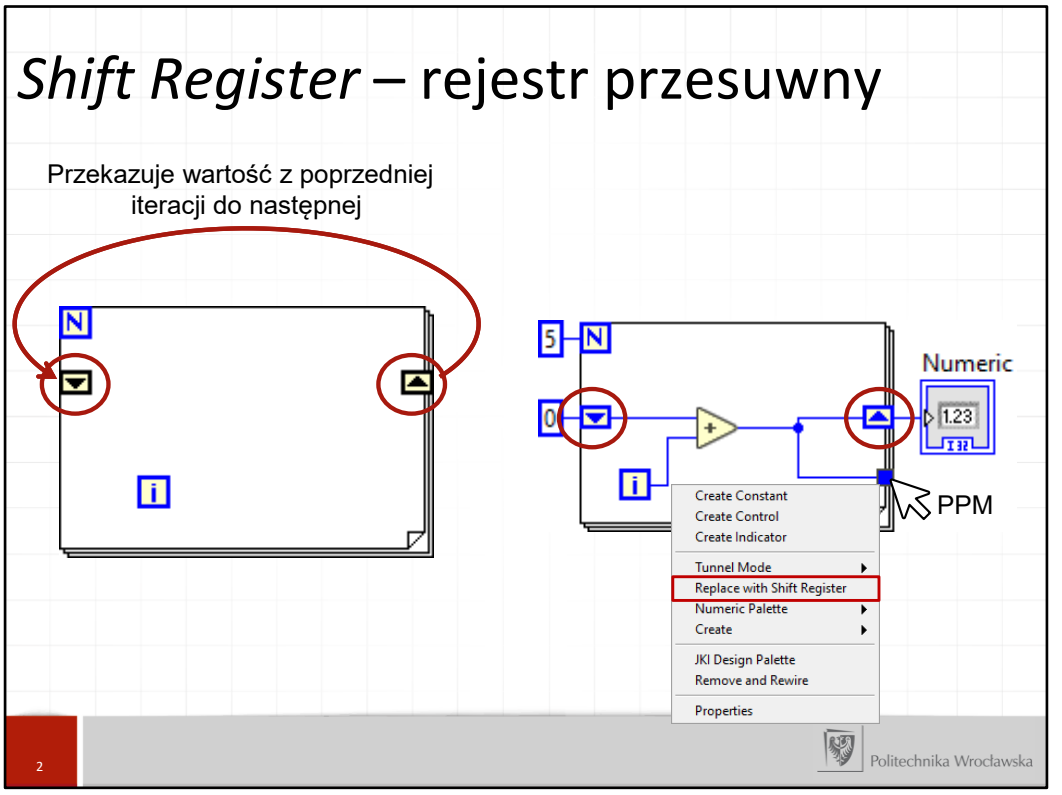
Wykład 3
15.03.2024 r.
Funkcje i struktury, cz. 2

Adrian Kaim

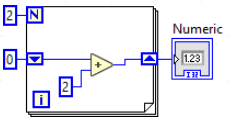
Politechnika Wrocławska



Shift Register (rejestr przesuwny) jest to rodzaj tunelu, pozwalający na przekazywanie informacji pomiędzy kolejnymi iteracjami pętli. Aby go stworzyć, należy kliknąć prawym przyciskiem myszy na zwykły tunel, a następnie wybrać opcję *Replace with Shift Register.*

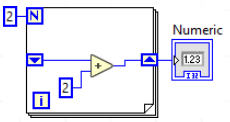
Inicjalizacja rejestru przesuwnego

Rejestr zainicjalizowany



	I uruchomienie		II uruchomienie	
	I iteracja	II iteracja	I iteracja	II iteracja
SR_{in}	0	2	0	2
SR_{out}	2	4	2	4
Numeric	-	4	-	4

