

Flowchart-based programming



Andrea Sterbini – sterbini@di.uniroma1.it

Flowcharts

Flowcharts show the possible execution paths of the program

Every program has a single input and single output (initial edge)

An edge can be a sub-flowchart/component with single IN/OUT

- single-thread execution (but what about fork/join?)
- Flowgorithm flowgorithm.org
- Algobuild algobuild.com
- Raptor raptor.martincarlisle.com (with OOP!)
- Visual Logic visuallogic.org
- PseInt pseint.sf.net (in Spanish)

Flowgorithm = Flow-chart + Algorithm

Executable flow-charts

Personalized flow-chart STYLE and COLOURS

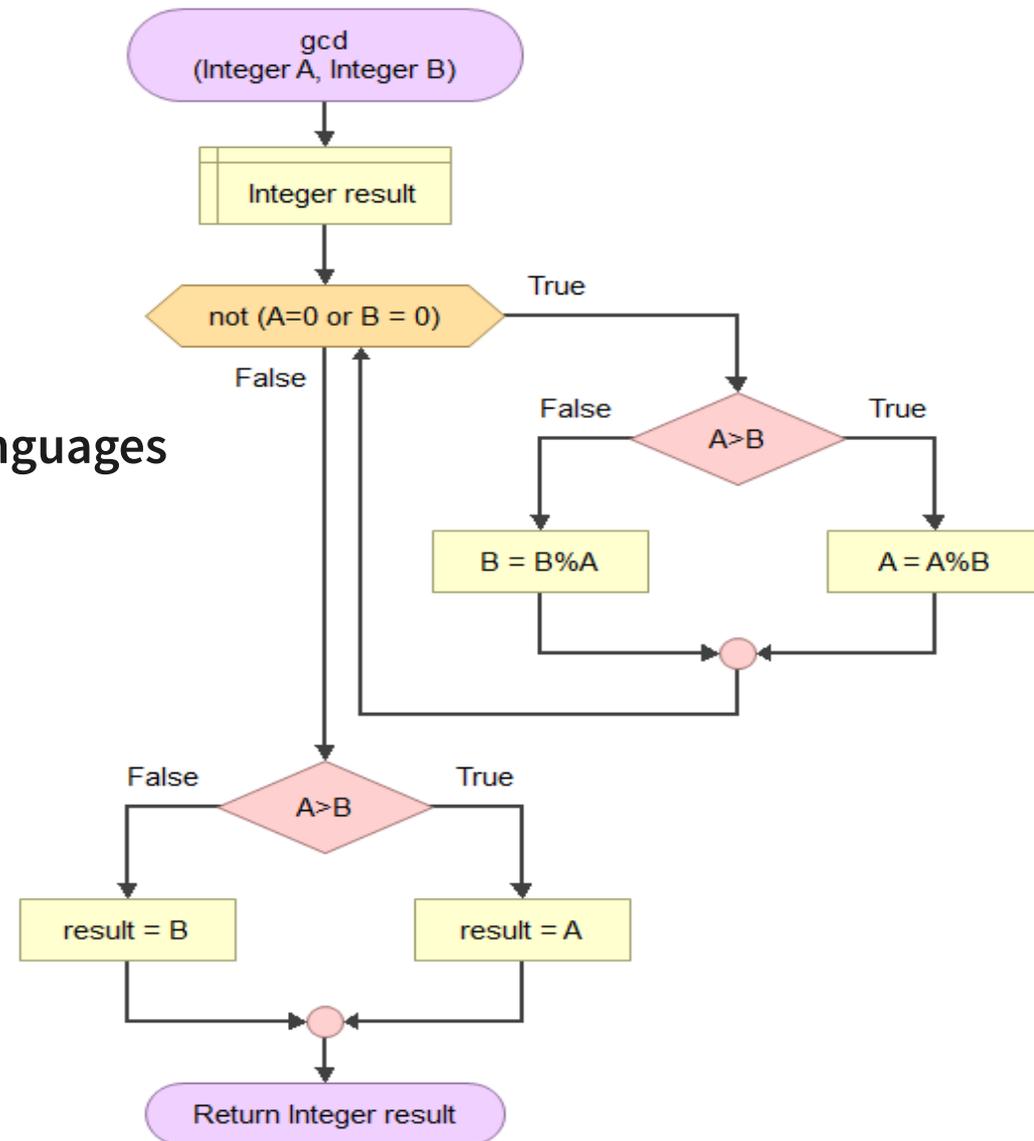
Generate your code in many programming languages

MISSING: loading a program source
and generating its flow-chart
(BUT there are tools for that)

- code2flow.com

- ...

