Тема 1. Делегати та події

1.1. C # - делегат

Функція може мати один або декілька параметрів різних типів даних, проте параметром також може і інша функція, яка називається делегатом. Затдопомогою делегатів реалізуються функції зворотного виклику та обробники подій.

1.1.1. Створення делегату

Делегат є вказівником на метод і належить до типів даних посиланням. Усі делегати є похідними від класу **System.Delegate** . Делегат може бути оголошений за допомогою ключового слова **delegate** та наступним описом сигнатури функцій на які він може вказувати.

Синтаксис опису делегата

```
<модифікатор> delegate <тип_результату> <ім'я_делегата> (<параметри>);
```

Приклад 1. Оголошення делегату

```
public delegate void Print (int value);
```

Делегат **Print** може бути використаний для вказівки на будь-який метод, який має такий же тип результату. В наступному прикладі цей делегат використовується щоб вивести декілька чисел.

Приклад 2. Делегат C

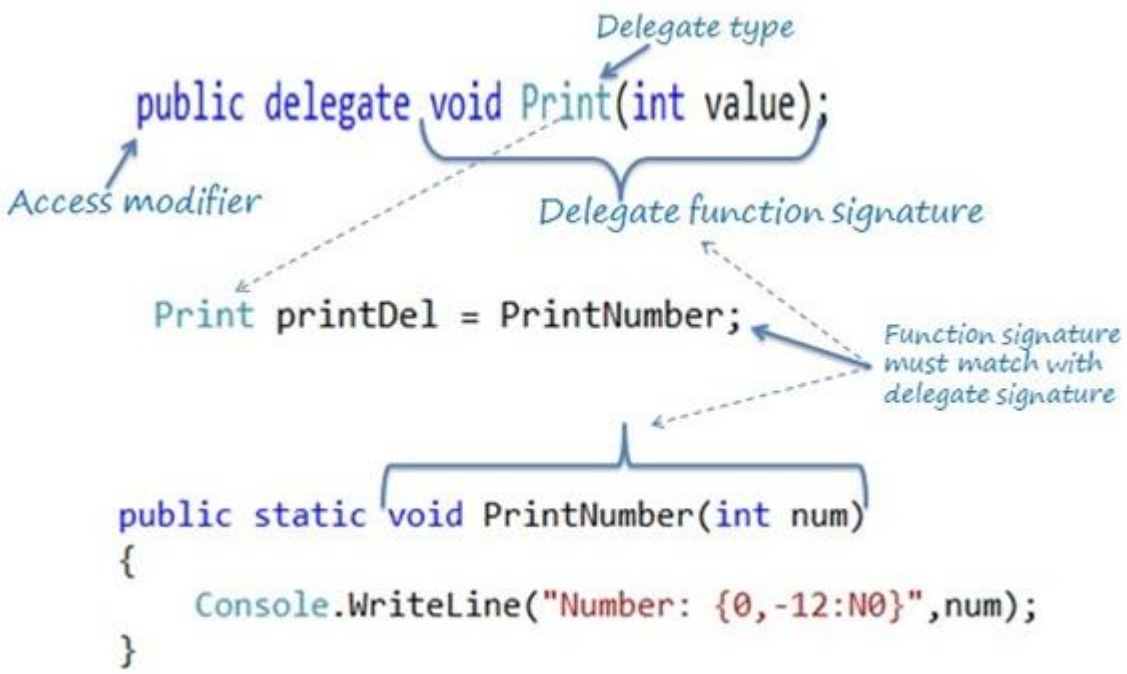
```
class Program
{
     public delegate void Print(int value); // declare delegate
     static void Main(string[] args)
     {
        Print printDel = PrintNumber; // Print delegate points to PrintNumber
        // or  Print printDel = new Print(PrintNumber);

        printDel(100000);
        printDel(200);

         printDel = PrintMoney; // Print delegate points to PrintMoney
         printDel(10000);
         printDel(200);
    }
    public static void PrintNumber(int num)
    {
        Console.WriteLine("Number: {0,-12:N0}",num);
    }
    public static void PrintMoney(int money)
    {
        Console.WriteLine("Money: {0:C}", money);
    }
```

```
}
```

Output:

```
Number: 10,000
Number: 200
Money: $ 10,000.00
Money: $ 200.00
```

У прикладі вище оголошено делегат **Print,** який отримує параметр типу *int* і не повертає ніякого результату. У методі Main () оголошується змінна **PrintDel** типу **Print,** якій присвоюється адреса методу **PrintNumber**. Тепер, якщо викликати змінну-делегат **PrintDel** то вона буде, фактично, викликати метод **PrintNumber**. Таким же чином змінній **PrintDel** присвоюється адреса методу **PrintMoney**, то вона буде викликати і метод **PrintMoney** в іншій частині програми.
Наступний малюнок ілюструє використання делегата.



Делегат в C #

Зауваження. Об'єкт-делегат може бути створений за допомогою оператора **new** та вказівки імені методу, як показано нижче:

```
Print printDel = new Print(PrintNumber);
```

1.1.2. Виклик делегата

Делегат можна викликати як метод, тому що він є посиланням на метод. Використовувати делегат можна двома способами. Перший розглянули в попередньому прикладі – змінній-делегати присвоїли посилання на метод і передали йому значення параметра. В другому способі у змінній-делегаті використовується метод Invoke() для виклику методу на який вона вказує (див. нижче).

Приклад 3. Виклик делегата

```
Друк printDel = PrintNumber;
printDel.Invoke (10000);  // або printDel (10000);
```

Вихідні дані

```
Номер: 10000
Номер: 10000
```

1.1.3. Передача делегата, як параметра

Метод може мати параметр типу делегат і викликати його.