Rejestr niezainicjalizowany



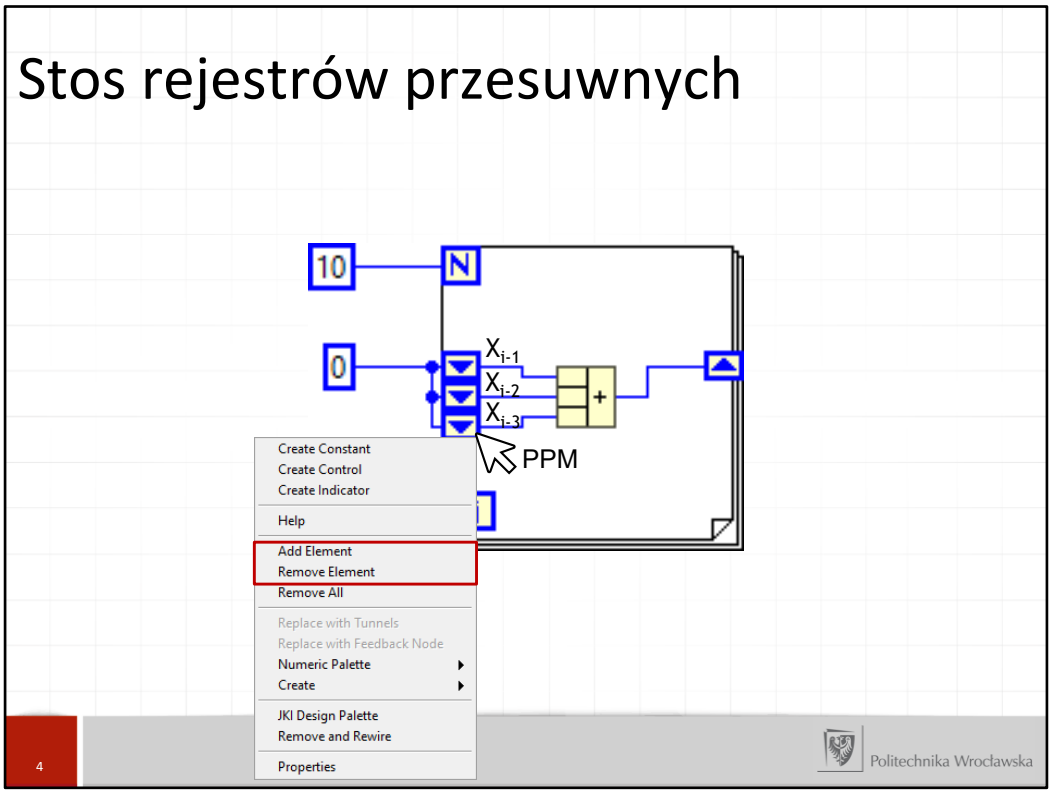
	I uruchomienie		II uruchomienie	
	I iteracja	II iteracja	I iteracja	II iteracja
SR_{in}	0	2	4	6
SR_{out}	2	4	6	8
Numeric	-	4	-	8

