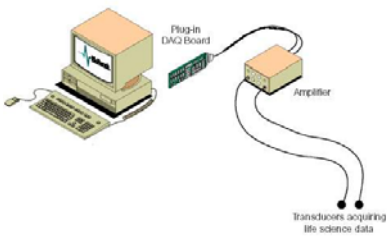
VIRTUELNA INSTRUMENTACIJA

- LabVIEW-

I deo

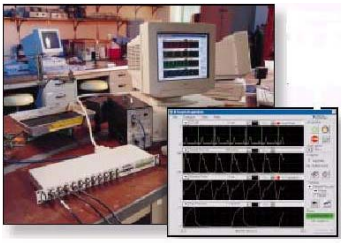
VIRTUELNA INSTRUMENTACIJA

- Virtuelna instrumentacija predstavlja metodologiju za projektovanje instrumenata, koja koristi standardni PC računar, specijalne hardverske komponente za akviziciju i digitalnu konverziju signala, i računarske programe koji omogućuju prikupljanje, obradu i prikaz signala na računaru
- Virtuelna instrumentacija omogućava objedinjavanje različitih tipova instrumenata u jedan insturment – PC računar
- Obezbeđuje lako programiranje instrumenata, reprogramiranje i nadogradnju potojećih instrumenata
- Omogućava iskorišćenje postojećih resursa PC računara: memorijski prostor, brza obrada velike količine podataka, baze podataka, Internet, e-mail, LAN...
- Olakšana je upotreba instrumenata jer su zasnovani na PC korisničkom interfejsu
- *Reusability*



VIRTUELNA INSTRUMENTACIJA

- Rodonačelnik virtuelne instrumentacije je firma *National Instruments* iz SAD-a
- *National Instruments* se bavi proizodnjom:
  - *Hardware* – DAQ kartice, uređaji za automatiku i merenje
  - *Software* – LabVIEW, aktuelna verzija v7.1
  - Measurement Studio (Visual Basic, VC++, LabWindows C)
- Primena drugih alata u virtuelnoj instrumentaciji: Matlab (Data Acquisition Toolbox)

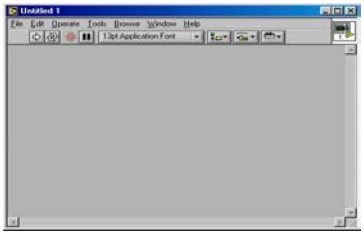


Laboratorija opremljena virtuelnim instrumentima

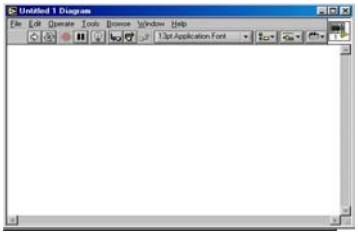
DAQ - kartica, *National Instruments*

LabVIEW

- LabVIEW je grafički orijentisan programski jezik koji koristi ikonice umesto teksa za kreiranje aplikacija
- LabVIEW - Laboratory Virtual Instrument Engineering Workbench.
- Umesto pisanih instrukcija, koristi se tok podataka (*data-flow*) dijagram za pisanje koda
- Sastoji se iz *front panela* u kome se projektuje korisnički interfejs i *block diagram-a* u kome se programira kod.



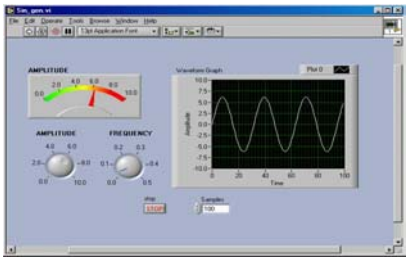
front panel



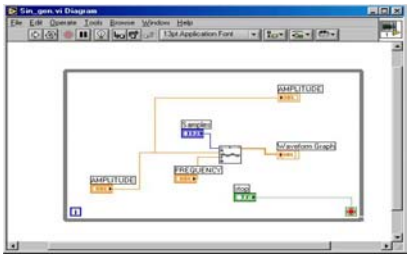
block diagram

LabVIEW

- U LabVIEW- u, korisnički interfejs se pravi koristeći već gotov set kontrola, indikatora i objekata (prekidači, grafici, digitalne kontrole)
- Kada se isprojekuje interfejs, tada se u *block diagram* prozoru 'piše kod'
- Blok dijagram predstavlja mesto gde se "piše kod" - logički se povezuju kontrole i uslovi izvršavanja programa
- U blok dijagramu se koristi tok podataka (*data flow*), a ne sekvencijalne instrukcije kao u tekstualnim programskim jezicima
- Upravo tok podatak određuje redosled izvršavanja delova programa



