Innovative Teaching Approaches in development of Software Designed Instrumentation and its application in real-time systems

Podstawy Projektowania Przyrządów Wirtualnych

Wykład 6: Wzorce programistyczne

Co-funded by the Erasmus+ Programme of the European Union

Powtórka z wykładu 5

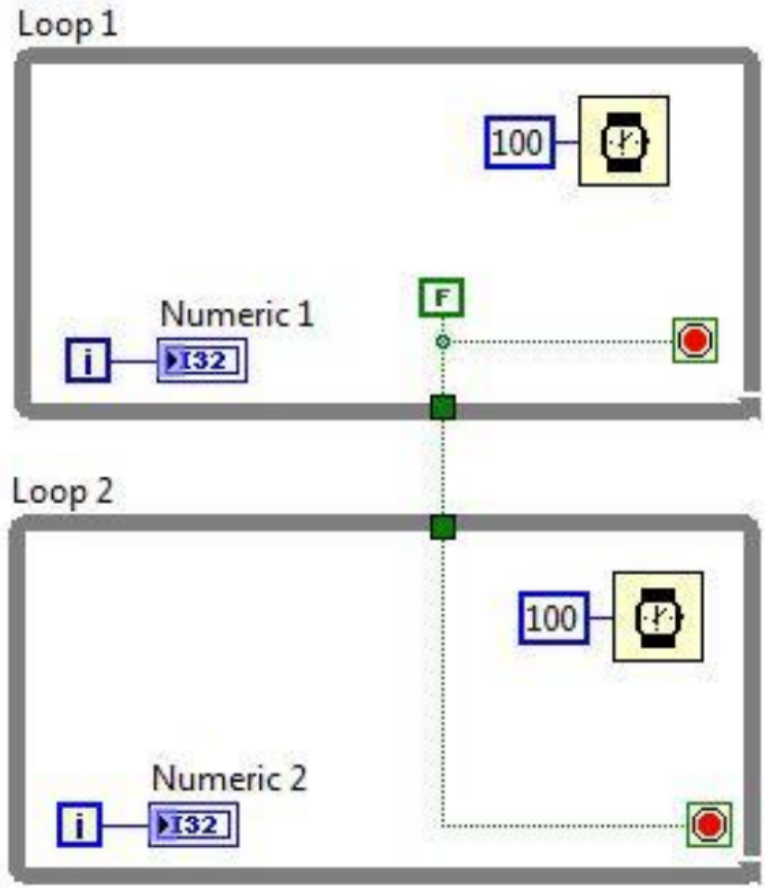
1/6 Jak będzie działał poniższy program?

a) Pętla 1 i 2 będą wykonywać się równolegle.

b) Oby dwie pętle uruchomią się tylko jeden raz.

c) Pętla 2 będzie wykonywać się po zatrzymaniu działania pętli 1.

d) Pętla 1 będzie działała w nieskończoność a pętla 2 nigdy się nie wykona.



Powtórka z wykładu 5

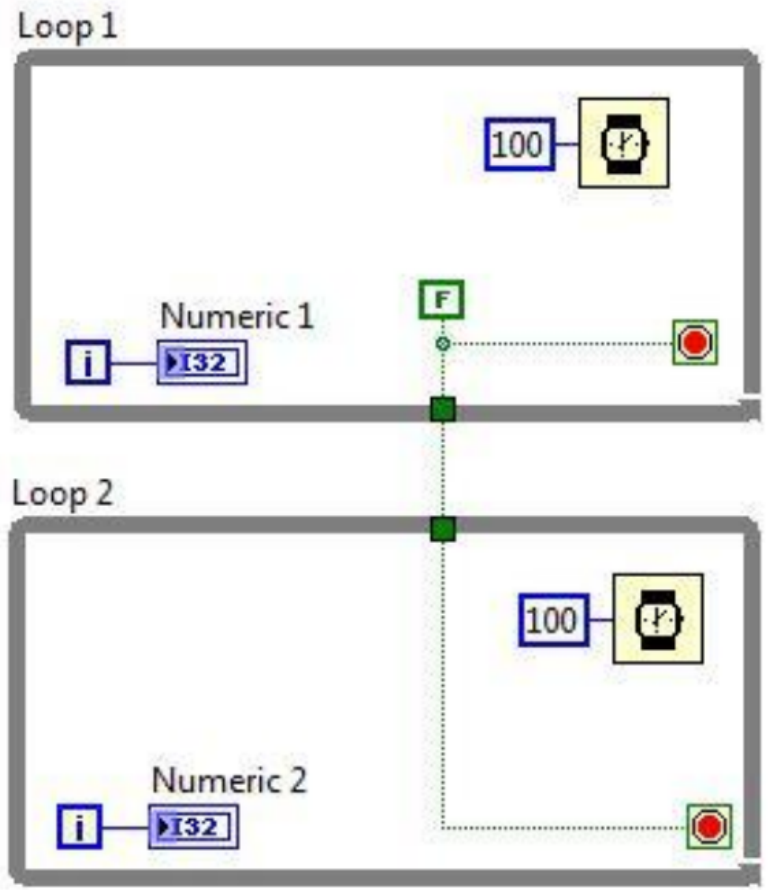
1/6 Jak będzie działał poniższy program?

a) Pętla 1 i 2 będą wykonywać się równolegle.

b) Oby dwie pętle uruchomią się tylko jeden raz.

c) Pętla 2 będzie wykonywać się po zatrzymaniu działania pętli 1.

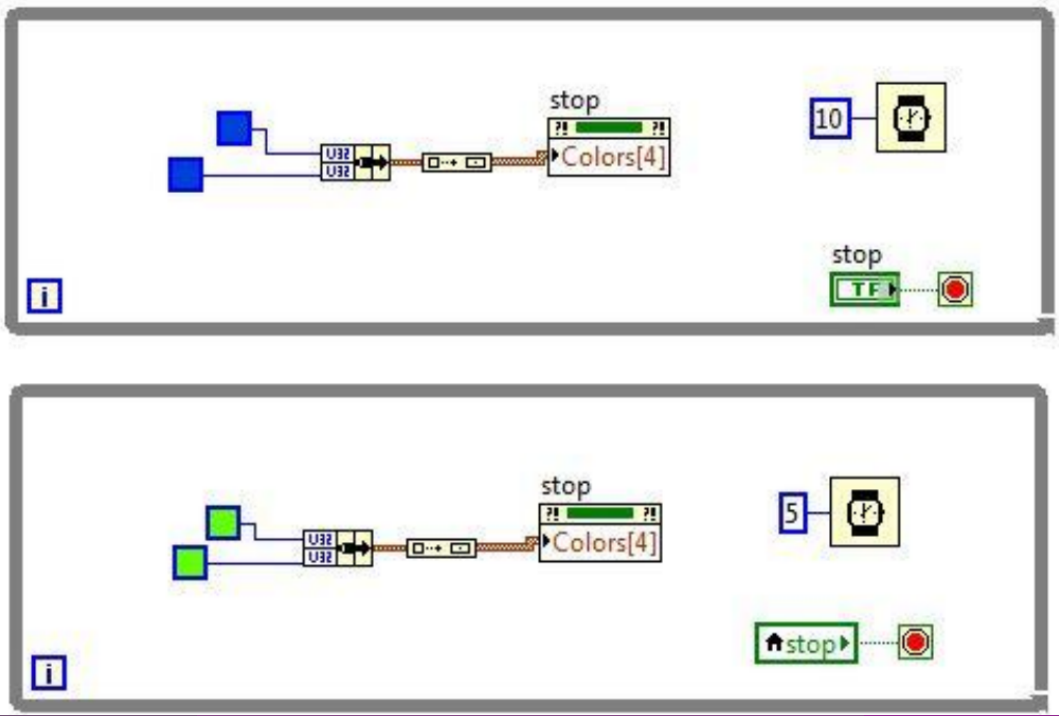
d) **Pętla 1 będzie działała w nieskończoność a pętla 2 nigdy się nie wykona.**



Powtórka z wykładu 5

2/6 Poniższy kawałek kodu wykonywał się 11 ms. Jaki kolor będzie miał przycisk stop po zakończeniu programu?

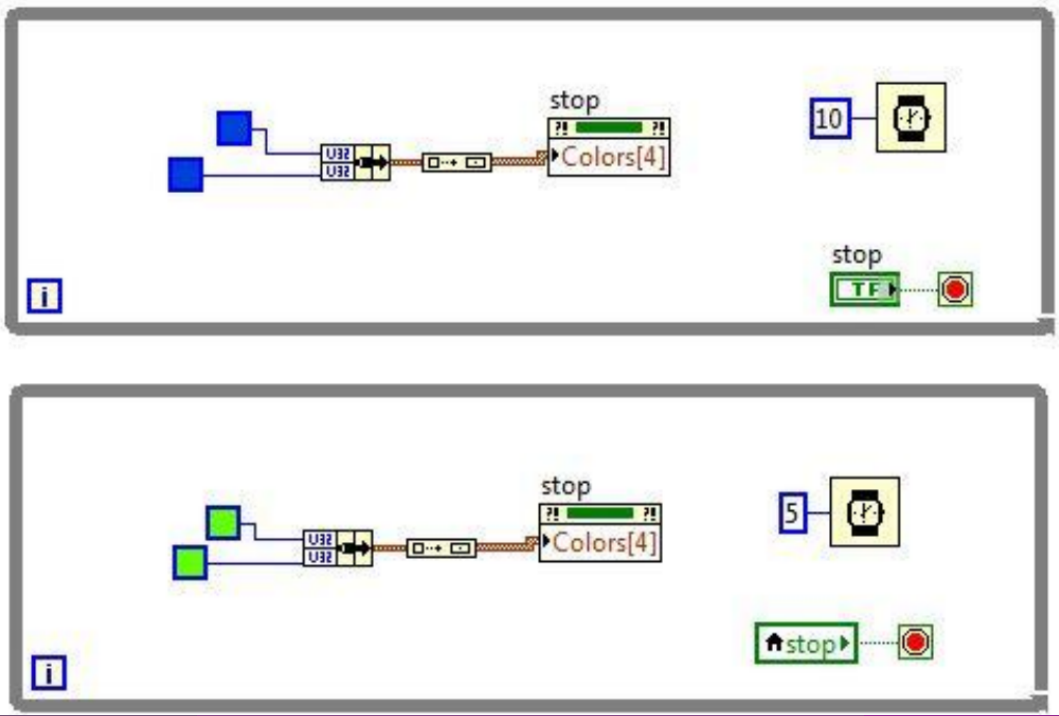
a) Niebieski
b) Zielony
c) Zielono-niebieski
d) Nie wiadomo



Powtórka z wykładu 5

2/6 Poniższy kawałek kodu wykonywał się 11 ms. Jaki kolor będzie miał przycisk stop po zakończeniu programu?

