Wstęp do graficznego programowania w środowisku **LabVIEW**

Miejska Biblioteka Publiczna w Piekarach Śląskich we współpracy ze Studenckim Kołem Naukowym Robotyki „Encoder” działającym przy Politechnice Śląskiej w Gliwicach

Niniejsza instrukcja ma formę skróconą i jest materiałem pomocniczym dla uczestników warsztatów.

Spis treści

1. Opis środowiska LabVIEW

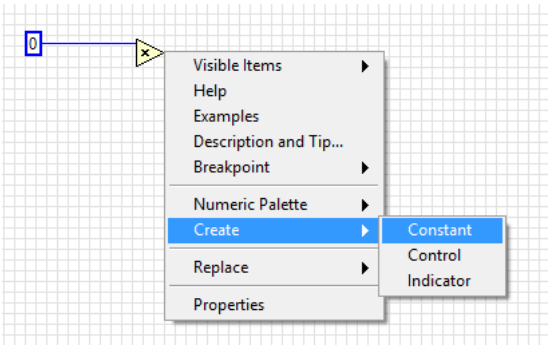
LabVIEW jest środowiskiem programowania graficznego stworzonym przez firmę National instruments. Ideą tego środowiska jest zmiana podejścia do tworzenia programów: zamiast pisać, rysuje się programy. Takie podejście pozwala sobie wizualizować program w sposób intuicyjny dla człowieka. Panuje zasada „przepływu danych", która powoduje, że kolejne funkcje nie zostaną uruchomione, jeżeli każdy z dołączonych do nich kabli nie będzie miał ustawionej jakiejś wartości. Dopiero, gdy wszystkie sygnały są doprowadzone, funkcja może być wykonana.

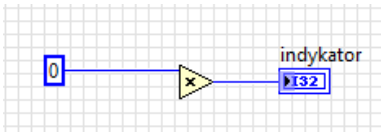
Aplikacja napisana w LabVIEW składa się z kodu programu (stworzonego w oknie „Block Diagram") oraz panelu czołowego, który umożliwia komunikację z użytkownikiem (stworzonego w oknie „Front Panel").

Funkcje i elementy interfejsu są dostępne w zasobniku, który ukazuje się po kliknięciu Prawym Przyciskiem Myszy (PPM). Panel czołowy i okno programu posiadają różne zasobniki.

Aby wykorzystać jakąś funkcję/element należy sprawdzić, ile terminali posiada (miejsc, w które można coś podpiąć). Przykładowo, operator mnożenia posiada dwa wejściowe terminale i jeden wyjściowy  . Aby podłączyć kabel należy kliknąć w kropkę (kabel przyczepi się do terminala) i przypiąć kabel do terminala innego obiektu, również przez kliknięcie kropki. Dopóki kabel nie jest przypięty, jest przerywany i jest koloru czarnego.

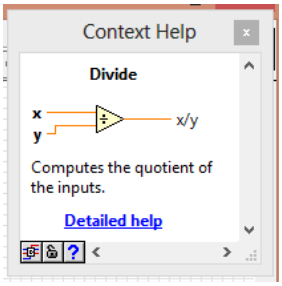
Aby przyspieszyć pracę, można skorzystać z opcji „Create” i wybrać z listy Constant/Control/Indicator.





Po stworzeniu programu, należy go uruchomić. W tym celu należy kliknąć białą strzałkę  w lewym górnym rogu programu. Kliknięcie  zapętli działanie programu – program będzie się wykonywał tak długo, aż użytkownik nie przerwie jego działania przez ponowne wciśnięcie tego przycisku. Wciśnięcie  natychmiastowo przerywa program w sytuacjach awaryjnych. Jeżeli strzałka jest przerwana  , oznacza to, że program zawiera błędy i nie może zostać uruchomiony.

Przydatną funkcją jest okienko pomocy, które włączamy przez wciśnięcie skrótu Ctrl+H. Okienko to pokazuje podstawowy opis każdej funkcji i elementu, na który najedzie się myszką.



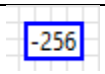
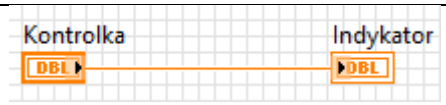
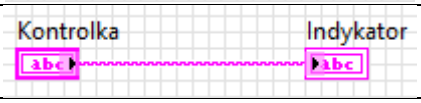
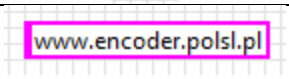
2. Podstawowe narzędzia programisty

2.1. Stałe, zmienne i podstawowe typy danych

LabVIEW jest językiem wysokiego poziomu, w którym zmienne wyrażone symbolem zostały zastąpione kablami przechodzącymi przez obszar programu. Zawartość komórki pamięci (zmiennej) jest przechowywana „w kablu" i można ją sprawdzić tak, jakbyśmy mierzyli napięcie elektryczne w przewodach elektrycznych.

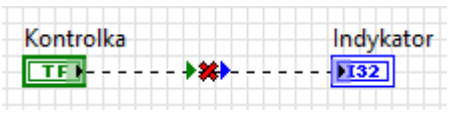
Stałe to takie zmienne, których wartość nie może być modyfikowana podczas działania programu.

Podstawowe typy danych to:

Typ	Wygląd kabla	Stała
Integer – dodatnie i ujemne liczby całkowite o różnym maksymalnym rozmiarze		
Unsigned Integer – tylko nieujemne liczby całkowite o maksymalnym rozmiarze dwukrotnie większym niż ich odpowiedniki Integer		
Single precision, **Double precision**, **Extended precision** – liczby zmiennoprzecinkowe (zapis ułamków) o różnym rozmiarze		
Boolean – wartość logiczna prawda/fałsz		
String – ciąg znaków tekstowych o dowolnej długości		

Jeżeli element na wejściu kabla wystawia dane innego typu, niż domyślny typ danych elementu wyjściowego, może nastąpić konwersja typu wejściowego na typ wyjściowy, co jest sygnalizowane czerwoną kropką:

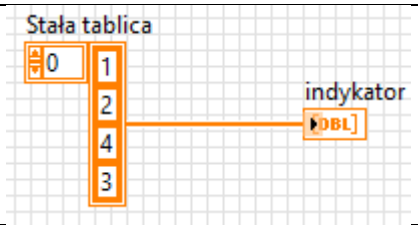
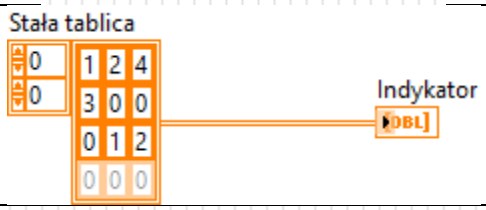
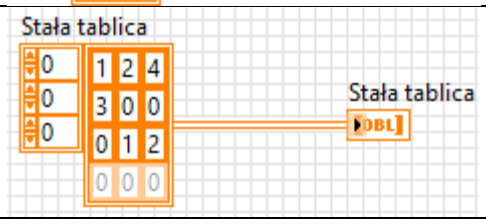
Konwersja typów następuje tylko w przypadku, gdy dane są ze sobą kompatybilne (typy danych nie są sprzeczne ze sobą). Przy konwersji może wystąpić utrata części informacji, np. konwersja z typu zmiennoprzecinkowego DBL (Double Precision) na Integer spowoduje utratę cyfr po przecinku. W przypadku, kiedy typy danych rażąco odbiegają od siebie, ujrzymy uszkodzony kabel:



2.2. Tablice

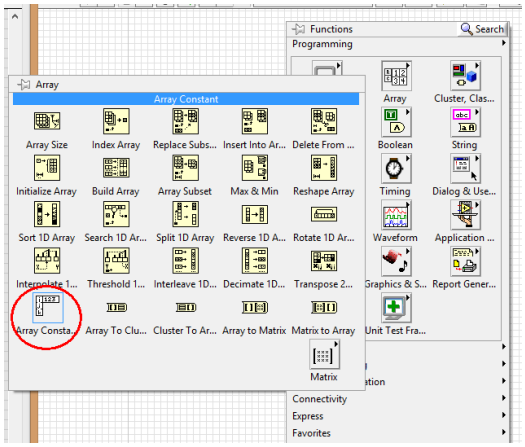
LabVIEW oferuje możliwość grupowania danych w złożone typy danych. Tak jak większość języków programowania, możliwe jest tworzenie tablic. Tablica to ciąg komórek pamięci, w których przechowujemy wartości. Tablice mogą być jednowymiarowe i dwuwymiarowe – takie łatwo sobie wyobrazić, jak również trzy i więcej wymiarowe.