Sinus generator - *front panel*



Sinus generator – *block diagram*

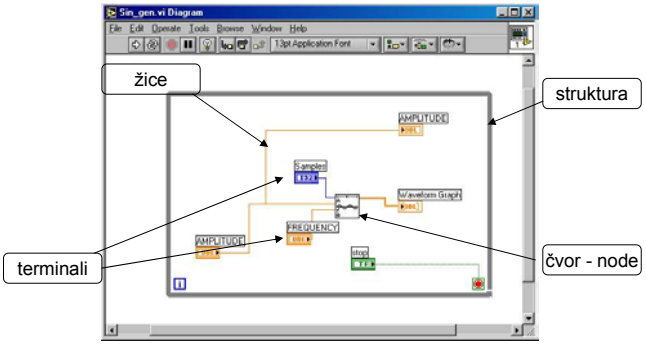
Programiranje u LabVIEW-u

- Programi u LabVIEW- u se nazivaju virtuelni instrumenti i obeležavaju sa VI
- Na front panelu se postavljaju kontrole: grafici, prekidači, indikatori i drugi elementi. U njemu se projektuje interfejs koji treba da podseća na konvencionalni fizički instrument (osciloskop, multimetar...)
- Interfejs treba da omogući lak i jednostavan način korišćenja virtuelnih instrumenata
- Za projektovan interfejs, formira se programski kod u blok dijagramu
- Svaki isprogramirani virtuelni instrument može da se koristi kao podprogram (subVI) u nekom drugom VI

Blok dijagram

- U prozoru blok dijagrama se piše tj. projektuje programski kod virtuelnog instrumenta.
- Terminali koji se nalaze u blok dijagramu predstavljaju kontrole koje se nalaze na front panelu.
- Povezivanje kontrola (ožičavanje) i dodavanjem struktura i funkcija formira se programski kod LabVIEW-a.
- Osnovni elementi u blok dijagramu:
  - terminali (*terminals*)
  - čvorovi (*nodes, statments*)
  - žice (*wires*)
  - strukture (*structures*)

- Treminali u blok dijagramu su prezentacija kontrola sa front panela. Mogu biti ulazni (kontrole) i izlazi (indikatori)
- Čvorovi su elementi u blok dijagramu koji su predstavljeni ikonicama, i odgovaraju operatorima, funkcijama, podprogramima u tekstualnom programskom jeziku. Imaju ulaze i izlaze. One služe za izvršenje različitih programski instrukcija i izraza
- Žice služe za povezivanje terminala čvorova i struktura unutar blok dijagrama. One simbolizuju tok podataka
- Strukture predstavljau grafičke prezentacije programskih petlji *(For,While),* zatim *Case* struktura i sekvenci



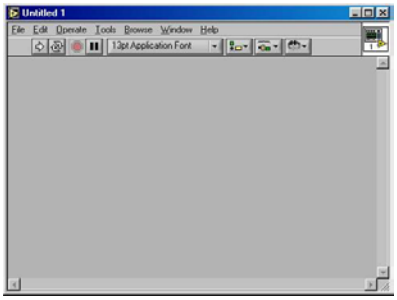
LabVIEW okruženje

- LabVIEW okruženje sastoji se od:
  - front panela,
  - blok dijgaram porzora,
  - paleta kontrola (*Controls Pallete*),
  - paleta funkcija (*Functions Pallete*),
  - paleta alata (*Tools Pallete*),
- Paleta kontrola (*Controls Pallete*) dostupna je samo kada je aktivan front panel prozor
- Paleta alata je dostupna u oba LabVIEW prozora. U ovoj paleti se vrši izbora alata koji se koristi pri grafičkom programiranju i različiti modovi pointetra miša: selektror, tekst, ožičavanje, *breakpoints*, bojenje i drugo

Paleta alata
Tools Pallete

Paleta kontrola
Controls Pallete

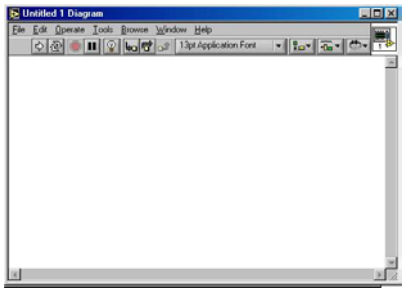
Front panel



LabVIEW okruženje

- Paleta funkcija (Function Pallete) dostupna je samo kada je aktivan blok dijagram prozor.
- U ovoj paleti nalazi se veliki broj programskih funkcija (numeričkih logičkih), programa i virtuelnih instrumentat za akviziciju analizu, obradu i prikazivanje podataka
- Paleta alata je dostupna u oba LabVIEW prozora. U ovoj paleti se podečavaju različiti modovi pointetra miša: selektro, tekst, ožičavanje, brakpoints, bojenje.
- Paleta funkcija (Function Pallete) dostupna je samo kada je aktivan blok dijagram prozor

