1. Что такое делегат? Как определить делегат?

*Делегат* – это объект, предназначенный для хранения ссылок на методы(указатель на функцию C++)  
функции обратного вызова + без. Типов  
1) используются для поддержки событий 2) как самостоятельная конструкция языка   
[атрибуты] [спецификаторы]  
 delegate тип имя\_делегата([параметры] )   
 new, public, protected, internal и private.  
Делегат может хранить ссылки на несколько методов и вызывать их поочередно - сигнатуры всех методов должны совпадать

public delegate void D(int i)  
class Class1  
 private static void Hello(int i) { }  
 static void Main()  
 D del //созд. переменную делегата  
 del = Hello //присв. пер-ной адрес метода  
. //del = new D(Hello)  
 del.Invoke(4) //вызываем метод  
 //del(4)

2. Назначение делегатов.

1)возможности определять вызываемый метод не при компиляции, а динамически во время выполнения программы;   
2)обеспечения связи между объектами по типу «источник — наблюдатель»;   
3)создания универсальных методов, в которые можно передавать другие методы;   
4)поддержки 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.

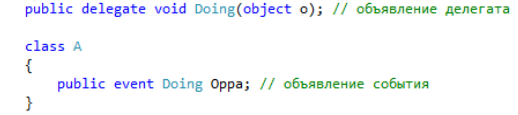
Вызов делегата

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 к-ру делегата

delegate int Summator(int b);  
static int result = 0;  
Summator someDelegat = delegate (int number)  
{  
 for (i=0; i<=number; i++)  
 result +=i; //захват переменной  
 return result;  
}

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

(x,y) => x + y;  
i => i \* i

9. Что такое ковариантность и контр-вариантность 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