**Вопрос 1** **Для чего используется интерфейс System.Runtime.Serialization.ISerializable**

нет такого интерфейса

в него преобразуется атрибут SerializableAttribute

для создания пользовательского формата сериализации

для сериализаци объектов System.Runtime.Serialization.

Formatters.Binary.BinaryFormatter и System.Xml.Serialization.XmlSerializer

только для сериализации объектов из BCL

для создания объекта SerializationInfo

**Ответ:** для создания пользовательского формата сериализации

**Вопрос 2** **Какой из следующих вариантов вызова функции WinAPI GetLastError в C# является наиболее корректным**

[System.Runtime.InteropServices.DllImport("Kernel32.dll", SetLastError=true)]

int GetLastError();

int error = GetLastError();

[System.Runtime.InteropServices.DllImport("Kernel32.dll", SetLastError=true)]

static extern int GetLastError();

int error = etLastError();

int error = GetLastError();

int error = Marshal.GetLastWin32Error();

uint error = Marshal.GetLastWin32Error();

[System.Runtime.InteropServices.DllImport("Kernel32.dll")]

uint GetLastError();

uint error = GetLastError();

**Ответ:** [System.Runtime.InteropServices.DllImport("Kernel32.dll", SetLastError=true)]  
int GetLastError();

**Вопрос 3** **При разработке собственного класса атрибута, сколько раз один атрибут можно использовать для сущности?**

1 раз

это недопустимое понятие

неограниченное к-во раз

если свойство AllowMultiple установлено в true, то неограниченное к-во раз, в противном случае - 1 раз

до 10 раз

2 раза

**Ответ:** если свойство AllowMultiple установлено в true, то неограниченное к-во раз, в противном случае - 1 раз

**Вопрос 4** **Что произойдет, если для атрибута DllImportAttribute свойство SetLastError будет установлено в true**

в случае, если точка входа в неуправляемую динамическую библиотеку не найдена, вызовет функцию SetLastError с соответствующим кодом ошибки

в случае, если точка входа в неуправляемую динамическую библиотеку не будет найдена, вызовет FileNotFoundException

вызываемый объект вызовет функцию SetLastError интерфейса Win32 API перед возвращением из метода, использующего атрибуты

вызываемый объект вызовет функцию SetLastError интерфейса Win32 API перед возвращением из метода, использующего атрибуты, а CLR вызовет исключение, в зависимости от того, что вернет функция GetLastError

в случае ошибки, вызываемый объект вызовет функцию интерфейса Win32 API SetLastError

в случае, если неуправляемая динамическая библиотека не будет найдена, вызовет FileNotFoundException

**Ответ:** вызываемый объект вызовет функцию SetLastError интерфейса Win32 API перед возвращением из метода, использующего атрибуты, а CLR вызовет исключение, в зависимости от того, что вернет функция GetLastError

**Вопрос 5** **Какие форматы сериализации в С# вы знаете**

soap

xpath

xhtml

coap

coap

двоичный

**Ответ:** soap   
двоичный

**Вопрос 6** **В какой неуправляемый тип можно привести строки типа StringBuilder для неуправляемого кода**

LPTStr

в любой из перечисленных

LPStr

LPWStr

AnsiBStr

BStr

**Ответ:** LPStr

**Вопрос 7** **Какое соглашение используется, чтобы отличать атрибуты от других типов .NET Framework**

для них применяется атрибут AttributeUsage

все имена атрибутов заканчиваются словом "AttributeValue"

все имена атрибутов заканчиваются словом "Attribute"

все имена атрибутов заканчиваются словом "AttributeType"

