INSTYTUT TELEKOMUNIKACJI MULTIMEDIALNEJ

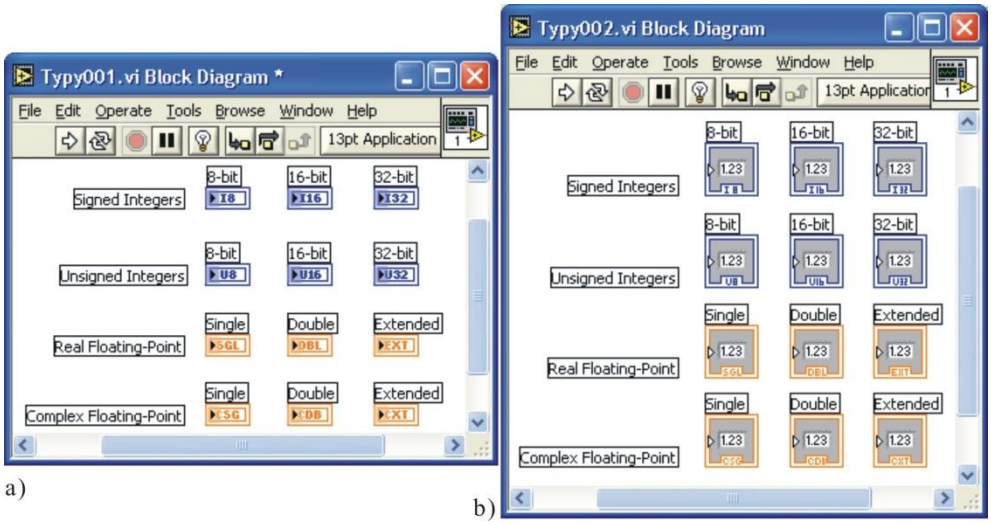
LABORATORIUM KOMPUTEROWYCH SYSTEMÓW POMIAROWYCH

Instrukcja do ćwiczenia:

Podstawy programowania w języku G

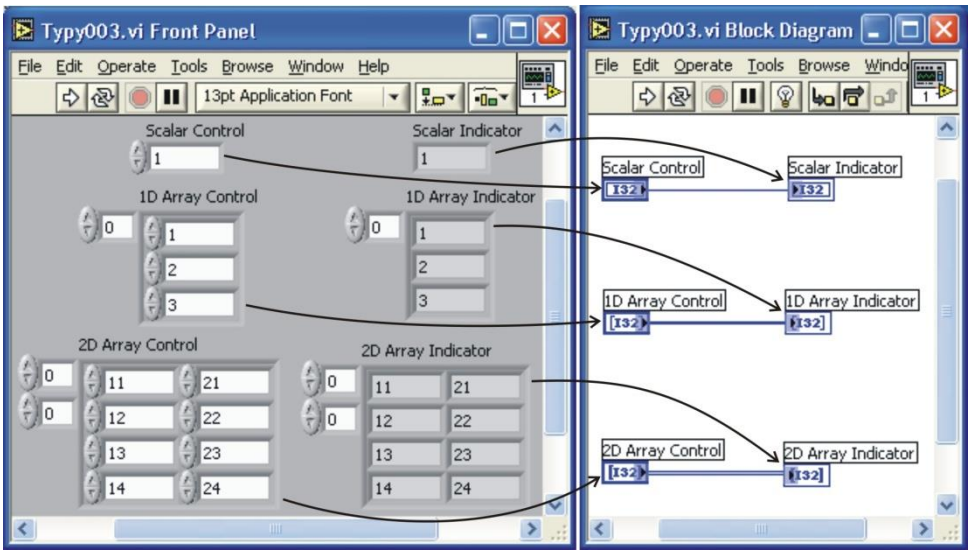
1. Wprowadzenie

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.



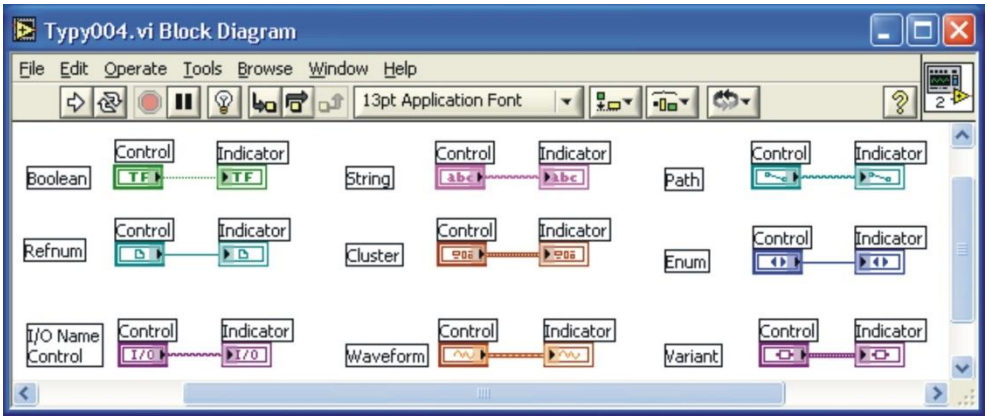
Rys.1. Podstawowe typy zmiennych numerycznych w języku G

