# LECTURE 01B. INTRODUCTION TO UIPATH STUDIO

**Robotic Process Automation**
**[03 October 2024]**

Elective Course, 2024-2025, Fall Semester

Andreea-Diana Pop, Lecturer PhD
Babeş-Bolyai University

## Acknowledgements

This course is presented to our Faculty with the support of UiPath Romania.

UiPath

Robotic Process Automation

## Contents

# Automation Project. Definition

- An **activity** is
  - the smallest *action* in UiPath;
  - a **step** in a process workflow;

- An **automation project** is
  - A **set of steps** that allows to perform a meaningful task;
  - a graphical representation of the business **process**;
- it allows to automate a rule-based process, formed by custom set of steps;

- E.g.:
  - Click on a button;
  - Read a file;
  - Write to a log file.

# Automation Project. Types

- Types of supported projects
  - **Sequences -** for linear processes;
    - it connects one activity to another without cluttering the project;
    - *when to use:* simple scenarios, activities follows one after another;
    - easy to assemble and understand;
  - **Flowcharts -** for more complex processes;
    - it integrates decisions and connects activities in a more diverse manner through multiple branching and logic operators;
    - it provides a two dimensional view of the workflow;
    - *when to use:* to show decision points in a process, visual appealing;
    - *cons:* prone to chaotic interweaving of activities;
  - **State machine -** for very large projects;
    - it applies to projects that use a finite number of states during execution which are triggered by a condition or an activity;
    - *when to use:* to represent standard high-level process diagram of transactional business process templates.

*see* **Demo1 - FirstProject**

UiPath™

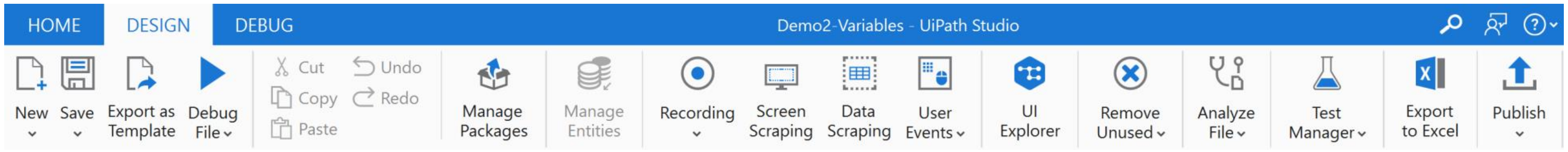
# Automation Project. Structure

- by default:
  - **Main.xaml file –** it consists of the main flow;
    - a sequence or a flowchart can be initially added;
    - other .xaml files may be added;
    - at run time this file will be executed only ==> all other .xaml files should be connected in **Main.xaml** through the **Invoke Workflow File** activity;
    - an .xaml file can be set as *main module* by choosing the **Set as Main** option in the *pop-up menu*;
  - **.screenshots folder –** it is generated if the project uses UI automation;
    - to save the screenshot;
  - **project.json –** it contains details of the automation project.

*see* **Demo1 - FirstProject**

# The User Interface. Top Ribbon

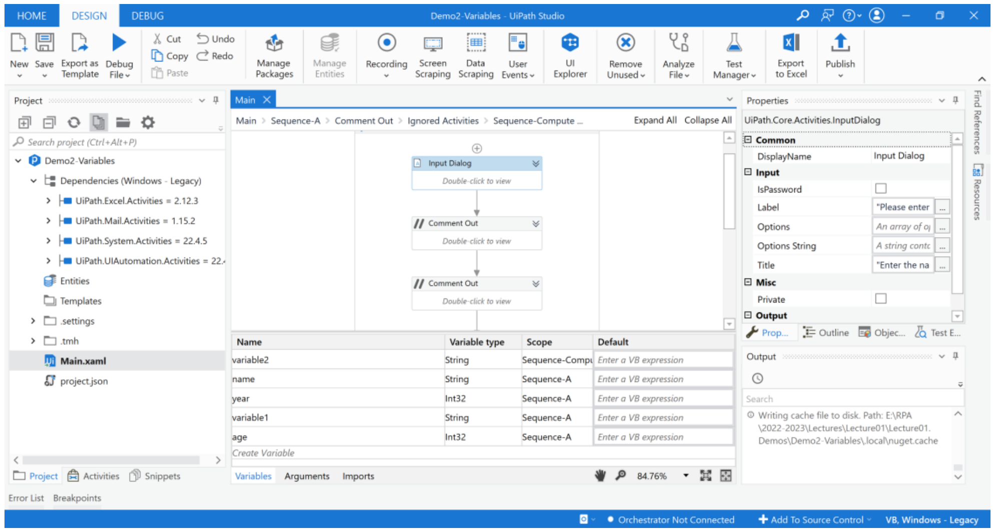
- There are 3 menus on the top ribbon:
  - **Start/Home -** to create a new project, i.e., a new process;
    - It Connects one activity to another without cluttering the project;
  - **Design -** to design the  process;
    - Actions allowed: add activities (sequences, flowcharts, state machine), UI interaction, export to Excel, publish to Orchestrator;
  - **Execute/Debug -** debug related actions;
    - Actions allowed: validate, run, debug, monitor the execution step by step;



*see* **Demo1 - FirstProject**

# The User Interface. Panels

- Main areas (panels) in UiPath Studio:
  - design time:
    - **Project, Activities, Snippets;**
    - **Designer;**
    - **Variables, Arguments, Imports;**
    - **Properties Panel, Outline Panel;**
  - run/debug time:
    - **Output Panel, Locals Panel;**
    - **Error List, Breakpoints.**



| Name | Variable type | Scope | Default |
|---|---|---|---|
| variable2 | String | Sequence-Compu | Enter a VB expression |
| name | String | Sequence-A | Enter a VB expression |
| year | Int32 | Sequence-A | Enter a VB expression |
| variable1 | String | Sequence-A | Enter a VB expression |
| age | Int32 | Sequence-A | Enter a VB expression |
| Create Variable | | | |

