На какие методы может ссылаться делегат? (методы с таким же типом возвращаемого значения и теми же типами параметров (порядок следования типов важен, кстати (int, string != string, int) могут быть в делегате, ну то есть делегат может на них указывать)

Как соединить делегат и событие?

Чтобы связать событие с уже существующим экземпляром делегата, нужно использовать оператор + =

ссылки на какие методы назначаются делегату?

делегат может ссылаться на любой тип объекта, если для этого объекта есть метод, соответствующий сигнатуре делегата. Если делегат создан в качестве оболочки для статического метода, он ссылается только на метод.

Как соединить делегат и метод?

Делегат можно связать с именованным методом. При создании экземпляра делегата с использованием именованного метода этот метод передается в качестве параметра

Как соединить делегат, событие и метод?

Параметр стринг будет реф (может)

Да, тогда это будет передаваться по ссылке, а не значению

Как соединить делегат без события?

Через запятую?

Для чего нужно событие?

сигнализируют системе о том, что произошло определенное действие

Как подписаться на событие?

Чтобы присоединить обработчик событий к событию, используйте оператор присваивания сложения (+=).

Чтобы указать обработчик событий, можно также воспользоваться лямбда-выражением

Событие может существовать без делегата

Я считаю, что технически вы можете проводить мероприятия без делегата, как описано в ответах здесь .

Однако делегаты используются для передачи информации о том, откуда было вызвано событие. Без него программа не знает, откуда было вызвано событие и какую информацию передать.

Где событие подкрепляется к делегату?

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

Как создать цепочку из делегатов? (Цепочка делегатов — это набор объектов-делегатов, которой позволяет вызывать все методы, представленные делегатами набора. выстроить цепочку по которой сами по себе запускаются методы)

Func от Action отличается

Делегат Action является обобщенным, предназначен для ссылки на метод, принимает параметры и возвращает значение void. Как правило, этот делегат передается в качестве параметра метода и предусматривает вызов определенных действий в ответ на произошедшие действия.

**Func** возвращает результат действия и может принимать параметры, Данный делегат также часто используется в качестве параметра в методах.

Func и Action что указываем в параметрах

Func: Тип и сам параметр метода, инкапсулируемого этим делегатом.

Action:

Func и Action ссылаются на какие методы

на методы с любым возвращаемым значением

На какой метод может ссылаться Func в котором сколько параметров

На метод с таким же числом параметров

1. Что такое делегат? Как определить делегат?

*Делегат* – это указатели на методы и с помощью делегатов мы можем вызвать данные методы.

Для объявления делегата используется ключевое слово **delegate**, после которого идет возвращаемый тип, название и параметры.

Делегат может хранить ссылки на несколько методов и вызывать их поочередно - сигнатуры всех методов должны совпадать

2. Назначение делегатов.

1)возможности определять вызываемый метод не при компиляции, а динамически во время выполнения программы;   
2)обеспечение связи между объектами по типу «источник — наблюдатель»;   
3)создание универсальных методов, в которые можно передавать другие методы;   
4)поддержка механизма обратных вызовов.  
3. Какие есть способы присваивания делегату адреса метода?

Для объявления делегата используется ключевое слово **delegate**, после которого идет возвращаемый тип, название и параметры.

delegate void Message();  
Есть еще один способ - создание объекта делегата с помощью конструктора, в который передается нужный метод:  
delegate void SomeDel(int a, double b);  
Добавление методов в делегат  
се методы в делегате попадают в специальный список - список вызова или invocation list. И при вызове делегата все методы из этого списка последовательно вызываются. И мы можем добавлять в этот спиок не один, а несколько методов:  
class Program

{

    delegate void Message();

    static void Main(string[] args)

    {

        Message mes1 = Hello;

        mes1 += HowAreYou;  // теперь mes1 указывает на два метода

        mes1(); // вызываются оба метода - Hello и HowAreYou

        Console.Read();

    }

    private static void Hello()

    {

        Console.WriteLine("Hello");

    }

    private static void HowAreYou()

    {

        Console.WriteLine("How are you?");

    }

}

Делегаты можно объединять в другие делегаты.   
 static void Main(string[] args)

    {

        Message mes1 = Hello;

        Message mes2 = HowAreYou;

        Message mes3 = mes1 + mes2; // объединяем делегаты

        mes3(); // вызываются все методы из mes1 и mes2

        Console.Read();

    }

4. Поясните назначение метода Invoke.

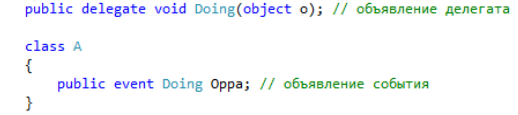
Вызов делегата

5. Что такое групповая адресация делегата?

Создание списка, или цепочки вызовов, для методов, кот. вызываются авто- при обращении к делегату

6. Как создать событие?

**Событие** – это элемент класса, позволяющий ему посылать другим объектам уведомления об изменении своего состояния

События построены на основе делегатов: с помощью делегатов вызываются методы-обработчики событий. Поэтому создание события в классе состоит из следующих частей:   
♣ описание делегата, задающего сигнатуру обработчиков событий;   
♣ описание события;   
♣ описание метода (методов), инициирующих событие  
  
7. Как события связаны с делегатами? Опишите и поясните схему взаимодействия.

События построены на основе делегатов: с помощью делегатов вызываются методы-обработчики событий.

8. Что такое лямбда-выражения? Приведите пример лямбда-выражения с несколькими параметрами.

Анонимные ф-и – безымянный кодовый блок, передаваемый конструктору делегата

Лямбда-выражения – упрощенная запись анонимных методов (нет return, типа)

9. Что такое ковариантность и контрвариантность делегатов? Что это дает?

Ковариантность – позволяет возвращать из метода объект, тип которого является производным от типа, возвращаемого делегатом.

delegate Person PersonFactory(string name);

static void Main(string[] args)

{

    PersonFactory personDel;

    personDel = BuildClient; // ковариантность

    Person p = personDel("Tom");

    Console.WriteLine(p.Name);

    Console.Read();

}

private static Client BuildClient(string name)

{

    return new Client {Name = name};

}

Контрвариантность – предполагает возможность передавать в метод объект, тип которого является более универсальным по отношению к типу параметра делегата.  
delegate void ClientInfo(Client client);

static void Main(string[] args)

{

    ClientInfo clientInfo = GetPersonInfo; // контравариантность

    Client client = new Client{Name = "Alice"};

    clientInfo(client);

    Console.Read();

}

private static void GetPersonInfo(Person p)

{

    Console.WriteLine(p.Name);

}

10. Поясните разницу между встроенными делегатам Action и Func.

Action – приним. параметры, возвр. void  
Func – назначает значения типа int