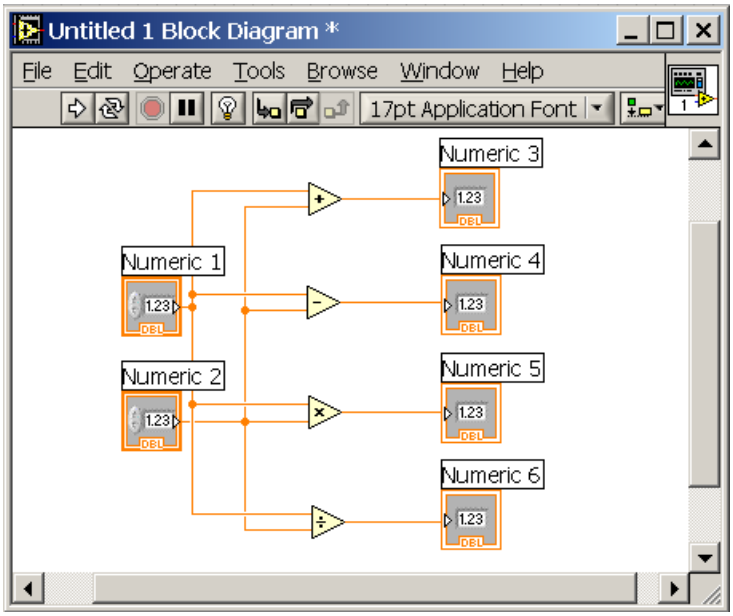
Wstęp do programowania w LabView

Poniżej opisano szereg programów jakie osoba ucząca się LabView może utworzyć chcąc poznać to środowisko programistyczne. W przypadku początkowych kilku najprostszych zadań przedstawiono właściwy „kod" programów tj. obrazy okien diagramów LabView. W przypadku dalszych trudniejszych programów przedstawiono wyłącznie obrazy interfejsu programów, tak aby zachęcić osoby uczące się do samodzielnego utworzenia programów.

LabView jest środowiskiem programistycznym („programem") pozwalającym na tworzenie złożonych aplikacji („innych programów"). Jest to narzędzie typu RAD (*Rapid Application Development*) tzn. środowisko szybkiego tworzenia aplikacji. Oznacza to, że tak jak w przypadku wielu innych środowisk tego typu (MS Visual Basic, Visual C#, Embercadero C++ Builder czy Delphi) tworzenie aplikacji opiera się na wizualnym tworzeniu interfejsu użytkownika oraz szybkim tworzeniu kodu programu dzięki zastosowaniu gotowych komponentów i funkcji. LabView wyróżnia jednak to, że kod programu podobnie jak interfejs jest tworzony w sposób wizualny. Każdemu elementowi języka, komponentowi, funkcji, strukturze przypisane są odmienne elementy graficzne. Obiekty te mają intuicyjną szatę graficzną dzięki czemu programowanie w LabView jest łatwe, a jego nauka przebiega szybko. Z drugiej strony LabView zapewnia również możliwość korzystania z kodu tworzonego w innych środowiskach programistycznych, włącznie z niskopoziomowym asemblerem. Wadą LabView jest duży rozmiar tworzonych aplikacji i często nie najwyższa wydajność (szybkość) tworzonych aplikacji. W przypadku fizyków, szczególnie eksperymentatorów, wady te nie mają znaczenia wobec szybkości tworzenia programów wspomagających eksperyment. W przypadku informatyków LabView stanowi ciekawy przykład graficznego środowiska programistycznego, niezwykle elastycznego, wykorzystywanego w przemyśle np. w procesie sterowania linią montażową.

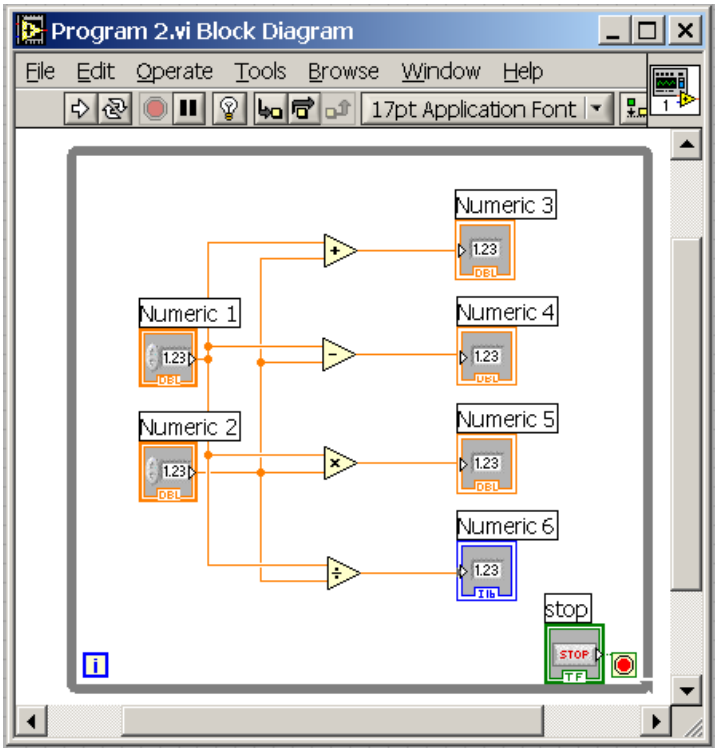
Na podstawie doświadczeń osób prowadzących zajęcia z LabView na Wydziale Fizyki, zarówno tych, które wcześniej nie miały do czynienia z innymi środowiskami programistycznymi, jak i takich, które mają wieloletnie doświadczenie w programowaniu np. w C++ czy w Delphi, można stwierdzić, że LabView stanowi idealne narzędzie dla osób chcących tworzyć własne, kompletne aplikacje, umożliwiające komunikację z różnymi urządzeniami pomiarowo-kontrolnymi, a jednocześnie pozwalające na bardzo zaawansowaną analizę danych zbieranych z tych urządzeń.

1. Zapoznaj się przy użyciu poniższego programu ze sposobem wykonywania podstawowych operacji arytmetycznych w LabView.



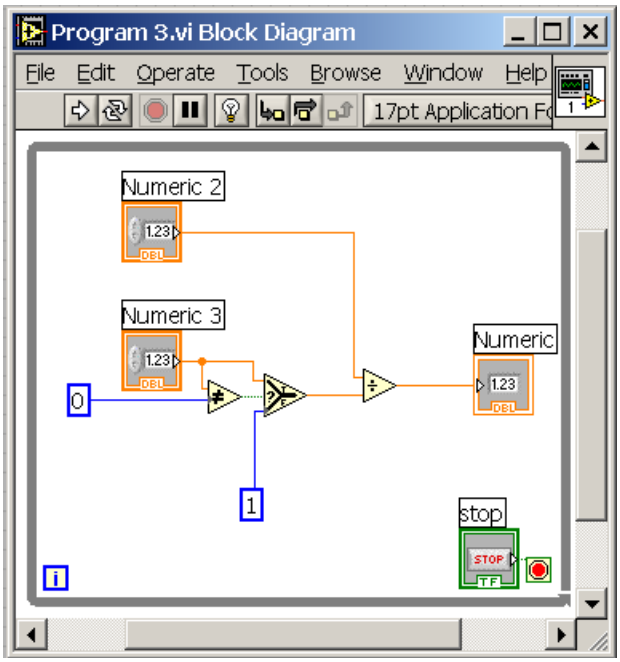
Przestaw sposób zmiany typu reprezentowanej liczby np. przez wskaźnik ilorazu (w *Representation* w menu kontekstowym ikony *Numeric 6* przestawić na *I32*). Zwróć uwagę na rolę koloru, grubości i wzoru linii łączących ikony w reprezentacji typu danych jakie przez te linie są przekazywane między obiektami.

