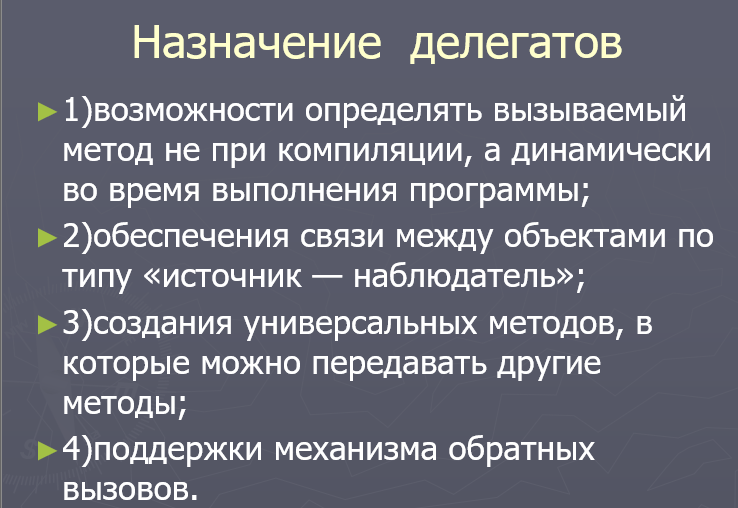
1. Что такое делегат? Как определить делегат?

Делегат — это объект, предназначенный для хранения ссылок на методы (с определенным списком параметров и типом возвращаемого значени).

delegate void MyDelegate(int number);

1. Назначение делегатов.



### 

1. Какие есть способы присваивания делегату адреса метода?

Делегату можно назначить любой метод из любого доступного класса или структуры, соответствующий типу делегата.

**1)С использованием имени метода**

Example example = new Example();

MyDelegate del = example.DisplayNumber; // Присвоение адреса метода

**2. Использование анонимных методов (ключевого слова delegate)**

Анонимные методы позволяют создавать методы "на лету" и присваивать их делегату.

MyDelegate del = delegate (int number)

{

Console.WriteLine($"Anonymous Number: {number}");

};

del(10); // Вызов анонимного метода

**3. Использование лямбда-выражений**

Лямбда-выражения — это более современный и компактный способ определения методов, которые можно присваивать делегатам.

MyDelegate del = (number) => Console.WriteLine($"Lambda Number: {number}");

del(15); // Вызов метода через делегат

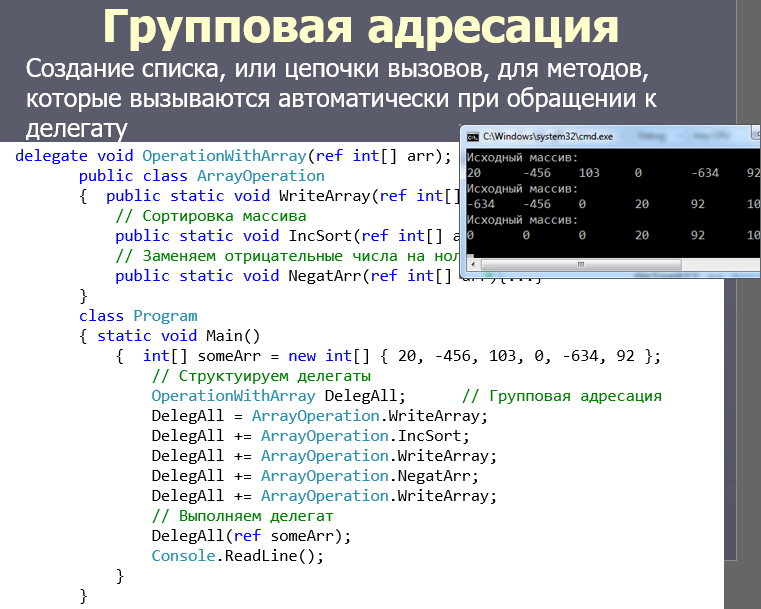
1. Поясните назначение метода Invoke.

Метод Invoke в делегатах используется для вызова метода, на который ссылается делегат.

 **Обработка исключений:**  
При вызове метода через Invoke, если метод выбрасывает исключение, оно будет передано обратно в код, который вызвал Invoke. Это позволяет обрабатывать исключения, возникающие в методах, вызываемых через делегаты.

 **Передача параметров:**  
Метод Invoke позволяет передавать параметры методу, на который указывает делегат, что делает его гибким инструментом для вызова различных методов с разными параметрами.

1. Что такое групповая адресация делегата?

