Flowchart-based programming

By Devin Cook of Sacramento State Univ.

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 become a sub-flowchart/component with single IN/OUT

- single-thread execution

Many executable flowchart editors

- Flowgorithm	flowgorithm.org	
- Algobuild	algobuild.com	
- Raptor	raptor.martincarlisle.com	(with OOP!)
- Visual Logic	visuallogic.org	
- PseInt	pseint.SF.net	(in Spanish)
- …		

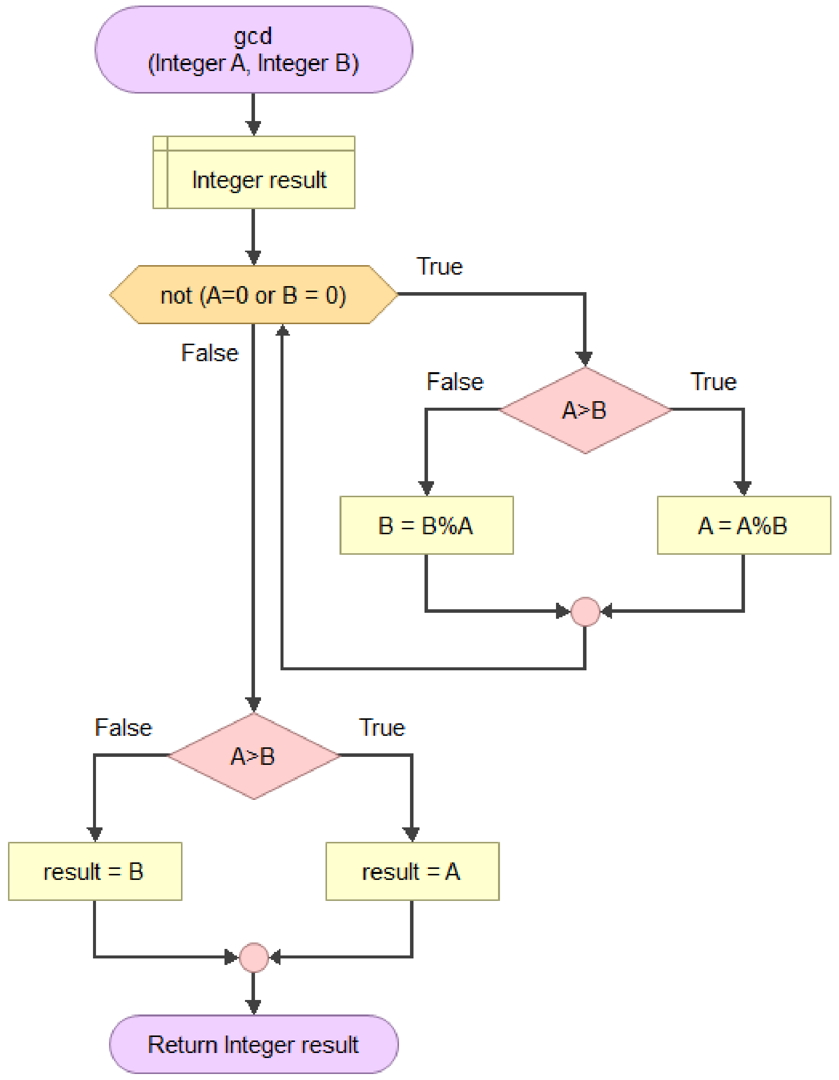
Flowgorithm = Flow-chart + Algorithm

Executable flow-charts

Personalized flow-chart STYLE and COLOURS

Generate your code in many languages

MISSING: load a program
and generate its flow-chart



Code generation by templates

Code generation from flow-charts to many programming languages (even personalized)

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...

