Міністерство освіти і науки, молоді та спорту України
Національний технічний університет України
«Київський політехнічний інститут»

О.Г. Кисельова, А.В. Соломін

Технологія створення програмних продуктів

Програмування в NI LabVIEW

Навчальний посібник

Рекомендовано Методичною радою НТУУ «КПІ»

Київ
НТУУ «КПІ»
2012

УДК 004.438Ni_LabVIEW(075.8)
ББК 32.973.26-018.1 Ni_LabVIEWя73
К44

Рекомендовано Методичною радою НТУУ «КПІ»

(Протокол № 8 від 26.04.2012 р.)

Рецензенти: *С.Г. Стіренко,* канд. техн. наук, доц.,
Національний технічний університет України «Київський політехнічний інститут»
В.П. Денісюк, д-р фіз.-мат. наук, проф.,
Національний авіаційний університет

Відповідальний редактор: *Є.А. Настенко,* канд. техн. наук, д-р біологічних наук,
Національний технічний університет України «Київський політехнічний інститут»

К44 Технологія створення програмних продуктів. Програмування в NI LabVIEW [текст] : навч. посіб. / О.Г. Кисельова, А.В. Соломін. – К.: НТУУ «КПІ», 2012. – 200 с. – Бібліогр.: с. 197. – 100 пр.

Описано середовище розробки лабораторних віртуальних приладів NI LabVIEW, яке дозволяє реалізовувати системи збирання, оброблення та аналізу різних видів сигналів та зображень, зокрема медико-біологічних. Показано, як за допомогою NI LabVIEW можна розробити системи керування, тестування, симуляції різних процесів та явищ з метою автоматизації проведення експериментів та досліджень. Система NI LabVIEW інтегрується з мовою програмування С, з системою MatLab та пакетом Multisim, що суттєво розширює можливості її застосування.

Для проведення занять з дисципліни «Технологія створення програмних продуктів» напряму підготовки «Комп'ютерні науки», а також рекомендовано студентам, аспірантам та викладачам усіх спеціальностей, які вивчають сучасні інформаційні технології та займаються автоматизацією різних процесів.

УДК 004.438Ni_LabVIEW(075.8)
ББК 32.973.26-018.1 Ni_LabVIEWя73

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ASCIII	American Standard Code for Information Interchange	Американський стандартний код обміну інформацією
CIN	Code Interface Node	Спеціальний вузол блок-діаграми, за допомогою якого можна передати текстовий програмний код у програму
DLL	Dynamic link library	Бібліотека виконуваних функцій або даних, використовуваних застосуваннями Windows
DAQ	Data Acquisition	Збір даних
G		Мова графічного програмування у середовищі NI LabVIEW
GPIB	General purpose interface bus	Інтерфейс КОП (канал загального користування), він же – HP-IB (Hewlett Packard interface bus), він же – IEEE 488.2
HTML	HyperText Markup Language	Мова гіпертекстового маркування документів в мережі Internet
HTTP	HyperText Transfer Protocol	Протокол передачі гіпертексту. Мережевий протокол для отримання і передачі гіпертекстових документів і

		даних між браузером та сервером
I/O	Input/Output	Введення/виведення даних
LabVIEW	Laboratory Virtual Instrumentation Engineering Workbench	Середовище розробки віртуальних лабораторних приладів
LED	Light-emitting diode	Світлодіод
NI	National Instruments	Компанія – розробник системи LabVIEW
NI-DAQ	National Instruments - Data Acquisition	Набор драйверів для устаткування National Instruments
NI-MAX	National Instruments Measurement and Automation Explorer	Програма налаштування, що взаємодіє з NI-DAQ та дозволяє конфігурувати устаткування National Instruments
NRSE	Nonreferenced single-ended	Схема із загальний незаземленим дротом
OLE	Object linking and embedding	Технологія зв’язування та впровадження об’єктів в інші документи та об’єкти
OOP	Object-oriented programming	Об'єктно-орієнтоване програмування (ООП)
RSE	Referenced single-ended	Схема із загальний заземленим дротом
SubVI	Sub virtual instrument	Віртуальний підприлад (аналог підпрограми, ВПП),

		представляє собою віртуальний прилад (ВП), що викликається у програмному коді іншого ВП.
URL	Uniform Resourse Locator	Стандартизований спосіб запису адреси ресурсу у мережі Internet
USB	Universal Serial Bus	Унівесальна послідовна шина для з'єднання пристроїв
VI	Virtual instrument	Віртуальний прилад (ВП), представляє собою програму, розроблену у середовищі LabVIEW
XML	Extensible Markup Language	Мова маркування гіпертекста
ВП	Virtual Instrument	Віртуальний прилад
ВПП	Sub. Virtual Instrument	Віртуальний підприлад
ПП	Software	Програмний продукт
->		Позначення послідовності виклику пунктів вкладених меню або розділів діалогових вікон

1. ПРОГРАМНЕ СЕРЕДОВИЩЕ LABVIEW

LabVIEW або Laboratory Virtual Instrument Engineering Workbench (середовище розробки лабораторних віртуальних приладів) є середовищем графічного програмування, яке широко використовується в промисловості, освіті, науково-дослідних лабораторіях та ін. як стандартний інструмент для збору даних, управління та тестування приладів, обладнання, технологічних ліній, а також для автоматизації різних видів експериментів.

За допомогою програмного середовища LabVIEW можна розробляти програмно-апаратні комплекси тестування, вимірювання, обробки та аналізу даних і управління зовнішнім устаткуванням.

LabVIEW – це 32-х розрядний компілятор, який може створювати як автономні модулі (*.exe), так динамічні бібліотеки (*.dll) [1, 34].

1.1. Мета та задачі

Метою заняття є ознайомлення з середовищем розробки лабораторних віртуальних приладів NI LabVIEW. Для цього необхідно виконати такі задачі:

- дати визначення поняттю віртуальний прилад (ВП);
- охарактеризувати обробку потоку даних у LabVIEW;
- розібратися з організацією програмного середовища LabVIEW (вікна, меню, інструменти);
- навчитися користуватися довідковою системою середовища LabVIEW.

1.2. Вступна частина – повторення матеріалу за тематикою програмування на традиційних мовах

- Вимоги до якісного програмного продукту.
- Основні етапи програмування.
- Технології програмування.
- Компілятори і інтерпретатори.
- Моделі життєвого циклу програмного продукту.

1.3. Вступ у LabVIEW

NI LabVIEW – середовище розробки програмного забезпечення на графічній мові програмування високого рівня G. У NI LabVIEW можна створити будь-яку програму, від систем збору та обробки даних до комп'ютерної гри. Але основним призначенням LabVIEW є розробка прикладного програмного забезпечення для організації взаємодії з контрольно-вимірювальною апаратурою, збору, обробки і відображення інформації і результатів розрахунків, а також моделювання як окремих об'єктів, так і автоматизованих систем в цілому. За експертними оцінками економія часу на розробку таких систем в LabVIEW становить до 40%.

Останні версії LabVIEW містять функції для реалізації практично будь-якої задачі, але, якщо все ж таки потрібної функції не буде, то її можна реалізувати на мові C та у вигляді динамічної бібліотеки інтегрувати у LabVIEW. Система LabVIEW написана на мові C.

Розробником LabVIEW є американська компанія National Instruments.

На відміну від текстових мов, таких як C, Pascal та ін., де програми складаються у вигляді рядків тексту, в LabVIEW програми створюються у

вигляді графічних діаграм, подібних до звичайних блок-схем. **Ця своєрідна графічна мова називається «G».**

Графічний підхід до програмування дозволяє виключити безліч синтаксичних деталей, якими багаті текстові мови програмування, наприклад, крапки з комою або фігурні дужки. Спрощений синтаксис тепер не відволікає від аналізу самого алгоритму.

Програмування в системі LabVIEW максимально наближене до поняття алгоритм. Після того, як розробником запропоновано алгоритм роботи програми, залишиться лише «намалювати» блок-схему цього алгоритму з використанням графічної мови програмування "G".

Програма, написана в середовищі LabVIEW, називається віртуальним приладом **(ВП, Vitrual Instrument (VI)**). ВП можуть бути комп'ютерними моделями реальних фізичних приладів, наприклад, осцилографа або мультиметра. LabVIEW містить повний набір інструментів (драйверів, інструментаріїв тощо) для збору, аналізу та зберігання даних.

Робоча зона розробки програми у LabVIEW представляє собою два вікна: **Front Panel** (передня панель, тобто вікно для розробки інтерфейсу користувача) та **Block Diagram** (блок-діаграма, тобто вікно для розробки програмного коду на графічній мові програмування G).

У LabVIEW інтерфейс користувача створюється за допомогою елементів управління (кнопки, перемикачі та ін.) і відображення (графіки, світлодіоди та ін.). Після цього на блок-діаграмі ВП здійснюється програмування з використанням графічних представлень функцій для управління об'єктами на передній панелі.

1.3.1. Віртуальні прилади

Будь-яка програма, створена в системі LabVIEW, називається **віртуальний прилад** (ВП) (по-англійськи VI – **Virtual Instrument**). Саме тому розширенням файлів, розроблених у середовищі LabVIEW, є *.vi.

Компонентами, тобто складовими ВП, є передня панель, блок-діаграма і **піктограма (конектор, connector).**

Передня панель реалізує призначений для користувача інтерфейс ВП, дозволяє задавати вхідні дані і відображувати результати роботи ВП. Блок-діаграма є аналогом традиційної програми і реалізує функціональні можливості ВП. Налаштування піктограми (конектора) дозволяє використовувати ВП як підпрограму **(SubVI, віртуальний підприлад, ВПП)**. Як правило, ВП у LabVEIW мають ієрархічну структуру і складаються з набору ВПП.

1.3.2. Передня панель

На передній панелі (Front Panel) створюються елементи управління і відображення, які є інтерактивними засобами введення і виведення даних для ВП. **Елементами управління (controls)** є кнопки, перемикачі і інші види елементів для введення даних. **Елементами відображення (indicators)** є аналогічні елементи, але вони призначені для відображення результатів роботи програми.

Для розміщення елементів управління і відображення даних на передній панелі використовується палітра Controls. Палітра Controls доступна лише з передньої панелі. Для виводу на екран палітри Controls слід вибрати пункти головного меню Window -> Show Controls Palette або натиснути праву кнопку миші в робочому просторі передньої панелі.

1.3.3. Блок-діаграма

Блок-діаграма складається з **вузлів (Node), терміналів (Terminal) і провідників даних (Wire).**

Об'єкти передньої панелі на блок-діаграмі відображуються у вигляді **терміналів даних** (графічне зображення прямокутної форми з літеро-чисельними позначеннями). Літеро-чисельне позначення на терміналі даних визначає тип даних, який може використовуватися в елементах управління або відображення. Наприклад, DBL-термінал визначає, що даний елемент управління використовує числа подвійної точності з плаваючою комою.

Термінал даних може відображуватися у вигляді іконки. Для цього досить натиснути праву кнопку миші в полі терміналу даних і вибрати View as Icon (відображувати у вигляді іконки) з контекстного меню. Зняти мітку – для відображення в стандартному вигляді. Відображення терміналу даних в стандартному вигляді дозволяє зберегти місце на блок-діаграмі.

Термінали даних забезпечують обмін даними між передньою панеллю і блок-діаграмою, вони подібні змінним і константам у текстових мовах програмування.

1.3.4. Вузли

Вузли (Nodes) – це об'єкти на блок-діаграмі, які мають одне або більше полів введення/виведення даних і виконують алгоритмічні операції у ВП. Вони аналогічні операторам, функціям і підпрограмам у текстових мовах програмування. Вузли включають функції, віртуальні підпрограми (ВПП) і структури.

Підпрограма (Віртуальний підприлад, ВПП, SubVI) – віртуальний прилад, який можна використовувати на блок-діаграмі іншого

ВП як підпрограму.

Структури (Structure) – це елементи управління процесом, такі як структура Case (Структура варіанту вибору), цикл While (цикл за умовою) та інші [1, 74-78].

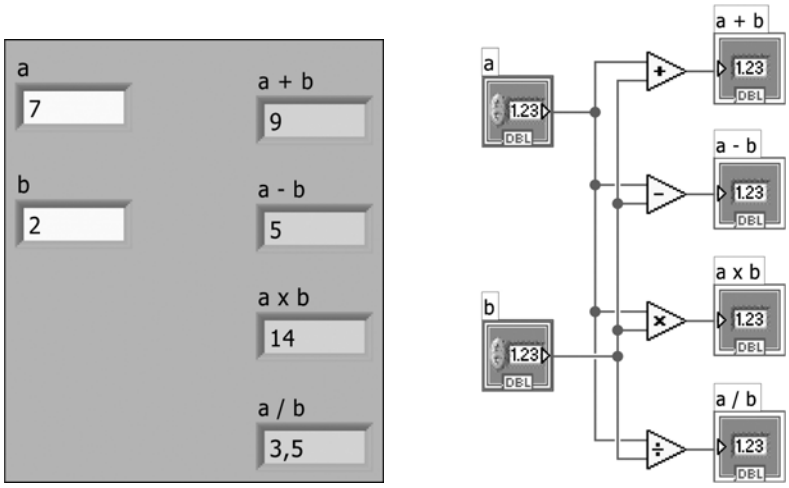
1.4. Приклади

Приклад 1.1. На рис. 1.1 зображено приклад реалізації ВП порівняння двох чисел, введених користувачем через числові елементи управління на передній панелі. Якщо Число1 більше або дорівнює Числу 2, то включається світлодіод (індикатор) під назвою "Число 1>=Число 2".

а) б)

Рис. 1.1. Приклад реалізації ВП порівняння двох чисел (а – передня панель (інтерфейс користувача), б – програмний код)

Приклад 1.2. На рис. 1.2 зображено приклад реалізації ВП для проведення математичних операцій з 2-ма числами, а саме після введення двох чисел (операндів) в управляючі елементи "a" і "b" і запуску програми на виконання у елементах-індикаторах відображається їх сума, різниця, добуток і частка від ділення.



а) б)

Рис. 1.2. Приклад реалізації ВП для проведення математичних операцій з 2-ма числами (а – передня панель (інтерфейс користувача), б – програмний код)

1.5. Завдання на самостійну роботу

Завдання 1

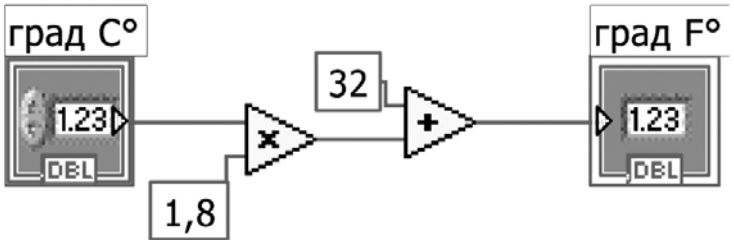
Розробити віртуальний прилад, який переводить значення температури за шкалою Цельсія в значення за шкалою Фаренгейту і навпаки.

$$t_F = t_c * 1{,}8 + 32,$$

де t_F – значення температури у градусах за школою Фарінгейту, t_c - значення температури у градусах за школою Цельсію.

Варіант реалізації інтерфейсу користувача та програмного коду для переводу градусів за школою Цельсію у градуси за школою Фаренгейту наведено на рис. 1.3.

а)



б)

Рис. 1.3. Приклад інтерфейсу користувача (а) та програмного коду (б)

Завдання 2

Розробити ВП, який імітує функції роботи програми «Scientific Calculator», що знаходиться у розділі «Стандартні програми» операційної системи Windows (рис. 1.4). Розробити максимально наближений графічний інтерфейс користувача.

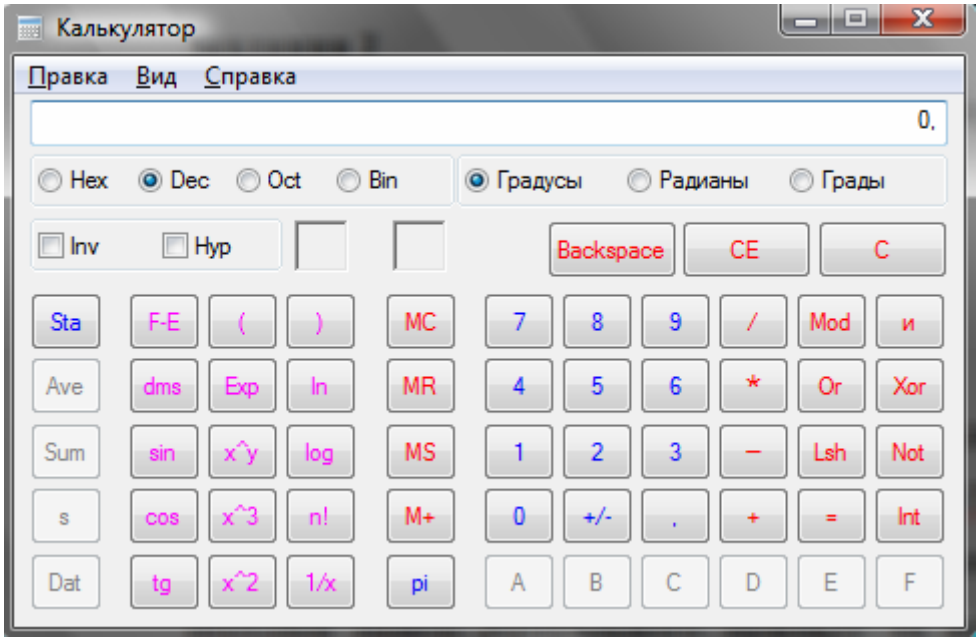


Рис. 1.4. Інтерфейс програми «Інженерний калькулятор» операційної системи Windows

1.6. Контрольні питання

1. Дайте визначення поняттю «Віртуальний прилад» та в чому полягає принцип розробки програм у середовищі LabVIEW?
2. Чим відрізняються між собою індикатор і елемент управління на передній панелі віртуального приладу та які властивості вони мають?
3. Що означає поняття «тип даних»? Яким чином в системі LabVIEW визначаються типи даних? Які типи даних існують?
4. Що означає "розірваний" зв’язок на блок-діаграмі?
5. Охарактеризуйте основні можливості системи LabVIEW.

2. РЕДАГУВАННЯ ТА НАЛАШТУВАННЯ ВІРТУАЛЬНИХ ПРИЛАДІВ

Найважливішим етапом роботи над програмним продуктом є його налаштування, а необхідними для цього складовими програмного середовища є засоби редагування і тестування.

LabVIEW містить всі необхідні компоненти і інструменти для реалізації цих функцій.

2.1. Мета та задачі

Метою заняття є вивчення можливостей створення програмних продуктів у середовищі LabVIEW. Для цього необхідно виконати наступні задачі:

- вивчити засоби редагування в LabVIEW;
- зрозуміти принцип управління потоком даних;
- розібратися із засобами налаштування в LabVIEW;
- закріпити знання про довідкову систему в LabVIEW.

2.2. Вступна частина – повторення матеріалу за тематикою попереднього заняття на прикладі домашніх завдань

- Панель інструментів **(Tools Pallet)** та як з нею працювати.
- Ефективність довідкової системи LabVIEW.
- Опис (документація), спливаючі підказки, коментарі, мітки – важливі складові програмного продукту.
- Дизайн і «дружній» інтерфейс користувача.

2.3. Основи редагування віртуальних приладів

У традиційних мовах програмування Basic, C++, Java і більшості інших порядок виконання програми визначається розташуванням функцій у програмі.

У середовищі LabVIEW використовується потокова модель обробки даних. Вузли блок-діаграми виконують закладені в них функції, якщо дані поступили на всі необхідні поля вводу/виводу. Після закінчення виконання операції одним вузлом результати по провідниках даних передаються наступному вузлу і так далі. Іншими словами, готовність вхідних даних визначає послідовність виконання вузлів блок-діаграми.

У середовищі LabVIEW **провідники (wires)** даних використовуються для з'єднання множини терміналів даних. Поля вводу/виводу мають бути сумісними з типами даних, що передаються їм по провідниках. Наприклад, не можна сполучати поле виведення масиву з полем введення даних чисельного типа. Компонентами, що визначають сумісність з'єднання, є тип даних елементу управління чи відображення і тип даних поля вводу/виводу.

Кожен провідник даних має єдине джерело даних, але може передавати їх багатьом ВПП та функціям всередині них. Провідники даних розрізняються кольором, стилем і товщиною лінії, залежно від типу даних, що передаються. Наприклад, для чисельного типу даних провідник помаранчевий (для чисел з плаваючою точкою) або блакитний (для цілих чисел), для логічних даних – зелений, для строкових – бузковий. Для простих даних – провідник тонкий; для масивів, кластерів та інших структур даних – товще і з певною структурою лінії [1, 78-79].

2.3.1. Інструменти, меню, палітри робочого вікна

Робоче вікно LabVIEW включає в себе головне меню у верхній частині вікна. Воно багато в чому стандартне та містить такі пункти як Open, Save, Copy, Paste, а також специфічні лише для LabVIEW розділи.

Контекстне меню (Context menu) використовується найчастіше. Всі об'єкти LabVIEW, робочий простір передньої панелі і блок-діаграми мають свої контекстні меню. Контекстне меню використовується для редагування об'єктів блок-діаграми і передньої панелі. **Контекстне меню викликається натисненням правої кнопки миші на відповідному об'єкті.**

Окрім головного меню LabVIEW має три допоміжні палітри, що використовуються для створення, редагування і виконання ВП: **Tools Palette (Палітра Інструментів), Controls Palette (Палітра Елементів) і Functions Palette (Палітра Функцій).** Ці палітри можна помістити в будь-якому місці екрану.

Палітра Інструментів (**Tools Palette**) доступна як на передній панелі, так і на блок-діаграмі. Термін інструмент пов'язано зі спеціальним операційним режимом курсору миші. При виборі певного інструменту значок курсору змінюється на значок даного інструменту. Палітра Інструментів доступна через пункт головного меню Window -> Show Tools Palette (але може бути інакше у різних версіях LaBVIEW). Утримуючи натиснутою клавішу <Shift> і натиснувши праву кнопку миші, можна вивести на екран тимчасову версію Tools Palette (Палітри Інструментів).

Палітра Елементів (**Controls**) і палітра Функцій (**Functions**) містять розділи, в яких розміщені об'єкти для розробки ВП. При натисненні на значок розділу на екран виводиться вікно, що містить його об'єкти. Для використання об'єкту палітри слід натиснути на ньому лівою кнопкою

миші і помістити вибраний об'єкт на передню панель або блок-діаграму ВП.

Палітра Елементів (**Controls**), рис. 2.1, використовується для розміщення елементів управління і відображення на передній панелі. Вона доступна лише на передній панелі. Аби відображувати палітру Елементів, слід або вибрати в пункті головного меню Window -> Show Controls Palette, або натиснути праву кнопку миші в робочому просторі передньої панелі. Використовуючи кнопку у верхньому лівому кутку палітри (подібну кнопці), можна зафіксувати її на екрані. За умовчанням палітра Елементів з'являється в експрес-вигляді і містить лише найбільш часто використовувані елементи. Використовуючи кнопку All Controls, що знаходиться в правому нижньому кутку, можна відобразити всі елементи на палітрі.

Рис. 2.1. Приклад палітри Елементів (Controls)

Палітра Функцій (**Functions**), див. рис. 2.2, використовується для створення блок-діаграми. Вона доступна лише у вікні блок-діаграми. Аби відображувати палітру Функцій, слід або вибрати в пункті головного меню Window -> Show Functions Palette, або натиснути праву кнопку миші в робочому просторі блок-діаграми. Використовуючи кнопку у верхньому лівому кутку палітри можна зафіксувати її на екрані. За умовчанням палітра Функцій з'являється в експрес-вигляді і відображує экспрес-ВП. **Експрес-ВП** – вузли функцій, які можна налаштувати за допомогою діалогового вікна. Вони використовуються для реалізації найбільш часто використовуваних функцій.

Рис. 2.2. Приклад палітри Функцій (Functions)

Вікно контекстної довідки **Context Help** допомагає при створенні і редагуванні ВП. Детальніша інформація розташована в LabVIEW Help (Вбудована допомога).

Вікно **Context Help (контекстної довідки)** виводиться на екран з пункту головного меню Help -> Show Context Help або введенням

<CTRL-H> з клавіатури.

При наведенні курсора на об'єкт передньої панелі або блок-діаграми у вікні Context Help (контекстної довідки) з'являються ікони підпрограми, ВП, функції, константи, елементів управління або відображення даних з вказівкою всіх полів вводу/виводу даних. При наведенні курсора на опції діалогового вікна у вікні Context Help (контекстної довідки) з'являється опис цих опцій. При цьому поля, **обов'язкові** для з'єднання, виділені жирним шрифтом, поля, що **рекомендуються** для з'єднання, представлені звичайним шрифтом, а **додаткові (необов'язкові)** поля – виділені сірим або взагалі не показані [1, 101-121].

2.3.2. Налаштування віртуальних приладів

Якщо ВП не запускається, це означає, що він не готовий до роботи. В процесі створення або редагування ВП **кнопка Run** набирає вигляду розірваної стрілки. Якщо після завершення редагування блок-діаграми стрілка все ще має розірваний вигляд, то ВП працювати не буде.

Для відображення списку помилок потрібно натиснути **кнопку Run** або обрати пункт головного меню Windows -> Show Error List. У вікні "Список помилок" буде перераховано всі допущені помилки. Після подвійного клацання лівою кнопкою миші на описі помилки виділиться об'єкт, що містить цю помилку.

Режим анімації («з підсвіткою») виконання блок-діаграми активується клацанням правої кнопки миші по кнопці **Highlight Execution.** Після натиснення кнопки «лампочка» активізується режим візуалізації виконання алгоритму, потоку управління та всіх проміжних значень виконання роботи програми. При цьому числові значення даних, що передаються, відображуватимуться на екрані у вигляді спливаючих вікон.

Цей режим використовується для покрокового налаштування ВП і спостереження за виконанням блок-діаграми.

Режим покрокового налаштування ВП використовується для перегляду виконання ВП на блок-діаграмі. Активація покрокового режиму здійснюється натисненням кнопок **Step Over** або **Step Into** на інструментальній панелі вікна Блок-діаграми. Аби побачити підказку, слід помістити курсор поверх кнопок Step Over, Step Into або Step Out. Підказка описує подію, яка відбудеться після натиснення цієї кнопки. Покроковий режим можна використовувати і для перегляду виконання підпрограми ВП [1, 182-191].

2.4. Приклади

Приклад 2.1. Зображений на рис. 2.3 віртуальний прилад демонструє особливості функціонування управління потоком даних в LabVIEW. Розглянемо блок-діаграму, в якій додаються два числа, а потім отримана сума множиться на 2. В цьому випадку блок-діаграма виконується зліва направо не тому, що об'єкти розміщені в цьому порядку, а тому, що одне з полів введення функції множення не визначене, поки не виконалася функція додавання і не передались дані до функції множення, що демонструє незвичайні для текстових мов програмування особливості визначення порядку виконання операцій. Те, що сума (a + b) виконується першою, в текстових мовах вимагало б взяття її в дужки, а тут не лише немає дужок, але навіть порядок розташування операцій несуттєвий, важливе з'єднання провідників. Не слід забувати, що вузол функції (у даному випадку це сума та добуток двох чисел) виконується лише тоді, коли визначені всі його поля введення даних.

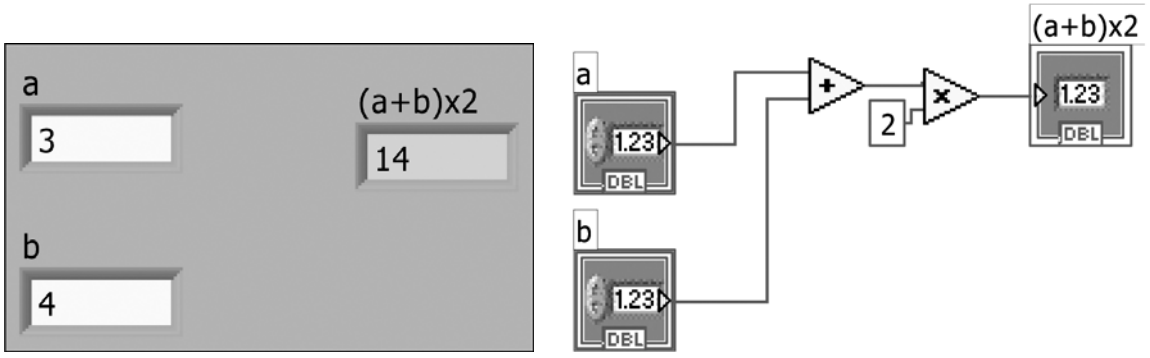
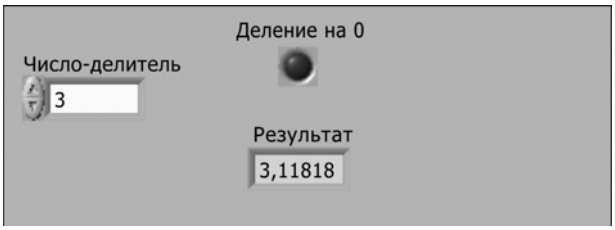


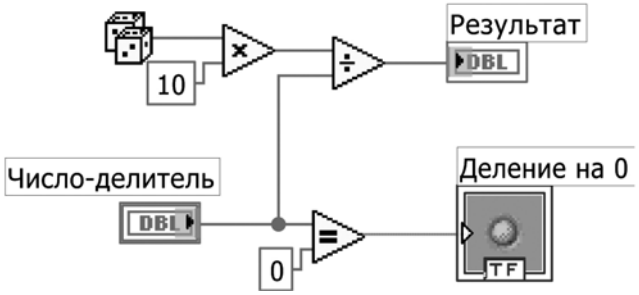
Рис. 2.3. ВП, що демонструє особливості функціонування управління потоком даних

Приклад 2.2. На рис. 2.4 зображено ВП генерації випадкового числа в діапазоні від 0.0 до 10.0, яке потім ділиться на введене з передньої панелі число-дільник. Результат відображується на передній панелі, а якщо проводиться спроба ділення на 0, запалюється світлодіод під ім'ям "Ділення на 0".

У даному прикладі використовуються функції генератора випадкових чисел, множення, ділення, порівняння двох чисел.



а)

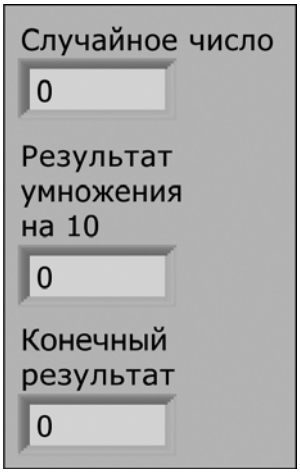


б)

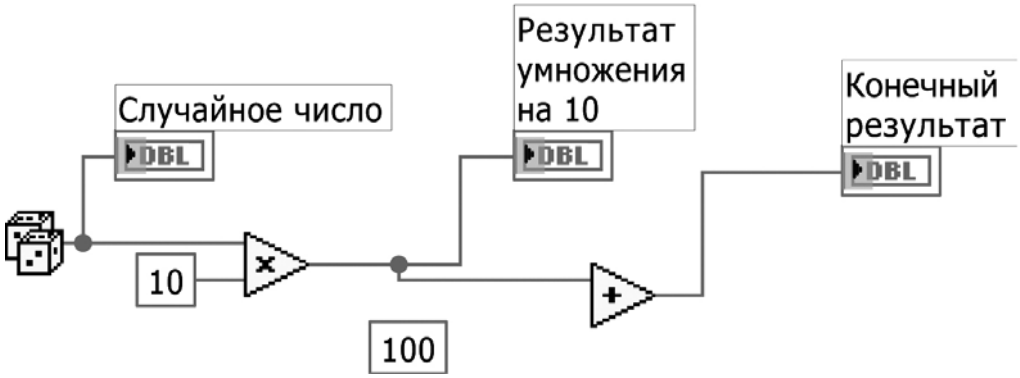
Рис. 2.4 (а, б). ВП, що ділить випадкове число на введене з передньої панелі число-дільник (а – інтерфейс користувача, б – програмний код)

2.5. Завдання на самостійну роботу

Знайти і усунути несправності в віртуальному приладі, зображеному на рис. 2.5. Крім того, для відладки роботи програми необхідно використовувати інструменти налаштування програм, включаючи підсвічування під час виконання, режим покрокових дій і використання пробника.



а)



б)

Рис. 2.5 (а, б). ВП з помилкою, яку слід виявити і усунути (а – інтерфейс користувача, б – програмний код з помилкою)

2.6. Контрольні питання

1. Як працювати з контекстною допомогою?
2. У чому особливості управління потоком даних?
3. Що таке пробник і що таке контрольна точка?
4. Як в LabVIEW організована робота з кольорами?
5. Які Ви знаєте методи і інструменти для пошуку несправностей?
6. Яким чином в LabVIEW розробляється графічний інтерфейс користувача і які інструменти для цього існують?

3. ІЄРАРХІЧНА СТРУКТУРА ВІРТУАЛЬНИХ ПРИЛАДІВ. ВІРТУАЛЬНІ ПРИЛАДИ І ПІДПРИЛАДИ

3.1. Мета та завдання

Метою заняття є вивчення модульного підходу до розробки програмних засобів. Для цього необхідно вирішити наступні задачі:

- навчитися ефективно користуватися ієрархічною структурою віртуальних приладів;
- з'ясувати та засвоїти основні вимоги для створення і оформлення повноцінного якісного програмного продукту.

3.2. Вступна частина

У середовищі LabVIEW віртуальні прилади мають модульну (ієрархічну) структуру. Це означає, що будь-який ВП може використовуватися у іншому в якості **ВПП (підприладу).** Можливість представлення ВП набором ВПП дає можливість незалежної розробки та тестування великих програмних систем. Модульна ієрархія дозволяє ефективно розробляти, модифікувати, замінювати та комбінувати різні ВП згідно потреб користувача на етапах розробки та супроводження програмного продукту.

3.3. Опис технології віртуальних підприладів

Без досконалого знання ієрархічної природи віртуальних приладів (ВП) неможливе ефективне використання всіх переваг LabVIEW. Суть в тому, що будь-який ВП може використовуватись іншим ВП в якості віртуального підприладу (ВПП). Якщо ВП є аналогом програми класичних

мов програмування, то ВПП є аналогом підпрограми, об’єкту, класу, бібліотеки тощо, в залежності від мови програмування.

Новий ВП можна використовувати в якості ВПП на блок-діаграмі ВП вищого рівня при умові реалізації в ньому **вузла даних**, який передбачає власну іконку **(Icon)** та «активізацію» з’єднувальної панелі, яку ще називають піктограмою **(Connector).**

Блок-діаграма може містити декілька однакових вузлів даних ВПП, які викликають один і той же ВПП кілька разів. Розміщення існуючого ВП на блок-діаграмі, аби користуватися ним як ВПП виконується за допомогою кнопки Вибрати ВП (**Select a VI**) в меню палітри Функції. Вибір цієї опції викликає діалогове вікно менеджера файлів, в якому можна вказати будь-який ВП. Після цього його іконка з'явиться на блок-діаграмі.

3.3.1. Створення іконки

Кожен ВПП повинен мати іконку для представлення його на блок-діаграмі вищого за ієрархією ВП. Іконка – це графічний символ ВПП. Іконку можна створити, вибравши опцію Редагувати іконки (**Edit Icon**) з контекстного меню іконки у верхньому правому кутку передньої панелі. Відкрити **Редактор** іконки можна, двічі клацнувши мишею по іконці. З'являється вікно **Редактора**. Кількість функцій, шаблонів та можливостей створення іконок відрізняються для різних версій LabVIEW.

3.3.2. Призначення функцій терміналам з’єднувальної панелі

Перш ніж використовувати ВП в якості ВПП, необхідно задати функції терміналам з’єднувальної панелі аналогічно тому, як задаються параметри для підпрограми в класичній мові програмування. З’єднувальна

панель (**Connector**) – це механізм LabVIEW для передачі даних у ВПП і повернення їх з нього. З'єднувальна панель ВП пов'язує елементи управління (control) та відображення (indicator) з вхідними і вихідними параметрами. Аби побачити з'єднувальну панель приладу, треба натиснути праву кнопку миші на іконці приладу і вибрати опцію **Показати з'єднувальну панель (Show Connector)**, і навпаки, щоб знову побачити іконку, – натиснути праву кнопку миші на з'єднувальній панелі і вибрати пункт **Показати іконку (Show Icon)**. LabVIEW формує з'єднувальну панель, базуючись на числі елементів управління і індикації передньої панелі приладу. Якщо треба вибрати іншу з'єднувальну панель, використовуйте меню Шаблони (Patterns) з контекстного меню з'єднувальної панелі. Можна також обертати і перевертати з'єднувальну панель, якщо вона незручно розташована в просторі, – для цього служать команди в контекстному меню панелі.

Для призначення відповідності терміналу і елементу управління або індикації необхідно виконати наступні операції:

1. Натиснути мишею по терміналу з'єднувальної панелі. Курсор автоматично перетвориться на інструмент з'єднання і забарвиться в чорний колір.

2. Натиснути мишею по елементу управління або індикатору, який має бути призначений відповідним вибраному терміналу. Цей елемент обрамлюється рухомою пунктирною лінією.

3. Натиснути мишею по відкритій області передньої панелі. Пунктирна лінія зникне, а термінал забарвиться в колір, відповідний типові призначеного елементу, що означає правильне підключення елементу управління або індикації до терміналу.

Якщо термінал має білий або чорний колір, то підключення є неправильним. Повторіть у разі потреби алгоритм підключення, описаний вище. З’єднувальна панель ВП може включати до 28 терміналів.

В разі помилки скористайтеся кнопкою **Відключити (Disconnect)** з контекстного меню з’єднувальної панелі, аби відключити певний термінал, або кнопкою **Відключити все (Disconnect All)**, аби відключити всі термінали. Крім того, всі підключення (конектори) діляться на 3 види: обов’язкові (Required), рекомендовані (Recommended) та необов’язкові (Optional).

3.3.3. Створення віртуального підприладу з блок-діаграми

В системі LabVIEW є можливість створення ВПП шляхом перетворення частини коду вихідного ВП. Для цього потрібно виділити той фрагмент блок-схеми, який необхідно перетворити на ВПП, а потім вибрати опцію **Створити ВПП (Create SubVI)** з меню Edit. Система замістить виділену частину ВП підприладом та автоматично задасть значення всім терміналам з’днювальної палені. Після цього можна двічі клацнути по новому ВПП, аби побачити його передню панель, відредагувати іконку, перевірити з’єднувальну панель і зберегти під новим ім'ям.

3.3.4. Документування роботи

Документування ВП є важливим процесом. Воно необхідне, аби інші люди розуміли призначення і функціонування ВП, а також є необхідним на етапі супроводження програмного продукту.

Опція **Опис і підказка (Description and Tip)** з контекстного меню об'єкту дозволяє створити спливаючу підказку до кожного елемента (Tip) та опис його функціонального призначення (Description). LabVIEW показує

текст опису в контекстному вікні допомоги (Contex Help), коли курсор підводиться до елементу управління або індикації на передній панелі.

Підказка відображується як спливаючий напис при зупинці курсору на об'єкті передньої панелі в режимі роботи ВП незалежно від того, відкрито вікно допомоги чи ні.

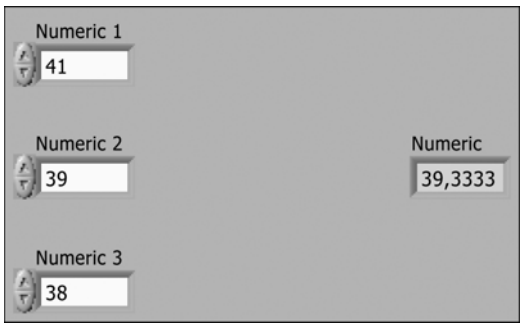
Найбільш повний опис приладу здійснюється у закладці **Documentation** вікна **VI Properties** меню **File.** Ця інформація потім з‘являтиметься у вікні контекстної допомоги і в довідковій системі LabVIEW.

Кращим способом організації швидкої допомоги при роботі з ВП є введення підказок і описів для всіх їх елементів управління і індикації, а також для функцій приладу. Крім того доцільне використання коментарів на елементи програмного коду в області блок-діаграми [1, 194-209].

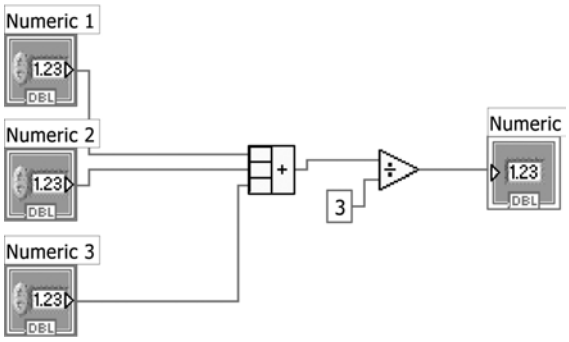
3.4. Завдання на самостійну роботу

Розробити ВП, що включає у себе декілька ВПП, а також розробити документацію на саму програму, на кожен з її елементів, включаючи вспливаючі підказки, а також написати коментарі до програмного коду.

Прикладом ВПП може бути програма обчислення середнього арифметичного значення трьох чисел, що задаються користувачем (рис. 3.1).



а)

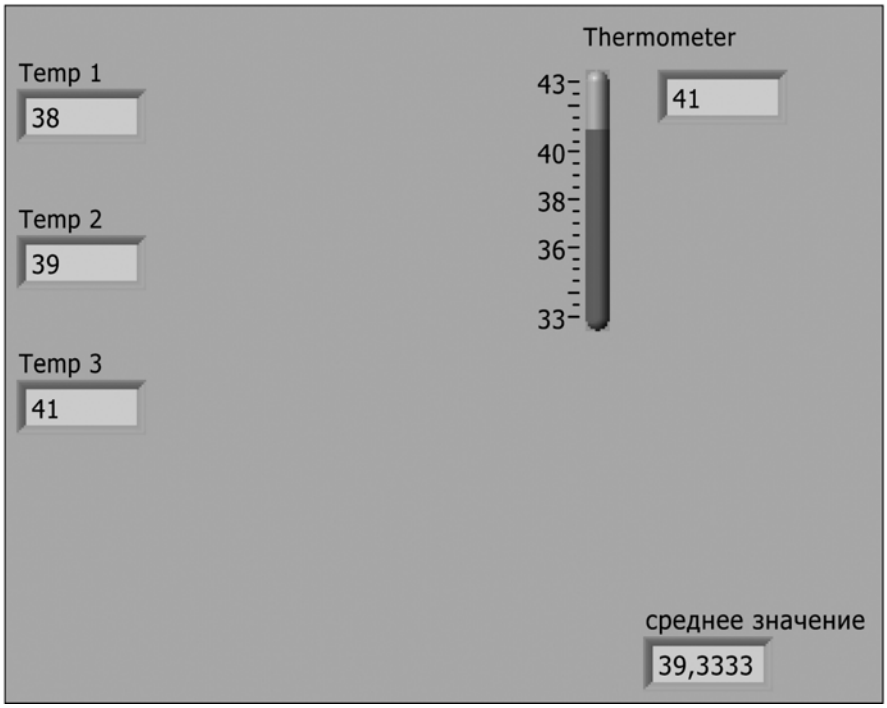


б)

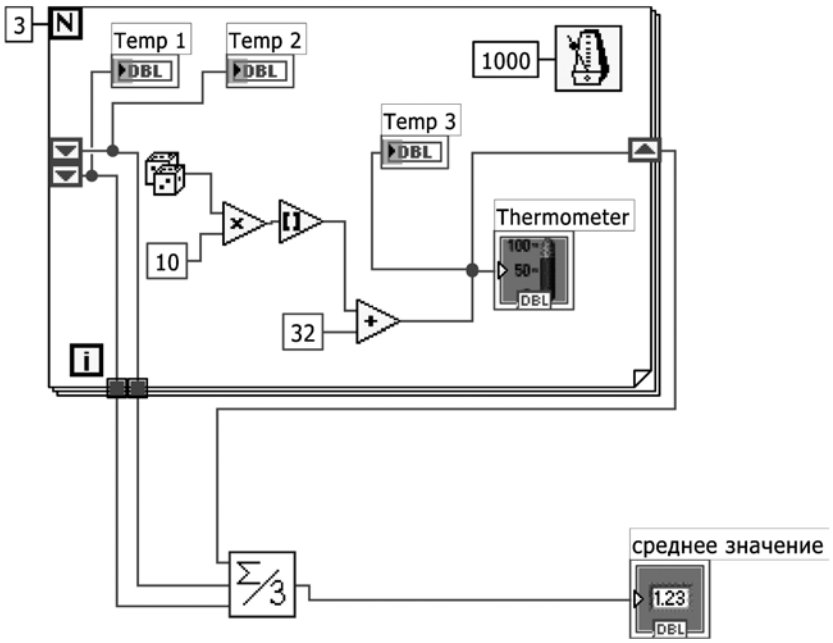
Рис. 3.1 (а, б). ВП для обчислення середнього арифметичного значення трьох чисел (а – передня панель, б – блок-діаграма)

Прикладом ВП, може бути прилад, що імітує вимір температури пацієнта з частотою 1 раз за секунду, а потім обчислює середнє значення (рис. 3.2).

Для реалізації даного алгоритму доцільним є використання **зсувних регістрів (Shift registers**), див. Розділ 4.



а)



б)

Рис. 3.2 (а, б). ВП, який імітує вимір температури пацієнта 3 рази, а потім обчислює середнє значення із застосуванням ВПП (а – передня панель, б – блок-діаграма)

3.5. Контрольні запитання

1. Охарактеризуйте основні критерії якісного програмного продукту.
2. Охарактеризуйте процес створення ВПП у LabVIEW.
3. Які можливості документування є у системі LabVIEW?

4. УПРАВЛІННЯ ВИКОНАННЯМ ПРОГРАМ ЗА ДОПОМОГОЮ СТРУКТУР. ЦИКЛИ

Структури в LabVIEW – це елементи програмного коду, що управляють порядком та умовами виконання операцій в програмі. Основними структурами управління є цикли (**Loop**). В LabVIEW існує 2 види циклів: цикл за умовою (**While loop**) та цикл з фіксованим числом ітерацій (**For loop**). Крім того, слід зазначити, що у LabVIEW структура **While loop** відповідає роботі алгоритму **Do-While,** і тому незалежно від виконання умови цикл виконається хоча б один раз.

4.1. Мета та завдання

Метою заняття є ознайомлення з управляючими структурами повторення у системі LabVIEW, для чого було поставлено такі задачі:

- навчитися застосовувати цикл за умовою і цикл з фіксованим числом ітерацій і зрозуміти відмінності між ними;
- вивчити особливості використання зсувних регістрів, відсутніх в інших мовах програмування. Переконатися в їх необхідності і ефективності;
- закріпити знання з використання віртуальних підприладів (SubVI).

