / 2 = (" & ToString(Base) & " * " & Tostring(Height) & ") / 2 = " & ToString(Area)
Base = 2 * Area / height
Output "Base Given Area = " & "2 * Area / Height = 2 * " & ToString(Area) & " / " & Tostring(Height) & " = " & ToString(Base)
Height = 2 * Area / Base
Output "Height Given Area = " & "2 * Area / Base = 2 * " & ToString(Area) & " / " & Tostring(Base) & " = " & ToString(Height)
Output "You can calculate Area of a generic triangle if you know Perimeter, Side A, Side B and Side C. Because of this we'll consider another triangle. I.e.:"
Output "Input Side A "
Input sideA
Output "Input Side B"
Input sideB
Output "Input Side C"
Input sideC
Perimeter = sideA + sideB + sideC
Output "Perimeter = " & "sideA + sideB + sideC = " & ToString(sideA) & " + " & Tostring(sideB) & " + " & Tostring(sideC) & " = " & ToString(Perimeter)
Area = sqrt((Perimeter / 2) * ((Perimeter / 2) - sideA) * ((Perimeter / 2) - sideB) * ((Perimeter / 2) - sideC))
Output "Area Given Perimeter (P), Side A (A), Side B (B) and Side C (C) called Heron's formula = " & ToChar(13) & toChar(10) & "Square Root ((P / 2) * (P / 2 - A) * (P / 2 - B) * (P / 2 - C)) = Square Root ((" & ToString(Perimeter) & " / 2) * (" & ToString(Perimeter) & " / 2 - " & ToString(sideA) & ") * (" & ToString(Perimeter) & " / 2 - " & ToString(sideB) & ") * (" & ToString(Perimeter) & " / 2 - " & ToString(sideC) & ")) " & " = " & ToString(Area)
End

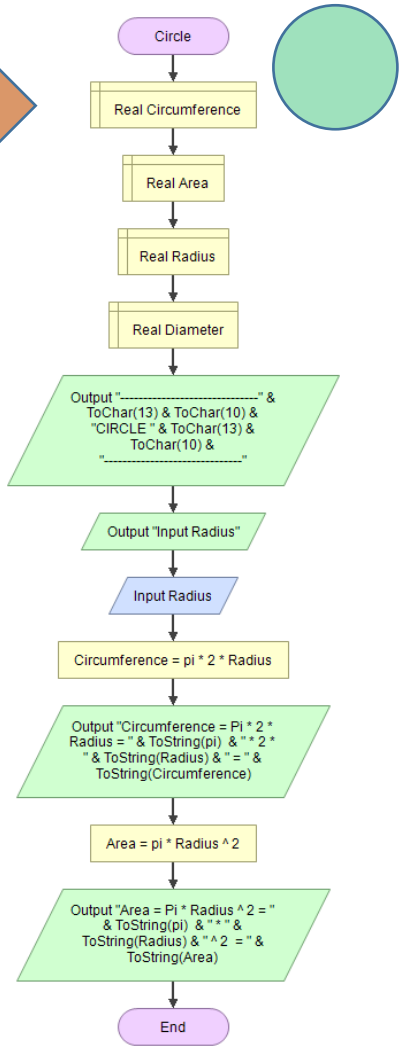

RightAngledTriangle
Real Area
Real Height
Real sideA
Real sideB
Real sideC
Real Perimeter
Output "----------" & ToChar(13) & ToChar(10) & "RIGHT ANGLED TRIANGLE " & ToChar(13) & ToChar(10) & "----------"
Output "Input Side B"
Input sideB
Output "Input Side C"
Input sideC
Area = SideB * SideC / 2
Output "Area = " & "SideB * SideC = " & ToString(SideB) & " * " & Tostring(SideC) & " / 2 = " & ToString(Area)
SideA = sqrt(SideB ^ 2 + SideC ^ 2)
Output "SideA Given SideB and SideC = " & "SQRT(SideB ^ 2 + SideC ^ 2) = SQRT(" & ToString(SideB) & "^2 + " & Tostring(SideC) & " ^2) = " & ToString(SideA)
SideC = sqrt(SideA ^ 2 - SideB ^ 2)
Output "SideC Given SideA and SideB = " & "SQRT(SideA ^ 2 - SideB ^ 2) = SQRT(" & ToString(SideA) & " ^2 - " & Tostring(SideB) & " ^2) = " & ToString(SideC)
SideB = sqrt(SideA ^ 2 - SideC ^ 2)
Output "SideB Given SideA and SideC = " & "SQRT(SideA ^ 2 - SideC ^ 2) = SQRT(" & ToString(SideA) & " ^2 - " & Tostring(SideC) & " ^2) = " & ToString(SideB)
Perimeter = sideA + sideB + sideC
Output "Perimeter = " & "sideA + sideB + sideC = " & ToString(sideA) & " + " & Tostring(sideB) & " + " & Tostring(sideC) & " = " & ToString(Perimeter)
Height = (SideB * SideC) / SideA
Output "Height = " & "(SideB * SideC) / SideA = (" & ToString(SideB) & " * " & Tostring(SideC) & ") / " & Tostring(SideA) & " = " & ToString(Height)
Area = (SideA * Height) / 2
Output "Area Given Height = " & "(SideA * Height) / 2 = (" & ToString(SideA) & " * " & Tostring(Height) & " / 2 = " & ToString(Area)
End


EquilateralTriangle
Real Height
Real side
Real Area
Real Perimeter
Output "----------" & ToChar(13) & ToChar(10) & "EQUILATERAL TRIANGLE " & ToChar(13) & ToChar(10) & "----------"
Output "Input Side"
Input side
Height = side * (sqrt(3) / 2)
Output "Height = " & "Side * (SQRT(3) / 2 = " & ToString(side) & " * sqrt(3) / 2 = " & ToString(Height)
side = 2 * Height / sqrt(3)
Output "Side Given Height = " & "2 * Height / SQRT(3) = 2 * " & ToString(Height) & " / sqrt(3) = " & ToString(side)
Area = side * height / 2
Output "Area = " & "Side * Height = " & ToString(side) & " * " & Tostring(Height) & " / 2 = " & ToString(Area)
Perimeter = 3 * side
Output "Perimeter = 3 * Side = 3 * " & ToString(side) & " = " & ToString(Perimeter)
End


RegularPolygon
Real side
Integer NumSides
Real apothem
Real Perimeter
Real Area
Output "----------" & ToChar(13) & ToChar(10) & "REGULAR POLYGON " & ToChar(13) & ToChar(10) & "----------"
Output "Measure of the side:"
Input side
Output "Num of sides:"
Input NumSides
Perimeter = side * NumSides
Output "Perimeter = " & "NumSides * Side = " & ToString(NumSides) & " * " & Tostring(side) & " = " & ToString(Perimeter)
Output "Apothem:"
Input Apothem
Area = Perimeter * Apothem / 2
Output "Area = " & "(Perimeter * Apothem) / 2 = (" & ToString(Perimeter) & " * " & Tostring(Apothem) & ") / 2 = " & ToString(Area)
End
Radius
Apothem

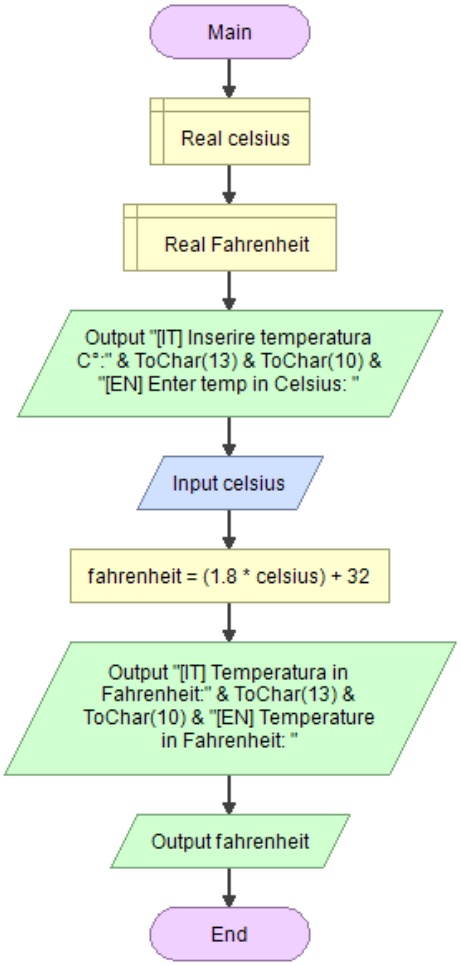

Trapezoid
Real LongBase
Real ShortBase
Real Height
Real Area
Oblique Side = Leg
Real Leg1
Real Leg2
Output "----------" & ToChar(13) & ToChar(10) & "TRAPEZOID " & ToChar(13) & ToChar(10) & "----------"
Output "Long Base: "
Input LongBase
Output "Short Base: "
Input ShortBase
Output "Input Height: "
Input Height
Area = ((LongBase + ShortBase) * Height) / 2
Output "Area = " & "((LongBase + ShortBase) * Height) / 2 = ((" & ToString(LongBase) & " + " & ToString(ShortBase) & ") * " & Tostring(Height) & ") / 2 = " & ToString(Area)
End


Rhombus
Real Diagonal1
Real Diagonal2
Real Area
Real Side
Real Perimeter
Output "----------" & ToChar(13) & ToChar(10) & "RHOMBUS " & ToChar(13) & ToChar(10) & "----------"
Output "Diagonal n° 1: "
Input Diagonal1
Output "Diagonal n° 2: "
Input Diagonal2
Area = (Diagonal1 * Diagonal2) / 2
Output "Area = " & "(Diagonal1 * Diagonal2) / 2 = (" & ToString(Diagonal1) & " * " & Tostring(Diagonal2) & ") / 2 = " & ToString(Area)
Diagonal1 = 2 * Area / Diagonal2
Output "Diagonal1 Given Area = " & "(2 * Area) / Diagonal2 = (2 * " & ToString(Area) & ") / " & Tostring(Diagonal2) & " = " & ToString(Diagonal1)
Diagonal2 = 2 * Area / Diagonal1
Output "Diagonal2 Given Area = " & "(2 * Area) / Diagonal1 = (2 * " & ToString(Area) & ") / " & Tostring(Diagonal1) & " = " & ToString(Diagonal2)
side = sqrt((Diagonal1 / 2) ^ 2 + (Diagonal2 / 2) ^ 2)
Output "Side = " & "Sqrt((Diagonal1 / 2) ^ 2 + (Diagonal2 / 2) ^ 2) = Sqrt((" & ToString(Diagonal1) & " / 2) ^ 2 + (" & Tostring(Diagonal2) & " / 2) ^ 2) = " & ToString(side)
Perimeter = 4 * side
Output "Perimeter = 4 * Side = 4 * " & ToString(side) & " = " & ToString(Perimeter)
End


Circle
Real Circumference
Real Area
Real Radius
Real Diameter
Output "----------" & ToChar(13) & ToChar(10) & "CIRCLE " & ToChar(13) & ToChar(10) & "----------"
Output "Input Radius"
Input Radius
Circumference = pi * 2 * Radius
Output "Circumference = Pi * 2 * Radius = " & ToString(pi) & " * 2 * " & ToString(Radius) & " = " & ToString(Circumference)
Area = pi * Radius ^ 2
Output "Area = Pi * Radius ^ 2 = " & ToString(pi) & " * " & ToString(Radius) & " ^ 2 = " & ToString(Area)
End

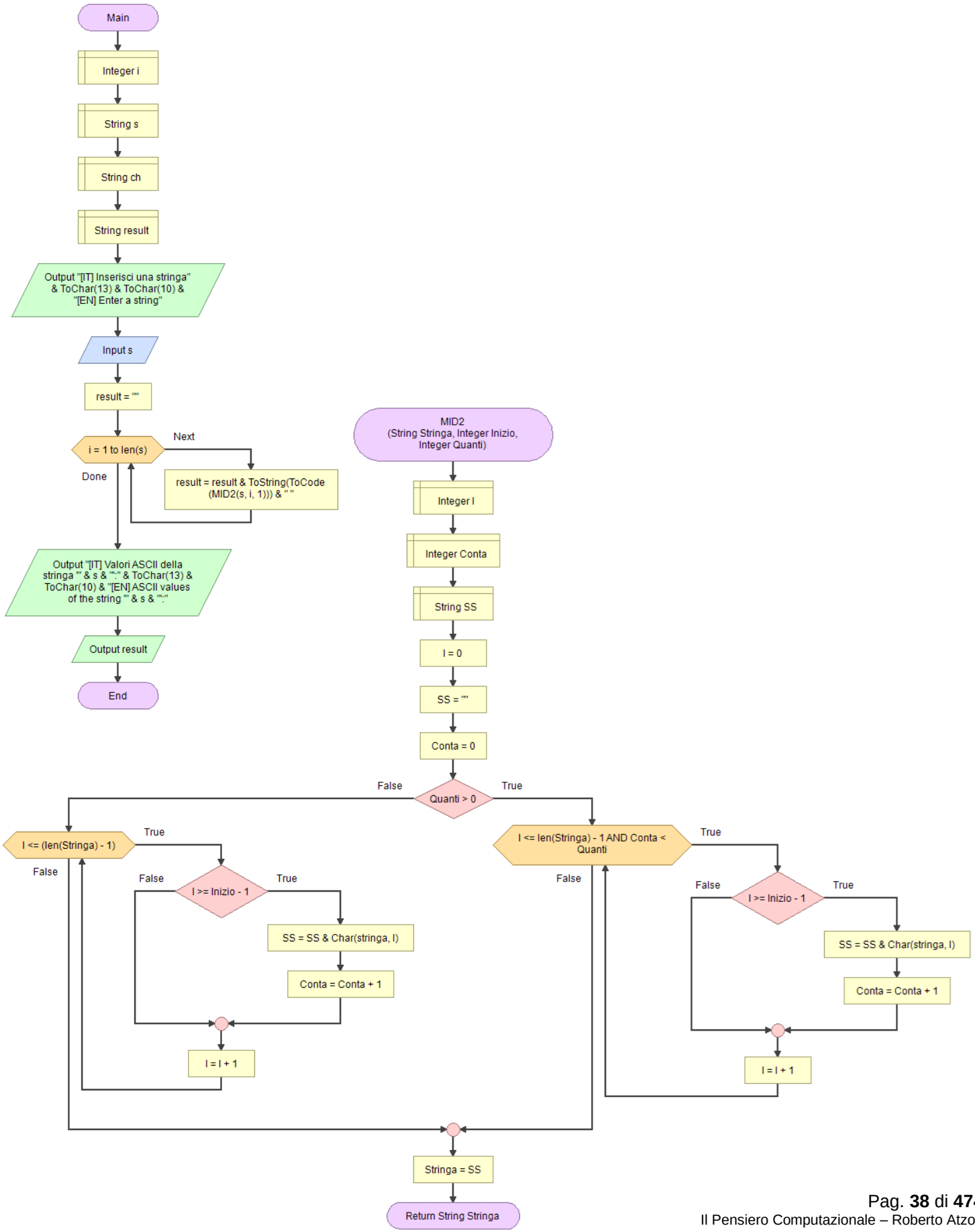
16. G ‖ Conversione gradi centigradi in Fahrenheit

Converting degrees Celsius to Fahrenheit



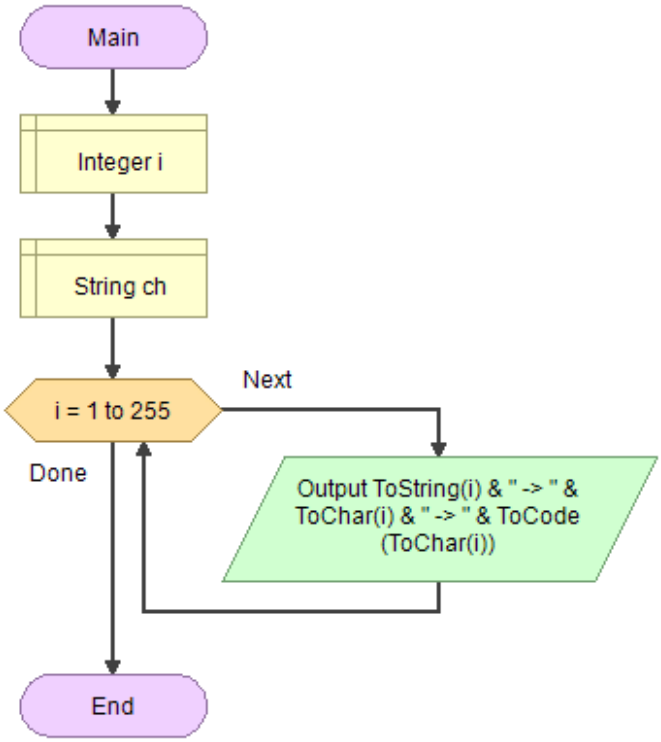
17. **G F** | Converte una stringa nei corrispettivi valori ASCII

Convert string into ASCII values



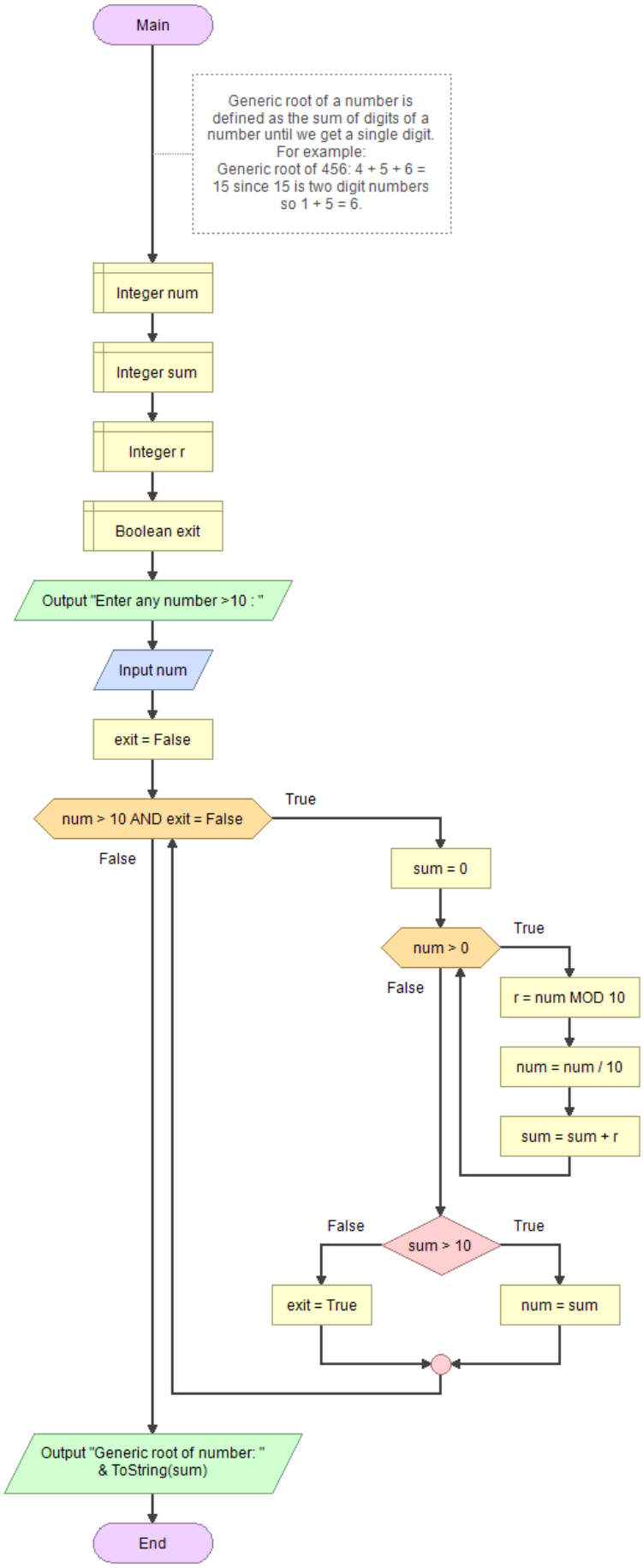
18. G ‖ Tabella dei valori ASCII (come usare le funzioni Flowgorithm ToString, ToChar e ToCode)

Table of ASCII values (how to use the Flowgorithm functions ToString, ToChar and ToCode)



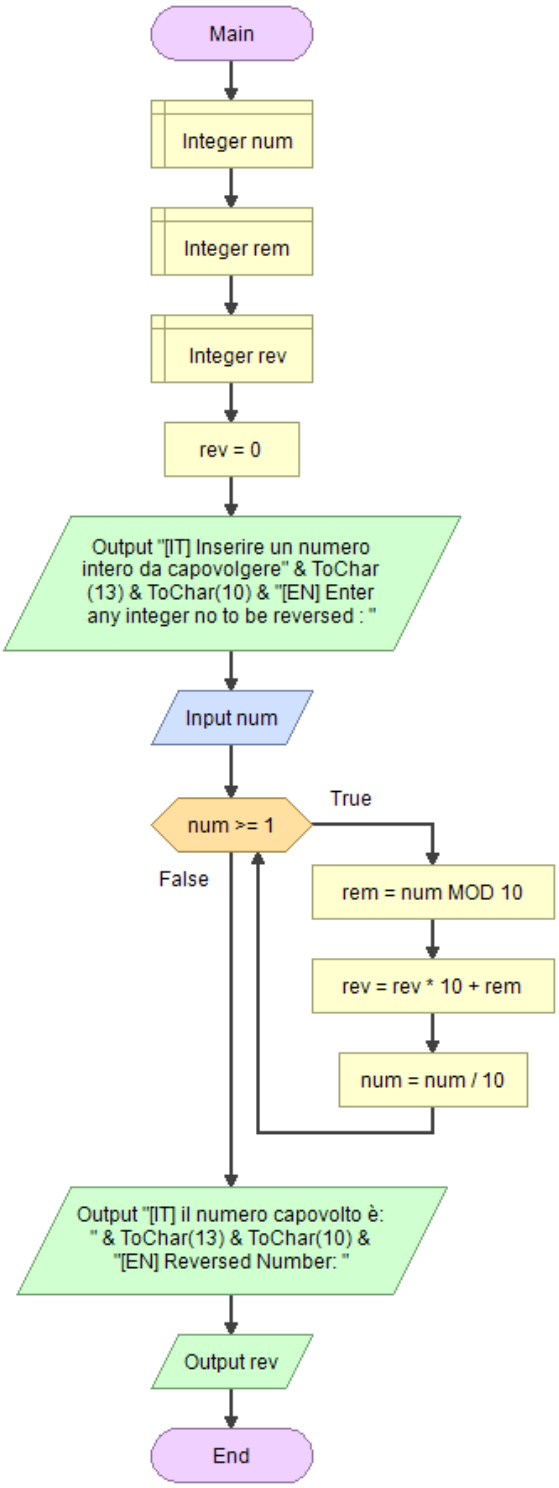
19. G ‖ Somma delle cifre di un numero fino ad ottenere una sola cifra

Sum of digits of a number unit we don't get a single digit



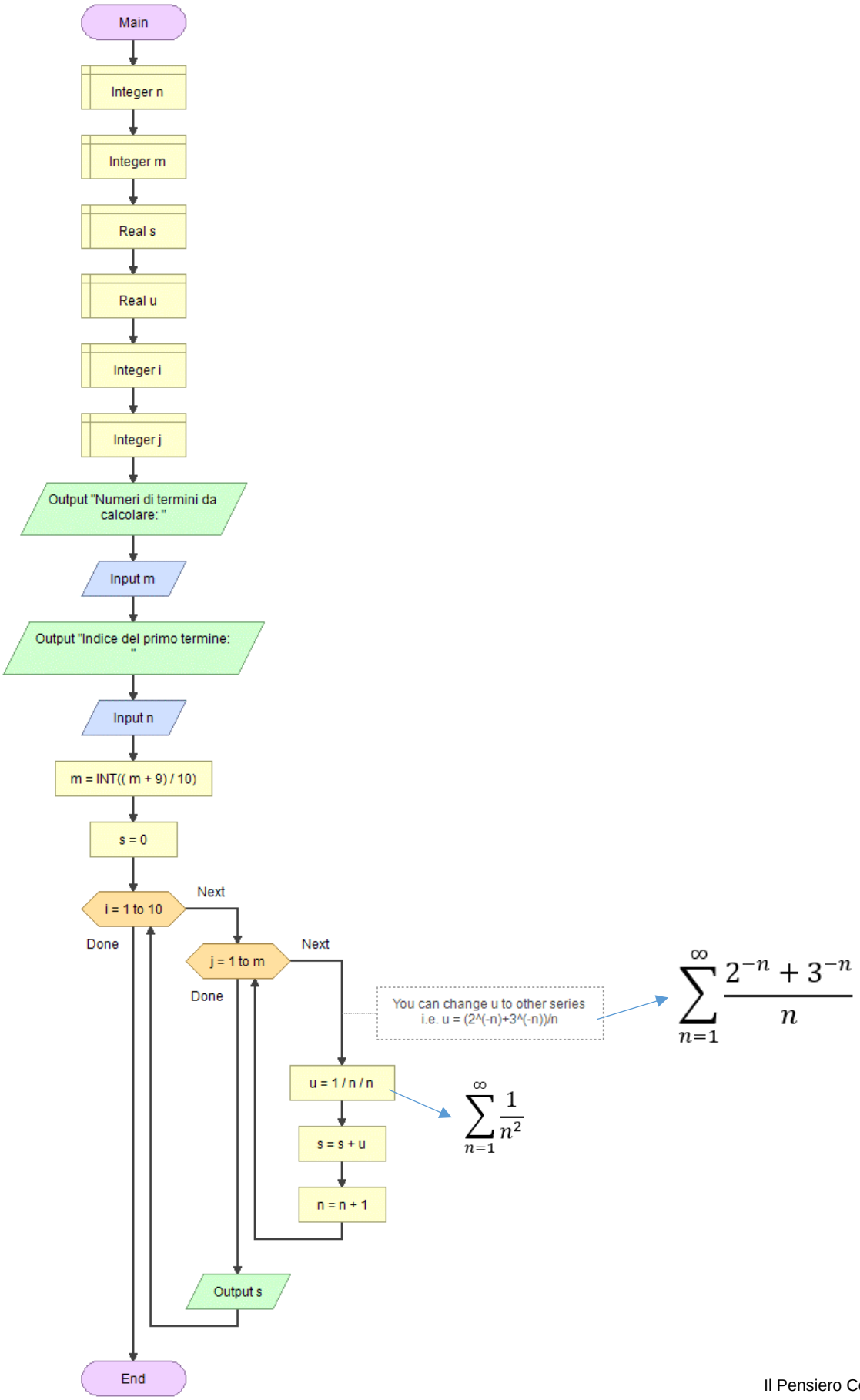
20. G ‖ Inverso di un numero

Reverse a number



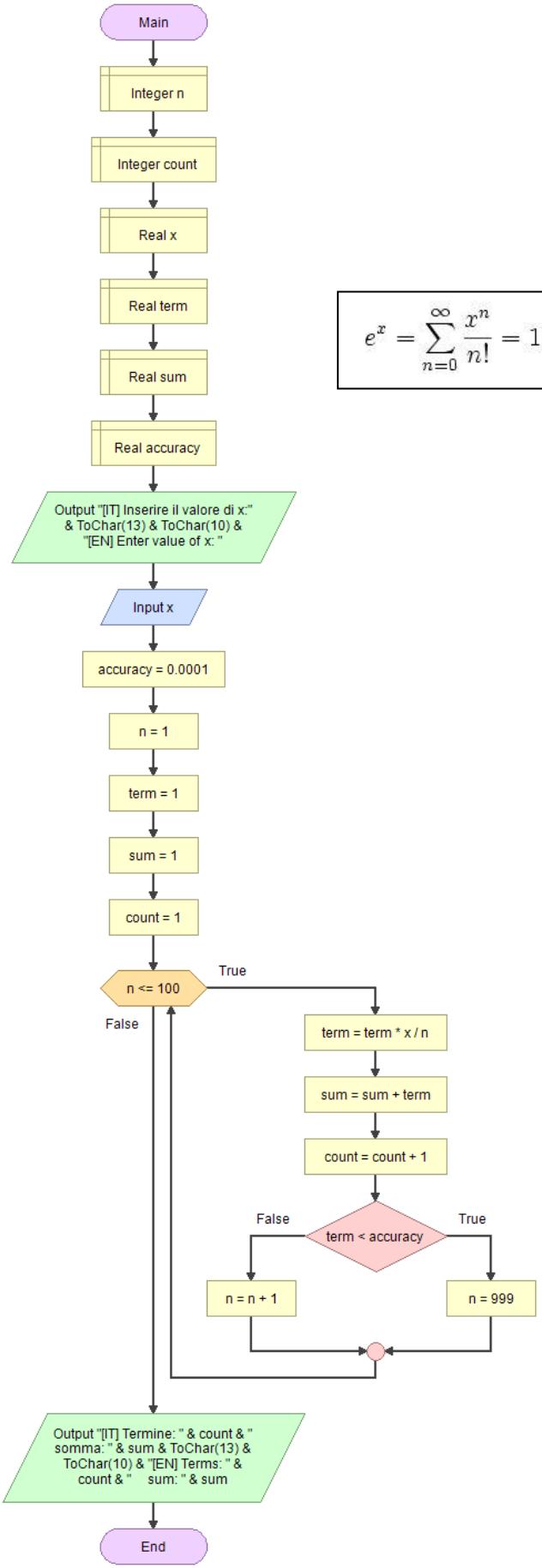
21. G ‖ Calcolo di una serie

Series's calc



22. G ‖ Serie di potenze

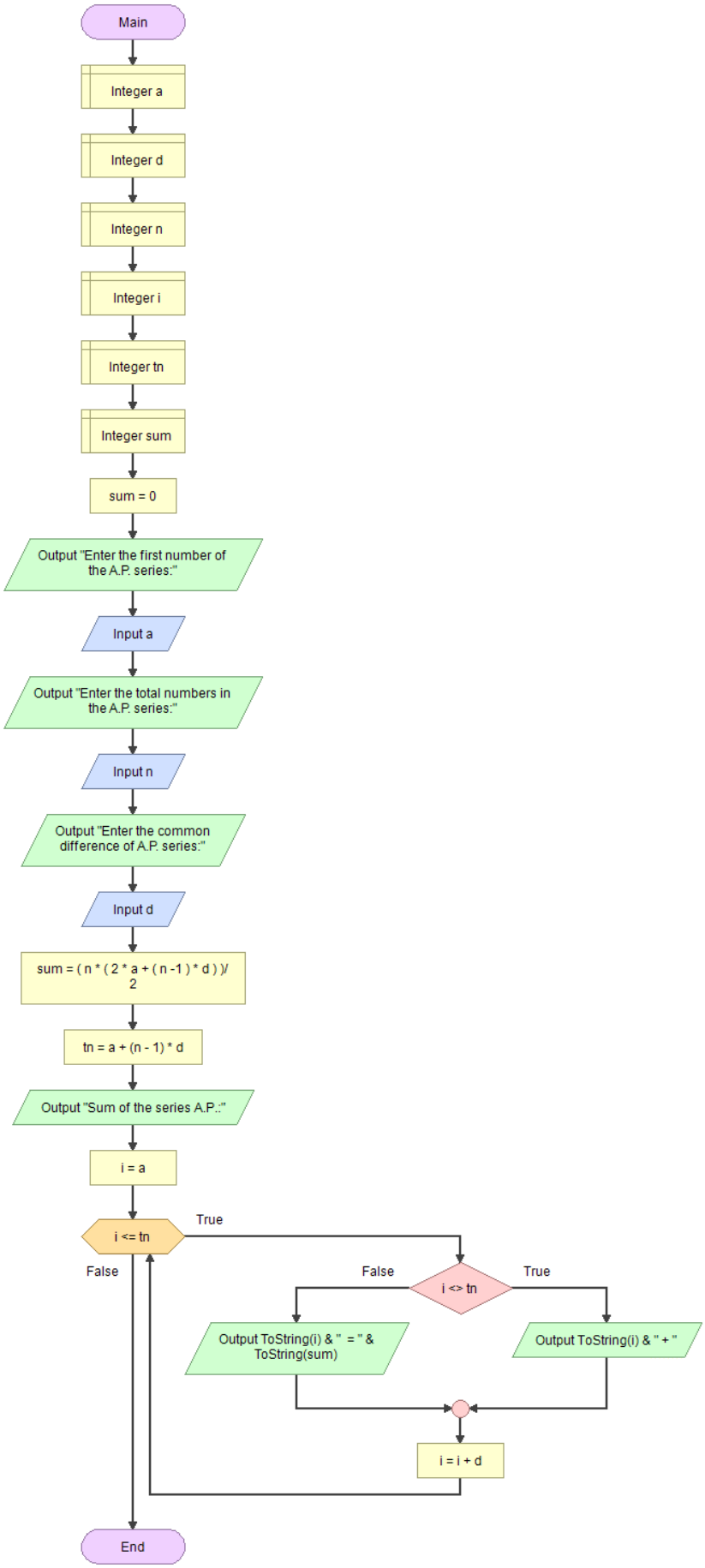
Exponent Power Series



$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

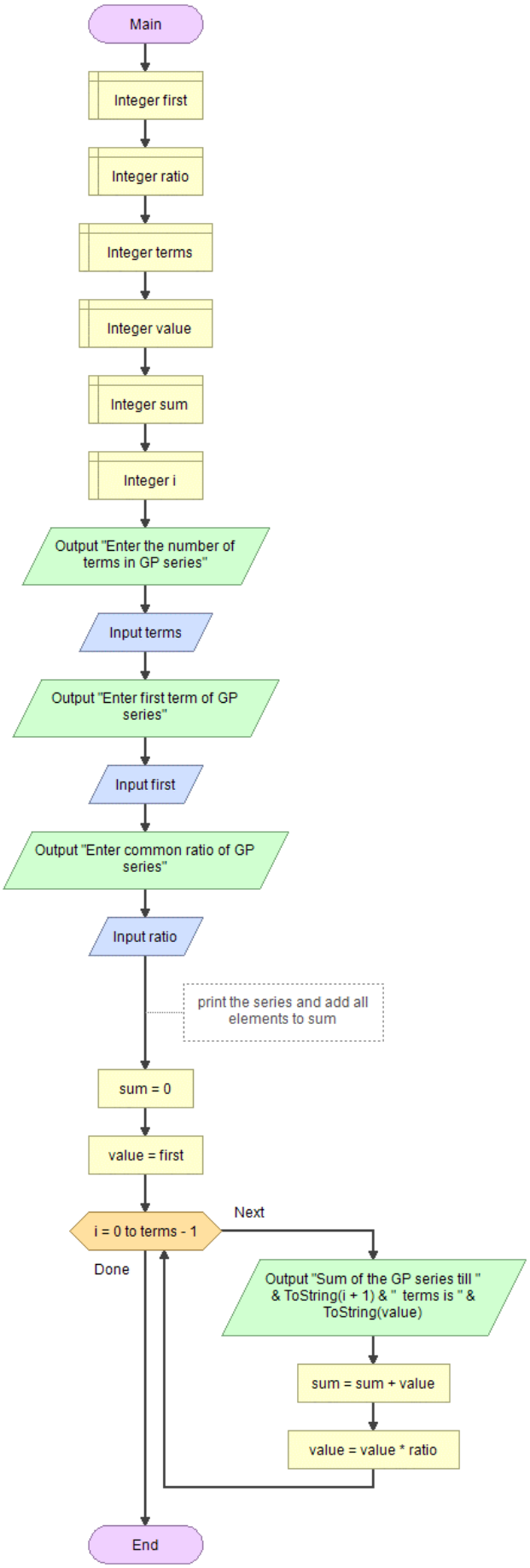
23. G ‖ Progressione aritmetica

arithmetic progression (A.P.)



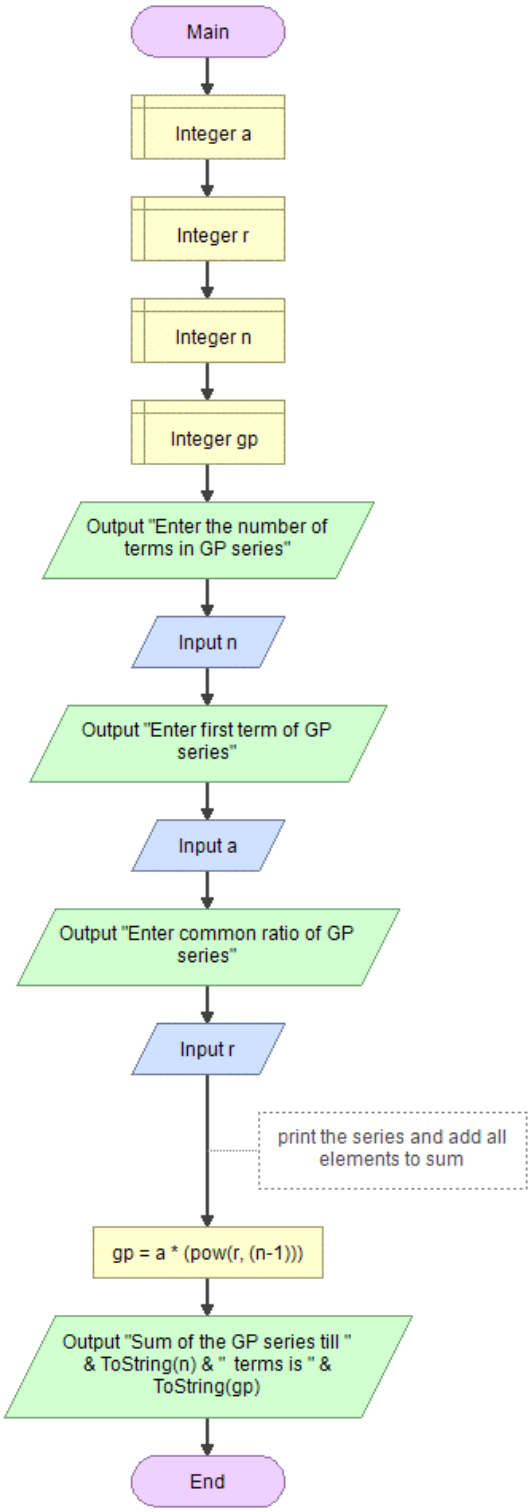
24. G ‖ Progressione geometrica

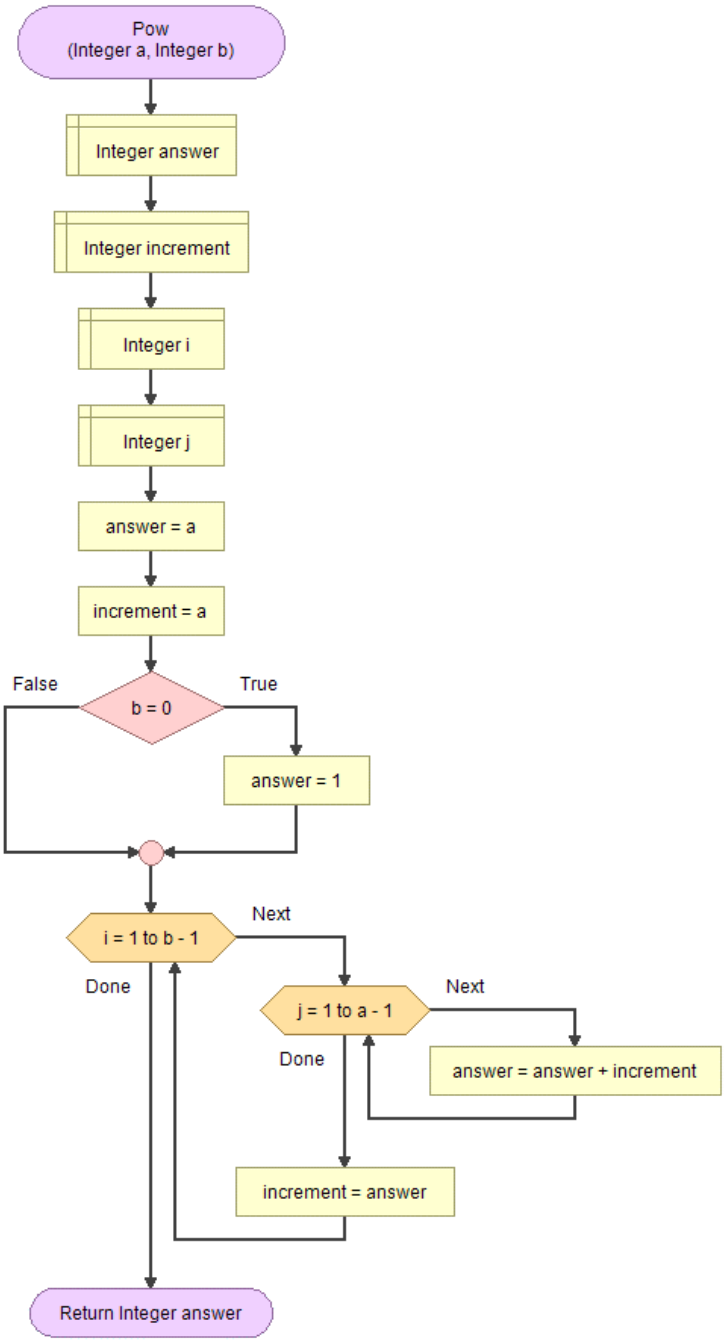
Geometric Progression (G.P.)



25. G F ‖ Progressione geometrica con funzione

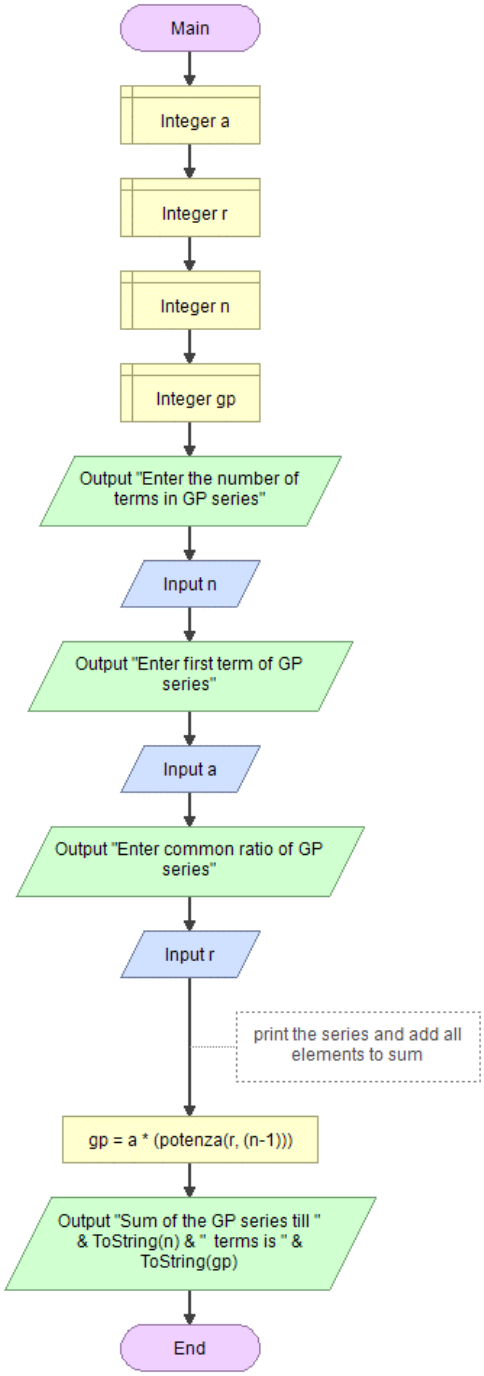
Geometric Progression (G.P.) - with function

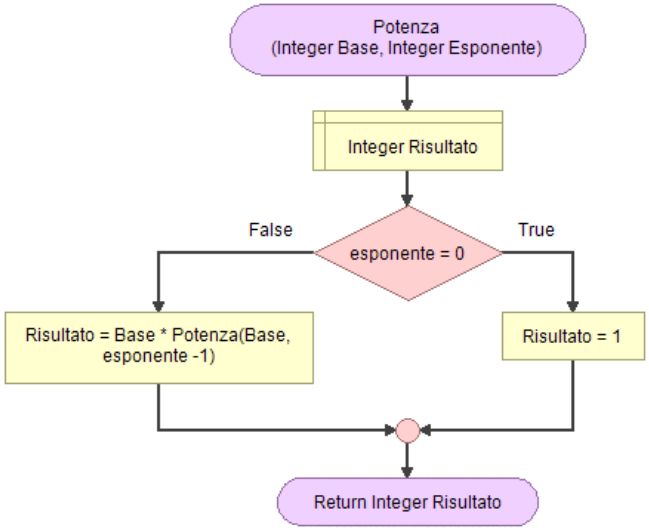




26. G F R | Progressione geometrica con funzione (ricorsiva)

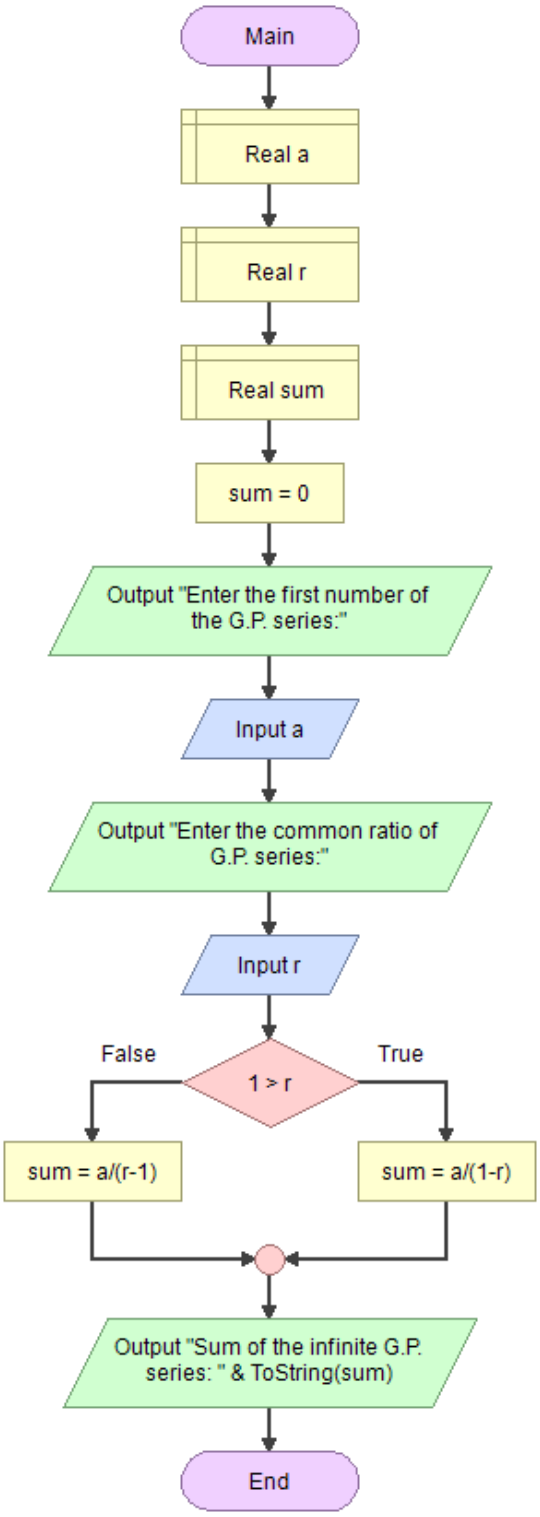
Geometric Progression (G.P.) - with function (recursive)





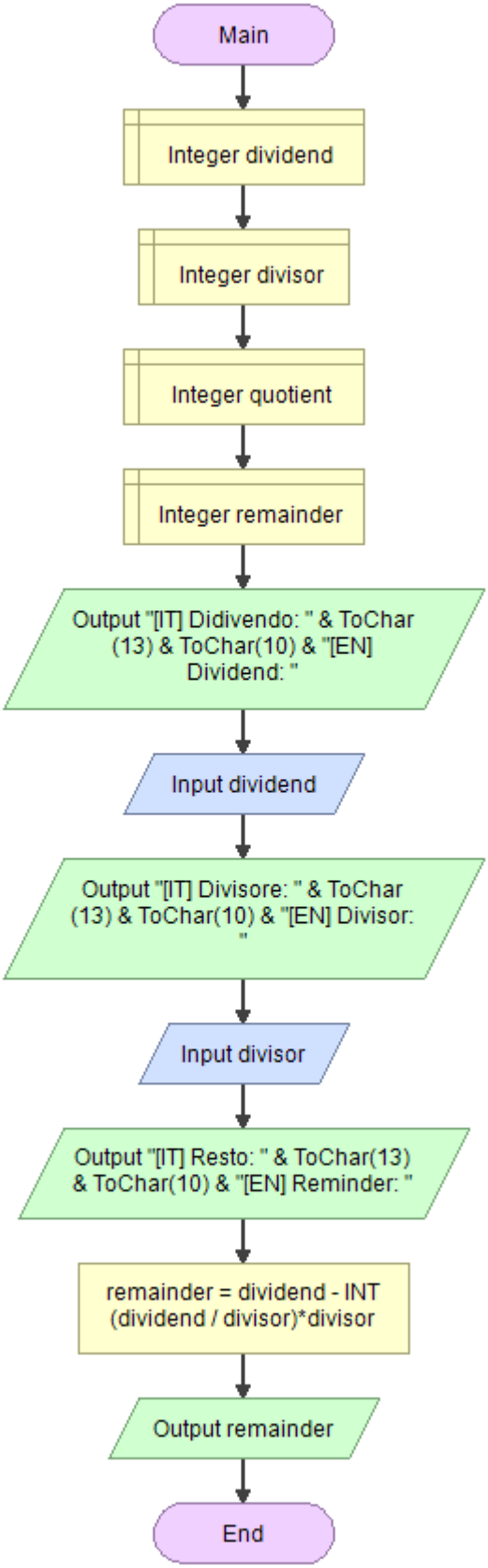
27. G ‖ Somma di una progressione geometrica infinita

Sum of infinite Geometric Progression (G.P.)



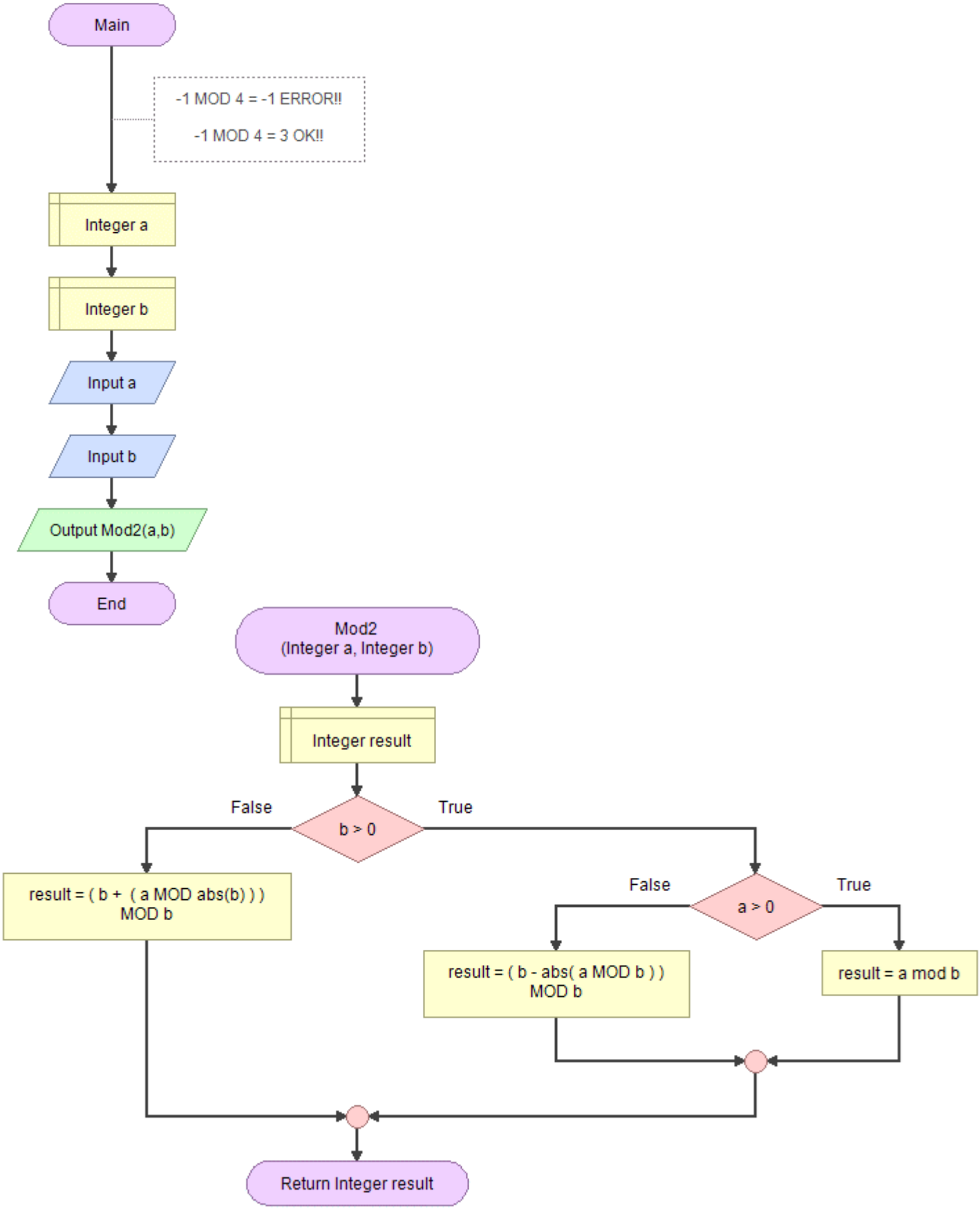
28. G ‖ Calcolo del resto nella divisione tra interi in N (naturali positivi) – MOD

Remainder calculus in division of integers (Modulation or MOD)



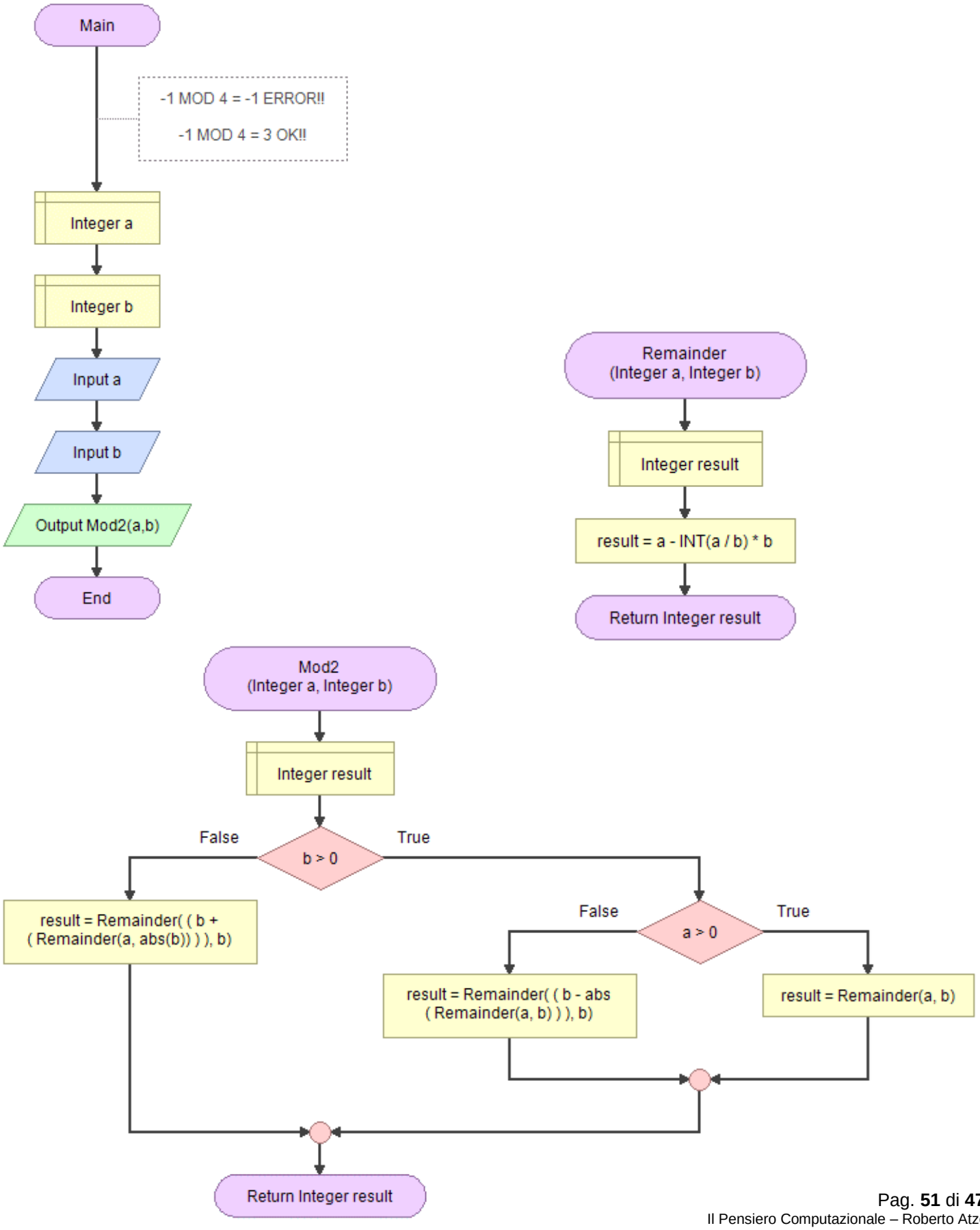
29. **G F** ‖ Calcolo del resto nella divisione tra interi in Z (numeri relativi) - funzione MOD

Remainder calculus in division of relative integers in Z (by MOD)



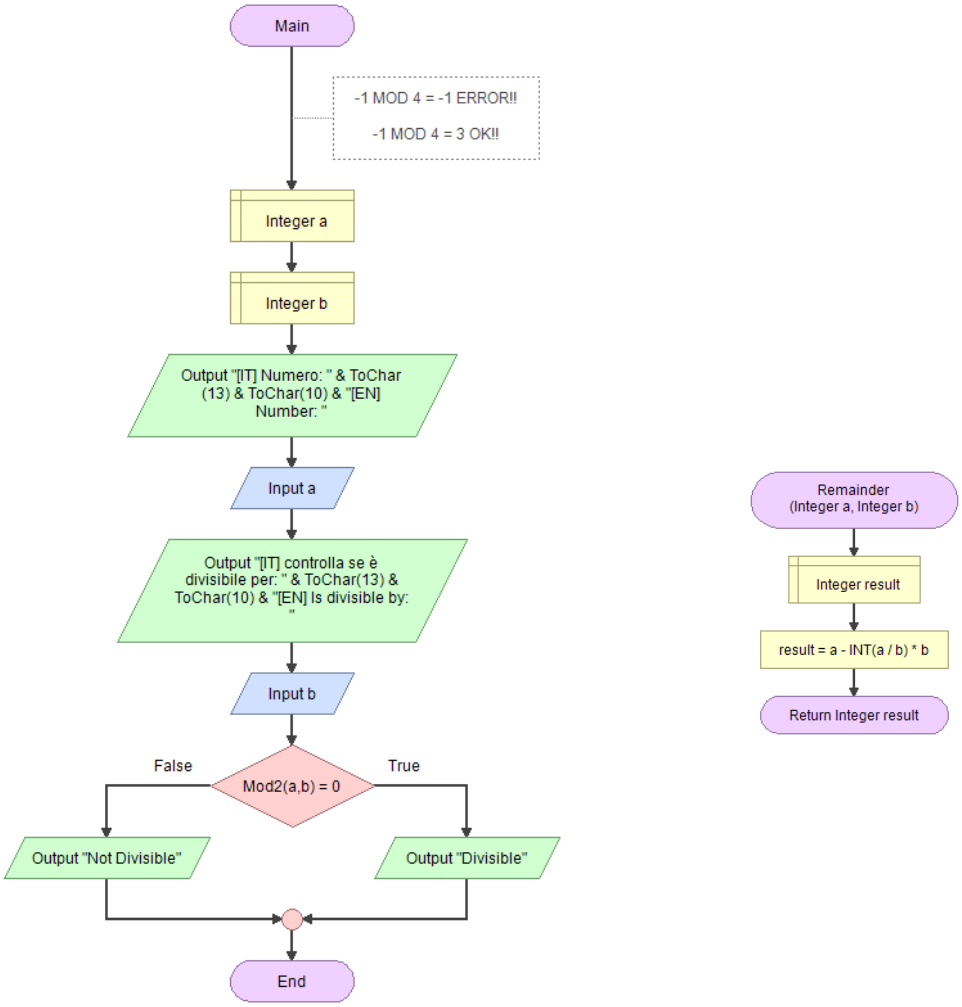
30. **G F** ‖ Calcolo del resto nella divisione tra interi in Z (numeri relativi) – senza funzione MOD

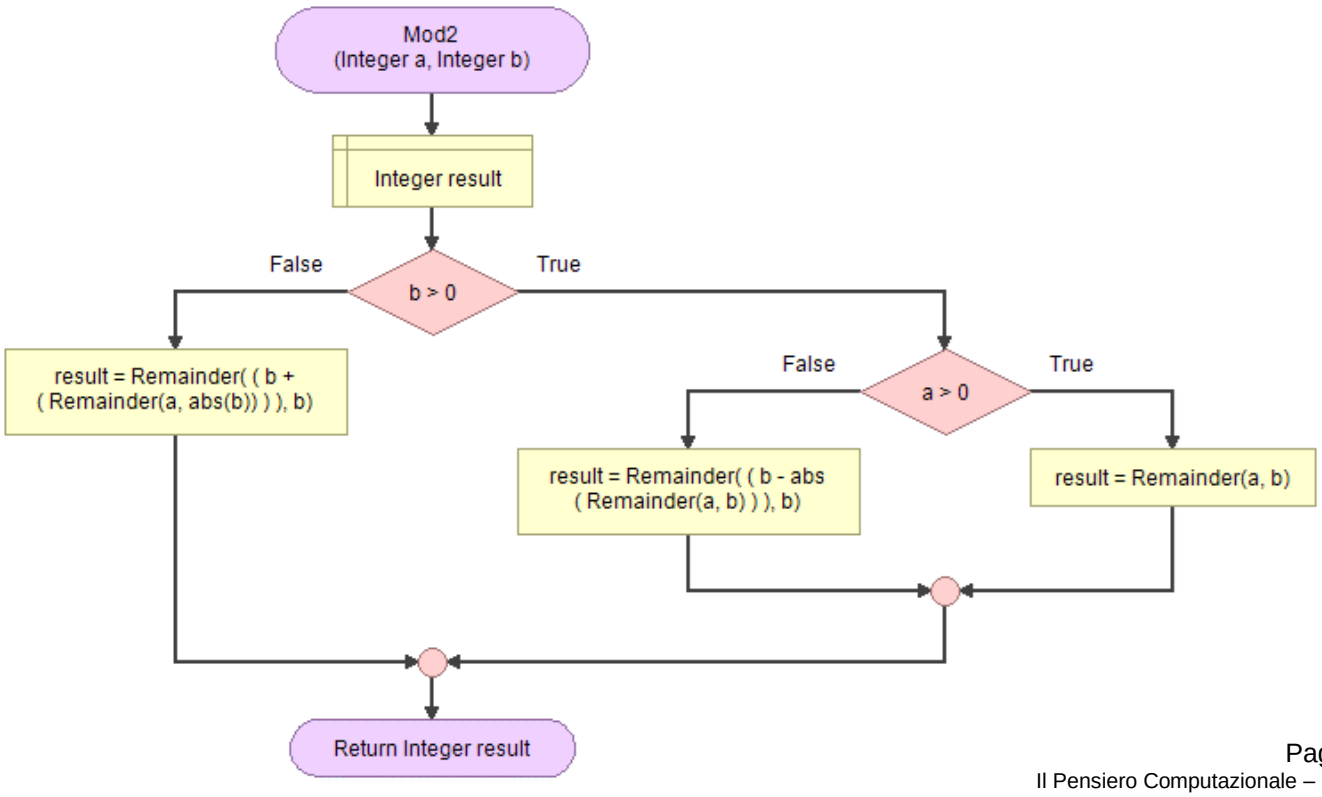
Remainder calculus in division of relative integers in Z (w/o MOD)



31. **G F** ‖ Controlla se un numero è divisibile per un altro – senza funzione MOD

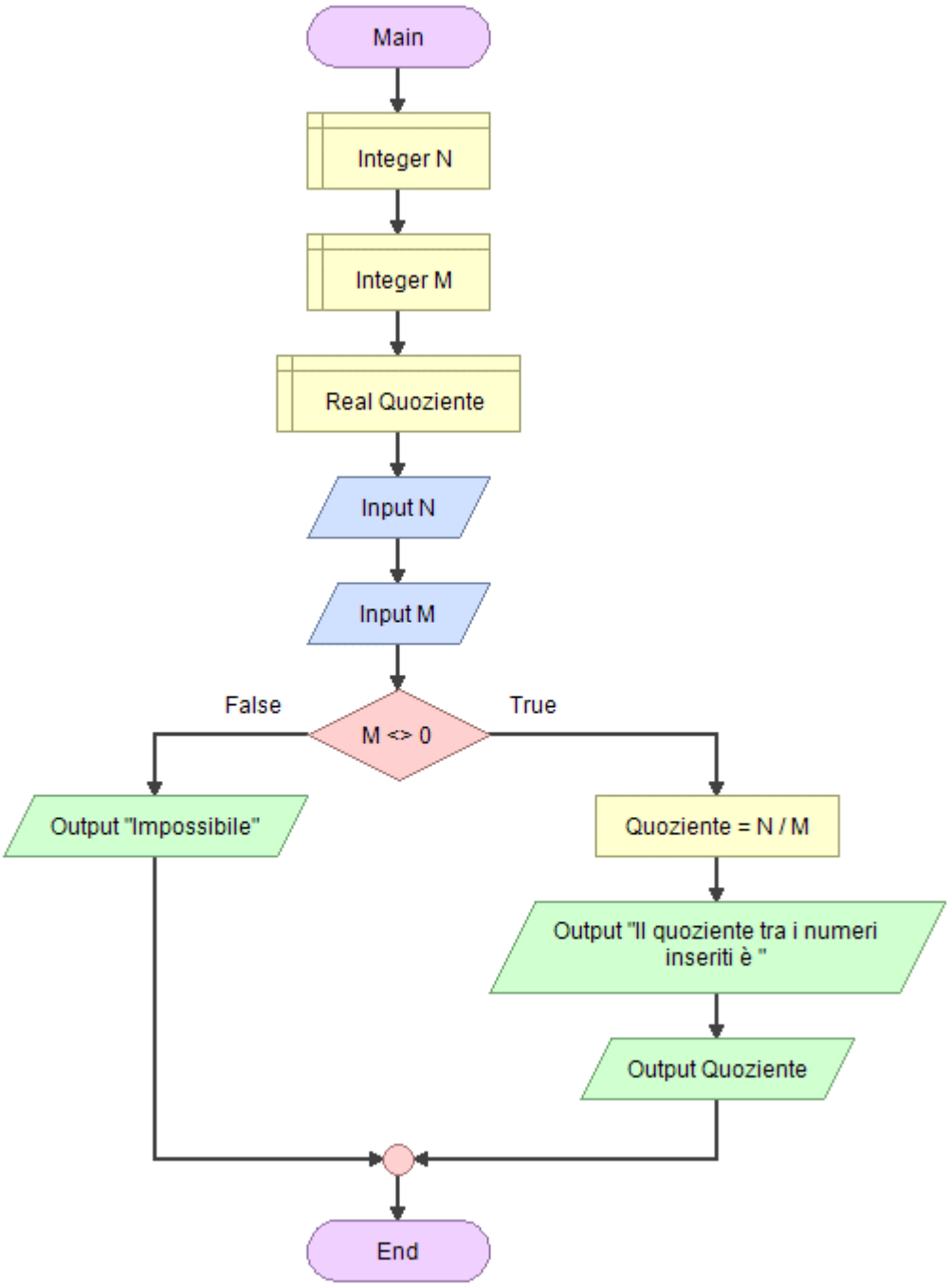
Check if a number is divisible by another - no MOD function





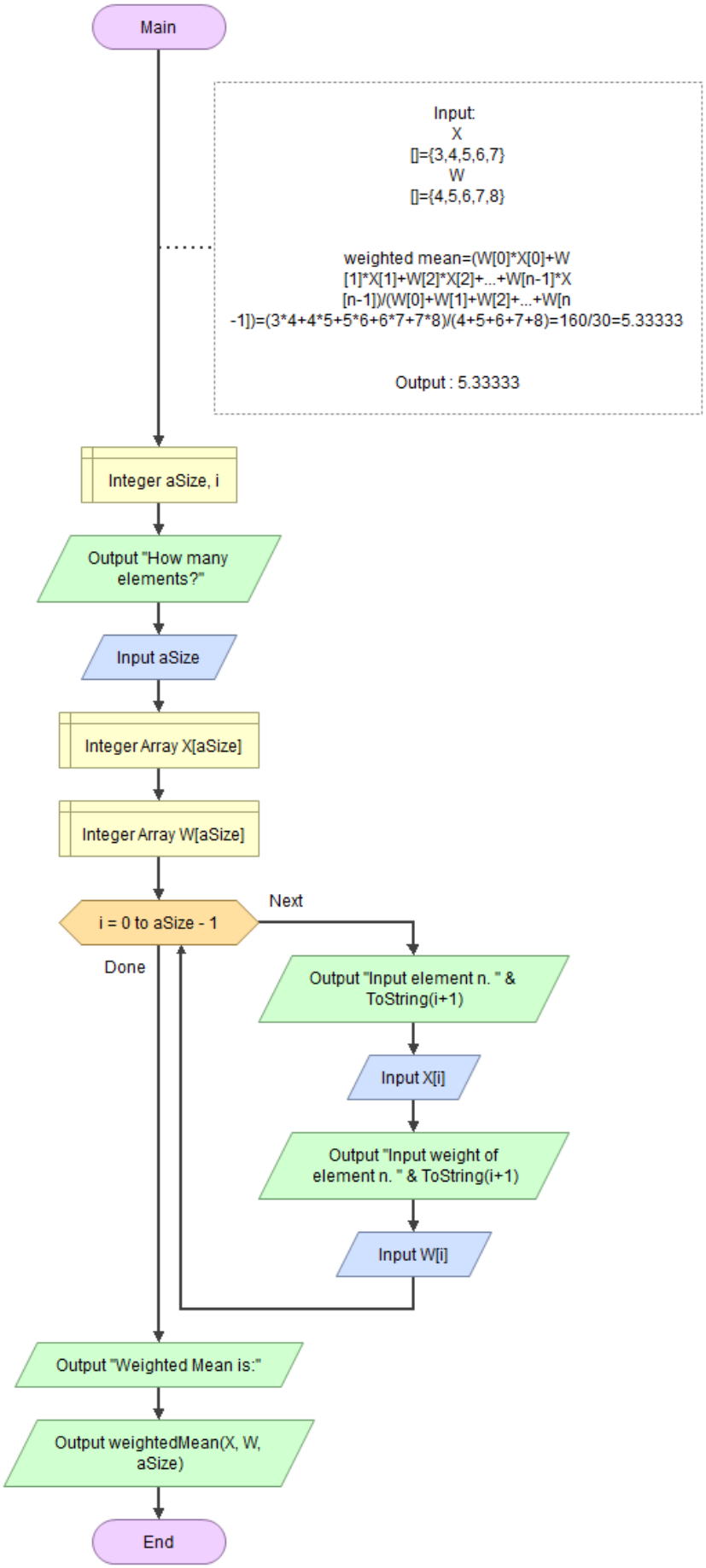
32. **G F** ‖ Dati due numeri calcolare il loro quoziente se il divisore è <> 0, ritornare “impossibile” se il divisore = 0

Compute the quotient of 2 numbers if the divisor is <> 0 or return "impossible" if divisor = 0

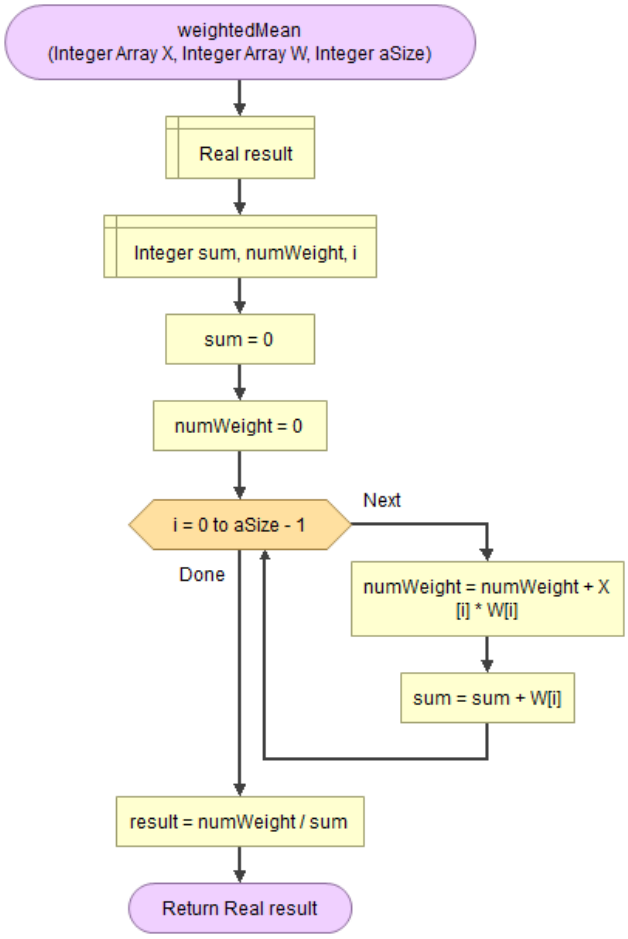


33. GAF ‖ Calcolo della Media Pesata

Weighted Mean of natural numbers



$$\bar{x} = \frac{\sum_{i=1}^{n} (x_i * w_i)}{\sum_{i=1}^{n} w_i}$$

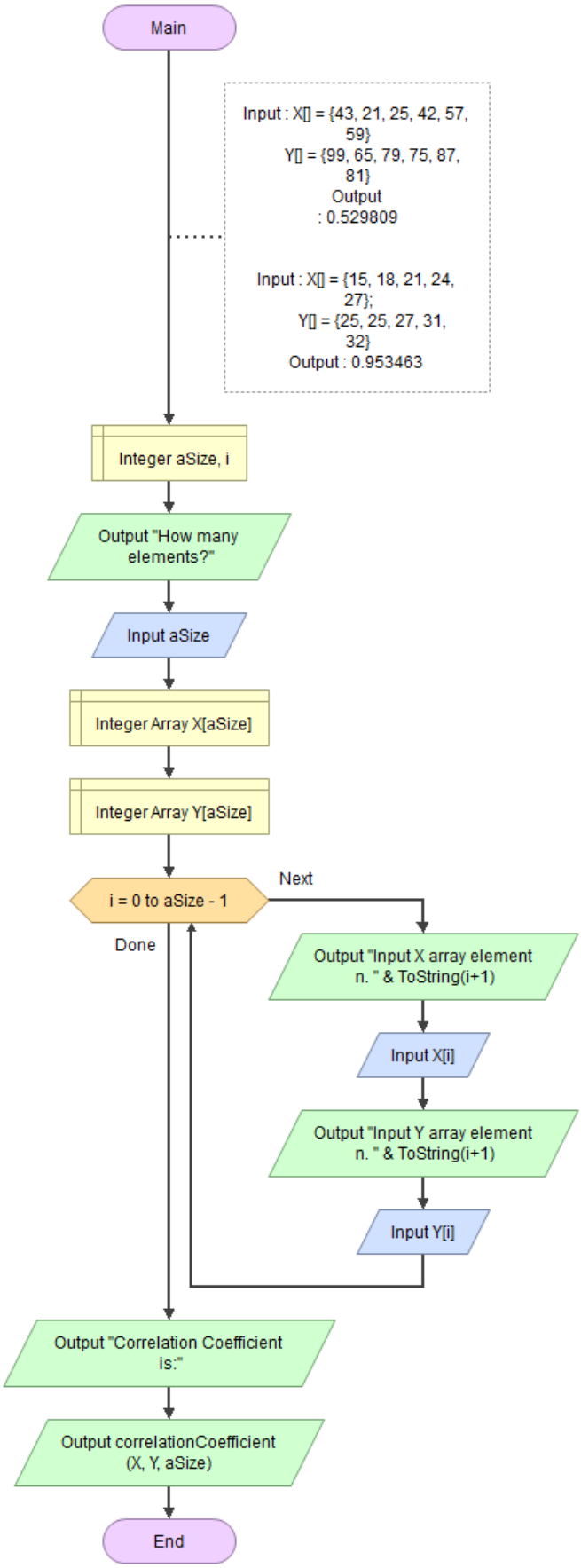


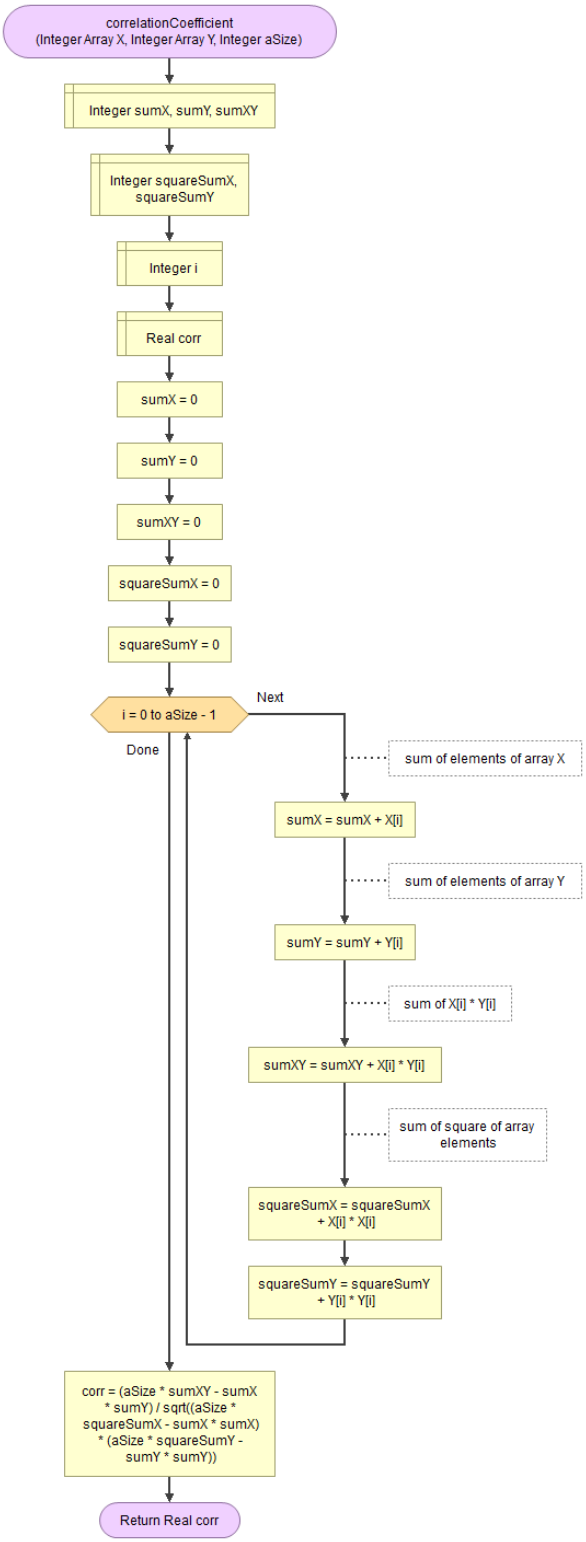
34. G F ‖ Calcolo del Coefficiente di Correlazione

Correlation Coefficient

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[\, n\sum x^2 - (\sum x)^2 \,]\,[\, n\sum y^2 - (\sum y)^2 \,]}}$$

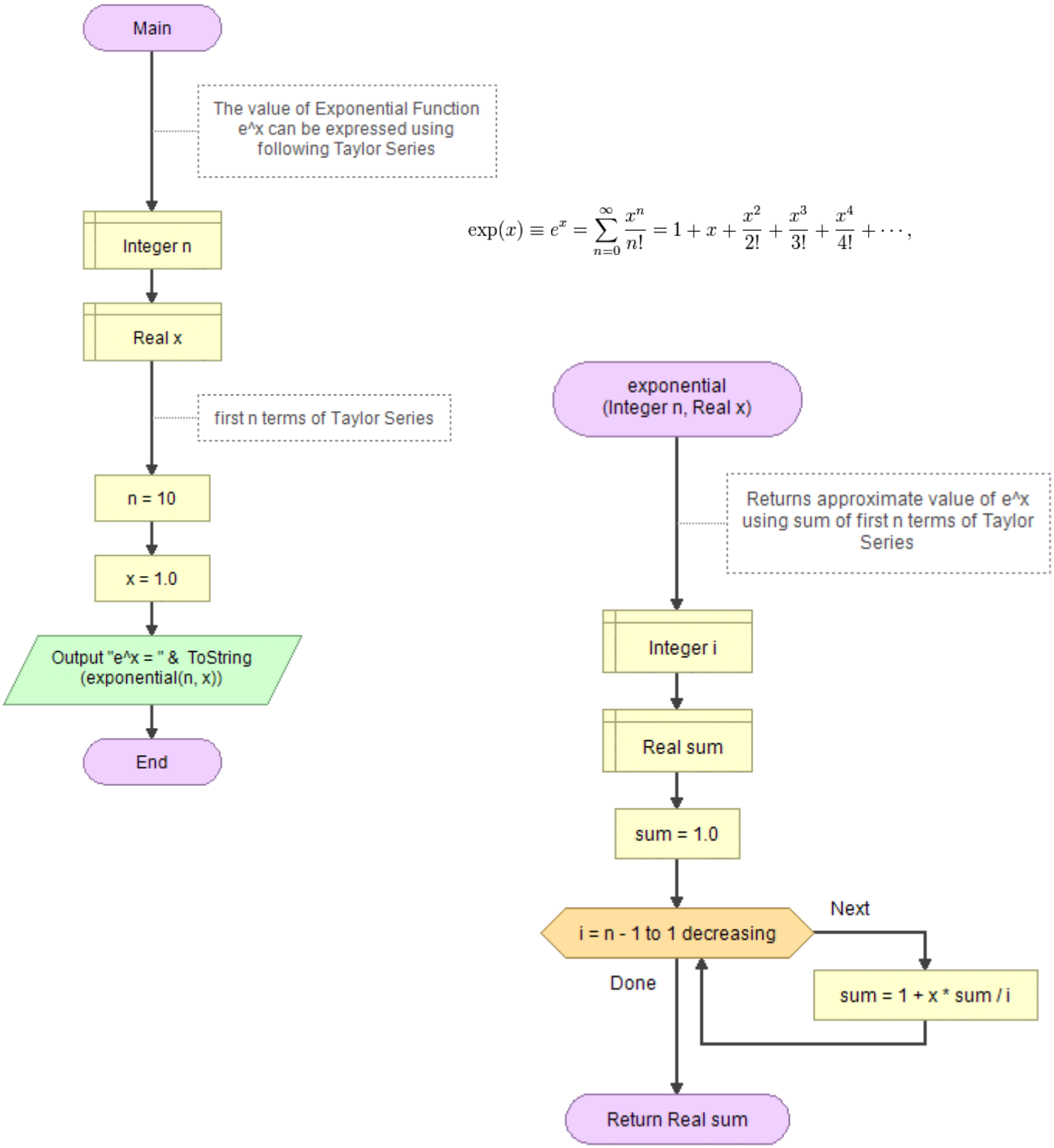
X	Y	X*Y	X*X	Y*Y
15	25	375	225	625
18	25	450	324	625
21	27	567	441	729
24	31	744	576	961
27	32	864	729	1024
∑X=105	∑Y=140	∑X*Y=3000	∑X*X=2295	∑Y*Y=3964





35. **G F** ‖ Calcolo di e^x (serie di Taylor)

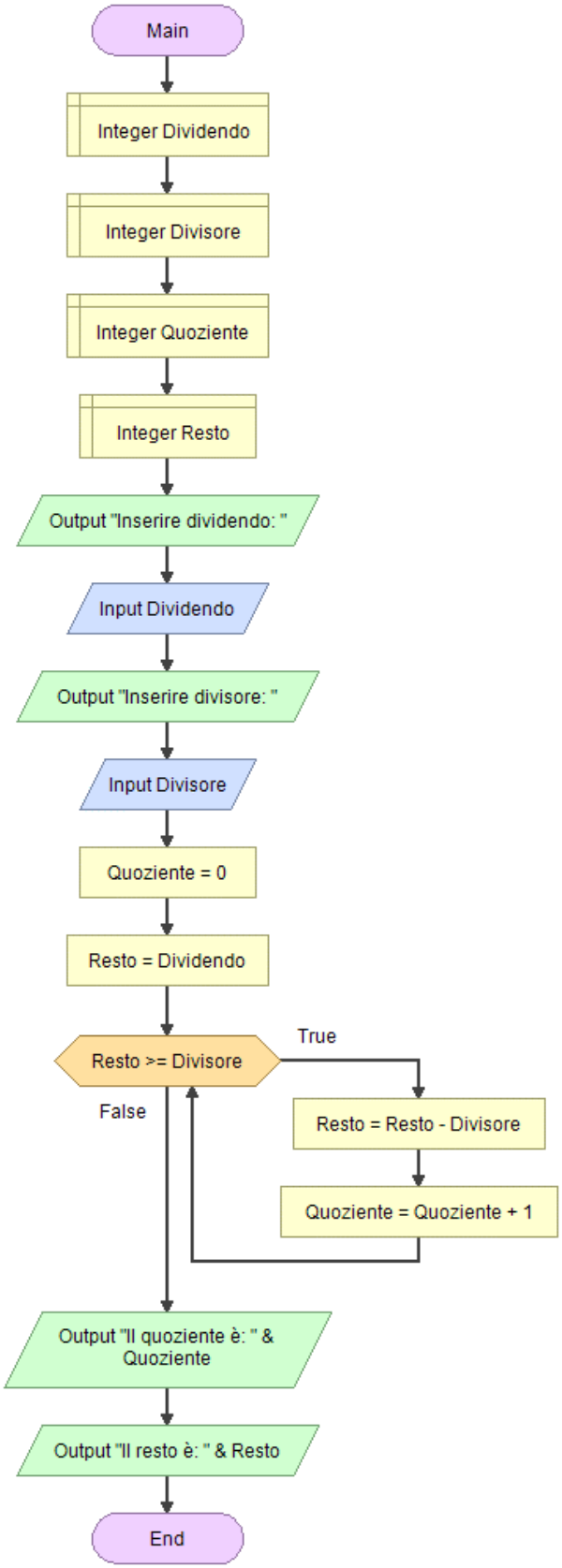
Calculate e^x (Taylor Series)



$$\exp(x) \equiv e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots,$$

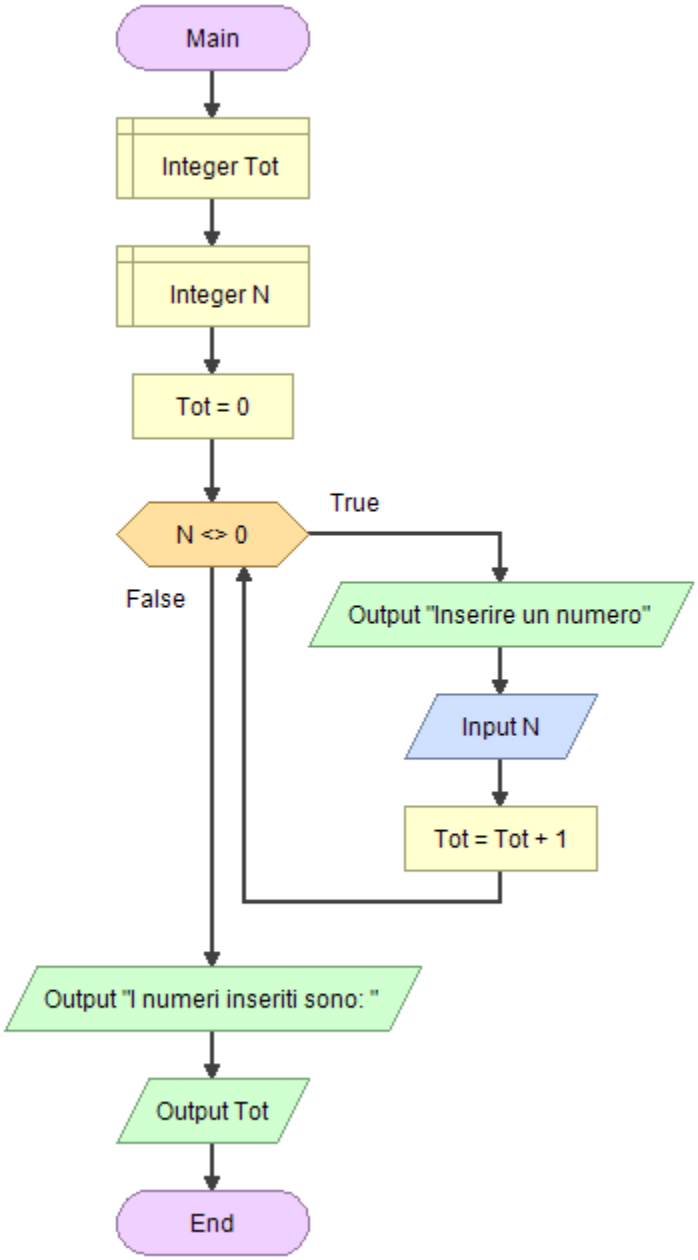
36. G ‖ Divisione di due numeri senza usare l'operatore di divisione "/"

Division of two numbers without using the division operator '/'



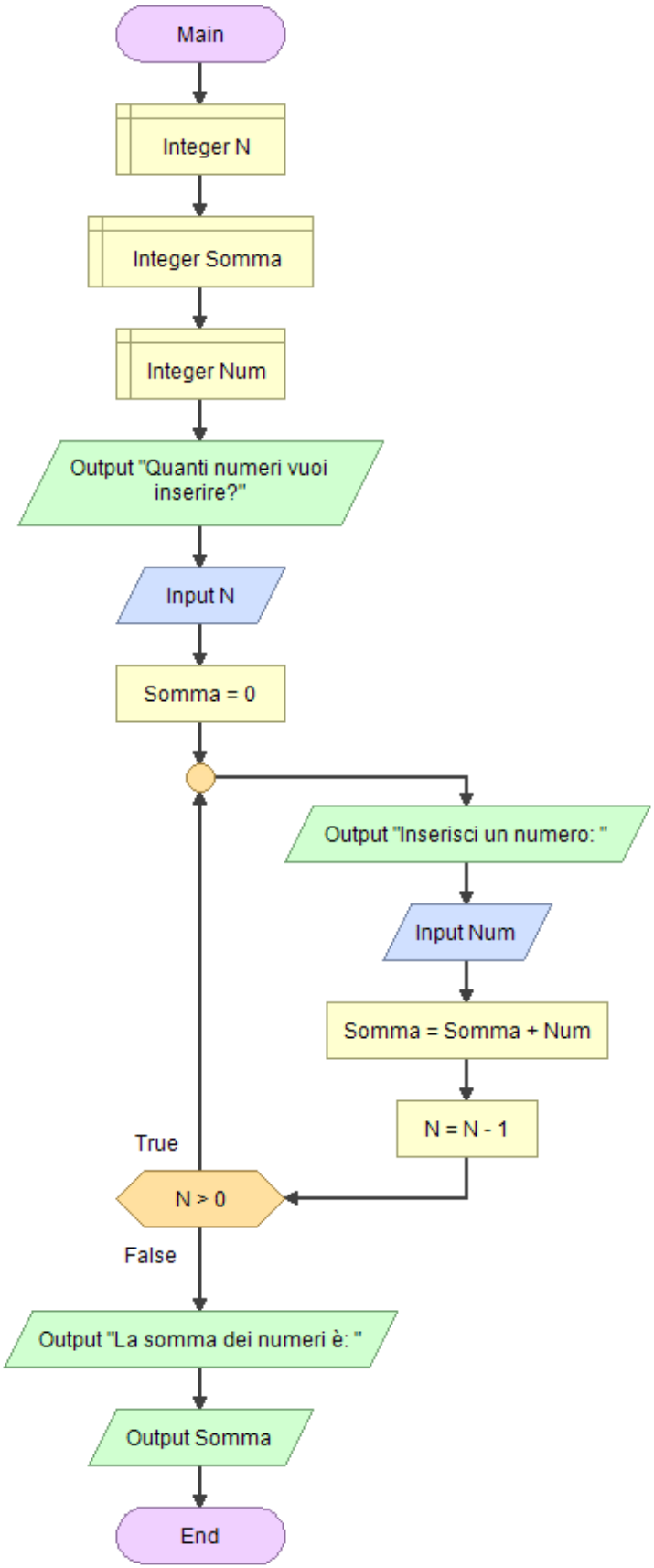
37. G | Inserire n numeri <> 0 (0 per finire), contare quanti sono i numeri inseriti

Counting inserting numbers



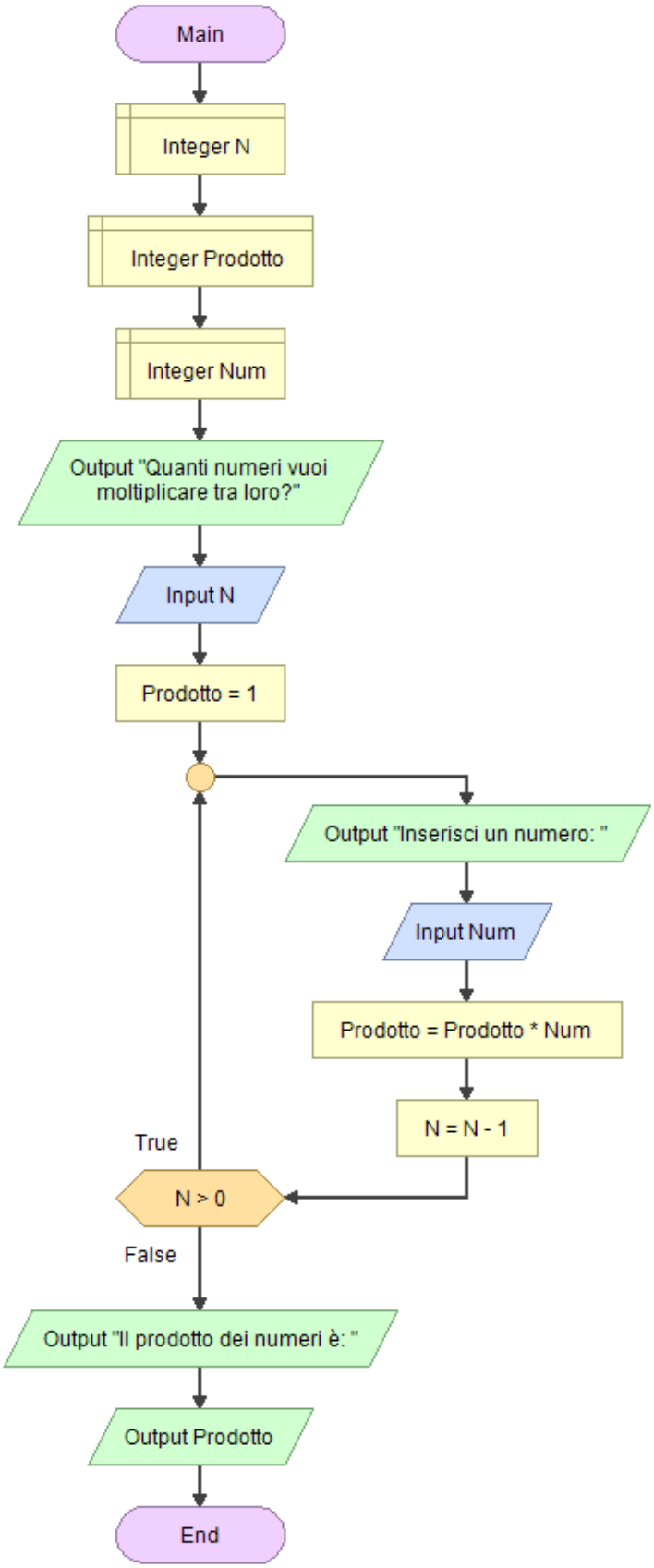
38. G ‖ Data una sequenza di n numeri interi, calcolare la somma ($S = a_1 + a_2 + \ldots + a_n$)

Given a sequence of n integers, calculate the sum ($S = a_1 + a_2 + \ldots + a_n$)



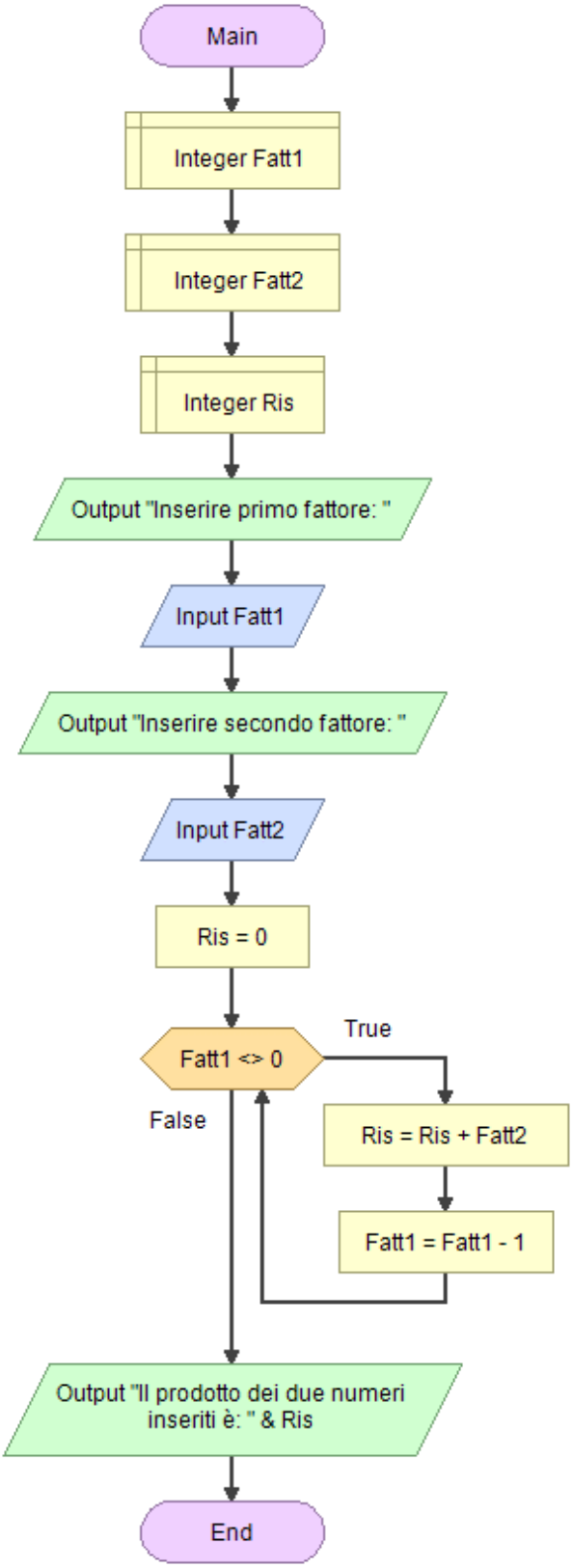
39. G ‖ Dati n numeri scrivere il loro prodotto

Given n numbers to write their product



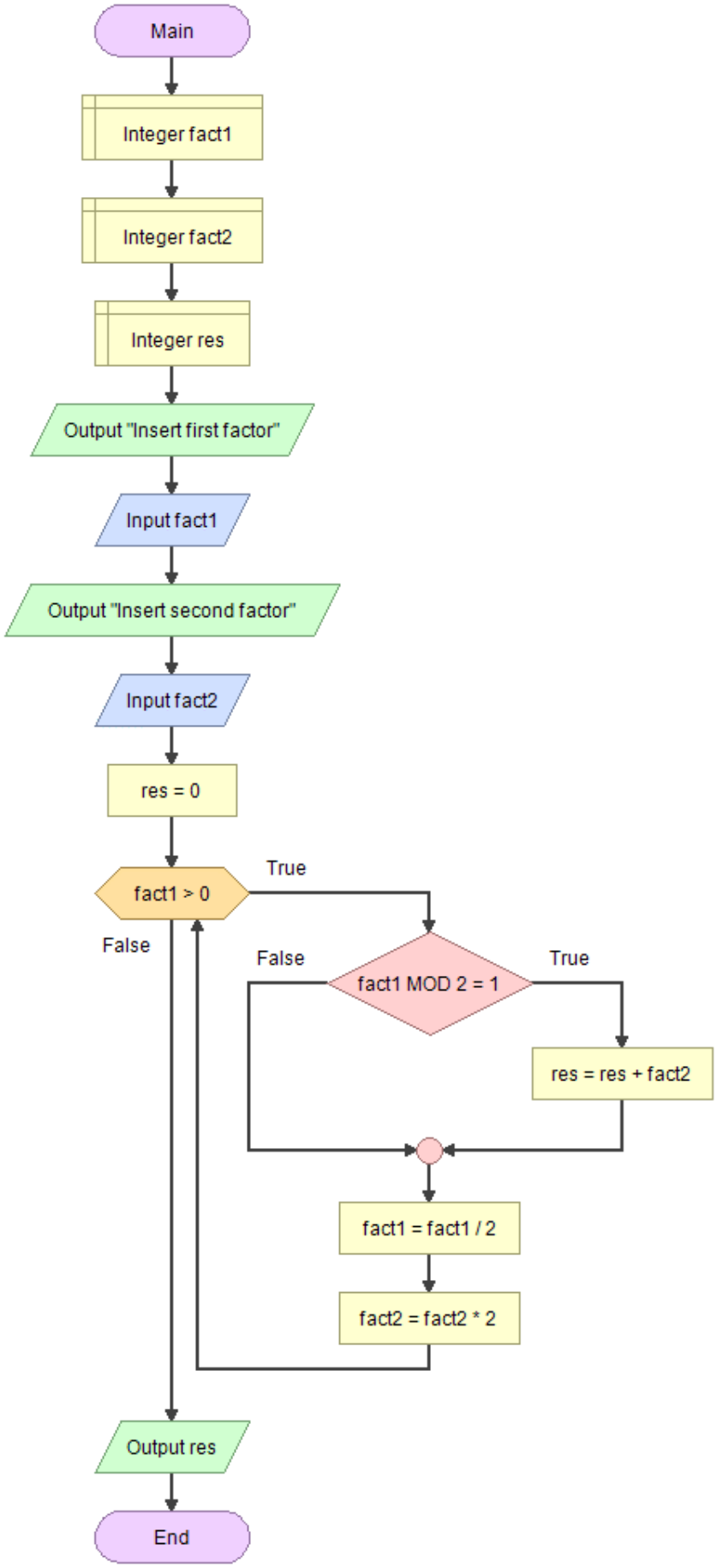
40. G ‖ Moltiplicazione di due numeri senza usare l'operatore di moltiplicazione "*"

Multiply two numbers without using the multiplication operator ''*



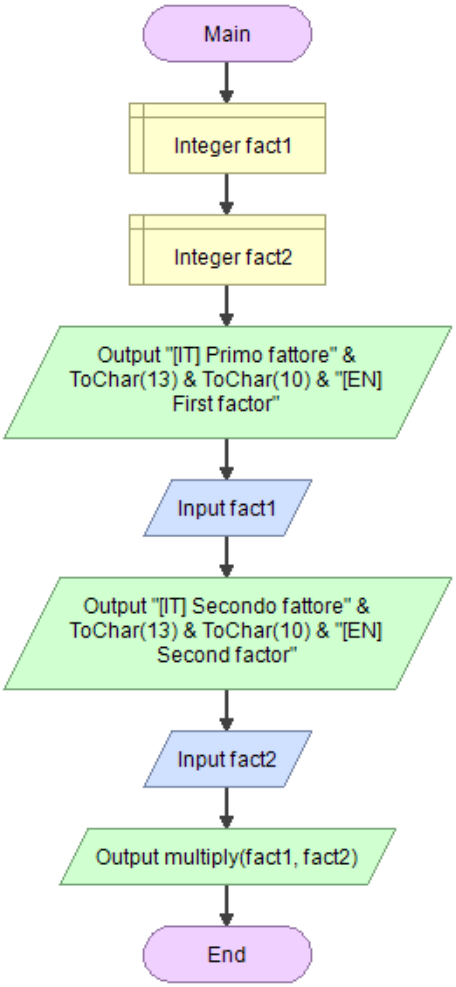
41. G ‖ Moltiplicazione di due numeri (algoritmo russo)

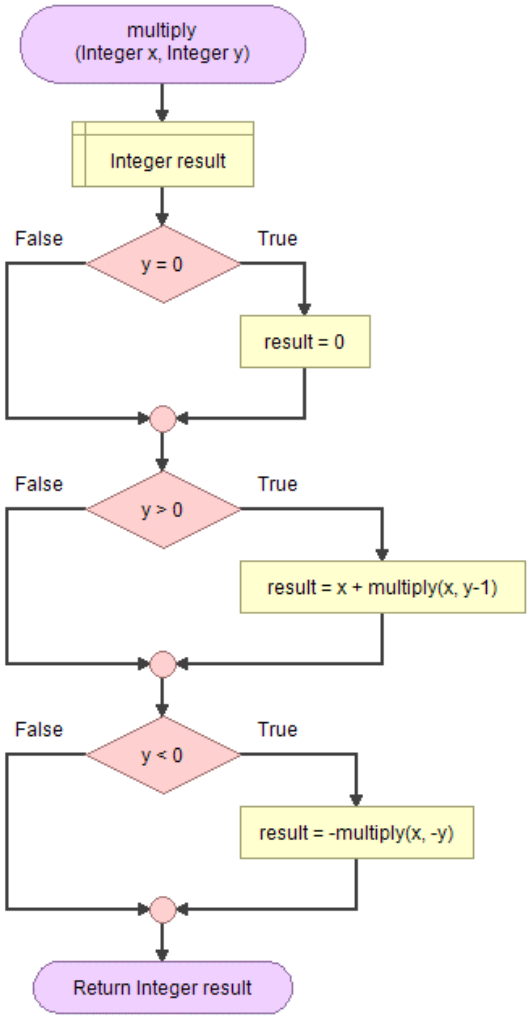
Multiply two numbers without using russian algorithm



42. **G F R** ‖ Moltiplicazione di due numeri senza usare l'operatore di moltiplicazione "*" (ricorsivo)

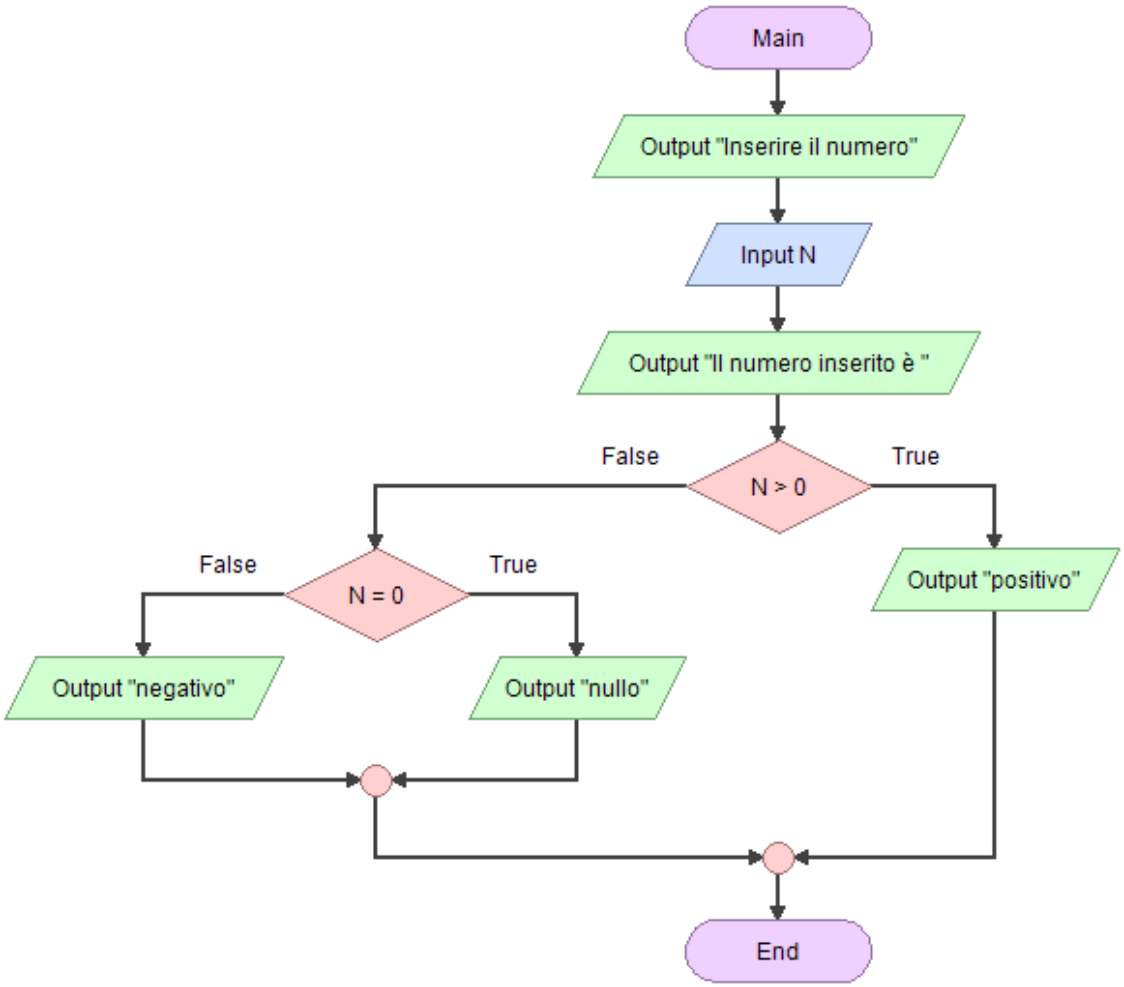
Multiply two numbers without using the multiplication operator '' (recursive)*





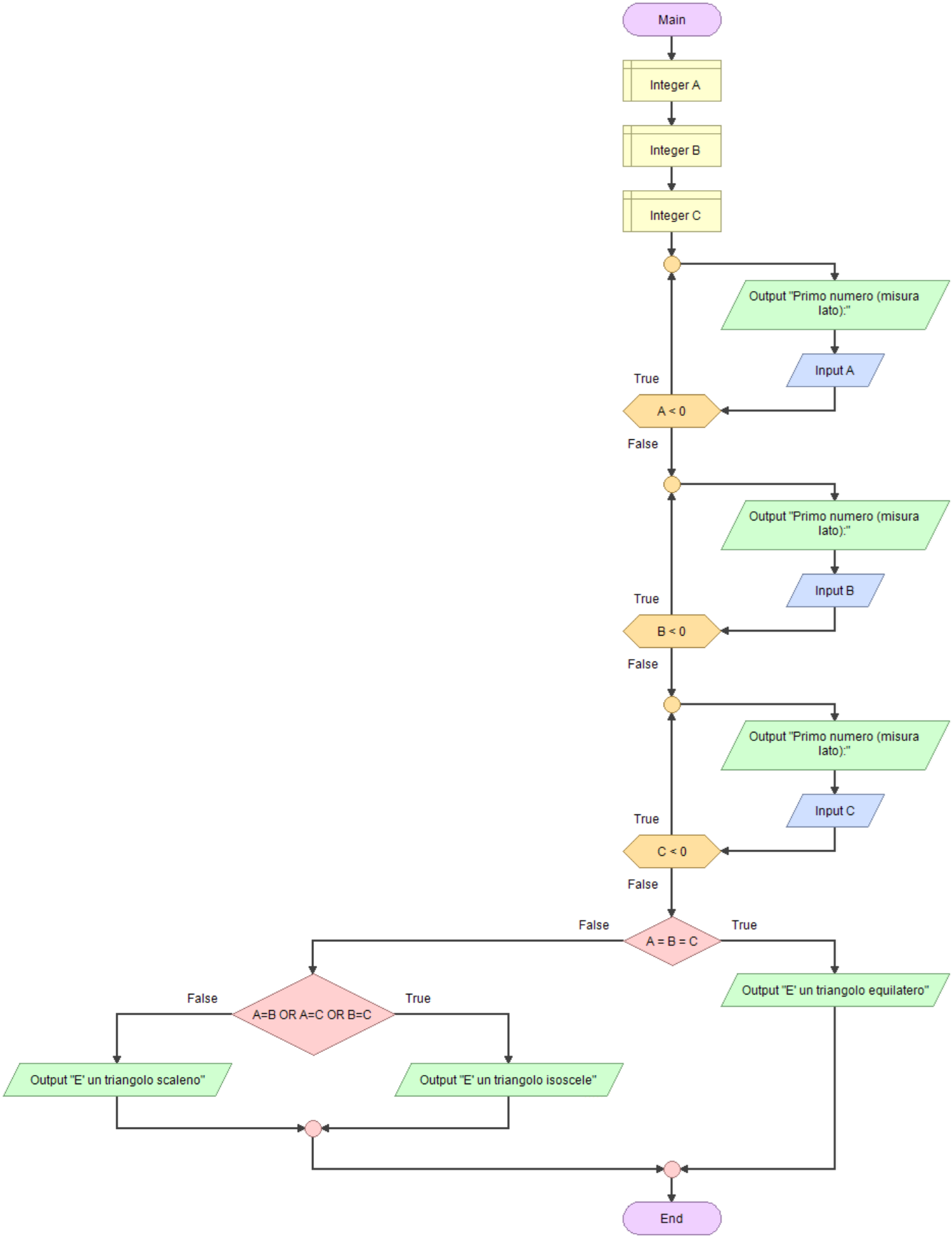
43. G ‖ Dato un numero determinare se è positivo, negativo o nullo

How the number I entered is?



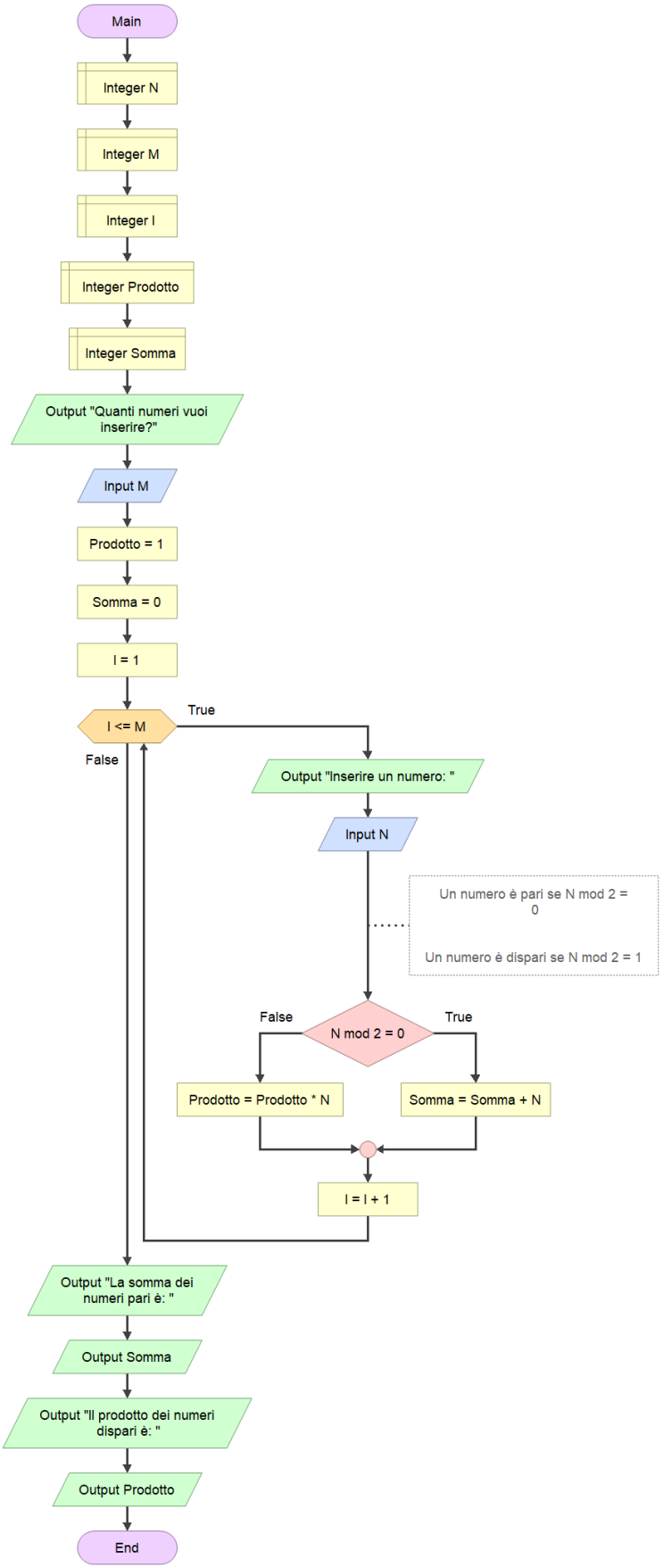
44. G ‖ **Dati tre numeri reali dire che tipo di triangolo essi formano** (classificazione dei triangoli in base ai lati).

Kind of triangle according to sides



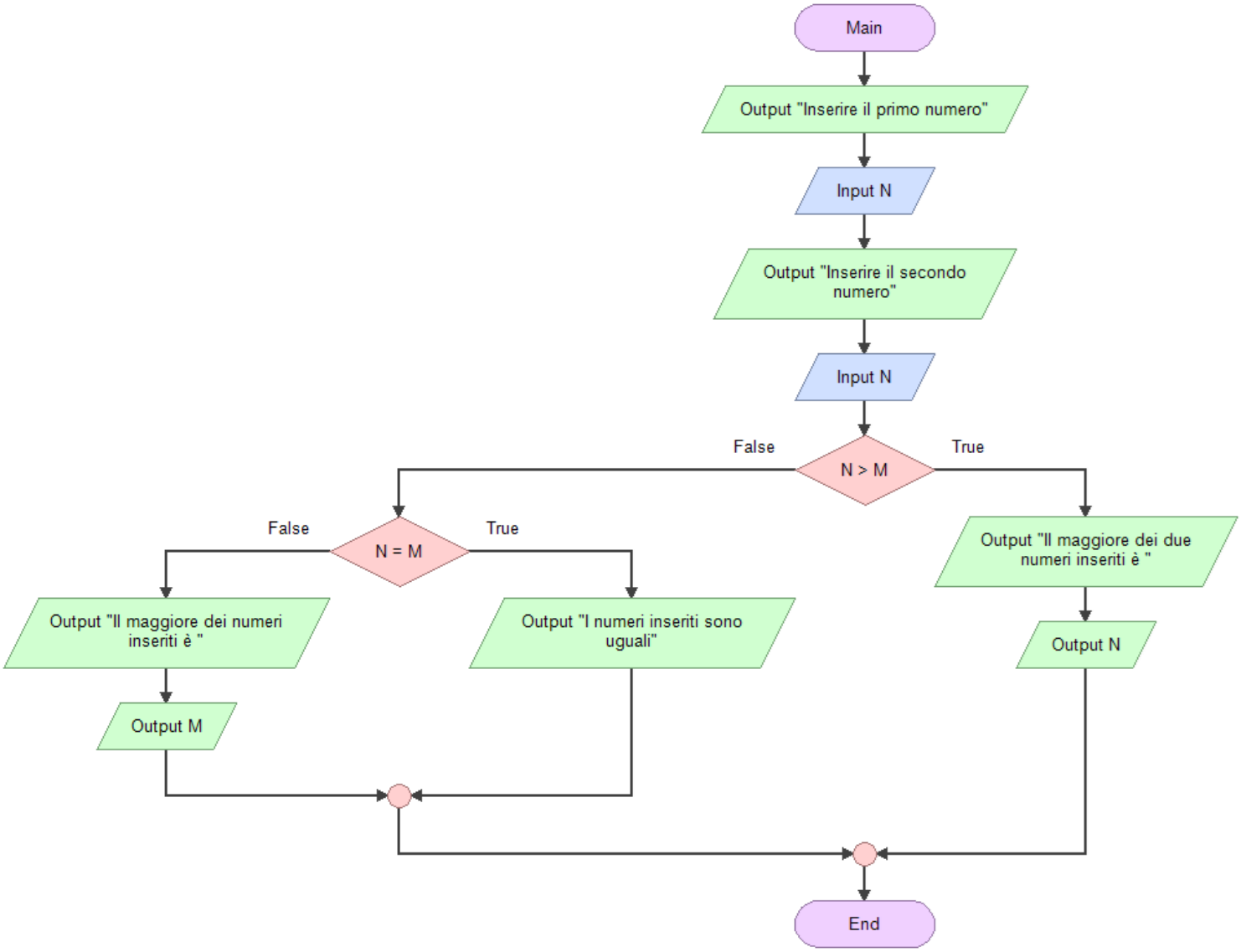
45. G | Data una sequenza di n numeri interi, calcolare la somma dei pari ed il prodotto dei dispari

Given N numbers calculate the sum of the even numbers and the product of odd numbers



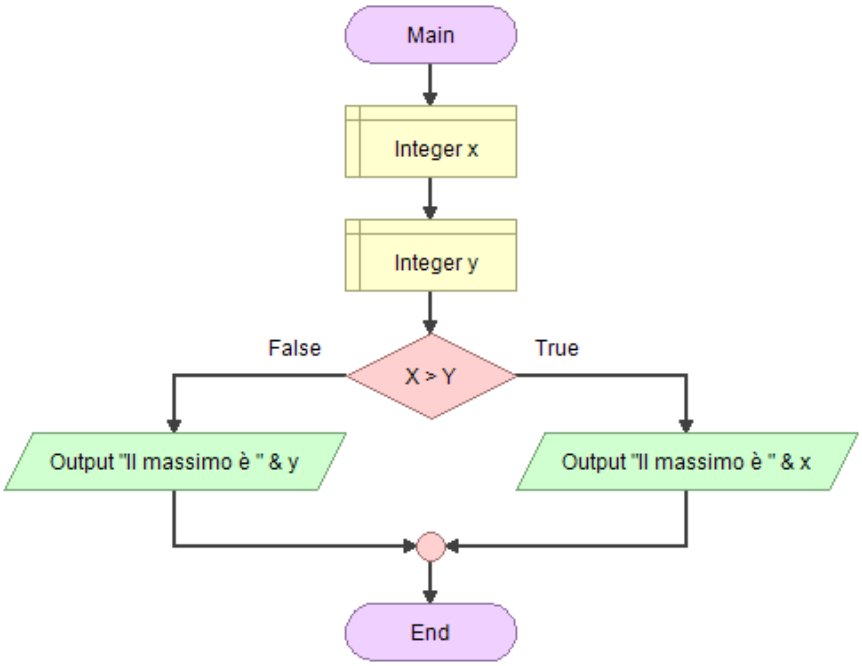
46. G ‖ Dati due numeri, determinare il maggiore (verificare anche se sono uguali)

Given two numbers, determine the major one (and check if they are equal)



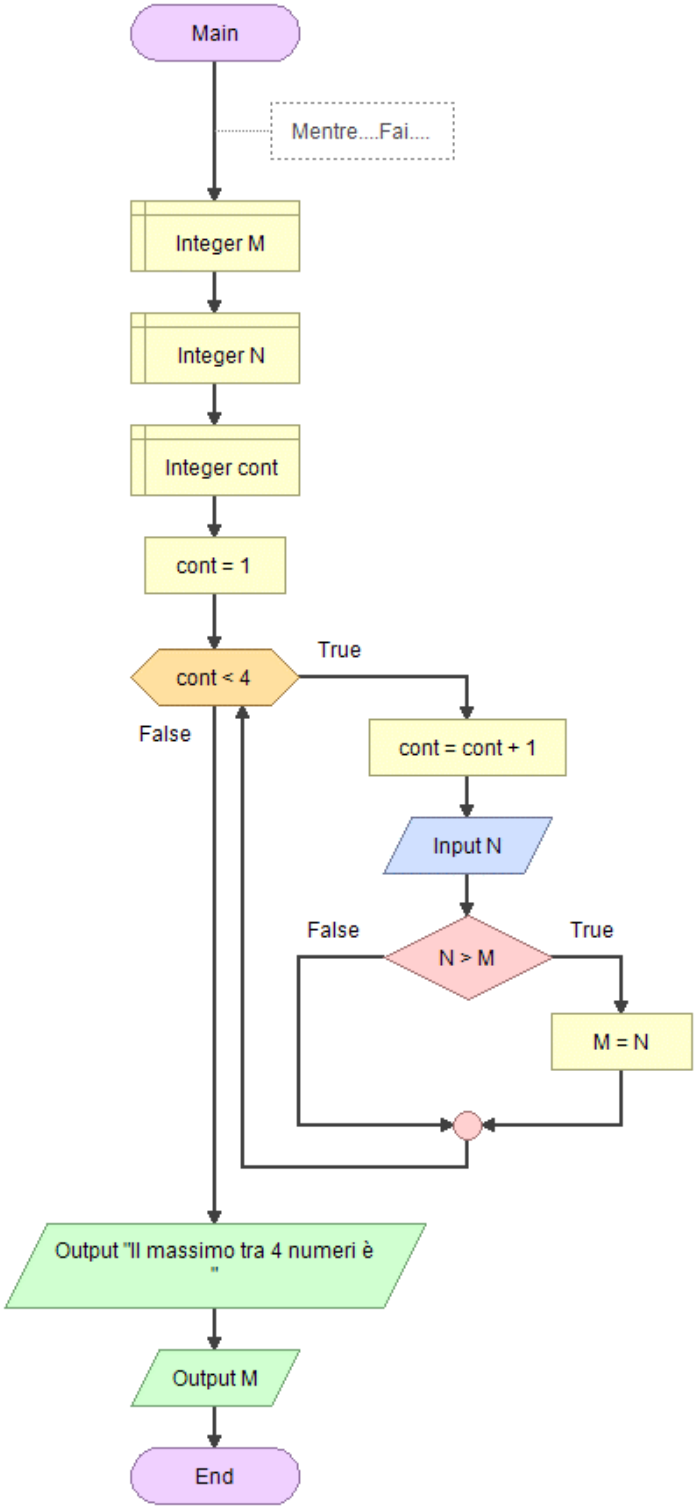
47. G ‖ Calcolare il massimo tra 2 numeri

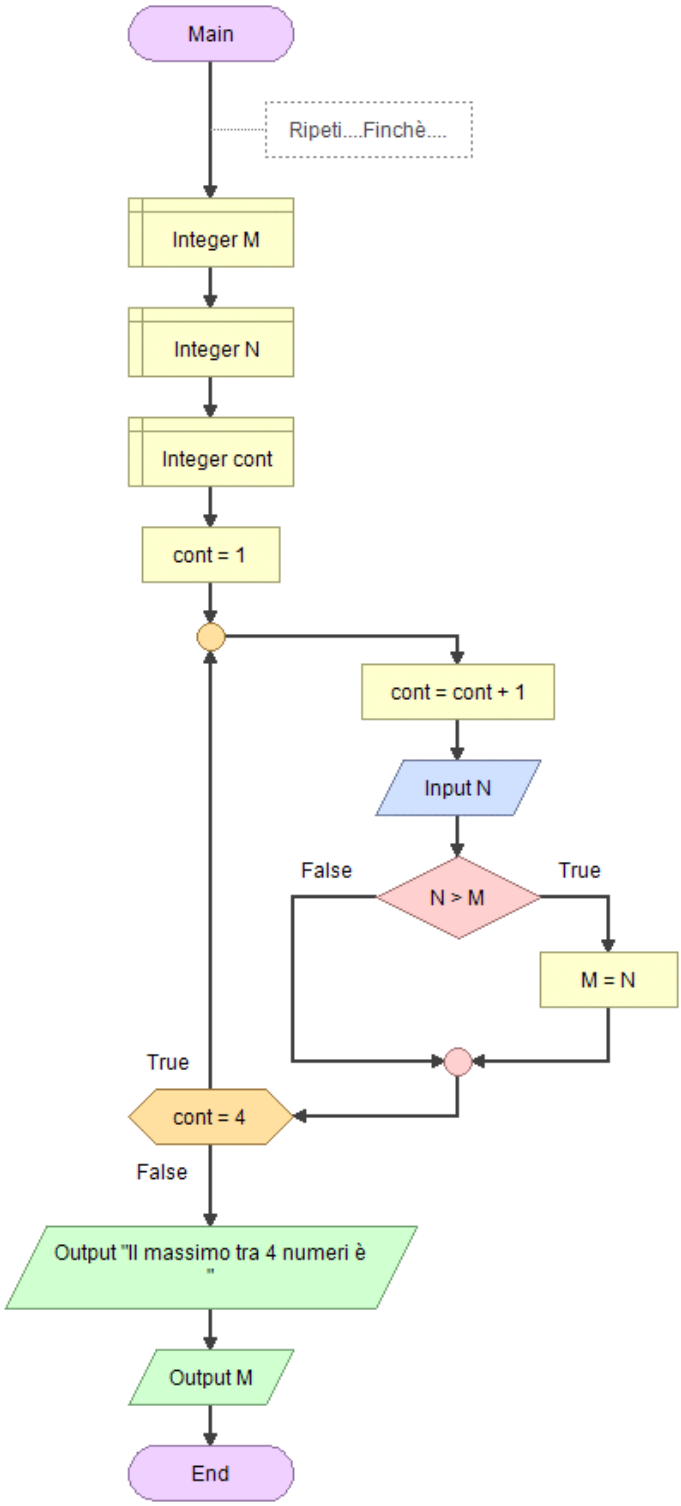
Calculate MAX of two numbers



48. G ‖ Calcolare il massimo tra 4 numeri con due cicli diversi

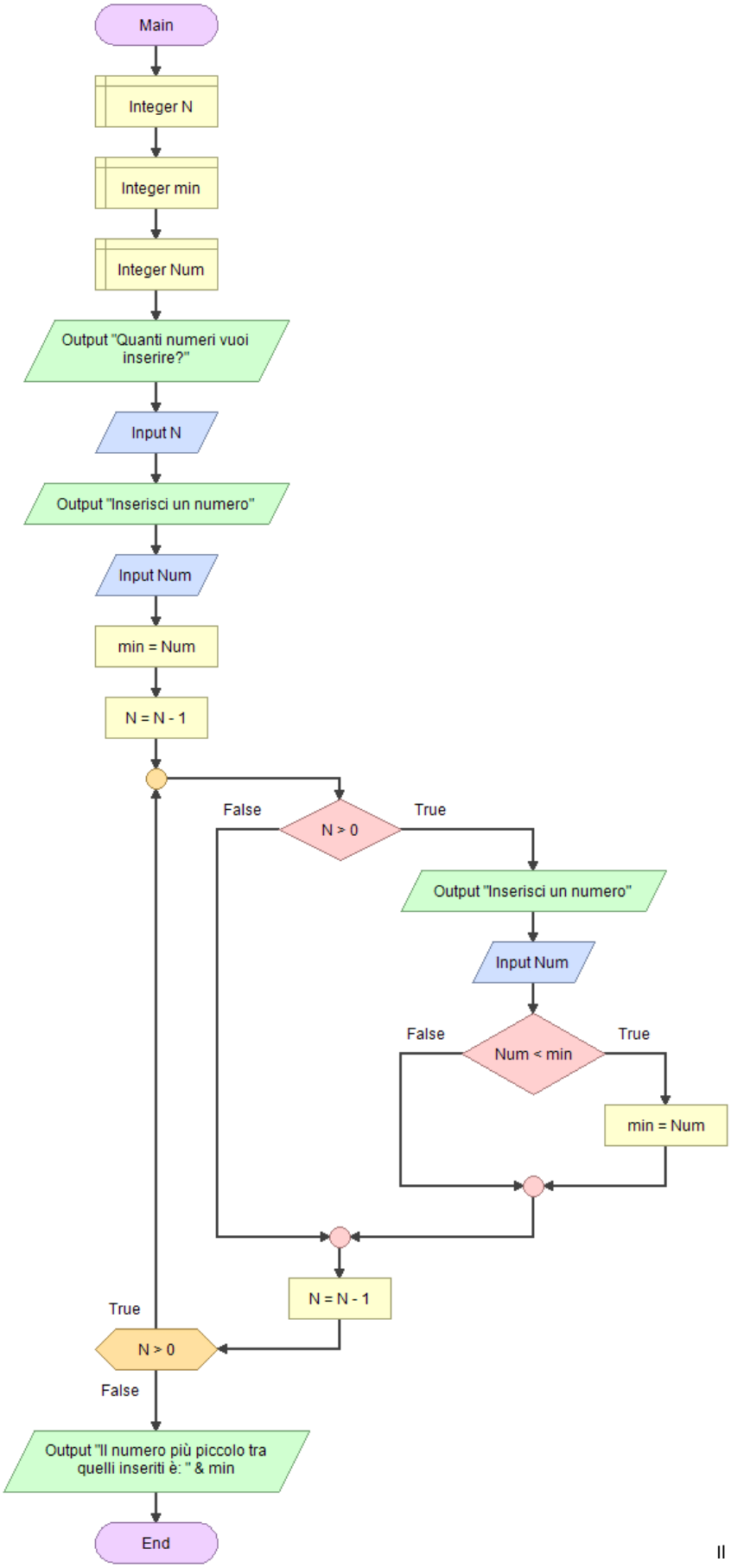
Calculate MAX of four numbers (with head and tail loop control)





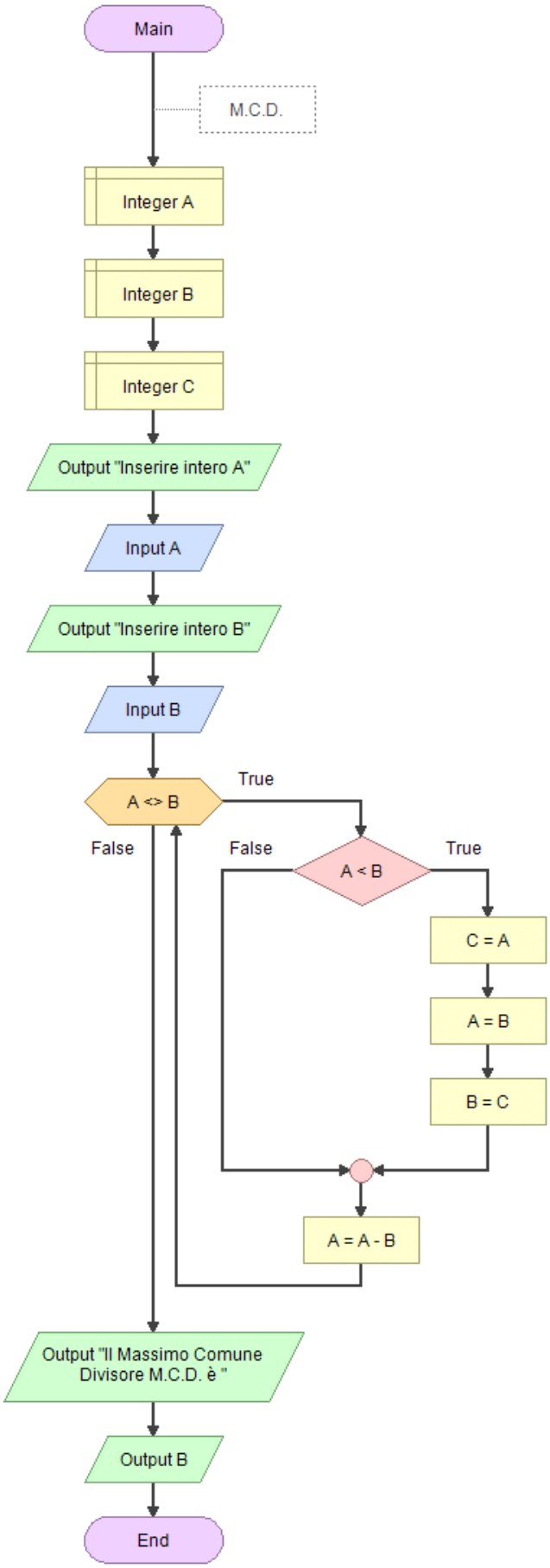
49. G ‖ Data una sequenza di numeri visualizzare il numero minimo

MIN number in a list of entered numbers



50. G ‖ Massimo Comune Divisore (M.C.D.) tra due numeri

Greatest Common Divisor (GCD) of two integers



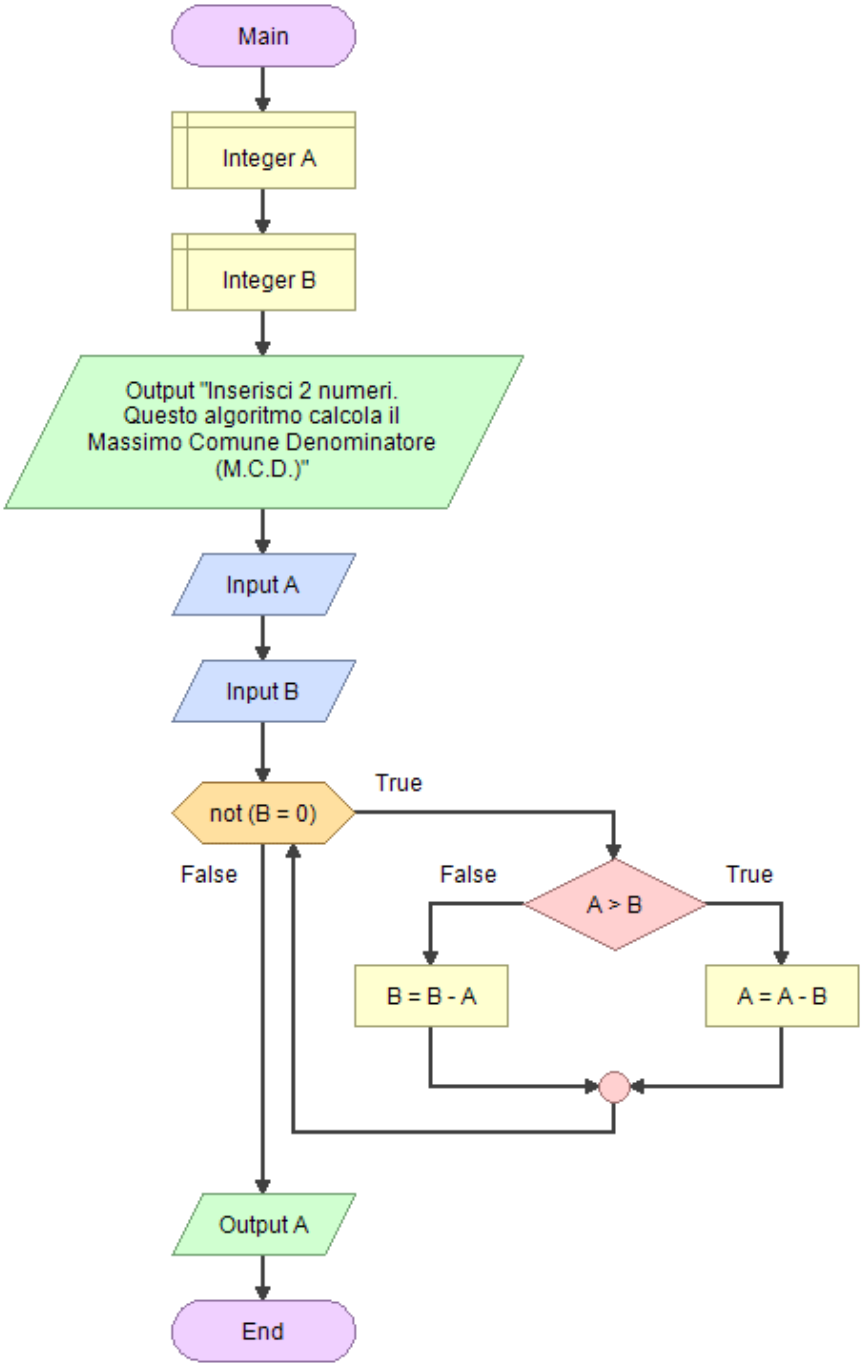
In matematica, il **massimo comune divisore** di due numeri interi a e b che non siano entrambi uguali a zero, si indica con $\mathrm{MCD}(a,b)$ ed è il numero naturale più grande per il quale possono entrambi essere divisi.
Se entrambi i numeri a e b sono uguali a 0, allora si pone $\mathrm{MCD}(a,b) = 0$

Ad esempio,

$$\mathrm{MCD}(12,18) = 6$$
$$\mathrm{MCD}(4,14) = 2$$
$$\mathrm{MCD}(5,0) = 5$$

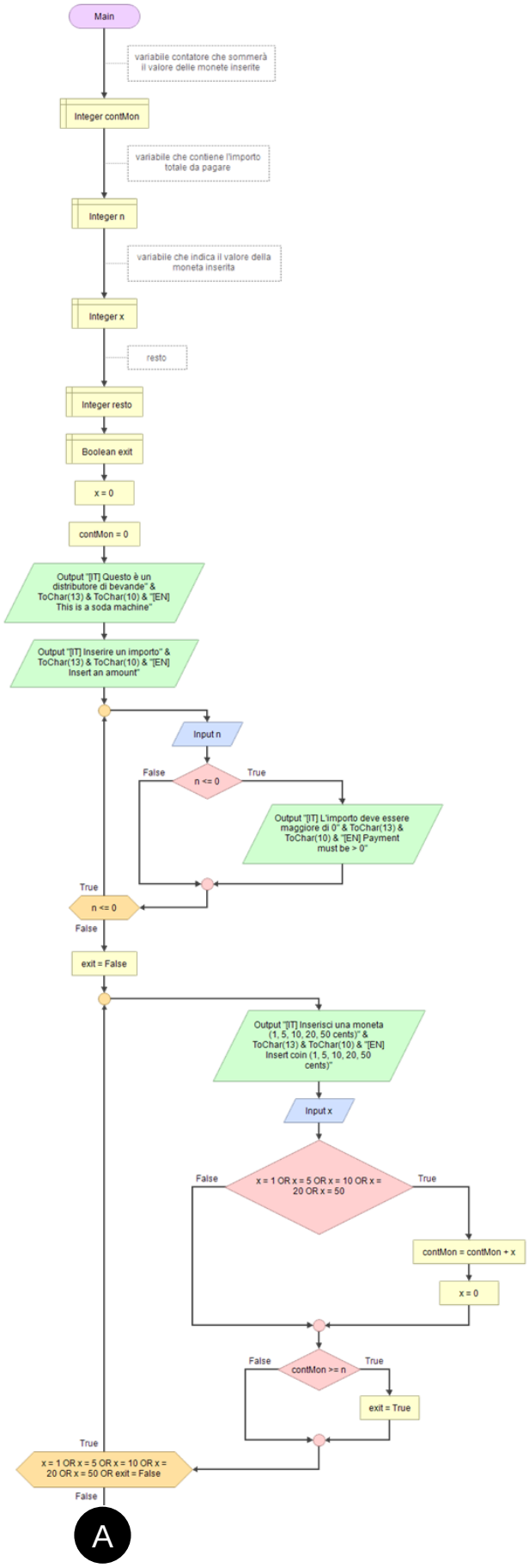
51. G ‖ Massimo Comune Divisore (M.C.D.) tra due numeri (algoritmo di Euclide)

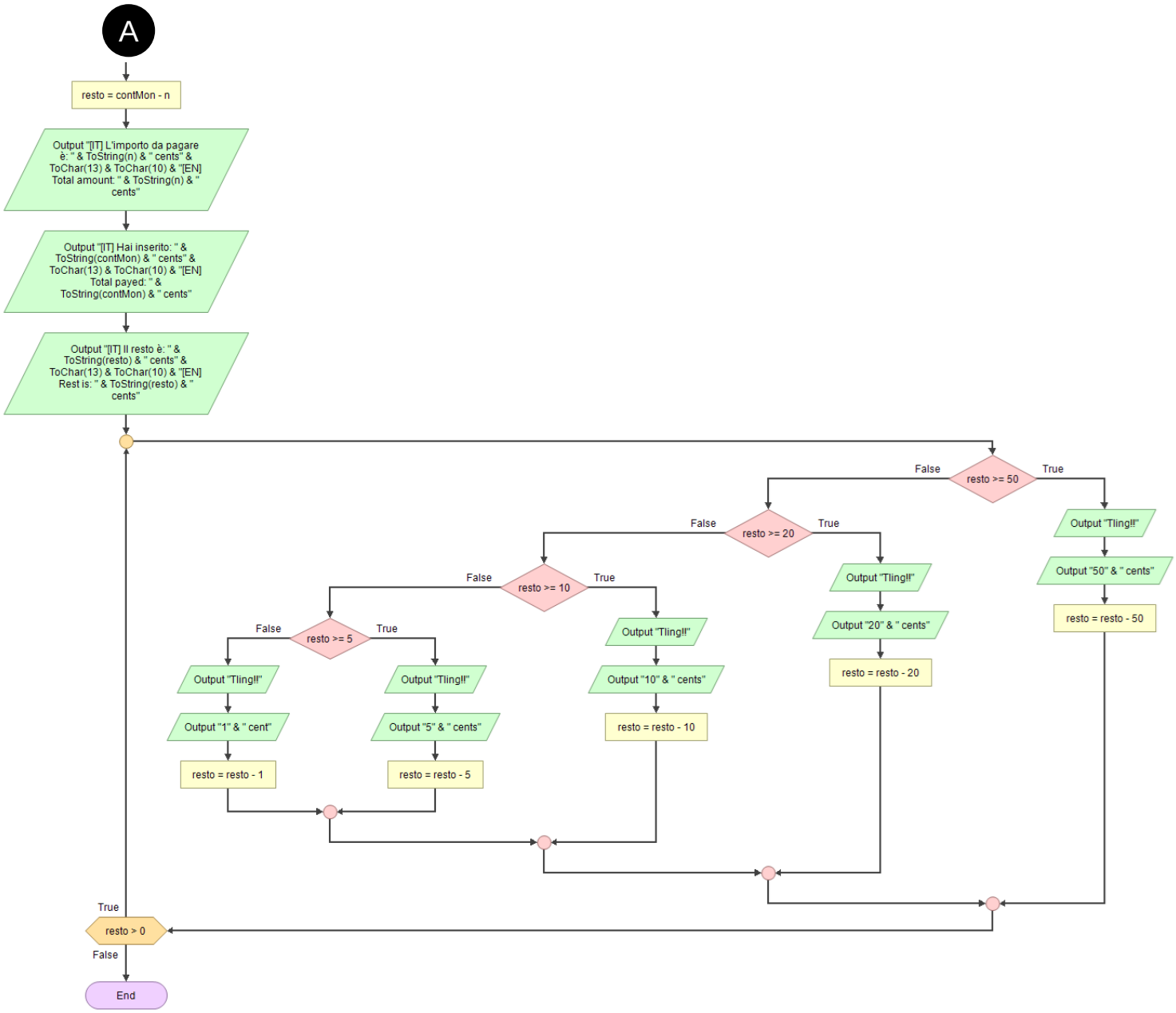
Greatest Common Divisor (GCD) of two integers (Euclide algorithm)



52. G ‖ Distributore di bevande

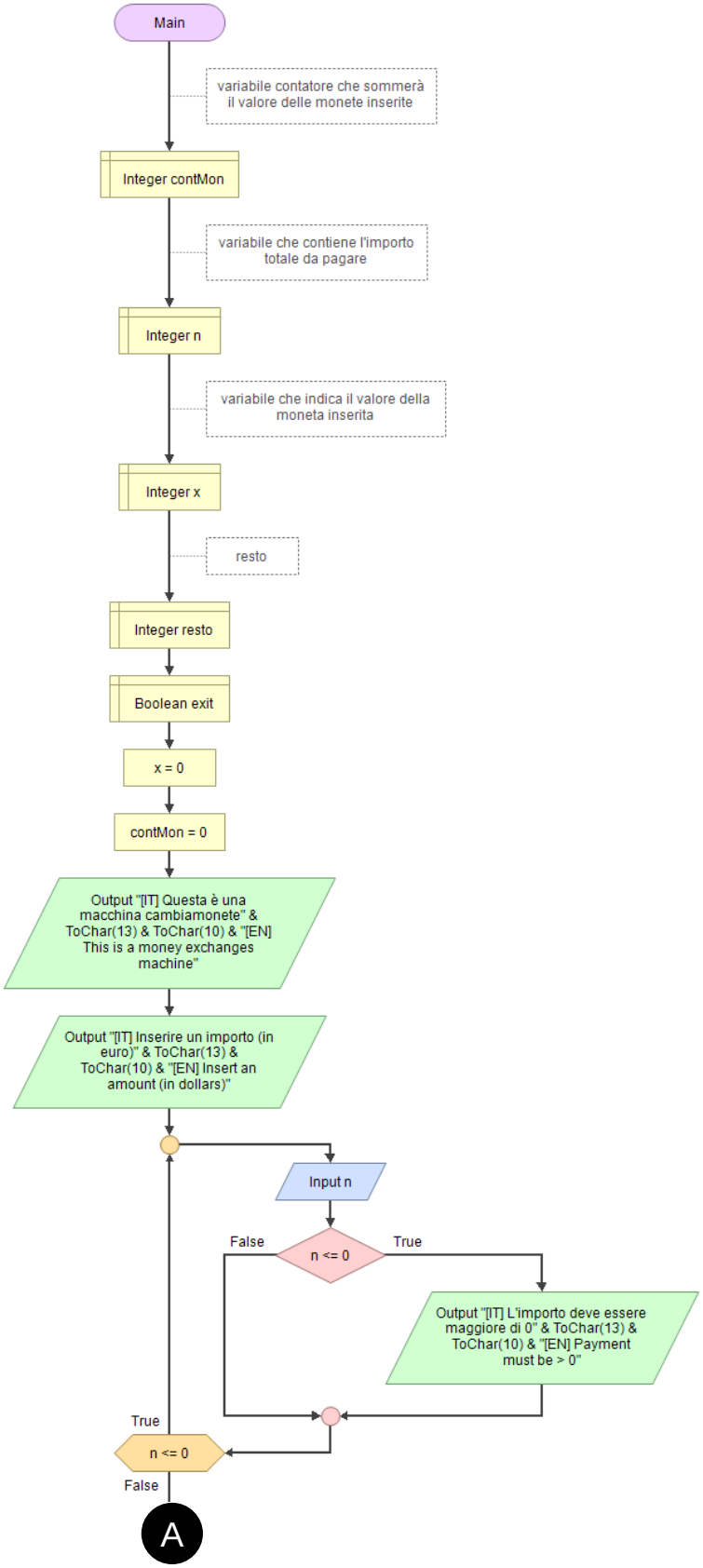
Soda machine

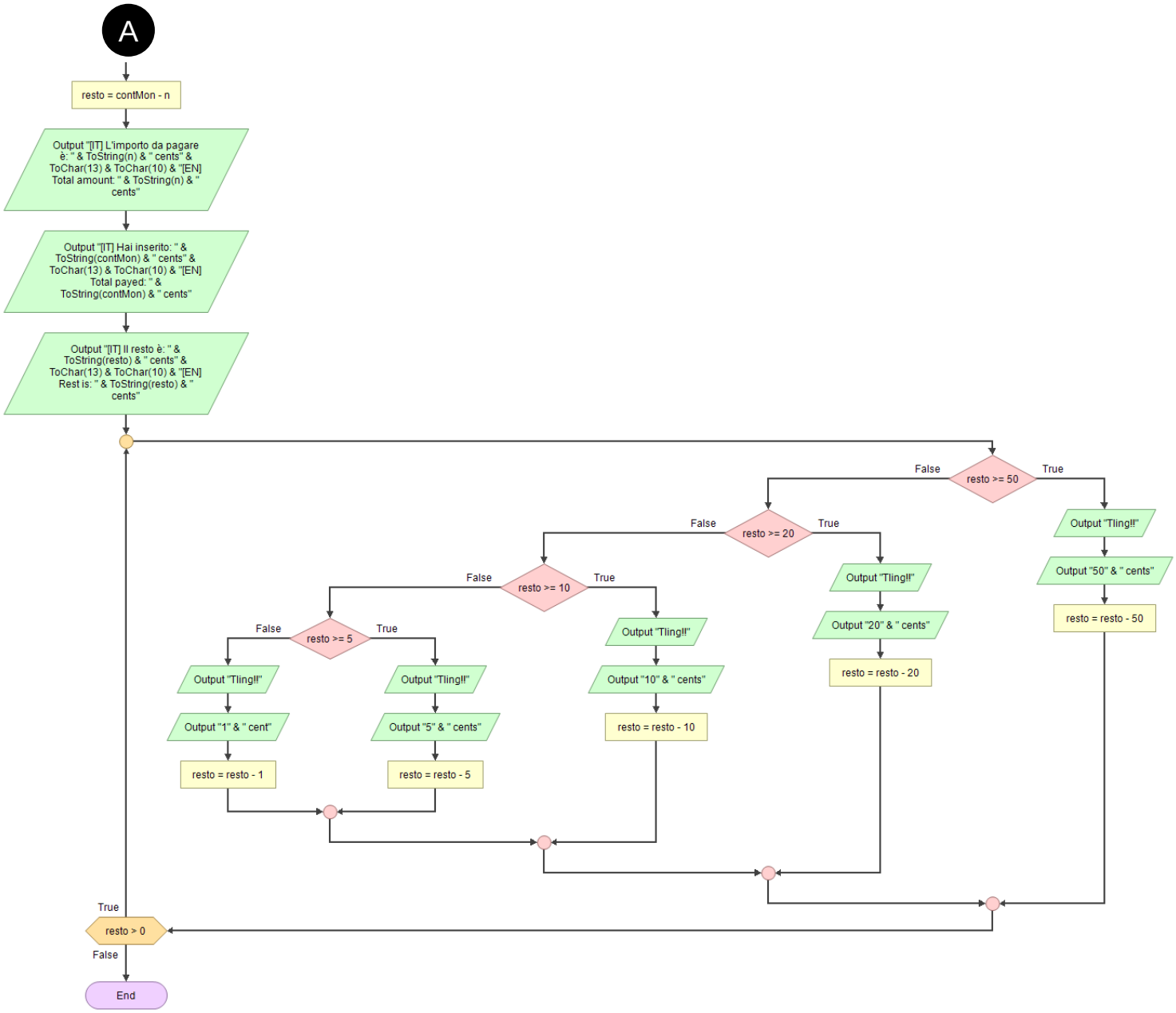



A
resto = contMon - n
Output "[IT] L'importo da pagare è: " & ToString(n) & " cents" & ToChar(13) & ToChar(10) & "[EN] Total amount: " & ToString(n) & " cents"
Output "[IT] Hai inserito: " & ToString(contMon) & " cents" & ToChar(13) & ToChar(10) & "[EN] Total payed: " & ToString(contMon) & " cents"
Output "[IT] Il resto è: " & ToString(resto) & " cents" & ToChar(13) & ToChar(10) & "[EN] Rest is: " & ToString(resto) & " cents"
resto >= 50
False
True
Output "Tling!!"
Output "50" & " cents"
resto = resto - 50
resto >= 20
False
True
Output "Tling!!"
Output "20" & " cents"
resto = resto - 20
resto >= 10
False
True
Output "Tling!!"
Output "10" & " cents"
resto = resto - 10
resto >= 5
False
True
Output "Tling!!"
Output "1" & " cent"
resto = resto - 1
Output "Tling!!"
Output "5" & " cents"
resto = resto - 5
resto > 0
True
False
End

53. G ‖ Cambia monete

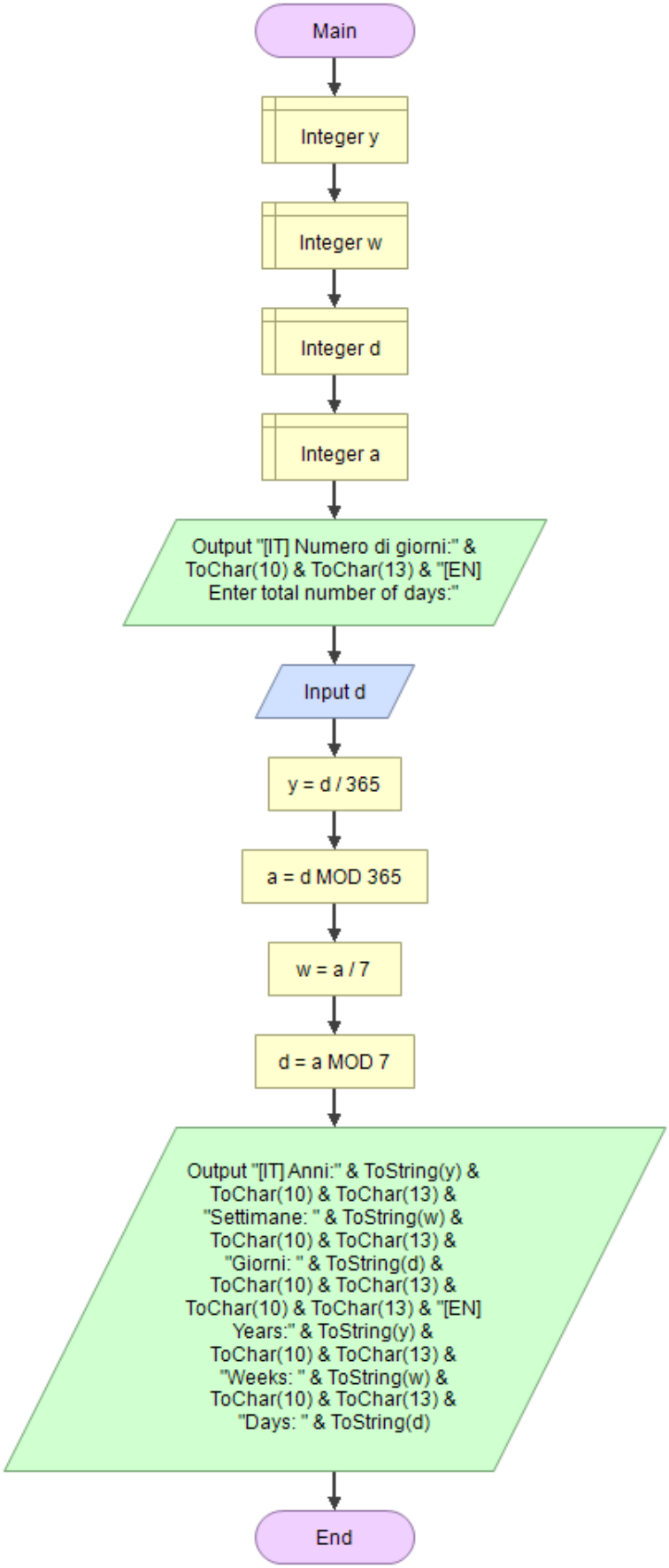
Money exchange




A
resto = contMon - n
Output "[IT] L'importo da pagare è: " & ToString(n) & " cents" & ToChar(13) & ToChar(10) & "[EN] Total amount: " & ToString(n) & " cents"
Output "[IT] Hai inserito: " & ToString(contMon) & " cents" & ToChar(13) & ToChar(10) & "[EN] Total payed: " & ToString(contMon) & " cents"
Output "[IT] Il resto è: " & ToString(resto) & " cents" & ToChar(13) & ToChar(10) & "[EN] Rest is: " & ToString(resto) & " cents"
resto >= 50
False
True
Output "Tling!!"
Output "50" & " cents"
resto = resto - 50
resto >= 20
False
True
Output "Tling!!"
Output "20" & " cents"
resto = resto - 20
resto >= 10
False
True
Output "Tling!!"
Output "10" & " cents"
resto = resto - 10
resto >= 5
False
True
Output "Tling!!"
Output "1" & " cent"
resto = resto - 1
Output "Tling!!"
Output "5" & " cents"
resto = resto - 5
resto > 0
True
False
End

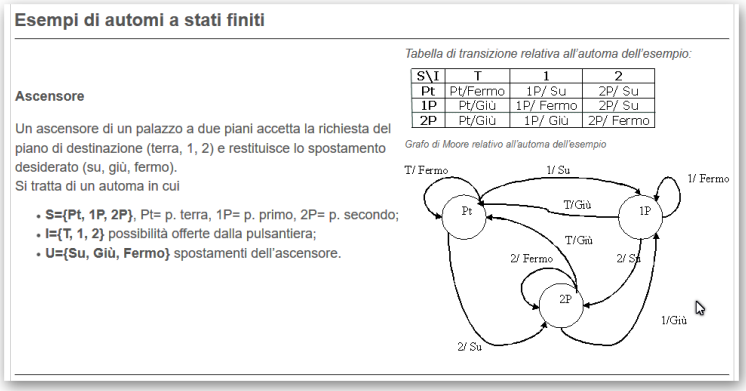
54. G ‖ Converte un certo numero giorni in anni, settimane, giorni

Convert given number of days into years, weeks and days



55. G ‖ Ascensore

Lift



Esempi di automi a stati finiti

Ascensore

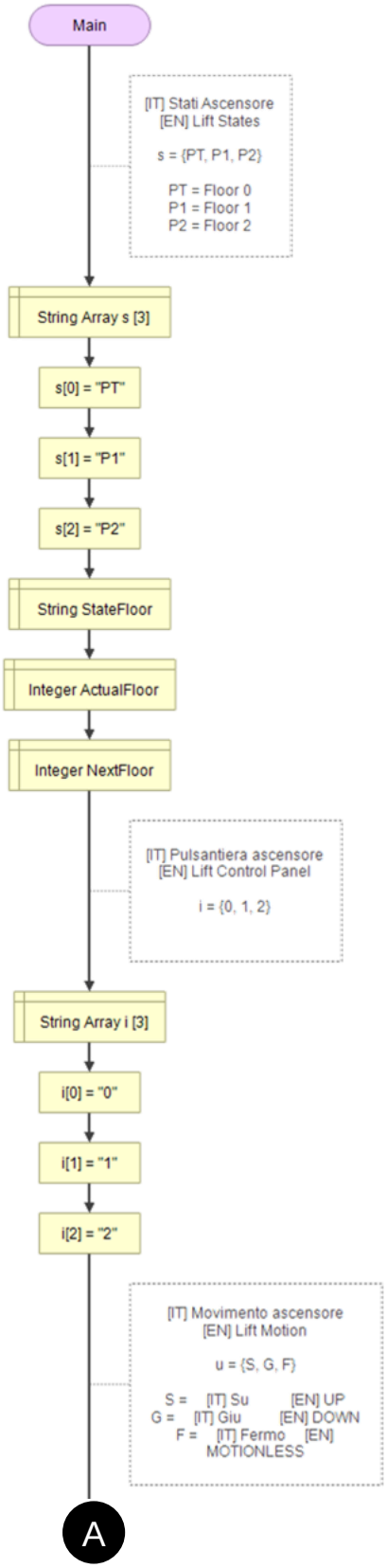
Un ascensore di un palazzo a due piani accetta la richiesta del piano di destinazione (terra, 1, 2) e restituisce lo spostamento desiderato (su, giù, fermo).
Si tratta di un automa in cui

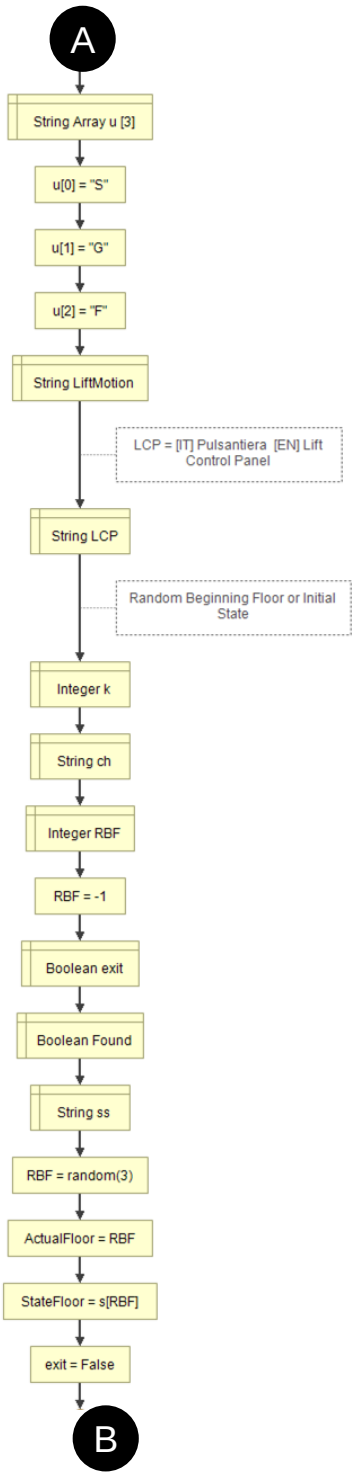
- **S={Pt, 1P, 2P}**, Pt= p. terra, 1P= p. primo, 2P= p. secondo;
- **I={T, 1, 2}** possibilità offerte dalla pulsantiera;
- **U={Su, Giù, Fermo}** spostamenti dell'ascensore.

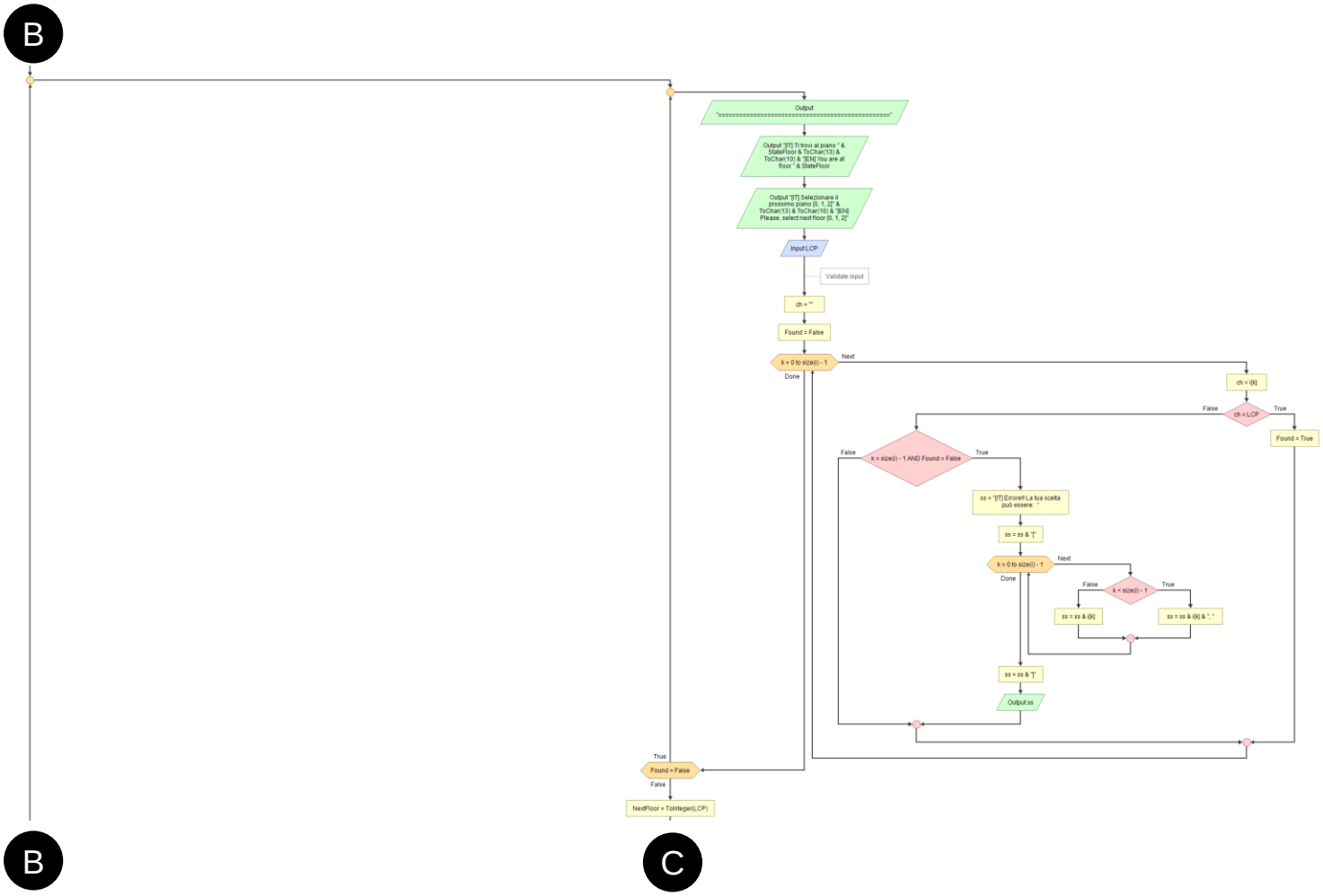
Tabella di transizione relativa all'automa dell'esempio:

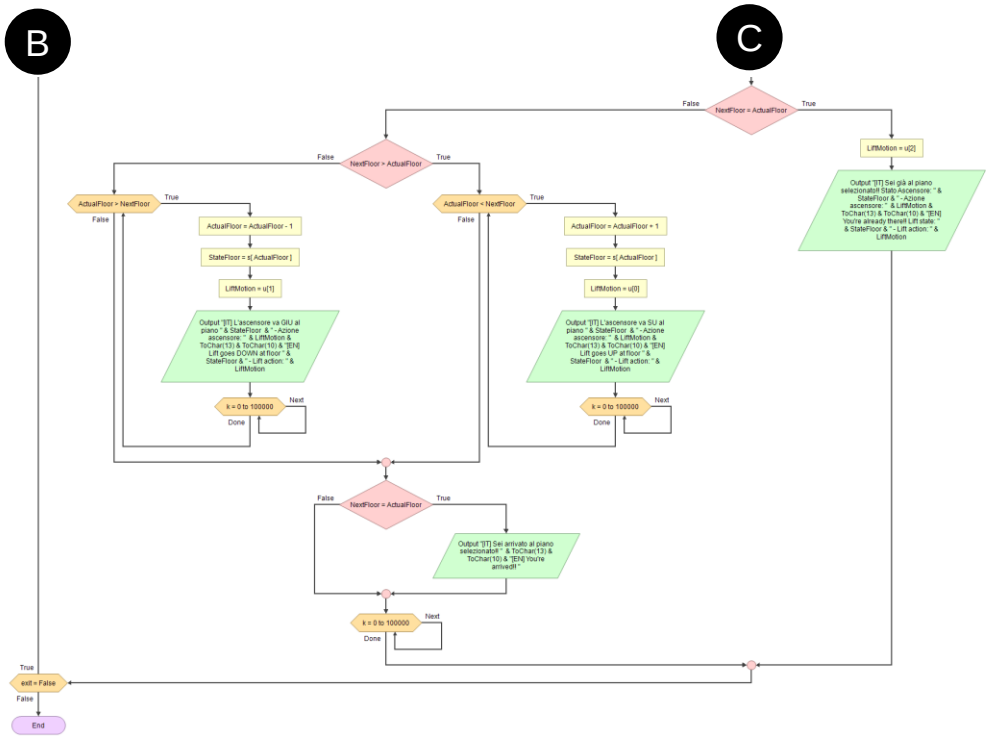
S\I	T	1	2
Pt	Pt/Fermo	1P/ Su	2P/ Su
1P	Pt/Giù	1P/ Fermo	2P/ Su
2P	Pt/Giù	1P/ Giù	2P/ Fermo

Grafo di Moore relativo all'automa dell'esempio



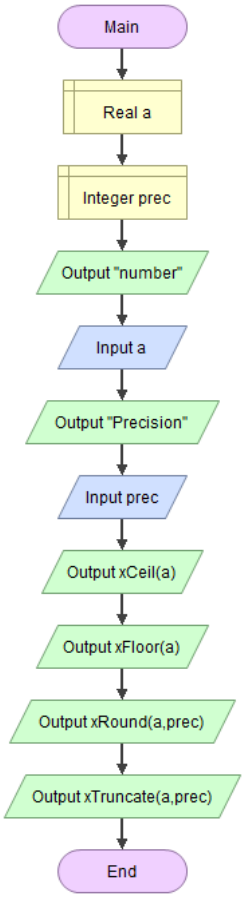



B
Output "==========="
Output "[IT] Ti trovi al piano " & StateFloor & ToChar(13) & ToChar(10) & "[EN] You are at floor " & StateFloor
Output "[IT] Selezionare il prossimo piano [0, 1, 2]" & ToChar(13) & ToChar(10) & "[EN] Please, select next floor [0, 1, 2]"
Input LCP
Validate input
ch = ""
Found = False
k = 0 to size() - 1
Next
Done
ch = ()[k]
ch = LCP
False
True
Found = True
k = size() - 1 AND Found = False
ss = "[IT] Errore!! La tua scelta può essere. "
ss = ss & "["
k = 0 to size() - 1
k = size() - 1
ss = ss & ()[k]
ss = ss & ()[k] & ", "
ss = ss & "]"
Output ss
Found = False
NextFloor = ToInteger(LCP)
B
C

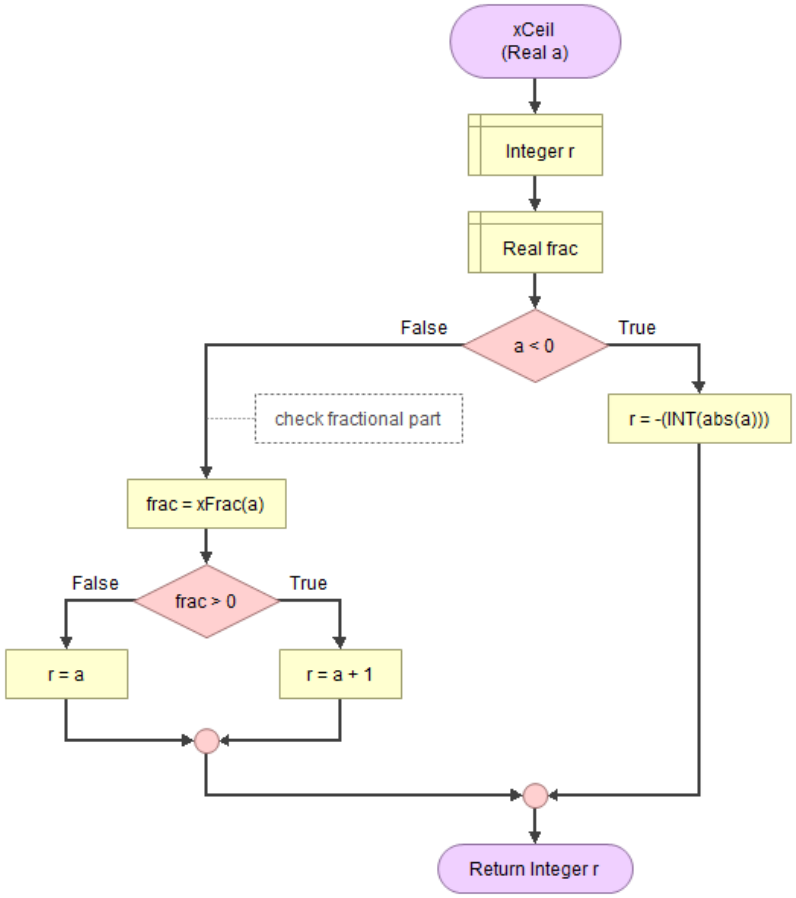

B
C
NextFloor = ActualFloor
False
True
LiftMotion = u[2]
Output "[IT] Sei già al piano selezionato!! Stato Ascensore: " & StateFloor & " - Azione ascensore: " & LiftMotion & ToChar(13) & ToChar(10) & "[EN] You're already there!! Lift state: " & StateFloor & " - Lift action: " & LiftMotion
NextFloor = ActualFloor
False
True
ActualFloor > NextFloor
True
False
ActualFloor = ActualFloor - 1
StateFloor = s[ActualFloor]
LiftMotion = u[1]
Output "[IT] L'ascensore va GIU al piano " & StateFloor & " - Azione ascensore: " & LiftMotion & ToChar(13) & ToChar(10) & "[EN] Lift goes DOWN at floor " & StateFloor & " - Lift action: " & LiftMotion
k = 0 to 100000
Next
Done
ActualFloor < NextFloor
True
False
ActualFloor = ActualFloor + 1
StateFloor = s[ActualFloor]
LiftMotion = u[0]
Output "[IT] L'ascensore va SU al piano " & StateFloor & " - Azione ascensore: " & LiftMotion & ToChar(13) & ToChar(10) & "[EN] Lift goes UP at floor " & StateFloor & " - Lift action: " & LiftMotion
k = 0 to 100000
Next
Done
NextFloor = ActualFloor
False
True
Output "[IT] Sei arrivato al piano selezionato!! " & ToChar(13) & ToChar(10) & "[EN] You're arrived! "
k = 0 to 100000
Next
Done
True
exit = False
False
End

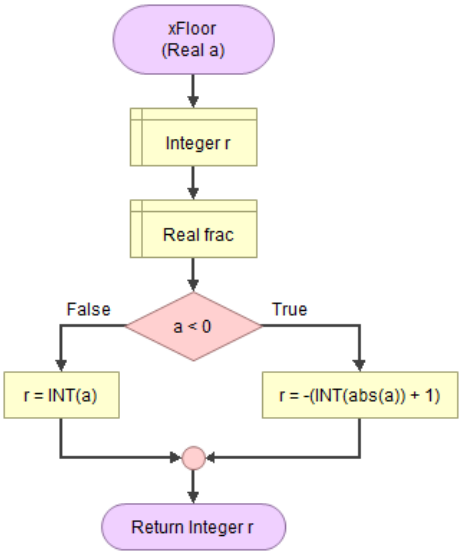
56. G F ‖ Funzioni Ceil, Floor, Frac, Truncate, Round, Mod (anche per numeri negativi)

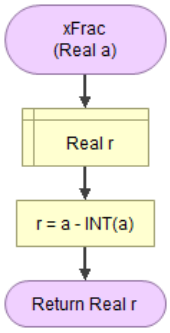
Extended functions Ceil, Floor, Frac, Truncate, Round, Mod (also for negative numbers)

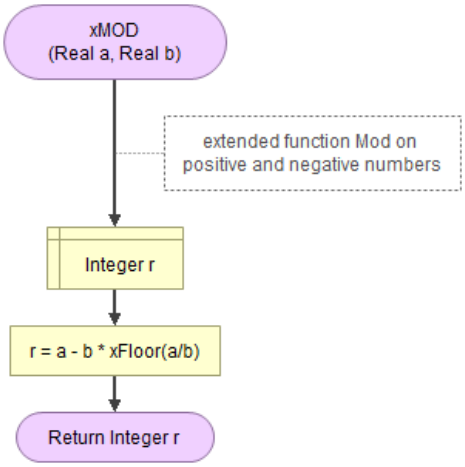


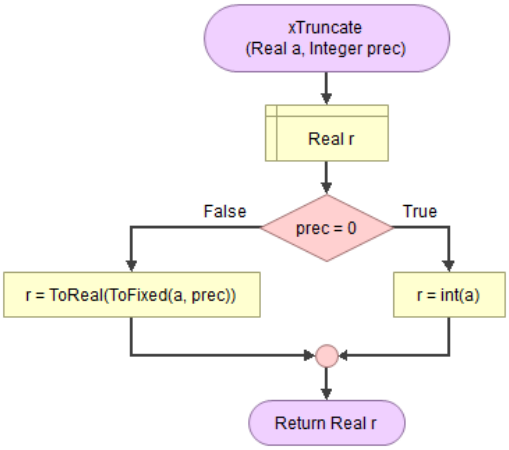
Num	ceil($numeo)	floor($numeo)	round($numeo)	round($numeo,$prec)	intval($numeo)	abs($numeo)
63.53342402	64	63	64	63.53	63	63.53342402
63.48956246	64	63	63	63.49	63	63.48956246
38.7747401	39	38	39	38.77	38	38.7747401
38.46072359	39	38	38	38.46	38	38.46072359
69.7090379	70	69	70	69.71	69	69.7090379
69.4140327	70	69	69	69.41	69	69.4140327
72.79048002	73	72	73	72.79	72	72.79048002
72.36195316	73	72	72	72.36	72	72.36195316
73.56401868	74	73	74	73.56	73	73.56401868
73.05623686	74	73	73	73.06	73	73.05623686
-23.76711166	-23	-24	-24	-23.77	-23	23.76711166
-23.29245825	-23	-24	-23	-23.29	-23	23.29245825
64	64	64	64	64	64	64
48.29816544	49	48	48	48.3	48	48.29816544
test57.64782119	0	0	0	0	0	0
36.27934206test	37	36	36	36.28	36	36.27934206

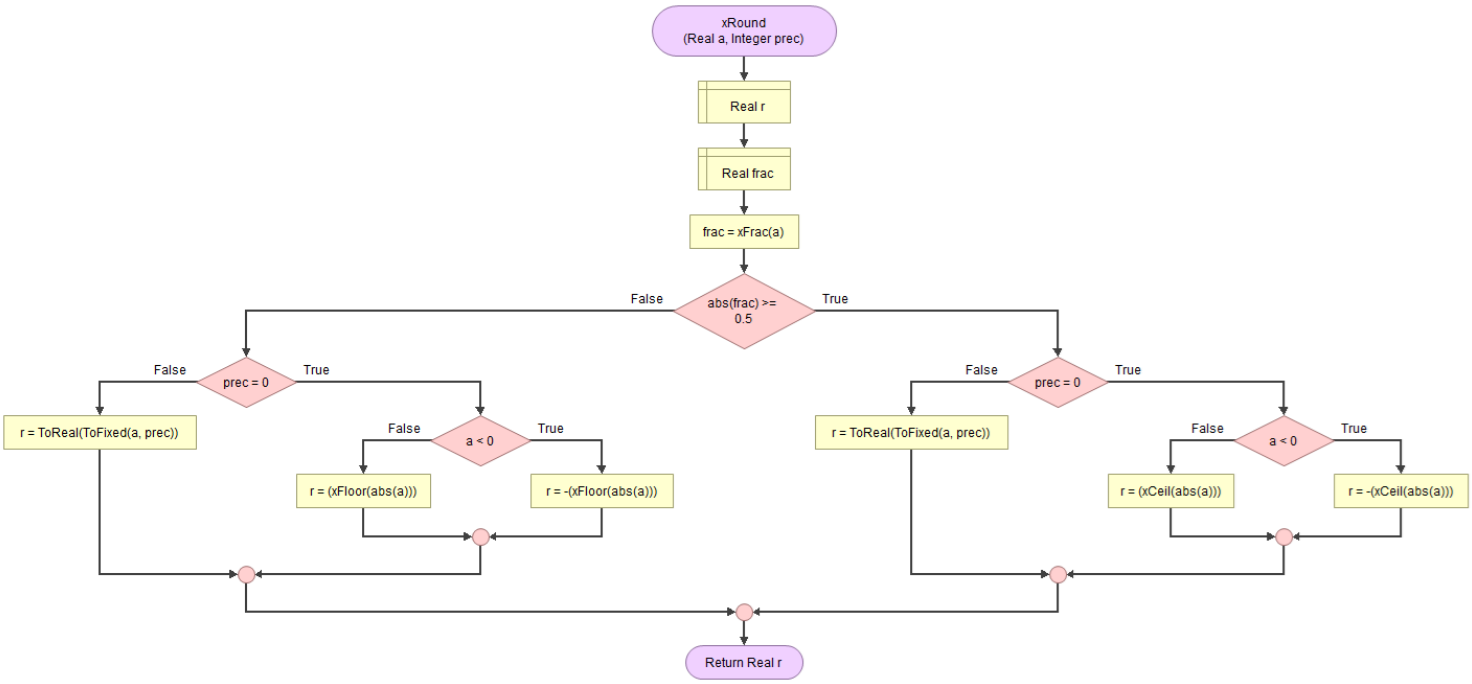





xFrac
(Real a)
Real r
r = a - INT(a)
Return Real r


xMOD
(Real a, Real b)
extended function Mod on positive and negative numbers
Integer r
r = a - b * xFloor(a/b)
Return Integer r


xTruncate
(Real a, Integer prec)
Real r
False
prec = 0
True
r = ToReal(ToFixed(a, prec))
r = int(a)
Return Real r

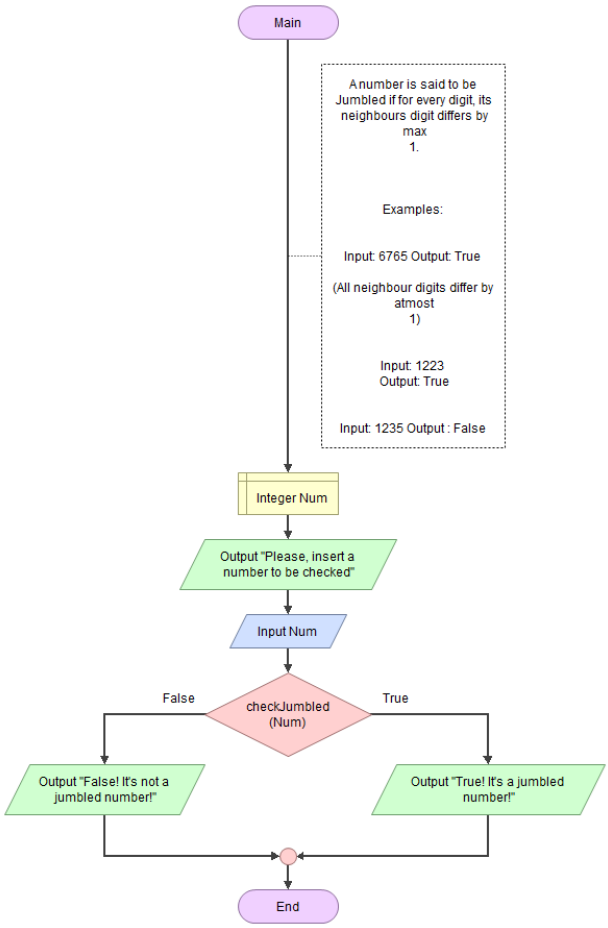

xRound
(Real a, Integer prec)
Real r
Real frac
frac = xFrac(a)
False
abs(frac) >= 0.5
True
False
prec = 0
True
r = ToReal(ToFixed(a, prec))
False
a < 0
True
r = (xFloor(abs(a)))
r = -(xFloor(abs(a)))
False
prec = 0
True
r = ToReal(ToFixed(a, prec))
False
a < 0
True
r = (xCeil(abs(a)))
r = -(xCeil(abs(a)))
Return Real r

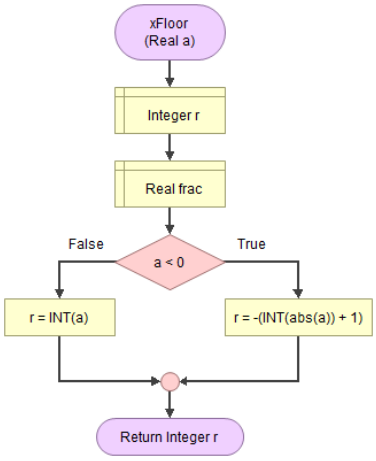
57. G F | Numeri sconnessi

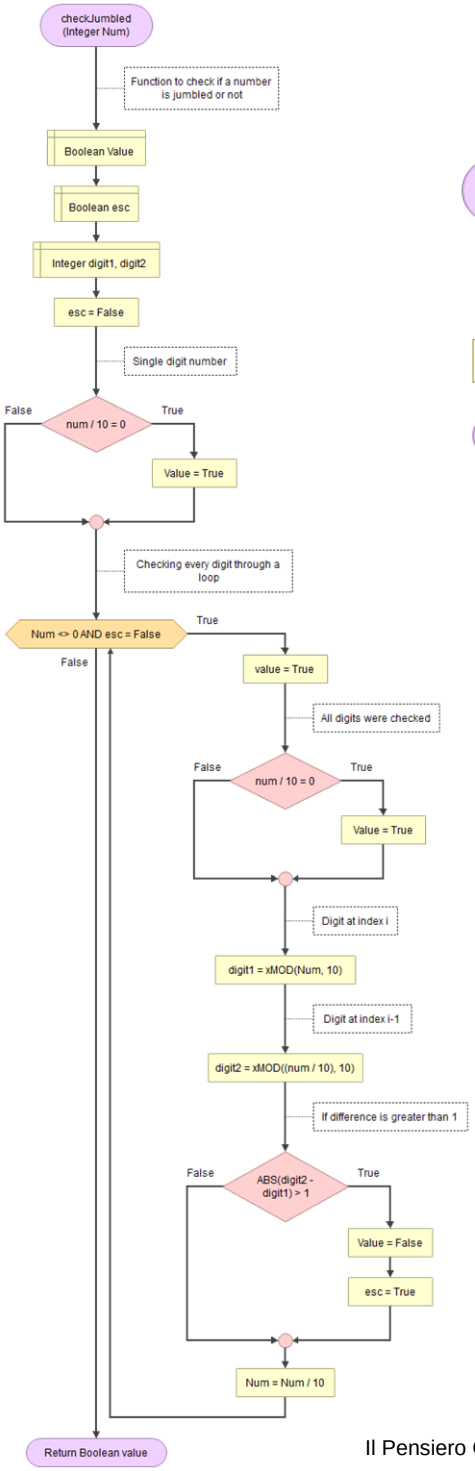
Check if a number is jumbled or not

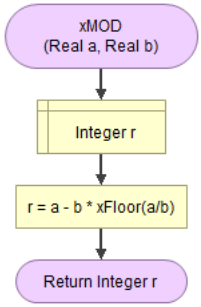
Un numero sconnesso è un numero in cui per ogni sua cifra, la cifra adiacente (a sinistra e a destra) dista max 1.
Es:

Numero	È sconnesso?
1234	vero
8987	vero
2689	falso
13	falso







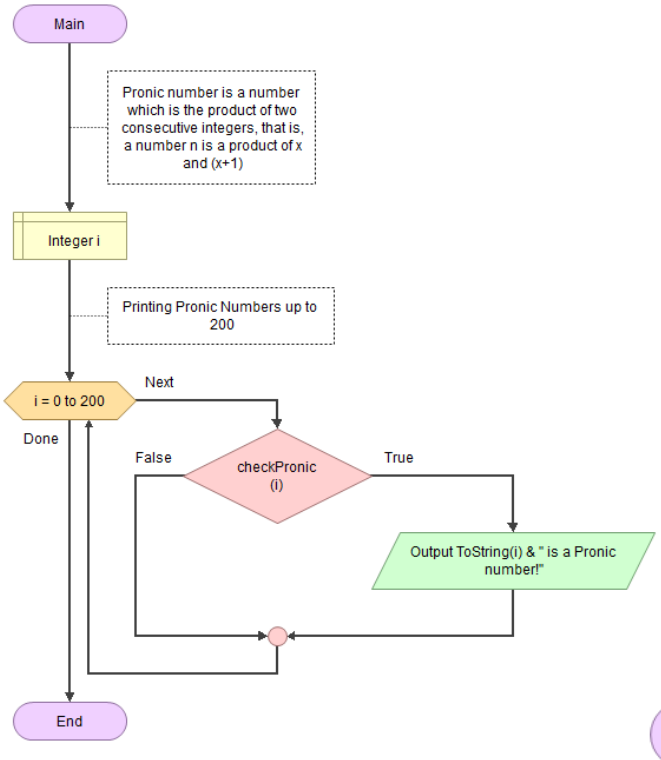


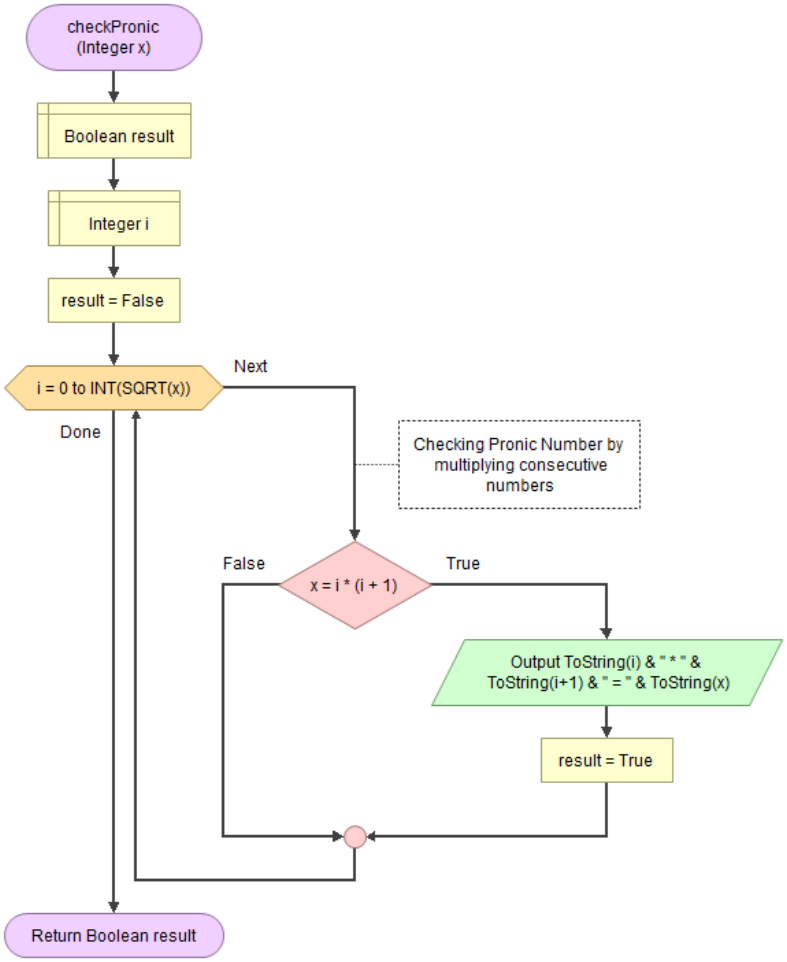
58. G F ‖ Controllare se un numero è pronico

Check if a given number is Pronic

Un **numero oblungo** (o **numero pronico** o anche **numero eteromecico**) è un numero che è il prodotto di due numeri consecutivi, cioè un numero nella forma *n(n+1)*. I primi numeri oblunghi sono:

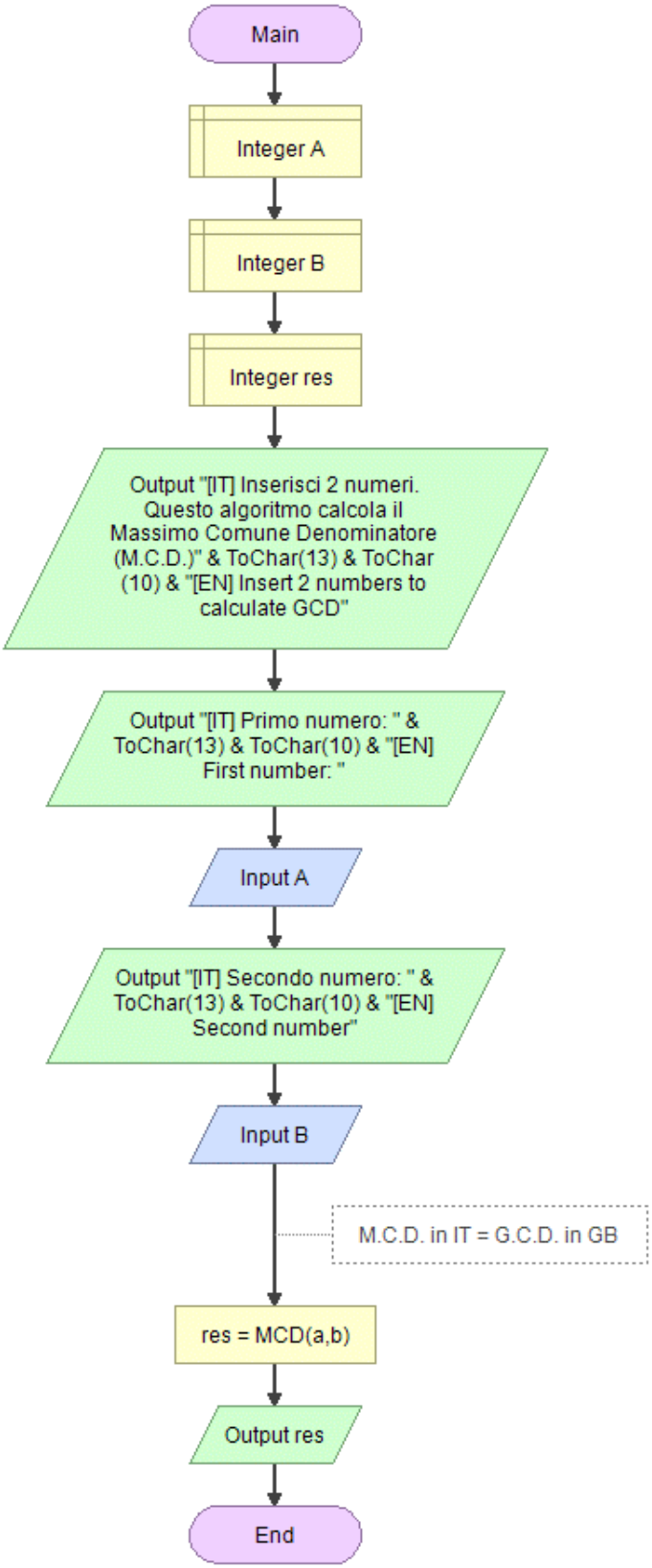
0, 2, 6, 12, 20, 30, 42, 56, 72, 90, 110, 132, 156, 182, 210, 240, 272, 306, 342, 380, 420, 462, ..

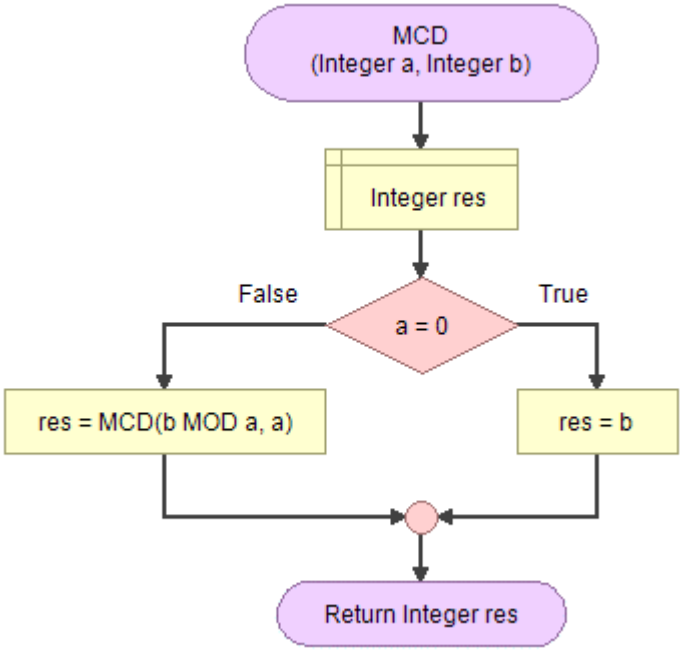




59. **G F R** | Massimo Comune Divisore (M.C.D.) tra due numeri (algoritmo di Euclide ricorsivo)

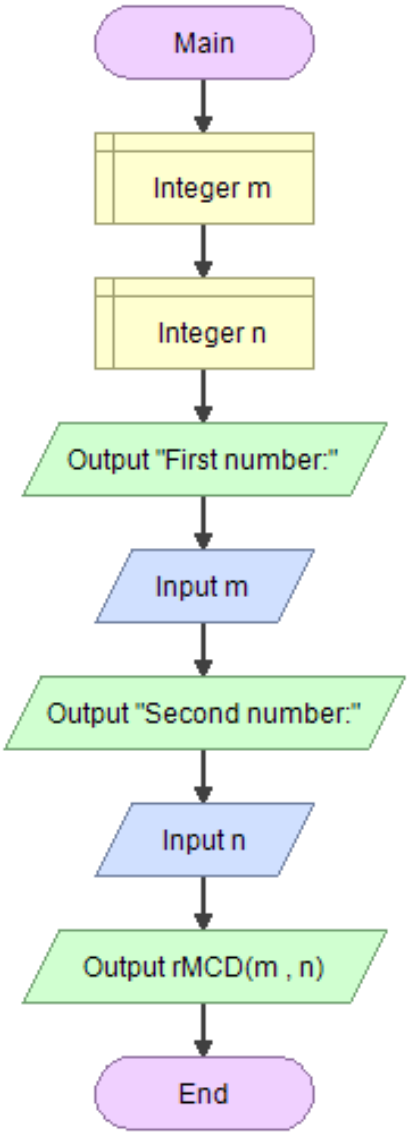
Greatest Common Divisor (GCD) of two integers (recursive Euclide algorithm)

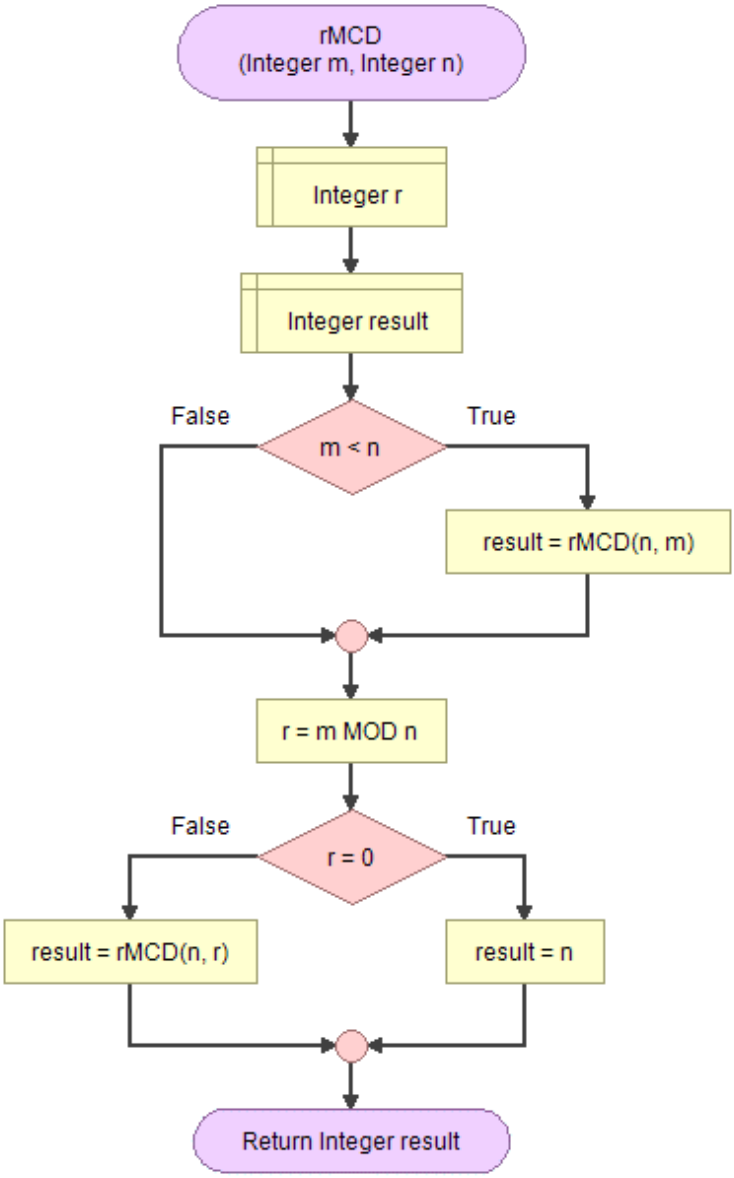




60. **G F R** ‖ M.C.D. ricorsivo tra due numeri interi

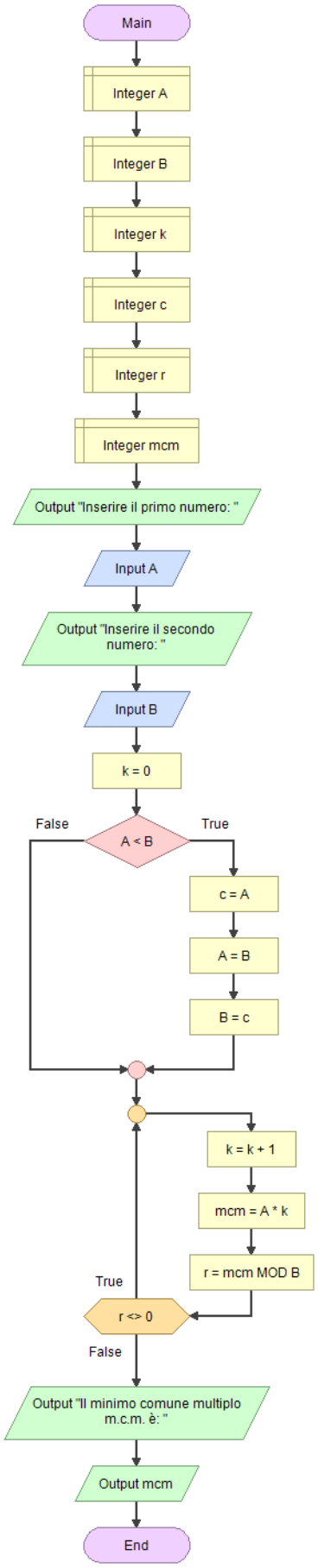
Greatest Common Divisor (GCD) of two integers (recursive)





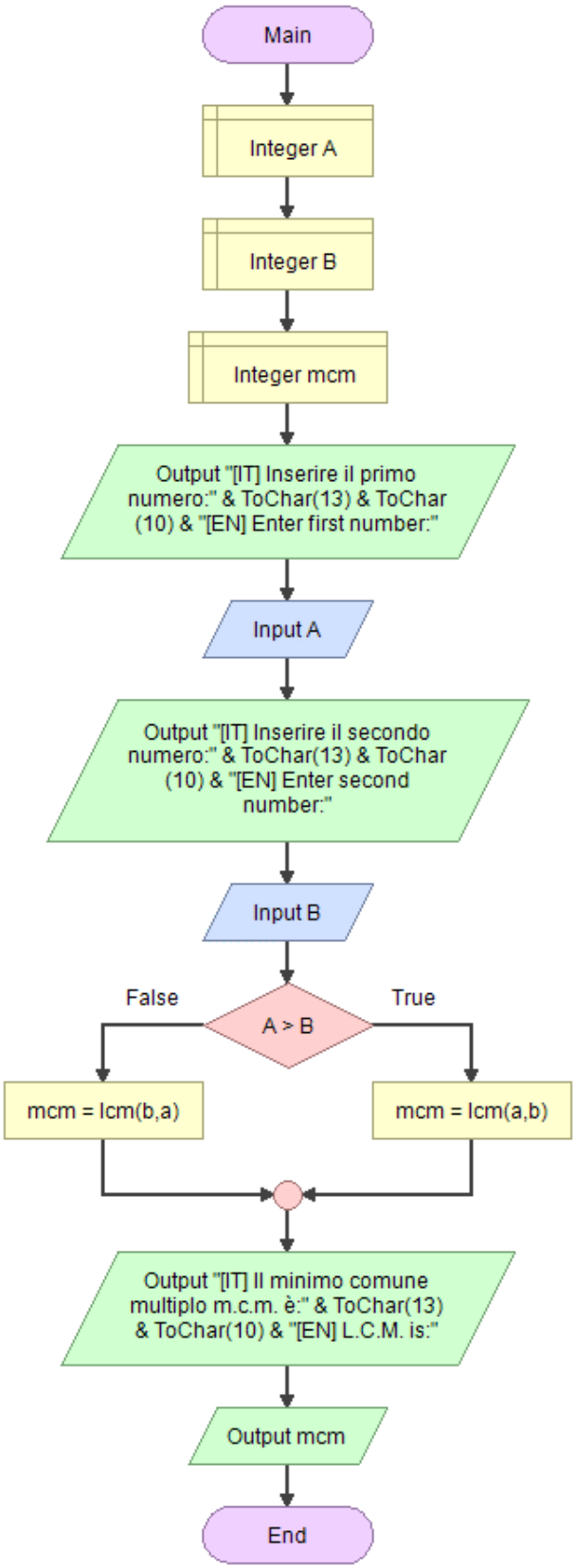
61. G ‖ Minimo comune multiplo (m.c.m.) tra due numeri

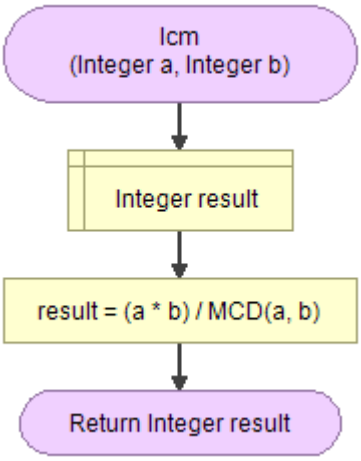
L.C.M. (Lowest Common Multiple) of two numbers

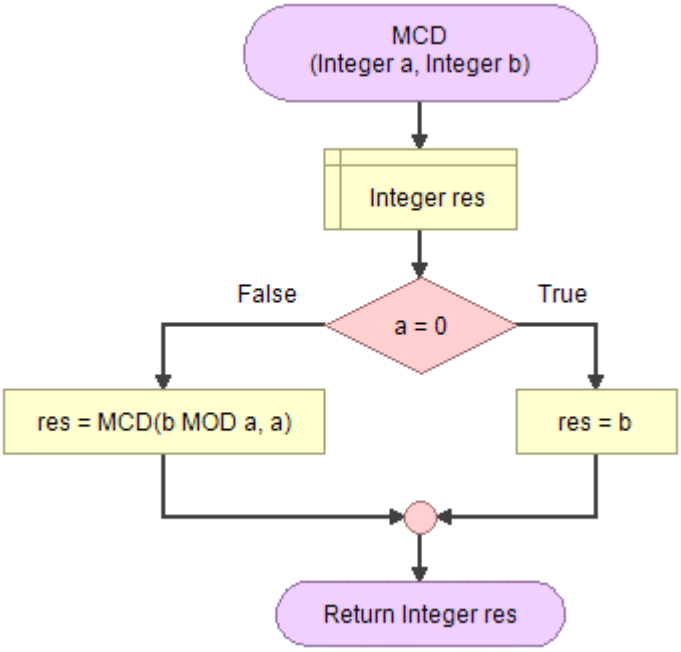


62. **G F R** ‖ Minimo comune multiplo (m.c.m.) tra due numeri ricorsivo

Recursive L.C.M. (Lowest Common Multiple) of two numbers

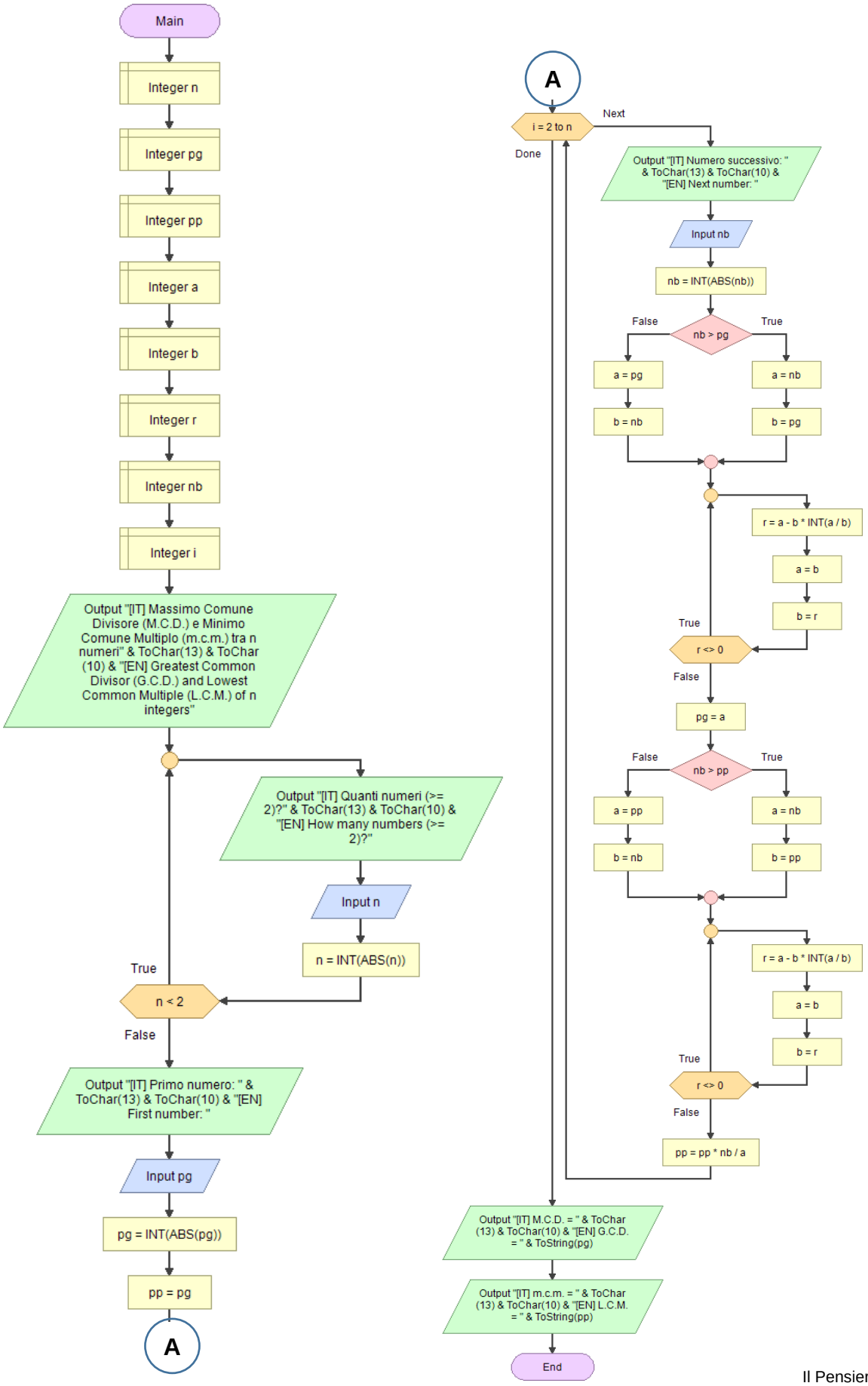






63. G ‖ Massimo Comune Divisore (M.C.D.) e Minimo Comune Multiplo (m.c.m.) tra n numeri

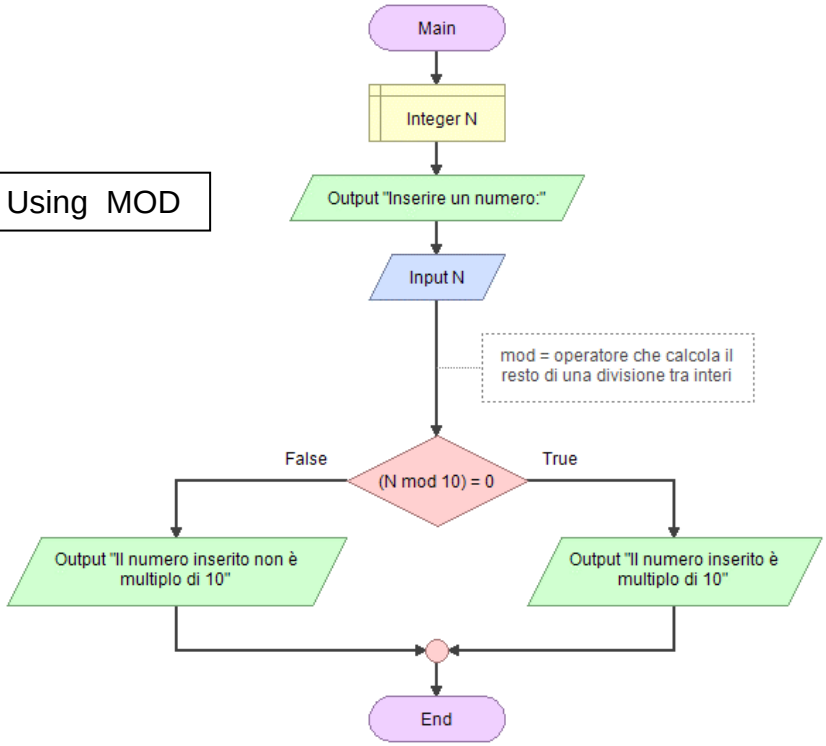
Greatest Common Divisor (G.C.D.) and Lowest Common Multiple (L.C.M.) of n integers



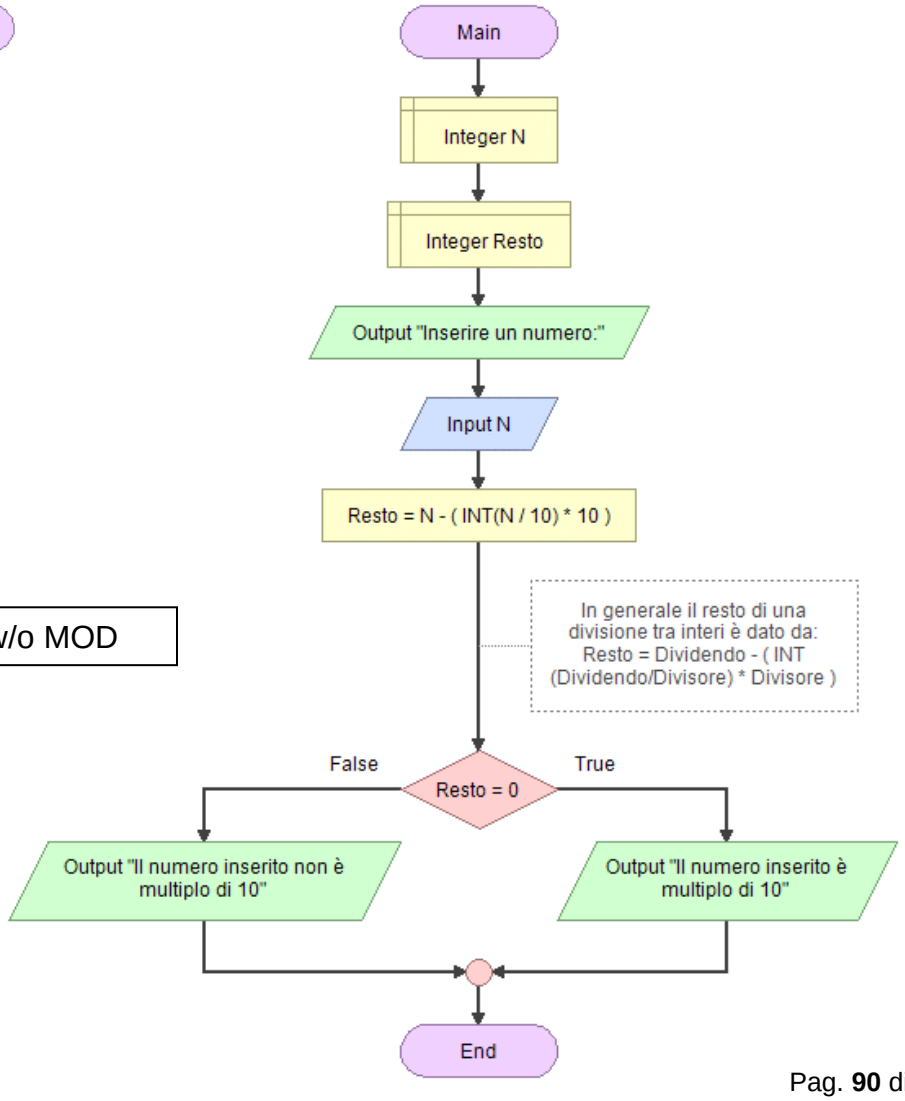
64. G ‖ Dato un numero scrivere se è multiplo di 10 con e senza MOD

Check if a number is a multiple of 10

Using MOD

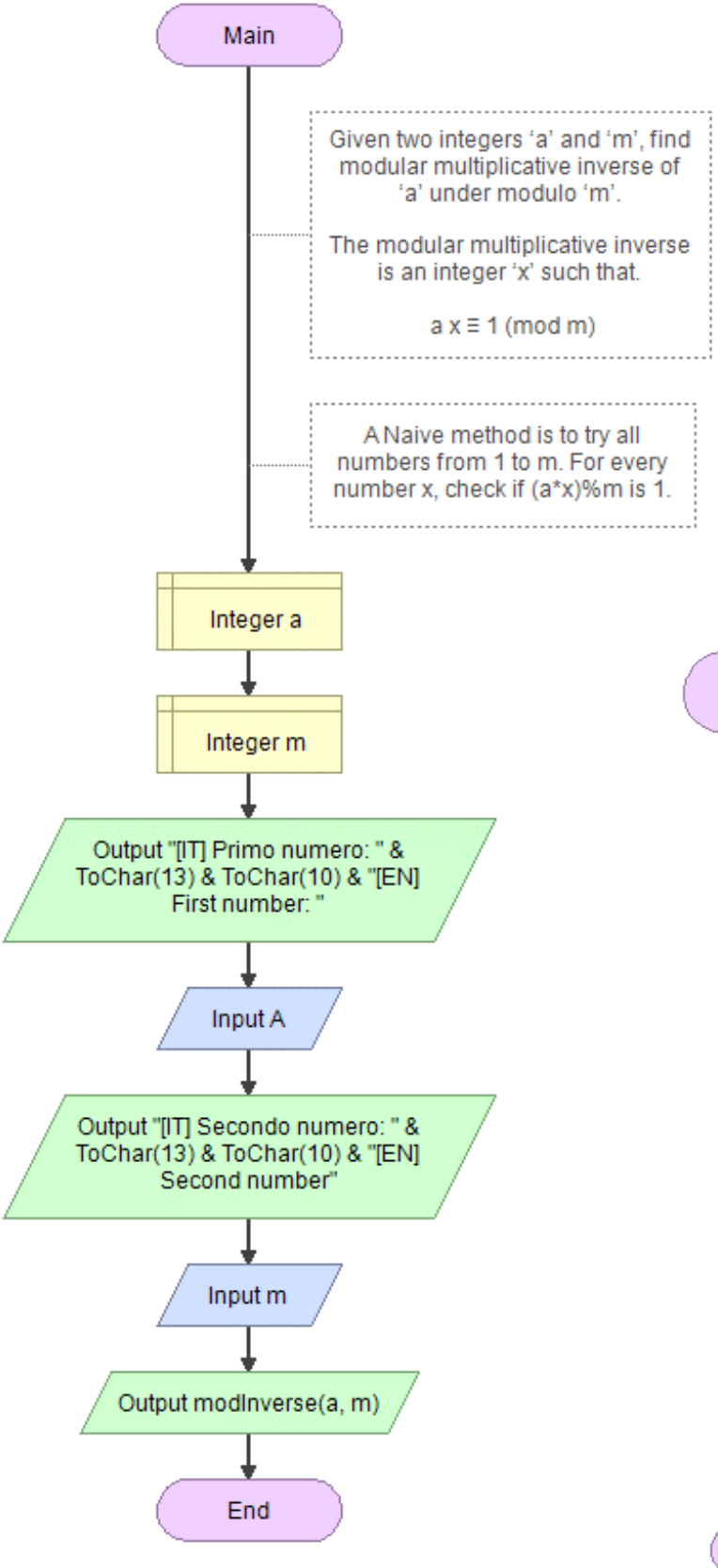


w/o MOD



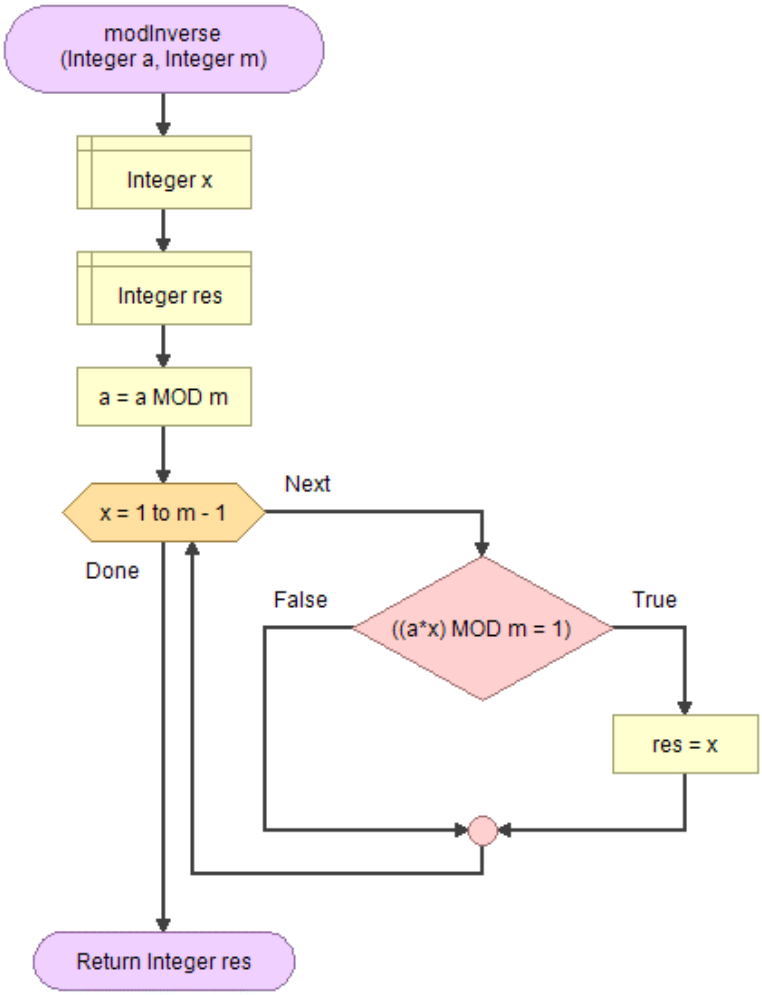
65. G ‖ Dati due interi 'a' e 'm', trovare il moltiplicativo modulare inverso di 'a' sotto modulo 'm'.

Find the modular multiplicative inverse of 'a' in MOD 'm'



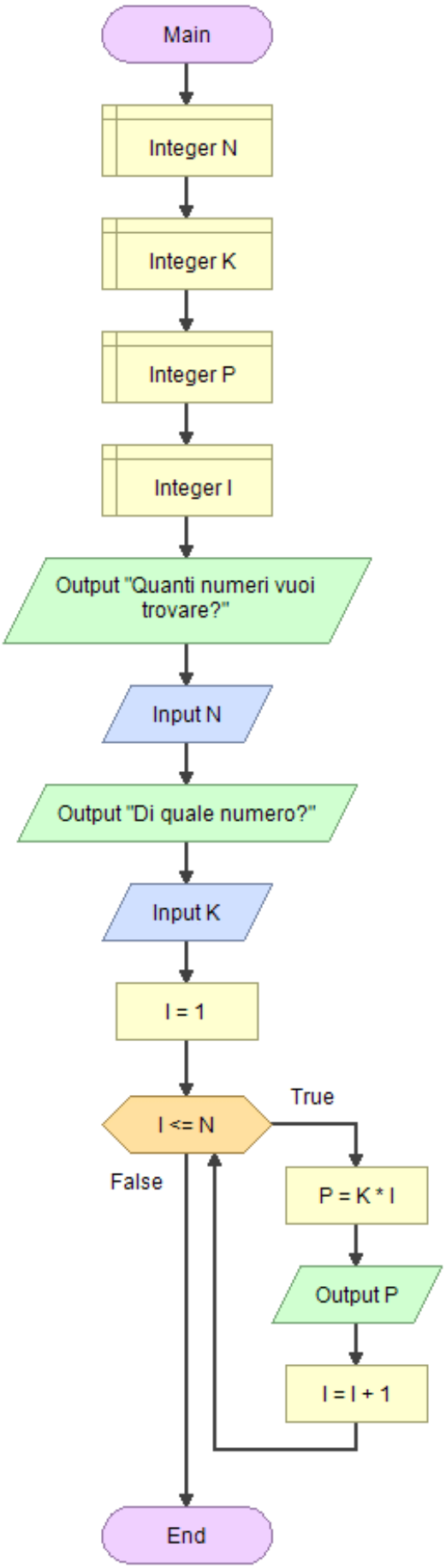
Il moltiplicativo modulare inverso di 'a' sotto modulo 'm' è un intero 'x' tale che:

(a*x) MOD m ≡ 1



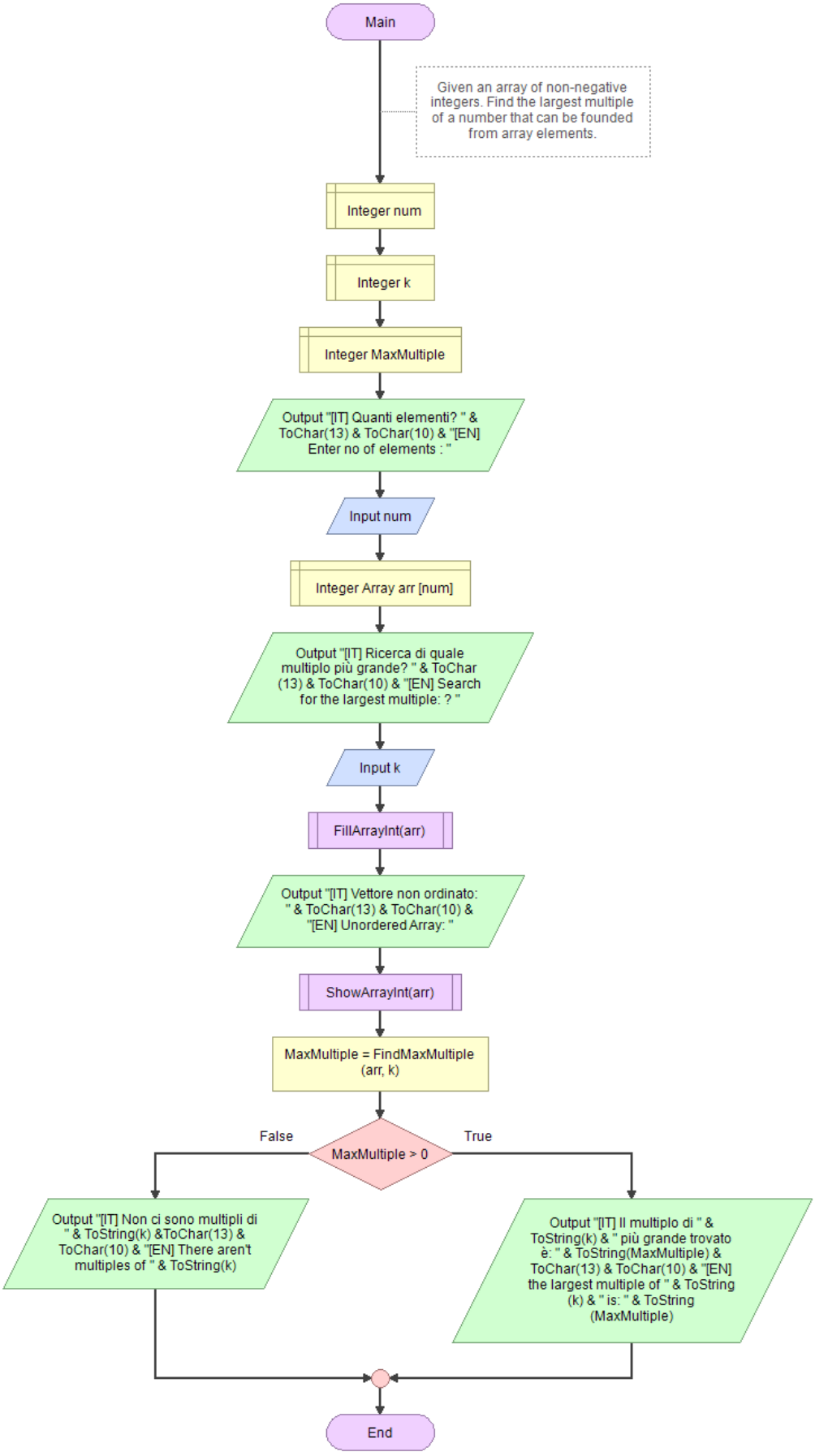
66. G ‖ Dati due interi n e k entrambi maggiori di 0, visualizzare i primi n multipli di k

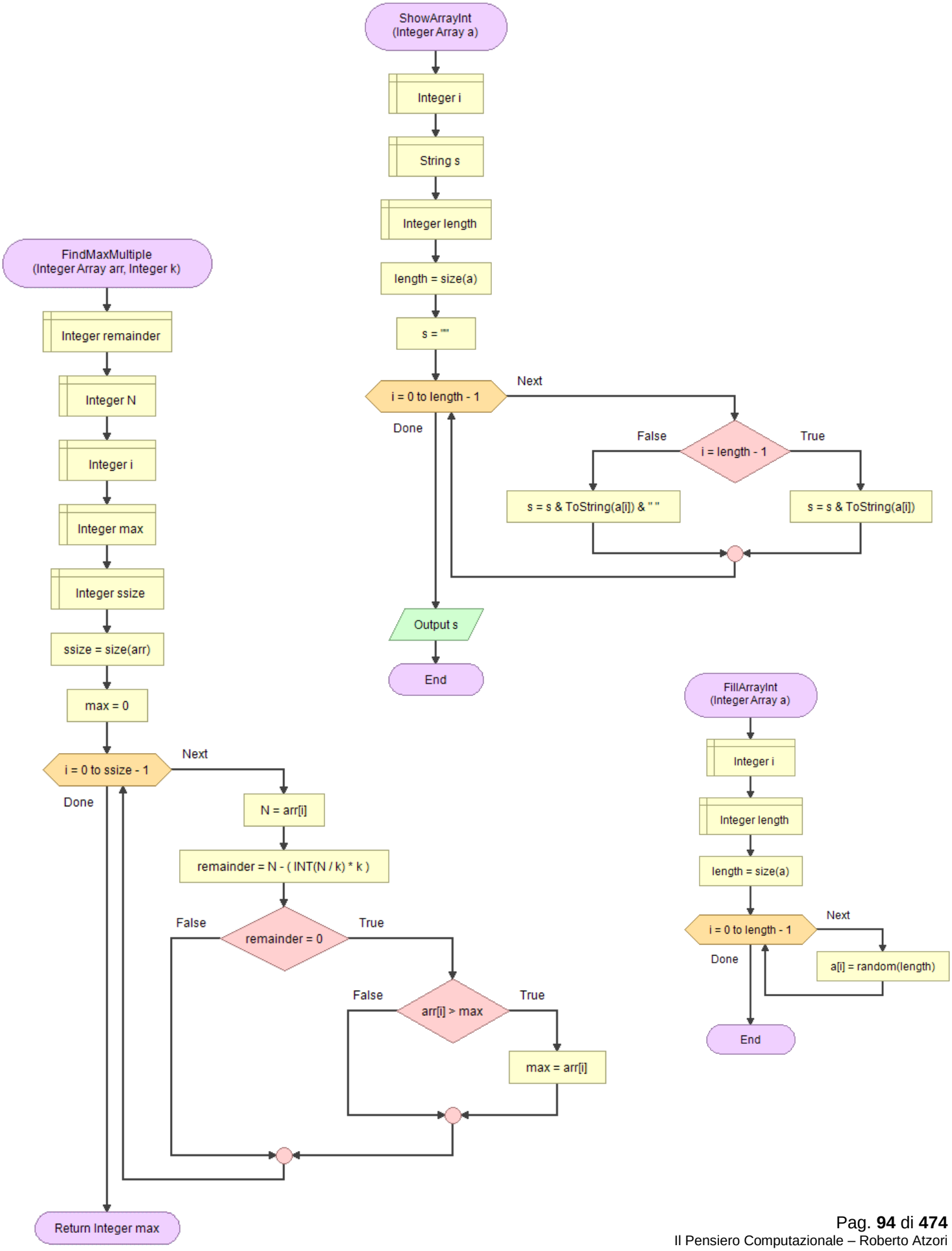
Given two integers N and K both greater than 0, display the first N multiple of K



67. **G S F** | Trovare il più grande multiplo di un numero in una lista di numeri casuali

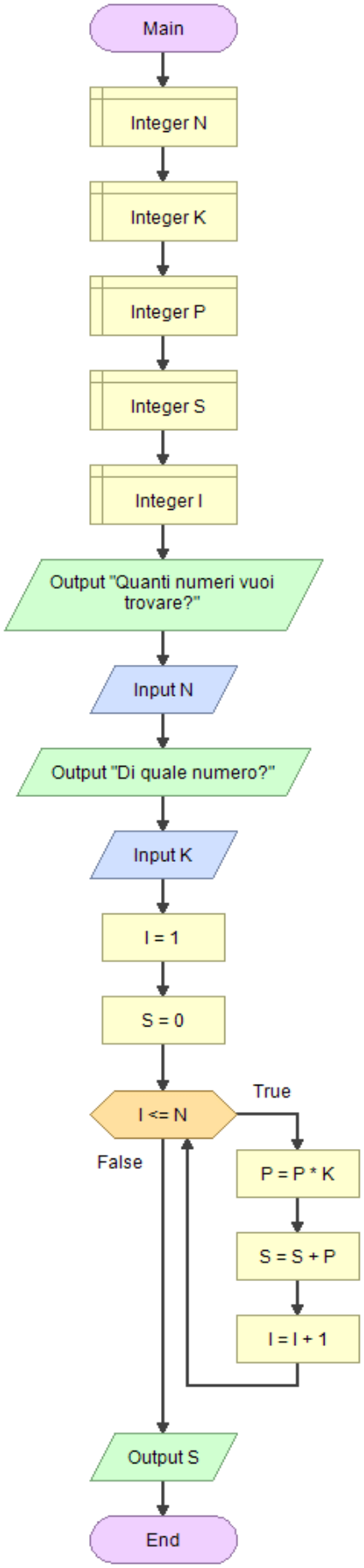
Find the greatest multiple of a number in a list of random numbers




ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
False
i = length - 1
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
End
FindMaxMultiple
(Integer Array arr, Integer k)
Integer remainder
Integer N
Integer i
Integer max
Integer ssize
ssize = size(arr)
max = 0
i = 0 to ssize - 1
Next
Done
N = arr[i]
remainder = N - (INT(N / k) * k)
False
remainder = 0
True
False
arr[i] > max
True
max = arr[i]
Return Integer max
FillArrayInt
(Integer Array a)
Integer i
Integer length
length = size(a)
i = 0 to length - 1
Next
Done
a[i] = random(length)
End

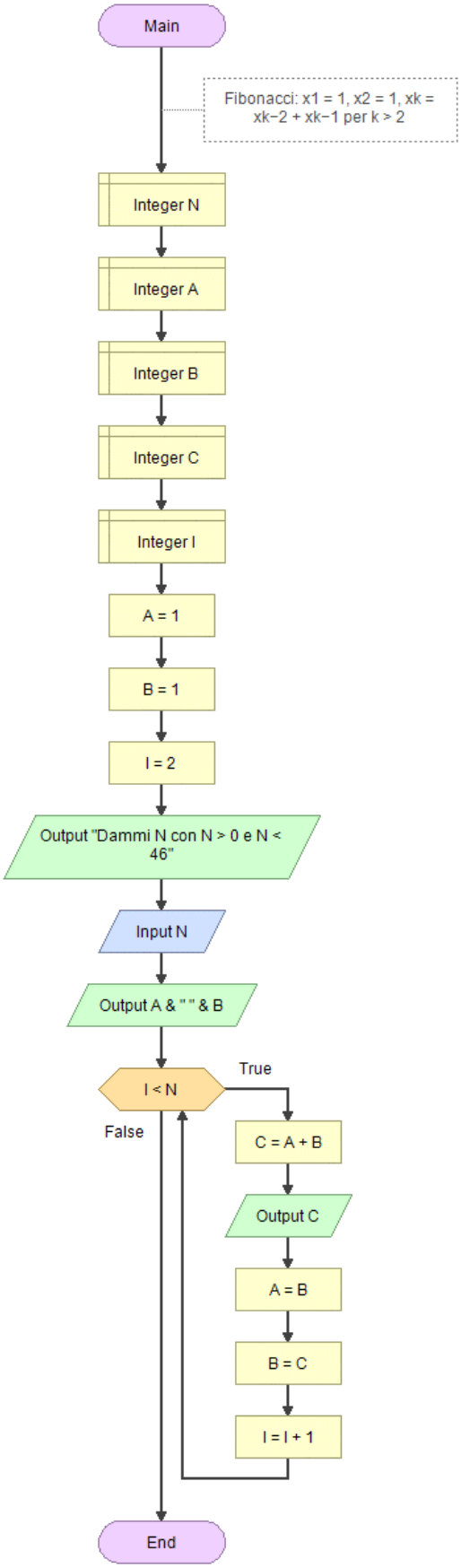
68. G ‖ Dati due interi N e K, entrambi maggiori di 0, stampare il risultato della sommatoria $k + k_2 + k_3 + \ldots + k_n$.

Given two integers N and K, both greater than 0, print the result of the sum $k + k_2 + k_3 + \ldots + k_n$.