2. Uruchamianie ciągłe programu (np. programu 1). Użycie przycisku  powoduje ciągłe inicjowanie wszystkich zmiennych, pamięci itd., a następnie ich zwalnianie. Postępowanie takie jest niewłaściwe. Wprowadź pętlę *While loop* jako narzędzie zapewniające uruchamianie aplikacji w trybie ciągłym, umożliwiające jednak jednokrotne inicjowanie pamięci. Zmodyfikuj program 1 tak by powstał program 2.

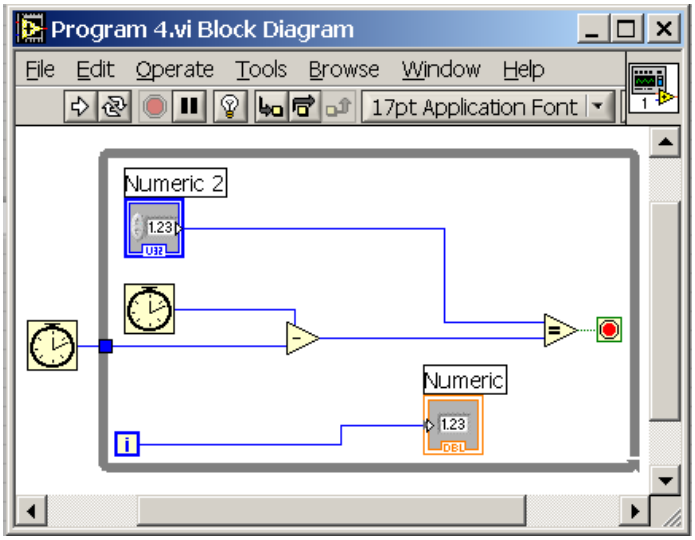


Jakie znaczenie mają pola i ? Zapamiętaj, że nie ma związku między ułożeniem obiektów na diagramie, a kolejnością wykonywania w programie (np. coś położone po prawej stronie pętli *While loop* nie musi zostać wykonane po jej zakończeniu, ale może być przed jej rozpoczęciem).

3. Utwórz program 3 prezentujący połączenie operacji logicznej i numerycznej na przykładzie dzielenia tak, aby uniknąć dzielenia przez 0.



4. Utwórz program 4 umożliwiający wyznaczenie czasu potrzebnego na wykonanie zadanej liczby iteracji pętli *While loop* lub liczby iteracji tej pętli wykonanej w zadanym czasie. W pierwszej wersji programu pętlę *While loop* możesz zastąpić pętlą *For loop*. Użyj funkcji *Tick Count*.



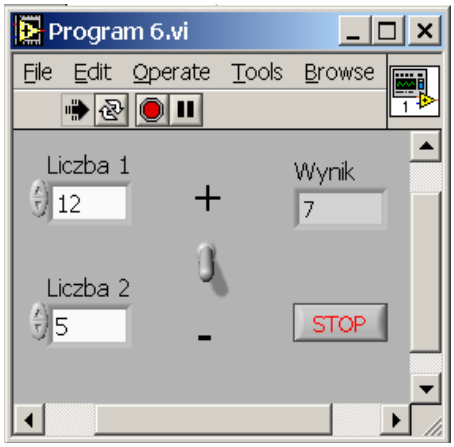
5. Utwórz program 5, który będzie losował trzy liczby z zakresu 0 – 9. Działanie programu zostanie przerwane w chwili gdy wszystkie losowane liczby osiągną wybraną stałą wartość np. 7. Losowane cyfry mogą być wyświetlane na trzech obok siebie leżących wskaźnikach. W chwili przerwania działania programu na odpowiednim wskaźniku powinna zostać wyświetlona liczba iteracji (losowań).

W programie należy wykorzystać funkcję losującą liczby *Random Number (0-1)*, wstrzymującą działanie programu *Wait*, funkcję rzutowania typu (do konwersji typu zmienno-przecinkowego na całkowity) *To Unsigned Byte Integer*, funkcję *Compound Arithmetic* wykonującą wybraną operację na więcej niż dwóch zmiennych (w tym

przypadku operację iloczynu logicznego *AND*) oraz funkcję wykonującą operację sumy logicznej *OR*.

Zauważ, losowość kodu sprawia, że liczba iteracji będąca wynikiem jego działania może być bardzo różna. We współpracy z prowadzącym spróbuj utworzyć program, który wykreśli histogram liczby iteracji potrzebnych do wylosowania trzech takich samych cyfr w kolejnych np. 10000 próbach.

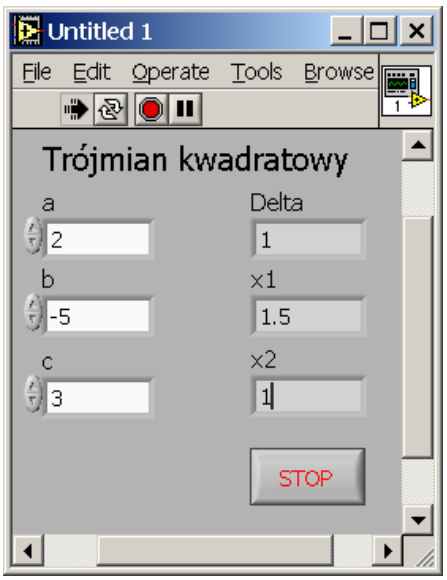
6. Utwórz program 6 wyznaczający sumę lub różnicę dwóch liczb. Rodzaj operacji musi jednak być określany na podstawie przełącznika logicznego.



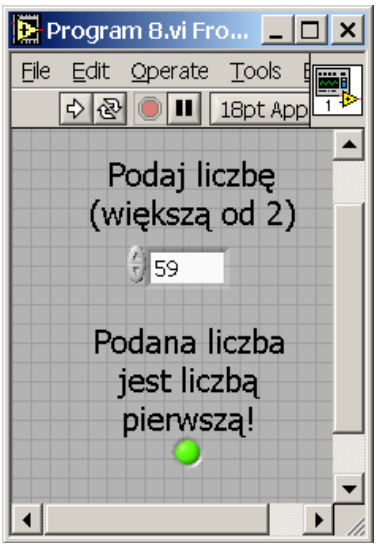
Należy użyć struktury wyboru *Case Structure*. Wykorzystaj ją najpierw w wersji programu z dwoma działaniami arytmetycznymi, a następnie użyj jej w zmodyfikowanej wersji programu uwzględniającej również operacje mnożenia i dzielenia (wybór operacji może być dokonywany np. za pomocą suwaka *Horizontal Pointer Slide*).

7. Utwórz program wyznaczający pierwiastki trójmianu kwadratowego. Na początku zaniedbaj problem osobnego traktowania przypadków ujemnej wartości wyróżnika trójmianu („delta”). Zwróć uwagę na to, że program sam radzi sobie z wyrażeniami typu ∞ (dla a = 0), czy $\sqrt{-1}$, nie powodując zatrzymania pracy.

