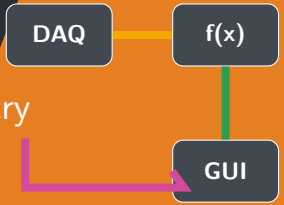
Od pomiaru do programu w LabVIEW

Środowisko, dataflow, podstawowe typy danych i klastry

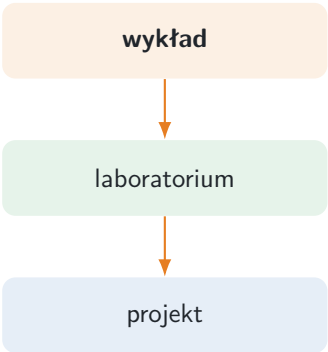


Wprowadzenie do LabVIEW i układów kontrolno-pomiarowych ▪ Wykład 1

dr inż. Angelika Tefelska

Organizacja przedmiotu

- **7 wykładów** po 2 godziny, co dwa tygodnie,
- laboratoria odbywają się **co tydzień**,
- wykład wyprzedza laboratorium: najpierw poznajemy narzędzia, potem używamy ich w praktyce,
- **sposób zaliczenia wykładu:** kolokwium organizowane na 7-mym wykładzie,
- **warunek zaliczenia wykładu:** uzyskanie powyżej 50% punktów,
- materiały: `https://labe.engined.eu/`,
- **konsultacje:** ustalane indywidualnie z osobami chętnymi.



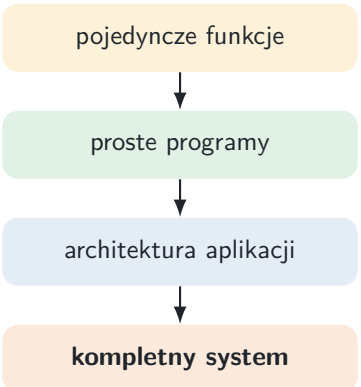
Szczegółowe zasady zaliczenia i terminy obowiązują zgodnie z aktualnym regulaminem przedmiotu dostępnym w USOS-ie.

Plan 7 wykładów

1. **Od pomiaru do programu w LabVIEW** — środowisko, dataflow, podstawowe typy danych, klastry.
2. **Dane i logika programu** — tablice, TypeDef, Case, For/While, tunele, Event Structure.
3. **Program, który pamięta i zapisuje** — wykresy, rejestry przesuwne, SubVI, błędy, pliki.
4. **Akwizycja danych** — DAQmx, myDAQ, próbkowanie, bufor, aliasing i podstawy analizy sygnału.
5. **Komunikacja z aparaturą** — VISA, SCPI, Analog Discovery II, maszyna stanów.
6. **Większe aplikacje** — local/global/shared, FGV, notifier, master/slave, kolejki, Producer–Consumer, Property/Invoke Nodes, references.
7. **Od programu do aplikacji (1h)** — dystrybucja, odporność na błędy, projekt; na końcu dodatkowe zagadnienia jak: grafika 2D/3D, kursor, Picture Ring. **Kolokwium (1h)**.
8. **Kolokwium poprawkowe**

Laboratoria i projekt — po co ten układ?

- 10 punktowanych spotkań laboratoryjnych,
- maksymalnie 10 pkt za każde laboratorium,
- miniprojekt w formie pracy domowej za 10 pkt.,
- projekt końcowy: **50 pkt**,
- projekt ma sprawdzić, czy potraficie połączyć poznane elementy w jedną aplikację.



Ocena końcowa

- Skala ocen stosowana do wyznaczenia oceny z wykładu i laboratorium:

Ocena	Zakres procentowy uzyskanych punktów
5,0	[91;100]%
4,5	[81;91)%
4,0	[71;81)%
3,5	[61;71)%
3,0	[51;61)%

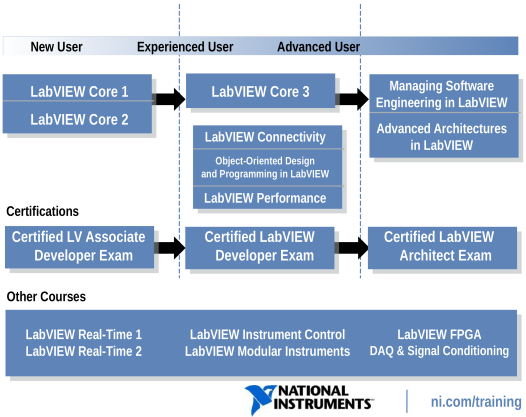
- Ocena końcowa wyznaczana jest na podstawie wzoru:

$$0.7 \cdot \textit{ocena z laboratorium} + 0.3 \cdot \textit{ocena z wykładu}. \tag{1}$$

- Uzyskanie z egzaminu CLAD [70;85)% zwiększa ocenę końcową o 0,5.
- Uzyskanie z egzaminu CLAD [85;100]% zwiększa ocenę końcową o 1.

Egzamin CLAD

- Egzamin CLAD jest przeprowadzany na Politechnice Warszawskiej przez firmę National Instruments.
- Podejście do egzaminu CLAD jest płatne lecz finansowane przez WF PW dla najlepszych studentów.
- Ranking najlepszych studentów jest ustalany na podstawie punktów **tylko** z zajęć punktowanych realizowanych na laboratoriach.
- Do zaliczenia egzaminu wymagane jest zdobycie co najmniej 70% punktów. Egzamin składa się z 40 pytań wielokrotnego wyboru.
- **Nieusprawiedliwiona nieobecność na egzaminie po zapisaniu się na niego skutkuje przyznaniem kary w wysokości: -30 pkt.**
- Certyfikat CLAD jest ważny przez 2 lata.



Co obecnie sprawdza CLAD?

Hardware	Environment	Fundamentals	Best Practices
10%	25%	50%	15%

Dzisiejszy wykład dotyka przede wszystkim **Programming Environment**, **Programming Fundamentals** oraz **Best Practices** — m.in. typów danych, dataflow i klastrów.

Instalacja LabVIEW

- W celu instalacji LabVIEW należy wejść na stronę: link
- Następnie należy wskazać swój system operacyjny: Windows, Linux (tylko dystrybucja Ubuntu), Mac OS.
- Należy wybrać dystrybucje ***Community*** aby pobrać darmową wersje LabVIEW.

* dostępność edycji zależy od systemu i licencji

Literatura

- **National Instruments, Core 1 i 2 – podręcznik oraz zeszyt ćwiczeń. Dostępny w bibliotece Wydziału Fizyki spod lady".**
- D.Tefelski, A.Tefelska, LabVIEW and Open Source Solutions, link do wersji on-line
- W. Tłaczała, Środowisko LabVIEW w eksperymencie wspomaganym komputerowo, Wydawnictwa Naukowo -Techniczne, Warszawa 2005
- **W. Tłaczała, Wirtualne laboratorium podstaw techniki cyfrowej, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2008**
- Thomas J. Bress, Ęffective LabVIEW Programming", NTS press, 2013
- M. Chruściel, LabVIEW w praktyce, Wydawnictwo BTC, Legionowo 2008

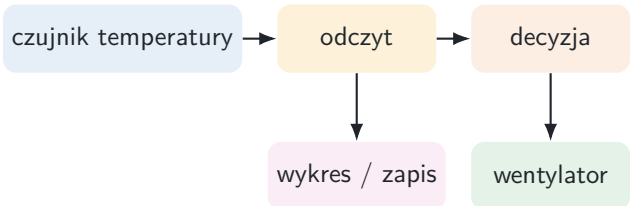
1. Od układu pomiarowego do VI

Najpierw problem, potem narzędzie

Co właściwie będziemy budować?

