Лабораторна робота №4. Створення та застосування структур у C#

Мета роботи: Ознайомитися з поняттям структур у C#, їх створенням, використанням, методами і властивостями. Навчитися застосовувати структури та методи для зберігання, обробки даних, а також вирішення прикладних задач.

Теоретичні відомості

Методи в C# — це іменовані блоки коду, які виконують конкретну задачу та можуть викликатися багаторазово у програмі. Вони дозволяють розбивати програму на логічні частини, що спрощує її структуру, підвищує читабельність і спрощує повторне використання коду. Наприклад, замість того, щоб писати однаковий код у різних частинах програми, можна створити метод, який виконує цю задачу, і викликати його тоді, коли це потрібно.

```
void PrintHello()
{
    Console.WriteLine("Привіт, світ!");
}
```

Метод PrintHello виконує одну задачу: виводить повідомлення на екран. Його можна викликати стільки разів, скільки це необхідно, використовуючи його ім’я:

```
PrintHello();
```

Методи оголошуються з визначенням типу повернення, імені методу, параметрів (якщо потрібні) та блоку коду, що виконується. Тип повернення вказує, який результат повертає метод. Якщо метод не повертає значення, використовується ключове слово void. Наприклад:

```
int Add(int a, int b)
{
    return a + b;
}
```

Метод Add приймає два цілі числа як параметри та повертає їх суму.

Передача параметрів у метод може здійснюватися за значенням або за посиланням. Передача за значенням означає, що в метод передається копія значення змінної, тому зміни всередині методу не впливають на оригінальну змінну. Передача за посиланням дозволяє змінювати початковий об’єкт або значення. Для цього використовуються ключові слова ref або out. Наприклад:

```
void DoubleValue(ref int number)
{
    number *= 2;
}

int value = 5;
DoubleValue(ref value);
Console.WriteLine(value);
```

Методи є основою структурного програмування, оскільки дозволяють організовувати код у вигляді модульних блоків. Вони допомагають уникати дублювання коду, оскільки однакову логіку можна інкапсулювати в методі та

викликати його з різних місць програми. Методи також дозволяють реалізувати принцип розділення обов'язків у програмі. Наприклад, одна частина програми відповідає за отримання даних, інша — за їх обробку, а ще одна — за виведення результатів:

```
int[] GetNumbers()
{
    return new int[] { 1, 2, 3, 4, 5 };
}

int CalculateSum(int[] numbers)
{
    int sum = 0;
    foreach (int number in numbers)
    {
        sum += number;
    }
    return sum;
}

void PrintResult(int result)
{
    Console.WriteLine($"Сума чисел: {result}");
}

// Використання методів
int[] numbers = GetNumbers();
int sum = CalculateSum(numbers);
PrintResult(sum);
```

Структури в C# – це типи даних, які дозволяють об'єднувати кілька змінних різних типів у єдиний логічний блок. Структури є типами значення (value types) і використовуються для моделювання об'єктів із фіксованими характеристиками, наприклад, точка на площині або запис про книгу. Структура оголошується за допомогою ключового слова struct. Усередині структури можна визначити змінні (поля) та методи для роботи з ними.

```
struct Point
{
    public int X;
    public int Y;

    public void Display()
    {
        Console.WriteLine($"Точка: ({X}, {Y})");
    }
}
```

Після оголошення структури її можна використовувати, створюючи змінні цього типу. Поля структури можна змінювати, а методи викликати для виконання операцій.

```
Point p;
p.X = 10;
p.Y = 20;

p.Display(); // Виведе: Точка: (10, 20)
```

Конструктор у структурі — це спеціальний метод, який використовується для ініціалізації полів під час створення нового екземпляра структури. Конструктори в структурах мають певні особливості, які важливо враховувати. Кожна структура автоматично має стандартний конструктор, який ініціалізує всі поля значеннями за замовчуванням. Для числових типів це 0, для булевих — false, а для nullable-типів (int?, double?) — null. Наприклад:

```
struct Point
{
    public int X;
    public int Y;
}

Point p; // Поля X і Y отримують значення 0.
p.X = 10; // Ініціалізація поля X.
p.Y = 20; // Ініціалізація поля Y.
```

