



Katedra za telekomunikacije

## Praktikum softverski alati 2

# LabVIEW

**Prof. dr Mirjana Simić-Pejović**

**kabinet: 108**

**[mira@etf.rs](mailto:mira@etf.rs)**



**NATIONAL INSTRUMENTS™  
LabVIEW™**

# Sadržaj

- Potprogrami u LV – SubVI
- Strukture i petlje
  - For i While
  - Case
  - Sequence
  - Formula Node
- Arrays
- Strings

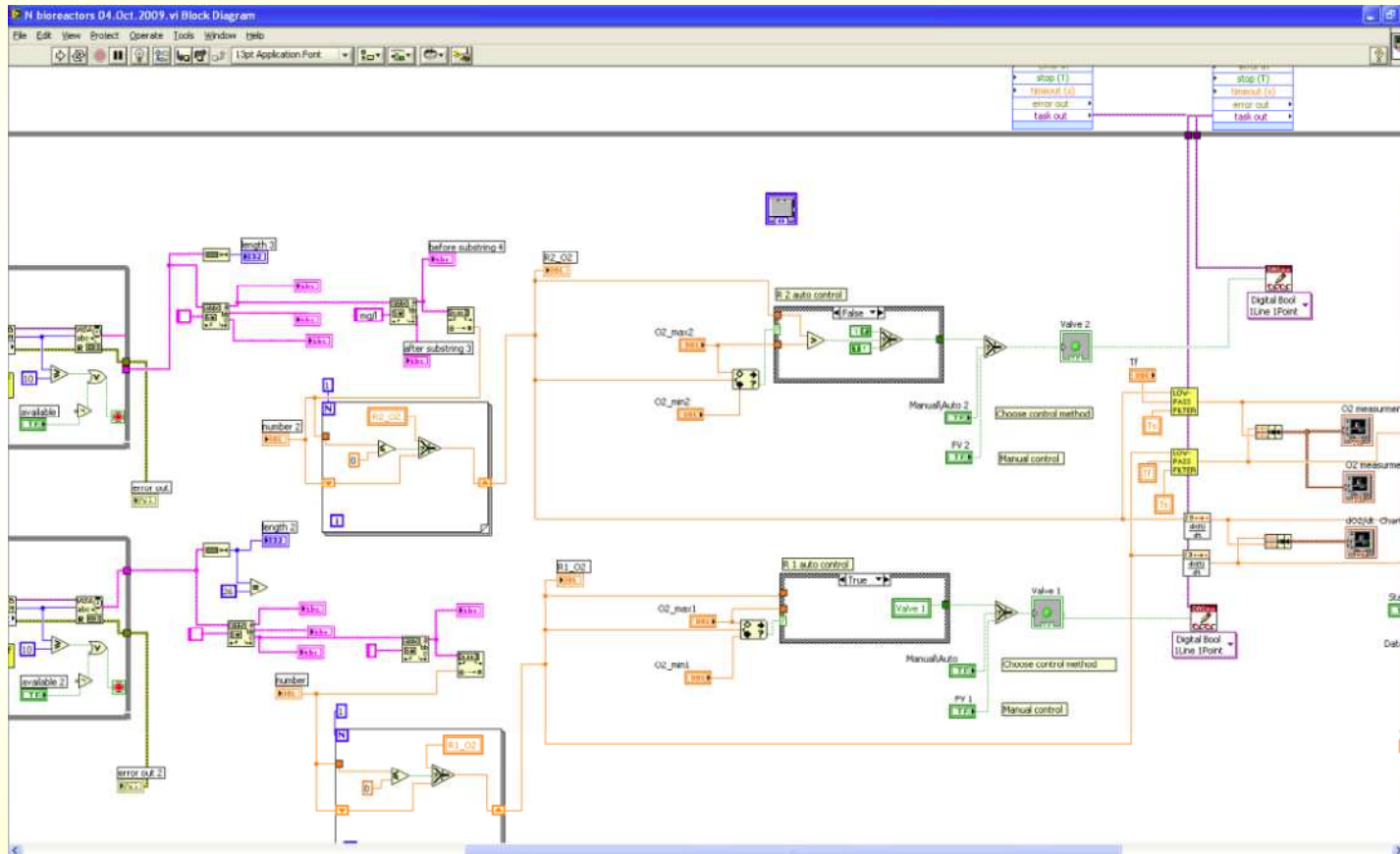
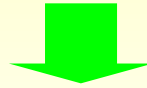


# Main vi & Sub vi

- Kada se neki LV program tj. **vi** unese u blok dijagram postojećeg **vi** programa (što se postiže kada se u Function palleti bira 'Select VI'), LV ga tretira kao potprogram tj. **Sub vi**. Postojeći **vi** u tom slučaju postaje **Main vi**.
- **Sub vi** (njegov front panel i blok dijagram) dostupan je duplim klikom na njegovu ikonu u blok dijagramu **vi** programa koji ga poziva (**Main vi**).
- Svaki **vi** se može koristiti i kao **Main vi**, i kao **Sub vi**. Razlika je u tome što ako nameravamo da ga koristimo kao potprogram tj. **Sub vi**, onda osim front panela i blok dijagrama, **vi** dobija i svoj treći deo:
  - **ikonu i konektore.**



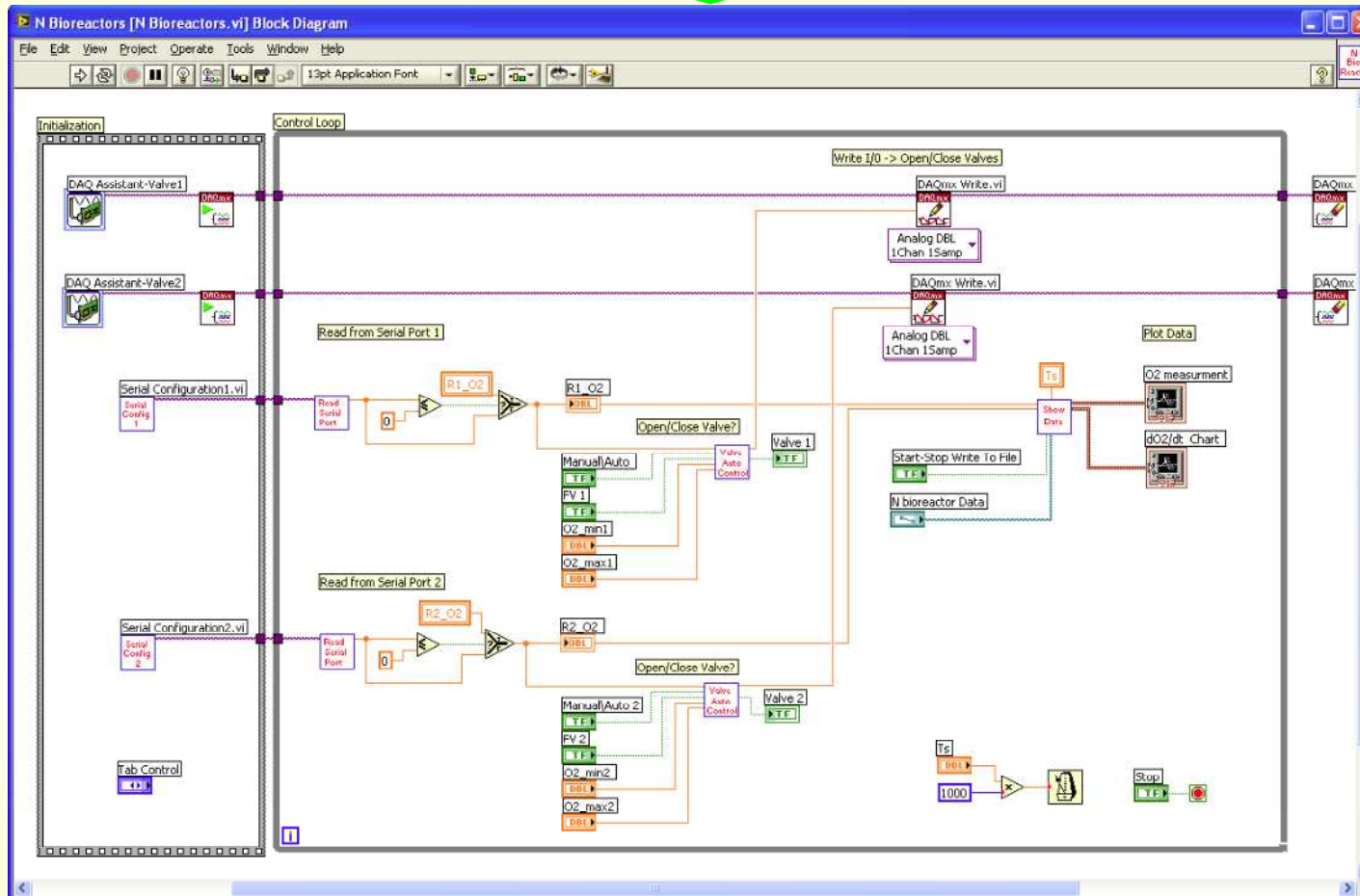
# Zašto je potrebno da postoje potprogrami, tj. Sub vi? primer LV programa gde se ne koriste potprogrami



- blok dijagram nije pregledan
- težak za razumevanje



# Zašto je potrebno da postoje potprogrami, tj. Sub vi? primer LV programa gde se koriste potprogrami



- veći deo koda **Main vi** je izmešten u **Sub vi** - blok dijagram pregledniji!
- mnogo jasniji za razumevanje

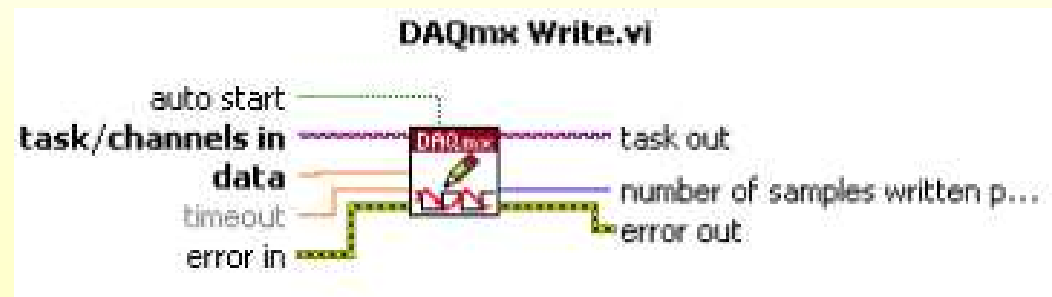


# Kako se pravi Sub vi?

- Kao što je poznato, da bi se neki **vi** koristio kao **Sub vi**, potrebno je da mu se:
  1. **postave konektori**
  2. **kreira ikona**
- Konektori služe da se u potprogram mogu uneti ulazni podaci kao i da se iz potprograma mogu koristiti izlazni podaci (rezultat).
- Postoje ulazni (input) i izlazni (output) konektori, što u drugim programskim jezicima odgovara ulaznim argumentima i izlazu, tj. rezultatu.