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

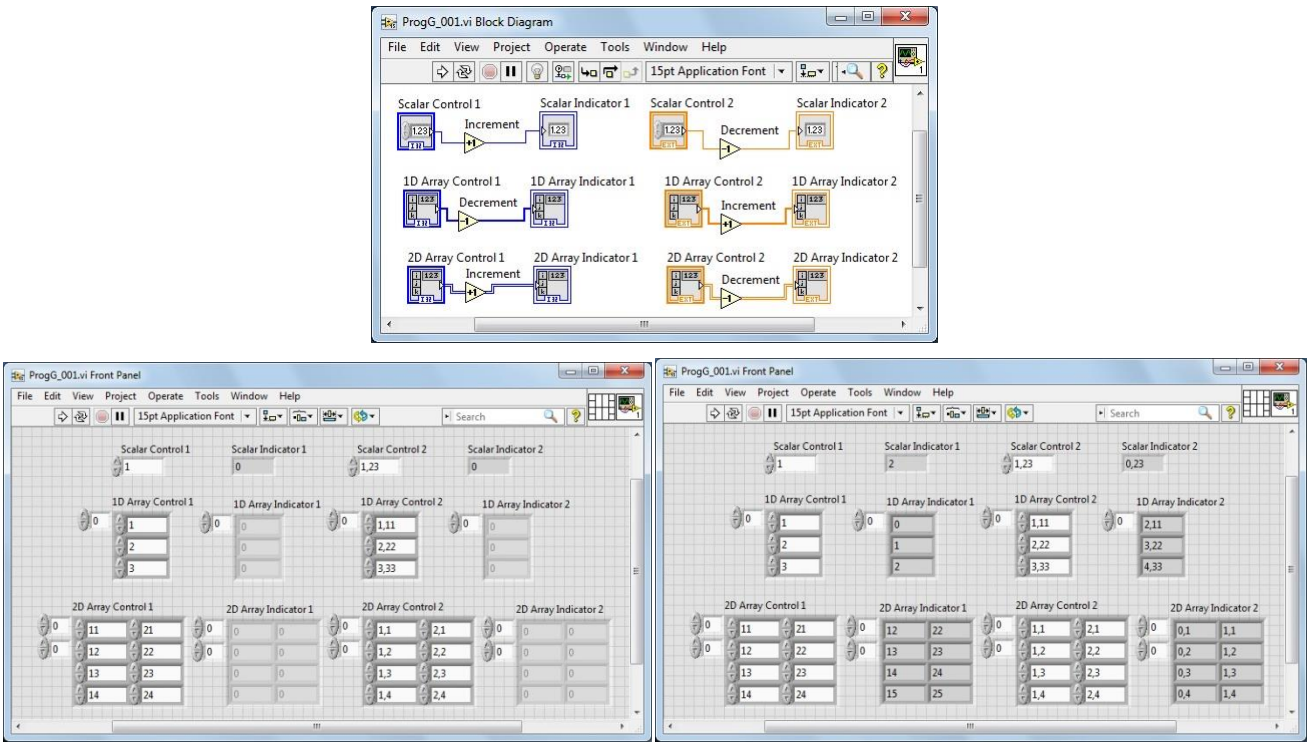
Celem ćwiczenia jest zapoznanie się ze sposobem programowania z wykorzystaniem języka G. W trakcie ćwiczenia studenci zapoznają się z:
– podstawowymi typami zmiennych w języku G,
– podstawowymi strukturami programowymi języka G,
– sposobem tworzenie podprogramów w języku G.

3. Przebieg ćwiczenia

3.1. Tworzenie zmiennych numerycznych w języku G

Uwaga! W palecie *Tools* programu LabView możliwy jest wybór opcji *Automatic Tool Selection*. Opcja ta powoduje automatyczny wybór narzędzi podczas tworzenia programu. Zdaniem autorów na etapie nauki programowania, w celu zrozumienia efektów działania poszczególnych narzędzi z palety *Tools* opcja *Automatic Tool Selection* powinna być wyłączona. Poniższy program ćwiczenia został napisany dla przypadku, kiedy opcja ta została wyłączona. Jeżeli osoba realizująca pogram ćwiczenia zna sposób działania narzędzi z palety *Tools* opcja *Automatic Tool Selection* może być włączona.

W celu zapoznania się ze sposobem tworzenia różnych typów zmiennych numerycznych w języku G zrealizować następujące zadania:
- uruchomić LabView i utworzyć nowy program VI wybierając projekt *BlankVI*,
- z palety *Tools* wybrać narzędzie *Position/Size/Select*,
- do okna *Front Panel* wstawić obiekt *Controls » Modern » Numerics » Numeric Control*,
- kursorem najechać na wstawiony obiekt i kliknąć prawym przyciskiem myszy, pojawi się menu obiektu,
- z menu wstawionego obiektu wybrać opcję *Representation » I32*, spowoduje to zmianę typu zmiennej,



Rys. 4. Okna *Block Diagram*, *Front Panel* (przed uruchomieniem) oraz *Front Panel* (po uruchomieniu) programu „ProgG_001.vi”.