Однак якщо потрібно задати значення полів під час створення структури, можна використовувати користувацький конструктор. У структурі можна визначити конструктор, який приймає параметри. Конструктор дозволяє одразу ініціалізувати всі поля, замість того щоб присвоювати значення кожному полю окремо. Особливість конструкторів у структурах полягає в тому, що вони повинні ініціалізувати всі поля, інакше виникне помилка компіляції.

```
struct Point
{
    public int X;
    public int Y;

    // Конструктор із параметрами
    public Point(int x, int y)
    {
        X = x;
        Y = y;
    }
}

// Використання конструктора
Point p = new Point(5, 10);
Console.WriteLine($"X: {p.X}, Y: {p.Y}"); // Виведе: X: 5, Y: 10
```

У цьому прикладі конструктор дозволяє створити точку з конкретними координатами, переданими як параметри.

Структури мають кілька важливих обмежень при роботі з конструкторами:

1. Стандартний конструктор: Його неможливо перевизначити. Усі поля, які не ініціалізуються явно, отримують значення за замовчуванням.

2. Обов’язкова ініціалізація: У користувацькому конструкторі потрібно обов’язково присвоїти значення всім полям, інакше компілятор виведе помилку.

3. Відсутність конструктора без параметрів: У структурах заборонено явно створювати конструктор без параметрів, оскільки його автоматично надає компілятор.

Зміст роботи

Завдання 1: Створити приватний репозиторій з назвою oop-lab4. Надати доступ викладачу.

Завдання 2: Створити консольний проект Visual C#. Назва рішення – OOPLab4. Назва проекту – OOPLab4.2

2.1 Приєднати віддалений репозиторій до локального

2.2 Реалізувати структуру згідно варіанту

2.3 Додайте конструктор, який дозволяє ініціалізувати всі обов'язкові поля.

2.4 Додайте метод для виведення інформації про об'єкт

1	**Анкета для студентів** Поля: - Ім'я - string - Прізвище - string - Дата народження - DateTime - Email - string - Стать - string - Група - string
2	**Система управління бібліотекою** Поля: - Назва - string - Автор - string - Рік видання - DateTime - Жанр - string - Видавництво - string - Опис книги – string
3	**Менеджер завдань (To-Do List)** Поля: - Назва завдання - string - Опис завдання - string - Дедлайн - string - Пріоритет (низький/середній/високий) - string - Категорія завдання – string
4	**Журнал спортивних тренувань** Поля: - Тип тренування - string - Дата тренування - DateTime - Тривалість - double - Витрачені калорії - double - Опис тренування – string
5	**Платформа замовлення їжі** Поля: - Назва страви - string

	- Кількість - int - Ціна за одиницю - double - Примітки до замовлення - string - Контактний телефон замовника – string
6	**Електронний розклад** Поля: - Назва події - string - Дата - DateTime - Час - TimeSpan - Локація - string - Опис події - string
7	**Система бронювання місць у кінотеатрі** Поля: - Назва фільму - string - Час сеансу - TimeSpan - Номер місця - int - Формат фільму (2D/3D/IMAX) - string - Додаткові побажання - string
8	**Книга контактів** Поля: - Ім’я - string - Номер телефону - string - Email - string - Адреса - string - Примітки – string
9	**Система управління подіями** Поля: - Назва події - string - Дата - DateTime - Локація - string - Опис події - string - Час події – TimeSpan
10	**Каталог автомобілів** Поля: - Марка - string - Модель - string - Рік випуску - int - Тип двигуна - string - Пробіг – int
11	**Місця для подорожей** Поля: - Назва місця - string - Країна - string - Опис - string - Рейтинг - int

	- Дата відвідування
12	**Домашній інвентар** Поля: - Назва предмета - string - Категорія - string - Місце зберігання - string - Примітки - string
13	**Бюджетна система** Поля: - Категорія (доходи/витрати) - string - Сума - double - Дата - DateTime - Опис транзакції - string - Спосіб оплати (готівка/карта) – string
14	**Система обліку замовлень** Поля: - Назва товару - string - Кількість - int - Ціна - double - Примітки до замовлення – string
15	**Платформа обліку медичних консультацій** Поля: - Ім’я пацієнта - string - Лікар - string - Дата консультації - DateTime - Опис стану пацієнта - string - Рекомендації лікаря - string
16	**Облік користувачів** Поля: - Логін - string - Пароль - string - Email - string - Дата реєстрації - DateTime - Роль користувача - string
17	**Система відстеження відправлень** Поля: - Номер відправлення - int - Відправник - string - Одержувач - string - Статус - string - Дата відправлення - DateTime - Дата прибуття - DateTime
18	**Облік відвідувань музею** Поля: - Номер квитка - int