3

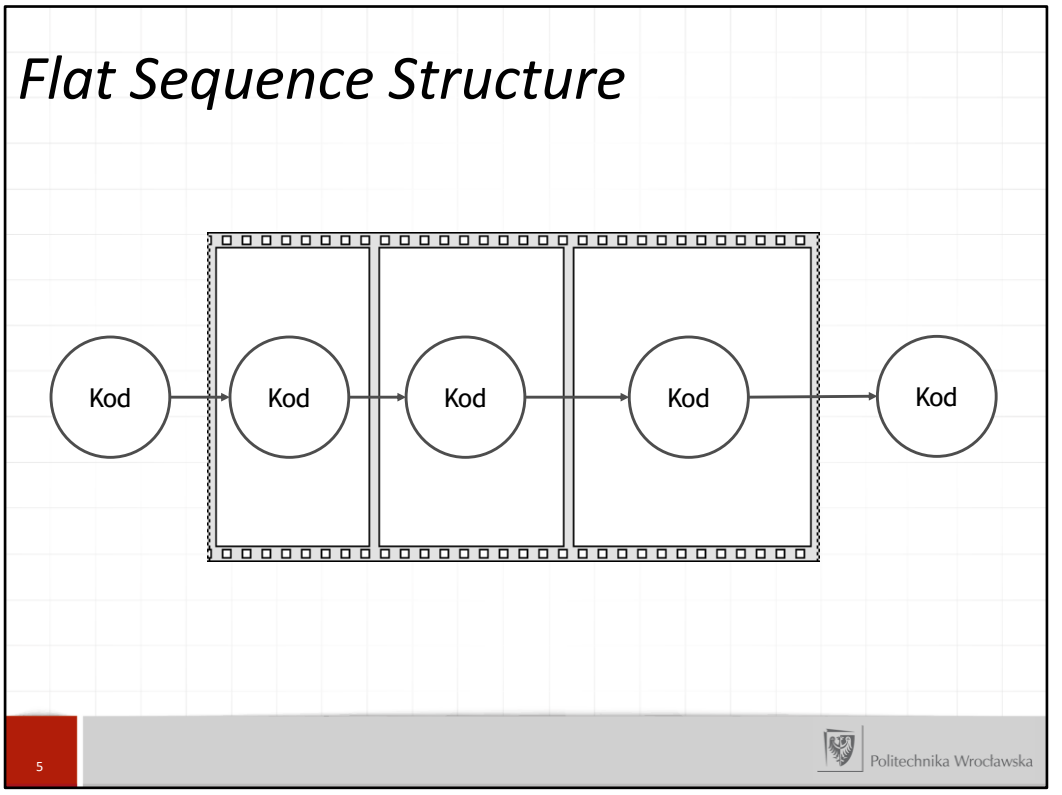
Politechnika Wrocławska

W zależności od potrzeb, rejestr przesuwny może zostać zainicjalizowany, choć nie jest to wymagane.

1. W przypadku gdy rejestr jest inicjalizowany wartością początkową, to wartość ta jest dostępna w pierwszej iteracji pętli. Po wykonaniu kodu zawartego w strukturze, do rejestru wpisywana jest nowa wartość, która to będzie dostępna w następnej iteracji. Zachowanie to jest powtarzanie aż do zakończenia działania pętli, a ostatnia wartość, którą wpisano do rejestru, przekazywana jest dalej, tak jak gdyby rejestr był zwykłym tunelem typu *Last Value*. Gdy pętla uruchomiona zostanie kolejny raz (przez nadrzędną pętlę, lub przez ponowne uruchomienie programu), rejestr zostanie zainicjalizowany od nowa i cały cykl się powtórzy.
2. W przypadku, gdy rejestr nie jest inicjalizowany, to w pierwszej iteracji dostępna jest domyślna wartość typu danych rejestru (np. 0 lub fałsz). Uruchamiając pętle ponownie, jako wartość początkowa, przyjęta zostanie ostatnia zapisana w pamięci podręcznej wartość rejestru.

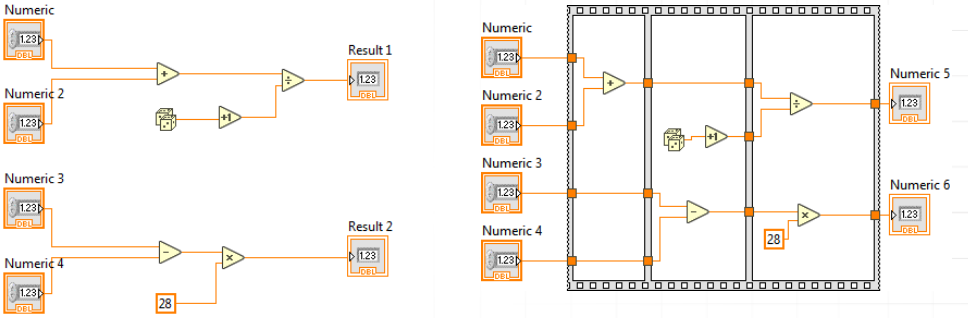


Możliwe jest stworzenie tzw. stosu rejestrów przesuwnych, który pozwala na odczytanie kilku poprzednich wartości, uzyskanych w kilku poprzednich iteracjach. Aby stworzyć stos, należy go rozciągnąć lub kliknąć prawym przyciskiem myszy i wybrać *Add Element*.