- LabVIEW nie jest celem samym w sobie — jest narzędziem do **pomiaru, analizy, prezentacji i sterowania**.
- Na kolejnych wykładach będziemy rozszerzać ten sam schemat o komunikację, synchronizację i architekturę programu.

Przykład: układ kontroli temperatury



- wejście,
- obliczenia,
- warunek,
- interfejs,
- wyjście,
- zapis danych.

Na końcu semestru wszystkie te elementy powinny przestać być osobnymi „klockami”.

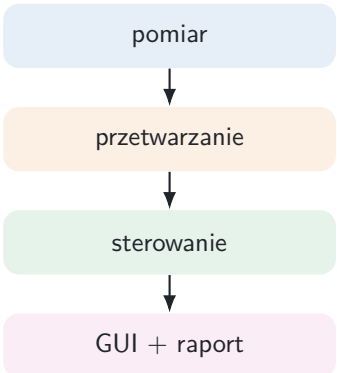
Co to jest LabVIEW?

- **LabVIEW** = Laboratory Virtual Instrument Engineering Workbench.
- Graficzne środowisko programistyczne oparte na języku **G**.
- LabVIEW zostało stworzone w celu oprogramowywania urządzeń pomiarowych, akwizycji danych i sterowania urządzeniami za pomocą intuicyjnego interfejsu użytkownika.
- Kolejność wykonywania operacji definiują połączenia pomiędzy węzłami, a nie kolejnością linii tekstu.
- Szczególnie dobrze pasuje do **pomiarów, testów, automatyzacji, analizy sygnałów i sterowania**.

Acquire Analyze Visualize Control

Dlaczego LabVIEW?

- łatwe tworzenie interfejsu użytkownika,
- duża liczba gotowych sterowników i przykładów do sprzętu pomiarowego,
- naturalna praca z **równoległymi** zadaniami,
- analiza i wizualizacja danych.

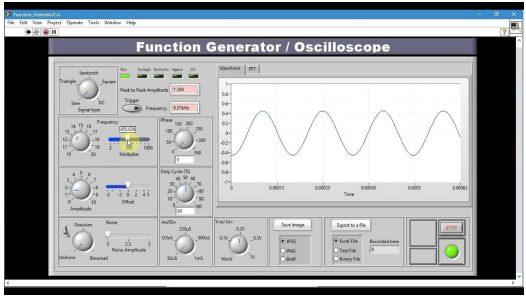


LabVIEW w praktyce

- laboratoria badawcze,
- systemy testowe,
- automatyka i produkcja,
- lotnictwo i motoryzacja,
- uczelnie i stanowiska dydaktyczne.

Przyrząd wirtualny — VI

- Program w LabVIEW nazywamy **Virtual Instrument (VI)**.
- Nazwa wynika z idei zastąpienia części funkcji tradycyjnego przyrządu pomiarowego programem.
- VI może mieć własny panel użytkownika i może być używany jako element większej aplikacji.



Źródło: `https://www.youtube.com/watch?v=ru1KFyL9Tgo`

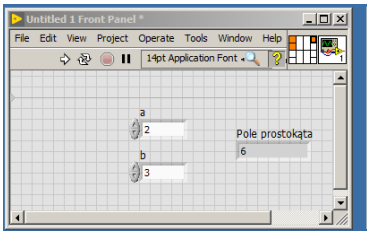
2. Jak zbudowany jest program w LabVIEW?

Front Panel, Block Diagram i dataflow

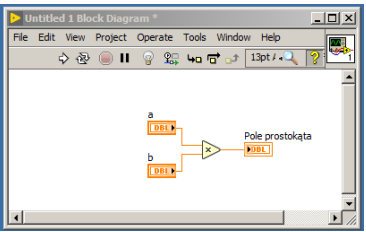
Trzy elementy VI

Każdy VI składa się z:

1. **Front Panel** — interfejs użytkownika,
2. **Block Diagram** — kod programu,
3. **Icon & Connector Pane** — sposób użycia VI jako SubVI.



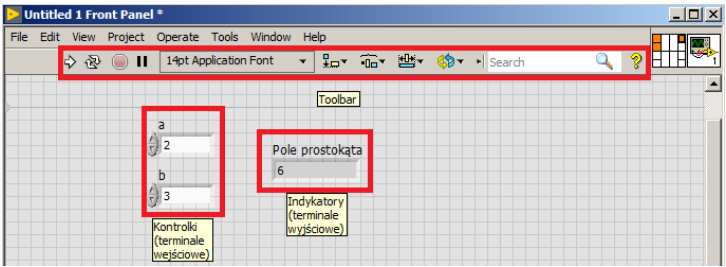
Front panel



Block diagram

Icon and conector pane

Front Panel



- **kontrolka** (eng. *control*) — dane wejściowe od użytkownika,
- **indykator** (eng. *indicator*) — dane wyjściowe czyli wyniki programu.

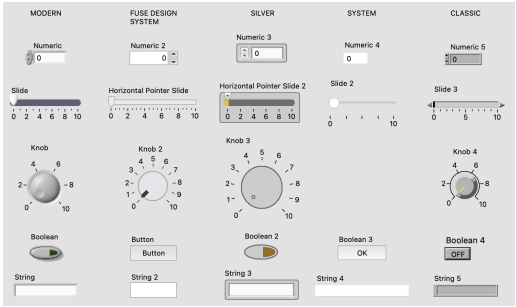
Control
użytkownik → program

→

Indicator
program → użytkownik

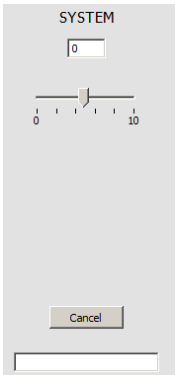
- Pokrętło wartości zadanej temperatury: **control**.
- Przycisk START: **control**.
- Pole z aktualnym pomiarem temperatury: **indicator**.
- Dioda „awaria”: **indicator**.

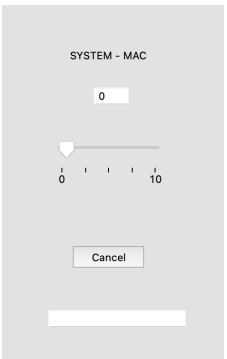
Wygląd kontrolek ma znaczenie



- Paleta *Modern* posiada kontrolki i indykatory o nowoczesnym wyglądzie w porównaniu do klasycznej palety. **Zastosowanie:** do większości aplikacji.
- Paleta *Fuse Design System* to paleta dedykowana do aplikacji, które mają być częścią innych platform NI, gdzie zastosowano ten sam styl.
- Paleta *Silver* charakteryzuje się szeroką gamą dostępnych kontrolek oraz indykatorów. **Zastosowanie:** interaktywne aplikacje przeznaczone dla szerokiej grupy osób.
- Paleta *Klasyczna* jest dedykowana do aplikacji, które mają być częścią większego projektu lub do aplikacji, w których kontrolki/indykatory będą dostosowywane do własnych potrzeb (zmiana wyglądu, rodzaju skali itd).
- Paleta *System* powinna być stosowana w VI, które mają pełnić funkcje okien dialogowych. Wygląd kontrolek z palety *System* będzie różny w zależności od systemu operacyjnego (Windows/Mac/Linux).

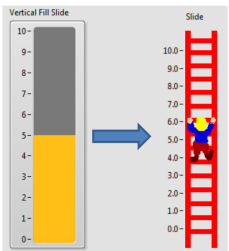
System controls — po co istnieją?

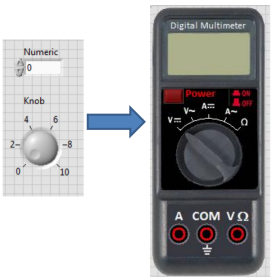




- kontrolki dopasowują się do wyglądu systemu operacyjnego,
- są użyteczne w oknach dialogowych,
- wygląd może różnić się między Windows, macOS i Linux.

