Laboratorium Teorii i Techniki Sterowania

TTS02: Wprowadzenie do LabView, cz. 2

1. Data Structures in LabVIEW – struktury danych w LabVIEW

W tej sekcji:

- String Data Type – łańcuchowy (tekstowy) typ danych
- Numeric Data type – numeryczny typ danych
- Boolean Data Type – dwuwartościowy typ danych
- Dynamic Data Type – dynamiczny typ danych
- Arrays – tablice
- Clusters - klastry
- Enums – typ wyliczeniowy

String Data Type

Łańcuch jest ciągiem znaków ASCII. Typ łańcuchowy to typ niezależny od platformy. Niektóre z bardziej popularnych zastosowań ciągów znaków są następujące:

- Tworzenie prostych wiadomości tekstowych,
- Kontrolowanie sprzętu poprzez wysyłanie poleceń tekstowych do urządzeń i przechwytywanie odpowiedzi,
- Przechowywanie danych numerycznych na dysku. W celu przechowywania danych numerycznych w pliku ASCII należy najpierw przekonwertować dane numeryczne na ciągi znaków,
- Instrukcje lub okna dialogowe.

Na panelu przednim łańcuchy pojawiają się jako tabele, pola tekstowe i etykiety. LabVIEW zawiera wbudowane podprogramy i funkcje, które można wykorzystać do manipulacji ciągami. LabVIEW reprezentuje ciąg znaków kolorem różowym.

Numeric Data Type

LabVIEW przedstawia dane liczbowe jako liczby zmiennoprzecinkowe, liczby stałoprzecinkowe, liczby całkowite ze znakiem i bez znaku oraz liczby zespolone. Dane pojedynczej i podwójnej precyzji są koloru pomarańczowego, natomiast liczby całkowite koloru niebieskiego

Uwaga: Różnica między numerycznymi typami danych to liczba bitów, które wykorzystywane są do przechowywania danych oraz wartości danych, które reprezentują.

Niektóre typy danych zapewniają także rozbudowane opcje konfiguracyjne. Na przykład, można skojarzyć fizyczne jednostki miary z danymi zmiennoprzecinkowymi, można także konfigurować kodowanie i zakres dla danych stałoprzecinkowych.

Boolean Data Type

LabVIEW przechowuje dane logiczne jak wartości 8-bitowe. Można użyć Boolean do reprezentowania 0 lub 1, a także PRAWDA lub FAŁSZ. Jeśli wartość 8-bitowa wynosi zero, to wartość Boolean to FAŁSZ. Każda wartość niezerowa oznacza TRUE. Typowe zastosowania dla danych logicznych to prezentacja danych cyfrowych oraz użycie przełączników w panelu przednim. W LabVIEW kolor zielony oznacza typ boolean.

Dynamic Data Type

Większość Express VI przyjmuje i/lub zwraca dynamiczny typ danych, który pojawia się jako ciemnoniebieski port.

Korzystanie z Convert to Dynamic Data oraz Convert from Dynamic Data, pozwala przekonwertować zmiennoprzecinkowe danych numeryczne lub logiczne następujących typów danych:

- tablica 1D przebiegów,
- tablica 1D skalarów,
- 1D tablica skalarów - ostatnia wartość,
- 1D tablica skalarów – jeden kanał,
- tablica 2D skalary (kolumny są kanałami),
- tablica 2D skalary (wiersze są kanałami),
- pojedyncze skalary,
- pojedyncze przebiegi.

Należy podłączyć dynamiczny typ danych do wskaźnika, który najlepiej przedstawi dane. Wskaźniki to np. wykresy, diagramy jak też wskaźniki numeryczne czy Boolean. Dynamiczne dane ulegają automatycznej konwersji, aby dopasować się do wskaźnika, do którego są podłączone, zastosowanie Express VI może spowolnić proces wykonania kodu.

Dynamiczny typ danych jest typowy dla Express VI. Większość VI i funkcji, które są dostarczane z LabVIEW nie akceptują tego typu danych. Aby skorzystać z wbudowanego VI lub funkcji do analizy lub przetwarzania danych dynamicznych należy je najpierw przekonwertować.

Arrays

Czasami korzystnie jest grupować pokrewne dane. Warto do tego celu użyć tablic i klastrów. Tablice łączą punkty danych tego samego typu w jedną strukturę(tablicę), natomiast klastry łączą dane różnych typów w strukturę danych.

Tablica składa się z elementów i może być wielowymiarowa. Elementy są to konkretne wartości danych w tablicy.

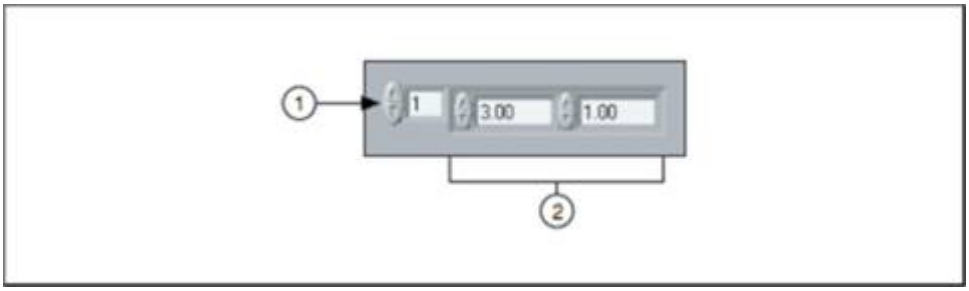
Można budować tablice typów numerycznych, boolean, tekstowych, przebiegów sygnałowych i klastrów. Zastanówmy się nad wykorzystaniem tablic podczas pracy z porcją podobnych danych i

podczas wykonywania powtarzalnych obliczeń. Tablice są idealne do przechowywania danych, które otrzymuje się z przebiegów sygnałów oraz danych generowanych w pętli, gdzie każda iteracja pętli produkuje jeden element tablicy.

Uwaga: indeksy tablicy w LabVIEW zaczynają się od 0. Indeks pierwszego elementu tablicy, bez względu na jej wymiar jest równy zero.

Elementy tablicy są uporządkowane. Tablica używa indeksu, dzięki czemu można łatwo uzyskać dostęp do konkretnego jej elementu. Indeks zaczyna się od zera, co oznacza, że zawiera się w zakresie od 0 do n-1, gdzie n oznacza liczbę elementów w tej tablicy. Na przykład, n-12 reprezentuje 12 miesięcy w roku, więc indeks tablicy przyjmuje wartości od 0 do 11. Marzec to trzeci miesiąc, więc ma indeks 2.

Rysunek 1 przedstawia przykład tablicy liczb. Pierwszy element przedstawiony w tablicy (3,00) ma indeks 1, a drugi element (1,00) ma indeks 2. Element o indeksie 0 nie został pokazany w tym rysunku, ponieważ wskaźnik indeksu zawiera wartość 1. Wskaźnik indeksu definiuje pierwszy wyświetlany element tablicy.

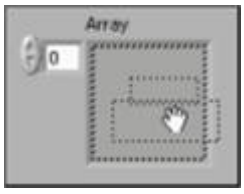


(1) Wskaźnik indeksu | (2) Wskaźnik wartości

Rysunek 1. Wskaźnik tablicy liczb

Creating Array Controls and Indicators

Zadajnik lub wskaźnik tablicy numerycznej tworzy się na przednim panelu przez dodanie warstwy tablicy, jak pokazano na rysunku 2, oraz przez przeciągnięcie obiektu lub elementu o konkretnym typie danych do pustej warstwy.



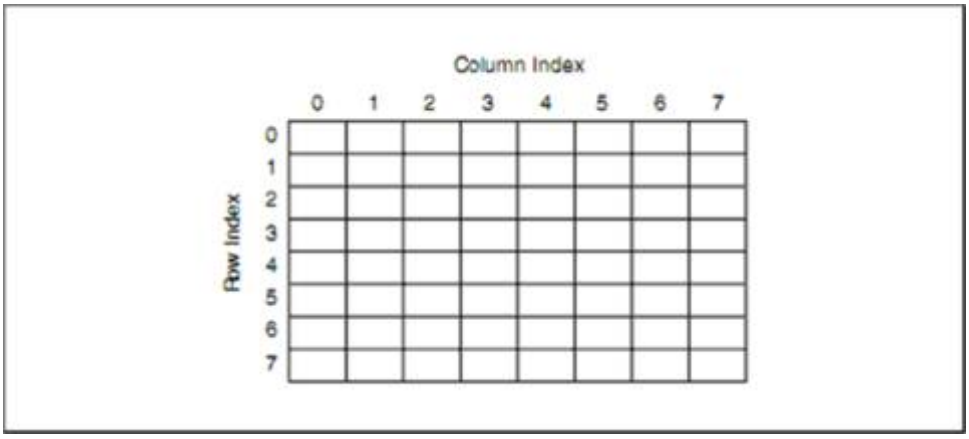
Rysunek 2. Umiejscowienie zadajnika numerycznego w warstwie tablicy

W przypadku próby przeciągnięcia nieprawidłowego zadajnika lub wskaźnika do warstwy tablicy, dany obiekt nie zagnieździ się w warstwie tablicy.