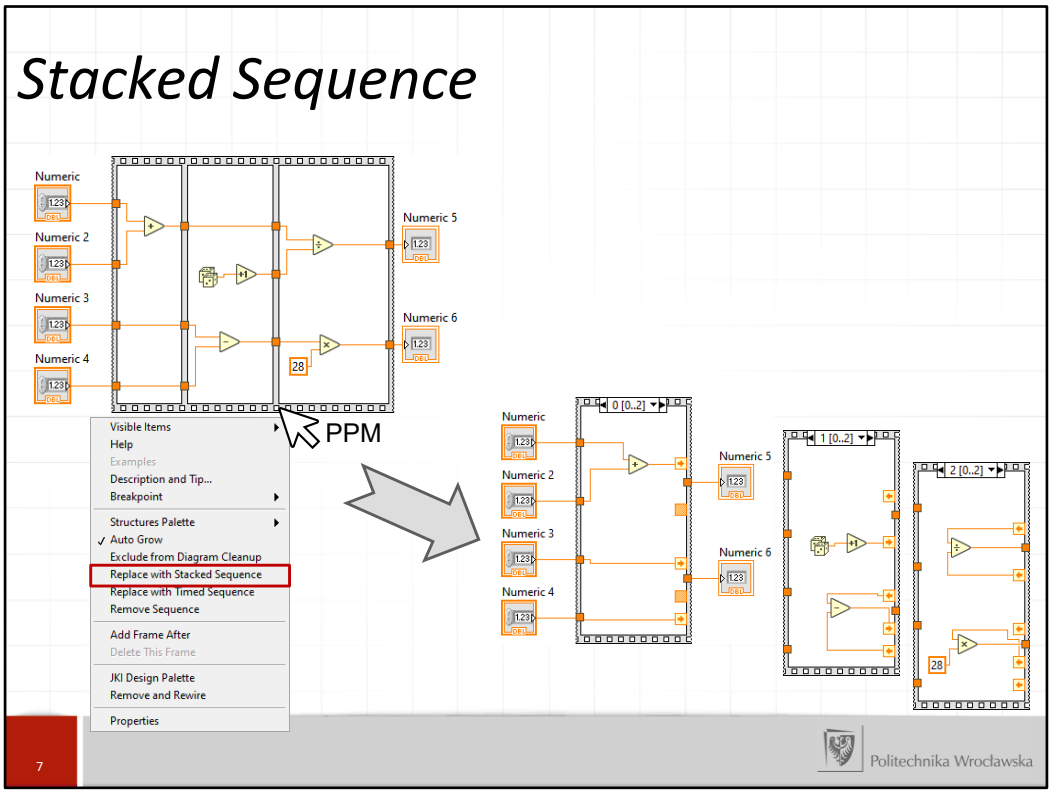
Do manipulowania kolejnością (oraz czasem wykonania) poszczególnych fragmentów kodu, wykorzystać można strukturę *Flat Sequence*. Struktura ta reprezentowana jest jako szereg klatek filmowych. W założeniu, w danym momencie wykonywany jest kod zawarty w jednej klatce, a dopiero po jego całkowitym zakończeniu następuje przejście do następnej klatki. Informacje pomiędzy kolejnymi klatkami przekazywane są tunelami, identycznymi jak w innych strukturach.

Flat Sequence Structure

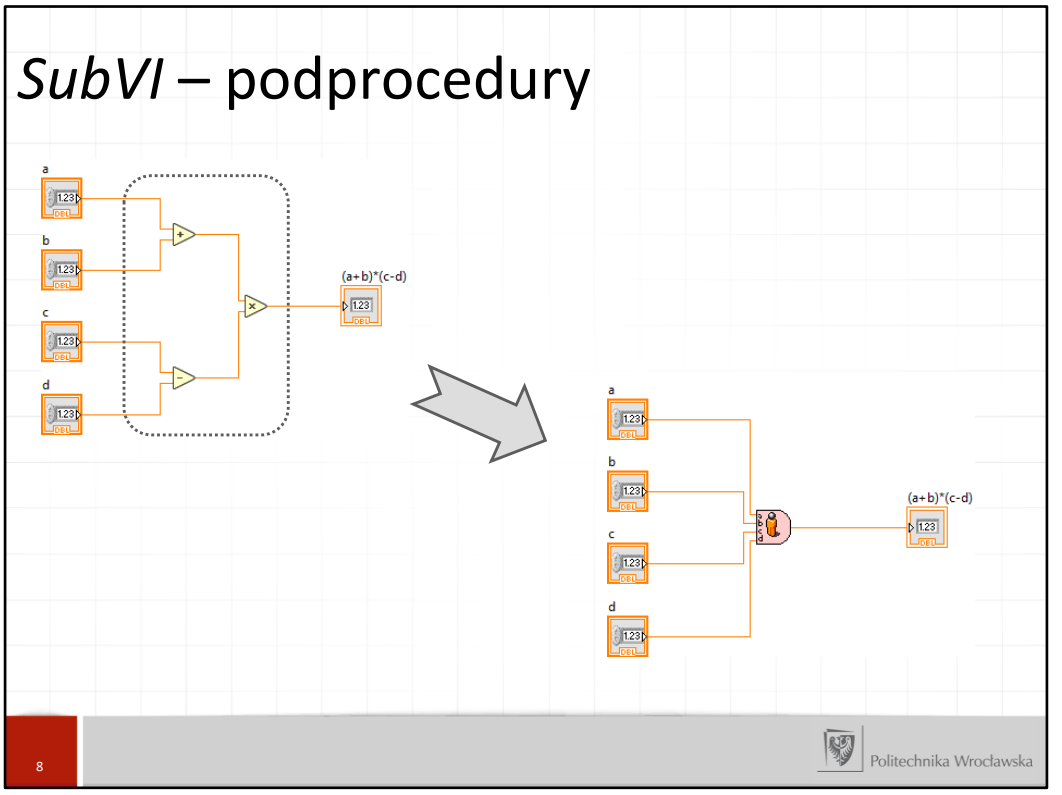


Pytanie 1 Która funkcja wykona się jako pierwsza w obydwu przypadkach?