Jeżeli przez kabel przekazujemy tablicę, to grubość kabla informuje o liczbie wymiarów tablicy:

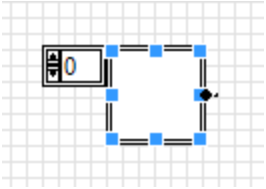
Tablica jednowymiarowa	
Tablica dwuwymiarowa	
Tablica trójwymiarowa	

Jednym z prostych sposobów na utworzenie tablicy jest:

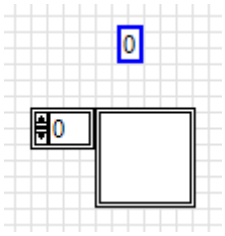
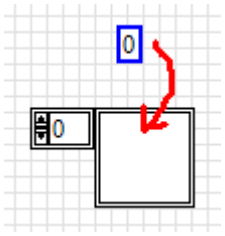
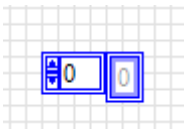
1) Wybrać z zasobnika (Prawy Przycisk Myszy) paletę Array w kategorii Programming
2) Z rozwiniętego okna wybrać Array Constant



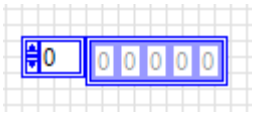
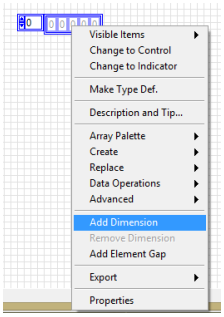
3) Położyć Stałą Tablicę w obszarze programu



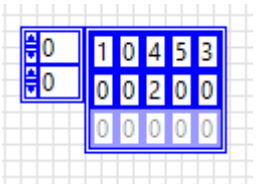
4) W dowolny sposób utworzyć stałą typu danych, który ma przechowywać tablica i włożyć stałą do tablicy. Stała Tablica zmieni nieco swój wygląd.

5) Dostosuj tablicę do swoich potrzeb poprzez rozciągnięcie jej lub dodanie kolejnego wymiaru (kliknąć prawym przyciskiem myszy na ramkę tablicy i wybrać opcję „Add Dimension”)

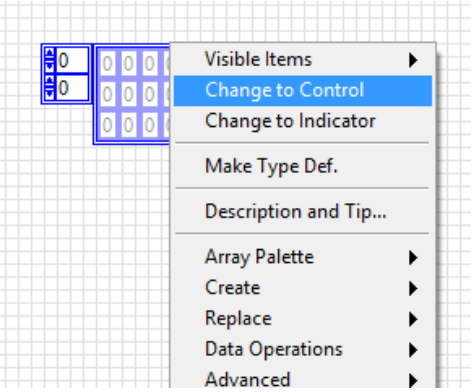
6) Uzupełnij tablicę danymi



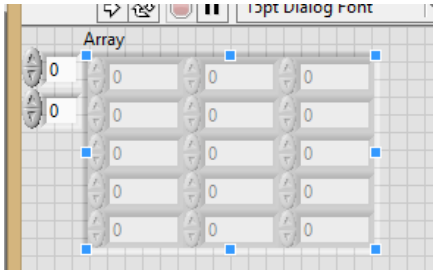
Jeżeli cała tablica lub któryś wiersz/kolumna nie zostanie zainicjaliowana wartościami, komórki tablicy są blade.

Aby umożliwić wprowadzanie danych do tablicy lub wyświetlanie danych, należy:

1) Kliknąć prawym przyciskiem myszy na ramkę tablicy i wybrać „Change to Control” albo „Change to Indicator”.



2) Przejść do okna z panelem czołowym programu i kliknąć ramkę tablicy. Rozciągnąć tablicę.



Wyświetlacze obok tablicy pozwalają ją przesuwać, aby dostać się do elementów, które wybiegają poza widoczny obszar.

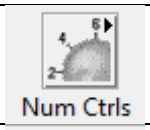
W grupie Array w zasobniku znajduje się dużo funkcji, które można wykonać na tablicach, między innymi sprawdzenie wartości wskazanego elementu (Index Array ), sprawdzenie rozmiarów tablicy (Array Size ), wyodrębnienie podzbioru tablicy (Array Subset ) i wiele innych.

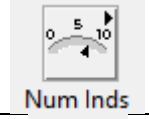
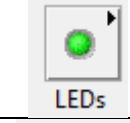
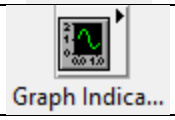
2.3. Kontrolki i indykatory

Bardzo często w programie konieczne jest wprowadzenie interakcji z użytkownikiem. Elementami komunikującymi się z nim są kontrolki i indykatory. Kontrolki służą do wprowadzania danych, natomiast indykatory do wyświetlania wyników.

Kontrolki wstawia się w oknie frontowym programu.

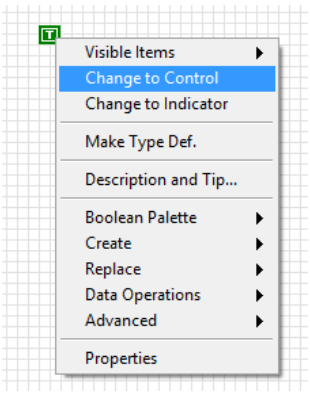
Podstawowe kategorie kontrolek i indykatorów, znajdujące się w kategorii Express, to:

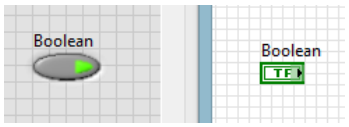
Numeryczne kontrolki (liczbowe)	
Przełączniki i przyciski (binarne kontrolki)	

Tekstowe kontrolki	
Numeryczne indykatory (liczbowe)	
Diody LED (binarne indykatory)	
Tekstowe indykatory	
Indykatory grafów (wykresy)	

Każdy typ danych, który występuje w LabVIEW, może mieć własną kontrolkę, indykator i stałą.

Kontrolkę lub indykator można utworzyć w alternatywny sposób, przez zmianę stałej w żądany typ:

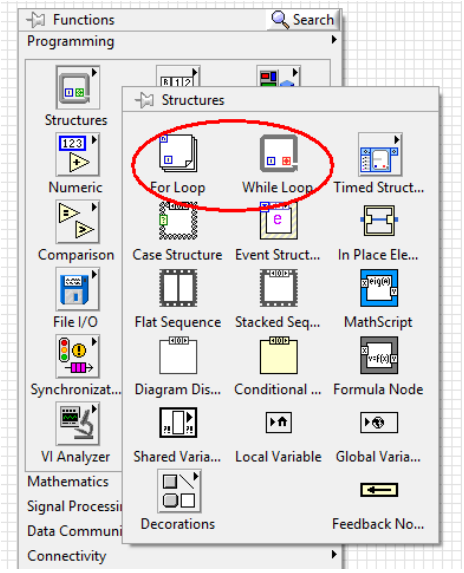




2.4. Pętle

Języki programowania umożliwiają cykliczne powtarzanie fragmentów programu dzięki zastosowaniu pętli.

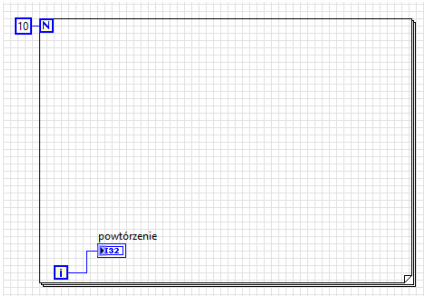
W LabVIEW pętle znajdziemy w zasobniku, w grupie Structures (kategoria Programming):



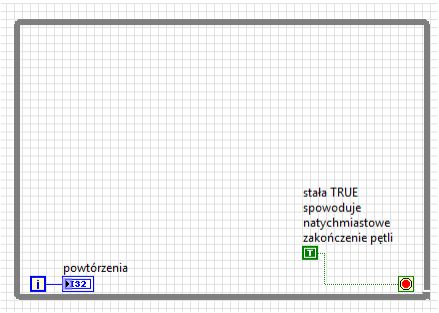
Pierwsza z zaznaczonych pętli to pętla „for”. Pętla ta umożliwia wykonanie fragmentu programu ze z góry znaną liczbą powtórzeń. W lewym górnym rogu podłączamy do terminala „N” liczbę powtórzeń, w lewym dolnym rogu mamy terminal „i”, który informuje, które powtórzenie (iteracja) jest aktualnie wykonywane.