Methods in Computer Science education



Flowgorithm IDE windows

factorial * - Flowgorithm

File Edit Appearance Program Tools Help

factorial

FLOWCHART

```
graph TD; Start([factorial (Integer N)]) --> Init[Integer result]; Init --> Cond{N < 2}; Cond -- False --> Calc["result = N * factorial(N-1)"]; Cond -- True --> Set1["result = 1"]; Calc --> Merge(( )); Set1 --> Merge; Merge --> End([Return Integer result]);
```

Font size set to 8pt. EN

Source Code Viewer

Python

GENERATED PROGRAM

```
0 def factorial(n):
1     if n < 2:
2         result = 1
3     else:
4         result = n * factorial(n - 1)
5
6     return result
7
8 # Main
9 print("Type a positive integer between 0 and 20")
10 n = int(input())
11 result = factorial(n)
12 print(result)
```

Turtle Graphics

TURTLE GRAPHICS

Console

CONSOLE (I/O)

Type a positive integer between 0 and 20

7

Variable Watch

STACK

factorial	
N	5
factorial	
N	6
result	Uninitialized
factorial	
N	7
result	Uninitialized
Main	
N	7
result	Uninitialized

Only simple data types (and 1 dimensional arrays)

T = Integer, Float, String, Boolean

1 dimensional Array of <T>

NO bigintegers (like Python) → you must consider range of possible values

NO lists or dynamic arrays

NO heterogeneous arrays

NO multidimensional arrays

NO objects

NO coroutines

NO function objects

Declare Properties

Declare A Declare Statement is used to create variables and arrays. These are used to store data while the program runs.

Variable Names:
A

Type:
Integer
Integer
Real
String
Boolean

Array?

OK Cancel

Statements

DECLARE/ASSIGN variable

INPUT & OUTPUT text or number

IF-THEN-ELSE

CALL procedure/function

WHILE-do / counted FOR / DO-while
(but NO foreach)

