МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Методичні вказівки до лабораторних робіт для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Професійна освіта (комп'ютерні технології)»
галузь знань – А Освіта
спеціальність – А5.39 Професійна освіта (Цифрові технології)
денної та заочної форм навчання

Луцьк 2026

УДК 004.8(07)
Т 47

До друку
Голова вченої ради факультету КІТ_____І.С. Кондіус

Електронна копія друкованого видання передана для внесення в репозитарій ЛНТУ
Директор бібліотеки _____Н.П. Поліщук

Затверджено вченою радою факультету КІТ,
протокол № __ від «__» _____ 2026 року.

Розглянуто і схвалено на засіданні кафедри комп'ютерних наук ЛНТУ,
протокол № __ від «__»_____2026 року.
Завідувач кафедри КН _____ В.О. Ліщина

Укладачі:
_____ Ю.Й. Тулашвілі, доктор педагогічних наук, професор кафедри комп'ютерних наук ЛНТУ
_____ А.А. Ящук, кандидат технічних наук, доцент кафедри інженерії програмного забезпечення ЛНТУ

Рецензент: _____ Н.М. Ліщина, кандидат технічних наук, доцент, завідувач кафедри інженерії програмного забезпечення ЛНТУ.

Відповідальний за випуск: _____ В.О. Ліщина, кандидат технічних наук, доцент, завідувач кафедри комп'ютерних наук ЛНТУ.

Т 47 **Об'єктно-орієнтоване програмування :** методичні вказівки до лабораторних робіт для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Професійна освіта (комп'ютерні технології)» галузь знань – А Освіта, спеціальність – А5.39 Професійна освіта (Цифрові технології) денної та заоч. форм навч. / уклад. Ю.Й. Тулашвілі, А.А.Ящук. Луцьк: ЛНТУ, 2026. 31с.

Видання містить необхідний матеріал, який дасть можливість студентам засвоїти навчальний матеріал з мінімальними витратами часу.

Призначене для студентів спеціальності А5.39 Професійна освіта (Цифрові технології) денної та заочної форми навчання.

ЗМІСТ

ВСТУП 4
Лабораторна робота №1 Ознайомлення з платформою Microsoft .NET. Створення простої C# програми 5
Лабораторна робота №2 Керуючі оператори в C# 7
Лабораторна робота №3 Масиви в C#. Алгоритми розв'язання задач сортування даних 9
Лабораторна робота №4 Текстові рядки в C# 11
Лабораторна робота №5 Регулярні вирази 14
Лабораторна робота №6 Створення і використання методів в C# 17
Лабораторна робота №7 Класи та об'єкти в C# 20
Лабораторна робота №8 Ієрархія класів. Успадковування 23
Лабораторна робота №9 Введення і виведення даних, робота з файлами 25
Лабораторна робота №10 Делегати і події 27
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 30

ВСТУП

Лабораторні роботи є невід'ємною частиною навчального процесу з програмування на мові C#. Вони дозволяють студентам на практиці застосовувати теоретичні знання, отримані на лекціях, та розвивати навички розробки програмного забезпечення. Послідовне виконання лабораторних робіт сприяє глибшому розумінню основних концепцій програмування, таких як об'єктно-орієнтоване програмування, робота з базами даних, багатопотоковість, обробка тексту та інші важливі аспекти розробки програм.

Кожна лабораторна робота фокусується на окремій темі, поступово розширюючи компетенції студентів. Від простих завдань, пов'язаних зі створенням базових програм, до складніших проектів, що вимагають інтеграції кількох концепцій, лабораторні роботи допомагають студентам закріпити знання та підготуватися до реальних задач у галузі програмування.

Особлива увага приділяється правильній організації коду, використанню ефективних алгоритмів та структур даних, а також дотриманню принципів чистого коду. Важливою складовою є також робота з інструментами розробки, такими як Visual Studio, що забезпечують зручне середовище для написання, тестування та налагодження програм.

При оцінюванні лабораторних робіт враховуються наступні критерії: повнота виконання завдань, коректність коду, теоретична обізнаність студента та вміння аргументувати вибір шляхів і методів виконання поставлених завдань, якість оформлення коду, відповідність вимогам завдання.

Лабораторна робота №1 Тема: Ознайомлення з платформою Microsoft .NET. Створення простої C# програми

Мета: ознайомитися з платформою .NET і мовою програмування C#, навчитися створювати прості програми.

Теоретичні відомості

Платформа .NET дає змогу розробляти програмне забезпечення, що запускається на різних операційних системах і пристроях. До її складу входить середовище виконання Common Language Runtime (CLR), яке відповідає за управління пам'яттю та виконання керованого коду, а також базова бібліотека класів, що містить набір готових до використання функціональних можливостей. Мова C# є строго типізованою, об'єктно-орієнтованою мовою, яка тісно інтегрується з усіма складовими платформи .NET. Створення простих програм зазвичай починається з консольних застосунків, де головна точка входу — метод Main. Одним із базових прикладів використання стандартної бібліотеки є звернення до класу Math, який надає методи для математичних обчислень (обчислення коренів, степенів, тригонометричних функцій, логарифмів тощо), що суттєво спрощують написання коду.

Завдання 1

Створити просту консольну програму на C# для виведення інформації про студента на екран.

Варіанти завдань

№	Параметри для відображення	Формат виведення (приклад)
1	Прізвище, ім'я	Прізвище Ім'я
2	Прізвище, ім'я, група	Студент: Прізвище Ім'я, група XXXX
3	Прізвище, ім'я, дата народження	Студент: Прізвище Ім'я, дата народження: 01.01.2000
4	Прізвище, ім'я, спеціалізація	Студент: Прізвище Ім'я, спеціалізація: Інженерія ПЗ
5	Прізвище, ім'я, рік навчання	Студент: Прізвище Ім'я, рік навчання: 2

Примітка: Якщо номер вашого варіанту більше 5, то віднімiть від нього 5, 10, 15, 20 тощо, щоб отримати число від 1 до 5 варіанту з таблиці.

Завдання 2

Створити програму для розрахунку математичних виразів з використанням методів класу Math. Значення x_n вводяться з клавіатури.

