Практична робота 15. Лямбда-вирази

Мета. Ознайомитись з синтаксисом та оголошенням лямбда-виразів. Навчитись використовувати їх для типових задач в застосуваннях різних типів.

План

15.1. Лямбда-вирази та лямда-оператор

15.2. Оголошення лямбда-виразів

15.3. Алгоритм застосування лямбда-виразів

15.4. Приклади використання лямбда-виразів в найпростіших програмах. Масиви, структури, класами як парамметри лямбда-виразів.

Зміст заняття

15.1. Лямбда-вирази та лямда-оператор

В C# для створення анонімних функцій використовуються анонімні методи або лямбда-вирази. Анонімні методи розглядались раніше. Лямбда-вирази використовуються з тією ж метою, що й анонімні методи, і є їм альтенативою**.**

Лямбда-вирази описують функції без використання імені за допомогою оператора, який позначається '=>' і називається ***лямбда-оператором***. Його зміст слід розміти як **"*значення передається в*".**

У лівій частині лямбда-оператора вказується один або декілька вхідних параметрів. У правій частині вказується вираз або блок операторів.

Лямди-вирази спрощують програмний код, особливо, якщо в програмі використовуються методи з різним призначенням але однаковою сигнатурою.

15.2. Оголошення лямбда-виразів

Формат оголошення блокового лямбда-виразу з одним вхідним параметром:

```
параметр =>
            {
                // оператори
            }
```

Формат оголошення блочного лямбда-виразу з декількома вхідними параметрами:

```
(список_параметрів) => {
                            // оператори
                        }
```

параметр, список_параметрів – вхідний, вхідні параметри виразу;

оператори – оператори, які виконють дії в програмі.

Приклад 1. Лямбда-вираз з вхідним цілим параметром x, який обчислює кількість цифр '5' у числі x.

Для числа 5854 лямбда-вираз має повернути, наприклад, результат 2.

```
x =>{
            int number = Math.Abs(x);
            int count = 0; // к-сть цифр '5'
            int digit;
            while (number > 0)
            {
                digit = number % 10;

                if (digit == 5)
                    count++;

                number /= 10; // поділити на 10
            }
            return count;
        };
```

Якщо блок операторів містить лише один вираз або виконує одну дію, то фігрні дужки і оператор опускаються і лямбда-вираз визначаються так:

параметр => вираз; - з одним вхідним параметром;

(список_параметрів) => вираз; - з декількома вхідними параметрами:

Приклад 2. Функцію `CalculateCos` обчислення значення функції y=cos(x+2) в програмі

```
double CalculateCos(double x)
 {
     return Math.Cos(x + 2);
 }
```

можна замінити лямда-виразом

```
x => Math.Cos(x + 2);
```

Приклад 3. Лямбда-вираз з 3 вхідними параметрами a, b, c. Вираз використовується для визначення, чи можна з довжин сторін a, b, c утворити трикутник?

```
(a, b, c) => ((a + b) > c) && ((a + c) > b) && ((b + c) > a);
```

У вищенаведеному лямбда-виразі використовується правило: сума двох будь-яких двох сторін трикутника більша за третю сторону.

Якщо в прикладі 2 замінити виклик функції `CalculateCos` запропонованим лямбда-виразом, то отримаємо помилку, тому що лябда-вираз – це лише “безіменний код”, посилання на який призначається делегату. Це код не може бти викликаний безпосередньо з іншої частини програми без нього.

15.3. Алгоритм застосування лямбда-виразів

Щоб у програмному коді застосувати лямбда-вираз, потрібно виконати таку послідовність дій:

Оголосити делегат, результат якого сумісний з типом результату лямбда-виразу.

Оголосити змінну цього типу делегату (екземпляр делегату).

Присвоїти змінній (екземпляру делегату) лямбда-вираз.

Викликати змінну (екземпляр делегату) у програмному коді.

Напишемо за цим алгоритмом програму для прикладу 1.

Приклад 4. Використати лямбда-вираз для знаходження кількості цифр у десятковому записі цілого числа x.

1. *Оголошення типу делегат*

```
delegate int GetDigitsCout(int x);
```

Ім'я типу делегату `GetDigitsCout`**.** Екземпляр делегату цього типу має отримувати один вхідний параметр (x) та повертати значення типу int.