Należy wstawić obiekt w warstwie tablicy przed użyciem tej tablicy w schemacie blokowym. W przeciwnym przypadku terminal tablicy będzie czarny z pustymi nawiasami i nie będzie miał przypisanego typu danych.

Two-Dimensional Arrays

Poprzednie przykłady korzystają z tablic jednowymiarowych (1D). Dwuwymiarowa (2D) tablica przechowuje elementy w siatce. Wymaga to indeksowania kolumny i indeksowania wiersza. Oba indeksy zaczynają się od zera. Rysunek 3 przedstawia 8 kolumn i 8 wierszy z tablicy 2D, która zawiera 8 x 8 = 64 elementów.

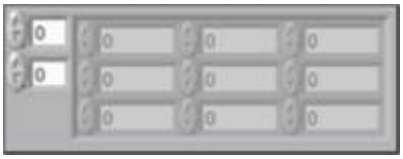


Rysunek 3. Tablica 2D

Aby dodać tablicę wielowymiarową na przednim panelu, należy kliknąć prawym przyciskiem myszy na wskaźnik indeksu i wybrać opcję Add Dimension z menu kontekstowego. Można również zmienić rozmiar wskaźnika indeksu, dopóki nie uzyska się żądanej ilości wymiarów tablicy.

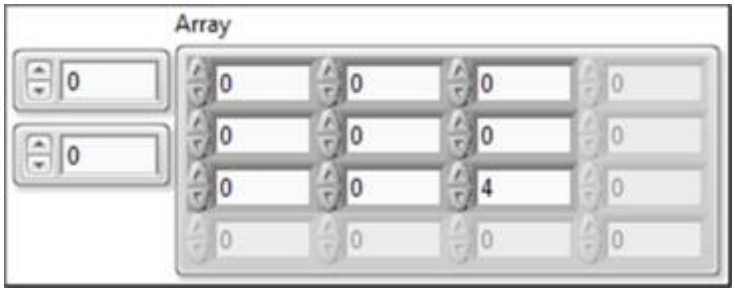
Initializing Arrays

Można zainicjować tablicę lub pozostawić niezainicjowaną. Gdy tablica jest inicjowana, to określa się ilość elementów w każdym wymiarze i zawartości każdego elementu. Niezainicjowana tablica zawiera stałą liczbę wymiarów, ale bez elementów. Rysunek 4 przedstawia niezainicjowany zadajnik tablicy 2D. Należy zauważyć, że wszystkie elementy są wygaszone. Oznacza to, że tablica jest niezainicjowana.



Rysunek 4. Niezainicjalizowana tablica 2D

W tablicy 2D, po zainicjowaniu elementów, każdy niezainicjowany element w kolumnie oraz w poprzednich kolumnach są inicjowane i wypełniane wartością domyślną dla danego typu danych. Na rysunku 5, wartość 4 została wpisana w kolumnie drugiej. Poprzednie elementy w kolumnie 0, 1 i 2 są inicjowane wartością 0, co jest wartością domyślną dla numerycznego typu danych.



Rysunek 5. Zainicjalizowana tablica 2D z dziewięcioma elementami.

Creating Array Constants

Aby utworzyć stałą w formie tablicy na schemacie blokowym, należy wybrać stałą tablicową na Functions Palette, umieścić warstwę tablicy na schemacie blokowym oraz dodać stałą o żądanym typie danych w warstwie tablicy. Można użyć stałej tablicowej do przechowywania danych lub jako podstawę do porównania z inną tablicą.

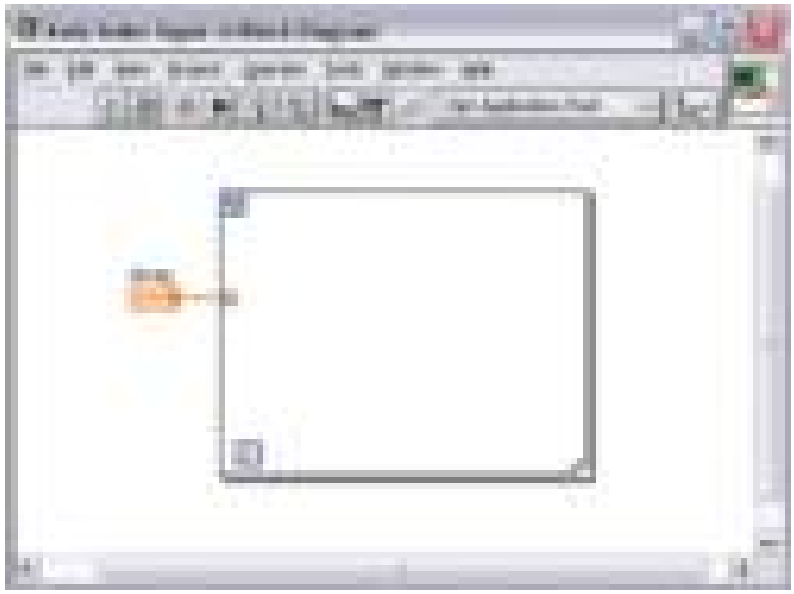
Auto-Indexing Array Inputs

Gdyby podłączyć tablicę do lub z pętlą for lub while, można byłoby skorelować każdą iterację pętli z elementem w tej tablicy. Umożliwia to automatyczne indeksowanie. Należy zmienić ikonkę tunelu (na obramowaniu pętli) na powyższą, aby włączyć automatyczne indeksowanie. W tym celu należy kliknąć prawym przyciskiem myszy i wybrać opcję Enable Indexing z menu kontekstowego. Opcja Disable Indexing działa analogicznie.

Array Inputs

Gdyby automatyczne indeksowanie zostało włączone dla tablicy podłączonej do gniazda wejściowego pętli For, LabVIEW ustawiłoby wartość portu licznika do rozmiaru tablicy, więc nie trzeba byłoby oddzielnie podłączać sygnału do portu licznika. Ponieważ można używać pętli do przetwarzania tablic jeden element na jeden obrót pętli, LabVIEW włącza automatyczne indeksowanie domyślnie dla każdej tablicy podłączonej do pętli for. Można wyłączyć automatyczne indeksowanie, jeśli nie trzeba przetwarzać tablicy jeden element na jeden obrót pętli.

Na rysunku 6, pętla for jest wykonywana tyle razy, ile jest elementów w tej tablicy. Zwykle, jeśli port licznika pętli for nie jest podłączony, to przycisk uruchamiania jest „uszkodzony”. Jednak w tym przypadku tak nie jest.



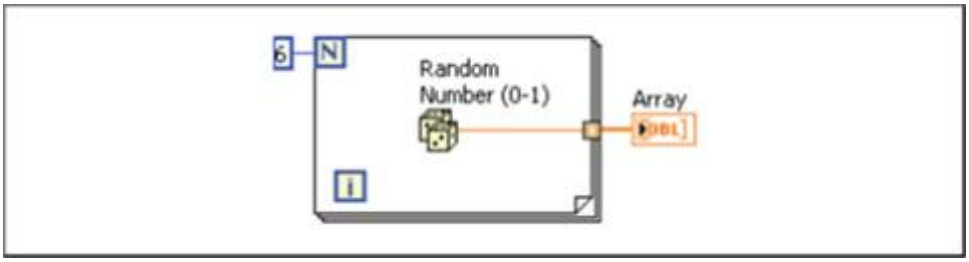
Rysunek 6. Tablica użyta do ustawienia licznika pętli for

W przypadku włączenia automatycznego indeksowania dla więcej niż jednego tunelu lub jeśli port licznika zostanie podłączony, rzeczywista liczba iteracji równa jest najmniejszej z tych wartości. Na przykład, jeśli dwie tablice automatycznie indeksowane (10 i 20 elementowa) podłączone są do pętli oraz podłączono wartość 15 do portu licznika, to pętla wciąż wykonuje się tylko 10 razy, indeksując wszystkie elementy pierwszej tablicy i tylko pierwsze 10 elementów drugiej tablicy.

Array Outputs

W przypadku włączenia autoindeksowania tablicy dla tunelu wyjściowego, tablica wyjściowa przyjmuje nowy element z każdej iteracji pętli. Dlatego autoindeksowane tablice wyjściowe są zawsze równe co do wielkości liczbie powtórzeń pętli.

Przewód z tunelu wyjściowego do wskaźnika tablicy staje się grubszy, ponieważ przewód zmienia się w tablicę na granicy pętli, a tunel wyjściowy zawiera nawiasy kwadratowe, co oznacza tablicę.



