МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет
науки і технологій**

Кафедра «Автоматика
та телекомунікації»

В авторській редакції

КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ В СИСТЕМАХ ЗАЛІЗНИЧНОЇ АВТОМАТИКИ

Частина 2

Навчально-методичні рекомендації
до лабораторних занять

Електронне видання

ДНІПРО
2024

УДК 004.432:656.25(076.5)
К 63

Упорядники:
Р. В. Рибалка, В. В. Маловічко, Н. В. Маловічко,
В. В. Лагута

Електронне видання

Схвалено Групою забезпечення якості освітньої програми
«Системи керування рухом поїздів»
спеціальність 273 «Залізничний транспорт»
Протокол № 4 від 16.11.2023

К 63 Комп’ютерні технології в системах залізничної автоматики : навчально-методичні рекомендації до лабораторних занять / упоряд. Р. В. Рибалка, В. В. Маловічко, Н. В. Маловічко, В. В. Лагута ; Укр. держ. ун-т науки і технологій. – Електрон. вид. – Дніпро : УДУНТ, 2024. – Ч. 2. – 49 с.

Навчально-методичні рекомендації призначено для використання здобувачами вищої освіти, які здобувають освітній ступінь «бакалавр» за освітньо-професійними програмами «Автоматика та автоматизація на транспорті» та «Системи керування рухом поїздів» під час виконання лабораторних робіт з навчальної дисципліни «Комп’ютерні технології в системах залізничної автоматики». Навчально-методичні рекомендації містять основні теоретичні відомості, порядок виконання лабораторних робіт та питання для самоконтролю.

Іл. 5. Табл. 11. Бібліогр.: 4 назв.

ЗМІСТ

ВСТУП

Характеристика місця та значення навчальної дисципліни для підготовки фахівця. Ці навчально-методичні рекомендації (далі – НМР) до лабораторних занять складено для опанування навчальної дисципліни «Комп'ютерні технології в системах залізничної автоматики» (далі – Дисципліна), яка викладається під час першого року навчання відповідно до освітньо-професійних програм (далі – ОПП) «Автоматика та автоматизація на транспорті» (далі – ААТ) спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» та «Системи керування рухом поїздів» (далі – СКРП) спеціальності 273 «Залізничний транспорт».

Опанування Дисципліни дозволяє здобувачу, який здобуває освіту за ОПП ААТ, отримати навики використання інформаційних і комунікаційних технологій, здатність вільно користуватись сучасними комп'ютерними та інформаційними технологіями для вирішення професійних завдань, програмувати та використовувати прикладні та спеціалізовані комп'ютерно-інтегровані середовища для вирішення задач автоматизації, робототехніки та зв'язку. Опанування Дисципліни дозволяє здобувачу, який здобуває освіту за ОПП СКРП, отримати навики використання інформаційних і комунікаційних технологій; розвинути здатність до абстрактного мислення, аналізу та синтезу; створити основу для застосування сучасних програмних засобів для розробки проектно-конструкторської та технологічної документації зі створення, експлуатації, ремонту та обслуговування систем керування рухом поїздів, пристроїв залізничної автоматики та їх елементів, а також для організації дії системи звітності та обліку (управлінського, статистичного, технологічного) роботи систем керування рухом поїздів та пристроїв залізничної автоматики, здійснювати діловодство, документування та управління якістю згідно нормативно-правових актів, інструкцій та методик.

Мета лабораторних занять з Дисципліни для цих НМР – сформувати у здобувача освіти (далі – Здобувач) навики: відповідно очікуваним результатам навчання, визначеним в робочій програмі Дисципліни, розробляти програму мовою програмування C# шляхом поєднання видів керування у структурованому програмуванні, декларування та викликів підпрограм; проводити експериментальні дослідження за напрямом застосування комп'ютерних технологій, у т.ч. порівнювати результати дослідження з теоретичними положеннями Дисципліни.

Вимоги до попередніх знань та умінь: знання основ роботи в інтернет-оглядачі (web browser), знання типових операцій з файлами та папками (створення, перейменування, копіювання тощо). Використання технічних засобів: лабораторні роботи (далі – ЛР) виконуються за допомогою комп'ютера, наданого університетом, або комп'ютера Здобувача.

Вимоги до оформлення протоколу / звіту з ЛР (далі – Звіт):

– Допустимі види оформлення Звіту: електронна, паперова.

- Формат аркушу Звіту: А4.
- У разі оформлення Звіту:
  - З використанням засобів комп'ютерної техніки: гарнітура основного тексту – Times New Roman; гарнітура тексту програм – Consolas, Cascadia або інша моноширинна; розмір шрифту – мінімум 10 пт.
  - Власноруч: допускається прикріплення додатків (рисунок, таблиця, текст програми тощо), які може бути надруковано за допомогою комп'ютерної техніки.
- Звіт повинен містити інформацію, яка однозначно ідентифікує його виконавця: прізвище та ім'я Здобувача, номер академічної групи, шифр Здобувача (номер індивідуального навчального плану), номер варіанту (якщо вимагається в ЛР).
- Наповнення Звіту повинне відповідати вимогам розділу «Зміст звіту» відповідної ЛР.
- Записи відповідно порядку виконання ЛР в тексті Звіту позначаються номером відповідного пункту порядку виконання ЛР цих НМР.
- Аналіз результатів експериментів (досліджень) в ЛР повинен містити порівняння цих результатів з теоретичними положеннями Дисципліни.

Порядок проведення ЛР: Здобувач виконує експериментальні дослідження відповідно цих НМР, оформлює та захищає Звіт. Здобувач допускається до захисту Звіту, якщо наповнення Звіту відповідає розділу «Зміст звіту».

Розподіл роботи над виданням між співавторами:

- Рибалка Р. В. – вступ, розробка ЛР № 3;
- Маловічко В. В. – розробка ЛР № 1;
- Маловічко Н. В. – розробка ЛР № 2;
- Лагута В. В. – розробка ЛР № 4.

Визначення номера варіанту завдання

Номер варіанту завдання (якщо вимагається в ЛР) позначається цілим числом і дорівнює сумі двох останніх цифр у шифрі Здобувача (номер індивідуального навчального плану), якщо не вказано іншого.

Якщо чисельне значення номера варіанту завдання перевищує максимальне значення у множині номерів варіантів певної ЛР, то номером варіанту вважати число, яке відображене на цю множину за принципом «mod N». *Приклад*: мінімальний номер варіанта в певній ЛР – один, максимальний номер – 15; номер варіанта Здобувача – 17 (або, наприклад, 32); тоді за номер варіанта Здобувача прийняти значення 17–15=2 (або 32–2×15=2, відповідно).

Якщо чисельне значення номера варіанту завдання менше за мінімальне значення у множині номерів варіантів певної ЛР, то номером варіанту вважати число, яке дорівнює мінімальному значенню множини номерів варіантів цієї ЛР. *Приклад*: мінімальний номер варіанта в певній ЛР – один; номер варіанта Здобувача – нуль; тоді за номер варіанта Здобувача прийняти значення один.

ВИМОГИ З ОХОРОНИ ПРАЦІ ПІД ЧАС ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

Загальні положення

Якщо ЛР виконуються в спеціально призначеному приміщенні університету, то Здобувач зобов'язаний виконувати вимоги правил техніки безпеки та поводження у цьому приміщенні (ці вимоги повідомляються Здобувачу перед початком або на початку першого лабораторного заняття).

Якщо ЛР виконуються на комп'ютері, наданому університетом, то:

- Здобувачу *заборонено* змінювати налаштування програмної та технічної частини комп'ютера, окрім випадків, визначених цими НМР та вказівками відповідального працівника університету (чергового по аудиторії, викладача тощо, далі – Відповідальний).
- У разі непередбаченої роботи обладнання Здобувач зобов'язаний терміново повідомити про це Відповідальному.

Обладнання робочого місця користувача комп'ютера (далі – Користувач):

- Основне: монітор, клавіатура, робочий стіл, стілець (крісло);
- Допоміжне: підставка для ніг, шафи, полиці та інше.

Взаємне розташування елементів робочого місця Користувача не повинно заважати виконанню необхідних рухів та переміщень для експлуатації комп'ютера; сприяти оптимальному режиму праці й відпочинку, зниженню втоми Користувача.

Поверхню екрана монітора потрібно розташувати в оптимальній зоні інформаційного поля в площині, перпендикулярній нормальній лінії погляду Користувача, який знаходиться в робочій позі. Допускається:

- відхилення від цієї площини не більше 45°;
- відхилення лінії погляду від нормальної не більше 30°.

Розташувати монітор на робочому місці необхідно так, щоб поверхня екрана знаходилась на відстані 500…600 мм від очей Користувача, залежно від розміру екрана.

Клавіатуру потрібно розташовувати на робочому столі (не допускаючи її хитання) або на окремому столі (якщо клавіатуру виконано як окремий пристрій) на відстані 100…300 мм від краю, що є ближчим до Користувача. Положення клавіатури та кут її нахилу (в межах 5…15°) повинні відповідати побажанням Користувача.

Крісло повинно забезпечувати підтримування раціональної робочої пози під час виконання основних операцій. Поверхня сидіння має бути плоскою, передні краї – закругленими.

Раціональна поза Користувача: розташування тіла, при якому ступні Користувача розташовані на площині підлоги або на підставці для ніг, стегна зорієнтовані у горизонтальній площині, верхні частини рук – вертикальні, кут ліктьового суглоба коливається у межах 70…90°, зап'ястя зігнуті під ку-

том не більше ніж 20°, нахил голови – у межах 15…20°, також виключені часті її повороти.

Вимоги безпеки перед початком роботи

Оглянути робоче місце щодо відсутності сторонніх предметів. Якщо комп'ютер виконано у версії, якою передбачено під'єднання периферійного обладнання за допомогою з'єднувальних шнурів (кабелів), то перевірити, чи все необхідне обладнання з'єднано відповідно.

Перевірити надійність встановлення апаратури на робочому столі. Монітор повинен розміщуватись *не* на краю стола. Повернути монітор так, щоб було зручно дивитися на екран – під прямим кутом (а не збоку) і трохи зверху вниз; при цьому екран має бути трохи нахиленим – нижній його край ближче до Користувача. Оглянути та перевірити загальний стан апаратури, справність електропроводки, з'єднувальних шнурів (кабелів), штепсельних вилок, розеток, заземлення захисного екрана.

У разі виявлення будь-яких несправностей чи невідповідностей – роботу *не* розпочинати і повідомити про це Відповідальному.

Вимоги безпеки під час роботи

Під час роботи на клавіатурі Користувач повинен сидіти прямо, не напружуватися.

Заборонено:

- самостійно ремонтувати та очищувати апаратуру на робочому місці;
- перевищувати тривалість безперервної роботи за монітором, що складає 2 години без регламентованої перерви.

Вимоги безпеки після закінчення роботи

У разі потреби – зберегти файли, які є відкритими та перебувають в режимі редагування. Забрати з робочого місця особисті речі Користувача.

Вимоги безпеки в аварійних ситуаціях

У випадку раптового припинення постачання електроенергії вимкнути комп'ютер у такій послідовності: периферійні пристрої (у т.ч. монітор), процесор, стабілізатор напруги (якщо є), витягнути штепсельні вилки з розеток.

У разі виявлення ознак горіння (дим, запах гару):

1. Вимкнути апаратуру, повідомити Відповідальному (у разі його відсутності – знайти джерела займання і вжити заходів щодо ліквідації займання).

2. Якщо немає можливості швидкого відключення електропроводів від джерел постачання, частину що горить, потрібно тушити тільки вуглекислотним вогнегасником або сухим піском.

Якщо стався нещасний випадок, потрібно:

1. Надати потерпілому першу медичну допомогу.

2. Повідомити Відповідальному.

3. У разі потреби – викликати «швидку допомогу».

ОПИС ЕКСПЕРИМЕНТАЛЬНОЇ УСТАНОВКИ

Усі ЛР виконуються з використанням комп'ютера. Вимоги до комп'ютера – установлено:

- інтегроване середовище розробки Visual Studio версії не раніше 2008 року з англійською локалізацією. Visual Studio Community 2022 доступно для завантаження за посиланням https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&channel=Release&version=VS2022&source=VSLandingPage&cid=2030&passive=false;
- .NET Framework версії не раніше 3.5 (доступно для завантаження на сторінці за посиланням https://dotnet.microsoft.com/en-us/download/dotnet-framework) або .NET (доступно для завантаження на сторінці за посиланням https://dotnet.microsoft.com/en-us/download).

ПОЧАТКОВІ НАЛАШТУВАННЯ

Якщо ЛР виконується на комп'ютері, наданому університетом, то для збереження результатів виконання ЛР Здобувач повинен створити папку (далі – Робоча папка):

1. Уточнити у викладача розташування, призначене для збереження файлів Здобувачів, та перейти до цього розташування (наприклад, за допомогою файлового менеджера).