## Sub vi sa konektorima

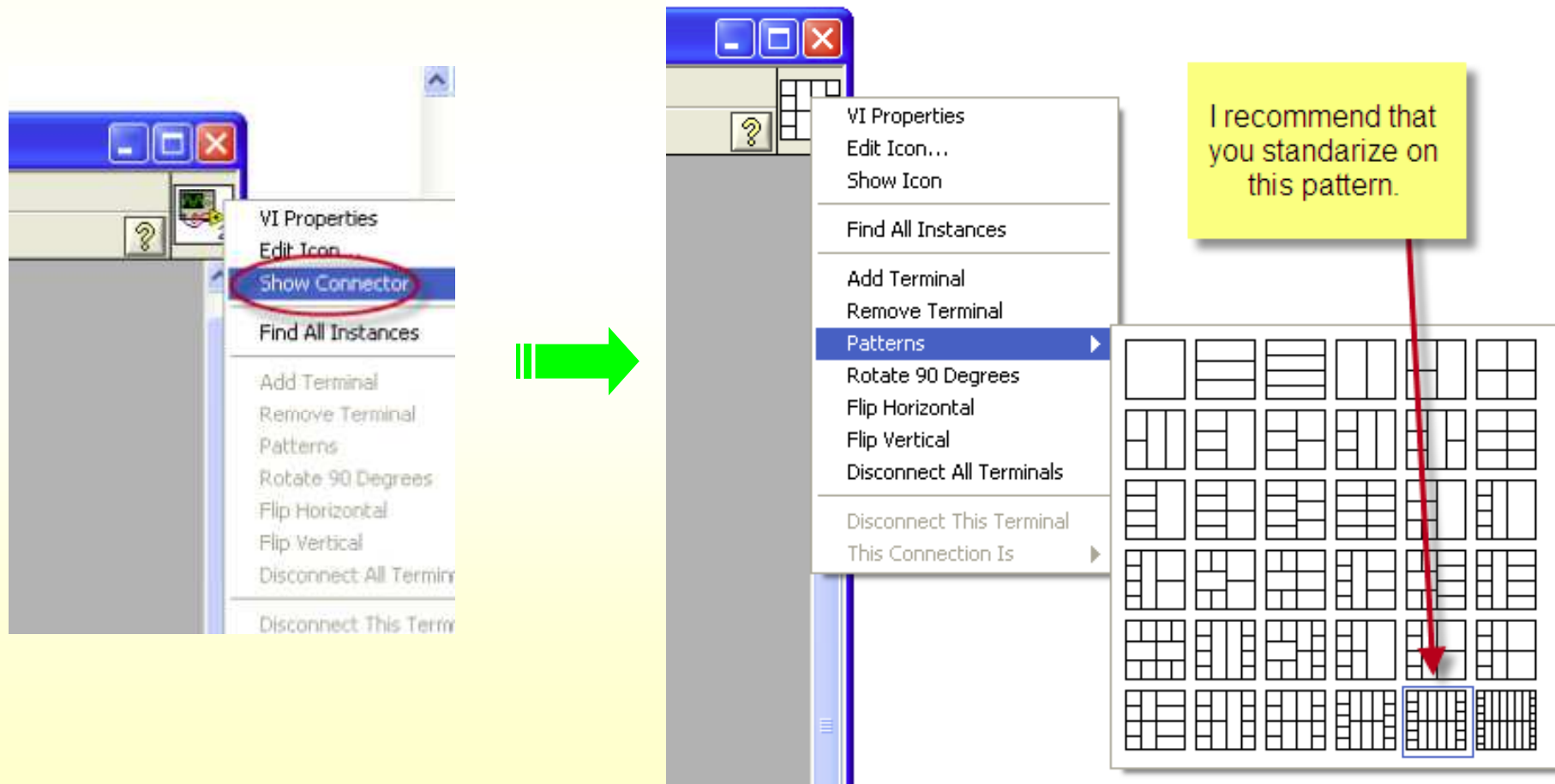
### Sub vi bez konektora



# Kako se pravi Sub vi?


- kreiranje konektora -

- Konektori se kreiraju desnim klikom na ikonu u gornjem desnom uglu vi programa i biranjem opcije "Show connector". Zatim se biraju različiti paterni u zavisnosti koliko ima ulaza/izlaza konkretan vi.



# Kako se pravi Sub vi?

- kreiranje konektora -

- Najzad, da bi se željenim kontrolerima i indikatorima sa front panela potprograma dodelili konektori, koristi se Wire tool. 
- Koristeći Wire tool, klikne se na željeni konektor, a zatim i na željeni kontroler/indikator na front panelu i time je dodeljivanje izvršeno!



korisni saveti/pravila pri  
dodeljivanju konektora



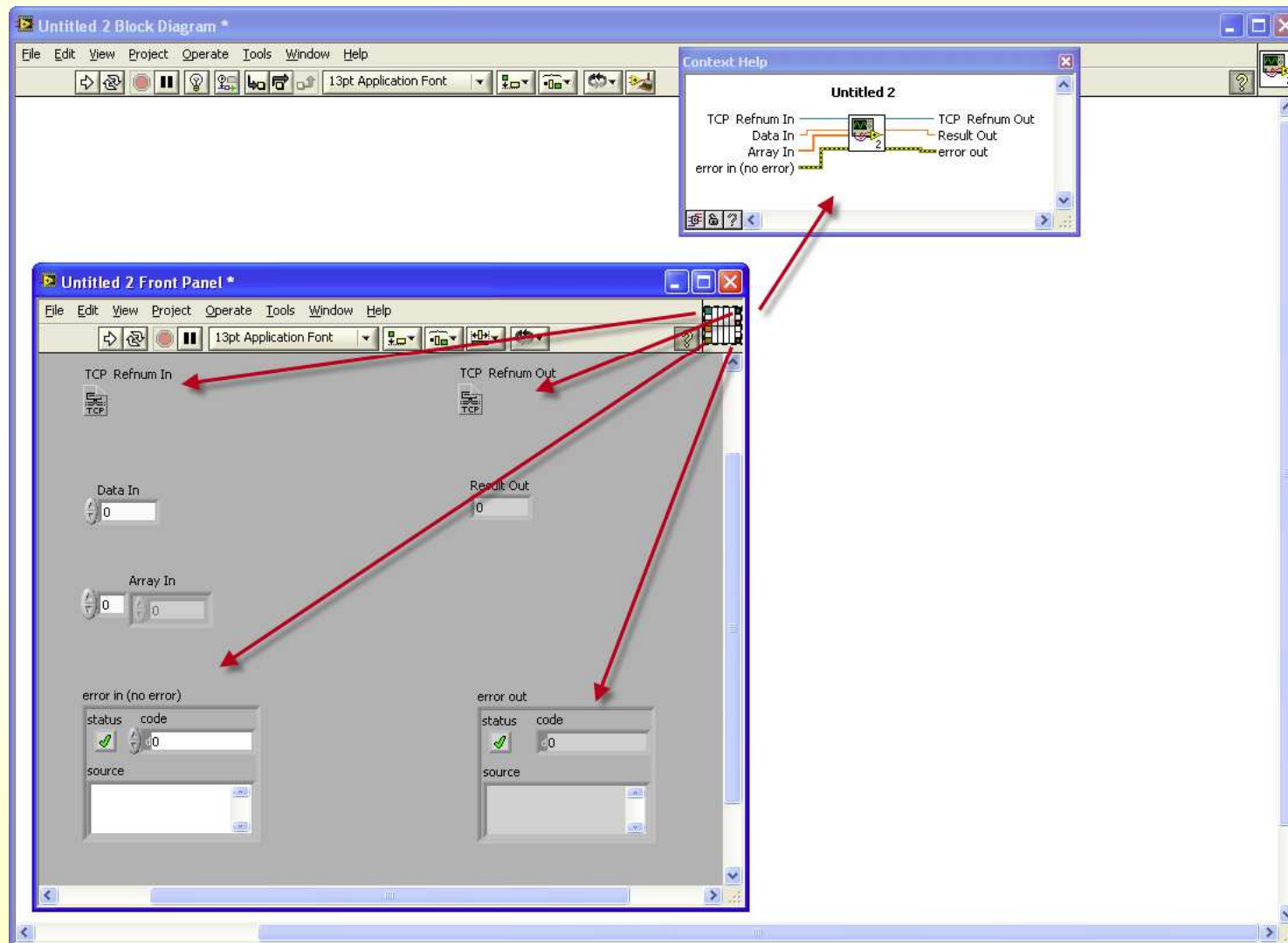
- Upper left connector: "Reference" In
- Upper right connector: "Reference" Out
- Lower left connector: Error In Cluster
- Lower right connector: Error Out Cluster