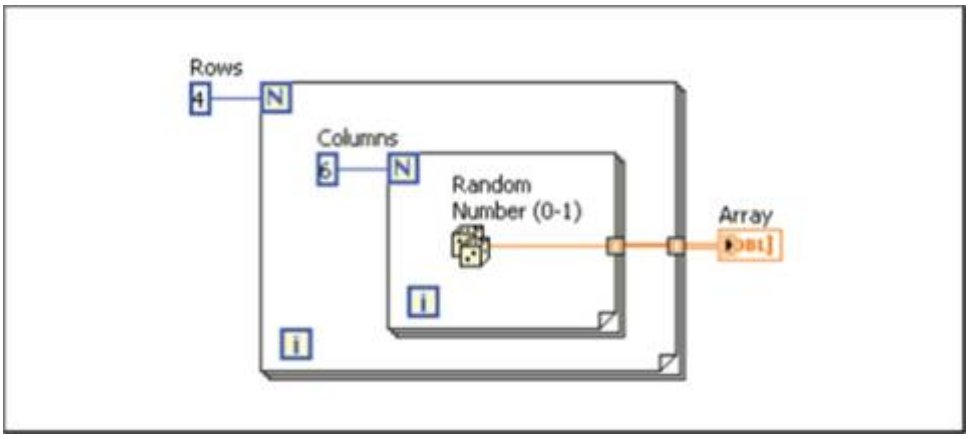
Rysunek 7. Autoindeksowane wyjście

Automatyczne indeksowanie dla pętli while jest domyślnie wyłączone.

Przykładowo należy wyłączyć automatyczne indeksowanie, jeśli potrzebna jest tylko ostatnia przekazana wartość z tunelu.

Creating Two-Dimensional Arrays

Można korzystać z dwóch pętli zagnieżdżonych jedna w drugiej w celu stworzenia tablicy 2D. Zewnętrzna pętla for tworzy elementy wierszy, a wewnętrzna pętla tworzy elementy kolumn.



Rysunek 8. Tworzenie tablicy 2D

Clusters

Klastry grupują dane różnych typów. Przykładem klastra jest klaster błędu LabVIEW, który łączy w sobie wartość logiczną(boolean), wartość liczbową i łańcuch. Klaster jest podobny do rekordu lub struktury w tekstowych językach programowania.

Połączenie kilku elementów danych w klastry eliminuje bałagan przewodów na schemacie blokowym i zmniejsza liczbę portów, których potrzebuje subVI. Panel złącza ma co najwyżej 28 portów. Jeśli przedni panel zawiera więcej niż 28 zadajników i wskaźników, które trzeba przekazać do innego VI, należy zgrupować niektóre z nich do klastra i przypisać klaster do portu w panelu złącza.

Większość klastrów w schemacie blokowym ma różowy wzór przewodu oraz portu. Klastry błędów mają ciemnożółty wzór przewodów i portów. Klastry wartości numerycznych, posiadają brązowy wzór przewodu i portów. Można połączyć brązowe numeryczne klastry z funkcjami numerycznymi, takimi jak dodawanie czy pierwiastek kwadratowy, aby wykonać te same czynności jednocześnie na wszystkich elementach klastra.

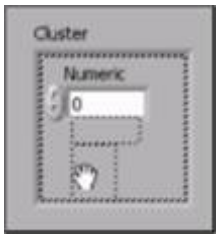
Order of Cluster Elements

Zarówno elementy klastrów jak i tablic są uporządkowane, jednak należy rozdzielać wszystkie elementy klastra na raz za pomocą funkcji rozdzielenia (Unboundle). Można użyć funkcji Unboundle By Name, aby rozdzielić elementy klastra według nazwy. Aby użyć w/w funkcji każdy element klastra musi mieć etykietę. Podobnie jak tablica, klaster może być zadajnikiem lub wskaźnikiem. Klaster nie może zawierać zarówno zadajników jak i wskaźników.

Creating Cluster Controls and Indicators

Zadajnik lub wskaźnik klastrowy na przednim panelu tworzy się przez dodanie pustej warstwy klastra, jak pokazano na rysunku 9, a następnie przeciągnięcie obiektu o konkretnym typie danych do wnętrza warstwy klastra.

Zmiana rozmiaru warstwy klastra polega na przeciągnięciu kursora podczas jej wstawiania.



Rysunek 9. Tworzenie zadajnika klastrowego

Rysunek 10 jest przykładem klastra zawierającego trzy zadajniki: ciąg znaków, przełącznik wartości logicznej i pole danych numerycznych.

Rysunek 10. Przykład zadajnika klastrowego

Creating Cluster Constants

Aby utworzyć stałą klastrową na schemacie blokowym, należy wybrać stałą klastrową na palecie Functions, następnie umieścić warstwę klastra na schemacie blokowym i umieścić stałą o konkretnym typie danych. Można użyć stałej klastrowej do przechowywania danych lub jako podstawę do porównania z innym klastrem.

Jeśli istnieje zadajnik lub wskaźnik klastrowy w oknie panelu przedniego i należy stworzyć stałą klastrową zawierającą te same elementy na schemacie blokowym, można przeciągnąć klaster z okna panelu przedniego do schematu blokowego lub też można kliknąć prawym przyciskiem myszy na klaster w oknie panelu przedniego i wybrać Create » Constant z menu kontekstowego.

Using Cluster Functions

Należy użyć funkcji klastrowych do tworzenia i manipulowania klastrami. Przykładowe zadania:

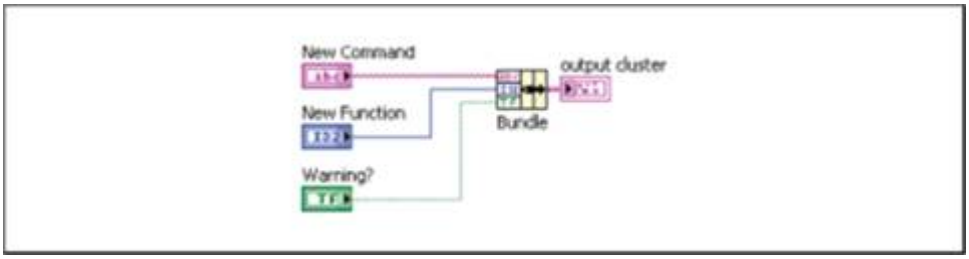
- Wyciąg danych z poszczególnych elementów klastra,
- Dodawanie poszczególne danych do klastra,
- Rozłożenie klastra na jego poszczególne składowe.
- Należy użyć funkcji:
- Bundle, aby utworzyć klaster,
- Bundle i Bundle by Name, aby zmodyfikować klaster,

- Unbundle i Unbundle by Name, aby rozłożyć klaster.

Można również umieścić te funkcje na schemacie blokowym klikając prawym przyciskiem na port klastra i wybierając Cluster, Class &Variant Palette z menu kontekstowego. Funkcje Bundle i Unbundle zawierają odpowiednią liczbę portów. Funkcje Bundle by Name i Unbundle by Name pojawiają się z pierwszym elementem klastra. Należy użyć narzędzia pozycjonowania (Positioning Tool), aby zmienić rozmiar powyższych funkcji i zobaczyć pozostałe elementy klastra.

Assembling Clusters

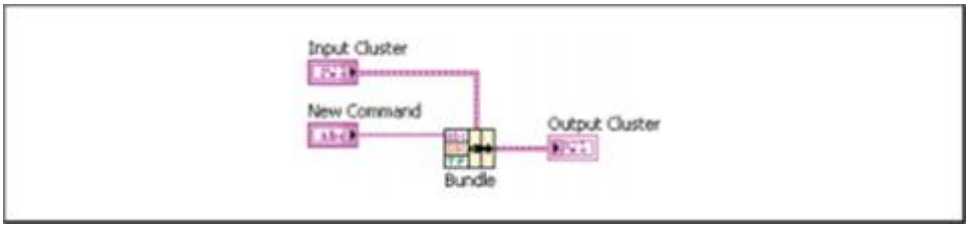
Należy użyć funkcji Bundle, aby utworzyć klaster z poszczególnych elementów lub zmienić wartości poszczególnych elementów istniejącego klastra bez konieczności określania nowych wartości dla pozostałych elementów. Za pomocą Positioning Tool należy zmienić rozmiar funkcji lub kliknąć prawym przyciskiem myszy na wejściowy port i wybrać Add Input z menu kontekstowego.



Rysunek 11. Tworzenie klastra na schemacie blokowym

Modifying a Cluster

Jeśli wejście klastra zostało podłączone, można podłączyć tylko te elementy, które użytkownik chce zmienić. Na przykład, klaster wejściowy (Input Cluster) pokazany na rysunku 12 zawiera trzy zadajniki.

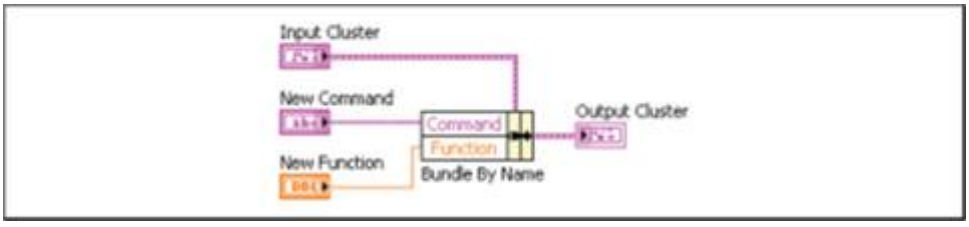


Rysunek 12. Modyfikacja klastra przy użyciu funkcji Bundle

Jeśli znana jest kolejność elementów w klastrze, można użyć funkcji Bundle, aby zmienić wartość Command tworząc okablowanie jak na rysunku 12.