2. Створити Робочу папку з назвою, яка має структуру «НомерГрупи»-«_»-«Прізвище», де «НомерГрупи» та «Прізвище» – номер групи та прізвище Здобувача відповідно. *Приклад*: «АБ1234_Шевченко» без подвійних лапок. В подальшому результати всіх ЛР зберігати лише у даній Робочій папці (якщо не вказано іншого).

3. Створити папку з назвою, яка має структуру «ЛР»-«_»-«номер», де «номер» – номер поточної ЛР. *Приклад*: «ЛР_1» без подвійних лапок.

4. Перейти до папки «ЛР_номер». В подальшому результати всіх ЛР зберігати у відповідних папках (якщо не вказано іншого).

ЛАБОРАТОРНА РОБОТА № 1
ДОСЛІДЖЕННЯ ВИКОНАННЯ ПРОГРАМИ В РЕЖИМІ ПЕРЕРИВАННЯ

Мета роботи: дослідити виконання програми в режимі переривання на прикладі задачі обчислення величини електричного струму в опорі; отримати уявлення про Visual Studio, будову і функціонування консольного додатку та додатку Windows Forms.

Основні теоретичні відомості

Для підвищення продуктивності розробки програмного забезпечення розробники, як правило, використовують певне інтегроване середовище розробки (integrated development environment або IDE). Основні переваги IDE:

- Окрім можливостей простого текстового редактора, IDE, як правило, надає візуальне виділення фрагментів тексту програми залежно від мови програмування («підсвічування синтаксису»). Це спрощує розробнику процес читання програми.
- IDE може пропонувати текст для доповнення існуючого тексту програми (у т.ч. з використанням штучного інтелекту) та перевіряти текст програми на наявність певних видів помилок.
- IDE, як правило, додатково надає такі можливості
  - організація файлів програми у певні логікові сутності («контейнери») для спрощення навігації у складній кодовій базі;
  - *налагодження* (або *debug* – виявлення, визначення та усунення помилок у програмах [1]): встановлювати точки переривання, досліджувати вміст змінних, виконувати покрокові операції тощо.

У даній ЛР в якості IDE використано *Visual Studio* (далі – VS) – інтегроване середовище розробки для .NET розробників.

Контейнери, реалізовані у VS:

- *Рішення* (solution) – контейнер, що містить:
  - проект (один або декілька);
  - елементи рішення (папки рішень, файли та метадані).
- *Проект* (project) – контейнер в межах *рішення* для логікового керування, побудови та налагодження елементів, які утворюють додаток. *Приклади*: бібліотека класів, консольний додаток тощо.

Після відкриття рішення VS автоматично завантажує усі проекти, які містяться у цьому рішенні. Під час створення нового проекту VS автоматично створює рішення для цього проекту, якщо цей проект не додається до існуючого рішення.

Програма під час налагодження у VS може перебувати у таких режимах:

- *Розробка* (design mode) – *не*запущений (nonrunning) стан налагодження VS [4]. Тобто процес написання тексту програми – процес подання програми мовою програмування (coding) [1].
- *Запускання* (run mode) – режим під час *запускання* (стосується стану задачі виконуваного завдання, у якому завдання на заданий момент призначено процесору [1]) програми у сесії налагодження [4]. Тобто здійснюється виконання комп'ютерної програми (execute) [1].
- *Переривання* (break mode) – режим, що виникає у разі призупинення виконання програми, яка налагоджується [4]. У цьому режимі розробник працює зі станом програми, який є актуальним на момент її призупинення, може аналізувати цей стан (наприклад, переглядати вміст змінних) і керувати виконанням програми (виконувати покрокові операції або повернутись до стану запускання).

Розробник може визначити точку в програмі, модулі чи операторі, де виконання може бути припинено залежно від конкретної умови чи події – *точка переривання* або *контрольна точка* (breakpoint) [1]. *Покрокова операція* (step-by-step operation) – режим роботи комп'ютера, за якого виконується окрема інструкція або частина інструкції у відповідь на зовнішній сигнал [1]. Інші назви: однокрокова операція (single-step operation), однокрокове виконання (single-step execution).

За допомогою .NET можна створювати різні програмні рішення, наприклад, веб-сайти, консольні додатки у Windows, Linux і macOS. У даній ЛР розробляються такі додатки:

- Консольний додаток – додаток з текстовим інтерфейсом.
- Додаток Windows Forms – додаток з графічним інтерфейсом.

Windows Forms – інтерфейс програмування додатків (application programming interface або API) для створення додатків з графічним інтерфейсом під Windows. *Форма* (form) у Windows Forms – візуальна поверхня, на якій відображено інформацію для користувача [4].

У даній ЛР розглядаються основні контейнери VS (одне рішення з двома проектами – консольний додаток та додаток Windows Forms), досліджується поведінка IDE під час розробки та налагодження програм з використанням точок переривання та покрокових операцій. При цьому використано класи та елементи керування `Windows Forms`, що подано нижче.

Деякі методи класу `Console` (простір імен `System`):

- `Console.Write` – метод (`static`), записує текстове представлення заданого значення до стандартного вихідного потоку [4].
- `Console.WriteLine` – метод (`static`), записує вказані дані з позначенням кінця поточного рядку до стандартного вихідного потоку [4].
- `Console.ReadLine` – метод (`static`), зчитує наступний рядок символів зі стандартного вхідного потоку [4].

Деякі методи класу `Convert` (простір імен `System`):

– `Convert.ToDouble` – метод (`static`), перетворює вказане значення на число з рухомою комою подвійної точності [4].
– `Convert.ToString` – метод (`static`), перетворює вказане значення на його еквівалентне рядкове представлення [4].

Деякі елементи керування `Windows Forms`:

– `TextBox` – клас (простір імен `System.Windows.Forms`), надає елемент керування «текстове поле» для Windows [4]. Деякі властивості:
  ◦ `ReadOnly` – властивість (типу `bool`), повертає чи задає значення, що визначає, чи є текст в текстовому полі доступним лише для читання [4].
  ◦ `Text` – властивість (типу `string`), отримує чи задає текстовий вміст текстового поля [4].
– `Button` – клас (простір імен `System.Windows.Forms`), надає елемент керування «кнопка» Windows [4].
– `Label` – клас (простір імен `System.Windows.Forms`), надає стандартну «мітку» Windows [4].

Деякі властивості класу `Control` (простір імен `System.Windows.Forms`):

– `Name` – властивість (типу `string`), повертає чи задає ім’я елементу керування [4].
– `Text` – властивість (типу `string`), повертає чи задає текст, співставлений з елементом керування [4].

Постановка задачі

Розробити консольний додаток, який:

– дозволяє користувачу: задавати числові значення електричної напруги (В) та опору (Ом);
– виводить на екран: значення сили електричного струму (А) в опорі, обчислену на основі заданих значень електричної напруги (В) та опору (Ом) за законом Ома для ділянки кола.

Розробити додаток Windows Forms відповідно до тих же вимог, що і консольний додаток (див. вище).

Порядок виконання лабораторної роботи

1. В Робочій папці створити папку для поточної ЛР.

2. Створити у VS рішення (solution) `Lr1` з проектом `ConsoleApp` (розташування та назва елементів інтерфейсу користувача VS залежить від версії VS):

2.1. Запустити VS.

2.2. Натиснути лівою кнопкою миші (далі – ЛКМ) по кнопці `Create a new project`.

2.3. З випадних списків обрати мову C#, платформу – Windows, тип проекту – Console.

2.4. ЛКМ по Console Application (або Console App). Натиснути кнопку Next (OK, або Create).

2.5. В полі Project Name ввести ConsoleApp, а в полі Solution name ввести Lr1.

2.6. Задати розташування (поле Location) ідентичне папці поточної ЛР.

2.7. З випадного списку обрати .NET Framework 3.5 (або більш нову версію), якщо доступно (залежить від обраної реалізації .NET та версії VS).

2.8. Натиснути кнопку Next (OK, або Create).

2.9. Якщо VS надає можливість *не* використовувати «оператори верхнього рівня» («do not use top-level statements»), то активувати цю можливість (залежить від обраної реалізації .NET та версії VS). Натиснути кнопку Next (OK, або Create).

3. У файловому менеджері операційної системи відкрити папку Lr1. Її елементи – до Звіту.

4. Виконати початкові налаштування VS: відобразити вікна (меню View) Solution Explorer, Error List, Toolbox, Properties Window.

5. Внести зміни до тіла методу Main. Текст програми вводити між фігурними дужками { }, які обмежують тіло методу Main:

5.1. Ввести код, після виконання якого:

5.1.1. До консольного вікна буде виведено рядок тексту «U, B: » (повідомлення для користувача, що від нього очікується введення значення напруги у вольтах). Для цього в редакторі коду надрукувати (разом з крапкою в кінці)

```
Console.
```

З переліку, який з'явився на екрані, обрати Write (якщо перелік не з'явився, продовжувати друкувати рядок самостійно): використовуючи ЛКМ чи курсор на клавіатурі виділити відповідний рядок випадного списку, натиснути клавішу Enter. Продовжувати вводити, щоб отримати такий рядок

```
Console.Write("U, B: ");
```

5.1.2. Консольний додаток очікуватиме на те, що користувач введе значення до консольного вікна і натисне клавішу Enter. Після натискання клавіші Enter програма прочитає це значення та присвоїть його змінній U типу double. Для цього в редакторі коду в наступному рядку надрукувати

```
double U = Console.ReadLine();
```

5.2. Скомпілювати рішення: меню Build\ Build solution. Опис помилок у вікні Error List – до Звіту.

5.3. Виправити помилку читання з консольного вікна: в рядку коду, в якому виконується присвоєння змінній U (п. 5.1.2), замість виклику методу

`Console.ReadLine()` викликати метод `Convert.ToDouble()` і в якості його аргументу передати виклик методу `Console.ReadLine()`, тобто

```
double U = Convert.ToDouble(Console.ReadLine());
```

5.4. Скомпілювати рішення. Результат порівняння вмісту вікна `Error List` при даній та попередній компіляції – до Звіту.

5.5. За аналогією з п. 5.1 починаючи з наступного рядка самостійно ввести код, після виконання якого:

5.5.1. До консольного вікна буде виведено рядок тексту «R, Ом: » (повідомлення для користувача про те, що від нього очікується введення значення опору в омах). Див. п. 5.1.1.

5.5.2. Консольний додаток очікуватиме на те, що користувач введе значення до консольного вікна і натисне клавішу `Enter`. Після натискання клавіші `Enter` програма прочитає це значення та присвоїть його змінній `R` типу `double`. Див. п. 5.3.

5.6. В наступному рядку ввести код, після виконання якого буде обчислено значення сили електричного струму `I` за законом Ома для ділянки кола,

```
double I = U / R;
```

та до консольного вікна виведено результат обчислення (відповідний код ввести в наступному рядку)

```
Console.WriteLine("I, A: " + I);
```

5.7. Дослідити виконання програми в режимі налагодження:

5.7.1. Скомпілювати рішення. Запустити проект в режимі налагодження: меню `Debug\ Start Debugging`.

5.7.2. В консольному вікні додатку після запрошення до введення значення напруги («U, В: ») ввести ціле число (обрати самостійно) і натиснути на клавішу `Enter`.

5.7.3. Аналогічно п. 5.7.2 після запрошення до введення значення опору («R, Ом: ») ввести ціле число і натиснути на клавішу `Enter`.

5.7.4. Опис результату виконання програми – до Звіту.

5.8. Дослідити виконання програми в режимі *без* налагодження:

5.8.1. Скомпілювати рішення. Запустити проект в режимі *без* налагодження: меню `Debug\ Start Without Debugging`.

5.8.2. Виконати пункти 5.7.2, 5.7.3, 5.7.4 (до Звіту).

5.8.3. Закрити вікно додатку.

5.9. В наступному рядку ввести код, після виконання якого до консольного вікна буде виведено рядок тексту «Для виходу натисніть Enter»

```
Console.WriteLine("Для виходу натисніть Enter");
```

та консольний додаток очікуватиме натиснення клавіші `Enter` в режимі налагодження, після чого закриється

```
Console.ReadLine(); // очікування натиснення Enter
```

5.10. Виконати п. 5.7 (до Звіту). Початкові дані – також до Звіту. Закрити вікно додатку.

5.11. Зберегти рішення: File\ Save All.

5.12. У файловому менеджері операційної системи відкрити папку Lr1. Назву файлу рішення (з розширенням «SLN») – до Звіту. Перейти до папки Lr1\ ConsoleApp, назви файлів проекту та висхідного коду – до Звіту.

6. Дослідити реакцію VS на помилку часу компіляції:

6.1. В тексті програми замінити double U на int U.

6.2. Скомпілювати рішення. Опис помилок у вікні Error List – до Звіту. Відмінити внесені зміни.

7. Дослідити реакцію VS та додатку на помилку часу виконання:

7.1. Виконати п. 5.7.1.