TRUMENTS™  
**IEW™**



# Kako se pravi Sub vi?

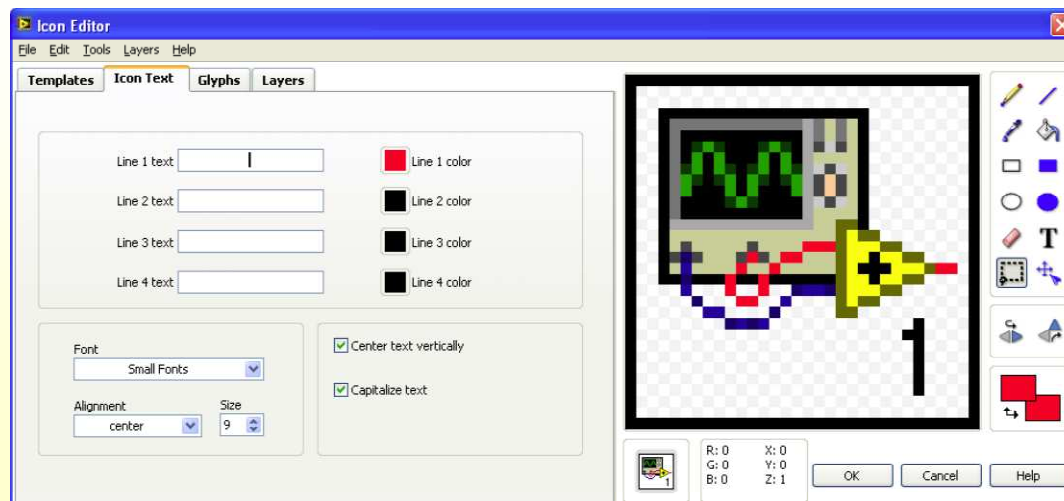
- kreiranje konektora -



# Kako se pravi Sub vi?

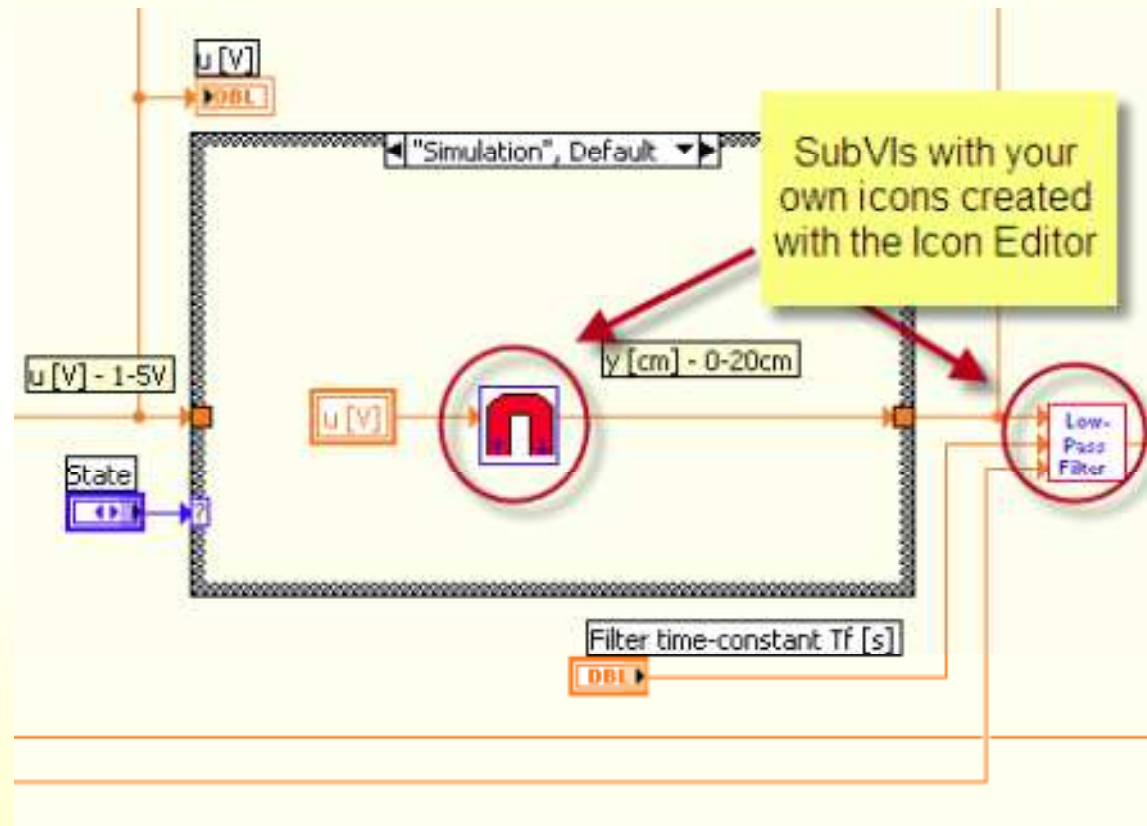
## - kreiranje ikone -

- Ikona se koristi kao vizuelni identifikator vašeg potprograma na blok dijagramu.
- Savet je da ikona svojim izgledom (slikom) ukazuje na to šta je namena tog potprograma.
- Do ikone se dolazi istim desnim klikom kao i u slučaju kreiranja konektora na ikonu u gornjem desnom uglu, samo biranjem opcije “Edit icon”.



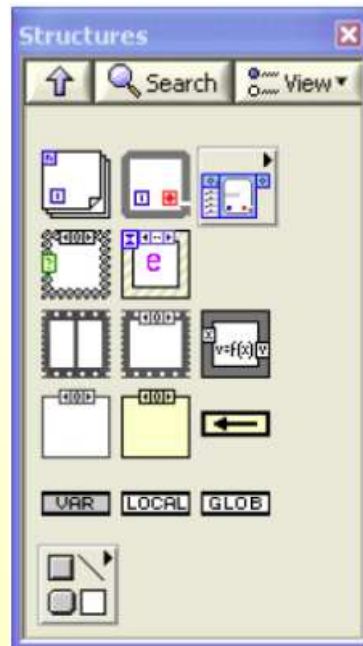
# Kako se pravi Sub vi?

- blok dijagram sa formiranim ikonama za Sub vi -



# Petlje i strukture

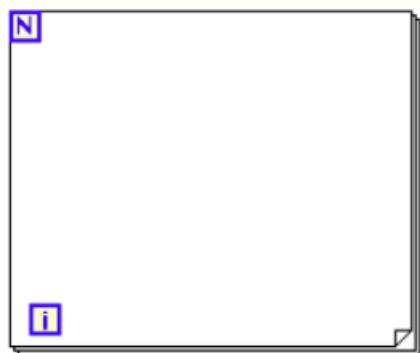
- **Petlje i strukture:**
  - For loop
  - While loop
  - Sekvenca
  - Case struktura
  - Formula node
- Sve su dostupne u okviru Functions palette (podmeni Structures) u blok dijagramu.



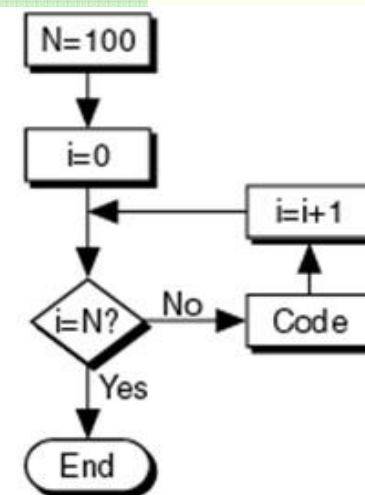
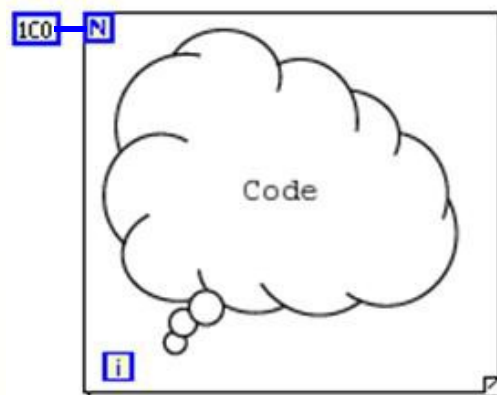
# Petlje i strukture

## - For petlja -

### For petlja - izgled



### For petlja - princip rada



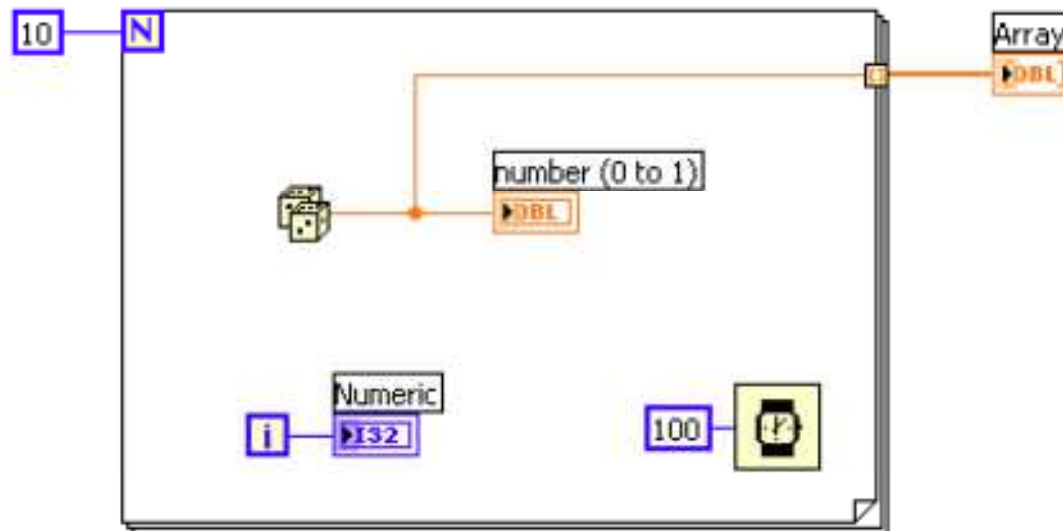
- For petlja izvršava deo programa sadržanog u okviru nje  $n$  puta, gde je  $n$  vrednost upisana u terminal  $N$
- Terminal  $i$  je brojač iteracija,  $i$  pokazuje koja je trenutna iteracija koja se vrši. U LabVIEW-i on uzima vrednosti:

od 0 do  $N-1$

# Petlje i strukture

## - For petlja -

primer: For petlja formira niz od 10 slučajnih brojeva



NATIONAL INSTRUMENTS™  
**LabVIEW™**

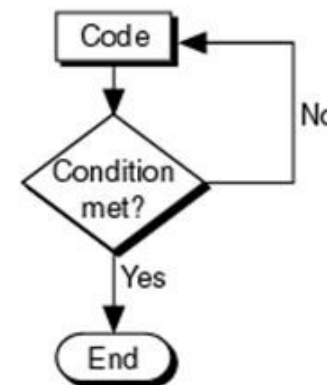
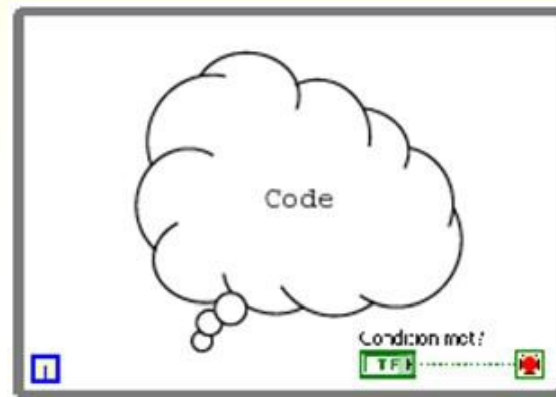
# Petlje i strukture



## - While petlja -

### While petlja - izgled



### While petlja - princip rada



- While petlja izvršava deo programa sadržanog u okviru nje sve dok se ne ispuni unapred zadati uslov. Taj uslov realizuje programer i on se dovodi na ulaz uslovnog terminala 
-  je brojač iteracija kao i kod For petlje
- Uslov zaustavljanja While petlje je ustvari Boolova promenljiva, a samo pravilo zaustavljanja petlje može da se menja:

Stop if True or Continue if True

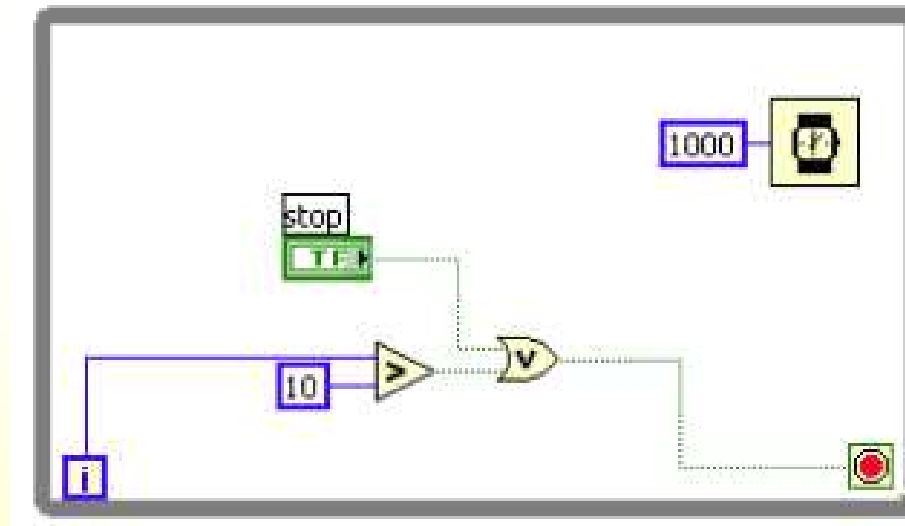


NATIONAL INSTRUMENTS™  
**LabVIEW™**

# Petlje i strukture

## - While petlja -

primer: While petlja se vrši sve dok korisnik ne klikne Stop kontroler na front panelu, ili dok brojač iteracija ne postane veći od 10



NATIONAL INSTRUMENTS™  
**LabVIEW™**

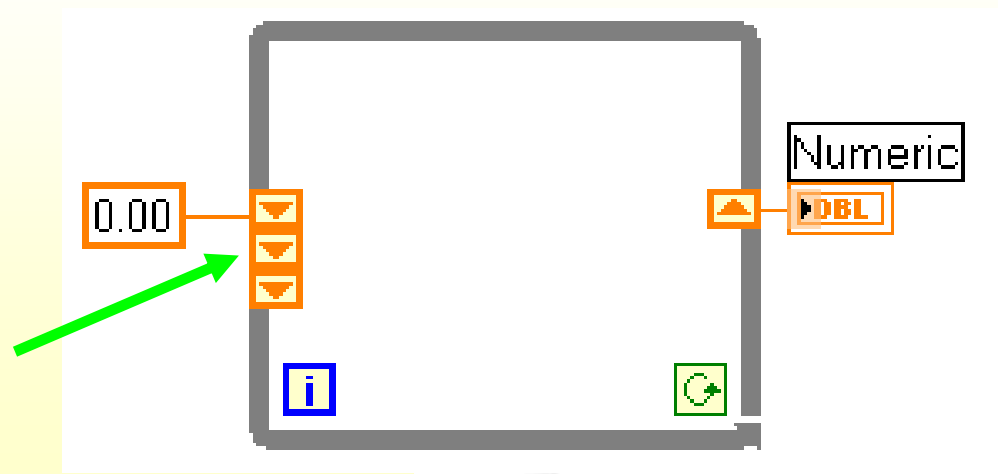


# For i While petlja

## - Shift registri -

- I For i While petlja mogu koristiti tzv. Shift registre, koji im omogućavaju pamćenje i prolazak izabranih vrednosti iz jedne iteracije u drugu.
- Do Shift registra se dolazi desnim klikom na okvir petlje i biranjem opcije “Add Shift Register”.
- Takođe, za pamćenje više rezultata prethodnih iteracija broj ćelija shift registra se može povećavati – *resize* na samu ćeliju shift registra.

primer: While petlja sa tri ćelije shift registra.



# For i While petlja

## - Autoindeksiranje -

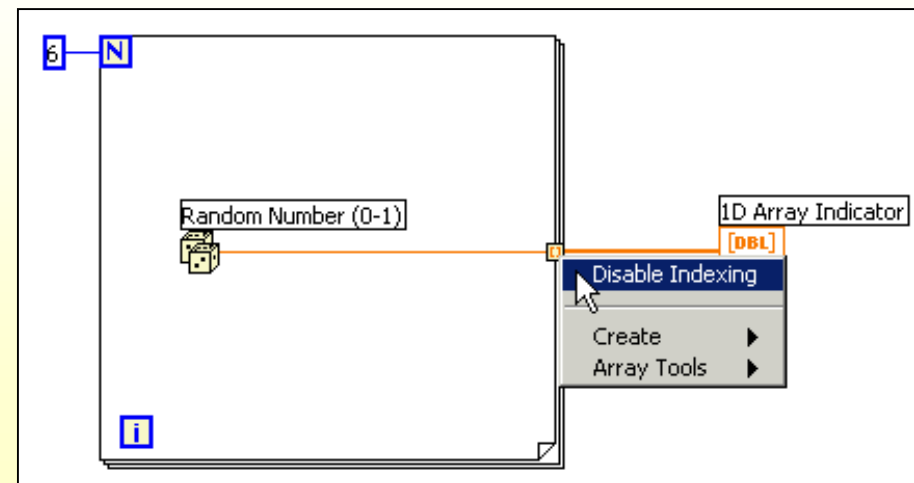
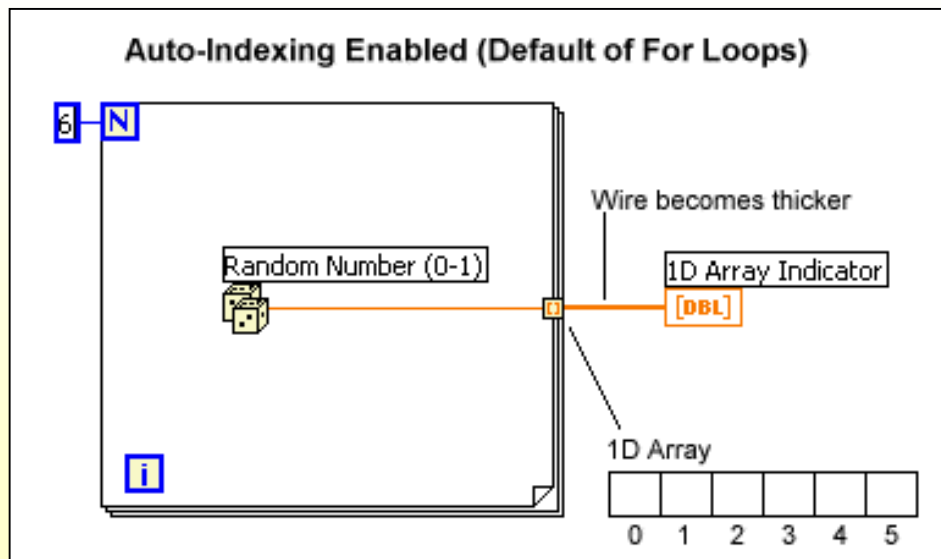
- Osim pomenute razlike For i While petlje (For se izvršava unapred definisan broj puta a While do ispunjenja nekog unapred zadatog uslova), postoji još jedna bitna razlika ove dve petlje a tiče se tzv. **autoindeksiranja**.
- Autoindeksiranje podrazumeva indeksiranje svega što “izlazi” iz petlje – što ima za posledicu pamćenje rezultata svih iteracija petlje i smeštanja istih u niz.
- Kaže se da petlja ima osobinu autoindeksiranja ako prethodni postupak indeksiranja radi “po definiciji” – tj. onako kako je izvučemo iz Functions palete (bez naknadnih podešavanja).



# For i While petlja

## - Autoindeksiranje -

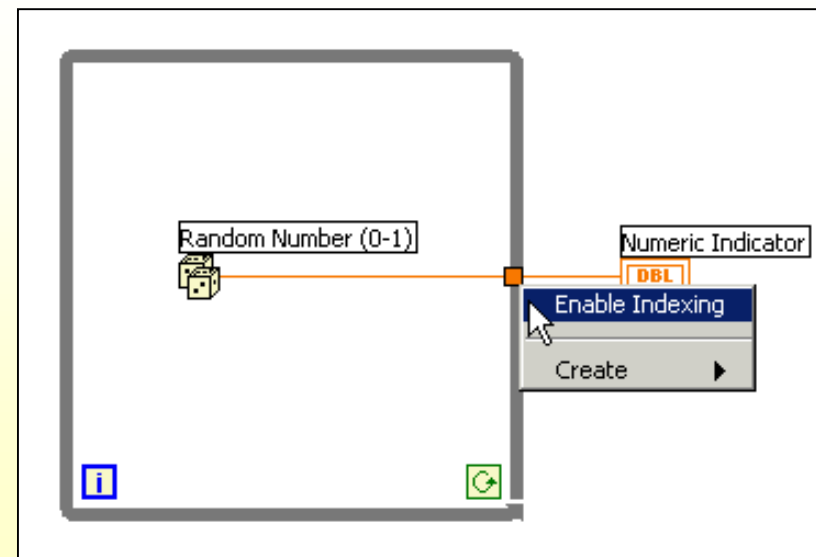
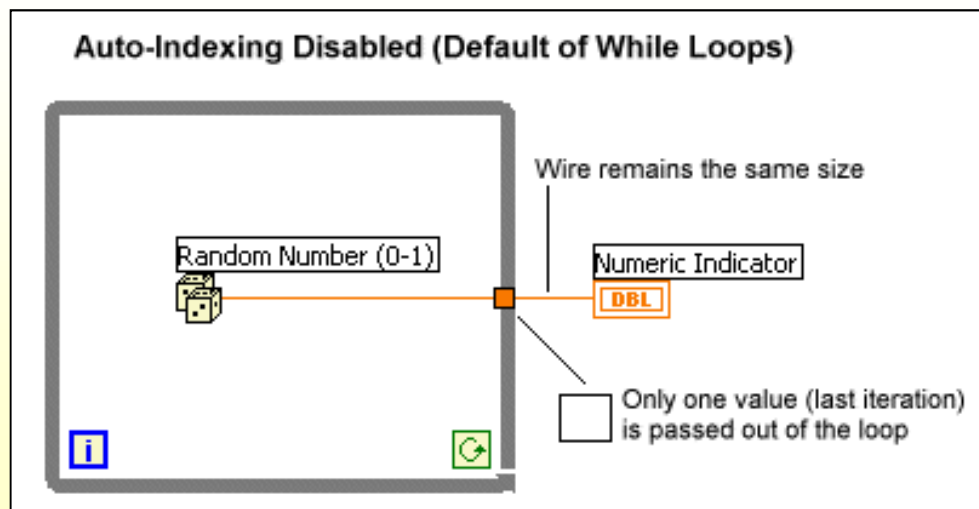
- U LabVIEW-u osobinu autoindeksiranja **po definiciji ima For petlja!**
- For petlja može postati petlja bez autoindeksiranja dodatnim podešavanjem. To se realizuje desnim klikom na “tunel” i biranjem “Disable Indexing”