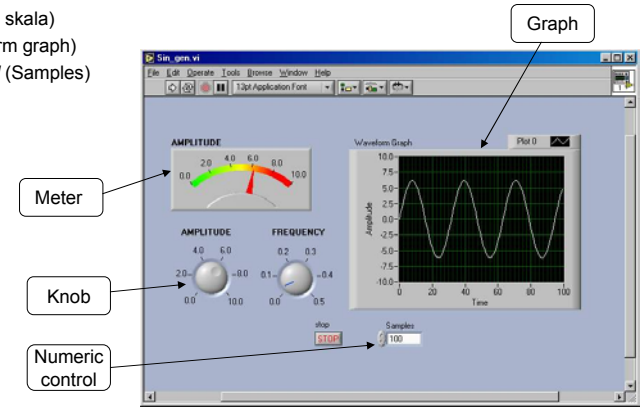
Function Pallete



Tools Pallete

Dizajniranje front panela

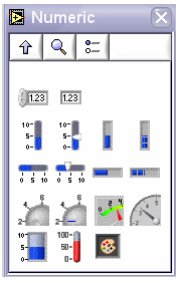
- Na front panel se postavljaju elementi po želji prevlačenjem sa palete kotrola. U okviru svake palete se nalaze nekoliko podpaleta sa većim izborom različitih kontrola
- Na slici je prikazan jedan primer virtulenog instrumenta koji generiše sinusni signal
- Elementi koji su korišteni su:
  - *Knob* (Amplitude, Frequency)
  - *Button* (Stop)
  - *Meter* (metrična skala)
  - *Graph* (Waveform graph)
  - *Numeric control* (Samples)

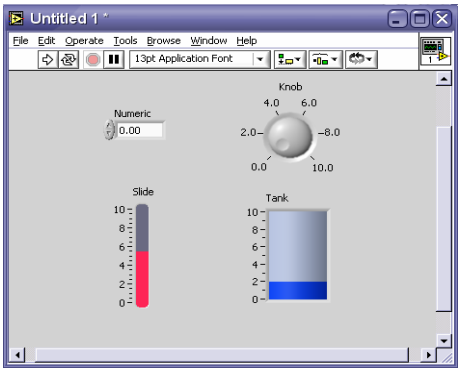


Numeričke kontrole

- Iz osnovne palete kontrola mogu se birati razne pod-palete
- Paleta numeričkih kontrola sadrži grafičke kontrole za zadavanje, kontrolu i prikazivanje numeričkih vrednosti u LabVIEW
- Većina kontrola vizuelno podseća na kontrole stvarnih fizičkih instrumenta
- Pored numeričkih kontrola nalaze se i numerički indikatori
- Na slici su prikazane neke kontrole koje se često koriste pri projektovanju virtuelnih instrumenata: *Numeric, Knob, Slide, Tank*

Numeric subpallete



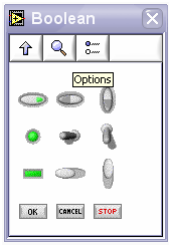


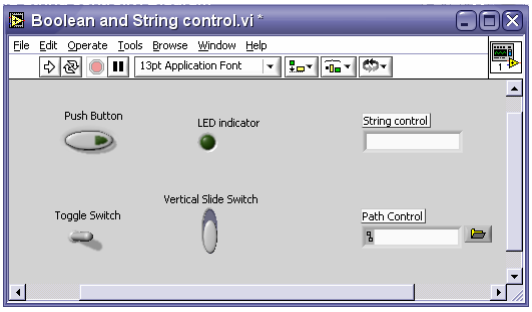
Boolean i *String* kontrole

- *Boolean* kontrole omogućavaju zadavanje logičkih *(boolean)* vrednosti preko korisnčkog interfejsa
- Najčešće su u obliku prekidača, koji mogu da imaju diskretne vrednosti u zavisnosti od položaja (On, Off)
- Boolean indikatori su najčešće predstavljeni LED indikatorima
- String kontrole obezbeđuju zadavanje tekstualnih poruka i naziva u programu
- U okviru ove pod palete nalaze se i *Path* kontrole, koje omogućuju zadavanje putanja za pristupanje fajlovima i aplikacijama
- Obe kontrole imaju i ekvivalentne indikatore

String subpallete

Boolean subpallete

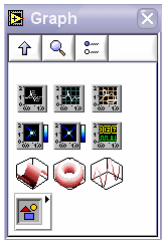


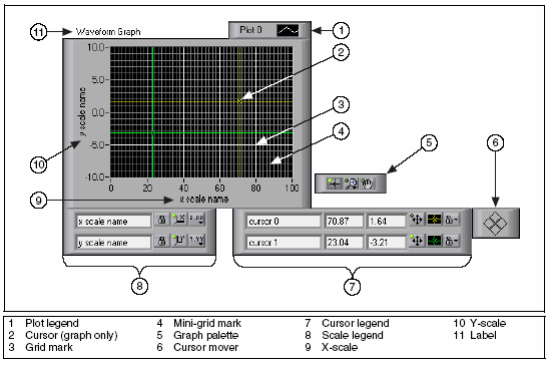


Grafici

- LabVIEW poseduje nekoliko tipova grafova: *Graph, Chart, XY Graph, 3D Graph, Digital Waveform Graph*
- Grafici služe za prikazivanje funkcija, merenih signala, rezultate obrade i analize
- Na jednom grafiku mogu se prikazivati više signala
- Mogu se podešavati i više skala različitih intertvala
- Razlika *graph* i *chart*: *chart* grafici imaju memoriju, tj pamte zadatu količinu tačaka bez obzira da li je prikazuju u tom trenutku