Politechnika Wrocławska

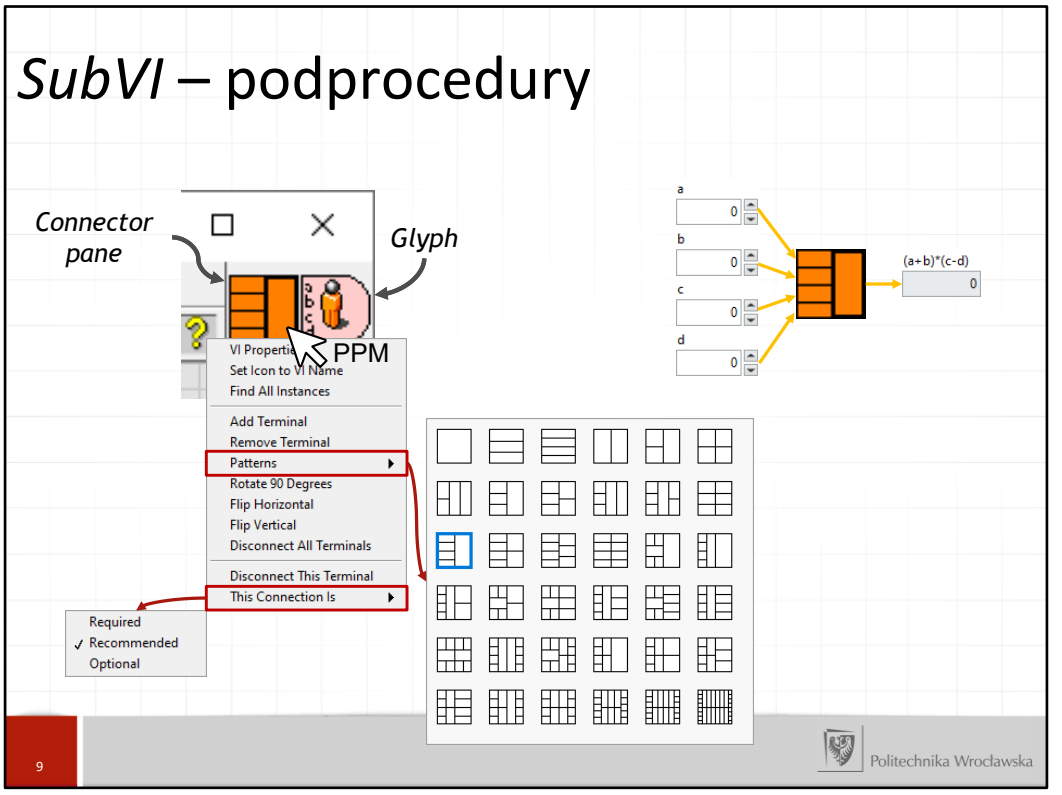


Strukturę *Flat Sequence* można zamienić w strukturę *Stacked Sequence*. Stworzony w ten sposób stos klatek pozwala zaoszczędzić miejsce na schemacie blokowym, co może pozytywnie wpłynąć na czytelność kodu. Podobnie jak w strukturze *Case*, na górnym boku umieszczony jest selektor, pozwalający na przełączanie widocznego fragmentu kodu.



