

События

	Вопросы
1.	Может ли событие содержать ссылки на несколько методов?
2.	<p>Выберите верные утверждения:</p> <ol style="list-style-type: none"> 1) событие - тип 2) событие можно объявить в блоке исполняемого кода 3) событие всегда является членом класса, либо членом структуры 4) событие хранит ссылки на методы, как и многоадресный делегат 5) событие не может быть статическим
3.	<p>Модификаторами события могут быть(выберете верные варианты)</p> <ol style="list-style-type: none"> 1) static 2) new 3) abstract 4) virtual 5) readonly
4.	Какие модификаторы видимости могут быть применены для событий?
5.	<p>Как подписаться на событие (методы, которые должны запуститься)</p> <ol style="list-style-type: none"> 1. <КлассИлиОбъект>.<ИмяСобытия> ()+= <КлассЧейМетодДолженЗапуститься>.<МетодПодходящийПоСигнатуре>() 2. <ИмяСобытия> += <КлассЧейМетодДолженЗапуститься>.<МетодПодходящийПоСигнатуре>() 3. <ИмяСобытия> += <КлассЧейМетодДолженЗапуститься>.<МетодПодходящийПоСигнатуре> 4. <КлассИлиОбъект>.<ИмяСобытия> += <КлассЧейМетодДолженЗапуститься>.<МетодПодходящийПоСигнатуре> 5. <КлассИлиОбъект>.<МетодПодходящийПоСигнатуре>+= <ИмяСобытия>
6.	<p>Что выведет программа?</p> <pre> delegate void MyEvent1(int n); class MyEvent { public event MyEvent1 Event1; public void OnEvent1(int n) { if (Event1 != null) Event1(n); } } class Program { static void Main() { MyEvent ev =new MyEvent(); ev.Event1 += (n) => Console.WriteLine(n++ * 8); ev.OnEvent1(3); ev.OnEvent1(2); } } </pre>
7.	<p>Встроенный делегат, обрабатывающий событие:</p> <ol style="list-style-type: none"> 1)public delegate void EventHandler(object sender) 2)public delegate void EventHandler(EventArgs e) 3)public delegate void EventHandler(object sender, EventArgs e) 4)public delegate void EventHandler() 5)C# не имеет встроенных делегатов для этого

8.	<p>Выберите верные утверждения:</p> <p>1)Для событий может использоваться модификатор области видимости internal</p> <p>2)События могут реализовываться при помощи модификатора abstract</p> <p>3)События НЕ могут быть членом класса static</p> <p>4)События НЕ могут быть помечены модификатором sealed</p>
9.	<p>Что будет выведено на экран после выполнения данного кода? (***) - ошибка компиляции)</p> <pre> namespace ImSorry { class Program { delegate void del(int x); private static event del growth = p_1; private static void p_1(int y) { growth += (int z) => { Console.WriteLine((int)z*z); }; } static void Main(string[] args) { for (int i = 0; i <= 5; i++) { if (i % 2 == 1) { growth(i); } } } } </pre>
10.	<p>- Что выведет следующая программа?</p> <pre> using System; namespace Task-1 { class MyEventArgs : EventArgs { public MyEventArgs() : base() { } } class Program { static event EventHandler OnEvent; static void Main(string[] args) { Program pr = new Program(); OnEvent += MyMethod; OnEvent(pr, EventArgs.Empty); } static void MyMethod(object sender, MyEventArgs args) { Console.WriteLine("This is my method!"); } } } </pre>