Варіанти завдань

Варіант	Завдання	Варіант	Завдання
1.	$y=\frac{\sqrt{x_1^{\,3}+x_2^{\,5}}}{1000}+65$	**2.**	$y=\sqrt{56x_1+\frac{x_1+x_2+\sin(x_1x_2)}{5-\cos(x_2^{\,2})}}$
3.	$y=\cos(x_1-x_2^{\,2})+31.55x_2x_1^{\,2}$	**4.**	$y=\sqrt[5]{0.1x_1\sin x_2\cos x_1^{\,2}+55x_1x_2}$
5.	$y=\frac{6-\cos(3+x_1)}{34-9x_2^{\,3}+x_2}$	**6.**	$y=\sqrt{\frac{\cos(2x_2)+x_1/x_2}{16x_2x_1}}$
7.	$y=\cos^4\left(x_1-\sqrt{\frac{x_2}{x_1+53x_2^{\,2}}}\right)$	**8.**	$y=\cos(\sqrt{x_2}+34x_1)-4\sin(x_2)$
9.	$y=23\sin^2(x_1^{\,3}x_2^{\,5})+2x_1+\cos(x_1x_2)$	**10.**	$y=\sin^2\left(x_1\frac{x_2}{x_1+53x_2^{\,2}}\right)$

Додаткове завдання

Здійснювати базову перевірку вхідних даних (наприклад, перевіряти, чи x не виходить за межі припустимих значень) та виводити повідомлення про помилку за потреби.

Контрольні запитання

1. Які особливості платформи .NET роблять її зручною для розробників?
2. Як оголошується метод Main у C# і в чому його роль у програмі?
3. Які основні можливості пропонує клас Math?
4. Чому важливо перевіряти припустимість вхідних даних перед обчисленням виразу?
5. У чому полягає різниця між виведенням тексту методом Console.WriteLine і Console.Write?

Лабораторна робота №2 Тема: Керуючі оператори в C#

Мета: навчитися використовувати конструкцію `if..else`, а також цикли `for`, `while`, `do…while` у програмах на C#.

Теоретичні відомості

У мові C# умовні та циклічні конструкції дозволяють керувати послідовністю виконання інструкцій у програмі. Конструкція `if..else` дає змогу перевіряти умову та виконувати певний блок коду, якщо ця умова істинна, або інший блок, якщо умова хибна. Щоби виконати повторювані дії, використовуються цикли. Цикл `for` застосовується здебільшого тоді, коли відома кількість ітерацій наперед; цикл `while` повторює свій блок доти, доки задана умова істинна; цикл `do..while` схожий на `while`, але перевіряє умову наприкінці циклу, тому тіло циклу виконується хоча б один раз. Правильне використання керуючих операторів дає змогу створювати більш гнучкі та ефективні алгоритми, адаптуючись до різноманітних логічних гілок і циклічних процесів.

Завдання 1

Створити консольну програму з використанням конструкції `if..else`.

Виконайте всі завдання з таблиці

Умова перевірки
Визначити чи число парне
`Визначити число` більше, менше або дорівнює нулю
Введений користувачем рік є високосним чи ні

Завдання 2

Створити консольну програму з використанням циклів `for`, `while`, `do..while`.

Виконайте всі завдання з таблиці

Циклічна конструкція	**Приклад логіки виконання**
`for`	Вивести числа від 1 до N, обчислити суму чи добуток
`while`	Обчислити факторіал числа, поки лічильник не перевищить його
`do..while`	Зчитувати введені користувачем числа, доки не введено 0

`for`, `while`, `do..while` разом	Порівняти час виконання або підрахувати кількість певних дій

Задання 3

Вдосконалити програми, що були створені в результаті виконання лабораторної роботи 1 наступним чином:

Додати меню вибору дії:

1 – вивести інформацію про студента

2 – обчислити заданий математичний вираз

0 – вийти з програми

При виборі пункту користувачем виконуються відповідні дії.

Зробити повторний запит після обчислення виразу: «Обчислити ще раз? (так/ні)» та реалізувати можливість багаторазових обчислень без перезапуску програми.

Додаткові завдання

1. Розробити програму, що порівнює три введені користувачем числа і визначає, чи можуть вони бути довжинами сторін трикутника, а якщо так – з'ясувати, який це тип трикутника (рівносторонній, рівнобедрений або різносторонній). При цьому використати конструкції `if..else` та хоча б один цикл для повторення обчислень, доки користувач не вирішить завершити.

2. Зробити програму, в якій виконується багаторазове введення і перевірка даних: якщо користувач вводить від'ємне значення, цикл припиняється. Для обчислень і перевірок застосувати конструкції `if..else` й один із циклів (`while` або `do..while`).

3. Створити програму для обчислення квадратних рівнянь.

Контрольні запитання

1. Чим відрізняється цикл while від циклу do..while?
2. Як працює конструкція if..else при вкладених умовах?
3. У чому полягає відмінність у використанні змінної–лічильника циклу for у тілі циклу та поза ним?
4. Як перевірити наявність кількох умов одночасно в інструкції if?
5. Яка загальна структура циклічної конструкції for і для чого призначені її три секції?

Лабораторна робота №3 Тема: Масиви в C#. Алгоритми розв’язання задач сортування даних

Мета: навчитися створювати і використовувати масиви, а також методи для роботи з ними і цикл `foreach` у програмах на C#.

Теоретичні відомості

Масив є структурою даних, яка дозволяє зберігати набір елементів одного типу в певному порядку. У C# масиви можуть бути одномірними, багатовимірними або зубчастими. Ініціалізація масиву виконується за допомогою оператора `new` і вказування розміру або без нього у випадку надання початкового набору значень. Для перебору елементів масиву часто використовують цикл `for`, однак цикл `foreach` робить цей процес зручнішим і більш безпечним, оскільки не допускає звернень за межі масиву. Окрім стандартних операцій додавання й читання елементів, у C# є методи та бібліотеки для сортування (наприклад, `Array.Sort`) та пошуку.

Завдання 1

Створити масив, наповнити його випадковими значеннями і виконати базові операції.

Варіанти завдань

№	Тип масиву	Дії
1	Одновимірний int	1) Ініціалізація випадковими значеннями. 2) Пошук мінімального та максимального елемента.
2	Одновимірний double	1) Ініціалізація випадковими дійсними числами. 2) Обчислення кількості додатних і від’ємних елементів.
3	Двозвимірний double	1) Ініціалізація випадковими дійсними числами. 2) Пошук найменшого та найбільшого елемента.
4	Двозвимірний int	1) Ініціалізація випадковими значеннями. 2) Пошук найменшого рядка за сумою елементів.

