# Dataflow programming languages: LabVIEW



Andrea Sterbini – sterbini@di.uniroma1.it

# Data-flow: interconnected functional units

Functional units connected by wires

- wires represent data exchanges (i.e. variables)
- they are typed (a different color/shape for each wire)
- multiple data can be aggregated in a single BUS (i.e. a record)
- each functional unit has a default GUI for testing its I/O

Granularity

- functional units can be defined and reused
- circuits/networks can be packaged as new functional sub-units

# LabVIEW

Created by National Instruments to interact with and manage digital data-acquisition electronics and control systems

Modelled over the circuit design and testing metaphor (you draw a circuit)

Each functional unit in the graphic language runs as soon all its input data are available

Multiple cores and threads are used to schedule the parallel execution of multiple active units

Programs are compiled into an intermediate “G” language
(but can also be compiled to native code)

You normally (need to) add explanation boxes to document your ideas

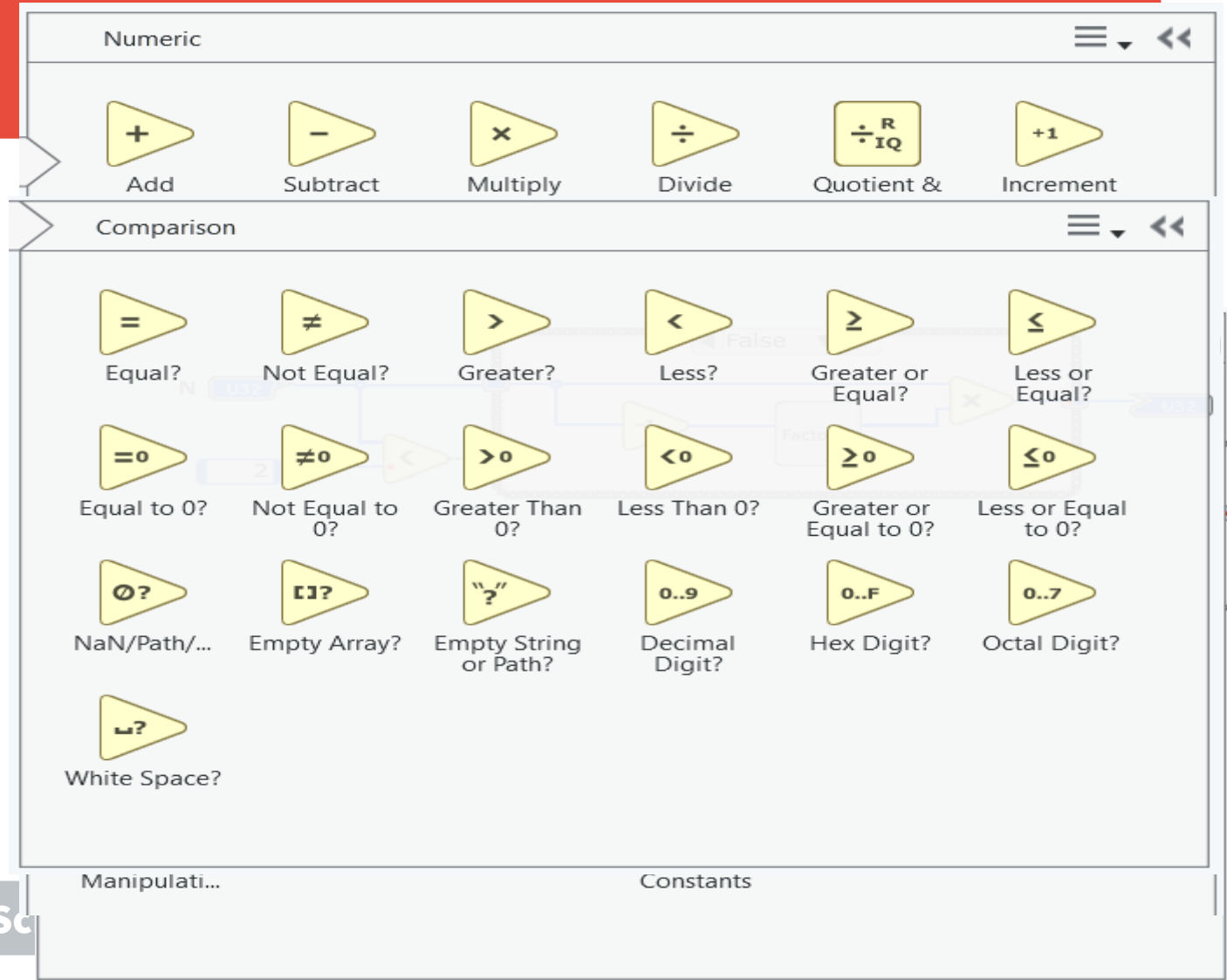
Free LabView Community edition available for personal usage