Można również skorzystać z Bundle by Name, aby zmienić lub dostać się do określonych elementów istniejącego klastra. Bundle by Name działa jak funkcja Bundle, lecz zamiast odniesienia do kolejności elementów klastra odniesieniem jest nazwa elementu klastra.

W przykładzie z rysunku 13 można użyć funkcji Bundle by Name w celu uaktualnienia wartości Command oraz Function wartościami New Command i New Function.



Rysunek 13. Bundle by Name użyte do modyfikacji klastra

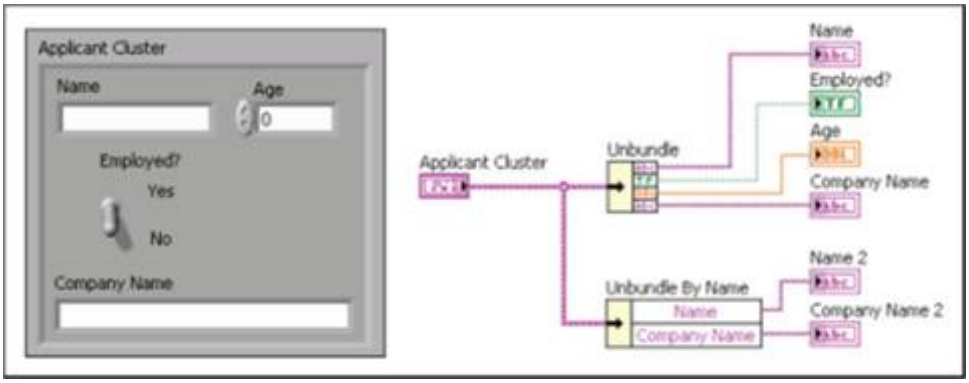
Należy użyć funkcji Bundle by Name dla struktur danych, które mogą się zmieniać w trakcie rozwoju oprogramowania. Gdyby dodać nowy element do klastra lub modyfikować jego kolejność, nie trzeba będzie ponownie oprzewodować funkcji Bundle by Name, ponieważ nazwy pozostaną aktualne.

Disassembling Clusters

Należy użyć funkcji Unbundle do podzielenia klastra na poszczególne elementy.

Należy użyć funkcji Unbundle by Name, aby dostać się do elementów składowych klastra, których nazwy można określić.

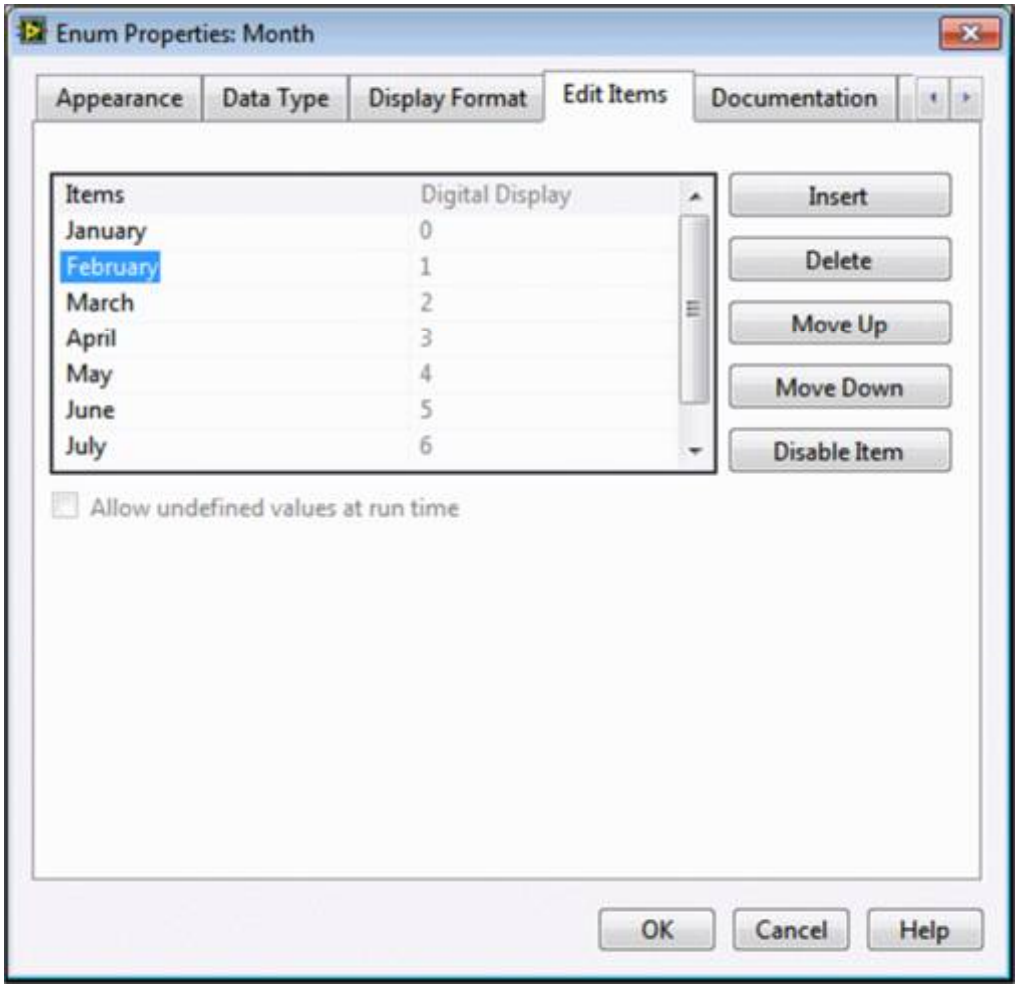
Na przykład, w przypadku korzystania z funkcji Unbundle na klastrze z rysunku 14 posiadającego cztery porty wyjściowe, które odpowiadają czterem obiektom w klastrze. Trzeba znać kolejność elementów klastra, dzięki czemu można powiązać prawidłowy port klastra z odpowiednim elementem w klastrze. W tym przykładzie, elementy są uporządkowane od góry do dołu, poczynając od elementu 0. Jeśli używasz Unbundle by Name można mieć dowolną liczbę zacisków wyjściowych i dostęp do poszczególnych elementów po nazwie w dowolnej kolejności.



Rysunek 14. Unbundle oraz Unbundle by Name

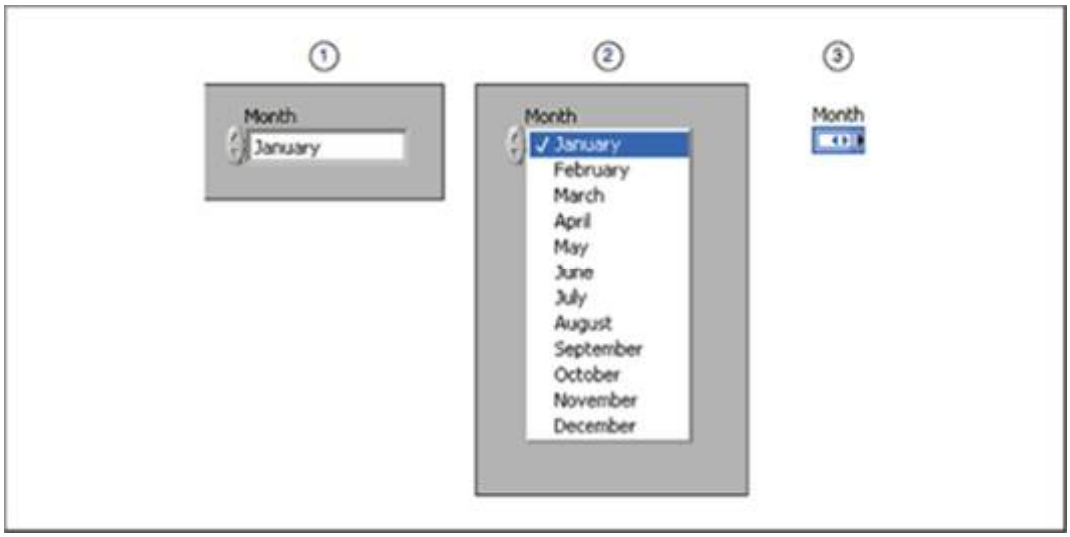
Enums

Enum (typ wyliczeniowy, spis) to zadajniki, stałe lub wskaźniki. Ten tym jest połączeniem dwóch typów danych: łańcucha i typu numerycznego występujących w parach. Na przykład, jeśli istnieje typ wyliczeniowy nazwie „Month”, możliwe pary wartości dla tej zmiennej to styczeń - 0, luty - 1, i tak dalej do grudnia -11. Rysunek 15 pokazuje przykład tych par danych w oknie dialogowym Properties. Jest ono bezpośrednio dostępne poprzez kliknięcie prawym przyciskiem myszy na obiekcie typu wyliczeniowego i wybraniu Edit Items.