	1) This is my method! 2) Пустую строку 3) Ничего - ошибка компиляции
11.	<p>Выберите верные утверждения:</p> <p>1) Методы экземпляра и статические методы могут быть использованы в качестве обработчиков событий.</p> <p>2) Когда в качестве обработчика используется метод экземпляра, то события адресуются конкретным экземплярам объектов.</p> <p>3) События не могут быть вызваны в теле метода.</p> <p>4) Как и делегаты, события поддерживают групповую адресацию.</p>
12.	<p>Как должны выглядеть .NET Framework совместимые обработчики событий, установленные корпорацией Microsoft:</p> <p>a) protected event обработчик (int a) { //.. }</p> <p>b) public event обработчик (var a) { //.. }</p> <p>c) event обработчик (object отправитель, EventArgs e) { //.. }</p> <p>d) event обработчик (ref a, ref b) { //.. }</p>
13.	<p>Что будет выведено в результате выполнения данного кода (***) - ошибка компиляции)</p> <pre>using System; namespace Application { public class Mass { public delegate void Del(ref double x); public event Del Inventor2; public static void Mas2(ref double x) { x = 100; } public static void Mas3(ref double x) { double sqrt = 0; sqrt = Math.Sqrt(x); Console.WriteLine("Sqrt = " + sqrt); } } class MainClass { public static void Print(double x) { Console.WriteLine(x); } public static void Main(string[] args) { Random gen = new Random(); double x = 0; Mass a = new Mass(); a.Inventor2 += Mass.Mas2; a.Inventor2 += Mass.Mas3; a.Inventor2(ref x); Print(x); } } }</pre>
14.	<p>Выберите неправильные ответы:</p> <p>1. Событие может быть декларировано в перечислении</p> <p>2. Событие может быть декларировано с модификатором abstract</p> <p>3. Событие может быть декларировано с модификатором sealed</p>

	<p>4. Событие может быть декларировано в классе</p> <p>5. Событие может быть декларировано с модификатором readonly</p>
15.	<p>В результате выполнения следующей программы:</p> <pre> delegate void RunHandler(ref byte d); class Program { static event RunHandler onRun; static void Main() { onRun += new RunHandler(one); onRun += new RunHandler(two); byte z = 7; while (z++ < 9) { onRun(ref z); Console.Write(z); } } static void one(ref byte x) { Console.Write(x += 2); } static void two(ref byte y) { Console.Write(y += 1); } } </pre> <p>на экран будет выведено:</p>
16.	<p>Какие из данных ключевых слов могут быть применены к событию?</p> <ol style="list-style-type: none"> 1) sealed 2) extern 3) abstract 4) sealed 5) volatile 6) virtual
17.	<p>Что выведет данный код:</p> <pre> class FirstHandler{ public void Message(){ Console.WriteLine("I'm here!"); } } class SecondHandler{ public void Message(){ Console.WriteLine("And there!"); } } class Program{ static void Main(){ delegate void WhereAreYou(); event WhereAreYou callMe; FirstHandler first = new FirstHandler(); SecondHandler second = new SecondHandler(); callMe += first.Message; callMe += second.Message; callMe(); } } </pre>

	<pre> } } </pre>
18.	<p>Что выведет предоставленный кусок кода? Если возникнет ошибка компиляции, напишите: *** Если ошибок и исключений нет, но на экран не выведется ничего, напишите: — Если возникнет ошибка исполнения или исключение, напишите: +++</p> <pre> public delegate void Delegate(string s); public static event Delegate Event; public static void ConsoleWorker(string message) { Console.WriteLine(message); } static void Main(string[] args) { Delegate del = message => Console.WriteLine(message); Event("Hello"); } </pre>
19.	<p>Выберете верные утверждения</p> <ol style="list-style-type: none"> 1) События могут быть объявлены в классе или в пространстве имен 2) События не запускаются извне классов, в которых они объявлены 3) Событие может быть локальной переменной метода 4) Ключевое слово event используется для объявления события 5) Для подписки метода на событие используется оператор +=
20.	<p>Может ли событие объявлено с модификатором abstract/override?</p>
21.	<p>Что выведет предоставленный кусок кода? Если возникнет ошибка компиляции, напишите: *** Если ошибок и исключений нет, но на экран не выведется ничего, напишите: --- Если возникнет ошибка исполнения или исключение, напишите: +++</p> <pre> class Counter { public delegate string Delegate(); public event Delegate onCount; public void Count() { for (int i = 0; i < 100; i++) { if (i == 71) { onCount(); } } } } class A { public void Message() { Console.WriteLine("Пора действовать, ведь уже 71!"); } } class B { public void Message() { Console.WriteLine("Точно, уже 71!"); } } </pre>