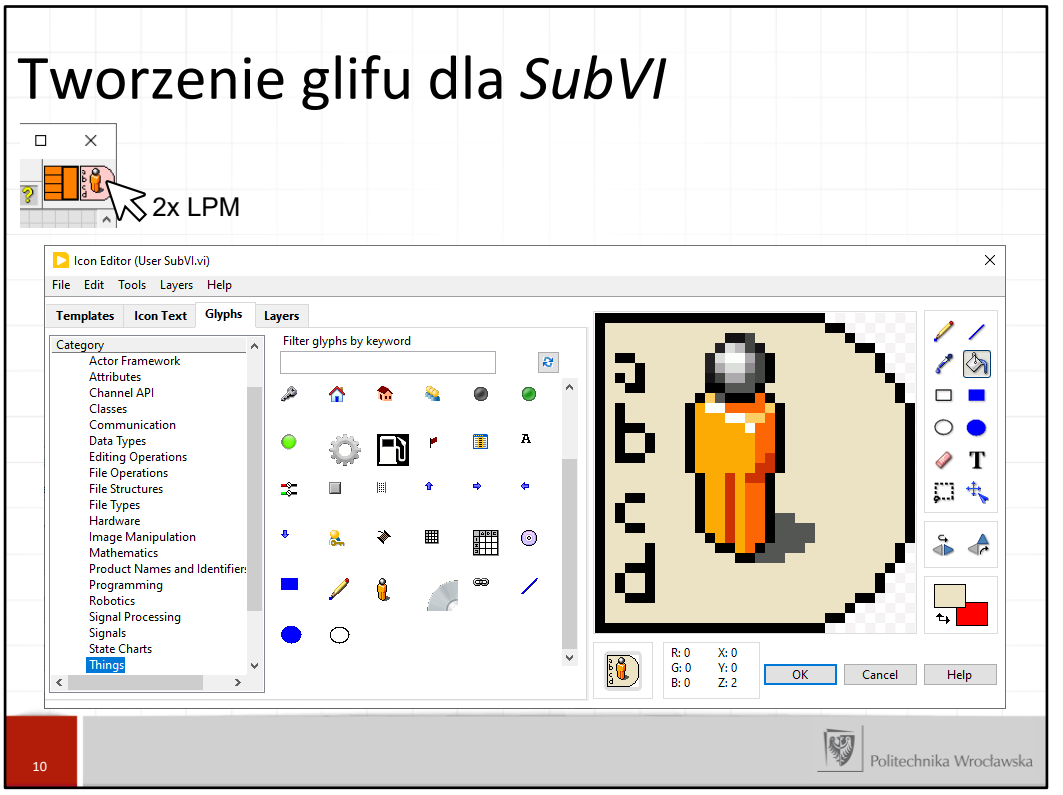
Każdy VI i każdy fragment kodu może zostać przekształcony w podprocedurę (*SubVI*). Kod taki staje się wówczas kolejnym węzłem na diagramie blokowym, i zachowuje się podobnie do funkcji dostępnych w palecie. Korzystanie z *SubVI* pozwala ograniczyć liczbę powtórzeń kodu, zmniejszyć (przestrzennie) kod, zwiększyć czytelność oraz ułatwia późniejsze modyfikacje. Wewnętrznie, każde *SubVI* posiada swój własny panel frontowy i schemat blokowy.

Aby przekształcić dany fragment kodu w *SubVI*, należy wybrać opcję *Edit>Create SubVI* po uprzednim zaznaczeniu tego fragmentu.



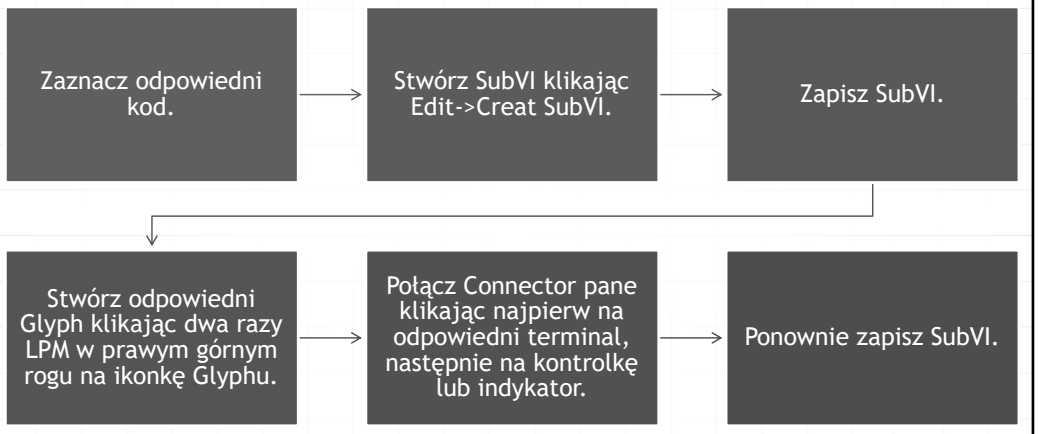
W prawym górnym rogu panelu frontowego każdego VI, także nowopowstałego *SubVI*, dostępne są dwa elementy pozwalające na określenie tego, w jaki sposób ma on współpracować z innymi *VI*.

