

01. Какой модификатор доступа применяется для классов по умолчанию?

- ☐ public
- ☐ private
- ☐ internal
- ☐ protected
- ☐ protected internal

02. Сколько прямых родителей может быть у структуры?

- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ Не ограничено

03. Где выделяется память для значимых типов:

- ☐ Всегда в управляемой куче
- ☐ Всегда на стеке
- ☐ На стеке или в управляемой куче, на усмотрение компилятора
- ☐ На стеке или в управляемой куче, в зависимости от реализации платформы
- ☐ На стеке или в управляемой куче, зависит от места объявления и боксинга

04. Поддержки каких операторов нет в C#?

- ☐ for
- ☐ foreach
- ☐ do while
- ☐ repeat until
- ☐ goto

05. Чем (из поддерживаемого в C#) может заканчиваться блок case оператора switch?

- ☐ throw
- ☐ goto
- ☐ goto case
- ☐ yield return
- ☐ бесконечным циклом

06. Выберите верные утверждения о статических конструкторах:

- ☐ Статический конструктор не может быть объявлен у значимых типов
- ☐ Статический конструктор для значимых типов не вызывается
- ☐ Статический конструктор всегда приватный
- ☐ Статический конструктор для значимых типов не может быть объявлен без параметров
- ☐ Исключение, выброшенное во время выполнения статического конструктора, приведёт к выбрасыванию `TypeInitializationException` в вызывающем коде

07. Выберите верные утверждения о ref/out:

- ☐ Переменная, передаваемая в качестве параметра, помеченного out, должна быть инициализирована
- ☐ Параметры, помеченные ref или out, всегда должны быть после остальных параметров
- ☐ Параметр, помеченный out, необходимо инициализировать до выхода из метода
- ☐ При вызове метода, необходимо указывать ref перед переменной, передаваемой в параметр, помеченный ref
- ☐ Нельзя создавать параметры помеченные ref или out в конструкторе экземпляра

08. Какое количество строк будет создано при выполнения следующего кода?

```
var s = "string".ToLower().Remove(1);
```

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5

09. Выберите верные утверждения о перечислениях (enum).

- ☐ Типы long, ulong, short, sbyte могут быть базовыми типами для перечисления
- ☐ Значение каждого перечислителя в перечислении необходимо задавать вручную
- ☐ Если указано значение одного из перечислителей в перечислении, то должны быть указаны значения остальных
- ☐ Значение первого перечислителя в перечислении равно 0;
- ☐ Атрибут FlagsAttribute влияет на результат вызова метода ToString();

10. Укажите операции в которых LinkedList асимптотически быстрее или равен List (амортизационную $O(1)$ считать равной $O(1)$)

- ☐ Добавление элемента в конец списка
- ☐ Добавление элемента в начало списка
- ☐ Удаление элемента из середины списка
- ☐ Удаление элемента с конца списка
- ☐ Взятие по элемента индексу

11. Выберите верные утверждения о делегатах:

- ☐ Делегат может содержать цепочку методов
- ☐ Делегаты являются неизменяемыми
- ☐ Делегат, указывающий на не статический метод класса, содержит ссылку на экземпляр этого класса
- ☐ Делегат - это указатель на метода, вместо него компилятор подставляет вызов через IntPtr
- ☐ Делегат нельзя создать из описателя метода (MethodInfo), полученного с помощью рефлексии

12. Отметьте правильные утверждения, показывающие разницу между методами `Object.Equals` и `Object.ReferenceEquals`.

- ☐ Метод `ReferenceEquals` может быть перегружен
- ☐ Метод `Equals` является статическим
- ☐ Метод `ReferenceEquals` может возвращать `false`, когда `Equals` возвращает `true` (реализации корректны)
- ☐ Метод `Equals` может возвращать `false`, когда `ReferenceEquals` возвращает `true` (реализации корректны)
- ☐ Реализация метода `Equals` по умолчанию, для ссылочных типов возвращает `true`, если это один и тот же объект в памяти

13. Отметьте правильные утверждения о финализаторах.

- ☐ Финализатор имеет детерминированное время вызова
- ☐ Финализатор вызывается в отдельном потоке
- ☐ Можно отменить вызов финализатора для выбранного объекта
- ☐ Финализатор можно объявить у значимого типов
- ☐ В финализаторе можно воскресить объект (поместить ссылку на него в живой объект)

14. Выберите способы приведения объекта `A` к другому типу, которые в случае неудачи не могут привести к выбросу исключения:

- ☐ `(string)A`
- ☐ `A as string`
- ☐ `A is string`
- ☐ `A as int`
- ☐ `A is int`

15. Какое исключение будет выброшено в результате выполнения следующего кода

```
object a = 5;  
string s = ((string) a).ToLower();
```

- ☐ InvalidOperationException
- ☐ ArgumentException
- ☐ InvalidCastException
- ☐ NullReferenceException
- ☐ Никакого

16. Укажите допустимые способы объявления свойств или индексов в классе:

- ☐ public int A { private get; set; }
- ☐ public int A { get { return 0; } }
- ☐ public int A { get; }
- ☐ public int this[int i, int j] { get { return A + i + j; } }
- ☐ public int A[int i] { get { return i; } }

