Flowchart-based learning / programming

Andrea Sterbini – sterbini@di.uniroma1.it

Flowcharts

Flowcharts show the possible execution paths of the program

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

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

- single-thread execution (but what about fork/join?)

Many executable flowchart editors exists

- 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 languages

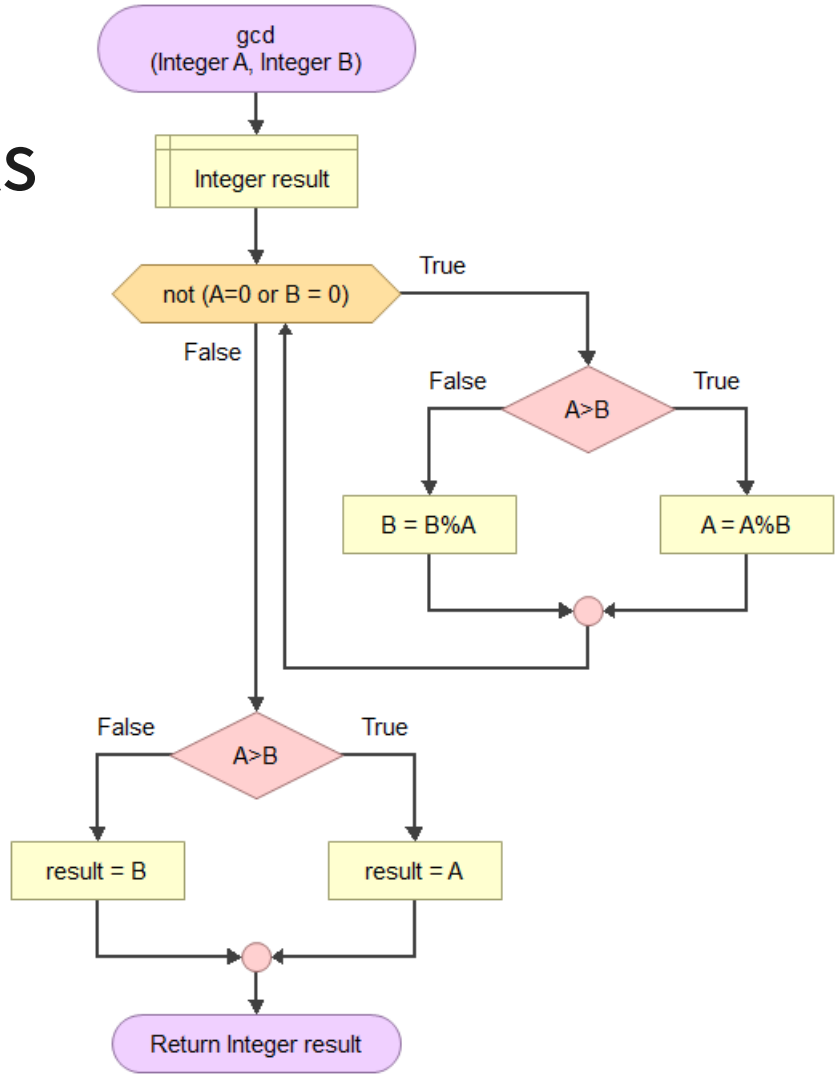
(Spoken or Programmed :-))

MISSING: loading a program source

and generating its flow-chart

(BUT there are tools for that)

- code2flow.com
- ...



Code generation by templates

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


C#


Perl


TypeScript


C++


PHP


VBA


Fortran 2003


Powershell


Visual Basic .NET


Java


Python


JavaScript


QBasic


Gaddis Pseudocode


Lua


Ruby


IBO Pseudocode


MATLAB


Scala


Auto Pseudocode


Nim


Smalltalk


Pascal


Swift


Open...