UWAGA! Terminal „i” rozpoczyna numerację powtórzeń od **0** – jeśli żądamy np. 10 powtórzeń, terminal „i” będzie liczył od 0 do 9.

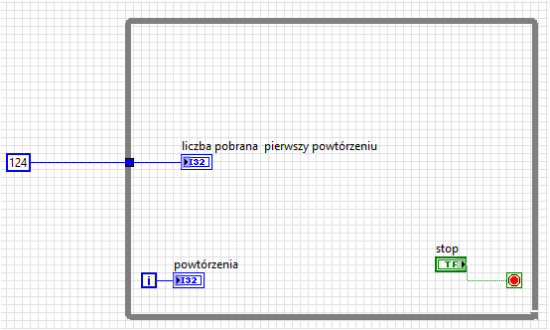
Pętla „for” wygląda następująco:



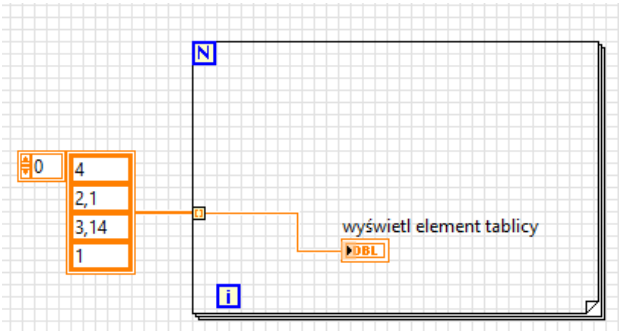
Druga z dostępnych pętli, pętla „while”, wykonuje powtórzenia programu, dopóki nie zostanie spełniony określony warunek zakończenia. Terminal „i” ma takie samo działanie jak w pętli „for”. Warunek zakończenia podłączamy do terminala z czerwoną kropką.



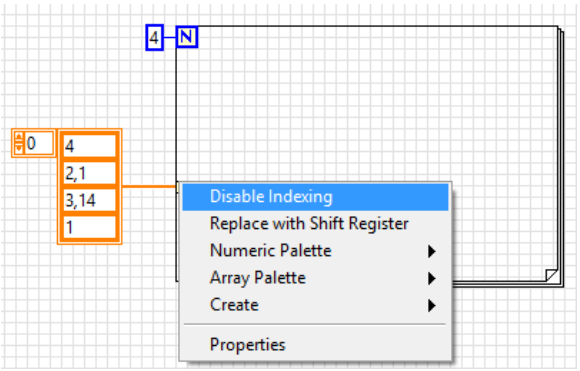
Dane do pętli można przekazywać przez kable. Zwykłe dane zostaną pobrane tylko przy wejściu do pętli i każda kolejna iteracja pętli będzie korzystała z tych samych danych.



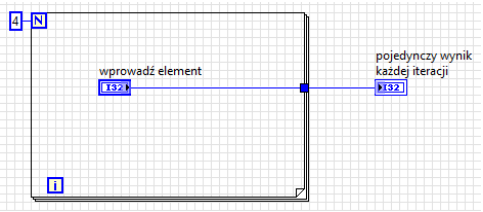
Jeżeli do pętli przekazujemy tablicę, można sprawić, by pętla wykonała tyle powtórzeń, ile jest elementów w tablicy. Wtedy każde powtórzenie pobierze kolejny jeden element z tablicy.

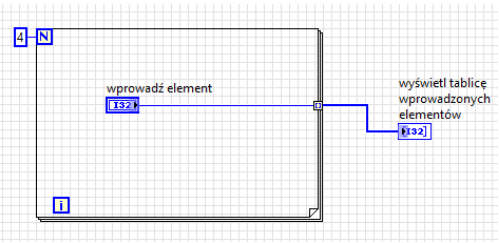


Jeżeli chcemy przekazać do środka tablicę jako całość, należy wyłączyć tę opcję poprzez kliknięcie PPM i wybranie „Disable indexing”. Konieczne jest jednak wtedy podanie, ile powtórzeń ma wykonać pętla:

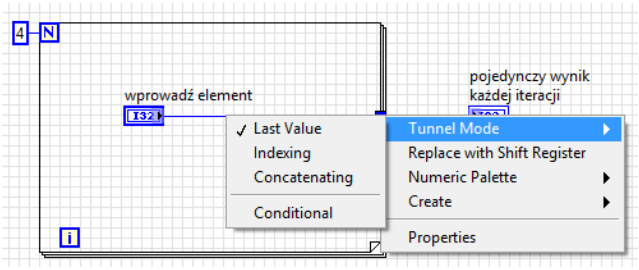


Analogicznie do wprowadzania danych działa wyciąganie danych z pętli. Można zmusić ją, by zwróciła tylko wynik ostatniego powtórzenia (po lewej), albo zwróciła tablicę elementów, które namnożyły się w wyniku jej powtórzeń (po prawej):



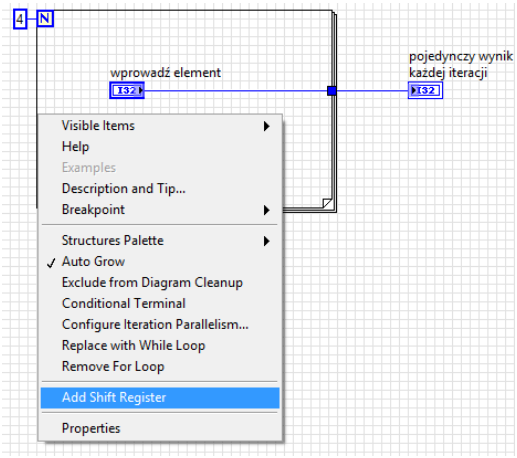


Pożądane zachowanie można ustawić po kliknięciu PPM na tunel wyjściowy i wybraniu jednej z opcji z grupy „Tunnel mode”:

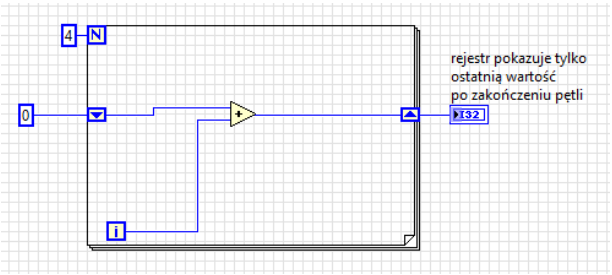


Ostatnią z zalet pętli jest Rejestr Przesuwny, czyli terminal umożliwiający zapamiętanie wartości z poprzedniej iteracji i przekazanie jej na początek kolejnej iteracji. Rejestr przesuwny dodaje się poprzez:

1) kliknięciem PPM na ramce pętli i wybranie opcji „Add Shift Register”.



2) Następnie należy podłączyć kabel, z którego dane mają przejść do kolejnej iteracji:

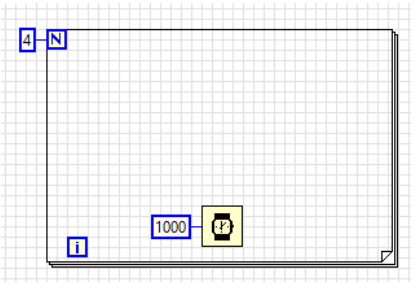


Na obrazku podłączono liczbę 0 do rejestru przesuwnego. Dzięki temu określamy, jaką wartość pokazuje rejestr w pierwszej iteracji.