Simple Data types (and arrays)

T = Integer, Float, String, Boolean

Array of <T>

NO bigintegers

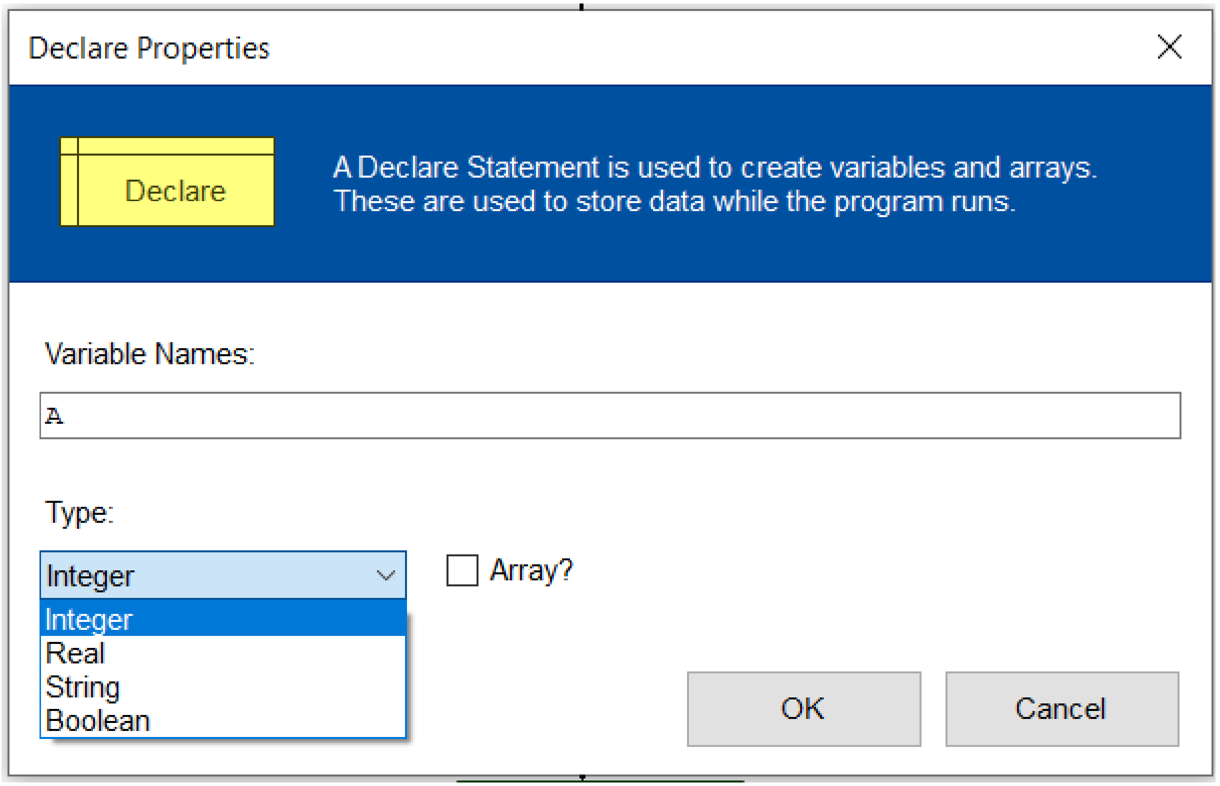
NO lists or dynamic arrays
NO heterogeneous arrays
NO multidim. arrays