69. G ‖ Letto $N > 0$ intero, stampa i primi N numeri della successione di Fibonacci: $x_1 = 1, x_2 = 1, x_k = x_{k-2} + x_{k-1}$ per $k > 2$

Display the first N numbers of the Fibonacci sequence $x_1 = 1, x_2 = 1, x_k = x_{k-2} + x_{k-1}$ per $k > 2$

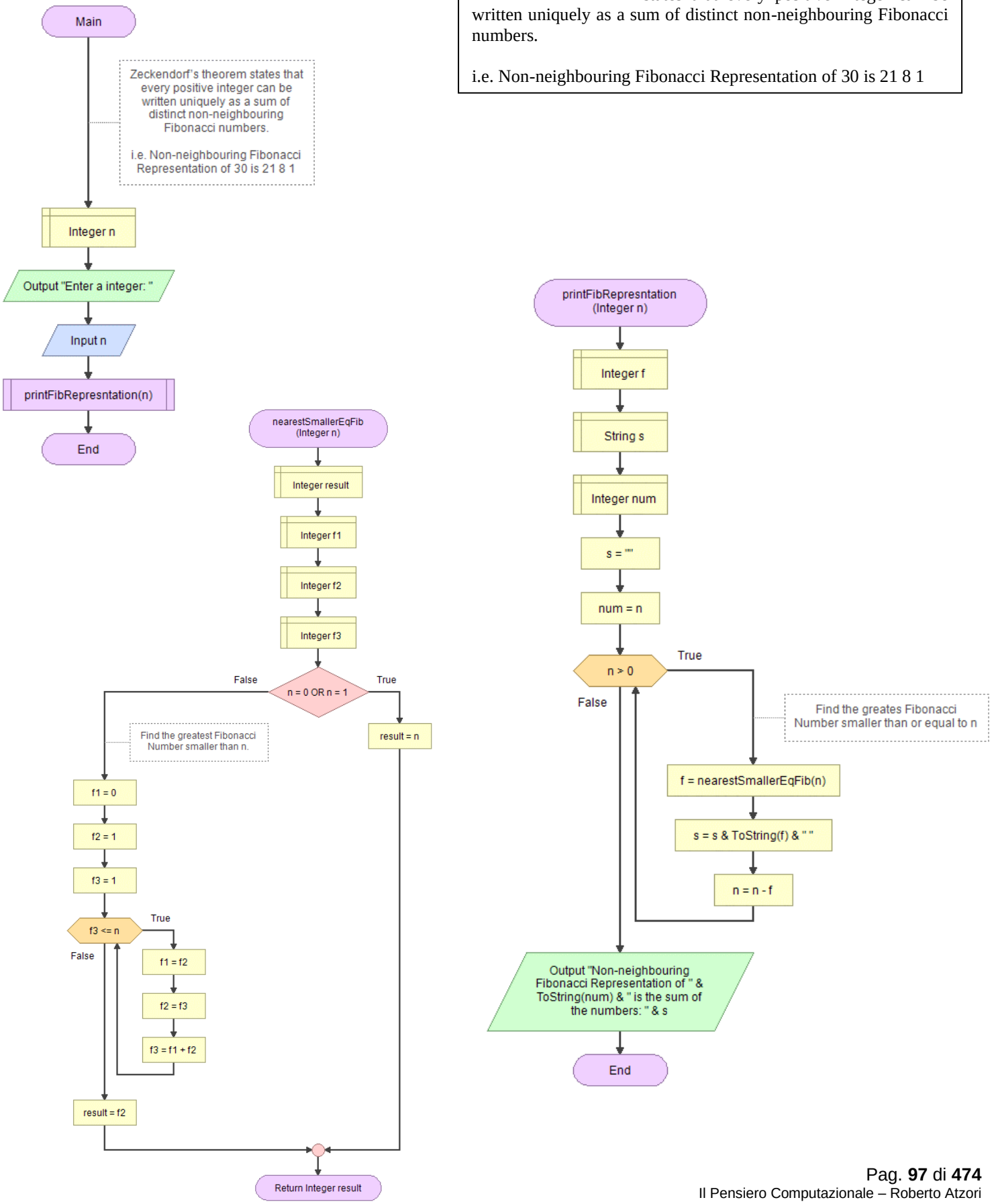


70. G F | Rappresentazione di un numero come somma di numeri non adiacenti di Fibonacci (Teorema di Zeckendorf)

Zeckendorf's Theorem (Non-Neighbouring Fibonacci Representation)

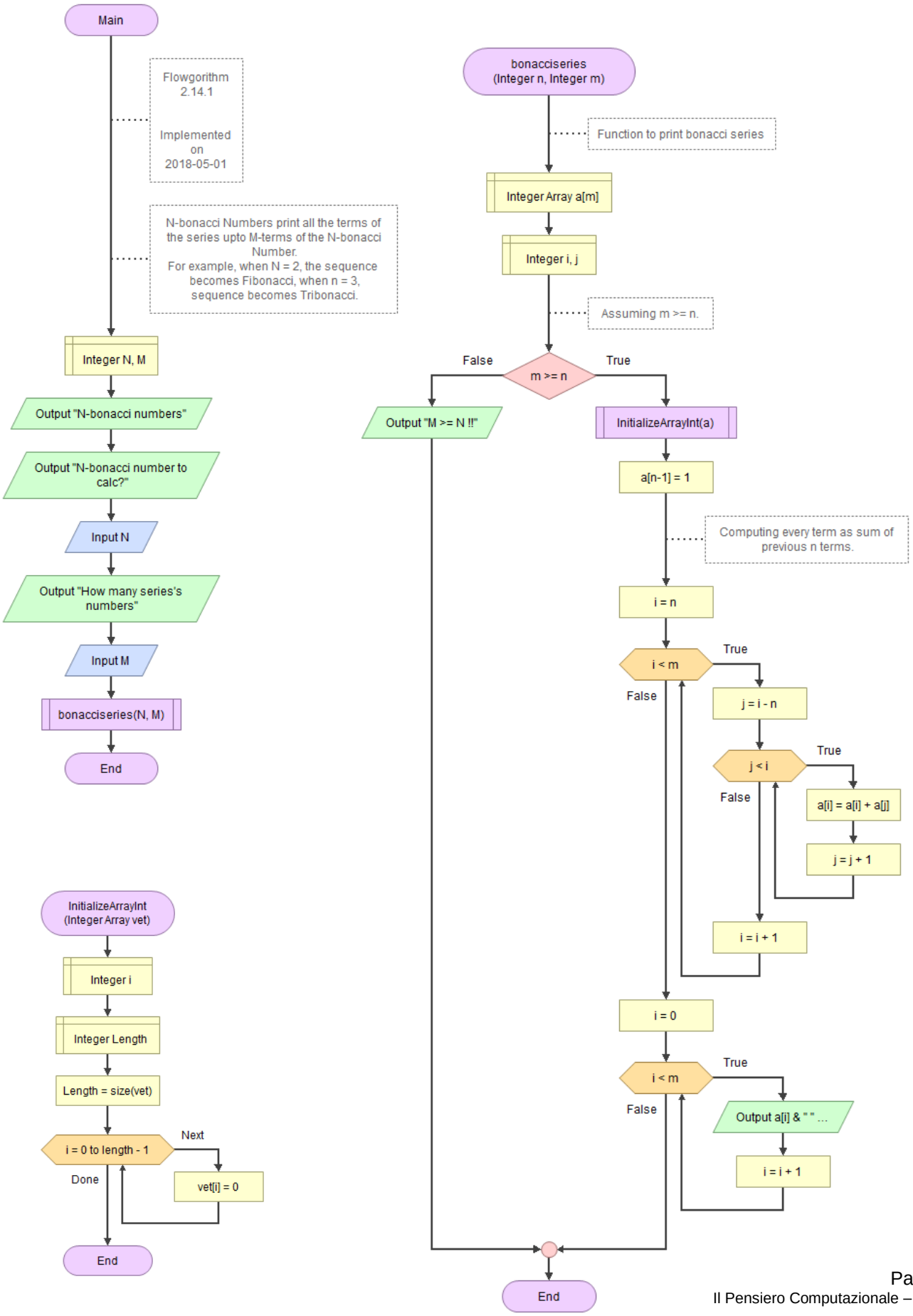
Zeckendorf's theorem states that every positive integer can be written uniquely as a sum of distinct non-neighbouring Fibonacci numbers.

i.e. Non-neighbouring Fibonacci Representation of 30 is 21 8 1



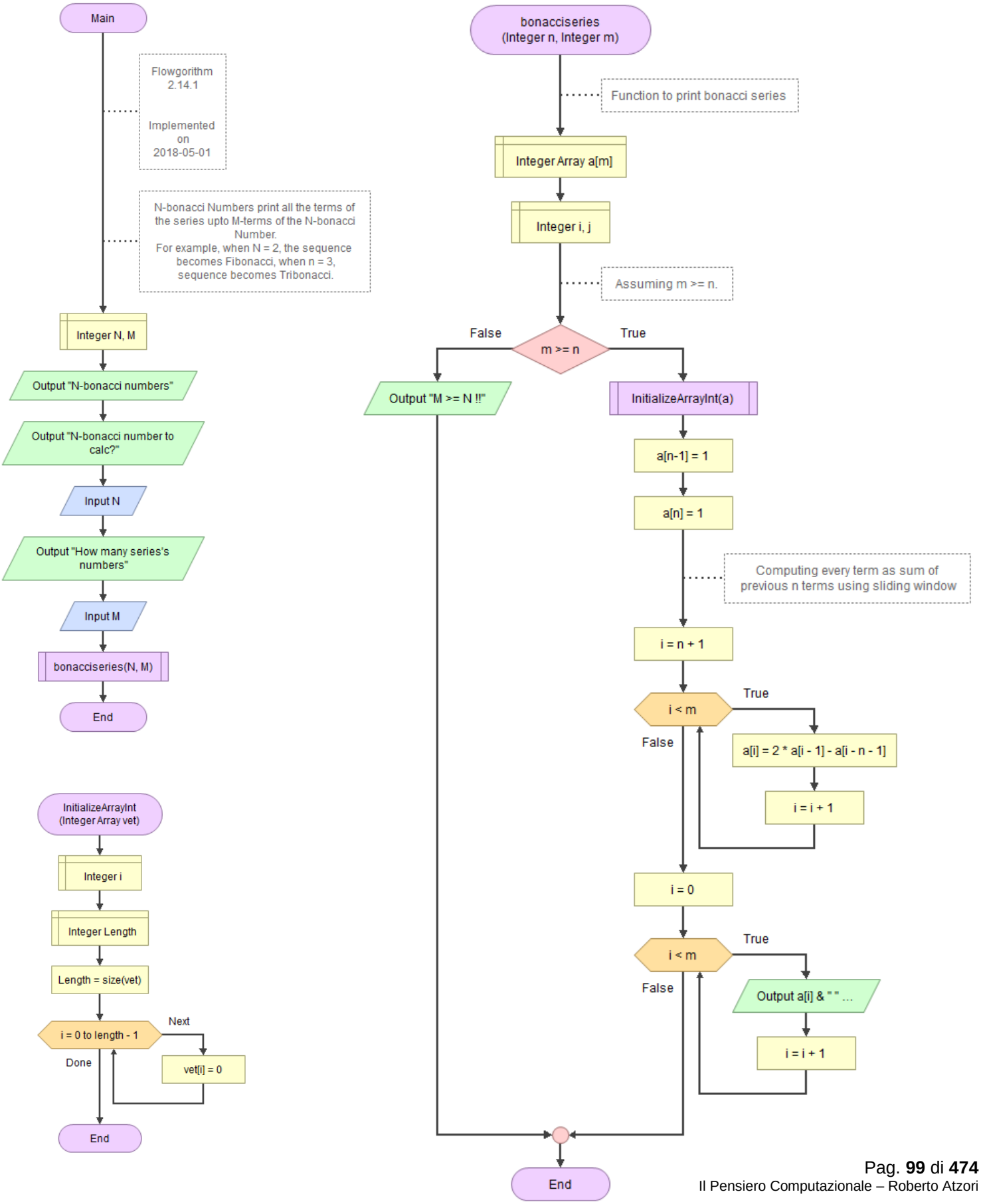
71. GAF ‖ Calcolo dei numeri N-bonacci

N-bonacci Numbers



72. GAF ‖ Calcolo dei numeri N-bonacci (metodo finestra scorrevole)

N-bonacci Numbers (sliding window method)

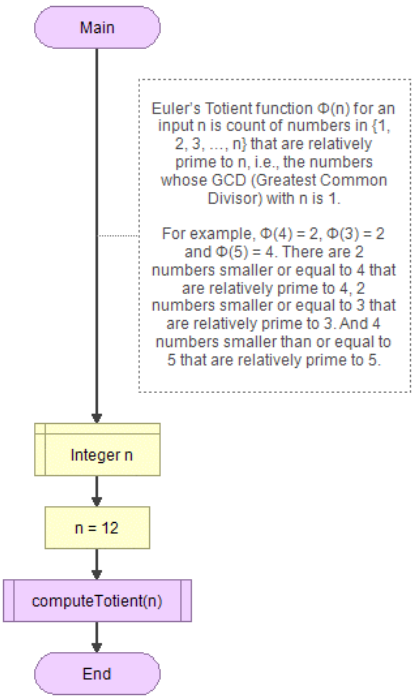


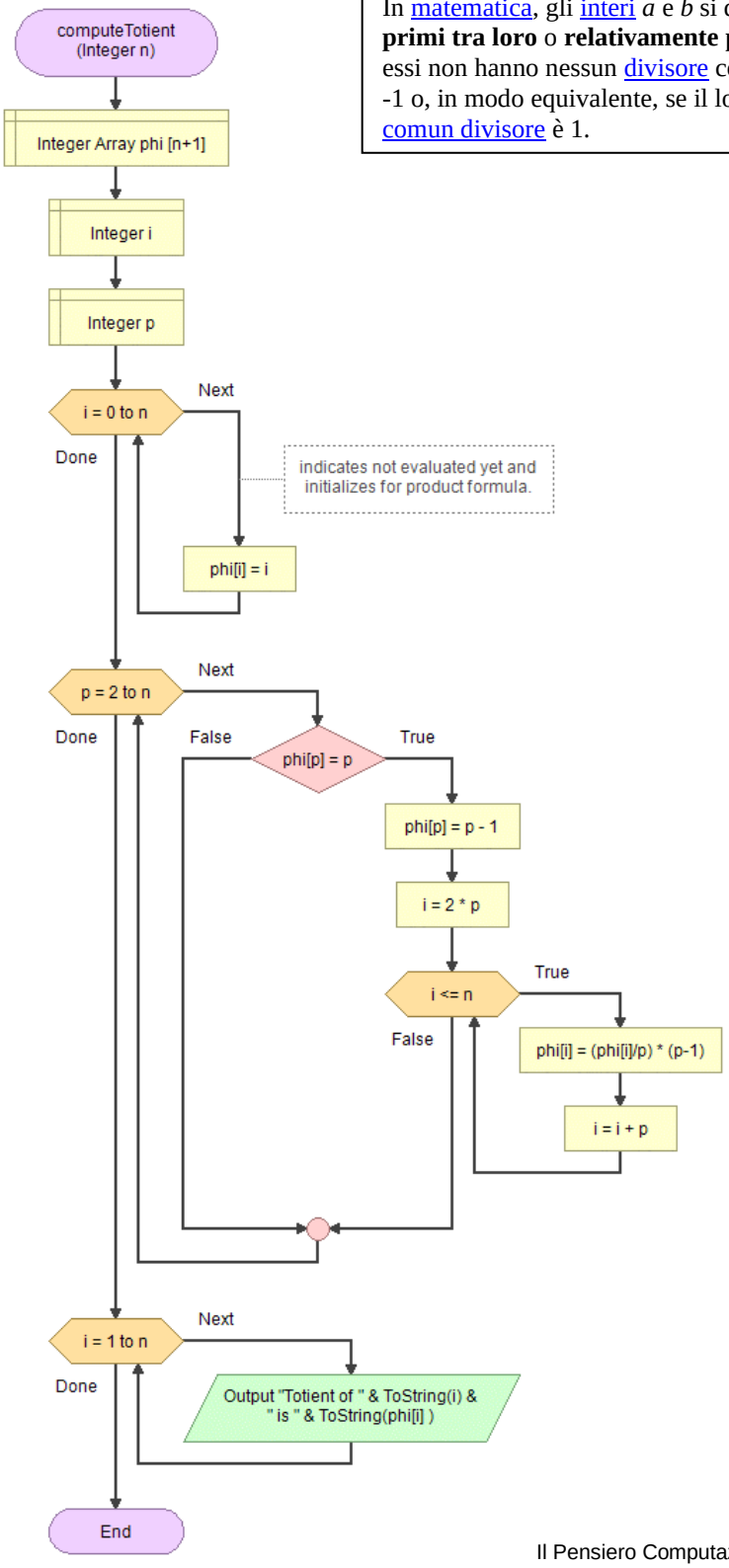
73. G F ‖ La funzione toziente di Eulero per tutti i numeri più piccoli o uguale a n

Euler's Totient function for all numbers smaller than or equal to n

In matematica, la **funzione φ di Eulero** o semplicemente **funzione di Eulero** o **toziente**, è una funzione definita, per ogni intero positivo n, come il numero degli interi compresi tra 1 e n che sono coprimi con n. Ad esempio, $\varphi(8) = 4$ poiché i numeri coprimi di 8 sono quattro: 1, 3, 5, 7.

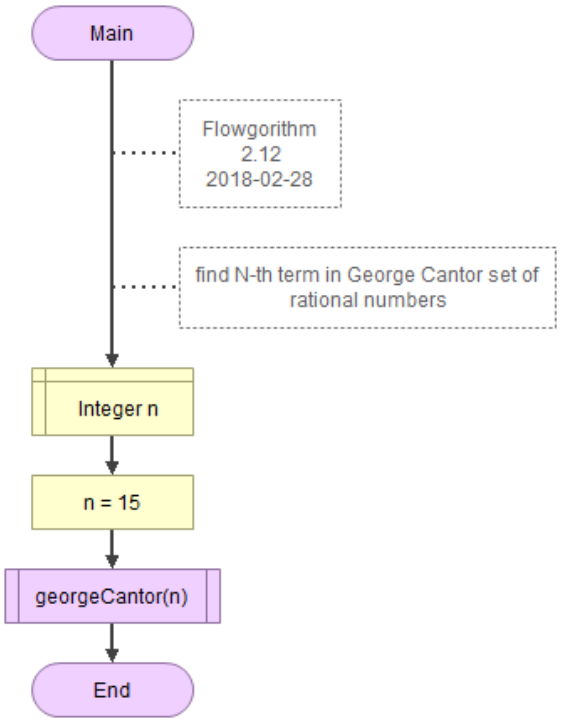
In matematica, gli interi *a* e *b* si dicono **coprimi** (o **primi tra loro** o **relativamente primi**) se e solo se essi non hanno nessun divisore comune eccetto 1 e -1 o, in modo equivalente, se il loro massimo comun divisore è 1.

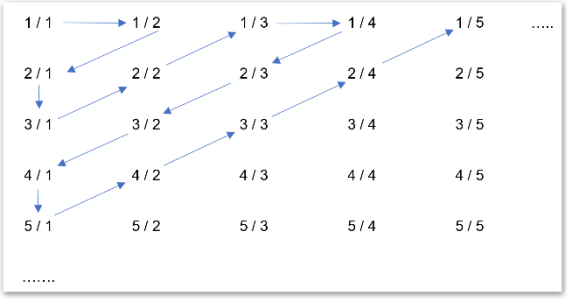


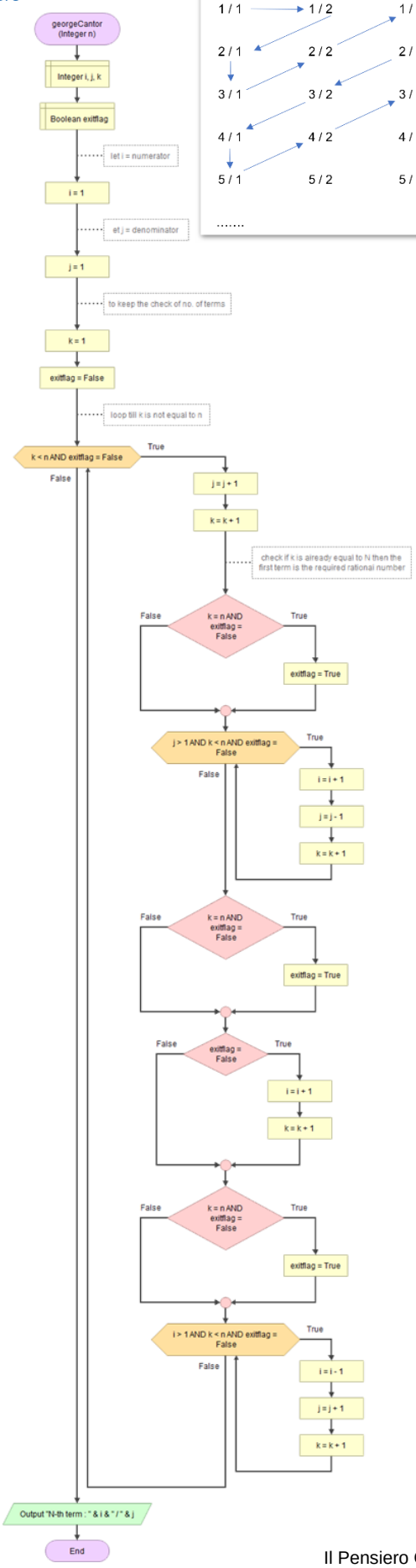


74. G F ‖ Ennesimo elemento di Cantor nell'insieme dei numeri razionali

N-th term of George Cantor set of rational numbers

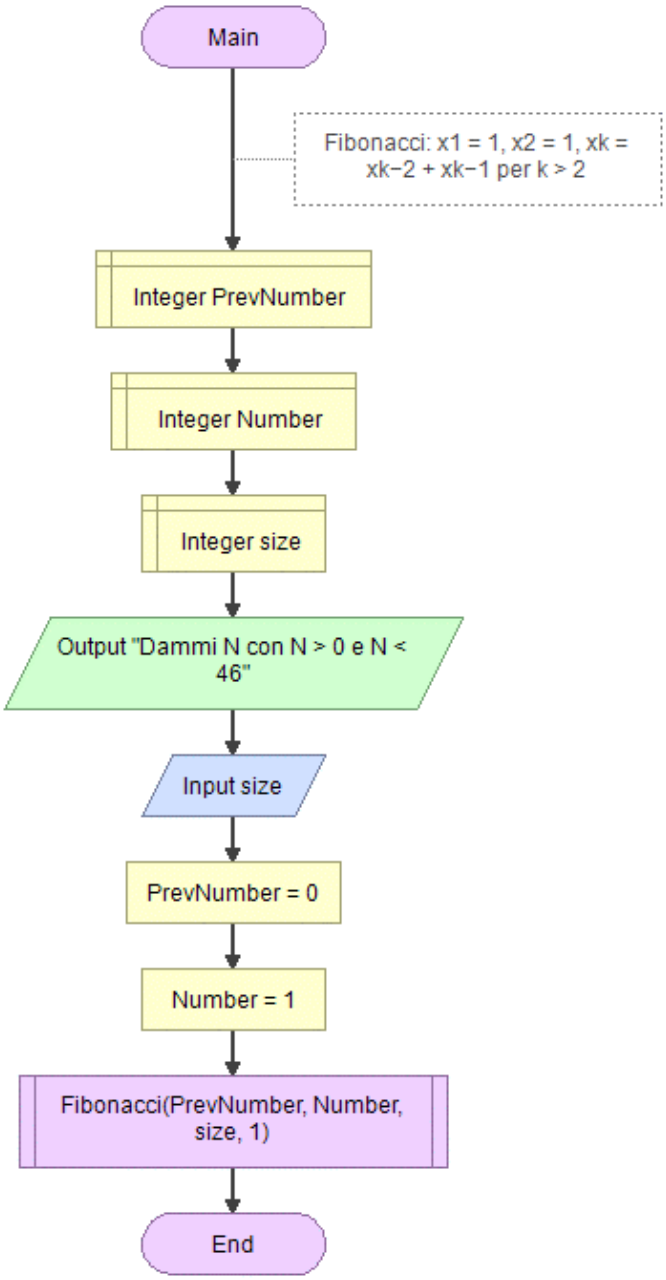


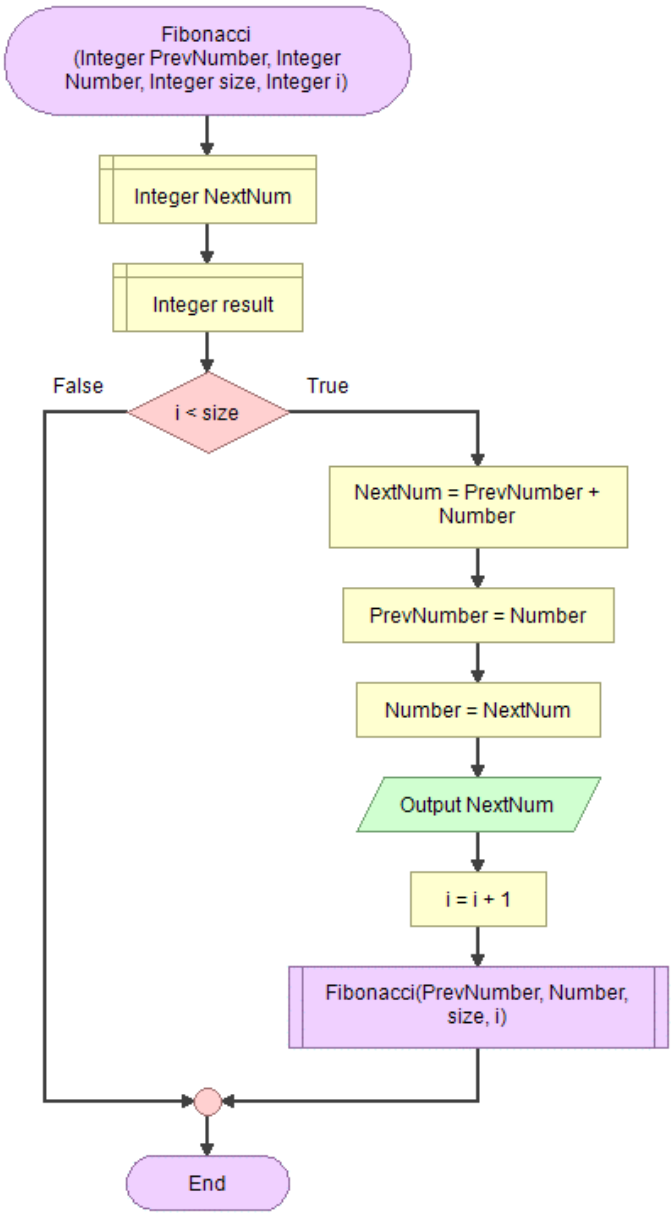




75. **G F R** ‖ Stampa i primi N numeri della successione di Fibonacci (ricorsivo)

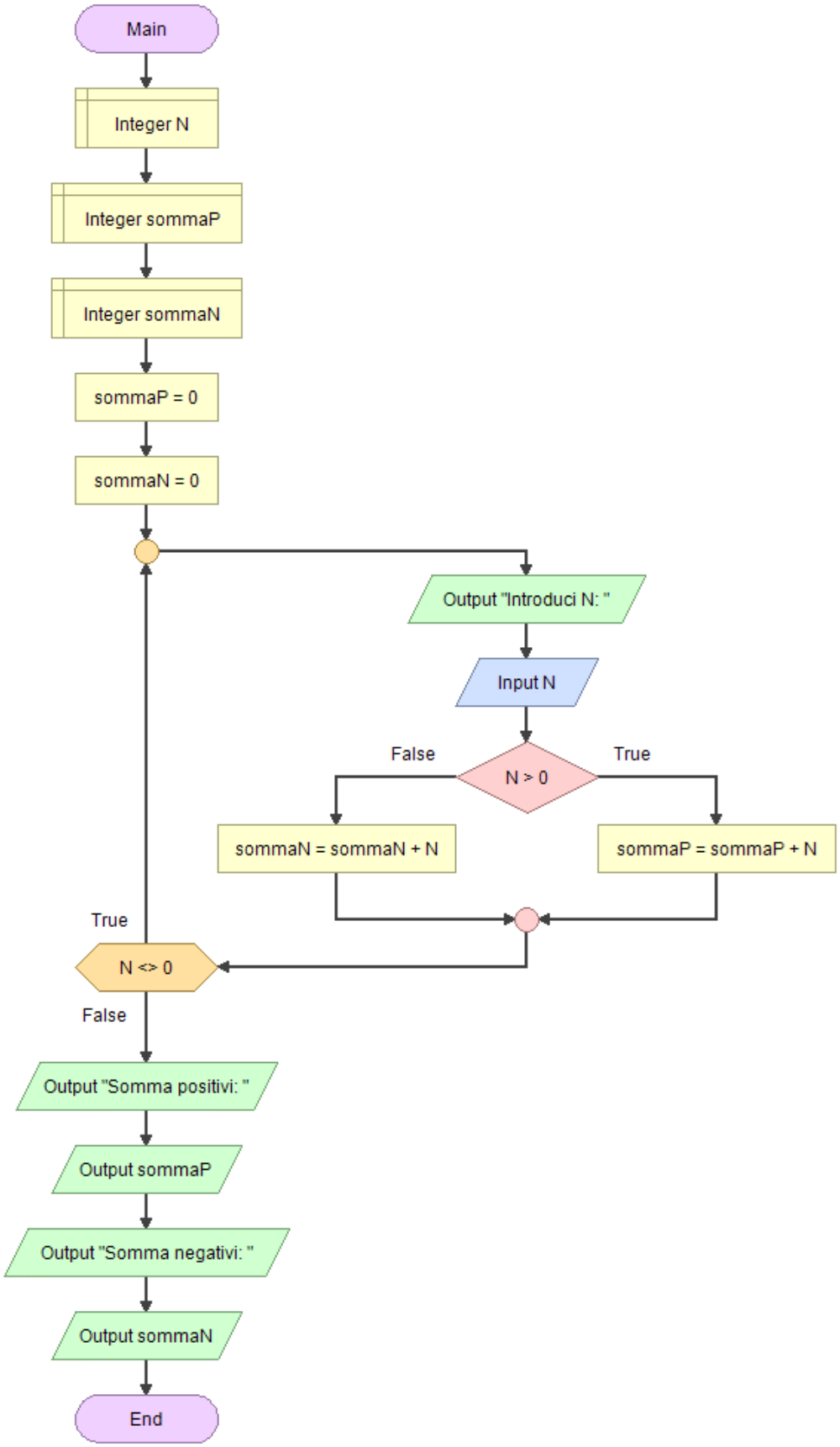
Display the first N numbers of the Fibonacci sequence (recursive)





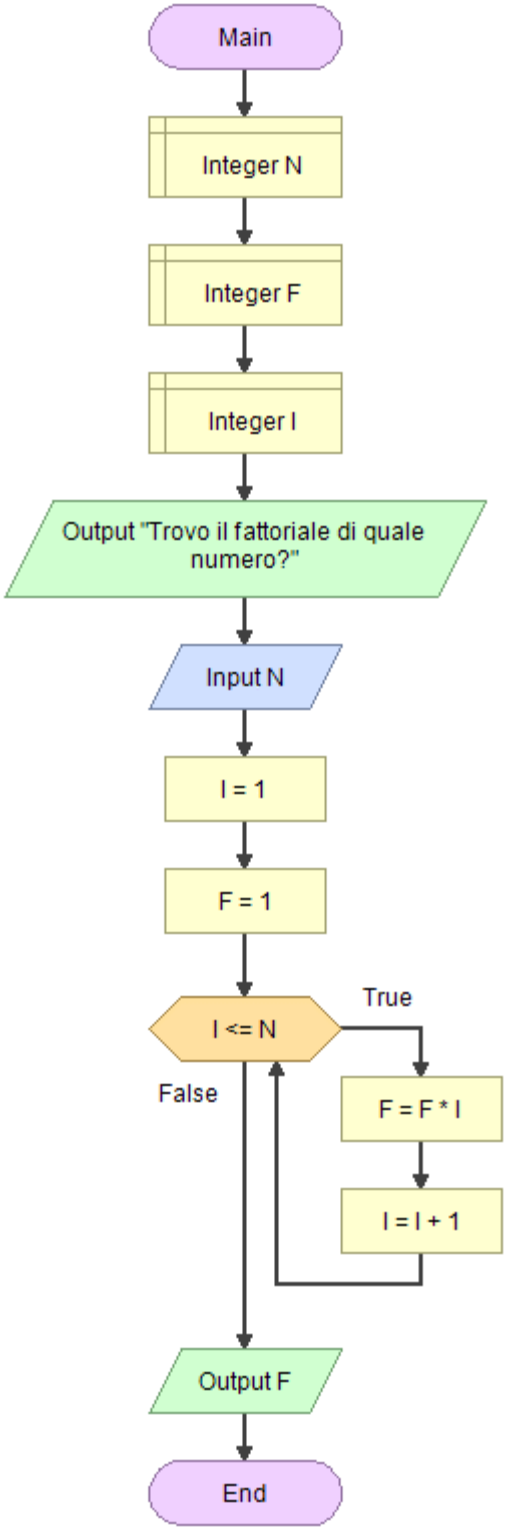
76. G ‖ Somma valori positivi e negativi

Sum of positive and negative values



77. G ‖ Letto un numero intero positivo ***n*** stampare il fattoriale: ***n***! = 1 · 2 · 3 · . . . ·***n***.

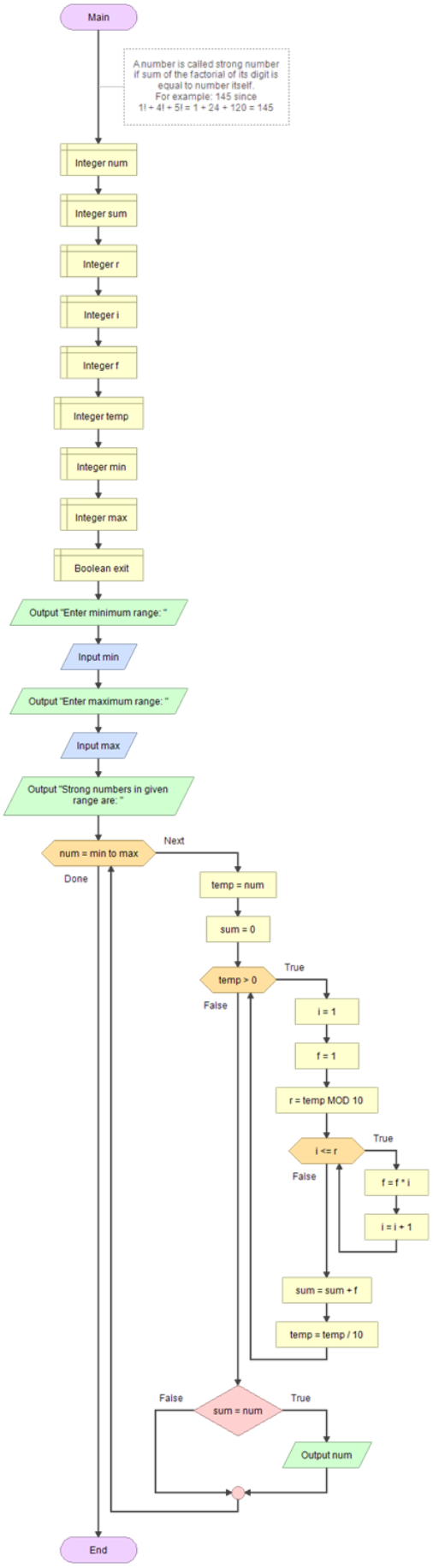
*Given a positive integer N find its factorial **n**! = 1 · 2 · 3 · . . . ·**n**.*



78. G ‖ Numero forte

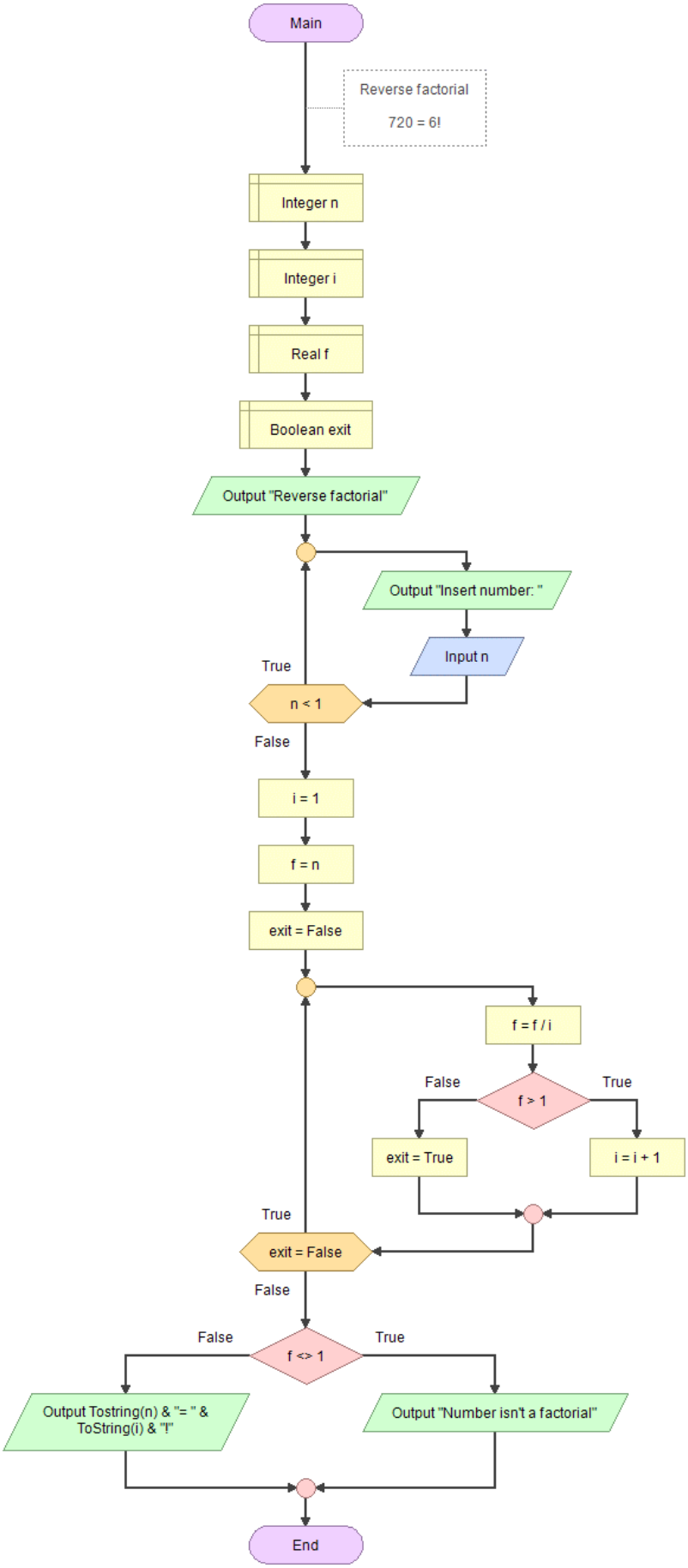
Strong number

145 è un numero forte perché
1! + 4! + 5! = 1 + 24 + 120 = 145



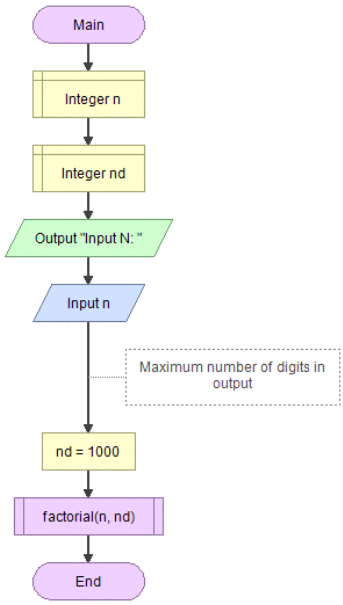
79. G ‖ Fattoriale inverso

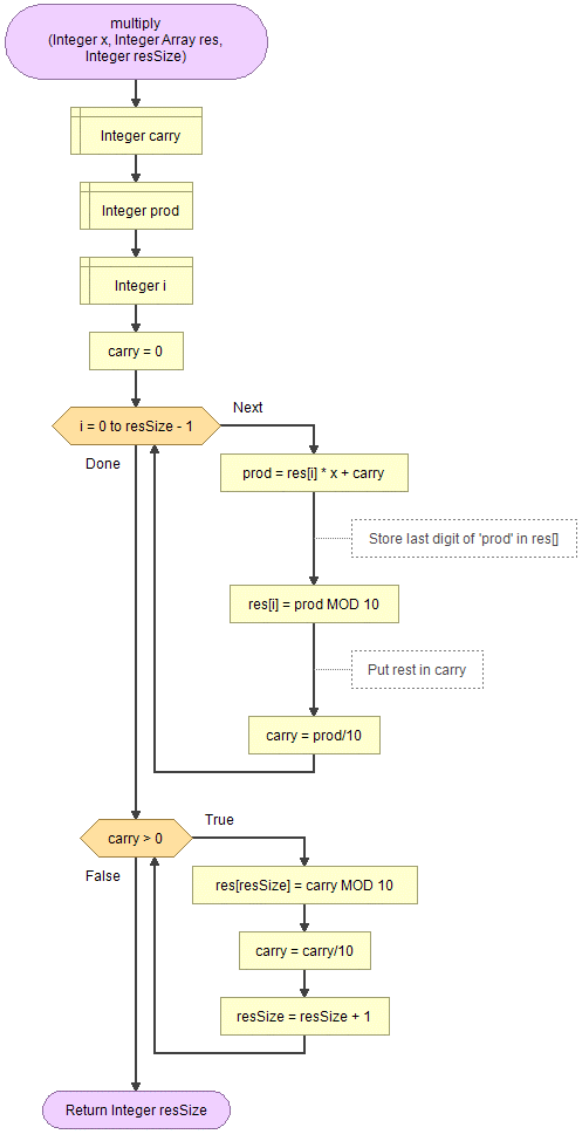
Reverse Factorial

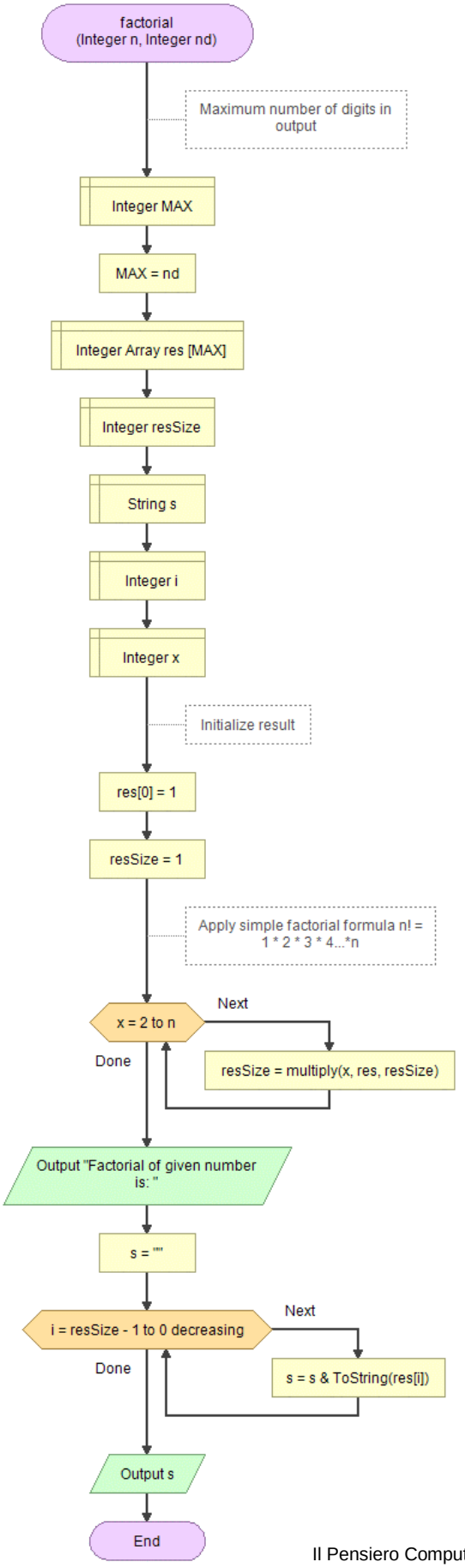


80. G F | Calcolo del fattoriale (visione di tutte le cifre)

Factorial of a large number (view all digits)

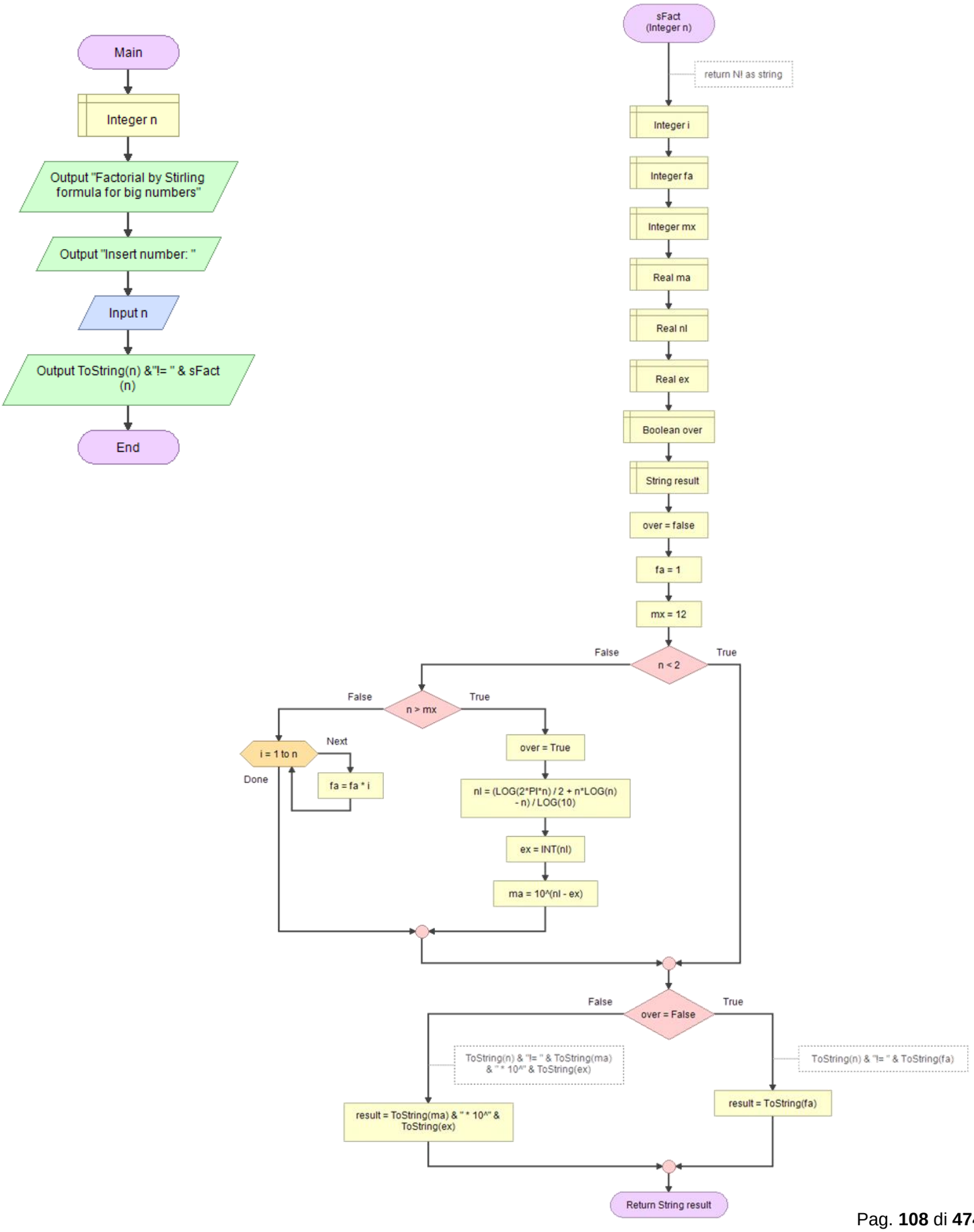




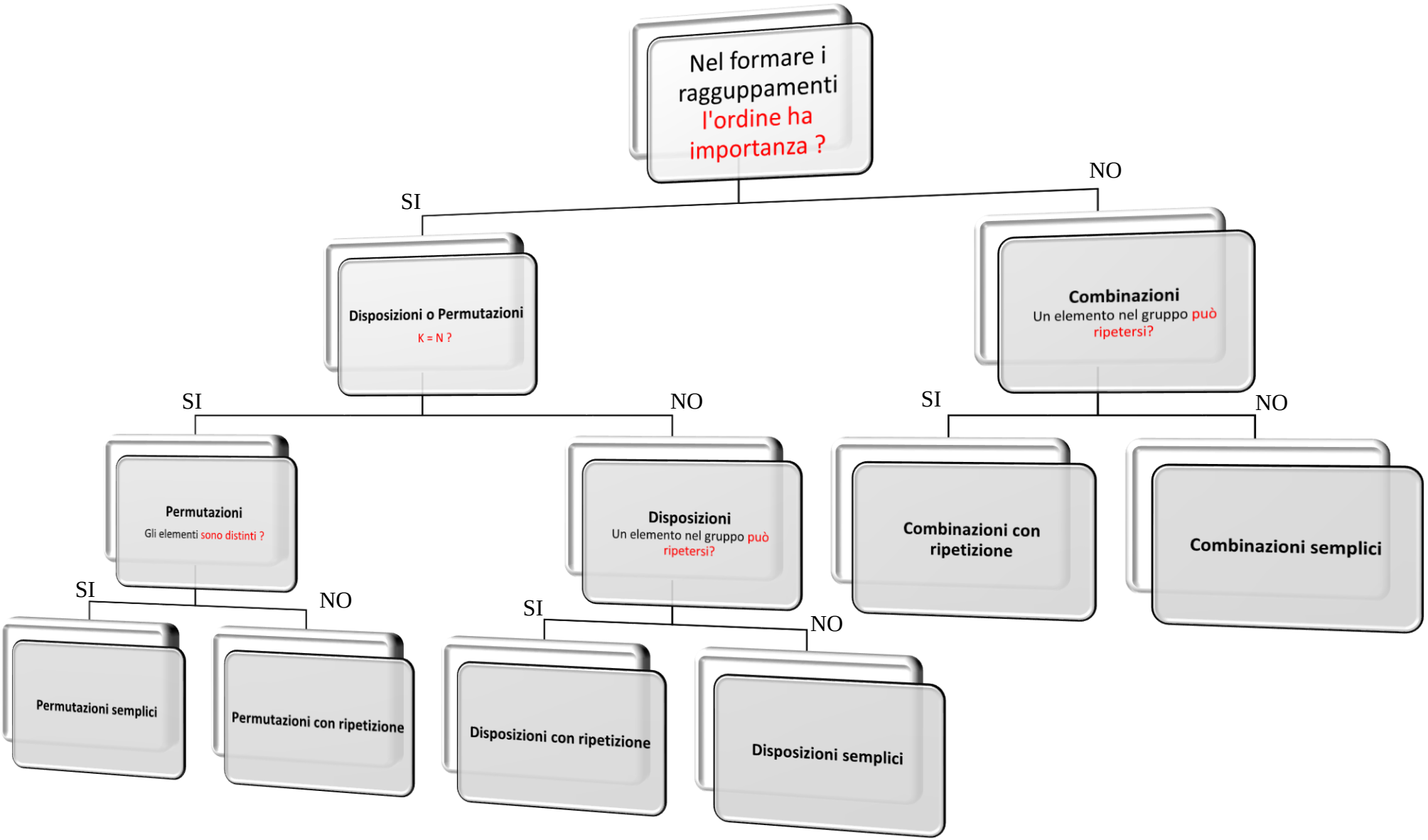


81. **G F** ‖ Calcolo del fattoriale utilizzando la formula di Stirling per i grandi numeri

Factorial by Stirling formula for big numbers

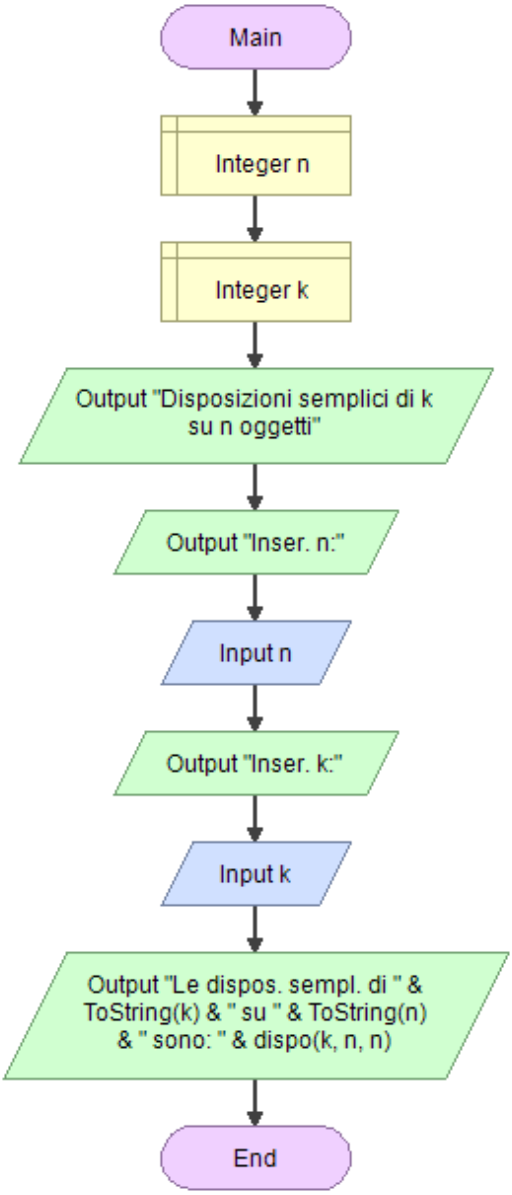


Quale formula usare?

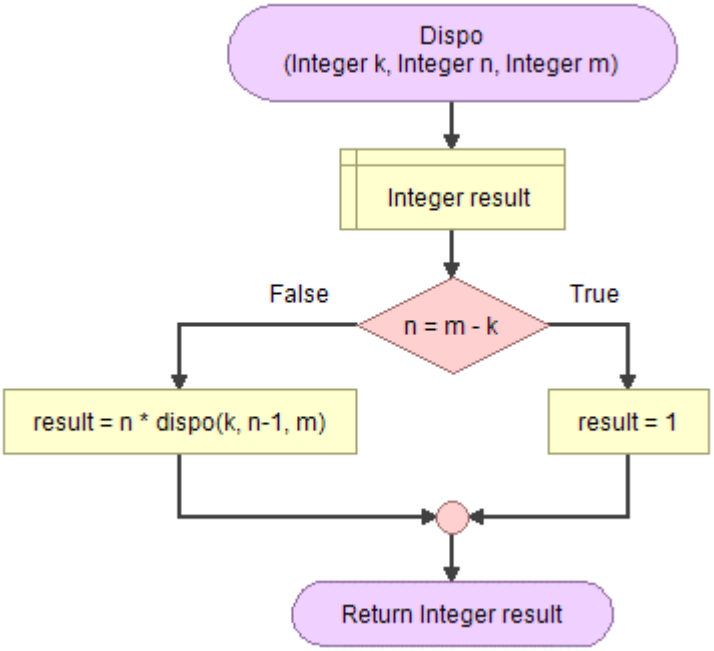


82. **G F R** ‖ Calcolo delle disposizioni semplici

Dispositions

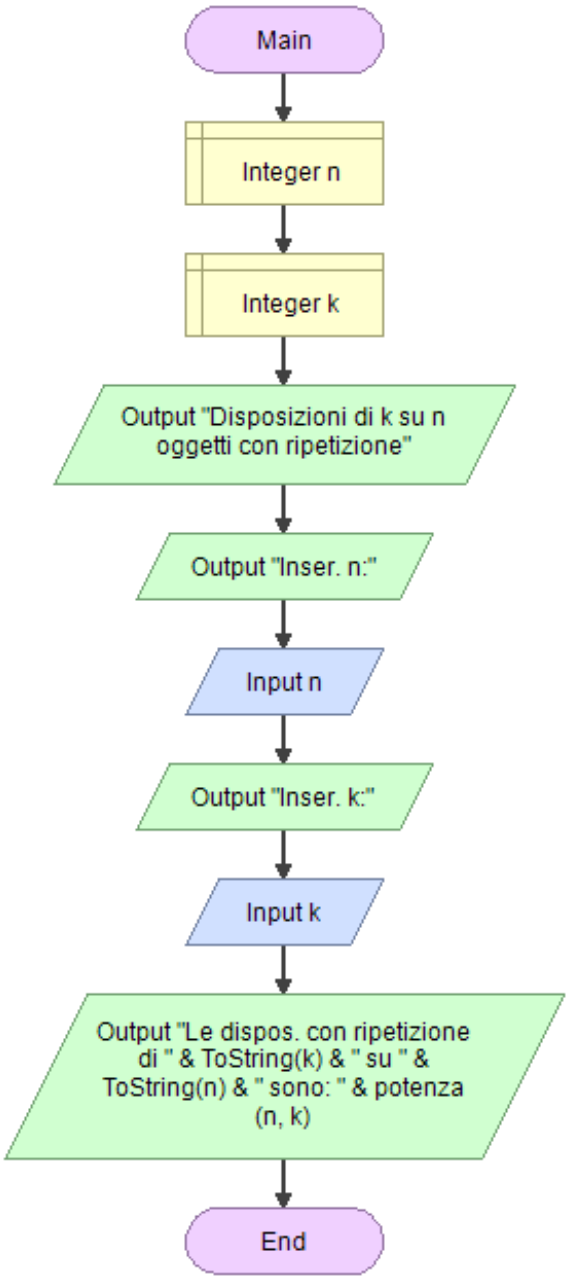


$$D_{n,k} = \frac{n!}{(n-k)!} = n \cdot (n-1) \cdot \ldots\ldots \cdot (n-k+1)$$

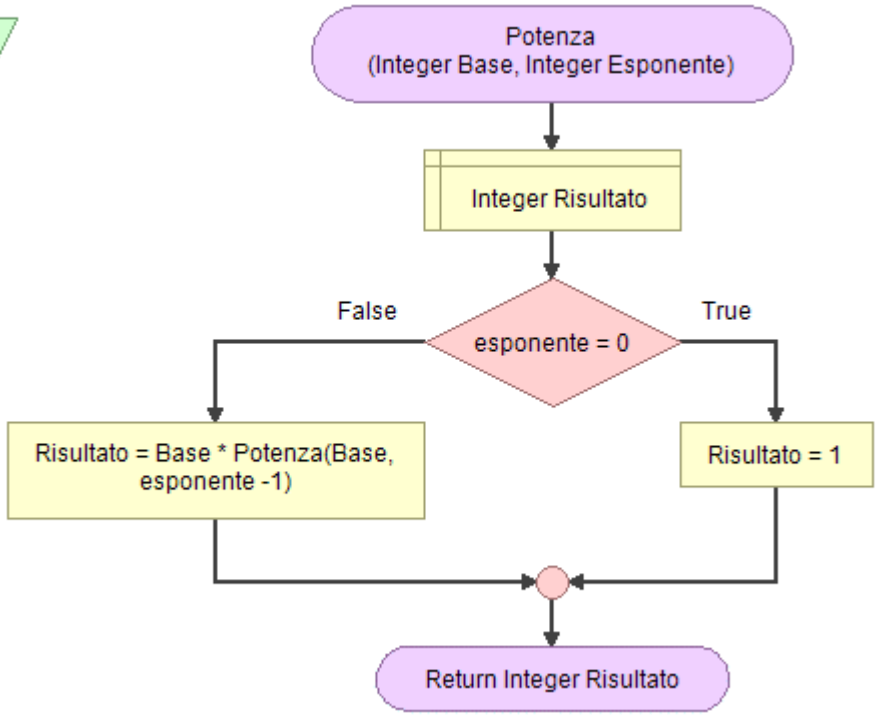


83. **G F R** | Calcolo delle disposizioni con ripetizione

Dispositions with repetitions

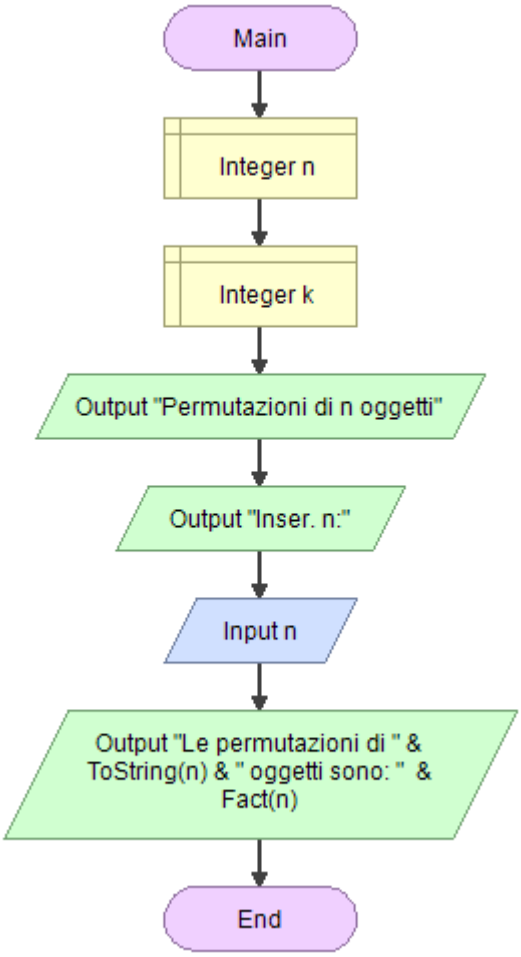


$$D'_{(n,k)} = n^k$$

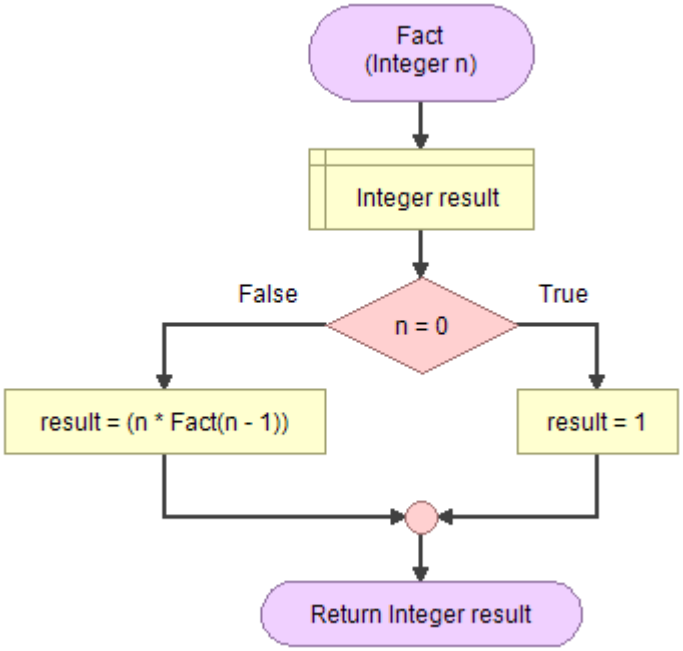


84. **G F R** | Calcolo delle permutazioni

Permutations

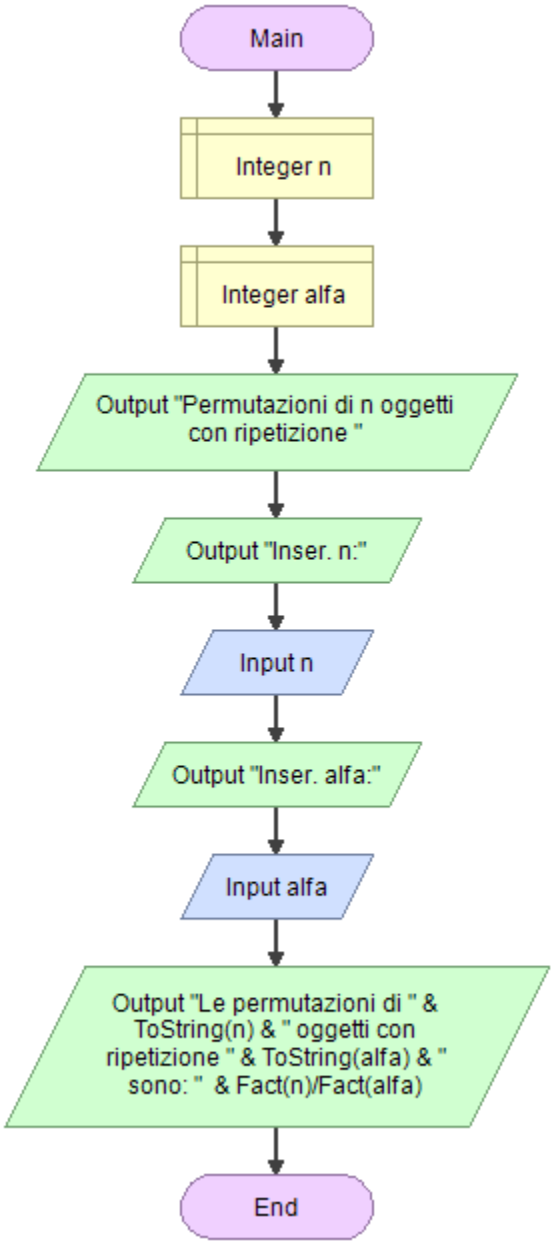


$$P_n = n!$$

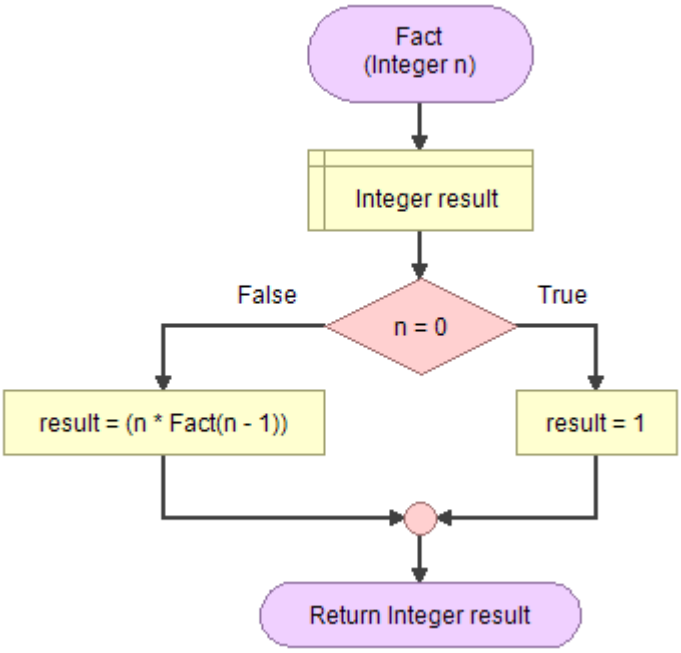


85. **G F R** | Calcolo delle permutazioni con ripetizione singola

Permutations with 1 repetition



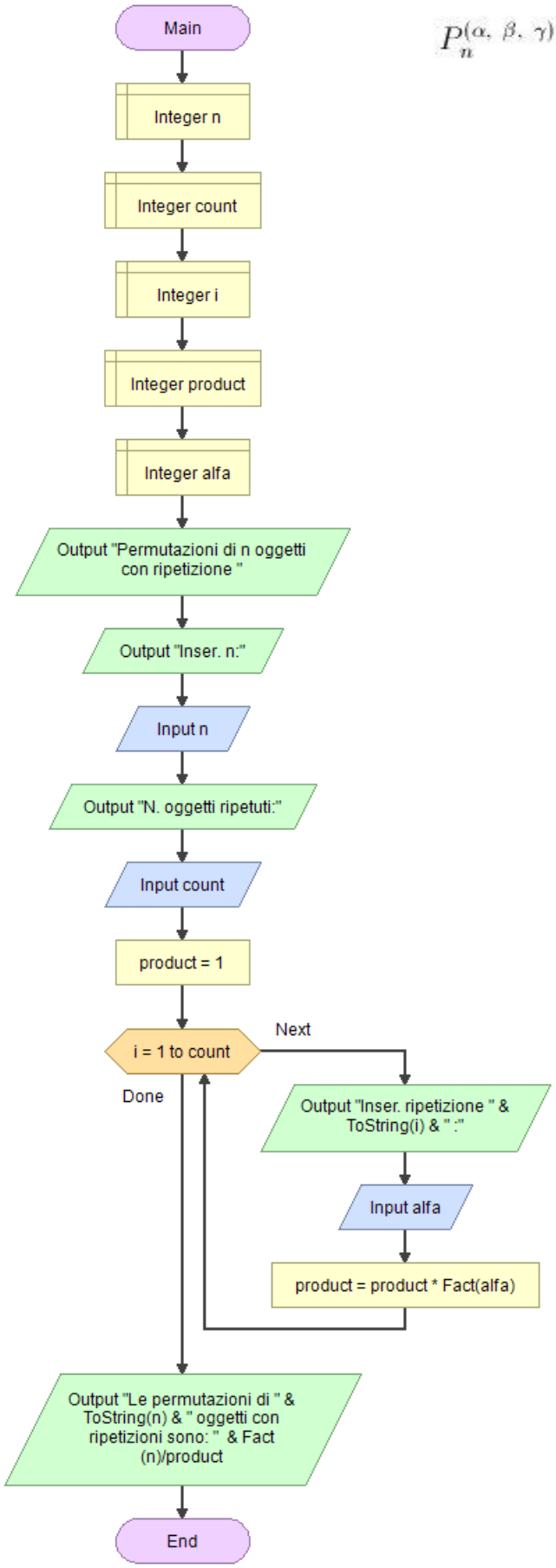
$$P_n^{(\alpha)} = \frac{n!}{\alpha!}$$

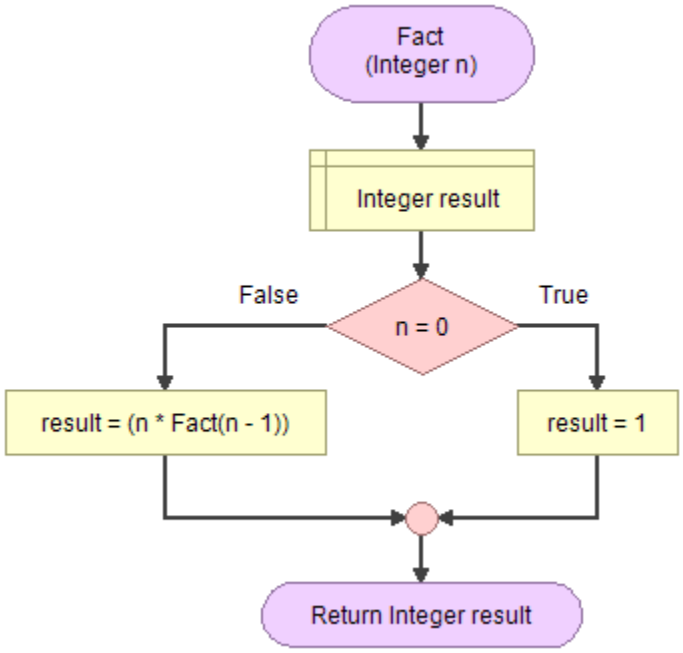


86. G F R ‖ Calcolo delle permutazioni con più ripetizioni

Permutations with repetitions

$$P_n^{(\alpha,\ \beta,\ \gamma)} = \frac{n!}{\alpha! \cdot \beta! \cdot \gamma!}$$

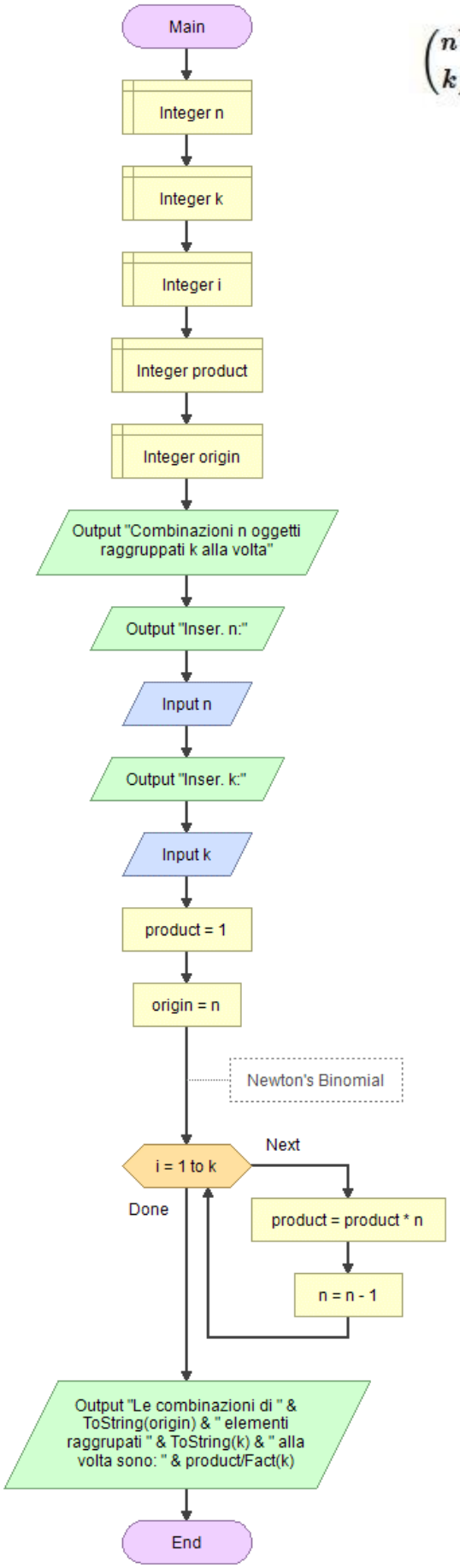


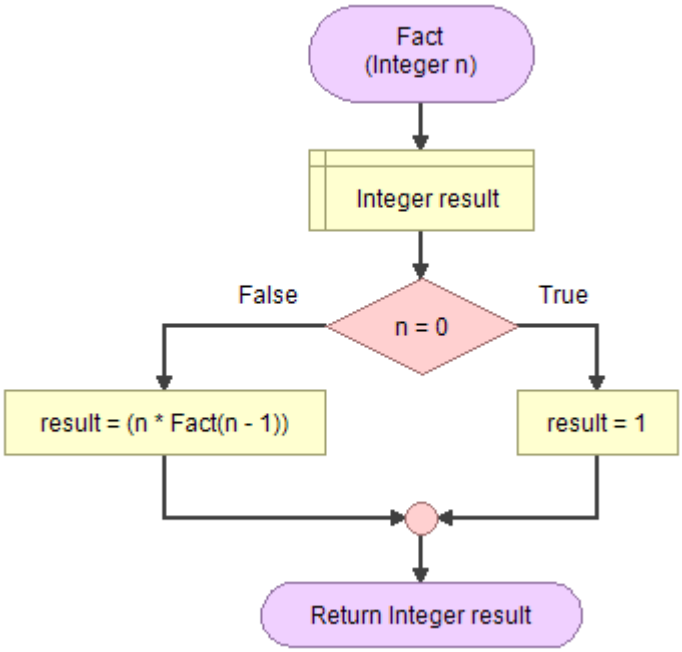


87. GFR | Calcolo delle combinazioni semplici

Combinations

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} = \frac{n(n-1)(n-2)\cdots(n-k+1)}{k!}.$$

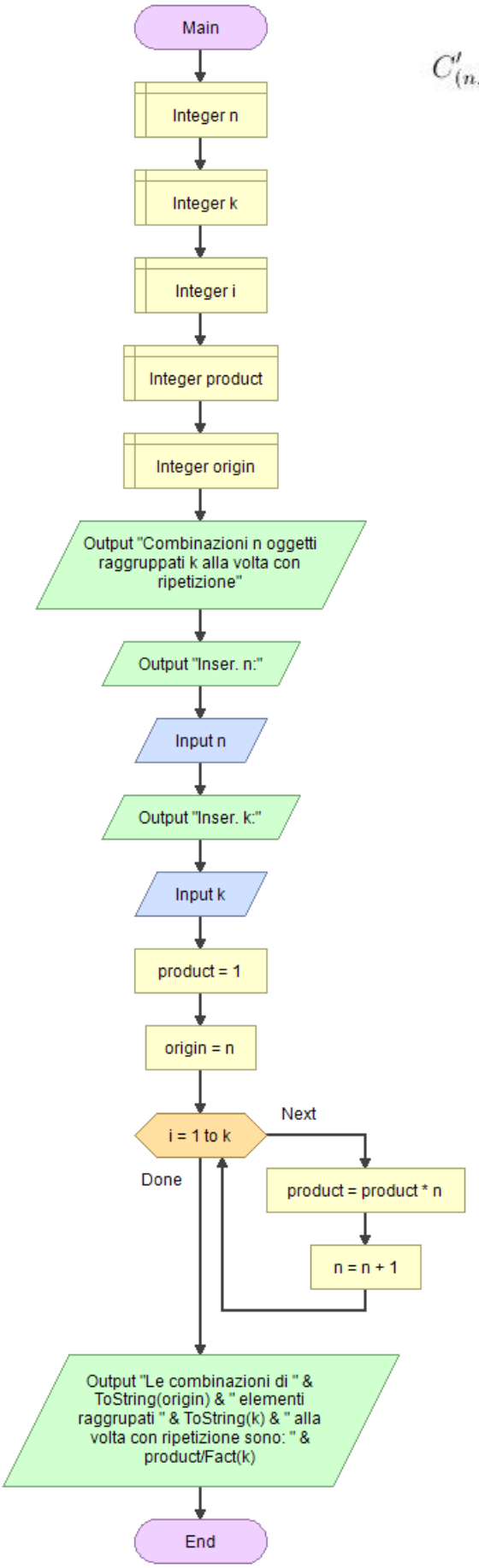


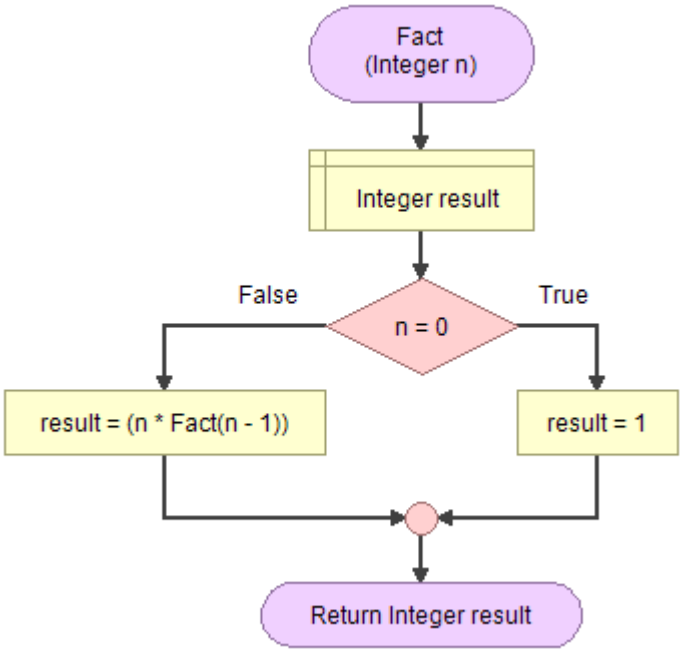


88. GFR | Calcolo delle combinazioni con ripetizioni

Combinations with repetitions

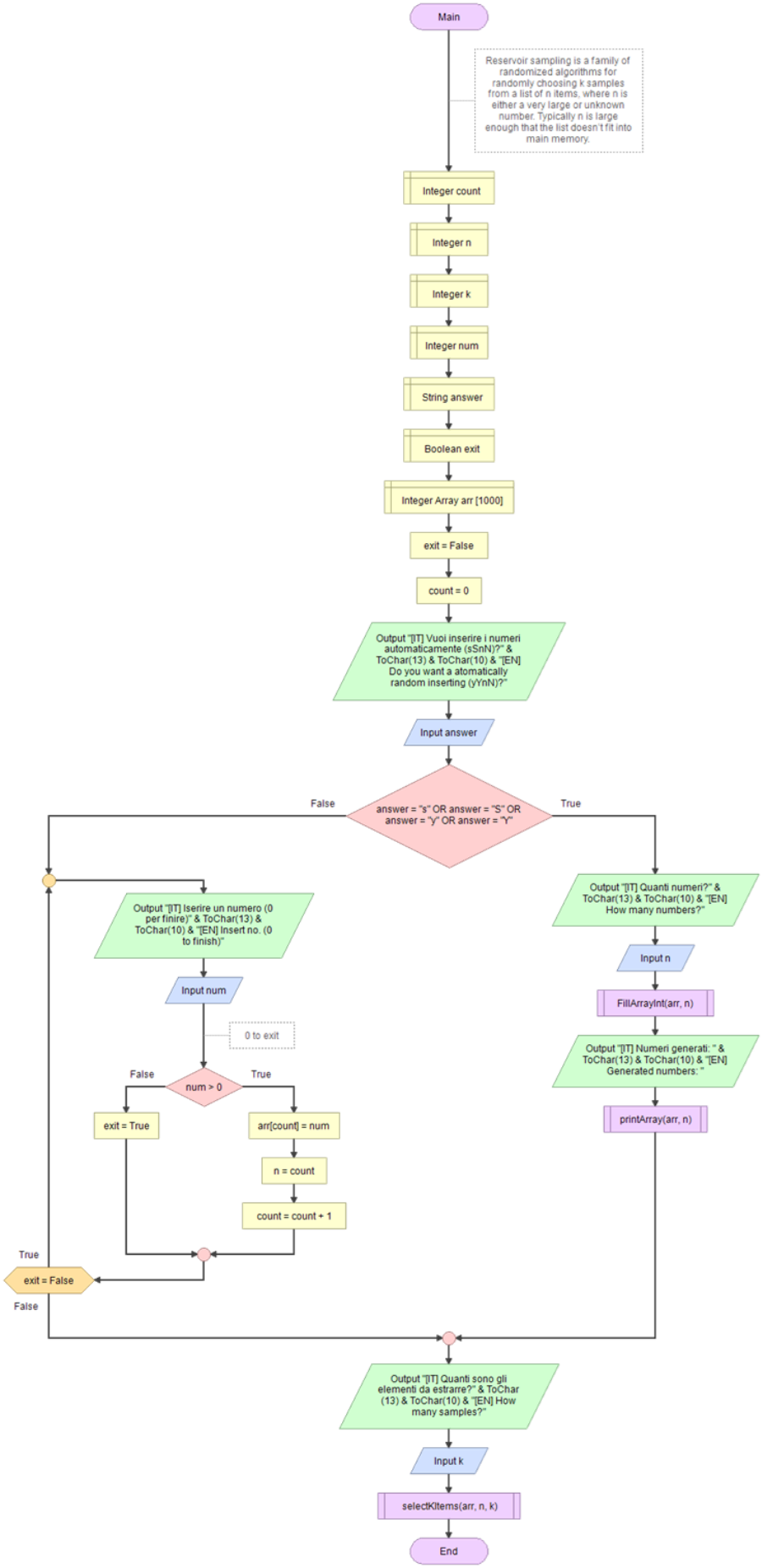
$$C'_{(n,k)} = \binom{n+k-1}{k} = \frac{(n+k-1)!}{k!(n-1)!}$$

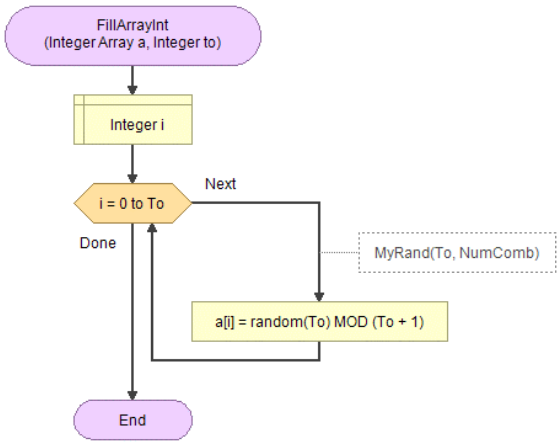


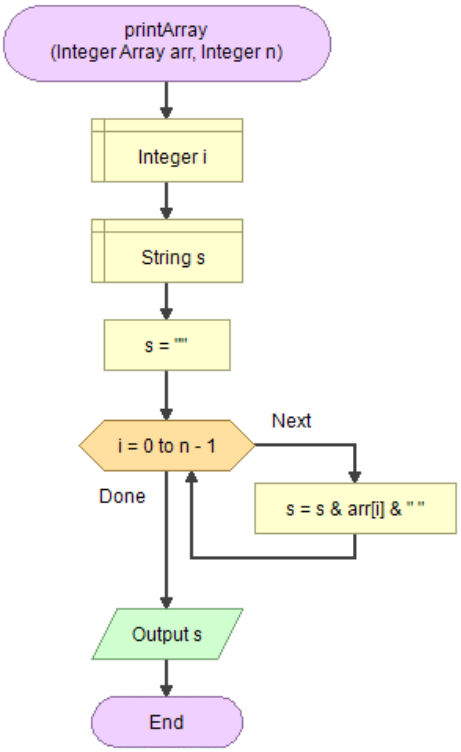


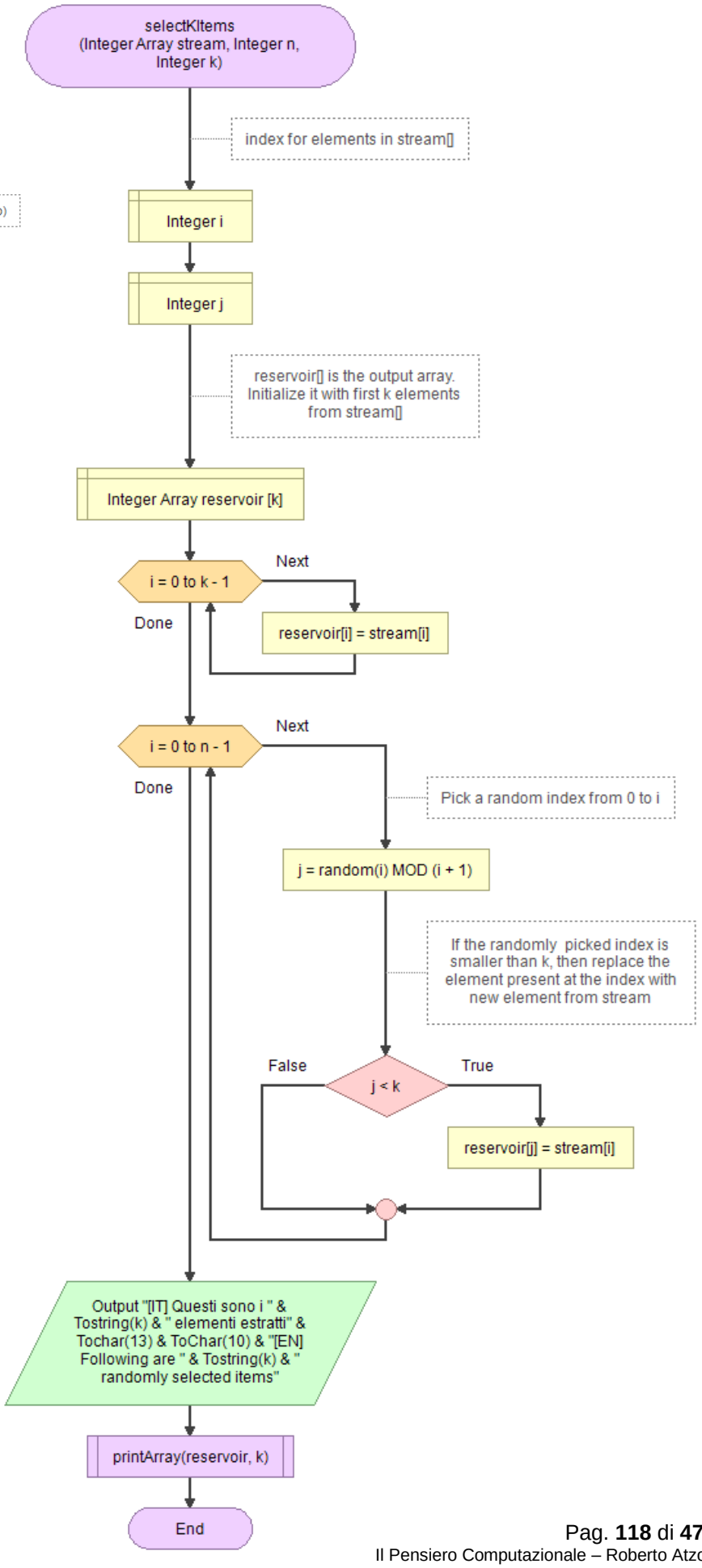
89. G ‖ Campionamento Reservoir: scegliere k gruppi da una lista di n elementi

Reservoir Sampling - choosing k samples from a list of n items



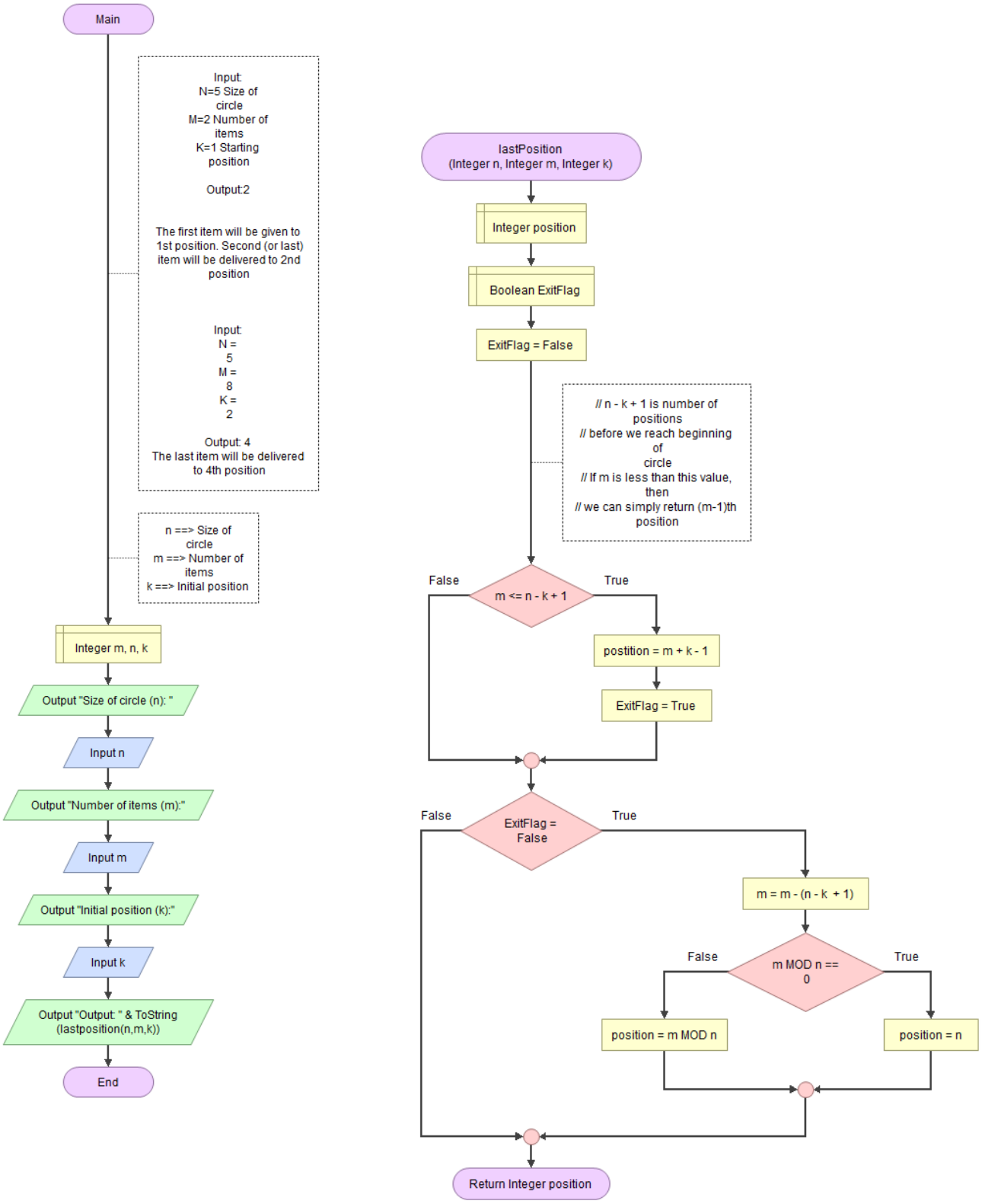

FillArrayInt
(Integer Array a, Integer to)
Integer i
i = 0 to To
Next
Done
MyRand(To, NumComb)
a[i] = random(To) MOD (To + 1)
End


printArray
(Integer Array arr, Integer n)
Integer i
String s
s = ""
i = 0 to n - 1
Next
Done
s = s & arr[i] & " "
Output s
End


selectKItems
(Integer Array stream, Integer n, Integer k)
index for elements in stream[]
Integer i
Integer j
reservoir[] is the output array. Initialize it with first k elements from stream[]
Integer Array reservoir [k]
i = 0 to k - 1
Next
Done
reservoir[i] = stream[i]
i = 0 to n - 1
Next
Done
Pick a random index from 0 to i
j = random(i) MOD (i + 1)
If the randomly picked index is smaller than k, then replace the element present at the index with new element from stream
False
j < k
True
reservoir[j] = stream[i]
Output "[IT] Questi sono i " & Tostring(k) & " elementi estratti" & Tochar(13) & ToChar(10) & "[EN] Following are " & Tostring(k) & " randomly selected items"
printArray(reservoir, k)
End

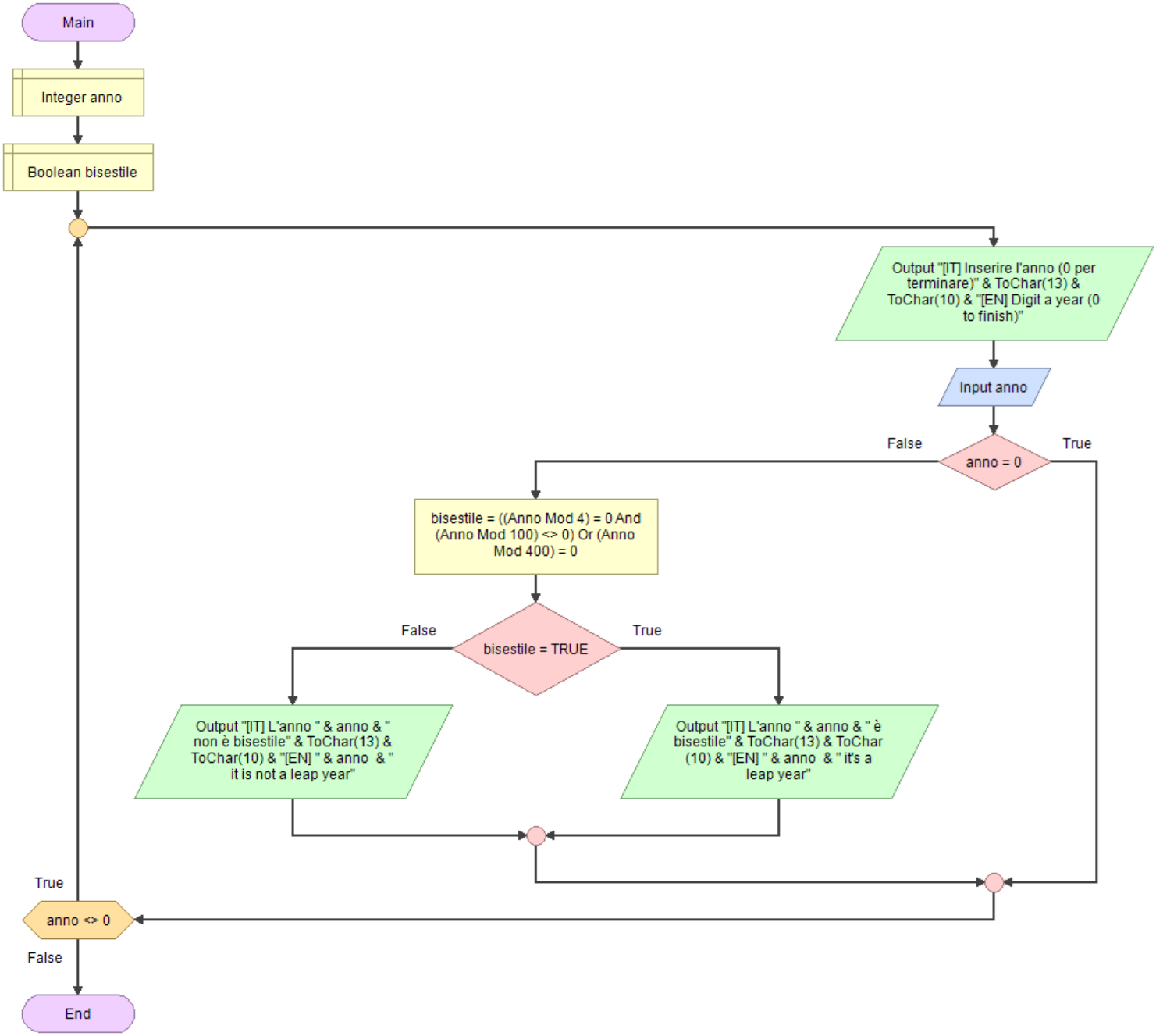
90. G F ‖ Distribuire M elementi in un cerchio di grandezza N partendo dalla k-esima posizione

Distributing M items in a circle of size N starting from K-th position



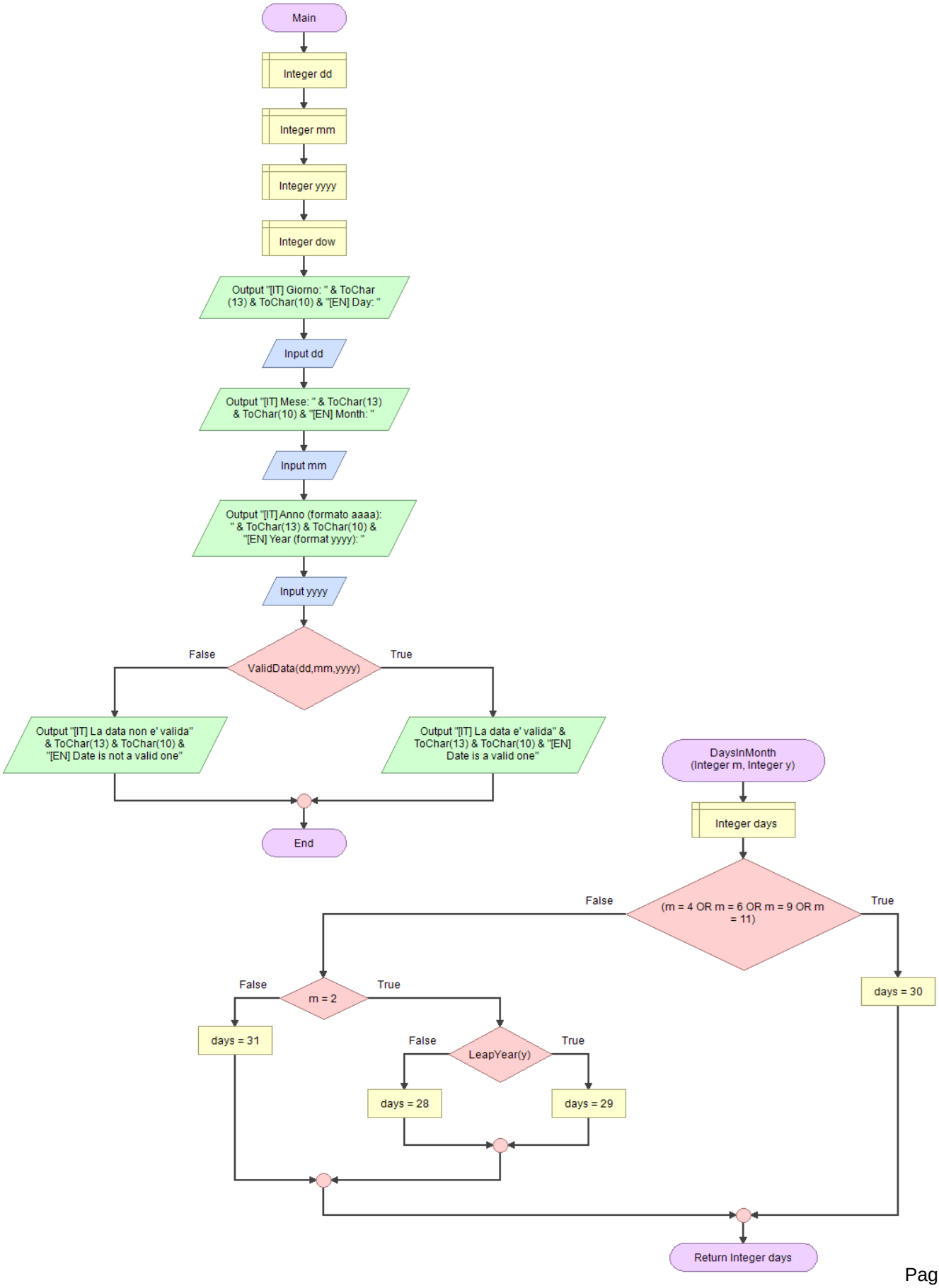
91. G ‖ Calcolare se un anno è bisestile o meno.

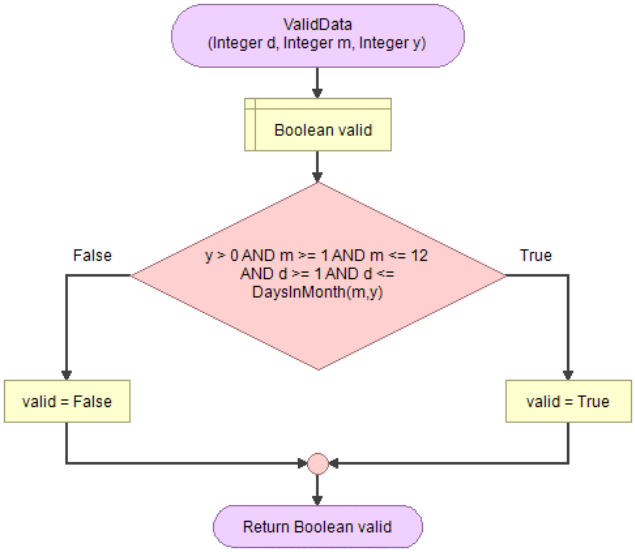
Leap year calculus

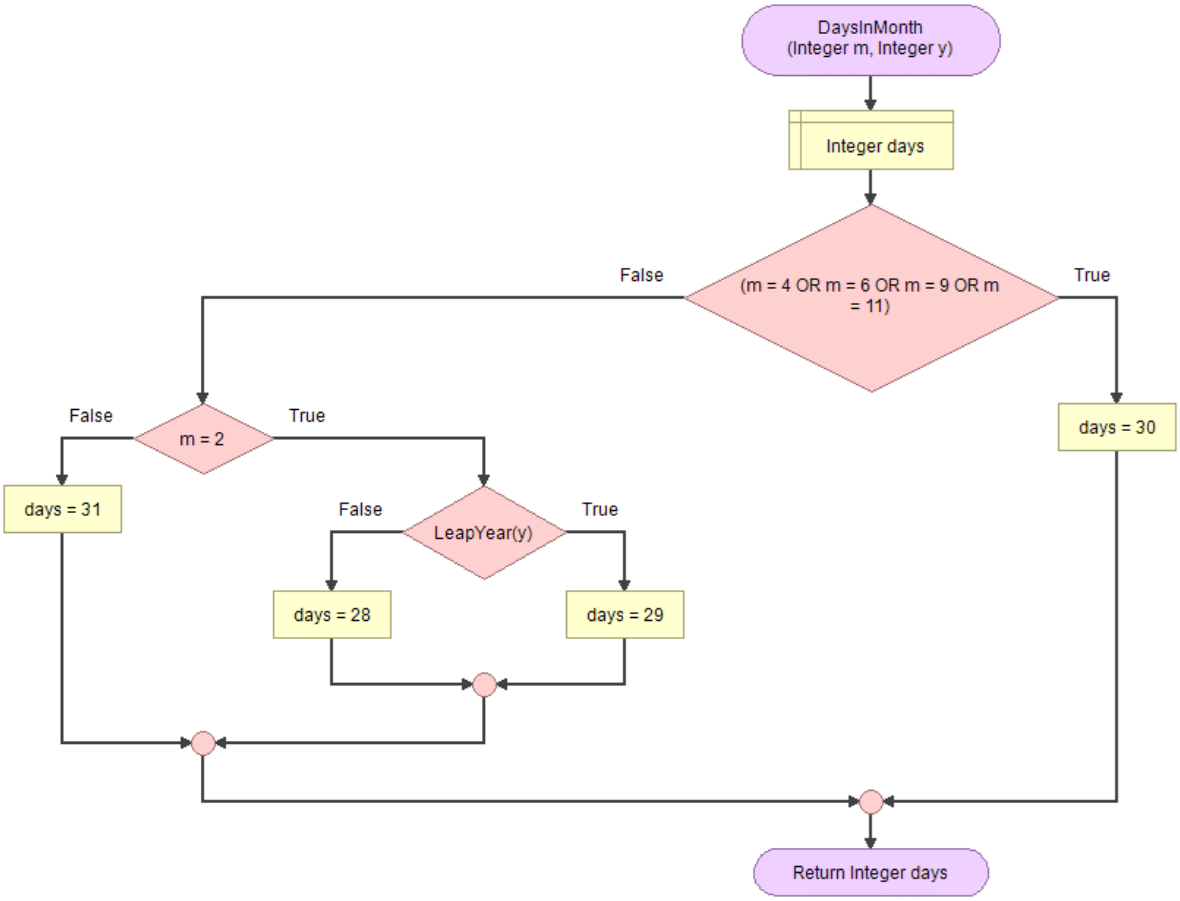


92. G F ‖ Controllo validità data

Right date control

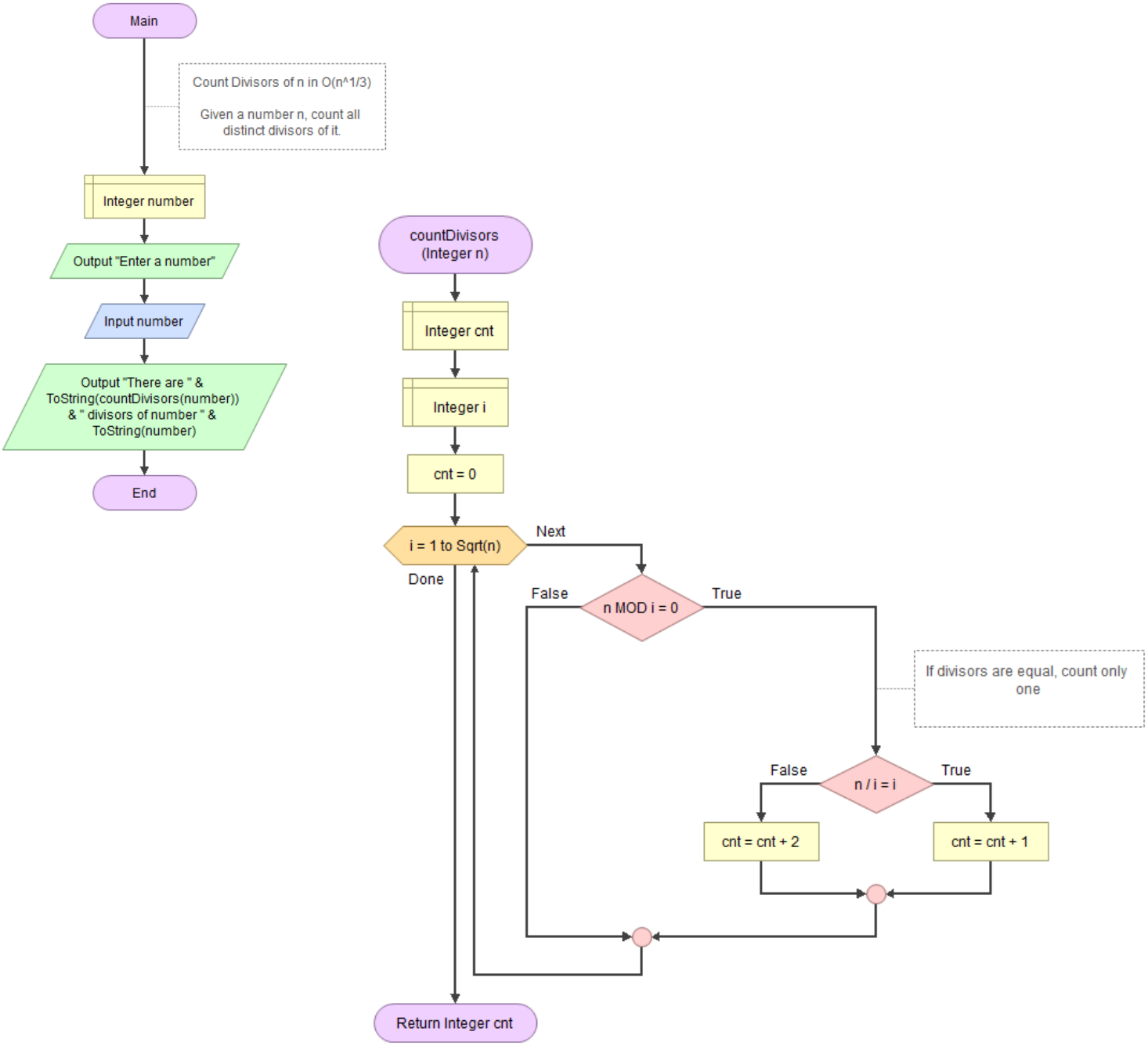



ValidData
(Integer d, Integer m, Integer y)
Boolean valid
y > 0 AND m >= 1 AND m <= 12 AND d >= 1 AND d <= DaysInMonth(m,y)
False
True
valid = False
valid = True
Return Boolean valid


DaysInMonth
(Integer m, Integer y)
Integer days
(m = 4 OR m = 6 OR m = 9 OR m = 11)
False
True
days = 30
m = 2
False
True
days = 31
LeapYear(y)
False
True
days = 28
days = 29
Return Integer days

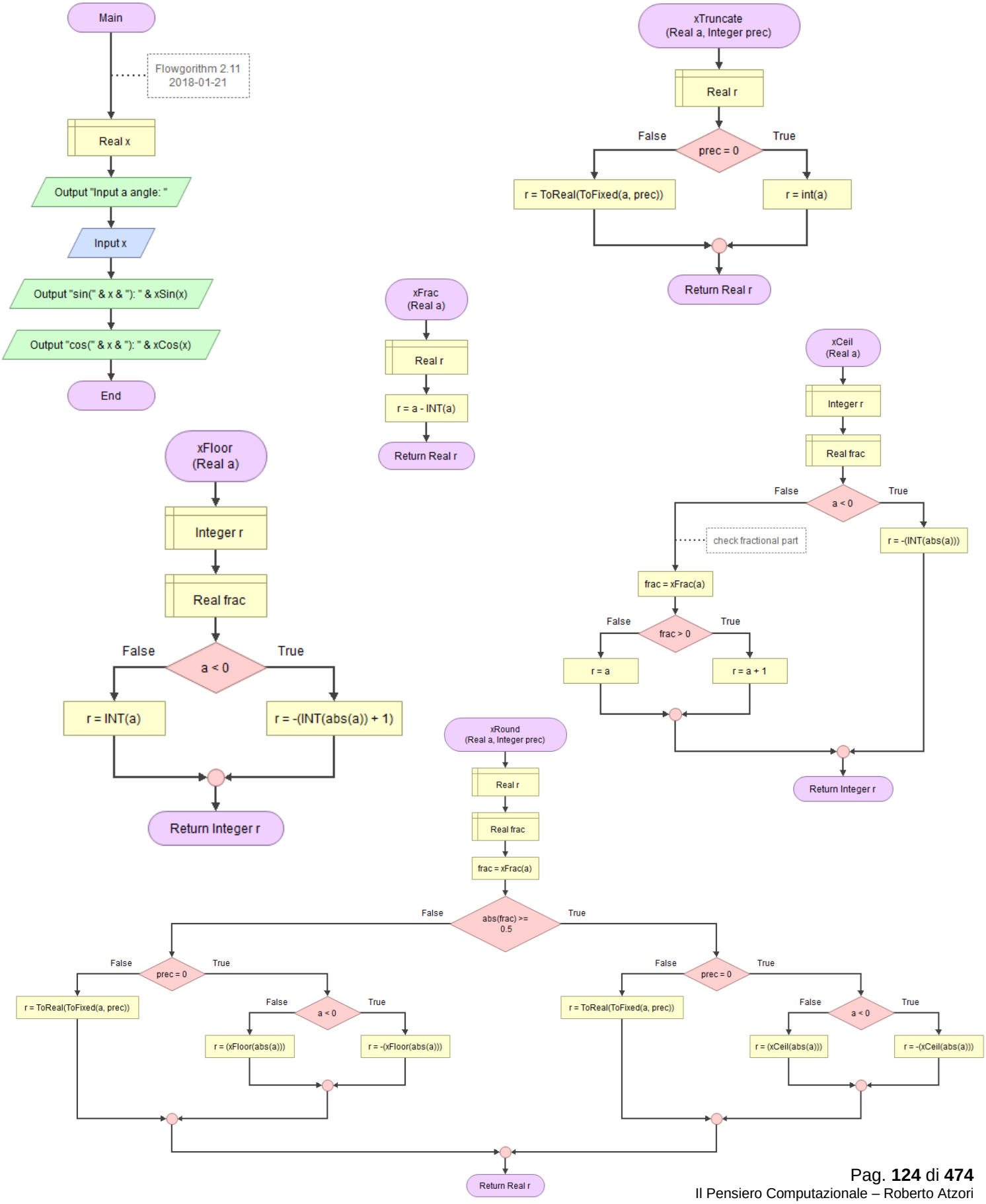
93. **G F** ‖ Dato un numero N intero contare quanti sono i suoi divisori

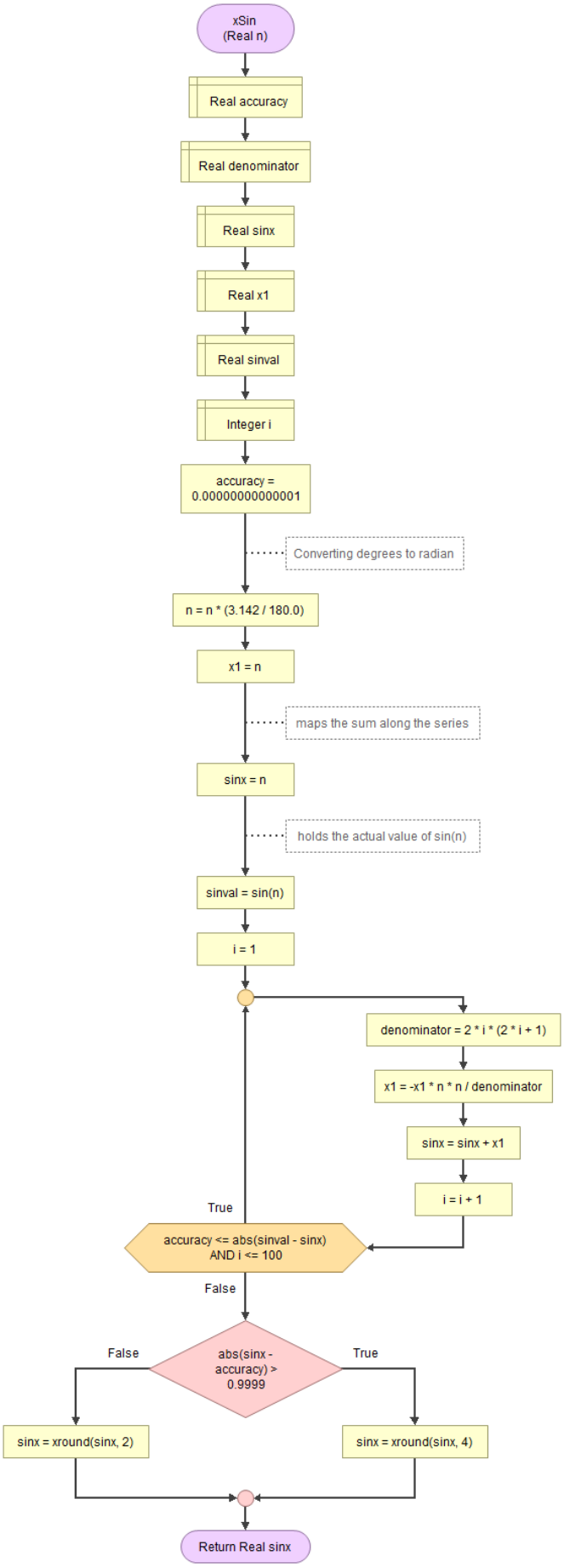
Count Divisors of n

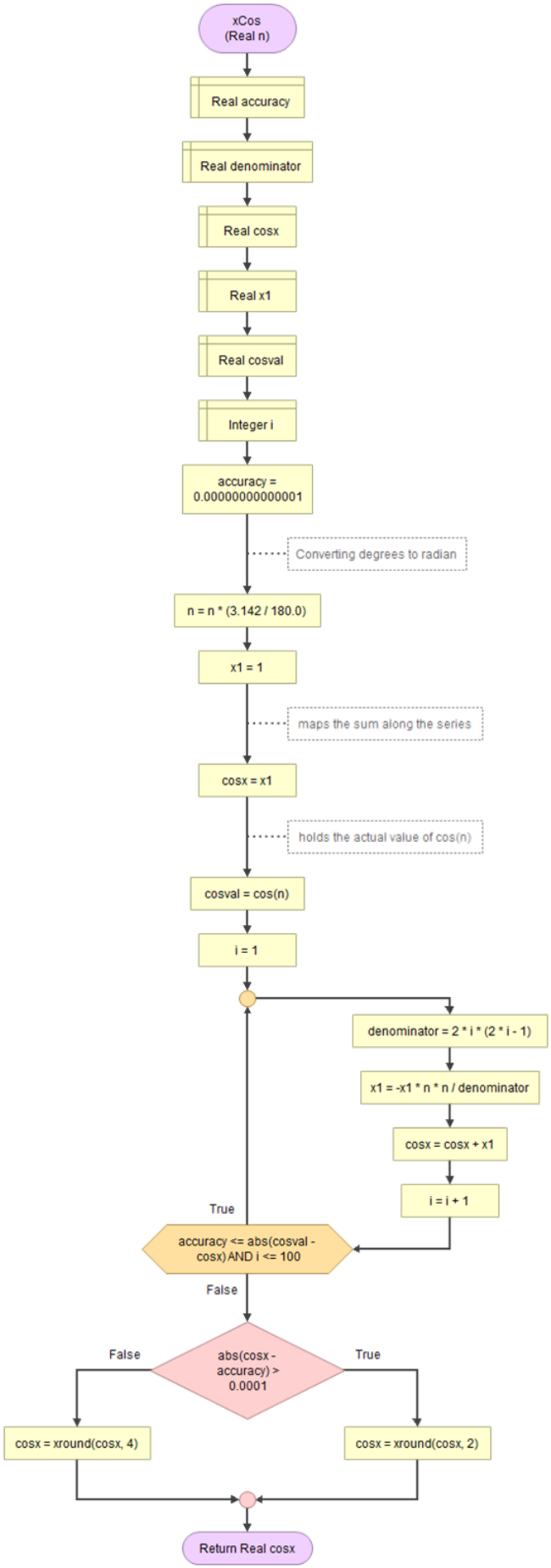


94. G F ‖ Calcolo del valore seno e del coseno di un angolo

Calculus of SIN e COS vale of a angle

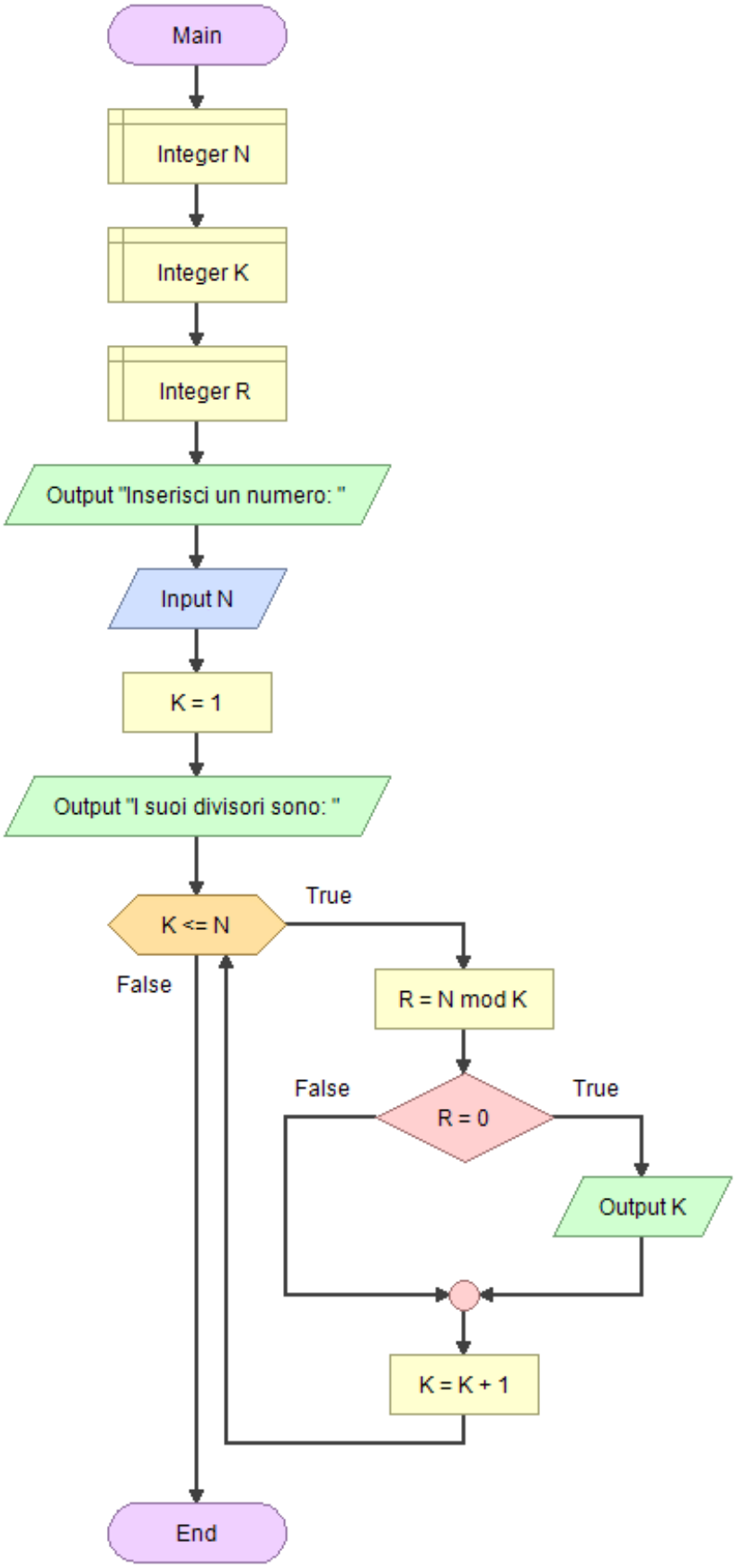



xSin
(Real n)
Real accuracy
Real denominator
Real sinx
Real x1
Real sinval
Integer i
accuracy =
0.00000000000001
Converting degrees to radian
n = n * (3.142 / 180.0)
x1 = n
maps the sum along the series
sinx = n
holds the actual value of sin(n)
sinval = sin(n)
i = 1
denominator = 2 * i * (2 * i + 1)
x1 = -x1 * n * n / denominator
sinx = sinx + x1
i = i + 1
True
accuracy <= abs(sinval - sinx)
AND i <= 100
False
False
abs(sinx -
accuracy) >
0.9999
True
sinx = xround(sinx, 2)
sinx = xround(sinx, 4)
Return Real sinx


xCos
(Real n)
Real accuracy
Real denominator
Real cosx
Real x1
Real cosval
Integer i
accuracy =
0.00000000000001
Converting degrees to radian
n = n * (3.142 / 180.0)
x1 = 1
maps the sum along the series
cosx = x1
holds the actual value of cos(n)
cosval = cos(n)
i = 1
denominator = 2 * i * (2 * i - 1)
x1 = -x1 * n * n / denominator
cosx = cosx + x1
i = i + 1
True
accuracy <= abs(cosval -
cosx) AND i <= 100
False
False
abs(cosx -
accuracy) >
0.0001
True
cosx = xround(cosx, 4)
cosx = xround(cosx, 2)
Return Real cosx

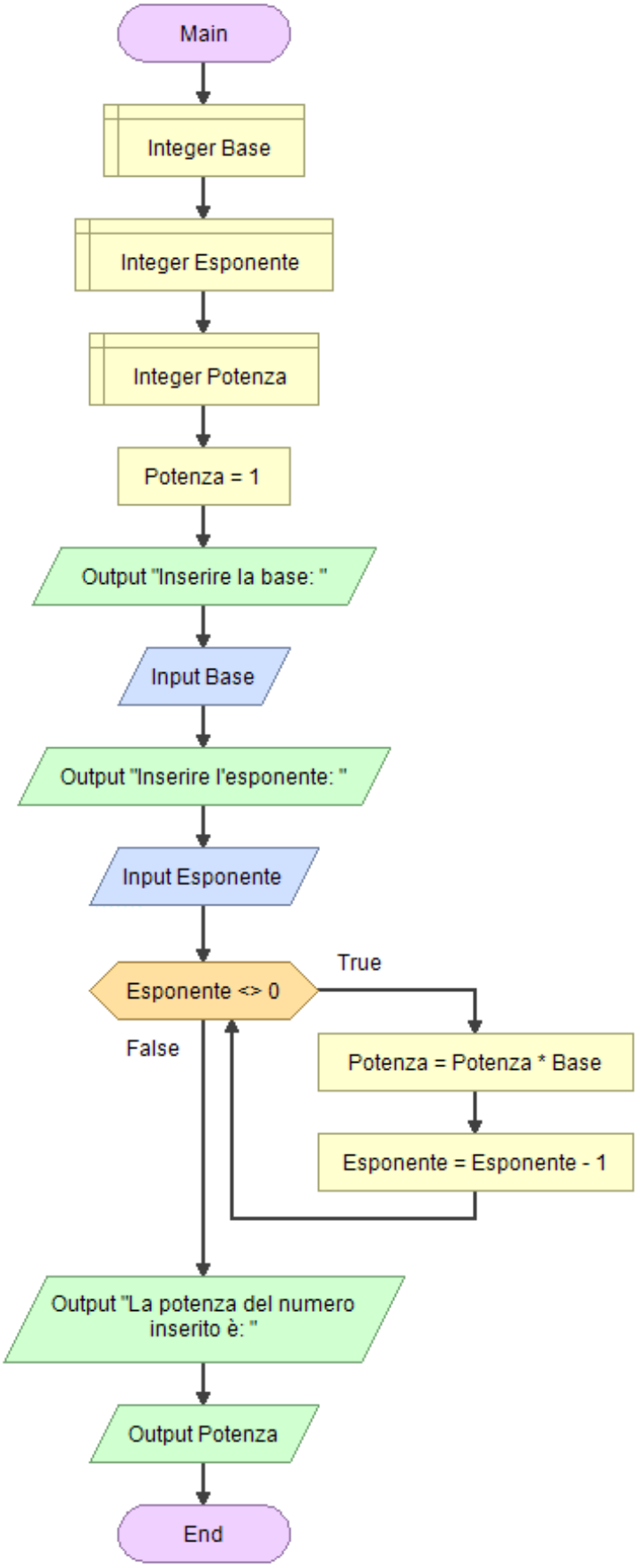
95. G ‖ Dato un numero N intero trovare tutti i suoi divisori

Find all divisors of an integer N



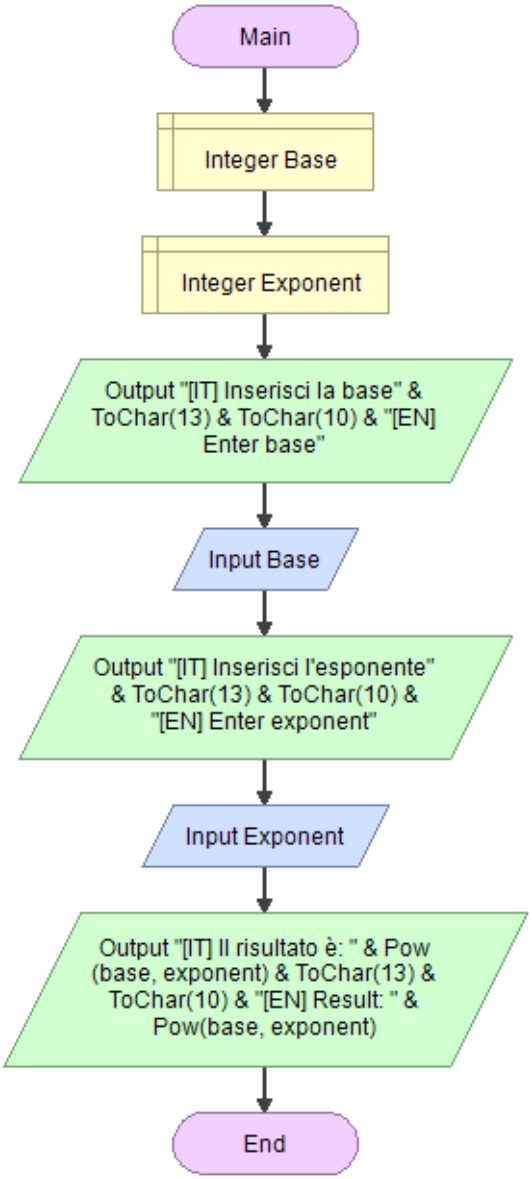
96. G ‖ Dati un numero b (base) ed un numero e (esponente) calcolare la potenza $p = b^e$

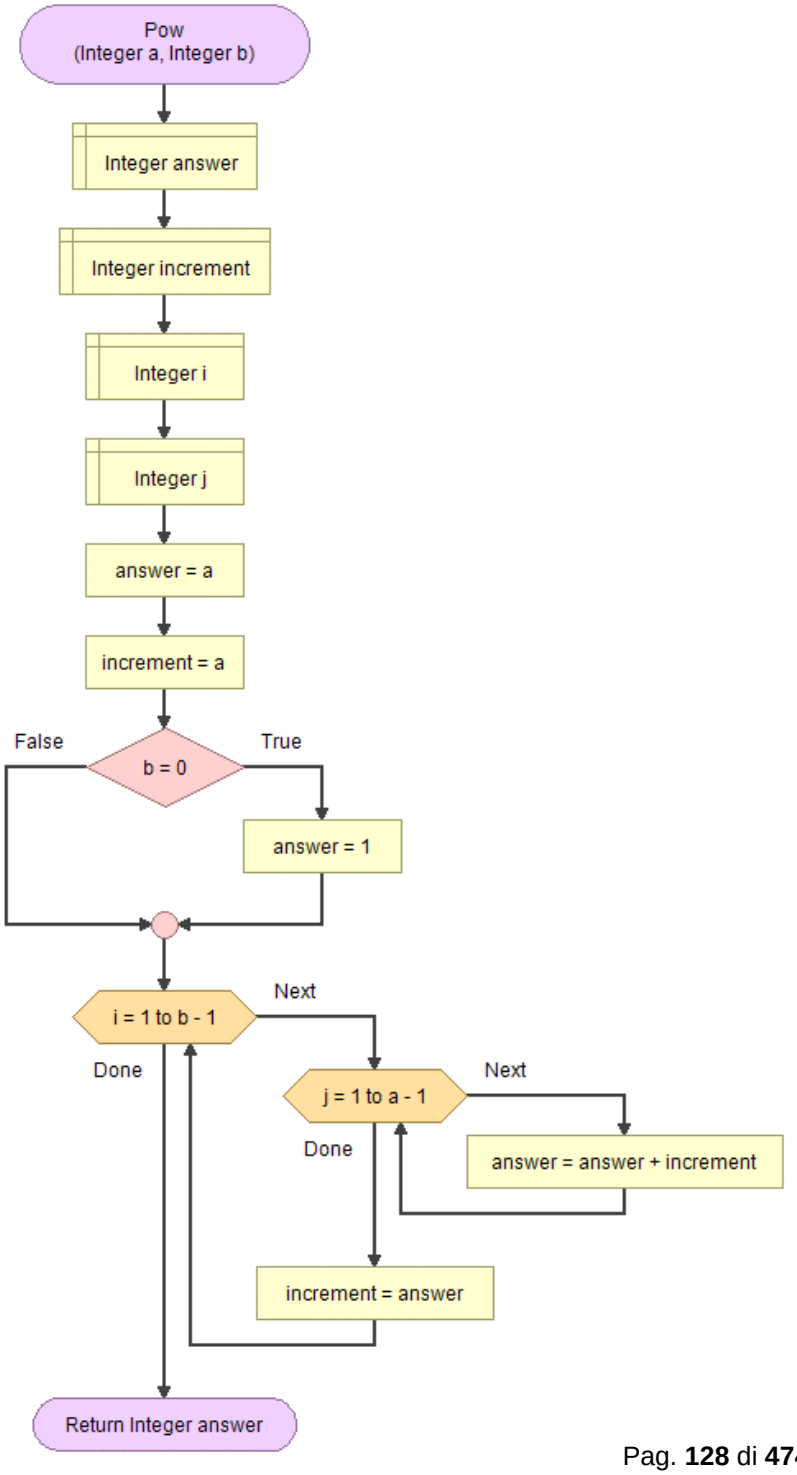
Given a number b (base) and a number e (exponent) calculate the power $p = b^e$



97. **G F** ‖ Calcolo di una potenza senza usare moltiplicazione o divisione

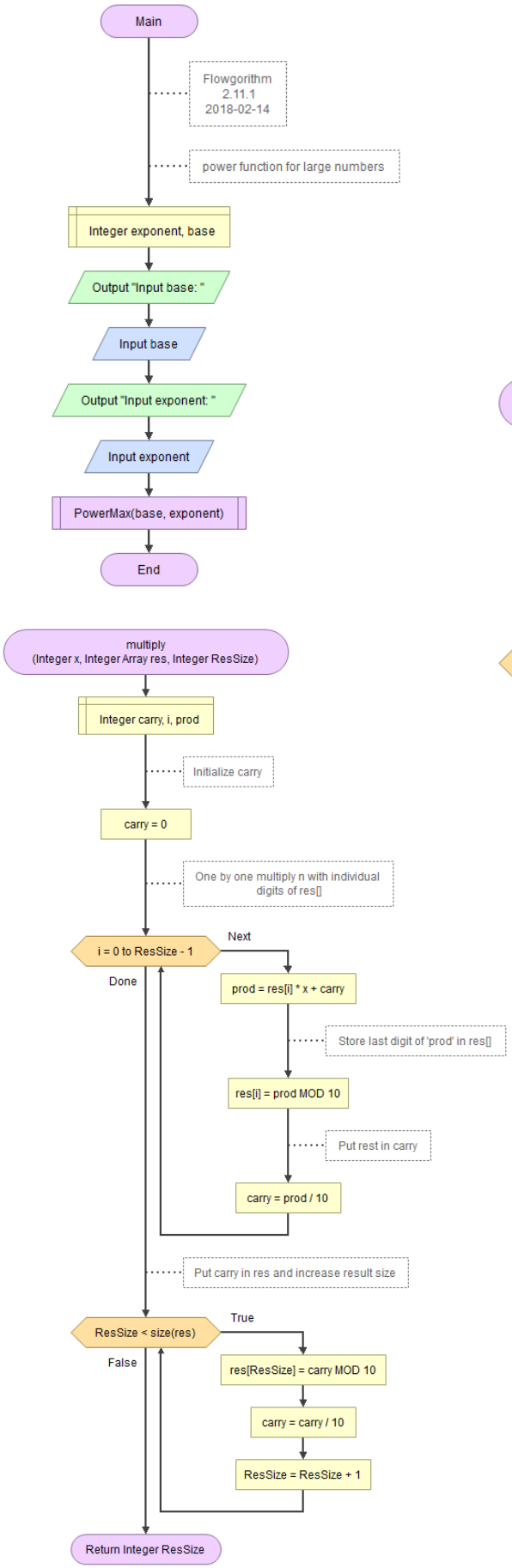
Given a base and an exponent calculate the power $p = b^e$ w/o using multiplication or division

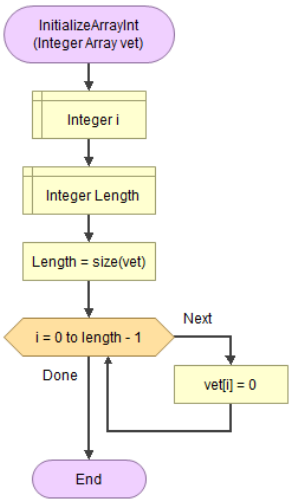


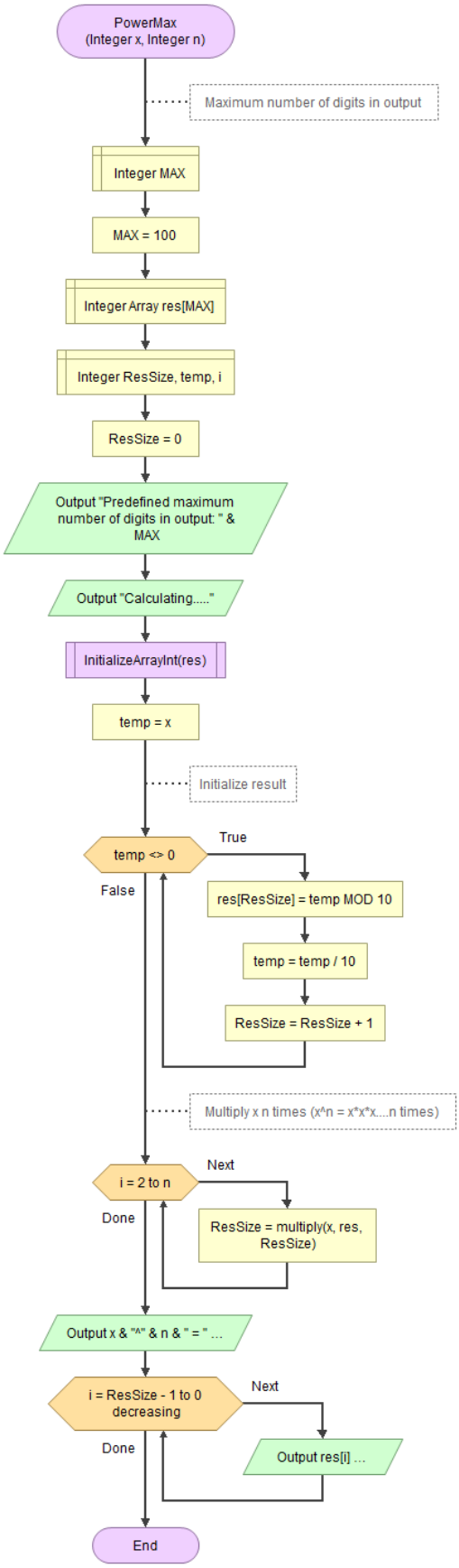


98. GAF ‖ Calcolo di una potenza per grandi numeri

Power function for large numbers

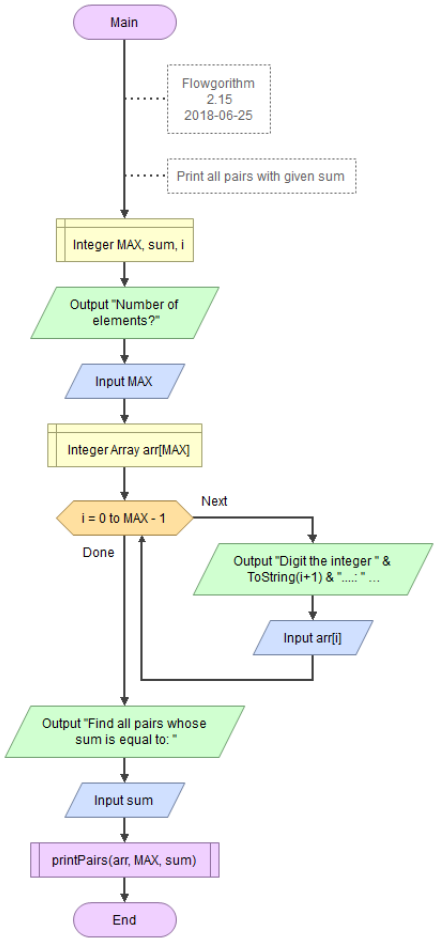


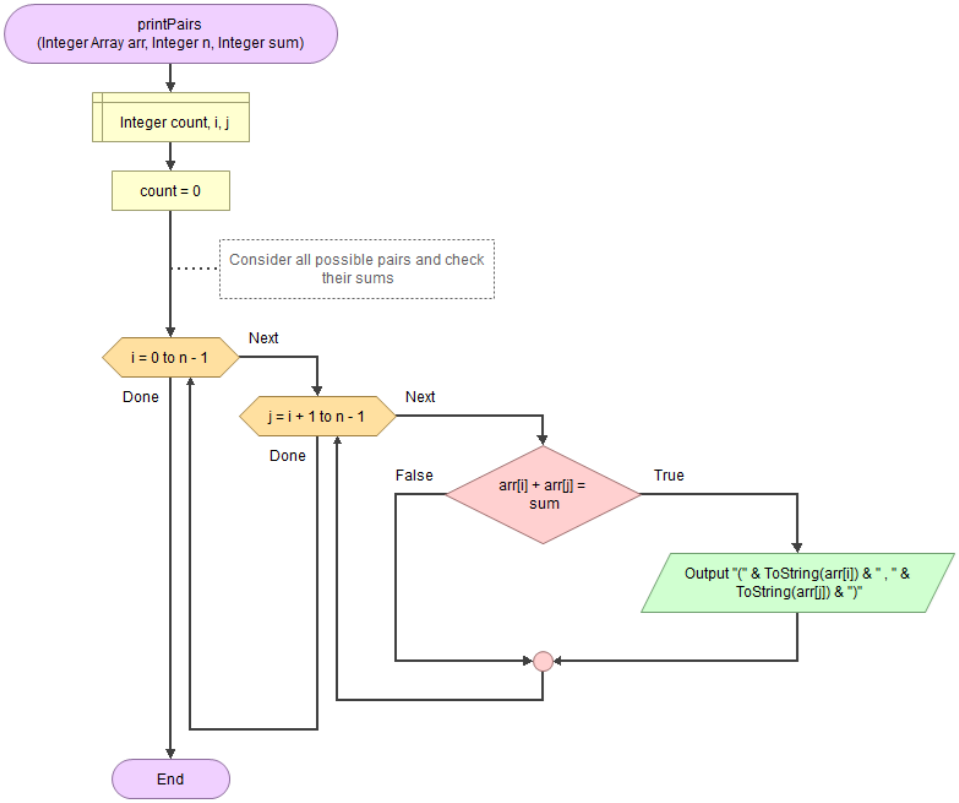




99. **G A F** ‖ Visualizza tutte le coppie di numeri la cui somma sia uguale a N

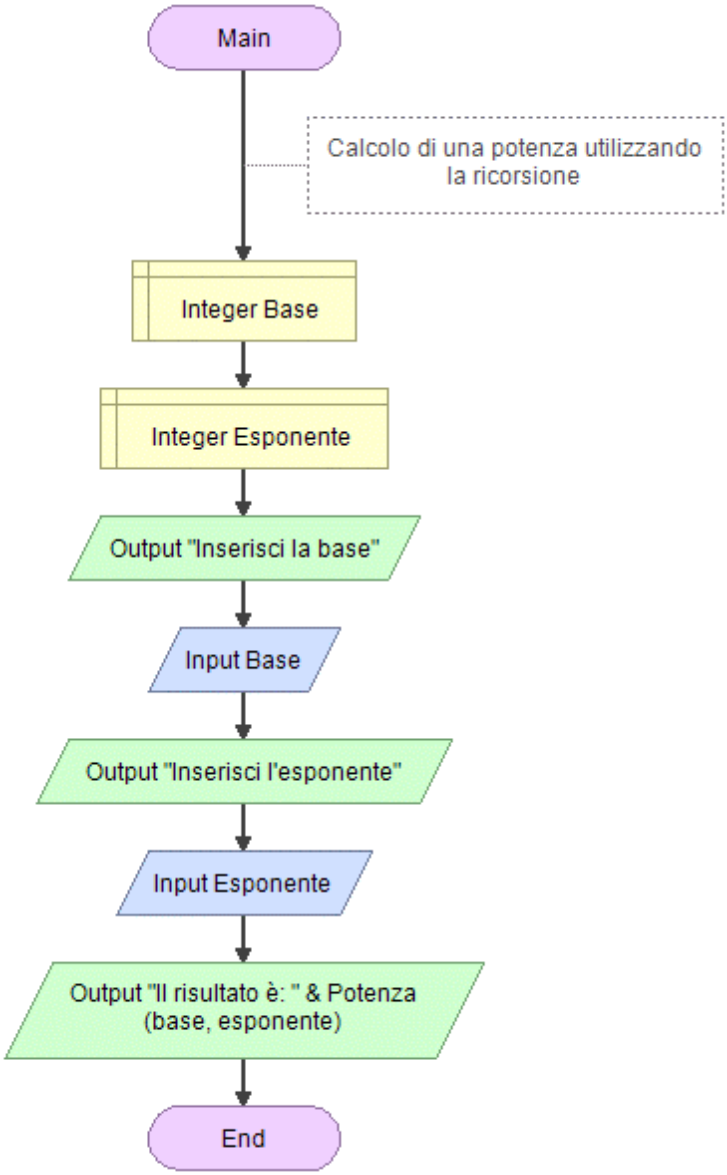
Print all pairs with given sum

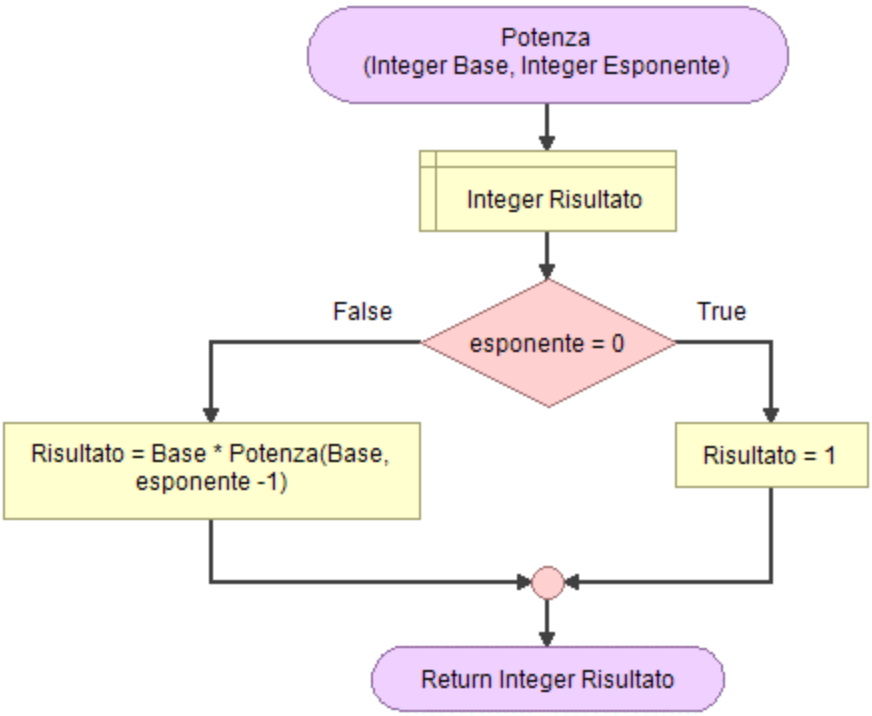




100.G F R ‖ Calcolo di una potenza utilizzando la ricorsione

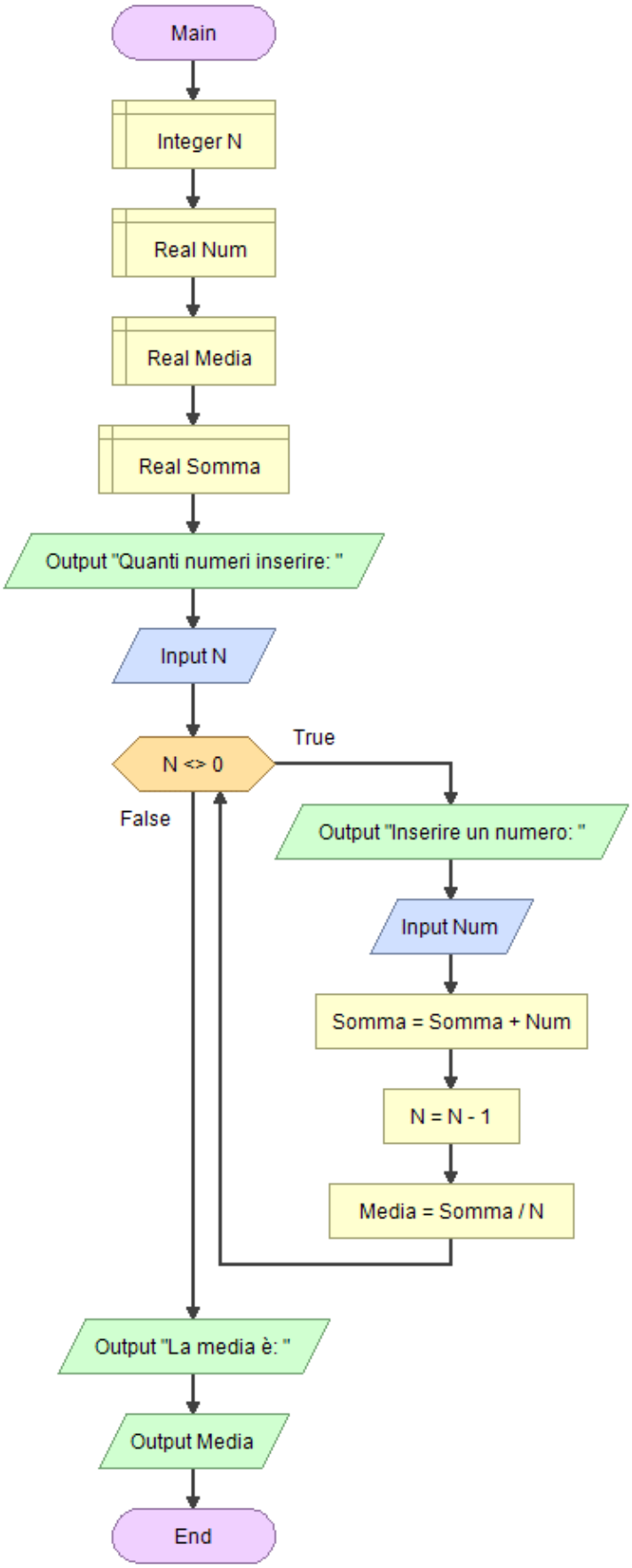
Calculating a power (recursive)





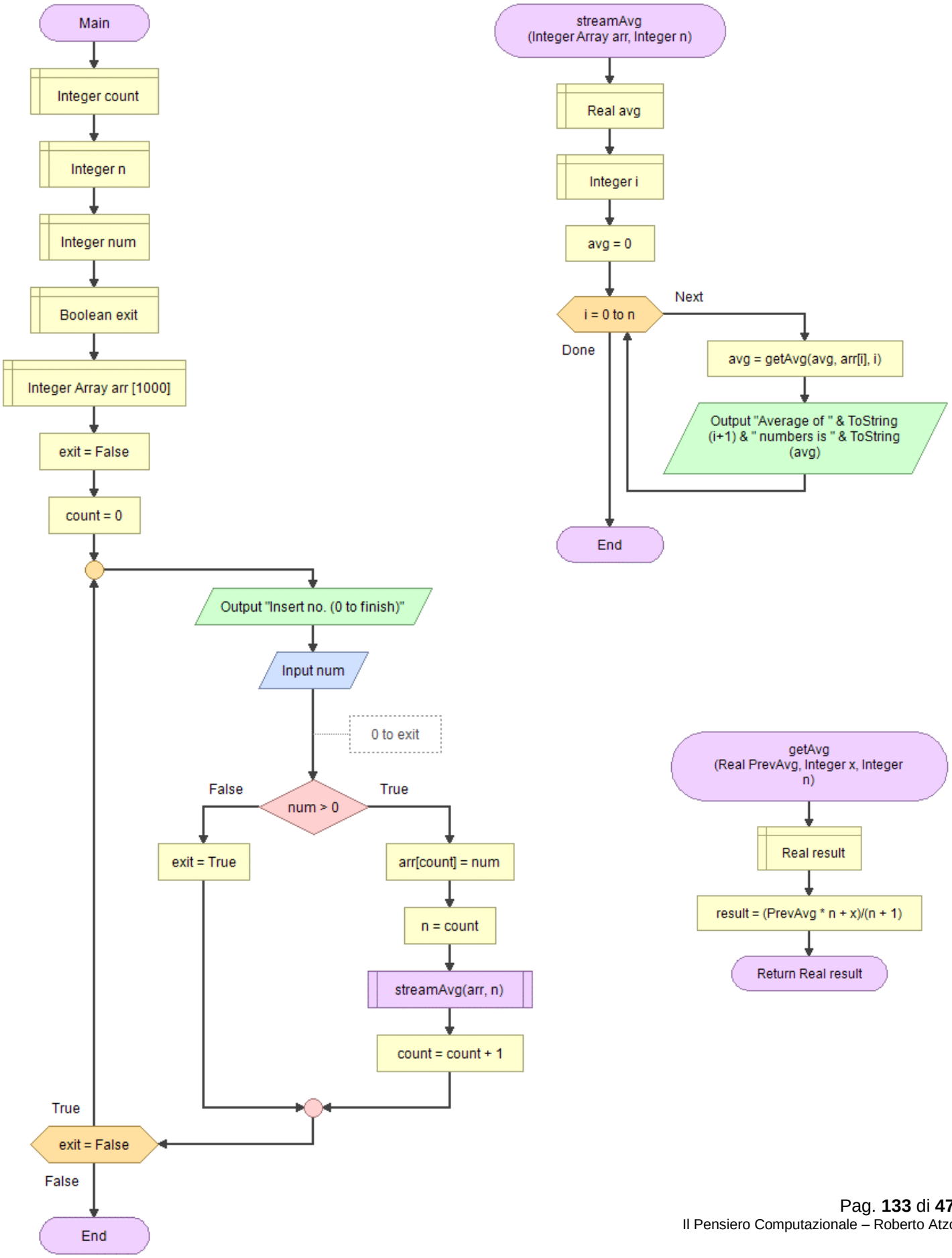
101.G ‖ Dati n numeri reali calcolarne la media aritmetica

Given n real numbers to calculate the arithmetic average



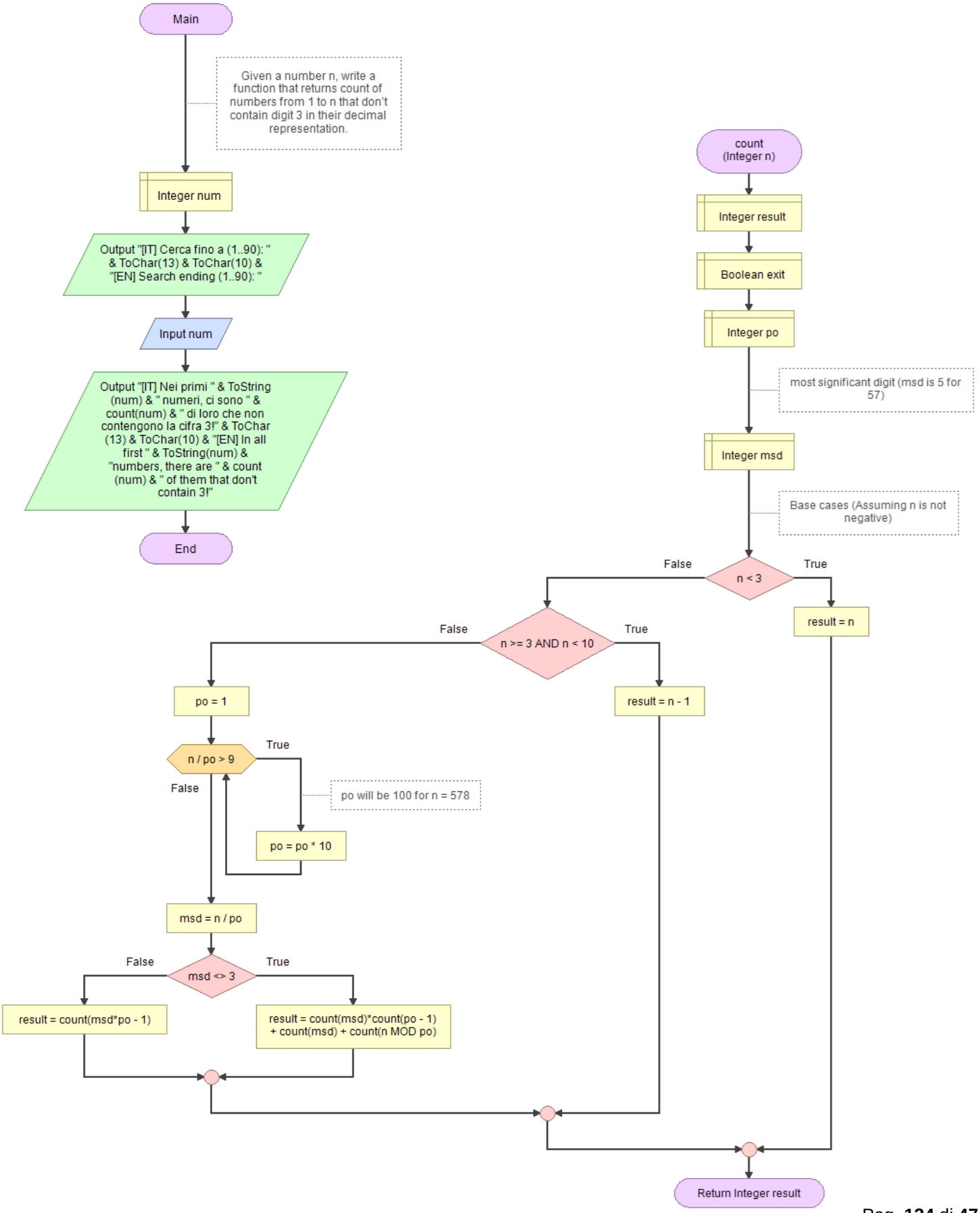
102. G F ‖ Calcolo della media di un flusso di numeri

Average of a stream of numbers



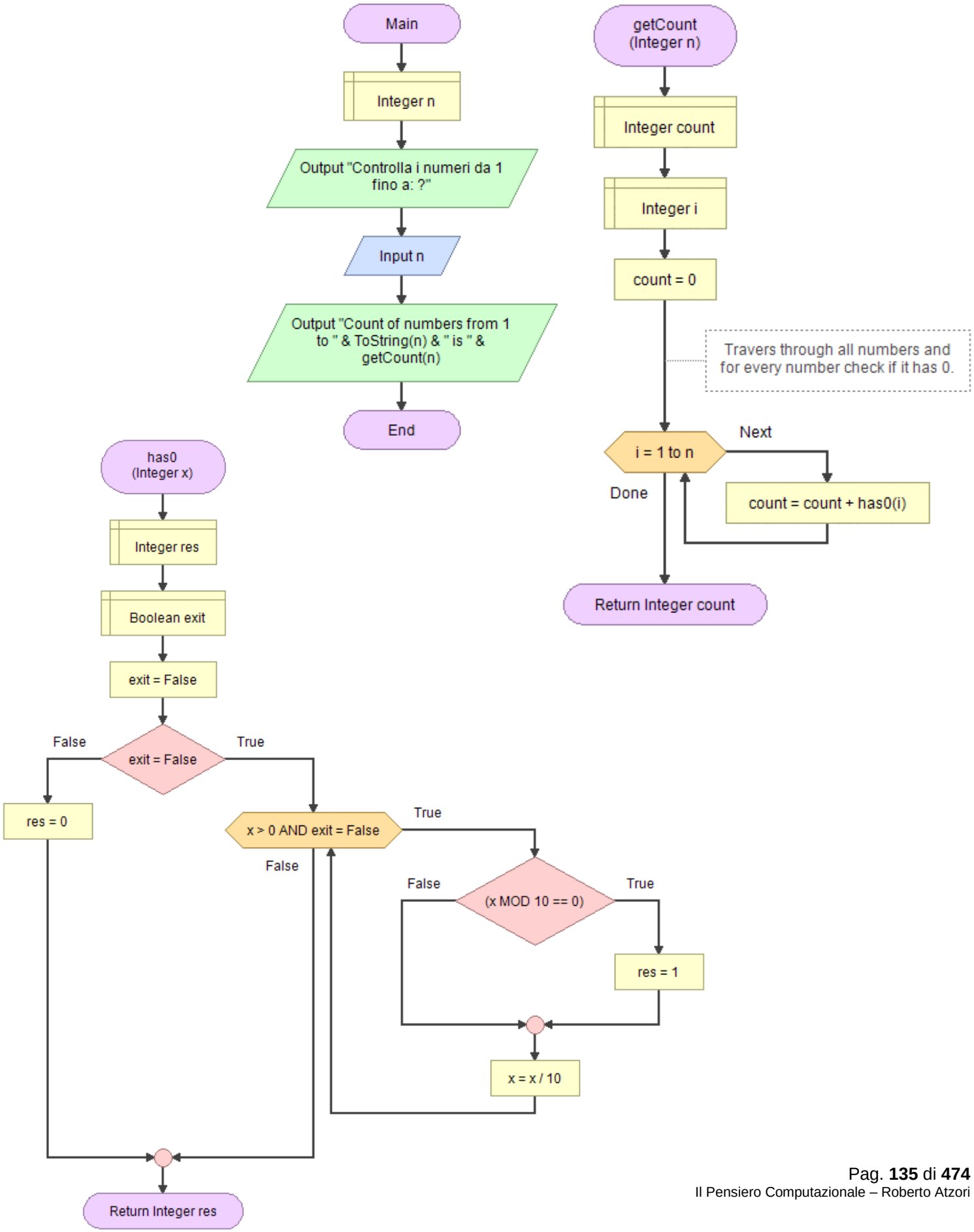
103. G F ‖ Conta i numeri che non contengono la cifra 3

Count numbers that don't contain 3



104. **G F** | Contare quanti numeri possiedono '0' come cifra

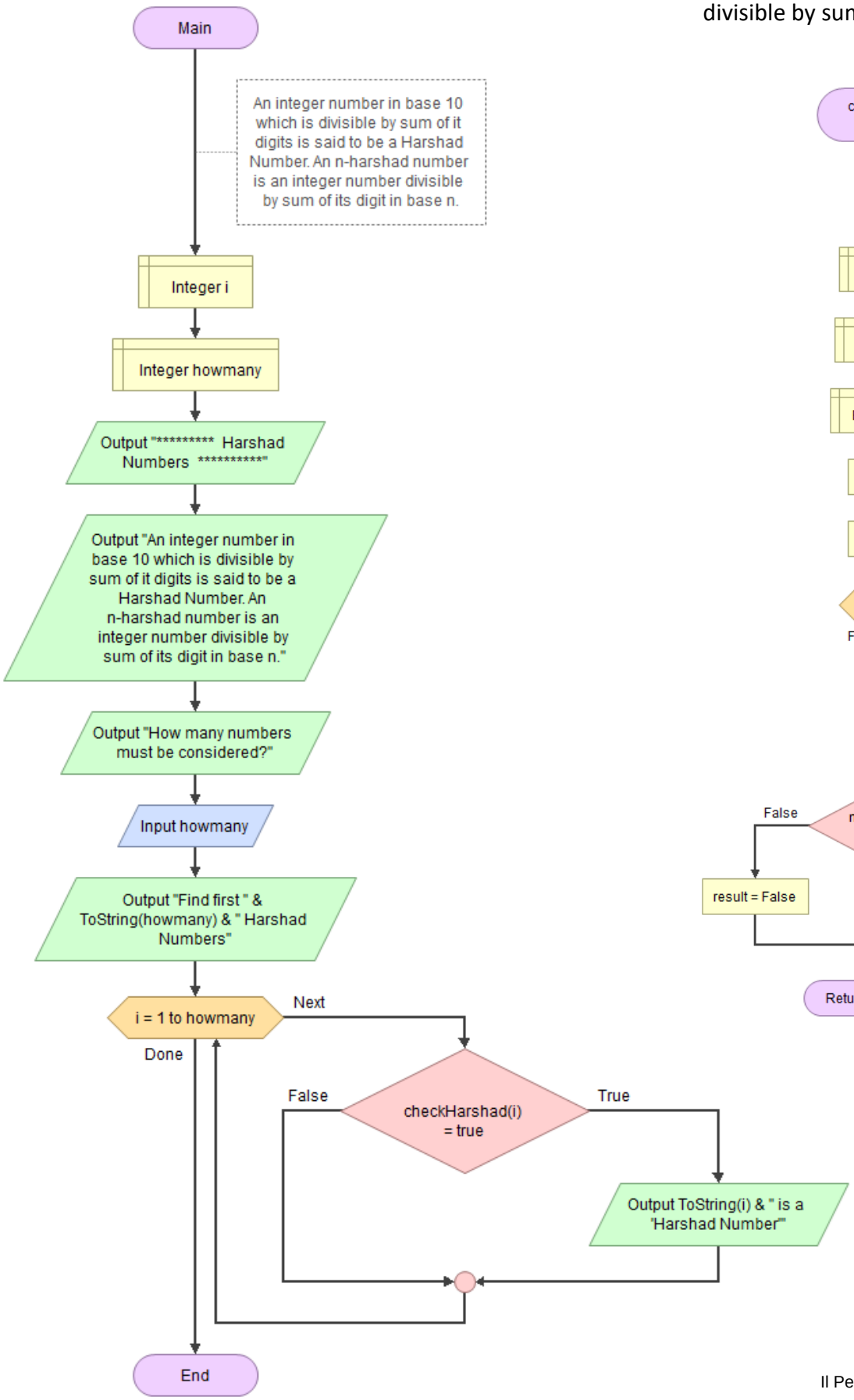
Count numbers that contain 0 (zero)

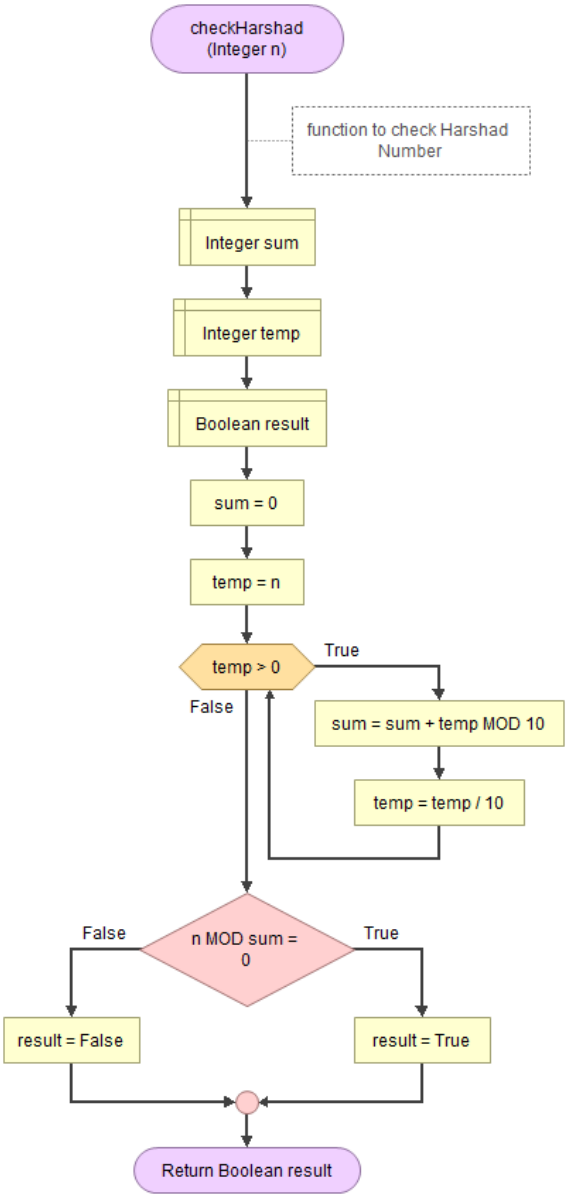


105.G F ‖ Numero di Harshad

Harshad number

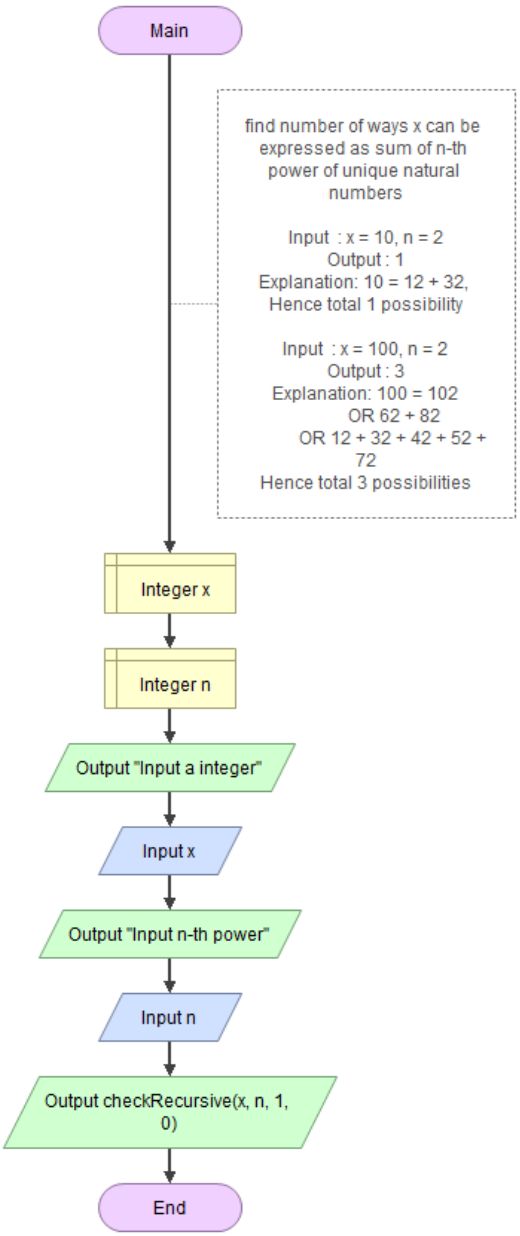
An n-harshad number is an integer number divisible by sum of its digit in base n.

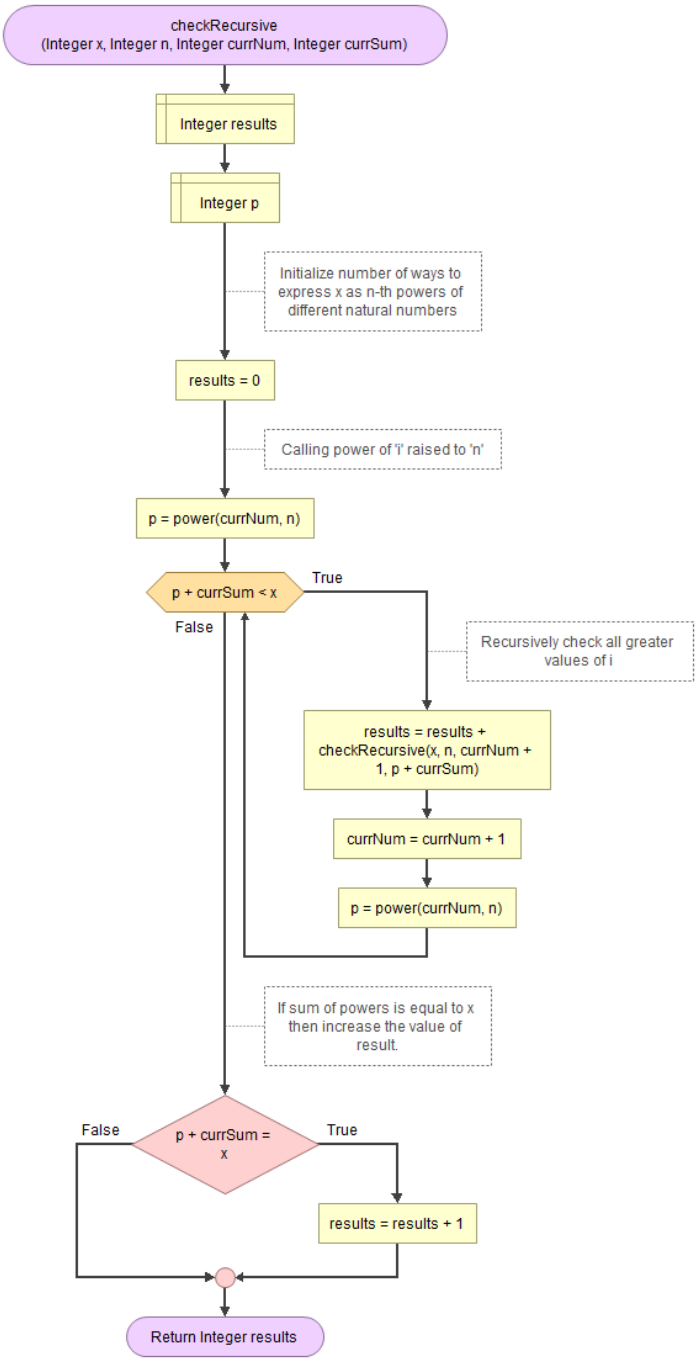


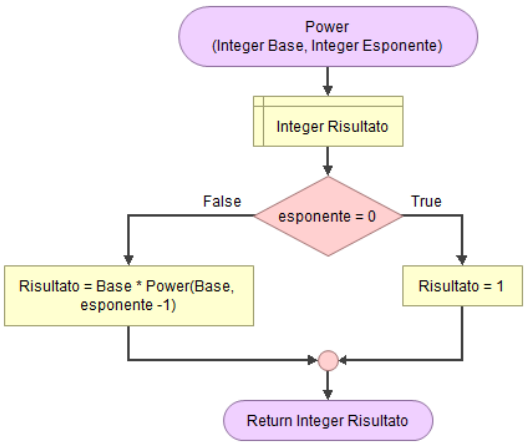


106.G F ‖ Intero come somma di numeri naturali unici all' n-esima potenza

Integer as sum of n-th power of unique natural numbers

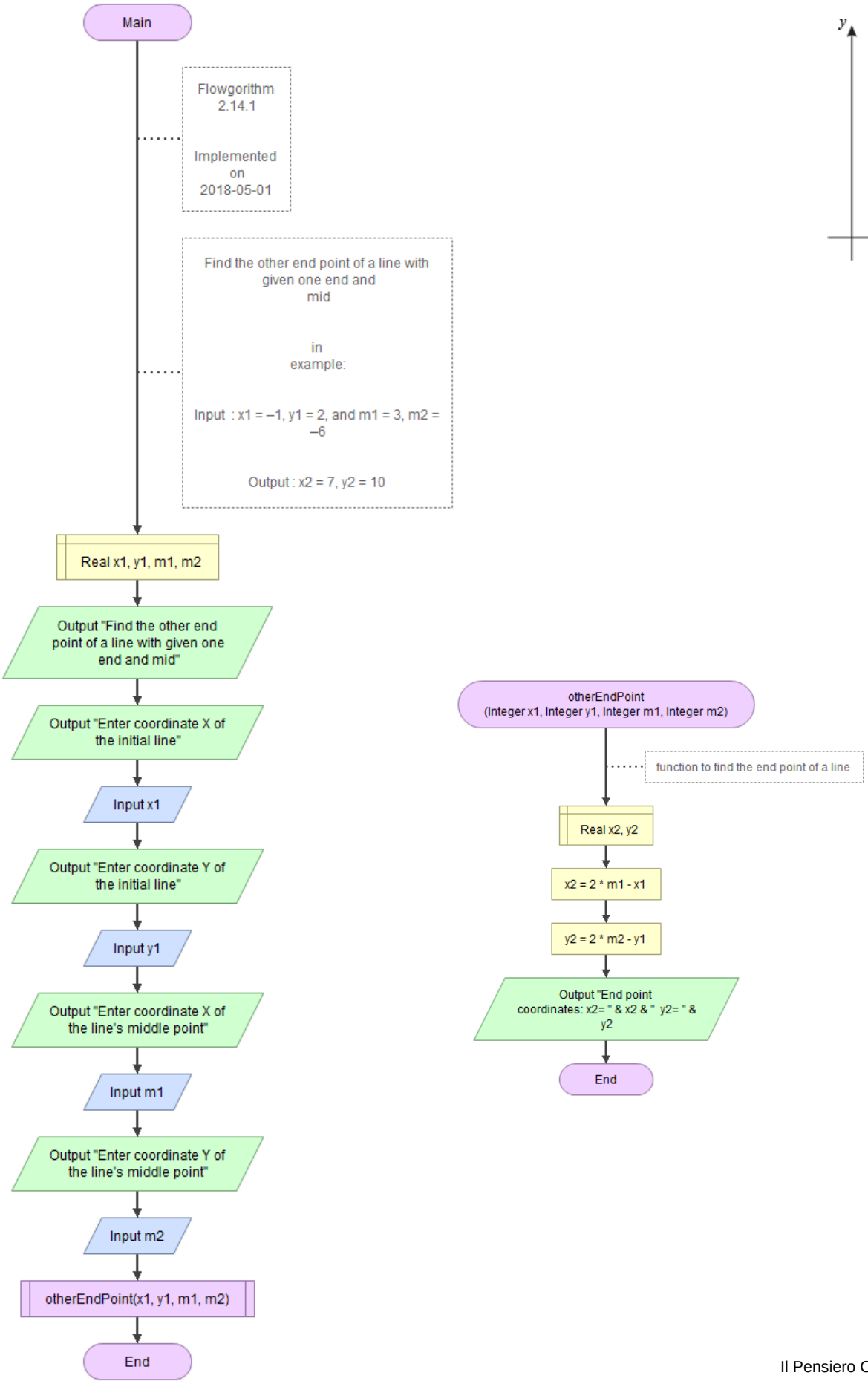


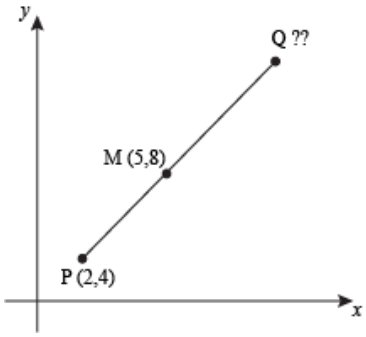




107.G F ‖ Trovare l'altra coordinata di un punto di una linea sapendo le coordinate di un punto e del punto intermedio

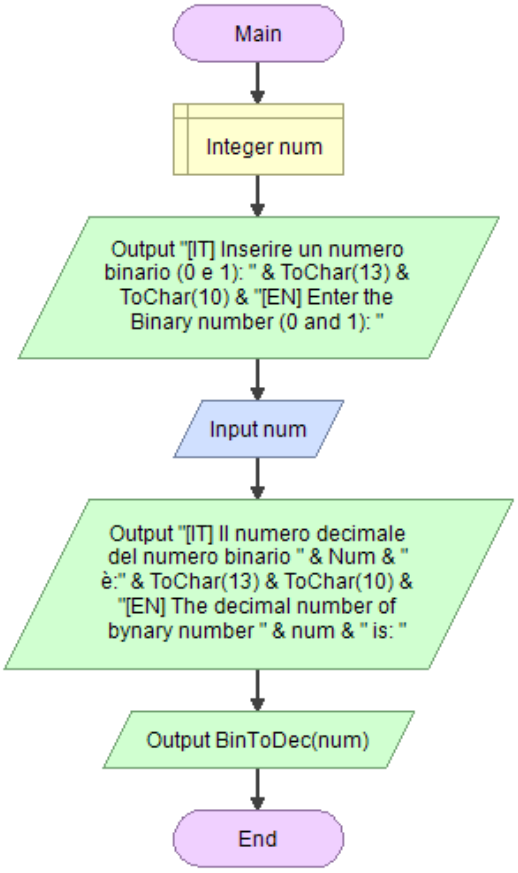
Find the other end point of a line with given one end and mid

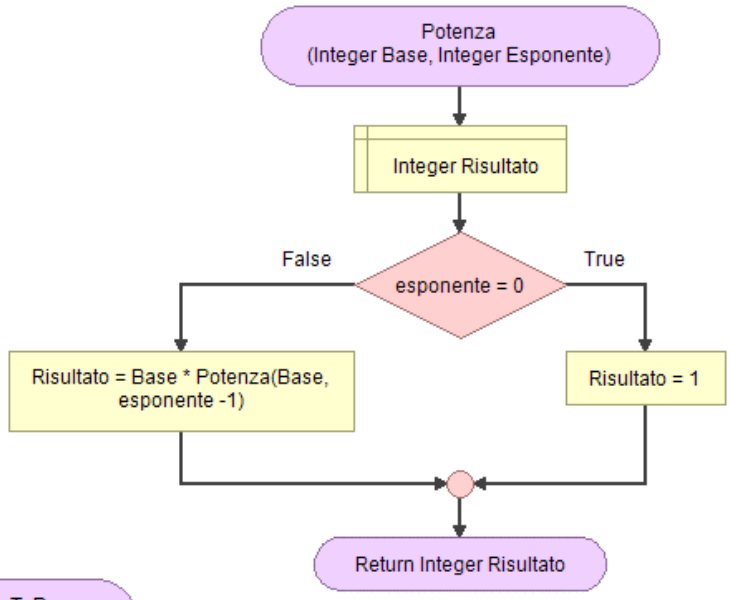


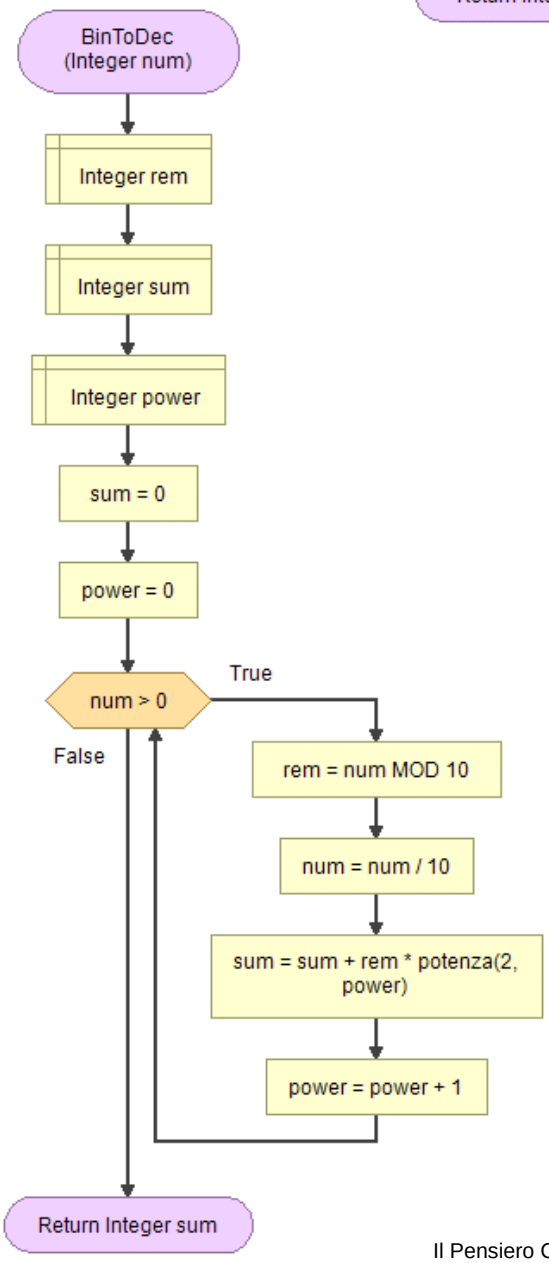


108.G F ‖ Conversione binario decimale

Convert Binary number to Decimal one



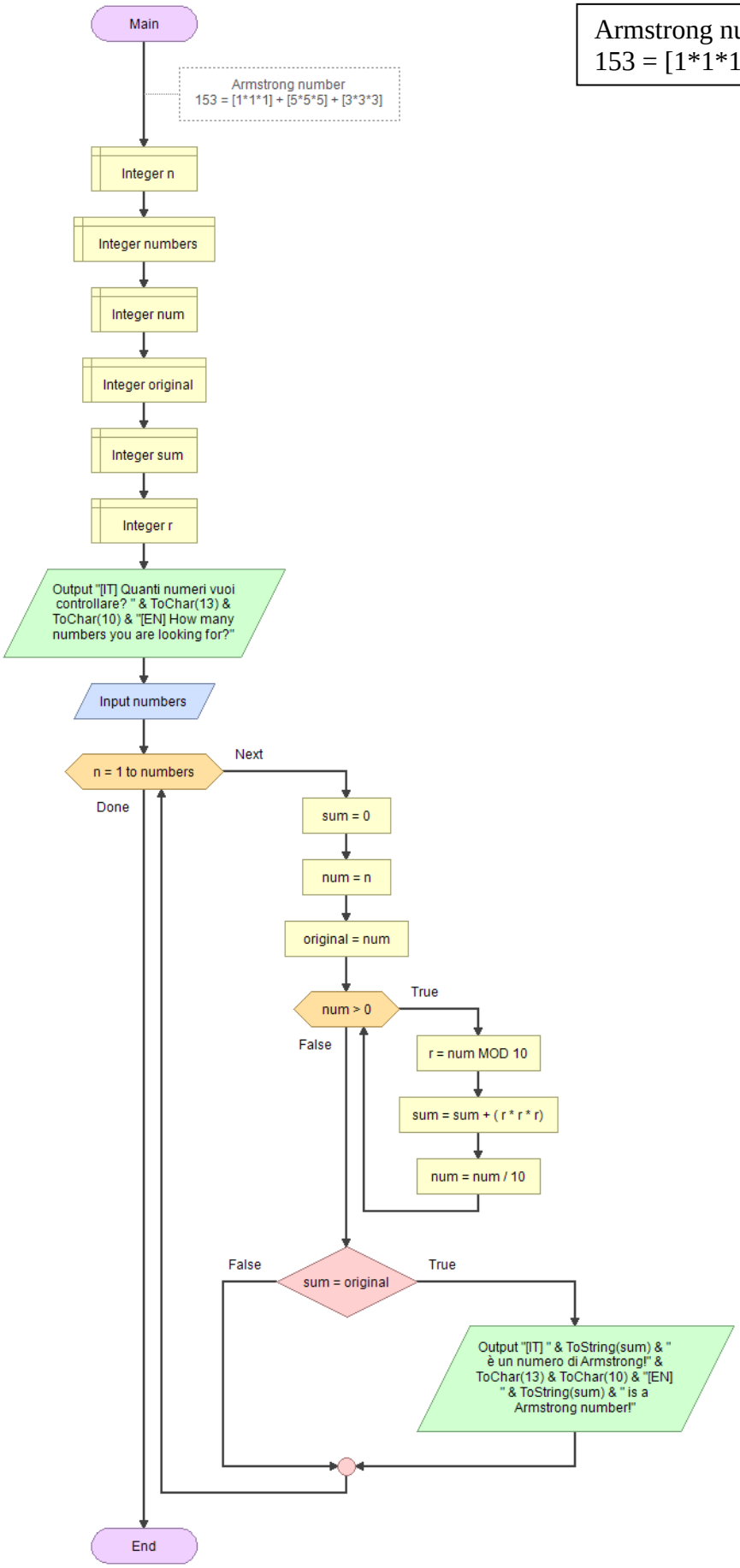




109.G ‖ Trovare i numeri di Armstrong

Check for Armstrong Numbers

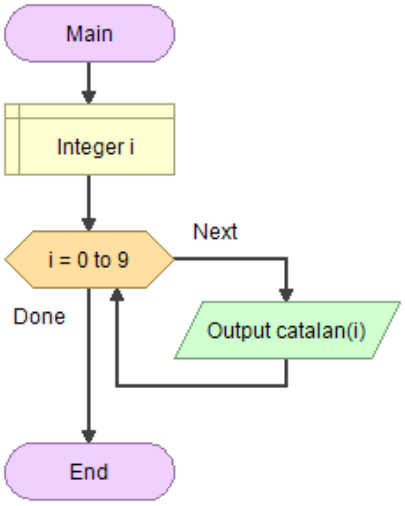
Armstrong number
153 = [1*1*1] + [5*5*5] + [3*3*3]

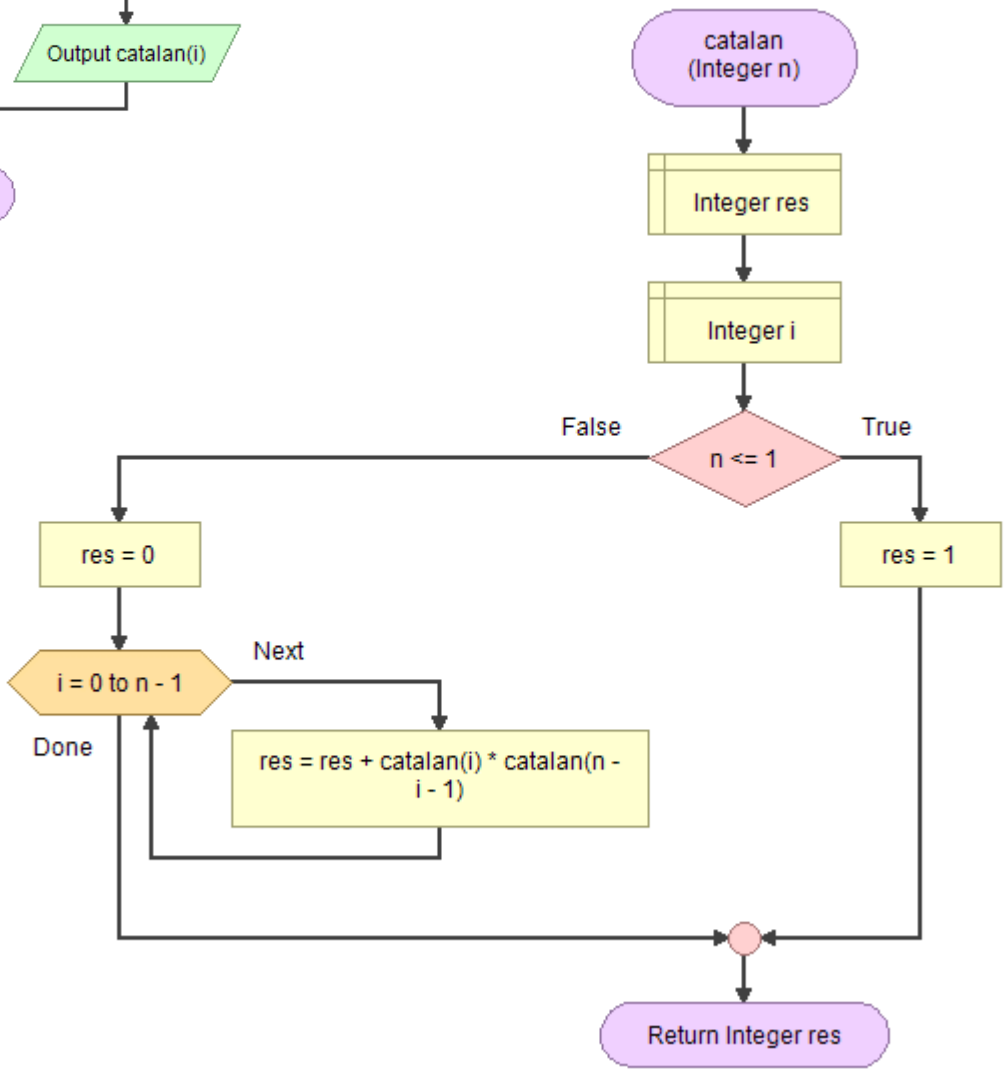


Catalan numbers

I numeri di Catalan soddisfano la seguente formula ricorsiva

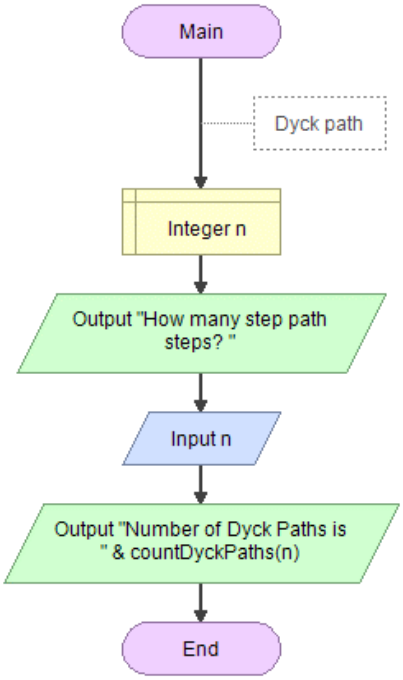
$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!} \qquad \text{per } n \geqslant 0$$





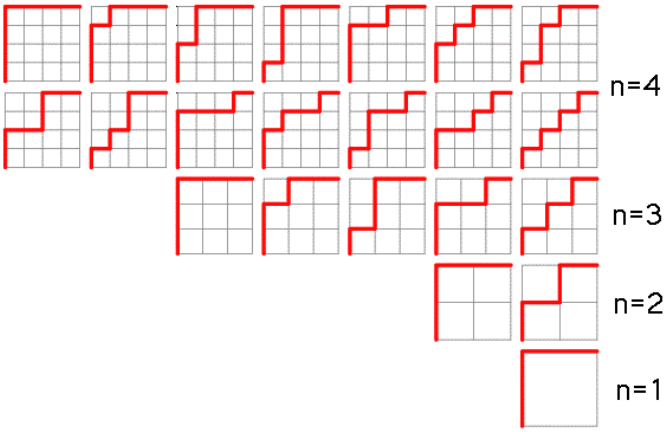
111.G F ‖ Percorso Dyck

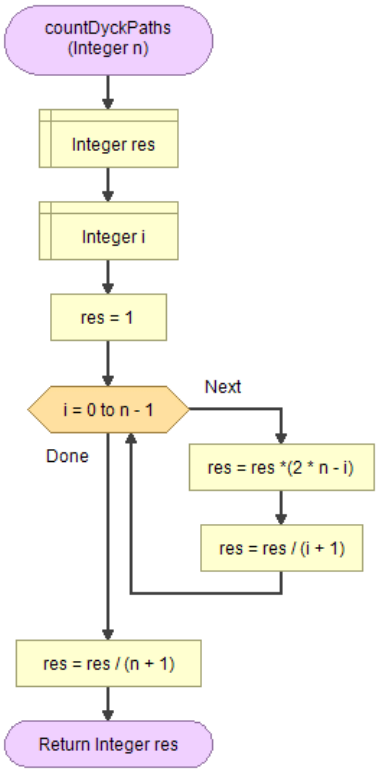
Dyck path



Consider a n x n grid with indexes of top left corner as (0, 0). Dyck path is a staircase walk from bottom left, i.e., (n-1, 0) to top right, i.e., (0, n-1) that lies above the diagonal cells (or cells on line from bottom left to top right).

The task is to count the number of Dyck Paths from (n-1, 0) to (0, n-1).





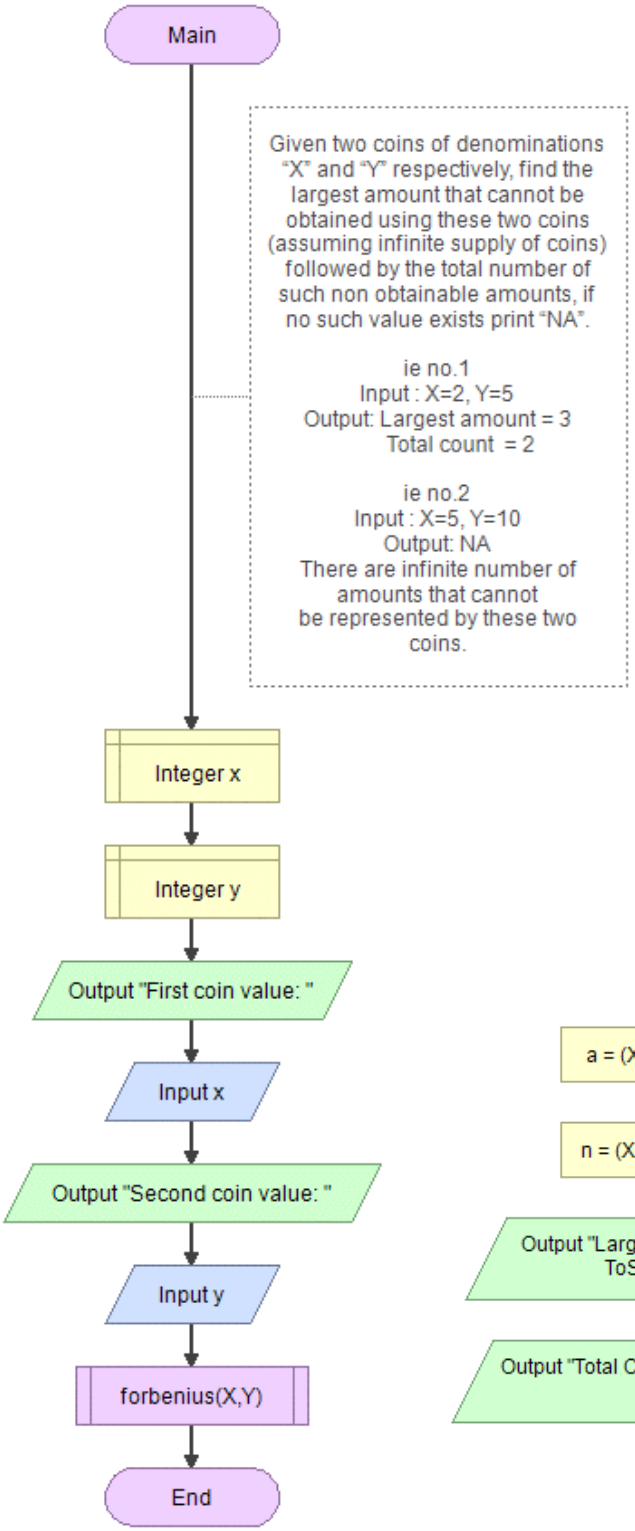
The number of Dyck paths from (n-1, 0) to (0, n-1) can be given by the Catalan numberC(n).

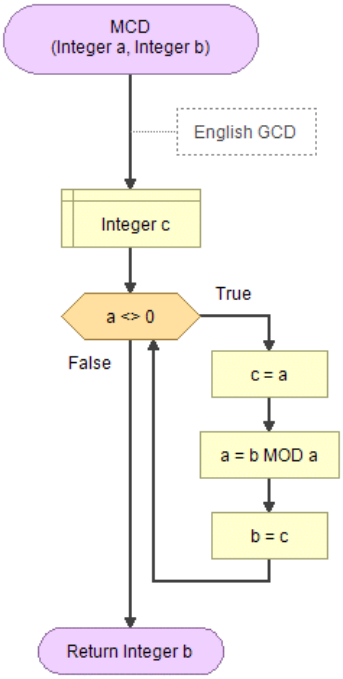
$$C_n = \frac{(2n)!}{(n+1)!\,n!} = \prod_{k=2}^{n} \frac{n+k}{k} \qquad \text{for } n \geq 0$$

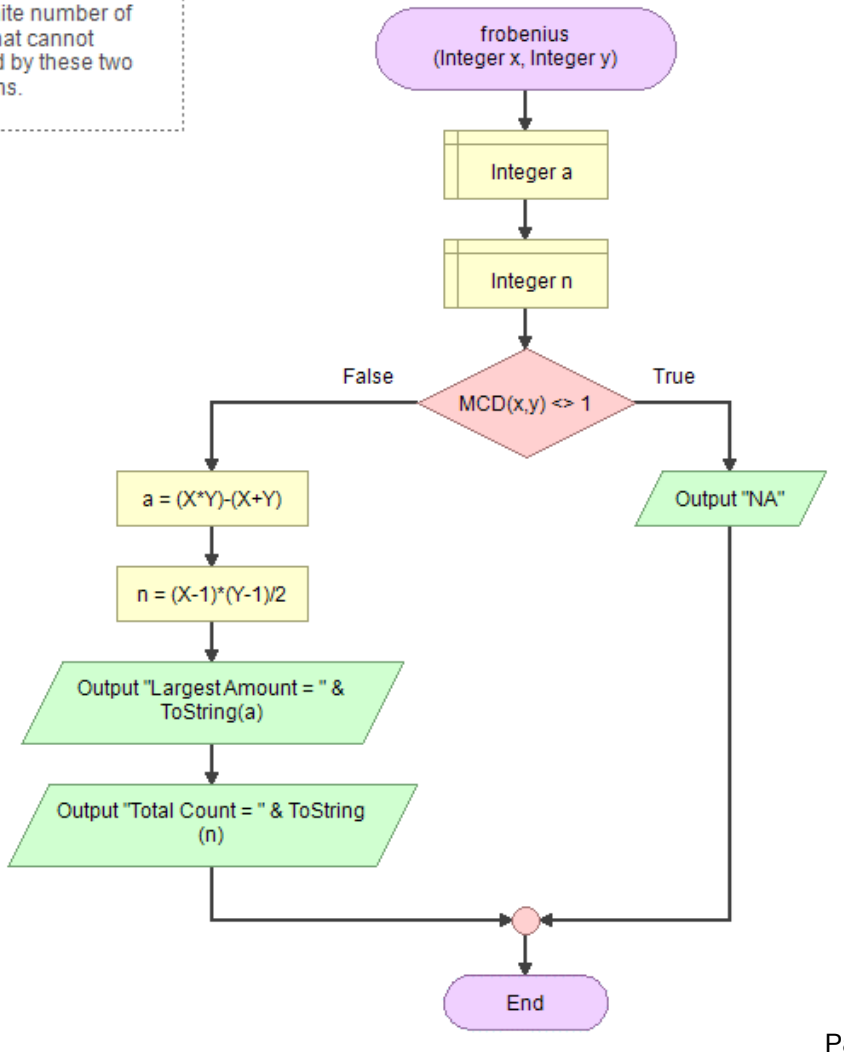
112. **G F** ‖ Date due monete di taglio "X" e "Y", trovare la più grande quantità che non può essere ottenuta utilizzando queste due monete (Frobenius coin)

Frobenius coin problem

https://it.wikipedia.org/wiki/Problema_delle_monete





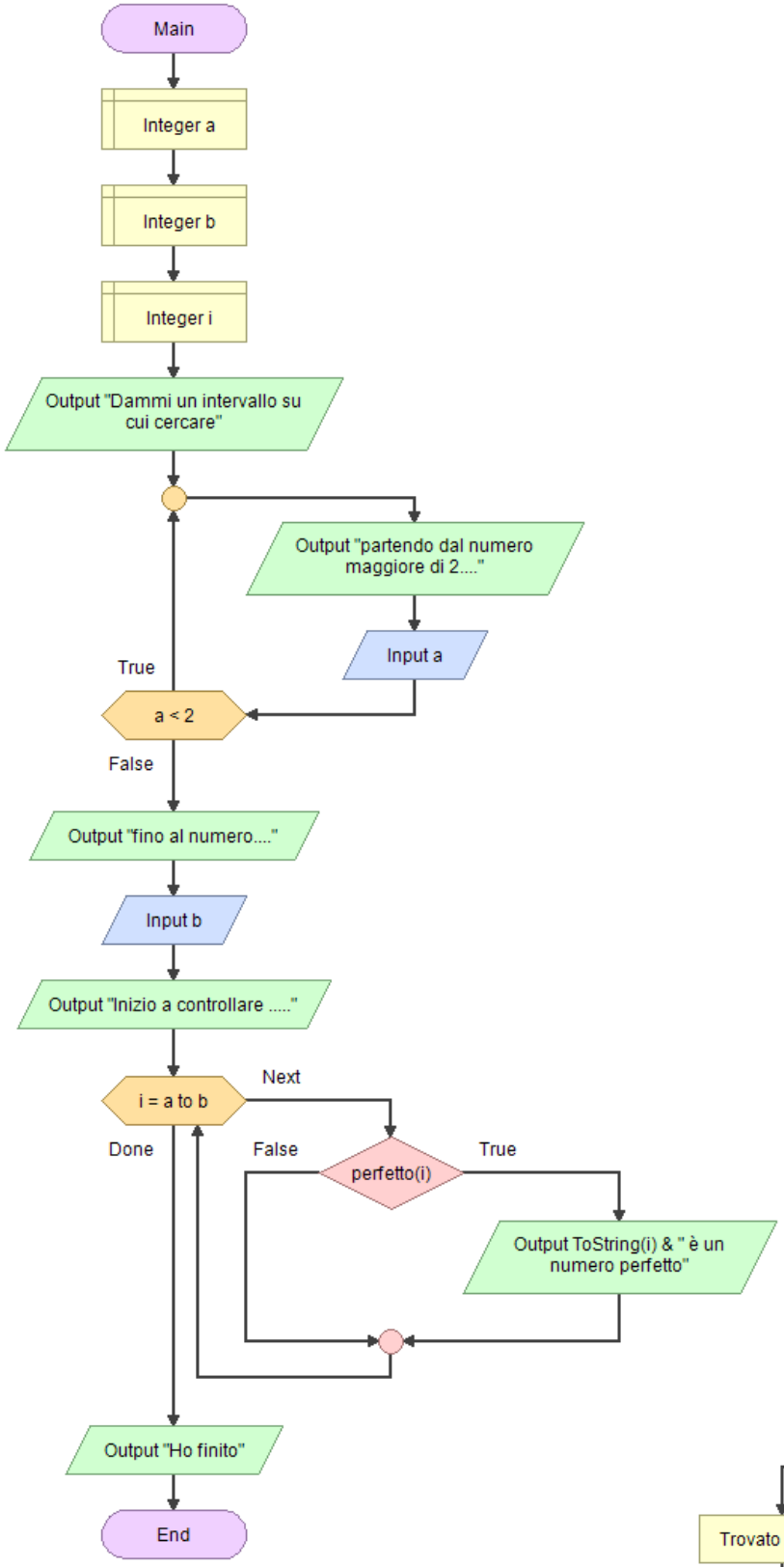


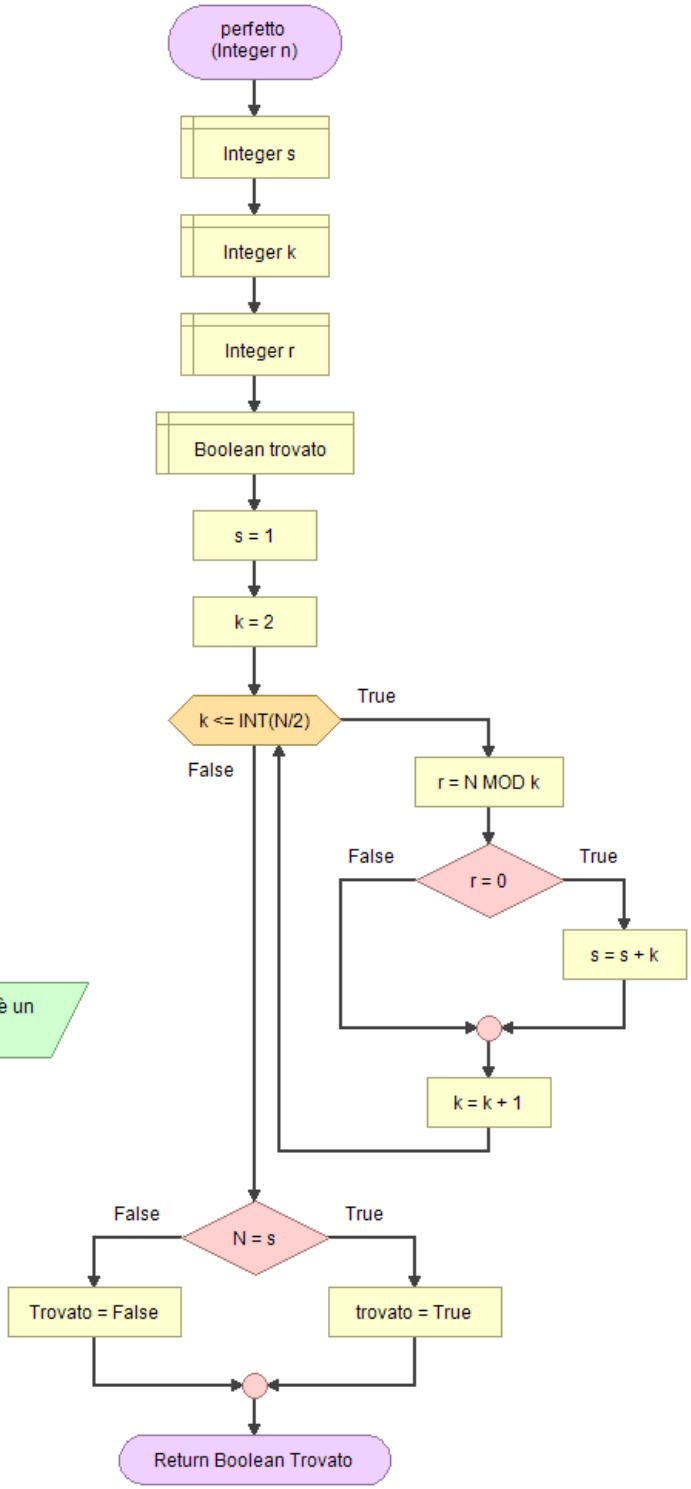
113. **G F** ‖ Dato un numero N intero (N > 0) verificare se è un numero perfetto.

Perfect numbers

Un numero naturale *n* si dice **perfetto** quando *n* è uguale alla somma dei suoi divisori. Ad esempio, il numero 28, divisibile per 1, 2, 4, 7, 14 è un numero perfetto e anche 6 che è divisibile per 1, 2 e 3.

6 = 1 + 2 + 3
28 = 1 + 2 + 4 + 7 + 14





114.G ‖ Dati due numeri interi N e M (N > 0 e M > 0) verificare sono amicabili (o amici).

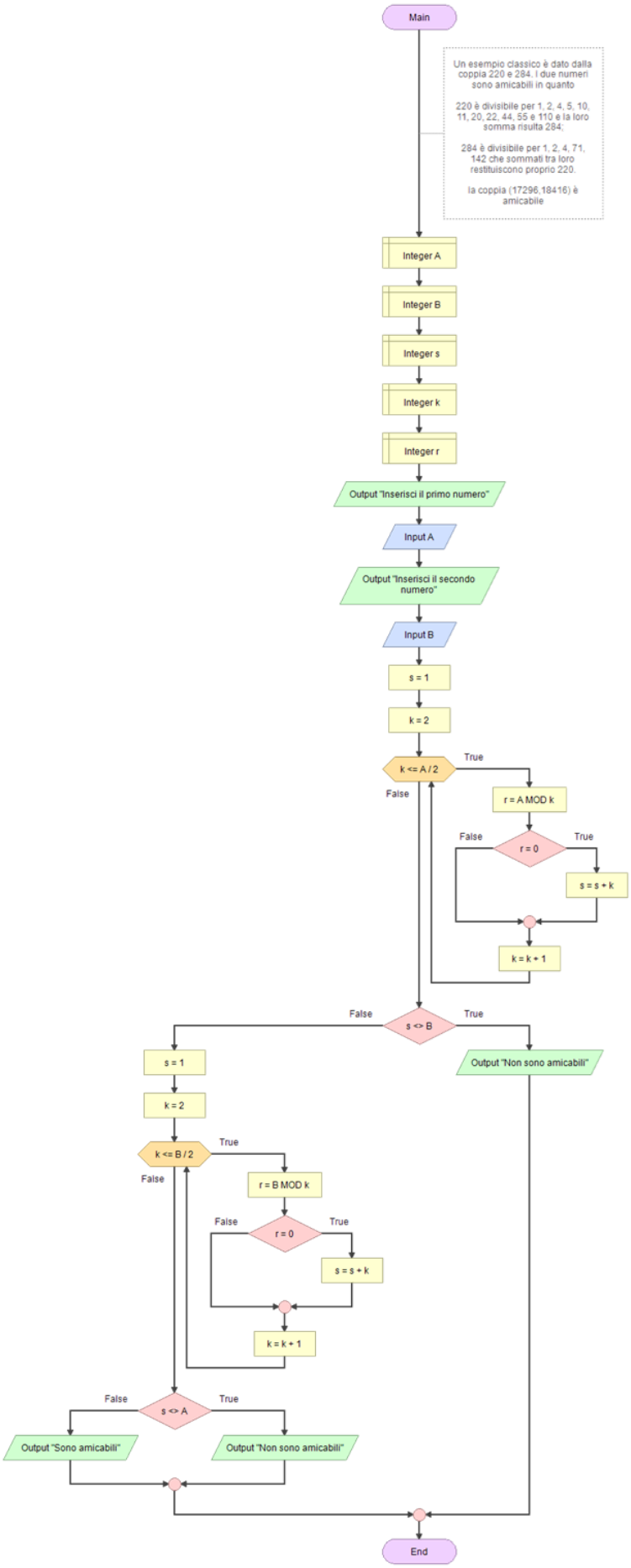
Amicable numbers

http://it.wikipedia.org/wiki/Numeri_amicabili

In matematica, sono **numeri amicabili** o **amici** due numeri per cui la somma dei divisori di uno (escluso il numero stesso) è uguale all'altro e viceversa.

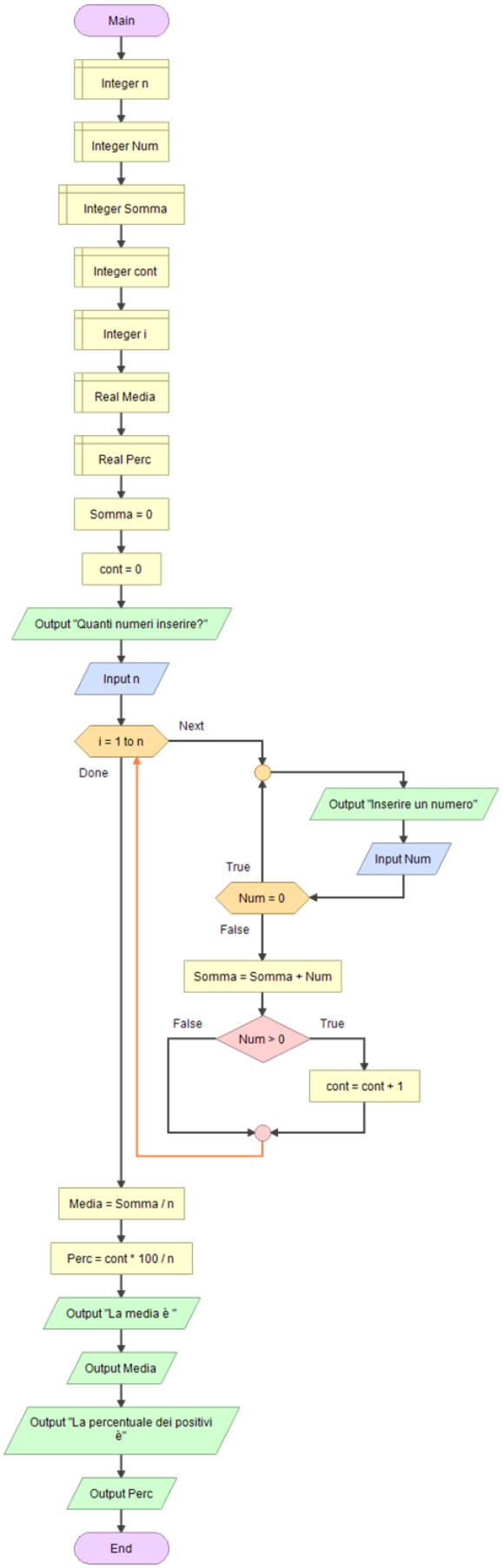
Un esempio classico è dato dalla coppia 220 e 284. I due numeri sono amicabili in quanto

- 220 è divisibile per 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 e 110 e la loro somma risulta 284;
- 284 è divisibile per 1, 2, 4, 71, 142 che sommati tra loro restituiscono



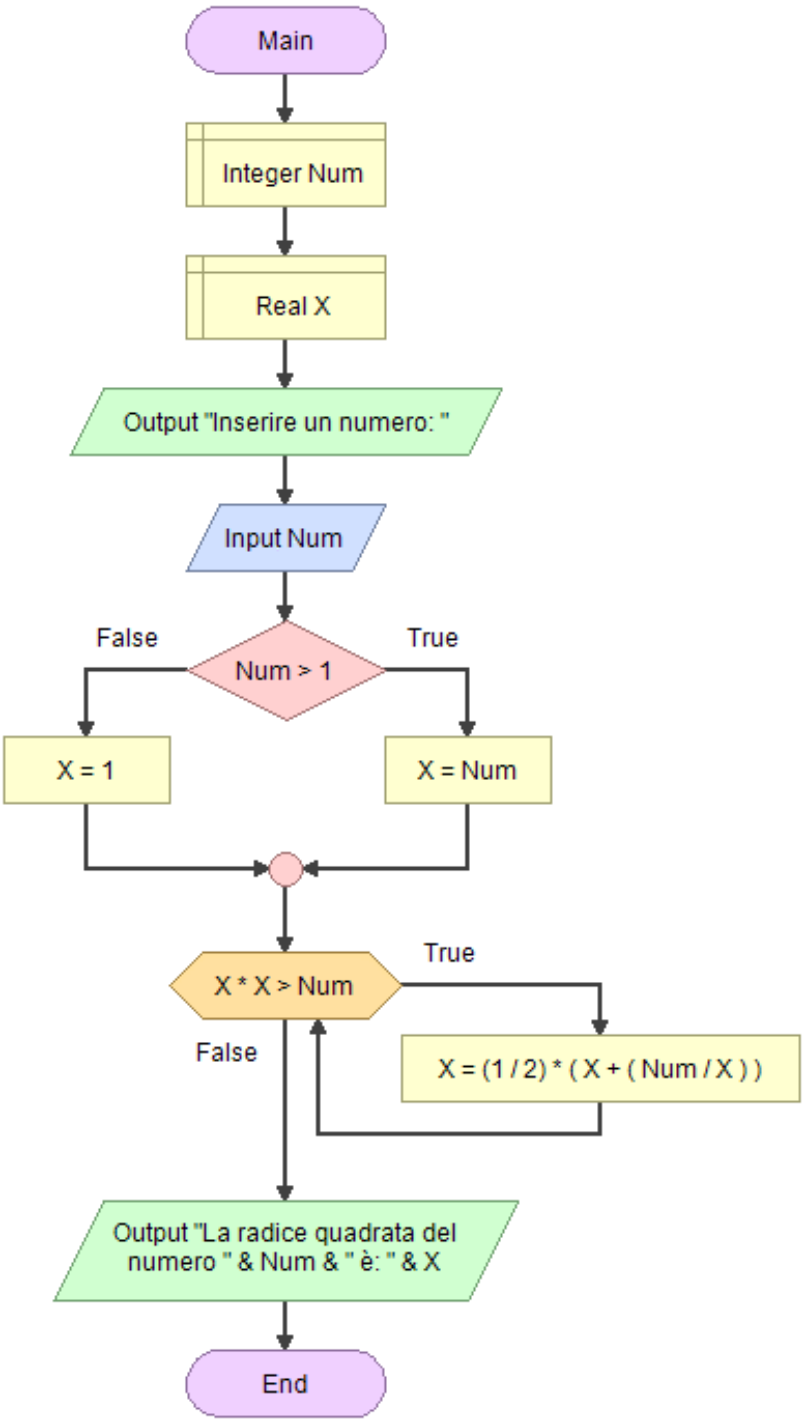
116.G ‖ Dati N valori diversi da zero, calcolare la media aritmetica dei medesimi e la percentuale dei positivi sul totale.

Given N not-zero values calculate the average and the percentage of positive numbers



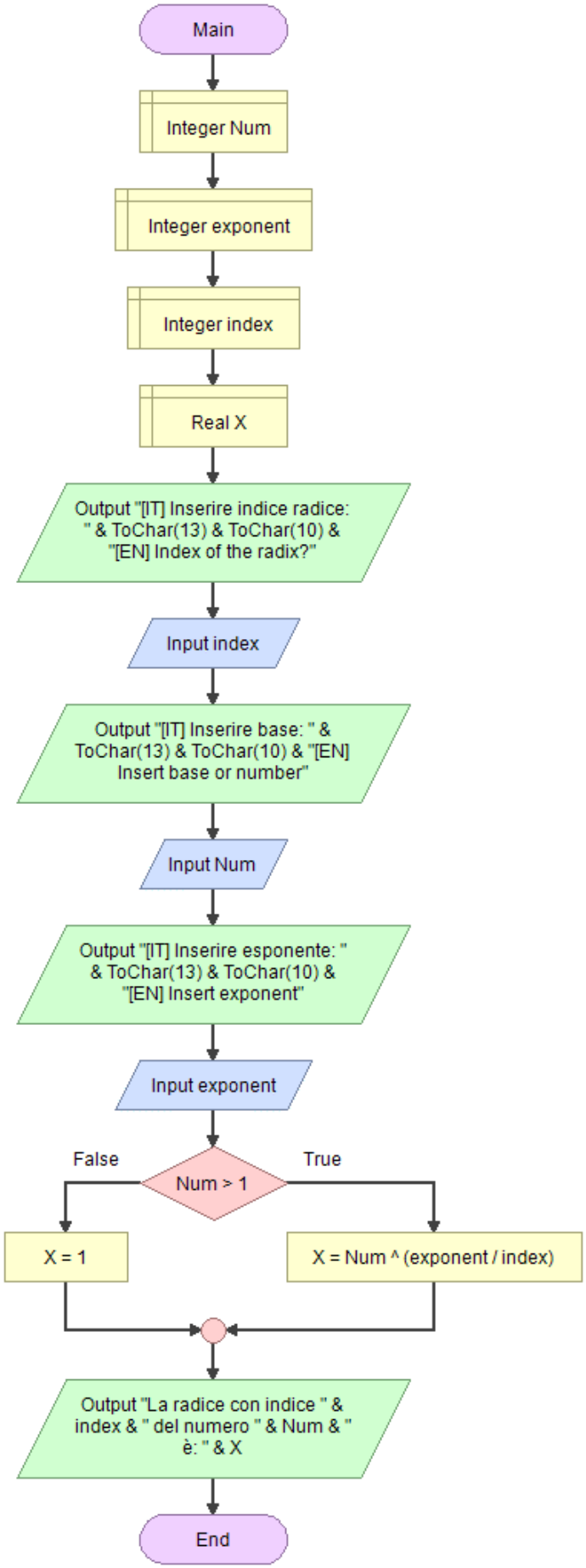
117.G ‖ Calcolo della radice quadrata di un numero

Square root of a number



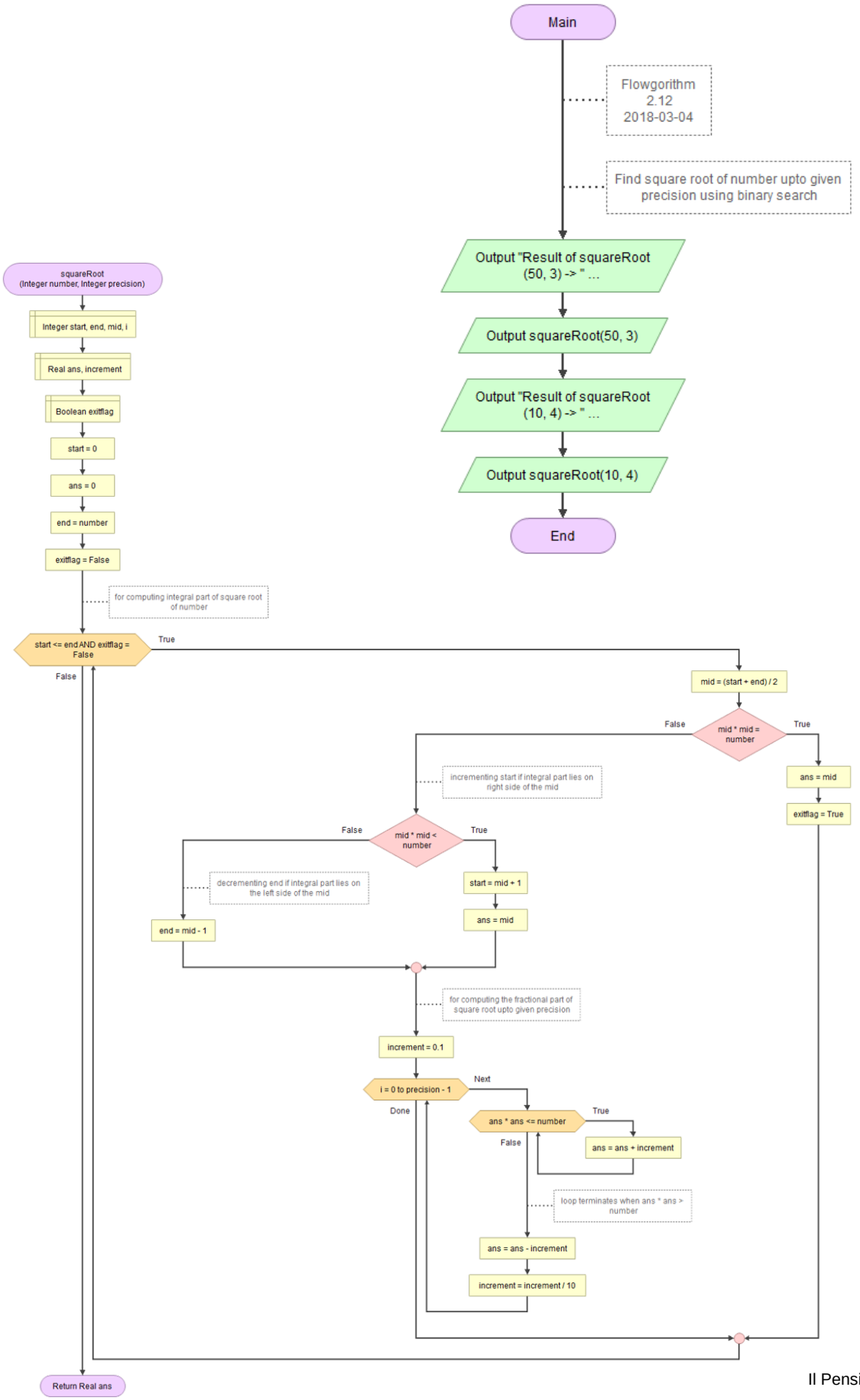
118.G ‖ Calcolo della radice n-esima di un numero

Nth root of a number



119. G F | Calcolo della radice di un numero con una data precisione usando la ricerca binaria

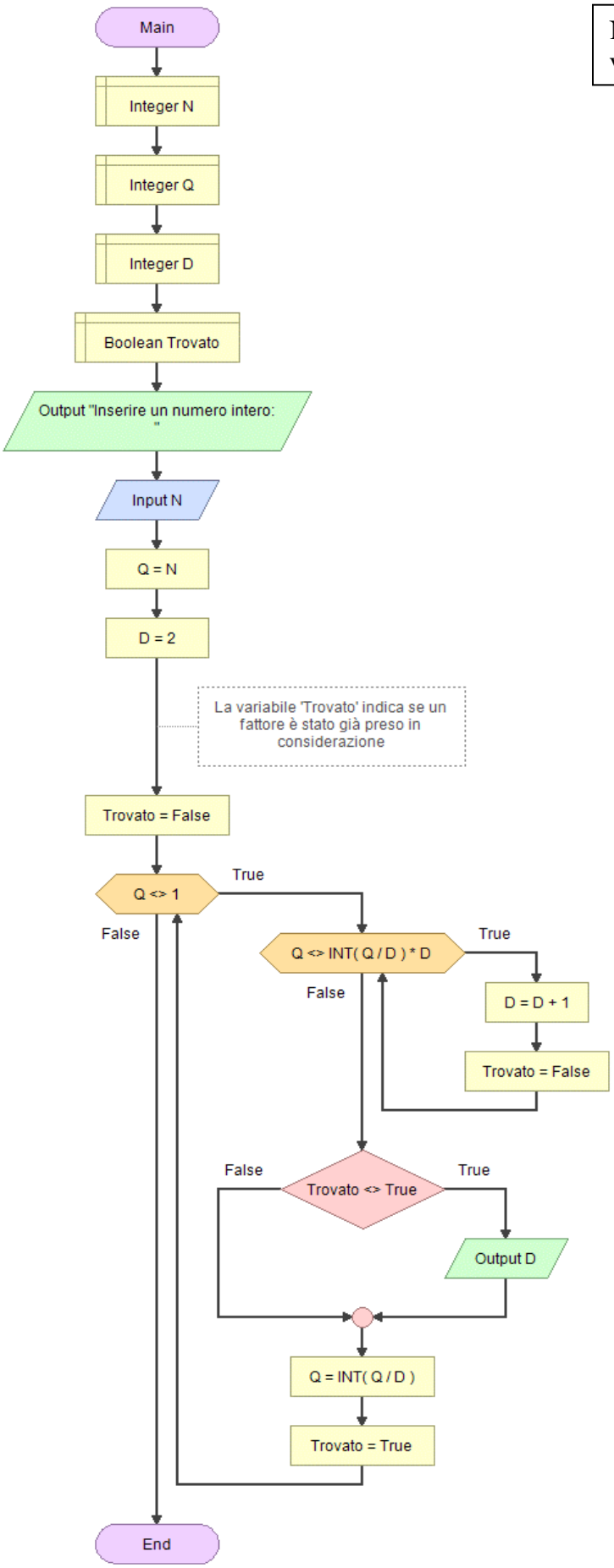
Find square root of number upto given precision using binary search



120.G ‖ Dato un intero N trovare tutti i suoi fattori primi

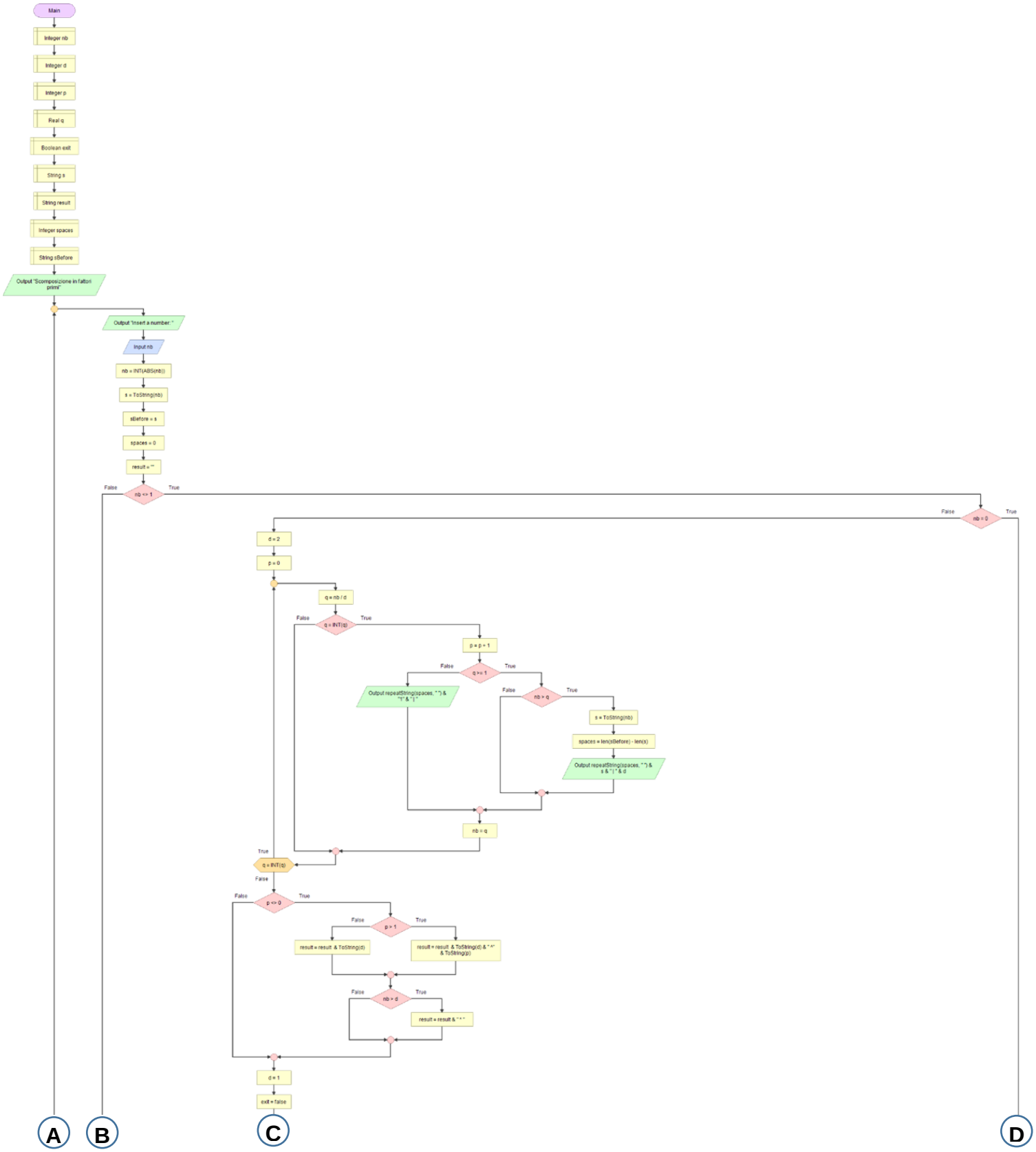
Given an integer N find all its prime factors

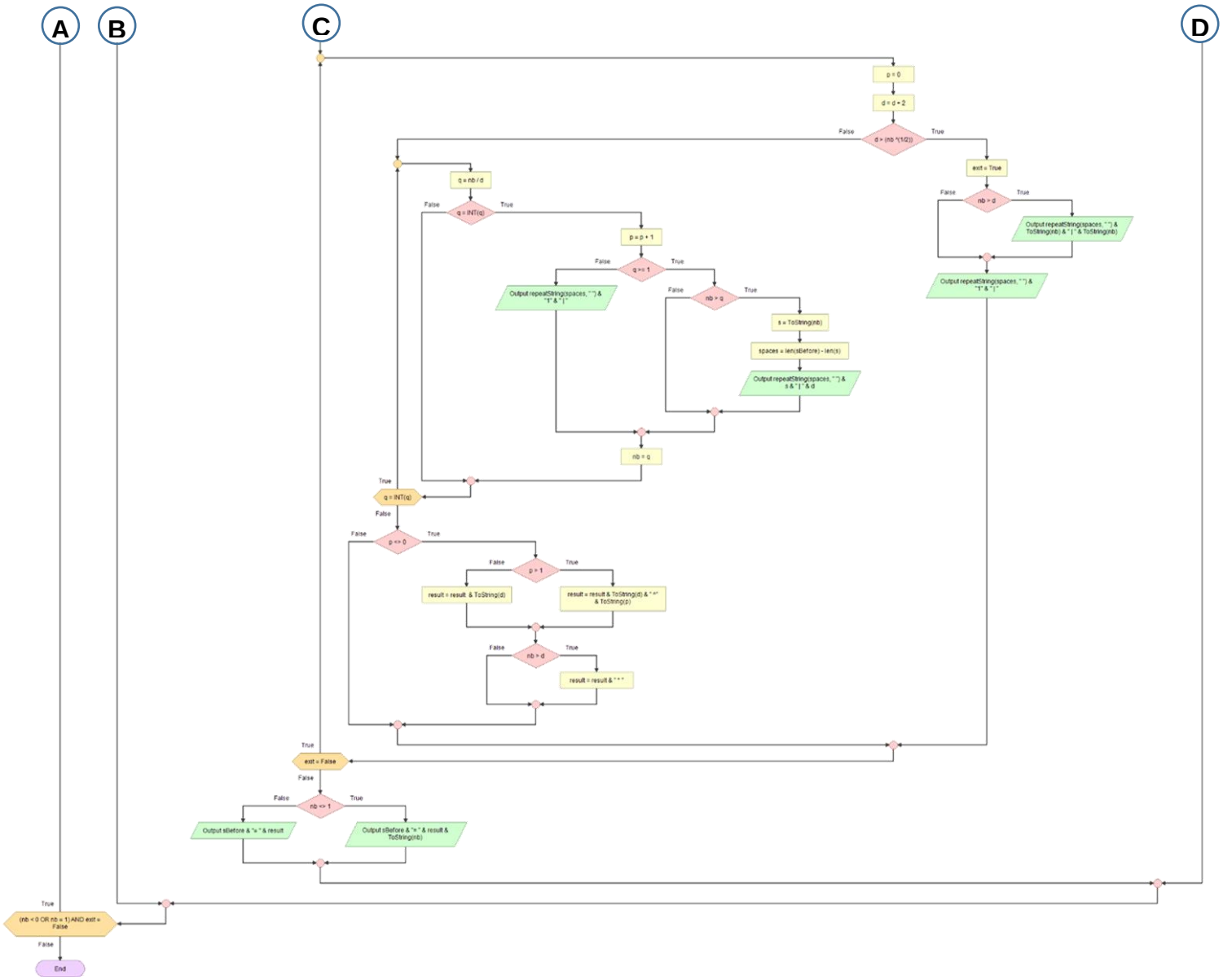
Note: l'espressione Q = (Q / D) * D se verificata mi indica che D è un divisore di Q

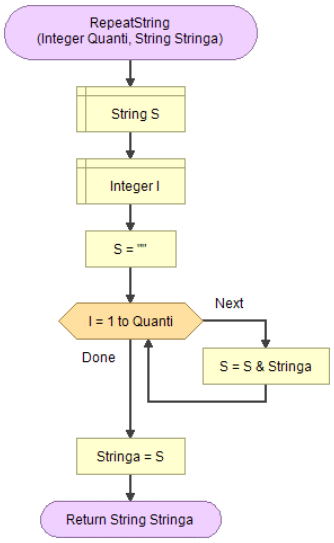


121.G F ‖ Scomposizione grafica in fattori primi

Graphical Prime Factorization

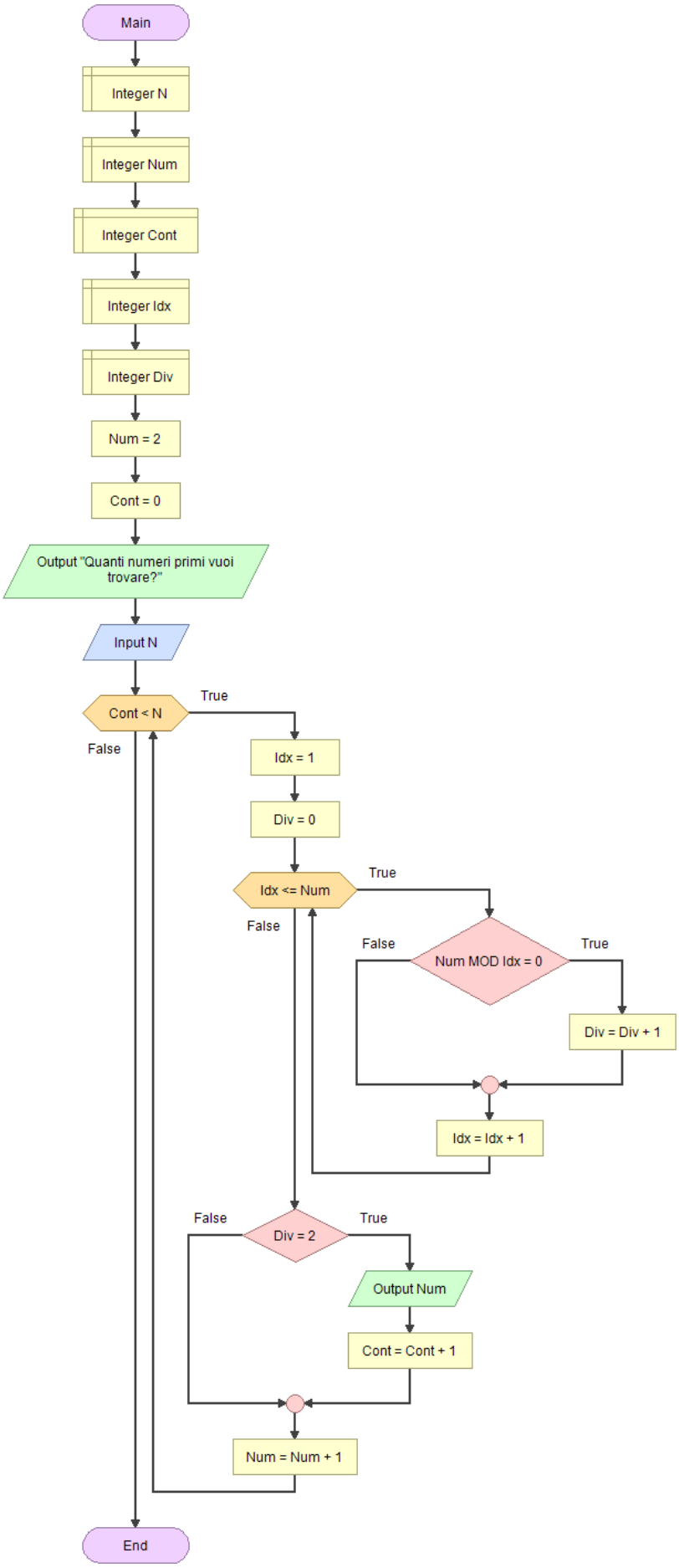



A
B
C
D
p = 0
d = d + 2
d > (nb ^(1/2))
False
True
ext = True
nb > d
Output repeatString(spaces, " ") & ToString(nb) & " | " & ToString(nb)
Output repeatString(spaces, " ") & "1" & " | "
q = nb / d
q = INT(q)
p = p + 1
q <> 1
Output repeatString(spaces, " ") & "1" & " | "
nb > q
s = ToString(nb)
spaces = len(sBefore) - len(s)
Output repeatString(spaces, " ") & s & " | " & d
nb = q
q = INT(q)
p <> 0
p > 1
result = result & ToString(d)
result = result & ToString(d) & " *" & ToString(p)
nb > d
result = result & " * "
exit = False
nb <> 1
Output sBefore & "= " & result
Output sBefore & "= " & result & ToString(nb)
(nb < 0 OR nb = 1) AND exit = False
End


RepeatString
(Integer Quanti, String Stringa)
String S
Integer I
S = ""
I = 1 to Quanti
Next
S = S & Stringa
Done
Stringa = S
Return String Stringa

122.G ‖ Visualizzare i primi N numeri primi

Display the first N prime numbers

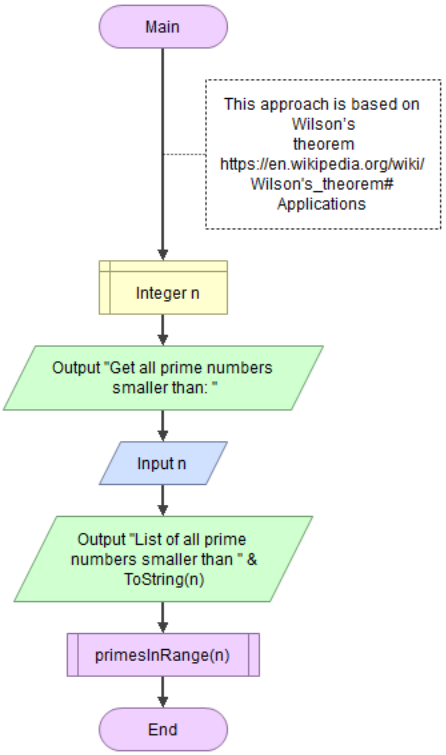


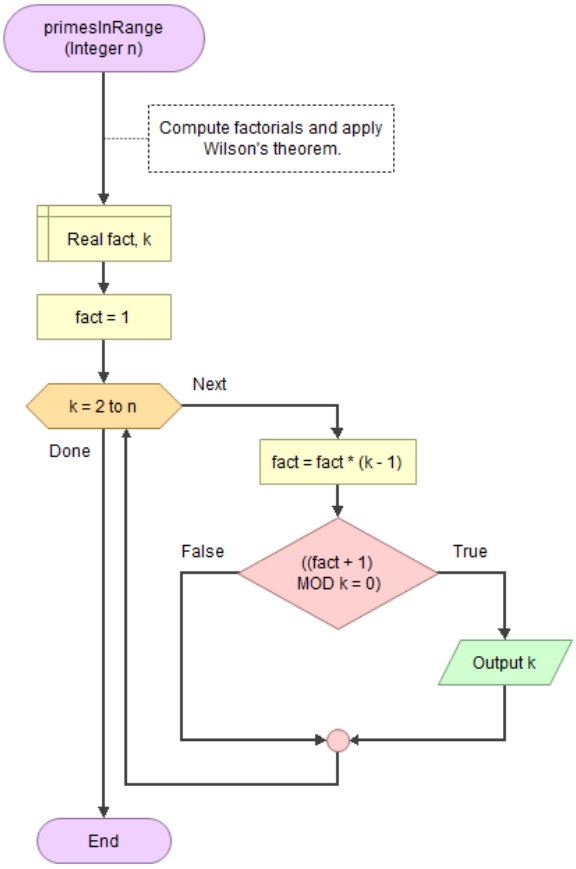
123.G ‖ Numeri primi in un intervallo

get all prime numbers smaller than n

In number theory, **Wilson's theorem** states that a natural number $n > 1$ is a prime number if and only if

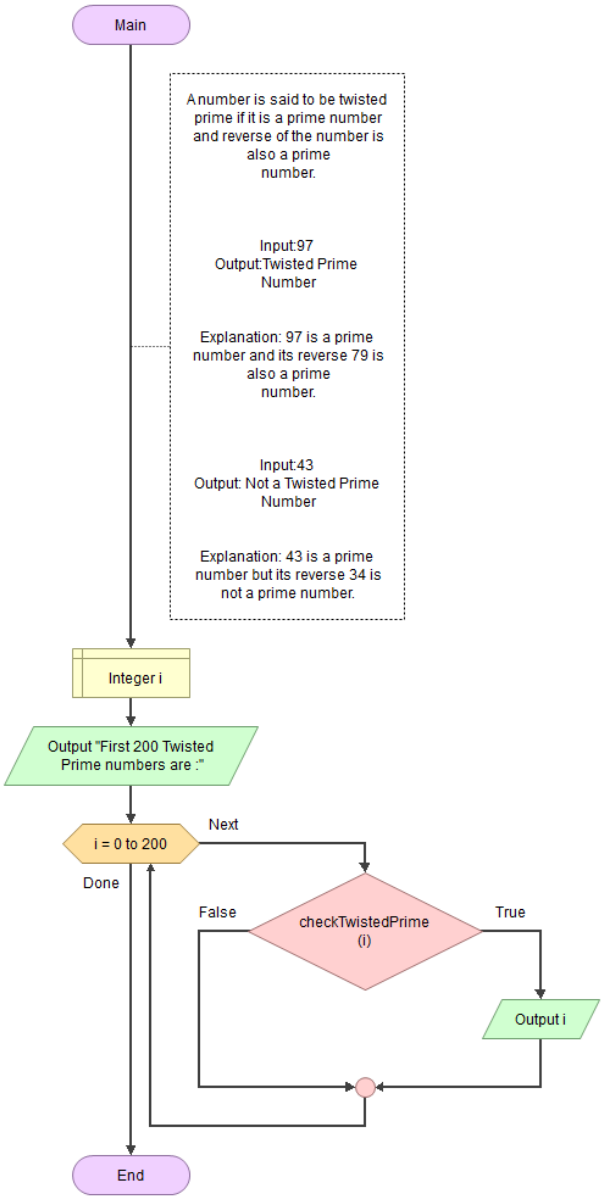
$$(n-1)! \equiv -1 \pmod{n}$$

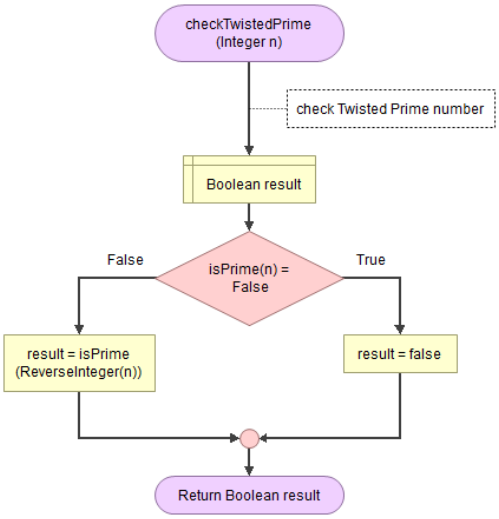


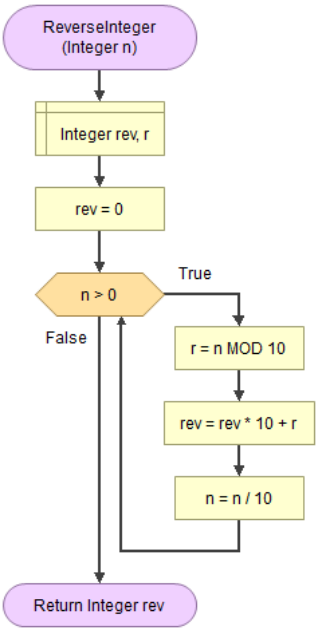


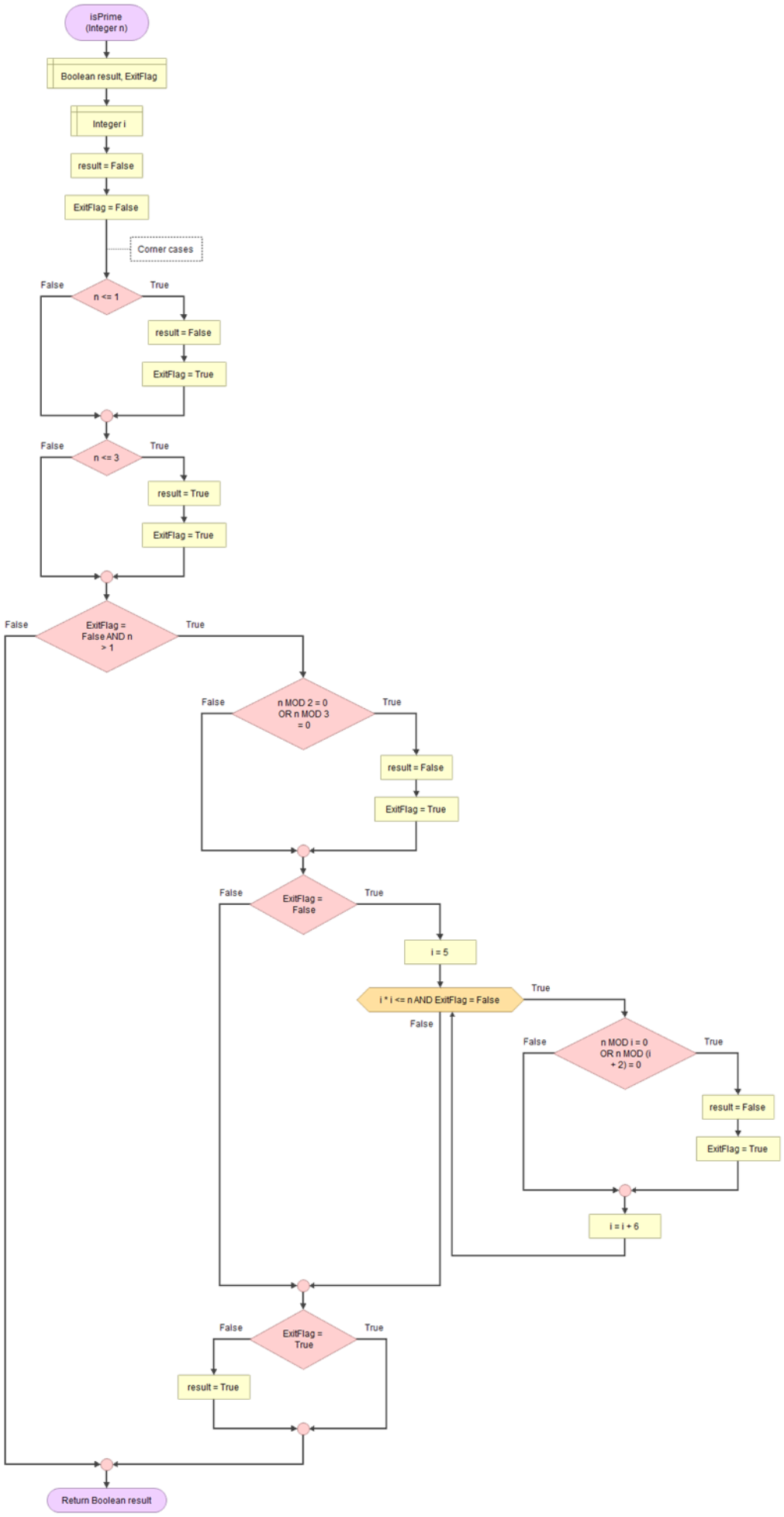
124.G F ‖ Numeri primi intrecciati

Twisted Prime Number



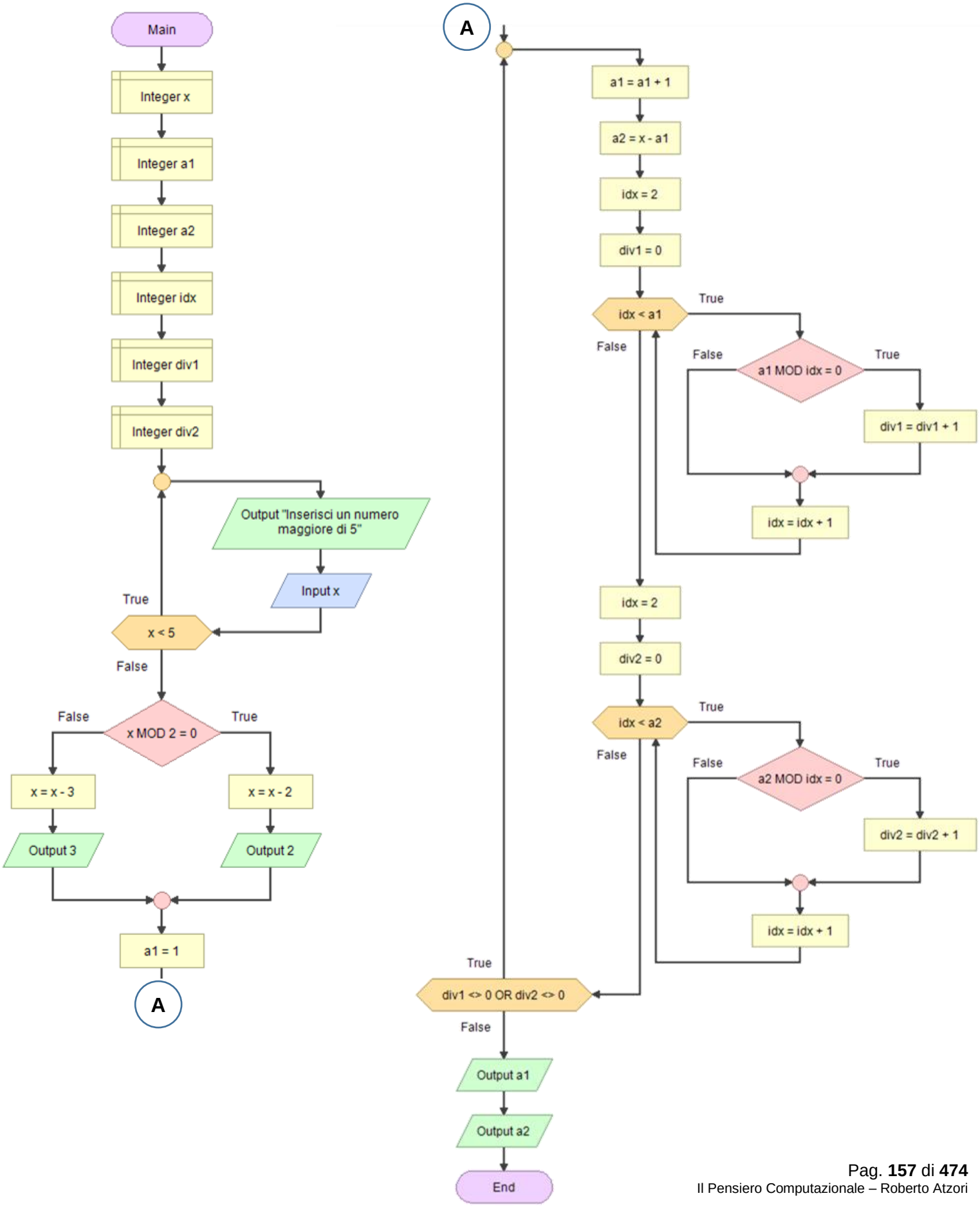





isPrime
(Integer n)
Boolean result, ExitFlag
Integer i
result = False
ExitFlag = False
Corner cases
False
n <= 1
True
result = False
ExitFlag = True
False
n <= 3
True
result = True
ExitFlag = True
False
ExitFlag = False AND n > 1
True
False
n MOD 2 = 0 OR n MOD 3 = 0
True
result = False
ExitFlag = True
False
ExitFlag = False
True
i = 5
i * i <= n AND ExitFlag = False
True
False
False
n MOD i = 0 OR n MOD (i + 2) = 0
True
result = False
ExitFlag = True
i = i + 6
False
ExitFlag = True
True
result = True
Return Boolean result

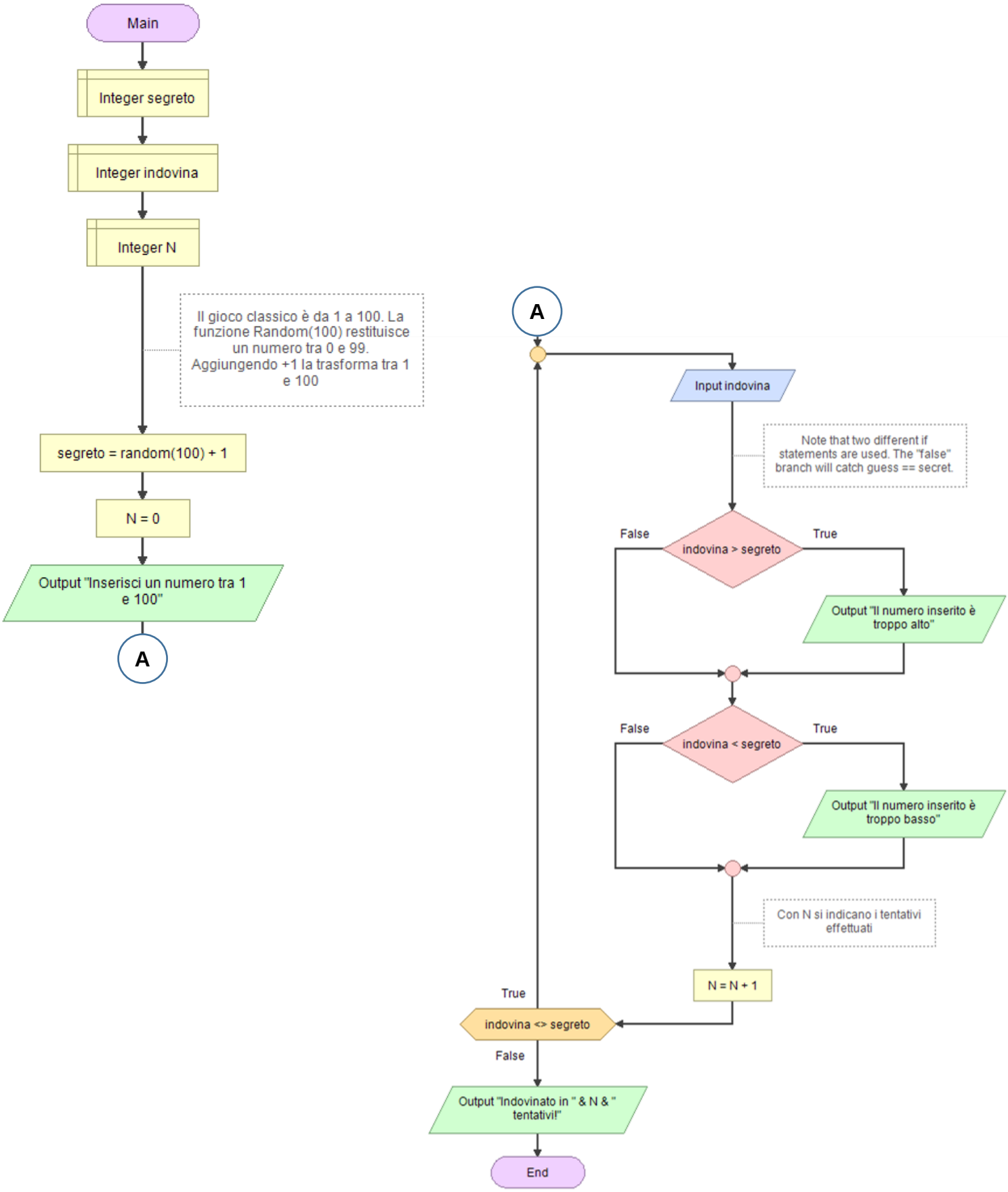
125.G ‖ Dato un numero *X* (>5) visualizzare i tre numeri primi (anche uguali) la cui somma è pari ad *X*.