они реализуют интерфейс IAttribute

для них применяется атрибут AttributeUsageAttribute

**Ответ:** все имена атрибутов заканчиваются словом "Attribute"

**Вопрос 8** **Что произойдет в результате выполнения данного кода на C#, если он не содержит синтаксическую ошибку.**

**Если он содержит ошибку, то какую?**

**class FullName{**

**string name, secondName;**

**public FullName(string name, string secondName){**

**this.name = name;**

**this.secondName = secondName;**

**}**

**public string Name{get{return name;}}**

**public string SecondName{get{return secondName;}}**

**public override string ToString(){**

**return name + " " + secondName;**

**}**

**}**

**[Serializable]**

**class User{**

**FullName fullName;**

**string password, email;**

**public User(FullName fullName, string password, string email){**

**this.fullName = fullName;**

**this.password = password;**

**this.email = email;**

**}**

**public string Password{get{return password;}}**

**public string EMail{get{return email;}}**

**public FullName FullName{get{return fullName;}}**

**}**

**class Program {**

**static void Main()**

**{**

**var ms = new System.IO.MemoryStream();**

**try**

**{ User user = new User(**

**new FullName("Michael", "Jordan"),**

**"jordansfeint", "jordan@mail.com");**

**var fmtr = new System.Runtime.Serialization.**

**Formatters.Binary.BinaryFormatter();**

**fmtr.Serialize(ms, user);**

**ms.Seek(0, System.IO.SeekOrigin.Begin)**

**user = null;**

**user = (User)fmtr.Deserialize(ms);**

**Console.WriteLine(user.FullName);**

**}catch(Exception z){**

**Console.WriteLine(z.Message);**

**}finally{**

**ms.Close();**

**}**

**}**

**}**

Michael Jordan

MichaelJordan

MichaelJordan jordan@mail.com

InvalidOperationException

OutOfMemoryException

SerializationException

**Ответ:** SerializationException

**Вопрос 9** **Выберите возможные причины переименования функции DLL с помощью DllImport.EntryPoint**

привести имена в соответствие с существующими стандартами именования

привести имена в соответствие с существующими стандартами именования

использовать функцию из неуправляемого кода, что применяет кодировку отличающуюся от кодировки приложения

избежать использования имен API-функций, чувствительных к регистру знаков

упростить применение API-интерфейсов, которые содержат функции версий для ANSI и Юникода

сделать возможным вызов функций, принимающих данные разных типов (объявляя несколько версий одной и той же функции DLL)

**Ответ:** привести имена в соответствие с существующими стандартами именования[[1]](#footnote-1)   
использовать функцию из неуправляемого кода, что применяет кодировку отличающуюся от кодировки приложения   
избежать использования имен API-функций, чувствительных к регистру знаков   
упростить применение API-интерфейсов, которые содержат функции версий для ANSI и Юникода   
сделать возможным вызов функций, принимающих данные разных типов (объявляя несколько версий одной и той же функции DLL)

**Вопрос 10** **Для чего используется DllImportAttribute**

для подключения настроек приложения

для подключения новых возможностей языка

для доступа к методу, что относится к неуправляемому API

для получения доступа к настройкам операционной системы

для подключения динамической библиотеки линковки

для загрузки другой локализации приложения

**Ответ:** для доступа к методу, что относится к неуправляемому API

**Вопрос 11** **Что произойдет в результате выполнения данного кода на C#, если он не содержит синтаксическую ошибку.**

**Если он содержит ошибку, то какую?**

**[Serializable]**

**class A {**

**public B b;**

**public string data = "dataA";**

**}**

**[Serializable]**

**class B**

**{**

**public A a;**

**public string data = "dataB";**

**}**

**class Program {**

**static void Main()**

**{**

**var ms = new System.IO.MemoryStream();**

**try**

**{**

**A user1 = new A();**

**B user2 = new B();**

**user1.b = user2;**

**user2.a = user1;**

**Console.WriteLine(user1.b == user2);**

**var fmtr = new System.Runtime.Serialization.**

**Formatters.Binary.BinaryFormatter();**

**fmtr.Serialize(ms, user1);**

**fmtr.Serialize(ms, user2);**

**ms.Seek(0, System.IO.SeekOrigin.Begin);**

**user1 = null;**

**user2 = null;**

**user1 = (A)fmtr.Deserialize(ms);**

**user2 = (B)fmtr.Deserialize(ms);**

**Console.WriteLine(user1.b == user2);**

**}**

**catch (Exception e)**

**{**

**Console.WriteLine(e.Message);**

**}**

**finally {**

**ms.Close();**

**}**

**}**

**}**

True  
False

SerializationException

IOException

True  
True

False  
False

NullReferenceExeption

**Ответ:** True  
False

**Вопрос 12** **Если вы создаете свой атрибут, как указать, что он должен применяться только для класса и интерфейса**

указать 1м параметром для атрубута AttributeUsage значение

AttributeTargets.Interface | AttributeTargets.Class

указать 2м параметром для атрубута AttributeUsage значение

new AttributeTargets[]{ AttributeTargets.Interface, AttributeTargets.Class }

среди вышеперечисленных нет правильного ответа

нельзя это указать

перегрузить св-ва класса Attribute bool IsForClassUsed{get;} и bool IsForInterfaceUsed{get;}