Example template: Python

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

The program is a template with required imports and definitions for some missing functions (you can extend it if you like)

Types are mapped to corresponding Python types

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

Functions definition and call templates

Diagram elements map to corresponding templates

DEMO

Simple Data types (and arrays)

T = Integer, Float, String, Boolean

1 dimensional Array of <T>

NO bigintegers (Python)

NO lists or dynamic arrays

NO heterogeneous arrays

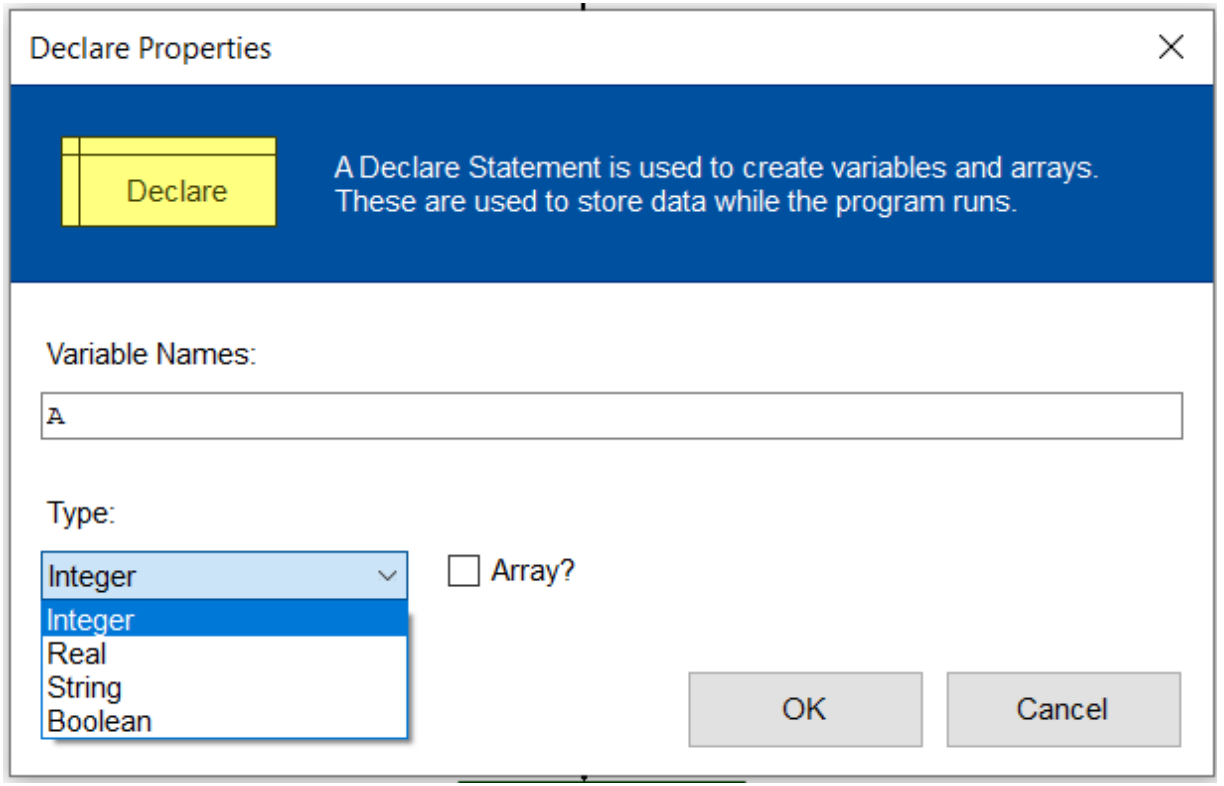
NO multidim. arrays

NO objects

NO coroutines

NO function objects