2. *Оголошення змінну цього типу делегату***.** Змінна оголошується в програмному коді, яким може бути код довільного методу класу, код обробника події тощо.

```
GetDigitsCout getDigitFiveCout;
```

3. *Присвоїти змінній лямбда-вираз.*

У коді змінній-делегату присвоюється посилання на код лябда-виразу:

```
getDigitFiveCout = x =>{
                int number = Math.Abs(x);
                int count = 0; // к-сть цифр '5'
                int digit;
                while (number > 0)
                {
                    digit = number % 10;

                    if (digit == 5)
                        count++;

                    number /= 10; // поділити на 10
                }
                return count;
            };
```

Змінна може бути ініціалізована лямбда-виразом і під час її оголошення:

```
GetDigitsCout getDigitFiveCout= x =>
                                    {
                ...
                                    };
```

4. *Викликати змінну***.** У тому ж методі де оголошено лямбда-вираз реалізовано виклик змінної–делегату `getDigitFiveCout`:

```
//оголошення лямбда-виразу
GetDigitsCout getDigitFiveCout= x =>
                                    {
                ...
                                    };
//використання лямбда-виразу
            int a = 615;
            int res = getDigitFiveCout(a); // res = 1
```

Код програми

```
public class Program
     {
          //Оголошення  типу делегат
          delegate int GetDigitsCout(int x);
          static void Main()
          {
               //Оголошення змінної цього типу делегату
               GetDigitsCout getDigitFiveCout;

               //Присвоїти змінній лямбда-вираз
               getDigitFiveCout = x =>
               {
                    int number = Math.Abs(x);
                    int count = 0; // кількість цифр '5'
                    int digit;
                    while (number > 0)
                    {
                         digit = number % 10;
                         if (digit == 5)
                              count++;
                         number /= 10; // поділити на 10
                    }
                    return count;
               };

               //використання лямбда-виразу
               int a = 615;
               int res = getDigitFiveCout(a); // res = 1

               Console.WriteLine("Результат="+res);
               Console.ReadKey();
          }
     }
```

15.4. Приклади використання лямбда-виразів в найпростіших програмах

Приклад 5. Використання лямбда-виразу, в якому обчислюється значення функції $y = \sin^2 x$.

У застосуванні типу Windows Forms спочатку оголошується тип делегату з одним параметром x і результатом типу float.

В обробнику події button1_Click() використовується лямбда-вираз.

```
using System;
using System.Windows.Forms;

 namespace Example5
 {
   public partial class Form1 : Form
    {
```

```
        // оголошення делегату з 1 параметром і результатом типу float
        delegate float GetSinusSquared(float x);

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            // Лямбда-вираз, що отримує 1 параметр і повертає значення
            GetSinusSquared
                    getSinusSquared = x => (float)(Math.Sin(x) * Math.Sin(x));

            // Використання лямбда-виразу
            float z;
            const float Pi = 3.1415f;
            z = getSinusSquared (0.0f); // z = 0
            z = getSinusSquared((float)(Pi / 2.0)); // z = 1
            z = getSinusSquared(0.7f); // z = 0.4150164

            label1.Text = z.ToString();
        }
    }
}
```

Приклад 5. Використати лямбда-вираз з двома параметрами x, y, для яких обчислити значення функції $z = \sin x^2 - 2 \cdot \cos y$.

Оголошення типу делегату з двома параметрами і результатом типу double:

```
delegate double GetTrigonometricValue(double x, double y);
```

Виклик лямбда-виразу з іншого методу (наприклад, обробника події)

```
GetTrigonometricValue getTrigonometricValue;

// оголосити лямбда-вираз з відними параметрами
getTrigonometricValue = (x, y) => Math.Sin(x * x) - 2 * Math.Cos(y);

// використати лямбда-вираз для розрахунку
double t;
t = getTrigonometricValue(0.0, 0.0); // результат t = -2
t = getTrigonometricValue(3.3, 1.8);  // результат t = -0.540028
```

Завдання 1 (самотійно). Дано три різних цілих числа. Реалізувати лямбда-вираз, який знаходить найбільше з цих трьох чисел.

Приклад 6. Реалізація декількох лямбда-виразів, які відповідають одному типу делегату. Нехай оголошено тип делегату, що отримує три параметри цілого типу.

```
// оголошення типу делегату
delegate int GetIntegerValue(int a, int b, int c);
```