   W dalszej kolejności możesz wprowadzić warunkową reakcję na niewłaściwe parametry równania (korzystając ze struktury *Case Structure*). W końcowym kroku spróbuj użyć struktury *Formula Node*, szczególnie jeśli jesteś zaznajomiony z programowaniem w C.



8. Utwórz program 8 sprawdzający czy podana liczba jest liczbą pierwszą.



Niech analizowana liczba *P* będzie liczbą naturalną, większą od 2. Sprawdzenie tego warunku odbywa się przy pomocy struktury *Case Structure*. Zgodnie z definicją liczby pierwszej, powinna dzielić się ona przez 1 i samą siebie. A więc wykazanie, że dzieli się ona „bez reszty" również przez którąkolwiek z liczb z zakresu od 2 do *P*/2, będzie wskazywać, że badana liczba nie jest liczbą pierwszą. Jeżeli w wyniku prowadzonego dzielenia „reszta" będzie zawsze różna od 0, to oznaczać będzie, że wprowadzona liczba jest pierwszą. Algorytm: Wewnątrz pętli *While loop* sprawdzamy, czy reszta (ang. remainder) z dzielenia liczby *P* przez liczby całkowite z zakresu od 2 do *P*/2 jest równa 0. Wykorzystujemy tu funkcję numeryczną *Quotient & Remainder*. Jeżeli reszta będzie równa 0, to pętla zostaje zatrzymana, a wskaźnikowi logicznemu przypisany zostaje fałsz (lampka gaśnie). W przypadku, gdy reszta pozostanie różna od zera, pętla zostaje zatrzymana w momencie, w

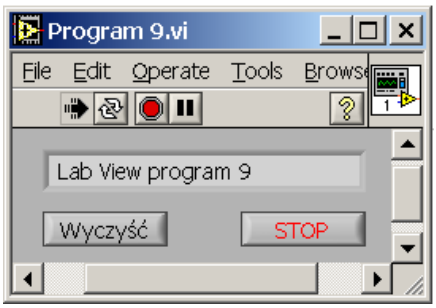
którym indeks pętli powiększony o 2 osiągnie wartość *P*/2. Wówczas zapalamy lampkę sygnalizująca znalezienie liczby pierwszej.

Alternatywnie, we współpracy z prowadzącym możesz użyć np. algorytmu Sita Eratostenesa (patrz Wikipedia), którego kod w j. C++ może mieć postać

```
void SitaEratostenesa()
{
        for (int i = 2; i <= sqrt(N); ++i) {
                if (prime[i]) {
                        for (int j = i * i; j <= N; j += i) {
                                prime[j] = false;
                        }
                }
        }
}
```

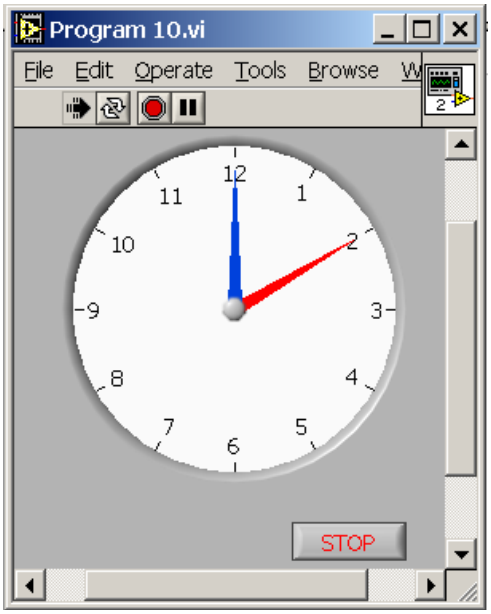
gdzie prime to tablica wartości logicznych. Test odbywa się w zakresie od 2 do N i te elementy tablicy, które mają na koniec wartości prawdy (true) odpowiadają liczbom pierwszym.

9. Utwórz program 9 wykorzystujący obsługę zdarzeń w LabView. Program posiada wskaźnik łańcuchowy, w którym pojawiać się mają litery odpowiadające naciśniętym klawiszom klawiatury w czasie działania programu. Przyciśnięcie przycisku *Wyczyść* powinno czyścić zawartość wskaźnika łańcuchowego.

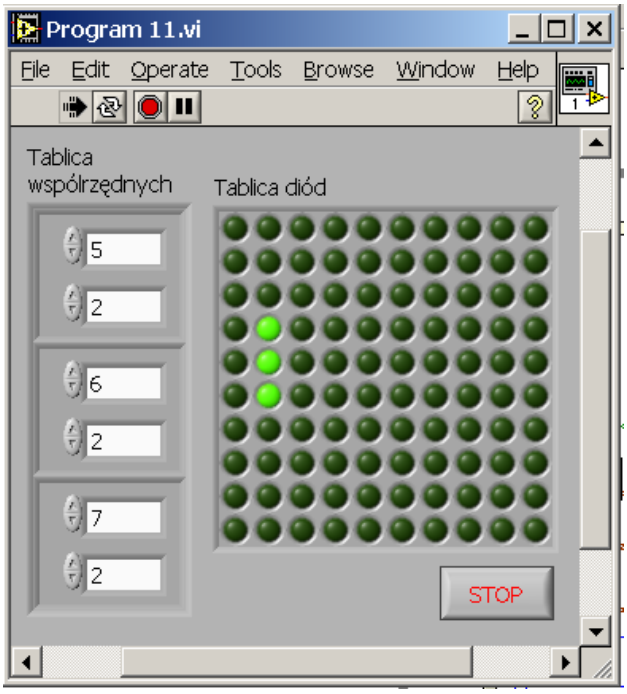


W programie należy użyć funkcji rzutującej *Type Cast* (zamieniającej) znak zapisany w kodzie ascii (*Char*) na typ łańcuchowy. Ponieważ rzutowanie wstawia spację przed znak naciśniętego klawisza, należy za pomocą funkcji *String Subset* pobrać drugi znak. Funkcja *Concatenate Strings* służy do połączenia dwóch łańcuchów w jeden, a zmienne lokalne (*Local Variable*) dostępne z menu kontekstowego nad ikoną obiektu, którego zmienną chcemy utworzyć, należy użyć aby mieć dostęp do wskaźnika łańcuchowego w różnych miejscach kodu.

10. Utwórz program 10 odmierzający czas od chwili jego uruchomienia i wyświetlający ten czas na tarczy zegara dwu-wskazówkowego. Jedna wskazówka powinna wskazywać sekundy, a druga minuty. Należy użyć funkcji zwracającej czas *Tick Count* oraz funkcji dzielenia modulo *Quotient & Remainder*.