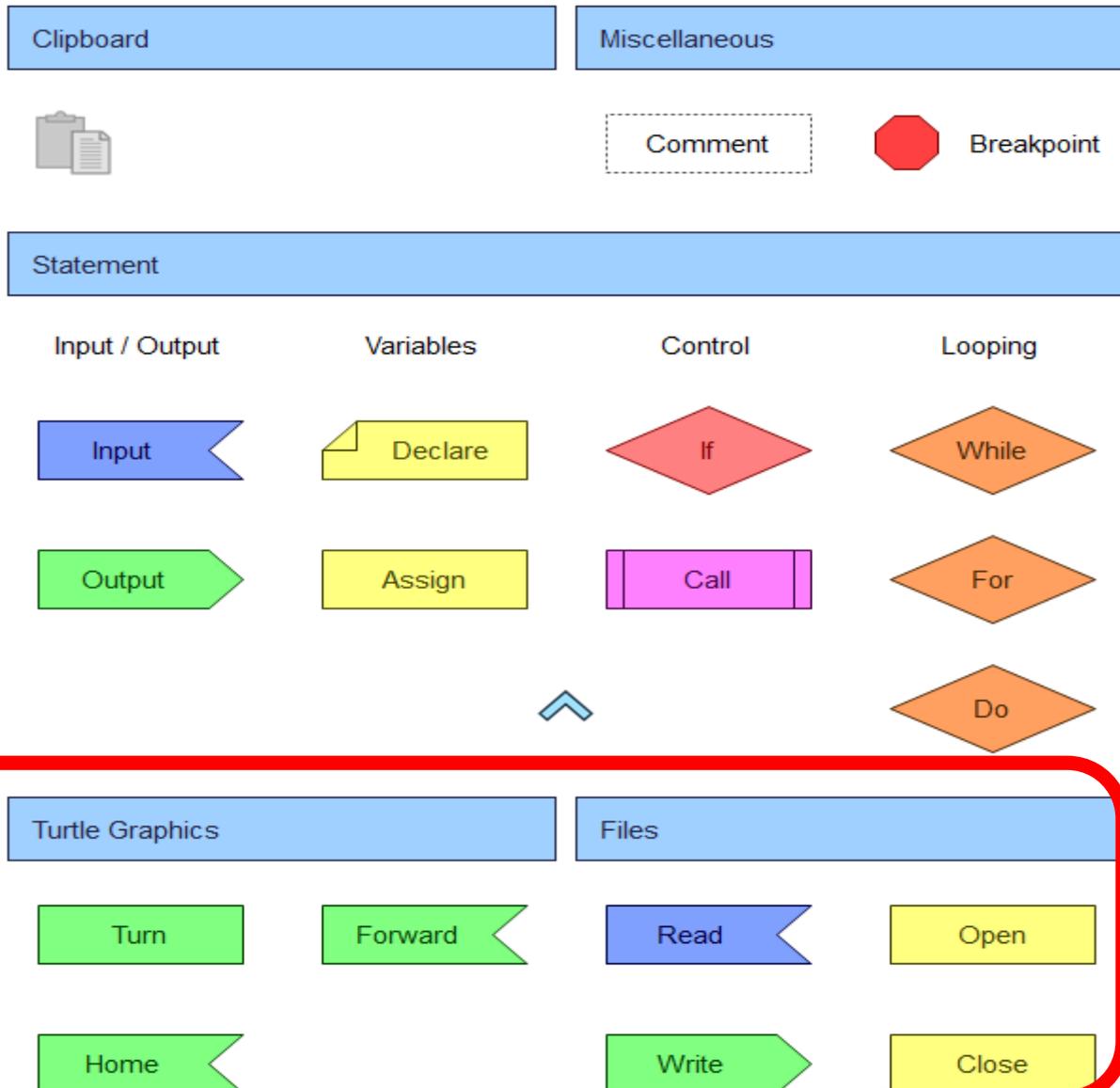
COMMENTS & BREAKPOINTS

TURTLE GRAPHICS (new!!!)

FILES I/O (new!!!)

ONLY 1 open file at a time!!!

Methods in Computer Science ed



Expressions and operators

Function calls

Logic: and, or, not, comparison

Math: +, -, *, /, %, ^, sign, trigonometry, log/pow, random, round

String: concat, len, char(S, i)

Arrays: size

Conversions: char, ascii, int, float, str, round

Precedence of operators in expressions as usual

Control flow

Functions?	YES	
args by reference?	NO	(except for arrays like C)
multiple return values?	NO	(single simple types only)
Recursion?	YES	
ONE entry and ONE exit per function/diagram		
NO early return		(use an IF to skip the rest of the code)
NO break		(use an IF to skip the rest of the code)
Multiple assignments?	NO	
Concurrency/multi threading?	NO	
Events?	NO	
Exceptions?	NO	(errors are shown but you cannot catch them)

Programming style

PROCEDURAL/SEQUENTIAL?	YES	
FUNCTIONAL?	NO	no functions as arguments
STRUCTURED?	YES	
DECLARATIVE?	NO	
EVENT-BASED?	NO	
CONCURRENT?	NO	
MODULARIZATION?	YES	by function/procedure
ANALYSIS		
TOP-DOWN?	YES	
BOTTOM-UP?	NO	
OBJECT-ORIENTED?	NO	no objects

Debug support

Step-by-step execution (both flow-chart AND generated code)

NOTE: the generated code is NOT executed (only shown)

View Variables content (both simple values and arrays)

Show the Stack content (good to understand recursion)