2.5. Czasomierze

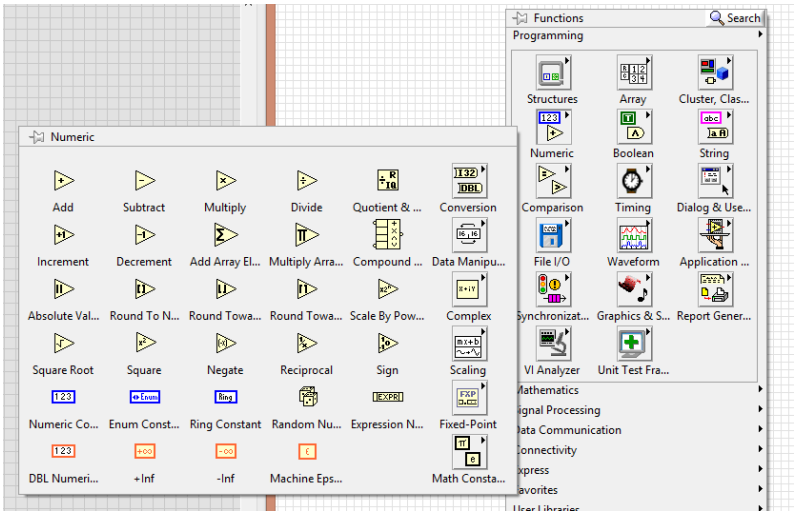
Z pętlami niewątpliwie wiąże się funkcja czasomierzy. Jeżeli w programie wykorzystamy pętlę, to LabVIEW postara się wykonywać ją najszybciej jak potrafi, skupiając na niej całą dostępną moc obliczeniową procesora. Aby rozdzielić moc obliczeniową na inne elementy programu, konieczne jest zastosowanie funkcji opóźniającej, która pozwala „odetchnąć” procesorowi.

Najpopularniejszym sposobem opóźnienia pętli jest wykorzystanie funkcji „Wait (ms)”, dostępnej w zasobniku w palecie Timing (kategoria Programming). Parametr wchodzący do niej to czas opóźnienia wyrażony w milisekundach. 1000 milisekund (ms) to 1 sekunda (s).



2.6. Operatory działań matematycznych

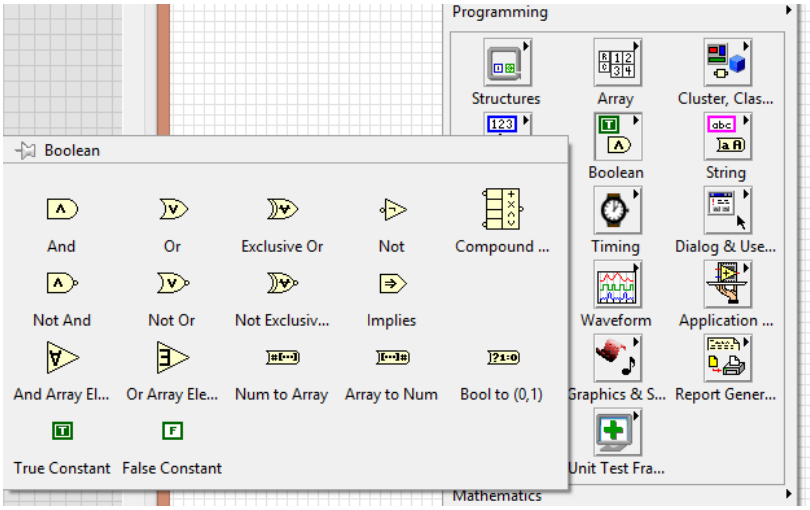
Paleta Numeric zasobnika dostarcza wielu podstawowych funkcji i operacji matematycznych.



Możliwe jest stosowanie podstawowych operatorów matematycznych, jednak z punktu widzenia programisty, szczególnie przydatne wydają się operatory inkrementacji (zwiększenia o jeden)  i dekrementacji (zmniejszenia o jeden)  , zaokrąglenia do najbliższej liczby całkowitej  oraz funkcja liczb losowych  . Co do tej ostatniej można powiedzieć, że zwraca ona liczbę z przedziału od 0 do 1, czyli liczbę zmiennoprzecinkową.

2.7. Operatory binarne

Analogicznie to operacji matematycznych, LabVIEW dostarcza palety działań binarnych. Paleta ta nosi w zasobniku nazwę Boolean:



Podstawowe operacje binarne to AND (logiczne „i”), OR (logiczne „lub”) i NOT (zaprzeczenie). Tablice działania tych operatorów znajdują się poniżej.

Wejście A \ Wejście B	**0**	**1**
0	*0*	*0*
1	*0*	*1*

Tabela dla operacji AND

Wejście A \ Wejście B	**0**	**1**
0	*0*	*1*
1	*1*	*1*

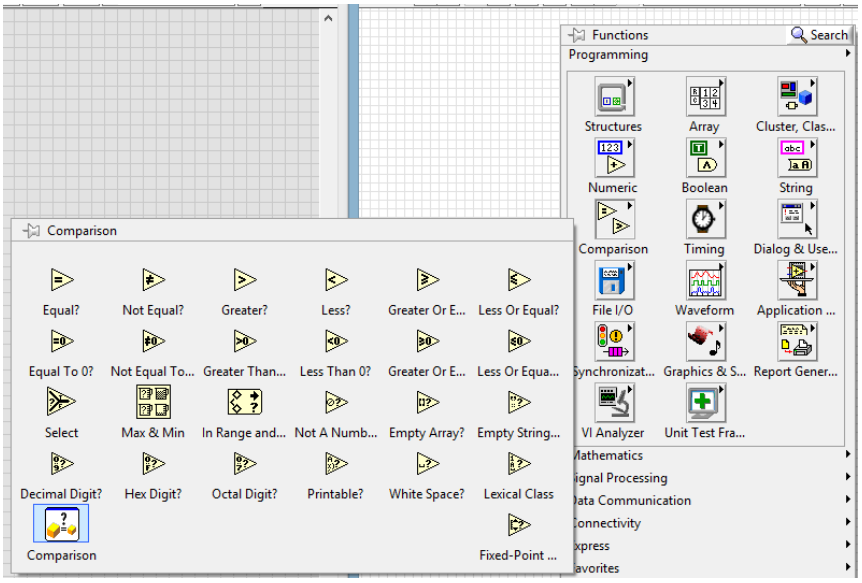
Tabela dla operacji OR

Wejście	**Wynik**
0	*1*
1	*0*

Tabela dla operacji NOT

2.8. Operatory porównania

Ważną paletą w LabVIEW jest paleta Comparison. Dzięki niej podejmiemy decyzje na podstawie wartości przepływających w programie.

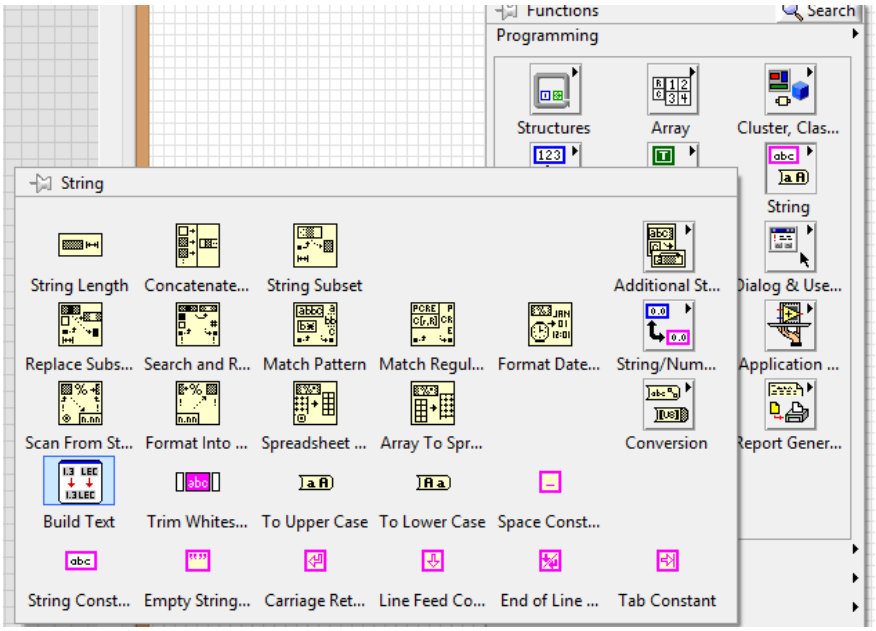


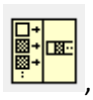
Wynik zwracany przez operatory porównania jest typu logicznego (binarnego) Boolean.