UiPath.Core.Activities.InputDialog

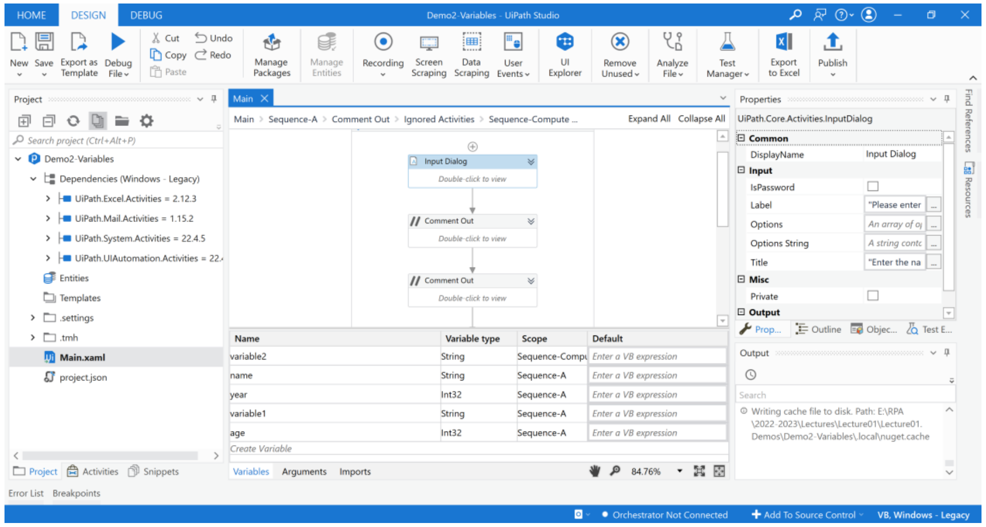
| | |
|---|---|
| **Common** | |
| DisplayName | Input Dialog |
| **Input** | |
| IsPassword | ☐ |
| Label | "Please enter |
| Options | An array of o |
| Options String | A string conta |
| Title | "Enter the na |
| **Misc** | |
| Private | ☐ |
| **Output** | |

```
Writing cache file to disk. Path: E:\RPA
\2022-2023\Lectures\Lecture01\Lecture01.
Demos\Demo2-Variables\.local\nuget.cache
```

*see* **Demo1 – FirstProject, Demo2 – Variables**

# Demo 1 - FirstProject

- Tasks:
  - Create a simple project of type **Process**;
  - Work with **Sequence** and **Flowchart** containers.
  - Read and write some data (numeric, text);
    - work with **Input Dialog** and **Write Line** and **Message Box** activities;
    - **Input Dialog** activity versatility.



Main > Sequence-A > Comment Out > Ignored Activities > Sequence-Compute ...

| Name | Variable type | Scope | Default |
|---|---|---|---|
| variable2 | String | Sequence-Compu | Enter a VB expression |
| name | String | Sequence-A | Enter a VB expression |
| year | Int32 | Sequence-A | Enter a VB expression |
| variable1 | String | Sequence-A | Enter a VB expression |
| age | Int32 | Sequence-A | Enter a VB expression |

Properties: UiPath.Core.Activities.InputDialog

| | |
|---|---|
| **Common** | |
| DisplayName | Input Dialog |
| **Input** | |
| IsPassword | ☐ |
| Label | "Please enter |
| Options | An array of o |
| Options String | A string contc |
| Title | "Enter the na |
| **Misc** | |
| Private | ☐ |
| **Output** | |

```
Writing cache file to disk. Path: E:\RPA
\2022-2023\Lectures\Lecture01\Lecture01.
Demos\Demo2-Variables\.local\nuget.cache
```

*see* **Demo1 – FirstProject**

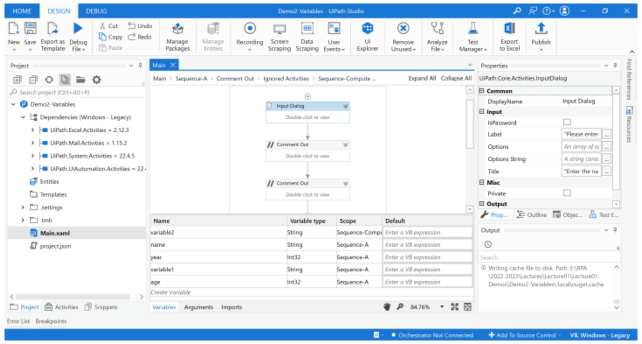
# Variables. Data Types

- **Variables** are used to store different types of data: numeric, text, image, file, colour;
- main types of variables:
  - **Int32**;
  - **String** – with quotes, e.g., “abc”, “123”;
  - **Boolean** = {True, False};
  - **GenericValue** – almost any data type;
  - **Array of [T]** – all values have the same type;
- a variable defined within a scope (e.g., **Sequence**, **Flowchart**) is available in all scopes included in it;
- **Variable Panel** shows the properties of the defined variables:
    - **Name**, **Type**, **Scope**, **Default value**;
  - it presents the variables available in the selected activity.
- VB. Net operators used: **Not** for **!**; **<>** for **!=**; **And** for &; **Or** for |; **=** for ==; **Mod** for % (see https://en.wikipedia.org/wiki/Comparison_of_C_Sharp_and_Visual_Basic_.NET#Comparers).

*see* **Demo2 - Variables**

# Demo 2 - Variables

- Tasks:
  - Compute the age in years;
    - variables of type String and Int32;
    - work predefined objects and properties, e.g., property Now from class **System**;
  - Identifiers
    - duplicated identifiers
    - not case sensitiveness in UiPath;
    - work with **Sequence** and **Flowchart** containers.
  - Generic variables
    - **GenericValue** data type, converted to **String** or **Int32**
      - c=a+b
      - c=b+a
      - different meanings for the + operator, according to the closest type of the first operand.



| Name | Variable type | Scope | Default |
|---|---|---|---|
| variable2 | String | Sequence-Compu | Enter a VB expression |
| name | String | Sequence-A | Enter a VB expression |
| year | Int32 | Sequence-A | Enter a VB expression |
| variable1 | String | Sequence-A | Enter a VB expression |
| age | Int32 | Sequence-A | Enter a VB expression |

*see* **Demo2 – Variables**

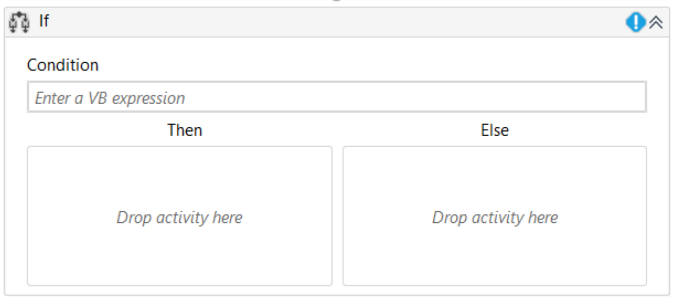
## Choices

| Activity | Sequence | Flowchart | Assign |
|---|---|---|---|
| **If** activity | Yes | Yes | No |
| **Else If** | Yes | Yes | No |
| **Flow Decision** | No | Yes, similar to **Else If** activity in a **Sequence** | No |
| **If** operator (VB) | No | No | Yes, similar to **If** activity in a **Sequence** |
| **?:** operator (C#) | No | No | Yes, similar to **If** activity in a **Sequence** |
| **Switch** | Yes | Yes | No |
| **Flow Switch** | No | Yes | No |

*see* **Demo3 – Choices (VB, C#), Demo4 – IfOperator (VB)**

# Choices. If Activity

- **If** activity:
  - it splits the sequence vertically;
  - adequate for short linear branches;
- *cons:*
  - more than one *if else if* chained affects perception on the screen;