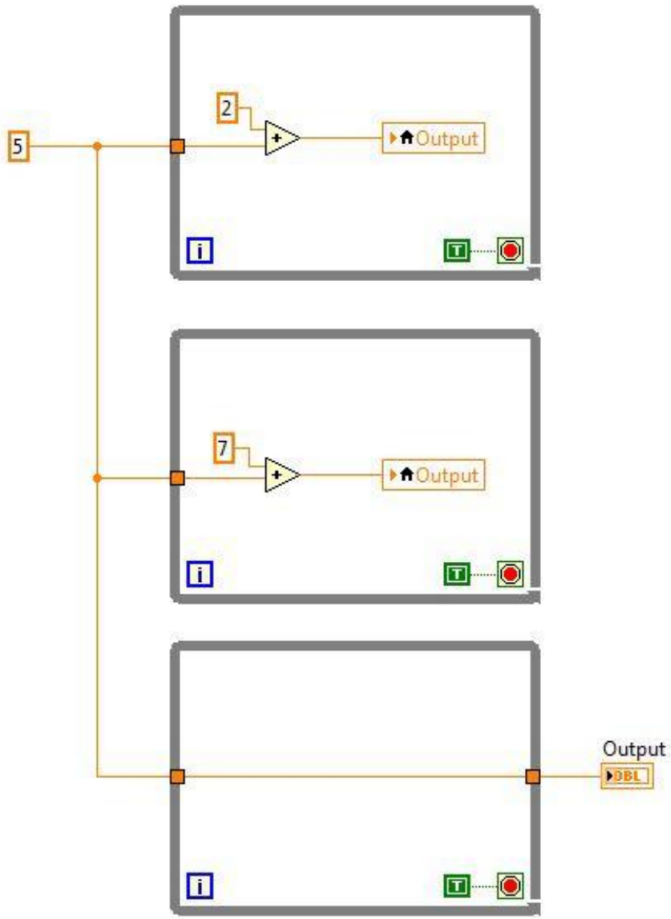
a) Niebieski
b) Zielony
c) Zielono-niebieski
d) **Nie wiadomo**



Powtórka z wykładu 5

3/6 Jaka wartość zostanie wyświetlona w *Output* po zakończeniu działania programu?

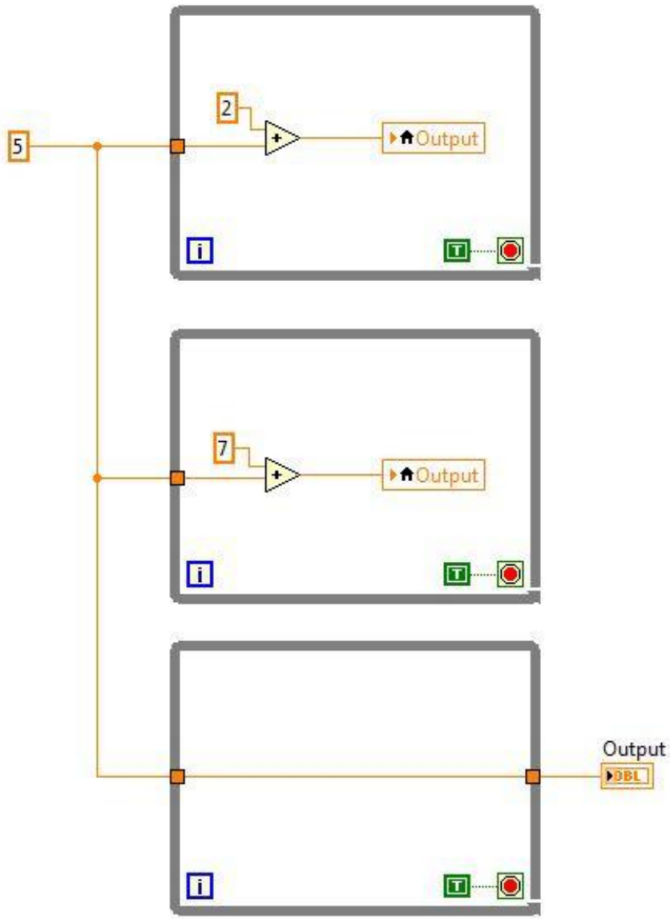
a) 5
b) 7
c) 12
d) Nie wiadomo



Powtórka z wykładu 5

3/6 Jaka wartość zostanie wyświetlona w *Output* po zakończeniu działania programu?

a) 5
b) 7
c) 12
d) **Nie wiadomo**



Powtórka z wykładu 5

4/6 Jeśli potrzebujesz zmienić programistycznie wartość w Numeric Control, jaka strategia będzie najlepsza?

a) Poprzez użycie zmiennej funkcyjnej.

b) Poprzez użycie zmiennej lokalnej.

c) Poprzez ustawienie wartości domyślnej.

d) Poprzez użycie referencji.

Powtórka z wykładu 5

4/6 Jeśli potrzebujesz zmienić programistycznie wartość w Numeric Control, jaka strategia będzie najlepsza?

a) Poprzez użycie zmiennej funkcyjnej.

b) **Poprzez użycie zmiennej lokalnej.**

c) Poprzez ustawienie wartości domyślnej.

d) Poprzez użycie referencji.

Powtórka z wykładu 5

5/6 Które zdanie poniższe przedstawia korzyści ze stosowania zmiennej globalnej w stosunku do zmiennej lokalnej?

a) Zmienna globalna może przekazywać dane pomiędzy dwoma działającymi równolegle VI.

b) Tylko zmienna globalna może przekazywać tablicę danych.

c) Zmienna globalna zachowuje model przepływu danych i nie może dojść do *race conditions*.

d) Zmienna globalna nie potrzebuje własnych *label* do działania poprawnie.

Powtórka z wykładu 5

5/6 Które zdanie poniższe przedstawia korzyści ze stosowania zmiennej globalnej w stosunku do zmiennej lokalnej?

a) **Zmienna globalna może przekazywać dane pomiędzy dwoma działającymi równolegle VI.**

b) Tylko zmienna globalna może przekazywać tablicę danych.

c) Zmienna globalna zachowuje model przepływu danych i nie może dojść do *race conditions*.

d) Zmienna globalna nie potrzebuje własnych *label* do działania poprawnie.

Powtórka z wykładu 5

6/6 Który element nie jest ważnym elementem przy tworzeniu globalnej zmiennej funkcyjnej?

a) Rejestr przesuwny do przechowywania danych.

b) Zmiana sposobu działania VI na *Reentrant*.

c) Ustawienie VI jako *inline*.

d) Ustawienie pętli While w taki sposób aby wykonywała się tylko raz.

Powtórka z wykładu 5

6/6 Który element nie jest ważnym elementem przy tworzeniu globalnej zmiennej funkcyjnej?

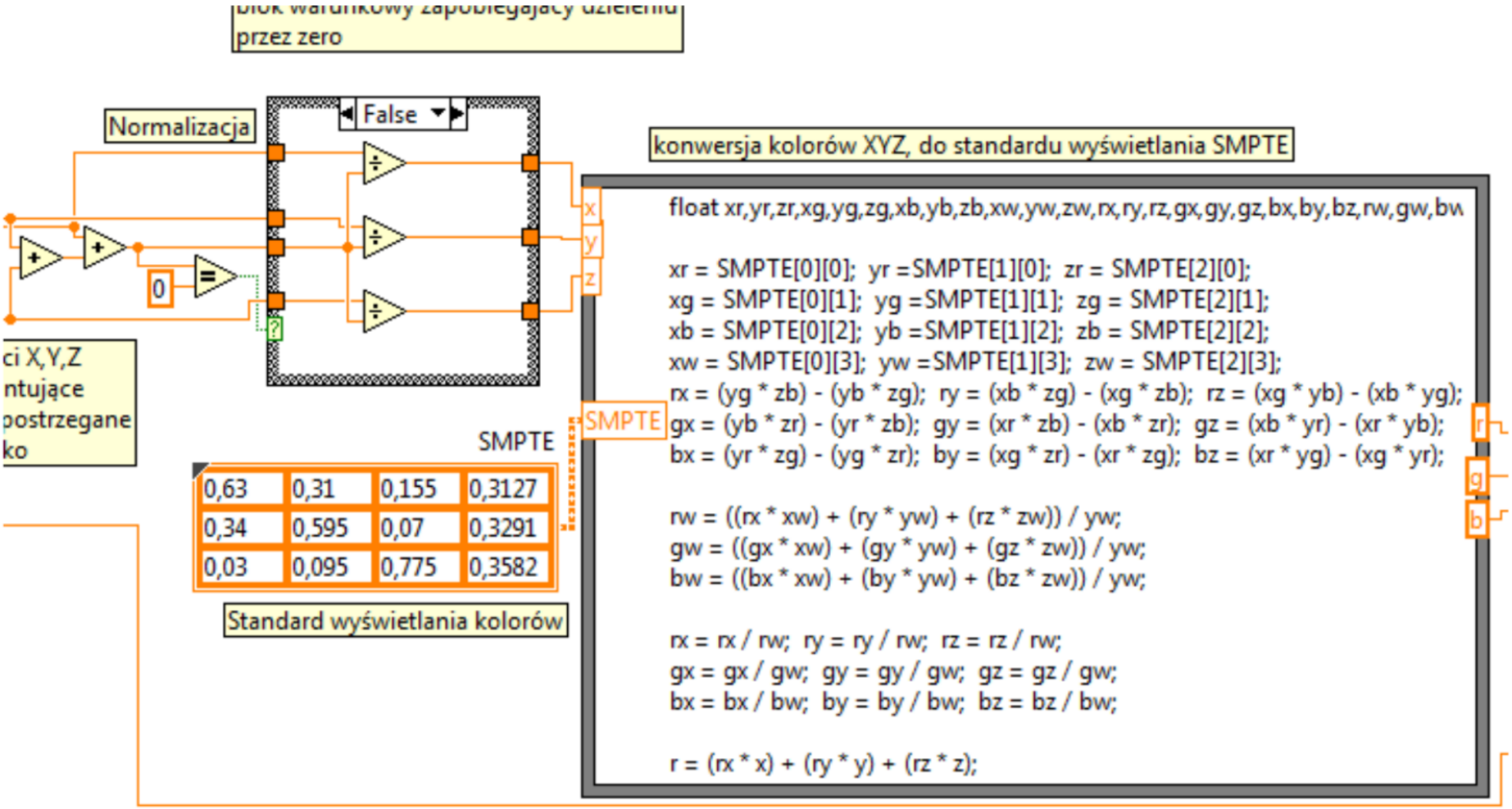
a) Rejestr przesuwny do przechowywania danych.

b) Zmiana sposobu działania VI na *Reentrant*.

