

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни

«Основи технології .Net»

для студентів спеціальності 123 Комп'ютерна інженерія
усіх форм навчання

Методичні вказівки до виконання лабораторних робіт з дисципліни «Основи технології .Net» для студентів спеціальності 123 Комп'ютерна інженерія усіх форм навчання. / Укл.: С.О. Сгадов – Запоріжжя: Національний університет «Запорізька політехніка», 2025. – 40с.

Укладач: С.О. Сгадов , ст. викладач

Рецензент: С.Ю. Скрупський, доцент, к.т.н.

Відповідальний
за випуск: С.О. Сгадов, ст. викладач

Затверджено
На засіданні кафедри
«Комп'ютерні системи та мережі»
Протокол № 7
від “28” 02 2025.

Рекомендовано до видання
НМК факультету КНТ
Протокол № 7
від “24” 03 2025.

ЗМІСТ

Вимоги до оформлення звіту з лабораторної роботи	4
1 Дані, вирази, ідентифікатори, оператори.....	5
1.1 Теоретичні відомості.....	5
1.2 Завдання	6
1.3 Контрольні запитання	9
2 Використання методів у мові C#	10
2.1 Теоретичні відомості.....	10
2.2 Завдання	11
2.3 Контрольні запитання	12
3 Класи та об'єкти	13
3.1 Теоретичні відомості.....	13
3.2 Завдання	14
3.3 Контрольні запитання	15
4 Індикатори та властивості в C#.....	16
4.1 Теоретичні відомості.....	16
4.2 Завдання	18
4.3 Контрольні запитання	18
5 Перевантаження операцій та операторів.....	19
5.1 Теоретичні відомості.....	19
5.2 Завдання	20
5.3. Контрольні запитання	21
6 Успадкування класів	22
6.1 Теоретичні відомості.....	22
6.2 Завдання	24
6.3 Контрольні запитання	28
7 Поліморфізм, інтерфейси, колекції	29
7.1 Теоретичні відомості.....	29
7.2 Завдання	33
7.3 Контрольні запитання	34
8 Делегати в C#	35
8.1 Теоретичні відомості.....	35
8.2 Завдання	38
8.3 Контрольні запитання	39
Рекомендована література	40

Вимоги до оформлення звіту з лабораторної роботи

Звіт є акуратно оформленим документом, що містить результати виконання лабораторного завдання. Звіт, підписаний студентом, надається викладачеві наприкінці лабораторного заняття. Він одночасно є допуском до виконання наступної лабораторної роботи. На титульному листі звіту слід вказати номер і назву роботи, прізвище і групу студента.

Звіт повинен містити:

- завдання;
- короткий опис роботи алгоритму;
- програмну реалізацію (мова C#);
- відповіді на контрольні запитання.

Готовий звіт надається викладачеві для затвердження в кінці заняття.

1 ЛАБОРАТОРНА РОБОТА №1

Дані, вирази, ідентифікатори, оператори

Мета роботи: ознайомитися з інтегрованим середовищем розробки Visual Studio, типами даних у C# та виразами, навчитися правильно використовувати типи даних, оператори та ідентифікатори.

1.1 Теоретичні відомості

1.1.1 Мова програмування C#

Мова програмування C# була створена компанією Microsoft у 1999 році. Початкова назва мови – COOL (скорочення від C-like Object-Oriented Language) – згодом була змінена на C#. Ініціатором розроблення мови став Андерс Гейлсберг (Anders Hejlsberg). Основною метою створення C# було усунення обмежень і недоліків, властивих іншим мовам програмування, які використовувалися на той час.

Для розроблення застосунків на C# оптимальним є використання інтегрованого середовища розробки (IDE) Visual Studio 2022, яке надає широкий спектр засобів для підвищення ефективності програмування.

1.1.2 Структура програми

Структура типової консольної програми наведена у Лістингу 1.1. Виконання починається з функції Main.

Лістинг 1.1

```
using System;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            текст програми
        }
    }
}
```

1.1.3 Приклад.

Завдання: Напишіть програму для виведення на екран таблиці квадратів і кубів цілих чисел в межах від -5 до 5.

Рішення: Розв'язання цього завдання здійснюється у два етапи. На першому етапі виводиться заголовок таблиці, а на другому - одним оператором циклу виводиться: значення базового числа, його квадрат та його куб. Усі значення визначаються як константи, тому в програмі відсутня частина введення початкових даних з клавіатури (Лістинг 1.2):

Лістинг 1.2

```
using System;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("x\tx^2\tx^3");
            for (int i = -5; i <= 5; i++)
                Console.WriteLine(i+"\t"+i*i+"\t"+Math.Pow(i, 3))
            Console.ReadKey();
        }
    }
}
```

1.2 Завдання

1.2.1 Завдання 1

Напишіть програми для розв'язання наступних задач:

Варіант 1. У цьому завданні необхідно написати програму, що оголосить дві змінні: перша змінна – змінна цілочисельного (int) типу зі значенням 99, друга змінна – змінна дійсного (double) типу зі значенням 0.3525. Перемножте ці дві змінні та результат виконання призначте двом новим змінним: цілочисельного та дійсного типу. Виведіть ці дві останні змінні окремо в новому рядку в такому порядку:

цілочисельна змінна, дійсна змінна.

Варіант 2. Напишіть програму, яка обчислює суму, добуток, різницю, частку та остачу від ділення двох заданих чисел.

Варіант 3. Напишіть програму, яка конвертує суму грн. у євро.

Варіант 4. Напишіть програму з умовою, що суму у грн. та курс валюти задає користувач.

Варіант 5. У C# немає функції, яка обчислює квадрат числа. Написати програму, яка обчислює квадрат введенного числа.

Варіант 6. Дано сторону квадрата (вводиться з клавіатури користувачем). Визначити довжину діагоналі.

Варіант 7. Дано катети прямокутного трикутника (вводяться з клавіатури користувачем). Визначити довжину гіпотенузи.

1.2.2 Завдання 2

Уважно ознайомтеся з прикладами та виконайте свій варіант.

Варіант 1. Виведіть на екран таблицю $\cos(x) \cdot \sin(y)$ при зміні y від 0° до 90° з кроком 15° (виведення по горизонталі) і зміні x від 0° до 180° з кроком 30° (виведення по вертикалі).

Варіант 2. Виведіть на екран таблицю $1 + \sin(x)$, $1 - \sin(x)$, $1 + \tan(x)$, $1 - \tan(x)$ (виведення по горизонталі) при зміні аргументу від 0° до 180° з кроком 15° (виведення по вертикалі). В програмі передбачити неможливість обчислення $\tan(90^\circ)$.

Варіант 3. Виведіть на екран таблицю $\sin(x) + \sin(y)$ при зміні y від 0° до 90° з кроком 30° (виведення по горизонталі) і зміні x від 0° до 90° з кроком 10° (виведення по вертикалі).