указать первым параметром для атрубута AttributeUsage значение

new AttributeTargets[]{ AttributeTargets.Interface, AttributeTargets.Class }

**Ответ:** указать 1м параметром для атрубута AttributeUsage значение AttributeTargets.Interface | AttributeTargets.Class

**Вопрос 13** **Как правильно написать свойство, что возвращает маршаллируемый тип**

вариант 4

decimal \_money;

[MarshalAs(UnmanagedType.Currency)]

public decimal Money

{

get { return this.\_money; }

}

вариант 3

decimal \_money;

public decimal Money

{

[returns: MarshalAs(UnmanagedType.Currency)]

get { return this.\_money; }

}

вариант 5

decimal \_money;

[return MarshalAs(UnmanagedType.Currency)]

public decimal Money

{

get { return this.\_money; }

}

вариант 2

decimal \_money;

public decimal Money

{

[return: MarshalAs(UnmanagedType.Currency)]

get { return this.\_money; }

}

вариант 1

decimal \_money;

public decimal Money

{

[return MarshalAs(UnmanagedType.Currency)]

get { return this.\_money; }

}

вариант 6

decimal \_money;

[returns MarshalAs(UnmanagedType.Currency)]

public decimal Money

{

get { return this.\_money; }

}

**Ответ:** вариант 2  
decimal \_money;  
  
public decimal Money  
{  
 [return: MarshalAs(UnmanagedType.Currency)]  
 get { return this.\_money; }  
}

**Вопрос 14** **Выберите верные названия атрибутов, что входят в BCL**

Combine, CombineAttribute

DescriptionAttribute, DllImport

AttributeAttribute, DllImportAttribute

Attribute, Description

Combine, MemoryAttribute

DrawAttribute, Memory

**Ответ:** DescriptionAttribute, DllImport   
Attribute, Description

**Вопрос 15** **Какие классы BCL для сериализации вы знаете**

BinaryFormatter

XmlFormatter

ObjectSerializer

XmlSerializer

ObjectFormatter

SoapFormatter

**Ответ:** BinaryFormatter   
XmlSerializer   
SoapFormatter

**Вопрос 16** **Для чего используется поле ThrowOnUnmappableChar атрибута DllImport**

включает или отключает поведение наилучшего сопоставления при преобразовании знаков Юникода в знаки ANSI

включает и отключает возникновение исключений для неотображаемых символов Юникода, преобразующихся в знаки вопроса ("?") ANSI

включает или отключает проверку, на эквивалентность значения параметра, предаваемого неуправляемой функции, значению, что находится в функциии вызывает исключение, емли значения не эквивалентны

показывает, вызывает ли вызываемый объект функцию SetLastError интерфейса Win32 API перед возвращением из метода, использующего атрибуты

включает и отключает вызов исключения, если динамическую библиотеку с неуправляемым кодом с указанным именем не удалось найти

включает или отключает возникновение исключений, если строку нельзя преобразовать в int

**Ответ:** включает или отключает поведение наилучшего сопоставления при преобразовании знаков Юникода в знаки ANSI

**Вопрос 17** **Выберите все верные утверждения**

Метаданные хранят данные, которые программа может взять из ресурсов

Метаданные сдержатся в сборке .NET, если программист определил атрибуты для классов

К типу можно определить только один атрибут

Для задания необходимых дополнительных сведений о типах можно добавить атрибуты

Атрибуты могут принимать аргументы точно так же, как методы и свойства

Атрибуты добавляют в программу метаданные

**Ответ:** Метаданные хранят данные, которые программа может взять из ресурсов   
Метаданные сдержатся в сборке .NET, если программист определил атрибуты для классов   
Для задания необходимых дополнительных сведений о типах можно добавить атрибуты   
Атрибуты могут принимать аргументы точно так же, как методы и свойства   
Атрибуты добавляют в программу метаданные