# Circuit metaphor

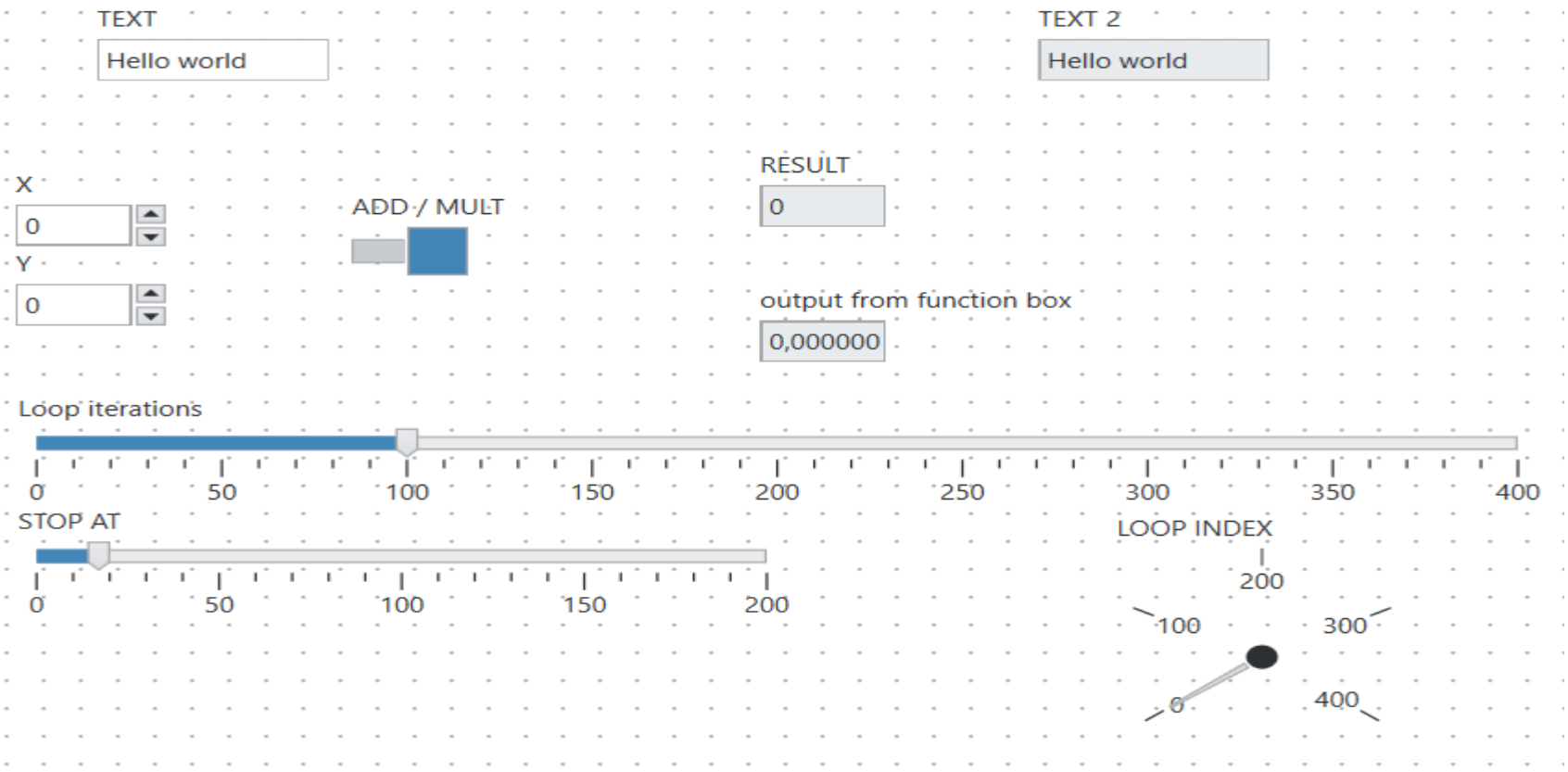
| | | |
|---|---|---|
| PROGRAM | ==> | CIRCUIT |
| VARIABLE | ==> | WIRE |
| FUNCTION | ==> | CIRCUIT COMPONENT (defined with a sub-circuit) |
| ARGUMENTS | ==> | INPUT CONNECTORS |
| RETURN VALUEs | ==> | OUTPUT CONNECTORS |
| IF-THEN-ELSE | ==> | MULTIPLE CIRCUITS (one for each different condition, same I/O) |
| LOOPS | ==> | REPEATING CIRCUITS (with stop conditions & state “wires”) |
| CONCURRENCY | ==> | IMPLICIT in the data-flow execution |