Modyfikowanie stylu kontrolek

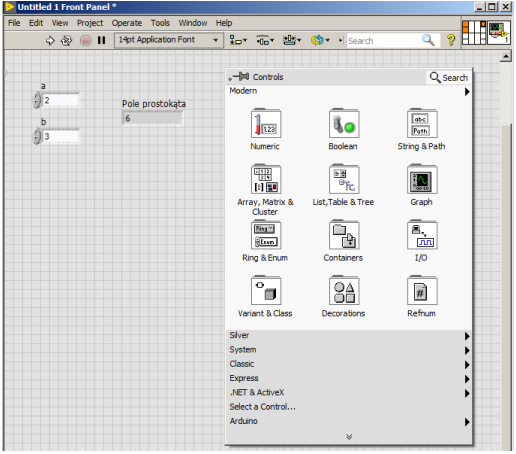




- kolor i skala,
- etykiety,
- zakres,
- elementy dekoracyjne,
- własny wygląd do projektu.

Paleta Functions i Controls

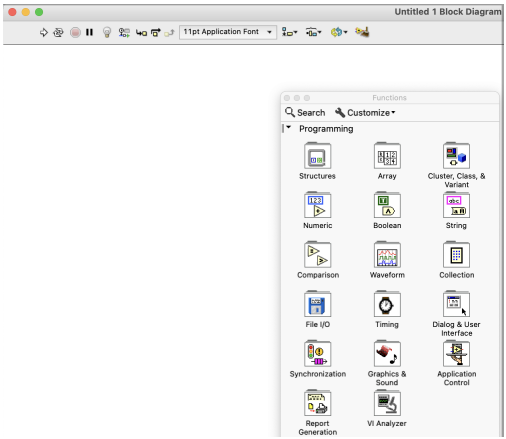
Paleta *Controls*



Paletę otwieramy na Front Panel.

Elementy są pogrupowane według funkcji i stylu.

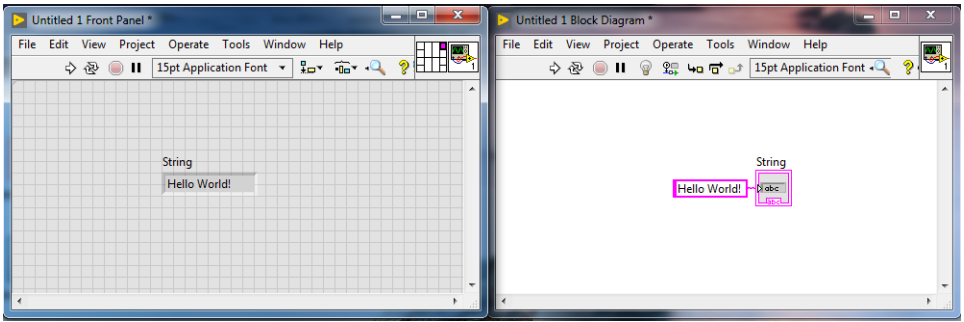
Paleta *Functions*



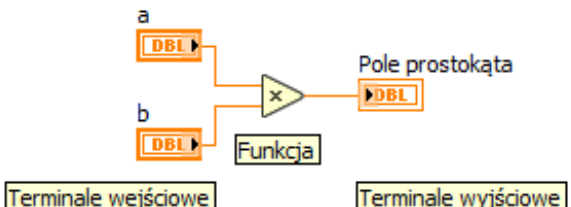
Na Block Diagram umieszczamy funkcje, struktury programistyczne i terminale.

Pierwszy VI: wejście → obliczenie → wyjście

- W przeciwieństwie do tekstowych języków, w LabVIEW zmienne oraz funkcje są w postaci ikon.
- W tekstowych językach kolejność wykonywania komend odbywa się linia po linii. W LabVIEW kierunek jest determinowany poprzez węzły (*data flow programming*).
- Niezależne od siebie fragmenty kodu mogą wykonywać się równolegle.



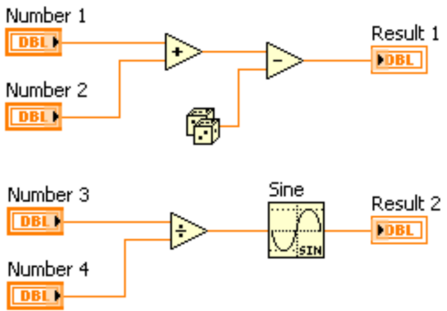
Dataflow — najważniejsza zasada LabVIEW



- węzeł uruchamia się, gdy ma **wszystkie wymagane dane wejściowe**,
- nie decyduje położenie „z lewej na prawą” czy "góra → dół",
- zatem mnożenie wykona się wtedy gdy dostanie wartość *a* i *b*.

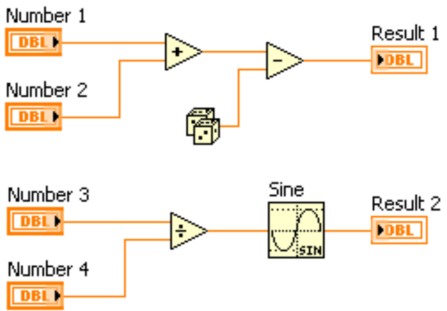
Która funkcja wykona się pierwsza?

(a) dodawanie,
(b) odejmowanie,
(c) losowanie liczby,
(d) dzielenie,
(e) sinus.



Odpowiedź: dataflow pozwala na równoległość

- **dodawanie** — możliwe,
- **losowanie liczby** — możliwe,
- **dzielenie** — możliwe,
- odejmowanie i sinus czekają na dane.



Pytanie kontrolne

Dwa niezależne fragmenty Block Diagram nie są połączone przewodem. Co można powiedzieć o kolejności ich wykonania?

(a) zawsze wykonuje się fragment po lewej
(b) zawsze wykonuje się fragment położony wyżej
(c) LabVIEW może wykonywać je równolegle, jeśli nie ma innej zależności
(d) kolejność zależy wyłącznie od nazw kontrolek

dataflow

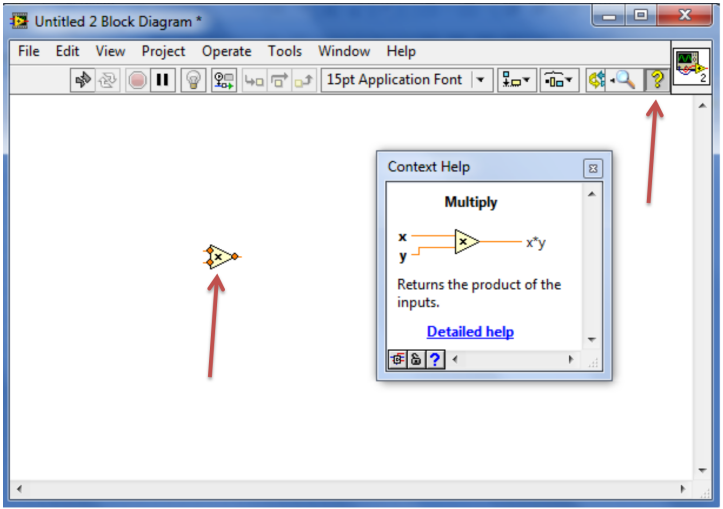
Pytanie kontrolne

Dwa niezależne fragmenty Block Diagram nie są połączone przewodem. Co można powiedzieć o kolejności ich wykonania?