If

Condition

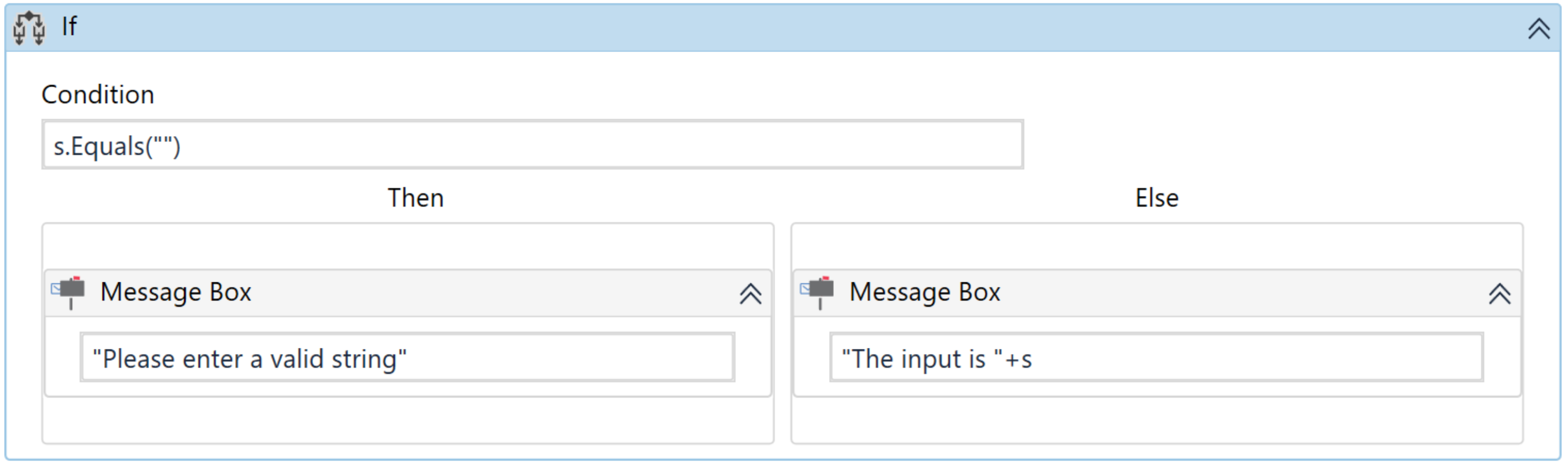
Enter a VB expression

Then

Drop activity here

Else

Drop activity here


If

Condition

s.Equals("")

Then

Message Box

"Please enter a valid string"

Else

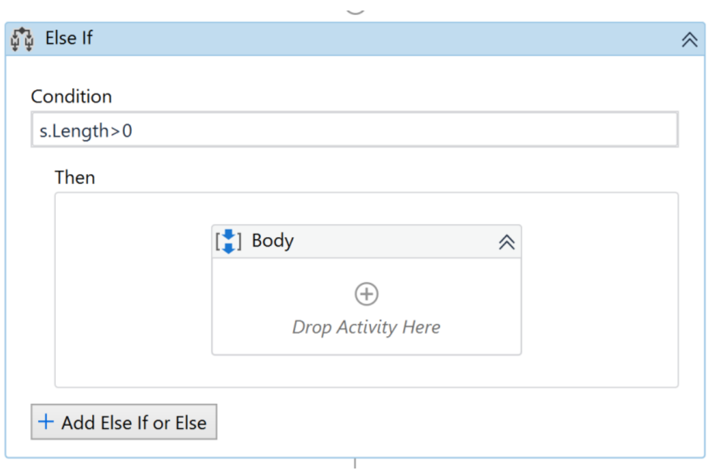
Message Box

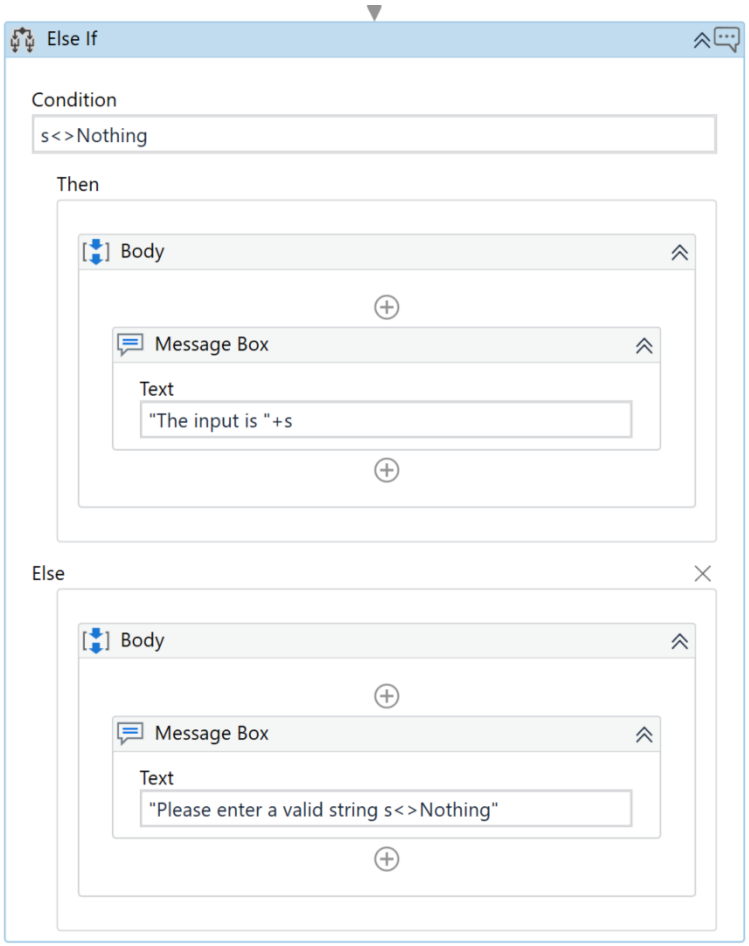
"The input is "+s

*see* **Demo3 – Choices (VB, C#)**

# Choices. Else If Activity

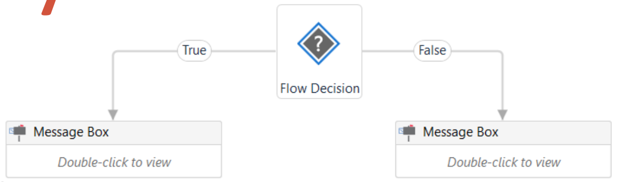
- **Else If** activity:
  - suitable for cases when the project takes different courses of action, depending on whether a *series of specific conditions are met*;
  - the **Else** or **Else If** condition is **optional;**