4.2. Вступна частина. Повторення матеріалу попереднього заняття

- Вимоги до якісного програмного продукту.
- Основні етапи створення якісного ВП, придатного для використання в якості ВПП:
  - функціонально обґрунтований і «дружній» інтерфейс користувача,
  - оптимальна побудова блок-діаграми, її тестування,

- дизайн іконки,
- з'єднувальна панель,
- документування всього приладу і його елементів, коментарі.

4.3. Структури циклів в LabVIEW

У LabVIEW реалізовано дві структури циклів для управління операціями, що повторюються, у віртуальному приладі: цикл з фіксованим числом ітерацій (For Loop) і цикл за умовою (While Loop). Цикл з фіксованим числом ітерацій виконується зазначену кількість разів, а цикл за умовою виконується до тих пір, поки певна умова не стане виконаною. Обидва цикли знаходяться в підпалітрі Структури (Structures) палітри Функції.

4.3.1. Цикл з фіксованим числом ітерацій

Цикл з фіксованим числом ітерацій (For Loop) виконує код в його межах (піддіаграму) деяке число ітерацій (count). Це число дорівнює величині, введеній в термінал числа ітерацій „N” (count terminal). Його можна встановити, подаючи певне значення ззовні циклу на термінал числа ітерацій. Якщо підключити до цього терміналу значення 0, цикл не виконуватиметься, а на його вихідних терміналах будуть значення за умовчанням.

Термінал лічильника ітерацій „i” (**iteration terminal**) містить поточне число завершених ітерацій циклу: 0 – під час першої ітерації, 1 – під час другої і так далі до N-1, де N – кількість виконань циклу, яке задане в терміналі числа ітерацій.

Цикл з фіксованим числом ітерацій (рис. 4.1) еквівалентний наступному псевдокоду:

```
for i = 0 to N-1
Execute subdiagram
```

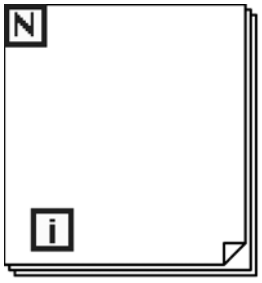


Рис. 4.1. Цикл з фіксованим числом ітерацій

4.3.2. Цикл за умовою

Цикл за умовою (While Loop) виконує код до тих пір, поки логічне значення (Boolean value), підключене до терміналу умови виходу з циклу (**conditional terminal**), не перейде в стан ІСТИНА (True) («так» – зупинити цикл або навпаки, при зміні умов виходу із циклу). LabVIEW перевіряє термінал умови виходу після закінчення кожної ітерації. Якщо значення ХИБНІСТЬ (False), то виконується наступна ітерація. За умовчанням умовою виходу є значення терміналу ІСТИНА.

Термінал лічильника ітерацій (**iteration terminal, i**) поводиться точно так, як і у випадку циклу з фіксованим числом ітерацій, тобто відлік йде від 0 до (N-1), де 0 – кількість ітерацій після першого «проходу» циклу, а (N-1) – кількість ітерацій після N разів виконання циклу. Цикл за умовою (рис. 4.2) еквівалентний наступному псевдокоду:

```
Do
Execute subdiagram
While condition is FALSE
```

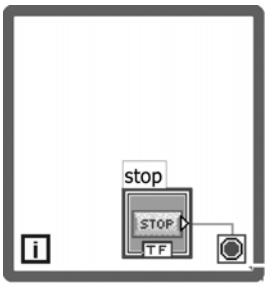


Рис. 4.2. Цикл за умовою

Можна змінити стан, який перевіряє термінал умови виходу з циклу, а саме замінити виконання циклу доки ХИБНІСТЬ (while false) на виконання доки ІСТИНА (while true). Для цього необхідно натиснути праву кнопку миші на терміналі умови і вибрати опцію Продовжити, якщо Істина (**Continue if True**) (рис. 4.3).

Такий алгоритм описується наступним псевдокоду:

```
Do
Execute subdiagram
While condition is TRUE (NOT FALSE)
```

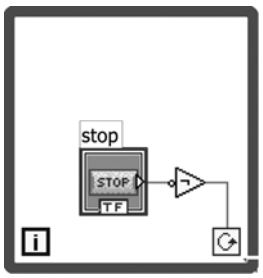


Рис. 4.3. Зміна умови завершення циклу

4.3.3. Термінали всередині циклів

Дані надходять в цикл і виходять з нього через маленьке вікно на границі циклу, яке має назву – **точка входу/виходу в структуру (тунель, tunnel).** Оскільки LabVIEW працює відповідно до принципу потоку даних, то

перш ніж цикл почне виконуватися, вхідні точки повинні передати в нього свої дані. Вихідні точки циклу виводять дані лише після завершення всіх ітерацій.

Також відповідно до принципу потоку даних, для того, щоб термінал оновлював свої значення на кожній ітерації циклу, він повинен знаходитись всередину циклу. Наприклад, цикл за умовою, рис. 4.2, перевіряє стан логічного елементу управління на кожній ітерації. Якщо з елементу зчитується значення ІСТИНА, то цикл завершує роботу.

Якщо термінал логічного елементу управління помістити за межами циклу за умовою, як це показано на рис. 4.4, то буде цикл або з нескінченним числом ітерацій, або з однократним виконанням – залежно від початкового значення логічного елементу. **Відповідно до принципу потоку даних LabVIEW прочитує логічне значення перш, ніж воно надходить в цикл, а не в процесі виконання циклу або після його завершення.**

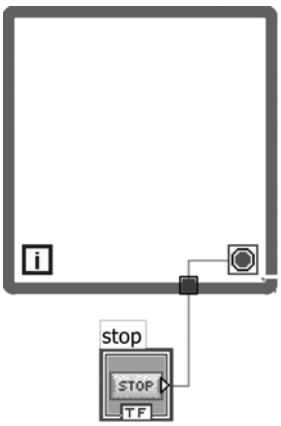


Рис. 4.4. Помилкове розміщення кнопки "Стоп"

На індикатор (Slide) в циклі на рис. 4.5 буде передаватися нове, сгенероване генератором випадкового числа, значення під час кожної ітерації виконання циклу. Але після закінчення циклу користувач зможе побачити лише значення на останній ітерації, через те, що індикатор є

скаляром, а не структурою даних (масивом).

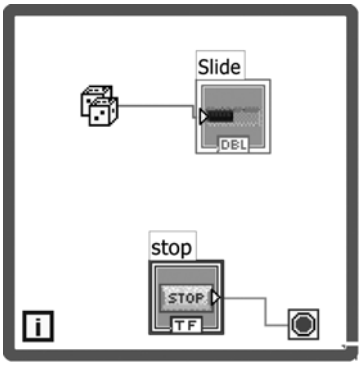


Рис. 4.5. Правильне розміщення індикатора

Значення на індикатор на рис. 4.6 буде передано лише один раз після завершення циклу, тобто після натиснення кнопки STOP. Він міститиме випадкове число, що з'явилося при виконанні останньої ітерації циклу.

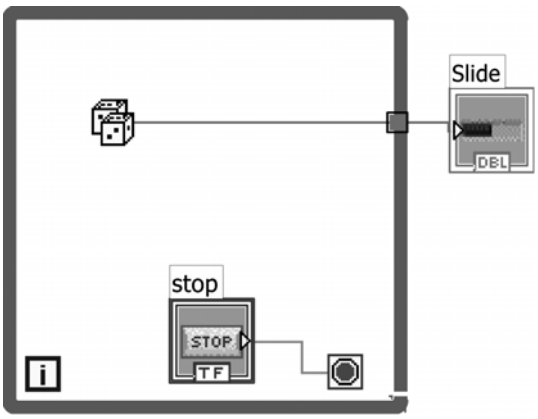


Рис. 4.6 Помилкове розміщення індикатора

Для видалення циклу, не видаляючи його вмісту, потрібно викликати контекстне меню циклу, натиснувши правою кнопкою миші по його границі, і вибрати опцію Видалити цикл за умовою (**Remove While Loop**) або Видалити цикл з фіксованим числом ітерацій (**Remove For Loop**). Крім того, цикли можна замінювати один на одний (**Replace While/For Loop**).

4.3.4. Зсувні регістри

Зсувні регістри (**shift registers**), застосовуються у циклах та є особливим типом змінної, яка використовується для передачі значень змінної з однієї ітерації циклу в наступну. Вони унікальні і необхідні в LabVIEW – графічному середовищі програмування. Зсувний регістр створюється натисканням правої кнопки миші на лівій або правій межі циклу і вибором опції Додати зсувний регістр (**Add Shift Register**) в контекстному меню.

Зсувний регістр складається з пари терміналів, розташованих один напроти одного на вертикальних сторонах границь циклів. У правому терміналі зберігаються дані, отримані при завершенні ітерації циклу, у лівому – результати виконання попередньої ітерації.

Зсувний регістр може містити дані будь-якого типу – числові, логічні, строкові, масиви і тому подібне. Зсувний регістр автоматично підстроюється до типу даних першого об'єкту, до якого його приєднують.

Можна конфігурувати зсувний регістр для запам'ятовування значень, отриманих під час декількох попередніх ітерацій. Для цього потрібно створити додаткові термінали, натиснувши праву кнопку миші на лівому терміналі регістра і вибравши опцію Додати елемент (**Add Element**) в контекстному меню.

4.3.5. Ініціалізація зсувних регістрів

Аби уникнути непередбаченої і, можливо, неправильної роботи циклів, завжди слід ініціалізувати зсувні регістри. Аби ініціалізувати, тобто записати в регістр певні значення, необхідно підключити початкове значення змінної (ініціалізуючу константу) до лівого терміналу зсувного регістра ззовні циклу. Ця процедура є необхідною тому, що в LabVIEW

після кожного запуску програми значення у терміналах зсувного регістра не обнуляються.

4.3.6. Вузол зворотного зв'язку

У загальному випадку не можна зациклювати провідник. Таке з'єднання викликає протиріччя з правилами управління потоком даних: фрагмент коду не почне виконання, поки не отримає всі вхідні дані, а з іншого боку вихідні дані з'являються лише після закінчення роботи коду. Проте, в циклі можна додавати **вузол зворотного зв'язку** між виходом і входом, як показано на рис. 4.7, і код працюватиме.

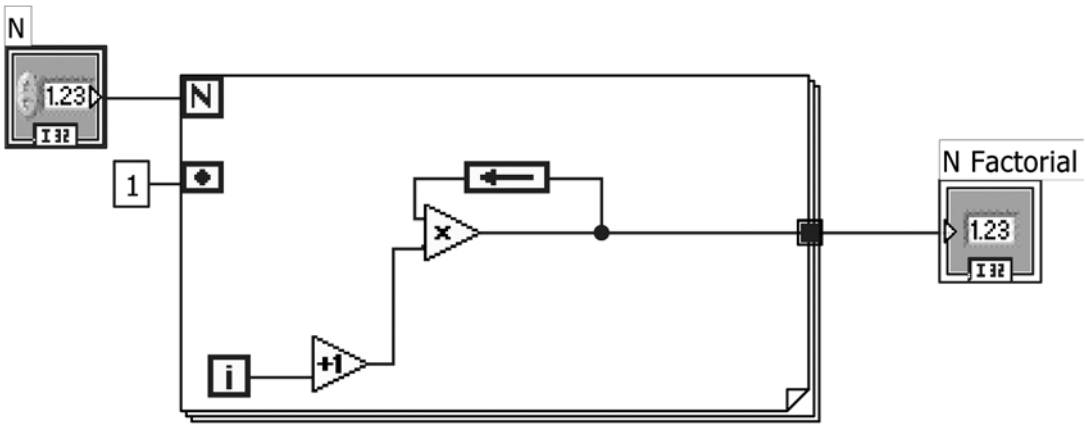


Рис. 4.7. ВП з вузлом зворотного зв'язку. Обчислення факторіалу часла N

Аби зрозуміти, як працює вузол зворотного зв'язку, слід знати, що це інша реалізація алгоритму роботи зсувного регістра. Фрагменти програмного коду на рис. 4.7 і рис. 4.8 реалізують один і той самий алгоритм.

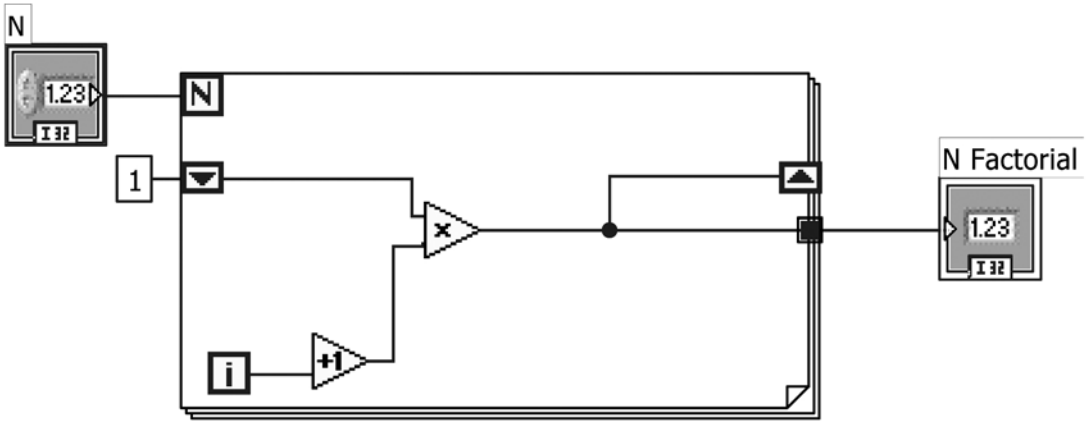


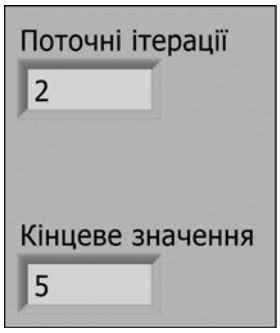
Рис. 4.8. Обчислення факторіалу числа N в циклі із зсувним регістром

Стрілка усередині терміналу функції зворотного зв’язку показує напрям потоку даних. Для ініціалізації вузла зворотного зв'язку є спеціальний термінал (**Initializer Terminal**), який можна показати або заховати за допомогою відповідного пункту контекстного меню. LabVIEW дозволяє швидко замінити зсувний регістр на вузол зворотного зв'язку і навпаки за допомогою пункту контекстного меню **Replace with...** (Замінити на) [1, 215-234].

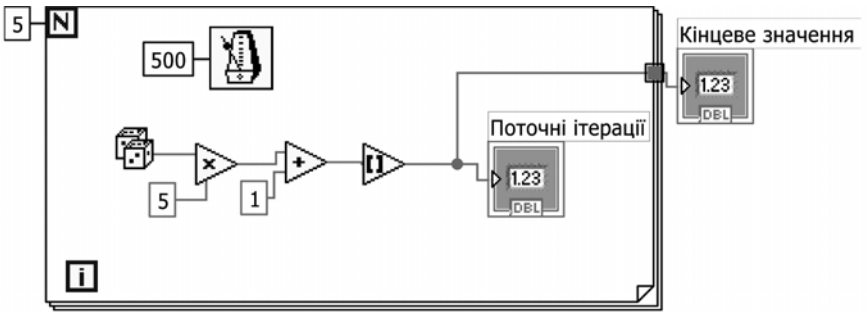
4.4. Завдання на самостійну роботу

Приклад 1. Розробити ВП, використовуючи цикл з фіксованим числом ітерацій (із затримкою часу 500 мс на кожну ітерацію), в якому випадковим чином вибираються цілі числа в діапазоні від 1 до 6. При цьому на одному числовому індикаторі відображується поточне значення випадкових чисел, а на іншому – кінцеве, по закінченні циклу.

Приклади реалізації передньої панелі і блок-діаграми ВП зображено на рис. 4.9.



а)

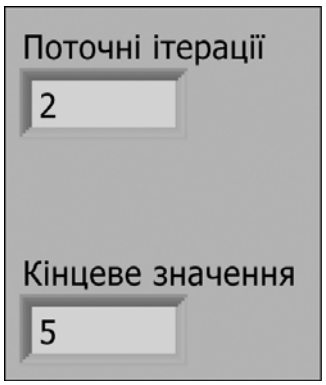


б)

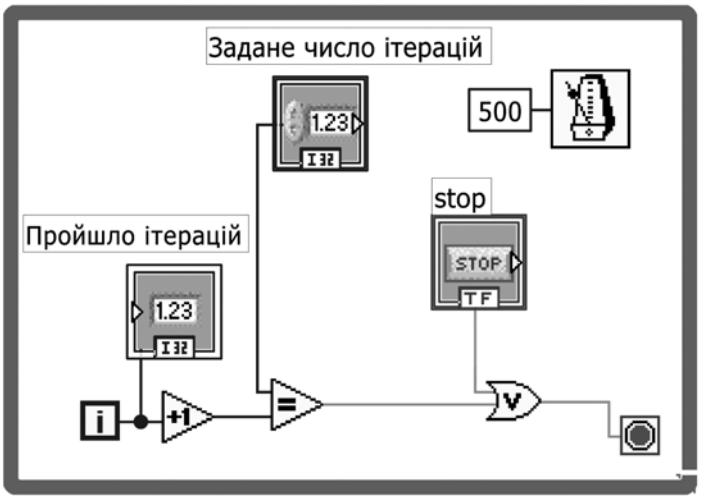
Рис. 4.9. Демонстрація роботи циклу For

Приклад 2. Використовуючи лише цикл за умовою та логічні елементи, розробити ВП, що зупиняється або при досягненні заданого користувачем числа ітерацій, або при натисканні кнопки Стоп.

Приклади реалізації передньої панелі і блок-діаграми ВП зображено на рис. 4.9.



а)



б)

Рис. 4.10. ВП, що зупиняється або при досягненні заданого користувачем числа ітерацій, або при натисканні кнопки Стоп

4.5. Контрольні питання

1. Охарактеризуйте принципи роботи управляючих структур.
2. Охарактеризуйте принцип роботи циклу While як в загальному сенсі, так і щодо реалізації у системі LabVIEW.
3. Охарактеризуйте принцип роботи циклу For як в загальному сенсі, так і щодо реалізації у системі LabVIEW.
4. Що означає термін «точка входу/виходу у цикл» та які характеристики вона має?
5. Дайте визначення поняттю «Зсувний регістр». Які властивості має даний елемент?

5. УПРАВЛІННЯ ВИКОНАННЯМ ПРОГРАМ ЗА ДОПОМОГОЮ СТРУКТУР. СТРУКТУРИ ВАРІАНТУ

Традиційно алгоритми описуються 3 типами структур даних: структурою послідовності, повторення та розгалуження. В LabVIEW структури розгалуження реалізовано за **допомогою структур варіанту (Case Structure) та функцією вибору (Select).**

5.1. Мета та завдання

Метою заняття є вивчення реалізації у системі LabVIEW алгоритмів з розгалудженням. Для цього було поставлено такі задачі:

- навчитися застосовувати структури варіанту;
- вивчити особливості використання структур;
- ознайомитись із принципами застосування діалогових вікон, як важливого елементу інтерфейсу.

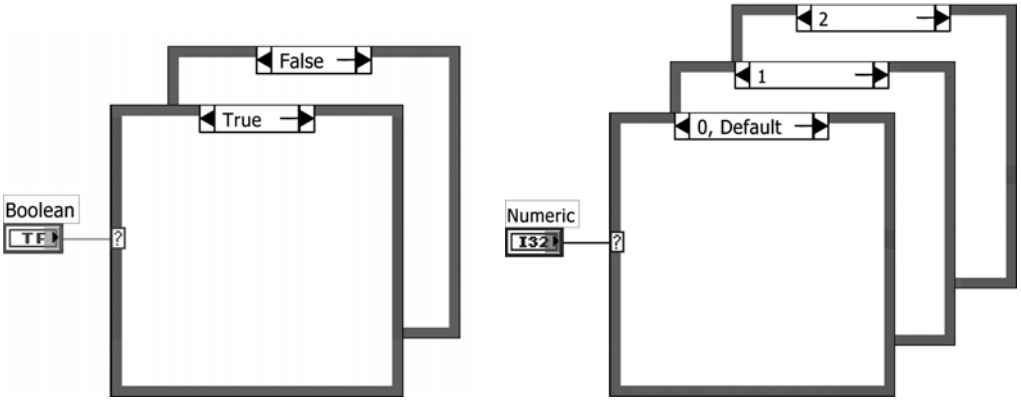
5.2. Структура варіанту в LabVIEW

Цю структуру можна знайти в підпалітрі Структури (Structures) палітри Функції (Functions). Структура варіанту, що показана на рис. 5.1, має дві або більше піддіаграм або варіантів. Лише одна з них виконується залежно від логічного, числового або строкового значення, яке подається на **термінал вибору (select)** структури варіанту.

Якщо до терміналу вибору структури варіанту підключено логічне значення, то структура матиме два варіанти: ХИБНІСТЬ і ІСТИНА.

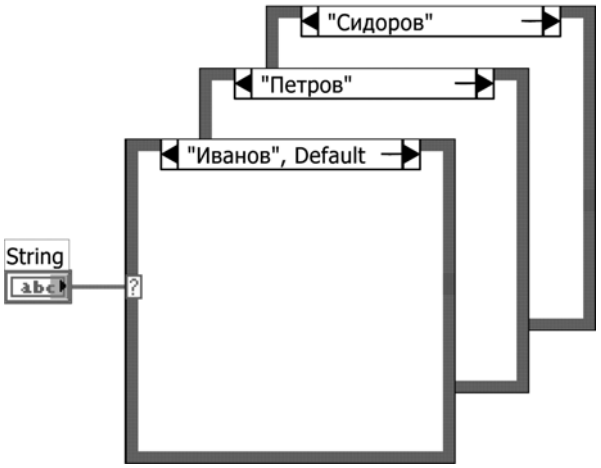
Якщо до терміналу селектора підключені числові або строкові дані, то структура може мати майже необмежену кількість варіантів, починаючи з нульового. За замовченням, структура включає лише два варіанти. Для

збільшення їх кількості потрібно у контекстному меню структури вибору задати необхідну кількість «умов». Крім того, завжди можна вибрати варіант «За умовчанням (Default)», який виконуватиметься, якщо величина, що подається на термінал вибору структури, не відповідає жодному іншому варіанту.



а)

б)



в)

Рис. 5.1 (а, б, в). Структура варіанту (а – структура двоваріантного вибору, б - структура багатоваріантного вибору за числовими мітками, в - структура багатоваріантного вибору за символьними мітками)

При розміщенні структури варіанту на передній панелі вперше вона буде представлена в логічній формі. Для того, щоб використовувати в

структурі числові значення, необхідно подати дані числового типу на термінал вибору.

Для відображення на екрані відповідної піддіаграми структури вибору необхідно нажати праву кнопку миші на границі структури і вибрати опцію **Показати варіант (Show Case).**

Якщо подати на термінал вибору число з плаваючою крапкою, LabVIEW округлить це число до найближчого цілого. LabVIEW автоматично переводить від'ємні числа до нуля і зменшує будь-яке значення, яке перевищує найбільший номер варіанту, для прирівнювання його до найбільшого номера.

Для строкових типів даних, що подаються на термінал вибору, потрібно точно визначити величини, а саме взяти в лапки ці дані. Єдиним виключенням є слово Default, яке в лапки ніколи не береться.

5.2.1. Підключення терміналів вводу/виводу

Дані у всіх вхідних терміналах (точках введення і терміналі вибору) структури варіанту доступні для всіх варіантів. При роботі з варіантами не обов'язково використовувати вхідні дані або виводити дані із структури, але якщо в одному варіанті дані виводяться, то для всіх варіантів повинно бути реалізовано вихід даних.

5.2.2. Додавання варіантів

Якщо натиснути праву кнопку миші на границі структури варіанту, то в меню, що з'явиться, будуть опції Створити варіант після (**Add Case After**) і Створити варіант перед (**Add Case Before**) поточним варіантом. Можна також скопіювати поточний варіант, вибравши опцію (копіювати варіант (**Duplicate Case**). Видалити поточний варіант (і все, що в ньому знаходиться)

можна за допомогою опції Видалити варіант (**Delete This Case**).

5.2.3. Функція вибору

У деяких випадках для реалізації логіки if-then-else зручніше використовувати функцію **Вибір (Select),** яка працює так само, як і структура варіанту.

Функція Вибір, що знаходиться в підпалітрі **Порівняння** (**Comparison**) палітри Функції, повертає значення з верхнього терміналу, якщо вхідне значення на середньому терміналі є ІСТИНА, і значення з нижнього терміналу, якщо на середньому терміналі – ХИБНІСТЬ.

Прилад, зображений на рис. 5.2, обчислює квадратний корінь з числа, що задається оператором. Якщо це число негативне, то на вихідному індикаторі відображується число -99999, як умовний символ помилки.

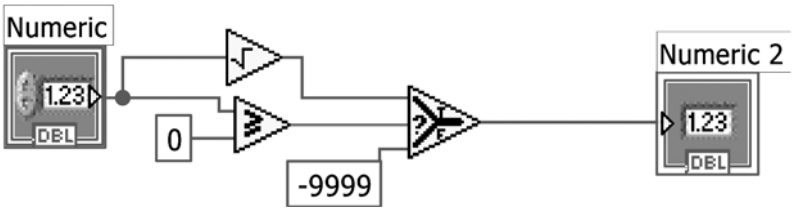


Рис. 5.2. Обчислення квадратного кореня

5.2.4. Діалогові вікна

В LabVIEW реалізовано можливості спливаючих діалогових вікон.

Найчастіше використовуються Функції **Однокнопковий діалог (One Button Dialogue) та Двокнопковий діалог (Two Button Dialogue)**, зображені на рис. 5.3.

а) б)

Рис. 5.3 (а, б). Функції «Однокнопковий діалог» (а) та «Двокнопковий діалог» (б)

Вони викликають діалогове вікно, що містить введену вами інформацію. Ці функції знаходяться в підпалітрі Час і діалоги (**Time & Dialogue**) палітри Функції. Діалогове вікно функції Однокнопковий діалог залишатиметься відкритим до тих пір, поки користувач не натисне кнопку ОК, а вікно функції Двокнопковий діалог буде відкритим, поки не буде натиснута кнопка ОК або Cancel. Дозволяється перейменувати ці кнопки, подавши на відповідні входи функцій строкові дані «ім'я кнопки» (**button name**). Ці діалогові вікна є модальними: неможливо активізувати інше вікно LabVIEW, поки вони відкриті [1, 234-245].

ВП, зображений на рис. 5.4, як і раніше описаний прилад (рис. 5.2), обчислює квадратний корінь з числа, що задається користувачем. Якщо це число негативне, то на вихідному індикаторі відображується число -99999, як символ помилки, проте крім усього іншого з'являється діалогове вікно з застереженням «Помилка! Негативне число!» і з питанням «Продовжити?», виконання програми зупиняється, поки користувач не обере відповідь.

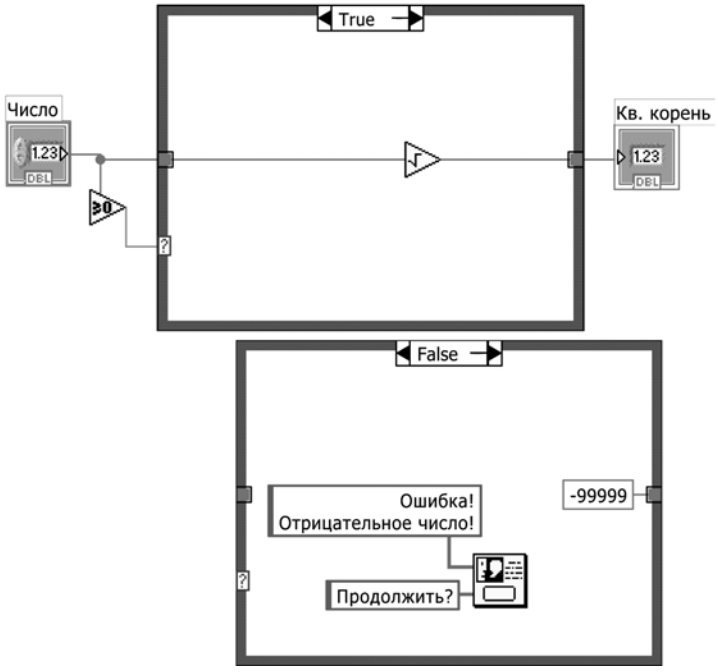


Рис. 5.4. Структура Варіанту для обчислення квадратного кореня

5.3. Завдання на самостійну роботу

Розробити ВП, який у відповідь на введену користувачем назву місяця повинен видати кількість днів цього місяця, причому, якщо це лютий, то має з'явитись додаткове уточнююче питання, чи високосний це рік.

Приклад реалізації передньої панелі і блок-діаграми наведено на рис.5.5 – 5.8. У першому варіанті оператор вводить місяць, як строкову величину, словом або цифрами порядкового номера місяця (рис. 5.5, 5.6). У другому використовується тип даних Кільцевий список (**Text Ring або Menu Ring**), (рис. 5.7, 5.8).

Рис. 5.5. Передня панель ВП, що нагадує кількість днів у введеному користувачем місяці року

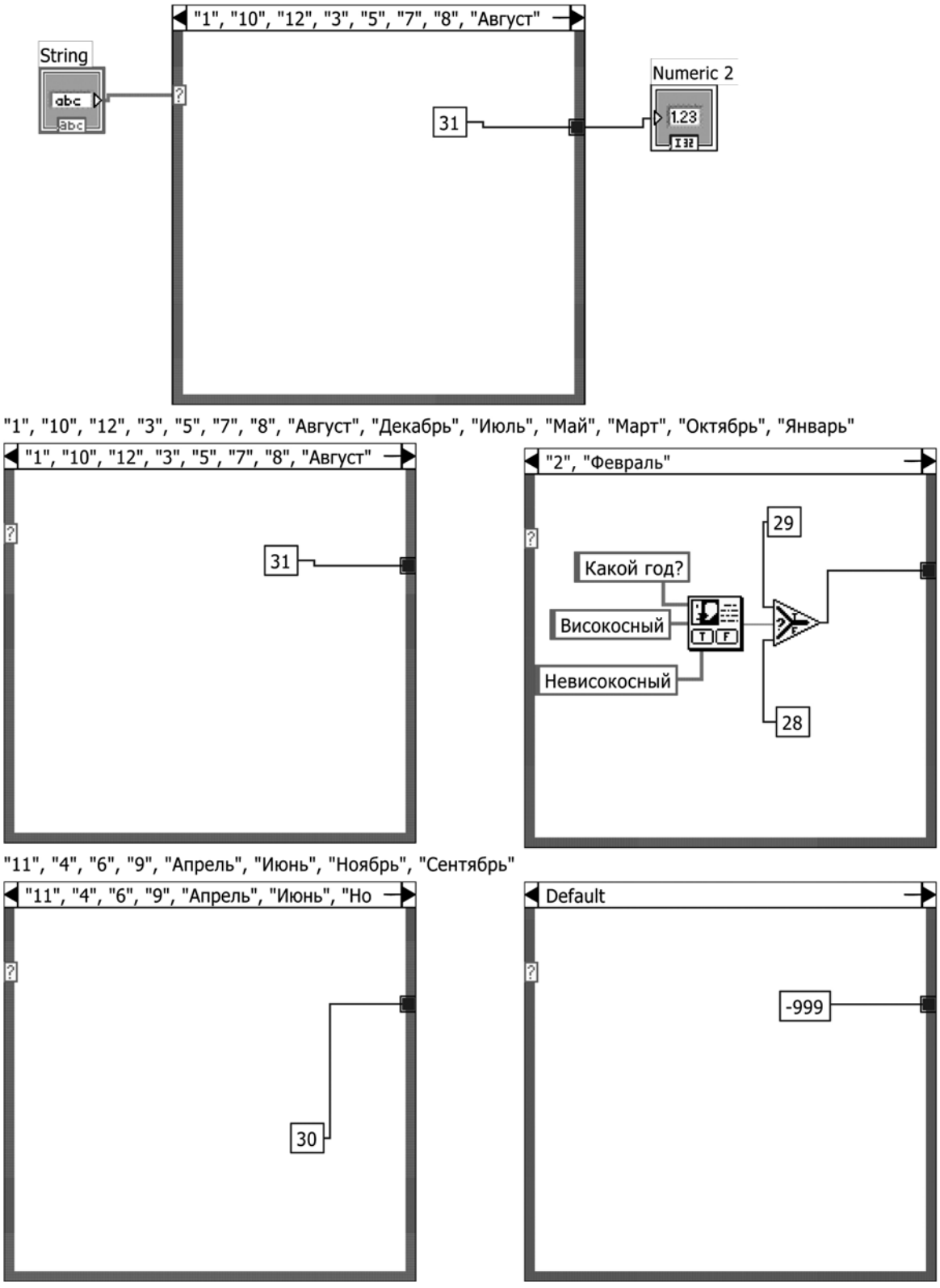


Рис. 5.6. Блок-діаграма ВП до передньої панелі, зображеної на рис. 5.5

Рис. 5.7. Другий варіант передньої панелі ВП, що нагадує кількість днів у введеному користувачем місяці року

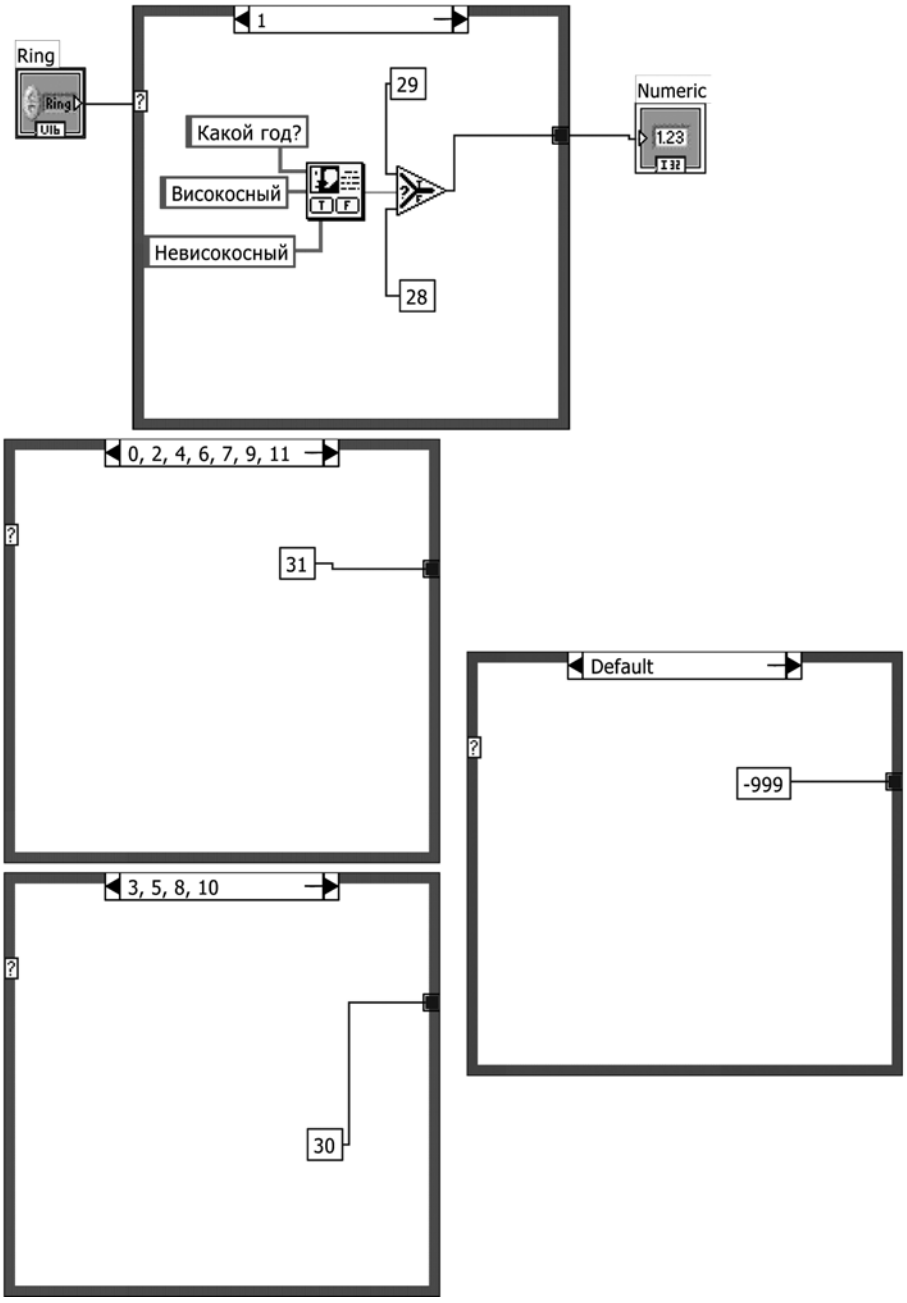


Рис. 5.8. Блок-діаграма ВП, що стосується передньої панелі на рис. 5.7

5.4. Контрольні питання

1. Дайте визначення структури Варіанту (Case Structure) у системі LabVIEW. Охарактеризуйте її основні можливості та область застосування.
2. Наведіть псевдокод роботи структури Варіанту (Case Structure).
3. Чи може вихідний тунель залишитись непідключеним в одному або кількох варіантах структури Варіанту?
4. Як працювати з діалоговими вікнами?
5. Як працювати з типом даних Text Ring або Menu Ring?

6. МАСИВИ

Масив (Array) – пронумерований, безперервний, необмежений набір однотипних даних. Кожен елемент масиву має набір індексів, відповідний розмірності масиву: одновимірний – 1 індекс, двовимірний – 2 індекси і так далі. Нумерація елементів масиву завжди починається з 0. По кожній розмірності масив може мати кількість елементів до $(2^{31}–1)$. Дані, що складають масив, можуть бути будь-якого типу: числовими, логічними, строковими та ін. Не можна лише створити масив, що складається з масивів, таблиць та графіків. Масив може складатися лише з елементів управління або лише з елементів відображення даних. Використовувати масиви зручно при роботі з групами даних одного типу і при накопиченні даних після обчислень, що повторюються. Масиви ідеально підходять для зберігання даних, отриманих з графіків, або накопичених під час роботи циклів, причому кожна ітерація циклу створює один елемент масиву.

6.1. Мета та завдання

Метою заняття є вивчення поняття масив та методів його реалізації і застосування у системі LabVIEW. Для цього було поставлено наступні задачі:

- навчитися створювати масиви в LabVIEW;
- вивчити вбудовані функції роботи з масивами;
- зрозуміти концепцію поліморфізму в LabVIEW;
- з'ясувати практичне застосування масивів і циклів з індексуючими тунелями.

6.2. Методи створення масивів і роботи з ними

Для створення масиву елементів управління або відображення даних необхідно вибрати шаблон масиву з палітри Масив і кластер (All Controls -> **Array & Cluster**) і помістити його на передню панель. Потім помістити в шаблон масиву необхідний елемент управління або відображення даних. Графічно масив виглядає як прямокутна область, через яку можна переглядати елементи масиву. Поряд з лівим верхнім кутом цієї області відображуються індекси. Значення цих індексів відповідають елементу масиву, показаному в лівому верхньому кутку. Одновимірний масив (вектор) – рядок або стовпець. Двовимірний масив (матриця, таблиця) – таблиця з декількох рядків і декількох стовпців. Масиви великої розмірності на площині екрану монітора відображувати неможливо, тому вони виглядають як таблиці, що є «зрізом» по певних індексах. Масив може містити дані довільного типа. Наприклад, може бути масив тумблерів (дискретні регулювальники), або масив цілих чисел, або масив строкових змінних. Для зміни розмірності масиву можна використовувати спливаюче меню властивостей індексу(-ів) масиву.

Дуже важливо відзначити, що графічне відображення масиву не дає інформації про те, скільки елементів містить масив, тобто скільки елементів визначено. Прямокутна область, що відображує масив є лише "переглядовим вікном".

6.2.1. Функції роботи з масивами. Поліморфізм

У палітрі функцій є всі необхідні засоби для роботи з масивами, наприклад визначення кількості елементів в масиві, пошук найбільшого або найменшого по величині елементу, здобуття елементу по індексу (індексам), сортування масиву, видалення елементів або зміна значень

елементів масиву. Більшість функцій для роботи з масивами є поліморфними і автоматично настроюються під конкретний варіант масиву і необхідне завдання. Поліморфізм – дуже зручна властивість багатьох функцій LabVIEW, що дозволяє проводити різні операції з даними різних типів за допомогою одних і тих же функцій. Наприклад, функція "додавання" може скласти два довільних числа, вона ж може скласти поелементно два масиви, за її допомогою можна додати до кожного елементу масиву яке-небудь число. Таким чином, поліморфні функції автоматично можуть підстроїтися до вхідних даних, проводячи різні дії і отримуючи різні результати.

6.2.2. Автоіндексація

Цикл For і цикл While можуть автоматично накопичувати дані на границях циклу і проводити їх індексацію, тобто генерувати масиви. Ця властивість називається автоіндексацією. Кожна ітерація створює наступний елемент масиву. Після завершення циклу масив виходить з нього і передається на елемент відображення або якийсь вузол діаграми. Дані в масиві недоступні до тих пір, поки цикл не завершиться.

На рис. 6.1 зображено приклад генерації одновимірного масиву. Лінія провідника після виходу з иклу є візуально більш товстою, а точка виходу з циклу – є індексуючою. У циклах For автоіндексація реалізована за замовченням.

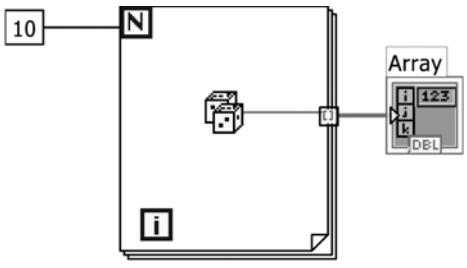


Рис. 6.1. Автоіндексація в циклі

Автоіндексація відключається клацанням правої кнопки миші по терміналу входу/виходу з циклу і вибором пункту контекстного меню **Disable Indexing**.

Для циклу While автоіндексація за умовчанням відключена. Для того, щоб включити автоіндексацію, необхідно натиснути праву кнопку миші на терміналі входу/виходу з циклу і вибрати в контекстному меню пункт **Enable Indexing** [1, 268-276].

6.3. Приклади

Приклад 6.1. Розробити ВП, в якому генерується двовимірний масив випадкових чисел (10*10), (рис. 6.2).

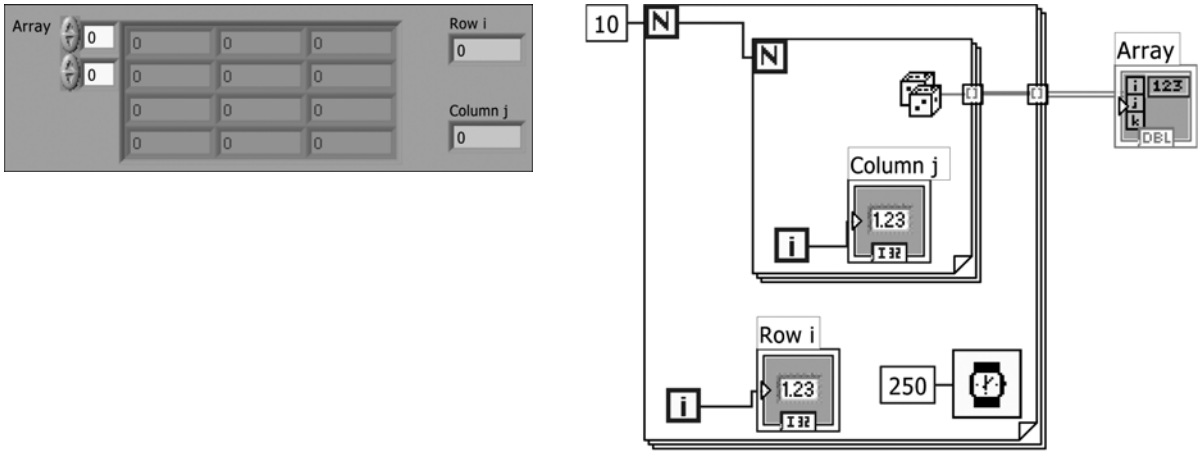


Рис. 6.2. Два цикли For, що створюють двовимірний масив

Приклад 6.2. Розробити ВП, що генерує одновимірний масив з випадкових чисел за допомогою циклу While, причому кількість елементів масиву обмежується або натисненням кнопки «Стоп», або заданим у блок-діаграмі числом 100 (рис. 6.3).

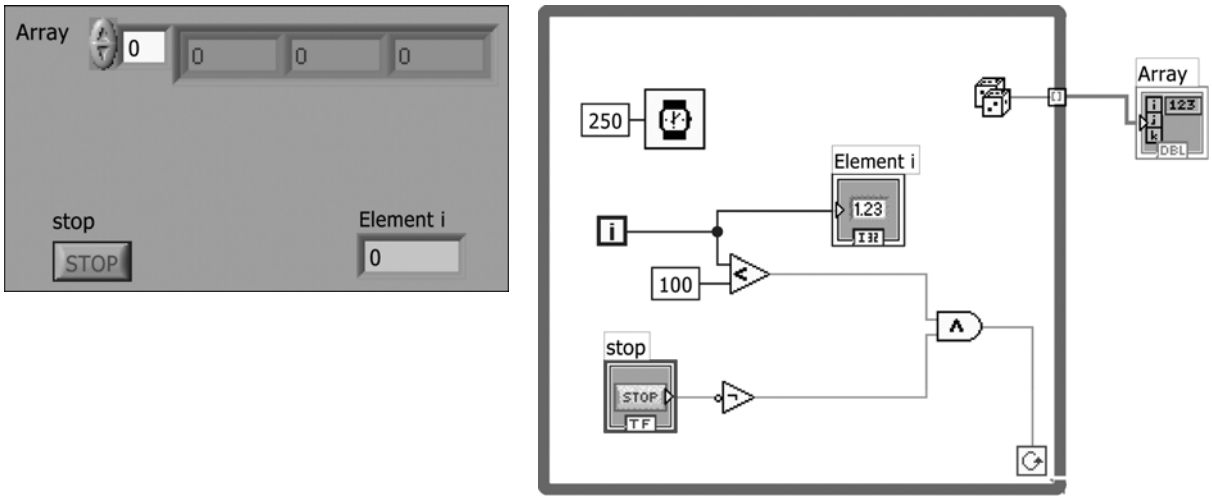


Рис. 6.3. Створення одновимірного масиву в циклі While

Приклад 6.3. Розробити ВП, який реалізує алгоритм відображення змін параметрів одного масиву в іншому (рис. 6.4). Тут слід зазначити дві важливі обставини. По-перше, ВП реалізовано з можливістю зупинки кнопкою «Стоп». По-друге, в ньому застосовано елемент «**Wait For Front Panel Activity**», який дозволяє звільнити ресурси комп'ютера для виконання інших завдань, поки на передній панелі ВП не виконується жодних дій. Перевірити, що ітерації циклу здійснюються лише при активності на передній панелі, дозволяє розміщений нами індикатор «Кол-во итераций».