NO objects

NO coroutines

NO function objects

NO files



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? ☐

Statements

DECLARE variable
ASSIGN variable

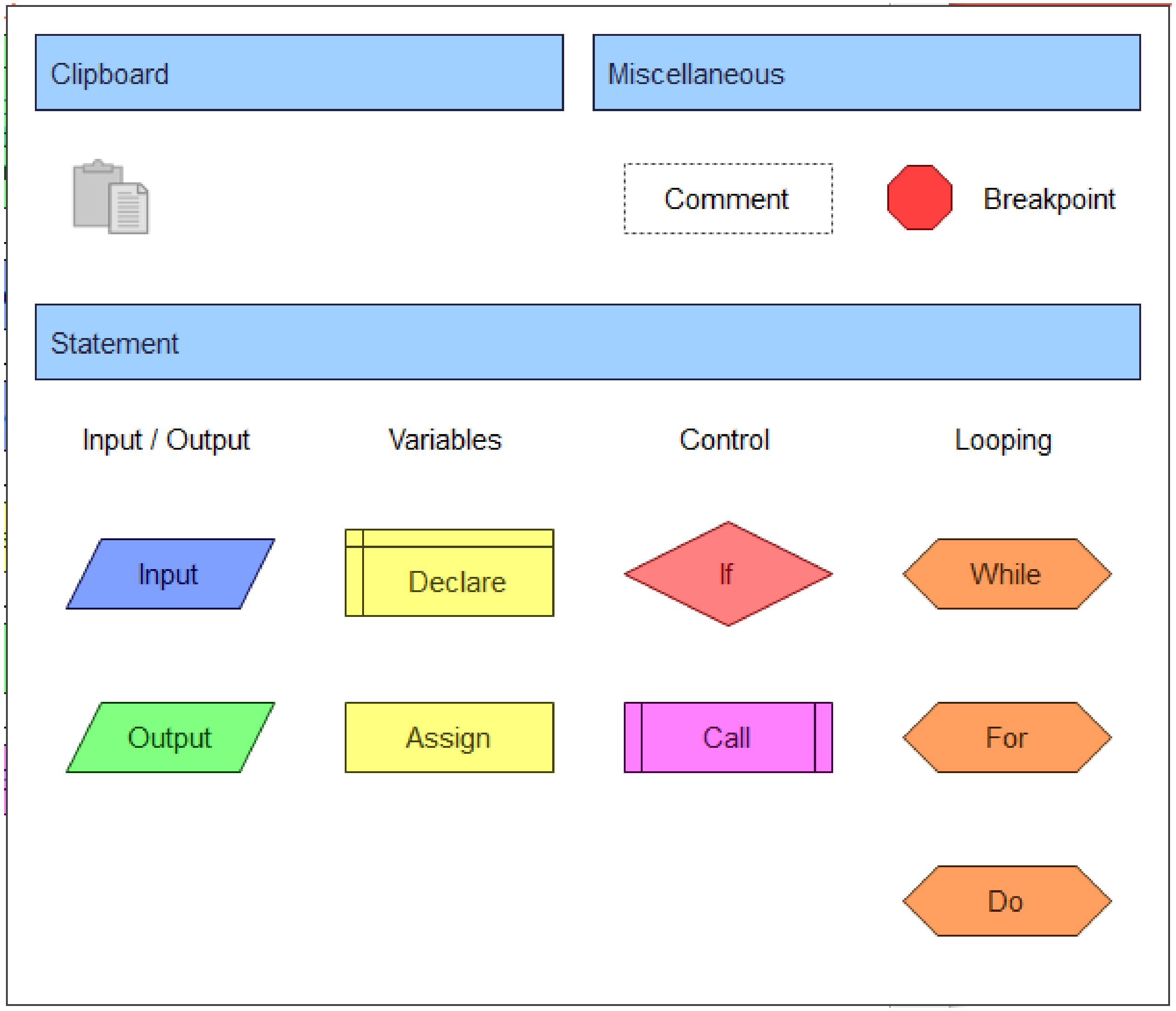
INPUT
OUTPUT

IF

CALL procedure/function

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

COMMENTS



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)
multiple return values?	NO	(single simple types only)

<u>ONE entry</u> and <u>ONE exit</u> 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

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:

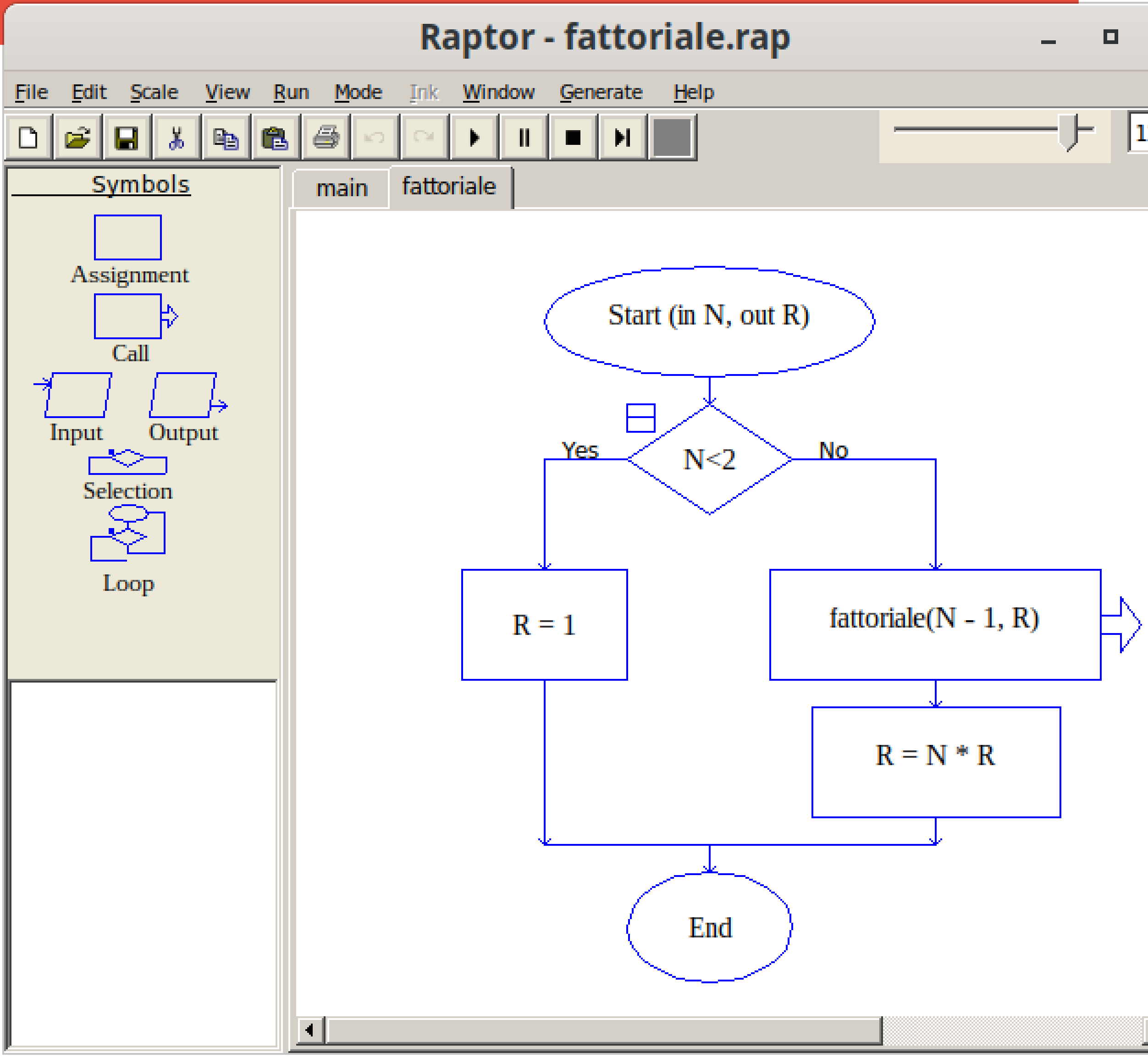
- Title, Author, Description
- BUT: they are NOT present in the generated code!!!

