

Budowa i oprogramowanie komputerowych systemów sterowania

## Laboratorium 1

Wprowadzenie do Labview

LabVIEW jest narzędziem firmy National Instruments. Jest to graficzne środowisko programowania, oparte o język G. W odróżnieniu od często spotykanych środowisk programowania - LabVIEW oparty jest o koncepcję łączenia ikon. Ikony określają na ogół funkcje. Korzystając z LabVIEW nie zachodzi zatem potrzeba pisania jakiegokolwiek kodu źródłowego, a jedynie jego graficzne przedstawienie za pomocą ikon.

Połączone ikony tworzą diagram obrazujący przepływ danych - od źródła informacji - do końca. Źródłem informacji może być zewnętrzne urządzenie elektroniczne lub dane wprowadzone przez użytkownika (na diagramie reprezentowane jest odpowiednią ikoną). Analogicznie jest z wyjściem. Dane przetworzone mogą zostać zwrócone do zewnętrznego urządzenia, bądź mogą zostać wyświetlone na ekranie monitora.

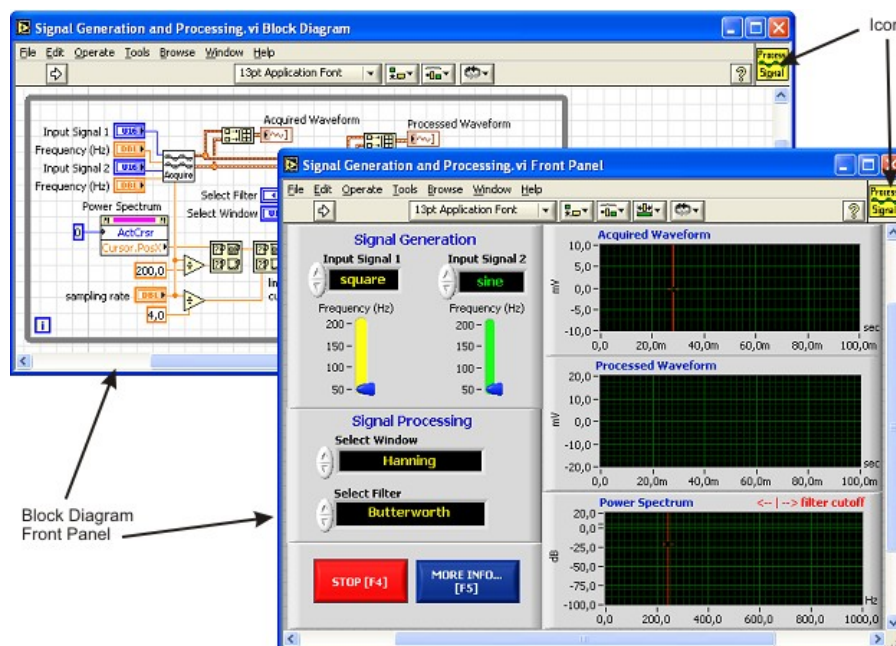
Możliwość łatwej współpracy programu LabVIEW z urządzeniami zewnętrznymi - pozwala na traktowanie komputera jako urządzenia kontrolno-pomiarowego.

Labview jest narzędziem firmy National Instruments, służącym do tworzenia programów. Istotą różnicy względem standardowych narzędzi jest: sposób opracowywania programów - nie są one pisane w sekwencyjny sposób; projekt stanowi rysunek przepływu informacji  
łatwa komunikacja z zewnętrznymi urządzeniami pomiarowymi  
Programy stworzone przez Labview nazywane są Wirtualnymi Narzędziami (VI - Virtual Instrument). Przetwarzają one (najczęściej) dane pomiarowe pozyskane z interfejsów komputera. Tak więc komputer stanowi niejako instrument - przedstawiający opracowane dane, czy też podejmujący ściśle określone zadanie.

Standardowo projekt tworzony jest w dwóch oknach:

- Panelu frontowy
- Diagramie blokowym

Istotna jest również ikona opisująca program (VI).



Panel frontowy nich stanowi interfejs graficzny przewidziany dla użytkownika. Diagram blokowy opisuje sposób przepływu informacji, źródło ich pozyskania itp. Diagram blokowy nie jest

widzialny dla użytkownika programu.

Zatem standardowym sposobem tworzenia aplikacji: będzie stworzenie kontrolki na panelu frontowym. Z ich pomocą użytkownik będzie mógł wciskać przyciski, jak również oglądać wszelkiego rodzaju wykresy i wartości przedstawiane przez program. Kontrolki na panelu frontowym są ściśle powiązane z elementami diagramu blokowego. Umieszczenie na panelu frontowym elementu spowoduje pojawienie się nowego elementu w diagramie blokowym. Diagram blokowy opisuje sposób przepływu informacji, oraz funkcjonowania algorytmu. Zachowanie się kontrolki na panelu frontowym jest właśnie określone przez diagram blokowy.

## **Palety panelu frontowego**

Po rozpoczęciu pierwszego projektu (Blank VI) otrzymujemy poniższy ekran. W celu zapoznania się z paletami - otworzymy je. Wybieramy z zakładki:

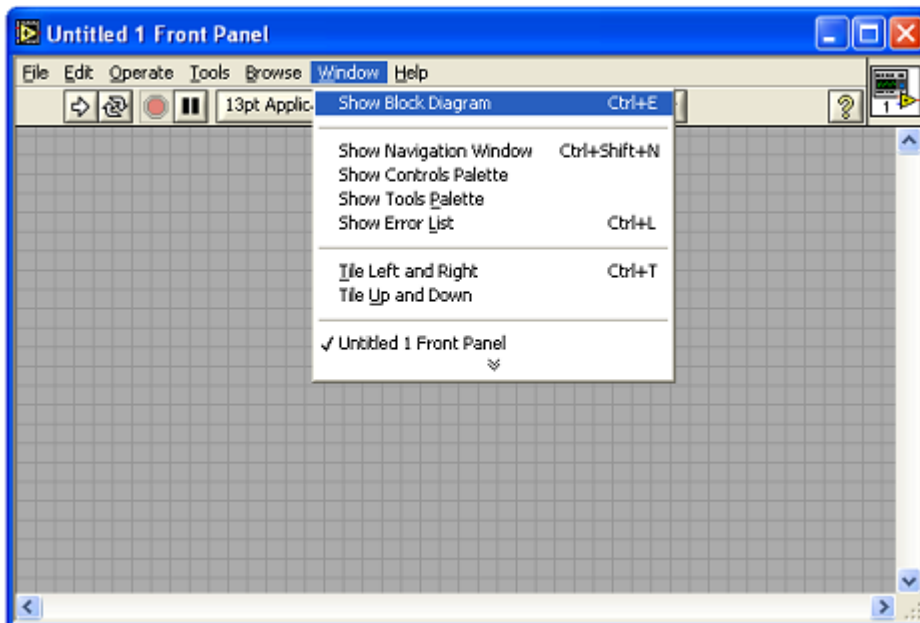
Menu -> Window -> Navigation Window

Menu -> Window -> Tools Palette

Menu -> Window -> Controls Palette

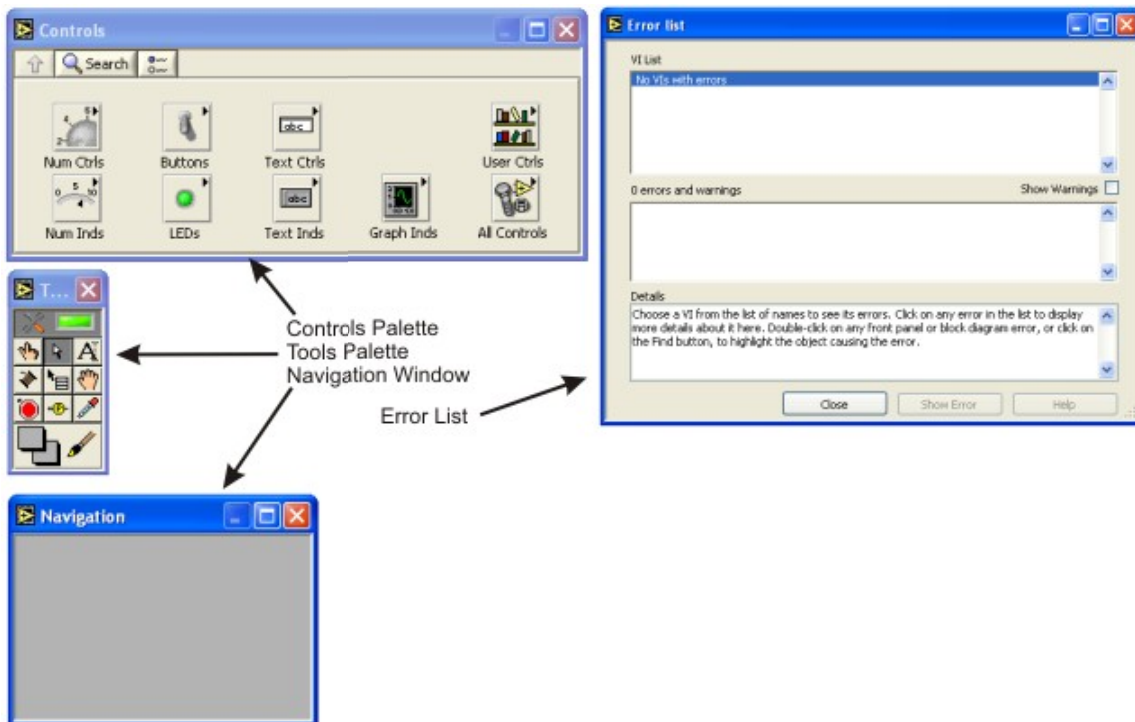
Menu -> Window -> Show Errors List

Później przejdziemy do diagramu blokowego.



LabVIEW - Panel frontowy

Wymienione okienka/palety używane są do tworzenia i nawigacji projektu. Są one nieodzownym elementem pracy z LabVIEW. Przedstawiono je na kolejnym rysunku.

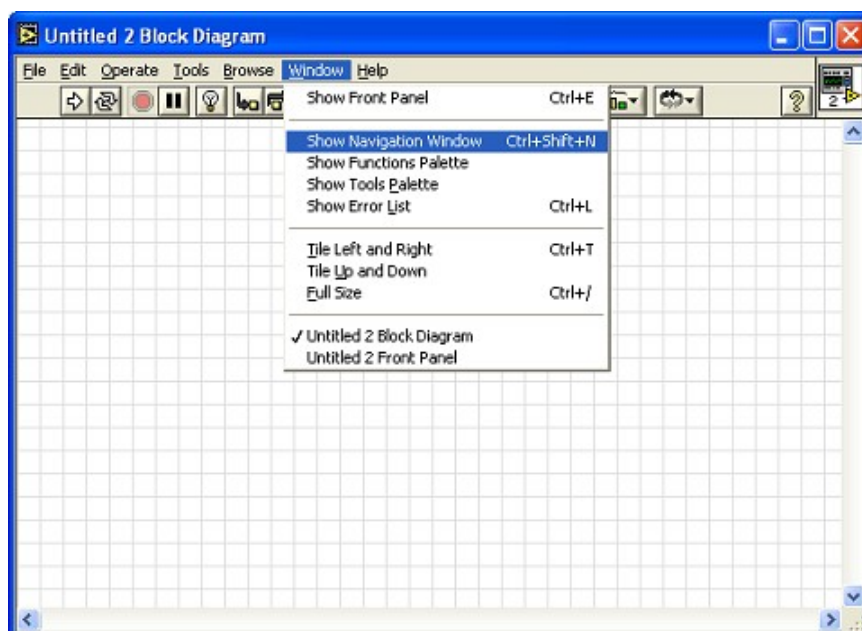


Okna które możemy pokazać to:

- Navigation Window - okno pokazuje rzut projektu w pomniejszeniu
- Controls Palette - paleta z funkcjami które możemy użyć do projektu
- Tools Palette - paleta z narzędziami do manipulowania elementami
- Error List - lista błędów w projekcie

## Panele diagramu blokowego

Jak wspomniano - program tworzony jest w dwóch oknach (panelu frontowym i diagramie blokowym). W celu przejścia do diagramu blokowego należy skorzystać z opcji menu: Menu -> Window -> Show Block Diagram lub wcisnąć skrót klawiszowy CTRL+E.