Najważniejsze z tego zbioru funkcji są: operator porównania  i funkcja wyboru  . Po zastosowaniu operatora porównania naturalnym następstwem jest podjęcie decyzji na podstawie wyniku zwróconego przez niego. Do funkcji wybory wpinamy dwie alternatywne wartości – pierwsza zostanie przekazana na wyjście, gdy sygnał sterujący przyjmie wartość „true”, druga, gdy sygnał sterujący przyjmie „false”.

2.9. Operacje na ciągach tekstowych

Warto wspomnieć, że LabVIEW daje niezwykle rozbudowane możliwości operowania na zmiennych tekstowych. Operacje te są intuicyjne, w przeciwieństwie do niektórych starszych tekstowych języków programowania.

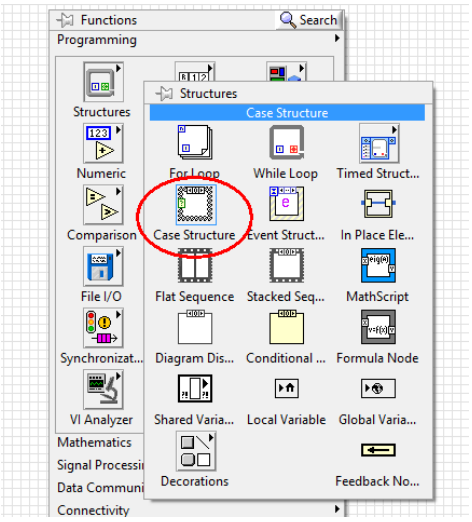


W każdej aplikacji, umożliwiającej wyświetlanie komunikatów, przydadzą się takie funkcje jak konkatenacja (łączenie ciągów tekstowych)  , czy funkcja zwracająca długość ciągu tekstowego (liczbę znaków)  .

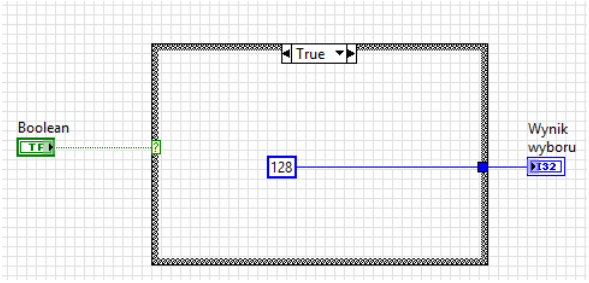
Przydatne mogą okazać się też funkcje wyszukiwania w ciągu tekstowym pewnych zdefiniowanych słów  .

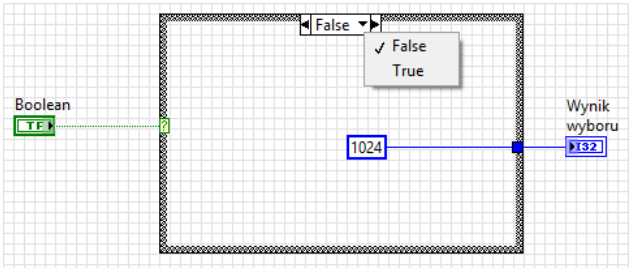
2.10. Instrukcje warunkowe

W ramach palety Comparison, w punkcie 3.8, opisano działanie funkcji wyboru  . Dzięki niej wybierzemy, która wartość ma zostać przekazana do dalszej obróbki. Jednak czasem konieczne jest zaprogramowanie zupełnie innego zachowania programu, w zależności od logicznego sygnału sterującego otrzymanego z operatora porównania. Do takich zastosowań stosuje się Strukturę Warunkową (Case Structure) dostępną na palecie Structures w zasobniku:

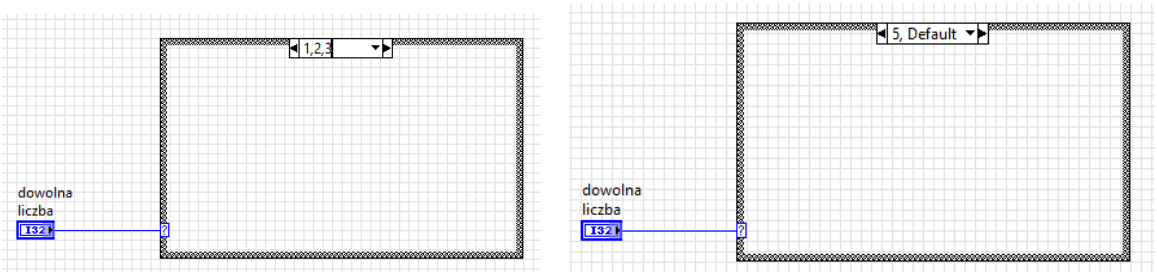


Decyzję o wyborze jednego z wariantów programu, zapisanych na kartach tej struktury, dokonuje się poprzez podanie sygnału sterującego na terminal ze znakiem zapytania „?”. Najprostszym sposobem na wykorzystanie tej struktury jest podpięcie sygnału o typie Boolean. Do alternatywnych kart dostaniemy się przez kliknięcie wskaźnika u góry ramki.



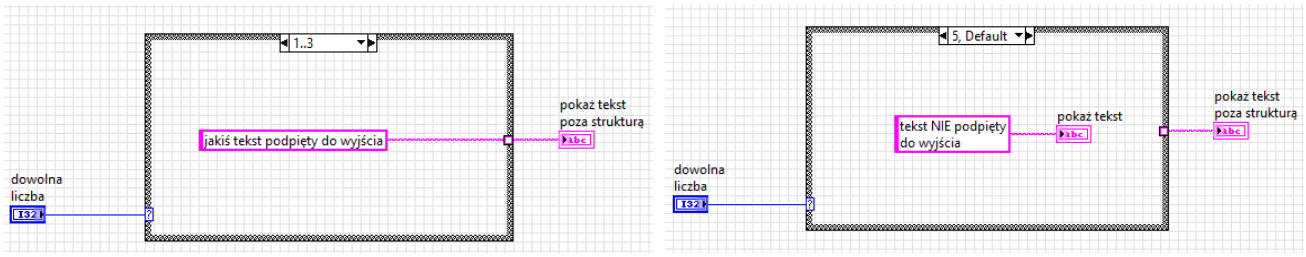


Niepodważalną zaletą jest, że struktura ta dostosowuje się do typu danych podpiętego do terminala wyboru „?”. Możliwe jest stworzenie kart, które mogą być zmieniane w zależności od wartości sygnału sterującego, który może być liczbą:



W powyższym przykładzie jedna karta będzie wybrana gdy kontrolka „dowolna liczba” przyjmie wartości 1, 2 albo 3, natomiast druga karta będzie wybrana gdy pojawi się liczba 5. Dodatkowo, słowo Default po przecinku oznacza, że jeżeli pojawi się liczba, która nie została jawnie podana, to program wybierze tę kartę jako domyślną.

Ostatnią uwagą, jaką można powiedzieć o strukturach warunkowych, jest to, że aby program poprawnie działał, konieczne jest podpięcie kabli do wyjść na wszystkich alternatywnych kartach. Program napisany w sposób pokazany na poniższym rysunku nie zadziała, bo nie będzie wiedział, co wyświetlić na indykatorze „pokaż tekst poza strukturą” w drugim przypadku (po prawej):