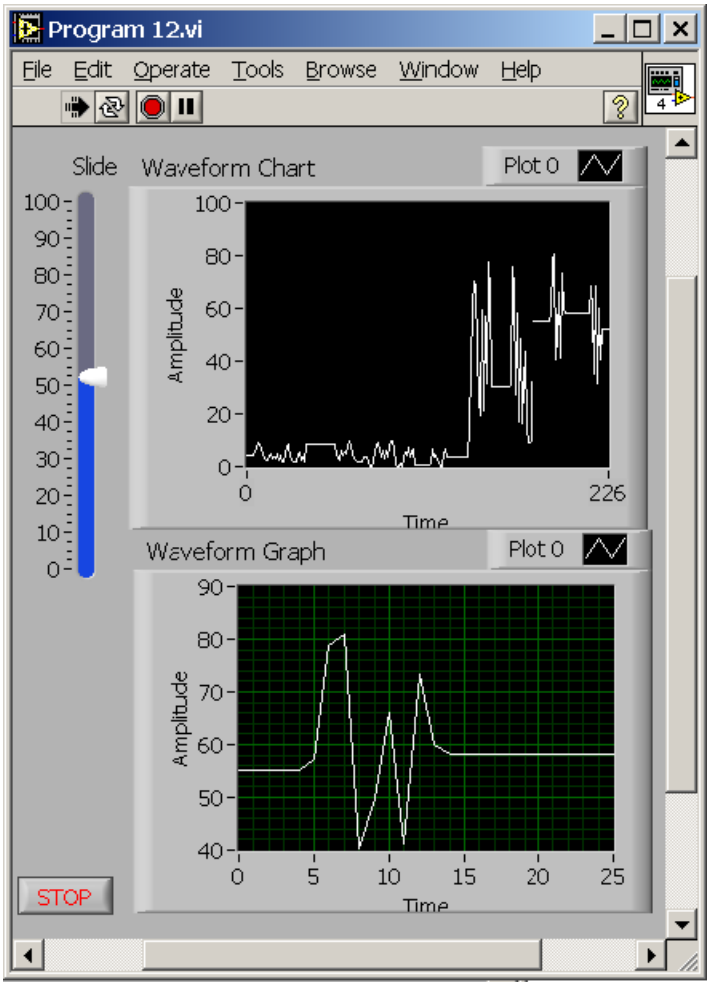
11. **Zadanie na 2(3) ćwiczenia**. Utwórz program 11 będący prostą wersją gry „Snake" w wersji podstawowej tj. tylko ruch węża, bez wydłużania się. Grę zrealizuj na bazie wskaźnika tablicy sygnalizatorów logicznych (pole) oraz tablicy par współrzędnych (wąż). Kierowanie wężem powinno się odbywać poprzez obsługę zdarzeń związanych z klawiaturą. Obsłuż tylko ruch lewo/prawo (względem kierunku poruszania się węża) lub w czterech kierunkach (względem użytkownika programu).



W pierwszym kroku należy zainicjować wartościami początkowymi tablicę diod (zapalić odpowiednie diody określone tablicą par współrzędnych). Należy zaznajomić się z funkcją inicjującą tablicę, wstawiającą do niej nowe wartości, z operacją przenoszenia do następnego kroku pętli wartości zmiennej z poprzedniego kroku (*Shift Register*) i z automatycznym pobieraniem elementów wybranej tablicy za pomocą pętli *For loop*. Należy

zwrócić uwagę na to, że inicjowanie tablic musi nastąpić w pierwszej kolejności. Chcąc to osiągnąć można zastosować struktury sekwencyjne *(Stacked Sequence Structure* i *Flat Sequence Structure*) albo oprzeć się na kolejności przesyłania wartości zmiennych pomiędzy kolejnymi fragmentami programu (jeśli pętla *While loop* na swoim wejściu przyjmuje wartość wyznaczaną wcześniej w pętli *For loop* to zawsze ta druga będzie wykonywana w pierwszej kolejności).
W następnym kroku należy utworzyć właściwą część programu. Ponieważ sterowanie „wężem" ma odbywać sie poprzez naciskanie klawiszy klawiatury, konieczne jest użycie *Event Structure*. Program w każdym kroku pętli aktualizuje tablicę diod. W dwóch zmiennych całkowitych należy zachowywać obecny kierunek poruszania się „węża". W kierunku pionowym, -1 – góra, 0 – brak ruchu, +1 – dół, a w kierunku poziomym, -1 – lewo, 0 – brak ruchu, a +1 – prawo. Zmienną tę należy w każdym kroku dodawać do odpowiednich współrzędnych „głowy węża" tj. Ostatniego elementu tablicy par współrzędnych. Tak uzyskane nowe współrzędne należy zapisać w tablicy par współrzędnych. Wcześniej należy jednak zgasić diodę odpowiadającą „ogonowi węża" tj. pierwszemu elementowi tablicy współrzędnych, podmieniając wartość w tablicy diod na *false*. Współrzędne „ogona" są już niepotrzebne zatem na ich miejscu możemy zapisać współrzędne „nowej głowy". Współrzędne te znajdą się jednak w tablicy współrzędnych na niewłaściwym miejscu bo na jej początku. Należy ją zrotować, tak aby „nowa głowa" znalazła się na ostatniej pozycji tablicy. Wreszcie współrzędne te należy wykorzystać, aby zapalić odpowiednią diodę w tablicy diod. Należy pamiętać, że ruch w prawo uniemożliwia skręcenia w lewo, podobnie w drugą stronę jak i w kierunku pionowym.
Program można rozwinąć w wersję sieciową dla dwóch graczy korzystając z funkcji obsługi sieci TCPIP dostępnych w LabView i przesyłając drugiemu graczowi współrzędne początkowe i aktualizowany kierunek ruchu.

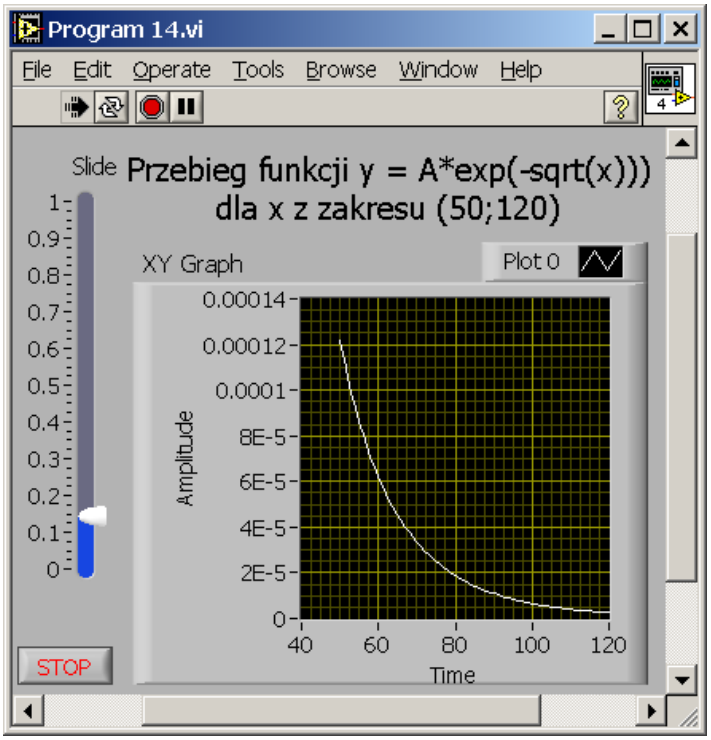
12. Utwórz program 12 wykreślający wartość przekazywaną przez kontrolkę numeryczną na *Waveform Chart* i *Waveform Graph*. Zwróć uwagę na polecenie *Enable Indexing* w pętli *While loop*. Prezentowana wersja programu nie czyści wykresów w chwili uruchomienia programu. Opcję taką można dodać do programu korzystając ze zmiennych lokalnych.



