Podstawy programowania w LabView (3)

Tablice i klastry danych

Do tej pory poznaliśmy środowisko programistyczne i podstawowe elementy języka programowania graficznego. Teraz zaczniemy pracę z tablicami pozwalającymi na operowanie na dużych ilościach danych. Poznamy również typ złożony, jakim jest Cluster (Klaster) danych będący odpowiednikiem struktury znanej z języka C.

Największą zaletą przetwarzania komputerowego jest możliwość wykonywania operacji na ich dużej ilości danych. Dane te najczęściej gromadzone są w tablicach. LabView udostępnia zestaw funkcji pozwalający na wykonywanie na nich różnych operacji. Dlatego tę część kursu rozpoczniemy od zapoznania się nimi.

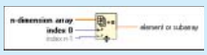
Paleta funkcji Array

Wszystkie funkcje pozwalające na manipulowanie danymi w tablicach zostały zgromadzone w palecie *Array* pokazanej na **rysunku 28**. Część z nich obsługuje tablice jednowymiarowe, a część jest skalowalna do tablic o dowolnym wymiarze. Funkcje – po dołączeniu tablicy wejściowej – zwykle same dopasowują liczbę wejść sterujących. Można to także zrobić łapiąc myszką i rozciągając funkcję, tak by uzyskać pożądaną liczbę wejść. Jeśli na ikonie w opisie poniżej znajduje się symbol kursora z dwoma strzałkami w górę i w dół, można rozciągnąć ikonę i dopasować liczbę wejść do rozmiaru tablicy.

array – tablica wejściowa może zawierać elementy dowolnego typu.

size(s) – skalar lub tablica jednowymiarowa zawierająca informacje o wymiarze tablicy *array*.

Array Size – zwraca liczbę elementów tablicy. Badając tablicę jednowymiarową otrzymujemy liczbę określającą wielkość tablicy. Badając tablicę n-wymiarową otrzymujemy tablicę jednowymiarową, a liczba elementów odpowiada wymiarowi tablicy. Na przykład, badając tablicę dwuwymiarową otrzymamy tablicę zawierającą dwa elementy. Pierwszy z parametrów informuje o liczbie wierszy, a drugi o liczbie kolumn.



n-dimension array – wejściowa tablica, dowolnego typu i wymiaru.

index 0..n-1 – zestaw indeksów określających zwracane przez funkcję elementy.

element or subarray – odczytany element lub fragment tablicy.

Index Array – Zwraca element z tablicy lub fragment tablicy. Funkcja jest skalowalna i może operować na tablicach o dowolnym rozmiarze. Do wejścia *n-dimension array* dołączamy badaną tablice, za pomocą wejść *index 0..n-1* określamy, który element chcemy odczytać. Dla tablic jednowymiarowych, w celu precyzyjnego określenia elementu, wystarczy podać numer elementu w tablicy. Dla tablic n-wymiarowych należy podać n indeksów. Na przykład, jeśli chcemy z tablicy dwuwymiarowej pobrać cały wiersz lub kolumnę, podajemy tylko jeden indeks z numerem interesującego nas wiersza lub kolumny, natomiast drugi pozostawiamy niepodłączony. Otrzymamy w ten sposób tablicę jednowymiarową, zawierającą wszystkie elementy z wybranego wiersza lub kolumny.

Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 63241, pass: 741obq51

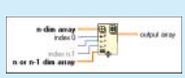
n-dimension array – tablica wejściowa, dowolnego typu i rozmiaru.

index 0..n-1 – zestaw indeksów określający miejsce wstawienia nowych elementów.

new element/subarray – nowy elementu lub tablica; gdy zamieniamy jedną kolumnę, podajemy wartość skalarną, w wypadku zamiany fragmentu tablicy, podajemy tablicę; dane dołączone do tej końcówki muszą mieć dokładnie taki sam typ, jak elementy tablicy.

output array – tablica wynikowa.

Replace Array Subset – Zamienia element lub fragment tablicy. Funkcja jest skalowalna i może operować na tablicach o dowolnych wymiarach. Dołączenie tablicy wejściowej powoduje pojawienie się odpowiedniej liczby wejść indeksowych. Do wejścia *n-dimension array* dołączamy tablicę, a korzystając z wejść *index 0..n-1* określamy pozycję, którą chcemy zamienić. Na przykład, aby w tablicy dwuwymiarowej zamienić wartość jednej komórki, musimy określić kolumnę i wiersz. Jeśli chcemy zamienić fragment tablicy np. wiersz, wskazujemy wiersz, który chcemy zamienić, a indeks określający kolumnę zostawiamy niepodłączony.