Варіант 4. Виведіть на екран таблицю $\ln(x)$, e^x при зміні аргументу від 0 до e з кроком $e/20$. В програмі передбачити неможливість обчислення $\ln(0)$.

Варіант 5. Виведіть на екран таблицю $\tan(x)$, $\cot(x)$ (по горизонталі) при зміні аргументу від 0° до 90° з кроком 5° (виведення по вертикалі). В програмі передбачити неможливість ділення на 0.

Варіант 6. Виведіть на екран таблицю $1/\sin(x)$, $1/\cos(x)$, $1/(\sin(x) \cdot \cos(x))$ (по горизонталі), при зміні аргументу від 0° до 180° з кроком 10° (виведення по вертикалі). В програмі передбачити неможливість ділення на 0.

Варіант 7. Виведіть на екран таблицю множення у вигляді квадрату Піфагора. Таблицю вивести при зменшенні множників від 10 до 1.

Варіант 8. Виведіть на екран таблицю множення на 7 при зростанні першого множника від 1 до 10.

Варіант 9. Виведіть на екран таблицю множення на 8 при зменшенні першого множника від 10 до 1.

Варіант 10. Виведіть на екран таблицю квадратів перших N цілих додатних непарних чисел. Кількість таких чисел N повинна вводитись під час виконання програми.

Варіант 11. Виведіть на екран таблицю ступенів двійки від 0 до 10.

Варіант 12. Виведіть на екран таблицю значень функції $y = -2,4x^2 + 5x - 3$ у діапазоні від -2 до 2 з кроком 0,5.

Варіант 13. Виведіть на екран середнє арифметичне значення довільної послідовності N дійсних чисел. Кількість чисел і їхні значення вводяться з клавіатури.

Варіант 14. Виведіть таблицю вартості яблук в діапазоні від 50г до 1кг з кроком 50г. Вартість 1кг яблук повинна вводитися під час роботи програми.

Варіант 15. Виведіть на екран таблицю значень функції $y = |x-2| + |x-1|$ у діапазоні від -4 до 4 з кроком 0,5.

Варіант 16. Виведіть на екран таблицю $\cos(x) + \sin^2(y)$ при зміні y від 0° до 90° з кроком 30° (виведення по горизонталі) і зміні x від 0° до 180° з кроком 15° (виведення по вертикалі).

Варіант 17. Виведіть на екран таблицю $\sin(x)$, $\cos(x)$, $\operatorname{tg}(x)$, $\operatorname{ctg}(x)$ (виведення по горизонталі) при зміні аргументу від 0° до 180° з кроком 10° (виведення по вертикалі). В програмі передбачити неможливість обчислення $\operatorname{tg}(90^\circ)$, $\operatorname{ctg}(0^\circ)$, $\operatorname{ctg}(180^\circ)$.

Варіант 18. Виведіть на екран таблицю $\sin(x) \cdot \sin(y)$ при зміні y від 0° до 90° з кроком 15° (виведення по горизонталі) і зміні x від 0° до 90° з кроком 15° (виведення по вертикалі).

Варіант 19. Виведіть на екран таблицю x^a , a^x при зміні a від 0 до e з кроком $e/20$. Значення дійсної змінної x повинно вводитися з клавіатури.

Варіант 20. Виведіть на екран таблицю x^n при зміні n від 1 до 5 (виведення по горизонталі) та при зміні x від 1 до 9 з кроком 2 (виведення по вертикалі).

1.3 Контрольні запитання

1. Які є основні (базові) типи даних в мові C#?
2. Які є особливості використання цілочисельних типів даних?
3. Які є особливості типів даних з плаваючою комою (дійсні типи)?
4. Які є базові символічні типи даних в мові C#?
5. Які види або категорії операторів є в мові C#?
6. Які оператори належать до категорії арифметичних операторів?
7. Формат оператора циклу `for`. Поясніть поняття ініціалізація, вираз, модифікації.
8. Як працюють вкладені один в один оператори циклу?
9. Поясніть поняття: "тіло циклу", "ітерація".
10. Поясніть поняття: "початкові установки", "параметр циклу".
11. Вкажіть формат оператора циклу `do...while`, наведіть блок-схему алгоритму, який він реалізує.
12. Вкажіть формат оператора циклу `while`, наведіть блок-схему алгоритму, який він реалізує.
13. Доки виконуватиметься оператор циклу `while`?
14. Що є спільним і у чому полягає різниця між операторами циклу `do...while` та `while`?
15. Яким чином можна впливати на умову завершення операторів циклу `do...while` та `while`? Для чого це слід робити?

2 ЛАБОРАТОРНА РОБОТА №2

Використання методів у мові C#

Мета роботи: навчитися створювати та застосовувати з методи.

2.1 Теоретичні відомості

2.1.1 Методи

Практично кожен застосунок, розроблений на базі Microsoft.NET Framework та з використанням мови C#, містить велику кількість методів, кожен з яких виконує свою певну функцію. Деякі методи є необхідними для забезпечення працездатності програми. Наприклад, усі застосунки для персональних комп'ютерів, написані на C#, повинні мати метод Main, який є точкою входу в програму. Під час запуску такого застосунку Common Language Runtime (CLR) першочергово виконує саме метод Main.

Елементи сигнатури методу визначаються:

- ідентифікатор доступу – відповідає за доступність методу (хто може викликати цей метод):

- 1) private – дозволяє викликати метод тільки всередині класу чи структури;
- 2) protected – дозволяє викликати цей метод з поточного класу або його спадкоємців;
- 3) public – дозволяє викликати цей метод з від усюди;
- 4) internal – дозволяє викликати цей метод всередині збірки;

- static – визначає, що метод статичний член класу, а не член якогось об'єкта;

- тип даних, що повернеться – визначає, який тип даних буде повернуто – якщо метод нічого не повертає, потрібно вказати void;

- назва методу – усі методи повинні мати назви;

- параметри – це список параметрів, що метод приймає як вхідні дані – якщо параметрів декілька, їх треба розділити комою.

Приклад простого методу:

```
public Boolean StartService(string serviceName)
{ реалізація методу }
```

де `public` – ідентифікатор доступу, `Boolean` – тип даних, що буде повернуто, `StartService` – назва методу, `string serviceName` – параметр. Зверніть увагу, що параметр складається з типу даних та назви параметра.

2.2 Завдання

Створіть програму, яка містить метод, вказаний у вашому варіанті, та надрукує результат його виклику.

Варіант 1. Напишіть функцію `Procent`, яка повертає відсоток від отриманого як аргумент числа.

Варіант 2. Напишіть функцію `Factorial` та програму, яка використовує цю функцію для виведення таблиці факторіалів.