# For i While petlja

## - Autoindeksiranje -

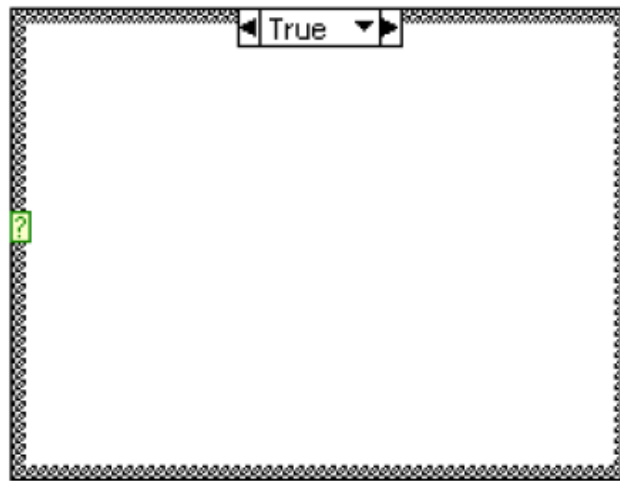
- **While petlja po definiciji nema** osobinu autoindeksiranja!
- While petlja može postati petlja sa autoindeksiranja dodatnim podešavanjem. To se realizuje desnim klikom na “tunel” i biranjem “Enable Indexing”



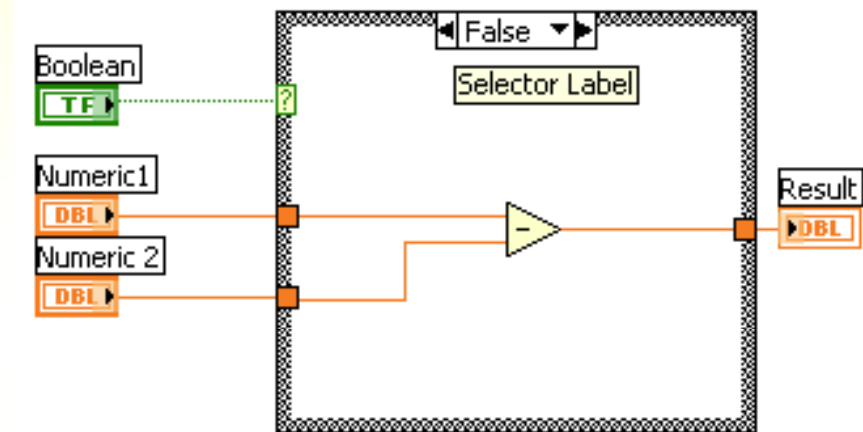
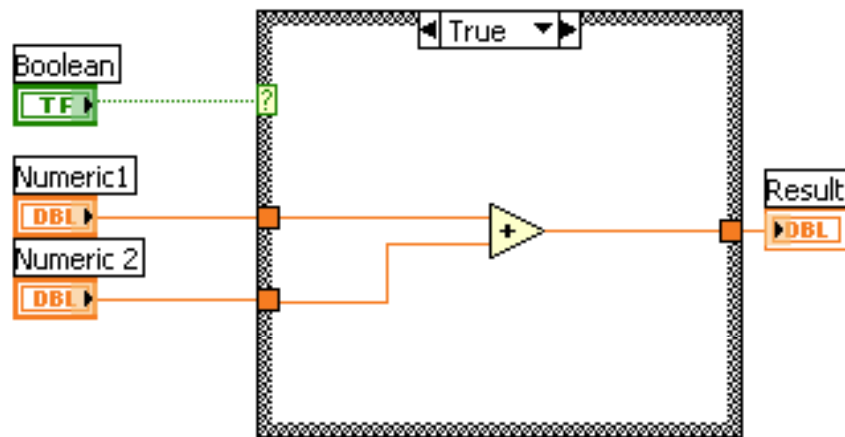
# Case struktura

- **Case** struktura se sastoji od nekoliko slučajeva (poddijagrama u okviru rama Case strukture) od kojih se od svih slučajeva izvršava samo jedan!
- Koji će se slučaj (deo programa) izvršiti, zavisi od toga šta je na ulazu tzv. uslovnog terminala (selektora).
- Selektor može biti Bulova promenljiva, integer, string.

uslovni terminal



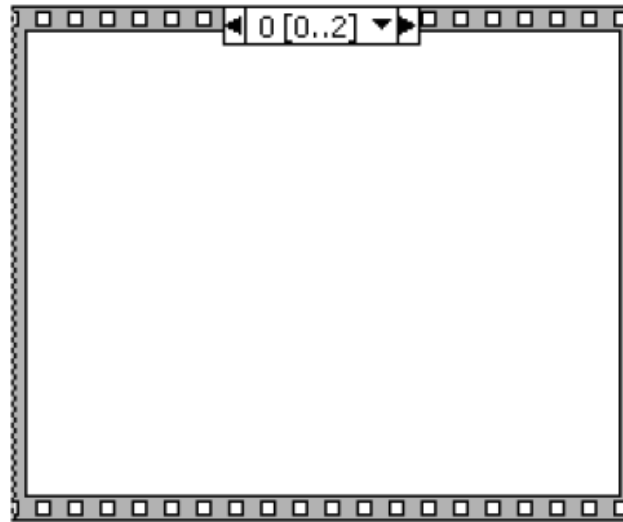
# Case struktura



- Primer Case strukture kada je selektor Bulova promenljiva.
- Kako ona može imati vrednosti True ili False, postoje i dva slučaja koja se mogu izvršavati.
- Ako je vrednost selektora True vrši se sabiranje dva broja, a ako je False vrši se oduzimanje dva broja!

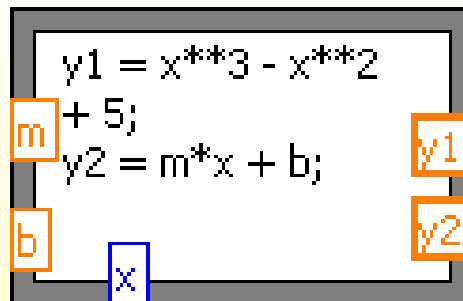
# Sequence struktura

- **Sequence** je bitna struktura u LabVIEW-u koja omogućava da se definiše željeni redosled izvršavanja delova programa.
- Izgled joj podseća na filmsku traku, kako bi se vizuelno ukazalo da se delovi programa po ramovima izvršavaju sekvencijalno (0, 1, 2, ...)



# Formula Node struktura

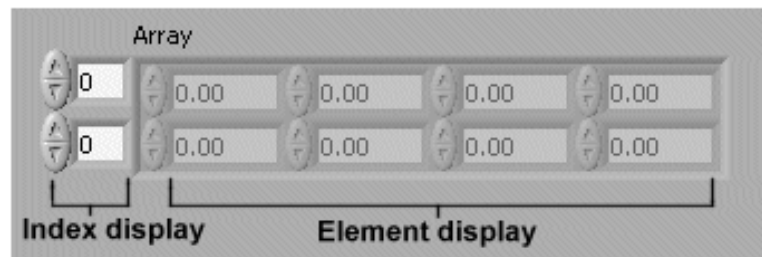
- **Formula node** je struktura u Lab VIEW koja se koristi radi povećanja preglednosti programa.
- Koristi se kada je potrebno izračunati neku vrednost primenom formula a da bi se izbeglo ubacivanje većeg broja sabirača, množača, ...
- Željena formula se u tom slučaju jednostavno otkuca u strukturi Formula Node.





# Arrays - nizovi

- Kao i u drugim programskim jezicima, nizovi predstavljaju skupove podataka istog tipa.
- Kontroleri/indikatori niza se sastoje se iz dva dela:
  - index display: pokazuje indeks elementa
  - element display: pokazuje vrednost elementa
- Važno je napomenuti da se nizovi u LabVIEW-u indeksiraju od nule!



2D Array

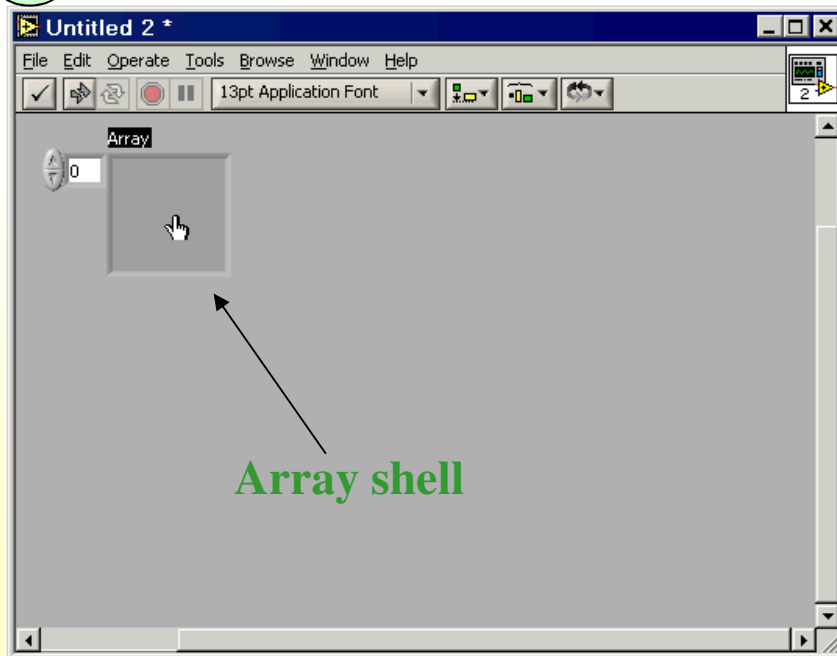
	0	1	2	3	4	5	6
0							
1							
2							
3							
4							