13. Często zachodzi potrzeba umieszczenia na jednym wskaźniku graficznym dwóch lub więcej wykresów. Umieszczanie dodatkowych przebiegów na wskaźnikach oraz *Waveform Graph* różni się trochę ze względu na różne typy danych przyjmowane przez te dwa wskaźniki. Aby wykorzystać *Waveform Chart* do prezentacji większej ilości wykresów, pojedyncze liczby tworzące przebiegi należy połączyć w tzw. klaster (*Cluster*). W przypadku *Waveform Graph* należy stworzyć tablicę (*Array*) dwuwymiarową, w której każda kolumna odpowiadać będzie kolejnemu wykreślanemu wektorowi. Użyj poprzedniego przykładu. Źródłem danych wykreślanych na pierwszym przebiegu niech pozostanie kontrolka numeryczna, natomiast drugi przebieg niech przedstawia np. wartości losowe. Połącz obie wartości w klaster wykorzystując funkcję *Bundle* i prześlij do wskaźnika *Waveform Chart*. Aby umieścić drugi przebieg na wskaźniku *Waveform Graph* należy zbudować dodatkową tablicę wygenerowanych liczb losowych w taki sam sposób jak w poprzednim przykładzie tj. odkładając wartości na krawędzi pętli. Po zakończeniu działania pętli, wygenerowane wektory należy połączyć w tablicę dwukolumnową (*Build Array*) i przesłać na wskaźnik graficzny. Interfejs zmodyfikowanego programu 12 nie ulega zmianie, natomiast kod tak.

14. Utwórz program 14 wykreślający przebieg dowolnej funkcji y(x) dla x należącego do

określonego przedziału wartości i zmieniającego się z zadanym krokiem. Wyznaczanie wartości funkcji można zrealizować za pomocą *Expression Node* w sytuacji gdy x jest jedyną zmienną y, a za pomocą *Formula Node* gdy funkcja y ma inne parametry (o wartościach podanych np. poprzez kontrolki).



W programie należy wykreślić zależność y(x). Dlatego należy użyć *XY Graph*.

15. **Zadanie na 2(3) ćwiczenia. Obsługa karty dźwiękowej w LabView**.
Karta dźwiękowa obsługiwana jest w LabView poprzez standardową bibliotekę API (Windows). Traktowana jest jak większość urządzeń pomiarowych: wymaga zainicjowania zadania (*task*), ustawienia trybu pracy, wartości parametrów, obsługi danych wchodzących lub wychodzących oraz zamknięcia na koniec pracy. Karta dźwiękowa może służyć zarówno do celów „akustycznych", związanych z generowaniem lub detekcją dźwięku jak i do celów czysto elektronicznych, pozostających w sferze zmiennych sygnałów elektrycznych (przetwornik A/C i C/A). Ze względu na powszechność występowania plików dźwiękowych LabView dostarcza narzędzi do czytania i tworzenia plików typu WAV. Przeprowadzenie ćwiczenia wymaga znajomości pojęć częstotliwości próbkowania, rozdzielczości 8 i 16 bitowej danych, wielkości buforu karty, analizy Fourierowskiej, pojęcia sygnału w LabView.

W pierwszej kolejności należy przeanalizować przykładowy program dostarczany razem z

LabView, *Continuous Sound Output*, przeznaczony do emisji dźwięku o zadanej częstotliwości poprzez kartę dźwiękową. W następnym kroku należy przeanalizować przykładowy program dostarczany razem z LabView, *Revord Wave File*, służący do rejestracji dźwięku poprzez kartę. Oba te programy stanowią punkt wyjścia do aplikacji tworzonej w ramach ćwiczeń.

Zadaniem tej aplikacji (Program 15 – autor mgr M. Pochylski) jest przesyłanie tekstu pomiędzy dwoma komputerami. Tekst ten ma być przenoszony między nimi poprzez zwykły przewód łączący wyjście karty dźwiękowej jednego z komputerów z wejściem karty drugiego z nich. Kodowanie tekstu (a ściślej kilku wyrazów) ma odbywać się zgodnie z następującym schematem. Każdej kolejnej literze tekstu przypisywana jest kolejna, coraz wyższa częstotliwość dźwięku (sygnału sinusoidalnego). Sygnał przekazywany z jednostki przekazującej tekst ma być sygnałem będącym superpozycją (sumą) poszczególnych dźwięków (sinusoid) przypisanych kolejnym literom. Chcąc zapisać w wyjściowym sygnale informację o kolejności liter w tekście nie można danej literze alfabetu przypisać określonej częstotliwości. Częstotliwość przypisana danej literze wyrazu musi zawierać informację o tym jaka to litera i o jej pozycji w tekście. W tym celu obieramy pewną częstotliwość bazową np. 50 Hz. Kolejnym literom alfabetu przypisujemy kolejne liczby tak, że a,b,c,d,e,... odpowiada 1,2,3,4,5,... . Częstotliwość pierwszej litery tekstu wyznaczamy dodając do częstotliwości bazowej iloczyn jej pozycji w alfabecie i dowolnej stałej wartości przedziału np. 10 Hz. Częstotliwość drugiej litery tekstu wyznaczamy dodając do częstotliwości przypisanej pierwszej literze tekstu iloczyn pozycji drugiej litery w alfabecie i stałej wartości przedziału (10 Hz). Kolejne częstotliwości ustalamy według tego samego schematu jaki zastosowaliśmy dla drugiej litery tekstu. W ten sposób np. wyraz „obsada” zostanie zapisany sygnałem składającym się z sinusoid o następujących częstotliwościach:

a-1,b-2,c-3,d-4,e-5,f-6,g-7,h-8,i-9,j-10,k-11,l-12,m-13,n-14,o-15,p-16,r-17,s-18,t-19,u-20,v-21,w-22,x-23,y-24,z-25

„o” - 50 Hz + 15*10 Hz = 200 Hz,

„b” - 200 Hz + 2*10 Hz = 220 Hz,

„s” - 220 Hz + 18*10 Hz = 400 Hz,

„a” - 400 Hz + 1*10 Hz = 410 Hz,

„d” - 410 Hz + 4*10 Hz = 450 Hz,

„a” - 450 Hz + 1*10 Hz = 460 Hz

co symbolicznie można zapisać

$$\nu_1 = 50Hz + j_0 \cdot 10Hz$$
$$\nu_i = \nu_{i-1} + j_i \cdot 10Hz$$

gdzie i to pozycja danej litery w tekście, j_i to pozycja danej litery w alfabecie, a ν_i to częstotliwość przypisana tej literze.

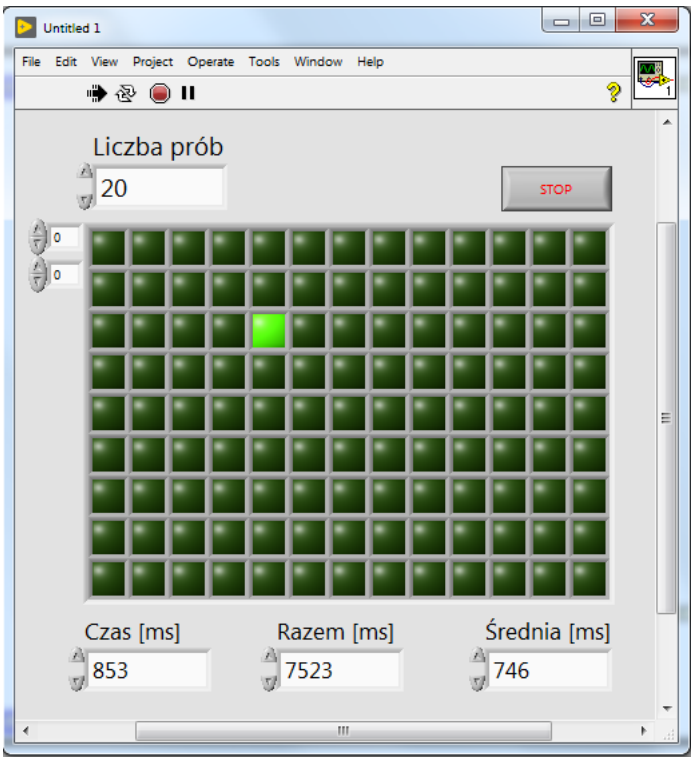
Po wyznaczeniu powyższych częstotliwości należy wygenerować sygnał będący sumą odpowiadających im sinusoid i przesłać go na wyjście karty dźwiękowej.