Given a number X (with X greater than 5) display the first three numbers (also equal) whose sum is equal to X



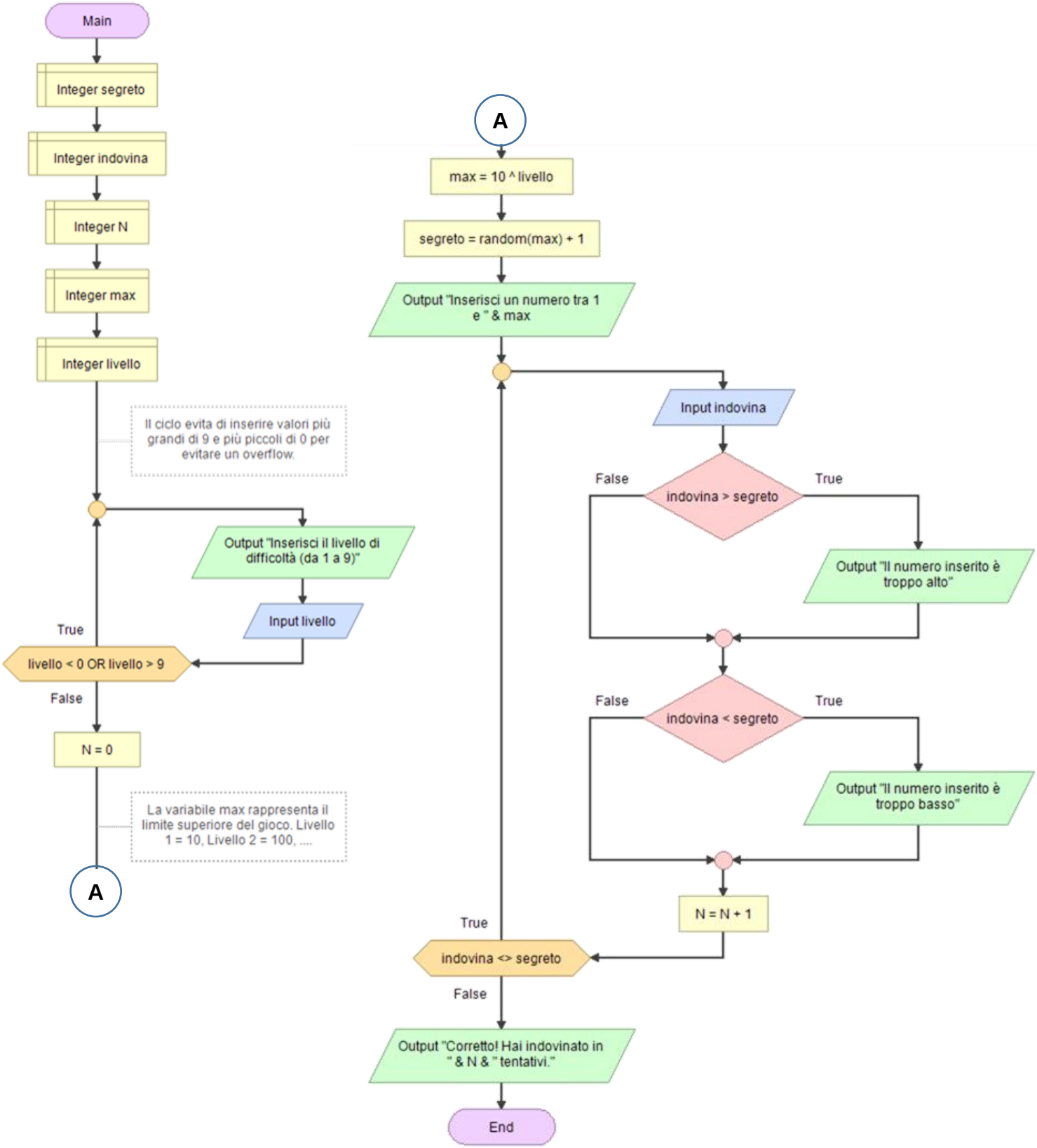
126.G ‖ Dato un numero segreto tra 0 e 100, visualizzare in quanti tentativi N l'utente ha indovinato.

Secret Number (high/low game) - Basic



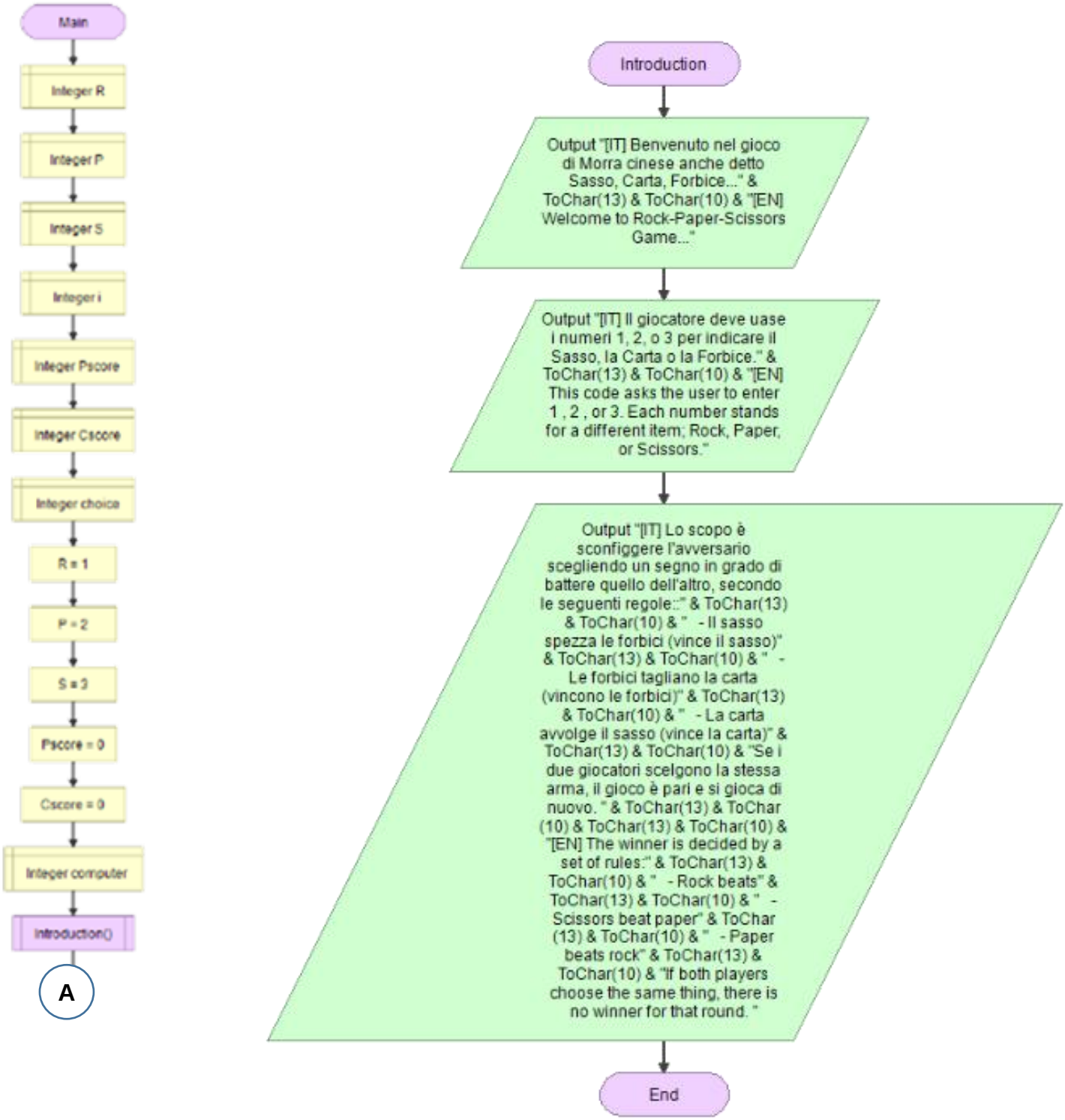
127.G ‖ Dato un numero segreto visualizzare in quanti tentativi N l'utente ha indovinato (con livelli di difficoltà).

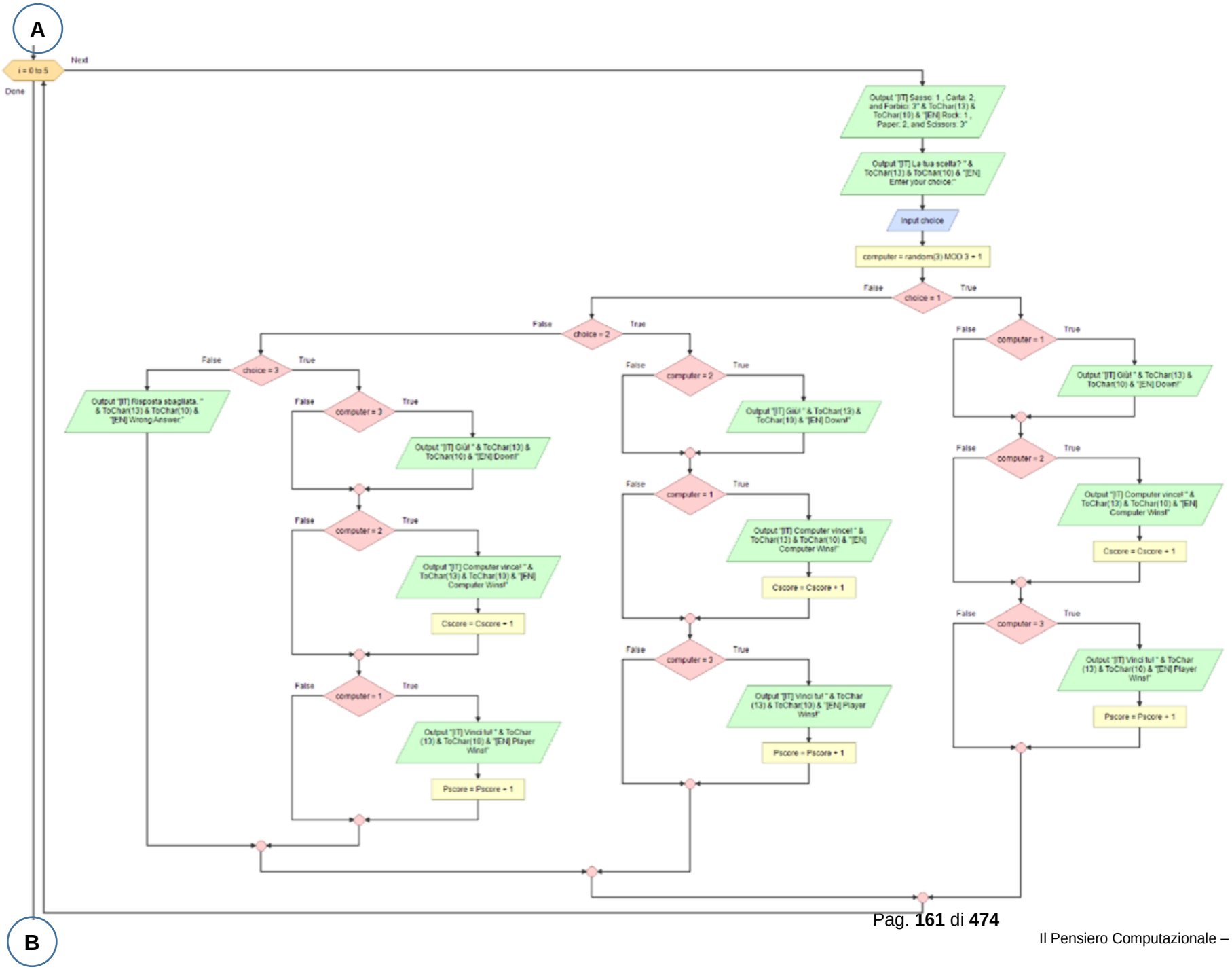
Secret Number (high/low game) - Counter, Level Selection

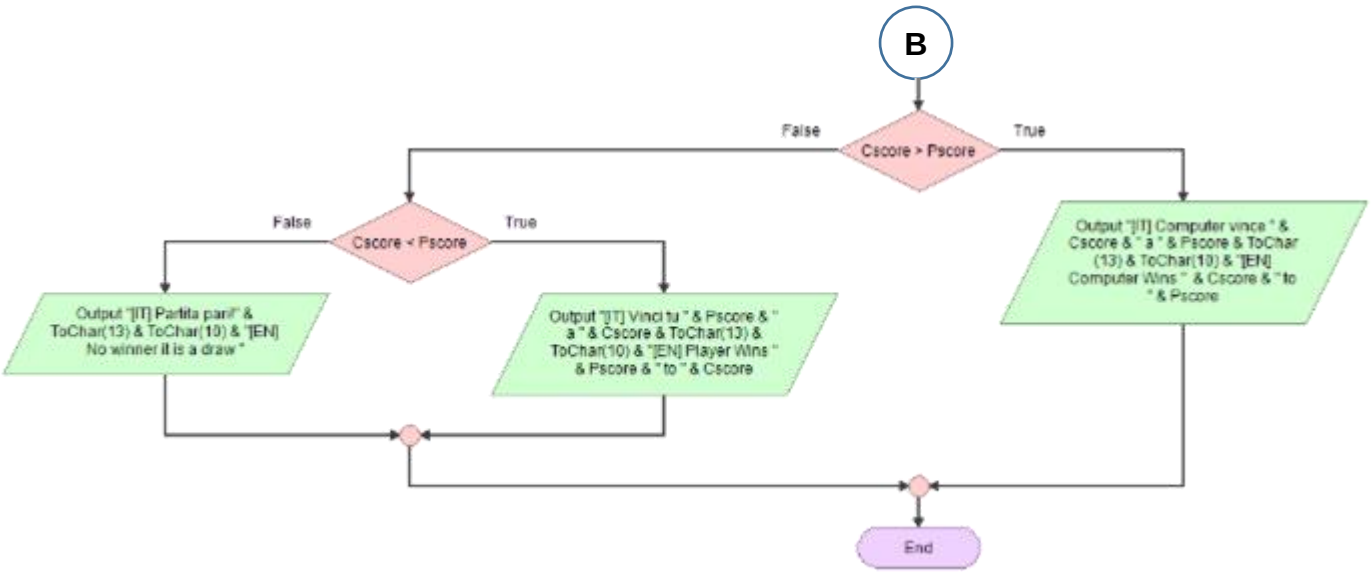


128.G ‖ Il gioco della Morra Cinese

Rock-Paper-Scissors Game

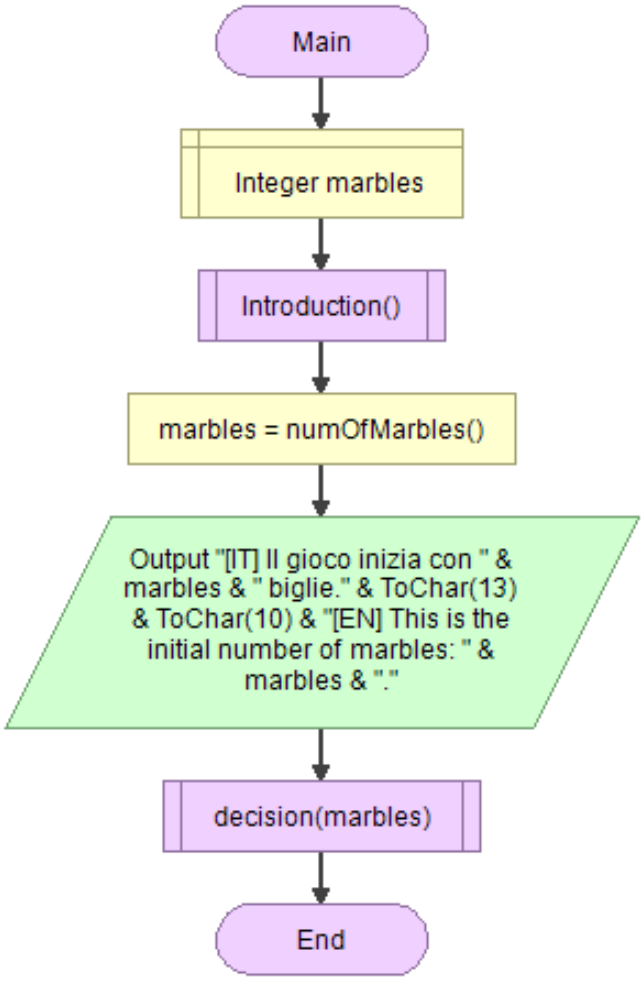


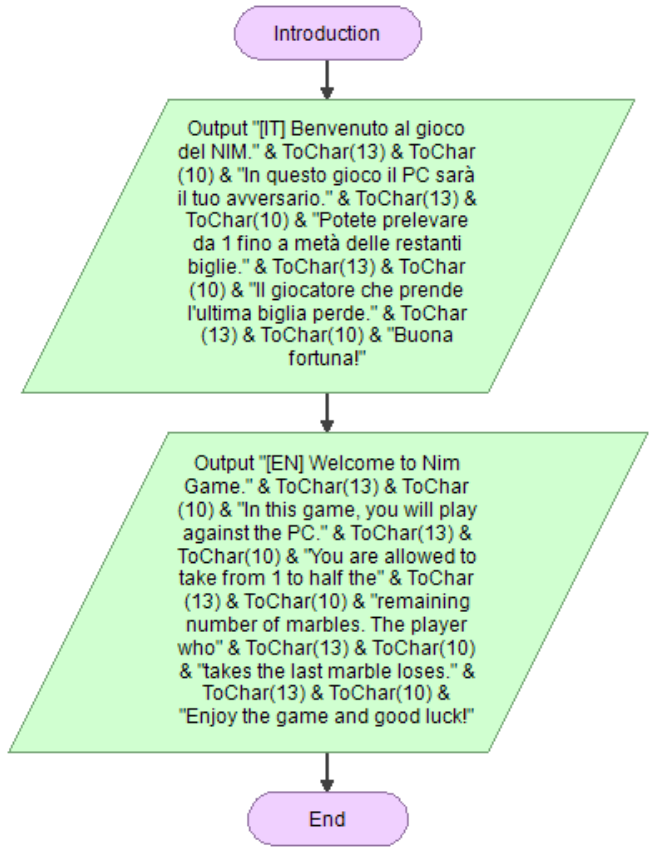

A
i = 0 to 5
Next
Done
Output "[IT] Sasso: 1 , Carta: 2, and Forbici: 3" & ToChar(13) & ToChar(10) & "[EN] Rock: 1 , Paper: 2, and Scissors: 3"
Output "[IT] La tua scelta? " & ToChar(13) & ToChar(10) & "[EN] Enter your choice:"
Input choice
computer = random(3) MOD 3 + 1
choice = 1
False
True
choice = 2
choice = 3
Output "[IT] Risposta sbagliata. " & ToChar(13) & ToChar(10) & "[EN] Wrong Answer."
computer = 3
Output "[IT] Giù! " & ToChar(13) & ToChar(10) & "[EN] Down!"
computer = 2
Output "[IT] Computer vince! " & ToChar(13) & ToChar(10) & "[EN] Computer Wins!"
Cscore = Cscore + 1
computer = 1
Output "[IT] Vinci tu! " & ToChar (13) & ToChar(10) & "[EN] Player Wins!"
Pscore = Pscore + 1
B

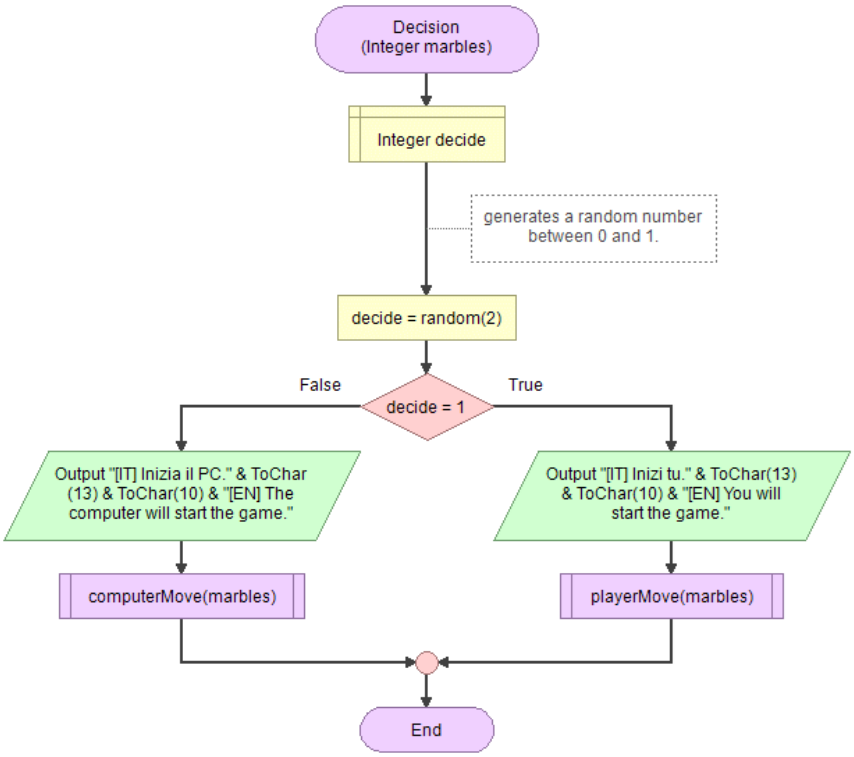

B
Cscore > Pscore
False
True
Cscore < Pscore
False
True
Output "[IT] Partita pari!" & ToChar(13) & ToChar(10) & "[EN] No winner it is a draw "
Output "[IT] Vinci tu " & Pscore & " a " & Cscore & ToChar(13) & ToChar(10) & "[EN] Player Wins " & Pscore & " to " & Cscore
Output "[IT] Computer vince " & Cscore & " a " & Pscore & ToChar (13) & ToChar(10) & "[EN] Computer Wins " & Cscore & " to " & Pscore
End

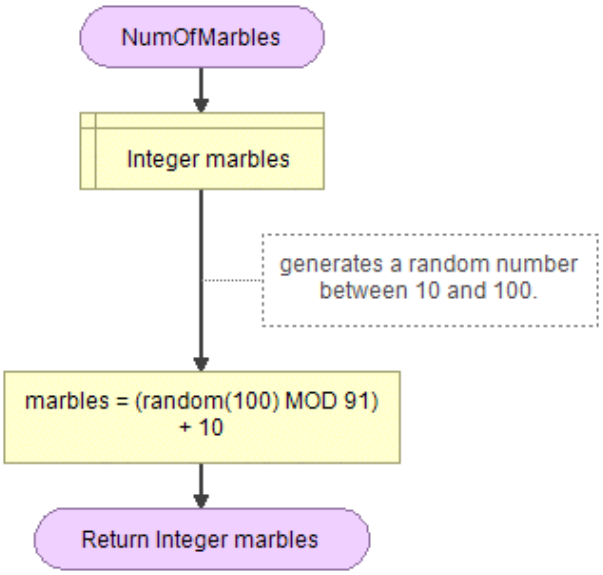
129.G F ‖ Il gioco del NIM

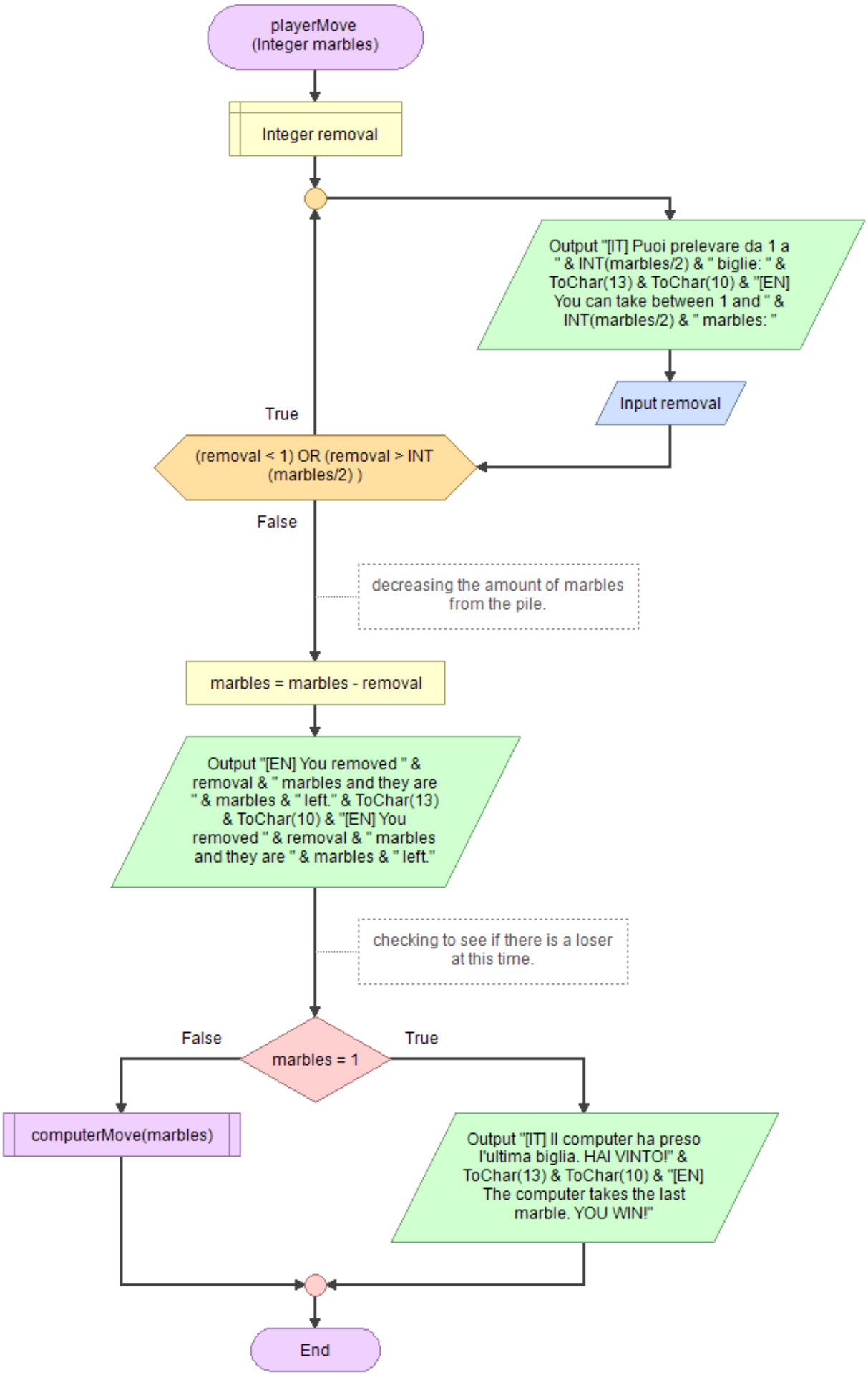
Marbles Nim Game

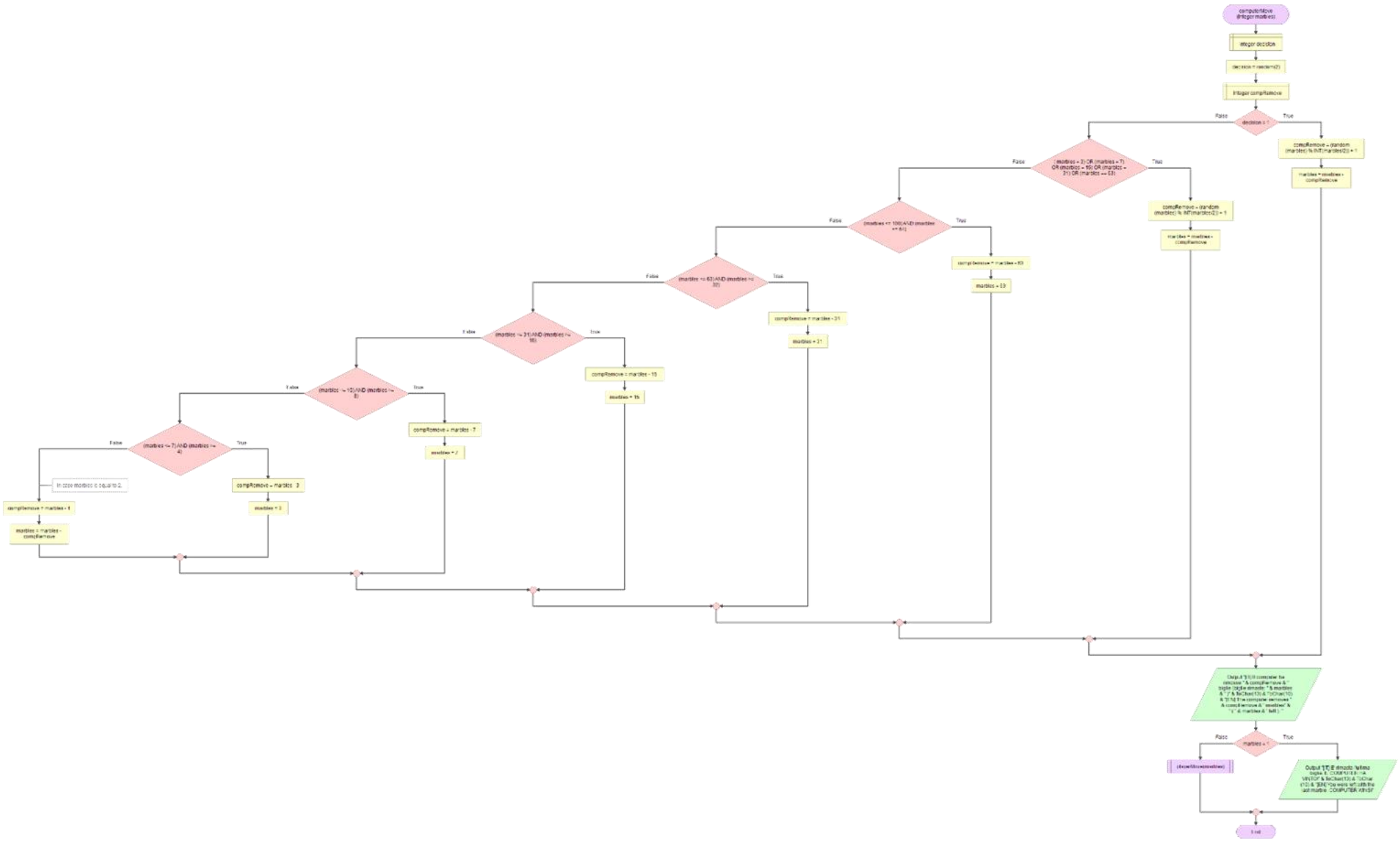






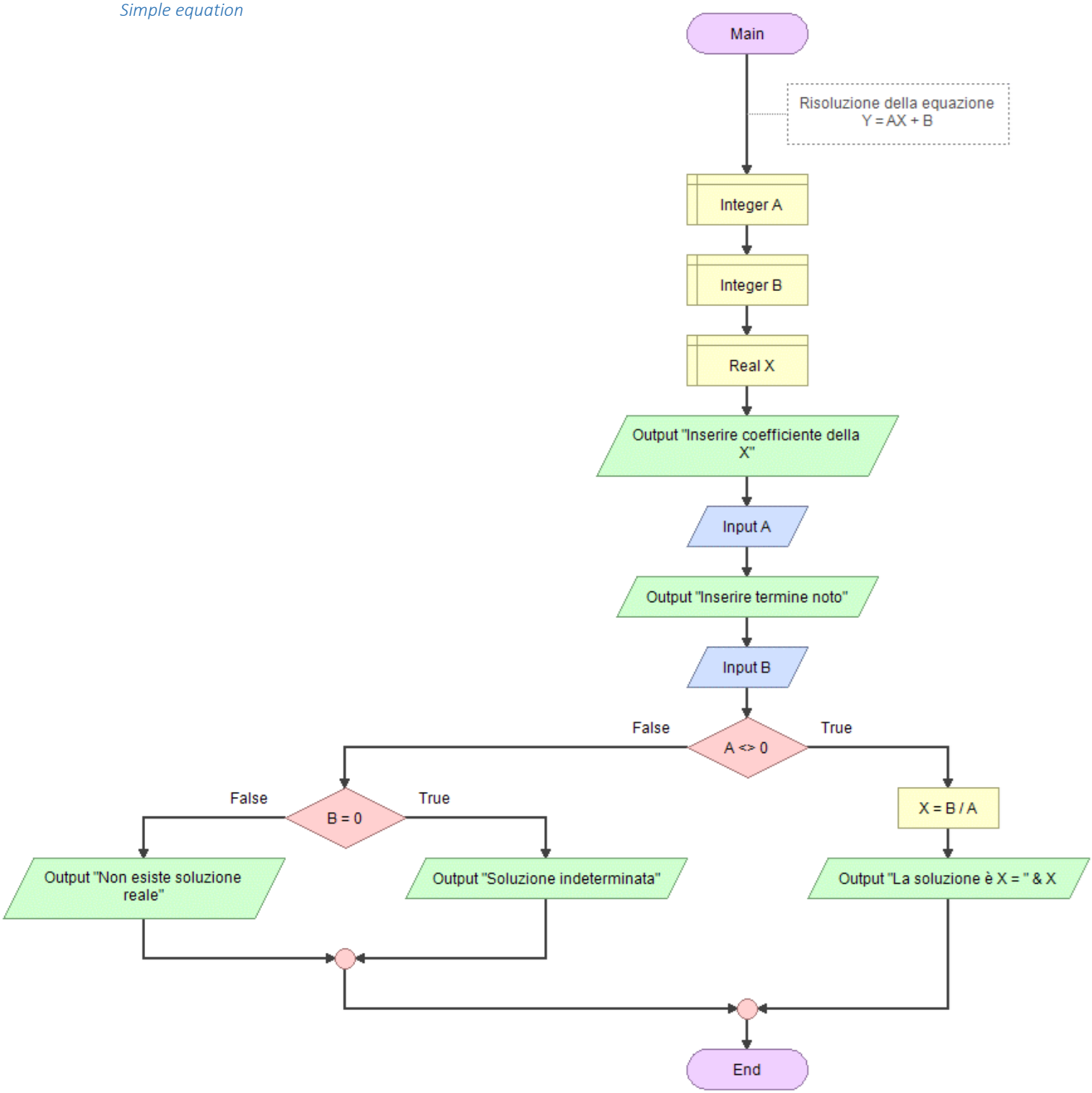



playerMove
(Integer marbles)
Integer removal
Output "[IT] Puoi prelevare da 1 a " & INT(marbles/2) & " biglie: " & ToChar(13) & ToChar(10) & "[EN] You can take between 1 and " & INT(marbles/2) & " marbles: "
Input removal
True
(removal < 1) OR (removal > INT (marbles/2))
False
decreasing the amount of marbles from the pile.
marbles = marbles - removal
Output "[EN] You removed " & removal & " marbles and they are " & marbles & " left." & ToChar(13) & ToChar(10) & "[EN] You removed " & removal & " marbles and they are " & marbles & " left."
checking to see if there is a loser at this time.
False
marbles = 1
True
computerMove(marbles)
Output "[IT] Il computer ha preso l'ultima biglia. HAI VINTO!" & ToChar(13) & ToChar(10) & "[EN] The computer takes the last marble. YOU WIN!"
End


computerMove (integer marbles)
integer decision
integer compRemove
decision = 1
False
True
compRemove = random (marbles) % INT(marbles/2) + 1
marbles = marbles - compRemove
(marbles = 3) OR (marbles = 7) OR (marbles = 15) OR (marbles = 31) OR (marbles = 63)
compRemove = random (marbles) % INT(marbles/2) + 1
marbles = marbles - compRemove
(marbles <= 100) AND (marbles >= 64)
compRemove = marbles - 63
marbles = 63
(marbles <= 63) AND (marbles >= 32)
compRemove = marbles - 31
marbles = 31
(marbles <= 31) AND (marbles >= 16)
compRemove = marbles - 15
marbles = 15
(marbles <= 15) AND (marbles >= 8)
compRemove = marbles - 7
marbles = 7
(marbles <= 7) AND (marbles >= 4)
compRemove = marbles - 3
marbles = 3
In caso marbles è uguale a 2.
compRemove = marbles - 1
marbles = marbles - compRemove
marbles = 1
(decrementa marbles)
End

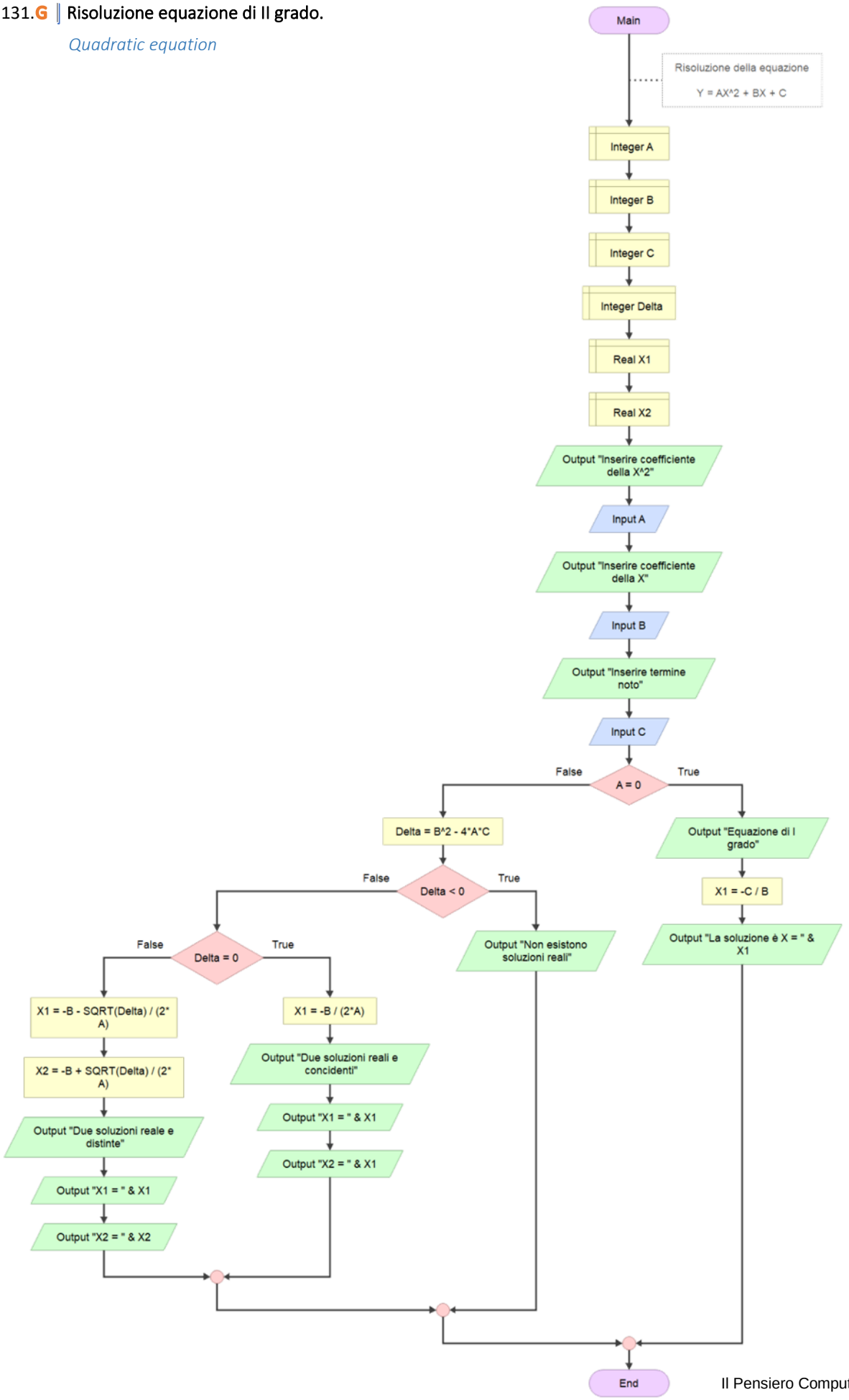
130.G ‖ Risoluzione equazione di I grado

Simple equation



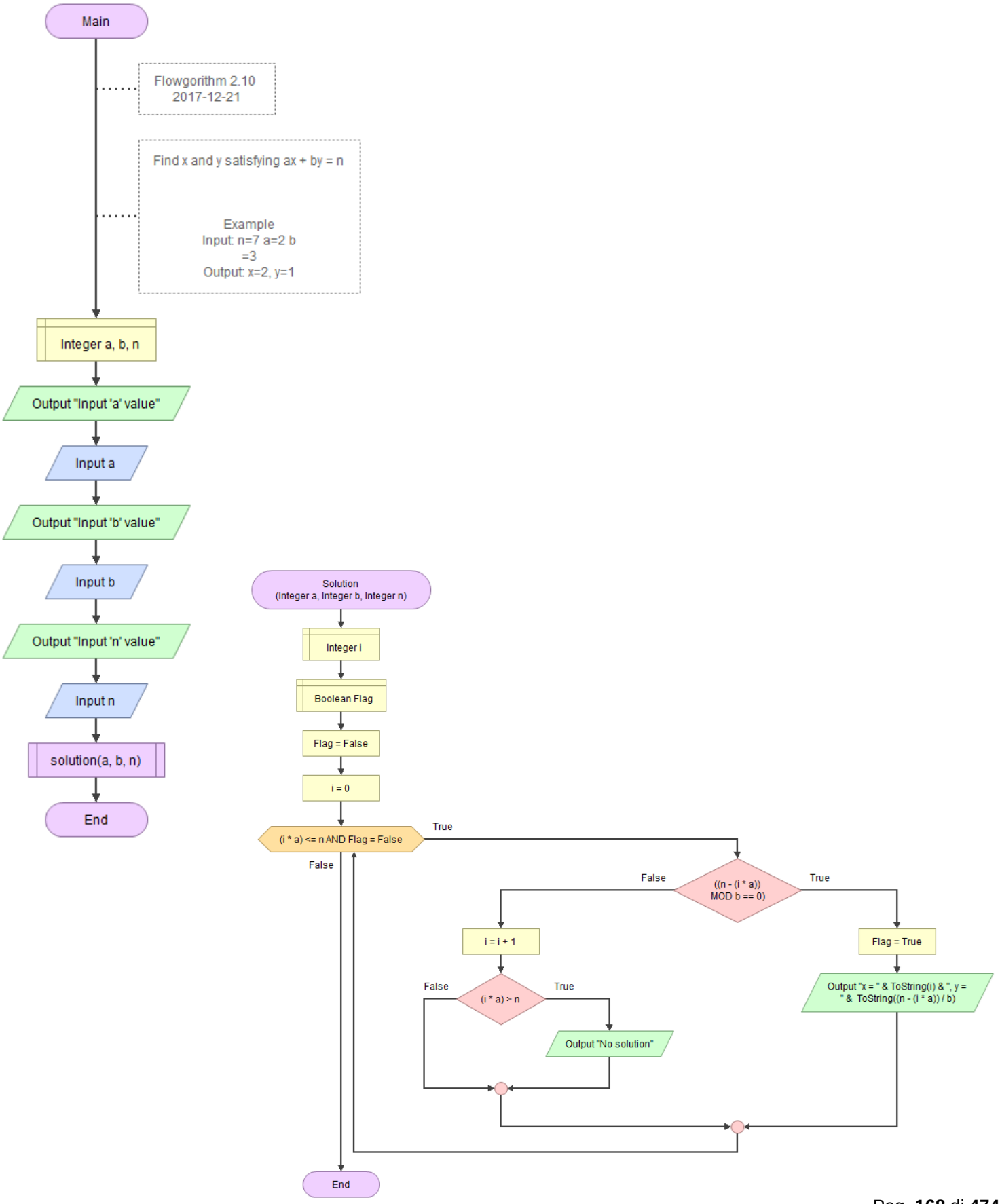
131.G ‖ Risoluzione equazione di II grado.

Quadratic equation



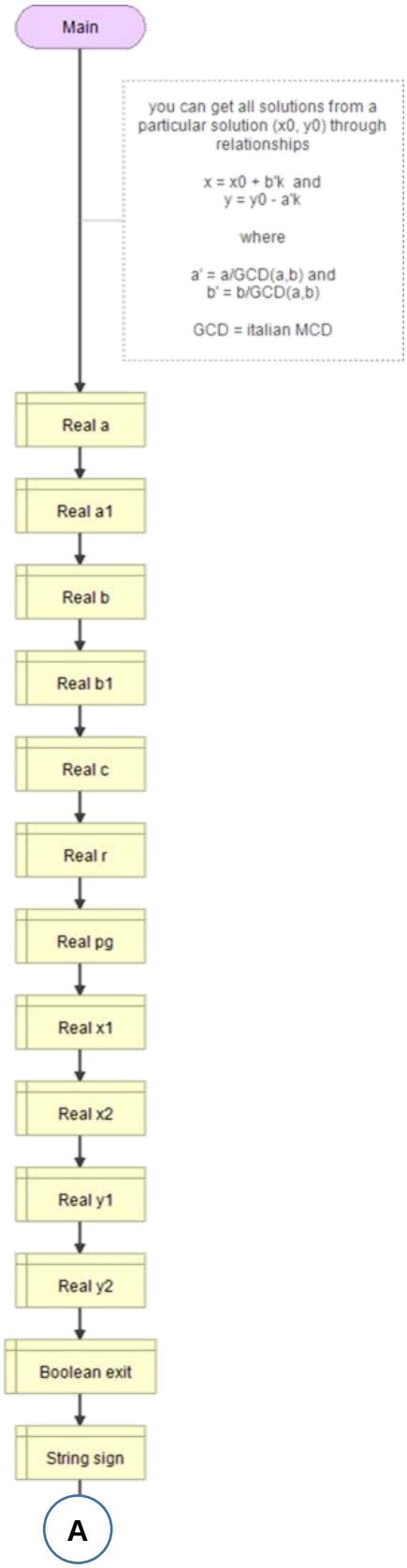
132. **G F** | Trovare x e y che soddisfino l'equazione ax + by = c

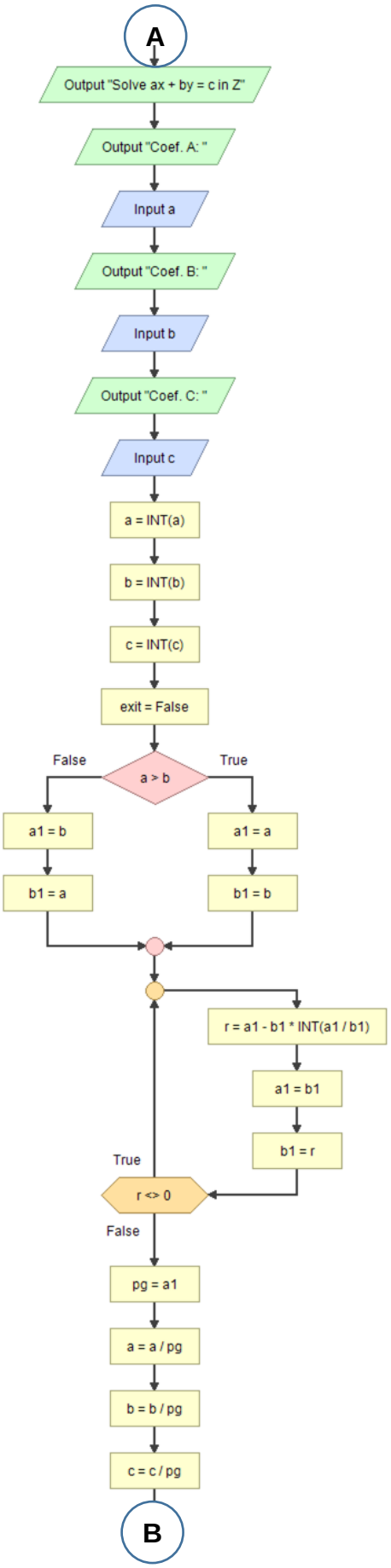
Find x and y satisfying ax + by = n

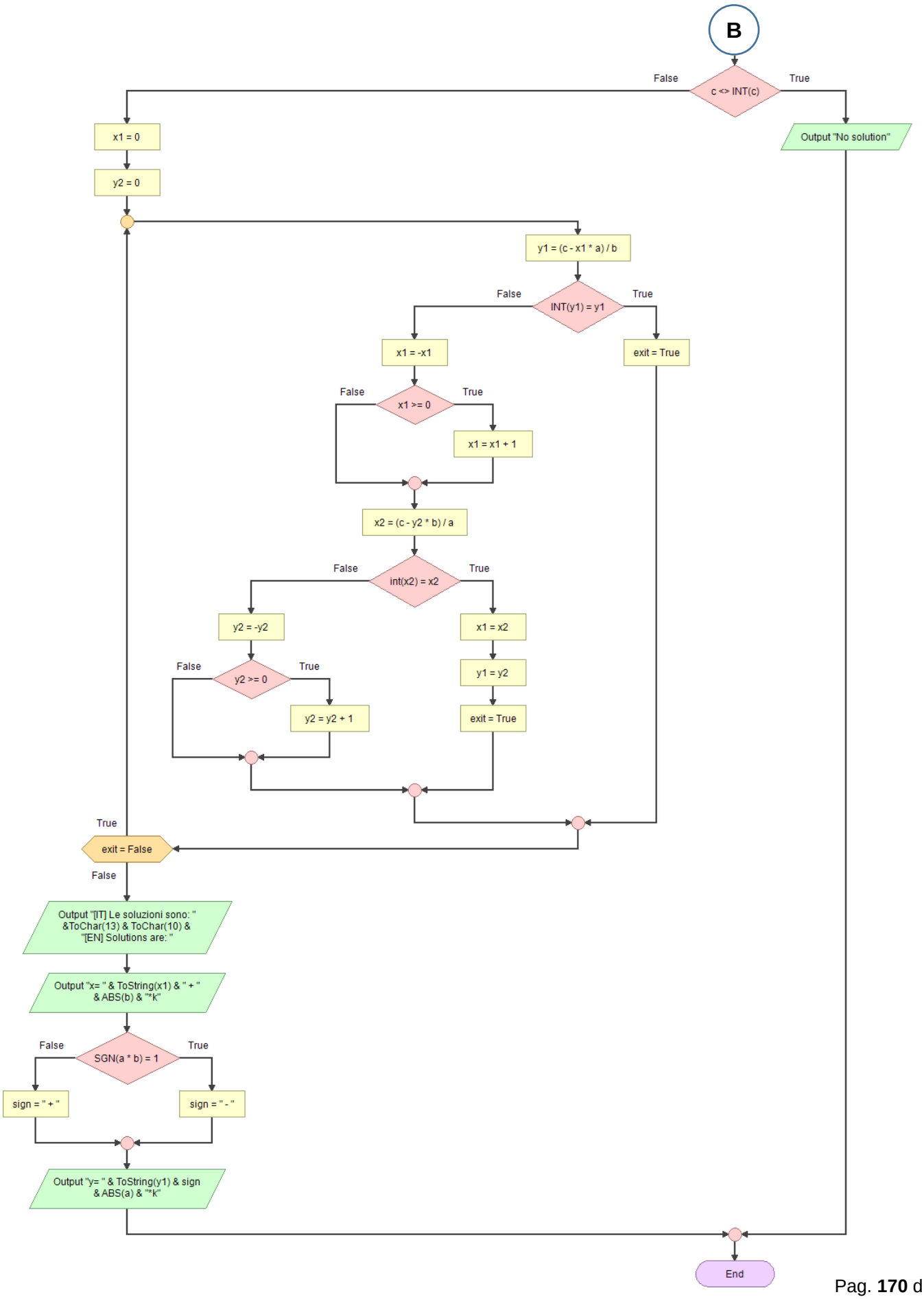


133. G F ‖ Risoluzione in Z dell'equazione ax + by = c

Solve ax + by = c in Z

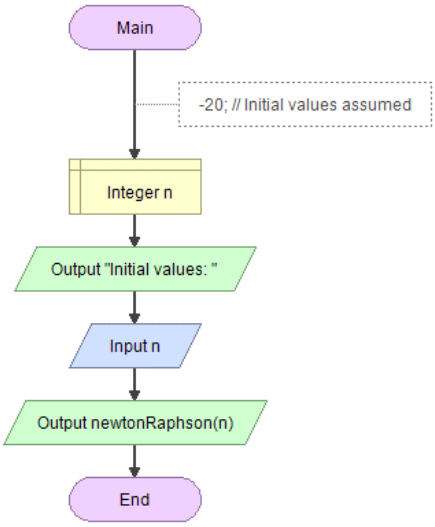





B
c <> INT(c)
False
True
Output "No solution"
x1 = 0
y2 = 0
y1 = (c - x1 * a) / b
INT(y1) = y1
False
True
x1 = -x1
exit = True
x1 >= 0
False
True
x1 = x1 + 1
x2 = (c - y2 * b) / a
int(x2) = x2
False
True
y2 = -y2
x1 = x2
y2 >= 0
False
True
y1 = y2
y2 = y2 + 1
exit = True
True
exit = False
False
Output "[IT] Le soluzioni sono: " &ToChar(13) & ToChar(10) & "[EN] Solutions are: "
Output "x= " & ToString(x1) & " + " & ABS(b) & "*k"
SGN(a * b) = 1
False
True
sign = " + "
sign = " - "
Output "y= " & ToString(y1) & sign & ABS(a) & "*k"
End

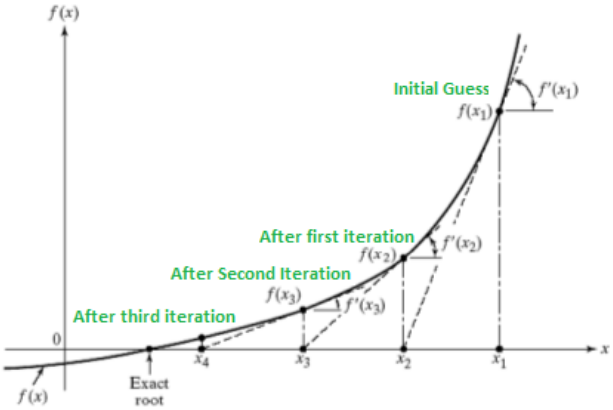
134.G F ‖ Soluzione di una equazione algebrica trascendente (metodo Newton Raphson)

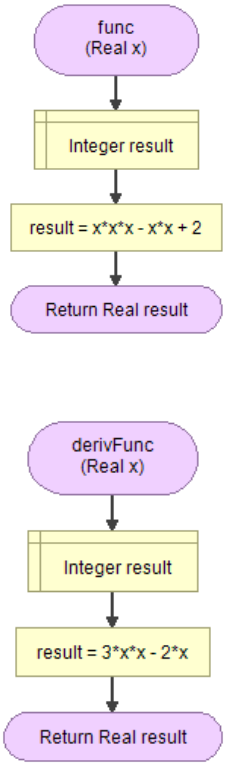
Solution of Algebraic and Transcendental Equations (Newton Raphson Method)

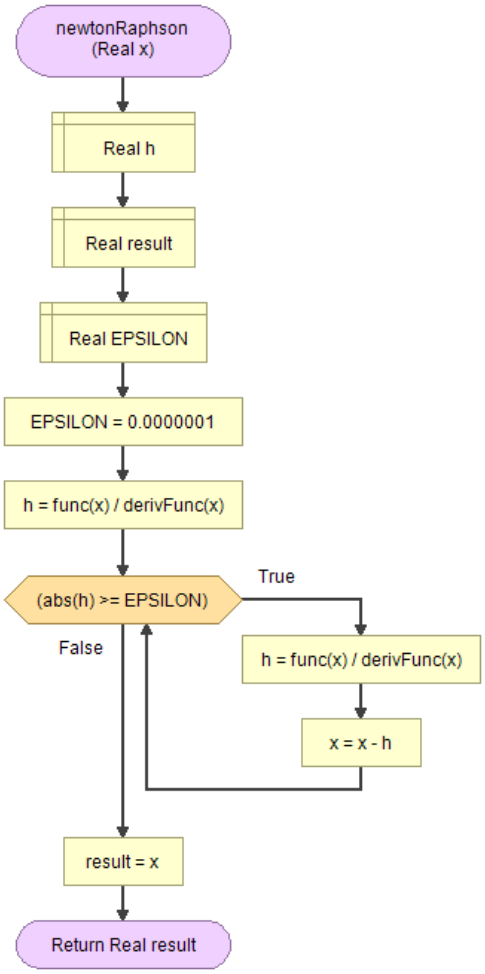


Starting from initial guess x_1, the Newton Raphson method uses below formula to find next value of x, i.e., x_{n+1} from previous value x_n.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

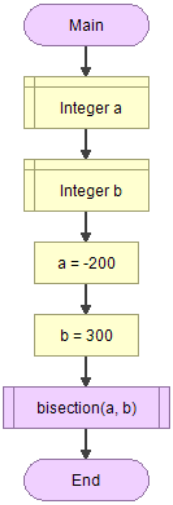


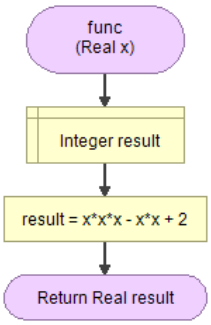




135.G F | Soluzione di una equazione algebrica trascendente (metodo della bisezione)

Solution of Algebraic and Transcendental Equations (Bisection Method)

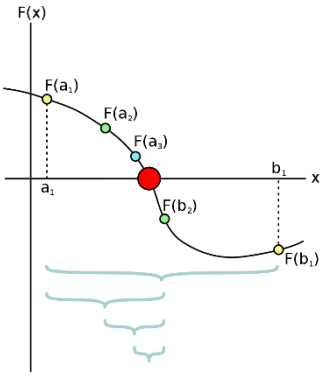


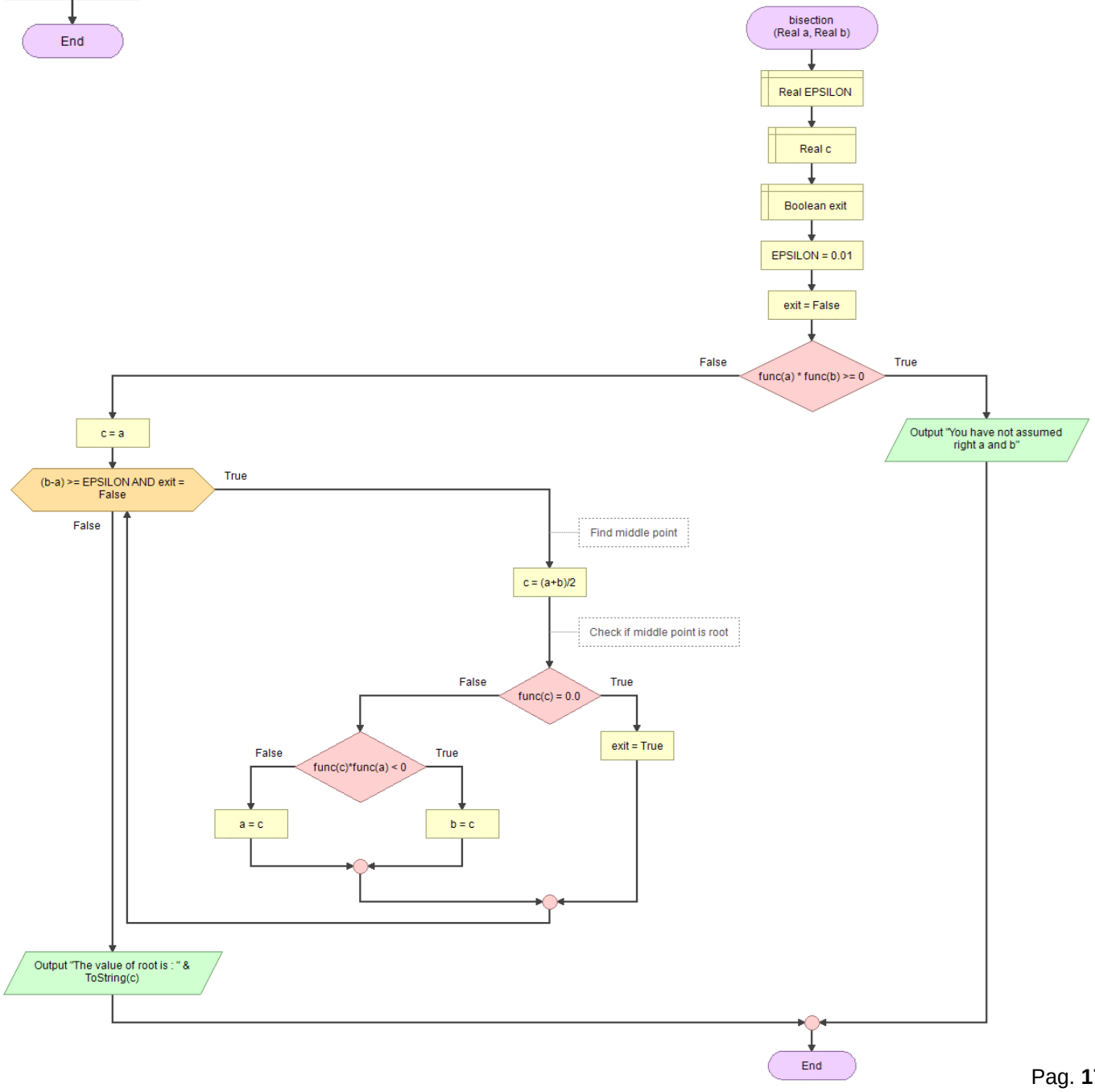


What is Bisection Method?

The method is also called the interval halving method, the binary search method or the dichotomy method. This method is used to find root of an equation in a given interval that is value of 'x' for which f(x) = 0 .

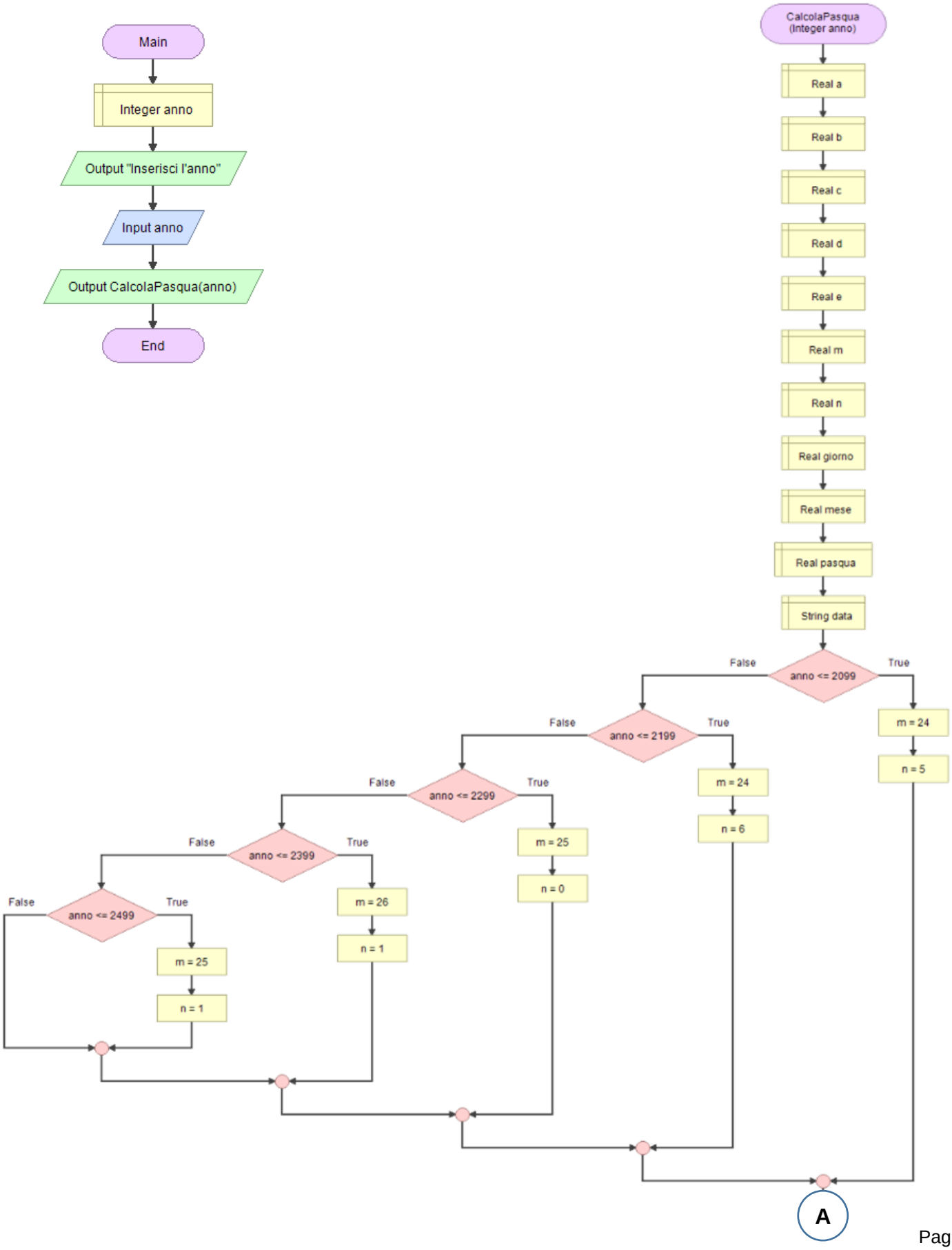
The method is based on **The Intermediate Value Theorem** which states that if f(x) is a continuous function and there are two real numbers a and b such that f(a)*f(b) < 0. Then f(x) has at least one zero between a and b. If for a function (f(a) < 0 and f(b) > 0) or (f(a) > 0 and f(b) < 0), then it is guaranteed that it has at least one root between them.

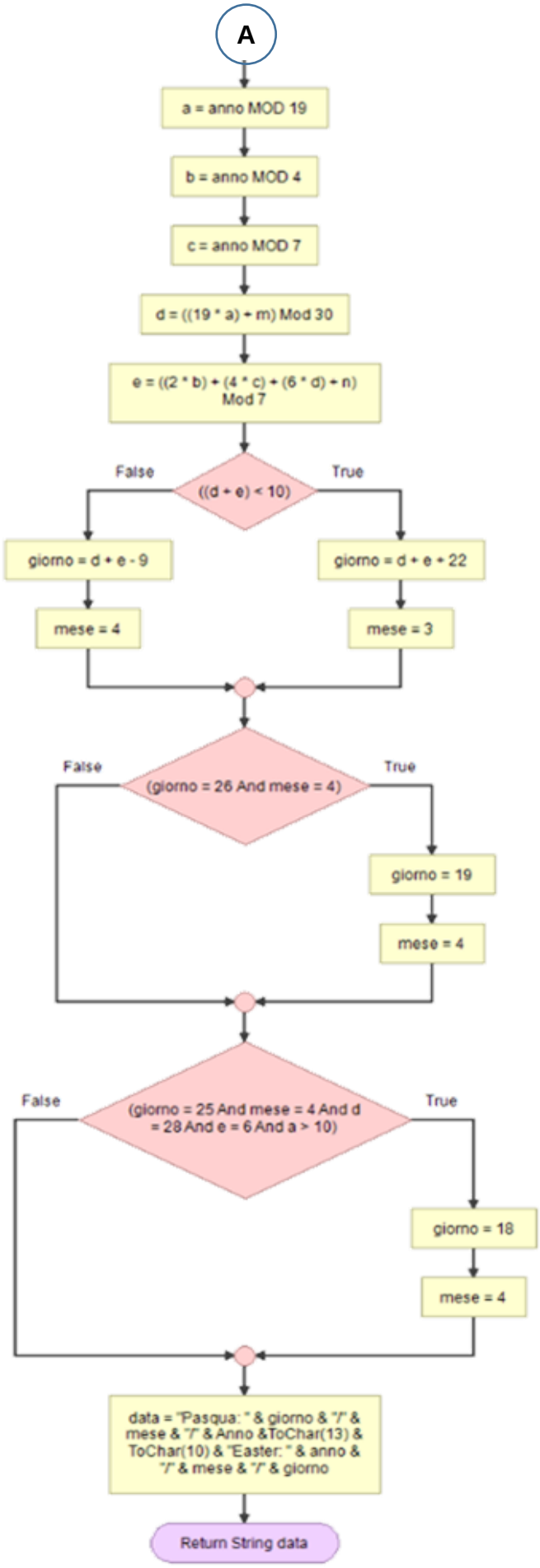




136. G F | Calcolo del giorno di Pasqua

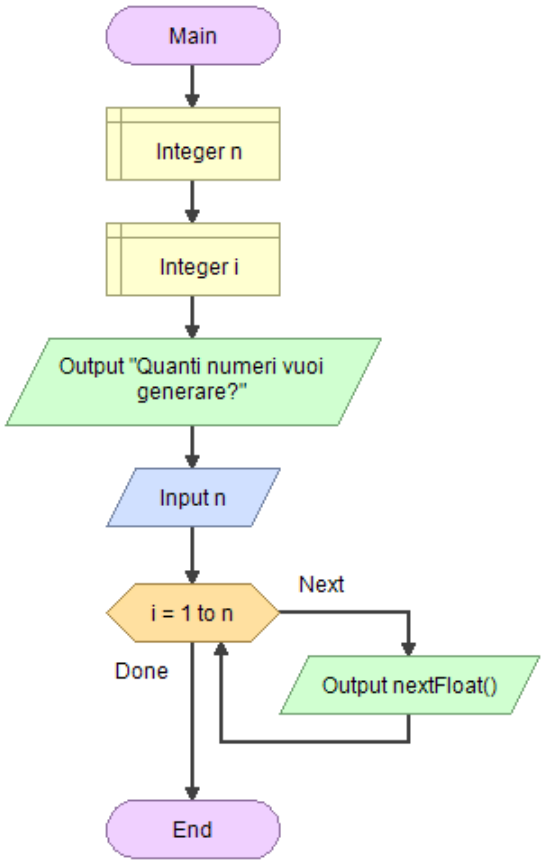
Easter Day calculus

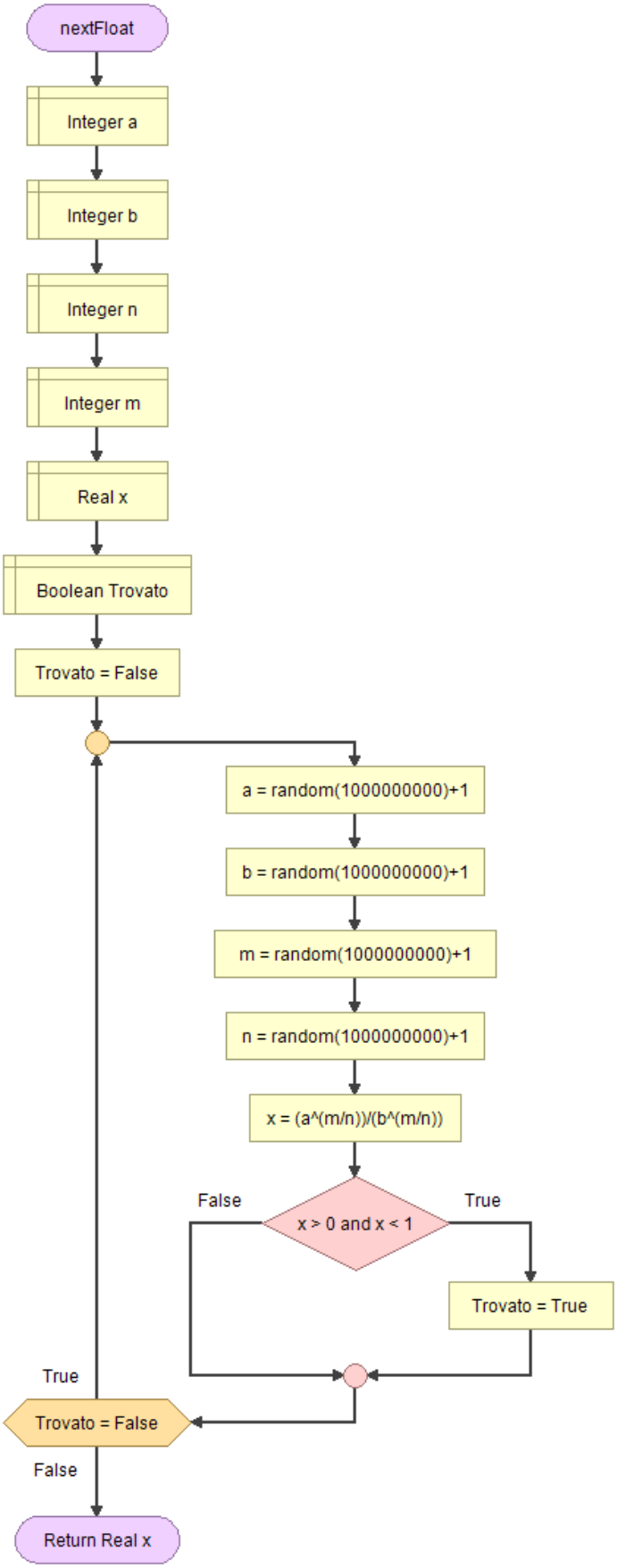



A
a = anno MOD 19
b = anno MOD 4
c = anno MOD 7
d = ((19 * a) + m) Mod 30
e = ((2 * b) + (4 * c) + (6 * d) + n) Mod 7
False
((d + e) < 10)
True
giorno = d + e - 9
giorno = d + e + 22
mese = 4
mese = 3
False
(giorno = 26 And mese = 4)
True
giorno = 19
mese = 4
False
(giorno = 25 And mese = 4 And d = 28 And e = 6 And a > 10)
True
giorno = 18
mese = 4
data = "Pasqua: " & giorno & "/" & mese & "/" & Anno &ToChar(13) & ToChar(10) & "Easter: " & anno & "/" & mese & "/" & giorno
Return String data

137.G F ‖ Generatore Numeri reali Pseudo Casuali maggiori di 0 e minori di 1

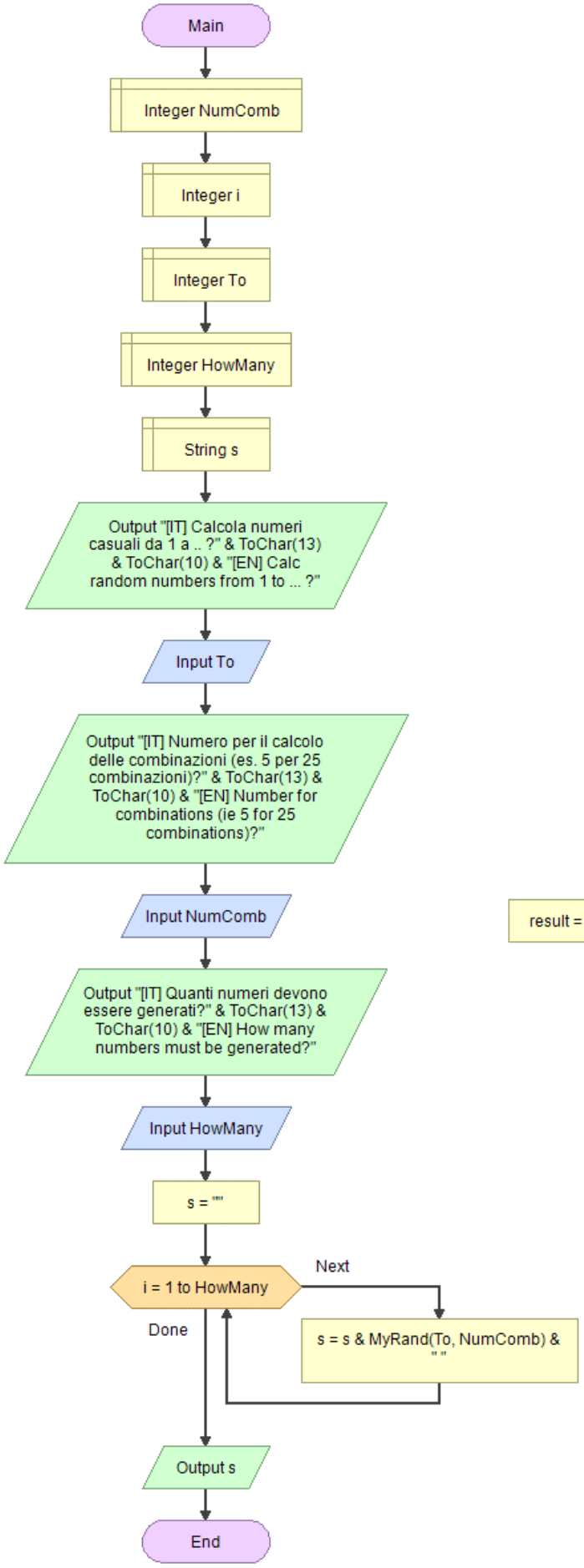
Random real numbers generation between 0 and 1

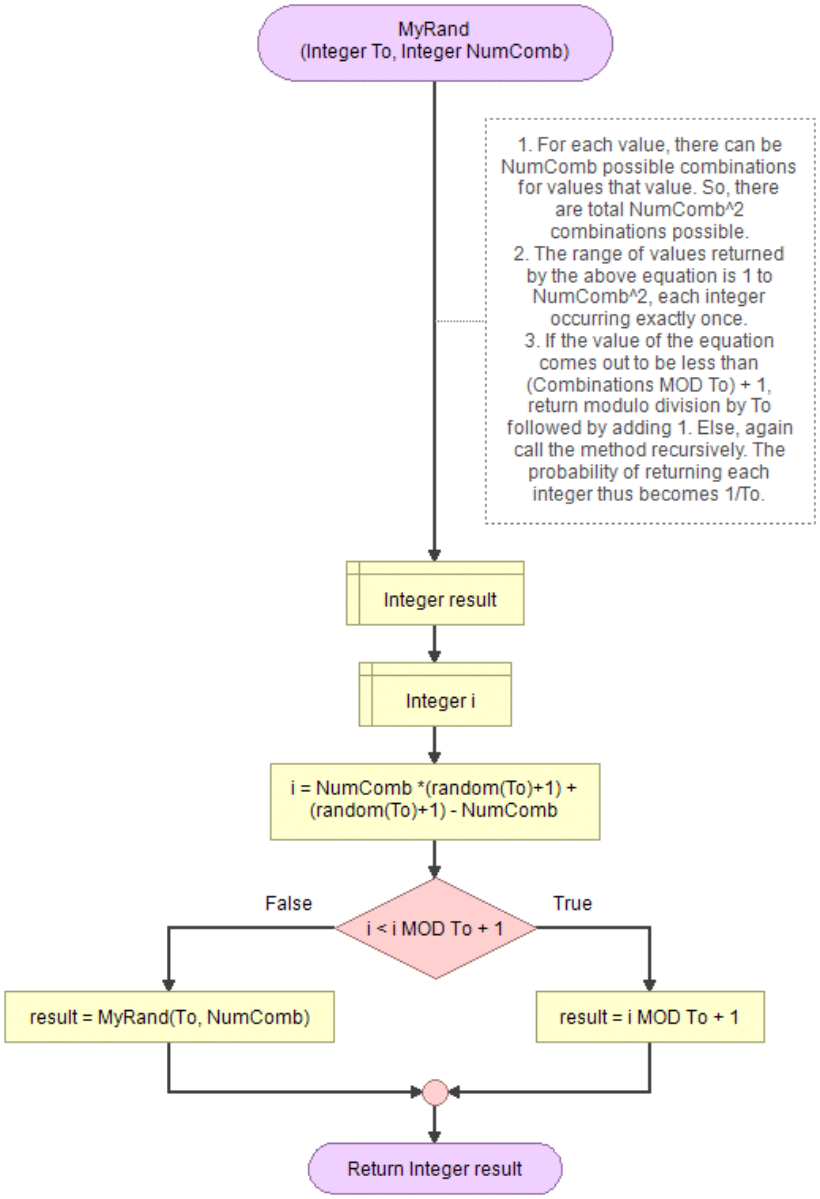




138. G F R ‖ Generare numeri casuali da 1 a N con uguale probabilità

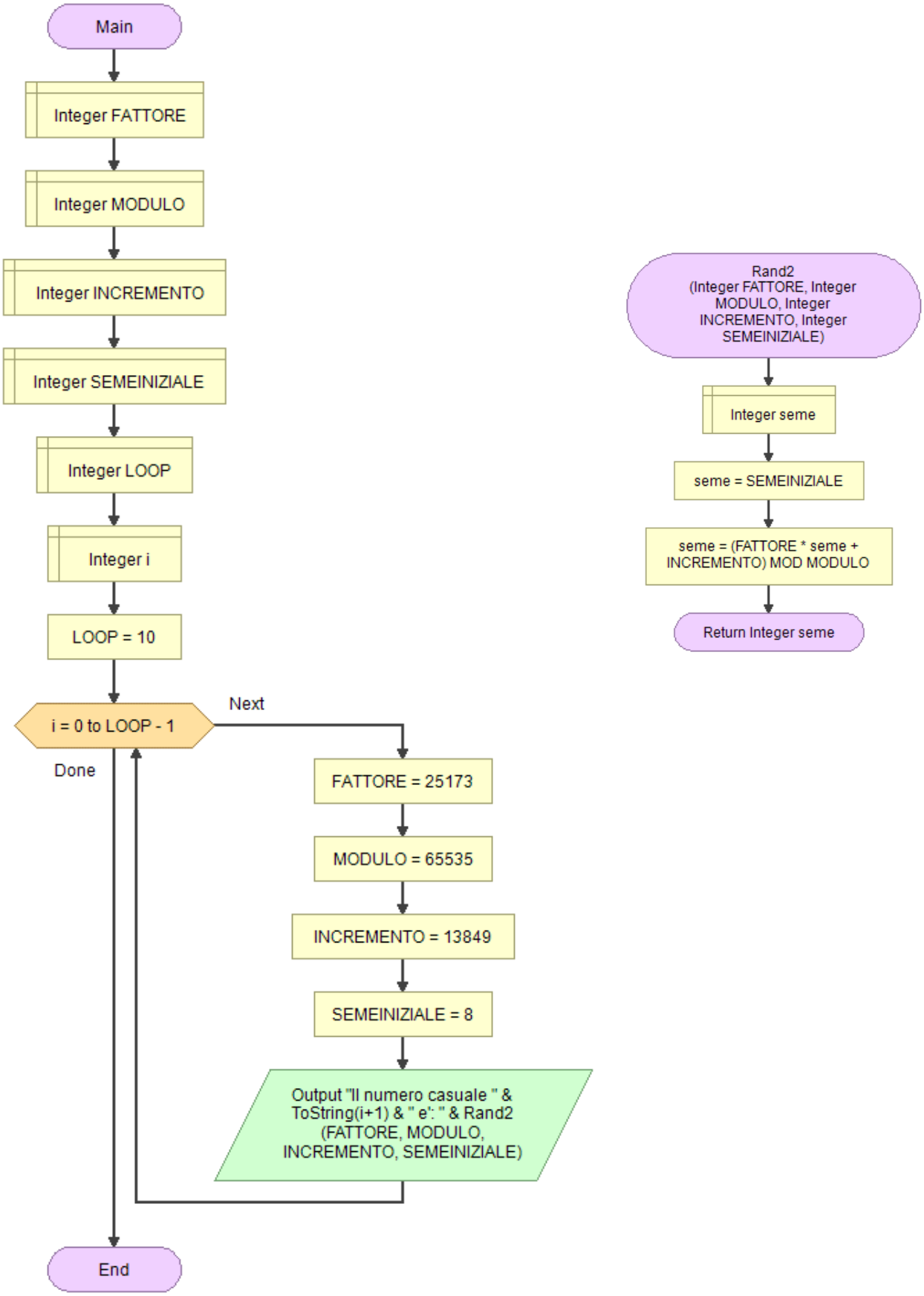
Generate random integers from 1 to N with equal probability



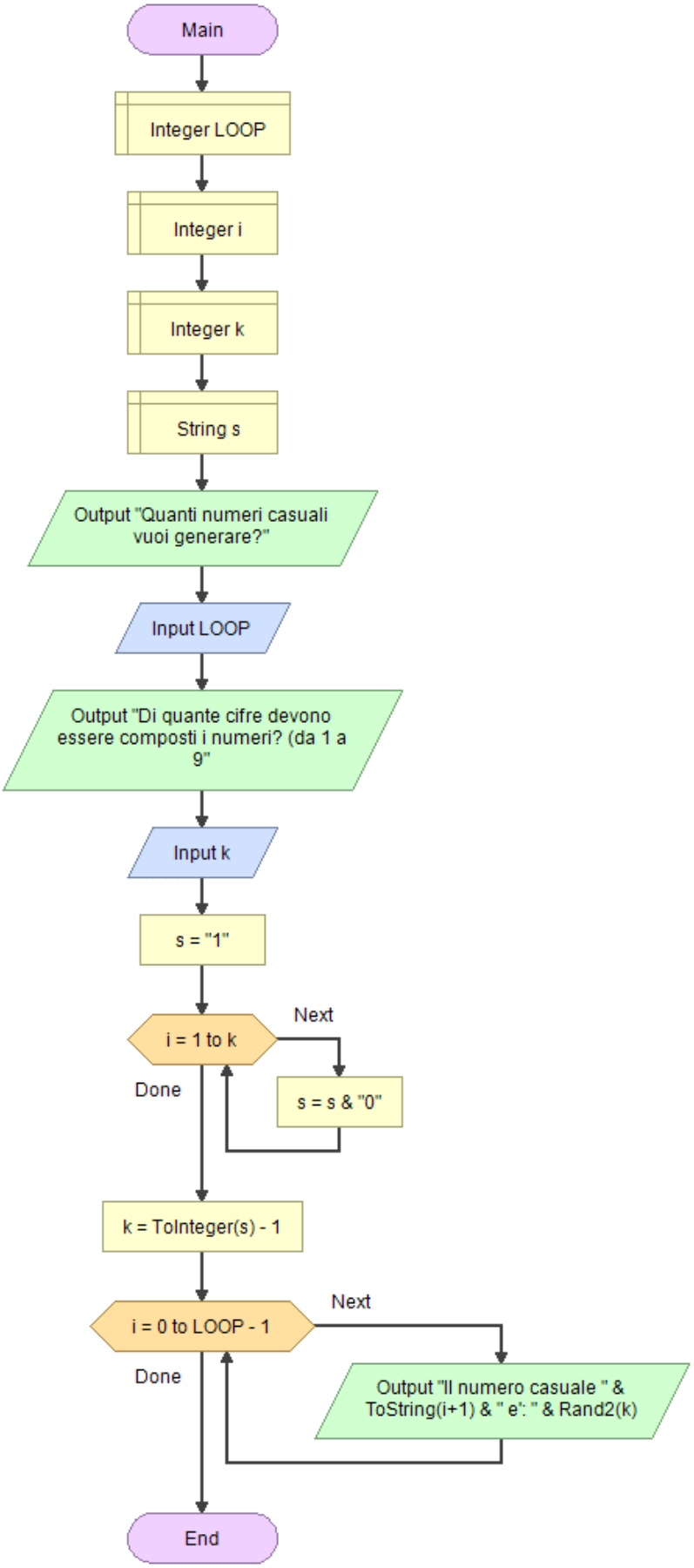


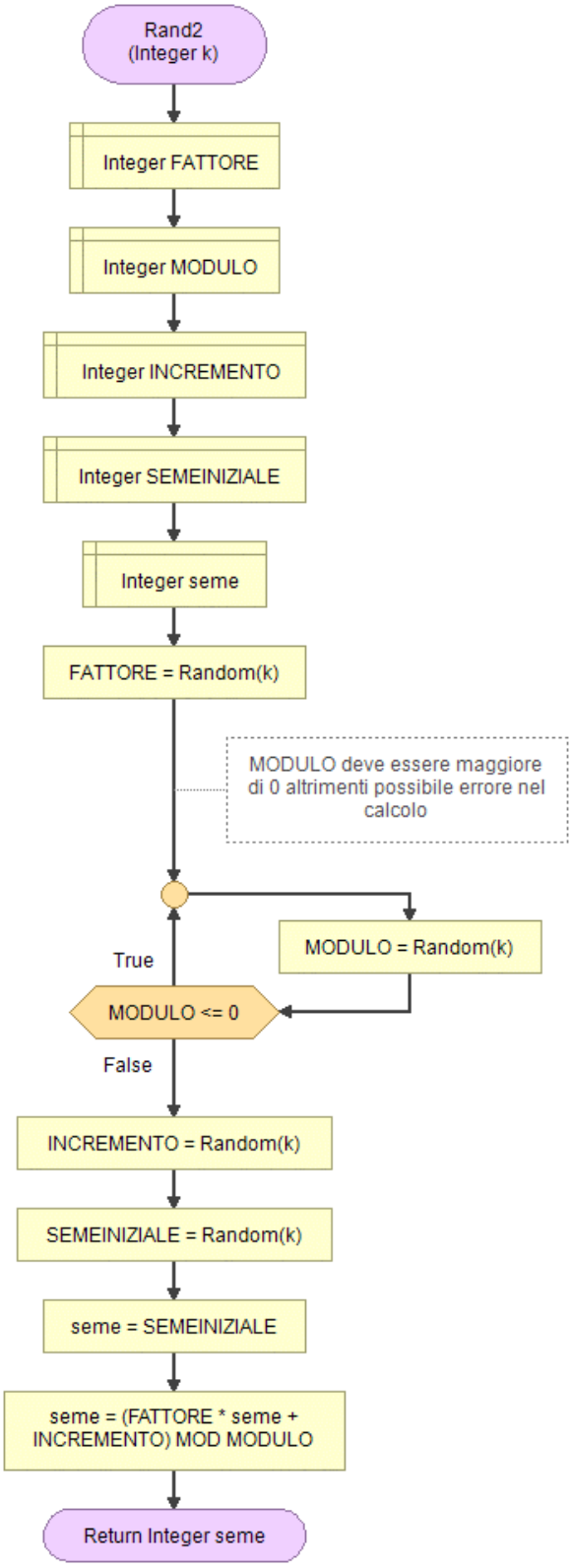
139.G F ‖ Generazione di numeri pseudo-casuali

pseudo-random sequence ver 1



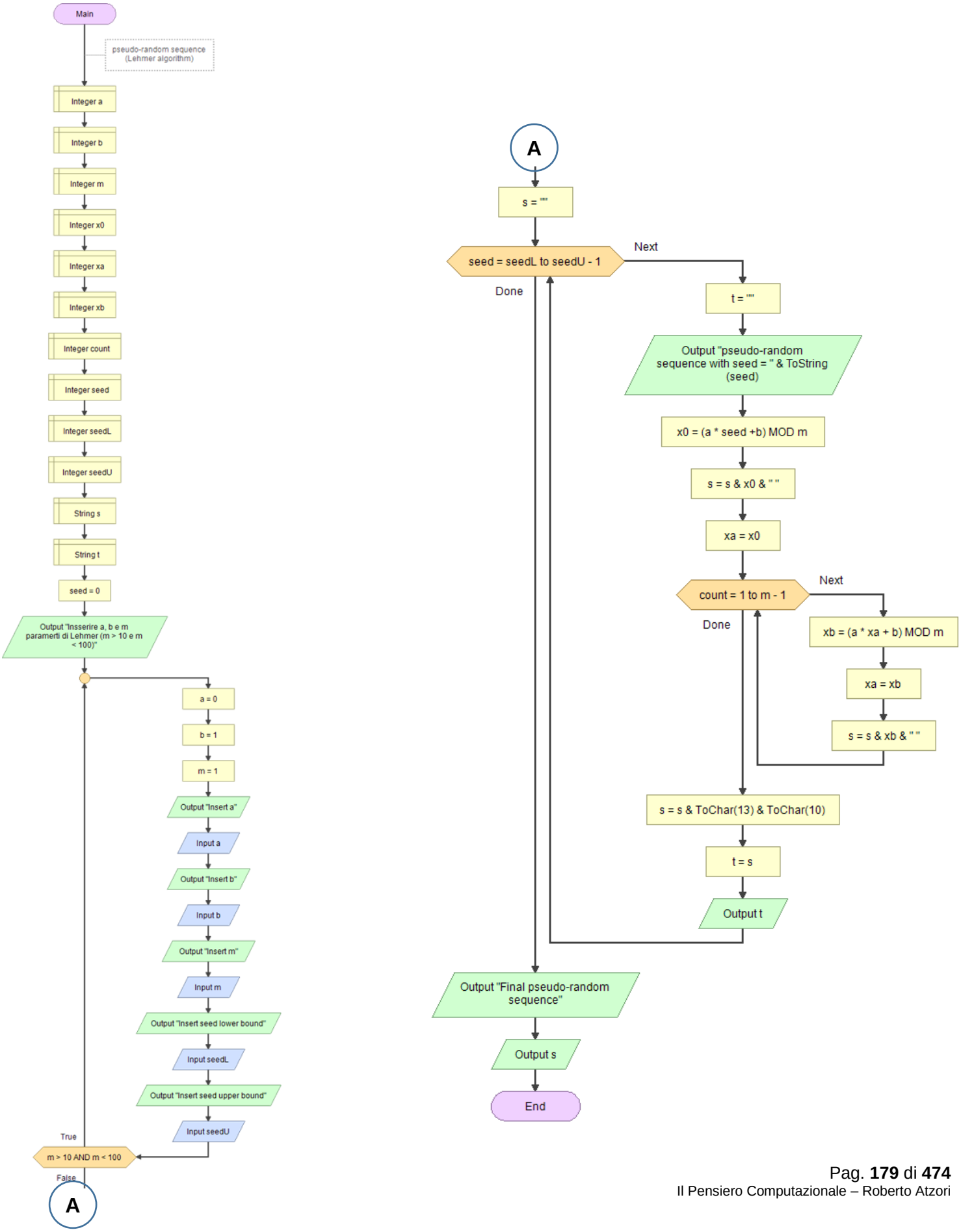
pseudo-random sequence ver 2





141.G F ‖ Generazione di numeri pseudo-casuali (metodo di Lehmer)

pseudo-random sequence (Lehmer's method)

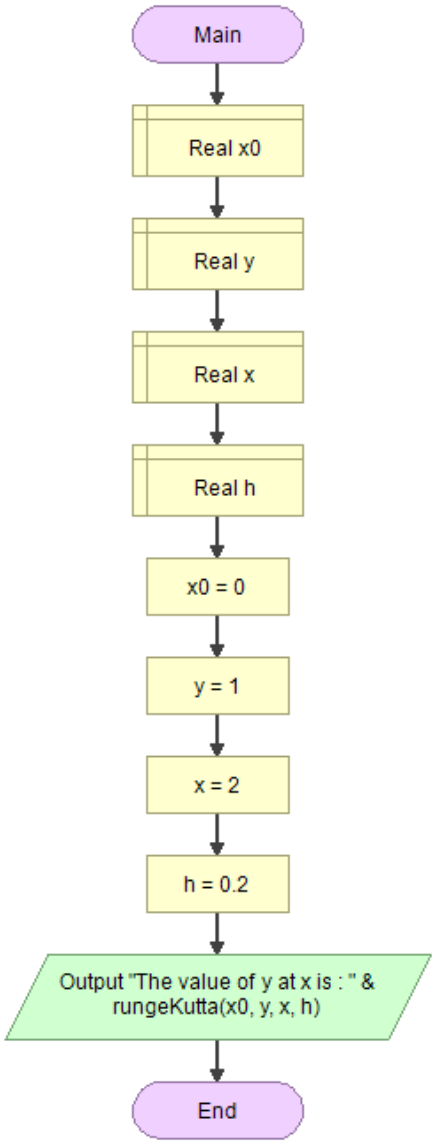


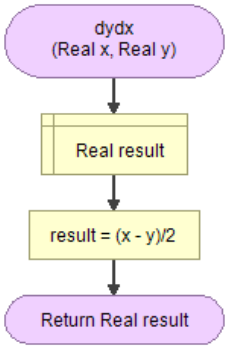
Runge-Kutta 4th Order Method to Solve Differential Equation

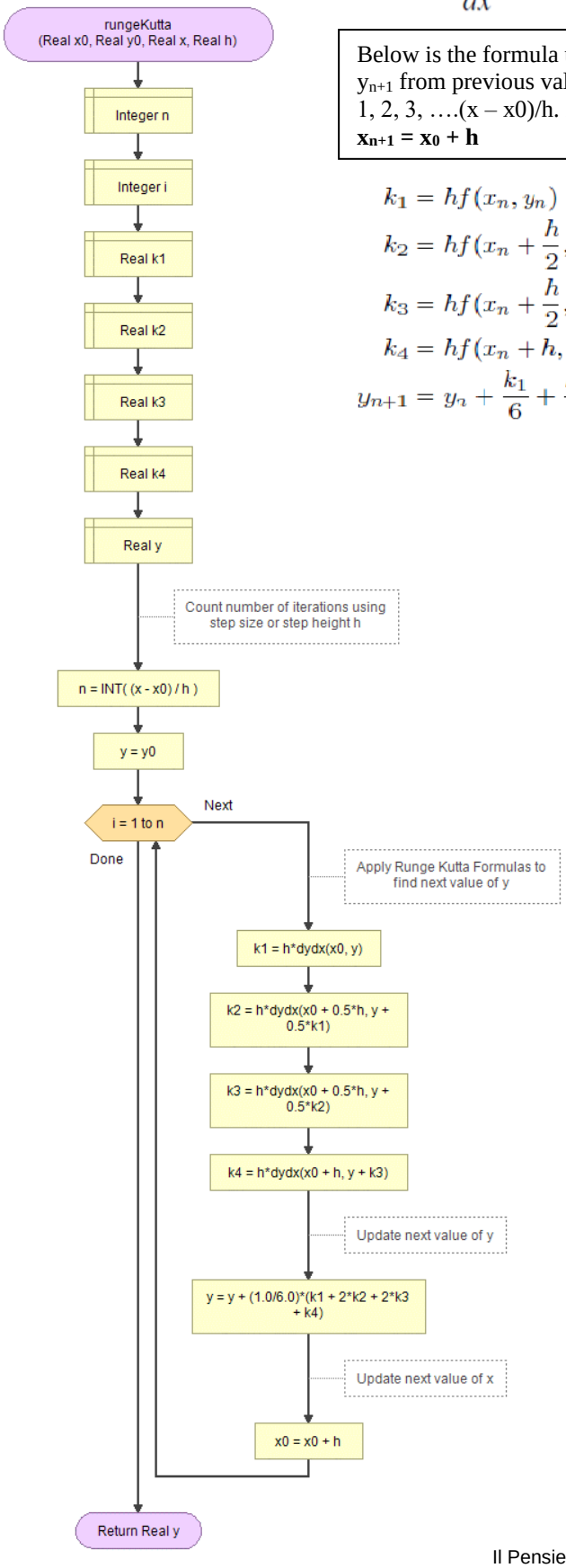
$$\frac{dy}{dx} = f(x, y), y(0) = y_0$$

Below is the formula used to compute next value y_{n+1} from previous value y_n. The value of n are 0, 1, 2, 3, ….(x – x0)/h. Here **h is step height** and $\mathbf{x_{n+1} = x_0 + h}$

$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$
$$k_3 = hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$
$$k_4 = hf(x_n + h, y_n + k_3)$$
$$y_{n+1} = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5)$$



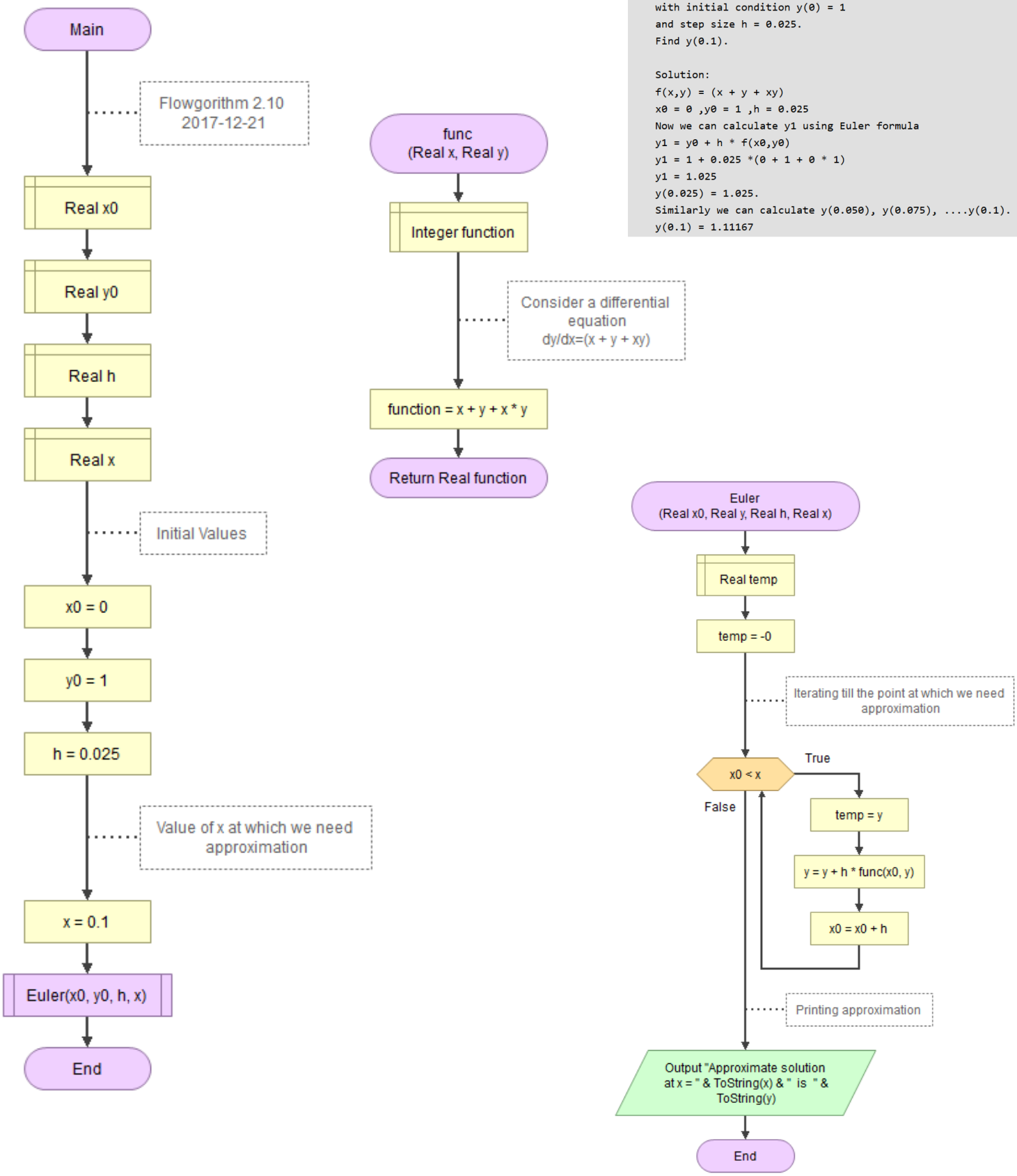




143.G F ‖ Metodo di Eulero per il calcolo di una equazione differenziale

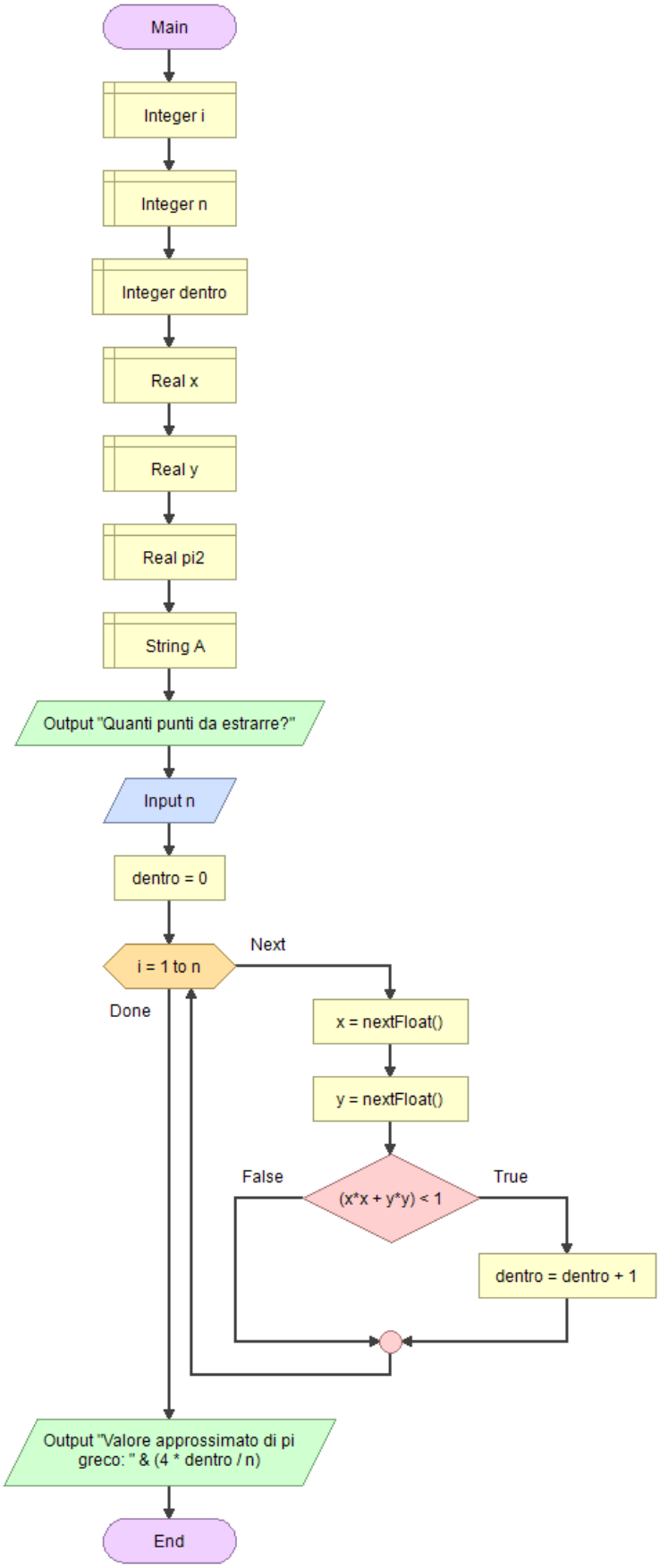
Euler Method for solving differential equation

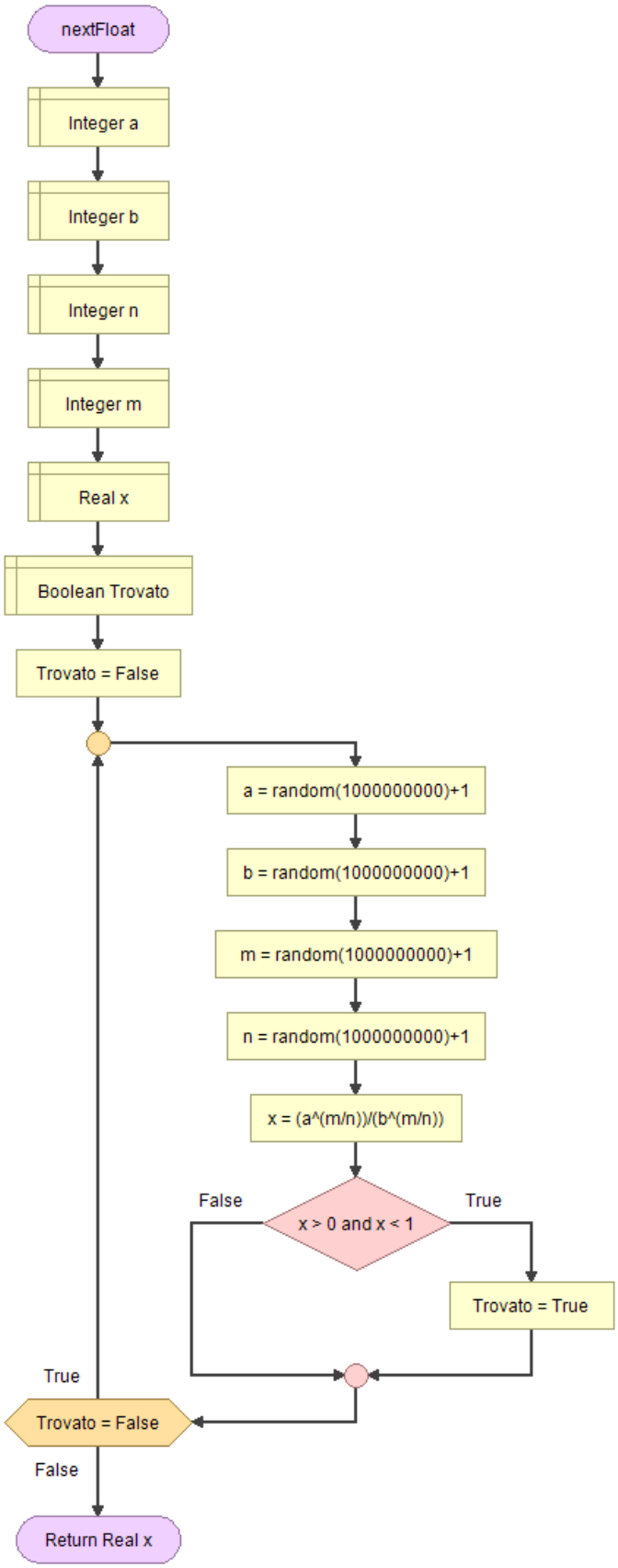
```
Example:
    Consider below differential equation
            dy/dx = (x + y + xy)
    with initial condition y(0) = 1
    and step size h = 0.025.
    Find y(0.1).

    Solution:
    f(x,y) = (x + y + xy)
    x0 = 0 ,y0 = 1 ,h = 0.025
    Now we can calculate y1 using Euler formula
    y1 = y0 + h * f(x0,y0)
    y1 = 1 + 0.025 *(0 + 1 + 0 * 1)
    y1 = 1.025
    y(0.025) = 1.025.
    Similarly we can calculate y(0.050), y(0.075), ....y(0.1).
    y(0.1) = 1.11167
```



144.**G F** ‖ Calcolare il valore di pi greco usando il metodo di Montecarlo (circle problem approximation)

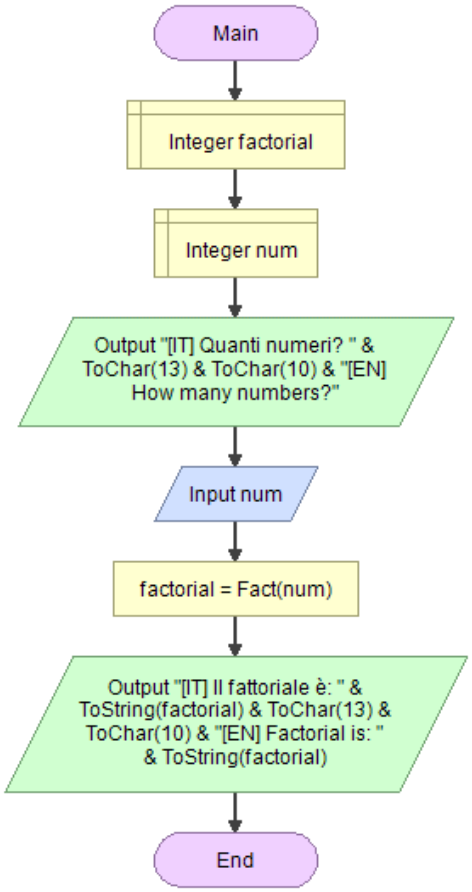
PI calculation by circle problem approximation

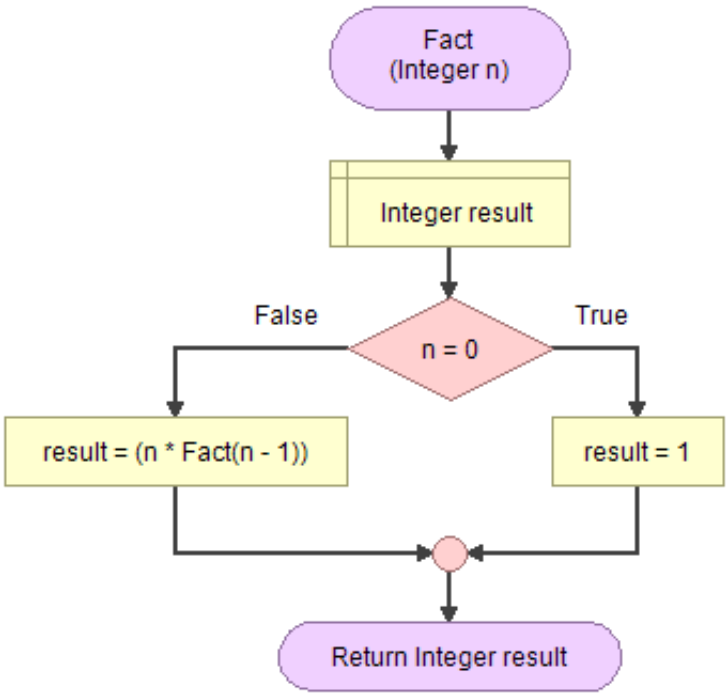




145. G F R | Fattoriale di un numero con ricorsione

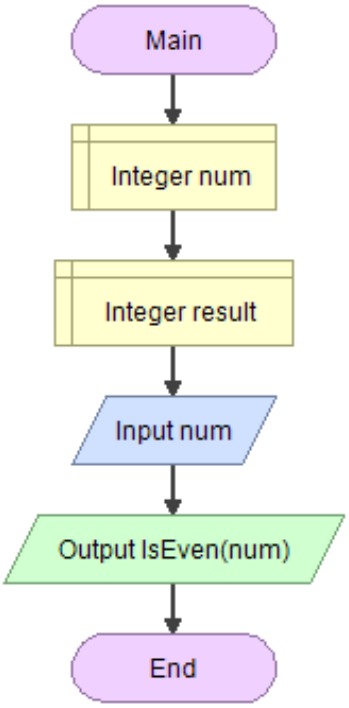
Factorial of Number Using Recursion

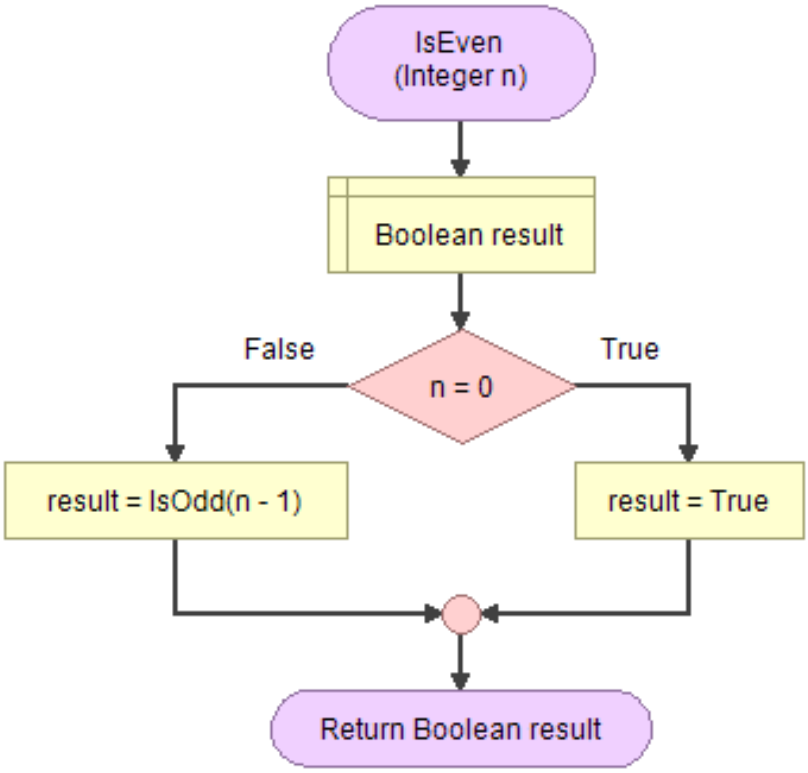


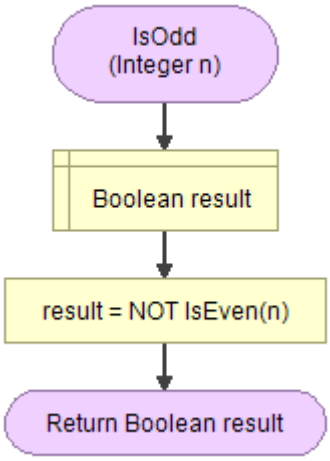


146. **G F R** ‖ Funzione pari o dispari mutuamente ricorsiva

Mutually recursive function to determine if a number is even or odd



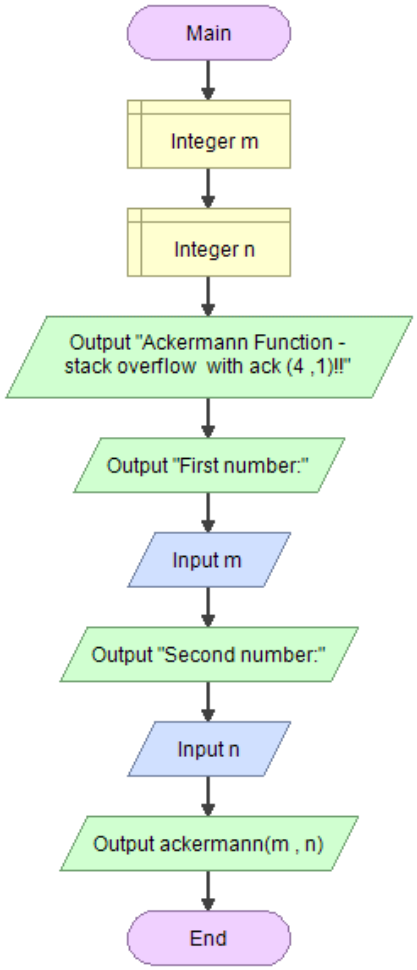


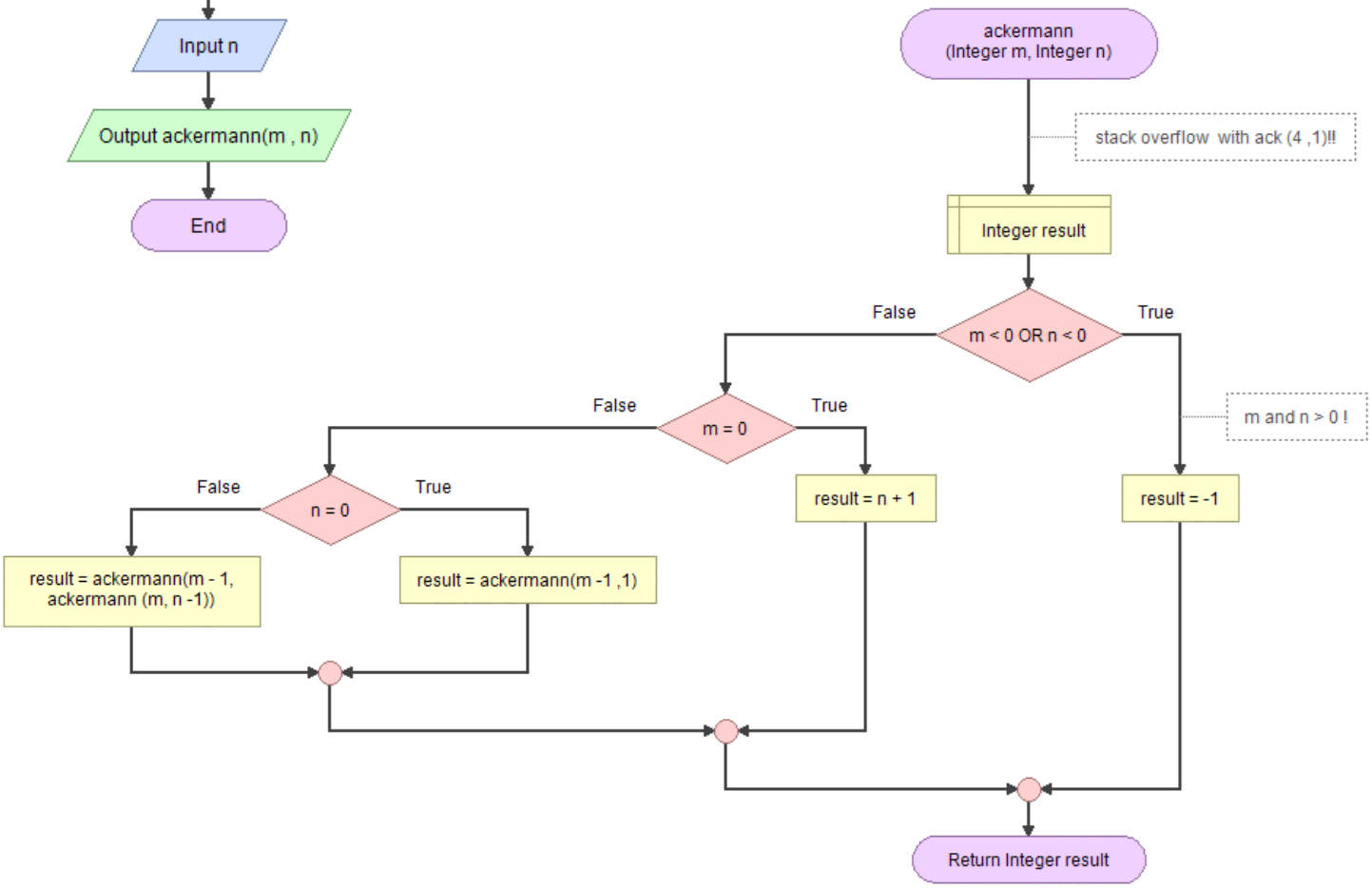


147.G F R ‖ Funzione ricorsiva annidata per calcolare la funzione di Ackerman

Nested recursive function to calculate the Ackermann function

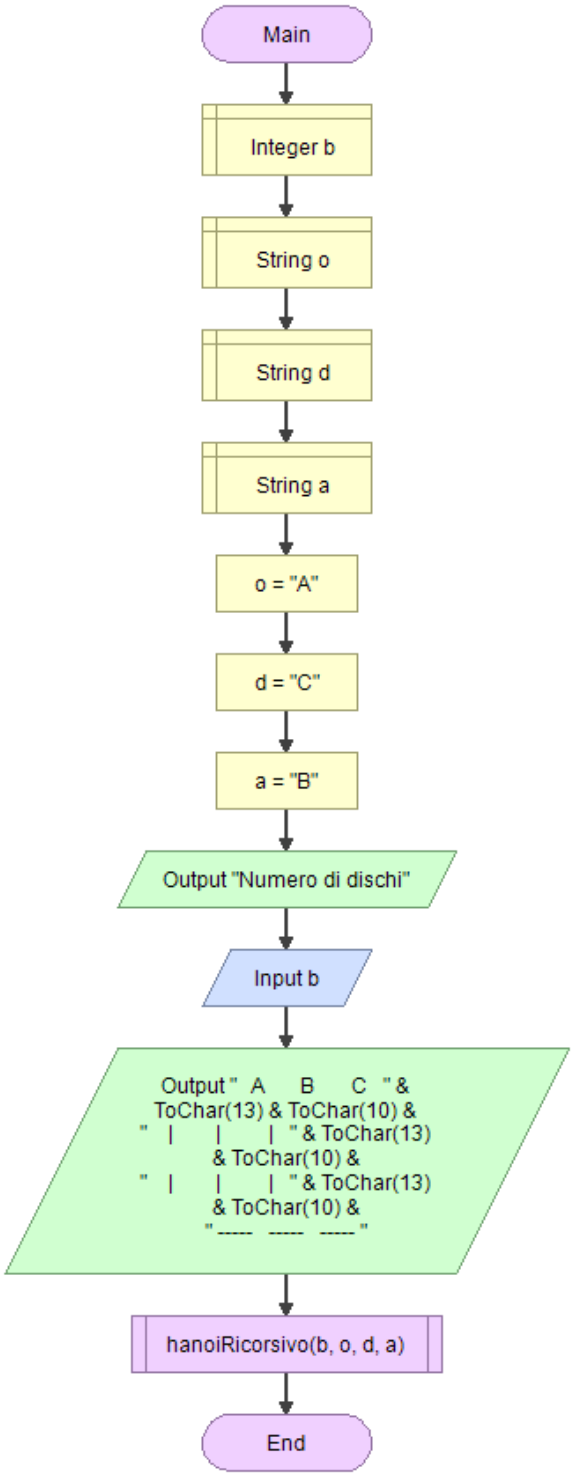
$$A(m,n) = \begin{cases} n+1 & \text{se } m = 0 \\ A(m-1,1) & \text{se } m > 0 \text{ e } n = 0 \\ A(m-1, A(m,n-1)) & \text{se } m > 0 \text{ e } n > 0. \end{cases}$$

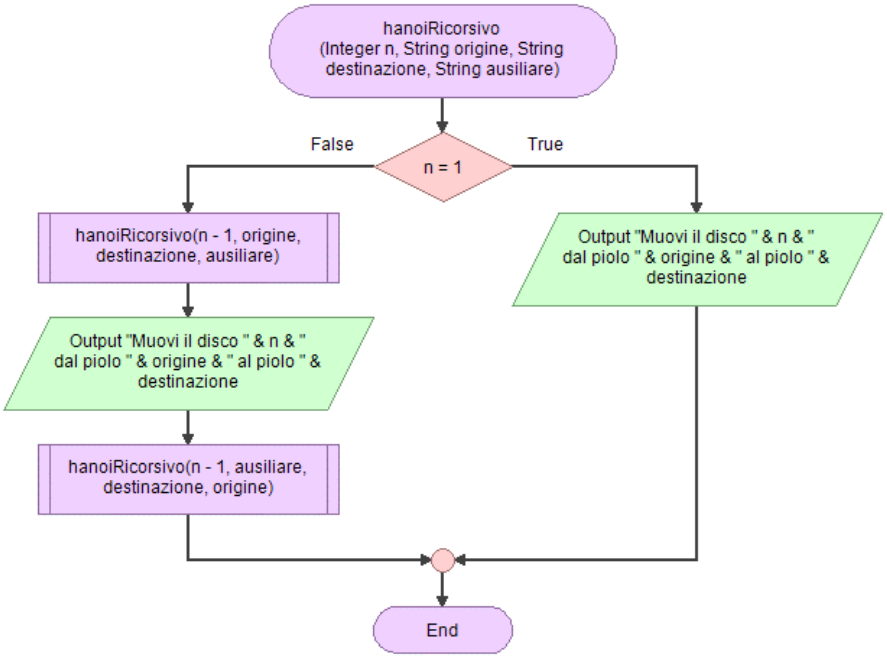




148.GFR | Torre di Hanoi

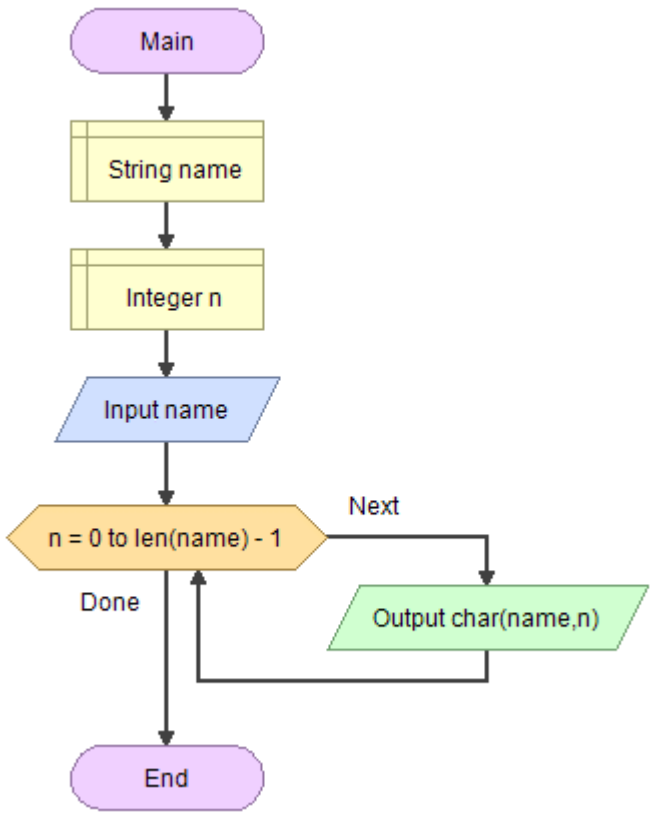
Hanoi Tower





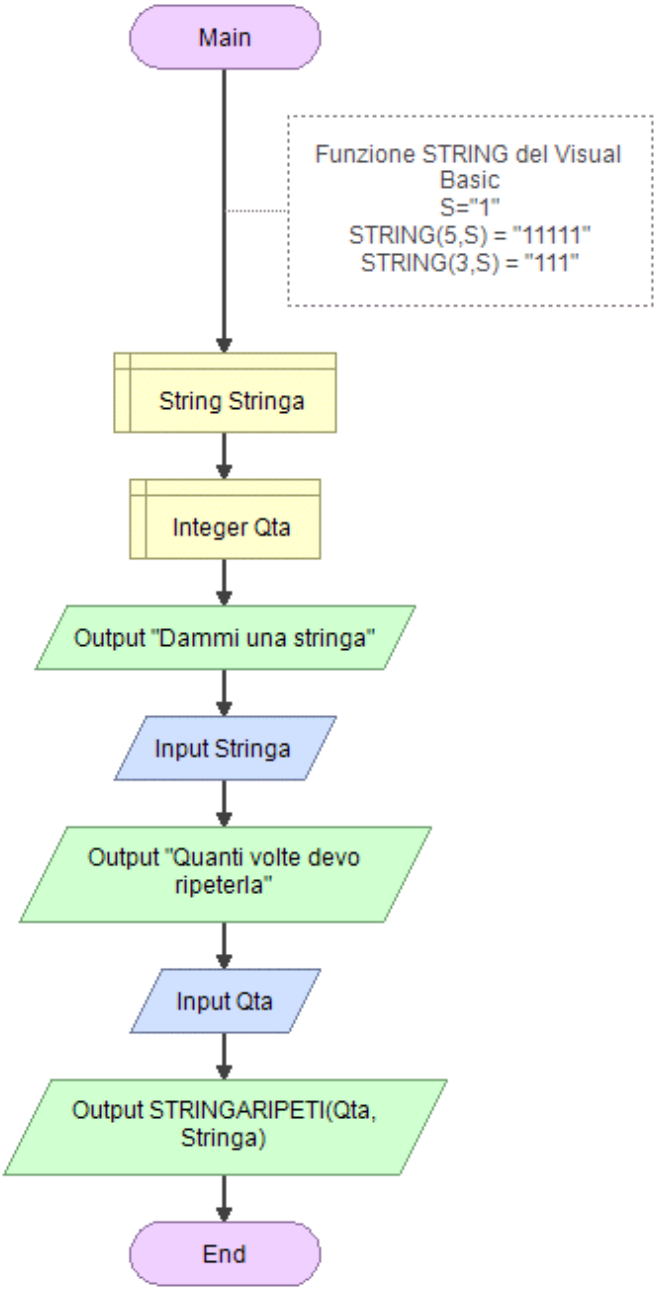
149. **G S** ‖ Visualizzazione carattere per carattere di una stringa in input

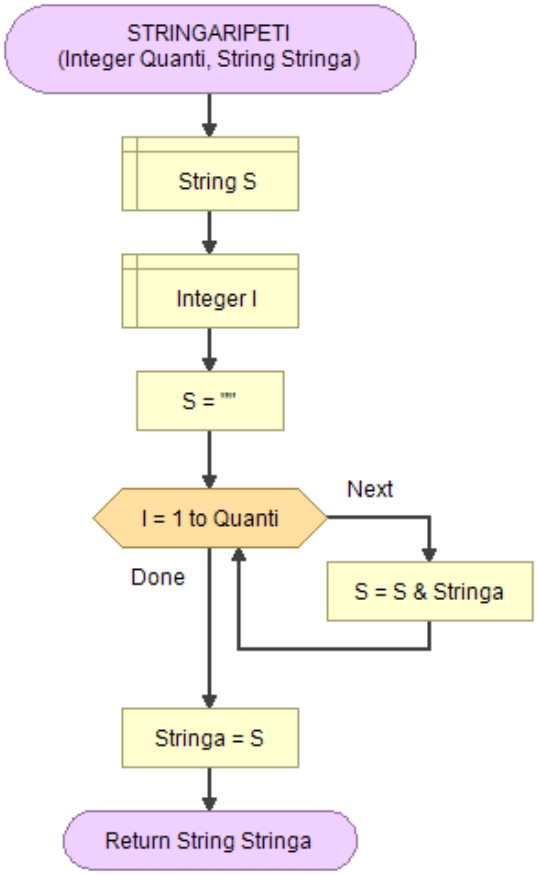
String - Character Per Line



150.GSF ‖ Funzione STRING del Visual Basic - utilizzo con funzione

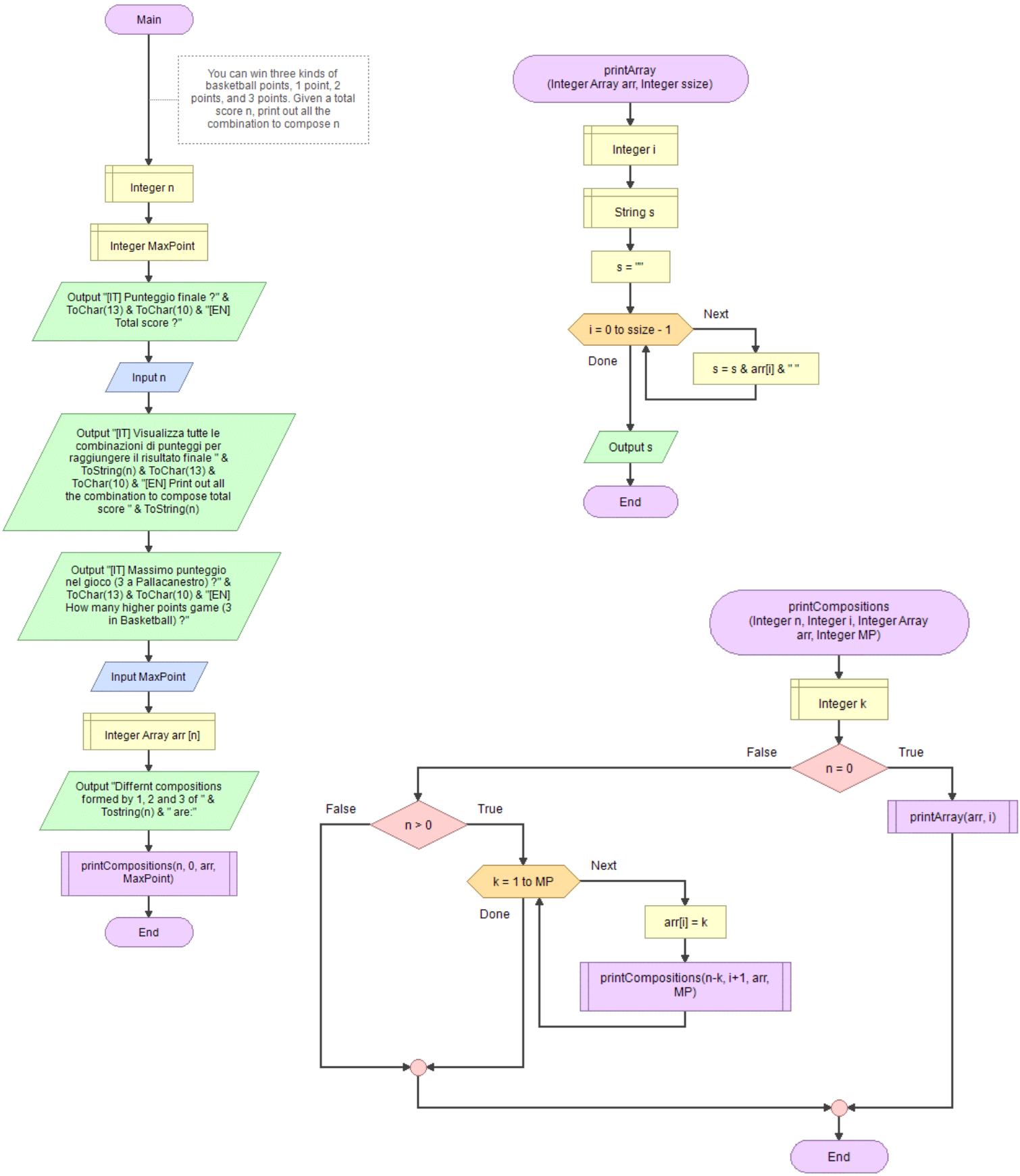
Function STRING in Visual Basic 6





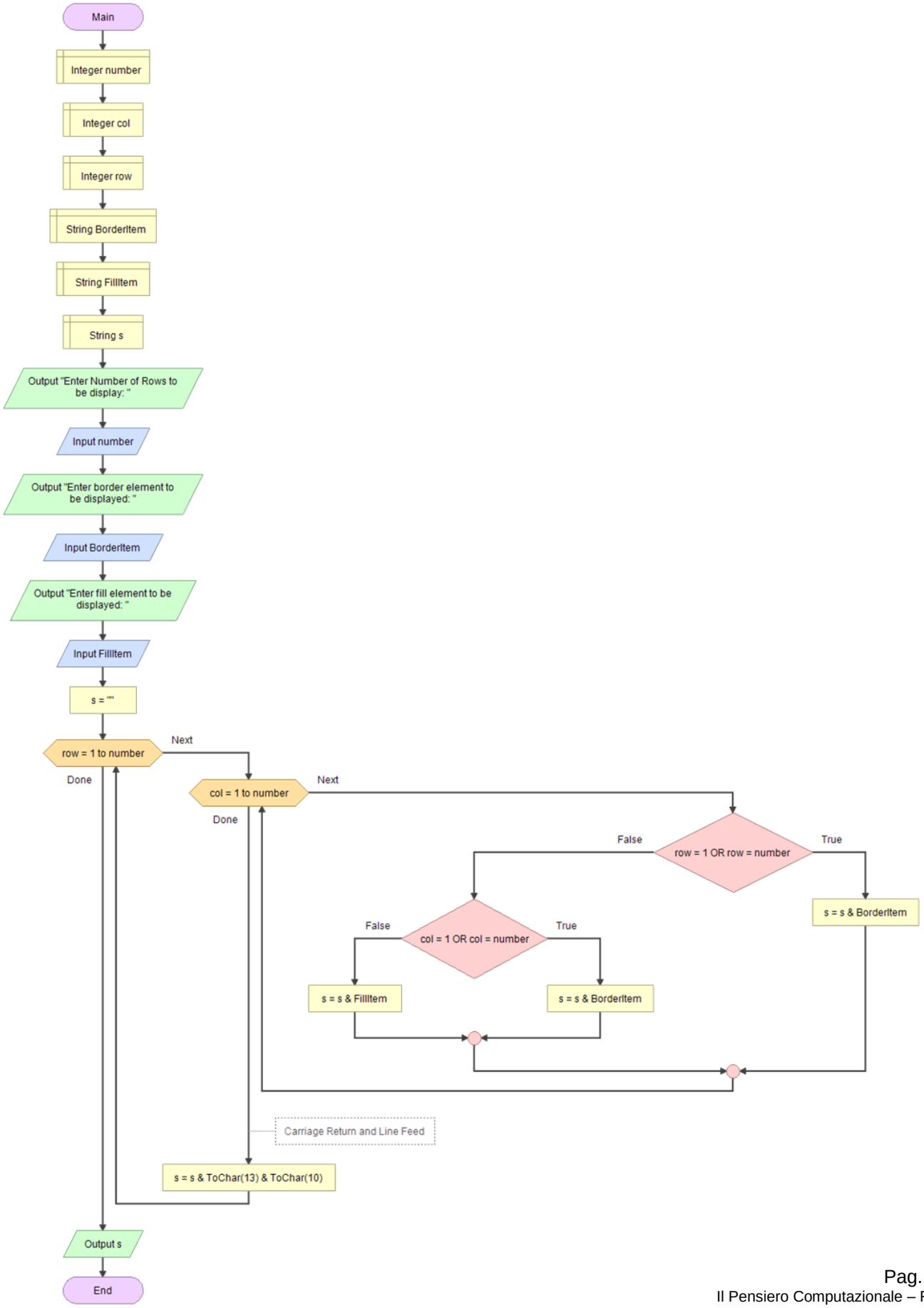
151.GSFR ‖ Visualizza tutte le combinazioni di punteggi per raggiungere il risultato finale

Print all combinations of points that can compose a given number



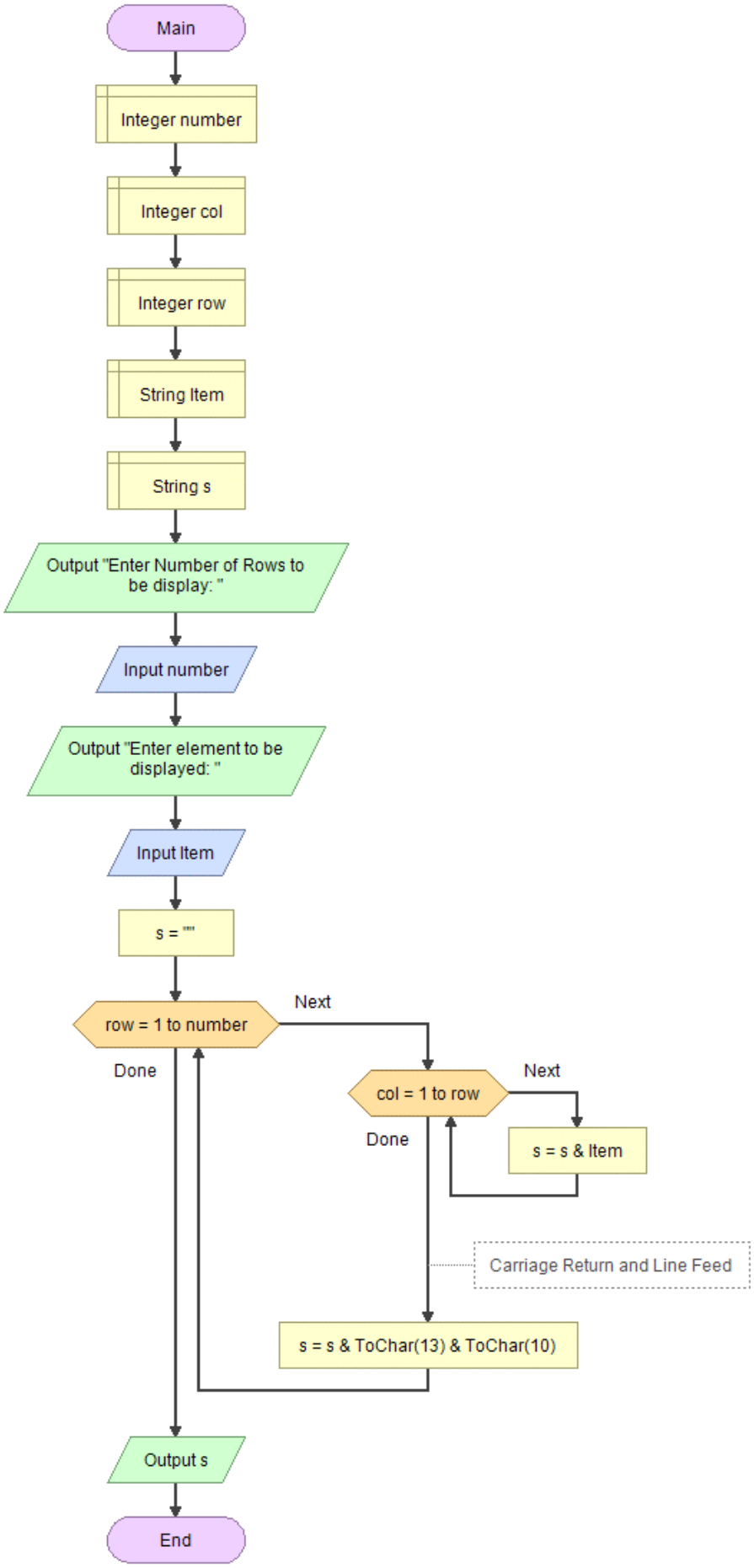
152.GS | Generare figura contornata e riempita da simboli

Generate item border rectangular pyramid



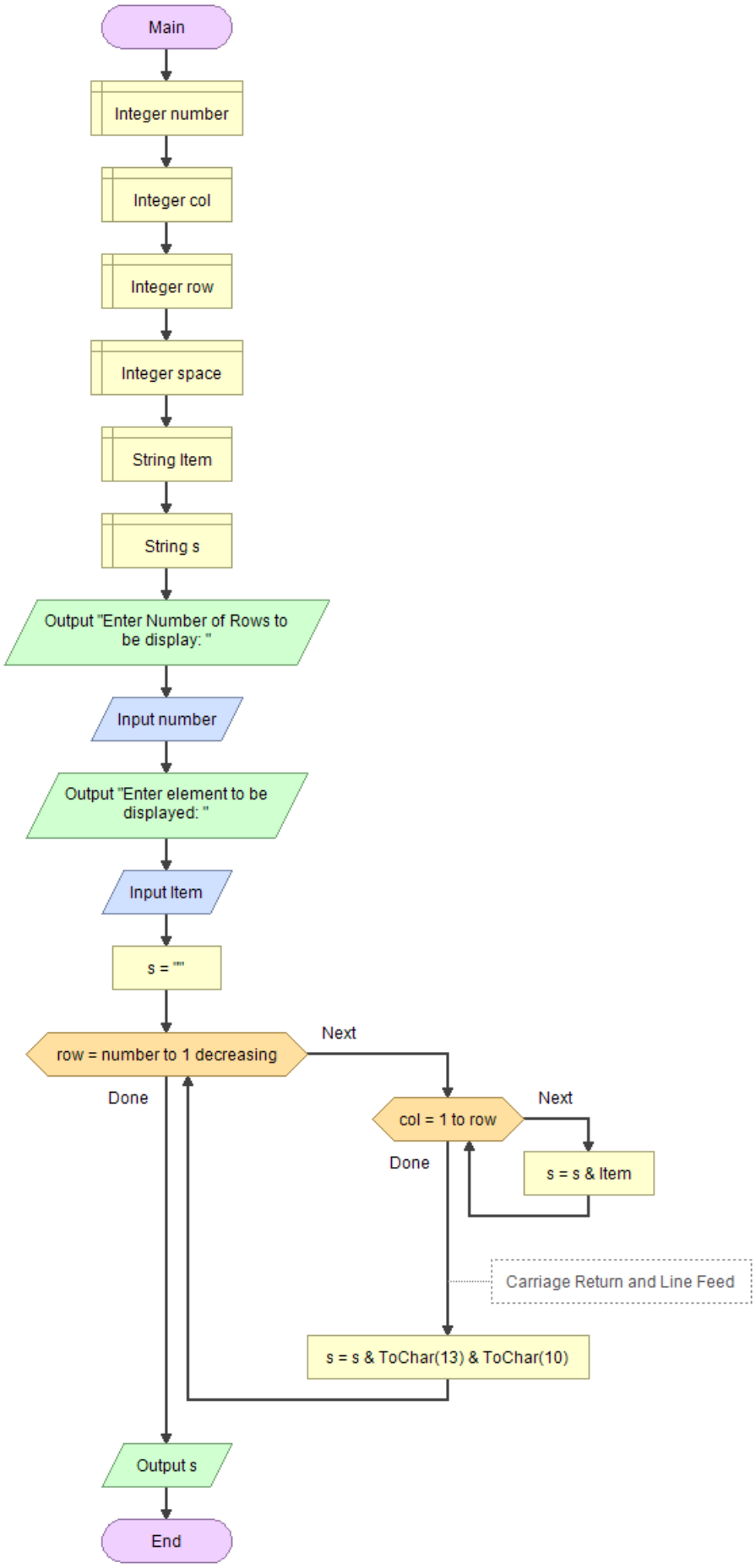
153. **G S** ‖ Generare piramide invertita angolata a destra

Generate inverted right angle pyramid



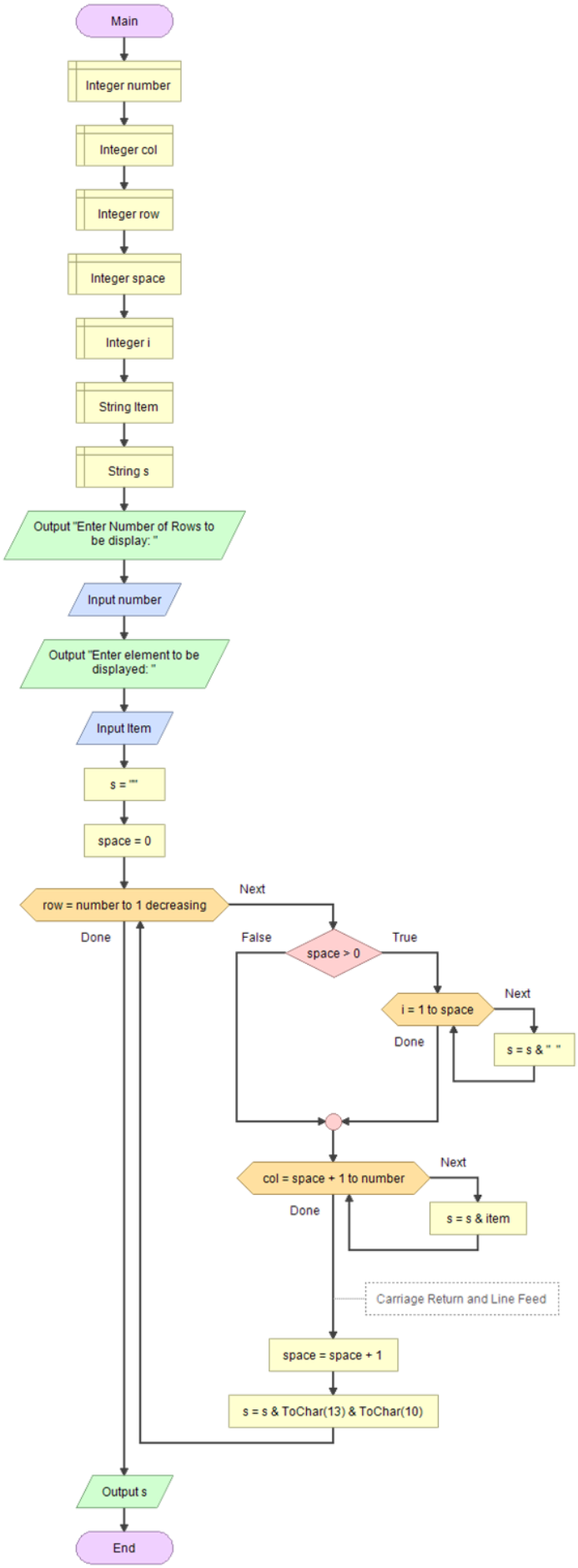
154. G S ‖ Generare piramide angolata a sinistra

Generate left angle pyramid



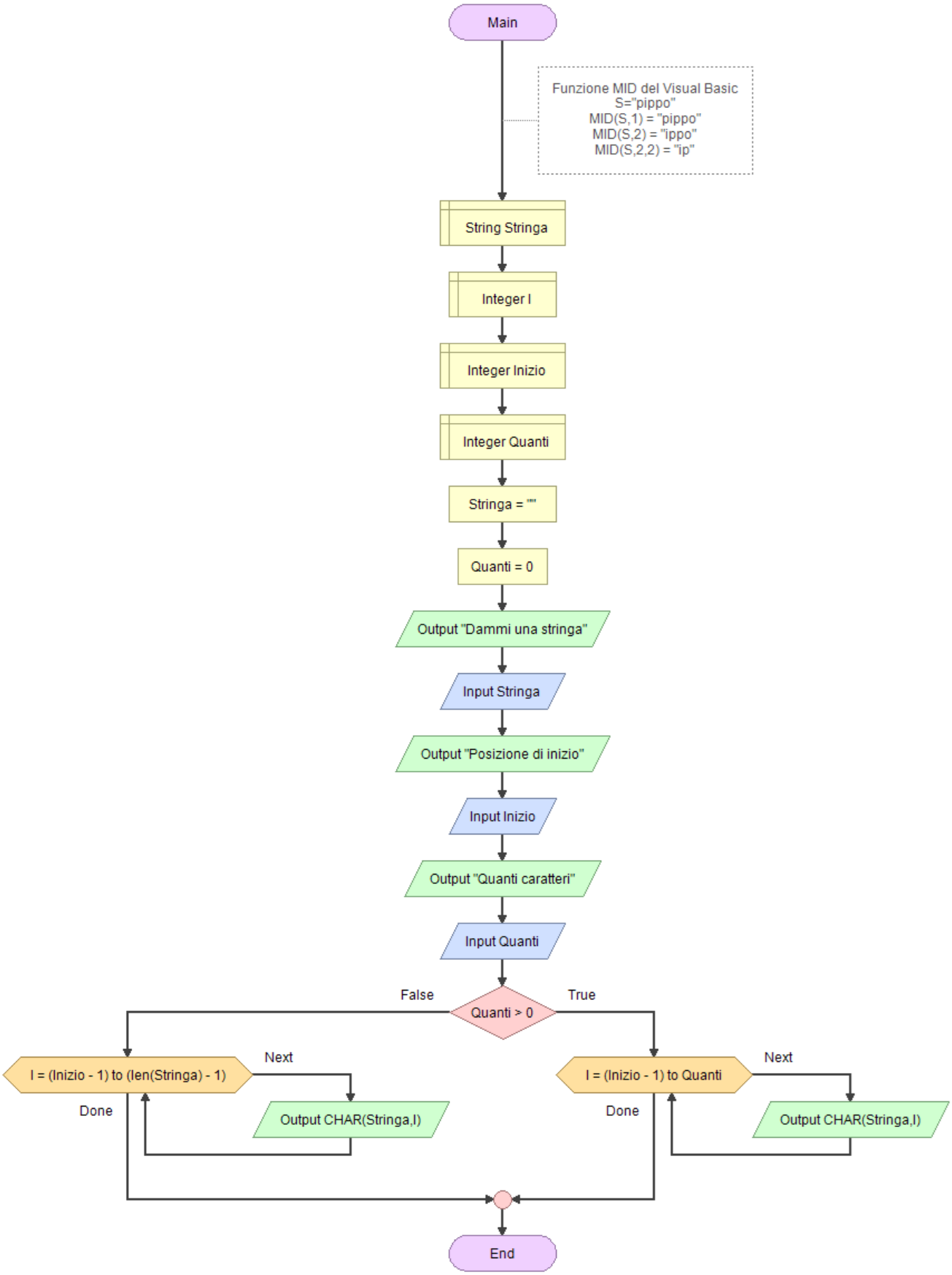
155. G S ‖ Generare piramide angolata a destra

Generate right angle pyramid



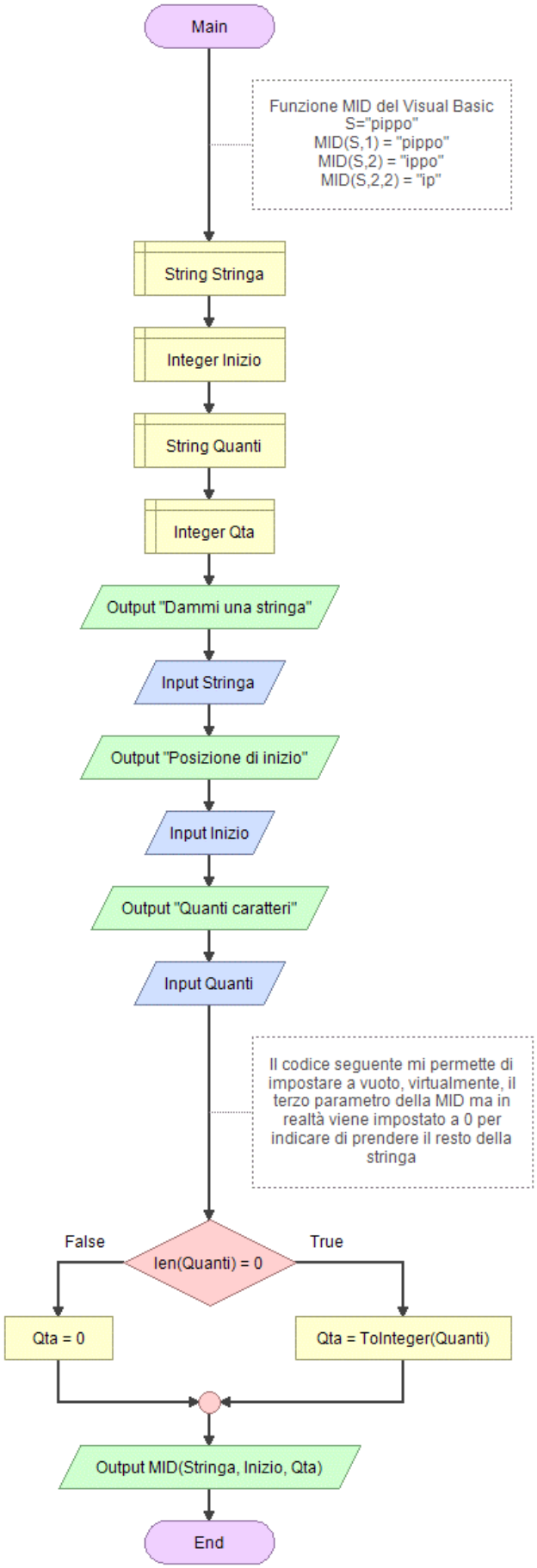
156.GS ‖ Funzione MID del Visual Basic

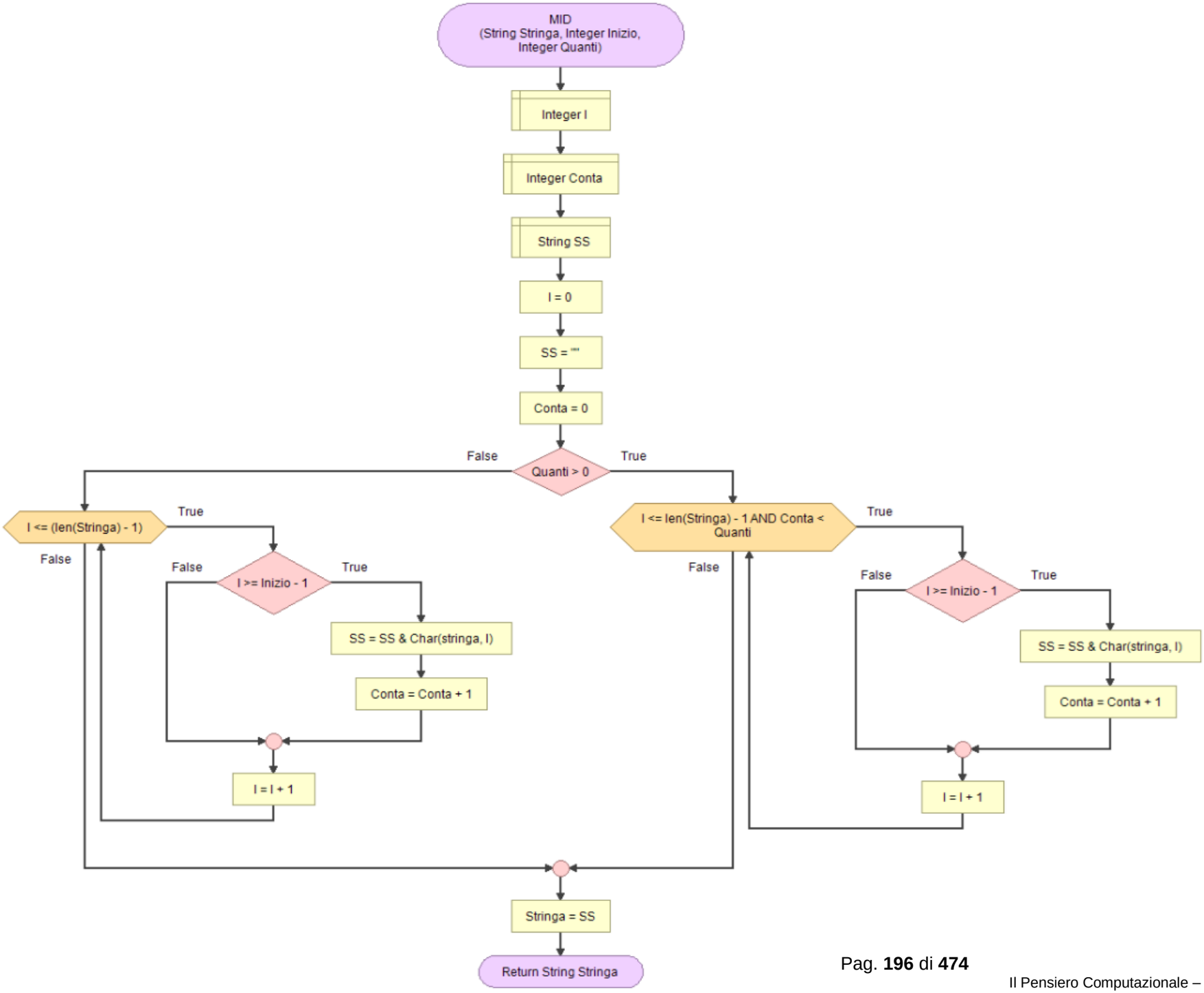
MID code Visual Basic 6



157.GSF ‖ Funzione MID del Visual Basic - utilizzo con funzione

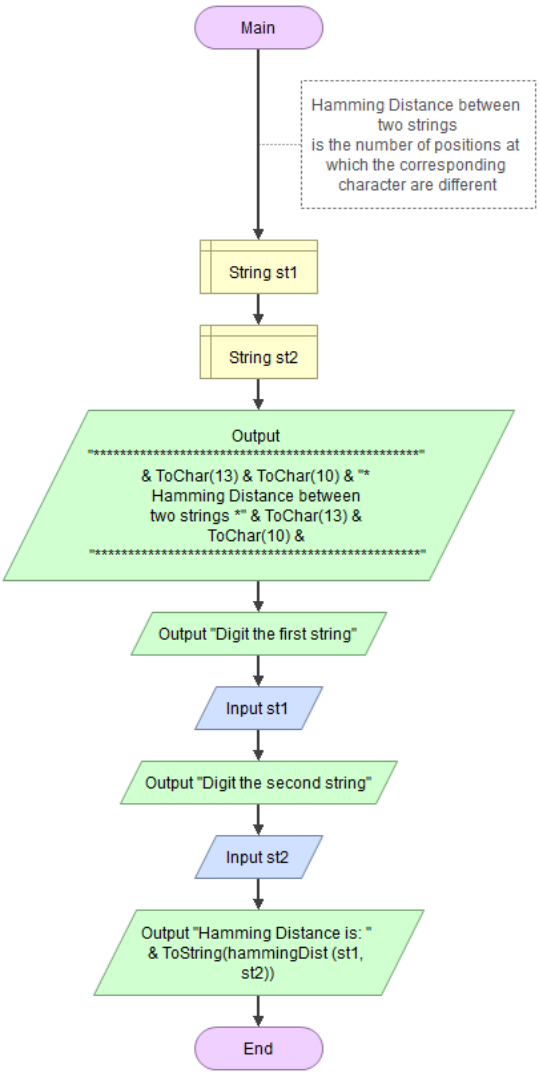
MID function Visual Basic 6

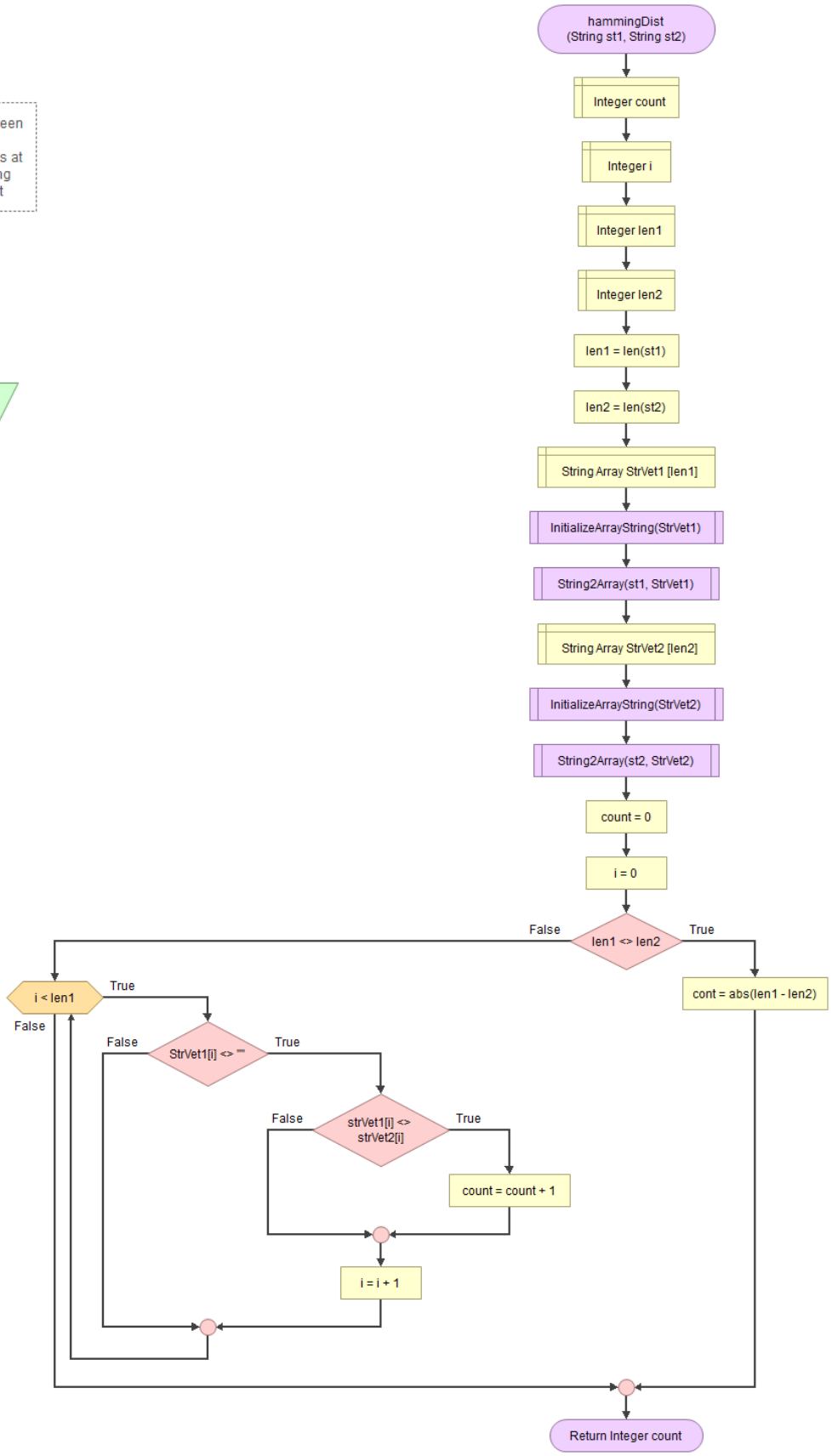


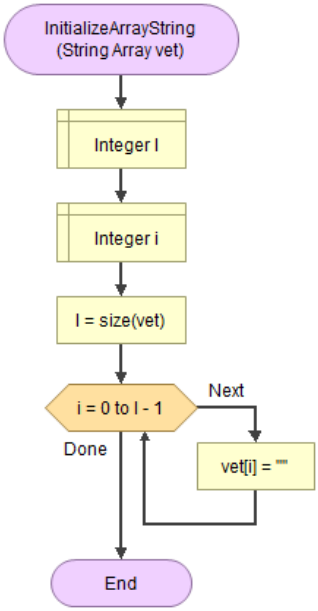

MID
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

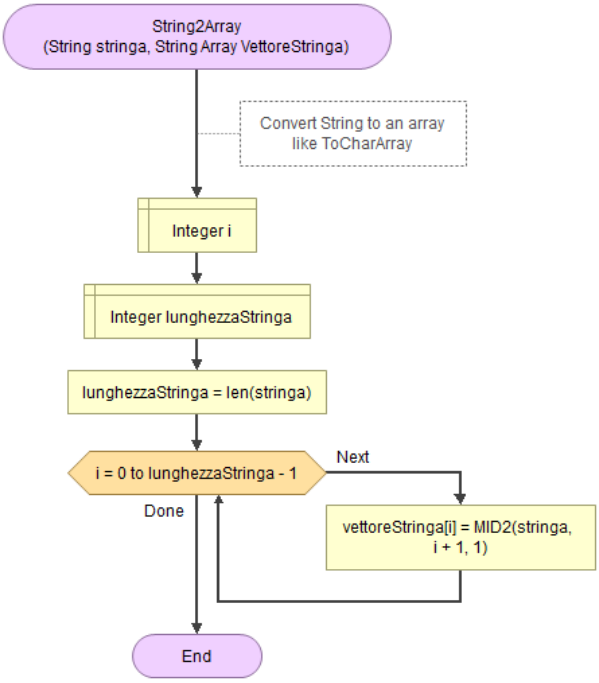
158.GSF | Distanza di Hamming tra due stringhe

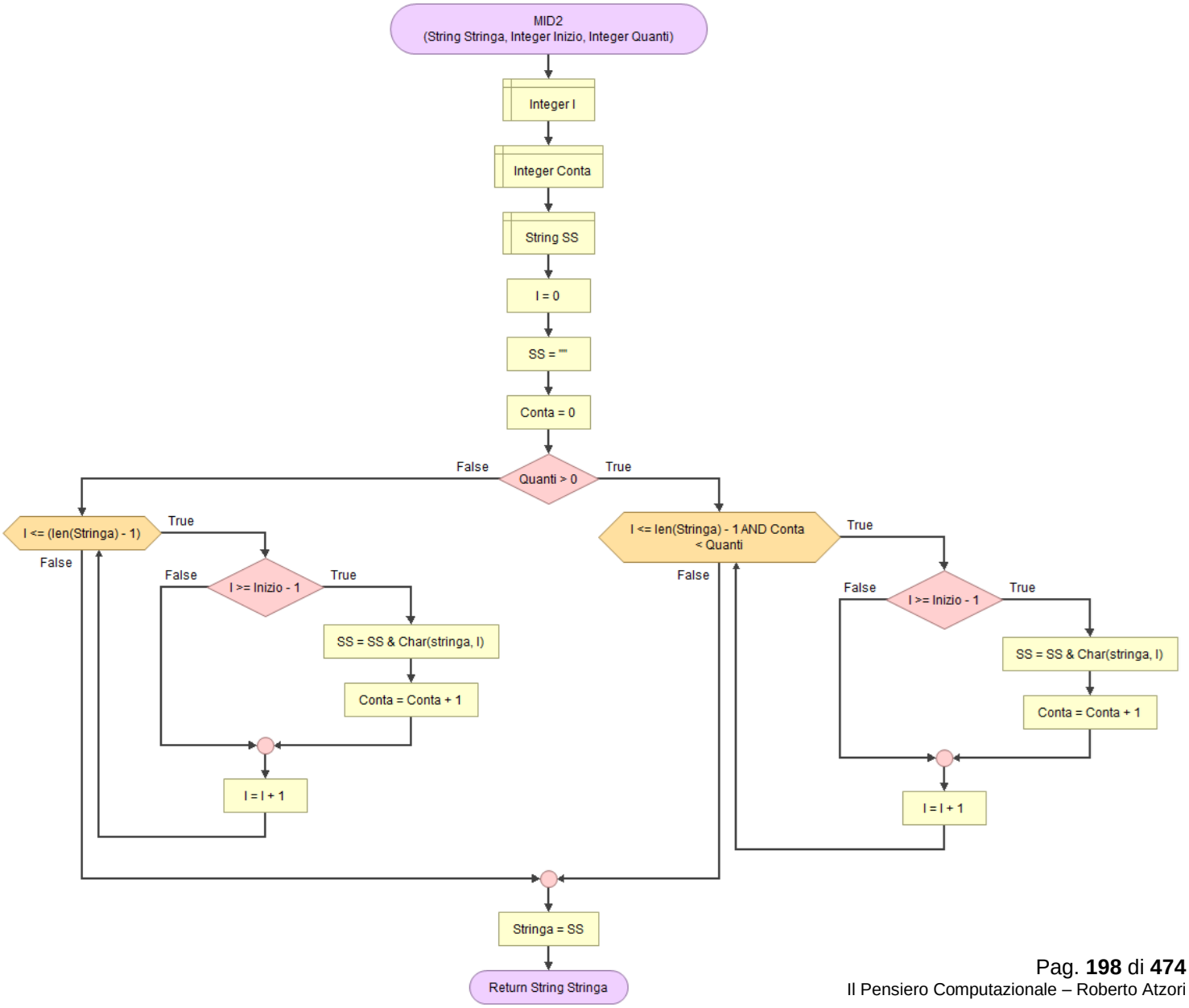
Hamming Distance between two strings





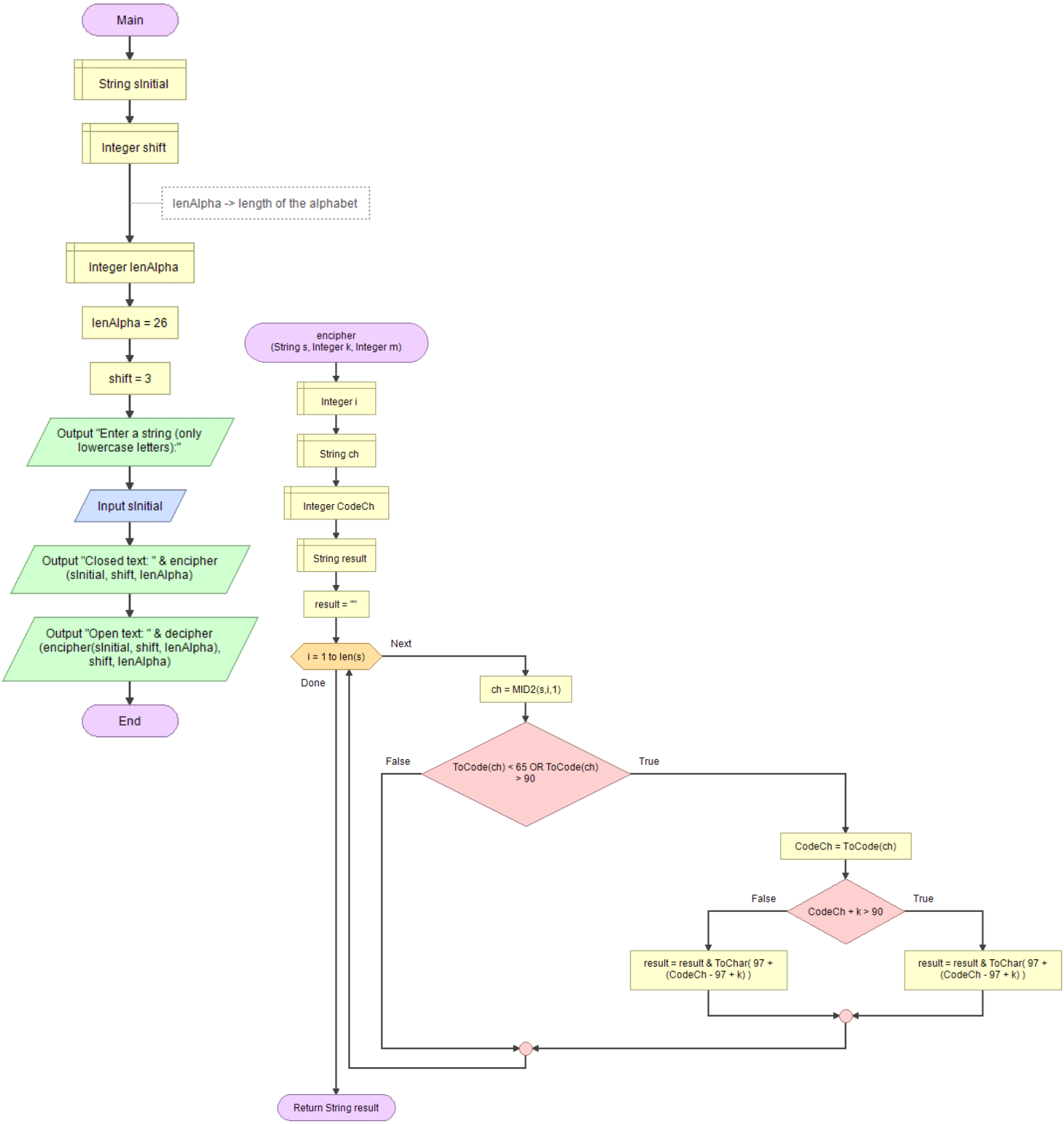

InitializeArrayString
(String Array vet)
Integer l
Integer i
l = size(vet)
i = 0 to l - 1
Next
Done
vet[i] = ""
End

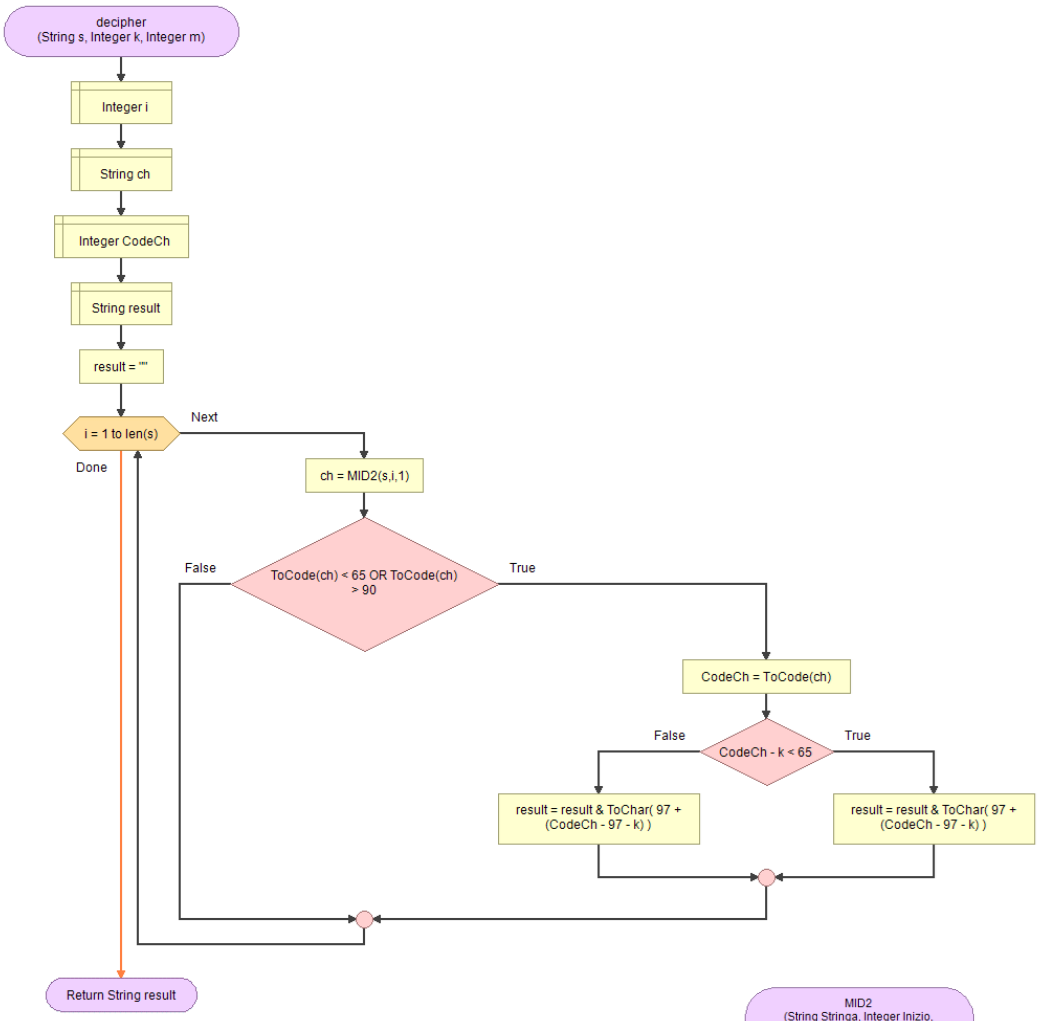

String2Array
(String stringa, String Array VettoreStringa)
Convert String to an array
like ToCharArray
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len(stringa)
i = 0 to lunghezzaStringa - 1
Next
Done
vettoreStringa[i] = MID2(stringa,
i + 1, 1)
End

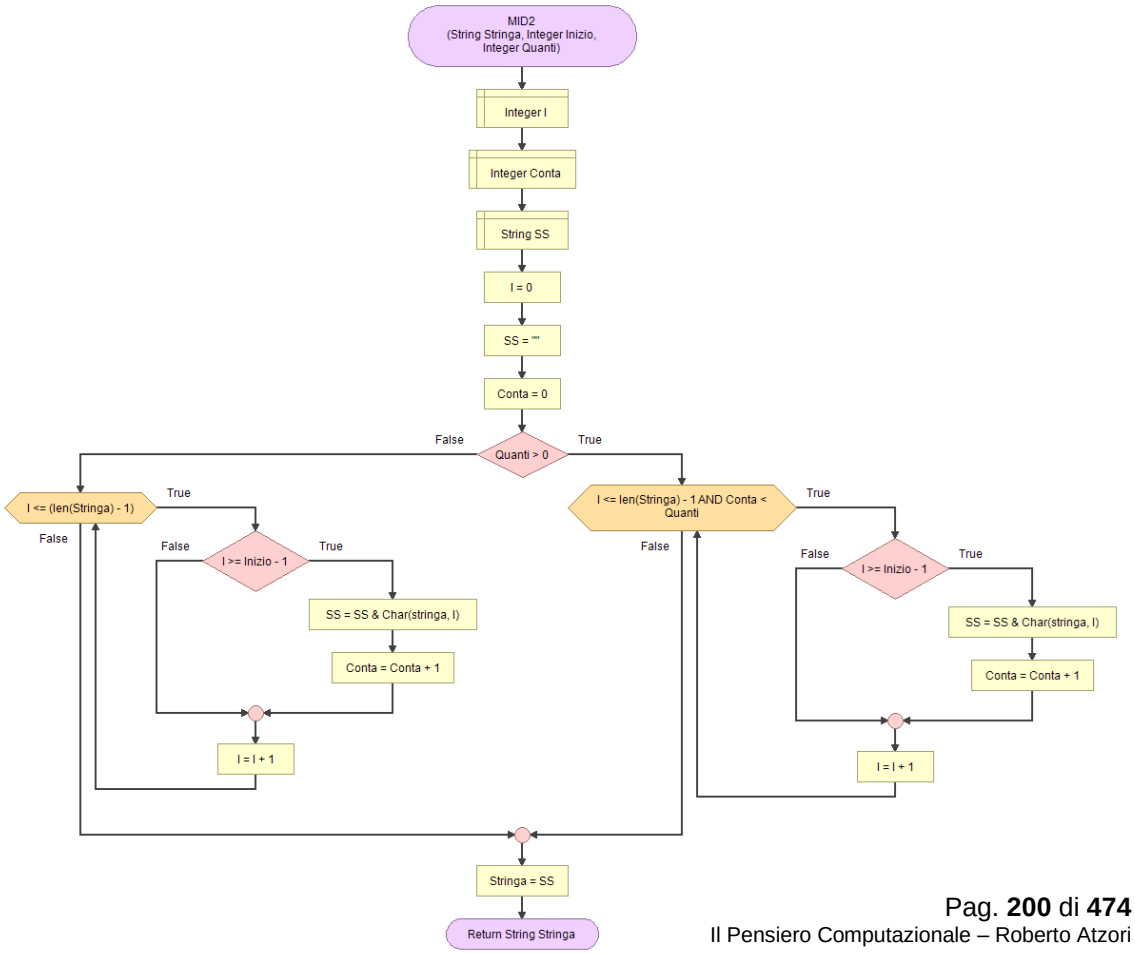

MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer l
Integer Conta
String SS
l = 0
SS = ""
Conta = 0
Quanti > 0
False
True
l <= (len(Stringa) - 1)
True
False
l >= Inizio - 1
False
True
SS = SS & Char(stringa, l)
Conta = Conta + 1
l = l + 1
l <= len(Stringa) - 1 AND Conta
< Quanti
True
False
l >= Inizio - 1
False
True
SS = SS & Char(stringa, l)
Conta = Conta + 1
l = l + 1
Stringa = SS
Return String Stringa

159.GSF ‖ Crittografia – Cifratura di Cesare (cifratura traslata)

Cryptografy - Caesar cipher (shift cipher)

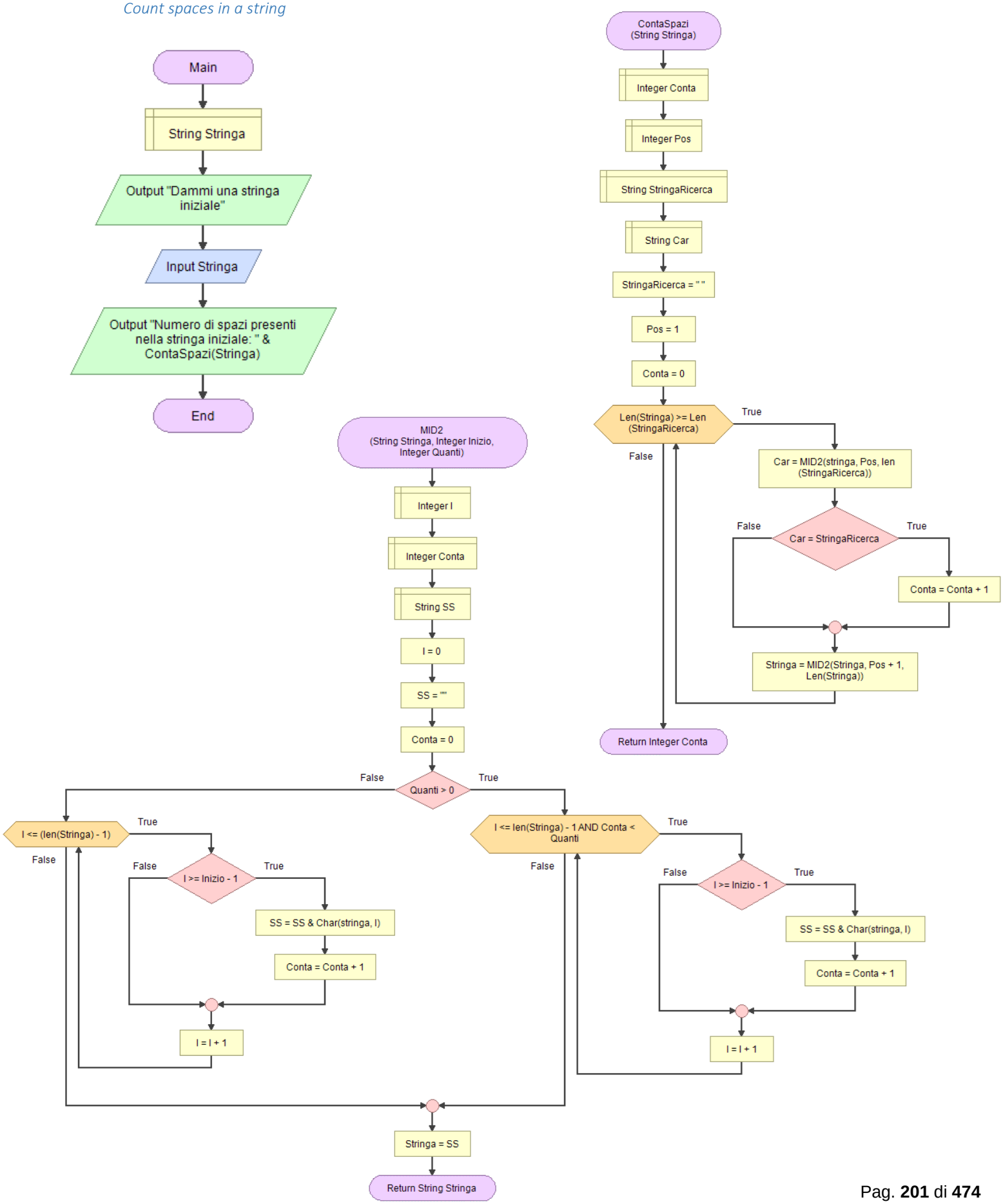



decipher
(String s, Integer k, Integer m)
Integer i
String ch
Integer CodeCh
String result
result = ""
i = 1 to len(s)
Next
Done
ch = MID2(s,i,1)
ToCode(ch) < 65 OR ToCode(ch) > 90
False
True
CodeCh = ToCode(ch)
CodeCh - k < 65
False
True
result = result & ToChar(97 + (CodeCh - 97 - k))
result = result & ToChar(97 + (CodeCh - 97 - k))
Return String result


MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

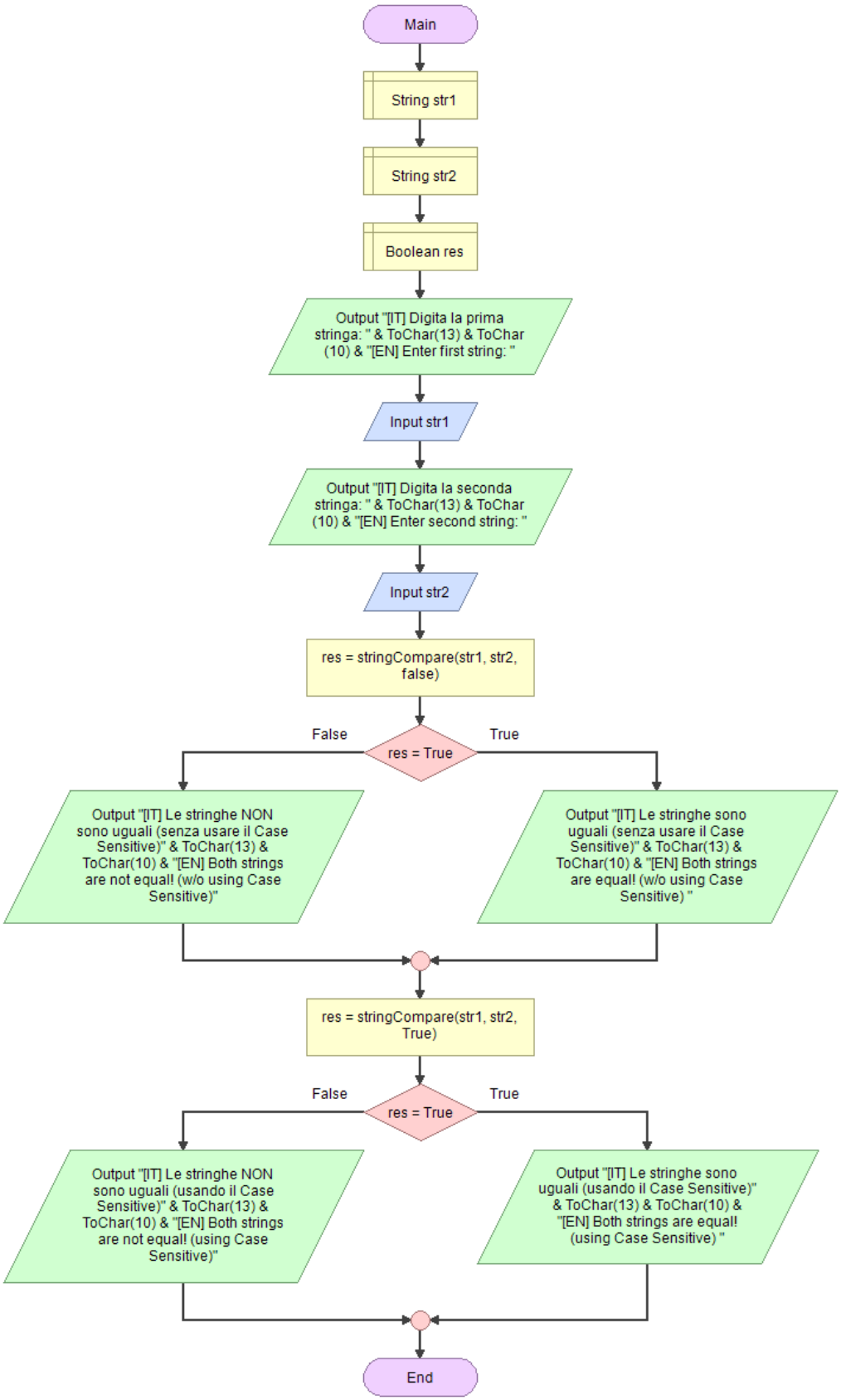
160. G S F | Data una stringa di caratteri contare gli spazi in essa contenuti

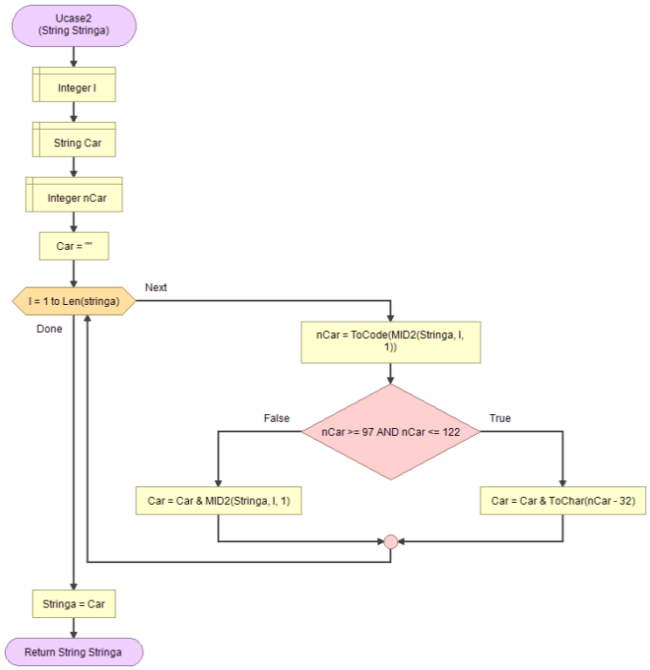
Count spaces in a string

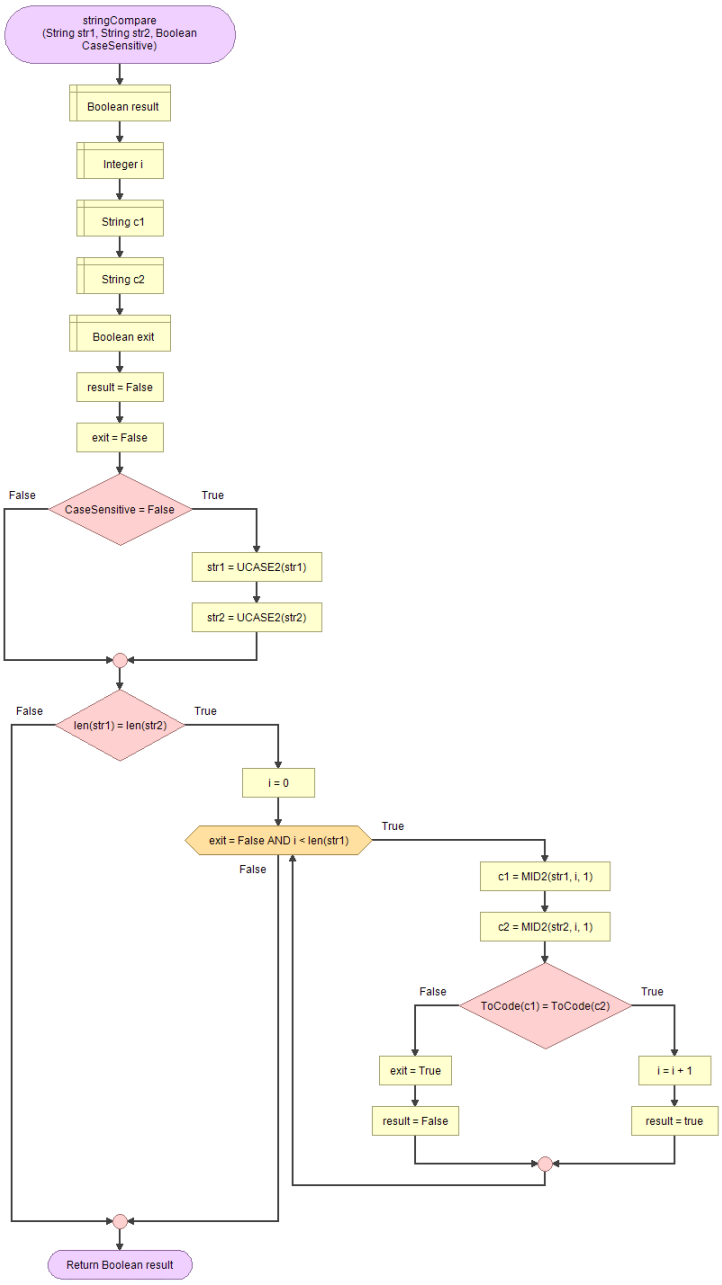


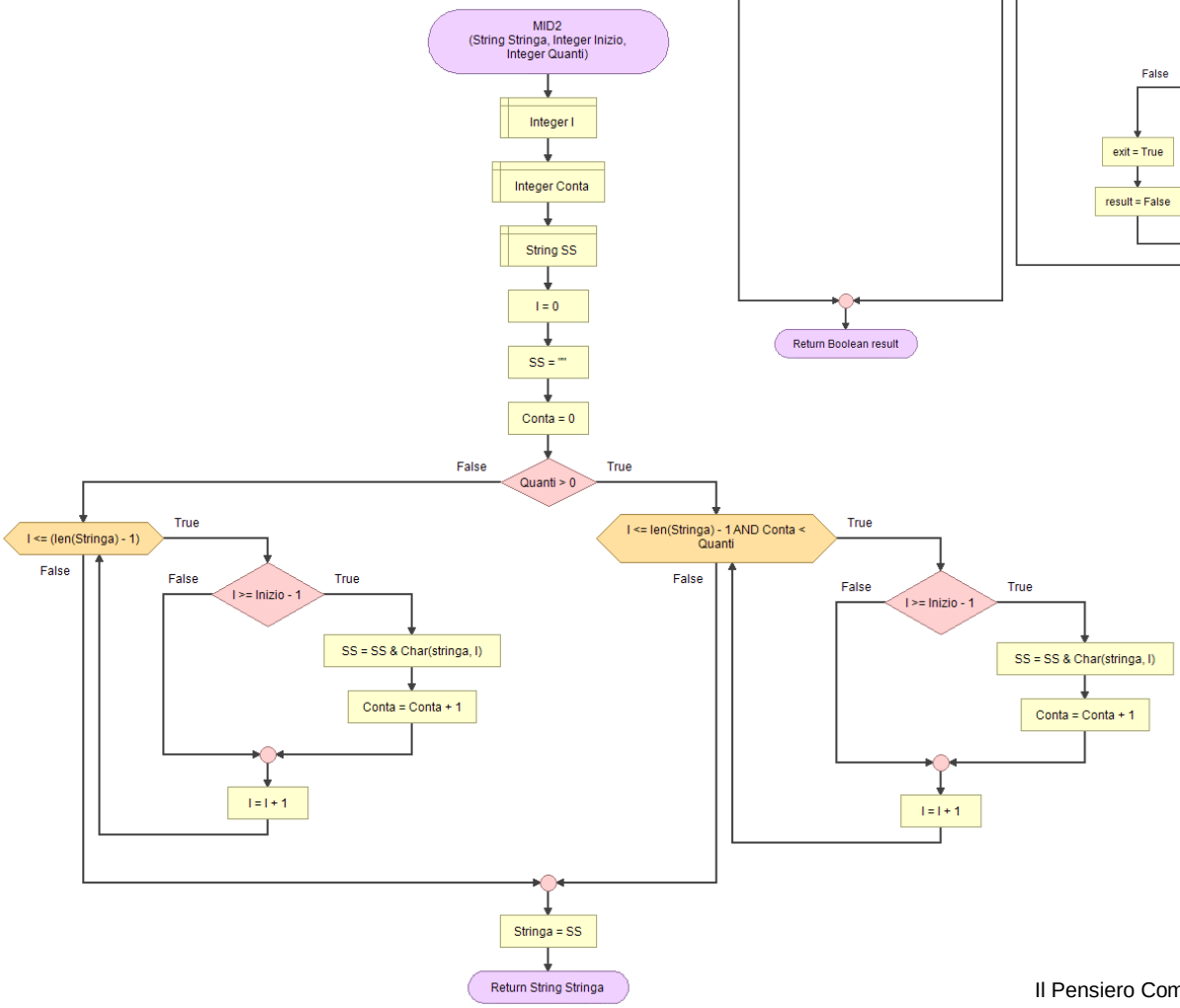
161.GSF ‖ Confronta due stringhe con o senza Case Sensitive

Compare two strings with or w/o Case Sensitive mode



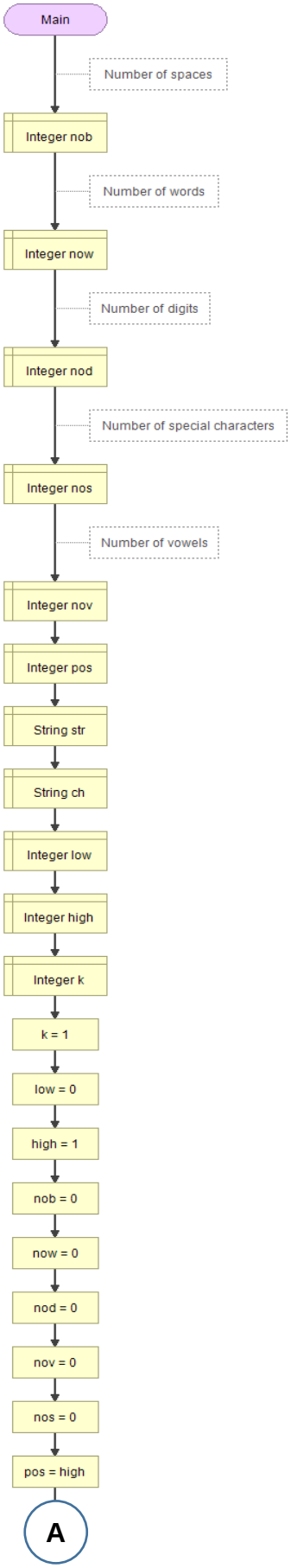

Ucase2
(String Stringa)
Integer I
String Car
Integer nCar
Car = ""
I = 1 to Len(stringa)
Next
Done
nCar = ToCode(MID2(Stringa, I, 1))
nCar >= 97 AND nCar <= 122
False
True
Car = Car & MID2(Stringa, I, 1)
Car = Car & ToChar(nCar - 32)
Stringa = Car
Return String Stringa

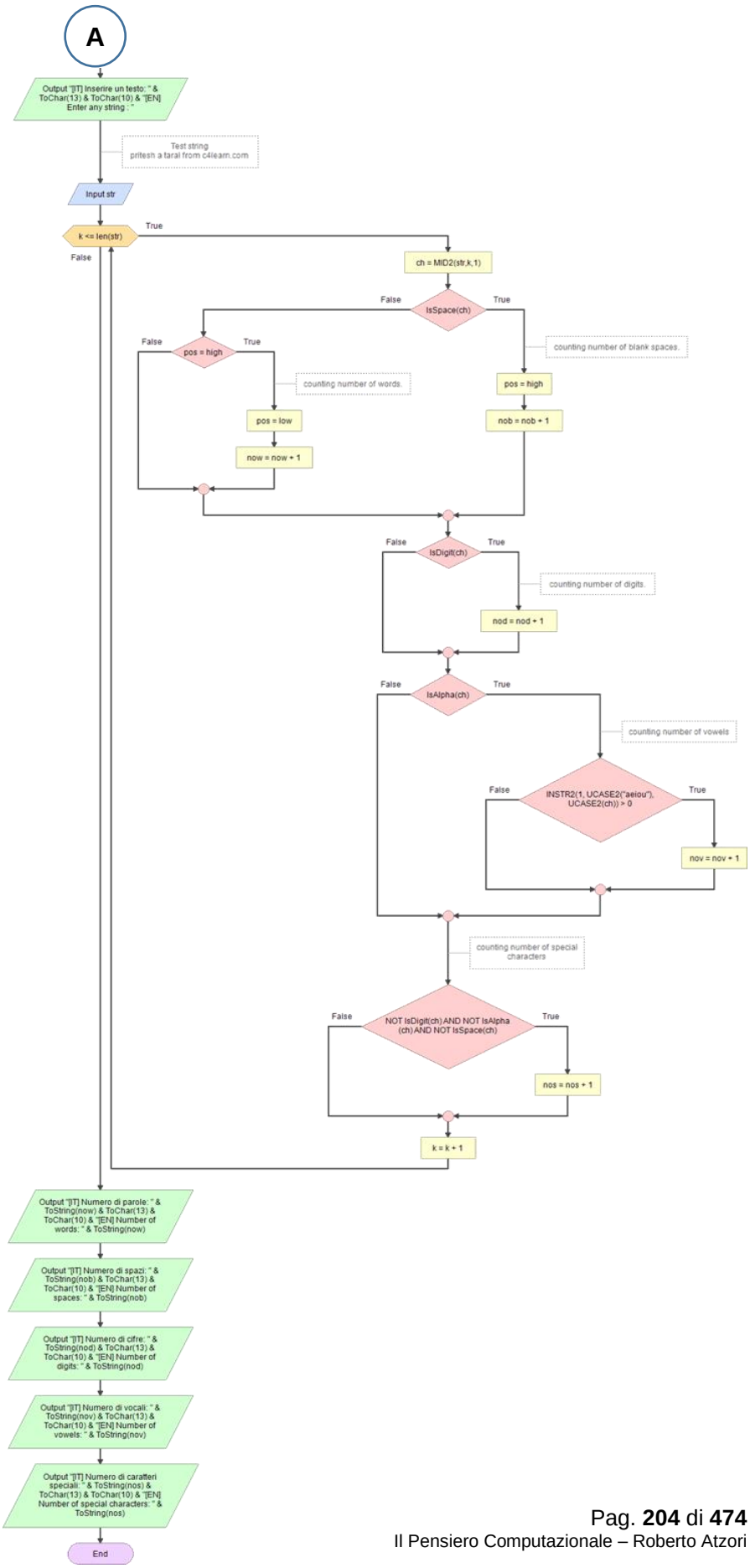

stringCompare
(String str1, String str2, Boolean CaseSensitive)
Boolean result
Integer i
String c1
String c2
Boolean exit
result = False
exit = False
CaseSensitive = False
False
True
str1 = UCASE2(str1)
str2 = UCASE2(str2)
len(str1) = len(str2)
False
True
i = 0
exit = False AND i < len(str1)
True
False
c1 = MID2(str1, i, 1)
c2 = MID2(str2, i, 1)
ToCode(c1) = ToCode(c2)
False
True
exit = True
result = False
i = i + 1
result = true
Return Boolean result

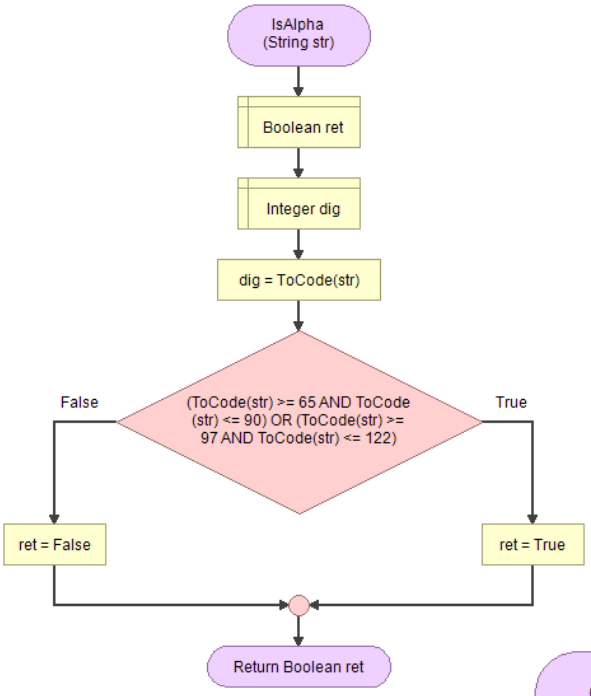

MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

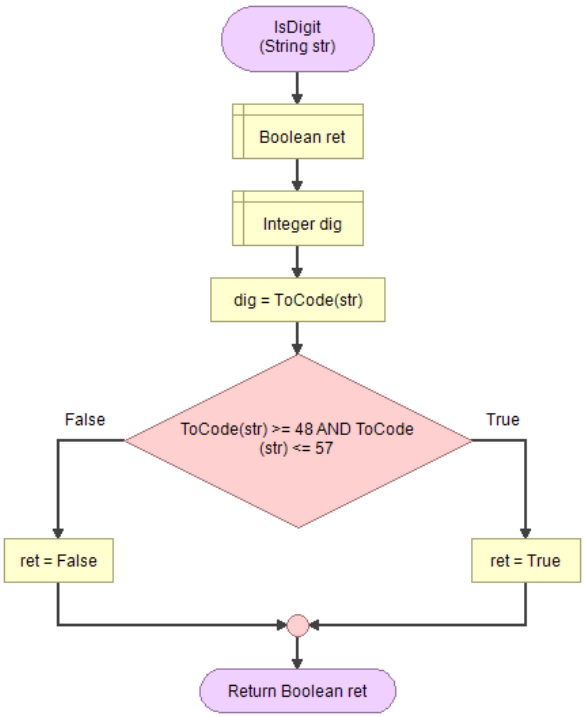
162.GSF | Conta numero di parole, cifre, vocali e caratteri speciali

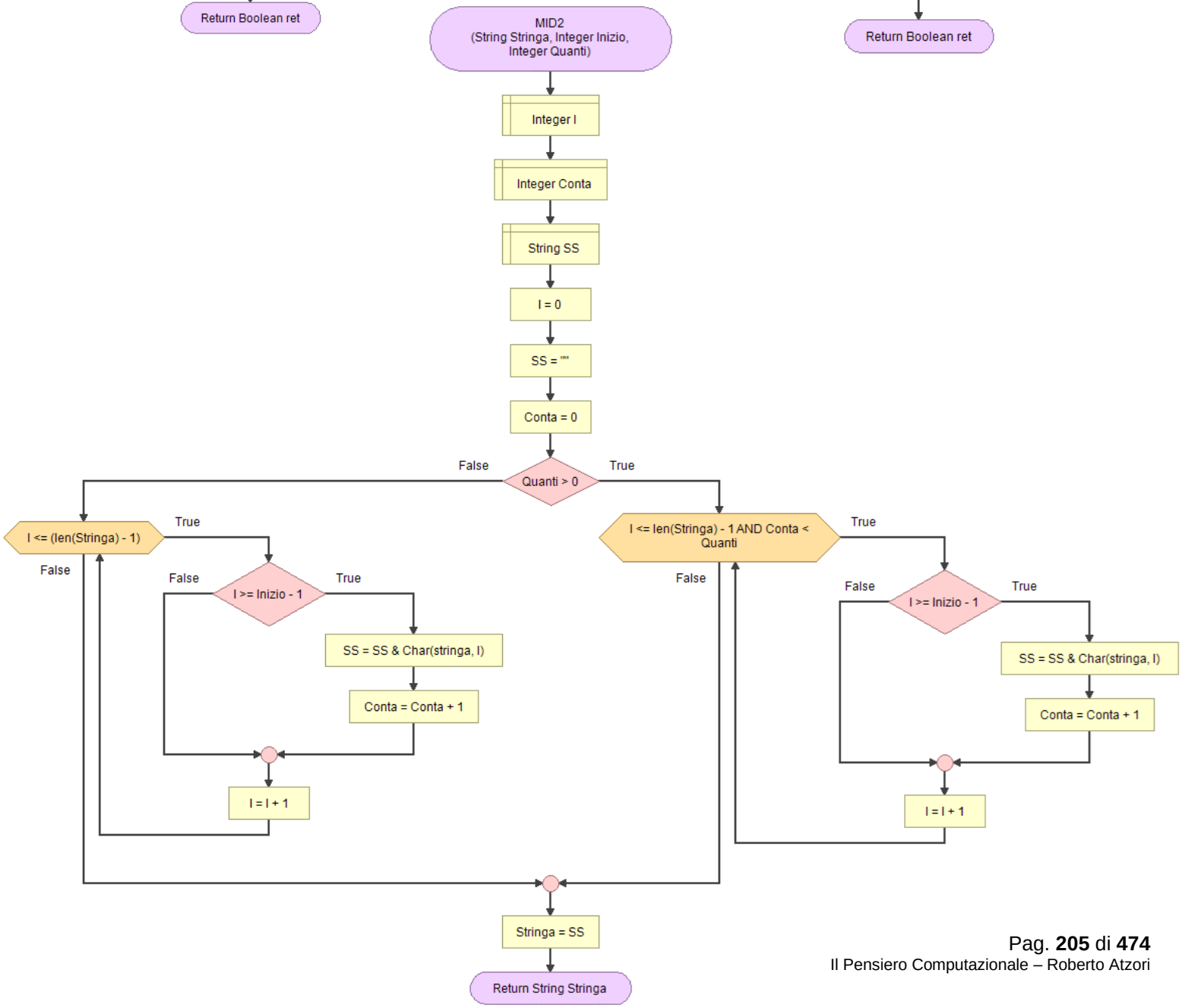
Count number of words,digits,vowels, spaces and special characters

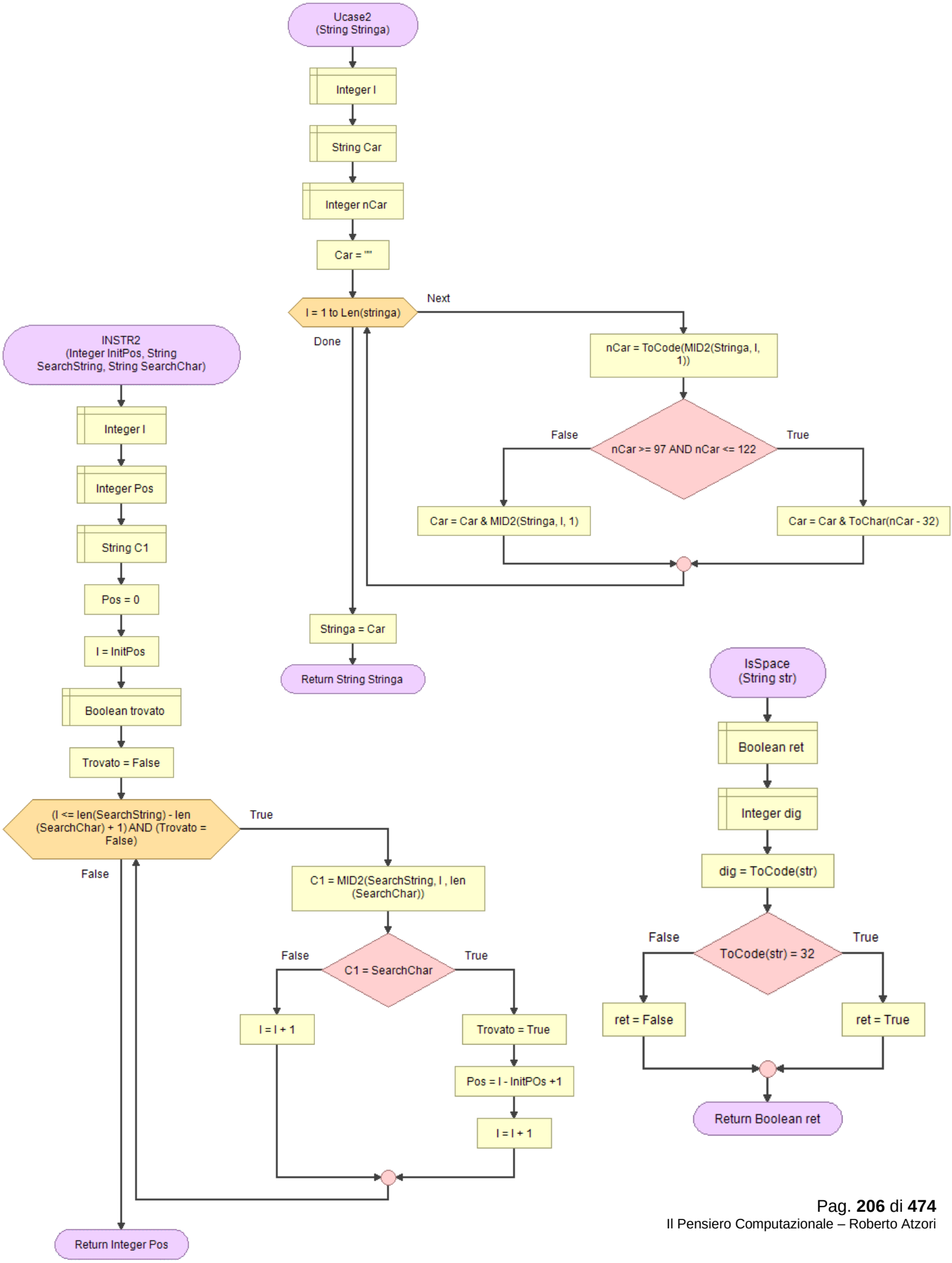





IsAlpha
(String str)
Boolean ret
Integer dig
dig = ToCode(str)
(ToCode(str) >= 65 AND ToCode (str) <= 90) OR (ToCode(str) >= 97 AND ToCode(str) <= 122)
False
True
ret = False
ret = True
Return Boolean ret

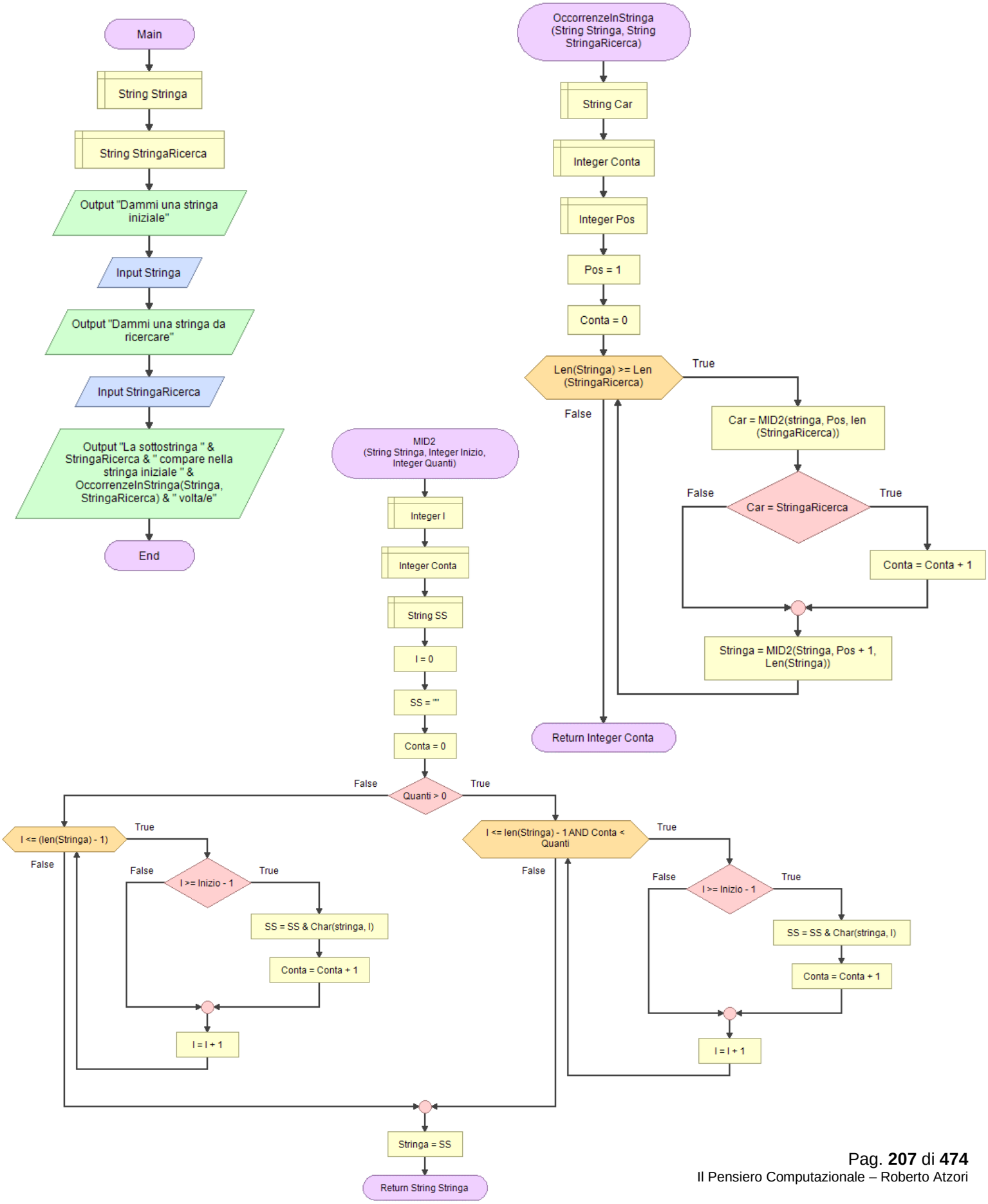

IsDigit
(String str)
Boolean ret
Integer dig
dig = ToCode(str)
ToCode(str) >= 48 AND ToCode (str) <= 57
False
True
ret = False
ret = True
Return Boolean ret


MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa


Ucase2
(String Stringa)
Integer I
String Car
Integer nCar
Car = ""
I = 1 to Len(stringa)
Next
Done
nCar = ToCode(MID2(Stringa, I, 1))
nCar >= 97 AND nCar <= 122
False
True
Car = Car & MID2(Stringa, I, 1)
Car = Car & ToChar(nCar - 32)
Stringa = Car
Return String Stringa
INSTR2
(Integer InitPos, String SearchString, String SearchChar)
Integer I
Integer Pos
String C1
Pos = 0
I = InitPos
Boolean trovato
Trovato = False
(I <= len(SearchString) - len (SearchChar) + 1) AND (Trovato = False)
True
False
C1 = MID2(SearchString, I , len (SearchChar))
C1 = SearchChar
False
True
I = I + 1
Trovato = True
Pos = I - InitPOs +1
I = I + 1
Return Integer Pos
IsSpace
(String str)
Boolean ret
Integer dig
dig = ToCode(str)
ToCode(str) = 32
False
True
ret = False
ret = True
Return Boolean ret

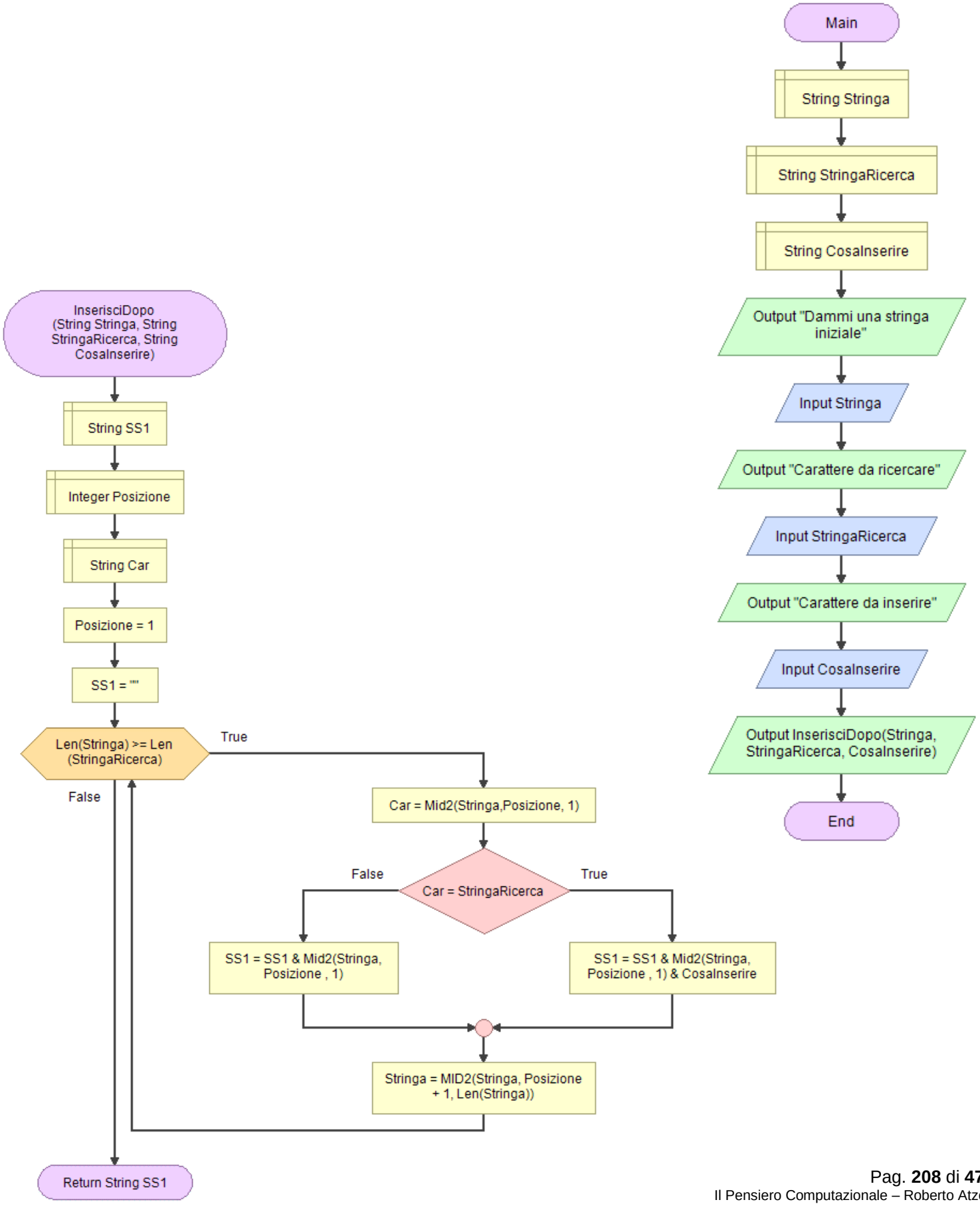
163. **G S F** | Data una stringa di caratteri dire quante volte è contenuta in essa una data sottostringa

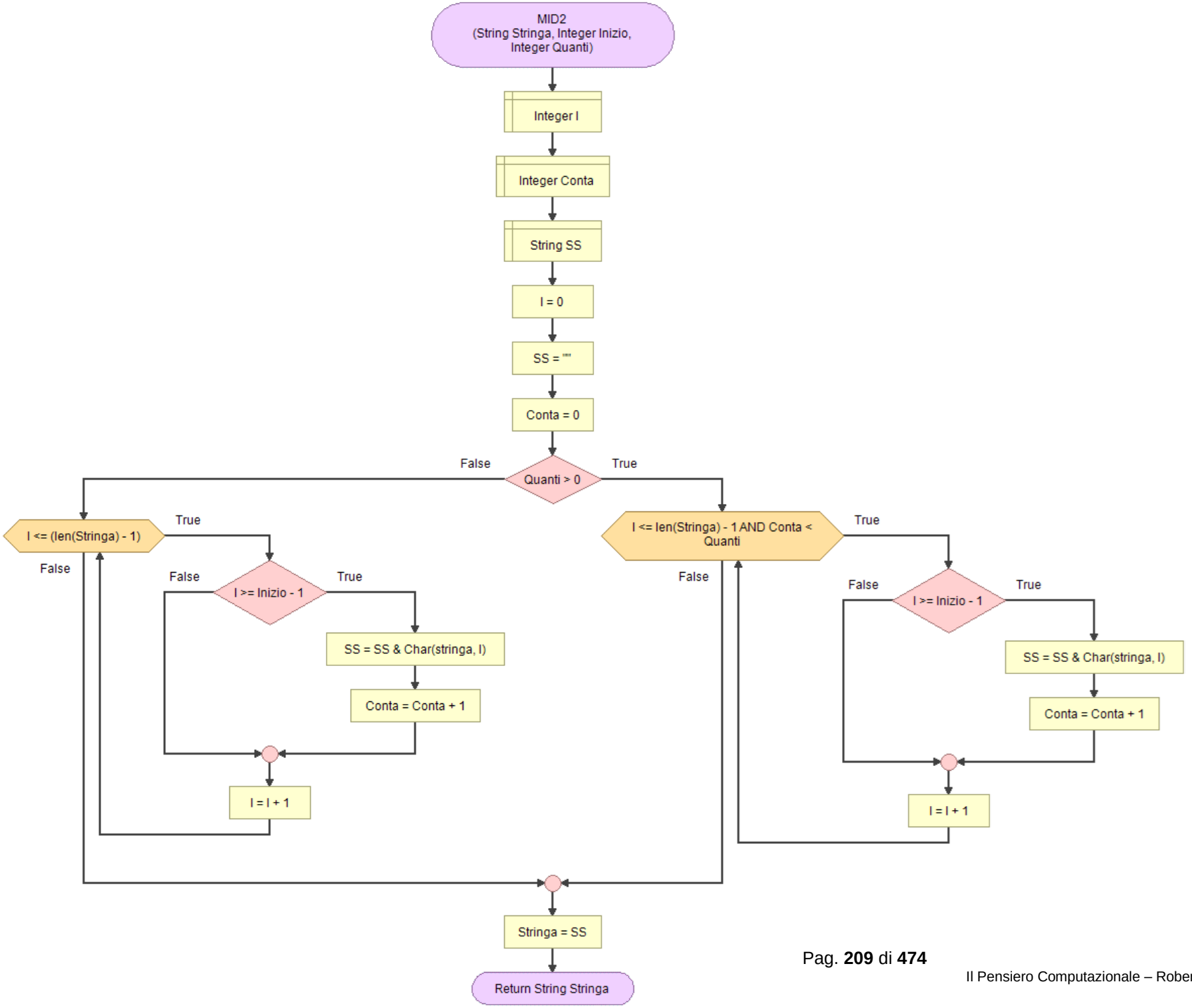
Counts substring occurrences in a string



164. **G S F** ‖ Data una stringa di caratteri inserire un carattere dopo una data lettera

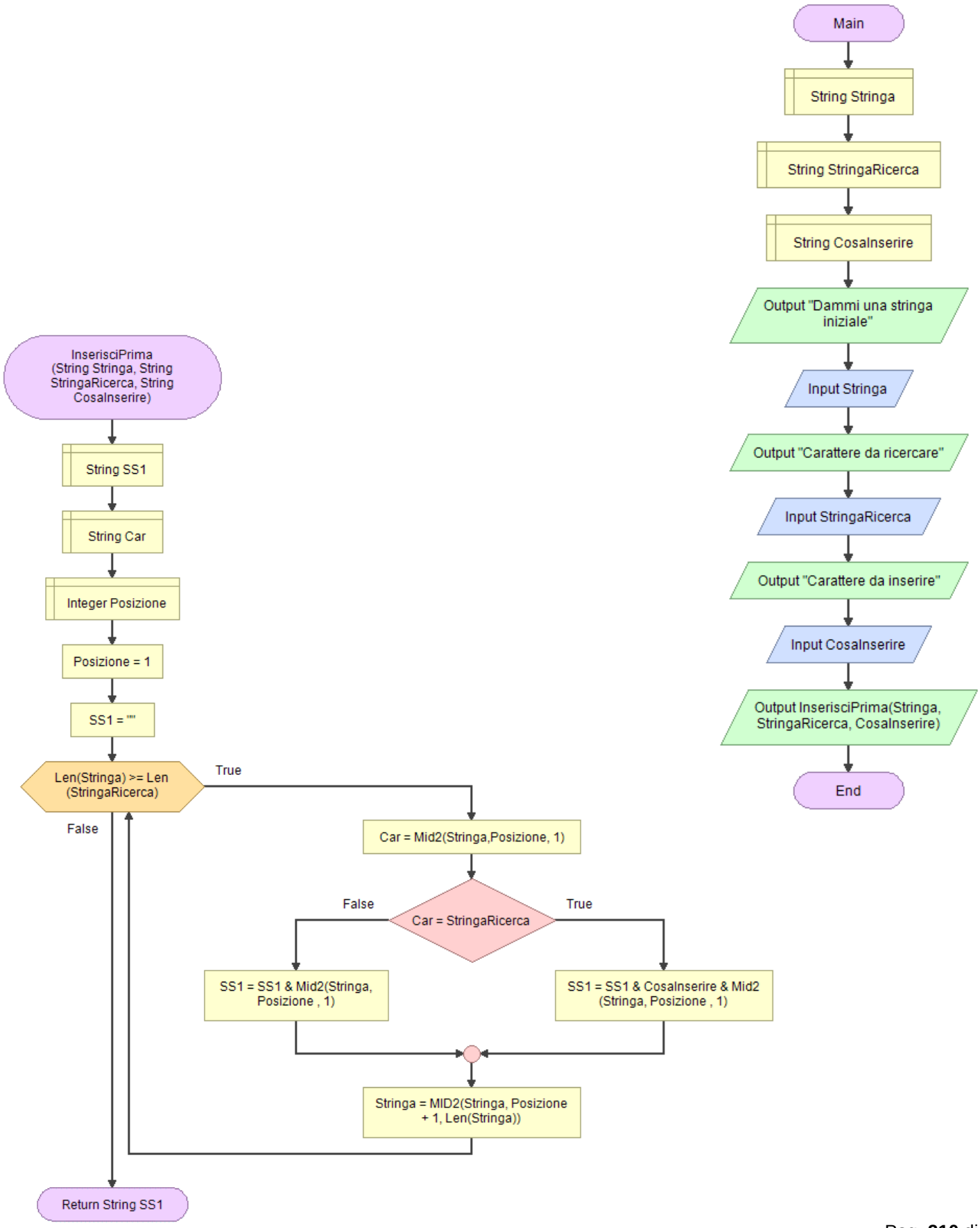
Inserts characters (substring) after a searched single char

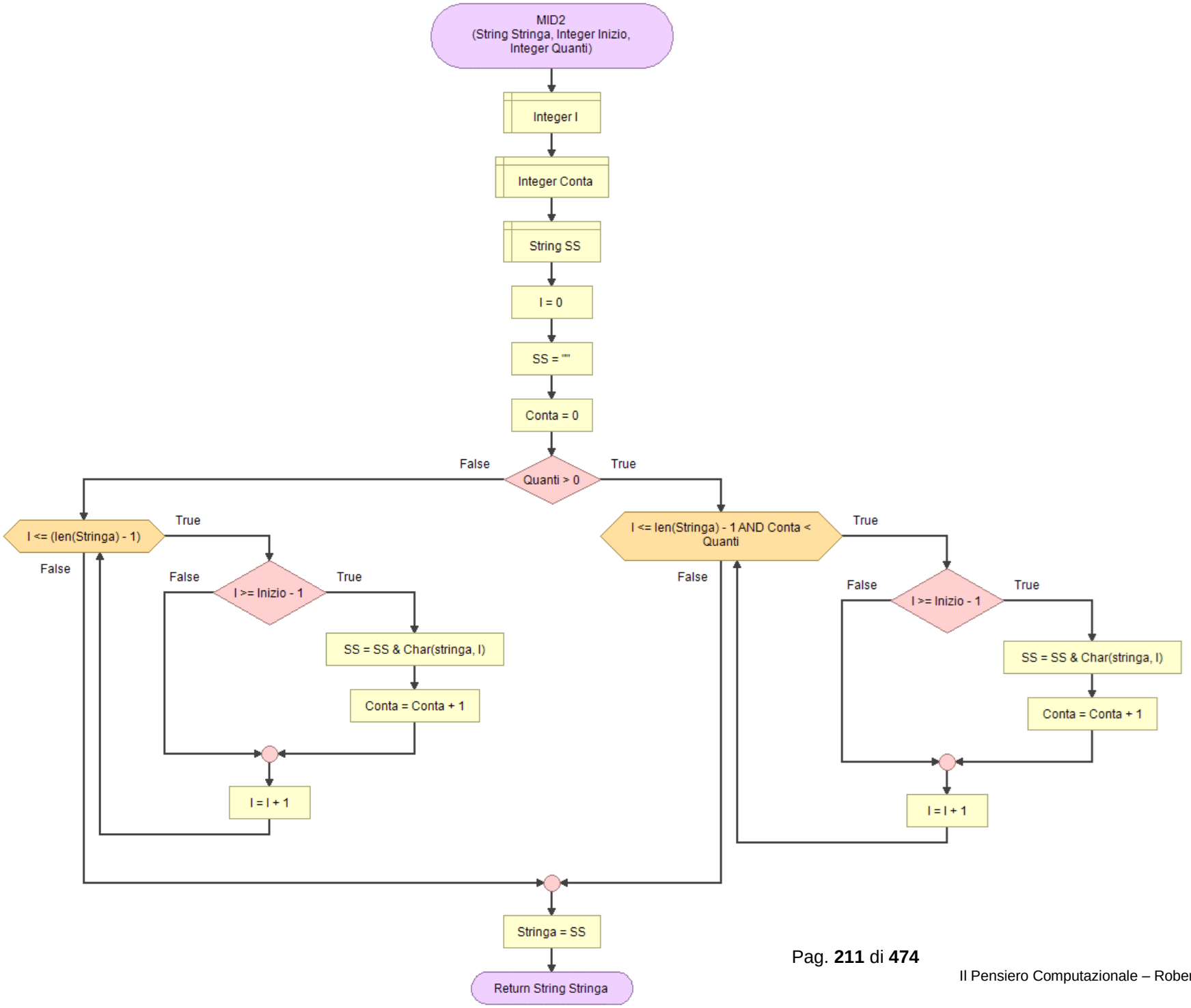



MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

165. **G S F** ‖ Data una stringa di caratteri inserire un carattere prima di una data lettera

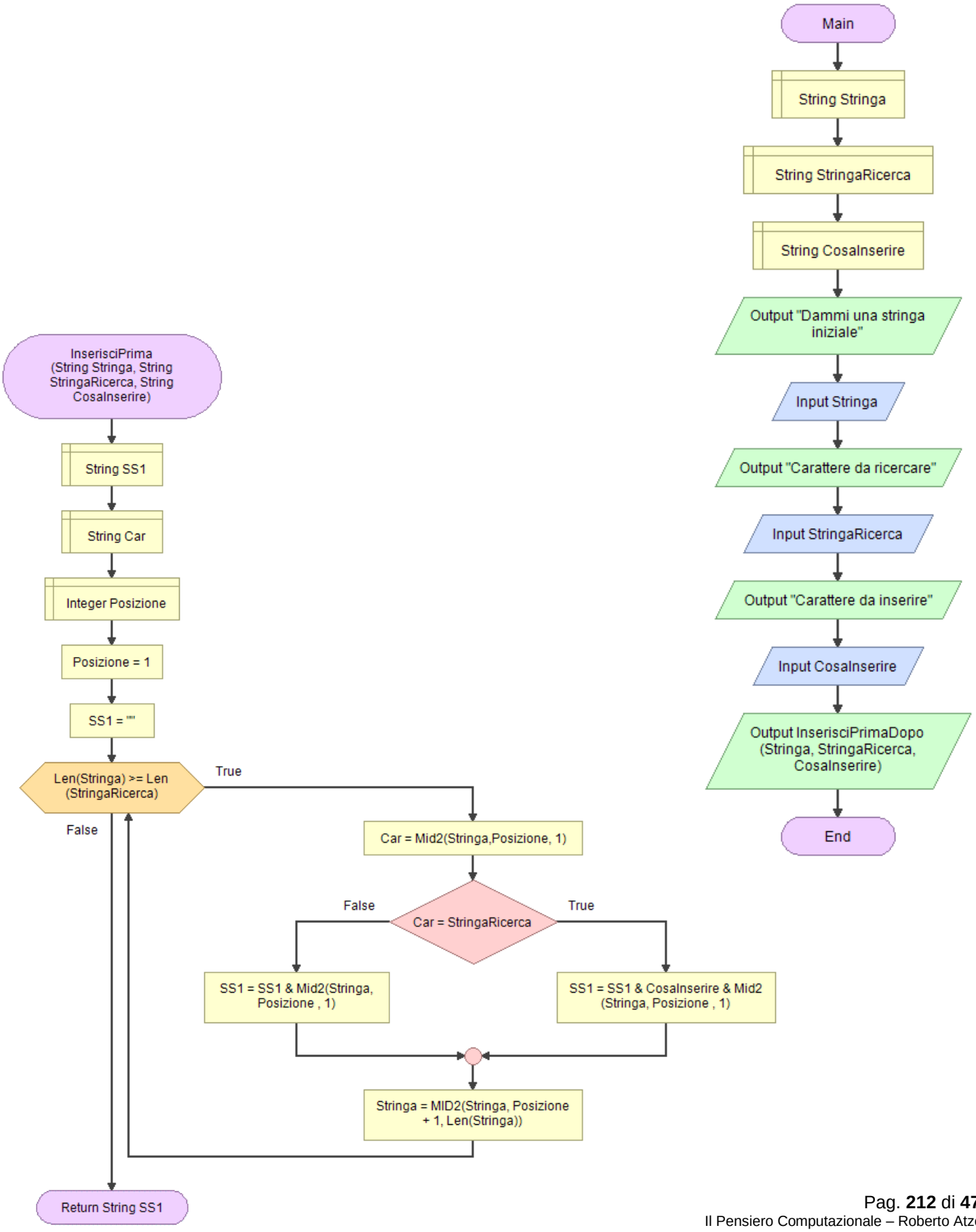
Inserts characters (substring) before a searched single char

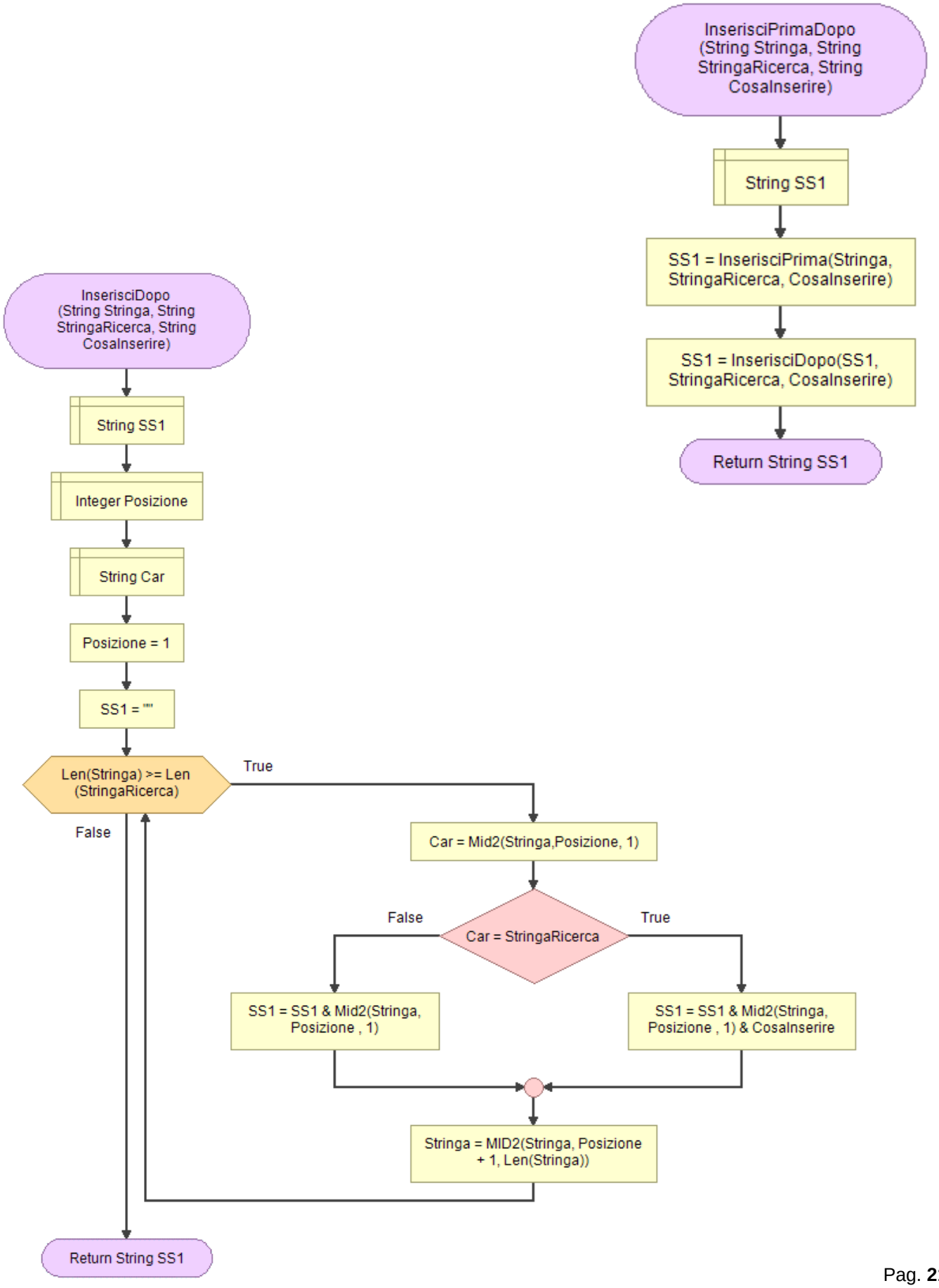


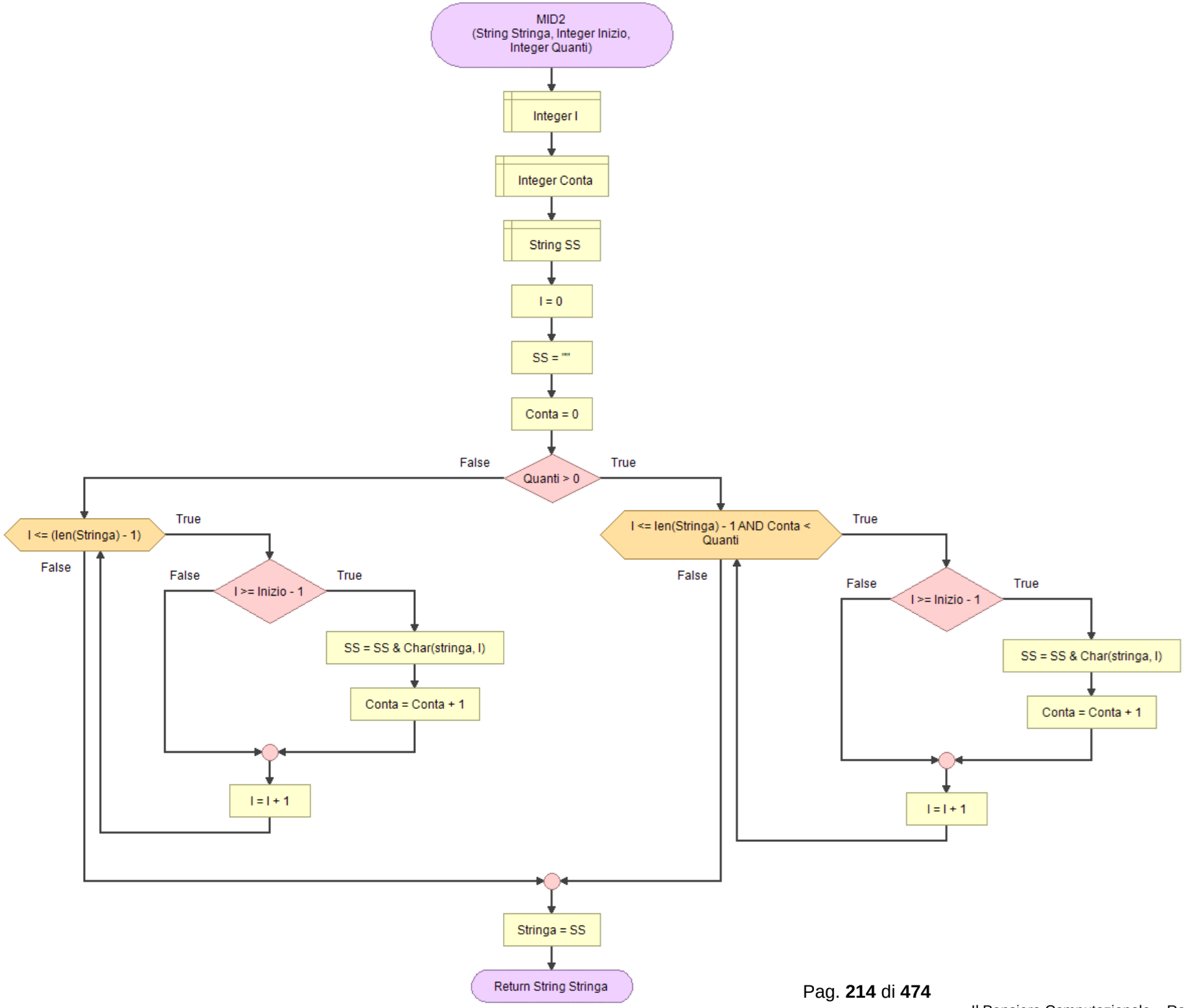

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= len(Stringa) - 1
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

166. **G S F** ‖ Data una stringa di caratteri inserire un carattere prima e dopo una data lettera

Inserts characters (substring) after and before a searched single char

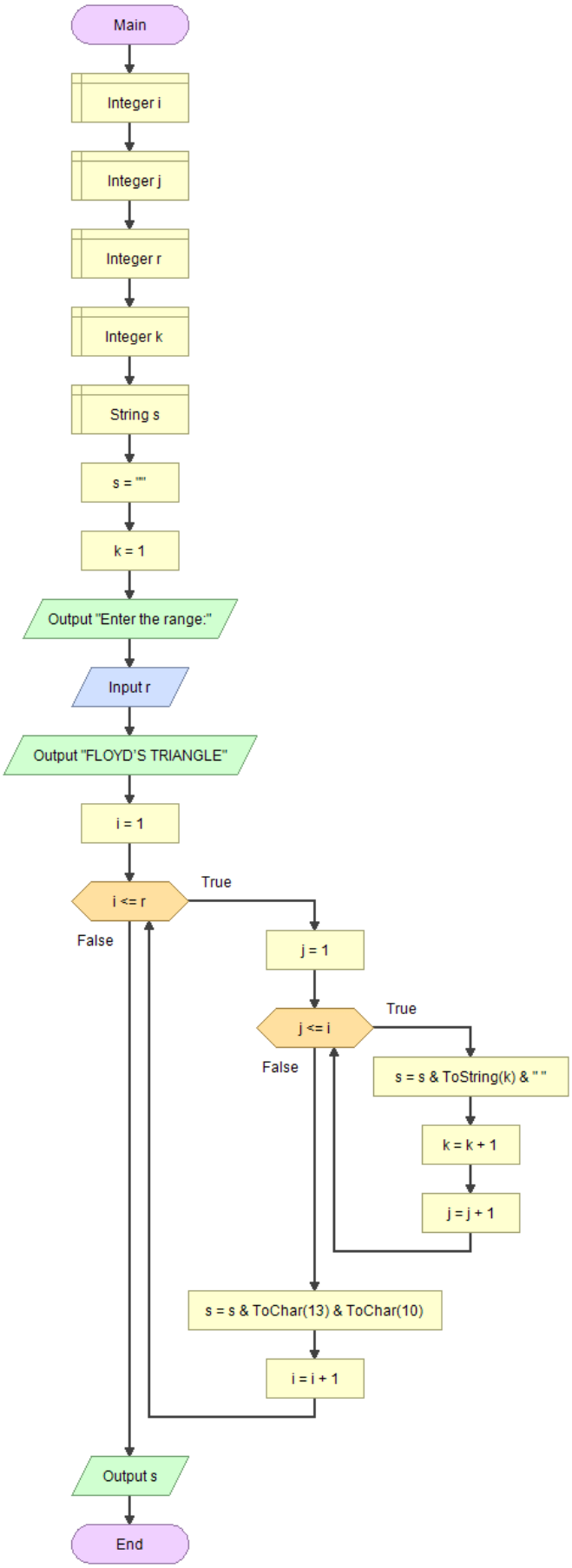



InserisciPrimaDopo (String Stringa, String StringaRicerca, String CosaInserire)
String SS1
SS1 = InserisciPrima(Stringa, StringaRicerca, CosaInserire)
SS1 = InserisciDopo(SS1, StringaRicerca, CosaInserire)
Return String SS1
InserisciDopo (String Stringa, String StringaRicerca, String CosaInserire)
String SS1
Integer Posizione
String Car
Posizione = 1
SS1 = ""
Len(Stringa) >= Len (StringaRicerca)
True
False
Car = Mid2(Stringa,Posizione, 1)
False
Car = StringaRicerca
True
SS1 = SS1 & Mid2(Stringa, Posizione , 1)
SS1 = SS1 & Mid2(Stringa, Posizione , 1) & CosaInserire
Stringa = MID2(Stringa, Posizione + 1, Len(Stringa))
Return String SS1


MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

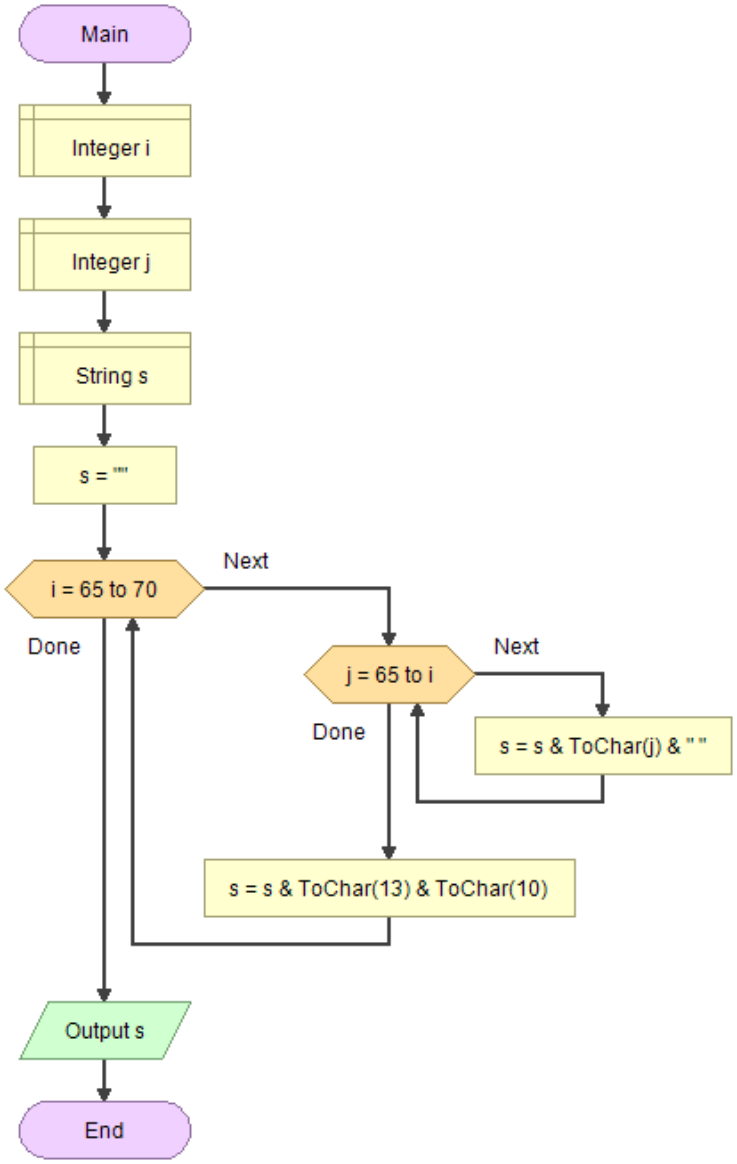
167.G S ‖ Triangolo di Floyd con numeri

Print numerical as Floyd's triangle pattern



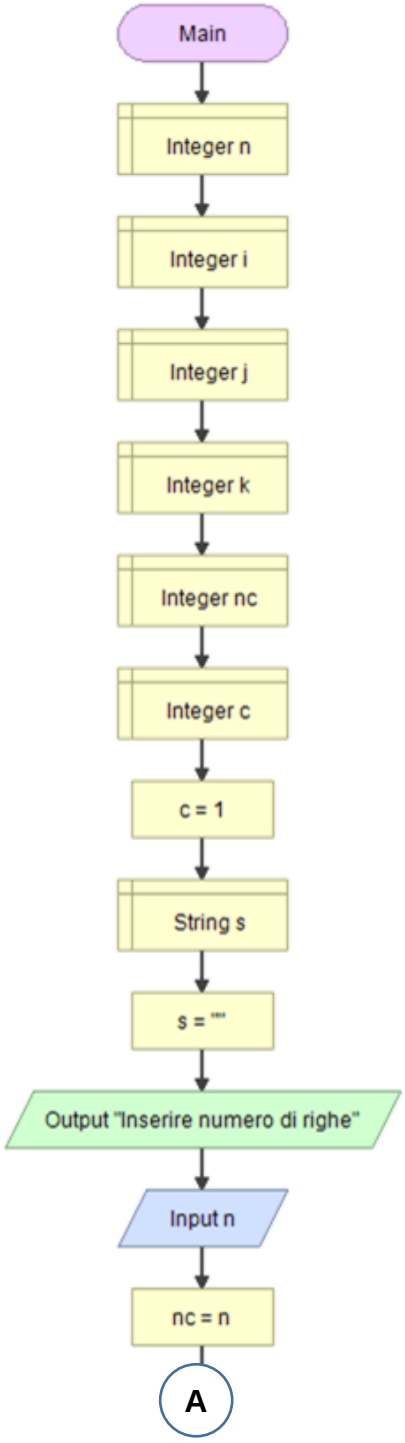
168. G S ‖ Triangolo di Floyd con caratteri

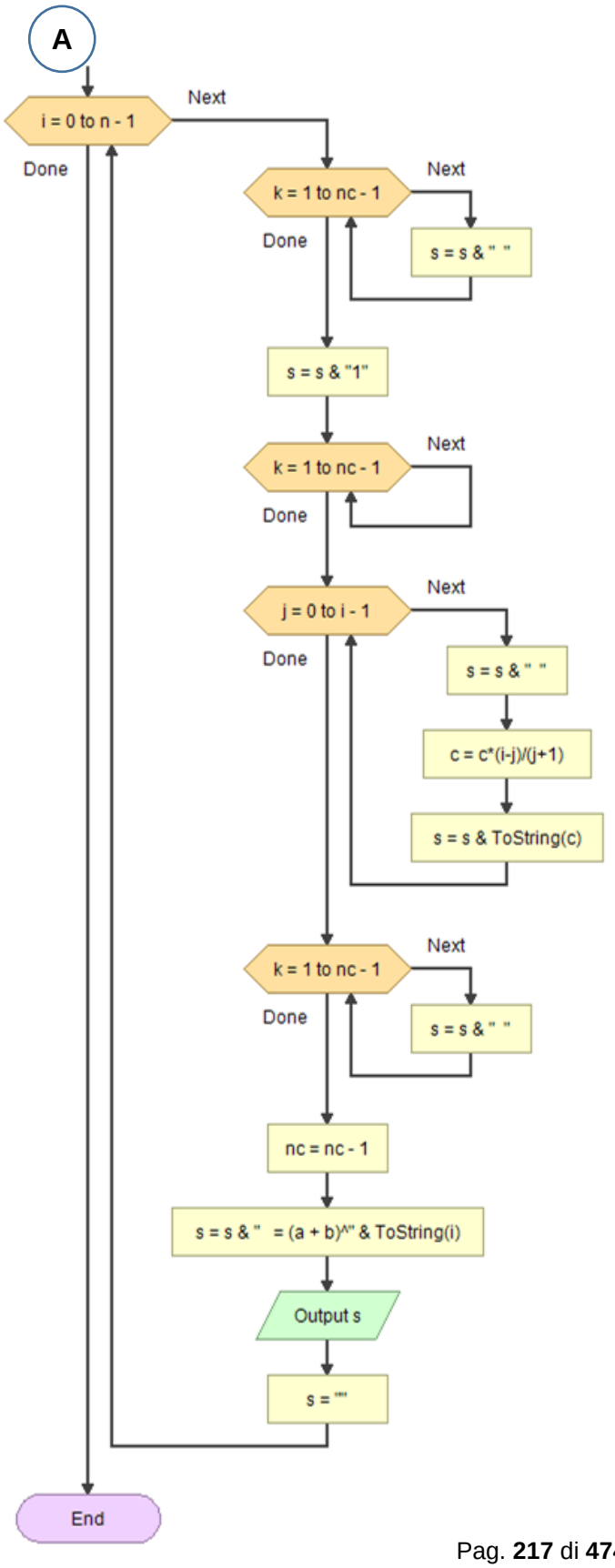
Print character as Floyd's triangle pattern



169.GS ‖ Triangolo di Tartaglia

Pascal's Triangle





170.GSF | Alfabeto Farfallino

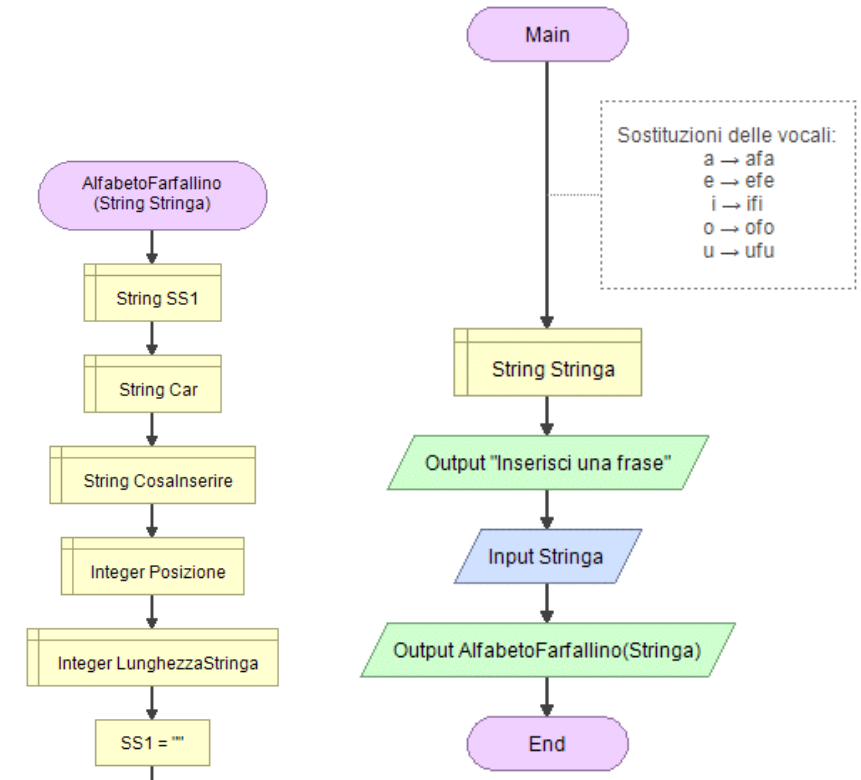
The farfallino alphabet

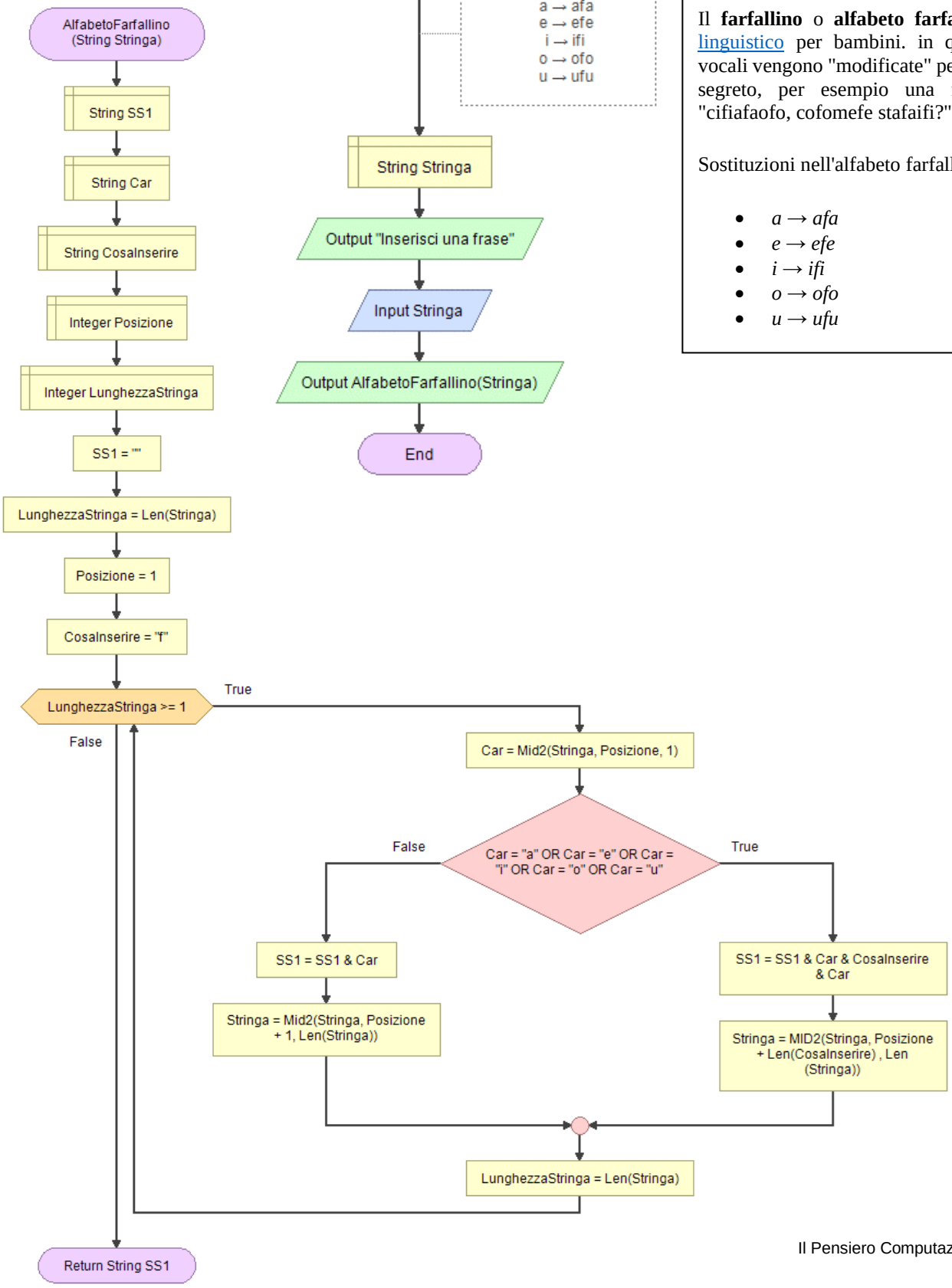
The farfallino alphabet (in Italian "Alfabeto Farfallino") is a language game used primarily in Italy. The usual rules for farfallino alphabet are based on the substitution of each vowel with a 3 letter sequence where the vowel itself is repeated with an interceding f.