NO files



Statements

DECLARE variable

ASSIGN variable

INPUT

OUTPUT

IF

CALL procedure/function

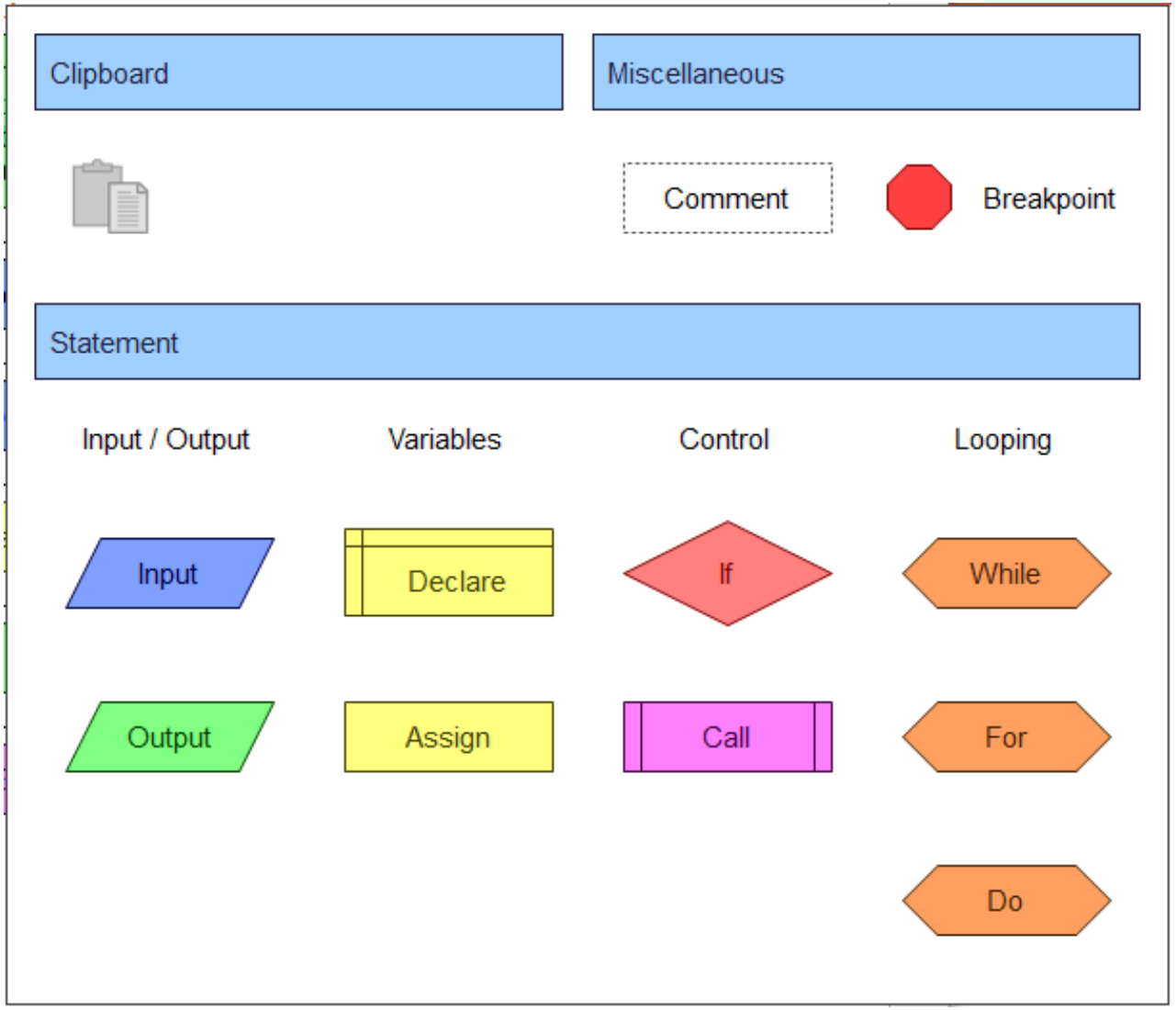
WHILE-do

counted FOR

DO-while

(NO foreach)

COMMENTS & BREAKPOINTS



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

Precedences as usual

Control flow

Functions?	YES	
args by reference?	NO	(except for arrays like C)
multiple return values?	NO	(single simple types only)

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
Recursion?	YES
Exceptions?	NO

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)

Breakpoints

Assertions? (by hand)

Exceptions? NO

IDE support

Refactoring PARTIAL (cut/paste into new functions)

