Обратный вызов

	Вопросы
1.	Какую структуру можно передавать в качестве callback метода (функции)? (Передавать как аргумент, к примеру, методу Array.Sort)
2.	Выберите верные утверждения: 1) реализация callback'а возможна только с помощью делегата 2) callback это тип 3) callback это интерфейс 4) callback это функция

3.	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: +++
	
	static void PrintArray(int[] a)
	{
	Console.WriteLine();
	for (int i = 0; i < a.Length; i++)
	{
	Console.Write(a[i].ToString("") + " ");
	}
	}
	
	static void Main(string[] args)
	{
	int[] a = { 1, 3, -4, 7, 5 };
	Array.ForEach(a, x => { x = 5 * x;});
	PrintArray(a);
	}

4.	В результате выполнения следующего фрагмента программы будет выведено:_____
	
	namespace Test
	{
	public delegate int ConvertRule(int numb);
	class Converter
	{
	public int Convert(int numb , ConvertRule Conv)
	{
	return Conv(numb);
	}
	}
	class Program
	{
	public static double Rule(int numb)
	{
	if (numb % 3 == 0) return numb / 3;
	else return numb - (numb % 3);
	}
	static void Main(string[] args)
	{
	ConvertRule crMethod = Rule;
	Converter testConverter = new Converter();
	Console.WriteLine(testConverter.Convert(5, Rule));
	}
	}
	}
	
	Примечание:
	Если возникнет ошибка компиляции, введите: ***

	Если ошибок и исключений нет, но на экран не выведется ничего, введите: --- Если возникнет ошибка исполнения или исключение, введите: +++

5.	Что будет выведено на экран после выполнения данного кода? (*** - ошибка компиляции)
	using System;
	namespace test2
	{
	delegate int A(int x);
	public class MainClass
	{
	public static int x(int x)
	{
	return (int)Math.Pow(x, 5);
	}
	public static int y(int x, A a)
	{
	return a(x);
	}
	public static void Main(string[] args)
	{
	int p = 2;
	A a = new A(x);
	Console.WriteLine(y(p,a));
	}
	}
	}

6.	Отметьте верные утверждения: 1. Все функции DLL, вызываемые в вызове неуправляемого кода, требуют для своего выполнения наличия в управляемом коде функции обратного вызова. 2. Метод Array.Sort требует обязательного наличия функции обратного вызова среди переданных ему аргументов. 3. Метод Array.ConvertAll требует обязательного наличия функции обратного вызова среди переданных ему аргументов. 4. Вызов неуправляемого кода автоматически преобразует экземпляр делегата в знакомый формат обратного вызова.

7.	Что выведет данный фрагмент кода?
	public static void Main()
	{
	
	PointF[] apf = {
	new PointF(27.8F, 32.62F),
	new PointF(99.3F, 147.273F),
	new PointF(7.5F, 1412.2F) }; //system.drawings
	
	Point[] ap = Array.ConvertAll(apf,
	new Converter<PointF, Point>(PointFToPoint));
	
	foreach(Point p in ap)
	{
	Console.WriteLine(p.X + " " + p.Y);
	}
	}
	
	public static Point PointFToPoint(PointF pf)
	{
	return new Point(((int) pf.X), ((int) pf.Y));
	}
	
	Примечание:
	Если возникнет ошибка компиляции, введите: ***
	Если на экран не выведется ничего, введите: ---
	Если возникнет ошибка исполнения или исключение, введите: +++

8.	Вызов обработчиков события является частью самого события? 1 да 2 нет
9.	Что обычно возвращают функции обратного вызова сообщая о сбое? 1)1 2)-1 3)0 4)false 5)null
10.	Выберите верные утверждения для Callback-a: 1)Обратный вызов позволяет в функции исполнять код, который задаётся в аргументах при её вызове. 2)Этот код может быть определён в других контекстах программного кода и быть недоступным для прямого вызова из этой функции. 3)Этот НЕ код может быть определён в других контекстах программного кода и быть недоступным для прямого вызова из этой функции.
11.	Выбрать методы статики 1. Array.Sort() 2. Array.ConvertAll() 3. Array.ForEach() 4. Array.IndexOf() 5. Array.Find()

12.	-- Что выведет программа? ---
	
	using System;
	
	namespace Preparation
	{
	delegate string Delegate(int a);
	
	class Program
	{
	delegate double Delegate(int a);
	
	static void Main(string[] args)
	{
	Delegate del = (x) => x;
	var val = del(1);
	Console.WriteLine(val.GetType());
	}
	}
	}
	