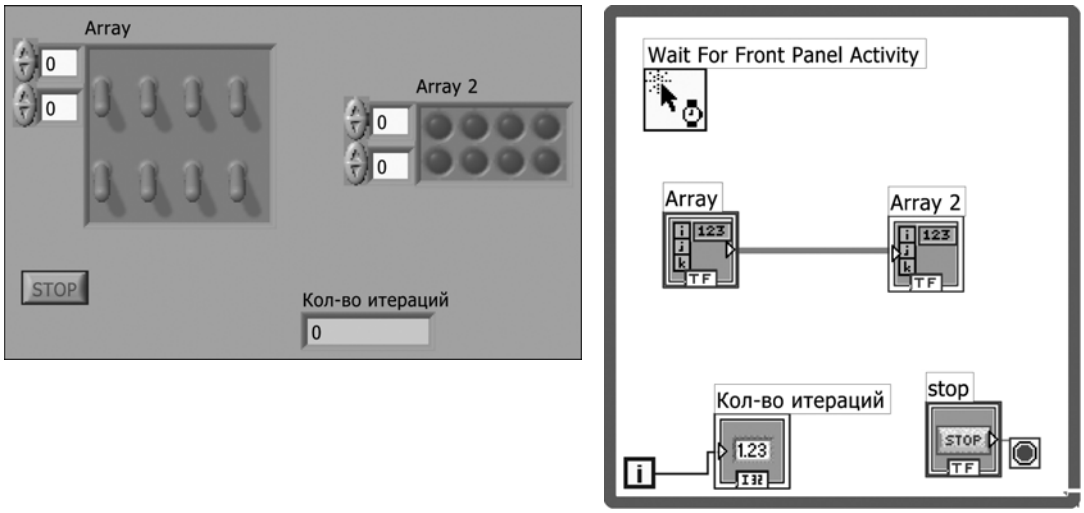


Рис. 6.4. Взаємодія двох масивів

Приклад 6.4. Розробити ВП, що обчислює факторіал числа за допомогою визначення добутку елементів масиву, сформованого індексуючим тунелем циклу (рис. 6.5).

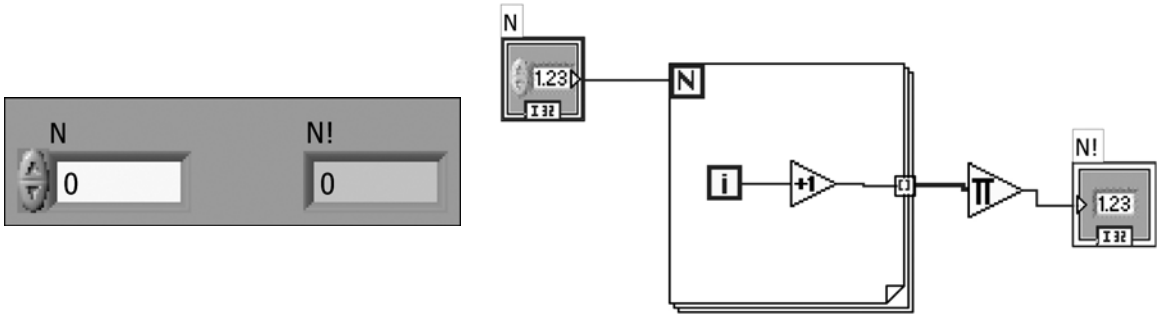
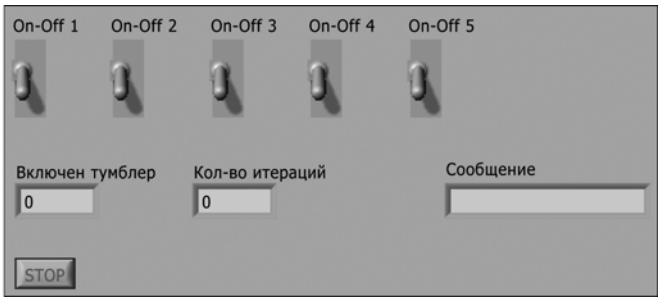


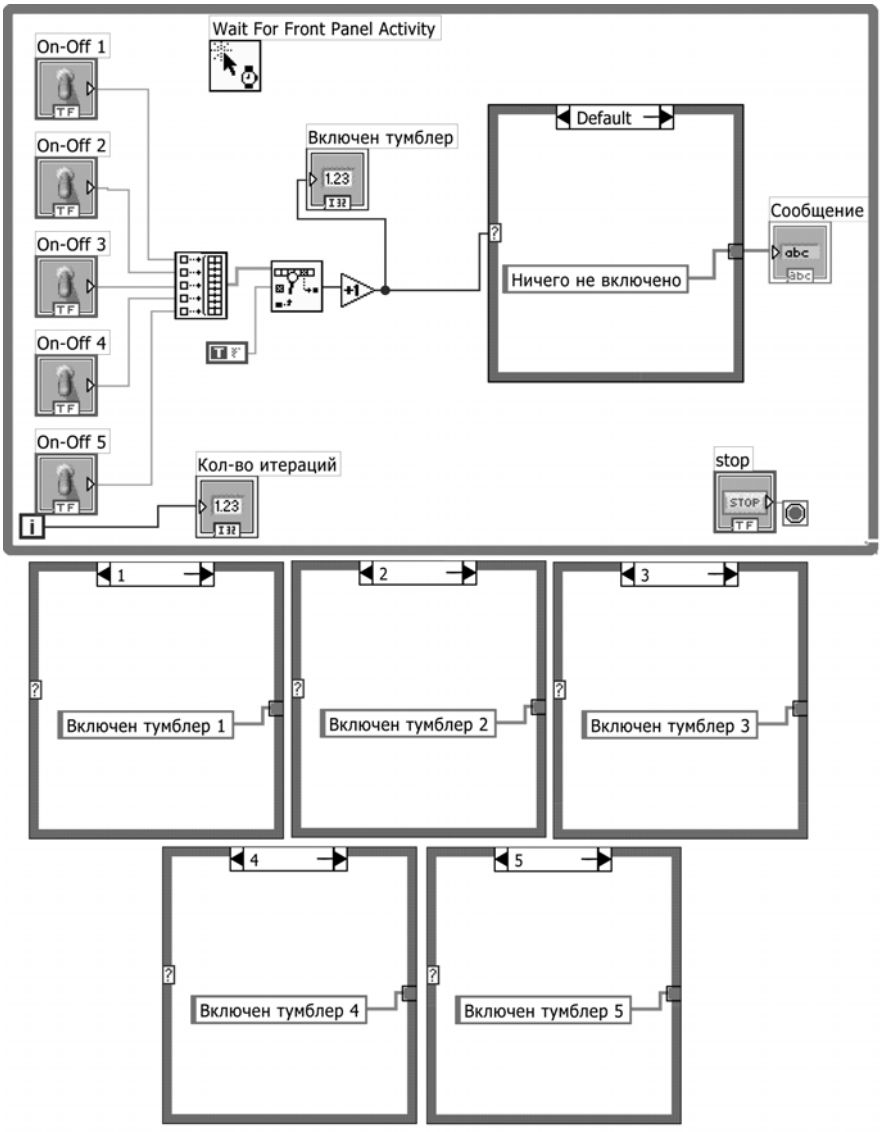
Рис. 6.5. Обчислення функції факторіалу числа N

6.4. Завдання на самостійну роботу

Приклад 1. Розробити ВП, що містить п'ять пронумерованих тумблерів, і щоб при включенні якого-небудь з них виводилося повідомлення «Включений тумблер 1» або інший номер тумблера. При цьому прилад повинен мати можливість бути вимкненим кнопкою «Стоп» і звільняти ресурси комп'ютера у відсутності активності на передній панелі. Варіант реалізації приведений на рис. 6.6.



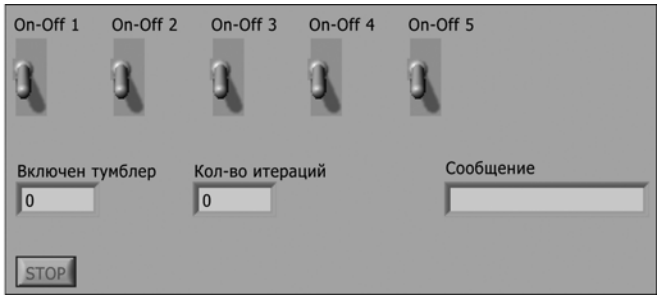
а)



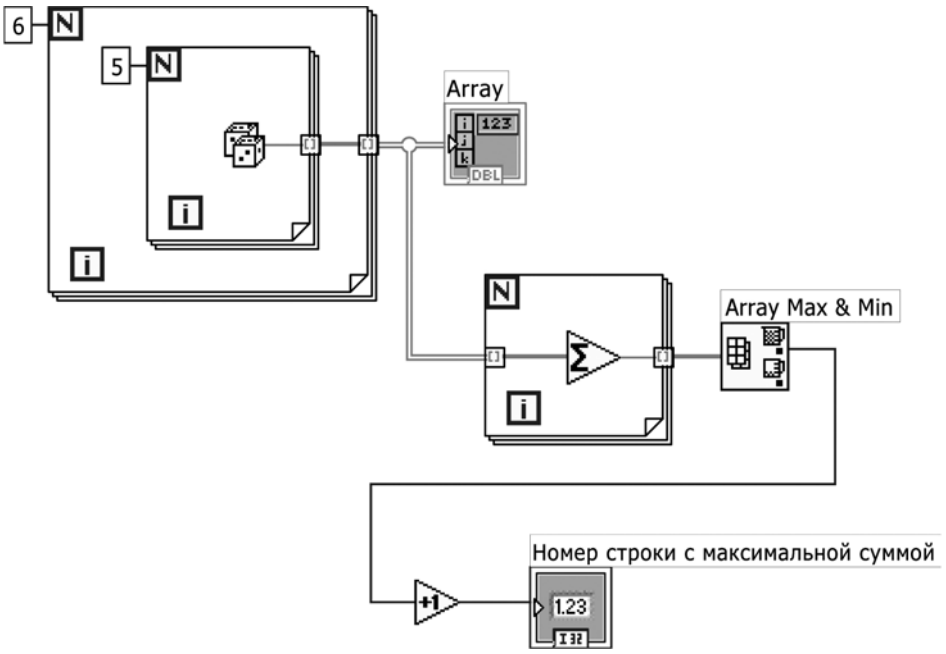
б)

Рис. 6.6. Індикація ввімкненого тумблера (а – передня панель, б – блок-діаграма)

Приклад 2. Розробити ВП, який спочатку генерує двовимірний масив випадкових чисел (5*6), а потім визначає і виводить на індикатор номер рядка цього масиву, в якому сума значень елементів максимальна. Варіант реалізації приведений на рис. 6.7.



а)



б)

Рис. 6.7. ВП, що визначає номер рядка масиву, в якому сума значень елементів є максимальною

6.5. Контрольні питання

1. Дайте визначення поняттю масив.
2. Які існують основні функції роботи з масивами?
3. Які типи даних можуть бути елементами масиву?
4. Як працює вхідний індексуючий тунель циклу?
5. Які можливості генерації масивів існують?

7. КЛАСТЕРИ

Як і масив, **кластер (cluster)** є структурою, що групує дані. Проте на відміну від масиву кластер може групувати дані різних типів (числові, логічні і так далі). Це поняття аналогічне **struct** в мові програмування С або об'єктам даних, визначеним як елементи класу, в С++ або Java.

Можливість об'єднання декількох груп даних в кластер допомагає класифікувати дані на передній панелі та зменшити кількість полів вводу/виводу даних, необхідних при використанні підпрограми. Це пов’язано із тим, що максимально можлива кількість полів вводу/виводу даних у ВП дорівнює 28. Якщо передня панель містить більше 28 елементів, які необхідно використовувати у ВП, можна деякі з них об'єднати в кластер і пов'язати кластер з полем вводу/виводу даних. Як і масив, кластер може бути елементом управління або відображення даних, проте кластер не може одночасно містити елементи і управління і відображення.

У кластері, як і в масиві, всі елементи впорядковані, але звернутися по індексу до них не можна, необхідно спочатку розділити їх.

Доступ до елементів кластера можна отримати шляхом їх повного розділення (**unbundling**) або розділення по імені елементу. Метод розділення залежить від обраної функції та має свою сферу застосування. На відміну від масивів, які можуть динамічно змінювати розмір, кластери мають фіксований розмір.

Кластери дуже зручні та грають особливу роль при відображенні графіків і роботі з помилками.

7.1. Мета та завдання

Метою заняття є вивчення типу даних кластер та можливостей його застосування при реалізації програмних систем. Для цього було поставлено наступні задачі:

- навчитися створювати кластери і працювати з ними;
- вивчити вбудовані функції роботи з кластерами;
- з'ясувати переваги і правила використання кластерів при відображенні графіків;
- ознайомитися з поняттям «Кластер помилок».

7.2. Створення кластерів

Для створення кластера елементів управління або відображення даних необхідно вибрати шаблон кластера **(Cluster)** з палітри Масив і кластер (All Controls -> **Array & Cluster**) і помістити його на передню панель. Потім помістити в шаблон кластера необхідні елементи управління або відображення даних.

Об'єктами усередині кластера можуть бути лише елементи управління або лише індикатори. Не можна помістити елементи управління і індикатори в одному кластері, оскільки сам кластер має бути або першого, або другого типу. Кластер стає елементом управління або індикації залежно від типу першого внесеного до нього об'єкту. У разі потреби можна змінити розміри кластера за допомогою інструменту переміщення.

Кожен елемент кластера має свій порядковий номер, не пов'язаний з положенням елементу в шаблоні. Першому поміщеному в кластер елементу автоматично присвоюється номер 0, другому елементу – 1 і так

далі. При видаленні елементу порядкові номери автоматично змінюються.

Порядок елементів в кластері визначає те, як елементи кластера будуть розподілені по терміналах функцій Bundle (об'єднання) і Unbundle (розділення) на блок-діаграмі.

Переглянути і змінити порядковий номер об'єкту, поміщеного в кластер, можна, натиснувши праву кнопку миші на краю кластера і обравши з контекстного меню пункт **Reorder Controls In Cluster**. Панель інструментів і кластер наберуть вигляду, показаного нижче на рис. рис. 7.1.

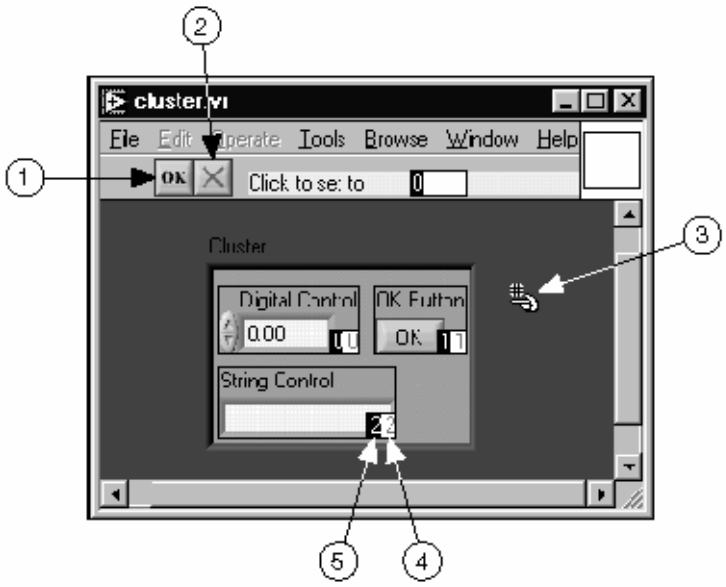


Рис. 7.1. Панель для зміни порядкового номеру об'єкту, поміщеного в кластер

1. Кнопка підтвердження (Confirm button)
2. Кнопка відміни (Cancel button)
3. Курсор визначення порядку (Cluster order cursor)
4. Поточний порядковий номер (Current order)
5. Новий порядковий номер (New order)

У білому полі (4) вказаний поточний порядковий номер елементу, в чорному (5) – новий порядковий номер. Для встановлення порядкового

номера елементу потрібно в полі введення тексту **Click to set to** ввести число і натиснути на елемент. Порядковий номер елементу зміниться. При цьому коректуються порядкові номери інших елементів. Зберегти зміни можна, натиснувши кнопку Confirm на панелі інструментів. Повернути первинні установки можна, натиснувши кнопку Cancel.

7.2.1. Функції роботи з кластерами

Для створення і управління кластерами використовуються функції, розташовані на палітрі Functions -> Cluster. Функції Bundle (Об'єднати) і Bundle by Name (Об'єднати по імені) використовуються для об’єднання елементів в кластер. Функції Unbundle (Розділити) і Unbundle by Name (Розділити по імені) використовуються для «розбирання» кластерів по елементам.

Ці функції також можна викликати, натиснувши праву кнопку миші на терміналі даних кластера і вибравши з контекстного меню підменю Cluster Tools. Функції Bundle і Unbundle автоматично містять правильну кількість полів вводу/виводу даних. Функції Bundle by Name і Unbundle by Name в полях вводу/виводу даних містять ім'я першого елементу кластера.

Для об’єднання окремих елементів в кластер використовується функція **Bundle.** Ця ж функція використовується для зміни даних в елементі вже існуючого кластера. Інструмент виділення і переміщення використовується для додавання полів введення даних (розтягненням вниз), для цього також можна натиснути праву кнопку на полі введення даних і вибрати з контекстного меню пункт Add Input.

Функція **Bundle by Name** працює так само як функція Bundle, але замість звернення до елементу кластера по його порядковому номеру звертається до нього по його власній мітці (імені). При цьому можна

дістати доступ лише до елементів, що мають власну мітку. Кількість полів введення даних не вимагає відповідності з кількістю елементів в кластері.

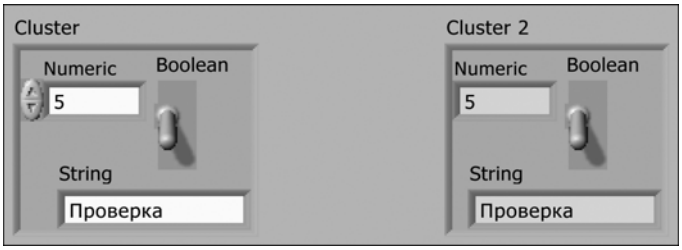
За допомогою елементу управління («палець») можна клацнути по полю введення даних терміналу і вибрати бажаний елемент з меню, що випадає. Можна також натиснути праву кнопку миші на полі введення даних і вибрати елемент в розділі контекстного меню **Select Item.**

Функція **Unbundle** використовується для розділення кластерів на окремі елементи. Вихідні компоненти розташовані зверху вниз у тому ж порядку, що і в кластері.

Функція **Unbundle by Name** використовується для виділення з кластера елементів за їх іменем. Кількість полів виведення даних не залежить від кількості елементів в кластері [1, 291-296].

7.3. Приклади

Приклад 7.1. Ілюструється два безпосередньо сполучених ідентичних кластери (один з яких складається з управляючих елементів, а другий – з індикаторів), таких, що містять різнорідні елементи (число, логічний елемент та строку) (рис. 7.2). У працюючому приладі елементи другого кластера (індикатори) повністю відтворюють стани елементів першого кластера.

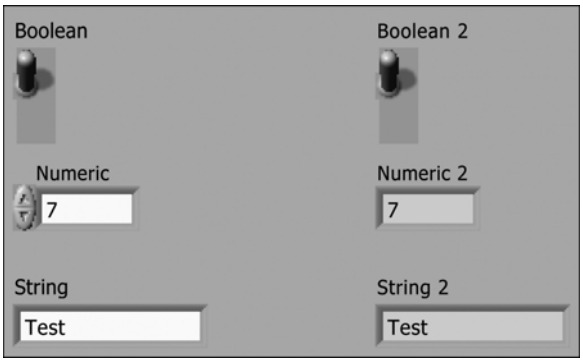


а)

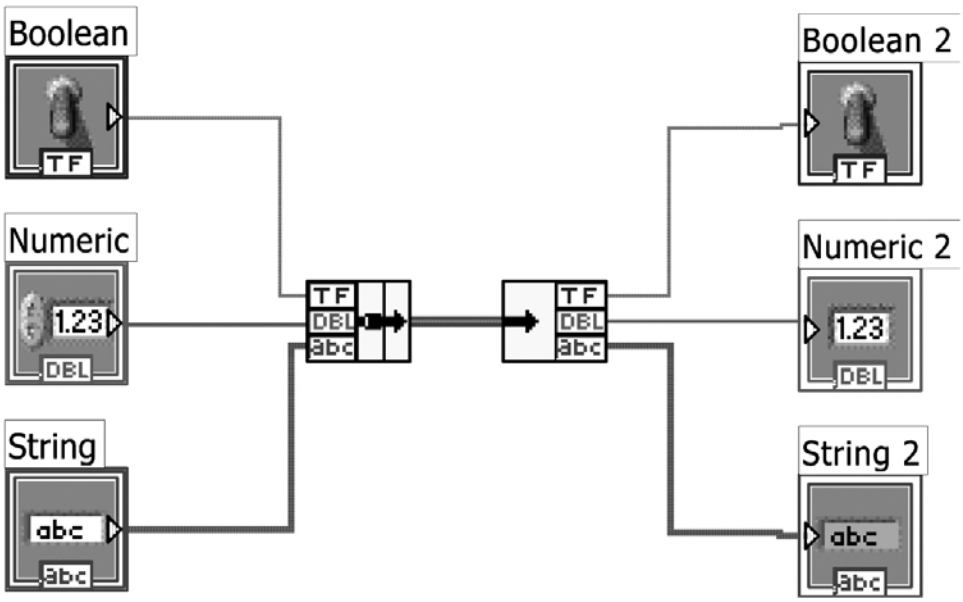
б)

Рис. 7.2 (а, б). Демонстрація ВП зв'язку двох кластерів

Приклад 7.2. Ілюструється створення кластера із управляючих елементів передньої панелі за допомогою функції Bundle і розділення на індикатори за допомогою функції Unbundle (рис. 7.3).



а)

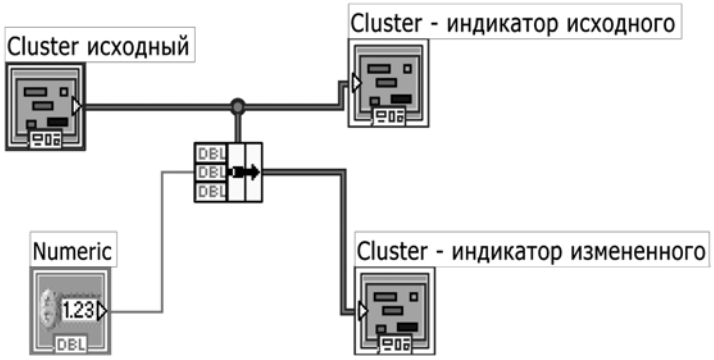


б)

Рис. 7.3. Приклад використання функцій Bundle та Unbundle (а – передня панель, б – блок-діаграма)

Приклад 7.3. Ілюструється віртуальний прилад (рис. 7.4), в якому за допомогою функції Bundle проводиться заміна одного з елементів у складі кластера.

а)

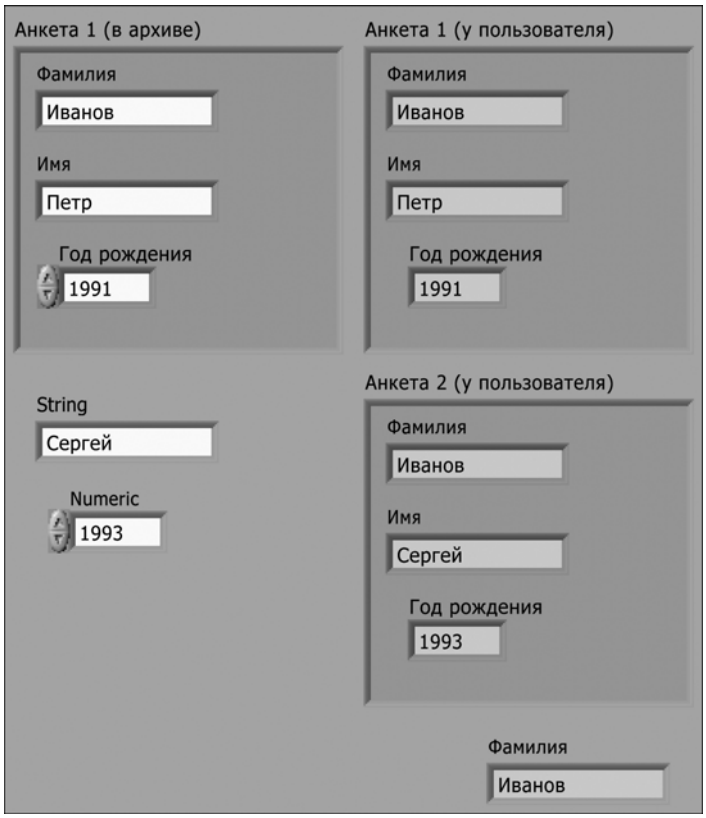


б)

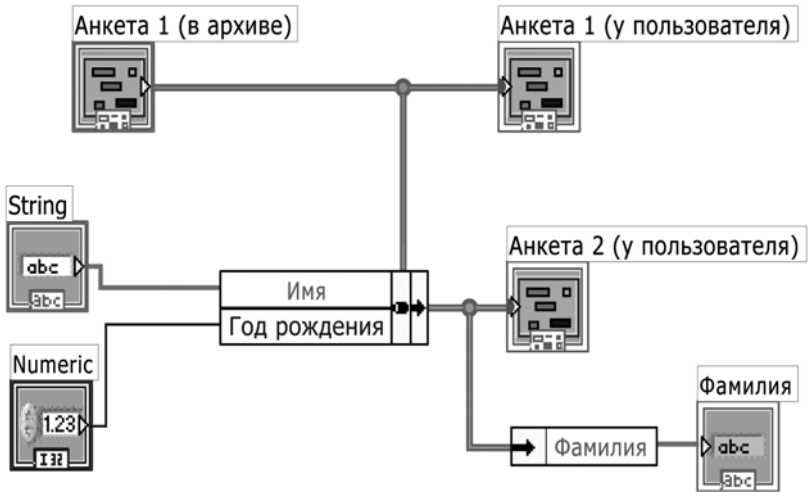
Рис. 7.4. Заміна елементу в кластері (а – передня панель, б – блок-діаграма)

Приклад 7.4. Ілюструється віртуальний прилад (рис. 7.5), який за допомогою функції Bundle by Name замінює в кластері (наприклад, архівній картці студента) ім'я і рік народження на нові значення, при цьому створюється новий кластер із наслідуванням структури початкового

(наприклад, коли в учбовий заклад поступив брат студента).



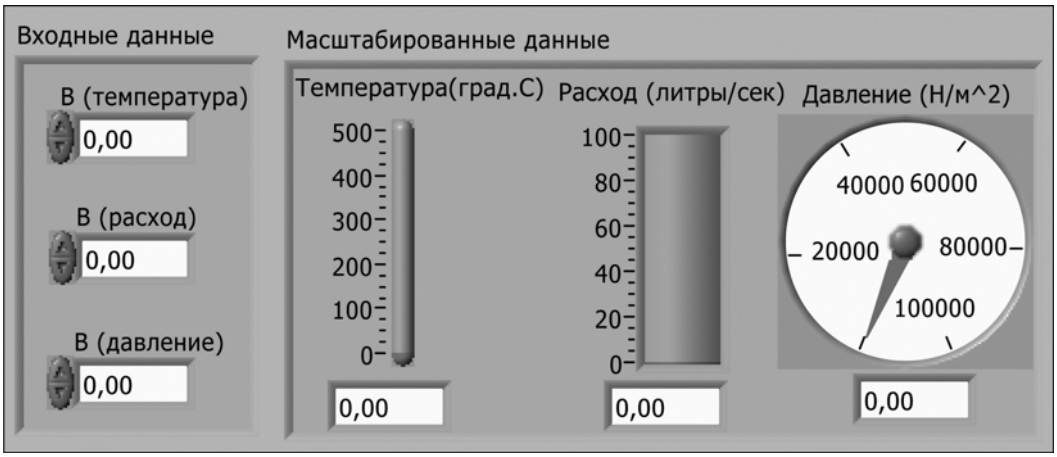
а)



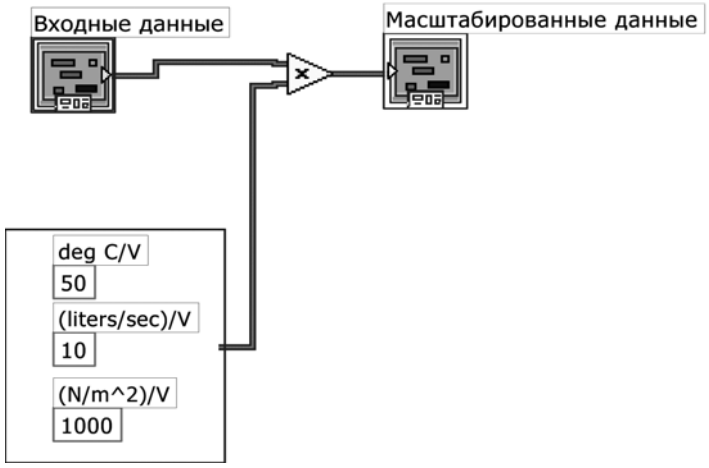
б)

Рис. 7.5 (а, б). Створення нового кластера із наслідуванням структури початкового (а – передня панель, б – блок-діаграма)

Приклад 7.5. Ілюструється віртуальний прилад (рис. 7.6), в якому моделюється така ситуація: три датчики вимірюють, відповідно, температуру, витрату рідини і тиск, проте на виході видають електричну напругу у вольтах. Калібрування датчиків, тобто відповідність напруги і фізичних величин, є відомою. У приведеному приладі об'єднані в кластер показники датчиків перемножуються на кластер калібрувальних коефіцієнтів, внаслідок чого на індикатори (також об'єднані в кластер) надходять вже значення відповідних фізичних величин. Даний приклад демонструє поліморфізм функції множення при роботі із кластерами.



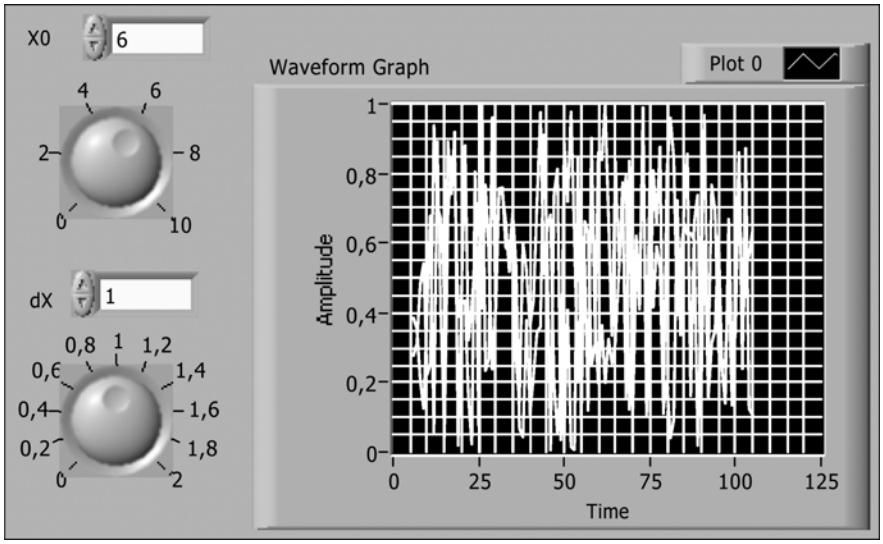
а)



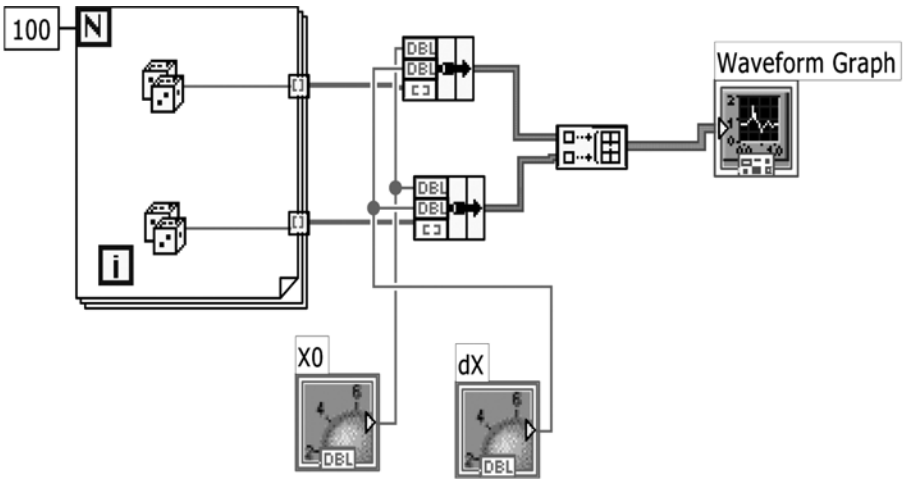
б)

Рис. 7.6 (а, б). Поліморфізм функції множення при роботі із кластерами

Приклад 7.6. Ілюструється віртуальний прилад (рис. 7.7), в якому демонструється, яким чином формується кластер типу **Waveform** для роботи з графіками і як поєднуються разом два графіки за допомогою функції Build Array.



а)

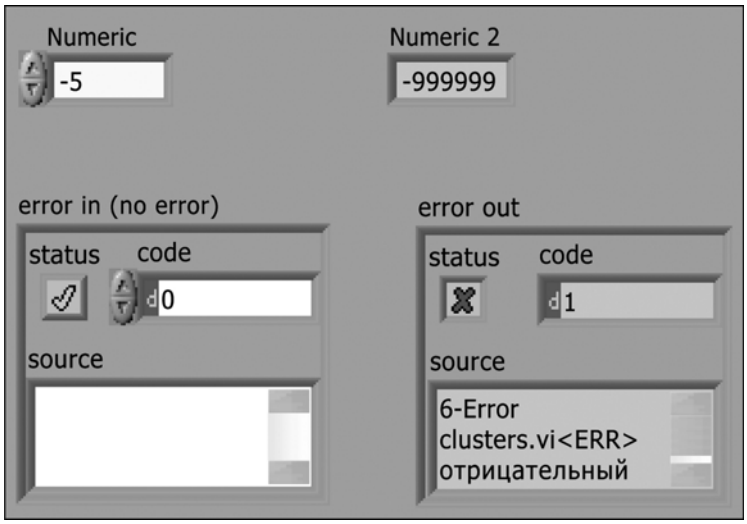


б)

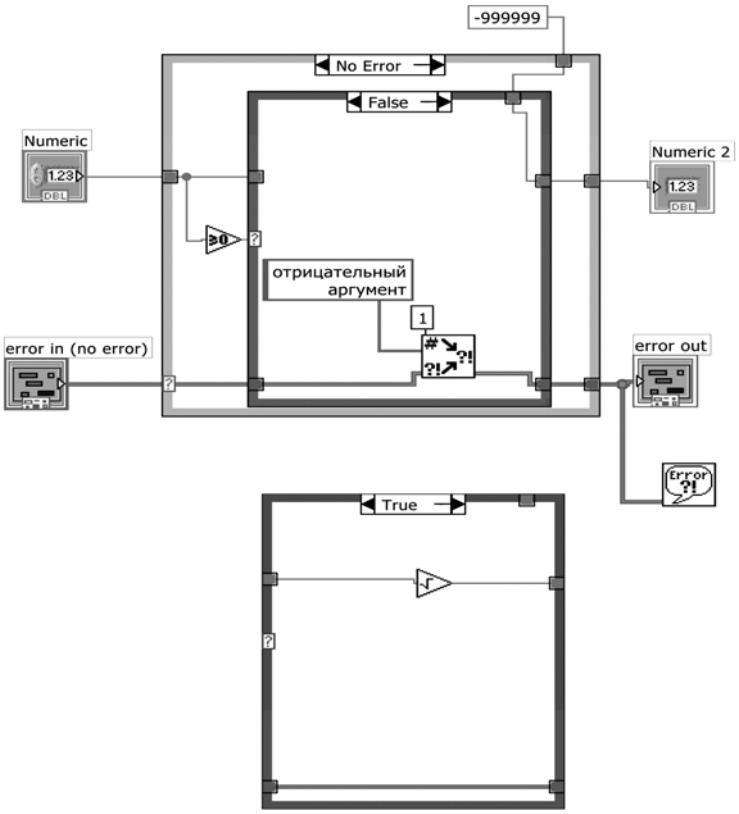
Рис. 7.7 (а, б). Формування кластера типу Waveform

Приклад 7.7. Ілюструється приклад реалізації кластеру помилок (рис. 7.8). Помилка виникає при спробі обчислення квадратного кореня з

негативного числа. У прикладі показана будова кластера помилок і спеціальний тип структури Case для обслуговування помилок.



а)



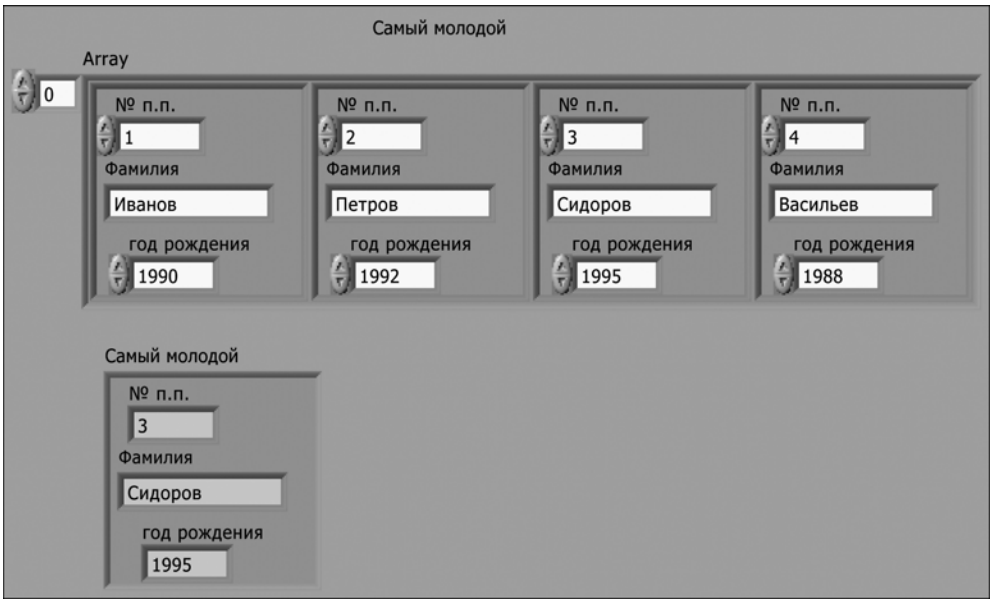
б)

Рис. 7.8 (а, б). Приклад реалізації кластеру помилок (а – передня панель, б – блок-діаграма)

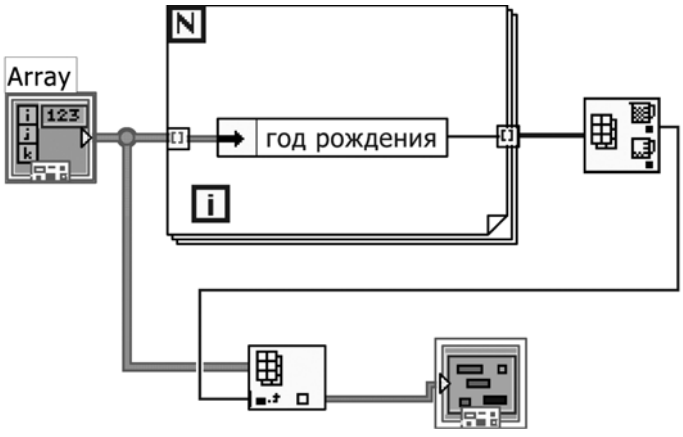
7.4. Завдання на самостійну роботу

Розробити ВП, який вибирає наймолодшого студента за інформацією, що зберігається у вигляді масиву кластерів. Кластер містить три поля: № п/п (ціле число), прізвище (тип даних – рядок), рік народження (ціле число). Знайдена інформація про студента має бути представлена в такому ж вигляді (у вигляді кластера).

Варіант реалізації приведено на рис. 7.9.



а)



б)

Рис. 7.9 (а, б). Пошук наймолодшого за віком студента в архіві (а – передня панель, б – блок-діаграма)

7.5. Контрольні питання

1. Дайте визначення поняттю «Кластер». Зробіть порівняльний аналіз даного поняття для різних мов програмування.
2. Для яких задач і коли є доцільним використання Кластерів у LabVIEW?
3. Які існують вбудовані функції роботи з кластерами?
4. Що вийде, якщо звичайну функцію додавання застосувати до двох кластерів?
5. Яку будову має кластер Waveform?
6. Що містить кластер помилок?

8. ГРАФІКИ ОСЦИЛОГРАМ (WAVEFORM CHART)

Графіки осцилограм в LabVIEW дозволяють відображувати дані в графічній формі. Графіки осцилограм (**Waveform chart**) будують осцилограми послідовно, додаючи до старих даних нові по мірі їх надходження, тобто на екрані постійно відображується вміст деякого буфера визначеного користувачем розміру, а сам буфер організований таким чином, що у міру надходження нових даних найстаріші, такі, що не поміщаються в буфер, видаляються. Відмінністю від іншого типу даних «Графік» (**Waveform graph і XY graph**) є те, що останні відображують масиви даних, що вже згенеровані, як правило, не зберігаючи попередніх.

8.1. Мета та завдання

Метою заняття є вивчення генерації та застосування графіків осцилограм. Для цього було поставлено такі задачі:

- вивчити властивості, параметри та елементи управління графіків осцилограм;
- вміти розрізняти три режими оновлення графіків осцилограм: панорамний, тимчасовий графік і тимчасовий графік з маркером;
- зрозуміти функціональну відмінність графіків (Waveform graph) і графіків осцилограм (Waveform chart);
- навчитися покращувати зовнішній вигляд графіків, змінюючи їх масштаб і використовуючи палітру управління, панелі редагування і курсори.

8.2. Основи роботи з графіками осцилограм

Крива на графіку є графічним відображенням залежності величини Y від величини X. Зазвичай величина Y представляє значення даних, тоді як величина X представляє час. Графік осцилограми (**Waveform chart**), що розташований в підпалітрі Графік **(Graph)** палітри Елементи управління (**Controls**), є особливим числовим елементом відображення, який може показати в графічному вигляді одну або декілька кривих. Найчастіше графіки осцилограм використовуються всередині циклів. У них зберігаються і відображуються на дисплеї, що постійно оновлюється, дані, які були отримані раніше, а також нові дані по мірі їх надходження. У графіку осцилограми Y представляє значення нових даних, а X – час (найчастіше кожне значення Y створюється під час ітерації циклу; таким чином, значення X є моментом часу виконання одного циклу).

8.2.1. Режими оновлення графіку осцилограми

Графік осцилограми має три режими оновлення: панорамний ("стрічковий") (**strip chart mode**), тимчасовий графік (**scope chart mode**) і тимчасовий з маркером (**sweep chart mode**). Режим оновлення можна змінити, натиснувши праву кнопку миші на графіку осцилограми і вибравши одну з опцій в меню Додатково -> Режим оновлення (Advanced -> Update Mode).

Графіки осцилограм можуть містити більш ніж один графік. Проте, оскільки не є можливим з'єднання декількох елементів на блок-діаграмі з одним терміналом графіка, то спочатку слід об'єднати дані, використовуючи функцію Об'єднати (Bundle) з палітри кластерів. Для того, щоб побудувати більше графіків, просто збільшують кількість вхідних терміналів функції Bundle шляхом зміни її розміру інструментом

переміщення («стрілка»).

8.2.2. Очищення вмісту графіку осцилограми

Коли необхідно видалити всі раніше отримані дані з графічного індикатора графіка осцилограми, вибирають опцію Операції з даними -> Очистити графік (**Data Operations -> Clear Chart**) в контекстному меню графіка, коли прилад знаходиться в режимі редагування. Якщо ВП знаходиться в режимі виконання, то опція Очистити графік знаходиться в контекстному меню, а не на вкладці Операції з даними.

8.2.3. Відображення кривих на графіку

Якщо відображується графік осцилограми з кількома кривими, можливо вибрати один з варіантів: показувати всі криві відносно загальної осі Y (поєднаний режим) або для кожного графіка використовувати власну вісь Y (режим окремих графіків). Для вибору типа відображення служать опції Набір графіків (**Stack Plots**) або Поєднані графіки (**Overlay Plots**) в контекстному меню графіка.

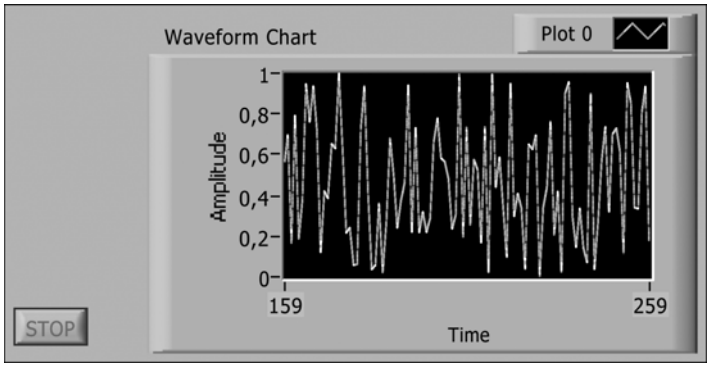
8.2.4. Довжина графіка

За умовчанням графік осцилограми зберігає до 1024 точок. Якщо потрібно зберегти більшу або меншу кількість даних, вибирають опцію Довжина історії графіка (**Chart History Length**) в контекстному меню графіка і встановлюють нове значення величиною до 100000 точок. Зміна розміру буфера не впливає на кількість даних, що вимальовуються на графічному індикаторі. Для візуалізації більшої або меншої кількості даних на індикаторі просто змінюють його розмір. Проте збільшення розміру

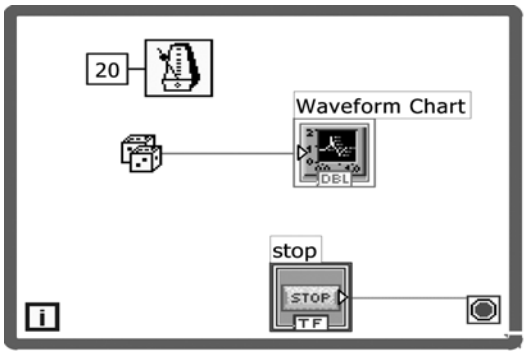
буфера збільшує кількість даних, які можна переглянути за допомогою прокрутки, тобто збільшується історія, що зберігається [1, 318-326].

