Тема 2. Класи та їх складові. Конструктори класів

1. **Загальний синтаксис опису класів. Синтаксис опису складових класів.**
2. **Призначення та синтаксис опису методів.**
3. **Локальні змінні та константи.**
4. **Формальні та фактичні параметри методів.**
5. **Конструктори та деструктори.**

Рекомендована література:

Базова

1. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд.: Пер. с англ. / Г. Буч, Р. А. Максимчук, М. У. Энг и др. – М.: Вильямс, 2008. – 720 с.
2. Брила А. Ю. Основи об’єктно-орієнтованого програмування у C#. Методичні вказівки до лабораторних робіт для студентів І-го курсу математичного факультету спеціальності "Прикладна математика" / [А. Ю. Брила, П. П. Антосяк, М. І. Глебена та ін.]. – Ужгород, 2014. – 73 с.
3. Васильев А. C#. Объектно-ориентированное программирование. Учебный курс / Алексей Васильев. – СПб.: Питер, 2012. – 320 с.
4. Стиллмен Э. Изучаем C#. 3-е изд / Эндрю Стиллмен, Дженнифер Грин. – СПб.: Питер, 2014. – 816 с.
5. Троелсен Э. Язык программирования C# 6.0 и платформа .NET 4.6. 7-е изд. / Эндрю Троелсен, Филипп Джепикс. – СПб.: Диалектика-Вильямс, 2018. – 1440 с.
6. Албахари Д. C# 6.0. Справочник. Полное описание языка: Пер. с англ. / Джозеф Албахари, Бен Албахари. – М.: Вильямс, 2018. – 1040 с.
7. C#. Спецификация языка. Версия 5.0 / Microsoft Corporation, 2012. – 577 с.
8. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке C#. Мастер-класс: Пер. с англ. / Дж. Рихтер. – СПб.: Питер, 2007. – 656 с.

Допоміжна

9. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++, 2-е изд.: Пер. с англ. / Г. Буч. – М.: "Издательство Бином" 1998. – 560 с.
10. Кнут Д. Е. Искусство программирования для ЭВМ. Т. 3: Сортировка и поиск / Д. Е. Кнут. – 2-е изд. – М.: Вильямс, 2008. – 824 с.
11. Жарков В. А. Самоучитель Жаркова по Visual Studio .Net: Visul Basic .NET, Visual C# .NET, Visual C++ .NET, Visual J# .NET / В. А. Жарков. – М.: Жарков Пресс, 2002. – 592 с.
12. Климов Л. П. C#. Советы программистам / Л. П. Климов. – СПб.: БХВ-Петербург, 2008. – 544 с.
13. Задачи по программированию / С. А. Абрамов,, Г. Г. Гнездилова, Е. Н. Капустина и др.. – М.: Наука, 2000. – 596 с.

1. В C# класи використовуються в двох аспектах: клас як модуль і клас як тип даних.

При використанні **класу як модуля** C# дозволяє звертатися до поля чи методу класу без створення екземпляра. Для цього сам клас, його поля і методи мають бути статичними і описуватися з модифікатором **static**.

Класи-типи даних належать до посилкових типів і тому для кожного об'єкта такого класу необхідно виділяти місце під його дані з допомогою оператора **new**.

Загальний синтаксис опису класів:

[<атрибут класу>] [<модифікатор (рівень) доступу>] class **<назва класу>**
[:<назва батьківського класу>]
{<дані>
<методи>} .

Якщо батьківський клас не вказано, то наслідування автоматично відбувається від базового класу *Object*. Перед класом можуть використовуватися наступні **атрибути класу**:

- **partial** – розділюваний клас, що може описуватися в декількох місцях, при чому в одному місці може зазначатися заголовок методу, а в іншому – його реалізація;
- **abstract** – клас є базовим і використовується для породження інших класів. Створити екземпляр такого класу неможливо;
- **sealed** – заборона наслідування від даного класу.
- **unsafe** – дозволяє небезпечний код з використанням вказівників.

Крім цих атрибутів можуть застосовуватися такі **модифікатори доступу**:

- **public** – клас має необмежений доступ і може використовуватися у всьому проекті.
- **internal** – доступ обмежений поточною збіркою (по замовчуванню).

Складовими (**членами**) **класу можуть бути поля, властивості, методи, події та ін.** (наприклад, **константи, індексатори, вкладені типи**). **Для членів** класу визначені такі **модифікатори доступу**:

- **public** – член має необмежений доступ і може використовуватися у всьому рішенні;
- **private** – доступ обмежений лише своїм класом (по замовчуванню);
- **internal** – доступ обмежений поточною збіркою;
- **protected** – доступ обмежений активним класом і породженими класами.

2. Методи використання для виконання повторюваних операцій та для опису дій об’єктів. Типовий приклад методів `WriteLine`, який виводить дані користувача у вікно консольного додатку. Кожен метод має своє унікальне ім’я в сукупності з аргументами. Виклик методу виконується вказуванням його назви, після чого зазначаються круглі дужки, де перераховуються аргументи.