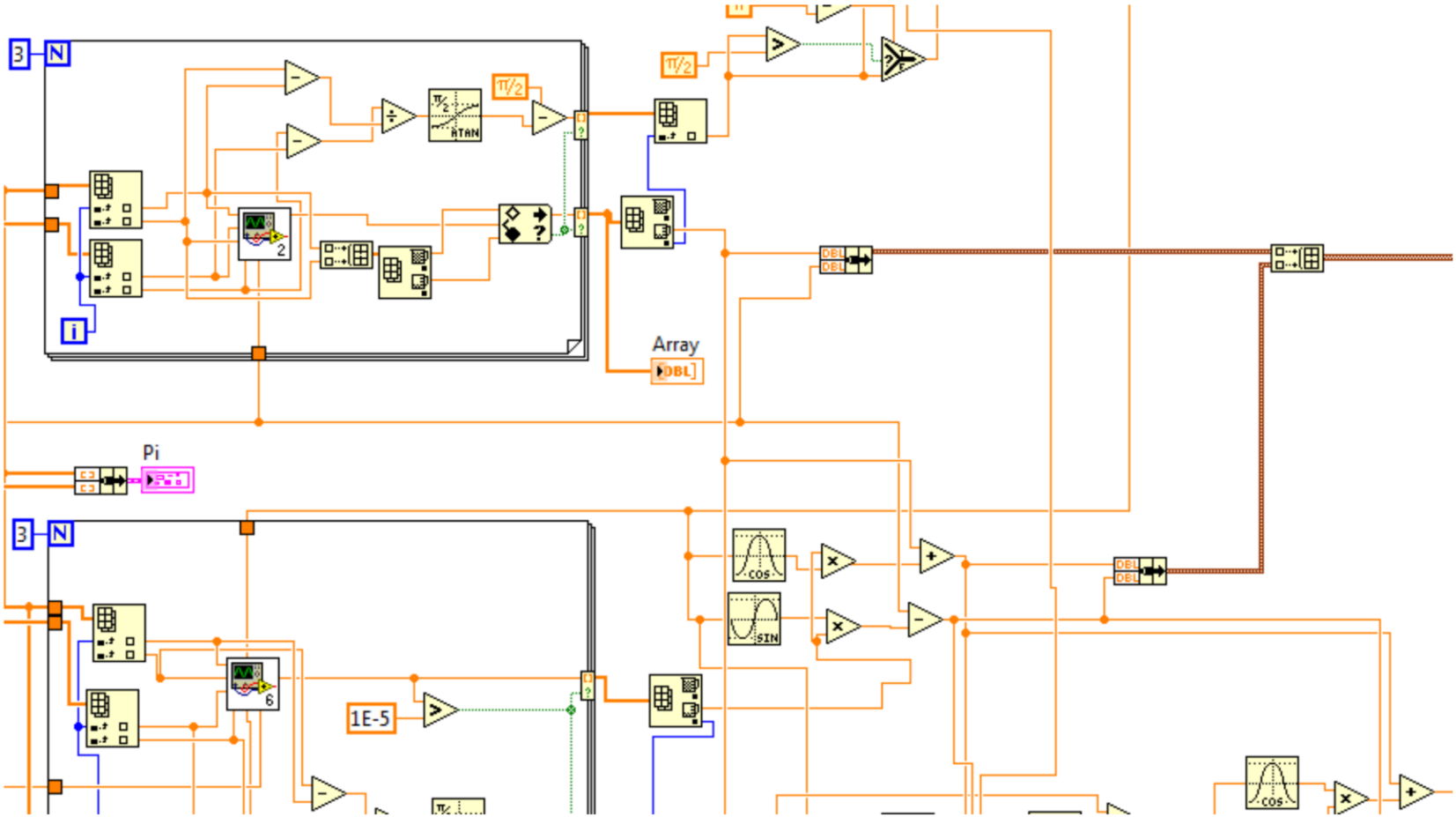
c) **Ustawienie VI jako *inline*.**

d) Ustawienie pętli While w taki sposób aby wykonywała się tylko raz.

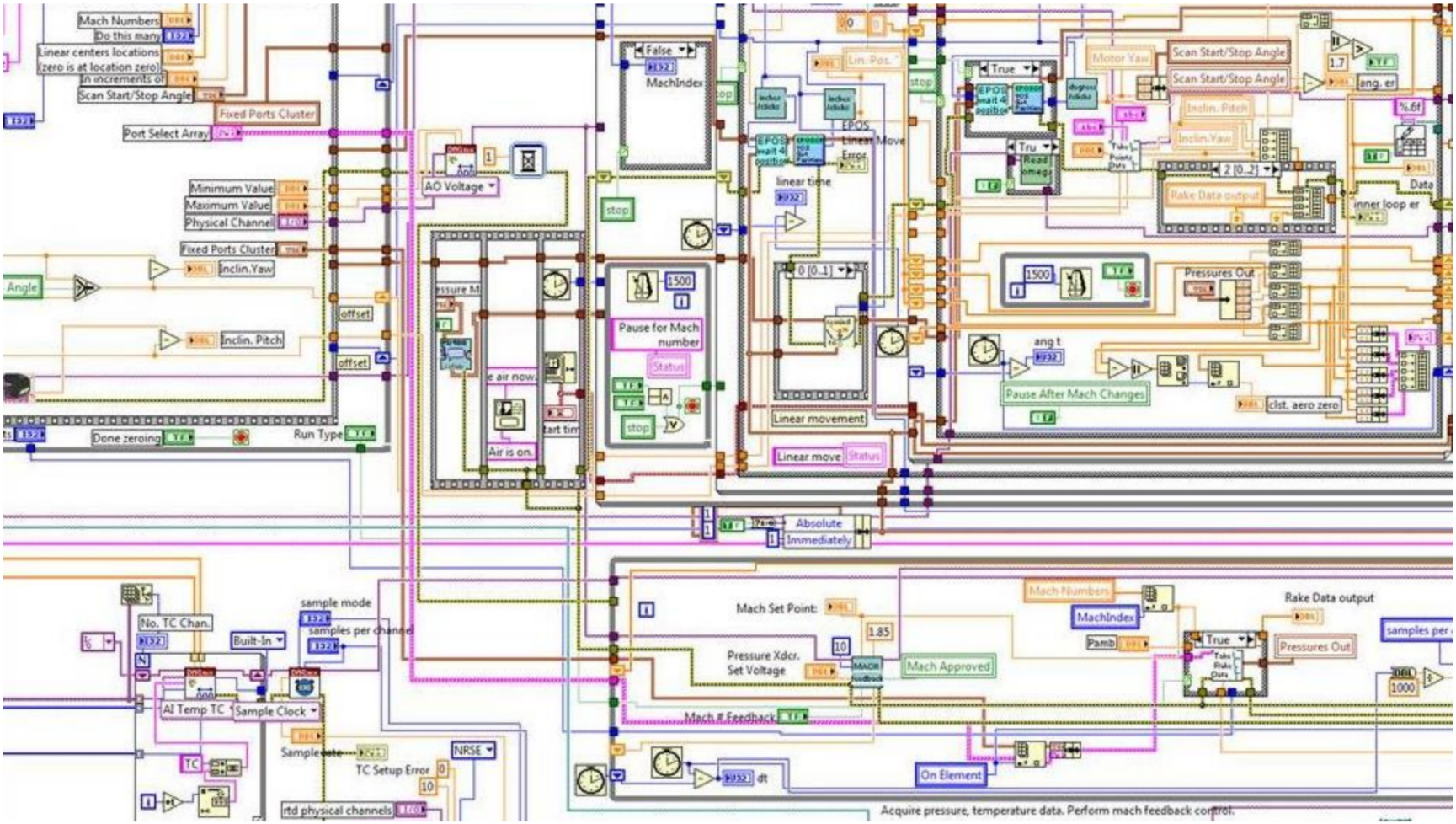
Przykłady złych praktyk



Przykłady złych praktyk → *Spaghetti code*



Przykłady złych praktyk → *Spaghetti code*



Dobre praktyki

Dobrze stworzony program powinien być:

- modułowy
- Łatwy w utrzymaniu
- skalowalny
- czytelny

Modułowość

- Każdy z modułów w programie powinien wykonywać określoną czynność.
- Moduł powinien stanowić odrębną całość i zależeć od innych modułów na tyle na ile to jest konieczne.
- Połączenia pomiędzy modułami powinny być przemyślane.

Skalowalność

- Wszędzie gdzie to ma sens należy używać definicji typów.
- Złożone typy danych skalują się lepiej od typów prostych.
- Program należy pisać tak, aby dodanie dodatkowej funkcji nie wymagało przebudowania programu.

Czytelność

- Program powinien być tak napisany aby był czytelny zarówno dla autora jak i innych programistów.
- Zrozumiała funkcja danych SubVI.
- Przemyślane nazwy SubVI, kontrolek, stałych.
- Symboliczne ikony SubVI.
- Dokumentacja kodu.

Ogólne wzkazówki

- Prosty, czytelny interfejs użytkownika
- Programy pisane od lewej do prawej
- Program powinien mieścić się w jednym oknie, ewentualnie rozszerzać się tylko w jednym kierunku.
- Schludne prowadzenie połączeń.