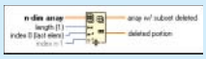


n-dim array – tablica wejściowa, dowolnego typu i rozmiaru.

index 0..n-1 – zestaw indeksów określających miejsce wstawienia nowych elementów.

n or n-1 dim array – jest elementem, kolumną lub wierszem który zostanie wstawiony w tablicę wejściową.

Insert Into Array – Wstawia nowy element lub tablicę w miejsce wskazane przez zestaw indeksów. Funkcja jest skalowalna i operuje na tablicach o dowolnych wymiarach.



n-dim array – tablica wejściowa o dowolnym typie i wymiarach.

length – określa, ile elementów ma być usuniętych.

index 0..n-1 – zestaw indeksów określających miejsce początku usuwanych elementów.

array w/subset deleted – wynikowa tablica niezawierająca usuniętych elementów.

deleted portion – usunięte elementy.

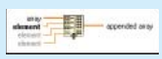
Delete From Array – usuwa z tablicy element lub fragment tablicy o określonej długości.

element **–** element, którym będzie wypełniona tablica – musi to być wartość skalarna. Typ danych określa rodzaj elementów w tablicy.

dimension size 0..n-1 – określa wymiary tablicy.

initialized array – wygenerowana tablica wypełniona elementami o tej samej wartości.

Initialize Array – generuje tablicę o podanym rozmiarze wypełnioną elementami o zdefiniowanej wartość i typie.

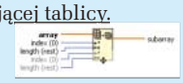


array – może to być tablica n-wymiarowa dowolnego typu.

element – wartość skalarna, pojedynczy element tablicy.

appended array – tablica wynikowa.

Build Array – Tworzy nową tablicę lub dodaje nowe komórki do istniejącej tablicy.



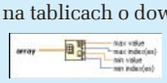
array **–** tablica wejściowa może zawierać elementy dowolnego typu.

index – określa pozycję pierwszego elementu w tablicy wejściowej, który zostanie zwrócony jako tablica wyjściowa.

length – określa ile kolejnych elementów kolumn lub wierszy zostanie zwróconych jako tablica wyjściowa.

Subarray – tablica wyjściowa; zawiera elementy tego samego typu, co wejściowa.

Array Subset – zwraca fragment tablicy wejściowej określony przez zestaw indeksów i długość. Funkcja jest skalowalna operuje na tablicach o dowolnych wymiarach.



Array – tablica wejściowa, może zawierać elementy dowolnego typu.

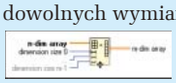
max value – maksymalna wartość znaleziona.

max index(es) – indeks lub zestaw indeksów określających pozycję wartości maksymalnej.

min value – minimalna wartość znaleziona.

min index(es) – indeks lub zestaw indeksów określających pozycję wartości minimalnej.

Array Max & Min – zwraca wartości minimalne i maksymalne z tablicy wejściowej, określa również miejsca w których się znajdują. Funkcja jest skalowalna i operuje na tablicach o dowolnych wymiarach.



n-dim array – tablica wejściowa, może zawierać elementy dowolnego typu.

dimension size 0..m-1 – określa wymiary tablicy wyjściowej; liczba terminali musi być zgodna z wymiarami tablicy wyjściowej.

m-dim array – tablica wyjściowa o wymiarach zgodnych z liczbą terminali *dimension size* i o wymiarach określonych przez te wartości.

Reshape Array – zmienia wymiary tablicy. Funkcja jest skalowalna i operuje na tablicach o dowolnych wymiarach. Jeśli wymiary tablicy wyjściowej są mniejsze od wejściowej, to elementy niemieszczące się zostaną pominięte. W przeciwnym wypadku, nadmiarowe elementy zostaną wypełnione zerami.

array – tablica wejściowa, jednowymiarowa, dowolnego typu.

sorted array – tablica posortowana od elementów najmniejszych do największych.

Sort 1D Array – funkcja sortuje rosnąco elementy tablicy jednowymiarowej.

1D array – jednowymiarowa tablica wejściowa; może zawierać elementy dowolnego typu.

element – wartość wyszukiwana w tablicy wejściowej.

start index – indeks, od którego rozpoczyna się wyszukanie; brak wartości powoduje przeszukanie od 0.