8.3. Приклади

Приклад 8.1. Ілюструється найпоширеніший спосіб роботи з графіком осцилограми (рис. 8.1), а саме, дані у вигляді випадкових чисел створюються прямо в циклі, а графік осцилограм знаходиться в середині циклу. На кожній ітерації циклу відображається одна точка на графіку осцилограми. По мірі надходження даних, вони заповнюють буфер і відображуються на графіку, а коли буфер заповниться, нові дані, витісняючи найстаріші, дописуються і відображуються праворуч на графіку.



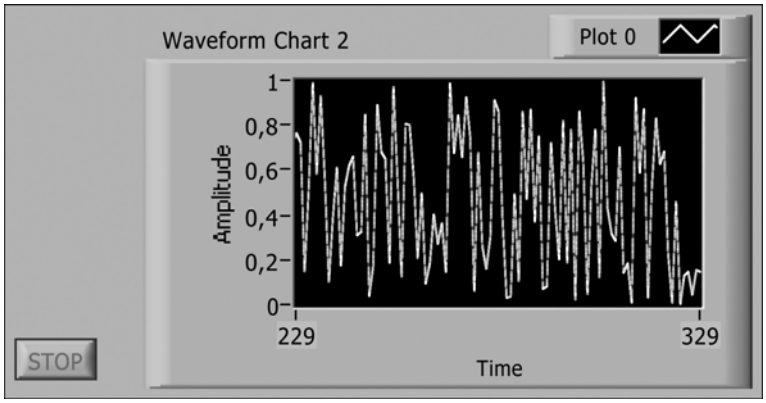
а)



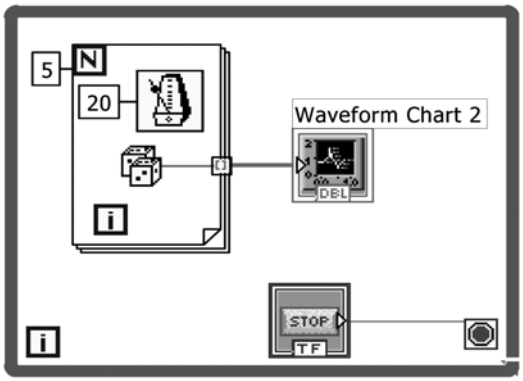
б)

Рис. 8.1 (а, б). Найпоширеніший спосіб роботи з графіком осцилограми

Приклад 8.2. На рис. 8.2 показано, що на графік осцилограми можна передавати масив даних, додаючи таким чином по декілька точок за один раз (за одну ітерацію).



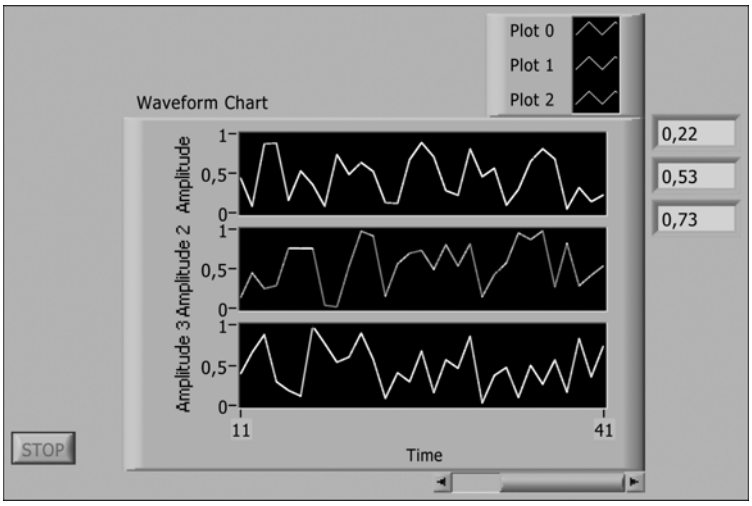
а)



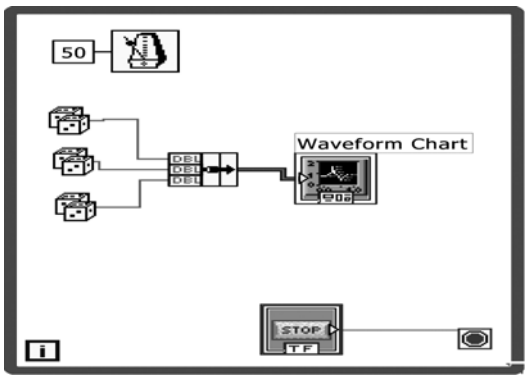
б)

Рис. 8.2 (а, б). Відображення масиву даних на графік осцилограми

Приклад 8.3. На рис. 8.3 показано, яким чином слід об'єднувати декілька потоків даних для відображення їх на одному графіку осцилограми. Для цього використовується функція Об'єднати (Bundle) з палітри кластерів.



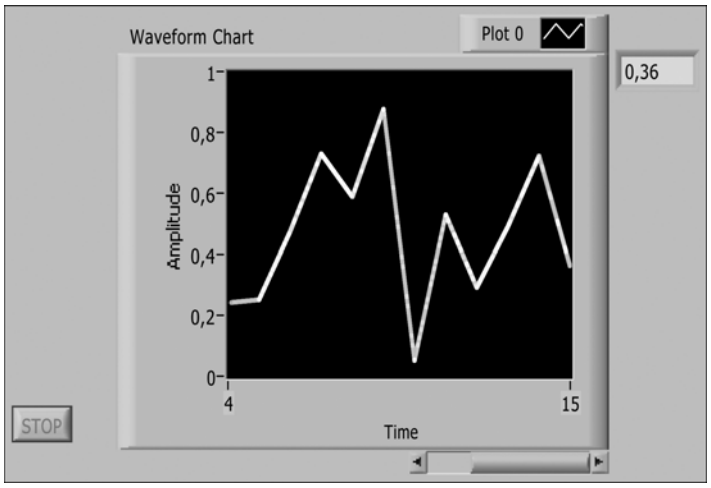
а)



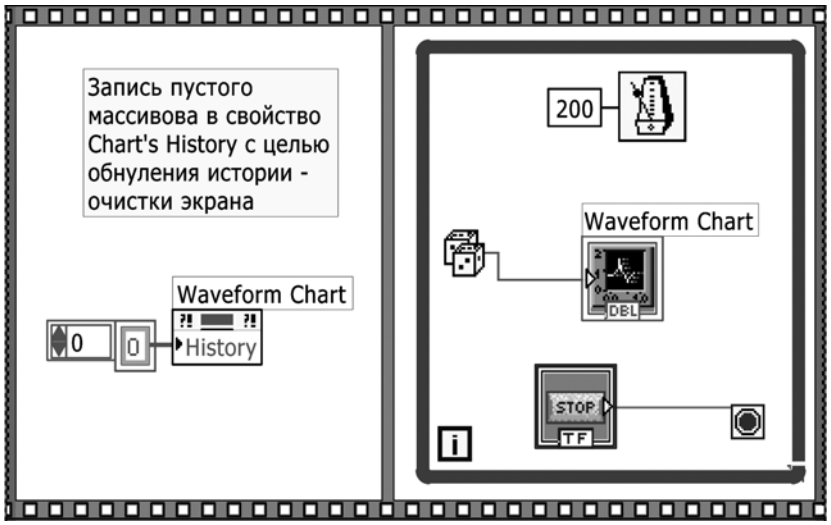
б)

Рис. 8.3. Об'єднання декілька потоків даних для відображення на графіку осцилограми

Приклад 8.4. На рис. 8.4 проіллюстровано метод програмного очищення екрану графіка осцилограми (обнулення історії, **clean history**) шляхом запису порожнього масиву у властивість Chart's History. При цьому для доступу до властивості **Chart's History** використовується **Property Node**, а для гарантування того, що очищення екрану проводиться на початку роботи приладу, – структура послідовності (**Sequence**). У першому кадрі структури послідовності записується порожній масив у властивість Chart's History, а в другому кадрі – послідовно генеруються і записуються (відображуються) випадкові числа.



а)



б)

Рис. 8.4 (а, б). Програмне очищення екрану графіка осцилограми

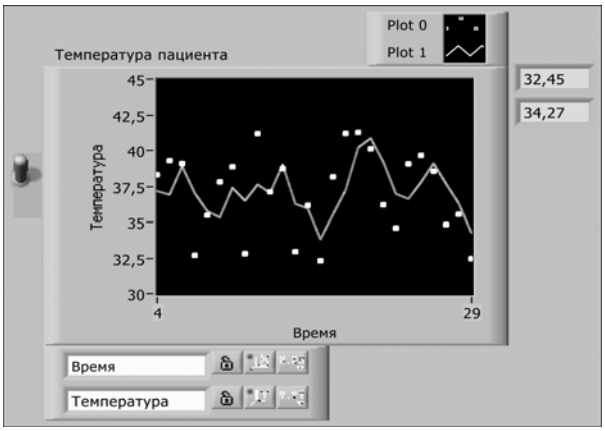
8.4. Завдання на самостійну роботу

Розробити ВП, в якому:

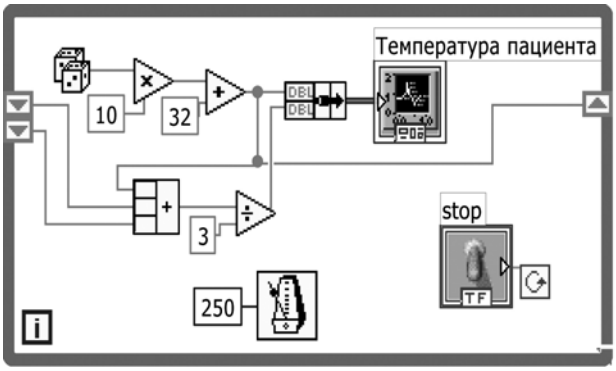
1) по випадковому закону генерується температура деякого пацієнта в діапазоні від 32°С до 42°С;

2) на екрані відображуються дві криві: одна – поточне значення температури, друга – середнє арифметичне по трьом останнім значенням.

Варіант реалізації приведений на рис. 8.5.



а)



б)

Рис. 8.5. Відображення на графіку поточного значення температури і середнього арифметичного по трьом останнім значенням

Додаткове завдання: передбачити можливість встановлення та відображення на графіку нижньої та верхньої границь температури.

8.5. Контрольні питання

1. Як змінити масштаб відображення на графіку осцилограми?
2. Як включити і відключити зображення одного з графіків, як змінити його колір і стиль?
3. Якою функцією можна об'єднати декілька графіків на одному екрані?

9. ГРАФІКИ (WAVEFORM GRAPH, XY GRAPH)

Графіки, так само як і графіки осцилограм, в LabVIEW призначені для відображення даних у графічній формі. На відміну від графіків осцилограм (chart), які відображають дані по мірі їх надходження (подібно до самописця), **графіки (graph)** відразу візуалізують вже сформовані масиви даних. Вони не можуть додавати нові значення до вже створених і відображених даних. LabVIEW пропонує декілька видів графіків для забезпечення зручності роботи: **графіки (Waveform graph), двокоординатні графіки (XY graph), графіки інтенсивності, тривимірні графіки, графіки цифрових осцилограм і деякі особливі види графіків (криві Сміта, графіки в полярних координатах, криві максимумів-мінімумів і криві розподілу).** Даний розділ присвячено графікам та двокоординатним (XY) графікам.

9.1. Мета та завдання

Метою заняття є огляд можливостей генерації графіків у системі LabVIEW. Для цього потрібно вирішити наступні задачі:

- зрозуміти функціональну відмінність графіків (graph) і графіків осцилограм (chart);
- з'ясувати відмінність у використанні графіків (Waveform graph) і двокоординатних графіків (XY graph);
- визначити відмінність типів вхідних даних для графіків (Waveform graph) і двокоординатних графіків (XY graph);
- навчитися змінювати масштаб графіків та використовувати елементи управління, панелі редагування;
- з'ясувати переваги і правила використання кластерів при відображенні

графіків.

9.2. Основи роботи з графіками

На передній панелі ВП графіки (Waveform graph) і двокоординатні графіки (XY graph) виглядають ідентично, але мають абсолютно різні функції.

Обидва типи графічних індикаторів можна викликати з підпалітри Графік (**Graph**) палітри Елементи управління. Графік (**Waveform graph**) відображає лише однозначні функції (одне значення Y відповідає певному значенню X) з рівномірно розташованими точками. Графік (Waveform graph) є ідеальним інструментом відображення масивів даних, в яких точки розподілені рівномірно.

Двокоординатний графік (**XY graph**) є декартовим графіком, використовуваним для відображення масивів даних з часовими інтервалами, що змінюються, або даних з декількома значеннями координати Y для кожного значення X, наприклад графік кола. Обидва типи графіків на передній панелі виглядають однаково, але мають різні типи вихідних даних, тому слід їх не плутати.

9.2.1. Однопроменевий графік

Для побудови однопроменевих графіків можна підключити масив значень Y безпосередньо до терміналу графіка. Цей метод заснований на припущенні, що початкове значення по осі X рівне 0 і що значення dX (тобто приріст координати X) дорівнює 1. Термінал графіка на блок-діаграмі з'являється у вигляді елементу відображення масиву.

Інколи потрібно змінити часові параметри графіка. Наприклад, дані почали збиратися в час, відмінний від початкового X = 0 (або $X_0 = 0$), або

відстань між вибірками не дорівнює стандартному приросту dX = 1. Для зміни часових параметрів необхідно об'єднати значення X_0, dX і масив даних у кластер, а потім з'єднати кластер з графіком.

9.2.2. Багатопроменевий графік

Для побудови на графіку декількох кривих необхідно створити масив (або двовимірний масив) даних. Термінали графіків міняють свій зовнішній вигляд залежно від структури даних, підключених до них (масив, кластер, масив кластерів і так далі), і типу даних (U16, DBL і тому подібне). Якщо на вході масив, то вважається, що початкове значення X рівне 0 і значення dX дорівнює 1.

Функція Створити масив (**Build array**) створює двовимірний масив з двох одновимірних масивів. Причому, цей двовимірний масив має два рядки і багато стовпців. За умовчанням графіки відображують кожен рядок у вигляді окремої осцилограми. Якщо дані є організованими стовпцем, то масив необхідно транспонувати перед тим, як відображати на графіку. Транспонування означає заміну індексів стовпців на індекси рядків і навпаки. В LabVIEW дана процедура здійснюється через контекстне меню графіка, опцією Транспонувати масив (**Transpose Array**) – ця опція меню має сірий колір (тобто не є активною), якщо до неї не підключений двовимірний масив. Це ж саме можна зробити, за допомогою функції Транспонувати 2D-массив (**Transpose 2D Array**), що знаходиться в підпалітрі Масив (**Array**) палітри Функції (**Functions**).

9.2.3. Двокоординатні графіки

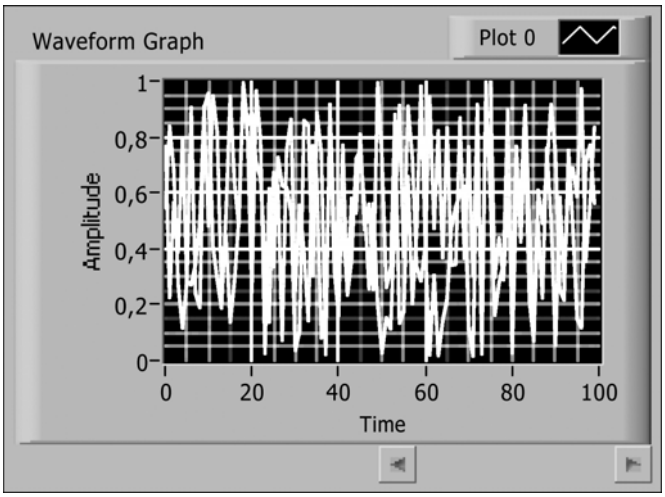
На вхід двокоординатного графіка подаються масиви X (верхній вхід) і Y (нижній вхід), об'єднані в кластер. Функція Об'єднати (Bundle)

об'єднує масиви X і Y в кластер, підключений до двокоординатного графіка. Термінал двокоординатного графіка виглядає у цьому випадку як елемент відображення кластера.

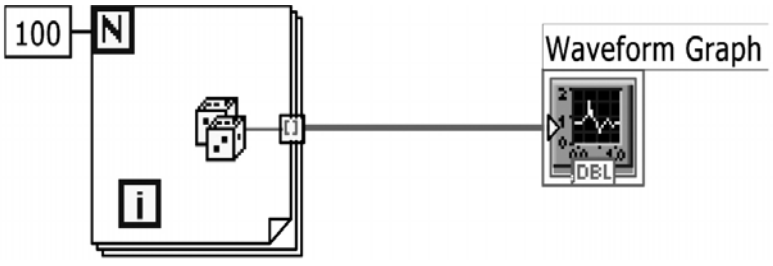
Для побудови багатопроменевих двокоординатних графіків слід створити масив кластерів значень X і Y [1, 331-349].

9.3. Приклади

Приклад 9.1. Ілюструється найпростіший спосіб роботи з графіком (Waveform graph) (рис. 9.1) – дані у вигляді випадкових чисел створюються в циклі, а на графік подаються у вигляді масиву, сформованого індексуючим тунелем.



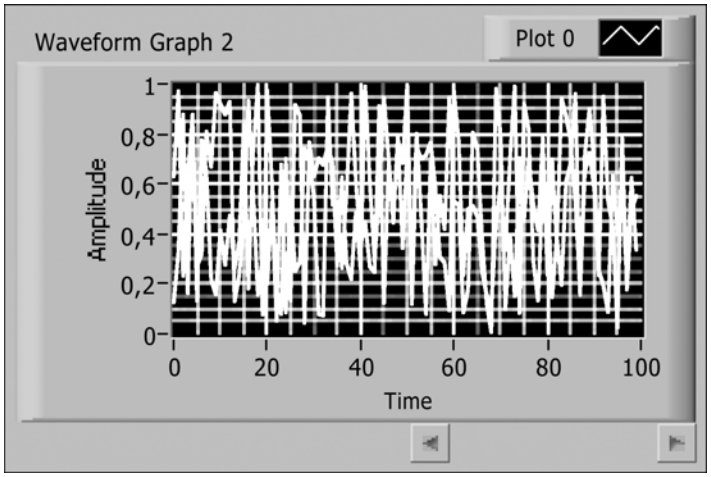
а)



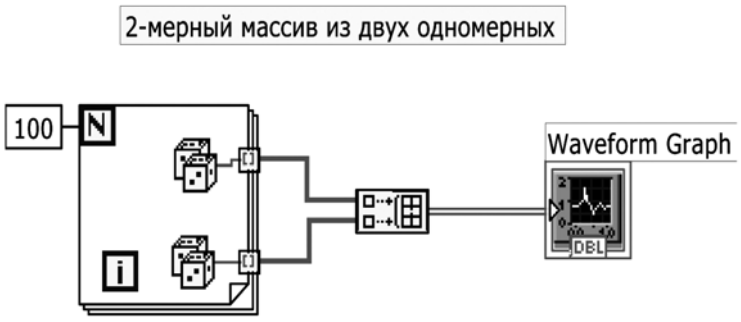
б)

Рис. 9.1 (а, б). Демонстрація роботи з графіком (Waveform graph)

Приклад 9.2. На рис. 9.2. ілюструється спосіб побудови на графіку (Waveform graph) декількох кривих шляхом створення двовимірного масиву даних, ідентичних тим, що використовувались в прикладі з однопроменевим графіком на рис. 9.1.



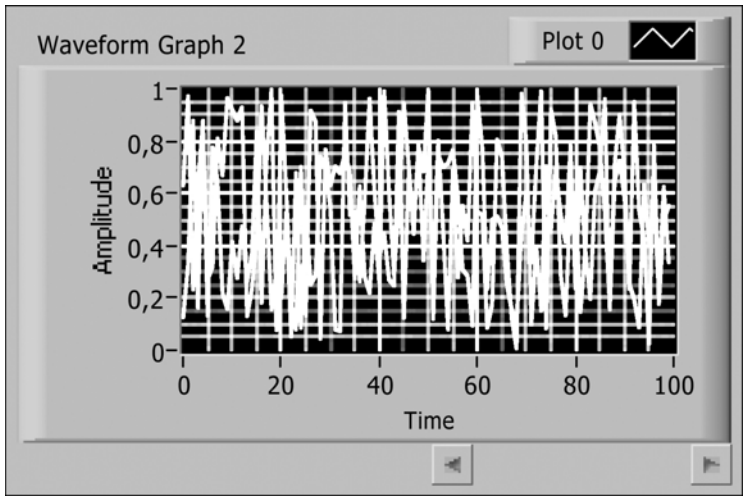
а)



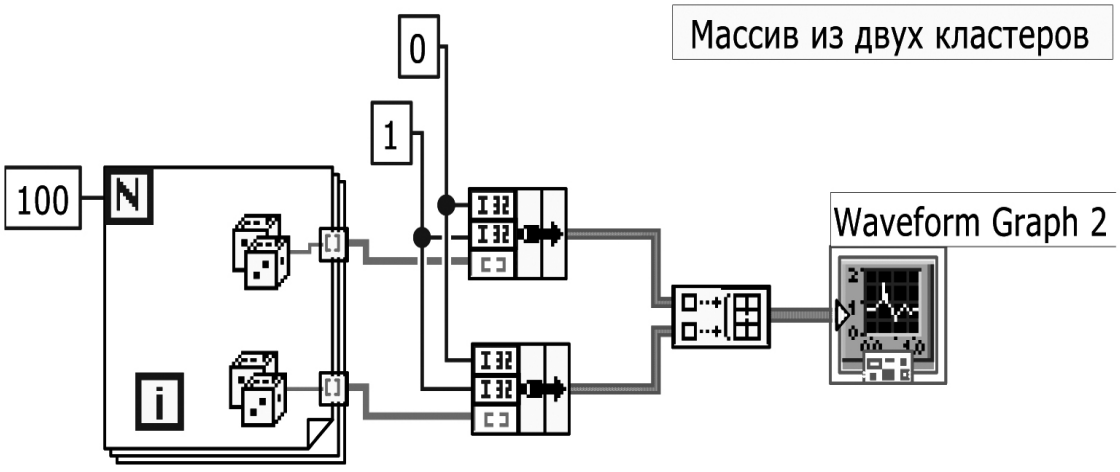
б)

Рис. 9.2 (а, б). Побудова на графіку (Waveform graph) декількох кривих

Приклад 9.3. Ілюструється спосіб побудови на графіку (Waveform graph) декількох кривих шляхом створення двовимірного масиву кластерів у разі, коли значення X_0 і dX задаються спеціально (рис. 9.3).



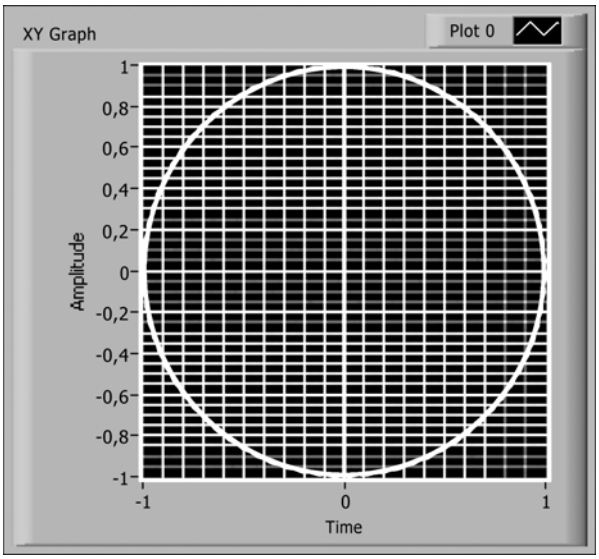
а)



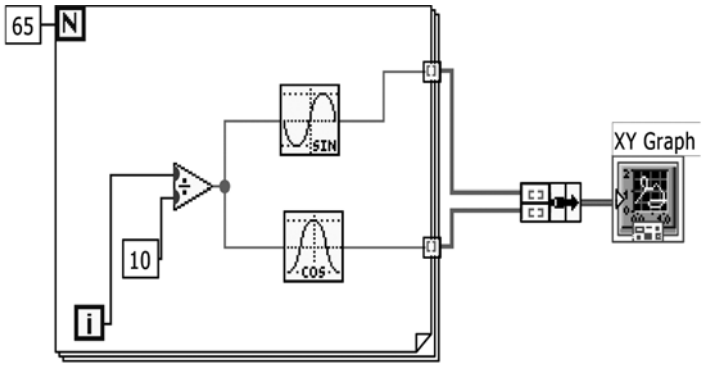
б)

Рис. 9.3 (а, б). Побудова на графіку (Waveform graph) двох кривих шляхом створення двовимірного масиву кластерів

Приклад 9.4. На рис. 9.4 показано, яким чином за допомогою двокоординатного графіку (XY Graphs) можна намалювати коло, тобто реалізувати можливість цього типу графіків відображувати для кожного значення x більш ніж одне значення y. Слід підкреслити, що на вхід двокоординатного графіку (XY Graphs) подається кластер масивів значень x (верхній вхід) і y (нижній вхід).



а)

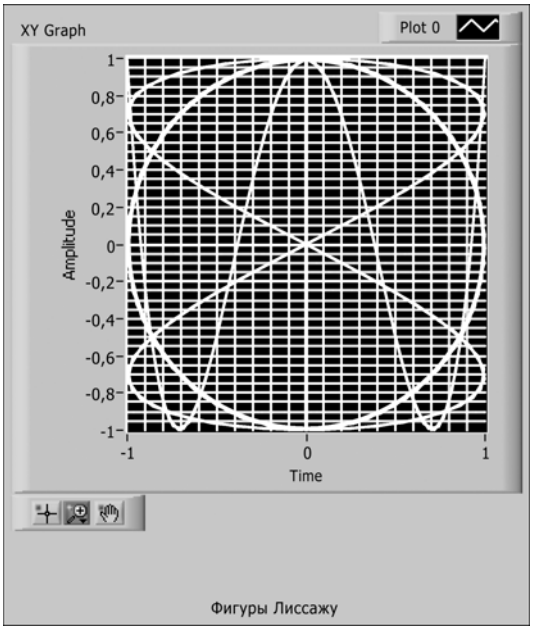


б)

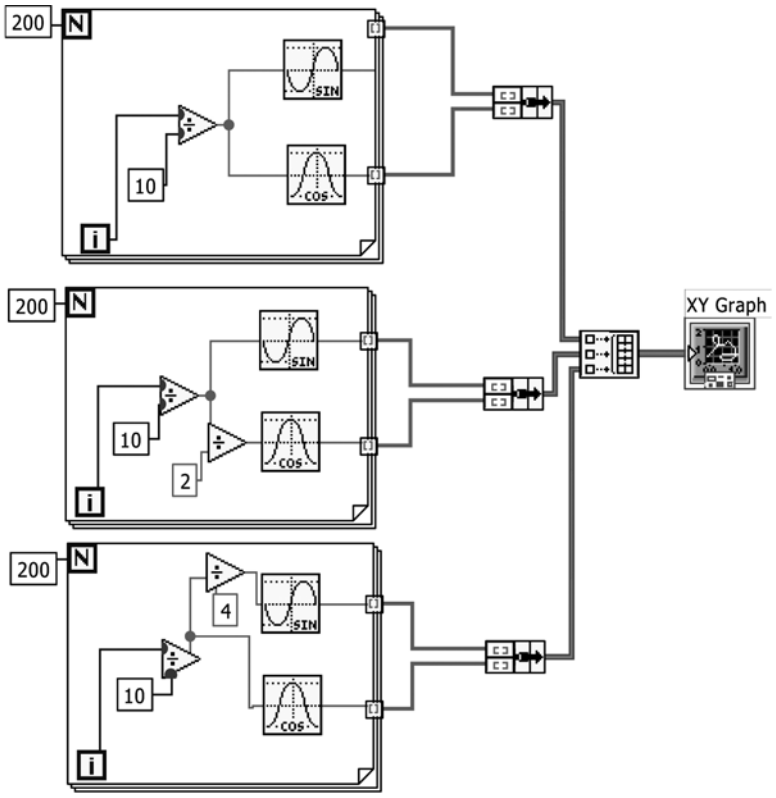
Рис. 9.4 (а, б). Ілюстрація можливості двокоординатних графіків відображувати для кожного значення x більш ніж одне значення y

9.4. Завдання на самостійну роботу

Розробити ВП, в якому на екрані графіку відображаються три криві, так звані фігури Ліссажу, а саме одна – це коло, тобто по осях координат відкладається sin(x) і cos(x); друга і третя аналогічно, лише по одній з вісей координат частота зміни значень більша в кратне число разів і складає, відповідно, cos(2x) і cos(4x). Варіант реалізації наведений на рис. 9.5.



а)



б)

Рис. 9.5 (а, б). ВП, що викреслює три графіки фігур Ліссажу

9.5. Контрольні питання

1. Дайте визначення різним типам графіків у LabVIEW.
2. Наведіть приклади застосування різних типів графіків у LabVIEW.
3. Які типи вхідних даних використовуються для кожного з типів графіків у LabVIEW?

10. ГРАФІКИ ІНТЕНСИВНОСТІ

Необхідність побудови залежності трьох змінних на двовимірному екрані або папері виникає досить часто. Для цього в LabVIEW застосовуються тривимірні графіки і графіки інтенсивності (**Intensity charts i graphs**), останні відображують дані в трьох вимірах на площині шляхом використання кольору як третього виміру (значення по осі Z).

Графіки інтенсивності є необхідним інструментом для відображення таких даних, як топографічні або температурні карти, де колір представляє висоту, глибину або розподіл температури.

10.1. Мета та завдання

Метою роботи є ознайомлення із методами застосування та візуалізації даних на графіках інтенсивності. Для цього необхідно виконати наступні задачі:

- навчитися працювати з графіками інтенсивності, необхідними для відображення тривимірних даних;
- вивчити тип даних «інтервал» (Time Stamp);
- вивчити тип даних «осцилограма» (Waveform), з яких компонентів він складається і як його використовувати.

10.2. Принципи роботи з графіками інтенсивності

Графіки інтенсивності діють по аналогії з двовимірними графіками, а колір представляє третю змінну. За допомогою опції Колірна шкала (**Color Scale**) ви можете встановити і відображувати карту-схему кольорів. Курсор графіка інтенсивності додатково показує значення Z.

Аби призначити колір значенню на колірній шкалі, клацніть правою кнопкою миші по відповідному маркеру і виберіть опцію Колір маркера (**Marker Color**). Можна створювати нові маркери, вибравши функцію Додати маркер (**Add Marker**) з контекстного меню колірної шкали, потім переносити їх в будь-яке зручне місце і призначати їм новий колір.

На вхід графіків інтенсивності необхідно подавати двовимірні масиви чисел, де кожне число в масиві є значенням кольору. Індекси кожного елементу масиву представляють місце розташування цього кольору на графіці.

10.2.1. Типи даних: інтервал і осцилограма

У багатьох технологічних і наукових дослідженнях велика частина даних, з якими працюють, є набором значень, що змінюються в часі. Наприклад, електрокардіограми це зміни напруги з часом. У LabVIEW є зручний засіб організації і роботи з подібними залежними від часу даними, – **типи даних інтервал (Time Stamp) і осцилограма (Waveform).** Осцилограма дає можливість зберегти не лише значення даних, але також відмітку про час набуття першого значення (за допомогою даних типу інтервал), часовий дискрет між точками даних і коментарі до даних. Як і при роботі з масивами і кластерами, тут можна додавати, віднімати і здійснювати багато інших дій безпосередньо з осцилограмами.

Можна створити елемент управління **Осцилограма (Waveform)** на передній панелі, узявши його з палітри Ввід/вивід (I/O). Відповідне відображення на блок-діаграмі – це термінал коричневого кольору.

Тип даних осцилограма є особливим типом кластера, який складається з чотирьох компонентів, – Y, t0, dt і Властивості (Attributes):

• Y – одновимірний масив даних, якими можуть бути або точки

даних, або інші осцилограми залежно від дії. Представленням одновимірного масиву є DBL;

• t0 – скалярна величина, яка представляє час (відповідно до системного годинника) надходження першої точки в масиві Y. Цю величину можна назвати початковим часом;

• dt – скалярна величина, що показує різницю часу між точками масиву Y;

• Властивості – за умовчанням цей компонент є «прихованим» (його можна побачити, якщо натиснути праву кнопку миші і вибрати опцію Видимі елементи -> Властивості. Це строковий тип даних, який дає можливість приєднати до даних осцилограми іншу інформацію, таку як номер приладу або номер каналу системи надходження даних.

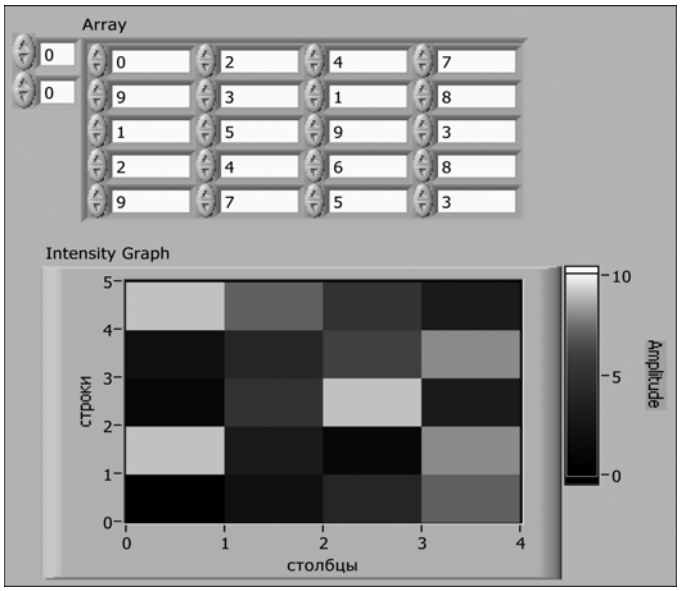
У палітрі Функції є підпалітра **Осцилограма (Waveform)**, присвячена роботі з осцилограмами.

Інтервал – це особливий тип даних для зберігання абсолютних моментів часу, наприклад, часу початку збору даних, з дуже високою точністю (19 знаків після коми). Наприклад, початковий момент часу (t0) зберігається в осцилограмі у вигляді даних типу інтервал.

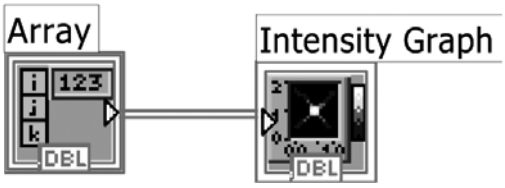
Інтервал – це не лише точний спосіб зберігання абсолютного моменту часу, його контролер – це зручний спосіб перегляду і редагування дати і часу. Контролер інтервал (**Time Stamp Control**) знаходиться в палітрі All Controls -> Numeric. Там же знаходиться відповідний індикатор (**Time Stamp Indicator**). Інтервал побудований за принципом відліку кількості секунд, що пройшли від 1 січня 1904 року з прив'язкою до часового поясу місця знаходження користувача [1, 359-363, 366-371].

10.3. Приклади

Приклад 10.1. Ілюструється принцип роботи з графіком інтенсивності (рис. 10.1) : на графік інтенсивності подаються дані у вигляді масиву. За допомогою шкали кольору (справа) задається відповідність між кольорами елементів і чисельними значеннями елементів масиву.



а)



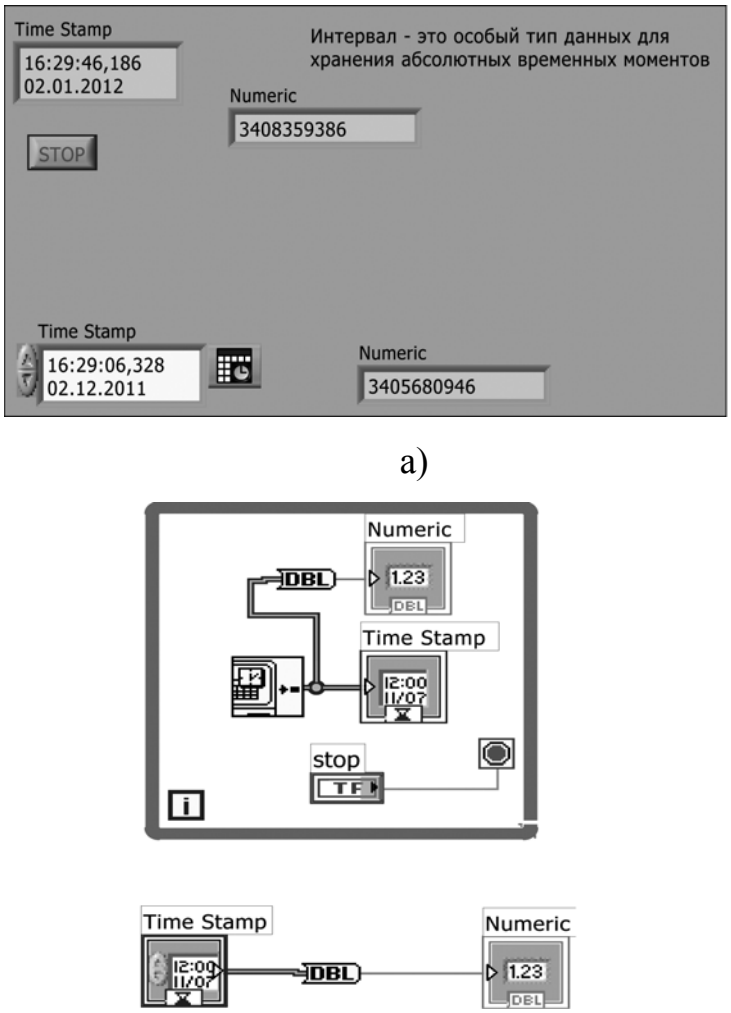
б)

Рис. 10.1. Ілюстрація принципу роботи з графіком інтенсивності (а – передня панель, б – блок-діаграма)

Приклад 10.2. Ілюструються прийоми роботи з типом даних інтервал (Time Stamp) (рис. 10.2). У верхній частині показаний віртуальний прилад, в якому використовується функція Get Date/Time In Seconds (з підпалітри Time & Dialog), що повертає інтервал (Time Stamp) поточного часу. Цей інтервал відображується на індикаторі Time Stamp, в результаті

чого отримуємо працюючий годинник-календар. Також інтервал перетворюється на тип даних DBL і відображується на звичайному числовому індикаторі. Тут ми можемо переконатися, що Time Stamp – це фактично число секунд, що пройшли від 1 січня 1904 року.

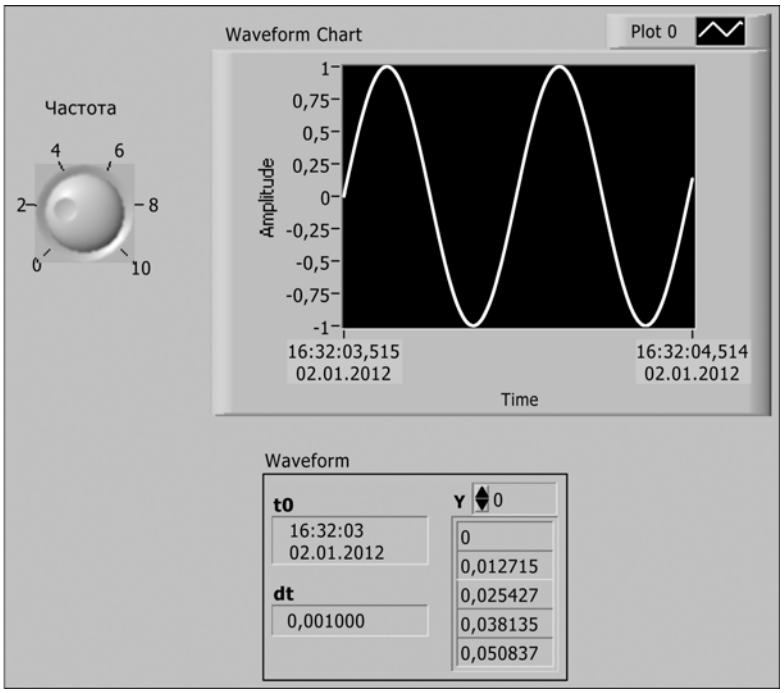
У нижній частині передньої панелі (Рис. 10.2) значення даних типу інтервал, задане за допомогою контролера Time Stamp, перетворюється на тип даних DBL і відображується на звичайному числовому індикаторі. Міняючи значення дати і часу на контролері, можна побачити на числовому індикаторі відповідне число секунд, що пройшли до цього значення часу.



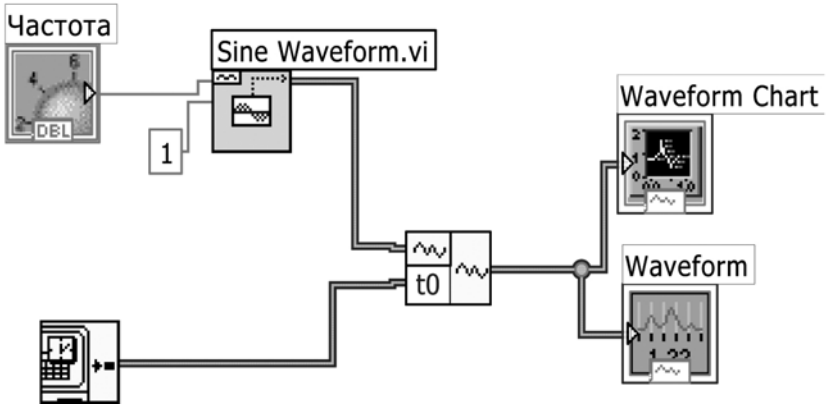
а)

Рис. 10.2. Ілюстрація прийомів роботи з даними типу інтервал (Time Stamp)

Приклад 10.3. На рис. 10.3 показано, яким чином формуються дані типу Waveform. Для цього використовується функція Build Waveform з підпалітри Waveform, яка фактично об'єднує в кластер масив даних (що генеруються функцією Sine Waveform) і Time Stamp реального часу, що повертається функцією Get Date/Time In Seconds. На горизонтальній вісі графіка видно, що час фіксується не відносно, а абсолютно.



а)



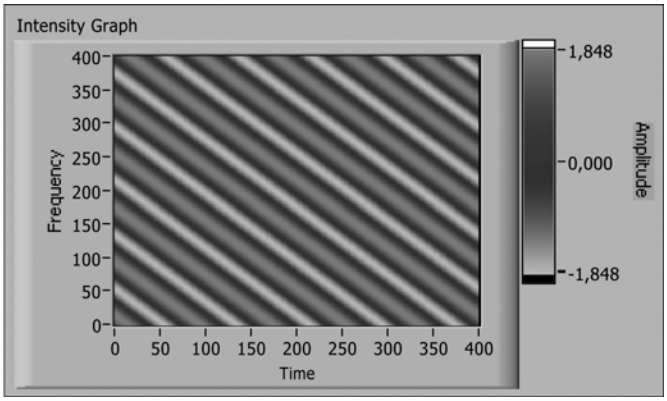
б)

Рис. 10.3. Формування даних типу Waveform (а – передня панель, б – блок-діаграма)

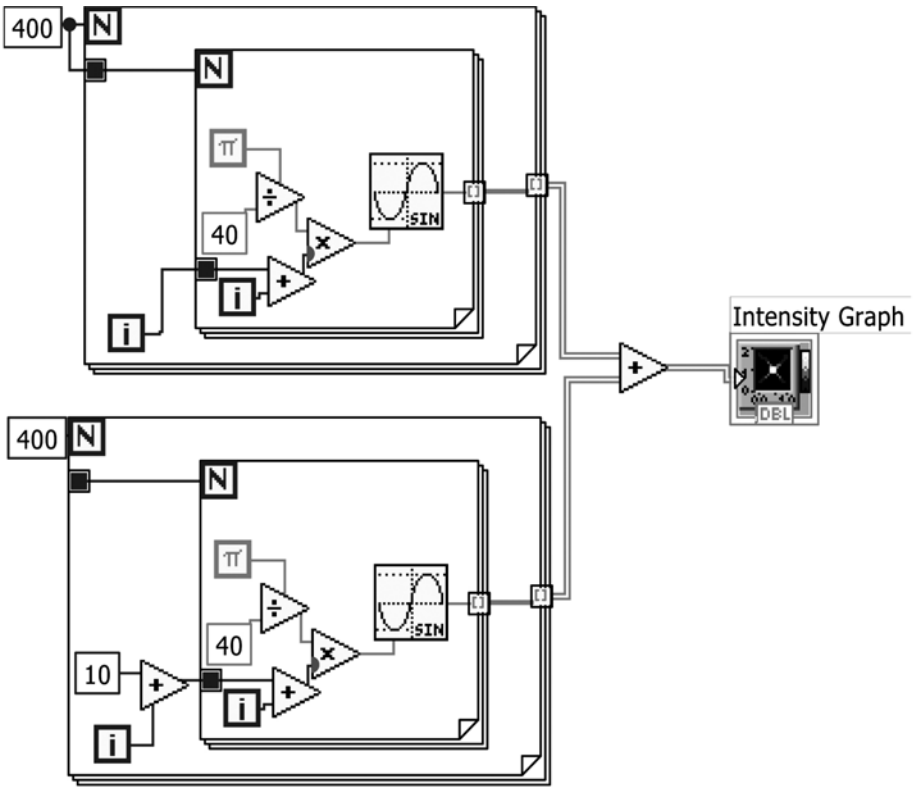
10.4. Завдання на самостійну роботу

Розробити ВП, в якому на графіку інтенсивності відображується інтерференція двох синусоїдальних функцій, зсунутих по фазі одна відносно іншої.

Варіант реалізації наведений на рис. 10.4.



а)



б)

Рис. 10.4. Інтерференція двох синусоїдальних функцій

10.5. Контрольні питання

1. Які задачі можливо вирішувати за допомогою графіків інтенсивності?
2. Як побудовані дані типа Waveform?
3. Які існують функції для роботи з даними типа Waveform і де вони знаходяться у LabVIEW?
4. У чому зручність застосування даних типу Time Stamp?