Analogicznie jak poprzednio - przyjrzymy się paletom diagramu blokowego. Wybieramy z zakładki:

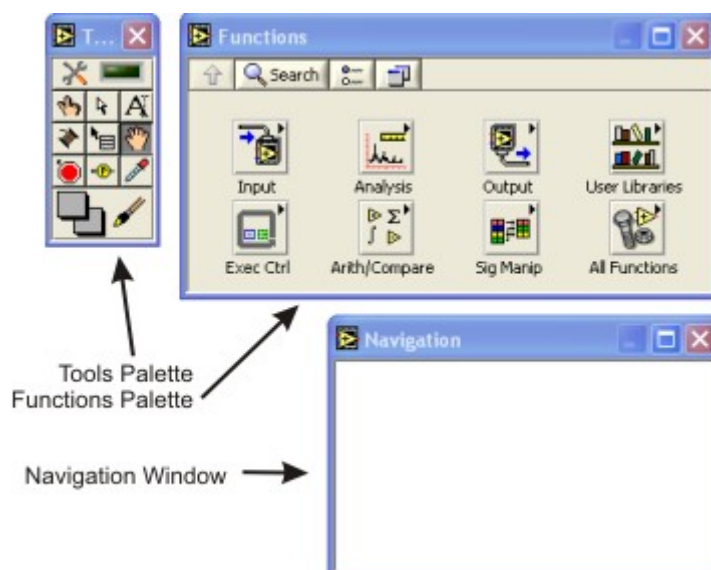
Menu -> Window -> Navigation Window

Menu -> Window -> Tools Palette

Menu -> Window -> Functions Palette.

Okno Show Errors List jest takie samo dla diagramu blokowego - jak i dla panelu frontowego, więc pominiemy je.

Należy zwrócić przede wszystkim uwagę na Functions Palette. Funkcją tej palety jest dostarczenie elementów służących do tworzenia algorytmu. W panelu Functions Palette diagramu blokowego znajdziemy m.in. operacje arytmetyczne, logiczne, pętle i inne elementy służące do programowania. Efekt obliczeń może trafić do elementów ekspozycyjnych panelu frontowego (chcąc wyświetlić wynik - wystarczy podłączyć terminal zawierający interesujący nas wynik do wskaźnika).



Navigation Window - analogicznie jak poprzednio okno pokazuje rzut projektu w pomniejszeniu

Functions Palette - paleta z funkcjami które możemy użyć do projektu

Tools Palette - analogicznie jak paleta z narzędziami do manipulowania elementami diagramu blokowego

## **LabVIEW - Kontrolki, wskaźniki stałe**

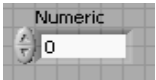
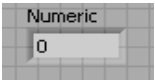



W poprzednim rozdziale stosowaliśmy kilka rodzajów kontrolki. Jedne z nich pobierały dane, inne wyświetlały. Warto podzielić w jakiś sposób typy danych stosowanych w Labview.

kontrolka (Control, np. Numeric Control) - służy do ustalania wartości; ona może być źródłem danych

wskaźnik (Indicator, np. Numeric Indicator) - służy do wyświetlania wartości (np. wartości numerycznej)

stała (Constant, Numeric Constant) - jest źródłem informacji; w przeciwieństwie do dwóch pierwszych - stałe występują tylko i wyłącznie na diagramie blokowym

Dla przykładu - posłużmy się wartością liczbową (Numeric). Poniżej przedstawione są różnice w ich wyglądzie

	<b>Numeric Control</b> kontrolka	<b>Numeric Indicator</b> wskaznik	<b>Numeric Constant</b> stała
<b>Panel Frontowy</b>			<b>Brak</b>
<b>Diagram blokowy</b>			
<b>Działanie</b>	Pozwala użytkownikowi wprowadzić wartość; zostaje ona przekazana do kolejnych funkcji. Na panelu frontowym widzimy po lewej stronie przyciski umożliwiające zmianę wartości. Mały trójkąt znajdujący się na ikonie diagramu blokowego (z prawej strony) - symbolizuje że funkcja jest źródłem danych)	Wyświetla dane na panelu frontowym. Na panelu frontowym - brak przycisków pozwalających na zmianę zawartości komórki.  Mały trójkącik znajdujący się na ikonie diagramu blokowego (z lewej strony) - symbolizuje że funkcja przyjmuje dane)	Stała dana; jest ona przekazywana do programu; nie ma żadnej reprezentacji na panelu frontowym  Nie ma trójkącika, ale podobnie wyglądający obiekt jest zawsze źródłem informacji.

Aby zmienić kontrolkę (Control) na wskaźnik (Indicator) należy kliknąć prawym przyciskiem myszy na interesującą nas ikonę i wybrać z menu opcję "Change to Indicator". W odwrotną stronę - klikamy na element, lecz tym razem wybieramy opcję "Change to Control".

Na diagramie blokowym - klikając na interesującą nas ikonę - możemy konwertować ją w analogiczny sposób do jednej z dwóch pozostałych typów. Dostępna jest opcja konwersji na stałą ("Change to Constant"), której nie zobaczymy na panelu frontowym.

## **LabVIEW - Typy danych**

Jak mogliśmy zauważyć w poprzednich przykładach - każdy typ danych ma charakterystyczny dla siebie kolor na diagramie blokowym. Kolor ten jest taki sam dla przewodów (Wire) łączących terminale określonych typów.

Do tej pory mieliśmy okazję zauważyć że kolor danej boolowskiej (boolean) jest zielony, zaś danej zmiennoprzecinkowej (double) - pomarańczowy. Z czasem będą pojawiać się kolejne kolory - takie jak chociażby różowy, oznaczający łańcuch.