- przejść do okna *Block Diagram*, z palety *Tools* wybrać przycisk *Connect Wire* najechać na wyjście wstawionego obiektu, kliknąć prawym przyciskiem myszy i z menu obiektu wybrać opcję *Create » Indicator*

czynności powtórzyć dla drugiego obiektu *Controls » Modern » Numerics » Numeric Control* wybierając typ zmiennej *EXT*,
- przejść do okna *Front Panel* i wstawić obiekt *Controls » Modern » Array,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*, z palety *Tools* wybrać przycisk *Connect Wire* najechać na wyjście wstawionego obiekty, nacisnąć prawy przycisk myszy i z menu obiektu wybrać opcję *Create » Indicator*,
- czynności powtórzyć dla drugiej pary obiektów *Controls » All Controls » Aray&Cluster » Array* i *Controls » All Controls » Numeric » Numeric Control* wybierając typ zmiennej *EXT*,
- w opisany powyżej sposób utworzyć dwie tablice 2D pierwszą ze zmiennymi typu *I32* drugą ze zmiennymi typu *EXT*, do zamiany tablicy 1D na tablicę 2D wykorzystać opcję *Add Dimension* z menu obiektu *Controls » Modern » Aray&Cluster » Array*,
- zmodyfikować nazwy i zawartość obiektów zgodnie z rys. 4 (okno *Front Panel* przed uruchomieniem),
- przejść do okna *Block Diagram*, pomiędzy wszystkie obiekty *Control* i *Indicator* wstawić obiekty *Functions » Programming » Numeric » Increment* i *Functions » Programming » Numeric » Decrement* zgodnie ze schematem przedstawionym na rys. 4,
- w oknie *Front Panel* wybrać opcję *Edit* » *Make Current Values Default* a następnie *File* » *Save* i zapisać program, nadając mu nazwę „ProgG_001_xx.vi" (gdzie xx inicjały twórcy programu).
- uruchomić program i sprawdzić jego działanie (okno *Front Panel* po uruchomieniu). **Jak obiekty *Increment* i *Decremrement* modyfikują tablice 1D i 2D?**

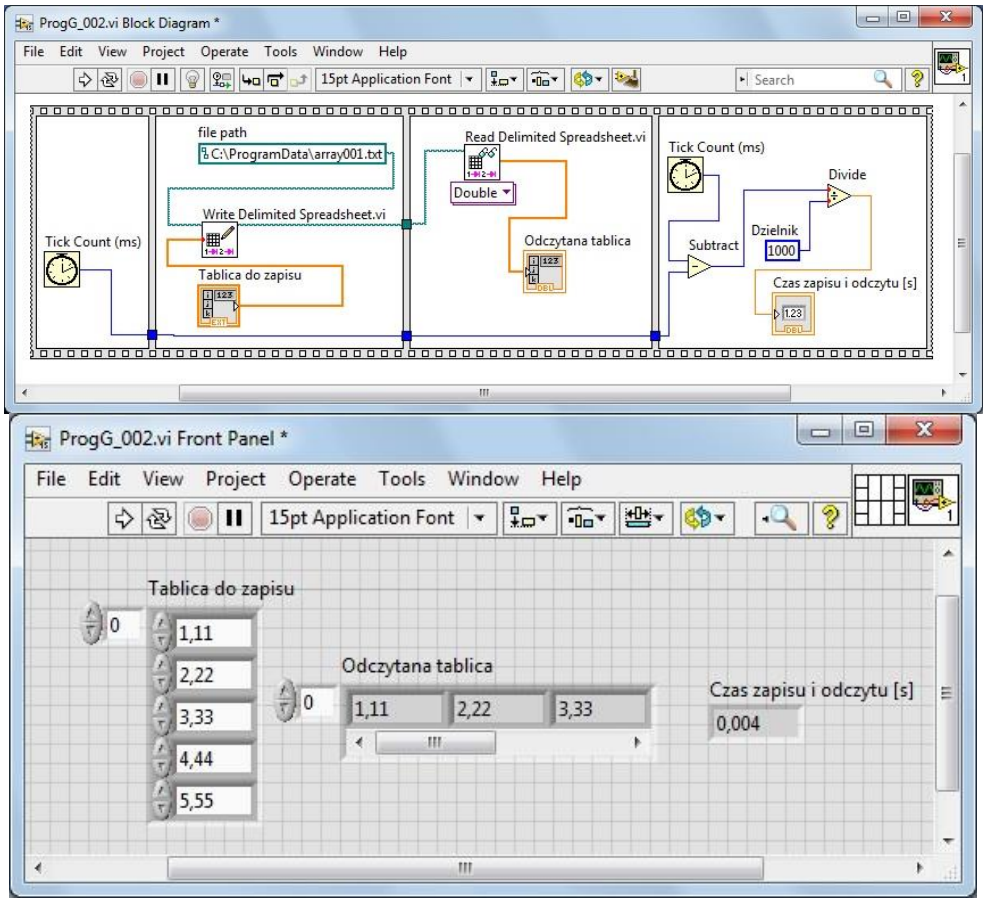
3.2 Obiekty sterujące w języku programowania G

Jeżeli w programie języku G nie użyto obiektów sterujących poszczególne obiekty wykonywane są zgodnie z przepływem danych liniami łączącymi poszczególne obiekty. Omówione poniżej obiekty sterujące służą do zarządzania kolejnością wykonywania obiektów.