7.2. В консольному вікні додатку після запрошення до введення напруги («U, B: ») ввести довільний рядок *нечислових* даних (наприклад, текст) і натиснути на клавішу Enter. Опис реакції VS – до Звіту.

7.3. Зупинити налагодження: меню Debug\ Stop Debugging.

7.4. Виконати п. 5.8.1.

7.5. Виконати п. 7.2. Опис реакції додатку – до Звіту.

7.6. Закрити вікно додатку.

8. Зберегти рішення (п. 5.11). Остаточний текст програми – до Звіту.

9. В рішенні Lr1 створити проект WinApp: у вікні Solution Explorer натиснути правою кнопкою миші (далі – ПКМ) по назві рішення (Lr1), Add\ New Project…, Widows Forms Application (залежно від версії VS може знаходитись в категорії Desktop). В полі Name ввести WinApp. Якщо VS надає можливість *не* використовувати «оператори верхнього рівня» («do not use top-level statements»), то активувати цю можливість (залежить від обраної реалізації .NET та версії VS). Натиснути кнопку Next (OK, Create тощо).

10. У файловому менеджері операційної системи відкрити папку Lr1. Назву *доданої* папки – до Звіту.

11. У VS встановити проект WinApp *стартовим* проектом: у вікні Solution Explorer ПКМ по назві проекту WinApp, Set As Startup Project.

12. В режимі *конструктора* (designer) форми створити графічний інтерфейс додатку. За потреби: у вікні Solution Explorer ПКМ по Form1.cs, у контекстному меню ЛКМ по View Designer:

12.1. Елементи з вікна Toolbox розташувати на формі подібно до прикладу вікна додатку на рис. 1.1.

12.2. Задати властивості елементам відповідно табл. 1.1. Для цього для кожного елементу з тих, які розташовано на формі: ЛКМ по елементу, у вікні Properties, на вкладці Properties () знайти потрібну властивість (задано в колонці «Властивість» табл. 1.1), у полі праворуч задати потрібне значення (задано в колонці «Значення» табл. 1.1), натиснути клавішу Enter.

12.3. До Звіту занести:

– ескіз (або скріншот) форми (подібно до рис. 1.1);
– перелік відповідностей «елемент керування на ескізі (скріншоті) форми» – «властивість `Name` цього елемента» (подібно до рис. 1.1).

Примітки: на рис. 1.1 подано перелік відповідностей «елемент керування» – «ім'я *класу* цього елемента», а не «*властивість* `Name` цього елемента».

12.4. Скомпілювати та запустити проект.

12.5. У вікні додатку ввести довільні цілочислові значення. Натиснути на кнопку `Обчислити`. Опис реакції додатку – до Звіту. Закрити додаток.

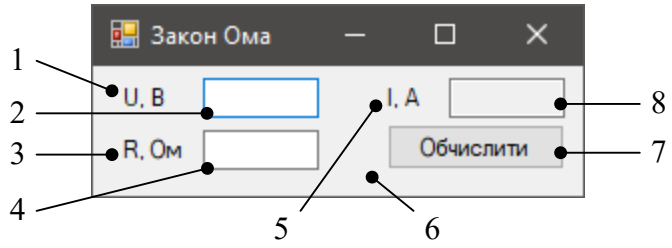


Рис. 1.1. Вікно проекту `WinApp` з елементами керування:
1, *3*, *5* – `Label`; *2*, *4*, *8* – `TextBox`; *6* – `Form`; *7* – `Button`

Таблиця 1.1

Елементи керування на формі проекту `WinApp`

№ на формі	Назва	Властивість	Значення
1	`label1`	`Text`	`U, B`
2	`textBox1`	`Name`	`txtBxU`
3	`label2`	`Text`	`R, Ом`
4	`textBox2`	`Name`	`txtBxR`
5	`label3`	`Text`	`I, A`
6	`Form1`	`Text`	`Закон Ома`
7	`button1`	`Name`	`btnOK`
		`Text`	`Обчислити`
8	`textBox3`	`Name`	`txtBxOut`
		`ReadOnly`	`True`

13. Створити *обробники подій* (далі – обробники) для компонентів форми відповідно табл. 1.2. Для цього для кожного елементу з тих, які вказано в колонці «Компонент» табл. 1.2: в режимі конструктора ЛКМ по компоненту, у вікні `Properties`, на вкладці `Events` (), знайти потрібну подію (задано в колонці «Подія» табл. 1.2), подвійний ЛКМ у полі напроти відповідної події.

14. Доповнити тіло обробника `btnOK_Click` (між фігурними дужками `{ }` – тіло методу) такими рядками:

```
double U = Convert.ToDouble(txtBxU.Text);
double R = Convert.ToDouble(txtBxR.Text);
double I = U / R;
txtBxOut.Text = Convert.ToString(I);
```

15. Скомпілювати та запустити проект в режимі налагодження. Ввести довільні цілочислові значення, натиснути кнопку `Обчислити`. Опис результату – до Звіту. Закрити додаток.

Таблиця 1.2

Події для компонентів форми проекту `WinApp`

Компонент	Подія	Ім'я обробника	Призначення
`btnOK`	`Click`	`btnOK_Click`	виконання обчислень
`Form1`	`Load`	`Form1_Load`	налаштування форми під час завантаження форми

16. Доповнити тіло обробника `Form1_Load` (між фігурними дужками `{ }` – тіло методу) кодом, який властивостям `Text` елементів `txtBxU` і `txtBxR` присвоює числові значення. При цьому задати хоча б одне значення з *рухомою комою* (*цілочисловий розділовий знак* – символ, який розділяє цілу та дробову частину числа – крапка). *Приклад*:

```
txtBxU.Text = Convert.ToString(2.7);
txtBxR.Text = Convert.ToString(50);
```

17. Дослідити роботу програми при введенні *не*числових даних:

17.1. Скомпілювати та запустити проект в режимі налагодження. Зміни у відображенні вікна додатку та цілочисловий розділовий знак (як правило, *крапка* або *кома*) з відповідного `TextBox` – до Звіту.

17.2. У вікні додатку в будь-якому `TextBox` (`txtBxU` або `txtBxR`) замінити цілочисловий розділовий знак з коми на крапку (чи навпаки, залежно від налаштувань середовища виконання).

17.3. Натиснути кнопку `Обчислити`. Опис результату – до Звіту. Зупинити налагодження.

18. Дослідити виконання програми в режимі переривання:

18.1. Встановити точку переривання на рядку

```
double U = Convert.ToDouble(txtBxU.Text);
```

Для цього: встановити курсор на даному рядку коду, меню `Debug\ Toggle Breakpoint`.

18.2. Скомпілювати та запустити проект в режимі налагодження, натиснути кнопку `Обчислити`.

18.3. Перемістити фокус введення (перейти) до редактора коду (ЛКМ по редактору коду). Розташувати вказівник миші поверх змінної U (вказівник миші візуально перекриває змінну). Значення змінної U в підказці (або на вкладці Locals чи Autos) – до рядка «Перед присвоюванням» табл. 1.3 (колонка «U»). Аналогічно записати значення R та I.

18.4. Виконати операцію (присвоювання значення змінній U – рядок в п. 18.1): меню Debug\ Step Over. Значення U занести до рядка «Після присвоювання» табл. 1.3 (колонка «U»).

Таблиця 1.3

Значення змінних

Параметр	Ім'я (ідентифікатор) змінної		
	U	R	I
Перед присвоюванням			
Після присвоювання			

18.5. Виконати операцію присвоювання значення змінній R (меню Debug\ Step Over). Значення R – до рядка «Після присвоювання» табл. 1.3 (колонка «R»).

18.6. Виконати операцію присвоювання значення змінній I (меню Debug\ Step Over). Значення I – до рядка «Після присвоювання» табл. 1.3 (колонка «I»).

18.7. Заповнену табл. 1.3 – до Звіту.

18.8. Продовжити виконання програми: меню Debug\ Continue. Закрити програму.

18.9. В редакторі коду видалити точку переривання, яку встановлено в п. 18.1. Для цього встановити курсор на даному рядку коду, Debug\ Toggle Breakpoint.

19. Зберегти проект. Остаточний текст програми – до Звіту. Закрити поточне рішення.

Зміст звіту

1. Номер, тема, мета ЛР.
2. Пункти порядку виконання ЛР: 3, 5.2, 5.4, 5.7.4, 5.8.2, 5.10, 5.12, 6.2, 7.2, 7.5, 8, 10, 12.3, 12.5, 15, 17.1, 17.3, 18.7, 19.
3. Висновки.

Контрольні запитання

1. Інтегроване середовище розробки. Контейнери VS.
2. Налагодження. Режими програми під час налагодження у VS.
3. Налагодження. Покрокова операція. Точка переривання

4. Консольний додаток та додаток Windows Forms.
5. Компіляція програм. Запуск з (без) налагодження.

ЛАБОРАТОРНА РОБОТА № 2
ДОСЛІДЖЕННЯ ЕФЕКТУ РЕАЛІЗАЦІЇ ЗАХИСТУ ВІД НЕКОРЕКТНИХ ДАНИХ

Мета роботи: дослідити величину надлишковості тексту програми, що внесена через захист від некоректних даних на прикладі задачі наближеного обчислення енергії сигналу; отримати уявлення про застосування операторів вибору і циклу та про захист від некоректних вхідних даних.

Основні теоретичні відомості

Під час розробки програми розробник, як правило, має можливість визначити різні шляхи виконання послідовності, які можуть пройти через програму, тобто *потік керування* (control flow) або *логіку керування* (flow of execution) [1], залежно від задачі, яку вирішує програма. Потік керування можна змінювати, зокрема, використовуючи інструкції для переходу, які визначають умову переходу, тобто *інструкції умовного переходу* (conditional jump instruction). Ці інструкції реалізуються за допомогою таких операторів:

- *Оператор вибору* (select statement) – зіставний оператор, який дає змогу задачі виклику чи викликаному завданню вибрати альтернативний план дій або очікувати [1]. *Приклад*: якщо в програмі перевірка певної умови повертає результат «виявлено помилку», то можна, наприклад, сповістити про це користувача і припинити виконання програми.
- *Оператор ітерації* (iteration statement) або *оператор циклу* (loop statement) – комбінований оператор, який містить механізм керування повторним виконанням вкладених у нього операторів [1]. *Приклад*: якщо потрібно на екран консольного додатку вивести 10 рядків з однаковим текстом, то замість десяти відповідних рядків у тексті програми можна написати цикл, який повторить цю дію необхідну кількість разів (у цьому прикладі – 10) і при цьому займе значно меншу кількість рядків тексту програми.

Послідовне, умовне (оператор вибору) й ітераційне (оператор циклу) керування застосовують у структурованому програмуванні. *Структуроване програмування* (structured programming) – спосіб побудови програм з використанням лише єрархічно розташованих конструкцій, кожна з яких має єдину точку входу та єдину точку виходу [1].

Вхідні дані, що введені користувачем до програми, можуть бути некоректними. *Приклади*:

– Очікується ціле число, а користувач ввів текст, або число з розділовим знаком цілої та дробової частини.
– Очікується дійсне число, в якому розділовим знаком цілої та дробової частини у поточному середовищі є кома (,), а користувач ввів дійсне число, в якому розділовий знак цілої та дробової частини – крапка (.).
– Очікується, що число a буде меншим за число b, але користувач ввів числа, за яких число b менше за число a.

В загальному випадку, оброблення некоректних даних зводиться до перевірки виконання певних умов та реалізації відповідної поведінки програми у разі наявності та відсутності виявлених некоректних даних. Як правило, це збільшує кількість рядків тексту програми порівняно з програмою, в якій такий захист від некоректних даних відсутній.

У даній ЛР досліджується величина надлишковості тексту програми, що внесена через захист від некоректних даних, на прикладі задачі наближеного обчислення енергії сигналу. При цьому використано класи та елементи керування Windows Forms, що подано нижче.

NumericUpDown – клас (простір імен System.Windows.Forms), представляє «регулятор» Windows (елемент керування «вгору-вниз»), який відображає числові значення [4]. Деякі властивості:

– Value – властивість (типу decimal), повертає чи задає значення регулятору [4].
– Maximum – властивість (типу decimal), повертає чи задає максимально допустиме значення регулятору [4].
– Minimum – властивість (типу decimal), повертає чи задає мінімально допустиме значення регулятору [4].

Деякі складові елементу керування TextBox:

– ScrollBars – властивість (типу ScrollBars), повертає чи задає значення, що визначає, які смуги прокрутки присутні в TextBox [4].
– Multiline – властивість (типу bool), повертає чи задає значення, що визначає, чи є даний елемент керування *багаторядковим* текстовим полем [4].
– AppendText – метод, додає текст у кінець існуючого тексту в текстовому полі [4].

Math – клас (простір імен System), надає константи та статичні методи для тригонометричних, логарифмічних й інших загальних математичних функцій [4].