Comments in the flow-chart

NO free text

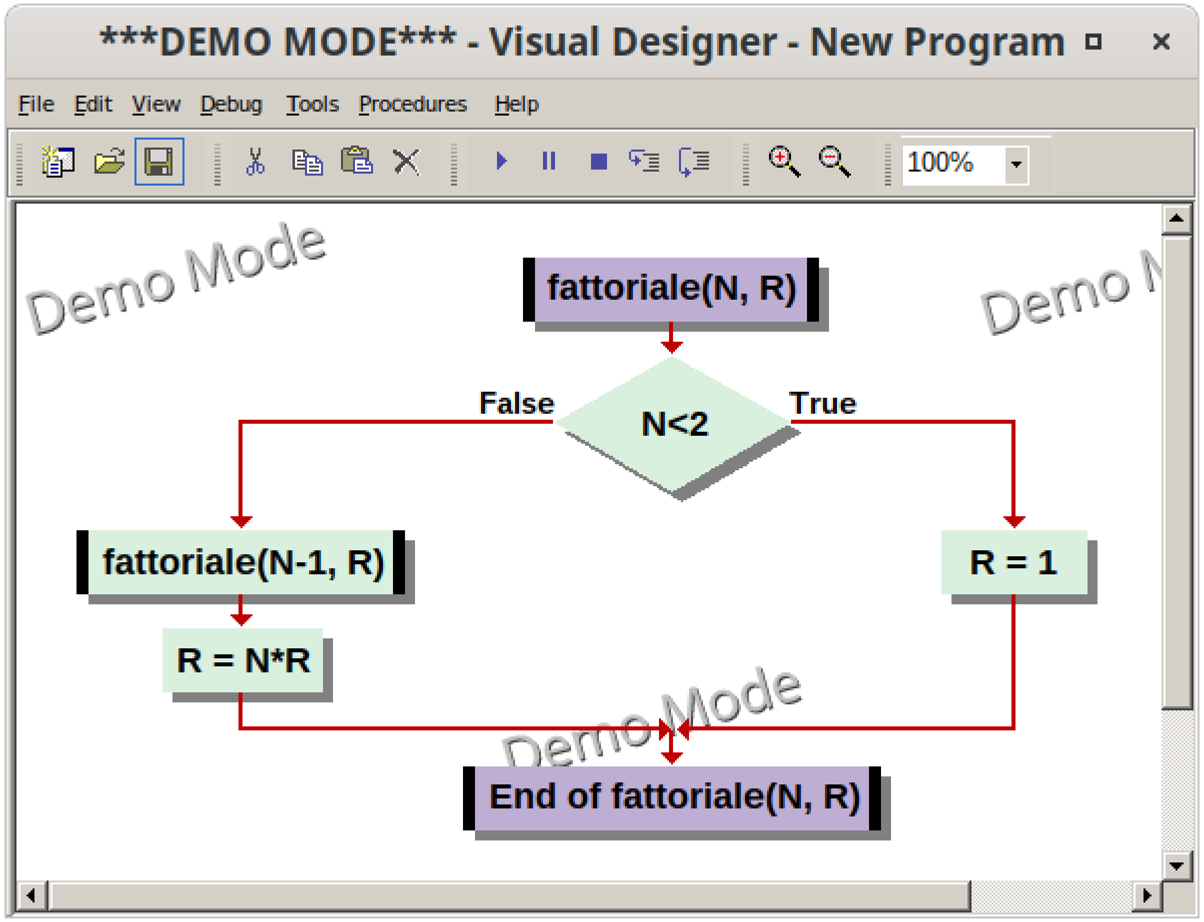
Raptor

Procedures	YES
(with IN/OUT args)	
Recursion	YES
Functions	NO?
(procedures + OUT args)	
<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