**Вопрос 18** **Какой вариант указанной функции будет вызван**

**[DllImport("user32.dll", CharSet=CharSet.Auto)]**

**public static extern int MessageBox(int hWnd, String text,**

**String caption, uint type);**

эта функция не зависит от кодировки

в кодировке windows-1251

зависит от кодировки в сборке

в кодировке Unicode

в кодировке ANSI

windows-1252

**Ответ:** в кодировке ANSI

**Вопрос 19** **Для чего используется поле EntryPoint атрибута DllImport**

для указания имени динамической библиотеки

для указания местонахождения функции main

для задания функции в динамической библиотеке

для задания имени или порядкового номера точки входа динамической библиотеки

для указания местонахождения функции WinMain в сборке

для переименования функций

**Ответ:** для задания имени или порядкового номера точки входа динамической библиотеки

**Вопрос 20** **Для чего необходимы атрибуты**

для связывания метаданных или декларативной информации с кодом (сборками, типами, методами, свойствами и т.д.)

для создания отдельной сборки, например динамической библиотеки

для связывания типов в сборке

для описания метаданных в программе

для создания типов, которые будут работать на различных операционных системах

для получения информации о метаданных

**Ответ:** для связывания метаданных или декларативной информации с кодом (сборками, типами, методами, свойствами и т.д.)

**Вопрос 21** **В какой тип по-умолчанию преобразуются строки C# для методов COM**

BStr

LPWStr

LPStr

ни один из названных

LPTStr

AnsiBStr

**Ответ:** LPWStr

**Вопрос 22** **Для какого типа (типов) данных нельзя использовать StringBuilder как тип поля, если его объект (этого типа) передается в неуправляемую функцию**

шаблонные типы, где StringBuilder выступает типом поля класс

struct

не один из вышеперечиленных

class

шаблонные типы, где StringBuilder выступает параметром типа

interface

**Ответ:** шаблонные типы, где StringBuilder выступает типом поля класс

**Вопрос 23** **Для чего используется контекстно зависимое ключевое слово param (не путать с params)**

нет такого слова в C#

для передачи одного параметра, как маршаллируемого типа, а не нескольких, в отличие от params

для указания, что тип является маршаллируемым

для получения длины массива, если он передается из метода неуправляемого API

для указания, что тип является неуправляемым

для указания, что маршаллируемый тип используется как принимаемый параметр

**Ответ:** нет такого слова в C#

**Вопрос 24** **Для чего используется атрибут NonSerialized**

для запрета сериализации методов

для запрета сериализации классов

для получения информации, какое поле не былы сериадизовано

для запрета сериализации структур

для получения информации, какой метод не был сериадизован

для запрета сериализации полей

**Ответ:** для запрета сериализации полей

**Вопрос 25** **Для чего используется MarshalAsAttribute**

для управления освобождением памятью

для преобразования типа из управляемого в неуправляемый

задает уникальный идентификатор для неуправляемого типа

для указания того, как выполняется маршалинг данных между управляемой и неуправляемой памятью

позволяет задать тип неуправляемого объекта

позволяет задать тип управляемого объекта

**Ответ:** для указания того, как выполняется маршалинг данных между управляемой и неуправляемой памятью

**Вопрос 26** **Для чего используется атрибут Obsolete?**

Атрибутом Obsolete помечается тип, имя которого встречается в других пространствах имен

Атрибутом Obsolete помечается нерекомендуемая программа

Атрибутом Obsolete помечается класс с которого начинает исполняться приложение

Атрибутом Obsolete помечается нерекомендуемая программа

Атрибутом Obsolete помечается класс, который спроектирован с нарушением рекомендаций по разработке ПО