Wzorce projektowe

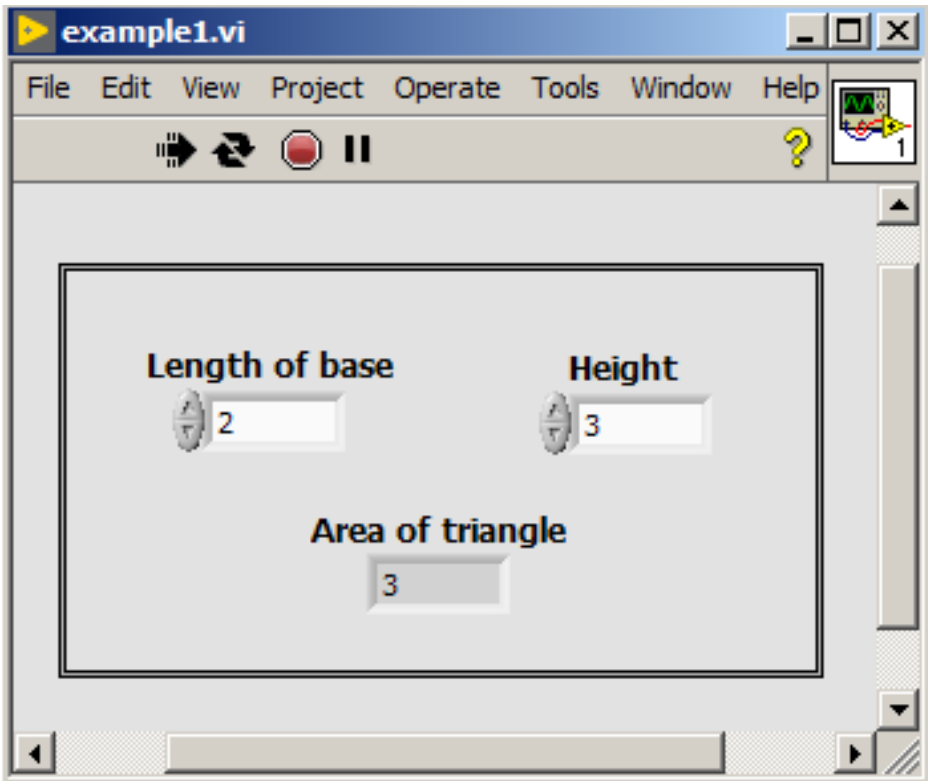
- **Wzorzec programistyczny** to sposób implementowania kodu programu mający na celu rozwiązanie konkretnych problemów programistycznych.
- **Dlaczego warto stosować wzorce?**
  - Gotowe rozwiązania - szablony zapewniające optymalną architekturę programu,
  - Spełniają kryteria dobrego programowania (skalowalne, czytelne, modułowe).

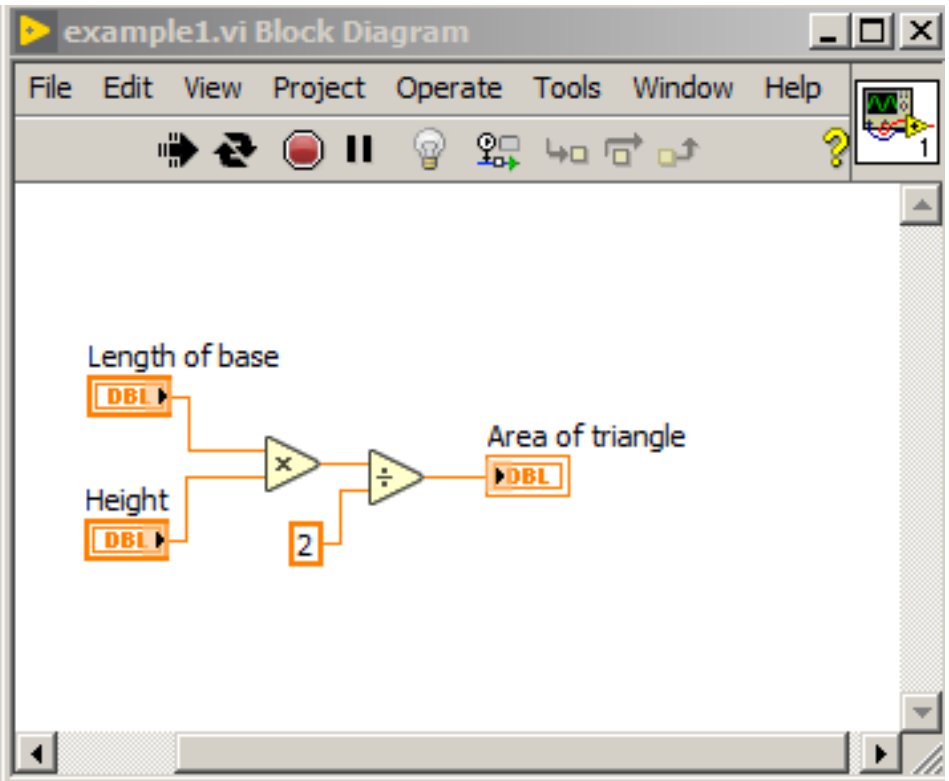
Podstawowe wzorce

- Do podstawowych wzorców zalicza się:
  - Simple VI Pattern
  - General VI
  - State Machine
  - Event-Based State Machine

Simple VI Pattern

- Stosowany domyślnie przez każdego początkującego programistę LabVIEW.
- Charakteryzuje się tworzeniem kodu od lewej strony do prawej bez dodatkowych struktur:

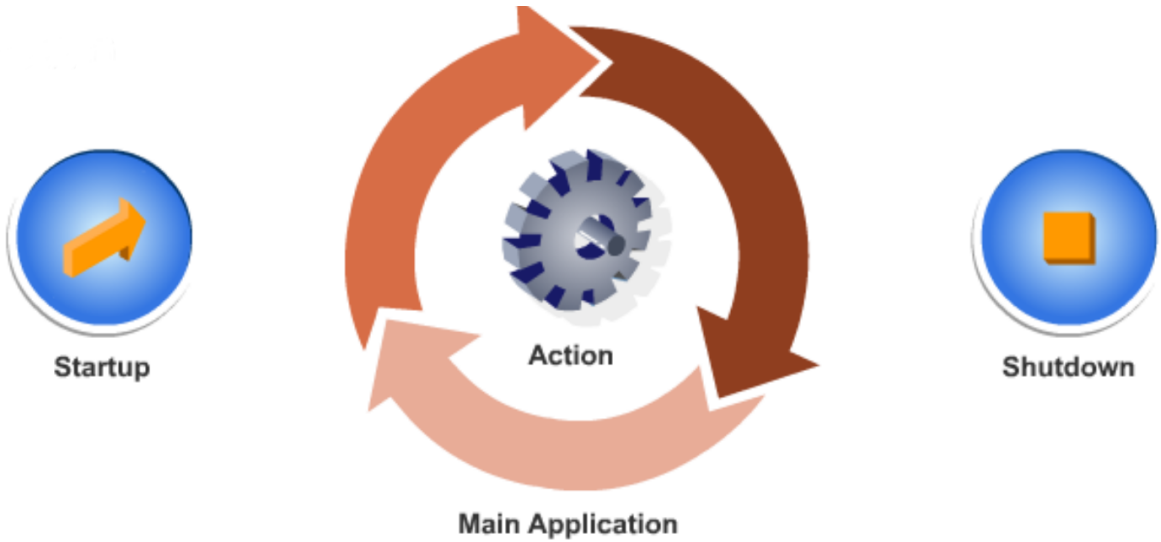




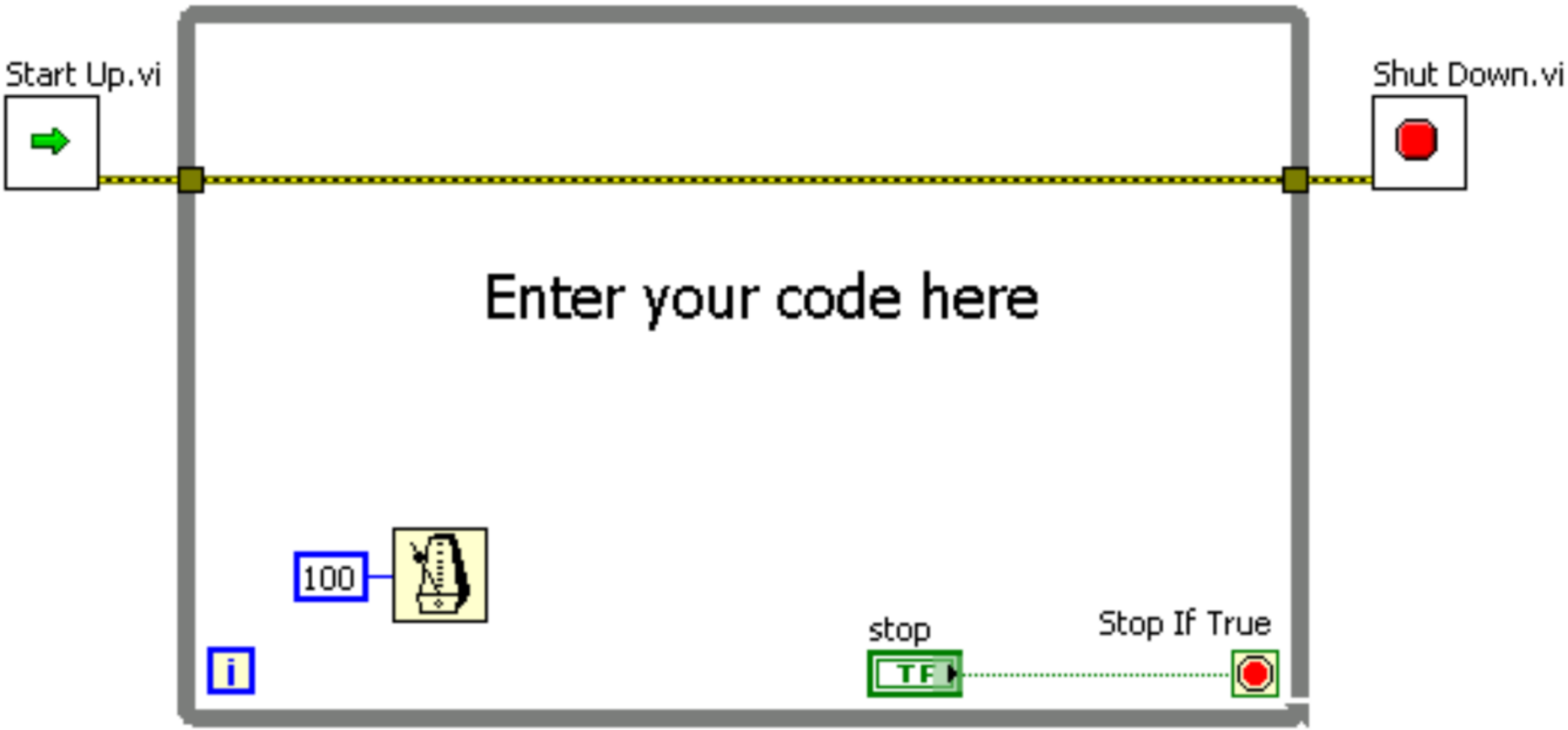
General VI Pattern

General VI pattern składa się z trzech kroków:

- **Start-up** - polecenia wykonywane na początku np. otwieranie pliku, inicjalizacja indykatorów/kontrolek wartościami domyślnymi,
- **Main application** - główny rdzeń programu, polecenia wykonywane w pętli while,
- **Shut-down** - procedury końcowe np. zamknięcie referencji do pliku.