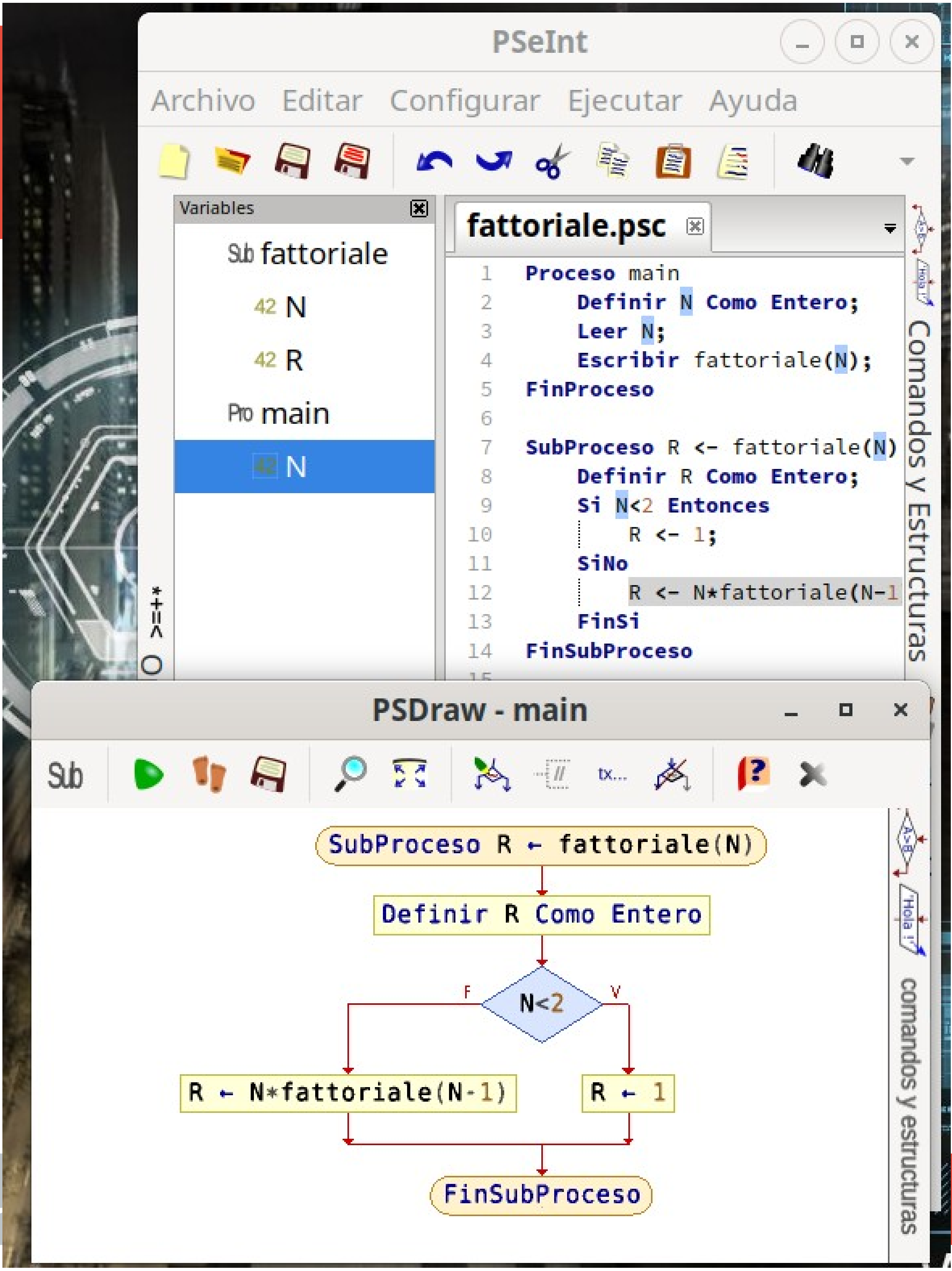
Procedures	YES
(with IN/OUT args)	
Recursion	YES
Functions	NO?
(procedures + OUT args)	
OOP	NO
Sub-charts	NO
Concurrency	NO
Events	NO
Step-by-step debug	YES
Code generation	YES
VB + Pascal	