11. СТРОКОВІ ДАНІ

Строкові дані – один із типів даних в системі LabVIEW. Рядок в системі LabVIEW є набором символів ASCII. Часто рядки використовуються для простих текстових повідомлень. У інших випадках, наприклад, при управлінні зовнішніми приладами числові дані передаються у вигляді рядків символів, а потім для обробки даних рядки перетворюються на числа. Для зберігання числових даних на диску також часто використовуються рядки. Перш ніж зберегти числові значення у файлі, в багатьох підпрограмах вводу/виводу у/з файл(у) LabVIEW переводить їх у строкові дані.

11.1. Мета та завдання

Метою роботи є ознайомлення із строковими типами даних в системі LabVIEW. Для цього було поставлено такі задачі:

- вивчити тип даних строка (String);
- вивчити опції елементів управління строками і їх відображенням;
- ознайомитись із використанням функцій системи LabVIEW для роботи із строками;
- навчитися перетворювати числові дані в строкові та навпаки.

11.2. Основи роботи із строковими даними

Елементи управління і індикатори строкової змінної мають такі опції як режим відображення символів, які зазвичай є такими, що не відображуються, зокрема, знак повернення на один символ назад, знак повернення каретки і табуляція. Якщо вибрати опцію «Кодоване

відображення» (**Codes Display**) замість функції «Нормальне відображення» (**Normal Display**) з контекстного меню строк, то буде відображено всі символи.

Дані в рядку не змінюються при виборі режиму відображення; міняється лише вигляд деяких символів. Режим «Кодоване відображення» необхідний для налаштування програм і визначення «прихованих» символів, які потрібні для роботи з приладами, послідовним портом та іншими інструментами.

Рядки також мають опцію Приховане відображення (**Password Display**), яка заставляє елемент управління або відображення показувати «зірочку» замість кожного введеного символу, тому ніхто не зможе побачити, що ви друкуєте. Тоді як на передній панелі видно лише послідовність знаків (*****), на блок-діаграмі в строку передаються реальні дані.

Опція **Шістнадцятиричне відображення (Hex Display)** відображає строку у вигляді шістнадцятиричних символів, а не алфавітно-цифровых знаків, які виводяться за замовченням.

11.2.1. Таблиці

Таблиця в LabVIEW є спеціальною структурою, яка відображує двовимірний масив строк. Вона знаходиться в підпалітрі **Списки і таблиці (Lists & Tables) палітри Елементи управління (Controls).** У таблицях є заголовки рядків і стовпців, які можна зробити видимими або «прихованими»; заголовки відокремлені від даних тонкою подвійною лінією. Є можливість зміни заголовку тексту, використовуючи інструменти введення тексту (A) або управління («палець»). Також допустимо оновлювати або прочитувати заголовки за допомогою вузлів властивостей.

11.2.2. Використання функцій обробки строк

Функції роботи зі строками знаходяться у підпалітрі **Строка (String)** палітри Функції. Наприклад,

- функція **Довжина строки (String Length)** повертає число символів в даній строці.
- функція **Об'єднання строк (Concatenate Strings)** об'єднує всі вхідні строки в одну вихідну. Функція, що розміщена на блок-діаграмі, має вигляд ікони з одним входом.
- Функції перетворення строк у числа та навпаки: функція **Перетворити в строку (Format Into String)** і **Перегляд строки (Scan From String)**. Функція Перетворити в строку форматує вхідний аргумент (який записаний в числовому форматі) як строку відповідно до визначення формату, заданого у вхідній змінній формату строки (format string). Специфікації детально наводяться в керівництві по роботі з LabVIEW. Дана функція застосовує сформований шаблон до вхідних даних і видає результати на термінал результуючої строки (resulting string).

Символ «%» говорить про початок форматування. Наприклад в разі (% число1.число2) число1 визначає довжину результуючої строки, а число2 визначає точність (тобто кількість цифр після десяткової коми), f форматує вхідне число як число з плаваючою комою, d – як ціле, і e – як число з плаваючою комою в науковій системі позначень (експоненціальне представлення чисел).

Розмір функції «Перетворити в строку» може бути змінений для одночасного перетворення кількох значень в одну строку.

Функція **Викликати рядок дати/часу (Get Date/Time String)** з палітри «Час і діалоги» (Time&Dialog) виводить строку, що містить

поточну дату, і строку, що містить поточний час.

Функція **Виділення підстроки (String Subset)** здійснює доступ до окремої частини строки. Вона повертає підстроку, починаючи із символу з заданим номером (offset) та із заданою довжиною символів (length). Номером першого символу є 0.

Функція **Шаблон рядка (Match Pattern)** використовується для пошуку заданої структури символів в строці. Функція шукає і повертає знайдену підстроку. Ця функція шукає вираз відповідно до заданого (**regular expression**), починаючи із заданого символу. Як тільки функція знаходить вираз, вона розбиває рядок на три підрядки. Якщо ж вираз не виявлено, то вміст виводу відповідної підстроки (match substring) виявляється порожнім, а значення зсуву (offset past match) встановлюється в -1.

Функція **Перегляд рядка (Scan from String)** – зворотна до функції **Перетворити в рядок,** тобто вона перетворює рядок, що містить дійсні числові символи (від 0 до 9, +, -, e, E) в числові дані. Ця функція починає перегляд вхідної строки (input string) з місця початку пошуку (initial search location) і перетворює дані відповідно до специфікації формату строки. Дану функцію можна збільшити для одночасного перетворення декількох значень.

Як функція Перетворити в рядок, так і функція Перегляд рядка мають панель **Редагування виділення/створення строки (Edit Scan/Format String),** за допомогою якого можна задати формат строки. У цьому діалоговому вікні встановлюється формат, точність, тип даних і довжина значення [1, 389-398].

11.3. Приклади

Приклад 11.1. Розробити ВП, передня панель якого зображено на Рис. 11.1. Для реалізації застосувати функцію Об'єднання строк (**Concatenate Strings**), елемент «повернення каретки» та цикл For.

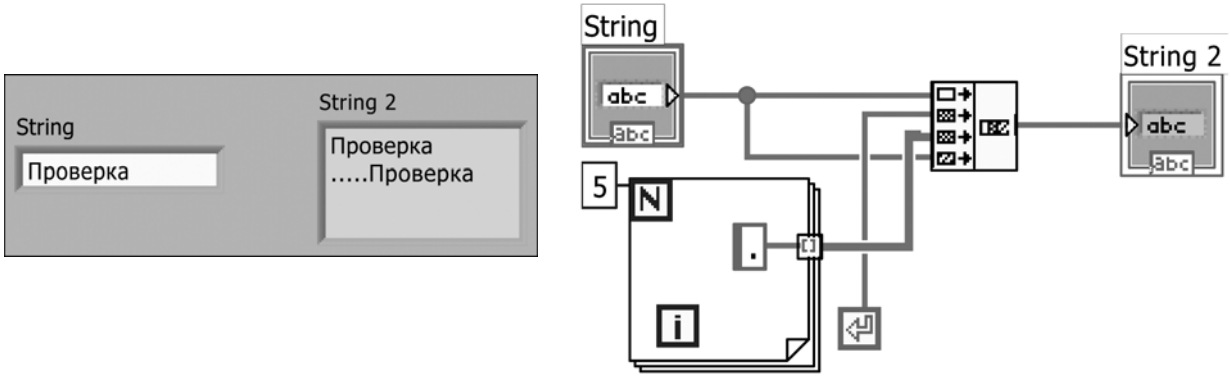


Рис. 11.1. Принципи роботи з функцією об'єднання рядків

Приклад 11.2. Розробити ВП, в якому використовується елемент Багатостовбцевого списку (**Multi-column Lisbox**) (рис. 11.2). Крім того, при виділенні строки у вікні-списку на індикаторі з'являється номер відповідної строки.

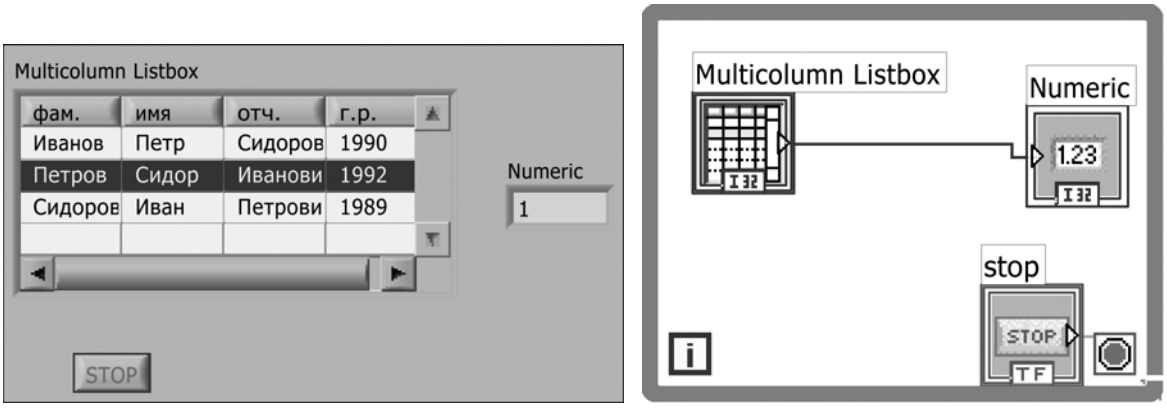
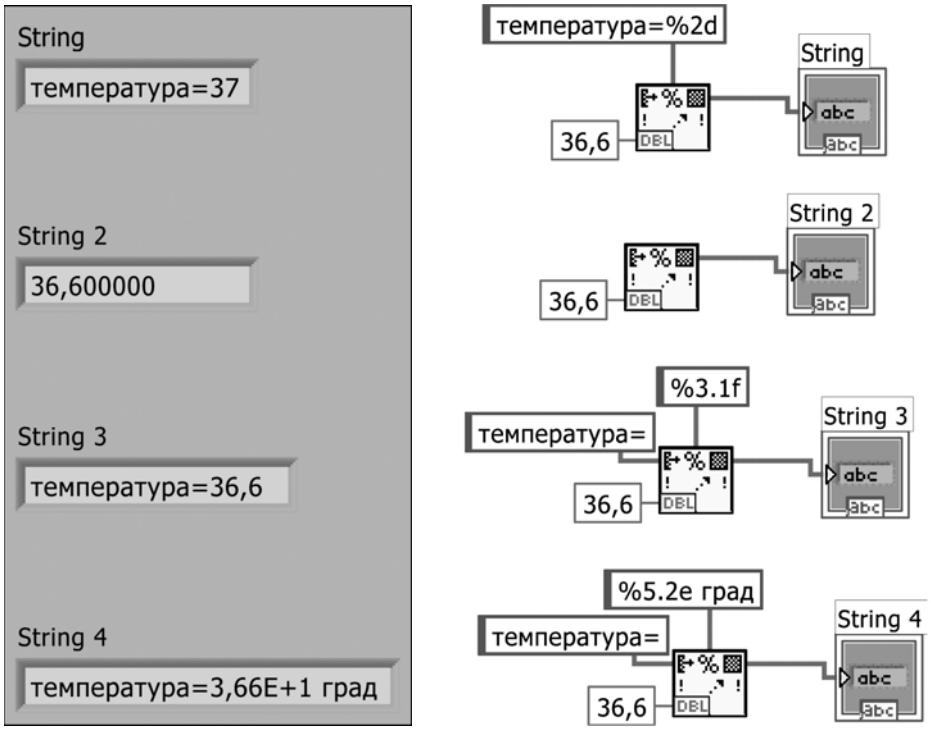


Рис. 11.2. Прийоми роботи з багатостовбцевим вікном-списком

Приклад 11.3. Розробити ВП, в якому за допомогою функції Перетворити в рядок (**Format Into String**) відображуються результати

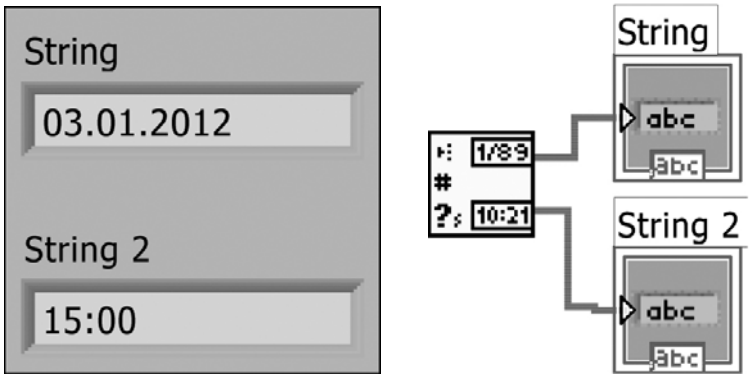
роботи програми відповідно до різних значень параметрів форматування.



а) б)

Рис. 11.3. Ілюстрація дії функції Format Into String (а – передня панель, б – блок-діаграма)

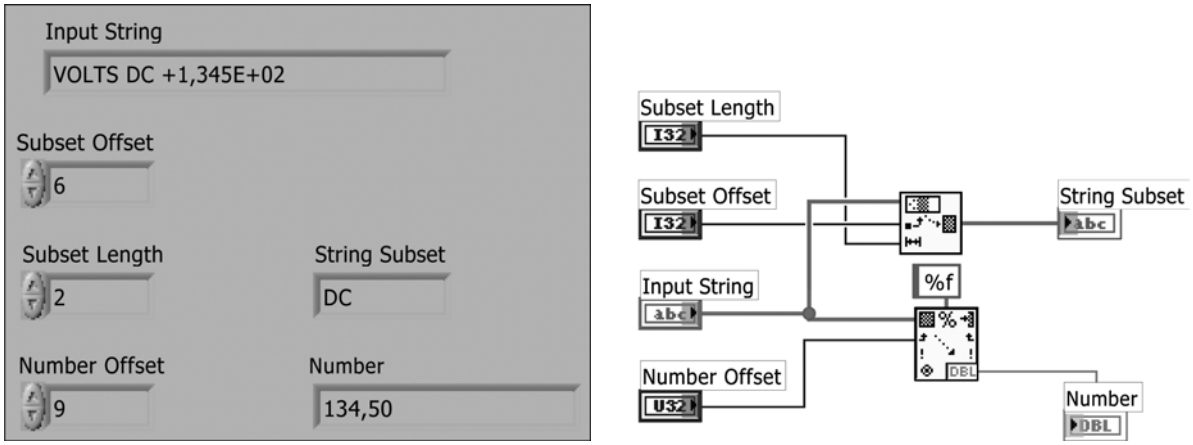
Приклад 11.4. На рис. 11.4 продемонстровано результати роботи функції Викликати рядок дати/часу (**Get Date/Time String**).



а) б)

Рис. 11.4. Ілюстрація дії функції Get Date/Time String (а – передня панель, б – блок-діаграма)

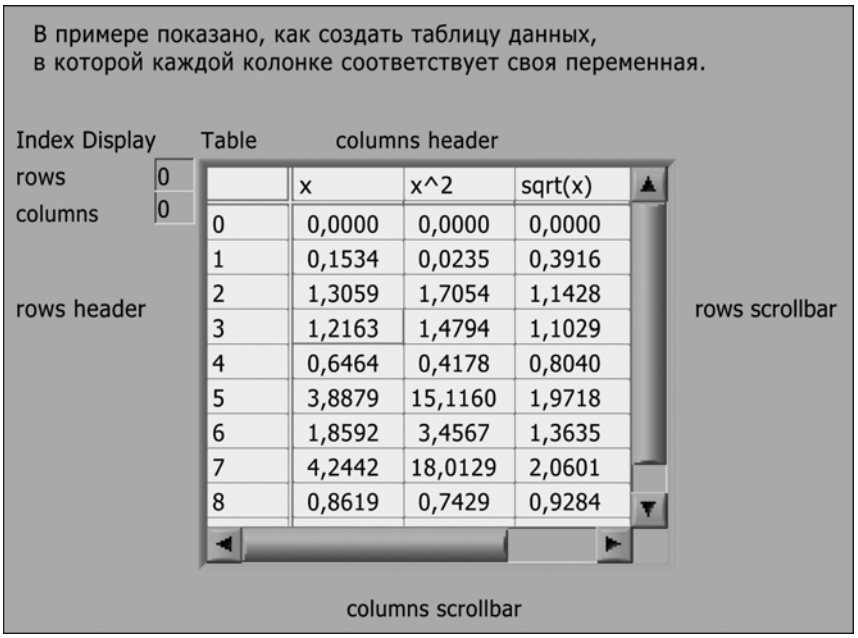
Приклад 11.5. На рис. 11.5 продемонстровано результати роботи функцій String Subset (що повертає підстроку відповідно до параметрів зсуву і довжини) і Scan From String (що перетворює частину строки у тип даних число).



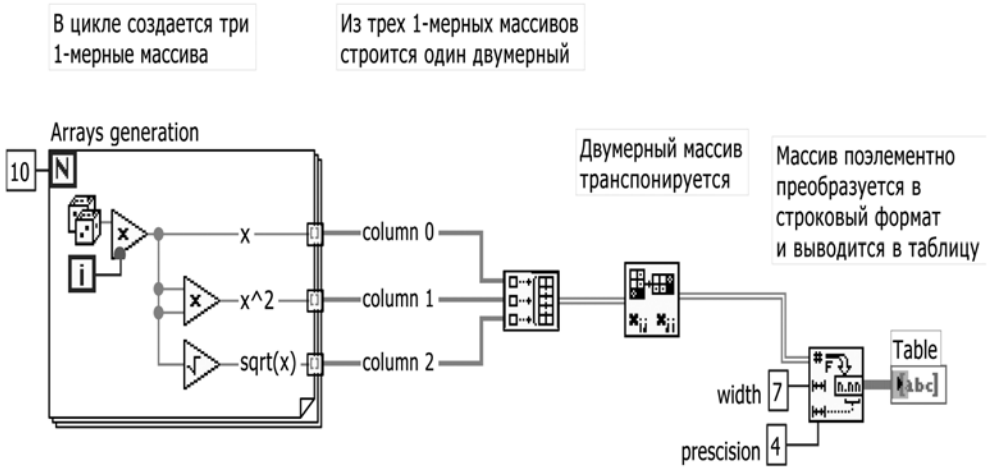
а) б)

Рис. 11.5. Ілюстрація дії функції String Subset (а – передня панель, б – блок-діаграма)

Приклад 11.6. На рис. 11.6 продемонстровано результати розробки таблиці із даними, де в кожній колонці своя змінна.



а)



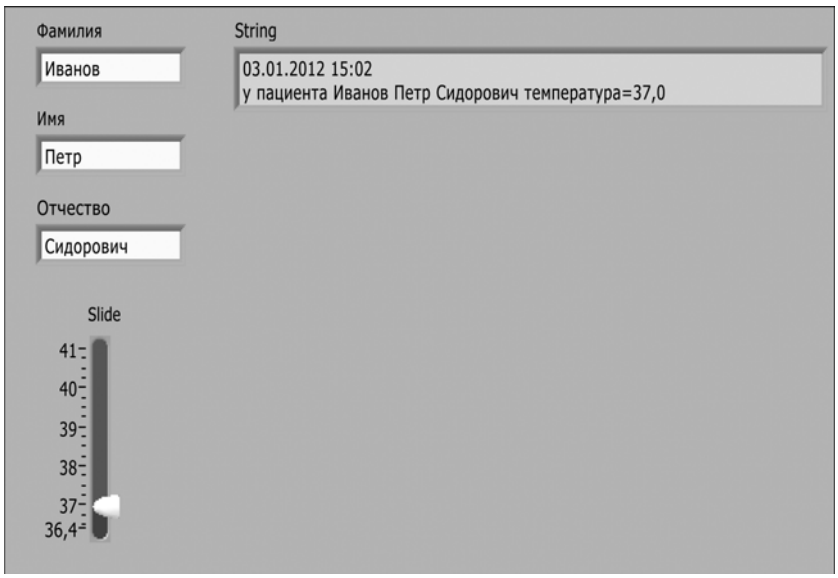
б)

Рис. 11.6. Ілюстрація створення таблиці даних, в якій кожній колонці відповідає своя змінна (а – передня панель, б – блок-діаграма)

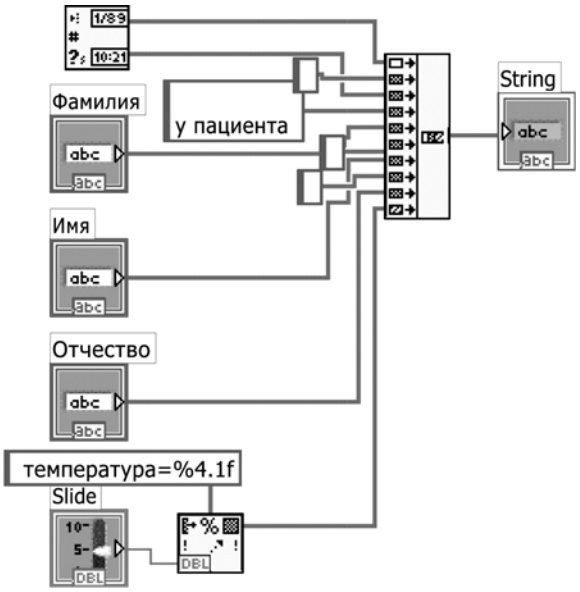
11.4. Завдання на самостійну роботу

Розробити ВП, який об'єднує і виводить в строку наступну інформацію: дата, час, “у пацієнта” Прізвище Ім'я По батькові температура = (значення зі змінної Slide).

Варіант реалізації наедено на рис. 11.7.



а)



б)

Рис. 11.7. Приклад роботи з даними типу String (а – передня панель, б – блок-діаграма)

11.5. Контрольні питання

1. Яка відмінність між числом в строковому форматі і в числовому?
2. Які існують функції для роботи із строковими даними? Охарактеризуйте їх.

12. ЗАПИС ТА ЗЧИТУВАННЯ ФАЙЛІВ

Функції запису/зчитування даних у/з файл(у) знаходяться в підпалітрі **Ввід/вивід файлів (File I/O)** палітри Функції.

12.1. Мета та завдання

Метою заняття є ознайомитись із можливостями запису або зчитування даних з/у файли різного формату. Для цього потрібно вирішити наступні задачі:

- навчитися використовувати операції вводу/виводу файлів для збереження і зчитування даних на диску;
- зрозуміти, в яких випадках доцільніше застосовувати табличний, текстовий або двійковий формат запису файлів;
- розібратися з типом даних «path» і правилами його використання;
- Вивчити варіанти створення і відкриття файлів для читання, запису, дозапису, перезапису і навчитися їх застосовувати.

12.2. Функції файлового вводу/виводу

Палітра функцій файлового вводу/виводу розділена на три частини: **функції високого рівня (high level File I/O), функції низького рівня (low level File I/O) і підпалітра функцій розширених можливостей (advanced File I/O).**

Функції файлового вводу/виводу високого рівня розташовані у верхньому рядку палітри Functions -> File I/O. Вони призначені для виконання основних операцій по вводу/виводу даних. Використання функцій файлового вводу/виводу високого рівня дозволяє скоротити час і

зусилля програмістів при записі або зчитуванні даних у/з файл(у). Функції файлового вводу/виводу високого рівня виконують запис і зчитування даних та операції відкриття і закриття файлу. За наявності помилок у програмному коді функції файлового вводу/виводу високого рівня відображують діалогове вікно з описом помилок і пропонують на вибір: продовжити виконання програми або зупинити її.

Функції файлового вводу/виводу низького рівня розташовані в середньому рядку палітри Functions -> File I/O. Додаткові функції роботи з файлами (Advanced File I/O) розташовані в палітрі Functions -> File I/O -> Advanced File Functions та призначені для управління окремими операціями над файлами.

Функції файлового вводу/виводу низького рівня використовуються для створення нового або звернення до раніше створеного файлу, запису і зчитування даних і закриття файлу. Функції низького рівня роботи з файлами підтримують всі операції, необхідні при роботі з файлами.

Стандартні операції вводу/виводу даних у/з файлу складаються з наступної послідовності дій:

- створення або відкриття файлу. Зазначення розташування існуючого файлу або «адреси» (path) для створення нового файлу відбувається за допомогою діалогового вікна LabVIEW. Після відкриття файлу у LabVIEW створюється посилання на нього (**refnum**);
- виконання операцій зчитування або запису даних у/з файл(у);
- закриття файлу;
- обробка помилок.

Для здійснення основних операцій файлового вводу/виводу використовуються наступні ВП і функції:

Open/Create/Replace File – відкриває, перезаписує існуючий файл, або

створює новий. Якщо file path (місце розміщення файлу) не вказано, ВП виводить на екран діалогове вікно, в якому можна створити новий або вибрати вже існуючий файл.

Read File – зчитує дані з файлу, визначеного по посиланню refnum, і передає дані у поле data, на поле count подається значення кількості зчитуваних даних. Зчитування даних починається з місця, визначеного елементами pos mode і pos offset, і залежить від формату файлу.

Write File – записує дані у файл, визначуваний по посиланню refnum.

Close File – закриває вказаний в посиланні refnum файл.

12.2.1. Обробка помилок

Підпрограми, ВП і функції низького рівня містять інформацію про помилки. Для їх обробки використовуються підпрограми обробки помилок, такі як Simple Error Handler.VI (ВП «Простий обробник помилок»), розташований в палітрі Functions -> Time & Dialog. Поля введення error in і виведення error out інформації про помилки використовуються в кожному ВП для обміну інформацією про помилки між ВП.

Під час роботи ВП LabVIEW перевіряє наявність помилок в кожному вузлі. Якщо LabVIEW не знаходить помилок, то вузол виконується нормально. Якщо LabVIEW виявляє помилку в одному вузлі, то його виконання припиняється, а інформація про помилку передається наступному вузлу. І так далі, де в кінці виконання система LabVIEW повідомляє про помилки.

12.2.2. Збереження даних в новому або існуючому файлі

У файл, створений (або відкритий) за допомогою функцій файлового вводу/виводу, можна записати дані будь-якого типу. При необхідності

доступу до файлу з боку інших програм або користувачів, слід записувати дані у вигляді строки ASCII символів.

Доступ до файлу можна здійснити або безпосередньо у програмі або з використанням діалогового вікна.

При відкритті файлу, ВП або при мережевому з'єднанні LabVIEW створює посилання на об'єкт. Посилання (refnum) є унікальним ідентифікатором для таких об'єктів як файл, програма та мережеве з'єднання. Всі операції з відкритими об'єктами виконуються з використанням посилань.

Кластер помилок і посилання на файл послідовно передаються від одного вузла до іншого. Оскільки вузол не може виконатися, поки не визначені всі його вхідні поля даних, ці два параметри заставляють вузли працювати в певному порядку. Підпрограма ВП Open/Create/Replace File. VI передає посилання на файл і кластер помилок функції Write File, яка проводить запис файлу на диск. Функція Close File закриває файл після отримання кластера помилок і посилання на файл з функції Write File.

Підпрограма ВП Simple Error Handler.VI перевіряє наявність помилок і виводить інформацію про них в діалоговому вікні. Якщо в одному з вузлів допущена помилка, подальші вузли не виконуються, і кластер помилок передається в підпрограму ВП Simple Error Handler.VI.

12.2.3. Функції файлового вводу/виводу високого рівня

Функції файлового вводу/виводу високого рівня розташовані у палітрі Functions -> File I/O. Вони призначені для виконання дій файлового вводу або виводу даних наступних типів:

- символів в/із текстових файлів;
- строк з текстових файлів;
- одновимірних або двовимірних масивів числових даних одинарної

точності в/із файлу електронної таблиці;

- одновимірних або двовимірних масивів числових даних одинарної точності або цілочисельних 16-розрядних в/із бінарного файлу.

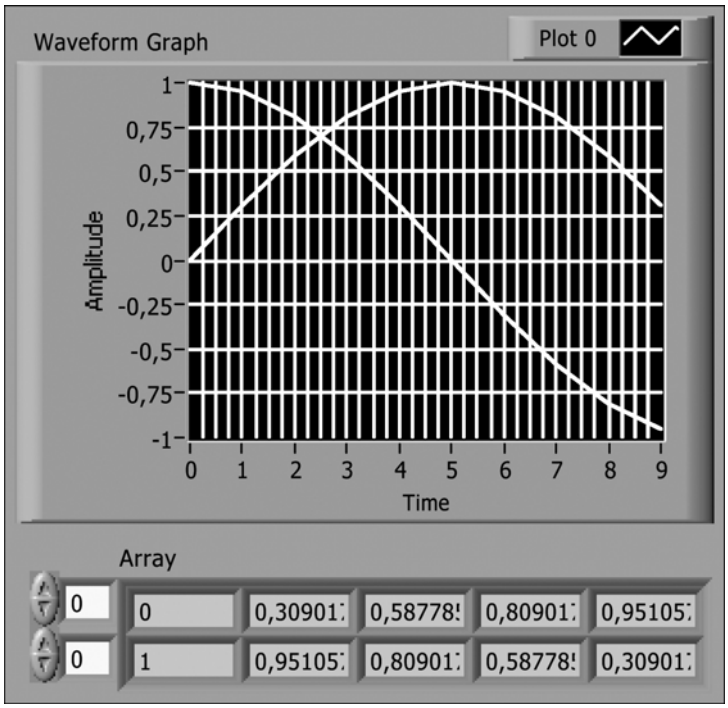
Функції файлового вводу/виводу високого рівня включають:

- Write to Spreadsheet File – перетворює 2D або 1D масив числових даних одинарної точності в текстовий рядок і записує рядок в новий або додає у вже існуючий файл. При цьому можна також транспонувати дані.
- Read From Spreadsheet File – прочитує певне число рядків від початкового зсуву start of read offset і перетворює дані в 2D масив числових даних одинарної точності. ВП відкриває файл перед читанням і після всіх операцій закриває його. Цей ВП можна використовувати для читання таблиці символів, збереженої в текстовому форматі.
- Write Characters to File – записує рядок символів в новий файл або додає його в той, що вже існує. ВП відкриває або створює файл перед записом і після всіх операцій закриває його.
- Read Characters From File – прочитує кількість символів number of characters від початкового зсуву start of read offset. ВП відкриває файл перед читанням і після всіх операцій закриває його.
- Read Lines From File – прочитує певне число рядків з текстового або бінарного файлу з положення start of read offset. ВП відкриває файл перед читанням і закриває його після.
- Binary File – читає і записує файл в бінарному форматі. Дані можуть бути цілочисельного типу або числовими даними одинарної точності з плаваючою крапкою [1, 407-412].

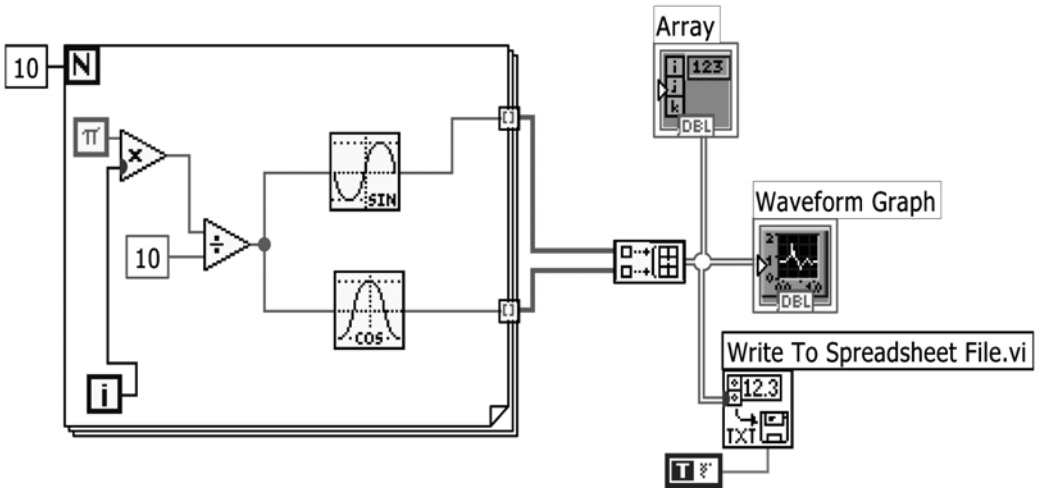
12.3. Приклади

Приклад 12.1. На рис. 12.1 проілюстровано використання функції

високого рівня запису в файл табличного формату Write to Spreadsheet File. У циклі створюється два одновимірні 10-елементні масиви значень функцій sin(x) і cos(x), які потім об'єднуються в 2-мірний масив. Цей масив відображується на передній панелі у вигляді індикатора масиву і у вигляді графіка, а також за допомогою функції Write to Spreadsheet File дані записуються у файл, адреса якого запрошується в діалоговому вікні.



а)



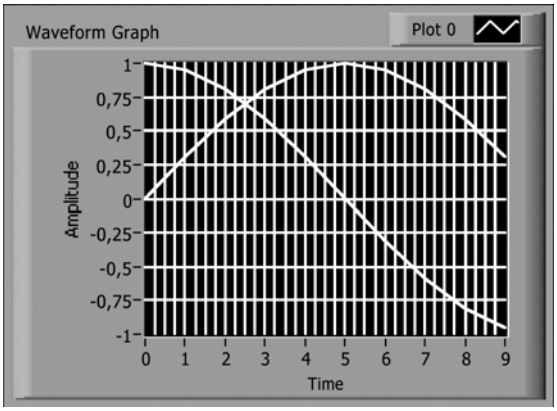
б)

0	1
0,309	0,951
0,588	0,809
0,809	0,588
0,951	0,309
1	0
0,951	-0,309
0,809	-0,588
0,588	-0,809
0,309	-0,951

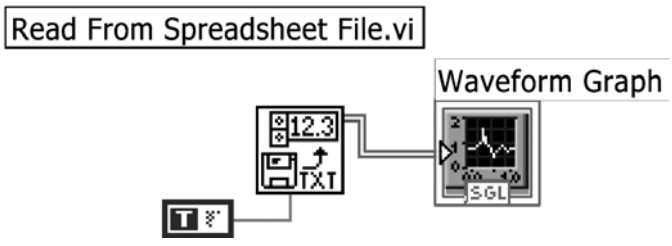
в)

Рис. 12.1. Приклад запису інформації в файл табличного формату (а – передня панель, б – блок-діаграма, в – результат роботи програми)

Приклад 12.2. На рис. 12.2 проілюстровано віртуальний прилад для зчитування даних з файлу за допомогою функції Read From Spreadsheet File. Дані було отримано у Прикладі 12.1.



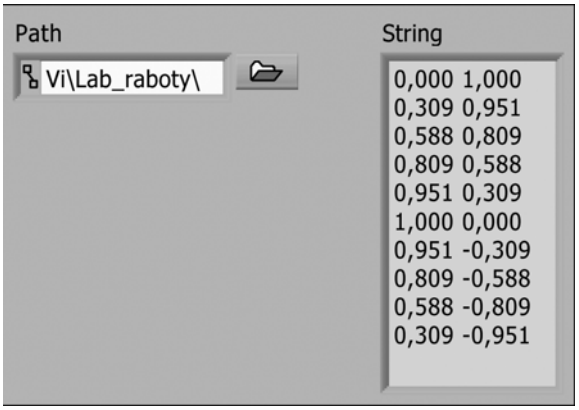
а)



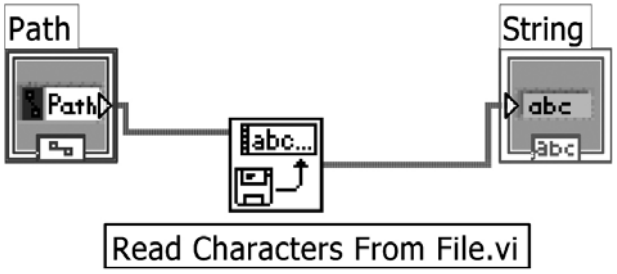
б)

Рис. 12.2. Приклад зчитування інформації з файлу табличного формату (а – передня панель, б – блок-діаграма)

Приклад 12.3. На рис. 12.3 демонструється зчитування файлу, створеного в прикладі 12.1, за допомогою функції символьного введення Read Characters From File. В результаті бачимо ті ж значення у вигляді таблиці. Адреса файлу задається за допомогою управляючого елементу ”Path” (Путь).



а)



б)

Рис. 12.3. Приклад зчитування інформації з файлу за допомогою функції Read Characters From File (а – передня панель, б – блок-діаграма)

Приклад 12.4. На рис. 12.4 продемонстровано використання функції Write To SGL File для запису у файл значень синусоїди в чисельному (а не в текстовому) вигляді. Ці ж значення відображуються на графіку і на індикаторі масиву.

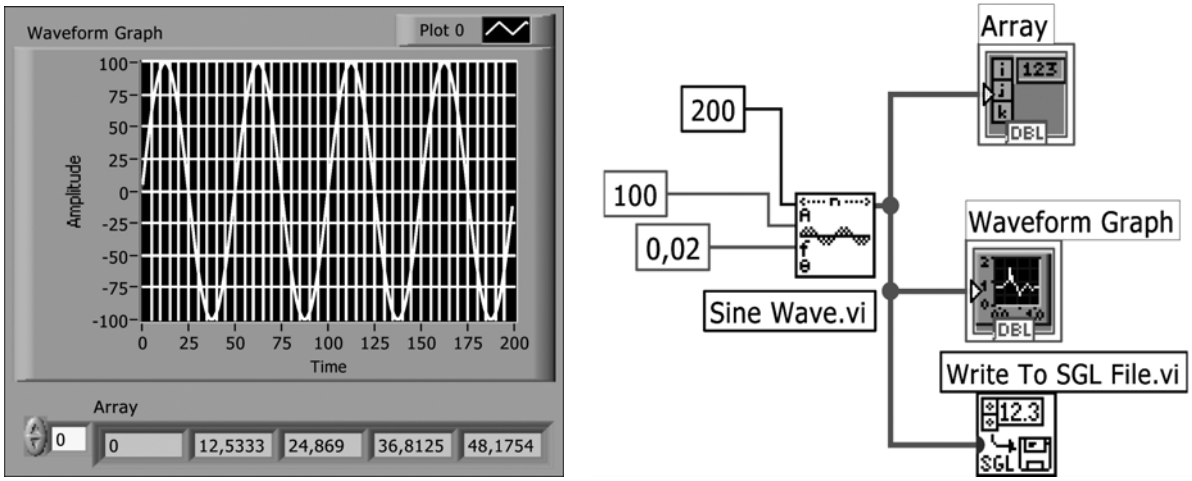


Рис. 12.4. Використання функції Write To SGL File
для запису інформації у файл

Приклад 12.5. На рис. 12.5 продемонстровано зчитування даних за допомогою функції Read From SGL File і відображення на графіку інформації, записаної у файл в прикладі 12.4.

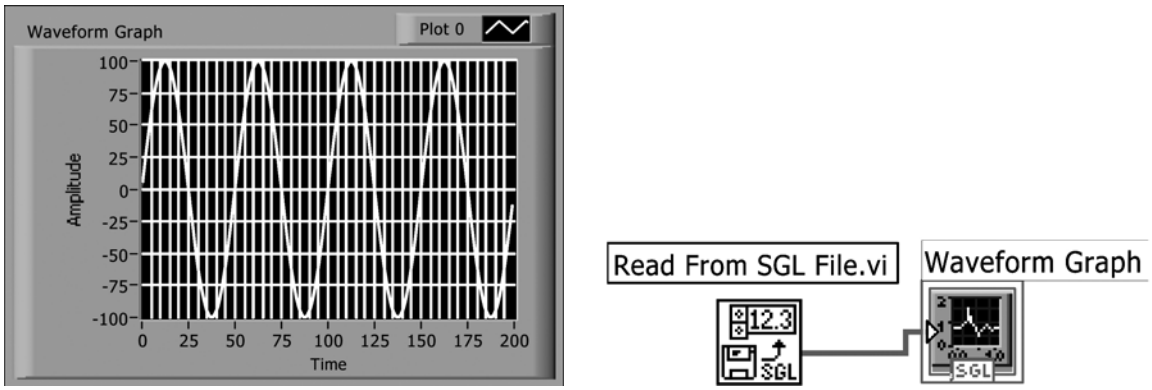
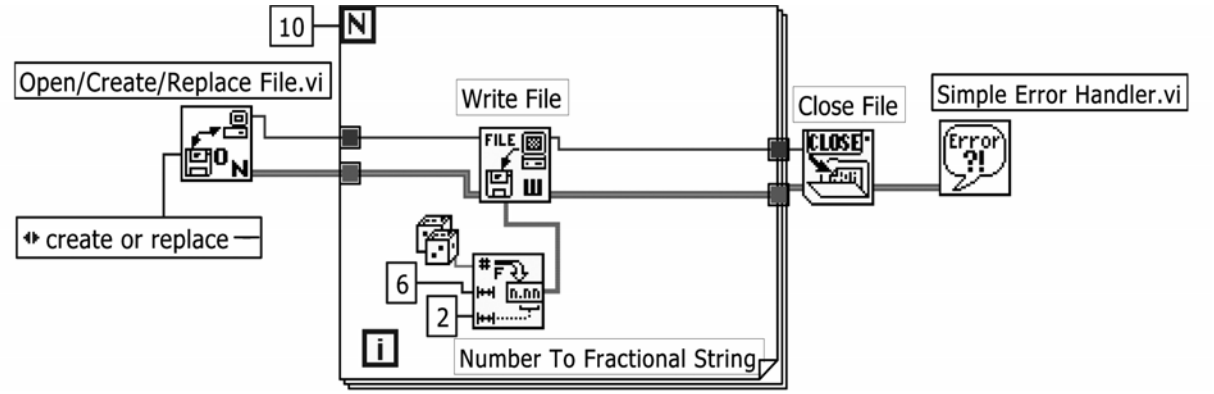


Рис. 12.5. Використання функції Read From SGL File
для зчитування інформації з файлу

Приклад 12.6. На рис. 12.6 продемонстровано використання функцій вводу/виводу низького рівня. Функцією Open/Create/Replace File з управляючим параметром create or replace відкривається існуючий файл або створюється новий. Далі в циклі генерується 10 випадкових чисел, які

перетворюються функцією Number To Fractional String в строковий формат (довжина разом з розділяючими пробілами дорівнює 6, а число знаків після коми дорівнює 2). Далі вони записуються функцією Write File в раніше створений або відкритий файл. Маршрут до файлу передається між функціями за допомогою посилання (refnum) – верхній з'єднувальний зв'язок. Кластер помилок передається по нижньому провіднику. На завершення файл закривається функцією Close File, а можливі помилки обробляються функцією Simple Error Handler.



а)

0,57 0,08 0,35 0,38 0,80 0,37 0,50 0,05 0,53 0,51

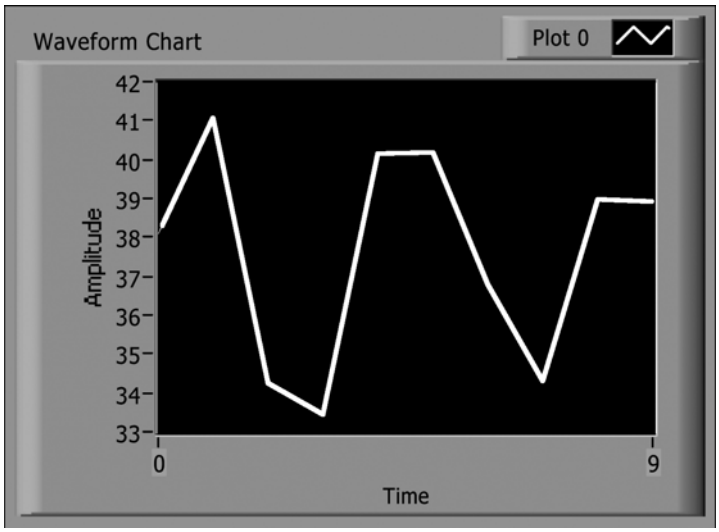
б)

Рис. 12.6. Приклад застосування функцій вводу/виводу низького рівня (а – блок-схема програми, б – результат роботи програми)

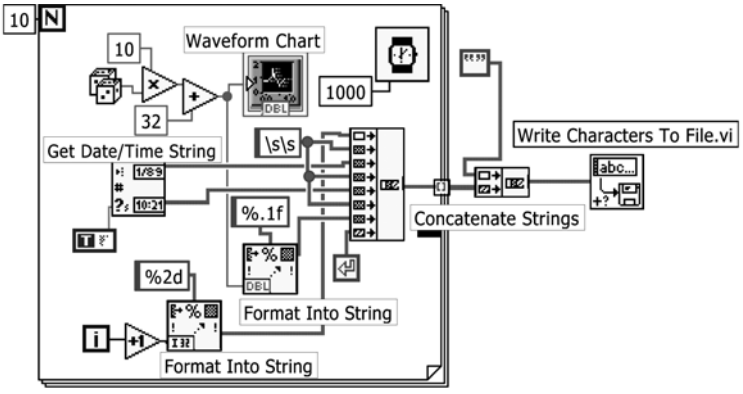
12.4. Завдання на самостійну роботу

Розробити ВП, який проводить 10 вимірів температури з частотою один раз в секунду і відображає їх на графіку осцилограми. Результати програми зберігаються у табличній формі в форматі, зображеному на рис. 12.7 (в).

Варіант реалізації приведено на рис. 12.7.



а)



б)

```
 1  04.01.2012  11:40:27  38,1
 2  04.01.2012  11:40:28  41,1
 3  04.01.2012  11:40:29  34,2
 4  04.01.2012  11:40:30  33,4
 5  04.01.2012  11:40:31  40,1
 6  04.01.2012  11:40:32  40,2
 7  04.01.2012  11:40:33  36,8
 8  04.01.2012  11:40:34  34,3
 9  04.01.2012  11:40:35  39,0
10  04.01.2012  11:40:36  38,9
```

в)

Рис. 12.7. ВП, який проводить 10 вимірів температури через кожну секунду і будує їх на графіку осцилограми (а,б), а також записує в файл (в)

12.5. Контрольні питання

1. Яким чином в системі LabVIEW реалізовано алгоритм запису/зчитування даних з/у файл?
2. Яким чином в системі LabVIEW реалізовано алгоритм обробки помилок? Що таке кластер помилок?
3. Які існують функції для запису і зчитування файлів?

13. ЛОКАЛЬНІ ЗМІННІ

Локальні і глобальні змінні в LabVIEW з технічної точки зору є структурами. Аналогічні поняття є в традиційних текстових алгоритмічних мовах програмування (C, Паскаль тощо).

13.1. Мета та завдання

Метою заняття є ознайомитись з поняттям «локальні змінні» та з принципами їх застосування у системі LabVIEW. Для цього потрібно вирішити такі задачі:

- ознайомитись з принципи роботи локальних змінних;
- вивчити, як, коли і де є доцільною реалізація локальних змінних;
- засвоїти, що вживання локальних змінних порушує основний принцип LabVIEW – управління потоком даних;
- розібратися з використанням режимів (mode) запису (WRITE) і читання (READ) локальних змінних.

13.2. Основи роботи з локальними змінними

Локальні змінні (local variables, або locals) забезпечують доступ до об'єктів передньої панелі з різних точок блок-діаграми одного і того ж віртуального приладу в тих випадках, коли немає можливості підключення провідника до терміналу об'єкту.