Обробник події можна написати так:

```
private void button1_Click(object sender, EventArgs e)
  {
    // Оголосити делегат
    GetIntegerValue getIntegerValue;

    // 1. Описати лямбда-вираз, що отримує 3 параметри x, y, z
    // і повертає максимальне значення з цих параметрів
     getIntegerValue = (x, y, z) => {
                int max = x;
                if (max < y) max = y;
                if (max < z) max = z;
                 return max;
               };

              // використати лямбда-вираз для пошуку максимуму
              int a = 8, b = 5, c = 10;
              int Max;

              Max = getIntegerValue(a, b, c); // Max = 10
              label1.Text = "Максимум = " + Max.ToString();

              Max = getIntegerValue(a + 8, b - 3, c + 1); // Max = 16
              Label2.Text = "Максимум = " + Max.ToString();

             // 2. Лямбда-вираз, що обчислює суму 3-х чисел
             getIntegerValue = (x, y, z) =>
             {
                 int s; s = z + x + y; return s;
             };
             int sum = CM(4, 3, 2);
             label1.Text = "Сума чисел = " + getIntegerValue(4,3,2).ToString();
                                                                  //  = 9

             // 3. Лямбда-вираз, що обчислює добуток з 3-х чисел
             getIntegerValue = (t, u, v) => t * u * v;
             label1.Text = $"Добуток = {getIntegerValue(5, 3, 10)}”; // = 150
        }
```

У прикладі реалізовано 3 лямбда-вирази одного типу `GetIntegerValue`, які виконують різні операції:

- знаходять максимальне значення;
- знаходять суму параметрів;
- знаходять добуток параметрів.

Отже делегат одніє сигнатури (оголошена в типі) може вказувати на код різного призначення.

Такий підхід є ефективним при написанні великих програмних систем, коли різні фрагменти коду (об’єкти) дають сигнали одному делегату про те, що йому потрібно виконати якусь роботу. Яку саме роботу має виконати делегат – визначається в

залежності від ситуації та об’єкту який згенерував повідомлення. У результаті генеруються події, які обробляються з допомогою делегатів

Приклад 7. Написати програму, яка обчислює середнє арифметичне масиву цілих чисел. Для обчислень використати лябда-вираз.

```
// Оголошення типу делегату з параметром - масив цілих чисел
delegate double AverageArray(int[] a);
```

Реалізація іншому програмному коді (наприклад, обробнику події) лямбда-виразу:

```
private void button1_Click(object sender, EventArgs e)
   {
     // Оголошення лямбда-виразу, що отримує масив цілих чисел
     // Лямбда-вираз повертає середнє арифметичне в масиві A
      AverageArray averageArray = (int[] a) =>
                    {
                     int i;
                     double avg = 0;

                     for (i = 0; i < A.Length; i++)
                         avg += A[i];

                    avg /= A.Length;
                    return avg;
                    };

            // Оголошення масиву array
            int[] array = new int[5];

            // заповнення масиву array деякими значеннями
            for (int i = 0; i < array.Length; i++)
                array[i] = i * i;

            // виклик лямбда виразу
            double Avg = averageArray(array); // Avg = 6
        }
```

Приклад 7. Написати прогараму, яка виводить на форму дані про Автора книги, Назву, Рік видання та Ціну. Використат лямбда-вираз з вхідним параметром структурою.

У програмі потрібно оголосити структуру тBook, а потім делегат з параметром структура Book.

В обробнику події *button1_Click()* потрібно оголосити лямбда-вираз, що виводить інформацію про вміст структури в елементи управління label1, label2, label3, label4 типу Label Windows форми.

Додамо до проекту файл Book.cs

```
namespace WindowsExample7
{
    struct Book
    {
        public string Author;
```

```
        public string Title;
        public int Year;
        public float Price;
    }
}
```

Змінимо файл Form1.cs

```
delegate void DisplayBook(Book book);
private void button1_Click(object sender, EventArgs e)
    {
            // Лямбда-вираз, що отримує параметром структуру BOOK
            DisplayBook displayBook = (b) =>
            {
                // Виведення інформації про структуру на форму
                label1.Text = b.Author;
                label2.Text = b.Title;
                label3.Text = b.Year.ToString();
                label4.Text = b.Price.ToString();
            };

            // змінна типу структура
            Book book;

            book.Author = "Сергій Жадан ";
            book.Title = "Інтернат";
            book.Year = 2017;
            book.Price = 170.50f;

            // Виклик лямбда-виразу
            displayBook(book);
    }
```

Приклад 8. Написати програму, яка обчислює найбільшу відстань між двома точками, координати яких збережені в масиві. Кожен елемент масиву є структура типу Point, яка описує точку на координатній площині.