Завдання 2

Відсортувати масив за зростанням і за спаданням, визначити розмір масиву, а також вивести його елементи за допомогою циклу `foreach`.

Варіанти завдань

№	Тип масиву	Дії
1	Одновимірний int	1) Сортування за зростанням (наприклад, методом `Array.Sort`). 2) Сортування за спаданням (зворотній порядок або власна реалізація). 3) Виведення елементів за допомогою `foreach`.
2	Одновимірний double	1) Сортування за зростанням стандартним методом. 2) Сортування за спаданням. 3) Підрахунок кількості елементів за допомогою `Length`.

Додаткові завдання

1. Відсортувати масив за зростанням і за спаданням, визначити розмір масиву, а також вивести його елементи за допомогою циклу `foreach`.

1	Багатовимірний int	1) Перетворення даних у одновимірну форму та сортування. 2) Відновлення структури й виведення.
2	Зубчастий масив int	1) Сортування кожного «рядка» окремо. 2) Визначення сумарного розміру (всіх рядків). 3) Виведення за допомогою `foreach`.

2. Розробити програму для обчислення значень певної функції (наприклад, $f(x) = \sin(x^2)$ або $f(x) = \ln(x + 1)$) у заданому діапазоні аргументів із кроком dx, зберегти результати в масиві та вивести їх на екран. При цьому використати цикл `foreach` для виведення елементів і здійснити сортування масиву за отриманими значеннями.

3. Реалізувати пошук елемента в масиві за допомогою власної реалізації алгоритму «лінійного пошуку» або «бінарного пошуку» й вивести відповідне повідомлення, якщо елемент не знайдено. Використовувати лише механізми, розглянуті до цієї теми.

Контрольні запитання

1. У чому полягають основні відмінності між одномірними, багатовимірними та зубчастими масивами в C#?

2. Як ініціалізувати масив випадковими значеннями, не перевищуючи межі типу?
3. Які методи класу Array найчастіше використовуються для сортування та пошуку?
4. Як за допомогою циклу foreach можна вивести всі елементи багатовимірного масиву?
5. Яка різниця між довжиною масиву Length та іншими можливими показниками розміру (наприклад, GetLength(dimension)) у випадку багатовимірних масивів?

Лабораторна робота №4 Тема: Текстові рядки в C#

Мета: навчитися працювати з типом даних string і класом StringBuilder для обробки тексту в програмах на C#.

Теоретичні відомості

Текстові рядки в C# представлені типом string, що є посилальним типом і зберігається як незмінний (immutable) об’єкт. Будь-яка операція, яка змінює рядок (наприклад, конкатенація), фактично створює новий екземпляр. Це робить роботу з рядками безпечною, проте не завжди ефективною з точки зору пам’яті та швидкості. Для інтенсивної обробки тексту, коли виконується багато модифікацій у рядках, доцільно використовувати клас StringBuilder, що надає методи для динамічної зміни вмісту без постійного створення нових об’єктів. У C# також доступні методи для пошуку, заміни, розбиття та інших операцій із рядками (наприклад, методи класу string: Contains, Replace, Split, ToUpper, ToLower тощо). Дбайливе використання цих методів допомагає ефективно реалізувати пошук і форматування даних у тексті, а також виконувати аналітичні завдання, пов’язані з обробкою великих обсягів текстових відомостей.

Завдання 1

Розробити програму для роботи з текстом з використанням вбудованих методів класу string. Здійснити пошук і заміну фрагмента тексту, підрахунок кількості слів, видалення розділових знаків, а також перетворення тексту з нижнього регістру у верхній (або навпаки).

Варіанти завдань

№	Операції над рядками
1	1) Пошук першого входження заданого слова 2) Заміна знайденого слова іншим

	3) Перетворення всього рядка в нижній регістр
2	1) Підрахунок кількості слів у рядку 2) Видалення розділових знаків (.,!?: тощо) 3) Виведення рядка, перетвореного у верхній регістр
3	1) Видалення всіх цифр із тексту 2) Пошук і заміна фрагмента, введеного користувачем 3) Конкатенація результату з якимось іншим рядком
4	1) Розбиття вихідного рядка на кілька частин (Split) за заданим символом 2) Підрахунок кількості унікальних слів 3) Перетворення першої літери кожного слова в UpperCase
5	1) Перевірка наявності в рядку конкретного фрагмента (Contains) 2) Вилучення зайвих пробілів (Trim, Replace тощо) 3) Формування нового рядка з упорядкованих слів початкового тексту

Завдання 2

Створити програму для роботи з текстом із використанням класу StringBuilder та його методів. Виконати складні перетворення тексту, що передбачають додавання, вставлення, видалення й заміну символів або фрагментів.

Варіанти завдань

№	Операції з StringBuilder
1	1) Ініціалізація StringBuilder рядком 2) Вставка певного слова у вказану позицію 3) Видалення фрагмента між двома індексами
2	1) Додавання (Append) нового рядка 2) Перевірка певної послідовності символів у результаті (через додаткову перевірку в string) 3) Заміна першого входження конкретного символа або фрагмента
3	1) Формування довгого рядка з результатів обчислень (циклічна конкатенація) 2) Застосування Insert для вставки додаткового тексту 3) Видалення символів, що не є літерами
4	1) Створення великого абзацу тексту з повторенням фраз (AppendFormat, AppendLine) 2) Вставка розділювачів між реченнями 3) Заміна певної підрядкової послідовності на іншу

5	1) Порівняння продуктивності багаторазового додавання рядків між string та StringBuilder (вимір часу виконання) 2) Здійснення кількох вставок і видалень для моделювання сценарію редагування тексту

Додаткові завдання

1. Сформувати рядок випадкових літер, символів і цифр заданої довжини, використовуючи StringBuilder, а потім видалити з нього всі цифри та символи, залишивши лише літери.

Зчитати з клавіатури кілька речень (кількість задається користувачем). Зберегти ці речення в масиві рядків і виконати такі дії:

- відсортувати елементи масиву за довжиною речень (за зростанням або за спаданням);
- у реченнях, довжина яких перевищує певний поріг (наприклад, 50 символів), замінити всі літери «а» на «@»;
- вивести результати, використавши цикл `foreach`.