# Functional units

- Many numeric and signal processing elements
- Multiple values can be bundled in buses
- Wires have types



# Each functional unit has a default GUI to test it
# many widgets are available to personalize it (active or read-only)



# Data types describe what a wire could transfer (i.e. the content of the variable)

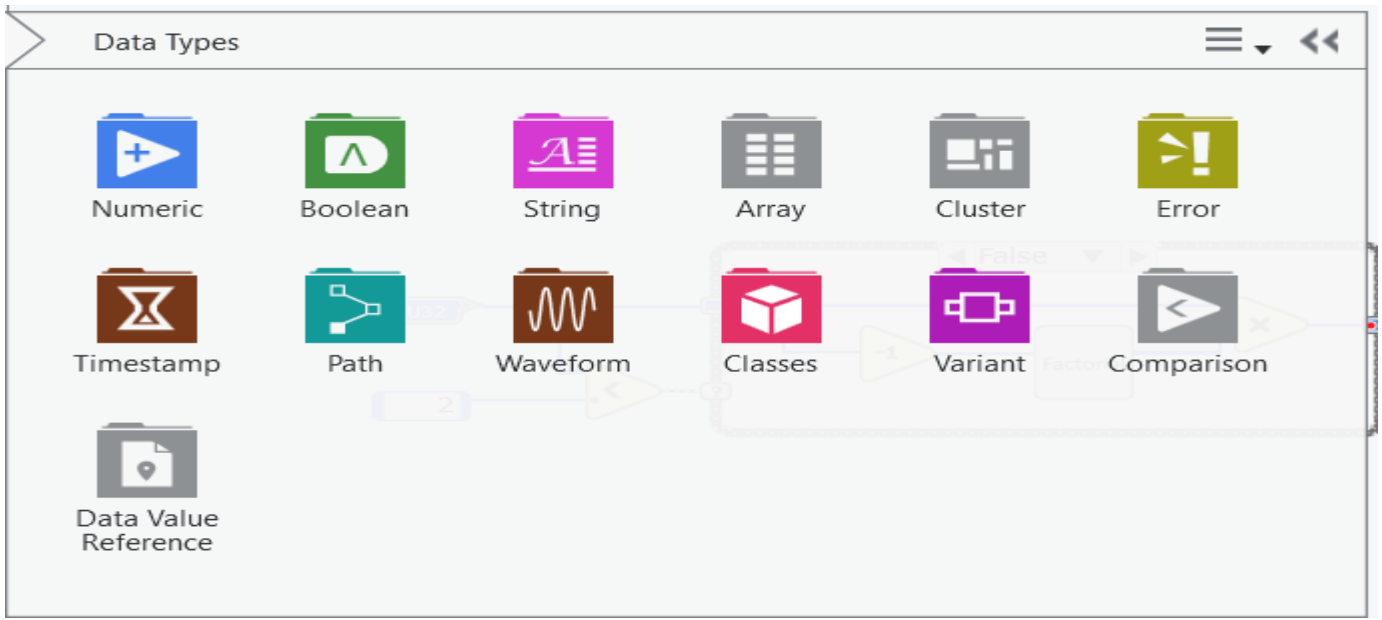
Many numeric types (to interface with hardware and to do signal processing)