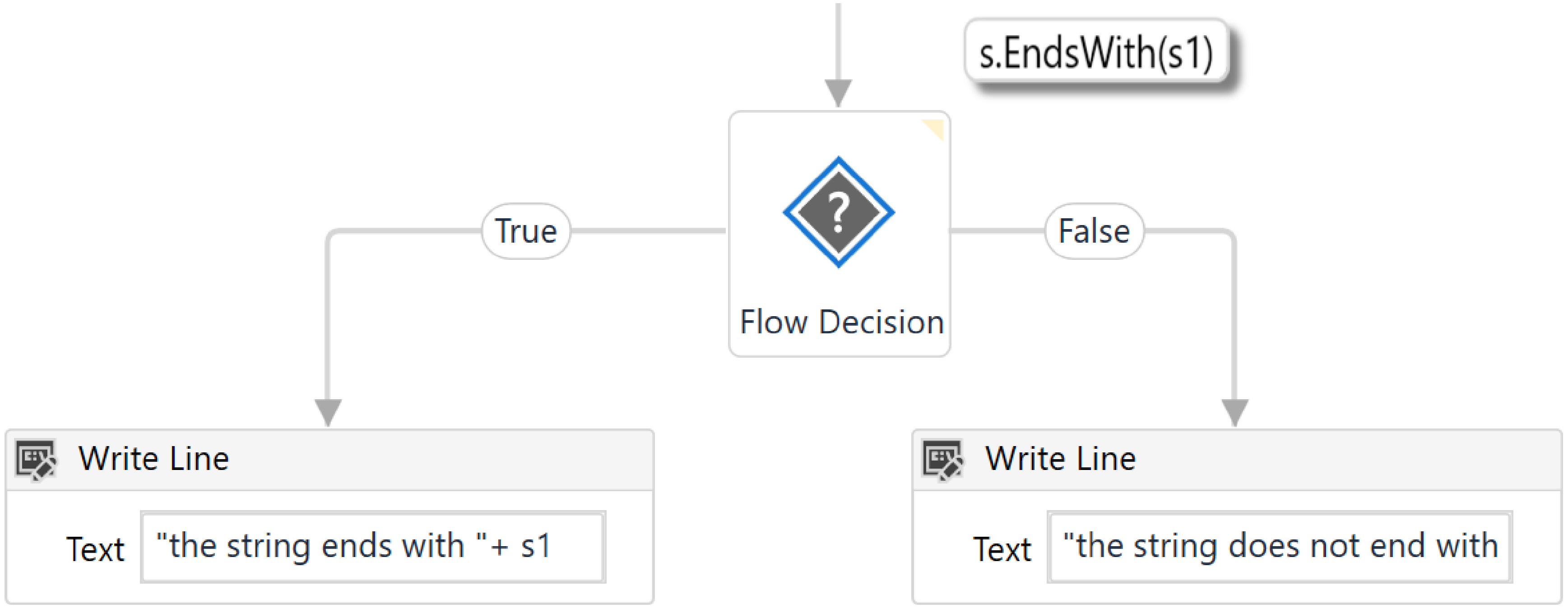


*see* **Demo3 – Choices (VB, C#)**

# Choices. Flow Decision Activity



- **Flow Decision** activity:
  - it shows important decision logic and related conditions;
  - **Flow Decision** activity in *flowcharts* = **If** *or* **Else If** activity in *sequences*;



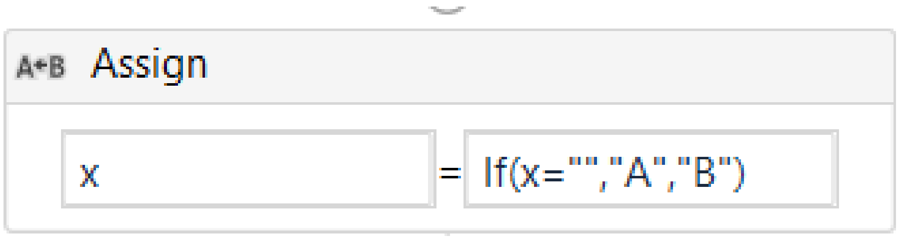
"the string ends with "+ s1

"the string does not end with

*see* **Demo3 – Choices (VB, C#)**

## Choices. If Operator

- **If** operator:
  - this is a VB operator;
  - useful for **small local conditions** or **data computations**;
  - it reduces the block to a single **Assign** activity;

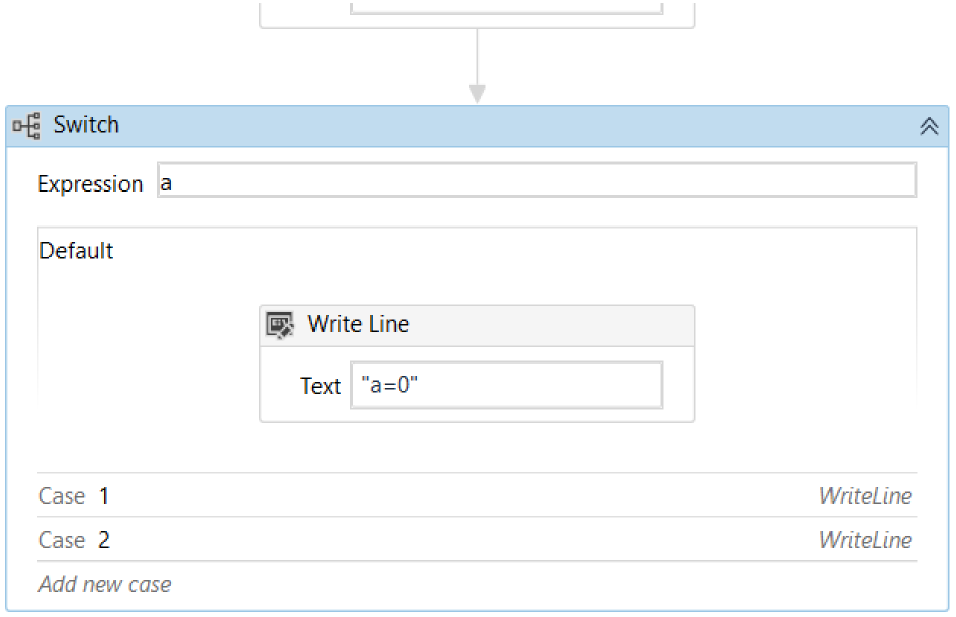


```
x = If(x="","A","B")
```

*see* **Demo3 – Choices (VB), Demo4 – IfOperator (VB)**

# Choices. Switch Activity

- **Switch** activity:
  - it can be used together with **If** operator; to streamline and compact *if else if* cascade, with distinct conditions and activities per branch;