Search 1D Array – funkcja wyszukuje pozycję podanego elementu w tablicy jednowymiarowej i zwraca indeks, pod którym znajduje się element. Jeśli nie zostanie on znaleziony, to zwraca wartość -1. Można określić początek wyszukiwania.

array – tablica jednowymiarowa z elementami dowolnego typu.

index – miejsce podziału tablicy.

first subarray – pierwsza tablica zawierająca elementy od *array [0]* do *array [index-1]***.**

second subarray – druga tablica zawierająca pozostałe elementy tablicy *array***.**

Split 1D Array – funkcja dzieli tablicę jednowymiarową w miejscu wskazanym przez indeks na dwie tablice jednowymiarowe.

array – tablica jednowymiarowa zawierająca elementy dowolnego typu.

reversed array – tablica zawierająca elementy w odwróconej kolejności.

Reverse 1D Array – funkcja odwraca kolejność elementów w tablicy jednowymiarowej.

array – tablica jednowymiarowa zawierająca elementy dowolnego typu.

n – liczba rotacji.

array (last n elements first) – tablica wynikowa.

Rotate 1D Array – funkcja dokonuje rotacji elementów w tablicy jednowymiarowej. Gdy wartość *n* jest dodatnia, rotacja odbywa się zgodnie z kierunkiem zaznaczonym strzałką na ikonie, czyli pierwszy element staje się drugim, ostatni pierwszym itd. Jeśli wartość *n*

jest ujemna, to pierwszy element przechodzi na koniec tabeli, a ostatni staje się przedostatnim, natomiast drugi pierwszym.

array 0..n-1 – jednowymiarowe tablice wejściowe zawierające elementy dowolnego typu.

interleaved array – tablica wyjściowa zawierająca przeplecione elementy tablic wejściowych.

Interleave 1D Arrays – funkcja przeplata elementy jednowymiarowych tablic wejściowych w taki sposób, że pierwszy element tablicy wynikowej jest pierwszym elementem pierwszej tablicy, drugi element jest pierwszym elementem drugiej tablicy. Element *n-1* jest pierwszym elementem ostatniej tablicy. Element *n* jest drugim elementem pierwszej tablicy itd. Jeśli liczba elementów w tablicach nie jest równa, to przeplot ogranicza się tylko do tylu elementów, ile znajduje się w najmniejszej tablicy.

array – jednowymiarowa tablica zawierająca elementy dowolnego typu.

elements 0, n, 2n – pierwsza tablica wyjściowa zawierająca elementy *0, n,2n…*

elements 1, n+1, 2n+1 – druga tablica wyjściowa zawierająca elementy *1, n+1, 2n+1…*

Decimate 1D Array – rozplata elementy jednowymiarowej tablicy w taki sposób, że pierwszy element jest pierwszym elementem pierwszej tablicy wyjściowej. Drugi jest pierwszym elementem drugiej tablicy wyjściowej. Element *n* jest drugim elementem pierwszej tablicy wyjściowej, element *n+1* jest pierwszym elementem drugiej tablicy wyjściowej itd., gdzie *n* jest równe liczbie tablic wyjściowych. Jest to funkcja odwrotna do *Interleave 1D Arrays*.

2D array – dwuwymiarowa tablica wejściowa, może zawierać elementy dowolnego typu.

transposed array – transponowana tablica.

Transpose 2D Array – funkcja dokonuje transpozycji tablicy dwuwymiarowej, zamieniając wiersze z kolumnami.

array – tablica wejściowa może zawierać elementy dowolnego typu.

cluster – klaster danych jest zbudowany w ten sposób, że jego pierwszy element jest pierwszym elementem tablicy wejściowej, drugi jest drugim itd.

Array To Cluster – funkcja dokonuje konwersji typu tablicowego do typu *Cluster*. Tablica wejściowa musi być jednowymiarowa. Klikając prawym przyciskiem myszki możemy zdefiniować wielkość klastra danych. Domyślnie jest on równy 9, a maksymalnie może wynieść 256.

cluster – wejściowe dane typu *cluster*; nie może to być tablica.

Array – tablica wyjściowa.

Cluster To Array – funkcja dokonuje konwersji danych typu cluster na typ tablicowy. Działa w taki sposób, że pierwszy element w klastrze danych jest pierwszym elementem tablicy wyjściowej, drugi element jest drugim elementem tablicy wyjściowej itd.