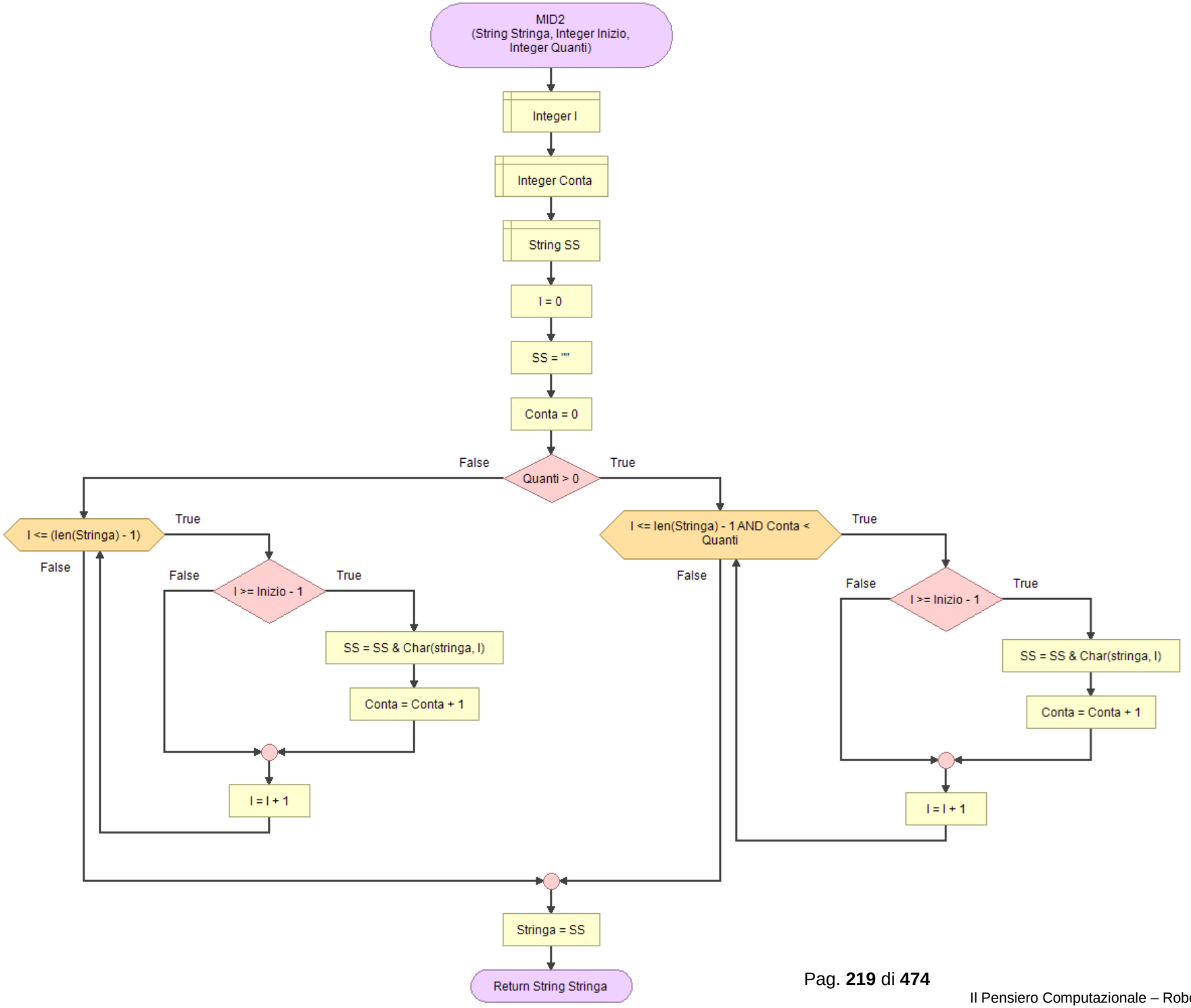
Il **farfallino** o **alfabeto farfallino** è un gioco linguistico per bambini. in questo alfabeto le vocali vengono "modificate" per parlare in codice segreto, per esempio una frase comune è: "cifiafaofo, cofomefe stafaifi?"

Sostituzioni nell'alfabeto farfallino:

- *a → afa*
- *e → efe*
- *i → ifi*
- *o → ofo*
- *u → ufu*

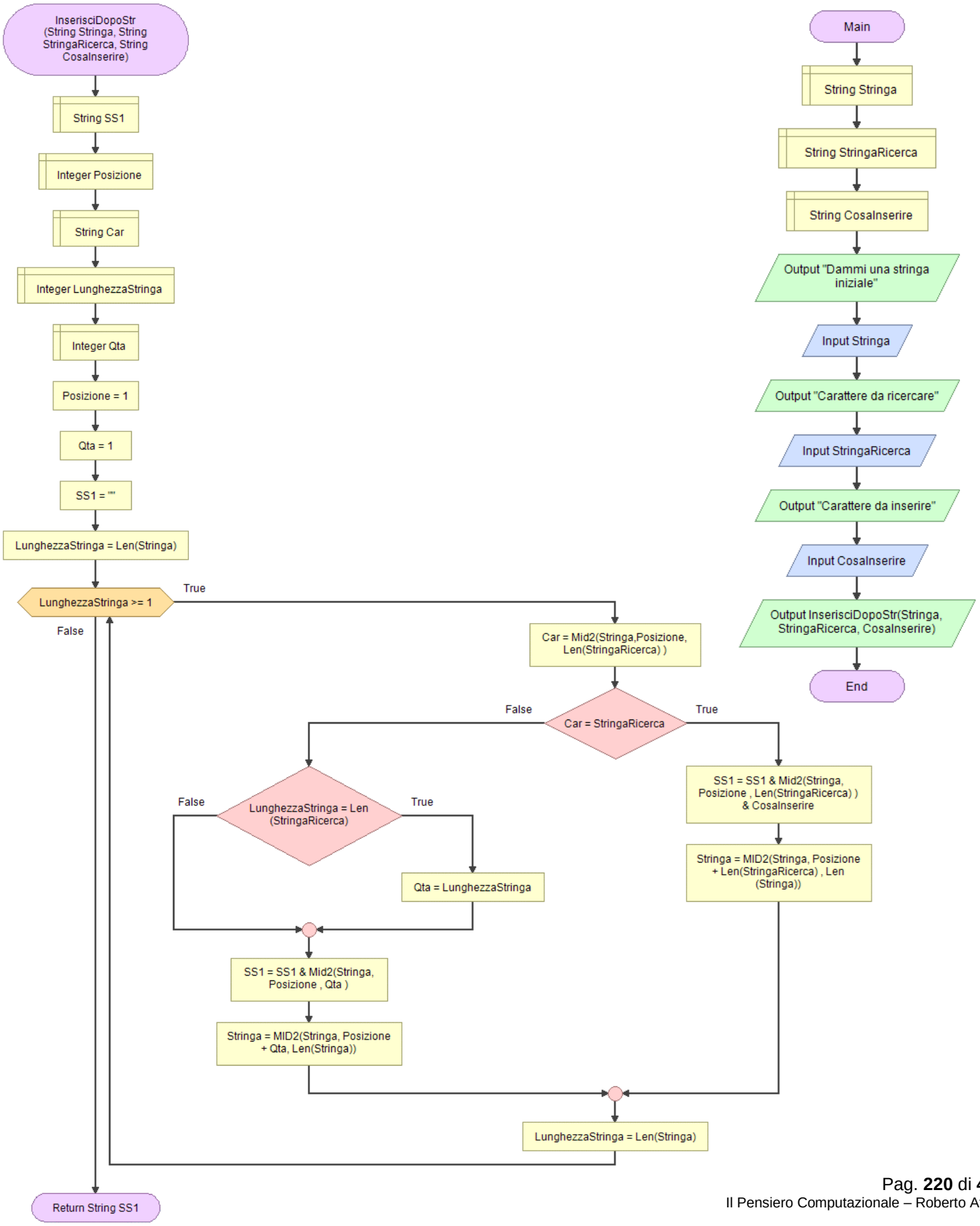


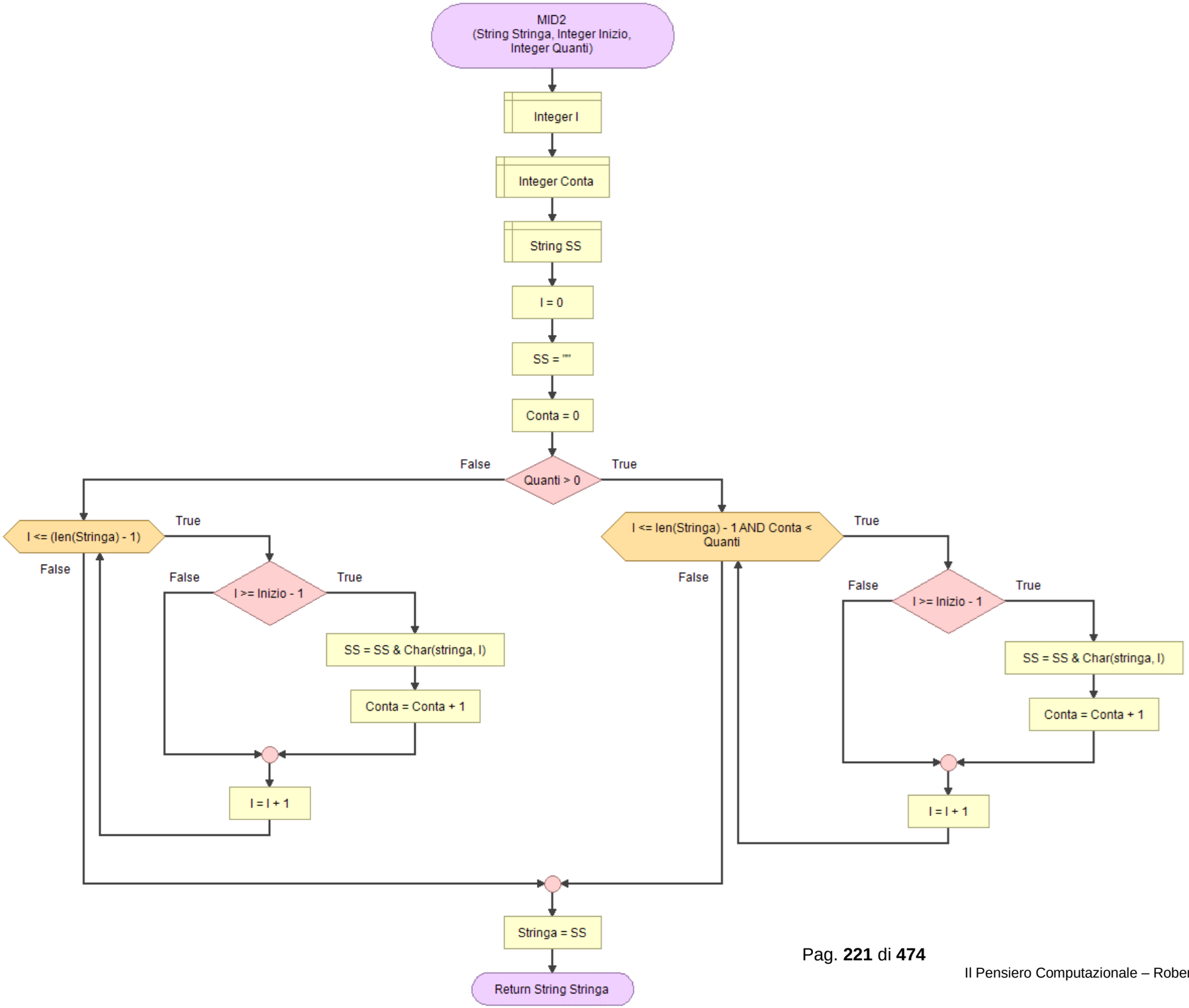



MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

171.**G S F** ‖ Data una stringa di caratteri inserire uno o più caratteri dopo una data sottostringa

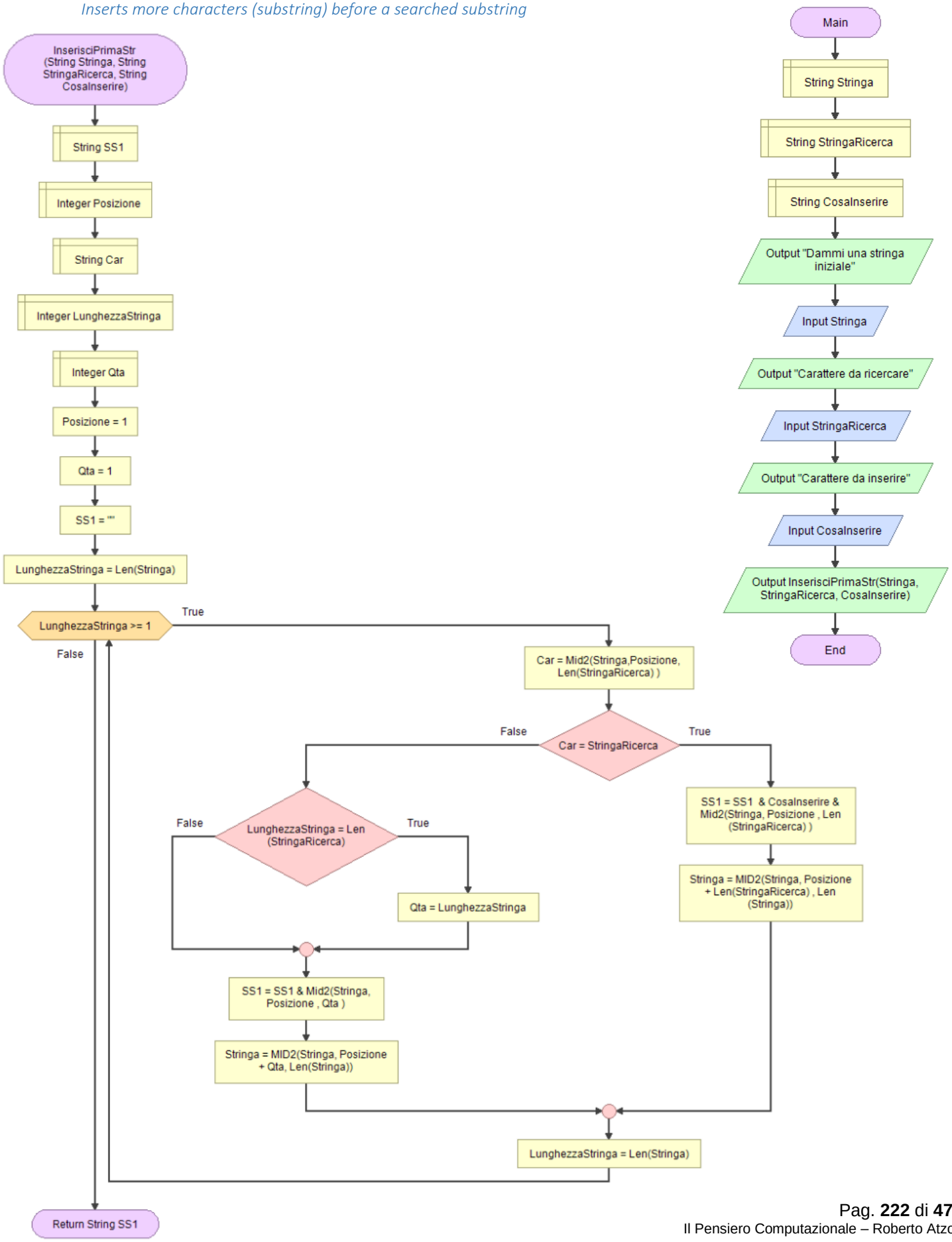
Inserts more characters (substring) after a searched substring

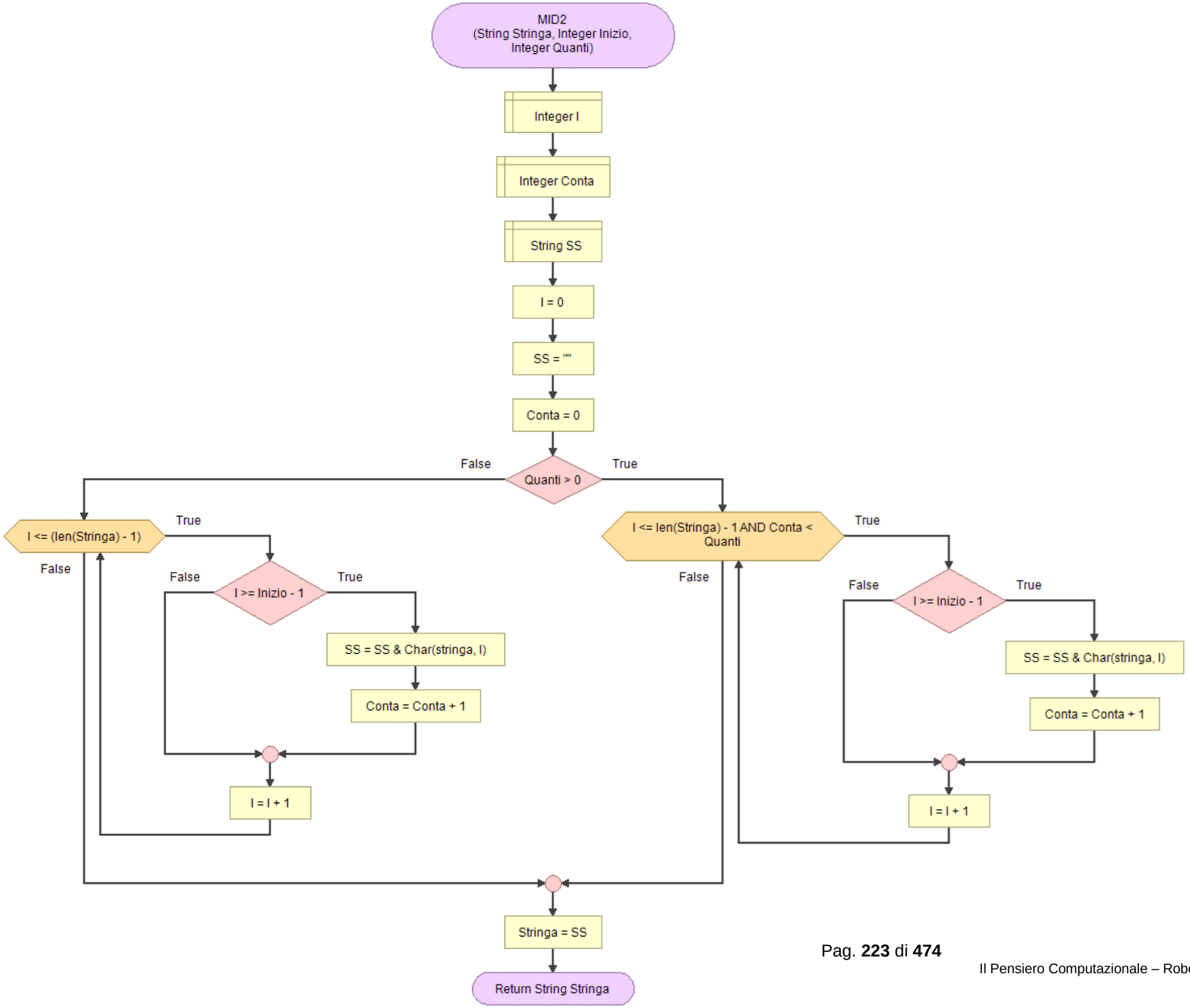



MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

172.**G S F** ‖ Data una stringa di caratteri inserire uno o più caratteri prima di una data sottostringa

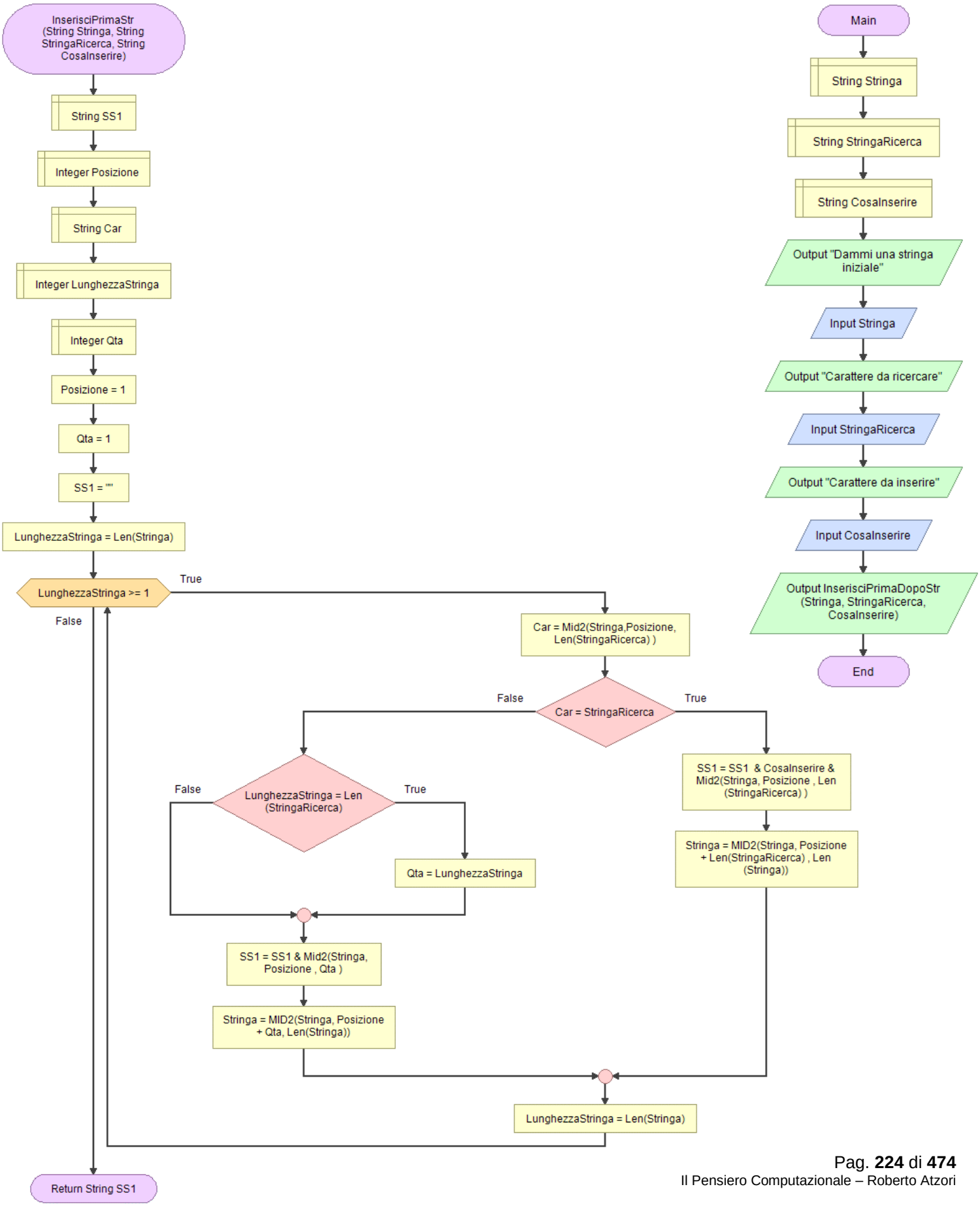
Inserts more characters (substring) before a searched substring

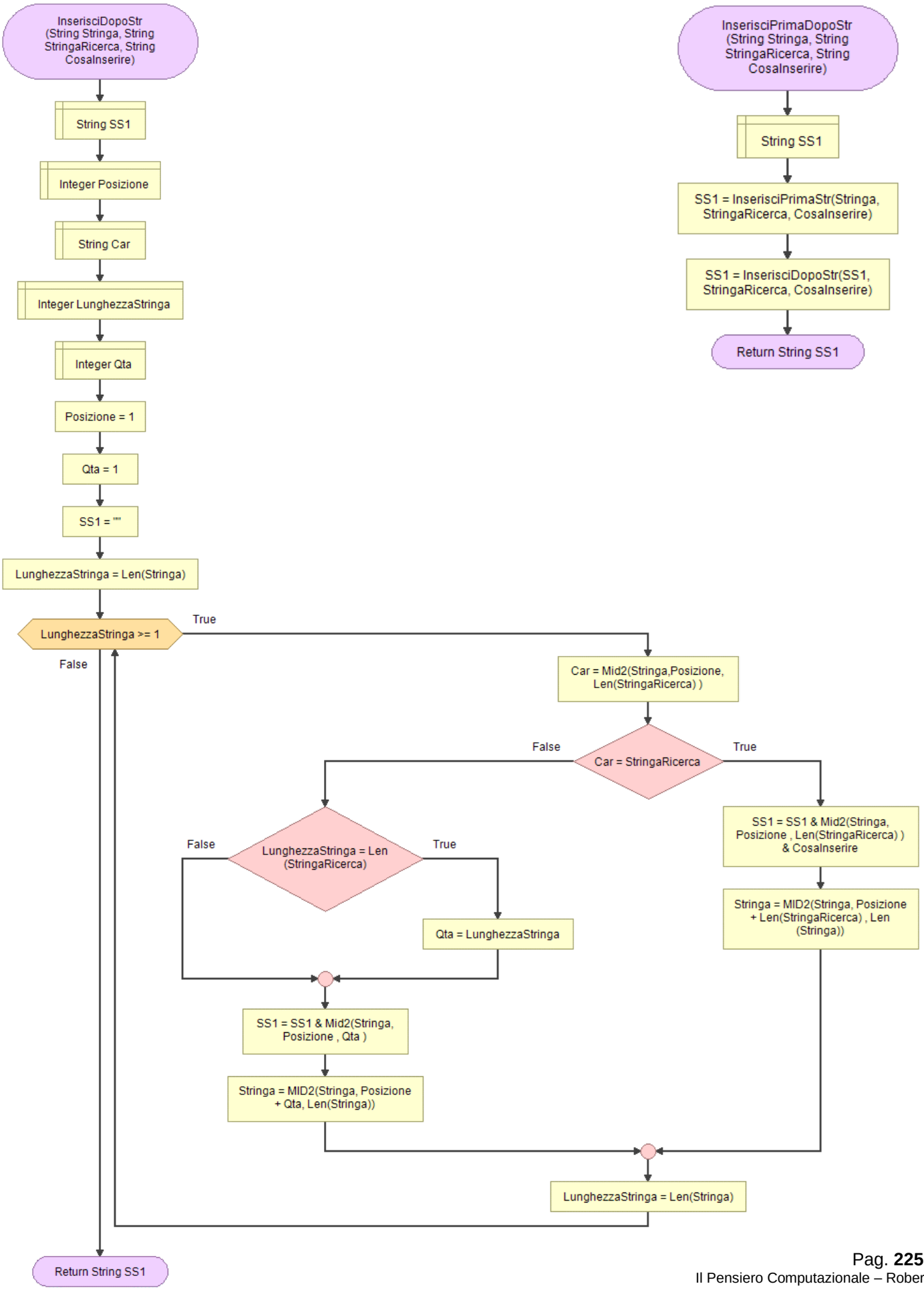


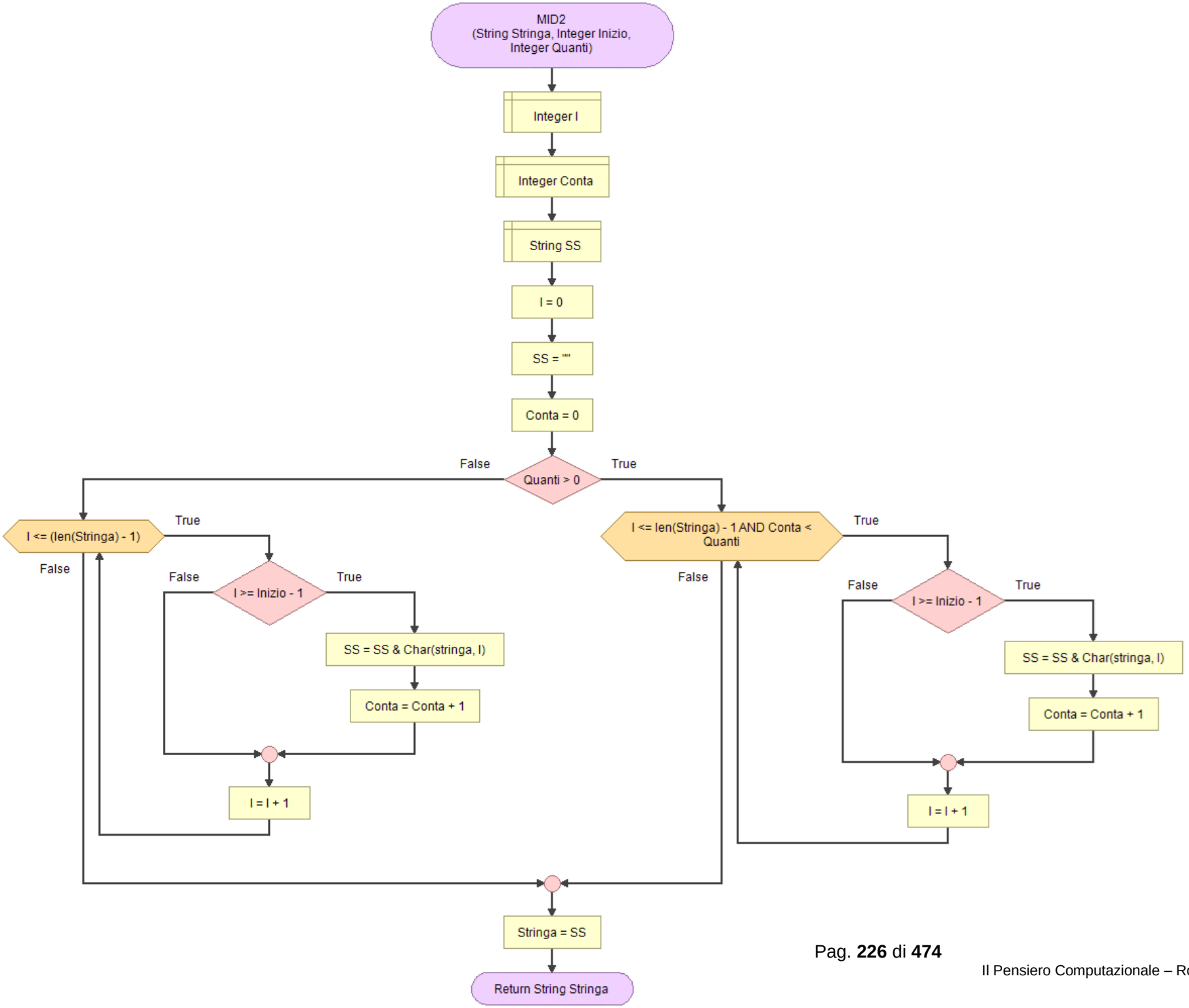

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

173.**G S F** ‖ Data una stringa di caratteri inserire uno o più caratteri prima e dopo una data sottostringa

Inserts more characters (substring) after and before a searched substring

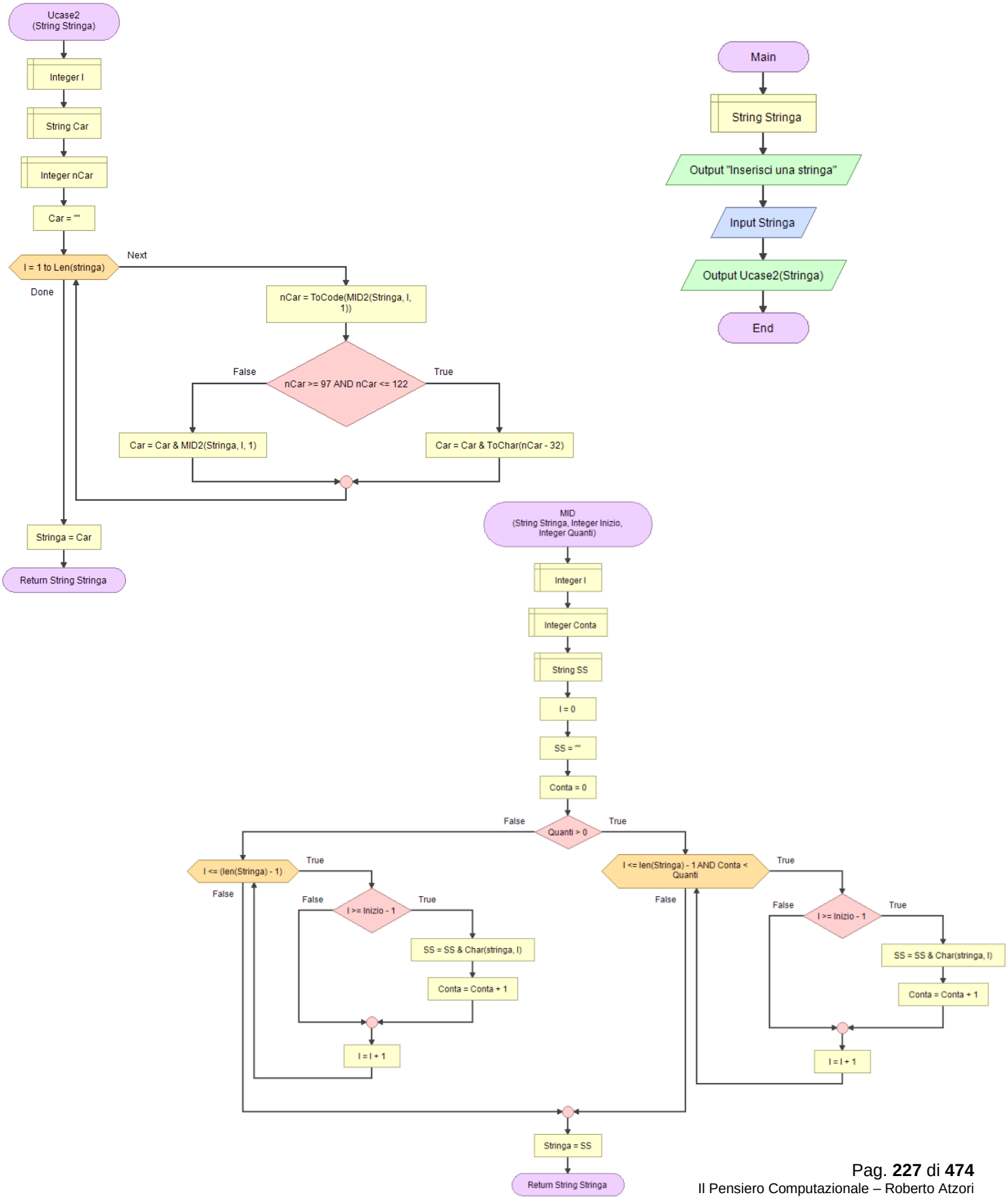



InserisciDopoStr (String Stringa, String StringaRicerca, String CosaInserire)
String SS1
Integer Posizione
String Car
Integer LunghezzaStringa
Integer Qta
Posizione = 1
Qta = 1
SS1 = ""
LunghezzaStringa = Len(Stringa)
LunghezzaStringa >= 1
True
False
Car = Mid2(Stringa,Posizione, Len(StringaRicerca))
Car = StringaRicerca
False
True
LunghezzaStringa = Len (StringaRicerca)
False
True
Qta = LunghezzaStringa
SS1 = SS1 & Mid2(Stringa, Posizione , Qta)
Stringa = MID2(Stringa, Posizione + Qta, Len(Stringa))
SS1 = SS1 & Mid2(Stringa, Posizione , Len(StringaRicerca)) & CosaInserire
Stringa = MID2(Stringa, Posizione + Len(StringaRicerca) , Len (Stringa))
LunghezzaStringa = Len(Stringa)
Return String SS1
InserisciPrimaDopoStr (String Stringa, String StringaRicerca, String CosaInserire)
String SS1
SS1 = InserisciPrimaStr(Stringa, StringaRicerca, CosaInserire)
SS1 = InserisciDopoStr(SS1, StringaRicerca, CosaInserire)
Return String SS1


MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

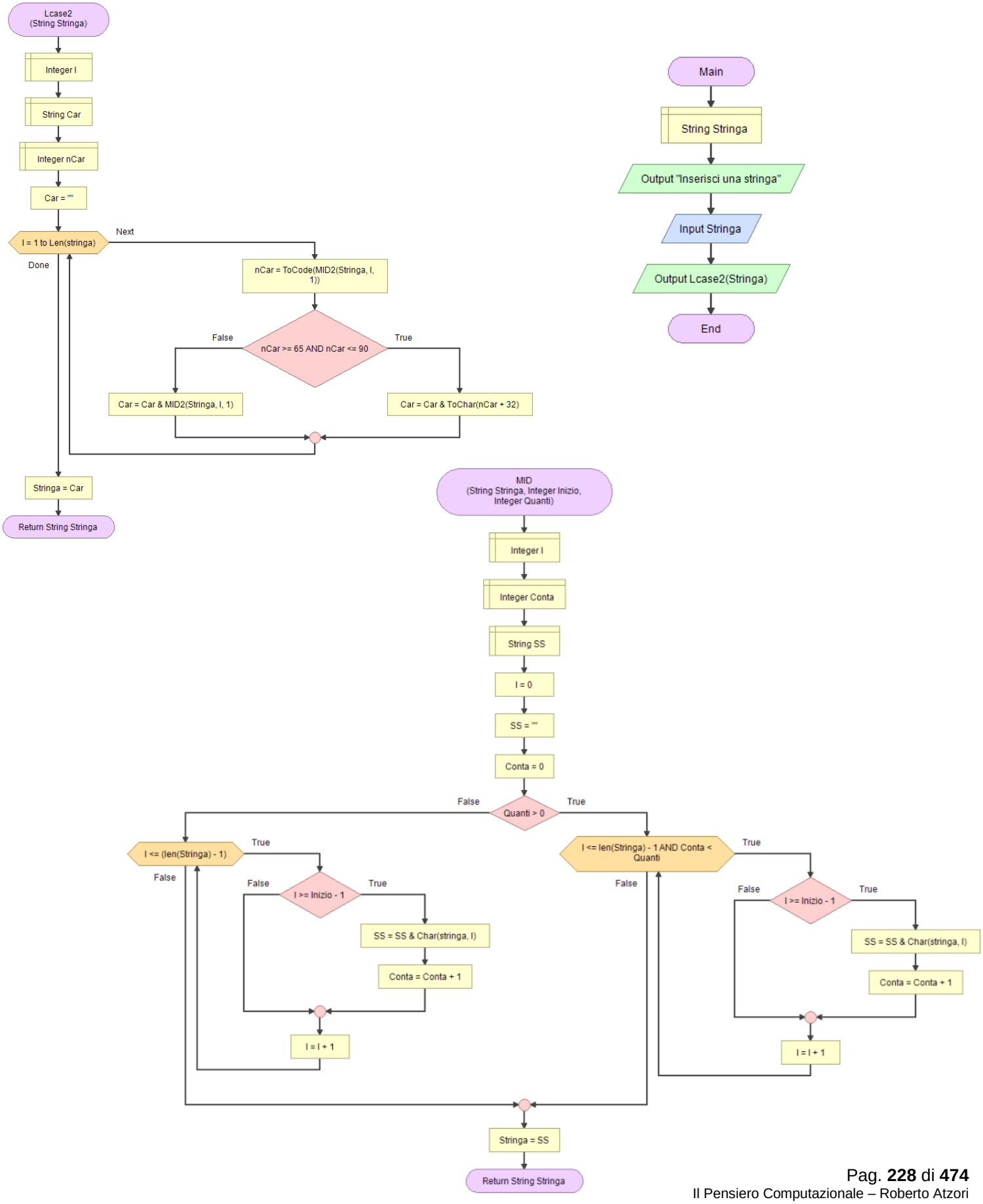
174.GSF | Funzione UCASE del Visual Basic - utilizzo con funzione

Function UCASE Visual Basic 6 like



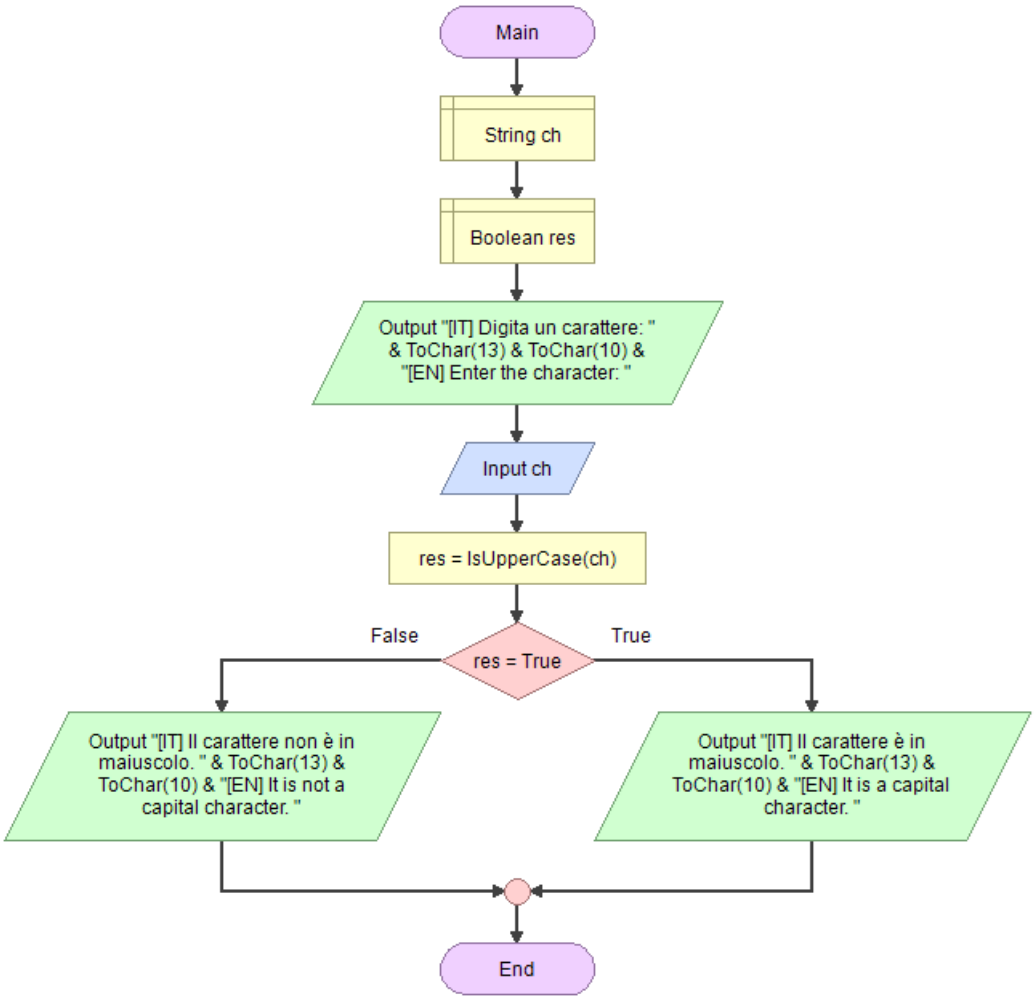
175.GSF ‖ Funzione LCASE del Visual Basic - utilizzo con funzione

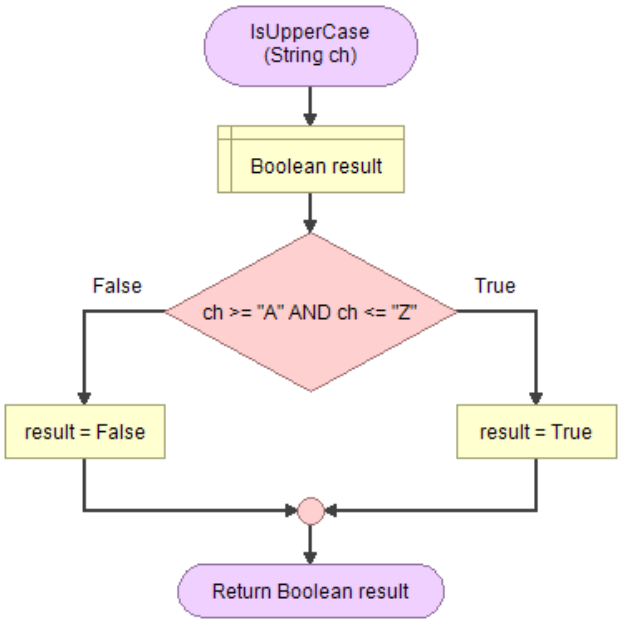
Function LCASE Visual Basic 6 like



176. **G S F** ‖ Controlla se il carattere inserito è in forma maiuscola

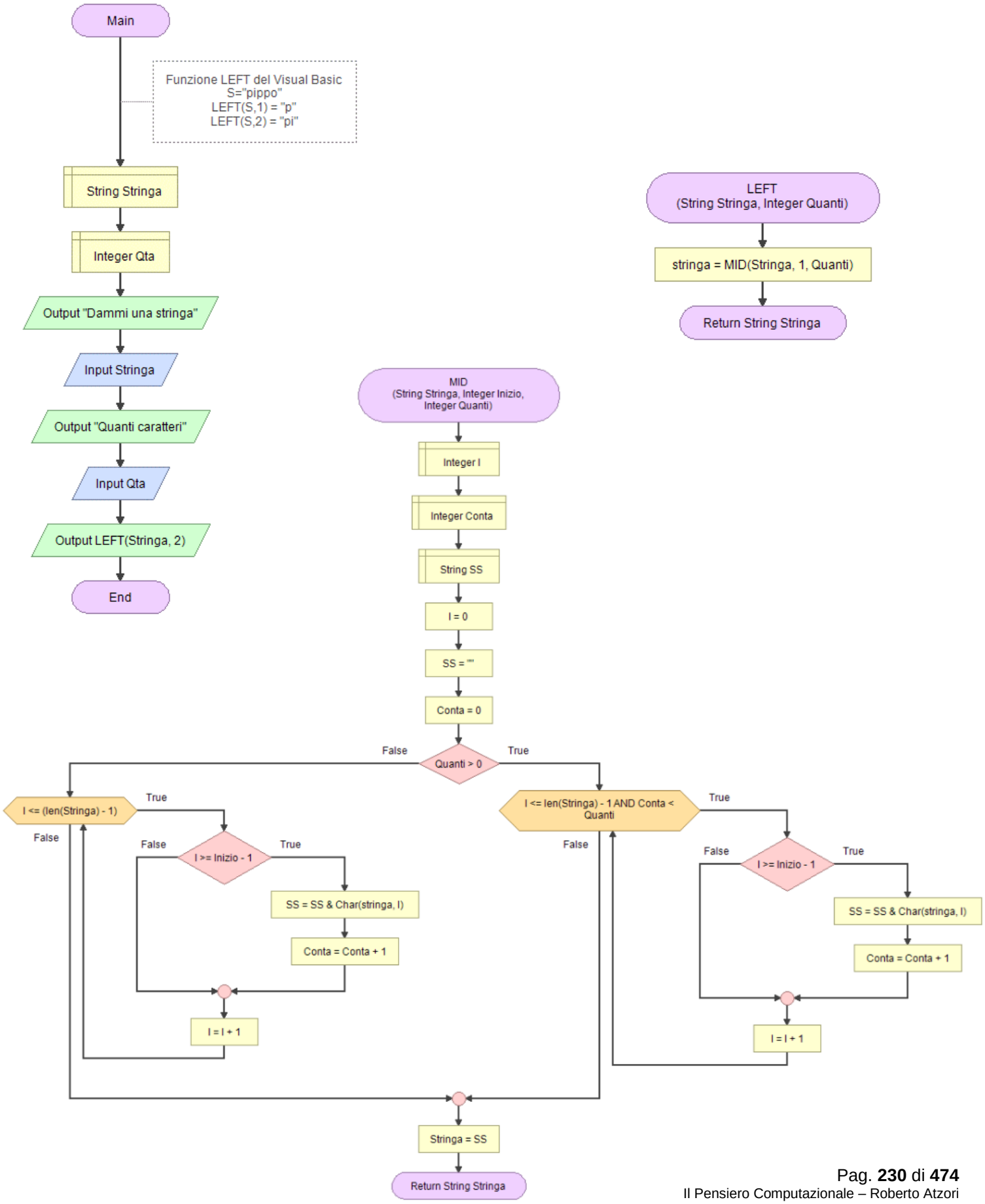
Check whether entered character is uppercase letter or not





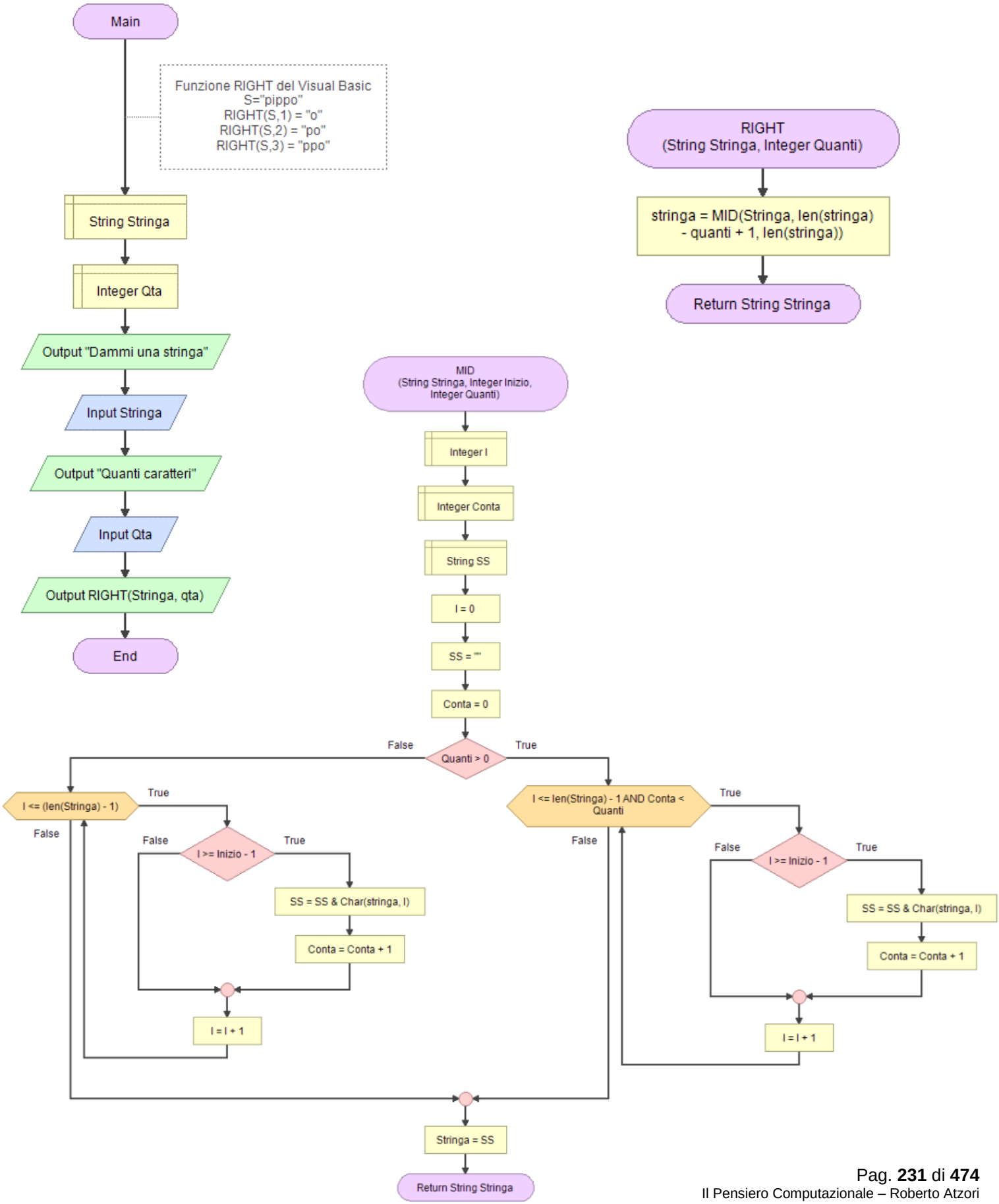
177.GSF ‖ Funzione LEFT del Visual Basic - utilizzo con funzione

Function LEFT Visual Basic 6 like



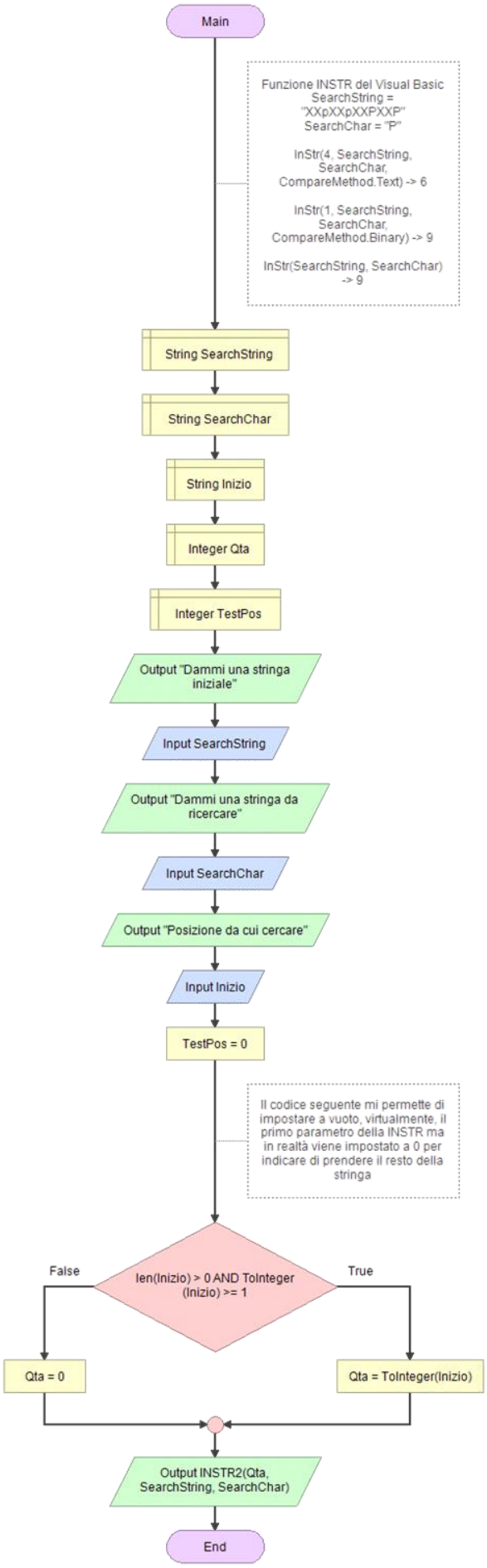
178.GSF ‖ Funzione RIGHT del Visual Basic - utilizzo con funzione

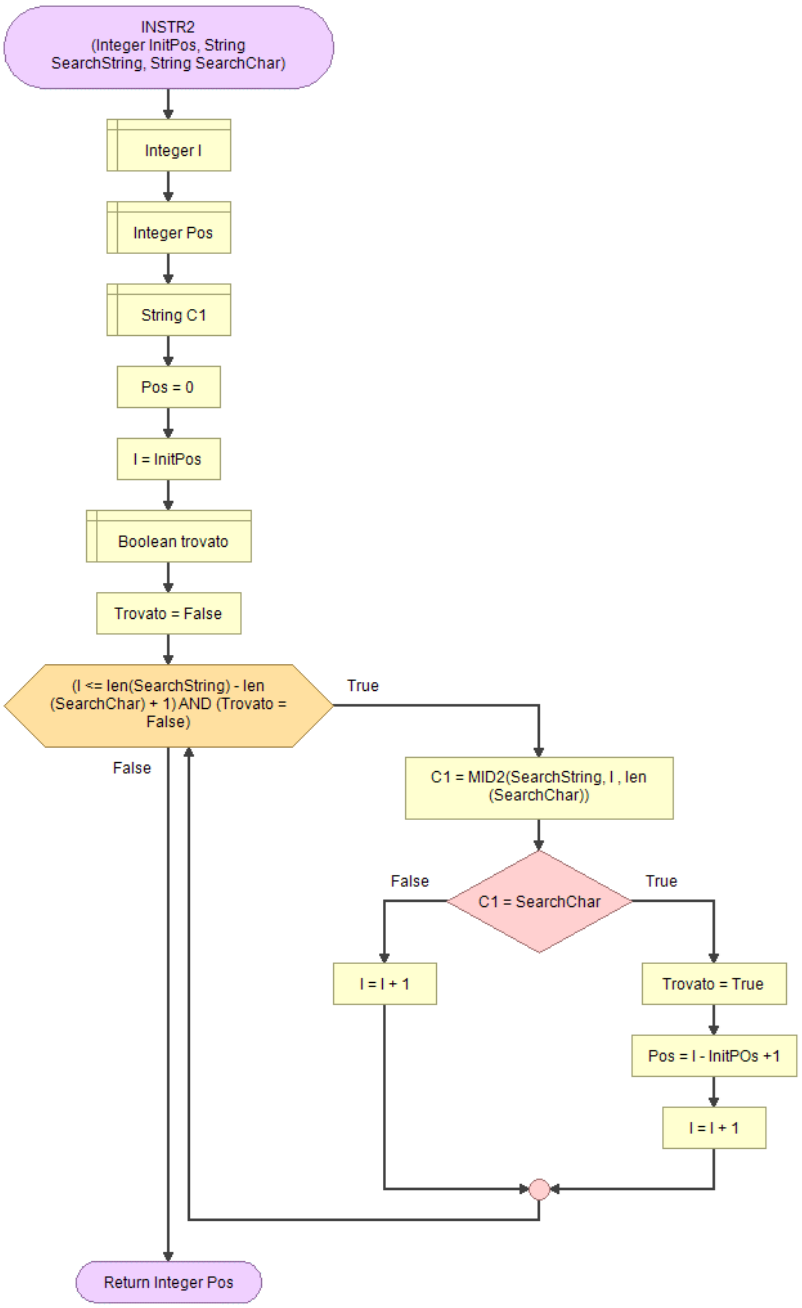
Function RIGHT Visual Basic 6 like

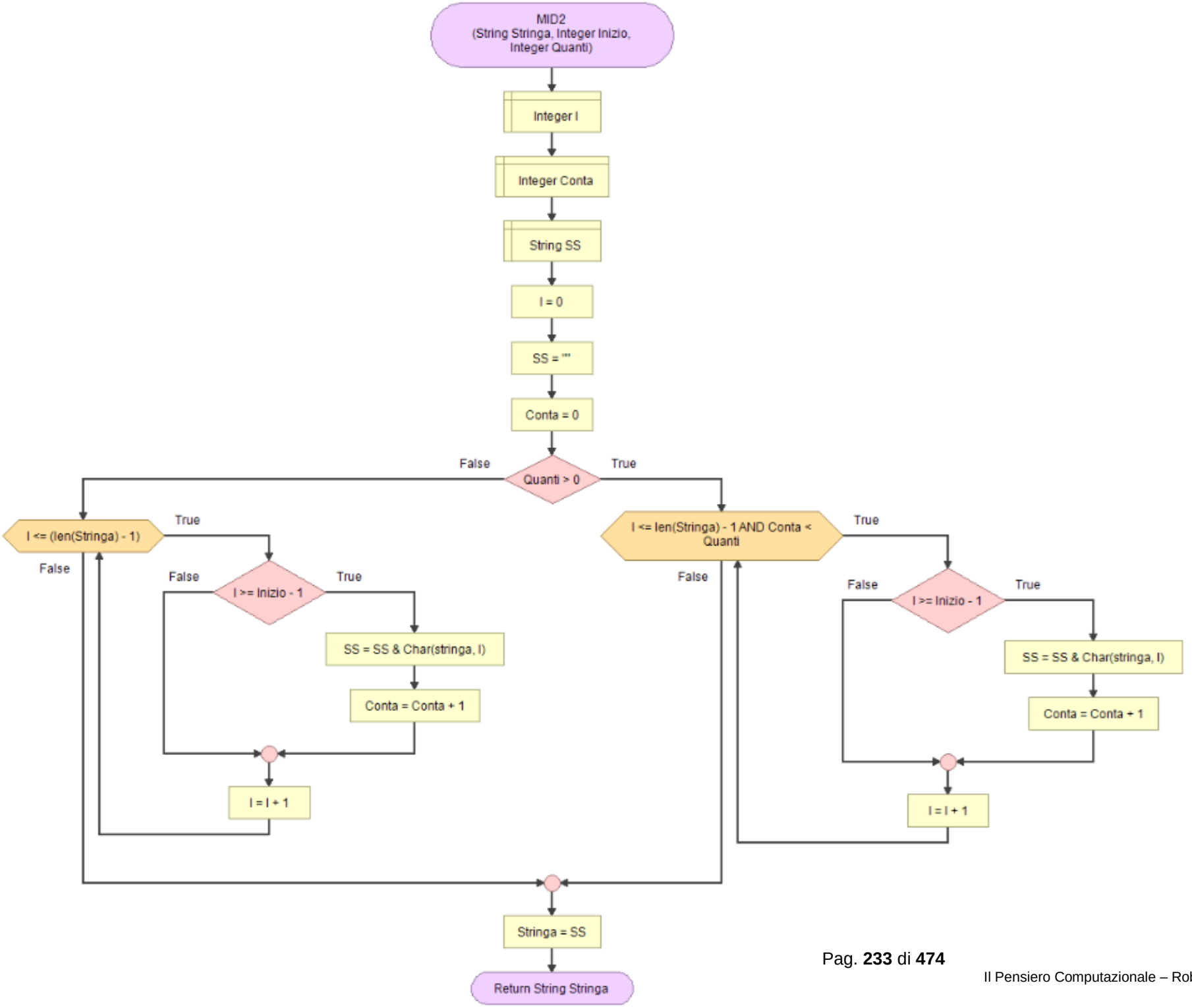


179.GSF ‖ Funzione INSTR del Visual Basic - utilizzo con funzione

Function INSTR Visual Basic 6 like

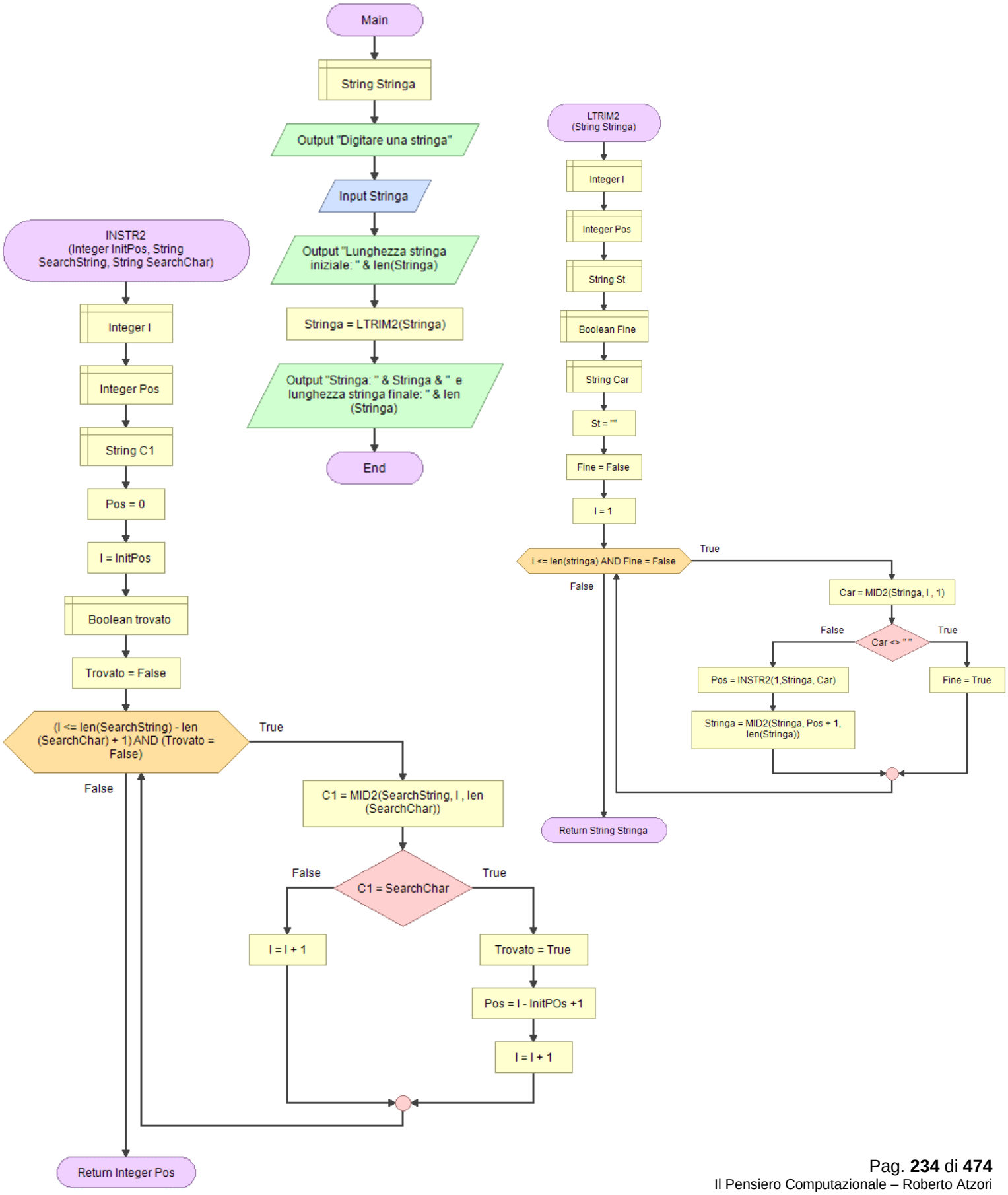


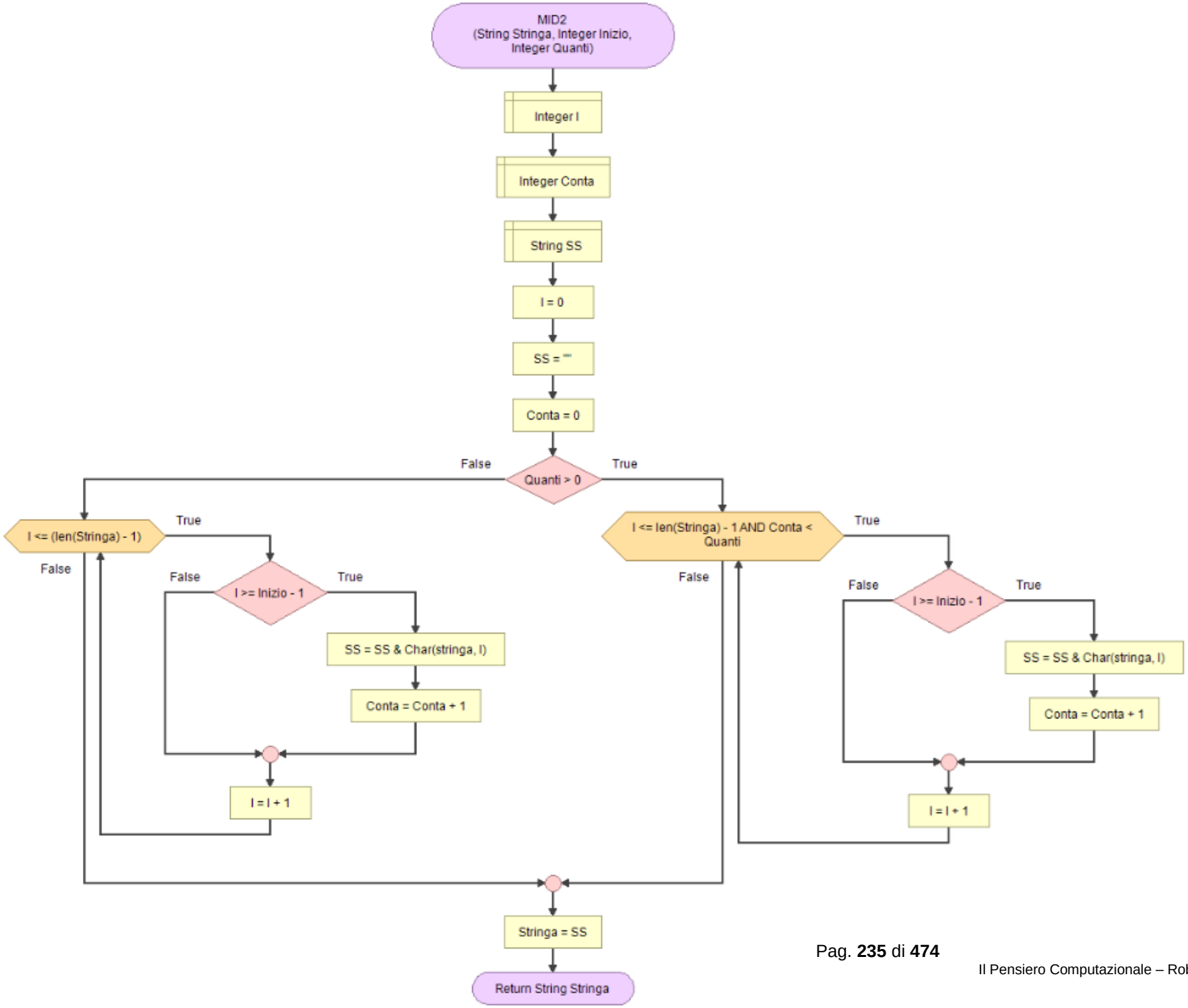



MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

180.GSF ‖ Funzione LTRIM del Visual Basic - utilizzo con funzione

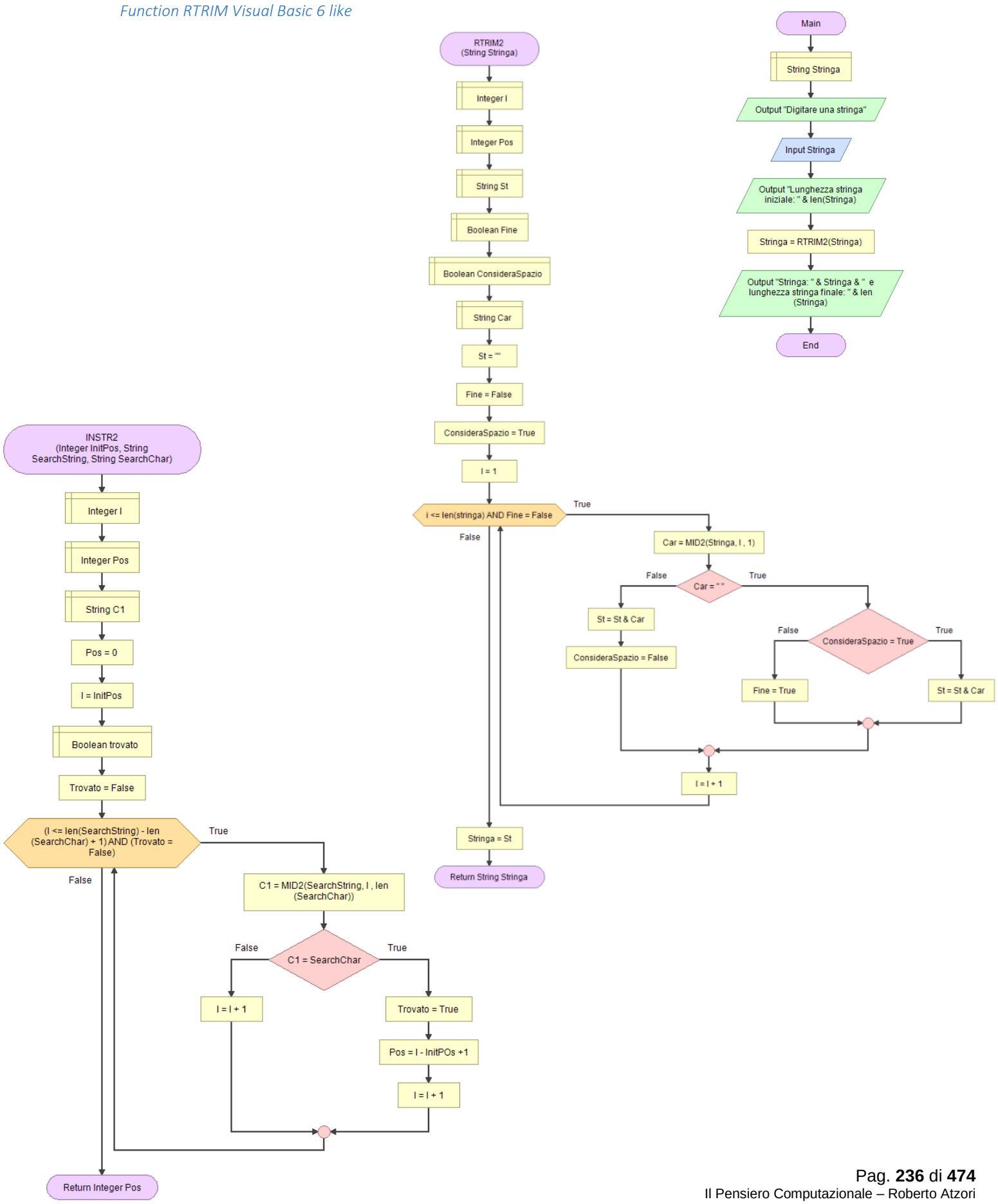
Function LTRIM Visual Basic 6 like

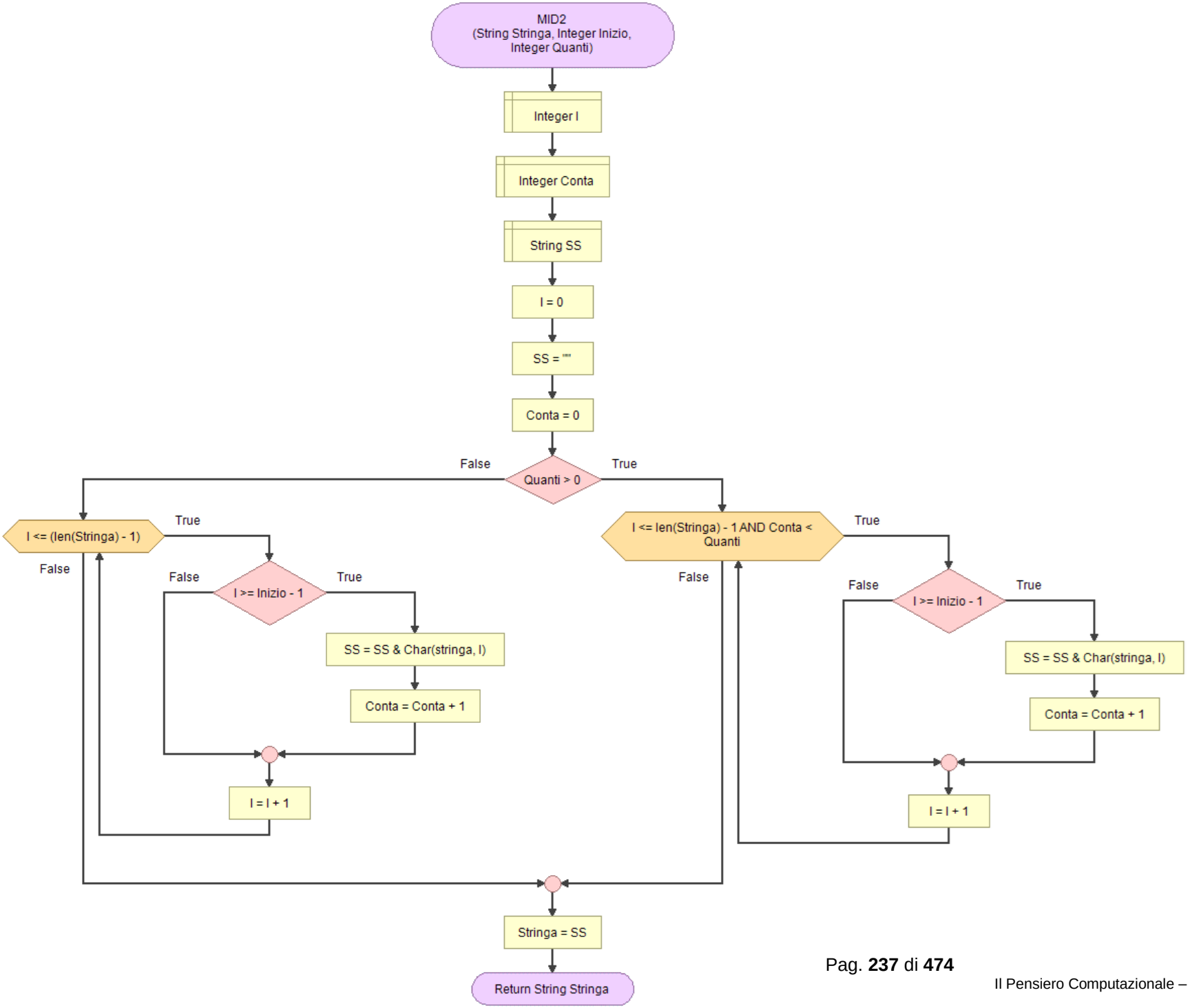



MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

181.GSF ‖ Funzione RTRIM del Visual Basic - utilizzo con funzione

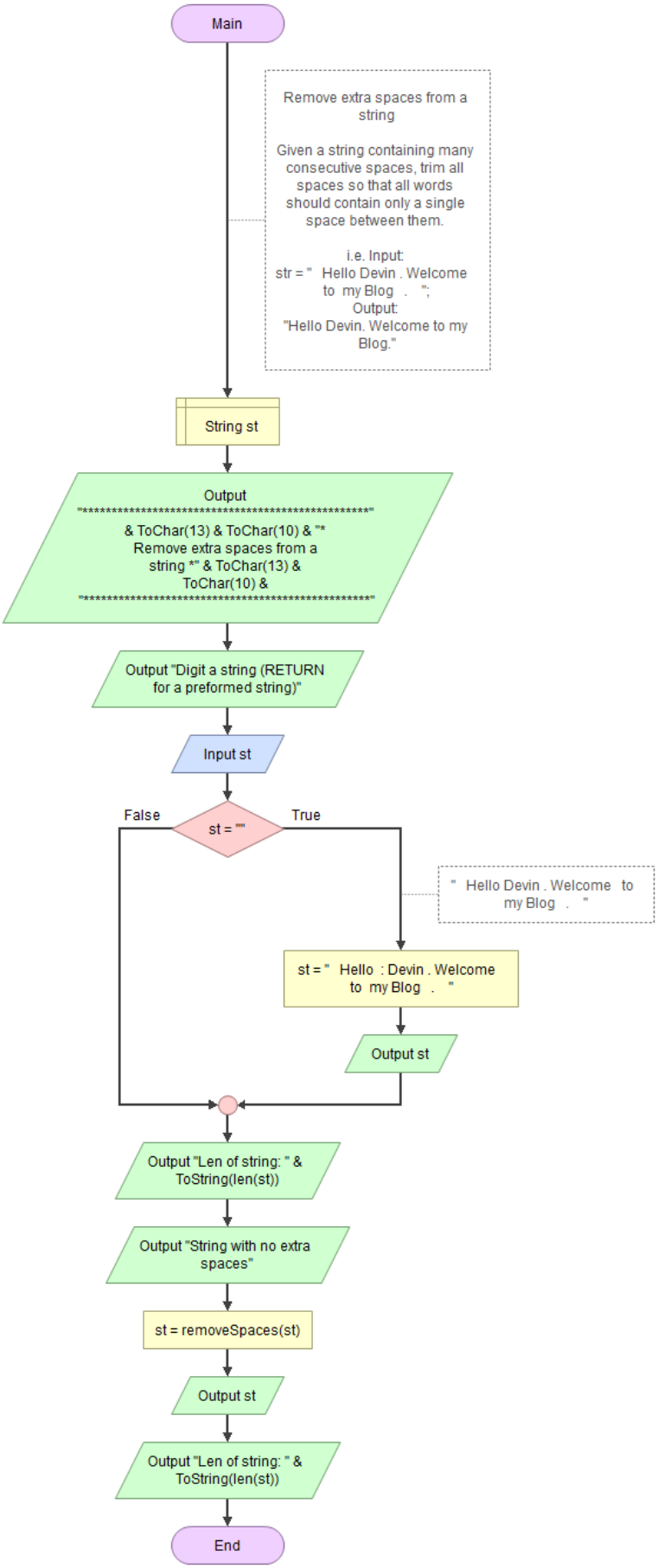
Function RTRIM Visual Basic 6 like

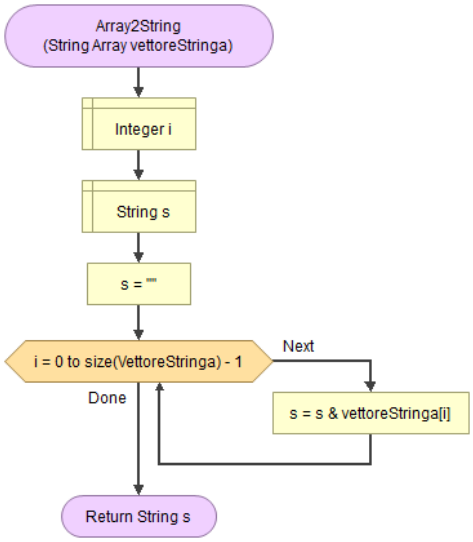


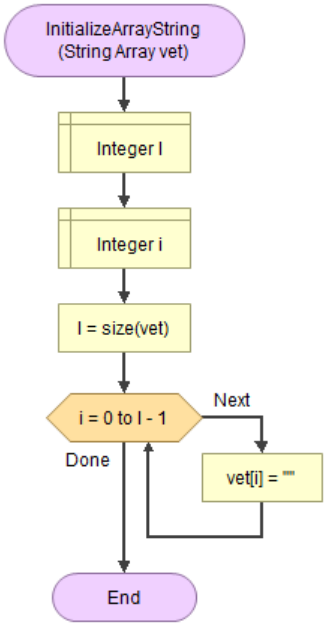

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

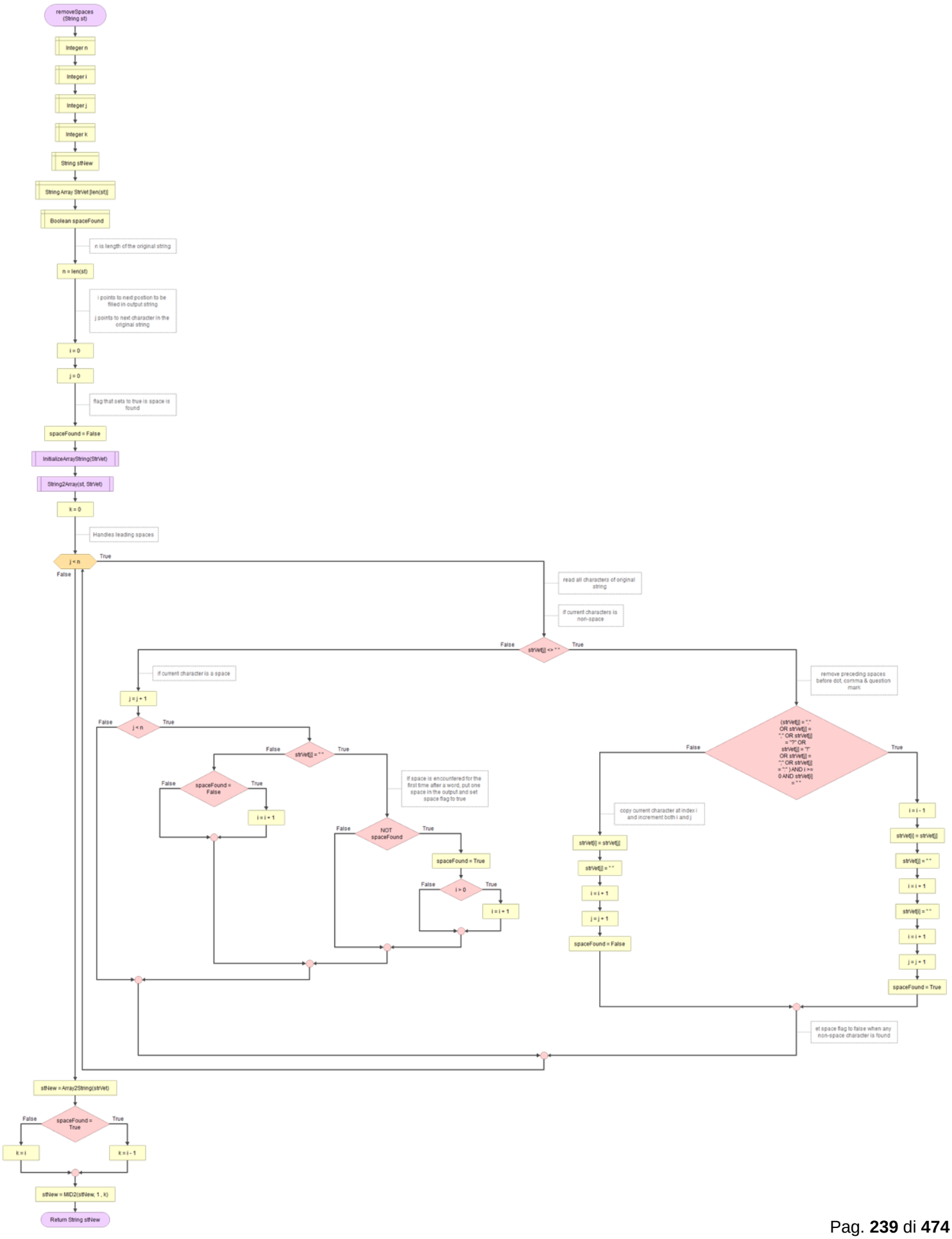
182.GSF ‖ Rimuovere tutti gli spazi non necessari in una stringa

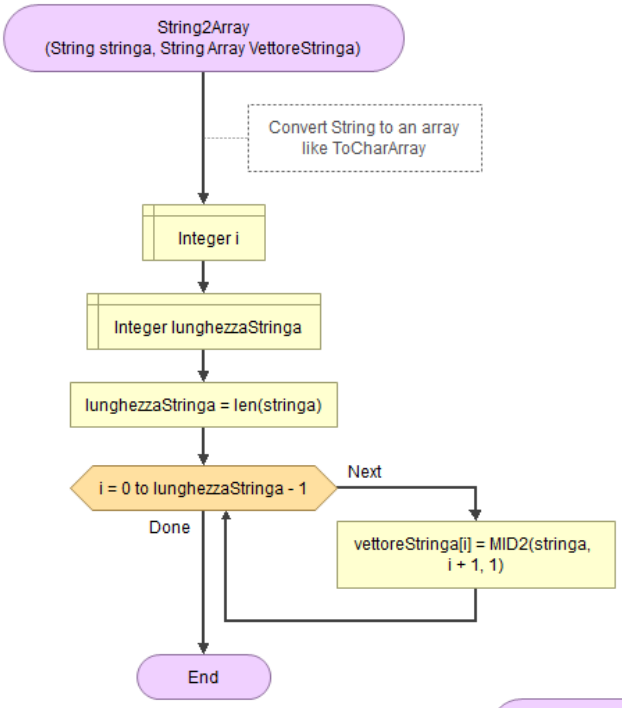
Remove extra spaces from a string

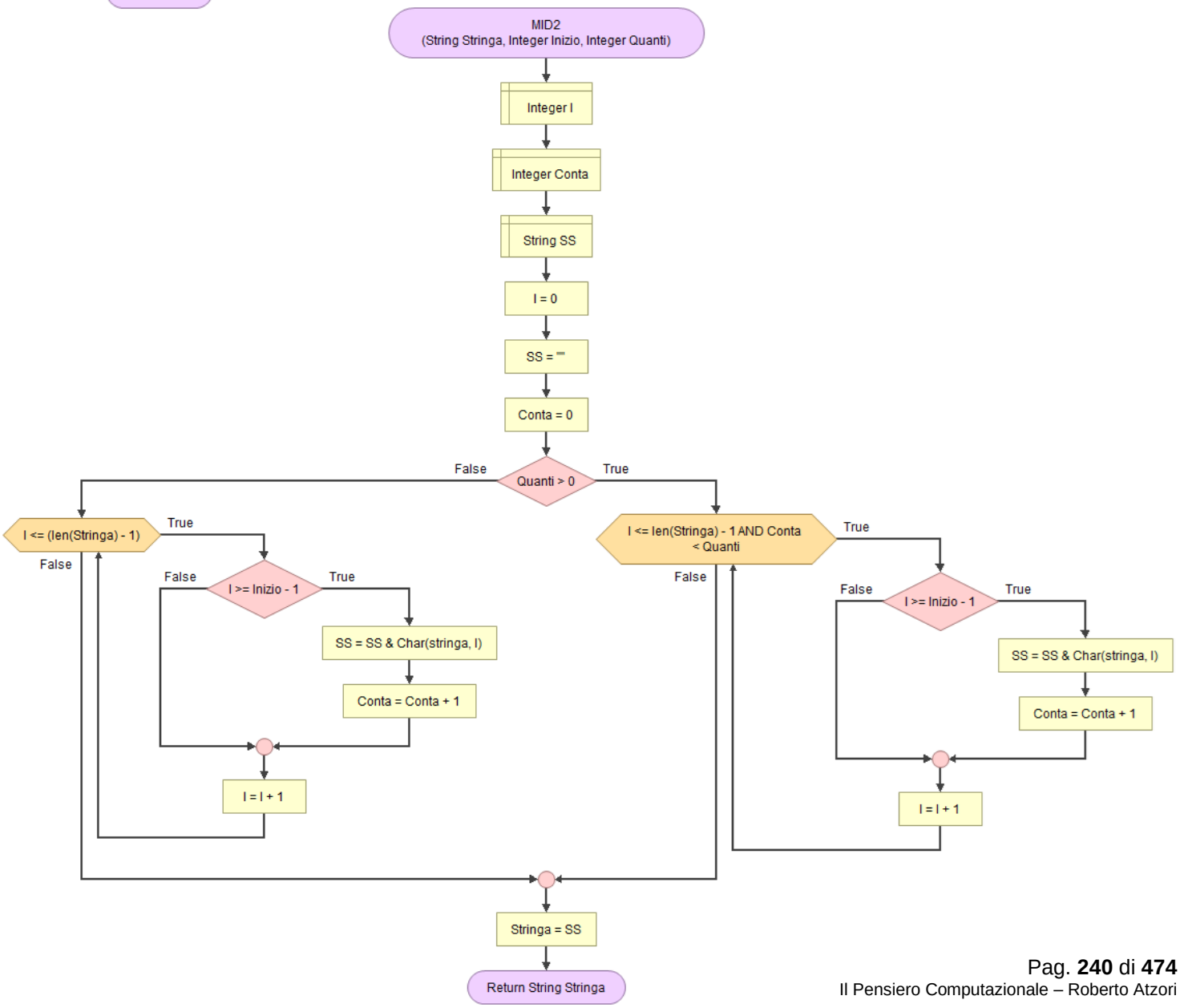






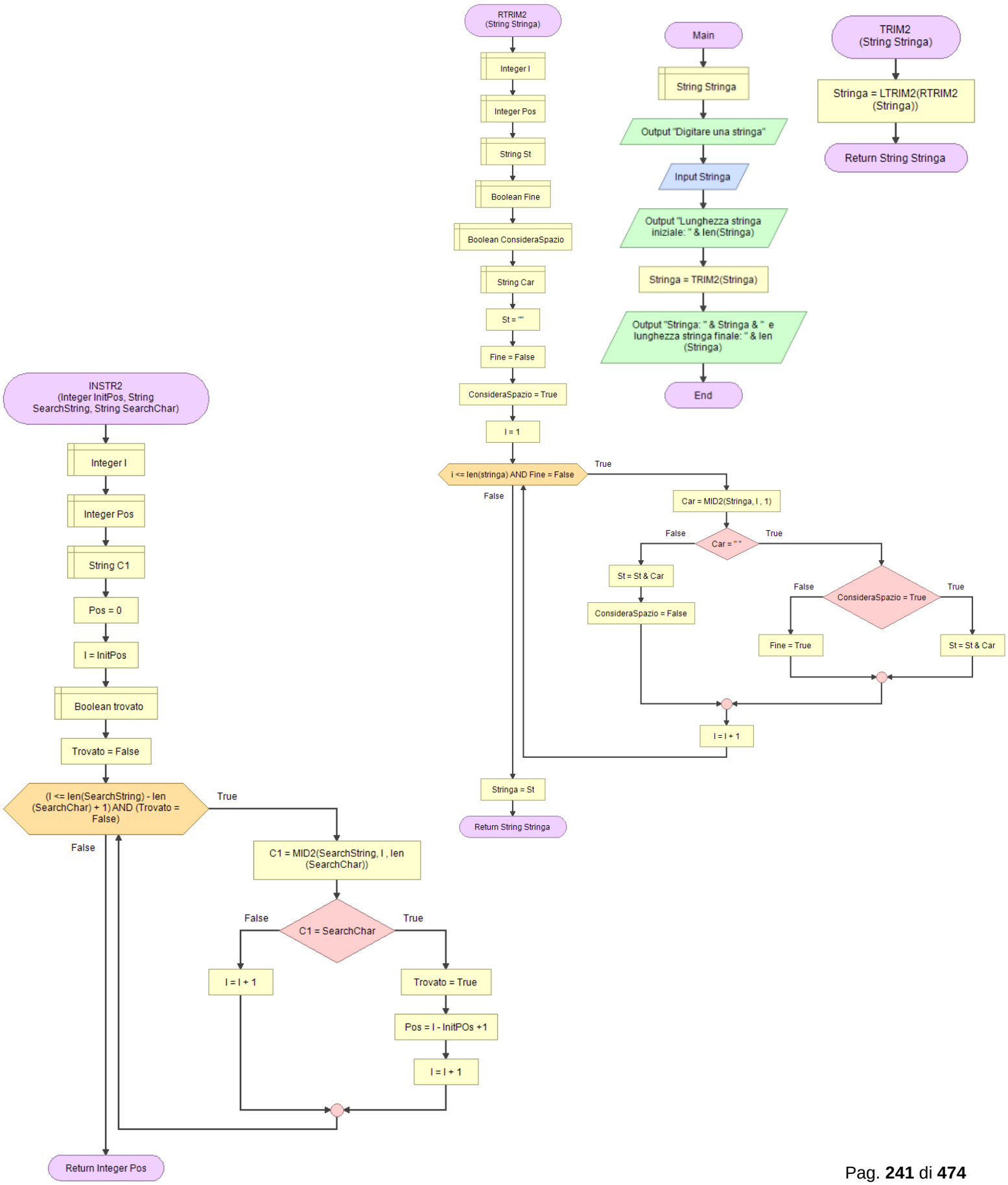

removeSpaces (String st)
Integer n
Integer i
Integer j
Integer k
String stNew
String Array StrVet [len(st)]
Boolean spaceFound
n is length of the original string
n = len(st)
i points to next position to be filled in output string
j points to next character in the original string
i = 0
j = 0
flag that sets to true is space is found
spaceFound = False
InitializeArrayString(StrVet)
String2Array(st, StrVet)
k = 0
Handles leading spaces
j < n
True
False
read all characters of original string
if current characters is non-space
strVet[j] <> " "
False
True
if current character is a space
j = j + 1
j < n
False
True
strVet[j] = " "
False
True
spaceFound = False
False
True
i = i + 1
if space is encountered for the first time after a word, put one space in the output and set space flag to true
NOT spaceFound
False
True
spaceFound = True
i > 0
False
True
i = i + 1
remove preceding spaces before dot, comma & question mark
(strVet[j] = " " OR strVet[j] = "." OR strVet[j] = "?" OR strVet[j] = "!" OR strVet[j] = "," OR strVet[j] = ";") AND i >= 0 AND strVet[i] = " "
False
True
copy current character at index i and increment both i and j
strVet[i] = strVet[j]
strVet[j] = " "
i = i + 1
j = j + 1
spaceFound = False
i = i - 1
strVet[i] = strVet[j]
strVet[j] = " "
i = i + 1
strVet[i] = " "
i = i + 1
j = j + 1
spaceFound = True
set space flag to false when any non-space character is found
stNew = Array2String(strVet)
spaceFound = True
False
True
k = i
k = i - 1
stNew = MID2(stNew, 1 , k)
Return String stNew

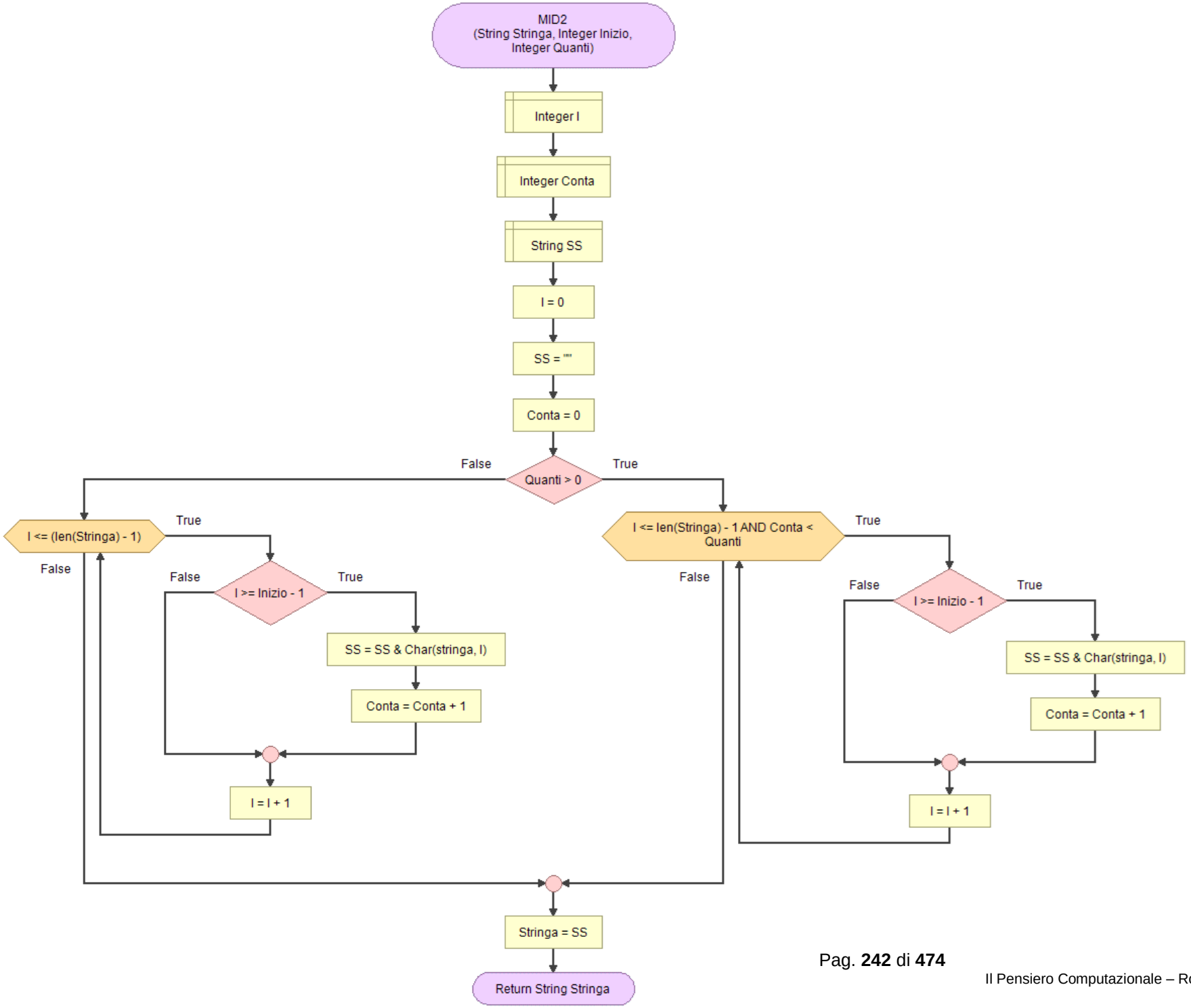

String2Array
(String stringa, String Array VettoreStringa)
Convert String to an array like ToCharArray
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len(stringa)
i = 0 to lunghezzaStringa - 1
Next
Done
vettoreStringa[i] = MID2(stringa, i + 1, 1)
End


MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

183.GSF ‖ Funzione TRIM del Visual Basic - utilizzo con funzione

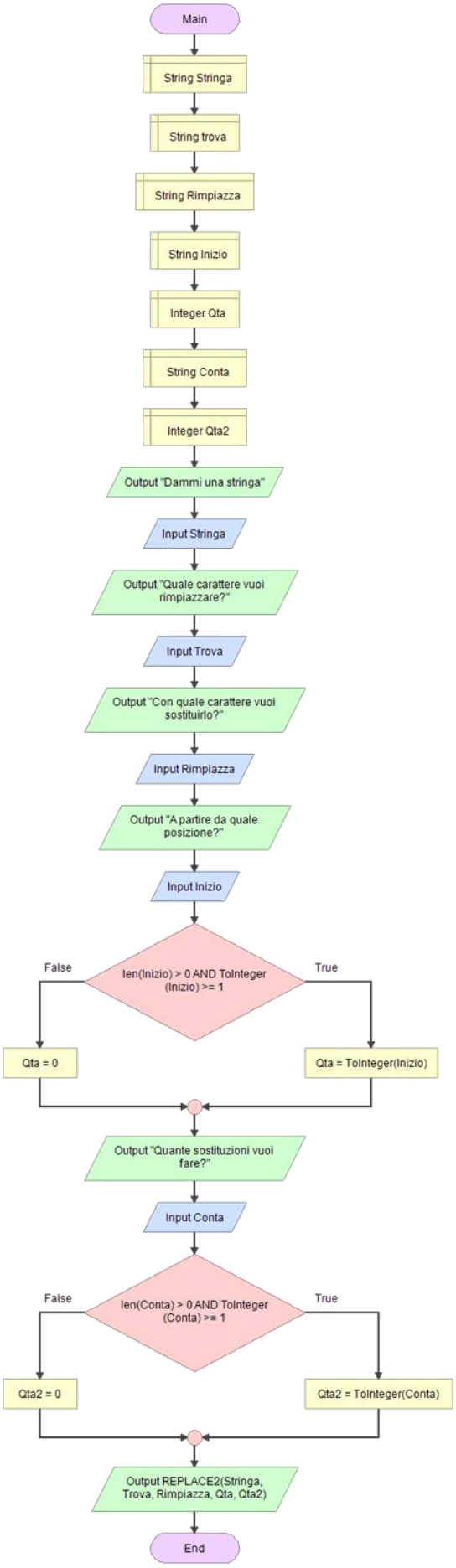
Function TRIM Visual Basic 6 like

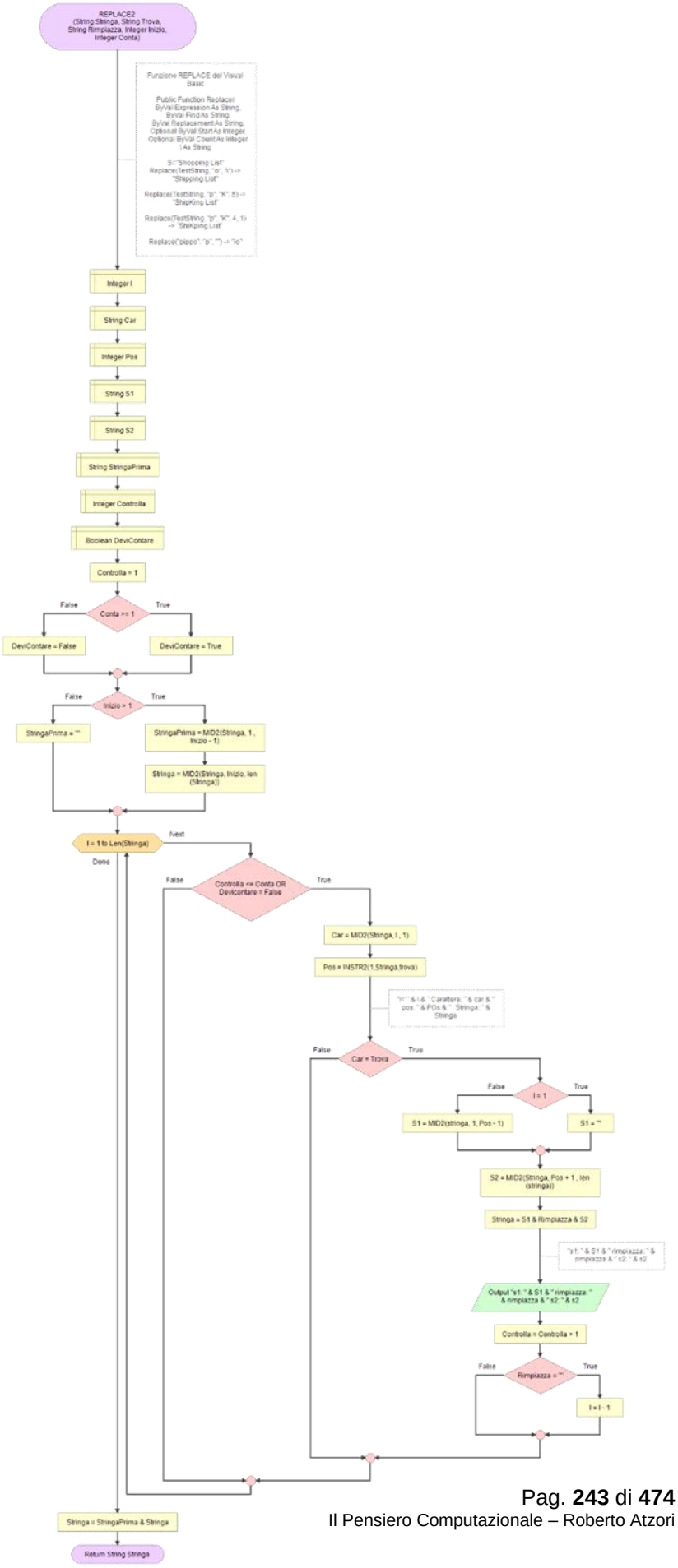


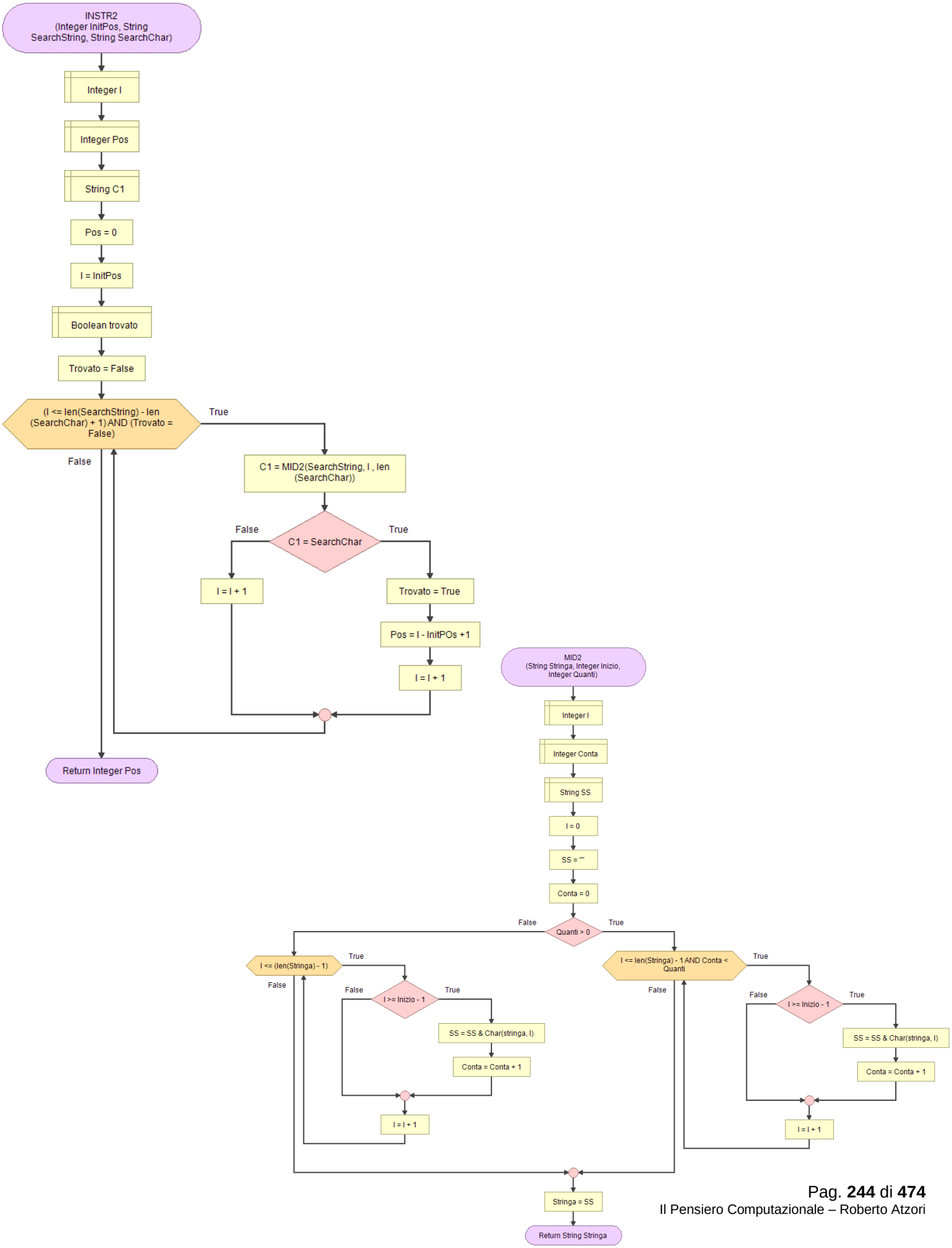

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

184.GSF | Funzione REPLACE del Visual Basic - utilizzo con funzione

Function REPLACE Visual Basic 6 like

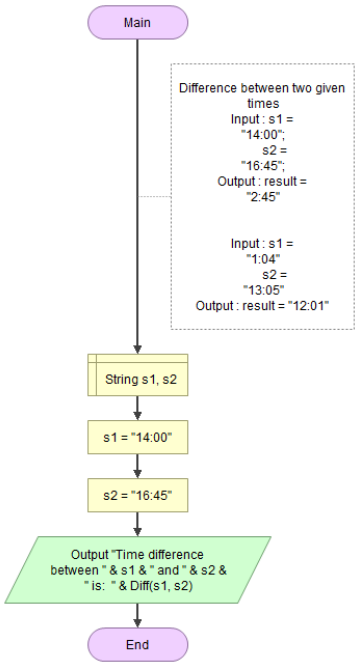


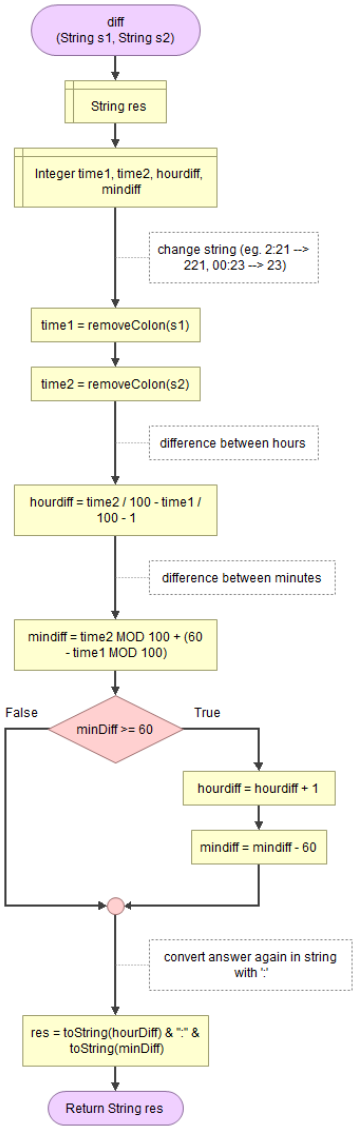


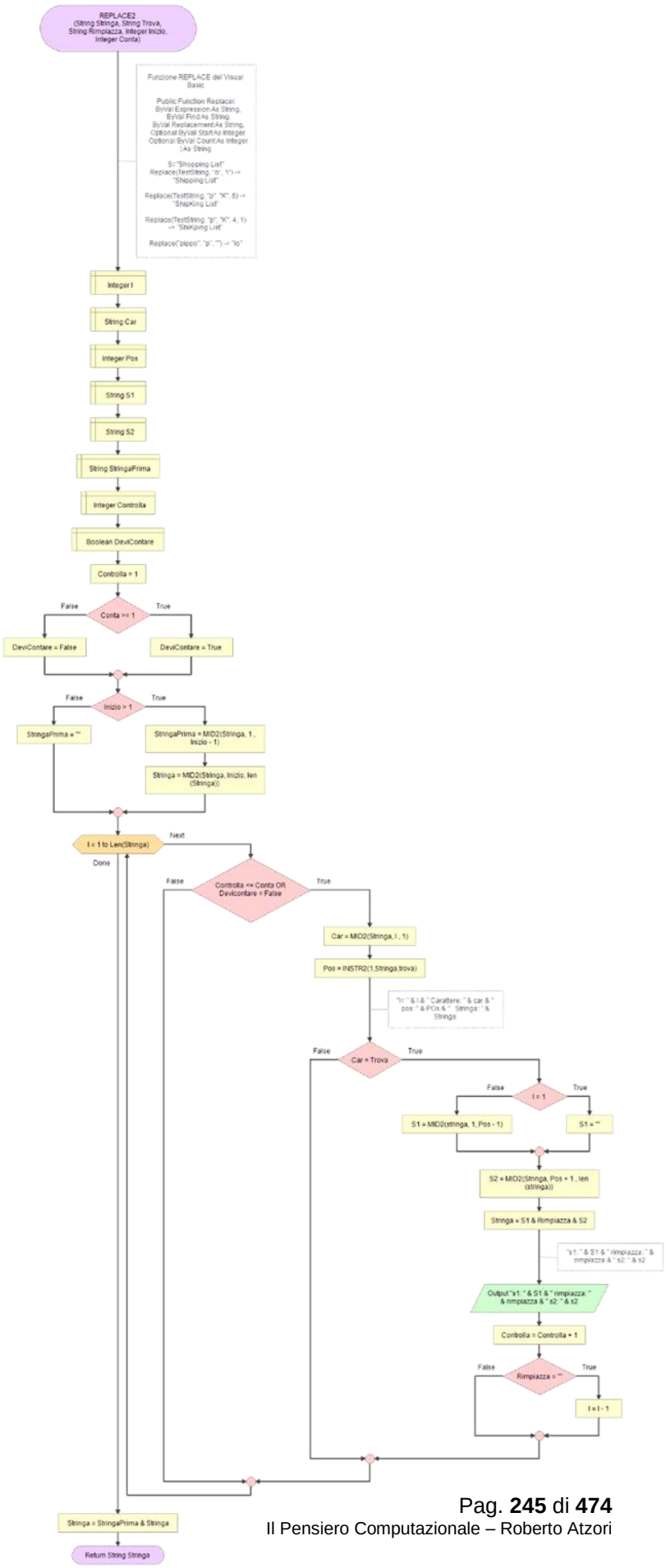

INSTR2
(Integer InitPos, String SearchString, String SearchChar)
Integer I
Integer Pos
String C1
Pos = 0
I = InitPos
Boolean trovato
Trovato = False
(I <= len(SearchString) - len (SearchChar) + 1) AND (Trovato = False)
True
False
C1 = MID2(SearchString, I , len (SearchChar))
C1 = SearchChar
False
True
I = I + 1
Trovato = True
Pos = I - InitPOs +1
I = I + 1
Return Integer Pos
MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

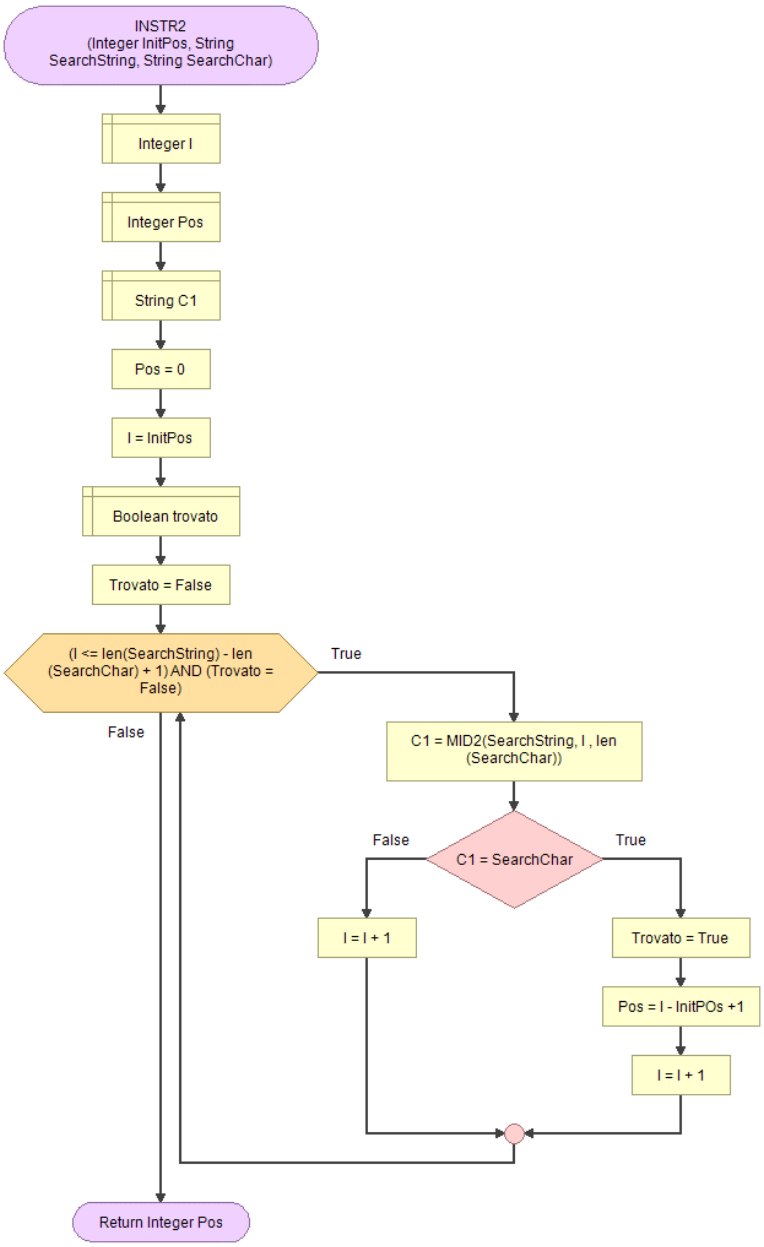
185.GSF | Differenza tra due orari

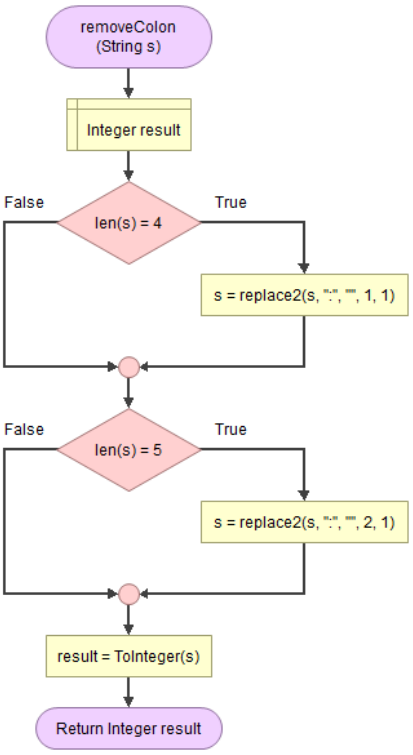
Difference between two given times

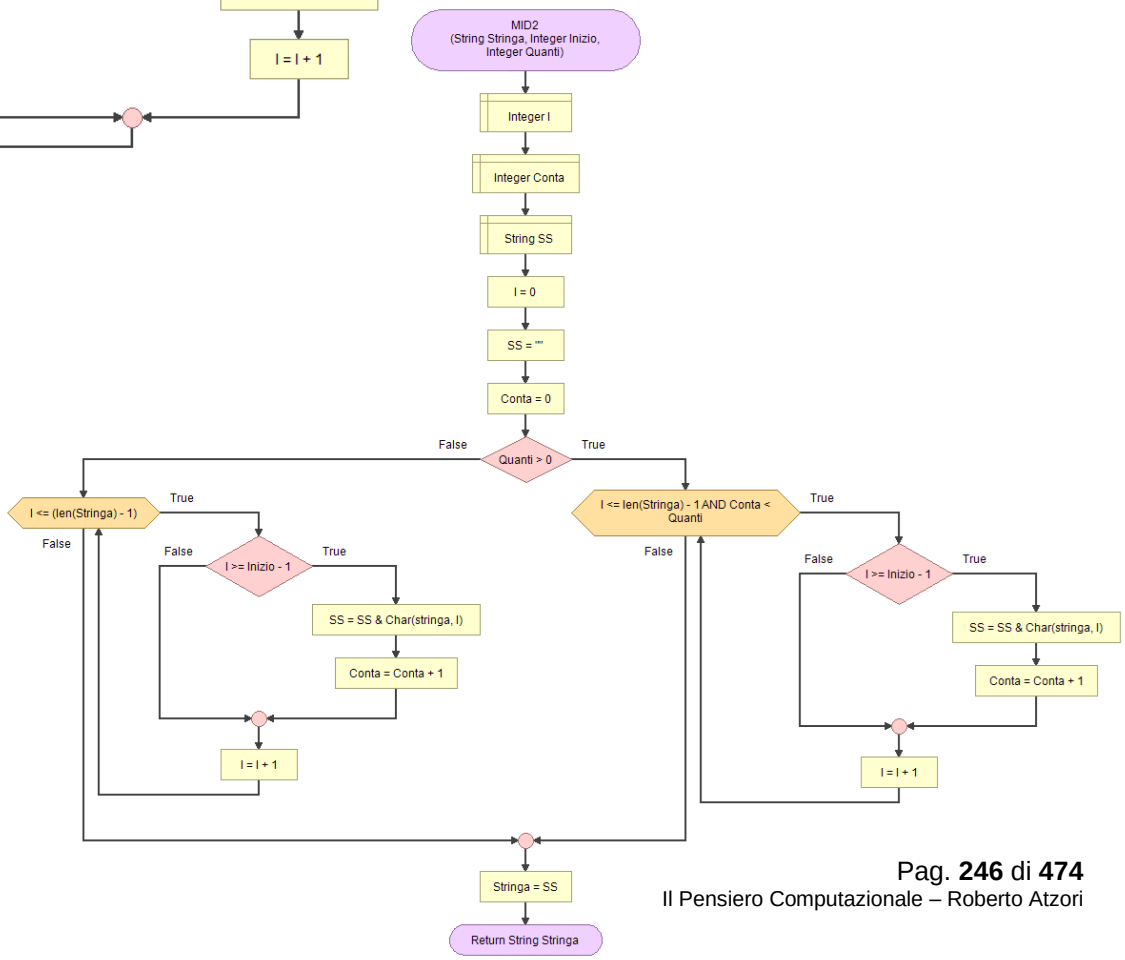






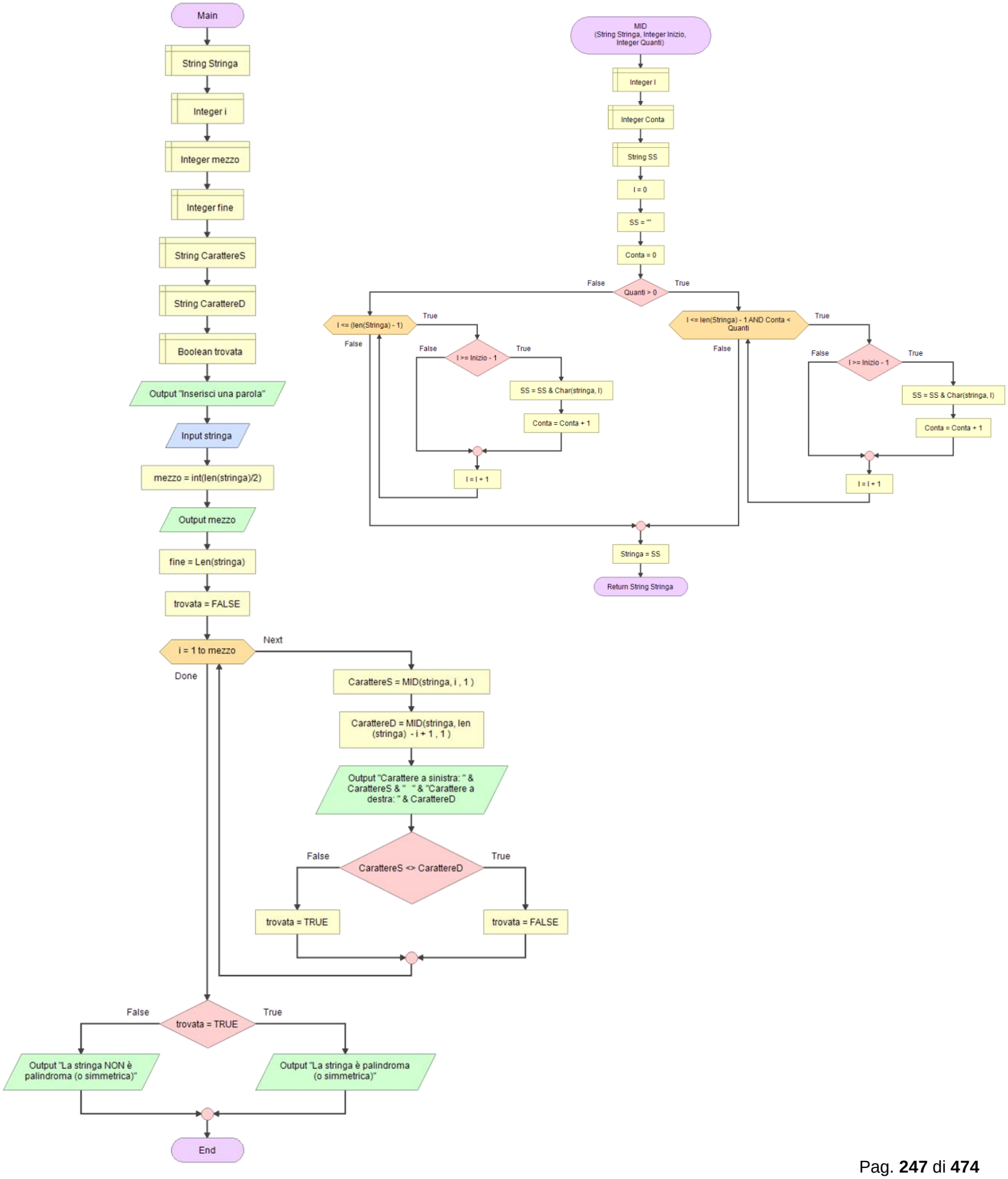

INSTR2
(Integer InitPos, String SearchString, String SearchChar)
Integer I
Integer Pos
String C1
Pos = 0
I = InitPos
Boolean trovato
Trovato = False
(I <= len(SearchString) - len (SearchChar) + 1) AND (Trovato = False)
True
False
C1 = MID2(SearchString, I , len (SearchChar))
C1 = SearchChar
False
True
I = I + 1
Trovato = True
Pos = I - InitPOs +1
I = I + 1
Return Integer Pos


removeColon
(String s)
Integer result
len(s) = 4
False
True
s = replace2(s, ":", "", 1, 1)
len(s) = 5
False
True
s = replace2(s, ":", "", 2, 1)
result = ToInteger(s)
Return Integer result


MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

186. G S F | Verificare che data una stringa in input essa sia palindroma o simmetrica

Check if a entered string is a palindrome or not

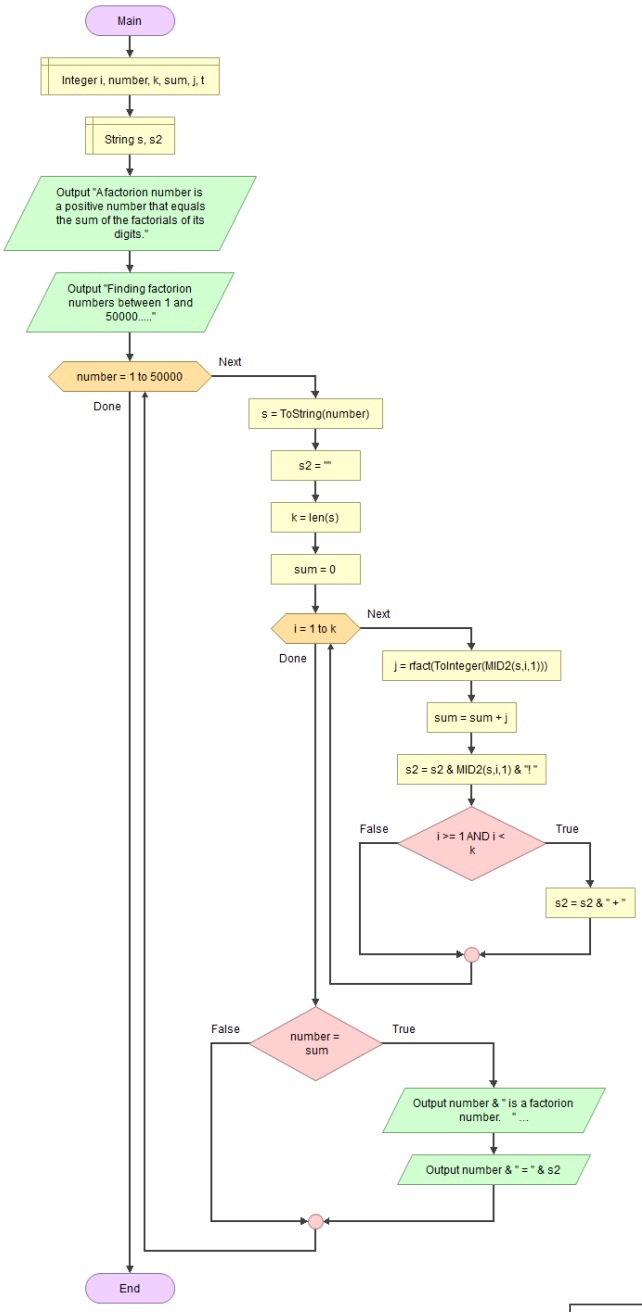


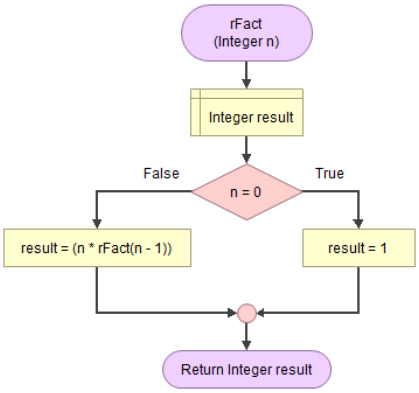
187.GSFR ‖ Trovare i numeri “factorion”

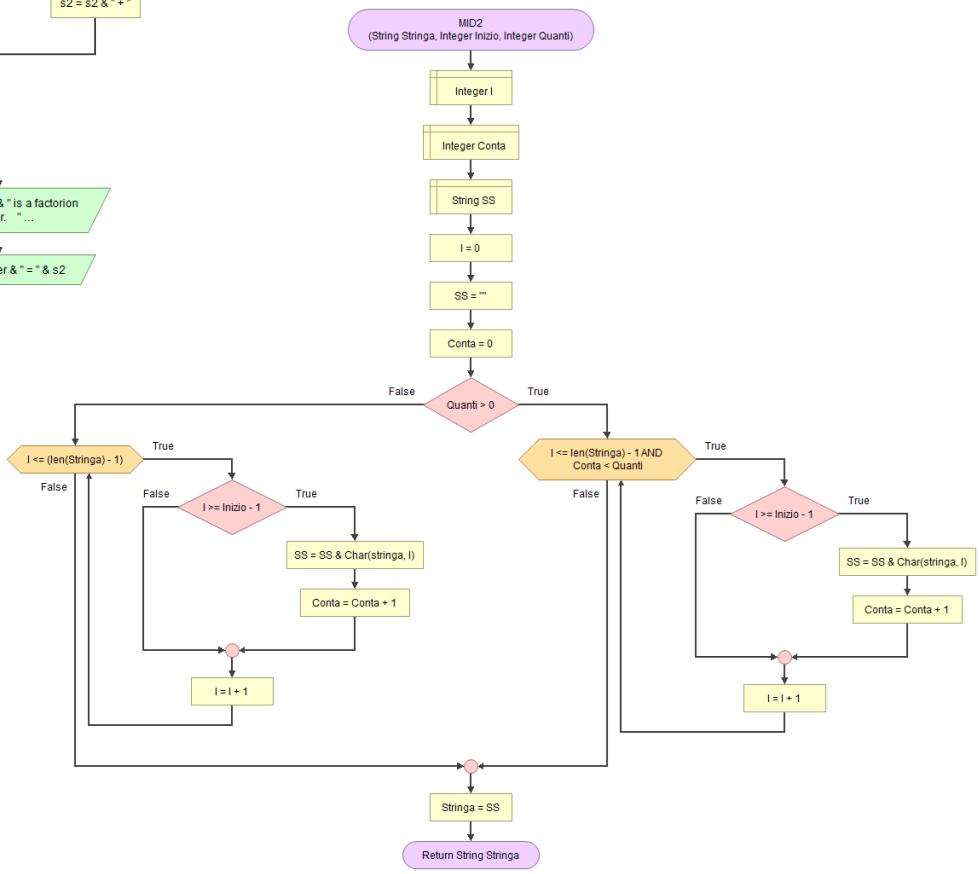
Find a factorion number

A factorion is a positive integer that equals the sum of the factorials of its digits.

1 = 1!
2 = 2!
145 = 1! + 4! + 5!
40585 = 4! + 0! + 5! + 8! + 5!

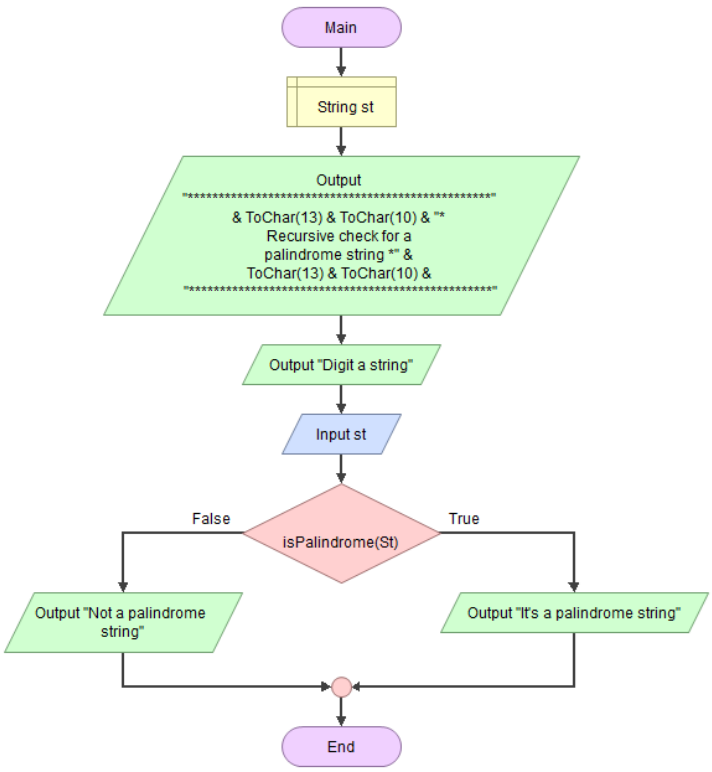


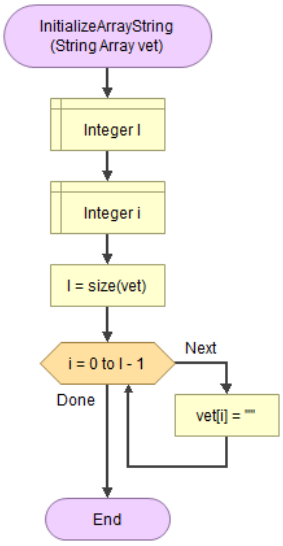


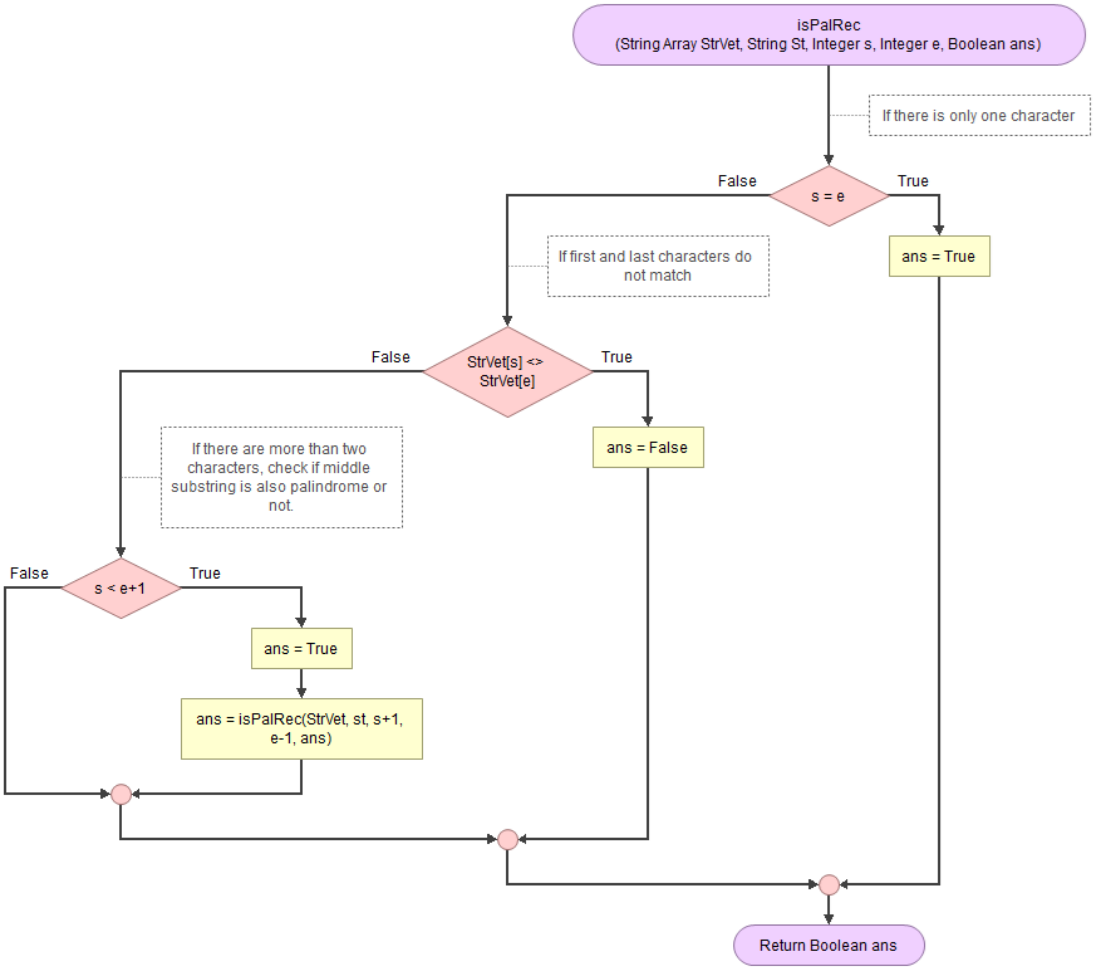


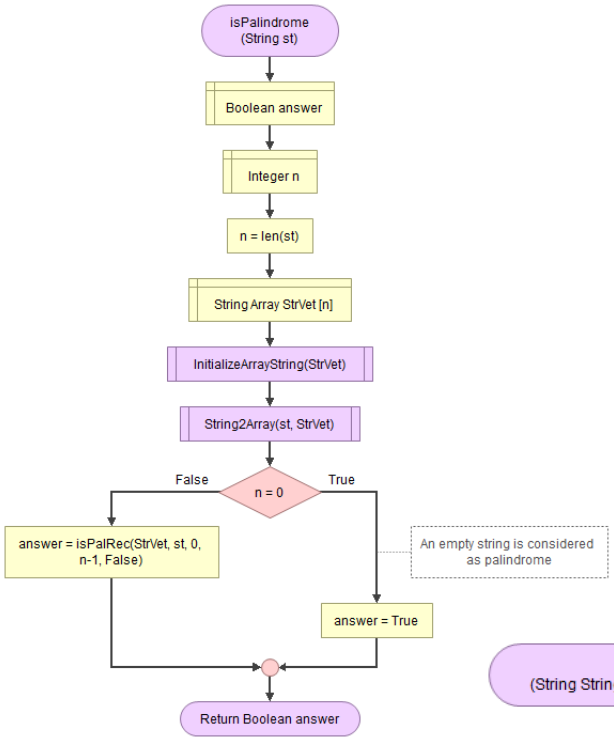
188. G S F R ‖ Verificare che data una stringa in input essa sia palindroma o simmetrica - ricorsivo

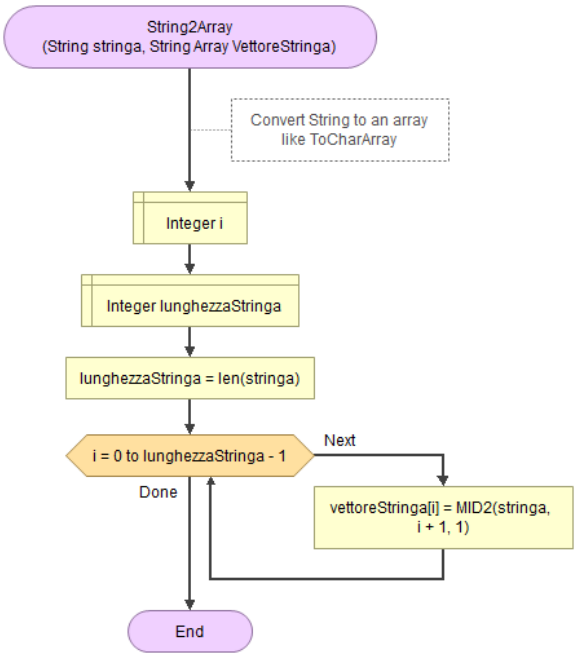
Check if a string is palindrome - recursive

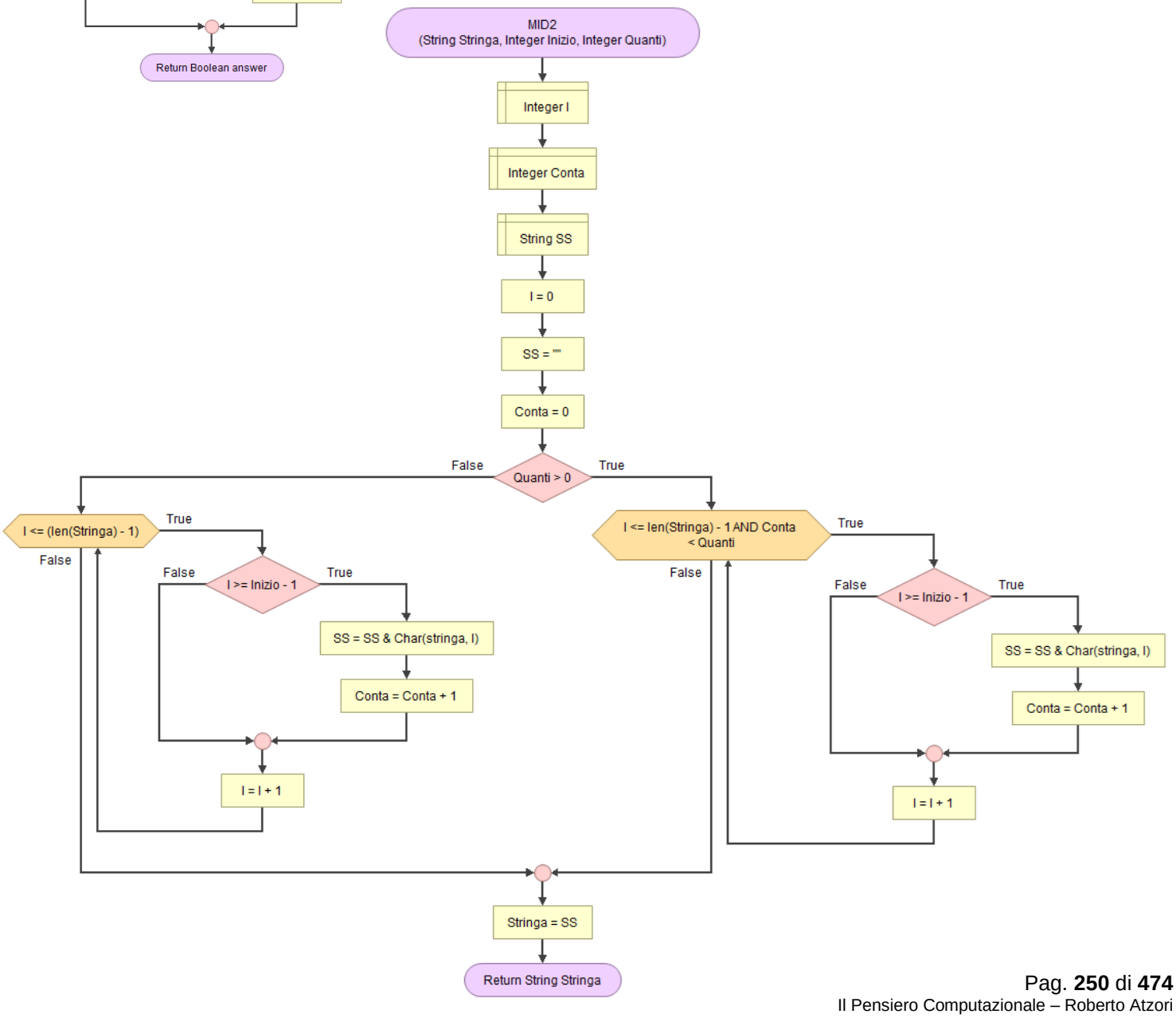






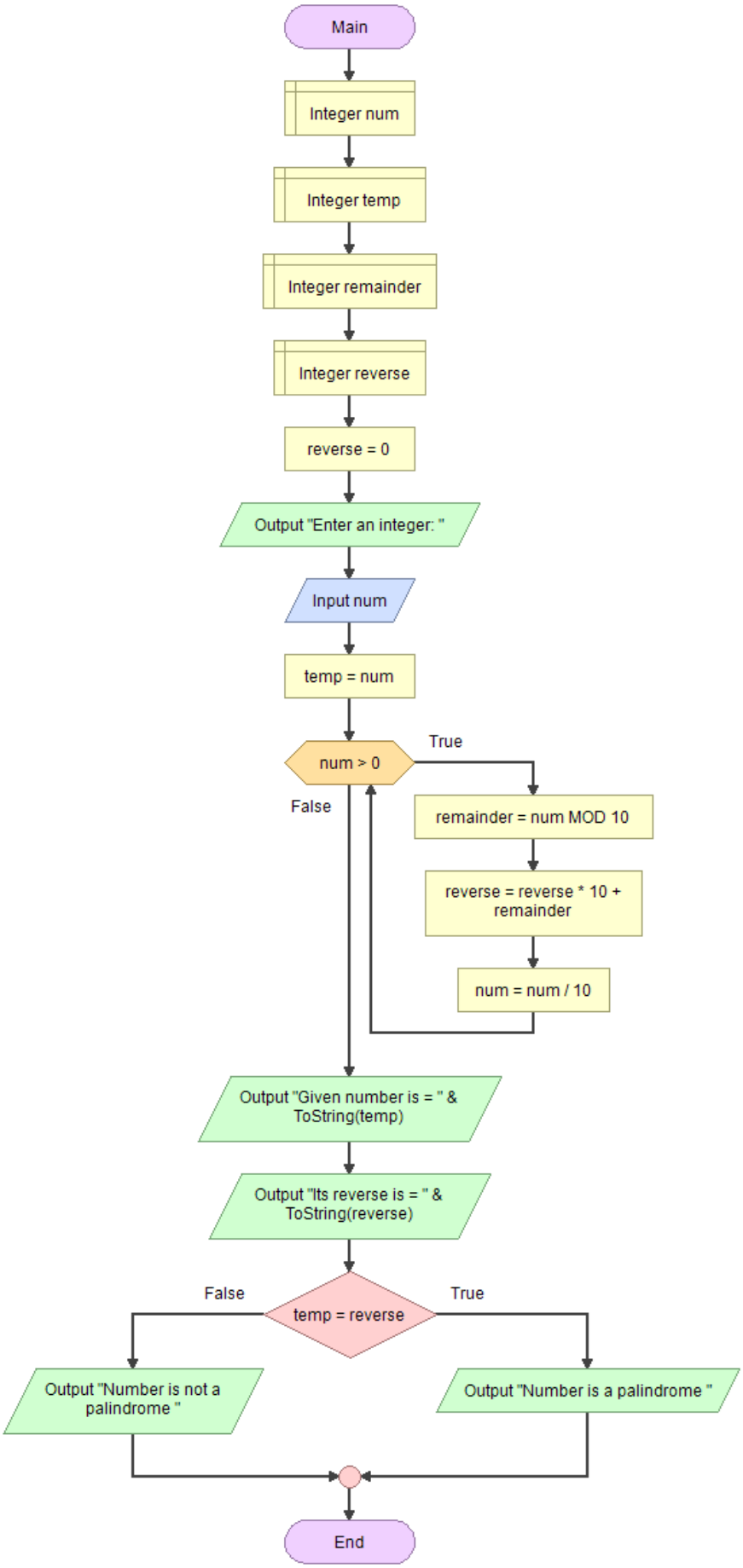

isPalindrome
(String st)
Boolean answer
Integer n
n = len(st)
String Array StrVet [n]
InitializeArrayString(StrVet)
String2Array(st, StrVet)
n = 0
False
True
answer = isPalRec(StrVet, st, 0, n-1, False)
An empty string is considered as palindrome
answer = True
Return Boolean answer


String2Array
(String stringa, String Array VettoreStringa)
Convert String to an array like ToCharArray
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len(stringa)
i = 0 to lunghezzaStringa - 1
Next
Done
vettoreStringa[i] = MID2(stringa, i + 1, 1)
End


MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

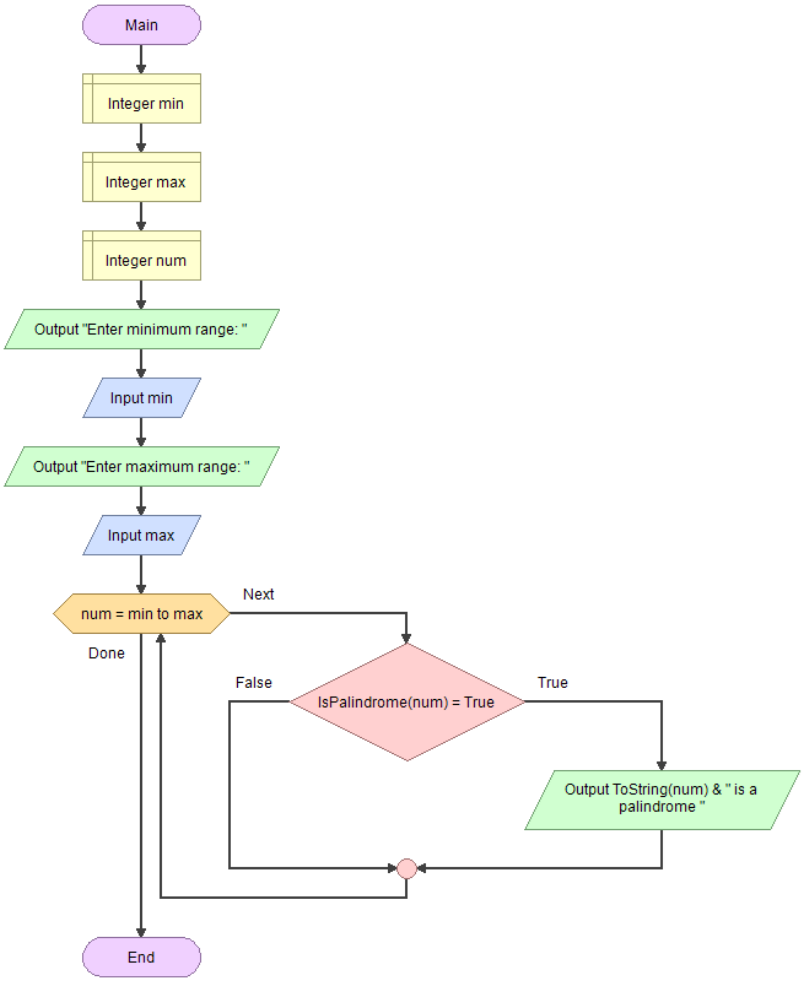
189.G ‖ Verificare se un numero è palindromo

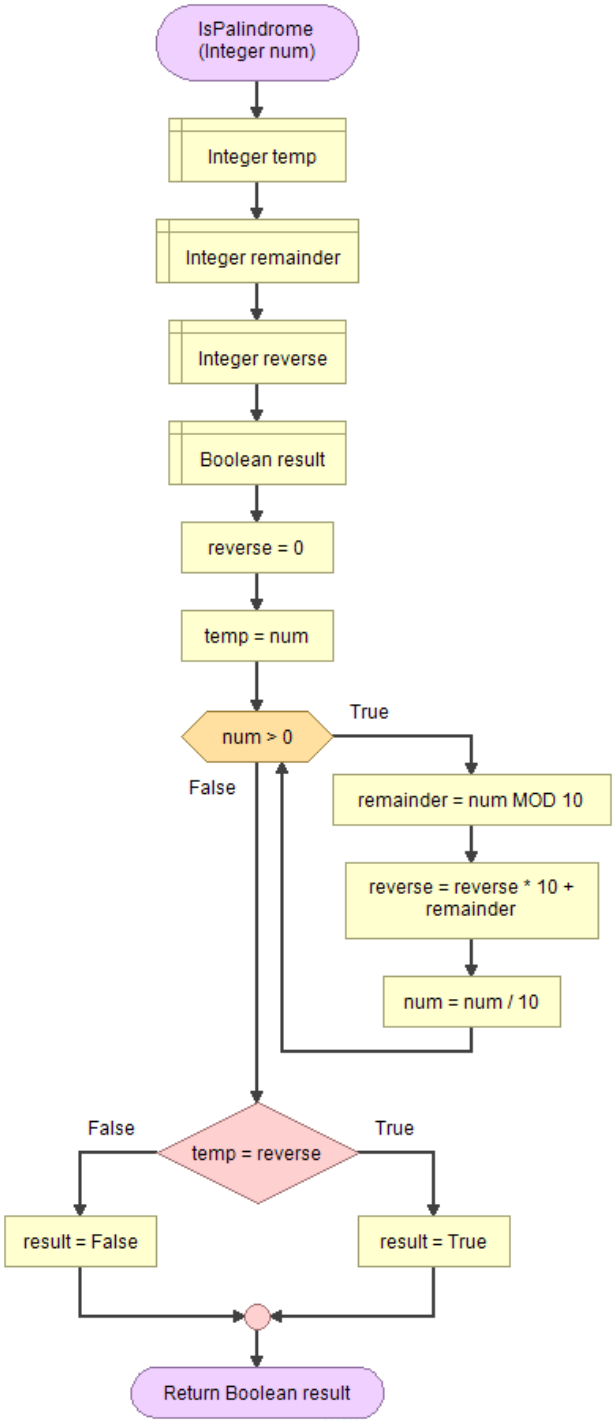
Check if number is a palindrome



190. G F ‖ Ricerca un numero palindromo in un intervallo

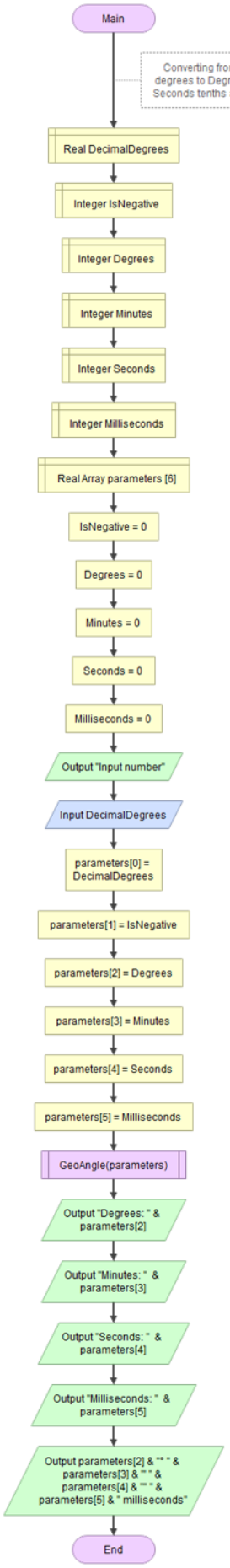
Check if number is a palindrome in a range

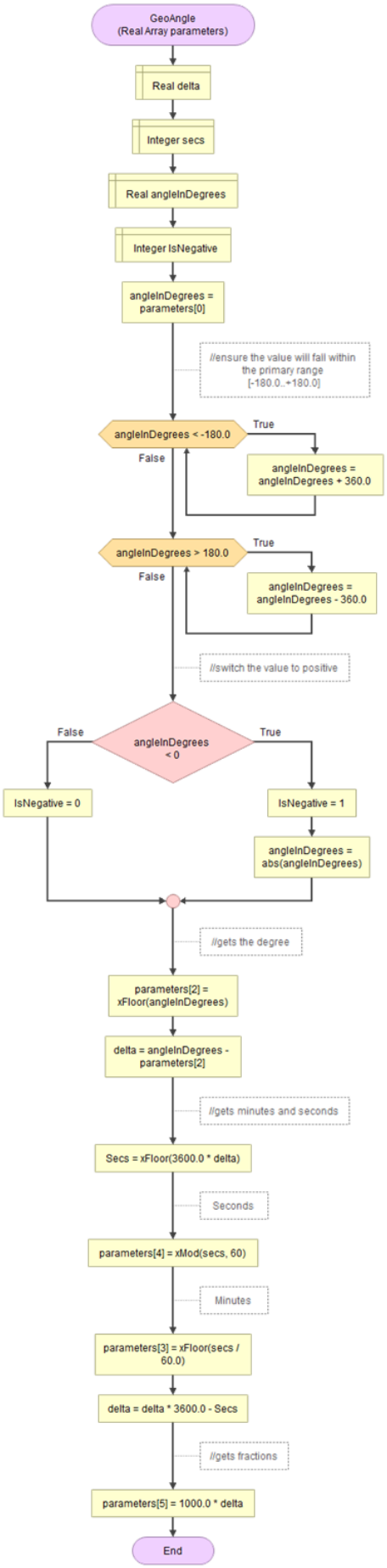


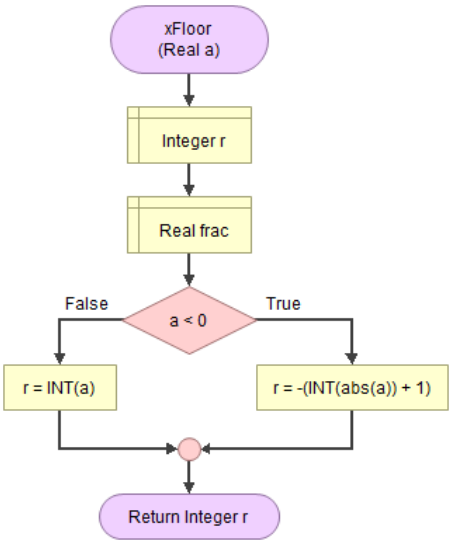


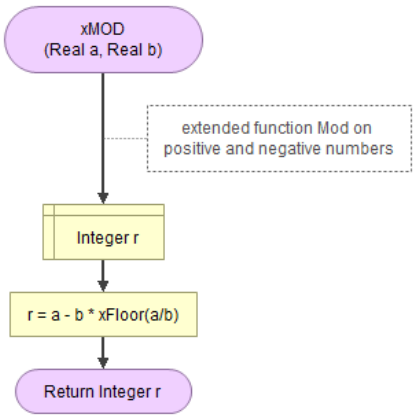
191.GSAF | Trasformazione angoli in gradi sessagesimali (angolo goniometrico)

Decimal degrees to Degrees Minutes Seconds tenths



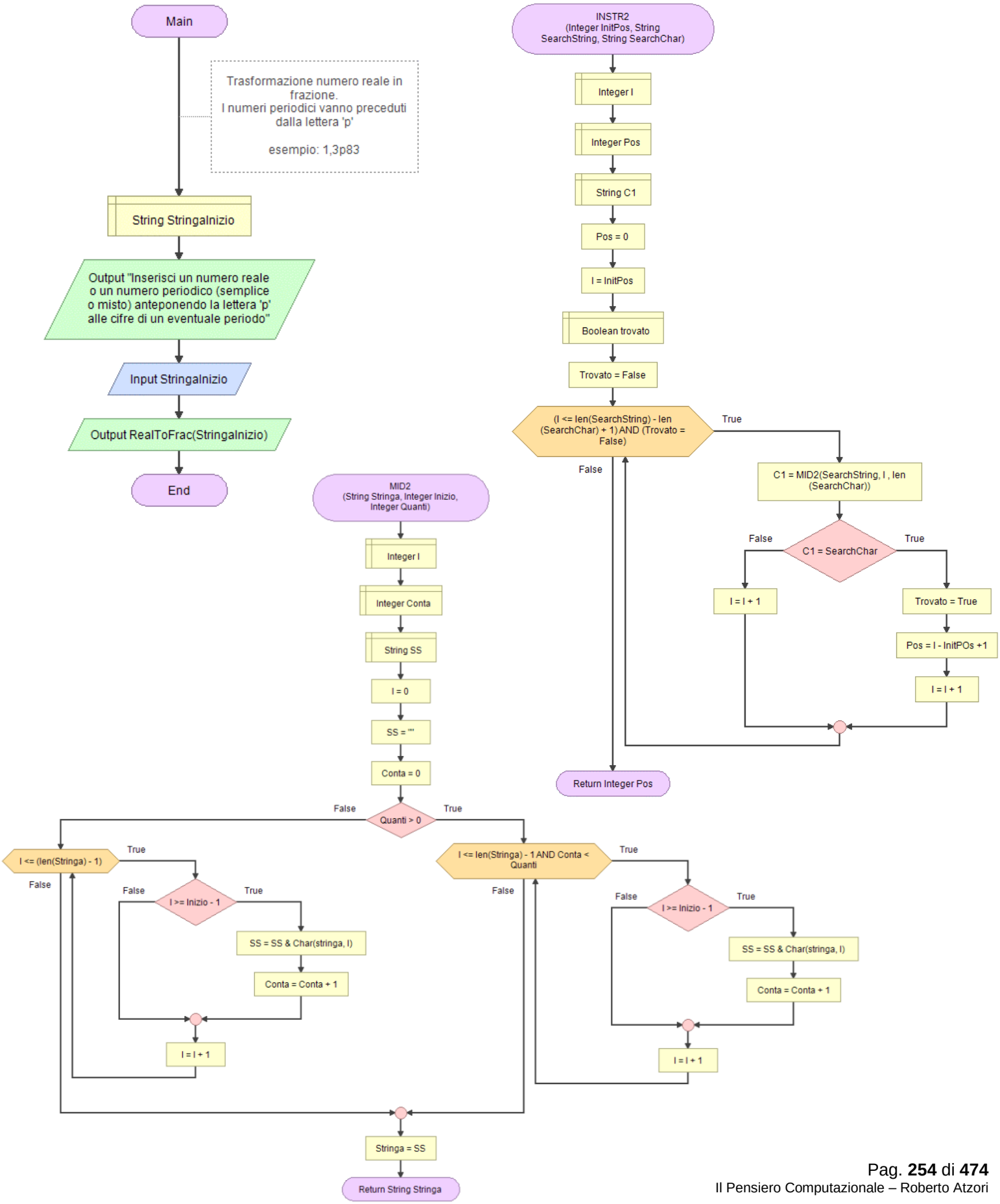


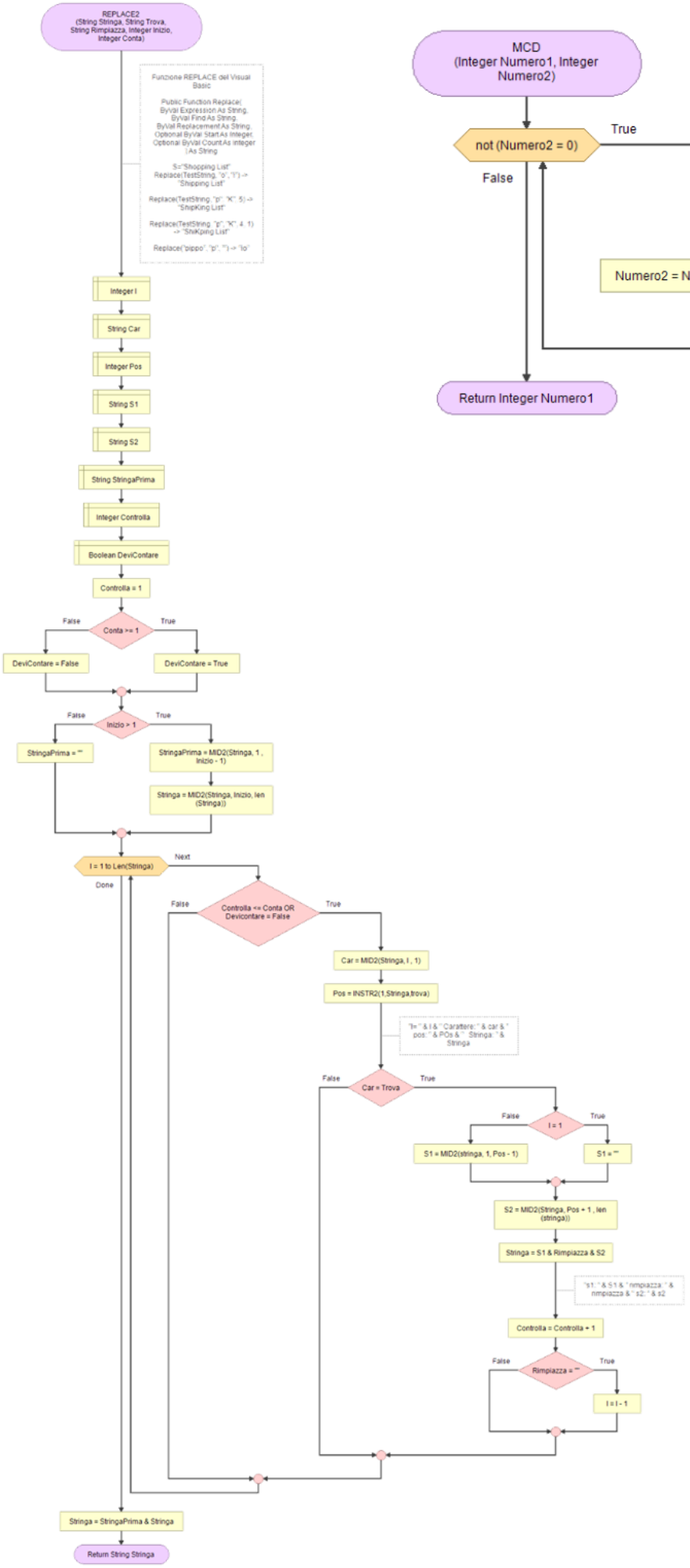


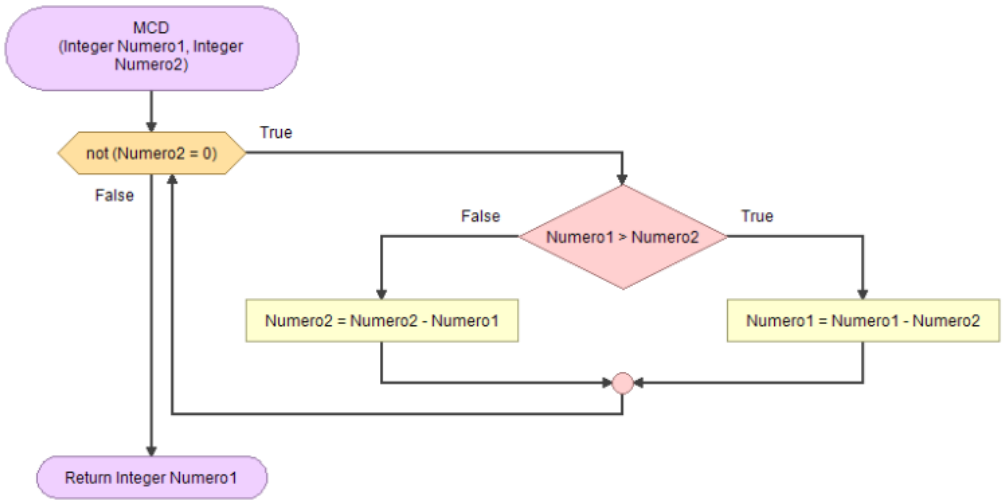


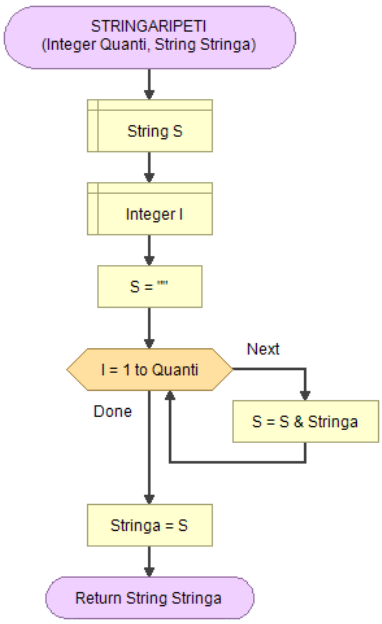
192.GSF | Trasformazione numero reale o periodico (semplice o misto) in frazione con semplificazione

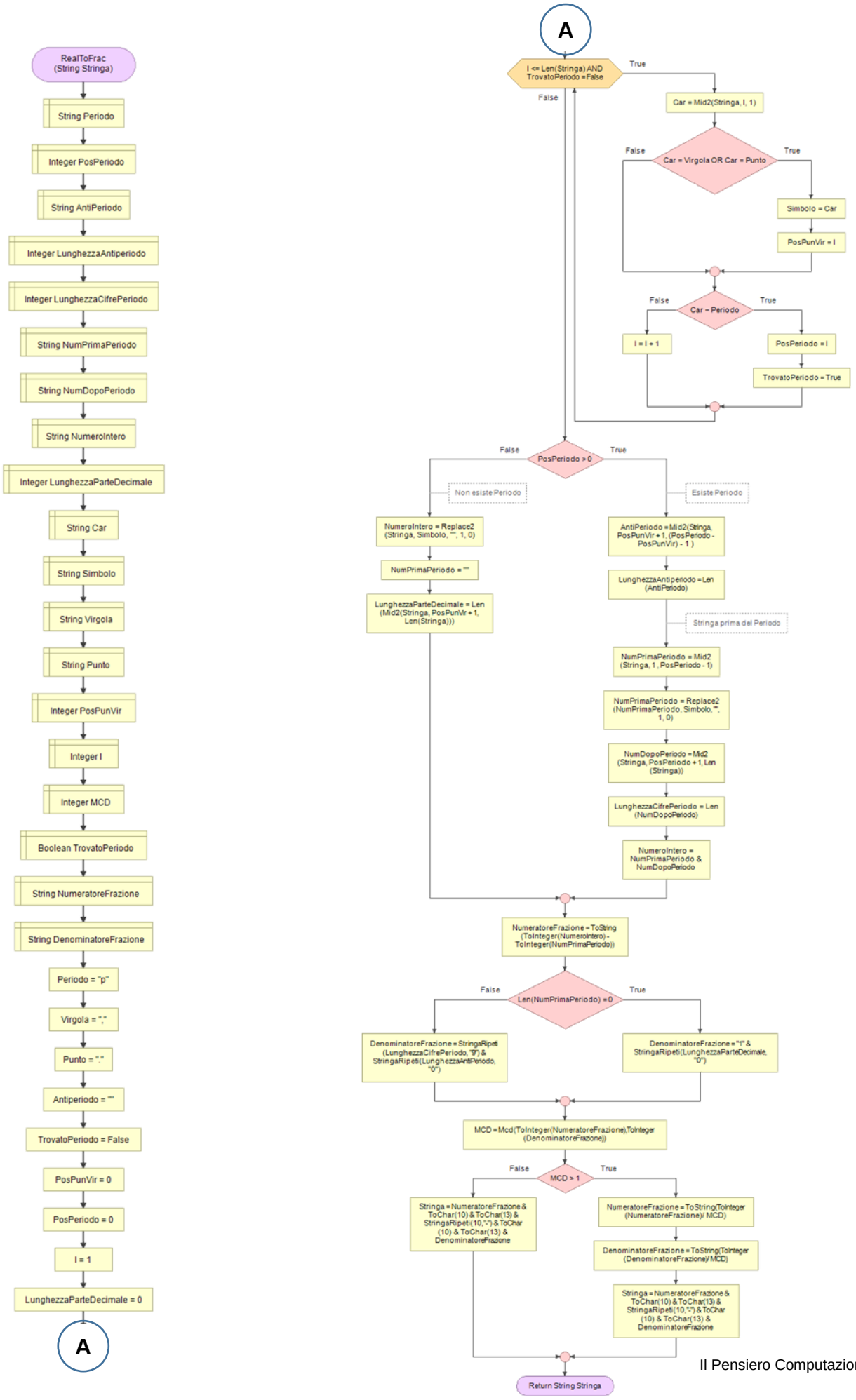
Transform real or periodic number into a semplified fraction




REPLACE2
(String Stringa, String Trova, String Rimpiazza, Integer Inizio, Integer Conta)
Funzione REPLACE del Visual Basic
Public Function Replace(ByVal Expression As String, ByVal Find As String, ByVal Replacement As String, Optional ByVal Start As Integer, Optional ByVal Count As Integer) As String
S="Shopping List"
Replace(TestString, "o", "I") -> "Shipping List"
Replace(TestString, "p", "K", 5) -> "ShipKing List"
Replace(TestString, "p", "K", 4, 1) -> "ShiKping List"
Replace("pippo", "p", "") -> "Io"
Integer I
String Car
Integer Pos
String S1
String S2
String StringaPrima
Integer Controlla
Boolean DeviContare
Controlla = 1
False
Conta >= 1
True
DeviContare = False
DeviContare = True
False
Inizio > 1
True
StringaPrima = ""
StringaPrima = MID2(Stringa, 1 , Inizio - 1)
Stringa = MID2(Stringa, Inizio, len (Stringa))
I = 1 to Len(Stringa)
Next
Done
False
Controlla <= Conta OR Devicontare = False
True
Car = MID2(Stringa, I , 1)
Pos = INSTR2(1,Stringa,trova)
"I=" & I & " Carattere: " & car & " pos: " & POs & " Stringa: " & Stringa
False
Car = Trova
True
False
I = 1
True
S1 = MID2(stringa, 1, Pos - 1)
S1 = ""
S2 = MID2(Stringa, Pos + 1 , len (stringa))
Stringa = S1 & Rimpiazza & S2
"s1: " & S1 & " rimpiazza: " & rimpiazza & " s2: " & s2
Controlla = Controlla + 1
False
Rimpiazza = ""
True
I = I - 1
Stringa = StringaPrima & Stringa
Return String Stringa

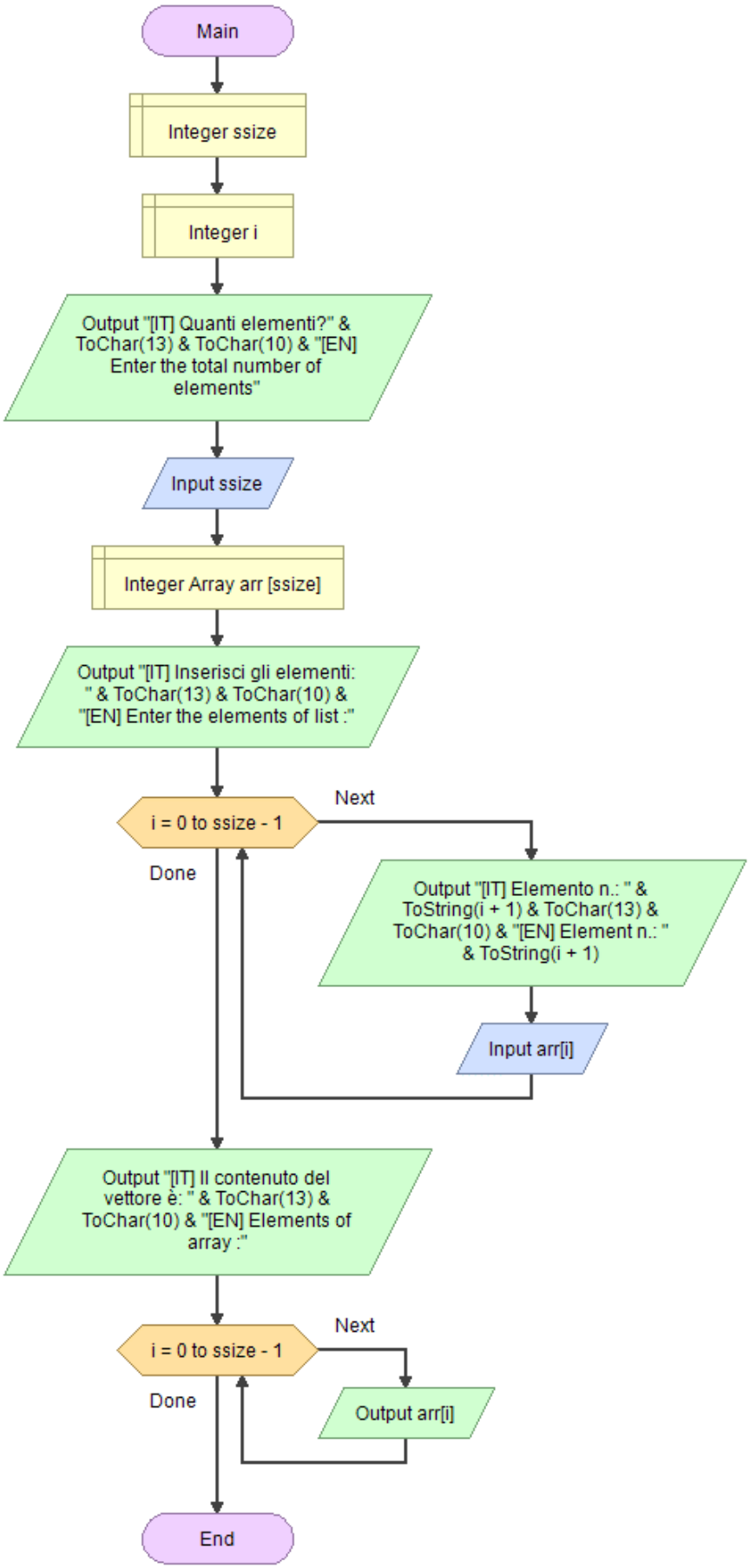

MCD
(Integer Numero1, Integer Numero2)
not (Numero2 = 0)
True
False
False
Numero1 > Numero2
True
Numero2 = Numero2 - Numero1
Numero1 = Numero1 - Numero2
Return Integer Numero1


STRINGARIPETI
(Integer Quanti, String Stringa)
String S
Integer I
S = ""
I = 1 to Quanti
Next
Done
S = S & Stringa
Stringa = S
Return String Stringa


RealToFrac (String Stringa)
String Periodo
Integer PosPeriodo
String AntiPeriodo
Integer LunghezzaAntiperiodo
Integer LunghezzaCifrePeriodo
String NumPrimaPeriodo
String NumDopoPeriodo
String NumeroIntero
Integer LunghezzaParteDecimale
String Car
String Simbolo
String Virgola
String Punto
Integer PosPunVir
Integer I
Integer MCD
Boolean TrovatoPeriodo
String NumeratoreFrazione
String DenominatoreFrazione
Periodo = "p"
Virgola = ","
Punto = "."
Antiperiodo = ""
TrovatoPeriodo = False
PosPunVir = 0
PosPeriodo = 0
I = 1
LunghezzaParteDecimale = 0
A
A
I <= Len(Stringa) AND TrovatoPeriodo = False
True
False
Car = Mid2(Stringa, I, 1)
Car = Virgola OR Car = Punto
False
True
Simbolo = Car
PosPunVir = I
Car = Periodo
False
True
I = I + 1
PosPeriodo = I
TrovatoPeriodo = True
PosPeriodo > 0
False
True
Non esiste Periodo
Esiste Periodo
NumeroIntero = Replace2 (Stringa, Simbolo, "", 1, 0)
NumPrimaPeriodo = ""
LunghezzaParteDecimale = Len (Mid2(Stringa, PosPunVir + 1, Len(Stringa)))
AntiPeriodo = Mid2(Stringa, PosPunVir + 1, (PosPeriodo - PosPunVir) - 1)
LunghezzaAntiperiodo = Len (AntiPeriodo)
Stringa prima del Periodo
NumPrimaPeriodo = Mid2 (Stringa, 1 , PosPeriodo - 1)
NumPrimaPeriodo = Replace2 (NumPrimaPeriodo, Simbolo, "", 1, 0)
NumDopoPeriodo = Mid2 (Stringa, PosPeriodo + 1, Len (Stringa))
LunghezzaCifrePeriodo = Len (NumDopoPeriodo)
NumeroIntero = NumPrimaPeriodo & NumDopoPeriodo
NumeratoreFrazione = ToString (ToInteger(NumeroIntero) - ToInteger(NumPrimaPeriodo))
Len(NumPrimaPeriodo) = 0
False
True
DenominatoreFrazione = StringaRipeti (LunghezzaCifrePeriodo, "9") & StringaRipeti(LunghezzaAntiPeriodo, "0")
DenominatoreFrazione = "1" & StringaRipeti(LunghezzaParteDecimale, "0")
MCD = Mcd(ToInteger(NumeratoreFrazione),ToInteger (DenominatoreFrazione))
MCD > 1
False
True
Stringa = NumeratoreFrazione & ToChar(10) & ToChar(13) & StringaRipeti(10,"-") & ToChar (10) & ToChar(13) & DenominatoreFrazione
NumeratoreFrazione = ToString(ToInteger (NumeratoreFrazione)/ MCD)
DenominatoreFrazione = ToString(ToInteger (DenominatoreFrazione)/ MCD)
Stringa = NumeratoreFrazione & ToChar(10) & ToChar(13) & StringaRipeti(10,"-") & ToChar (10) & ToChar(13) & DenominatoreFrazione
Return String Stringa

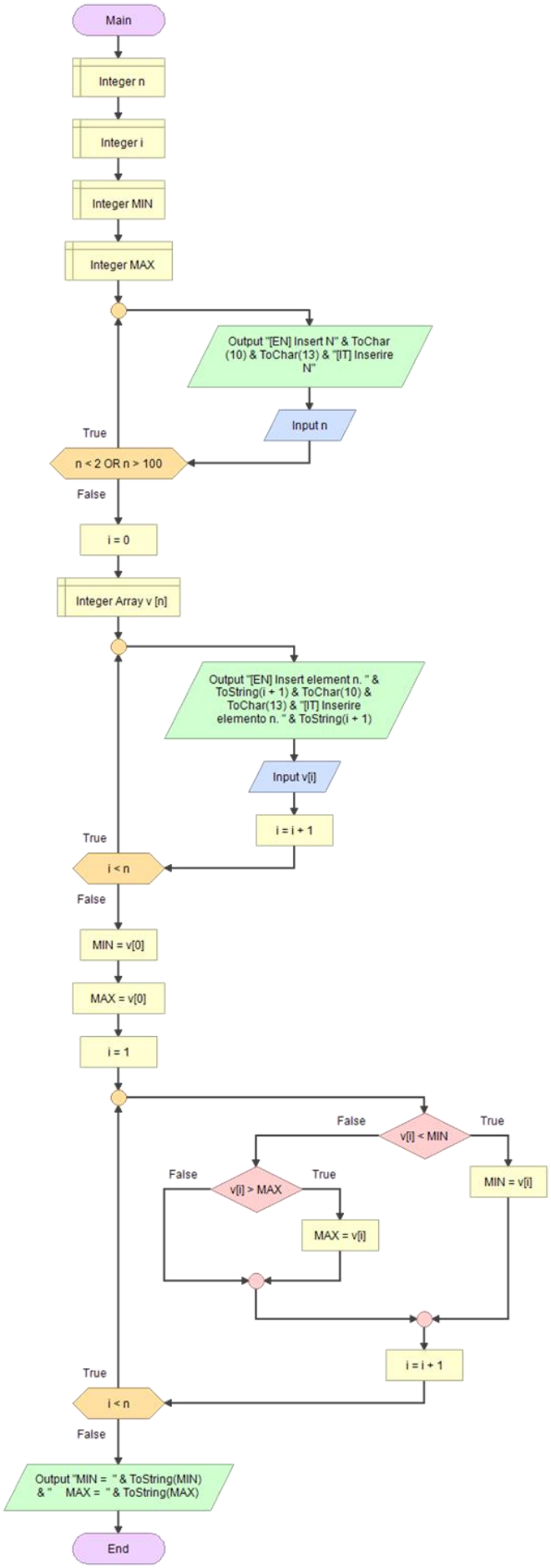
193.G A ‖ Lettura e scrittura di elementi in un vettore

Read and Print Array Elements



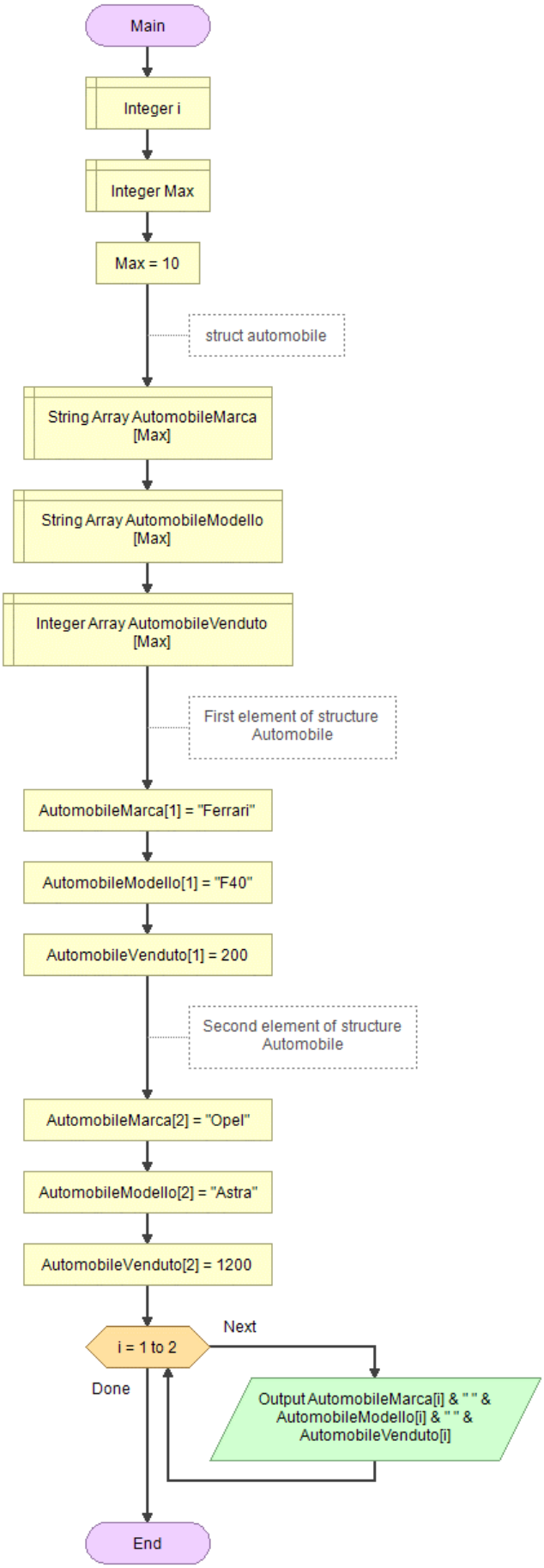
194. **G A** ‖ Data una sequenza di numeri visualizzare il numero massimo e quello minimo

Given a sequence of numbers display the maximum number and the minimum



195.GA | Come definire una struttura

How to define a structure STRUCT

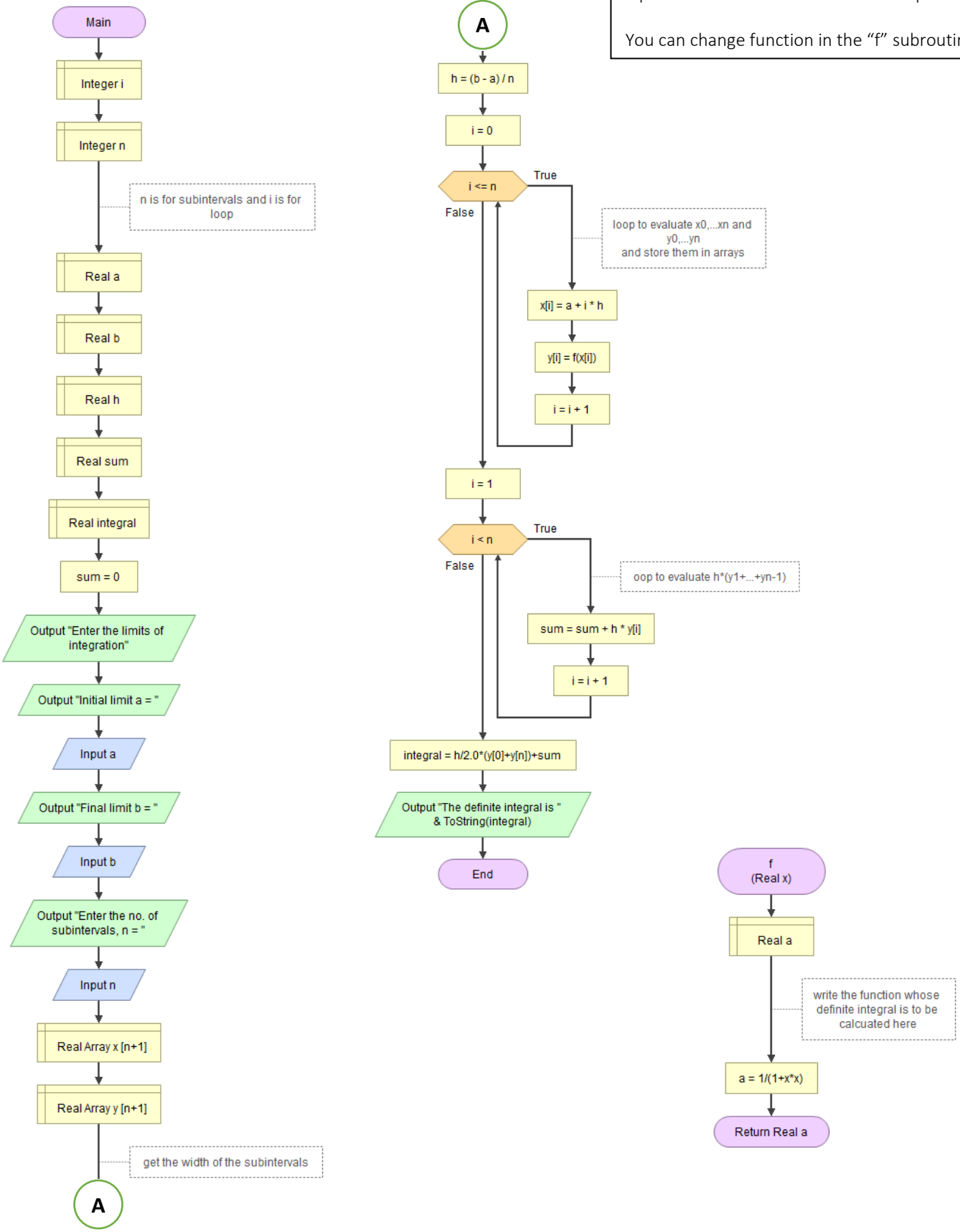


196. G A F ‖ Calcolo di un integrale definito (metodo trapezoidale)

Trapezoidal Method for the evaluation of Definite Integrals

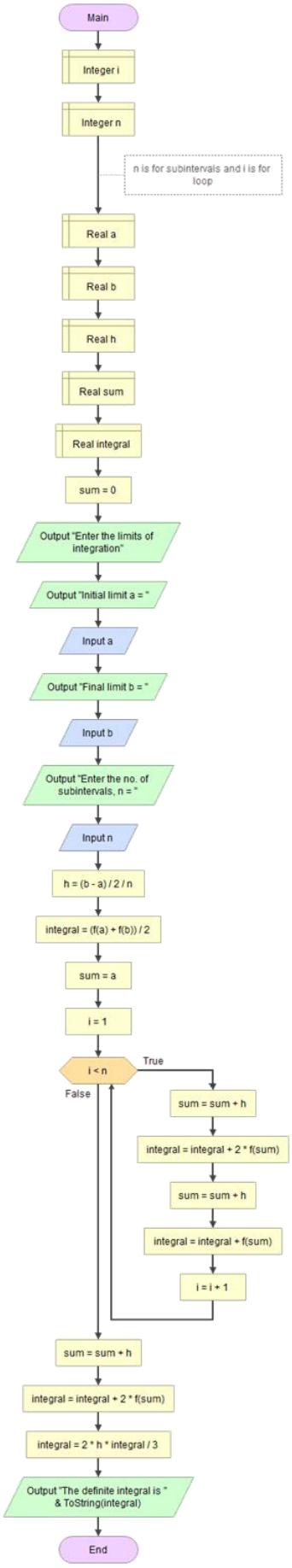
È possibile cambiare la funzione nella procedura "f"

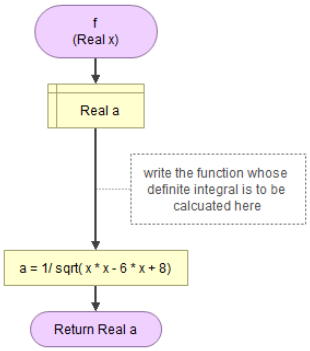
You can change function in the "f" subroutine



197.G F ‖ Calcolo di un integrale definito (metodo Simpson)

Simpson Method for the evaluation of Definite Integrals



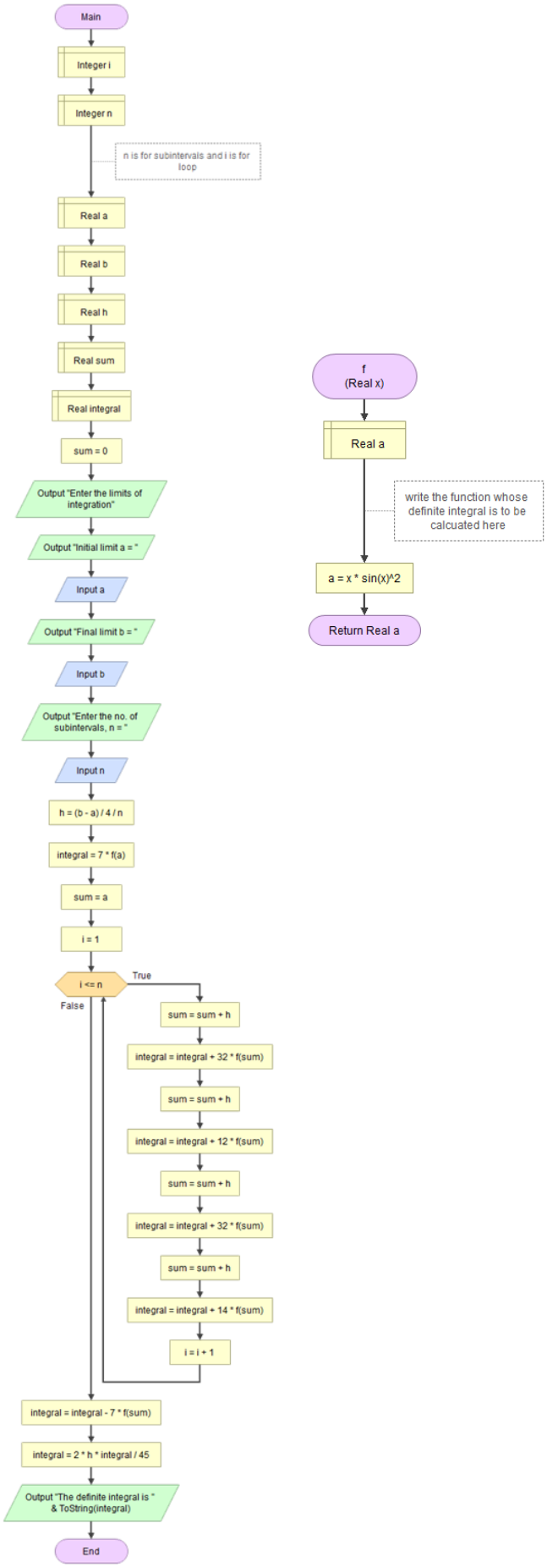


È possibile cambiare la funzione nella procedura "f"

You can change function in the "f" subroutine

198. G F ‖ Calcolo di un integrale definito (metodo Villarceau)

Villarceau Method for the evaluation of Definite Integrals

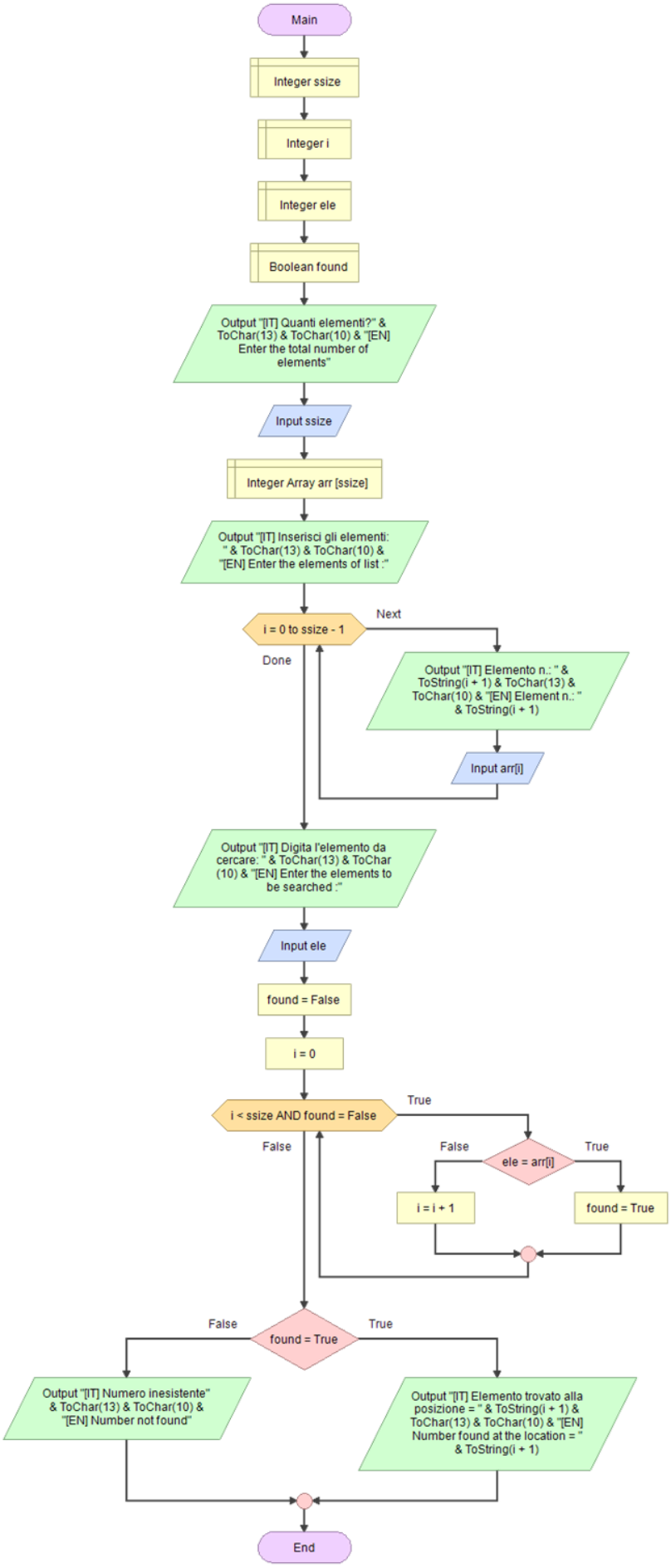


È possibile cambiare la funzione nella procedura "f"

You can change function in the "f" subroutine

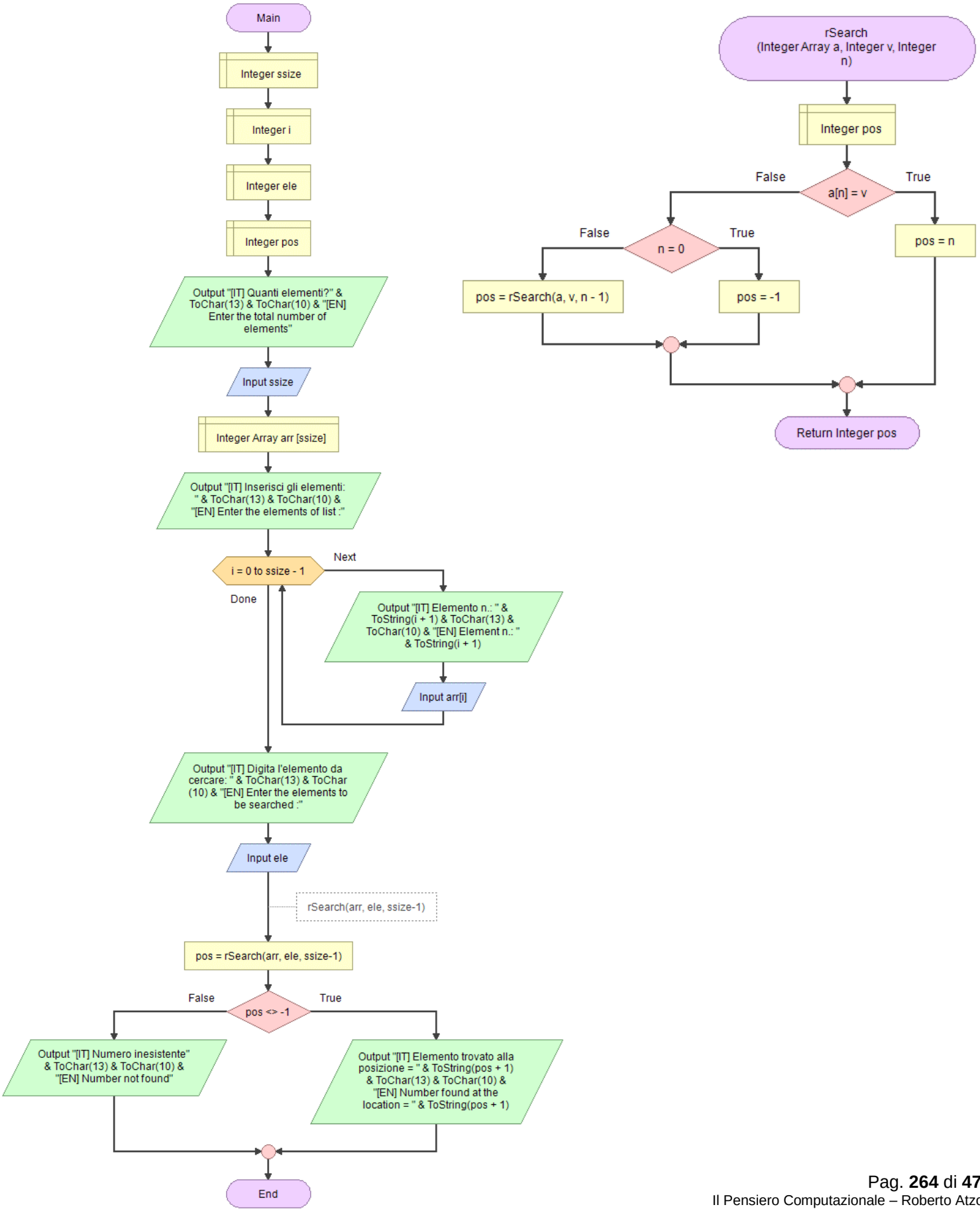
199. G A | Ricerca sequenziale di un elemento in un vettore

Sequential search of an element in Array



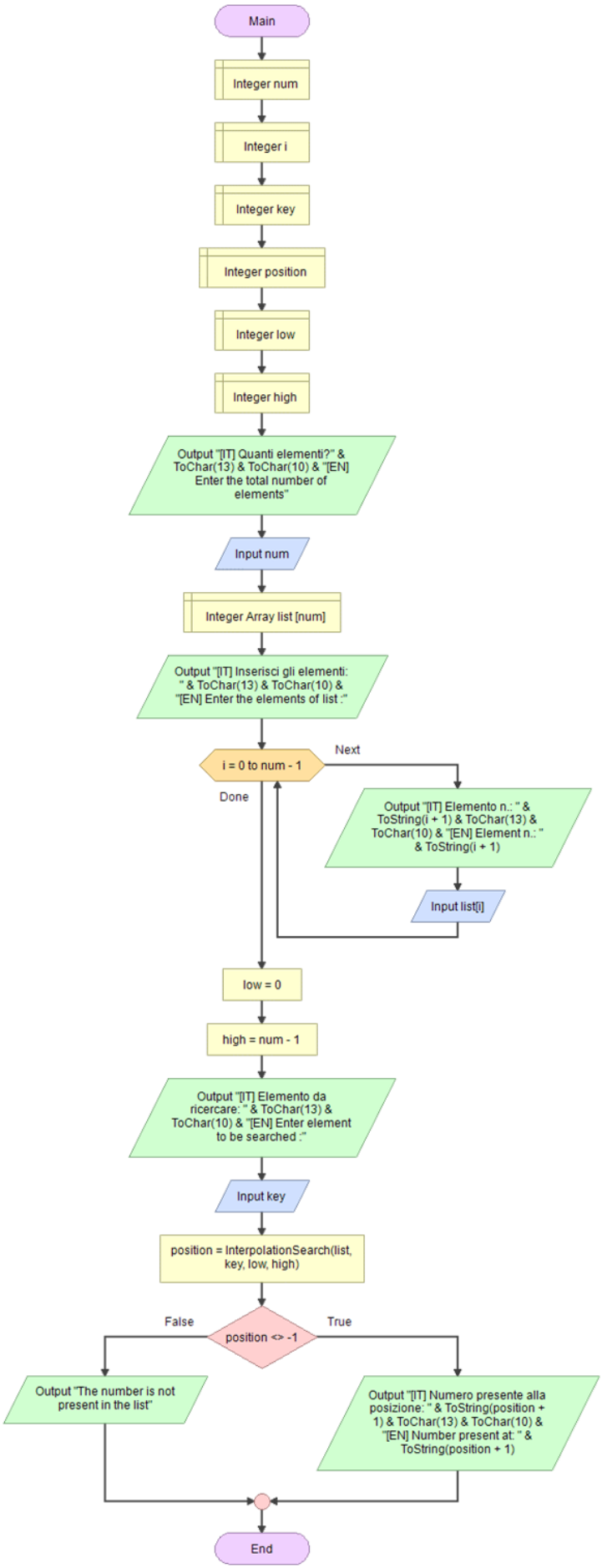
200.GAFR | Ricerca sequenziale di un elemento in un vettore (ricorsiva)

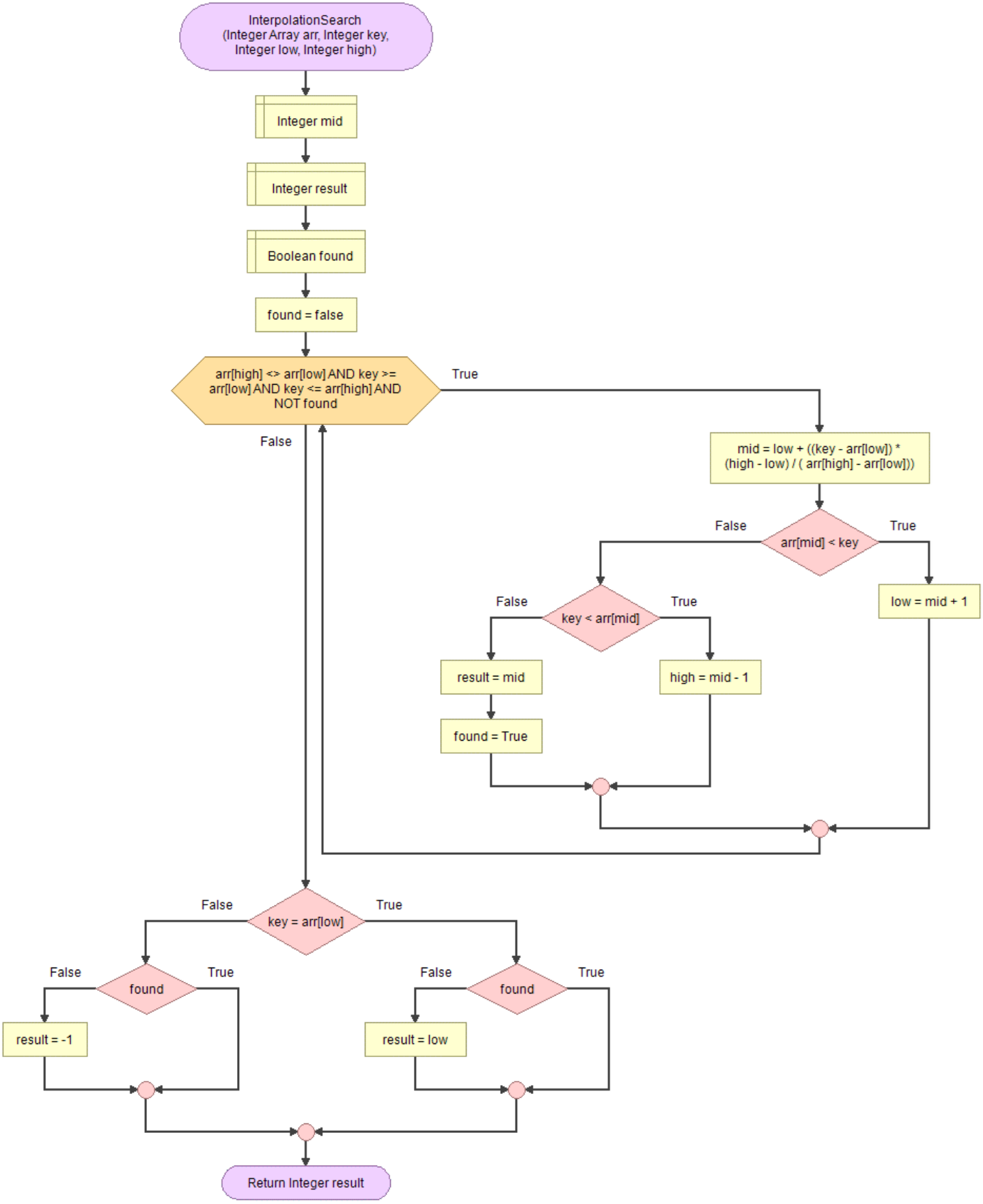
Recursive sequential search of an element in Array



201.GAF ‖ Ricerca per interpolazione su array

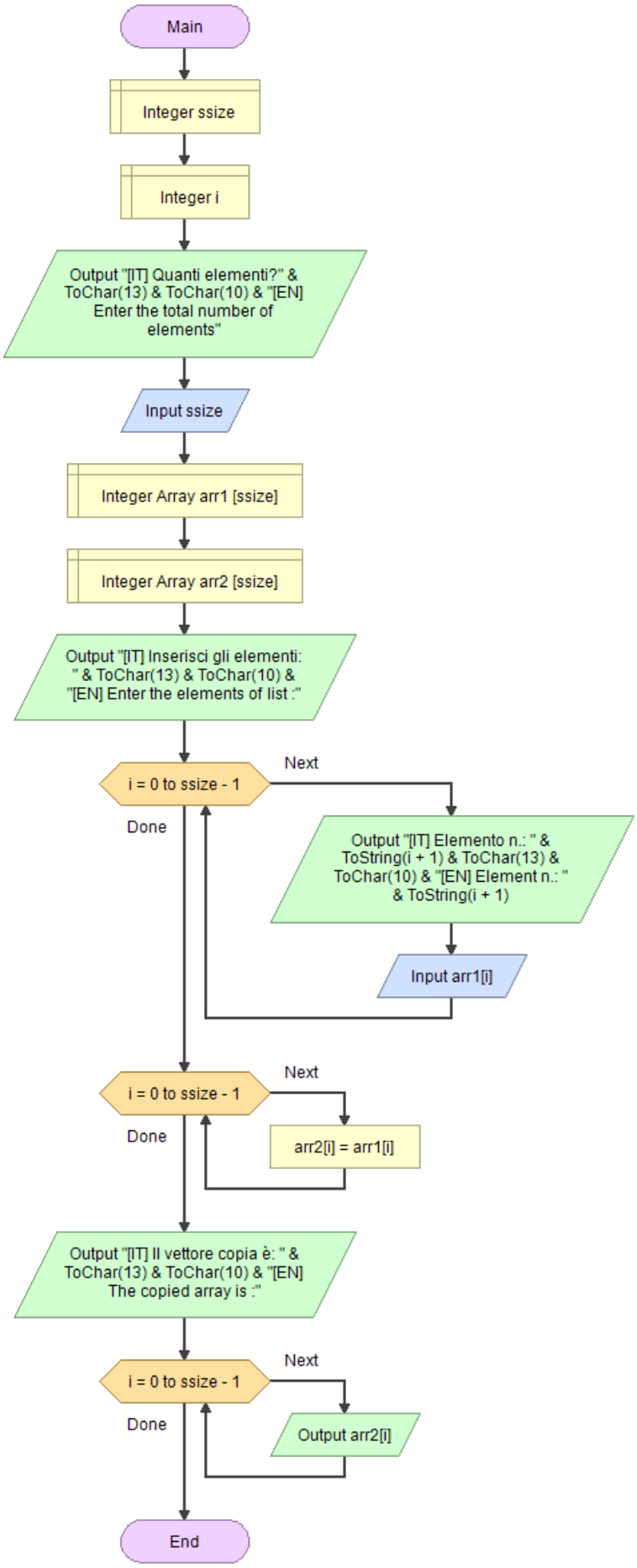
Interpolation search




InterpolationSearch
(Integer Array arr, Integer key,
Integer low, Integer high)
Integer mid
Integer result
Boolean found
found = false
arr[high] <> arr[low] AND key >=
arr[low] AND key <= arr[high] AND
NOT found
True
False
mid = low + ((key - arr[low]) *
(high - low) / (arr[high] - arr[low]))
False
arr[mid] < key
True
low = mid + 1
False
key < arr[mid]
True
result = mid
high = mid - 1
found = True
False
key = arr[low]
True
False
found
True
result = -1
False
found
True
result = low
Return Integer result

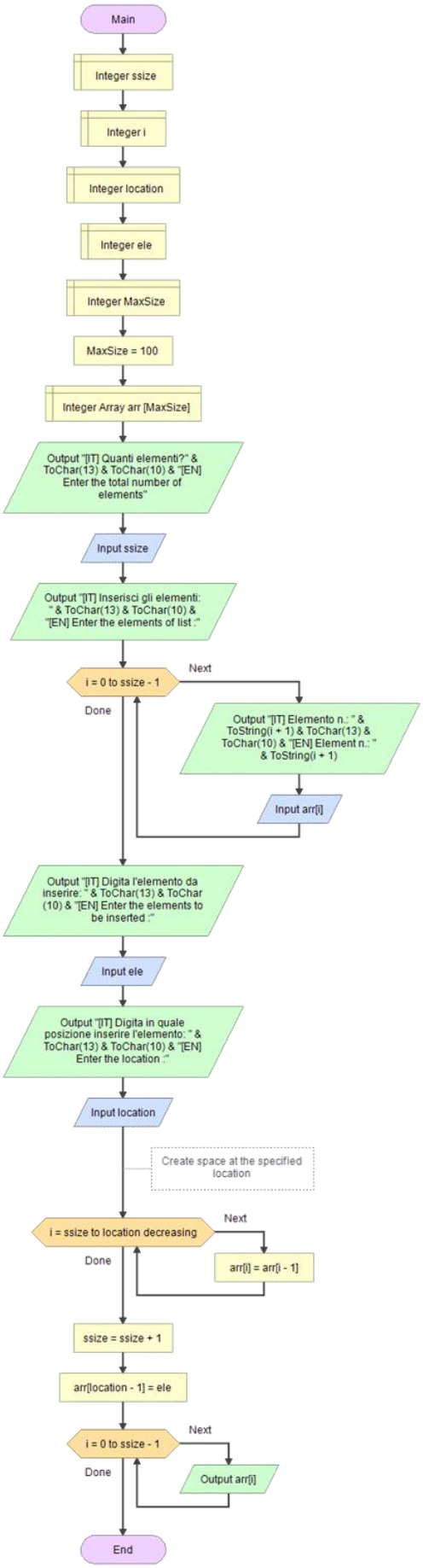
202.GA ‖ Copiare tutti gli elementi di un vettore in un altro vettore