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

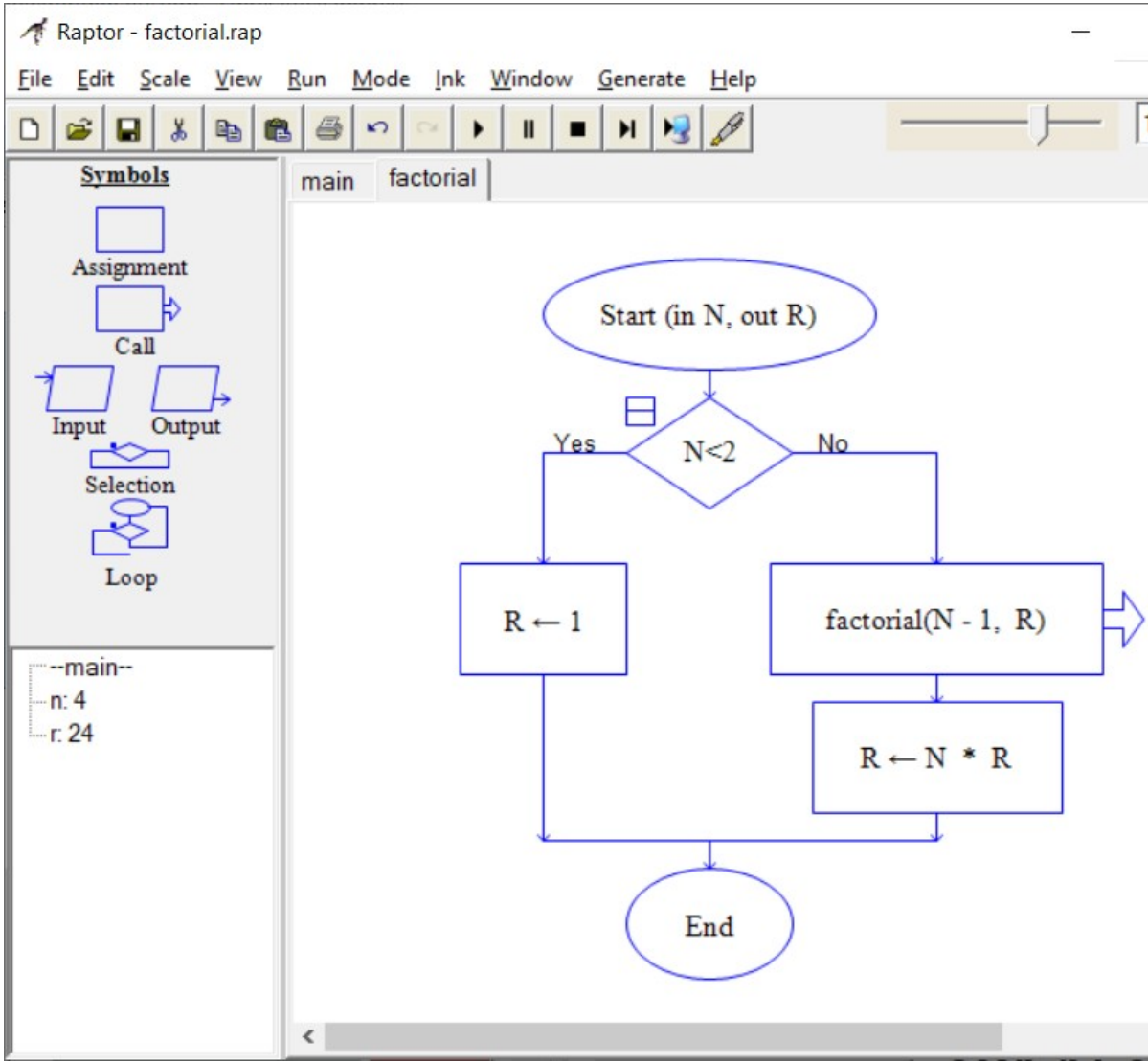
Examples

DEMO

(segue)