Варіант 3. Напишіть функцію `Dohod`, яка обчислює дохід за вкладом. Вхідними даними для функції є: величина вкладу, процентна ставка (річних) і термін вкладу (кількість днів).

Варіант 4. Напишіть функцію `glasn`, яка повертає 1, якщо символ, отриманий функцією як аргумент, є голосною буквою алфавіту, і 0 – в іншому випадку.

Варіант 5. Напишіть функцію `sogl`, яка повертає 1, якщо символ, отриманий функцією як аргумент, є приголосною буквою алфавіту, і 0 – в іншому випадку.

Варіант 6. Напишіть функцію `frame`, яка виводить на екран рамку. Як параметри функції повинні передаватися координати лівого верхнього кута і розмір рамки.

Варіант 7. Напишіть функцію, яка обчислює значення $a*b$. Числа a й b можуть бути будь-якими дробовими позитивними числами.

Варіант 8. Напишіть функцію, що забезпечує розв'язання квадратного рівняння. Параметрами функції повинні бути коефіцієнти та корені рівняння. Значення, які обраховує функція, мають передаватися в програму, яка викликається з інформацією про наявність коренів рівняння: 2 – два різних корені, 1 – корені однакові, 0 – рівняння не має коренів. Крім того, функція повинна перевіряти коректність вихідних даних. Якщо вихідні дані невірні, то функція повинна повертати – 1.

Варіант 9. Напишіть функцію, яка виводить рядок, що складається з однакових символів. Довжина рядка та символ є параметрами процедури.

Варіант 10. Напишіть функцію, яка обчислює об'єм і площу поверхні паралелепіпеда.

2.3 Контрольні запитання

1. Що таке метод у мові програмування C#?
2. Чим відрізняються прості методи від статичних?
3. Як оголошуються методи в C#?
4. У чому полягає призначення параметрів `ref` та `out`?
5. Що таке перевантаження методів і як воно реалізується?
6. Як реалізується підтримка змінної кількості параметрів у методі?
7. Що є віртуальні методи та для чого вони використовуються?
8. У чому полягає суть підміни методів (`overriding`) у спадкуванні?

3 ЛАБОРАТОРНА РОБОТА №3

Класи та об'єкти

Мета роботи: Навчитися описувати класи мовою C# та створювати екземпляри класів. Сформувати навички структурування інформації за допомогою класів.

3.1 Теоретичні відомості

3.1.1 Оголошення класу та створення об'єкта

Кожен об'єкт характеризується певними властивостями та методами. Властивість – це характеристика об'єкта, наприклад, маса, висота, кількість ніг. Об'єкти можуть мати нащадків. Метод – це операція або дія, яку можна виконати з об'єктом. При оголошенні класу визначаються об'єкти, які до нього входять, надаючи загальний опис їхніх спільних рис.

Опишемо клас об'єктів Тварина (Animal). Опис класу здійснюється службовим словом `class`:

Лістинг 3.1 - Оголошення класу та створення об'єкта.

```
using System;
namespace c1 {
class Animal
{public string kindOfAnimal;
  public string name;
  public int numberOfLegs;
  public float mass;
  public void ShowAnimal()
  {
    Console.WriteLine("Ім'я тварини {0}", this.name);
    Console.WriteLine("Її маса {0} кг", this.mass);
  }
}

class Program
{
public static void Main(string[] args)
{
```

```

Animal Horse = new Animal(); // створення об'єкта
Horse.name = "Плотва";
Horse.mass = 300;
Console.WriteLine("Ім'я тварини {0}", Horse.name);
Console.WriteLine("Її маса {0} кг", Horse.mass);
Console.ReadLine();
}
}
}

```

3.2 Завдання

Описати клас відповідно до обраного варіанта. Реалізувати консольну програму, яка створює екземпляр цього класу та викликає його методи.

Варіант 1. Створіть клас `tovar` (товар) з полями `price` (ціна), `weight` (вага), `date` (дата випуску), `kind` (їжа, канцтовари, будівельний, запчастини тощо). Створіть метод-конструктор для введення даних властивостей з клавіатури.

Варіант 2. Створіть клас `computers` (комп'ютери) з полями `price` (ціна), `processor` (тип процесора), `hddSize` (об'єм жорсткого диска), `ramSize` (обсяг оперативної пам'яті), `frequency` (тактова частота). Створіть метод-конструктор для введення даних властивостей з клавіатури.

Варіант 3. Створіть клас `student` (студент) з полями `firstName` (ім'я), `lastName` (прізвище), `faculty` (факультет), `yearIn` (рік вступу), `курс` (курс). Створіть метод-конструктор для введення даних властивостей з клавіатури.

Варіант 4. Створіть клас `company` (фірма) з полями `name` (назва), `year` (рік заснування), `gem` (короткий опис), `budget` (уставний капітал). Створіть метод-конструктор для введення даних властивостей з клавіатури.

Варіант 5. Створіть клас `book` (книга) з полями `price` (ціна), `size` (кількість сторінок), `date` (дата видання), `kind` (жанр), `index` (шифр, наприклад, К32-4 тощо). Опишіть метод-конструктор для введення даних властивостей з клавіатури.

Варіант 6. Створіть клас `employer` (робітник) з полями `firstName` (ім'я), `lastName` (прізвище), `nameTseh` (назва цеху де працює), `birthDay`

(дата народження), stag (стаж). Створіть метод для виводу значень усіх полів на консоль.

Варіант 7. Створіть клас car (легковий автомобіль) з полями price (ціна), power (потужність), date (дата випуску), kind (Mercedes, Lanos, Opel тощо), maxHuman (пасажиромісткість). Створіть метод для виводу значень усіх полів на консоль.

Варіант 8. Створіть клас weapon (зброя) з полями kind (рушниця, пістолет, автомат, кулемет тощо), weight (маса), date (дата випуску), speed (початкова швидкість польоту кулі), mark (марка, наприклад, Beretta, Rhino, AR-416). Створіть метод для виводу значень усіх полів на консоль.

Варіант 9. Створіть клас plane (літак) з полями name (марка, наприклад, Boeing, F-14), length (довжина), width (ширина), power (потужність), date (дата випуску), kind (гвинтовий, турбореактивний), enginCount (кількість двигунів), maxSpeed (максимальна швидкість), top (максимальна висота польоту). Створіть метод для виводу значень усіх полів на консоль.

Варіант 10. Створіть клас truck (вантажний автомобіль) з полями price (ціна), power (потужність), date (дата випуску), kind (Mercedes, MAN TGS, FAUN SLT 50-3 тощо), maxCargo (вантажність). Створіть метод для виводу значень усіх полів на консоль

3.3 Контрольні запитання

1. Поясніть поняття класу та об'єкта.
2. Опишіть синтаксис оголошення класу.
3. Поясніть поняття поля класу. Які специфікатори полів вам відомі?
4. Поясніть поняття методу класу.
5. Опишіть синтаксис оголошення методу класу.
6. У якому випадку в заголовку методу вказується тип void?
7. Опишіть синтаксис оголошення параметрів методу.
8. Поясніть зв'язок між параметрами та аргументами методу.
9. Які існують способи передачі параметрів та типи параметрів?
10. У чому полягає відмінність використання статичних методів?
11. В яких випадках використовується ключове слово this?
12. Поясніть термін «конструктор» та його властивості.