Rysunek 15. Okno Properties dla zadajnika Month

Obiekty typu wyliczeniowego są przydatne, ponieważ na schemacie blokowym łatwiej jest manipulować numerami niż łańcuchami znaków. Rysunek 16 pokazuje zadajnik Month, wybór pary danych w zadajniku i skorelowany terminal na schemacie blokowym.



(1) Zadajnik na panelu przednim | (2) Lista wyboru | (3) Terminal na schemacie blokowym

Rysunek 16. Zadajnik Month

2. Execution Structures in LabVIEW – Struktury wykonawcze w LabView

W tej sekcji:

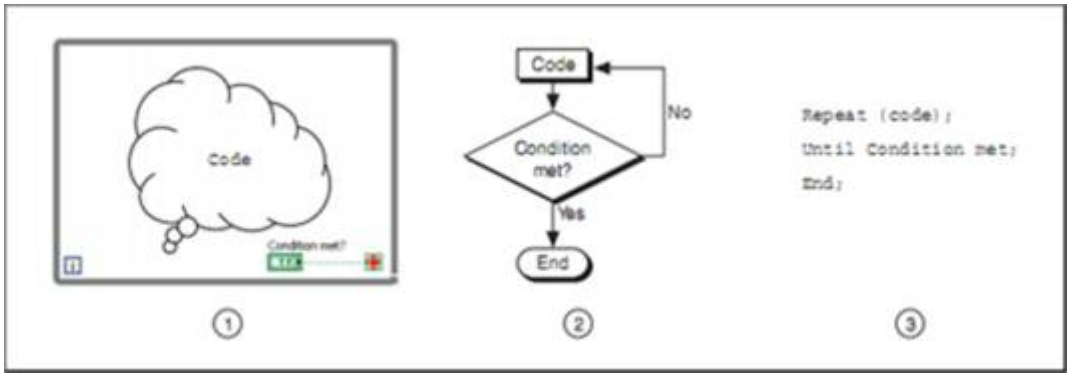
- Loops - pętle
- Case Structures – struktury case
- Other Structures – pozostałe struktury

Struktury wykonawcze zawierają sekcje kodu graficznego i kontrolują jak i kiedy kod wewnątrz jest wykonywany. Najczęstsze struktury wykonawcze to pętle While, For oraz struktura Case, których można użyć do uruchomienia tego samego fragmentu kodu wiele razy lub do wykonania różnych sekcji kodu w oparciu o pewny warunek.

Loops

While Loops

Pętla While jest podobna do pętli Do lub Repeat-Until w tekstowych językach programowania. Pętla While, pokazana na rysunku 1, wykonuje kod, który zawarty jest wewnątrz, dopóki warunek przerwania nie zostanie spełniony.



(1) pętla While | (2) diagram przepływu | (3) Pseudokod

Rysunek 1 Pętla While w LabVIEW

Aby wstawić pętlę while należy wybrać ją z palety Structures, a następnie za pomocą kursora, narysować prostokąt w schemacie blokowym. Do pętli dodaje się obiekty metodą „przeciągnij i upuść".

Pętla while wykonuje kod, który jest wewnątrz niej dopóki nie otrzyma określonego sygnału typu boolean na porcie wejściowym warunku zatrzymania.

Można również wykonać podstawową obsługę błędów przy użyciu portu warunku pętli while. Kiedy klaster błędu zostanie podłączony do portu warunku, tylko wartość parametru stanu klastra błędu (True lub False) przechodzi do portu. Ponadto, pozycje w menu: Stop if True i Continue if True zmieniają się na Stop if Error and Continue if Error.

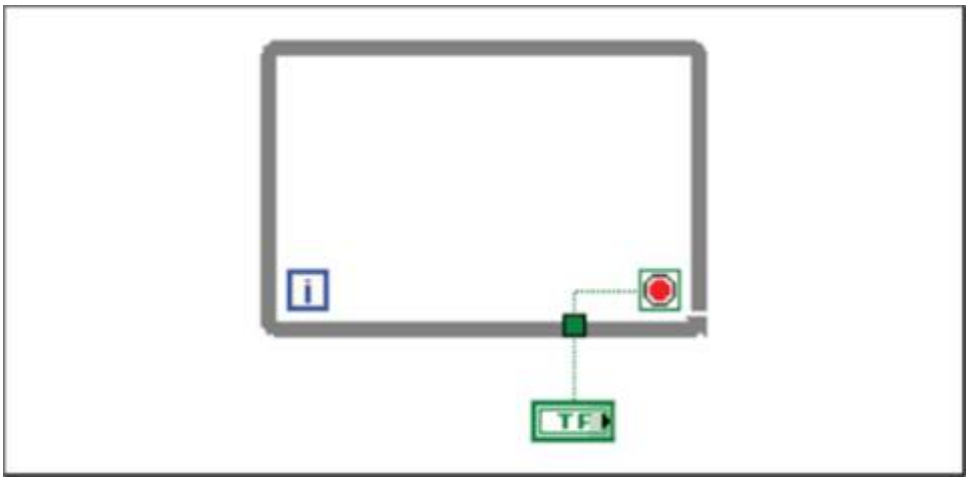
Infiniti Loops

Port iteracji to port wyjściowy, który zawiera liczbę ukończonych iteracji.

Liczba iteracji dla pętli while zawsze zaczyna się od zera.

Uwaga: Pętla While zawsze wykonuje się co najmniej raz.

Nieskończone pętle to częsty błąd programowania. Jeśli port warunku to Stop if True, a użytkownik umieści obiekt Boolean poza pętlą while, i w momencie rozpoczęcia pętli wartość na porcie będzie fałszem to spowoduje to pętlę nieskończoną.

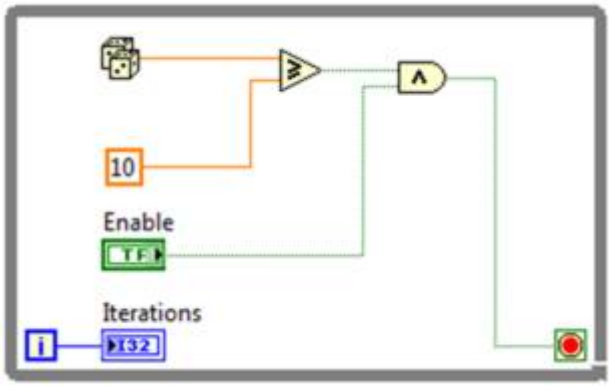


Rysunek 2. Zadajnik boolean poza pętlą While

Zmiana wartości zadajnika nie zatrzymuje nieskończonej pętli, ponieważ wartość ta czytana jest tylko raz przed rozpoczęciem pętli. Aby skorzystać z zadajnika zatrzymującego pętlę while, należy umieścić zadajnik wewnątrz pętli. Aby zatrzymać nieskończoną pętlę, należy przerwać VI klikając przycisk Execution Abort na pasku narzędzi.

Na rysunku 3 jest pętla while, która wykonuje się, dopóki funkcja Random Number zwróci wynik większy bądź równy 10.00 i zadajnik Enable będzie miał wartość True. Funkcja „i" zwraca wartość True tylko wtedy, gdy oba wejścia są prawdziwe. W przeciwnym wypadku zwraca False.

Pętla na rysunku 3 jest nieskończona, ponieważ funkcja Random nie generuje wartości równej lub większej niż 10,00.

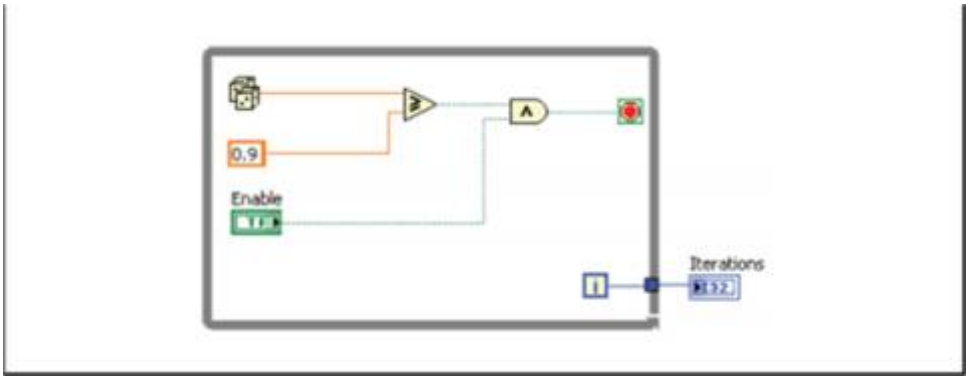


Rysunek 3. Pętla nieskończona

Structure Tunnels

Tunele przesyłają dane do i z struktur. Tunel pojawia się jako kwadrat na granicy pętli while (lub innych). Blok ma kolor typu danych podłączonego do tunelu. Dane wychodzą z pętli po zakończeniu jej wykonywania. Gdy tunel przekazuje dane do pętli, pętla wykonywana jest dopiero wtedy, gdy dane dotrą przez tunel.