Strings

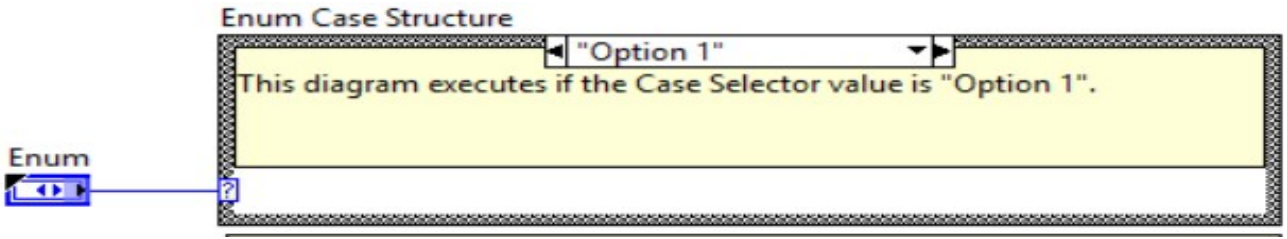
Arrays and records (Clusters)

Classes

…



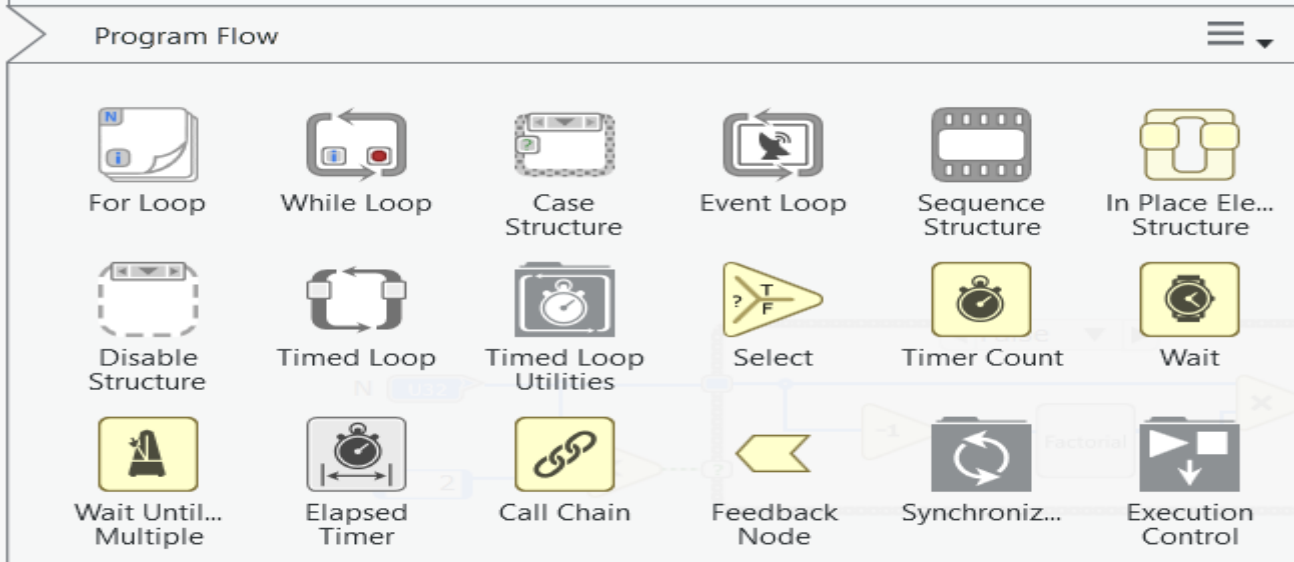
# Control structures and scope




Control structures are represented as boxes

- on the border there is a conditional/control input connector
- the box is the equivalent of a parenthesis
- multiple cases (if-then-else, switch-case) become “pages”
- the box title contains the options of the case/condition
- all “pages” share the same external inputs / outputs
- control values (index)
  are present in all pages