Copy all elements of an array into another array



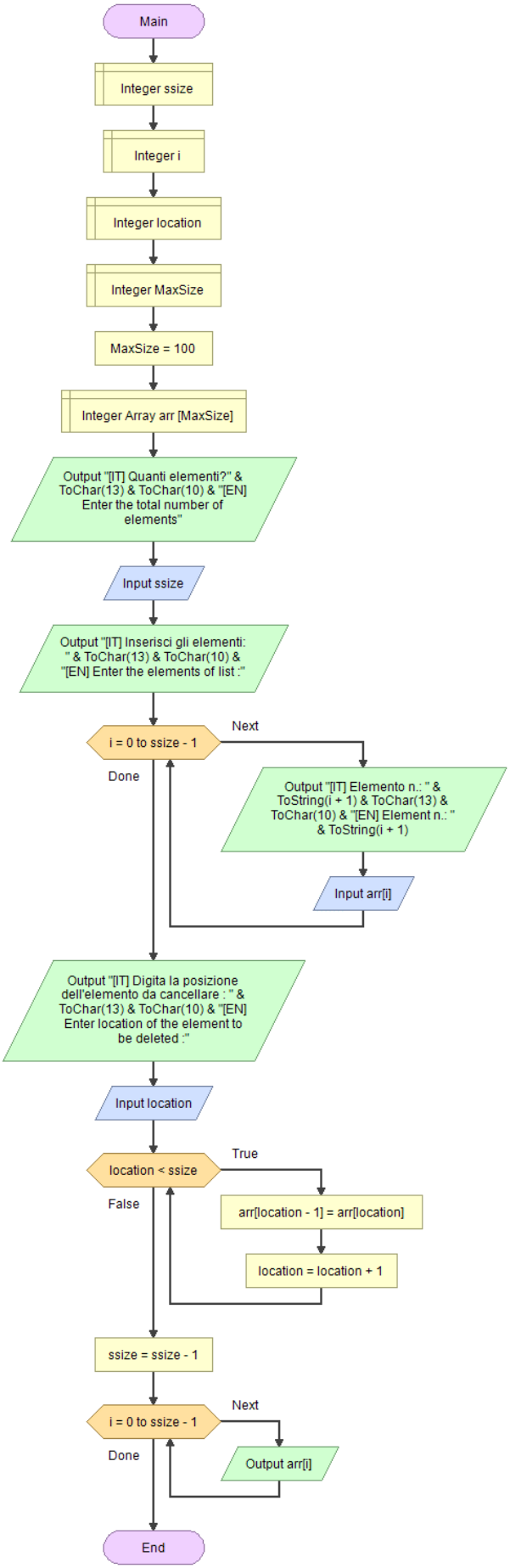
203.G A ‖ Inserire un elemento in un vettore

Insert an element in an Array



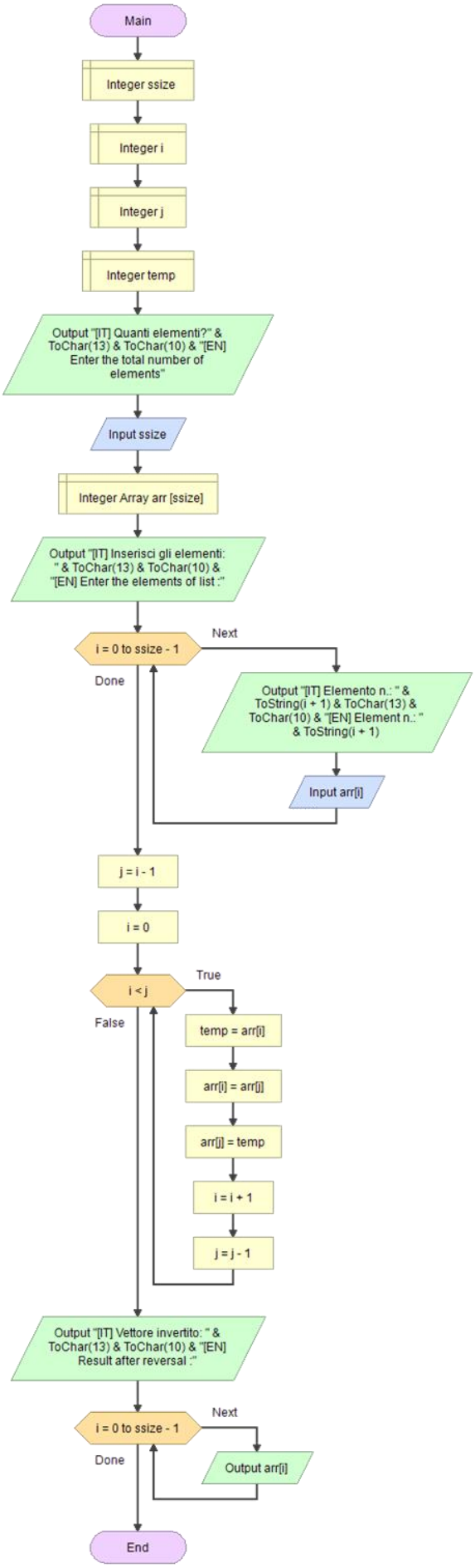
204.G A | Cancellare un elemento in un vettore

Delete an element from the specified location from Array



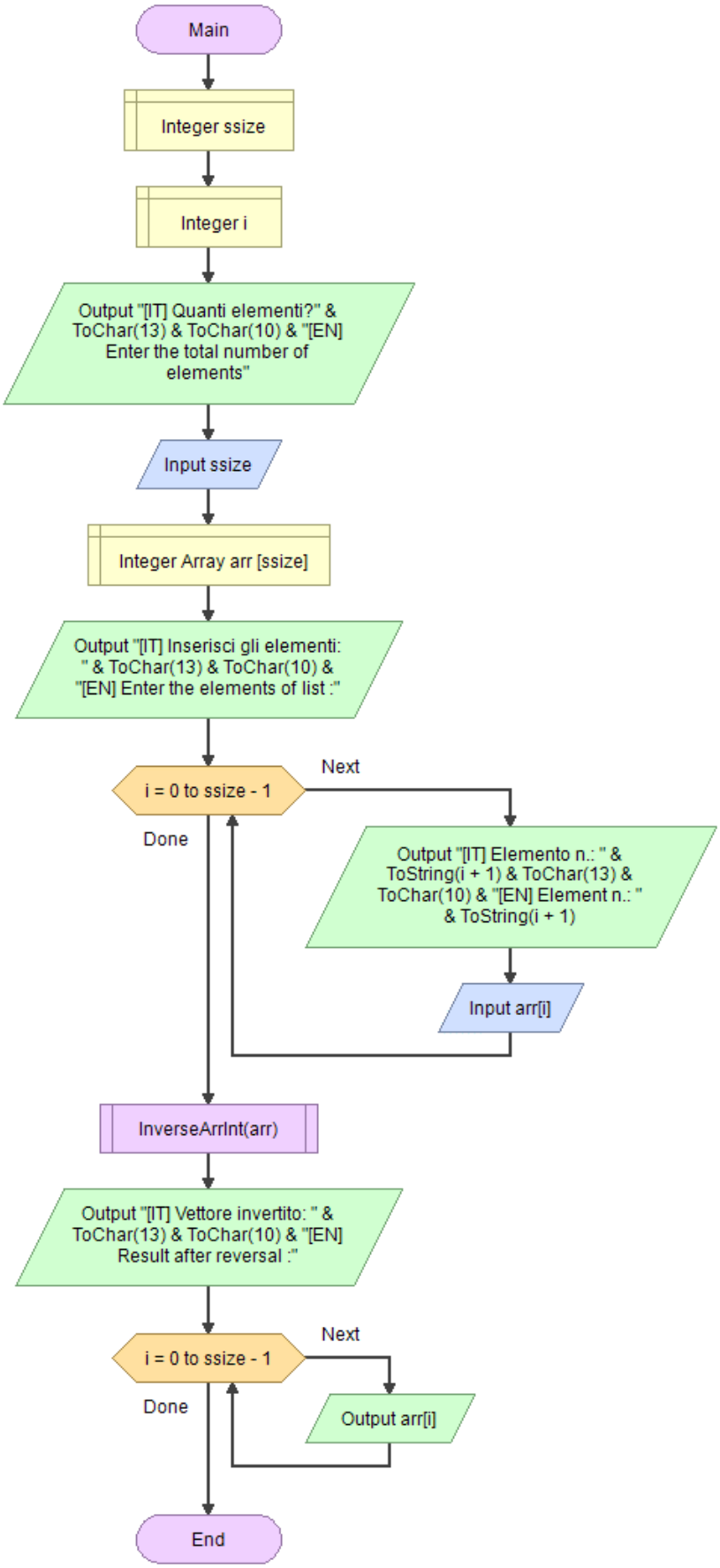
205.GA | Inversione degli elementi di un vettore

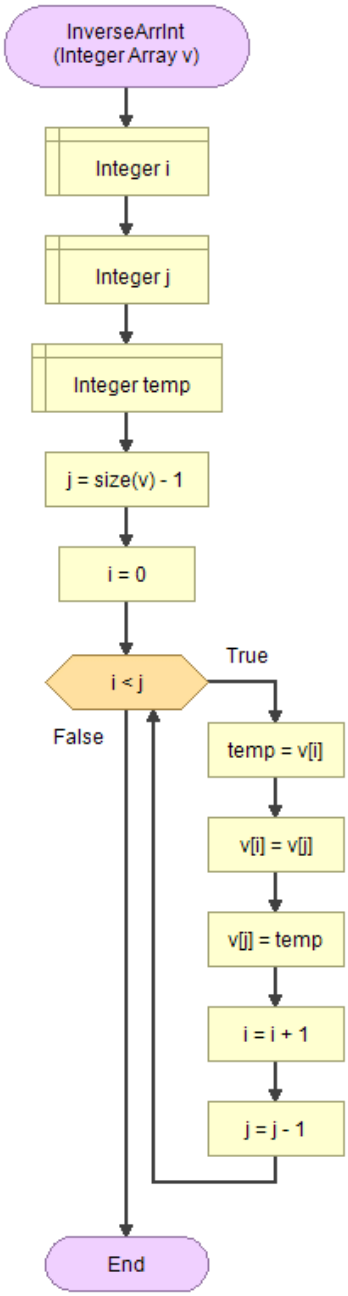
Reversing Array Elements



206. G A F ‖ Inversione degli elementi di un vettore (con funzione)

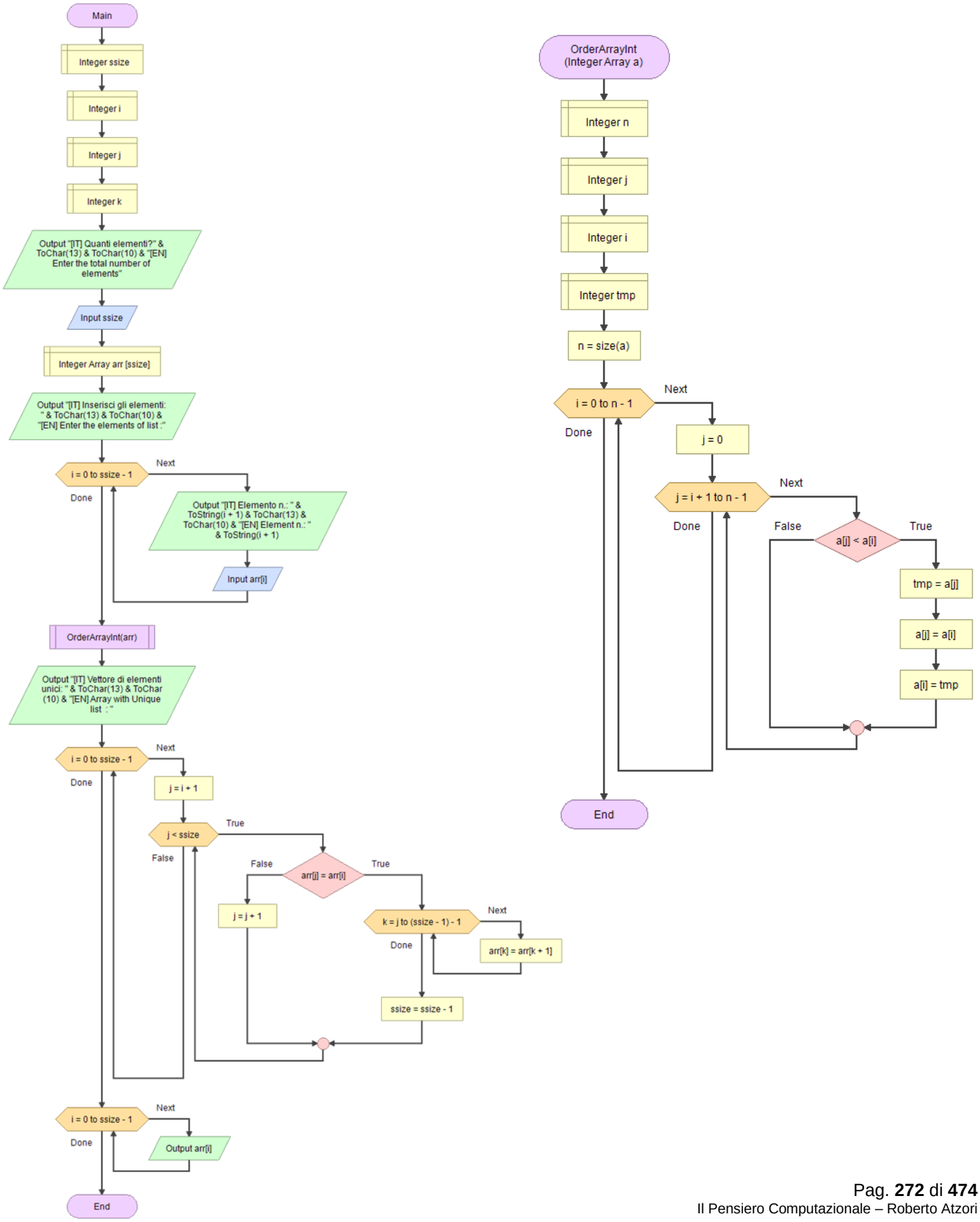
Reversing Array Elements (function)





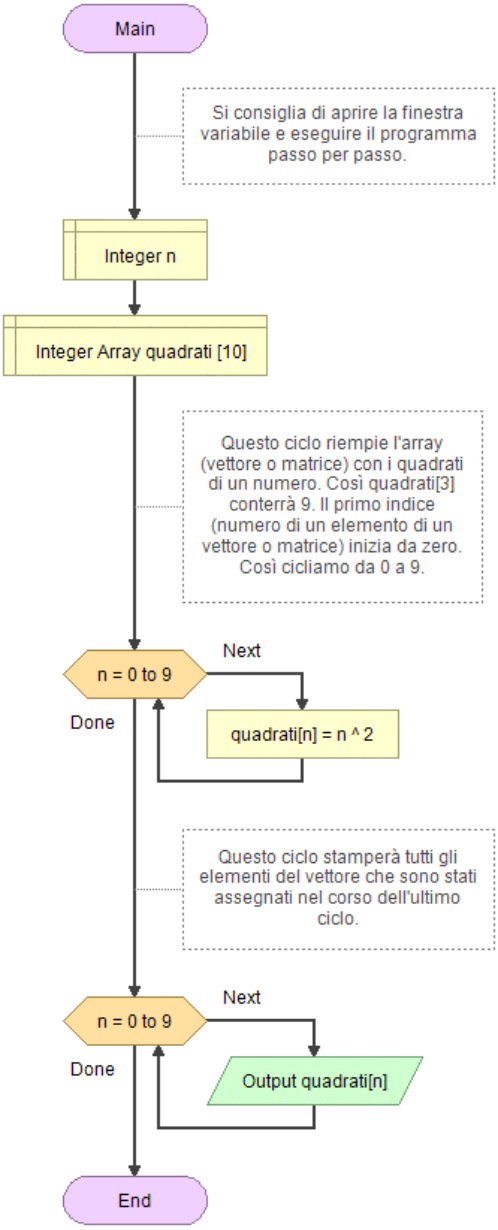
207.GAF ‖ Cancellare gli elementi duplicati di un vettore

Delete duplicate elements in an array



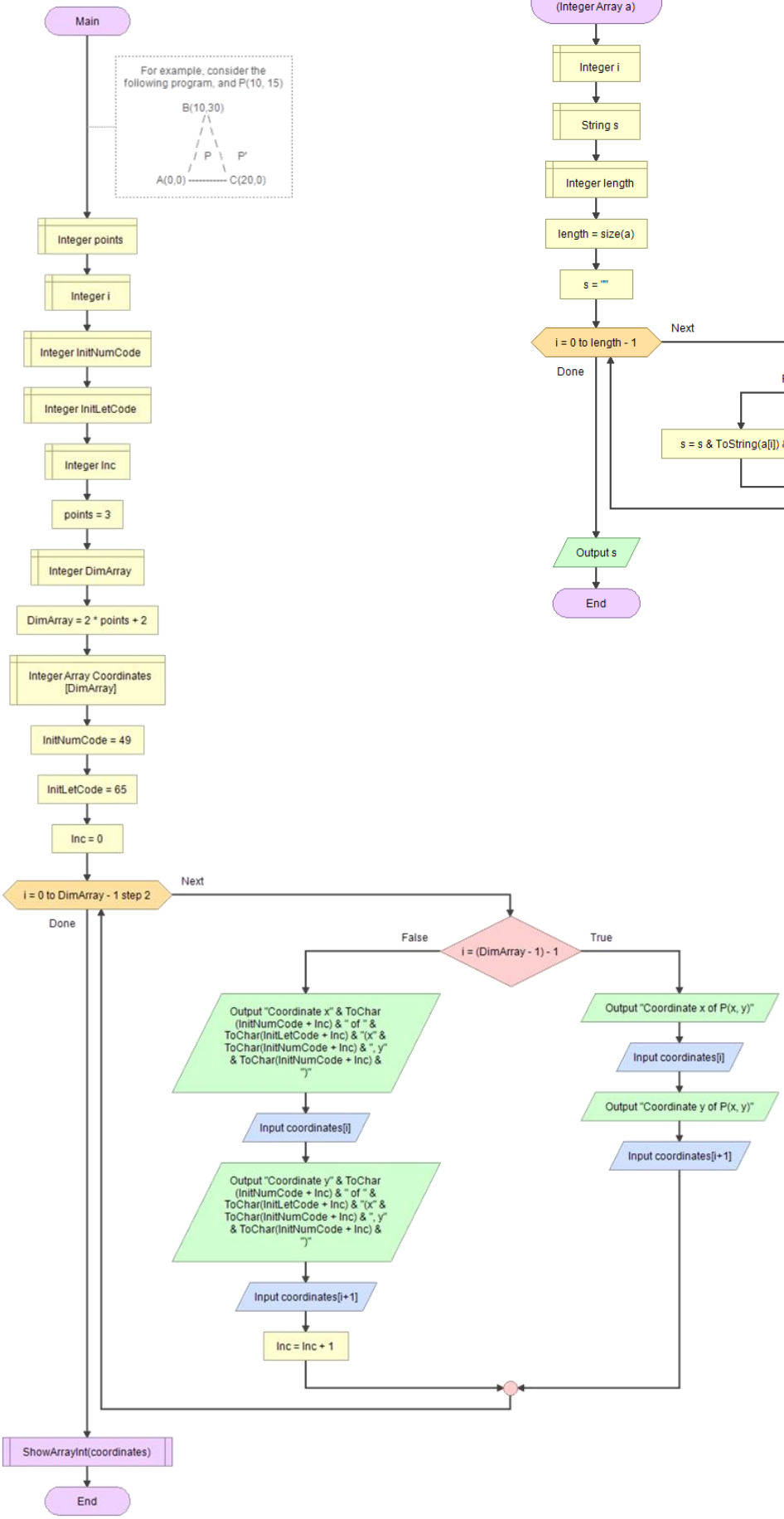
208. G A ‖ Riempire un vettore con i quadrati dei numeri

Fill an array with squared numbers



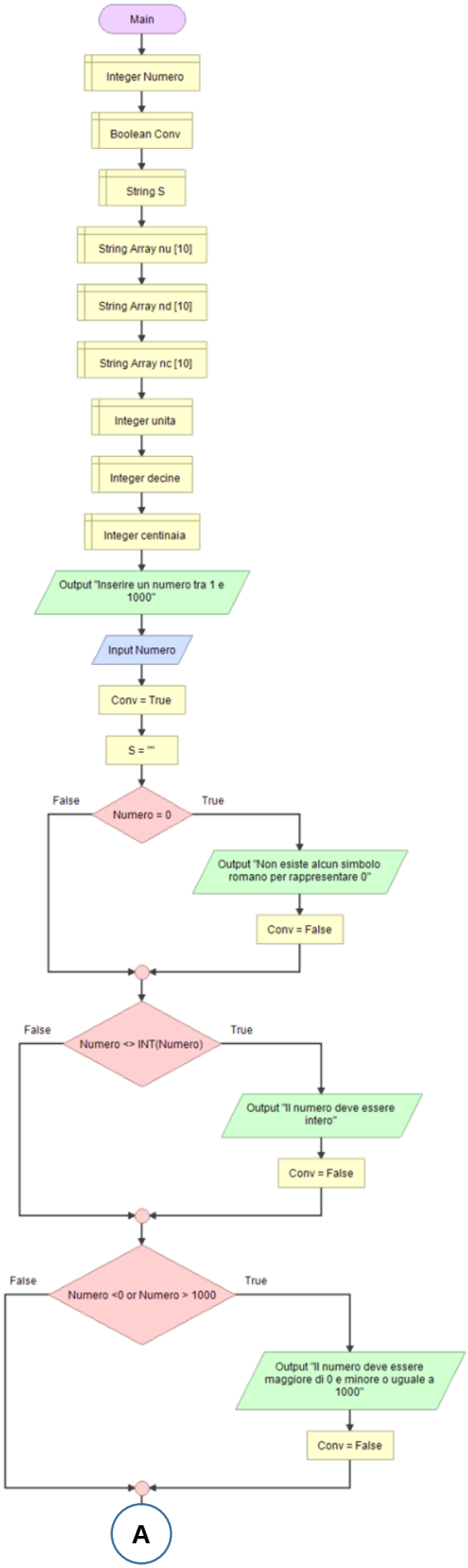
209.G A ‖ Inserire delle coordinate in un array

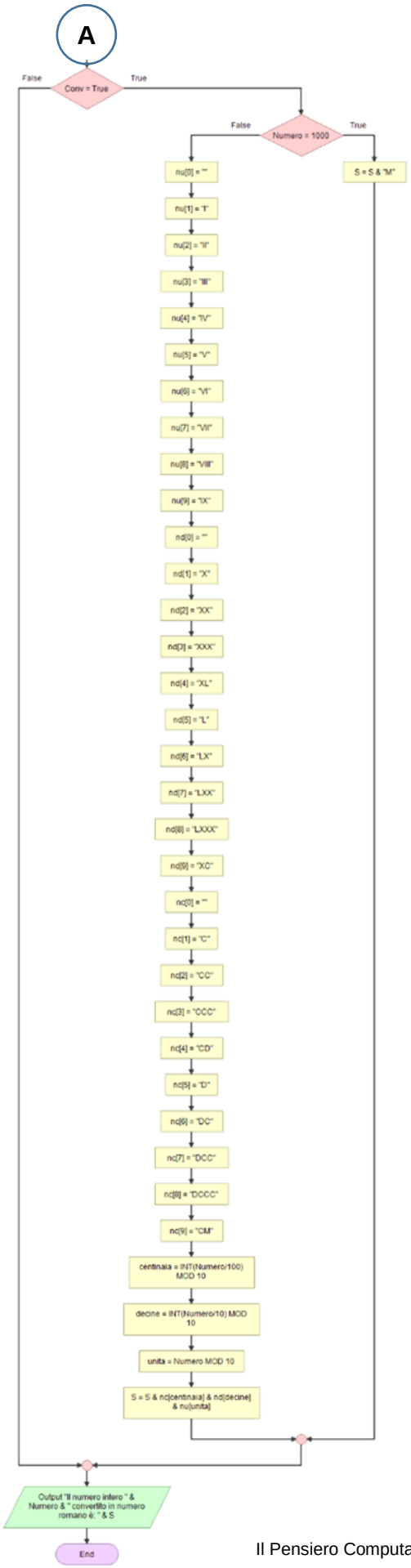
Insert coordinates into the array



210.GA | Conversione di un numero intero in numero romano

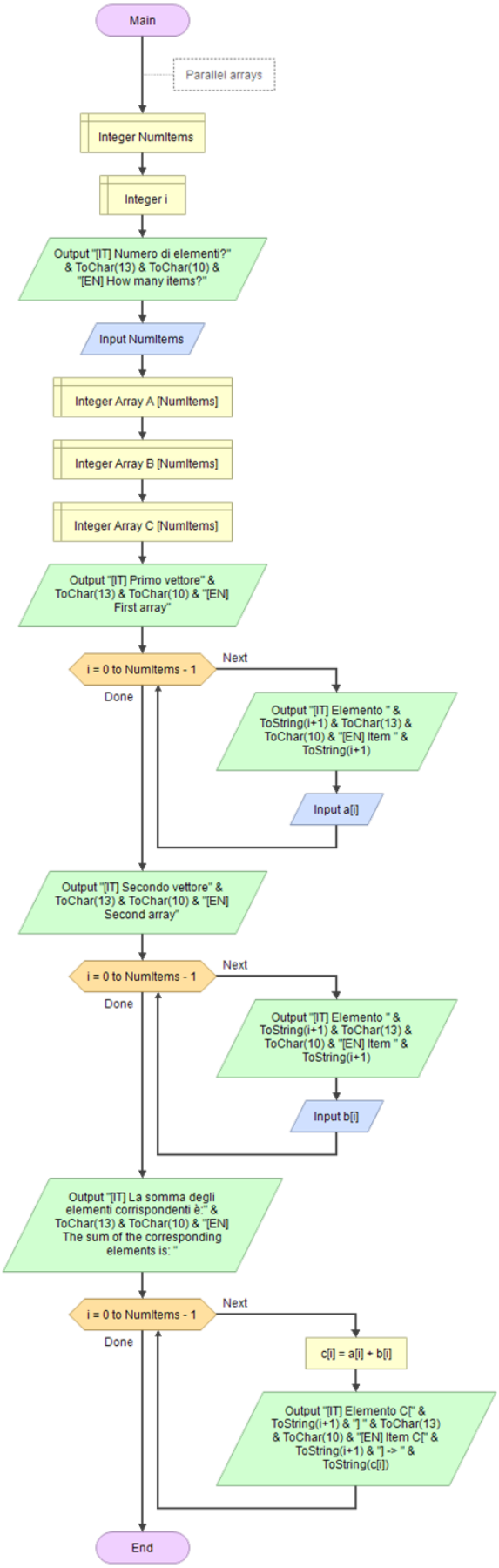
Integer to romanic number





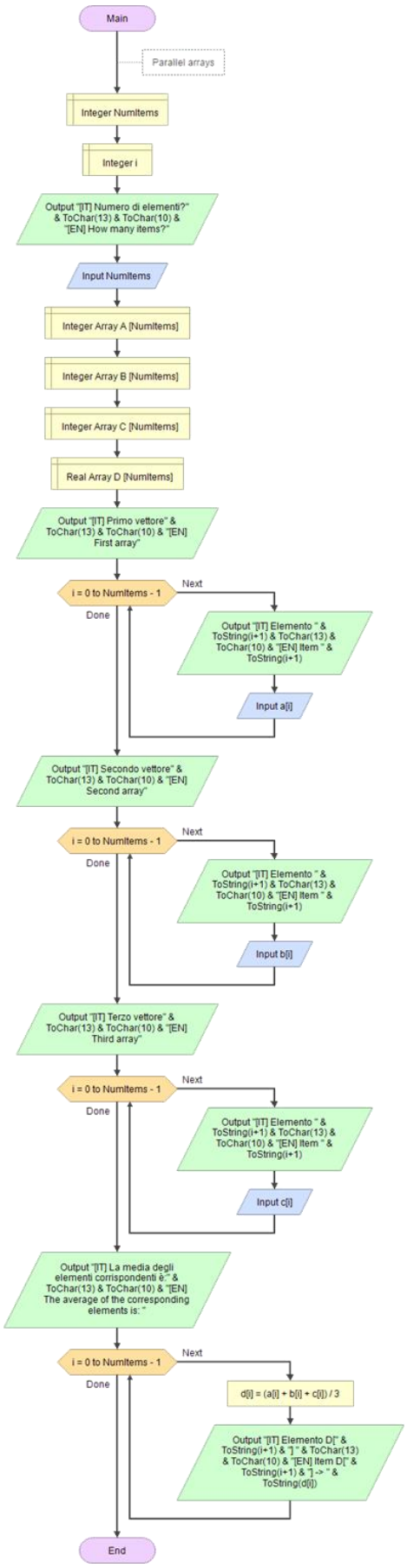
211. **G A** ‖ Letti in input due array di interi con lo stesso numero *N* di elementi, costruire un terzo array di numeri con la somma degli elementi corrispondenti nei primi due (es.: $C_i = (A_i + B_i)$, per $i = 0, 1, \ldots, n-1$). Visualizza l'array *C*. (vettori paralleli)

Sum of the corresponding elements of two parallel arrays



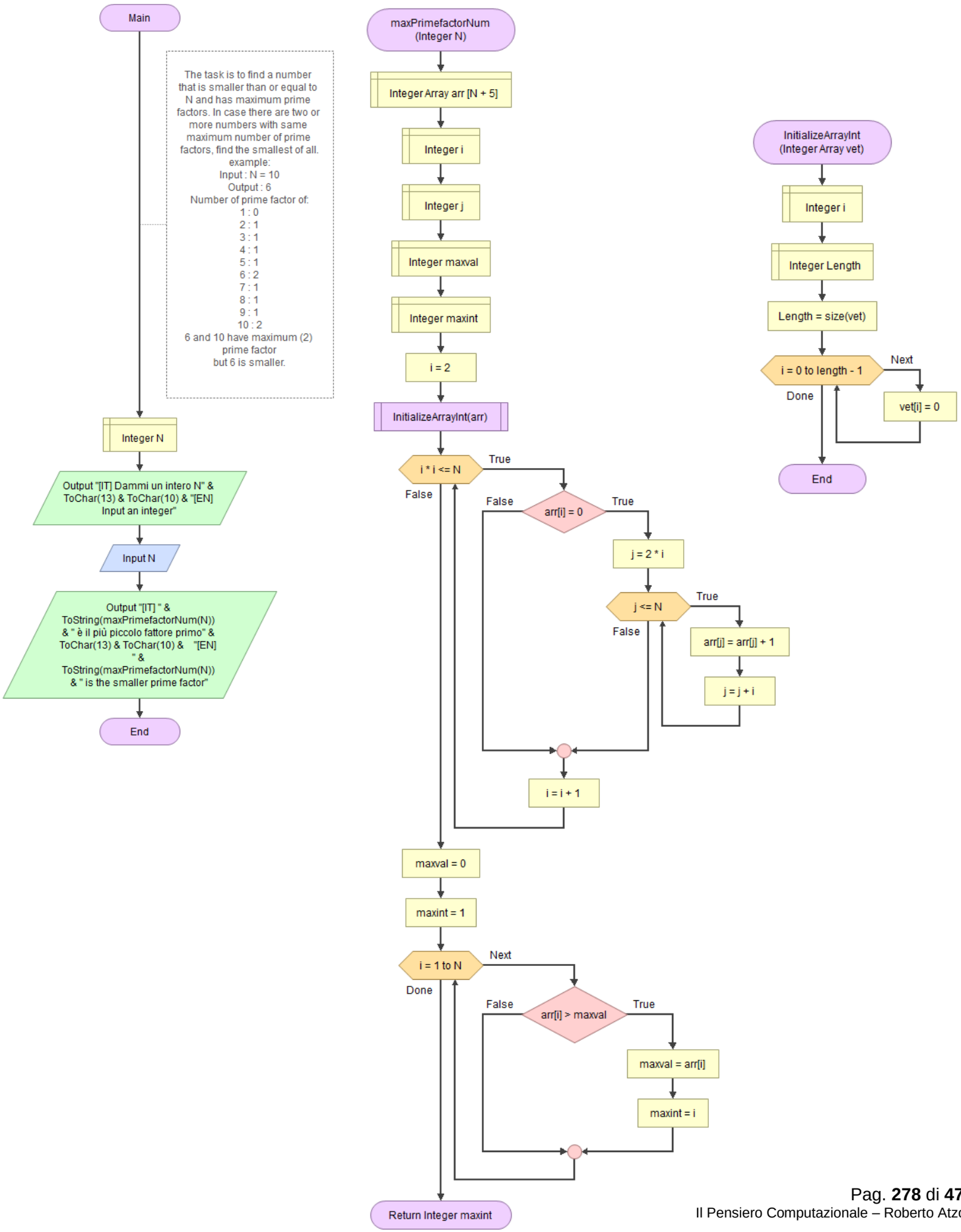
212. **G A** ‖ Letti in input tre array di interi con lo stesso numero *N* di elementi, costruire un quarto array di numeri con la media aritmetica degli elementi corrispondenti nei primi tre (es.: $D_i = (A_i + B_i + C_i)/3$, per $i = 0, 1, \ldots, n-1$). Visualizza l'array *D*. (vettori paralleli)

Average of the corresponding elements of three parallel arrays



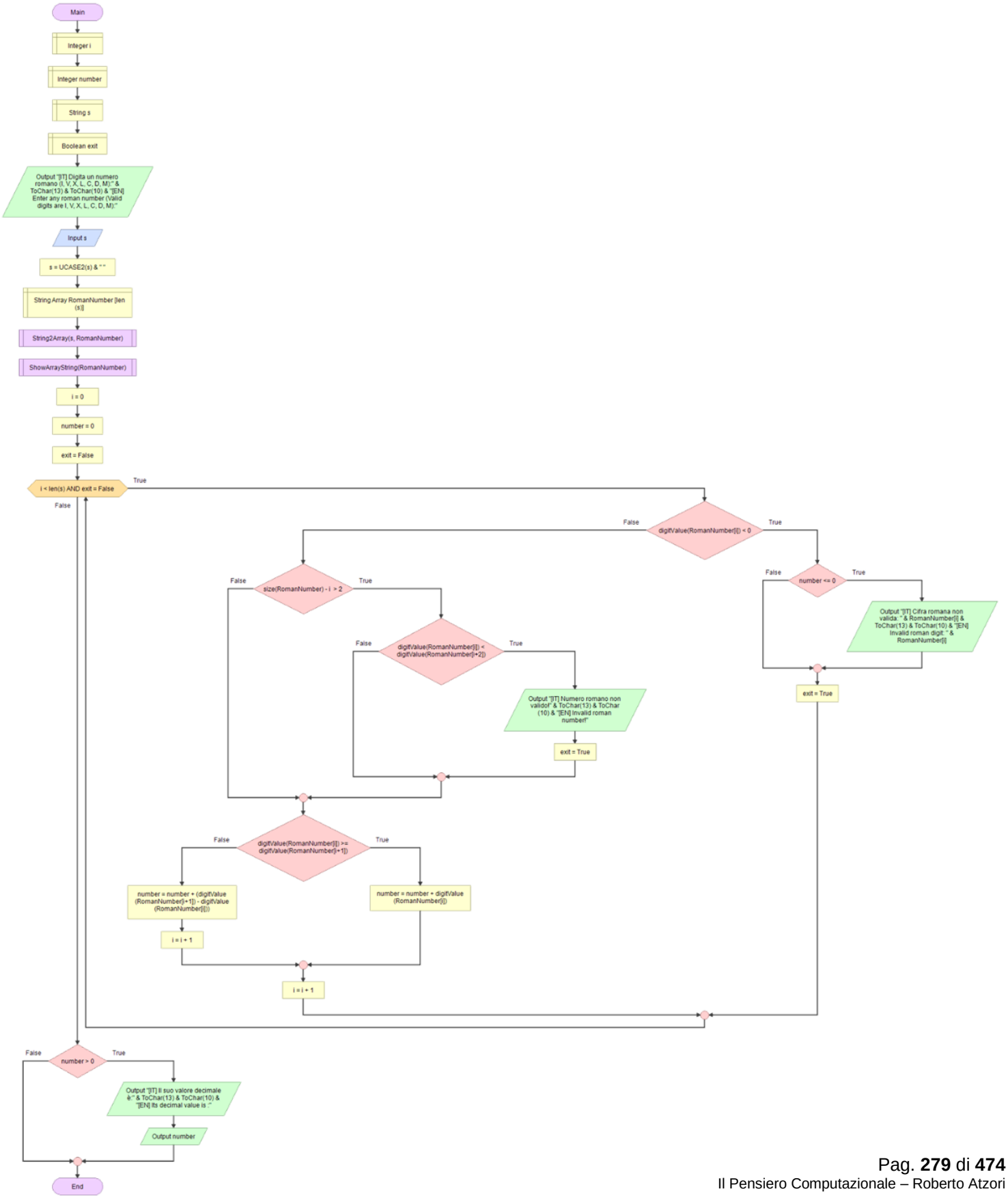
213. G A F ‖ Trovare un numero con un numero massimo di fattori primi

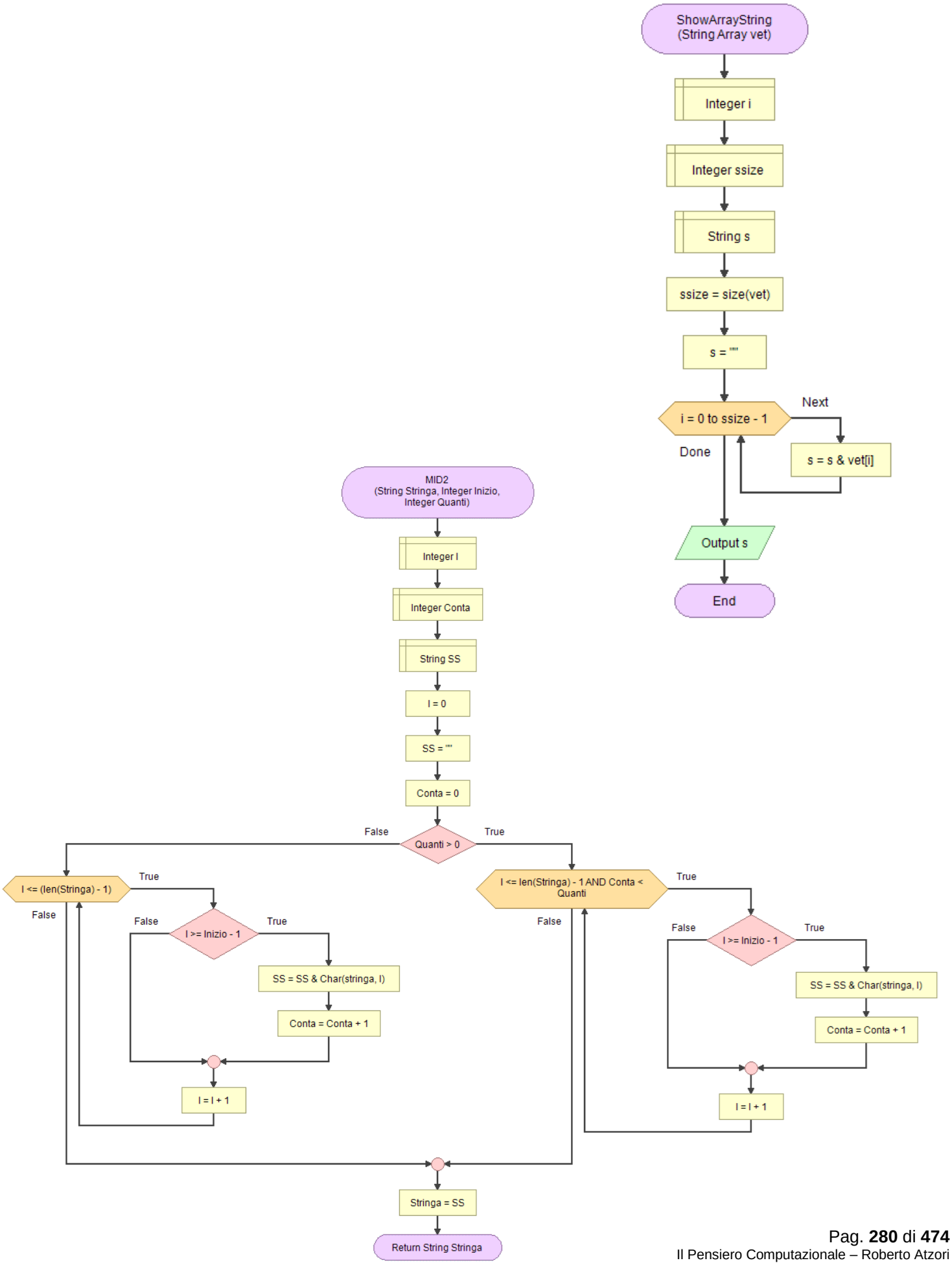
Number with maximum number of prime factors

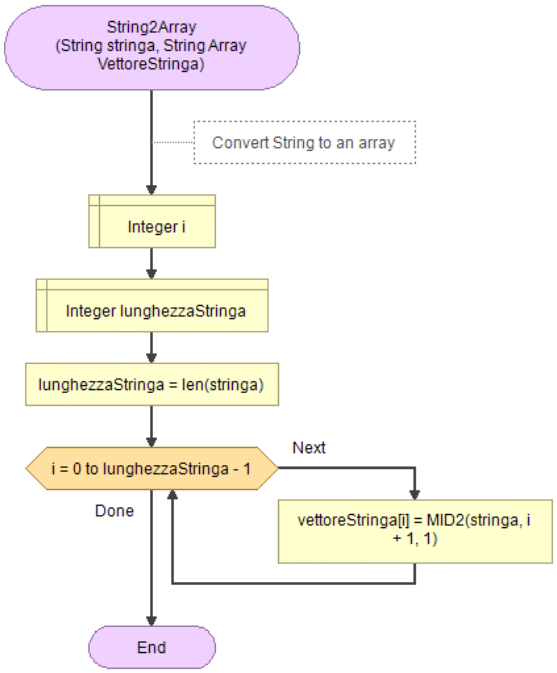


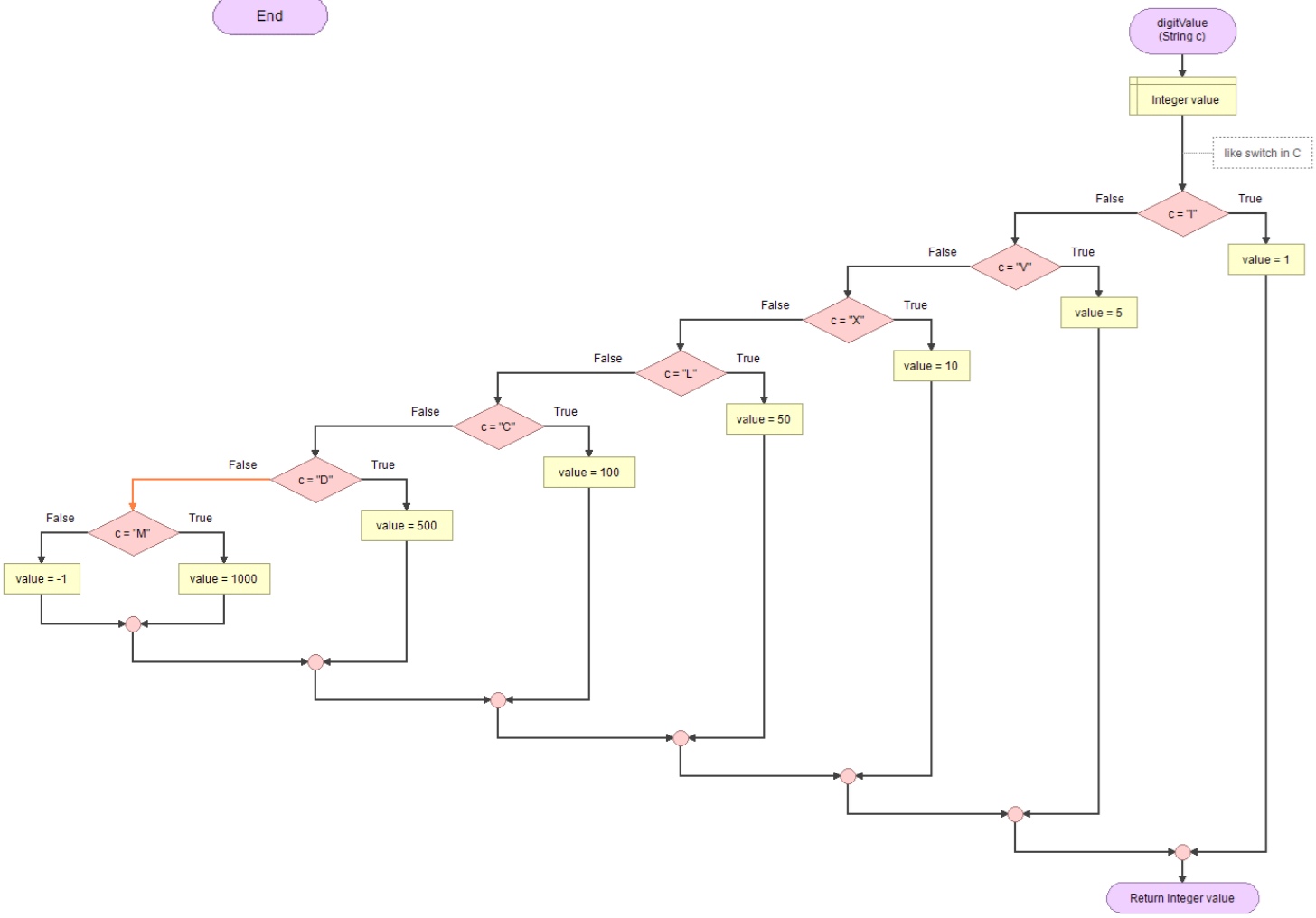
214. G A F ‖ Conversione di un numero romano in intero

Convert from roman numbers to integers



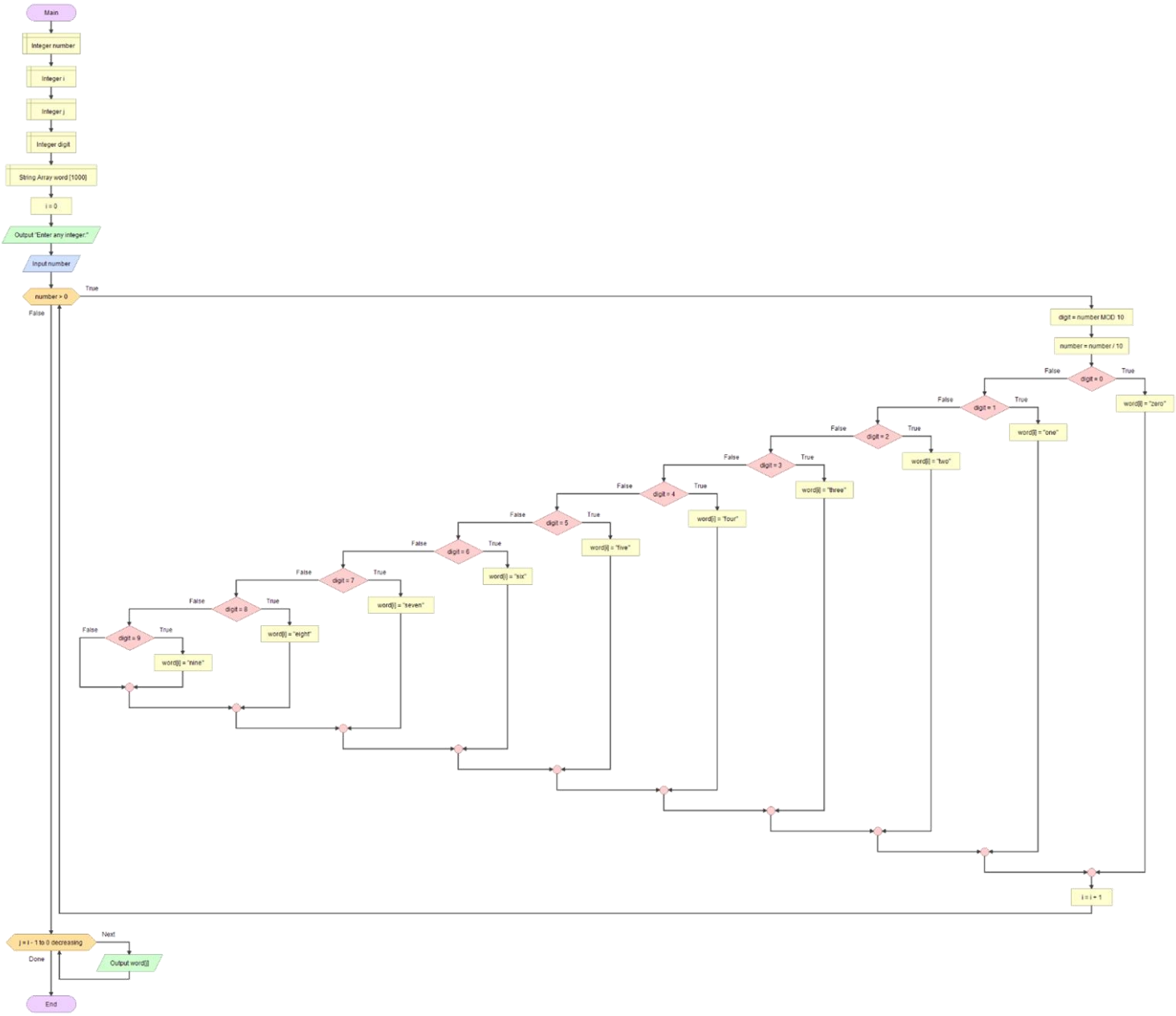

ShowArrayString
(String Array vet)
Integer i
Integer ssize
String s
ssize = size(vet)
s = ""
i = 0 to ssize - 1
Next
Done
s = s & vet[i]
Output s
End
MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa


String2Array
(String stringa, String Array
VettoreStringa)
Convert String to an array
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len(stringa)
i = 0 to lunghezzaStringa - 1
Next
Done
vettoreStringa[i] = MID2(stringa, i
+ 1, 1)
End


digitValue
(String c)
Integer value
like switch in C
c = "I"
False
True
value = 1
c = "V"
value = 5
c = "X"
value = 10
c = "L"
value = 50
c = "C"
value = 100
c = "D"
value = 500
c = "M"
value = -1
value = 1000
Return Integer value

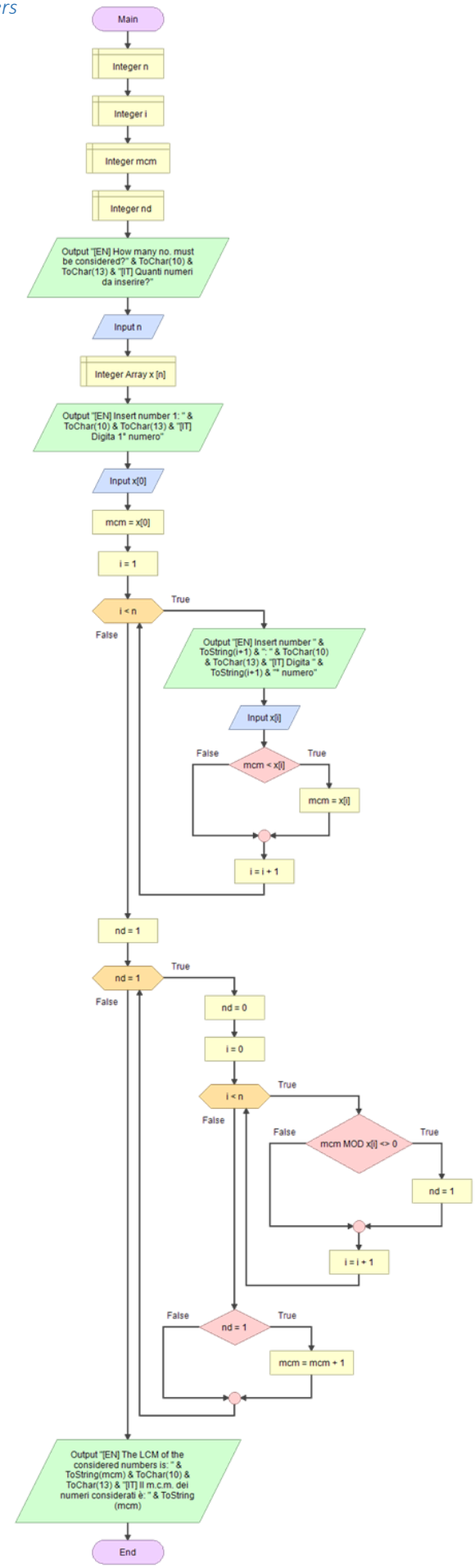
215. G A | Convertire le cifre di un numero intero in inglese

Convert each digits of a number in English word



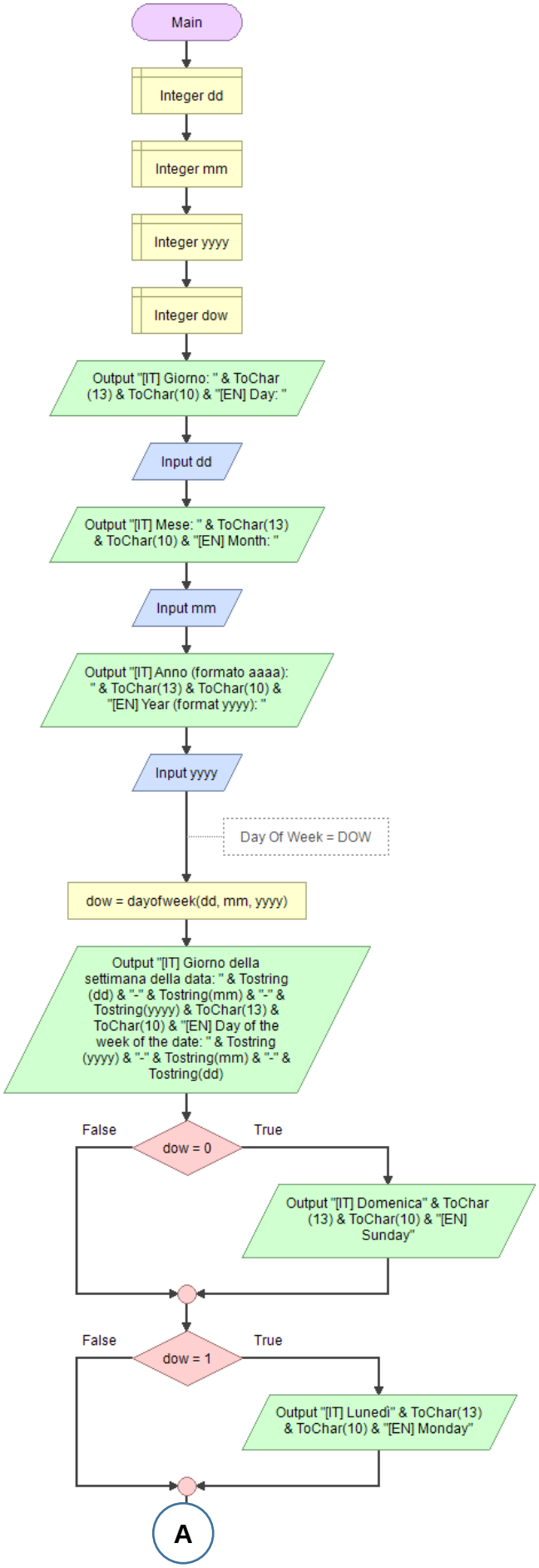
216. **G A** ‖ Acquisire N numeri naturali e visualizzare il minimo comune multiplo dei numeri acquisiti (con array).

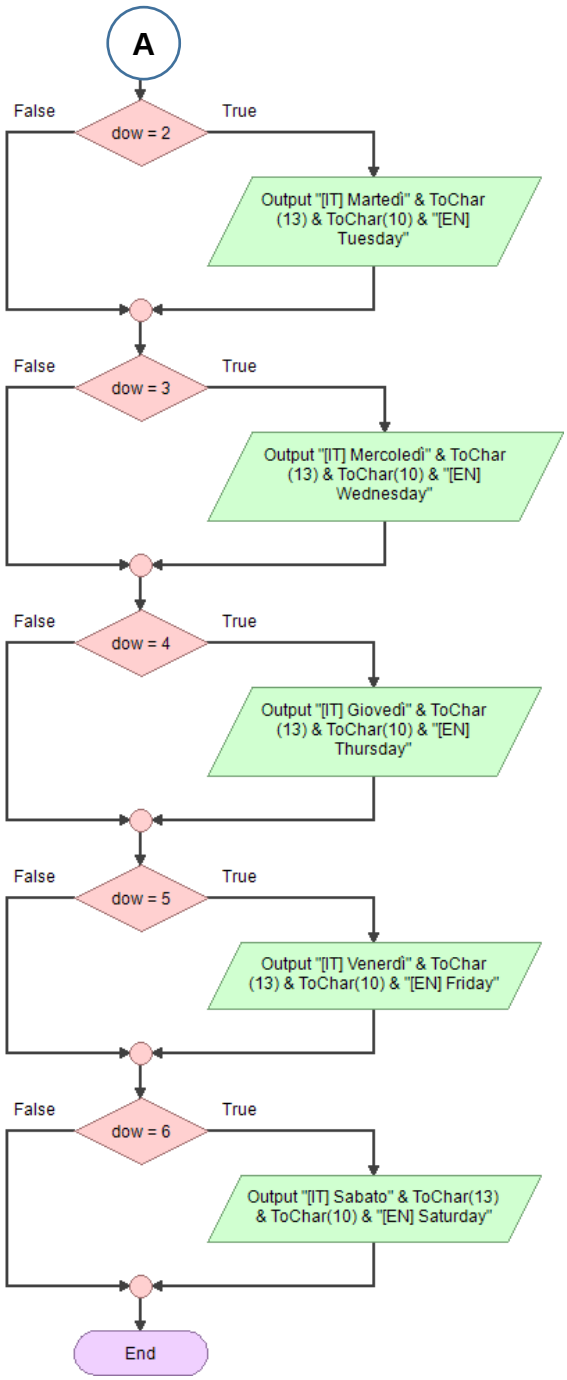
LCM of N entered numbers

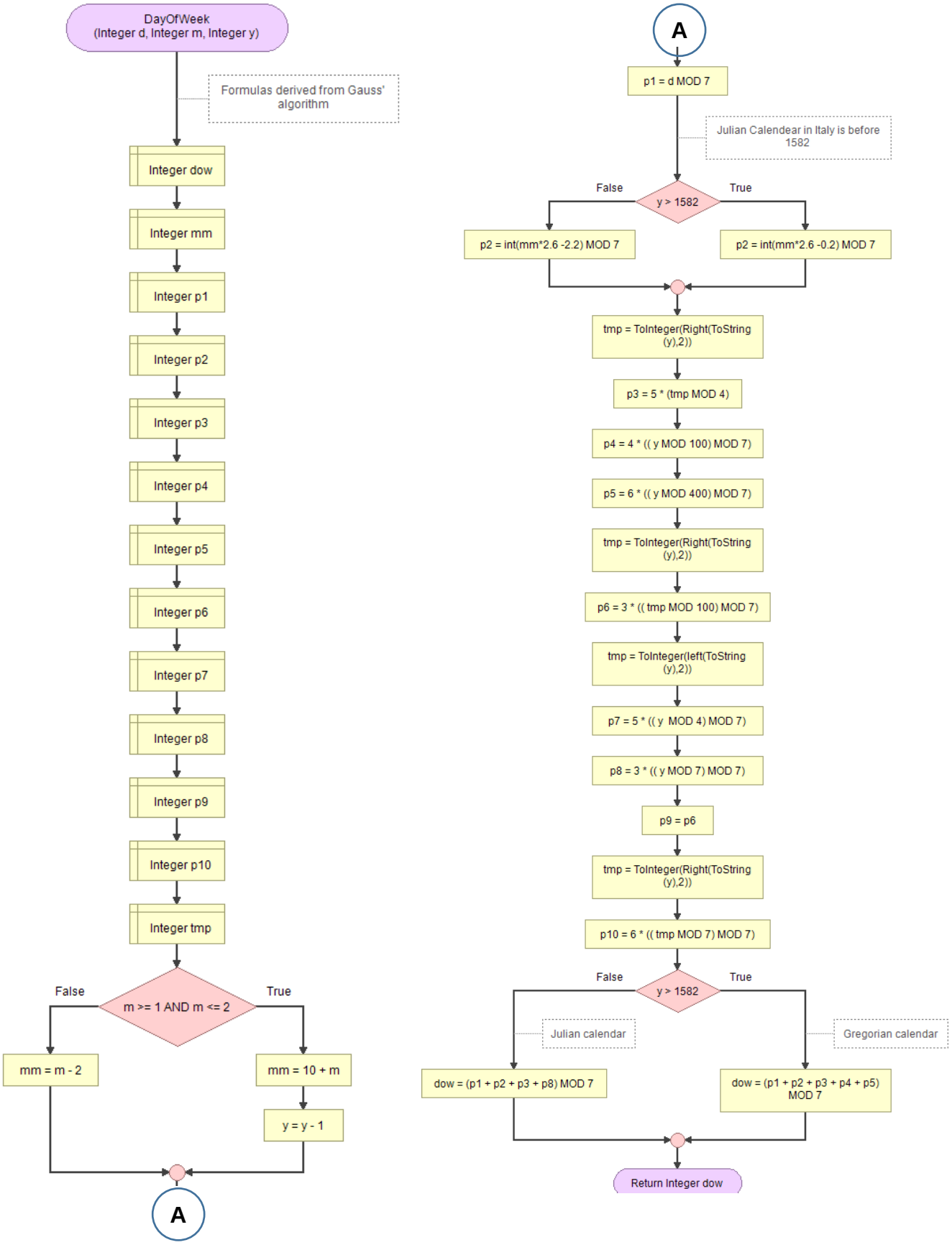


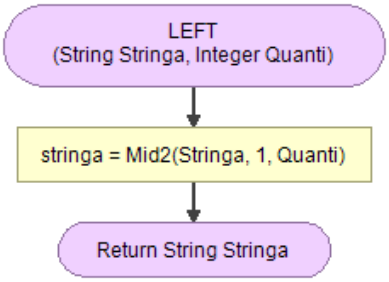
217.GSAF | Calcolo del giorno della settimana di una data

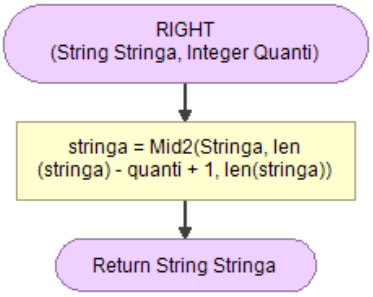
Calc of DoW (Day of the Week)

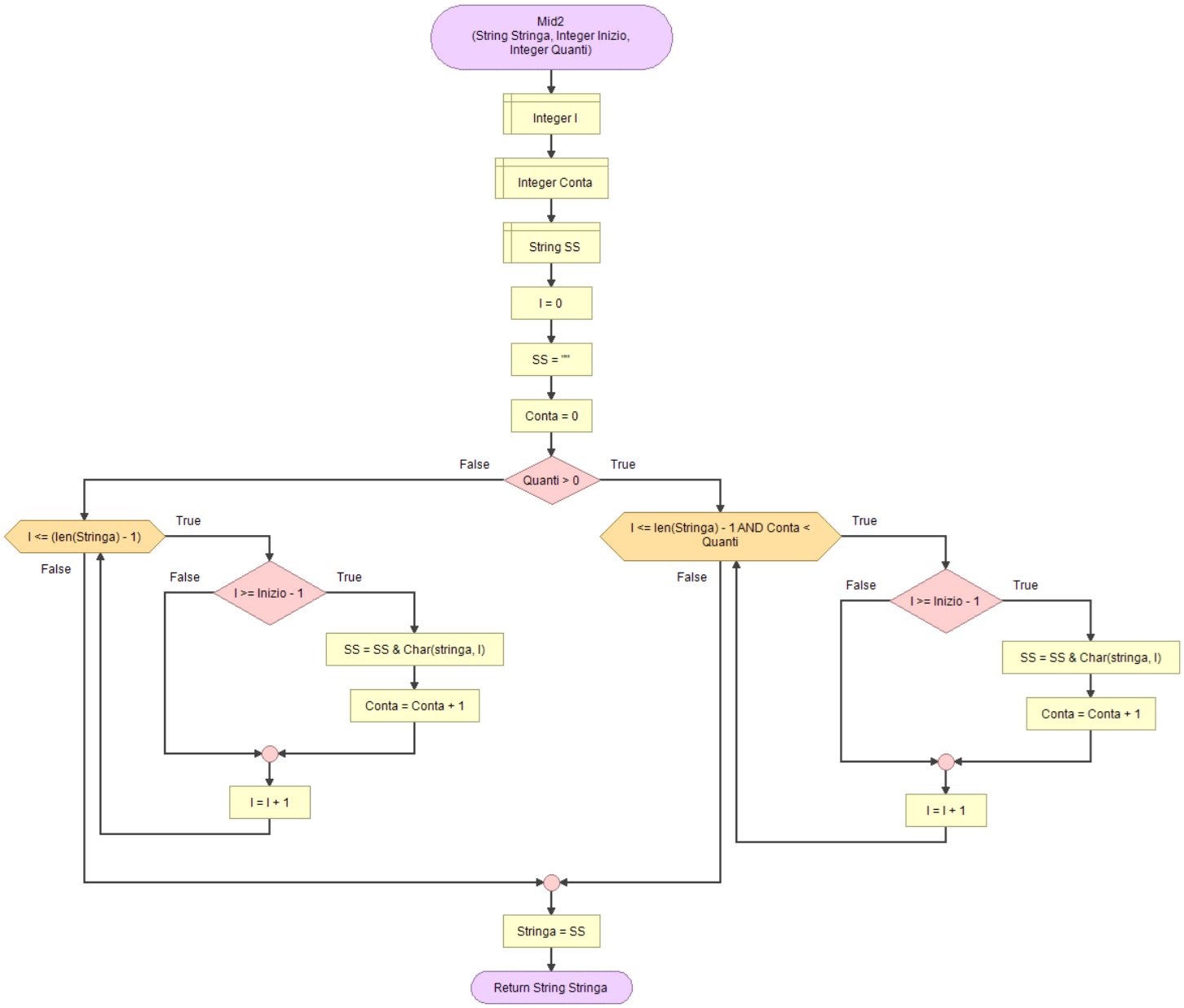





DayOfWeek
(Integer d, Integer m, Integer y)
Formulas derived from Gauss' algorithm
Integer dow
Integer mm
Integer p1
Integer p2
Integer p3
Integer p4
Integer p5
Integer p6
Integer p7
Integer p8
Integer p9
Integer p10
Integer tmp
m >= 1 AND m <= 2
False
True
mm = m - 2
mm = 10 + m
y = y - 1
A
A
p1 = d MOD 7
Julian Calendear in Italy is before 1582
y > 1582
False
True
p2 = int(mm*2.6 -2.2) MOD 7
p2 = int(mm*2.6 -0.2) MOD 7
tmp = ToInteger(Right(ToString (y),2))
p3 = 5 * (tmp MOD 4)
p4 = 4 * ((y MOD 100) MOD 7)
p5 = 6 * ((y MOD 400) MOD 7)
tmp = ToInteger(Right(ToString (y),2))
p6 = 3 * ((tmp MOD 100) MOD 7)
tmp = ToInteger(left(ToString (y),2))
p7 = 5 * ((y MOD 4) MOD 7)
p8 = 3 * ((y MOD 7) MOD 7)
p9 = p6
tmp = ToInteger(Right(ToString (y),2))
p10 = 6 * ((tmp MOD 7) MOD 7)
y > 1582
False
True
Julian calendar
Gregorian calendar
dow = (p1 + p2 + p3 + p8) MOD 7
dow = (p1 + p2 + p3 + p4 + p5) MOD 7
Return Integer dow

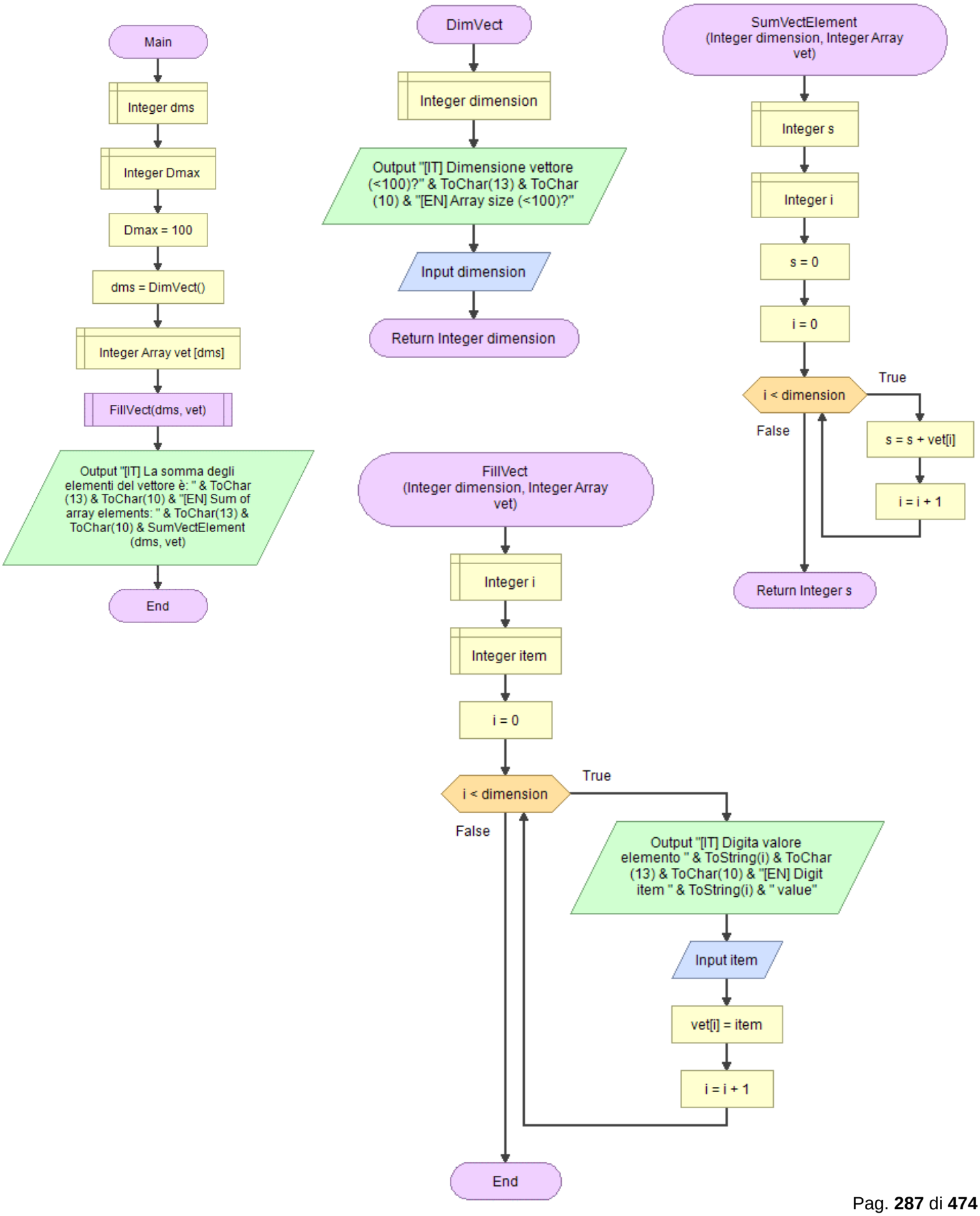

LEFT
(String Stringa, Integer Quanti)
stringa = Mid2(Stringa, 1, Quanti)
Return String Stringa


RIGHT
(String Stringa, Integer Quanti)
stringa = Mid2(Stringa, len (stringa) - quanti + 1, len(stringa))
Return String Stringa


Mid2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

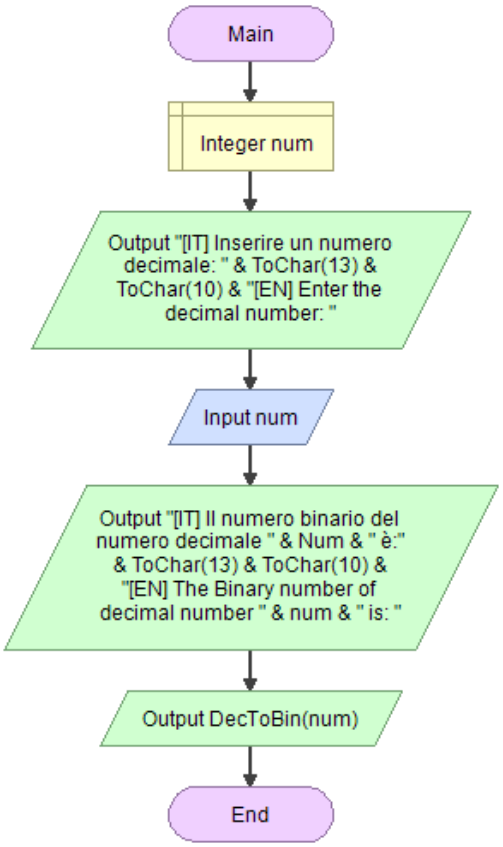
218.**G A F** ‖ Calcolare la somma degli elementi di un vettore

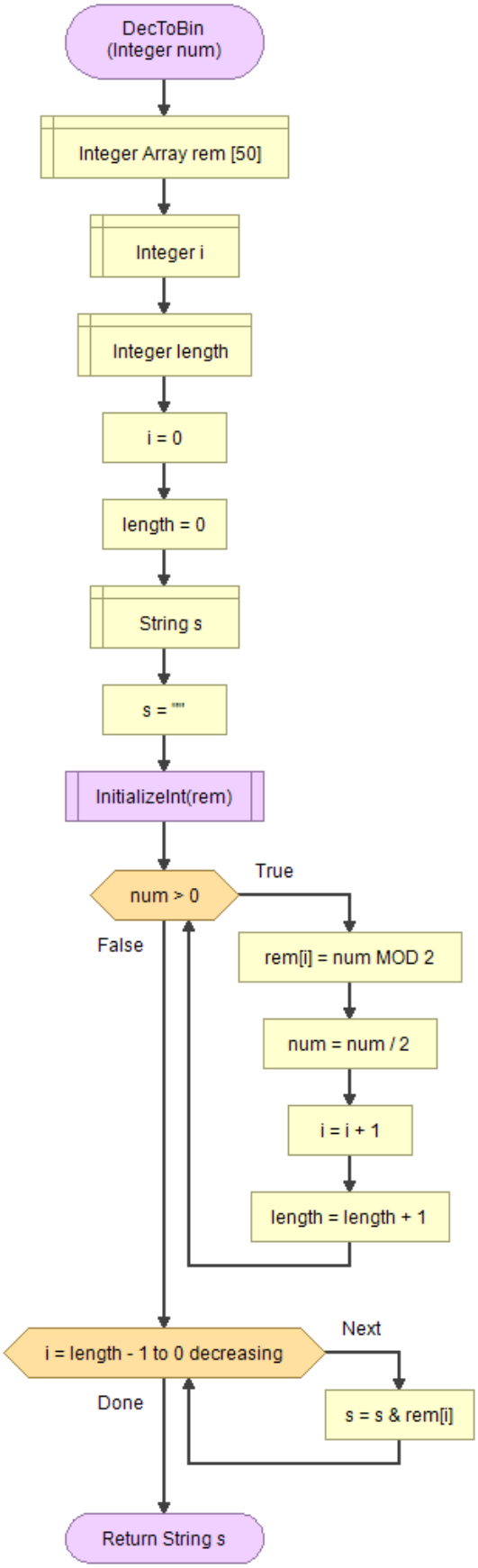
Calculate the sum of the elements of a array

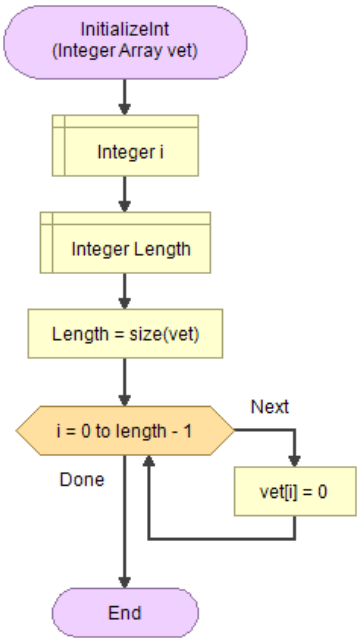


219.GAF ‖ Conversione decimale binario

Convert Decimal Number to Binary Number

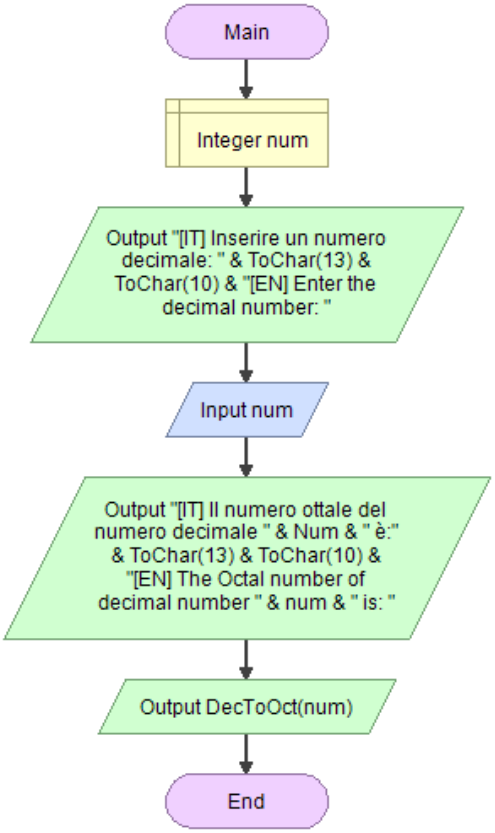


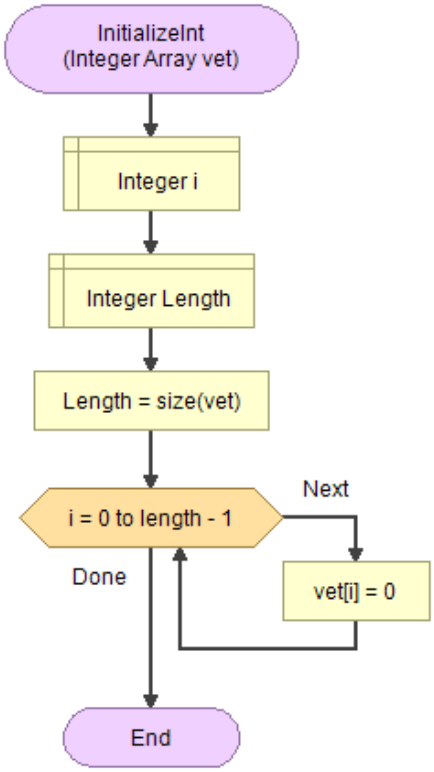


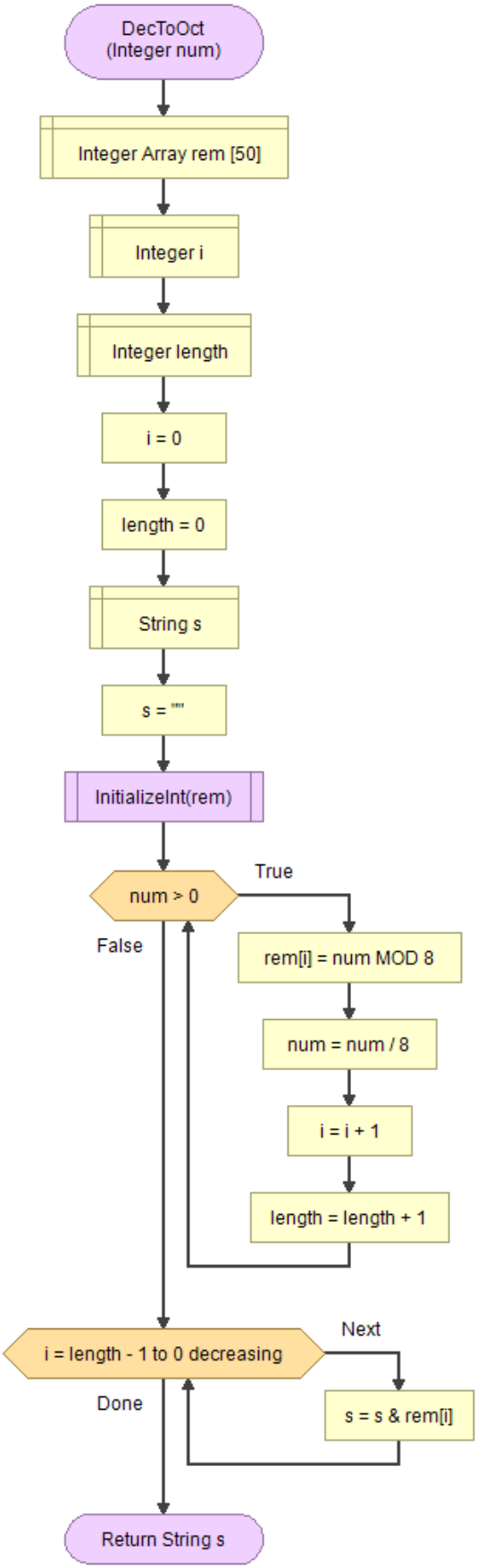


220.GAF ‖ Conversione decimale ottale

Convert Decimal Number to Octal Number

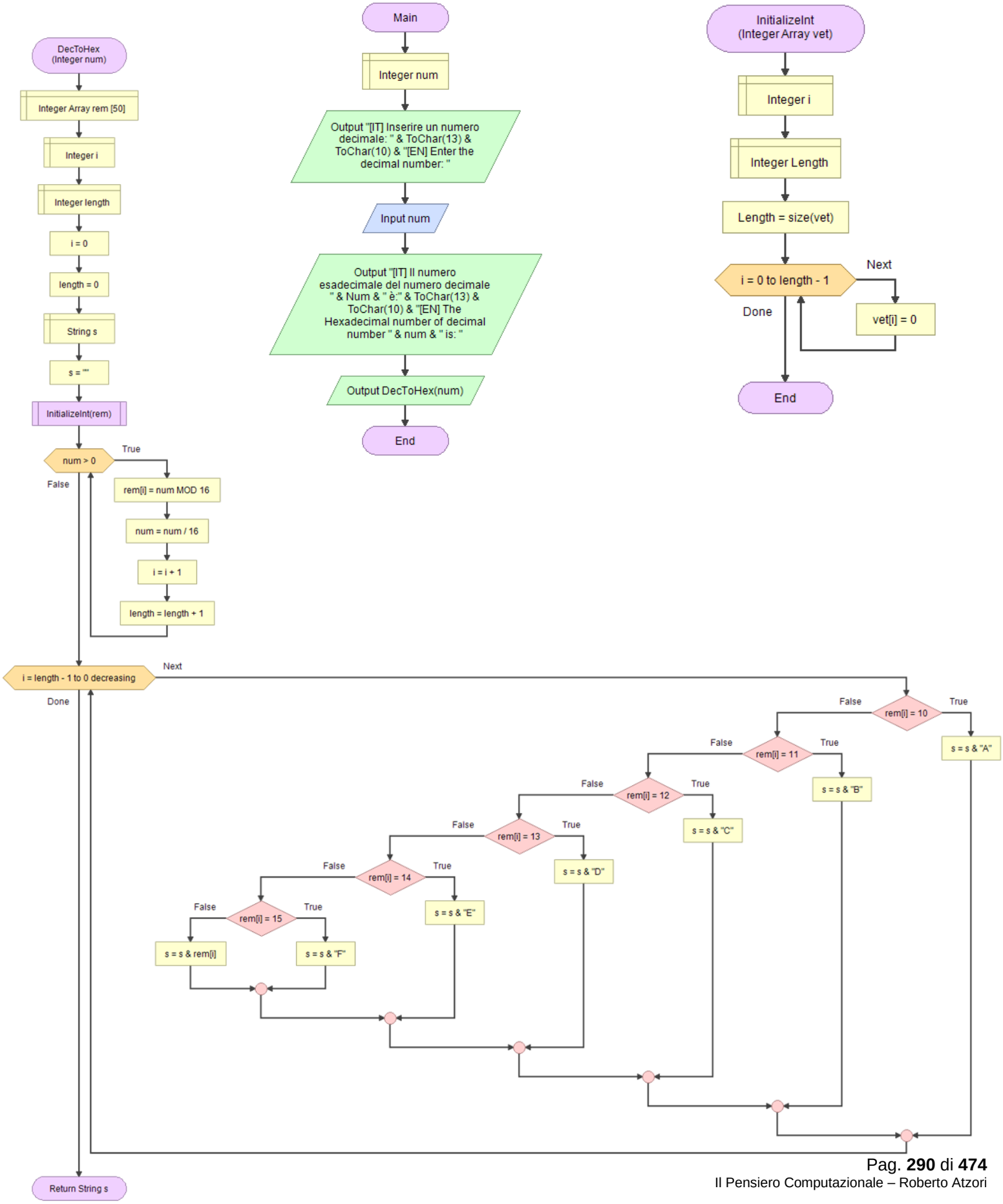






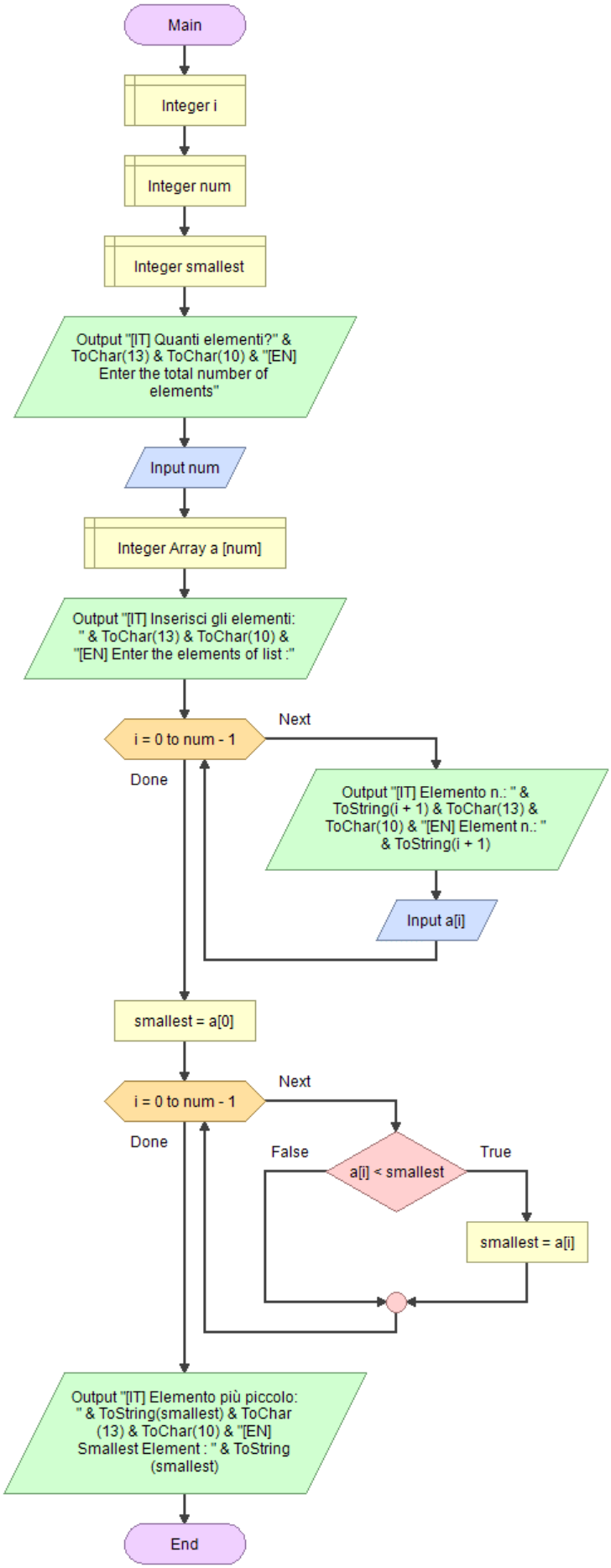
221.GAF ‖ Conversione decimale esadecimale

Convert Decimal Number to Hexadecimal Number



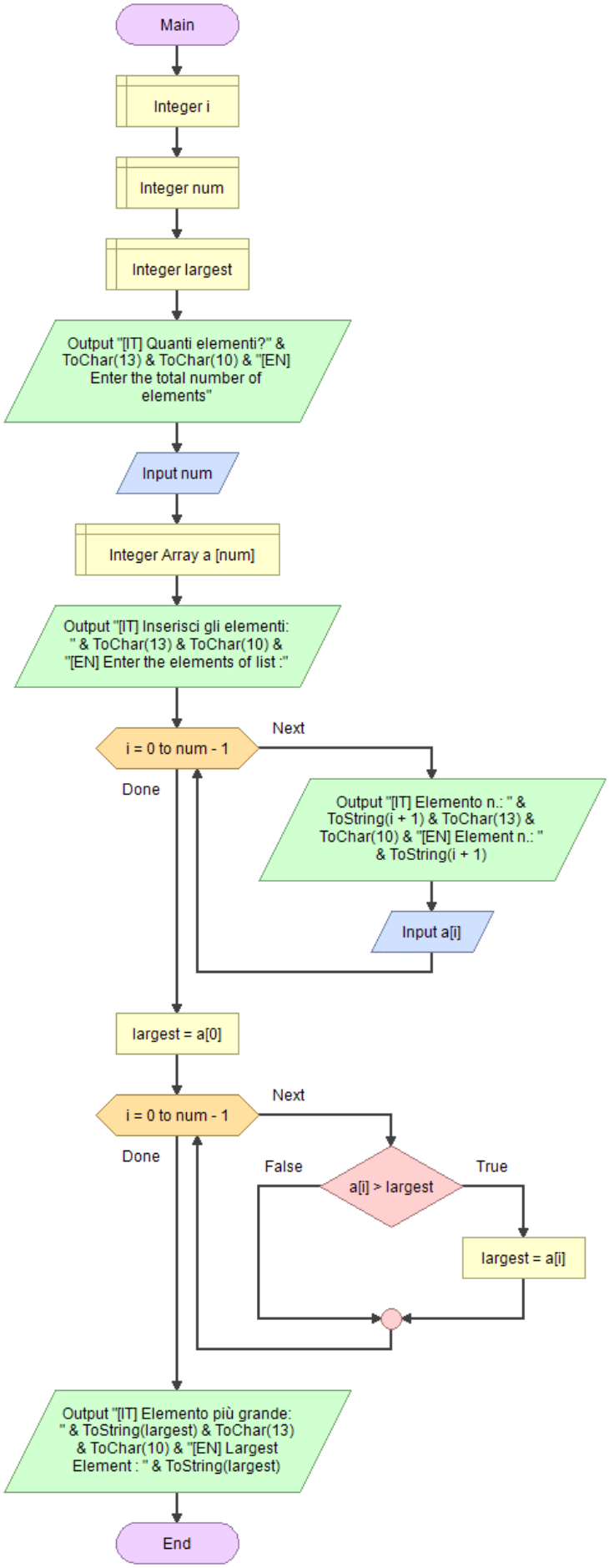
222.GA | Ricercare l'elemento minore in un vettore

Find Smallest Element in Array



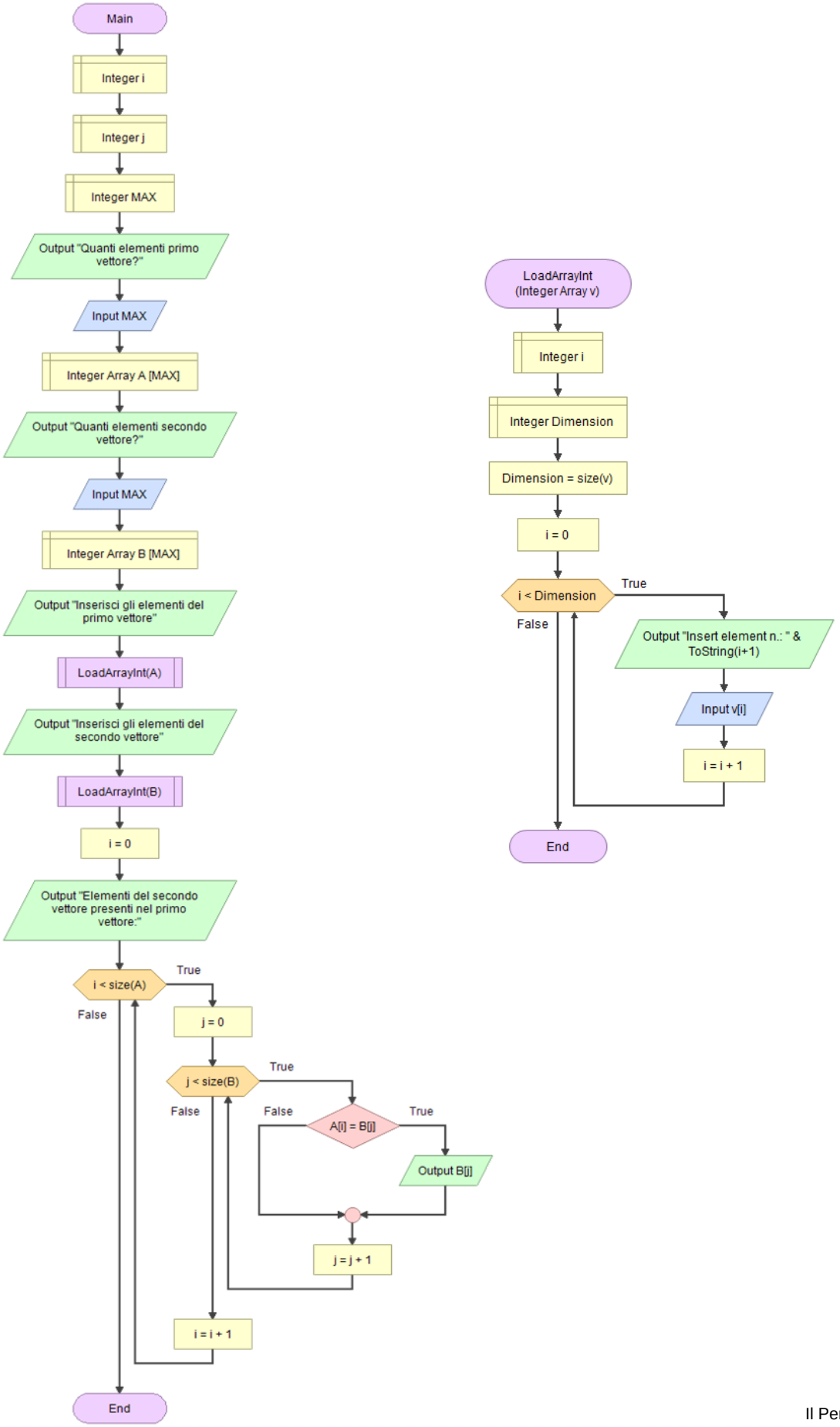
223.GA | Ricercare l'elemento maggiore di un vettore

Find Largest Element in Array

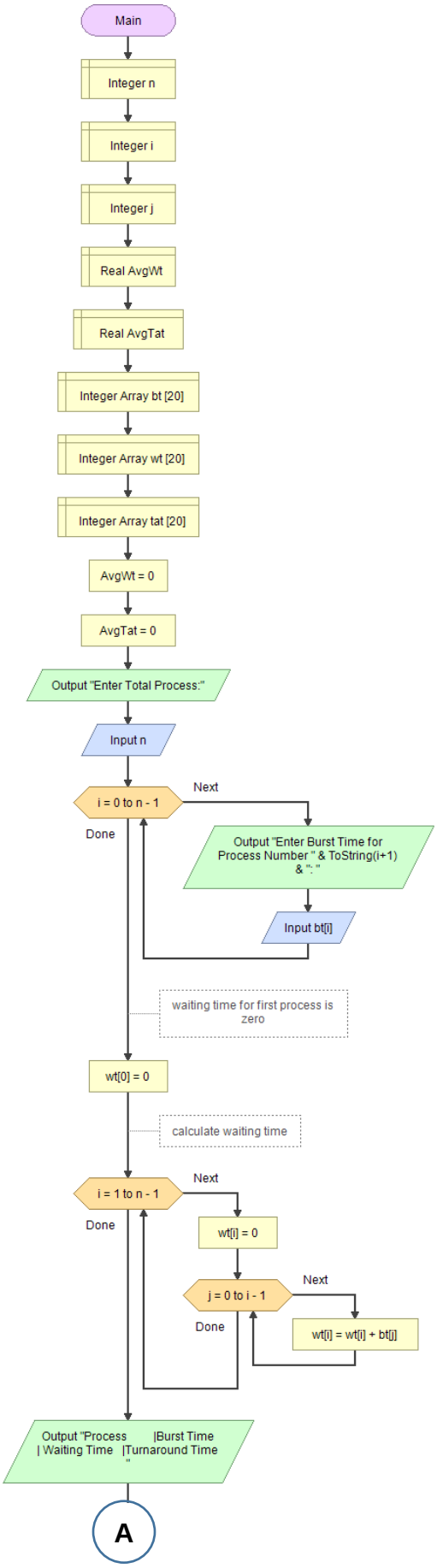


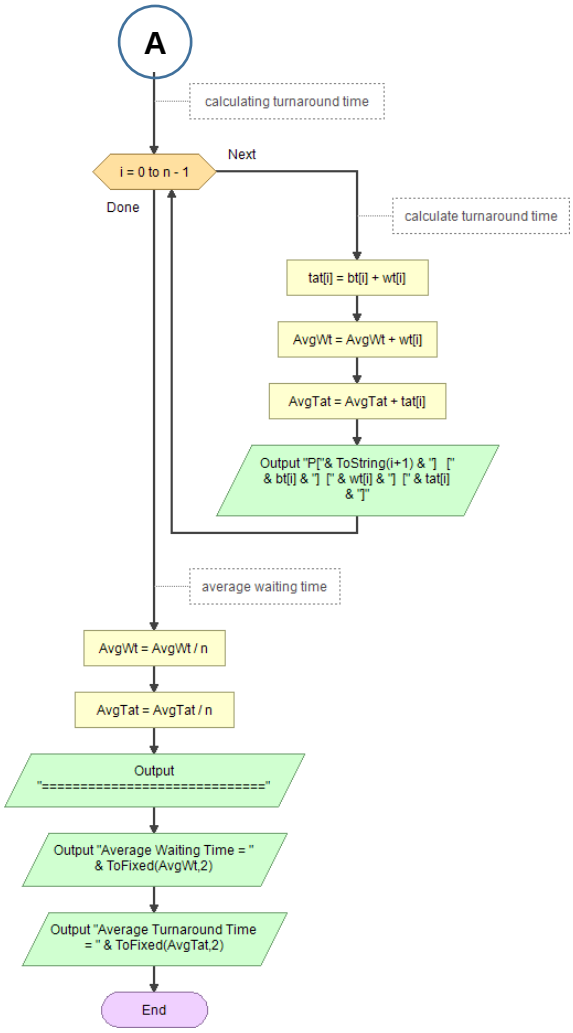
224. **G A** | Visualizza gli elementi del secondo vettore comuni a quelli del primo vettore

Print elements of second array commons in the first array

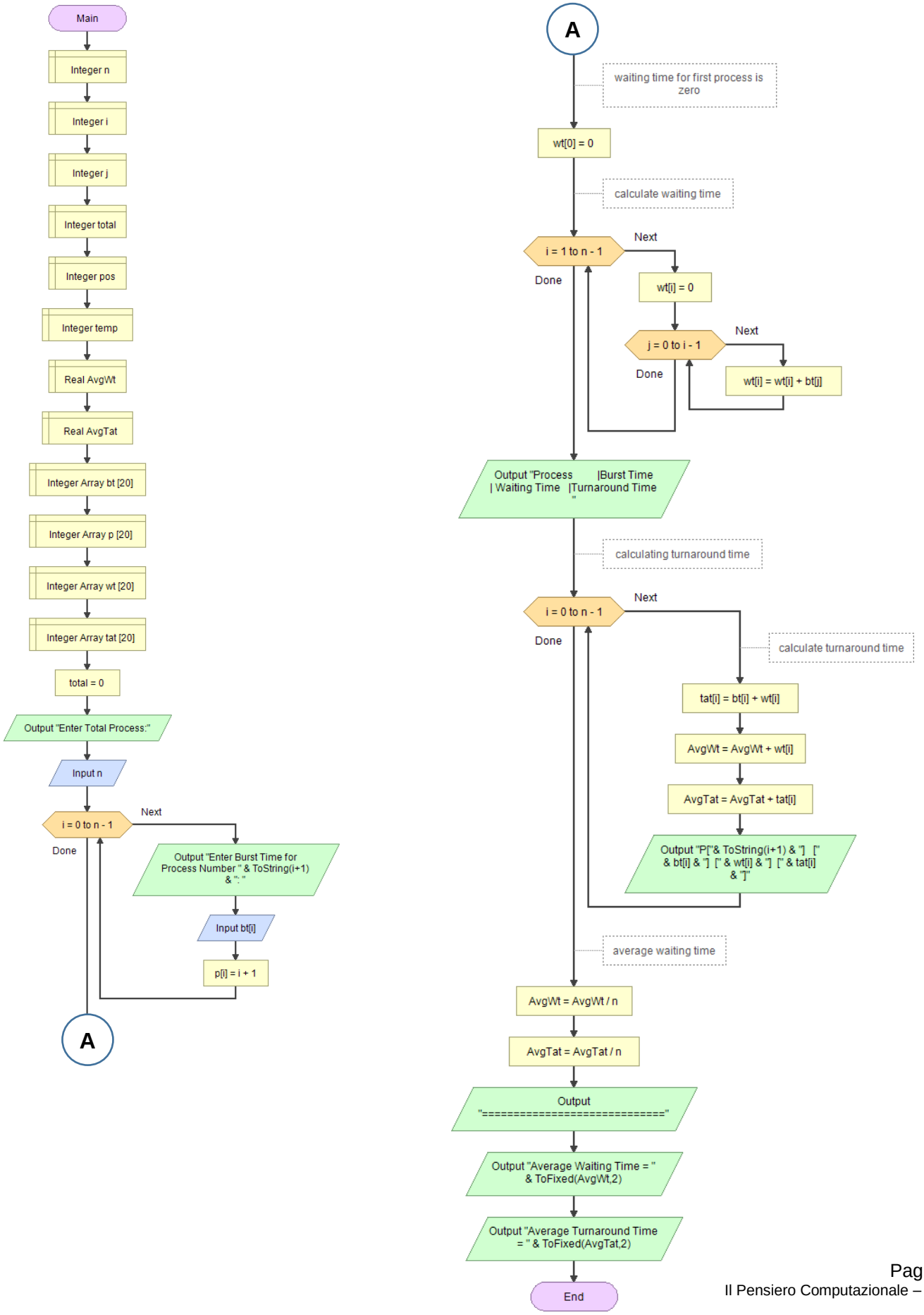


Process Scheduling - First Come First Served (FCFS)

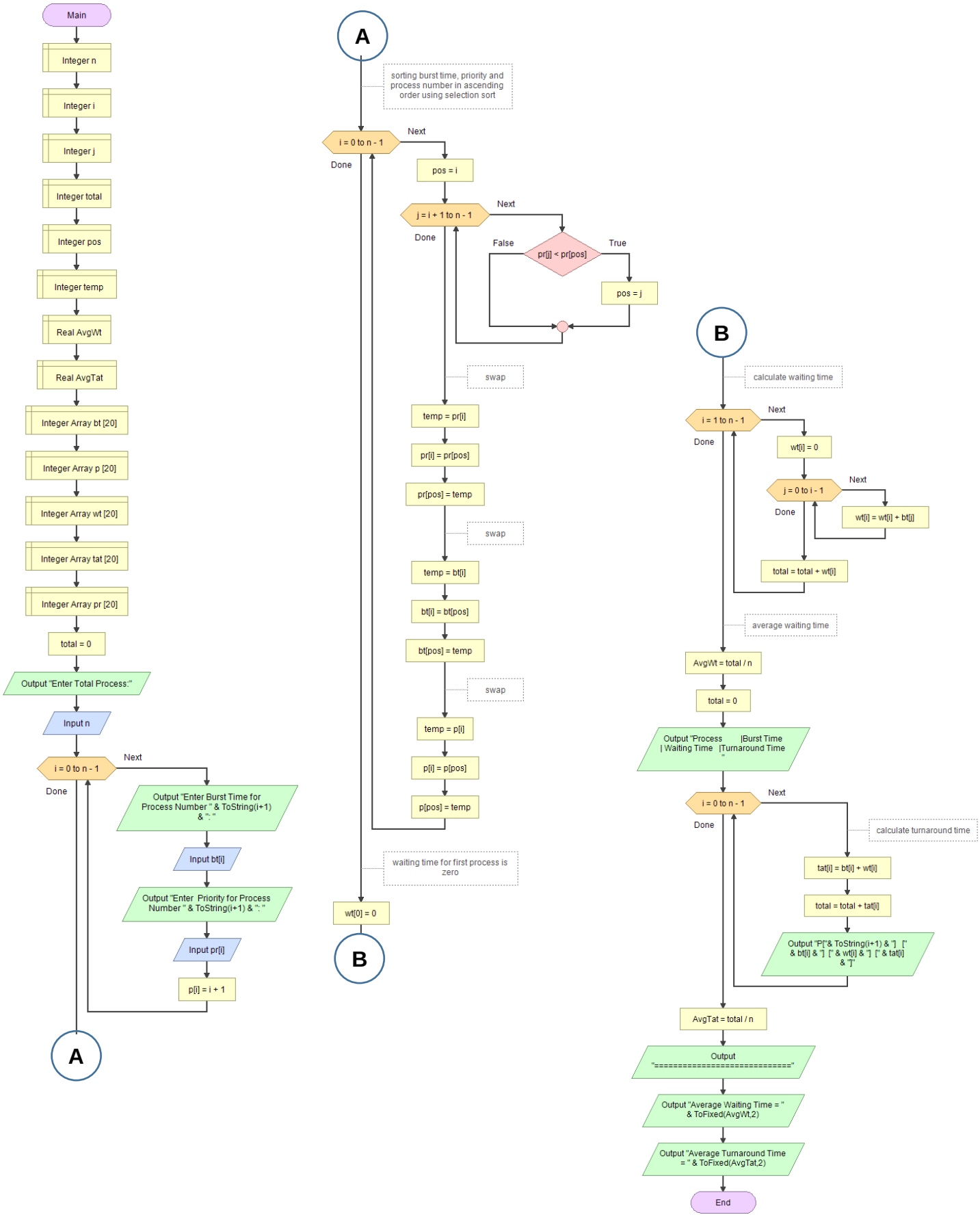




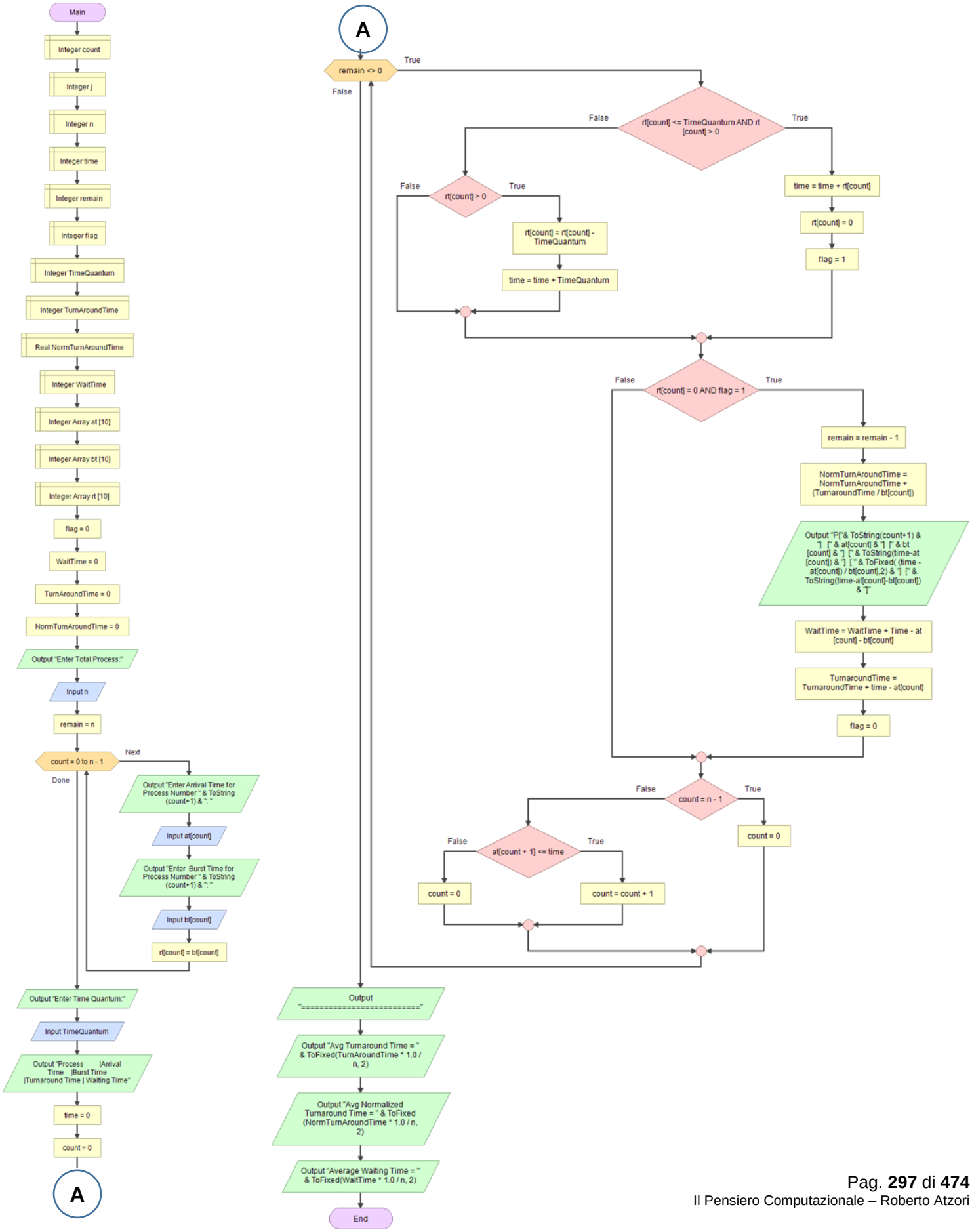
Process Scheduling - Shortest Job First (SJF)



Process Scheduling - Priority



Process Scheduling - Round Robin



Process Scheduling – Next Fit

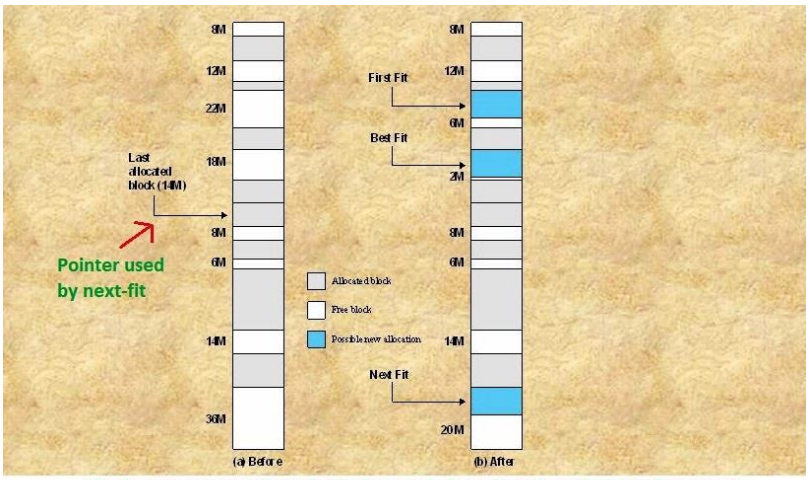
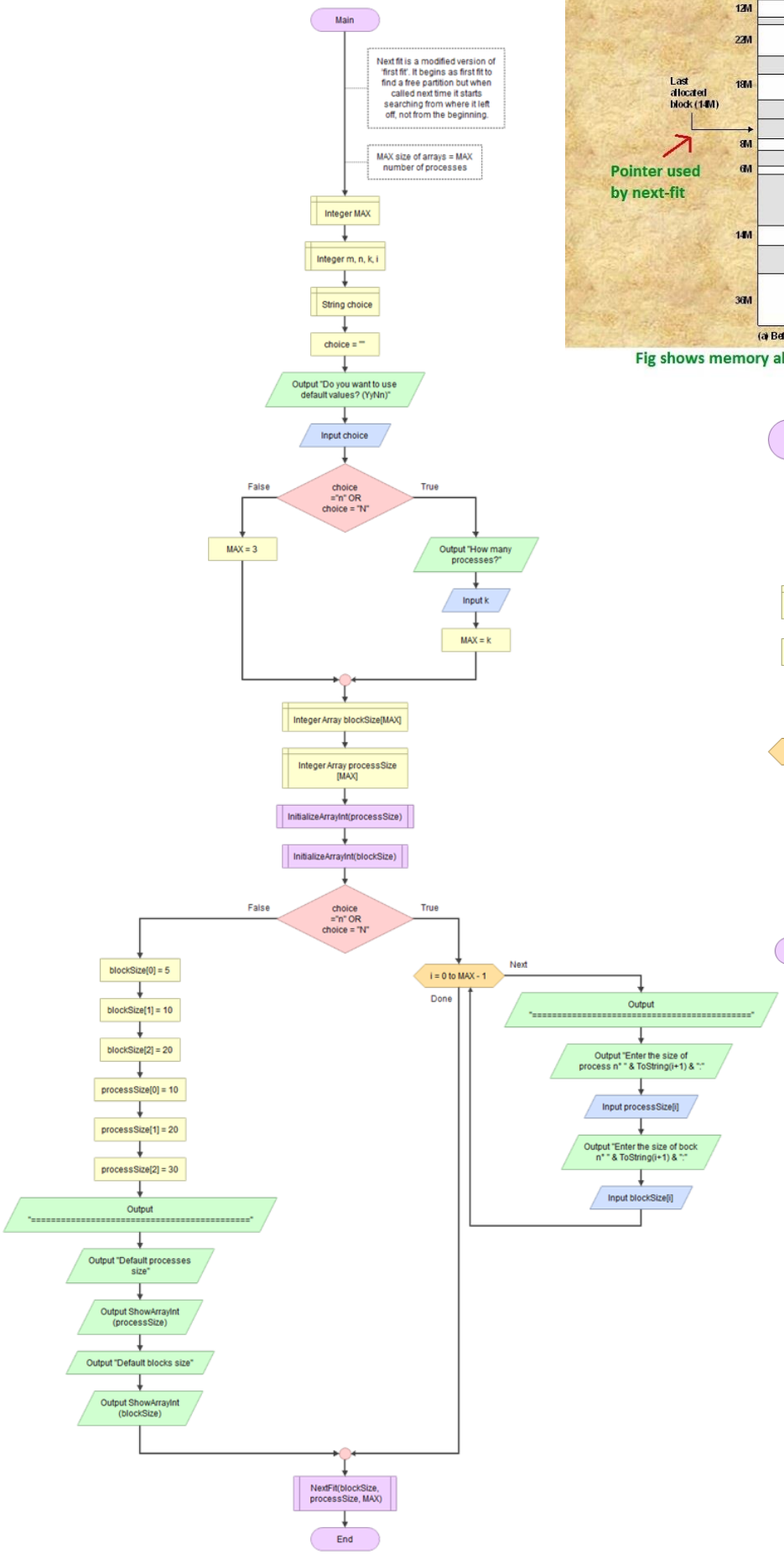
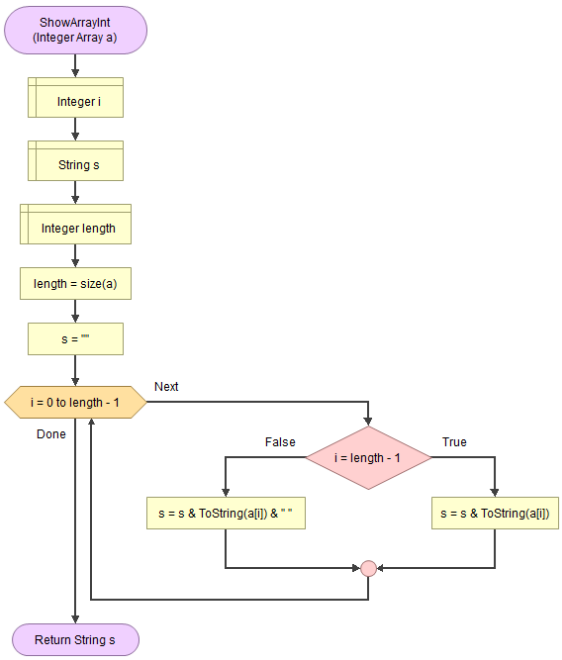
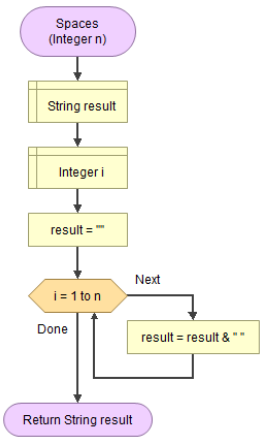
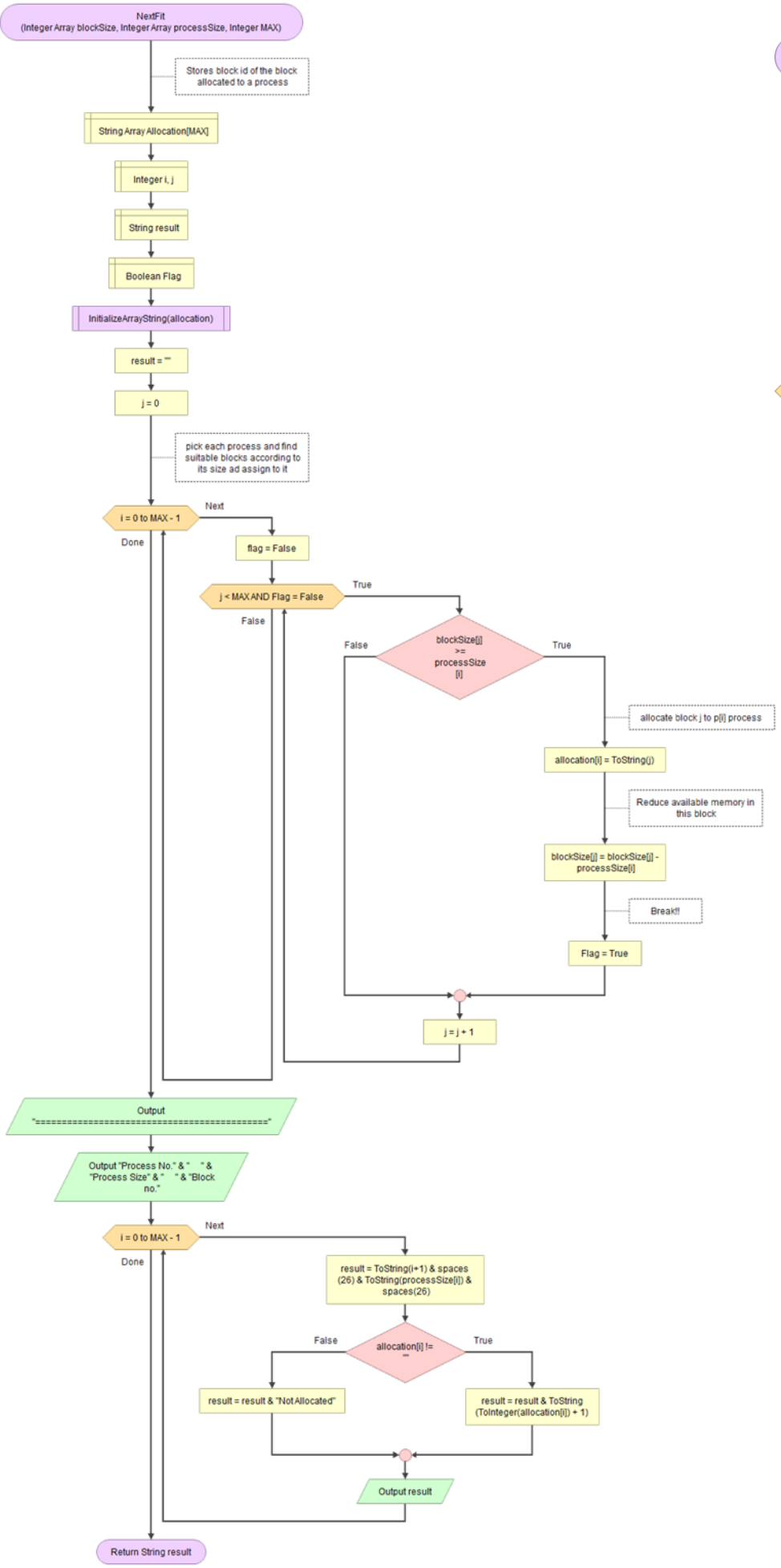


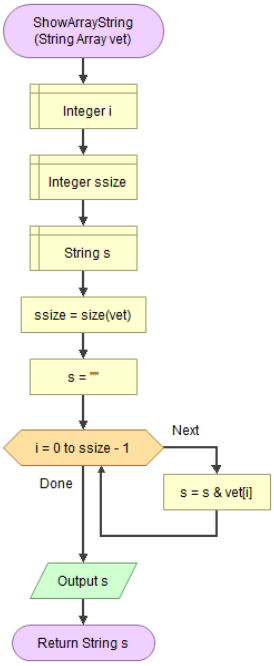
Fig shows memory allocation before and after allocation of 16 M of memory

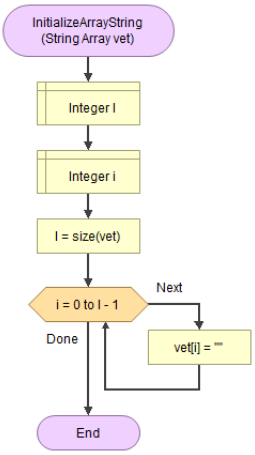


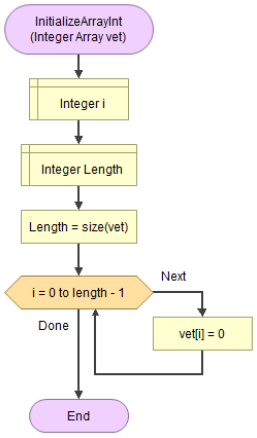





NextFit
(Integer Array blockSize, Integer Array processSize, Integer MAX)
Stores block id of the block allocated to a process
String Array Allocation[MAX]
Integer i, j
String result
Boolean Flag
InitializeArrayString(allocation)
result = ""
j = 0
pick each process and find suitable blocks according to its size ad assign to it
i = 0 to MAX - 1
Next
Done
flag = False
j < MAX AND Flag = False
True
False
blockSize[j] >= processSize [i]
False
True
allocate block j to p[i] process
allocation[i] = ToString(j)
Reduce available memory in this block
blockSize[j] = blockSize[j] - processSize[i]
Break!!
Flag = True
j = j + 1
Output
"==========="
Output "Process No." & " " & "Process Size" & " " & "Block no."
i = 0 to MAX - 1
Next
Done
result = ToString(i+1) & spaces (26) & ToString(processSize[i]) & spaces(26)
allocation[i] != ""
False
True
result = result & "Not Allocated"
result = result & ToString (ToInteger(allocation[i]) + 1)
Output result
Return String result

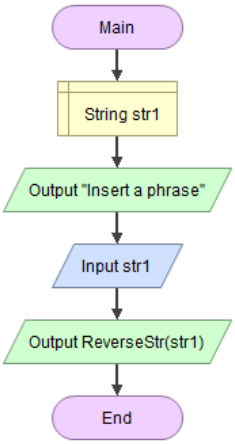

ShowArrayString
(String Array vet)
Integer i
Integer ssize
String s
ssize = size(vet)
s = ""
i = 0 to ssize - 1
Next
Done
s = s & vet[i]
Output s
Return String s

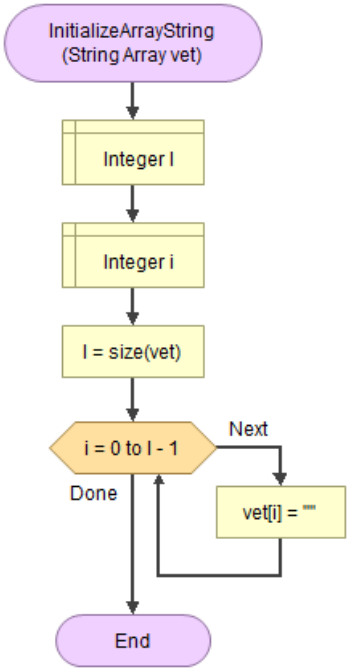

InitializeArrayString
(String Array vet)
Integer l
Integer i
l = size(vet)
i = 0 to l - 1
Next
Done
vet[i] = ""
End

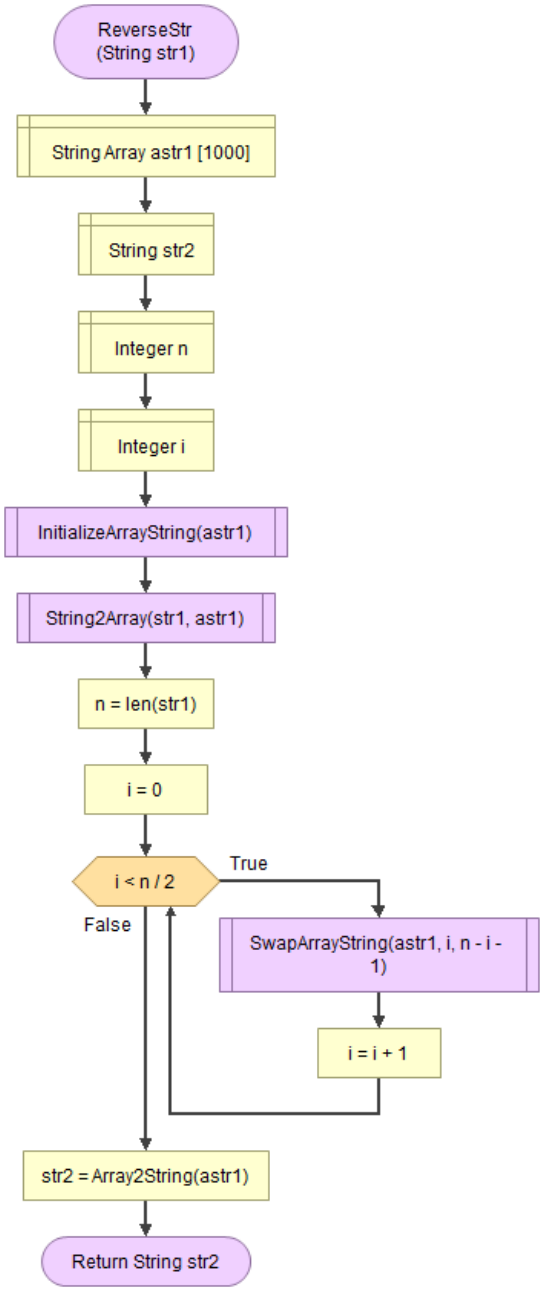

InitializeArrayInt
(Integer Array vet)
Integer i
Integer Length
Length = size(vet)
i = 0 to length - 1
Next
Done
vet[i] = 0
End

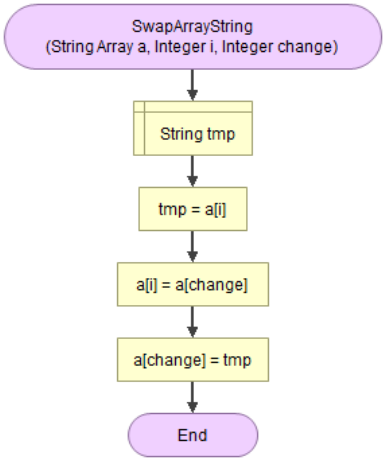
230.GSAF | Scrivere una frase al contrario

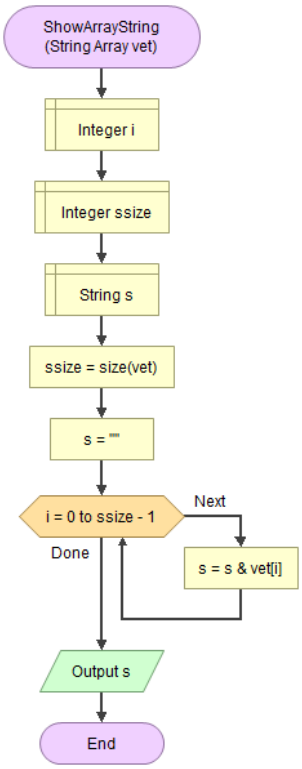
Reverse a phrase

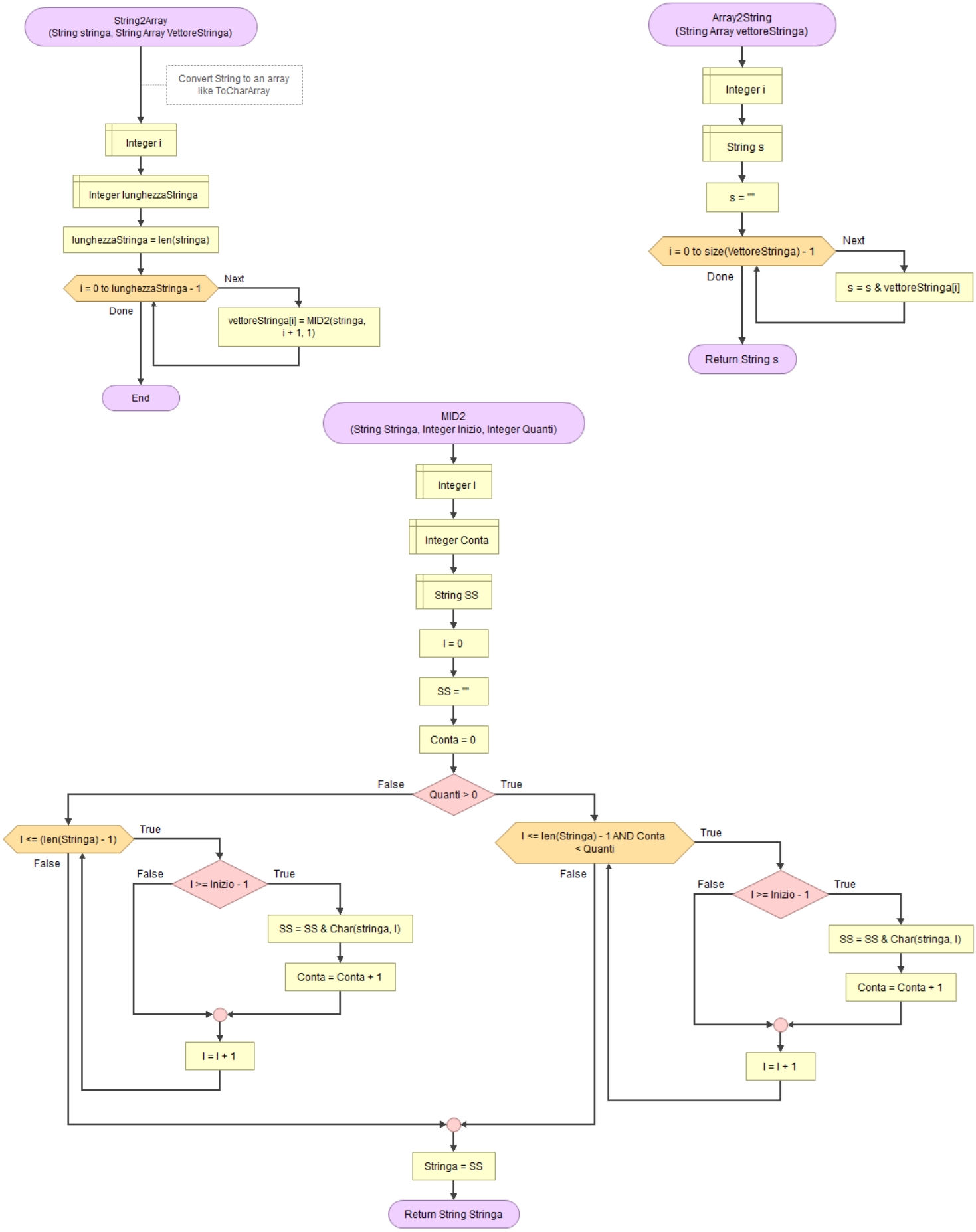






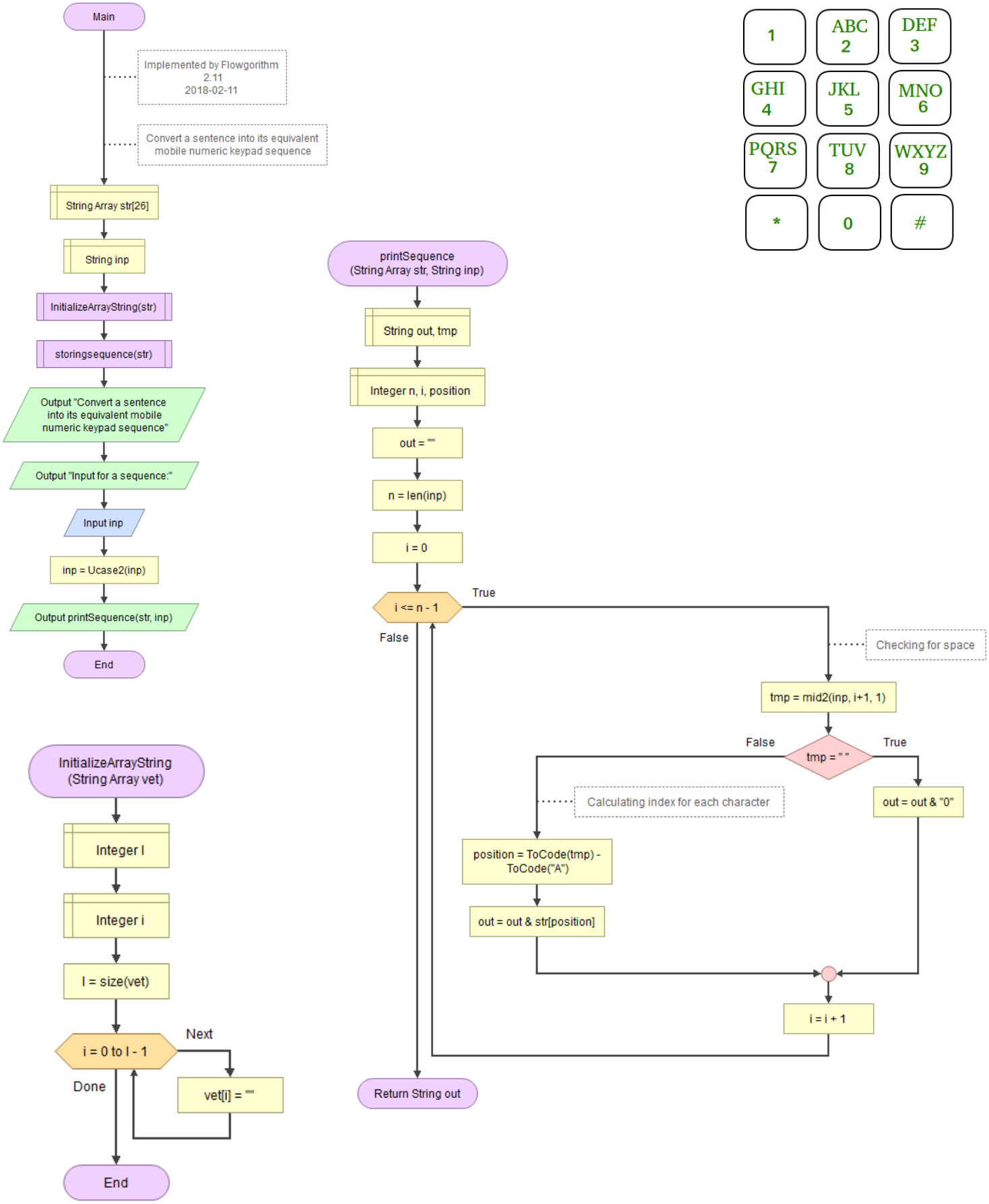


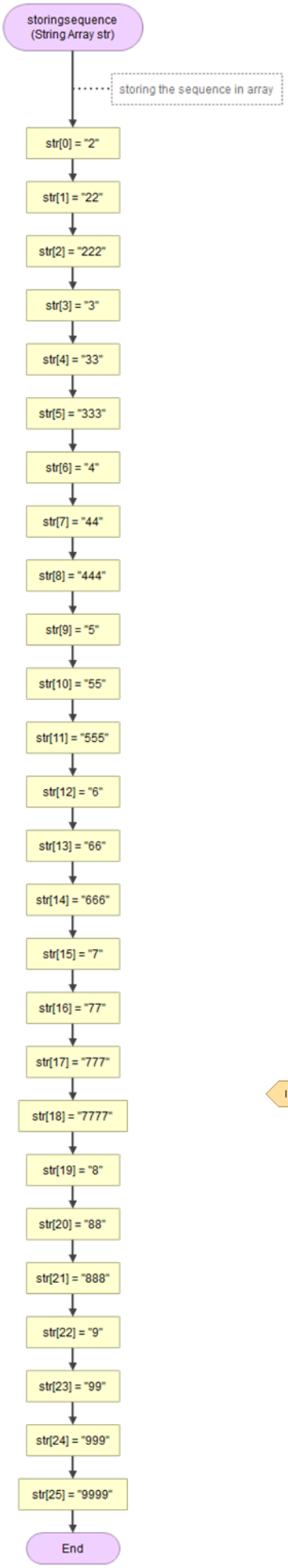


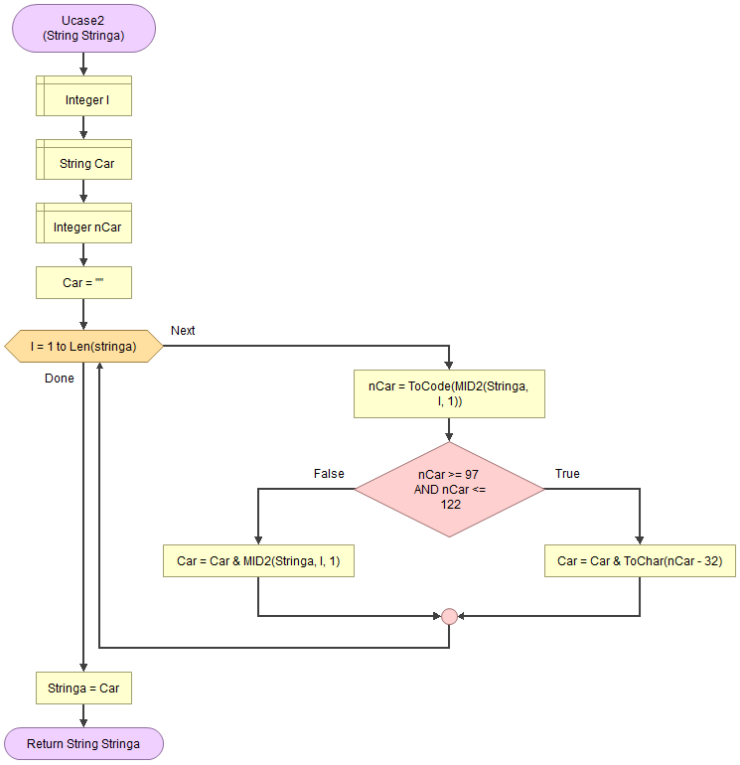

String2Array
(String stringa, String Array VettoreStringa)
Convert String to an array like ToCharArray
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len(stringa)
i = 0 to lunghezzaStringa - 1
Next
Done
vettoreStringa[i] = MID2(stringa, i + 1, 1)
End
Array2String
(String Array vettoreStringa)
Integer i
String s
s = ""
i = 0 to size(VettoreStringa) - 1
Next
Done
s = s & vettoreStringa[i]
Return String s
MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

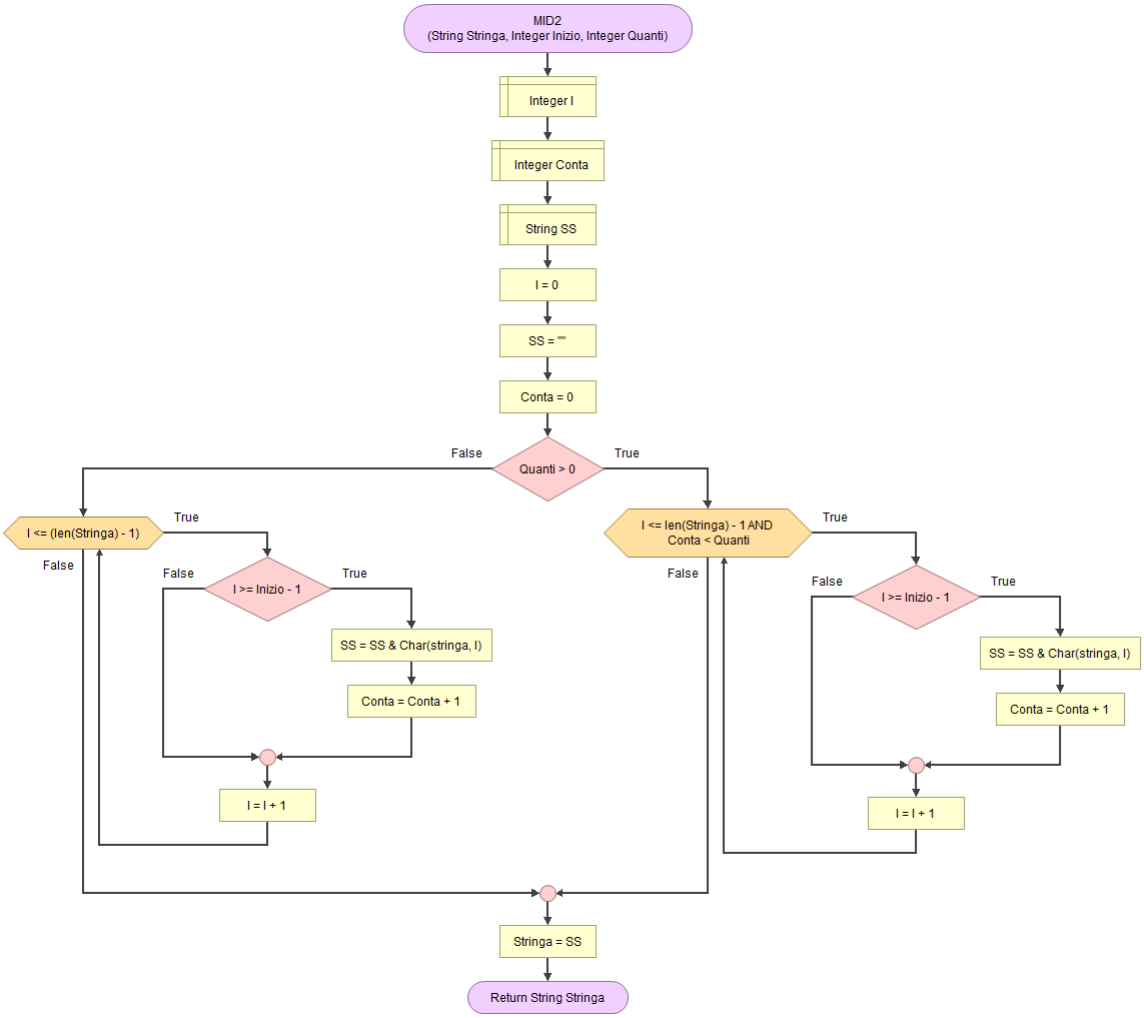
231.GSAF | Converte una stringa nella equivalente sequenza numerica del tastierino del cellulare

Convert a sentence into its equivalent mobile numeric keypad sequence



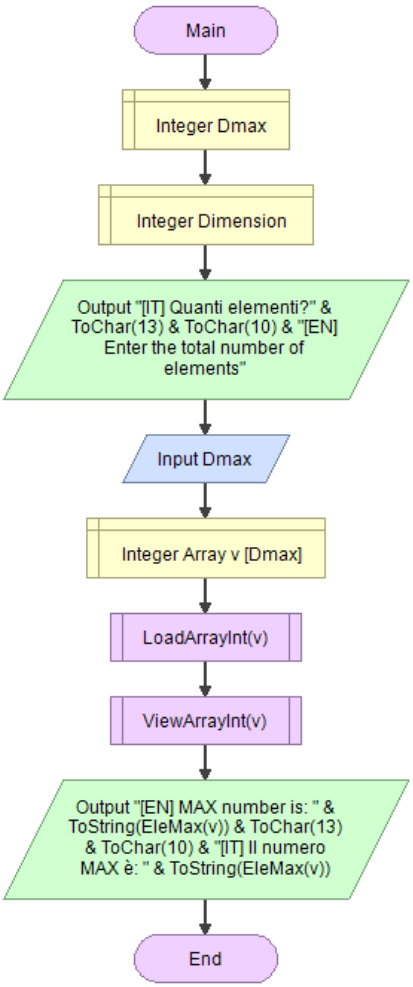

storingsequence
(String Array str)
storing the sequence in array
str[0] = "2"
str[1] = "22"
str[2] = "222"
str[3] = "3"
str[4] = "33"
str[5] = "333"
str[6] = "4"
str[7] = "44"
str[8] = "444"
str[9] = "5"
str[10] = "55"
str[11] = "555"
str[12] = "6"
str[13] = "66"
str[14] = "666"
str[15] = "7"
str[16] = "77"
str[17] = "777"
str[18] = "7777"
str[19] = "8"
str[20] = "88"
str[21] = "888"
str[22] = "9"
str[23] = "99"
str[24] = "999"
str[25] = "9999"
End

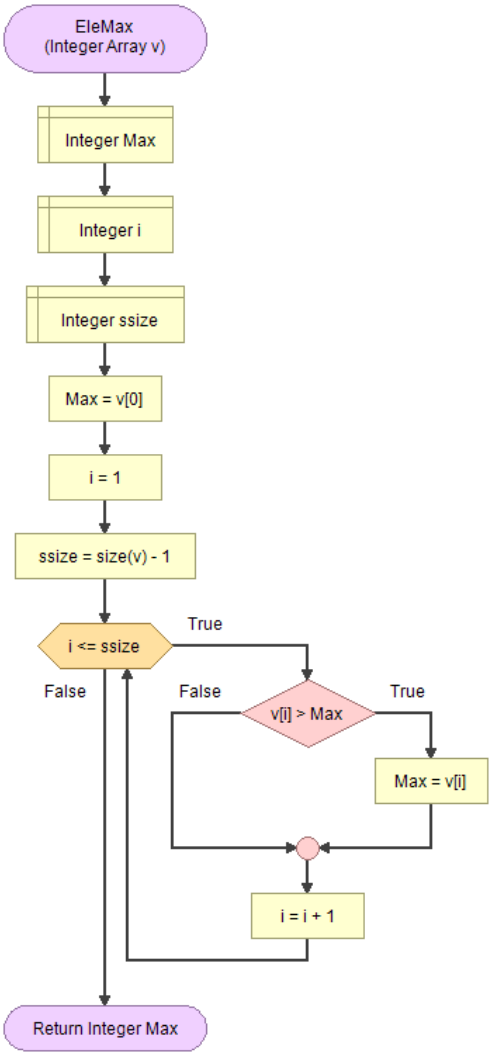

Ucase2
(String Stringa)
Integer I
String Car
Integer nCar
Car = ""
I = 1 to Len(stringa)
Next
Done
nCar = ToCode(MID2(Stringa, I, 1))
nCar >= 97 AND nCar <= 122
False
True
Car = Car & MID2(Stringa, I, 1)
Car = Car & ToChar(nCar - 32)
Stringa = Car
Return String Stringa

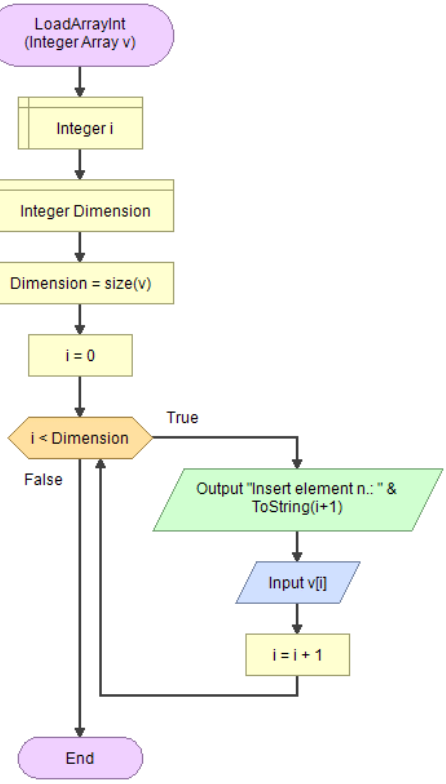

MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

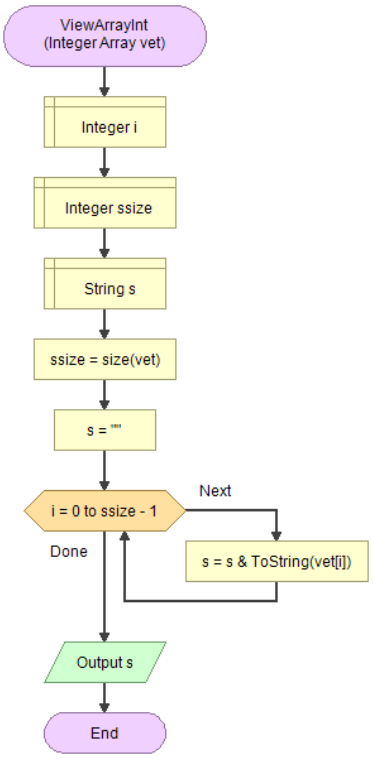
232.GSAF | Ricercare l'elemento maggiore di un vettore (con funzioni)

Find Largest Element in Array (by functions)



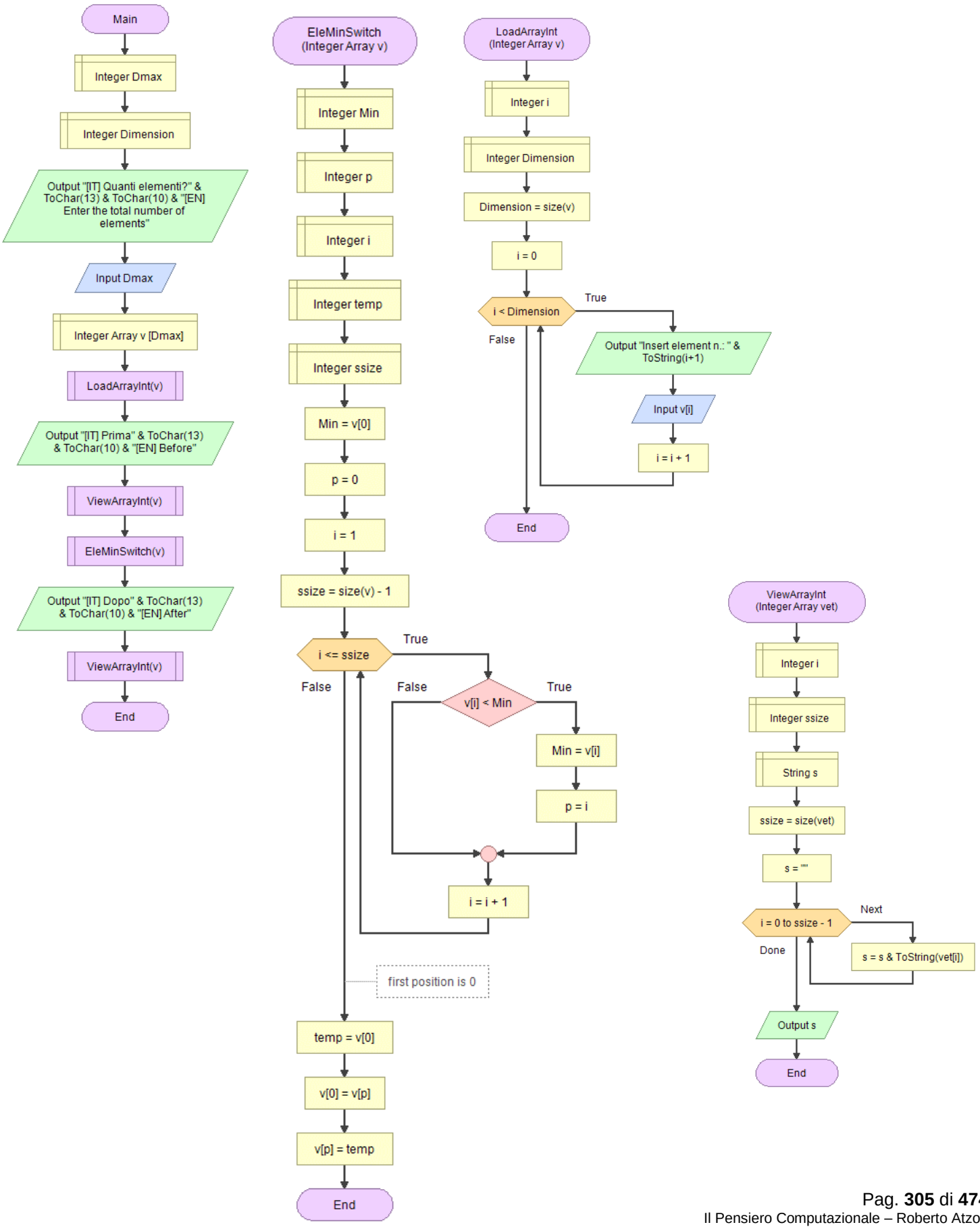






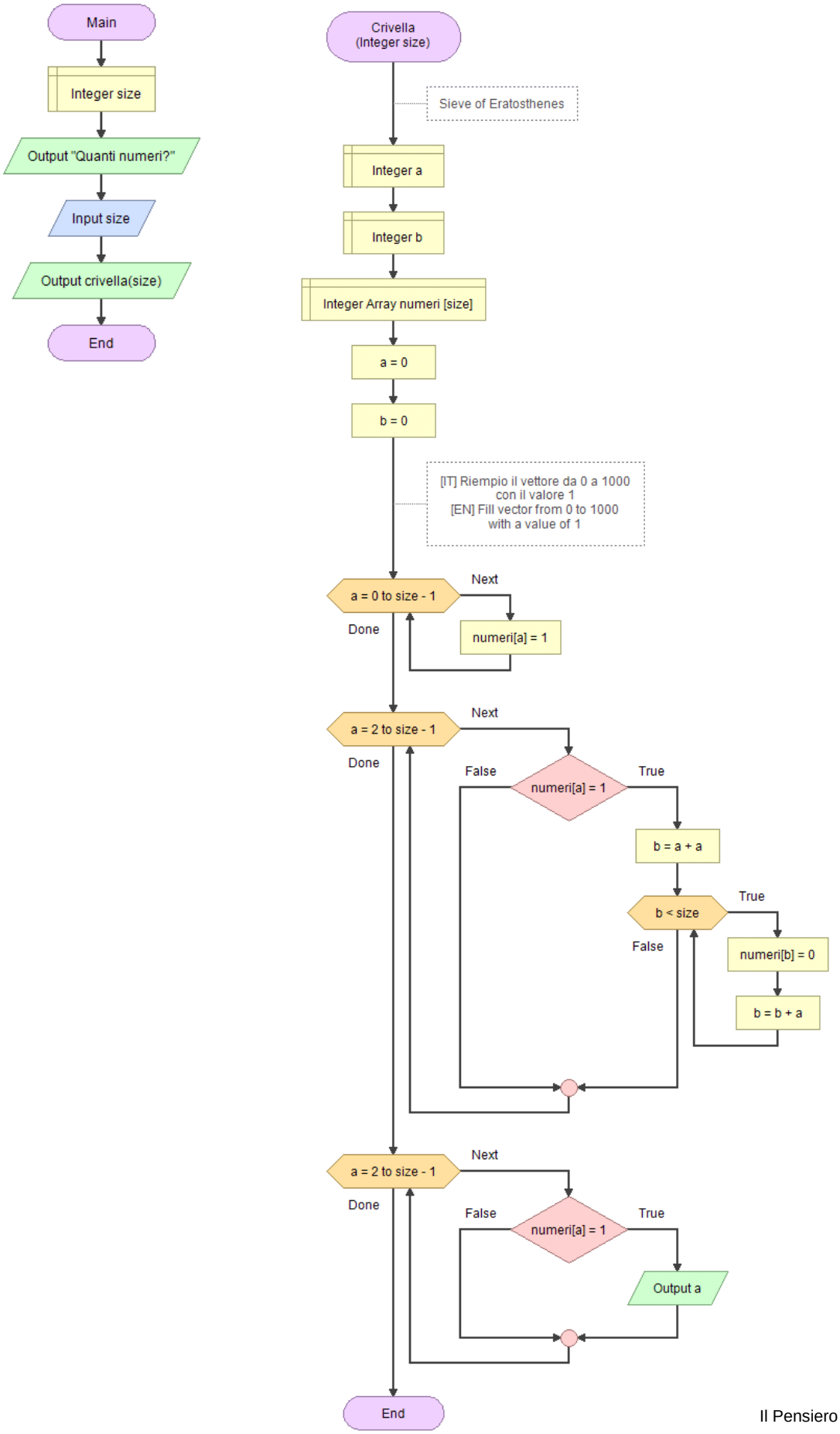
233.GSAF | Ricercare l'elemento minore di un vettore e scambiarlo con l'elemento nella prima posizione (con funzioni)

Find Smallest Element in Array and swap it with the first one (by functions)



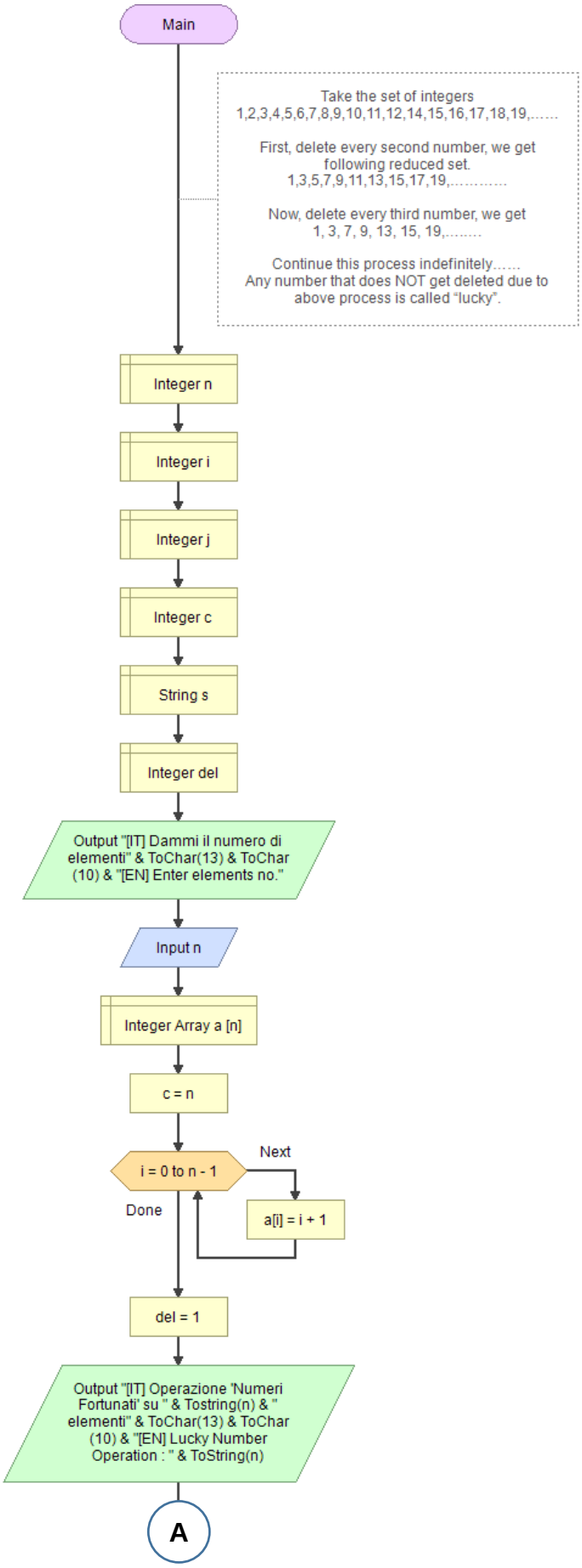
234.GAF ‖ Numeri primi con il Crivello di Eratostene

Prime numbers calculation by Sieve of Eratosthenes



235.GAF ‖ Numeri fortunati

Lucky Numbers

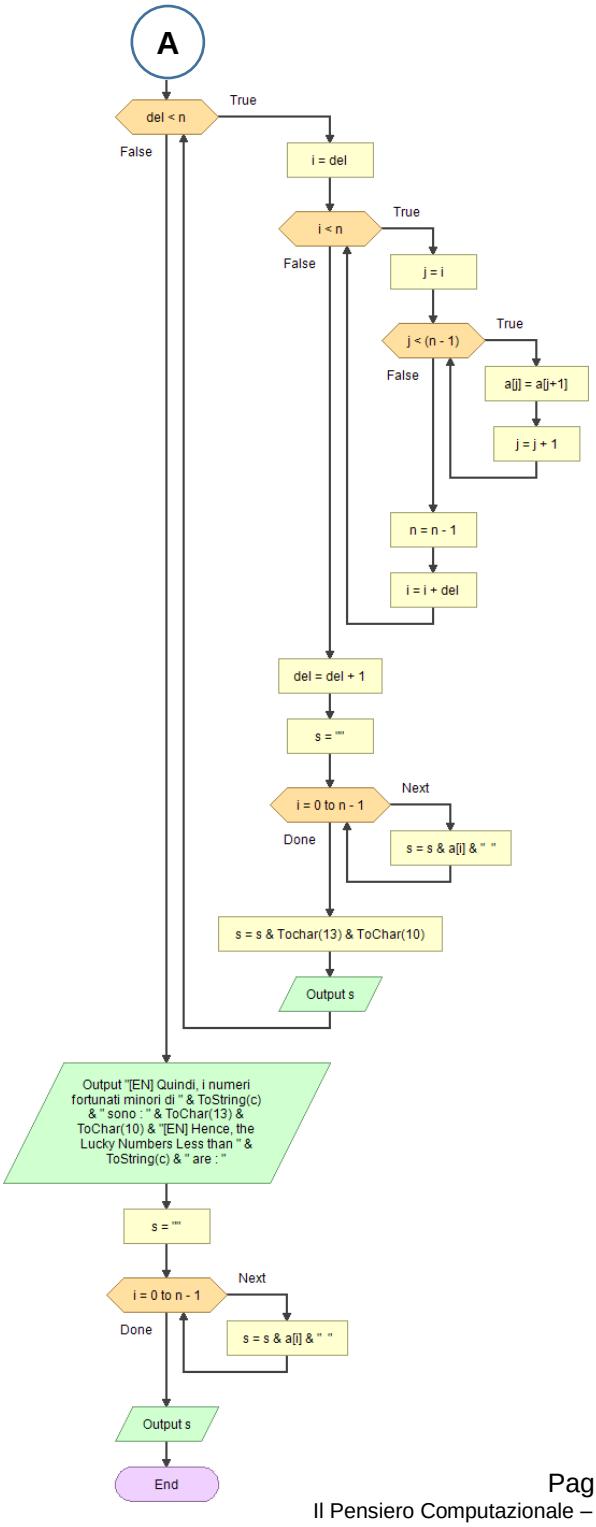


Take the set of integers
1,2,3,4,5,6,7,8,9,10,11,12,14,15,16,17,18,19,……

First, delete every second number, we get following reduced set.
1,3,5,7,9,11,13,15,17,19,……

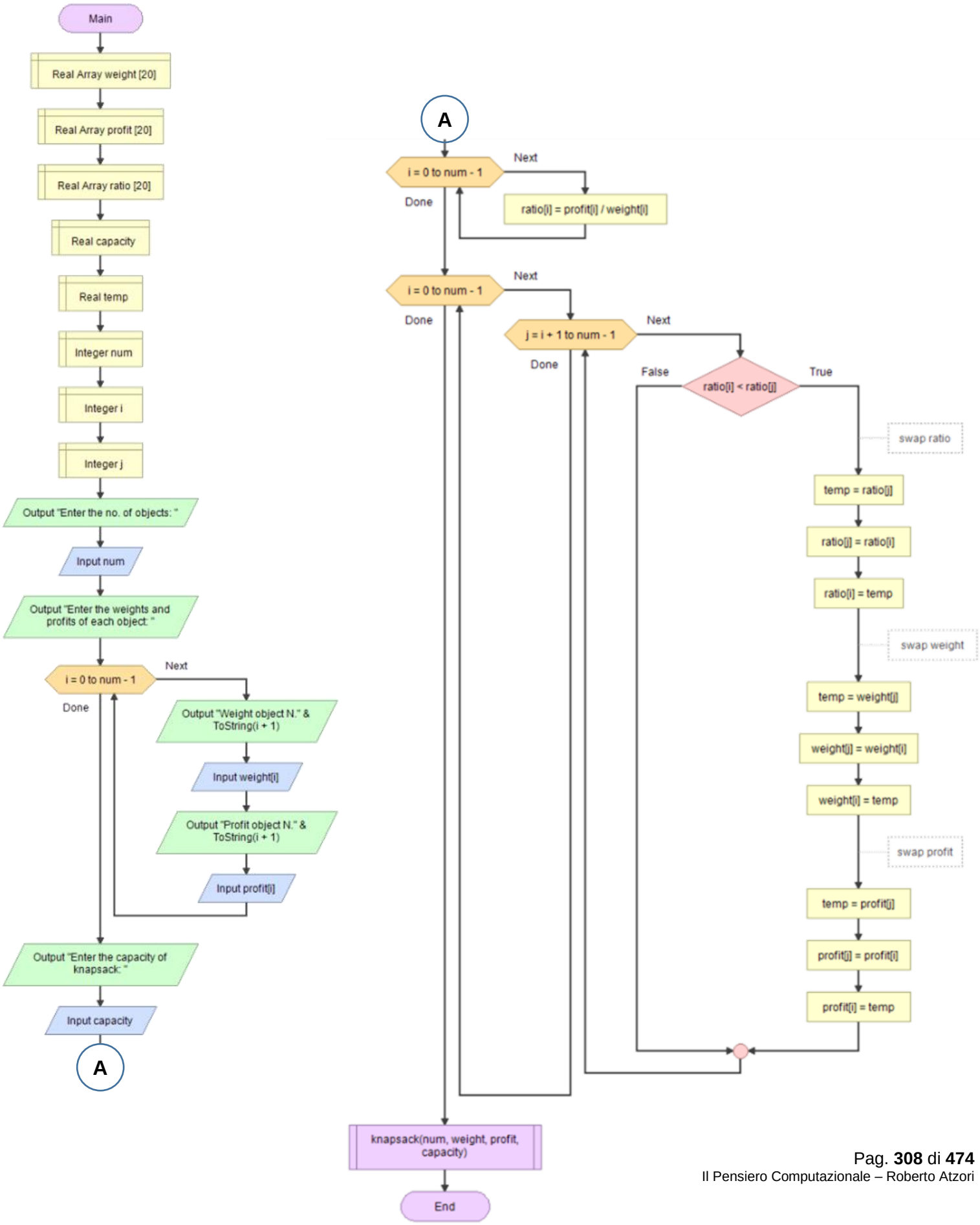
Now, delete every third number, we get
1, 3, 7, 9, 13, 15, 19,……..

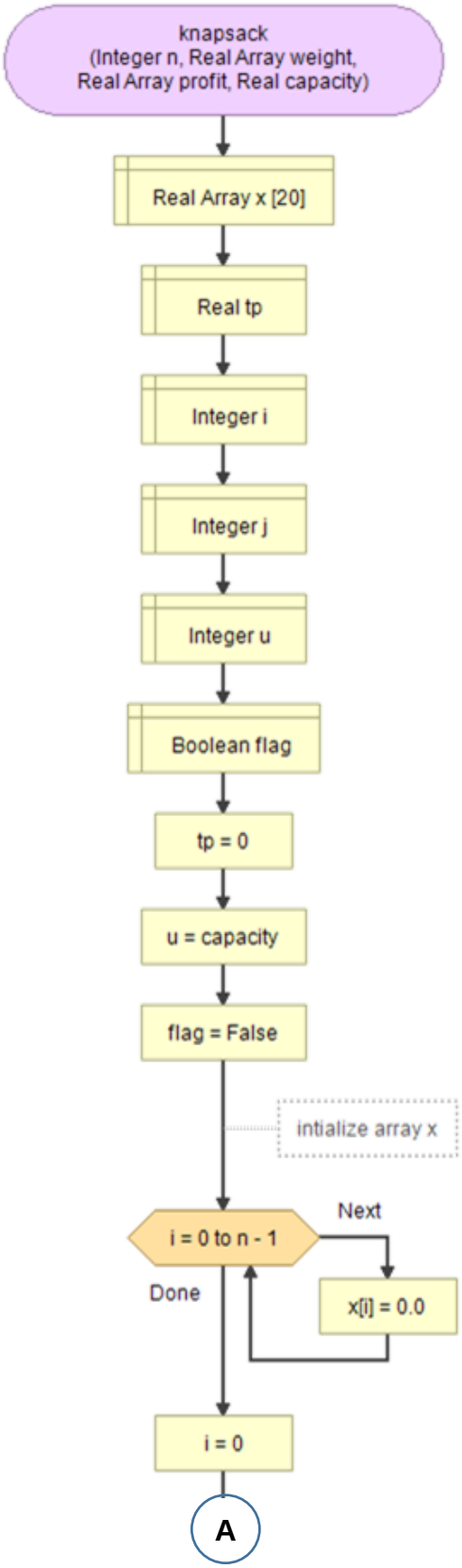
Continue this process indefinitely……
Any number that does NOT get deleted due to above process is called "lucky".

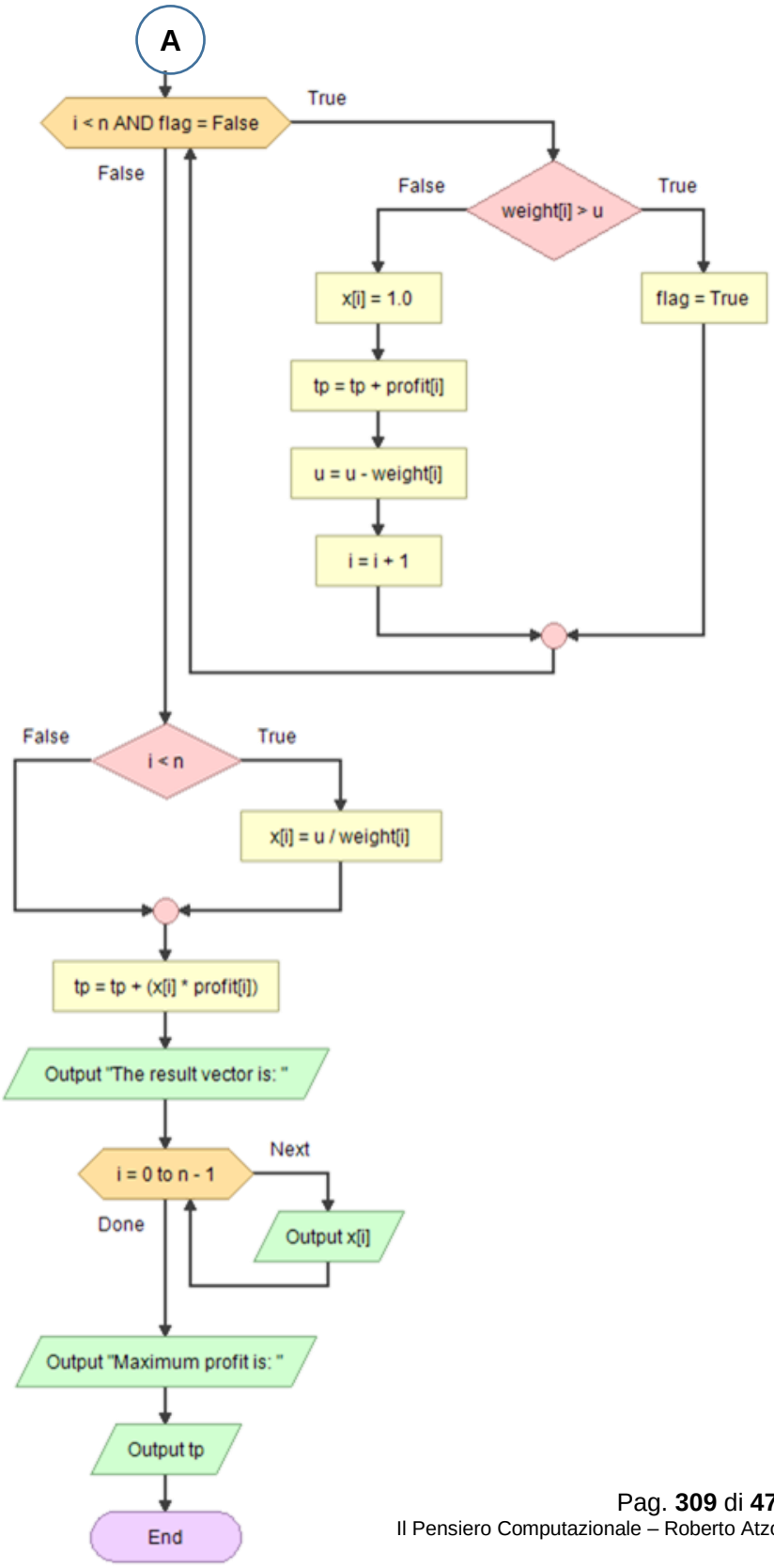


236. G A F ‖ Problema dello zaino con il metodo greedy

Greedy Knapsack

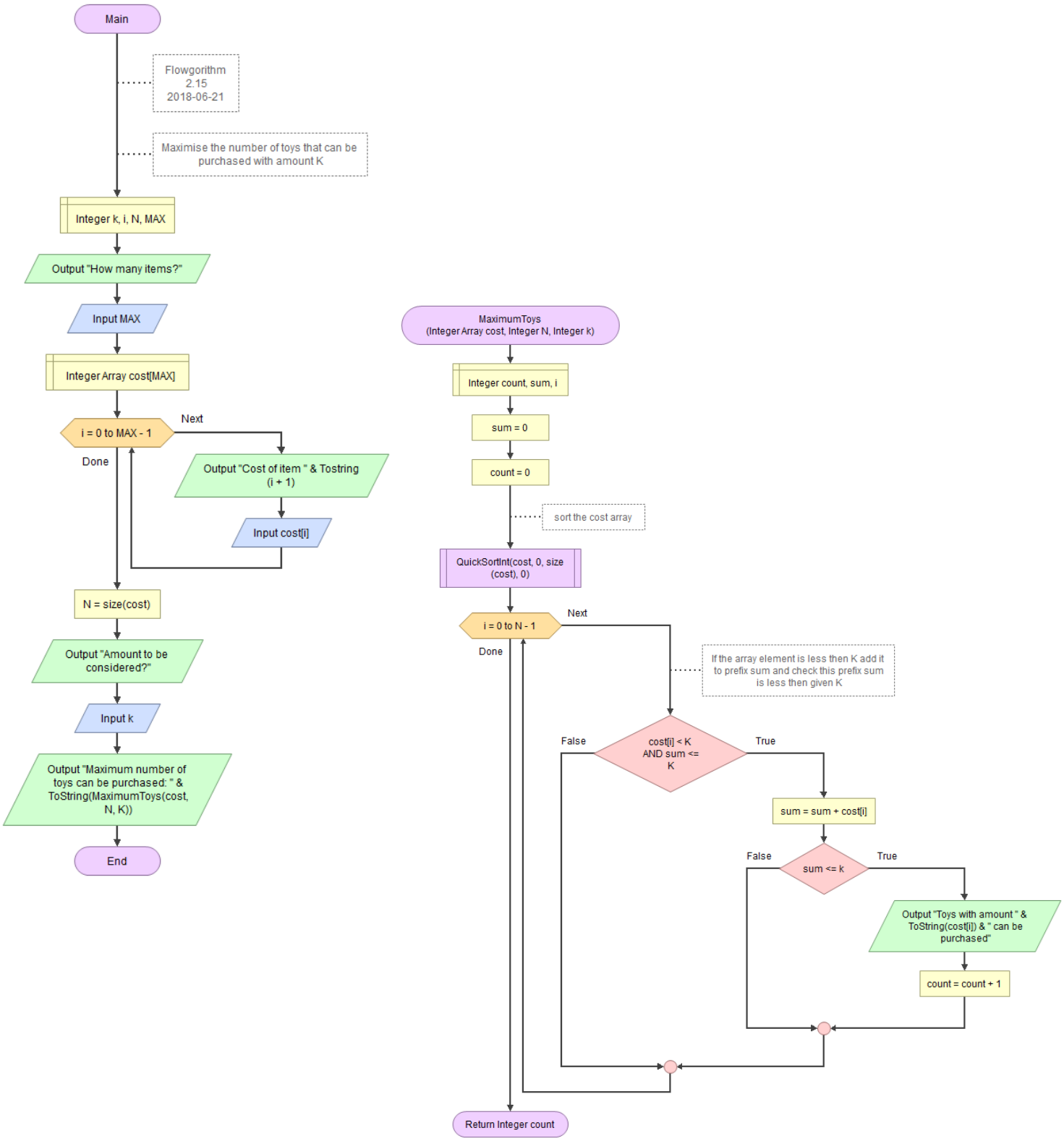


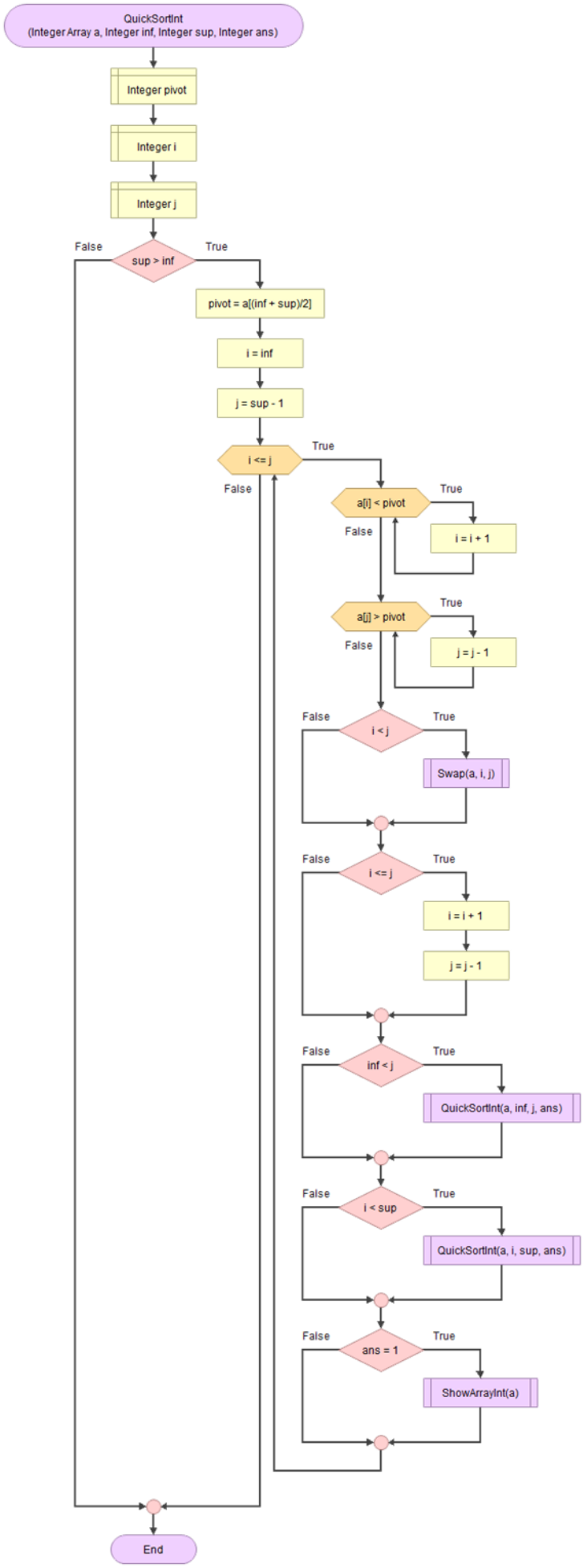

knapsack
(Integer n, Real Array weight,
Real Array profit, Real capacity)
Real Array x [20]
Real tp
Integer i
Integer j
Integer u
Boolean flag
tp = 0
u = capacity
flag = False
intialize array x
i = 0 to n - 1
Next
Done
x[i] = 0.0
i = 0
A

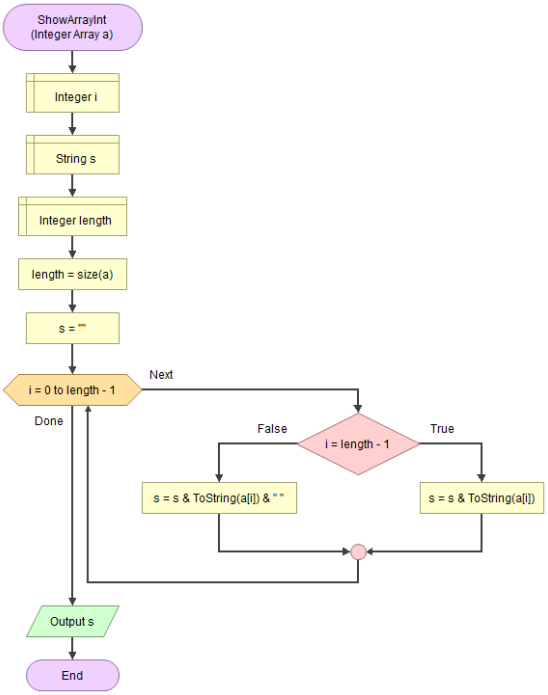

A
i < n AND flag = False
True
False
False
weight[i] > u
True
x[i] = 1.0
flag = True
tp = tp + profit[i]
u = u - weight[i]
i = i + 1
False
i < n
True
x[i] = u / weight[i]
tp = tp + (x[i] * profit[i])
Output "The result vector is: "
i = 0 to n - 1
Next
Done
Output x[i]
Output "Maximum profit is: "
Output tp
End

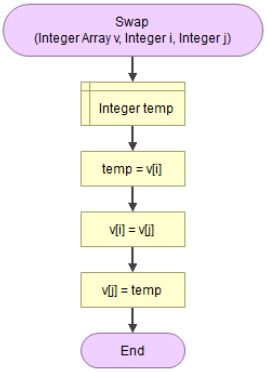
237.GAFR ‖ Massimizzare l'acquisto di giocattoli da comprare avendo una certa somma da spendere

Maximise the number of toys that can be purchased with amount K



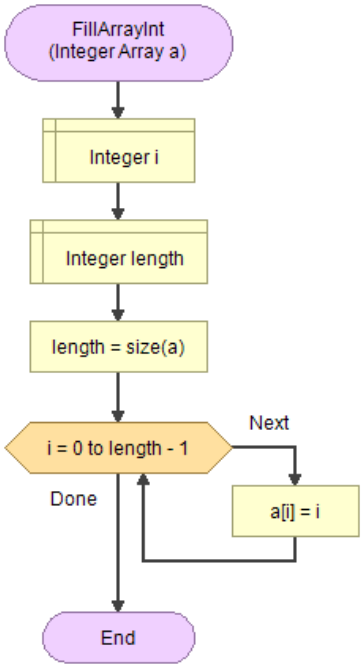

QuickSortInt
(Integer Array a, Integer inf, Integer sup, Integer ans)
Integer pivot
Integer i
Integer j
sup > inf
False
True
pivot = a[(inf + sup)/2]
i = inf
j = sup - 1
i <= j
True
False
a[i] < pivot
True
False
i = i + 1
a[j] > pivot
True
False
j = j - 1
i < j
False
True
Swap(a, i, j)
i <= j
False
True
i = i + 1
j = j - 1
inf < j
False
True
QuickSortInt(a, inf, j, ans)
i < sup
False
True
QuickSortInt(a, i, sup, ans)
ans = 1
False
True
ShowArrayInt(a)
End

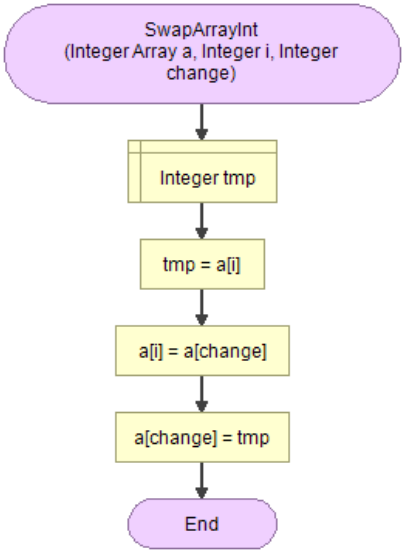

ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
i = length - 1
False
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
End

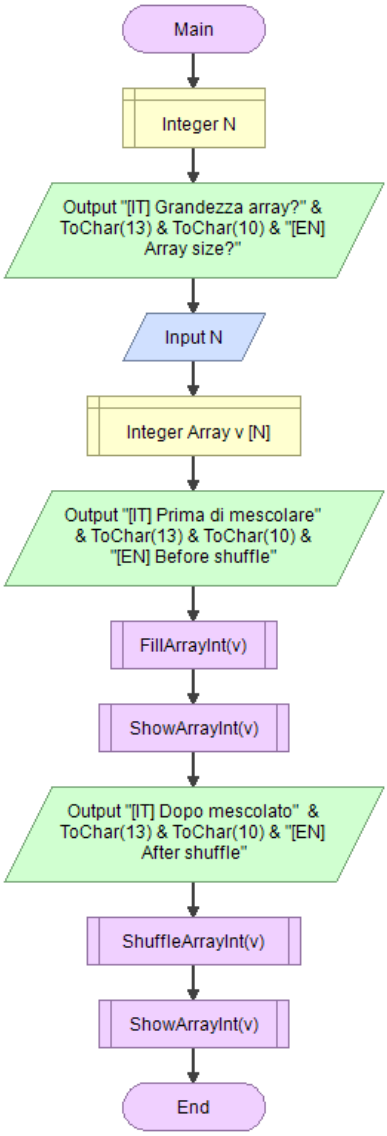

Swap
(Integer Array v, Integer i, Integer j)
Integer temp
temp = v[i]
v[i] = v[j]
v[j] = temp
End

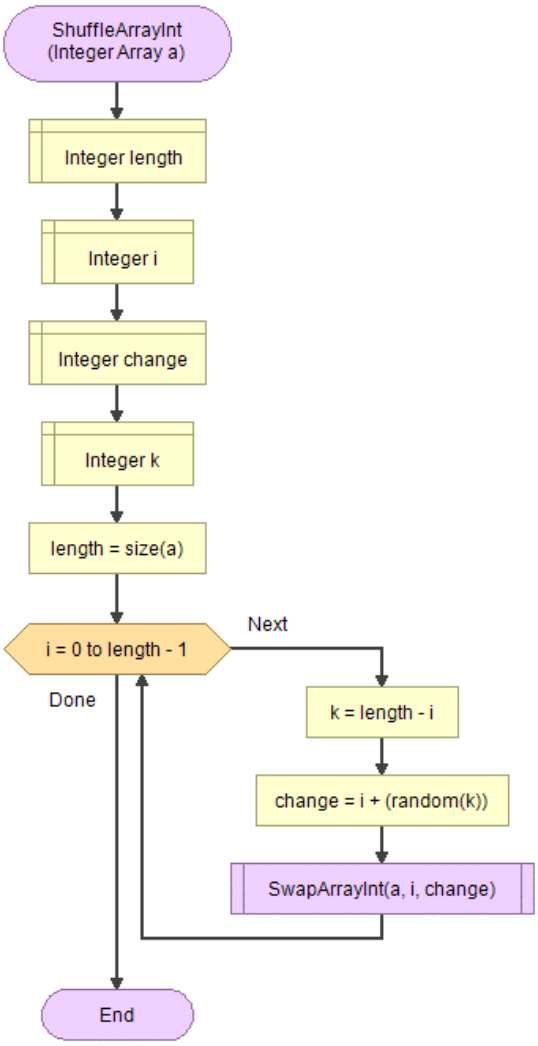
238. G A F ‖ Mescola array di interi

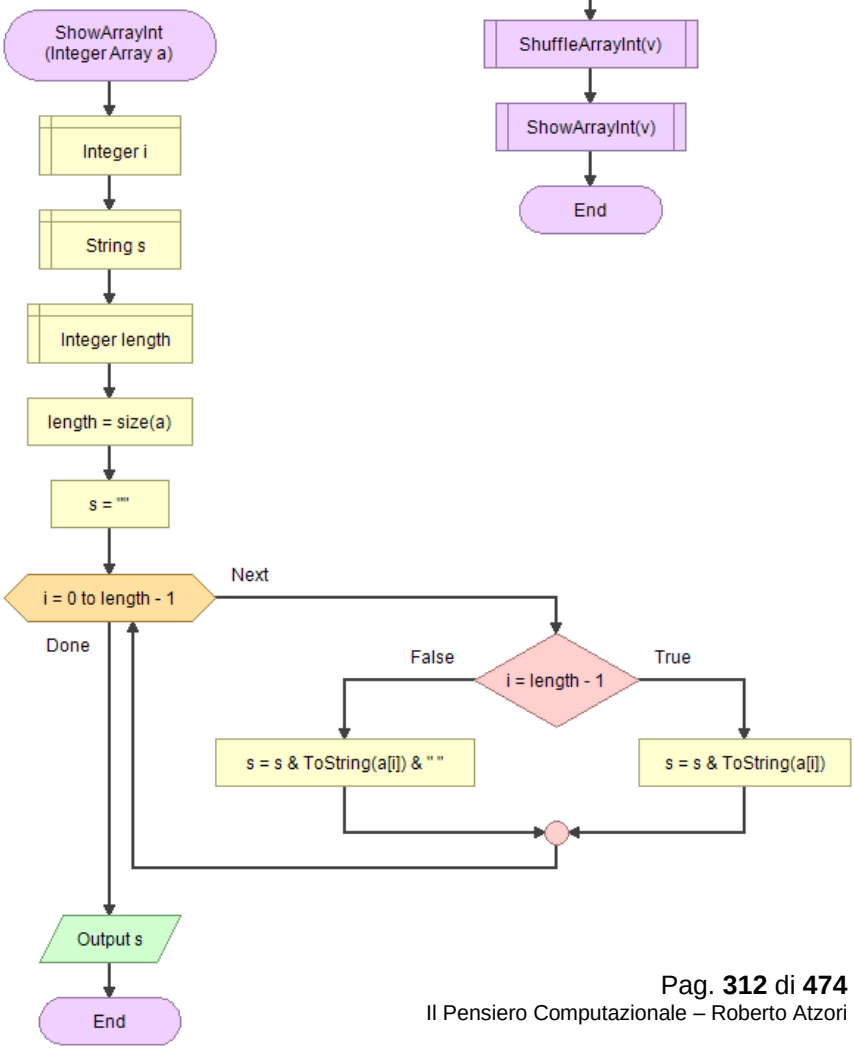
Shuffle Integers Array





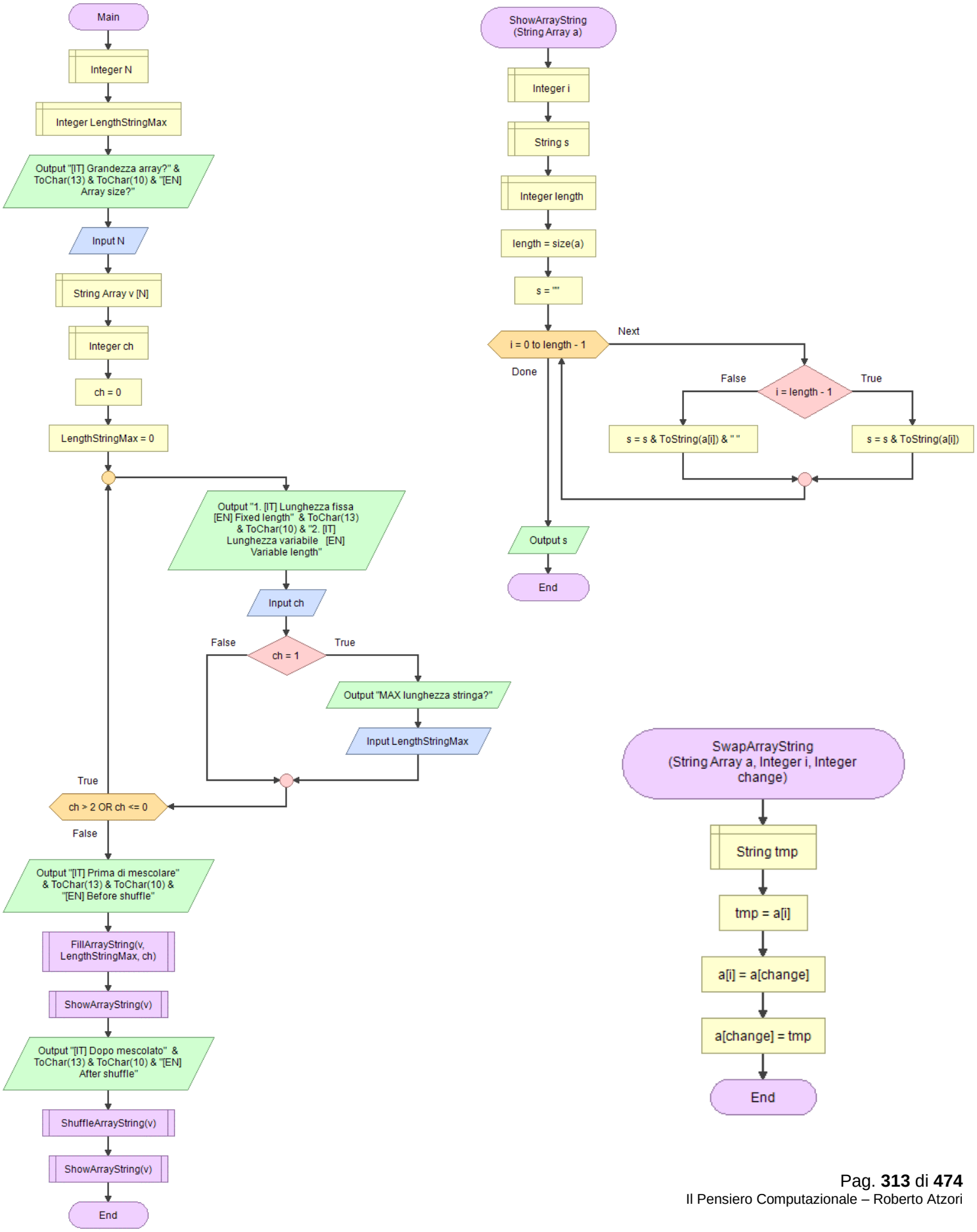


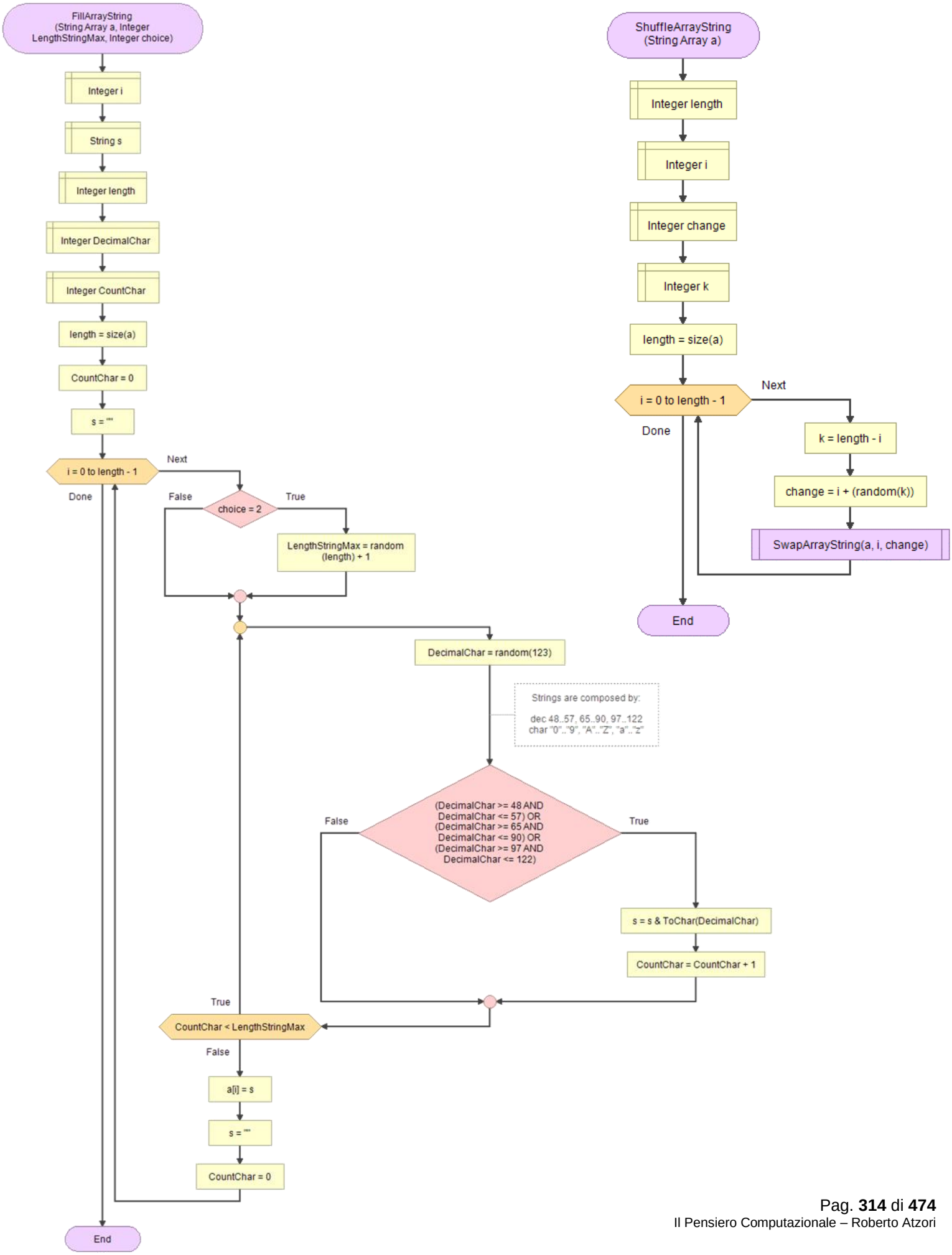




239.GAF ‖ Mescola array di stringhe casuali di grandezza fissa o variabile

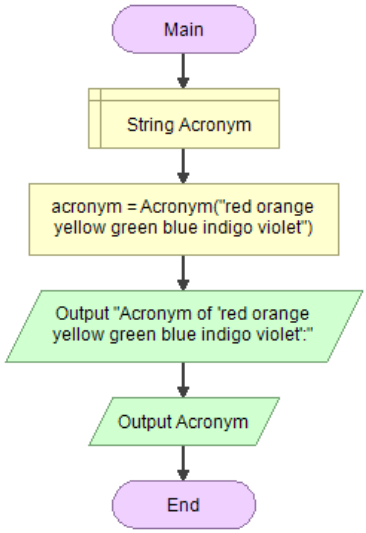
Shuffle random Strinsgs Array with fixed or variable size

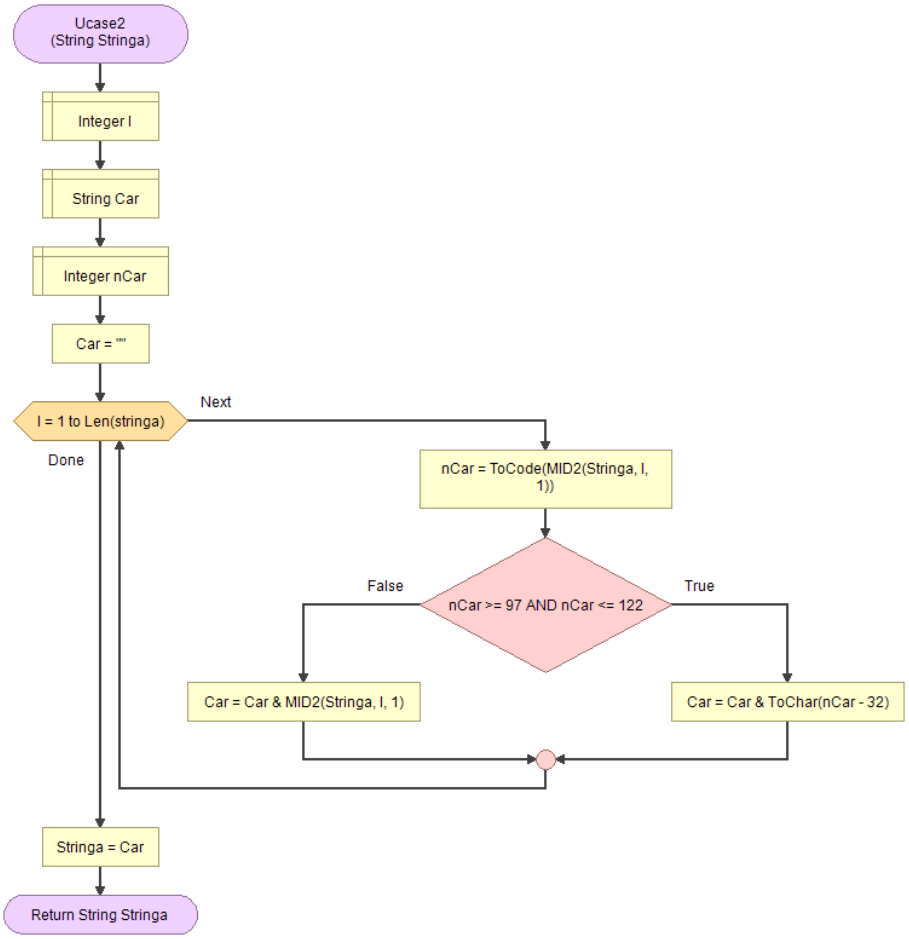


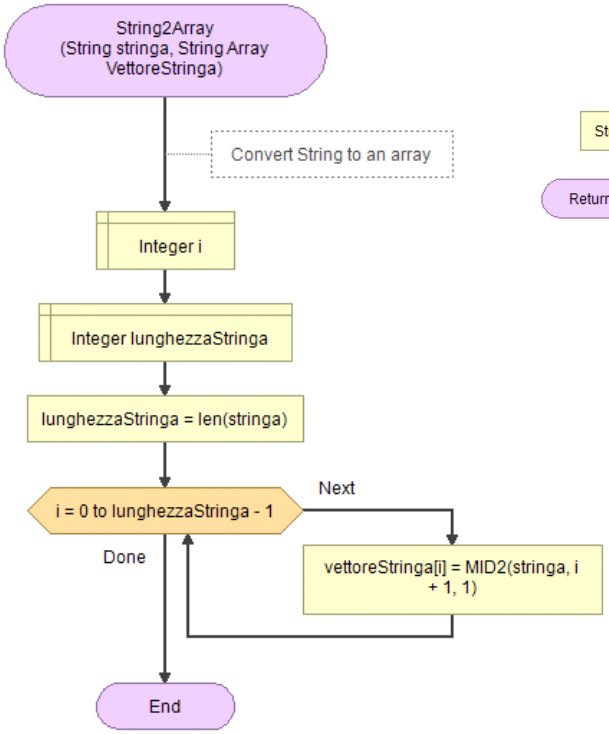

FillArrayString
(String Array a, Integer
LengthStringMax, Integer choice)
Integer i
String s
Integer length
Integer DecimalChar
Integer CountChar
length = size(a)
CountChar = 0
s = ""
i = 0 to length - 1
Next
Done
False
choice = 2
True
LengthStringMax = random
(length) + 1
DecimalChar = random(123)
Strings are composed by:
dec 48..57, 65..90, 97..122
char "0".."9", "A".."Z", "a".."z"
(DecimalChar >= 48 AND
DecimalChar <= 57) OR
(DecimalChar >= 65 AND
DecimalChar <= 90) OR
(DecimalChar >= 97 AND
DecimalChar <= 122)
False
True
s = s & ToChar(DecimalChar)
CountChar = CountChar + 1
True
CountChar < LengthStringMax
False
a[i] = s
s = ""
CountChar = 0
End
ShuffleArrayString
(String Array a)
Integer length
Integer i
Integer change
Integer k
length = size(a)
i = 0 to length - 1
Next
Done
k = length - i
change = i + (random(k))
SwapArrayString(a, i, change)
End

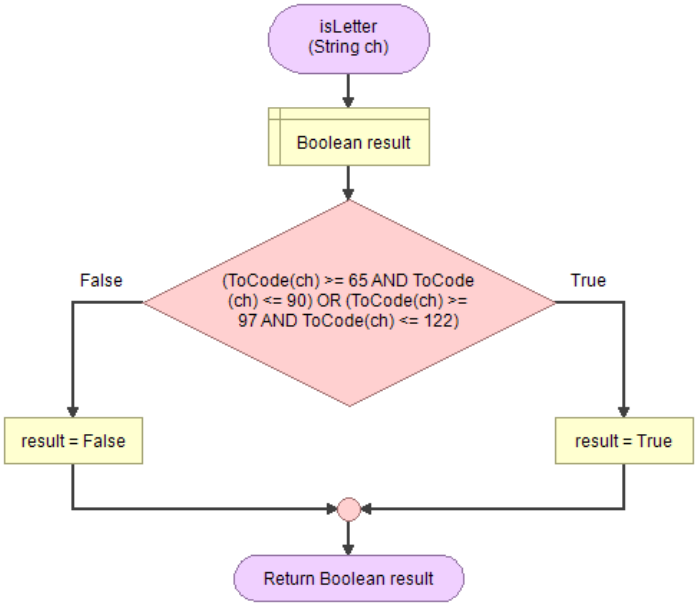
240.GAF | Acronimo

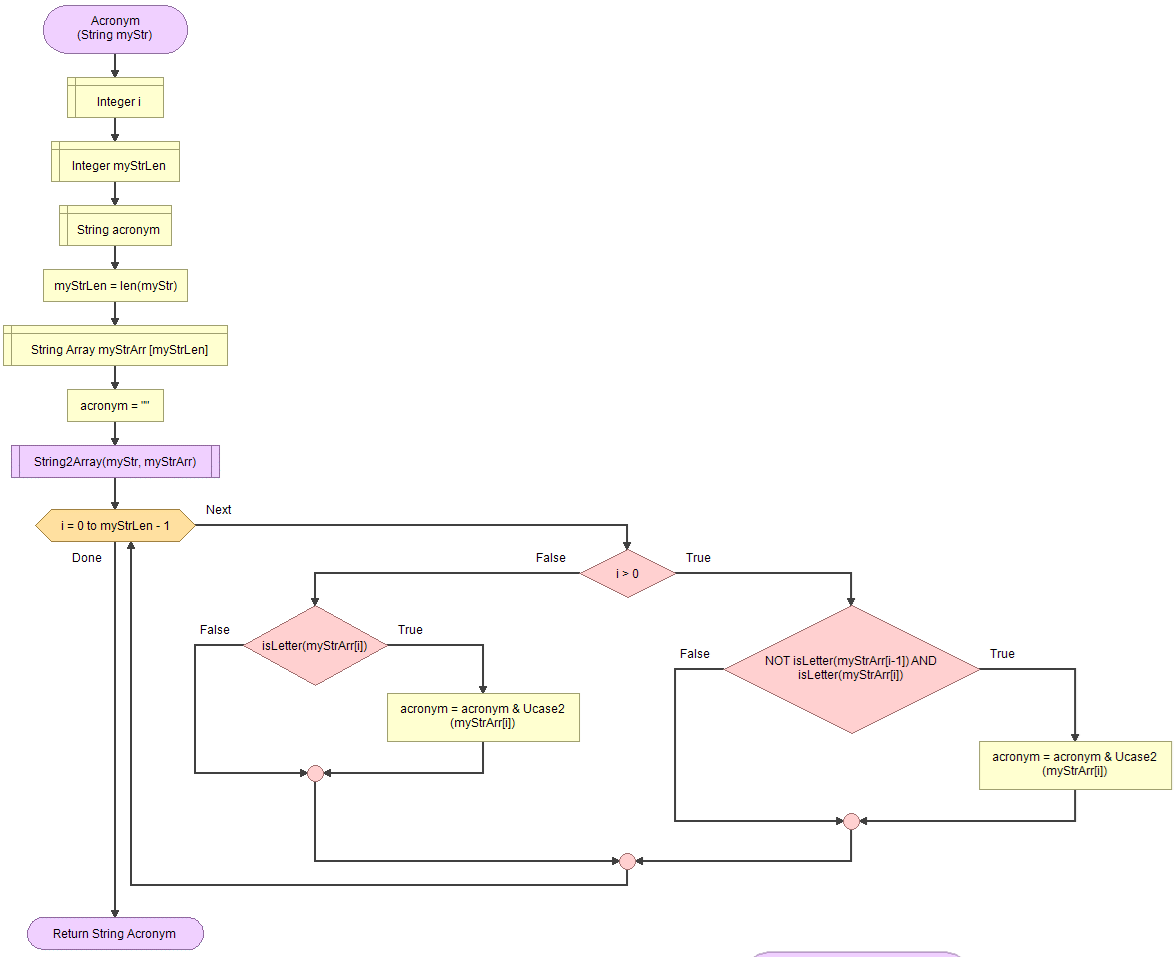
Acronym

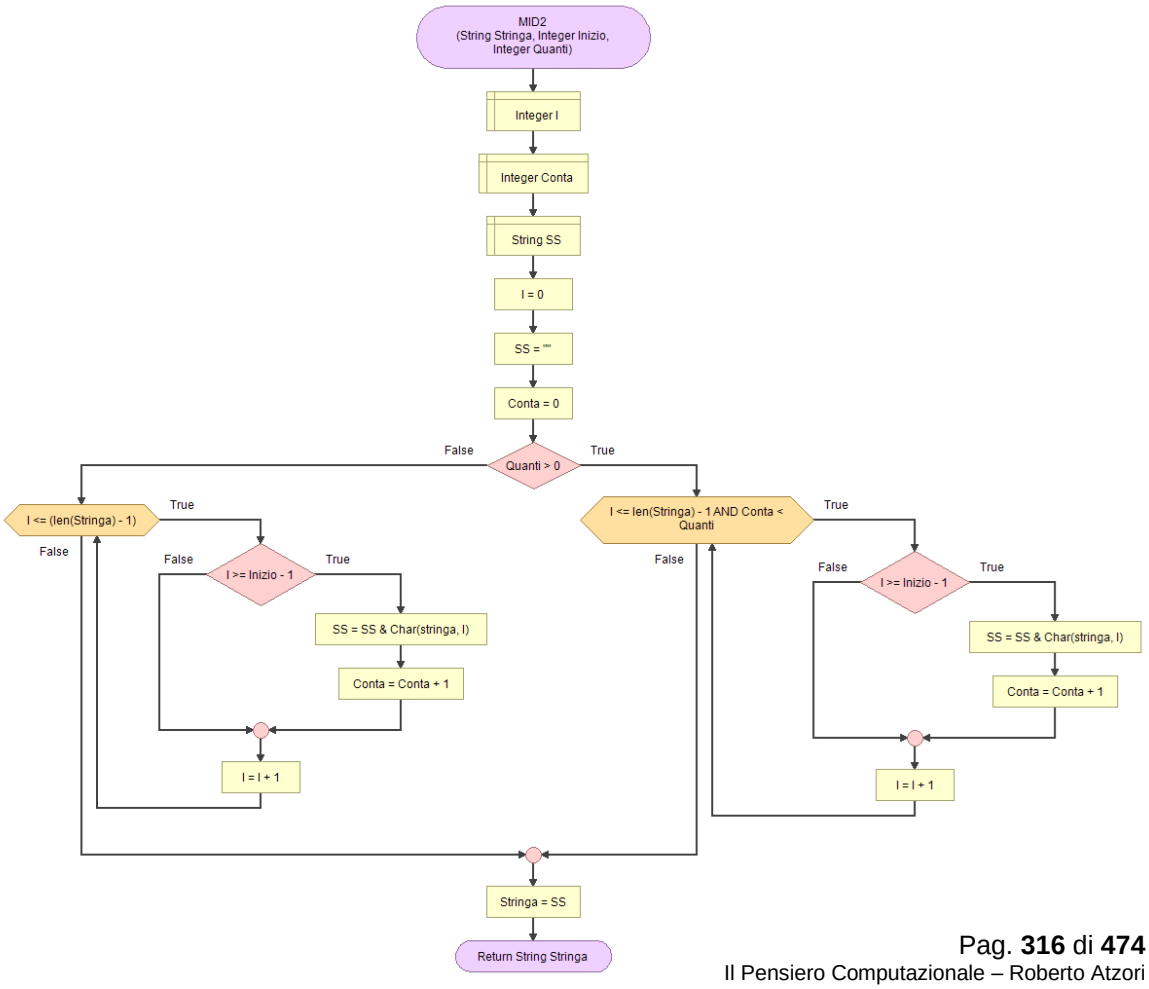






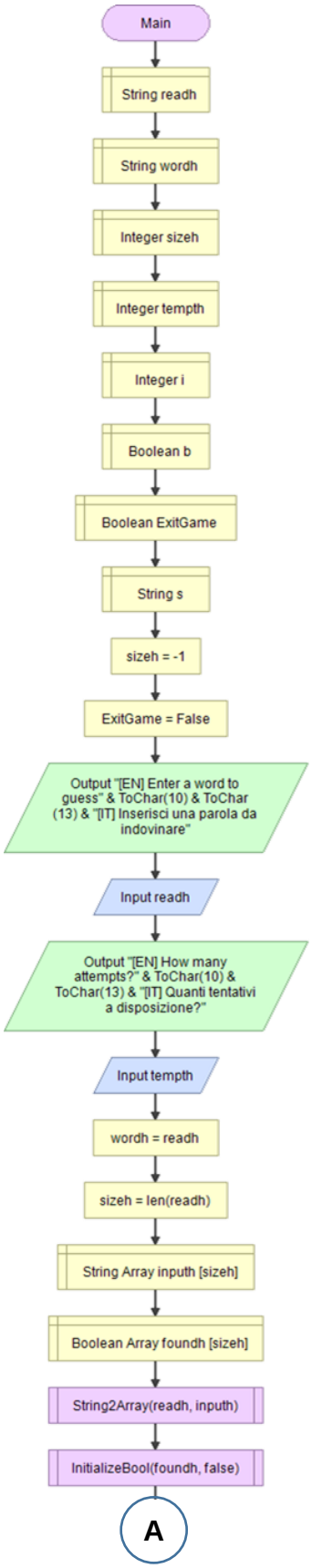


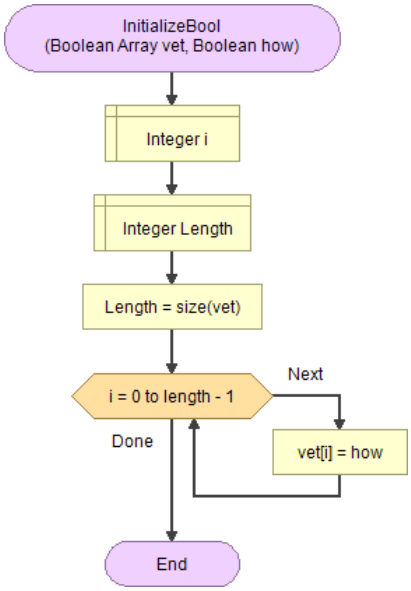

Acronym
(String myStr)
Integer i
Integer myStrLen
String acronym
myStrLen = len(myStr)
String Array myStrArr [myStrLen]
acronym = ""
String2Array(myStr, myStrArr)
i = 0 to myStrLen - 1
Next
Done
i > 0
False
True
isLetter(myStrArr[i])
False
True
acronym = acronym & Ucase2 (myStrArr[i])
NOT isLetter(myStrArr[i-1]) AND isLetter(myStrArr[i])
False
True
acronym = acronym & Ucase2 (myStrArr[i])
Return String Acronym

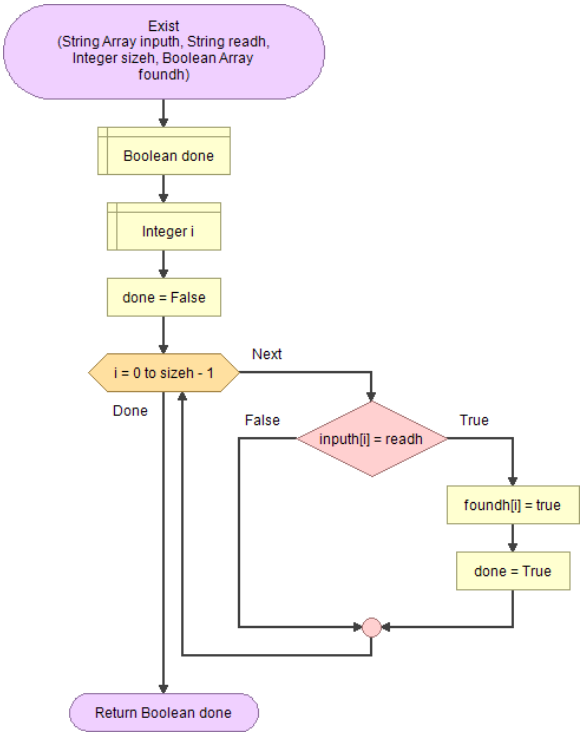

MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

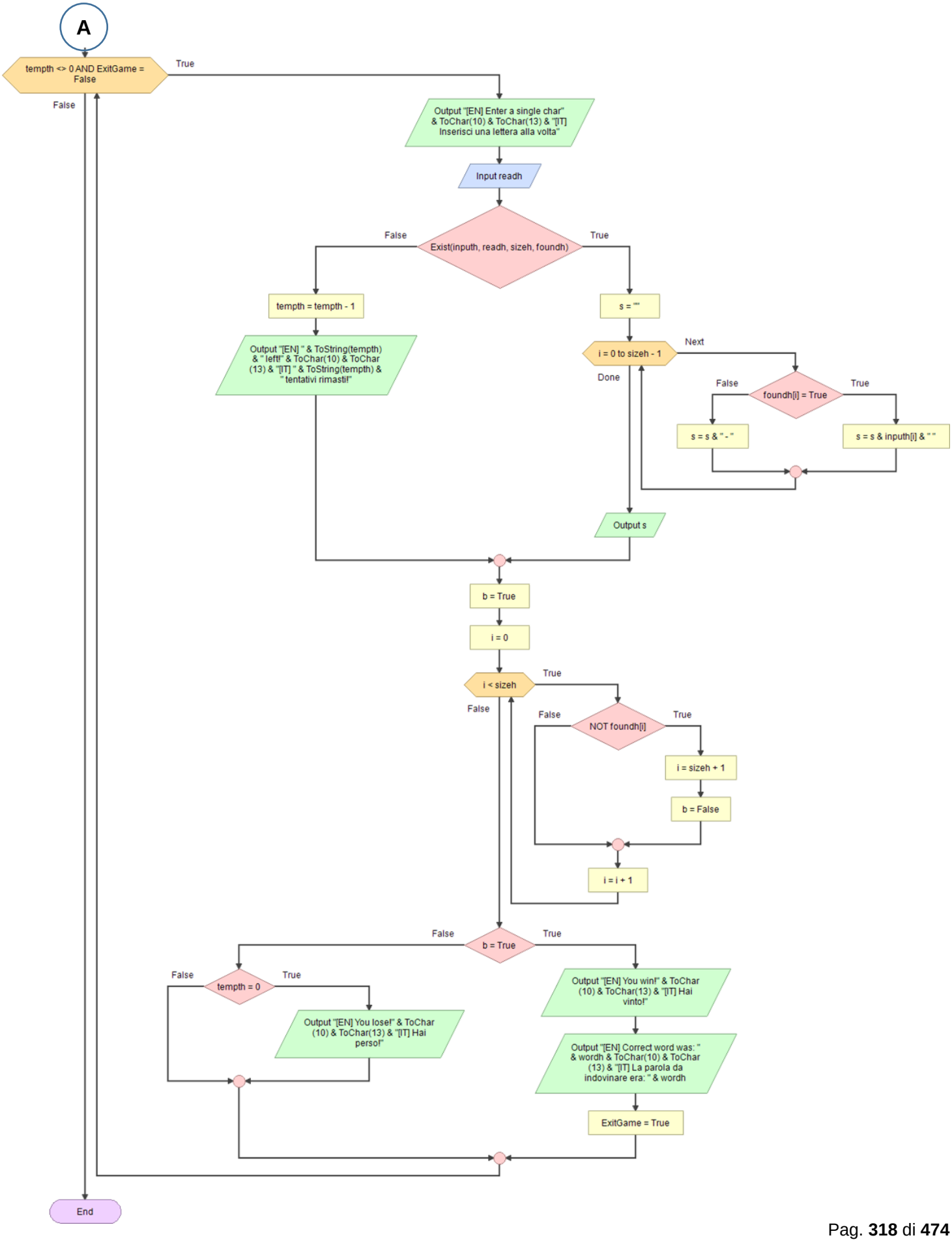
241.GAF ‖ Il gioco dell'impiccato

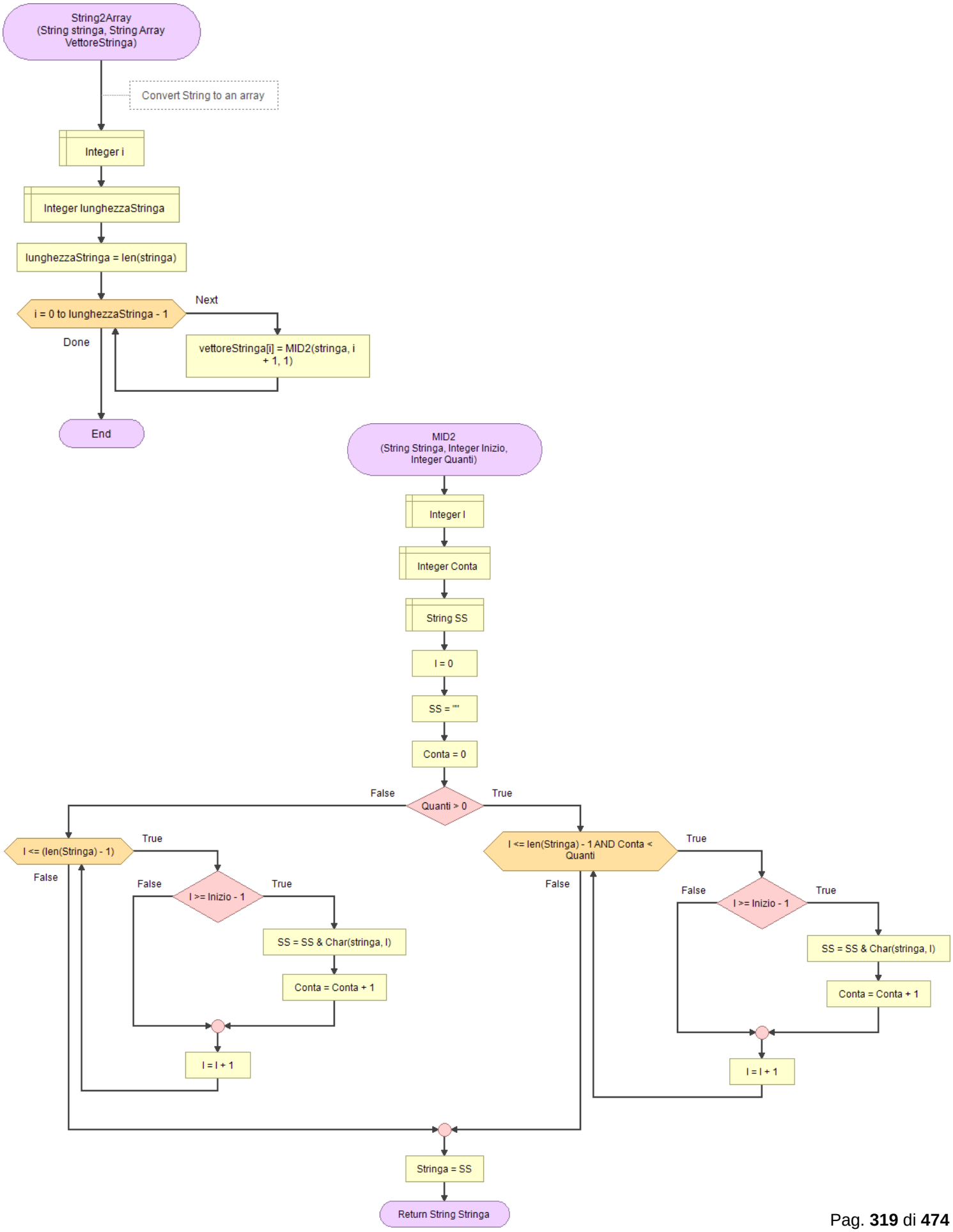
Hangman game





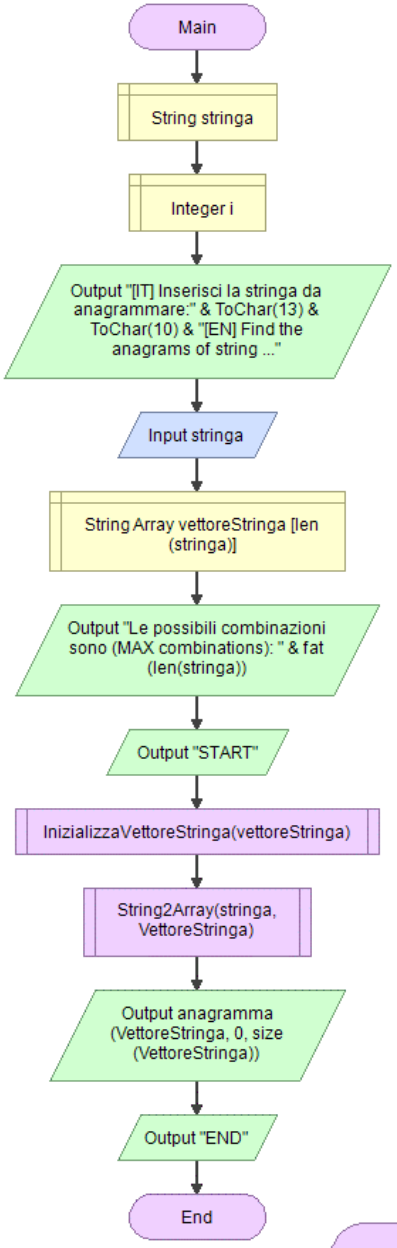


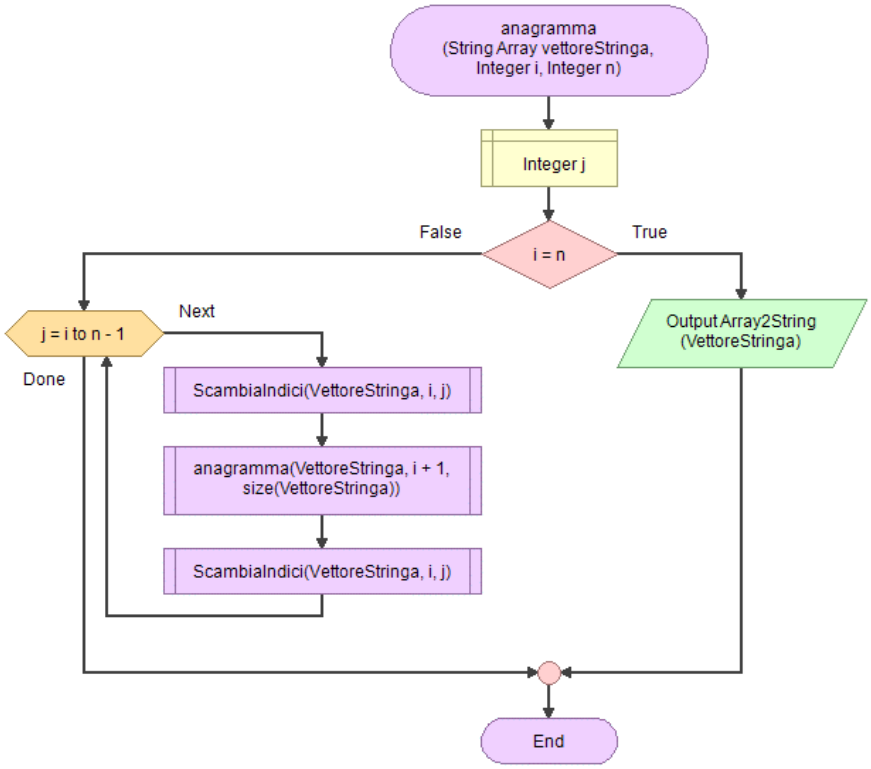

A
tempth <> 0 AND ExitGame = False
True
False
Output "[EN] Enter a single char" & ToChar(10) & ToChar(13) & "[IT] Inserisci una lettera alla volta"
Input readh
Exist(inputh, readh, sizeh, foundh)
False
True
tempth = tempth - 1
Output "[EN] " & ToString(tempth) & " left!" & ToChar(10) & ToChar (13) & "[IT] " & ToString(tempth) & " tentativi rimasti!"
s = ""
i = 0 to sizeh - 1
Next
Done
foundh[i] = True
False
True
s = s & " - "
s = s & inputh[i] & " "
Output s
b = True
i = 0
i < sizeh
True
False
NOT foundh[i]
False
True
i = sizeh + 1
b = False
i = i + 1
b = True
False
True
tempth = 0
False
True
Output "[EN] You lose!" & ToChar (10) & ToChar(13) & "[IT] Hai perso!"
Output "[EN] You win!" & ToChar (10) & ToChar(13) & "[IT] Hai vinto!"
Output "[EN] Correct word was: " & wordh & ToChar(10) & ToChar (13) & "[IT] La parola da indovinare era: " & wordh
ExitGame = True
End

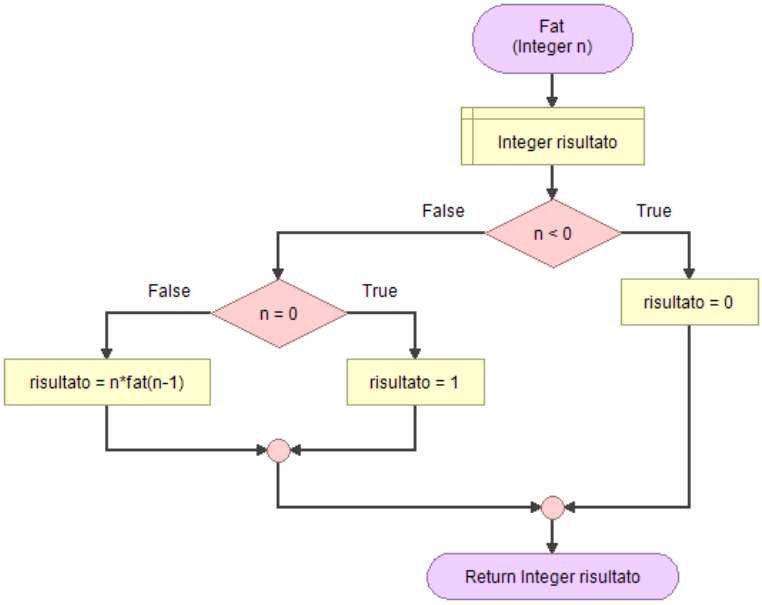

String2Array
(String stringa, String Array VettoreStringa)
Convert String to an array
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len(stringa)
i = 0 to lunghezzaStringa - 1
Next
Done
vettoreStringa[i] = MID2(stringa, i + 1, 1)
End
MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

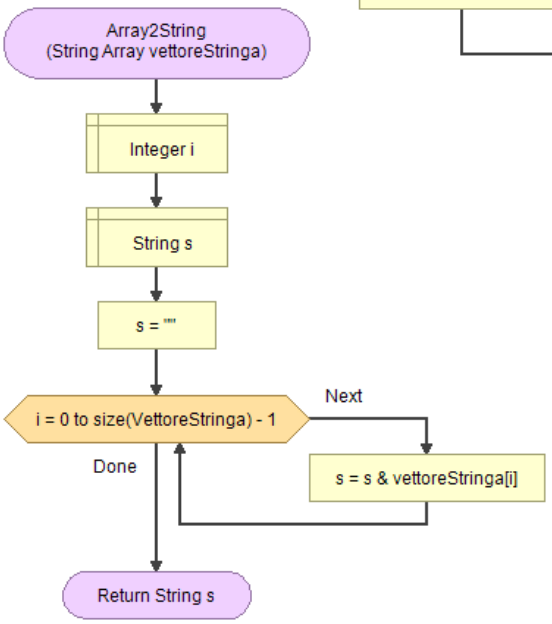
242.GAFR | Anagramma di una stringa

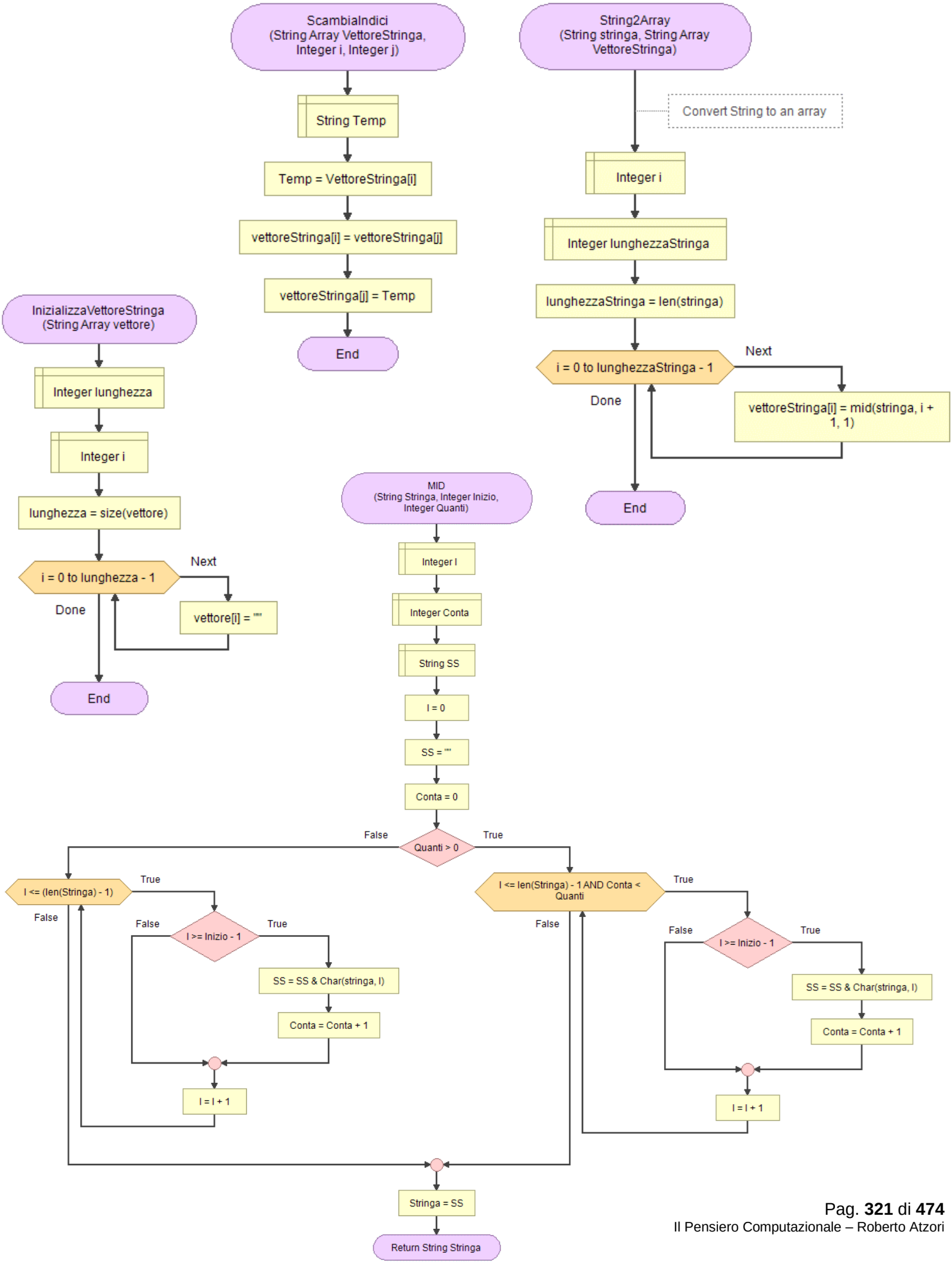
String scramble





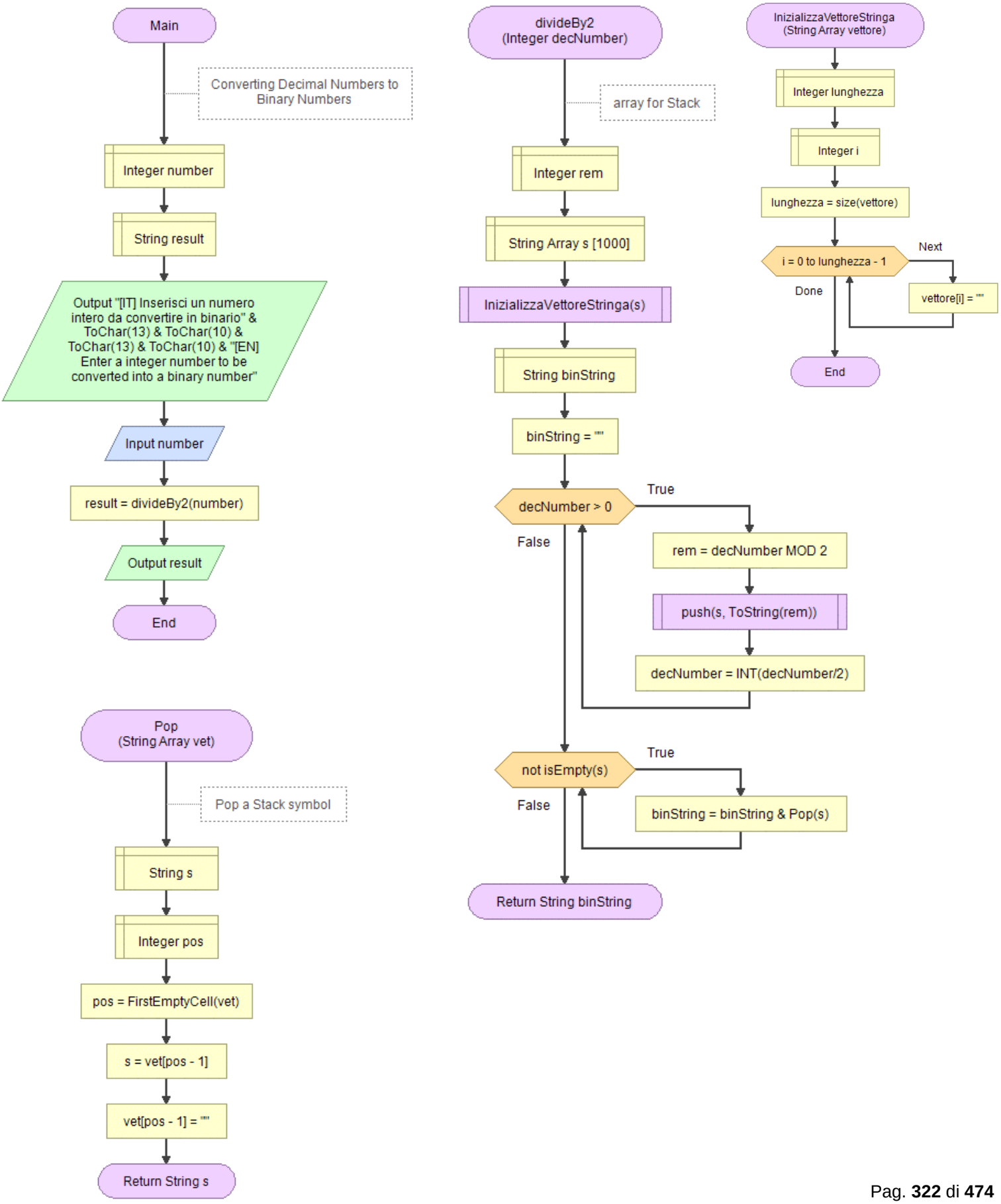


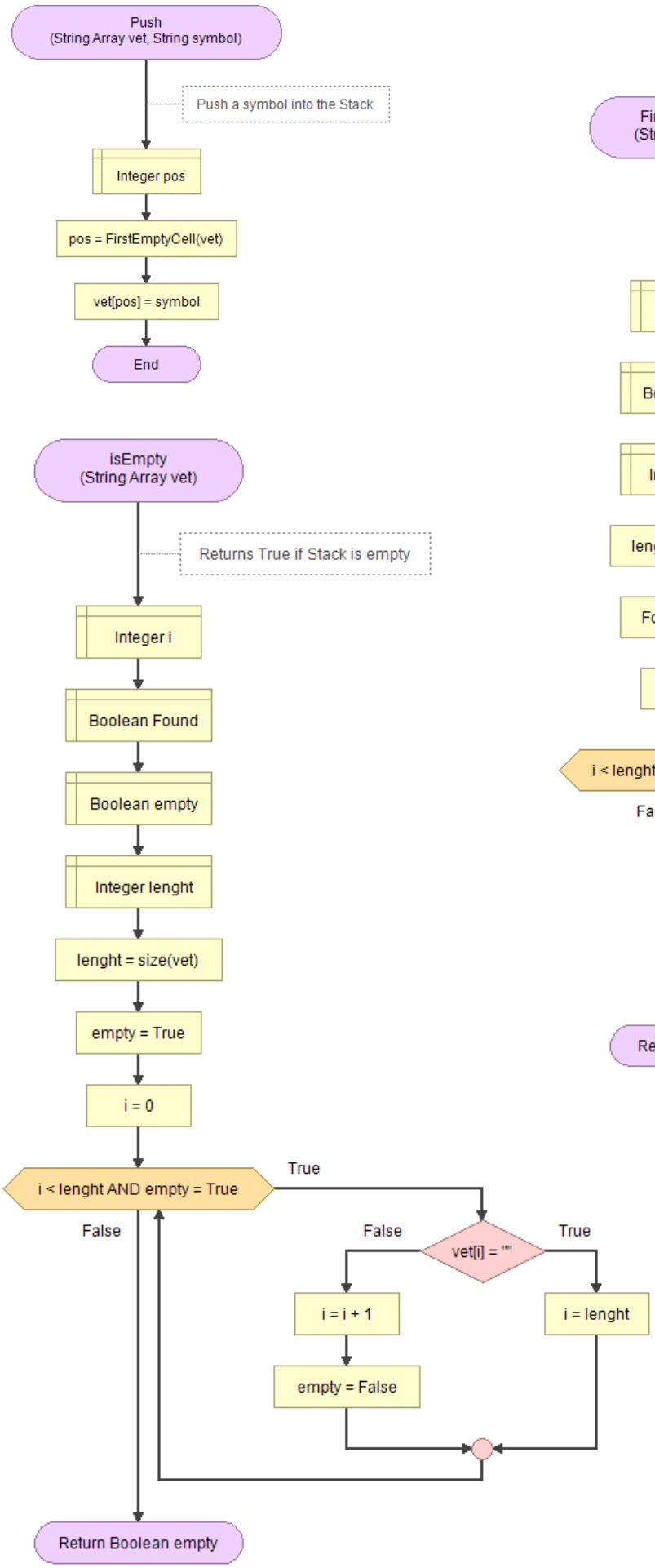


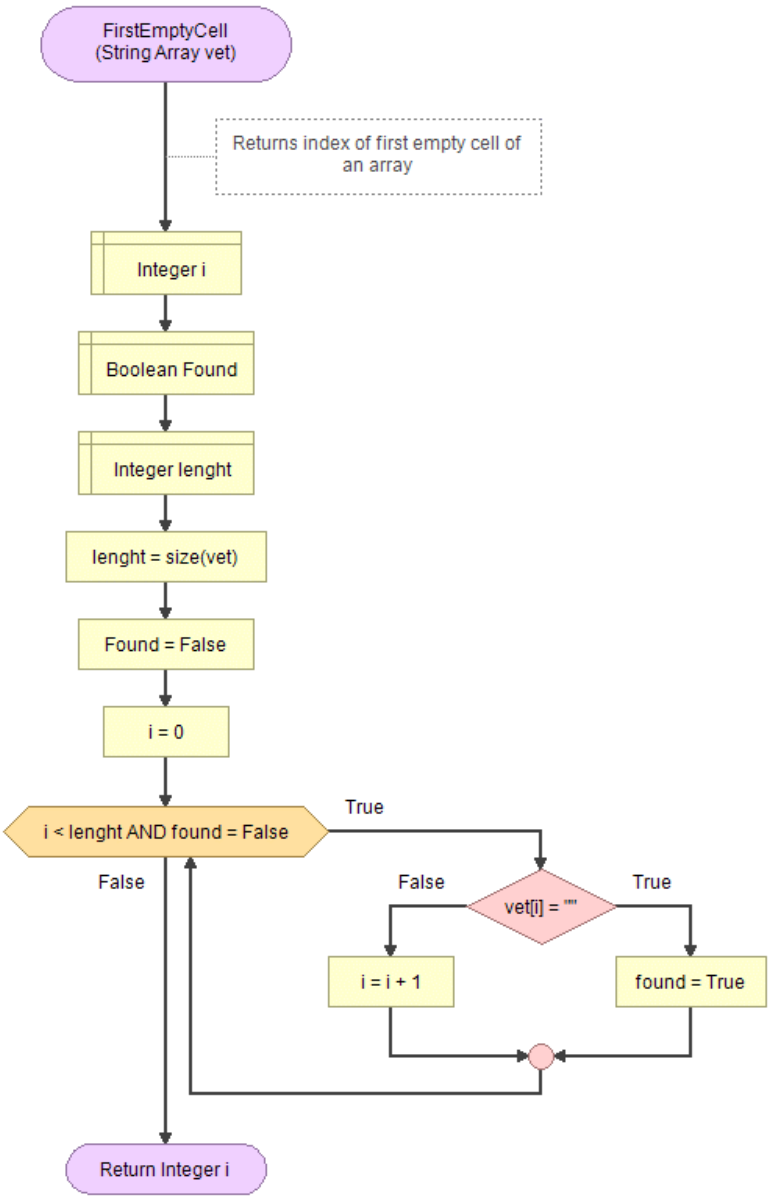

ScambiaIndici
(String Array VettoreStringa, Integer i, Integer j)
String Temp
Temp = VettoreStringa[i]
vettoreStringa[i] = vettoreStringa[j]
vettoreStringa[j] = Temp
End
String2Array
(String stringa, String Array VettoreStringa)
Convert String to an array
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len(stringa)
i = 0 to lunghezzaStringa - 1
Next
Done
vettoreStringa[i] = mid(stringa, i + 1, 1)
End
InizializzaVettoreStringa
(String Array vettore)
Integer lunghezza
Integer i
lunghezza = size(vettore)
i = 0 to lunghezza - 1
Next
Done
vettore[i] = ""
End
MID
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