Switch

Expression a

Default

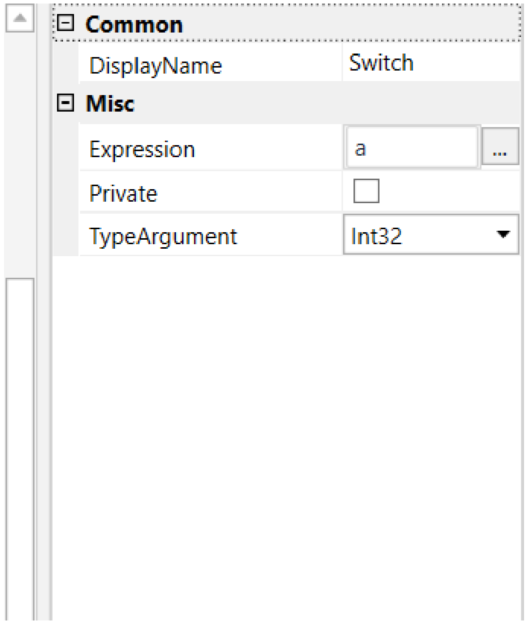
Write Line

Text "a=0"

Case 1 *WriteLine*

Case 2 *WriteLine*

*Add new case*



Common

| | |
|---|---|
| DisplayName | Switch |

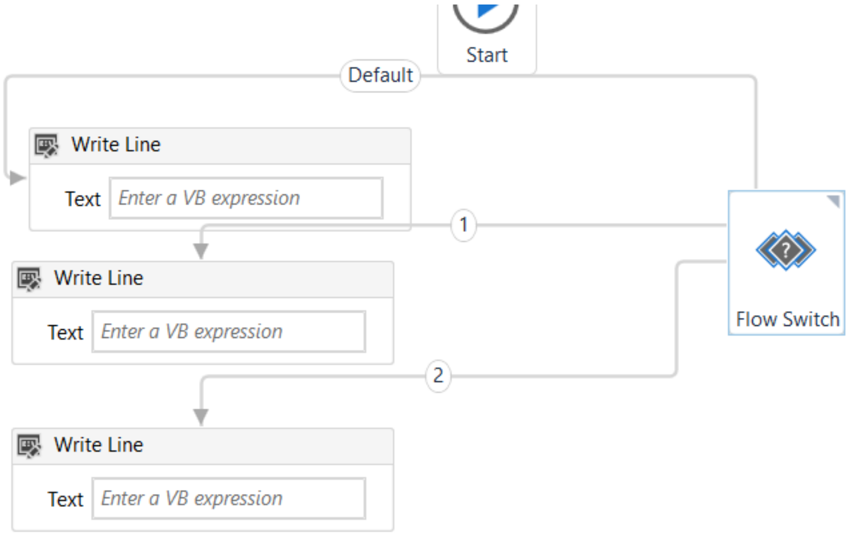
Misc

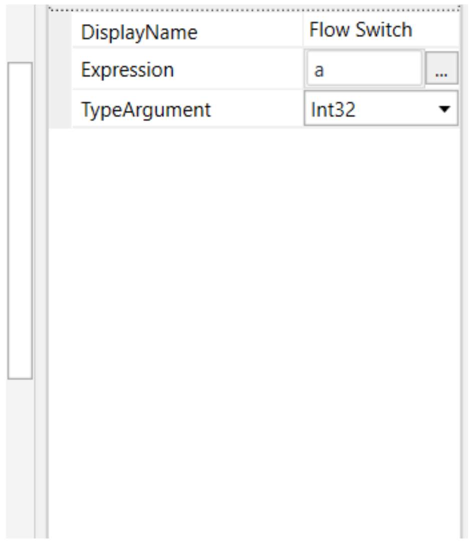
| | |
|---|---|
| Expression | a |
| Private | ☐ |
| TypeArgument | Int32 |

*see* **Demo3 – Choices (VB, C#)**

# Choices. Flow Switch Activity

- **Flow Switch** activity:
  - an **If** activity that selects the next node depending on the value of expression;
  - **Flow Switch** activity in *flowcharts* = **Switch** activity in *sequences*;



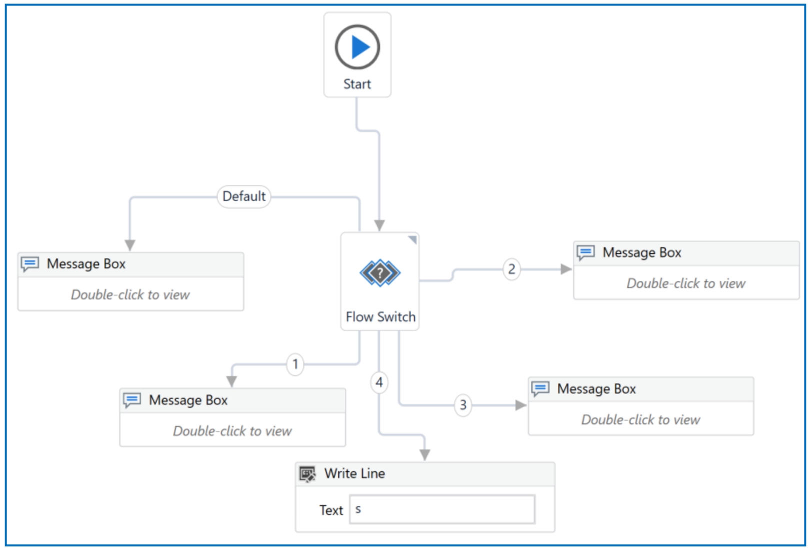


| | |
|---|---|
| DisplayName | Flow Switch |
| Expression | a |
| TypeArgument | Int32 |

*see* **Demo3 – Choices (VB, C#)**

# Demo 3 - Choices

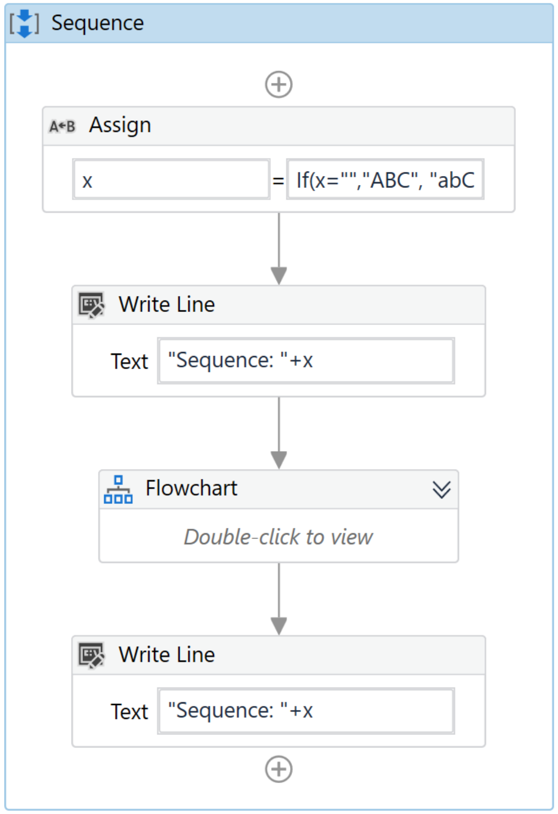
- Tasks:
  - **If** activity
    - checks whether a string  equals to “”;
  - **Else If** activity
    - checks whether a string  is null;
  - **Flow Decision** activity
    - checks if a string ends with “T”;
  - **If** operator (VB) *or* **?:** ternary operator (C#)
    - checks if a string ends with “T”;
  - **Switch** activity
    - checks the value of some expression and acts accordingly
  - **Flow Switch** activity
    - checks the value of some a string related expression and acts accordingly