	- Ім’я відвідувача - string - Дата відвідування - DateTime - Тип квитка - string - Вартість - double
19	**Система оренди велосипедів** Поля: - Код велосипеда - string - Тип велосипеда - string - Стан - string - Дата оренди - DateTime - Дата повернення - DateTime - Вартість - double
20	**Каталог навчальних матеріалів** Поля: - Назва - string - Автор - string - Тип - string - Тривалість або кількість сторінок - int - Посилання на ресурс - string - Дата публікації - DateTime
21	**Список учасників кіноклубу** Поля: - Ім’я - string - Прізвище - string - Улюблений жанр - string - Кількість відвіданих сеансів - int - Дата останнього відвідування - DateTime
22	**Система управління кредитами:** Поля: - Ім’я - string - Прізвище - string - Сума кредиту - double - Відсоткова ставка - double - Термін - int - Дата оформлення - DateTime - Статус - string
23	**Каталог вин** Поля: - Назва - string - Виробник - string - Рік урожаю - DateTime - Країна походження - string - Вміст алкоголю - double
24	**Моніторинг якості повітря** Поля:

	- Локація - string - Дата вимірювання - DateTime - Рівень CO2 - double - Температура - double - Вологість - double
25	**Журнал кіберспортивних турнірів** Поля: - Назва - string - Гра - string - Дата проведення - DateTime - Призовий фонд - double - Команда-переможець - string
26	**Система керування таксі** Поля: - Номер замовлення - string - ПІБ пасажира - string - Місце подачі - string - Пункт призначення - string - Дата замовлення - DateTime - Вартість - double
27	**Облік використання електроенергії** Поля: - Локація - string - Дата зняття показників - DateTime - Спожита електроенергія (кВт-год) - double - Вартість - double - Тариф - string
28	**Управління системою метеоспостережень** Поля: - Локація - string - Дата вимірювання - DateTime - Температура - diuble - Атмосферний тиск - double - Вітер - double - Опади - bool
29	**Система планування рекламних кампаній** Поля: - Назва кампанії - string - Бюджет - doble - Тип реклами - string - Дата початку - DateTime - Дата завершення - DateTime
30	**Система моніторингу комп’ютерної техніки** Поля: - Інвентарний номер - string

	- Марка - string - Модель - string - Дата купівлі - DateTime - Поточний стан - string

Завдання 3: Реалізувати список структур згідно обраної теми. Додати можливість виведення всіх елементів списку. Забезпечити зручне відображення даних у консоль (чітке форматування таблиці записів).

Завдання 4: Реалізувати можливість додавання елементу в список, видалення елементу з списку.

Завдання 5: Реалізувати сортування списку структур (два поля для сортування на вибір).

Завдання 6: Реалізувати метод для пошуку об’єкта в списку (два поля для пошуку на вибір).

Завдання 7: Реалізувати меню для роботи з програмою. При запуску програми користувачеві пропонується меню для виконання наступних операцій:

1. Вивести на екран записи
2. Додати запис
3. Видалити запис (за індексом)
4. Сортування

* при виборі цього пункту відображати додаткове меню для вибору параметра сортування

5. Пошук

* при виборі цього пункту відображати додаткове меню для вибору параметра пошуку

6. Завершити роботу

Завдання 8: Створити коміт та завантажити зміни в віддалений репозиторій.

Контрольні запитання

1. Що таке структура в C# і для чого вона використовується?
2. Які дані можуть зберігатися в структурі?
3. Як оголошується структура у C#? Наведіть приклад.
4. Чи можна зробити поле в структурі необов’язковим (nullable)? Як це зробити?
5. Чи можуть структури мати конструктори? Які обмеження накладаються на їх використання?

6. Чим відрізняється стандартний конструктор структури від користувацького?
7. Які методи можуть бути визначені в структурі?
8. Як передаються структури в методи: за значенням чи за посиланням? Як це впливає на продуктивність?
9. Як викликати метод структури для обробки її даних?
10. Що станеться, якщо не ініціалізувати поле в структурі?