Тема 8. Методи класів

Мета лекції – отримати теоретичні та практичні навики розробки програм з класами, що містять найпростіші методи.

Дана лабораторна робота сприяє напрацюванню наступних **компетентностей** у відповідності до Національної рамки кваліфікації:

Знання:

формат запису методів;

способи повернення з методів;

правила передачі параметрів – значень.

Уміння:

для заданої наочної області визначити клас, який інкапсулює елементи-методи та елементи-дані, що потребують подальшої обробки;

написати програму, яка обробляє інформацію (обчислення і вивід на екран результатів обчислень і вмісту елементів-даних) про декілька об'єктів одного класу заданої наочної області.

Комунікації:

обґрунтування рекомендацій команді учасників проекту щодо доцільності застосування методів згідно певного сценарію взаємодії відповідних об’єктів;

робота в команді над окремими класами, які містить перелік методів.

Автономність і відповідальність:

прийняття рішення щодо доцільності включення в структуру класу певного методу;

самостійне обґрунтування можливих варіантів реалізацій відповідних методів.

Основні положення

Мова C# є об'єктно-орієнтованою і тому всі програми на її основі містять класи. Класи необхідні для опису об’єктів, на базі яких створюються їх певні екземплярів. Реалізація сценарію (алгоритму) здійснюється як взаємодія об’єктів. Взаємодія відбувається за допомогою виклику методів і подальшому обміну відповідною інформацією між ними.

Методи мають багато спільного з функціями, основні принципи роботи з якими були розглянуті при вивченні дисципліни "Основи програмування". Тому перед виконанням даної лабораторної роботи слід повторити матеріал пов'язаний з практичним застосуванням функцій при процедурному стилі програмування.

Якщо функції – це засіб, що дозволяє виконувати деякі ділянки коду в довільному місці додатка, то метод - це функціональний елемент класу, який реалізує обчислення або інші дії, що виконуються класом або його екземпляром (об'єктом).

Методи класу, як правило, маніпулюють даними, визначеними в класі, і забезпечують доступ до цих даних.

Метод Main () в попередній лабораторній роботі (приклад 1) обчислював площу, що припадає на одну людину, шляхом ділення загальної площі будівлі на кількість мешканців.

Незважаючи на формальну коректність, ці обчислення виконані не найвдалішим чином. Адже з обчисленням площі, що припадає на одну людину, цілком може впоратися сам клас Building, оскільки ця величина залежить тільки від значень змінних area і occupants, які інкапсульовані в класі Building.

Якщо обчислення площі буде «закріплено» за цим класом, то в іншої програмі, яка його використовує, не доведеться виконувати цю дію «вручну». Тут у наявності не просто зручність для «інших» програм, а запобігання невиправданого дублювання коду.

Нарешті, вносячи в клас Building метод, який обчислює площу, що припадає на одну людину, ви покращуєте його об'єктно-орієнтовану структуру, інкапсуліруя всередині розглянутого класу величини, пов'язані безпосередньо з будівлею.

Щоб додати в клас Building метод, необхідно визначити його всередині оголошення класу.

Приклад 1. Додавання методу в клас Building.

Наступна версія класу Building (в порівнянні с прикладом 1 попередньої лабораторної роботи №1) містить метод з ім'ям areaPerPerson (), який відображає значення площі конкретної будівлі, що припадає на одну людину.

```
using System;
class Building
{
```

```
        public int floors;          // кількість поверхів
        public int area;            // загальна площа будівлі
        public int occupants;       // кількість мешканців
// Відображаємо значення площі, що припадає на одну людину.
        public void areaPerPerson ()
        {
        Console.WriteLine ("" + area / occupants +
        "  Припадає на одну людину");
        }
}
// Використовуємо метод areaPerPerson ().
class BuildingDemo
{
        public static void Main ()
        {
        Building house = new Building ();
        Building office = new Building ();
        // Привласнюємо значення полям у об'єкті house
        house.occupants = 4;
        house.area = 2500;
        house.floors = 2;
        // Привласнюємо значення полям у об'єкті office
        office.occupants = 25;
        office.area = 4200;
        office.floors = 3;
        // Вивід результатів
        Console.WriteLine ("Будинок має: \n" +
        house.floors + " поверху \n" +
        house.occupants + " мешканця \n" +
        house.area + " кв.м. загальної площі, з них");
        house.areaPerPerson ();
        Console.WriteLine ("Офіс має: \n" +
        office.floors + " поверху \n" +
        office.occupants + " працівників \n" +
        office.area + " кв.м. загальної площі, з них");
        office.areaPerPerson ();
        }
```

```
}
```