	1) System.Double
	2) System.String
	3) Ничего - ошибка компиляции

13.	Делегат включает в качестве полей ссылку на объект, для которого нужно вызвать метод, и имя конкретного метода. Если ссылка на объект равна null, то… 1) Возникнет ошибка компиляции 2) Возникнет ошибка исполнения или исключение 3) Имя метода воспринимается как имя статического метода 4) Имя метода воспринимается как имя класса 5) Имя метода доступно по ссылке this

14.	using System;
	
	namespace ConsoleApp
	{
	public delegate int MyComp(int a, int b);
	
	class Program

	{
	static void Main()
	{
	MyComp comp = delegate (int a, int b) { return a > b ? 1 : -1; };
	int[] arr = new int[] { 5, 4, 3, 4, 1 };
	Array.Sort(arr, comp);
	foreach(var memb in arr) Console.Write(memb);
	}
	}
	}
	
	Что выведет программа?

15.	Что такое CallBack?
16.	Если делегат с многоадресной передачей возвращает какое либо значение (а не void), то вызывающий код: 1) принимает только значение, возвращенное последним методом в цепочке вызовов 2) принимает значение только первого метода в цепочке вызовов 3) такой делегат обязательно должен быть типа void 4) произойдет ошибка компиляции
17.	Делегат и событие отличаются тем, что: 1) делегат - тип, событие - интерфейс 2) делегат может быть только одноадресным, а событие - нет 3) прослушивание событий необязательно, в отличии от делегатов 4) возвращаемыми значениями
18.	delegate void EventHandler(object sender,EventArgs e) позволяет: 1)передавать ограниченное кол-во информации через sender 2)передавать ограниченное кол-во информации через EventArgs 3)передавать любое кол-во информации через наследник sender 4)передавать любое кол-во информации через наследник EventArgs 5)указывать объект, запустивший событие
19.	Что делает ключевое слово "static" примененное к событию: 1) Делает событие доступным для вызывающих объектов в любое время, даже если экземпляр класса не существует. 2) Указывает, что для производных классов оно больше не является виртуальным. 3) Запрещает вызывать событие без создания экземпляра класса. 4) Делает класс статическим.
20.	Какая запись синтаксически верная? 1) Array.Sort(data, SortKey()); 2) Array.Sort(data, SortKey); Примечание: data - инициализированный ранее массив SortKey - объявленый ранее метод

21.	Что выведет следующая программа?
	
	using System;
	using System.Collections.Generic;
	using System.Linq;
	using System.Text;
	using System.Threading.Tasks;
	delegate void CallMethod(string s);
	
	class Core
	{
	public static void CoreMethod(CallMethod callBackFunction)
	{
	Console.Write("meow");
	callBackFunction("hello");
	}
	}

	class Level1
	{
	public static void method(string s)
	{
	Console.WriteLine(s);
	}
	
	static void Main(string[] args)
	{
	CallMethod Method1 = Level1.method;
	Core.CoreMethod(Method1);
	Console.ReadKey();
	}
	}

22.	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: +++
	
	static void Main(string[] args)
	{
	int[] a = { 1, 3, -4, 7, 5 };
	Array.ForEach(a, x => { x = 5 * x; Console.Write(x.ToString("") + " "); });
	}

23.	В результате выполнения следующего фрагмента программы будет выведено:_____
	
	namespace Task
	{
	class Program
	{
	static void Main(string[] args)
	{
	int[] ar = { 0, -5, 4, -4, 3, 2, -7};
	Array.Sort(ar,
	(x, y) => {
	if (x < y) return 1;
	return -1;
	}
	);
	
	ar = Array.ConvertAll(ar,
	x =>
	{
	if (x % 2 == 0)
	return x + 1;
	
	return x * 2;
	});
	Array.ForEach(ar, Console.Write);
	Console.ReadLine();
	}
	}
	}
	
	Примечание:
	Если возникнет ошибка компиляции, введите: ***
	Если ошибок и исключений нет, но на экран не выведется ничего, введите: ---
	Если возникнет ошибка исполнения или исключение, введите: +++

24.	Что будет выведено на экран после выполнения данного кода? (*** - ошибка компиляции)
	using System;
	namespace test2
	{
	public class MainClass
	{
	public static void Main(string[] args)
	{
	int k = 0;
	double[] x = new double[5];
	Array.ConvertAll(x, y => { return k++; });
	Array.ForEach(x, y=> {Console.Write(y); });
	}
	}
	}

25.	Что будет выведено на экран в результате выполнения данной программы? *** - ошибка компиляции. +++ - ошибка выполнения.
	
	static void Main(string[] args)
	{
	int[] arr = new int[] { 1, 2, 3 };
	Array.ForEach(arr, x => Console.Write(++x));
	Array.ForEach(arr, x => Console.Write(x));
	}

