INSTYTUT TELEKOMUNIKACJI MULTIMEDIALNEJ

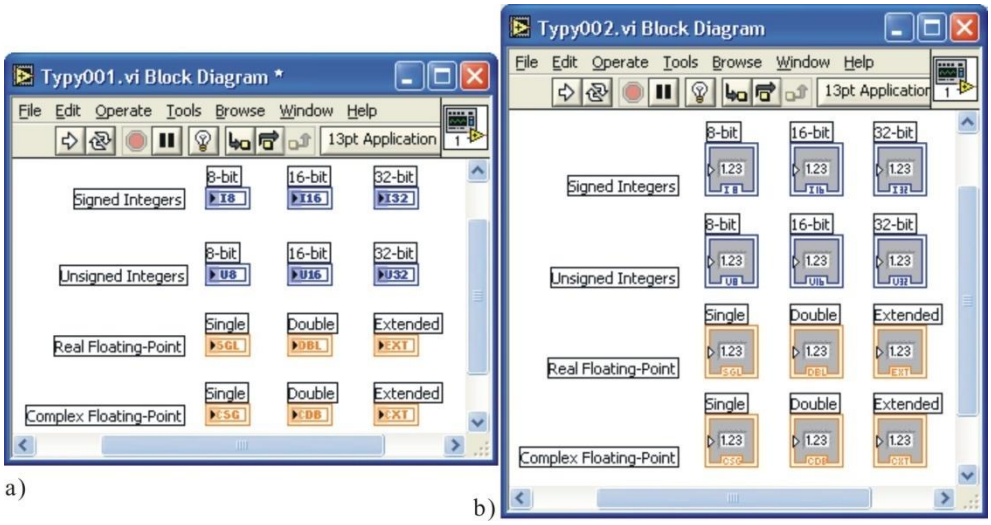
LABORATORIUM KOMPUTEROWYCH SYSTEMÓW POMIAROWYCH

Instrukcja do ćwiczenia:

Podstawy programowania w języku G (cz. 2)

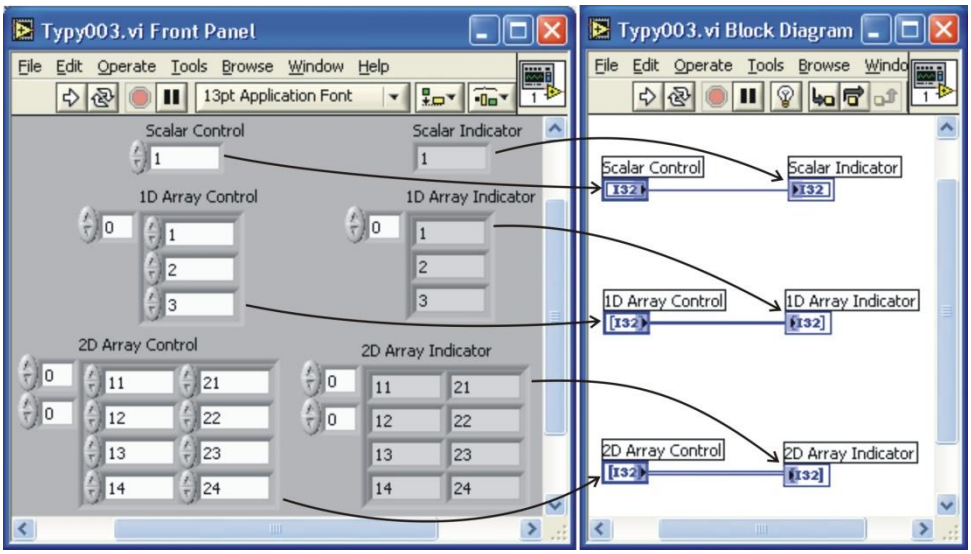
1. Wprowadzenie

Środowisko LabVIEW wykorzystuje programowanie graficzne w języku G (ang. graphical programming language). Programowanie polega na wyborze odpowiednich obiektów graficznych udostępnianych przez język oraz ich łączeniu. Obiekty graficzne symbolizują fragmenty kodu wykonywanego przez system. Linie łączące poszczególne obiekty programu pokazują przepływ danych w programie. W języku G każdy typ zmiennych posiada swój symbol graficzny rysowany przyporządkowanym do tego typu zmiennych kolorem. Oznaczenia graficzne typów zmiennych numerycznych w oknie *Blok Diagram* przedstawiono na rys. 1. Dla każdego typu zmiennych możliwe są dwa oznaczenia w postaci terminala (rys. 1a) lub w postaci ikony (rys. 1b). W oknie *Front Panel* wszystkie przedstawione na rys. 1 typy zmiennych będą miały taką samą reprezentację graficzną.



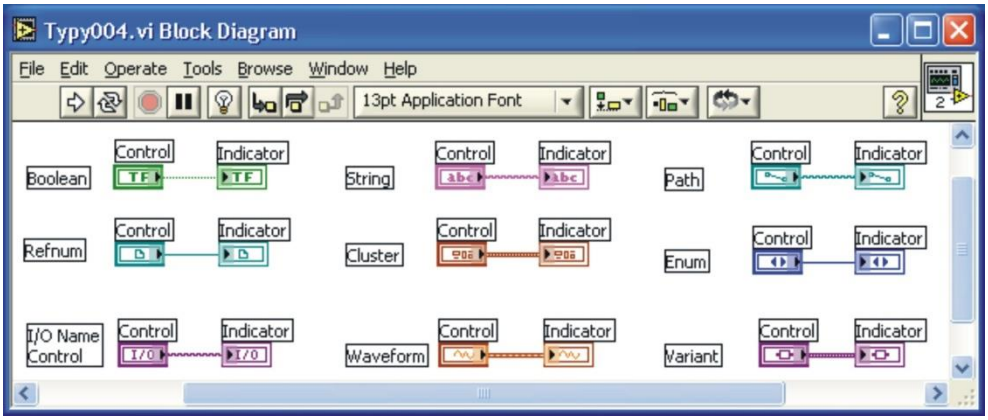
Rys.1. Podstawowe typy zmiennych numerycznych w języku G

Każdy z przedstawionych na rysunku 1 typów zmiennych numerycznych może występować w postaci skalara, tablicy 1D i tablicy wielowymiarowej ponadto wszystkie postaci zmiennych mogą pełnić funkcję obiektu *Control* lub *Indicator* lub *Constant*. Na rys. 2 przedstawiono symbole dla zmiennych numerycznych *I32* w postaci skalara, tablicy 1D i tablicy 2D w oknach *Front Panel* i *Block Diagram*. W oknie *Blok Diagram* zmienne numeryczne narysowane są w postaci terminali. Dla porównania na rys. 4 przedstawiono zmienne typu *I32* w postaci ikon. Przedstawiony na rys. 4 program realizuje przesyłanie danych z obiektu *Control* do obiektu *Indicator*. Każda z postaci zmiennych numerycznych przedstawionych na rys. 2 (patrz także rys. 4) jest oznaczana swoim symbolem graficznym.



Rys. 2. Symbole graficzne dla różnych postaci zmiennych numerycznych typu *I32* w języku G

Różnie są także rysowane linie łączące obiekty. Obiekty *Control* są źródłami danych a obiekty *Indicator* służą do wyświetlania danych (prezentacji wyników). Posiadają one swoją reprezentację graficzną w oknach *Front Panel* i *Block Diagram*. Obiekt *Control* służy do wprowadzania danych przez użytkownika programu (np.: nastaw, danych do obliczeń). Dane wpisane do tych obiektów mogą być modyfikowane po uruchomieniu programu. Jeżeli nastawy czy dane do obliczeń są wprowadzane przez programistę należy skorzystać z obiektu *Constant*. Obiekty *Constant* posiadają swoją reprezentację graficzną tylko w oknie *Block Diagram*. Dane do tych obiektów mogą być wpisane tylko raz na etapie tworzenia programu i po uruchomieniu programu nie mogą być modyfikowane. Na rys. 3 przedstawiono wybrane typy zmiennych nienumerycznych dostępne w języku G oraz wygląd linii łączących obiekty tych typów. Każda ze zmiennych nienumerycznych posiada unikalny symbol graficzny i jest rysowana przyporządkowanym do danego typu zmiennej kolorem.