Na rysunku 4 port iteracji jest połączony z tunelem. Wartość w tunelu nie przejdzie do wskaźnika iteracji dopóki pętla while nie zakończy wykonywania.

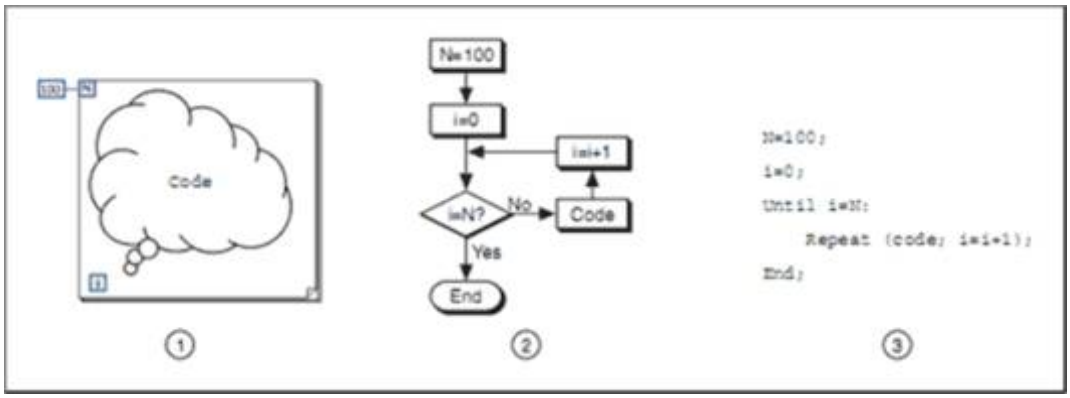


Rysunek 4. Tunel pętli While

Tylko ostatnia wartość iteracji zostanie wyświetlona we wskaźniku iteracji.

For Loops

Pętla for wykonuje fragment kodu określoną liczbę razy. Rysunek 5 prezentuje pętlę for w LabVIEW, ekwiwalent schematu blokowego i przykład pseudokodu dla tej pętli.



(1) Pętla for w LabVIEW | (2) Diagram przepływu | (3) Pseudokod

Rysunek 5. Pętla for

Pętla for jest na palecie Structures. Można również umieścić ją na schemacie blokowym klikając prawym przyciskiem myszy granicę pętli while i wybierając Replace with For Loop z menu kontekstowego.

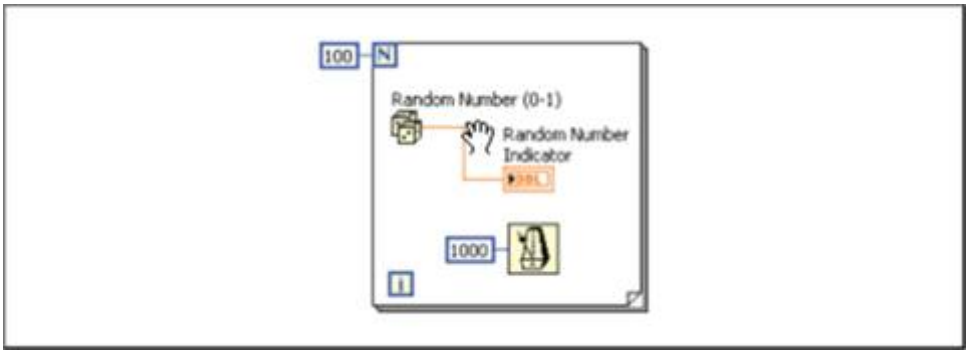
Port licznika iteracji pętli for to port wejściowy. Decyduje on ile razy powtórzyć kod z pętli.

Port iteracji to port wyjściowy, który zawiera liczbę ukończonych iteracji.

Liczba iteracji w pętli For zawsze zaczyna się od zera.

Pętla for różni się od pętli while tym, że pętla for wykonuje określoną liczbę razy, a pętla while zaprzestaje wykonywania tylko wtedy, gdy warunek zostanie spełniony.

Pętla for na rysunku 6 generuje losową liczbę co sekundę (do 100 sekund) i wyświetla losowe numery wskaźnikiem numerycznym.



Rysunek 6. Przykład pętli for

Adding Timing to Loops

Gdy pętla zakończy wykonywanie jednej iteracji, natychmiast rozpocznie wykonywanie kolejnej iteracji, o ile nie osiągnie warunku zatrzymania. Najczęściej trzeba kontrolować częstotliwość iteracji. Na przykład, jeśli pozyskuje się dane, a trzeba zdobyć dane raz na 10 sekund, potrzebny jest sposób na określenie czasu iteracji pętli na 10 sekund. Nawet jeśli nie potrzeba, aby wykonanie pętli nastąpiło z określoną częstotliwością, należy dać procesorowi czas na ukończenie innych zadań, takich jak obsługa interfejsu użytkownika.

Wait Function

Umieszczenie funkcji Wait wewnątrz pętli umożliwia VI przeczekanie określonego czasu. Pozwala to procesorowi na realizację innych zadań w czasie oczekiwania. Funkcje Wait używają milisekund jako podstawy zegara.

Funkcja Wait (ms) czeka, aż licznik milisekund odliczy czas równy czasowi określonemu w porcie wejściowym funkcji. Funkcja ta gwarantuje, że czas wykonywania jednej iteracji pętli jest równy co najmniej określonemu czasowi.

Case Structures

Konstrukcja Case ma dwa lub więcej podprogramy, czyli przypadki.

Tylko jeden przypadek jest widoczny na raz, a struktura wykonuje tylko jeden dany przypadek w tym samym czasie. Wartość wejściowa określa, który przypadek zostanie wykonany. Konstrukcja Case jest podobna do switch lub if... then... else w tekstowych językach programowania.

Etykieta „selektor przypadku” w górnej części konstrukcji Case zawiera nazwę wartości sterującej, która odpowiada przypadkowi.

Należy kliknąć na strzałki obok, aby przewinąć dostępne przypadki. Można również kliknąć strzałkę w dół obok nazwy przypadków i wybrać konkretny z menu rozwijanego.

Należy podłączyć przewód z wartością wejściową do portu wyboru w celu określenia, który przypadek (case) ma zostać wykonany.

Należy podłączyć liczbę całkowitą, wartość logiczną, ciąg lub wartość typu wyliczeniowego do portu wyboru. Można ustawić port wyboru w dowolnym miejscu na lewej krawędzi struktury Case. Jeśli typ danych portu wyboru jest wartością logiczną (Boolean) struktura ma dwa przypadki: dla wartości Prawda i Fałsz. Jeśli port wyboru jest liczbą całkowitą, łańcuchem, lub typem wyliczeniowym struktura Case może mieć dowolną liczbę przypadków.

Uwaga: Domyślnie, łańcuch wartości podłączony do wyboru jest wrażliwy na wielkość liter. Aby uczynić go niewrażliwym, należy podłączyć łańcuch znaków do portu wyboru, kliknąć prawym przyciskiem myszy granicę struktury Case i wybrać Case Insensitive Match z menu kontekstowego.

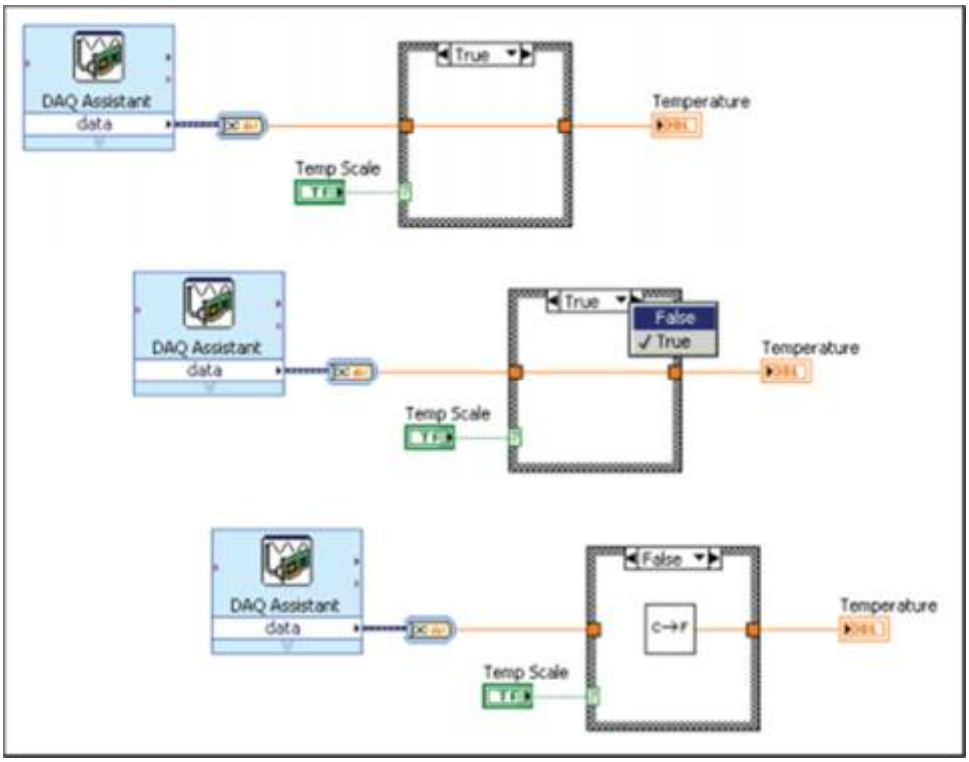
Jeśli domyślny przypadek dla struktury Case obsługujący wartości spoza zakresu nie zostanie określony, to trzeba wyraźnie określić kod dla każdej możliwej wartości wejściowej. Na przykład, jeśli przełącznik jest liczbą całkowitą i zostały określone przypadki dla wartości 1, 2 i 3,to należy także określić domyślny przypadek do wykonania, gdyby wartość wynosiła 4 lub więcej.