Результат виконання програми.

```
Будинок має:
2 поверху
4 мешканця
2500 кв.м. загальної площі, з них
625  Припадає на одну людину
Офіс має:
3 поверху
25 працівників
4200 кв.м. загальної площі, з них
168  Припадає на одну людину
```

Розглянемо ключові елементи цієї програми, починаючи з самого методу areaPerPerson ().

Перший рядок цього методу виглядає так:

```
public void areaPerPerson ()
```

У цьому рядку оголошується метод з ім'ям areaPerPerson (), який не має параметрів. Цей метод визначений з використанням специфікатора доступу public, тому його можуть використовувати всі інші частини програми. Метод areaPerPerson () повертає значення типу void, тобто не повертає ніякого значення.

Тіло методу areaPerPerson () складається з єдиної інструкції:

```
{
Console.WriteLine ("" + area / occupants +
"  Припадає на одну людину");
}
```

Ця інструкція відображає площа будівлі, яка припадає на одну людину, шляхом ділення значення змінної area на значення змінної occupants.

Оскільки кожен об'єкт типу Building має власну копію значень area і occupants, то при виклику методу areaPerPerson () використовуватимуться копії цих змінних.

Метод areaPerPerson () завершується фігурної дужкою, тобто управління програмою передається конкретному об'єкту, який цей метод визвав.

Розглянемо рядок коду з методу Main ():

```
house.areaPerPerson ();
```

Ця інструкція викликає метод areaPerPerson () для об'єкта house. Для цього використовується ім'я об'єкта, за яким слід оператор "крапка". При виклику методу управління виконанням програми передається тілу методу, а після його завершення управління повертається автору виклику, і виконання програми поновлюється з рядка коду, яка розташована відразу за викликом методу.

В даному випадку в результаті виклику house.areaPerPerson () відображається значення площі, яка припадає на одну людину, для будівлі, визначеного об'єктом house.

Точно так же в результаті виклику office.areaPerPerson () відображається значення площі, яка припадає на одну людину, для будівлі, визначеного об'єктом office.

Змінні екземпляра area і occupants використовуються всередині методу areaPerPerson () без яких би то не було атрибутів, тобто їм не передує ні ім'я об'єкта, ні оператор "крапка". Це обумовлено тим, що метод обробляє змінну екземпляра, яка визначена в його класі, він робить це безпосередньо, без явної посилання на об'єкт і без оператора «крапка». Адже метод завжди викликається для деякого об'єкта конкретного класу. І якщо вже виклик відбувся, об'єкт, стало бути, відомий. Таким чином, немає необхідності вказувати всередині методу об'єкт вдруге. Це значить, що значення area і occupants всередині методу areaPerPerson () неявно вказують на копії цих змінних, які належать об'єкту, який викликає метод areaPerPerson ().

Повернення з методу.

В загальному випадку існує два варіанти умов для повернення з методу.

Перший варіант пов'язано з виявленням фігурної дужки, що позначає кінець тіла методу (як продемонстровано на прикладі методу areaPerPerson ()).

Другий варіант полягає у виконанні інструкції return. Можливі дві форми використання інструкції return: одна призначена для void-методів (які не повертають значень), а інша - для повернення значень.

Негайне завершення void-методу можна організувати за допомогою наступної форми інструкції return:

return;

При виконанні цієї інструкції управління програмою передається автору виклику методу, а залишився код опускається.