Приклад 4. Параметр-делегат

```
public static void PrintHelper(Print delegateFunc, int numToPrint)
{
    delegateFunc(numToPrint);
}
```

У прикладі метод **PrintHelper** має параметр делегата типу `Print` і викликає його як функцію: **`delegateFunc (numToPrint)`**.

Наступний приклад показує, як можна використати метод PrintHelper.

Приклад 5. Використання параметру делегата

```
class Program
{
    public delegate void Print(int value);
    static void Main(string[] args)
    {
        PrintHelper(PrintNumber, 10000);
        PrintHelper(PrintMoney, 10000);
    }

    public static void PrintHelper(Print delegateFunc, int numToPrint)
    {
           delegateFunc(numToPrint);
    }
    public static void PrintNumber(int num)
    {
        Console.WriteLine("Number: {0,-12:N0}",num);
    }
    public static void PrintMoney(int money)
    {
        Console.WriteLine("Money: {0:C}", money);
    }
}
```

Output:

```
Number: 10,000
Money: $ 10,000.00
```

1.1.4. Багатоадресні делегати

Делегат може вказувати на декілька методів. Такий делегат називається *багатоадресним (multicast delegate)*.

Оператор "+" додає функцію в список посилань об'єкта-делегата, а оператор "-" видаляє існуючу функцію з цього списку.

Приклад 6. Багатоадресний делегат

```
public delegate void Print(int value);
static void Main(string[] args)
{
   Print printDel = PrintNumber;
         printDel += PrintHexadecimal;
         printDel += PrintMoney;
         printDel(1000);

         printDel -=PrintHexadecimal;
         printDel(2000);
}
public static void PrintNumber(int num)
{
   Console.WriteLine("Number: {0,-12:N0}",num);
}
public static void PrintMoney(int money)
{
   Console.WriteLine("Money: {0:C}", money);
}
public static void PrintHexadecimal(int dec)
{
   Console.WriteLine("Hexadecimal: {0:X}", dec);
}
```

Output:

```
Number: 1,000
Hexadecimal: 3EB
Money: $ 1,000.00
Number: 2,000
Money: $2,000.00
```

В прикладі змінна **`printDel`** типу делегат **Print** є багатоадресним(груповим) об’єктом-делегатом, який вказує на три методи - **PrintNumber** , **PrintMoney** і **PrintHexadecimal** . Виклик printDel призводе до **послідовного** виклику всіх трьох методів.

1.2. C # - анонімні методи

1.2.1. Визначення анонімного методу

Як випливає з назви, анонімний метод - це метод без імені. Анонімні методи в C# можуть бути визначені за допомогою ключового слова delegate і оператора присвоєння блоку коду змінній-делегату.

Приклад 6. Анонімний метод

```
public delegate void Print(int value);
 static void Main(string[] args)
```

```
{
    Print print = delegate(int val)
      {
              Console.WriteLine("Inside Anonymous method. Value: {0}", val);
      };      print(100); }
```

Output:

Inside Anonymous method. Value: 100

Анонімні методи можуть отримати доступ до змінних, визначених у зовнішній функції.

Приклад 14. Анонімний метод

```
публічний делегат void Print ( значення int );
static void Main ( string [] аргументи)
{      int i = 10;
      Друкувати prnt = delegate ( int val)
       { val + = i;
          Консоль .WriteLine ( "Анонімний метод: {0}" , val);
       };
    prnt (100);
}
```

Вихід :

Анонімний метод: 110

Анонімні методи також можуть бути передані методу, який приймає делегат як параметр. У наступному прикладі **PrintHelperMethod ()** приймає перші параметри делегата **Print** :

Приклад 15. Анонімний метод як параметр

```
public delegate void Print(int value);
static void Main(string[] args)
{     int i = 10;
      Print prnt = delegate(int val)
       {  val += i;
          Console.WriteLine("Anonymous method: {0}", val);
       };
    prnt(100);
}
```

Output:

Anonymous method: 110

Анонімні методи можуть використовуватися як обробники подій:

Приклад 16. Анонімний метод як обробник подій

```
saveButton.Click + = делегат ( Object o, EventArgs e) {
    Система .Windows.Forms.MessageBox.Show ( "Зберегти успішно!" );
};
```

У версії C # 3.0 місить лямбда-вирази, які однією з форм анонімних методів.

1.3.2. Обмеження анонімного методу

- Не може містити оператор стрибка, як-от goto, break або продовжувати.
- Не може отримати доступ до параметра ref або out зовнішнього методу.
- Не може мати або не мати доступу до небезпечного коду.
- Його не можна використовувати з лівого боку оператора.

Висновки

1. Делегат є вказівником на функцію.
2. Синтаксис опису делегата

```
<модифікатор> delegate <тип_результату> <ім'я_делегата> (<параметри>);
```

3. Метод, адреса якого присвоюється делегату, повинен мати таку саму сигнатуру, що і делегат.
4. Делегаті можна викликати як звичайну функцію або за допомою методу Invoke ().
5. Багатоадресний делегат може містити декілька методів, які додаються в його список методів за допомогою оператора "+".
6. Анонімний метод можна визначити за допомогою ключового слова delgate.
7. Анонімний метод повинен бути призначений делегату.
8. Анонімний метод може отримати доступ до зовнішніх змінних або функцій.
9. Анонімний метод може передаватися як параметр.
10. Анонімний метод може використовуватися як обробник подій.