Uwaga: Nie można określić domyślnego przypadku jeśli podłączony został sygnał typu boolean. Należy ustawić zadajnik Boolean na PRAWDA lub FAŁSZ, aby ustalić przypadek do wykonania.

Kliknij prawym przyciskiem myszy na granicę struktury Case, aby dodać, powielić, usunąć lub zmienić kolejność przypadków oraz wybrać domyślny przypadek.

Selecting a Case

Rysunek 7 przedstawia VI, który wykorzystuje strukturę Case do wykonania różnych fragmentów kodu w zależności od tego, czy użytkownik wybierze °C czy °F jako jednostkę temperatury. Środkowa część schematu blokowego pokazuje przypadek dla wartości True na wejściu selektora. W dolnej części schematu blokowego został pokazany przypadek dla wartości Fałsz. Aby wybrać przypadek, należy wprowadzić wartość w etykiecie przypadku lub użyć narzędzia Labeling, aby edytować wartości. Po wybraniu innego przypadku, wyświetlany jest on na schemacie blokowym, jak pokazano w dolnej części rysunku 7.



Rysunek 7. Zmiana widoku przypadku w strukturze Case

Jeśli wprowadzi się wartość wyboru, która nie jest tego samego typu co obiekt podłączony do portu wyboru, to wartość ta zmienia kolor na czerwony. Oznacza to, że VI nie będzie działać do czasu usunięcia lub edycji tej wartości. Ponadto, ze względu na możliwość wystąpienia błędów zaokrągleń występującego w arytmetyce zmiennoprzecinkowej, nie należy używać liczb zmiennoprzecinkowych jako wartości selektora przypadku. Jeśli podłączy się wartość zmiennoprzecinkową do portu wyboru to LabVIEW zaokrągla ją do najbliższej liczby całkowitej. Gdyby wpisać wartość zmiennoprzecinkową w etykiecie selektora przypadku wartość ta zmieni kolor na czerwony, wskazując, że należy usunąć lub zmienić ją zanim struktura będzie mogła zostać wykonana.

Input and Output Tunnels

Można utworzyć wiele tuneli wejściowych i wyjściowych dla struktury Case. Wejścia są dostępne dla wszystkich przypadków, ale przypadki nie muszą używać każdego wejścia. Jednakże, należy zdefiniować wartość tunelu wyjściowego dla każdego przypadku.

Rozważmy następujący przykład: struktura case na schemacie blokowym ma tunel wyjściowy, ale co najmniej w jednym z przypadków, nie ma wartości wyjściowej podłączonej do tunelu. Jeśli uruchomi się taki przypadek, to LabVIEW nie wie, co ma zwrócić do wyjścia (tunelu). LabVIEW oznaczy ten błąd, kolorując wnętrze tunelu na biało. Aby naprawić ten błąd, należy wyświetlić przypadki, które zawierają nieokablowane tunele wyjściowe i oprzewodować sygnały wyjściowe do tunelu. Można także kliknąć prawym przyciskiem myszy na tunel wyjściowy i wybrać Use Default If Unwired z menu kontekstowego, aby użyć domyślnej wartości dla konkretnego typu danych tunelu. Gdy wyjście jest oprzewodowane we wszystkich przypadkach, tunel wyjściowy jest w jednolitym kolorze.

Należy unikać używania powyższej opcji. Używanie tej opcji nie wygląda dobrze na schemacie blokowym i może zmylić innych programistów używających takiego kodu. Opcja ta utrudnia również debugowanie. W przypadku korzystania z tej opcji, należy pamiętać, że używana wartość domyślna jest adekwatna dla typu danych, który jest podłączony do tunelu. Na przykład, jeśli tunel jest ma typ danych Boolean to wartość domyślna to false.

Data Type	Default Value
Numeric	0
Boolean	FALSE
String	empty (“”)

Table 1. Domyślne wartości typów danych

Other Structures

LabVIEW ma inne, bardziej zaawansowane rodzaje struktur wykonawczych, takich jak struktury zdarzeń (Event, używane do obsługi zadań z przerwaniami jak np. interakcja użytkownika), struktury sekwencji (Sequence, używane, aby wymusić kolejność wykonywania kodu), są one poza zakresem niniejszego materiału wprowadzającego. Aby dowiedzieć się więcej na temat tych struktur, należy odnieść się do odpowiedniego działu pomocy LabVIEW.

3. Passing Data Between Loop Iterations in LabVIEW – przenoszenie danych pomiędzy iteracjami pętli

W tej sekcji:

- Shift Registers – rejestry przesuwne
- Initializing Shift Registers – inicjalizacja rejestrów przesuwnych
- Stacked Shift Registers – stosowy rejestr przesuwny

Podczas programowania pętli, często trzeba uzyskać dostęp do danych z poprzednich iteracji. Na przykład, jeśli poddaje się akwizycji porcję danych w każdej iteracji pętli i trzeba obliczyć ich średnią co pięć iteracji, to należy zachowywać te dane.

Shift Registers

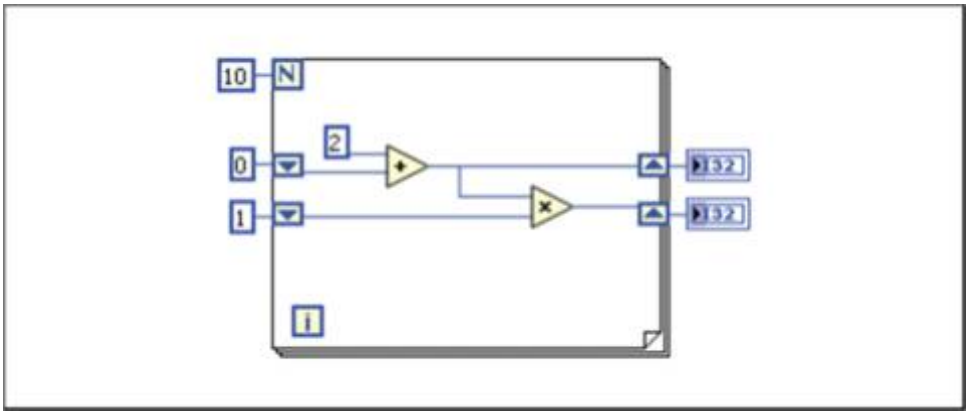
Należy użyć rejestrów przesuwnych, gdy trzeba przekazać wartości z poprzednich iteracji pętli do następnych iteracji. Rejestr przesuwny pojawia się jako para portów naprzeciw siebie na pionowych bokach obramowania pętli.

Port z prawej strony pętli zawiera strzałkę skierowaną do góry i przechowuje dane po zakończeniu iteracji. LabVIEW przekazuje dane zachowane z prawej strony do kolejnej iteracji. Po wykonaniu pętli port po prawej stronie pętli zwraca ostatnią wartość przechowywaną w rejestrze przesuwnym.

Należy utworzyć rejestr przesuwny klikając prawym przyciskiem myszy na lewej lub prawej granicy pętli i wybrać Add Shift Register z menu kontekstowego.

Rejestr przesuwny przenosi dowolny typ danych i automatycznie zmienia kolor zgodnie z typem danych pierwszego obiektu podłączonego do niego. Dane które podłącza się do portów tego samego rejestru przesuwnego muszą być tego samego typu.