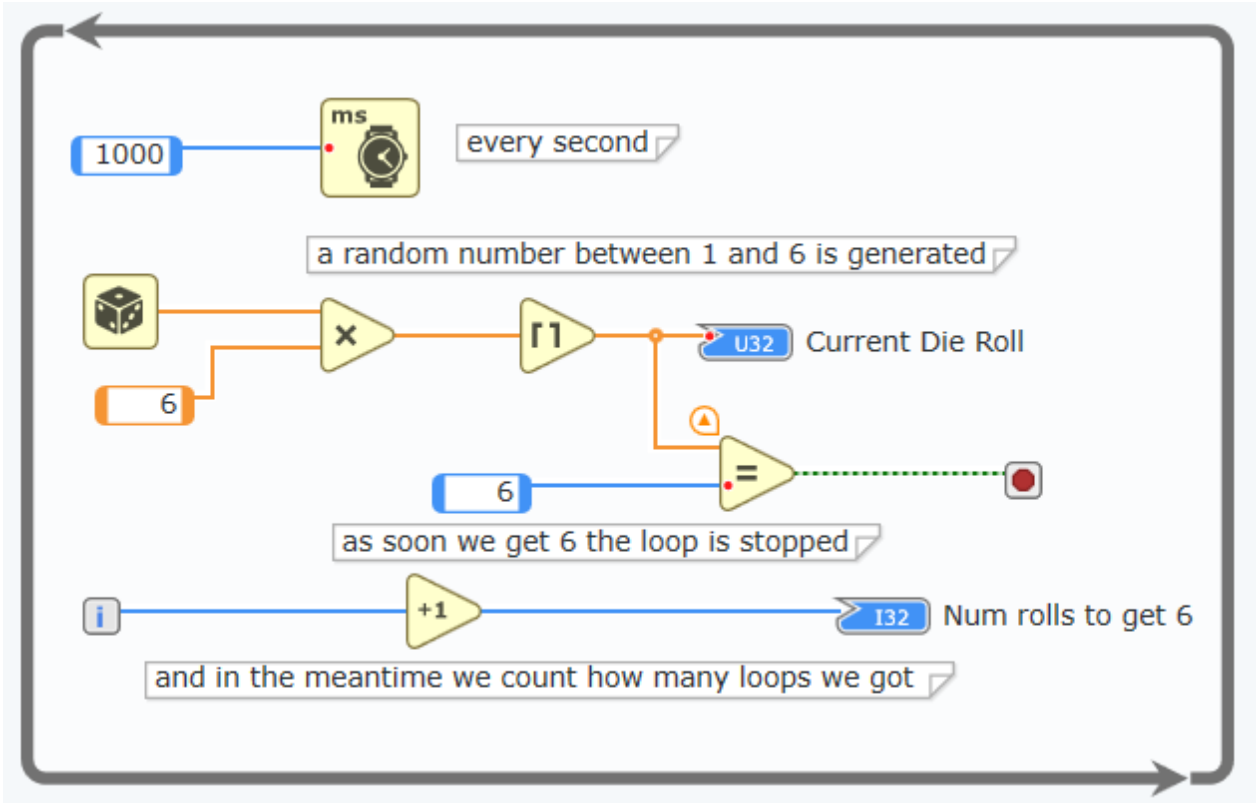
There are also boxes for
formulas or
external code (ASM/C/C++)