Convert.ToInt32 – метод (static), перетворює вказане значення на еквівалентне 32-бітне ціле зі знаком [4].

Деякі методи класу Double (простір імен System):

– ToString – метод, перетворює числове значення екземпляра на еквівалентне йому рядкове представлення з використанням указаних ві-

домостей щодо особливостей форматування для даної мови та регіональних стандартів [4]. Специфікатор формату `f` – представлення з фіксованою точністю (задається довжина дробової частини).

– `Double.TryParse` – метод (`static`, типу `bool`), перетворює рядкове представлення числа на еквівалентне йому число подвійної точності з рухомою комою [4]. Повертає значення, що вказує, чи виконано перетворення *успішно*.

`Environment.NewLine` – властивість (`static`, типу `string`), повертає рядок, який в даному середовищі позначає початок нового рядку [4].

`MessageBox` – клас (простір імен `System.Windows.Forms`), відображає вікно повідомлення з текстом для користувача [4]. Деякі складові:

– `MessageBox.Show` – метод (`static`), відображає вікно повідомлення із заданим текстом [4].

Об'єкт моделювання

Енергія, яку виділено в опорі 1 Ом на відрізку $t \in (a,b)$ (час, с) сигналом $u(t)$ (електрична напруга, В) [2]:

$$E = \int_a^b u^2(t)\,dt \ \left[\text{В}^2\times\text{с}\right]. \tag{2.1}$$

В даній ЛР обчислюється наближене значення інтегралу (2.1) за допомогою квадратурної формули прямокутників (середніх прямокутників). Для пояснення методу середніх прямокутників (далі – СП) розглянемо визначений інтеграл

$$S = \int_a^b u(t)\,dt \tag{2.2}$$

від неперервної на відрізку (a,b) функції $u(t)$. Квадратурна формула, що визначається вагами q_i та вузлами t_i, це приблизна рівність

$$S \approx \sum_{i=1}^{n} q_i \cdot u(t_i),$$

де q_i – деякі числа, t_i – деякі точки відрізку (a,b).

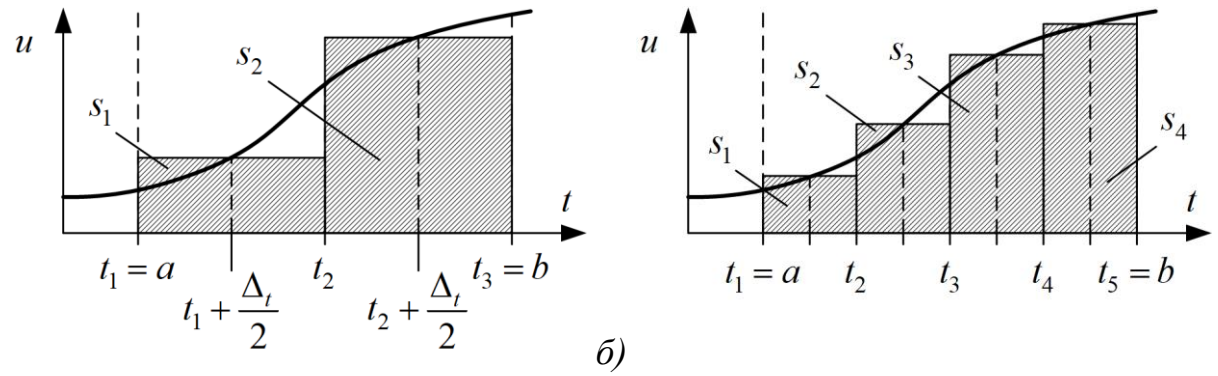
Для наближеного обчислення S (тобто площі фігури, обмеженої кривою $u(t)$ та віссю абсцис) необхідно розбити відрізок (a,b) на n частин (інтервалів) з кроком $\Delta_t = (b-a)/n$. На кожному інтервалі замінити функцію $u(t)$ на поліном $p(t)$, інтеграл від якого обчислюється легко. Обчислити інтеграл від $p(t)$ на кожному i-му інтервалі (t_i, t_{i+1}), тобто

$$s_i = \int_{t_i}^{t_{i+1}} p(t)\,dt\,.$$

Шуканий інтеграл наближено дорівнює

$$S \approx \sum_{i=1}^{n} s_i = S_n .$$

В методі СП $p(t) = u(t_i + \Delta_t/2)$, тобто це пряма (паралельна осі абсцис), яка перетинає $u(t)$ в точці посередині інтервалу (t_i, t_{i+1}). Тоді $s_i = u(t_i + \Delta_t/2) \cdot \Delta_t$, тобто це площа прямокутника висотою $u(t_i + \Delta_t/2)$ та шириною Δ_t. Графічну інтерпретацію інтегрування методом СП подано на рис. 2.1.



а) *б)*

Рис. 2.1. Чисельне інтегрування функції $u(t)$ методом середніх прямокутників, якщо відрізок (a,b) розділено на *а)* 2 інтервали; *б)* 4 інтервали

Оцінка похибки обчислення значення інтегралу при $2n$ інтервалах за правилом Рунге складає

$$S - S_{2n} \approx |S_{2n} - S_n|/3 \qquad (2.3)$$

і процес обчислення закінчується в разі виконання умови $|S_{2n} - S_n|/3 < \varepsilon$, де ε – задана точність. В реальних задачах значення S є невідомим або таким, яке досить складно обчислити. Саме тому і застосовують методи, подібні до методу СП.

Приклад. Припустимо, що користувачу потрібно наближено обчислити значення визначеного інтегралу (2.2) (тобто «площу під кривою») методом СП із бажаною точністю $\varepsilon = 0.01$. Також припустимо, що результат обчислення методом СП за умови розбиття відрізку (a,b) на два інтервали (див. приклад на рис. 2.1 *а*) дорівнює $S_2 \approx 0.149$. Для того, щоб оцінити величину похибки обчислення значення (2.2) за правилом Рунге (2.3), потрібно обчислити значення (2.2) методом СП за умови розбиття відрізку (a,b) на вдвічі більшу кількість інтервалів (див. приклад на рис. 2.1 *б*). Тобто, якщо два ін-

тервали прийняти за n, то $2n=4$. Припустимо, що $S_{2n}=S_4\approx 0.09$. Обчислимо (2.3):

$$S-S_4\approx|S_4-S_2|/3=|0.09-0.149|/3\approx 0.0197,$$

що *не* менше за $\varepsilon=0.01$, отже бажаної точності не досягнуто.

Збільшимо кількість інтервалів, на яку розбито відрізок (a,b) вдвічі, і перевіримо, чи буде досягнуто бажану точність. Для цього приймемо чотири інтервали за n. Тоді потрібно обчислити значення (2.2) методом СП для $2n=8$ інтервалів. Припустимо, що $S_{2n}=S_8\approx 0.094$. Обчислимо (2.3):

$$S-S_8\approx|S_8-S_4|/3=|0.094-0.09|/3\approx 0.0013,$$

що менше за $\varepsilon=0.01$, отже бажану точність досягнуто і продовжувати збільшувати кількість інтервалів, на яку розбито відрізок (a,b), не потрібно.

Постановка задачі

Розробити додаток Windows Forms, який

- дозволяє користувачу задавати границі відрізку (a,b) та кількість інтервалів n;
- виводить на екран: наближене значення енергії E (2.1) сигналу $u(t)$ на відрізку $t\in(a,b)$ методом СП для кількості інтервалів n.

Сигнал $u(t)$ задано гармонічною функцією виду

$$u(t)=U\cdot\sin(2\pi ft+\varphi), \tag{2.4}$$

де U – амплітуда, В,
f – частота, Гц,
t – час, с,
φ – фаза, рад.

Вид функції sin або cos визначено у колонці «Функція» дод. 1 відповідно варіанту. Аргументи функції $u(t)$ визначено у колонках «Амплітуда, В», «Частота, Гц», «Фаза, рад.» дод. 1 відповідно варіанту.

Для всіх номерів варіантів прийняти значення: $a=0$, $b=1/f$.

Порядок виконання лабораторної роботи

1. В Робочій папці створити папку для поточної ЛР. Отримати у викладача папку з рішенням `Lr2` та проектом `SignalEnergy`, що містить заготовку форми для додатку. Перемістити отримані файли до Робочої папки.

2. Ознайомитися з формою проекту `SignalEnergy` (рис. 2.2) в режимі конструктора. Для цього: відкрити проект `SignalEnergy` (розширення файлу «CSPROJ») або відповідне рішення (розширення файлу «SLN») у VS; у вікні `Solution Explorer` ПКМ по `Form1.cs`, у контекстному меню ЛКМ по

View Designer; почергово ЛКМ по кожному компоненту на формі, під час цього у вікні Properties, на вкладці Properties () переглядати значення властивості Name.

3. В режимі конструктора доповнити форму проекту SignalEnergy елементами NumericUpDown та TextBox (*3* та *5* на рис. 2.2). Задати властивості (вікно Properties, вкладка Properties ) відповідно табл. 2.1.

4. До Звіту занести:

– ескіз (або скріншот) форми (подібно до рис. 2.2);
– перелік відповідностей «елемент керування на ескізі (скріншоті) форми» – «властивість Name цього елемента» (подібно до рис. 2.2).

Примітки: на рис. 2.2 подано перелік відповідностей «елемент керування» – «ім'я *класу* цього елемента», а не «*властивість* Name цього елемента».

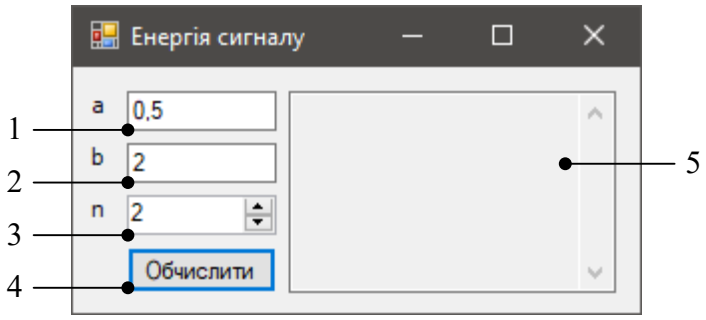


Рис. 2.2. Вікно проекту SignalEnergy з елементами керування:
1, *2*, *5* – TextBox; *3* – NumericUpDown; *4* – Button

Таблиця 2.1

Елементи керування на формі проекту SignalEnergy

№ на формі	Назва	Властивість	Значення
3	numericUpDown1	Name	nmrcUpDnN
5	textBox1	Name	txtBxOut
		ScrollBars	Vertical
		ReadOnly	True
		Multiline	True

5. Створити обробники подій для компонентів форми відповідно табл. 2.2. Для цього для кожного елементу з тих, які вказано в колонці «Компонент» табл. 2.2: в режимі конструктора ЛКМ по компоненту, у вікні Properties, на вкладці Events (), знайти потрібну подію (задано в колонці «Подія» табл. 2.2), подвійний ЛКМ у полі напроти відповідної події.

6. Доповнити тіло обробника Form1_Load:

6.1. Властивості `Text` елемента `txtBxA` присвоїти значення початку a відрізку (a,b) обчислення енергії сигналу, відповідно розділу «Постановка задачі». Аналогічно – властивості `Text` елемента `txtBxB` присвоїти значення кінця b відрізку (a,b).

Примітки: типом даних властивості `TextBox.Text` є `string`. Тобто для присвоєння числового значення потрібно виконати відповідне перетворення типів.

6.2. Присвоїти значення `2M` (літерал типу `decimal`) властивості `Value` елемента `nmrcUpDnN`.

Призначення: властивість `Value` елемента `nmrcUpDnN` містить значення кількості інтервалів у відрізку (a,b).

6.3. Присвоїти значення `1e3M` (літерал типу `decimal`) властивості `Maximum` елемента `nmrcUpDnN`.

Призначення: обмежити максимальне значення, яке можна задати властивості `Value` елемента `nmrcUpDnN`, до 1000 включно.

Таблиця 2.2

Події для компонентів форми проекту `SignalEnergy`

Компонент	Подія	Ім'я обробника	Призначення
`btnOK`	`Click`	`btnOK_Click`	виконання обчислень
`Form1`	`Load`	`Form1_Load`	налаштування форми під час завантаження форми

7. Доповнити тіло обробника `btnOK_Click`:

7.1. Декларувати змінні `a` та `b` типу `double`.

Призначення: містять границі відрізку (a,b) для обчислення енергії (2.1) сигналу (2.4).

7.2. Присвоїти змінній `a` значення властивості `Text` елемента `txtBxA`. Аналогічно – присвоїти змінній `b` значення властивості `Text` елемента `txtBxB`.

Примітки: оскільки типом даних властивості `TextBox.Text` є `string`, а змінних `a` та `b` – `double`, то потрібно виконати відповідне перетворення типів.

7.3. Декларувати змінну `n` типу `int`. Присвоїти змінній `n` значення властивості `Value` елемента `nmrcUpDnN`.