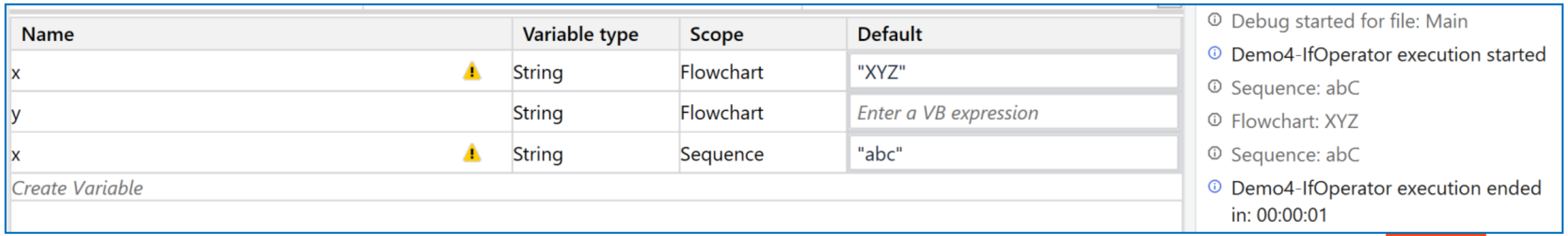


*see* **Demo3 – Choices (VB, C#), Demo4 – IfOperator (VB)**

# Demo 4 – If Operator (VB)

- Tasks:
  - **If** operator (VB)
    - x=If (x="", "ABC", "abC")
  - working with duplicated identifiers in different scopes





| Name | Variable type | Scope | Default |
|---|---|---|---|
| x ⚠ | String | Flowchart | "XYZ" |
| y | String | Flowchart | Enter a VB expression |
| x ⚠ | String | Sequence | "abc" |
| Create Variable | | | |

```
Debug started for file: Main
Demo4-IfOperator execution started
Sequence: abC
Flowchart: XYZ
Sequence: abC
Demo4-IfOperator execution ended
in: 00:00:01
```

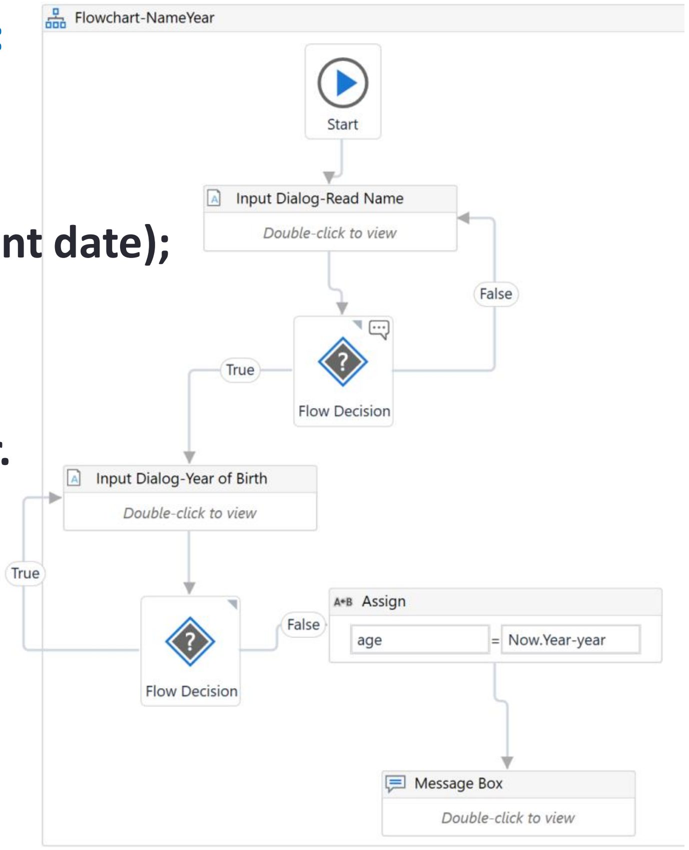
*see* **Demo4 – IfOperator (VB)**

# Demo 5 – Name Age

- Create a process that performs the following actions:
  - 1. *read* the name of the person;
  - 2. *read* the birth year;
  - 3. *compute* the age in years (considering the current date);
  - 4. *print* "Congratulations, Z! You are x years old!"

- Do not allow empty name and/or negative birth year.
  - *What is the easiest way to achieve it?*