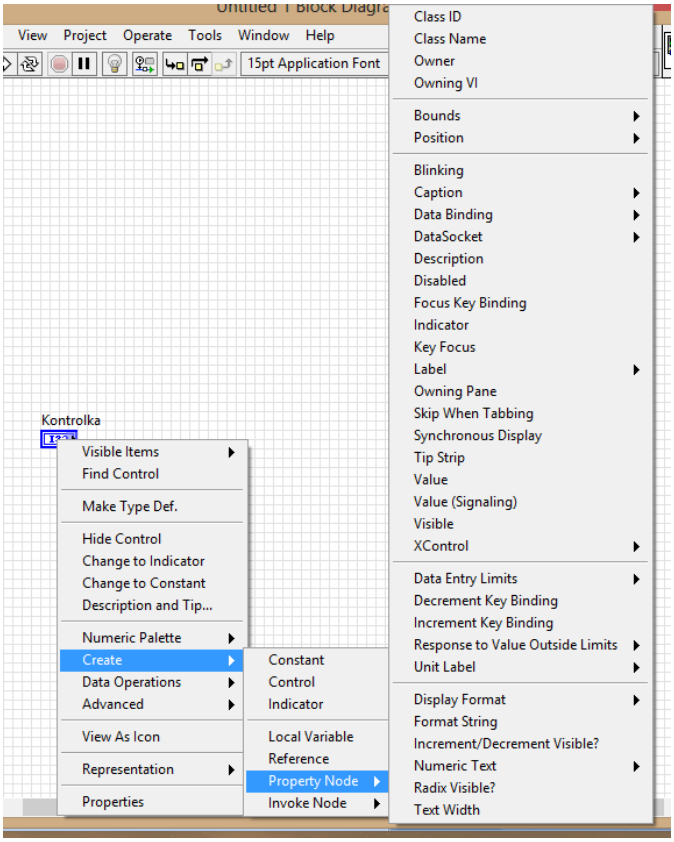


3. Zaawansowane narzędzia programisty

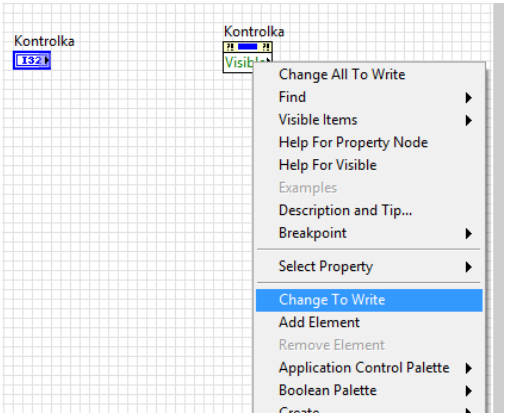
3.1. Własności kontrolek

Własności kontrolek, tzw. Property Nodes, to zestaw parametrów kontrolki i opis jej działania, który można edytować (programowo). Najprostszy sposób na zmianę zachowania kontrolki to:

1) Kliknąć obiekt PPM, z listy wybrać „Create”, następnie „Property Node” i z długiej listy wybrać własność, którą chcemy edytować.



2) Domyślnie Property Node są ustawione na „Read”. Jeżeli chcemy tylko sprawdzić stan kontrolki (czy wybrana własność jest włączona czy wyłączona), należy tak zostawić i wykorzystać zwracaną wartość w programie. Jeżeli jednak chcemy zmienić zachowanie kontrolki, należy kliknąć PPM i wybrać „Change to Write”.



3) Podpiąć kabel z nowym parametrem.

Na tym przykładzie ustawiamy programowo widoczność kontrolki na „false”, czyli użytkownik nie zobaczy jej na panelu czołowym programu.

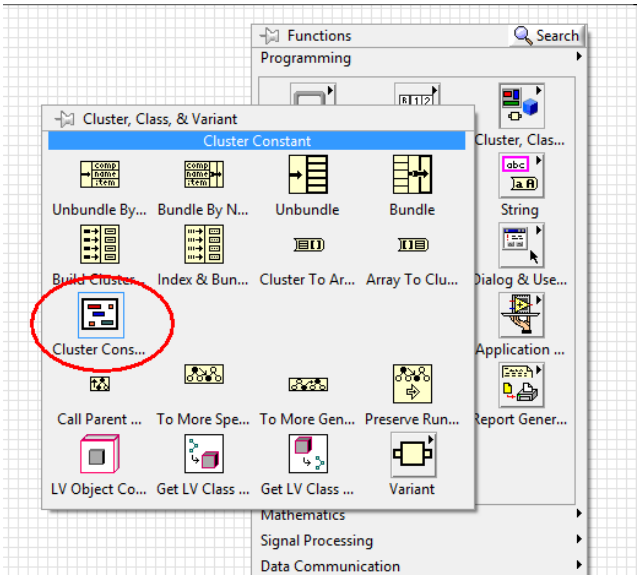
3.2. Klastry

Tablice są klasycznym sposobem przechowywania danych o takim samym typie. Jeżeli istnieje konieczność przechowania danych o różnych typach, które np. tworzą opis pewnego obiektu, konieczne jest zastosowanie KLASTRÓW.

Klastry to takie twory, które mogą przechowywać w swoich komórkach kilka wartości dowolnego typu.

Klaster można utworzyć:

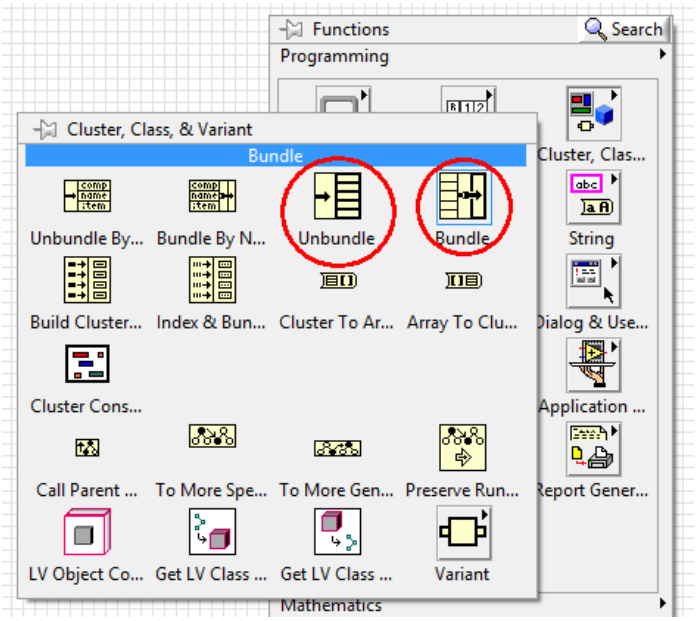
1) wybierając grupę Cluster, Class & Variant z kategorii Programming w zasobniku (PPM). Należy wybrać Cluster Constant i położyć go na obszarze programu.



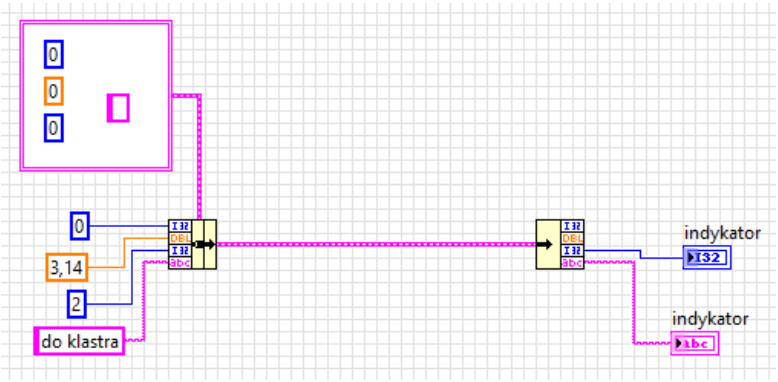
2) Podobnie jak w przypadku tablic, należy włożyć do klastra elementy o takim typie danych, jaki sobie życzymy. Różnica jest taka, że elementów może być kilka i mogą mieć różne typy danych. Jeżeli w klastrze znajdą się tylko typy liczbowe, kolor klastra będzie brązowy (po lewej). Jeżeli w klastrze znajdą się też inne elementy, kolor klastra zmieni się na różowy (po prawej).

Do poszczególnych elementów utworzonego klastra można wpisać informacje. Aby to zrobić, należy:

1) z palety Cluster… w zasobniku wybrać Bundle. Aby odczytać poszczególne elementy, należy z palety Cluster… w zasobniku wybrać Unbundle.



2) Połączyć je w sposób pokazany poniżej

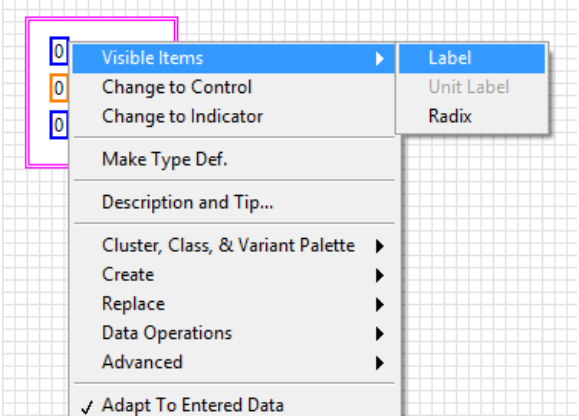


Nie jest konieczne odczytanie wszystkich pól klastra – można to zrobić wybiórczo.