Graph subpallete

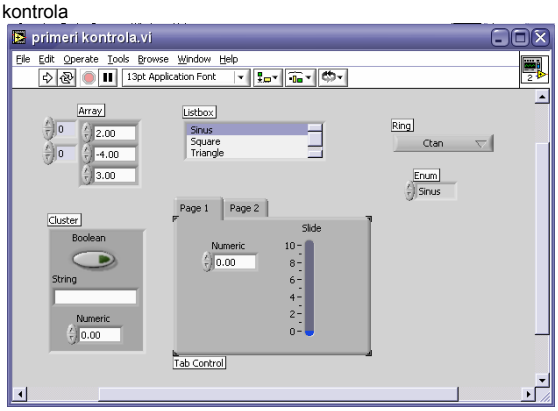




1 Plot legend	4 Mini-grid mark	7 Cursor legend	10 Y-scale
2 Cursor (graph only)	5 Graph palette	8 Scale legend	11 Label
3 Grid mark	6 Cursor mover	9 X-scale	

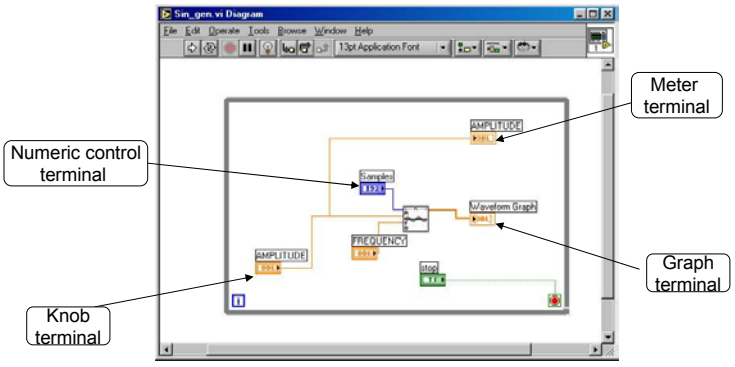
Primeri drugih kontrola

- U LabVIEW-u se nalaze i druge kontrole koje omogućavaju projektovanje kvalitetnog grafičkog korisničkog interfejsa
- U pod-paleti *List &Tables* se nalaze kontrole: *list box, tables*…
- U pod paleti *Rings&Enum* se nalaze kontrole*: Ring, Enum, Menu*…
- U pod paleti *Array&Cluster* se nalaze kontrole*: Array, Cluster, Tab*…
- U pod paleti *Dialog Controls* se nalaze kontrole dialoga, padajućeg menija*, radio button, checkbox* i druge
- Na slici su prikazani primeri nekih kontrola



Dizajniranje blok dijagrama

- Objekti koji se nalaze na front panelu predstaljeni su u blok dijagramu kao terminali
- Povezivanje kontrola (ožičavanje) i dodavanjem različitih programskih funkcija formira se programski kod LabVIEW-a.
- Na slici je primer blok dijagrama programa za generisanje sinusnog signal, čiji je front panel pirkazan na nekom od predhodnih slajdova
- Terminali nose ista imena kao i kontrole na front panelu



Tipovi podataka u LabVIEW

- LabVIEW je grafički programski jezik te je stoga svaki tip ili struktura podatka prikazan na specifičan način, korišćenjem različitih boja i oblika. Svaki tip ima svoju boju i oblik
- Spajanjem u programu žice takođe oslikavaju protok različitih tipova podataka. Žice su različite debljine i boje za različite tipove podataka
- U tabeli je dat prikaz tipova podataka u LabVIEW-u

Control	Indicator	Data Type	Color
		Single-precision floating-point numeric	Orange
		Double-precision floating-point numeric	Orange
		Extended-precision floating-point numeric	Orange
		Complex single-precision floating-point numeric	Orange
		Complex double-precision floating-point numeric	Orange
		Complex extended-precision floating-point numeric	Orange
		Signed 8-bit integer numeric	Blue
		Signed 16-bit integer numeric	Blue
		Signed 32-bit integer numeric	Blue
		Unsigned 8-bit integer numeric	Blue
		Unsigned 16-bit integer numeric	Blue
		Unsigned 32-bit integer numeric	Blue
		Enumerated type	Blue
		Boolean	Green
		String	Pink