Tworzenie tablic na panelu czołowym

Tablice mogą przechowywać dane dowolnego typu, dlatego utworzenie elementu tablicowego wymaga kilku etapów. Jako przykład pokażę, jak przygotować na panelu czołowym kontrolkę *(Controls)* przechowującą dane w tablicy, określić jej wymiary i typ przechowywanych danych. Utworzenie pozostałych, czyli wskaźnika *(Indicator)* i stałej *(Constans)* przebiega analogicznie z tą różnicą, że tworząc wskaźnik tablicowy wstawiamy element typu wskaźnik, a w wypadku stałych wszystkie czynności wykonujemy w oknie diagramu i posługujemy się stałymi.

Przygotowując tablicę na należy wykonać następujące czynności:

1. W oknie panelu czołowego, z palety *Array, Matrix & Cluster* przeciągnąć element *Array*. Symbolizuje on tablicę, ale nie jest ona jeszcze kompletna, ponieważ nie znamy typu danych, który będzie przez nią przechowywany.
2. Określić typ przechowywanych danych. Można to zrobić przeciągając do wnętrza tablicy element wybranego typu np. *Numeric Control*, jeśli chcemy utworzyć tablice przechowujące wartości liczbowe. Teraz mamy już tablicę jednowymiarową zawierającą elementy wybranego typu.
3. W razie potrzeby rozciągnąć tablicę, aby była widoczna większa liczba elementów. Robimy to najeżdżając na krawędzie tablicy, chwytając za czarny kwadracik, który się tam pojawi. Trzymając go rozciągamy tablicę do pożądanych wymiarów. Należy zwrócić uwagę na to, że „czarne kwadraciki" mogą pojawić się na krawędzi tablicy i na krawędzi elementów wewnątrz tablicy. W pierwszym wypadku, zmiana dotyczy tablicy, a w drugim elementów wewnątrz tablicy.
4. Określić wymiary tablicy. Domyślnie tablica jest jednowymiarowa i jeśli jest potrzebna większa, należy określić jej wymiary. Można to zrobić na dwa sposoby: poprzez wybór z menu lokalnego, rozwijanego po kliknięciu prawym przyciskiem myszy na krawędzi tablicy i wybór *Add Dimension* lub *Remove Dimension*, albo poprzez rozciągnięcie za pomocą myszki liczby indeksów znajdujących się po lewej stronie tablicy. Na **rysunku 29** pokazano kolejne etapy tworzenia tablicy na panelu czołowym.

Programowe generowanie tablic

Oprócz przygotowania zmiennej tablicowej na panelu czołowym, można wygenerować tablicę programowo. Najkorzystniej jest wykorzystać do tego celu pętle, ponieważ mechanizm indeksowania tuneli wyjściowych jest najbardziej wydajny, a program niepotrzebnie nie wywołuje funkcji dodatkowych. Podczas omawiania pętli *For* wygenerowaliśmy tablicę jednowymiarową, w kolejnym przykładzie przygotujemy tablicę dwuwymiarową i przy okazji policzymy tabliczkę mnożenia. Zrobimy to na dwa sposoby: korzystając tylko z pętli *For* oraz z funkcji palety *Array*.

W nowym pliku przygotujmy program, jak na **rysunku 30**. Należy pamiętać, aby tunele wyjściowe gromadziły dane w tablicach. Wewnętrzna pętla w tunelu wyjściowym gromadzi wartości licznika iteracji zwiększone o 1. Po każdym jej wywołaniu zostaje wygenerowana tablica jednowymiarowa z elementami od 1 do 10. Zewnętrzna

pętla mnoży otrzymaną tablicę przez licznik iteracji powiększony o 1. W ten prosty sposób obliczyliśmy tabliczkę mnożenia. Należy zwrócić uwagę na funkcję mnożącą. Widać tu, że nie jest to prosta funkcja wykonująca operację mnożenia liczby przez liczbę, ale mnoży każdy element tablicy przez skalar. Po podłączeniu zmiennych, dopasowuje się do ich typu. Może również wykonywać mnożenie dwóch tablic. W kolejnym przykładzie przygotujemy program, który również wyliczy tabliczkę mnożenia, ale skorzystamy z kilku funkcji palety *Array*. Porównując te dwa programy pokażę zalety dobrze przemyślanego kodu.