Breakpoints (then step by step)

Assertions? NO (add if-then by hand)

Exceptions? PARTIAL (some errors are generated, but cannot be handled)

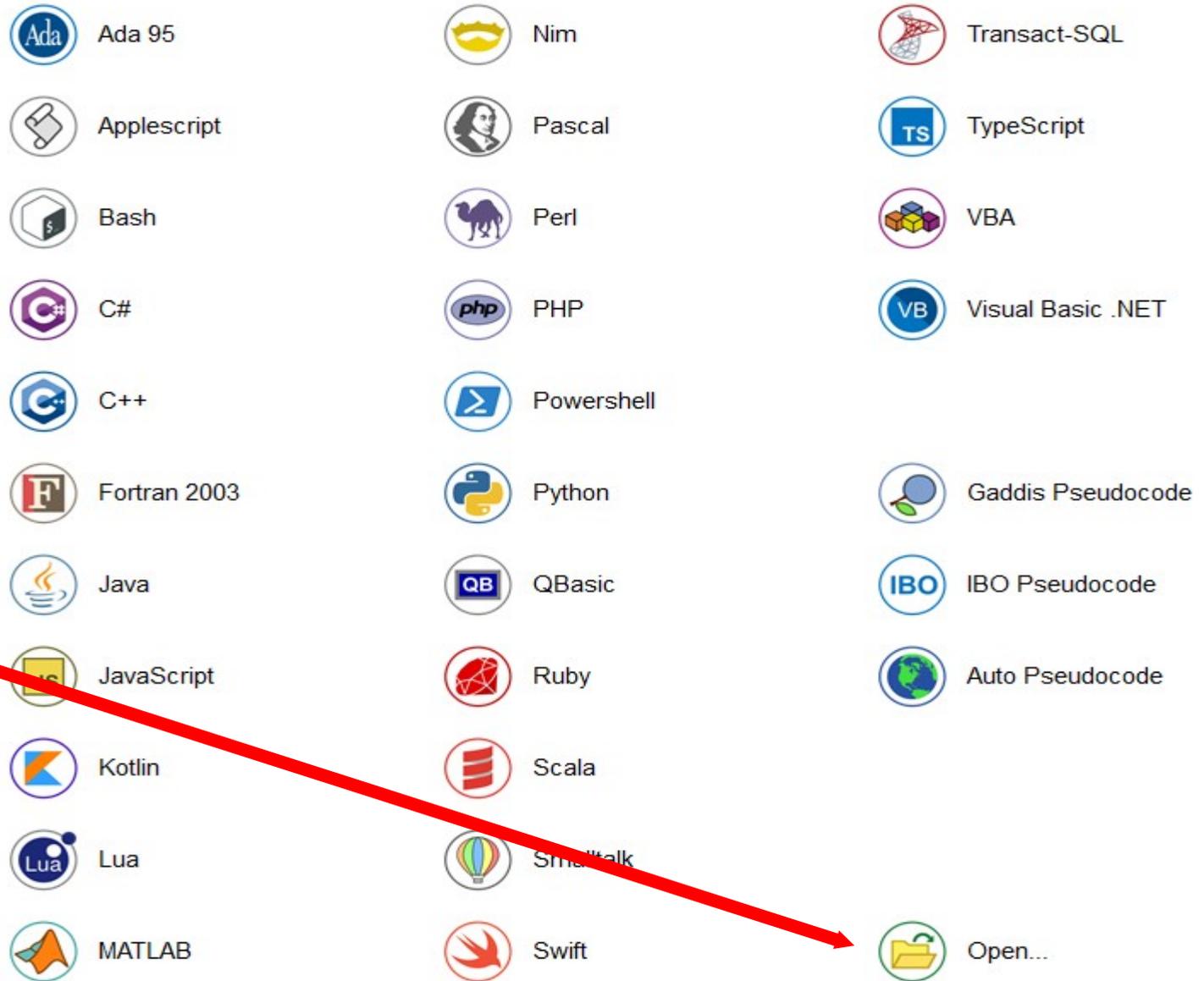
IDE support

Refactoring PARTIAL (cut/paste flowchart into new functions)

Code is generated by templates

Code generation
from flow-charts
to many
programming
languages
(custom also)

Methods in Compute



Example template: Python

A section with some global info (keywords, ext, case-sensitive ...)