PSeInt

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

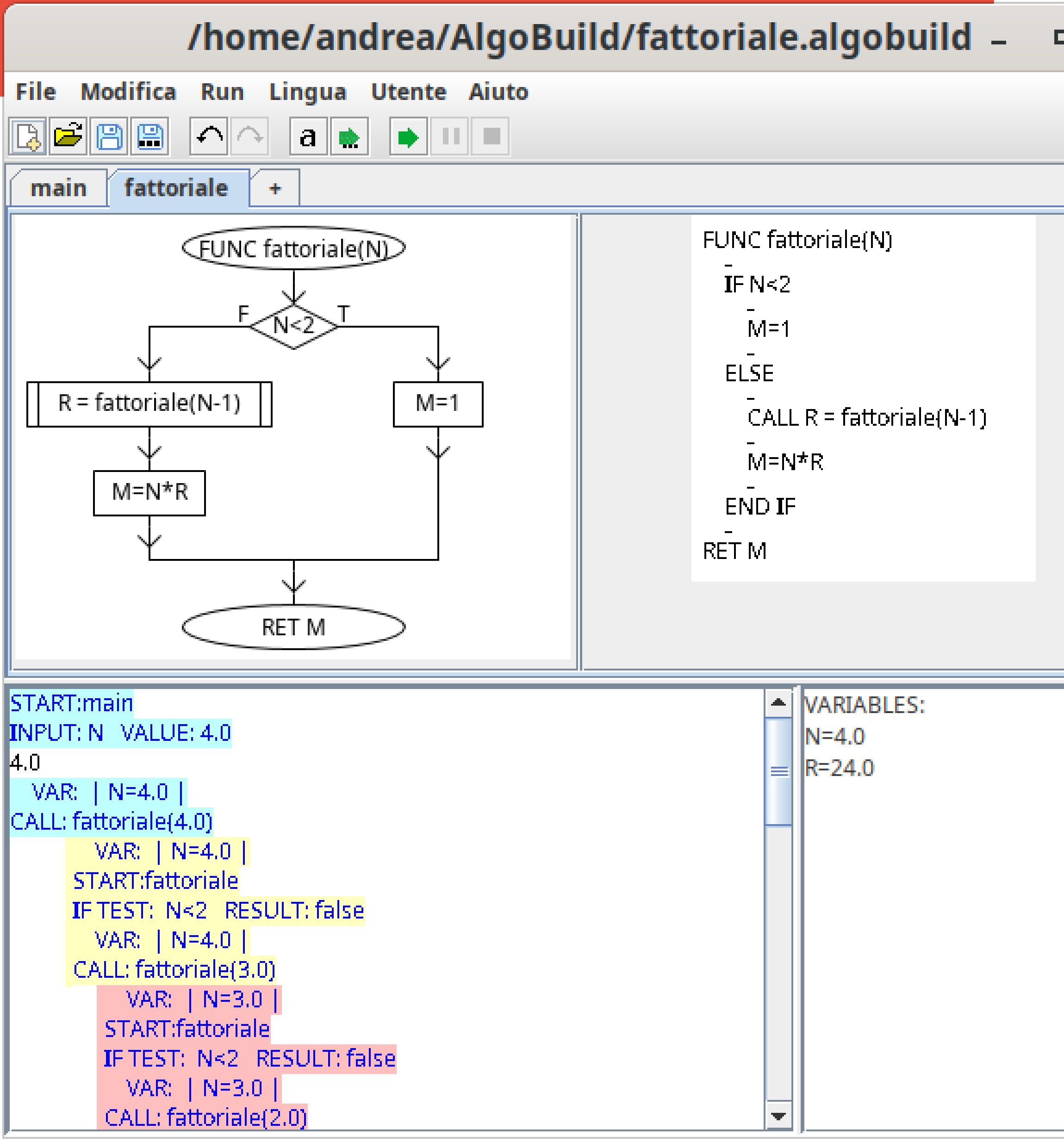


```
Proceso main
    Definir N Como Entero;
    Leer N;
    Escribir fattoriale(N);
FinProceso

SubProceso R <- fattoriale(N)
    Definir R Como Entero;
    Si N<2 Entonces
        R <- 1;
    SiNo
        R <- N*fattoriale(N-1
    FinSi
FinSubProceso
```

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


/home/andrea/AlgoBuild/fattoriale.algobuild
File Modifica Run Lingua Utente Aiuto
main
fattoriale
+
FUNC fattoriale(N)
N<2
F
T
R = fattoriale(N-1)
M=1
M=N*R
RET M

```
FUNC fattoriale(N)
  IF N<2
    M=1
  ELSE
    CALL R = fattoriale(N-1)
    M=N*R
  END IF
RET M
```

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

Demo

DEMO