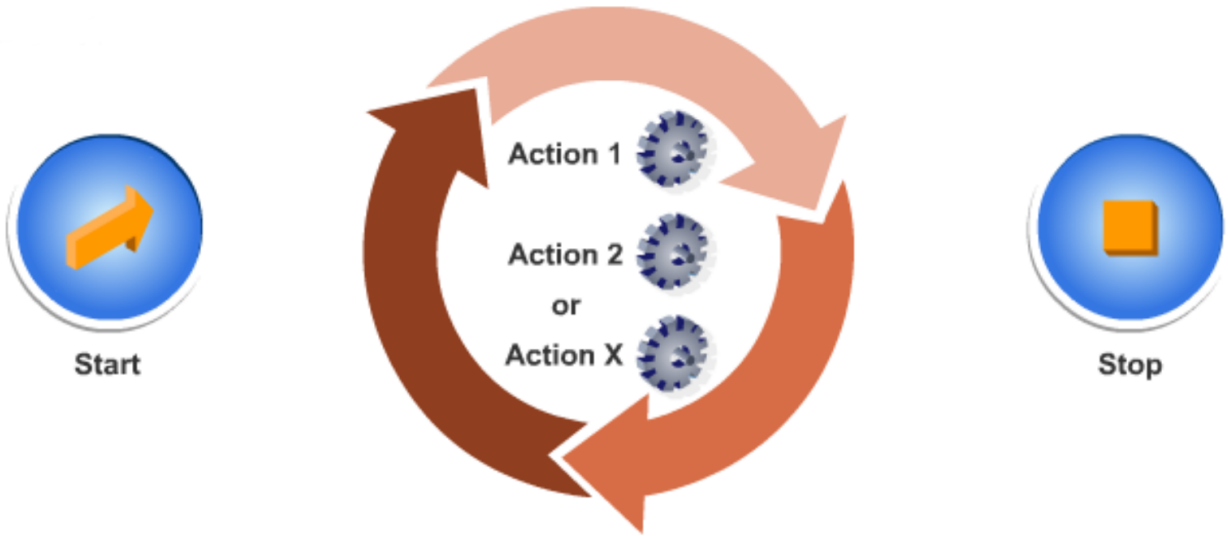
General VI Pattern



General VI oparty na strukturze zdarzeniowej

Trzy etapy programu:

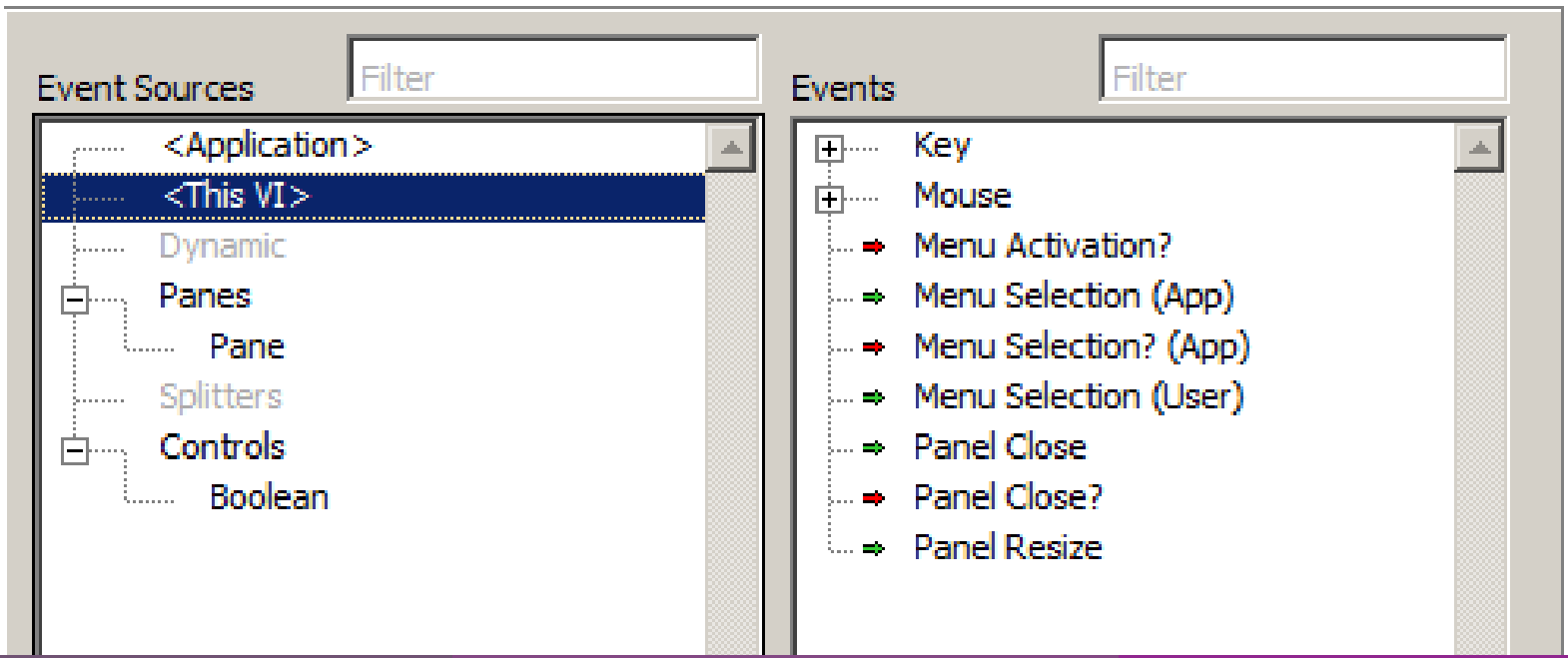
- **Start-up** - polecenia wykonywane na początku np. otwieranie pliku, inicjalizacja indykatorów/kontrolek wartościami domyślnymi,
- **Response to events** - główny rdzeń programu, polecenia wykonywane są po wystąpieniu zdarzenia,
- **Shut-down** - procedury końcowe np. zamknięcie referencji do pliku.



Struktura zdarzeniowa

Zdarzenia dzielimy na:

- Notify Events (zielone strzałki) - po wystąpieniu zdarzenia zostanie wykonana odpowiednia ramka struktury zdarzeniowej,
- Filter Events (czerwone strzałki) - domyślna akcja dla danego zdarzenia się nie wykona lecz zostanie wywołana dana ramka struktury zdarzeniowej np. nie zostanie zamknięty VI po naciśnięciu "X" lecz zostanie wywołany kod z struktury zdarzeniowej.



Dlaczego struktura zdarzeniowa?

- Każde zdarzenie powoduje wywołanie prostego, unikatowego fragmentu kodu.
- Pozwala na tworzenie oprogramowania nastawionego na interakcję z użytkownikiem.
- Łatwa w rozbudowie.

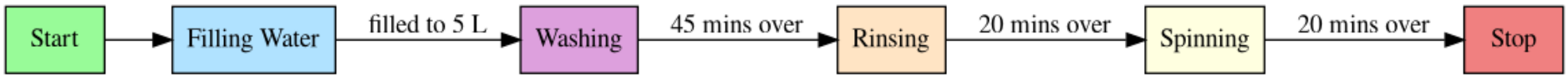
Zalecenia przy użyciu struktury zdarzeniowej

- Struktury zdarzeniowej nie stosuje się poza pętlą.
- W jednej pętli co najwyżej jedna struktura Event.
- Nie ustawiamy tego samego zdarzenia w dwóch różnych strukturach Event.
- Najlepiej ograniczyć się do jednej struktury Event w programie.
- Kod reagujący na zdarzenie powinien być krótki.