Five-row by seven-column  
array of 35 data elements

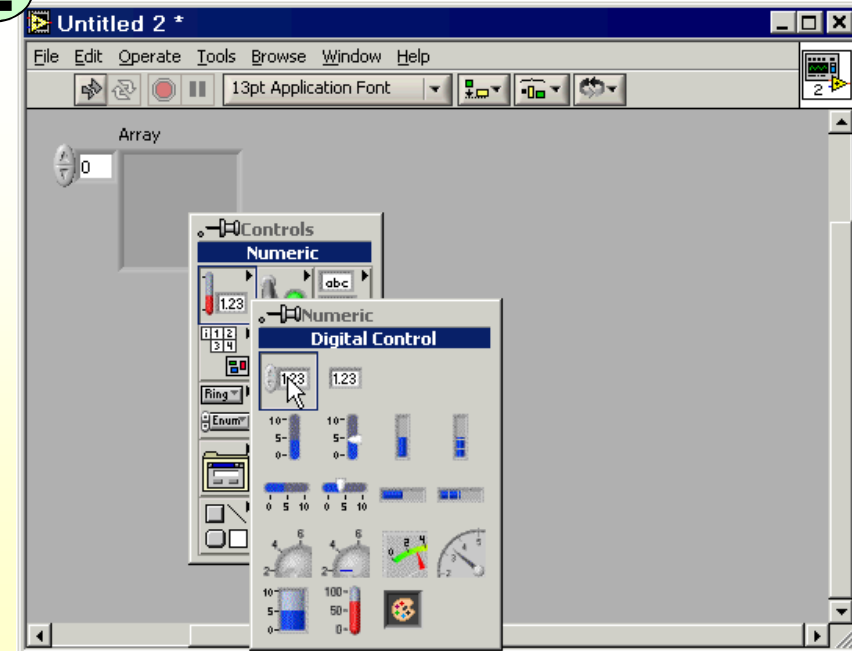
# Arrays - nizovi

- Za razliku od kontrolera/indikatora ostalih tipova podataka (koji se samo prebace iz odgovarajuće grupe iz Controls palette), nizovi se u LabVIEW-u unose u dva koraka:
  1. unos kontrolera/indikatora niza tzv. Array shell (specificirate niz)
  2. unos kontrolera/indikatora tipa podataka u nizu (numerički, bulovi, ... – specificirate tip podataka u nizu)

1

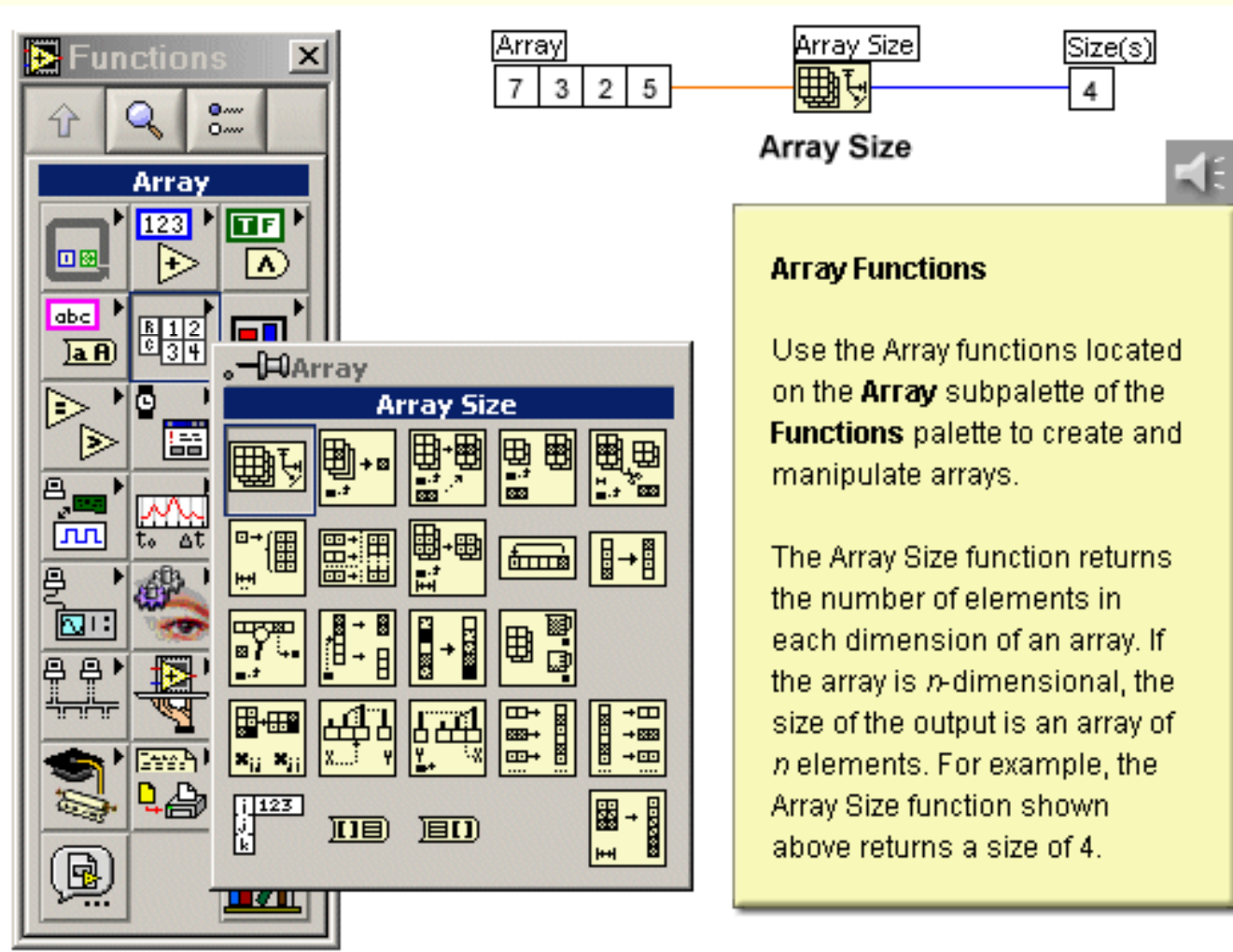


2



# Arrays – nizovi

- bitnije funkcije sa nizovima -



The image shows a screenshot of the NI Instruments Functions palette. The 'Array' subpalette is selected, showing various array-related functions. The 'Array Size' function is highlighted, and its output is shown as a box containing the number '4'. A diagram above the function shows an input array [7, 3, 2, 5] connected to the 'Array Size' function, which outputs '4'. A text box on the right explains the function's purpose and provides an example.

**Array Functions**

Use the Array functions located on the **Array** subpalette of the **Functions** palette to create and manipulate arrays.

The Array Size function returns the number of elements in each dimension of an array. If the array is  $n$ -dimensional, the size of the output is an array of  $n$  elements. For example, the Array Size function shown above returns a size of 4.

# Arrays – nizovi

- bitnije funkcije sa nizovima -

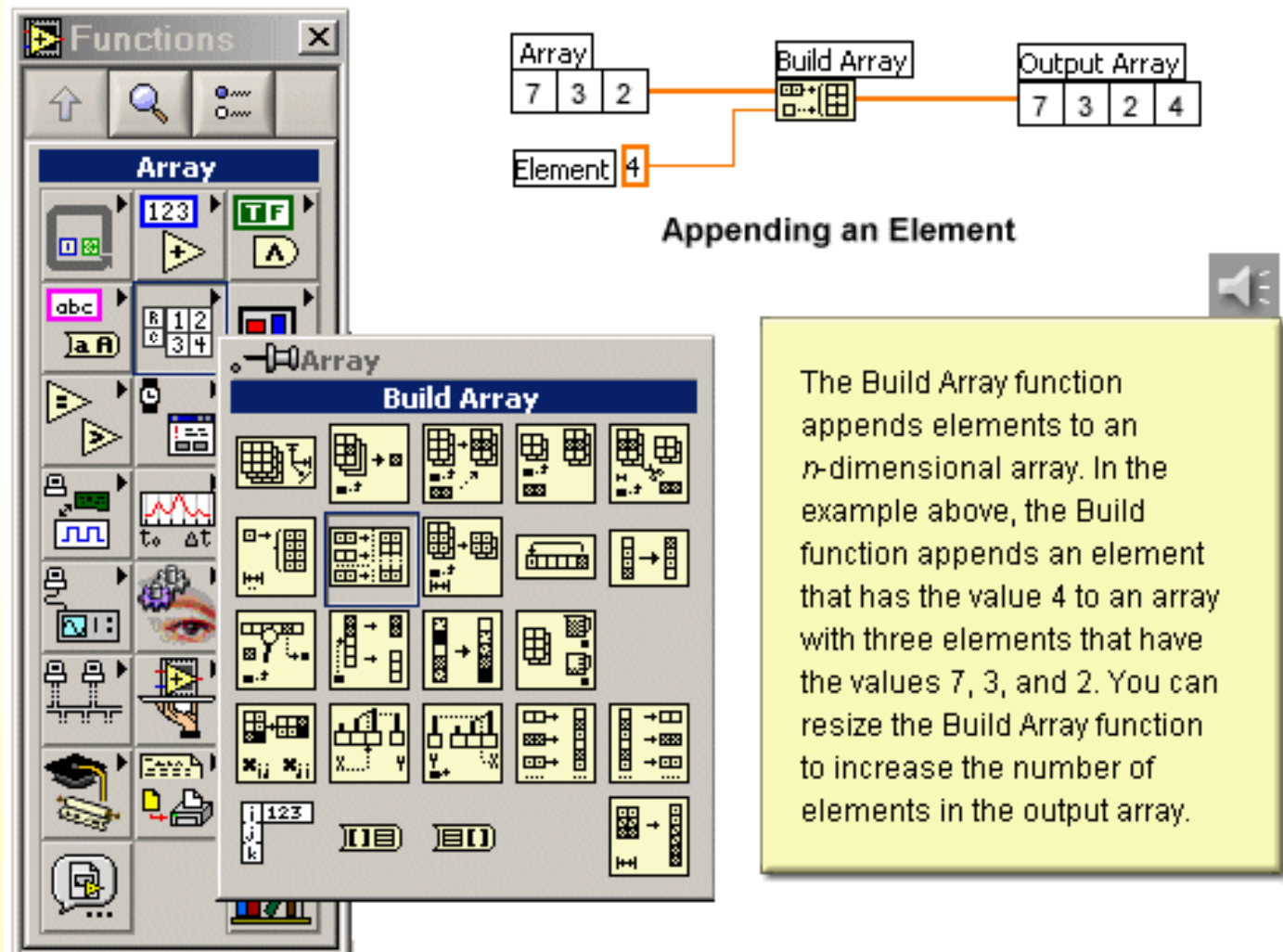
The screenshot shows a software interface with a 'Functions' panel on the left. The 'Array' category is selected, and the 'Initialize Array' function is highlighted. A diagram to the right illustrates the function's operation: 'Element Data' is set to 4, 'Dimension Size' is set to 3, and the 'Output Array' contains three 4s. A text box explains that the function creates an  $n$ -dimensional array where every element is initialized to the value specified at the Element Data terminal.