The template contains required imports and definitions for some missing functions (you can extend it if you like)

Types are mapped to corresponding Python types

Diagram elements map to corresponding templates

Each Flowgorithm expression operator or intrinsic function is mapped to the corresponding Python one (with precedence levels)

Functions definition and call templates

DEMO

Literate programming / Documentation?

Program properties:

Title, Author, Description

BUT: they are NOT present in the generated code!!!

Comments in the flow-chart

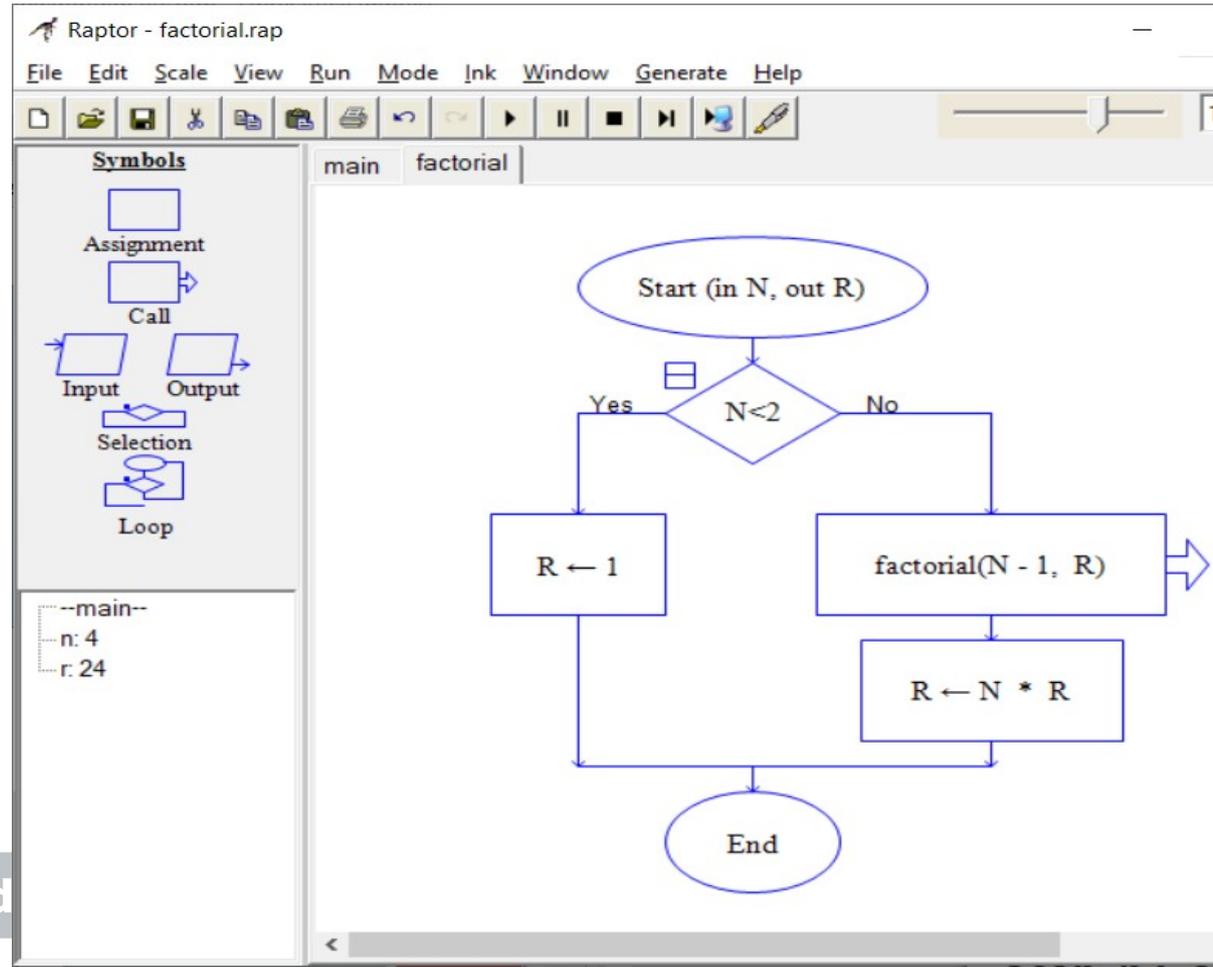
NO free text

Examples

DEMO
(segue)