Програмний код модуля, створеного за шаблоном Windows Forms Application.

```
struct Point
  {
    int x; int y;
    public int X
    {
       get { return x; }
    }
  public int Y
   {
     get { return y; }
   }
   public void SetPoint(int x, int y) // You can use also contructor Point
   {
     this.x = x;
     this.y = y;
   }
```

```
 };
delegate double GetMaxLenght(Point[] p);
private void button1_Click_1(object sender, EventArgs e)
 {
     // Оголошення змінної-делегату типу GetMaxLenght
     GetMaxLenght getMaxLenght;

     // оголошення лямбда-виразу, що обчислює відстань між двома найбільш
     // віддаленими точками масиву структур array
       getMaxLenght = (Point[] array) =>{
       double x1, x2, y1, y2, d, max = 0;

       for (int i = 0; i < array.Length - 1; i++)
         for (int j = i + 1; j < array.Length; j++)
         {
           x1 = array[i].X; y1 = array[i].Y;
           x2 = array[j].X; y2 = array[j].Y;

           d = Math.Sqrt((x1-x2) * (x1-x2) + (y1-y2) * (y1-y2));
           if (max < d)
             max = d;
         }
       return max;
      };
     // Масив з 5 структур типу Point
     Point[] points = new Point[5];

     // ініціалізація полів структур деякими значеннями
     for (int i = 0; i < 5; i++)
       points[i].SetPoint(i+1, i*2+3);// points[i] =new Point(i+1, i*2+3);

     // Використання лямбда-виразу
     label1.Text = getMaxLenght(points).ToString();// = 8.944
   }
```

Приклад 9. Написати WPF-застосування з використанням лямбда-виразу, що виводить на форму координати точки x, y. Параметром лямбда-вираз має бути об’єкт класу *Point*, який описує точку на екрані монітору.

Передача об’єкта класу в лямбда-вираз здійснюється так само як і передача екземпляра структури.

Оголошення в файлі Point.cs клас *Point*.

```
namespace WpfApp1
{
    class Point
    {
        int x;
        int y;
        int color;

        public int X
        {
```

```
            get => x;
            set => x = value;
        }
        public int Y
        {
            get => y;
            set => y = value;
        }
        public int Color
        {
            get => color;
            set => color = value;
        }
        public Point(int x, int y, int color)
        {
            this.x = x;
            this.y = y;
            this.color = color;
        }
    }
}
```

Оголошення делегату в файлі `MainWindow.xaml.cs` *з вхідним параметром об'єкт класу Point.*

```
namespace WpfApp1
{
    public partial class MainWindow : Window
    {
        // Оголосити тип делегату з параметром об'єкт класу Point
        delegate void PointInfo(Point point);

        public MainWindow()
        {
            InitializeComponent();
        }
        private void button_Click(object sender, RoutedEventArgs e)
        {
            // Оголосити змінну-делегат displayInfo та
            //ініціалізувати її кодом лямбда-виразу
            DispalyPointInfo displayInfo = (Point p) =>
            {
                // виведення на форму координат x, y
                label1.Content = "Coordinate X=" + p.X.ToString();
                label2.Content =  string.Format($"Coordinate Y ={p.Y}");
            };
            //виклик лямбда-виразу
            Point point = new Point(5, 6, 11);
            // виведення на форму координат за допомогою делегату
            displayInfo(point);
  }
 }
}
```