Przygotujmy program, jak na **rysunku 31**. Do zainicjowania tablicy wypełnionej wartością 1 wykorzystamy funkcję *Initialize Array*. Wynik jej działania widzimy na panelu czołowym, w tabeli *initialized array*. Do określenia rozmiaru tablicy posłużymy się funkcją *Array Size*, której wynik działania widzimy w tabeli *Rozmiar tablicy:* pierwszy element określa liczbę wierszy, natomiast drugi – kolumn. Do pobrania z tablicy liczb określających ilości wierszy i kolumn użyjemy *Index Array*. Otrzymane wartości posłużą do określenia liczby iteracji każdej z pętli. W pętli wewnętrznej należy dodać mnożenie.

Przyjrzyjmy się dwóm ostatnim przykładom. Zawartość pętli nie zmieniła się znacząco. Mamy tylko dodatkowe mnożenie, które w naszym wypadku wykona się 100 razy wydłużając czas wykonania się programu. Przed pętlą wywołaliśmy jeszcze trzy funkcje, które również mogą wpłynąć na czas wykonania się programu, a przy tym zwiększają wielkość kodu wynikowego oraz zaciemniają diagram. W tym miejscu możemy nieco uprościć kod rezygnując ze sprawdzania wielkości tablicy, pamiętając o korzyściach płynących z indeksowania. Nie musimy znać wymiarów tablic, pętle wykonają się tylko tyle razy, ile elementów znajduje się w tablicy. Zaciski ***N*** na krawędziach pętli mogą zostać niepodłączone.

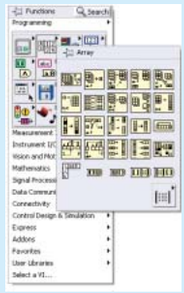
Celem tego przykładu było nie tylko przećwiczenie posługiwania się funkcjami, ale również pokazanie, jak można zoptymalizować kod uzyskując dzięki temu oszczędność czasu, przejrzystość kodu i zyskać na szybkości wykonywania.

Operacje na tablicach z wykorzystaniem funkcji palety Array

Przygotujemy kilka programów demonstrujących działanie niektórych funkcji z palety **Array**. Jako pierwszy przygotujmy program:

- generujący tablicę dziesięcioelementową,
- zapisujący do niej wartości od 0 do 9.

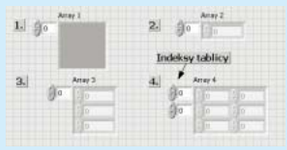
Zainicjowanie tablicy wypełnionej zerami odbywa się podobnie, jak w poprzednim przykładzie za pomocą funkcji *Initialize Array*. Wartość dołączona do końcówki *dimension size* posłużyła również do określenia liczby iteracji pętli. Takie ujęcie pozwoli na swobodne określanie wymiarów tablicy bez ingerowania w program. Nie możemy tutaj skorzystać z indeksowania elementów tablicy, ponieważ wykorzystujemy rejestr przesuwny. Wewnątrz pętli funkcja *Replace Array Subset*, przy każdym obiegu pętli, zamienia jeden element tablicy. Do wskazywania na określony element używamy licznika iteracji pętli. Wartości, które są wpisywane do tablicy, też pochodzą z licznika. Rejestr przesuwny przekazuje do funkcji *Replace Array Subset* tablicę zmodyfikowaną w poprzedniej iteracji. Diagram powinien wyglądać jak na **rysunku 32**. Proponuję uruchomienie programu w trybie animacji i śledzenie wartości w tabeli *Wynik fun. Replace* na panelu czołowym.



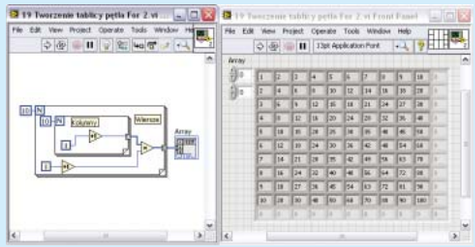
Rysunek 28. Paleta funkcji Array

W kolejnym przykładzie wykorzystamy funkcje wpisujące i usuwające elementy tablicy. Program ma za zadanie wygenerowanie tablicy dziesięcioelementowej zawierającej elementy od 0 do 10, a następnie usunięcie z niej wybranych elementów.