**Initialize Array**

The Initialize Array function creates an  $n$ -dimensional array in which every element is initialized to the value that you specify at the Element Data terminal of the function. In the example above, the Initialize Array function returns an array with 3 elements. Each element has the value 4. You can resize the Initialize Array function to increase the number of dimensions of the output array.

# Arrays – nizovi

- bitnije funkcije sa nizovima -



The image displays a software interface with a 'Functions' panel on the left and a diagram on the right. The 'Functions' panel is divided into two sections: 'Array' and 'Build Array'. The 'Array' section contains various icons for array operations, including a numeric array '123', a string array 'abc', and a 2x2 matrix. The 'Build Array' section shows a grid of icons for different ways to construct arrays, with one icon highlighted. The diagram on the right, titled 'Appending an Element', shows an 'Array' with elements 7, 3, and 2. An 'Element' with the value 4 is being added to the 'Build Array' function block. The output of this function is an 'Output Array' with elements 7, 3, 2, and 4.

Functions

Array

Build Array

Array

7 3 2

Build Array

Output Array

7 3 2 4

Element 4

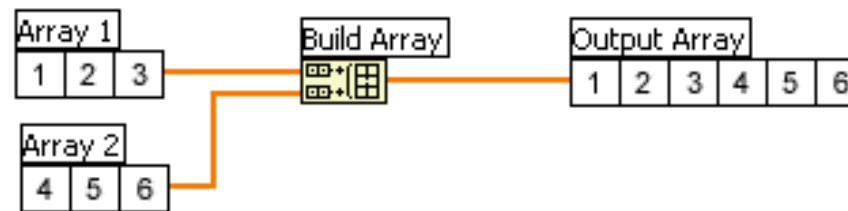
Appending an Element

The Build Array function appends elements to an  $n$ -dimensional array. In the example above, the Build function appends an element that has the value 4 to an array with three elements that have the values 7, 3, and 2. You can resize the Build Array function to increase the number of elements in the output array.

MENTS™  
EW™

# Arrays – nizovi

- bitnije funkcije sa nizovima -



**Concatenate Inputs (Default)**

By default, the Build Array function concatenates multiple arrays. To concatenate the inputs into a longer array of the same dimension as shown above, right-click the Build Array function node, and then click **Concatenate Inputs** from the shortcut menu.

# Arrays – nizovi

- bitnije funkcije sa nizovima -

The screenshot shows a software interface with a 'Functions' panel on the left. The 'Array' category is selected, and the 'Array Subset' function block is highlighted. A diagram illustrates the function's operation: an 'Input Array' [1, 2, 7, 3, 2, 5, 6] is processed with an 'Array Index' of 2 and an 'Array Length' of 4 to produce an 'Output Array' [7, 3, 2, 5].

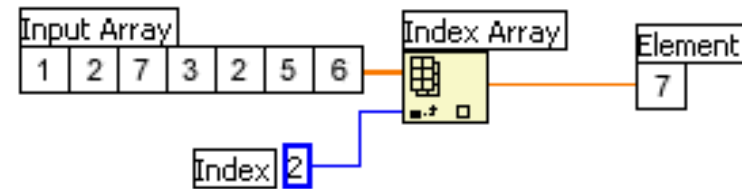
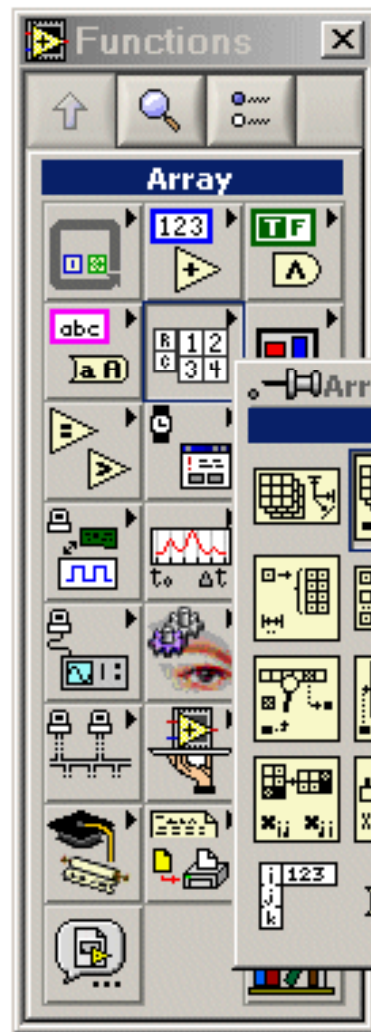
**Array Subset**

The Array Subset function returns a portion of an array. To specify the starting point, supply the index location of the starting element at the Array Index terminal of the function. To specify the length of the output array, supply a value at the Array Length terminal of the function. In the example above, the Array Subset function starts with the third element of the input array, which has the value 7, and produces an output array with 4 elements. **Tip:** Remember that an array index is zero-based: the first element is at index location 0.

UMENTS™  
**EW**™

# Arrays – nizovi

- bitnije funkcije sa nizovima -



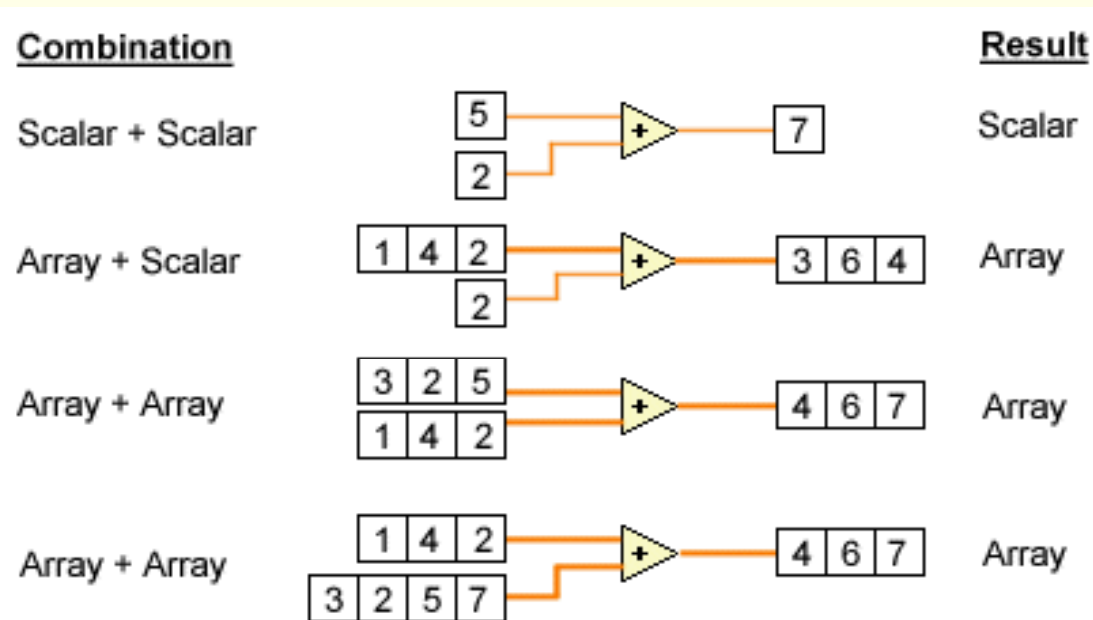
Extracting an Element

The Index Array function returns an element of an array at the index location that you specify for the Index terminal of the function. In the example above, the index location is 2. The Index Array function returns the value 7, which is the third element in the input array.



# Arrays – nizovi

- bitnije funkcije sa nizovima -

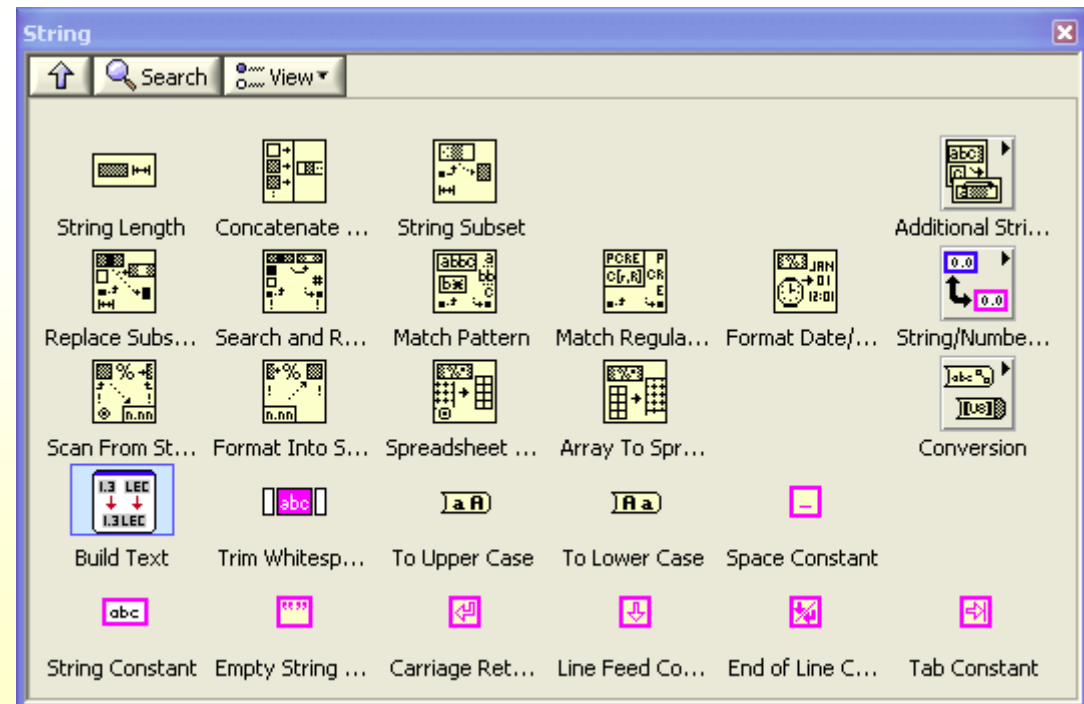
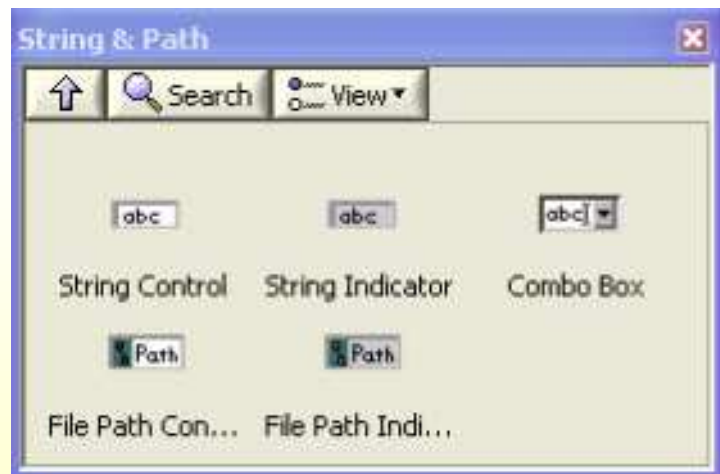


polimorfizam

When you add two arrays of different sizes, the Add function returns an array that is the size of the smaller input array, as shown in the fourth example above.