Атрибутом Obsolete помечается нерекомендуемая динамическая библиотека

**Ответ:** Атрибутом Obsolete помечается нерекомендуемая программа[[2]](#footnote-2)

**Вопрос 27** **Выберите все правильные модификаторы для создания прототипа неуправляемой функции в управляемом коде**

internal extern

static extern

public static extern

private extern

extern

public extern

**Ответ:** static extern   
extern

**Вопрос 28** **Что произойдет в результате выполнения данного кода на C#, если он не содержит синтаксическую ошибку.**

**Если он содержит ошибку, то какую?**

**[Serializable]**

**class User**

**{**

**string login, password;**

**public User(string login, string password)**

**{**

**this.login = login;**

**this.password = password;**

**}**

**public string Password { get { return password; } }**

**public string Login { get { return login; } }**

**public Storage Storage { get; set; }**

**}**

**[Serializable]**

**class Storage**

**{**

**Dictionary<User, string> items = new Dictionary<User, string>();**

**public string this[User user] { get { return items[user]; } }**

**public void Add(User u, string cashe)**

**{**

**items.Add(u, cashe);**

**}**

**public int Count { get { return items.Count; } }**

**public override string ToString()**

**{**

**StringBuilder sb = new StringBuilder(30);**

**foreach (var item in items)**

**{**

**var k = item.Key;**

**sb.Append(k.Login).Append(" ").Append(k.Password);**

**}**

**return sb.ToString();**

**}**

**} class Program {**

**static void Main()**

**{**

**Storage st = new Storage();**

**User u1 = new User("login1","password1");**

**User u2 = new User("login2","password2");**

**u1.Storage = st;**

**u2.Storage = st;**

**st.Add(u1, "1234567123456789");**

**st.Add(u2, "abcdefghij");**

**Console.WriteLine(st[u2]);**

**Console.WriteLine();**

**var s = new System.IO.FileStream("data", System.IO.FileMode.Create);**

**var fmtr = new System.Runtime.Serialization.**

**Formatters.Binary.BinaryFormatter();**

**fmtr.Serialize(s, st);**

**s.Close();**

**st = null;**

**s = new System.IO.FileStream("data", System.IO.FileMode.Open);**

**st = (Storage)fmtr.Deserialize(s);**

**s.Close();**

**Console.WriteLine(st[u2]);**

**Console.WriteLine();**

**}**

SerializationException

abcdefghij  
abcdefghij

KeyNotFoundException

123456789  
abcdefghij

IOException

NullReferenceException

**Ответ:** abcdefghij  
abcdefghij[[3]](#footnote-3)

**Вопрос 29** **Что произойдет в результате выполнения данного кода на C#, если он не содержит синтаксическую ошибку.**

**Если он содержит ошибку, то какую?**

**[Serializable]**

**class User**

**{**

**[NonSerialized]**

**string password;**

**string login;**

**public User(string password, string login)**

**{**

**this.password = password;**

**this.login = login;**

**}**

**public string Password { get { return password; } }**

**public string Login { get { return login; } }**

**}**

**class Program {**

**static void Main()**

**{**

**var ms = new System.IO.MemoryStream();**

**try**

**{**

**User user = new User("Michael", "Jordan");**

**var fmtr = new System.Runtime.Serialization.**

**Formatters.Binary.BinaryFormatter();**

**fmtr.Serialize(ms, user);**

**ms.Seek(0, System.IO.SeekOrigin.Begin);**

**user = null;**

**user = (User)fmtr.Deserialize(ms);**

**Console.WriteLine(user.Password + " " + user.Login);**

**}catch(Exception z){**

**Console.WriteLine(z);**

**}finally{**

**ms.Close();**

**}**

**}**

**}**

Michael Jordan

Jordan

SerializationException

Michael

NullReferenceExeption

IOException

**Ответ:** Jordan

**Вопрос 30** **Выберите все верные утверждения**

Программа может проверить только собственные метаданные

аргументы атрибутов могут хранить только в типах BCL

метаданные представляют собой сведения о типах, определенных в программе

все сборки .NET содержат заданный набор метаданных, описывающих типы и члены типов, определенных в сборке

один или несколько атрибутов могут применяться к сборкам, модулям или более мелким программным элементам, таким как классы и свойства

Нельзя создавать собственные атрибуты, если программа предназначается для использования в нескольких операционных системах

**Ответ:** метаданные представляют собой сведения о типах, определенных в программе   
все сборки .NET содержат заданный набор метаданных, описывающих типы и члены типов, определенных в сборке   
один или несколько атрибутов могут применяться к сборкам, модулям или более мелким программным элементам, таким как классы и свойства

1. Выбрать одно из двух значений? (вероятнее всего **первое**) – привести имена в соответствие с существующими стандартами именования [↑](#footnote-ref-1)
2. Выбрать одно из двух значений? (вероятнее всего **первое**) – Атрибутом Obsolete помечается нерекомендуемая программа [↑](#footnote-ref-2)
3. На самом деле - abcdefghij [↑](#footnote-ref-3)