4 ЛАБОРАТОРНА РОБОТА №4

Індексатори та властивості в C#

Мета роботи: навчитися використовувати властивості у класах C#.

4.1 Теоретичні відомості

4.1.1 Властивості

Властивості – це функції або їх пари, які клієнтський код може використовувати так само як відкриті поля класу. Вони створені для того, щоб поводитися як звичайні поля, але при цьому виконують додаткову логіку. Щоб визначити властивість в C#, використовується наступний синтаксис:

```
public string Myproperty
{
    get
    {
        return Myfield;
        //повертає значення поля Myfield
    }
    set
    {
        Myfield=value;
        //задаємо значення поля Myfield
    }
}
```

Рекомендується починати імена полів з маленької літери, а імена відповідних властивостей – з великої, зберігаючи назву поля.

Існує можливість створення властивостей лише для читання (без set) або лише для запису (без get). Для set та get можна використовувати різні модифікатори доступу, але модифікатор одного з них повинен збігатися з модифікатором усієї властивості.

4.1.2 Індексатори

Індексція дозволяє об'єктам поводити себе подібно до масивів або колекцій, звертаючись до членів класу як до елементів масиву.

Наприклад, якщо об'єкт складається з кількох елементів (як `listbox` з рядків), індексція дає змогу звертатися до них за індексом:

```
class Stringlist
{
private string[] list;
public string this[int index]
{ get{ return list[index];}
  set{ list[index] = value;}
}
}
```

Індексатор – це властивість з іменем `this`, квадратними дужками й індексом у них. Для використання індексатора, звертаємося до об'єкта:

```
Stringlist mylist = new Stringlist();
mylist[3] = "ok"; // індексатор записує
string mystring = mylist[3]; // індексатор читає
```

Індексатори можуть бути статичними членами класу та підлягати перевантаженню для різних типів індексів. Індексатори не зберігають значення, тому їх не можна передавати як параметри `ref` та `out`.

При оголошенні індексаторів необхідно визначити щонайменше один параметр. Можна визначити кілька параметрів для одного індексатора:

```
class Multipleparameters
{
public string this[int one, int two]
{ get{...} set{...} }
}
...
Multipleparameters mp = new Multipleparameters();
string s = mp[2,3];
}
```

4.2 Завдання

Завдання 1. У класі з лабораторної роботи 3 замінить усі поля на властивості. Напишіть тестовий приклад. Перевірте, що виводить програма.

Завдання 2. У класі з лабораторної роботи 3 напишіть індексатор для читання полів за імені. Напишіть тестовий приклад. Перевірити, що виводить програма.

4.3 Контрольні запитання

1. Яким чином можна оголосити властивість тільки для читання?
2. Чи може клас містити індексатор, що залежить від двох параметрів різного типу?
3. Як у класі оголосити індексатор тільки для читання?

5 ЛАБОРАТОРНА РОБОТА №5

Перевантаження операцій та операторів

Мета роботи: навчитися здійснювати перевантаження операторів для типів користувача, вивчити операції, що підлягають перевантаженню, одержати практичні навички перевантаження операторів.

5.1 Теоретичні відомості

Перевантаження операторів дає змогу програмістам задавати власну реалізацію вже наявних операторів C# для випадків, коли один або обидва операнди належать до визначеного користувачем типу. Наприклад, оператор додавання (+) для класу, що представляє вектор, може бути визначений як операція векторного додавання, а не звичайного числового додавання. Це дозволяє писати код, який є більш виразним та інтуїтивно зрозумілим, особливо при роботі з математичними чи іншими абстрактними типами даних. Перевантаження операторів є однією з форм поліморфізму, оскільки один і той же оператор може мати різну поведінку залежно від типів операндів, до яких він застосовується.

У C# існує набір операторів, які можуть бути перевантажені для користувацьких типів. Ці оператори можна класифікувати на кілька груп:

- унарні оператори: до цієї групи належать оператори, що діють на один операнд:

- 1) арифметичні: + (унарний плюс), - (унарний мінус);
- 2) логічні: ! (логічне заперечення);
- 3) побітові: ~ (побітове доповнення);
- 4) інкремент та декремент: ++, -- (префіксна та постфіксна форми потребують окремого перевантаження);

5) оператори true та false: ці оператори повинні бути перевантажені разом і використовуються для визначення логічного значення істинності об'єкта користувацького типу;

- бінарні арифметичні оператори, які виконують арифметичні дії над двома операндами:

- 5) додавання: +;

- 6) віднімання: -;
- 7) множення: *;
- 8) ділення: /;
- 9) остача від ділення: %.

- бінарні побітові та логічні оператори, які виконують побітові або логічні операції над двома операндами:

- 1) побітове і: &;
- 2) побітове або: |;
- 3) побітове виключне або: ^;
- 4) зсув вліво: <<;
- 5) зсув вправо: >>.

- оператори порівняння, які використовуються для порівняння двох операндів:

- 1) рівність: ==;
- 2) нерівність: !=;
- 3) більше: >;
- 4) менше: <;
- 5) більше або дорівнює: >=;
- 6) менше або дорівнює: <=.

Для оголошення перевантаження оператора використовується ключове слово `operator`. Оголошення оператора повинно бути `public` та `static`. Унарний оператор приймає один вхідний параметр, а бінарний – два. При цьому, щонайменше один з параметрів повинен бути типу `T` або `T?`, де `T` – це тип, що містить оголошення оператора. `<=`. Важливо зазначити, що оператори порівняння повинні перевантажуватися парами: `==` з `!=`, `<` з `>`, та `<=` з `>=`. Якщо один оператор з пари перевантажений, інший також повинен бути перевантажений.

5.2 Завдання

Створіть консольний застосунок. Розробіть клас з перевантаженням операторів `+`, `-`, `/`, `*`, `%`, `==`, `>` на свій розсуд (наприклад, порівняння можна провести за ім'ям). Продемонструйте результати роботи операцій, які Ви перевантажили.

Варіант 1. Створіть клас "Вектор у просторі" з перевантаженням операторів `+`, `-`, `*` (скалярний добуток) та `==`.

Варіант 2. Створіть клас "Матриця з перевантаженням операторів `+`, `-`, `*`, `==`."

Варіант 3. Створіть клас "Співробітник" з перевантаженням операторів $!=$, $==$.

Варіант 4. Створіть клас "Комп'ютер" з перевантаженням операторів $!=$, $==$, $>$, $<$.