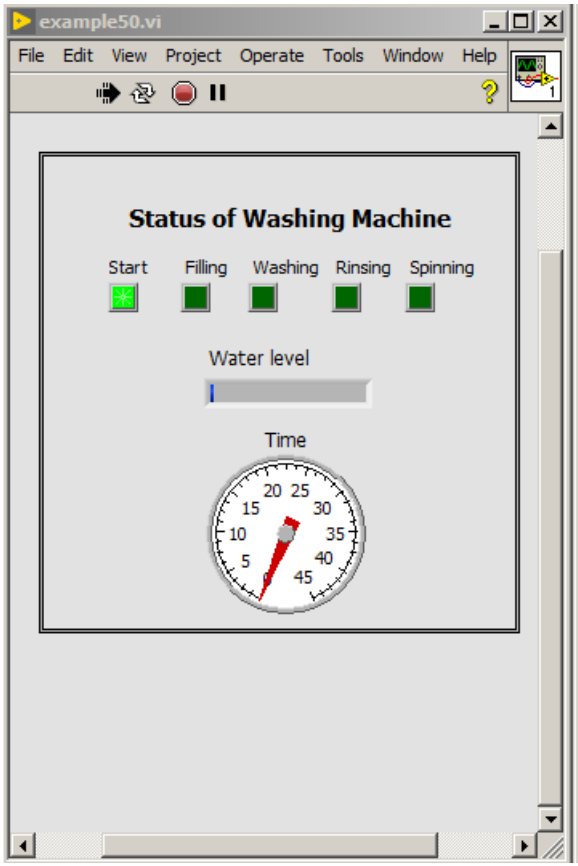
Maszyna stanów

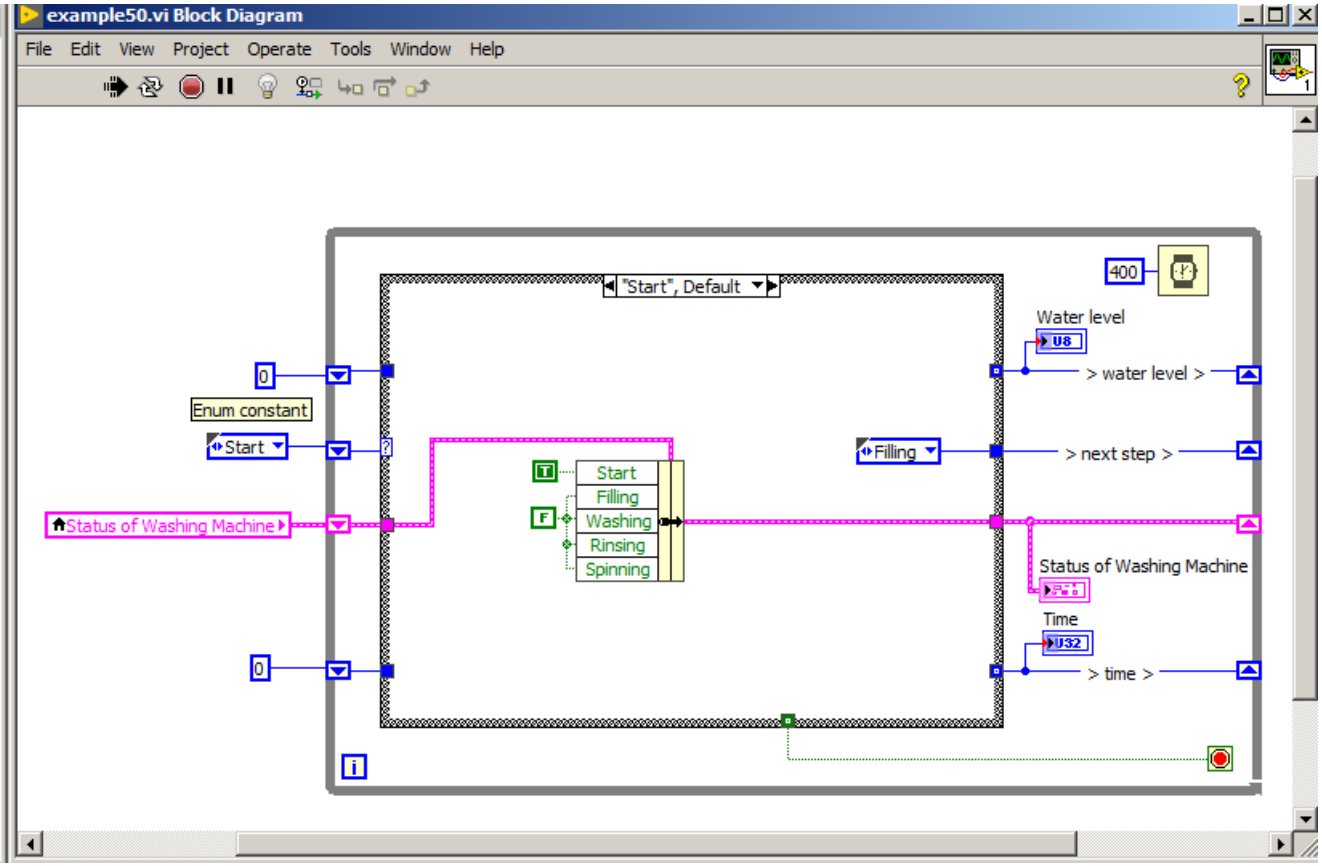
- Maszyna stanów składa się z pętli while, struktury case z podłączonym typem wyliczeniowym do case selector oraz rejestru przesuwnego.
- Poszczególne stany są umieszczone w osobnych ramkach struktury case.
- Listę wszystkich stanów zawiera enum a za pomocą rejestru przesuwnego jest przekazywana kolejna wartość z enum na wejście case selector.
- Zastosowanie maszyny stanów pozwala kontrolować wykonywanie się programu krok po kroku.
- Poszczególne kroki są podzielone co sprawia redukcję przestrzeni na block diagramie oraz przejrzystość programu.