26.	Что выведет данный фрагмент кода?
	double [] arr = new double [] {1.23, 1F, 2.45};
	Array.Sort(arr, (double x, double y) =>
	{
	if (x<y) return -1;
	if (x==y) return 0;
	return 1;
	});
	Примечание:
	Если возникнет ошибка компиляции, введите: ***
	Если на экран не выведется ничего, введите: ---
	Если возникнет ошибка исполнения или исключение, введите: +++

27.	Что будет выведено после выполнения данного кода?
	
	public static void DelegateMethod(string message)
	{
	System.Console.WriteLine(message);
	}
	
	Del handler = DelegateMethod;
	
	public void MethodWithCallback(int param1, int param2, Del callback)
	{
	callback("The number is: " + (param1 + param2).ToString());
	}
	
	MethodWithCallback(1, 2, handler);

28.	Что является признаком необходимости обратного вызова для функции ? 1) lpEnumFunc 2)EnumWindows 3)WNDENUMPROC 4)LPARAM 5)lParam
29.	Что выведет данный код: ... public void MethodWithCallback(int param1, int param2, Del callback)

	{
	callback("The number is: " + (param1 + param2).ToString());
	}
	static void Main()
	{
	...
	MethodWithCallback(1, 2, handler);
	...
	}

30.	
	using System;
	class MainClass
	{
	public static void Main(string[] args)
	{
	double[] mynumbers = { 2.55, 3.66, 10.100 };
	Array.ConvertAll(mynumbers, new Converter<double, int>((double input) =>
	Convert.ToInt32(input * 2)));
	foreach (var a in mynumbers)
	Console.Write("{0} ", a);
	}
	}
	
	на экран будет выведено:
	Примечание:
	Если возникнет ошибка компиляции, введите: ***
	Если ошибок и исключений нет, но на экран не выведется ничего, введите: ---
	Если возникнет ошибка исполнения или исключение, введите: +++

31.	-- Что выведет программа? ---
	
	using System;
	
	namespace Preparation
	{
	class Program
	{
	delegate int Function(int a, Function f);
	
	static void Main(string[] args)
	{
	Function func = (a, f) => a;
	int res = func(10, func);
	Console.WriteLine(res);
	}
	}
	}
	1) Ничего - ошибка Runtime
	2) 10
	3) Ничего - ошибка компиляции

32.	Что выведет следующая программа?
	public delegate void Deleg(string result);
	
	public static void DoSmth(Deleg callback)
	{
	callback("Hello!");
	}
	
	public static void Testing(string result)
	{
	Console.WriteLine(result);
	}

	public static void Test()
	{
	Deleg callback = Testing;
	DoSmth(callback);
	}
	
	static void Main(string[] args)
	{
	Test();
	Console.ReadKey();
	}

33.	using System;
	
	namespace ConsoleApp
	{
	class Program
	{
	static void Main()
	{
	Console.Write(SomeMethod(delegate (int a, int b) { Console.Write(a + b); }));
	}
	
	static public int SomeMethod(Action<object, object> action = null, Predicate<int> restriction = null)
	{
	int a = 1;
	if (restriction != null || restriction(a))
	action(a, a);
	return a;
	}
	}
	}
	
	Что выведет программа?

34.	
35.	Отметьте верные суждения: 1) делегат не может принимать и возвращать разные по типу значения 2) делегаты нельзя складывать или вычитать 3) экземпляры делегатов являются типизированными и ссылаются на один или более методов 4) каждый параметр лямбда выражения соотносится с параметром делегата, а тип выражения соотносится с типом, возвращаемым делегатом 5) для анонимных методов нельзя полностью опустить объявление параметров, если их ожидает делегат

36.	Задан аргумент события Args, выберите номера строк, которые неверны:
	
	/*1*/public class FileSearcher
	{
	/*2*/ public event EventHandler<Args> event;
	
	/*3*/ public void Method()
	{
	/*4*/ for(int i = 0; i < 10; i++)
	{
	/*5*/ event?.Invoke(this, new Args(i));
	}
	}
	}

37.	По умолчанию EventArgs делегата EventHandler передаёт: 1)объект, вызвавший событие 2)тип объекта, вызвавший событие 3)ссылку на объект 4)ничего 5)метку на точку вызова события
38.	Стандартной сигнатурой делегата события .NET является: 1) delegate void EventName(object sender, EventArgs args); 2) internal delegate void EventName(object sender, EventArgs args); 3) void EventName(object sender, EventArgs args); 4) EventName(object sender, EventArgs args); 5) void EventName;