Rys. 3. Podstawowe typy zmiennych nienumerycznych w języku G

Podstawą programowania w języku G jest zrozumienie i opanowanie stosowania obiektów sterujących, do których zaliczamy: obiekty sterującą *Sequence* (*Stacked Sequence* i *Flat Sequence*), obiekt sterujący *Case Structure*, obiekt *Formula Node*, pętle *For Loop*, pętle *While Loop* oraz operator *Shift Register*. Obiekty *Sequence* odpowiadają instrukcji grupującej „{}" w języku C. Służą ona do grupowania obiektów języka G, co zwiększa przejrzystość programu i umożliwia wprowadzenie zależności czasowych pomiędzy fragmentami wykonywanego kodu. Obiekt sterujący *Case Structure* jest odpowiednikiem instrukcji wyboru *if* i *switch* w jezyku C. Obiekt ten umożliwia wykonanie odpowiedniego fragmentu kodu programu w zależności od wyników kodu programu wykonanego wcześniej. Pętle *For Loop* i *While Loop* wraz z operatorem *Shift Register* odpowiadają instrukcją iteracyjnym *do .. while* i *for* w języku C. Konstrukcje te służą do wielokrotnego wykonywania fragmentu kodu programu. Po zastosowaniu elementu *Shift Register* można je wykorzystać do obliczeń iteracyjnych. Obiekt *Formula Node* służy do wprowadzania wyrażeń znanych z języka C. W obrębie tego obiektu można stosować: operatory arytmetyczne („+" – znak plus, „-" –znak minus, „++" – inkrementacja, „--" – dekrementacja, „+" – dodawanie, „-" – odejmowanie, „*" – mnożenie, „/" – dzielenie, „%" – reszta z dzielenia„**" – wykładnik potęgi), operatory logiczne („!" – negacja, „&&" – koniunkcja argumentów, „||" – alternatywa argumentów), operatory bitowe („~" – negacja zestawu bitów, „&" – koniunkcja zestawu bitów, „^" – różnica symetryczna zestawu bitów, „|" – alternatywa zestawu bitów, „>>" – przesunięcie w prawo, „<<" – przesunięcie w lewo), operatory relacji („= =" – równy, „!=" – różny, „<" – mniejszy, „>" – większy, „<=" – mniejszy bądź równy, „>=" – większy bądź równy) oraz warunkowy („?"). Dostępne są również następujące funkcje wbudowane : *abs, acos, acosh, asin, asinh, atan, atanh, ceil, cos, cosh, cot, csc, exp, expm1, floor, getexp, getman, int, intrz, ln, lnp1, log, log2, max, min, mod, rand, rem, sec, sign, sin, sinc, sinh, sqrt, tan, tanh*. W obrębie obiektu *Formula Node* można deklarować zmienne oraz stosować instrukcje sterujące znane z języka C. Można zadeklarować zmienne następujące typów: *float, float32, float64, int, int8, int16, int32, uInt8, uInt16, uInt32*. Nazwy zmiennych muszą się składać ze znaków alfanumerycznych (od „a" do „z", od „A" do „Z", od „0" do „9" oraz „_"). Do instrukcji języka C, które można stosować w obiekcie *Formula Node* zaliczamy: instrukcję warunkową *if*, instrukcję wyboru *switch*, instrukcje iteracyjne; *for, while, do ... while*, oraz instrukcje *break* i *continue*.

2. Cel ćwiczenia

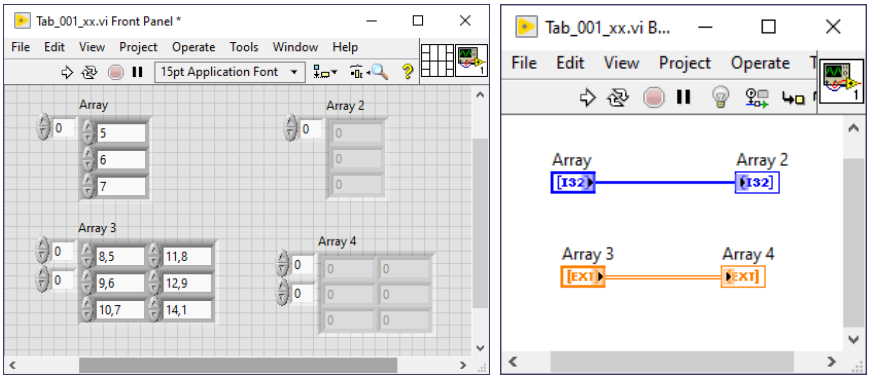
Celem ćwiczenia jest zapoznanie się ze sposobem programowania z wykorzystaniem języka G. W trakcie ćwiczenia studenci zapoznają się z:
– wykorzystaniem tablic w LabVIEW,
– zastosowaniem obiektów typu *Cluster*,
– lokalnymi odwołaniami do zmiennych w programie VI,
– zmianą parametrów obiektów za pomocą *Property Node*.

3. Przebieg ćwiczenia

3.1. Tablice.

3.1.1. Deklaracja i inicjalizacja tablicy (sposób 1).

W celu zapoznania się ze sposobem deklaracji tablicy zrealizować opisane poniżej zadania.
- Uruchomić LabView i utworzyć nowy program VI wybierając *BlankVI*.
- Przejść do okna *Front Panel* i wstawić obiekt *Controls » Modern » Aray,Matrix&Cluster » Array*, następnie wybrać obiekt *Controls » Modern » Numeric » Numeric Control* i przeciągnąć w obręb obiektu *Array*.
- Z menu obiektu *Numeric Control* wybrać opcję *Representation » I32*.
- Rozmiary obiektu *Array* zwiększyć tak, aby wewnątrz tego obiektu pojawiły się 3 obiekty *Numeric Control*.
- Przejść do okna *Block Diagram*, narzędziem *Connect Wire* najechać na wyjście wstawionego obiekty, nacisnąć prawy przycisk myszy i z menu obiektu wybrać opcję *Create » Indicator*.
- W opisany powyżej sposób utworzyć tablicę 2D ze zmiennymi *EXT*, do zamiany tablicy 1D na tablicę 2D wykorzystać opcję *Add Dimension* z menu obiektu *Array*,
- Zapisać program, nadając mu nazwę „Tab_001_xx.vi” gdzie xx inicjały twórcy programu.
- W celu inicjalizacji tablic należy do tablic wpisać odpowiednie wartości (patrz rys. 3.1.1).
- Uruchomić program i sprawdzić jego działanie.
- Przed przejściem do następnego zadania z menu okna *Front Panel* wybrać *Edit>>Make Current Values Default* i zapisać program.

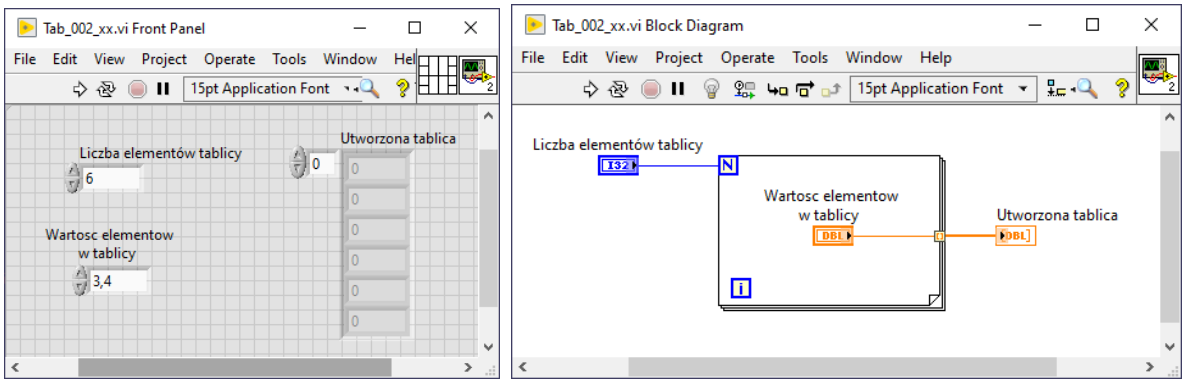


Rys. 3.1.1. Deklaracja i inicjalizacja tablic. Okna *Front Panel* i *Block Diagram* programu Tab_001_xx.VI.