Przykład wykorzystania maszyny stanów - symulator pralki

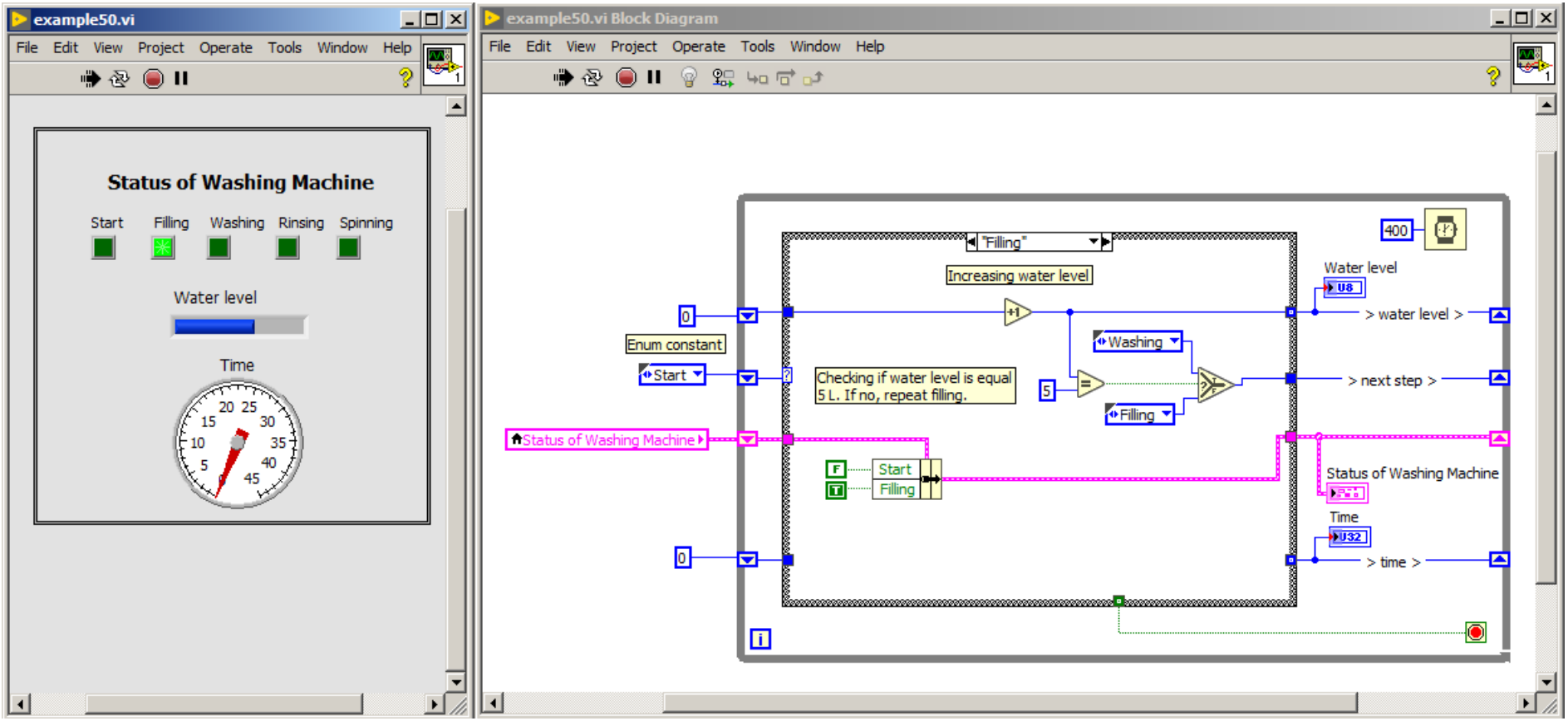


Przykład wykorzystania maszyny stanów - symulator pralki

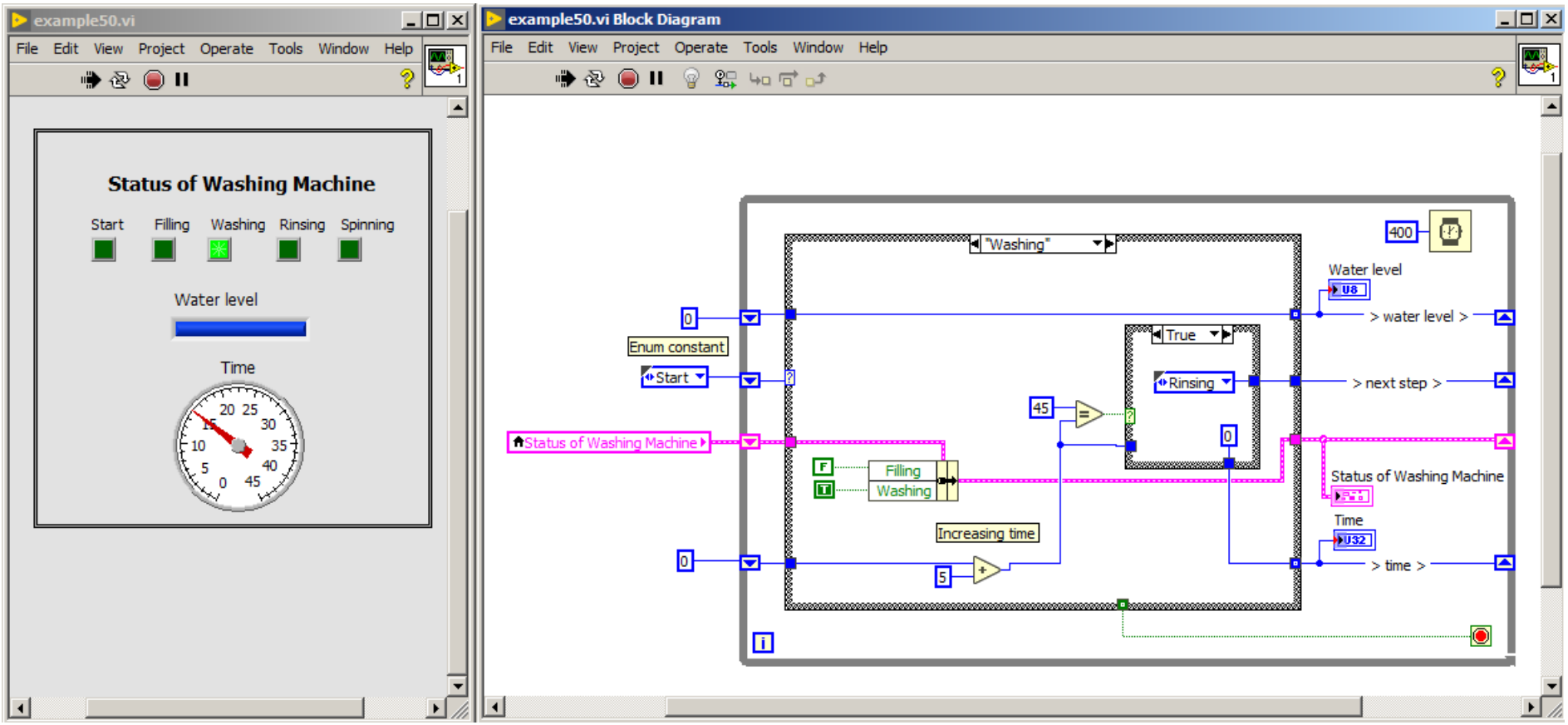




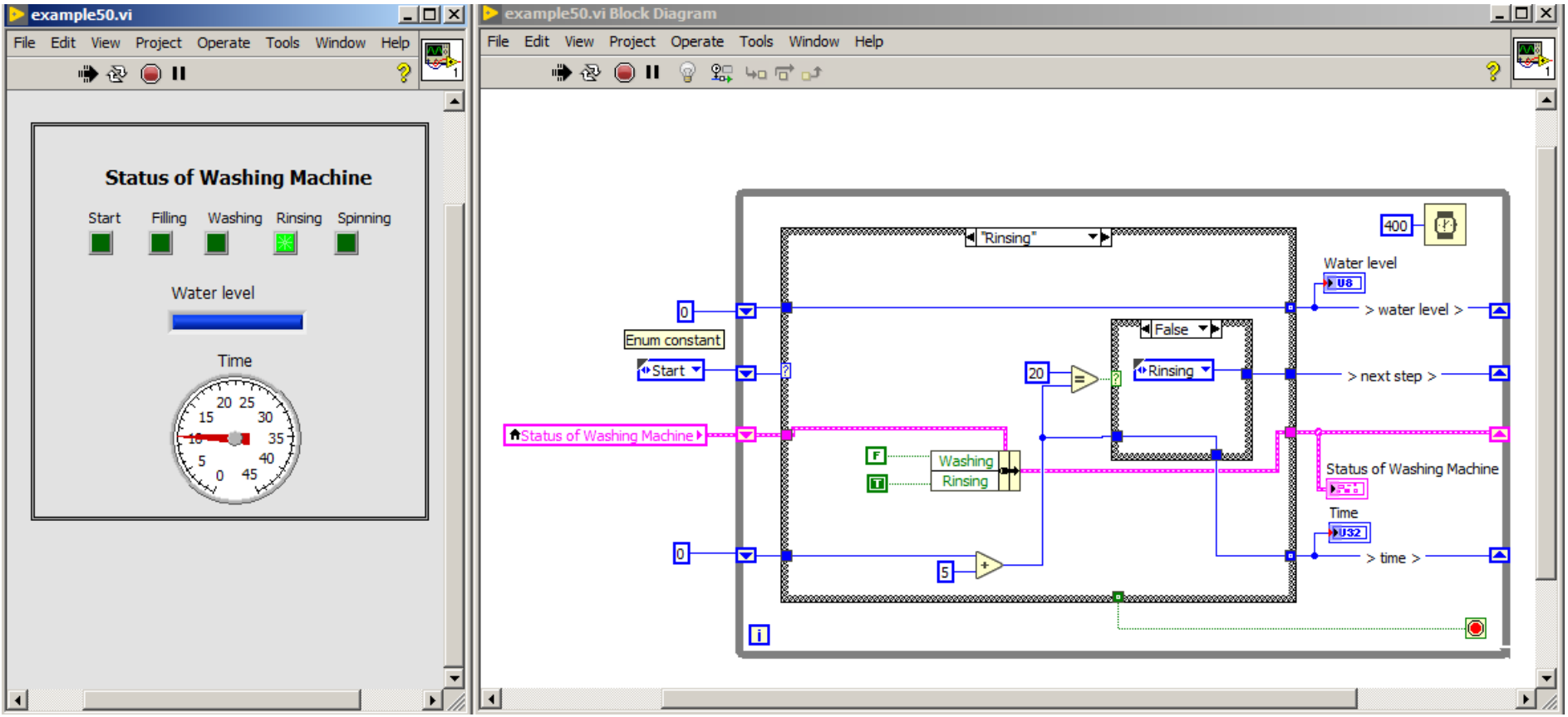
Przykład wykorzystania maszyny stanów - symulator pralki



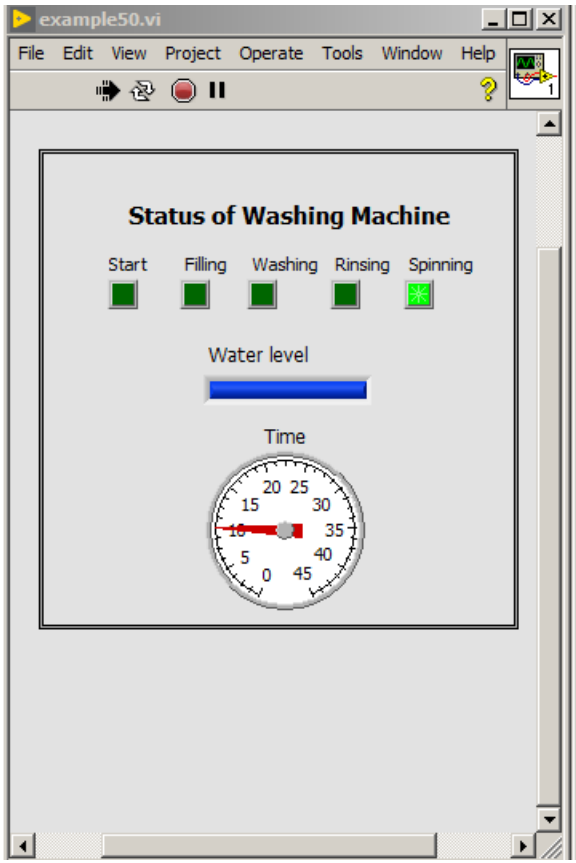
Przykład wykorzystania maszyny stanów - symulator pralki