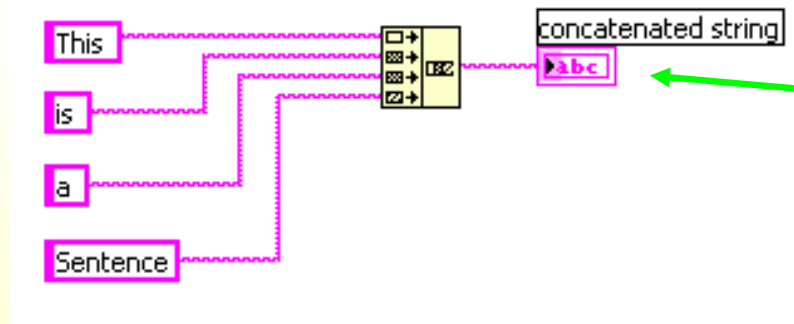
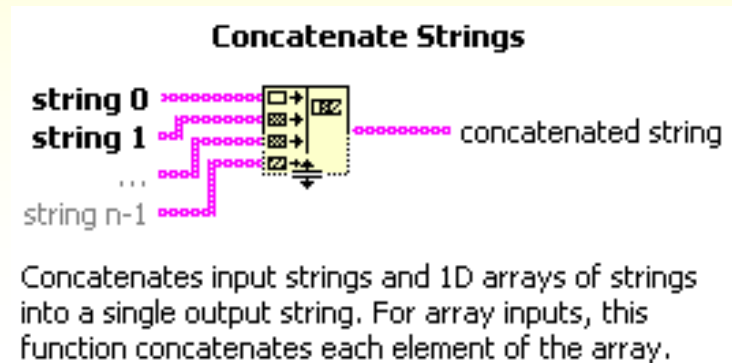
# Strings

- Kao i ostali tipovi podataka, i stringovi imaju svoje kontrolere i indikatore.
- Oni su dostupni u okviru Controls palette – grupa String & Path.
- U okviru blok dijagrama, funkcije za manipulaciju stringovima dostupne su u okviru Functions palette, grupa String.



# Strings

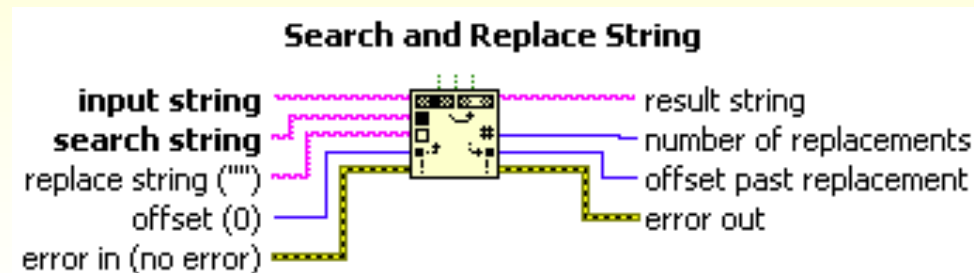
- bitnije funkcije sa stringovima -



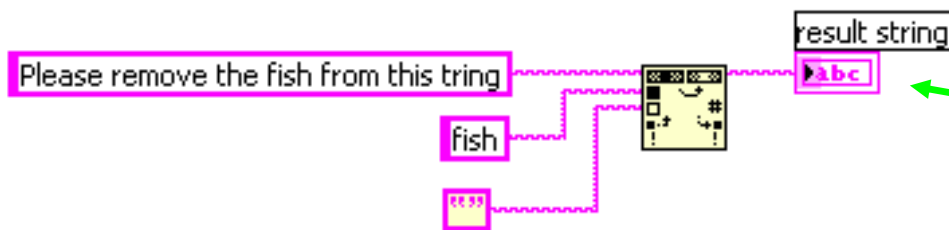
sadržaj indikatora:  
**This is a Sentence**

# Strings

- bitnije funkcije sa stringovima -



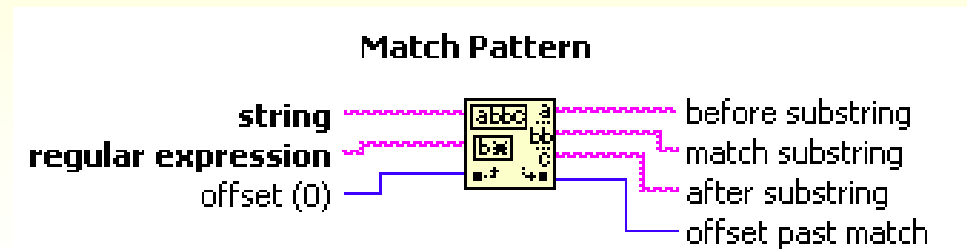
Replaces one or all instances of a substring with another substring. To include the **multiline?** Boolean input, right-click the function and select **Regular Expression**.



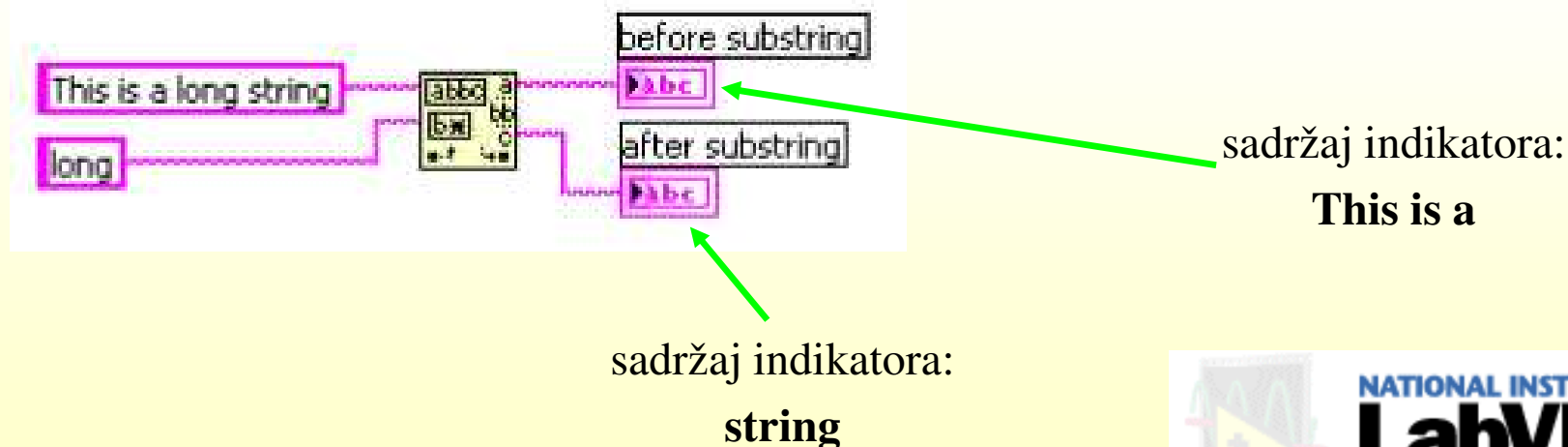
sadržaj indikatora:  
Please remove the from this tring

# Strings

- bitnije funkcije sa stringovima -

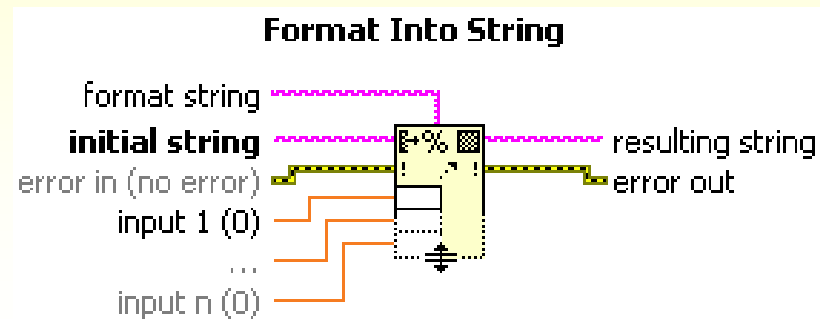


Searches for **regular expression** in **string** beginning at **offset**, and if it finds a match, splits **string** into three substrings. A regular expression requires a specific combination of characters for pattern matching. For more information about special characters in regular expressions, refer to the **regular expression** input description in the detailed help.

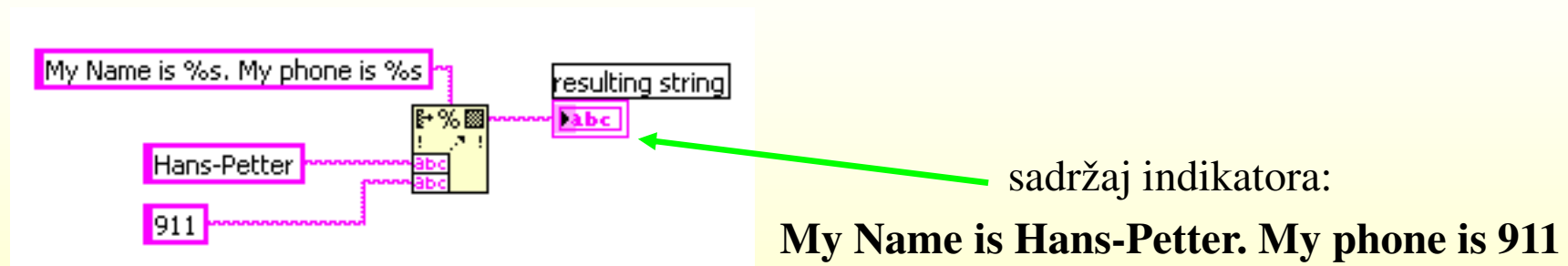


# Strings

- bitnije funkcije sa stringovima -



Formats string path, enumerated type, time stamp, Boolean, or numeric data as text.



Hvala na pažnji!