3.1.2. Deklaracja i inicjalizacja tablicy (sposób 2).

W celu zapoznania się z drugim, dynamicznym sposobem deklaracji i inicjalizacji tablicy zrealizować opisane poniżej zadania.
- Utworzyć nowy program VI wybierając *BlankVI*.
- Przejść do okna *Block Diagram Panel* i wstawić pętle *For Loop*.
- Narzędziem *Connect Wire* najechać na wyjście *N* wstawionej pętli, nacisnąć prawy przycisk myszy i z menu obiektu wybrać opcję *Create Control.*
- Do pętli wstawić *DBL Numeric Constant*. Następnie narzędziem *Position/Size/Select* kliknąć PPM na wstawiony obiekt i z menu wybrać *Change to Control*.

- Wyjście utworzonego obiektu *Control* połaczyć z prawą pionową ramką pętli *For Loop*. Zostanie utworzone wyjście z pętli, wewnątrz którego jest widoczny symbol tablicy [ ]. Utworzone wyjście jest w trybie *Auto Indexed Tunnel* co oznacza, że na tym wyjściu z pętli, w sposób dynamiczny, jest tworzona tablica. Tryb wyjścia można sprawdzić/ustawić po kliknięciu na wyjście PPM (narzędzie *Position/Size/Select*) i wyborze z menu *Tunnel Mode*.
- Narzędziem *Connect Wire* najechać na utworzone wyjście z tablicy nacisnąć prawy przycisk myszy i z menu obiektu wybrać opcję *Create>>Indicator*.
- Zawartość okna *Front Panel* i *Block Diagram* porównać z rys. 3.1.2.
- Zapisać program, nadając mu nazwę „Tab_002_xx.vi" gdzie xx inicjały twórcy programu.
- Uruchomić program i sprawdzić jego działanie.
- Przed przejściem do następnego zadania z menu okna *Front Panel* wybrać *Edit>>Make Current Values Default* i zapisać program.

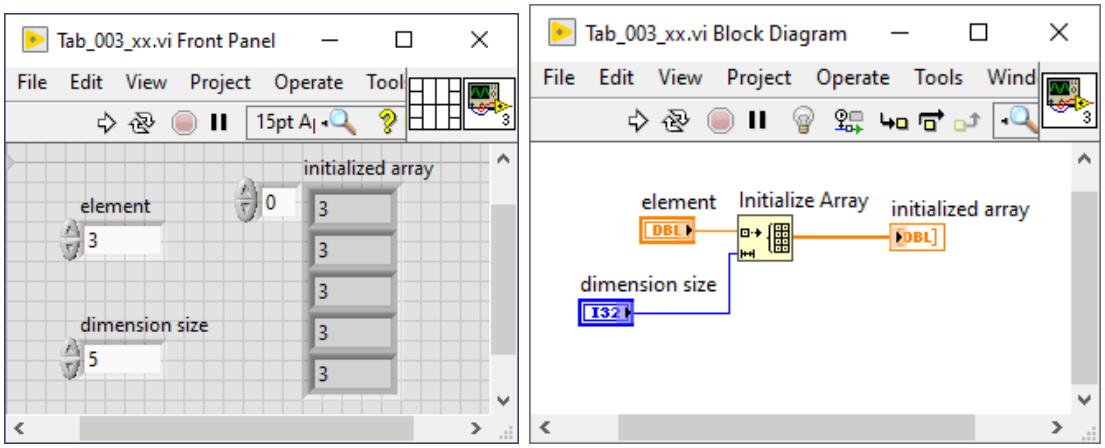


Rys. 3.1.2. Dynamiczna deklaracja i inicjalizacja tablicy. Okna *Front Panel* i *Block Diagram* programu Tab_002_xx.VI.

3.1.3. Deklaracja i inicjalizacja tablicy (sposób 3).

W celu zapoznania się z trzecim sposobem deklaracji i inicjalizacji tablicy zrealizować opisane poniżej zadania.

- Utworzyć nowy program VI wybierając *BlankVI*.
- Zapisać program, nadając mu nazwę „Tab_003_xx.vi" gdzie xx inicjały twórcy programu.
- Przejść do okna *Block Diagram* i z palety *Functions* wstawić *Initialize Array*.
- Narzędziem *Position/Size/Select* kliknąć PPM na wstawiony obiekt i z menu wybrać *Create>>All Controls and Indicators*. Następnie z menu okna *Block Diagram* wybrać *Edit>>Clean Up Diagram*.
- Zawartość okna *Front Panel* i *Block Diagram* porównać z rys. 3.1.3.

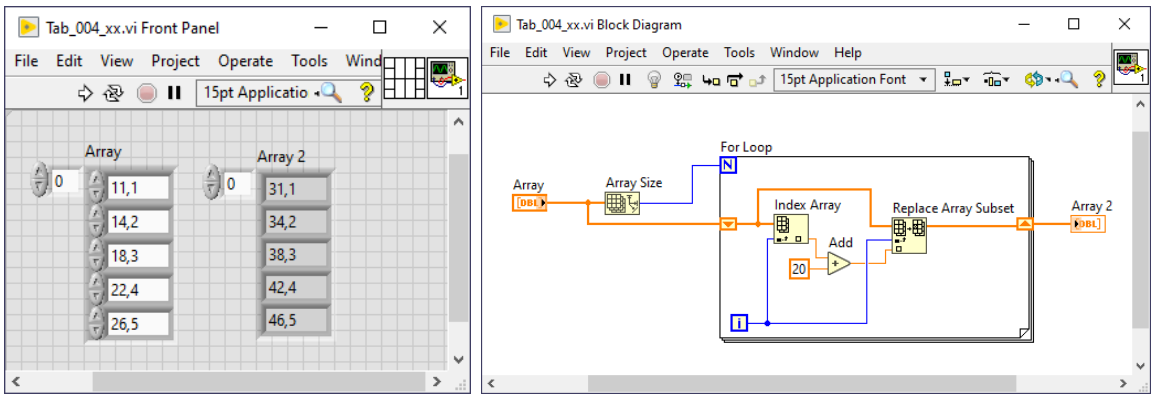


Rys. 3.1.3. Dynamiczna deklaracja i inicjalizacja tablicy. Okna *Front Panel* i *Block Diagram* programu Tab_003_xx.VI.

3.1.4. Zmiana wartości elementów w tablicy.

W celu zapoznania się ze sposobem zmiany wartości elementów w tablicy zrealizować opisane poniżej zadania.
- Utworzyć nowy program VI wybierając *BlankVI*.
- Zapisać program, nadając mu nazwę „Tab_004_xx.vi" gdzie xx inicjały twórcy programu.
- W oknie *Front Panel* utworzyć tablicę. Do tablicy *Array* wpisać wartości początkowe elementów zgodnie z rysunkiem 3.1.4.