243.GAF ‖ Conversione da numero decimale a numero binario

Convert Decimal Number to Binary Number

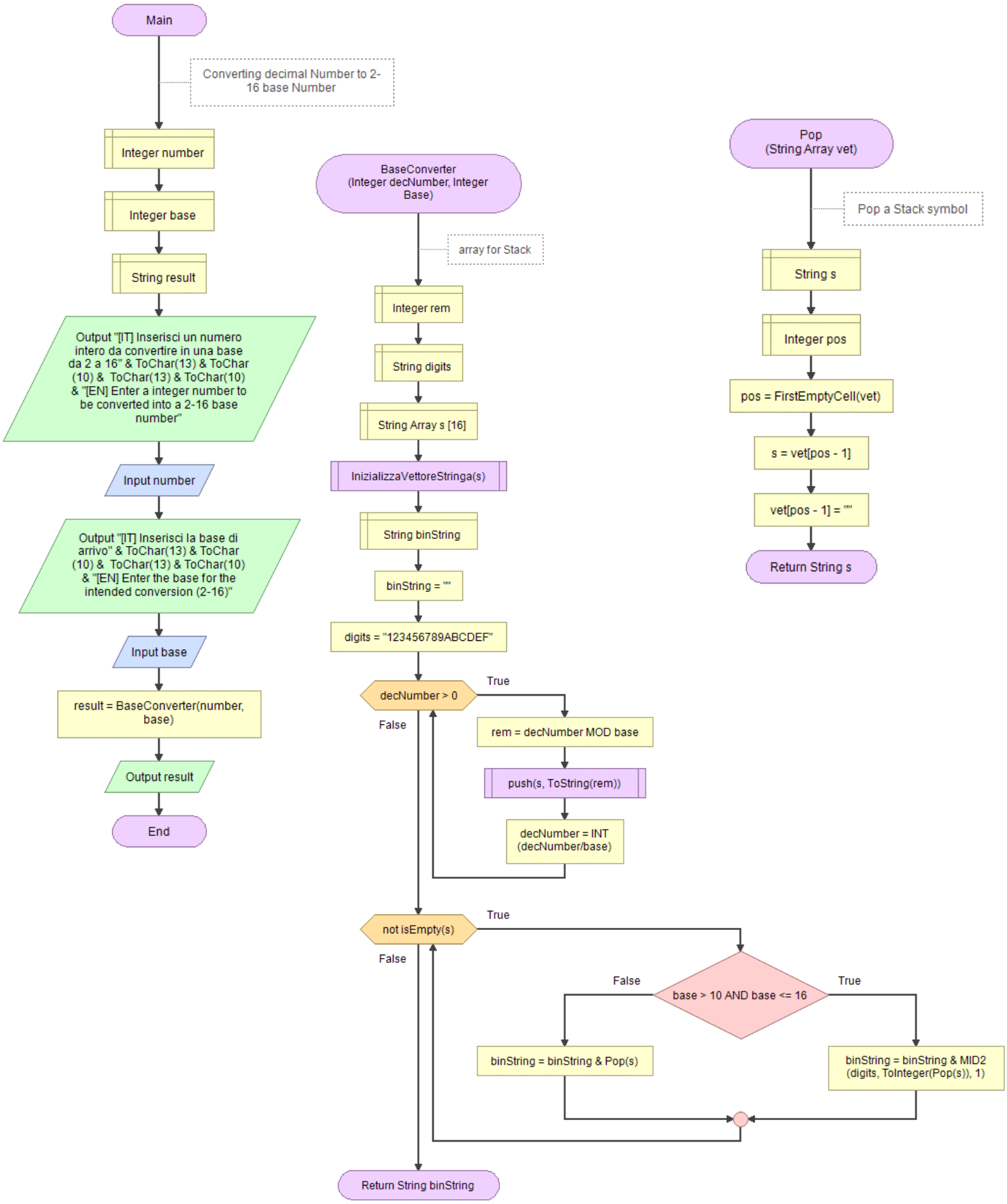


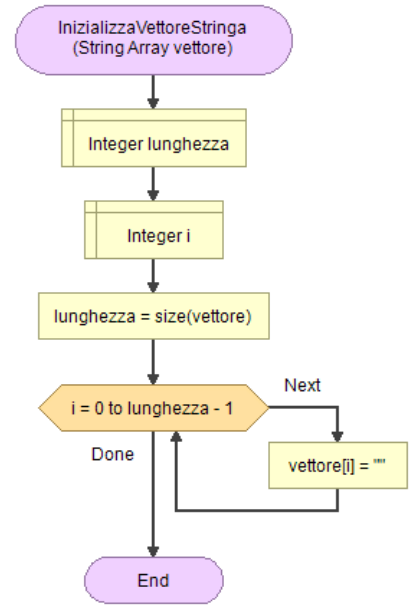

Push
(String Array vet, String symbol)
Push a symbol into the Stack
Integer pos
pos = FirstEmptyCell(vet)
vet[pos] = symbol
End
isEmpty
(String Array vet)
Returns True if Stack is empty
Integer i
Boolean Found
Boolean empty
Integer lenght
lenght = size(vet)
empty = True
i = 0
i < lenght AND empty = True
True
False
False
vet[i] = ""
True
i = i + 1
i = lenght
empty = False
Return Boolean empty

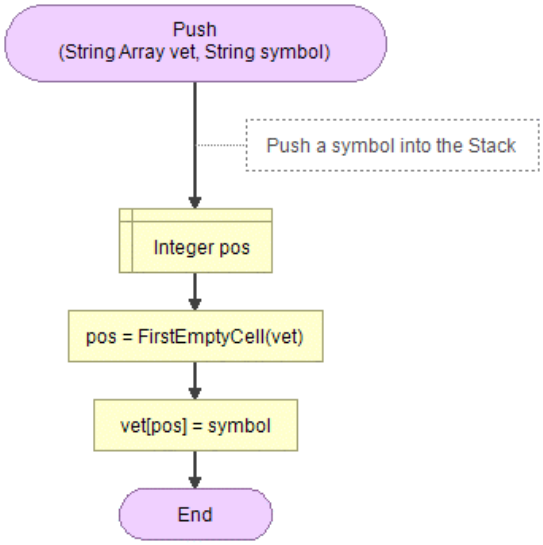

FirstEmptyCell
(String Array vet)
Returns index of first empty cell of an array
Integer i
Boolean Found
Integer lenght
lenght = size(vet)
Found = False
i = 0
i < lenght AND found = False
True
False
False
vet[i] = ""
True
i = i + 1
found = True
Return Integer i

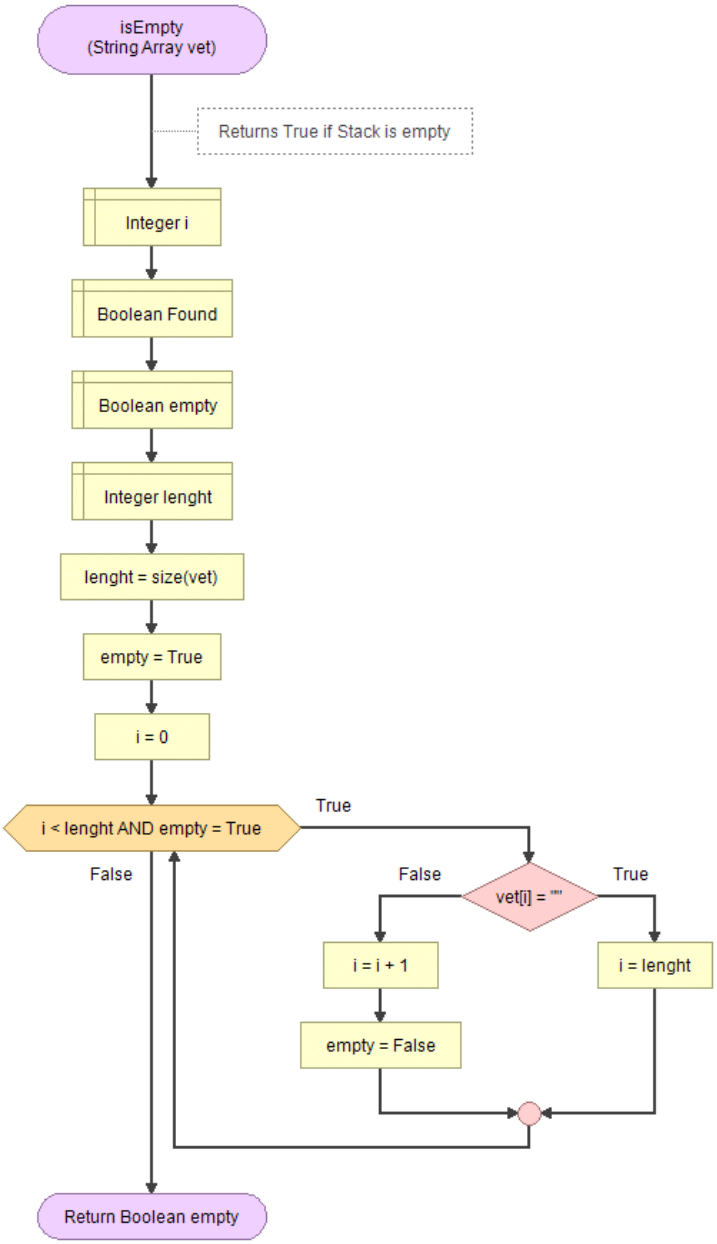
244.GAF ‖ Conversione da numero decimale a numero in base tra 2 e 16

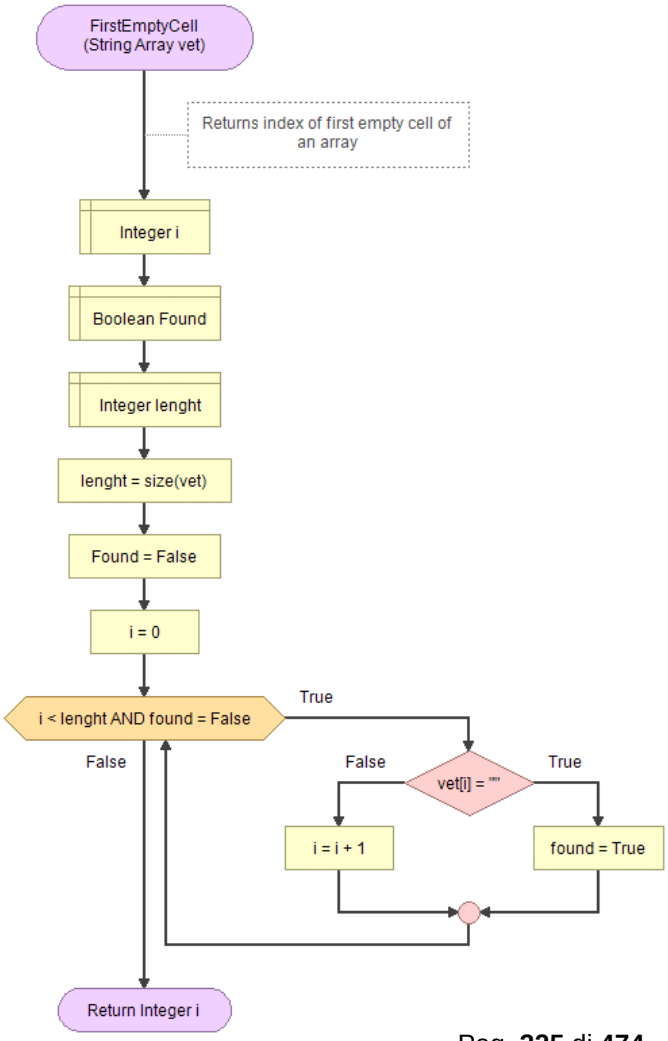
Converting Decimal Numbers to Numbers with 2-16 base

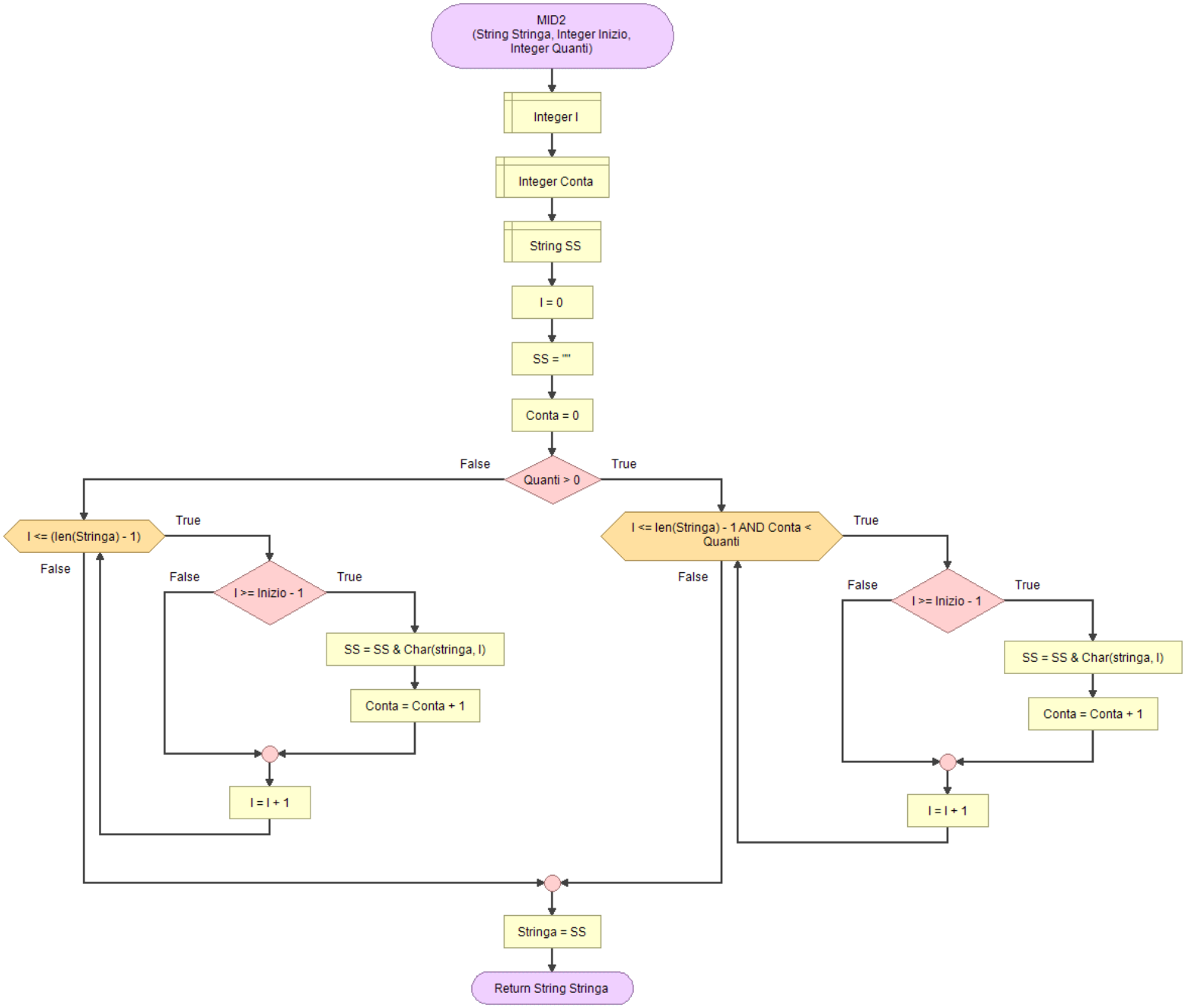



InizializzaVettoreStringa
(String Array vettore)
Integer lunghezza
Integer i
lunghezza = size(vettore)
i = 0 to lunghezza - 1
Next
vettore[i] = ""
Done
End


Push
(String Array vet, String symbol)
Push a symbol into the Stack
Integer pos
pos = FirstEmptyCell(vet)
vet[pos] = symbol
End

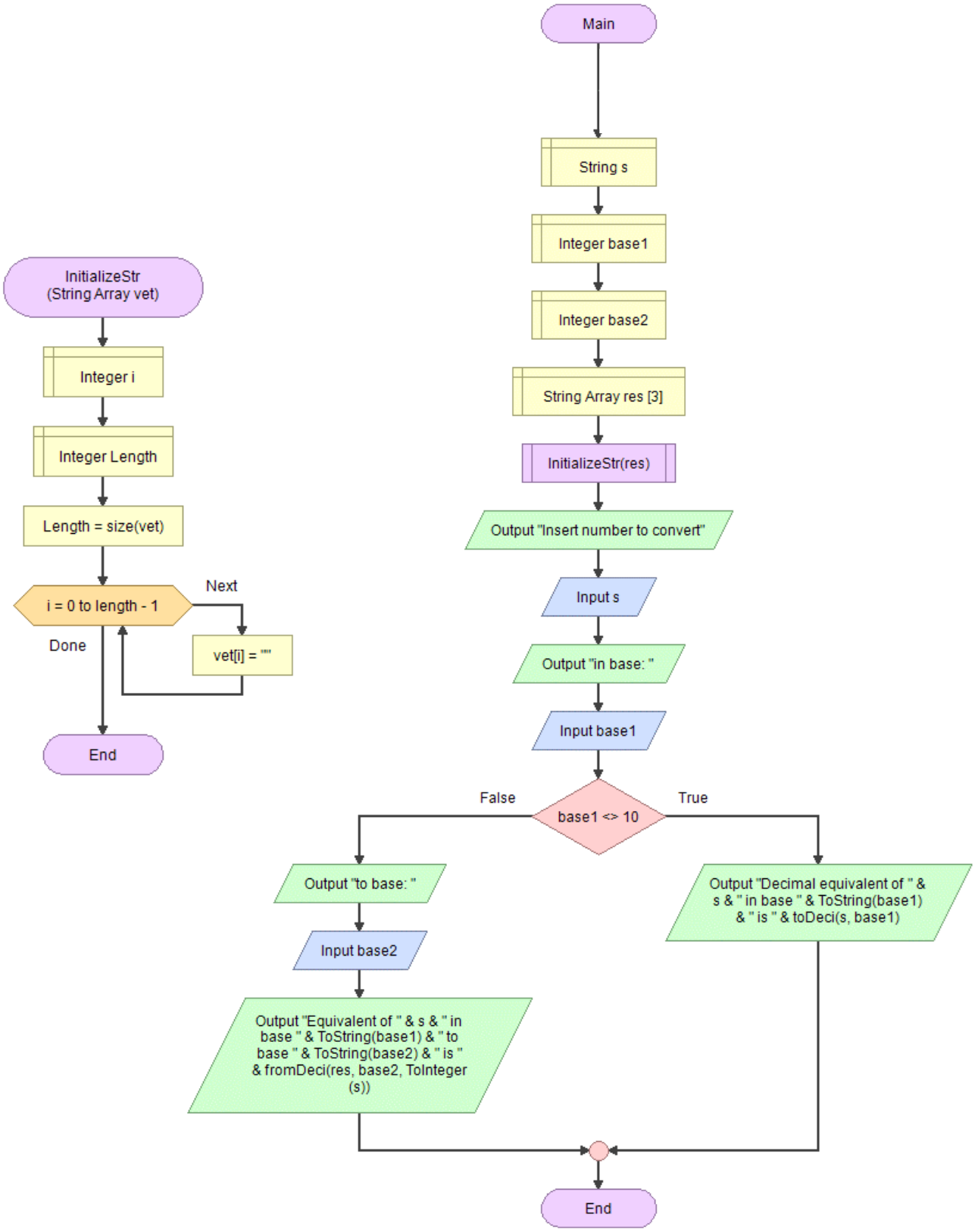

isEmpty
(String Array vet)
Returns True if Stack is empty
Integer i
Boolean Found
Boolean empty
Integer lenght
lenght = size(vet)
empty = True
i = 0
i < lenght AND empty = True
True
False
vet[i] = ""
False
True
i = i + 1
i = lenght
empty = False
Return Boolean empty

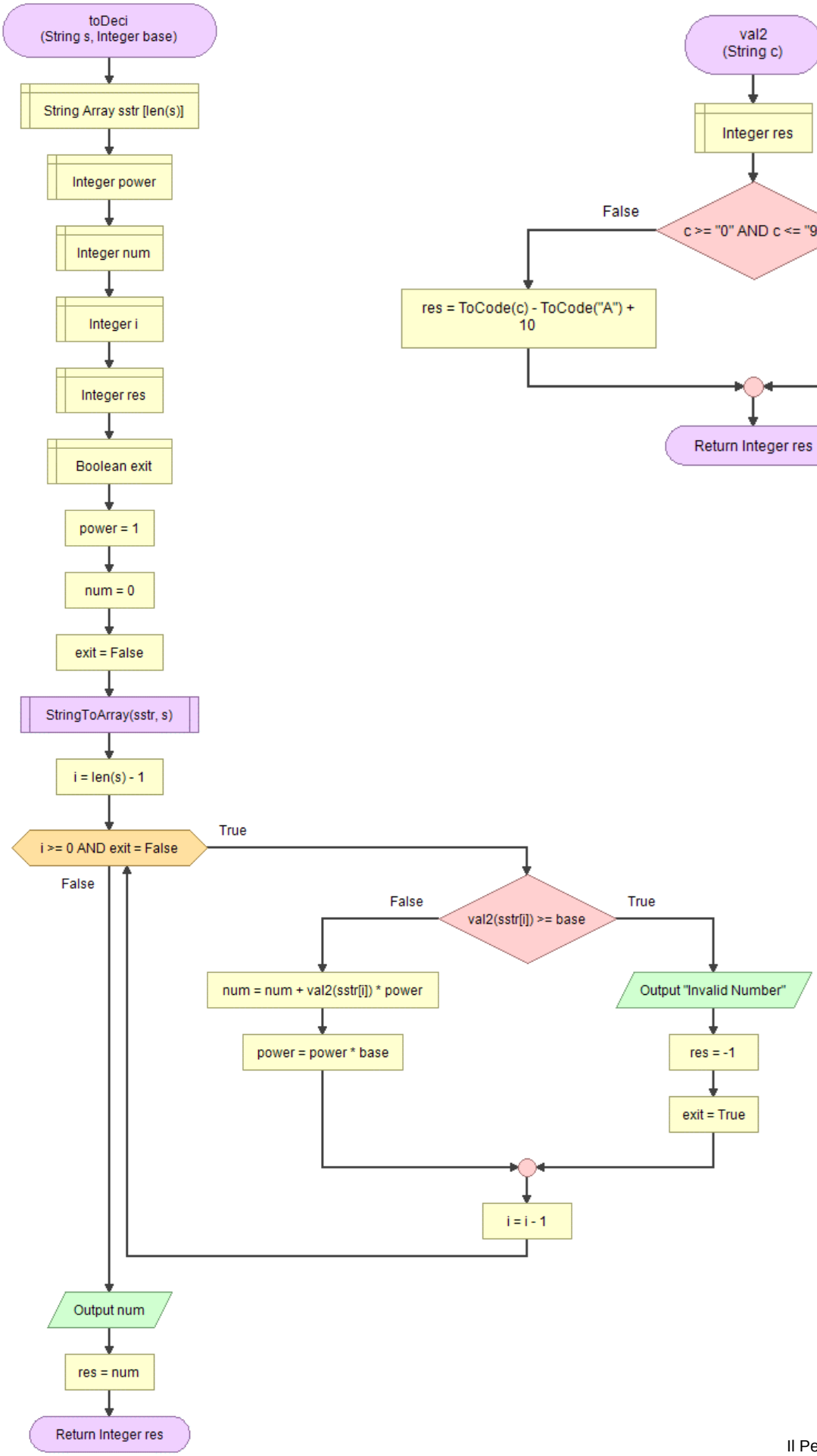

FirstEmptyCell
(String Array vet)
Returns index of first empty cell of an array
Integer i
Boolean Found
Integer lenght
lenght = size(vet)
Found = False
i = 0
i < lenght AND found = False
True
False
vet[i] = ""
False
True
i = i + 1
found = True
Return Integer i

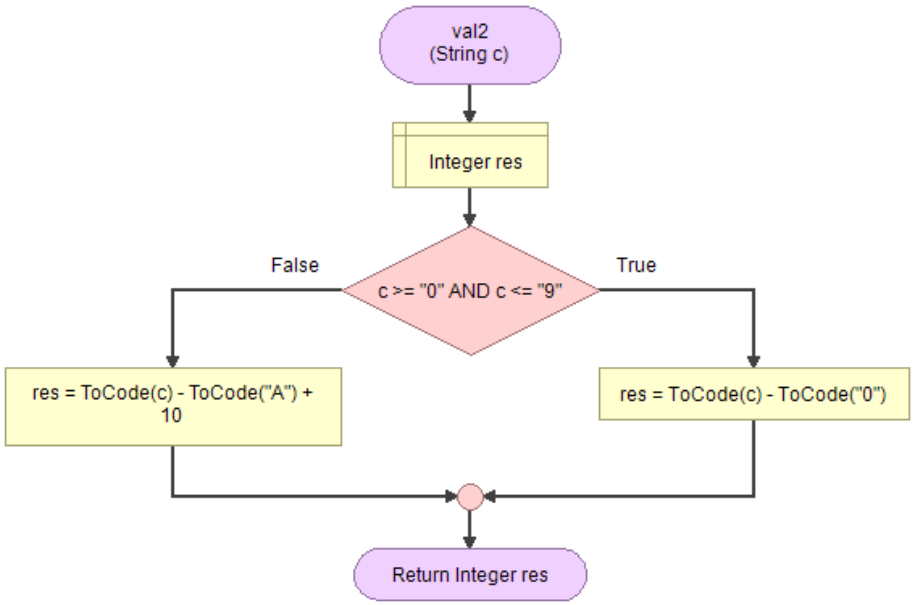

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

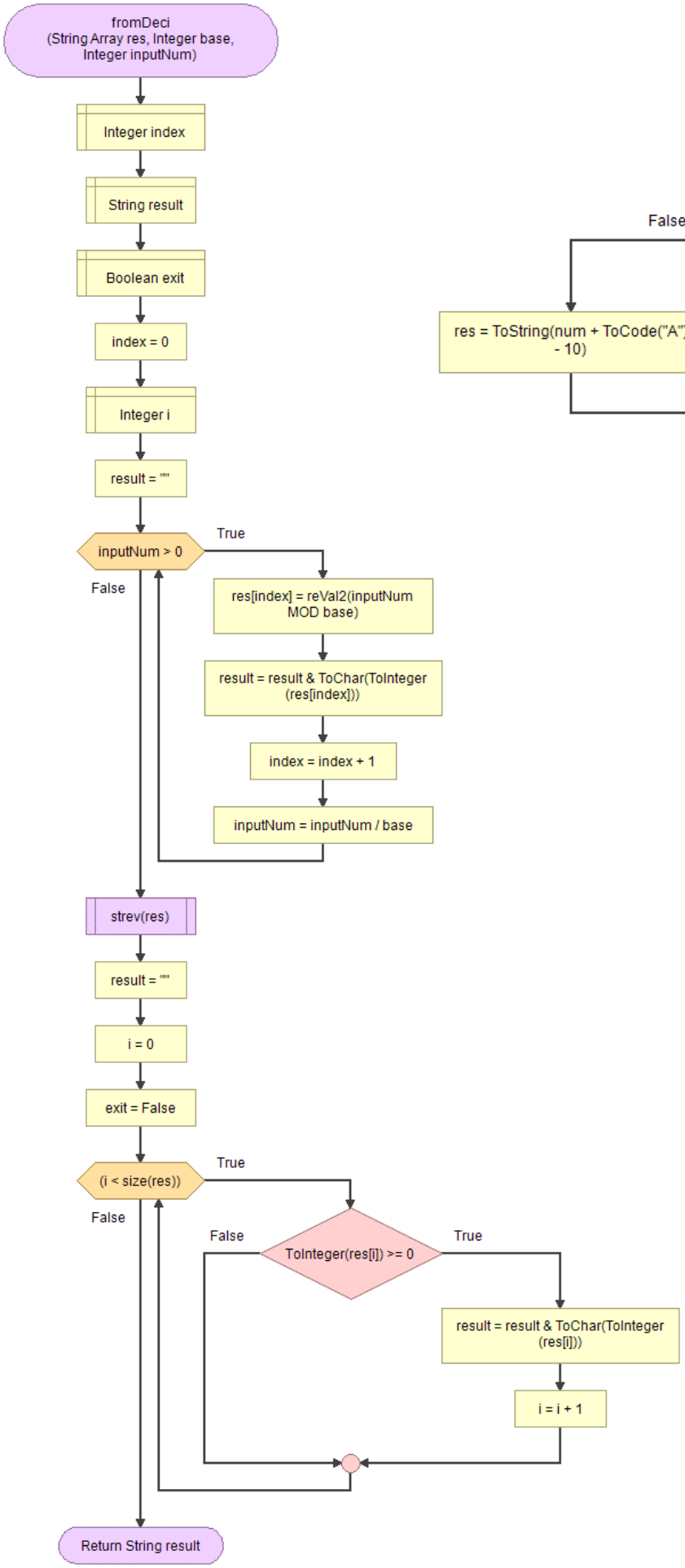
245.GSAF | Conversione da numero in qualsiasi base in numero decimale e viceversa

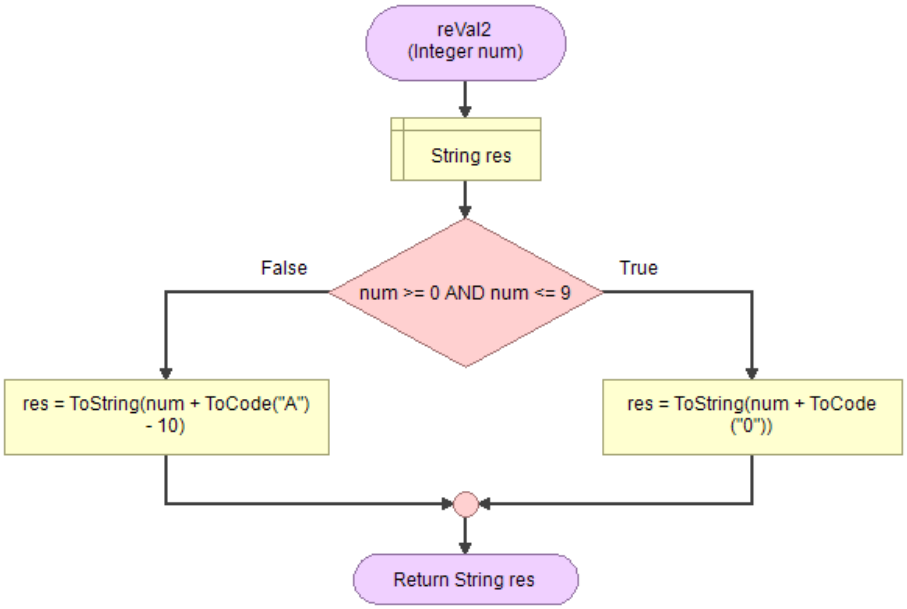
Convert any number from base to base (base from 2 to 36)

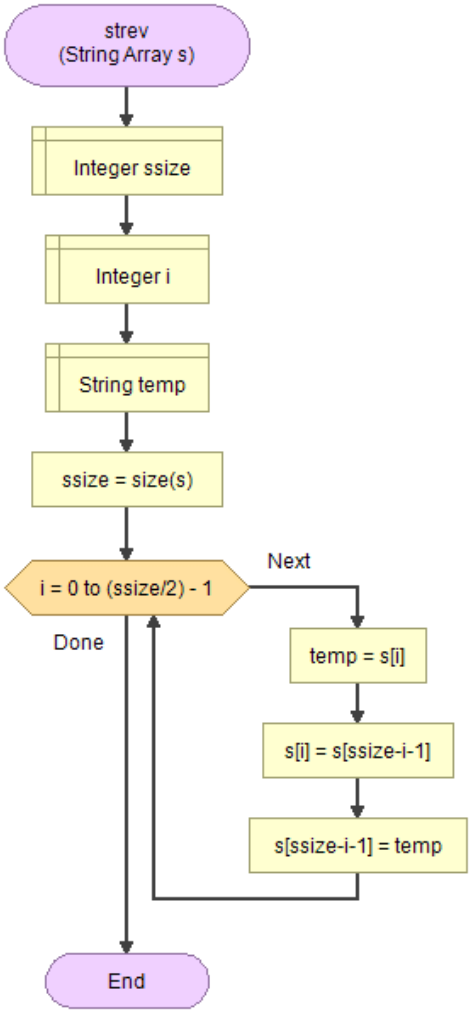


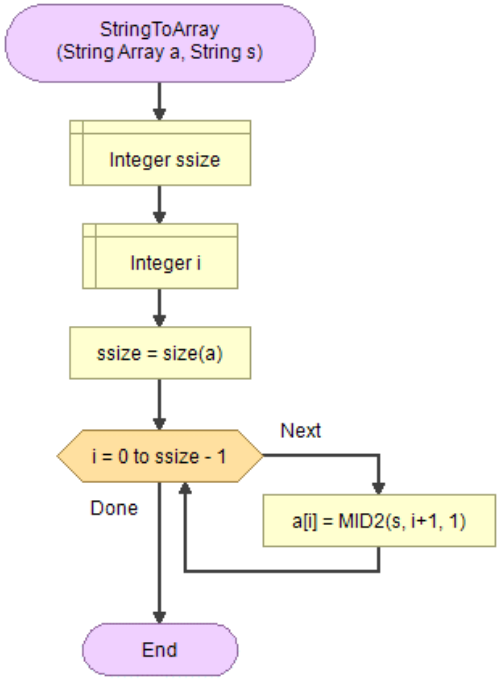

toDeci
(String s, Integer base)
String Array sstr [len(s)]
Integer power
Integer num
Integer i
Integer res
Boolean exit
power = 1
num = 0
exit = False
StringToArray(sstr, s)
i = len(s) - 1
i >= 0 AND exit = False
True
False
val2(sstr[i]) >= base
False
True
num = num + val2(sstr[i]) * power
power = power * base
Output "Invalid Number"
res = -1
exit = True
i = i - 1
Output num
res = num
Return Integer res

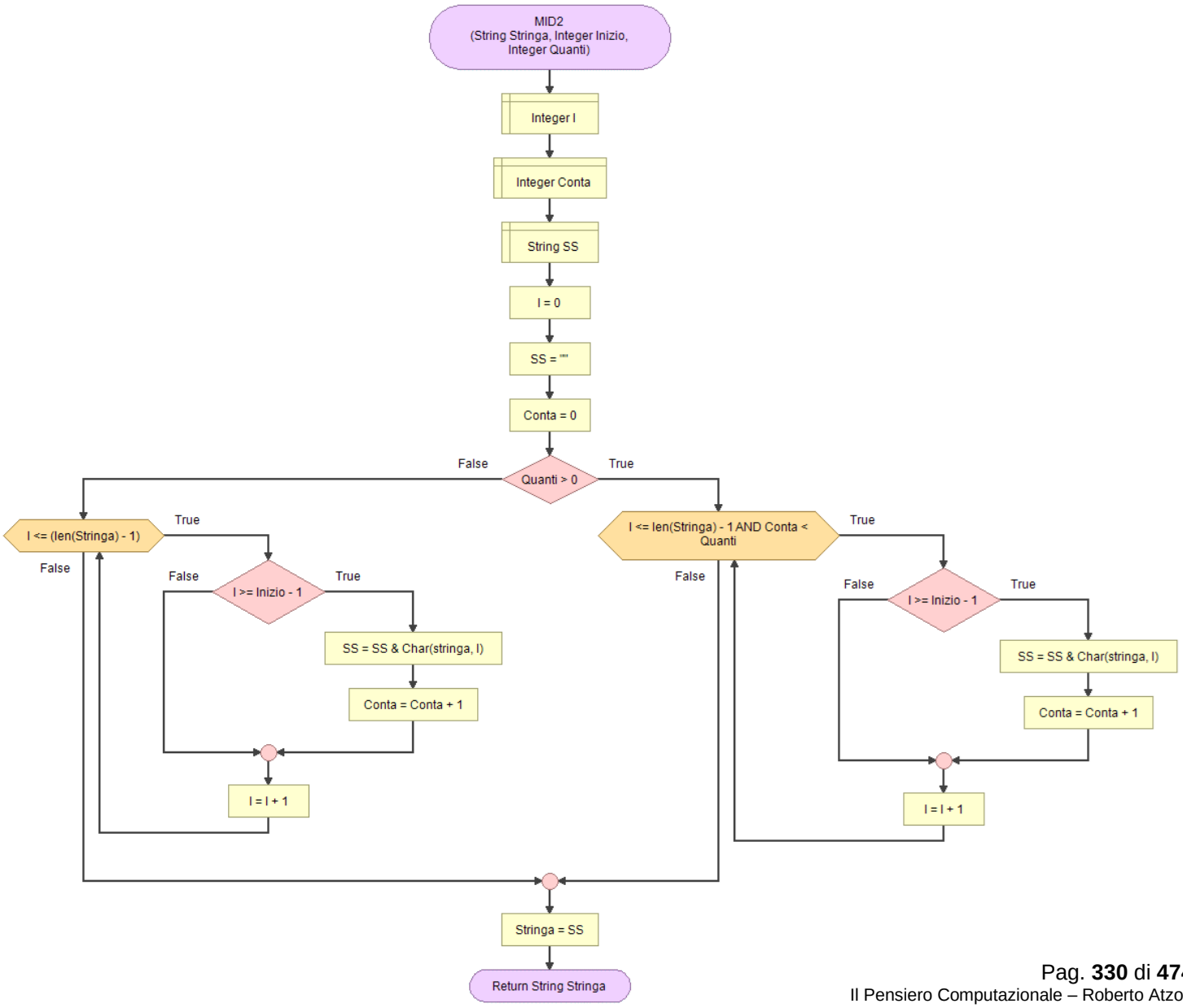

val2
(String c)
Integer res
False
c >= "0" AND c <= "9"
True
res = ToCode(c) - ToCode("A") + 10
res = ToCode(c) - ToCode("0")
Return Integer res


fromDeci
(String Array res, Integer base, Integer inputNum)
Integer index
String result
Boolean exit
index = 0
Integer i
result = ""
inputNum > 0
True
False
res[index] = reVal2(inputNum MOD base)
result = result & ToChar(ToInteger (res[index]))
index = index + 1
inputNum = inputNum / base
strev(res)
result = ""
i = 0
exit = False
(i < size(res))
True
False
ToInteger(res[i]) >= 0
False
True
result = result & ToChar(ToInteger (res[i]))
i = i + 1
Return String result


reVal2
(Integer num)
String res
num >= 0 AND num <= 9
False
True
res = ToString(num + ToCode("A") - 10)
res = ToString(num + ToCode ("0"))
Return String res

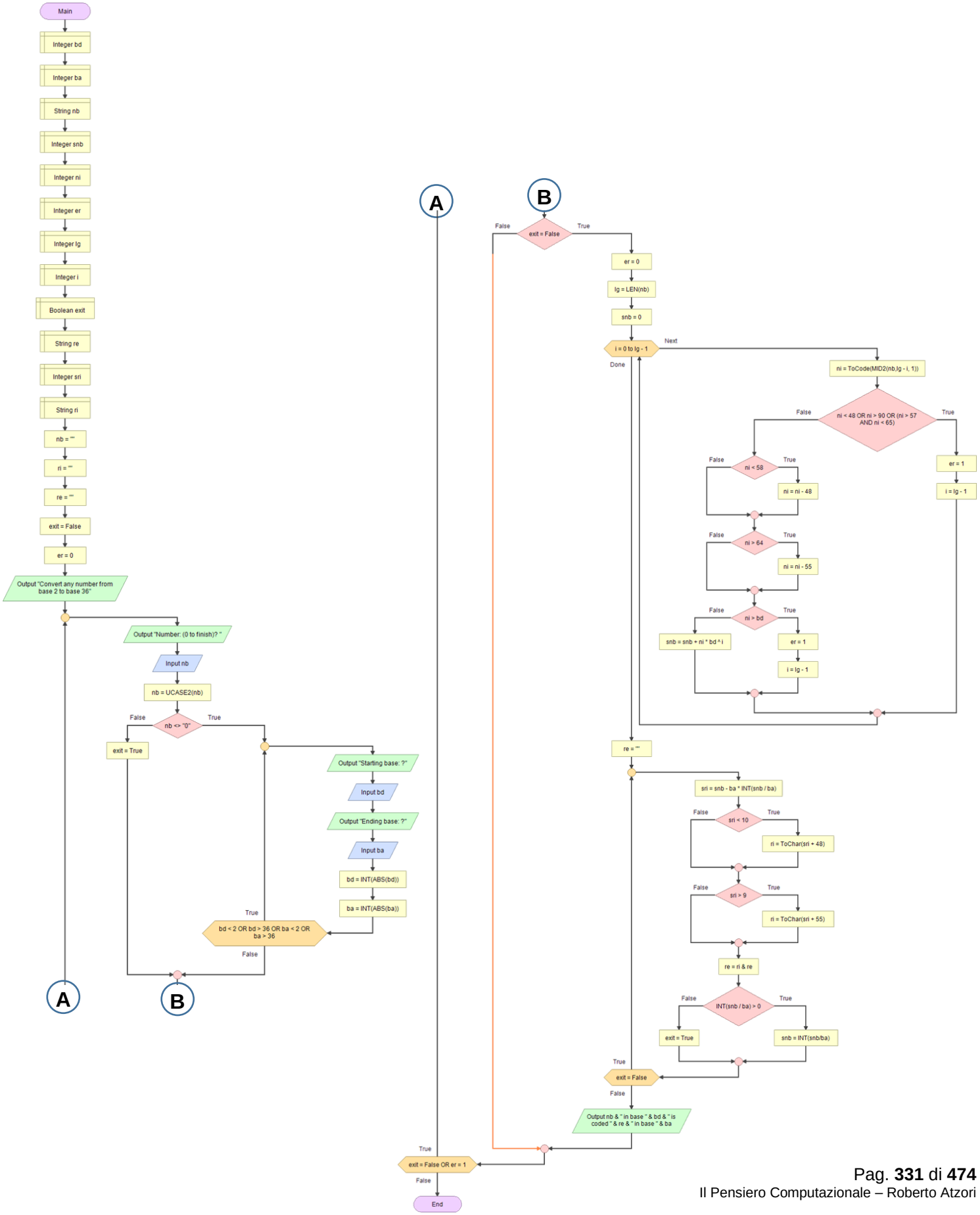

strev
(String Array s)
Integer ssize
Integer i
String temp
ssize = size(s)
i = 0 to (ssize/2) - 1
Next
Done
temp = s[i]
s[i] = s[ssize-i-1]
s[ssize-i-1] = temp
End

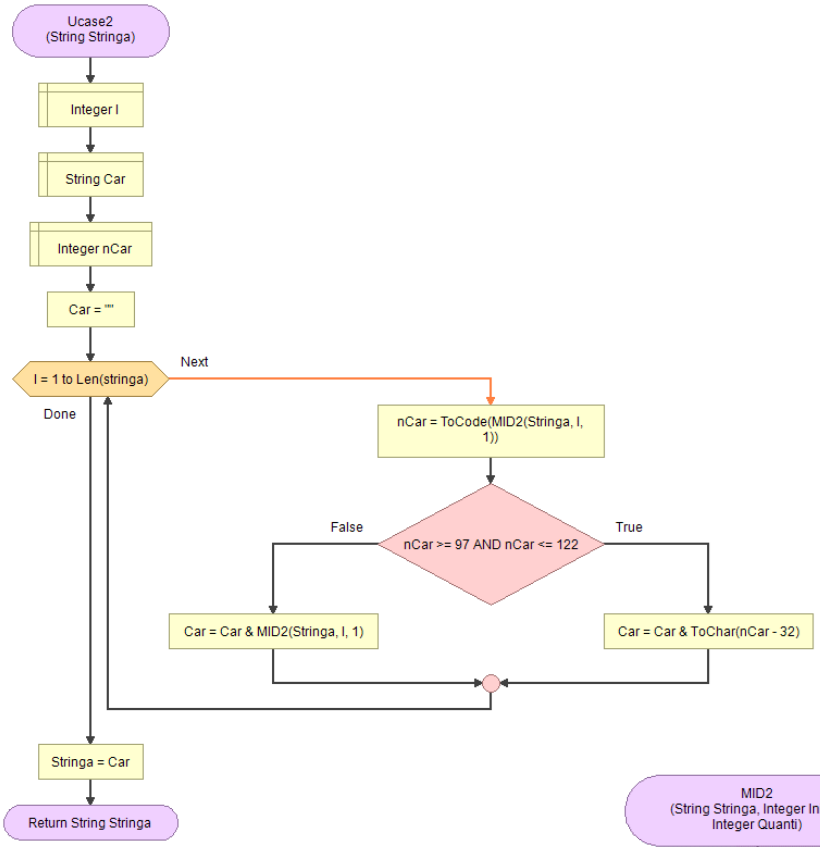

StringToArray
(String Array a, String s)
Integer ssize
Integer i
ssize = size(a)
i = 0 to ssize - 1
Next
Done
a[i] = MID2(s, i+1, 1)
End

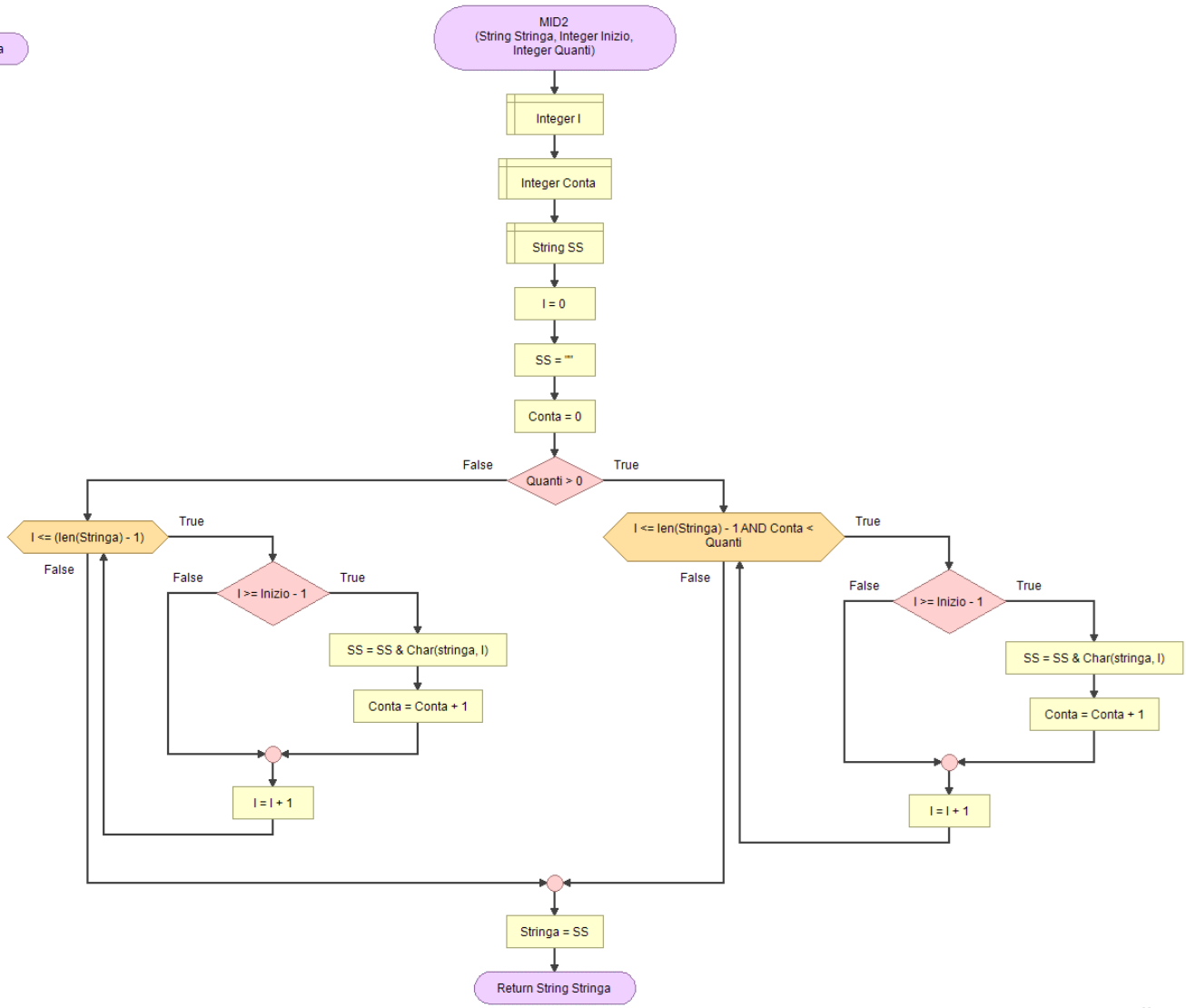

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

246.GS | Conversione da numero in qualsiasi base in numero decimale e viceversa ver. 2

Convert any number from base to base (base from 2 to 36) ver 2

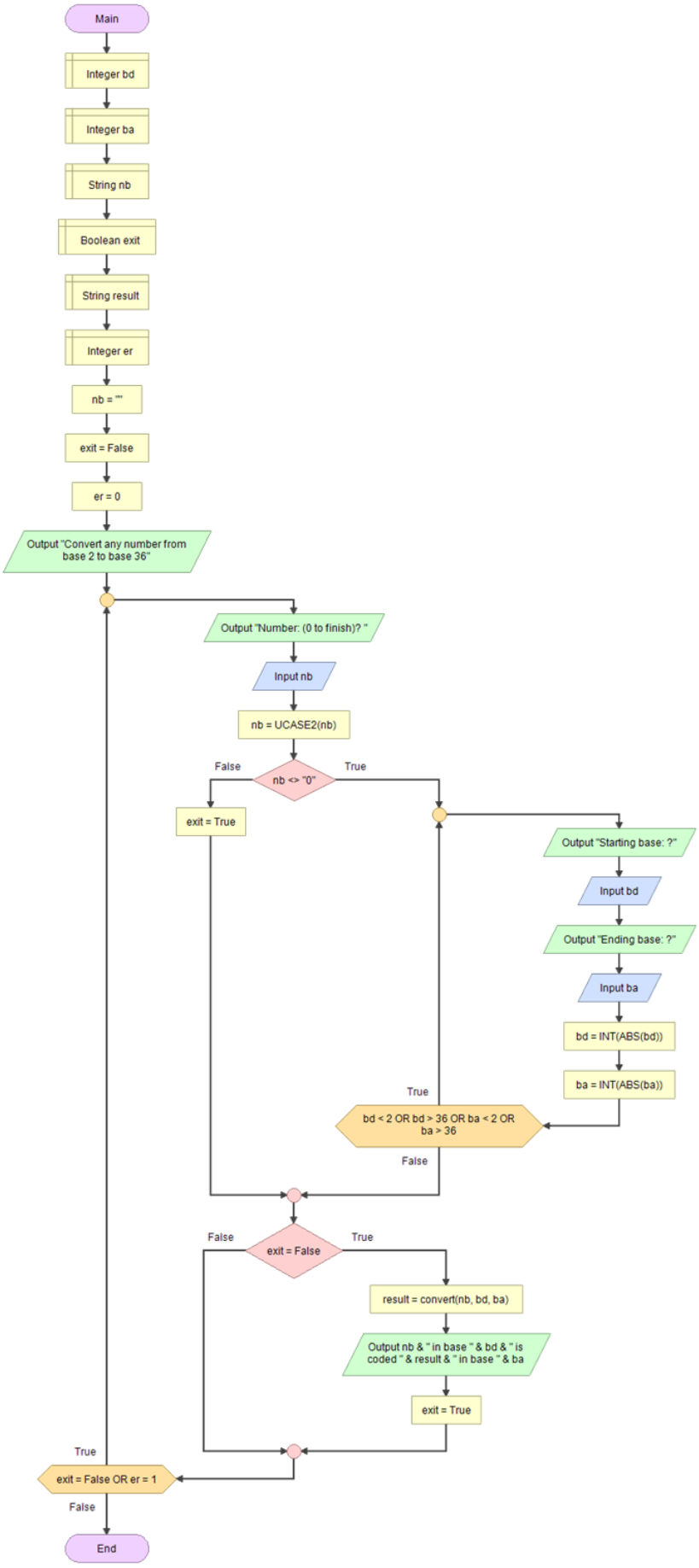


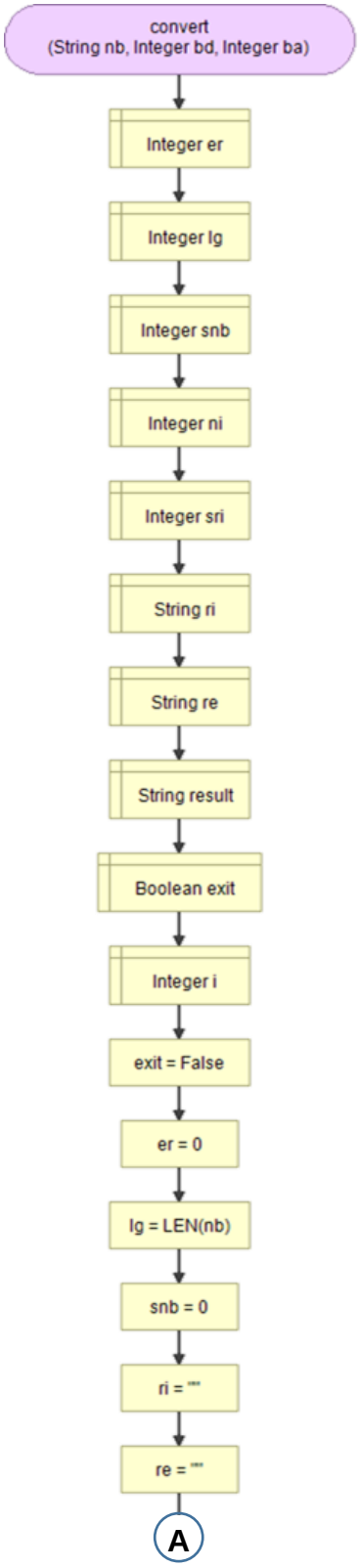

Ucase2
(String Stringa)
Integer I
String Car
Integer nCar
Car = ""
I = 1 to Len(stringa)
Next
Done
nCar = ToCode(MID2(Stringa, I, 1))
nCar >= 97 AND nCar <= 122
False
True
Car = Car & MID2(Stringa, I, 1)
Car = Car & ToChar(nCar - 32)
Stringa = Car
Return String Stringa

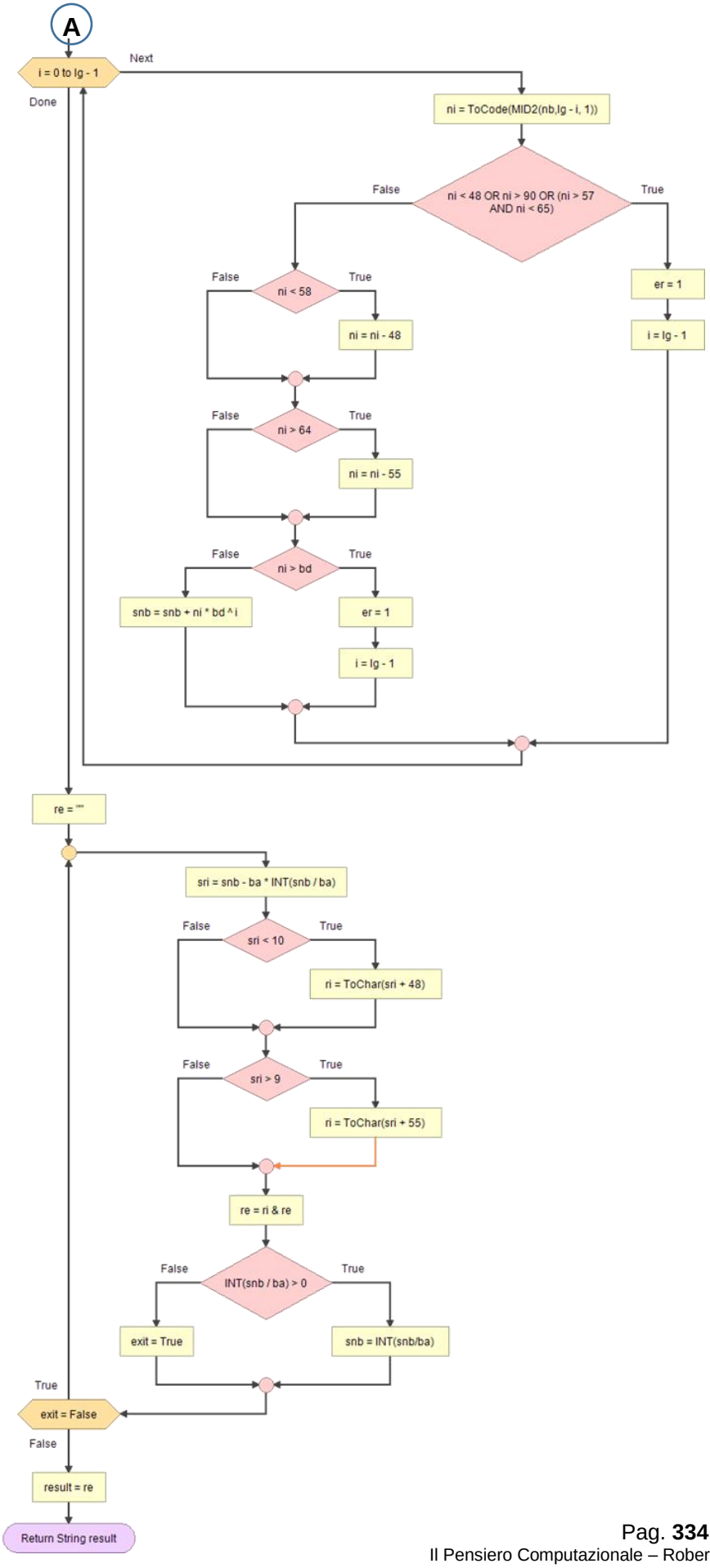

MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

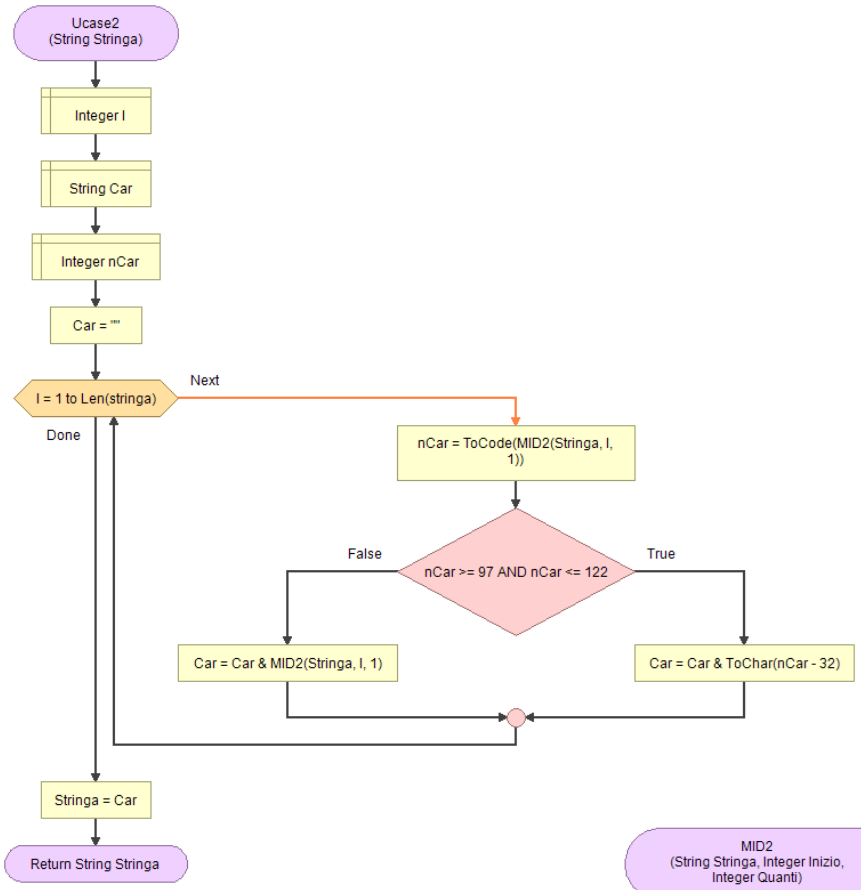
247.GSF ‖ Conversione da numero in qualsiasi base in numero decimale e viceversa tramite funzione

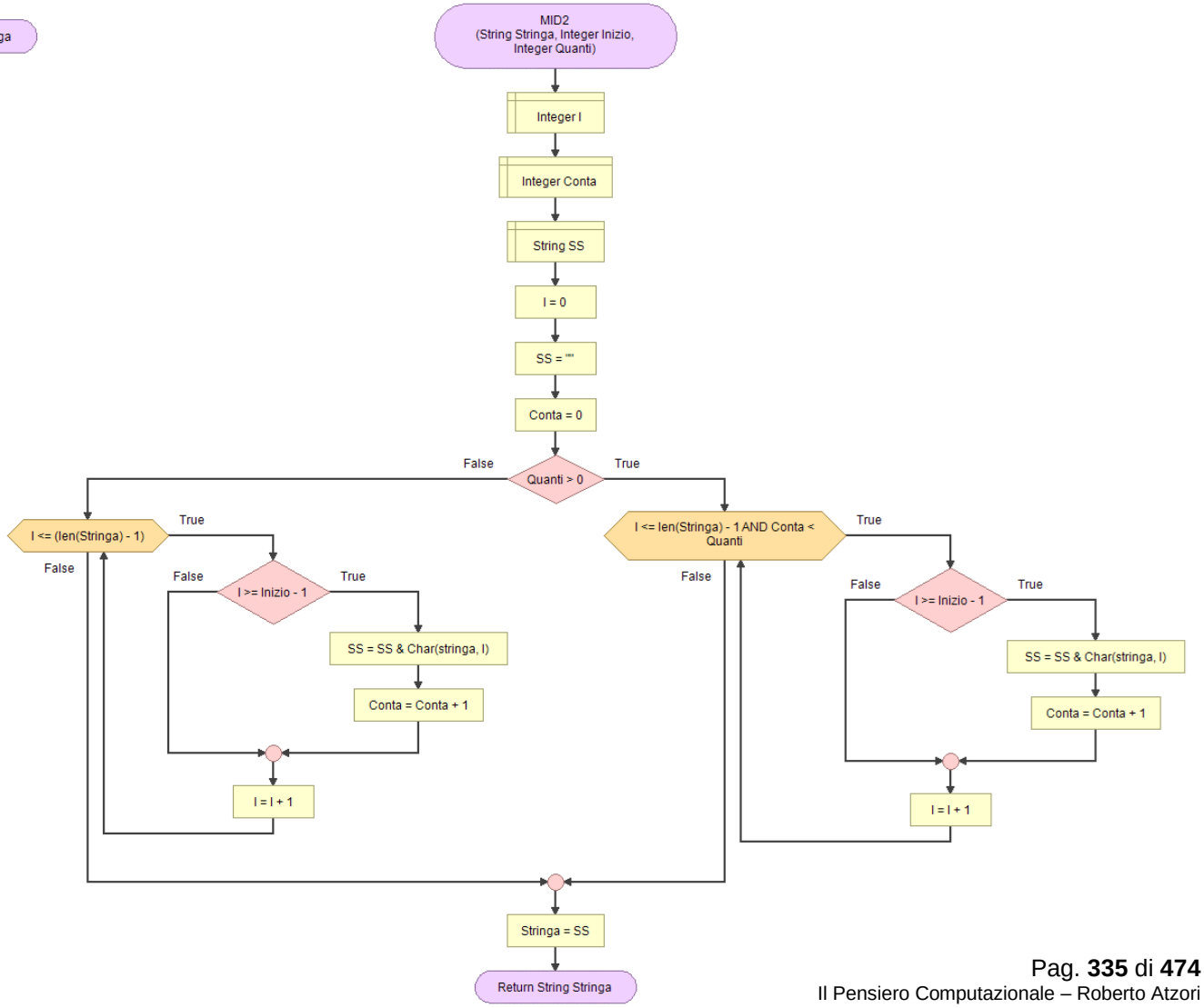
Convert any number from base to base (base from 2 to 36) by function




convert
(String nb, Integer bd, Integer ba)
Integer er
Integer lg
Integer snb
Integer ni
Integer sri
String ri
String re
String result
Boolean exit
Integer i
exit = False
er = 0
lg = LEN(nb)
snb = 0
ri = ""
re = ""
A

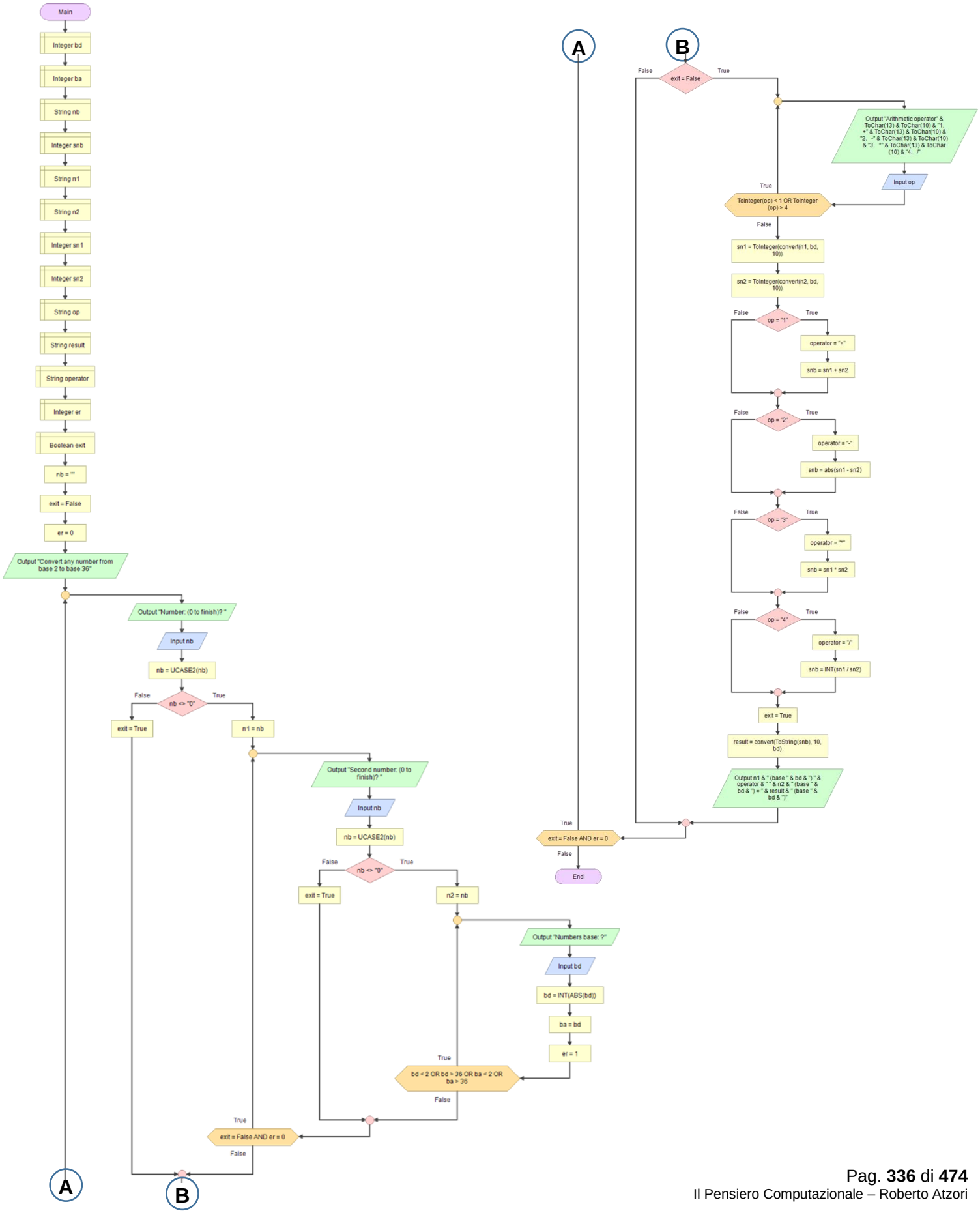

A
i = 0 to lg - 1
Next
Done
ni = ToCode(MID2(nb,lg - i, 1))
ni < 48 OR ni > 90 OR (ni > 57 AND ni < 65)
False
True
er = 1
i = lg - 1
ni < 58
ni = ni - 48
ni > 64
ni = ni - 55
ni > bd
snb = snb + ni * bd ^ i
er = 1
i = lg - 1
re = ""
sri = snb - ba * INT(snb / ba)
sri < 10
ri = ToChar(sri + 48)
sri > 9
ri = ToChar(sri + 55)
re = ri & re
INT(snb / ba) > 0
exit = True
snb = INT(snb/ba)
exit = False
result = re
Return String result

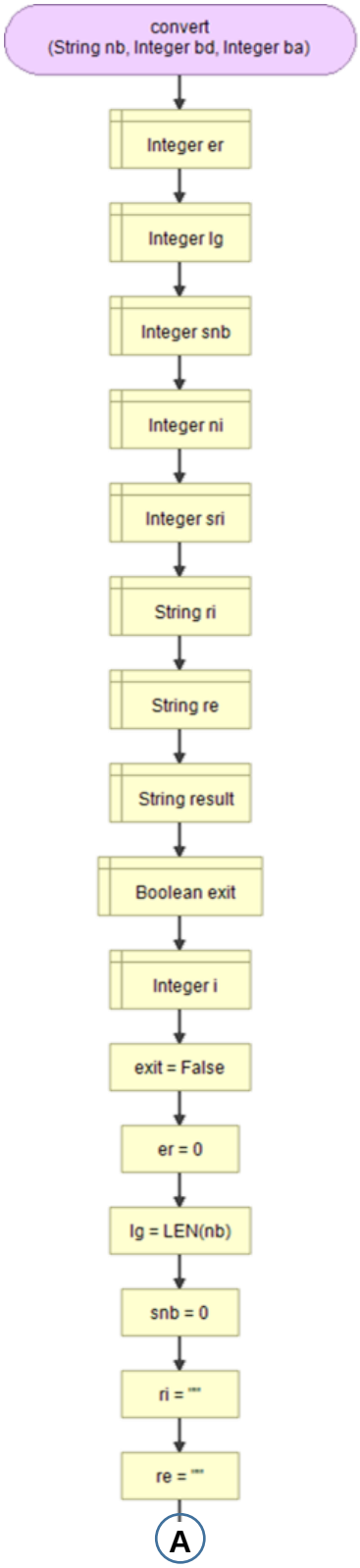

Ucase2
(String Stringa)
Integer I
String Car
Integer nCar
Car = ""
I = 1 to Len(stringa)
Next
Done
nCar = ToCode(MID2(Stringa, I, 1))
False
nCar >= 97 AND nCar <= 122
True
Car = Car & MID2(Stringa, I, 1)
Car = Car & ToChar(nCar - 32)
Stringa = Car
Return String Stringa

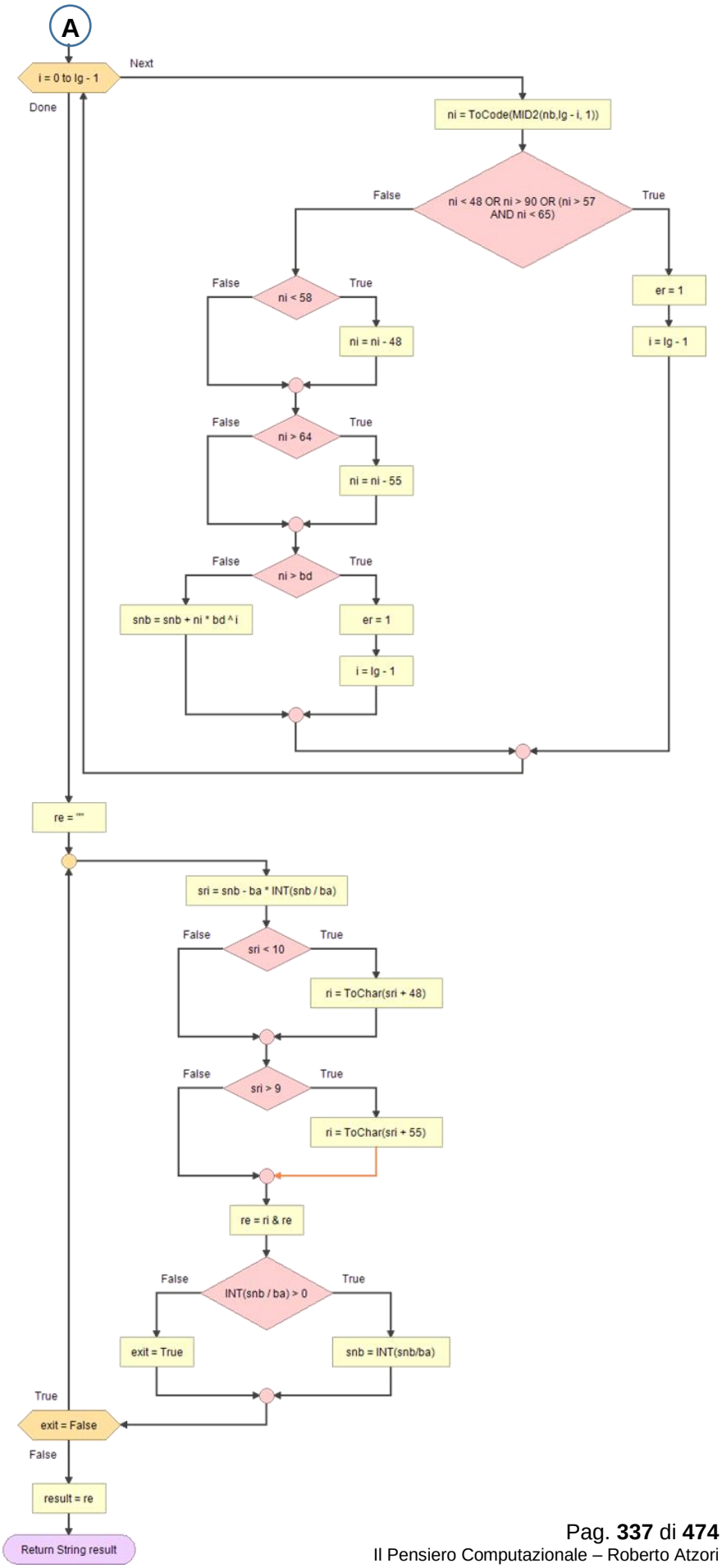

MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
False
Quanti > 0
True
I <= (len(Stringa) - 1)
True
False
False
I >= Inizio - 1
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
False
I >= Inizio - 1
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

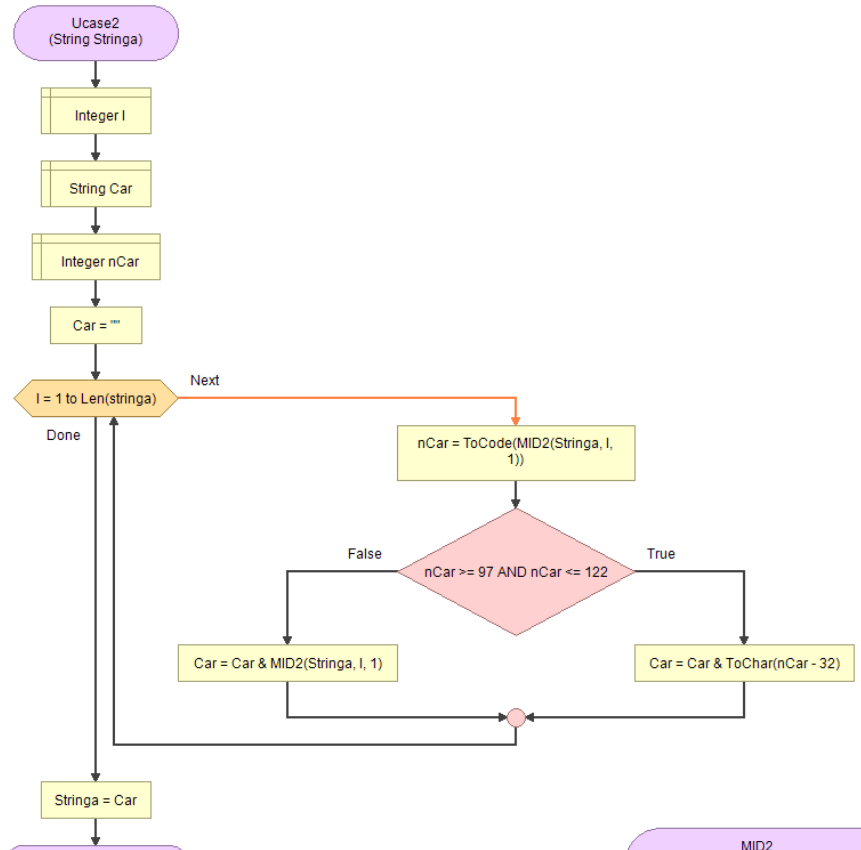
248.GSAF | Operazioni aritmetiche tra due numeri in qualsiasi base (da 2 a 36)

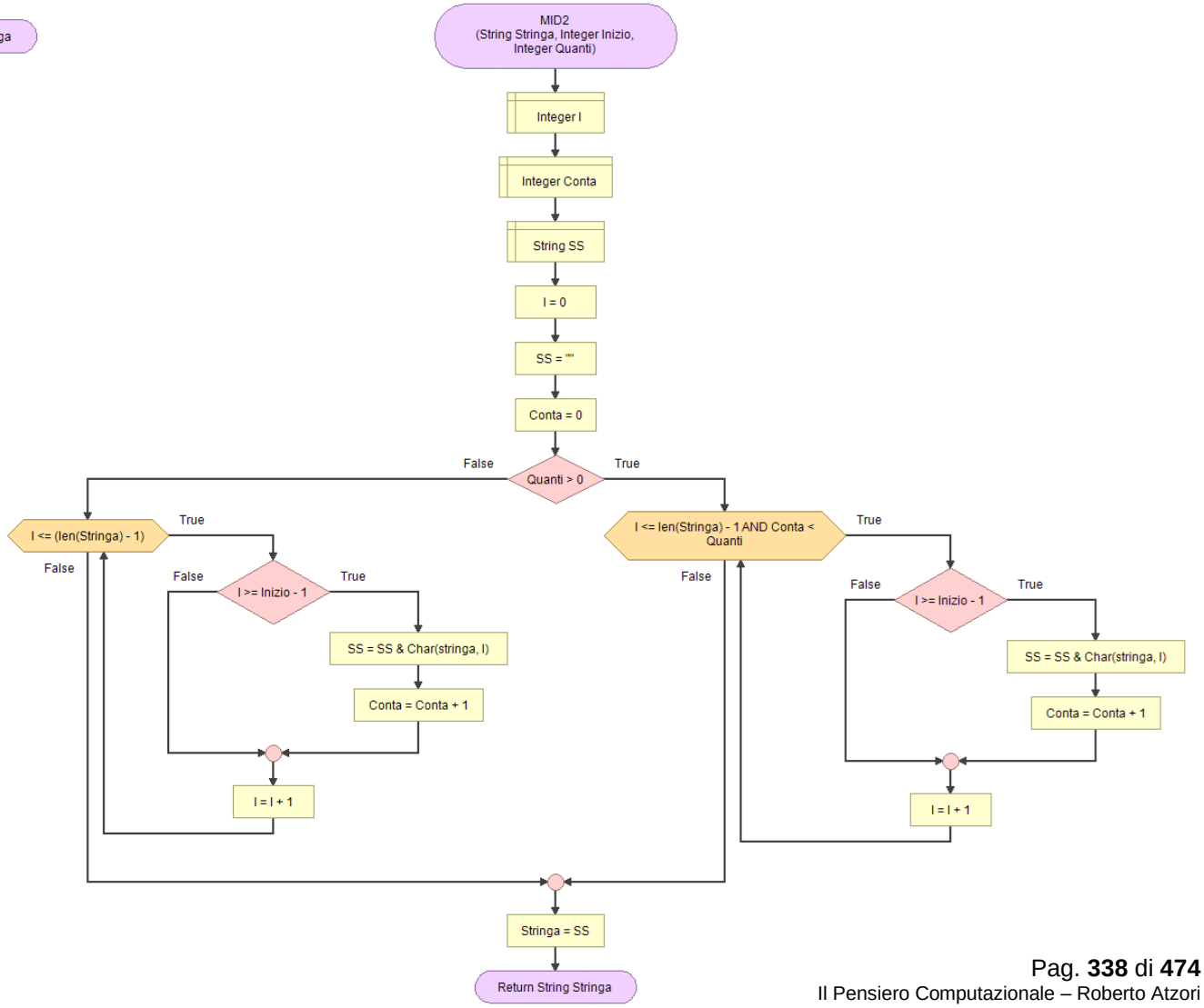
Arithmetic operations with 2 numbers of any base (base from 2 to 36)




convert
(String nb, Integer bd, Integer ba)
Integer er
Integer lg
Integer snb
Integer ni
Integer sri
String ri
String re
String result
Boolean exit
Integer i
exit = False
er = 0
lg = LEN(nb)
snb = 0
ri = ""
re = ""
A

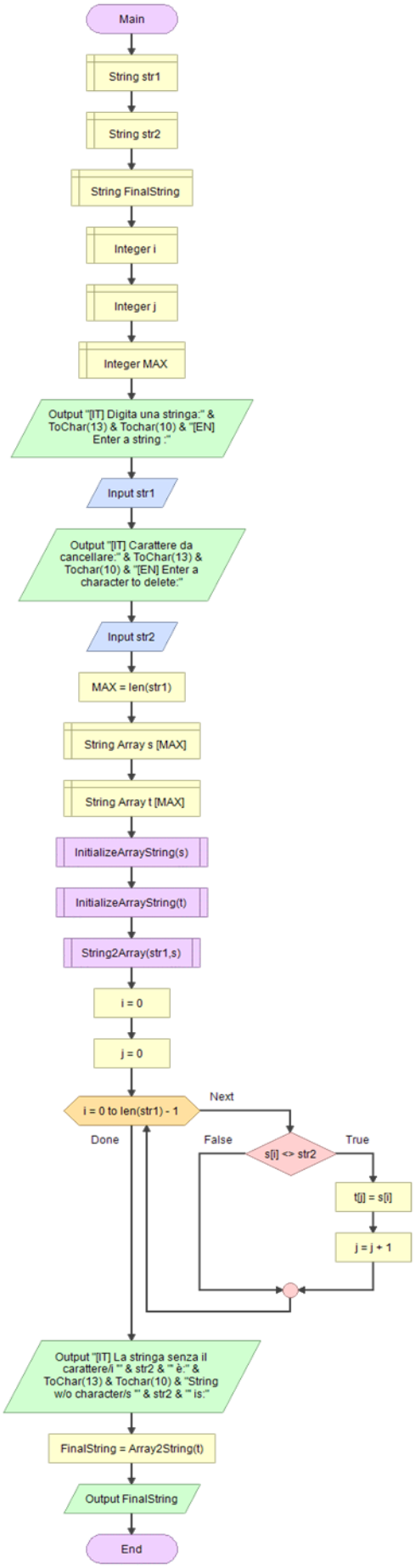

A
i = 0 to lg - 1
Next
Done
ni = ToCode(MID2(nb,lg - i, 1))
ni < 48 OR ni > 90 OR (ni > 57 AND ni < 65)
False
True
er = 1
i = lg - 1
ni < 58
ni = ni - 48
ni > 64
ni = ni - 55
ni > bd
snb = snb + ni * bd ^ i
er = 1
i = lg - 1
re = ""
sri = snb - ba * INT(snb / ba)
sri < 10
ri = ToChar(sri + 48)
sri > 9
ri = ToChar(sri + 55)
re = ri & re
INT(snb / ba) > 0
exit = True
snb = INT(snb/ba)
exit = False
result = re
Return String result

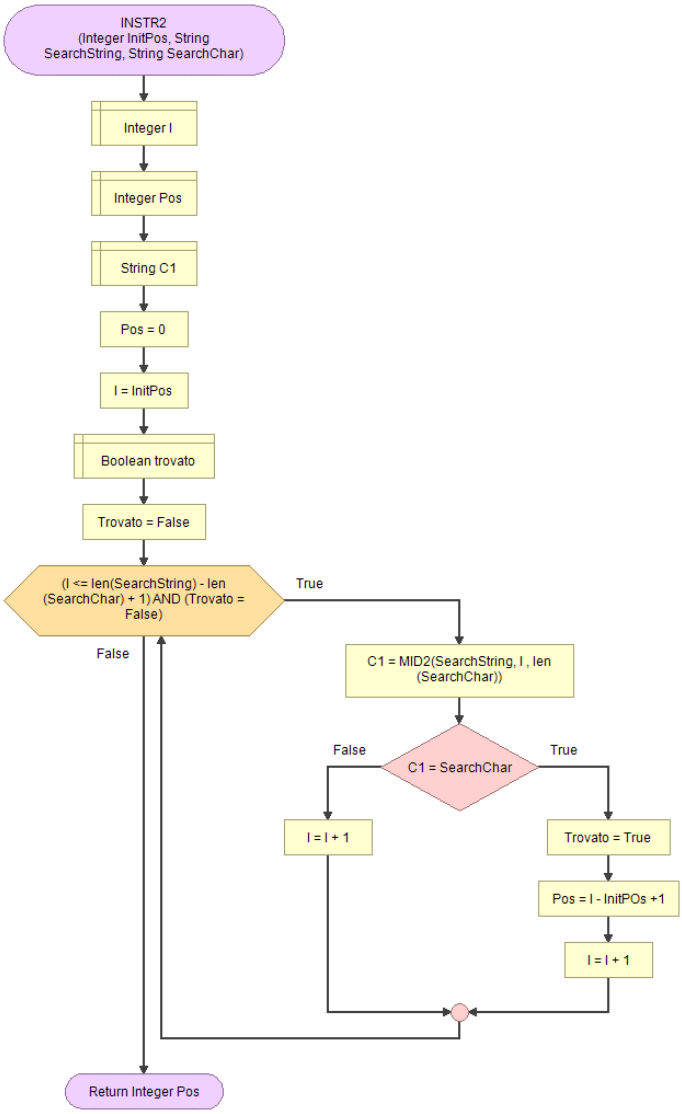

Ucase2
(String Stringa)
Integer I
String Car
Integer nCar
Car = ""
I = 1 to Len(stringa)
Next
Done
nCar = ToCode(MID2(Stringa, I, 1))
False
nCar >= 97 AND nCar <= 122
True
Car = Car & MID2(Stringa, I, 1)
Car = Car & ToChar(nCar - 32)
Stringa = Car
Return String Stringa

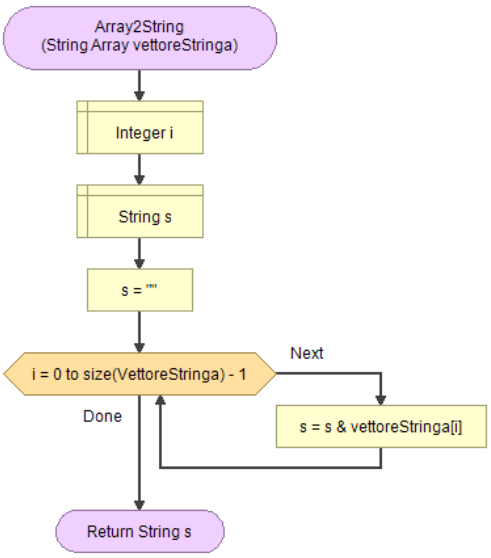

MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
False
Quanti > 0
True
I <= (len(Stringa) - 1)
True
False
False
I >= Inizio - 1
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
False
I >= Inizio - 1
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

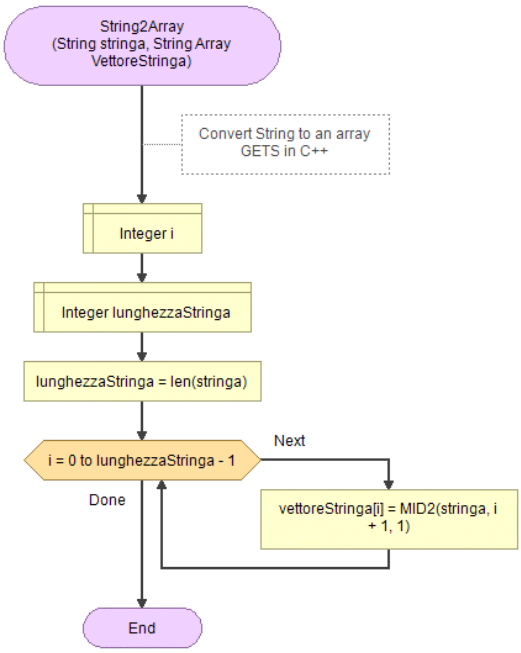
249.GSAF | Cancella le occorrenze di un carattere da una stringa

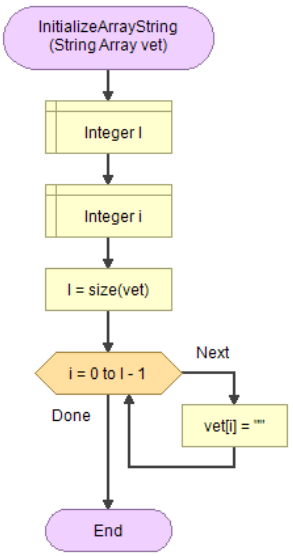
Delete a character from string (delete all the occurences)

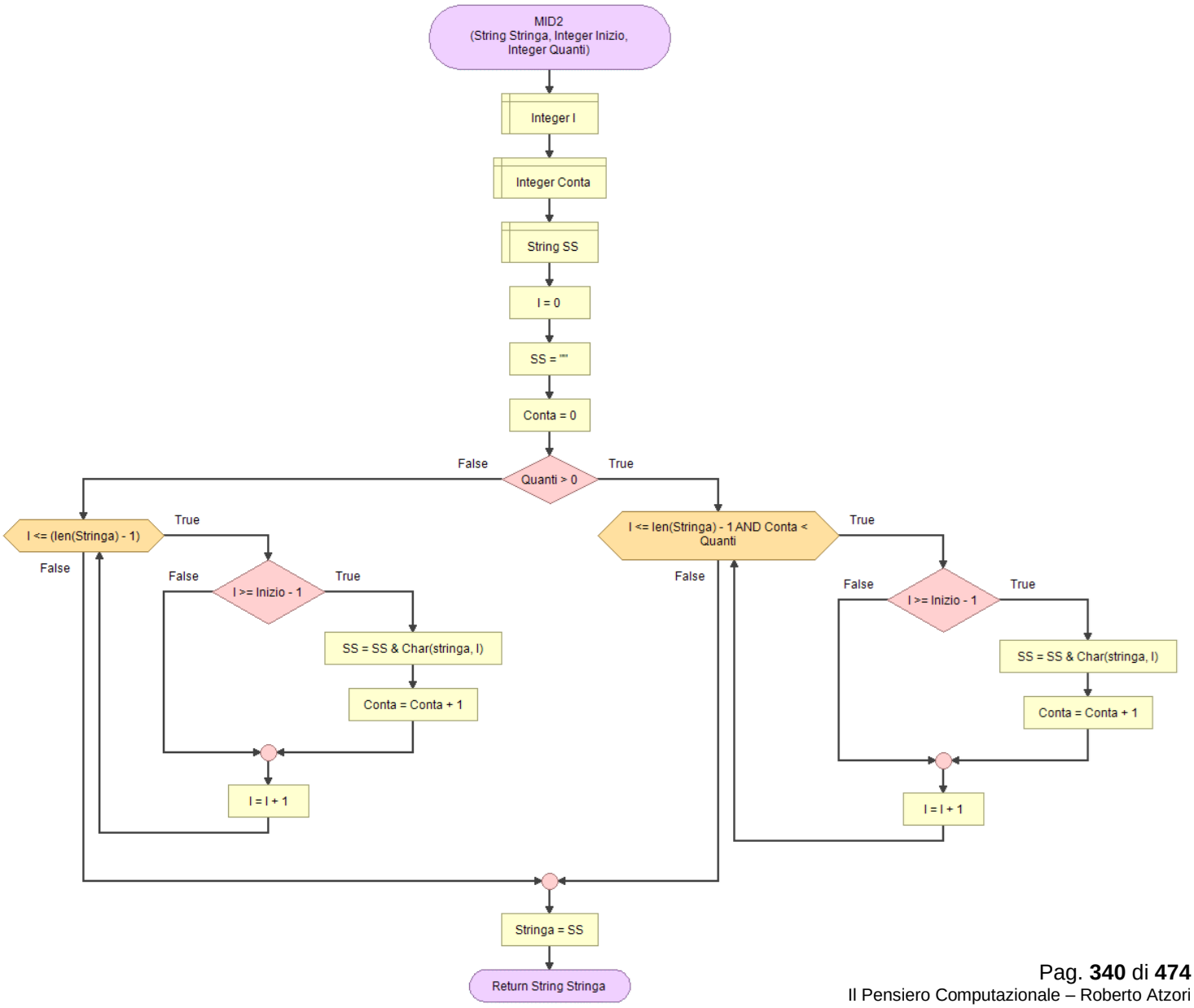





Array2String
(String Array vettoreStringa)
Integer i
String s
s = ""
i = 0 to size(VettoreStringa) - 1
Next
s = s & vettoreStringa[i]
Done
Return String s

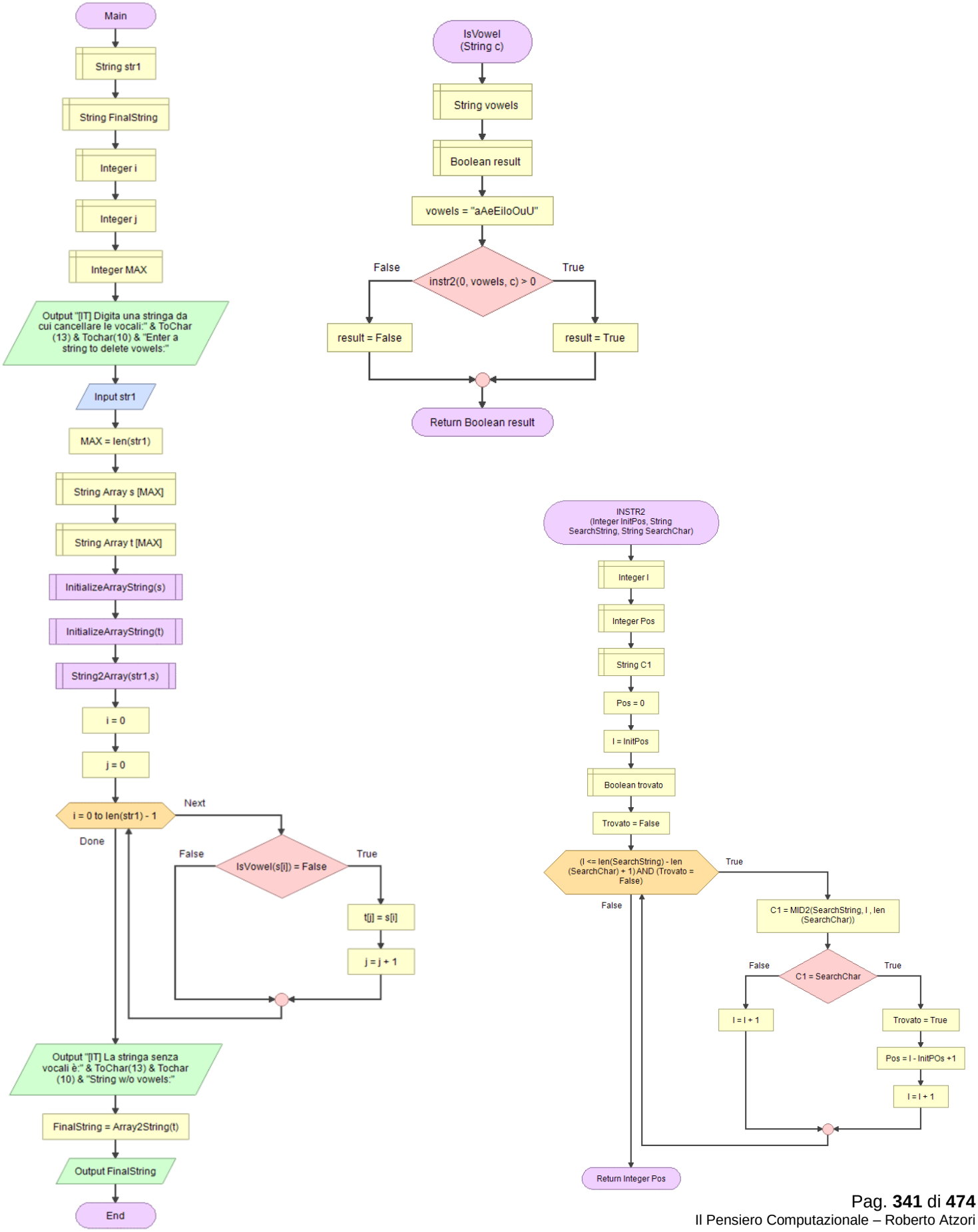

String2Array
(String stringa, String Array
VettoreStringa)
Convert String to an array
GETS in C++
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len(stringa)
i = 0 to lunghezzaStringa - 1
Next
vettoreStringa[i] = MID2(stringa, i
+ 1, 1)
Done
End

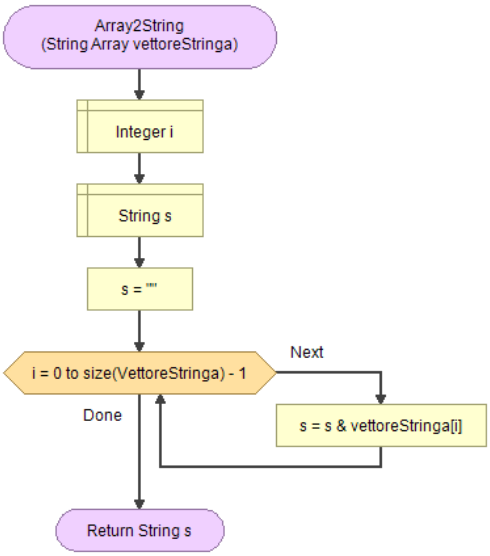

InitializeArrayString
(String Array vet)
Integer l
Integer i
l = size(vet)
i = 0 to l - 1
Next
vet[i] = ""
Done
End

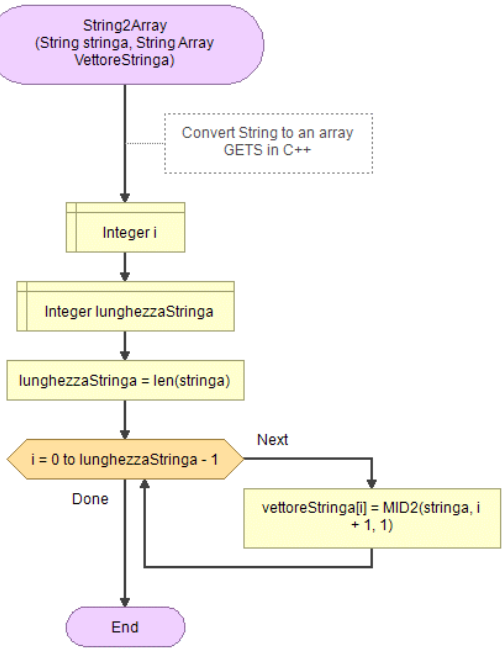

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer l
Integer Conta
String SS
l = 0
SS = ""
Conta = 0
Quanti > 0
False
True
l <= (len(Stringa) - 1)
True
False
l >= Inizio - 1
False
True
SS = SS & Char(stringa, l)
Conta = Conta + 1
l = l + 1
l <= len(Stringa) - 1 AND Conta <
Quanti
True
False
l >= Inizio - 1
False
True
SS = SS & Char(stringa, l)
Conta = Conta + 1
l = l + 1
Stringa = SS
Return String Stringa

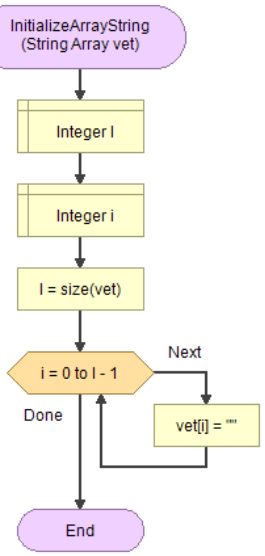
250.GSAF | Cancella le occorrenze vocali da una stringa

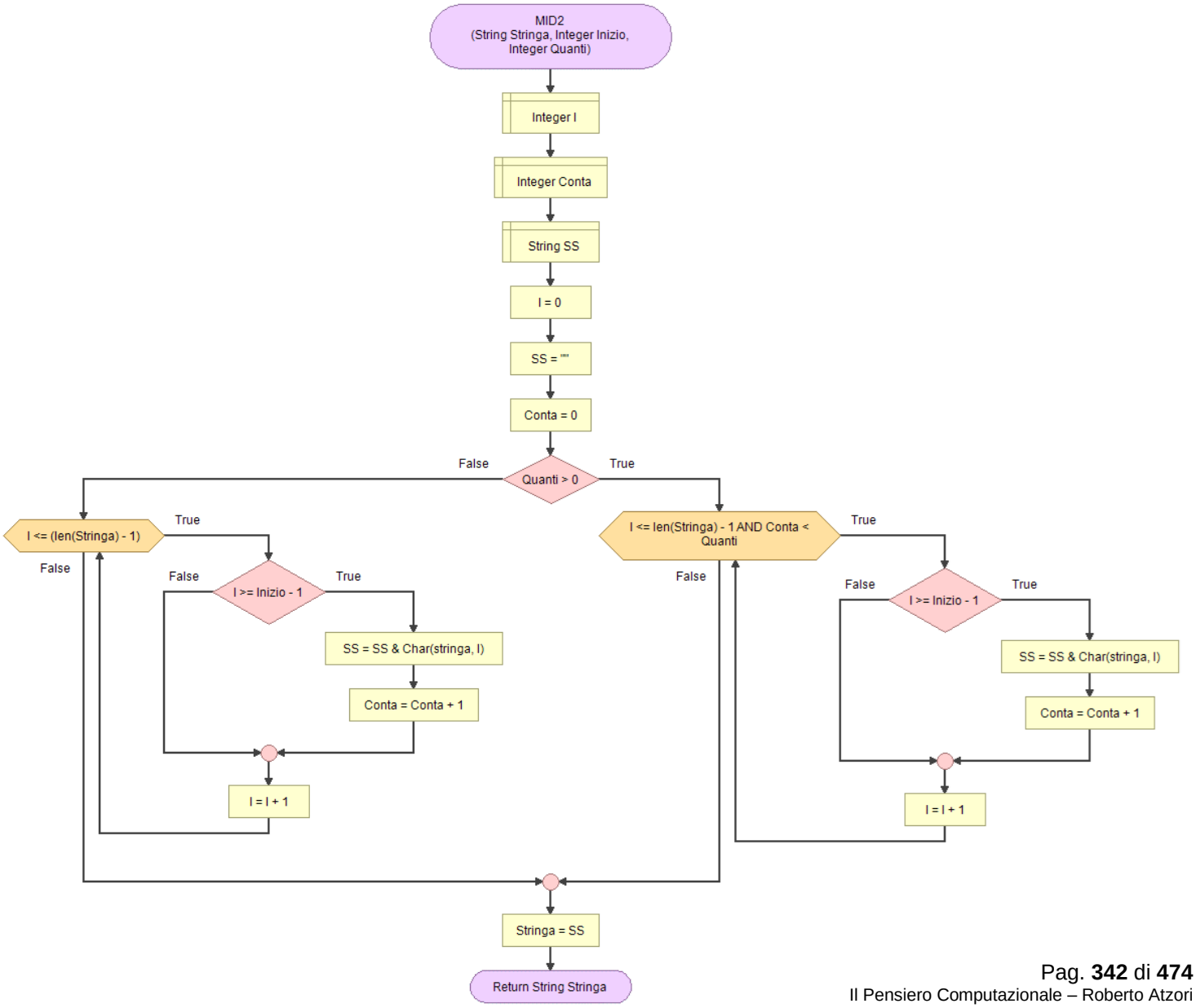
Delete vowels from string (delete all the occurrences)




Array2String
(String Array vettoreStringa)
Integer i
String s
s = ""
i = 0 to size(VettoreStringa) - 1
Next
s = s & vettoreStringa[i]
Done
Return String s

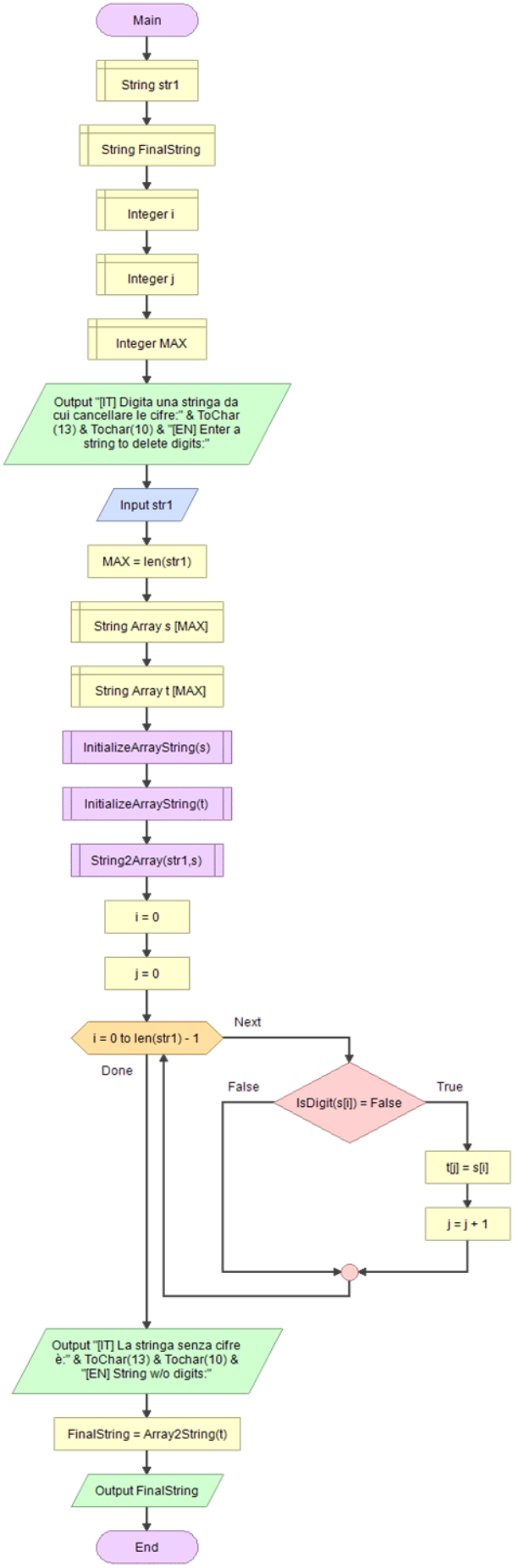

String2Array
(String stringa, String Array
VettoreStringa)
Convert String to an array
GETS in C++
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len(stringa)
i = 0 to lunghezzaStringa - 1
Next
vettoreStringa[i] = MID2(stringa, i
+ 1, 1)
Done
End

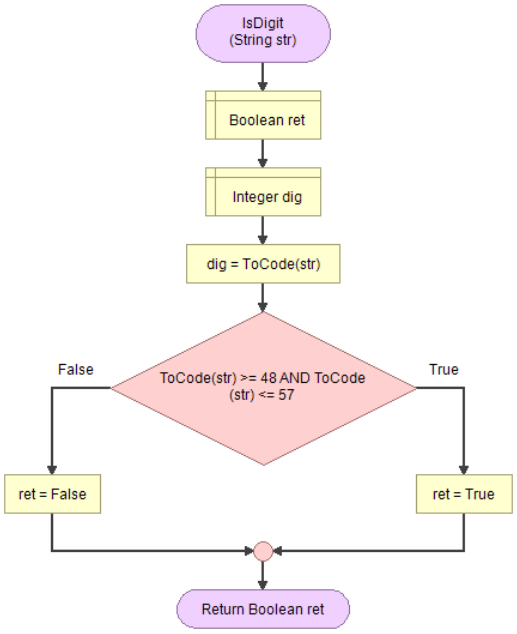

InitializeArrayString
(String Array vet)
Integer l
Integer i
l = size(vet)
i = 0 to l - 1
Next
vet[i] = ""
Done
End

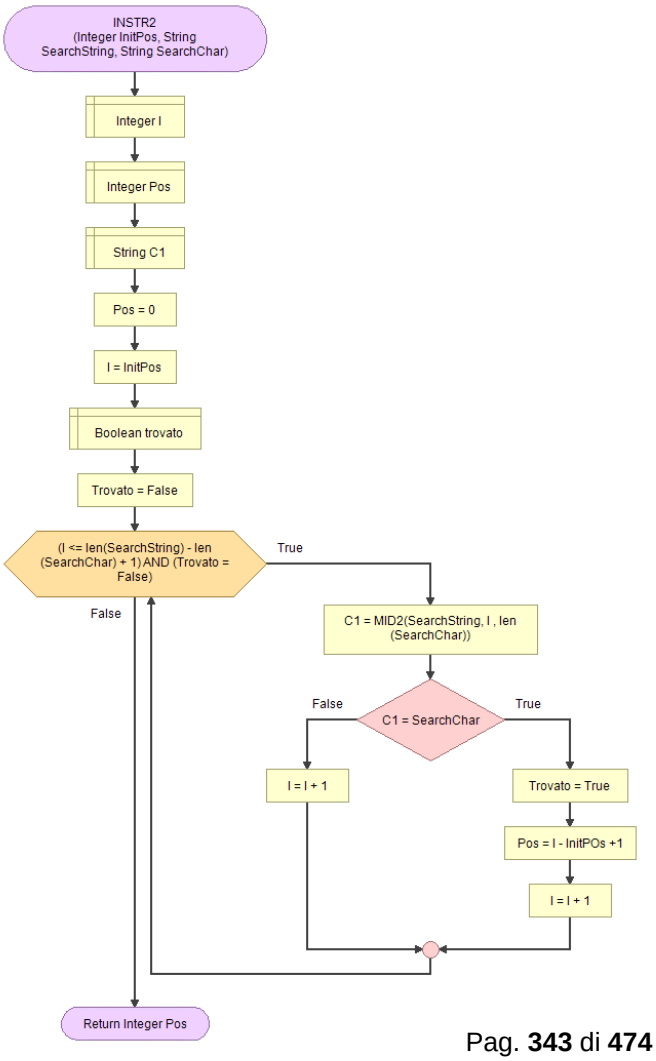

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer l
Integer Conta
String SS
l = 0
SS = ""
Conta = 0
Quanti > 0
False
True
l <= (len(Stringa) - 1)
True
False
l >= Inizio - 1
False
True
SS = SS & Char(stringa, l)
Conta = Conta + 1
l = l + 1
l <= len(Stringa) - 1 AND Conta <
Quanti
True
False
l >= Inizio - 1
False
True
SS = SS & Char(stringa, l)
Conta = Conta + 1
l = l + 1
Stringa = SS
Return String Stringa

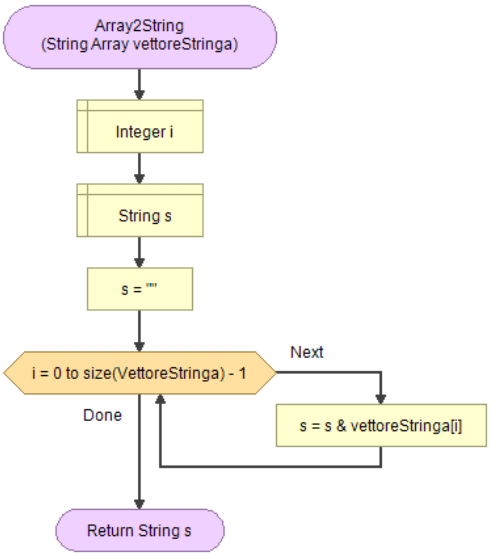
251.GSAF | Cancella le occorrenze cifre da una stringa

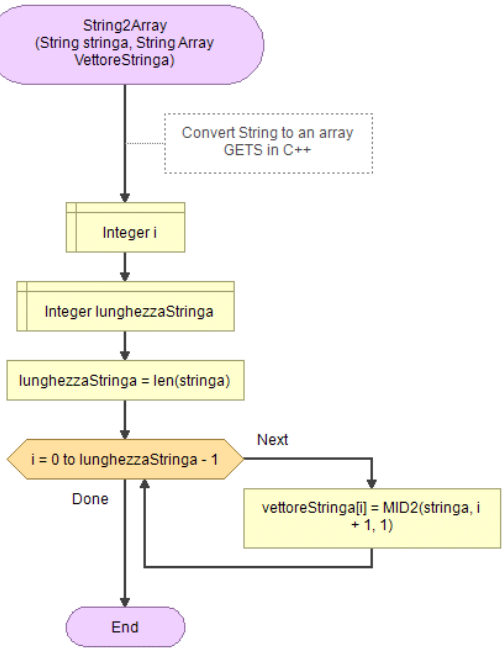
Delete digits from string (delete all the occurrences)

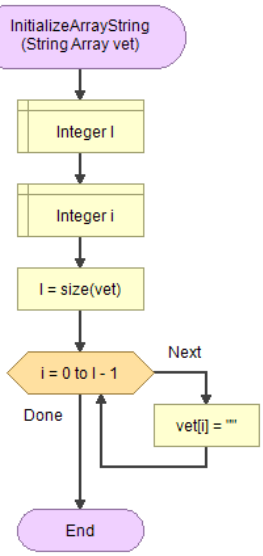


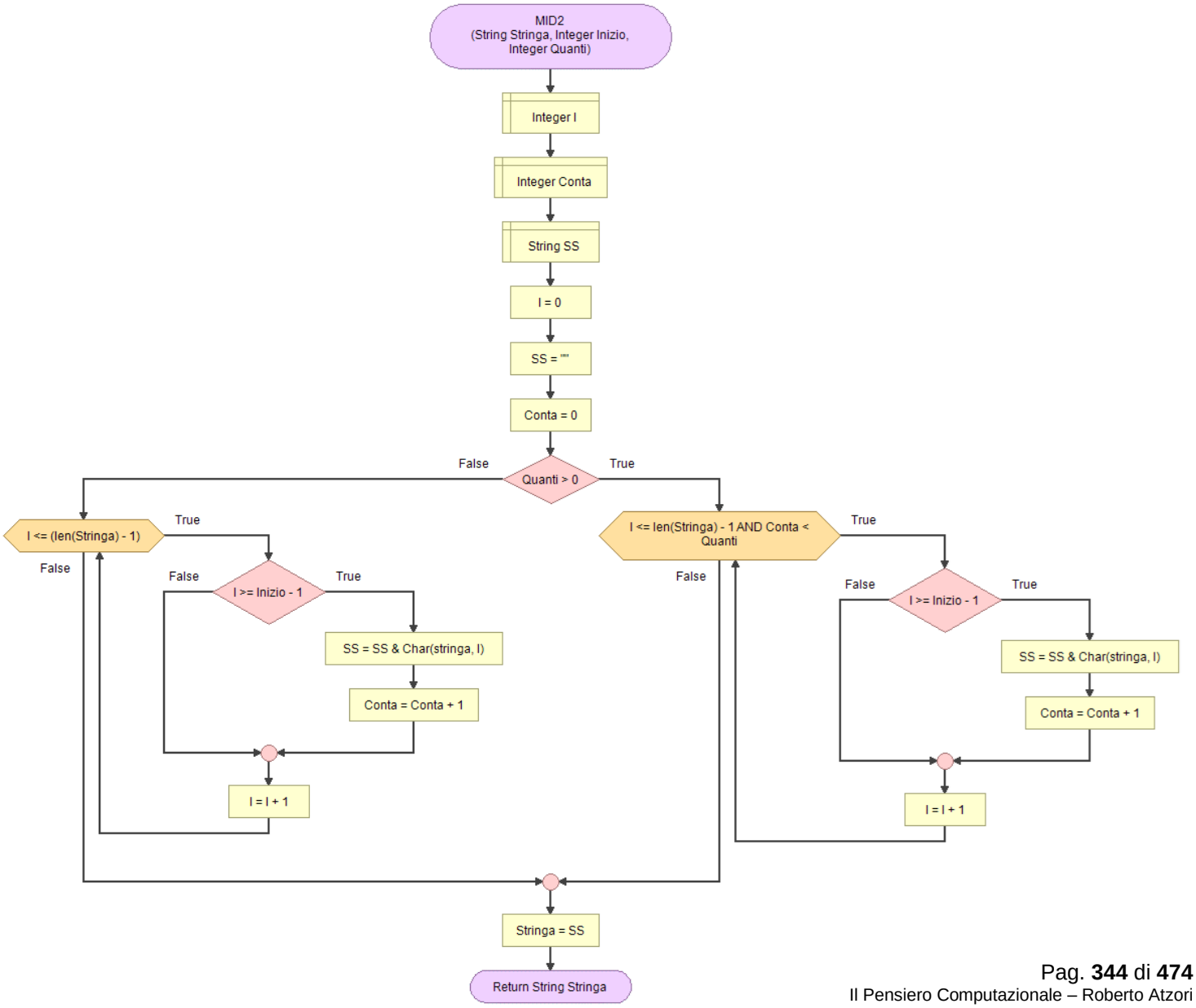





Array2String
(String Array vettoreStringa)
Integer i
String s
s = ""
i = 0 to size(VettoreStringa) - 1
Next
s = s & vettoreStringa[i]
Done
Return String s

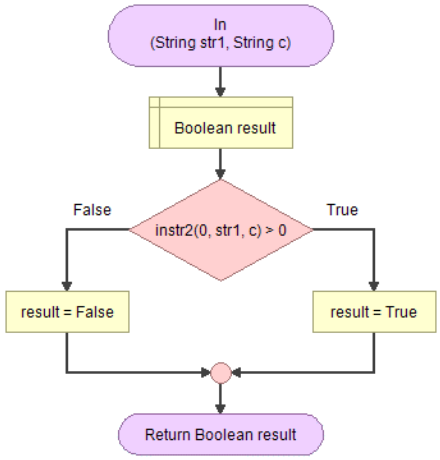

String2Array
(String stringa, String Array
VettoreStringa)
Convert String to an array
GETS in C++
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len(stringa)
i = 0 to lunghezzaStringa - 1
Next
vettoreStringa[i] = MID2(stringa, i
+ 1, 1)
Done
End

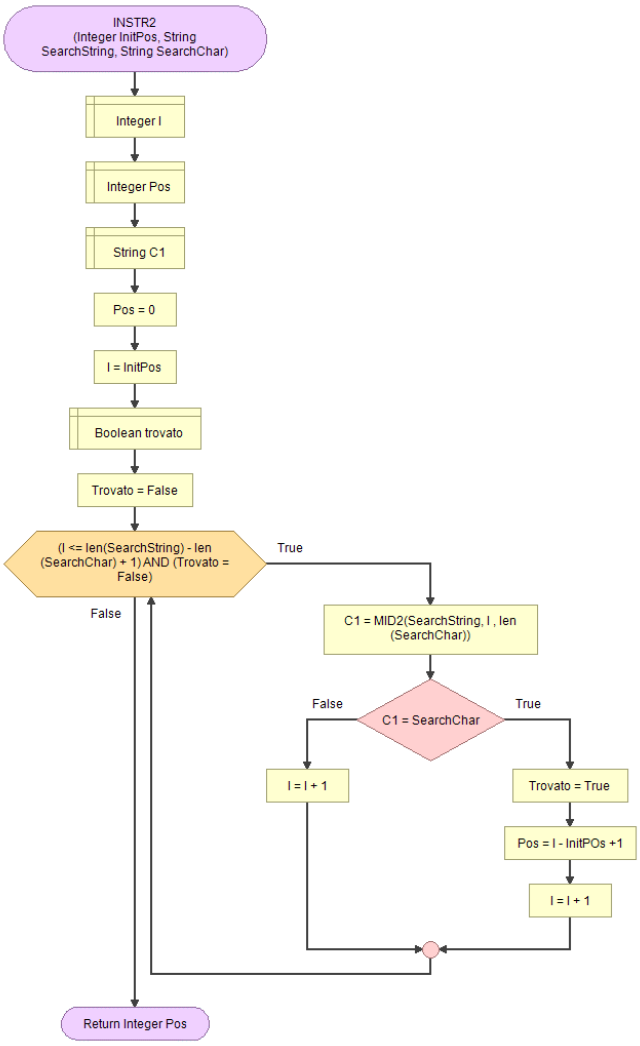

InitializeArrayString
(String Array vet)
Integer I
Integer i
I = size(vet)
i = 0 to I - 1
Next
vet[i] = ""
Done
End

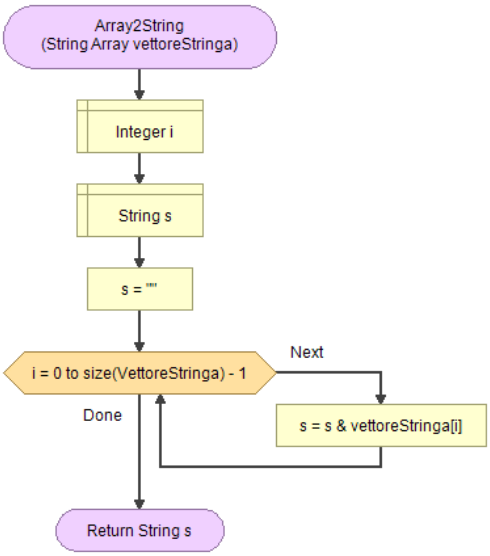

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

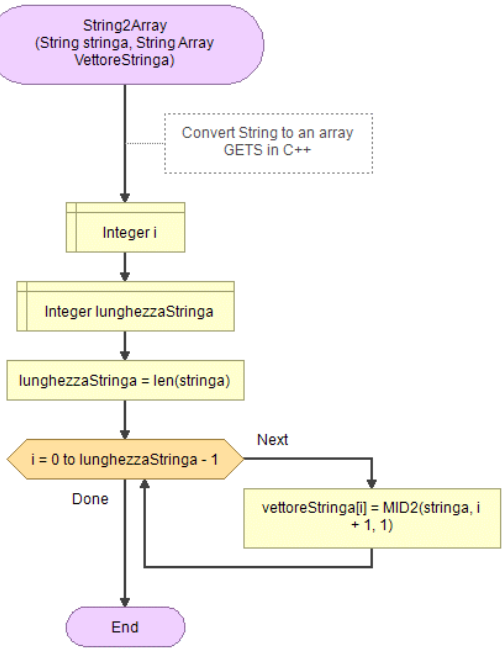
252.GSAF | Cancella le occorrenze caratteri di una prima stringa da una seconda stringa

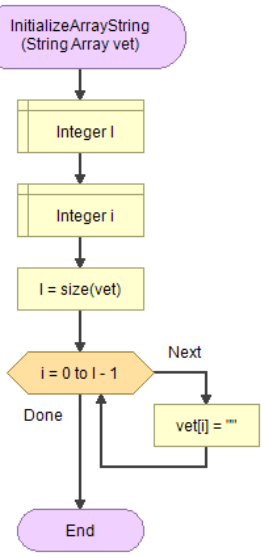
Delete all characters in second string which are present in first string (delete all the occurrences)

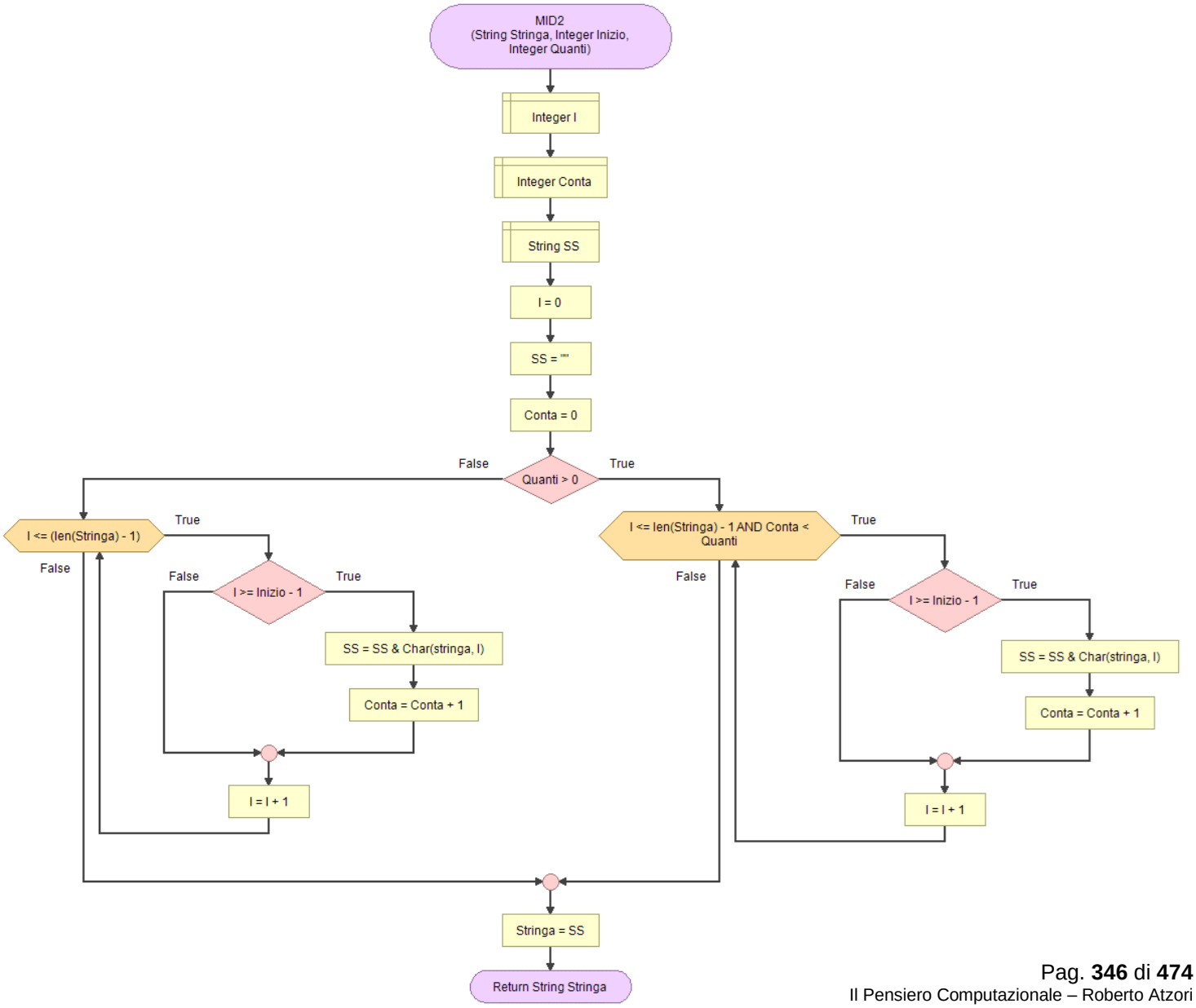





Array2String
(String Array vettoreStringa)
Integer i
String s
s = ""
i = 0 to size(VettoreStringa) - 1
Next
s = s & vettoreStringa[i]
Done
Return String s

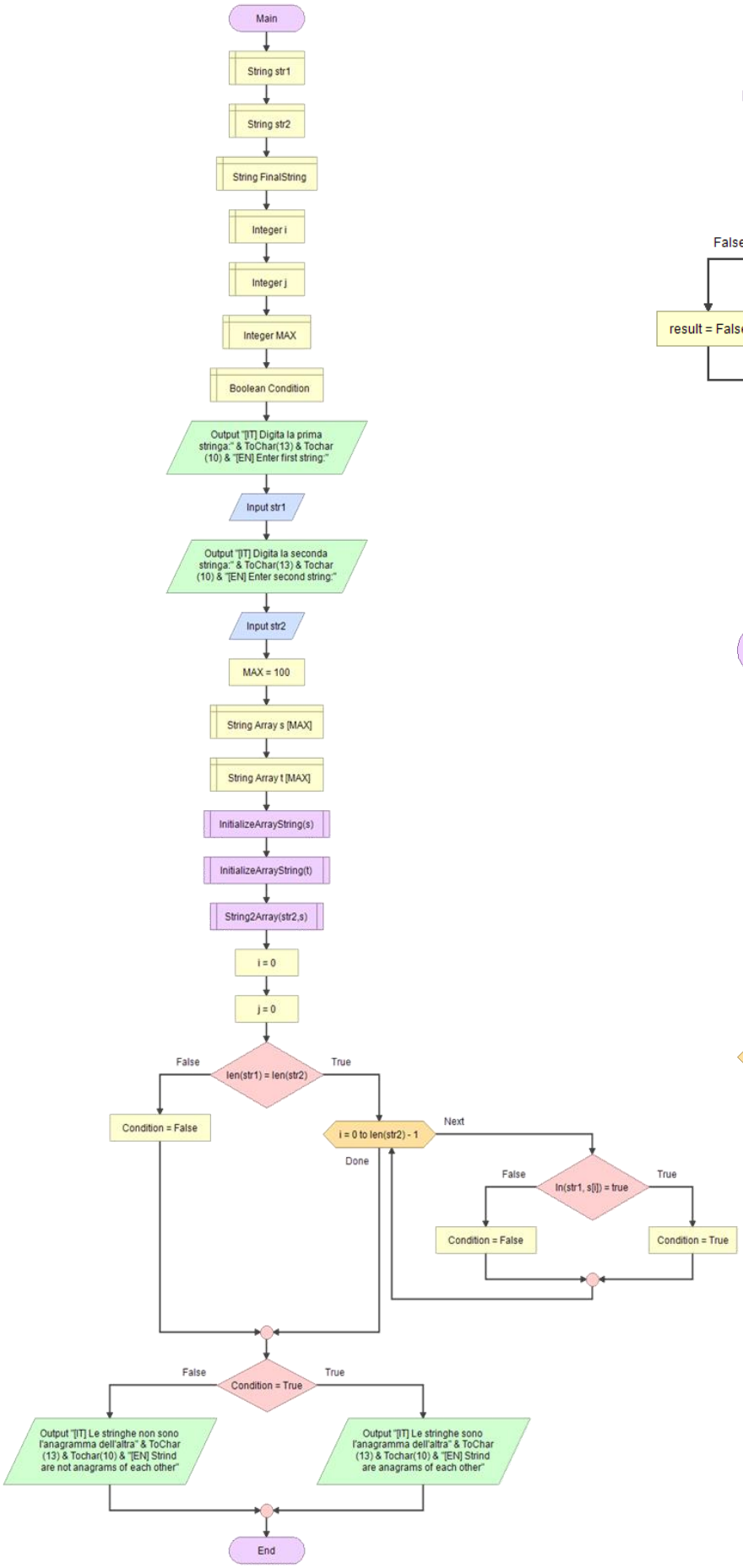

String2Array
(String stringa, String Array
VettoreStringa)
Convert String to an array
GETS in C++
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len(stringa)
i = 0 to lunghezzaStringa - 1
Next
vettoreStringa[i] = MID2(stringa, i
+ 1, 1)
Done
End

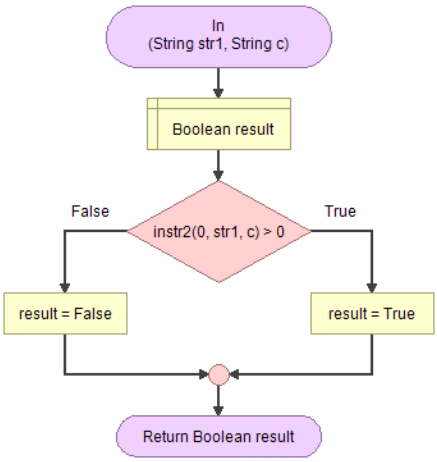

InitializeArrayString
(String Array vet)
Integer l
Integer i
l = size(vet)
i = 0 to l - 1
Next
vet[i] = ""
Done
End

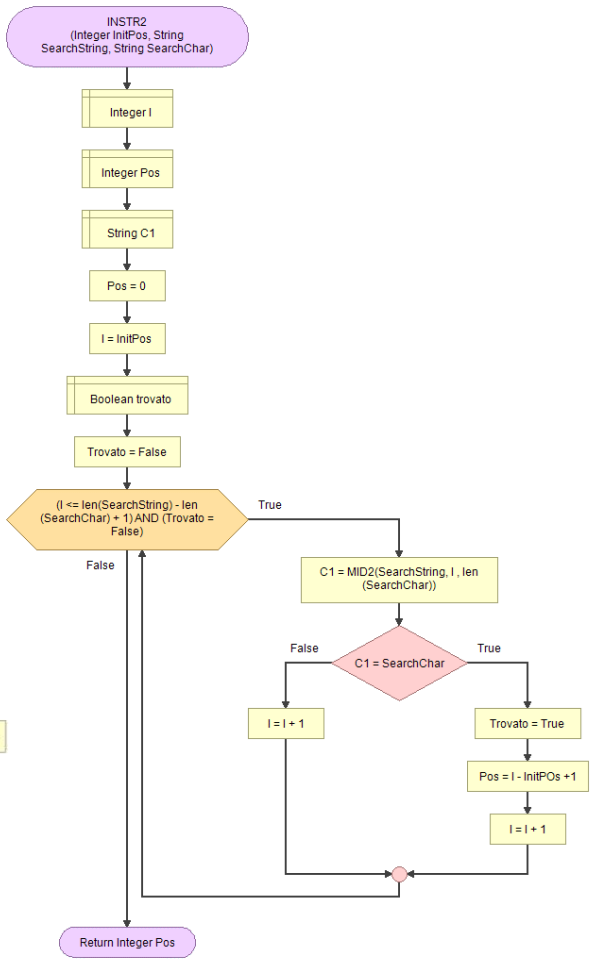

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer l
Integer Conta
String SS
l = 0
SS = ""
Conta = 0
Quanti > 0
False
True
l <= (len(Stringa) - 1)
True
False
l >= Inizio - 1
False
True
SS = SS & Char(stringa, l)
Conta = Conta + 1
l = l + 1
l <= len(Stringa) - 1 AND Conta <
Quanti
True
False
l >= Inizio - 1
False
True
SS = SS & Char(stringa, l)
Conta = Conta + 1
l = l + 1
Stringa = SS
Return String Stringa

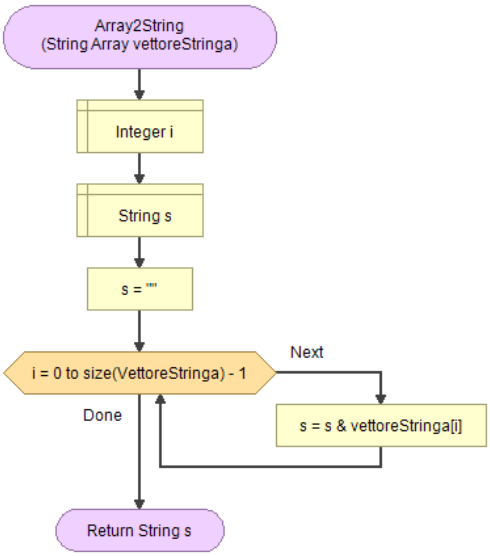
253.GSAF | Controlla se due stringhe sono una l'anagramma dell'altra

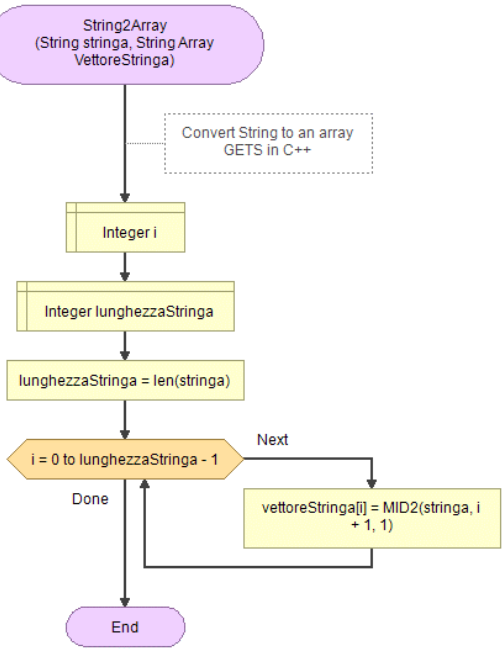
Check whether two strings are anagrams or not

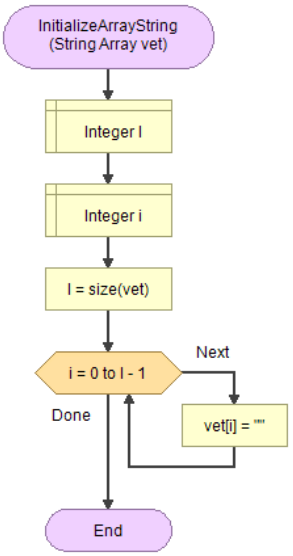


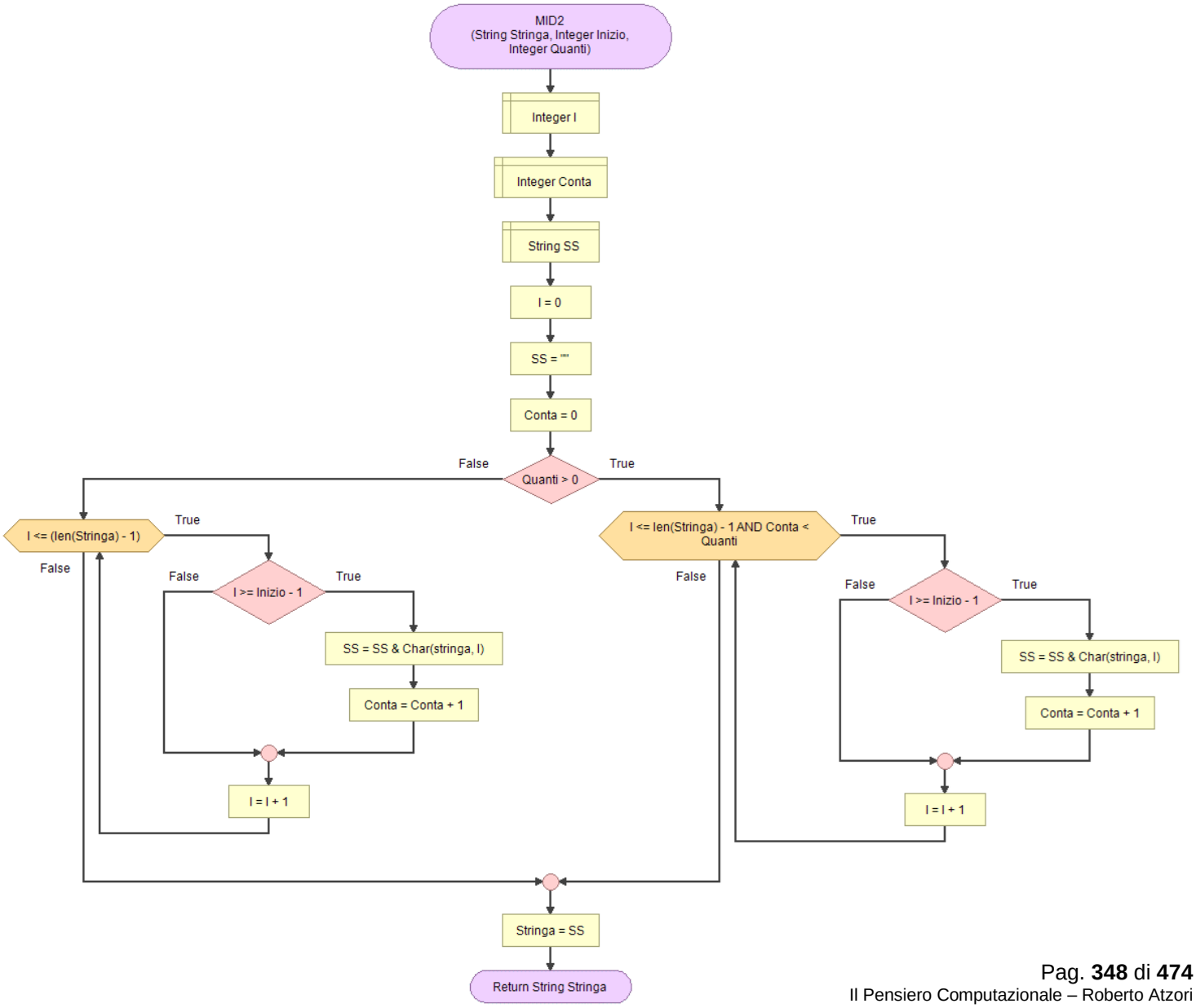





Array2String
(String Array vettoreStringa)
Integer i
String s
s = ""
i = 0 to size(VettoreStringa) - 1
Next
s = s & vettoreStringa[i]
Done
Return String s

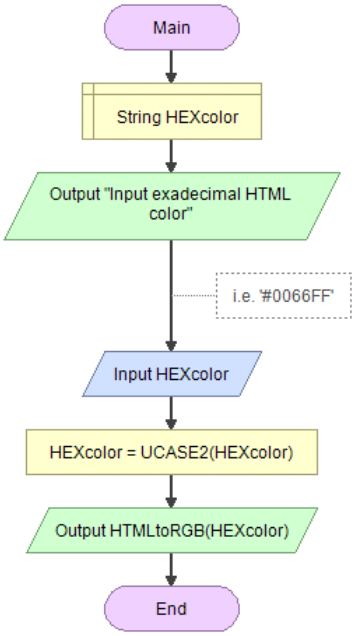

String2Array
(String stringa, String Array
VettoreStringa)
Convert String to an array
GETS in C++
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len(stringa)
i = 0 to lunghezzaStringa - 1
Next
vettoreStringa[i] = MID2(stringa, i
+ 1, 1)
Done
End

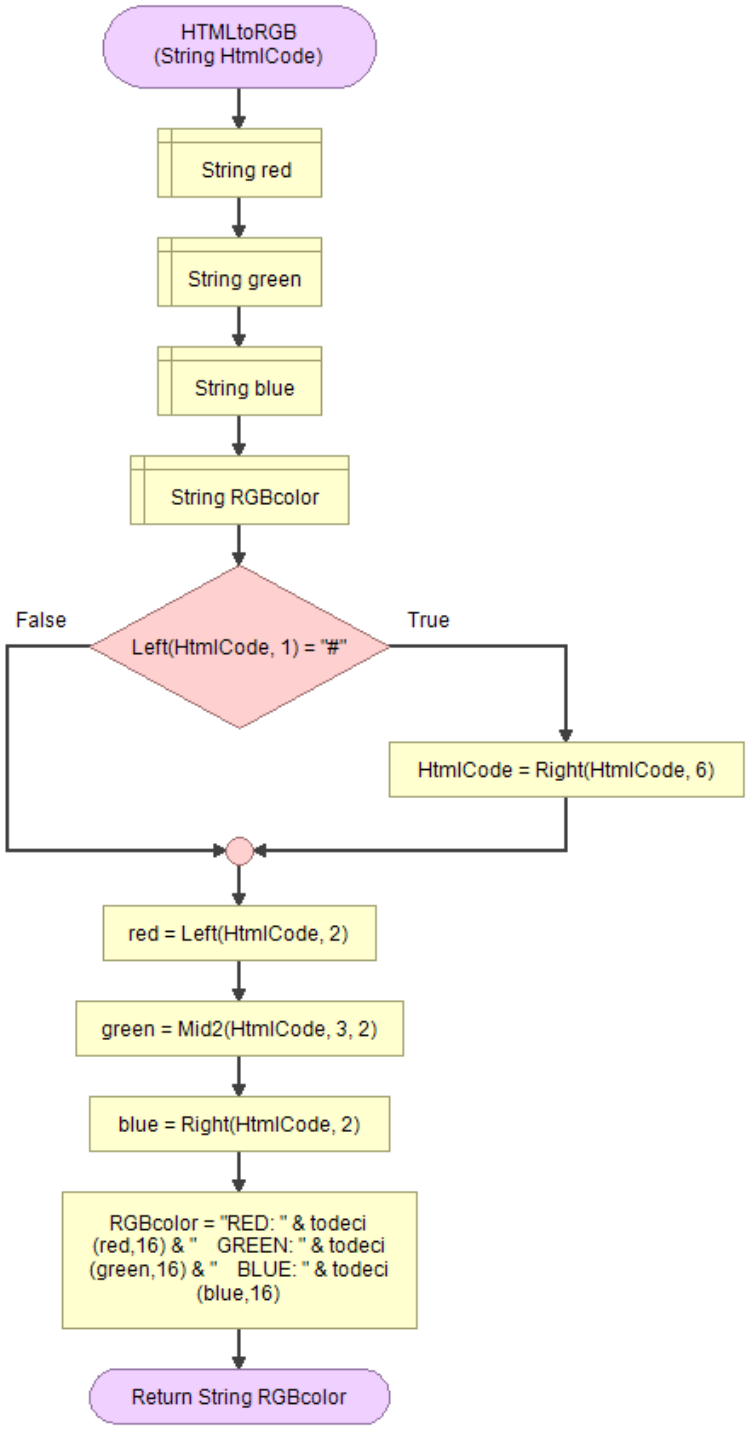

InitializeArrayString
(String Array vet)
Integer l
Integer i
l = size(vet)
i = 0 to l - 1
Next
vet[i] = ""
Done
End

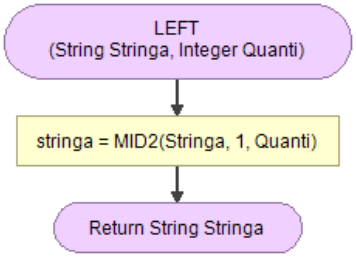

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer l
Integer Conta
String SS
l = 0
SS = ""
Conta = 0
Quanti > 0
False
True
l <= (len(Stringa) - 1)
True
False
l >= Inizio - 1
False
True
SS = SS & Char(stringa, l)
Conta = Conta + 1
l = l + 1
l <= len(Stringa) - 1 AND Conta <
Quanti
True
False
l >= Inizio - 1
False
True
SS = SS & Char(stringa, l)
Conta = Conta + 1
l = l + 1
Stringa = SS
Return String Stringa

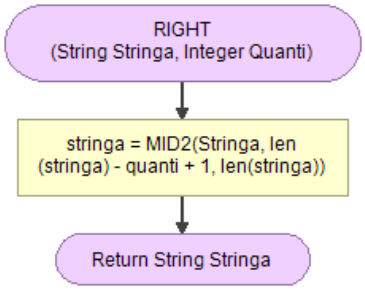
254.GSAF | Conversione dal colore HTML al colore RGB

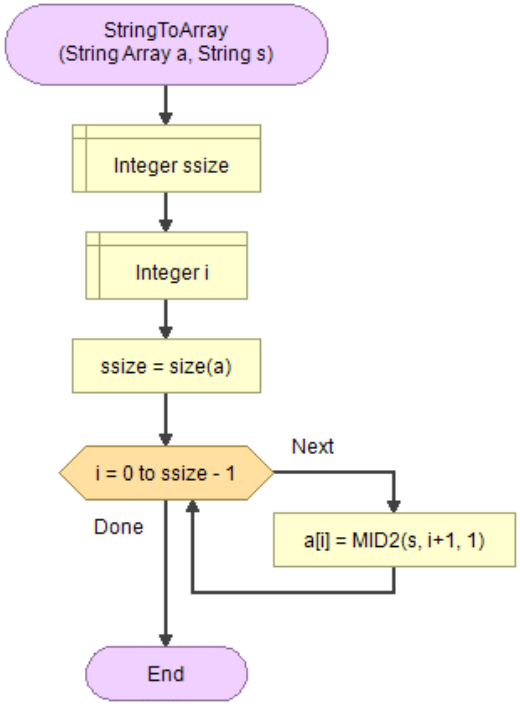
Convert HTML color to RGB

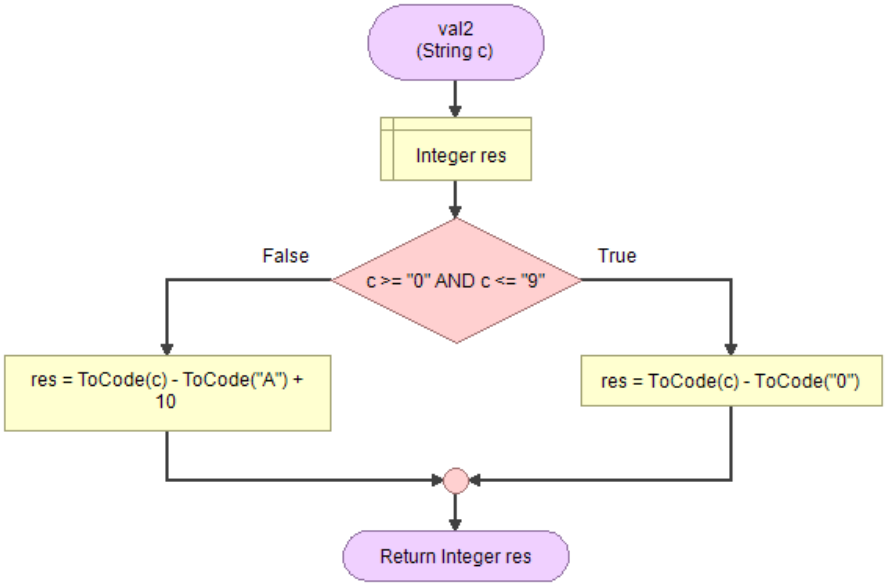


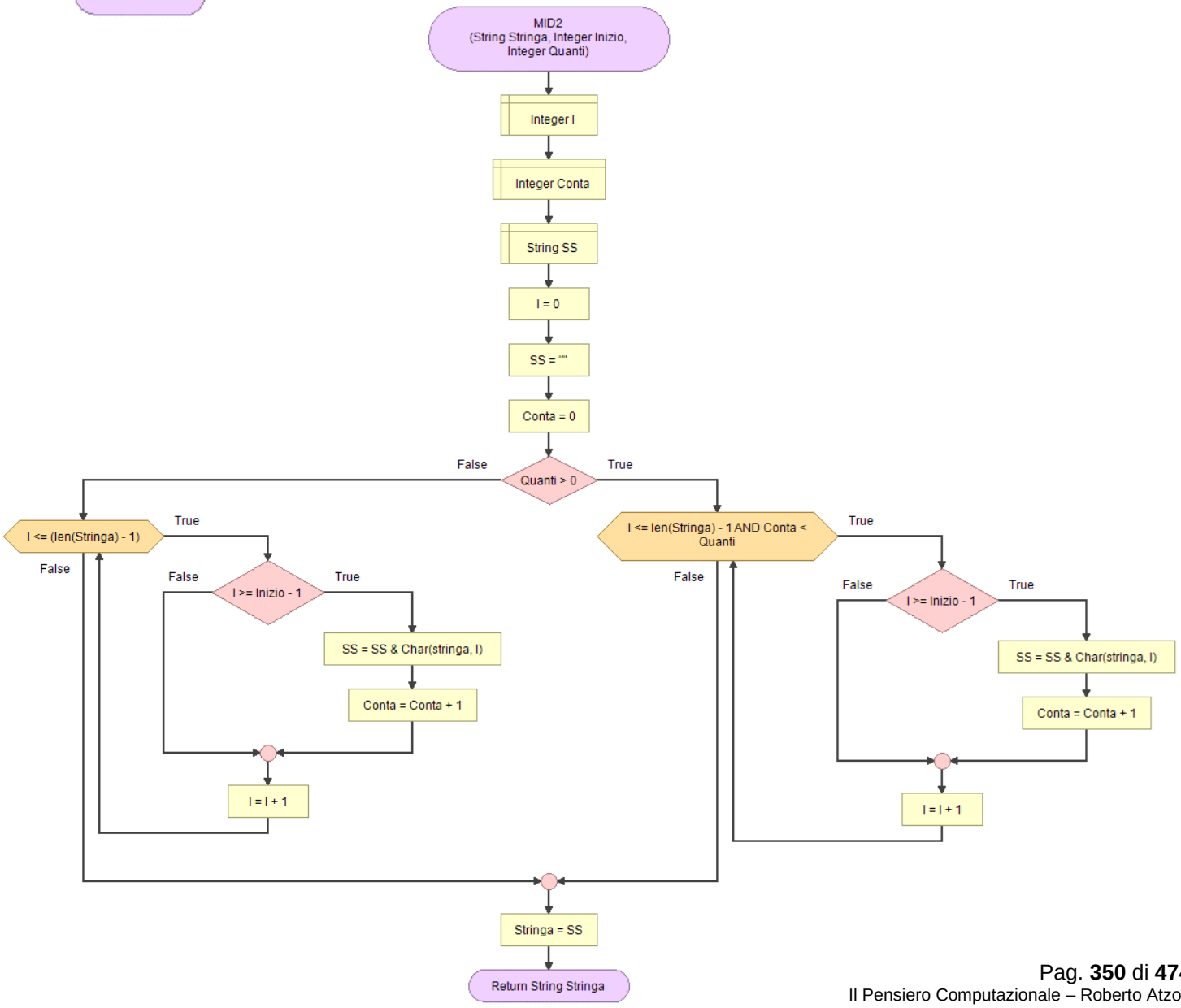


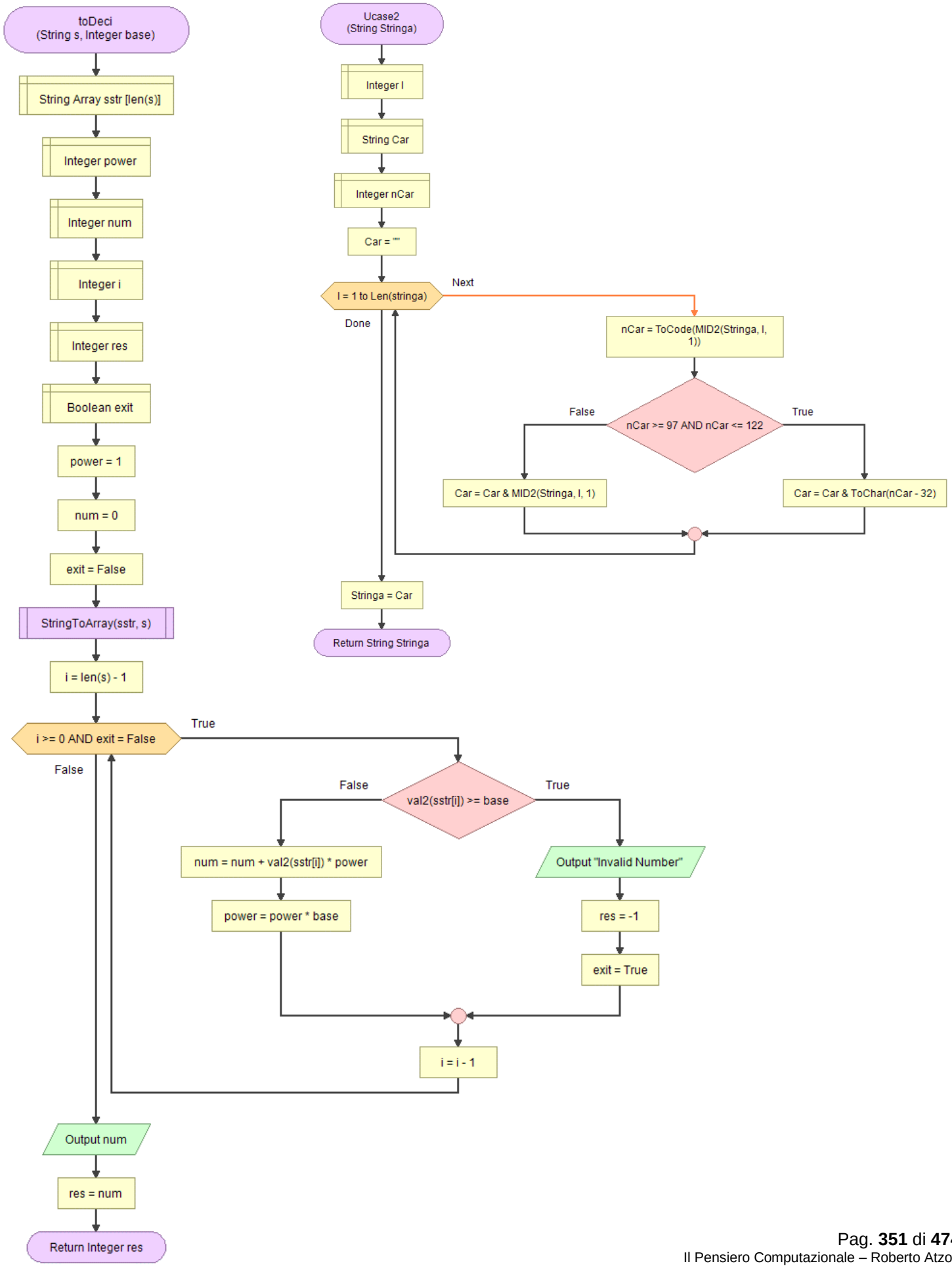




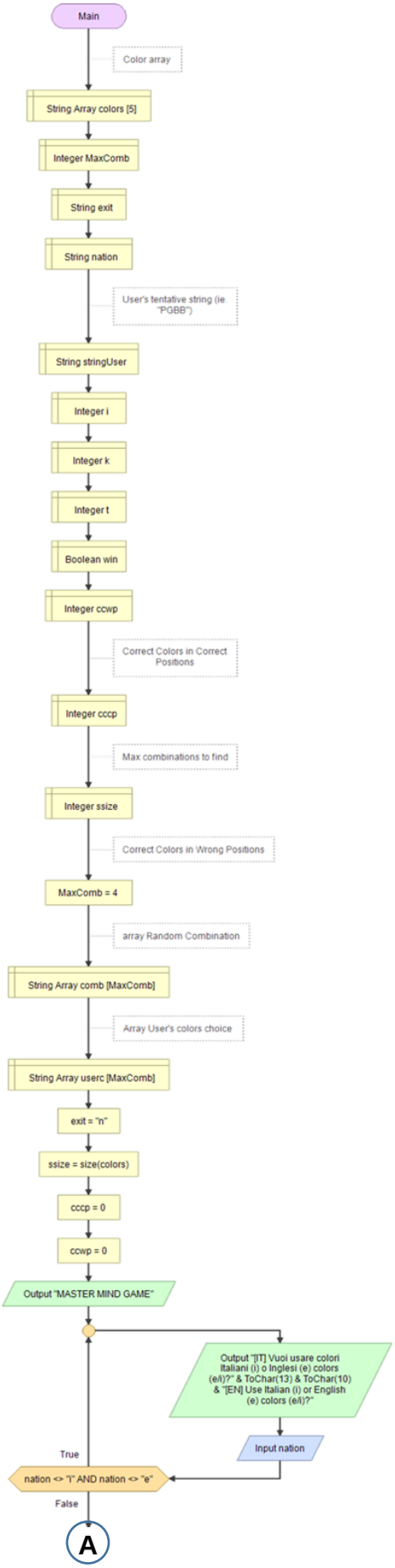

StringToArray
(String Array a, String s)
Integer ssize
Integer i
ssize = size(a)
i = 0 to ssize - 1
Next
Done
a[i] = MID2(s, i+1, 1)
End

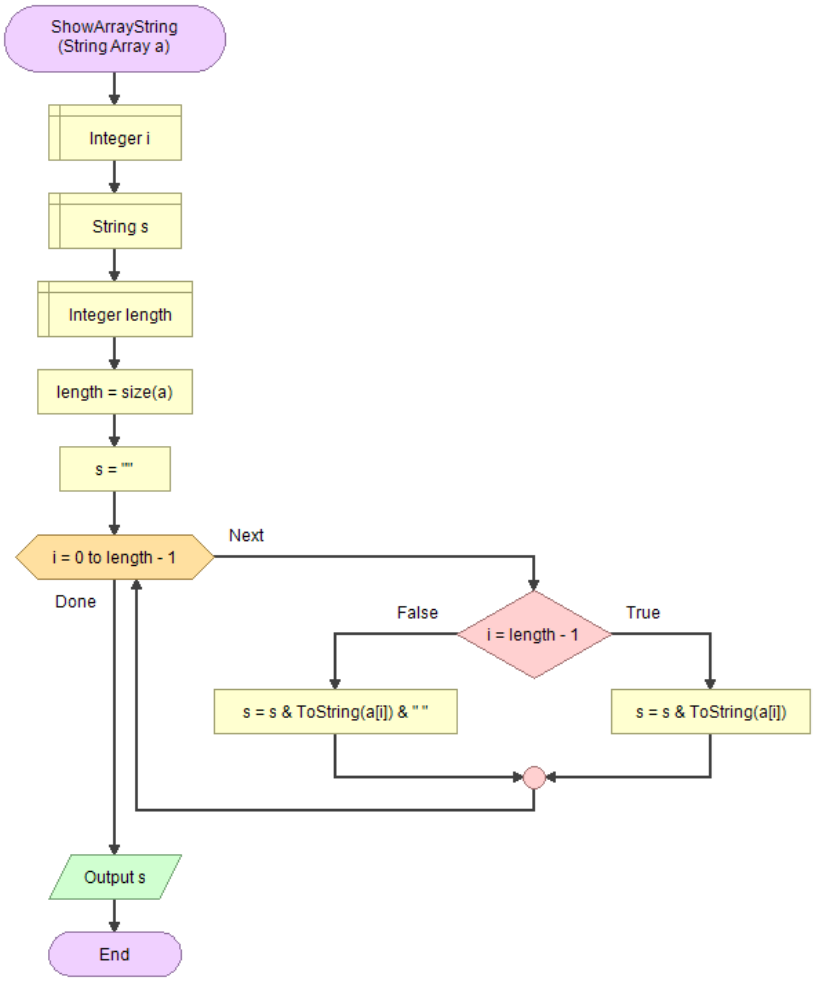

val2
(String c)
Integer res
False
c >= "0" AND c <= "9"
True
res = ToCode(c) - ToCode("A") + 10
res = ToCode(c) - ToCode("0")
Return Integer res

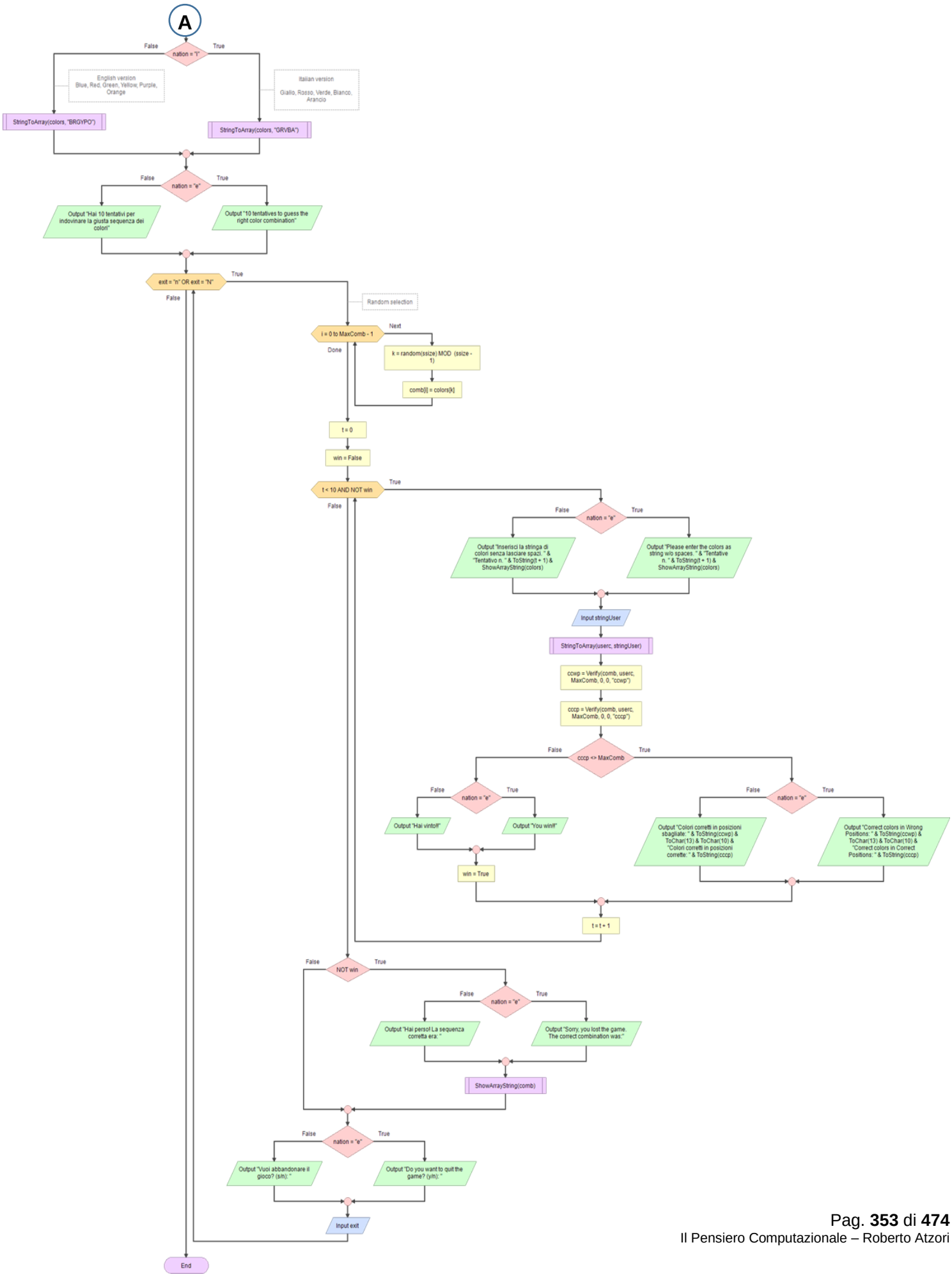

MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
False
Quanti > 0
True
I <= (len(Stringa) - 1)
True
False
False
I >= Inizio - 1
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
False
I >= Inizio - 1
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

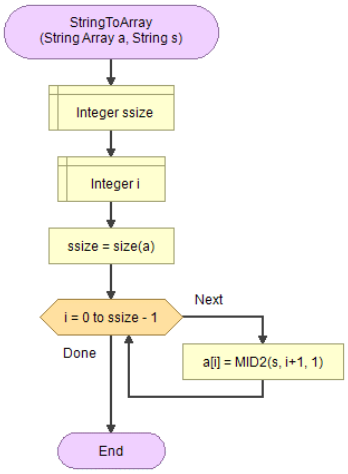

toDeci
(String s, Integer base)
String Array sstr [len(s)]
Integer power
Integer num
Integer i
Integer res
Boolean exit
power = 1
num = 0
exit = False
StringToArray(sstr, s)
i = len(s) - 1
i >= 0 AND exit = False
True
False
val2(sstr[i]) >= base
False
True
num = num + val2(sstr[i]) * power
power = power * base
Output "Invalid Number"
res = -1
exit = True
i = i - 1
Output num
res = num
Return Integer res
Ucase2
(String Stringa)
Integer I
String Car
Integer nCar
Car = ""
I = 1 to Len(stringa)
Next
Done
nCar = ToCode(MID2(Stringa, I, 1))
False
nCar >= 97 AND nCar <= 122
True
Car = Car & MID2(Stringa, I, 1)
Car = Car & ToChar(nCar - 32)
Stringa = Car
Return String Stringa

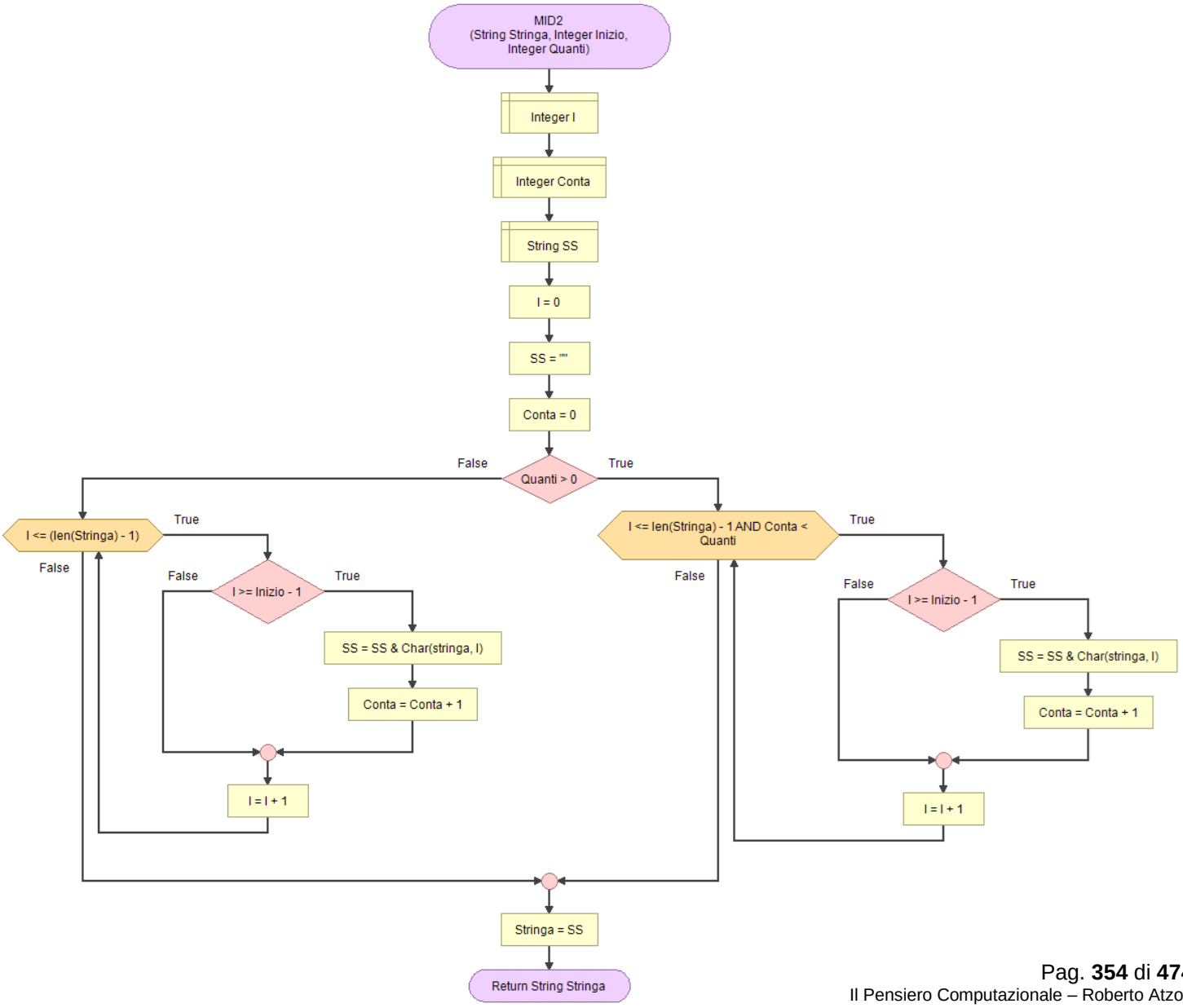
Master Mind game

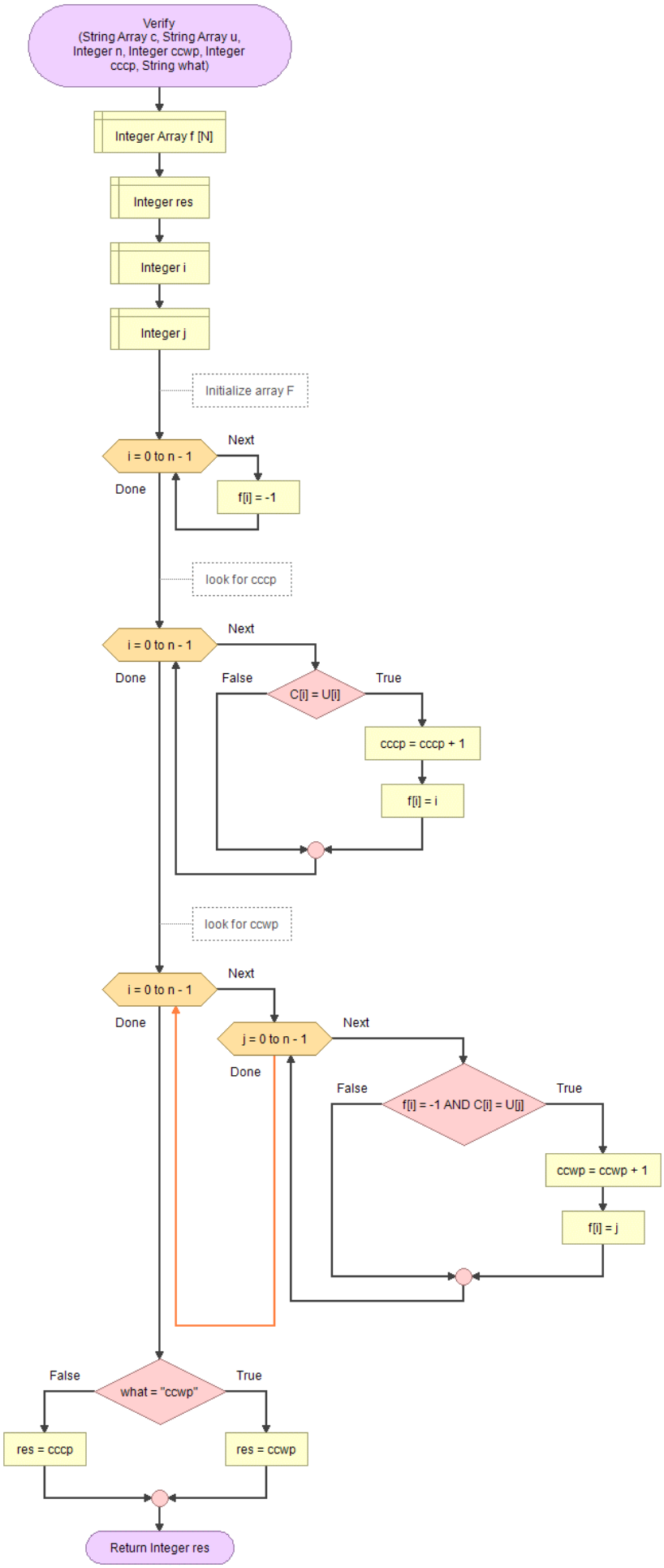





A
nation = "I"
False
True
English version
Blue, Red, Green, Yellow, Purple, Orange
Italian version
Giallo, Rosso, Verde, Bianco, Arancio
StringToArray(colors, "BRGYPO")
StringToArray(colors, "GRVBA")
nation = "e"
Output "Hai 10 tentativi per indovinare la giusta sequenza dei colori"
Output "10 tentatives to guess the right color combination"
exit = "n" OR exit = "N"
Random selection
I = 0 to MaxComb - 1
Next
Done
k = random(ssize) MOD (ssize - 1)
comb[i] = colors[k]
t = 0
win = False
t < 10 AND NOT win
Output "Inserisci la stringa di colori senza lasciare spazi. " & "Tentativo n. " & ToString(t + 1) & ShowArrayString(colors)
Output "Please enter the colors as string w/o spaces. " & "Tentative n. " & ToString(t + 1) & ShowArrayString(colors)
Input stringUser
StringToArray(userc, stringUser)
ccwp = Verify(comb, userc, MaxComb, 0, 0, "ccwp")
cccp = Verify(comb, userc, MaxComb, 0, 0, "cccp")
cccp <> MaxComb
Output "Hai vinto!!"
Output "You win!!"
win = True
Output "Colori corretti in posizioni sbagliate: " & ToString(ccwp) & ToChar(13) & ToChar(10) & "Colori corretti in posizioni corrette: " & ToString(cccp)
Output "Correct colors in Wrong Positions: " & ToString(ccwp) & ToChar(13) & ToChar(10) & "Correct colors in Correct Positions: " & ToString(cccp)
t = t + 1
NOT win
Output "Hai perso! La sequenza corretta era: "
Output "Sorry, you lost the game. The correct combination was:"
ShowArrayString(comb)
Output "Vuoi abbandonare il gioco? (s/n): "
Output "Do you want to quit the game? (y/n): "
Input exit
End

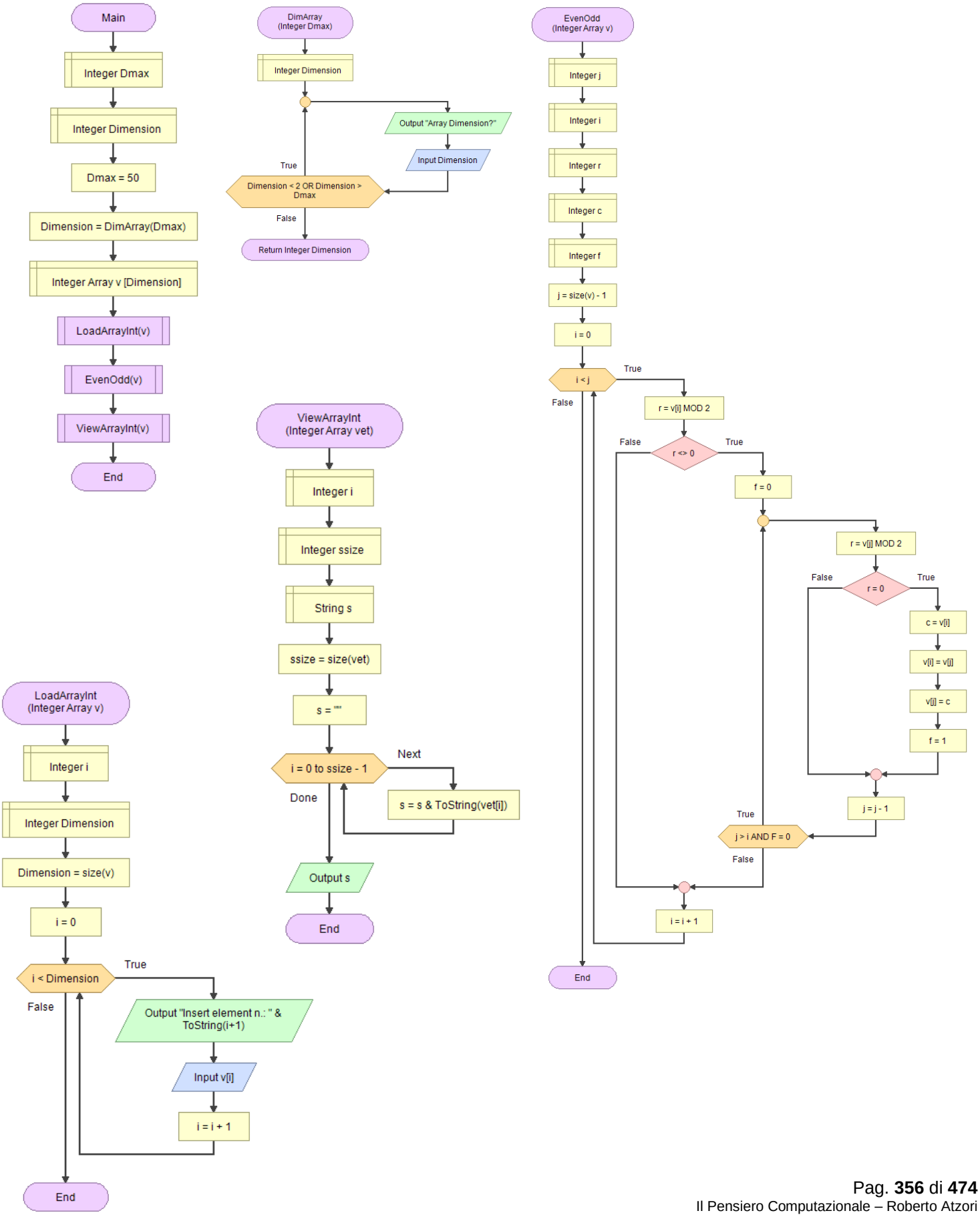

StringToArray
(String Array a, String s)
Integer ssize
Integer i
ssize = size(a)
i = 0 to ssize - 1
Next
a[i] = MID2(s, i+1, 1)
Done
End


MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

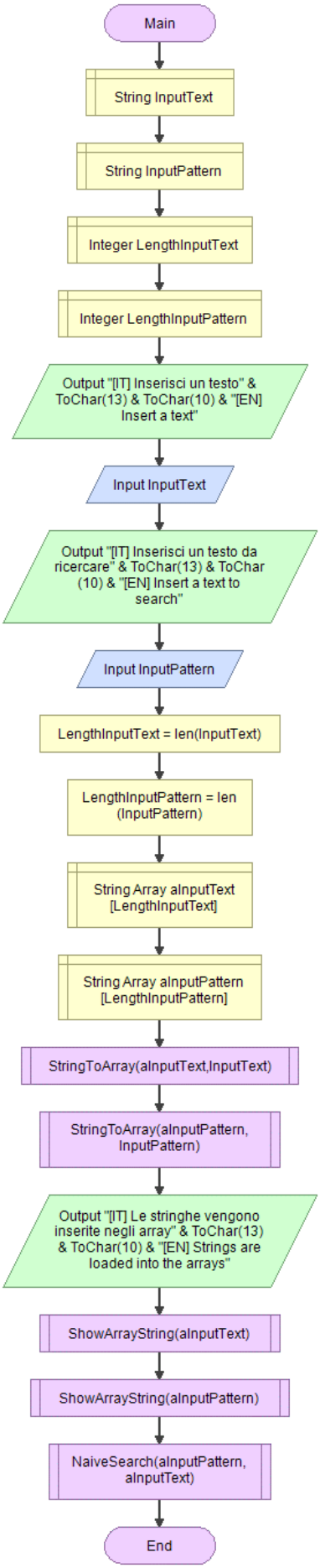

Verify
(String Array c, String Array u, Integer n, Integer ccwp, Integer cccp, String what)
Integer Array f [N]
Integer res
Integer i
Integer j
Initialize array F
i = 0 to n - 1
Next
Done
f[i] = -1
look for cccp
i = 0 to n - 1
Next
Done
False
C[i] = U[i]
True
cccp = cccp + 1
f[i] = i
look for ccwp
i = 0 to n - 1
Next
Done
j = 0 to n - 1
Next
Done
False
f[i] = -1 AND C[i] = U[j]
True
ccwp = ccwp + 1
f[i] = j
False
what = "ccwp"
True
res = cccp
res = ccwp
Return Integer res

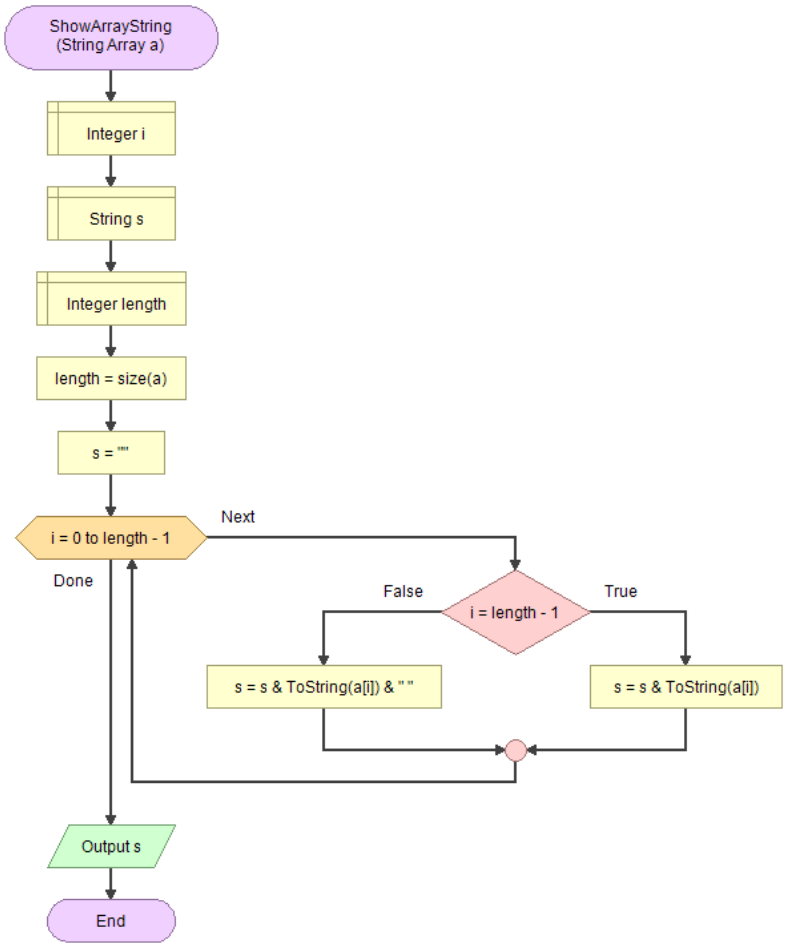
256.GSAF | Disporre gli elementi pari di un vettore nelle posizioni iniziali e gli elementi dispari nelle posizioni finali

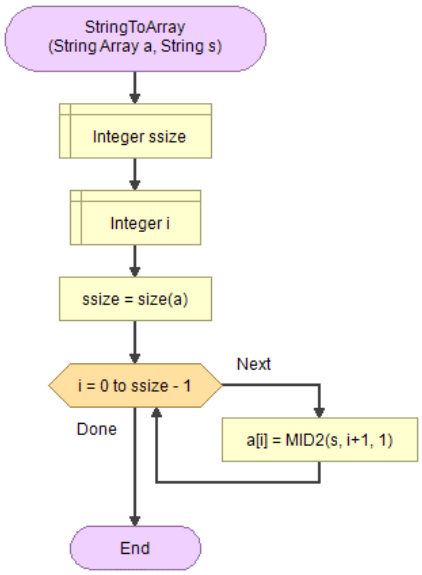
Put the even elements of a vector in the initial positions and the odd elements in the end positions

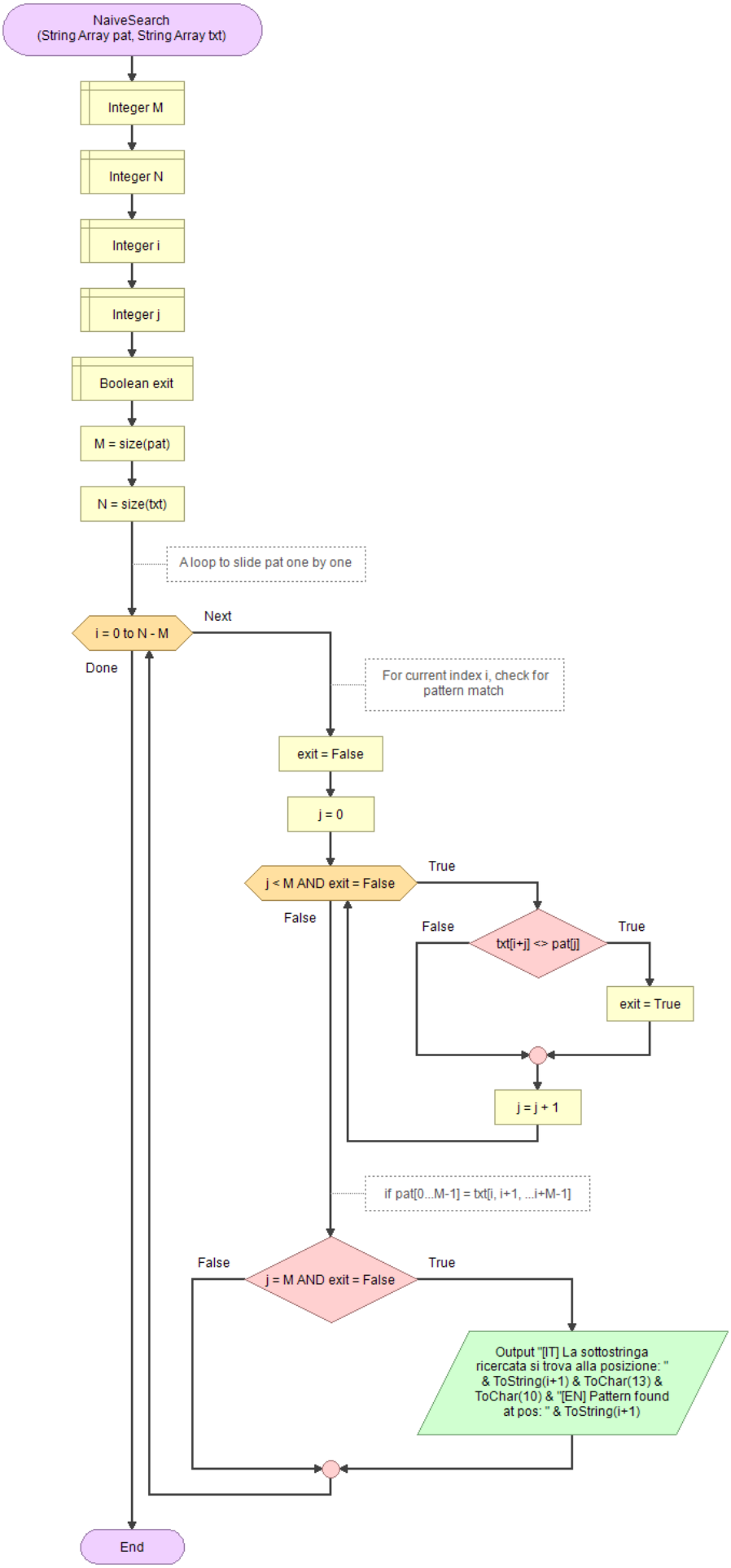


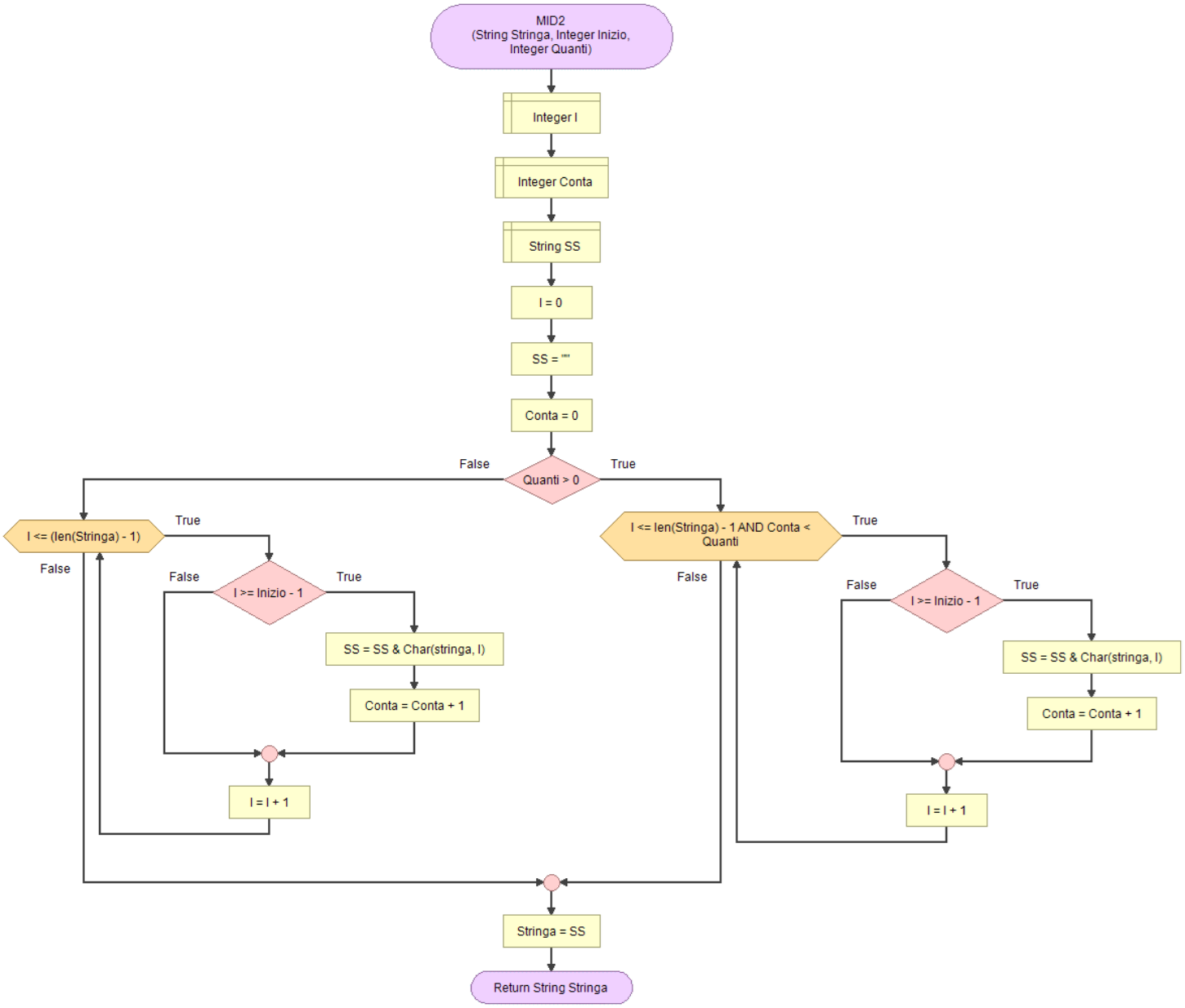
Naive Pattern Searching





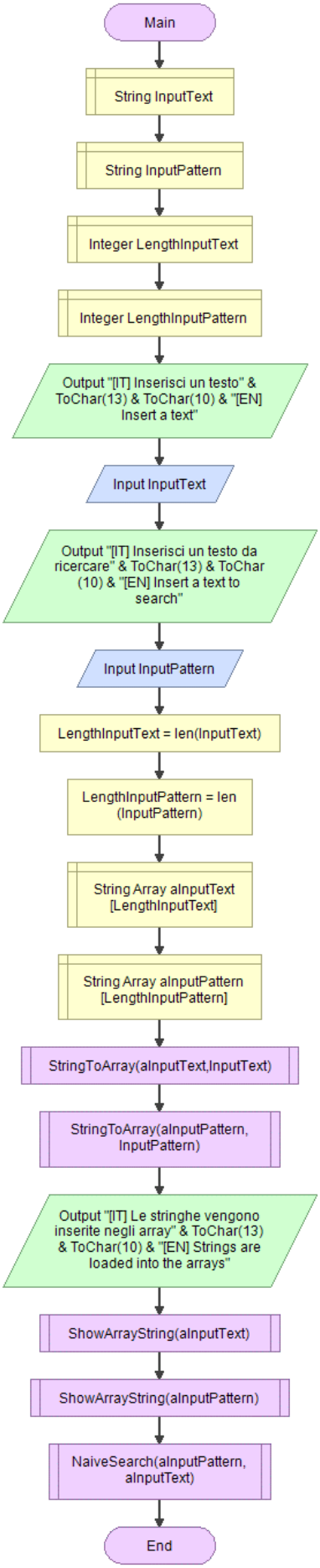


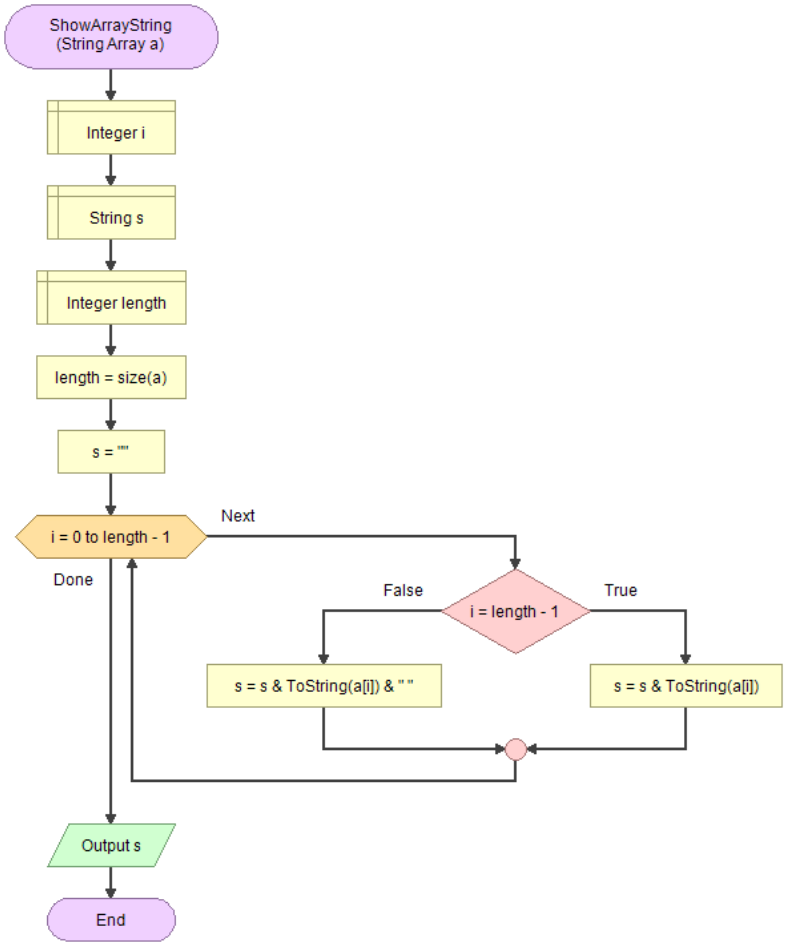

NaiveSearch
(String Array pat, String Array txt)
Integer M
Integer N
Integer i
Integer j
Boolean exit
M = size(pat)
N = size(txt)
A loop to slide pat one by one
i = 0 to N - M
Next
Done
For current index i, check for pattern match
exit = False
j = 0
j < M AND exit = False
True
False
txt[i+j] <> pat[j]
False
True
exit = True
j = j + 1
if pat[0...M-1] = txt[i, i+1, ...i+M-1]
j = M AND exit = False
False
True
Output "[IT] La sottostringa ricercata si trova alla posizione: " & ToString(i+1) & ToChar(13) & ToChar(10) & "[EN] Pattern found at pos: " & ToString(i+1)
End

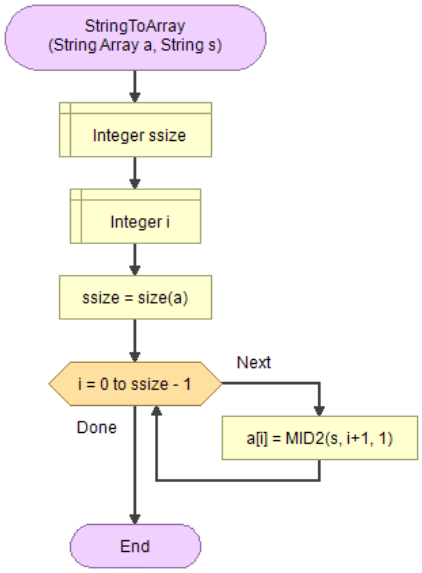

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

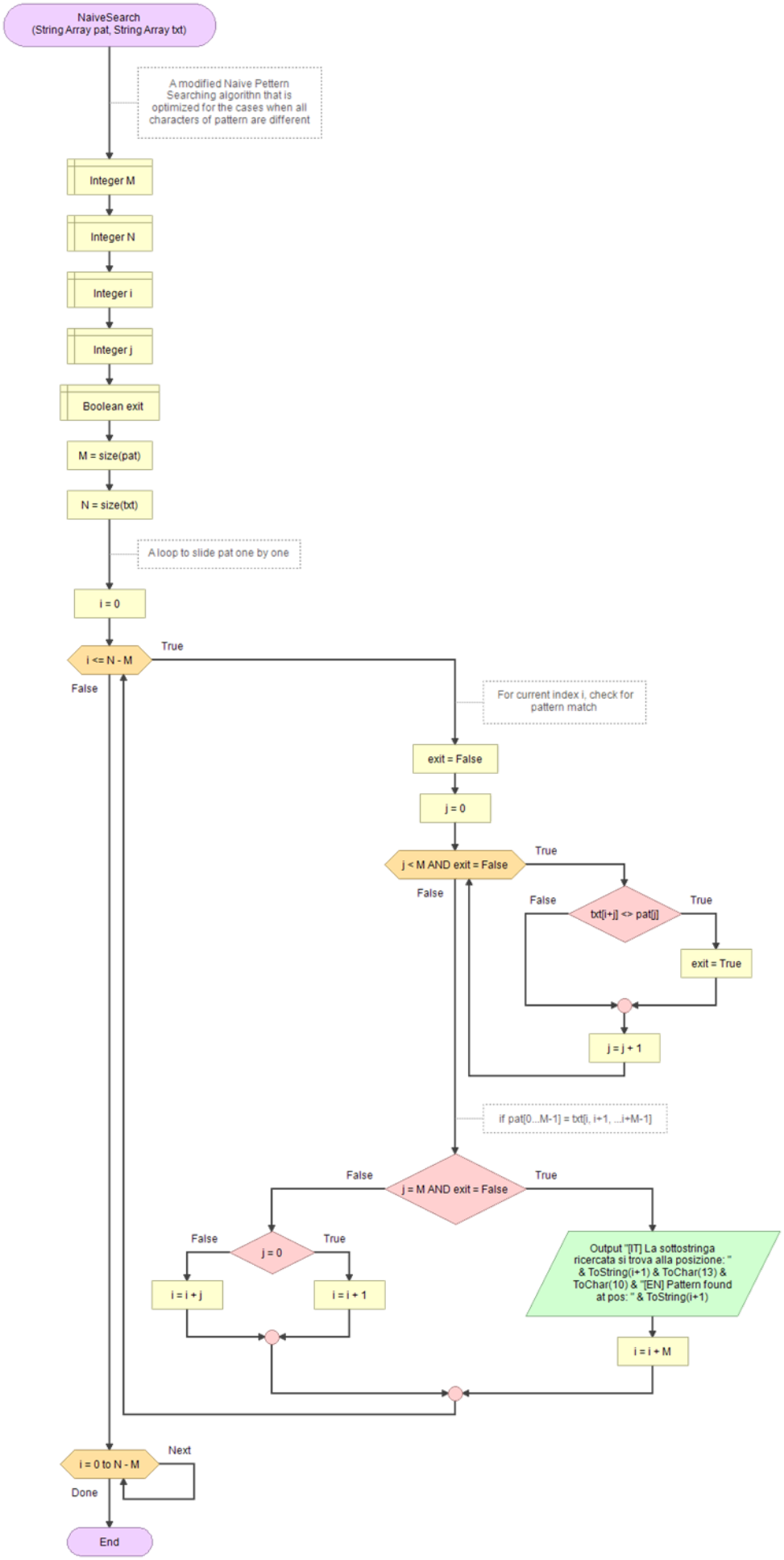
258.GSAF | Ricerca Naive ottimizzata

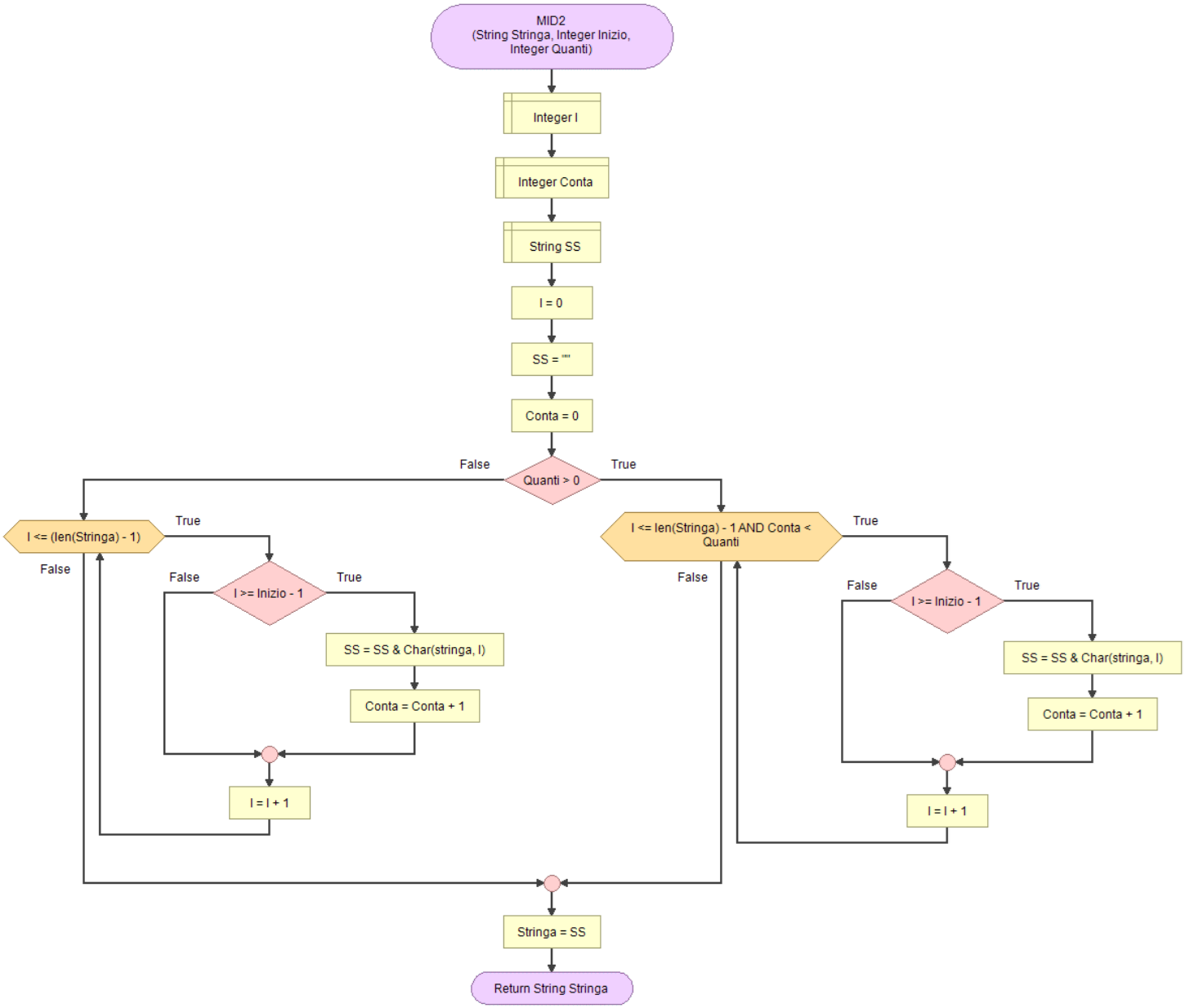
Optimized Naive Pattern Searching



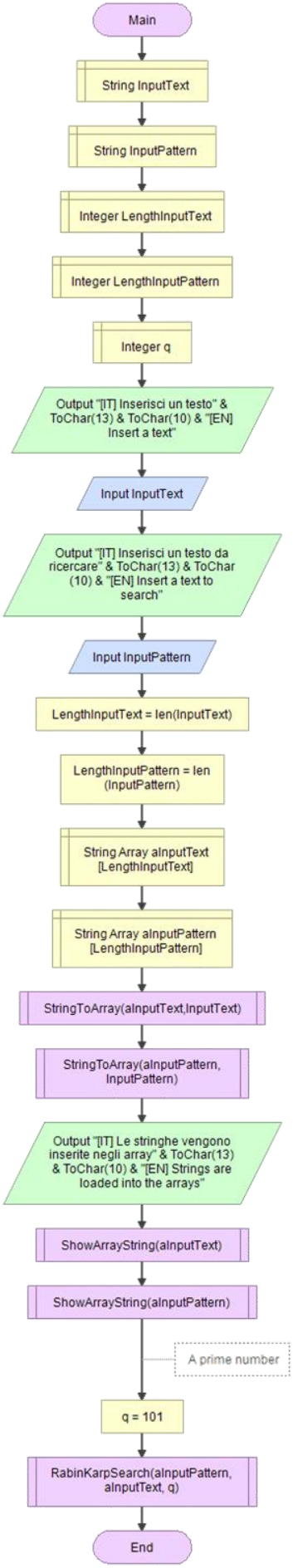


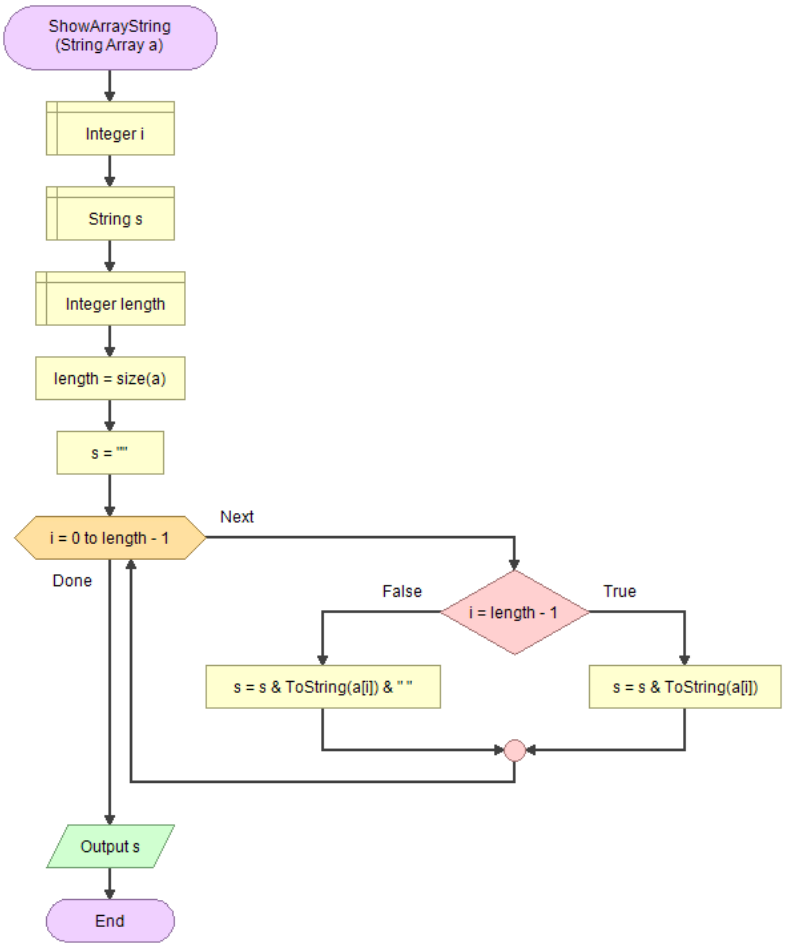


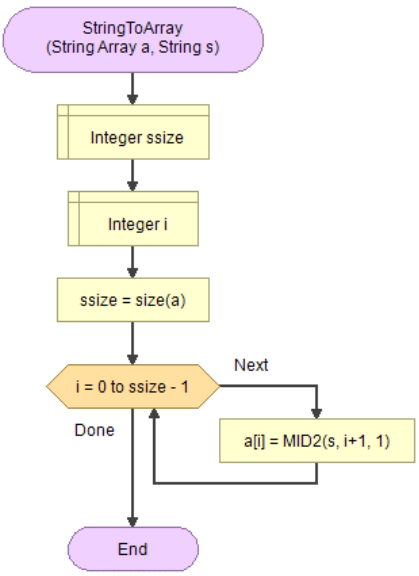

NaiveSearch
(String Array pat, String Array txt)
A modified Naive Pettern Searching algorithn that is optimized for the cases when all characters of pattern are different
Integer M
Integer N
Integer i
Integer j
Boolean exit
M = size(pat)
N = size(txt)
A loop to slide pat one by one
i = 0
i <= N - M
True
False
For current index i, check for pattern match
exit = False
j = 0
j < M AND exit = False
True
False
False
txt[i+j] <> pat[j]
True
exit = True
j = j + 1
if pat[0...M-1] = txt[i, i+1, ...i+M-1]
False
j = M AND exit = False
True
False
j = 0
True
i = i + j
i = i + 1
Output "[IT] La sottostringa ricercata si trova alla posizione: " & ToString(i+1) & ToChar(13) & ToChar(10) & "[EN] Pattern found at pos: " & ToString(i+1)
i = i + M
i = 0 to N - M
Next
Done
End

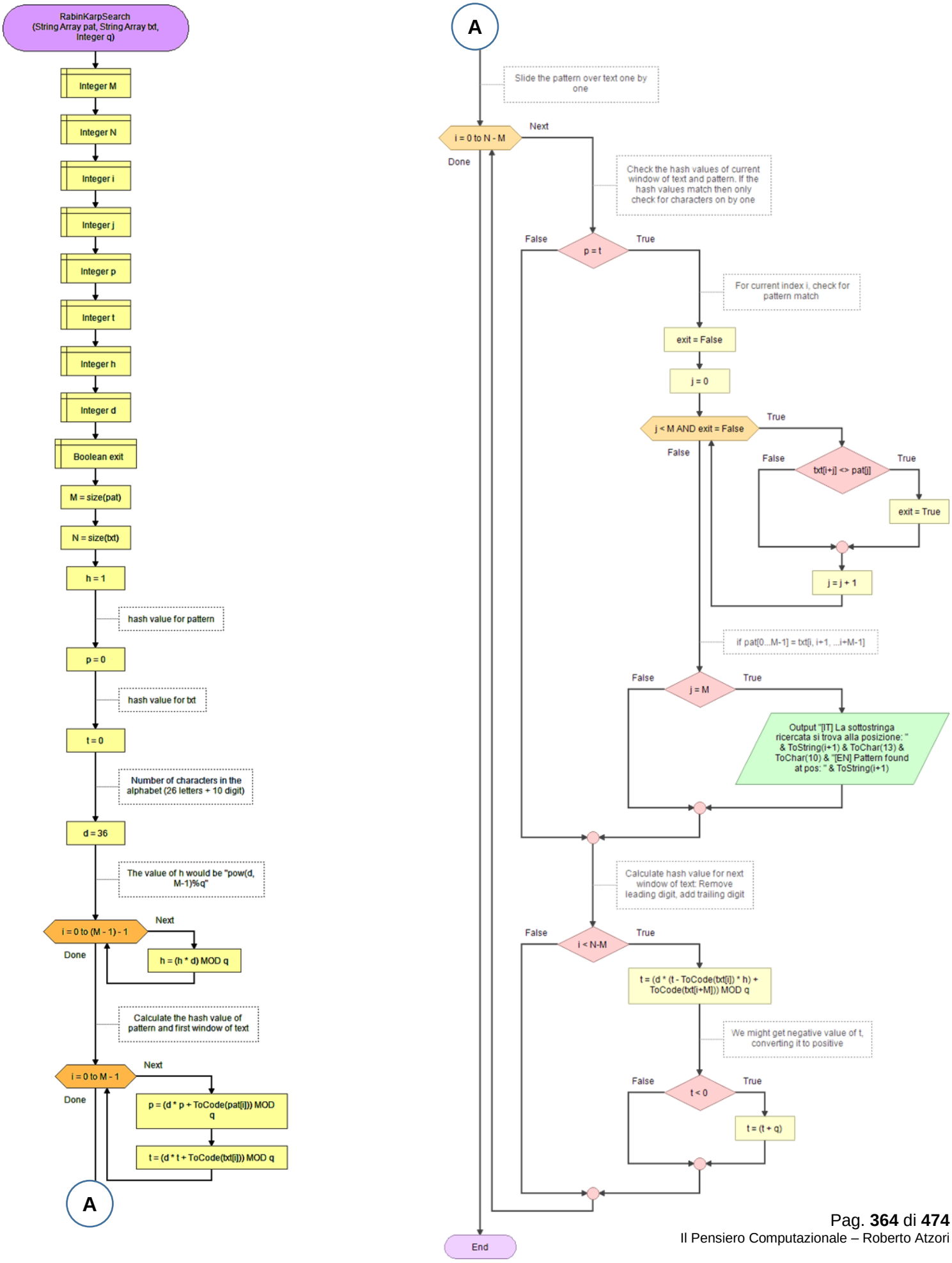

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

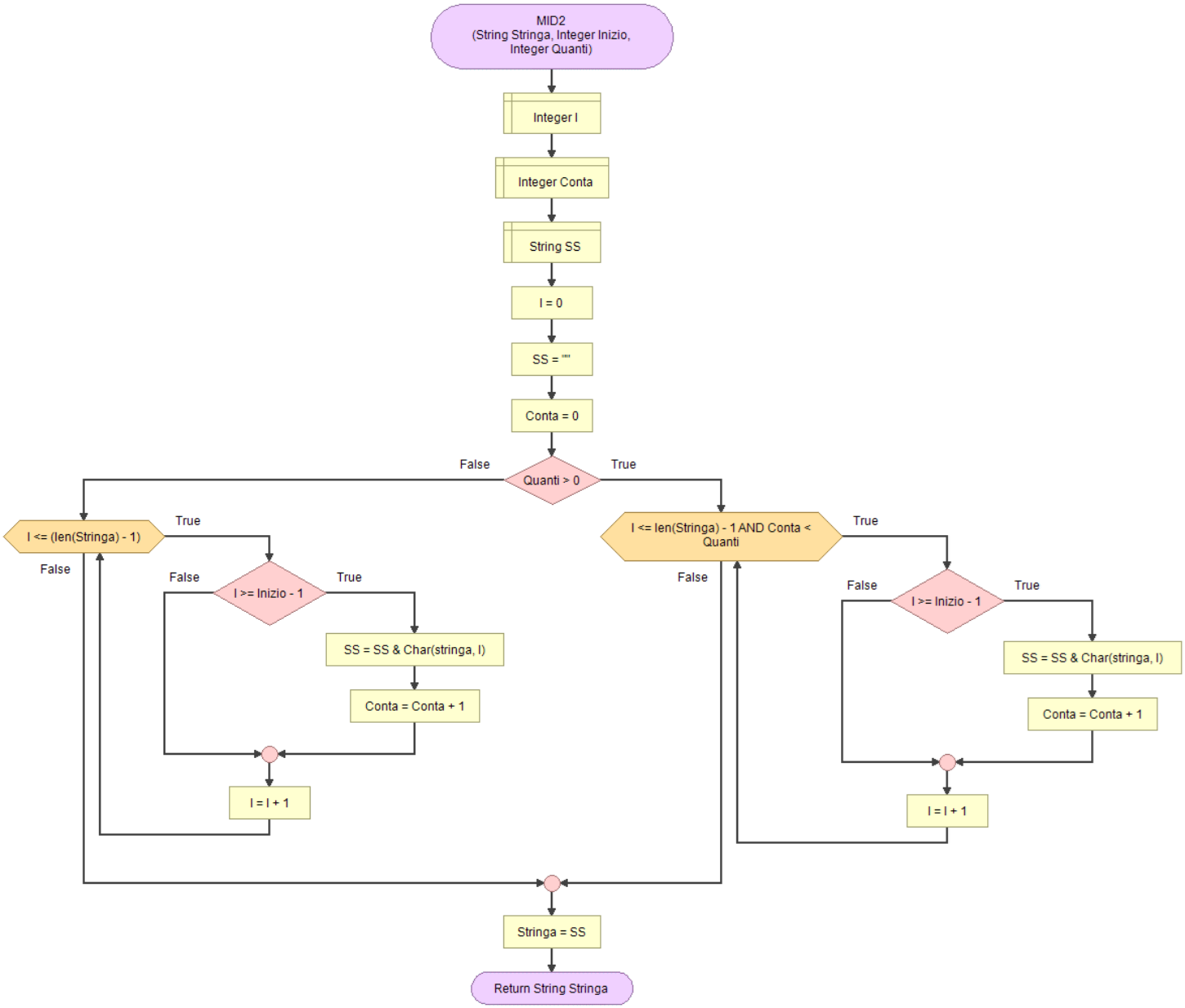
Rabin-Karp Pattern Searching





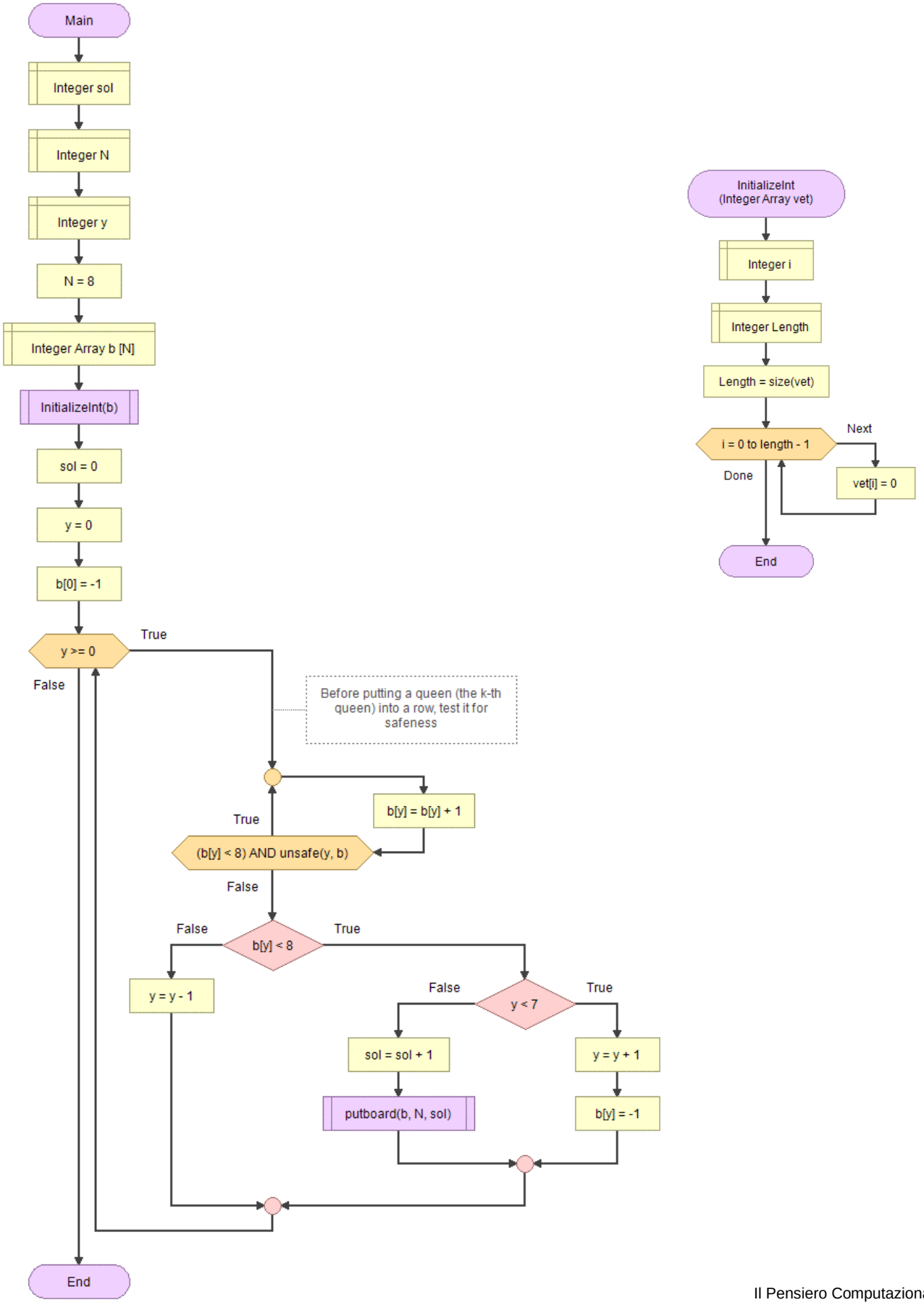


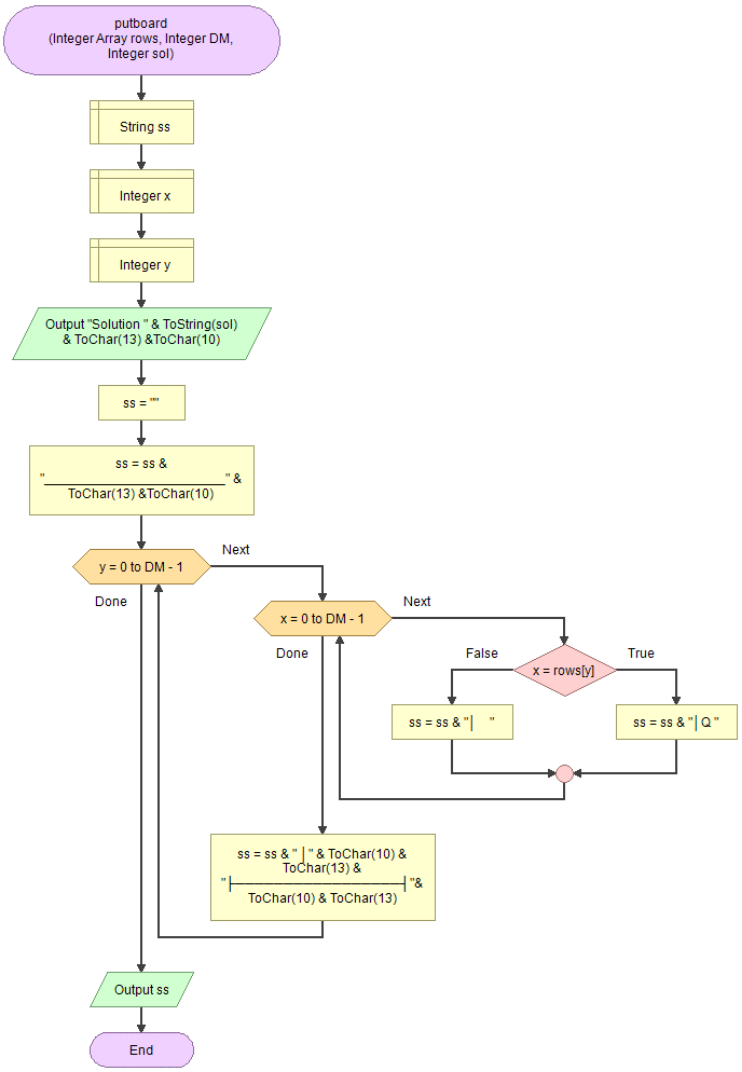

RabinKarpSearch
(String Array pat, String Array txt, Integer q)
Integer M
Integer N
Integer i
Integer j
Integer p
Integer t
Integer h
Integer d
Boolean exit
M = size(pat)
N = size(txt)
h = 1
hash value for pattern
p = 0
hash value for txt
t = 0
Number of characters in the alphabet (26 letters + 10 digit)
d = 36
The value of h would be "pow(d, M-1)%q"
i = 0 to (M - 1) - 1
Next
Done
h = (h * d) MOD q
Calculate the hash value of pattern and first window of text
i = 0 to M - 1
Next
Done
p = (d * p + ToCode(pat[i])) MOD q
t = (d * t + ToCode(txt[i])) MOD q
A
A
Slide the pattern over text one by one
i = 0 to N - M
Next
Done
Check the hash values of current window of text and pattern. If the hash values match then only check for characters on by one
False
p = t
True
For current index i, check for pattern match
exit = False
j = 0
j < M AND exit = False
True
False
False
txt[i+j] <> pat[j]
True
exit = True
j = j + 1
if pat[0...M-1] = txt[i, i+1, ...i+M-1]
False
j = M
True
Output "[IT] La sottostringa ricercata si trova alla posizione: " & ToString(i+1) & ToChar(13) & ToChar(10) & "[EN] Pattern found at pos: " & ToString(i+1)
Calculate hash value for next window of text: Remove leading digit, add trailing digit
False
i < N-M
True
t = (d * (t - ToCode(txt[i]) * h) + ToCode(txt[i+M])) MOD q
We might get negative value of t, converting it to positive
False
t < 0
True
t = (t + q)
End

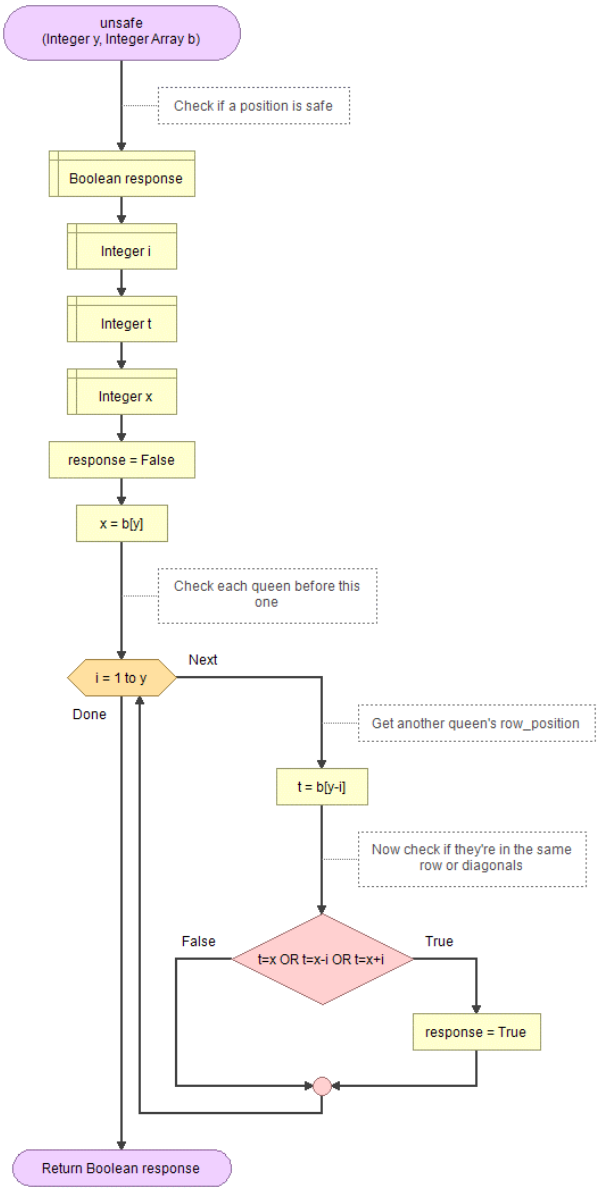

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta <
Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

260.GSAF | Problema delle 8 regine

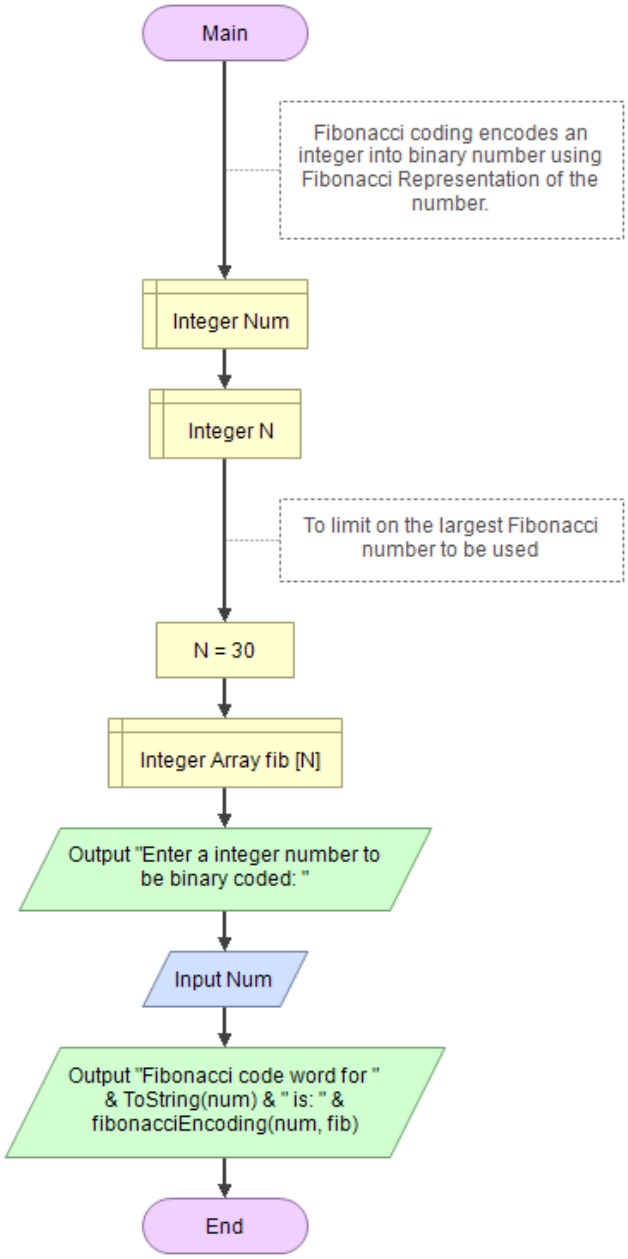
8 queens problem




putboard
(Integer Array rows, Integer DM, Integer sol)
String ss
Integer x
Integer y
Output "Solution " & ToString(sol) & ToChar(13) &ToChar(10)
ss = ""
ss = ss & "————————————" & ToChar(13) &ToChar(10)
y = 0 to DM - 1
Next
Done
x = 0 to DM - 1
Next
Done
False
x = rows[y]
True
ss = ss & "| "
ss = ss & "| Q "
ss = ss & " | " & ToChar(10) & ToChar(13) & "|————————————|"& ToChar(10) & ToChar(13)
Output ss
End


unsafe
(Integer y, Integer Array b)
Check if a position is safe
Boolean response
Integer i
Integer t
Integer x
response = False
x = b[y]
Check each queen before this one
i = 1 to y
Next
Done
Get another queen's row_position
t = b[y-i]
Now check if they're in the same row or diagonals
False
t=x OR t=x-i OR t=x+i
True
response = True
Return Boolean response

Fibonacci Coding



Fibonacci coding encodes an integer into binary number using Fibonacci Representation of the number. The idea is based on Zeckendorf's Theorem which states that every positive integer can be written uniquely as a sum of distinct non-neighbouring Fibonacci numbers (0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 141, ……..).

```
Fibonacci code word for 143 is 01010101011
```

Illustration

Suppose we start with n=**143.** The first **f** will be **89**. We mark it as used:

Fibonaccis	1	2	3	5	8	13	21	34	55	**89**	144
Usage bit	0	0	0	0	0	0	0	0	0	**1**	-

Now **n** = 143 - 89 = **54**. Fibonacci in hand is 55 which is > than 54. We mark it unused:

Fibonaccis	1	2	3	5	8	13	21	34	**55**	89	144
Usage bit	0	0	0	0	0	0	0	0	**0**	1	-

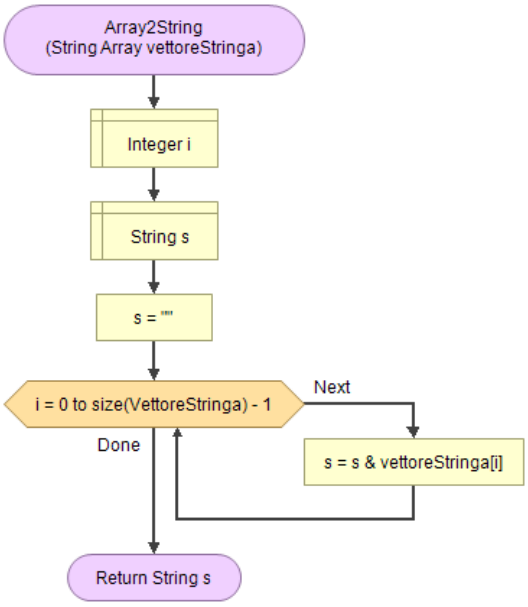
n = 54. f = 34. We mark it as used:

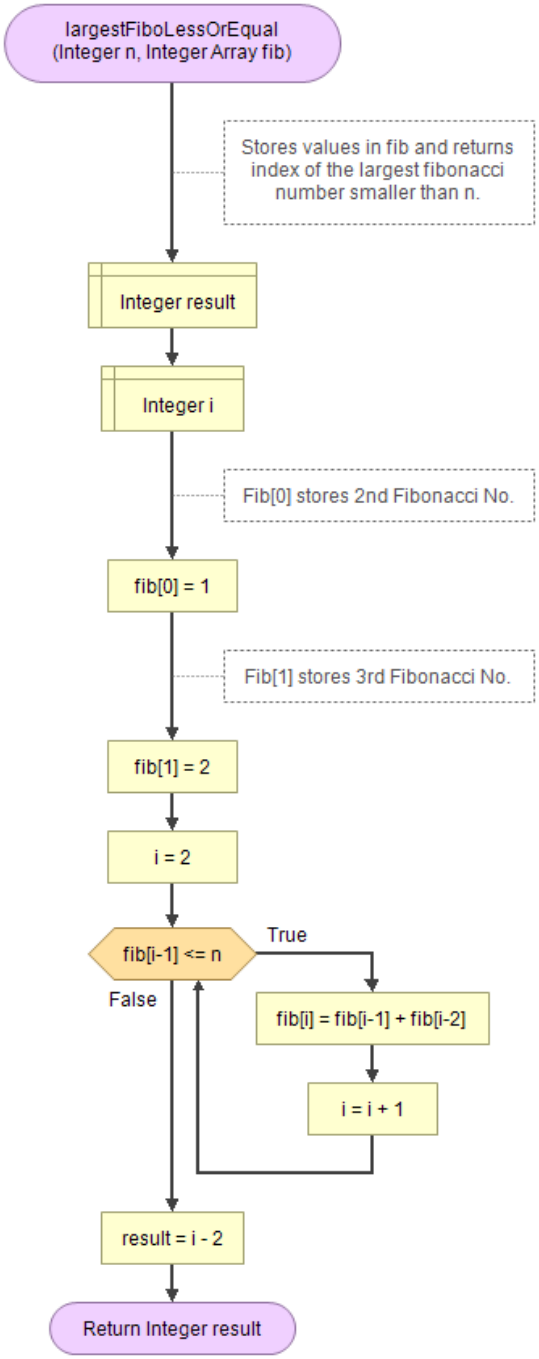
Fibonaccis	1	2	3	5	8	13	21	**34**	55	89	144
Usage bit	0	0	0	0	0	0	0	**1**	0	1	-

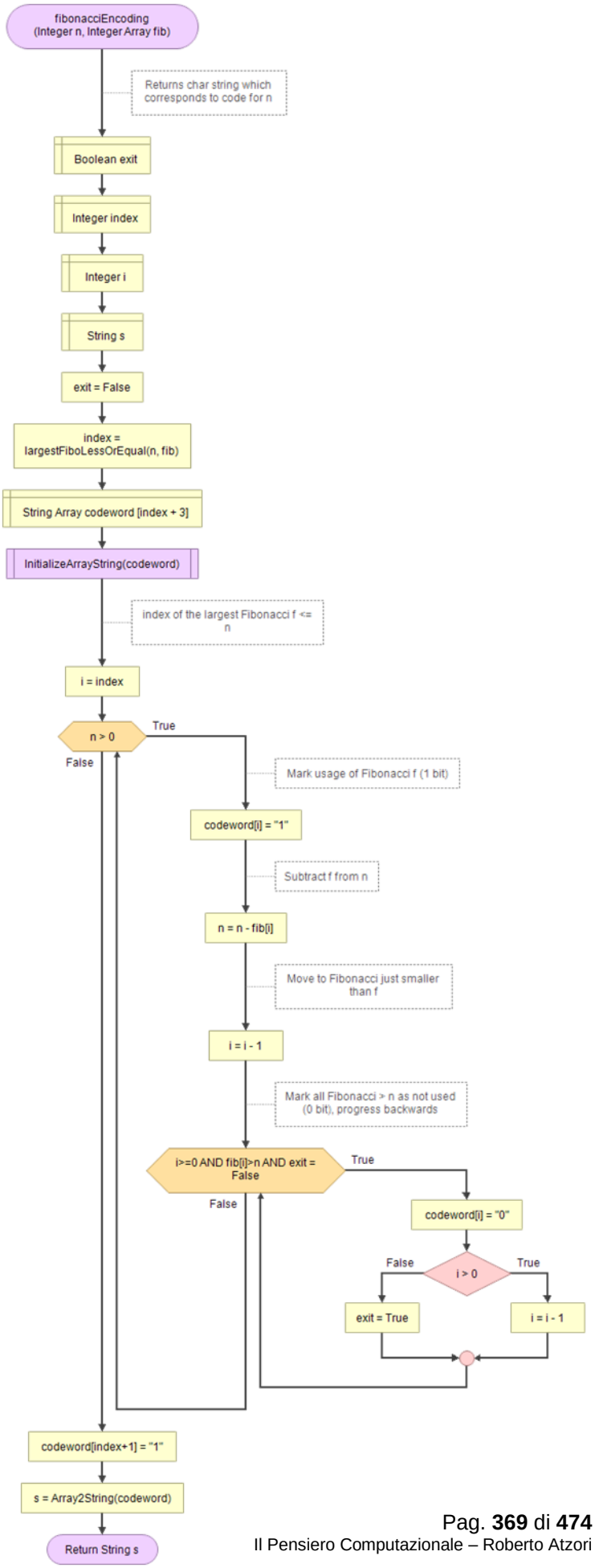
And finally to n = **0:**

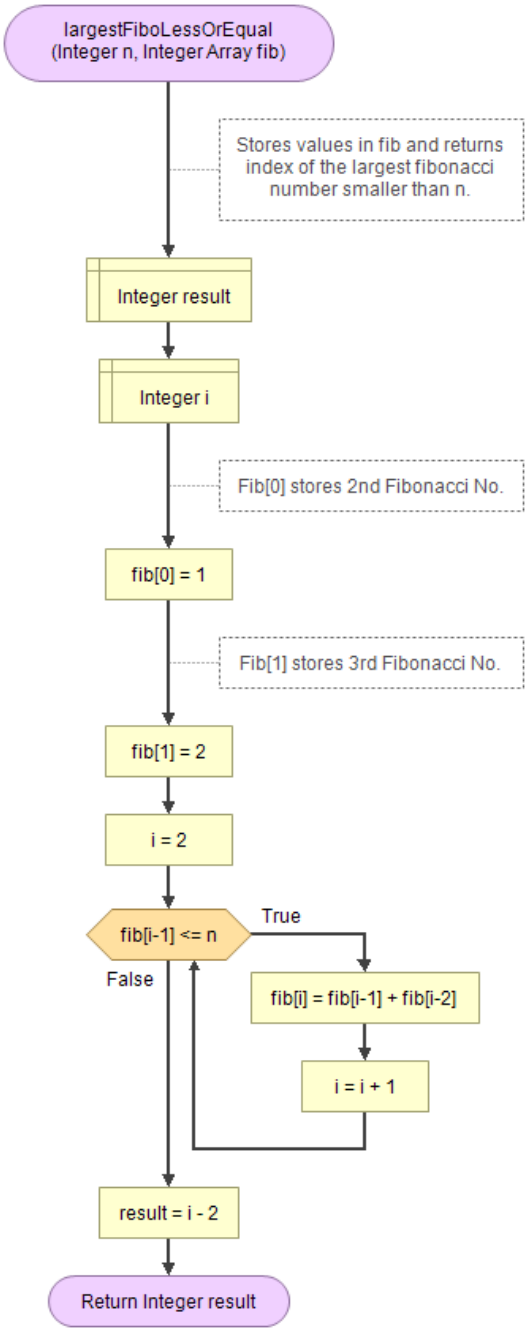
Fibonaccis	1	2	3	5	8	13	21	34	55	89	144
Usage bit	0	1	0	1	0	1	0	1	0	1	-

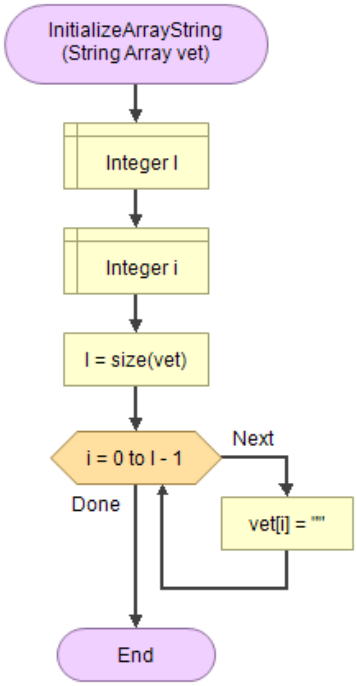
For the codeword, read the second row of above table from left to right: 0101010101
Append additional '1' bit: 01010101011
Final codeword for 143 = 01010101011




largestFiboLessOrEqual
(Integer n, Integer Array fib)
Stores values in fib and returns index of the largest fibonacci number smaller than n.
Integer result
Integer i
Fib[0] stores 2nd Fibonacci No.
fib[0] = 1
Fib[1] stores 3rd Fibonacci No.
fib[1] = 2
i = 2
fib[i-1] <= n
True
False
fib[i] = fib[i-1] + fib[i-2]
i = i + 1
result = i - 2
Return Integer result