Bardziej złożona jest część odpowiedzialna za analizę (dekodowanie) sygnału i prezentację przesłanego tekstu. Po zebraniu sygnału z karty dźwiękowej należy poddać go analizie Fourierowskiej za pomocą np. funkcji *Auto Power Spectrum*. Następnie w zwracanym przez tę funkcję *spektrum mocy* należy odnaleźć (ustalając jakiś próg) piki odpowiadające częstotliwościom składowym zebranego sygnału za pomocą funkcji *Peak Detector*. Położenia tych pików (*Locations*) należy przeskalować do właściwych wartości korzystając z wartości *df* funkcji *Auto Power Spectrum*. Na koniec wyznaczając różnicę pomiędzy kolejnymi składowymi częstotliwościami i dzieląc ją przez stałą wartość przedziału (10 Hz) można wyznaczyć kolejne litery przesłanego tekstu.

16. Wśród przykładowych programów można również znaleźć dwa programy obrazujące użycie *Event Structure* i możliwości modyfikacji interfejsu w LabView. Pierwszy program to odpowiednik systemowego kalkulatora (wersji standardowej, obsługiwany wyłącznie przez mysz – spróbuj dodać do niego obsługę klawiatury), a drugi to program imitujący jedno-oktawowe piano, wykorzystujący dźwięki emitowane poprzez głośniczek systemowy w obudowie komputera. Program ten prezentuje sposób bezpośredniego dostępu do portów komputera z poziomu LabView.

Inne potencjalne programy do stworzenia:

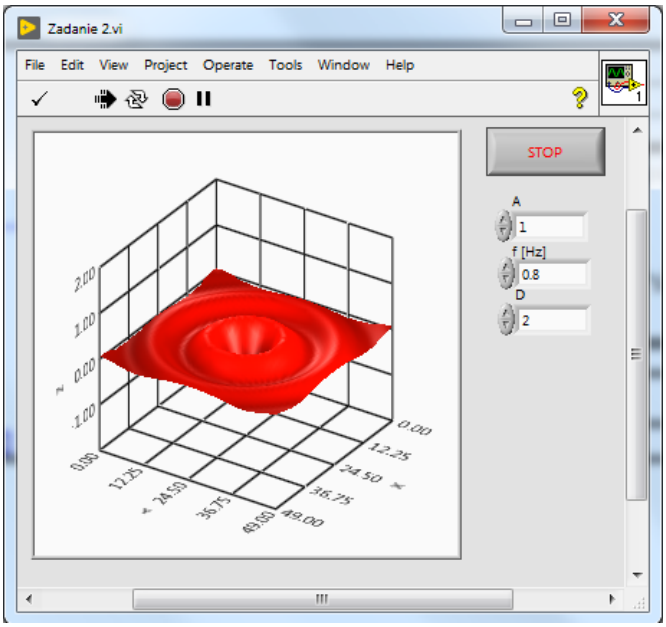
17. Utwórz program, który będzie zliczał czas potrzebny badanemu na kliknięcie na diodę matrycy diod. Przy czym dioda ma być zapalana w chwili losowej z zakresu czasu 1.3s – 3s od poprzedniego zgaszenia diody w losowej pozycji na matrycy. Program ma zliczać średni czas potrzebny użytkownikowi na zgaszenie diod w kolejnych próbach, których liczba ma również być ustawiana przez użytkownika. Ponadto ma być zliczany łączny czas potrzebny badanemu na zgaszenie diod. Przy czym chodzi o czas między zapaleniem, a zgaszeniem diody – nie o całkowity czas uruchomienia programu.



18. Utwórz program, który na wykresie 3D wyświetli zmieniającą się w czasie powierzchnię funkcji

$$z(f, D, r, t) = A \cdot \sin(2\pi \cdot f \cdot t) \cdot \frac{\sin(r / D)}{r / D}$$

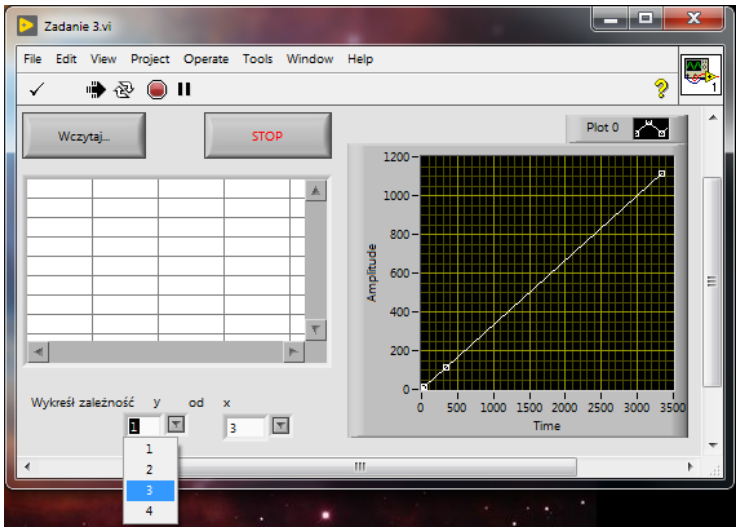
Przy czym dla r = 0 funkcja przyjmie wartość $A \cdot \sin(2\pi \cdot f \cdot t)$. Wielkości A, f i D mają być zadawane z kontrolek, t to upływający czas.



19. Utwórz program, który po naciśnięciu przycisku Load… i wyborze pliku tekstowego przez Ciebie przygotowanego, wczyta zbiór liczb zapisanych w nim

xx yy yy yy yy
xx yy yy yy yy
xx yy yy yy yy
xx yy yy yy yy
xx yy yy yy yy

gdzie xx i yy to dowolne liczby przez ciebie wybrane. Następnie wyświetli zawartość pliku w postaci tablicy liczba na kontrolce typu *Table*. Interfejs powinien być zaopatrzony dodatkowe dwa pola i wyświetlacz wykresu. Pola te powinny być polami rozwijanymi zawierającymi numery kolumn tabeli, które posiadają jakieś dane. Po wczytaniu pliku pola te powinny być ustawione domyślnie na 1. Czyli wskazywać na pierwszą kolumnę (na potrzeby użytkownika programu iterujemy od 1). Automatycznie też na wykresie powinna zostać wyświetlona zależność y(x) (jak na załączonym niżej obrazku). Zmiana dowolnego z dwóch pól rozwijanych powinna skutkować wyświetleniem wybranej zależności y(x).