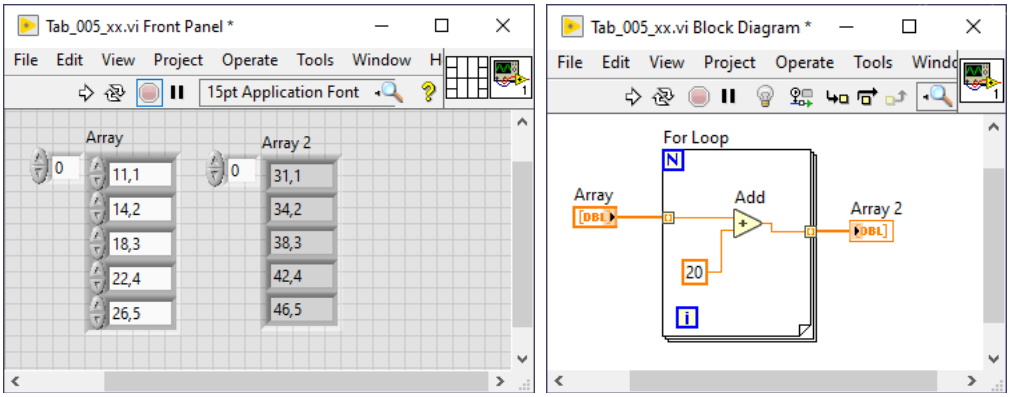
- Przejść do okna *Block Diagram*. Z palety *Functions* wstawić *Array Size*, *Index Array* i *Replace Array Subset*. Korzystając z pomocy kontekstowej LabVIEW zapoznać się ze sposobem działania tych obiektów.
- Do okna *Blok Diagram* dodać pętlę *For Loop* z rejestrem przesuwającym. Obiekty połączyć zgodnie ze schematem przedstawionym na rys. 3.1.4.
- Uruchomić program i sprawdzić efekty jego działania. **Dlaczego w pętli *For Loop* zastosowano rejestr przesuwający?**
- Przed przejściem do następnego zadania z menu okna *Front Panel* wybrać *Edit>>Make Current Values Default* i zapisać program.



Rys. 3.1.4. Zmiana wartości elementów w tablicy. Okna *Front Panel* i *Block Diagram* programu Tab_004_xx.VI.

3.1.5. Zmiana wartości elementów w tablicy sposób 2.

W celu zapoznania się z 2 sposobem zmiany wartości elementów w tablicy zrealizować opisane poniżej zadania.
- Utworzyć nowy program VI wybierając *BlankVI*.
- Zapisać program, nadając mu nazwę „Tab_005_xx.vi" gdzie xx inicjały twórcy programu.
- W oknie *Front Panel* utworzyć tablicę *Array* i wpisać do niej wartości zgodnie z rys. 3.1.5.
- Do okna *Blok Diagram* wstawić pętlę *For Loop* oraz obiekt *Add* (patrz rys. 3.1.5). Tablicę *Array 2* utworzyć w sposób opisany w punkcie 3.1.2.
- W oknie *Block Diagram* wyjście tablicy *Array* połączyć z wejściem obiektu *Add*. Na ramce pętli zostanie narysowany symbol tablicy [] co oznacza, że wejście pętli jest w trybie *Auto-Indexed Tunnel*. Oznacza to, że pętla zostanie uruchomiona tyle razy ile elementów zawiera tablica. Przy pierwszym uruchomieniu pętli do pętli z tablicy *Array* zostanie przesłana wartość elementu tablicy o indeksie 0, przy drugim uruchomieniu pętli wartość elementu o indeksie 1, …, przy ostatnim uruchomieniu pętli wartość ostatniego elementu z tablicy. Tryb wejścia pętli można zmienić poprzez kliknięcie PPM na symbol wejścia [] i wybór z menu rozwijanego *Disable Indexing*.
- Uruchomić program i sprawdzić efekty jego działania. **Czy na wejściu i wyjściu pętli *Foor Loop* dane znajdują się w tej samej tablicy?**
- Przed przejściem do następnego zadania z menu okna *Front Panel* wybrać *Edit>>Make Current Values Default* i zapisać program.

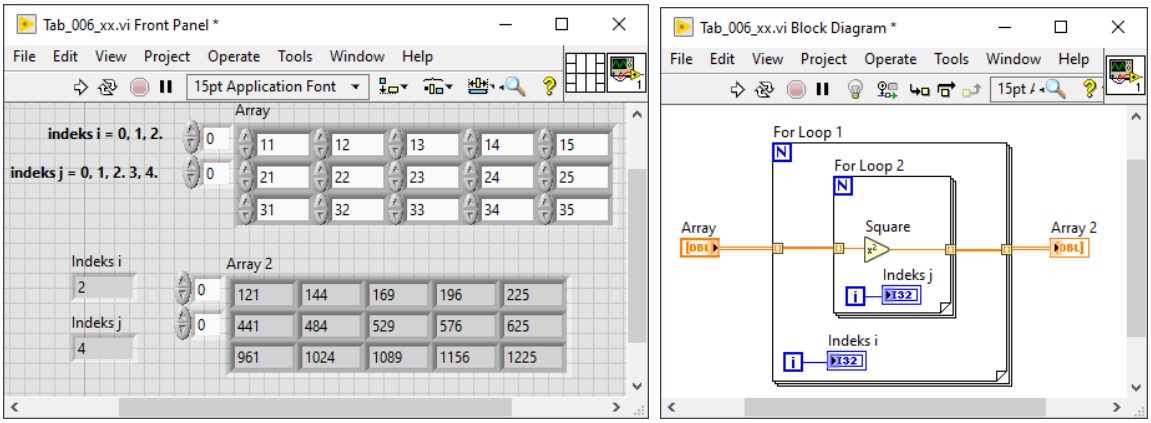


Rys. 3.1.5. Zmiana wartości elementów w tablicy. Okna *Front Panel* i *Block Diagram* programu Tab_005_xx.VI.

3.1.6. Zmiana wartości elementów w tablicy 2D.

W celu zapoznania się ze sposobem zmiany wartości elementów w tablicy 2D zrealizować opisane poniżej zadania.

- Utworzyć nowy program VI wybierając *BlankVI*.
- Zapisać program, nadając mu nazwę „Tab_006_xx.vi" gdzie xx inicjały twórcy programu.
- W oknie *Front Panel* utworzyć tablicę 2D (*Array*[*i*,*j*]) i wpisać do niej wartości zgodnie z rys. 3.1.6. Pierwszy indeks odnosi się do wierszy (i = 0, 1, 3) drugi indeks odnosi się do kolumn (j = 0, 1, 2, 3, 4).
- Do okna *Blok Diagram* wstawić 2 pętlę *For Loop* oraz obiekt *Square*. Obiekty w oknie *Block Diagram* połączyć zgodnie z rys. 3.1.6.
- Uruchomić program i sprawdzić efekty jego działania. **Ile razy jest wywoływana pętla *For Loop 1*? Jaki obiekt jest przekazywany na wejście pętli *For Loop 2*? Ile razy jest wywoływana pętla *For Loop 2*? Jaki obiekt jest tworzony na wyjściu pętli *For Loop 1*? Jaki obiekt jest tworzony na wyjściu pętli *For Loop 2*?**
- Przed przejściem do następnego zadania z menu okna *Front Panel* wybrać *Edit>>Make Current Values Default* i zapisać program.



Rys. 3.1.6. Zmiana wartości elementów w tablicy. Okna *Front Panel* i *Block Diagram* programu Tab_006_xx.VI.

3.2. Klastry.

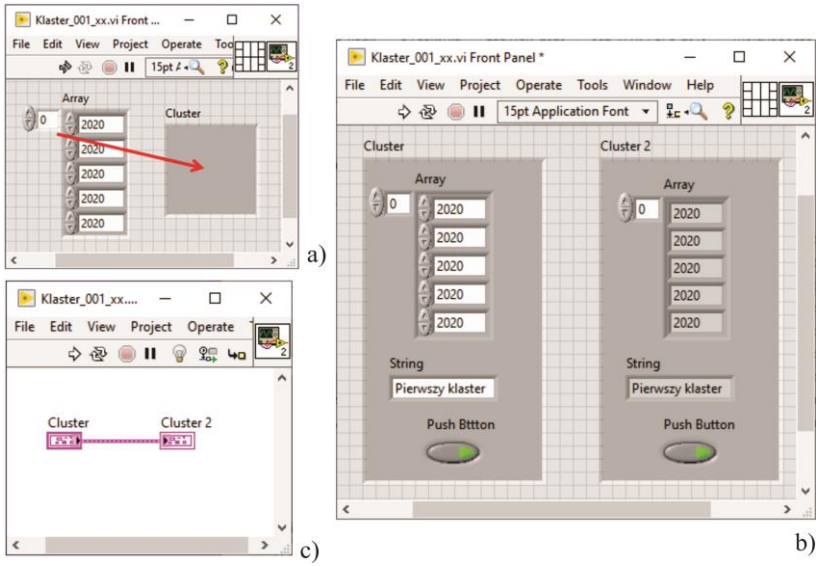
Klastry w LabVIEW służą do grupowania danych różnych typów.