	<pre> } class Program { static void Main(string[] args) { Counter Counter = new Counter(); A Handler1 = new A(); B Handler2 = new B(); Counter.onCount += Handler1.Message; Counter.onCount += Handler2.Message; Counter.Count(); } } </pre>
22.	<p>Выберите верные утверждения:</p> <ol style="list-style-type: none"> 1) Если к событию применить ключевое слово <code>virtual</code>, то это позволит производным классам переопределять поведение события при помощи ключевого слова <code>override</code> 2) Нельзя к событию применить ключевое слово <code>static</code> 3) События поддерживают только операторы <code>+=</code> и <code>-=</code> 4) Аксесорные формы событий могут быть абстрактными
23.	<p><code>event</code> допускает модификаторы:</p> <ol style="list-style-type: none"> 1) <code>sealed</code> 2) <code>virtual</code> 3) <code>abstract</code> 4) <code>static</code> 5) <code>public</code>
24.	<p>Что выведется на экран, после выполнения программы:</p> <p>*** - Ошибка компиляции ** - Необработанное исключение * - Ничего не выведется</p> <pre> namespace ConsoleApp9 { class A { public event EventHandler OnSomething; public void Test() { for (int i = 0; i < 10; i++) { if (i == 1) { OnSomething(); } } } } public delegate void EventHandler(); class B { public void Event() { Console.WriteLine("1"); } } class Program { static void Main(string[] args) { </pre>

	<pre> A a = new A(); B b = new B(); a.OnSomething += b.Event; } } } </pre>
25.	<p>Что из перечисленного может быть назначено обработчиком события?</p> <ol style="list-style-type: none"> 1. Автоматически реализуемые свойства 2. Делегат (объект) 3. Любая функция 4. Статический метод с сигнатурой делегата, от которого он образован
26.	<p>- Какие параметры может иметь функция, совместимая с делегатом для событий EventHandler? Выберите все возможные ответы</p> <ol style="list-style-type: none"> 1) (object sender, EventArgs args) 2) (int sender, EventArgs args) 3) (int sender, myEventArgs args), где MyEventArgs - производный класс от EventArgs
27.	<p>Верно ли данное утверждение: События представляют собой специальный вид многоадресного делегата, который можно вызвать только из класса или структуры, в которых он объявлен.</p> <ol style="list-style-type: none"> 1) Да 2) Нет
28.	<p>Для чего нужны аксессоры событий?</p> <ol style="list-style-type: none"> a) не нужны b) для инициализации обработчиков событий c) для управления списком обработчиков d) для слежения за обработчиками событий, при групповой адресации
29.	<p>Что будет выведено в результате выполнения данного кода (***) - ошибка компиляции)</p> <pre> using System; namespace Application { delegate void Del(); public class A { public event Del ev; public void Creator() { ev(); } } class MainClass { public static void Pr() { Console.WriteLine("Hi"); } public static void Main(string[] args) { A a = new A(); a.ev += Pr; a.Creator(); } } } </pre>
30.	-
31.	<p>Что делает данная программа?</p> <pre> public int counter = 0; </pre>

	<pre> public Form1() { InitializeComponent(); this.MouseEnter = Form1_MouseEnter; } private void Form1_MouseEnter(object sender, EventArgs e) { this.ourLabel.Text = \$"Mouse entered {counter++} times"; } </pre> <p>Примечание: Если возникнет ошибка компиляции, введите: *** Если возникнет ошибка исполнения или исключение, введите: +++</p>
32.	<p>Выберете аксессоры событий:</p> <ol style="list-style-type: none"> 1) get 2) add 3) remove 4) set 5) addRange 6) update