Призначення: містить кількість інтервалів у відрізку (a,b). *Примітки*: оскільки типом даних властивості `NumericUpDown.Value` є `decimal`, а змінної `n` – `int`, то потрібно виконати відповідне перетворення типів.

7.4. Декларувати змінну `E` типу `double`. Ініціювати значенням 0.

Призначення: містить шукане наближене значення енергії сигналу (2.4).

7.5. Декларувати змінну `deltaT` типу `double`. Присвоїти їй вираз

```
(b - a) / n
```

Призначення: містить крок Δ_t за абсцисою (часом).

7.6. Створити цикл `for`. Елементам цього оператора `for` (секціям «ініціалізатор», «умова» та «ітератор») відповідно задати

```
double t = a + deltaT / 2; t <= b; t = t + deltaT
```

Призначення: під час ітерацій циклу змінювати значення моменту часу t починаючи з $t = a + \Delta_t/2$ із кроком Δ_t поки $t \le b$. Тобто послідовно задавати моменти часу для обчислення відповідних значень $u^2(t)$, що дозволить наближено обчислити значення енергії сигналу $u(t)$ методом СП.

7.7. У вбудованому операторі (embedded statement) циклу `for` змінній `E` присвоїти вираз, який є сумою таких складових:

- самої змінної `E`
- та значення сигналу (частина виразу (2.4) за варіантом дод. 1, яку розміщено праворуч від знаку «дорівнює»), що піднесено до *квадрату* (див. (2.1)), в момент часу `t`.

Примітки: для запису математичних функцій рекомендовано використовувати елементи класу `Math`:

- `Math.Sin(x)` та `Math.Cos(x)` – методи (`static`), які повертають значення (типу `double`) синусу та косинусу кута `x` (типу `double`), що заданий в радіанах, відповідно.
- `Math.Pow(x, y)` – метод (`static`), який повертає значення (типу `double`), що дорівнює значенню `x` (типу `double`), яке піднесено до степеня `y` (типу `double`).
- `Math.PI` – поле (`const`), яке представляє відношення довжини кола до його діаметру, що визначено константою π.

7.8. *Після* тіла оператора `for` (після символу `}`) присвоїти змінній `E` вираз

```
E * deltaT
```

Призначення: обчислює шукане наближене значення енергії сигналу (2.4) як площу прямокутника висотою, рівною сумі значень $u^2(t)$ в моментах часу, що визначені в заголовку циклу `for`, та шириною, рівною Δ_t.

7.9. В наступному рядку ввести код

```
txtBxOut.AppendText(
    "К-сть інтервалів: " + n + Environment.NewLine +
    "Енергія сигналу: " + E.ToString("f5") +
    Environment.NewLine);
```

Призначення: доповнити текст, який вже міститься у властивості `Text` елемента `txtBxOut`, новим текстом (кількість інтервалів `n`, для якої виконано

обчислення, значення енергії сигналу `E` для цієї кількості інтервалів). Причому значення `E` представлено з обмеженням до 5 знаків після десяткового роздільника (задано специфікатором формату **“`f5`”**).

8. Обчислити кількість рядків у тілі обробника `btnOK_Click` (порожні рядки та коментарі не враховувати). Результат позначити як L_1 і занести до Звіту.

9. За допомогою програми комп’ютерної математики (чи власноруч) обчислити значення (2.1) для сигналу (2.4) за варіантом (дод. 1). Результат прийняти за еталонний, позначити як $E_{\text{ет.}}$ і занести до Звіту.

10. За допомогою `SignalEnergy` заповнити колонку «Результат програми S_n» табл. 2.3. Результат – до Звіту. Закрити програму.

Таблиця 2.3

№ з/п	Кількість інтервалів n	Результат програми S_n	Похибка обчислення
1	2		
2	4		
3	8		
4	16		
5	32		
6	64		
7	128		

До колонки «Похибка обчислення» табл. 2.3 занести похибку обчислення відповідно правій частині виразу (2.3) починаючи з рядка № 2 (для цього рядка кількість інтервалів дорівнює 4).

Приклад: вираз похибки обчислення для рядка № 2 табл. 2.3 дорівнює $\left|S_4 - S_2\right|/3$; для рядка № 3 – дорівнює $\left|S_8 - S_4\right|/3$.

11. На основі даних табл. 2.3 до Звіту занести:

- графік залежності результату обчислення енергії сигналу (2.4) (колонка «Результат програми S_n» табл. 2.3) від кількості інтервалів (колонка «Кількість інтервалів n» табл. 2.3) методом СП;
- на тому ж графіку показати рівень $E_{\text{ет.}}$.

12. Дослідити виконання циклу обчислення енергії сигналу:

12.1. Встановити точку переривання на першому операторі у тілі циклу `for` (той, що накопичує суму у змінній `E`).

12.2. Скомпілювати та запустити проект в режимі налагодження, задати у вікні запущеного додатку кількість інтервалів рівною 4, натиснути кнопку `Обчислити`.

12.3. Виконуючи програму в режимі *покрокових операцій* заповнити табл. 2.4 значеннями змінних *після* кожної ітерації циклу, тобто досягнення фігурної дужки, що закривається }. Результат – до Звіту. Закрити програму (зупинити налагодження). Видалити точку переривання.

13. Дослідити реакцію програми при некоректних вхідних даних:

13.1. Скомпілювати та запустити проект в режимі налагодження. До будь-якого текстового поля у вікні додатку ввести довільне *нечислове* значення, натиснути кнопку `Обчислити`. Вміст рядку, в якому виникла помилка, та опис помилки – до Звіту. Зупинити налагодження.

13.2. Скомпілювати та запустити проект в режимі налагодження. У вікні запущеного додатку поміняти значення границь a та b місцями (скопіювати значення з `txtBxB` та вставити замість значення `txtBxA`, а у `txtBxB` ввести значення, яке було раніше у `txtBxA`), натиснути кнопку `Обчислити`. Порівняти результат з $E_{ет.}$. Результат порівняння разом з відповідним поясненням – до Звіту. Закрити програму.

13.3. Скомпілювати та запустити проект в режимі налагодження. У вікні запущеного додатку задати кількість інтервалів рівною 0 (тобто $n=0$), натиснути кнопку `Обчислити`. Результат – до Звіту. Закрити програму.

Таблиця 2.4

№ ітерації	a	b	deltaT	t	E
1					
2					
3					
4					

14. Реалізувати в програмі захист від некоректних даних на формі:

14.1. Замінити рядки читання з `TextBox` кодом (розмістити *після* декларації змінних `a` та `b`)

```
// Прочитати з форми значення границь відрізку (a,b).
if (!double.TryParse(txtBxA.Text, out a) ||
    !double.TryParse(txtBxB.Text, out b))
{   MessageBox.Show("a та b повинні бути дійсними числами");
    return;
}
```

Призначення: якщо користувач на формі додатку хоча б до одного елементу керування з `txtBxA` або `txtBxB` ввів значення, яке не може бути перетворене на дійсне число (наприклад, текст або число з використанням символу десяткового роздільника, що відрізняється від прийнятого у середовищі, в якому виконується додаток), то вивести повідомлення про помилку і вийти з поточного методу. Тобто опрацювати ситуацію, коли спроба перетворити ря-

док, який міститься у властивості `Text` елемента `txtBxA` або `txtBxB`, на тип даних `double` буде неуспішною.

```
// Перевірити границі відрізку (a,b) на узгодженість.
if (a >= b)
{   MessageBox.Show("a повинно бути меншим за b");
    return;
}
```

Призначення: якщо значення початку a відрізку не менше за значення кінця b відрізку, то вивести повідомлення про помилку і вийти з поточного методу.

14.2. Після рядку читання з `NumericUpDown` доповнити кодом

```
// Перевірити кількість інтервалів у відрізку (a,b).
if (n < 1)
{   MessageBox.Show(
        "Кількість інтервалів повинна бути більша нуля");
    return;
}
```

Призначення: якщо кількість інтервалів, на які потрібно розбити відрізок (a,b) менша за 1, то вивести повідомлення про помилку і вийти з поточного методу.

15. Виконати п. 13. Результат – до Звіту.

16. Обчислити кількість рядків у тілі обробника `btnOK_Click` (порожні рядки та коментарі не враховувати). Результат позначити як L_2 і занести до Звіту. Обчислити частину тексту програми, яку займає захист від некоректних даних на формі додатку, відносно загальної кількості рядків у тілі обробника: $(L_2 - L_1)/L_2$. Результат – до Звіту.

17. Зберегти проект. Остаточний текст програми – до Звіту. Закрити поточне рішення.

Зміст звіту

1. Номер, тема, мета ЛР.
2. Пункти порядку виконання ЛР: 4, 8 – 11, 12.3, 13.1, 13.2, 13.3, 15 – 17.
3. Висновки.

Контрольні запитання

1. Потік керування. Оператор вибору.
2. Потік керування. Оператор циклу.
3. Читання значень з елементів керування `TextBox` та `NumericUpDown`.
4. Реалізація захисту від некоректних вхідних даних.

5. Метод обчислення наближеного значення визначеного інтегралу методом середніх прямокутників. Оцінка похибки обчислення за правилом Рунге.

ЛАБОРАТОРНА РОБОТА № 3
ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МЕТОДІВ СОРТУВАННЯ

Мета роботи: дослідити ефективність методів сортування одномірного масиву на прикладі задачі сортування масиву методами «вставок» та Шелла; отримати уявлення про методи класу та опрацювання винятків.

Основні теоретичні відомості

Одним з фундаментальних понять у розробці програмного забезпечення є *структура даних* (data structure – фізичні чи логікові зв'язки між блоками даних та самими даними). Різні структури даних дозволяють ефективно зберігати *дані* (переосмислене подання інформації формалізованим способом, придатним для обміну, інтерпретації чи опрацювання [1]) та маніпулювати даними. Серед найбільш відомих структур даних є *масив* (array), який дозволяє працювати з колекцією елементів даних, що *подібні* між собою. *Масив C#* – структура даних, яка містить сукупність *однотипних* змінних (елементи масиву), до яких можна отримати доступ через *індекси* [4]. *Індекс* (index) – ціле число, яке визначає позицію елемента даних у послідовності елементів даних [1].

Якщо для чіткого визначення елемента масиву достатньо одного індексу, то такий масив називається *одномірним* (single-dimensional). *Приклади* застосування одномірних масивів – зберігання:

- рядка тексту (елементами масиву є символи);
- значень рівня (амплітуди) цифрового аудіосигналу (елементами масиву є числа).

Т.ч. за допомогою індексу можна звернутись (для читання чи запису), наприклад, до елемента масиву, який розташовано п'ятим з початку. Індекс початкового елемента масиву C# дорівнює нулю, тому індекс елемента, який розташовано п'ятим з початку, дорівнює чотирьом.

Поширеною операцією з масивом є його *сортування* (розподіл елементів на групи відповідно до зазначених критеріїв, *не обов'язково впорядковуючи* елементи в кожній групі [1]). *Упорядкувати* (order) означає розмістити елементи у місці розташування відповідно до зазначених правил [1]. Наприклад, сортувати масив цілих чисел в порядку *не*спадання їх значень (у даному випадку – упорядкувати). *Приклад* використання впорядкованого масиву: в такому масиві можна значно швидше (використовуючи бінарний пошук) знайти потрібне значення ніж у невпорядкованому масиві. В загальному випадку, для невпорядкованого масиву потрібно переглянути *всі* елементи.

Існує багато алгоритмів сортування (впорядкування) масиву, наприклад, «обміном», «вибором», «вставками» (дод. 2), Шелла (дод. 3) тощо [3]. Вони відрізняються тривалістю виконування (execution duration), обсягом зарезервованої пам'яті, способом переміщення елементів даних або їх атрибутів тощо. Досить часто, за інших рівних умов, розробники надають перевагу тому алгоритму, який має меншу тривалість виконування.

Для спрощення читання, розробки та підтримання програми процедурною мовою програмування розробники використовують *модулі* (module) – частини програми, розроблені як дискретні чи ідентифіковані стосовно таких дій, як компіляція, зв'язування чи виконання, і які можуть взаємодіяти з іншими програмами або частинами програм [1]. Прикладом модуля є підпрограма (subprogram). В об'єктно-орієнтованій мові програмування в якості підпрограм, зокрема, використовуються *методи* (method) – операції, які об'єкт виконує після отримання *повідомлення* (запит до об'єкта на виконання однієї з його операцій) [1].

Досить часто для виконання методу йому потрібно передати фактичні параметри (аргументи) з коду, який викликає цей метод. Аргументи передаються до методу шляхом їх асоціювання з формальними параметрами методу. Способи передачі аргументів до параметрів методу в C#:

- за значенням (by value) – передається копія змінної;
- за посиланням (by reference) – передається доступ (посилання) до змінної.