Варіант 5. Створіть клас "Товар" з перевантаженням операторів $+$, $-$, $==$, $!=$.

Варіант 6. Створіть клас "Об'єкт нерухомості" з перевантаженням операторів $==$, $!=$.

Варіант 7. Створіть клас "Комплексне число" з перевантаженням операторів $+$, $-$, $/$, $*$, $==$.

Варіант 8. Створіть клас "Куб" з перевантаженням операторів $+$, $-$, $/$, $==$.

Варіант 9. Створіть клас "Автомобіль" з перевантаженням операторів $+$, $-$, $==$.

Варіант 10. Створіть клас "Студент" з перевантаженням операторів $==$, $>$, $<$.

Варіант 11. Створіть клас "Книга" з перевантаженням операторів $==$, $>$, $<$.

Варіант 12. Створіть клас "Дисципліна (предмет)" з перевантаженням операторів $==$, $>$, $<$.

5.3. Контрольні запитання

1. Що є конструктор?
2. Що є перевантаження операторів? Коли застосовується даний механізм?
3. Допустимо, що для класу створена перевантажена операція додавання. Чи може бути створена ще одна операція додавання для даного класу?
4. Які операції не можна перевантажити?
5. Які операції необхідно перевантажувати попарно?

6 ЛАБОРАТОРНА РОБОТА №6

Успадкування класів

Мета роботи: вивчити механізм успадкування класів, навчитися оголошувати похідні класи та створювати ієрархії класів, а також використовувати інтерфейси.

6.1 Теоретичні відомості

Успадкування дозволяє класу-нащадку отримувати (успадковувати) дані й функціональність базового класу. До класу-нащадку можна додавати нові поля та методи, окрім успадкованих від батьківського класу. Клас-нащадок може перевизначити (заміщати) методи батьківського класу. Успадкування від класу також називають розширенням базового класу. Якщо клас А успадковує від класу В, то клас А називають нащадком (похідним класом), а В – предком (базовим класом). Синтаксично це пишеться в такий спосіб:

```
class A:B {...}
```

Клас-нащадок успадковує всі члени (поля, методи, властивості тощо) базового класу, крім конструкторів і деструктора.

Усі `public`-члени базового класу залишаються `public` у нащадку. `private`-члени також успадковуються, але доступні лише всередині базового класу. Рівень доступу класу-нащадку не може бути вищим (більш доступним), ніж рівень доступу базового класу.

Для виклику конструктора базового класу із нащадку використовується ключове слово `base`:

```
A(список параметрів): base(параметри для передачі в
конструктор базового класу)
{
    // Реалізація конструктора нащадка
}
```

Ключове слово `base` надає доступ до членів базового класу (зокрема, до його конструкторів).

Якщо конструктор нащадку не викликає явно конструктор базового класу, компілятор автоматично додає виклик конструктора базового класу без параметрів (`base()`). Тому для конструктора за замовчуванням цей виклик можна не прописувати. Якщо ж базовий клас не має конструктора без параметрів, або потрібно викликати інший конструктор базового класу, його необхідно викликати явно за допомогою `base(...)`:

```
class Token
{
protected Token(string name) { ... }
}
class Commenttoken: Token
{ public Commenttoken(string name) {... } //помилка
}
```

У прикладі вище одержимо помилку при компіляції, тому що буде викликатися конструктор за замовчуванням, але він є відсутній у класі `Token`. Для правильної роботи необхідно записати конструктор у наступному виді:

```
public Commenttoken(string name) : base(name)
{ ... }
```

Якщо всі конструктори базового класу є `private`, то від цього класу не можна успадковувати (а отже, і створювати конструктор класу-нащадку).

Ключове слово `new` приховує як віртуальні, так і невіртуальні методи, розв'язує проблему збігу імен, приховує методи з однаковою сигнатурою.

У `C#` можна заборонити успадкування від класу за допомогою ключового слова `sealed`.

```
public sealed class String {...}
public class Mystr:String;
//помилка - від класу String успадковувати не можна
```

Абстрактні класи використовуються для створення базових класів, що містять часткову реалізацію, яка доповнюється або

завершується в класах-нащадках. Абстрактний клас оголошується за допомогою ключового слова `abstract`. Загалом, правила створення абстрактного класу подібні до правил для звичайних класів.

Тільки абстрактні класи можуть містити абстрактні члени. Абстрактний член не має реалізації (тіла). Абстрактні методи самі є неявно віртуальними та мають бути перевизначені (`override`) у першому неабстрактному класі-нащадку. Абстрактний метод також може перевизначити (`abstract override`) віртуальний або абстрактний метод базового класу:

```
class Token
{
    public virtual string Name( ) { ... }
}
abstract class Force: Token
{
    public abstract override string Name( );
}
```

6.2 Завдання

Виконайте завдання 1 та 2 згідно з варіантом. Створіть вказані інтерфейси та класи. Напишіть програму, що демонструє функціонування класів.

6.2.1 Завдання 1

Створіть інтерфейс `ICipher`, який визначає методи підтримки шифрування рядків. В інтерфейсі оголошуються два методи `encode()` і `decode()`, які використовуються для шифрування й дешифрування рядків, відповідно.

Потім створіть клас `ACipher`, що реалізує інтерфейс `ICipher`. Клас шифрує рядок за допомогою зсуву кожного символу на одну "алфавітну" позицію вище. Наприклад, у результаті такого зсуву буква А стає буквою Б.

Ще створіть клас `BCipher`, що теж реалізує інтерфейс `ICipher`. Клас шифрує рядок, виконуючи заміну кожної букви, що розташована в алфавіті на *i*-й позиції, на букву того ж регістру, розташовану в

алфавіті на і-й позиції з кінця алфавіту. Наприклад, буква В замінюється на букву Є.

6.2.2 Завдання 2

Створіть систему класів відповідно до обраного варіанта. Реалізуйте метод виведення на екран, який має відображати стан усіх полів об'єкта. Проаналізуйте доцільність оголошення методів в інтерфейсі. Визначте, чи доцільним є створення абстрактного класу; укажіть, які поля та методи мають бути оголошені в ньому. Визначте, які методи повинні бути оголошені як *virtual*, а які — перевантаженими (*overloaded*). Протестуйте реалізацію методу виведення для кожного створеного класу.

Варіант 1.

Клас: тех. засіб

- а) автомобіль;
 - 1) вантажівка;
 - 2) легковий автомобіль;
- б) гвинтокрил.

Варіант 2.

Клас: Комп. техніка

- а) сервер;
- б) ноутбук;
- в) ПК.

Варіант 3.

Клас: геометрія

- а) чотирикутник;
 - 1) квадрат;
 - 2) ромб;
- б) трикутник.

Варіант 4.