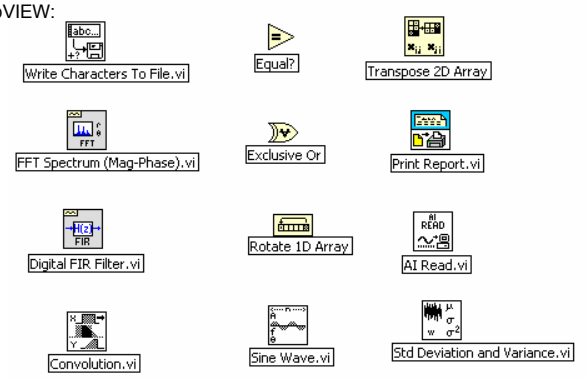
Tipovi podataka u LabVIEW - nastavak

Control	Indicator	Data Type	Color
		Array—Encloses the data type of its elements in square brackets and takes the color of that data type.	Varies
		Cluster—Encloses several data types. Cluster data types are brown if the elements of the cluster are the same type or pink if the elements of the cluster are different types.	Brown or Pink
		Path	Aqua
		Waveform—Cluster of elements that carries the data, start time, and Δt of a waveform. Refer to the *Waveform Data Type* section of Chapter 11, *Graphs and Charts*, for more information about the waveform data type.	Brown
		Reference number (refnum)	Aqua
		Variant—Includes the control or indicator name, the data type information, and the data itself. Refer to the *Handling Variant Data* section of this chapter for more information about the variant data type.	Purple
		Polymorphic—Indicates that a VI or function accepts more than one kind of data type. Refer to the *Polymorphic VIs and Functions* section of this chapter for more information about polymorphism VIs and functions.	Purple
		I/O name—Passes DAQ channel names, VISA resource names, and IVI logical names you configure to I/O VIs to communicate with an instrument or a DAQ device. Refer to the *I/O Name Controls and Indicators* section of Chapter 4, *Building the Front Panel*, for more information about the I/O name data type.	Purple
		Picture—Displays pictures that can contain lines, circles, text, and other types of graphic shapes. Refer to the *Using the Picture Indicator* section of Chapter 12, *Graphics and Sound VIs*, for more information about the picture data type.	Blue

Funkcije u LabVIEW-u

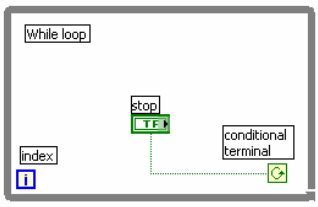
- U LabVIEW-u se nalazi veliki broj funkcija za rad sa bojevima, stringovima, booleanima, nizovima, funkcije koje omogućavaju obradu signala, merenja i analizu
- Funkcije u LabVIEW su: numeričke funkcije, strukture, boolean, string, nizovi - array, cluster, file I/O, waveform, dialog, DAQ, funkcije za analizu i obradu signala...
- Postoje i napredne funkcije za upravljanje aplikacijama (*advanced function, appliction control function*)
- Sve funkcije su predstavljene ikonicama, i poseduju ulaze i izlaze na koje se vezuju žice, čime se funkcije implementiraju u programski kod
- Primeri nekih funkcija u LabVIEW:

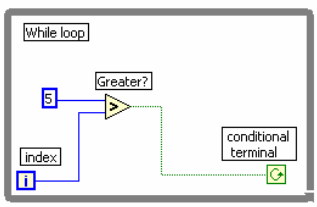
Paleta funkcija



While petlja

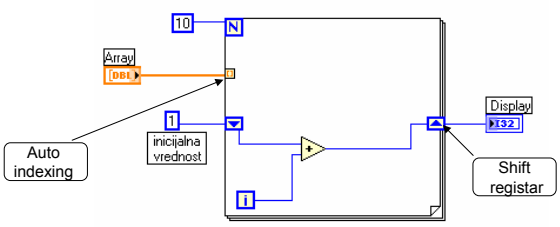
- U *LabVIEW*-u strukture u programu su prikazane grafički.
- Strukture u *LabVIEW* su: *While loop, For loop, Sequence, Case, Formula node, Event structure*
- While petlja (prikazana na slici) je predstavljena kao kvadrat sivih zadebljanjih stranica. Ovaj kvadrat može proizoljno da se smanjuje ili povećava.
- Sve što se nalazi unutar kvadrata predstavlja programski kod koji se izvršava unutar *While* petlje.
- Petlja se izvršava dok nije zadovoljen uslov. Uslov za zaustavljanje petlje se zadaje preko kondicionog terminala (conditional) koji je *Boolean* tipa
- *Index* terminal vrši brojanje iteracija *While* petlje
- Petlja može da se zaustavi signalom sa kontrole koja se nalazi na front panelu (npr. Stop) ili programski iz same petlje, ako je zadovoljen uslova zaustavljanja
- While petlja se ne može zaustaviti naredbom koja se zadaje van petlje u trenutku izvršenja