 <b>unsigned byte</b> (bajt (bez znaku), 8-bit.)	 <b>byte stream</b> (strumień bajtów)
 <b>unsigned word</b> (słowo (bez znaku), 16-bit.)	 <b>enum</b> (typ enum)
 <b>unsigned long</b> (podwójne słowo (b/znaku), 32-bit.)	 <b>cluster mixed</b> (klaster, różne elementy)
 <b>byte</b> (bajt (ze znakiem), 8-bit.)	 <b>cluster numeric</b> (klaster, el. numeryczne)
 <b>word</b> (słowo (ze znakiem), 16-bit.)	 <b>array 1D</b> (tablica 1-wymiarowa)
 <b>long</b> (podwójne słowo (ze znakiem), 32-bit.)	 <b>array 2D</b> (tablica 2-wymiarowa)
 <b>single</b> (rzecz., pojedyncza precyzja)	 <b>waveform graph</b> (wykres)
 <b>double</b> (rzecz., podwójna precyzja)	 <b>time stamp</b> (znacznik czasu, <64.64>-bit.)
 <b>extended</b> (rzecz., rozszerzona prec.)	 <b>waveform</b> (wykres)
 <b>complex extended</b> (zespolona, rozszerz. precyzja)	 <b>digital waveform</b> (wykres cyfrowy)
 <b>complex double</b> (zesp., podwójna precyzja)	 <b>digital data</b> (dane cyfrowe)
 <b>complex single</b> (zesp., pojedyncza precyzja)	 <b>I/O name</b> (nazwa zasobu wejścia / wyjścia)
 <b>boolean</b> (typ boolowski)	 <b>picture</b> (obraz)
 <b>string</b> (łańcuch znakowy)	 <b>variant</b> (wariant)
 <b>path</b> (ścieżka (dostępu do pliku))	

Typy danych typu byte, word, long - dotyczą liczb całkowitych. Różnią się one długością bitową (8, 16, 32-bity). Występujący przed nimi przedrostek unsigned (bez znaku) oznacza, że dane mogą być jedynie wartościami nieujemnymi.

Danymi rzeczywistymi są typy single, double, extended. Różnicę pomiędzy nimi stanowi ich precyzja.

Zmienne zawierające przedrostek complex oznaczają liczby zespolone

String - określa łańcuch znakowy. Służy on do m.in. do przechowywania tekstu. Może również służyć do przechowywania danych zapisywanych lub odczytywanych z plików. Jest to zaprezentowane w dalszej części kursu.

Path - jest ścieżką dostępu do plików. Z typem tym można się spotkać przy obsłudze plików (otwieranie plików).

Cluster Mixed - klaster w którego skład wchodzi różne typy danych określony jest kolorem różowym. Alternatywnym typem klastra jest klaster liczbowy (Cluster Numeric), w skład którego wchodzi jedynie dane liczbowe (Numeric). Ten ostatni będzie oznaczony kolorem brązowym. Więcej o klastrach będzie można przeczytać w kolejnych częściach kursu.

Tablice jedno i dwu- wymiarowe (Array 1D, Array 2D) posiadają charakterystyczne nawiasy kwadratowe [ ]. W środku nawiasu kwadratowego możemy zobaczyć typ danych jaki dana tablica niesie. W powyższym zestawieniu obie tablice przechowują typ double (DBL). Nawiasy kwadratowe [DBL], [DBL] mogą być pogrubione lub nie. Niepogrubiony nawias kwadratowy oznacza jednowymiarowość tablicy. Pogrubienie ma miejsce w przypadku gdy tablica ma co najmniej 2 wymiary.

Waveform graph - służy do przekazywania danych do tworzenia wykresu (grafu). Możemy zauważyć, że ikona jest podobna do ikony tablicy jednowymiarowej. Niesione dane są bowiem reprezentowane za pomocą tablicy Array 1D. Waveform różni się od Waveform graph tym, że zawiera dodatkowe dane - jakimi są czas startu wykresu, oraz okres próbkowania. Więcej na temat



wykresów będzie można znaleźć w dalszej części kursu. Kolejnym z typów danych służących do tworzenia wykresu jest Digital waveform. Służy on do tworzenia wykresów cyfrowych. Typ danych zawiera czas początku, cyfrowe dane i wszystkie atrybuty cyfrowego wykresu.

Digital data - zawiera dane cyfrowe.

I/O name - przekazuje zasób wejścia/wyjścia, który użytkownik konfiguruje, do komunikacji. Typ ten jest używany do komunikacji m.in. z urządzeniami zewnętrznymi, aparaturą pomiarową.

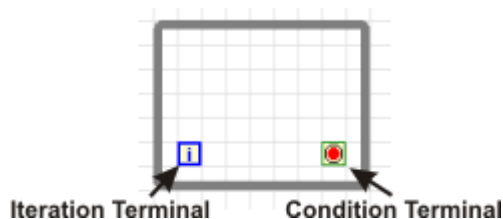
Picture - dane zawierające obraz. LabVIEW pozwala na tworzenie i przedstawianie obrazów. Jest to przydatna opcja - szczególnie w celach wizualizacji niektórych wyników pomiarowych.

## **LabVIEW - Loop(1) - While**

W LabVIEW - podobnie jak w innych językach programowania - można realizować pętle. Do wyboru mamy pętle typu:

- while
- for

Pętle powinny składać się z warunku do kiedy powinna być pętla realizowana (Conditional Terminal - w przypadku pętli while; count number - w przypadku pętli for) oraz z wartości iterowanej (Iteration Terminal). Ta ostatnia jest dostępna wewnątrz pętli.



Warunki zakończenia pętli (Condition Terminal) - pętla while

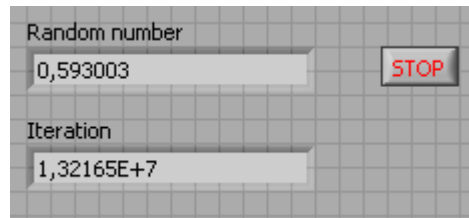
Można je zmienić typ terminala warunku - klikając na niego prawym przyciskiem myszy. Istnieją dwa warianty.

- Stop if true - pętla zakończy się gdy warunek podany do terminala będzie prawdziwy
- Continue if true - pętla będzie trwała dopóki warunek podany do terminala będzie prawdziwy

Przykład

Stwórzmy zatem następujący panel frontowy (pomińmy przycisk stop; w standardowej konfiguracji zostanie on automatycznie dodany w momencie dodawania pętli while do diagramu blokowego).





Panel frontowy loop1vi.vi

Oraz połączmy elementy diagramu blokowego w poniższy sposób.