(a) zawsze wykonuje się fragment po lewej
(b) zawsze wykonuje się fragment położony wyżej
(c) **LabVIEW może wykonywać je równolegle, jeśli nie ma innej zależności**
(d) kolejność zależy wyłącznie od nazw kontrolek

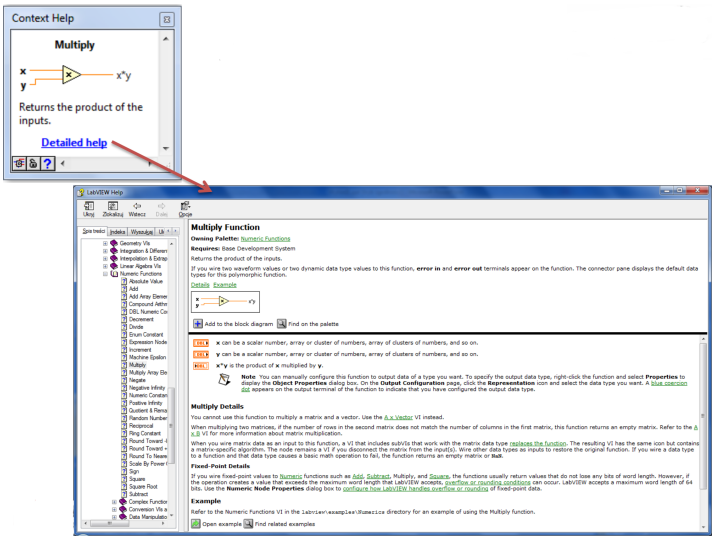
dataflow

Pomoc kontekstowa



- szybki opis funkcji,
- wejścia i wyjścia,
- typy danych,
- **Ctrl+H**.

Pomoc szczegółowa i przykłady

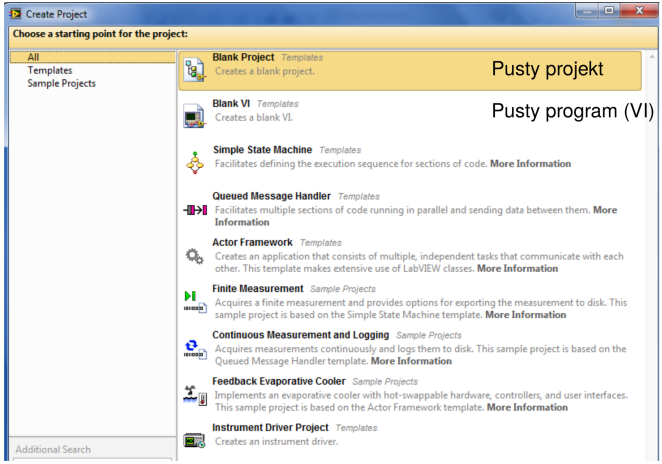


- szczegółowa dokumentacja,
- Examples / Example Finder,
- gotowe VI jako punkt startowy,
- uczymy się **czytać Help**, nie zapamiętywać każdej funkcji.

Skróty, które naprawdę oszczędzają czas

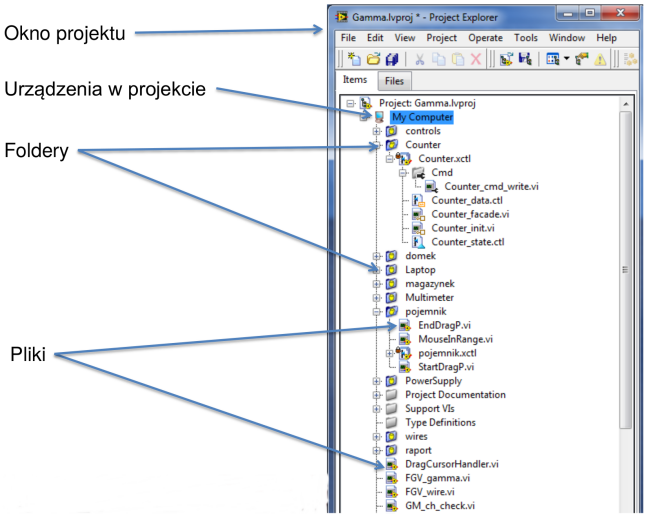
Ctrl+E	przełączanie Front Panel $\leftrightarrow$ Block Diagram
Ctrl+T	ułożenie Front Panel i Block Diagram obok siebie
Ctrl+H	Context Help
Ctrl+B	**usuwanie błędnych przewodów**
Ctrl + przeciąganie	kopiowanie elementu
Ctrl+Z	cofnięcie ostatniej operacji

Projekt LabVIEW



- projekt porządkuje pliki,
- przechowuje VI, biblioteki i zależności,
- od początku pracujemy tak, jak przy większej aplikacji.
- Aby stworzyć projekt wybieramy: **File** → **Create Project**

Project Explorer



- `.lvproj` — projekt,
- `.vi` — Virtual Instrument,
- `.ctl` — custom control / TypeDef,
- później: biblioteki, build specifications itd.

Czas na podsumowanie tej części wykładu!

Zamiast tekstu — seria pytań kontrolnych.

SPRAWDŹMY, CO JUŻ UMIEMY

Pytanie kontrolne

Który element interfejsu powinien służyć do podania przez użytkownika wartości zadanej temperatury?

(a) Indicator
(b) Control
(c) Wire
(d) Connector Pane

środowisko

Pytanie kontrolne

Który element interfejsu powinien służyć do podania przez użytkownika wartości zadanej temperatury?

(a) Indicator
(b) **Control**
(c) Wire
(d) Connector Pane

środowisko

Pytanie kontrolne

Z jakich elementów składa się VI?

(a) Front panel
(b) Block diagram
(c) Projekt
(d) Icon and Connector Pane

środowisko

Pytanie kontrolne

Z jakich elementów składa się VI?

(a) **Front panel**
(b) **Block diagram**
(c) Projekt
(d) **Icon and Connector Pane**

środowisko

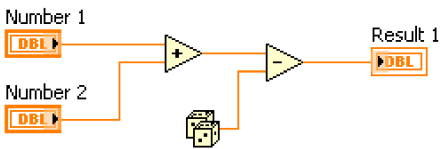
Pytanie kontrolne

Która funkcja wykona się jako pierwsza?

a) Dodawanie
b) Odejmowanie
c) Losowanie liczby
d) Nie wiadomo

Która funkcja wykona się jako ostatnia

a) Dodawanie
b) Odejmowanie
c) Losowanie liczby
d) Nie wiadomo



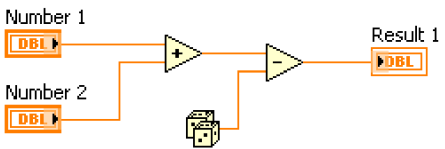
Pytania kontrolne

Która funkcja wykona się jako pierwsza?

a) Dodawanie
b) Odejmowanie
c) Losowanie liczby
d) **Nie wiadomo**

Która funkcja wykona się jako ostatnia?

a) Dodawanie
b) **Odejmowanie**
c) Losowanie liczby
d) Nie wiadomo



3. Podstawowe typy danych

Kolor przewodu niesie informację

Typy danych w LabVIEW

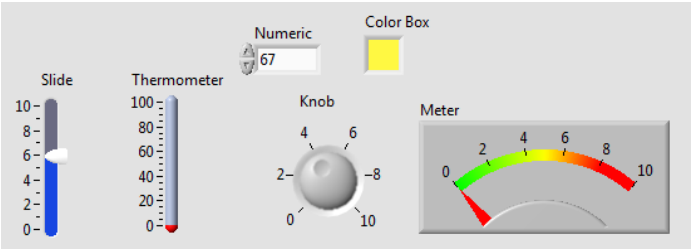
numeric | **Boolean** | **string** | **enum**

array
następny wykład

cluster
dzisiaj

Waveform i bardziej specjalistyczne typy pojawią się później, gdy będą potrzebne.

Dane numeryczne



- liczby całkowite,
- liczby zmiennoprzecinkowe,
- signed / unsigned,
- różna liczba bitów = inny zakres i pamięć.