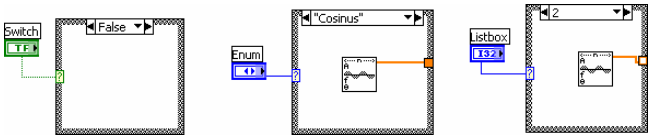
For petlja

- *For* petlja (prikazana na slici) je grafički predstavljena kao kvadrat. Ovaj kvadrat može proizoljno da se smanjuje ili povećava.
- Sve što se nalazi unutar petlje kvadrata predstavlja programski kod koji se izvršava unutar *For* petlje.
- Broj izvršavanja zadaje se preko *count "N"* terminla
- Iteracioni terminal *"i"* sadrži trenutni redni broj izvršenja *For* petlje
- Broj iteracija *For* petlje može biti i *Auto-indexing*, tj. ako se neka višedimenzionalna struktura (niz, matrica) dovede na ulaz petlje, tada će petlja imati onoliko iteracija kolika je dimenzija ulazne stukture (prikazano na slici)
- Po završetku petlje izlazni parameter se prosleđuje dalje u program.
- Kada se želi prenosti vrednost parametra untar petlje iz prethoden u sledeću iteraciju korsti se *Shift registar.* Ova opcija se bira na sledeći način: desnim klikom miša na ivicu *For* petlje, dobija se padajući meni iz kog se bira *Add Shift Registar.*
- Ovih registara može biti više u petlji, mogu biti svih tipova. Inicijalne vrednisti se mogu zadavati van petlje, a na kraju izvršenja petlje, poslednja vrednost parametra je ujedno i izlazna vrednost



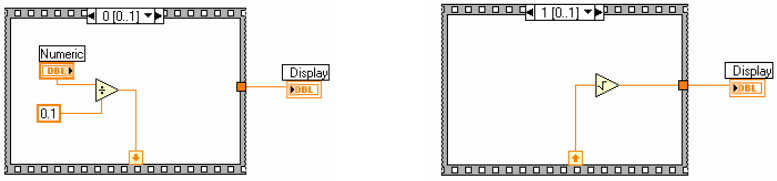
Case struktura

- *Case* struktura (prikazana na slici) predstavljena je grafički kao sivi uramljeni kvadrat. Ova struktura odgovara *Case* naredbi u standardnom tekstualnom programskom jeziku
- Kao prethodne grafičke strukture i ova može da se povećava ili smanjuje.
- Sve što obuhvati ova grafička struktura predstavlja programski kod koji se izvršava unutar *Case* izraza
- *Case* struktura sadrži više pod-dijagramskih struktura - "*cases*" (najmanje dve), koji predstavljaju programski kod za svaki od zadatih ulaznih uslova.
- Samo je jedan dijagram vidljiv, i samo se jedan pod-dijagram izvršava u jedno trenutku
- *Case selector* terminal predstavlja ulaz u *Case* struktura. On može bit različitog tipa: boolean, integer, enum ili string.
- U vrhu Case strukture nalazi se labela "*Case selector label"* koja sadrži trenutnu vrednost izabranog *case*-a
- Izlaz iz petlje predstavljen je kroz grafički tunel. Ukoliko se želi ilaz iz case-a, svi pod-dijagrmi moraju da imaju izlaz istog tipa. U protivnom program će javiti grešku



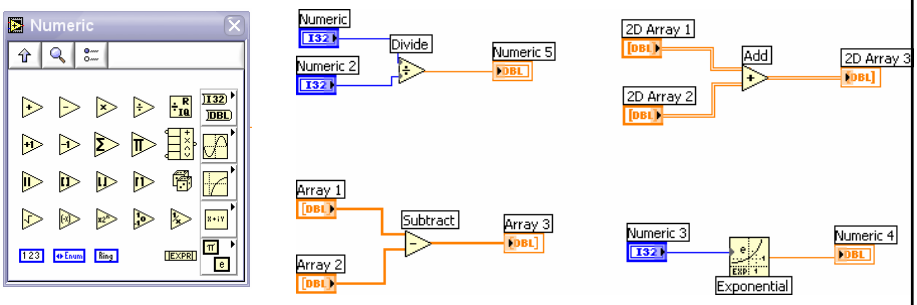
Sequence struktura

- *Sequence* struktura (prikazana na slici) je grafički predstavljena kao sivi uramljeni kvadrat.
- Sve što obuhvati ova grafička struktura predstavlja programski kod koji se izvršava unutar *Sequence* strukture
- *Sequnce* struktura sadrži više pod-dijagramskih struktura - "*sequences*" (najmanje dve), koji predstavljaju programski kod za svaki od zadatih ulaznih uslova.
- Služi za vođenje računa o redoslednom izvršenju delova programa
- Samo je jedan dijagram vidljiv, i samo se jedan pod-dijagram izvršava u jedno trenutku
- U vrhu *Sequence* strukture nalazi se labela "*Sequence label"* koja sadrži broj sekvence koja se trenutno izvršava
- Primenom sekvence omogućena je kontrola izvršenja toka programa
- Ako se žele prenositi parametri i vrednosti iz jedne u drugu sekvencu, koristi se *sequence local terminal*
- Ovaj terminal je grafički predstavljen strelicama, i simbolčno pokazuje smer toka podataka (strelica gore, strelica dole)