3.2.1. Deklaracja i inicjalizacja klastra.

W celu zapoznania się ze sposobem deklaracji i inicjalizacji klastra zrealizować opisane poniżej zadania.

- Utworzyć nowy program VI wybierając *BlankVI*.
- Zapisać program, nadając mu nazwę „Klaster_001_xx.vi" gdzie xx inicjały twórcy programu.
- W oknie *Front Panel* utworzyć tablicę 1D zawierającą dane typu *I 16*. Zainicjalizować 5 elementów tablicy wpisując do każdego elementu wybraną wartość typu *I 16* (np.: 2020).

- Do okna *Front Panel* z palety *Controls* wstawić obiekt *Cluster*. Utworzoną wcześniej tablicę przeciągnąć do obiektu *Cluster* (patrz rys. 3.2.1).
- Z palety *Controls* do obiektu *Cluster* wstawić obiekty *String Contol* i *Push Button*. Nadać wartości początkowe wstawionym obiektom.
- Przejść do okna *Block Diagam* kliknąć na wyjście obiektu *Cluster* i z menu rozwijanego wybrać *Create Indicator*.
- Uruchomić program i sprawdzić efekty jego działania. Zawartość okien *Front Panel* i *Block Diagram* porównać z rys. 3.2.1.
- Przed przejściem do następnego zadania z menu okna *Front Panel* wybrać *Edit>>Make Current Values Default* i zapisać program.

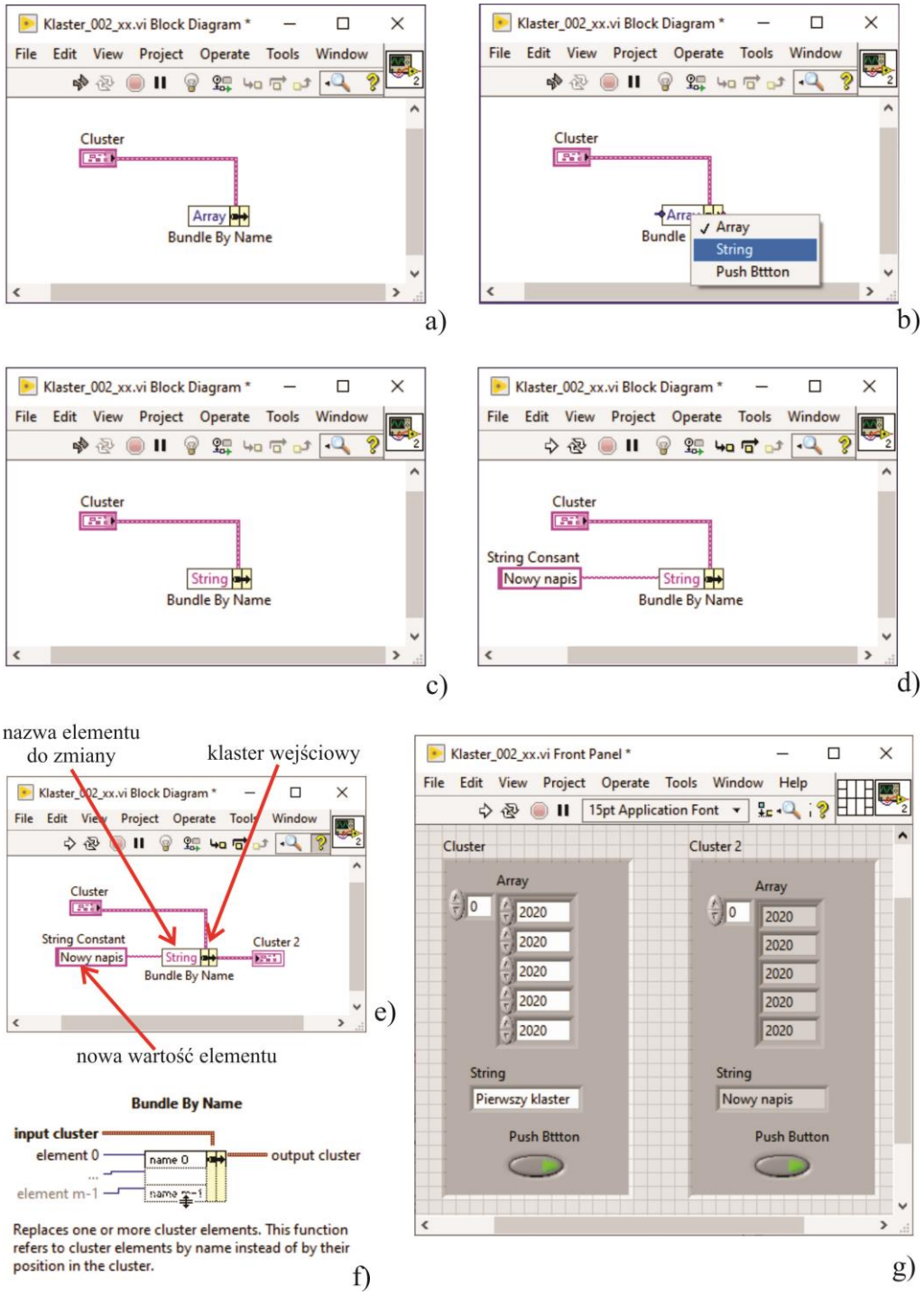


Rys. 3.2.1. Deklaracja i inicjalizacja klastra: a) wstawianie tablicy do klastra, b) i c) Okna *Front Panel* i *Block Diagram* programu Klaster_001_xx.VI.

3.2.2. Zmiana wartości elementu w klastrze.

W celu zapoznania się ze sposobem zmiana wartości elementu w klastrze zrealizować opisane poniżej zadania.

- Utworzyć nowy program VI wybierając *BlankVI*.
- Program VI z poprzedniego punktu zapisać pod nazwą „Klaster_002_xx.vi" gdzie xx inicjały twórcy programu.
- Do okna *Block Diagram* wstawić obiekt *Bundle By Name*. Do wejścia *input claster* wstawionego obiektu podłączyć strukturę *Cluster* (rys. 3.2.2a). Obiekt *Bundle By Name* domyślnie zawiera 1 wejście odpowiadające pierwszemu elementowi w klastrze w tym przypadku *Array*.
- Kliknąć na *Array* (narzędzie *Operate Value*) i z menu wybrać drugi element z klastra: *String* (rys. 3.2.2b i 3.2.2c).
- Z palety *Funtions* wstawić obiekt *Sting Constat*. Zawartość wstawionego obiekt zmodyfikować zgodnie z rys. 3.2.2d. Wyjście *Sting Constat* połączyć z wejściem *String* obiektu *Bundle By Name*.
- Program w oknie *Blok Diagram* zmodyfikować zgodnie z rys. 3.2.2e. Korzystając z rysunków 3.2.2e i 3.2.2f zapoznać się ze sposobem działania obiektu *Bundle By Name.*
- Uruchomić program i efekty jego działania porównać z rys. 3.2.2g.
- Przed przejściem do następnego zadania z menu okna *Front Panel* wybrać *Edit>>Make Current Values Default* i zapisać program.