# While loop example


The dice generates floats from 0 to 1

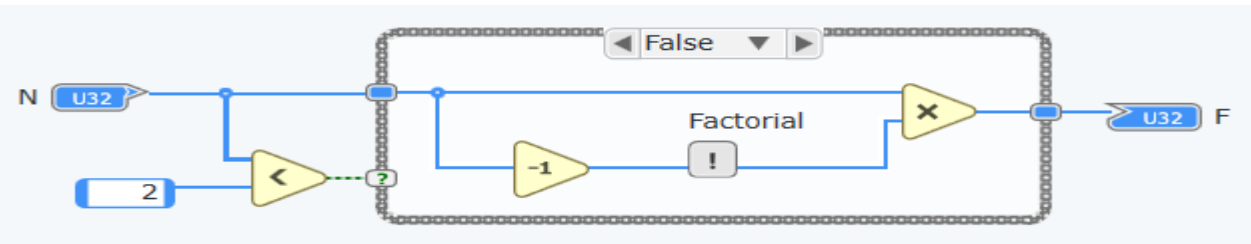
The counter **i** starts from 0

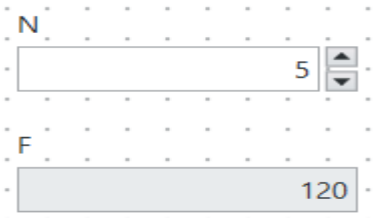


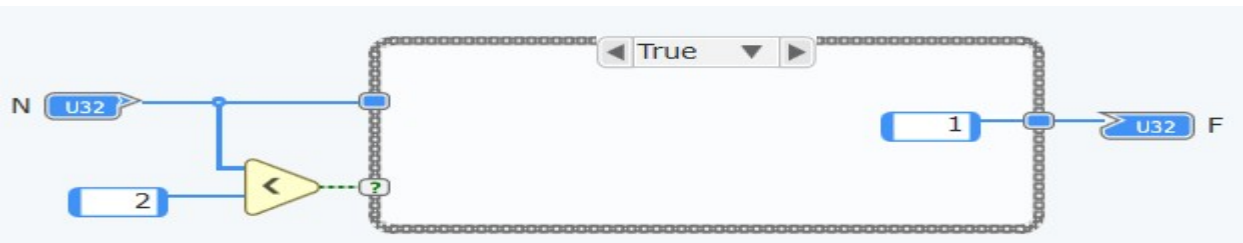
# Recursion? YES

Define a block as “reentrant” (i.e. allowing multiple parallel copies)

Then you can call it inside the same block or one of its sub-blocks




NOTE: you can also define “code” blocks with C

# Concurrency? YES (depends on how parallel is your circuit)

Inherently parallel data-flow implementation

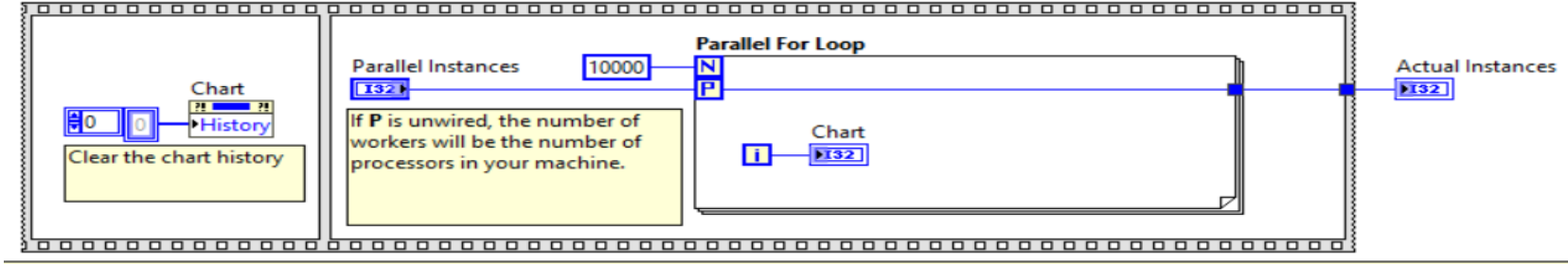
- linked units must run sequentially because of data dependency
- NON-linked units run in parallel (emulated on available cores)

Synchronization

- a block starts computing when all its input data is available

Sequencing constraints can be implicit or explicit

- data dependencies (links induce time order)
- you can add time dependencies without data exchange with the following construct



# LabView programming style

Data-flow visual design

- Visual construction of the data-flow circuit diagram
- Visual test of the diagram
  - all blocks have their GUI showing IN/OUT data
  - probes can be added to show internal wires' values

Inherently parallel (you just forget about sequentiality constraints)

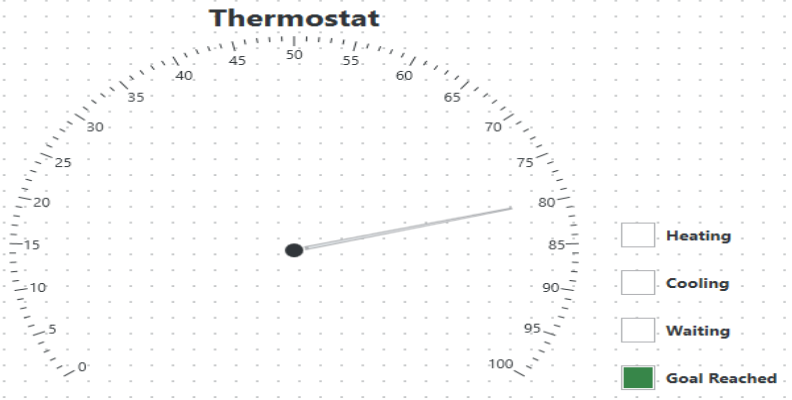
Object-Oriented (classes)