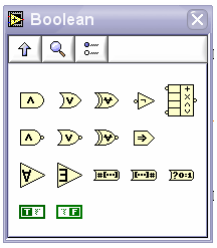
Numeričke funkcije

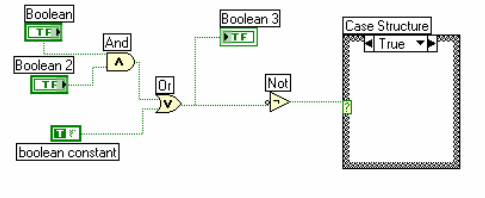
- LabVIEW poseduje implemeniran veliki broj funkcija koje vrše numeričke operacije
- Na slici je prikazana paleta Numeric (Function>>Numeric) u kojoj se mogu naći osnovne funkcije za sabiranje oduzimanje, množenje, korenovanje, trigonometrijske funkcije, logaritamske funkcije, rad sa kompleksnim brojevima i druge
- LabVIEW ima podržane operacije i za nizove i matrice (primer na slici)



Boolean funkcije

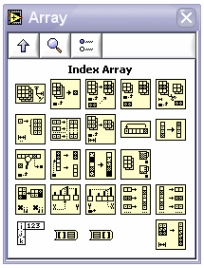
- LabVIEW poseduje implemeniran logičke - *boolean* funkcije
- Na slici je prikazana paleta Boolean funkcija u kojoj se mogu naći osnovne logičke funkcije: And, Or, Xor i druge (na slici)
- Ove funkcije se najčešćće koriste za dobijanje uslova potrebnih za izvršenje delova programa *Case* struktura

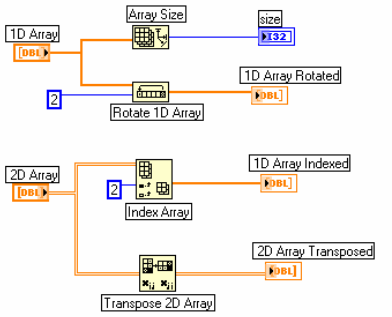




Funkcije za rad sa nizovima

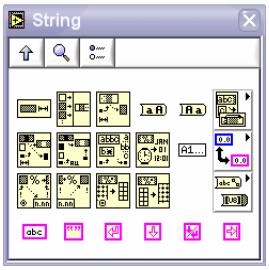
- LabVIEW poseduje implemenirane funkcije koje omogućavaju rad sa jednodimenzionalnim i višedimenzionalnim nizovima
- Funkcije za rad sa nizovima: pretraživanje, indeksiranje, rotiranje, spajanje, promena dimenzija, dodavanje dimenzija, dodavanje ili brisanje elemenata iz niza, transpozicija, pronalaženje minimalnog ili maksimalnog elementa, formiranje višedimenzionalnih struktura i druge
- Nizovi su skup elemenata istog tipa, koji su indeksirani. Najčešće se koriste jednodimenzionalni (vektori), dvodimenzionlni (tabele)
- Nizovi mogu biti formirani od gotovo svih tipova u LabVIEW: brojeva, stringova, boolean promenljivih, klastera.
- Elementima nizova se pristupa preko indeksa
- Nizovi u LabVIEW najčešće se koriste za smeštanje veće količine podataka pri merenjima, analizi i obradi signala

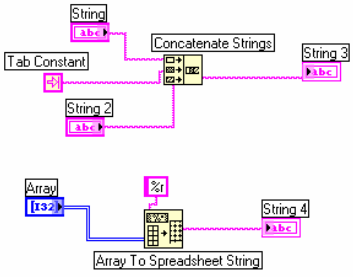




Funkcije za rad sa stringovima

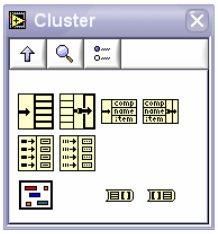
- LabVIEW poseduje veliki broj funkcija koje omogućavaju rad sa stringovima
- Stringovi se koriste za prikazivanje tekstualnih poruka, za prenos podataka između instrumentata, smeštanje podataka u fajlove i sl.
- Funkcije za rad sa stringovima omogućuju pretraživanje i zamenu karatkera unutar stringa, menjanje teksta, spajanje strigova, rotiranje, brisanje...
- String može da se predstavi u numeričkoj vrednosti i obrnuto. To je jedan od najčešćih načina korišćenja stringova u LabVIEW
- Da bi se podaci prikazali tekstualno ili u tabelarnoj formi koriste se isključivo stringovi
- Za prikaz numeričkih nizova u tabelarnoj formi (spreadsheet) koristi se konverzija brojeva u string