2. Створити програму, що приймає з клавіатури N рядків і зберігає їх у масиві. Для кожного рядка перевіряти:

- якщо рядок починається з певного символа (скажімо, «#»), пропустити його обробку (`continue`);
- якщо рядок порожній, перервати обробку всього масиву (`break`);
- інакше підрахувати кількість слів у рядку (використовуючи методи `Split` і `Length` або подібні).

Наприкінці вивести всі проаналізовані рядки з інформацією про кількість слів у кожному.

3. Згенерувати випадкові імена (наприклад, 5–10 імен) і зберегти їх в одновимірному масиві. Потім:

- за допомогою циклу `for` відфільтрувати імена, які починаються з букви, що вводить користувач (наприклад, «А»);
- для всіх імен, які залишилися після фільтрації, здійснити перетворення в верхній регістр;
- результат вивести на екран у відсортованому порядку (використати `Array.Sort` або інший метод).

4. Розробити програму, яка приймає з клавіатури рядок і поміщає окремі слова у масив. Для кожного слова здійснити низку перевірок, використовуючи конструкції `if..else`:

- якщо слово містить хоча б одну цифру, замінити цю цифру на символ «*»;

- якщо довжина слова перевищує 10 символів, відсікти зайві символи, залишивши перші 10;
- усі слова зберегти назад у рядок через розділювач «_» та вивести кінцевий результат.

Контрольні запитання

1. Чому рядки в C# вважаються незмінними (immutable)?
2. Коли доцільно застосовувати клас StringBuilder замість string?
3. Які основні методи класу StringBuilder корисні для редагування тексту?
4. Які підходи існують для видалення або заміни зайвих символів у рядку?
5. Як виміряти продуктивність операцій з рядками, якщо необхідно порівняти виконання кількох підходів?

Лабораторна робота №5 Тема: Регулярні вирази

Мета: навчитися використовувати регулярні вирази в програмах на C# для пошуку, перевірки та заміни текстових фрагментів.

Теоретичні відомості

Регулярні вирази (Regular Expressions) – це потужний інструмент для пошуку та обробки текстових даних за визначеними шаблонами. У середовищі C# для роботи з регулярними виразами використовується простір імен `System.Text.RegularExpressions`, що містить класи `Regex`, `Match`, `MatchCollection` та інші. Клас `Regex` надає методи `IsMatch` для перевірки відповідності рядка шаблону, `Match` і `Matches` для пошуку збігів, `Replace` для заміни фрагментів тексту і `Split` для розбиття рядка. Регулярні вирази допомагають легко відшукати структури на кшталт адрес електронної пошти, номерів телефонів, окремих слів певної довжини або набору символів, а також здійснити валідацію введених користувачем даних. Грамотне володіння синтаксисом регулярних виразів дає змогу швидко та ефективно обробляти рядкові дані в практично будьяких прикладних задачах.

Завдання 1

Створити програму для пошуку шаблонних текстів у заданому рядку з використанням регулярних виразів (`Regex`). Знайти всі електронні адреси, номери телефонів тощо залежно від варіанту.

Варіанти завдань

№	Тип даних для пошуку	Приклад шаблону або опис
1	Електронна адреса (email)	Формат: username@domain, де username і domain — з латинських літер, цифр, можливі крапки та підкреслення
2	Номери телефонів	Наприклад: `(XXX) XXX-XX-XX` або `+XX(XXX)XXXXXXX`
3	IP-адреса	Формат IPv4 (чотири числа 0–255 розділені крапками)
4	Дати в форматі `DD.MM.YYYY` або `YYYY-MMDD`	Перевірка коректності значень днів, місяців, року (на рівні регулярного виразу)
5	Пошук слів із певною кількістю літер	Можна вказати довжину слова, наприклад, 5 або 6 символів

Завдання 2

Знайти всі слова, які містять символи англійського алфавіту, або слова, що мають задану кількість літер. Реалізувати можливість заміни або вилучення знайдених фрагментів за потреби.

Варіанти завдань

№	Умова пошуку	Дія над знайденими фрагментами
1	Слова, що містять лише символи англійського алфавіту (`a-z`, `A-Z`)	Видалити або замінити іншим рядком
2	Слова, довжиною рівно N символів	Замінити на символ «*» або будь-який інший
3	Слова, що починаються з великої літери	Перетворити у всі малі літери
4	Слова, що містять певний фрагмент (наприклад, «test»)	Вилучити з речення або перенести в окремий масив
5	Додаткові умови пошуку (наприклад, слова з апострофом)	Узгоджуються залежно від поставленої задачі

Додаткові завдання

Варіант 1: Створення додатку для пошуку та заміни тексту Опис: Розробити графічний інтерфейс користувача (GUI) за допомогою Windows Forms, WPF, MAUI або іншої технології, який дозволяє користувачу вводити

текст, задавати регулярний вираз для пошуку та заміни, а також відображати результати.

Етапи виконання:

- інтерфейс користувача;
- поле для вводу тексту;
- поле для введення регулярного виразу (шаблону) для пошуку;
- поле для введення тексту, на який буде здійснюватися заміна;
- кнопка "Пошук" для відображення знайдених фрагментів;
- кнопка "Замінити" для виконання заміни;
- поле для відображення результатів (наприклад, `TextBox` або `ListBox`).

Функціонал:

- при натисканні на "Пошук" використовувати клас `Regex` для знаходження всіх збігів у введеному тексті та відображати їх у відповідному полі;
- при натисканні на "Замінити" виконувати заміну всіх знайдених фрагментів на введений текст і відображати змінений текст.

Валідація:

- перевіряти коректність введеного регулярного виразу. у разі помилки показувати повідомлення користувачу.

Варіант 2: Створення додатку для валідації даних користувача

Опис: Розробити GUI-додаток, який дозволяє користувачу вводити різні типи даних (наприклад, email, телефон, IP-адресу) та перевіряти їх відповідність заданим регулярним виразам.

Етапи виконання:

Інтерфейс користувача:

- комбінація `ComboBox` для вибору типу даних (Email, Телефон, IP-адреса);
- поле `TextBox` для введення даних;
- кнопка "Перевірити" для запуску валідації;
- поле `Label` або `TextBlock` для відображення результату перевірки.