Po co różne reprezentacje liczb?

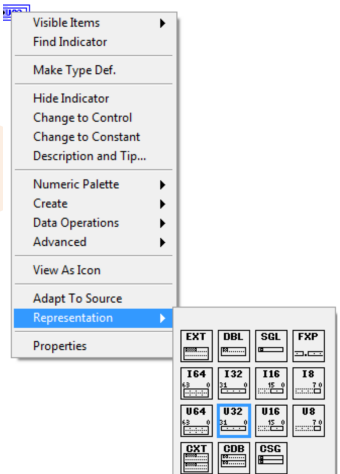
U8 / I16 / I32
liczby całkowite

→

SGL / DBL
liczby rzeczywiste

→

wybór
zakres + precyzja



- Indeks elementu tablicy nie wymaga DBL.
- Wynik pomiaru napięcia często naturalnie zapisujemy jako DBL.
- Typ powinien pasować do znaczenia danych, a nie być wybierany przypadkowo.

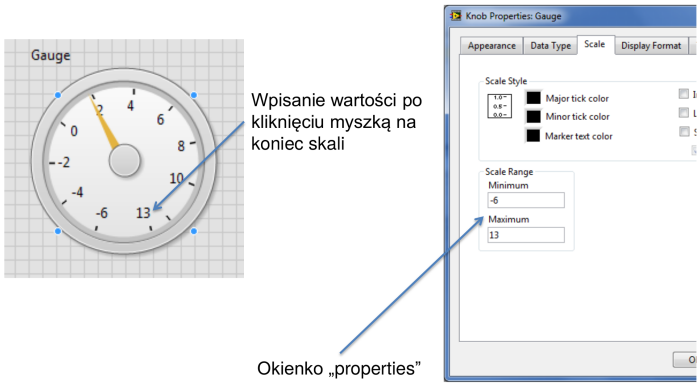
Prawy przycisk myszy → Representation

Zakres zależy od reprezentacji

Typ	Liczba bitów	Zakres
Single-precision, floating-point	32	+1.40e–45 do +3.40e+38 oraz –1.40e–45 do –3.40e+38
Double-precision, floating-point	64	+4.94e–324 do +1.79e+308 oraz –1.79e+308 do –4.94e–324
Extended-precision, floating-point	128	+6.48e–4966 do +1.19e+4932 oraz –1.19e+4932 do –6.48e–4966
Byte signed integer	8	-128 do 127
Word signed integer	16	-32768 do 32767
Long signed integer	32	–2 147 483 648 do 2 147 483 647
Quad signed integer	64	–1e19 do 1e19
Byte unsigned integer	8	0 do 255
Word unsigned integer	16	0 do 65 535
Long unsigned integer	32	0 do 4 294 967 295
Quad unsigned integer	64	0 do 2e19

Dobierając typ całkowity, pytamy przede wszystkim: **jaki zakres wartości jest możliwy?**

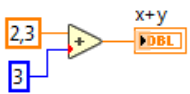
Zakres kontrolki to coś innego niż typ danych



- typ definiuje reprezentację danych,
- Data Entry określa dopuszczalne wartości w interfejsie,
- oba ustawienia mogą być potrzebne jednocześnie.

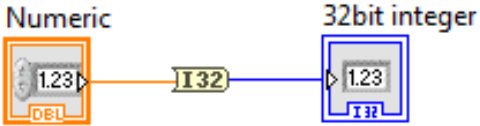
Coercion Dot — LabVIEW wykonuje konwersję

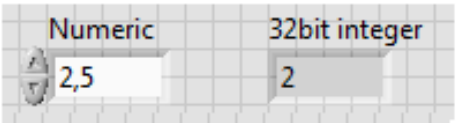
- Jeśli do funkcji numerycznej podłączymy dwie zmienne o różnej reprezentacji to funkcja zwróci nam wartość w **"większym"formacie** np:

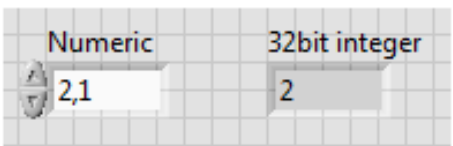


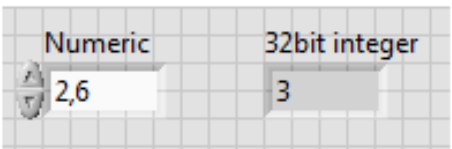
- Jeśli podłączymy do funkcji **signed integer** oraz **unsigned integer** to na wyjściu funkcji uzyskamy **unsigned integer**.
- Wszelka konwersja liczby przez funkcję będzie zaznaczona jako ***Coercion Dot***.

Konwersja jawna jest czytelniejsza



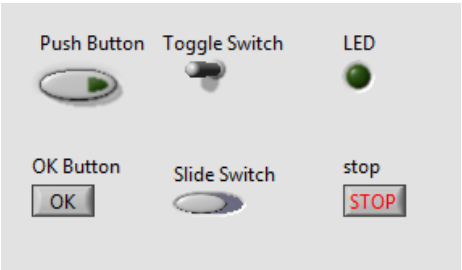






- Gdy konwersja jest częścią logiki programu, lepiej pokazać ją jawnie.
- Przy konwersji DBL $\rightarrow$ integer trzeba pamiętać o zasadach zaokrąglania i zakresie.
- W przypadku konwersji DBL na całkowite, VI zwróci najbliższą całkowitą wartość np. 2.8 $\rightarrow$ 3.
- Jeśli wartość jest idealnie pośrodku dwóch całkowitych liczb, VI zwróci parzystą liczbę całkowitą np. 2.5$\rightarrow$2.

Boolean — tylko dwa stany

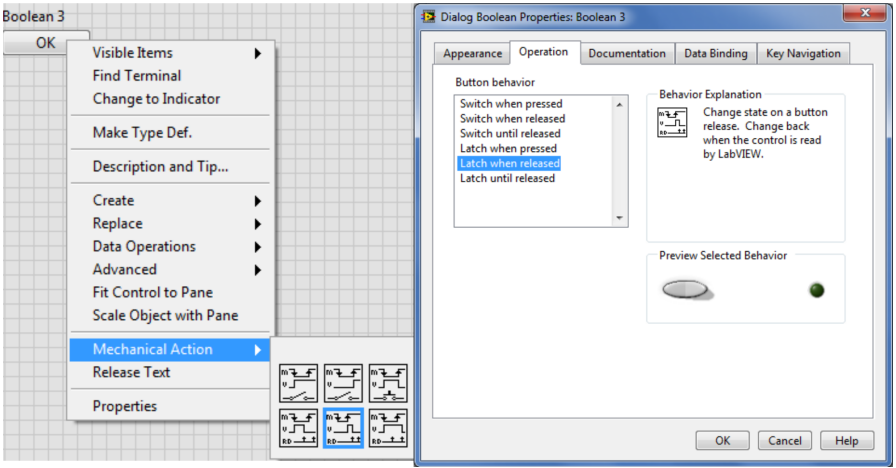


- TRUE / FALSE,
- ON / OFF,
- stan alarmu,
- decyzja: grzać / nie grzać,
- przewód ma kolor **zielony**.