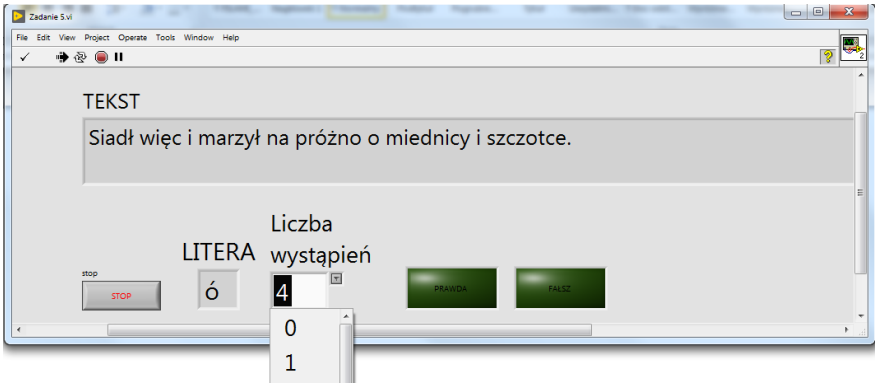


20. Utwórz program, który będzie służył określeniu stopnia symetrii twarzy danej osoby. W pierwszej kolejności ściągnij z internetu kilka zdjeć wykonanych na wprost (to ważne). Program przez ciebie wykonany powinien:
a) wczytać wybrane zdjęcie po naciśnięciu przycisku „Wczytaj…”
b) zakończyć działanie po naciśnięciu przycisku „Stop”,
c) wyświetlić wczytane zdjęcie na wskaźniku 2D Picture,
d) po naciśnięciu przycisku „Analizuj” program na drugim wskaźniku 2D Picture powinien wyświetlić zdjęcie będące wynikiem działania operacji A:

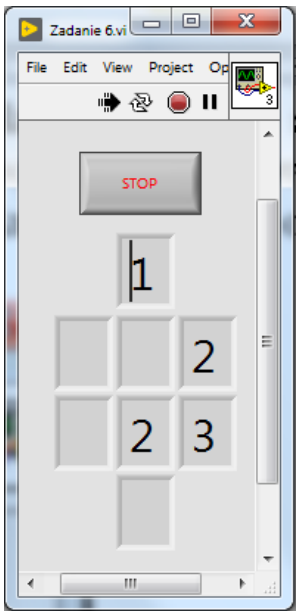
Operacja A. Po uruchomieniu analizy, wczytane zdjęcie powinno zostać zamienione na tablicę 2D. Tablica ta powinna zostać podzielona na dwie, lewą i prawą cześć zdjęcia. Linia przecięcia powinna być zadawana za pomocą np. suwaka numerycznego, tak by móc na żywo obserwować wpływ jego położenia na ostateczny wynik. Każda połowa powinna zostać powielona, ale odbita lustrzanie. Tak by po sklejeniu ze swoim pierwowzorem otrzymać idealnie symetryczną twarz. Odbicie symetryczne możesz wykonać wykonując sukcesywnie na kolejnych wierszach odwrócenie tablicy. Sklejenie dwóch tablic możesz wykonać sklejając ich wiersze. Następnie jedna wersja zdjęcia twarzy powinna zostać odjęta od drugiej wersji. Zwróć uwagę, że powstałe symetryczne tablice nie musza mieć takiej

samej liczby kolumn. Różnicę w ich liczbie można ustalić na podstawie wartości suwaka numerycznego określającego środek (pionowy) zdjęcia.

21. Napisz program, który będzie sprawdzał spostrzegawczość badanego. W okienku tekstowym ma na określony czas pojawić się tekst. Badany ma w określonej kontrolce podać ile razy w czytanym tekście wystąpiła dana litera, wskazana w innym wskaźniku. Program powinien z wcześniej przygotowanego tekstu wczytać zbiór tekstów do wyświetlenia (mogą być to kolejne wiersze). Następnie powinien wybrać losowo jeden z tekstów. Dalej, program powinien wylosować literę alfabetu. Następnie sprawdzić czy występuje ona w wylosowanym tekście. Jeśli tak to tekst zostaje wyświetlony. Jeśli nie to losowana jest nowa litera. Poprawnie wylosowana litera, występująca w tekście, zostaje wyświetlona na wskaźniku, jako ta której wystąpienia ma zliczyć badany. Następnie program czeka np. 3s na odpowiedź badanego i ukrywa tekst. Badany ma kolejne np. 3 s na wpisanie liczby wystąpień do kontrolki. Dwie diody LED (Dobrze! i Źle!) zapalają się w zależności od tego czy badany prawidłowo określił liczbę wystąpień dane litery. Ma to wyglądać tak:

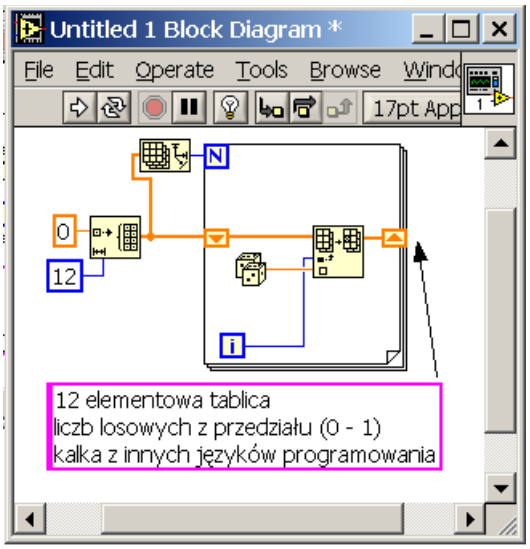


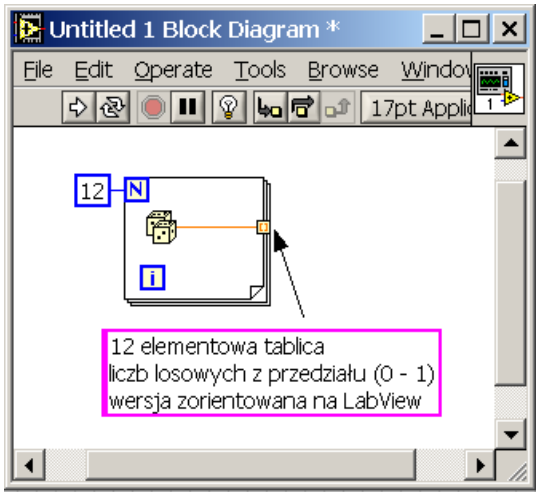
22. Stwórz program Memo. Ma to być odpowiednik znanej gry pamięciowej polegającej na odnajdywaniu par obrazków. Zamiast obrazków gracz będzie szukał pary tych samych cyfr. Cyfry mają należeć do przedziału od 0 do 3. Daje to 8 pól (2 x 4). Stwórz wyświetlacz z 8 polami łańcuchowymi. Zakrycie obrazków ma polegać na wyświetleniu pustych znaków. Użytkownik klika myszką na kolejne dwa pola. Oba kliknięcia prowadzą do odsłonięcia klikniętego pola. Jeśli są takie same, wtedy pola pozostają wypełnione odkrytymi cyframi. Jeśli są różne to po 2 s pola ponownie stają się puste. Program kończy działanie na skutek naciśnięcia przycisku Stop, lub po odkryciu wszystkich par.