Клас: видання

- а) книга;
- б) журнал;
- в) ел. ресурс.

Варіант 5.

Клас: мультимедіа

- а) зображення;

- 1) растрове;
- 2) векторне;
- б) відео.

Варіант 6.

Клас: машини

- а) авіазасіб;
 - 1) вертоліт;
 - 2) винищувач;
- б) вантажівка.

Варіант 7.

Клас: рослина

- а) трава;
- б) дерево;
- в) чагарник.

Варіант 8.

Клас: програмні засоби

- а) OS;
 - 1) Free_OS;
 - 2) Commercial_OS;
- б) прикладні.

Варіант 9.

Клас: комп'ютер

- а) мобільний;
 - 1) планшет;
 - 2) ноутбук;
- б) стаціонарний.

Варіант 10.

Клас: електроніка

- а) телефон;
- б) фотоапарат;
- в) планшет.

Варіант 11.

Клас: тех. засіб

- а) автомобіль;
 - 1) вантажний;
 - 2) легковий;
- б) вертоліт.

Варіант 12.

Клас: побутова техніка

- а) телевізор;
- б) праска;
- в) мікрохвильова піч.

Варіант 13.

Клас: товари

- а) ручка;
- б) швидкозшивач;
- в) зошит.

Варіант 14.

Клас: нерухомість

- а) квартира;
- б) будинок;
- в) гараж.

Варіант 15.

Клас: машинобудування

- а) важке;
- б) середнє;
- в) сільське;
- г) легке.

Варіант 16.

Клас: частка

- а) фотон;
- б) електрон;
- в) протон.

Варіант 17.

Клас: фахівець

- а) програміст;
- б) лікар;
- в) льотчик.

Варіант 18.

Клас: меблі

- а) стіл;
- б) м'які меблі;
- в) шафа.

6.3 Контрольні запитання

1. У чому різниця між рівнями доступу `public`, `private` і `protected` для членів класу?
2. Як перевизначити метод базового класу в класі-нащадку?
3. Що таке абстрактний клас?
4. Що таке інтерфейс? У чому відмінність інтерфейсу від абстрактного класу?
5. Чи допустиме в C# множинне успадкування класів? А інтерфейсів?
6. Що таке успадкування реалізації (успадкування від класу)? Як синтаксично описати успадкування класу?
7. Для чого використовується ключове слово `base`?
8. За яких умов можна перевизначити метод базового класу в похідному класі?
9. Що таке властивості класу? Що таке поля (дані) класу?
10. Для чого використовується ключове слово `virtual`?
11. Для чого використовується ключове слово `override`?
12. Скільки безпосередніх базових класів може мати клас у C#?

7 ЛАБОРАТОРНА РОБОТА №7

Поліморфізм, інтерфейси, колекції

Мета роботи: навчитися створювати та реалізовувати інтерфейси, та використовувати стандартні колекції.

7.1 Теоретичні відомості

7.1.1 Створення інтерфейсів

Інтерфейси схожі на класи, але без реалізації. Вони визначають множину характеристик і типів поведінки, задаючи сигнатури для методів, властивостей, подій без їхньої імплементації. Якщо клас реалізує інтерфейс, він має надати реалізацію для кожного члена інтерфейсу. Реалізуючи інтерфейс, клас гарантовано має повністю надати функціональність, задану в інтерфейсі. Зверніть увагу на важливу різницю у використанні інтерфейсів. Клас реалізує інтерфейс, на відміну від успадкування базового класу. Ви можете уявити інтерфейс як контракт. Реалізуючи певний інтерфейс, клас гарантує, що він матиме функціональність, задану в контракті.

Синтаксис визначення інтерфейсу схожий на синтаксис визначення класу. Ви використовуєте ключове слово `interface` для визначення інтерфейсу, як це показано в прикладі нижче:

```
// Визначення інтерфейсу
public interface IBeverage
{
    // Methods, properties, events, and indexers go
    // here.
}
```

Згідно з конвенцією написання коду в C#, імена інтерфейсів мають починатися з літери I (наприклад, `IShape`, `IDisposable`). Хоча це й не є обов'язковою вимогою мови, дотримання цього правила сприяє читабельності та уніфікації коду.

Визначення інтерфейсів можуть містити модифікатор доступу, що схоже на задання класів. Можливо використовувати модифікатори доступу під час декларації вашого інтерфейсу.

Інтерфейс визначає сигнатуру членів, але не задає імплементацію. Інтерфейс може включати методи, властивості, події та індексатори.

Члени інтерфейсу не включають модифікатори доступу. Усі члени є `public`. Інтерфейси не можуть включати членів, що відносяться лише до внутрішньої функціональності класу, такі як поля, константи, оператори або конструктори.

7.1.2 Явна та неявна реалізації

Створюючи клас, який реалізує інтерфейс, ви обираєте явний або неявний спосіб реалізації. Для неявної реалізації інтерфейсу ви реалізуєте кожен член інтерфейсу із сигнатурою, що відповідає визначенню в інтерфейсі. Для явної реалізації інтерфейсу ви вказуєте повне ім'я кожного члена (з іменем інтерфейсу), щоб було зрозуміло, якому саме інтерфейсу належить цей член.

Приклад неявної реалізації:

```
interface ILogger {
    void Log(string message);
}
class FileLogger : ILogger {
    public void Log(string message) {
        // Реалізація логування
    }
}
```

Приклад явної реалізації:

```
class FileLogger : ILogger {
    void ILogger.Log(string message) {
        // Реалізація логування
    }
}
```

Явна реалізація дозволяє:

- виключити визначення інтерфейсу із класу, якщо він не цікавий користувачам класу;
- забезпечити класу кілька визначень різних методів інтерфейсів однакової сигнатури.

У більшості випадків вибір між явною та неявною реалізацією інтерфейсу є лише питанням стилю кодування.

7.1.3 Колекції

Коли ви створюєте багато елементів одного типу (чи то цілі числа, рядки, чи об'єкти класу), може знадобитися спосіб обробляти ці елементи разом, подібно до масиву. Може виникнути потреба підраховувати кількість елементів, додавати або видаляти елементи, а також переглядати їх по черзі. Для цих цілей можна використовувати колекції.

Колекції – це важливий інструмент для обробки багатьох елементів. Вони також потрібні й для розробки графічних додатків. Такі елементи керування, як спадний список чи меню, зазвичай прив'язуються до даних у колекції

Всі колекції мають спільні властивості. Для обробки колекції елементів ви повинні мати можливість:

- додавати елементи до колекції;
- видаляти елементи з колекції;
- отримувати конкретні елементи з колекції;
- підраховувати кількість елементів в колекції;
- перебрати елементи в колекції один за одним.

Кожен клас колекції в C# надає методи та властивості, які допомагають здійснити ці основні операції. Але крім цих операцій ви захочете керувати колекціями по-різному залежно від конкретних вимог програми. Колекції поділяються на категорії.