3.2.1. Obiekt sterujący *Flat Sequence*

W celu zapoznania się ze sposobem wykorzystania obiektu sterującego *Flat Sequence* zostanie napisany program mierzący czas zapisu i odczytu danych w pliku tekstowym:
- utworzyć nowy program VI wybierając projekt *BlankVI*,
- do okna *Block Diagram* wstawić obiekt *Functions » Programmingl » Structures » Flat Sequence*,
- z menu obiektu *Sequence* wybrać trzykrotnie *Add » Frame After*,
- z palety *Functions* wybrać i umieścić w oknie *Block Diagram* następujące obiekty
  Functions » Programming » Timing » Tick Count (ms),
  Functions » Programming » File I/O » File Constants » Path Constant,
  Functions » Programming » File I/O » Write Delimited Spreadsheet,
  Functions » Programming » File I/O » Read Delimited Spreadsheet,
  Functions » Mathematics » Numeric » Subtract,
  Functions » Mathematics » Numeric » Divide,
- naniesione obiekty połączyć zgodnie ze schematem przedstawionym na rys. 5 (okno *Block Diagram*),
- obiekt *Dzielnik* utworzyć wybierając z palety *Tools* narzędzie *Connect Wire*, następnie najechać kursorem na wejście *y* obiektu *Divide* kliknąć prawym przyciskiem myszy i wybrać z menu obiektu opcję *Create » Constans*,
- w podobny sposób utworzyć obiekty *Mnożnik* wybierając z menu obiektu opcję *Create » Costant*,
- obiekt *Czas zapisu i odczytu [s]* utworzyć wybierając po najechaniu kursorem na wyjście *x/y* obiektu *Divide* z jego menu opcję *Create » Indicator*,
- tablicę *Tablica do zapisu* utworzyć zgodnie z zasadami opisanymi w punkcie 3.1 i połączyć z wejściem *1D Array* obiektu *Write Delimited Spreadsheet*,
- obiekt *Odczytana tablica* utworzyć wybierając z palety *Tools* narzędzie *Connect Wire*, następnie najechać kursorem na wyjście *first row* obiektu *Read Delimited Spreadsheet* kliknąć prawym przyciskiem myszy i wybrać z menu obiektu opcję *Create » Indicator*,
- zmodyfikować nazwy i zawartości obiektów zgodnie z rys. 5 (okno *Front Panel* i okno *Block Diagram*),

- w oknie *Front Panel* wybrać opcję *Edit » Make Current Values Default* a następnie zapisać program nadając mu nazwę „ProgG_002.vi_xx” (gdzie xx inicjały twórcy programu).
- uruchomić program i sprawdzić zawartość pliku „*array001.txt*”.
- zmienić nazwę tego pliku ponownie dwukrotnie uruchomić program. **Z czego wynika różnica w zmierzonych czasach dla pierwszego i drugiego uruchomienia programu? Jak działa obiekt sterujący *Flat Seqence*?**

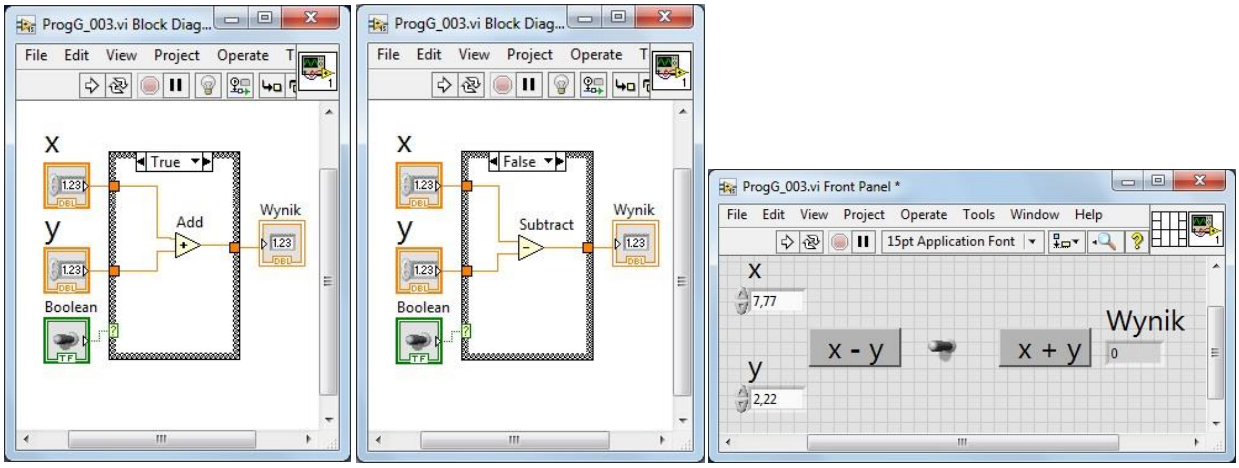


Rys. 5. Okna *Block Diagram, Front Panel* „ProgG_002.vi”

3.2.2. Obiekt sterujący *Case structure*

W celu zapoznania się ze sposobem wykorzystania obiektu sterującego *Case Structure* zrealizować następujące zadania:

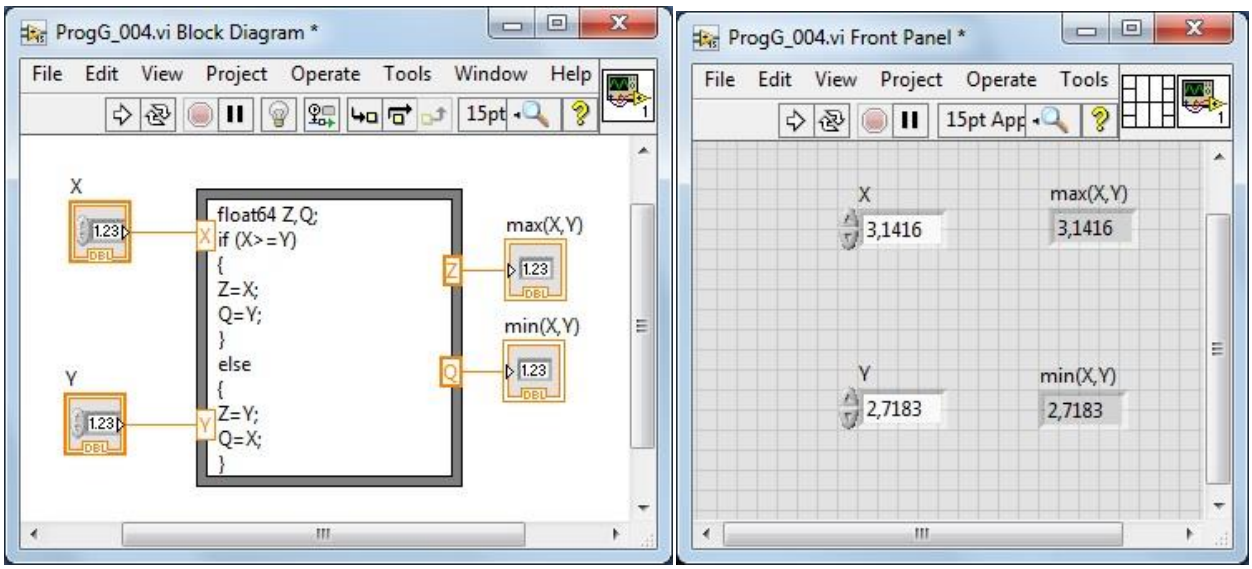
- utworzyć nowy program VI wybierając projekt *BlankVI*,
- do okna *Block Diagram* wstawić obiekt *Functions » Programming » Structures » Case Structure*,
- do ramki *True* obiektu *Case Structure* wstawić obiekt *Functions » Mathematics » Numeric » Add*,
- do ramki *False* obiektu *Case Structure* wstawić obiekt *Funtions » Mathematics » Numeric » Subtrack*,
- przejść do okna *Front Panel* i z palety *Controls* wybrać i umieścić w oknie następujące obiekty:
  Controls » Modern » Boolean » Horizontal Toggle Switch,
  Controls » Modern » Numeric» Numeric Control (2X),
  Controls » Modern » Numeric» Numeric Indicator,
  Controls » Modern » Decorations » Raised Box (2X),
- zmodyfikować nazwy i zawartość obiektów zgodnie z rys. 6 (okno *Front Panel*),
- naniesione obiekty połączyć zgodnie ze schematem przedstawionym na rys. 6 (okno *Block Diagram*),
- w oknie *Front Panel* wybrać opcję *Edit » Make Current Values Default* a następnie zapisać program nadając mu nazwę „ProgG_003.vi”, uruchomić, sprawdzić jego działanie. **Jakim instrukcjom z języka C odpowiada obiekt sterujący *Case Structure*?**



Rys. 6. Okna *Block Diagram*, *Front Panel* „ProgG_003.vi”

3.2.3. Obiekt *Formula Node*

W celu zapoznania się ze sposobem wykorzystania obiektu *Formula Node* zrealizować następujące zadania:
- utworzyć nowy program VI wybierając projekt *BlankVI*,
- do okna *Block Diagram* wstawić obiekt *Funtions » All Functions » Structures » Formula Node*,
- z menu obiektu *Formula Node* wybrać dwukrotnie opcję *Add » Input* i dwukrotnie opcję *Add » Output*, spowoduje to pojawienie się dwóch wejść i dwóch wyjść w obiekcie,
- przejść do okna *Front Panel* i z palety *Controls* wybrać i umieścić w oknie następujące obiekty:
  Controls » Num Ctrls » Num Ctrl (2X),
  Controls » Num Inds » Num Ind (2X),



Rys. 7. Okna *Block Diagram*, *Front Panel* „ProgG_004.vi”

- naniesione obiekty połączyć zgodnie ze schematem przedstawionym na rys. 7 (okno *Block Diagram*),
- do obiektu *Formula Node* wpisać tekst używając do tego celu narzędzia *Text* z palety *Tools*,
- zmodyfikować nazwy i zawartość obiektów zgodnie z rys. 7 (okno *Front Panel*),
- w oknie *Front Panel* wybrać opcję *Edit » Make Current Values Default* a następnie zapisać program nadając mu nazwę „ProgG_004.vi_xx” (gdzie xx inicjały twórcy programu), uruchomić i sprawdzić jego działanie. **Jakie instrukcje języka C można używać w obiekcie *Formula Node*?**