Під час виконання програми, може виникнути стан, що може спричинити відхилення послідовності виконання від звичайної. Наприклад, внаслідок спроби доступу до файлу, якого не існує, або цілочислового ділення на нуль. Такий стан, для якого є засоби визначення, підвищення, розпізнавання, ігнорування чи опрацювання, називається *виняток* (exception) [1]. Використання винятків у C#, зокрема, дозволяє спростити опрацювання помилок: виняток, який утворено в одній частині програми, може бути перехоплено в іншій частині програми «перестрибнувши» одразу через декілька підпрограм (методів), які були почергово викликані.

У даній ЛР розглядаються алгоритми сортування «вставками» та Шелла, а також досліджується їх ефективність. Критерій ефективності – найменша кількість операцій, виконаних програмою. При цьому використано класи та елементи керування `Windows Forms`, що подано нижче.

`System.Array` – клас, надає методи для створення, зміни, пошуку та сортування масивів [4]. Деякі елементи:

- `Length` – властивість (типу `int`), отримує загальну кількість елементів у всіх вимірах `Array` [4].
- `Clone` – метод, створює *поверхневу* (shallow, неглибоку, неповну) копію масиву [4].

`Random` – клас (простір імен `System`), надає генератор псевдовипадкових чисел (пристрій, що видає послідовність чисел, які відповідають певним статистичним критеріям випадковості) [4]. Деякі елементи:

– `Next(max)` – метод, повертає випадкове число $x \geq 0$ типу `int`. $x \in [0, \max)$ [4].

`Exception` – клас (простір імен `System`), представляє помилки, які виникають під час виконання додатку [4]. Деякі елементи:

– `Message` – властивість (типу `string`), отримує повідомлення, яке описує даний виняток [4].

Деякі складові елементу керування `TextBox`:

– `Clear` – метод, очищає весь текст з елементу керування `TextBox` [4].

`ComboBox` – клас (простір імен `System.Windows.Forms`), надає елемент керування полем зі списком [4]. Деякі складові:

– `Items.Add()` – метод, додає елемент до списку елементів (властивість `Items`) `ComboBox` [4].
– `SelectedIndex` – властивість (типу `int`), повертає чи задає індекс, який вказує на поточний обраний елемент [4].

Постановка задачі

Розробити додаток Windows Forms, який:

– дозволяє користувачу:
  ◦ задавати довжину `n` одномірного масиву `a` цілих чисел;
  ◦ обирати спосіб ініціювання елементів масиву цілими числами – за зростанням, за спаданням, або псевдовипадковими цілими числами з інтервалу $[0,100)$;
  ◦ обирати метод сортування – «вставками», або Шелла.
– виводить на екран:
  ◦ масив перед сортуванням `unsorted`;
  ◦ масив після сортування `sorted`;
  ◦ кількість виконаних операцій `movesCount` – збільшує своє значення на одиницю щоразу, коли під час сортування елементу масиву присвоєно якесь значення.

Порядок виконання лабораторної роботи

1. В Робочій папці створити папку для поточної ЛР. Отримати у викладача папку з рішенням `Lr3` та проектом `ArraySort`, що містить заготовку форми для додатку. Перемістити отримані файли до Робочої папки.

2. Ознайомитися з формою проекту `ArraySort` (рис. 3.1) в режимі конструктора. Для цього: відкрити проект `ArraySort` (розширення файлу «CSPROJ») або відповідне рішення (розширення файлу «SLN») у VS; у вікні `Solution Explorer` ПКМ по `Form1.cs`, у контекстному меню ЛКМ по

View Designer; почергово ЛКМ по кожному компоненту на формі, під час цього у вікні Properties, на вкладці Properties () переглядати значення властивості Name.

3. В режимі конструктора доповнити форму проекту ArraySort елементами ComboBox (*2* та *3* на рис. 3.1). Задати властивості (вікно Properties, вкладка Properties ) відповідно табл. 3.1.

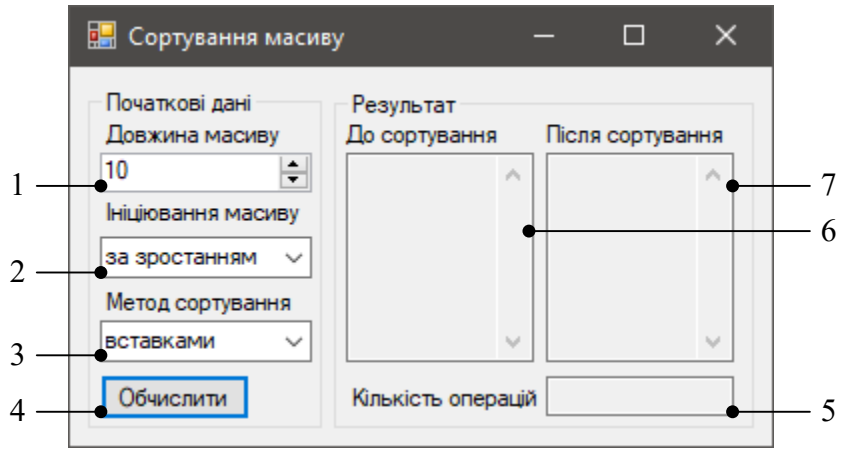


Рис. 3.1. Вікно проекту ArraySort з елементами керування:
1 – NumericUpDown; *2*, *3* – ComboBox; *4* – Button; *5–7* – TextBox

Таблиця 3.1

Елементи керування на формі проекту ArraySort

№ на формі	Назва	Властивість	Значення
2	comboBox1	Name	cmbBxArrayInitialization
3	comboBox2	Name	cmbBxSortMethod

4. До Звіту занести:
- ескіз (або скріншот) форми (подібно до рис. 3.1);
- перелік відповідностей «елемент керування на ескізі (скріншоті) форми» – «властивість Name цього елемента» (подібно до рис. 3.1).

Примітки: на рис. 3.1 подано перелік відповідностей «елемент керування» – «ім'я *класу* цього елемента», а не «*властивість* Name цього елемента».

5. Отримати у викладача файл CodeFile3.cs. Перемістити його до папки проекту ArraySort та відкрити.

6. Доповнити тіло класу Form1 методами, визначеними у файлі CodeFile3.cs:
- InsertionSort(int[] a, int step). *Призначення*: сортувати елементи масиву a, які розміщено через інтервал step, за зростанням методом «вставок».

– `InsertionSort(int[] a)`. *Призначення*: сортувати масив `a` за зростанням методом «вставок».
– `ShellSort(int[] a)`. *Призначення*: сортувати масив `a` за зростанням методом Шелла.
– `InputData(out int[] a)`. *Призначення*: прочитати дані з форми додатку та повернути масив `a` залежно від налаштувань на формі.
– `SortArray(int[] a)`. *Призначення*: сортувати масив `a` за зростанням залежно від налаштувань на формі додатку.
– `OutResult(int[] unsorted, int[] sorted)`. *Призначення*: вивести результат (елементи масиву перед та після сортування, тобто масиви `unsorted` та `sorted`, відповідно) на форму додатку.

Закрити файл `CodeFile3.cs`. Назви методів, доданих до тіла класу `Form1` та їх призначення (з коментарів у тексті програми) – до Звіту.

7. Доповнити тіло методу `InputData` (в операторі `switch`):

7.1. В списку операторів секції `case 0` ініціювати масив цілими числами, значення яких розміщено *за зростанням*. Цього можна досягти у різний спосіб. Наприклад, для присвоєння кожному елементу масиву значення, що дорівнює індексу цього елемента:

7.1.1. Створити цикл, в якому змінній `i` (індекс елемента масиву) послідовно присвоюється значення, починаючи від 0, завершуючи останнім індексом в масиві (`a.Length - 1`) включно, з кроком 1.

7.1.2. В тілі циклу (п. 7.1.1) `i`-му елементу масиву `a` присвоїти значення `i`.

Примітки: в списку елемента `cmbBxArrayInitialization` елемент з індексом 0 відповідає рядку випадного списку «`за зростанням`».

7.2. В списку операторів секції `case 1` ініціювати масив цілими числами, значення яких розміщено *за спаданням*. Цього можна досягти у різний спосіб. Наприклад, для присвоєння останньому елементу масиву (його індекс дорівнює `a.Length - 1`) значення 0, передостанньому елементу – значення 1 і т.д.:

7.2.1. Виконати п. 7.1.1.

7.2.2. В тілі циклу (п. 7.2.1) `i`-му елементу масиву `a` присвоїти вираз

```
-i + a.Length - 1
```

Примітки: в списку елемента `cmbBxArrayInitialization` елемент з індексом 1 відповідає рядку випадного списку «`за спаданням`».

7.3. В списку операторів секції `case 2` ініціювати масив цілими числами, значення яких є *псевдовипадковими*. Наприклад, для ініціювання масиву цілими псевдовипадковими числами, зі значеннями в інтервалі $[0,100)$, де верхня границя інтервалу є відкритою (тобто ніколи не буде представлена у створюваному масиві):

7.3.1. Декларувати змінну `random` класу `Random`, присвоїти їй вираз

```
new Random()
```

7.3.2. Виконати п. 7.1.1.

7.3.3. В тілі циклу (п. 7.3.2) `i`-му елементу масиву `a` присвоїти вираз, який являється викликом методу `Next(100)` змінної `random`.

Примітки: в списку елемента `cmbBxArrayInitialization` елемент з індексом 2 відповідає рядку випадного списку «`випадково`».

8. Доповнити тіло методу `SortArray` (в операторі `switch`):

8.1. В списку операторів секції `case 0` сортувати масив методом «вставок»: викликати метод `InsertionSort` та передати його параметру аргумент `a`.

Примітки: в списку елемента `cmbBxSortMethod` елемент з індексом 0 відповідає рядку випадного списку «`вставками`».

8.2. В списку операторів секції `case 1` сортувати масив методом Шелла: викликати метод `ShellSort` та передати його параметру аргумент `a`.

Примітки: в списку елемента `cmbBxSortMethod` елемент з індексом 1 відповідає рядку випадного списку «`Шелла`».

9. Доповнити тіло обробника `btnOK_Click`:

9.1. Декларувати одномірні масиви `unsorted` та `sorted`, типом елементів яких є `int`.

Призначення: містять масиви цілих чисел *перед* сортування та *після* сортування, відповідно.

9.2. Ввести (прочитати) дані з форми додатку: в блоці операторів `try` викликати метод `InputData` та передати його параметру аргумент `unsorted` з модифікатором `out`.

9.3. Сортувати (впорядкувати) масив: в блоці операторів `try` після поверхневого копіювання (shallow copy) масиву викликати метод `SortArray` та передати його параметру аргумент `sorted`.

9.4. Вивести результат на форму: після оператора `try` (після останнього блоку операторів `catch`) викликати метод `OutResult` та передати його списку параметрів аргументи `unsorted`, `sorted`.

10. Оцінити кількість виконаних операцій (переміщень елементів масиву), виконаних під час сортування масиву методами «вставок» та Шелла:

10.1. Скомпілювати та запустити проект в режимі налагодження.

10.2. Заповнити табл. 3.2 (до Звіту).

10.3. Закрити програму.

10.4. На основі даних табл. 3.2 до Звіту занести:

- графіки залежностей кількості операцій (`movesCount`) від довжини масиву (`n`) для розглянутих методів сортування та способу ініціювання елементів масиву (табл. 3.2);
- порівняння ефективності методів сортування «вставками» та Шелла.

11. Дослідити реакцію програми на некоректні вхідні значення:

11.1. Встановити точку переривання в обробнику `btnOK_Click` на першому непорожньому операторі в блоці операторів `try`.

11.2. Скомпілювати та запустити проект в режимі налагодження. Задати довжину масиву `0`, натиснути кнопку `Обчислити`. Перемістити фокус введення (перейти) до редактора коду (ЛКМ по редактору коду), якщо цього не виконано автоматично.

11.3. Виконувати оператори *із заходом* в підпрограми (меню `Debug\ Step Into`) до досягнення фігурної дужки, що закривається `}`, обробника `btnOK_Click`. Під час виконання покрокових операцій описувати послідовність виконання у Звіті. Зупинити налагодження. Видалити точку переривання.

Таблиця 3.2

Кількість операцій

Опис масиву	Метод сортування							
	Вставками				Шелла			
Довжина масиву	50	100	150	200	50	100	150	200
Елементи ініційовано за зростанням								
Елементи ініційовано за спаданням								
Елементи ініційовано випадковими значеннями*								

* *Рядок «Елементи ініційовано випадковими значеннями» табл. 3.2 для кожної з довжин масиву заповнювати середнім значенням, яке отримано на основі 5 значень, повернутих програмою.*

12. Зберегти проект. Остаточний текст програми – до Звіту. Закрити поточне рішення.

Зміст звіту

1. Номер, тема, мета ЛР.
2. Пункти порядку виконання ЛР: 4, 6, 10.2, 10.4, 11.3, 12.
3. Висновки.