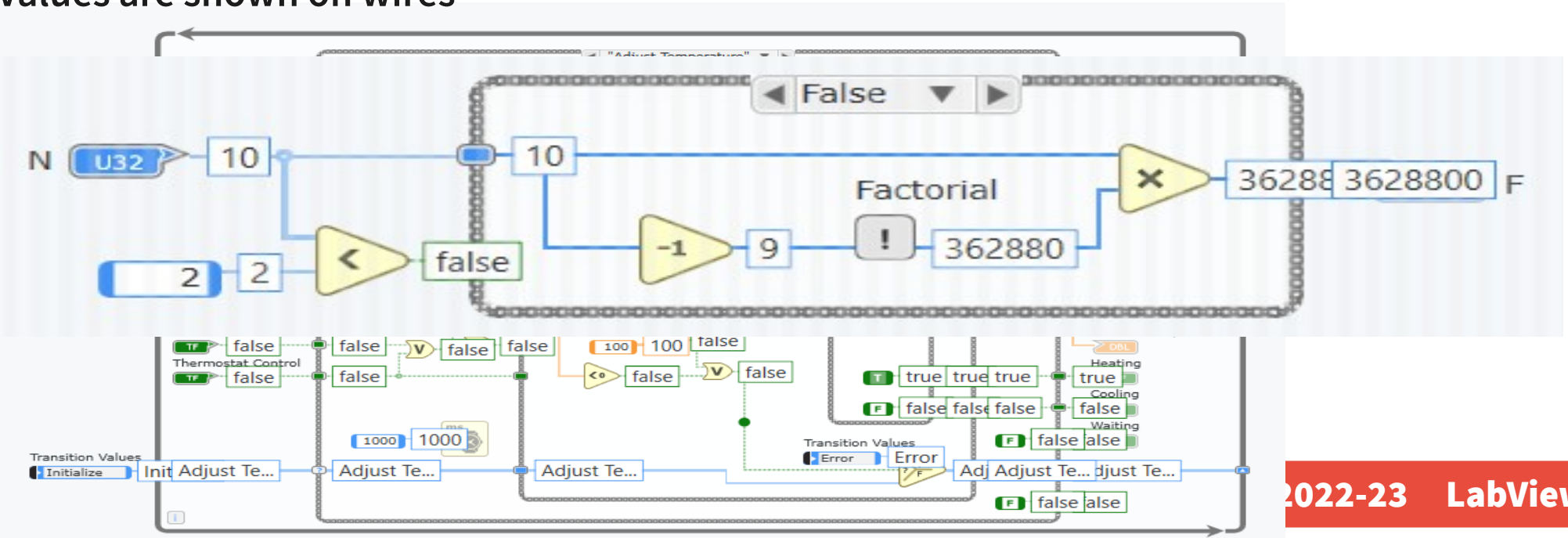
Interaction with other systems:

- Function blocks for data math manipulation
- Code blocks for special algorithms in C++ or Fortran
- Many libraries for Statistics, Signal analysis/manipulation, Math

# Debugging

Visual tracing of data on wires
GUI for blocks IN/OUT
Probes on wires show as widgets on GUI
Values are shown on wires



# LabView for teaching?

PRO

- radically different way to “think” a program
  perhaps suited for deaf/DSA students?
- concurrency
- some problems map naturally to circuits
  (e.g. signal analysis/generation)
- easy definition of feedback control systems
- easy interaction with robots or Arduino
- compiled programs can run inside Lego EV3
  or other microcontrollers

CON

- radically different way to “think” a program
- some algorithm is hard to map to circuits
  (e.g. symbolic problems, text analysis, …)

# Demo

DEMO