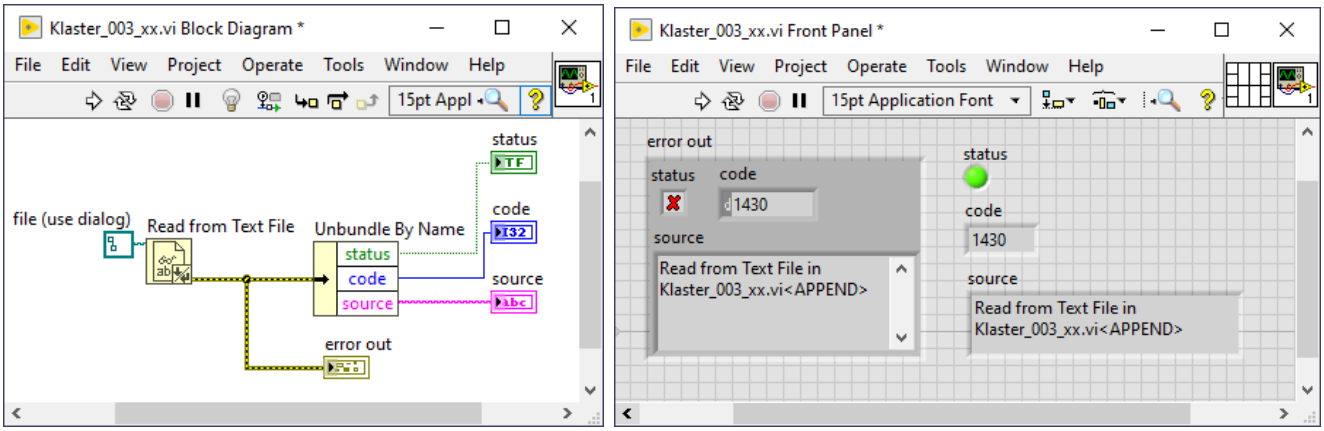
Rys. 3.2.2. Zmiana wartości elementu w klastrze program Klaster_002_xx.VI.

3.2.3. Odczyt wszystkich elementów z klastra.

W celu zapoznania się ze sposobem odczytu elementów z klastra zrealizować opisane poniżej zadania.

- Utworzyć nowy program VI wybierając *BlankVI*.
- Zapisać program, nadając mu nazwę „Klaster_003_xx.vi" gdzie xx inicjały twórcy programu.
- Do okna *Block Diagram* wstawić obiekt *Read from Text File*. Kliknąć PPM na wejście *file (use dialog)* i z menu rozwijanego wybrać *Create>>Constant*.
- Do okna *Block Diagram* wstawić obiekt *Unbundle By Name*. Wejście wstawionego obiektu połączyć z wyjściem *error out* obiektu *Read from Text File*. Na tym wyjściu po uruchomieniu obiektu *Read from Text File* pojawia się klaster złożony z trzech elementów zawierający informacje o błędach.

- Obiekt *Unbundle By Name* rozciągnąć tak, aby zawierał trzy wyjścia. Używając narzędzia *Operate Value* zmienić nazwy wyjść zgodnie z rys. 3.2.3.
- Program w oknie *Blok Diagram* zmodyfikować zgodnie z rys. 3.2.3a. Obiekty *error out*, *status*, *code* i *source* utworzyć wybierając odpowiednie wyjście obiektu *Unbudle By Name* i następnie z menu rozwijanego *Create>>Indicator*.
- Uruchomić program i efekty jego działania porównać z rys. 3.2.3 (okno *Front Panel*).
- Ponieważ plik do odczytu nie został odnaleziony na wyjściu *error out* obiektu *Read from Text File* pojawi się informacja o błędzie. Obiekt *error out* prezentuje zawartość całego klastra. Obiekty *status*, *code* i *source* prezentują zawartość poszczególnych elementów klastra.
- Przed przejściem do następnego zadania z menu okna *Front Panel* wybrać *Edit>>Make Current Values Default* i zapisać program.



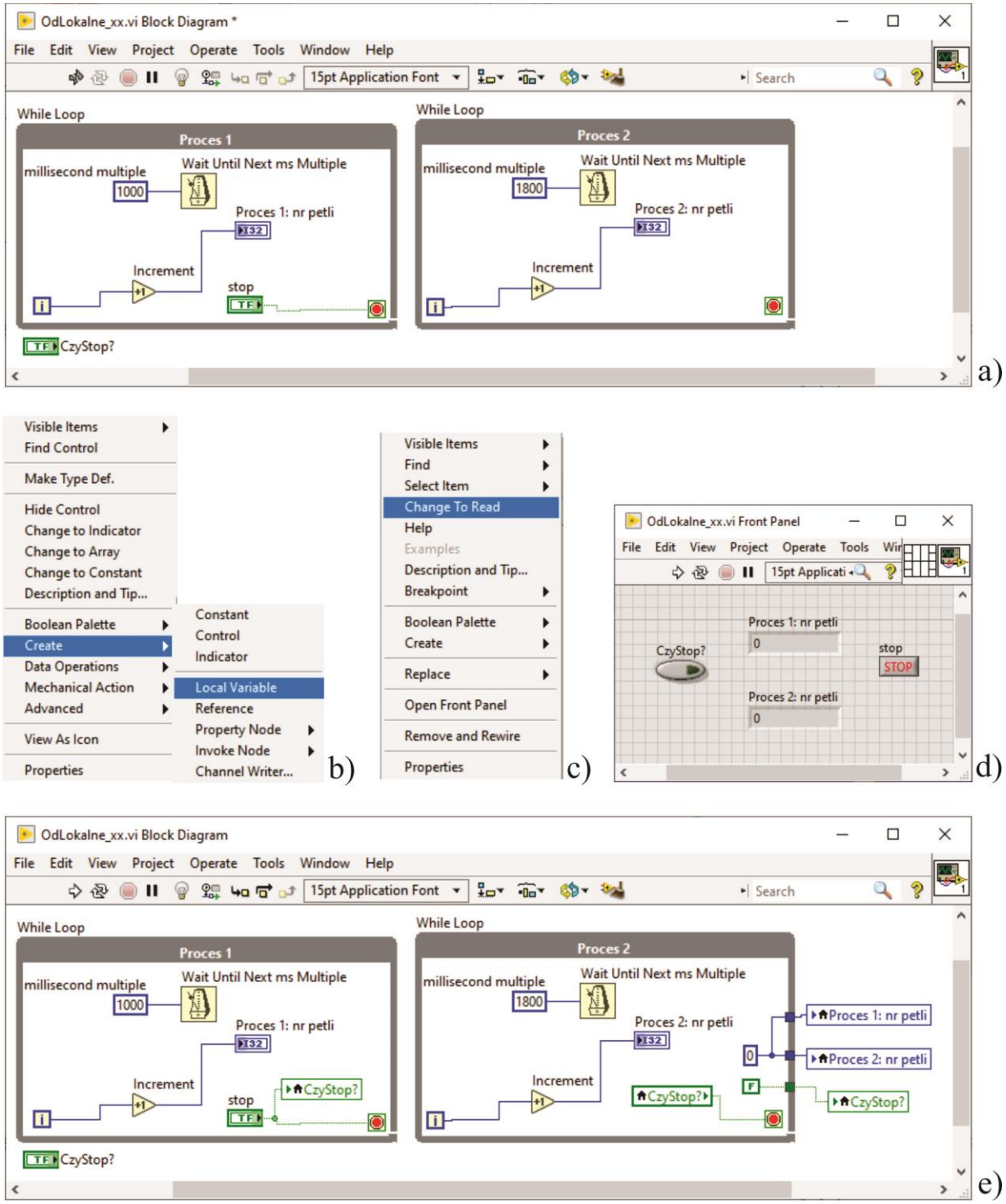
Rys. 3.2.3. Odczyt wszystkich elementów z klastra: okna *Front Panel* i *Block Diagram* programu Klaster_003_xx.VI.

3.3. Lokalne odwołania do zmiennych, zmiana właściwości obiektów.

3.3.1. Lokalne odwołania do zmiennych.

Zmienne związane z każdym obiektami *Control* lub *Indicator* są dostępne w zakresie całego program VI i można do nich tworzyć odwołania lokalne w dowolnym miejscu programu.
W celu zapoznania się ze sposobem tworzenia odwołań lokalnych zrealizować opisane poniżej zadania.