Файл MainWindow.xaml

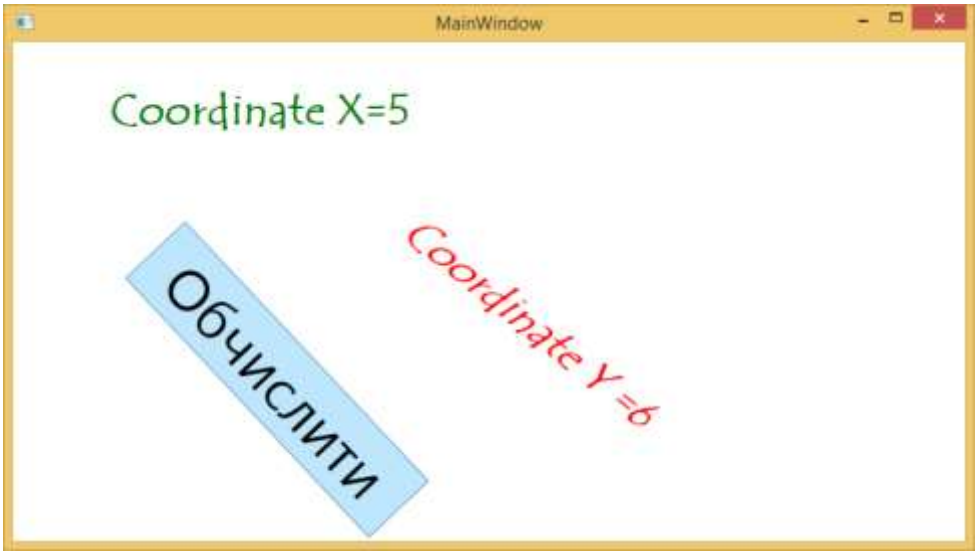
```
<Window x:Class="WpfApp1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WpfApp1"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800" Foreground="#FFDAA8A8">

    <Grid>
        <Button x:Name="button" Content="Обчислити"  HorizontalAlignment="Left"
Margin="71,244,0,0" VerticalAlignment="Top" Width="293" Background="#FFE0B5B5"
BorderBrush="#FF5D3232"  FontSize="48" RenderTransformOrigin="0.5,0.5"
Click="button_Click">

            <Button.RenderTransform>
                <TransformGroup>
                    <ScaleTransform/>
                    <SkewTransform/>
                    <RotateTransform Angle="46.788"/>
                    <TranslateTransform/>
                </TransformGroup>
            </Button.RenderTransform>
        </Button>

        <Label x:Name="label1" Content="Label" Foreground="Green"
           HorizontalAlignment="Left" Margin="75,28,0,0"
      VerticalAlignment="Top"
           Height="59" Width="547" FontSize="36" FontWeight="Bold"
      FontFamily="Tempus Sans ITC"/>

        <Label x:Name="label2" Content="Label" Foreground="Red"
           HorizontalAlignment="Left" Margin="279,244,0,0"
      VerticalAlignment="Top"
           Height="56" Width="395" FontSize="36" FontWeight="Bold"
      FontFamily="Tempus Sans ITC"
           RenderTransformOrigin="0.5,0.5" FontStyle="Oblique">
            <Label.RenderTransform>
                <TransformGroup>
                    <ScaleTransform/>
                    <SkewTransform/>
                    <RotateTransform Angle="39.035"/>
                    <TranslateTransform/>
                </TransformGroup>
            </Label.RenderTransform>
        </Label>
    </Grid>
</Window>
```



Приклад 10. Написати WPF-засосування з використанням лямбда-виразу, що виводить на WPF форму відстань між двома точками масиву точок. Параметрами лямбда-виразу має бути масив об'єктів класу *Point*, який описує точку на екрані монітору та індекси точок в масиві.

В файлі Point.cs оголосимо клас **Point.**

```
namespace WpfApp1
{
    class Point
    {
        int x;
        int y;
        int color;

        public int X
        {
            get => x;
            set => x = value;
        }
        public int Y
        {
            get => y;
            set => y = value;
        }
        public int Color
        {
            get => color;
            set => color = value;
        }

        public Point(int x, int y, int color)
        {
            this.x = x;
            this.y = y;
```

```
            this.color = color;
        }
    }
}
```

Оголошення делегату має вигляд:

```
delegate double SegmentLength(Point[] points, int index1, int index2);
```

В обробнику події button1_Click() реалізовано лямбда-вираз з вхідним параметром масив класів. Для кожного елементу масиву. При передачі масиву структур цього робити не потрібно, тому що структури зберігаються як типи-значення, а класи зберігаються як типи-посилання.

```
using System.Windows;

namespace WpfApp2
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        /// <summary>
        /// Оголошення  делегату
        /// </summary>
        /// <param name="points">масив об'єктів тип клас Point</param>
        /// <param name="index1">індекс першої точки</param>
        /// <param name="index2">індекс другої точки</param>
        /// <returns></returns>
        ///
        delegate double SegmentLength(Point[] points, int index1, int index2);

        public MainWindow()
        {
            InitializeComponent();
        }

        private void button_Click(object sender, RoutedEventArgs e)
        {

            // Реалізація лямбда-виразу - the length of the segment
            SegmentLength segmetLength = (points, i1, i2) => {

                int x1, x2, y1, y2;
                x1 = points[i1].X; y1 = points[i1].Y;
                x2 = points[i2].X; y2 = points[i2].Y;
                return System.Math.Sqrt((x1-x2) * (x1-x2) + (y1-y2) * (y1-y2));
            };

           // виділення пам'яті для масиву з 5 об'єктів
            Point[] myPoints = new Point[5];
```

```
            // виділення пам'яті для кожного елементу масиву
            for (int i = 0; i < myPoints.Length; i++)
                myPoints[i] = new Point();

            // заповнення масиву myPoints довільними значеннями
            for (int i = 0; i < 5; i++)
            {
                myPoints[i].X = 2 * i;
                myPoints[i].Y = 3 * i + 4;
                myPoints[i].Color= i + 3;
            }
            // виклик лямбда-виразу
            double length = segmetLength(myPoints, 2, 3);  //3.605551

            // виведення результату на форму
            label1.Content = string.Format($"Segment’s lenght is {length}");
        }
    }
}
```