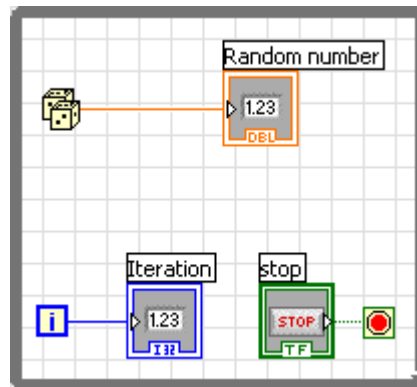


Diagram blokowy loop1vi.vi

Pętla while znajduje się na palecie funkcji (Functions Palette) Exec Ctrl > While Loop.  
Kostka symbolizująca zmienną losową znajduje się (Functions Palette) Arith/Compare > Numeric > Random Num

Po uruchomieniu przykładu (Run) możemy zobaczyć jak szybko wzrasta numer iteracji, oraz zmienne losowe.

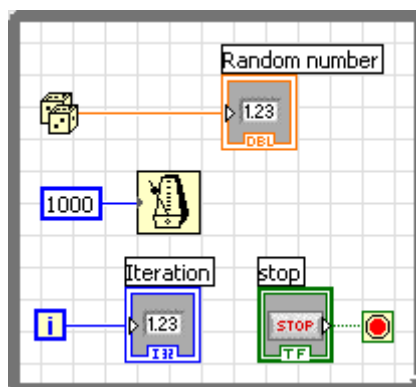
Dodawanie timingu

Pętla kończy iterację w momencie w którym wszystkie jej elementy zostaną wykonane. Oznacza to, że czas wykonania pojedynczej iteracji nie będzie krótszy niż czas trwania najdłuższej wykonywanej funkcji.

Dla przykładu możemy dodać do pętli opóźnienie wynoszące 1 sekundę. Będzie to najdłuższe opóźnienie w pętli, co będzie oznaczało, że pojedyncza iteracja będzie wykonywać się 1 sekundę. (w praktyce na wolnych komputerach może wykonać się nieznacznie dłużej).

Przykład

Dodajmy do poprzedniego VI - element opóźnienia - zgodnie z poniższym diagramem blokowym.  
Diagram blokowy loop2vi.vi



Element opóźniający możemy znaleźć na palecie funkcji All Functions > Time & Dialog > Wait Until Next ms Multiple. Umieszczenie tego elementu spowoduje, że program będzie czekał do kolejnej wielokrotności milisekundowego zegara komputera (wartość zegara komputera można uzyskać przy użyciu funkcji Tick Count z tej samej palety).

Po uruchomieniu zobaczymy, że:  
 pętla będzie oczekiwać na kolejną wielokrotność 1000ms wewnętrznego licznika (1 sekunda)  
 iteracja pętli zaczyna się od wartości 0

## LabVIEW - Loop(2) - For

Funkcję pętli można również realizować przy wykorzystaniu pętli for, dostępnej z palety funkcji. (All Functions > Structures > For Loop). Różnicą pomiędzy pętlą for, a pętlą while jest warunek wykonywania. W przypadku pętli for - z góry deklarujemy ilość iteracji, jaką pętla ma wykonać.

Przykład

Posłużmy się loop2vi.vi.

Usuńmy Przycisk STOP. Zawartość pętli while - wyciągnijmy poza pętlę. Usuńmy pustą pętlę i utwórzmy pętlę for (All Functions > Structures > For Loop). Do pustej pętli należy włożyć zachowane uprzednio elementy. Ilość iteracji (N) należy połączyć ze stałą wartością 10 - zgodnie z poniższym diagramem blokowym.

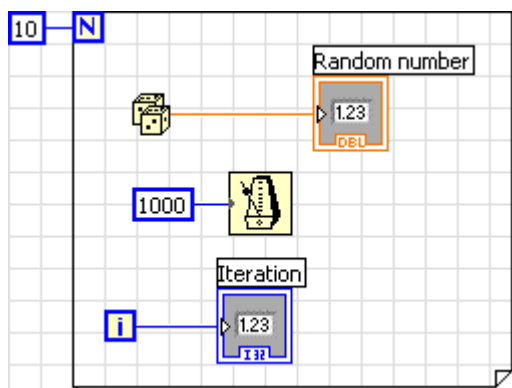


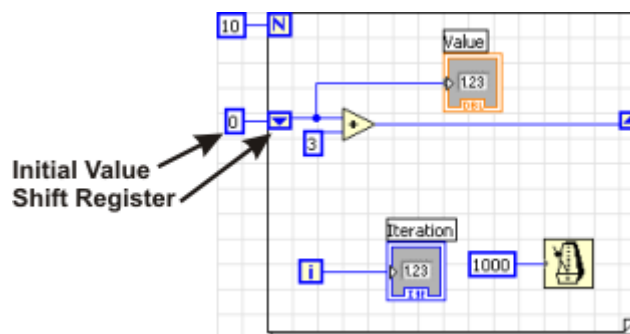
Diagram blokowy loop3vi.vi

Po uruchomieniu przykładu - zobaczymy, że pętla wykona się 10 razy.

## LabVIEW - Loop(3) - Rejestry przesuwne, indeksowanie

Niejednokrotnie zachodzi potrzeba przekazania wartości otrzymanej w iteracji (n) do kolejnej iteracji (n+1) pętli. Można to zrobić korzystając z rejestrów przesuwnych.

W celu dodania rejestru przesuwego do pętli (for lub while) należy kliknąć na lewą krawędź prawym przyciskiem myszy, a następnie wybrać Add Shift Register.

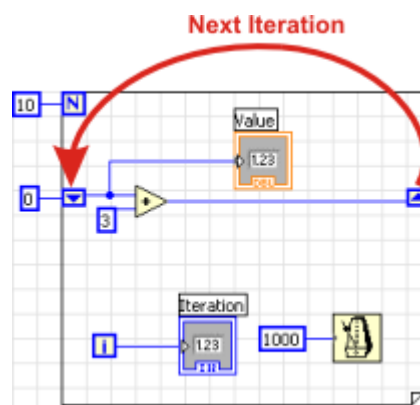


### Rejestr przesuwny

W pierwszej iteracji rejestr przesuwny (Shift Register) jest wypełniany wartością (Initial Value). W kolejnych iteracjach rejestr ten zawiera wartość obliczoną w poprzedniej iteracji.

Przeanalizujemy dwa przypadki.