Функціонал:

- в залежності від вибраного типу даних, використовувати відповідний регулярний вираз для перевірки введеної інформації;
- відображати повідомлення про успішну або неуспішну перевірку.

Валідація:

- забезпечити коректність введених даних та обробку можливих помилок у регулярних виразах.

Порада: Для розробки GUI-додатків можна використовувати Visual Studio, яка надає зручні засоби для створення форм та інтеграції з кодом C#. Ознайомтеся з основами Windows Forms , WPF або MAUI для ефективного впровадження графічних інтерфейсів у ваші проєкти.

Контрольні запитання

1. Які основні класи використовуються в C# для роботи з регулярними виразами?
2. Як реалізувати пошук усіх збігів у рядку замість лише першого?
3. Як виконати заміну знайдених збігів іншим текстом і який метод для цього використовується?
4. Яка різниця між жадібними (greedy) та ледачими (lazy) квантифікаторами в регулярних виразах?
5. Як перевірити, що вхідний рядок повністю відповідає заданому шаблону, а не містить лише частковий збіг?

Лабораторна робота №6 Тема: Створення і використання методів в C#

Мета: навчитися створювати і використовувати методи в програмах на C#, розуміти різницю між статичними та нестатичними методами, а також освоїти передачу параметрів за допомогою `ref`, `out` та `params`.

Теоретичні відомості

Методи в C# є основними будівельними блоками програм, які дозволяють організовувати код у логічні одиниці для повторного використання та полегшення розуміння. Статичні методи належать до самого класу і можуть викликатися без створення екземпляра класу, тоді як нестатичні методи потребують створення об'єкта для доступу до них. Використання параметрів `ref` і `out` дозволяє методам змінювати значення змінних, переданих у них, що може бути корисним для повернення кількох результатів одночасно. Параметр `params` надає можливість передавати змінну кількість аргументів одному методу, що робить методи більш гнучкими і універсальними.

Правильне використання методів сприяє підвищенню читабельності коду, його модульності та спрощує налагодження програм. Методи можуть мати різні рівні доступу (`public`, `private`, `protected`), що дозволяє контролювати їх видимість і використання в інших частинах програми. Крім того, методи можуть повертати значення або бути типу `void`, залежно від потреб програми. Освоєння

цих концепцій є ключовим етапом у навчанні об’єктно-орієнтованого програмування на мові C#.

Завдання 1

Створити програму зі статичними і нестатичними методами.

Варіанти завдань

№	Опис завдання
1	Створити клас `Calculator` зі статичним методом `Add`, який приймає два числа і повертає їх суму, та нестатичним методом `Multiply`, який приймає два числа і повертає їх добуток. Використати ці методи у головному класі для обчислення значень.
2	Розробити клас `Person` з нестатичним методом `DisplayInfo`, який виводить інформацію про особу. Додати статичний метод `GetPopulation`, який повертає кількість створених екземплярів класу `Person`. Використати ці методи у програмі.
3	Створити клас `MathOperations` зі статичним методом `Power`, який підносить число до заданої ступені, та нестатичним методом `Factorial`, який обчислює факторіал числа. Використати ці методи у програмі для обчислення різних математичних виразів.
4	Розробити клас `StringUtils` зі статичним методом `Concatenate`, який об'єднує два рядки, та нестатичним методом `Reverse`, який повертає зворотний порядок символів у рядку. Використати ці методи у програмі для обробки текстових даних.
5	Створити клас `TemperatureConverter` зі статичним методом `CelsiusToFahrenheit` та нестатичним методом `FahrenheitToCelsius`. Використати ці методи для конвертації температурних показників у програмі.

Завдання 2

Створити методи з використанням `ref` і `out` параметрів.

Варіанти завдань

№	Опис завдання
1	Створити метод `Swap`, який приймає два цілих числа за допомогою параметрів `ref` і міняє їх значення місцями. Використати цей метод у програмі для демонстрації роботи з `ref`.

2	Розробити метод `Divide`, який приймає два числа і повертає результат ділення через `out` параметр, а також повертає булеве значення, що вказує на успішність операції (наприклад, уникнення ділення на нуль).
3	Створити метод `CalculateStatistics`, який приймає масив чисел і за допомогою `out` параметрів повертає мінімальне, максимальне значення та середнє арифметичне масиву. Використати цей метод у програмі.
4	Розробити метод `TryParseInt`, який намагається перетворити рядок у ціле число, використовуючи `out` параметр. Метод повертає булеве значення, що вказує на успішність перетворення.
5	Створити метод `UpdateValues`, який приймає кілька змінних за допомогою `ref` і виконує їх оновлення відповідно до певної логіки (наприклад, збільшує їх на задане число).

Завдання 3

Створити метод з використанням параметра `params`.

Варіанти завдань

№	Опис завдання
1	Створити метод `SumAll`, який приймає змінну кількість цілих чисел за допомогою параметра `params` і повертає їх суму. Використати цей метод у програмі для обчислення суми чисел.
2	Розробити метод `FindMax`, який приймає змінну кількість чисел (цілих або дійсних) за допомогою `params` і повертає найбільше значення. Використати метод у програмі.
3	Створити метод `ConcatenateStrings`, який приймає змінну кількість рядків за допомогою параметра `params` і повертає об'єднаний рядок. Використати цей метод для з'єднання декількох рядків.
4	Розробити метод `Average`, який приймає змінну кількість чисел за допомогою `params` і повертає їх середнє арифметичне. Використати метод у програмі для обчислення середнього значення.
5	Створити метод `PrintAll`, який приймає змінну кількість об'єктів будь-якого типу за допомогою параметра `params` і виводить їх на екран. Використати цей метод для виведення різних типів даних.

Додаткові завдання

1. Створення та використання методів з різними параметрами: розробити програму, яка використовує всі типи методів, розглянутих у завданнях 1-3. Наприклад, створити калькулятор, який може виконувати різні операції за допомогою статичних і нестатичних методів, використовувати `ref` та `out` параметри для обміну значень між методами, а також параметр `params` для обчислення суми декількох чисел одночасно.

2. Інтеграція з попередніми темами: створити GUI-додаток за допомогою Windows Forms або WPF, який дозволяє користувачу вводити числа у текстові поля та обирати операції (додавання, множення, обмін значень тощо). Використати створені методи для виконання обчислень та відображення результатів у інтерфейсі. Забезпечити перевірку введених даних і обробку можливих помилок за допомогою `try..catch`.