- Utworzyć nowy program VI wybierając *BlankVI*.
- Zapisać program, nadając mu nazwę „Od_Lokal_xx.vi" gdzie xx inicjały twórcy programu.
- Do okna *Front Panel* z palety *Controls* wstawić obiekt *Push Button* i zmienić jego nazwę na *CzyStop?*. Następnie wstawić dwa obiekty *Numeric Control* oraz obiekt *Stop Button*. Nazwy obiektów *Numeric Control* zmienić na *Proces 1: nr petli* oraz *Proces 2: nr petli*. Kliknąć PPM na obiekt *Proces 1: nr petli* i z menu obiektu wybrać *Change to Indicator*. Czynności powtórzyć dla obiektu *Proces 2: nr petli*. Zmienne związane ze wstawionymi obiektami są widoczne w obrębie całego programu VI i są identyfikowane przez nazwy obiektów (w tym przypadku *CzyStop?*, *Proces 1: nr petli*, *Proces 2: nr petli* oraz *Stop Button*).
- Do okna *Block Diagram* z palety *Functions* wstawić dwie pętle *While Loop*.
- Do każdej pętli wstawić obiekty *Wait Until Next ms Multiple*, *Numeric Constant* i *Increment*. Obiekty rozmieścić i połączyć zgodnie z rys. 3.3.1a. Program zawiera dwa procesy realizowane przez dwie niezależne pętle. Naciśnięcie przycisku *stop* spowoduje przerywanie działania pierwszego procesu. Informacja o naciśnięciu przycisku *stop* zastanie przekazana do drugiej pętli za pomocą zmiennej *CzyStop?*.
- W celu utworzenia lokalnego odwołania do zmiennej *CzyStop?* w oknie *Block Diagram* kliknąć PPM na obiekt *CzyStop?* i z menu wybrać *Create>>Local Variable* (patrz rys. 3.3.1b). Utworzony obiekt wstawić do pętli *Proces 2*. Po wstawieniu obiekt jest domyślnie ustawiony w trybie do zapisu. Kliknąć PPM na obiekt (narzędzie *Position/Size/Select*) i zmienić jego tryb na tryb do odczytu poprzez wybór z menu *Change To Read*. (rys 3.3.1c).



Rys. 3.3.1. Lokalne odwołania do zmiennych: okna *Front Panel* i *Block Diagram* programu Od_Lokal_xx.VI.

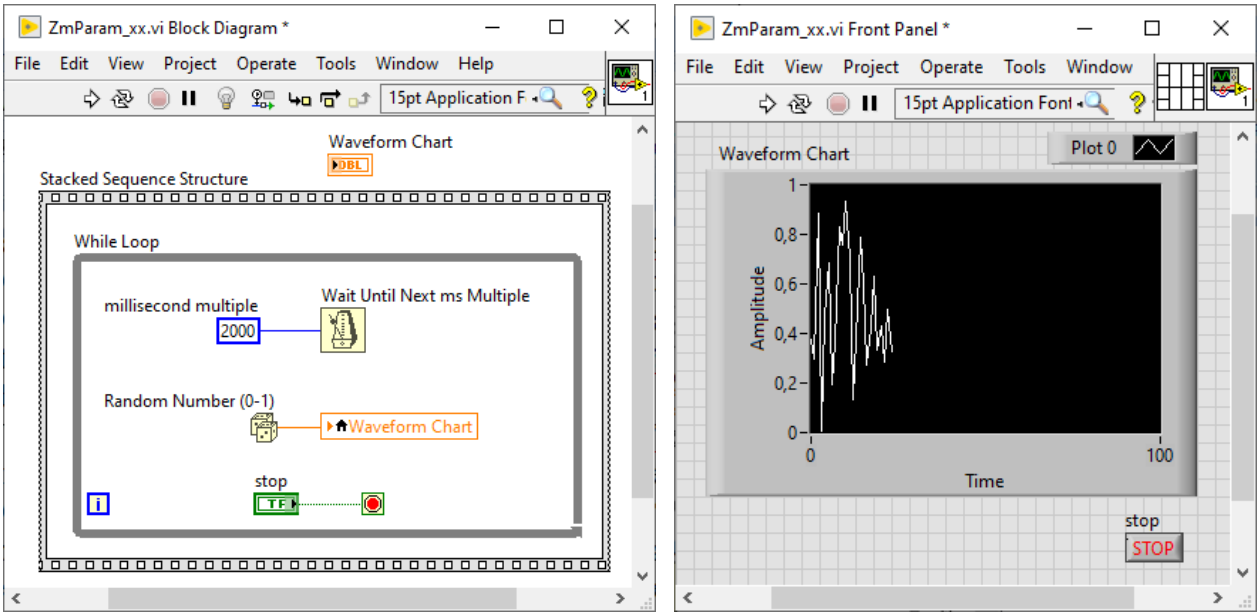
- W celu zapoznania się z drugim sposobem tworzenia lokalnego odwołania do zmiennej z palety *Functions* do pętli *Proces 1* wstawić obiekt *Local Variable*. Kliknąć na wstawiony obiekt (narzędzie *Operate Value*) i z menu wybrać zmienną *CzyStop?*. Uwaga. W menu widoczne są wszystkie zmienne z całego pogram VI.
- Utworzyć lokalne odwołania do zmiennych *Proces 1: nr petli*, *Proces 2: nr petli* i *CzyStop?*. Obiekty w oknie *Block Diagram* połączyć zgodnie z rys. 3.3.1e. Obiekty w oknie Front Panel rozmieścić zgodnie z rys. 3.3.1d.
- Uruchomić program i sprawdzić efekty jego działania. **W jakim celu zastosowano lokalne odwołania do zmiennych (*Proces 1: nr petli*, *Proces 2: nr petli* i *CzyStop*) uruchamiane po wyjściu z pętli *Proces 2*?** Zawartość okien *Front Panel* i *Block Diagram* porównać z rys. 3.2.1.

- Przed przejściem do następnego zadania z menu okna *Front Panel* wybrać *Edit>>Make Current Values Default* i zapisać program.

3.3.2. Zmiana właściwości obiektów.

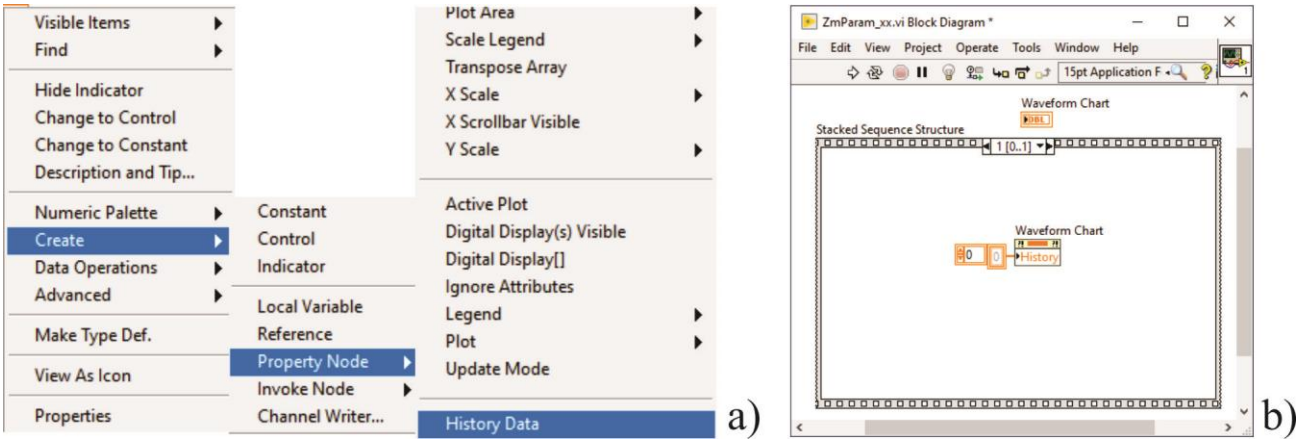
Każdy obiekt *Control* lub *Indicator* znajdujący się w programie VI posiada określoną liczbę parametrów, za pomocą których można modyfikować jego właściwości takie jak: rozmiar, zawartość, zachowanie, aktywność itp. Dostęp do tych parametrów jest możliwy w dowolnym miejscu programu VI z wykorzystaniem obiektu *Property Node*.
W celu zapoznania się ze wykorzystaniem obiektu *Property Node*.