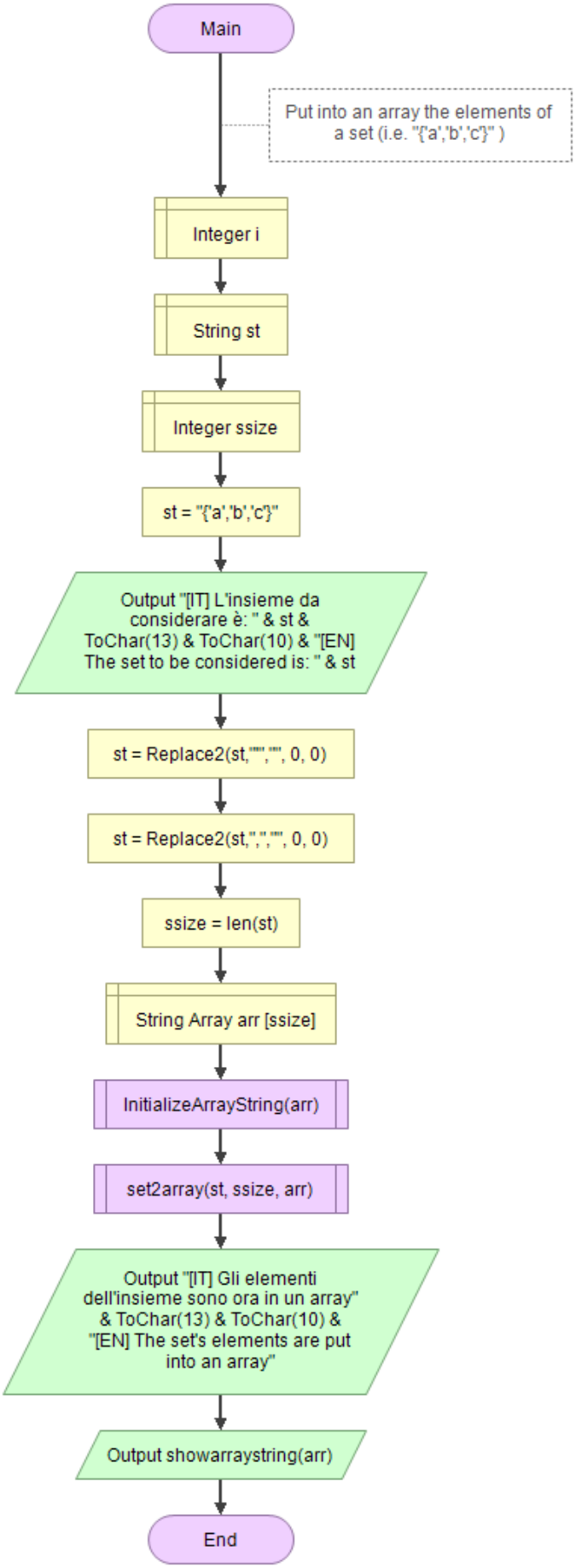

fibonacciEncoding
(Integer n, Integer Array fib)
Returns char string which corresponds to code for n
Boolean exit
Integer index
Integer i
String s
exit = False
index = largestFiboLessOrEqual(n, fib)
String Array codeword [index + 3]
InitializeArrayString(codeword)
index of the largest Fibonacci f <= n
i = index
n > 0
True
False
Mark usage of Fibonacci f (1 bit)
codeword[i] = "1"
Subtract f from n
n = n - fib[i]
Move to Fibonacci just smaller than f
i = i - 1
Mark all Fibonacci > n as not used (0 bit), progress backwards
i>=0 AND fib[i]>n AND exit = False
True
False
codeword[i] = "0"
i > 0
False
True
exit = True
i = i - 1
codeword[index+1] = "1"
s = Array2String(codeword)
Return String s

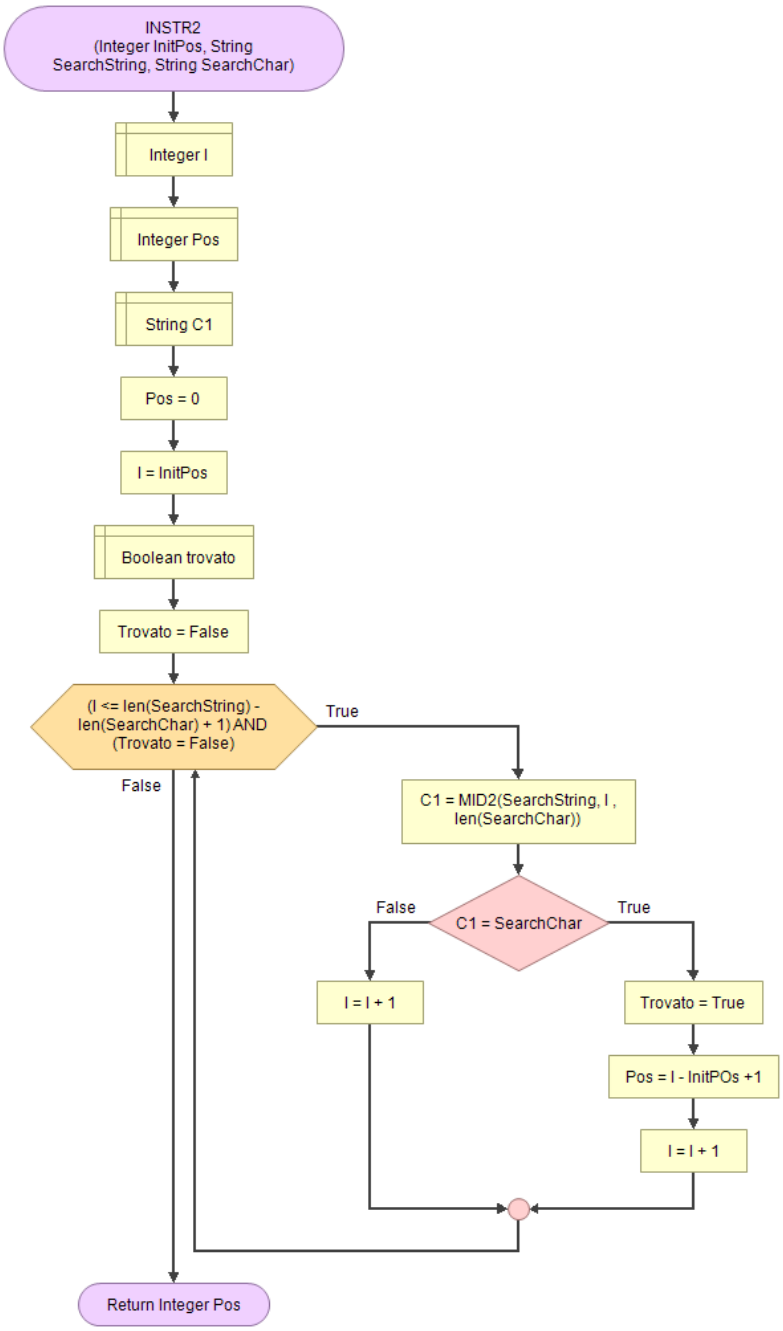

largestFiboLessOrEqual
(Integer n, Integer Array fib)
Stores values in fib and returns index of the largest fibonacci number smaller than n.
Integer result
Integer i
Fib[0] stores 2nd Fibonacci No.
fib[0] = 1
Fib[1] stores 3rd Fibonacci No.
fib[1] = 2
i = 2
fib[i-1] <= n
True
False
fib[i] = fib[i-1] + fib[i-2]
i = i + 1
result = i - 2
Return Integer result

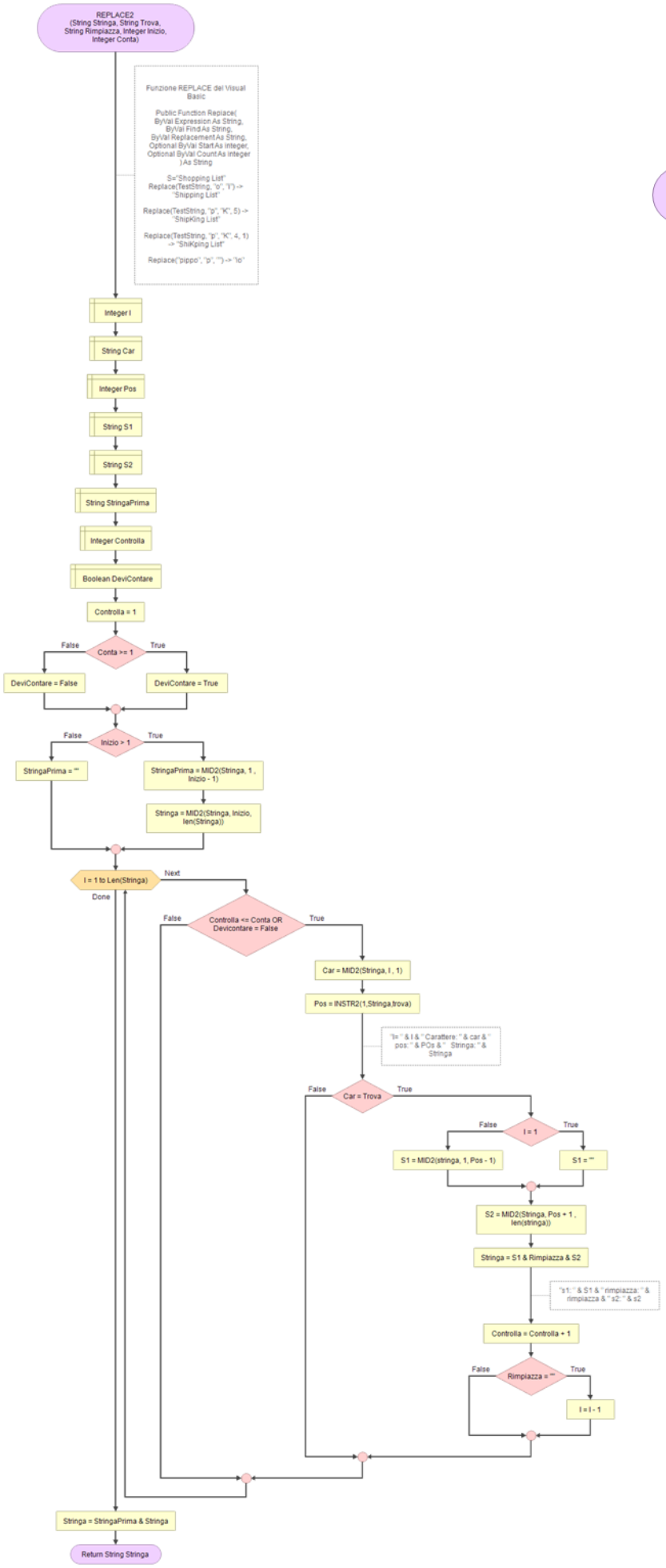

InitializeArrayString
(String Array vet)
Integer l
Integer i
l = size(vet)
i = 0 to l - 1
Next
Done
vet[i] = ""
End

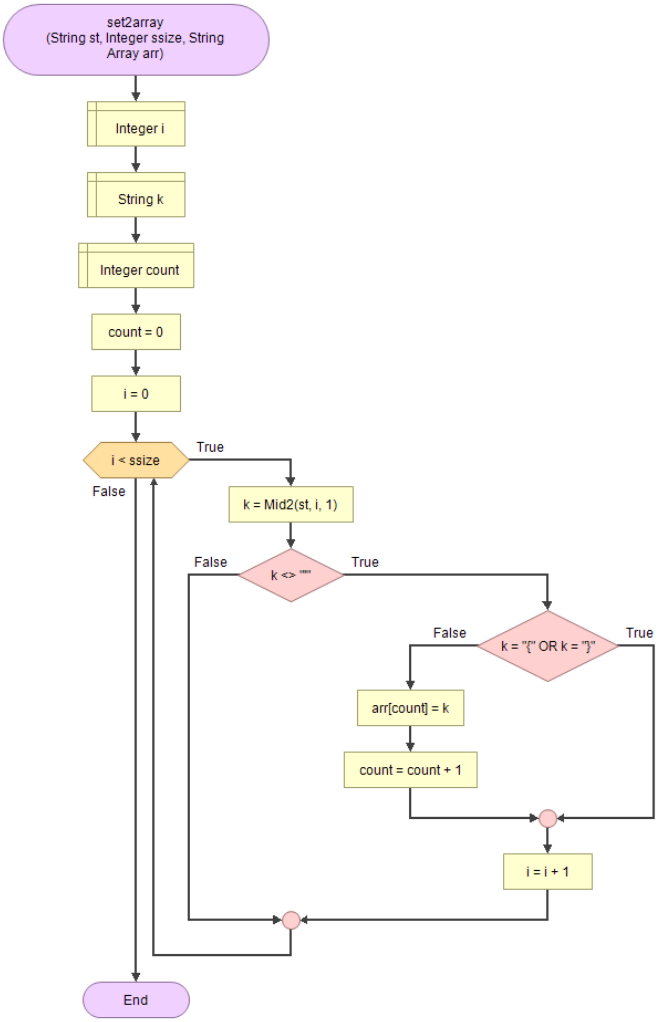
262.GSAF | Inserire in un array gli elementi di un insieme

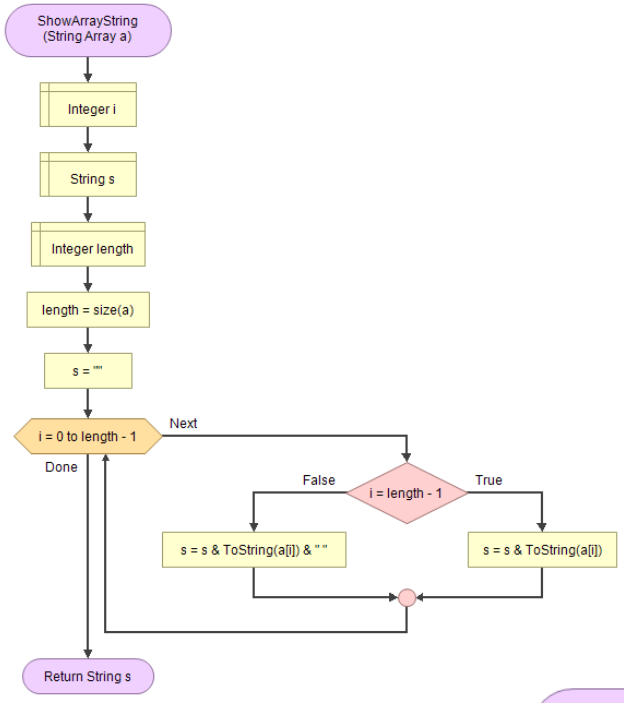
Put into an array the elements of a set

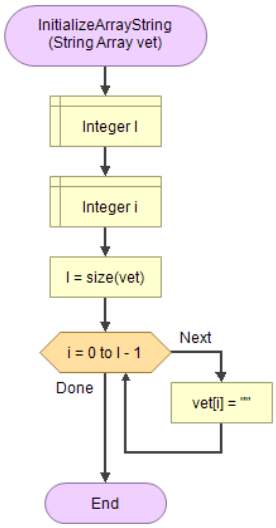


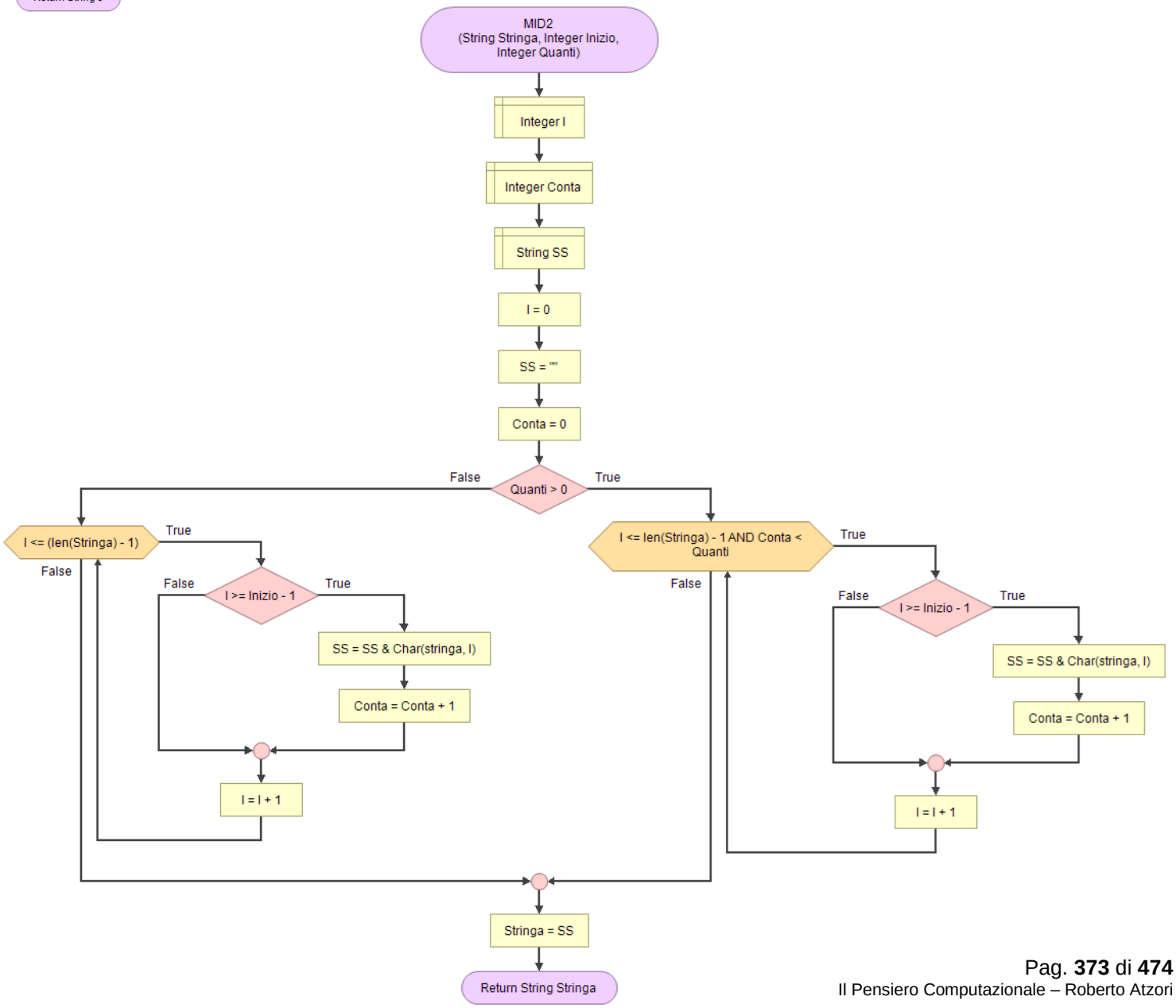



REPLACE2
(String Stringa, String Trova, String Rimpiazza, Integer Inizio, Integer Conta)
Funzione REPLACE del Visual Basic
Public Function Replace(ByVal Expression As String, ByVal Find As String, ByVal Replacement As String, Optional ByVal Start As Integer, Optional ByVal Count As Integer) As String
S="Shopping List"
Replace(TestString, "o", "I") -> "Shipping List"
Replace(TestString, "p", "K", 5) -> "ShipKing List"
Replace(TestString, "p", "K", 4, 1) -> "ShiKping List"
Replace("pippo", "p", "") -> "io"
Integer I
String Car
Integer Pos
String S1
String S2
String StringaPrima
Integer Controlla
Boolean DeviContare
Controlla = 1
Conta >= 1
False
True
DeviContare = False
DeviContare = True
Inizio > 1
StringaPrima = ""
StringaPrima = MID2(Stringa, 1 , Inizio - 1)
Stringa = MID2(Stringa, Inizio, len(Stringa))
I = 1 to Len(Stringa)
Next
Done
Controlla <= Conta OR Devicontare = False
Car = MID2(Stringa, I , 1)
Pos = INSTR2(1,Stringa,trova)
"I=" & I & " Carattere: " & car & " pos: " & POs & " Stringa: " & Stringa
Car = Trova
I = 1
S1 = MID2(stringa, 1, Pos - 1)
S1 = ""
S2 = MID2(Stringa, Pos + 1 , len(stringa))
Stringa = S1 & Rimpiazza & S2
"s1: " & S1 & " rimpiazza: " & rimpiazza & " s2: " & s2
Controlla = Controlla + 1
Rimpiazza = ""
I = I - 1
Stringa = StringaPrima & Stringa
Return String Stringa


set2array
(String st, Integer ssize, String Array arr)
Integer i
String k
Integer count
count = 0
i = 0
i < ssize
True
False
k = Mid2(st, i, 1)
k <> ""
k = "{" OR k = "}"
arr[count] = k
count = count + 1
i = i + 1
End

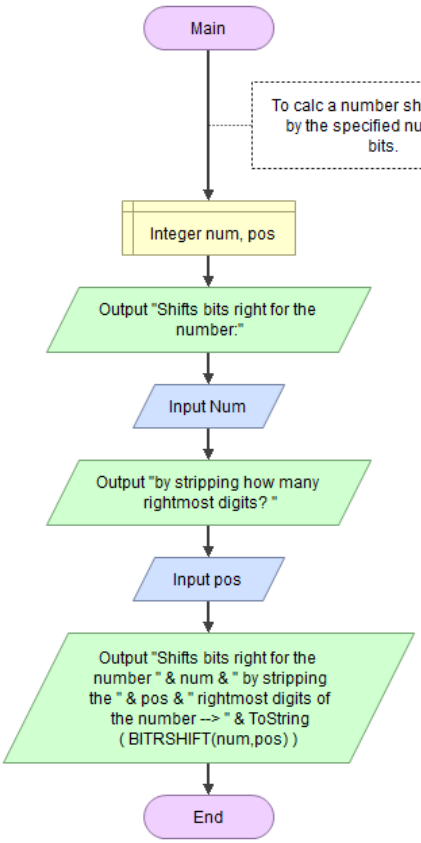

ShowArrayString
(String Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
i = length - 1
False
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Return String s

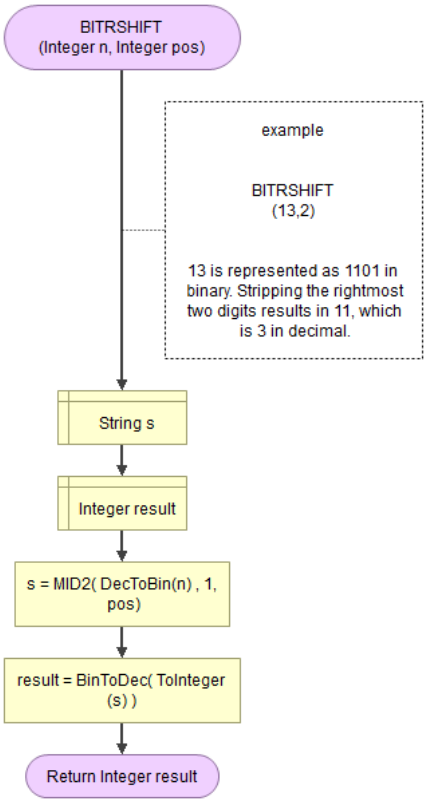

InitializeArrayString
(String Array vet)
Integer l
Integer i
l = size(vet)
i = 0 to l - 1
Next
Done
vet[i] = ""
End

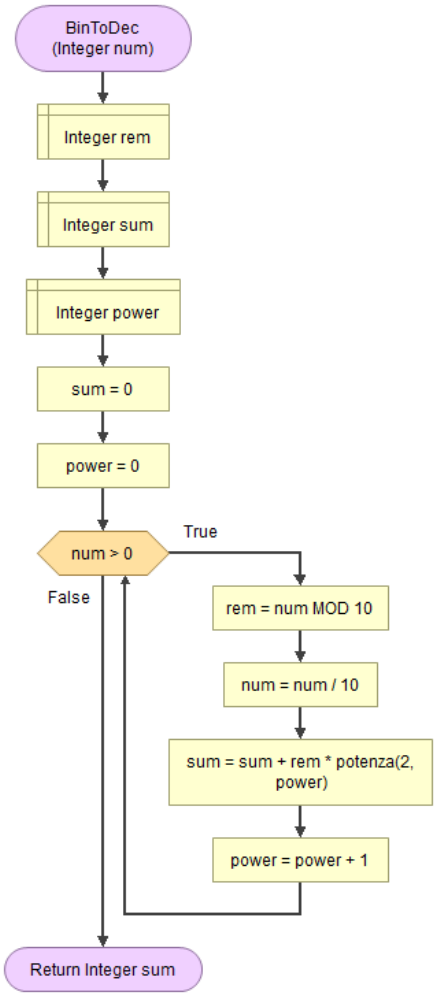

MID2
(String Stringa, Integer Inizio,
Integer Quanti)
Integer l
Integer Conta
String SS
l = 0
SS = ""
Conta = 0
Quanti > 0
False
True
l <= (len(Stringa) - 1)
True
False
l >= Inizio - 1
False
True
SS = SS & Char(stringa, l)
Conta = Conta + 1
l = l + 1
l <= len(Stringa) - 1 AND Conta <
Quanti
True
False
l >= Inizio - 1
False
True
SS = SS & Char(stringa, l)
Conta = Conta + 1
l = l + 1
Stringa = SS
Return String Stringa

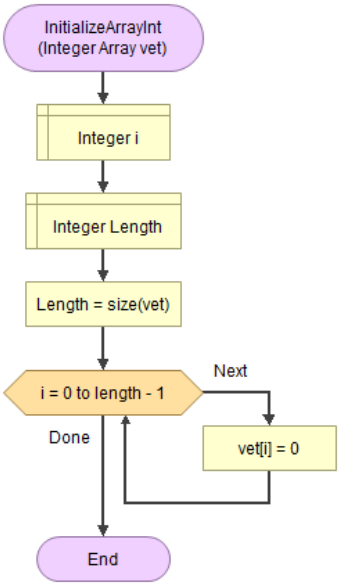
263.GSAFR | Spostamento a destra dei bit di un numero intero per un numero di posizioni specificato

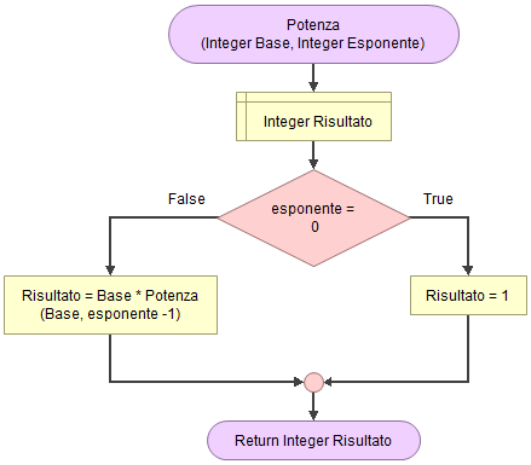
Shifts bits right for the number

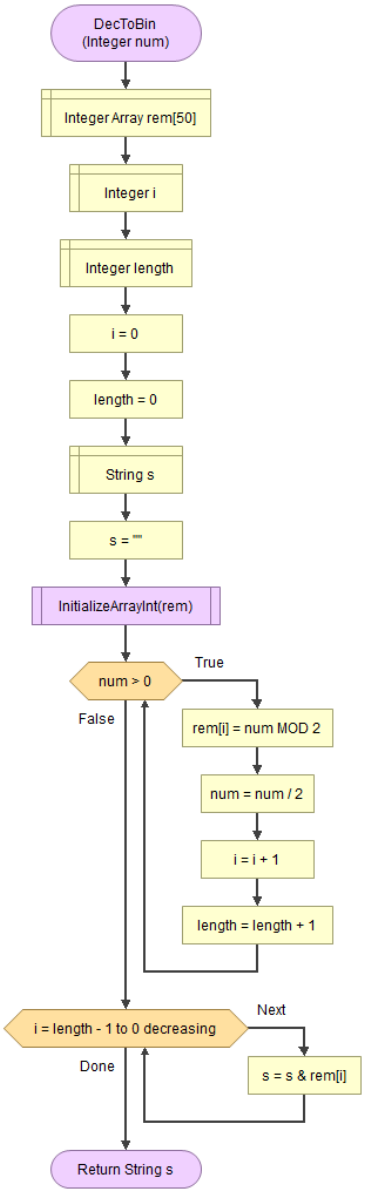


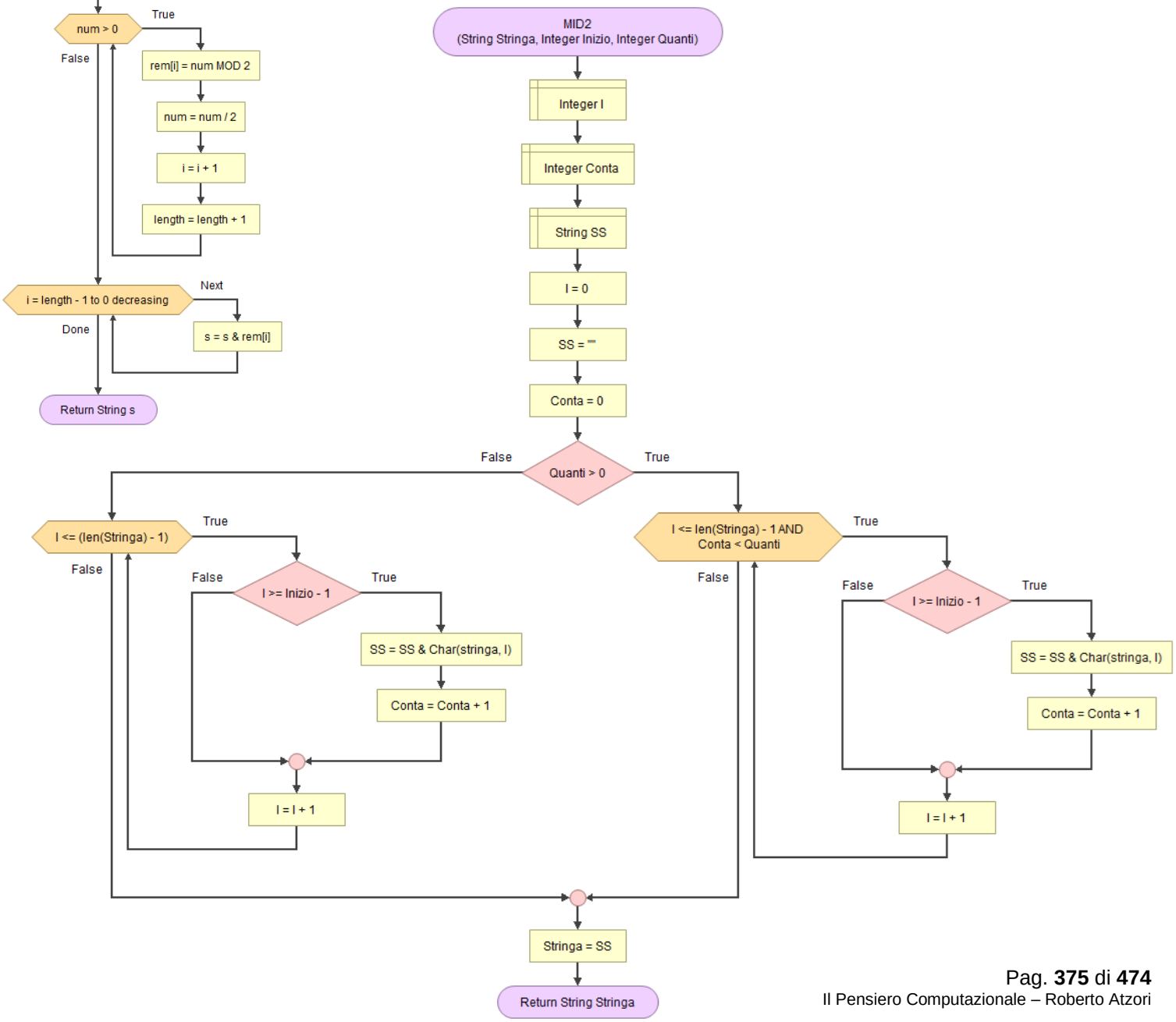






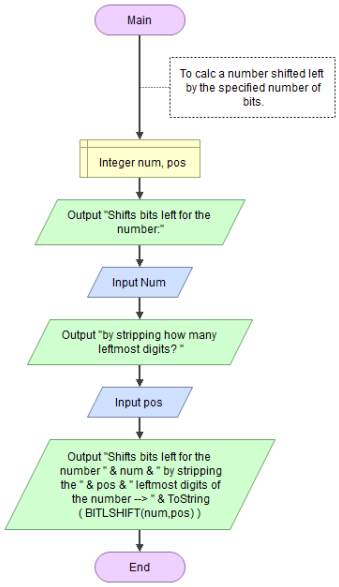


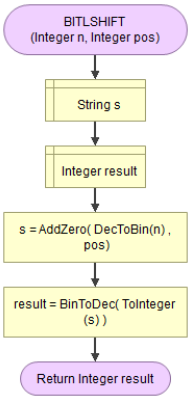

DecToBin
(Integer num)
Integer Array rem[50]
Integer i
Integer length
i = 0
length = 0
String s
s = ""
InitializeArrayInt(rem)
num > 0
True
False
rem[i] = num MOD 2
num = num / 2
i = i + 1
length = length + 1
i = length - 1 to 0 decreasing
Next
Done
s = s & rem[i]
Return String s

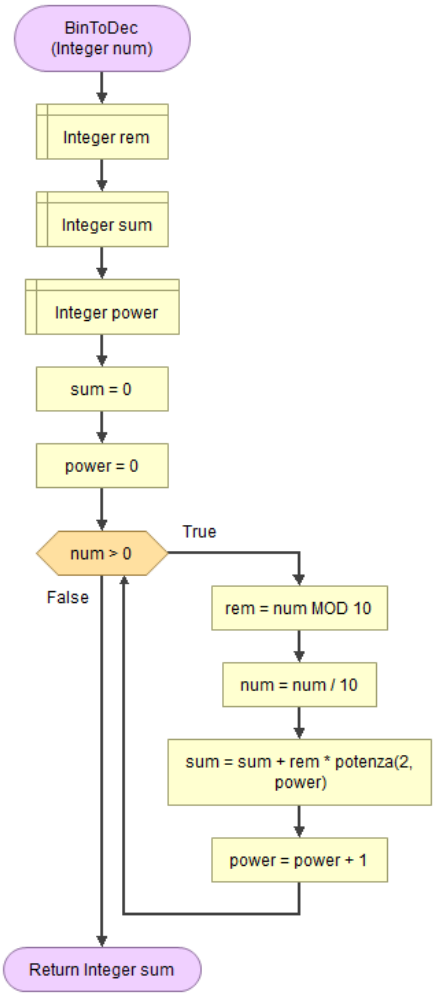

MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND
Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

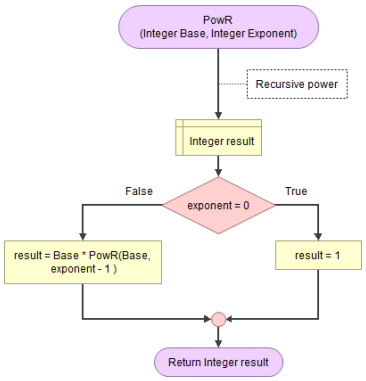
264.GSAFR | Spostamento a sinistra dei bit di un numero intero per un numero di posizioni specificato

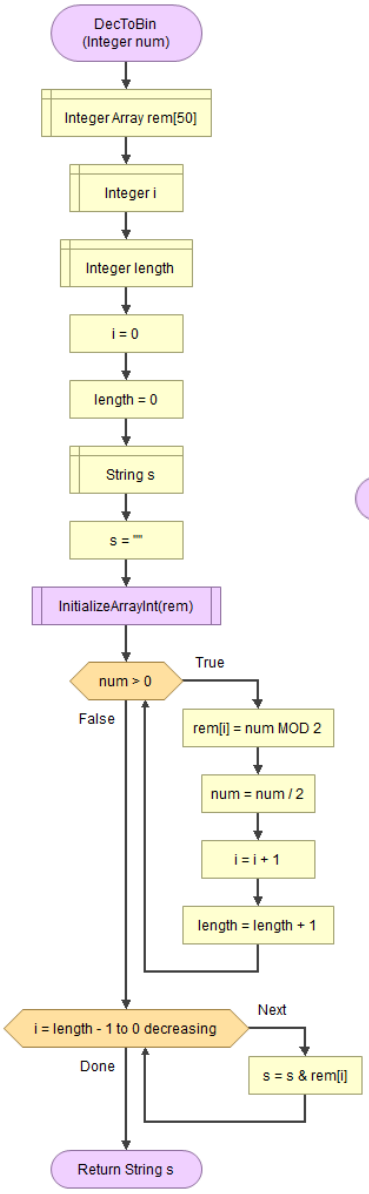
Shifts bits left for the number

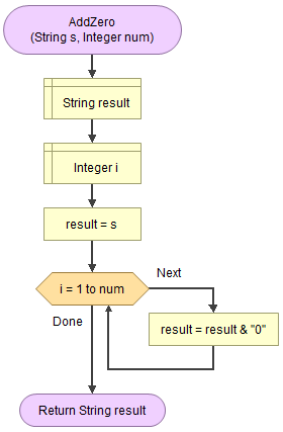


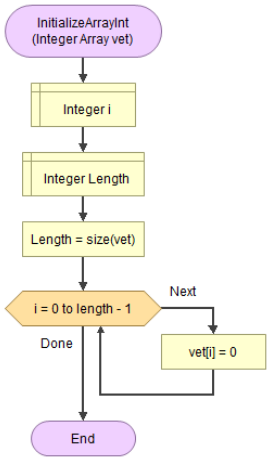






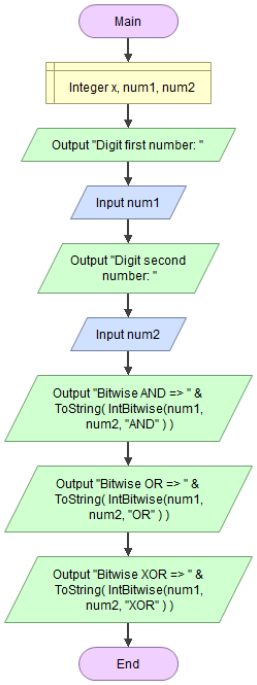


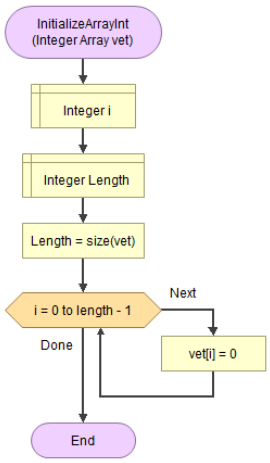


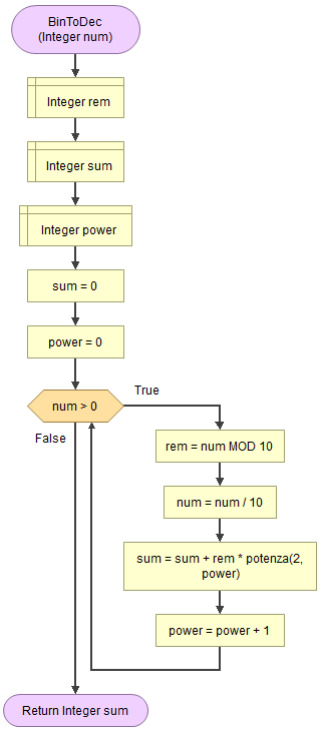


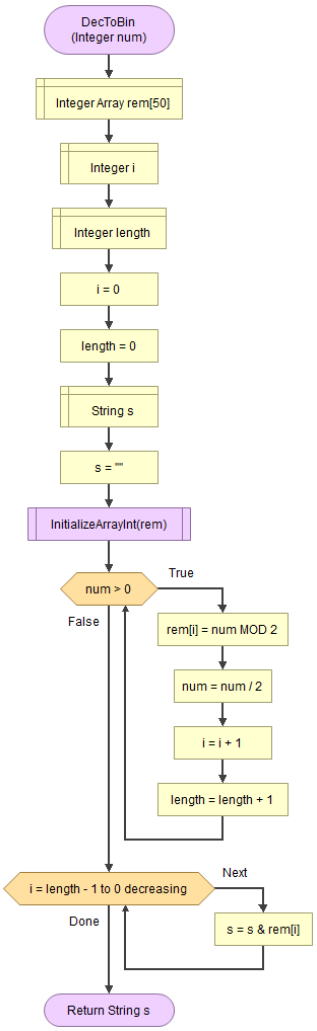
265.GSAFR | Operatori a livello del bit (AND, OR, XOR) con interi

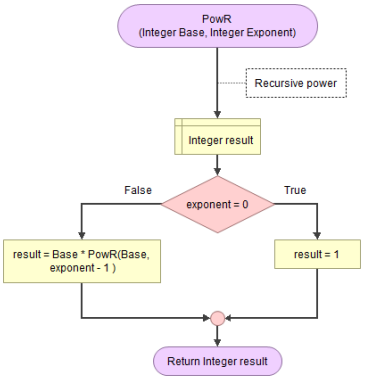
Integer bitwise AND OR XOR

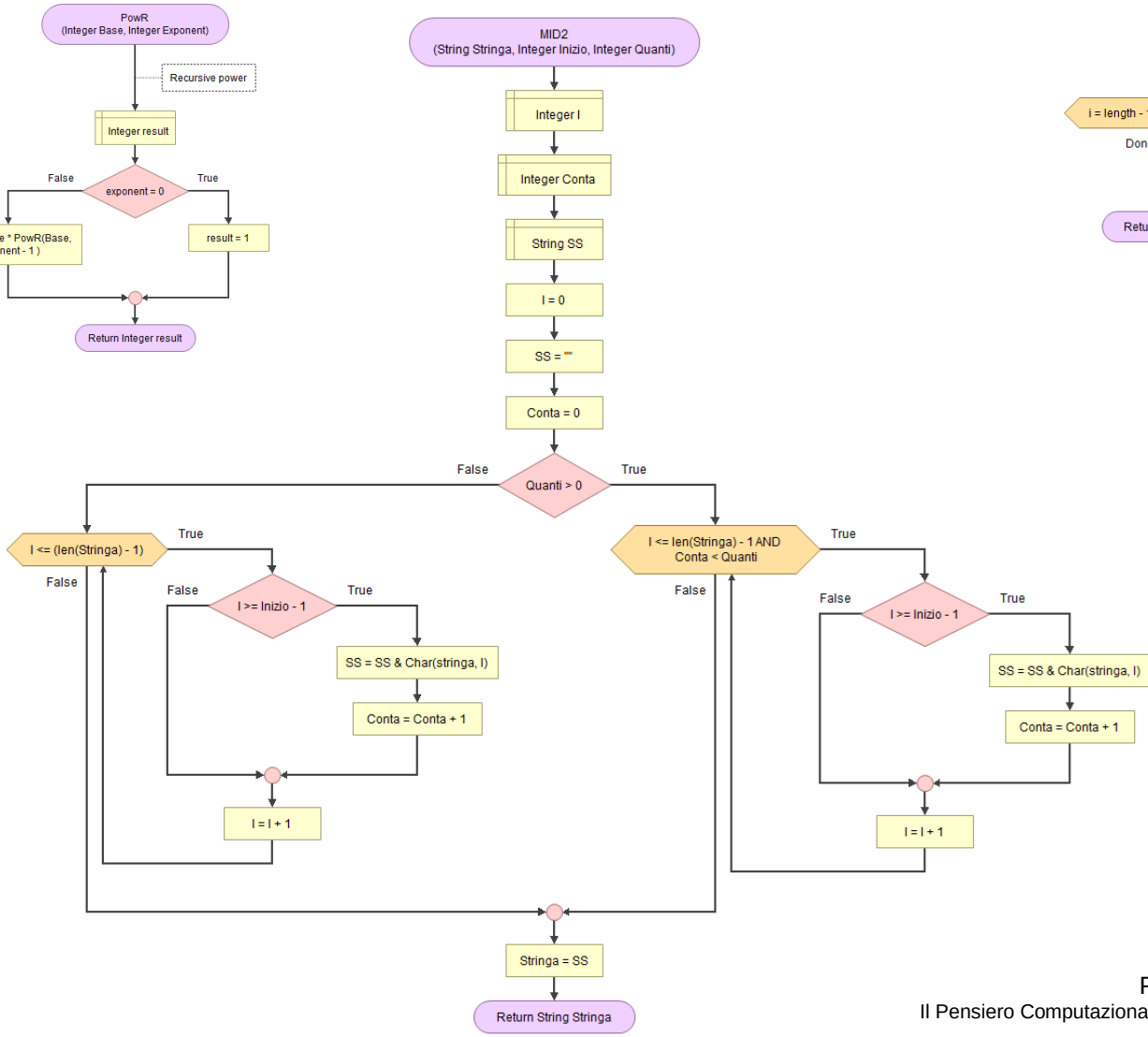


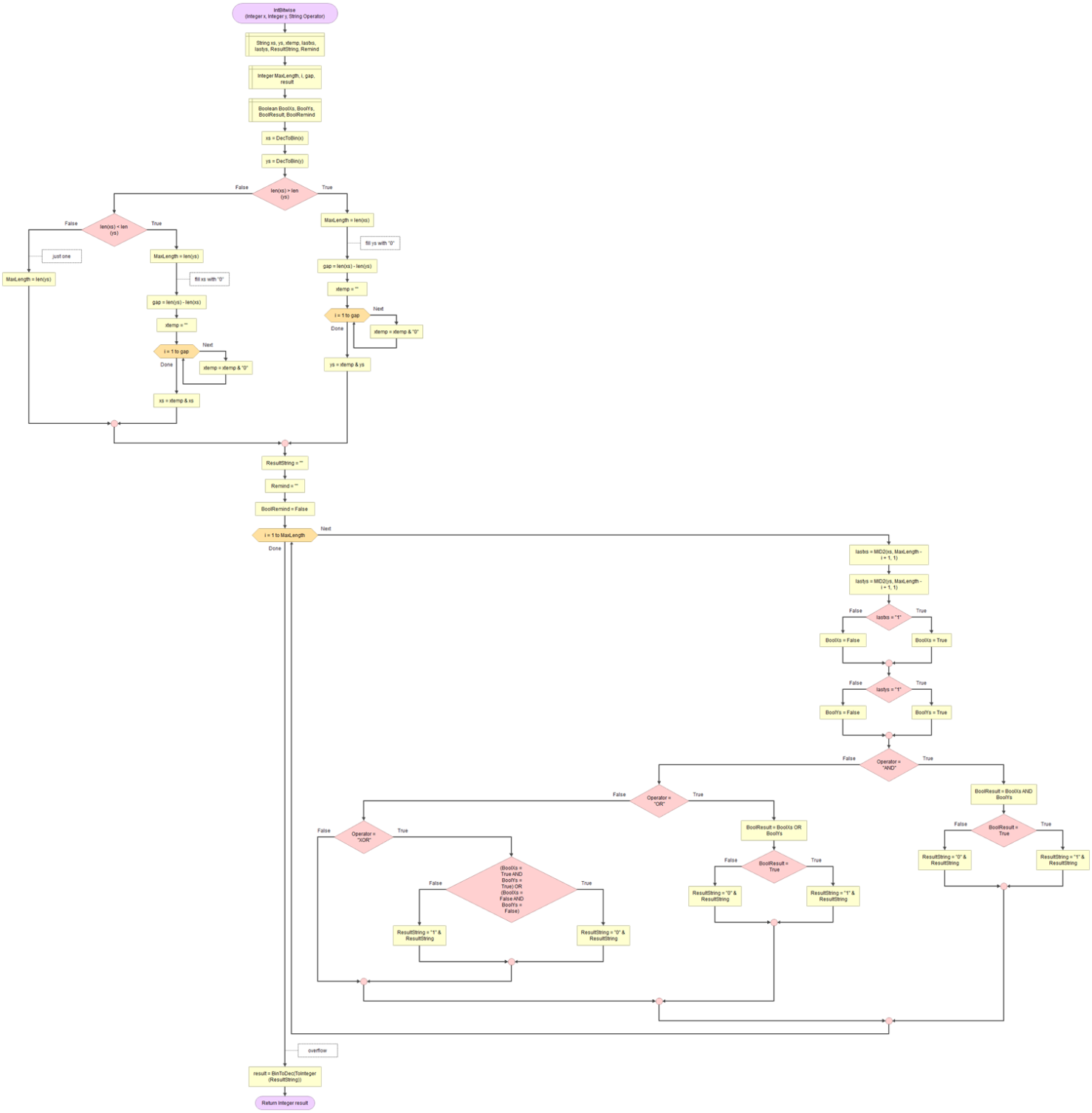






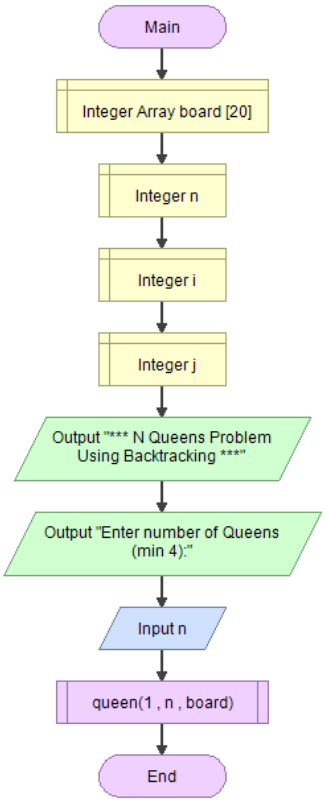


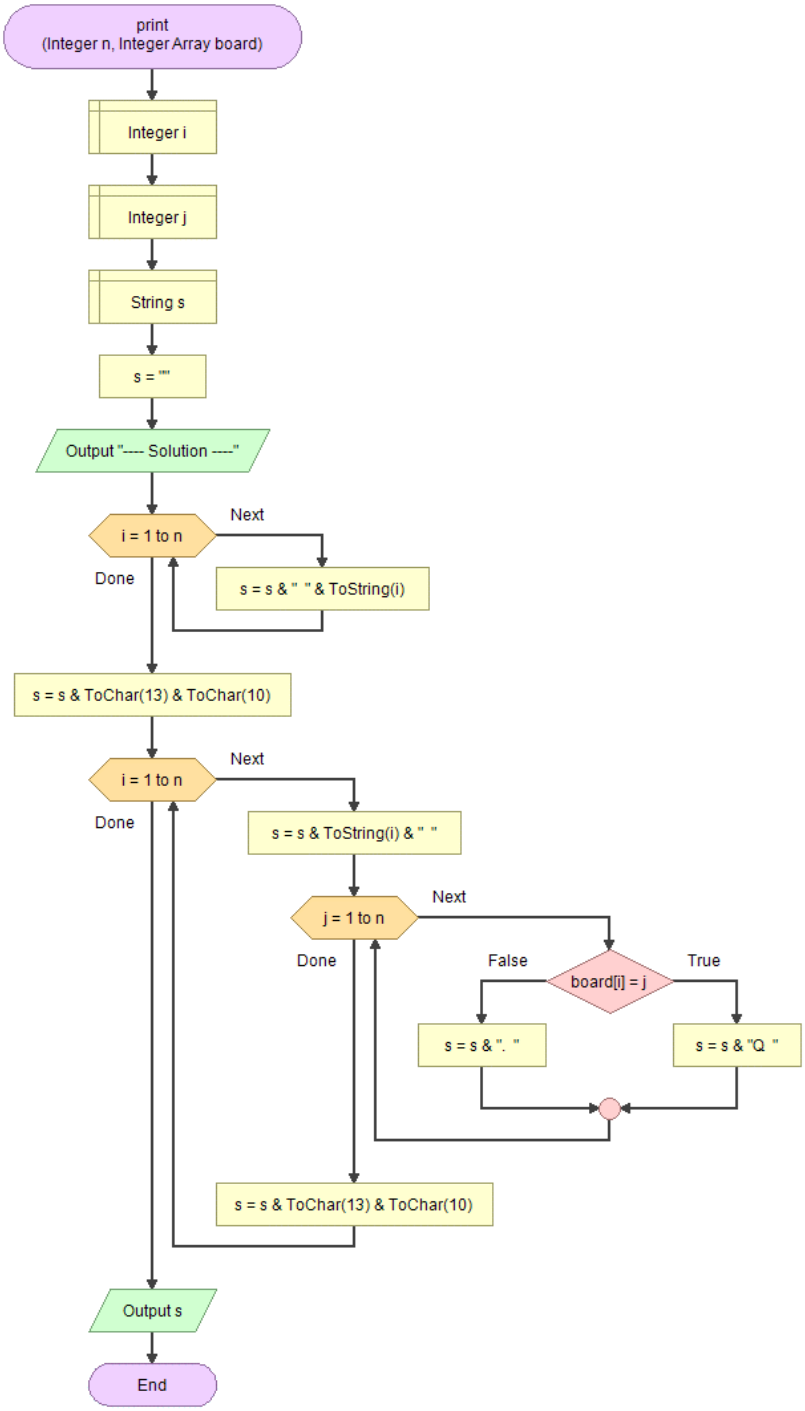


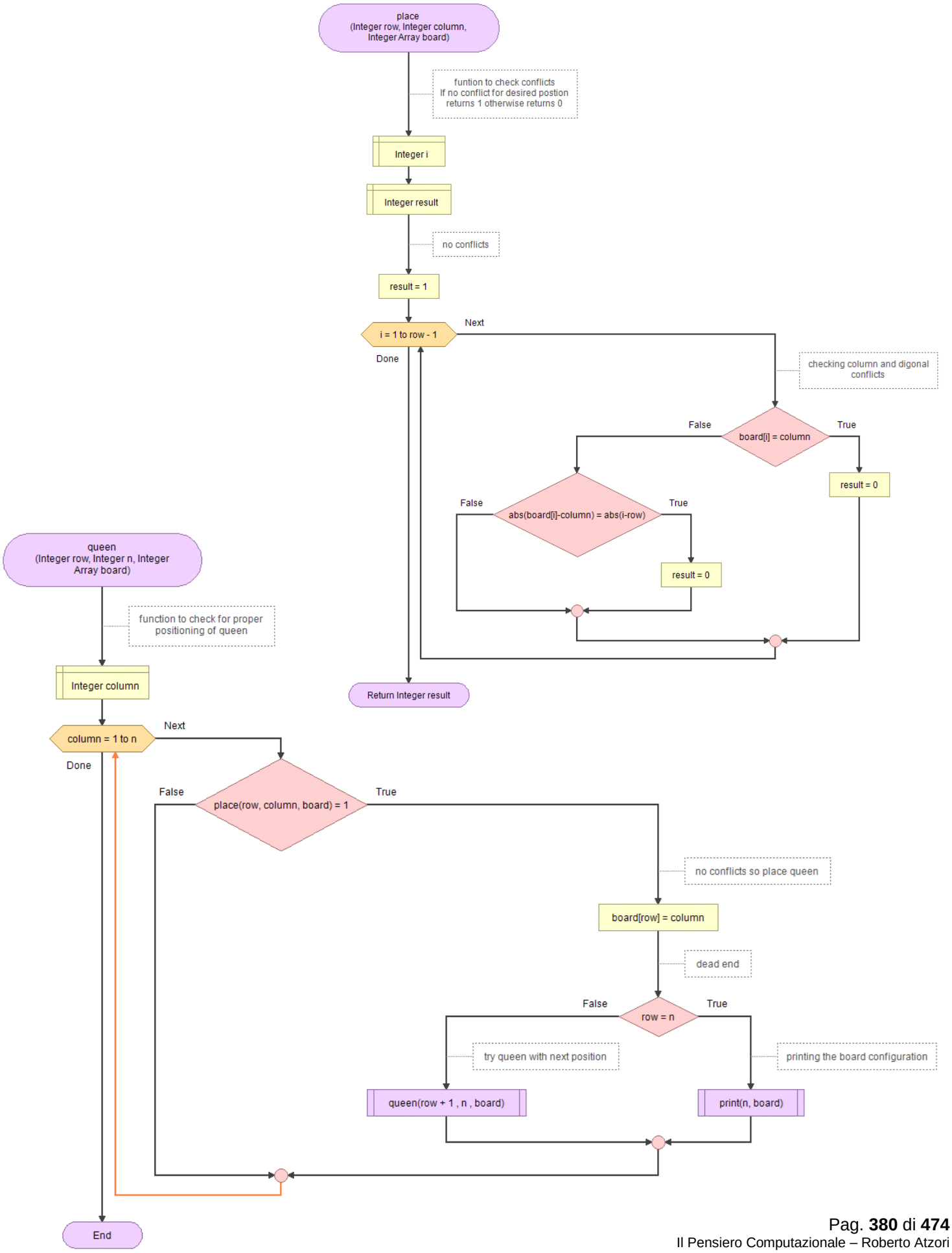

IntBitwise (Integer x, Integer y, String Operator)
String xs, ys, xtemp, lastxs, lastys, ResultString, Remind
Integer MaxLength, i, gap, result
Boolean BoolXs, BoolYs, BoolResult, BoolRemind
xs = DecToBin(x)
ys = DecToBin(y)
len(xs) > len (ys)
False
True
len(xs) < len (ys)
False
True
just one
MaxLength = len(ys)
MaxLength = len(xs)
fill xs with "0"
gap = len(ys) - len(xs)
xtemp = ""
i = 1 to gap
Next
Done
xtemp = xtemp & "0"
xs = xtemp & xs
MaxLength = len(xs)
fill ys with "0"
gap = len(xs) - len(ys)
xtemp = ""
i = 1 to gap
Next
Done
xtemp = xtemp & "0"
ys = xtemp & ys
ResultString = ""
Remind = ""
BoolRemind = False
i = 1 to MaxLength
Next
Done
lastxs = MID2(xs, MaxLength - i + 1, 1)
lastys = MID2(ys, MaxLength - i + 1, 1)
lastxs = "1"
BoolXs = False
BoolXs = True
lastys = "1"
BoolYs = False
BoolYs = True
Operator = "AND"
Operator = "OR"
Operator = "XOR"
BoolResult = BoolXs AND BoolYs
BoolResult = True
ResultString = "0" & ResultString
ResultString = "1" & ResultString
BoolResult = BoolXs OR BoolYs
BoolResult = True
ResultString = "0" & ResultString
ResultString = "1" & ResultString
(BoolXs = True AND BoolYs = True) OR (BoolXs = False AND BoolYs = False)
ResultString = "1" & ResultString
ResultString = "0" & ResultString
overflow
result = BinToDec(ToInteger (ResultString))
Return Integer result

266.GSAFR ‖ Problema delle 8 regine (con back tracking)

8 queens problem (back tracking)

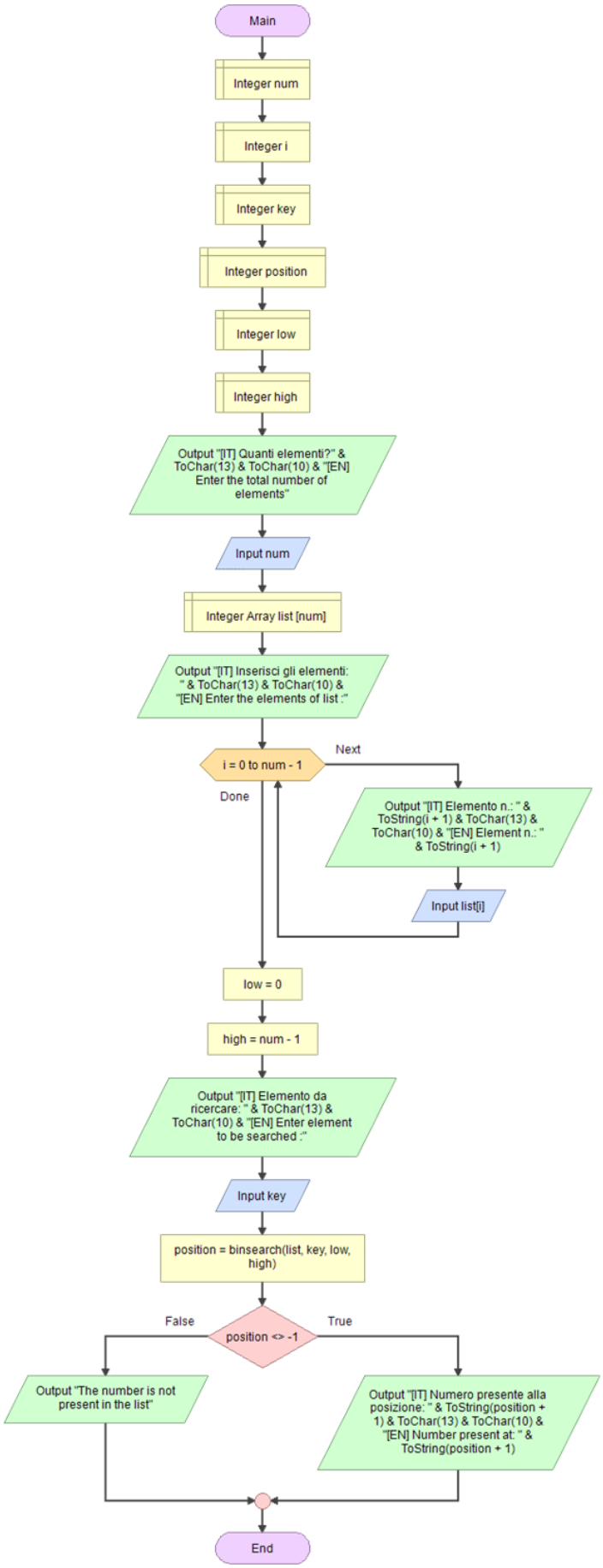


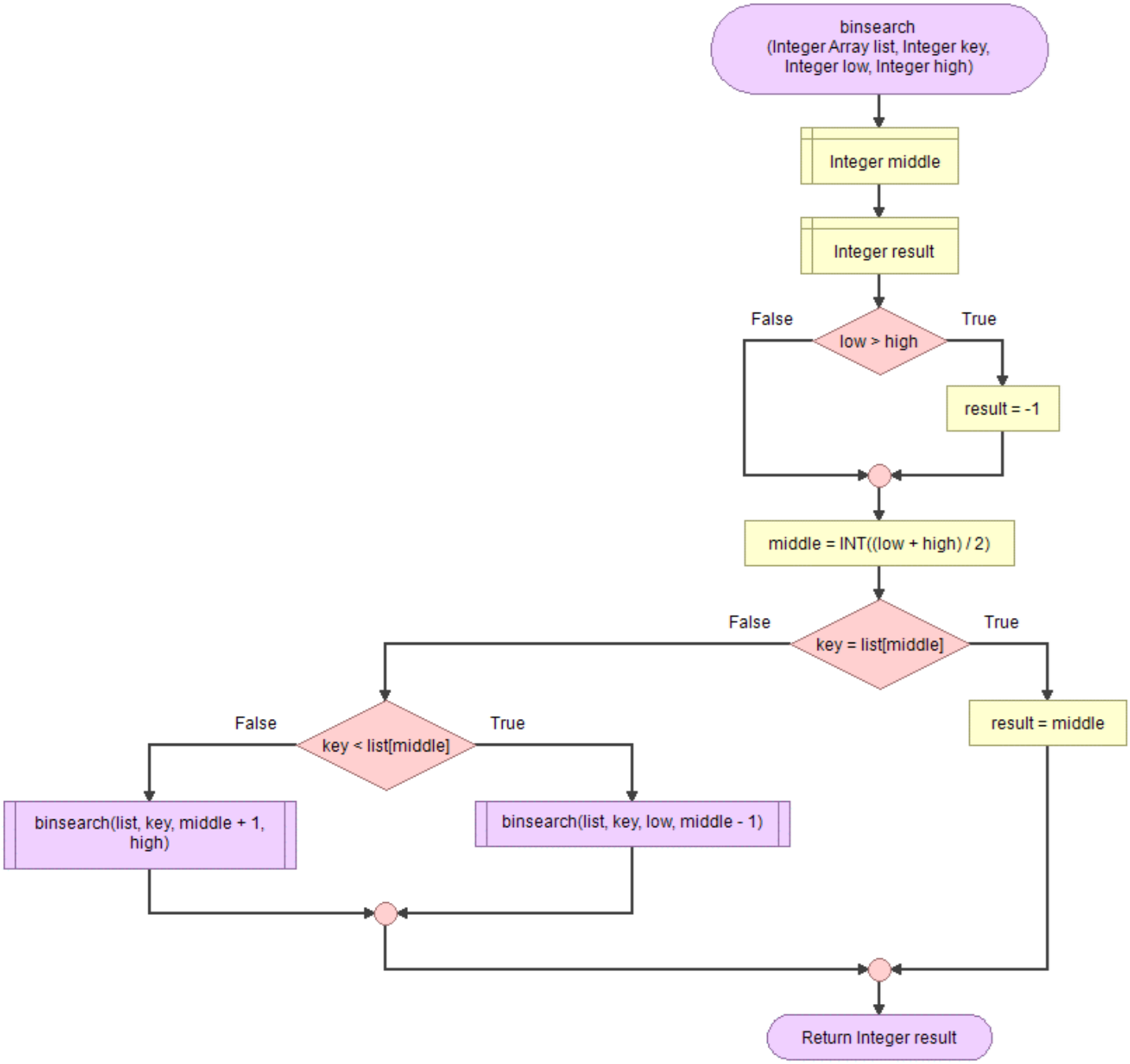



place
(Integer row, Integer column, Integer Array board)
funtion to check conflicts
If no conflict for desired postion returns 1 otherwise returns 0
Integer i
Integer result
no conflicts
result = 1
i = 1 to row - 1
Next
Done
checking column and digonal conflicts
board[i] = column
False
True
result = 0
abs(board[i]-column) = abs(i-row)
False
True
result = 0
Return Integer result
queen
(Integer row, Integer n, Integer Array board)
function to check for proper positioning of queen
Integer column
column = 1 to n
Next
Done
place(row, column, board) = 1
False
True
no conflicts so place queen
board[row] = column
dead end
row = n
False
True
try queen with next position
printing the board configuration
queen(row + 1 , n , board)
print(n, board)
End

267.GAFR | Ricerca binaria ricorsiva su array

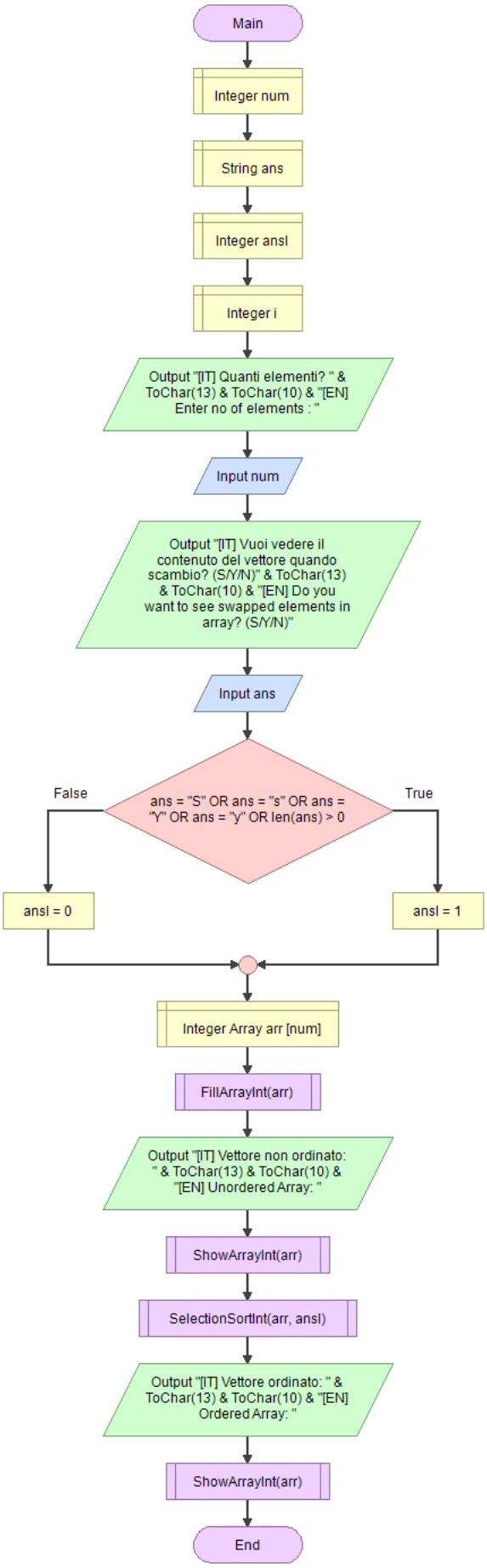
Recursive Binary Search

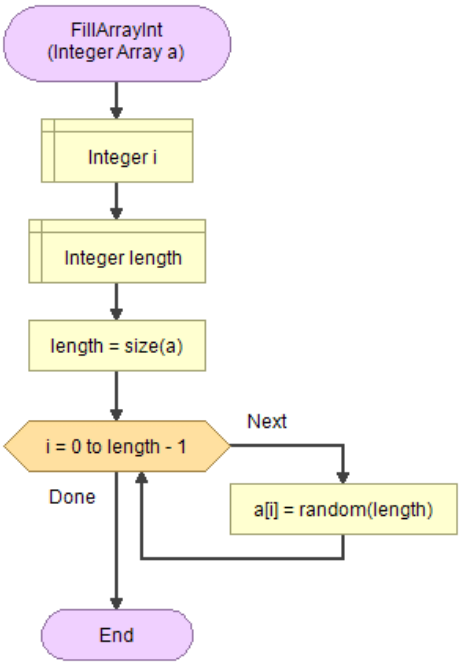


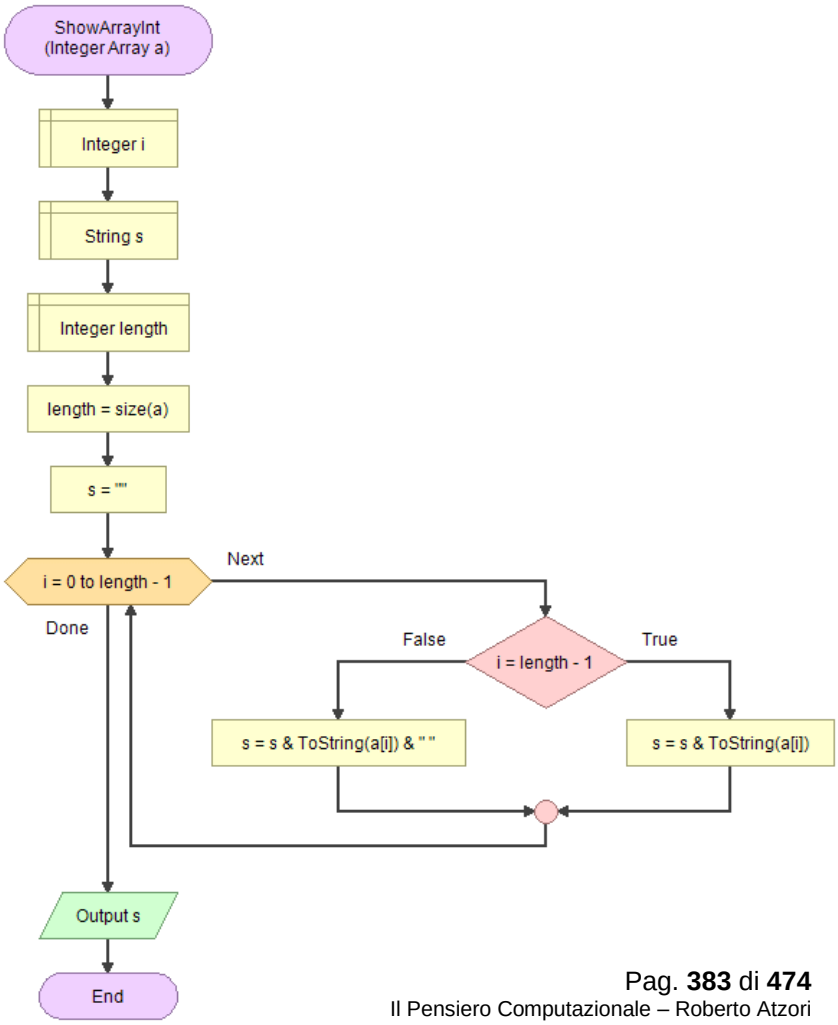

binsearch
(Integer Array list, Integer key,
Integer low, Integer high)
Integer middle
Integer result
False
low > high
True
result = -1
middle = INT((low + high) / 2)
False
key = list[middle]
True
result = middle
False
key < list[middle]
True
binsearch(list, key, middle + 1,
high)
binsearch(list, key, low, middle - 1)
Return Integer result

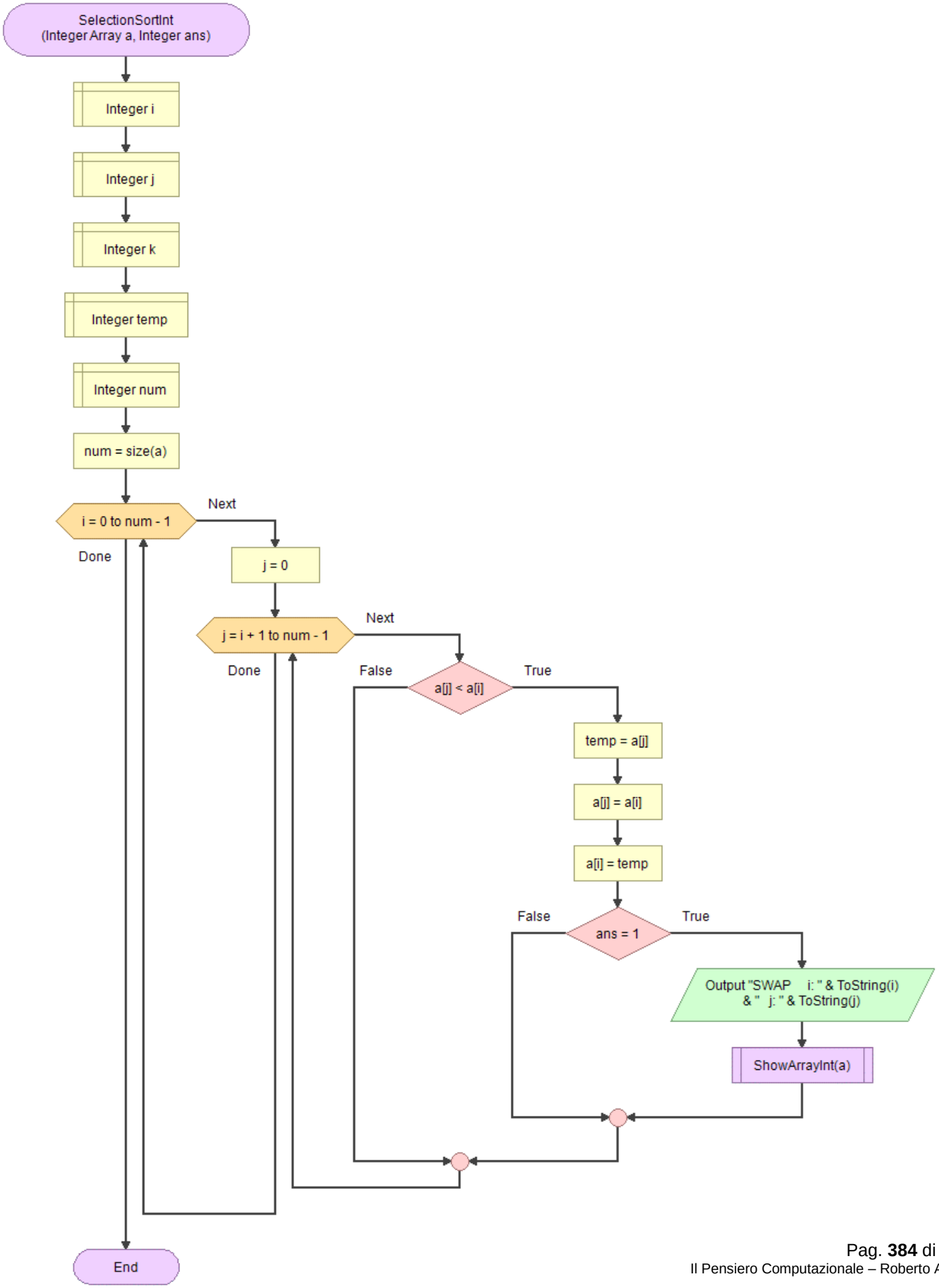
268.GAF ‖ Ordinamento per selezione

Selection Sort



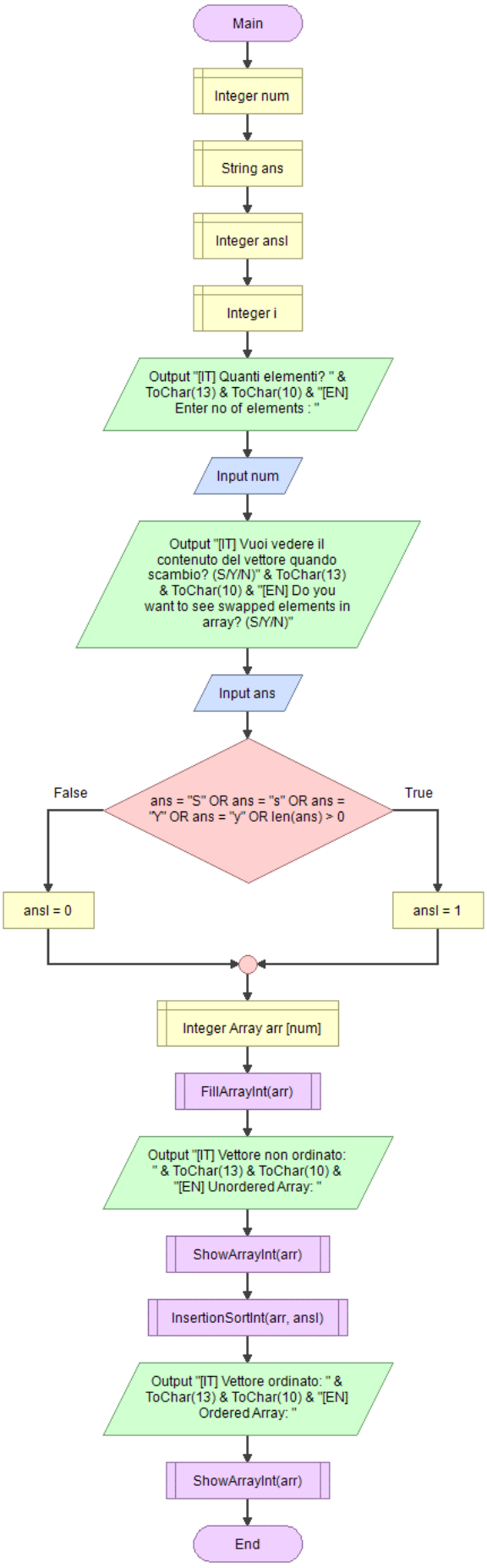


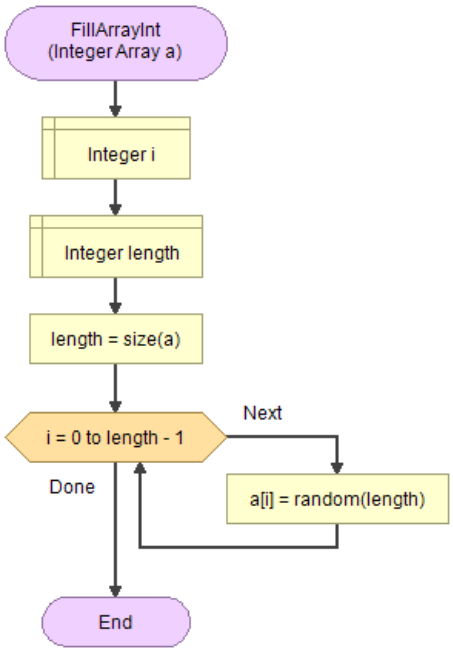


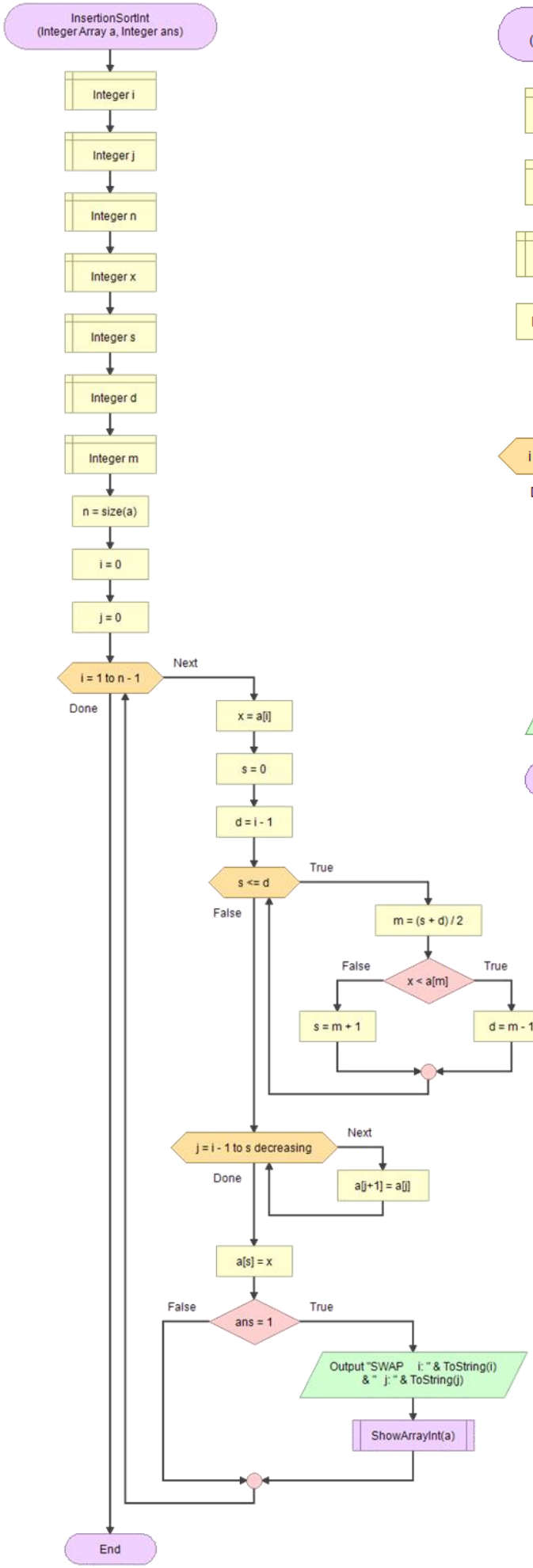

SelectionSortInt
(Integer Array a, Integer ans)
Integer i
Integer j
Integer k
Integer temp
Integer num
num = size(a)
i = 0 to num - 1
Next
Done
j = 0
j = i + 1 to num - 1
Next
Done
False
a[j] < a[i]
True
temp = a[j]
a[j] = a[i]
a[i] = temp
False
ans = 1
True
Output "SWAP i: " & ToString(i)
& " j: " & ToString(j)
ShowArrayInt(a)
End

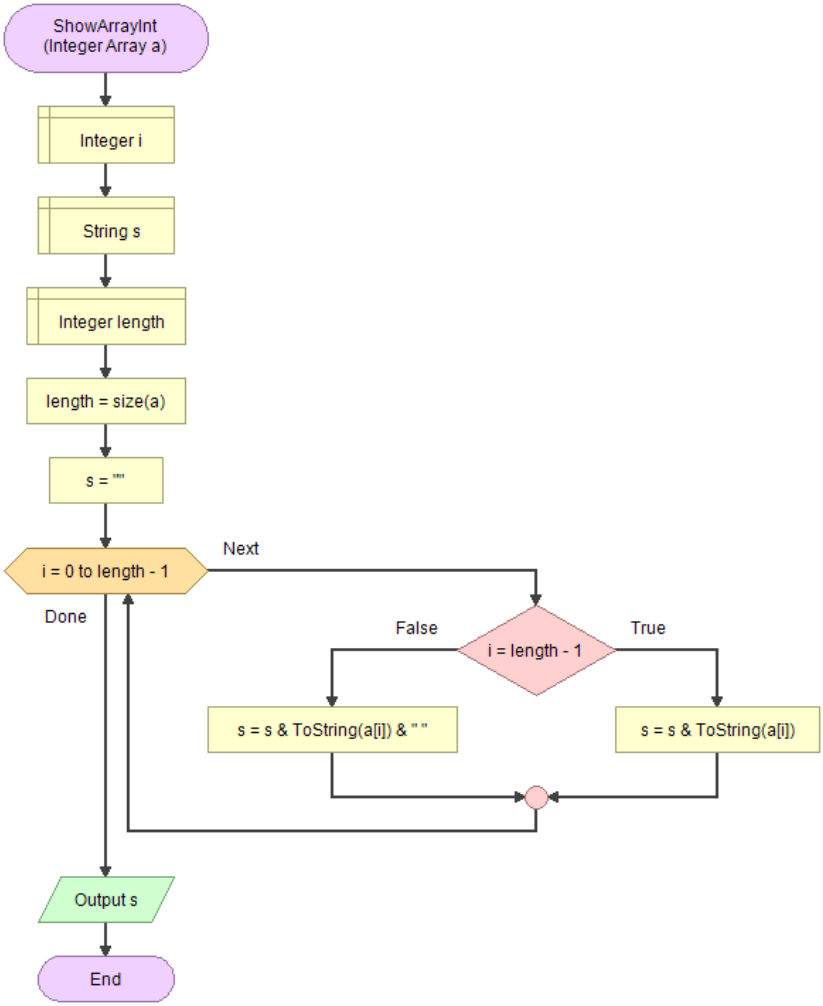
269.GAF ‖ Ordinamento per inserzione

Insertion Sort



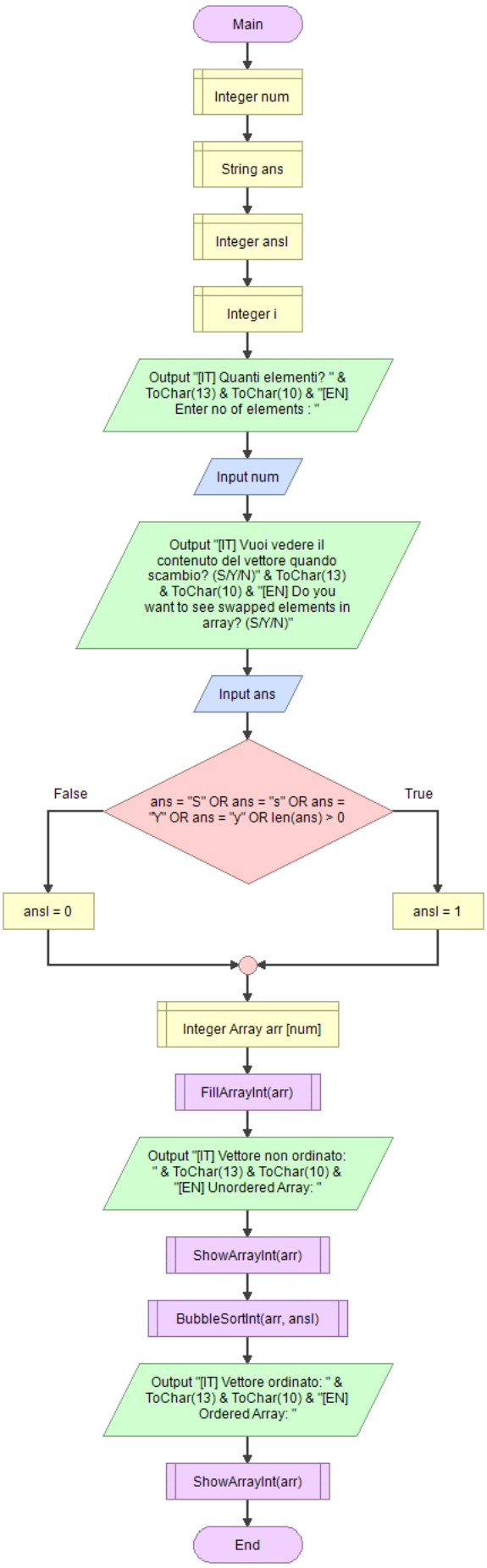


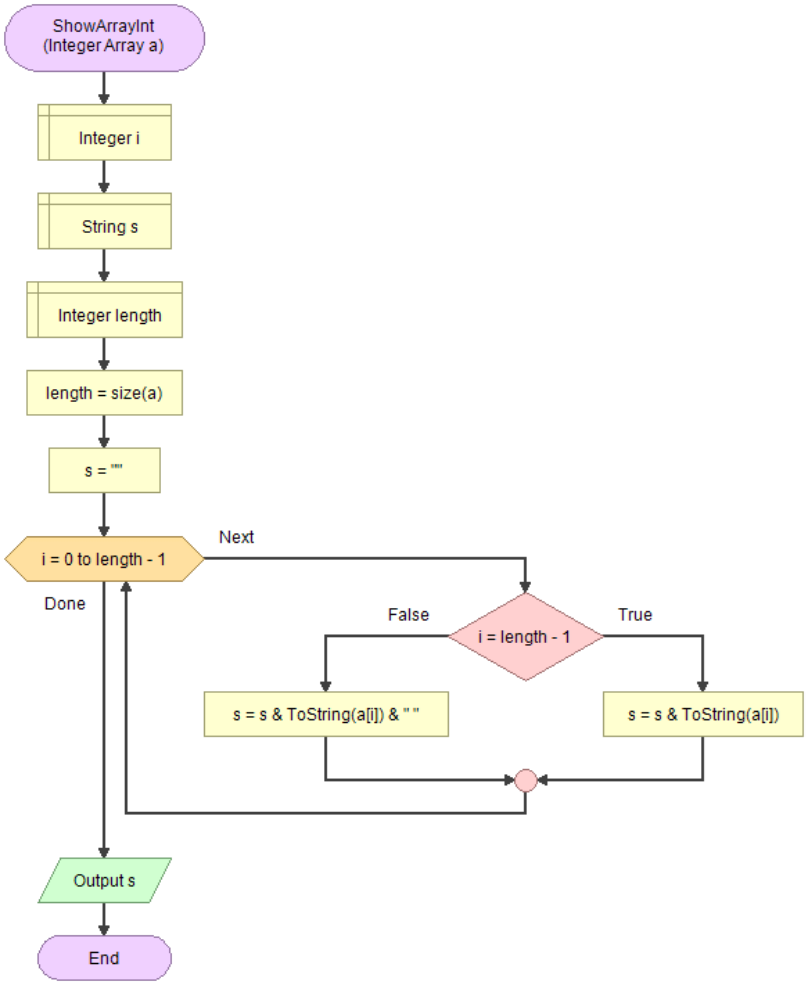

InsertionSortInt
(Integer Array a, Integer ans)
Integer i
Integer j
Integer n
Integer x
Integer s
Integer d
Integer m
n = size(a)
i = 0
j = 0
i = 1 to n - 1
Next
Done
x = a[i]
s = 0
d = i - 1
s <= d
True
False
m = (s + d) / 2
x < a[m]
False
True
s = m + 1
d = m - 1
j = i - 1 to s decreasing
Next
Done
a[j+1] = a[j]
a[s] = x
ans = 1
False
True
Output "SWAP i: " & ToString(i) & " j: " & ToString(j)
ShowArrayInt(a)
End

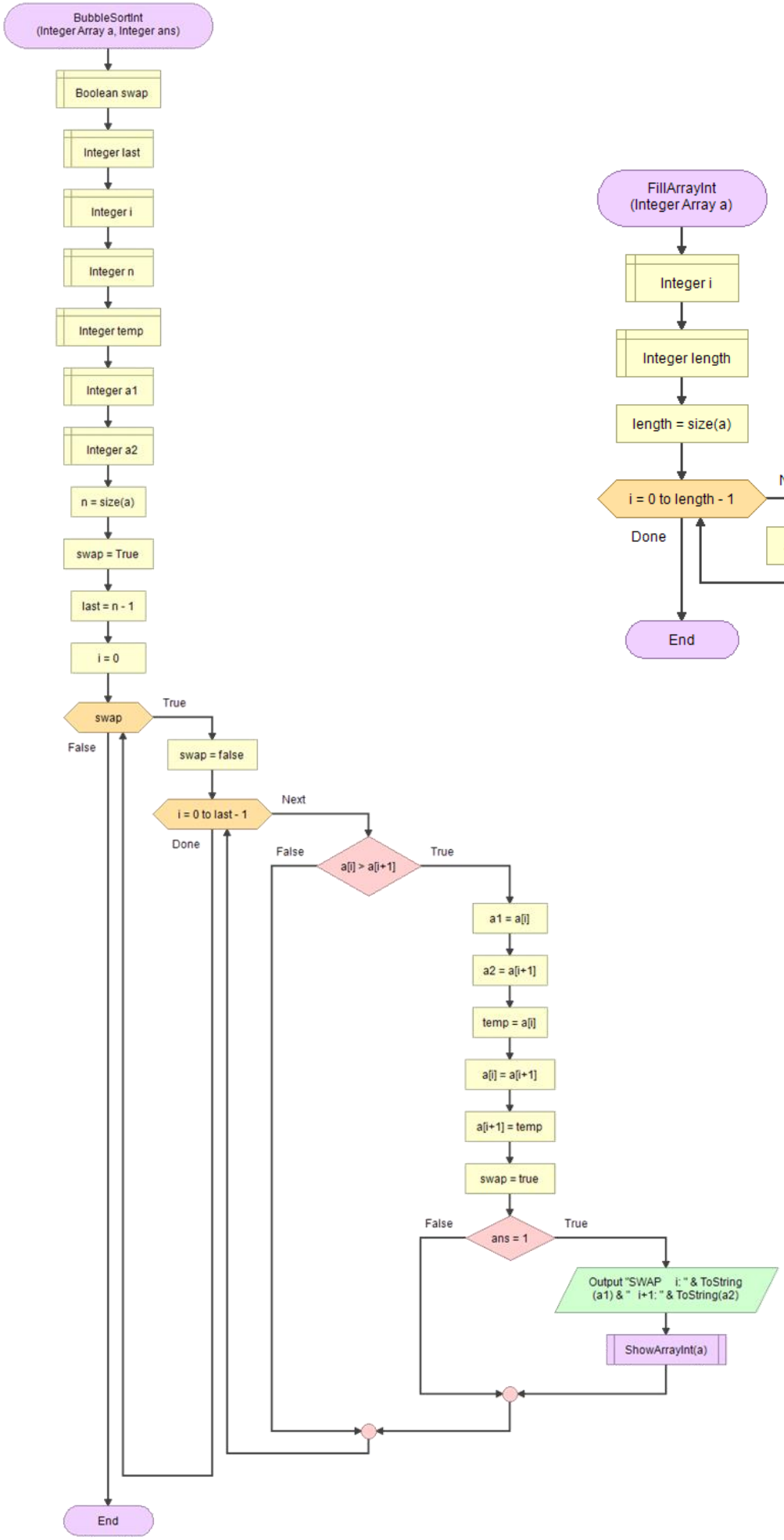

ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
False
i = length - 1
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
End

270.GAF ‖ Ordinamento a bolle

Bubble Sort

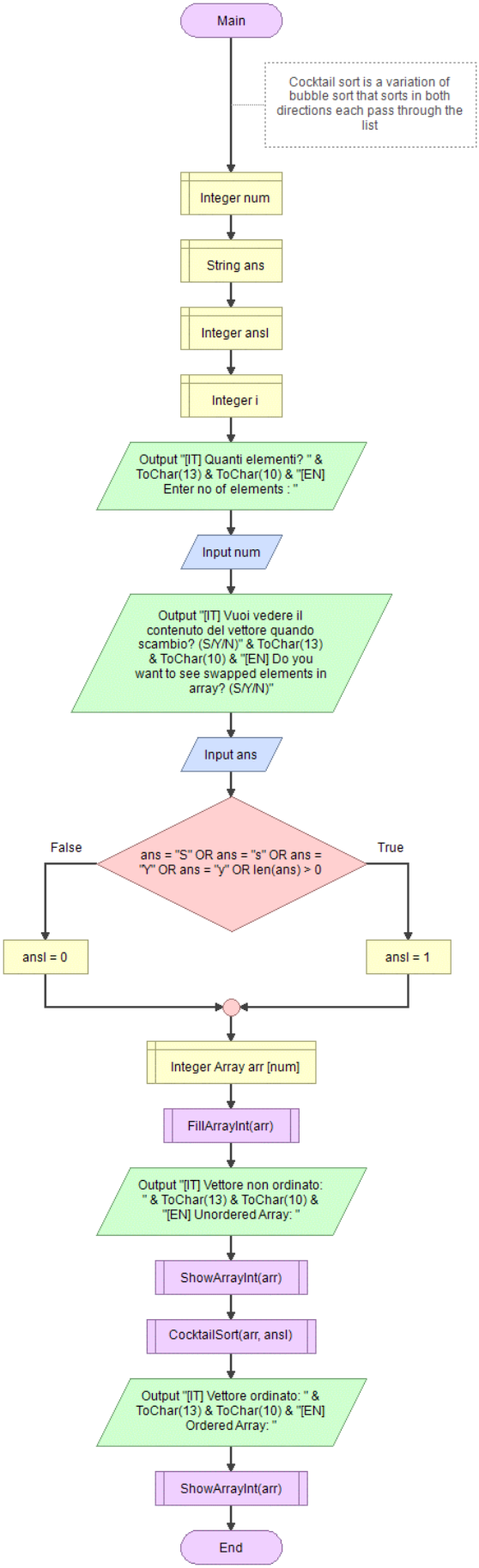


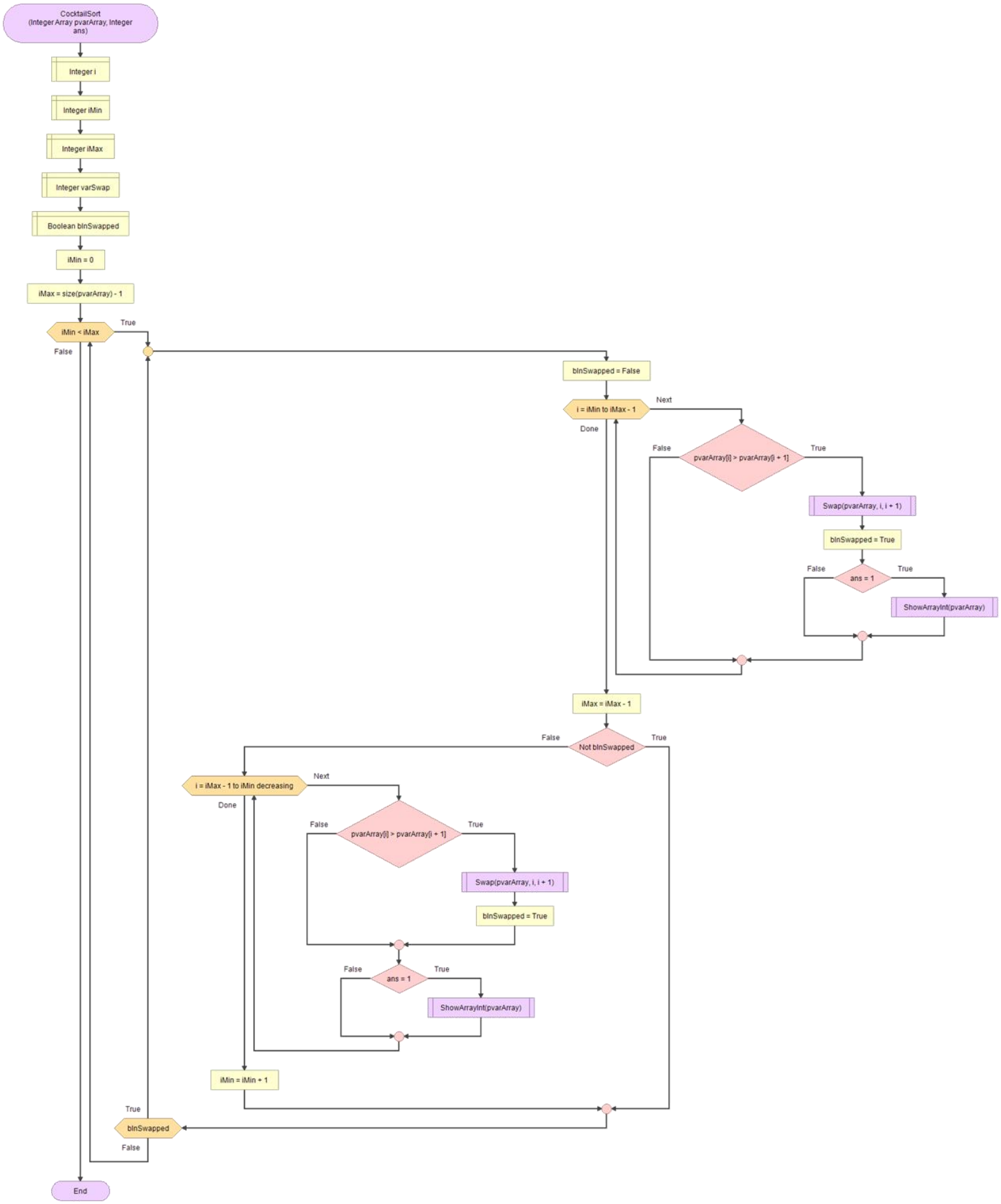


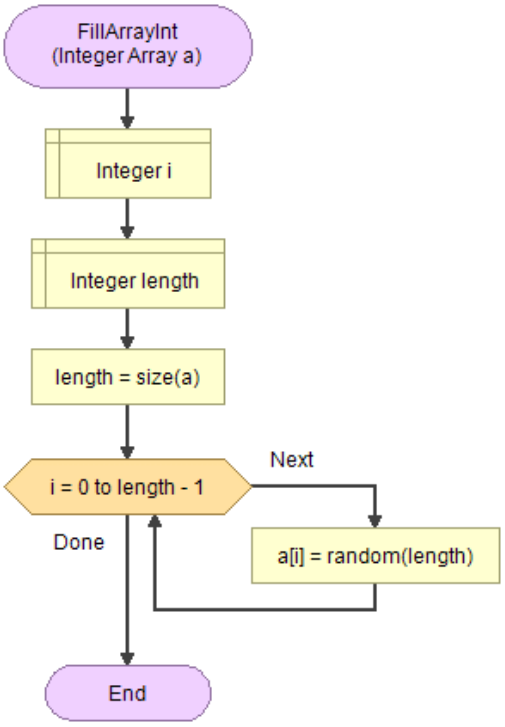

BubbleSortInt
(Integer Array a, Integer ans)
Boolean swap
Integer last
Integer i
Integer n
Integer temp
Integer a1
Integer a2
n = size(a)
swap = True
last = n - 1
i = 0
swap
True
False
swap = false
i = 0 to last - 1
Next
Done
a[i] > a[i+1]
False
True
a1 = a[i]
a2 = a[i+1]
temp = a[i]
a[i] = a[i+1]
a[i+1] = temp
swap = true
ans = 1
False
True
Output "SWAP i: " & ToString
(a1) & " i+1: " & ToString(a2)
ShowArrayInt(a)
End
FillArrayInt
(Integer Array a)
Integer i
Integer length
length = size(a)
i = 0 to length - 1
Next
Done
a[i] = random(length)
End

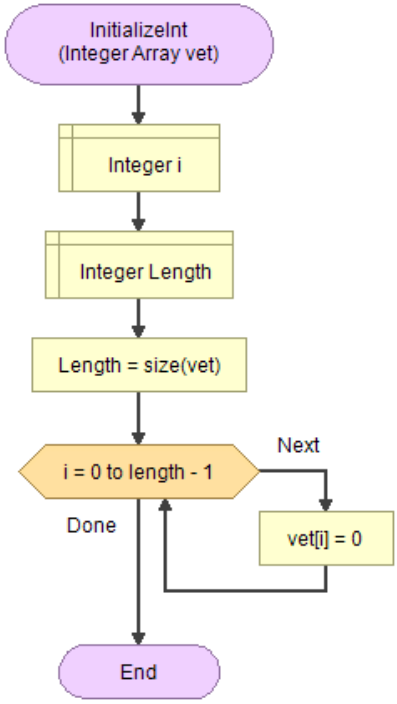
271.GAF ‖ Ordinamento Cocktail

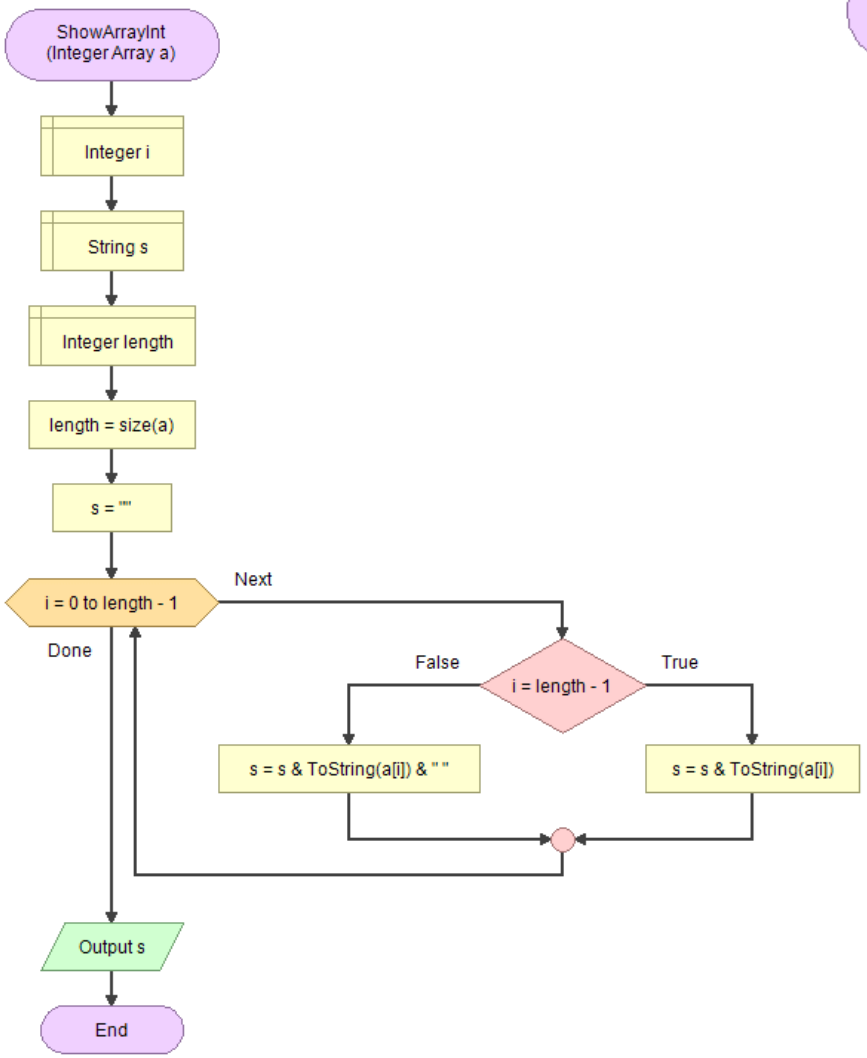
Cocktail Sort

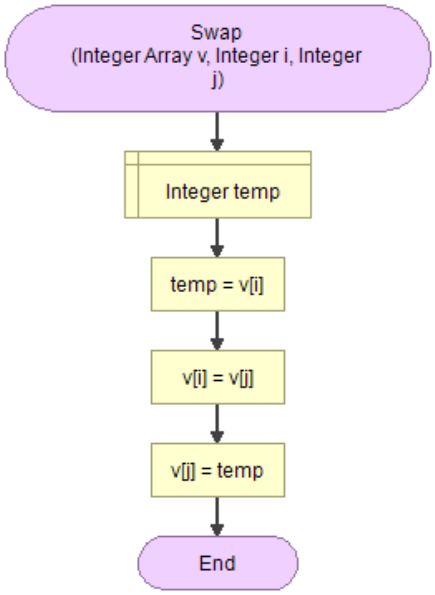



CocktailSort
(Integer Array pvarArray, Integer ans)
Integer i
Integer iMin
Integer iMax
Integer varSwap
Boolean blnSwapped
iMin = 0
iMax = size(pvarArray) - 1
iMin < iMax
True
False
blnSwapped = False
i = iMin to iMax - 1
Next
Done
pvarArray[i] > pvarArray[i + 1]
False
True
Swap(pvarArray, i, i + 1)
blnSwapped = True
ans = 1
False
True
ShowArrayInt(pvarArray)
iMax = iMax - 1
Not blnSwapped
False
True
i = iMax - 1 to iMin decreasing
Next
Done
pvarArray[i] > pvarArray[i + 1]
False
True
Swap(pvarArray, i, i + 1)
blnSwapped = True
ans = 1
False
True
ShowArrayInt(pvarArray)
iMin = iMin + 1
blnSwapped
True
False
End


FillArrayInt
(Integer Array a)
Integer i
Integer length
length = size(a)
i = 0 to length - 1
Next
Done
a[i] = random(length)
End

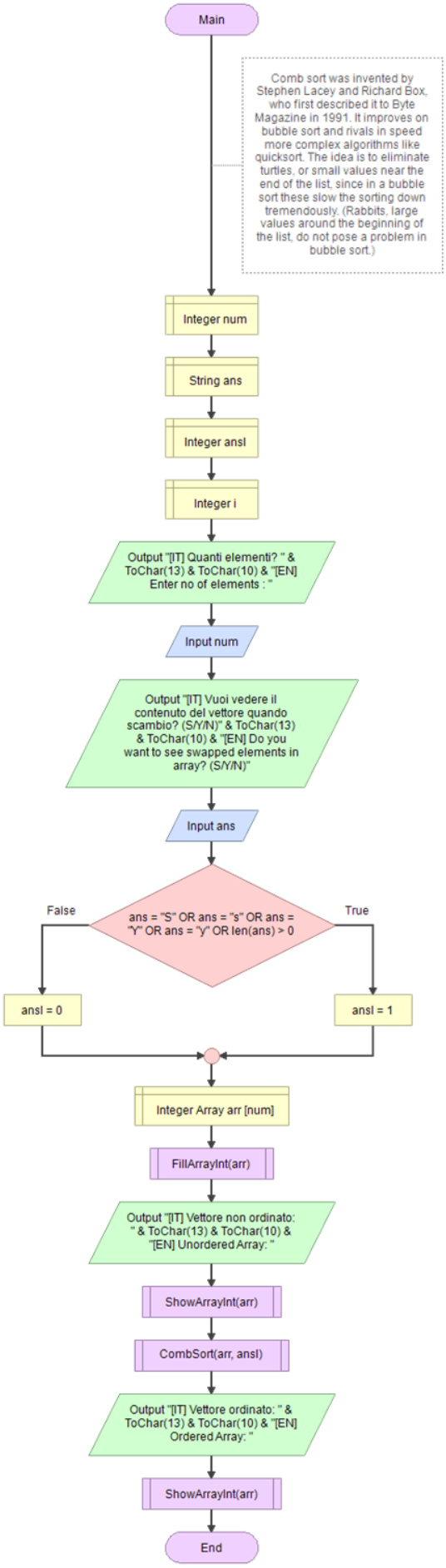

InitializeInt
(Integer Array vet)
Integer i
Integer Length
Length = size(vet)
i = 0 to length - 1
Next
Done
vet[i] = 0
End

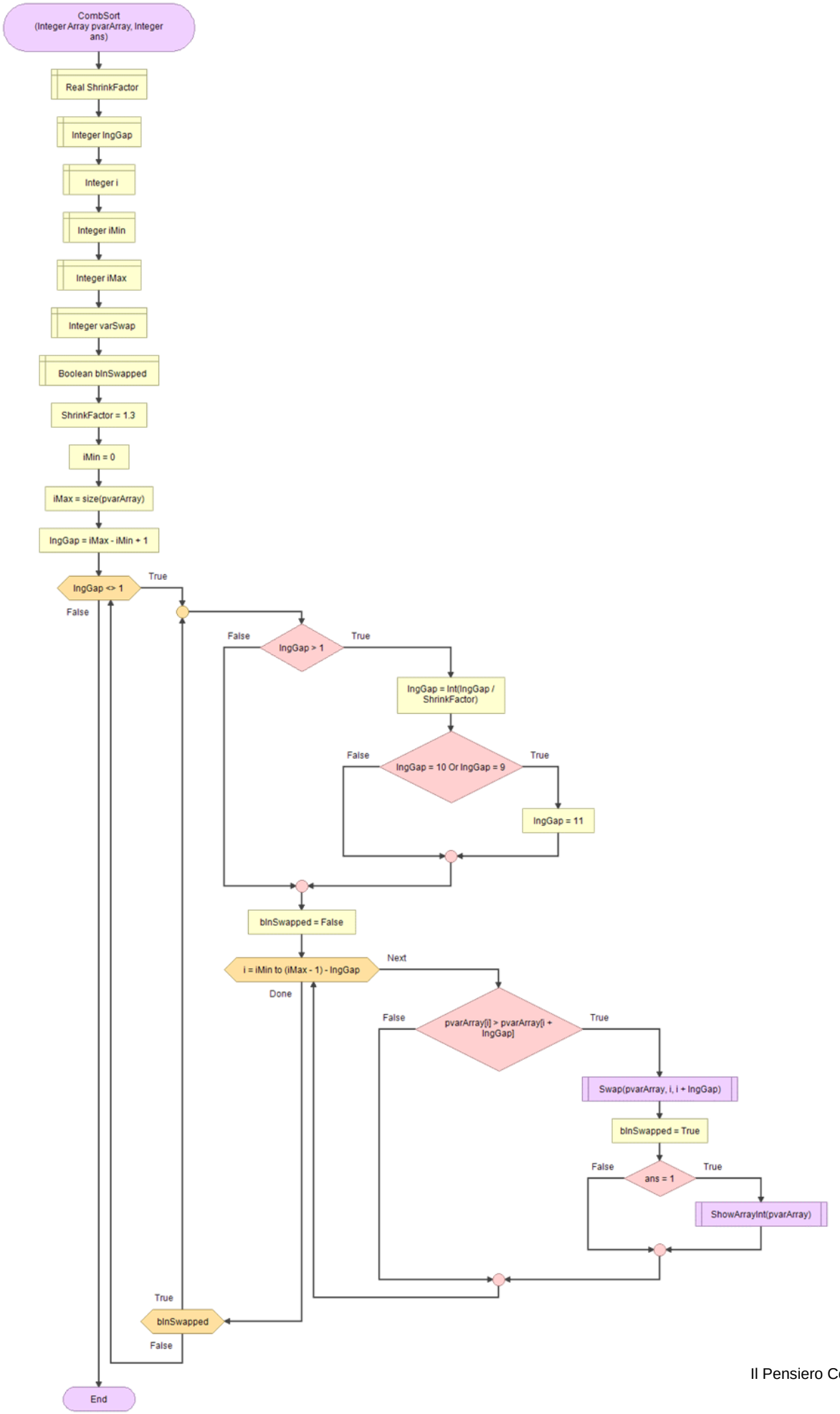

ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
False
i = length - 1
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
End

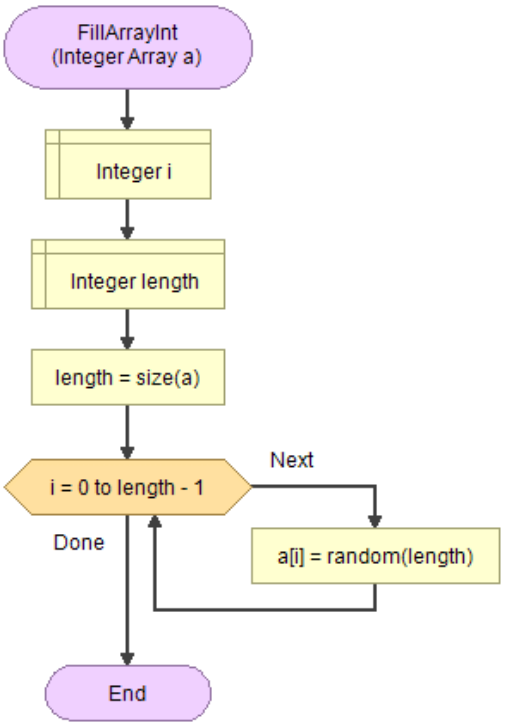

Swap
(Integer Array v, Integer i, Integer j)
Integer temp
temp = v[i]
v[i] = v[j]
v[j] = temp
End

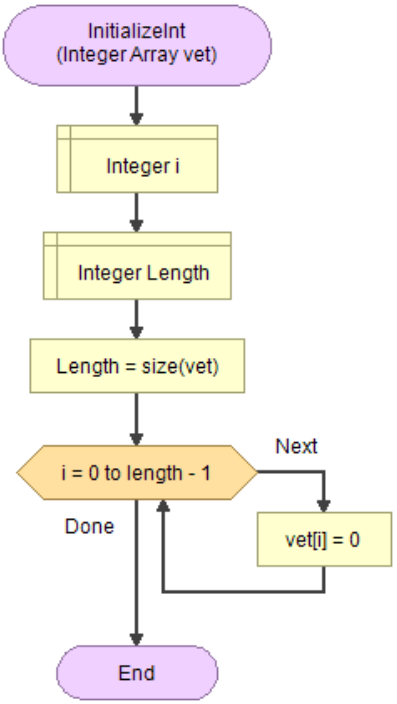
272.GAF ‖ Ordinamento Comb

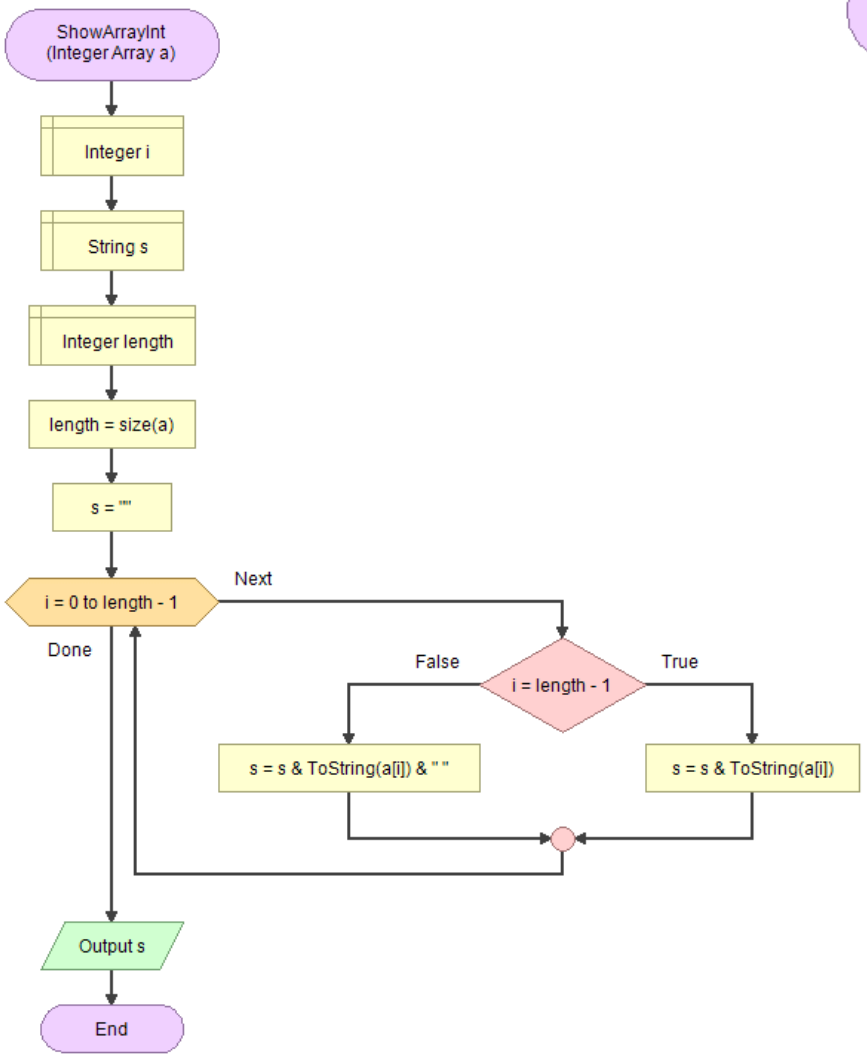
Comb Sort

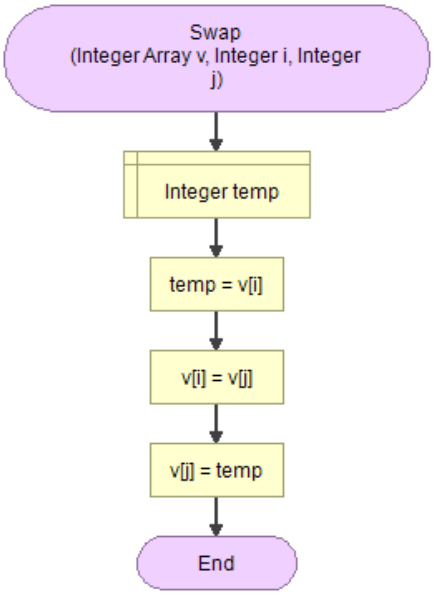



CombSort
(Integer Array pvarArray, Integer ans)
Real ShrinkFactor
Integer lngGap
Integer i
Integer iMin
Integer iMax
Integer varSwap
Boolean blnSwapped
ShrinkFactor = 1.3
iMin = 0
iMax = size(pvarArray)
lngGap = iMax - iMin + 1
lngGap <> 1
True
False
lngGap > 1
False
True
lngGap = Int(lngGap / ShrinkFactor)
lngGap = 10 Or lngGap = 9
False
True
lngGap = 11
blnSwapped = False
i = iMin to (iMax - 1) - lngGap
Next
Done
pvarArray[i] > pvarArray[i + lngGap]
False
True
Swap(pvarArray, i, i + lngGap)
blnSwapped = True
ans = 1
False
True
ShowArrayInt(pvarArray)
blnSwapped
True
False
End


FillArrayInt
(Integer Array a)
Integer i
Integer length
length = size(a)
i = 0 to length - 1
Next
a[i] = random(length)
Done
End

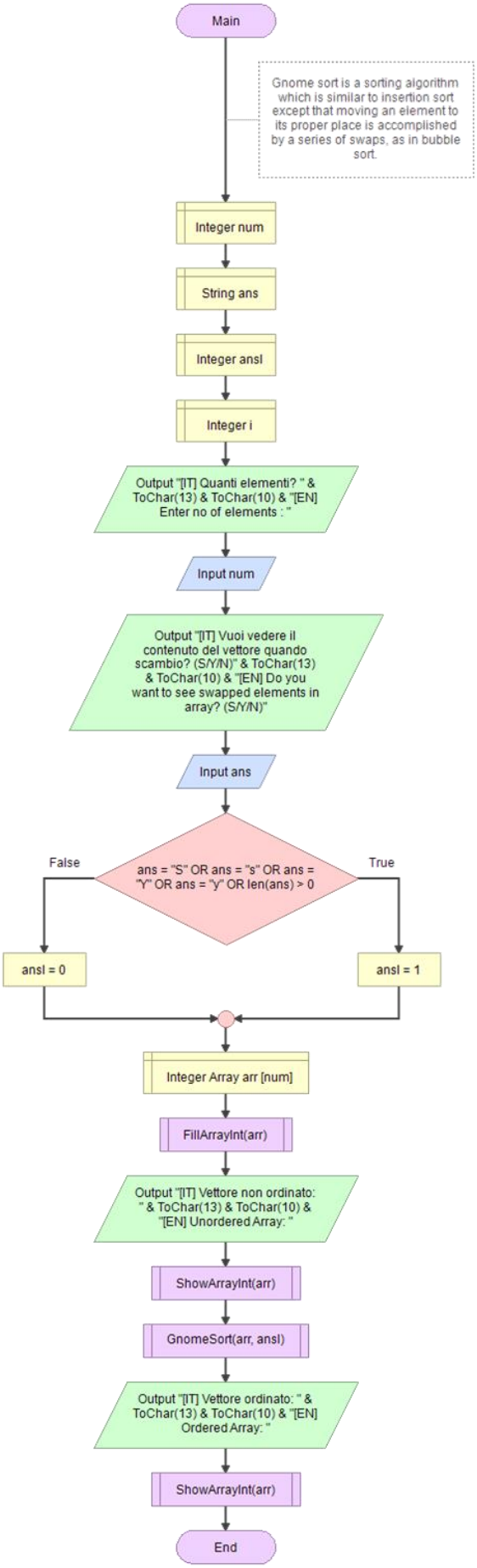

InitializeInt
(Integer Array vet)
Integer i
Integer Length
Length = size(vet)
i = 0 to length - 1
Next
vet[i] = 0
Done
End

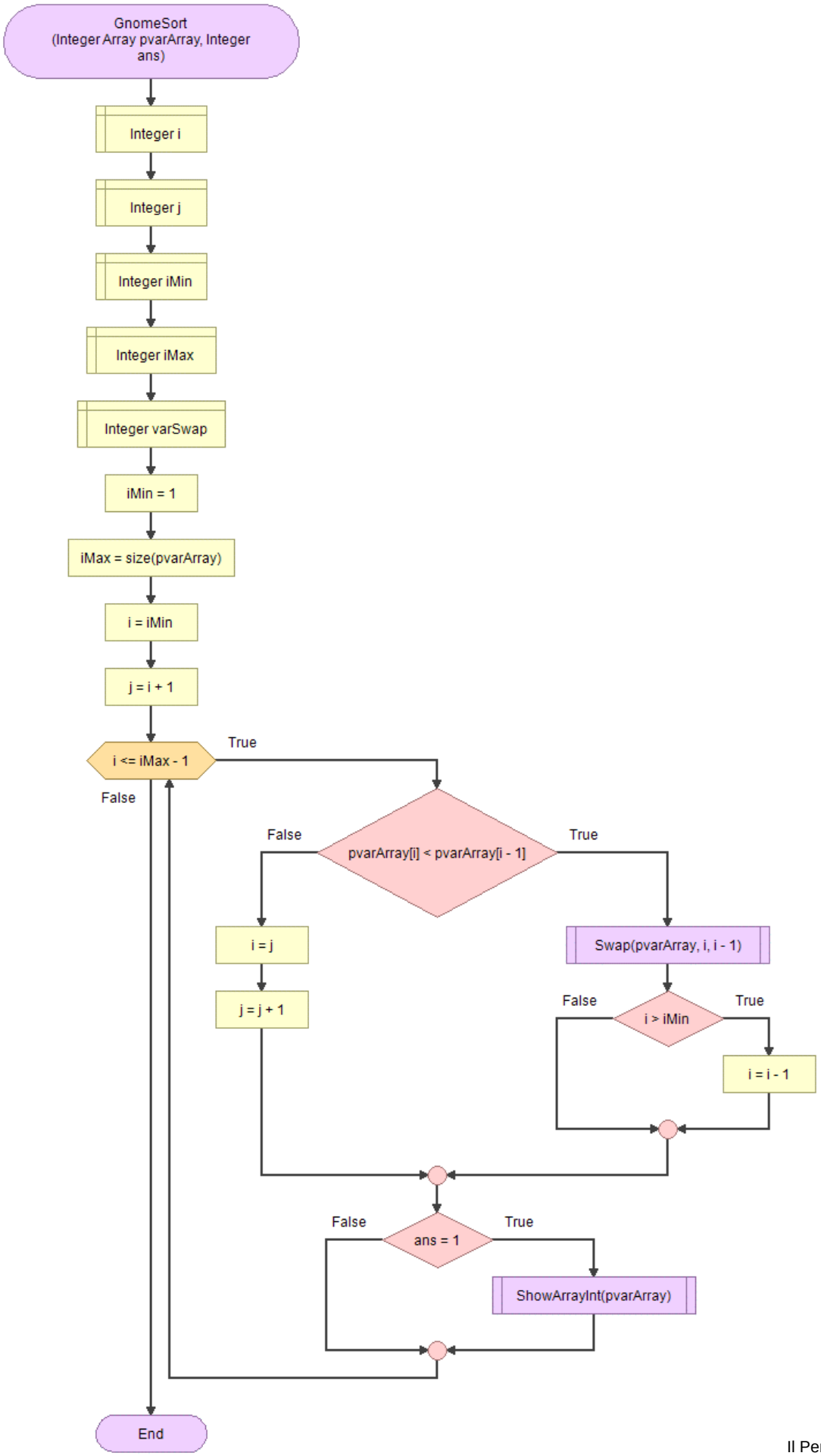

ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
False
i = length - 1
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
End

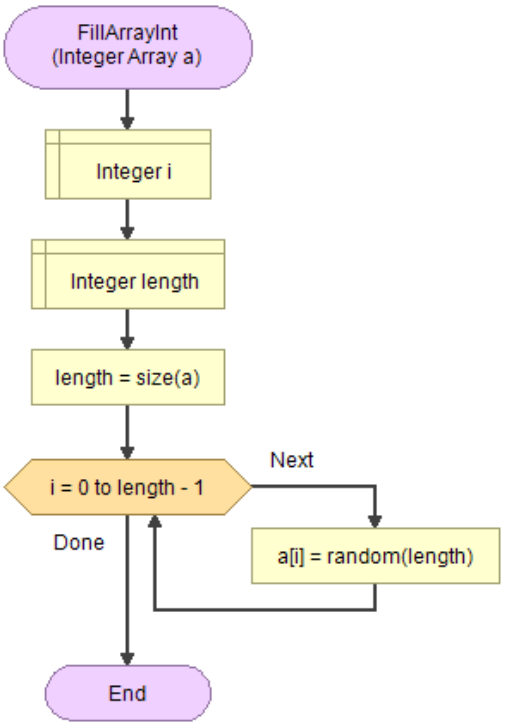

Swap
(Integer Array v, Integer i, Integer j)
Integer temp
temp = v[i]
v[i] = v[j]
v[j] = temp
End

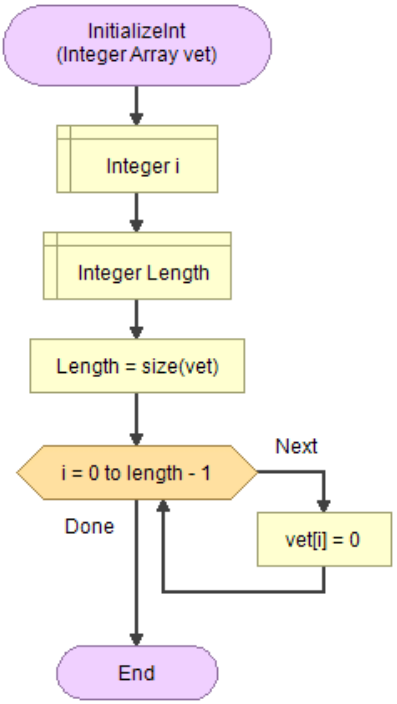
273.GAF ‖ Ordinamento Gnome

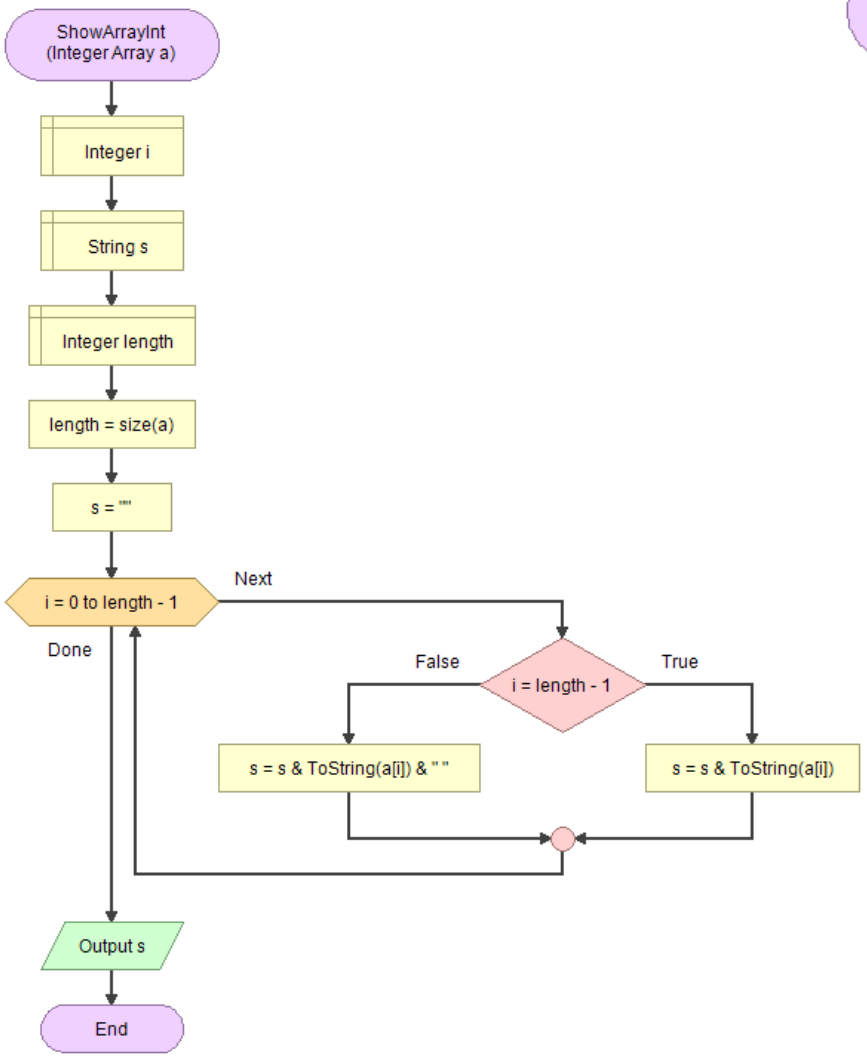
Gnome Sort

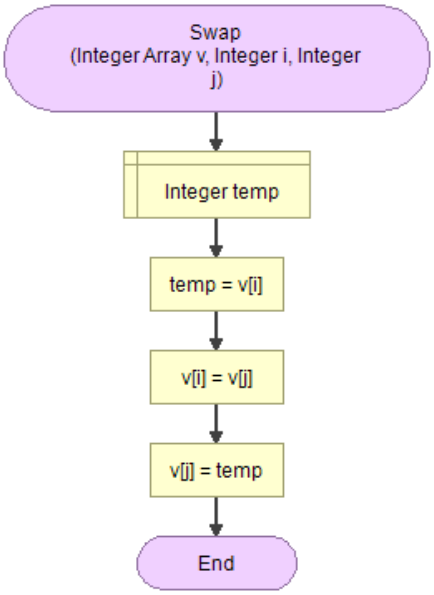



GnomeSort
(Integer Array pvarArray, Integer ans)
Integer i
Integer j
Integer iMin
Integer iMax
Integer varSwap
iMin = 1
iMax = size(pvarArray)
i = iMin
j = i + 1
i <= iMax - 1
True
False
pvarArray[i] < pvarArray[i - 1]
False
True
i = j
j = j + 1
Swap(pvarArray, i, i - 1)
i > iMin
False
True
i = i - 1
ans = 1
False
True
ShowArrayInt(pvarArray)
End


FillArrayInt
(Integer Array a)
Integer i
Integer length
length = size(a)
i = 0 to length - 1
Next
a[i] = random(length)
Done
End

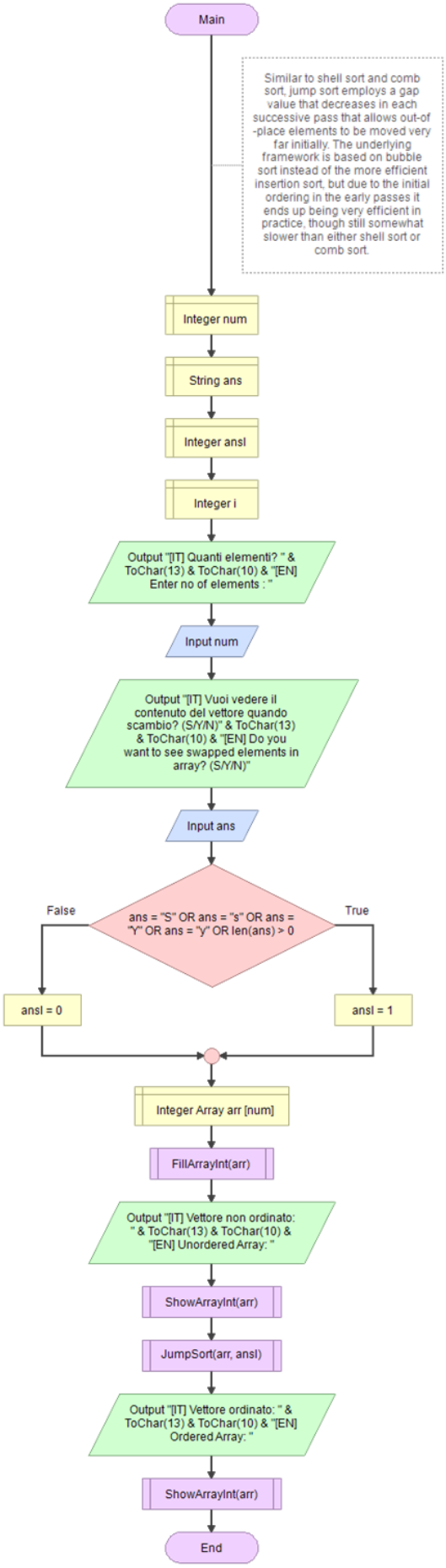

InitializeInt
(Integer Array vet)
Integer i
Integer Length
Length = size(vet)
i = 0 to length - 1
Next
vet[i] = 0
Done
End

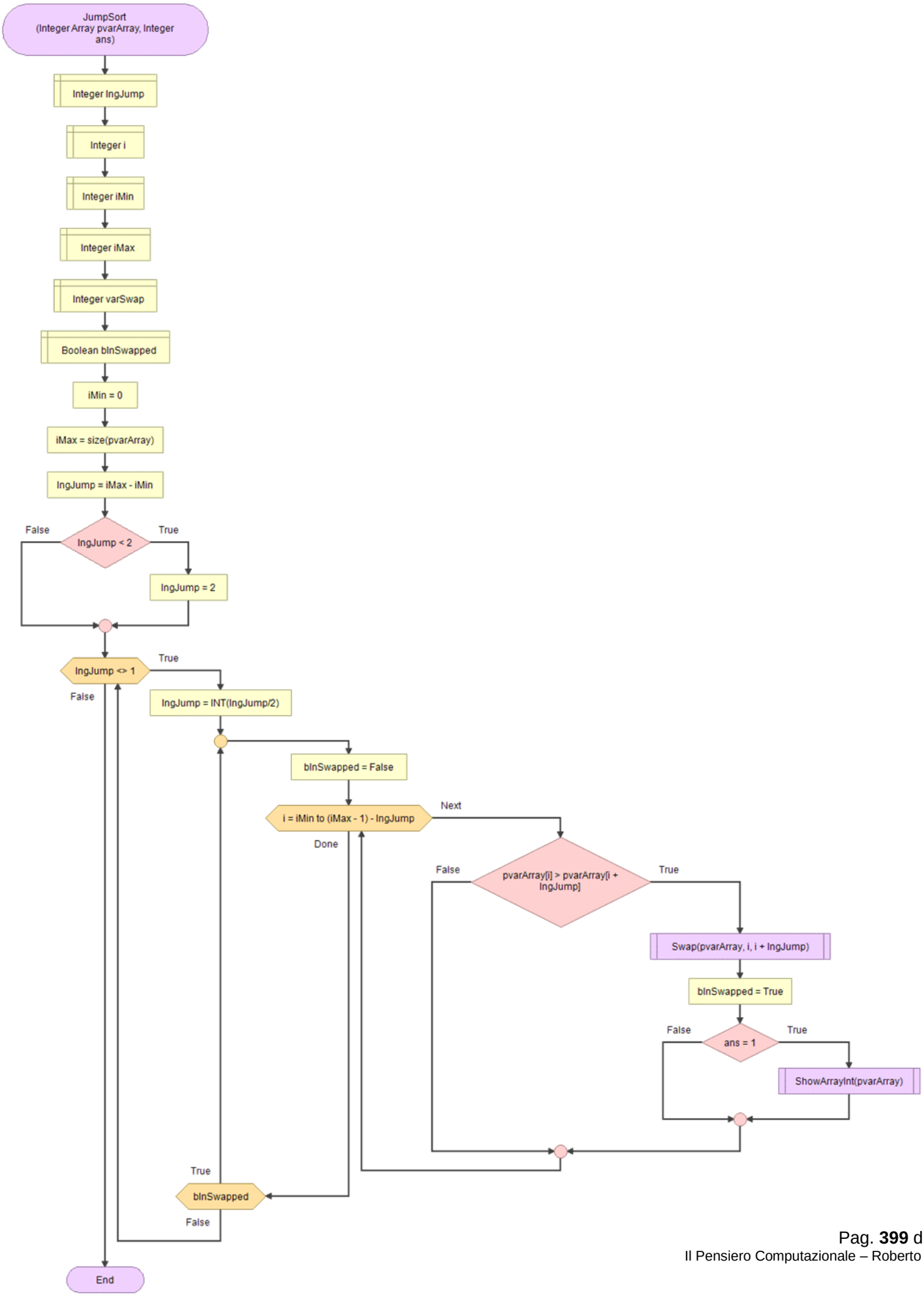

ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
i = length - 1
False
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
End

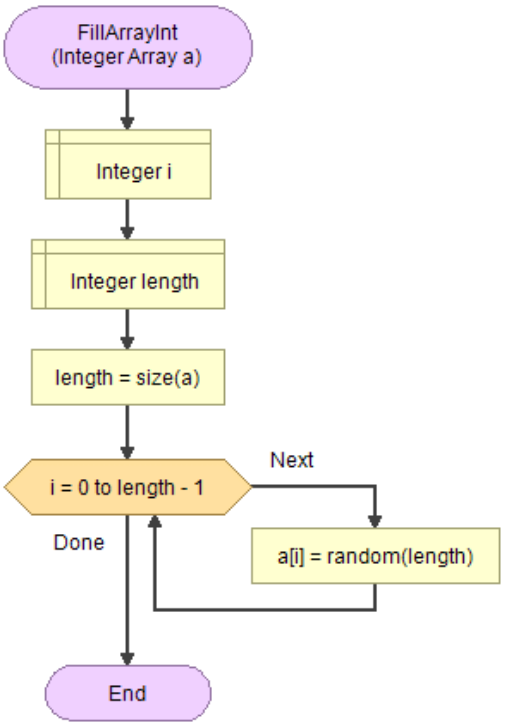

Swap
(Integer Array v, Integer i, Integer j)
Integer temp
temp = v[i]
v[i] = v[j]
v[j] = temp
End

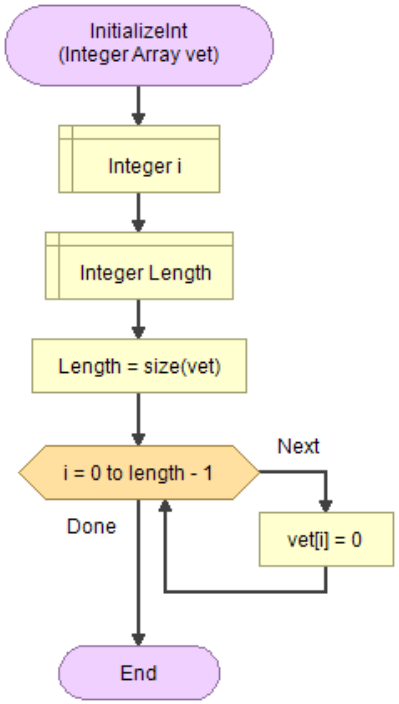
274.GAF ‖ Ordinamento Jump

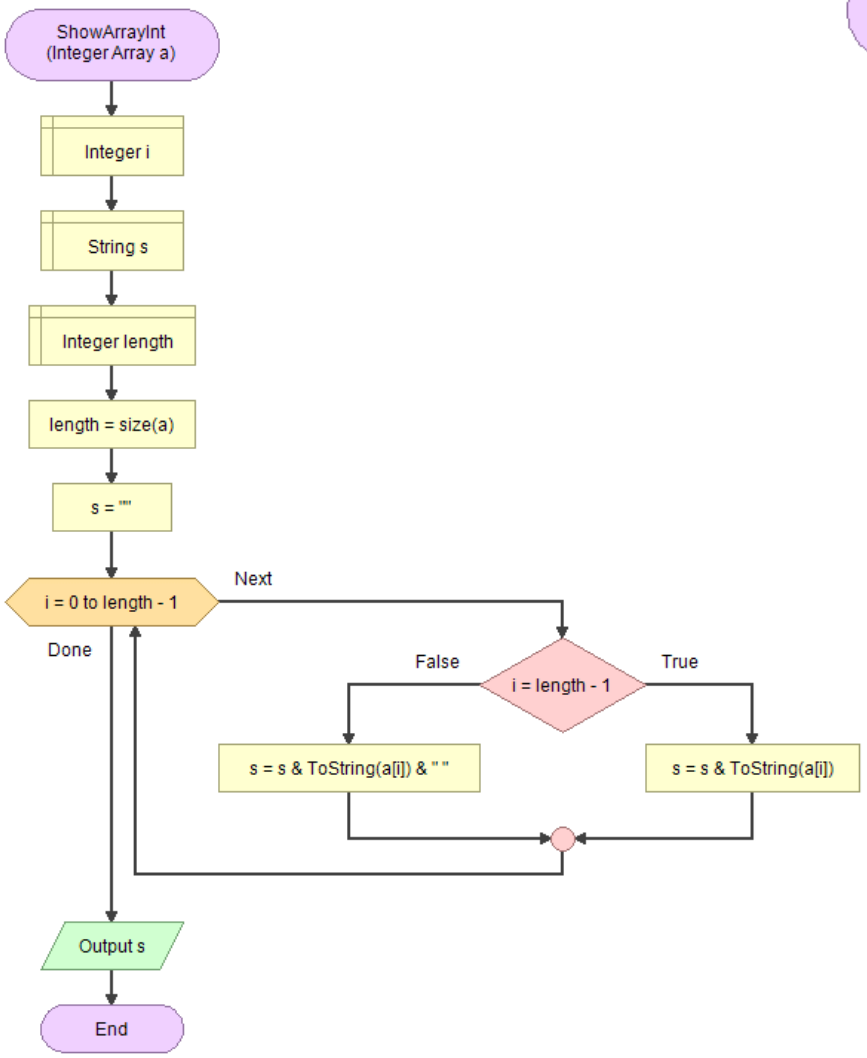
Jump Sort

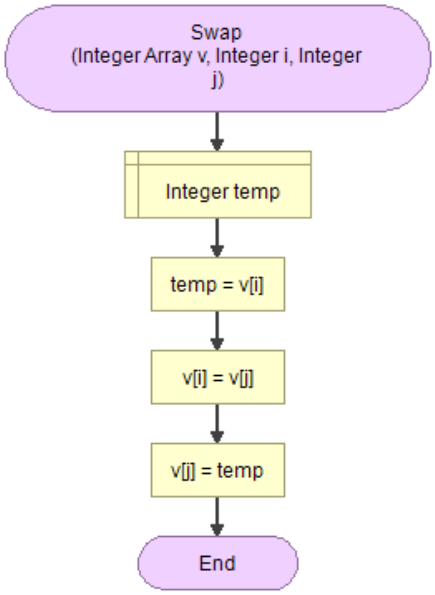



JumpSort
(Integer Array pvarArray, Integer ans)
Integer lngJump
Integer i
Integer iMin
Integer iMax
Integer varSwap
Boolean blnSwapped
iMin = 0
iMax = size(pvarArray)
lngJump = iMax - iMin
False
lngJump < 2
True
lngJump = 2
lngJump <> 1
True
False
lngJump = INT(lngJump/2)
blnSwapped = False
i = iMin to (iMax - 1) - lngJump
Next
Done
False
pvarArray[i] > pvarArray[i + lngJump]
True
Swap(pvarArray, i, i + lngJump)
blnSwapped = True
False
ans = 1
True
ShowArrayInt(pvarArray)
True
blnSwapped
False
End


FillArrayInt
(Integer Array a)
Integer i
Integer length
length = size(a)
i = 0 to length - 1
Next
Done
a[i] = random(length)
End

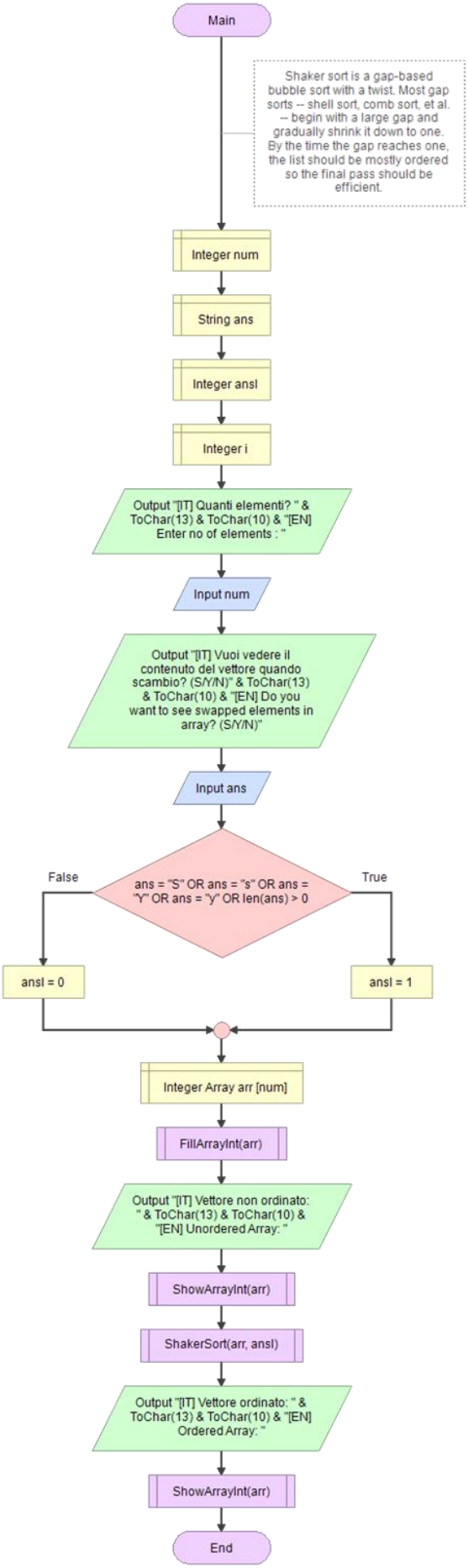

InitializeInt
(Integer Array vet)
Integer i
Integer Length
Length = size(vet)
i = 0 to length - 1
Next
Done
vet[i] = 0
End

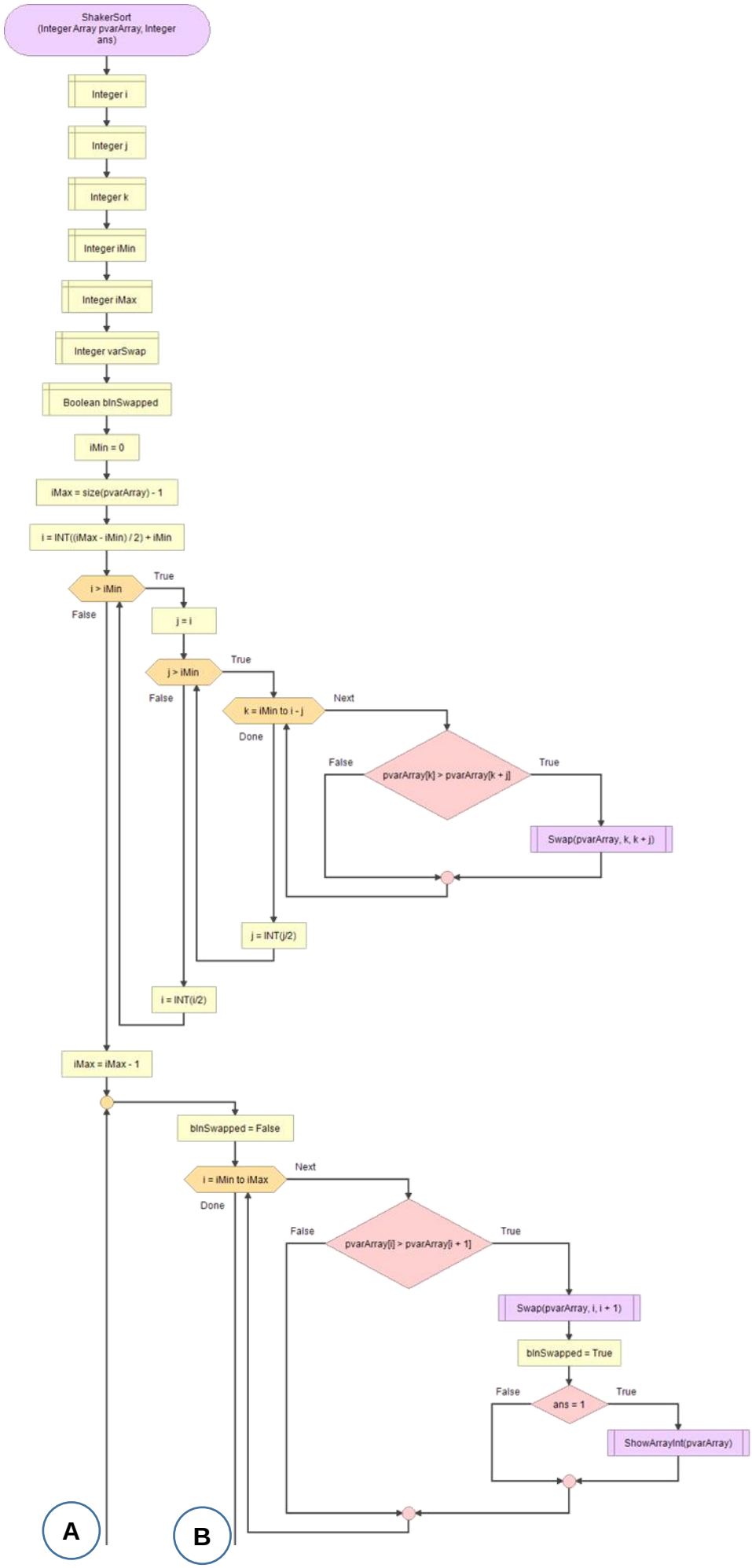

ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
False
i = length - 1
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
End

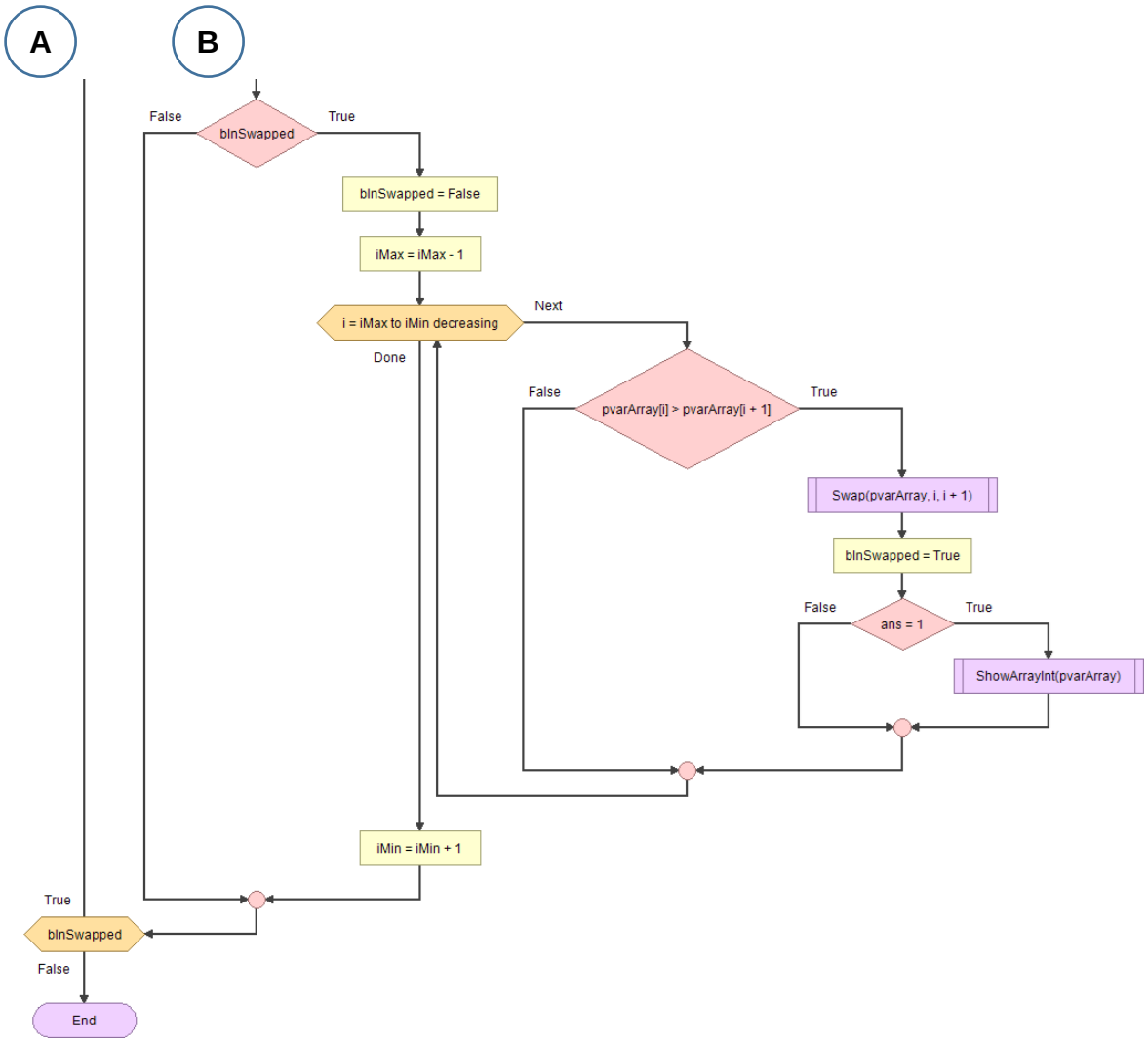

Swap
(Integer Array v, Integer i, Integer j)
Integer temp
temp = v[i]
v[i] = v[j]
v[j] = temp
End

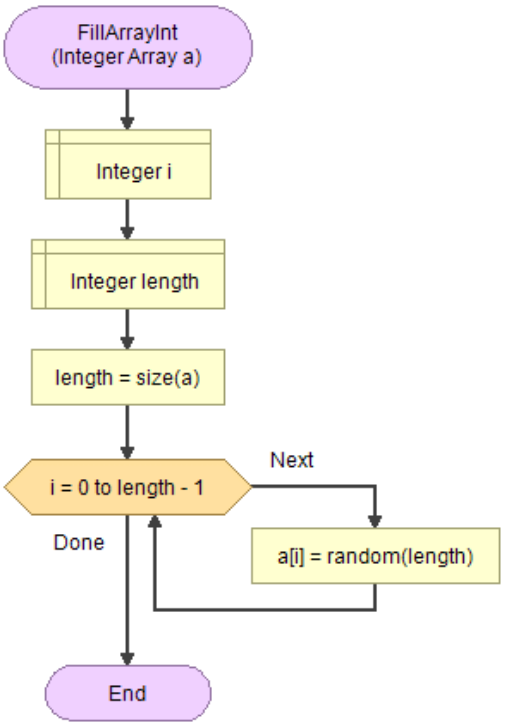
275.GAF ‖ Ordinamento Shaker

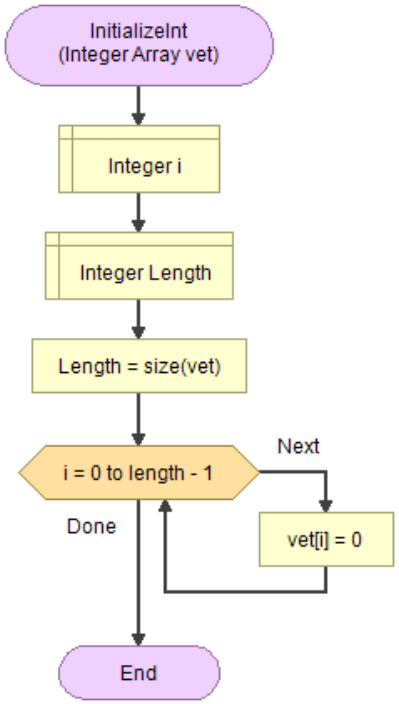
Shaker Sort

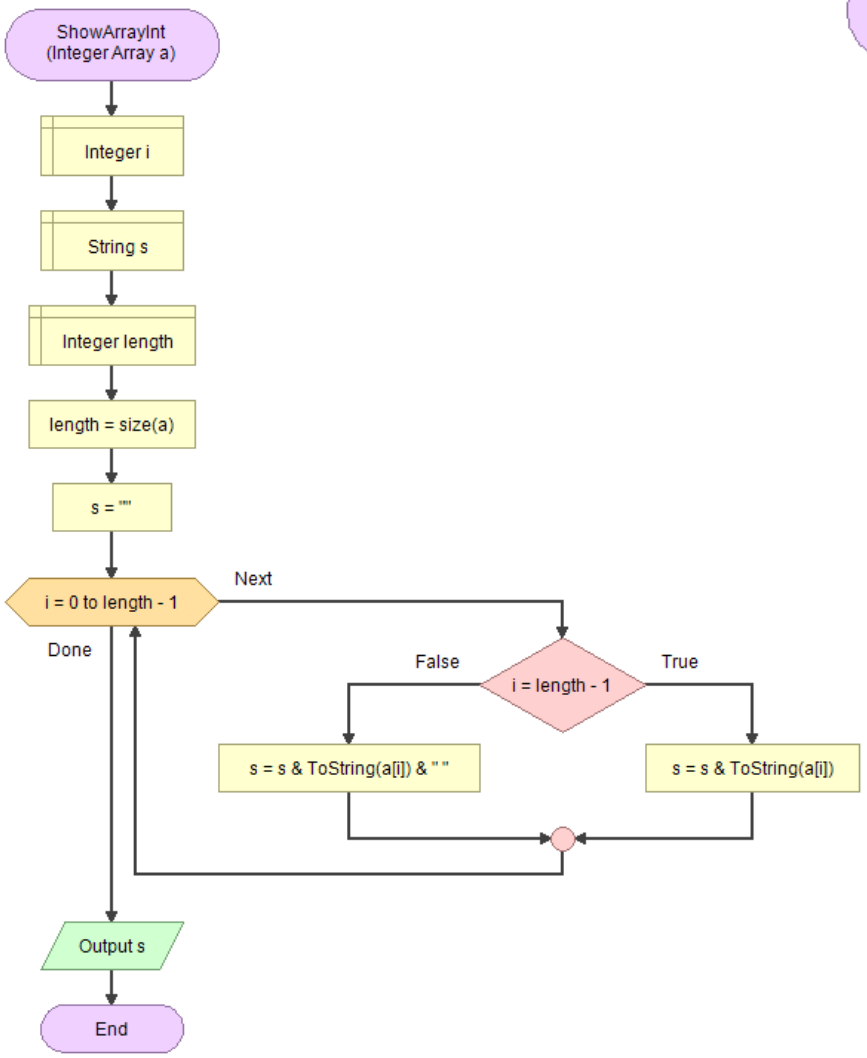


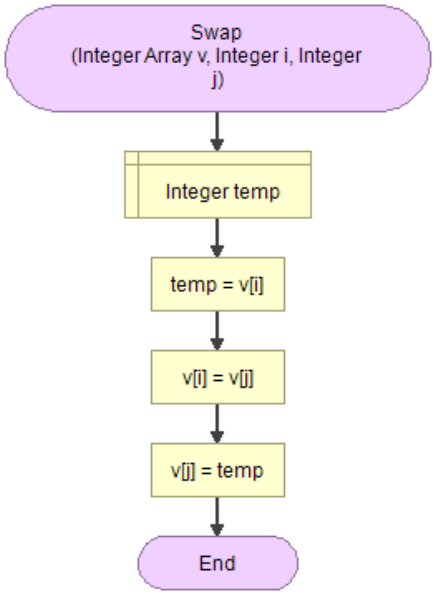

ShakerSort
(Integer Array pvarArray, Integer ans)
Integer i
Integer j
Integer k
Integer iMin
Integer iMax
Integer varSwap
Boolean blnSwapped
iMin = 0
iMax = size(pvarArray) - 1
i = INT((iMax - iMin) / 2) + iMin
i > iMin
True
False
j = i
j > iMin
True
False
k = iMin to i - j
Next
Done
pvarArray[k] > pvarArray[k + j]
False
True
Swap(pvarArray, k, k + j)
j = INT(j/2)
i = INT(i/2)
iMax = iMax - 1
blnSwapped = False
i = iMin to iMax
Next
Done
pvarArray[i] > pvarArray[i + 1]
False
True
Swap(pvarArray, i, i + 1)
blnSwapped = True
ans = 1
False
True
ShowArrayInt(pvarArray)
A
B


A
B
False
blnSwapped
True
blnSwapped = False
iMax = iMax - 1
i = iMax to iMin decreasing
Next
Done
False
pvarArray[i] > pvarArray[i + 1]
True
Swap(pvarArray, i, i + 1)
blnSwapped = True
False
ans = 1
True
ShowArrayInt(pvarArray)
iMin = iMin + 1
True
blnSwapped
False
End


FillArrayInt
(Integer Array a)
Integer i
Integer length
length = size(a)
i = 0 to length - 1
Next
Done
a[i] = random(length)
End

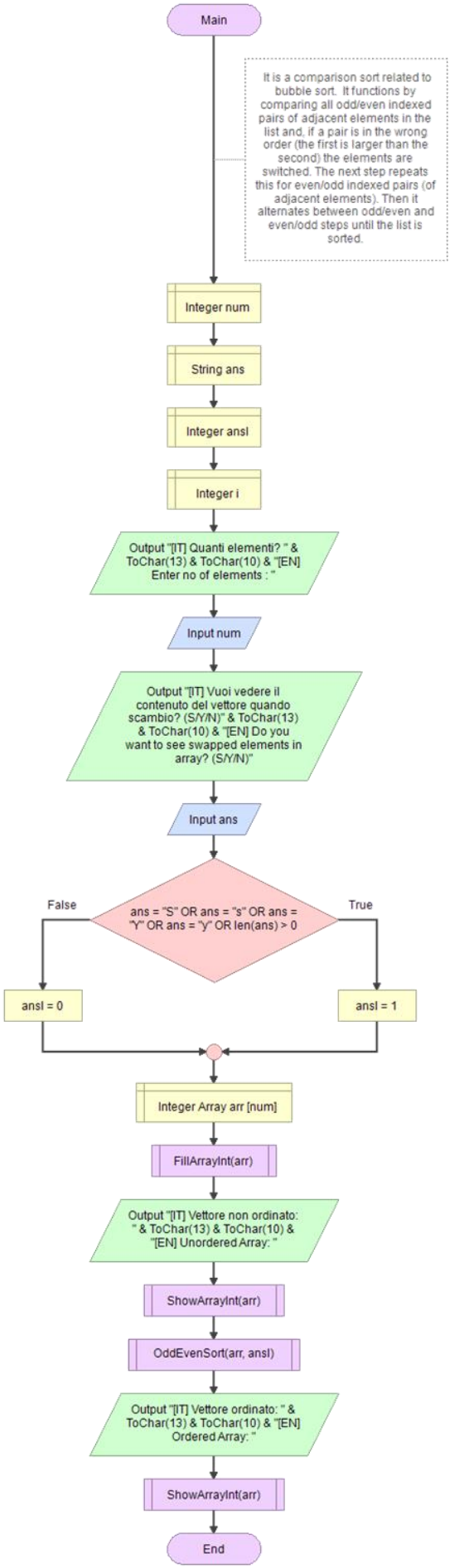

InitializeInt
(Integer Array vet)
Integer i
Integer Length
Length = size(vet)
i = 0 to length - 1
Next
Done
vet[i] = 0
End

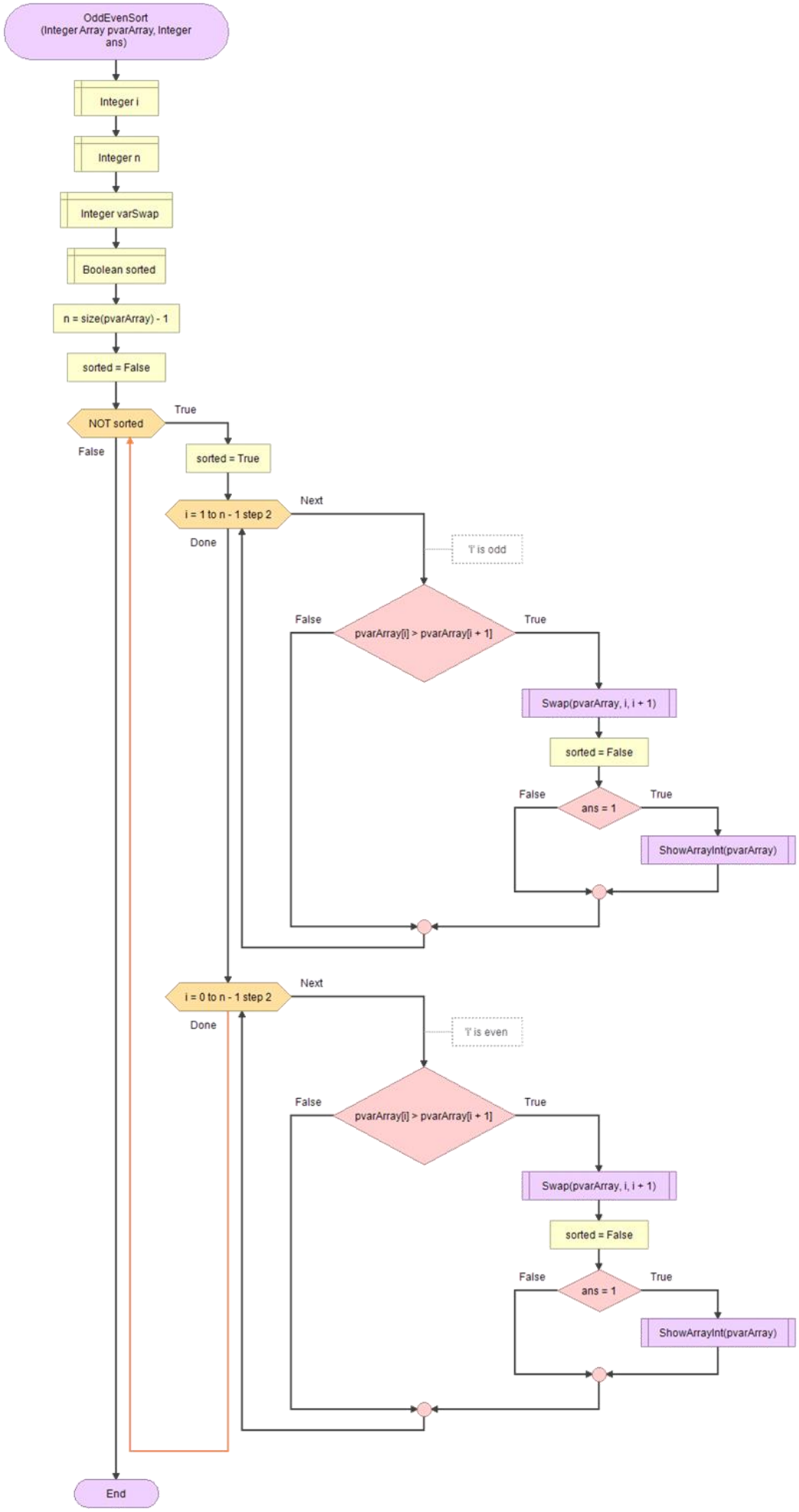

ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
False
i = length - 1
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
End

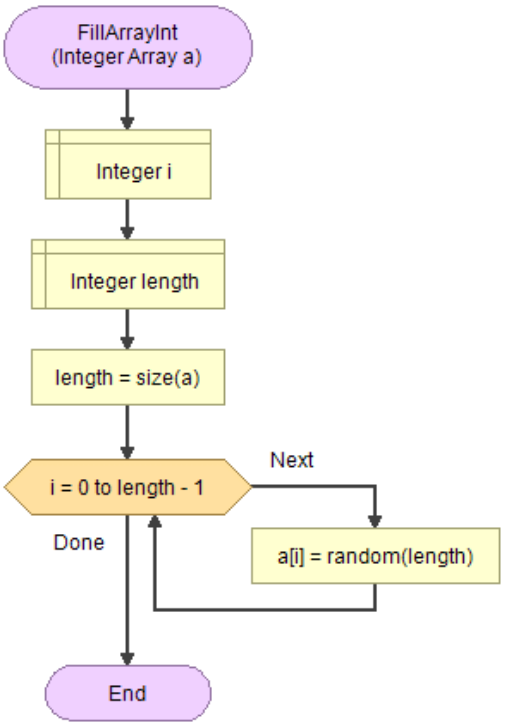

Swap
(Integer Array v, Integer i, Integer j)
Integer temp
temp = v[i]
v[i] = v[j]
v[j] = temp
End

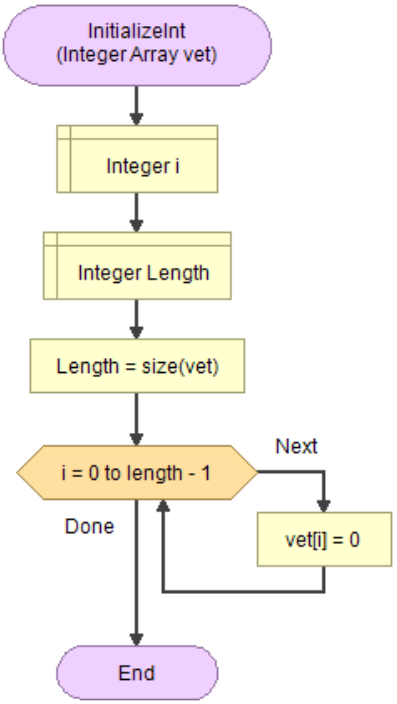
276.GAF ‖ Ordinamento Pari-Dispari

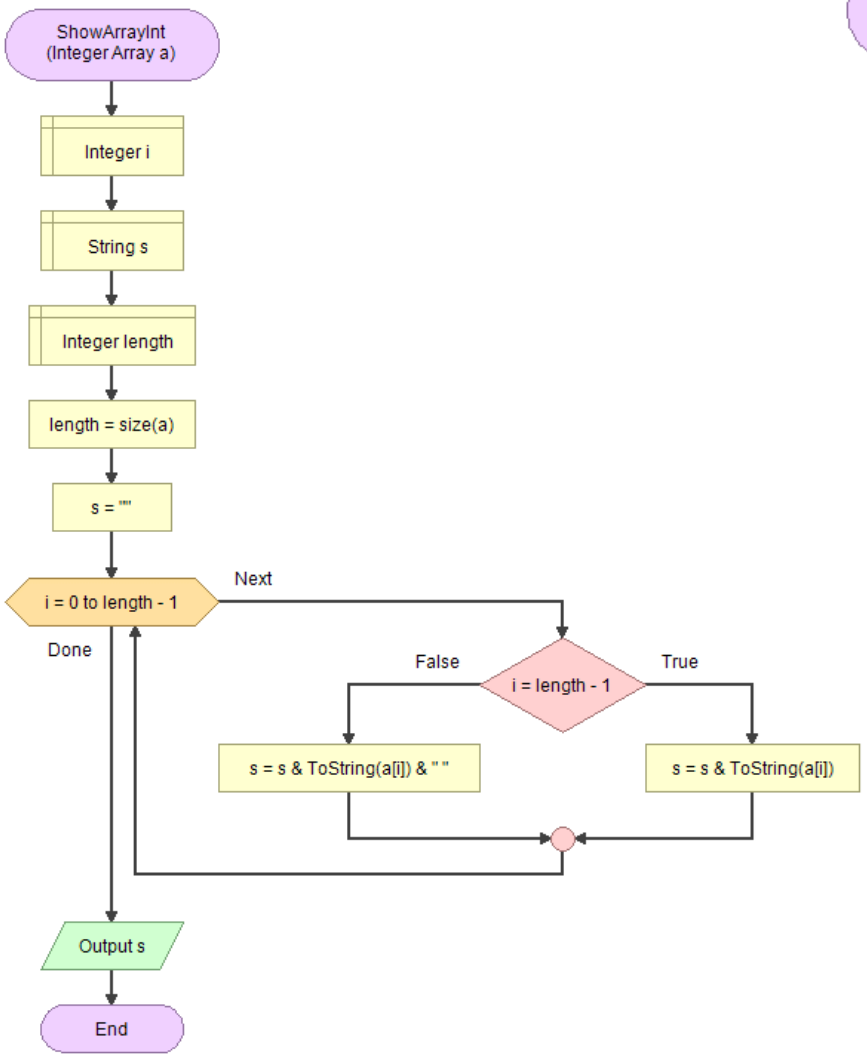
Odd-Even Sort

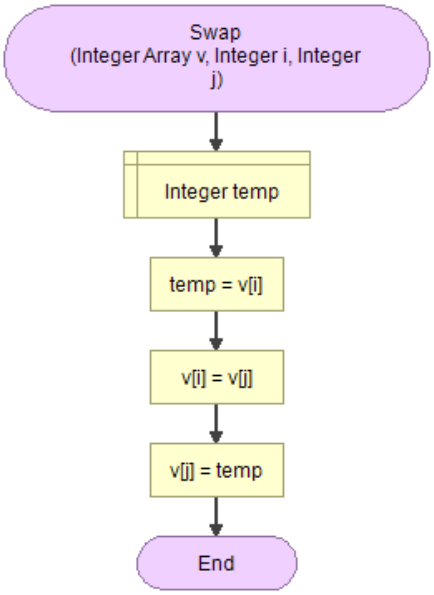



OddEvenSort
(Integer Array pvarArray, Integer ans)
Integer i
Integer n
Integer varSwap
Boolean sorted
n = size(pvarArray) - 1
sorted = False
NOT sorted
True
False
sorted = True
i = 1 to n - 1 step 2
Next
Done
'i' is odd
False
pvarArray[i] > pvarArray[i + 1]
True
Swap(pvarArray, i, i + 1)
sorted = False
False
ans = 1
True
ShowArrayInt(pvarArray)
i = 0 to n - 1 step 2
Next
Done
'i' is even
False
pvarArray[i] > pvarArray[i + 1]
True
Swap(pvarArray, i, i + 1)
sorted = False
False
ans = 1
True
ShowArrayInt(pvarArray)
End


FillArrayInt
(Integer Array a)
Integer i
Integer length
length = size(a)
i = 0 to length - 1
Next
a[i] = random(length)
Done
End

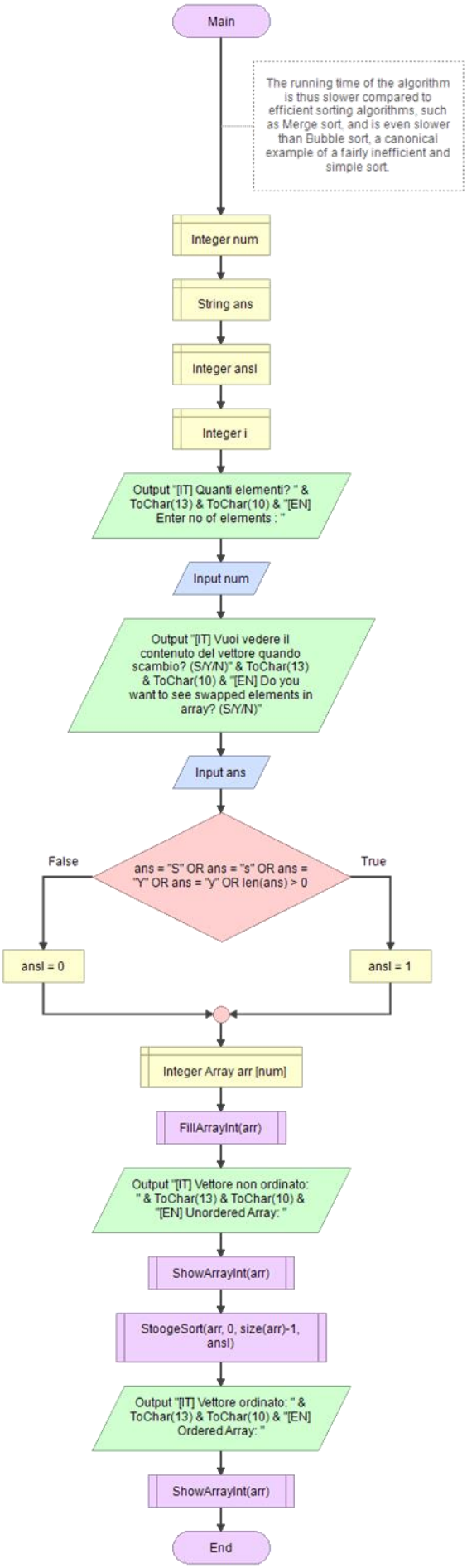

InitializeInt
(Integer Array vet)
Integer i
Integer Length
Length = size(vet)
i = 0 to length - 1
Next
vet[i] = 0
Done
End

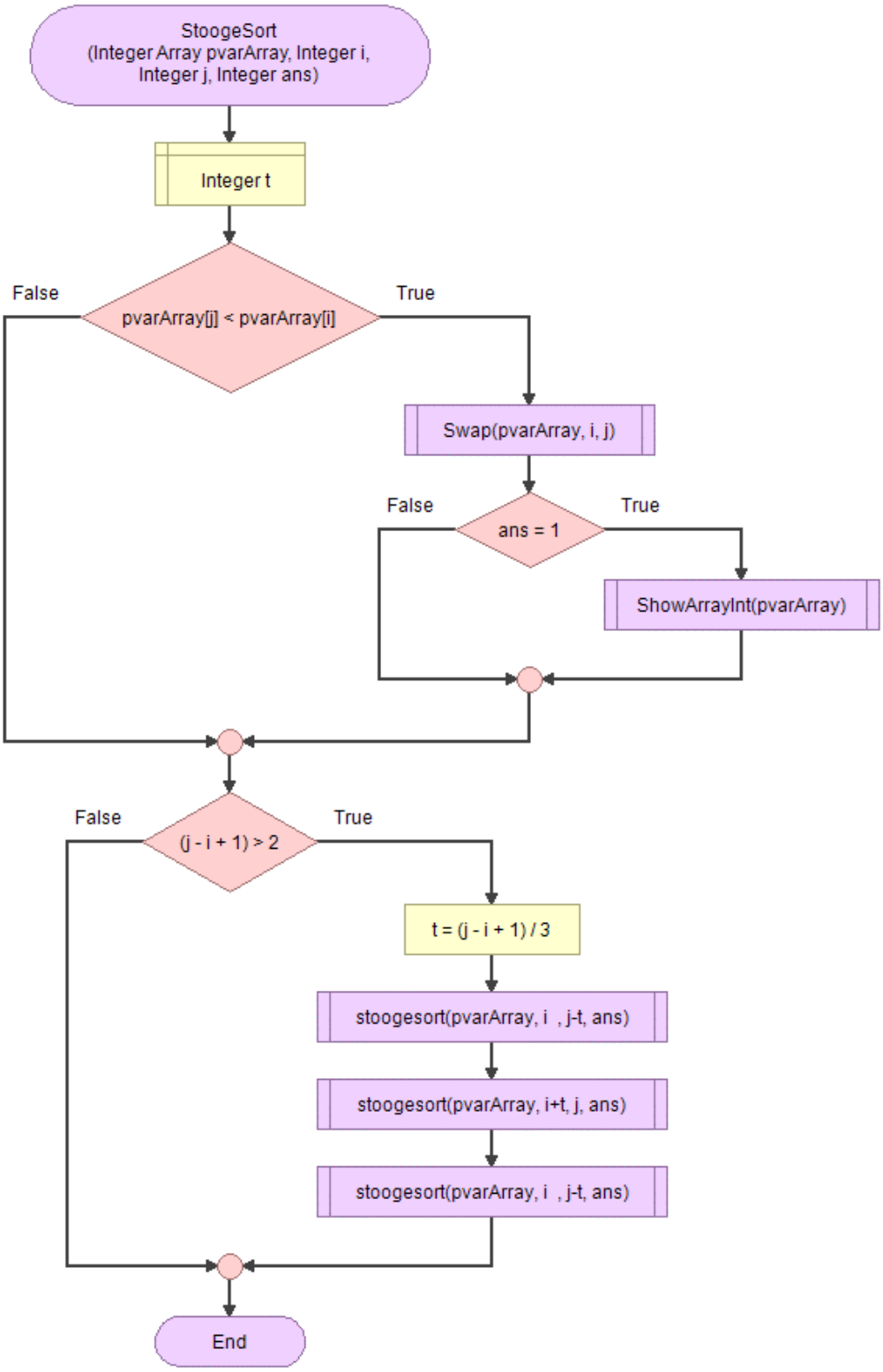

ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
i = length - 1
False
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
End

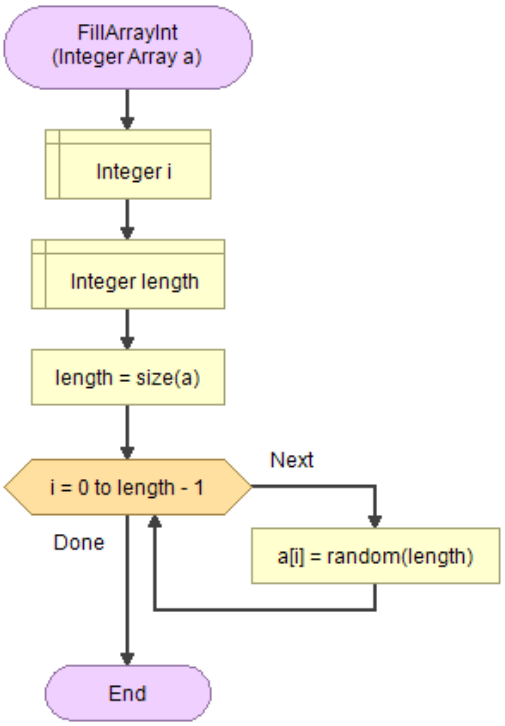

Swap
(Integer Array v, Integer i, Integer j)
Integer temp
temp = v[i]
v[i] = v[j]
v[j] = temp
End

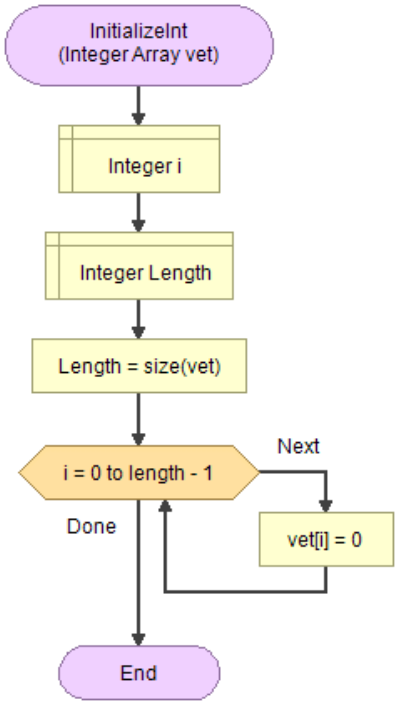
277.GAFR | Ordinamento Stooge (ricorsivo)

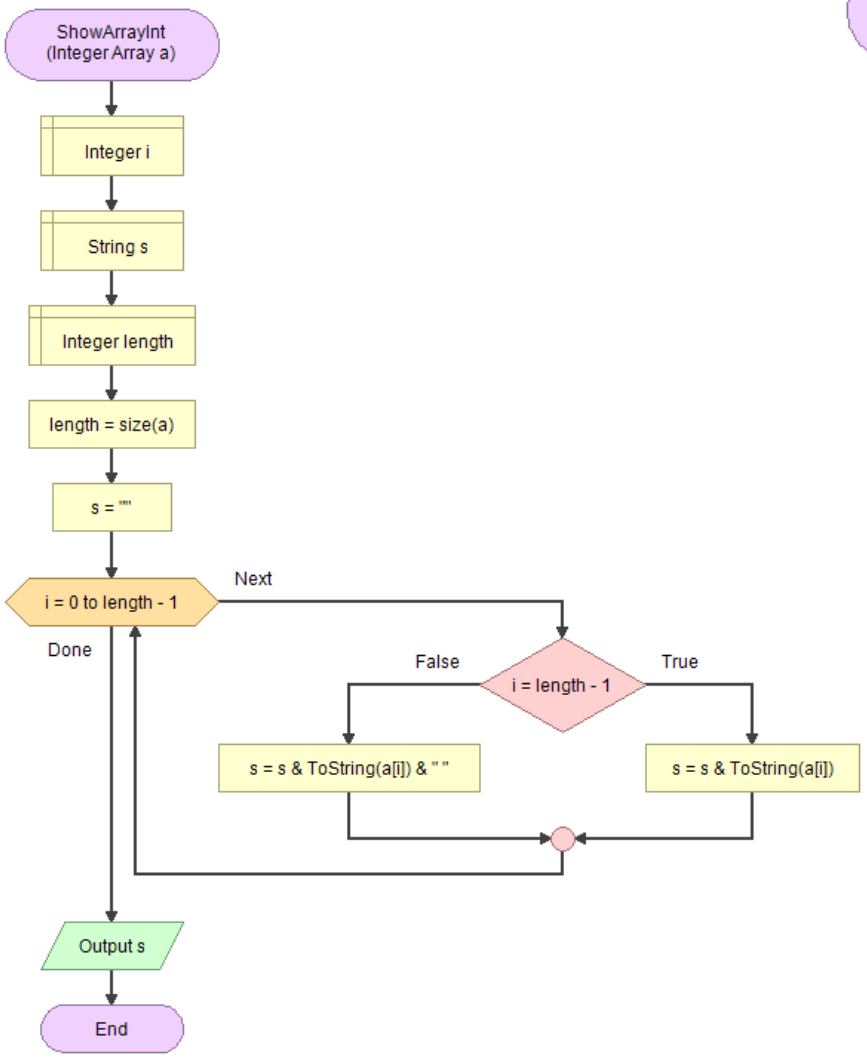
Stooge Sort (recursive)

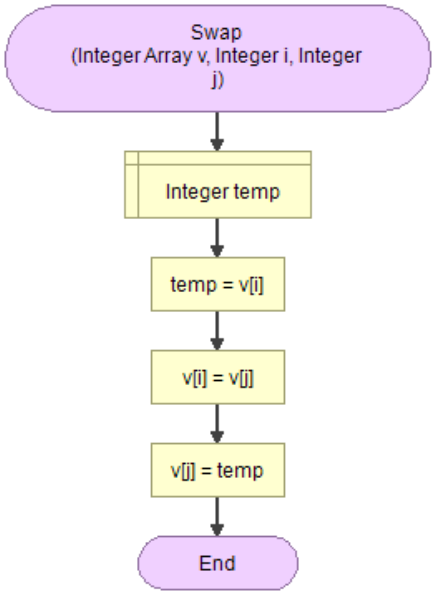



StoogeSort
(Integer Array pvarArray, Integer i,
Integer j, Integer ans)
Integer t
False
pvarArray[j] < pvarArray[i]
True
Swap(pvarArray, i, j)
False
ans = 1
True
ShowArrayInt(pvarArray)
False
(j - i + 1) > 2
True
t = (j - i + 1) / 3
stoogesort(pvarArray, i , j-t, ans)
stoogesort(pvarArray, i+t, j, ans)
stoogesort(pvarArray, i , j-t, ans)
End


FillArrayInt
(Integer Array a)
Integer i
Integer length
length = size(a)
i = 0 to length - 1
Next
Done
a[i] = random(length)
End

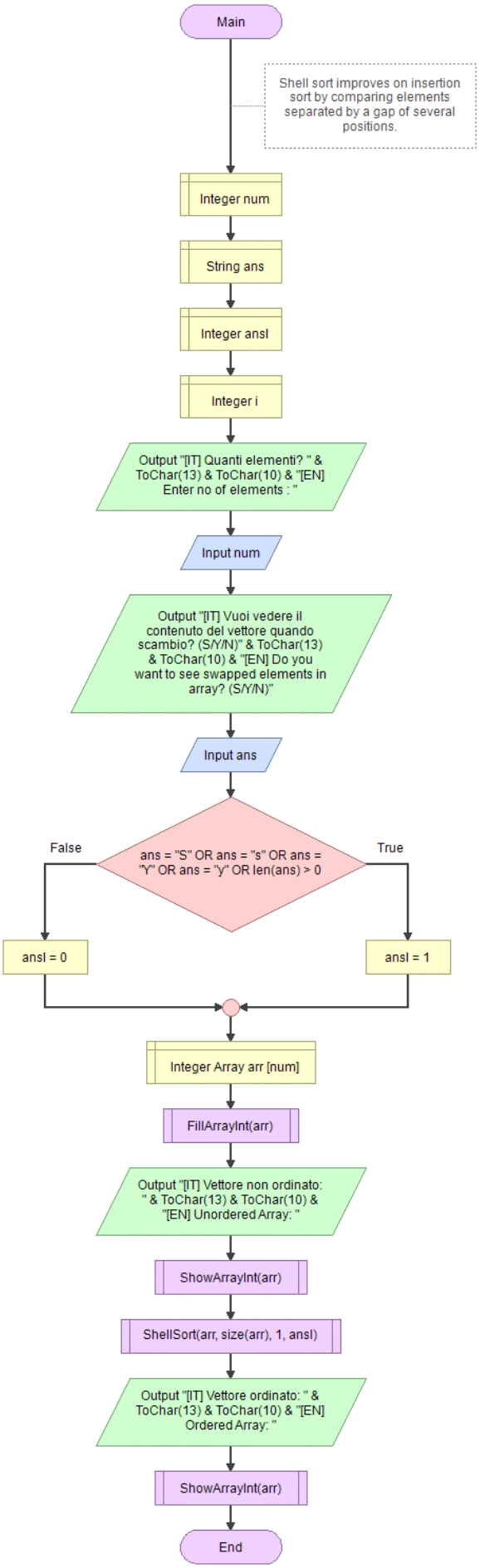

InitializeInt
(Integer Array vet)
Integer i
Integer Length
Length = size(vet)
i = 0 to length - 1
Next
Done
vet[i] = 0
End

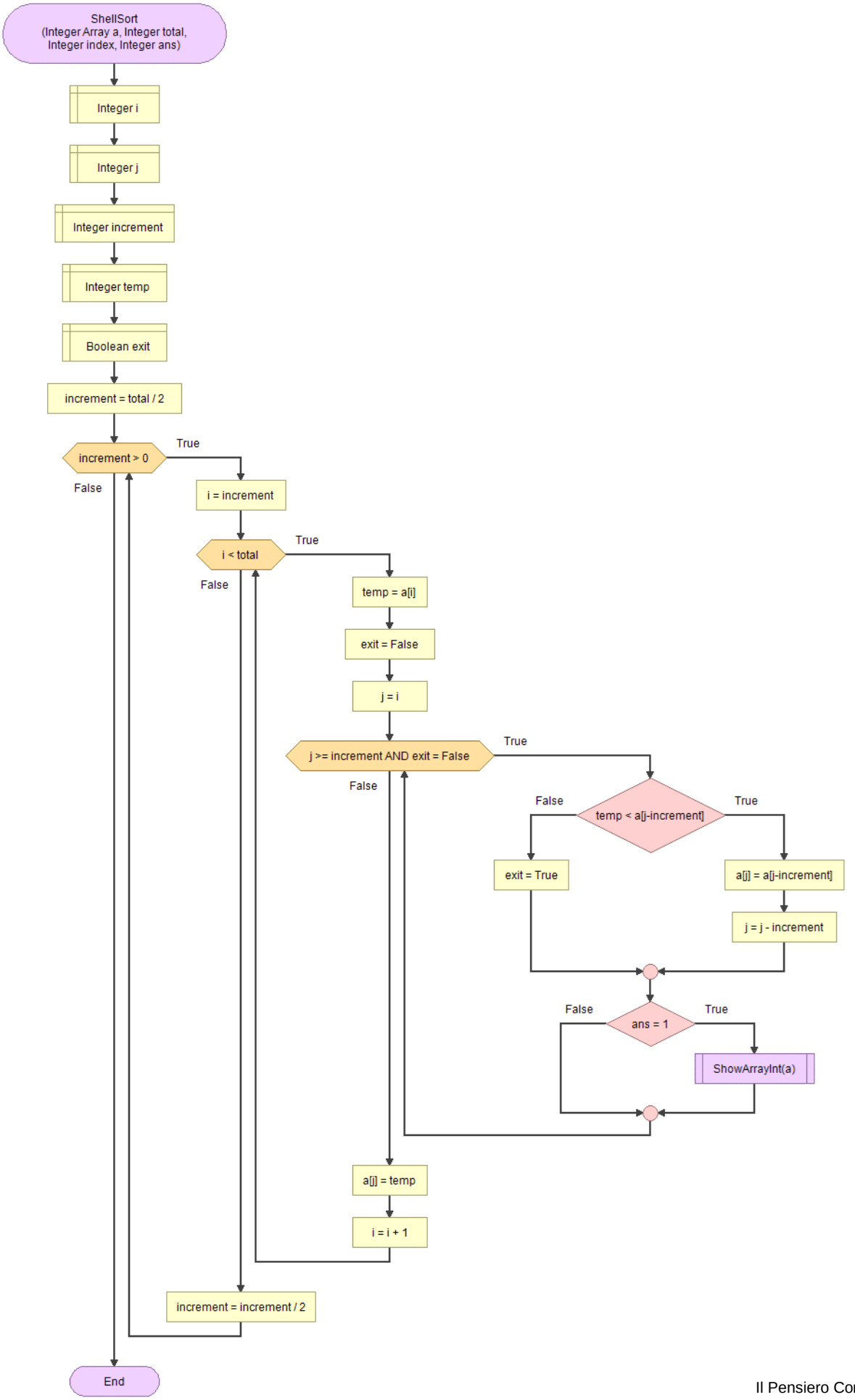

ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
False
i = length - 1
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
End

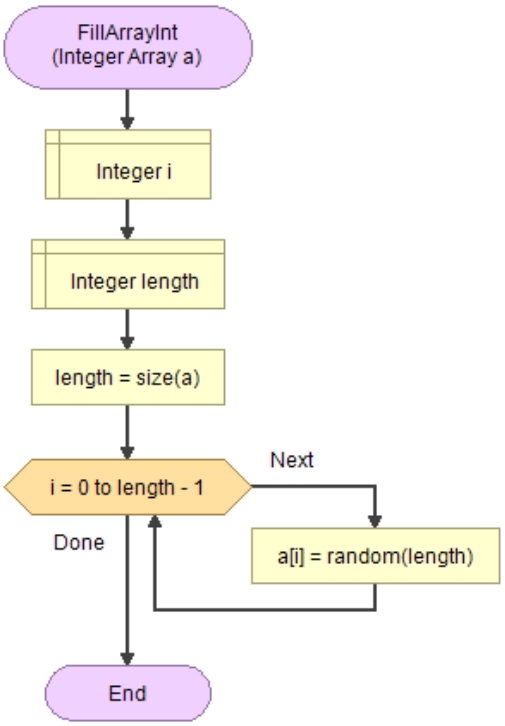

Swap
(Integer Array v, Integer i, Integer j)
Integer temp
temp = v[i]
v[i] = v[j]
v[j] = temp
End

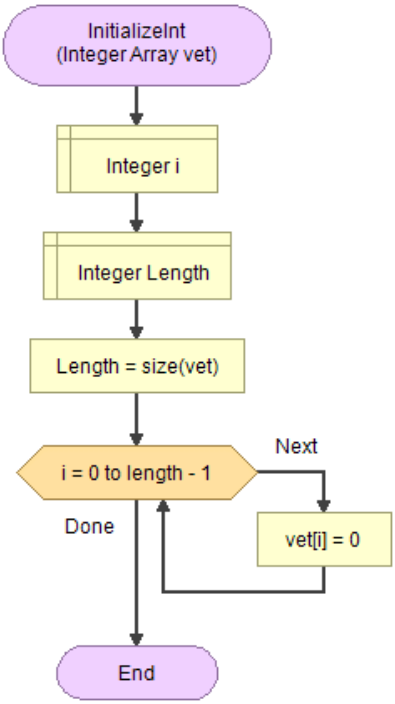
278.GAF ‖ Ordinamento Shell Sort

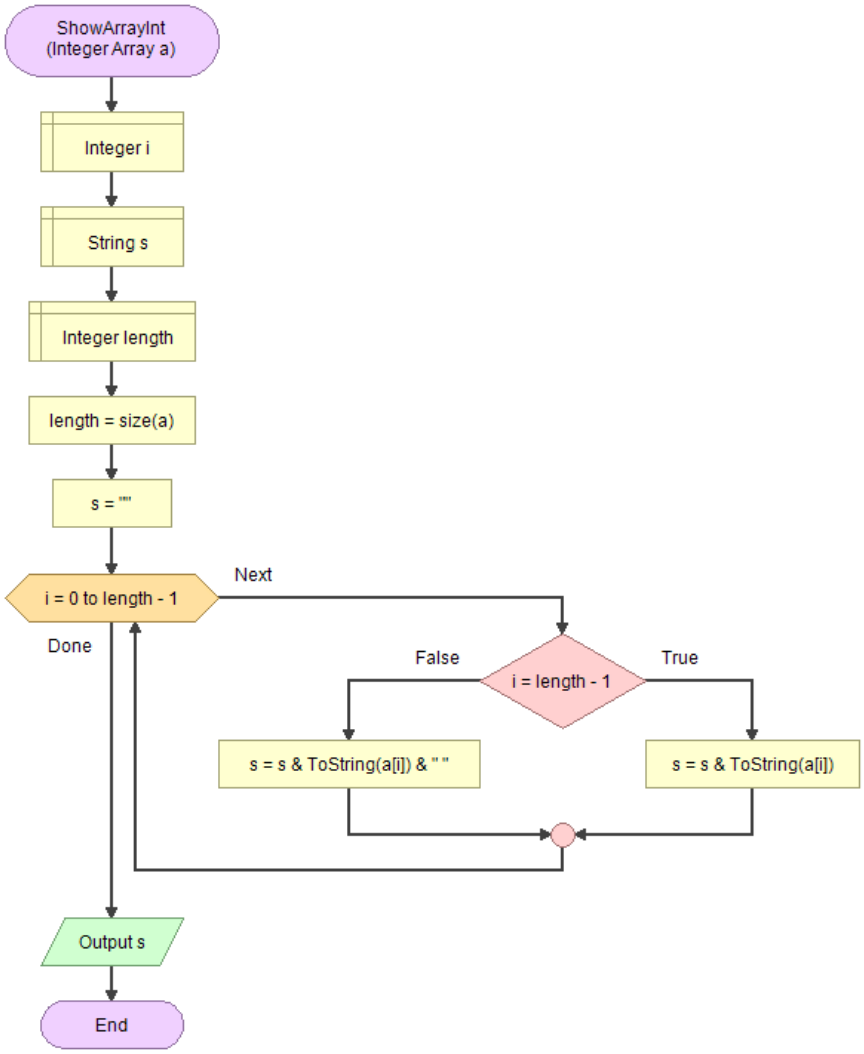
Shell Sort




ShellSort
(Integer Array a, Integer total,
Integer index, Integer ans)
Integer i
Integer j
Integer increment
Integer temp
Boolean exit
increment = total / 2
increment > 0
True
False
i = increment
i < total
True
False
temp = a[i]
exit = False
j = i
j >= increment AND exit = False
True
False
temp < a[j-increment]
False
True
exit = True
a[j] = a[j-increment]
j = j - increment
ans = 1
False
True
ShowArrayInt(a)
a[j] = temp
i = i + 1
increment = increment / 2
End

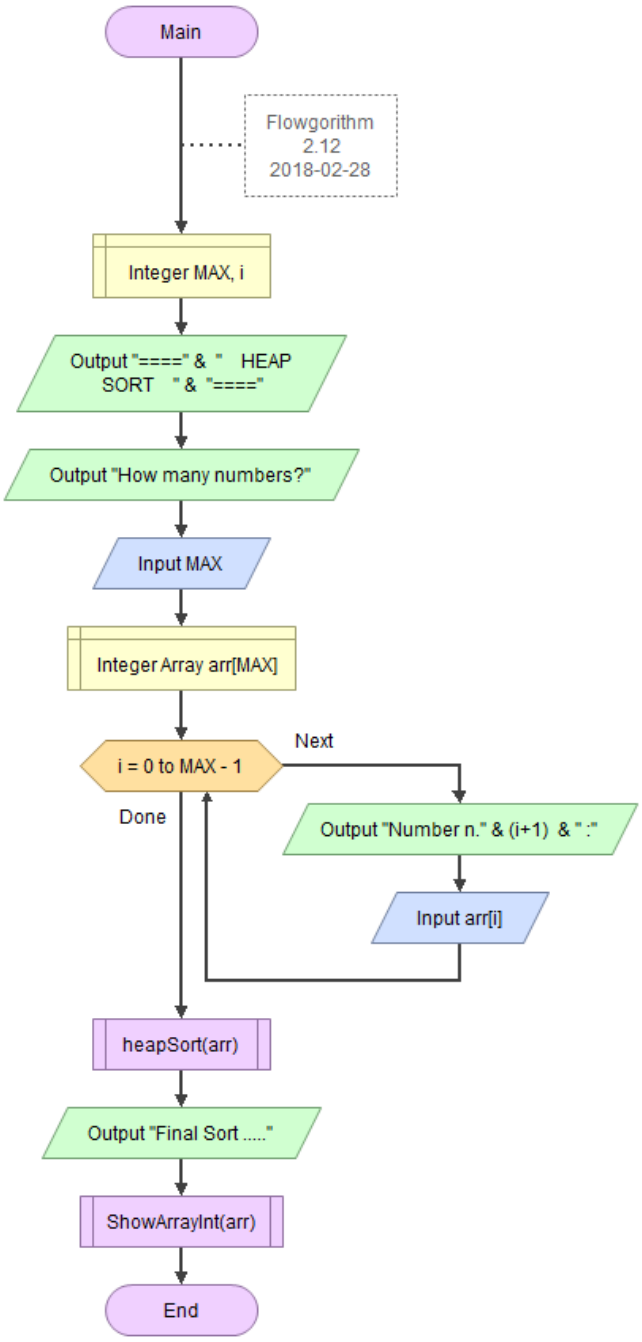

FillArrayInt
(Integer Array a)
Integer i
Integer length
length = size(a)
i = 0 to length - 1
Next
a[i] = random(length)
Done
End

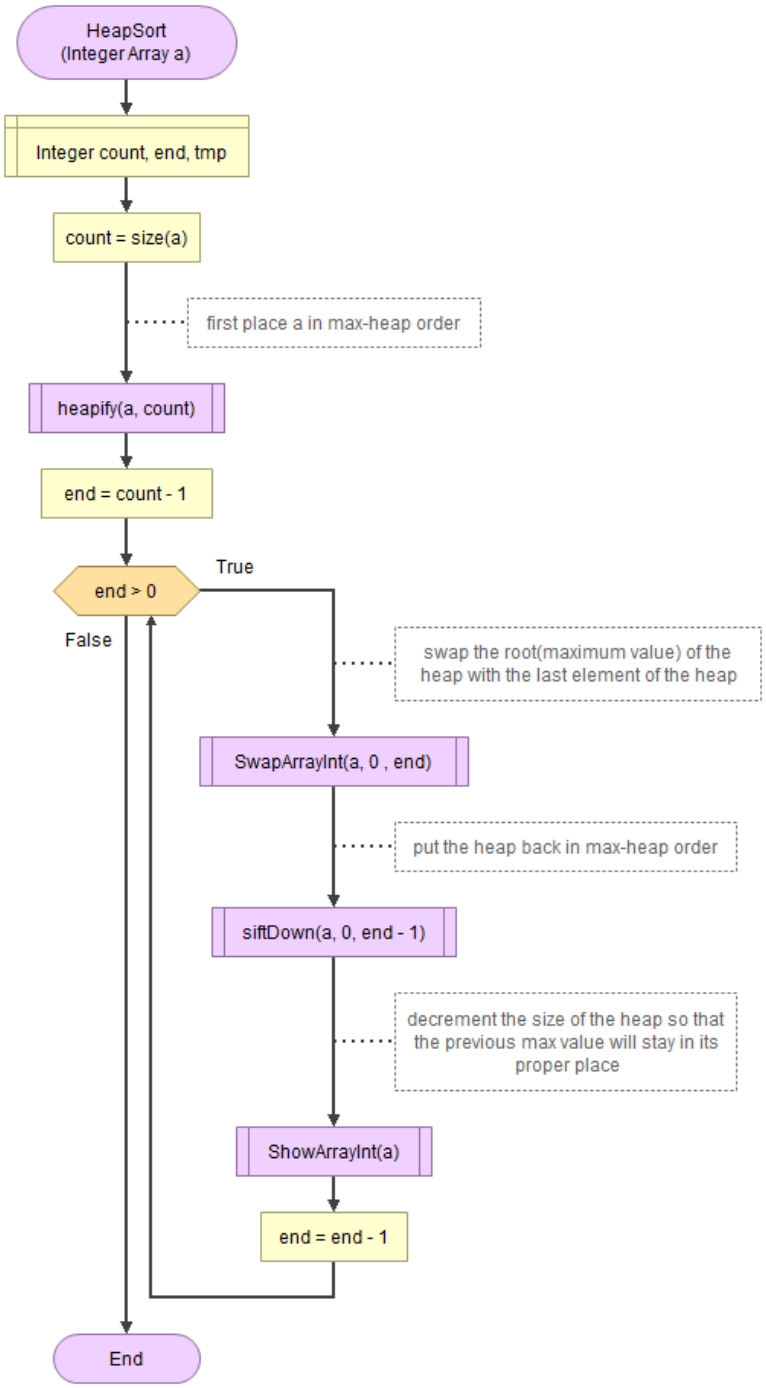

InitializeInt
(Integer Array vet)
Integer i
Integer Length
Length = size(vet)
i = 0 to length - 1
Next
vet[i] = 0
Done
End

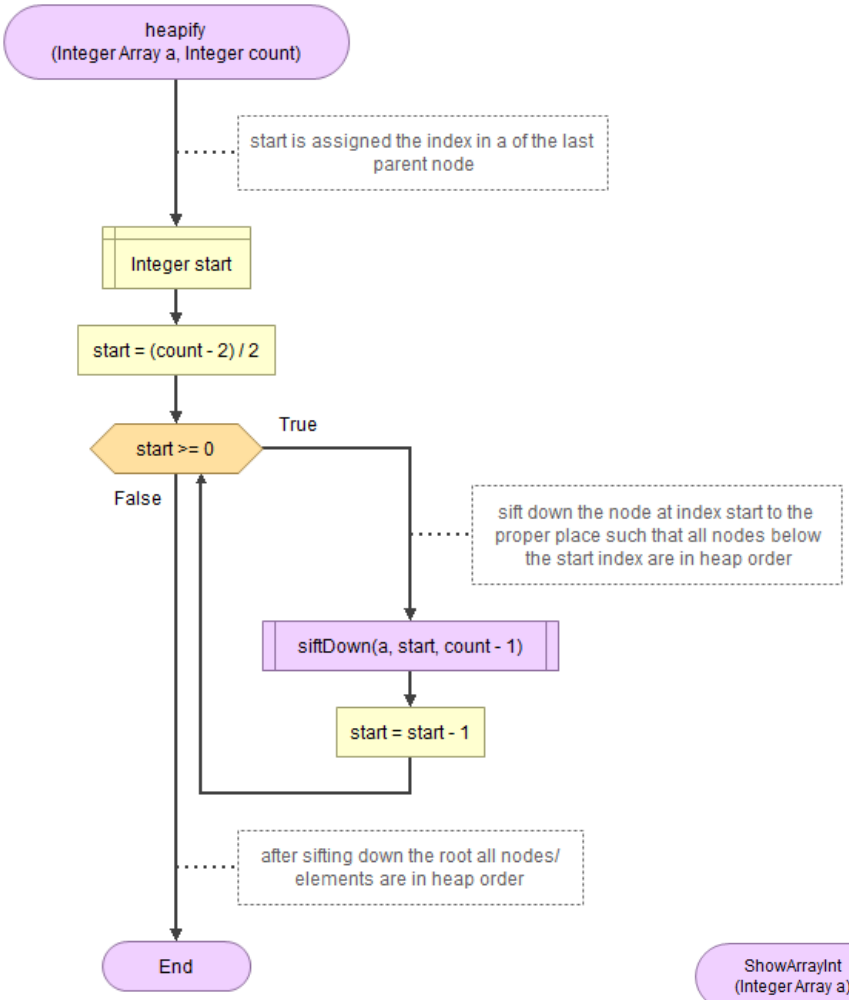

ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
False
i = length - 1
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
End

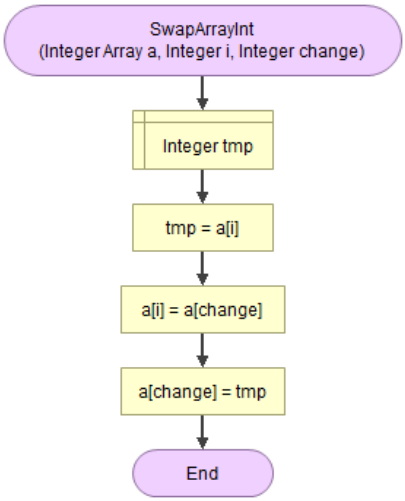
279.GAF ‖ Ordinamento Heap Sort

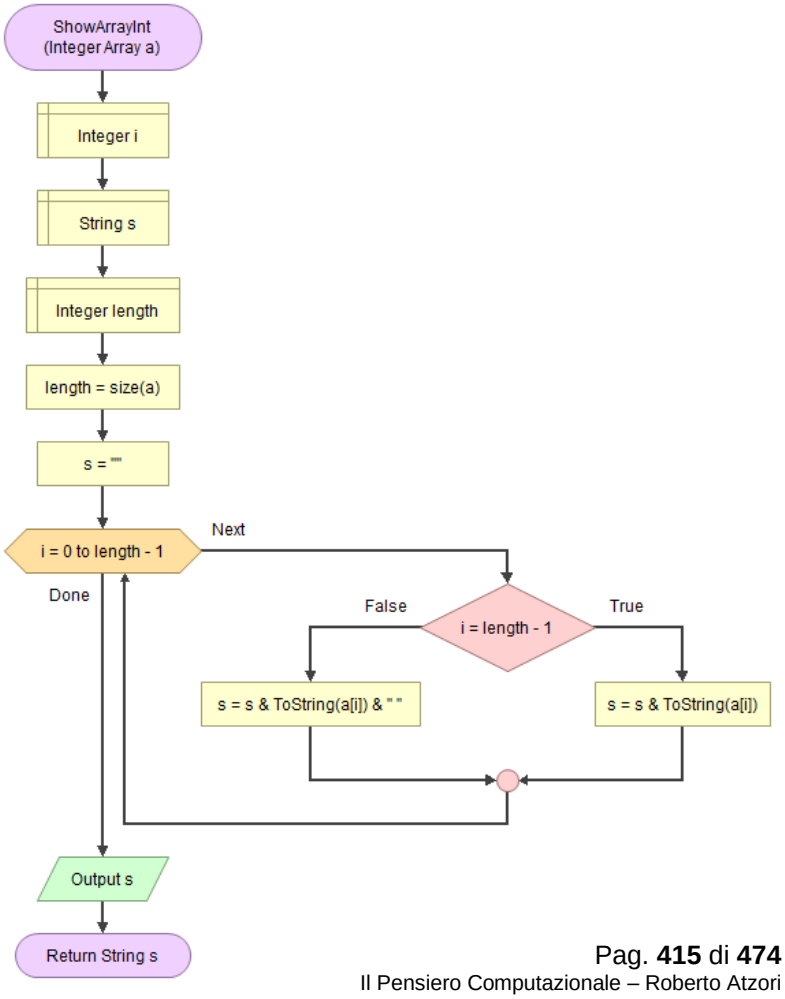
Heap Sort

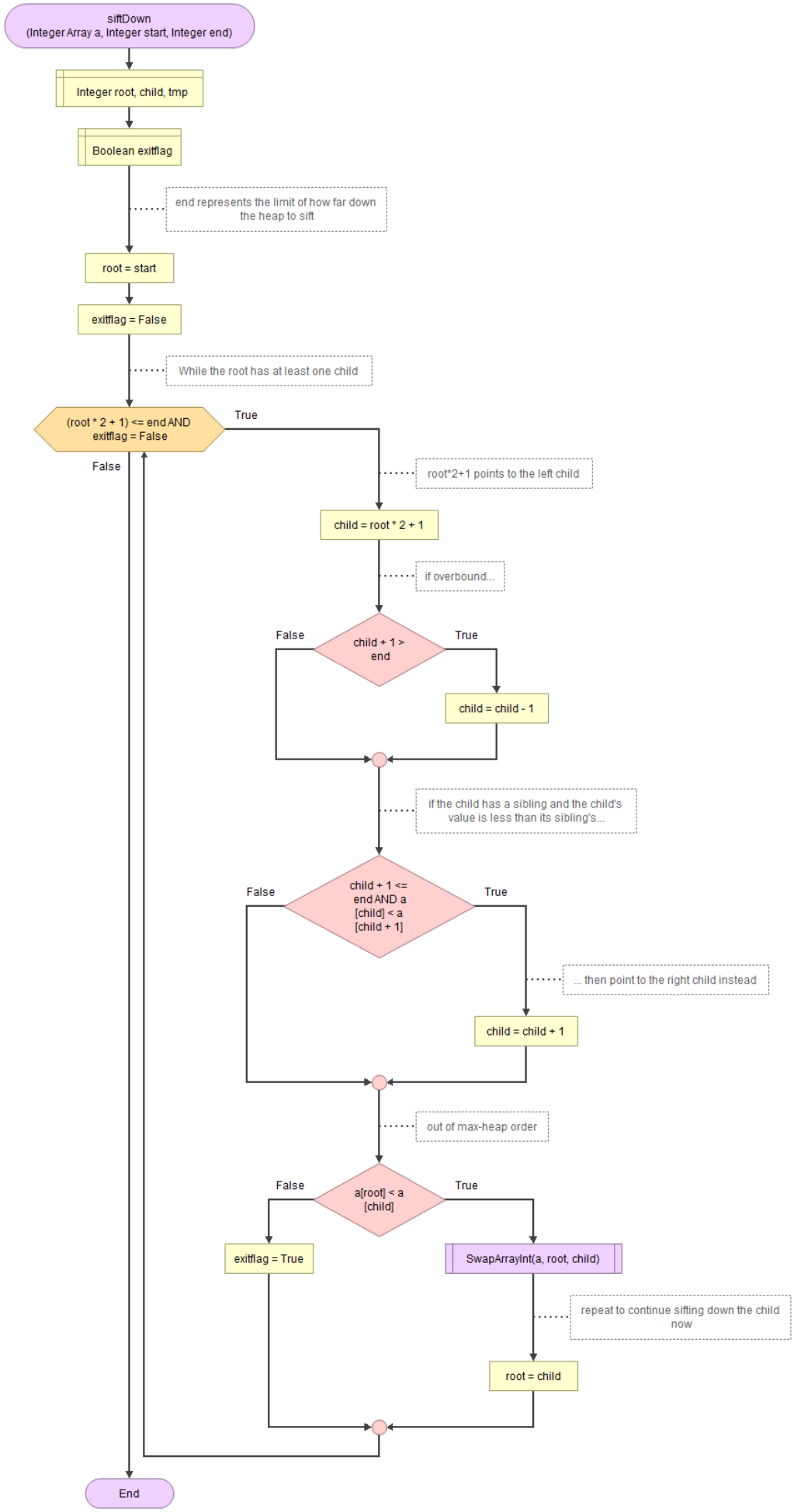





heapify
(Integer Array a, Integer count)
start is assigned the index in a of the last parent node
Integer start
start = (count - 2) / 2
start >= 0
True
False
sift down the node at index start to the proper place such that all nodes below the start index are in heap order
siftDown(a, start, count - 1)
start = start - 1
after sifting down the root all nodes/ elements are in heap order
End

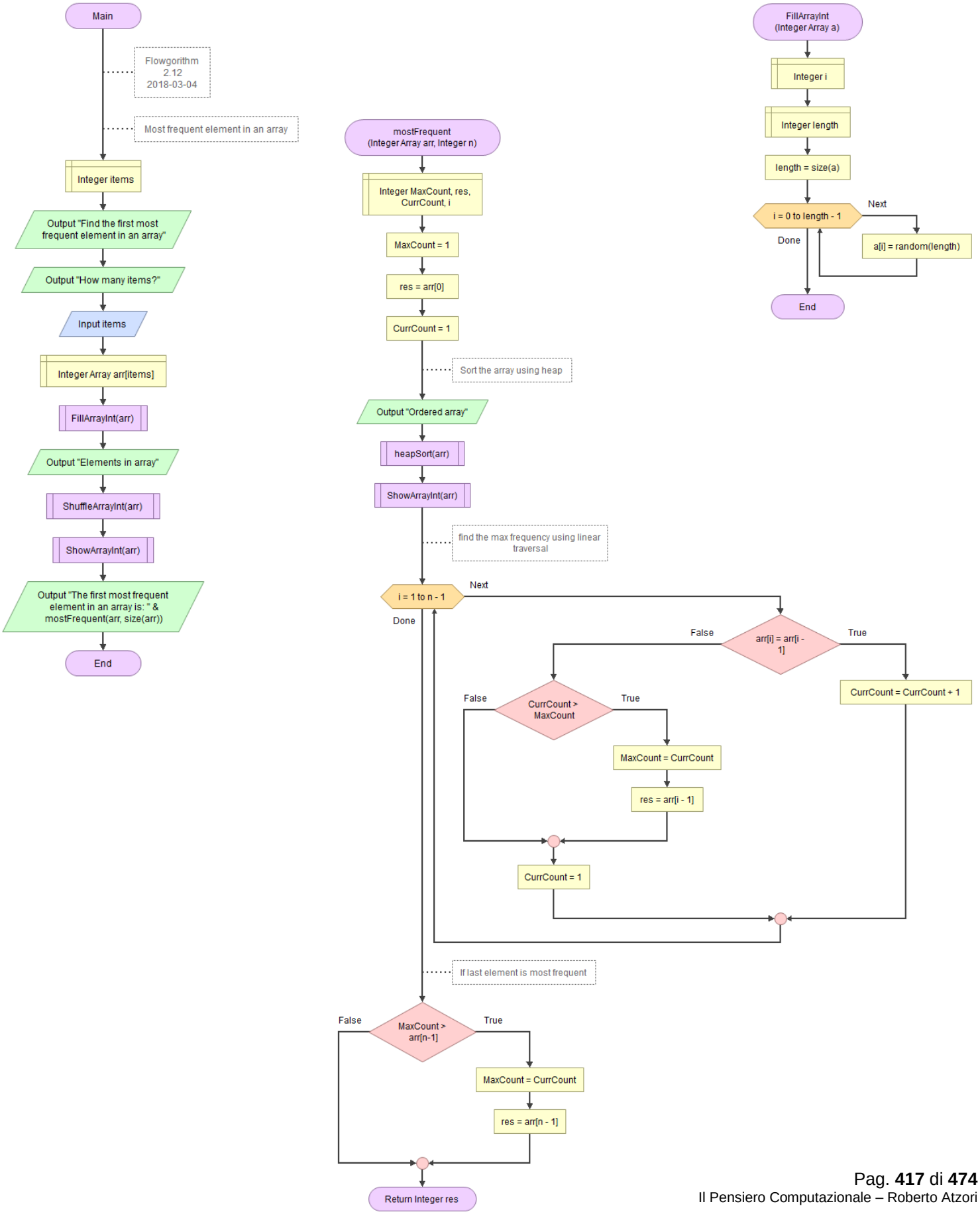

SwapArrayInt
(Integer Array a, Integer i, Integer change)
Integer tmp
tmp = a[i]
a[i] = a[change]
a[change] = tmp
End

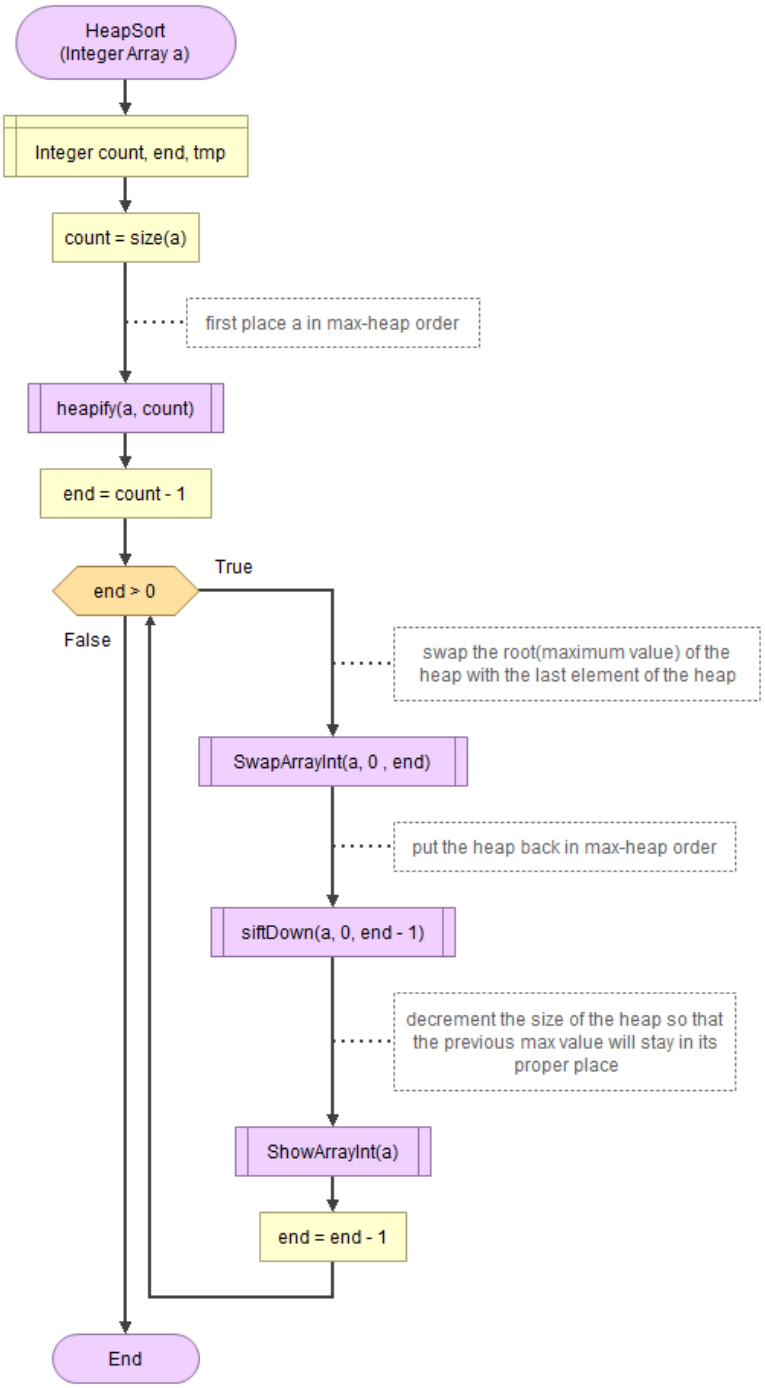

ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
False
i = length - 1
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
Return String s

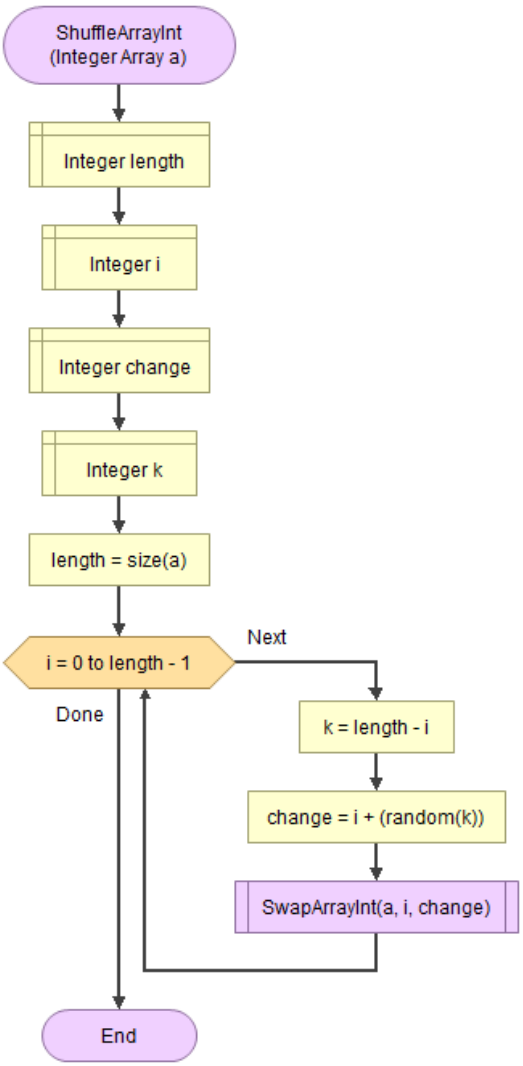

siftDown
(Integer Array a, Integer start, Integer end)
Integer root, child, tmp
Boolean exitflag
end represents the limit of how far down the heap to sift
root = start
exitflag = False
While the root has at least one child
(root * 2 + 1) <= end AND exitflag = False
True
False
root*2+1 points to the left child
child = root * 2 + 1
if overbound...
child + 1 > end
False
True
child = child - 1
if the child has a sibling and the child's value is less than its sibling's...
child + 1 <= end AND a [child] < a [child + 1]
False
True
... then point to the right child instead
child = child + 1
out of max-heap order
a[root] < a [child]
False
True
exitflag = True
SwapArrayInt(a, root, child)
repeat to continue sifting down the child now
root = child
End

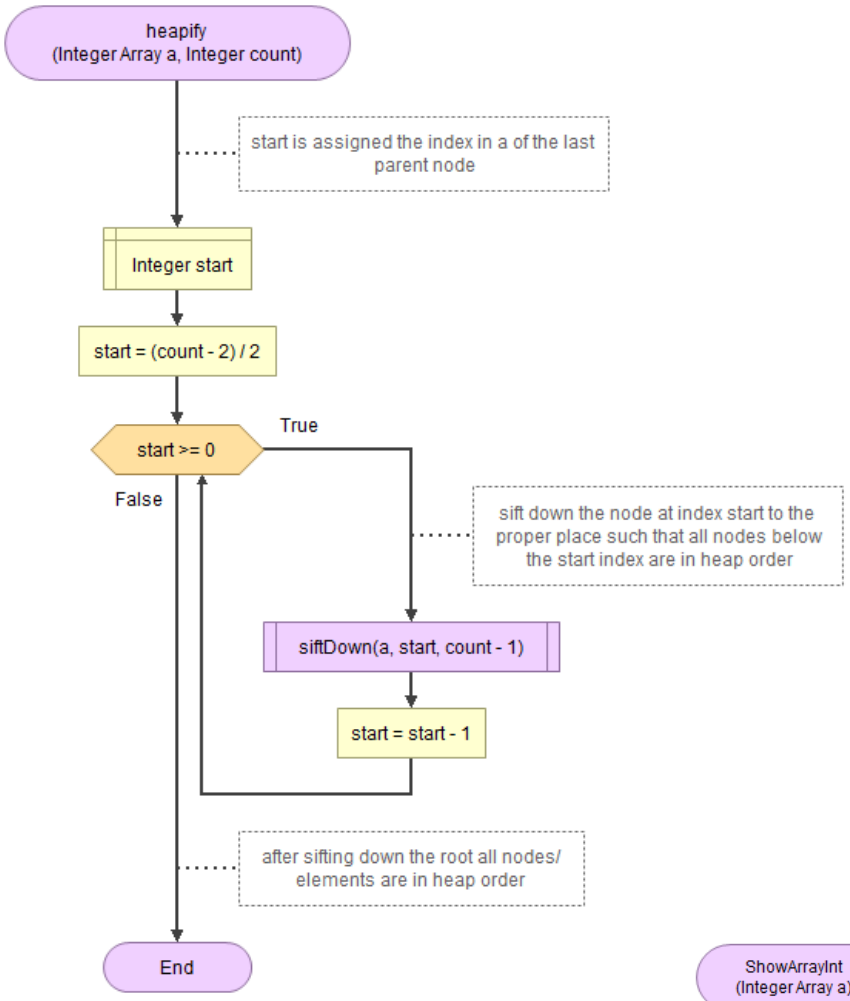
280. G A F ‖ Trovare il primo numero più frequente in una vettore (con Heap Sort)

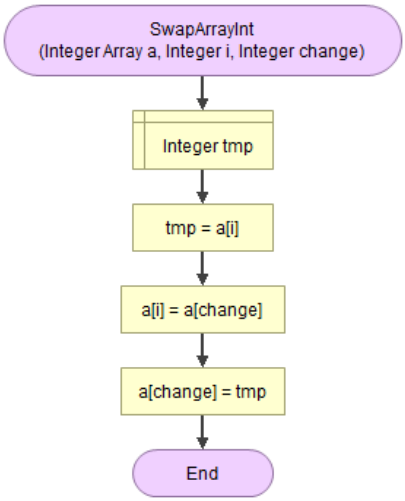
First most frequent element in an array (with heapsort)

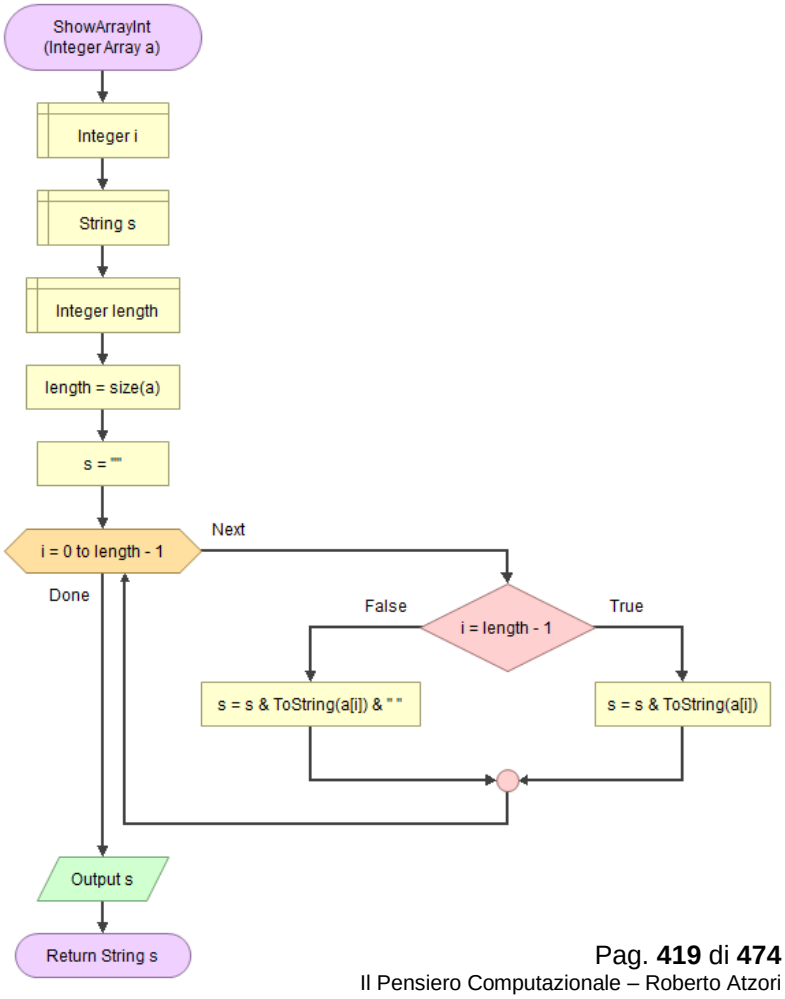


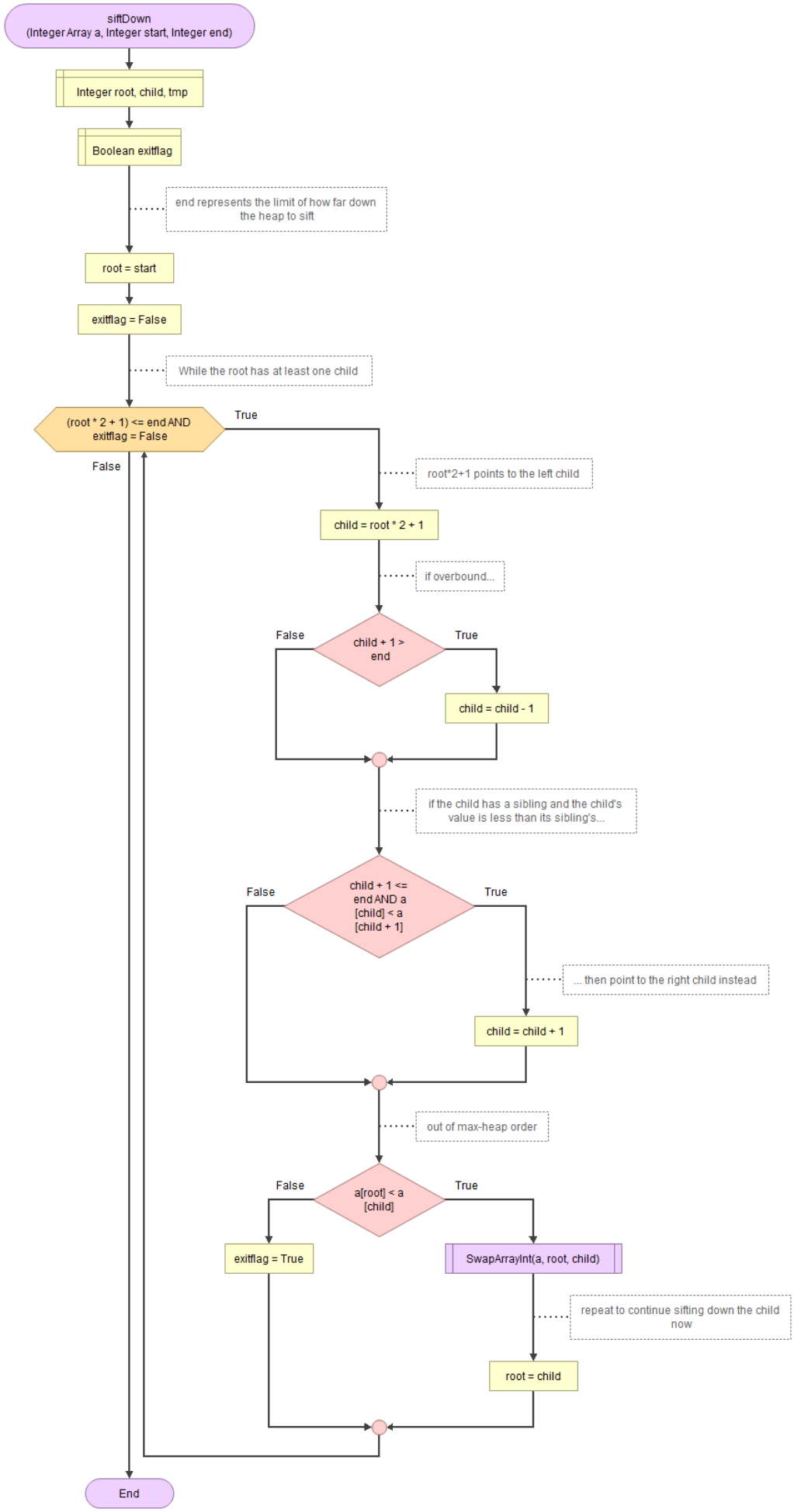

HeapSort
(Integer Array a)
Integer count, end, tmp
count = size(a)
first place a in max-heap order
heapify(a, count)
end = count - 1
end > 0
True
False
swap the root(maximum value) of the heap with the last element of the heap
SwapArrayInt(a, 0 , end)
put the heap back in max-heap order
siftDown(a, 0, end - 1)
decrement the size of the heap so that the previous max value will stay in its proper place
ShowArrayInt(a)
end = end - 1
End


ShuffleArrayInt
(Integer Array a)
Integer length
Integer i
Integer change
Integer k
length = size(a)
i = 0 to length - 1
Next
Done
k = length - i
change = i + (random(k))
SwapArrayInt(a, i, change)
End


heapify
(Integer Array a, Integer count)
start is assigned the index in a of the last parent node
Integer start
start = (count - 2) / 2
start >= 0
True
False
sift down the node at index start to the proper place such that all nodes below the start index are in heap order
siftDown(a, start, count - 1)
start = start - 1
after sifting down the root all nodes/ elements are in heap order
End

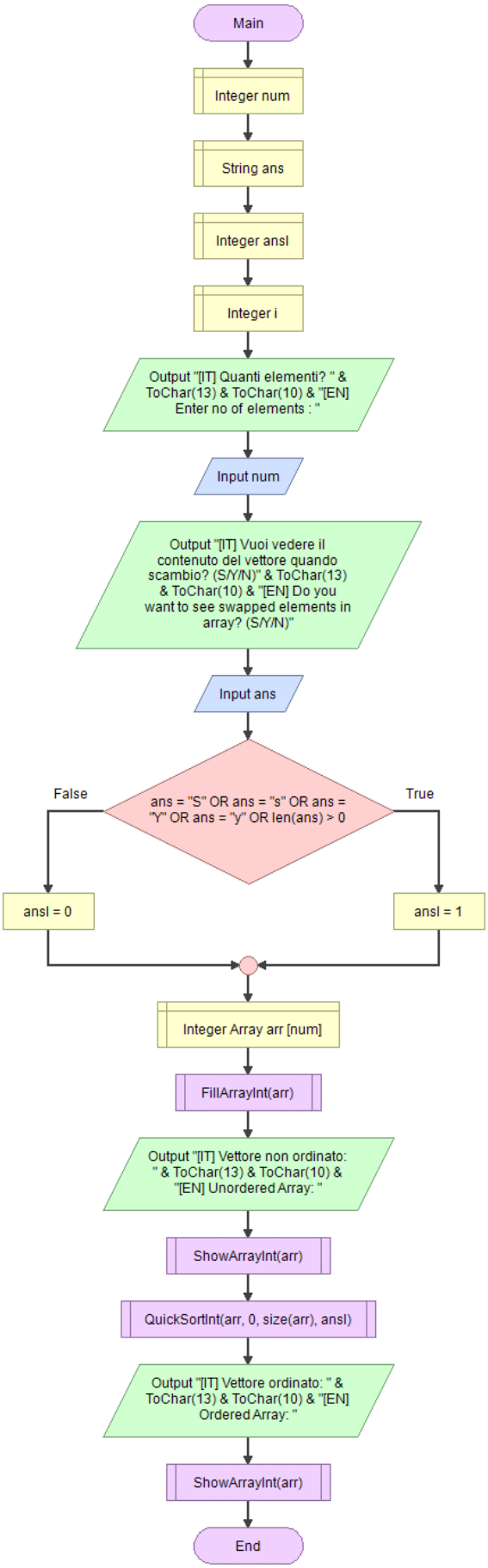

SwapArrayInt
(Integer Array a, Integer i, Integer change)
Integer tmp
tmp = a[i]
a[i] = a[change]
a[change] = tmp
End

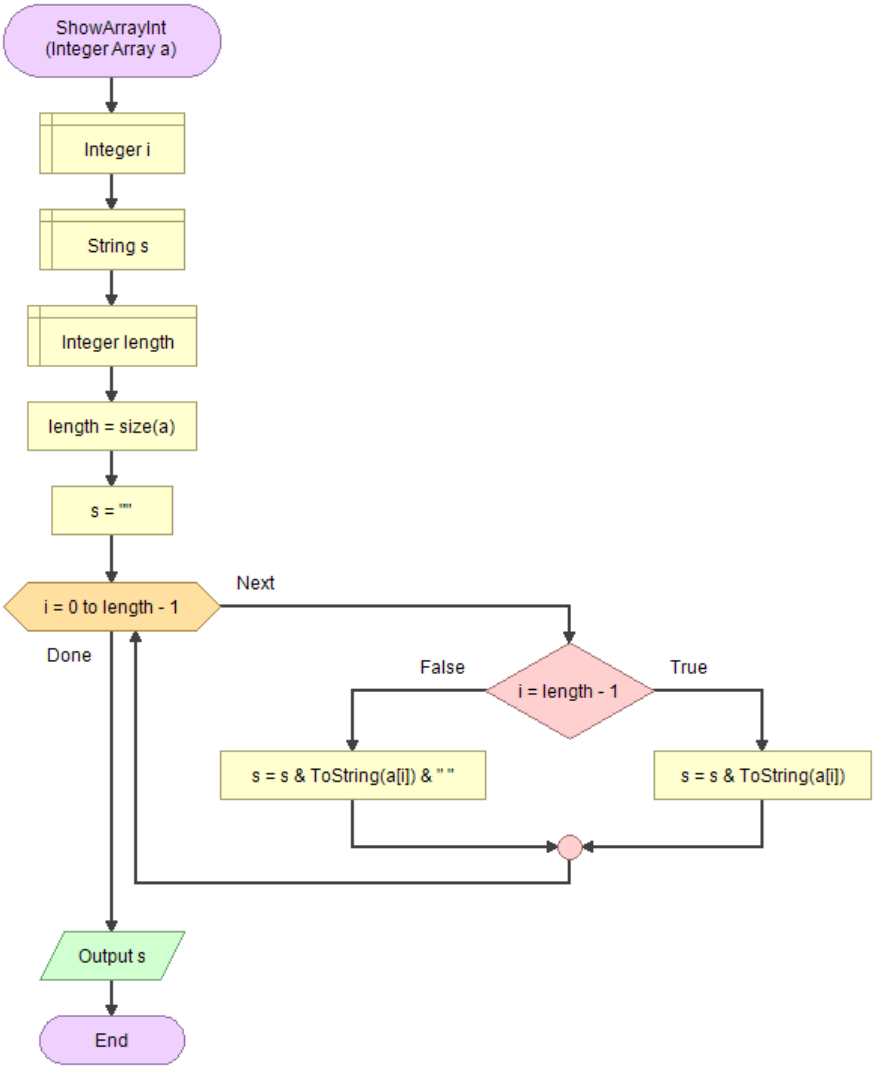

ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
i = length - 1
False
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
Return String s

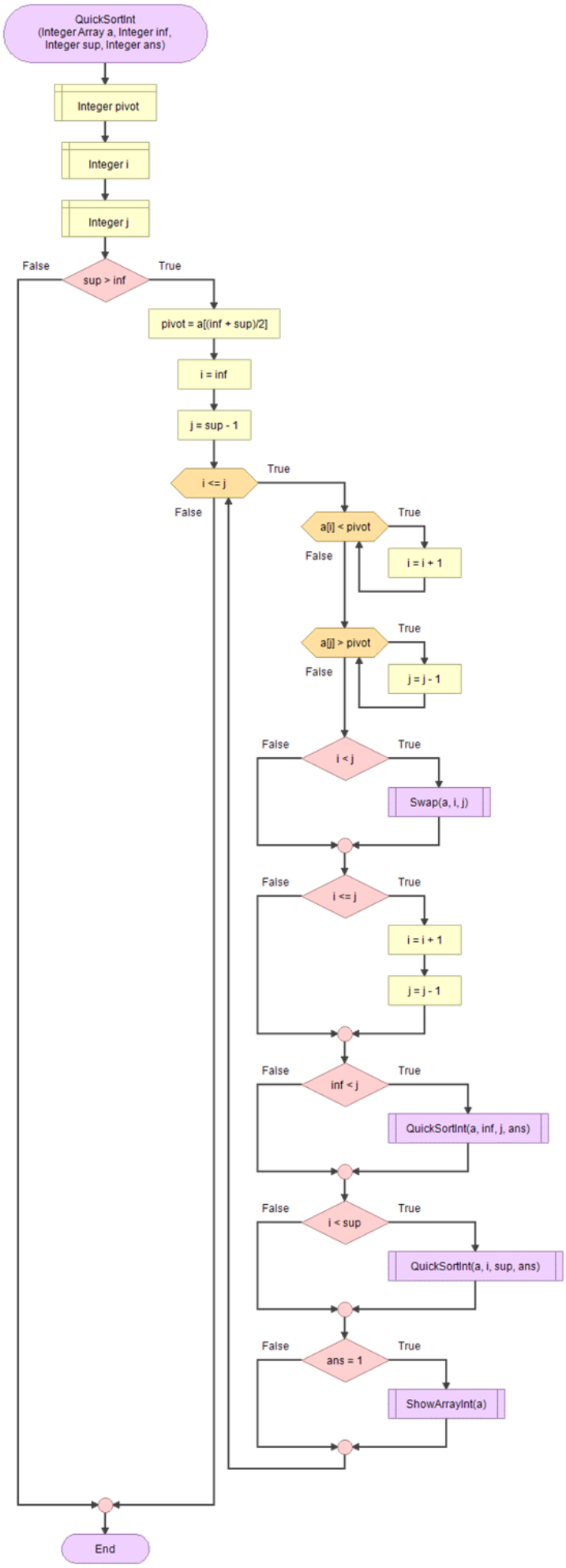

siftDown
(Integer Array a, Integer start, Integer end)
Integer root, child, tmp
Boolean exitflag
end represents the limit of how far down the heap to sift
root = start
exitflag = False
While the root has at least one child
(root * 2 + 1) <= end AND exitflag = False
True
False
root*2+1 points to the left child
child = root * 2 + 1
if overbound...
child + 1 > end
False
True
child = child - 1
if the child has a sibling and the child's value is less than its sibling's...
child + 1 <= end AND a [child] < a [child + 1]
False
True
... then point to the right child instead
child = child + 1
out of max-heap order
a[root] < a [child]
False
True
exitflag = True
SwapArrayInt(a, root, child)
repeat to continue sifting down the child now
root = child
End