1. W oparciu o loop2vi.vi należy zbudować poniższy przykład loop4shiftreg.vi.

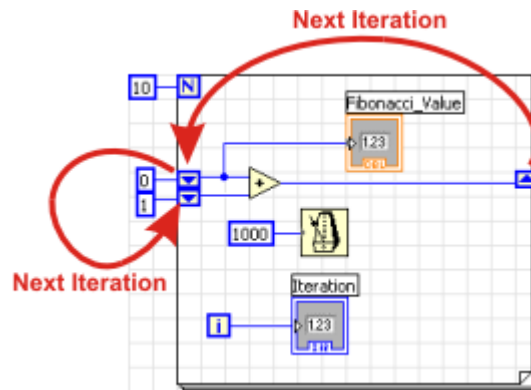


### Rejestr przesuwny - przykład loop4shiftreg.vi

Po uruchomieniu tego przykładu na ekranie będą wyświetlane kolejne wartości: 0, 3, 6, 9, 11, ... 27. Algorytm działania jest prosty: W pierwszym kroku rejestr (z lewej strony) zawiera wartość 0 (Initial Value). Do niej dodawana jest wartość 3 i przekazywana do rejestru po prawej stronie. W kolejnych krokach wartość rejestru z lewej strony jest uzupełniana ostatnio otrzymanym wynikiem (zapisanym w rejestrze po prawej stronie).

Tak więc w każdej kolejnej iteracji do poprzednio otrzymanej wartości dodajemy 3. Zapiszmy przykład pod nazwą loop4shiftreg.vi.

2. Przeanalizujemy bardziej skomplikowany przykład ciągu Fibonacciego. W tym celu należy stworzyć przykład - zgodny z poniższym diagramem blokowym.



Rejestr przesuwany - realizacja ciągu Fibonacciego (loop4fib.vi)

W tym przypadku sposób przekazywania argumentów jest bardziej skomplikowany. Rejestry po lewej stronie inicjalizowane są odpowiednio wartościami 0 i 1. Po wykonaniu pierwszej pętli wartość z prawej strony przypisana jest do górnego rejestru z lewej strony. Ten zaś przechodzi na miejsce dolnego rejestru.

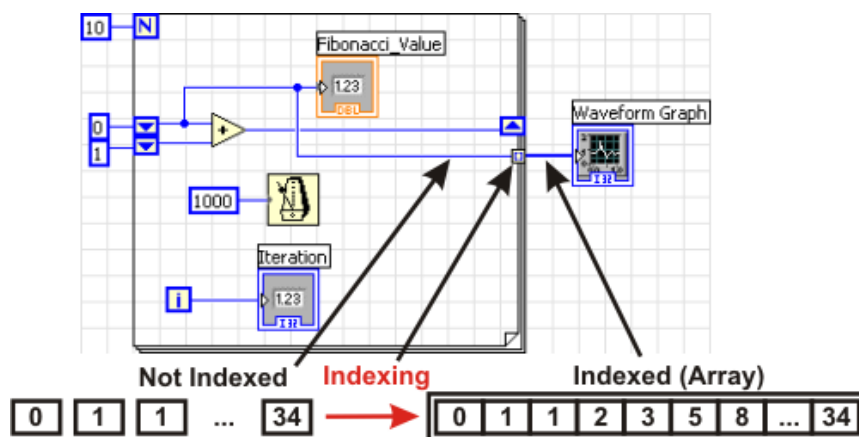
Grupowanie danych z wyjścia

Wynik działania pętli - może być zgrupowany (zindeksowany) do postaci tablicy (Array) lub pozostawiony bez grupowania.

Indeksowanie wyników sprowadza serię pojedynczych wartości do tablicy (Array). Array jest przekazywany dalej do programu - dopiero po pełnym zakończeniu pętli (po ostatniej iteracji). Odwrotnie jest w przypadku braku indeksowania. Poszczególne próbki nie są grupowane. Jedynie wynik pochodzący z ostatniej iteracji przekazywany jest dalej do programu.

Dwa odmienne poniższe przykłady obrazują różnicę.

Przykład 1 - z grupowaniem

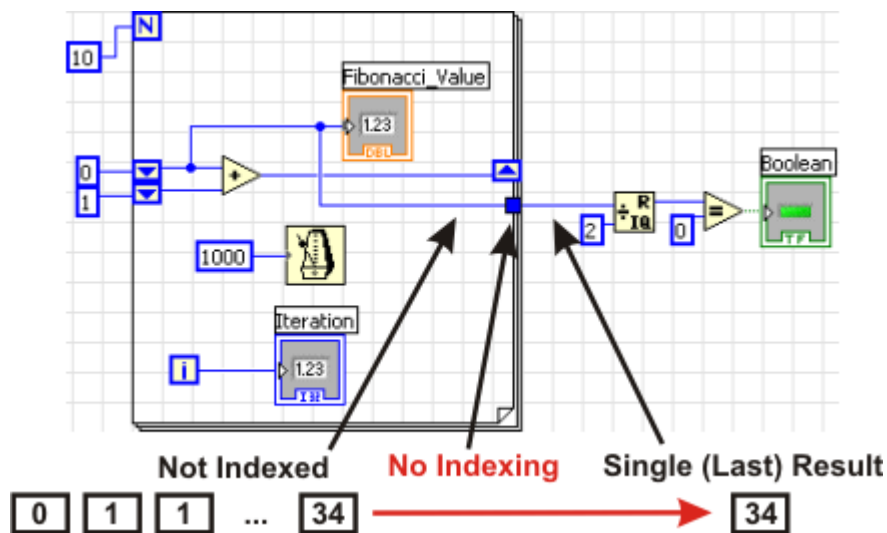


Indeksowanie wyników (loop5group1.vi)

Wykres - Waveform Graph możemy odnaleźć tworząc Panel Frontowy.  
Znajduje się on na palecie funkcji Graph Inds > Graph.

W powyższym przykładzie - po wykonaniu całej pętli - wynik zostanie przekazany do wykresu (Waveform Graph) - w zgrupowanej postaci. Argumentem wykresu jest Array (jednowymiarowy).  
W kolejnych częściach kursu o wielowymiarowych Arrayach. Wyprzedzając nieco w przyszłość - możemy powiedzieć, że wielowymiarowe szeregi (Array) można otrzymać przez wielokrotne indeksowanie wyników; kilka zagnieżdżonych pętli)

Przykład 2 - bez grupowania



Bez indeksowania wyników (loop5nogroup1.vi)

W powyższym przykładzie jedynie ostatni wynik jest wyrzucany na zewnątrz pętli. Tam też odbywa się testowanie czy wartość jest podzielna przez dwa (zapalona dioda) czy nie (zgaszona dioda).