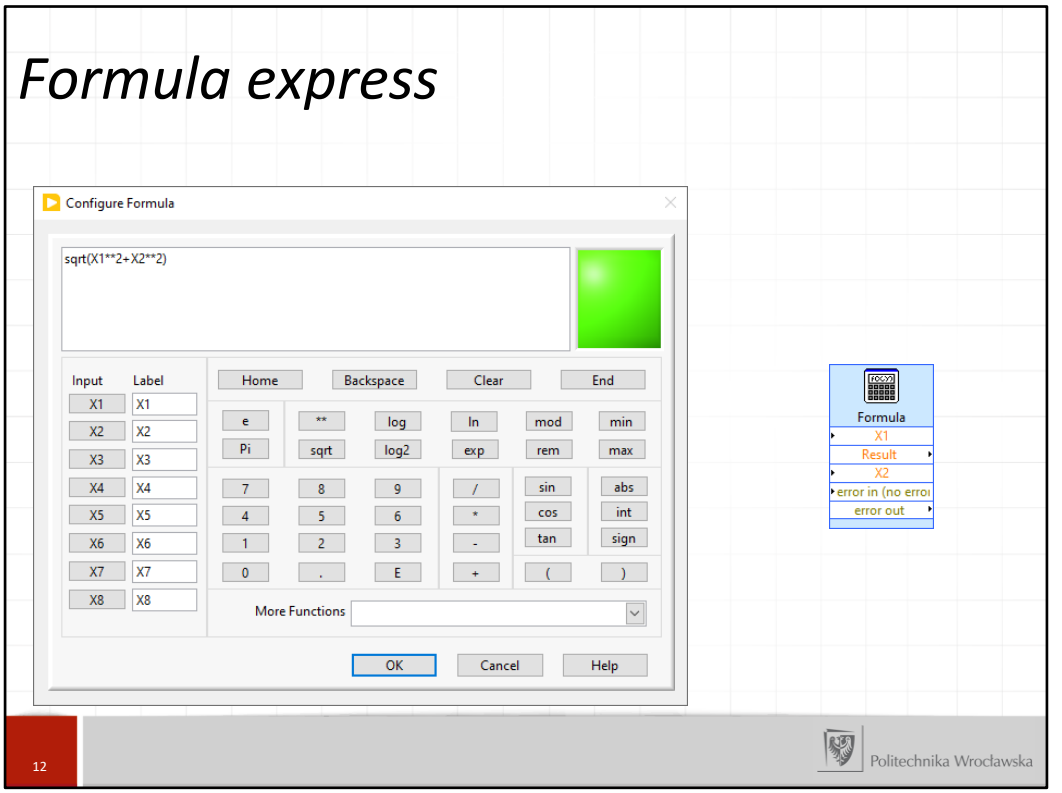
- *Connector pane* – panel określający sposób połączenia terminali dostępnych po umieszczeniu danego *SubVI* na schemacie blokowym. Terminale te umożliwiają wymianę informacji pomiędzy nadrzędnym programem i *SubVI*. Możliwa jest zmiana układu terminali oraz określenie czy dany terminal jest konieczny do poprawnego działania *SubVI*. Klikając lewym przyciskiem myszy na „pusty" (niepołączony) terminal, można powiązać go z dowolną kontrolką lub indykatorem na panelu frontowym.
- *Glyph* (glif) – ikona jaka widoczna będzie po umieszczeniu danego *SubVI* na schemacie blokowym. Glif powinien być zaprojektowany w taki sposób, aby symbolizował funkcjonalność i przeznaczenie *SubVI*.