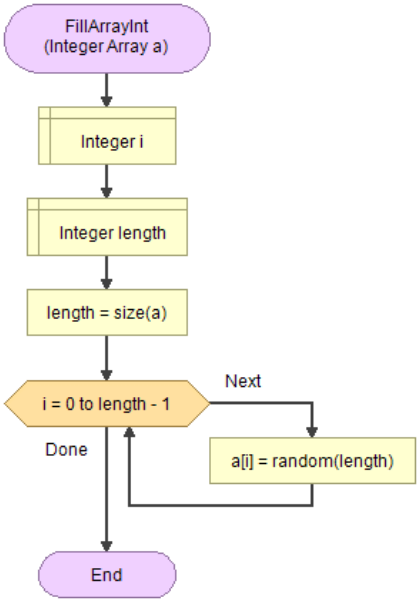
281.GAFR | Ordinamento Quick Sort (ricorsivo)

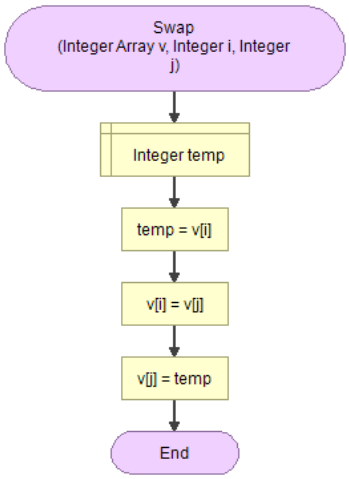
Quick Sort (recursive)





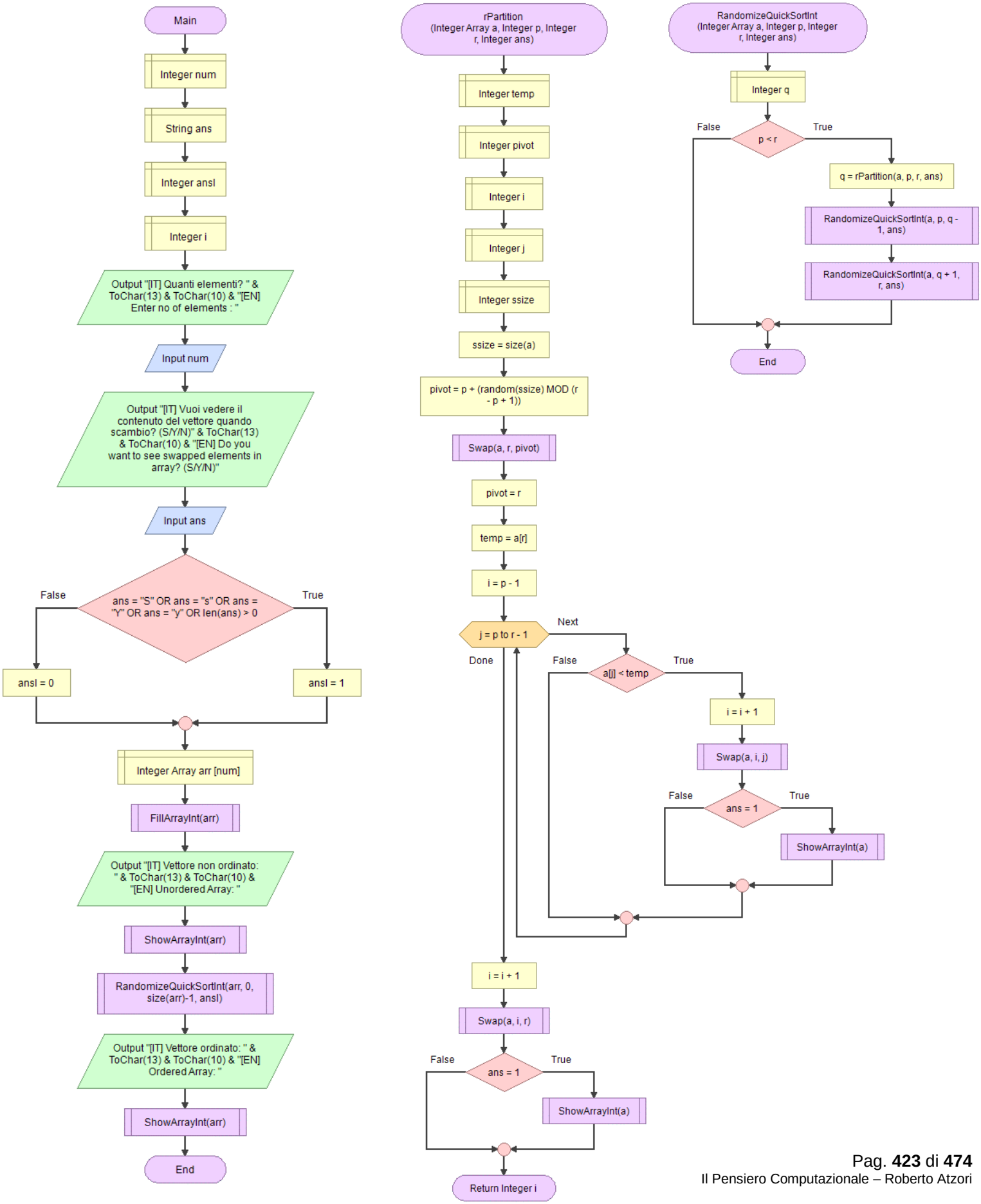

QuickSortInt
(Integer Array a, Integer inf, Integer sup, Integer ans)
Integer pivot
Integer i
Integer j
False
sup > inf
True
pivot = a[(inf + sup)/2]
i = inf
j = sup - 1
i <= j
True
False
a[i] < pivot
True
False
i = i + 1
a[j] > pivot
True
False
j = j - 1
False
i < j
True
Swap(a, i, j)
False
i <= j
True
i = i + 1
j = j - 1
False
inf < j
True
QuickSortInt(a, inf, j, ans)
False
i < sup
True
QuickSortInt(a, i, sup, ans)
False
ans = 1
True
ShowArrayInt(a)
End

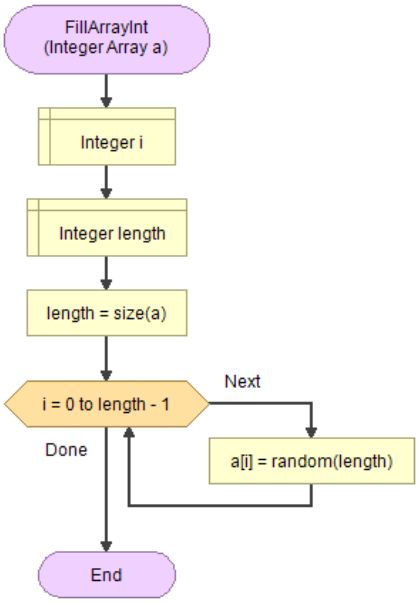

FillArrayInt
(Integer Array a)
Integer i
Integer length
length = size(a)
i = 0 to length - 1
Next
Done
a[i] = random(length)
End

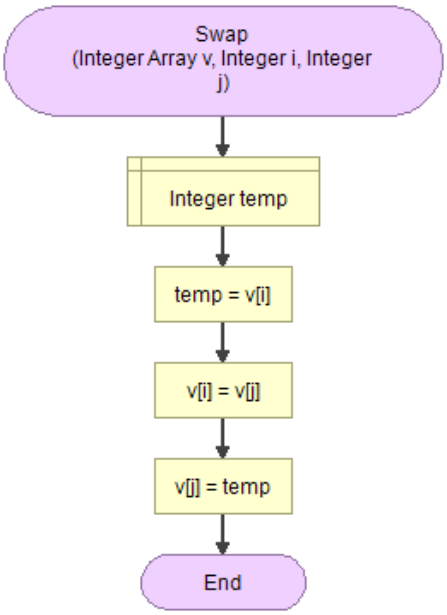

Swap
(Integer Array v, Integer i, Integer j)
Integer temp
temp = v[i]
v[i] = v[j]
v[j] = temp
End

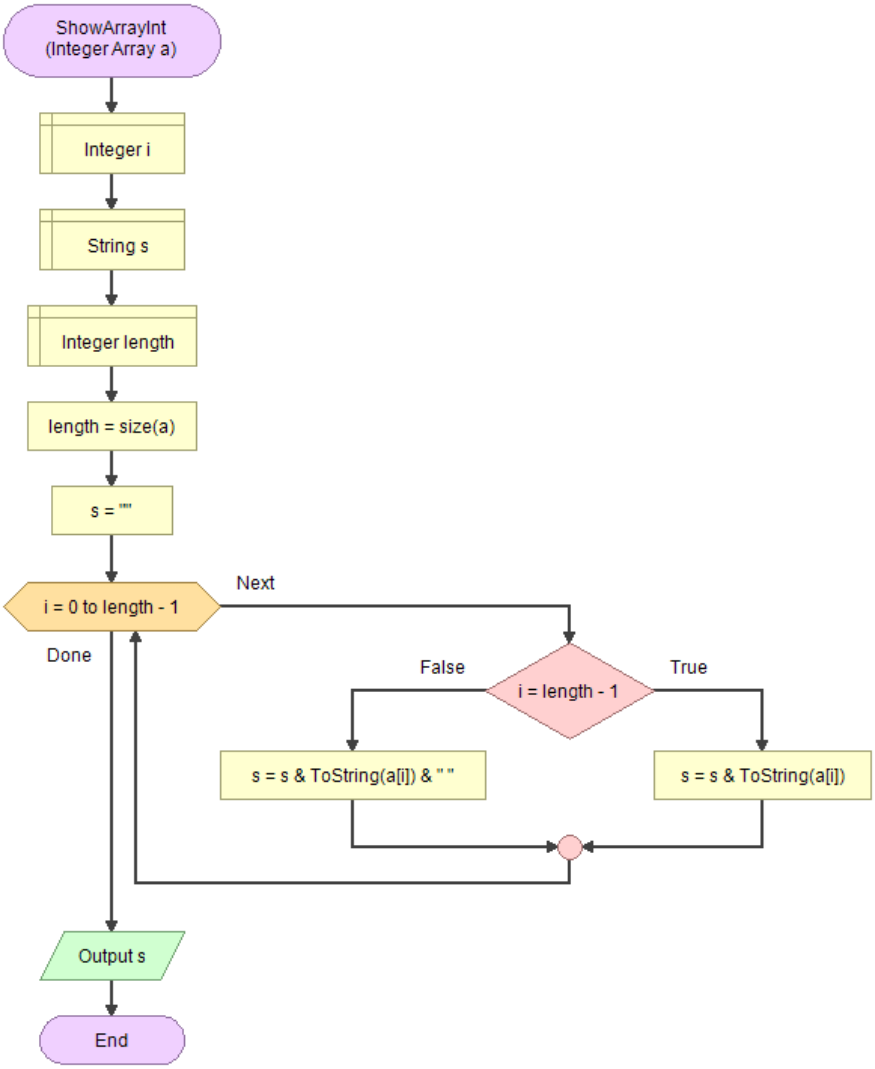
282.GAFR | Ordinamento Randomize Quick Sort (ricorsivo)

Randomize Quick Sort (recursive)



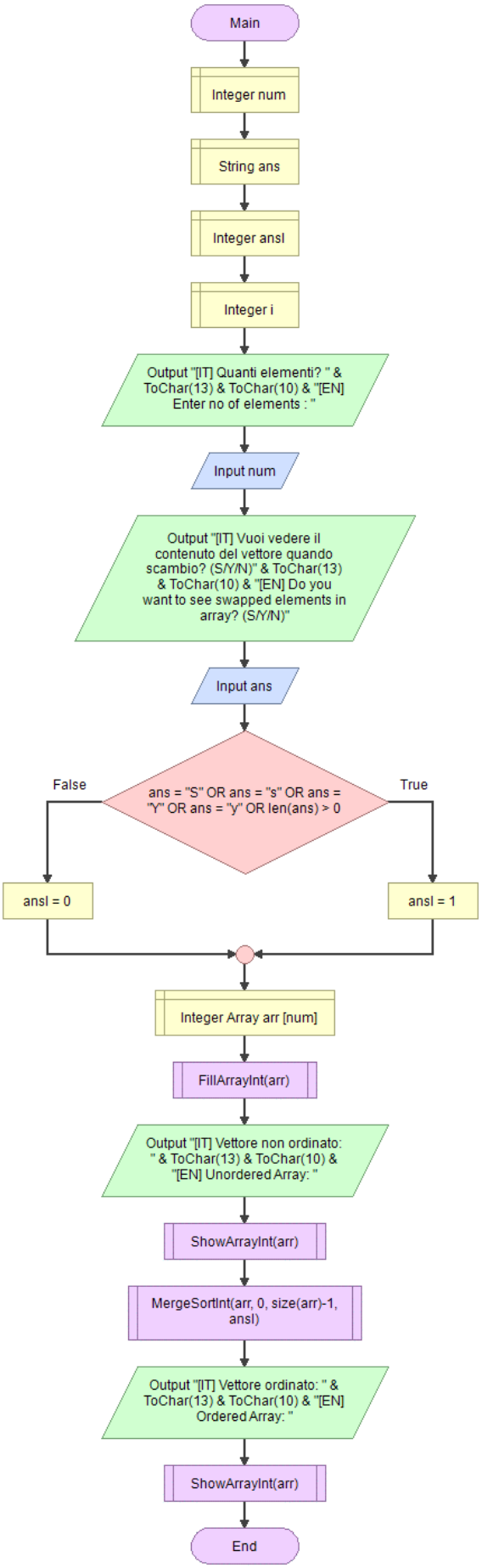

FillArrayInt
(Integer Array a)
Integer i
Integer length
length = size(a)
i = 0 to length - 1
Next
Done
a[i] = random(length)
End

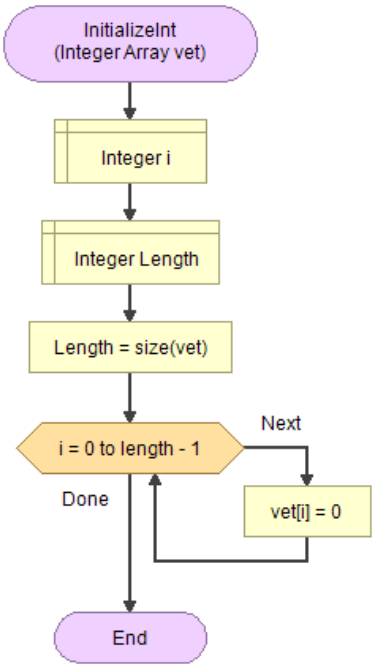

Swap
(Integer Array v, Integer i, Integer j)
Integer temp
temp = v[i]
v[i] = v[j]
v[j] = temp
End

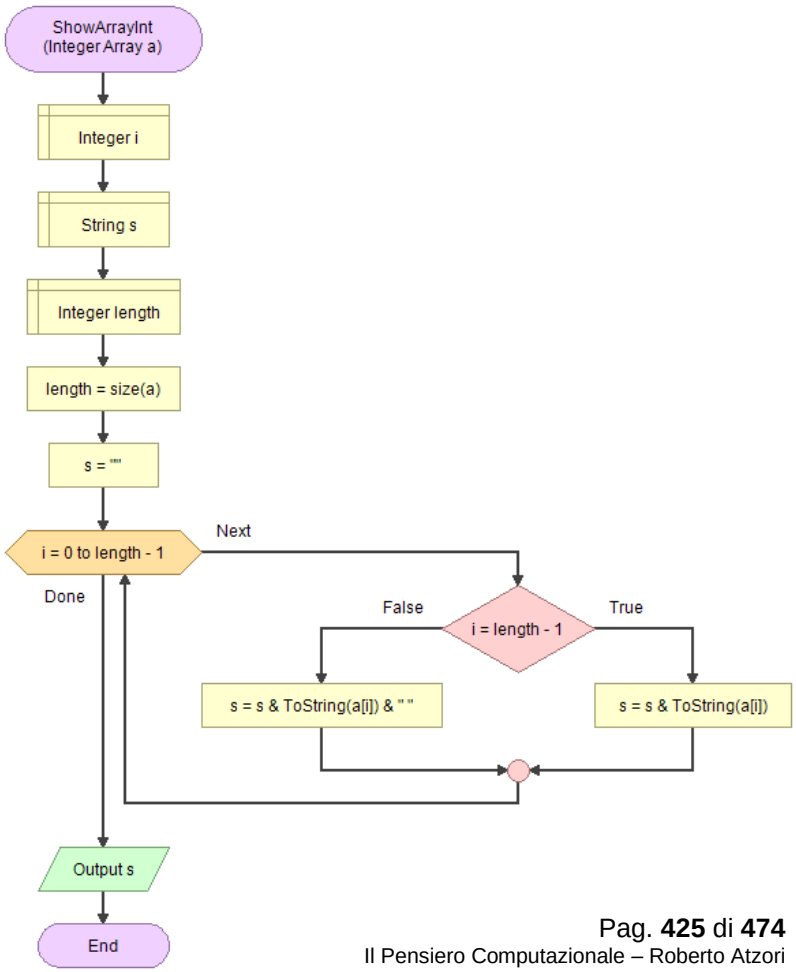

ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
False
i = length - 1
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
End

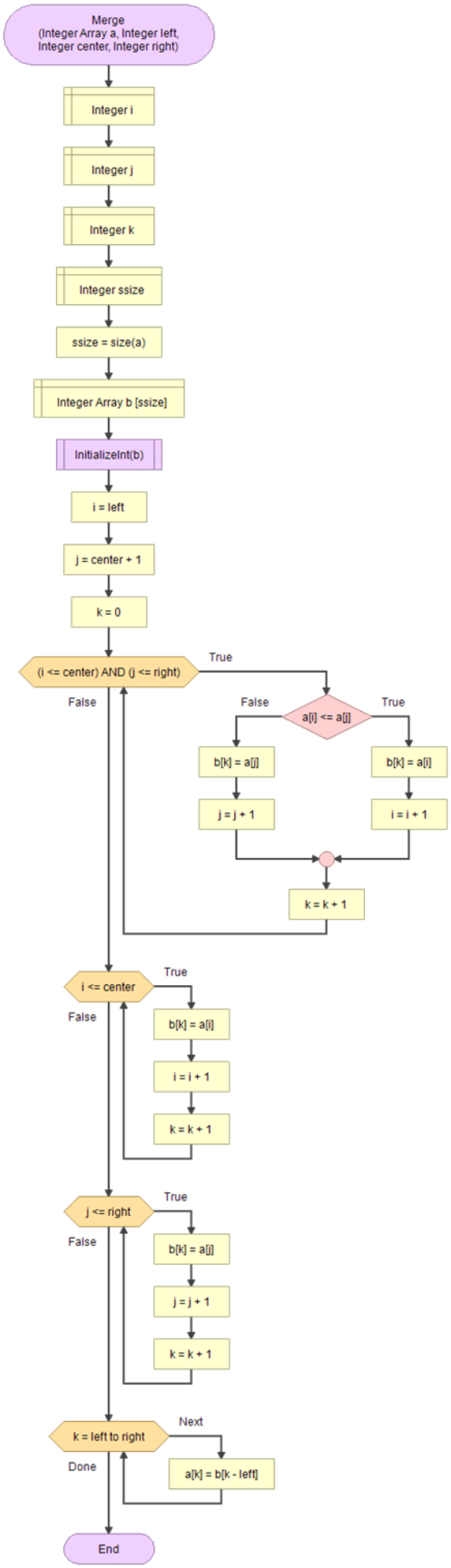
283.GAFR | Ordinamento Merge Sort (ricorsivo)

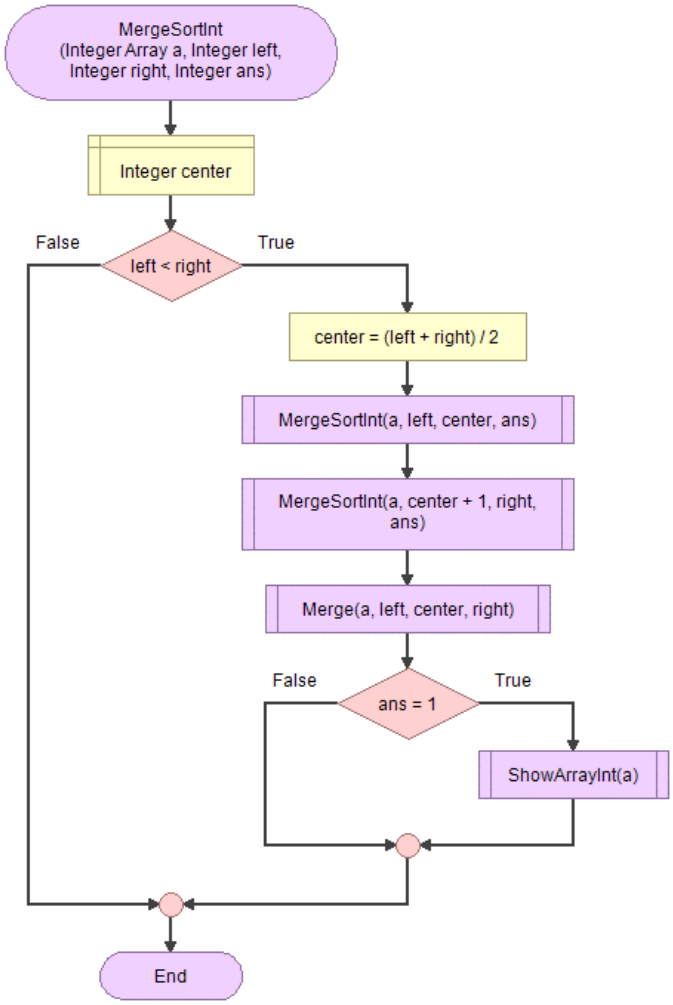
Merge Sort (recursive)

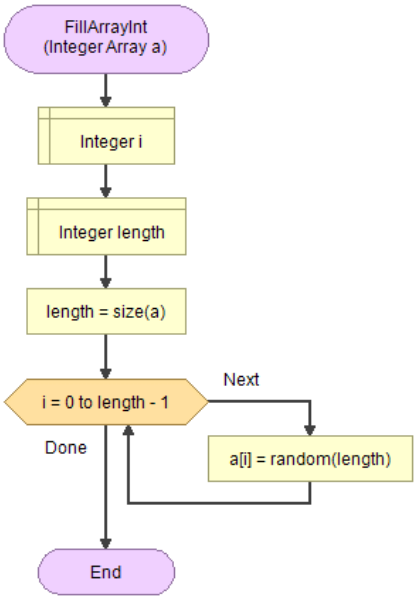






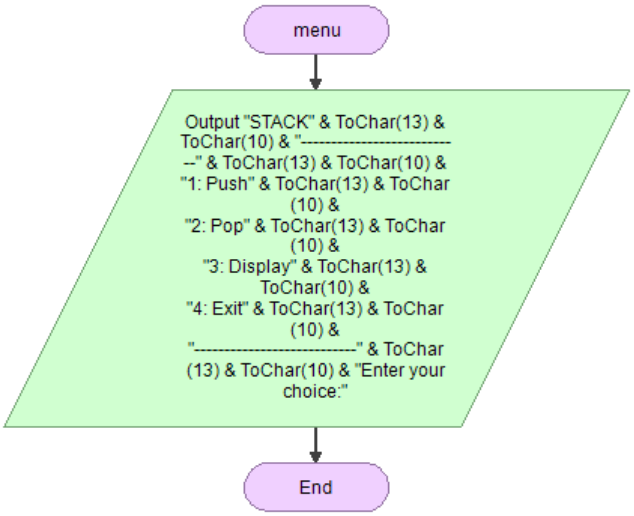

Merge
(Integer Array a, Integer left,
Integer center, Integer right)
Integer i
Integer j
Integer k
Integer ssize
ssize = size(a)
Integer Array b [ssize]
InitializeInt(b)
i = left
j = center + 1
k = 0
(i <= center) AND (j <= right)
True
False
a[i] <= a[j]
False
True
b[k] = a[j]
j = j + 1
b[k] = a[i]
i = i + 1
k = k + 1
i <= center
True
False
b[k] = a[i]
i = i + 1
k = k + 1
j <= right
True
False
b[k] = a[j]
j = j + 1
k = k + 1
k = left to right
Next
Done
a[k] = b[k - left]
End

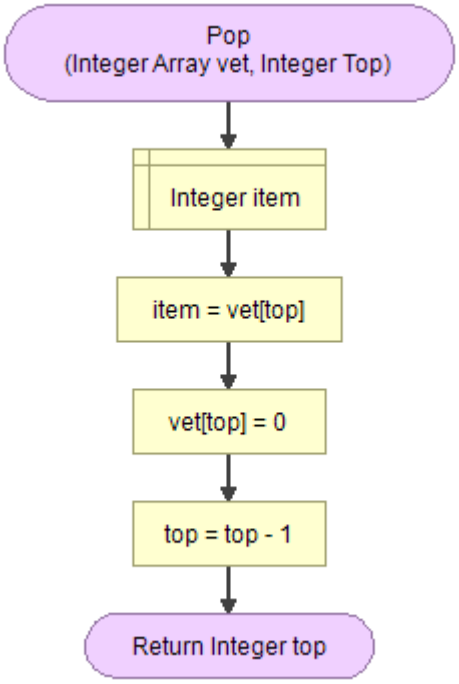

MergeSortInt
(Integer Array a, Integer left,
Integer right, Integer ans)
Integer center
False
left < right
True
center = (left + right) / 2
MergeSortInt(a, left, center, ans)
MergeSortInt(a, center + 1, right,
ans)
Merge(a, left, center, right)
False
ans = 1
True
ShowArrayInt(a)
End

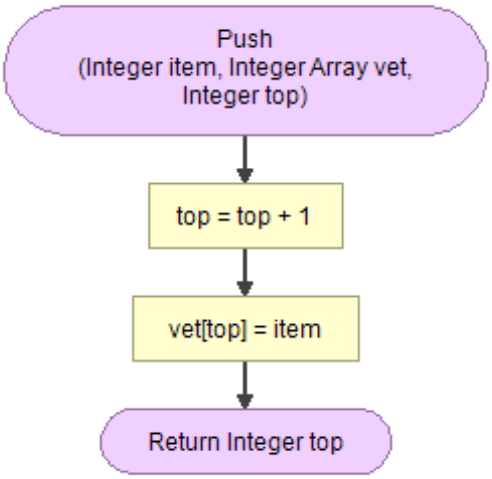

FillArrayInt
(Integer Array a)
Integer i
Integer length
length = size(a)
i = 0 to length - 1
Next
Done
a[i] = random(length)
End

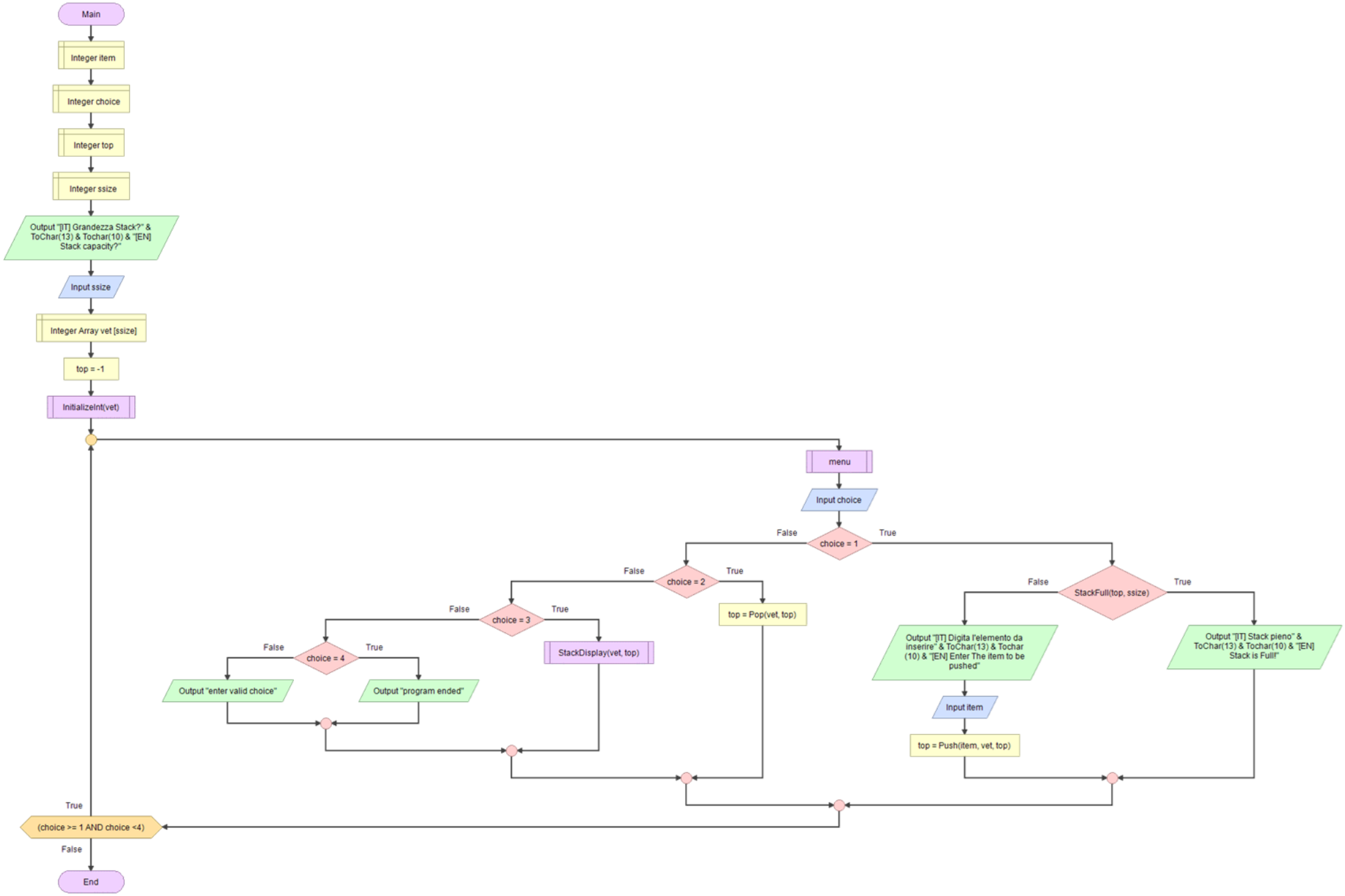
284.GAF | Stack con array

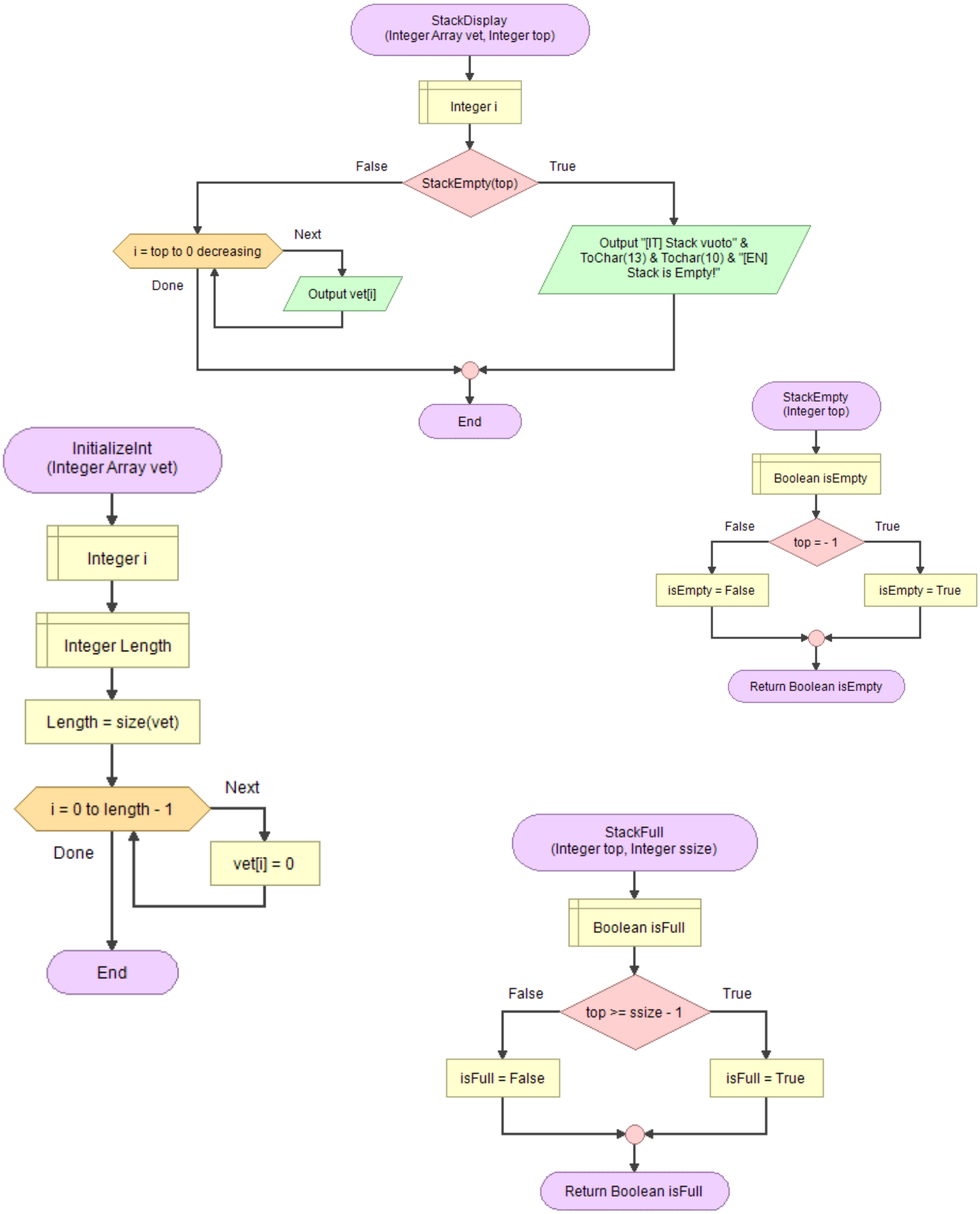
Stack





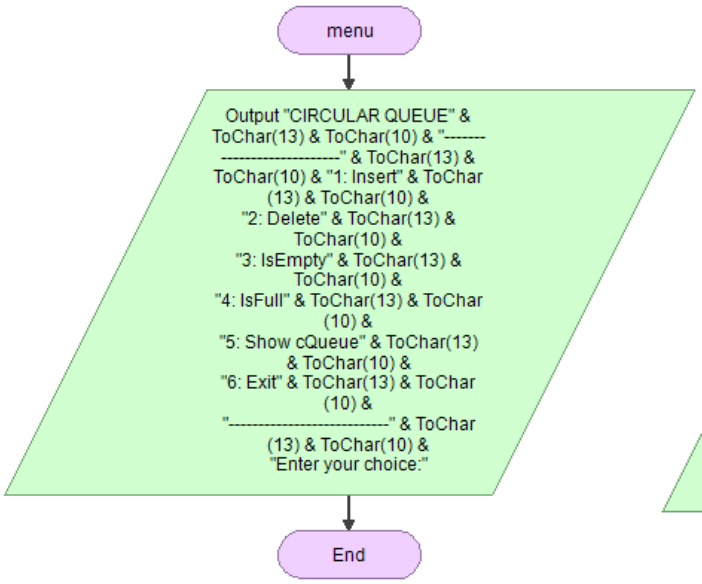


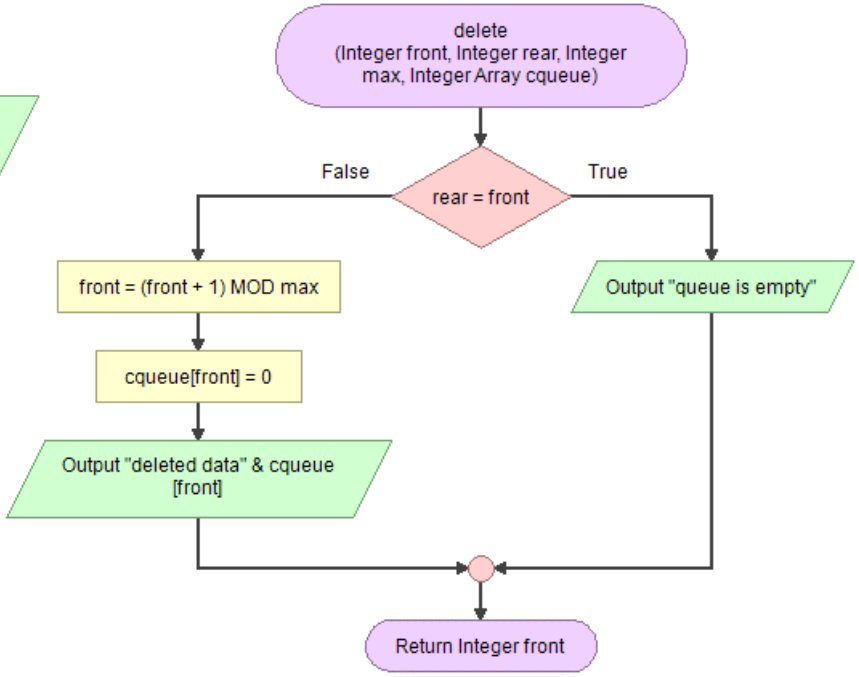

Main
Integer item
Integer choice
Integer top
Integer ssize
Output "[IT] Grandezza Stack?" & ToChar(13) & Tochar(10) & "[EN] Stack capacity?"
Input ssize
Integer Array vet [ssize]
top = -1
InitializeInt(vet)
menu
Input choice
choice = 1
False
True
choice = 2
False
True
top = Pop(vet, top)
choice = 3
False
True
StackDisplay(vet, top)
choice = 4
False
True
Output "enter valid choice"
Output "program ended"
StackFull(top, ssize)
False
True
Output "[IT] Digita l'elemento da inserire" & ToChar(13) & Tochar (10) & "[EN] Enter The item to be pushed"
Input item
top = Push(item, vet, top)
Output "[IT] Stack pieno" & ToChar(13) & Tochar(10) & "[EN] Stack is Full!"
True
(choice >= 1 AND choice <4)
False
End

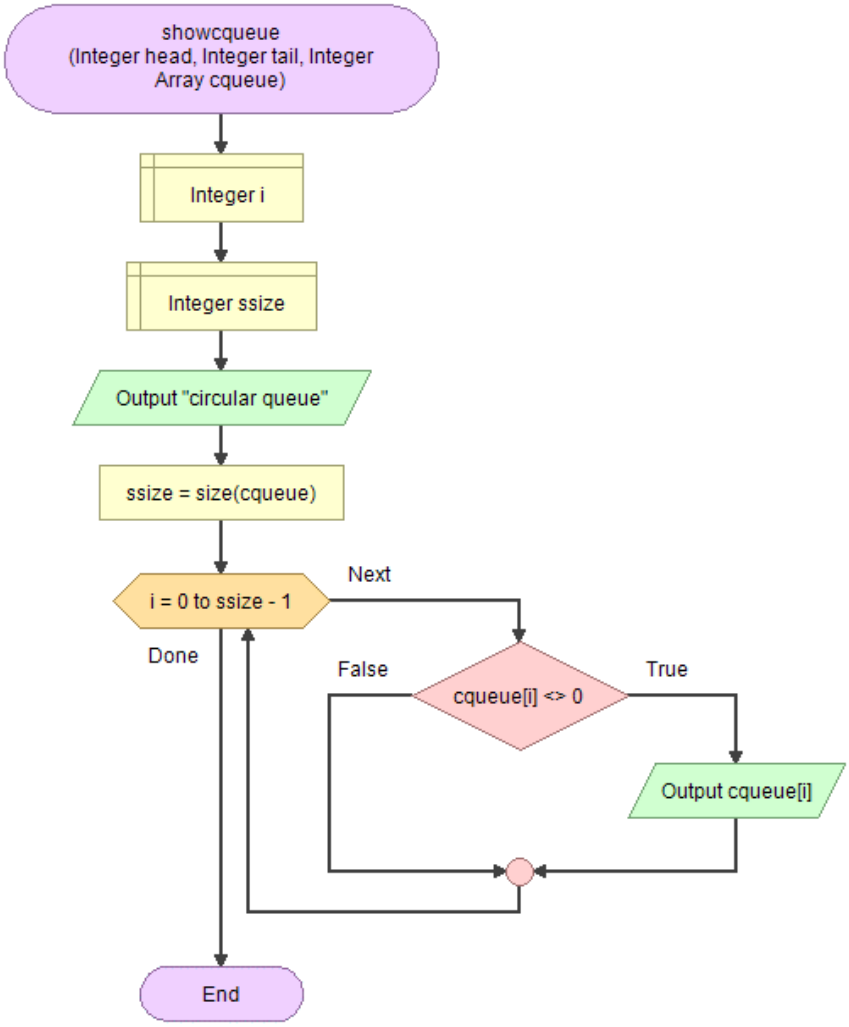

StackDisplay
(Integer Array vet, Integer top)
Integer i
False
StackEmpty(top)
True
i = top to 0 decreasing
Next
Done
Output vet[i]
Output "[IT] Stack vuoto" &
ToChar(13) & Tochar(10) & "[EN]
Stack is Empty!"
End
StackEmpty
(Integer top)
Boolean isEmpty
False
top = - 1
True
isEmpty = False
isEmpty = True
Return Boolean isEmpty
InitializeInt
(Integer Array vet)
Integer i
Integer Length
Length = size(vet)
i = 0 to length - 1
Next
Done
vet[i] = 0
End
StackFull
(Integer top, Integer ssize)
Boolean isFull
False
top >= ssize - 1
True
isFull = False
isFull = True
Return Boolean isFull

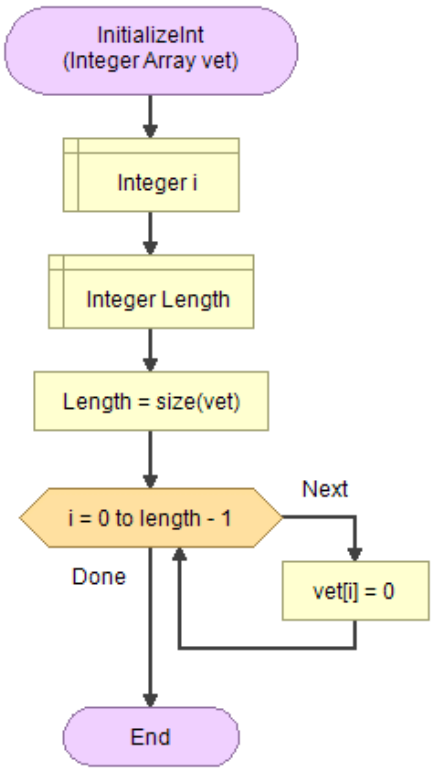
285.GAF ‖ Coda circolare con array

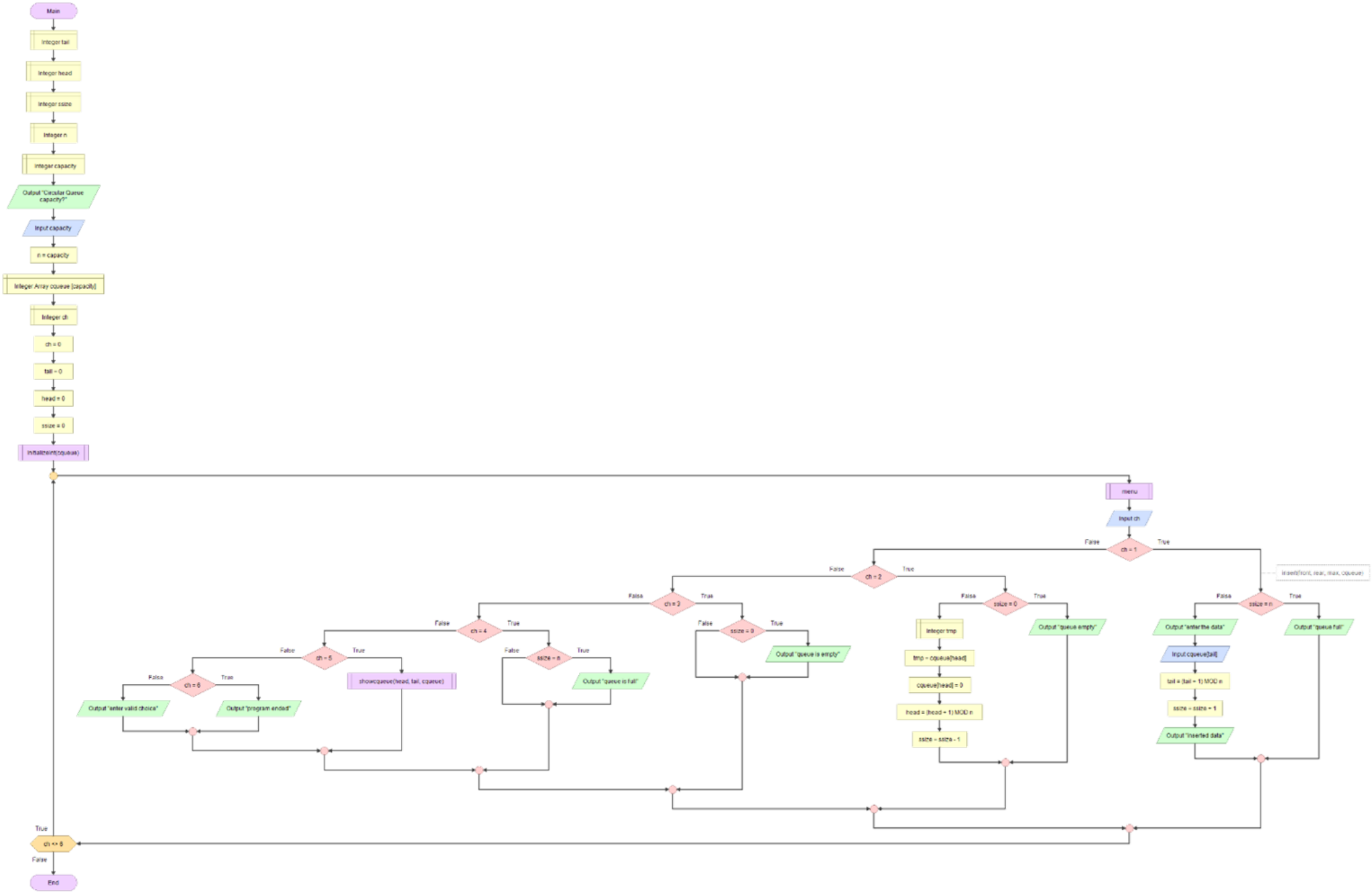
Circular Queue

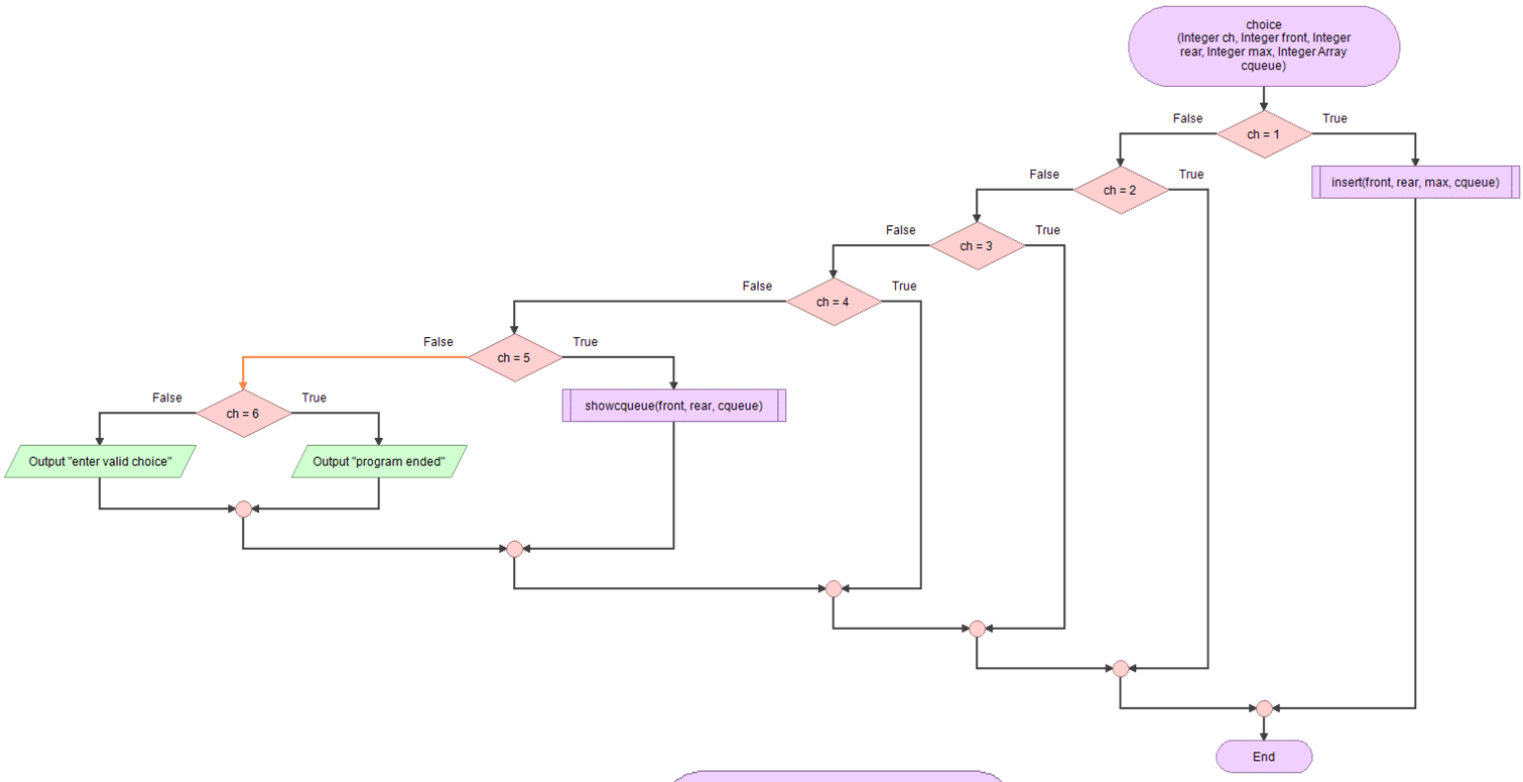


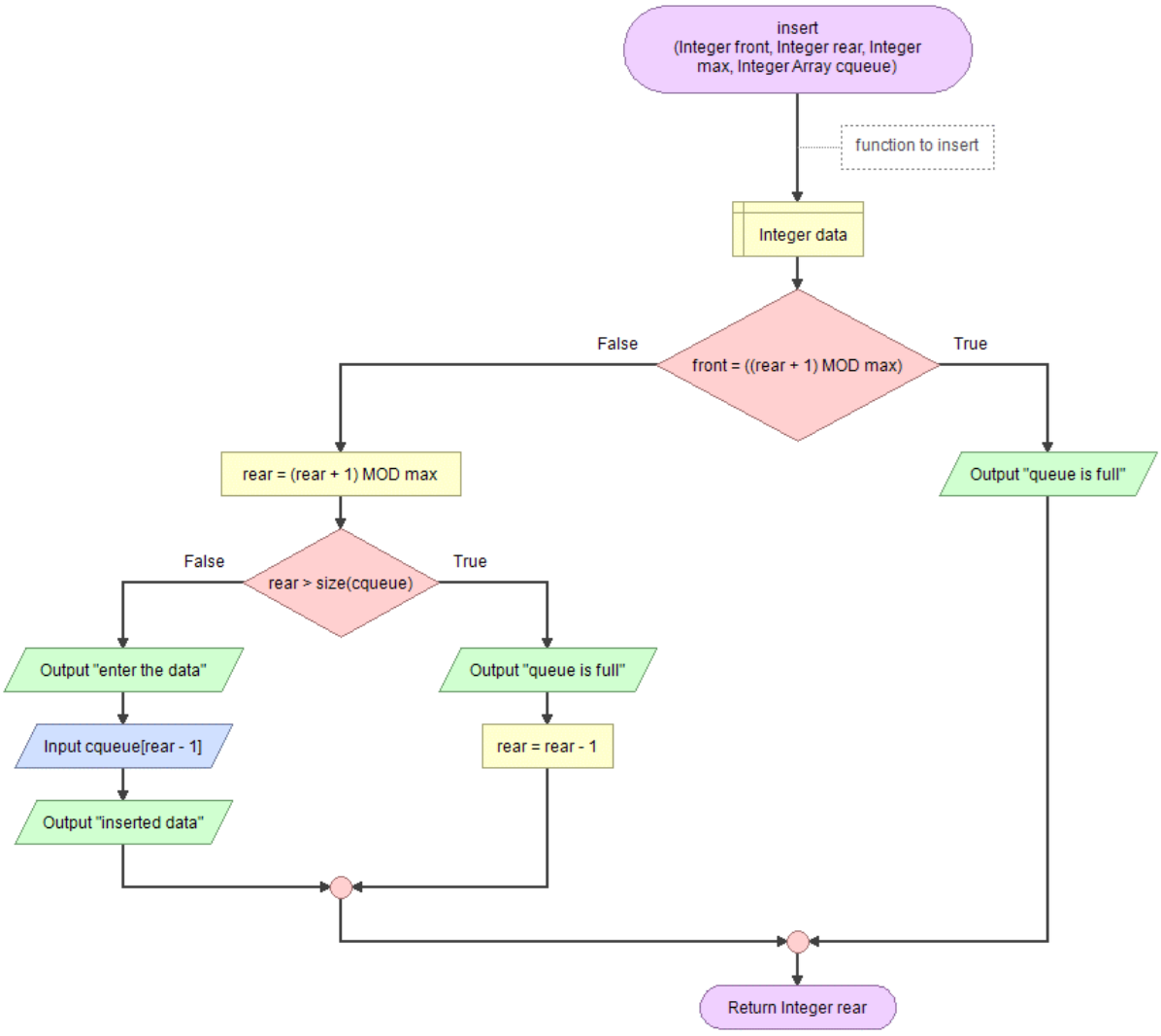







Main
Integer tail
Integer head
Integer ssize
Integer n
Integer capacity
Output "Circular Queue capacity?"
Input capacity
n = capacity
Integer Array cqueue [capacity]
Integer ch
ch = 0
tail = 0
head = 0
ssize = 0
initialize(cqueue)
menu
Input ch
ch = 1
False
True
ch = 2
ssize = 0
Integer tmp
tmp = cqueue[head]
cqueue[head] = 0
head = (head + 1) MOD n
ssize = ssize - 1
Output "queue empty"
ssize = n
Output "enter the data"
Input cqueue[tail]
tail = (tail + 1) MOD n
ssize = ssize + 1
Output "inserted data"
Output "queue full"
ch = 3
ssize = 0
Output "queue is empty"
ch = 4
ssize = n
Output "queue is full"
ch = 5
showcqueue(head, tail, cqueue)
ch = 6
Output "enter valid choice"
Output "program ended"
ch <> 6
End

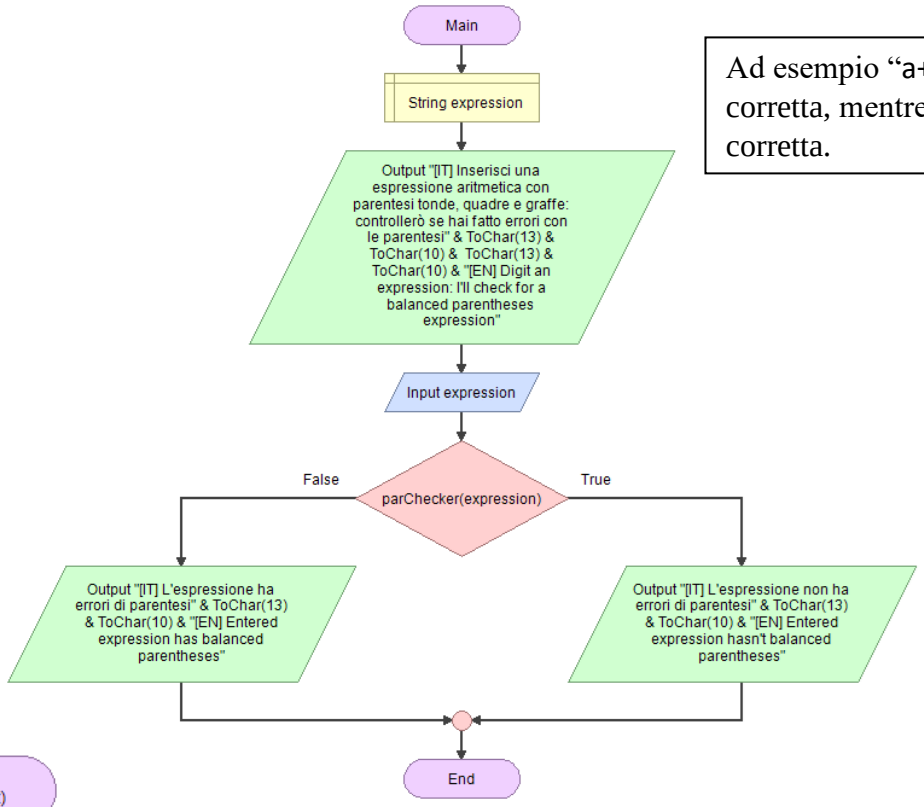

choice
(Integer ch, Integer front, Integer rear, Integer max, Integer Array cqueue)
ch = 1
False
True
insert(front, rear, max, cqueue)
ch = 2
False
True
ch = 3
False
True
ch = 4
False
True
ch = 5
False
True
showcqueue(front, rear, cqueue)
ch = 6
False
True
Output "enter valid choice"
Output "program ended"
End

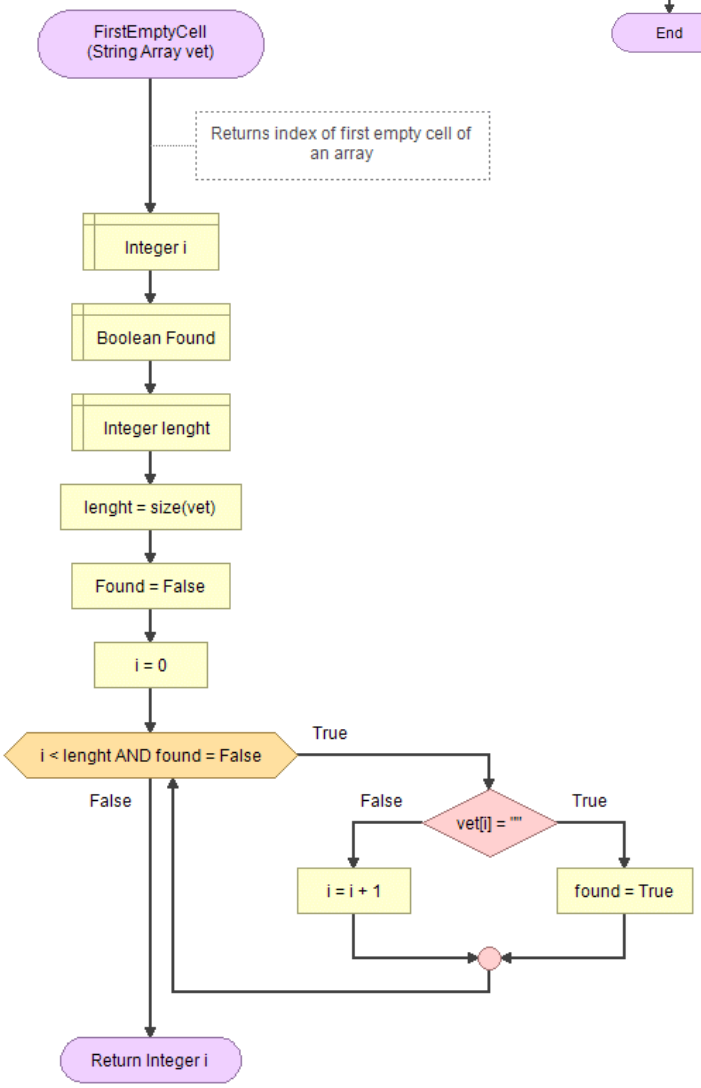

insert
(Integer front, Integer rear, Integer max, Integer Array cqueue)
function to insert
Integer data
front = ((rear + 1) MOD max)
False
True
rear = (rear + 1) MOD max
Output "queue is full"
rear > size(cqueue)
False
True
Output "enter the data"
Output "queue is full"
Input cqueue[rear - 1]
rear = rear - 1
Output "inserted data"
Return Integer rear

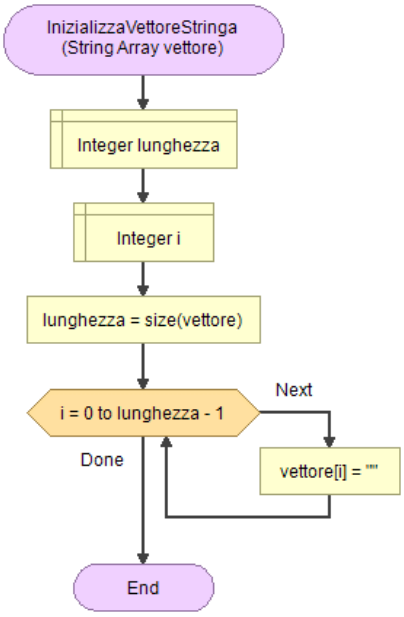
286. **G A F** ‖ Letta in input una stringa che rappresenta un'espressione aritmetica, verificare che le parentesi siano collocate correttamente. Bilanciamento parentesi.

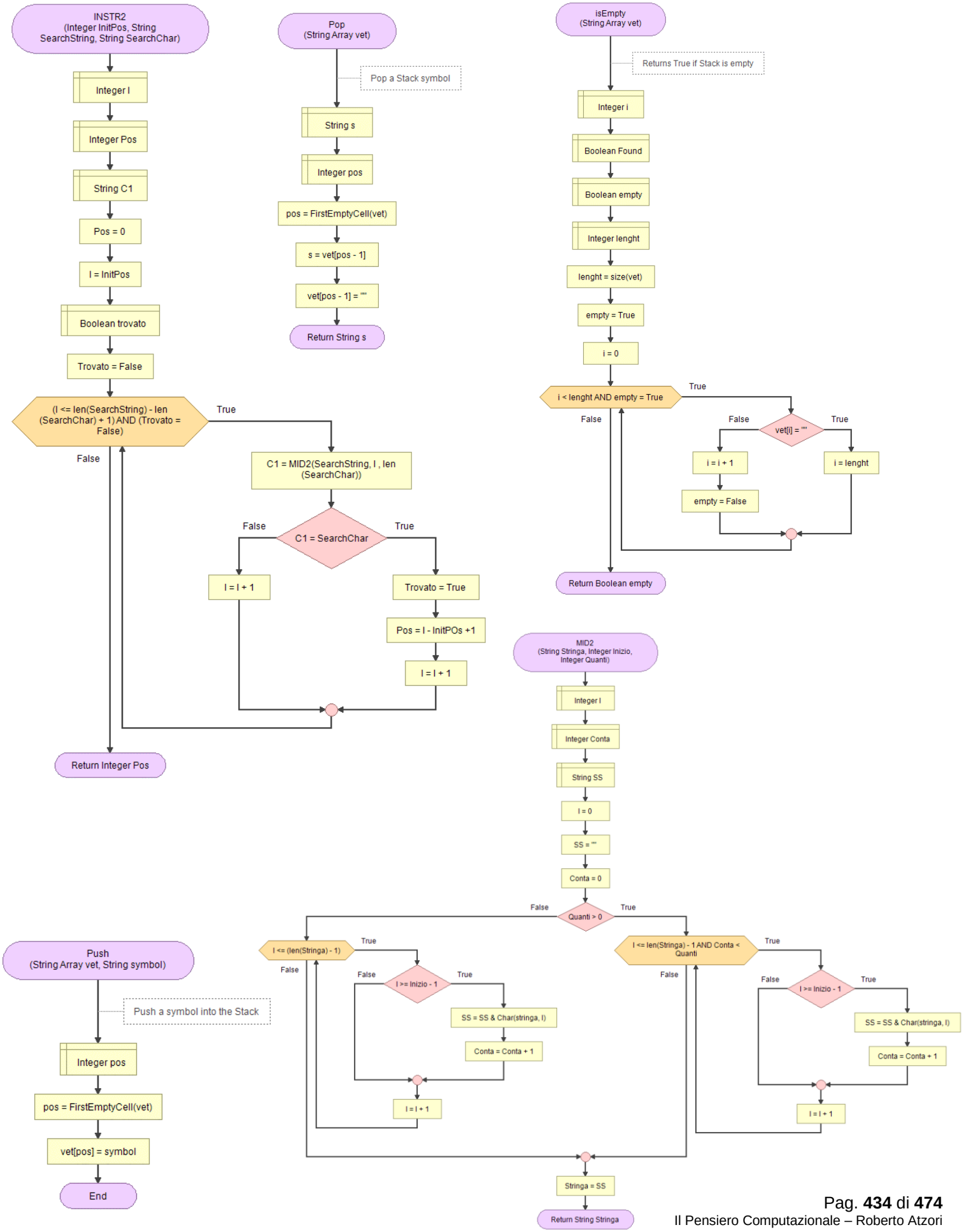
Balanced Parentheses

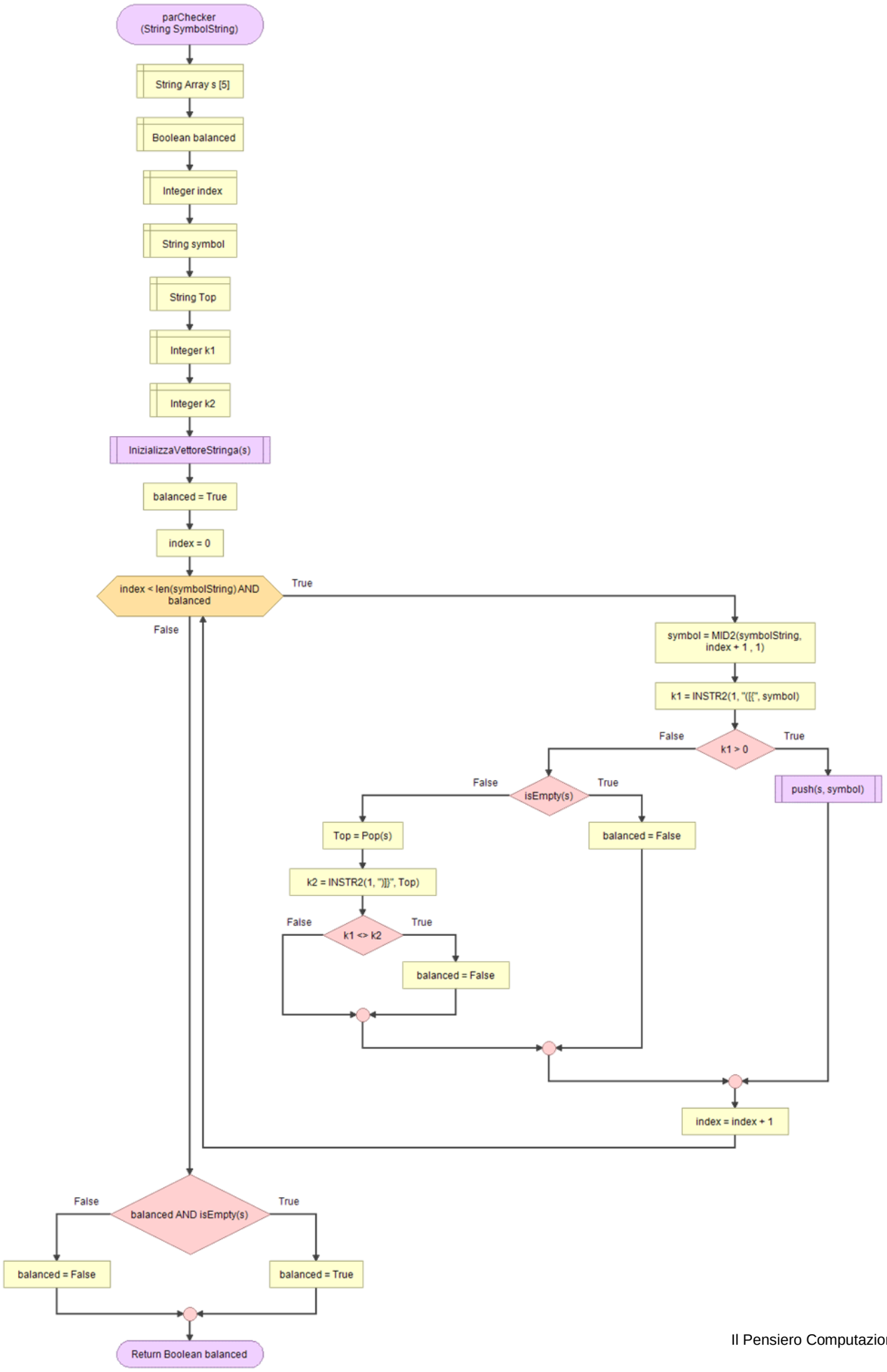
Ad esempio "a+b*(c-d)/(a+(b/2)+3/2)" è corretta, mentre "a+b)/(c+e)/(f+g))" non è corretta.



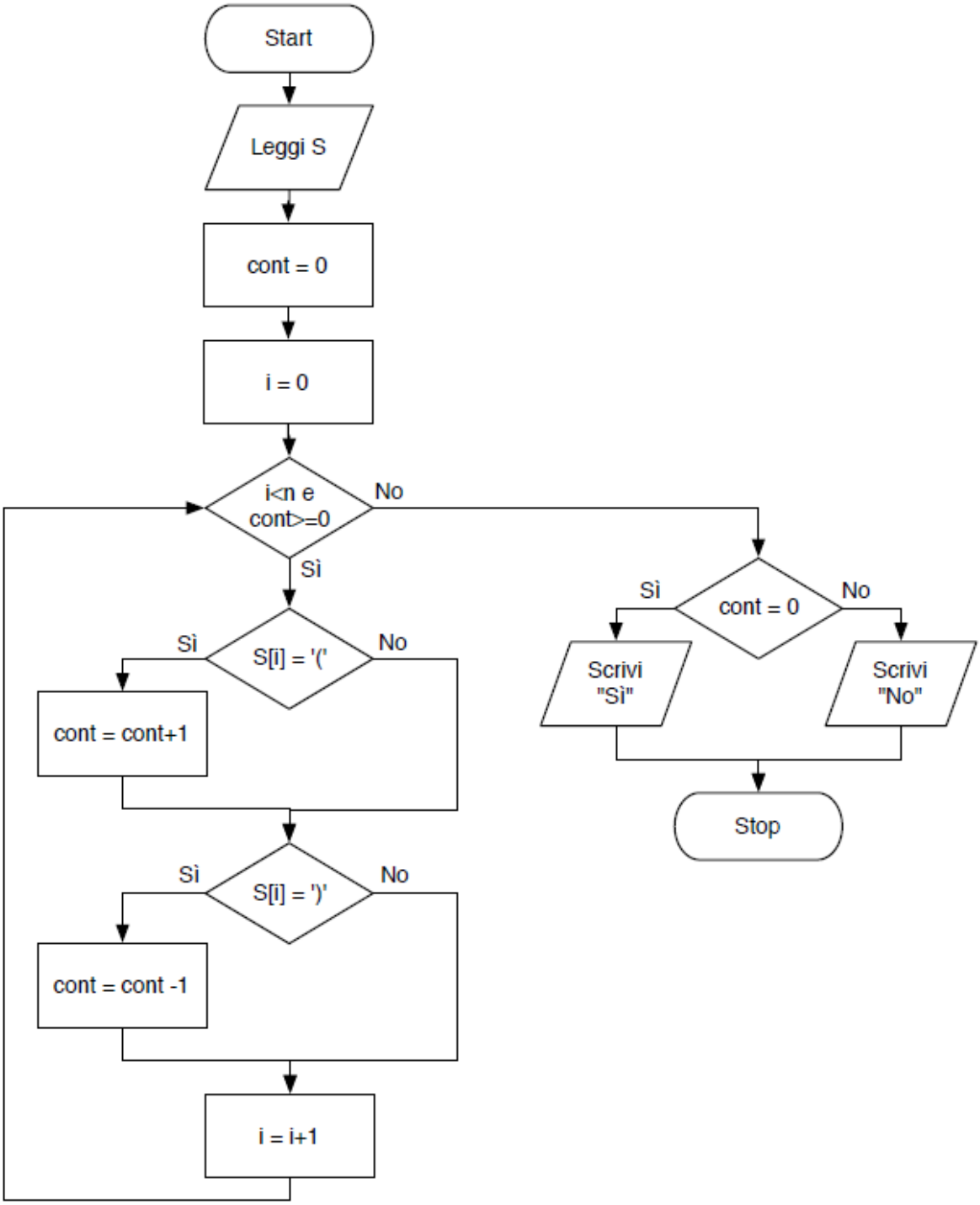





INSTR2
(Integer InitPos, String SearchString, String SearchChar)
Integer I
Integer Pos
String C1
Pos = 0
I = InitPos
Boolean trovato
Trovato = False
(I <= len(SearchString) - len (SearchChar) + 1) AND (Trovato = False)
True
False
C1 = MID2(SearchString, I , len (SearchChar))
C1 = SearchChar
False
True
I = I + 1
Trovato = True
Pos = I - InitPOs +1
I = I + 1
Return Integer Pos
Pop
(String Array vet)
Pop a Stack symbol
String s
Integer pos
pos = FirstEmptyCell(vet)
s = vet[pos - 1]
vet[pos - 1] = ""
Return String s
isEmpty
(String Array vet)
Returns True if Stack is empty
Integer i
Boolean Found
Boolean empty
Integer lenght
lenght = size(vet)
empty = True
i = 0
i < lenght AND empty = True
True
False
vet[i] = ""
False
True
i = i + 1
i = lenght
empty = False
Return Boolean empty
MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa
Push
(String Array vet, String symbol)
Push a symbol into the Stack
Integer pos
pos = FirstEmptyCell(vet)
vet[pos] = symbol
End

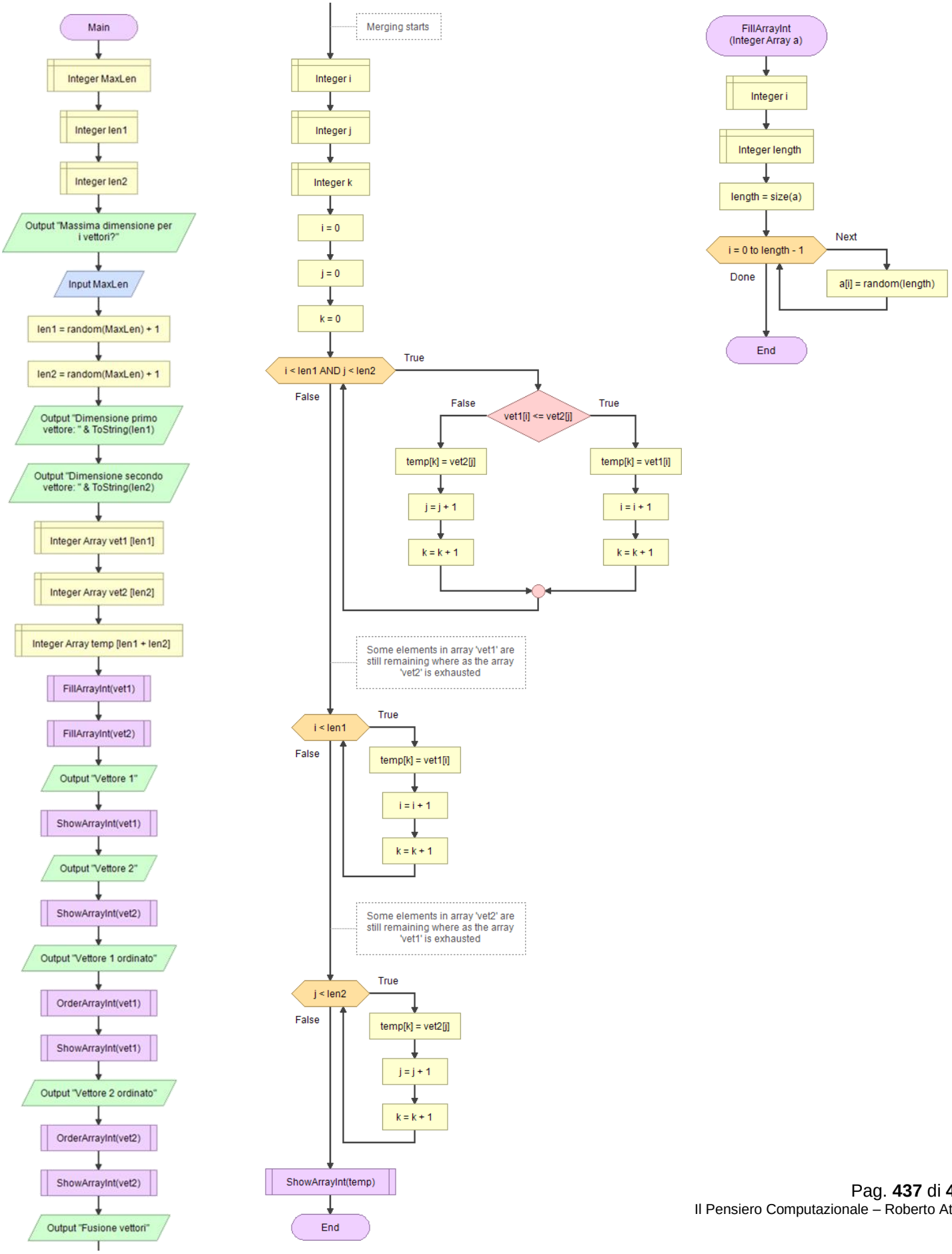

parChecker
(String SymbolString)
String Array s [5]
Boolean balanced
Integer index
String symbol
String Top
Integer k1
Integer k2
InizializzaVettoreStringa(s)
balanced = True
index = 0
index < len(symbolString) AND balanced
True
False
symbol = MID2(symbolString, index + 1 , 1)
k1 = INSTR2(1, "([{", symbol)
False
k1 > 0
True
push(s, symbol)
False
isEmpty(s)
True
Top = Pop(s)
balanced = False
k2 = INSTR2(1, ")]}", Top)
False
k1 <> k2
True
balanced = False
index = index + 1
False
balanced AND isEmpty(s)
True
balanced = False
balanced = True
Return Boolean balanced

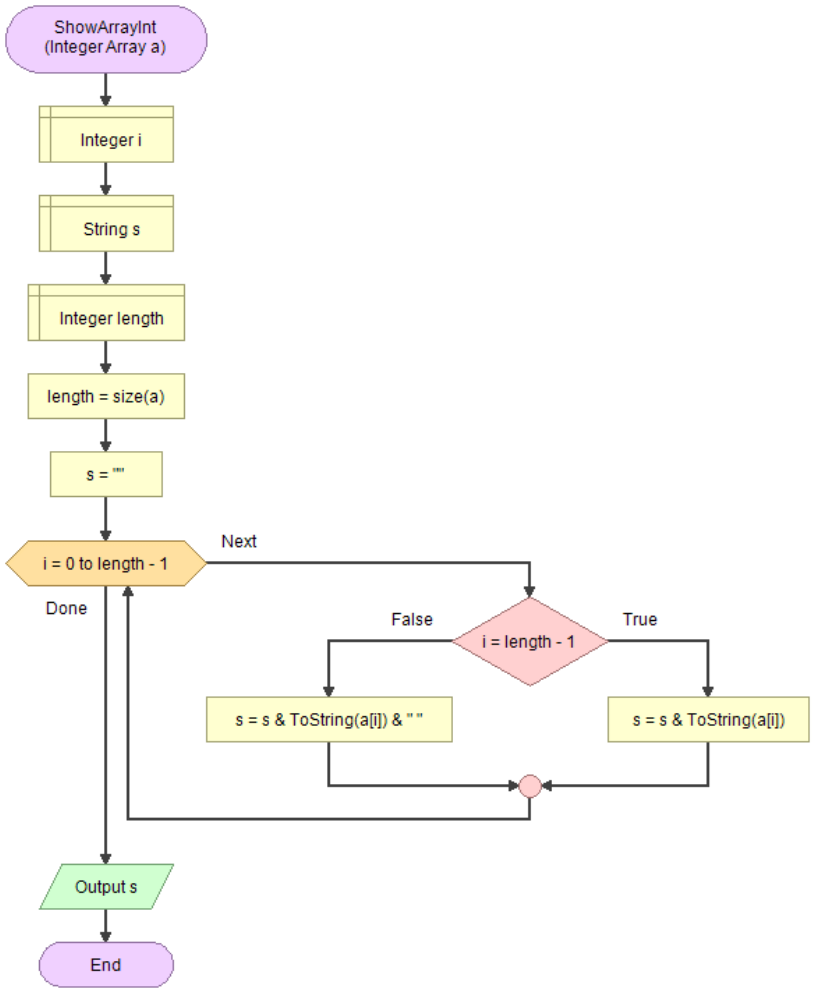
(esercizio solo con parentesi tonde)

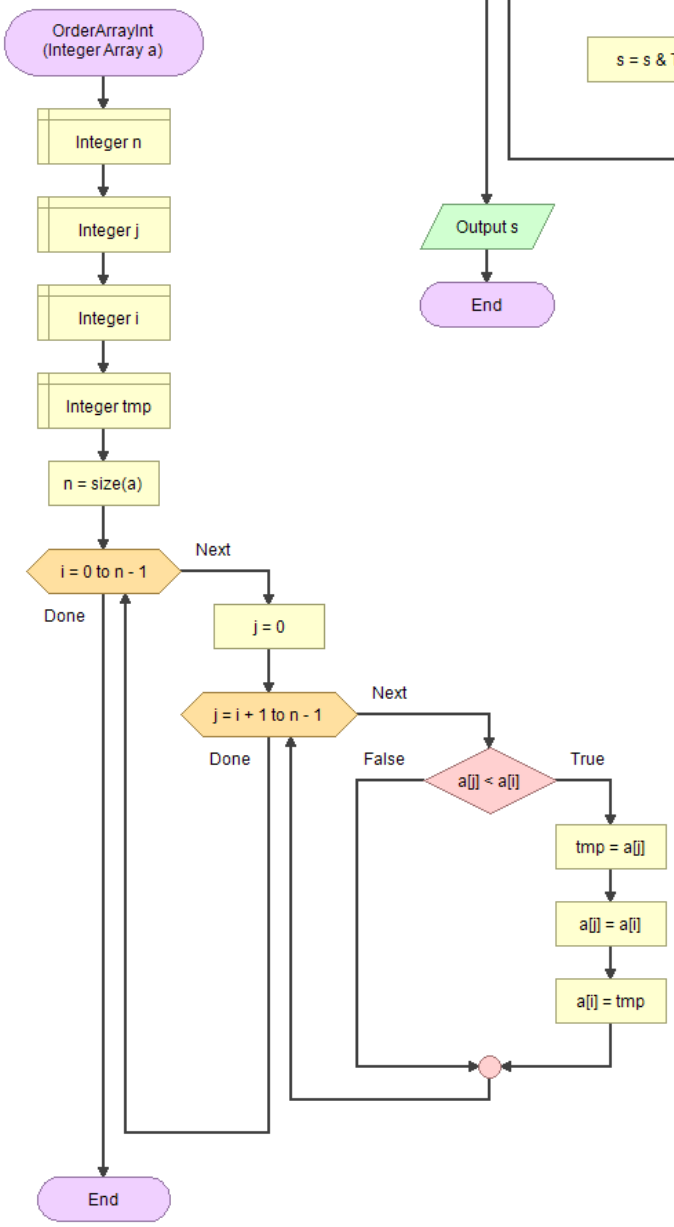


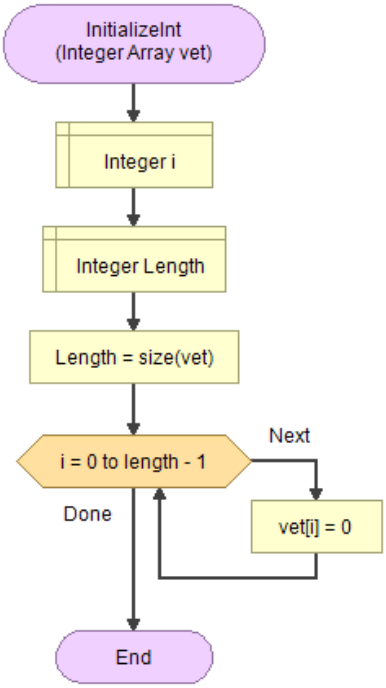
287.GAF ‖ Fusione tra due vettori di interi ordinati

Fusion of 2 integer ordered arrays



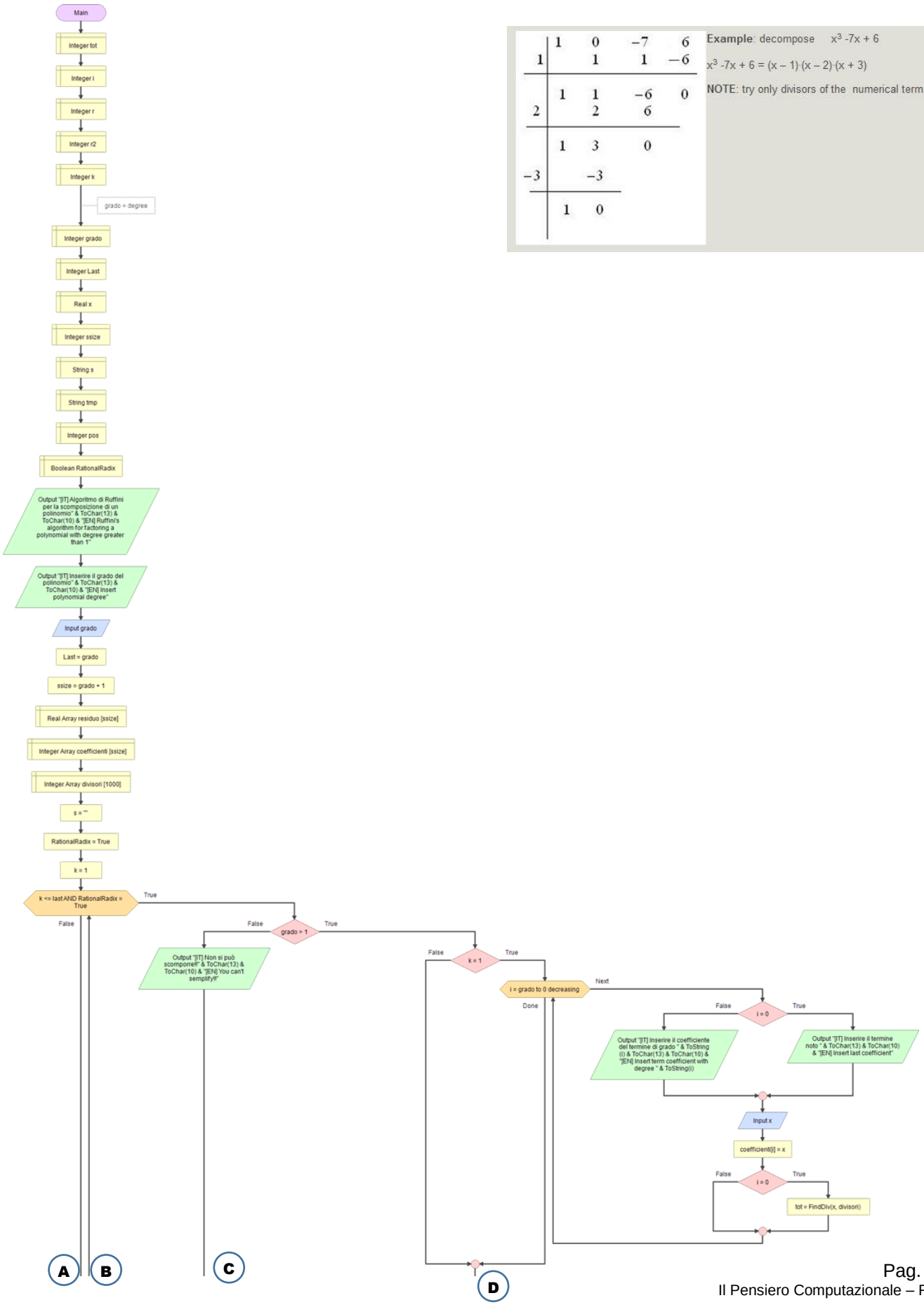

ShowArrayInt
(Integer Array a)
Integer i
String s
Integer length
length = size(a)
s = ""
i = 0 to length - 1
Next
Done
False
i = length - 1
True
s = s & ToString(a[i]) & " "
s = s & ToString(a[i])
Output s
End


OrderArrayInt
(Integer Array a)
Integer n
Integer j
Integer i
Integer tmp
n = size(a)
i = 0 to n - 1
Next
Done
j = 0
j = i + 1 to n - 1
Next
Done
False
a[j] < a[i]
True
tmp = a[j]
a[j] = a[i]
a[i] = tmp
End


InitializeInt
(Integer Array vet)
Integer i
Integer Length
Length = size(vet)
i = 0 to length - 1
Next
Done
vet[i] = 0
End

288. G A F ‖ Scomposizione di un polinomio con grado maggiore 1 con Ruffini

Ruffini's rule (polynomial factoring)

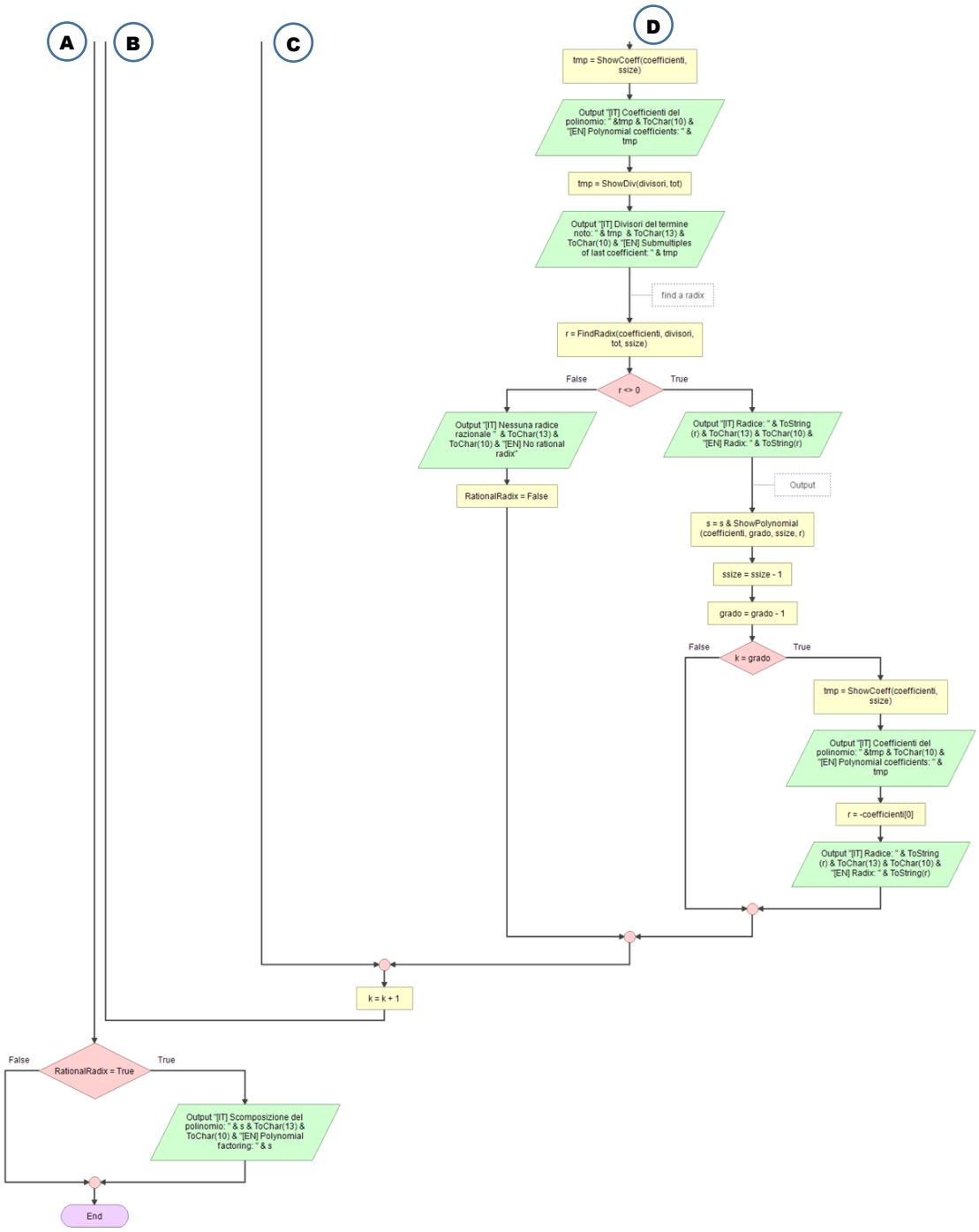


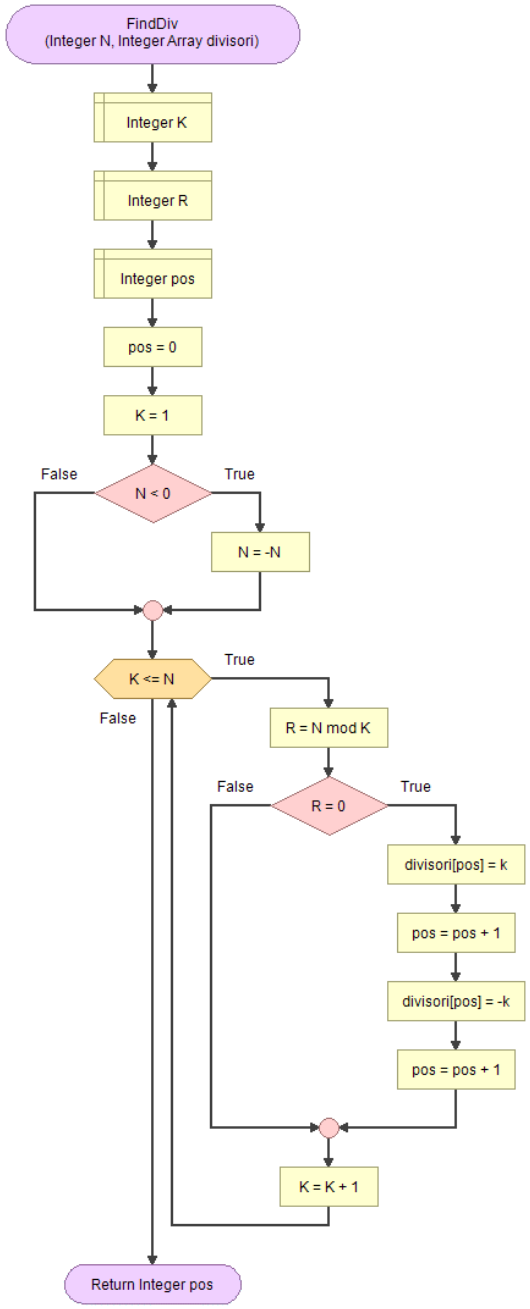
	1	0	−7	6
1		1	1	−6
	1	1	−6	0
2		2	6	
	1	3	0	
−3		−3		
	1	0		

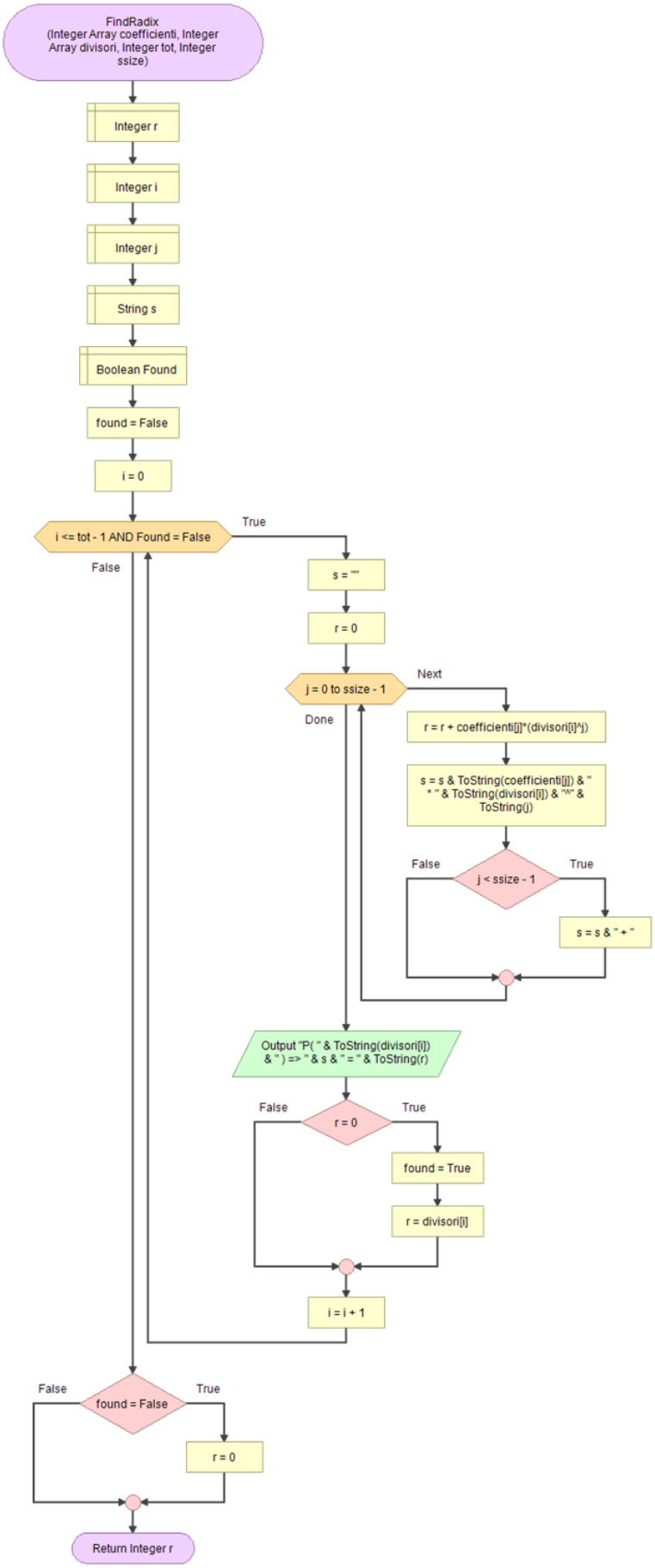
Example: decompose $x^3 - 7x + 6$

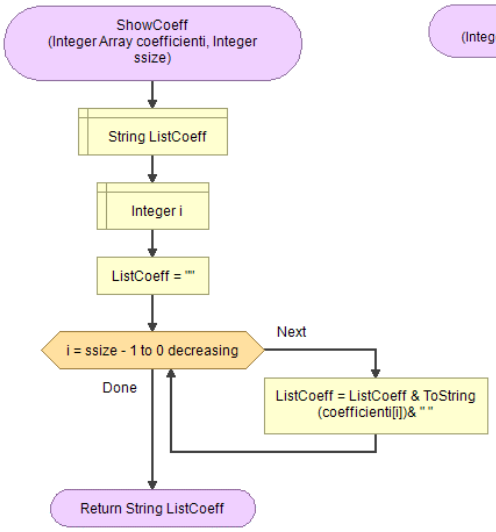
$x^3 - 7x + 6 = (x - 1)\cdot(x - 2)\cdot(x + 3)$

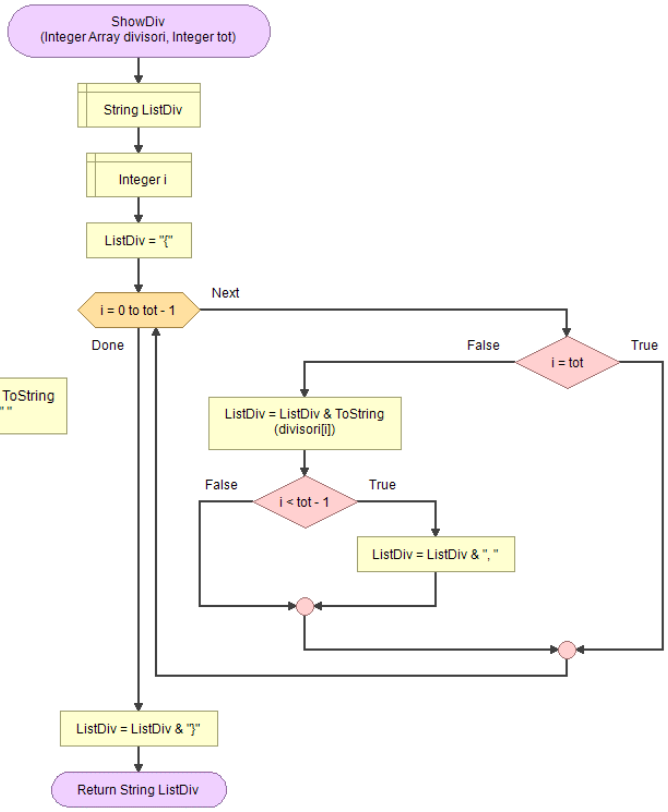
NOTE: try only divisors of the numerical term of the polynomial.

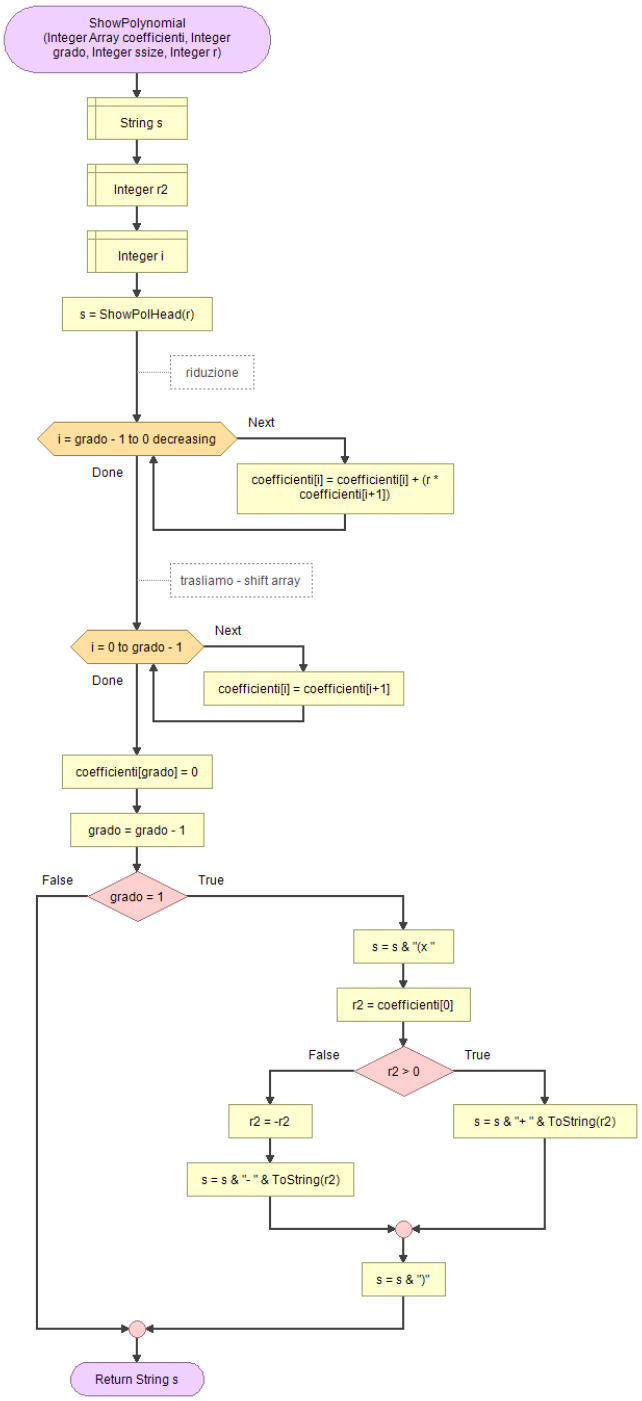

A
B
C
D
tmp = ShowCoeff(coefficienti, ssize)
Output "[IT] Coefficienti del polinomio: " &tmp & ToChar(10) & "[EN] Polynomial coefficients: " & tmp
tmp = ShowDiv(divisori, tot)
Output "[IT] Divisori del termine noto: " & tmp & ToChar(13) & ToChar(10) & "[EN] Submultiples of last coefficient: " & tmp
find a radix
r = FindRadix(coefficienti, divisori, tot, ssize)
False
r <> 0
True
Output "[IT] Nessuna radice razionale " & ToChar(13) & ToChar(10) & "[EN] No rational radix"
RationalRadix = False
Output "[IT] Radice: " & ToString (r) & ToChar(13) & ToChar(10) & "[EN] Radix: " & ToString(r)
Output
s = s & ShowPolynomial (coefficienti, grado, ssize, r)
ssize = ssize - 1
grado = grado - 1
False
k = grado
True
tmp = ShowCoeff(coefficienti, ssize)
Output "[IT] Coefficienti del polinomio: " &tmp & ToChar(10) & "[EN] Polynomial coefficients: " & tmp
r = -coefficienti[0]
Output "[IT] Radice: " & ToString (r) & ToChar(13) & ToChar(10) & "[EN] Radix: " & ToString(r)
k = k + 1
False
RationalRadix = True
True
Output "[IT] Scomposizione del polinomio: " & s & ToChar(13) & ToChar(10) & "[EN] Polynomial factoring: " & s
End

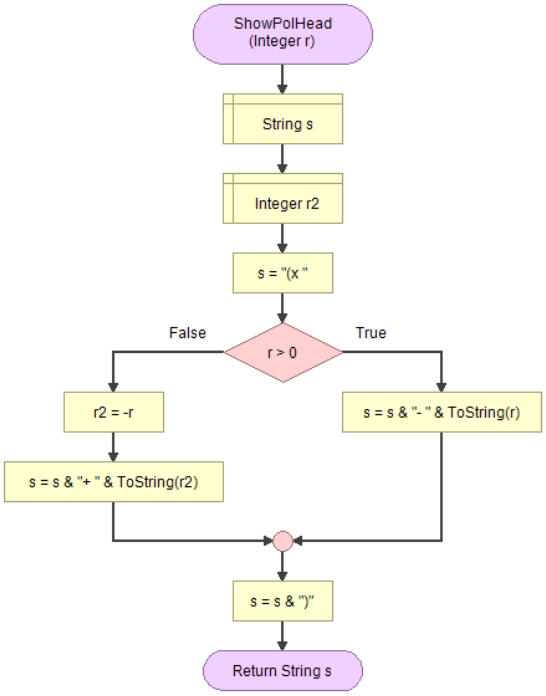

FindDiv
(Integer N, Integer Array divisori)
Integer K
Integer R
Integer pos
pos = 0
K = 1
False
N < 0
True
N = -N
K <= N
True
False
R = N mod K
False
R = 0
True
divisori[pos] = k
pos = pos + 1
divisori[pos] = -k
pos = pos + 1
K = K + 1
Return Integer pos


FindRadix
(Integer Array coefficienti, Integer Array divisori, Integer tot, Integer ssize)
Integer r
Integer i
Integer j
String s
Boolean Found
found = False
i = 0
i <= tot - 1 AND Found = False
True
False
s = ""
r = 0
j = 0 to ssize - 1
Next
Done
r = r + coefficienti[j]*(divisori[i]^j)
s = s & ToString(coefficienti[j]) & " * " & ToString(divisori[i]) & "^" & ToString(j)
False
j < ssize - 1
True
s = s & " + "
Output "P(" & ToString(divisori[i]) & ") => " & s & " = " & ToString(r)
False
r = 0
True
found = True
r = divisori[i]
i = i + 1
False
found = False
True
r = 0
Return Integer r


ShowCoeff
(Integer Array coefficienti, Integer ssize)
String ListCoeff
Integer i
ListCoeff = ""
i = ssize - 1 to 0 decreasing
Next
Done
ListCoeff = ListCoeff & ToString (coefficienti[i])& " "
Return String ListCoeff

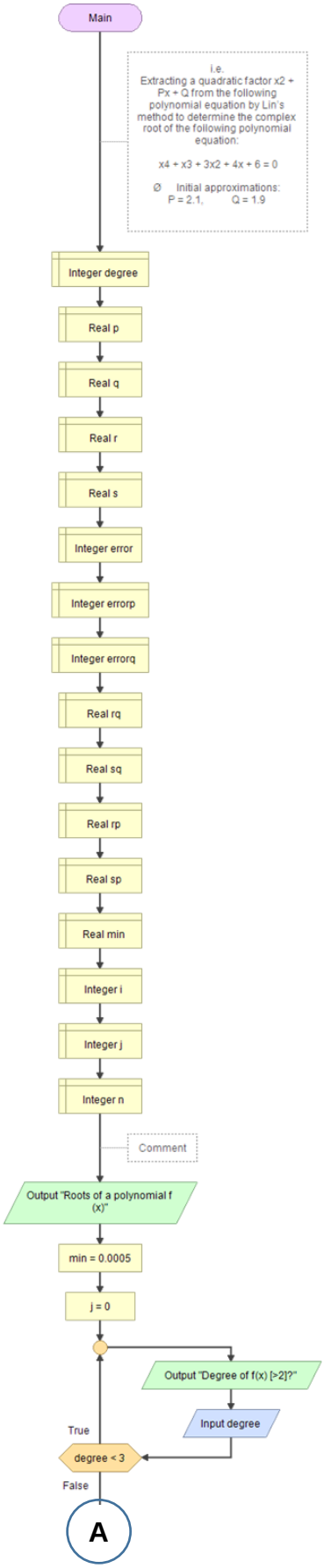

ShowDiv
(Integer Array divisori, Integer tot)
String ListDiv
Integer i
ListDiv = "{"
i = 0 to tot - 1
Next
Done
False
i = tot
True
ListDiv = ListDiv & ToString (divisori[i])
False
i < tot - 1
True
ListDiv = ListDiv & ","
ListDiv = ListDiv & "}"
Return String ListDiv


ShowPolynomial
(Integer Array coefficienti, Integer grado, Integer ssize, Integer r)
String s
Integer r2
Integer i
s = ShowPolHead(r)
riduzione
i = grado - 1 to 0 decreasing
Next
Done
coefficienti[i] = coefficienti[i] + (r * coefficienti[i+1])
trasliamo - shift array
i = 0 to grado - 1
Next
Done
coefficienti[i] = coefficienti[i+1]
coefficienti[grado] = 0
grado = grado - 1
False
grado = 1
True
s = s & "(x "
r2 = coefficienti[0]
False
r2 > 0
True
r2 = -r2
s = s & "- " & ToString(r2)
s = s & "+ " & ToString(r2)
s = s & ")"
Return String s


ShowPolHead
(Integer r)
String s
Integer r2
s = "(x "
False
r > 0
True
r2 = -r
s = s & "- " & ToString(r)
s = s & "+ " & ToString(r2)
s = s & ")"
Return String s

289. G A | Calcola le radici complesse di un polinomio f(x) col metodo di Bairstow

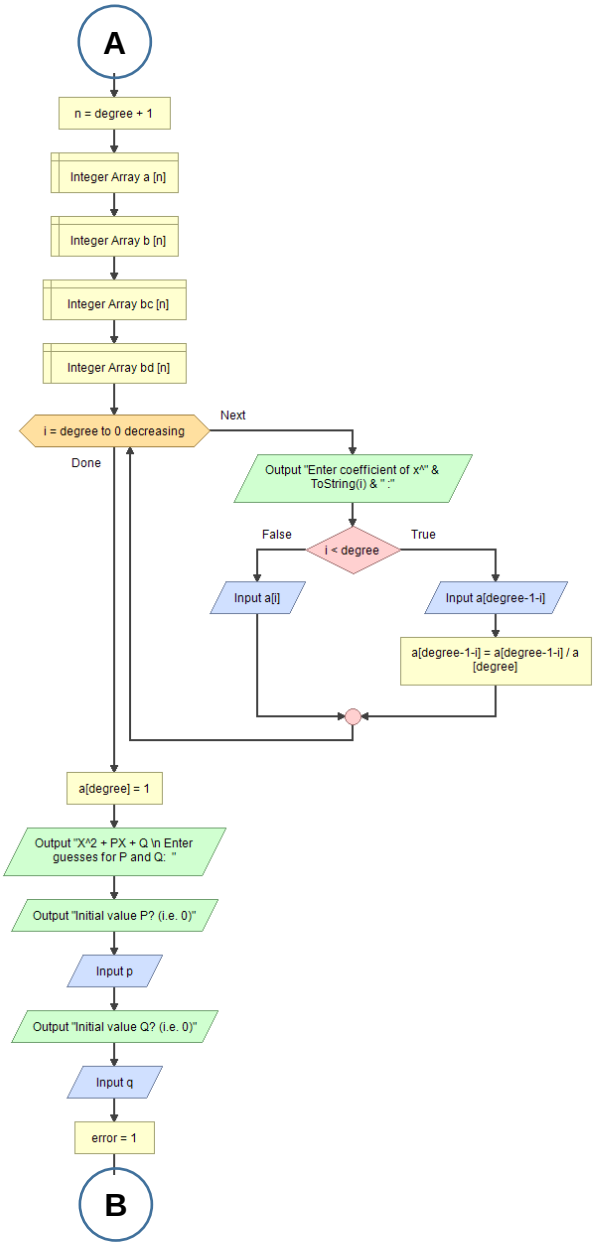
Calc the roots of a polynomial f(x) by Bairstow's method

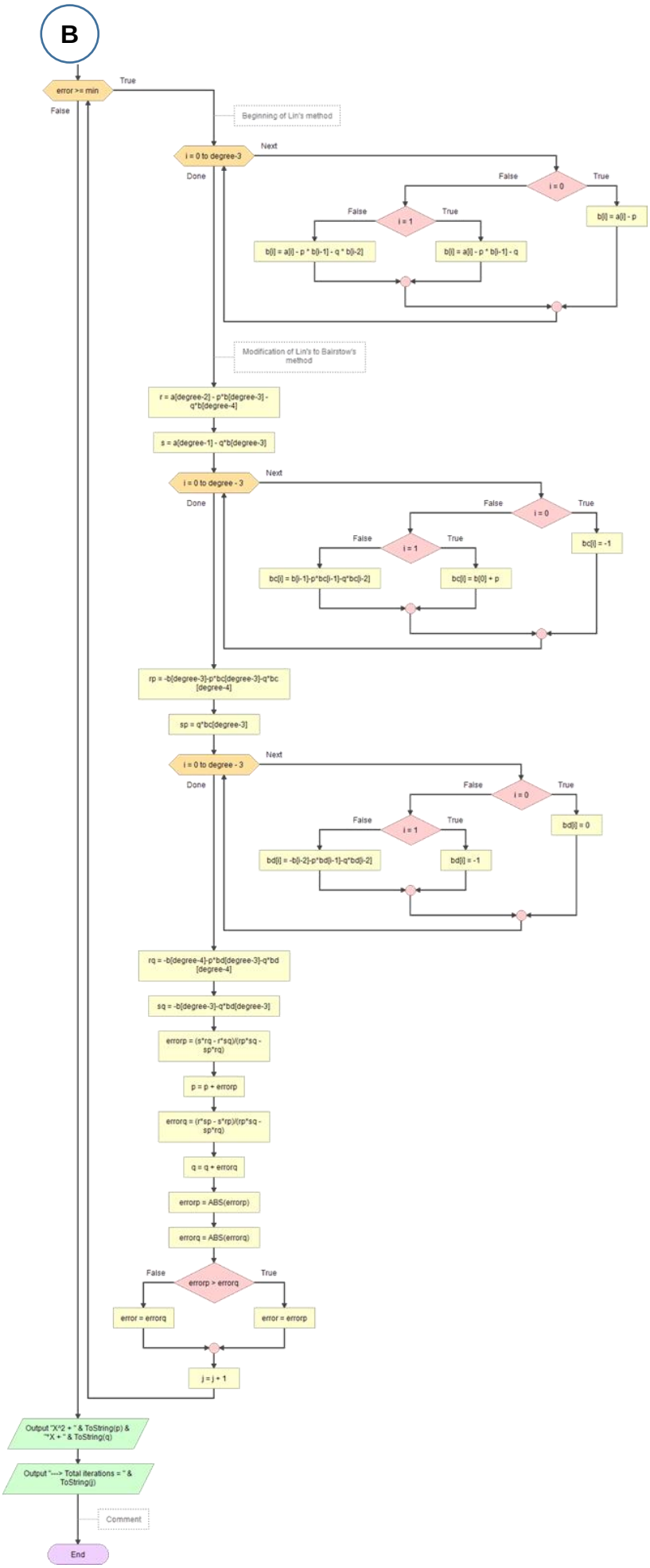


i.e.
Extracting a quadratic factor x2 + Px + Q from the following polynomial equation by Lin's method to determine the complex root of the following polynomial equation:

x4 + x3 + 3x2 + 4x + 6 = 0

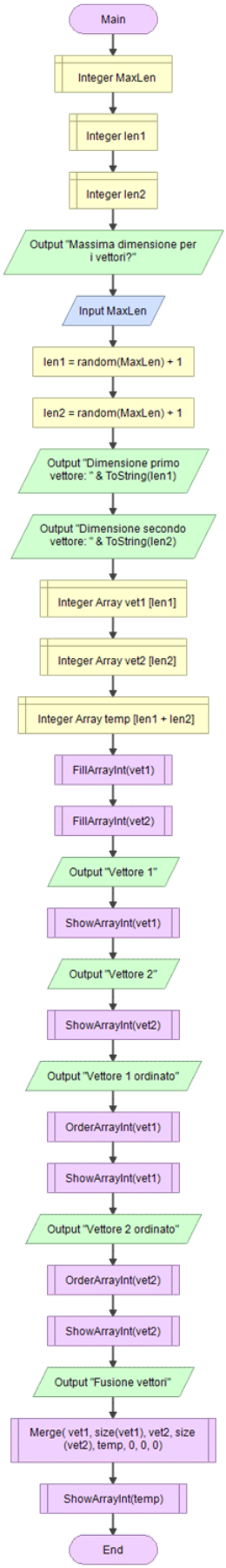
Ø Initial approximations: P = 2.1, Q = 1.9

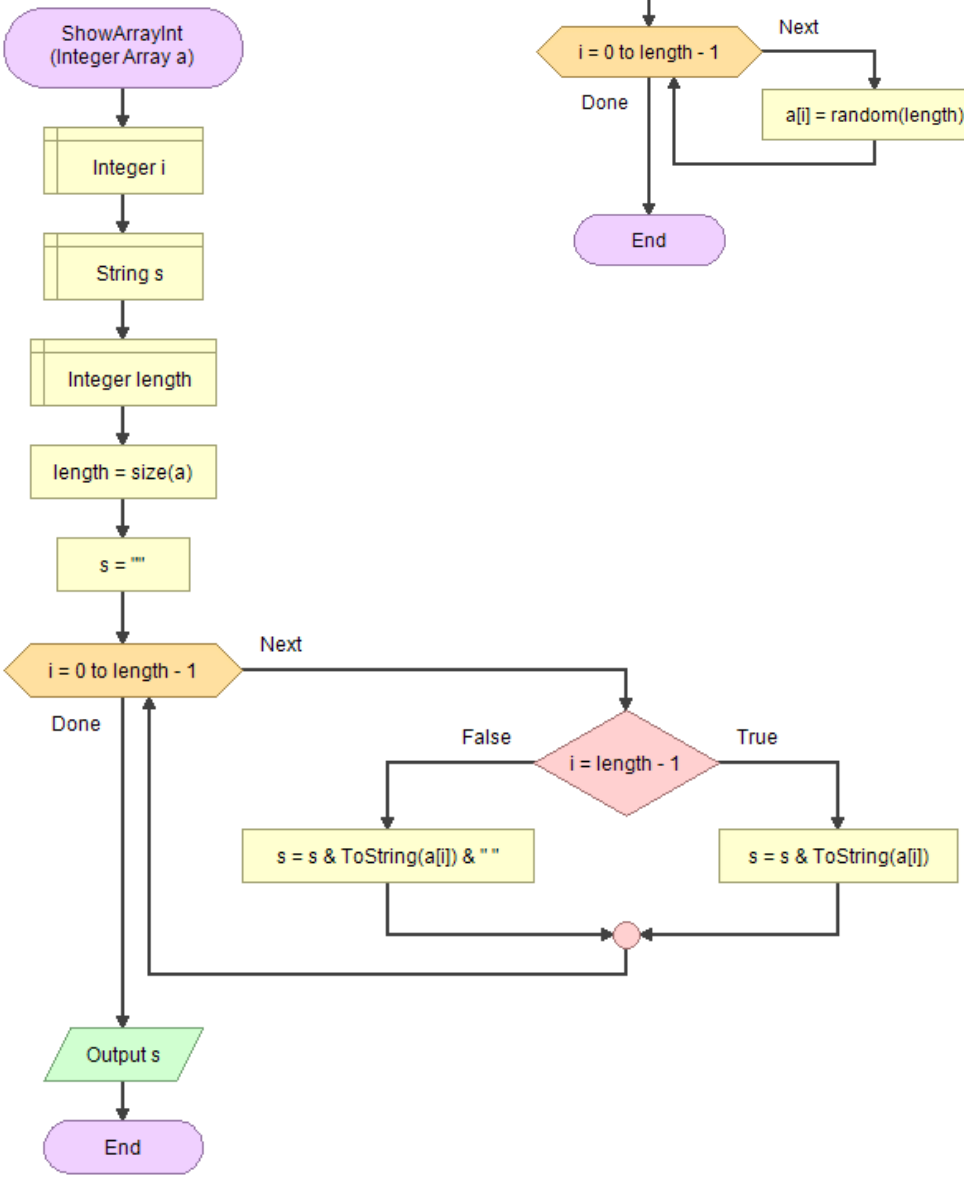


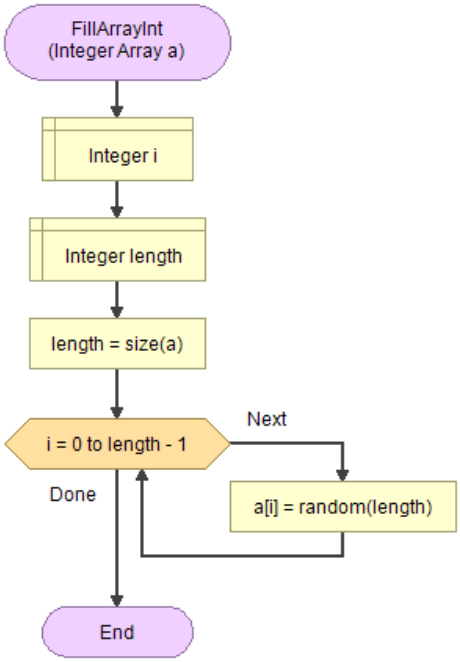

B
error >= min
True
False
Beginning of Lin's method
i = 0 to degree-3
Next
Done
i = 0
False
True
b[i] = a[i] - p
i = 1
False
True
b[i] = a[i] - p * b[i-1] - q * b[i-2]
b[i] = a[i] - p * b[i-1] - q
Modification of Lin's to Bairstow's method
r = a[degree-2] - p*b[degree-3] - q*b[degree-4]
s = a[degree-1] - q*b[degree-3]
i = 0 to degree - 3
Next
Done
i = 0
False
True
bc[i] = -1
i = 1
False
True
bc[i] = b[i-1]-p*bc[i-1]-q*bc[i-2]
bc[i] = b[0] + p
rp = -b[degree-3]-p*bc[degree-3]-q*bc[degree-4]
sp = q*bc[degree-3]
i = 0 to degree - 3
Next
Done
i = 0
False
True
bd[i] = 0
i = 1
False
True
bd[i] = -b[i-2]-p*bd[i-1]-q*bd[i-2]
bd[i] = -1
rq = -b[degree-4]-p*bd[degree-3]-q*bd[degree-4]
sq = -b[degree-3]-q*bd[degree-3]
errorp = (s*rq - r*sq)/(rp*sq - sp*rq)
p = p + errorp
errorq = (r*sp - s*rp)/(rp*sq - sp*rq)
q = q + errorq
errorp = ABS(errorp)
errorq = ABS(errorq)
False
errorp > errorq
True
error = errorq
error = errorp
j = j + 1
Output "X^2 + " & ToString(p) & "*X + " & ToString(q)
Output "---> Total iterations = " & ToString(j)
Comment
End

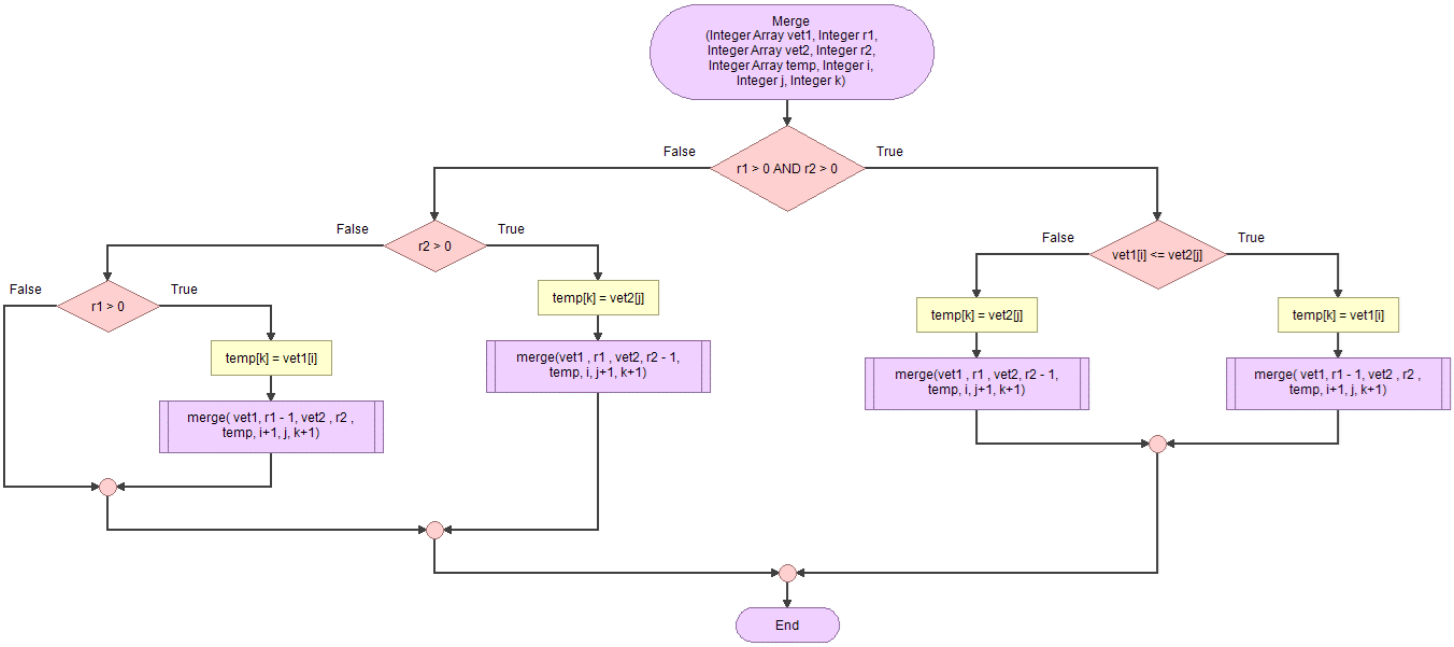
290.GAFR ‖ Fusione ricorsiva tra due vettori di interi ordinati

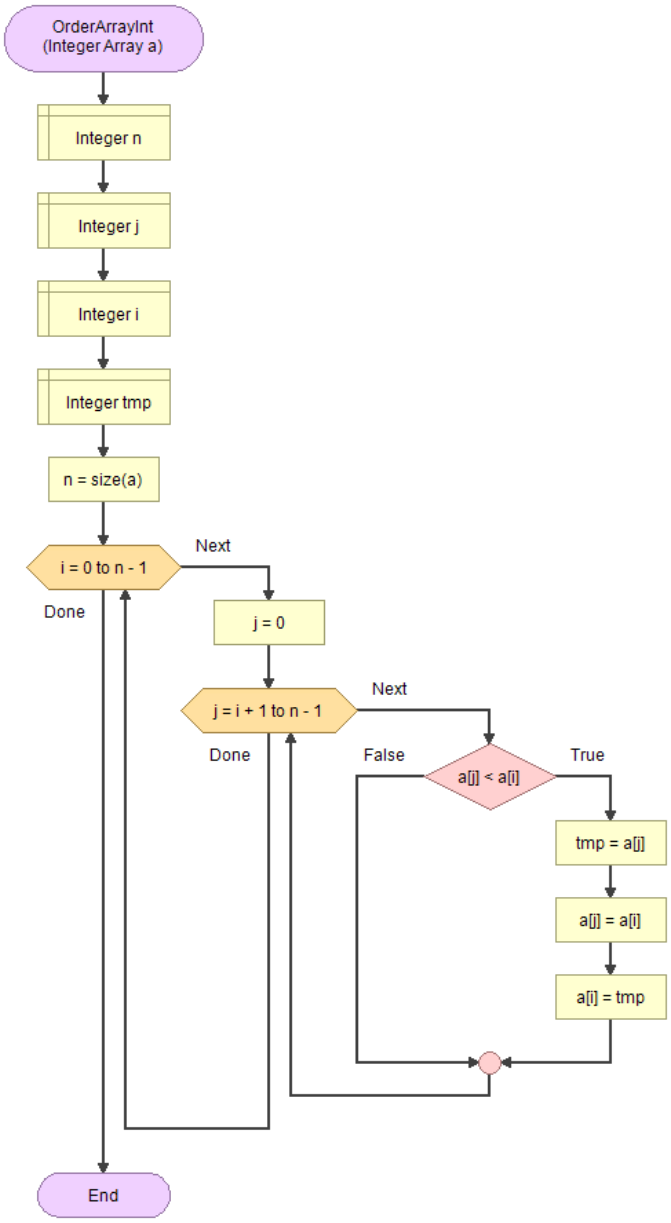
Recursive fusion of 2 integer ordered arrays

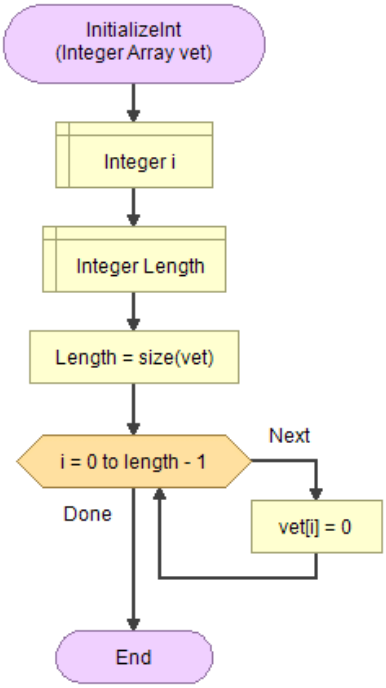






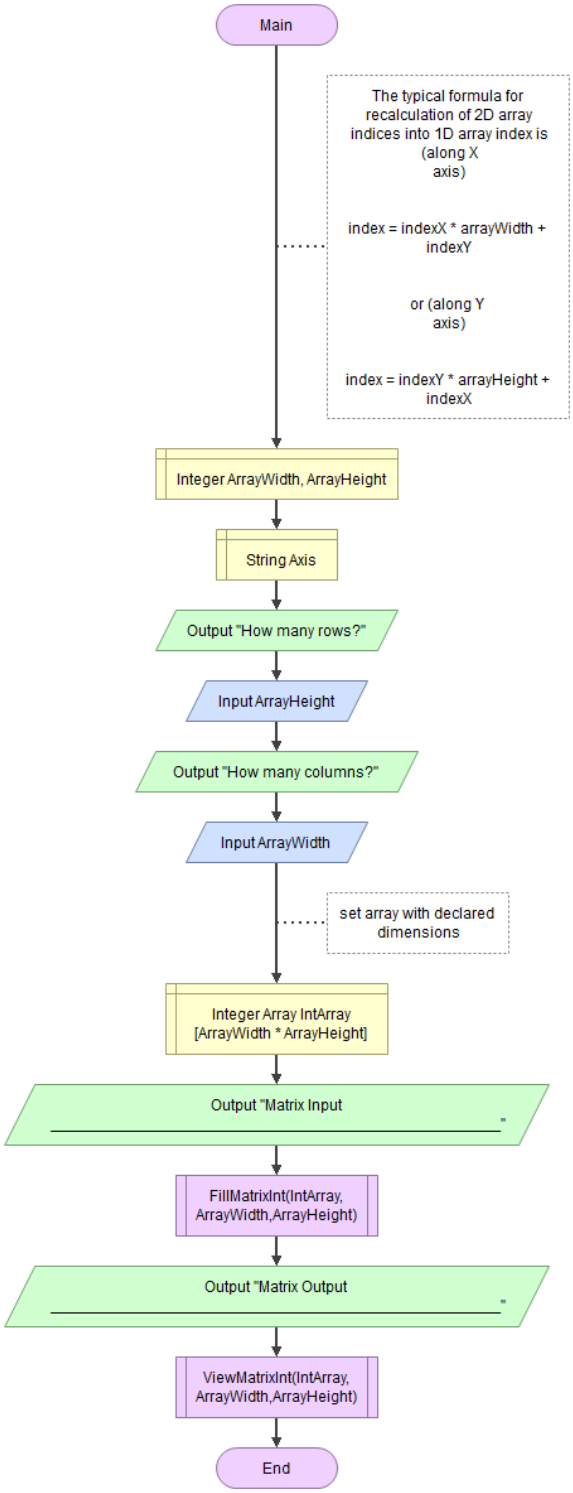

Merge
(Integer Array vet1, Integer r1,
Integer Array vet2, Integer r2,
Integer Array temp, Integer i,
Integer j, Integer k)
r1 > 0 AND r2 > 0
False
True
r2 > 0
False
True
r1 > 0
False
True
temp[k] = vet1[i]
merge(vet1, r1 - 1, vet2 , r2 ,
temp, i+1, j, k+1)
temp[k] = vet2[j]
merge(vet1 , r1 , vet2, r2 - 1,
temp, i, j+1, k+1)
vet1[i] <= vet2[j]
False
True
temp[k] = vet2[j]
merge(vet1 , r1 , vet2, r2 - 1,
temp, i, j+1, k+1)
temp[k] = vet1[i]
merge(vet1, r1 - 1, vet2 , r2 ,
temp, i+1, j, k+1)
End

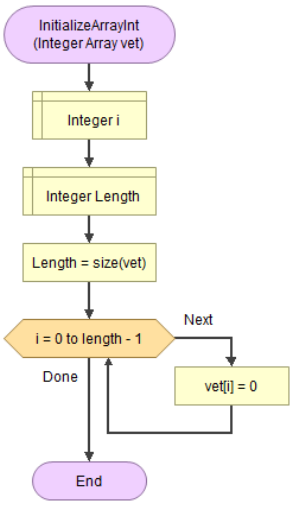

OrderArrayInt
(Integer Array a)
Integer n
Integer j
Integer i
Integer tmp
n = size(a)
i = 0 to n - 1
Next
Done
j = 0
j = i + 1 to n - 1
Next
Done
False
a[j] < a[i]
True
tmp = a[j]
a[j] = a[i]
a[i] = tmp
End

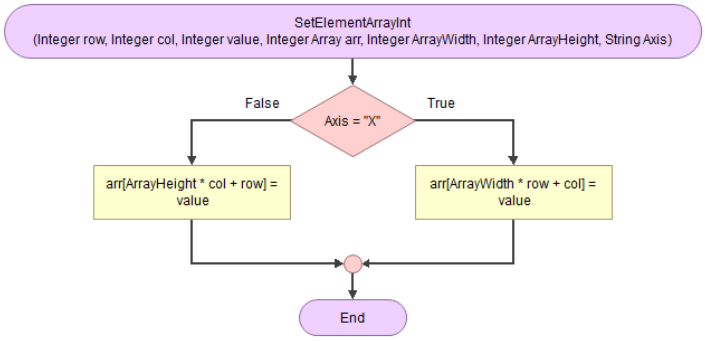

InitializeInt
(Integer Array vet)
Integer i
Integer Length
Length = size(vet)
i = 0 to length - 1
Next
Done
vet[i] = 0
End

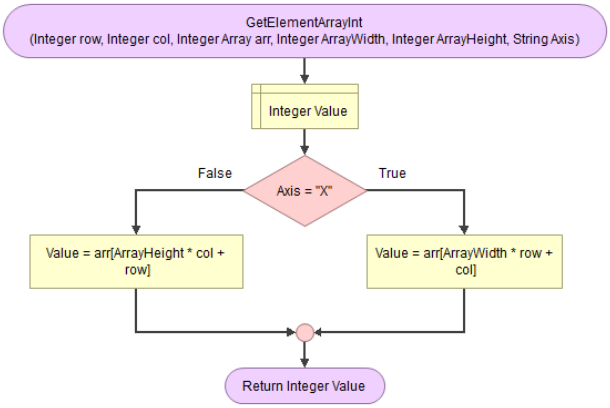
291. G A F ‖ Trasformazione array bidimensionale (Matrice) in array monodimensionale (Vettore)

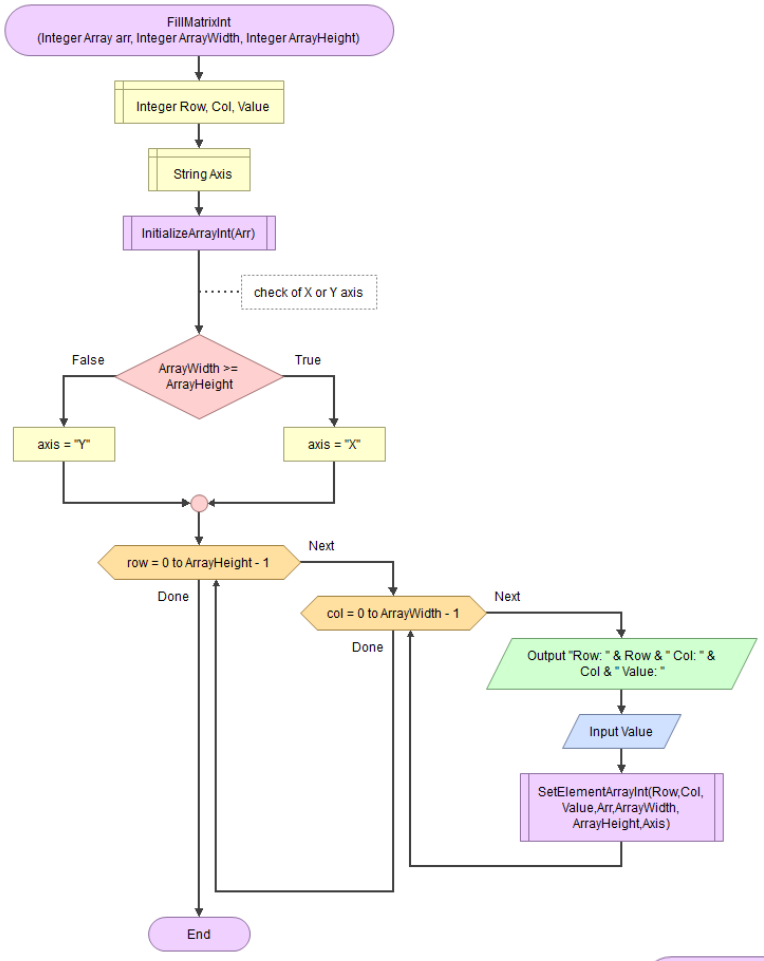
Map a 2D array onto a 1D array

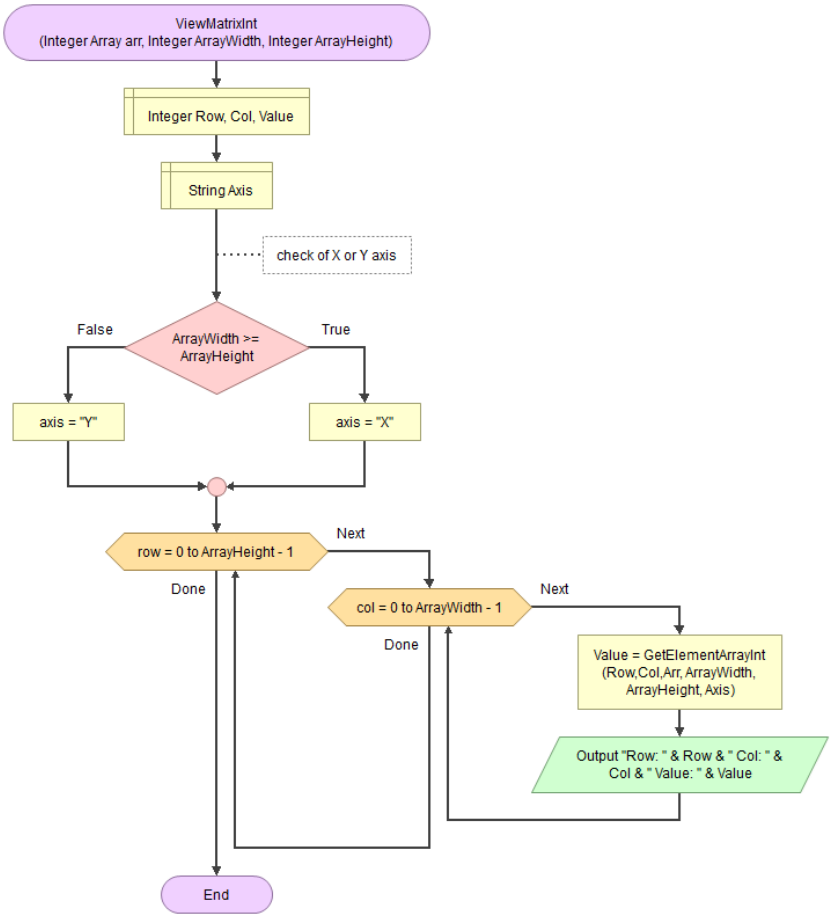






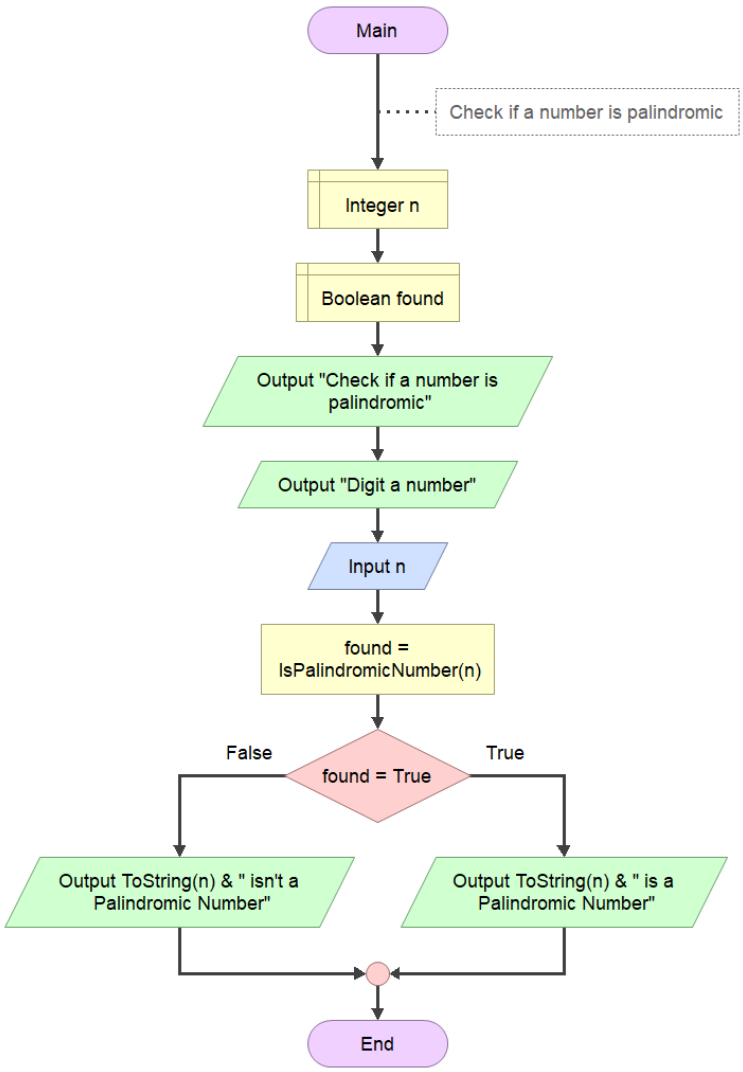


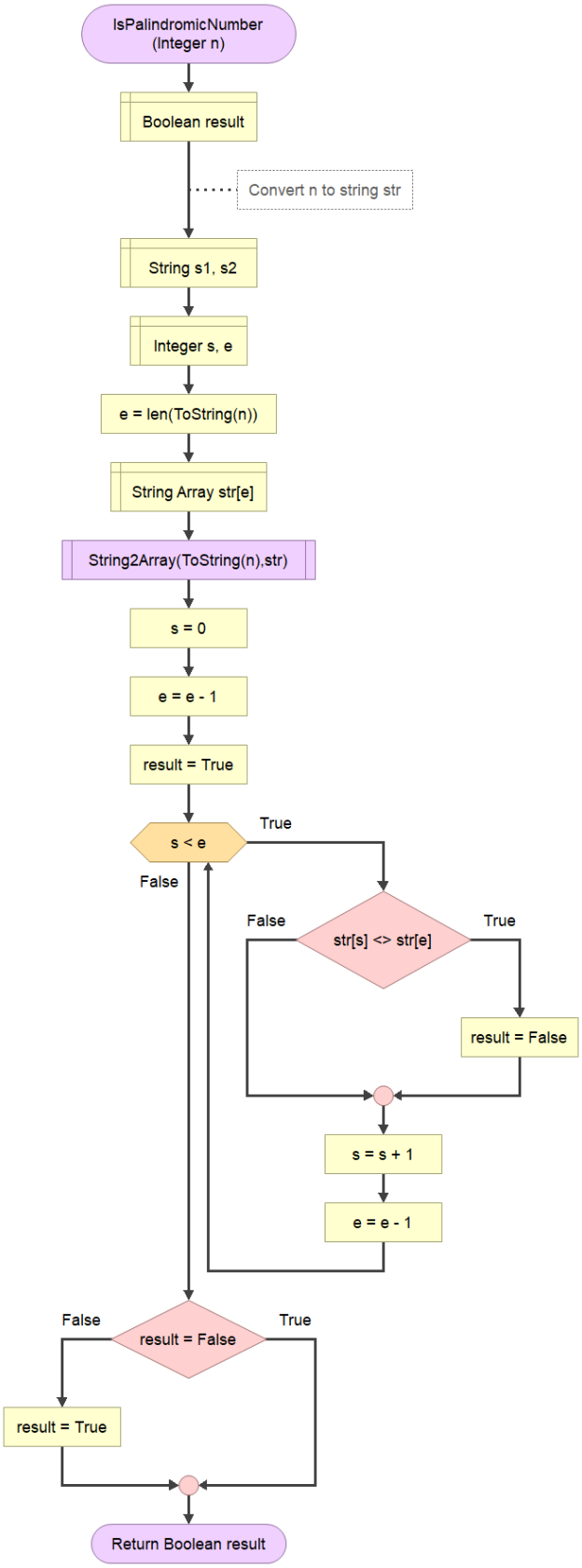

FillMatrixInt
(Integer Array arr, Integer ArrayWidth, Integer ArrayHeight)
Integer Row, Col, Value
String Axis
InitializeArrayInt(Arr)
check of X or Y axis
False
ArrayWidth >= ArrayHeight
True
axis = "Y"
axis = "X"
row = 0 to ArrayHeight - 1
Next
Done
col = 0 to ArrayWidth - 1
Next
Done
Output "Row: " & Row & " Col: " & Col & " Value: "
Input Value
SetElementArrayInt(Row,Col, Value,Arr,ArrayWidth, ArrayHeight,Axis)
End

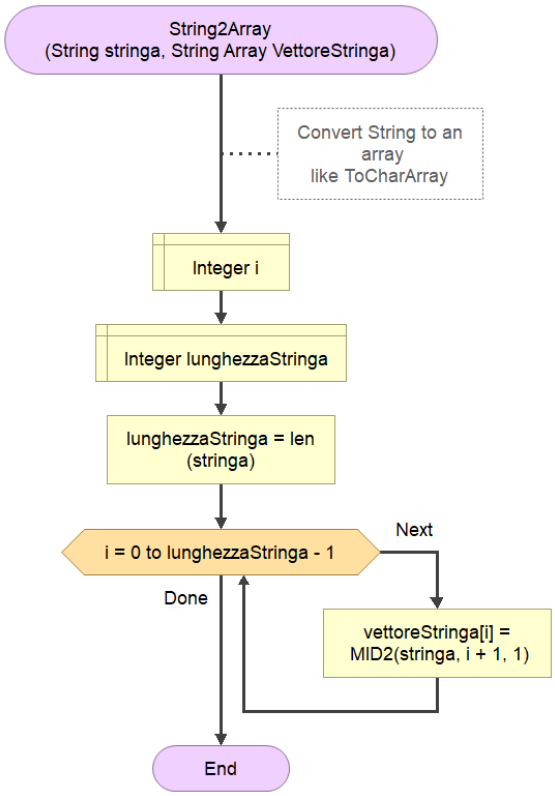

ViewMatrixInt
(Integer Array arr, Integer ArrayWidth, Integer ArrayHeight)
Integer Row, Col, Value
String Axis
check of X or Y axis
False
ArrayWidth >= ArrayHeight
True
axis = "Y"
axis = "X"
row = 0 to ArrayHeight - 1
Next
Done
col = 0 to ArrayWidth - 1
Next
Done
Value = GetElementArrayInt (Row,Col,Arr, ArrayWidth, ArrayHeight, Axis)
Output "Row: " & Row & " Col: " & Col & " Value: " & Value
End

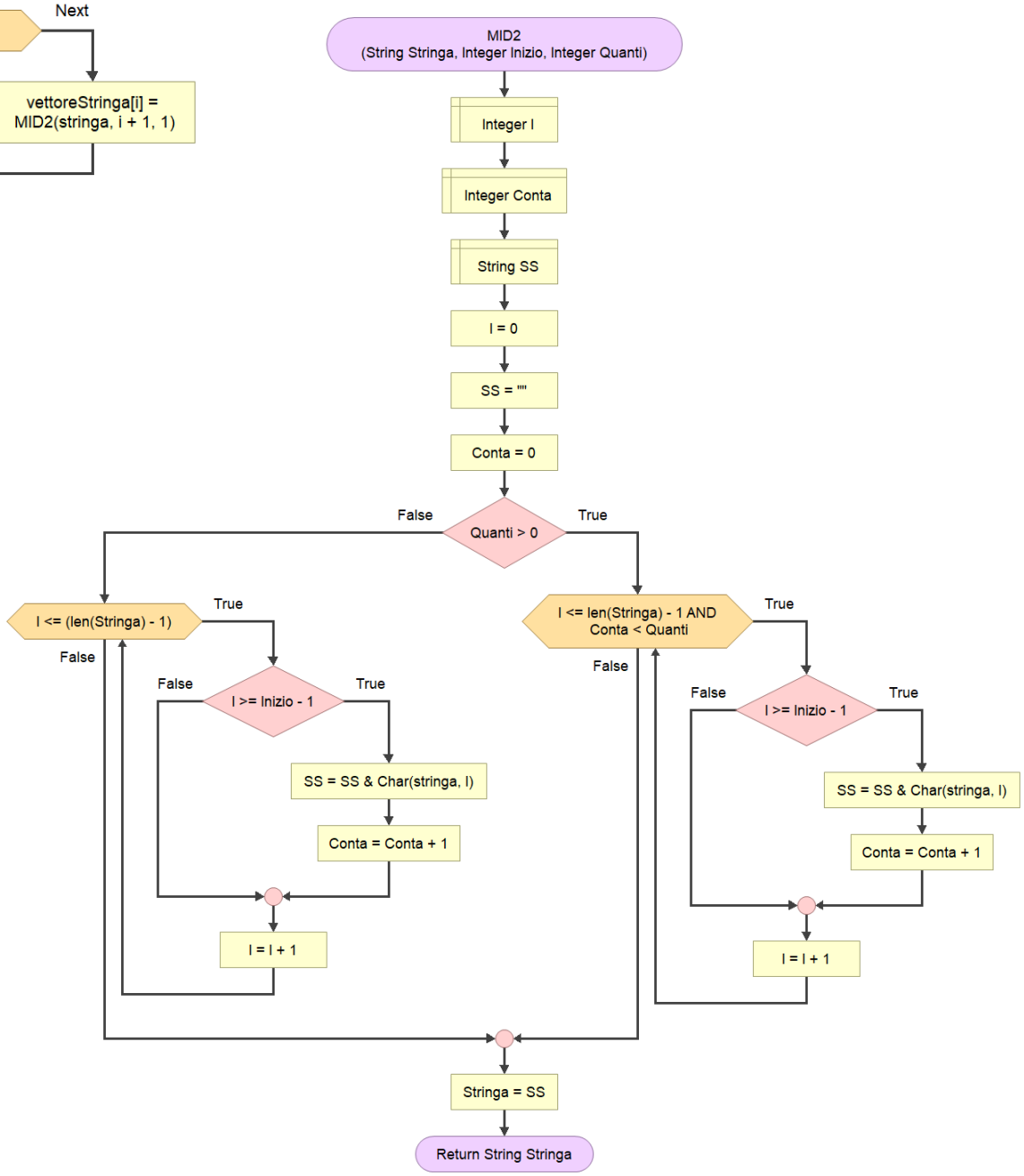
292.GAF ‖ Numero palindromo

Check if a number is palindromic



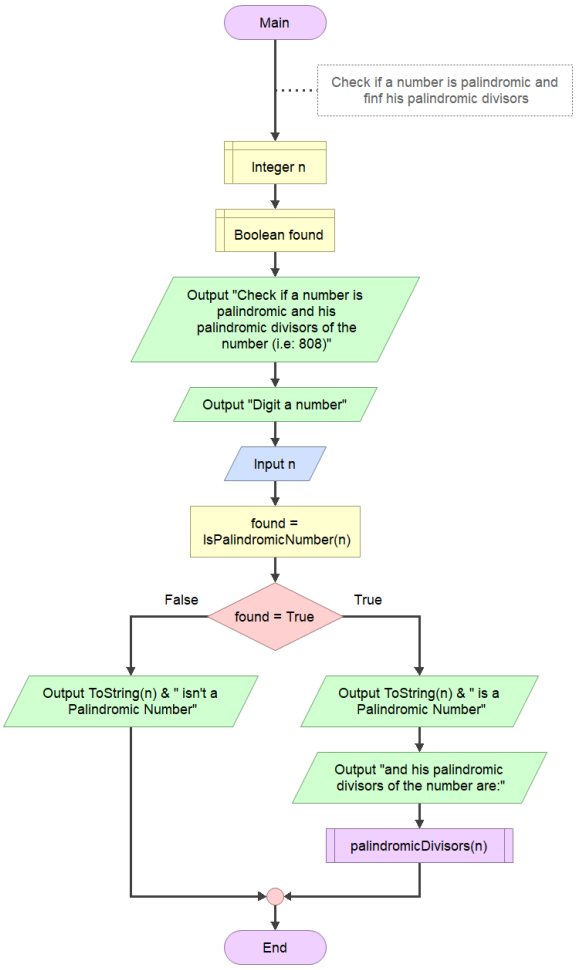


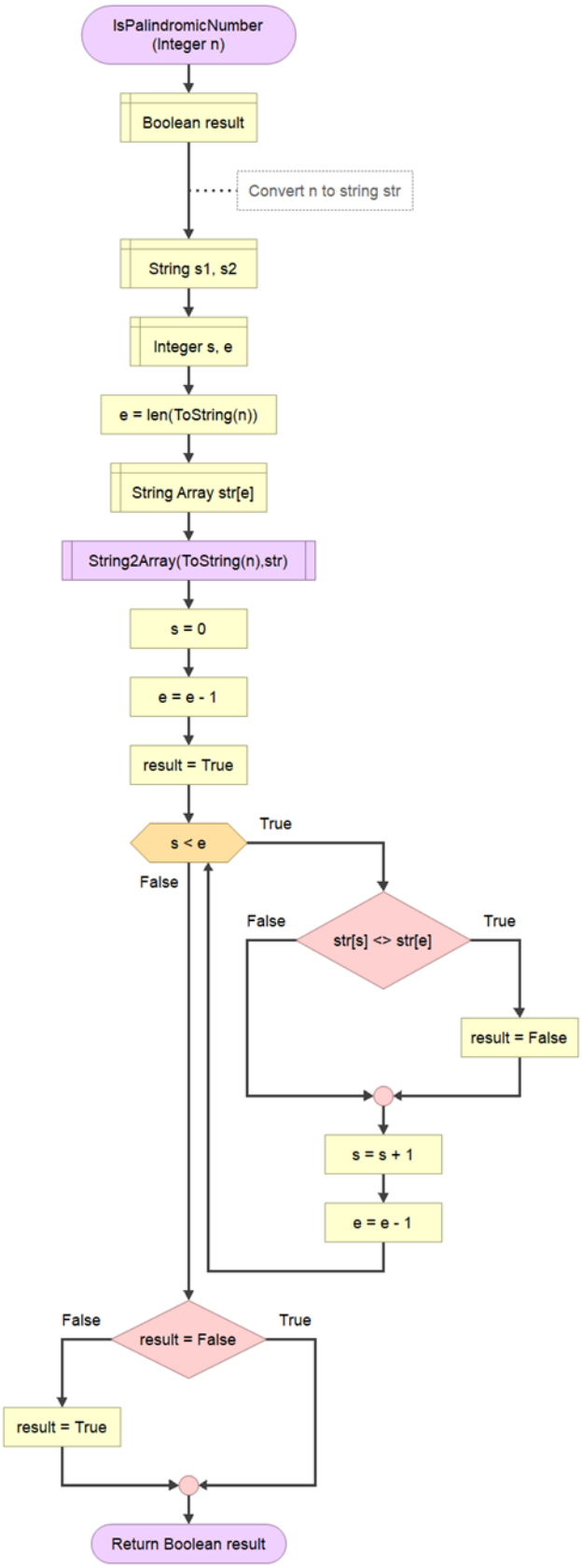

String2Array
(String stringa, String Array VettoreStringa)
Convert String to an array like ToCharArray
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len (stringa)
i = 0 to lunghezzaStringa - 1
Next
Done
vettoreStringa[i] = MID2(stringa, i + 1, 1)
End

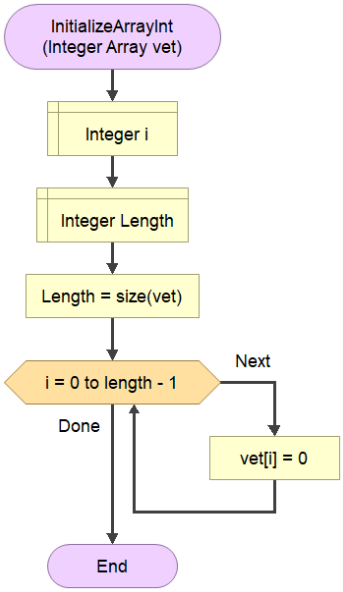

MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

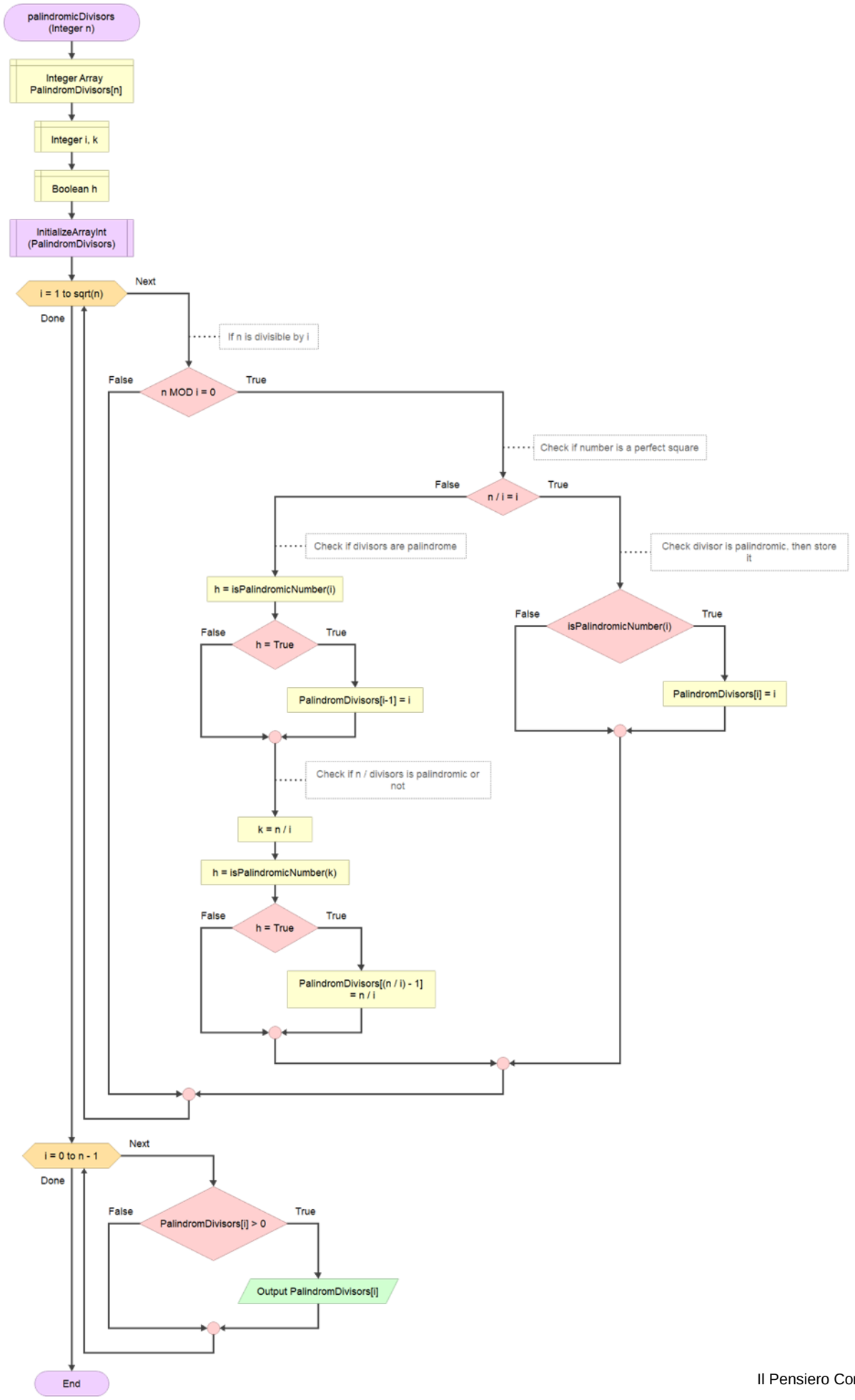
293.GAF ‖ Divisori palindromi di un numero

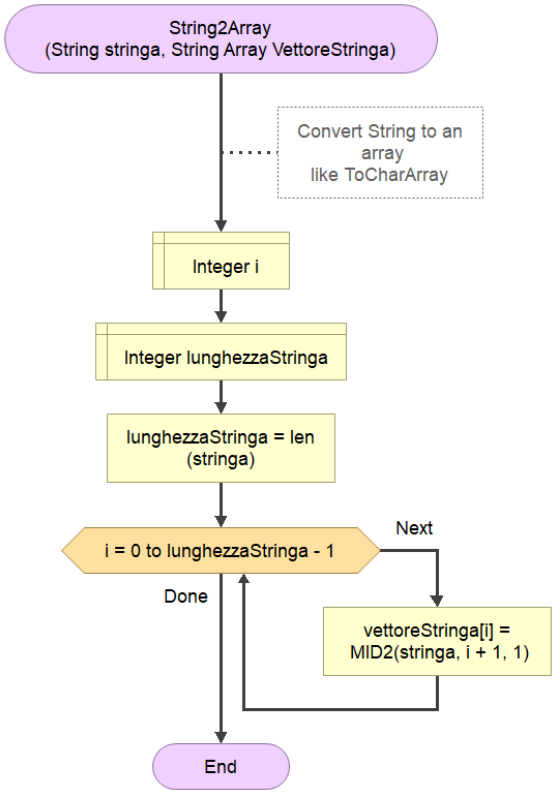
Check if a number is palindromic and find his palindromic divisors

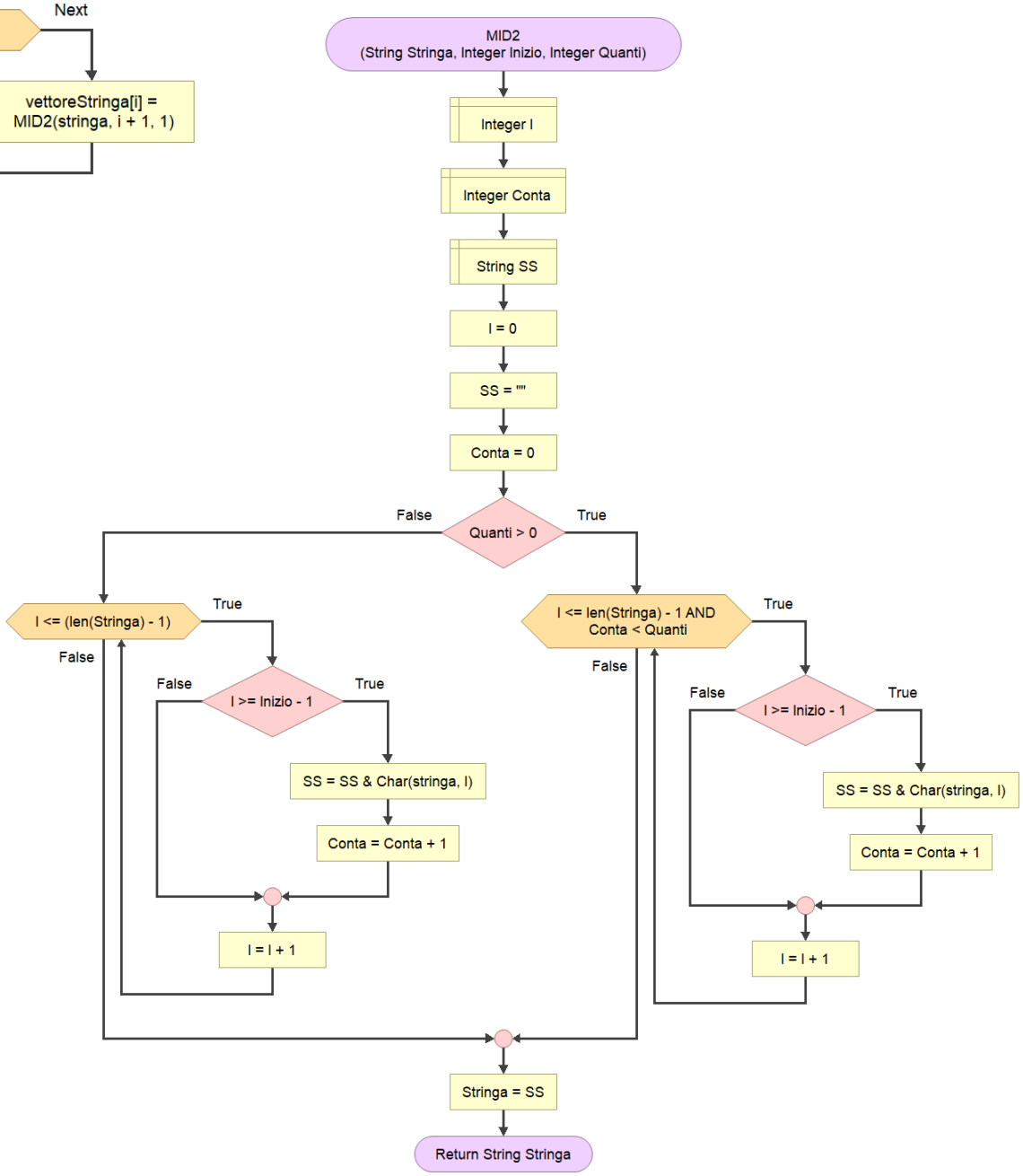






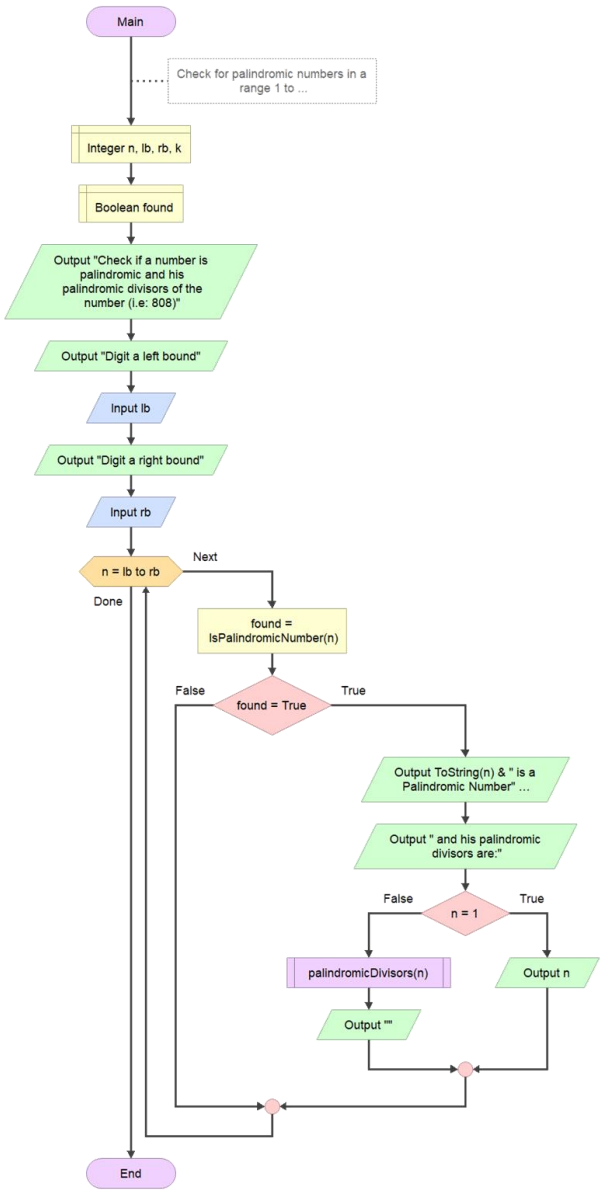

palindromicDivisors (Integer n)
Integer Array PalindromDivisors[n]
Integer i, k
Boolean h
InitializeArrayInt (PalindromDivisors)
i = 1 to sqrt(n)
Next
Done
If n is divisible by i
False
n MOD i = 0
True
Check if number is a perfect square
False
n / i = i
True
Check if divisors are palindrome
Check divisor is palindromic, then store it
h = isPalindromicNumber(i)
False
h = True
True
PalindromDivisors[i-1] = i
False
isPalindromicNumber(i)
True
PalindromDivisors[i] = i
Check if n / divisors is palindromic or not
k = n / i
h = isPalindromicNumber(k)
False
h = True
True
PalindromDivisors[(n / i) - 1] = n / i
i = 0 to n - 1
Next
Done
False
PalindromDivisors[i] > 0
True
Output PalindromDivisors[i]
End

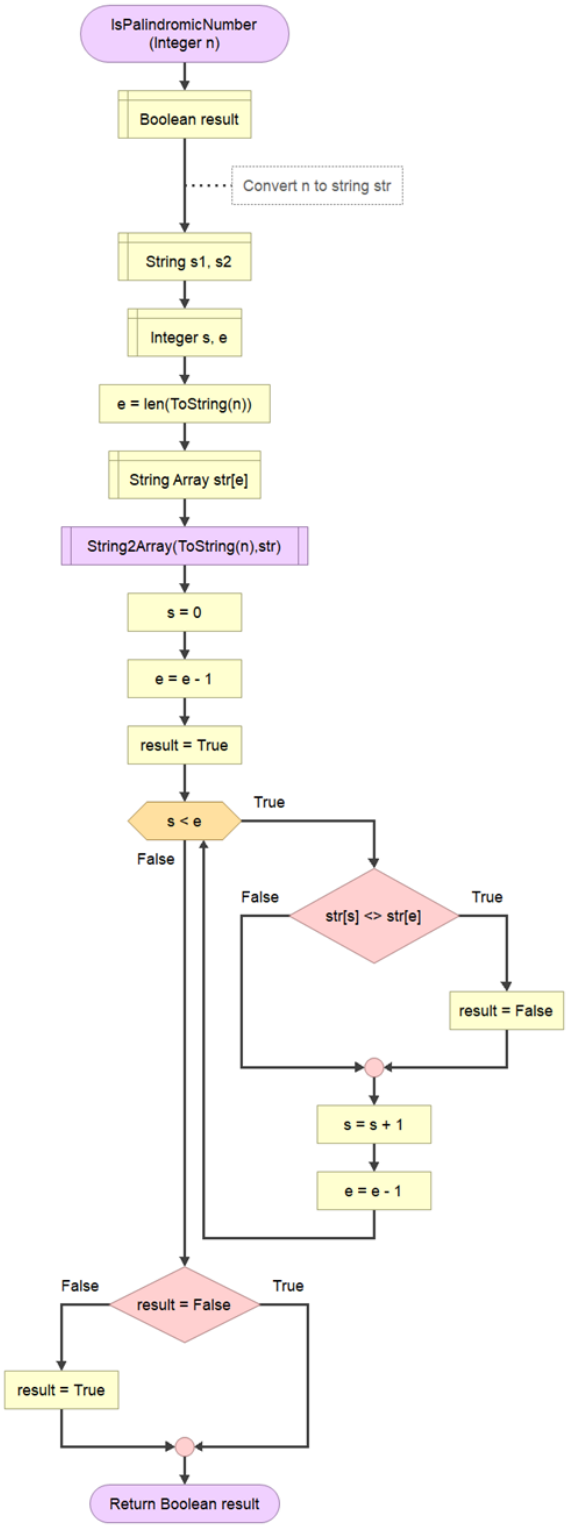

String2Array
(String stringa, String Array VettoreStringa)
Convert String to an array like ToCharArray
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len (stringa)
i = 0 to lunghezzaStringa - 1
Next
Done
vettoreStringa[i] = MID2(stringa, i + 1, 1)
End

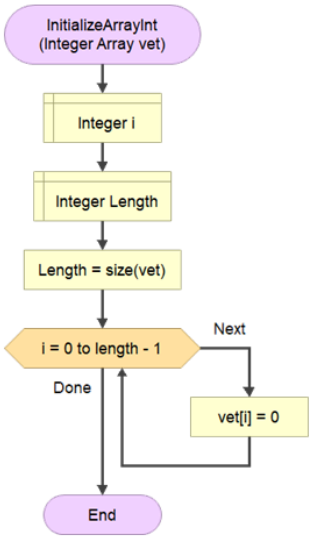

MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

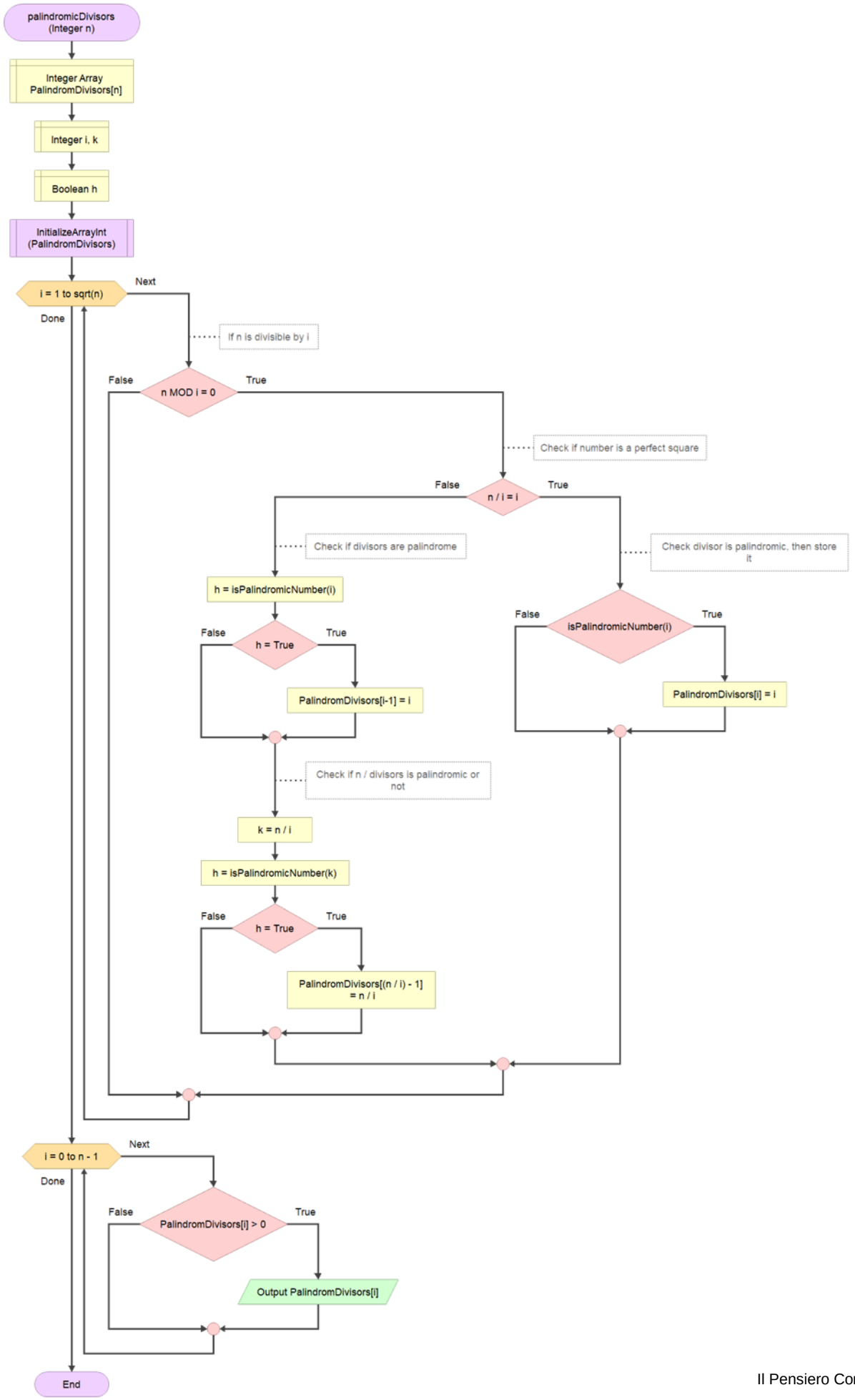
294.GAF ‖ Divisori palindromi in un intervallo numerico

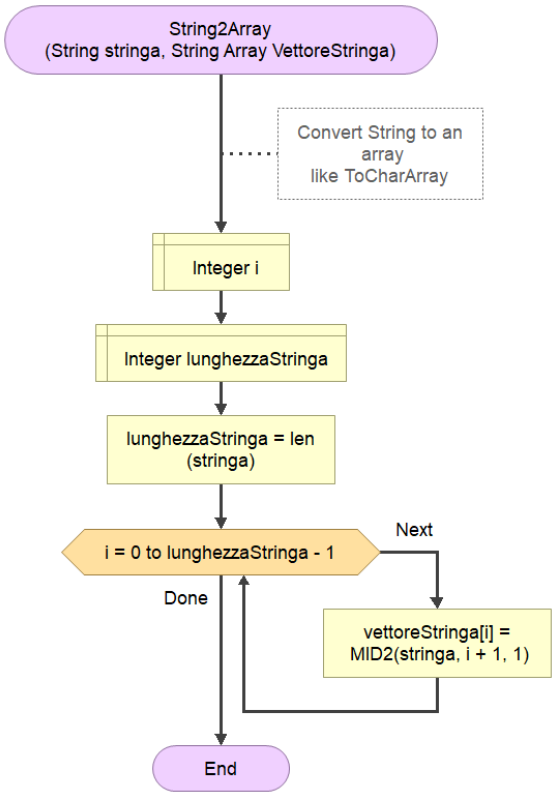
Check if a number is palindromic and find his palindromic divisors in a range

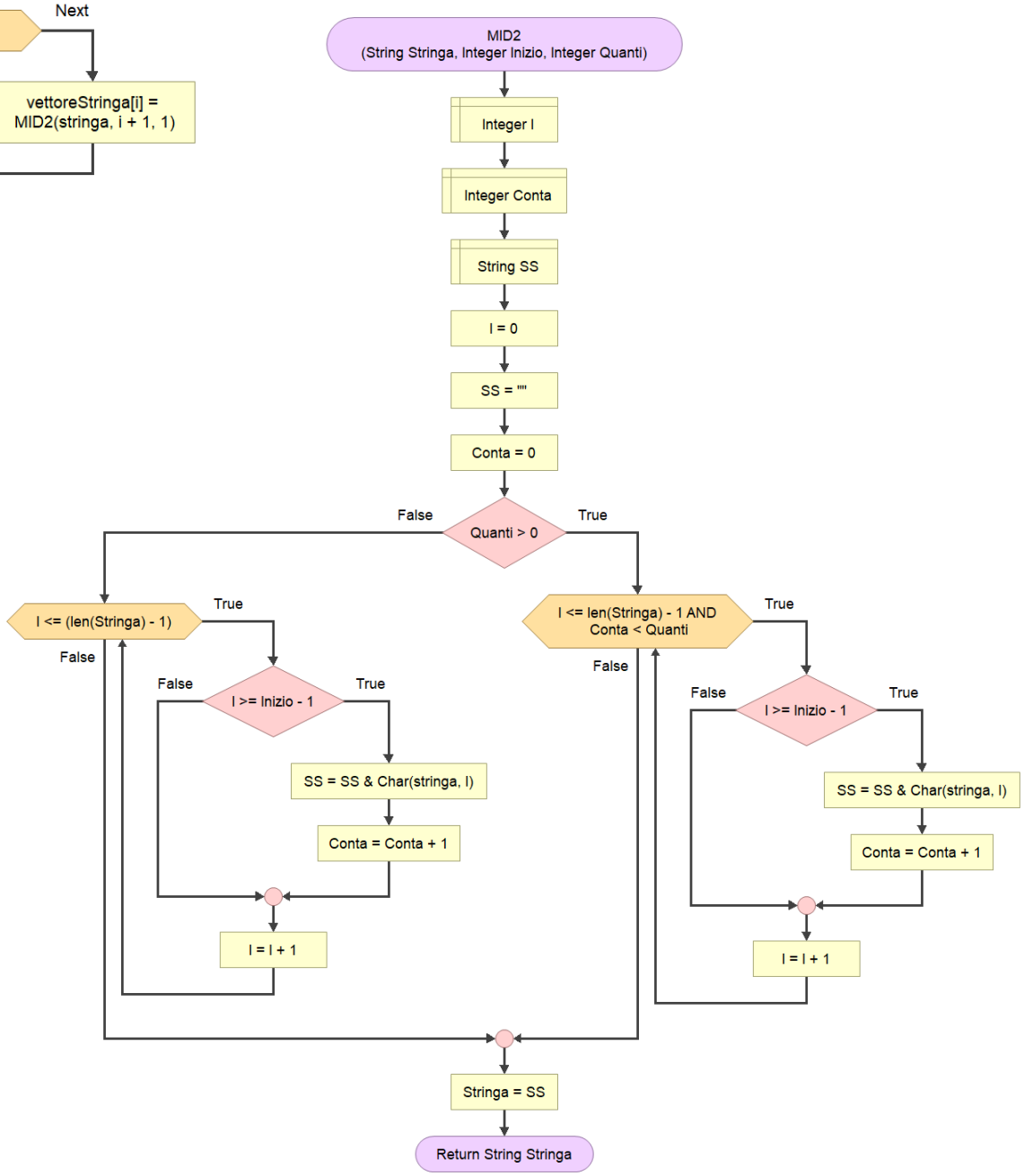






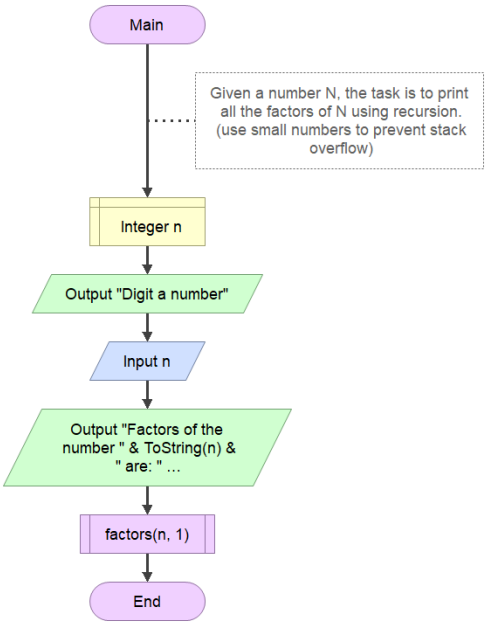

palindromicDivisors (Integer n)
Integer Array PalindromDivisors[n]
Integer i, k
Boolean h
InitializeArrayInt (PalindromDivisors)
i = 1 to sqrt(n)
Next
Done
If n is divisible by i
False
n MOD i = 0
True
Check if number is a perfect square
False
n / i = i
True
Check if divisors are palindrome
Check divisor is palindromic, then store it
h = isPalindromicNumber(i)
False
h = True
True
PalindromDivisors[i-1] = i
False
isPalindromicNumber(i)
True
PalindromDivisors[i] = i
Check if n / divisors is palindromic or not
k = n / i
h = isPalindromicNumber(k)
False
h = True
True
PalindromDivisors[(n / i) - 1] = n / i
i = 0 to n - 1
Next
Done
False
PalindromDivisors[i] > 0
True
Output PalindromDivisors[i]
End

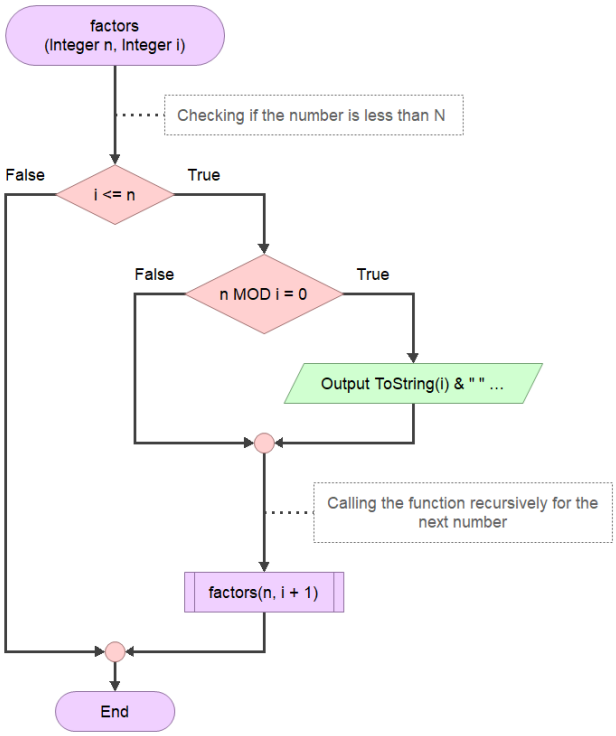

String2Array
(String stringa, String Array VettoreStringa)
Convert String to an array like ToCharArray
Integer i
Integer lunghezzaStringa
lunghezzaStringa = len (stringa)
i = 0 to lunghezzaStringa - 1
Next
Done
vettoreStringa[i] = MID2(stringa, i + 1, 1)
End


MID2
(String Stringa, Integer Inizio, Integer Quanti)
Integer I
Integer Conta
String SS
I = 0
SS = ""
Conta = 0
Quanti > 0
False
True
I <= (len(Stringa) - 1)
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
I <= len(Stringa) - 1 AND Conta < Quanti
True
False
I >= Inizio - 1
False
True
SS = SS & Char(stringa, I)
Conta = Conta + 1
I = I + 1
Stringa = SS
Return String Stringa

295.GFR ‖ Ricerca ricorsiva dei fattori di un numero

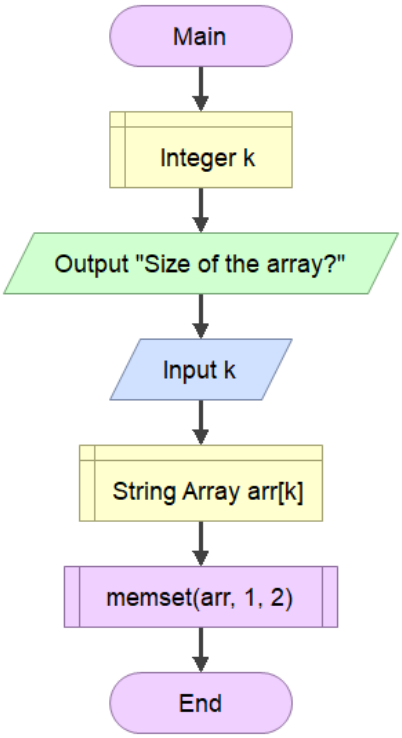
Find all Factors of a Number using recursion

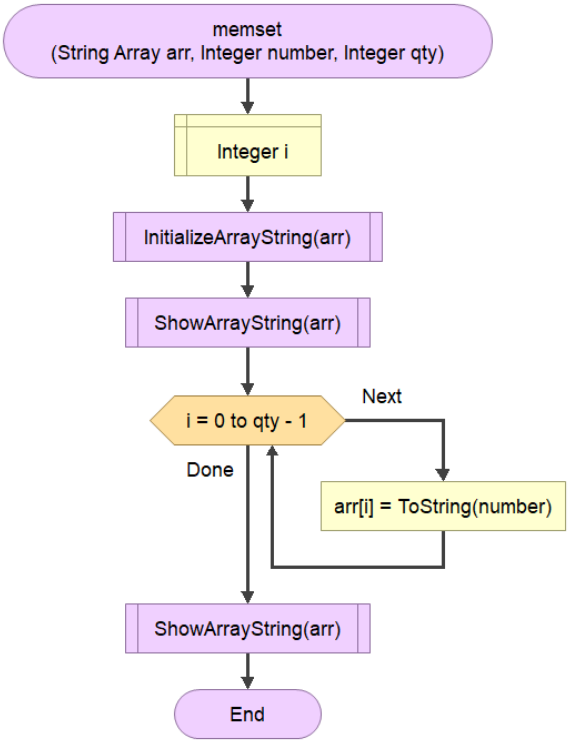


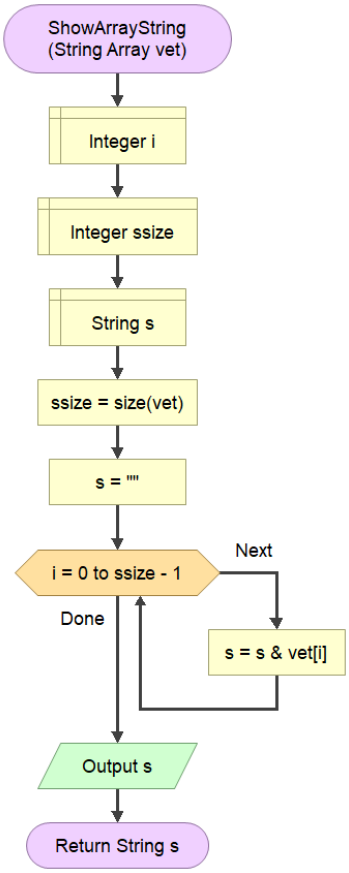


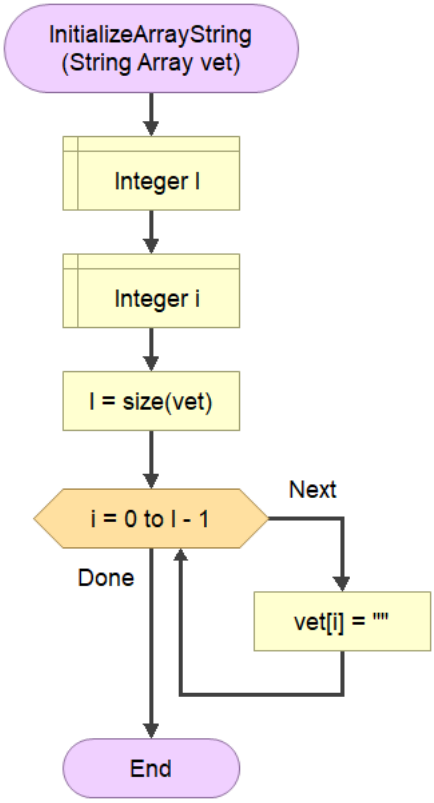
296.**G A F** ‖ Simulazione funzione 'memset' del linguaggio C

Like memset function in c



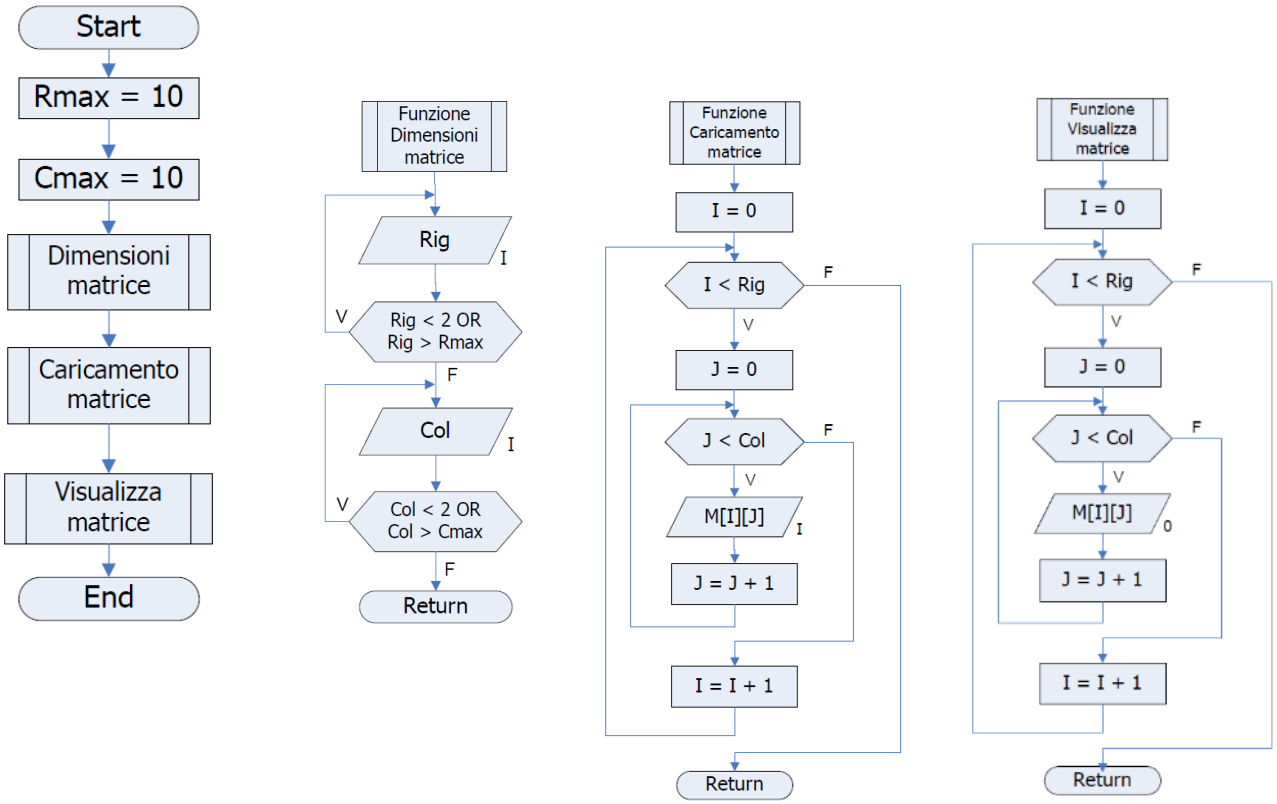






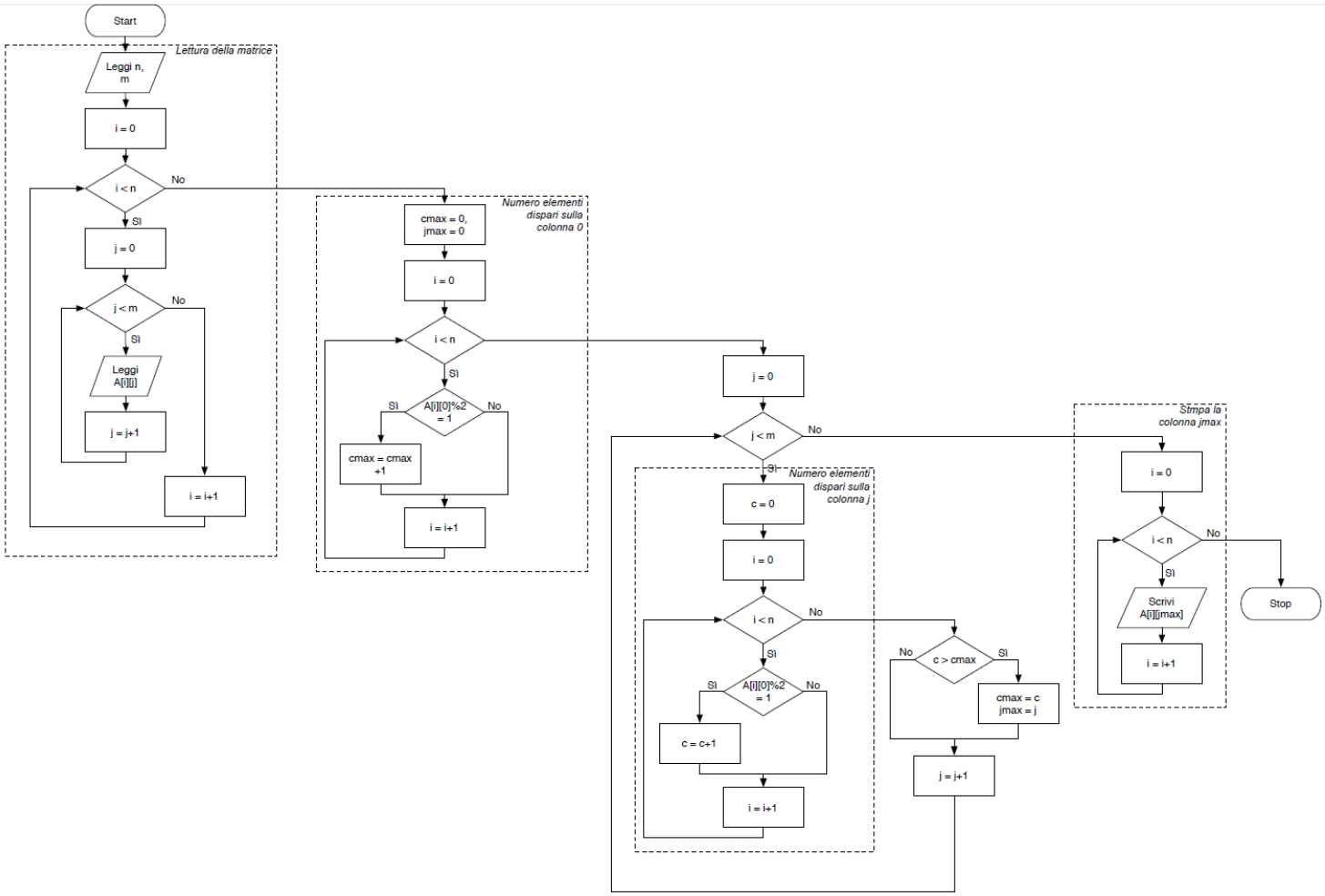
COMING SOON

297. [DIA] Funzioni per la gestione di una matrice.

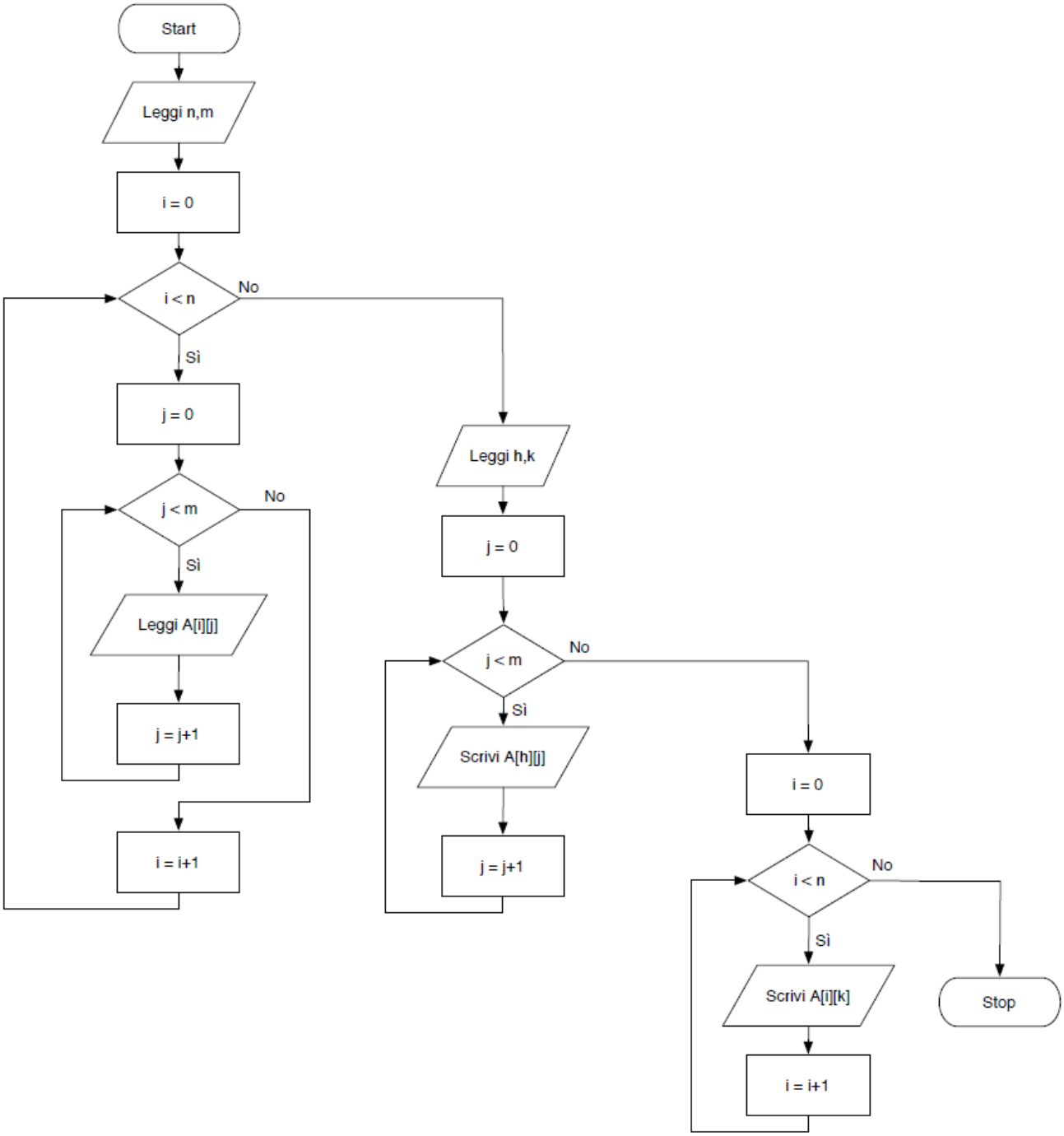


Identificatore	Descrizione	Tipo	Input	Output	Lavoro
Cmax	Costante : dimensione massima delle colonne della matrice	intero			
Rmax	Costante : dimensione massima delle righe della matrice	intero			
Col	colonne della matrice	intero	si		
Rig	righe della matrice	intero	si		
M[][]	matrice da caricare e visualizzare	intero	si	si	
I	**contatore** (indice) per scorrere le righe della matrice	intero			si
J	**contatore** (indice) per scorrere le colonne della matrice	intero			si

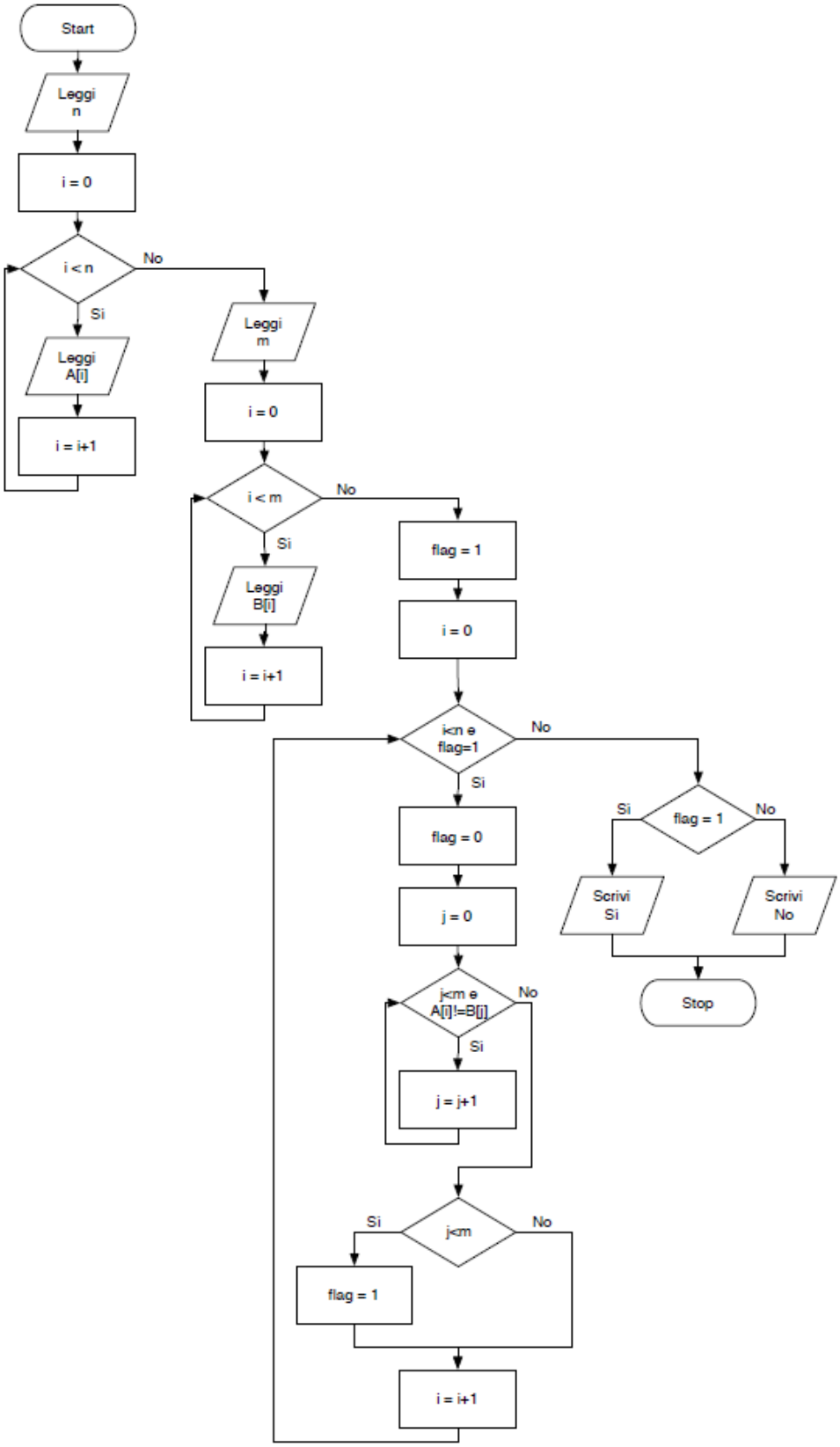
298. [DIA] Letta in input una matrice ***A*** di ***n*** ×***m*** numeri interi, stampa la colonna con il maggior numero di elementi dispari.



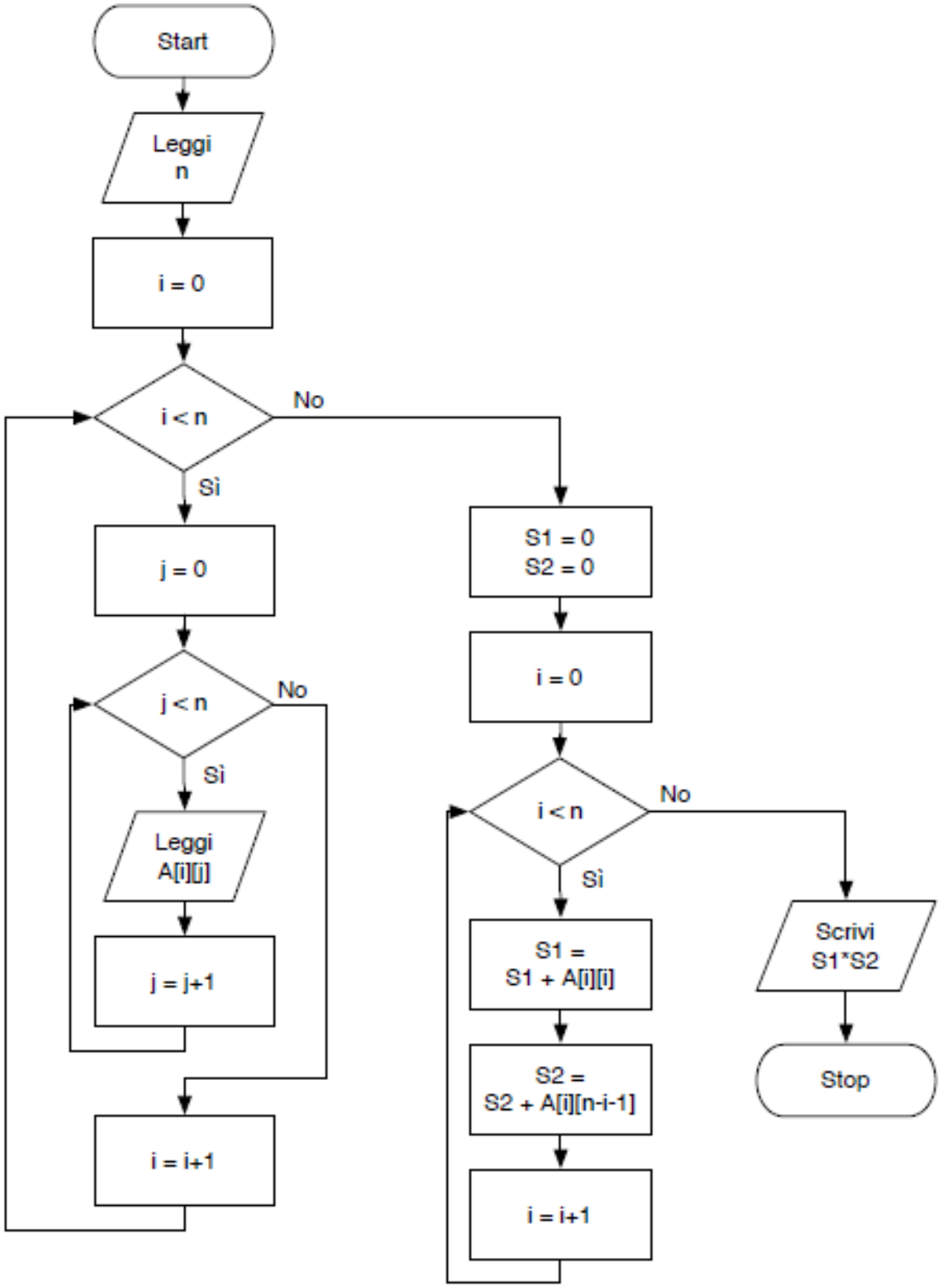
299. [DIA] Leggere una matrice ***n*** ×***m*** di numeri interi; letti in input due interi ***h*** e ***k*** (0 · ***h*** < ***n*** e 0 · ***k*** < ***m***) stampare gli elementi della riga ***h*** e della colonna ***k***.



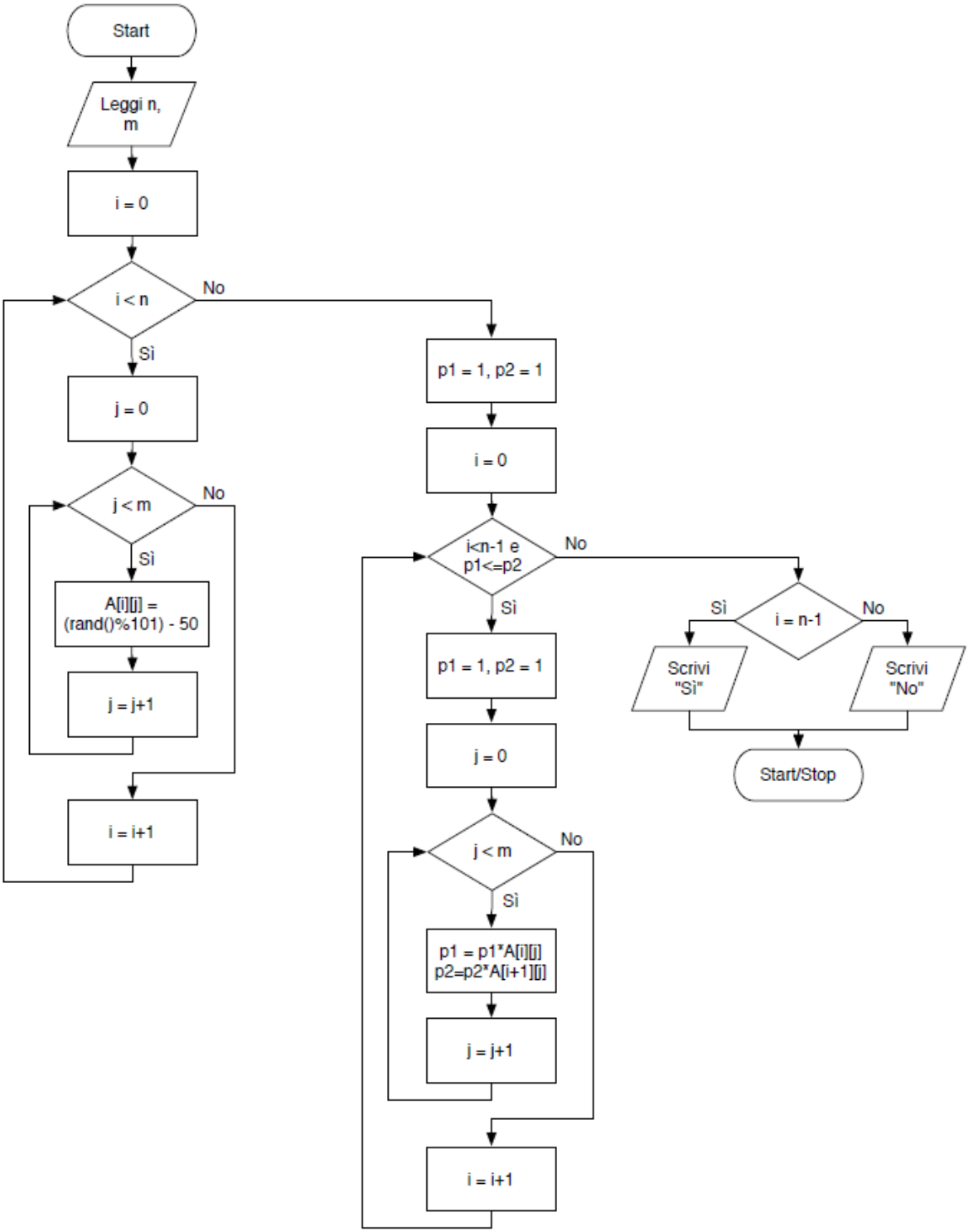
300. [DIA] Letti in input due array di numeri interi (rispettivamente di ***n*** e ***m*** elementi) verifica se tutti gli elementi del primo sono contenuti nel secondo (gli elementi di ***A*** = (1, 2,3) sono contenuti in ***B*** = (8, 1, 3, 7, 5,2,6)).



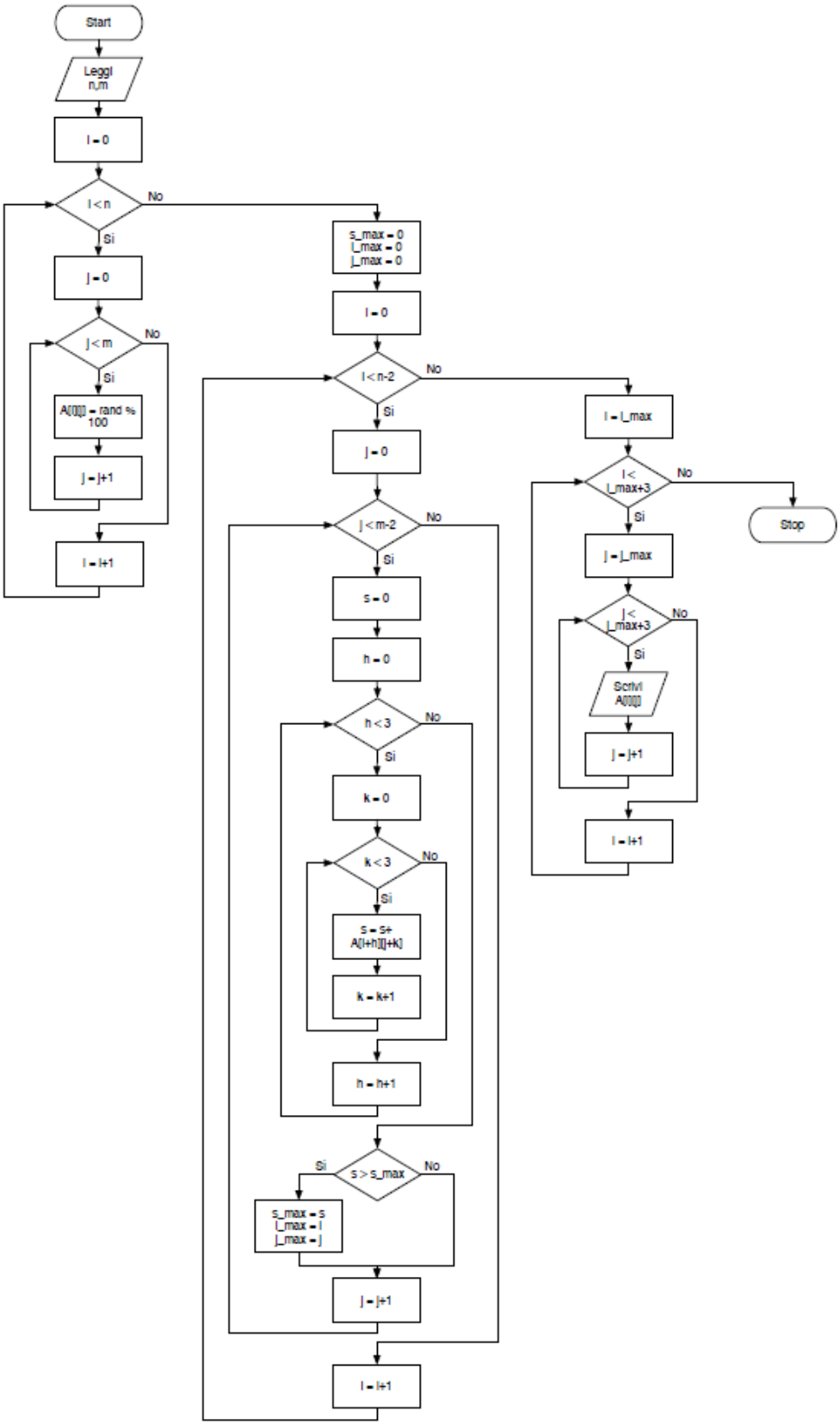
301. [DIA] Letta in input una matrice quadrata ***A*** di ***n*** ×***n*** numeri interi, stampa il prodotto della somma degli elementi della diagonale principale per la somma degli elementi della diagonale secondaria.



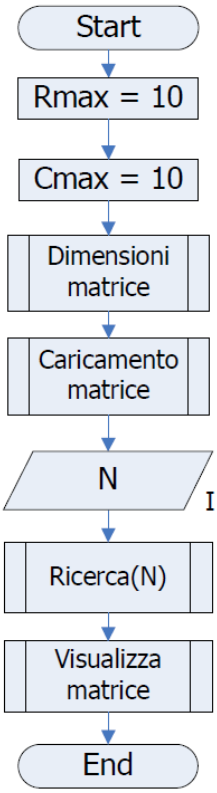
302. [DIA] Letti in input due interi positivi ***n*** e ***m***, generare in modo casuale una matrice ***A*** di ordine ***n*** ×***m*** di numeri interi $a_{i,j}$ tali che $-50 \cdot a_{i,j} \cdot 50$. Verificare se il prodotto di ogni riga è minore o uguale a quello della riga successiva. Stampare "sì" (una sola volta) se la condizione è verificata, stampare "no" (una sola volta) altrimenti.

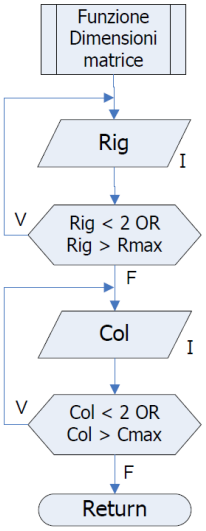


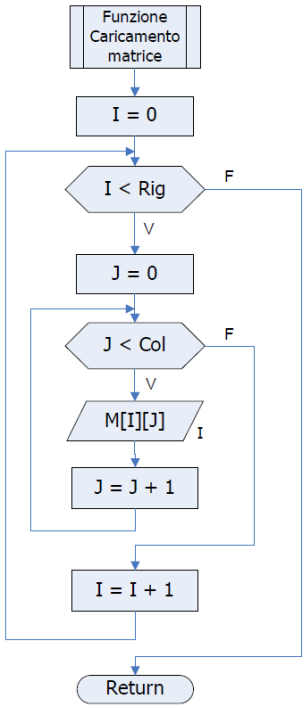
303. [DIA] Generare una matrice di ***n*** righe ed ***m*** colonne (***n***,***m*** > 3) di numeri interi casuali minori di 100. Stampare la sottomatrice quadrata 3×3 per cui la somma degli elementi sia massima.

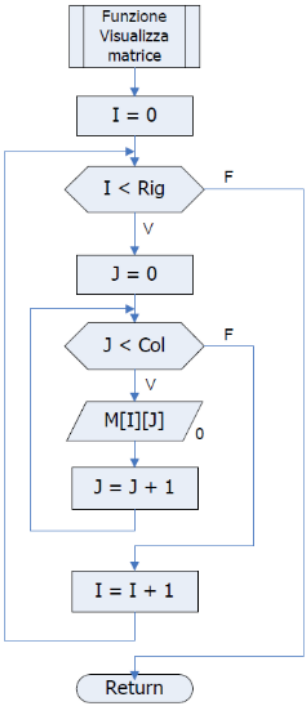


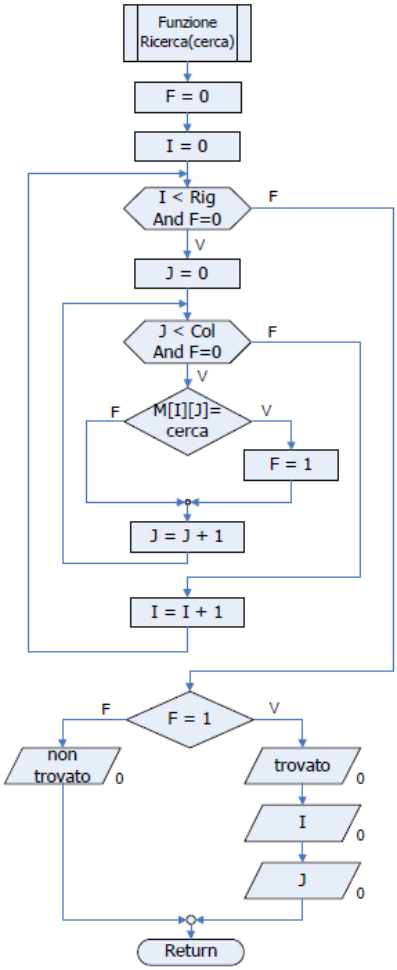
304. [DIA] Ricerca sequenziale di un elemento in una matrice





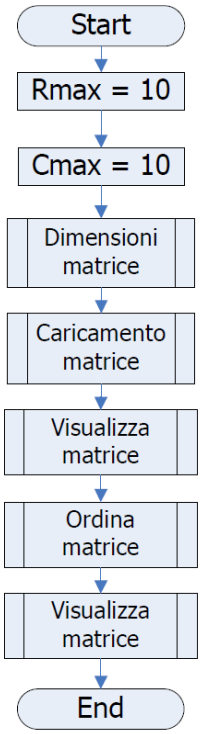


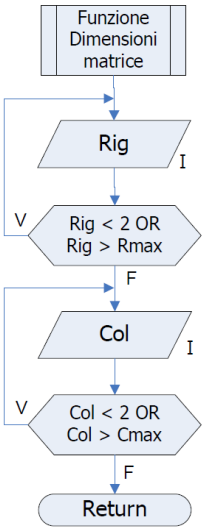


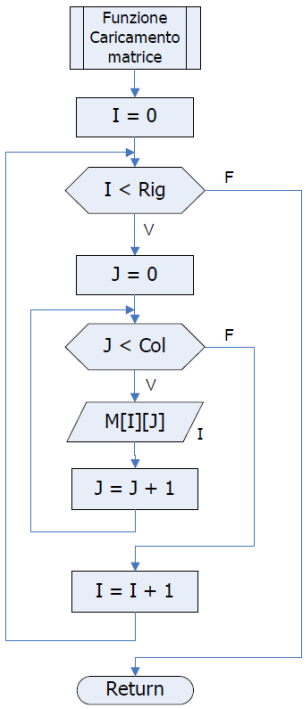


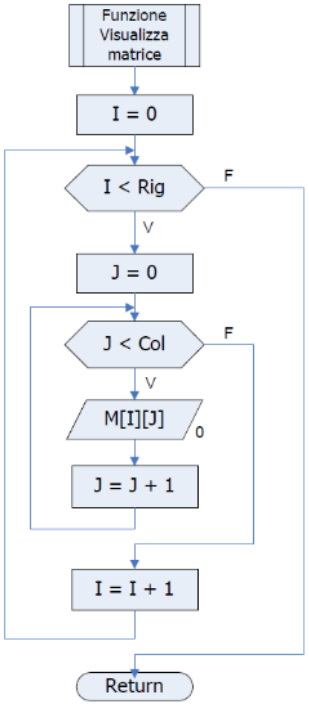
Identificatore	Descrizione	Tipo	Input	Output	Lavoro
Cmax	Costante : dimensione massima delle colonne della matrice	intero			
Rmax	Costante : dimensione massima delle righe della matrice	intero			
Col	colonne della matrice	intero	si		
Rig	righe della matrice	intero	si		
M[][]	matrice da caricare e visualizzare	intero	si	si	
I	**contatore** (indice fisso) per scorrere le righe della matrice	intero			si
J	**contatore** (indice fisso) per scorrere le colonne della matrice	intero			si
N	elemento da ricercare nella matice	intero	si		
F	flag per verificare se l'elemento da cercare nella matrice è stato o non è stato trovato	booleano			si
C	variabile per effettuare lo scambio	intero			si

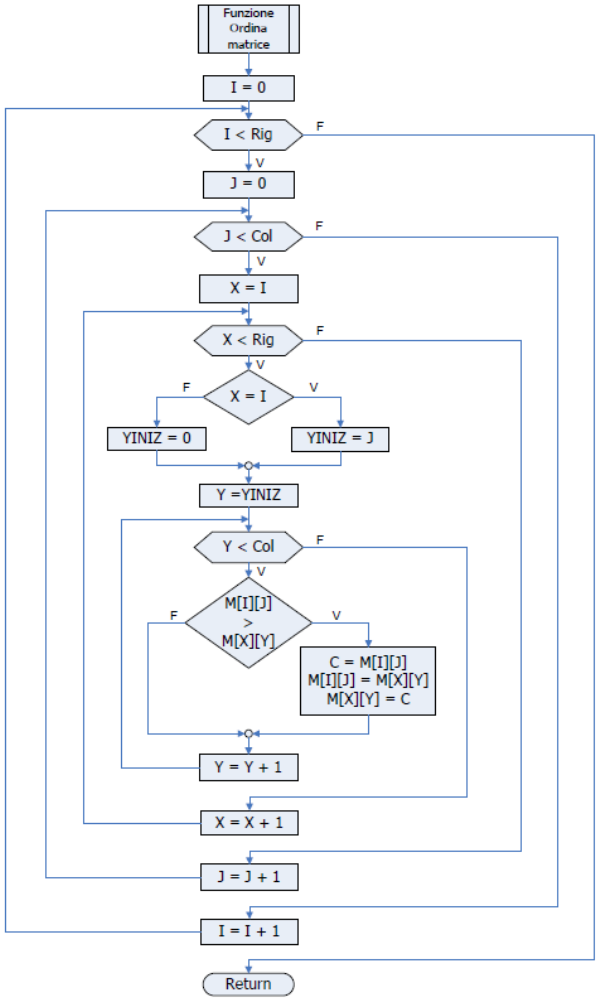
305. [DIA] Ordinamento di una matrice per sostituzione





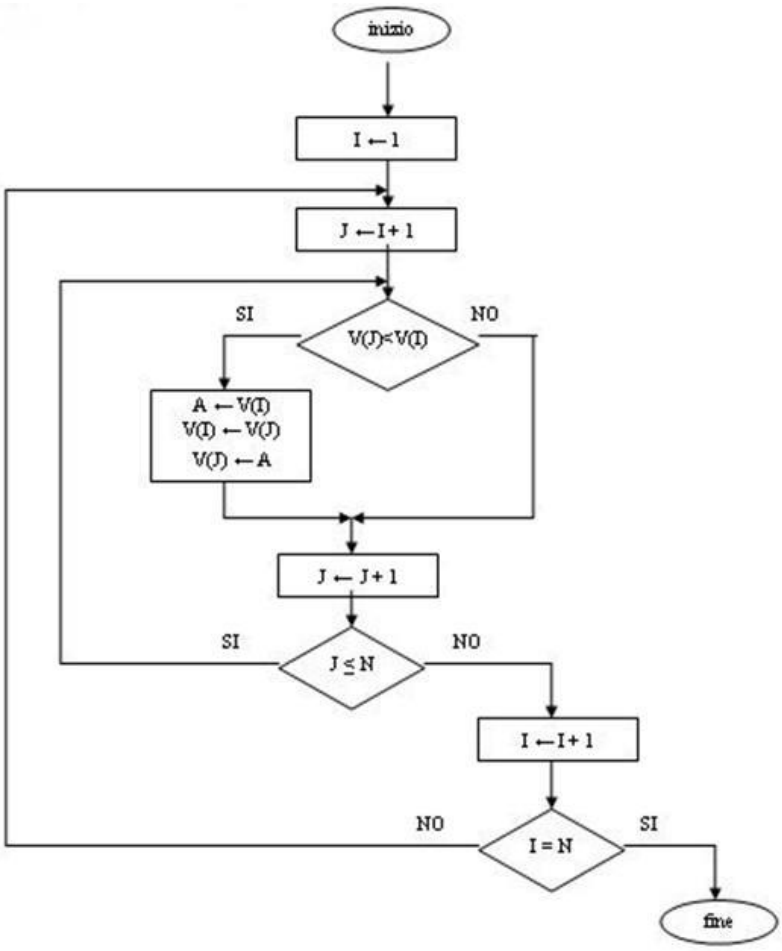




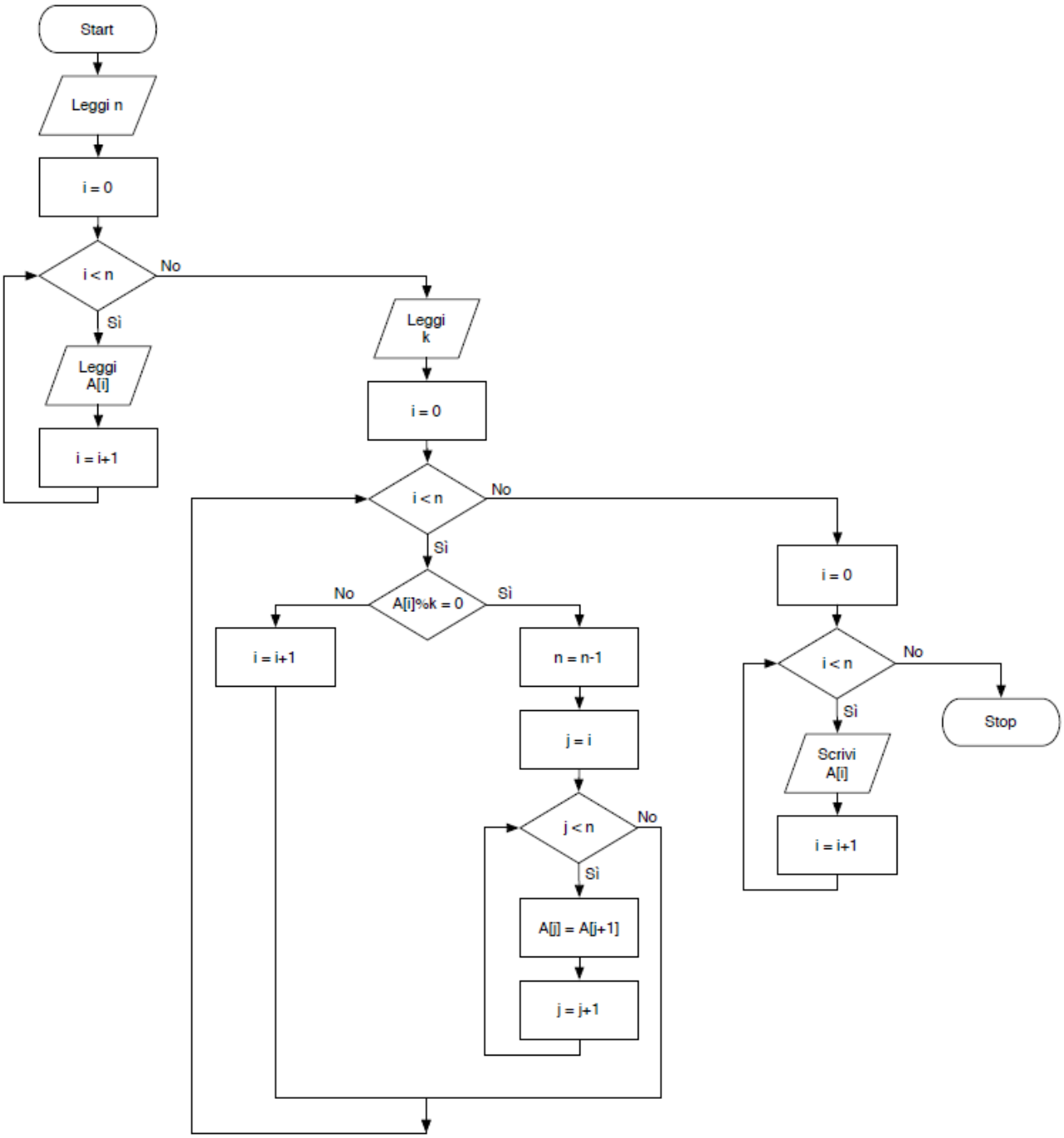


Identificatore	Descrizione	Tipo	Input	Output	Lavoro
Cmax	Costante : dimensione massima delle colonne della matrice	intero			
Rmax	Costante : dimensione massima delle righe della matrice	intero			
Col	colonne della matrice	intero	si		
Rig	righe della matrice	intero	si		
M[][]	matrice da caricare e visualizzare	intero	si	si	
I	**contatore** (indice fisso) per scorrere le righe della matrice	intero			si
J	**contatore** (indice fisso) per scorrere le colonne della matrice	intero			si
X	**contatore** (indice mobile) per scorrere le righe della matrice	intero			si
Y	**contatore** (indice mobile) per scorrere le colonne della matrice	intero			si
YINIZ	variabile utilizzata per attribuire all'indice Y il valore iniziale	intero			si
C	variabile per effettuare lo scambio	intero			si

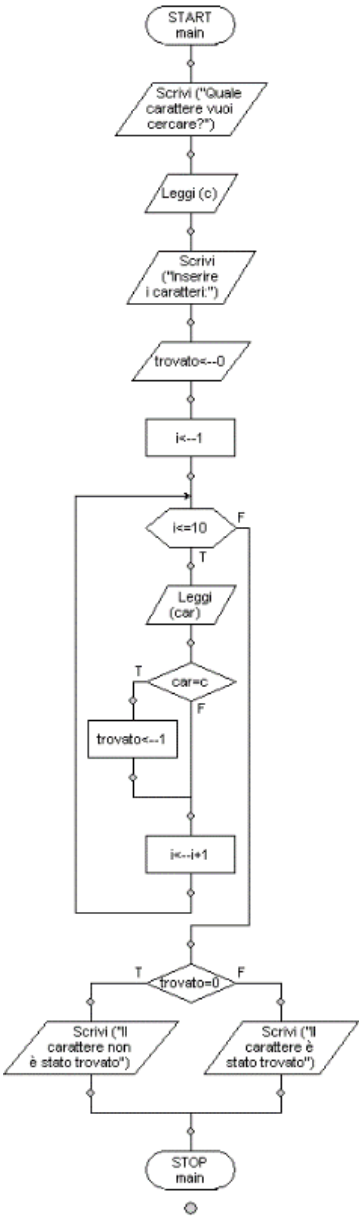
306. [DIA] Ordinamento per scambio di una sequenza di numeri in modo crescente



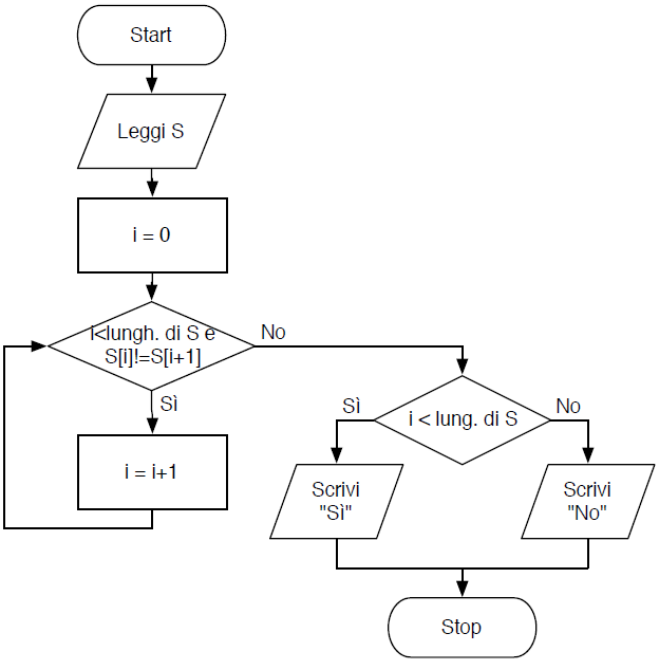
307. [DIA] Letto in input un array di ***n*** numeri interi ed un intero ***k*** > 1, eliminare dall'array tutti i multipli di ***k***, spostando "a sinistra" gli elementi successivi.



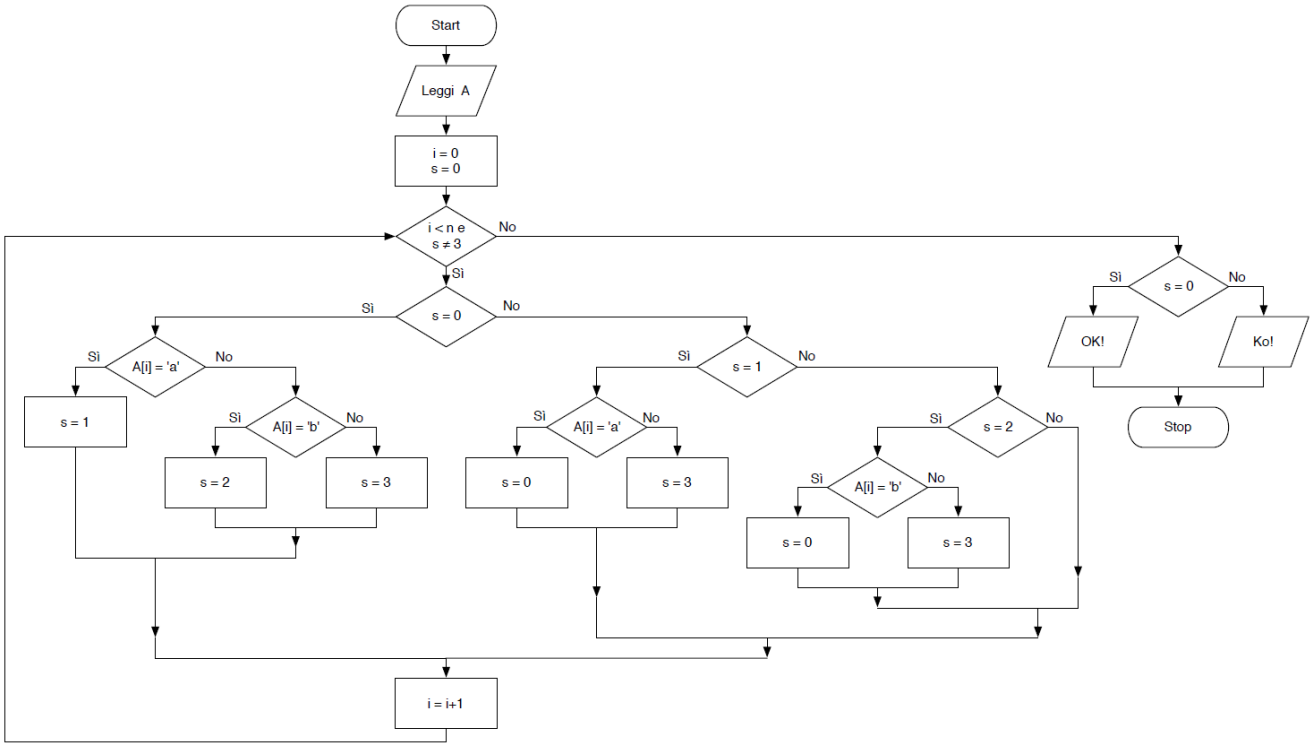
308. [DIA] Dati 10 caratteri e un carattere c dire se c è uno dei caratteri inseriti



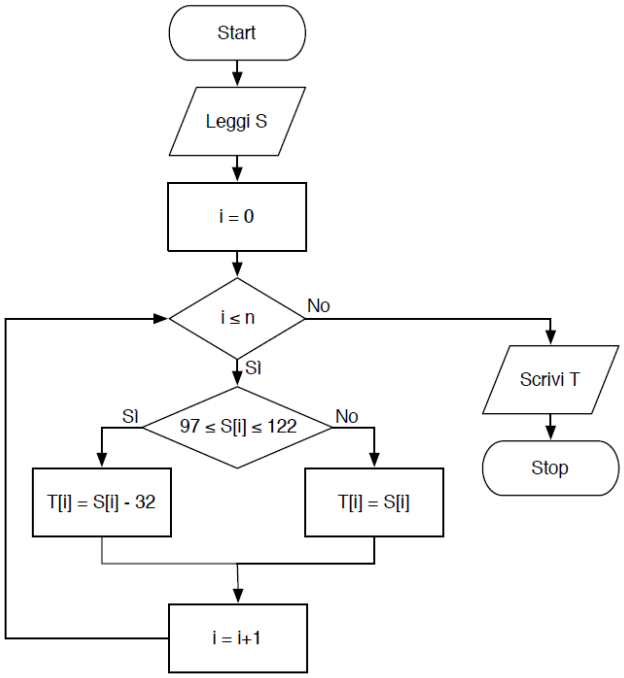
309. [DIA] Letta in input una stringa ***s*** stampare “Sì” se contiene lettere doppie, “No” altrimenti. Ad esempio se ***s*** = “cassa” allora stampa “Sì” perché ci sono le due “s”; se invece ***s*** = “casa” allora stampa “No” perché di “a” ce ne sono due, ma non sono adiacenti).



310. [DIA] Letta una stringa di caratteri verificare se à costituita da sequenze alternate di lunghezza pari di “**a**” e di“**b**”. Ad esempio la stringa “**aaaabbaaaaaabbbbaa**” rispetta la regola descritta nel testo dell’esercizio; viceversa le stringhe “**aabaabbb**” e “**aaaabbccaabbbb**” non la rispettano.



311. [DIA] Letta una stringa ***s***, costruire la stringa ***t*** con la stessa parola di ***s***, ma con caratteri maiuscoli (con array).



312. [DIA] Letta in input una sequenza di ***n*** numeri interi memorizzarla in un array ***A***. Stampare la più lunga sottosequenza di numeri crescenti contigui in ***A***.