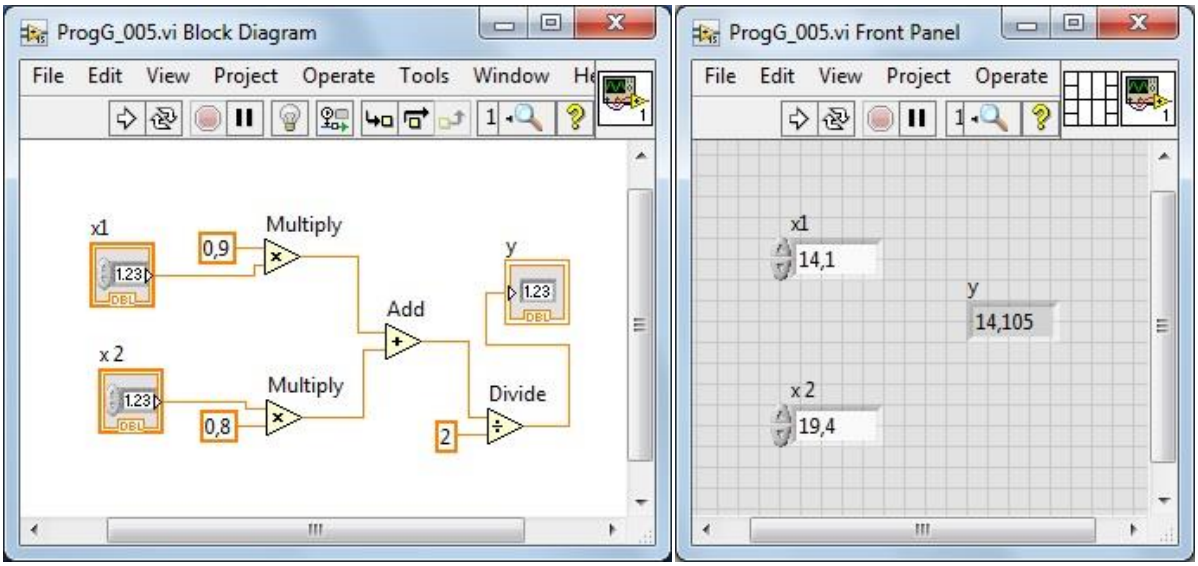
Obiekty sterujące *For Loop* i *While Loop* wraz z operatorem *Shift Register* zostały omówione w ćwiczeniu „Wprowadzenie do programowania w LabView”.

3.3. Tworzenie i użycie podprogramów

W celu ilustracji tworzenia i użycia podprogramów *VI* zostanie napisana procedura obliczająca wartość *y* na podstawie wzoru:

$$y = \frac{x_1 \cdot 0{,}9 + x_2 \cdot 0{,}8}{2}.$$

Zrealizować następujące zadania:
- utworzyć nowy program *VI* wybierając projekt *BlankVI*,
- do okna *Block Diagram* wstawić następujące obiekty:
  Functions » Mathematics » Numeric » Multiply (2X),
  Functions » Mathematics » Numeric » Add,
  Functions » Mathematics » Numeric » Divide,



Rys. 8. Okna *Block Diagram, Front Panel* „ProgG_005.vi”

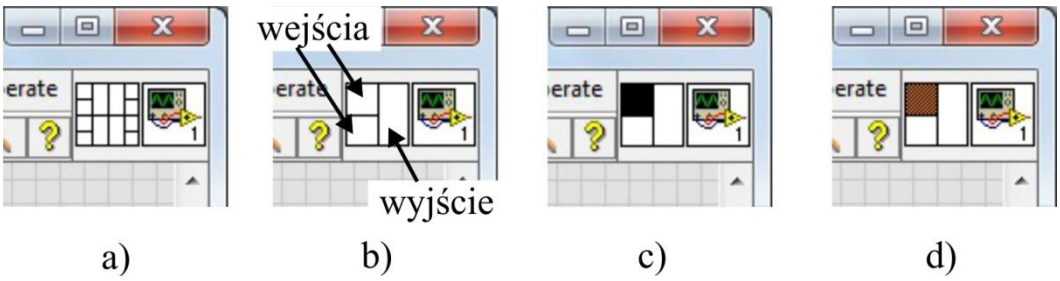
- z palety *Tools* wybrać narzędzie *Connect Wire*, kursorem najechać na jedno z wejść obiektu *Multiply* kliknąć prawym przyciskiem myszy i wybrać z menu obiektu opcję *Create » Constant*, czynności powtórzyć dla drugiego obiektu *Multiply* i wejścia *y* obiektu *Divide*,
- przejść do okna *Front Panel* i wstawić następujące obiekty:
  Controls » Modern » Numeric » Numeric Control (2X),
  Controls » Modern » Numeric » Numeric Indicator,
- naniesione obiekty połączyć zgodnie ze schematem przedstawionym na rys. 8 (okno *Block Diagram*),
- zmodyfikować nazwy i zawartość obiektów zgodnie z rys. 8 (okno *Front Panel* i okno *Block Diagram*),
- w oknie *Front Panel* wybrać opcję *Edit » Make Current Values Default* a następnie zapisać program nadając mu nazwę nadając mu nazwę „ProgG_005.vi_xx” (gdzie xx inicjały twórcy programu), uruchomić i sprawdzić jego działanie.
- utworzyć kopię zapasową „ProgG_005_kz.vi” programu wybierając File>>Save As.

Poniżej zostaną przedstawione dwa sposoby tworzenia podprogramów VI, Podprogram po utworzeniu i zapisaniu może być wołany z innego programu VI poprzez wybranie z palety Functions

3.3.1. Tworzenie podprogramu w programie VI- sposób 1.

Do demonstracji pierwszego sposobu tworzenia podprogramu zostanie wykorzystany program „ProgG_005.vi”. W prawym górnym rogu okna *Front Panel* znajduje się ikona reprezentująca program oraz siatka wejść i wyjść dla podprogramu, którą pokazano na rys. 9a. W celu utworzenia podprogramu wykonać następujące czynności:

- najechać kursorem na siatkę wejść i wyjść, kliknąć prawym przyciskiem myszy i z menu wybrać *Patterns* następnie siatkę złożoną z dwóch wejść i jednego wyjścia (rys. 9b),
- w celu przypisania obiektów programu do siatki wybrać z palety *Tools* narzędzie *Connect Wire*, najechać kursorem na górne wejście danych i kliknąć lewym przyciskiem myszy. Kwadrat reprezentujący wejście danych zmieni kolor (rys. 9c). Następnie najechać na obiekt *x1* w oknie *Front Panel* i kliknąć lewym przyciskiem myszy. Kwadrat reprezentujący wejście danych ponownie zmieni kolor (rys. 9d),
- w ten sam sposób przypisać obiekt *x2* do dolnego wejścia danych z siatki wejść i wyjść podprogramu oraz obiekt *y* do wyjścia danych z siatki,
- w oknie *Front Panel* wybrać opcję *Edit* » *Make Current Values Default* a następnie ponownie zapisać program.