Локальні змінні в LabVIEW є вбудованими об'єктами, доступ до них здійснюється з підпалітри Структури (Structures) палітри Функції. При виборі об'єкту “локальна змінна” на блок-діаграмі спочатку з'являється вузол, помічений знаком питання (?), що вказує на

невизначеність типу локальної змінної. Якщо клацнути інструментом управління («палець») по цьому вузлу, з'являється список всіх поточних індикаторів і елементів управління, вибір одного з них визначає значення локальної змінної.

Є можливість викликати контекстне меню локальної змінної (натиснувши праву кнопку миші) і обрати опцію “Вибрати елемент” (Select Item) для доступу до списку.

Ще один метод створення локальної змінної полягає у виклику контекстного меню терміналу об'єкту і виборі опції Створити -> Локальна змінна (Create -> Local Variable).

Існує дві основні причини доцільності використання локальних змінних у віртуальному приладі:

• можливість реалізації алгоритмів, таких як управління паралельними циклами за допомогою однієї змінної, що неможливо реалізувати іншим способом;

• будь-який елемент управління може також відображати дані (працювати як індикатор) і навпаки.

Локальні змінні мають два режими (mode): читання (READ) і запис (WRITE). Термінал локальної змінної може знаходитися лише в одному з цих режимів, але допустимо створити другий локальний термінал до тієї ж самої змінної в іншому режимі. Поняття режиму є досить очевидним: у режимі читання можна прочитувати значення з терміналу локальної змінної, а в режимі запису можна записати дані в термінал локальної змінної.

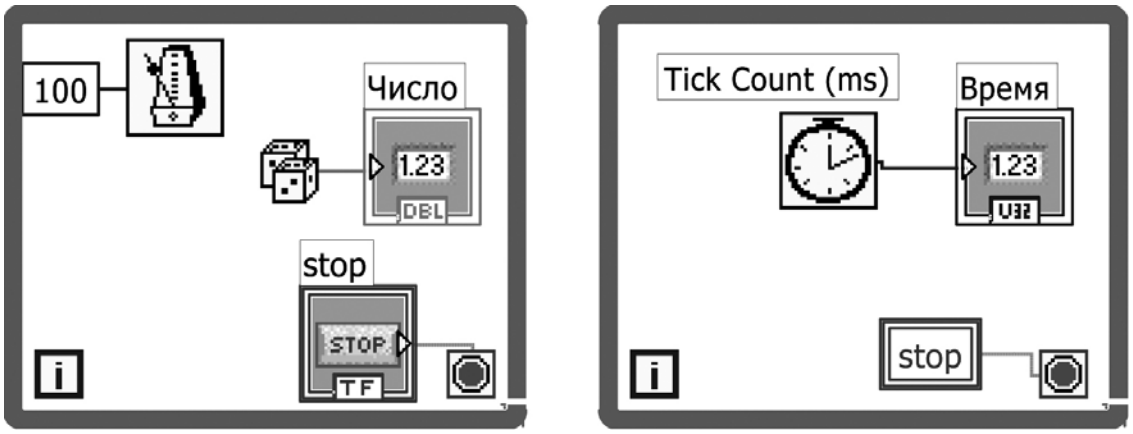
Встановлення локальної змінної в режим читання або запису здійснюється натисканням на праву кнопку миші на терміналі локальної змінної і вибором опції “Змінити на” (Change To). Термінал локальної змінної в режимі читання обведений жирнішою лінією, ніж в режимі

запису (точно так, як і елемент управління має товсту окантовку на відміну від індикатора) [1, 542-547].

13.3. Приклади

Приклад 13.1. Приклад застосування локальної змінної зображено на рис. 13.1. Тут потрібно завершити виконання двох незалежних циклів по умові за допомогою одного логічного елементу управління Стоп.

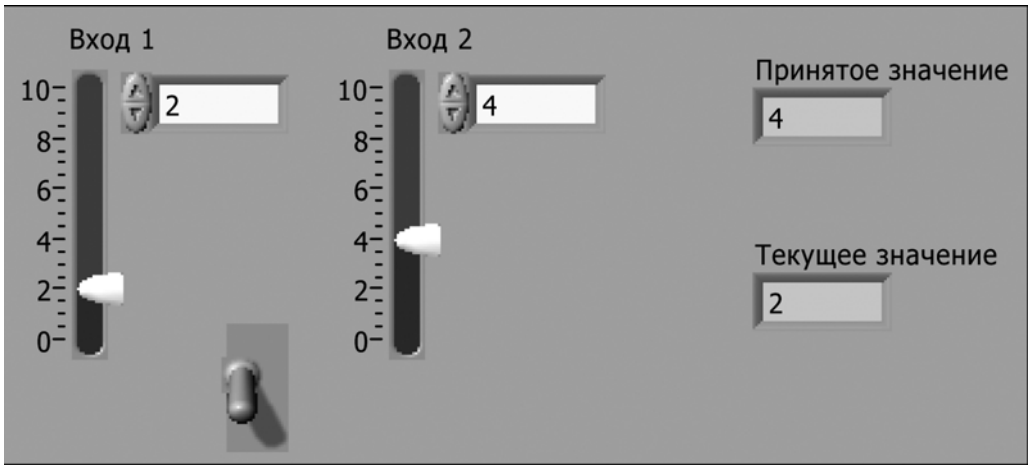
а)



б)

Рис. 13.1. Приклад використання локальної змінної (а – передня панель, б – блок-діаграма)

Приклад 13.2. На рис. 13.2 зображено приклад небажаного випадку створення умови змагання при застосуванні локальної змінної.

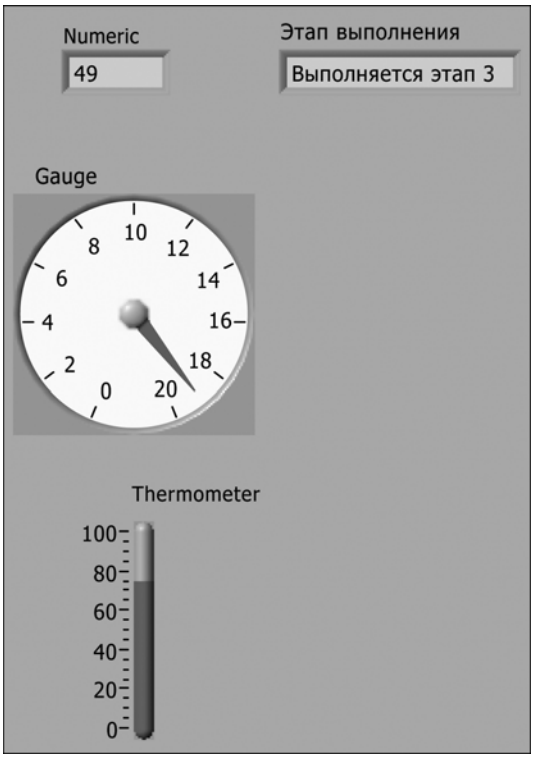


а)

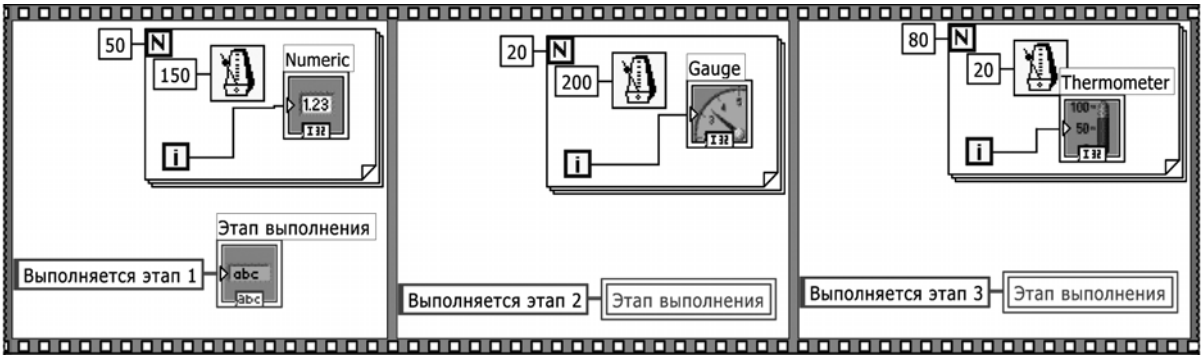
б)

Рис. 13.2. Приклад небажаного випадку створення умови змагання
(а – передня панель, б – блок-діаграма)

Приклад 13.3. На рис. 13.3 зображено результати роботи ВП, в якому передається певне значення у строковий індикатор за допомогою локальної змінної.



а)

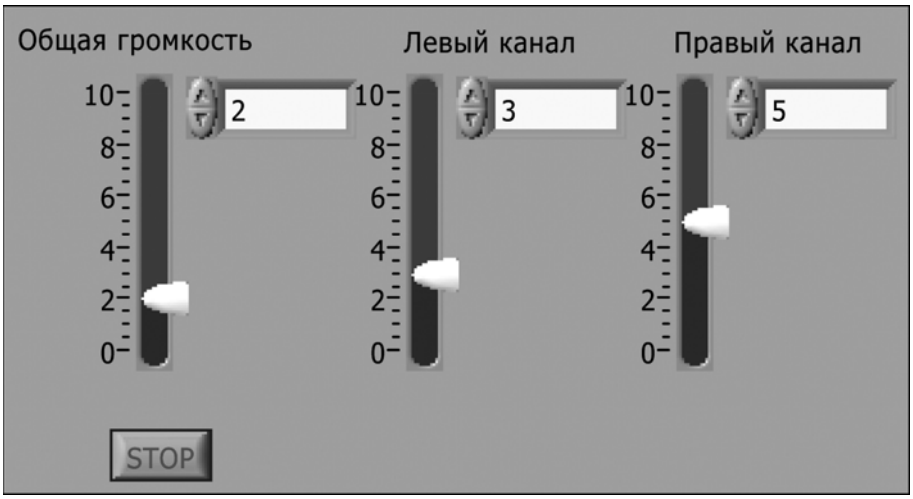


б)

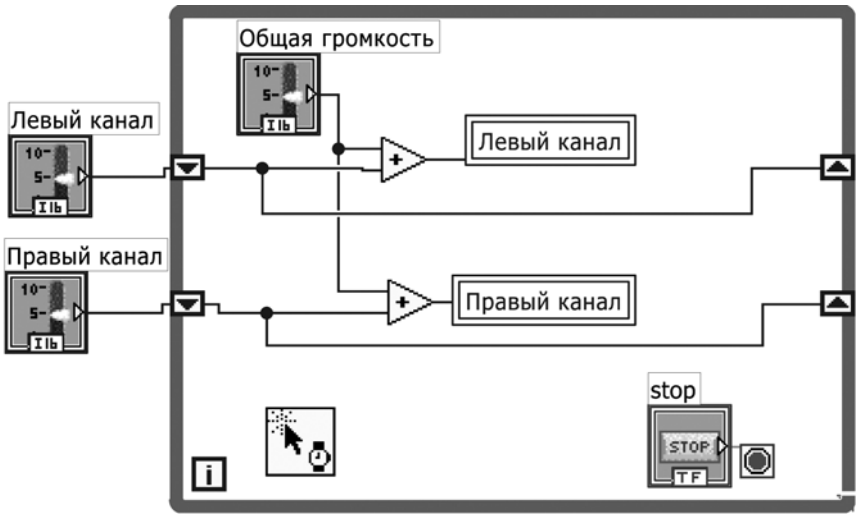
Рис. 13.3. Приклад реалізації локальної змінної (а – передня панель, б – блок-діаграма)

Приклад 13.4. На рис. 13.4 зображено результати роботи ВП: у стереосистемі є регулятори гучності лівого та правого каналів і регулятор загальної гучності. Потрібно, аби після установки балансу по каналах і включення приладу, регулятор загальної гучності піднімав або опускав загальну гучність без зміни балансу, тобто аби обидві гучності

змінювалися на однакову величину.



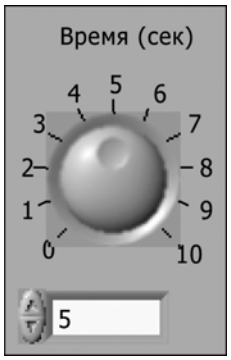
а)



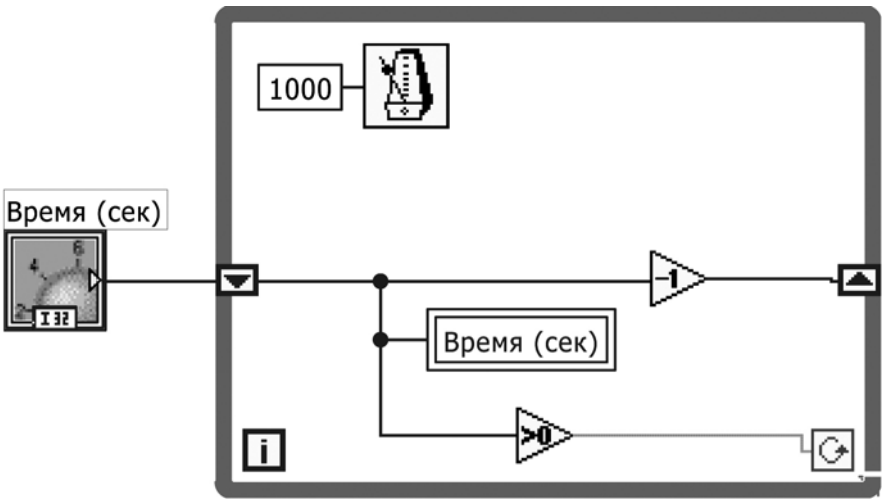
б)

Рис. 13.4. Приклад використання локальних змінних для керування гучністю віртуальної стереосистеми (а – передня панель, б – блок-діаграма)

Приклад 13.5. На рис. 13.5 зображено передню панель та блок-діаграму ВП, в якому програмується таймер зі зворотнім ходом.



а)



б)

Рис. 13.5. Використання локальної змінної для управління таймером

13.4. Завдання на самостійну роботу

Розробити ВП, в якому реалізовано таймер із можливістю зворотного ходу, а також після повернення до нульового значення таймера «ручка» повинна не зупинятися, а знову обертатися у бік збільшення значень до встановленого початкового, потім знову до нуля і так далі.

Варіант реалізації приведено на рис. 13.6.

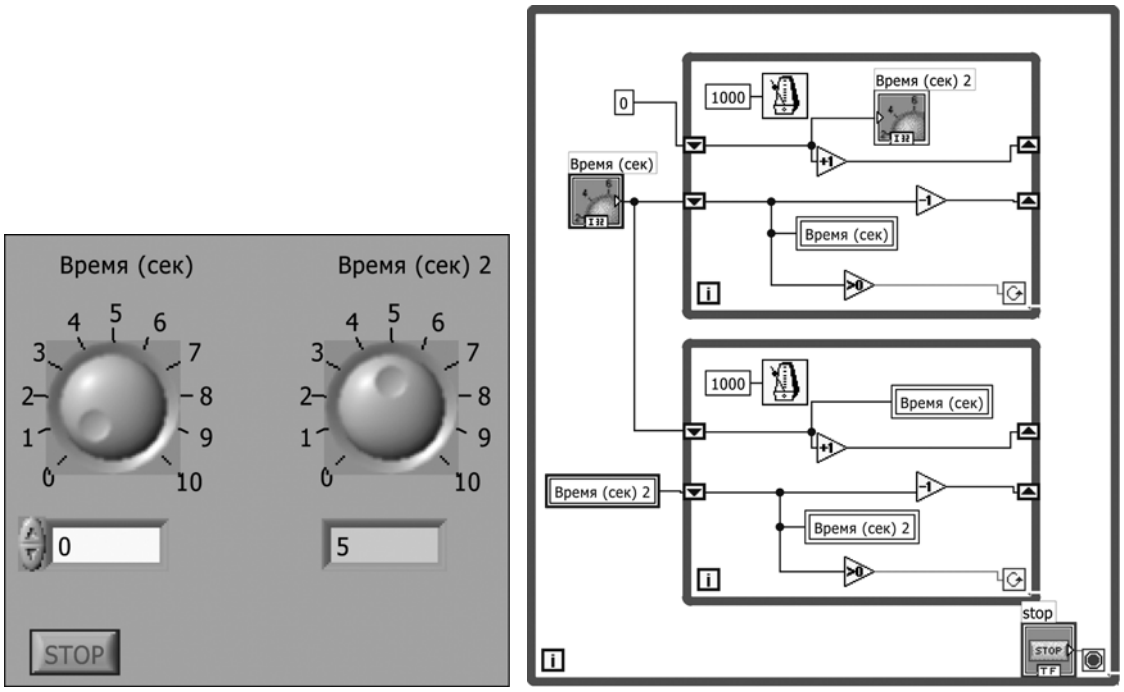


Рис. 13.6. ВП управління таймером

13.5. Контрольні питання

1. Для яких задач є доцільним використання локальних змінних?
2. Які обмеження існують при роботі з локальними змінними логічних елементів?
3. Яка властивість повинна бути встановленою для логічних елементів, коли для них створюються та/або підключаються локальні змінні?

14. ГЛОБАЛЬНІ ЗМІННІ

Глобальні змінні (global variables, або globals) подібні локальним, проте зона їх дії не обмежується одним ВП, тобто глобальні змінні можуть переносити дані між декількома ВП.

Локальні і глобальні змінні в LabVIEW з технічної точки зору є структурами.

14.1. Мета та завдання

Метою заняття є вивчити тип даних глобальні змінні. Для цього було поставлено наступні задачі:

- засвоїти принципи роботи з глобальними змінними;
- зрозуміти принципи ініціалізації змінних;
- визначити задачі, для яких використання глобальних змінних є доцільним, враховуючи що використання глобальних змінних порушує основний принцип LabVIEW – управління потоком даних;
- розглянути режими (mode) запису (WRITE) і читання (READ) глобальних змінних.

14.2. Основи роботи з глобальними змінними

Застосування глобальних змінних у LabVIEW поряд із структурою послідовності (Sequence structure) відноситься до «поганого» стилю програмування, і тому їх застосування рекомендується лише у тому випадку, якщо це є необхідним.

Глобальні змінні застосовують при необхідності передавати дані

між декількома віртуальними приладами, які запускаються одночасно, або коли іконки їх ВПП не можуть бути з‘єднані провідниками на одній блок-діаграмі.

Глобальні змінні, як і локальні, є структурою LabVIEW, доступ до якої здійснюється з підпалітри Структури. Подібно до локальних змінних, кожен термінал глобальної змінної може бути встановлений або в режим читання, або в режим запису.

На відміну від локальних змінних, різні ВП можуть незалежно викликати одну і ту ж глобальну змінну. Глобальні змінні є ефективним способом обміну даними між декількома віртуальними приладами без необхідності підключення провідників даних від одного ВП до іншого – глобальні змінні зберігають інформацію незалежно від окремих віртуальних приладів. Якщо один ВП записує значення в глобальну змінну, то будь-який ВП або ВПП, що здійснює читання з цієї глобальної змінної, може відображати записане значення.

Коли структура глобальної змінної вибрана з палітри, на блок-діаграмі з'являється піктограма, що відрізняється від піктограми локальної змінної наявністю символу земної кулі. Вона символізує глобальну змінну, яка ще не визначена. Двічі клацнувши лівою кнопкою миші по її іконці, можна викликати вікно, ідентичне передній панелі віртуального приладу. Можна сказати, що глобальні змінні це особливий вид віртуального приладу. Вони можуть містити будь-які типи і будь-яку кількість структур даних на передній панелі, але не мають відповідної блок-діаграми. Глобальні змінні зберігають дані, але самостійно не проводять з ними яких-небудь операцій. Розміщення елементів управління або індикаторів на передній панелі глобальної змінної здійснюється так, як і для звичайного віртуального приладу. Цікавою особливістю глобальних змінних є відсутність принципової різниці у виборі елементу управління

або індикатора для представлення певного типу даних, оскільки можна як зчитувати з глобальних змінних дані, так і записувати (передавати) дані в них [1, 551-558].

14.3. Приклади

Приклад 14.1. На рис. 14.1, 14.2, 14.3 зображено результати роботи ВП, що вирішує наступну задачу: припустимо, що є два ВП, які виконуються одночасно. Кожен ВП генерує поелементно дані та відображує на індикаторах-осцилографах. Один з віртуальних приладів має логічну кнопку Stop для припинення виконання обох віртуальних приладів. Для зупинки роботи двох різних програм за допомогою однієї кнопки доцільним є використання глобальних змінних.

На рис. 14.1 (а) зображено результати роботи ВП прикладу, в якому генерується синусоїда. На рис. 14.2 (а) зображено результати роботи другої програми, що генерує пилоподібну криву. На рис. 14.3 зображена глобальна змінна stop_glb.vi, яка фактично є окремим віртуальним приладом, що не містить блок-діаграму.

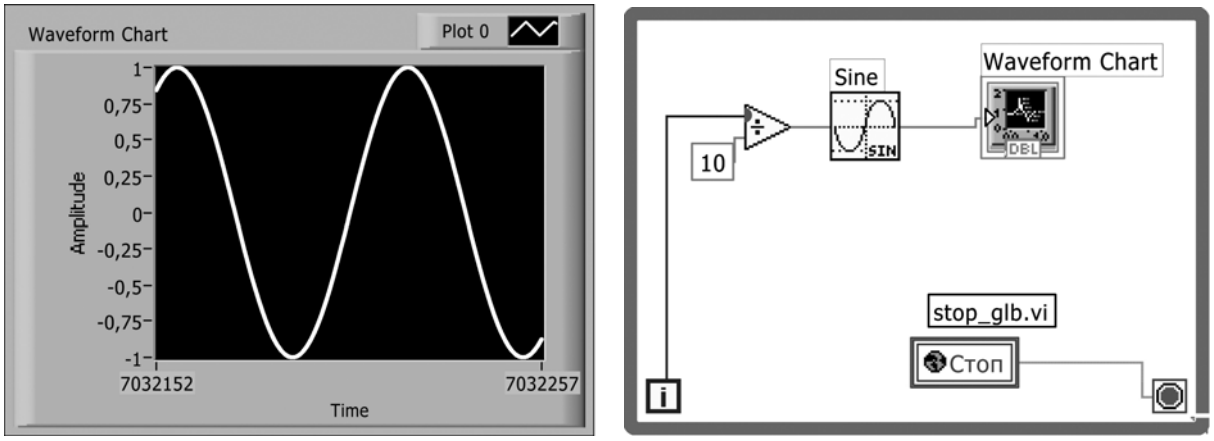
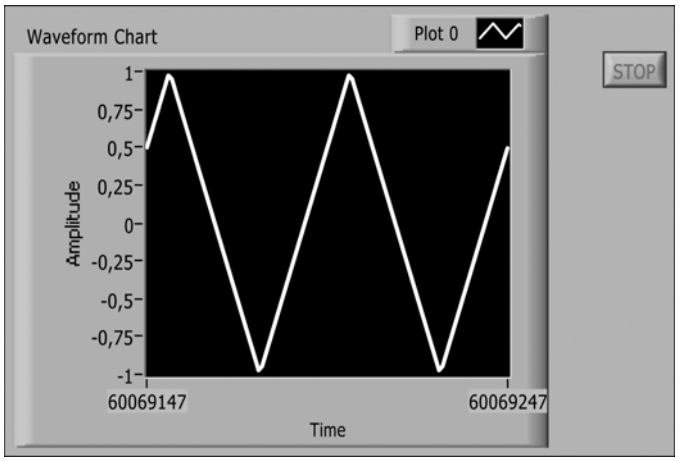
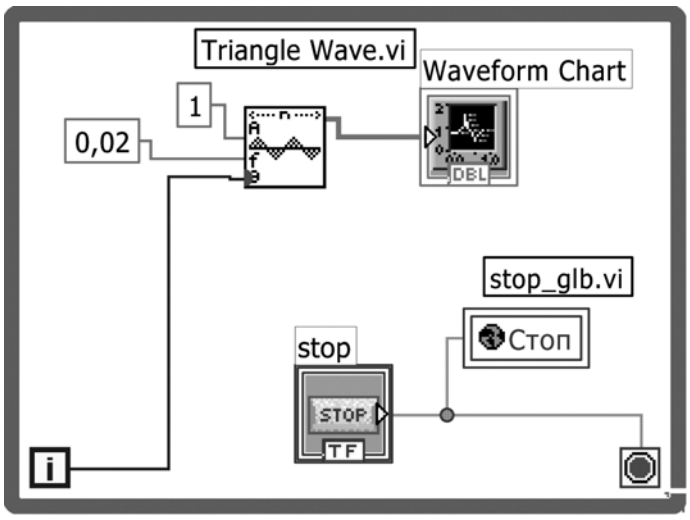


Рис. 14.1 (а, б). ВП, в якому генерується синусоїда

(а – передня панель, б – блок-діаграма)



а)



б)

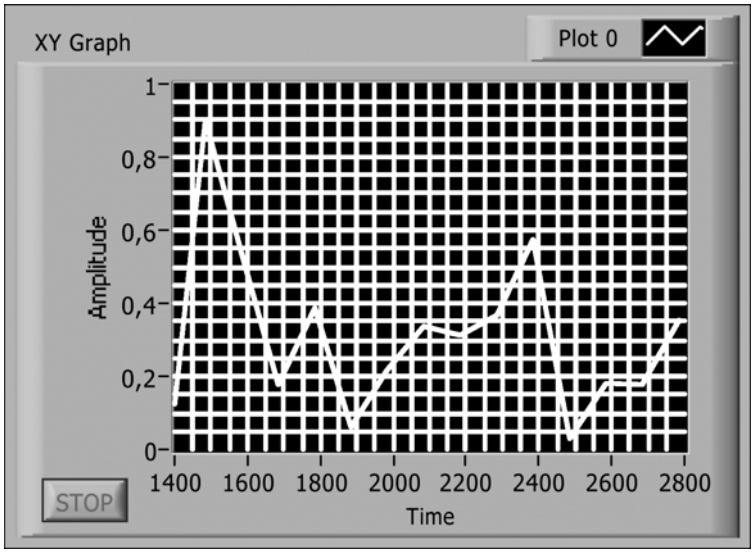
Рис. 14.2 (а, б). ВП, що генерує пилоподібну криву
(а – передня панель, б – блок-діаграма)

Рис. 14.3. Глобальна змінна (stop_glb.vi)

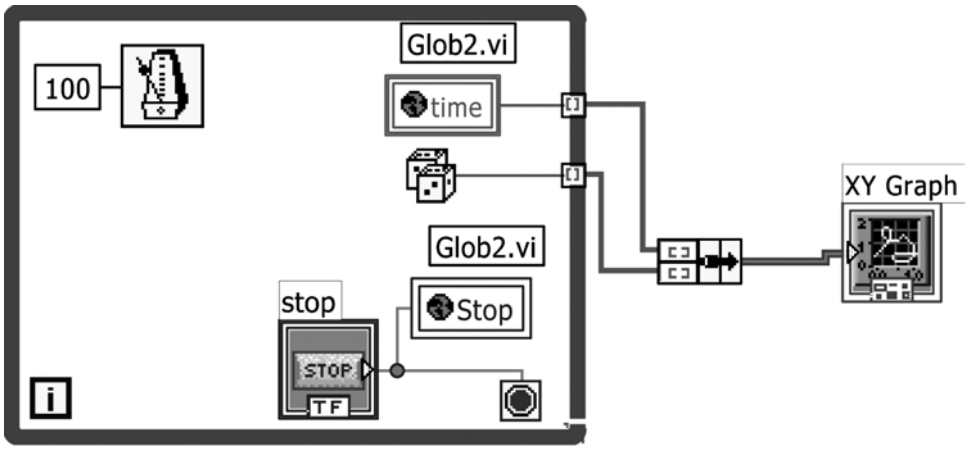
Приклад 14.2. На рис. 14.4 – 14.6 показано приклад використання двох глобальних змінних для виконання двох функцій: зупинка двох програм однією кнопкою Stop та передача в перший прилад з другого величини часу, що пройшов від моменту запуску програми.

Для цього використовується дві глобальні змінні: перша – логічного типу з ім'ям Stop для зупинки приладів, друга – чисельного типу з ім'ям time для передачі часу. Обидві глобальні змінні знаходяться в одному глобальному приладі Glob2.vi, зображеному на рис. 14.6, оскільки глобальні змінні можуть зберігати декілька даних різних типів, то краще групувати дані в одну глобальну змінну, а не створювати декілька глобальних змінних.

У структурі послідовності на рис. 14.5 в першому «кадрі» при включенні приладу фіксується початковий час, а далі в другому «кадрі» визначається за допомогою операції віднімання час, що пройшов від початкового, і передається через глобальну змінну time в перший віртуальний прилад (рис. 14.4), де відображується на графіку.



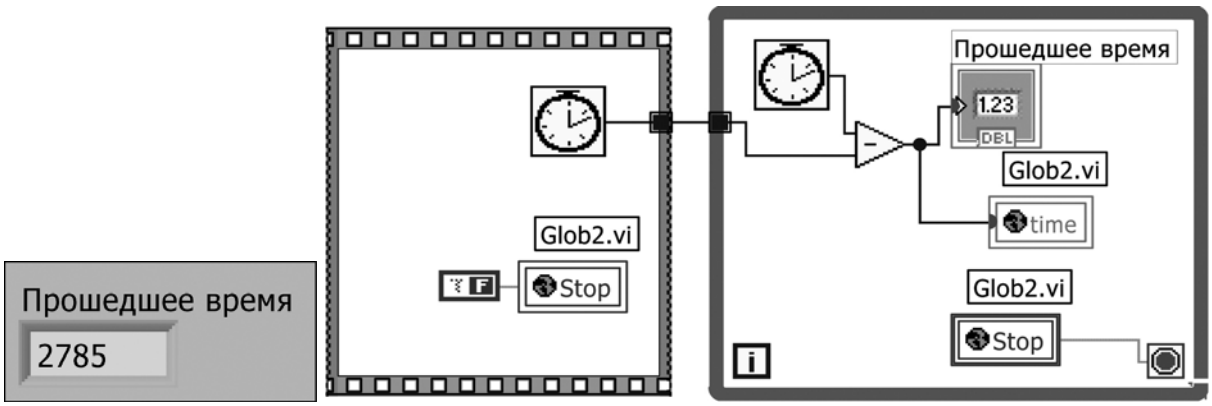
а)



б)

Рис. 14.4 (а, б). Приклад використання двох глобальних змінних

(а – передня панель, б – блок-діаграма)



а) б)

Рис. 14.5 (а ,б). Приклад використання двох глобальних змінних

(а – передня панель, б – блок-діаграма)

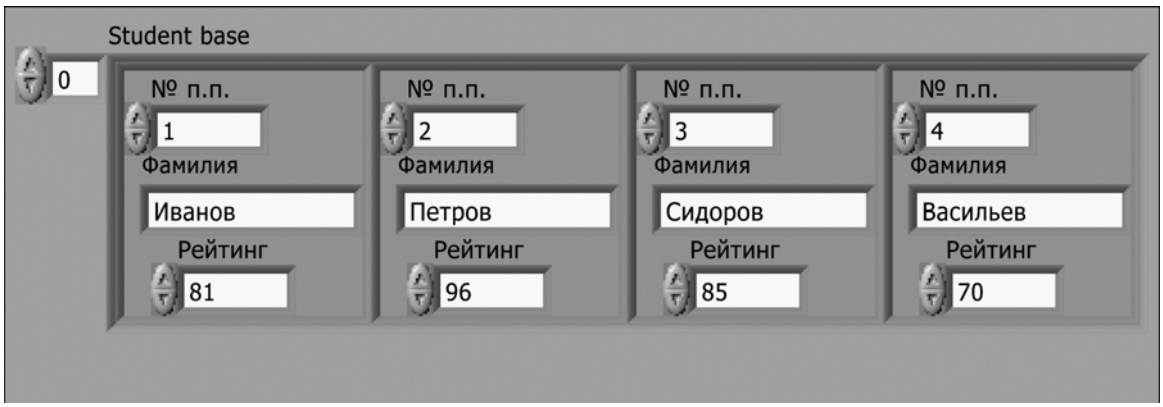
Рис. 14.6. Глобальна змінна Glob2.vi

14.4. Завдання на самостійну роботу

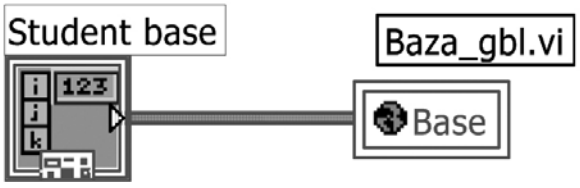
Розробити програмний продукт, який являє собою комплекс з двох віртуальних приладів і однієї глобальної змінної, через яку передаються дані між приладами.

У першому ВП деякий працівник архіву веде базу даних студентів у вигляді кластерів, що містять поля: порядковий номер, прізвище, рейтинг. У другому приладі деякий працівник деканату одержує доступ до цієї бази даних (через глобальну змінну), знаходить найуспішнішого студента і виводить на екран його дані у вигляді кластеру.

Варіант реалізації приведено на рис. 14.7 – 14.9. На рис. 14.7 і рис. 14.8 зображено перший і другий віртуальні прилади, а на рис. 14.9 – глобальна змінна Baza_gbl.vi.

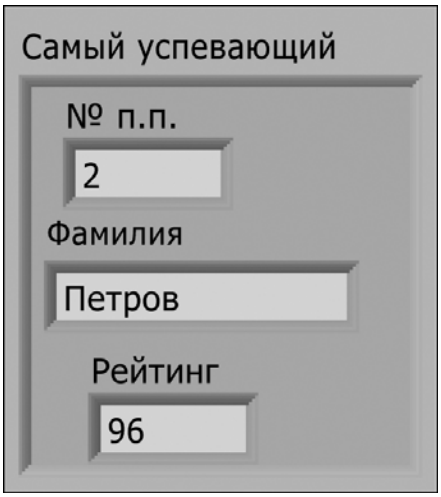


а)

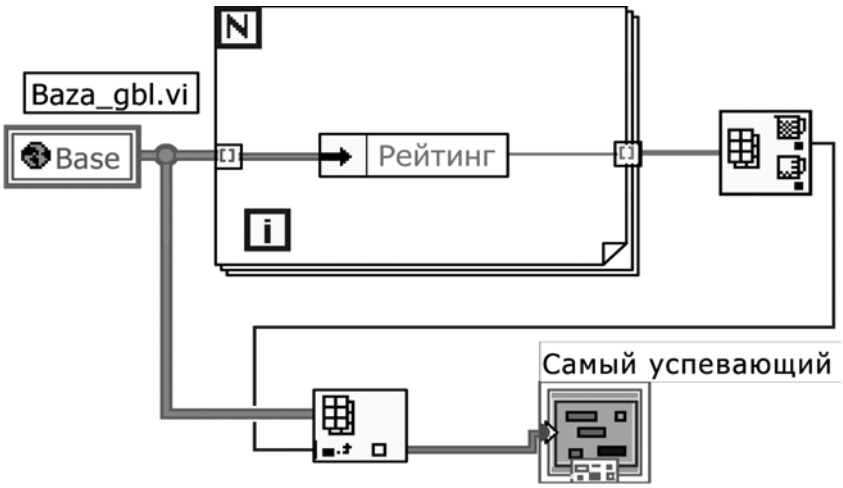


б)

Рис. 14.7 (а, б). База даних студентів у вигляді кластерів
(а – передня панель, б – блок-діаграма)



а)



б)

Рис. 14.8 (а, б). ВП для доступу до бази даних через глобальну змінну (а – передня панель, б – блок-діаграма)

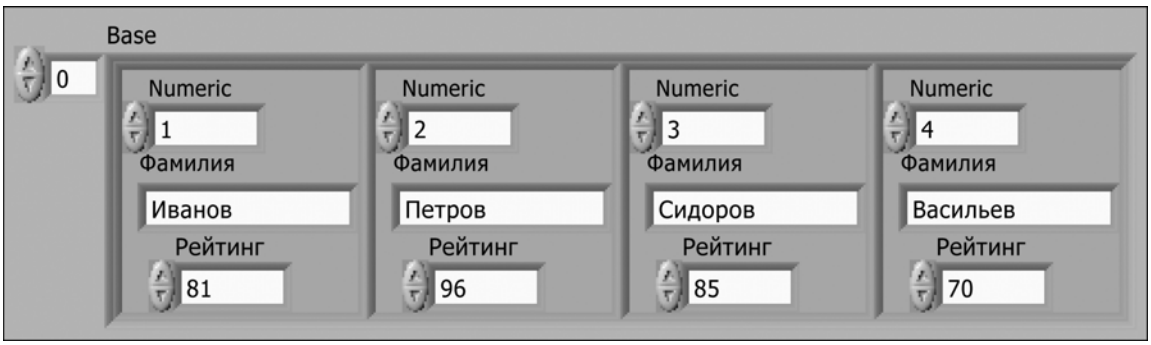


Рис. 14.9. Глобальна змінна Baza_gbl.vi

14.5. Контрольні питання

1. Чому глобальними змінними слід користуватися лише в тих випадках, коли без них обійтися неможливо?
2. Чому глобальні змінні слід ініціалізувати?

15. ВУЗЛИ ВЛАСТИВОСТЕЙ

За допомогою вузлів властивостей (**Property Nodes**) можна створювати потужніші програми з більш дружнім інтерфейсом користувача. Вузли властивостей дозволяють програмно управляти властивостями об'єктів передньої панелі, такими як колір, видимість на передній панелі, місце розташування, формат представлення чисел і так далі.

15.1. Мета та завдання

Метою даного заняття є ознайомлення з типом даних «вузли властивостей». Для досягнення поставленої мети були сформульовані наступні задачі:

- розібрати переваги використання вузлів властивостей;
- засвоїти принципи роботи з вузлами властивостей.

15.2. Основи роботи з вузлами властивостей

Для того, щоб створити вузол властивостей, потрібно натиснути праву кнопку миші на об'єкті передньої панелі або його терміналі на блок-діаграмі та вибрати опцію Створити -> Вузол властивостей (**Create -> Property Node**). На блок-діаграмі з'явиться термінал з тим же ім'ям, що і об'єкт передньої панелі.

Аналогічно локальним змінним вузли властивостей можуть зчитувати або записувати властивість об'єкту (хоча деякі властивості призначені лише для читання). Аби змінити режим властивості, слід викликати контекстне меню і вибрати опцію **Змінити на запис/читання (Change to Write/Read).** Маленька стрілка усередині терміналу вузла

властивостей вкаже, який режим використовується в даний момент.

Приведемо деякі базові властивостей об‘єктів:

• **Видимість (Visible).** Встановлює або прочитує статус видимості об'єкту на передній панелі. Об'єкт відображатиметься при установці властивості Видимість в стан ІСТИНА; в стані ХИБНІСТЬ він буде прихований. Управління видимістю об'єкту – краще рішення, ніж його «прозоре» зафарбовування, оскільки до прозорого об'єкту можливий випадковий доступ;

• **Заборона (Disabled).** Задає або прочитує статус прав доступу користувача до елементу управління. Нульове значення опції надає доступ користувача до елементу управління; значення 1 забороняє доступ без якої-небудь видимої індикації; значення 2 забороняє доступ до елементу управління і відмічає неактивність елементу сірим кольором;

• **Положення (Position).** Відображається кластером з двох чисел, які визначають координати верхнього лівого краю об'єкту на передній панелі;

• **Розмір (Size).** Відображається кластером з двох чисел, які визначають висоту і ширину зображення даного об'єкту на передній панелі;

• **Блимання (Blinking).** При установці в стан ІСТИНА зображення об'єкту на передній панелі стає блимаючим;

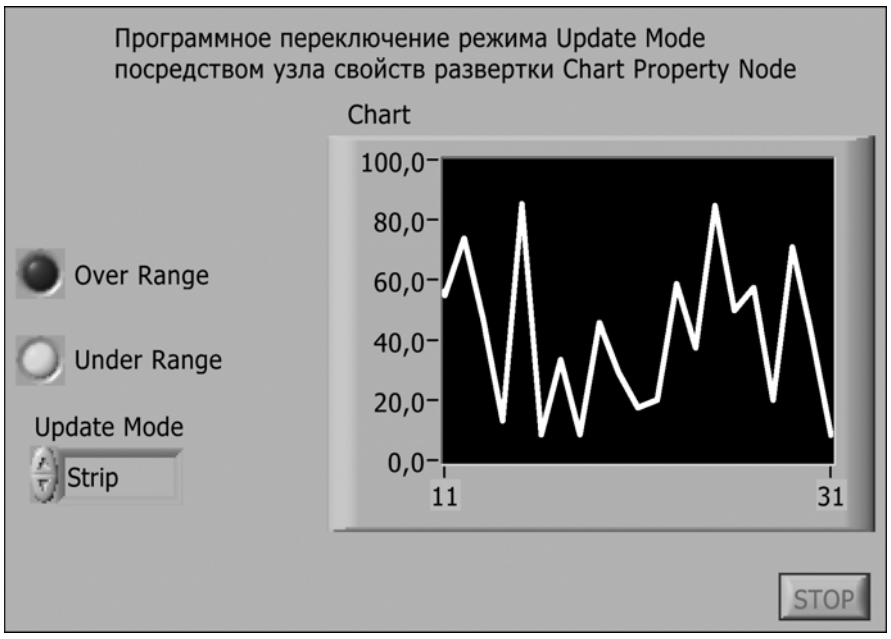
• **Формат і точність (Format and Precision).** Встановлює або прочитує параметри формату і точності відповідних числових елементів управління і відображення. Вхідний кластер містить два цілих числа: одне для формату, інше для точності;

• **Колір (Color).** Залежно від типу об'єкту властивість Колір може мати декілька варіантів. На вхід властивості допустимо підключити масив полів установки кольорів тексту, фону, атрибутів і так далі залежно від типу об'єкта.

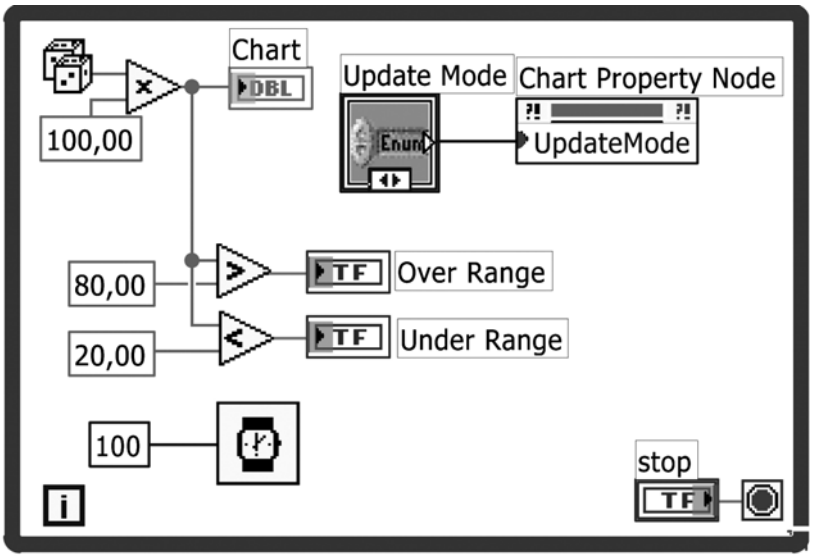
Майже у всіх елементів управління і індикації є базові властивості. Більшість елементів мають значний набір спеціальних властивостей, особливо таблиці і графіки (більше 100 властивостей). Отримати про них детальну інформацію легко у вбудованій довідковій системі LabVIEW [1, 559-564].

15.3. Приклади

Приклад 15.1. На рис. 15.1 зображено приклад, який показує можливість програмного управління одним з багатьох режимів відображення інформації на графіку. ВП ілюструє можливість програмної установки одного з трьох способів оновлення графіків: Strip, Scope і Sweep.



а)

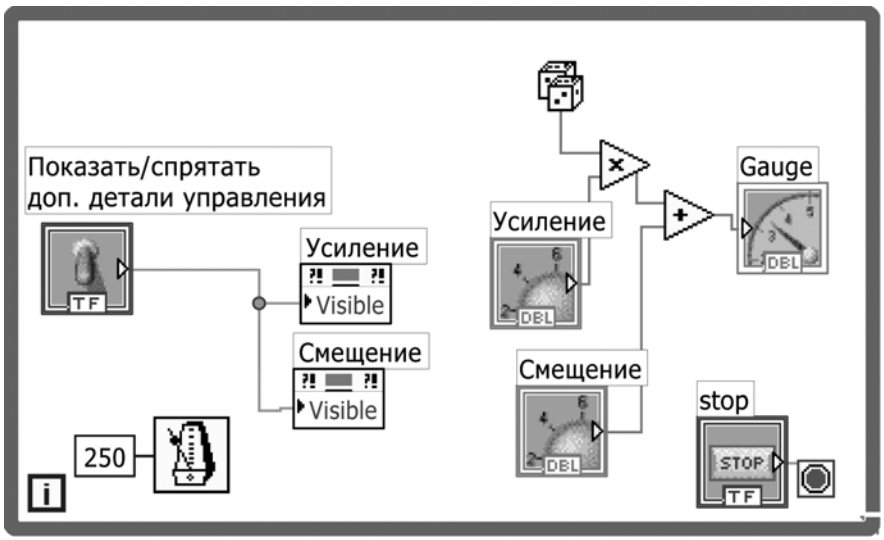


б)

Рис. 15.1 (а, б). Приклад програмного управління режимом відображення інформації на графіку (а – передня панель, б – блок-діаграма)

Приклад 15.2. На рис. 15.2 показано реалізацію наступного алгоритму: припустимо, що потрібно створити передню панель, на якій певні спеціалізовані елементи управління є «захованими» до моменту, коли вони будуть викликані. На передній панелі, зображеній на рис. 15.2, приведено індикатор значення деякої величини (у вигляді циферблату), також є тумблер “Показать/спрятать доп. детали управления”, який управляє появою регулювальників додаткових параметрів.