Aby zabronić indeksowania - należy kliknąć prawym przyciskiem na kwadrat - symbolizujący indeksowanie. Wybieramy Disable Indexing.

Dzielenie z resztą możemy odnaleźć w palecie funkcji All Functions > Numeric > Quotient & Remainder.

Porównywanie możemy odnaleźć w palecie funkcji Arith/Compare > Equal To 0?

Po uruchomieniu przykładu zauważymy, że dioda zapali się po osiągnięciu wartości 34.

## LabVIEW - Arrays

Osoby mające doświadczenie w programowaniu w językach takich jak C czy Pascal - zdają sobie sprawę z faktu iż programowanie bez wykorzystania tablic czy struktur jest praktycznie niemożliwe. Analogicznie tworzenie bardziej skomplikowanych VI bez wykorzystania podobnych mechanizmów było by niezwykle trudne.

Na szczęście z pomocą przychodzą nam tablice (Arrays) i klastry (Clusters).  
Porównując należało by porównać Array do tablicy w C czy też w Pascalu.

Wówczas Cluster stałby się rekordem w Pascalu, zaś strukturą w języku C.

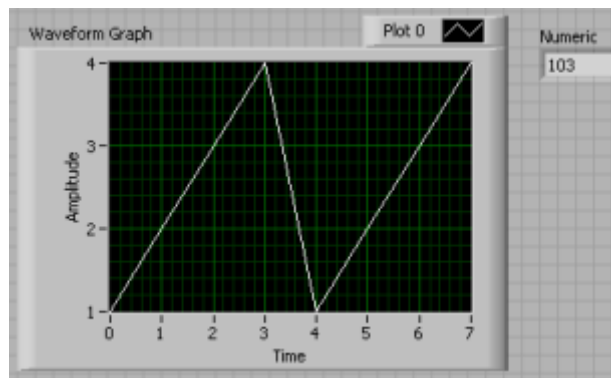
Tłumacząc to:

Array - tablica grupuje elementy jednego typu (np. tekst, wartość stałą, itp.); tablica może być wielowymiarowa

Cluster - kłaster grupujący elementy różnych typów (tekst, wartości stałe - w jednym klastrze jednocześnie)

Przykład

Stwórzmy poniższy panel frontowy.



Panel frontowy array1.vi

Diagram blokowy odzwierciedlony jest na poniższym rysunku.

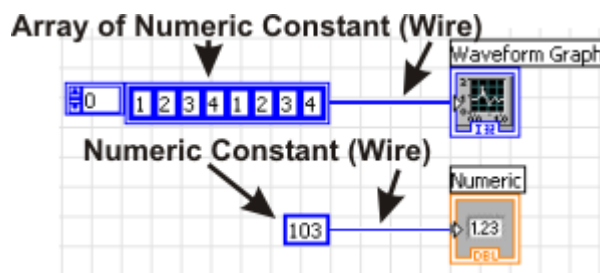


Diagram blokowy array1.vi, obrazujący jednocześnie różnice grubości przewodów

W celu stworzenia diagramu - wybieramy stałą tablicę: All Functions > Array > Array Constant. Po umiejscowieniu ikony w diagramie blokowym - będzie ona pusta. Oznaczać to będzie, że nie ma danych oraz że nie ma przypisanego typu danych. Aby dokończyć dzieło - należy wybrać stałą (tak jak wykonywano w poprzednich projektach (z palety funkcji): Arith/Compare > Numeric > Num Const. Stałą przeciągamy do środka ikony tablicy. Następnie powiększamy (rozciągając bok) rozmiar tablicy dożądanego.

Graf (Waveform Graph) można odnaleźć tworząc Panel Frontowy. Znajduje się on na palecie kontrolek (Controls Panel) Graph Inds > Graph.

Przewód (Wire) zawierające tablicę zgrupowanych danych jest koloru takiego samego jak dane

które grupuje. Różnica leży w grubości połączenia - Array grupujący typ danych A - jest grubszy. Grubość połączenia wzrasta ze wzrostem ilości wymiarów tablicy.

Przewód (Wire) symbolizujący połączone klastry - jest koloru różowego lub brązowego (w zależności od składników klastru).

## **LabVIEW - Clusters**

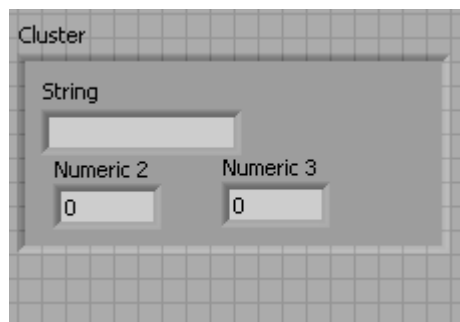
Klaster jest elementem podobnym do struktury w języku C, czy rekordu w Pascalu. Pozwala na zgrupowanie kilku różnych typów danych. Przykładowy klaster może zawierać liczby zmiennoprzecinkowe (double), łańcuchy znakowe (string), liczby całkowite (integer), oraz inne dane jednocześnie.

Istotna jest kolejność występowania elementów w klastrze. Jeżeli kolejność występowania elementów nie jest zachowana pomiędzy funkcjami - wówczas nie możemy połączyć funkcji. Zupełnie podstawowe funkcje realizowane przez klaster przedstawione są w poniższym przykładzie.