Wpisywanie i odczytywanie danych przez Bundle i Unbundle odbywa się w kolejności, która była kolejnością wkładania typów danych do nowego klastra. Wygodnym sposobem jest nazwanie wewnętrznych elementów i wpisywanie/odczytywanie elementów o zdefiniowanej nazwie.

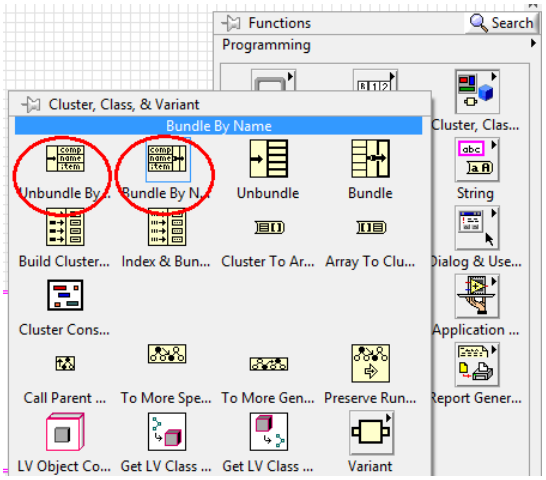
Aby nadać nazwy elementom klastra i korzystać z nich, należy:

1) kliknąć PPM na każdym elemencie i wybrać Visible Items, a następnie Label:

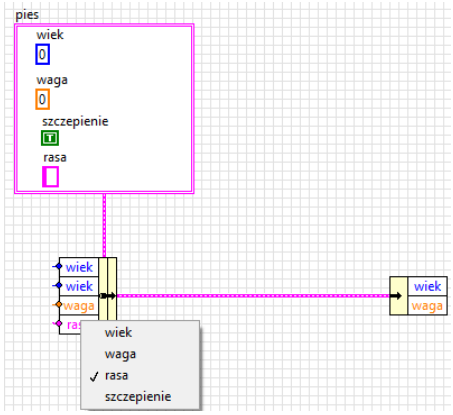


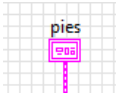
2) gdy kursor będzie migał, podać nazwę elementu. Zatwierdzić przez kliknięcie myszką gdzieś w obszarze programu.

3) W palecie Cluster… w zasobniku wybrać Bundle by Name i Unbundle by Name



4) Połączyć tak jak na obrazku, korzystając tylko z tych pól, na których nam zależy

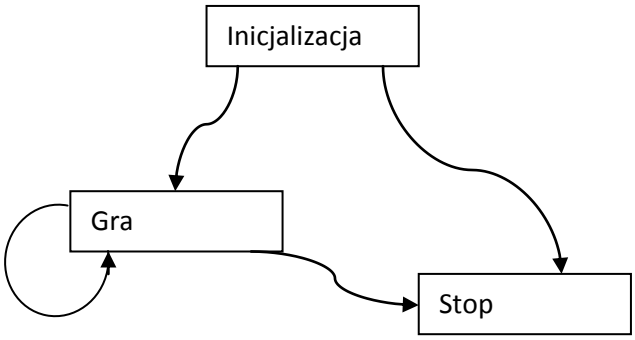


Aby klaster zajmował mniej miejsca, można dwukrotnie go kliknąć otrzymując zgrabną ikonę 

3.3. Wzorzec projektowy - Maszyna stanu

Programy o prostej funkcjonalności można zapisać na jednym ekranie z kablami przechodzącymi od lewej do prawej. Bardziej zaawansowane aplikacje wymagają wprowadzenia pewnych schematów programowania, dzięki którym zyskujemy możliwość rozbudowy programu w przyszłości oraz funkcjonalności, których próżno szukać w programach zapisanych w prosty sposób „od lewej do prawej”.

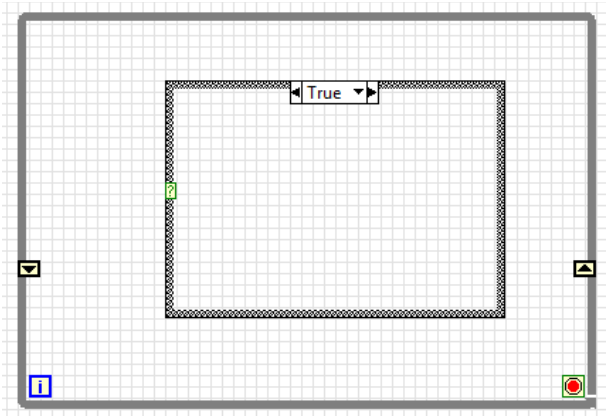
Wzorców projektowych jest wiele. Najprostszym, jednak bardzo często stosowanym, jest Maszyna Stanu. Maszyna stanu (State Machine) pozwala, w sposób programowy, zrealizować programy, które można zapisać w postaci grafu przejść:



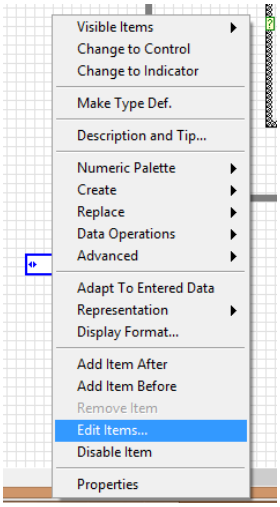
Cechą maszyny stanu jest „pamięć”, dzięki której kolejny stan wie, jaki był poprzedni stan wywołany w programie.

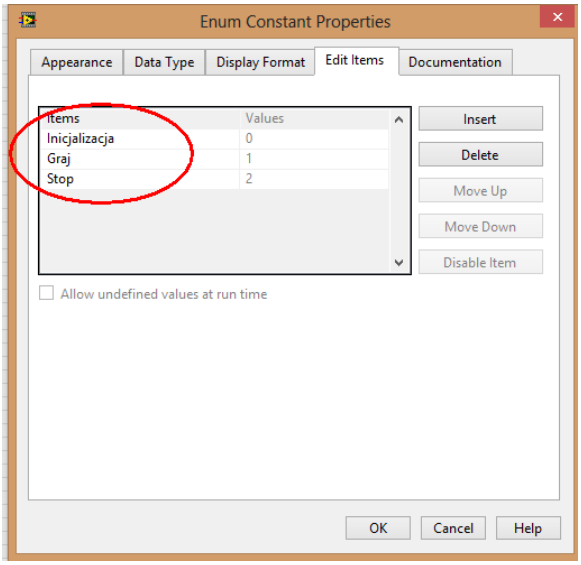
Maszynę stanu konstruuje się w następujący sposób:

1) Utworzyć pętlę while i włożyć do niej strukturę warunkową. Do pętli dodać rejestr przesuwny („pamięć" maszyny stanu).

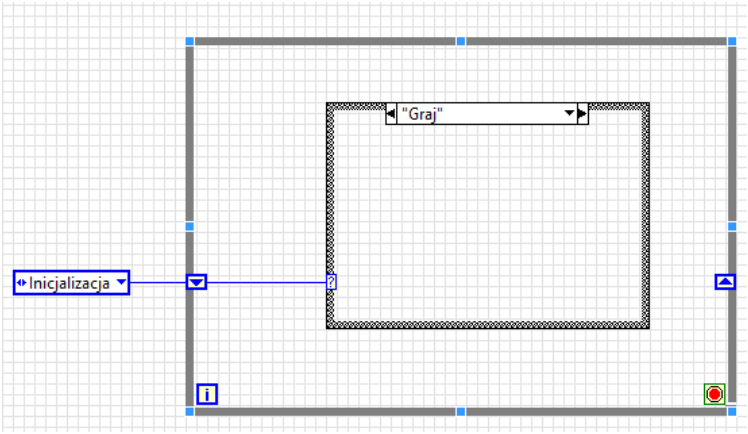


2) Utworzyć stałą typu Enum (można znaleźć ją w palecie Numeric w zasobniku) i dodać stany grafu przejść przez edytowanie jej elementów.