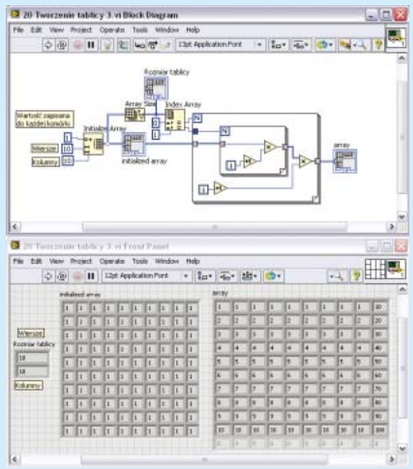
Tym razem wygenerujemy tablicę korzystając z funkcji *Insert Into Array*, która podczas każdego obiegu pętli wpisze do tablicy jedną wartość. Podobnie jak poprzednio, licznik iteracji pętli posłuży do wskazania miejsca zapisu i wartości. Rejestr przesuwny jest zainicjowany pustą stałą tablicową. Służy ona tylko do określenia wymiaru tablicy i typu przechowywanych elementów. Wartość wpisana do kontrolki *Ilość iteracji* decyduje o liczbie elementów. Gotową tablicę pobieramy z rejestru przesuwnego na prawej krawędzi pętli. Proponuję uruchomienie programu w trybie animacji i śledzenie jak jest tworzona tablica. Po każdej iteracji we wskaźniku *Tablica* na panelu czołowym można zobaczyć jak zapisywane są elementy. Usunięcie z tablicy wybranych elementów zostało zrealizowane za pomocą funkcji *Delete From Array*. Przed jej wywołaniem należy określić ile elementów chcemy usunąć i od którego należy zacząć. Do ustawiania tych parametrów przewidziałem dwie kontrolki na panelu czołowym: *Długość* określa ile elementów chcemy usunąć, a *Indeks* skąd rozpoczynamy usuwanie. Mogą one mieć postać suwaków. Uruchamiając program w trybie ciągłym i zmieniając nastawy suwaków można



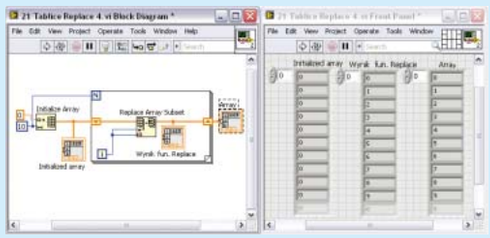
Rysunek 29. Etapy tworzenia tablicy



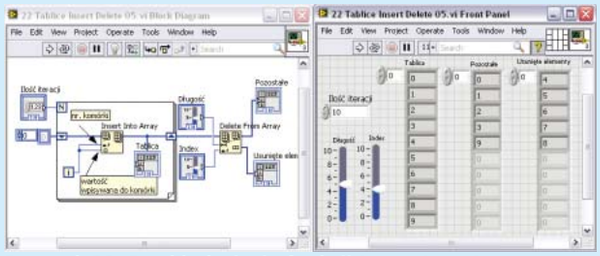
Rysunek 30. Generowanie tablicy dwuwymiarowej za pomocą pętli *for*



Rysunek 31. Przykład użycia funkcji z grupy *array*



Rysunek 32. Przykład użycia funkcji *Replace*



Rysunek 33. Przykład użycia funkcji *Insert Into Array* oraz *Delete From Array*

łatwo prześledzić działanie funkcji obserwując wskaźniki *Pozostałe* i *Usunięte elementy*. Gotowy diagram pokazano na **rysunku 33**.

W ostatnim przykładzie prześledźmy działanie funkcji dokonującej przeplotu elementów tablicy i ponownego ich rozplecenia. Aby wynik działania był czytelny przygotujmy trzy stałe tablicowe z tekstem. Wpiszmy do nich tekst, jak na **rysunku 34**. Niech liczba elementów w tablicach będzie zgodna z tą na rysunku. Po wstawieniu *Interleave 1D Arrays* są widoczne tylko dwa zaciski wejściowe. Korzystając z myszki należy rozciągnąć ikonę, aby było ich trzy. Podobnie postępujemy z funkcją *Decimate 1D Array*. Połączmy wszystko jak pokazano na **rysunku 34** i sprawdźmy jak działa program. Po uruchomieniu, efekt działania *Interleave 1D Arrays* widzimy na panelu czołowym w tablicy *Spleciona*. Patrząc na wartości w tabeli łatwo zorientować się jak działa funkcja. Popatrzmy teraz na tablice *Wynikowa1…3* – powinny one być prawie takie same, jak nasze tablice wejściowe, ale czegoś w nich brakuje. Zwróćmy uwagę na to, że druga tablica wejściowa zawierała pięć elementów, a pozostałe cztery. Wynikowe zawierają tylko cztery, piąty element z drugiej tablicy został „zgubiony". Wynika to z właściwości funkcji, dokonuje ona przeplotu tylu elementów ile zawiera najmniejsza tablica, należy o tym pamiętać korzystając z tych funkcji.