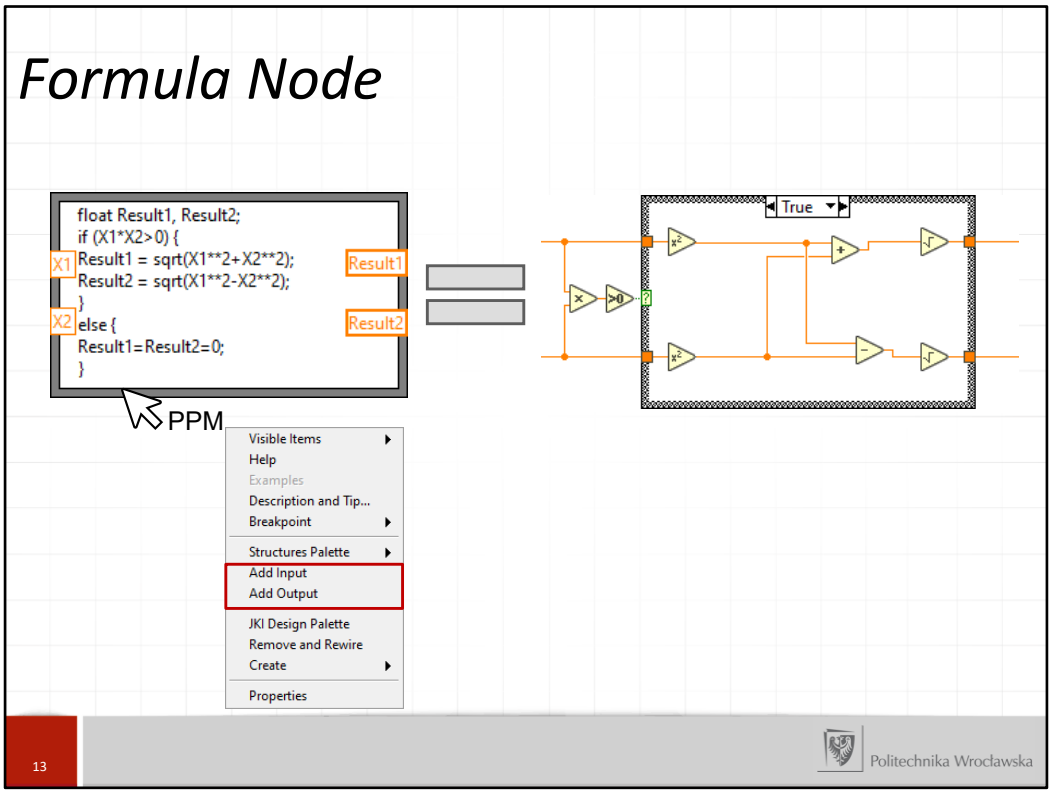
Dwukrotnie klikając na symbol glifu, znajdujący się w prawym górnym rogu panelu frontowego, uzyskuje się dostęp do edytora ikon. Pozwala on korzystać z szablonów, gotowych ikon, edytora tekstu oraz prostych narzędzi graficznych.

Tworzenie *SubVI*





Funkcja *Formula express*, to funkcja typu *Express VI*, pozwalająca na wykonywanie obliczeń matematycznych zdefiniowanych przy pomocy kalkulatora.



Struktura typu *Formula node* umożliwia wykonywanie prostego kodu o składni zbliżonej do C. W niektórych przypadkach ułatwia definiowanie operacji matematycznych i zwiększa ich czytelność. Klikając prawym przyciskiem myszy możliwe jest dodawanie dowolnej ilości terminali wejściowych i wyjściowych. Terminale wejściowe adaptują się automatycznie do typu przyłączonych danych, natomiast wyjściowe wymagają tekstowego zadeklarowania typu.

Więcej informacji na temat składni obsługiwanej przez *Formula node* dostępne jest w dokumentacji LabVIEW.

Zadanie 1

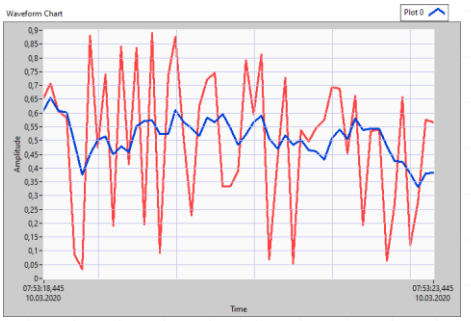
Napisz program do obliczania n!.

Zadanie 2

Stwórz SubVI do obliczania wartości n! na podstawie kodu z zadania 1. Wykorzystaj odpowiednie konektory oraz zaprojektuj odpowiednie glyphy.

Zadanie 3

Napisz program, który będzie losował liczbę z zakresu 0-1 oraz uśredniał 5 poprzednich losowań. Wyniki wyświetl na jednym wykresie typu *Waveform chart*. Pamiętaj o spowolnieniu pętli.



Politechnika Wrocławska

Zadanie 4

Wykorzystując Shift Register zasymuluj migającą diodę LED. Pozwól użytkownikowi wybrać czas świecenia diody.

Zadanie 5

Wykorzystując Formula node napisz program, który pozwoli na znalezienie współczynnika kierunkowego a oraz wyrazu wolnego b prostej $y = ax+b$ przechodzącej przez dwa zadanie punkty $P_1 = (x_1, y_1)$ oraz $P_2 = (x_2, y_2)$.

Zadanie 6

Napisz program do obliczenia:

$$\sum_{i=1}^{n} \frac{1}{(i+2)i!}$$

Podpowiedź: Wykorzystaj SubVI z zadania 2