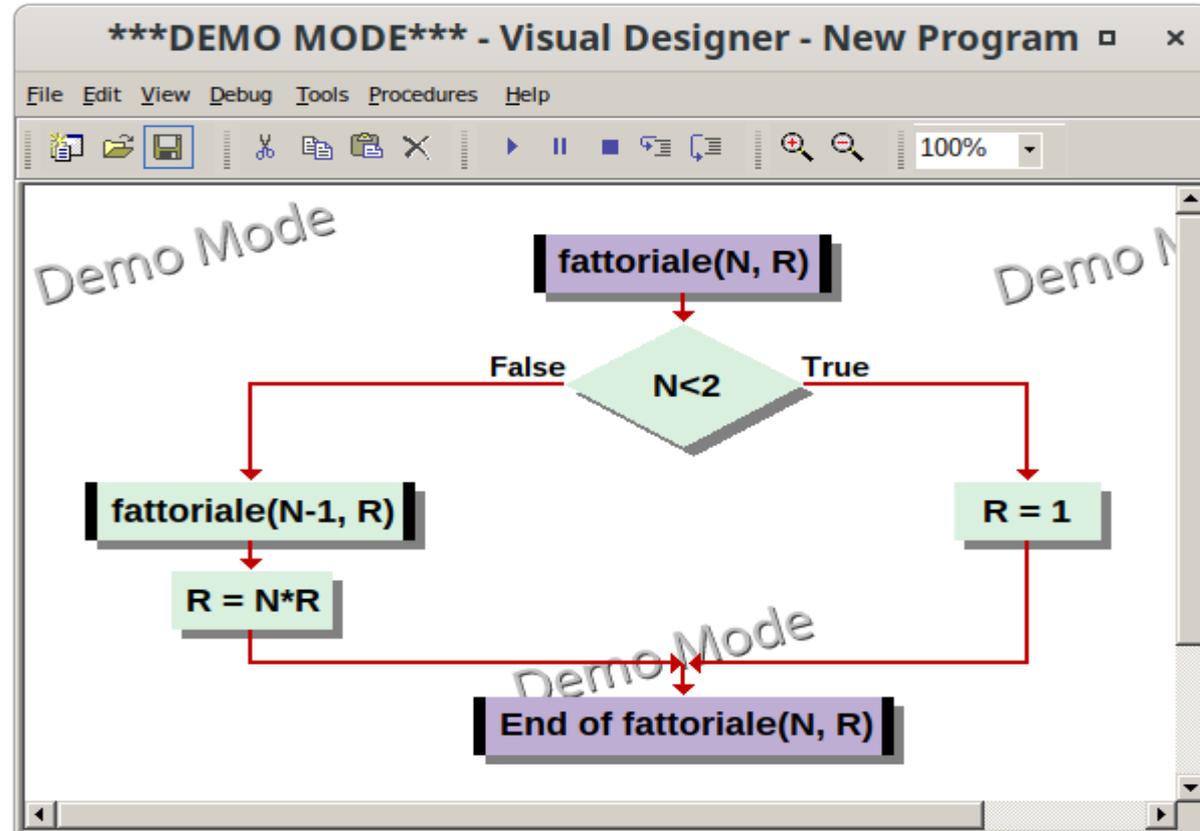
Raptor

Procedures (with IN/OUT args)	YES
Recursion	YES
Functions (procedures + OUT args!!!)	NO?
<u>OOP</u>	<u>YES</u>
<u>Sub-charts</u>	<u>YES</u>
Concurrency	NO
Events	NO
Step-by-step debug	YES
Code generation	YES
Ada, C#, C++, Java, VBA	