Клас List – клас списку як одновимірний масив, який динамічно розширюється в міру додавання елементів.

Клас Dictionary (Словник) зберігає колекцію пар ключ/значення. Кожен елемент в колекції складається з двох об'єктів – ключа і значення. Значення є об'єкт, який ви хочете зберігати та переглядати, а ключ є об'єктом, який ви використовуєте для індексування та пошуку значення. У більшості класів словник, ключ повинен бути унікальним.

Клас Queue (Черга) являє собою колекції об'єктів, що умовно називається "перший прийшов – перший пішов", як і відбувається зазвичай (якщо відкинути випадки "я тільки запитати") у звичайних чергах. Ми отримуємо елементи з колекції в тому ж порядку, в якому вони були додані.

Клас Stack (Стек) являє собою колекцію об'єктів, що умовно називається "останній прийшов – перший пішов". Елемент, який ви додали в колекцію останнім є першим елементом, який ви будете переглядати..

Найбільш поширеним неугальним класом колекції є ArrayList. ArrayList зберігає елементи як лінійну колекцію. Наступний приклад демонструє як додавати та проходити по колекції ArrayList:

```
//Додавання і проходження по елементах ArrayList
// Створення ArrayList колекції.
ArrayList beverages = new ArrayList();
// Створення елементів, які ми хочемо додати до
//колекції
Coffee coffee1 =new Coffee(4,"Arabica","Columbia");
Coffee coffee2 =new Coffee(3,"Arabica","Vietnam");
// Додавання елементів до колекції.
// Елементи неявно приводяться до типу Object,
// коли ви їх додаєте.
beverages.Add(coffee1);
beverages.Add(coffee2);
// Проходження по елементах у колекції.
// Елементи повинні бути явно перетворені назад до
//початкового типу.
Coffee firstCoffee = (Coffee)beverages[0];
Coffee secondCoffee = (Coffee)beverages[1];
```

Однією з найпоширеніших задач є перебирання елементів колекції. Зазвичай це означає послідовний перегляд кожного елемента колекції, щоб оцінити його за певними критеріями або виконати для нього якусь дію. Для того, щоб перебрати колекцію, можна використовувати цикл foreach. Цикл foreach відображає кожен елемент з колекції по черзі, використовуючи ім'я змінної, що ви задали під час декларації циклу.

Наступний приклад демонструє, як проходити по колекції `ArrayList`:

```
// Проходження по списку
foreach(Coffee coffee in beverages)
{ Console.WriteLine("Bean type: {0}", coffee.Bean);
}
```

7.1.4 Використання узагальнення для колекцій

Одним з найбільш загальних і важливих застосувань узагальнень є створення колекцій. Узагальнення для колекції поділяються на дві широкі категорії: узагальнення списку та узагальнення словників. Узагальнення списку зберігає колекцію об'єктів типу `T`.

Клас `List<T>` надає строго типізовані альтернативи для класу `ArrayList`.

Простір імен `System.Collections.Generic` також включає різні узагальнення колекцій, що надають більш спеціалізовану функціональність:

- клас `LinkedList<T>` надає узагальнення, в якому кожен елемент пов'язаний з попереднім і наступним елементом колекції. Кожен елемент в колекції представлений об'єктом `LinkedListNode<T>`, що містить значення типу `T`, посилання на екземпляр батька `LinkedList<T>`, посилання на попередній елемент в колекції та посилання на наступний елемент в колекції;

- клас `Queue<T>` представляє строго типізовану колекцію типу "перший зайшов - перший вийшов";

- клас `Stack<T>` представляє строго типізовану колекцію типу "останній зайшов - перший пішов".

7.2 Завдання

Реалізуйте інтерфейс `IPrint` для виведення полів класу із завдання лабораторної роботи 6.

Створіть та заповніть колекцію, яка може містити як екземпляри базового класу, так і успадкованих. Напишіть метод `PrintCollection()`, який виведе всі елементи колекції.

7.3 Контрольні запитання

1. Для чого використовуються інтерфейси?
2. Яка відмінність між інтерфейсами та абстрактними класами?
3. Скільки класів можуть мати реалізацію методів інтерфейсу?
4. Скільки інтерфейсів може бути реалізовано в одному класі?
5. Який загальний вигляд опису інтерфейсу?
6. Які елементи мови програмування можна вказувати в інтерфейсах?
7. Як виглядає загальна форма реалізації інтерфейсу в класі?
8. Яка загальна форма класу, що реалізує декілька інтерфейсів?
9. Яким чином в інтерфейсі описується властивість?
10. Які елементи програмування мови C# не можна описувати в інтерфейсах?
11. Що таке явна реалізація члену інтерфейсу?
12. В яких випадках краще використовувати інтерфейс, а в яких абстрактний клас?
13. Яка головна перевага колекцій?
14. Які є види колекцій?
15. Якими типами даних оперують неузагальнені колекції?
16. Якими типами даних оперують спеціальні колекції?
17. В якому просторі імен оголошуються спеціальні колекції?
18. Які стандартні структури даних реалізують узагальнені колекції?
19. Які особливості використання паралельних (багатопотокових) колекцій?
20. В якому просторі імен визначені паралельні (багатопотокові) колекції?

8 ЛАБОРАТОРНА РОБОТА №8

Делегати в C#

Мета роботи: навчитися викликати події та ознайомитися з принципами використання делегатів.

8.1 Теоретичні відомості

8.1.1 Події та делегати

Події (events) – це механізм, що дозволяє об'єктам повідомляти іншим об'єктам, коли щось відбувається. Наприклад, на вебсторінці генерується подія, коли користувач взаємодіє з елементом управління, натискаючи на кнопку. Ви можете створити код, що підписується на ці події та відповідає певними діями на них.

Делегати – це спеціальний тип, що визначає сигнатуру методів, іншими словами – тип даних, що повертається, та параметри методу. Виходячи з назви, делегат поводить себе як представник для методу з відповідною сигнатурою.

8.1.2 Створення подій та делегатів

Коли ви створюєте подію в структурі або класі, необхідно забезпечити механізм, який дає змогу іншим частинам коду підписуватися на цю подію. Отже, делегат – це спеціальний тип, що визначає сигнатуру методу, іншими словами тип і параметри методу. З імені зрозуміло, що делегат себе поводить як представник методу з відповідною сигнатурою.

Коли ви визначили подію, ви робите асоціацію вашого делегата з подією. Для того, щоб підписатися на подію з клієнтської частини коду, вам потрібно:

- створити метод з сигнатурою, що відповідає делегату події. цей метод відомий як обробник події;
- підписатися на подію, додавши ім'я обробника методу до публікації події, іншими словами, об'єкту, що буде формувати подію;
- коли подія створена, делегат запускає всі методи обробки події, що підписані на цю подію.