Контрольні запитання

1. Яка різниця між статичними та нестатичними методами у C#?
2. Як працюють параметри ref і out у методах, і в чому їхні основні відмінності?
3. Для чого використовується параметр params у методах, і які переваги він надає?
4. Які правила існують для оголошення та використання методів у C#? 5. Як можна викликати нестатичний метод класу в іншому класі?

Лабораторна робота №7 Тема: Класи та об’єкти в C#

Мета: навчитися створювати і використовувати класи і об’єкти в програмах на C#, розуміти основні принципи об’єктно-орієнтованого програмування, такі як інкапсуляція, абстракція, успадковування та поліморфізм.

Теоретичні відомості

У мові C# клас є основною одиницею об’єктно-орієнтованого програмування, яка служить шаблоном для створення об’єктів. Клас може містити поля (змінні), властивості, методи, конструктори та події. Поля використовуються для зберігання стану об’єкта, властивості – для доступу до цих полів з контролем, методи — для виконання дій або операцій, конструктори – для

ініціалізації об’єктів під час їх створення. Об’єкт є конкретним екземпляром класу, який має власні значення полів і може викликати методи класу. Успадковування дозволяє створювати нові класи на основі існуючих, розширюючи або модифікуючи їхню функціональність. Інкапсуляція забезпечує приховання внутрішньої реалізації класу від зовнішнього світу, дозволяючи взаємодіяти з об’єктом тільки через визначені інтерфейси. Поліморфізм дозволяє об’єктам різних класів бути обробленими через спільний інтерфейс, що спрощує розширення та підтримку коду.

Завдання 1

Створити клас, конструктор, методи, поля і властивості відповідно до обраного варіанту.

Варіанти завдань

№	Опис завдання
1	Створити клас `Car` з полями `Make`, `Model`, `Year`, властивостями для доступу до цих полів, конструктором для ініціалізації об’єктів, методом `DisplayInfo`, який виводить інформацію про автомобіль.
2	Розробити клас `Book` з полями `Title`, `Author`, `ISBN`, властивостями для доступу до цих полів, конструктором, методом `GetBookDetails`, який повертає інформацію про книгу у вигляді рядка.
3	Створити клас `Student` з полями `FirstName`, `LastName`, `Age`, властивостями, конструктором, методом `PrintStudent`, який виводить інформацію про студента.
4	Розробити клас `Rectangle` з полями `Length` та `Width`, властивостями, конструктором, методами `CalculateArea` та `CalculatePerimeter`, які обчислюють площу та периметр прямокутника відповідно.
5	Створити клас `Employee` з полями `Name`, `Position`, `Salary`, властивостями, конструктором, методом `DisplayEmployee`, який виводить інформацію про працівника, а також методом `IncreaseSalary`, який збільшує зарплату на задану суму.

Завдання 2

Створити екземпляри попередньо створеного класу, задати параметри з використанням конструктора і властивостей. Викликати методи через екземпляри класів.

Варіанти завдань

№	Опис завдання
1	Створити кілька об’єктів класу `Car`, ініціалізувати їх через конструктор, змінити деякі властивості через властивості класу та викликати метод `DisplayInfo` для кожного об’єкта.
2	Розробити програму, яка створює об’єкти класу `Book`, задає їхні властивості, а потім викликає метод `GetBookDetails` для відображення інформації про кожну книгу.
3	Створити кілька об’єктів класу `Student`, ініціалізувати їхні дані, змінити деякі властивості, і викликати метод `PrintStudent` для відображення інформації про кожного студента.
4	Розробити програму, яка створює об’єкти класу `Rectangle`, задає довжину та ширину через конструктор і властивості, а потім викликає методи `CalculateArea` та `CalculatePerimeter` для обчислення і виведення результатів.
5	Створити кілька об’єктів класу `Employee`, ініціалізувати їх через конструктор, змінити зарплату за допомогою методу `IncreaseSalary`, а потім викликати метод `DisplayEmployee` для відображення оновленої інформації.

Додаткове завдання

Використання масивів з класами. Створити масив об’єктів класу, наприклад, `Car[]`, заповнити його кількома екземплярами, а потім за допомогою циклу `foreach` вивести інформацію про кожен автомобіль, використовуючи методи класу.

Контрольні запитання

1. Яка різниця між статичними і нестатичними методами у класі C#?
2. Як реалізується інкапсуляція у класах C#?
3. Що таке конструктор класу і для чого він використовується?
4. Як здійснюється успадковування між класами у C#?
5. У чому полягає поліморфізм у контексті об’єктно-орієнтованого програмування?

Лабораторна робота №8 Тема: Ієрархія класів. Успадковування

Мета: навчитися створювати і використовувати похідні класи в програмах на C#, навчитись керувати доступом до членів базових і похідних класів.

Теоретичні відомості

Успадковування є однією з фундаментальних концепцій об'єктноорієнтованого програмування (ООП), яка дозволяє створювати нові класи на основі вже існуючих. У мові C# це досягається шляхом створення похідного класу, який наслідує властивості та методи базового класу. Такий підхід сприяє повторному використанню коду, полегшує підтримку та розширення програмних систем.

Клас, від якого успадковуються, називається базовим класом або суперкласом, а клас, що наслідує, – похідним класом або дочірнім класом. Похідний клас може додавати нові члени або перевизначати вже існуючі методи базового класу, забезпечуючи таким чином спеціалізацію поведінки.

У мові C# доступ до членів класу контролюється за допомогою модифікаторів доступу:

`public`: член доступний з будь-якого місця.

`private`: член доступний тільки всередині свого класу.

`protected`: член доступний всередині свого класу та в похідних класах.

`internal`: член доступний тільки всередині поточної збірки.

Використання цих модифікаторів дозволяє забезпечити інкапсуляцію, приховуючи внутрішню реалізацію класів і надаючи тільки необхідні інтерфейси для взаємодії з об'єктами.

Крім того, мова C# надає ключове слово `sealed`, яке використовується для заборони подальшого успадковування від певного класу. Це може бути корисним для запобігання створенню несанкціонованих похідних класів або для оптимізації продуктивності.