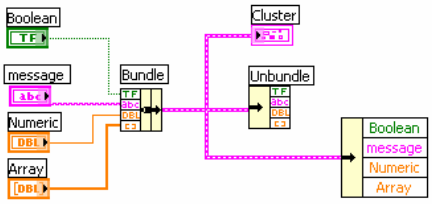




Cluster tipovi podataka

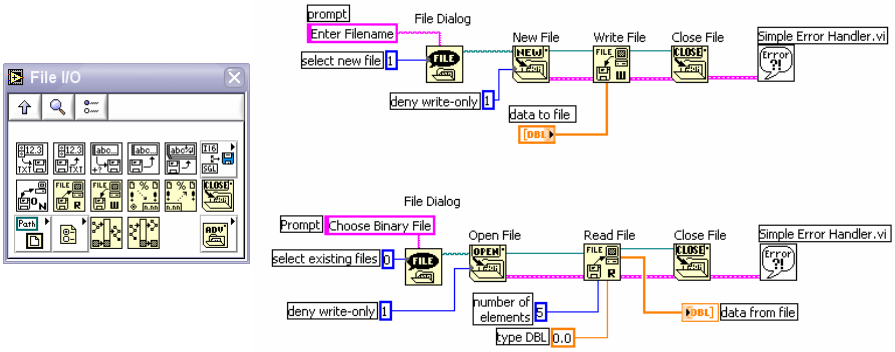
- *Clusteri* predstavljaju tip podataka koji je sastavljen od skup elemenata različitog tipa
- Klasateri su slični tipu *record* ili *struct* u tekstualnim programskim jezicima
- Formiraju se korišćenjem *bundle* konektora
- Skupljanjem elemenata različitog tipa u jedan formira se *record*, koji je predstavljen samo jednom žicom. Na taj način se smanjuje broj žica u blok dijagramu i povećava peglednost grafičkog koda
- Elementima unutar klastera prstupa se preko tipa. Mora se ceo klaster "rasformirati" (*Unbundle*) da bi se pristupilo elementu
- Elementima klasetera može se pristupiti i po imenu





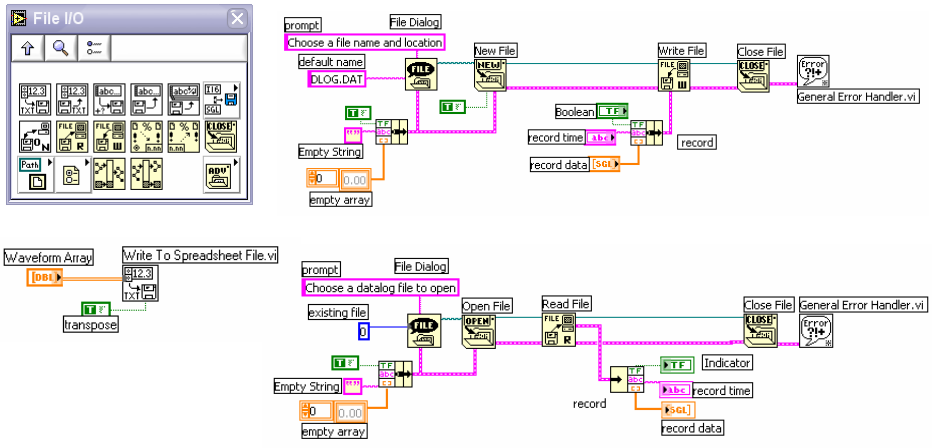
File I/O funkcije

- Za snimanje podataka u fajl i iščitavanje koriste se *File I/O* funkcije
- LabVIEW obezbeđuje snimanje u tri formata: tekstualni, binarni i datalog.
- Tekstualni se najčeše koristi jer se može čitati iz drugih aplikacija
- Binarni format se koristi u slučaju da se želi memorisati velika količina podata, recimo pri akviziciji
- Datalog format se koristi kada se želi formirati složeniji format podataka, i on može da se čita samo iz LabVIEW-a
- Ove funkcije obezbeđuju otvaranje i zatvaranje fajlova, upisivanje u fajlove i iščitavanje iz fajlova
- Takođe imaju podršku za rad sa podacima u tabelarnoj formi (*spread-sheet format*)



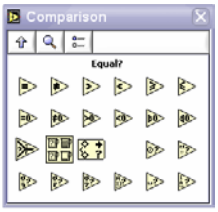
File I/O funkcije

- Tekstualni format se najčeše koristi jer se može čitati iz drugih aplikacija
- Datalog format se koristi kada se želi formirati složeniji format podataka (*record*), i on može da se čita samo iz LabVIEW-a
- *Record* se formira korišćenjem *Cluster* funkcija



Funkcije za komparaciju

- U funkcije za komparaciju spadaju standardne funkcije koje su podržane i u tekstualnim programskim jezicima
- To su: veće, manje, jednako, različito, jednako nuli i druge
- Primena ovih funkcija je veoma česta pri konroli izvršenja toka programa
- Može se vršiti kompracija gotovo svih tipova podataka u LabVIEW: brojeva, stringova, booleana, nizova, vešedimenzionalnih nizova, klastera...



Funkcije za kontrolu vremena i datuma

- U okviru LabVIEW postoje funkcije za kontrolu vremena i datuma
- Vremenski intervali se mogu proizvoljno zadavati. Najmanji interval je 1 ms, iako ovo vreme nije zagarantovano i uveliko je zavisno od zauzetosti porcesora (*system dependent*)
- Na istoj paleti se nalaze funkcije za rukovanje greškama
- Ovi podprogrami (*subVI*) obezbeđuju dialoge sa ispisanim kodovima greške, u slučaju da se pojavi greška pri izvršenju programa