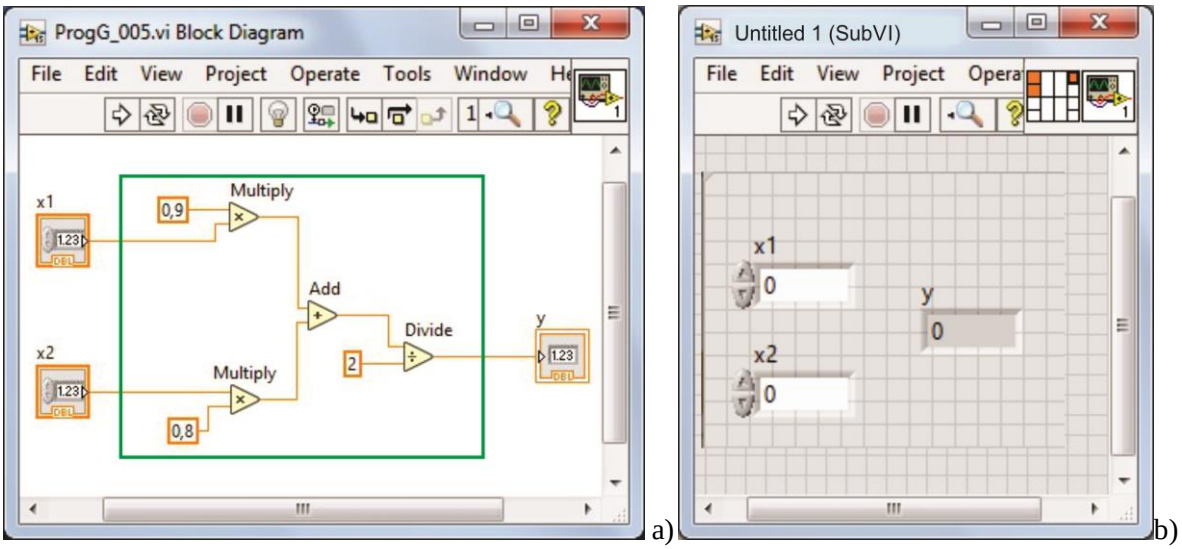


Rys. 9. Tworzenie podprogramu (sposób 1): (a) ikona reprezentująca podprogram i domyślna siatka wejść i wyjść, (b) siatka złożona z 2 wejść i 1 wyjścia, (c) i (d) zmiany kolorów wejścia danych podczas przypisywania mu obiektu programu.

3.3.2. Tworzenie podprogramu w programie VI- sposób 2.

Do demonstracji drugiego sposobu tworzenia podprogramu zostanie wykorzystana kopia zapasowa programu „ProgG_005_kz.vi".

- w oknie *Block Diagram* zaznaczyć obiekty znajdujące się na rys. 10a wewnątrz zielonego prostokąta,
- z menu okna wybrać *Edit>>Create SubVI*. LabVIEW automatycznie utworzy podprogram, wybierze siatkę wyprowadzeń oraz przypisze wejścia i wyjścia do siatki,
- kliknąć dwukrotnie na ikonę utworzonego podprogramu, sprawdzić i poprawić opisy wejść i wyjść podprogramu (patrz rys. 10b),