Завдання 1

Створити базовий і похідний класи. В базовому і похідному класі визначити конструктори, поля і властивості. Використати модифікатори доступу `private`, `public` і `protected`.

Варіанти завдань

№	Опис завдання
1	Створити базовий клас `Person` з полями `FirstName`, `LastName` (protected) і `Age` (private). Визначити властивості для доступу до цих полів. Додати конструктор для ініціалізації полів. Створити похідний

	клас `Student`, який додає поле `StudentID` та властивість для нього.
2	Розробити базовий клас `Animal` з полями `Name` (public) і `Age` (protected). Визначити властивості та метод `MakeSound()`. Створити похідний клас `Dog`, який перевизначає метод `MakeSound()` і додає поле `Breed` (private) з відповідною властивістю.
3	Створити базовий клас `Vehicle` з полями `Make`, `Model` (protected) і `Year` (private). Визначити властивості та конструктор. Створити похідний клас `Car`, який додає поле `NumberOfDoors` (public) та метод `DisplayInfo()`, що виводить інформацію про автомобіль.
4	Розробити базовий клас `Employee` з полями `Name`, `Position` (protected) і `Salary` (private). Визначити властивості та метод `Work()`. Створити похідний клас `Manager`, який додає поле `Department` (public) та метод `Manage()`, що виводить інформацію про керівника.
5	Створити базовий клас `Shape` з методом `CalculateArea()` (virtual) та полем `Color` (public). Визначити властивості та конструктор. Створити похідний клас `Circle`, який додає поле `Radius` (private) та перевизначає метод `CalculateArea()`.

Завдання 2

Створити клас, заборонений для успадковування з використанням ключового слова `sealed`.

Додаткове завдання

Створити GUI-додаток за допомогою Windows Forms або WPF, який дозволяє користувачу вводити дані про об’єкти базового та похідного класів. Використати різні модифікатори доступу для полів і властивостей, забезпечити можливість відображення інформації через інтерфейс. Наприклад, додати кнопки для створення об’єктів, введення даних та відображення інформації про них у текстових полях або списках.

Контрольні запитання

1. Що таке успадковування в об’єктно-орієнтованому програмуванні і які його переваги?
2. Які модифікатори доступу існують у C# і як вони впливають на видимість членів класу?
3. Яка різниця між базовим класом і похідним класом?

4. Для чого використовується ключове слово sealed і які наслідки його застосування? 5. Як можна перевизначити метод базового класу у похідному класі?

Лабораторна робота №9 Тема: Введення і виведення даних, робота з файлами

Мета: навчитися отримувати доступ до файлової системи, створювати, читати, редагувати файли за допомогою програми, написаної на C#.

Теоретичні відомості

Введення і виведення даних є невід'ємною частиною будь-якої програми, що дозволяє взаємодіяти з користувачем або зберігати інформацію для подальшого використання. У мові програмування C# робота з файлами здійснюється за допомогою простору імен `System.IO`, який надає різноманітні класи для створення, читання, запису та редагування файлів. Основні класи для роботи з файлами включають `File`, `FileStream`, `StreamReader`, `StreamWriter`, `BinaryReader` та `BinaryWriter`.

Клас `File` надає статичні методи для виконання операцій над файлами, таких як копіювання, переміщення, видалення та перевірка існування файлів. `FileStream` дозволяє працювати з файлами на байтовому рівні, що корисно для обробки двійкових даних. `StreamReader` та `StreamWriter` використовуються для читання та запису текстових файлів відповідно, забезпечуючи зручний спосіб обробки текстових рядків.

При роботі з файлами важливо враховувати обробку винятків, таких як `IOException`, яка може виникнути при спробі доступу до файлів, що не існують, або при відсутності прав доступу. Використання конструкцій `try..catch` дозволяє забезпечити стабільність програми шляхом належної обробки помилок.

Також важливо враховувати ефективність роботи з файлами, особливо при обробці великих обсягів даних. Буферизація вводу/виводу та використання асинхронних методів можуть значно покращити продуктивність програми.

Завдання 1

Створити програму для читання текстового файлу.

Варіанти завдань

№	Опис завдання
1	Створити програму, яка зчитує весь вміст текстового файлу та виводить його на екран. Використати клас `StreamReader` для читання файлу.
2	Розробити програму, яка зчитує текстовий файл рядок за рядком та рахує кількість слів у кожному рядку. Вивести результати на екран.
3	Створити програму, яка зчитує текстовий файл та знаходить всі унікальні слова, виводячи їх у відсортованому порядку. Використати `HashSet<string>`.
4	Розробити програму, яка зчитує великий текстовий файл та обчислює частоту кожного слова у файлі, відображаючи топ-10 найчастіших слів.
5	Створити програму, яка зчитує текстовий файл та визначає кількість рядків, які містять певне слово, введене користувачем.

Завдання 2

Створити програму для створення нового файлу та запису текстової інформації у файл.

Варіанти завдань

№	Опис завдання
1	Створити програму, яка створює новий текстовий файл і записує у нього кілька рядків тексту, введених користувачем. Використати клас `StreamWriter`.
2	Розробити програму, яка створює файл з поточною датою та часом у назві та записує у нього інформацію про користувача, таку як ім'я та вік.
3	Створити програму, яка приймає список продуктів від користувача та записує їх у CSV-файл з відповідними полями. Використати клас `StreamWriter` для запису даних.
4	Розробити програму, яка створює файл з результатами обчислень (наприклад, таблиця множення) та записує їх у форматі табличних даних.
5	Створити програму, яка створює текстовий файл і записує у нього лог-файли подій (наприклад, час запуску програми, виконані дії).

Додаткові завдання

Створення GUI-додатку для роботи з файлами: Розробити графічний інтерфейс користувача за допомогою Windows Forms або WPF, який дозволяє користувачу відкривати, редагувати та зберігати текстові файли. Забезпечити

функціонал пошуку та заміни тексту, а також відображення кількості слів у відкритому файлі.

Контрольні запитання

1. Як забезпечити безпечне відкриття та закриття файлів під час роботи з ними у програмі?
2. Які переваги надає використання конструкцій using при роботі з файлами?
3. Як обробляти виняткові ситуації, що виникають при доступі до файлів, щоб уникнути аварійного завершення програми?
4. Які методи класу StreamReader та StreamWriter використовуються для читання та запису рядків тексту?