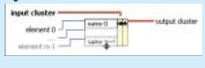
Tablice pozwalają przechowywać bardzo duże porcje danych, ale mogą to być dane tylko jednego typu. Często zachodzi potrzeba jednoczesnego przesłania różnych informacji. Aby nie prowadzić wielu połączeń i nie definiować wielu zmiennych w LabView powstał typ *Cluster* (Klaster) grupujący dane różnego typu w jednej zmiennej. Korzystanie z niego upraszcza znacznie pisanie programu i czyni diagram bardziej przejrzystym. Zmienną typu klaster można utworzyć na kilka sposobów. Opiszę jak zrobić to na panelu czołowym i na diagramie oraz przedstawię funkcje pozwalające na wykonywanie operacji na tych zmiennych.

Wybrane elementy z palety funkcji Cluster, Class, & Variant

input cluster – dane wejściowe typu klaster.

element 0..m-1 – rozgrupowane elementy.

Unbundle By Name – *rozgrupowuje* dane typu klaster. Pozwala wydobyć elementy posługując się ich nazwą. Możliwe jest jednoczesne wydobycie wszystkich składowych, lub tylko wybranych elementów. Przy pomocy myszki należy rozciągnąć ikonę tak aby funkcja zwracała określoną ich ilość.

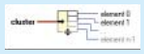


input luster – dane wejściowe typu klaster.

element 0..m-1 – elementy składowe klastra danych, identyfikuje się je dzięki nazwie.

output luster – dane wejściowe po zmodyfikowaniu wybranych elementów.

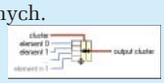
Bundle By Name – grupuje dane do typu klaster, pozwala na modyfikację wybranych elementów klastra danych.



cluster – klaster danych.

element 0..n-1 – elementy składowe klastra.

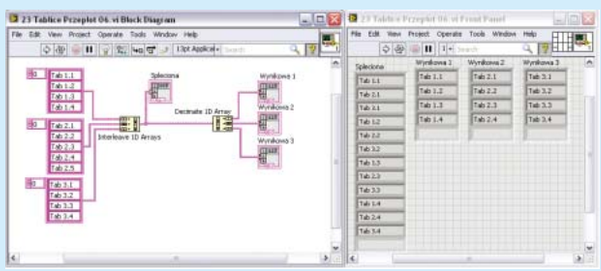
Unbundle – *rozgrupowuje* klaster danych. Kolejność elementów wyjściowych jest taka jak kolejności elementów w klasterze danych.



Bundle – grupuje dane do typu klaster, pozwala na modyfikację elementów klastra danych.

Tworzenie klastra na panelu czołowym

Na panelu czołowym możemy utworzyć kontrolkę i wskaźnik. Postępowanie w obu przypadkach jest identyczne z taką różnicą, że w pierwszym wypadku posługujemy się kontrolkami, a w drugim wskaźnikami. Analogicznie tworzymy stałą na diagramie posługując się stałymi.

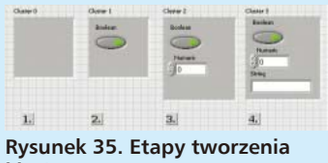


Rysunek 34. Przykład przeplotu

Dla przykładu zbudujmy kontrolkę zawierającą zmienną logiczną, numeryczną i tekstową. Rozpoczynamy od wstawienia elementu *Cluster* z palety *Array, Matrix & Cluster*. Następnie do jego wnętrza przeciągamy wszystkie potrzebne zmienne. **Rysunek 35** przedstawia wygląd klastra danych w kolejnych etapach jego tworzenia. Elementy klastra mają kolejność zgodną z kolejnością wstawiania elementów. Możemy to oczywiście zmienić, klikając prawym klawiszem myszki na krawędzi klastra i wybierając opcję *Reorder Controls In Cluster…*, wyświetli się aktualna kolejność zmiennych i pojawi się możliwość jej zmiany. Jest to istotna informacja, ponieważ nie można połączyć dwóch pozornie takich samych zmiennych, jeśli kolejność zmiennych wewnętrznych jest różna. Nie decyduje o tym również sposób ich ułożenia.