Контрольні запитання

1. Одномірний масив. Його декларування та ініціювання в C#.
2. Сортування масиву. Алгоритм сортування масиву «вставками».
3. Сортування масиву. Алгоритм сортування масиву Шелла.
4. Методи. Способи передачі аргументів до параметрів методу.
5. Винятки. Опрацювання та генерація винятків.

ЛАБОРАТОРНА РОБОТА № 4
ОЦІНКА СКЛАДНОСТІ АЛГОРИТМУ ОБРОБЛЕННЯ ДВОМІРНОГО МАСИВУ

Мета роботи: дослідити залежність кількості виконаних операцій від кількості елементів двомірного масиву на прикладі задачі отримання вибірки з двомірного масиву; отримати уявлення про оброблення двомірного масиву та оцінку часової складності алгоритму.

Основні теоретичні відомості

Вимірювати тривалість виконування алгоритму в одиницях виміру часу (секунди) не завжди доцільно, оскільки один і той самий алгоритм матиме різну тривалість виконування на комп'ютерах з різною обчислювальною потужністю. Для порівняння тривалості виконування алгоритмів незалежно від характеристик комп'ютера використовується *кількість операцій* (чітко визначених дій), які виконано відповідно алгоритму. На практиці використовується наближене значення – *апроксимація* кількості операцій.

Як правило, наближене значення *кількості* операцій, що виконано відповідно алгоритму (наприклад, для масиву з 10 елементами), не є дуже важливим. Важливою є оцінка, з якою *швидкістю зростає* наближене значення кількості операцій, що виконано відповідно алгоритму, залежно від зростання n – кількості елементів масиву, або більш загально – складності задачі (розміру задачі). Тобто – *часова складність алгоритму*, в якій під «часовою» розуміється *кількість* операцій (а не тривалість в секундах) залежно від n. Для вказаної оцінки часто використовують такий перелік функцій [3]:

- *константа* – кількість операцій дорівнює одиниці, тобто не залежить від n;
- *логарифмічна* – кількість операцій залежить від n за виразом $\log n$;
- *лінійна* – кількість операцій залежить від n за виразом n;
- *лінеарифмічна* – кількість операцій залежить від n за виразом $n\log n$;
- *квадратична* – кількість операцій залежить від n за виразом n^2;
- *експоненційна* – кількість операцій залежить від n за виразом c^n, де c – додатна константа.

Поведінка функції (див., наприклад, перелік вище), яка описує часову складність алгоритму в границі при збільшенні n називається *асимптотичною часовою складністю алгоритму*.

Оцінку часової складності алгоритму можна виконувати у т.ч. для алгоритмів, які обробляють масиви. Якщо для чіткого визначення елемента масиву потрібно більше ніж один індекс, то такий масив називається *багатомірним* (multi-dimensional), інколи називають «прямокутний» масив. Кількість індексів, які асоційовано з кожним елементом масиву – це *ранг* (rank) масиву.

Частковий випадок багатомірного масиву – двомірний масив. *Приклади* застосування двомірних масивів – для зберігання:

- таблиці текстових або числових значень (елементами масиву є символи або числа, відповідно);
- значень рівнів інтенсивності кольору (наприклад, червоного) в двомірних зображеннях RGB (елементами масиву є числа).

Масив у .NET представляє фіксовану (постійну) кількість елементів масиву одного і того ж типу. Цю кількість *не*можна змінити під час виконання програми. У разі потреби змінювати кількість елементів у структурі даних (наприклад, під час завантаження із зовнішнього джерела, коли кількість елементів наперед невідома) розробники використовують динамічні структури даних (списки, дерева тощо), що у .NET називаються *колекції*.

Для роботи з колекціями у .NET існують відповідні *класи колекцій*, які призначено для зберігання та отримання даних. Ці класи розміщено у декількох просторах імен (`System.Collections`, `System.Collections.Specialized`, `System.Collections.Generic` тощо). Способи зберігання елементів у класах колекцій:

- *Загальний* (generic) – елементи сумісні з колекцією за певним заданим типом даних.
- *Незагальний* (non-generic) – в колекції може бути збережено елементи *різних* типів даних.

В загальній (generic) колекцій (типі даних) потрібно визначити *параметр*(и) *типу*(ів) в тексті програми, який звертається до цієї загальної колекції (типу даних). В C# параметр типу позначається в кутових дужках (`< >`).

У даній ЛР досліджується часова складність алгоритму і виконується її асимптотична оцінка на прикладі задачі отримання вибірки з двомірного масиву (для звертання до елементів необхідно використовувати два індекси, які часто умовно називають «індекс рядка» та «індекс колонки») та обчислення числової характеристики цієї вибірки. При цьому використано класи та елементи керування `Windows Forms`, що подано нижче.

Деякі методи класу `Array`:

- `GetLength` – метод, отримує число типу `int`, що представляє кількість елементів у заданому вимірі `Array` [4].

`List<T>` – клас (простір імен `System.Collections.Generic`), представляє строго типізований список об’єктів, доступних за індексом [4].

`DataGridView` – клас (простір імен `System.Windows.Forms`), відображає дані в сітці (grid), яку можна налаштовувати [4]. Деякі елементи:

- `AllowUserToAddRows` – властивість (типу `bool`), отримує чи задає значення, яке вказує, чи відображається користувачу параметр додавати рядки [4].
- `AutoResizeColumns` – метод, корегує ширину всіх колонок за вмістом всіх їх комірок, включно з комірками заголовків [4].

– `RowCount` – властивість (типу `int`), отримує чи задає кількість рядків, показаних у `DataGridView` [4].
– `ColumnCount` – властивість (типу `int`), отримує чи задає кількість колонок, показаних у `DataGridView` [4].

Постановка задачі

Розробити додаток Windows Forms, який:
– дозволяє користувачу:
  ◦ задавати кількість рядків `rows` та колонок `cols` двомірного масиву цілих чисел `a`;
  ◦ задавати спосіб ініціювання елементів масиву `a`: за зростанням («зліва направо, згори донизу»), або псевдовипадковими цілими числами з інтервалу $[0,10)$.
– виводить на екран:
  ◦ початковий масив `a`;
  ◦ вибірку `subarray` (за варіантом дод. 4) з масиву `a`;
  ◦ числову характеристику `subarrayCharacteristic` (за варіантом дод. 5) вибірки `subarray`,
  ◦ кількість виконаних операцій `countOperations` під час отримання вибірки `subarray` з масиву `a`.

Звертання до елементів двомірного масиву у даній ЛР потрібно виконувати «зліва направо, згори донизу»: почати в порядку зростання індексу колонки до кінця рядка, потім збільшити індекс рядка, знову почати в порядку зростання індексу колонки до кінця рядка і т.д.

Порядок виконання лабораторної роботи

1. В Робочій папці створити папку для поточної ЛР. Отримати у викладача папку з рішенням `Lr4` та проектом `Array2D`, що містить заготовку форми для додатку. Перемістити отримані файли до Робочої папки.

2. Ознайомитися з формою проекту `Array2D` (рис. 4.1) в режимі конструктора. Для цього: відкрити проект `Array2D` (розширення файлу «CSPROJ») або відповідне рішення (розширення файлу «SLN») у VS; у вікні `Solution Explorer` ПКМ по `Form1.cs`, у контекстному меню ЛКМ по `View Designer`; почергово ЛКМ по кожному компоненту на формі, під час цього у вікні `Properties`, на вкладці `Properties` () переглядати значення властивості `Name`.

3. В режимі конструктора доповнити форму проекту `Array2D` елементом `DataGridView` (*6* на рис. 4.1). Задати властивості (вікно `Properties`, вкладка `Properties` ) відповідно табл. 4.1.

4. До Звіту занести:
– ескіз (або скріншот) форми (подібно до рис. 4.1);

– перелік відповідностей «елемент керування на ескізі (скріншоті) форми» – «властивість `Name` цього елемента» (подібно до рис. 4.1).

Примітки: на рис. 4.1 подано перелік відповідностей «елемент керування» – «ім'я *класу* цього елемента», а не «*властивість* `Name` цього елемента».

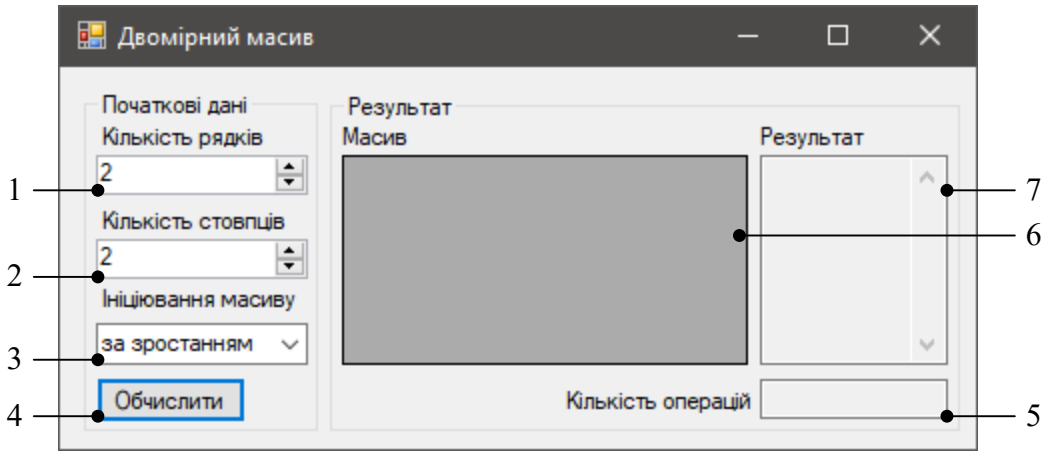


Рис. 4.1. Вікно додатку `Array2D` з елементами керування:
1, *2* – `NumericUpDown`; *3* – `ComboBox`; *4* – `Button`; *5*, *7* – `TextBox`; *6* – `DataGridView`

Таблиця 4.1

Елементи керування на формі проекту `ArraySort`

№ на формі	Назва	Властивість	Значення
6	`DataGridView`	`Name`	`dtGrdArray`

5. До Звіту занести назви методів (окрім обробників подій), які містяться в тілі класу `Form1`, та їх призначення (з коментарів у тексті програми):

– `ArrayInitialization(int rows, int cols, out int[,] a)`. *Призначення*: ініціювати та повернути масив `a` розміром `rows` рядків та `cols` колонок залежно від налаштувань на формі додатку.
– `InputData(out int[,] a)`. *Призначення*: прочитати дані з форми додатку та повернути масив `a` залежно від налаштувань на формі.
– `OutResult(int[,] a, List<int> subarray, double subarrayCharacteristic)`. *Призначення*: вивести результат (початковий масив `a`, вибірку `subarray`, числову характеристику `subarrayCharacteristic` вибірки `subarray`) на форму додатку.
– `ListIntToString(List<int> a)`. *Призначення*: перетворити список (`List`) `a` цілих чисел на рядок (тип даних `string`).
– `MatrixToDataGridView(DataGridView dtGrd, int[,] a)`. *Призначення*: записати (відобразити) двомірний масив `a` до `dtGrd` (типу `DataGridView`).

6. Доповнити тіло класу `Form1`:

6.1. Декларувати метод `GetSubarray`:

– тип результату, який повертається – `List<int>`
– список формальних параметрів – `int[,] a`
– тіло методу – самостійно розробити алгоритм розв'язання задачі за варіантом дод. 4 та реалізувати його в тексті програми.

Призначення: повертає вибірку `subarray`, що отримано з масиву `a`.

6.2. Декларувати метод `ProcessSubarray`:

– тип результату, який повертається – `double`
– список формальних параметрів – `List<int> a`
– тіло методу – самостійно розробити алгоритм розв'язання задачі за варіантом дод. 5 та реалізувати його в тексті програми.

Призначення: повертає числову характеристику `subarrayCharacteristic` вибірки `subarray`.

7. Доповнити тіло обробника `btnOK_Click`:

7.1. Перед оператором `try` декларувати змінну `subarray` типу `List`, який може містити лише елементи типу `int`.

Призначення: містить список цілих чисел, які є вибіркою, що отримано з масиву `a`.

7.2. Перед оператором `try` декларувати змінну `subarrayCharacteristic` типу `double`.

Призначення: містить числову характеристику, яку обчислено для вибірки, що отримано з масиву `a`.

7.3. В блоці операторів `try` після виклику методу `InputData` змінній `subarray` присвоїти вираз, який є викликом методу `GetSubarray` з переданим його параметру аргументом `a`.

7.4. В блоці операторів `try` після обчислення `subarray` (п. 7.3) змінній `subarrayCharacteristic` присвоїти вираз, який є викликом методу `ProcessSubarray` з переданим його параметру аргументом `subarray`.

7.5. Після оператора `try` (після закриваючої фігурної дужки останнього блоку `catch` або `finally`) викликати метод `OutResult` та передати йому аргументи `a`, `subarray`, `subarrayCharacteristic`.