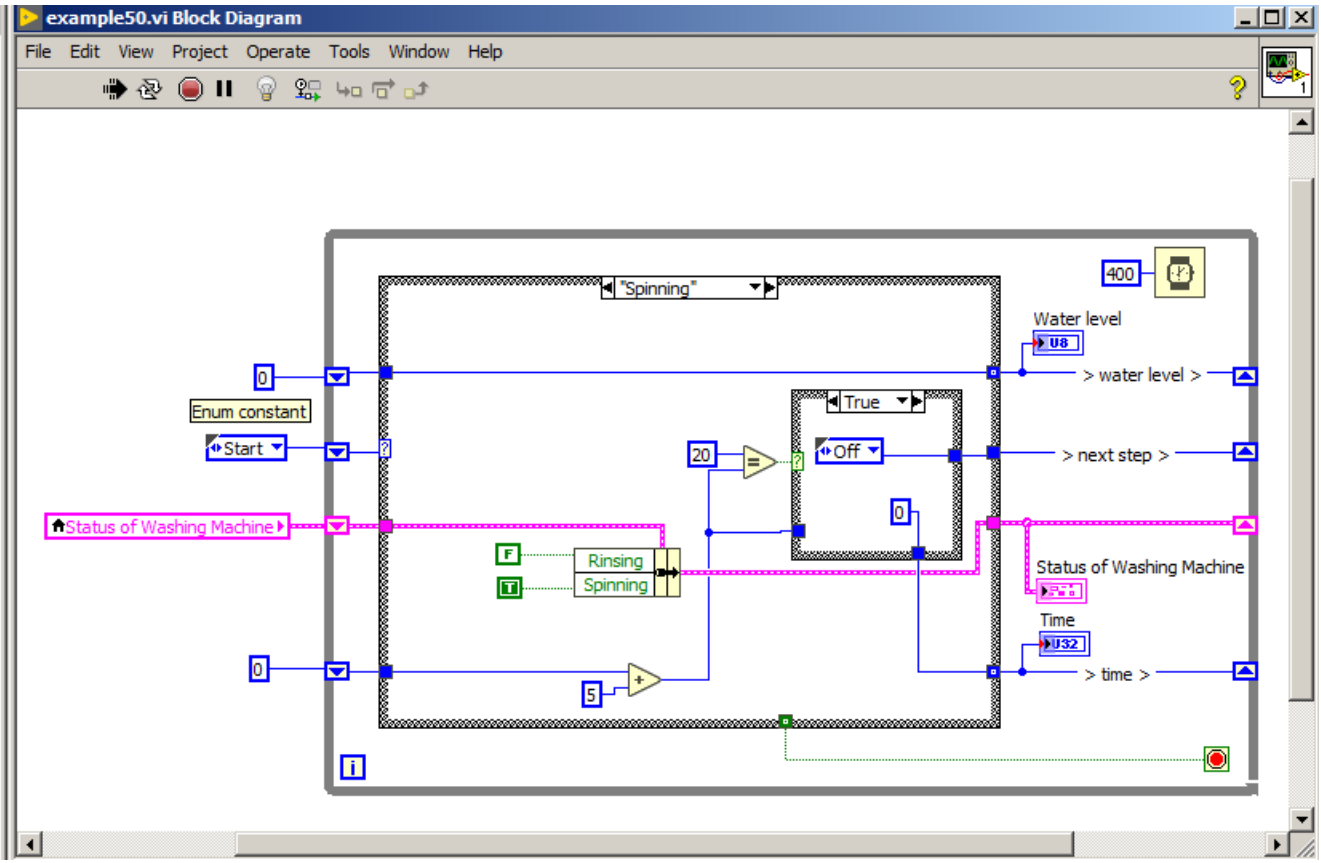


Przykład wykorzystania maszyny stanów - symulator pralki

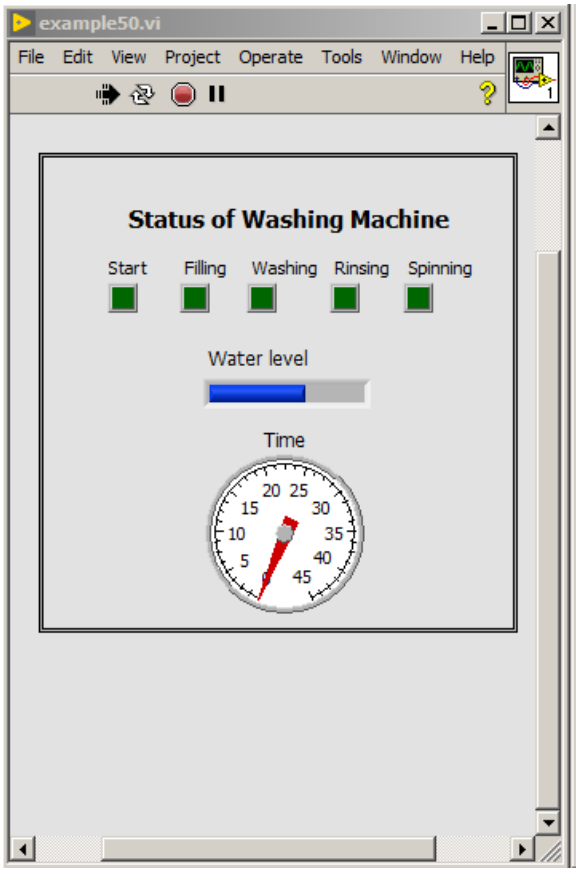


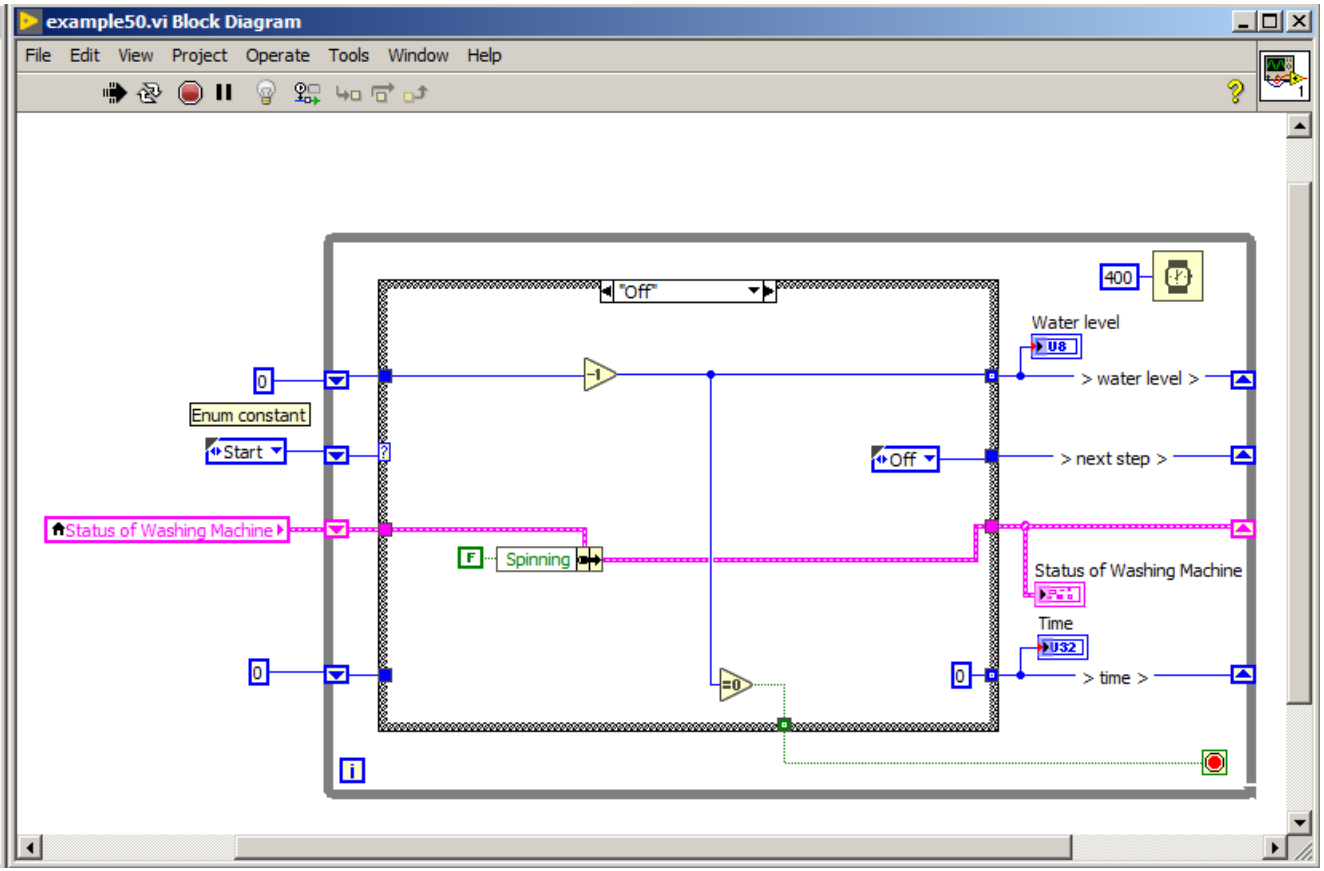
Przykład wykorzystania maszyny stanów - symulator pralki





Przykład wykorzystania maszyny stanów - symulator pralki





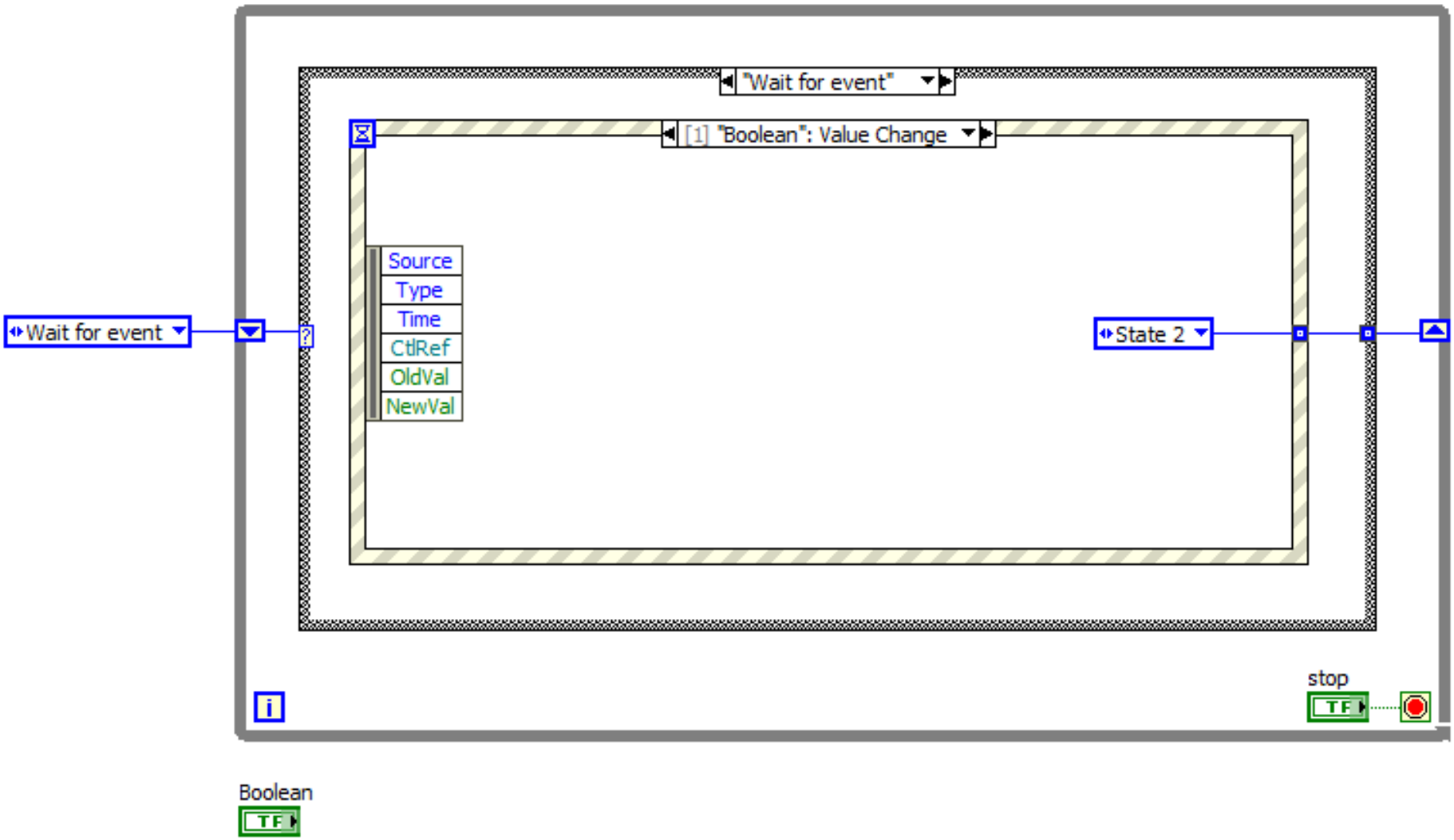
Maszyna stanów - zalety

- Pozwala łatwo realizować programy sekwencyjne o skomplikowanej strukturze.
- Łatwa w rozbudowie.
- Czytelna.

Maszyna stanów - wady

- Przejścia pomiędzy stanami muszą być znane i nie zmieniać się w trakcie wykonywania programu.
- Można zdefiniować wykonywanie się co najwyżej jednego stanu w przód.
- Utrudniona asynchroniczna obsługa interfejsu użytkownika.

Event-Based State Machine



Dziękuję za uwagę!

Wsparcie Komisji Europejskiej dla produkcji tej publikacji nie stanowi poparcia dla treści, które odzwierciedlają jedynie poglądy autorów, a Komisja nie może zostać pociągnięta do odpowiedzialności za jakiekolwiek wykorzystanie informacji w niej zawartych.

Wykład został opracowany w oparciu o materiały: "LabVIEW Core 1 Course Manual", "LabVIEW Core 2 Course Manual", pierwotną wersje wykładu: mgr. inż. Marcina Biedy oraz przykładowe egzaminy CLAD opublikowane na stronie www.ni.com .