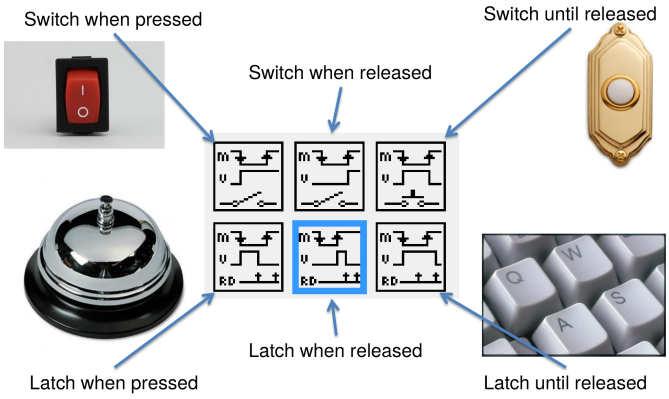
Mechanical Action przycisków

Dostęp poprzez shortcut menu

Dostęp poprzez okienko properties

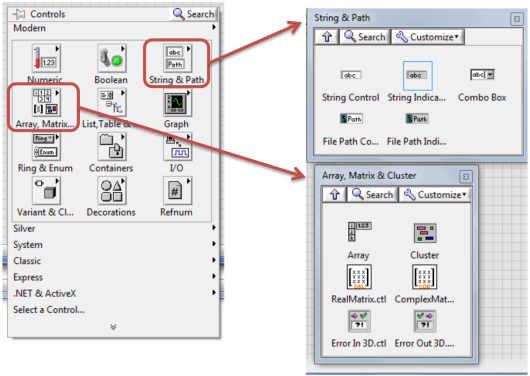


Switch vs Latch — myśl o zachowaniu użytkownika



- **Switch**: stan pozostaje po zmianie,
- **Latch**: stan wraca po odczycie przez program,
- przycisk STOP w pętli często naturalnie pasuje do mechaniki typu latch.

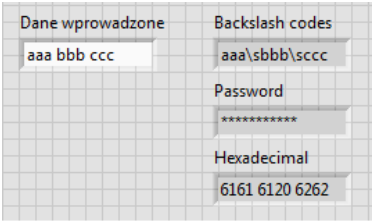
String — tekst jest również danymi



String - sekwencja ciągów znaków, zakodowanych przy użyciu ASCII, np:

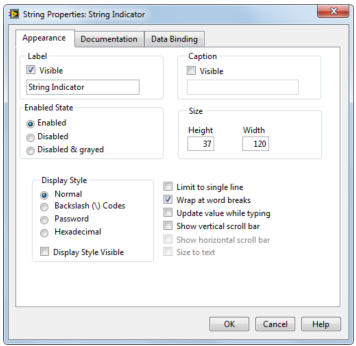
- nazwa pliku,
- komunikat dla użytkownika,
- komenda do przyrządu,
- odpowiedź z portu szeregowego,
- przewód ma kolor **różowy**.

Style wyświetlania łańcuchów znaków

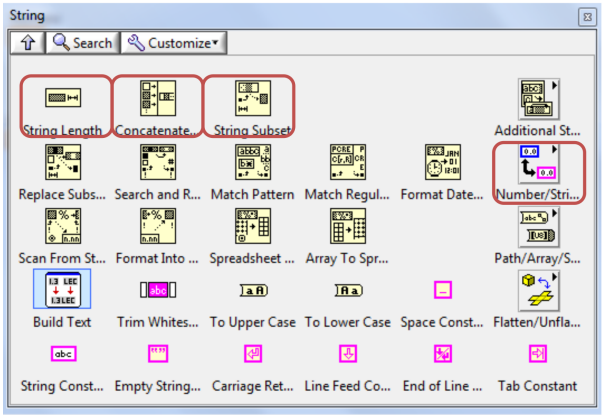


Kontrolki typu string mają 4 tryby wyświetlania:

- Normal
- Backslash codes
- Password
- Hexadecimal

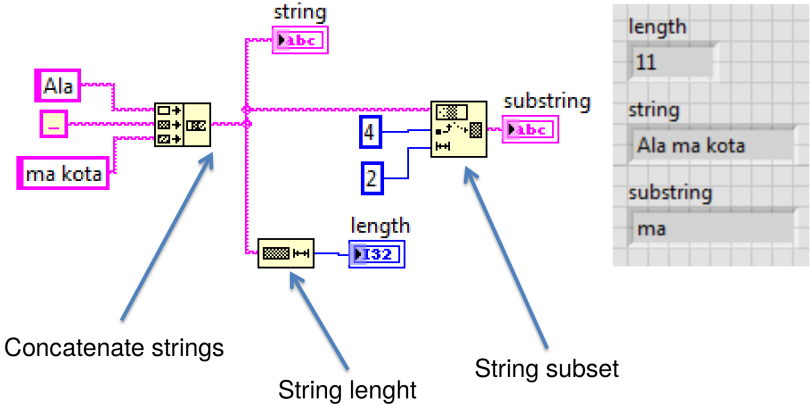


Podstawowe operacje na String



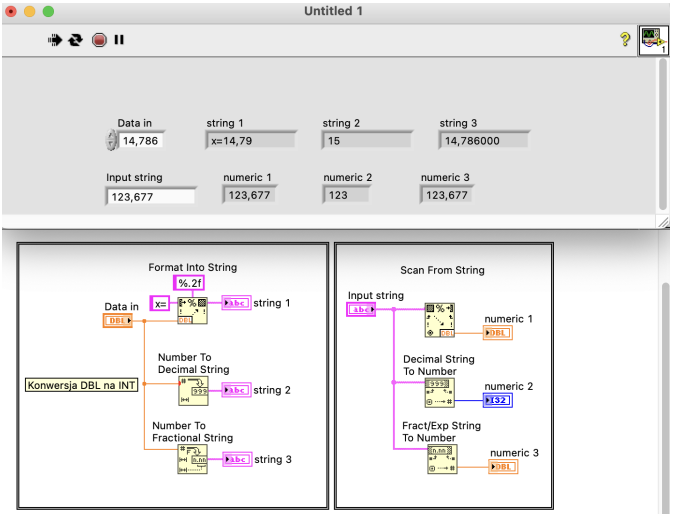
Typowe zadania: łączenie tekstów, wyszukiwanie fragmentu, długość, zamiana części łańcucha.

Więcej operacji na String



Nie uczymy się palety na pamięć — uczymy się rozpoznać problem i znaleźć właściwą funkcję.

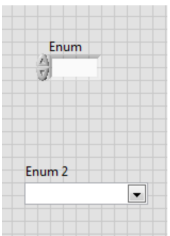
String ↔ Numeric



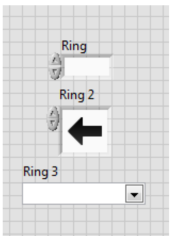
- prezentacja wyniku pomiaru w komunikacie,
- parsowanie liczby otrzymanej z urządzenia,
- zapis danych do pliku.

Typ wyliczeniowy

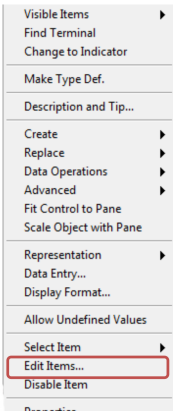
Enum



Ring

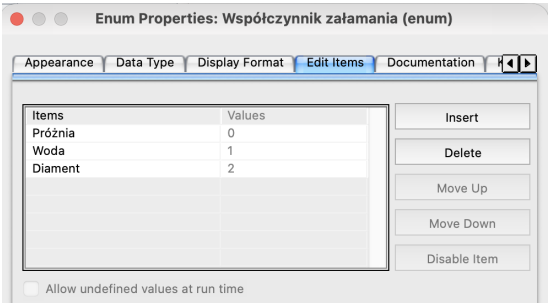


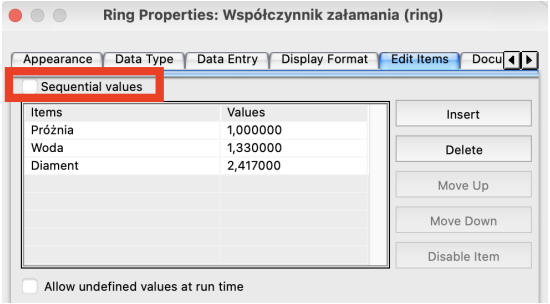
Obie kontrolki przyporządkowują każdej pozycji na liście wartość liczbową - unsigned int 16 (kolor niebieski).