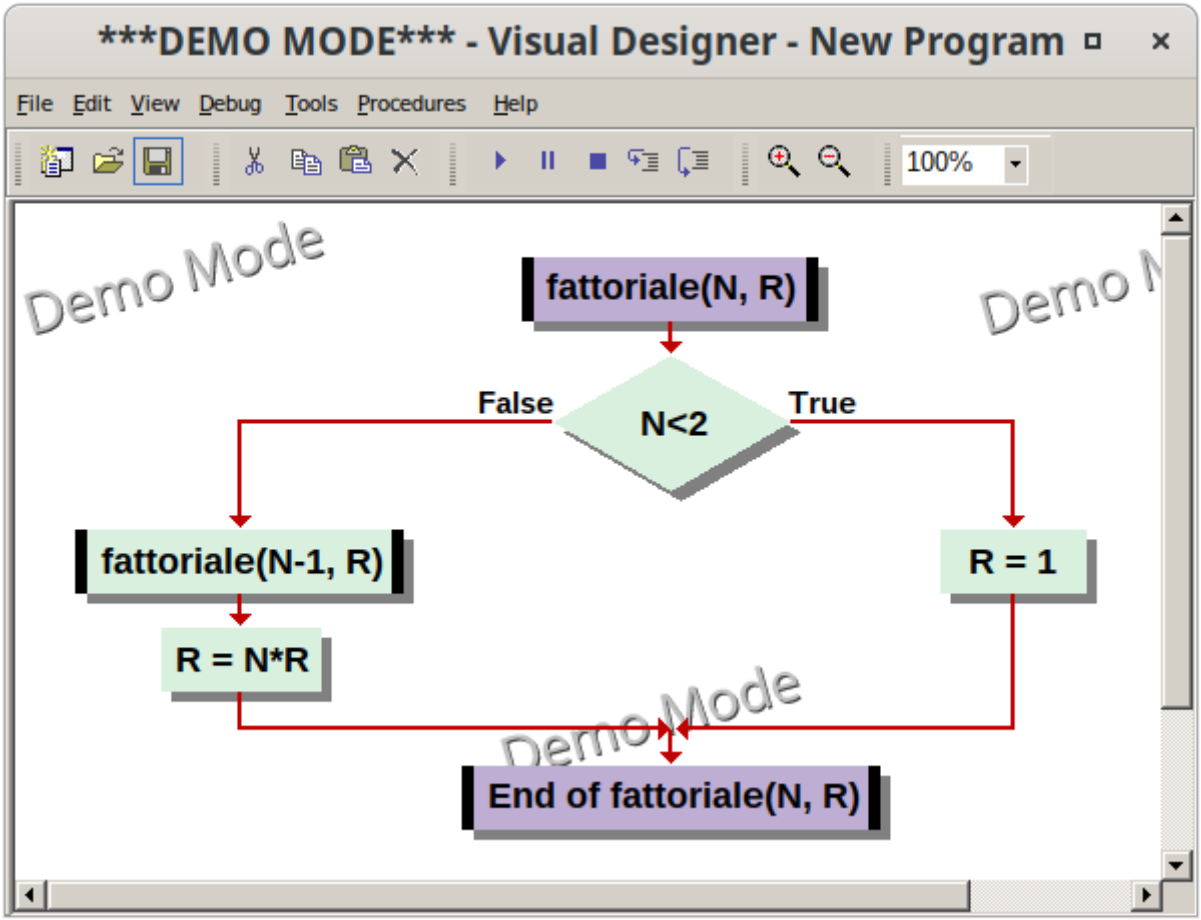
Raptor

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



Visual Logic

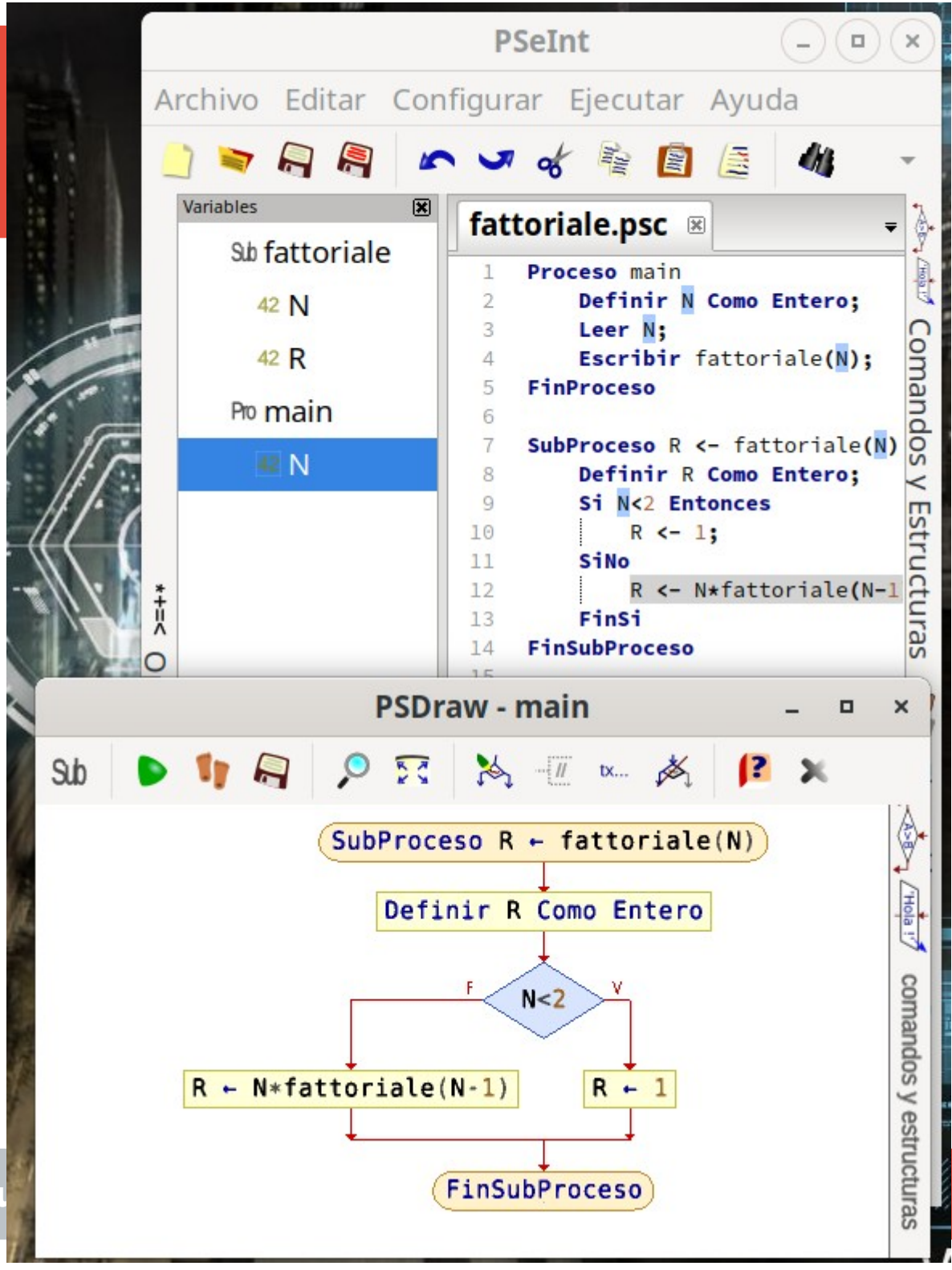
Procedures YES
(with IN/OUT args)
Recursion YES
Functions NO?
(procedures + OUT args!!!)
OOP NO
Sub-charts NO
Concurrency NO
Events NO
Step-by-step debug YES
Code generation YES
VB + Pascal



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 ...



