

Использование делегатов

Многократная адресация

- экземпляр делегата может ссылаться не на один, а сразу на несколько методов
- при вызове делегата выполняется цепочка вызовов: последовательно вызываются методы, на которые ссылается вызываемый экземпляр делегата
- Для добавления ссылки на метод экземпляру делегата используется оператор +=
- Для удаления ссылки на метод экземпляру делегата используется оператор -=

Листинг 2. многократная адресация

```
using System;

delegate void MyDelegate(); //объявление делегата

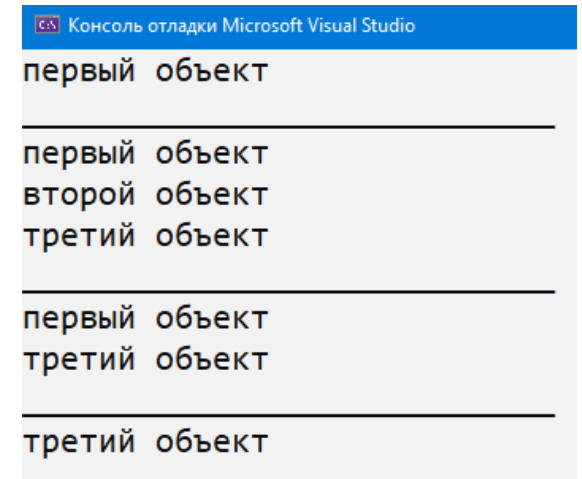
namespace MultipleDelegate
{
    class MyClass
    {
        public string name;
        public MyClass( string text)
        {
            name = text;
        }
        public void Show()
        {
            Console.WriteLine(name);
        }
    }
}
```

```
internal class Program
{
    static void MakeLine()
    { Console.WriteLine("_____"); }
    static void Main(string[] args)
    { //создание объектов
        MyClass obj1 = new MyClass("первый объект");
        MyClass obj2 = new MyClass("второй объект");
        MyClass obj3 = new MyClass("третий объект");

        MyDelegate meth; //объявление переменной делегата
        meth = obj1.Show; // присваивание переменной делегата ссылки на метод
        meth(); // вызов экземпляра делегата
        meth = MakeLine; // присваивание переменной делегата нового значения
        // добавление методов в список вызова
        meth += obj1.Show;
        meth += obj2.Show;
        meth = meth + obj3.Show;
```

```
    // вызов экземпляра делегата  
    meth();  
    // удаление метода из списка вызова 1 способ  
    meth -= obj2.Show;  
    // вызов экземпляра делегата  
    meth();  
    // удаление метода из списка вызова 2 способ  
    meth = meth - obj1.Show;  
    // вызов экземпляра делегата  
    meth();  
}  
  
}  
  
}
```

Результат выполнения программы



```
Консоль отладки Microsoft Visual Studio  
первый объект  
-----  
первый объект  
второй объект  
третий объект  
-----  
третий объект  
-----  
третий объект
```

Листинг 3. экземпляр делегата как поле класса

```
using System;

delegate void MyDelegate(string txt);

namespace DelegateAsFieldDemo
{
    class MyClass
    {
        //поле является ссылкой на экземпляр делегата
        public MyDelegate apply;
        public MyClass(MyDelegate md)
        {
            apply=md;
        }
    }

    class Alpha
    {
        private string name;

        public void set(string t)
        {
            name=t;
        }

        public override string ToString()
        {
            return name;
        }
    }
}
```

```
internal class Program
{
    static void Main(string[] args)
    {
        Alpha A = new Alpha(); //создание объекта

        //создание объекта (аргумент конструктора - ссылка на метод)
        MyClass obj = new MyClass(A.set);

        obj.apply("объект A");      // вызов экземпляра делегата
        Console.WriteLine(A);      // проверка поля объекта

        Alpha B = new Alpha();      //создание объекта
        obj.apply=B.set;             // полю значением присваивается ссылка на метод
        obj.apply("объект B");      // вызов экземпляра делегата
        Console.WriteLine(B);      // проверка поля объекта
    }
}
```

Листинг 3. продолжение

```
//добавление метода в список вызовов экземпляра делегата
obj.apply+=A.set;

obj.apply("объект X"); //вызов экземпляра делегата

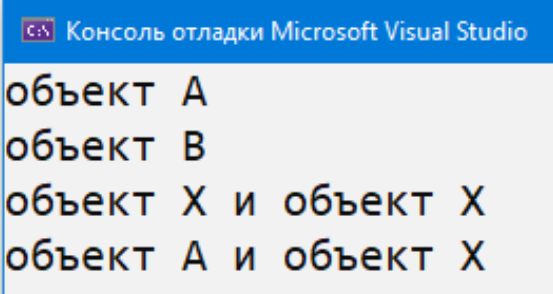
Console.WriteLine(A + " и " + B); //проверка полей объектов

obj.apply-=B.set;      // удаление метода из списка вызовов экз делегата

obj.apply("объект A"); //вызов экземпляра делегата

Console.WriteLine(A + " и " + B); // проверка полей объектов
}
}
}
```

Результат выполнения программы



Консоль отладки Microsoft Visual Studio

```
объект A
объект B
объект X и объект X
объект A и объект X
```


Листинг 4. передача метода в качестве аргумента

```
using System;

delegate int MyDelegate(int n);

namespace DelegateAsArgDemo
{
    internal class Program
    {
        //статический метод для вычисления нечетных чисел
        static int OddNumber(int n)
        { return 2*n+1; }

        // статический метод для вычисления четных чисел
        static int EvenNumber(int n)
        { return 2*n; }

        //статический метод для вычисления квадратов чисел
        static int SquareNumber(int n)
        { return n*n; }
    }
}
```

Листинг 4. продолжение

```
// статический метод, которому аргументом передается ссылка на метод
static void Display(MyDelegate F, int a, int b)
{
    Console.WriteLine("{0,-4}|{1,4}", "x", "F(x)");
    Console.WriteLine("-----");
    for (int k = a; k <= b; k++)
    {
        //команда с вызовом экземпляра делегата
        Console.WriteLine("{0,-4}|{1,4}", k, F(k));
    }
    Console.WriteLine();
}

static void Main(string[] args)
{
    int a = 0, b = 5;
    Console.WriteLine("нечетные числа");
    Display(OddNumber, a, b);
    Console.WriteLine("четные числа");
    Display(EvenNumber, a, b);
    Console.WriteLine("число в квадрате");
    Display(SquareNumber, a, b);
}
}
```

Результат выполнения программы

нечетные числа	
x	F(x)
0	1
1	3
2	5
3	7
4	9
5	11
четные числа	
x	F(x)
0	0
1	2
2	4
3	6
4	8
5	10
число в квадрате	
x	F(x)
0	0
1	1
2	4
3	9
4	16
5	25