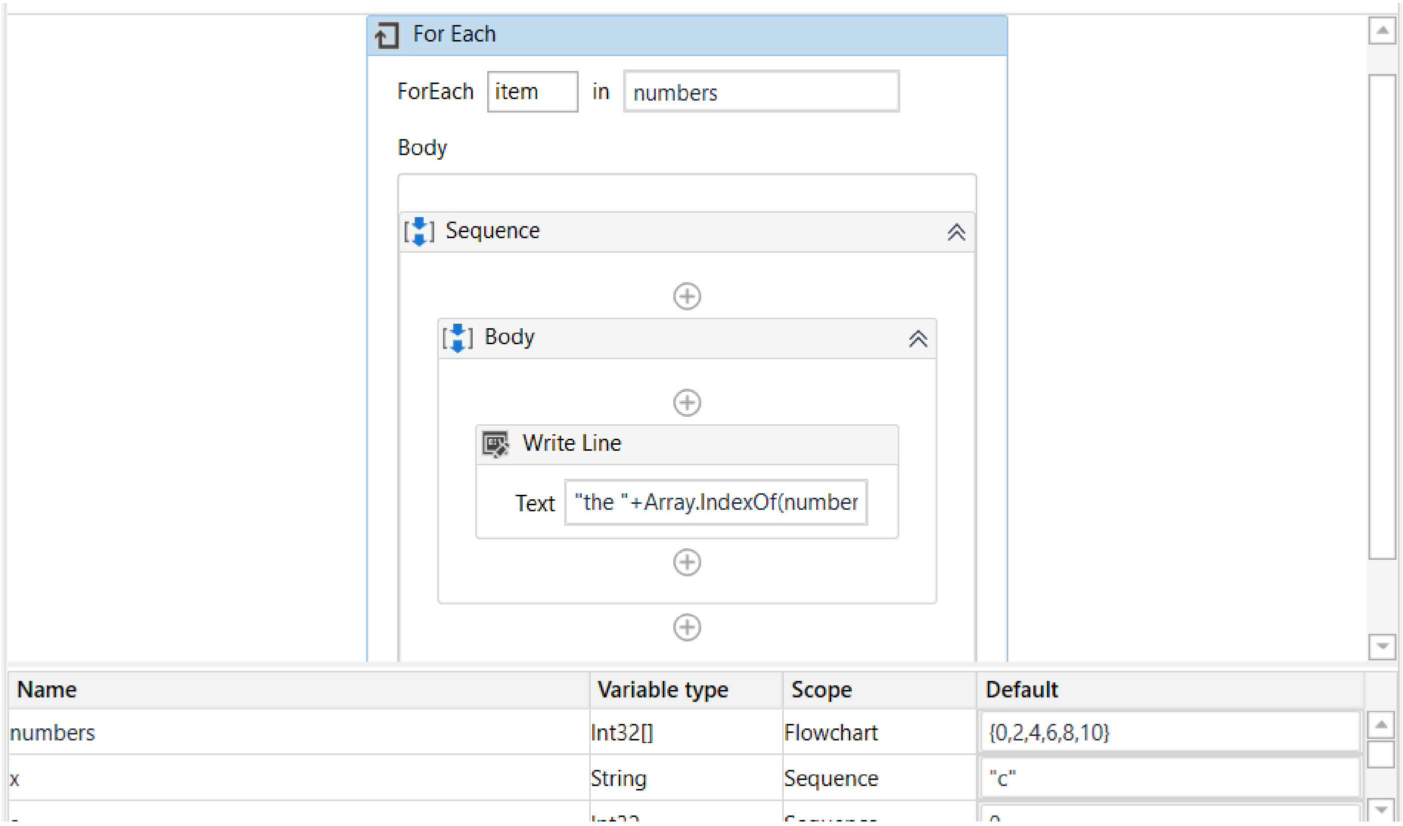
  ***sequence/flowchart.***



*see* **Demo5 - NameAge**

# Control Flow. For Each Activity

- **For Each** activity:



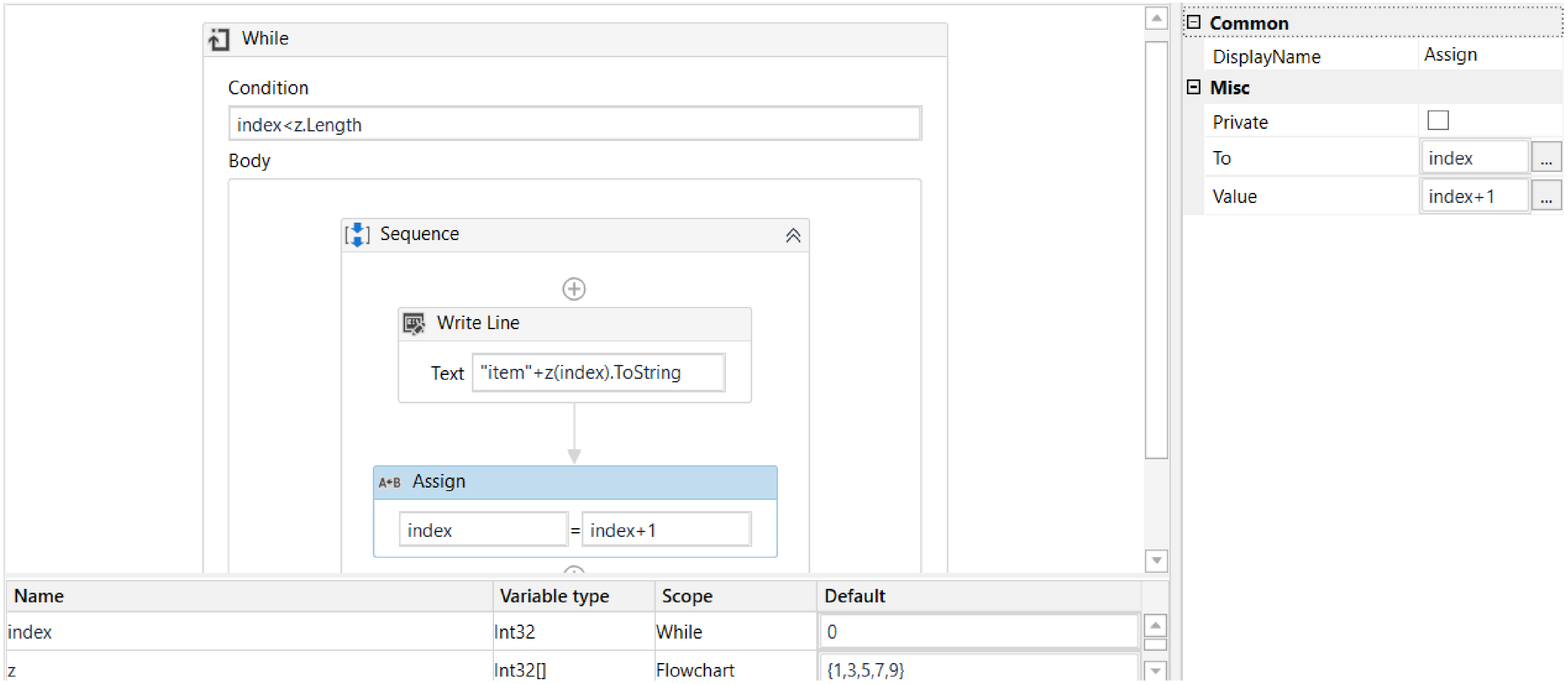
Text "the "+Array.IndexOf(number

| Name | Variable type | Scope | Default |
|---|---|---|---|
| numbers | Int32[] | Flowchart | {0,2,4,6,8,10} |
| x | String | Sequence | "c" |

*see* **Demo6 - ControlFlow**

## Control Flow. While Activity

- **While** activity:



While

Condition

```
index<z.Length
```

Body

Sequence

Write Line

Text "item"+z(index).ToString

Assign

index = index+1

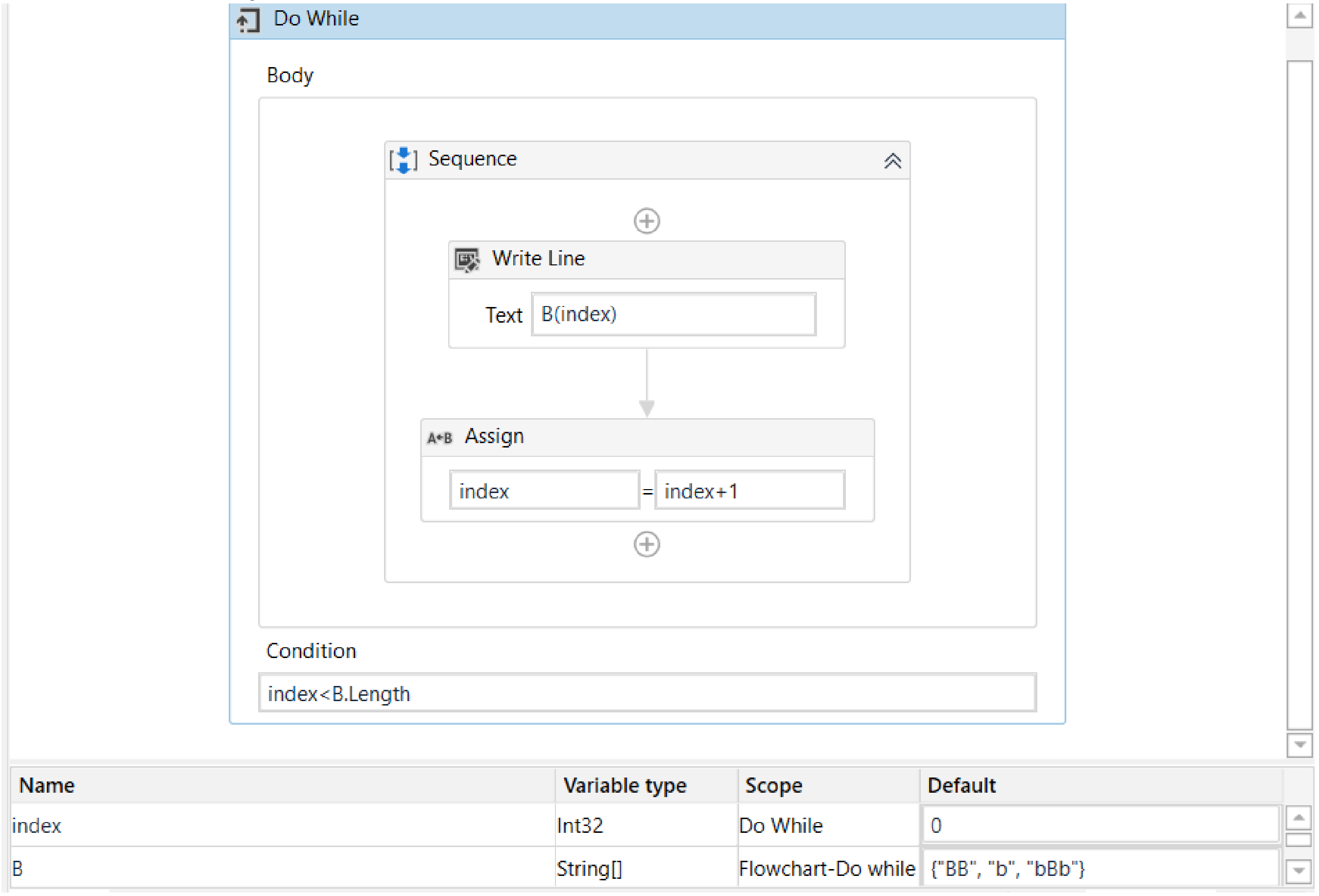
| Name | Variable type | Scope | Default |
|---|---|---|---|
| index | Int32 | While | 0 |
| z | Int32[] | Flowchart | {1,3,5,7,9} |

| | |
|---|---|
| **Common** | |
| DisplayName | Assign |
| **Misc** | |
| Private | ☐ |
| To | index |
| Value | index+1 |

*see* **Demo6 - ControlFlow**

# Control Flow. Do While Activity

- **Do While** activity:

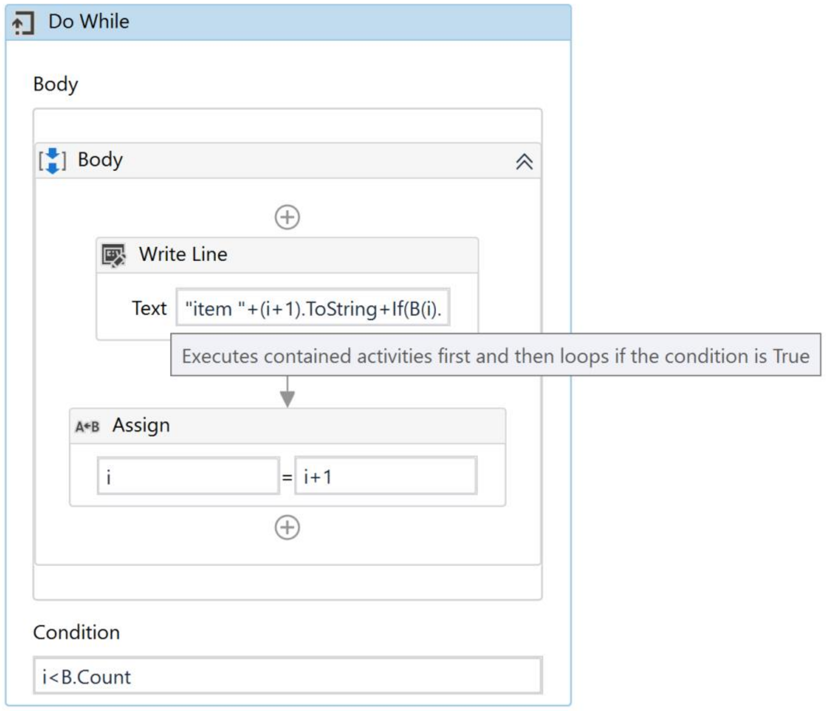


| Name | Variable type | Scope | Default |
|---|---|---|---|
| index | Int32 | Do While | 0 |
| B | String[] | Flowchart-Do while | {"BB", "b", "bBb"} |

*see* **Demo6 - ControlFlow**

# Demo 6 – Control Flow

- Tasks:
  - **For Each**
    - navigating over an array of integers {0,0,2,4,6,8,10}
    - accessing the item and index from some specific array position
  - **While**
    - navigating over an array of integers {1,3,5,7,9}
    - accessing the array items based on the index
  - **Do While**
    - navigating over an array of strings {"B", "bBb", "BB"}
    - checking whether some condition is kept for each item in the array


Do While
Body
Body
Write Line
Text "item "+(i+1).ToString+If(B(i).
Executes contained activities first and then loops if the condition is True
Assign
i = i+1
Condition
i<B.Count

*see* **Demo4 – IfOperator (VB)**

## Demo 7

- Create a process that performs the following actions:
  - 1. *generate* an integer number from 1 to 7;
  - 2. *read* a number to guess the generated number;
  - 3. *compare* the generated value
    - 3.1. print the message “Enter a smaller number!” or
    - 3.2. print the message “Enter a bigger number!”;
  - 4. *repeat* steps 2 and 3 until you succeed to find the number;
  - 5. *show* the message “Well done!!!”

- Is there a way to design the workflow without ForEach/While/DoWhile activities?

*see* Demo7 – GuessRandomNumber

UiPath™

## References

- UiPath Docs - https://docs.uipath.com
- UiPath Studio Docs - https://docs.uipath.com/studio/standalone/2023.4
- UiPath Forum - https://forum.uipath.com/
- UiPath Academy - https://academy.uipath.com/