- Utworzyć nowy program VI wybierając *BlankVI*.
- Zapisać program, nadając mu nazwę „ZmParam_xx.vi" gdzie xx inicjały twórcy programu.
- Do okna *Front Panel* z palety *Controls* wstawić obiekt *Waveform Chart*.
- Do okna *Block Diagram* z palety *Functions* wstawić obiekt *Stacked Seqence Structure* do którego wstawić pętlę *While Loop*. Do pęli *While Loop* wstawić lokalne odwołanie do zmiennej *Waveform Chart*. Pozostałe obiekty wstawić i połączyć zgodnie z rys. 3.3.2.
- Uruchomić program 2 lub 3 razy. W obiekcie *Waveform Chart* (okno *Front Panel*) po kolejnym uruchomieni programu widoczne są dane z poprzedniego uruchomienia programu. Do programu zastanie dodany kod czyszczący zawartość obiektu *Waveform Chart* przed wyjściem z programu.



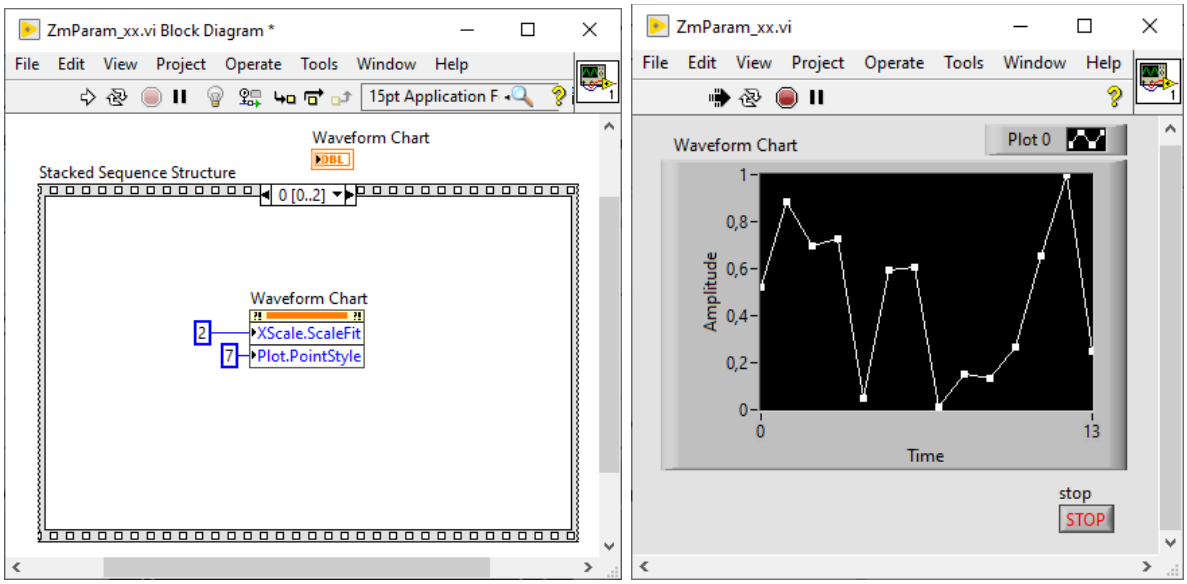
Rys. 3.3.2. Okna *Block Diagram* i *Front Panel* programu ZmParam_xx.VI.

- Kliknąć PPM na obiekt *Stacked Sequence Structure* i z menu *Add Frame After*.
- W celu utworzenia obiektu *Property Node* czyszczącego zawartość *Waveform Chart* w oknie *Block Diagram* kliknąć PPM na *Waveform Chart* i z menu wybrać *Create>>Property Node>>History Data* (patrz rys. 3.3.3).
- PPM kliknąć na wstawiony obiekt i z menu wybrać *Change To Write*. Kursorem (narzędzie *Connect Wire*) najechać na wejście *History* kliknąć PPM i wybrać *Create>>Constant*. Zawartość okna *Block Diagram* porównać z rys. 3.3.3.
- Zapisać, uruchomić program i zaobserwować efekty jego działania.



Rys. 3.3.3. Użycie obiekt *Property Node* w programie ZmParam_xx.VI.

- W celu dodania na początku programu kodu modyfikującego sposób wyświetlania wykresu w obiekcie *Waveform Chart* przed ramką 0 obiektu *Stacked Sequence Structure* wstawić nową ramkę.
- W sposób opisany powyżej dla obiektu *Waveform Chart* utworzyć obiekt *Property Node* wybierając *Create>>Property Node>>X Scale>>Scale Fit.* Kliknąć na wstawiony obiekt i z menu wybrać *Change All To Write.* Następnie rozciągnąć obiekt w celu dodania drugiego wejścia.
- Kliknąć (narzędzie *Operate Value*) na dodane wejście i z menu wybrać *Plot>>Point Style.* Zawartość ramki 0 obiektu *Stacked Sequence Structure* zmodyfikować zgodnie z rys. 3.3.4.
- Uruchomić program i sprawdzić efekty jego działania. Zawartość okna *Front Panel* porównać z rys. 3.3.4.
- Przed przejściem do następnego zadania z menu okna *Front Panel* wybrać *Edit>>Make Current Values Default* i zapisać program.



Rys. 3.3.4. Użycie drugiego obiekt *Property Node* w programie ZmParam_xx.VI.

4. Zadania sprawdzające do samodzielnej realizacji

W celu utrwalenia i sprawdzenia zrealizować następujące zadania.

- Napisać program porównujący działanie obiektów *Bundle* i *Bundle By Name* (nazwa programu: zadSpr_001_xx.VI).
- Napisać program porównujący działanie obiektów *Unbundle* i *Unbundle By Name* (nazwa programu: zadSpr_002_xx.VI).

- W programie z p. 3.3.2 dodać kolejne obiekty *Property Node* modyfikujące wybrane właściwości obiektu *Waveform Chart* (nazwa programu: zadSpr_003_xx.VI).

5. Pytania sprawdzające

- 5.1. Dlaczego w pętli *For Loop* zastosowano rejestr przesuwający? (pytanie do p. 3.1.4)
- 5.2. Czy na wejściu i wyjściu pętli *Foor Loop* dane znajdują się w tej samej tablicy? (pytanie do p. 3.1.5)
- 5.3. Ile razy jest wywoływana pętla *For Loop 1*? (pytanie do p. 3.1.6)
- 5.4. Jaki obiekt jest przekazywany na wejście pętli *For Loop 2*? (pytanie do p. 3.1.6)
- 5.5. Ile razy jest wywoływana pętla *For Loop 2*? (pytanie do p. 3.1.6)
- 5.6. Jaki obiekt jest tworzony na wyjściu pętli *For Loop 1*? (pytanie do p. 3.1.6)
- 5.7.Jaki obiekt jest tworzony na wyjściu pętli *For Loop 2*? (pytanie do p. 3.1.6)
- 5.8. W jakim celu zastosowano lokalne odwołania do zmiennych (*Proces 1: nr petli*, *Proces 2: nr petli* i *CzyStop*) uruchamiane po wyjściu z pętli *Proces 2*? (pytanie do p. 3.3.1)

6. Zawartość sprawozdania

Sprawozdanie z ćwiczenia powinno zawierać:
- krótki opis nowych poznanych w trakcie ćwiczenia obiektów z palety *Functions*,
- zrzuty ekranów uruchomionych programów,
- uruchomione programy w postaci plików (nazwy plików zgodne z instrukcją),
- ponumerowane odpowiedzi na pytania zadane w instrukcji (Odpowiedzieć tylko na pytania z punktu 5. Odpowiedzi umieścić w sprawozdaniu przed wnioskami.),
- uwagi i wnioski.