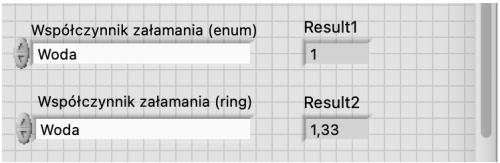


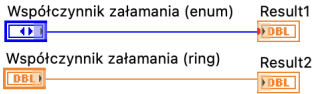
- przechowuje listę z nazwami, do których są przyporządkowane wartości numeryczne,
- bardzo ważny przy **Case Structure** i **maszynie stanów**.

Różnica między enum i ring

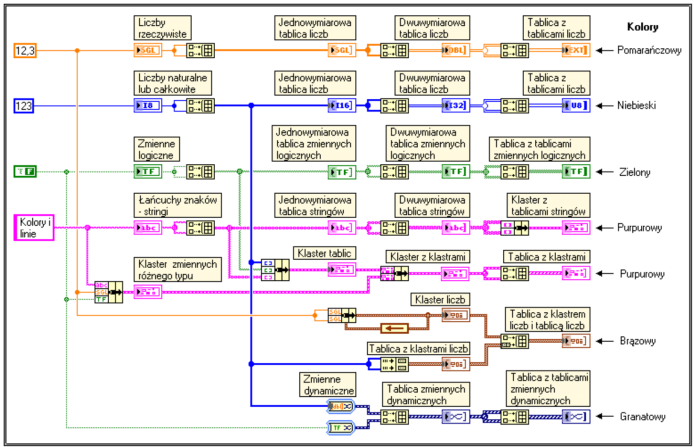








Kolor przewodu to podpowiedź



pomarańczowy — floating point **niebieski — integer**

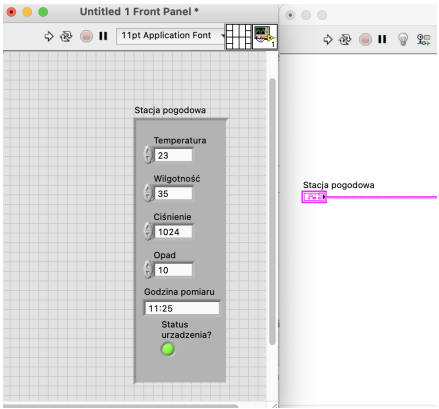
zielony — Boolean **różowy — String**

4. Klastry

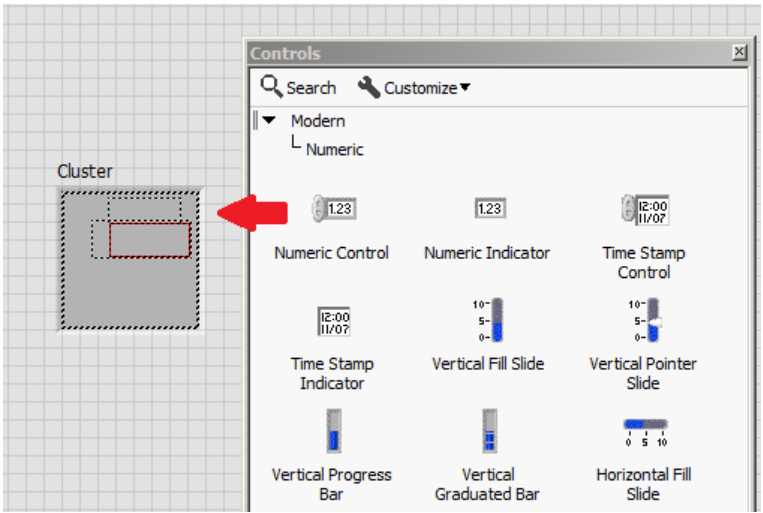
Kilka powiązanych danych jako jedna całość

Po co nam klaster już na pierwszym laboratorium?

- Klaster grupuje dane różnego typu - jak struktury w C++ czy listy w Python-ie
- Dzięki zgrupowaniu zmiennych, korzystamy w programie z jednej linii wyodrębniając potrzebne elementy a nie z kilku linii. Takie rozwiązanie poprawia czytelność kodu.
- Zastosowanie klastra ułatwia rozbudowę programu w przyszłości.
- **Ograniczenie:** wszystkie elementy w klastrze muszą być albo indykatorami albo kontrolkami.

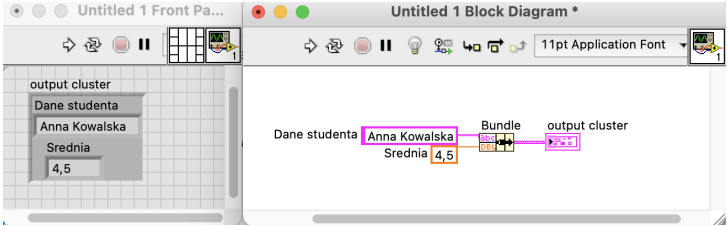


Tworzenie klastrów

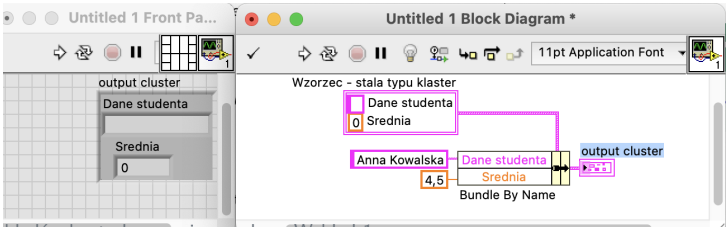


Tworzenie klastrów

- Funkcja ***Bundle*** - tworzy klaster z elementami, w takiej kolejności jak je podano na wejściu funkcji, np:



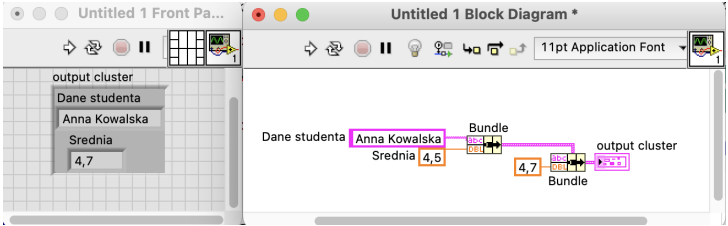
- Funkcja ***Bundle by Name*** - tworzy klaster na podstawie wzorca, np:



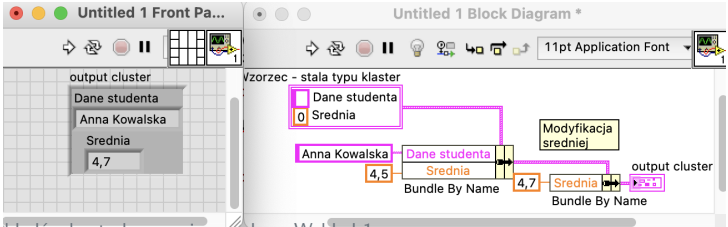
Modyfikacja elementów w klastrze

Modyfikacje elementów w klastrze wykonuje się za pomocą funkcji do tworzenia klastrów, czyli:

- ***Bundle***, np:

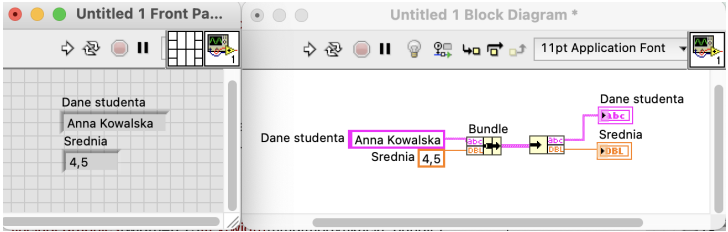


- ***Bundle by Name***, np:

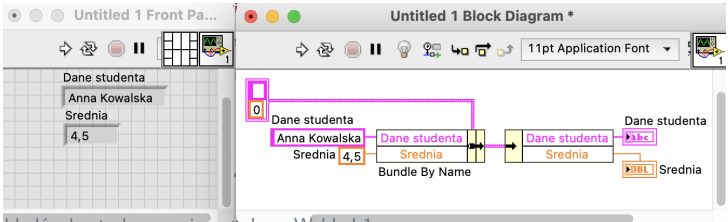


Odczytywanie elementów z klastra

- Funkcja ***Unbundle*** - zwraca elementy z klastra w takiej kolejności w jakiej były dodane do klastra, np:

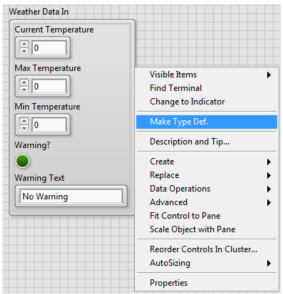


- Funkcja ***Unbundle by Name*** - zwraca wybrany po nazwie element z klastra, np:

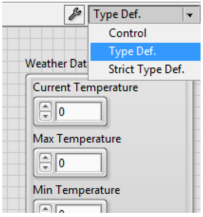


Definicja typu

- Definicja typu zawiera kopię typu danych(control, indicator, constant).
- Przy zmianie typu danych zmieniają się wszystkie kontrolki/indykatory z nim związane.
- Definicja typu obejmuje typ danych oraz wygląd kontrolki.
- Każda użyta definicja typu (*Type Definition*) może mieć własne: *caption*, *label*, *description*, *tip strip*, *default value*, *size*, *color*, *style of control/indicator*.
- Każda użyta definicja typu (*Strict Type Definition*) może mieć własne: *caption*, *label*, *description*, *tip strip*, *default value*.

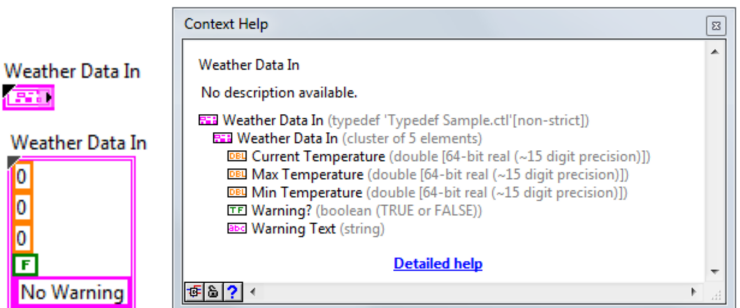


Plik .ctl może być kopią zwykłej kontrolki, ale może być również definicją typu.



Definicja typu na Block Diagram-ie

- Definicje typu są oznaczone przez trójkąt w rogu ikony
- Pomoc kontekstowa pokazuje zawartość typu danych



Czas na podsumowanie tej części wykładu!

Zamiast tekstu — seria pytań kontrolnych.

SPRAWDŹMY, CO JUŻ UMIEMY

Pytanie kontrolne

Masz dane: wartość napięcia (DBL), informację „pomiar poprawny” (Boolean) i opis kanału (String). Chcesz przesyłać je razem pomiędzy częściami programu. Najlepszy podstawowy wybór to:

(a) tablica DBL
(b) klaster
(c) pojedynczy String
(d) trzy zmienne globalne

cluster

Pytanie kontrolne

Masz dane: wartość napięcia (DBL), informację „pomiar poprawny” (Boolean) i opis kanału (String). Chcesz przesyłać je razem pomiędzy częściami programu. Najlepszy podstawowy wybór to:

(a) tablica DBL
(b) **klaster**
(c) pojedynczy String
(d) trzy zmienne globalne

cluster

Pytanie kontrolne

Chcesz odczytać z dużego klastra wyłącznie pole o nazwie `Valid`. Która funkcja jest najczytelniejsza?

(a) Unbundle by Name
(b) Build Array
(c) Number to String
(d) Bundle

cluster

Pytanie kontrolne

Chcesz odczytać z dużego klastra wyłącznie pole o nazwie `Valid`. Która funkcja jest najczytelniejsza?

(a) **Unbundle by Name**
(b) Build Array
(c) Number to String
(d) Bundle

cluster

Pytanie kontrolne

Do wejścia funkcji oczekującego DBL podłączono wartość I32 i pojawił się Coercion Dot. Co oznacza ta kropka?

(a) błąd składni i brak możliwości uruchomienia VI
(b) automatyczną konwersję typu danych
(c) utworzenie zmiennej globalnej
(d) przerwanie wykonywania dataflow

typy danych

Pytanie kontrolne

Do wejścia funkcji oczekującego DBL podłączono wartość I32 i pojawił się Coercion Dot. Co oznacza ta kropka?

(a) błąd składni i brak możliwości uruchomienia VI
(b) **automatyczną konwersję typu danych**
(c) utworzenie zmiennej globalnej
(d) przerwanie wykonywania dataflow

typy danych

Pytanie kontrolne

Chcesz zbudować przycisk „Wykonaj pomiar”, który ma dać pojedynczy impuls logiczny i wrócić do stanu FALSE po obsłużeniu. Która rodzina akcji mechanicznych jest najbardziej naturalna?

(a) Latch
(b) Switch
(c) Numeric
(d) Enum

Boolean

Pytanie kontrolne

Chcesz zbudować przycisk „Wykonaj pomiar”, który ma dać pojedynczy impuls logiczny i wrócić do stanu FALSE po obsłużeniu. Która rodzina akcji mechanicznych jest najbardziej naturalna?

(a) **Latch**
(b) Switch
(c) Numeric
(d) Enum

Boolean

Pytanie kontrolne

Przyrząd zwrócił przez komunikację tekst `"12.48"`. Chcesz użyć tej wartości w obliczeniach numerycznych. Czego potrzebujesz?

(a) konwersji String → Numeric
(b) zmiany Mechanical Action
(c) utworzenia klastra
(d) zmiany koloru przewodu

string

Pytanie kontrolne

Odczytano automatycznie z przyrządu tekst `"12.48"`. Chcesz użyć tej wartości w obliczeniach numerycznych. Czego potrzebujesz?

(a) konwersji String → Numeric
(b) zmiany Mechanical Action
(c) utworzenia klastra
(d) zmiany koloru przewodu

string

Pytanie kontrolne

Odczytano automatycznie z przyrządu tekst `"12.48"`. Chcesz użyć tej wartości w obliczeniach numerycznych. Czego potrzebujesz?

(a) **konwersji String → Numeric**
(b) zmiany Mechanical Action
(c) utworzenia klastra
(d) zmiany koloru przewodu

`string`

Pytanie kontrolne

Program ma trzy tryby: IDLE, MEASURE i ERROR. Który typ danych użyjesz do wyboru przez użytkownika trybu pracy?

(a) DBL
(b) String bez ograniczeń
(c) Enum
(d) Boolean

enum

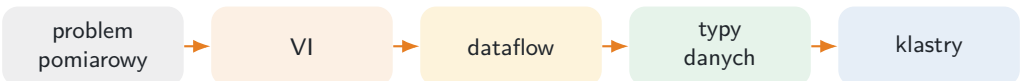
Pytanie kontrolne

Program ma trzy tryby: IDLE, MEASURE i ERROR. Który typ danych użyjesz do wyboru przez użytkownika trybu pracy?

(a) DBL
(b) String bez ograniczeń
(c) **Enum**
(d) Boolean

enum

Podsumowanie



Na następnym wykładzie: **tablice, TypeDef, Case, pętle i Event Structure**.