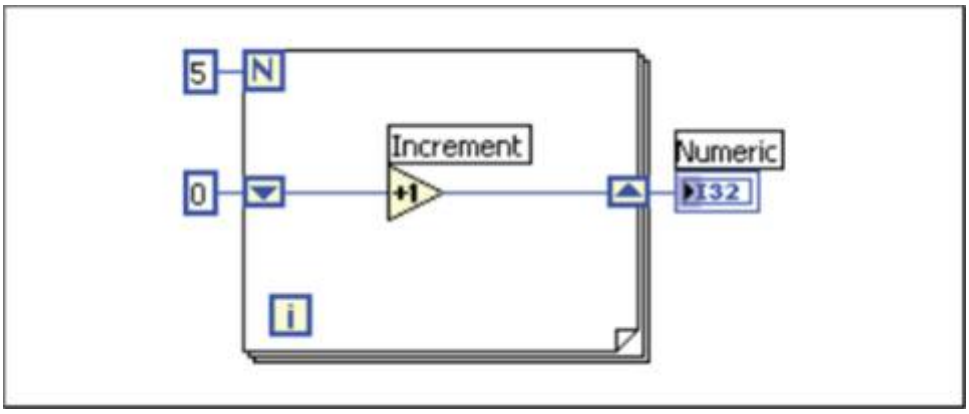
Można dodać więcej niż jeden rejestr przesuwny do pętli. Jeśli istnieje kilka operacji, które używają poprzednich wartości iteracji należy skorzystać z wielu rejestrów przesuwnych, jak pokazano na rysunku 1.



Rysunek 1. Używanie wielu rejestrów przesuwnych

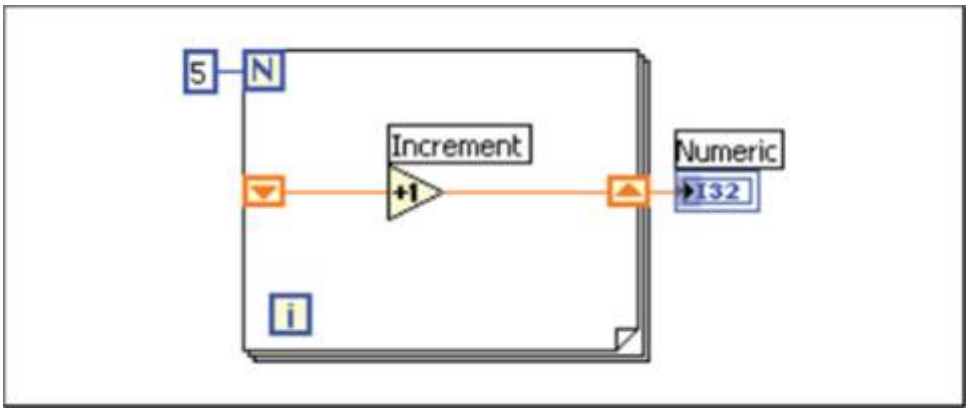
Initializing Shift Registers

Inicjowanie rejestru przesuwnego resetuje wartość, która używana jest do pierwszej iteracji pętli. Należy zainicjować pierwszą wartość w rejestrze przesuwnym przez podłączenie odpowiedniej wartości do portu rejestru przesuwnego po lewej stronie pętli, jak pokazano na rysunku 2.



Rysunek 2. Inicjalizacja wartości w rejestrze przesuwnym

Na rysunku 2pętla for wykonuje się pięć razy zwiększając wartość rejestru przesuwnego o jeden z każdą iteracją. Po pięciu iteracjach pętli for rejestr przesuwny wysyła ostateczną wartość 5do wskaźnika i VI kończy działanie. Po każdym uruchomieniu VI, wartość w rejestrze przesuwnym wynosi 0. Jeśli wartość początkowa rejestru przesuwnego nie została zainicjowana to pętla używa wartości wpisanej do rejestru przesuwnego podczas ostatniego wywołania lub, jeśli pętla nigdy nie została wykonana, domyślnej wartości dla typu danych. Należy użyć niezainicjowanego rejestru przesuwnego w celu zachowania informacji o stanie pomiędzy kolejnymi uruchomieniami VI.



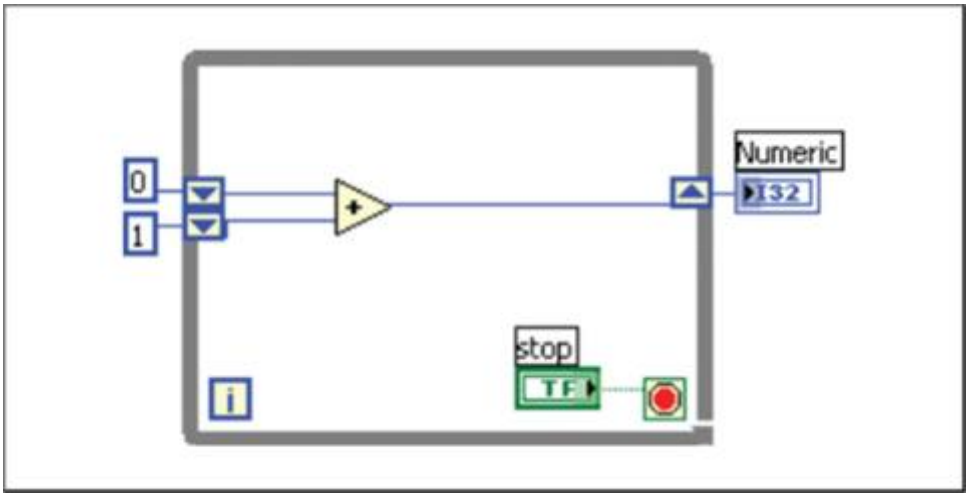
Rysunek 3. Niezainicjalizowany rejestr przesuwny

Na rysunku 3pętla for wykonuje się pięć razy zwiększając wartość rejestru przesuwnego o jeden z każdą iteracją. Przy pierwszym uruchomieniu VI rejestr przesuwny ma wartość 0, która jest wartością domyślną dla 32-bitowej liczby całkowitej. Po pięciu iteracjach pętli for rejestr przesuwny przesyła ostateczną wartość 5, do wskaźnika i VI kończy działanie. Następnym razem, gdy uruchomi się VI, rejestr przesuwny posiada wartość 5, która była ostatnią wartością z poprzedniego wykonania. Po pięciu iteracjach pętli for rejestr przesuwny przesyła ostateczną wartość 10 do wskaźnika. Gdyby ponownie uruchomić VI rejestr przesuwny zacząłby działanie od wartości 10 i tak dalej. Niezainicjowane rejestry przesuwne zachowują wartość poprzedniej iteracji, aż do zamknięcia VI.

Stacked Shift Registers

Stosując stosowe rejestry przesuwne można uzyskać dostęp do danych z poprzednich iteracji pętli. Stosowe rejestry przesuwne pamiętają wartości z wielu poprzednich iteracji i przekazują te wartości do kolejnych iteracji. Aby utworzyć stosowy rejestr przesuwny, należy kliknąć prawym przyciskiem myszy na lewy port i wybierać Add Element z menu kontekstowego.

Stosowy rejestr przesuwny może wystąpić tylko po lewej stronie pętli, ponieważ prawy zacisk przenosi dane generowane tylko z bieżącej iteracji.



Rysunek 4. Używanie stosowego rejestru przesuwnego

Gdyby dodać kolejny element do lewego zacisku w poprzednim schemacie blokowym, wartości z dwóch ostatnich iteracji zostałyby przeniesione do następnej iteracji, przy czym najświeższe dane są przechowywane w górnym porcie rejestru przesuwnego. Dolny port przechowuje najstarsze dane tj. z najwcześniej wykonanej iteracji.

4. Zadania do wykonania

1. Stworzyć kilkuelementową tablicę klastrów zawierającą kilka powiązanych danych dowolnego typu np. marka samochodu, model, rok prod., poj. silnika; uzupełniając ją dowolnymi danymi i wyświetlić ją na ekranie (tylko Front Panel)
2. Napisać program przeliczający stopnie Celsjusza, Farenheita i Kelwiny w każdej możliwej kombinacji przy wykorzystaniu struktury Case i typu wyliczeniowego Enum.
3. Wykorzystując kilka LED oprogramować węża świetlnego (można wykorzystać tablicę Boolean i rejestr przesuwny Shift Register).
4. Wypełnić tablicę 1D wartościami (y) wg jednej z danych funkcji: $y = x^2$, $y = x^3$, $y = 3^x$, $y = \ln(x)$ gdzie $0 \le x \le 30$ i, wykorzystując pętlę For, wyświetlić ją na ekranie w postaci wykresu.
   * wymusić spowolnienie rysowania – 1pkt/0.1 s.

Pytania kontrolne:

1. Co to jest klaster?
2. Czy można utworzyć klaster tablic, tablicę klastrów, klaster klastrów, tablicę tablic?
3. Z czego składa się typ wyliczeniowy?

4. Od jakiej wartości zaczynają się indeksy tablic?
5. Rodzaje tuneli przewodzących sygnał do pętli.
6. Rodzaje tuneli przewodzących sygnał z pętli.
7. Jaką funkcją tworzy się klaster?
8. Jaką funkcją modyfikuje się dane w klastrze?
9. Jaką funkcją wydobywa się dane z klastra?
10. Jak spowolnić działanie pętli?
11. Do czego służy struktura Case?
12. Jakie są domyślne wartości podstawowych typów danych?
13. Jak przenosić dane pomiędzy kolejnymi iteracjami pętli?