Наступний приклад демонструє як визначати делегати та події:

```
// Визначення Delegate та Event
public struct Coffee
{
    public EventArgs;
    public delegate void
        OutOfBeansHandler(Coffee coffee, EventArgs args);
    public event OutOfBeansHandler OutOfBeans;
}
```

У цьому прикладі ви визначаєте подію з назвою `OutOfBeans`. Ви асоціюєте її з ім'ям делегата `OutOfBeansHandler`. Делегат `OutOfBeansHandler` приймає два параметри, екземпляр `Coffee`, який є об'єктом, що створює подію, та екземпляр `EventArgs`, що може бути використаний, щоб отримати більше інформації про подію.

8.1.3 Виникнення та підписка подій

Після того, як ви визначили подію і делегат, ви можете писати код, що викликає цю подію за певних обставин. Коли ви викликаєте подію, делегат, що асоціюється з цією подією, буде викликати усі методи обробки події, які були підписані на цю подію.

Для того, щоб виникла подія, вам потрібно зробити наступні кроки:

- перевірити чи подія дорівнює `null` – подія буде дорівнювати `null`, якщо ніякий код на цю подію не підписаний;
- викликати подію та надати аргументи делегату.

Наступний приклад демонструє як викликати подію:

```
// Генерація події
public struct Coffee
{
    // Визначення події та делегата.
    public EventArgs = null;
    public delegate void OutOfBeansHandler(Coffee
coffee, EventArgs args);
    public event OutOfBeansHandler OutOfBeans;
    int currentStockLevel;
    int minimumStockLevel;
```

```

public void MakeCoffee()
{
    // Зменшення рівня запасу.
    currentStockLevel--;
    // Якщо рівень запасу опускається нижче мінімуму,
    // ми викликаємо подію.
    if (currentStockLevel < minimumStockLevel)
    {
        // Перевірка події на null.
        if (OutOfBeans != null)
        { // Генерація події.
            OutOfBeans(this, e);
        } } }
}

```

Для того, щоб викликати подію використовується схожий синтаксис як і для виклику метода. Ви надаєте аргументи, які збігаються з параметрами, що вимагаються делегатом. Перший аргумент – це об’єкт, що викликає подію. Ключове слово `this` використовується для зазначення поточного екземпляра `Coffee`. Другий параметр – це екземпляр `EventArgs`, що може бути `null`, якщо вам не потрібно надавати жодної іншої інформації тим, хто на нього підписаний.

Якщо ви хочете обробляти подію, вам потрібно зробити наступне:

- створити метод з сигнатурою, що відповідає делегату події;
- використати додатковий оператор `(+=)` для додання методу обробки події до вашої події.

Для того, щоб скасувати підписку на подію, використовуйте оператор віднімання `(-=)` для видалення вашого методу обробки події від події.

Наведений нижче приклад демонструє, як скасувати підписку на подію:

```

// Скасувати підписку на подію
public void UnsubscribeFromEvent()
{
    coffeel.OutOfBeans -= HandleOutOfBeans;
}

```

8.2 Завдання

Напишіть програму згідно з варіантом.

Варіант 1. Напишіть програму, в якій потрібно реалізувати ситуацію: при виклику примірника делегата кожен раз генерується чергове число в послідовності Фібоначчі.

Варіант 2. Напишіть програму, в якій оголошується делегат для методів з двома аргументами (символ і текст) і цілочисельним результатом. У головному класі необхідно описати два статичних методи. Один статичний метод повертає кількість входжень символу (перший аргумент) в текстовий рядок (другий аргумент). Другий метод повертає індекс першого входження символу (перший аргумент) в текстовий рядок (другий аргумент), або значення -1, якщо символ в текстовому рядку не зустрічається. У головному методі створити екземпляр делегата і за допомогою цього екземпляра викликати кожен зі статичних методів.

Варіант 3. Напишіть програму, в якій оголошується делегат для методів з символьним аргументом, які не повертають результат. Опишіть клас, в якому повинно бути символьне поле і метод, що дозволяє привласнити значення символьному полю об'єкта. У методу один символьний аргумент і метод не повертає результат. Створіть масив об'єктів даного класу. Створіть екземпляр делегата. У список викликів цього делегата необхідно додати посилання на метод (привласнює значення символьному полю) кожного об'єкта з масиву. Перевірити результат виклику такого екземпляра делегата.

Варіант 4. Напишіть програму, в якій є статичний метод. Першим аргументом статичному методу передається цілочисельний масив. Другим аргументом статичному методу передається посилання на інший метод. У метода - аргументу повинен бути цілочисельний аргумент, який повинен повертати цілочисельний результат. Створіть екземпляр делегата. У список викликів цього делегата необхідно додати посилання на метод (привласнює значення символьному полю) кожного об'єкта з масиву. Результатом статичний метод повертає цілочисельний масив. Елементи цього масиву обчислюються як результат виклику методу-аргументу, якщо йому передавати значення елементів з масиву-аргументу. Запропонуйте механізм перевірки функціональності даного статичного методу.

Варіант 5. Напишіть програму, в якій потрібно реалізувати ситуацію: за необхідності вводу буде викликатися екземпляр делегату, який реалізує введення з клавіатури. В цей спосіб введіть 2 числа та надрукуйте їх суму.

Варіант 6. Напишіть програму, в якій потрібно перехопити ситуацію поділу на нуль – буде викликатися екземпляр делегату, який реалізує реакцію на це. В цей спосіб введіть 2 числа та надрукуйте їх частку.

Варіант 7. Напишіть програму, в якій потрібно перехопити ситуацію існування файлу – буде викликатися примірник делегату, який реалізує друк його вмісту.

8.3 Контрольні запитання

1. Що таке події?
2. Як здійснюється виклик події?
3. Що таке делегати?
4. Що таке делегати з іменованими методами?
5. Що таке делегати з анонімними методами?

Рекомендована література

1. Джеффрі Р. CLR via C#. Програмування на платформі Microsoft .NET Framework 4.5 мовою C# / Р. Джеффрі. - К.: Print2print, 2020. – 896 с.
2. Троелсен Е. Мова програмування C# 5.0 та платформа .NET 4.5 / Е. Троелсен. К.: Діалектика, 2016 – 1312 с.
3. Фрімен А. ASP.NET 4.5 з прикладами C# 5.0 для професіоналів. / А.Фрімен. - К.: Print2print, 2014. - 1120 с.
4. Griffiths I. Programming C# 5.0: Building Windows 8, Web, and Desktop Applications for the .NET 4.5 Framework 1st Edition Вид. O'Reilly Media. 2012. – 882 с.
5. Скі Дж. C# для професіоналів. Тонкощі програмування.// C# in Depth, Jon Skeet, Third Edition: Третє видання, Вид. Manning, 2012 – 528 с.