Zadanie 6.vi
File
Edit
View
Project
STOP
1
2
2
3

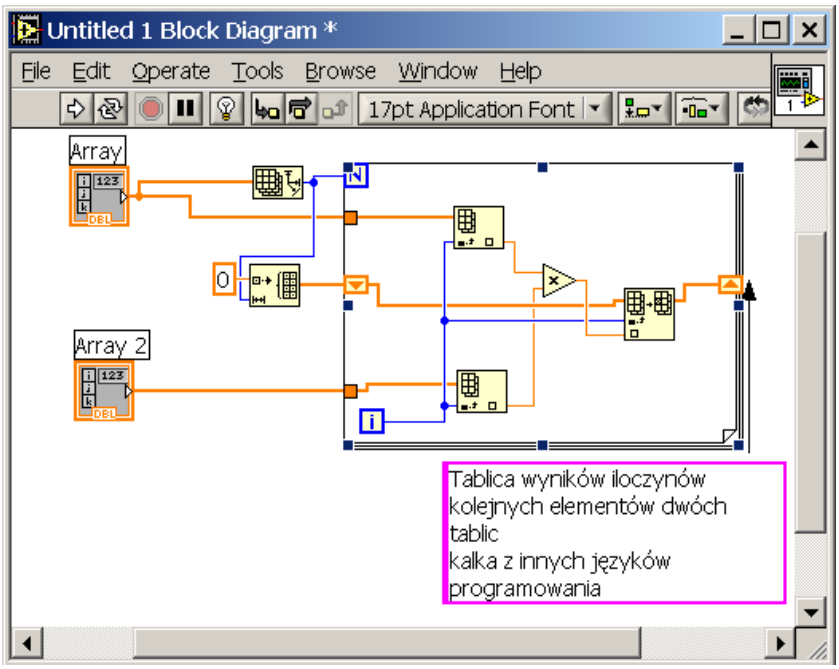
Przykłady błędnego i właściwego programowania w LabView.

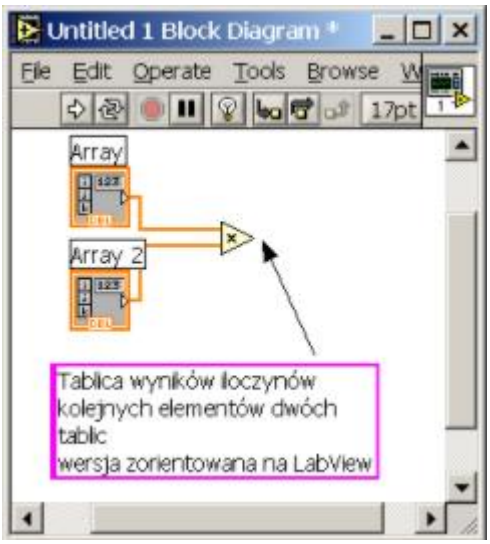
1. Operacje na tablicach: tworzenie tablic (wektorów, macierzy).





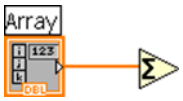
2. Proste operacje arytmetyczne i algebraiczne na tablicach (sumowanie, dodawanie stałej, mnożenie przez stałą, mnożenie skalarne wektorów).



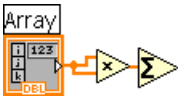


3. Poprawne sumowanie tablic.

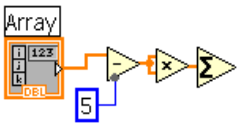
$\sum_i x_i$



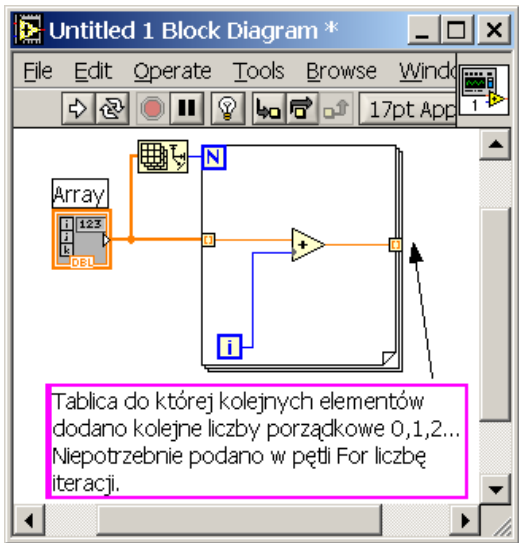
$\sum_i x_i^2$



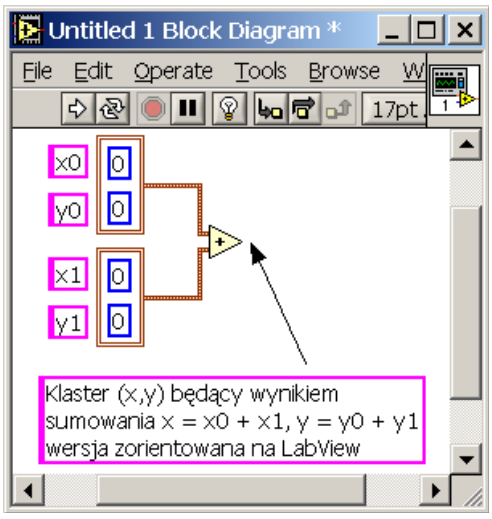
$\sum_i (x_i - 5)^2$

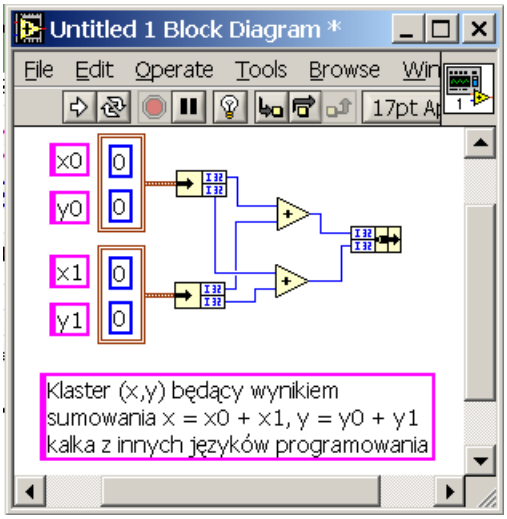


4. Niepotrzebne używanie rozmiarów tablicy w miejscach, gdzie nie jest to potrzebne (LabView na wejściu tablicy do pętli automatycznie określa rozmiar tablicy).



5. Operacje na klastrach: typowy błąd to rozbijanie klastra na elementy składowe przed wykonaniem identycznych operacji arytmetycznych na wszystkich elementach klastra, np. dodawanie klastrów, powiększanie elementów klastra o stałą lub mnożenie przez stałą, mnożenie klastrów.





Wszystkie programy przedstawione powyżej i dołączone zostały stworzone przez grupę prowadzącą zajęcia z LabView na Wydziale Fizyki UAM tj. dr J.Gapińskiego, dr G. Burdzińskiego, dr K. Dobka, mgr. M. Baranowskiego i mgr. M. Pochylskiego. Wszelkie pytania prosimy kierować do kogokolwiek z wyżej wymienionych.

Uruchomienie gotowych programów wymaga instalacji *LabVIEW Run-Time Engine*. Silnik ten jest instalowany razem ze środowiskiem LabView lub może zostać pobrany ze strony *www.ni.com*. Do przeglądnięcia kodu konieczne jest zainstalowanie LabView.