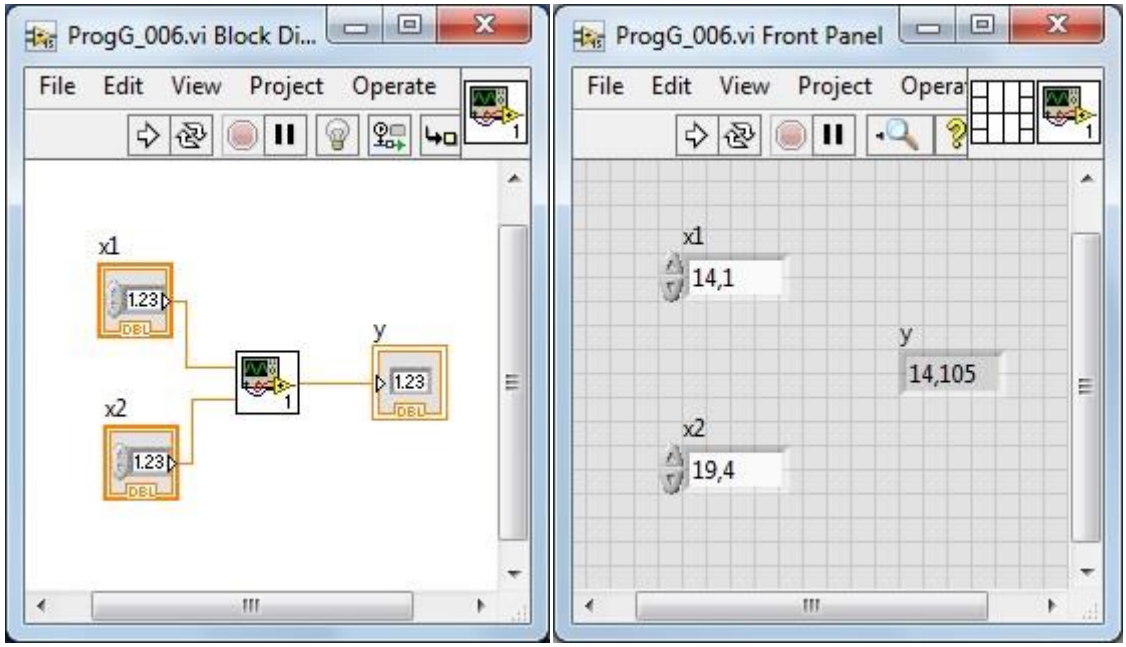


a) b)

Rys. 10. Tworzenie podprogramu (sposób 2): (a) zaznaczanie obiektów tworzących podprogram, (b) podprogram po utworzeniu.

3.3.3. Wywołanie programu VI z poziomu programu nadrzędnego.

W celu wykorzystania utworzonego podprogramu w innym programie LabView wykonać następujące czynności:
- utworzyć nowy program *VI* wybierając projekt *BlankVI*,
- przejść do okna *Blok Diagram* i z palety *Functions* wybrać *All Functions » Addons » Select a VI*, odnaleźć wcześniej utworzony podprogram „ProgG_005.vi" i wstawić do tworzonego programu,
- przejść do okna *Front Panel* i wstawić następujące obiekty:
  Controls » Modern » Numeric » Numeric Control (2X),
  Controls » Modern » Numeric » Numeric Indicator,
- naniesione obiekty połączyć zgodnie ze schematem przedstawionym na rys. 10 (okno *Block Diagram*),



Rys. 10. Okna *Block Diagram, Front Panel* „ProgG_006.vi"

- zmodyfikować nazwy i zawartość obiektów zgodnie z rys. 10 (okno *Front Panel*),
- w oknie *Front Panel* wybrać opcję *Edit » Make Current Values Default* a następnie zapisać program nadając mu nazwę „ProgG_006.vi_xx" (gdzie xx inicjały twórcy programu), uruchomić i sprawdzić jego działanie.

4. Zadania sprawdzające do samodzielnej realizacji

- Wykorzystując obiekt *Flat Sequence* opracować program do pomiaru czasu wykonania pętli *For Loop* obliczającej 99999999 wyraz ciągu arytmetycznego, dla którego pierwszy wyraz =5 oraz różnica =6. Nazwa programu „ProgG_007.vi_xx" (gdzie xx inicjały twórcy programu).
- W poprzednim programie zamiast struktury sterującej *Flat Sequence* użyć struktury sterującej *Stacked Sequence*. Nazwa programu „ProgG_008.vi_xx" (gdzie xx inicjały twórcy programu).
- Wykorzystując obiekt *Formula Node* opracować program sprawdzający czy 3 odcinki o podanych długościach mogą utworzyć trójkąt. Nazwa programu „ProgG_009.vi_xx" (gdzie xx inicjały twórcy programu).
- Na podstawie programu z poprzedniego podpunktu utworzyć podprogram. Utworzony podprogram wykorzystać we własnym samodzielnie zaproponowanym programie. Nazwa programu „ProgG_010.vi_xx" (gdzie xx inicjały twórcy programu).
- Zaproponować i opracować program wykorzystujący strukturę sterującą *Case Structure*. Nazwa programu „ProgG_011.vi_xx" (gdzie xx inicjały twórcy programu).

5. Pytania sprawdzające

- 5.1. Jak można zmienić typ dowolnej zmiennej numerycznej?
- 5.2. Czym różnią się linie łączące tablice 2D od linii łączących tablice 3D?

- 5.3. Jak można zmienić sposób rysowania zmiennej numerycznej w oknie *Block Diagram* z ikony na „terminal”?
- 5.4. Jak obiekty *Increment* i *Decremrement* modyfikują tablice 1D i 2D?
- 5.5. Jakie są różnice pomiędzy obiektami *Control* i *Indicator*?
- 5.6. Jakie są różnice pomiędzy obiektami *Control* i *Constant*?
- 5.7. Z czego wynika różnica w zmierzonych czasach dla pierwszego i drugiego uruchomienia programu ProgG_002.vi?
- 5.8. Jak działa obiekt sterujący *Flat Seqence*?
- 5.9. Jakim instrukcjom z języka C odpowiada obiekt sterujący *Case Structure*?
- 5.10. Czym różnią się obiekty *Stacked Sequence* i *Flat Sequence*?
- 5.11. Jakie instrukcje języka C można używać w obiekcie *Formula Node*?
- 5.12. Czy obiekt *Formula Node* rozróżnia duże i małe litery?
- 5.13. W jakim celu jest stosowane polecenie *Edit » Make Current Value Default* środowiska LabVIEW?
- 5.14. Do czego służy polecenie środowiska *Edit » Clean Up Diagram* środowiska LabVIEW?
- 5.15. Jak działa polecenie *Edit » Remove Broken Wires* środowiska LabVIEW?

6. Zawartość sprawozdania

Sprawozdanie z ćwiczenia powinno zawierać:
- krótką (pisemna) ocenę zalet i wad graficznego sposobu programowania,
- zrzuty ekranów napisanych 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.