Приклад 2. Припинення виконання void-методів.

```
using System;
class myReturn
{
   static public void myMeth()
   {
      int i;
      for (i = 0; i < 10; i++)
      {
         if (i == 5) return; // припинення виконання методу при i = 5
         Console.Write(" " + i);
      }
   }
}
   class Program
   {
         static void Main(string[] args)
         {
            myReturn.myMeth();
            Console.WriteLine(); // для паузи

         }
   }
```

Результат виконання програми.

```
 0 1 2 3 4
```

Тут цикл for працюватиме при значеннях змінної «i» в діапазоні тільки від 0 до 5, оскільки, як тільки значення «i» стане рівним 5, буде виконаний повернення з методу myMeth ().

Метод може мати декілька інструкцій return.

Хоча void-методи - не рідкість, більшість методів все ж повертають значення.

Значення, що повертаються методами, використовуються в програмуванні по різному. В одних випадках повертається значення, яке є результатом обчислень, в інших - воно просто означає, успішно чи ні виконані дії, що становлять метод, а в третіх - воно може являти собою код-стану. Однак незалежно від мети застосування, використання значень, що повертаються методами, є невід'ємною частиною C# - програмування. Вони використовують наступну форму інструкції return:

return значення;

Тут елемент «значення» представляє значення, що повертається методом.

Здатність методів повертати значення можна використовувати для поліпшення реалізації методу areaPerPerson (). Замість того щоб відображати значення площі, яка припадає на одну людину, метод areaPerPerson () буде тепер повертати це значення, яке можна використовувати в інших обчисленнях.

В наступному прикладі представлений модифікований варіант методу areaPerPerson (), який повертає значення площі, що припадає на одну людину, а не відображає його (як в попередньому варіанті).

Приклад 3. Повернення значення методом areaPerPerson ().

```
using System;
class Building
{
        public int floors;          // кількість поверхів
        public int area;            // загальна площа будівлі
        public int occupants;       // кількість мешканців
        // Повернення значення площі, яка припадає на одну людину
        public int areaPerPerson ()
        {
                return area / occupants;
```

```
        }
    }
    // Використання значення від методу areaPerPerson ().
    class BuildingDemo
    {
        public static void Main ()
        {
        Building house = new Building ();
        Building office = new Building ();
        int areaPP; // площа, яка припадає на одну людину
        // Привласнюємо значення полям у об'єкті house
        house.occupants = 4;
        house.area = 2500;
        house.floors = 2;
        // Привласнюємо значення полям у об'єкті office
        office.occupants = 25;
        office.area = 4200;
        office.floors = 3;
        // Отримуємо для об'єкта house площу, яка припадає на одну
людину
        areaPP = house.areaPerPerson ();
        Console.WriteLine ("Будинок має: \n" +
        house.floors + " поверху \n" +
        house.occupants + " мешканця \n" +
        house.area + " кв.м. загальної площі, з них \n" +
    areaPP + " припадає на одну людину");
        Console.WriteLine ();
        // Отримуємо для об'єкта office площу, яка припадає на одну
    людину
        areaPP = office.areaPerPerson ();
        Console.WriteLine ("Офіс має: \n" +
        office.floors + " поверху \n" +
        office.occupants + " працівників \n" +
        office.area + " кв.м. загальної площі, з них \n" +
        areaPP + " припадає на одну людину");
        }
    }
```

Результат виконання програми співпадає с попереднім результатом (див. приклад 1).

Використання параметрів.

При виклику методу можна передати одне або кілька значень, які називаються аргументами.

Змінна всередині методу, яка приймає значення аргументу, називається параметром.

Параметри оголошуються всередині круглих дужок, які слідують за ім'ям методу. Синтаксис оголошення параметрів аналогічний синтаксису, який застосовується для змінних.

Параметр знаходиться в області видимості свого методу, і, крім спеціального завдання отримання аргументу, діє подібно будь-якої локальної змінної.