Przykład

Na panelu frontowym wybieramy: All Controls > Array & Cluster > Cluster. Umieszczamy na panelu. W pole nowo dodanego klastra dodajemy wskaźnik tekstu: (String Indicator) Text Inds > String Ind. Umieszczamy również wewnątrz klastra - wskaźniki wartości (również w pole klastra). Num Inds > Num Ind

Powinniśmy otrzymać następujący panel frontowy:

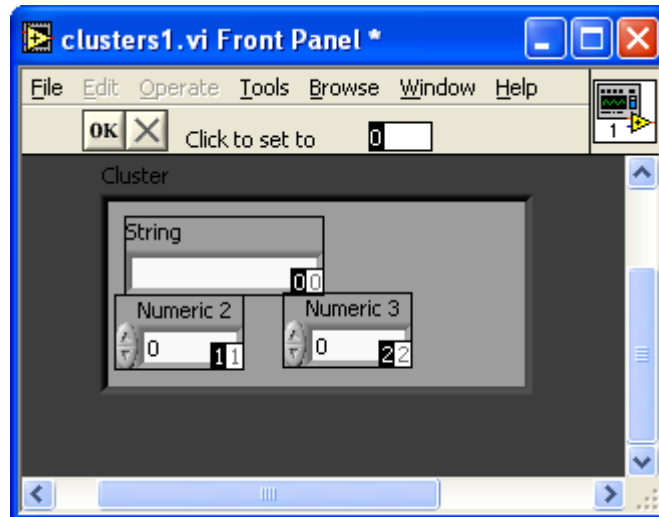


Panel frontowy clusters1.vi

W celu ustalenia kolejności w klastrze:

- kliknij prawym przyciskiem na krawędź klastra
- wybierz Reorder Controls in Cluster
- uzupełnij pole Click To Set wartością 0.
- kliknij na pole które ma być pierwsze (Numeric 2)
- kliknij na pole które ma być drugie (Numeric 3)
- kliknij na pole które ma być trzecie (String)
- zaakceptuj - klikając napis OK

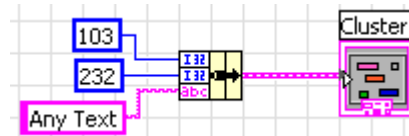




Panel frontowy clusters1.vi - ustalanie kolejności elementów w klastrze

W celu ustalenia klastra na panelu frontowym - jako wskaźnika  
 - kliknij prawym przyciskiem na krawędź klastra i wybierz Change to Indicator.

Diagram blokowy powinien wyglądać jak przedstawiono poniżej.



Panel frontowy array1.vi

Na diagramie blokowym z palety funkcji wybieramy All Functions > Clusters > Bundle. Element ten grupuje wchodzące z lewej strony dane i tworzy z niego klaster. Klaster ten zostanie przesłany do wskaźnika - zgodnie z przedstawionym diagramem.

Należy zwrócić szczególną uwagę na kolejność podłączania elementów po lewej stronie. Pierwsze dwa elementy to Numeric Constant. Trzeci z nich to String Constant (All Functions > String > String Constant)

## LabVIEW - Cases

Kolejne opisywane poniżej elementy języka - pozwalają nam na lepsze kontrolowanie przebiegu programu. Struktura case pozwala wybrać działanie w zależności od wartości.

Poniższy przykład zilustruje nam zasadę działania pętli case.

Warunek case będzie określony dla wartości 0, 1, 2. W każdym z tych trzech przypadków zostanie wypisany inny komunikat.

Diagram blokowy będzie miał następującą postać:

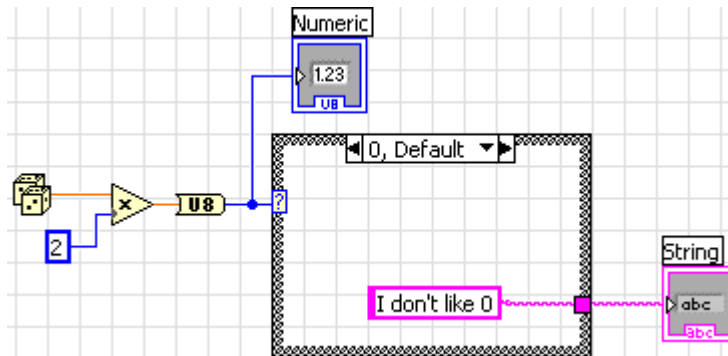
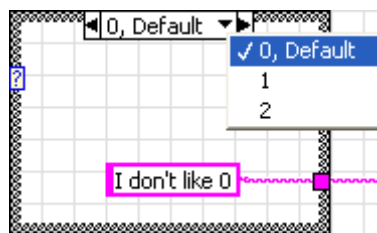


Diagram blokowy cases1.vi - dla wartości 0.

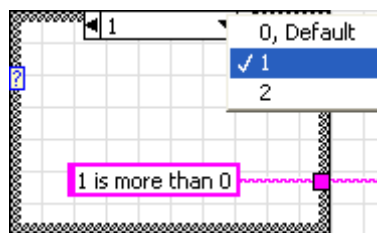
W celu wybrania struktury case należy z palety funkcji wybrać Exec Ctrl > Case Structure.

Case działa w oparciu o zasadę wyboru algorytmu. Wybór ten zależy od wartości sterującej strukturą case. W rozwinięciu warunku case - otrzymamy na wyjściu struktury poniższe łańcuchy.

W przypadku wartości 0, oraz w przypadku gdy uzyskamy wartość różną od (0,1,2) - zwrócony zostanie łańcuch (string) "I don't like 0"



W przypadku wartości 1, zwrócony zostanie łańcuch "1 is more than 0"



W przypadku wartości 2, zwrócony zostanie łańcuch "2 is very OK!"



## Przebieg ćwiczenia

1. Uruchomić program Labview, utworzyć pusty VI, zapoznać się z elementami dostępnymi w bibliotekach.
2. Utworzyć i uruchomić wszystkie przykłady znajdujące się w instrukcji.
3. Utworzyć program generujący tablicę liczb losowych z wykorzystaniem pętli while.
4. Utworzyć program generujący dwuwymiarową tablicę liczb losowych z wykorzystaniem pętli for.
5. Utworzyć program obliczający silnię dla zadanej wartości.
6. Utworzyć program pokazujący wykorzystanie różnych funkcji arytmetycznych i logicznych.
7. Utworzyć program obliczający wartość następującej funkcji:

$$f_2(x) = \begin{cases} x \leq -5; & 2x + 3\frac{1}{3} \\ -5 < x < 7; & \sqrt{|x|} + e^{\frac{x}{2}+1} \\ x \geq 7; & \sqrt[3]{\sin(2x) + x^2} \end{cases}$$

8. Utworzyć program prostego symulatora procesu, wartości mogą być zadawane ręcznie, pulpit ma zawierać przyciski START i STOP, wartości zmiennych procesu należy wyświetlać za pomocą różnych kontrolki graficznych, określać przekroczenia stanów maksymalnego i minimalnego i sygnalizować je na pulpicie.