Visual Logic

Procedures (with IN/OUT args)	YES
Recursion	YES
Functions (procedures + <u>OUT</u> args!!!)	NO?
OOP	NO
Sub-charts	NO
Concurrency	NO
Events	NO
Step-by-step debug	YES
Code generation	YES
Visual Basic + Pascal	



PseInt (Spanish only)

Procedures	YES
Recursion	YES
Functions	YES
OOP	NO
Sub-charts	NO
Concurrency	NO
Events	NO
Step-by-step debug	YES
Code generation	YES
- C, C++, C#, Java, JavaScript, MatLab	
- Pascal, PHP, Python 2/3	
- Qbasic, Visual Basic ...	

Methods in Computer Science edu

The image displays two windows from the PseInt IDE. The top window, titled 'PseInt', shows a code editor with a Pascal-like implementation of a factorial function. The code is as follows:

```
1 Proceso main
2   Definir N Como Entero;
3   Leer N;
4   Escribir fattoriale(N);
5 FinProceso
6
7 SubProceso R <- fattoriale(N)
8   Definir R Como Entero;
9   Si N<2 Entonces
10      R <- 1;
11   SiNo
12      R <- N*fattoriale(N-1)
13   FinSi
14 FinSubProceso
```

The left sidebar of the PseInt window shows a 'Variables' panel with a tree structure: 'Sb fattoriale' containing '42 N' and '42 R', and 'Pro main' containing 'N'. The bottom window, titled 'PSDraw - main', shows a flowchart that visualizes the execution of the 'SubProceso R <- fattoriale(N)' block. The flowchart starts with an oval 'SubProceso R <- fattoriale(N)', followed by a rectangle 'Definir R Como Entero', then a diamond decision 'N<2'. If 'V' (Verdadero), it goes to a rectangle 'R <- 1'. If 'F' (Falso), it goes to a rectangle 'R <- N*fattoriale(N-1)'. Both paths merge and lead to an oval 'FinSubProceso'. On the right side of the PSDraw window, there is a vertical sidebar with the text 'Comandos y Estructuras' and 'comandos y estructuras'.

AlgoBuild

Functions	YES
Recursion	YES
Simple data types	
- numbers, strings, 1D arrays	
Complex types	NO
OOP	NO
Concurrency	NO
Events	NO
Step-by-step debug	YES
Code generation	NO
Nice tracing of recursion with indentation	

The screenshot shows the AlgoBuild application window titled `/home/andrea/AlgoBuild/fattoriale.algobuild`. The interface includes a menu bar (File, Modifica, Run, Lingua, Utente, Aiuto) and a toolbar with icons for file operations and execution. The main workspace is divided into three sections:

- Flowchart:** A flowchart for the function `FUNC fattoriale(N)`. It starts with a decision diamond `N<2`. If true (T), it proceeds to `M=1`. If false (F), it proceeds to `R = fattoriale(N-1)`, then `M=N*R`. Both paths merge and lead to `RET M`.
- Code Editor:** Contains the following code:

```
FUNC fattoriale(N)
  IF N<2
    M=1
  ELSE
    CALL R = fattoriale(N-1)
    M=N*R
  END IF
RET M
```
- Execution Trace:** Shows the state of variables during execution:

```
START:main
INPUT: N VALUE: 4.0
4.0
VAR: | N=4.0 |
CALL: fattoriale(4.0)
  VAR: | N=4.0 |
  START:fattoriale
  IF TEST: N<2 RESULT: false
  VAR: | N=4.0 |
  CALL: fattoriale(3.0)
    VAR: | N=3.0 |
    START:fattoriale
    IF TEST: N<2 RESULT: false
    VAR: | N=3.0 |
    CALL: fattoriale(2.0)
```
- VARIABLES:** A list of current variables:

```
VARIABLES:
N=4.0
R=24.0
```

Demo

DEMO