Для додавання в клас Building нового засобу обчислення максимально допустимої кількості мешканців будівлі можна використовувати метод з параметрами. При цьому передбачається, що площа, яка припадає на кожну людину, не повинна бути менше певного мінімального значення. Назвемо цей новий метод maxOccupant () і наведемо його визначення.

```
public int maxOccupant (int minArea)
{
        return area / minArea;
}
```

При виклику методу maxOccupant () параметр minArea отримує значення мінімальної площі, необхідної для життєдіяльності кожної людини. Результат, що повертається методом maxOccupant (), виходить як частка від ділення загальної площі будівлі на це значення.

Приклад 4. Повне визначення класу Building, що включає метод maxOccupant ().

```
using System;
class Building
{
        public int floors;          // кількість поверхів
        public int area;            // загальна площа будівлі
        public int occupants;       // кількість мешканців

        public int areaPerPerson()
```

```
        {
                return area / occupants;
        }

        public int maxOccupant (int minArea)
        {
                return area / minArea;
        }
}
// Використання методу maxOccupant ().
class BuildingDemo
{
        public static void Main ()
        {
        Building house = new Building ();
        Building office = new Building ();
        // Привласнюємо значення полям у об'єкті house
        house.occupants = 4;
        house.area = 2500;
        house.floors = 2;
        // Привласнюємо значення полям у об'єкті office
        office.occupants = 25;
        office.area = 4200;
        office.floors = 3;
        // Отримуємо для об'єкта house площу, яка припадає на одну
людину
        int areaPP = house.areaPerPerson();
        Console.WriteLine("Будинок має: \n" +
        house.floors + " поверху \n" +
        house.occupants + " мешканця \n" +
        house.area + " кв.м. загальної площі, з них \n" +
        areaPP + " припадає на одну людину");
        // Отримуємо для об'єкта office площу, яка припадає на одну
людину
        areaPP = office.areaPerPerson();
        Console.WriteLine("Офіс має: \n" +
        office.floors + " поверху \n" +
        office.occupants + " працівників \n" +
        office.area + " кв.м. загальної площі, з них \n" +
        areaPP + " припадає на одну людину");
        Console.WriteLine ("Максимальне число осіб для дому, \n" +
"якщо на кожного повинно доводиться " +
300 + " квадратних метрів:" +
house.maxOccupant (300));
        Console.WriteLine ("Максимальне число осіб для офісу, \n" +
```

```
"якщо на кожного повинно доводиться " +
200 + " квадратних метрів:" +
office.maxOccupant (200));
        }
}
```

Результат виконання програми.

Будинок має:

2 поверху

4 мешканця

2500 кв.м. загальної площі, з них

625 припадає на одну людину

Офіс має:

3 поверху

25 працівників

4200 кв.м. загальної площі, з них

168 припадає на одну людину

Максимальне число осіб для дому,

якщо на кожного повинно доводиться 300 квадратних метрів: 8

Максимальне число осіб для офісу,

якщо на кожного повинно доводиться 200 квадратних метрів: 21

Завдання.

1. Набрати, відкомпілювати і запустити на виконання прикладі програм, які були наведені в розділі «Основні положення» даної лабораторної роботи.

2. Проекспериментуйте з програмами:

змініть вихідні дані;

досліджуйте, як впливають синтаксичні помилки на результат компіляції програми. Які при цьому виникають помилки компіляції?

Індивідуальна частина.

1. З таблиці 1 (вона приведена в методичних вказівках до лабораторної роботи №1) вибрати індивідуальний варіант предметної області.

2. Розробити відповідно предметної області клас, який інкапсулює елементи-дані і елементи-методи (без параметрів). Написати і налагодити програму, що демонструє роботу з об'єктом (або об'єктами), визначеного вище класу.

3. Визначити додатковий метод (з параметрами) і включити його до раніш розробленого класу. Написати і налагодити програму, що демонструє роботу з об'єктом (або об'єктами), визначеного вище класу.

Контрольні питання

1. Чим відрізняються методи від функцій?
2. Що таке сигнатура методу? Наведіть приклад.
3. Які типи даних може повертати метод?
4. Які варіанти повернення з методу Ви знаєте? У чому їх відмінності і коли їх доцільно застосовувати?
5. Опишіть алгоритм виклику та виконання методу.