Praca z klastrami danych

Obsługa typu klaster (*cluster*) ogranicza się do podstawowych operacji grupowania i rozgrupowania danych. Funkcje

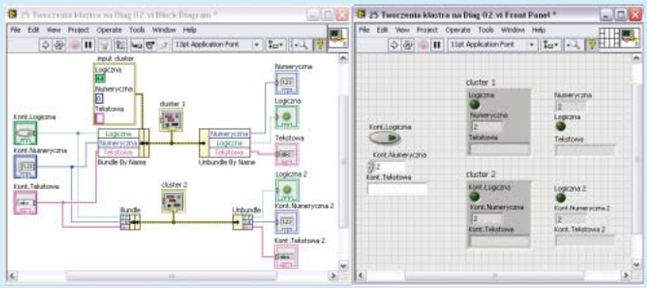


Rysunek 35. Etapy tworzenia klastra

do tego przeznaczone zostały opisane powyżej. Przykład demonstruje ich działanie.

Rysunek 36 przedstawia diagram demonstrujący tworzenie zmiennej typu *Cluster* korzystając z *Bundle* oraz *Bundle By Name* i ich rozgrupowanie za pomocą *Unbundle By Name* i *Unbundle*.

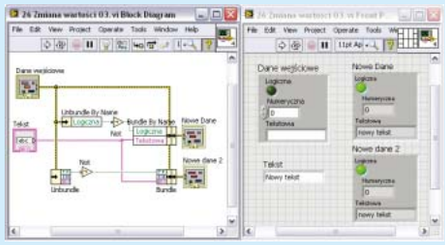
Pewnego komentarza wymaga tutaj różnica pomiędzy funkcjami. Otóż funkcja *Bundle* grupując elementy w jedną zmienną może posłużyć się nazwami elementów wejściowych i nie wymaga ich wcześniejszego zdefiniowania, natomiast *Bundle By Name* wymaga dołączenia przynajmniej stałej do wejścia *input cluster*. Stała ta musi mieć zdefiniowaną nazwę każdej składowej. W przeciwnym razie nie będzie możliwe dołączenie wejść. Zwróćmy uwagę na elementy *cluster 1* i *cluster 2* na panelu czołowym. W pierwszym, nazwy elementów składowych są zgodne ze zdefiniowanymi w stałej, natomiast w drugim są zgodne ze zmiennymi podpiętymi do wejść *Bundle*. *Rozgrupowanie* w obu przypadkach przebiega tak samo, ale korzystając z *Unbundle* kolejność zacisków jest zgodna z kolejnością składowych i zawsze widoczne są wszystkie zacisk. W przypadku *Unbundle By Name*, klikając na nazwach składowych możemy zmienić ich kolejność również ukryć te, które nas w danym momencie nie interesują.



Rysunek 36. Przykład budowania klastrów danych na diagramie

Modyfikacja elementów klastra danych

Rysunek 37 przedstawia przykład zmiany wartości wybranych elementów. Modyfikacji podlega zmienna logiczna, wartość numerycznej nie została zmieniona a do zmiennej tekstowej wpisano nową wartość. Rozgrupowanie elementów realizują funkcje *Unbundle* i *Unbundle By Name*. Różnice w ich stosowaniu są nieznaczne i nie wymagają komentarza. Do zapisu nowych wartości użyto funkcji *Bundle By Name* i *Bundle*. Użycie pierwszej z nich wymaga dołączenia do wejścia *Input Custer* danych wejściowych i rozciągnięciu jej tak, aby liczba zacisków wejściowych odpowiadała liczbie zmiennych, które chcemy zmodyfikować. Posługując się nazwami składowych elementów należy przesłać do odpowiednich zacisków nowe wartości. Korzystając z drugiej, po dołączeniu wejścia *Input Custer*, liczba zacisków wejściowych odpowiada liczbie zmiennych składowych. Zmiana wartości wybranych elementów odbywa się przez przesłanie nowych. Zaciski elementów, które nie podlegają modyfikacji, należy zostawić niepołączone lub podłączyć w odpowiednim wyjściem *Unbundle*.



Rysunek 37. Zmiana wartości wybranych elementów klastra

Podsumowanie

W tej części poznaliśmy pracę z tablicami i klastrami danych. W kolejnej poznamy pracę z tekstem, zasady tworzenia wykresów, operacje na plikach i programowe sterowanie właściwościami obiektów panelu czołowego.

Wiesław Szaj
wszaj@prz.edu.pl