AlgoBuild

Functions YES

Recursion YES

Simple data types

- numbers, strings, 1D arrays

Complex types NO

OOP NO

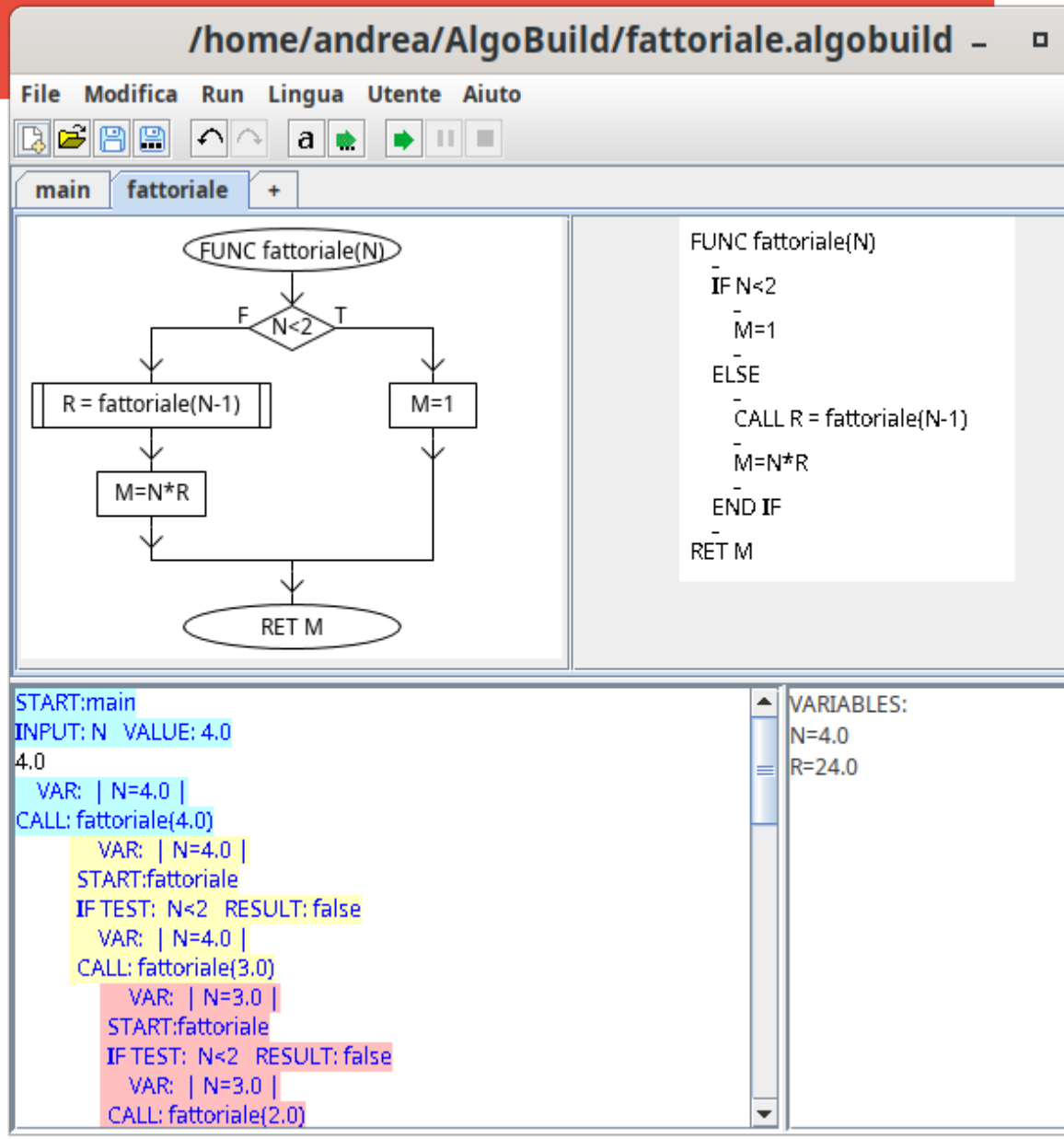
Concurrency NO

Events NO

Step-by-step debug YES

Code generation NO

Nice tracing of recursion



Demo

DEMO