а)

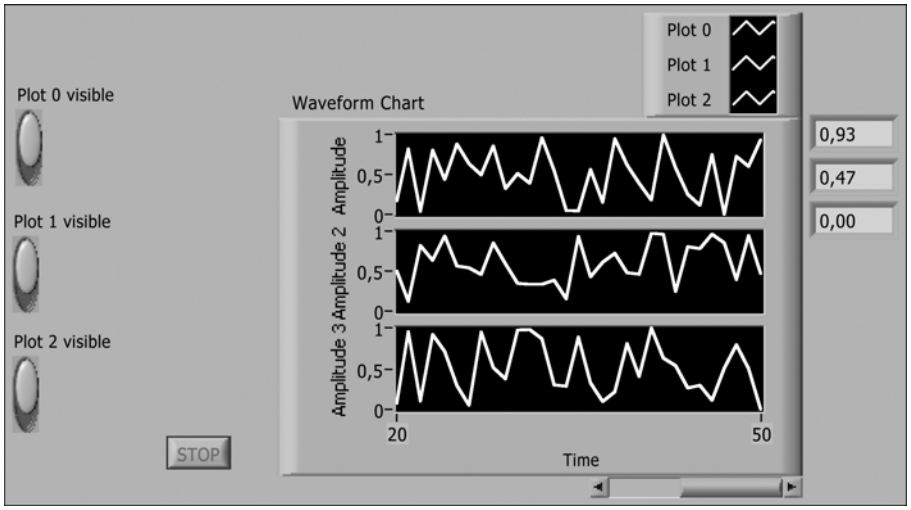


б)

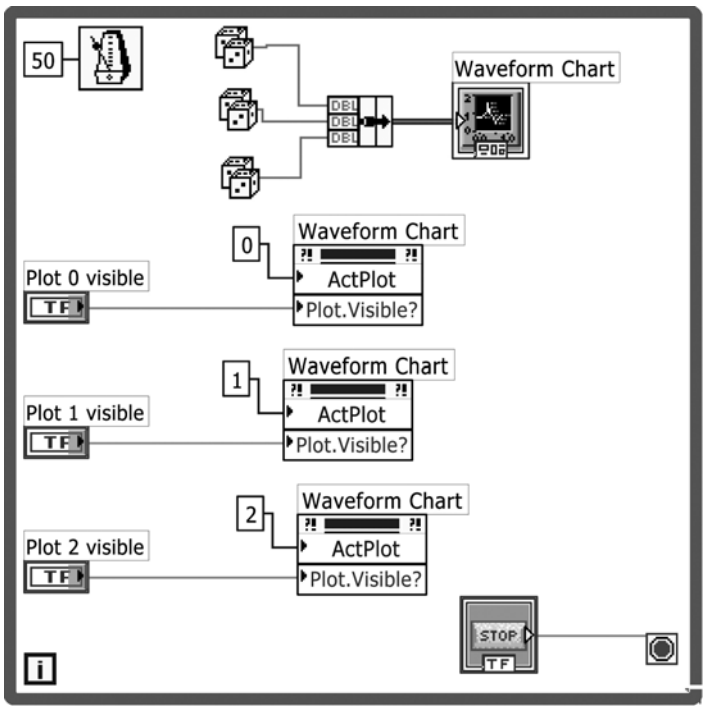
Рис. 15.2 (а, б). Приклад застосування елементів управління, що «заховані» до моменту, поки вони не будуть викликані (а – передня панель, б – блок-діаграма)

Приклад 15.3. На рис. 15.3 показано віртуальний прилад, який відображує три канали даних (експериментальних даних, введених в реальному часі з плати вводу/виводу, або, як в даному прикладі, просто випадкових чисел). У ВП реалізовано можливість «вмикання» або «вимикання» будь-якого з графіків.

Криві графіків нумеруються як 0, 1, … n. Особлива властивість, що має назву Активна крива (**Active Plot**), призначена для вибору кривої, властивості якої будуть змінюватися або зчитуватися.



а)

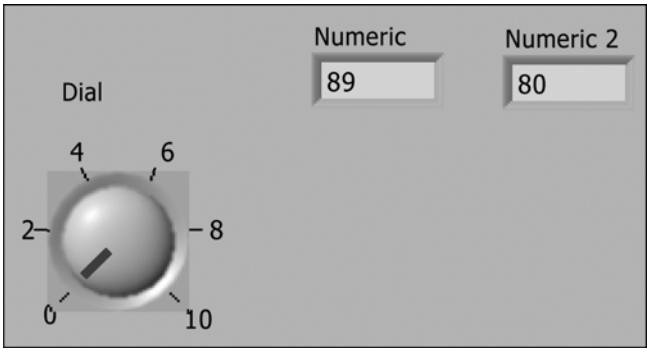


б)

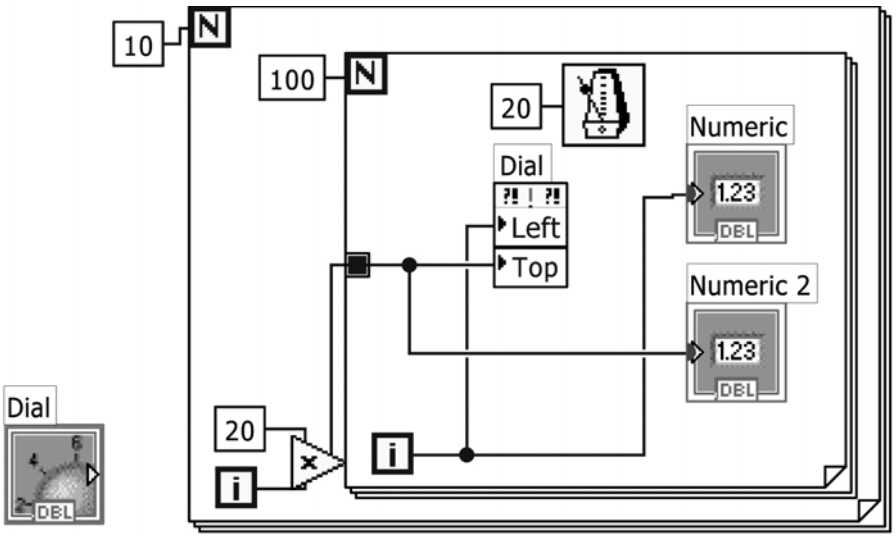
Рис. 15.3 (а, б). ВП з можливістю «вмикати» або «вимикати» будь-який з графіків (а – передня панель, б – блок-діаграма)

Приклад 15.4. На рис. 15.4 показано ВП, що виконує декілька жартівливу і незвичайну функцію, яка навряд чи знадобиться на практиці, але дуже показова в плані демонстрації можливостей вузлів властивостей.

При включенні приладу ручка управління Dial починає рухатися по передній панелі зліва направо і зверху вниз. При цьому на числових індикаторах відображуються її координати у пікселях.



а)



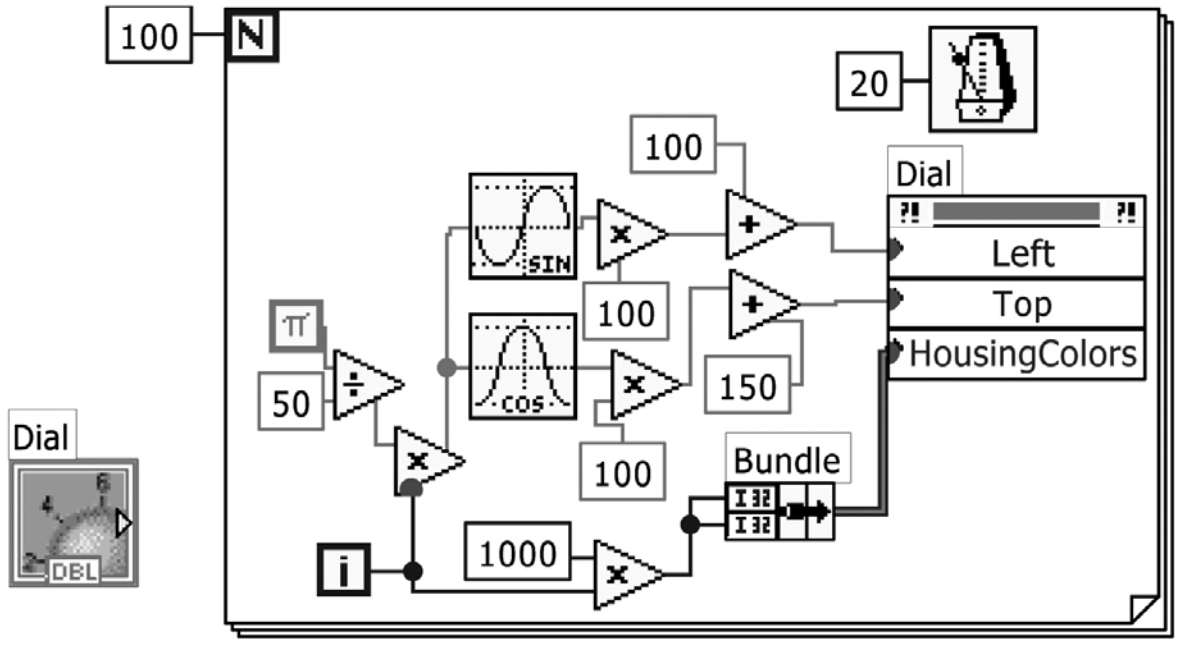
б)

Рис. 15.4 (а, б). ВП, в якому ручка управління Dial рухається по передній панелі зліва направо і зверху вниз (а – передня панель, б – блок-діаграма)

15.4. Завдання на самостійну роботу

Розробити ВП, який би модифікував завдання з прикладу 15.4 так, щоб ручка управління Dial при включенні ВП рухалася по колу і міняла колір. Варіант реалізації приведено на рис. 15.5.

а)



б)

Рис. 15.5. ВП, в якому ручка управління Dial рухається по колу і міняє колір (а – передня панель, б – блок-діаграма)

15.5. Контрольні питання

1. Які задачі вирішуються за допомогою вузлів властивостей?
2. Як можна одержати доступ до декількох властивостей одного елементу?
3. Яким чином обирається на багатопроменевих графіках крива, до якої має відношення зміна властивостей?

16. СТРУКТУРА ПОДІЙ

Структура Подій (**Event Structure**), що розташована в підпалітрі Structures, є надзвичайно потужним інструментом. Ця структура дозволяє створювати високоефективний код, який чекає на подію, на відміну від неефективного коду, що періодично перевіряє, чи не сталася подія.

16.1. Мета та завдання

Метою заняття є навчитися працювати із структурою Подій. Для цього необхідно вирішити заступні завдання:

- розглянути можливості використання структури Подій;
- зрозуміти принципи роботи із структурою Подій.

16.2. Основи роботи із структурою Подій

Подією у структурі подій може бути практично будь-яка подія, наприклад, натиснення клавіші мищі на передній панелі, зміна значення чисельної величини, курсор миші з'явився у вікні ВП і тому подібне. Події зазвичай пов'язані із зміною значень елементів управління і з елементами графічного інтерфейсу користувача.

Без використання структури Подій для того, щоб взнати, чи натиснув користувач кнопку Стоп, необхідно періодично опитувати даний елемент в циклі While. З використанням структури Подій дана програма "знає" коли подія відбулась.

Структура Подій складається з «варіантів». Кожен "варіант" структури Подій може бути зареєстрованим для обробки однієї або більшої кількості подій.

Коли структура Подій з'являється на блок-діаграмі ВП, вона за умовчанням конфігурована для обробки події "закінчення часу чекання". Подія "закінчення часу чекання" – це спеціальна подія, яка почне виконуватися, якщо протягом встановленого часу не станеться інших подій, що описуються структурою. Значення часу чекання подається на термінал у верхньому лівому кутку структури (позначений символом пісочного годинника). Значення за умовчанням (нічого не подається на термінал) рівне -1, що означає "ніколи" або "чекати нескінченно довго".

Якщо встановити значення часу чекання -1 і жодних інших подій ніколи не станеться, то ВП виявиться таким, що "не відповідає" (або таким, що «завис»). Ця ситуація схожа на нескінченний цикл, який так само "не відповідає".

16.2.1. Налаштування подій

Для того, щоб зареєструвати структуру Подій на обробку подій елементів управління передньої панелі, необхідно викликати контекстне меню на рамці структури Подій і вибрати опцію **Add Event Case**. З'явиться діалогове вікно налаштування подій.

Це вікно має наступні компоненти:

- Event Handler for Case – містить номер і назву всіх "варіантів" даної структури Подій;
- пункт Event Specifires – відображує повний список подій і їх джерело (програмний компонент, віртуальний прилад, елемент управління) для даного елементу;
- кнопка Insert Event – використовується для того, щоб створити подію, підтримувану даним "варіантом";
- кнопка Delete Event – використовується для видалення події,

підтримуваної даним "варіантом";

- дерево параметрів Event Sources – відображує всі джерела подій, відсортовані по класах;
- пункт Events – відображує список доступних подій для джерела, вибраного в секції Event Sources.

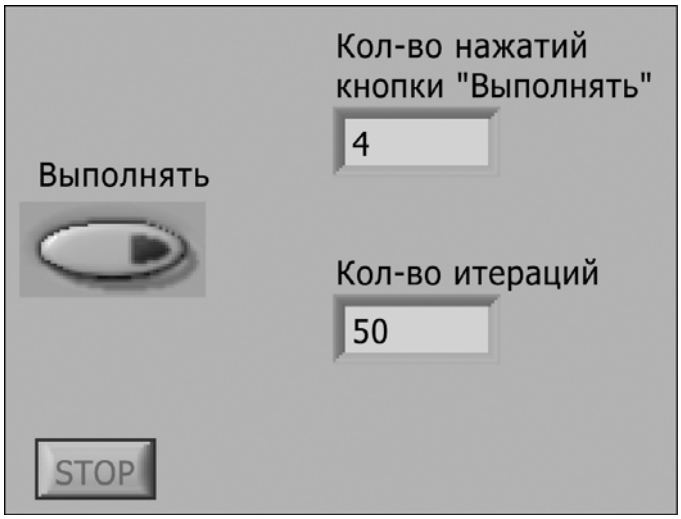
З контекстного меню структури Подій можна вибрати опцію **Edit Events Handled by this Case**, щоб модифікувати існуючий "варіант"; **Duplicate Event Case** – щоб створити копію "варіанту"; **Delete Event Case** – щоб видалити "варіант" [1, 570-577].

16.3. Приклади

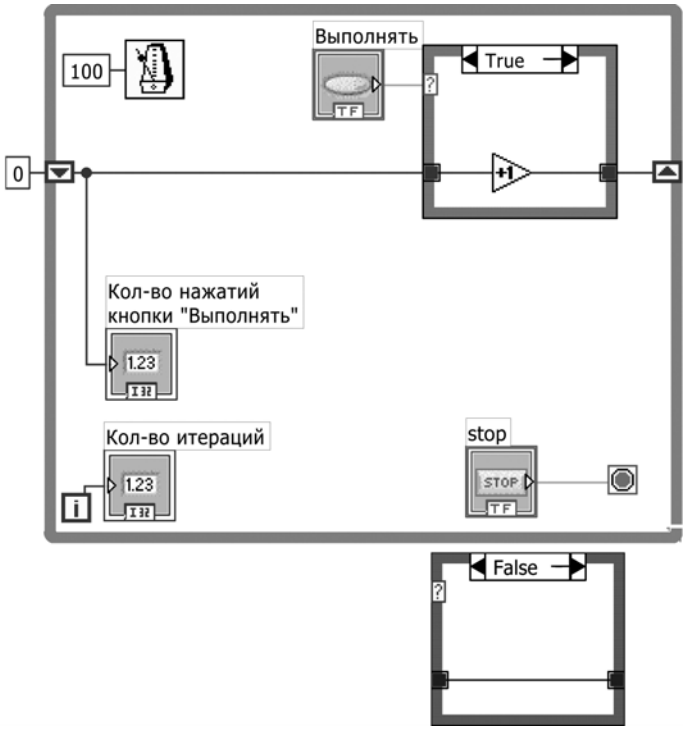
Приклад 16.1. На рис. 16.1 і рис. 16.2 зображено рішення однотипної задачі без використання і з використанням структури Подій.

Завдання полягає у підрахунку кількості натискань користувачем кнопки "Выполнять". У приладі на рис. 16.1 це реалізовано шляхом опитування в циклі While стану кнопки: якщо кнопка натиснута, то до значення в зсувному регістрі додається одиниця. На передній панелі ВП видно, що при 4-х натисканнях кнопки пройшло 50 ітерацій циклу, і це з урахуванням того, що в циклі встановлена затримка часу 100 мс. Без такої затримки пройшли б тисячі ітерацій, тобто комп'ютер велику частину робочого часу витрачав би на опитування кнопки.

На рис. 16.2 показана структура Подій, яка чекає події "Зміна значення кнопки". Як видно на передній панелі, користувач натискав кнопку " Выполнять " 4 рази, і цикл While виконався всього 4 рази – по одному разу на кожне натискання кнопки.

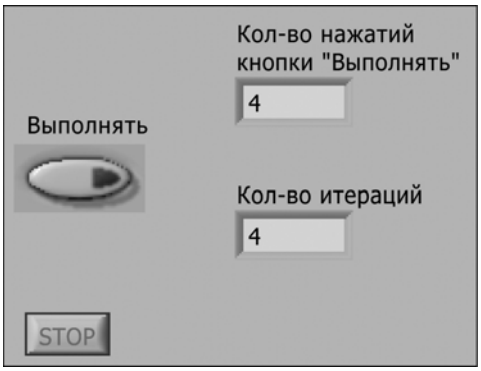


а)

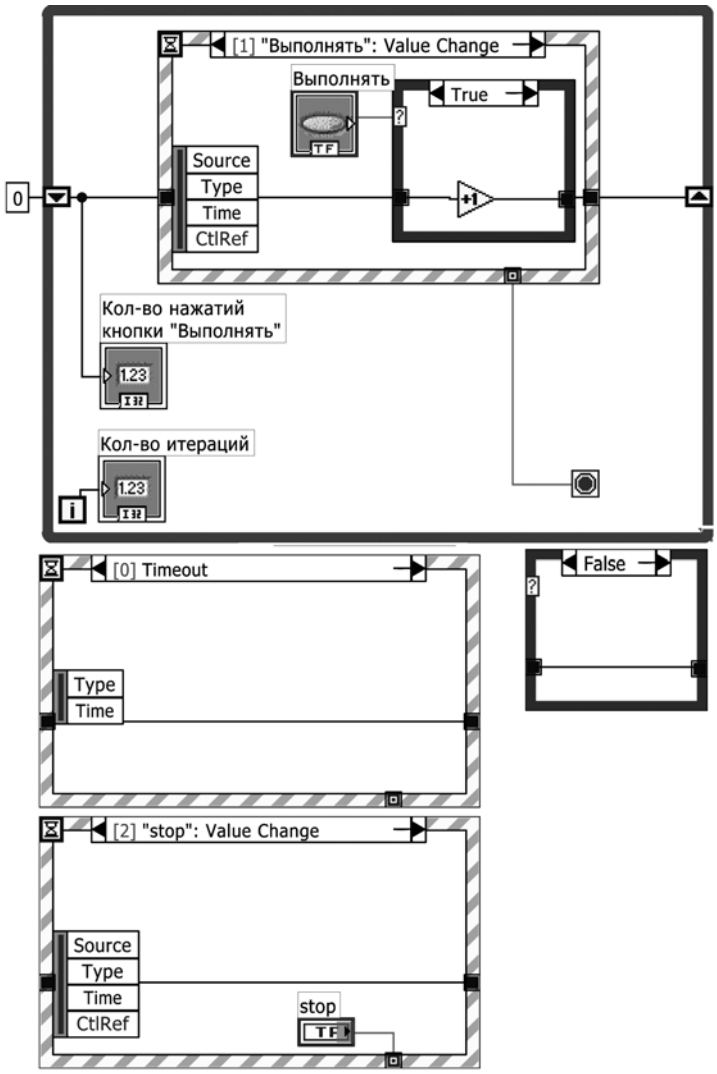


б)

Рис. 16.1 (а, б). ВП для підрахунку кількості натискань користувачем кнопки "Выполнять" шляхом опитування в циклі While
(а – передня панель, б – блок-діаграма)



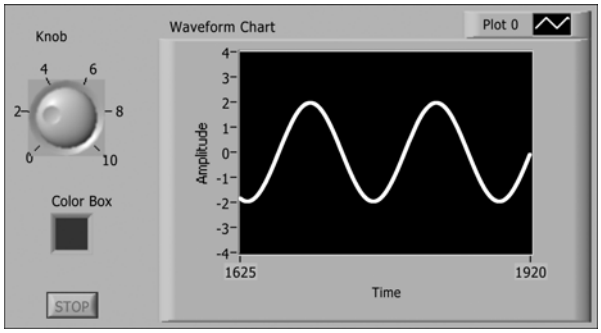
а)



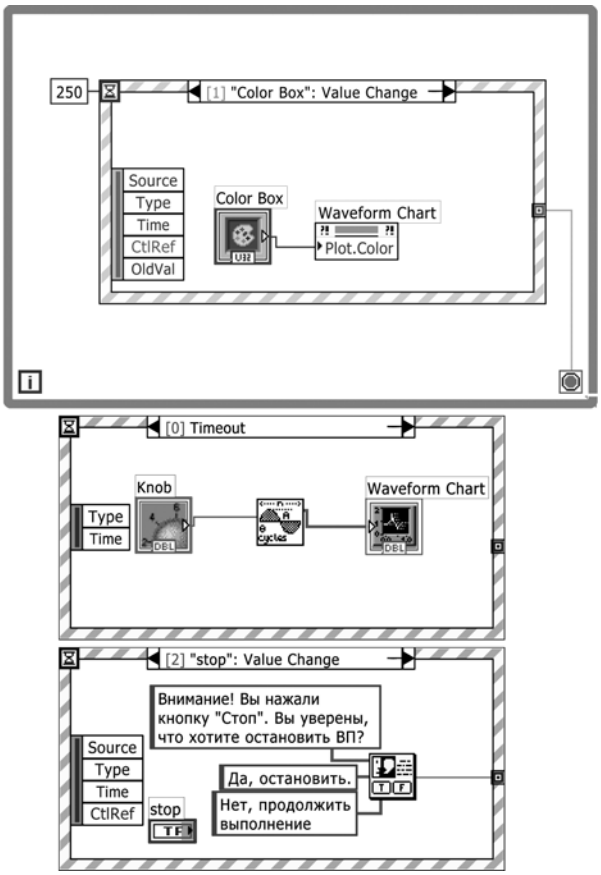
б)

Рис. 16.2 (а, б). ВП для підрахунку кількості натискань користувачем кнопки "Выполнять" з використанням структури Подій
(а – передня панель, б – блок-діаграма)

Приклад 16.2. На рис. 16.3 показано ВП, в якому використано структуру Подій з метою «контролю» циклу, в якому генерується синусоїда. Структура Подій виявляє натискання кнопки Стоп або зміну значення змінної plot color (колір графіка) і виконує відповідну дію.



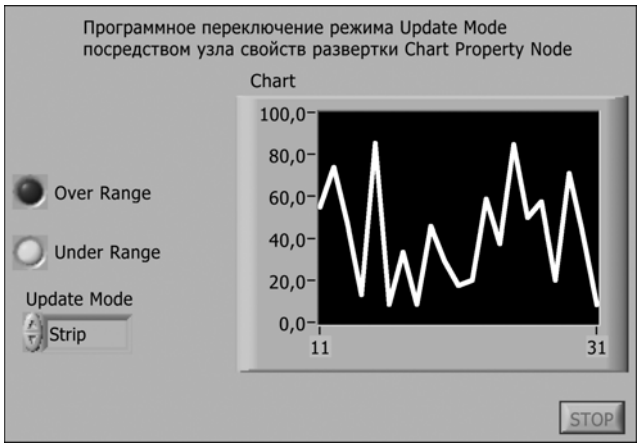
а)



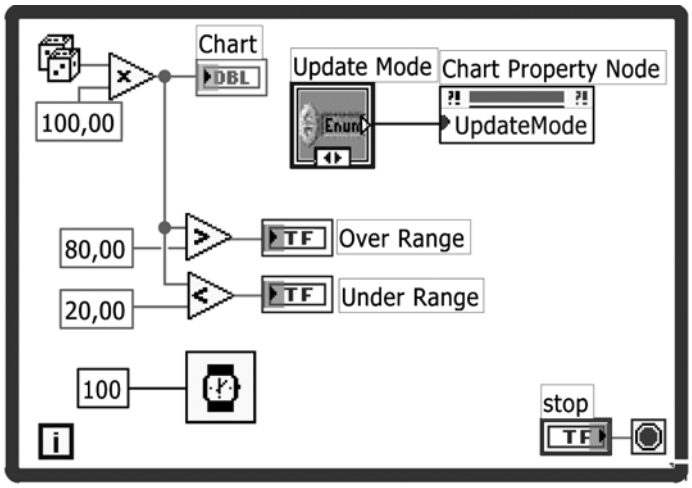
б)

Рис. 16.3 (а, б). Приклад використання структури Подій
(а – передня панель, б – блок-діаграма)

Приклад 16.3. На рис. 16.4 зображено віртуальний прилад, в якому є можливість вибору режиму відображення графіку.



а)



б)

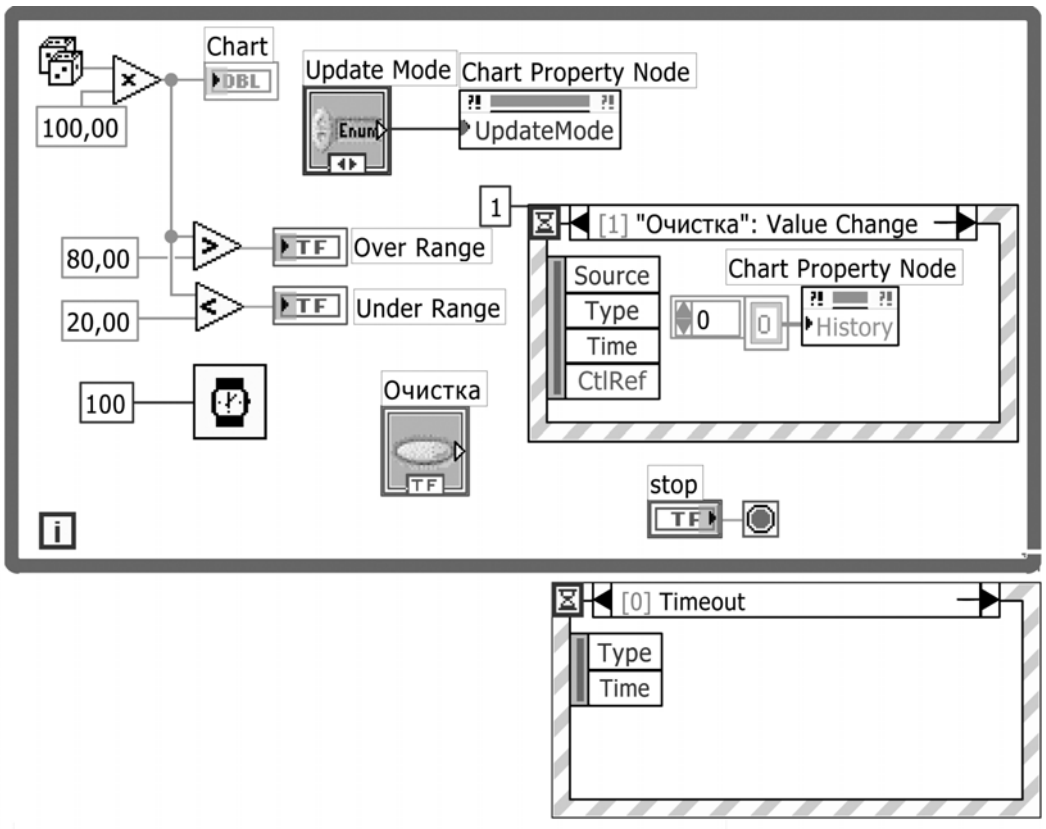
Рис. 16.4. ВП, де можна вибрати режим відображення графіка в процесі роботи (а – передня панель, б – блок-діаграма)

16.4. Завдання на самостійну роботу

Розробити ВП, що є модифікацією ВП з прикладу 16.3 так, щоб окрім управління режимом оновлення графіку (Update Mode) в процесі роботи програми можна було обнулити буфер (History) натисканням кнопки

"Очистка". Варіант реалізації наведений на рис. 16.5.

а)



б)

Рис. 16.5. ВП з можливістю обнулення буферу (History) за допомогою натискання на кнопку "Очистка" (а – передня панель, б – блок-діаграма)

16.5. Контрольні питання

1. Що дозволяє реалізувати структура подій?
2. У чому код з використанням структури Подій ефективніше за код з опитуванням стану управляючого елементу в циклі?

17. ЗБІР ДАНИХ І УПРАВЛІННЯ ПРИЛАДАМИ В LABVIEW

LabVIEW дає можливість користувачам перетворити комп'ютери на універсальні прилади і комплекси, що збирають дані з навколишнього світу. Це є однією з головних причин популярності LabVIEW.

17.1. Мета та завдання

Метою заняття є вивчення принципів збору та передачі даних у системі LabVIEW. Для цього було поставлено наступні завдання:

- зрозуміти принципи вводу/виводу зовнішніх даних в комп'ютер і взаємодії компонентів програмного забезпечення;
- з'ясувати порядок використання віртуальних приладів, необхідних для аналогових і цифрових вимірів;
- розібрати віртуальні прилади палітри «Збір даних».

17.2. Основи роботи з пристроями збору даних

Пристрої, що забезпечують збір даних, називаються DAQ. Сама абревіатура DAQ розшифровується як Data Acquisition і перекладається українською мовою як збір даних.

Всі DAQ пристрої мають драйвери, тобто програми, що написані на низькорівневому коді, та є необхідними для взаємодії елементів збору даних з комп'ютером.

Між NI-DAQ і LabVIEW функціонує зв'язуюча програма, що має назву MAX (Measurement & Automation Explorer – програма аналізу вимірів і автоматизації). MAX є програмним інтерфейсом Windows, який дає можливість доступу до всіх плат NI (а саме плати вводу/виводу, КОП,

VXI і так далі). MAX використовується в основному для конфігурації і тестування апаратної частини. Це необхідно зробити перед тим, як використовувати апаратуру в LabVIEW. MAX інсталюється за умовчанням під час установки LabVIEW. Після установки іконка програми MAX відображується на робочому столі Windows [1, 465-486].

17.3. Приклади

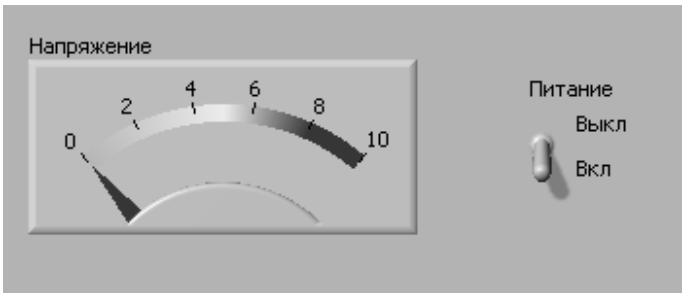
Приклад 17.1. На рис. 17.1 зображено приклад реалізації програмно-апаратного комплексу для введення аналогового сигналу за допомогою DAQ-пристрою, а саме: ВП, що вимірює напругу з датчика температури та відображує її значення на стрілочному індикаторі. Напруга на виході датчика пропорційна температурі.

Алгоритм реалізації:

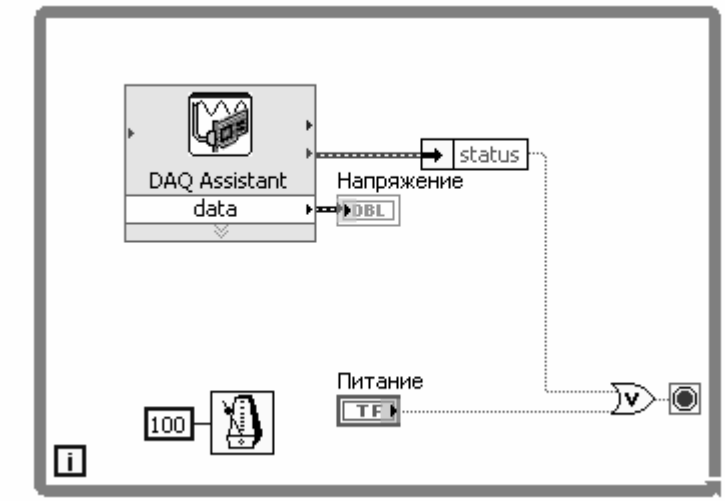
1) спочатку конфігуруємо канали DAQ-пристрою так, щоб датчик був підключений до каналу «0»;
2) поміщаємо на блок-діаграму экспрес-ВП **DAQmx Assistant** з палітри Measurements -> DAQmx-Data Acquisition;
3) вибираємо тип виміру Analog Input ->Voltage, фізичний канал Dev1 -> ai0;
4) у конфігураційному діалоговому вікні Analog Input Voltage Task Configuration встановлюємо метод вимірів Task Timing в положення Acquire 1 Sample (вимір однієї точки);
5) кнопкою OK закриваємо діалогове вікно. Всі установки для цього завдання будуть збережені локально в экспрес-ВП DAQmx Assistant.

Відзначимо, що на блок-діаграмі окрім экспрес-ВП DAQmx Assistant встановлена затримка часу на 100 мс, а також передбачена зупинка

приладу або тумблером "Питание" або при виникненні помилки в экспрес-ВП DAQmx Assistant. Обробка помилки реалізована шляхом виділення елементу status з кластера помилок за допомогою функції Unbundle by Name. За наявності помилки цей елемент матиме значення логічної одиниці.



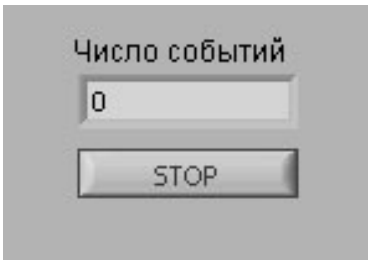
а)



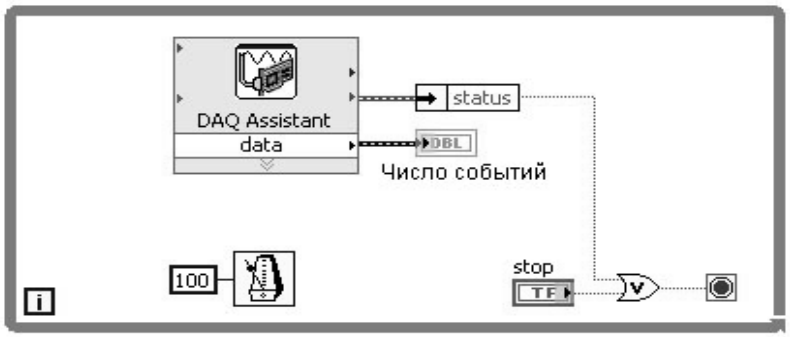
б)

Рис. 17.1 (а, б). Приклад завдання по введенню аналогового сигналу за допомогою DAQ-пристрою (а – передня панель, б – блок-діаграма)

Приклад 17.2. На рис. 17.2 показаний ВП для підрахунку подій, а саме, він рахує імпульси генератора сигнальної панелі DAQ-пристрою.



а)



б)

Рис. 17.2. Прилад для підрахунку подій

(а – передня панель, б – блок-діаграма)

Приклад 17.3. На рис. 17.4 зображено аналізатор спектру, що обробляє вхідний імпульсний сигнал через ВП аналогового входу, а також аналізує його спектральний склад.

Для введення сигналу застосовано віртуальний прилад AI Acquire Waveform (Отримати осцилограму), рис. 17.3, який зчитує серію вибірок з одного каналу. Він розташований в підпалітрі Functions -> NI Measurements -> Data Acquisition -> Analog Input.

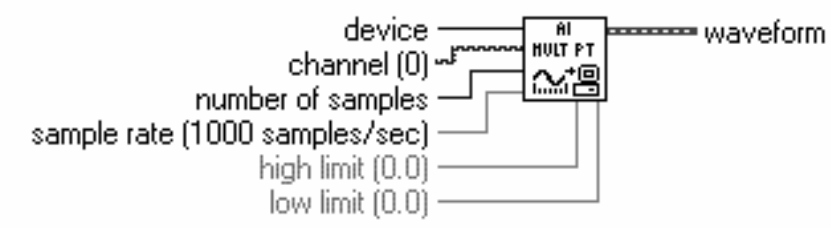


Рис. 17.3. Віртуальний прилад AI Acquire Waveform

Позначення для рис. 17.3:

Device (пристрій) – номер пристрою, призначений платі;

Channel (канал) – визначає фізичний канал на DAQ пристрої;

Number of samples – кількість вибірок на канал;

Sample rate – частота з якою проводитися дискретизація;

High, Low limit верхнє і нижнє обмеження по рівню сигналу.

Спочатку функція AI Acquire Waveform перетворює реальний сигнал в масив вибірок. На виході waveform (масив вибірок) виводиться на Waveform Graph, внаслідок чого на осцилографі спостерігається вхідний сигнал. За допомогою Basic Averaged DC-RMS знаходимо постійну складову DC value і діюче значення RMS value сигналу. Далі масив значень обробляється функцією Harmonic Distortion Analyzer, на вхід якої також подаємо кількість гармонік (в даному випадку 12), на виході отримуємо detected fundamental frequency – основну частоту і components level – масив амплітуд гармонік (починаючи з постійної складової). Далі, для того, щоб відображувати амплітудний спектр, створюємо масив частот в циклі (починаючи з нульової для постійної складової, далі для першої гармоніки, для другої і так далі). В результаті отримуємо два масиви амплітуд і частот гармонік, які збираємо в кластері і відображуємо на графіку XY Graph. Для того, щоб амплітуди гармонік побачити у вигляді лінійчатого спектру, потрібно викликати контекстне меню для легенди графіка і змінити властивість графіка Common Plots на гістограму.

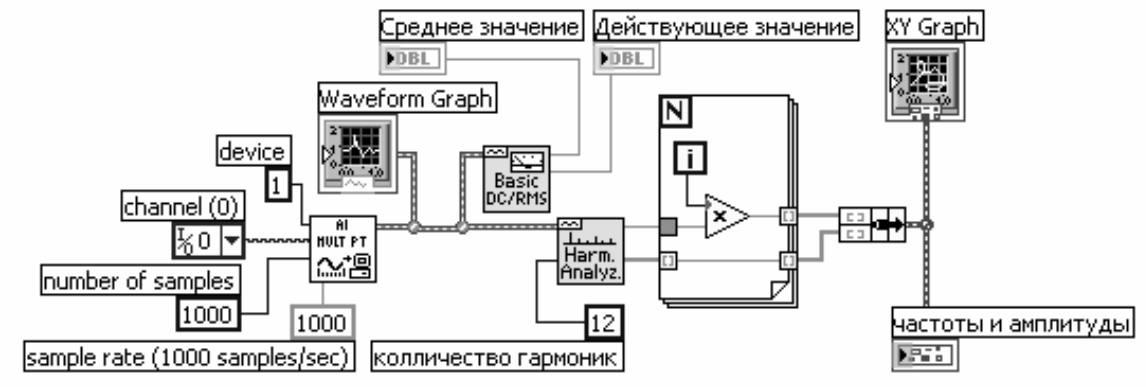


Рис. 17.4. Блок-схема ВП аналізатора спектру

17.4. Завдання на самостійну роботу

Розібратися із функціями аналогового введення низького рівня і реалізувати ВП для безперервного введення даних в один канал.

Варіант реалізації приведено на рис. 17.5.

Тут функція AI Config конфігурує зчитування для певного пристрою (номер якого подається на термінал device) і вказаних в input channels каналів. Створюється буфер розміром input buffer size. ВП AI Start відповідає за запуск зчитування, 0 поданий на термінал number of scans to acquire означає безперервне введення, також необхідним параметром є scan rate (частота опитування). Далі в циклі за допомогою ВП AI Read відбувається зчитування даних і відображення їх на графіку осцилограми. Зчитування закінчується у випадку, якщо користувач натискне кнопку stop або якщо виникне помилка у ВП AI Read. Після закінчення циклу ВП AI Clear завершує завдання. В разі виникнення помилок ВП Simple Error Handler видасть вікно з поясненням помилки.

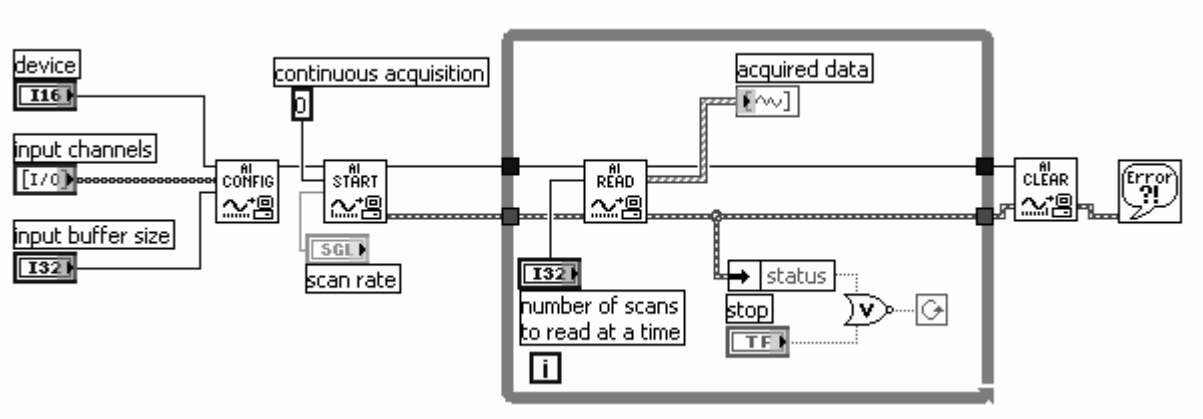


Рис. 17.5. Блок-схема ВП для безперервного введення даних по одному каналу

17.5. Контрольні питання

1. Охарактеризуйте можливості розробки та властивості програмно-апаратних комплексів. Коли і для якої мети їх розробка є доцільною?

18. ІНТЕГРАЦІЯ СЕРЕДОВИЩА LABVIEW З ІНШИМИ СЕРЕДОВИЩАМИ ПРОГРАМУВАННЯ

При розробці ПП вбудовані функціональні можливості системи LabVIEW інколи не є оптимальним рішенням для реалізації або ж їх недостатньо. Тому з'являється необхідність підключення до програм, написаних в системі LabVIEW, додаткового програмного коду або бібліотек. І хоча компілятор LabVIEW зазвичай створює код, який досить ефективний, вузли кодового інтерфейсу і динамічні бібліотеки є корисними інструментами для вирішення завдань, що виконуються в реальному часі, завдань, які вимагають великої кількості маніпуляцій з даними, або за умови, що вже є багато розробленого коду на інших мовах програмування.

Ще однією важливою причиною для впровадження динамічних бібліотек є підключення додаткового устаткування для збору даних. Віртуальні прилади LabVIEW безпосередньо призначені лише для роботи з системами збору даних виробництва National Instruments. Якщо ж використовується устаткування сторонніх розробників, то єдиним способом підключення драйверів таких пристроїв є використання зовнішнього програмного коду, зокрема, динамічних бібліотек. Крім того, інколи виникає необхідність обміну даними і взаємодії з іншими програмними продуктами. З цією метою в LabVIEW існує підтримка технологій *.NET та ActiveX. А завдяки інтеграції середовища програмування LabVIEW з системою MATLAB стає можливим використання у віртуальних приладах LabVIEW скриптових m-файлів, написаних в системі MATLAB [2].

18.1. Мета та завдання

Метою заняття є ознайомлення з можливостями інтеграції системи LabVIEW з іншими мовами та/або середовищами розробки програм. Для цього було поставлено такі завдання:

- з'ясувати можливості інтеграції LabVIEW з MatLab;
- зрозуміти можливості інтеграції у LabVIEW бібліотек dll;
- розібрати різні технології, які можна застосовувати при реалізації ПП у середовищі LabVIEW.

18.2. Можливості інтеграції в середовищі LabVIEW

Середовище графічного програмування LabVIEW має вбудовану підтримку більшості програмних інтерфейсів, таких як Win32 DLL, компоненти COM (ActiveX), технології *.NET, DDE (Dynamic Data Exchange), мережевих протоколів на базі IP, технологію DataSocket та ін. Програмні продукти, створені з використанням LabVIEW, можуть бути доповнені фрагментами коду, розробленими на традиційних алгоритмічних мовах програмування, наприклад, C/C++. І навпаки, модулі, розроблені в середовищі LabVIEW, можна використовувати в проектах, що створюються в інших середовищах розробки програмного забезпечення. Таким чином, LabVIEW дозволяє розробляти практично будь-які програмні продукти, що взаємодіють з різними видами програмних та апаратних засобів.

Звернення до функцій, написаних на зовнішніх мовах програмування, здійснюється за допомогою бібліотек колективного доступу (Shared Library) і впровадженням цих функцій у ВП. Бібліотека колективного доступу складається з набору функцій, до яких ПП звертається під час

виконання програми, а не при компіляції. У Windows ці бібліотеки називаються бібліотеками динамічної компоновки (Dynamic Link Library – DLL), в UNIX їх називають бібліотеками, що розділяються (Shared Objects, Shared Libraries); у MacOS це об'єктні структури (Frameworks).

Для виклику DLL в блок-діаграмі програми LabVIEW використовується вузол «Виклик бібліотечної функції» (Call Library Function Node).

У LabVIEW присутня спеціальна структура блок-діаграми, що називається вузлом кодового інтерфейсу – ВКІ (Code Interface Node – CIN), для інтеграції програмного коду, написаного на ANSI C, у віртуальний прилад. ВКІ викликає виконуваний код, передає в нього вхідні дані і повертає дані після виконання коду в блок-діаграму. ВКІ використовується тоді, коли програмний код незручно перетворювати в динамічну бібліотеку. Варто відзначити, що ВКІ статично вбудовуються в програму. Якщо змінюється код, то необхідно перекомпілювати ВП. ВП з ВКІ не переносяться на інші платформи.

Програма, написана в системі LabVIEW, може використовуватися як клієнт або сервер в технологіях ActiveX.

ActiveX – це набір технологій, які дозволяють програмним компонентам взаємодіяти один з одним по мережі або на локальній машині незалежно від того, на якій мові вони написані. В основі ActiveX лежить модель COM.

Модель COM (Component Object Model) – це модель багатокомпонентних об'єктів, яка реалізує стандартний механізм, за допомогою якого одна частина програмного забезпечення надає свої сервіси іншій.

Технологія Microsoft *.NET є подальшим розвитком технології ActiveX. Так само, як і ActiveX, *.NET використовується в LabVIEW для

забезпечення доступу до інших ПП Windows. LabVIEW може використовуватися як клієнт *.NET для доступу до об'єктів, властивостей і методів, пов'язаних з серверами *.NET. В той же час LabVIEW не є сервером *.NET, тобто інші ПП не можуть безпосередньо взаємодіяти з LabVIEW через *.NET. За допомогою ВП, що використовують *.NET, можна отримати доступ до сервісів Windows API (application programming interface, інтерфейс програмування інструментаріїв). Середовище періоду виконання *.NET Framework включає сервіси компонентів COM+, середовище періоду виконання для Web ASP.NET і підтримку ряду протоколів сервісів Web, таких як SOAP, WSDL і UDDI.