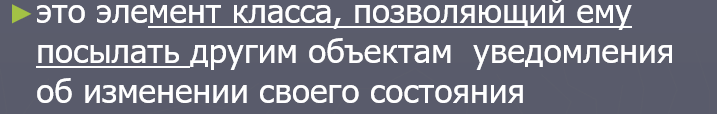


это возможность привязывать несколько методов к одному делегату

с помощью оператора +=

последовательно вызывает все методы, на которые он ссылается.

1. Как создать событие?



События объявляются в классе с помощью ключевого слова event, после которого указывается тип делегата, который представляет событие

public delegate void Fine(User obj, int position);

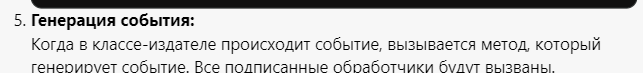
public event Fine FineEvent;

7. Как события связаны с делегатами? Опишите и поясните схему

взаимодействия.

События в C# связаны с делегатами, так как делегаты определяют сигнатуру методов, которые могут быть связаны с событием.





8. Что такое лямбда-выражения? Приведите пример лямбда-выражения

с несколькими параметрами.

Лямбда-выражения представляют упрощенную запись анонимных методов. Лямбда-выражения могут содержать один или несколько параметров и могут использоваться для описания функций, которые передаются как аргументы.

MathOperation add = (x, y) => x + y;

9. Что такое ковариантность и контравариантность делегатов? Что это

дает?

**Ковариантность: позволяет использовать более конкретный тип, чем заданный изначально** Ковариантность позволяет присваивать делегатам методы, которые имеют более производные (более специфические) типы параметров.

То есть, если делегат ожидает метод с базовым типом параметра, то можно присвоить ему метод с производным типом параметра.

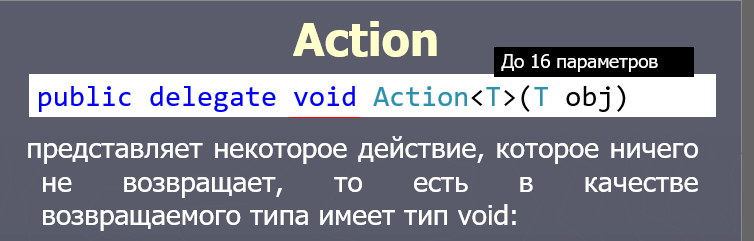
**Контравариантность: позволяет использовать более универсальный тип, чем заданный изначально.** Контравариантность позволяет присваивать делегатам методы, которые имеют более общие (более базовые) типы параметров.

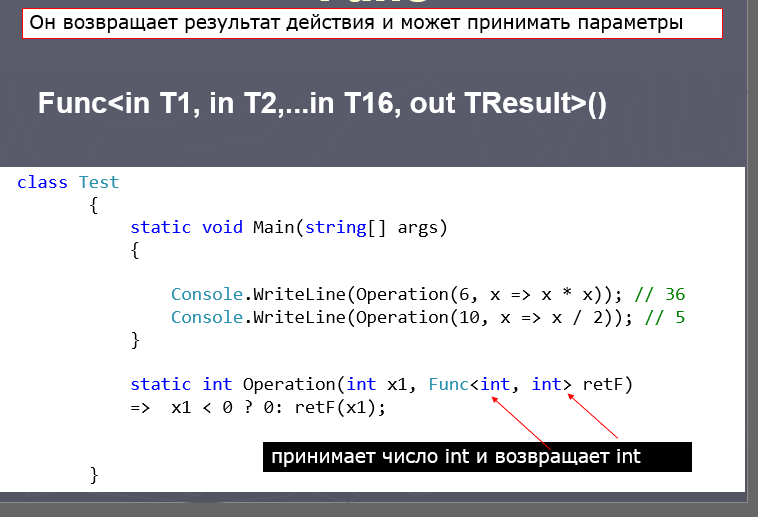
То есть, если делегат ожидает метод с производным типом параметра, то можно присвоить ему метод с базовым типом параметра.

### Что это дает?

1. **Повышение переиспользуемости:** Это дает возможность использовать общие интерфейсы и базовые классы для работы с производными типами, что улучшает реиспользуемость кода.
2. **Улучшение типизации:** Ковариантность и контравариантность делают код более безопасным, позволяя разработчикам использовать строгую типизацию, что помогает избежать ошибок времени выполнения.

10.Поясните разницу между встроенными делегатами Action и Func





Func возвращает результат действия и может принимать параметры. Он также имеет различные формы:

от Func<out T>(), где T - тип возвращаемого значения,

до Func<in T1, in T2,...in T16, out TResult>(),

то есть может принимать до 16 параметров.