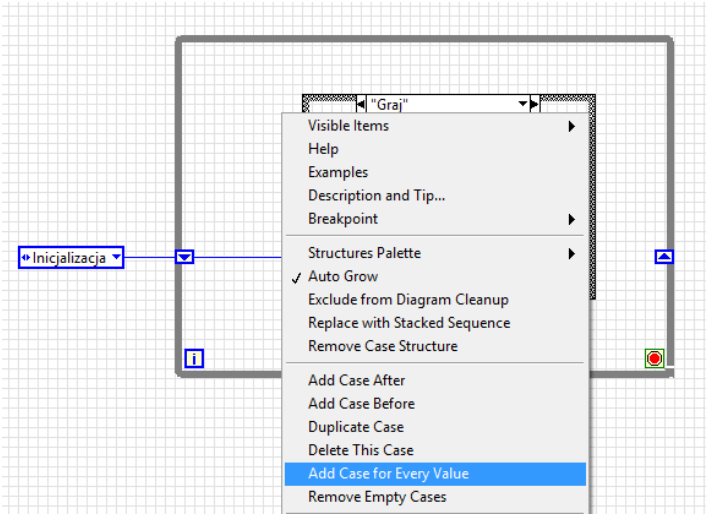




3) Podłączyć Enuma do rejestru przesuwnego i terminala wyboru „?”.

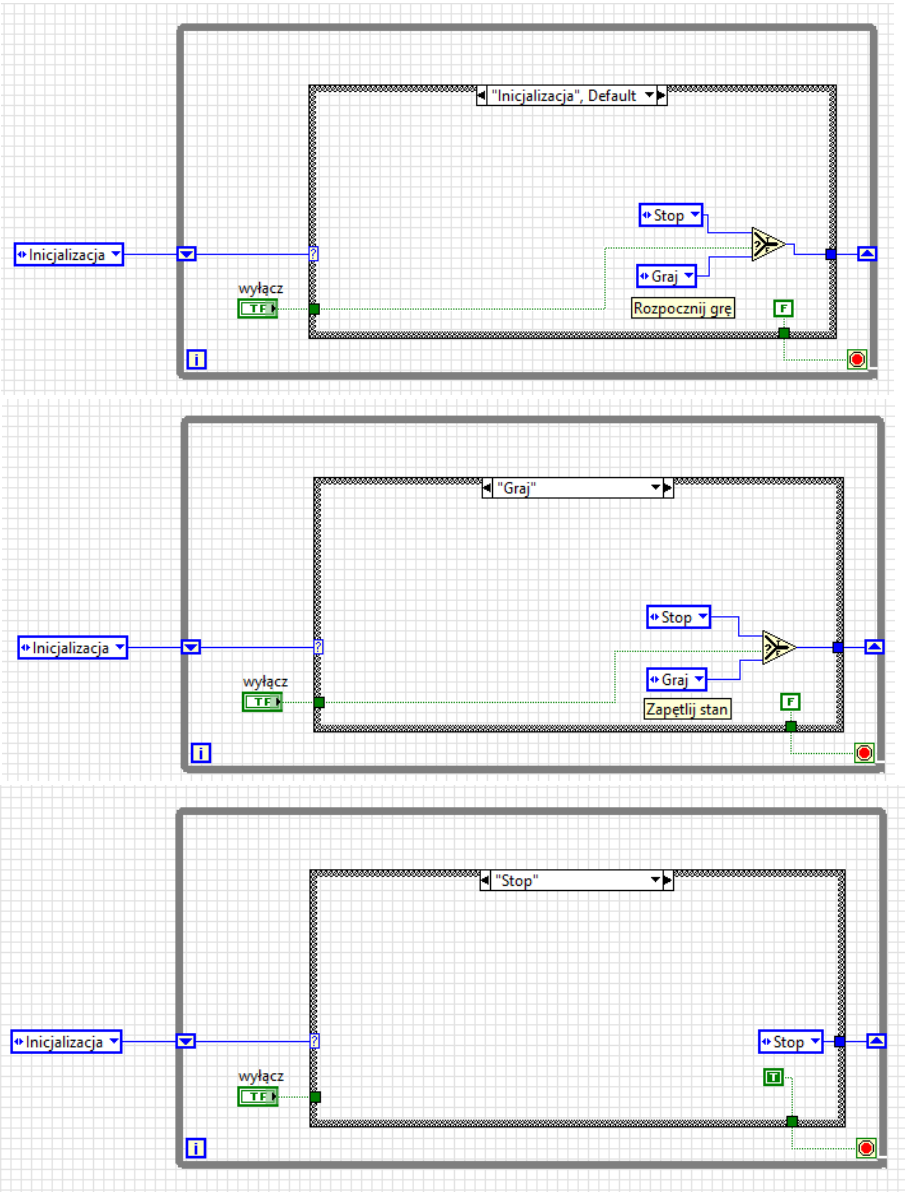


4) Kliknąć PPM na ramkę struktury warunkowej i wybrać „Add Case for Every Value”.

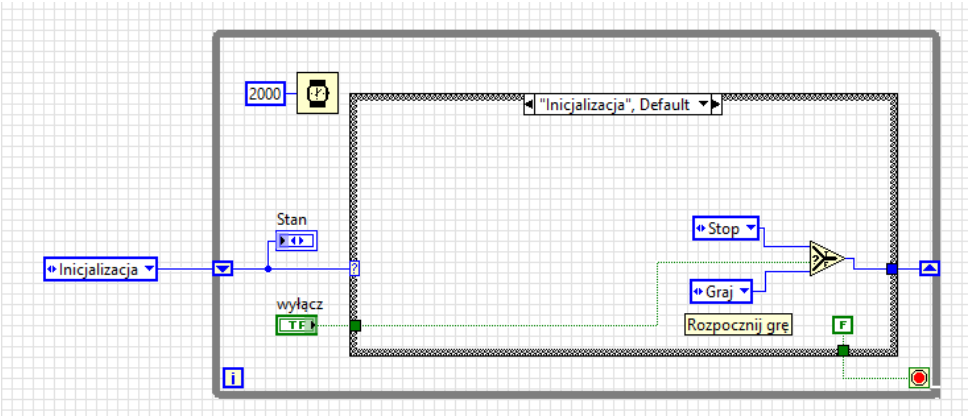


5) Na panelu czołowym dodaj przyciski sterujące maszyną stanu.

6) Podłącz przyciski w odpowiednich kartach struktury warunkowej. Skopiuj Enuma ze stanami grafu i przy pomocy funkcji wybierania  zdecyduj, który stan ma być wykonany jako kolejny w każdej karcie.



7) Skopiuj Enum i zmień go na Indykator. Podepnij go w pętli while do kabla przekazującego stan. Dzięki temu będzie możliwy podgląd, w jakim stanie znajduje się program. Dodaj opóźnienie.



Przetestuj maszynę stanu. Stan „Inicjalizacja” może służyć do przygotowania programu po uruchomieniu, stan „Stop” może służyć do „uprzątnięcia” (np. wyłączenie komunikacji radiowej lub zwolnienie zajmowanej pamięci) przed wyłączeniem programu.

4. Przydatne skróty klawiszowe

Skróty klawiszowe ułatwiają i przyspieszają pracę w LabVIEW. Najważniejsze są przedstawione w tabeli poniżej:

Ctrl + H	Otwiera przydatne okienko z opisem funkcji. Opis pojawia się, gdy najedziemy kursorem myszy na funkcję lub dowolny element w LabVIEW.
Ctrl + E	Przełącza między oknem panelu frontowego a oknem programu.
Ctrl + T	Ustawia okno panelu frontowego obok okna programu. Ułatwia pracę przy większych monitorach.
Ctrl + spacja	Włącza okienko szybkiego wyszukiwania. Możemy z niego skorzystać, jeżeli znamy nazwę poszukiwanej funkcji/elementu.
Ctrl + przeciągnięcie myszką	Jeżeli zrobiło się za ciasno w kodzie programu, skrót ten rozsuwa program, dzięki czemu zyskujemy więcej miejsca.
Wybrać element -> użyć strzałek	Dzięki strzałkom można precyzyjnie ustawić każdy element na panelu czołowym i w kodzie programu.
Wybrać element -> użyć Shift + strzałki	Wciśnięcie Shift i strzałek pozwala szybciej przesuwać elementy.
Ctrl + B	Czyści program z uszkodzonych kabli.