8. Скомпілювати та запустити проект в режимі налагодження. Перевірити функціонування програми для вирішення задач за варіантами дод. 4 та дод. 5. У разі отримання невірних результатів – усунути недоліки.

9. Самостійно модифікувати програму для:

– обчислення кількості виконаних операцій `countOperations` під час отримання вибірки `subarray` з масиву `a`.
– виведення результатів `countOperations` до `txtBxCount`.

Рекомендації:

– декларувати змінну `countOperations` (тип даних обрати самостійно) та ініціювати значенням нуль на початку тіла методу `btnOK_Click`;

– доповнити список формальних параметрів методу `GetSubarray` (в декларації методу та під час його виклику) параметром `countOperations` так, щоб його значення було збережене після завершення виконання методу `GetSubarray` (тобто після повернення до методу, який викликав);
– накопичити `countOperations` в тілі методу `GetSubarray` шляхом збільшення значення `countOperations` на одиницю кожного разу, коли в тілі методу `GetSubarray` виконується операція;
– доповнити список формальних параметрів методу `OutResult` (в декларації методу та під час його виклику) параметром `countOperations`, щоб передати значення кількості операцій у метод `OutResult` під час його виклику;
– вивести значення `countOperations` до властивості `Text` елементу керування `txtBxCount` в тілі методу `OutResult`.

10. Оцінити кількість операцій, які виконано під час отримання вибірки `subarray` з масиву `a`, залежно від кількості елементів в масиві `a`:

10.1. Заповнити табл. 4.2 (до Звіту).

Таблиця 4.2

Кількість операцій

Опис масиву	Довжина масиву			
	10	20	30	40
Елементи ініційовано за зростанням				
Елементи ініційовано випадковими значеннями*				

* *Рядок «Елементи ініційовано випадковими значеннями» табл. 4.2 для кожної з довжин масиву заповнювати середнім значенням, яке отримано на основі 5 значень, повернутих програмою.*

10.2. На основі даних табл. 4.2 до Звіту занести:
– графіки залежностей кількості операцій від довжини масиву (табл. 4.2) залежно від способу ініціювання елементів масиву;
– приблизну оцінку (класифікацію) алгоритму, реалізованого відповідно варіанту, на основі виду функції швидкості зростання кількості операцій, виконаних цим алгоритмом, як однієї з поширених функцій (константа, логарифмічна, лінійна, лінеарифмічна, квадратична або експоненційна).

11. Зберегти проект. Остаточний текст програми – до Звіту. Закрити поточне рішення.

Зміст звіту

1. Номер, тема, мета ЛР.

2. Пункти порядку виконання ЛР: 4, 5, 10.1, 10.2, 11.
3. Висновки.

Контрольні запитання

1. Багатомірний масив. Ранг масиву.
2. Двомірний масив. Декларування та ініціювання в C#.
3. Динамічні структури даних. Загальні та незагальні колекції.
4. Динамічні структури даних. Список `List<T>`.
5. Асимптотична оцінка часової складності алгоритму.

ДОДАТОК 1

Початкові дані для задач, в яких використано гармонічну функцію

№ варіанту	Функція	Амплітуда, В	Частота, Гц	Фаза, рад.
1	SIN	220	25	2
2	COS	127	50	5
3	SIN	380	75	4
4	COS	400	100	5
5	SIN	12	25	3
6	COS	230	50	3
7	SIN	2	75	3
8	COS	115	100	2
9	SIN	240	25	2
10	COS	3	50	3
11	SIN	120	75	2
12	COS	190	100	0
13	SIN	5	25	4
14	COS	415	50	3
15	SIN	48	75	0

ДОДАТОК 2

УЗАГАЛЬНЕНИЙ АЛГОРИТМ СОРТУВАННЯ МАСИВУ «ВСТАВКАМИ»

Попередні умови:

- Сортування виконується над масивом `a` довжиною `n`. Елементи масиву – будь-які елементи, які можна порівняти та отримати один з таких результатів порівняння: «менше», «дорівнює», «більше».
- Сортування елементів масиву `a` виконується за зростанням.
- Під час кожного часткового сеансу сортування масив `a` містить дві підмножини (підмасиви):
  - Підмножина `s`, яка містить вже сортовані елементи у *поточному* сеансі сортування.
  - Підмножина `u`, яка містить усі інші елементи масиву `a` у *поточному* сеансі сортування.
- Елементи підмножини `s` розміщено *перед* елементами підмножини `u`. Тобто індекс кожного з елементів підмножини `s` є меншим за будь-який індекс елемента підмножини `u`.

Узагальнений алгоритм сортування масиву `a` довжиною `n` «вставками» (insertion sort) за зростанням, який може бути використаним в алгоритмі сортування Шелла:

1. Декларувати змінну `step` – крок (інтервал), через який порівнюватимуться елементи масиву. Ініціювати додатним цілим числом.
   - для алгоритму «вставок» – одиниця;
   - для алгоритму Шелла – залежить від обраної послідовності кроків.
2. Декларувати змінну `i` – індекс першого (початкового) елемента масиву у несортованому підмасиві `u`. Ініціювати значенням `step`.
3. Для всіх (`i < n`) повторювати такі дії:
   3.1. Декларувати змінну `current` – поточний елемент з несортованого підмасиву, який буде порівняно з елементами сортованого підмасиву. Ініціювати значенням `a[i]`.
   3.2. Декларувати змінну `j` – індекс елемента у сортованому підмасиві, який порівнюватиметься з `current`. Ініціювати виразом (`i - step`).
   3.3. Поки (`j >= 0`) та (`a[j] > current`) повторювати такі дії:
      3.3.1. Скопіювати значення `a[j]` до `a[j + step]`. Тобто ніби перемістити («посунути») елемент `a[j]` на місце елемента `a[j + step]` при цьому *не* видаляючи значення `a[j]`.
      3.3.2. Зменшити значення `j` на `step`.
   3.4. Присвоїти елементу `a[j + step]` значення `current`.
   3.5. Збільшити значення `i` на одиницю.

ДОДАТОК 3

АЛГОРИТМ СОРТУВАННЯ ШЕЛЛА

Попередні умови:

– Сортування виконується над масивом `a` довжиною `n`. Елементи масиву – будь-які елементи, які можна порівняти та отримати один з таких результатів порівняння: «менше», «дорівнює», «більше».
– Сортування елементів масиву `a` виконується за зростанням.
– Під час кожного часткового сеансу сортування масив `a` містить дві підмножини (підмасиви):
  ◦ Підмножина `s`, яка містить вже сортовані елементи у *поточному* сеансі сортування.
  ◦ Підмножина `u`, яка містить усі інші елементи масиву `a` у *поточному* сеансі сортування.
– Елементи підмножини `s` розміщено *перед* елементами підмножини `u`. Тобто індекс кожного з елементів підмножини `s` є меншим за будь-який індекс елемента підмножини `u`.
– Послідовність інтервалів між елементами масиву, які порівнюються, визначено оригінальною послідовністю Шелла: `n/2, n/4, … 1`.

Алгоритм сортування масиву `a` довжиною `n` Шелла (Shell sort) за зростанням:

1. Декларувати змінну `step` – крок (інтервал), через який порівнюватимуться елементи масиву. Ініціювати значенням `n/2`.

2. Виконати сортування «вставками» за узагальненим алгоритмом сортування «вставками» починаючи з п. 2 дод. 2.

3. Зменшити значення `step` шляхом його цілочислового ділення на 2.

4. Якщо (`step > 0`), то виконати п. 2, 3.

ДОДАТОК 4

Умови задач на отримання вибірки з двомірного масиву a

№ варіанту	Вибіркою з двомірного масиву a є список*	Приклад правильного результату**
1, 2	Кількостей *не*нульових елементів, які містяться в кожному з рядків масиву a. *Примітки****: довжина вибірки завжди дорівнює кількості рядків масиву a.	2, 3
3, 4	Ненульових значень елементів масиву a, які розміщено по головній діагоналі (для таких елементів індекс рядка дорівнює індексу колонки). *Примітки****: вибірка не містить елементів, якщо по головній діагоналі масиву a розміщено нулі.	4
5, 6	Значень тих елементів i-го рядка масиву a, які більше ніж a[i, 0]. *Примітки****: вибірка не містить елементів, якщо в нульовій колонці масиву a розміщено значення, які є не меншими за інші елементи у своїх рядках (тих рядках, в яких розташовано ці значення).	1, 2, 4, 5
7, 8	Сум значень непарних елементів по рядках масиву a. *Примітки****: максимально можлива довжина вибірки дорівнює кількості рядків масиву a; вибірка не містить елементів, якщо в масиві a відсутні непарні елементи.	1, 8
9, 10	Усіх значень елементів тих рядків масиву a, в яких міститься хоча б один нульовий елемент. *Примітки****: вибірка не містить елементів, якщо в масиві a відсутні нульові елементи.	0, 1, 2
11, 12	Значень тих елементів масиву a, які не дорівнюють 0 та за абсолютним значенням (модулем) не перевищують 5. *Примітки****: вибірка не містить елементів, якщо в масиві a відсутні ненульові елементи, абсолютне значення (модуль) яких не перевищує 5.	1, 2, 3, 4, 5
13, 14	Значень тих елементів масиву a, які менше за половину значення максимального елемента масиву a. *Примітки****: вибірка не містить елементів, якщо в масиві a відсутні елементи, значення яких менше за половину значення максимального елемента масиву a.	0, 1, 2

№ варіанту	Вибіркою з двомірного масиву a є список*	Приклад правильного результату**
15, 16	Значень тих елементів масиву a, які є парними числами та розміщені у «верхній половині» рядків масиву a. «Верхня половина» рядків масиву a – це рядки, індекс яких починається з нуля («верх») та закінчується (включно з) індексом, що дорівнює цілому числу (відкинувши дробову частину) від результату ділення максимального індексу рядка в масиві a на два. *Примітки****: вибірка не містить елементів, якщо у «верхній половині» рядків масиву a відсутні парні числа.	2

* *Якщо для заданого двомірного масиву неможливо створити (накопичити) вибірку, яка відповідає умовам задачі, тобто вибірка не містить елементів, то вивести відповідне повідомлення і припинити виконання програми.*

** *Результати приведено з припущення, що початковим є масив*

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}.$$

*** *Потрібно послідовно перевірити всі рядки масиву* a *починаючи з рядка з індексом 0; результатами перевірки кожного* i*-го рядка масиву* a *потрібно доповнити вибірку (якщо це можливо).*

ДОДАТОК 5

Умови задач на отримання числової характеристики вибірки з масиву

№ варіанту	Числовою характеристикою* вибірки з масиву є число, яке дорівнює	Приклад правильного результату**
1–4	Середньому значенню елементів вибірки	2.5
5–8	Максимальному елементу вибірки	5
9–12	Сумі квадратів елементів вибірки	55
13–16	Кількості парних елементів вибірки	2

* *Числову характеристику вибірки з масиву неможливо обчислити, якщо вибірка не містить елементів. У цьому випадку потрібно вивести відповідне повідомлення і припинити виконання програми.*

** *Результати приведено з припущення, що початковою є вибірка*

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}.$$

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. ДСТУ ISO/IEC 2382:2017 (ISO/IEC 2382:2015, IDT). Інформаційні технології. Словник термінів. Чинний від 2019-01-01. Вид. офіц. Київ : ДП «УкрНДНЦ», 2017. 468 с.
2. Lathi B. P. Modern Digital and Analog Communication Systems. 3rd ed. Oxford : Oxford University Press, Inc., 1998. 781 p.
3. McConnell J. J. Analysis of Algorithms: An Active Learning Approach. Sudbury : Jones and Bartlett Publishers, 2001. 297 p.
4. Microsoft Learn. URL: https://learn.microsoft.com/en-gb/ (date of access: 27.07.2023).

Навчально-методичне видання

Рибалка Роман Володимирович,
Маловічко Володимир Володимирович,
Маловічко Наталія Валентинівна,
Лагута Василь Васильович

КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ В СИСТЕМАХ ЗАЛІЗНИЧНОЇ АВТОМАТИКИ

Частина 2

Навчально-методичні рекомендації
до лабораторних занять

Електронне видання

В авторській редакції
Комп'ютерна верстка Р. В. Рибалка

Експертний висновок склав д-р фіз.-мат. наук, проф. Володимир Гаврилюк

Зареєстровано НМВ УДУНТ (№ 683 від 25.01.2024)

Формат 60х84 1/16. Ум. друк. арк. 2,85. Обл.-вид. арк. 2,13.
Зам. № 109

Видавець: Український державний університет науки і технологій
вул. Лазаряна, 2, ауд. 2216, м. Дніпро, 49010.
Свідоцтво суб'єкта видавничої справи ДК № 7709 від 14.12.2022

Адреса видавця та дільниці оперативної поліграфії:
вул. Лазаряна, 2, Дніпро, 49010