Середовище LabVIEW дозволяє викликати сервер сценаріїв MatLab для виконання коду, написаного на мові MATLAB, а також включати у графічний код скриптові коди на MathScript – текстовій мові математичної обробки, що включає понад 800 математичних функцій, функцій обробки сигналів і аналізу результатів.

18.2.1. Варіанти підключення зовнішнього програмного коду в систему LabVIEW

Розглянемо три основні способи підключення зовнішнього програмного коду в програмний код середовища розробки LabVIEW.

Перший і найпростіший спосіб – запуск окремо виконуваної програми. В цьому випадку можна викликати будь-який виконуваний файл, написаний на іншій мові програмування, який виконує необхідні дії. Дані, отримані в результаті його роботи, можна використовувати далі. Виклик виконуваного файлу здійснюється за допомогою функції «Системний командний рядок» (System Exec) з підпалітри функцій середовища LabVIEW «Засоби взаємодії» – «Бібліотеки і виконувані

програми» (Connectivity -> Libraries & Executables) (рис. 18.1).

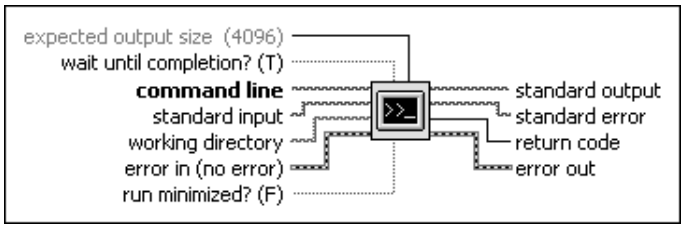


Рис. 18.1. Прототип вузла «Системний командний рядок» (System Exec)

Віртуальний прилад, зображений на рис.18.1, виконує системну команду, яка може включати будь-які параметри, що підтримуються зовнішнім ПП. Для запуску програми з опціями необхідно використовувати синтаксис: `имя_файла.exe+опция1+опция2`.

Також, можна створити файл з розширенням *.bat, який викликає виконуваний файл, і використовувати функцію «Системний командний рядок» для виклику файлу *.bat.

Проте, такий підхід досить незручний в тому випадку, якщо зовнішній ПП отримує або повертає великий об'єм інформації, оскільки дані приймаються/повертаються за допомогою стандартного потоку вводу/виводу у вигляді строки. Таким чином, необхідно аналізувати строкові дані, що є досить неефективно, особливо при поверненні багатовимірних масивів, структур і т.п. Даний спосіб краще всього підходить для виконання різних системних команд без необхідності приймати параметри, що повертаються.

Другий спосіб полягає в написанні необхідного коду на ANSI C і інтеграції його в блок-діаграму за допомогою функції «Вузол кодового інтерфейсу» – ВКІ (Code Interface Node – CIN) з підпалітри функцій LabVIEW «Засоби взаємодії» (Connectivity) – «Бібліотеки і виконувані програми» (Libraries & Executables). ВКІ має термінали для введення і

виведення даних. За умовчанням у ВКІ є лише одна пара терміналів (для вводу/виводу значень). Можна перевизначити розмір ВКІ для включення в нього заданого числа параметрів, як показано на рис. 18.2.

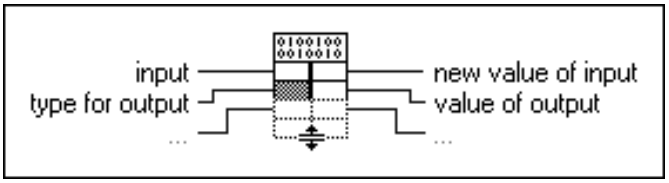


Рис. 18.2. Прототип «Вузла кодового інтерфейсу» (Code Interface Node)

Кожна пара терміналів (на рис. 18.2) є парою вхід/вихід: лівий термінал є входом, а правий – виходом. Проте, якщо функція повертає більше вихідних значень, ніж вхідних (або взагалі не має вхідних параметрів), то є можливість змінити тип терміналу на «Лише вихід». Файл *.c, що генерується LabVIEW в стилі мови програмування С (рис. 18.3 (а), де зображено контекстне меню ВКІ), є шаблоном, в якому надалі розміщується оригінальний С-код.

Приклад коду програми, що обчислює середнє значення масиву чисел, і процес інтеграції його в блок-діаграму (програмний код) ВП наведено нижче:

```
#include "extcode.h"
typedef struct {
     int32 dimSize;
     float64 Numeric[1];
     } TD1;
typedef TD1 **TD1Hdl;
MgErr CINRun(TD1Hdl InputData, float64 *arg1, int32
*errorCode0);
MgErr CINRun(TD1Hdl InputData, float64 *arg1, int32
*errorCode0)
```

```
{
       int i=0;
       for(i=0; i < (*InputData)->dimSize; i++)
         *arg1 += (*InputData) ->Numeric[i];
       *arg1 /= (*InputData)->dimSize;
       return noErr;
 }
```

Для успішної компіляції необхідно створити make-файл наступного змісту:

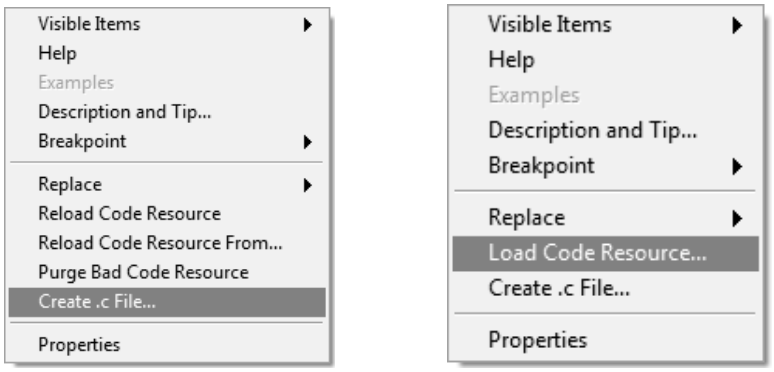
```
name = cinexample
type = CIN
!include $(CINTOOLSDIR)\ntlvsb.mak.
```

При використанні компілятора Microsoft Visual C++ можна скомпілювати файл середовища LabVIEW (*.lvm) за допомогою команди:

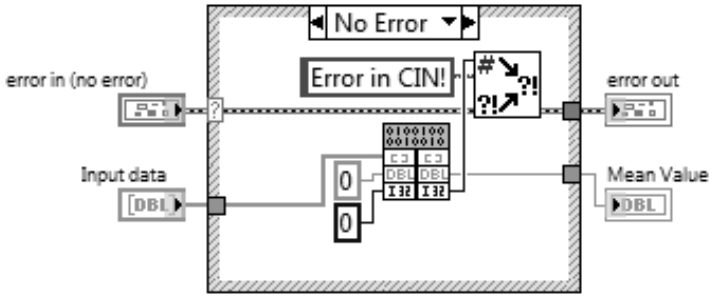
```
nmake /f cinexample.lvm.
```

Після компіляції і перетворення отриманого об'єктного коду в безпосередньо завантажуваний в LabVIEW формат, необхідно завантажити об'єктний код в пам'ять, вибравши опцію «Завантажити кодовий ресурс» (Load Code Resource) (рис. 18.3 (б)) з контекстного меню ВКІ і вказавши файл *.lsb, створений при компіляції коду.

Фрагмент блок-діаграми, що ілюструє підключення зовнішнього коду за допомогою ВКІ зображено на рис. 18.3 (в).



а) б)



в)

Рис. 18.3 (а, б, в). Налаштування Вузла кодового інтерфейсу

Третім, найбільш поширеним і зручним способом інтеграції зовнішнього коду у віртуальний прилад, є підключення бібліотеки динамічної компоновки (DLL) в Windows, бібліотеки, що розділяється (Shared Libraries), в UNIX або об'єктної структури (Frameworks) в MacOS за допомогою ВП «Виклик бібліотечної функції» (Call library function) (рис. 18.4) з підпалітри функцій «Засоби взаємодії» (Connectivity), яка називається «Бібліотеки і виконувані програми» (Libraries & Executables).

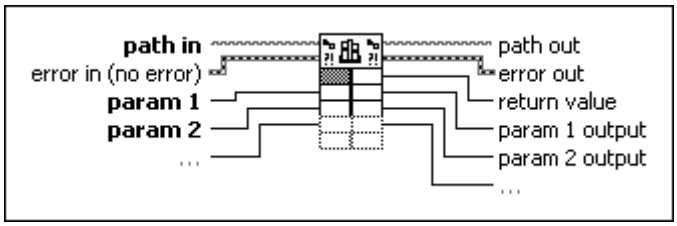


Рис. 18.4. Прототип ВП «Виклик бібліотечної функції»
(Call library function)

На відміну від ВКІ, для створення таких бібліотек можна використовувати будь-який компілятор процедурної мови, що дозволяє створювати бібліотеки динамічної компоновки (для Windows).

Даний спосіб є найбільш простим і ефективним у використанні. Можна скомпілювати DLL бібліотеку з декількома функціями і підключати один *.dll файл в декількох місцях блок-діаграми. Крім того, є можливість скористатися безліччю вже готових бібліотек DLL. Необхідно лише точно знати призначення функцій, що містяться в бібліотеці. У ряді випадків базові драйвери для мікросхем або плат збору/передачі даних поставляються їх виробниками у вигляді набору DLL, розрахованих на використання в текстових мовах.

18.2.2. Інтеграція системи LabVIEW з пакетом прикладних програм MATLAB

Існує два способи інтеграції скриптів m-файлів, розроблених в середовищі MATLAB, в програмні коди середовища LabVIEW.

Перший спосіб – використання вузла «MATLAB script» (рис. 18.5). Даний вузол викликає програмне забезпечення MATLAB для виконання необхідних скриптів.

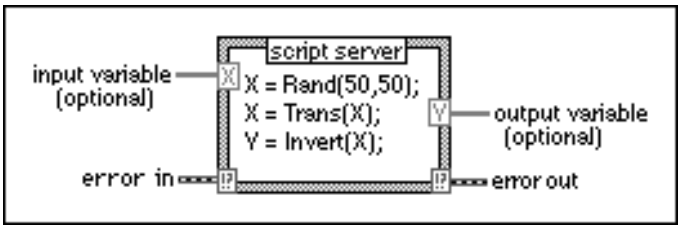


Рис. 18.5. Прототип вузла «MATLAB script»

Для використання «MATLAB script» необхідне програмне забезпечення MATLAB версії не більше 6.5. Оскільки LabVIEW

використовує технологію ActiveX для забезпечення зв'язку з MATLAB, вузол «MATLAB script» доступний лише на Windows платформах.

Використання вузла «MATLAB script» показане на прикладі розв'язання звичайного диференціального рівняння аттрактора Лоренца (рис. 18.6).

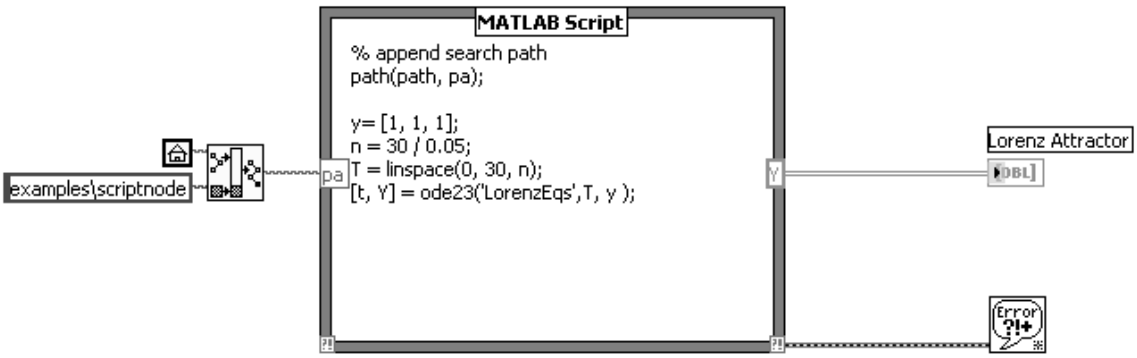


Рис. 18.6. ВП для розв'язання диференціального рівняння Лоренца

Другий спосіб – використання вікна і вузла MathScript. MathScript підтримує як інтерактивний, так і програмний інтерфейс з LabVIEW. Інтерактивний інтерфейс забезпечується за допомогою функції "Вікно MathScript" (MathScript Window), яке запускається з меню Tools системи LabVIEW (рис. 18.7).

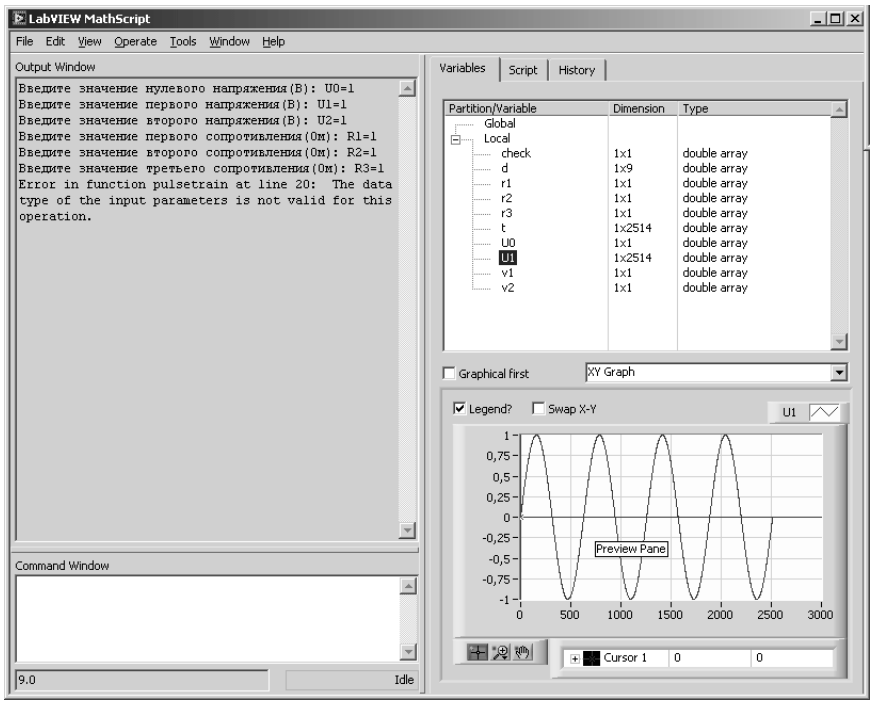


Рис. 18.7. Вікно MathScript (MathScript Window)

Вікно MathScript використовується для редагування і виконання математичних команд, створення математичних скриптів і відображення змінних в табличному або графічному вигляді. У цьому вікні виконується більшість скриптів, записаних відповідно до синтаксису мови MATLAB, але існує ряд функцій мови MATLAB, які не підтримуються у вікні MathScript системи LabVIEW.

Зв'язок функцій, інтегрованих з MatLab, з інтерфейсом користувача в LabVIEW здійснюється за допомогою вузла «MathScript» (MathScript Node) – структури, яка дозволяє виконувати скрипти в блок-діаграмі LabVIEW (рис. 18.8).

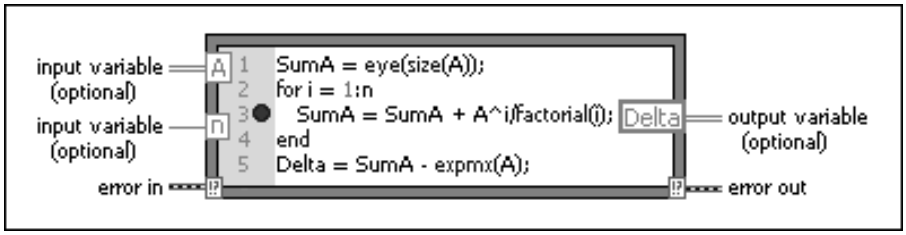


Рис. 18.8. Прототип вузла MathScript (MathScript Node)

Разом зі вбудованими функціями, функціонал вікна MathScript дозволяє користувачеві визначати власні функції. Після опису функції її необхідно зберегти у папці, шлях до якої вказаний в діалоговому вікні MathScript «Preferences». Діалогове вікно викликається за допомогою строки меню Файл (File) «Переваги» (Preferences). Ім'я файлу функції має бути таким же, як і ім'я самої функції, і мати розширення *.m в нижньому регістрі.

18.3. Експорт коду з середовища LabVIEW

Середовище розробки лабораторних віртуальних приладів LabVIEW може працювати і в іншому напрямі. Коли необхідно, аби зовнішній код звертався до програми на LabVIEW, то можна експортувати ВП у виконуваний *.exe файл, DLL бібліотеку в Windows, бібліотеку, що розділяється (Shared Libraries), в UNIX або об'єктну структуру (Frameworks) в MacOS і підключати її в інших програмах.

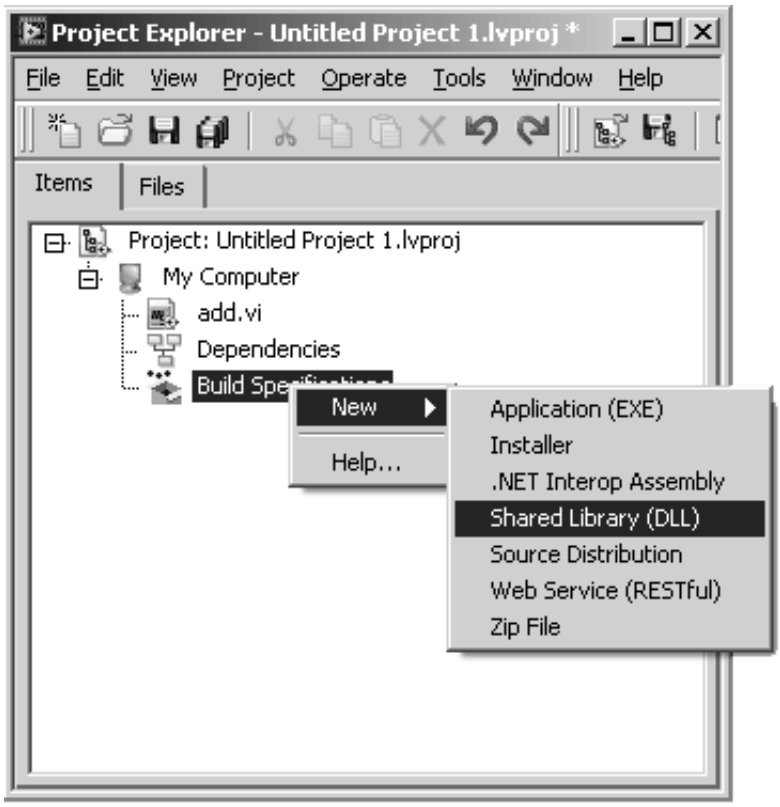
Для створення динамічної бібліотеки з ВП необхідно створити проект і, додавши в «Менеджері проекту» (Project Explorer) необхідний віртуальний прилад, вибрати елемент «Налаштування компоновки» (Build Specifications), а в ньому – «Бібліотека загального доступу (DLL)» (Shared Library (DLL)) (рис. 18.9(а)).

Після визначення функції (заздалегідь налаштувавши «З'єднувальну панель» (Connector) ВП), яку зображено на рис. 18.9 (б), можна створювати DLL-бібліотеку. Створені *.dll і *.lib файли підключають в сторонні проекти на інших мовах програмування. Єдиним недоліком використання таких динамічних бібліотек є необхідність установки модуля NI LabVIEW Run-Time Engine.

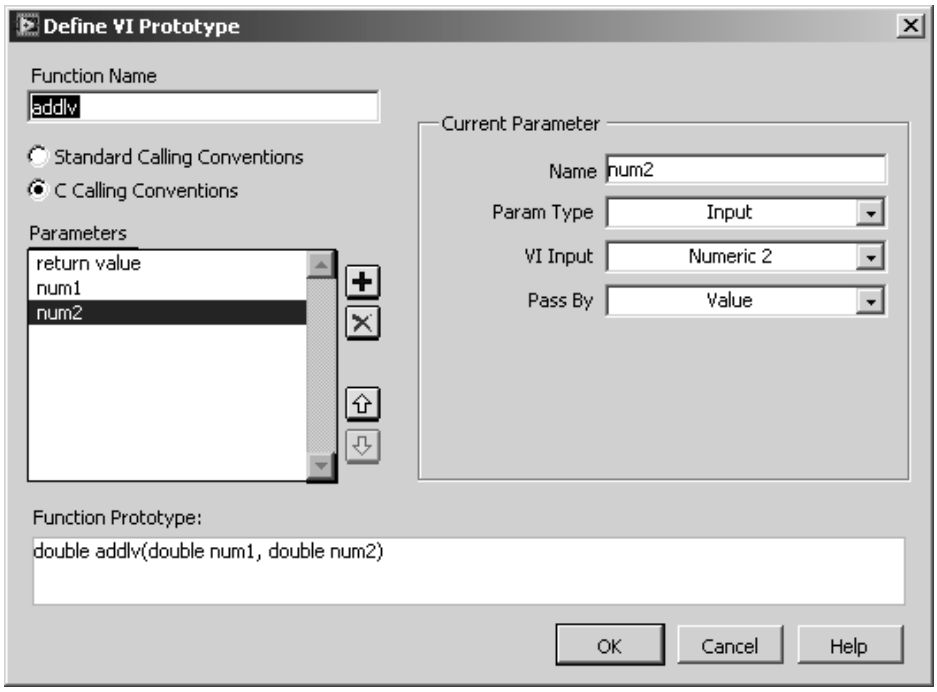
Ще одним способом експорту коду з LabVIEW є використання NI

LabVIEW C Generator для генерації ANSI C коду спроектованого ВП.

Також, LabVIEW надає доступ до властивостей і методів самого середовища або певного ВП іншим ПП ActiveX (наприклад, Microsoft Excel, Visual Basic та ін.) за допомогою інтерфейсу сервера ВП. Доступ до функцій сервера ВП здійснюється по протоколу TCP/IP (лише з інших ПП LabVIEW) через функції блок-діаграми середовища LabVIEW і за допомогою ActiveX. Дозволивши роботу ActiveX-сервера, можна створювати зовнішні програми на інших мовах програмування, наприклад, на Visual Basic або C++, які взаємодіятимуть з LabVIEW [2].



а)



б)

Рис. 18.9. Експорт ВП в динамічну бібліотеку DLL

18.4. Приклади

Приклад 18.1. На рис. 18.10 показано приклад використання вузла «Системний командний рядок» для звернення до програми-архіватора lzw2.exe з підключенням необхідних опцій, параметрів, директорій.

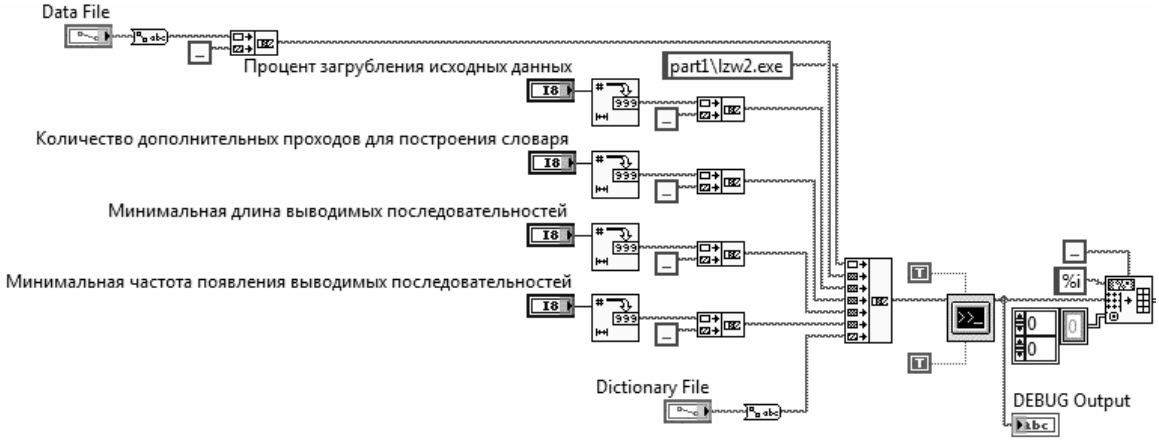
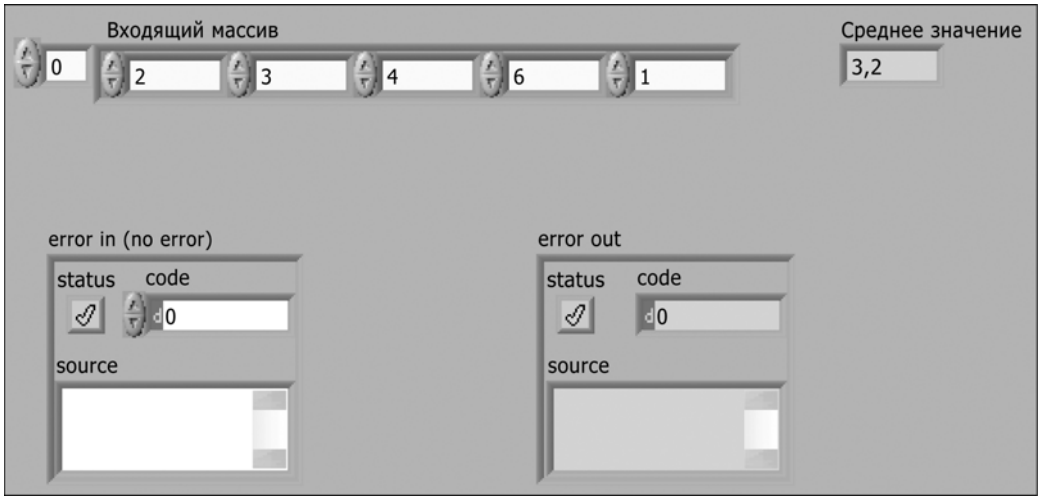
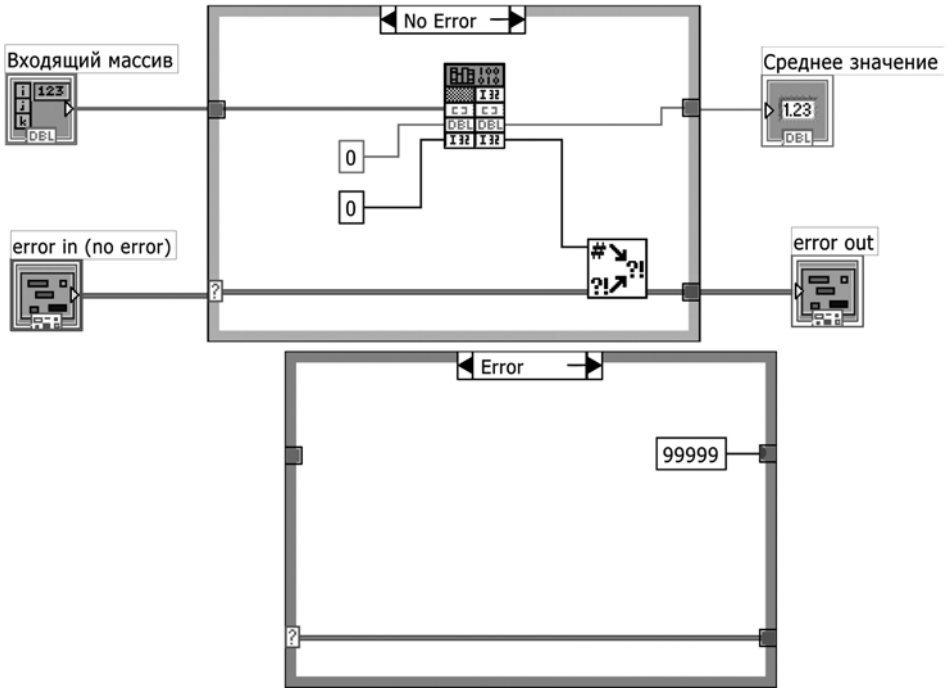


Рис. 18.10. Приклад використання вузла «Системний командний рядок»

Приклад 18.2. На рис. 18.11 зображено приклад виклику DLL-бібліотеки. В даному разі використовується бібліотека DLL, яка вже є в комп'ютері – lvanlys.dll, та призначена для обчислення середнього значення масиву.



а)



б)

Рис. 18.11. Приклад підключення DLL-бібліотеки

18.5. Контрольні питання

1. Які є варіанти підключення зовнішнього програмного коду в LabVIEW?
2. Які завдання вирішуються за допомогою інтеграції програмних середовищ?
3. Які труднощі виникають при застосуванні зовнішнього коду?

ТЛУМАЧНИЙ СЛОВНИК ТЕРМІНІВ

array	масив	Впорядкований, проіндексований набір елементів даних одного типу
auto-indexing	авто-індексація	Здатність циклічних структур формувати і розформовувати масиви на межі структури. При вході масиву в цикл з включеною автоіндексацією цикл розформовує одновимірний масив на скалярні величини, двовимірний – на одновимірні масиви і так далі (вимірність масиву зменшується на одиницю). При виході з циклу відбувається зворотна процедура – вимірність масиву підвищується на одиницю
autoscaling	автомасштабування	Здатність осей графіка автоматично налаштовувати свої масштаби з метою відображення всього діапазону даних
block diagram	блок-діаграма	Графічне представлення програми або алгоритму. В LabVIEW блок-діаграма, що складається з терміналів, виконуваних вузлів і провідників (зв'язків) даних являє собою програмний код ВП. Блок-діаграма розташовується у вікні блок-діаграм ВП
boolean controls	логічні елементи управління	Об'єкти лицьової панелі, використовувані для управління вхідними даними, що мають два стани "Увімкнено" або "Вимкнено" ("Істина" або "Хибність"). До цих елементів належать

		тумблери, перемикачі, кнопки
breakpoint	точка зупинки	Пауза у виконанні програми. Встановити точку зупинки можна на ВП, вузлі або провіднику за допомогою відповідного інструменту з палітри інструментів
breakpoint tool	інструмент для установки точки зупинки	Інструмент, що використовується для встановлення точки зупинки на ВП, вузлі або провіднику
broken VI	несправний ВП	ВП, який не може бути скомпільований або запущений із-за помилок в блок-діаграмі; індикатором несправності є зламана стрілка запуску ВП; при натисканні на неї відображується детальна інформація про помилки
bundle node	вузол об'єднання	Функція, що створює кластер з різноманітних елементів
byte stream file	двійкові файли	Файл, що зберігає дані у вигляді послідовності символів ASCII або байтів
case	варіант	Одна з піддіаграм (кадр) структури з умовним переходом
case structure	структура з умовним переходом	Структура управління з розгалуженням, яка виконує один з її варіантів залежно від стану входу-селектора. Її можна описати як комбінацію операторів IF, THEN, ELSE і CASE в традиційних мовах програмування

channel	канал	У LabVIEW прийнято, що це вивід або контакт, через який вводиться або виводиться аналоговий сигнал
channel name	ім'я каналу	Унікальне ім'я, привласнене програмою DAQ Channel Wizard каналу, що конфігурується
chart	графік осцилограми	Графічне відображення даних, що постійно доповнюються (аналогічно стрічці самописця)
CIN (Code Interface Node)	ВКІ (Вузол кодового інтерфейсу)	Спеціальний вузол блок-діаграми, за допомогою якого можна передати текстовий програмний код у ВП
cloning	копіювання	Для копіювання елементу управління або об'єкту LabVIEW слід клацнути по ньому лівою клавішею миші, утримуючи натиснутою клавішу "Ctrl", і перенести копію в необхідне місце
cluster	кластер	Впорядкований набір елементів даних будь-якого типу, включаючи числові, логічні, строкові, масиви або кластери. Всі елементи кластера мають бути або елементами управління або індикаторами
coercion	приведення типів	Автоматичне приведення (узгодження) типів даних в LabVIEW; змінюється числове представлення елементів даних з метою уникнення конфліктів
coercion dot	точка приведення типів	Спеціальний знак на вузлі або терміналі, що символізує наявність зміни числового представлення даних в цій точці

color copy tool	інструмент копіювання кольору	Копіює кольори для подальшого використання інструментом розфарбовування
coloring tool	інструмент розфарбовування	Інструмент для призначення кольору передньому плану і фону
compile	компіляція	Процес перетворення високорівневого коду в машинний код. LabVIEW автоматично компілює ВП перед першим запуском або запуском після корекції
condition terminal	термінал умови виходу	Термінал циклу по умові, що працює з логічними значеннями і визначає, чи буде ВП виконувати наступну ітерацію
connector	поле вводу/виводу (конектор)	Частина ВП або функціонального вузла, що містить всі вхідні та вихідні термінали, через які дані поступають в вузол або виходять з нього
context help	контекстна допомога	Довідкова інформація щодо об'єкту під курсором миші
control	елемент управління	Об'єкт передньої панелі для введення даних інтерактивно в ВП та автоматично в ВПП
control flow	потік управління	Система програмування, в якій послідовний порядок інструкцій визначає порядок виконання програми. Найпоширеніші текстові мови програмування C, Pascal, Basic є мовами такого типу
control palette	палітра елементів	Палітра, що містить елементи управління

	управління	
count terminal	термінал кількості ітерацій	Термінал циклу з фіксованою кількістю ітерацій, чиє значення визначає задане число повторень в циклі
DAQ (Data Acquisition)	збір даних	Процес збору даних з оточуючого середовища
data flow	потік даних	Система програмування, що складається з виконуваних вузлів, що виконуються лише тоді, коли всі необхідні дані надійдуть на їх входи. LabVIEW – це середовище програмування потоку даних
data type	тип даних	У LabVIEW це – numeric, array, string, boolen, path, refnum enumeration, waveform, cluster
destination terminal	термінал-приймач	Термінал, на який дані надходять
dimension	розмірність	Властивість, що описує розмір і структуру масиву
DLL (dynamic link library)	бібліотека динамічної компоновки	Бібліотека колективного доступу, що складається з набору функцій, до яких застосування звертається під час виконання програми, а не при компіляції
execution highlighting	підсвічу-вання виконання	Режим виконання програми, що дозволяє візуалізувати потік даних у ВП
file refnum	посилання до файлу	Ідентифікатор, який LabVIEW асоціює з файлом при його відкриванні. Використовується для

		ідентифікації файла, до якого застосовується функція або операція
flattened data	приведені дані	Дані будь-якого типу, які були перетворені в рядок (послідовність символів) зазвичай для подальшого запису у файл
for loop	цикл з фіксованою кількістю ітерацій	Структура, що циклічно повторюється – виконує свою піддіаграму задану кількість разів
formula node	вузол "формула"	Вузол, що обчислює формули, записані у вигляді тексту в синтаксисі мови С. Вирази можуть використовувати умовний оператор і оператори порівняння. Особливо корисні при застосуванні довгих формул, що складно представити у вигляді блок-діаграм
frame	кадр	Частина діаграми структури послідовності
free label	вільна мітка	Мітка (напис) на передній панелі ВП або блок-діаграмі, що не належить ніякому об'єкту
front panel	передня панель ВП	Інтерактивний інтерфейс користувача ВП. Побудований подібно до передніх панелей фізичних пристроїв і містить перемикачі, тумблери, графіки, кнопки, світлодіодні та інші індикатори та елементи управління
function	функція	Вбудований елемент, що виконує деяку функцію, схожий на оператор чи функцію в звичайних мовах програмування
functions	палітра	Палітра, що містить структури, константи,

palette	функцій	елементи взаємодії і функції блок-діаграми
G	G	Мова графічного програмування системи LabVIEW
global variable	глобальна змінна	Елемент (структура) LabVIEW, призначений для обміну даними між різними ВП на одному компьютері
GPIB (general purpose interface bus)	КОП (канал загального користу-вання)	Приладовий інтерфейс, він же – HP-IB (Hewlett Packard interface bus), він же – IEEE 488.2. Став фактично стандартом для зв'язку різноманітних приладів з компьютером
help windows	вікно довідки	Спеціальне вікно, що відображує імена і розташування терміналів функцій або ВПП, опис елементів управління і індикаторів, значення універсальних констант, описи і типи даних управляючих атрибутів
icon	іконка	Графічне представлення вузла на блок-діаграмі
indicator	індикатор	Об'єкт передньої панелі ВП для відображення даних-результатів
inf (infinity)	нескінчен-ність	Значення числового індикатора для відображення нескінченності в представленні чисел з плаваючою комою
instrument driver	драйвер приладу	ВП, що керує програмованим пристроєм або приладом
intensity chart/graph	графік інтенсив-ності	Метод відображення тривимірних масивів даних в площині шляхом використання кольору

I/O	ввід/вивід даних	Пересилання даних з комп’ютера або до нього, включаючи канали зв'язку, пристрої вводу/виводу, збір даних та інтерфейси управління
iteration terminal	термінал лічильника ітерацій	Термінал циклу з фіксованим числом ітерацій і циклу по умові, що містить поточне число виконаних ітерацій
label	мітка	Текстовий об'єкт, що використовується для опису інших об'єктів або їх груп на передній панелі ВП і блок-діаграмі
LabVIEW	LabVIEW	Laboratory Virtual Instrument Engineering Workbench (середовище розробки лабораторних віртуальних приладів)
LED	світлодіод	
legend	панель редагування	Об'єкт, що належить графіку, який відображує імена і стилі графіків і надає можливість їх редагування
local variable	локальна змінна	Елемент (структура) блок-діаграми, призначений для управління даними елементів передньої панелі одного ВП без використання провідників (інструменту "котушка")
NaN	NaN (Not a Number)	Значення числового індикатора для об'єкту, що не є числом в представленні чисел з плаваючою комою. Зазвичай з'являється при виконанні невизначеної операції, наприклад, log(-1)
NI-DAQ	NI-DAQ	Комплект драйверів для устаткування National Instruments. Це програмне забезпечення працює

		в якості інтерфейсу між LabVIEW та зовнішніми пристроями
NI-MAX	NI-MAX	National Instruments Measurement and Automation Explorer (програма аналізу вимірів і автоматизації) – програма налаштування, що взаємодіє з NI-DAQ та дозволяє конфігурувати устаткування National Instruments
nodes	вузли	Виконувані елементи блок-діаграми, що включають функції, структури, підприлади
Nondis-playable characters	символи, що не відобра-жуються	ASCII символи, які не можуть бути відображені, такі як "новий рядок", "табуляція" і тому подібне
numeric controls and indicators	числові елементи управління і індикації	Об'єкти передньої панелі ВП, що використовуються для введення і відображення вхідних і вихідних числових даних
Nyquist frequency	частота Найквіста-Котельни-кова	1/2 мінімального значення частоти вибірки, при якій "теоретично" будуть відсутні спотворення або хибні частоти в сигналі, що дискретизується в часі
object	об'єкт	Загальний термін для елементу на передній панелі ВП або блок-діаграмі, включаючи елементи управління і індикації, провідники і вузли, а також імпортовані картинки
operating tool	інструмент управління	Інструмент, що використовується для введення даних в елементи управління і маніпуляцій з

	"палець"	цими елементами на етапі виконання програми
pallete	палітра	Меню, що надає можливі опції, функції і т.п.
path	путь	Путь до файлу або директорії, що описує їх розташування в файловій системі
plot	графік	Графічне відображення масиву даних на графіку або графіку осцилограми
Poly-morphism	Полімор-фізм	Здатність вузла автоматично підстроюватися до даних різного представлення, типу або структури
positioning tool	інструмент переміщен-ня	Інструмент, що використовується для переміщення, виділення і зміни розмірів об'єкту
probe	пробник	Інструмент налаштування для перевірки проміжних значень у ВП
probe tool	інструмент установки пробників	Інструмент, що використовується для установки пробників на провідники
pseudocode	псевдокод	Спрощене, не залежне від мови програмування представлення програмного коду
Representa-tion	представ-лення	Підтипи цифрового типу даних: цілочисельні зі знаком і без нього з різною розрядністю – 8, 16 і 32, з плаваючою комою різної точності – одинарною, подвійною і розширеною, дійсні і комплексні
resizing handles	мітки-маніпуля-тори	Мітки по кутах об'єктів, що позначають точки зміни його розміру (розтягування, стискання)

ring control	кільцевий елемент управління	Особливий числовий елемент управління, що ставить у відповідність 32-битовое ціле число, починаючи з "0", елементу з набору текстових міток або картинок
sequence local	локальна змінна структури послідов-ності	Термінал, що передає дані між кадрами структури послідовності
sequence structure	структура послідов-ності	Структура управління програмою, що виконує кадри в установленому порядку
shift register	зсувний регістр	Механізм (вмикається за бажанням) циклічних структур, що використовується для передачі значень, отриманих на попередніх ітераціях, в поточну
software	програмне забезпе-чення	Програмне забезпечення (ПП)
string	строка	Строковий тип даних
string controls and indicators	строкові елементи управління та індикації	Об'єкти передньої панелі, що використовуються для управління і відображення тексту, що вводиться і генерується
strip mode	"стрічковий" режим	Режим оновлення розгортки осцилограми при надходженні нових даних, що моделює паперово-стрічковий самописець, де графік

		малюється на стрічці, що рухається
structure	структура	Елемент управління виконанням програми, такий як структура послідовності, варіанту, цикли по умові і з фіксованою кількістю ітерацій та ін..
subdiagram	піддіаграма	Блок-діаграмма усередині меж кадру структури
sub VI	ВПП	ВП, що використовується на блок-діаграмі іншого ВП. Аналог терміну "підпрограма"
sweep mode	розгортка з маркером	Режим оновлення при відображенні графіків, де спеціальний маркер відділяє старі дані від нових
terminal	термінал	Об'єкт або область вузла, через який поступають дані
tool	інструмент	Спеціальний режим роботи курсора в LabVIEW, що дозволяє виконувати певні дії
toolbar	лінійка інструмен-тів	Панель, що містить кнопки управління, які можна використовувати для запуску і налаштування ВП
tools palette	палітра інструмен-тів	Палітра, що містить інструменти редагування і налаштування передньої панелі ВП і блок-діаграми
trigger	запуск	Умова запуску або зупинки операції збору даних
tunnel	тунель, точка вводу в цикл/виво-ду з циклу	Термінал входу/виходу даних структури
VI (virtual	ВП	Програма LabVIEW, що моделює зовнішній

instrument)	(віртуаль-ний прилад)	вигляд і функції фізичного приладу або системи
VI server	сервер ВП	Механізм віддаленого програмного управління ВП
VISA (virtual instrument software architec-ture)	стандартна архітектура віртуальних приладів	Бібліотека для стандартизації управління пристроями з різними типами інтерфейсів GPIB (КОП), VXI, RS-232 і ін.
waveform	осцилограма	Тип даних в LabVIEW, який зазвичай представляє сигнал і містить наступні дані: Y – масив відліків значень реєстрованого сигналу з інтервалом дискретизації ΔX, а також X0 – час початку процесу відліків
while loop	цикл по умові	Циклічна структура, що повторює піддіаграму доти, поки не буде виконано умову завершення
wire	провідник (зв'язок)	Шлях проходження даних між вузлами
wire segment	сегмент провідника	Вертикальна або горизонтальна ділянка провідника
wiring tool	інструмент з'єднання	Інструмент "котушка", що використовується для створення шляху передачі даних від термінала-передавача до термінала-приймача

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Трэвис Дж. LabVIEW для всех : 4-е издание, переработанное и дополненное / Дж. Трэвис, Дж. Кринг. – М. : ДМК Пресс, 2011. – 904 с.
2. Киселева О.Г. Интеграция среды NI LabVIEW и других сред программирования / О.Г. Киселева, М.В. Герасимчук // Биомедицинская инженерия. – 2011. – №1. – С. 50–59.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Оксанич А.П. Програмні засоби систем управління (LabVIEW) : Навчальний посібник для студентів вищих навчальних закладів / Оксанич А.П., Притчин С.Е., Петренко В.Р. – Кривий Ріг : Мінерал, 2007. – 503 с.
2. Бондарев В.Н. Цифровая обработка сигналов: методы и средства : Учеб. пособие для вузов / Бондарев В.Н., Трёстер Г., Чернега В.С. – [2-е изд.]. – Х. : Конус, 2001. – 398 с.
3. Блюм П. LabVIEW: стиль программирования / Блюм П. ; пер. с англ. под ред. П. Михеева. – М. : ДМК Пресс, 2008. – 400 с.
4. Автоматизация физических исследований и эксперимента: компьютерные измерения и виртуальные приборы на основе LabVIEW 7 / [Бутырин П.А., Васьковская Т.А., Каратаев В.В., Материкин С.В.] ; под ред. П.А. Бутырина. – М. : ДМК Пресс, 2005. – 264 с.

5. Обработка и анализ цифровых изображений с примерами на LabVIEW и IMAQ Vision / [Визильтер Ю.В., Желтов С.Ю., Князь В.А. и др.]. – М. : ДМК Пресс, 2007. – 464 с.
6. Федосов В.П. Цифровая обработка сигналов в LabVIEW / В.П. Федосов, А.К. Нестеренко ; под ред. В.П. Федосова. – М. : ДМК Пресс, 2007. – 472 с.
7. Евдокимов Ю.К. LabVIEW для радиоинженера: от виртуальной модели до реального прибора : практическое руководство для работы в программной среде LabVIEW / Евдокимов Ю.К., Линдваль В.Р., Щербаков Г.И. – М. : ДМК Пресс, 2007. – 400 с.
8. LabVIEW: практикум по основам измерительных технологий : учебное пособие для вузов / [Батоврин В.К., Бессонов А.С., Мошкин В.В., Папуловский В.Ф.]. – М. : ДМК Пресс, 2005. – 208 с.
9. Батоврин В.К. LabVIEW: практикум по электронике и микропроцессорной технике : учебное пособие для вузов / Батоврин В.К., Бессонов А.С., Мошкин В.В. – М. : ДМК Пресс, 2005. – 182 с.
10. Кехтарнаваз Н. Цифровая обработка сигналов на системном уровне с использованием LabVIEW / Кехтарнаваз Н., Ким Н. ; пер. с англ. – М. : Издательский дом "Додэка-XXI", 2007. – 304 с.

Навчальне видання

Кисельова Ольга Геннадіївна
Соломін Андрій Вячеславович

Технологія створення програмних продуктів

Програмування в NI LabVIEW

Навчальний посібник

В авторській редакції
Надруковано з оригінал-макета замовника

Теиплан 2012 р., поз. 1-2-011

Підп. До друку 18.05.2012. Формат 60x84$^{1}/_{16}$. Папір офс. Гарнітура Times.
Спосіб друку – ризографія. Ум. друк. Арк. 23,95. Обл.-вид. арк.. 39,83. Наклад 100 пр. Зам. №12-138.

НТУУ «КПІ» ВПІ ВПК «Політехніка»
Свідоцтво ДК № 1665 від 28.01.2004 р.
03056, Київ, вул. Політехнічна, 14, корп.. 15
тел.. (44) 406-81-78