Лабораторна робота №10 Тема: Делегати і події

Мета: навчитися створювати і використовувати делегати і події в програмі на C#.

Теоретичні відомості

Делегати в C# є типами, які представляють посилання на методи з певною сигнатурою. Вони дозволяють передавати методи як параметри, зберігати їх у змінних та викликати динамічно. Це ключовий механізм для реалізації подій та зворотних викликів (callbacks) у C#. Делегати забезпечують типову безпеку, тобто методи, на які вони посилаються, повинні відповідати їхній сигнатурі.

Події є особливим видом делегатів, які використовуються для повідомлення інших об’єктів про певні дії або зміни стану. Вони забезпечують механізм сповіщення (notification) між об’єктами, дозволяючи одному об’єкту повідомити іншим про події, що відбуваються. Використання подій сприяє розділенню відповідальностей та зменшенню зв’язності між компонентами програми.

Ключові поняття, пов’язані з делегатами та подіями:

Делегати: типи, що представляють методи з певною сигнатурою.

Події: механізм сповіщення, заснований на делегатах.

EventHandler: стандартний делегат для подій, який приймає два параметри: об’єкт відправника та інформацію про подію.

Лямбда-вирази: зручний спосіб визначення анонімних методів для підписки на події.

Правильне використання делегатів і подій дозволяє створювати гнучкі та розширювані програми, де компоненти можуть взаємодіяти один з одним без жорсткої залежності, що сприяє кращій підтримці та розвитку коду.

Завдання 1

Створити консольну програму з використанням делегатів.

Варіанти завдань

№	Опис завдання
1	Створити делегат, який приймає два числа і повертає їх суму. Реалізувати метод, який відповідає цій сигнатурі, та викликати його через делегат.
2	Створити делегат для методу, який виводить рядок на екран. Реалізувати кілька методів з різною логікою виводу та викликати їх через делегат.
3	Створити делегат, який приймає рядок і повертає його довжину. Реалізувати метод, що обчислює довжину рядка, і використовувати делегат для виклику цього методу.
4	Створити декілька делегатів для різних математичних операцій (додавання, віднімання, множення, ділення) та використовувати їх для обчислення виразів.
5	Створити делегат, який приймає масив чисел та повертає їх середнє значення. Реалізувати відповідний метод і викликати його через делегат.

Завдання 2

Створити програму з використанням подій.

Варіанти завдань

№	Опис завдання
1	Створити клас `Clock`, який має подію `OnTick`, що сповіщає про кожну секунду. Реалізувати підписку на цю подію та виводити повідомлення про тік-так кожної секунди.
2	Створити клас `Button`, який має подію `Clicked`. Реалізувати метод `Click`, що викликає цю подію, та створити підписників, які реагують на натискання кнопки різними повідомленнями.
3	Розробити клас `Timer`, який має події `Started`, `Stopped` та `Elapsed`. Реалізувати логіку запуску та зупинки таймера та сповіщення підписників про події.
4	Створити клас `Publisher`, який має подію `DataReceived`. Створити клас `Subscriber`, який підписується на цю подію та обробляє отримані дані.
5	Розробити клас `Alarm`, який має подію `OnAlarm`. Реалізувати логіку перевірки умов (наприклад, часу) і виклик події, коли умови виконуються.

Додаткові завдання

Розробити графічний інтерфейс користувача за допомогою Windows Forms або WPF, який містить кнопку. Реалізувати подію `Clicked` для цієї кнопки, яка при натисканні змінює текст на іншій частині інтерфейсу або виконує певну дію. Використати делегати для обробки подій.

Контрольні запитання

1. Що таке делегати в C# і які основні їхні особливості?
2. Як події відрізняються від делегатів і для чого вони використовуються?
3. Які модифікатори доступу можна використовувати з делегатами та подіями?
4. Як підписатися на подію і як відписатися від неї у C#? 5. У чому полягає різниця між Action, Func та власними делегатами?

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Albahari J. C# 10 in a Nutshell: The Definitive Reference. O'Reilly Media, Incorporated, 2022.
2. Troelsen A., Japikse P. Pro C# 10 With . NET 6: Foundational Principles and Practices in Programming. Apress L. P., 2022.
3. Developing on AWS with C# : A Comprehensive Guide on Using C# to Build Solutions on the AWS Platform, Noah Gift, James Charlesworth, 2022. 258 p.
4. Denis Panjuta, Jafar Jabbarzadeh. Learning C# Through Small Projects, Springer, 2024. 404 p.
5. Тулашвілі Ю.Й., Турбал Ю. В. Програмна реалізація алгоритму побудови туристичного маршруту та його інформаційного супроводу. Вісник Національного університету водного господарства та природокористування», серія «Технічні науки» Випуск 1 (97). 2022. С. 281-290. https://doi.org/10.31713/vt120220..
6. Sivakovska, O., Rudynets, M., Yashchuk, A., Redko, R., Zabolotnyi, O. (2022). Project Safety Management Systems of Students with 3D Game Development. In: Knapčíková, L., Peraković, D., Behúnová, A., Periša, M. (eds) 5th EAI International Conference on Management of Manufacturing Systems. EAI/Springer Innovations in Communication and Computing. Springer, Cham. https://doi.org/10.1007/978-3-030-67241-6_36.
7. Yurii Tulashvili; Viktor Kosheliuk. Orchestrating honeypot deployment in lightweight container platforms to improve security. International Science Journal of Engineering & Agriculture 2025-02-01. URL: https://doi.org/10.46299/j.isjea.20250401.01.

Об’єктно-орієнтоване програмування : методичні вказівки до лабораторних робіт для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Професійна освіта (комп'ютерні технології)» галузь знань – А Освіта, спеціальність – А5.39 Професійна освіта (Цифрові технології) денної та заоч. форм навч. / уклад. А.А. Ящук, Ю.Й. Тулашвілі. Луцьк: ЛНТУ, 2026. 31с.

Комп’ютерний набір А.А. Ящук
Редактор Ю.Й. Тулашвілі

Підп. до друку «__» _____ 2026 р. Формат 60х84/57. Папір офс.
Гарн. Таймс. Ум. друк. арк. 6,3.
Тираж 50 прим.

Відділ іміджу та промоції
Луцького національного технічного університету
43018 м. Луцьк, вул. Львівська, 75
Друк – ВІП Луцького НТУ