В C# використовуються статистичні методи та методи класу. Статистичні методи розміщуються в пам’яті один раз як і їхні змінні, перед назвою статистичного методу ставиться слово `static`. Методи класу створюються вказаного класу і їх змінні створюються для кожного екземпляра.

Синтексис опису статистичних методів:

```
static <тип результату> <назва методу>(<список аргументів>);
```

Для виходу з методу використовується оператор `return`. Якщо метод не повертає жодне значення в місце виклику, то замість типу його результату вказується слово `void`. Тип результату після `return` обов’язково має співпадати з типом результату підпрограми.

3. Всередині кожної підпрограми можуть використовуватися свої змінні і константи, які називають локальними, вони створюються під час виклику методу і зникають після

завершення його роботи.кожній локальній змінній може відразу присвоюватися початкове значення. Наприклад: `int i = 0;` . C# забороняє використовувати змінні, доки їм не присвоєно значення.

4. В кожен метод можуть передаватися аргументи, які між собою відмежовані комами. Ім'я аргумента у списку аргументів і при виклику методу можуть не співпадати, але обов'язково має співпадати тип даних. Передача аргумента можлива по значенню і по змінній. Крім цього в C# ще й використовують аргументи результату.

При передачі по значенню створюються формальні параметри фактично формується нова змінна, в яку з основної програми заноситься початкове значення і надалі змінна програми і змінна підпрограми між собою незалежні.

При передачі за адресою дії відбуваються над змінними основної програми. Перед такою змінною в списку аргументів вказується службове слово ref, при цьому використовується фактичний параметр.

Для формування параметра виводу в списку аргументів і при виклику перед ним вказується слово out. В такий аргумент не заноситься початкове значення, а лише повертається результат.

Приклад:

Скласти підпрограму для коректного введення дійсного числа в діалоговому режимі.

Якщо не використовувати підпрограми, то фрагмент для коректного введення необхідно буде повторити при кожному введенні і використовувати різні мітки. Такі повтори необхідно буде використовувати і в інших програмах для введення дійсних чисел. Тому створимо власну підпрограму для введення дійсного числа і будемо її викликати потрібну кількість разів. Підпрограма може ввести і не ввести дані від користувача, тому буде повертати логічний результат.

```
static bool InputDouble(ref Double x, String Povidom)
 {String S;
  S = x.ToString();
 Povtor:
   S = Interaction.InputBox(Povidom, "Введення", S);
   try
   {x = Convert.ToDouble(S);
   }
   catch (System.FormatException)
   {if (MessageBox.Show("Ви ввели не число" + Strings.Chr(13) +
         Strings.Chr(13) + "Бажаєте повторити?", "Увага",
         MessageBoxButtons.YesNo, MessageBoxIcon.Warning) ==
        DialogResult.Yes)
     goto Povtor;
    else
      return false;
   }
  return true; }
```

5. Приклад опису класу. Створити клас *Паралелограм* з доступними процедурами визначення площі, периметра та друку його даних.

```
class Parallelogram
 {private double a, b, alfa;

  public double area()
   {return a * b * Math.Sin(alfa / 180 * Math.PI);
   }

  public double perimeter()
```

```
        {return 2 * (a + b);
        }

       public void Info()
        {MessageBox.Show("Дані паралелограма:\nдві сторони по "+a.ToString()+
            " та дві по "+b.ToString()+" од.;\nплоща: " + area().ToString() +
            " кв. од.;\nпериметр: " + perimeter().ToString() +
            " од.;\nдва кути по " + alfa.ToString() + " і два кути по " +
            (180 - alfa).ToString() + " градусів","Інформація",
            MessageBoxButtons.OK,MessageBoxIcon.Information);
        }
      }
```

В класах для задання початкових значень використовують спеціальні методи – **конструктори**. Вони не повертають жодного значення і мають назву, яка співпадає з назвою класу. Клас може мати декілька конструкторів з різною кількістю чи типами даних параметрів. Як правило, конструктори описують після полів класу, але перед його методами. Кожен клас може мати **конструктор без параметрів**, **конструктори з параметрами** та **конструктори копіювання** [2, с. 6-8]. Крім цього, у класі може бути **статичний конструктор** без параметрів, який виконується один раз перед першим використанням класу, та **закритий (private) конструктор** для недопущення створення об’єкта класу.

Приклад. Створити конструктор класу *Паралелограм*, в який будуть передаватися дві сторони і кут між ними та буде виводитися інформація про паралелограм.

```
public Parallelogram(double a, double b, double alfa)
 {this.a = s; this.b = b; this.alfa = alfa;
  Info();
 }
```

Після цього, якщо в програмі записати

```
Parallelogram p1 = new Parallelogram(5, 7, 60);   ,
```

то буде створений об'єкт-паралелограм.

Питання для самоконтролю

1. Синтаксис опису класів в C#
2. Поля та методи класів.
3. Області видимості в класах.
4. Підпрограми та їх використання в ООП.
5. Різновиди підпрограм.
6. Формальні та фактичні параметри підпрограм.
7. Локальні та глобальні змінні додатків.
8. Рекурсії та їх використання.