17. Выберите корректные (без ошибок компиляции) способы создания переменной типа double:

- ☐ var a = 0.5;
- ☐ var a = 0;
- ☐ var a = 0.5f;
- ☐ double a = new int();
- ☐ var a = new double();

18. Выберите строки, в которых произойдёт ошибка компиляции:

```
public struct A
{
    public bool val;
    public A(int v) { val = v == 0; }
// #1
    public A(char a) : this() { }
// #2
    public A(double a) { this = new A(); }
// #3
    private A(bool a) { val = a; }
// #4
    protected A(long a) { val = a == 0; }
// #5
}
```

☐ // #1

☐ // #2

☐ // #3

☐ // #4

☐ // #5



19. Укажите операторы, которые могут быть перегружены в C#:

☐ ++

☐ !

☐ true false

☐ ??

☐ =

20. Укажите строки, в которых произойдут ошибки компиляции:

```
interface IA
{
    int M1 { get; }
    int M2 { set; }
    int M3 { get; set; }
    int M4();
}

class A : IA
{
    int IA.M1 { get { return 0; } }
    public int M2 { set { } }
    public int M3 { get { return 0; } set {
} }
    public int M4() { return 0; }
}

class Program
{
    public static void Main()
    {
        A a = new A();
        var m1 = a.M1;    // #1
        a.M2 = 0;         // #2
        var m3 = a.M3;    // #3
        var m4 = a.M4();  // #4
    }
}
```


21. Укажите строчки, при выполнении которых будет выброшено исключение:

```
object o = 1;  
long a1 = (int)o;           // #1  
long a2 = (long)o;          // #2  
long a3 = (long)(int)o;     // #3  
long a4 = (int)(long)(int)o; // #4
```

- ☐ // #1
- ☐ // #2
- ☐ // #3
- ☐ // #4
- ☐ Код корректно выполнится

22. Укажите строчки, в которых произойдут ошибки компиляции:

```
object[] a1 = new string[0];  
object[] a2 = new int[0];  
string[] a3 = new object[0];  
Action<string> a4 = new Action<object>  
(delegate { });  
Action<object> a5 = new Action<string>  
(delegate { });
```

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5

23. Какое условие необходимо выполнить, что бы произвольный класс мог быть использован в цикле foreach?

- ☐ должен реализовывать интерфейс IEnumerable
- ☐ должен реализовывать интерфейс ICollection
- ☐ должен реализовывать интерфейс IList
- ☐ Реализовать метод GetEnumerator() возвращающий объект IEnumerator
- ☐ Реализовать метод GetEnumerator() возвращающий объект содержащий публичный метод bool MoveNext() и публичное свойство Current;

24. Отметьте правильные утверждения о конструкции using:

- ☐ В using можно объявить несколько объектов одного типа
- ☐ В using нельзя передать уже созданный объект
- ☐ В using можно передать null, и он не приведёт к выбросу исключения
- ☐ В объявлении using можно не создавать переменную
- ☐ В using можно передать любой объект, содержащий публичный метод void Dispose()

25. Выберите верные утверждения об обобщениях:

- ☐ Обобщения позволяют сделать код более читаемым, уменьшив количество приведений типов
- ☐ Обобщения боксят значимые типы для обеспечения совместимости
- ☐ CLR не позволяет создать экземпляр открытого типа
- ☐ Значимые типы не могут содержать обобщённых методов
- ☐ Для всех ссылочных типов JIT компилятор объявляет один класс

26. Что выведет на консоль следующий код?

```
public class A<T>
{
    public static int Count { get; set; }
    public A() { A<T>.Count++; }
}
public static void Main()
{
    new A<string>();
    new A<object>();
    new A<string>();
    Console.WriteLine(A<string>.Count);
}
```

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ Код выдаст ошибку компиляции
- ☐ Код выдаст исключение

27. Укажите ограничения, которые достаточно добавить к методу M для его корректной компиляции:

```
public static T M() //where T ...  
{  
    return new T();  
}
```

- ☐ where T: int
- ☐ where T: struct
- ☐ where T: class, new
- ☐ where T: ICollection
- ☐ where T: struct, IList

28. Выберите верные утверждения о функции GetHashCode:

- ☐ Функция GetHashCode всегда возвращает уникальное значение для каждого объекта, в рамках одного типа;
- ☐ Функция GetHashCode необходима для использования объектов в качестве ключа для контейнеров реализующих алгоритм хеш таблицы;
- ☐ Функция GetHashCode реализована в типе object и есть у всех типов;
- ☐ При перегрузке функции GetHashCode для корректной работы так же должна быть перегружена функция Equals;
- ☐ Для ссылочных типов функция GetHashCode (если не перегружена) возвращает адресс объекта в памяти;

29. Выберите верные утверждения о методах расширения:

- ☐ Метод расширения должен быть объявлен в статическом классе;
- ☐ Перед первым параметром метода расширения должно быть ключевое слово `this`;
- ☐ Для использования метода расширения его пространство имён должно быть явно импортировано в исходный код;
- ☐ Метод расширения должен быть объявлен в публичном классе;
- ☐ Метод расширения не может быть реализован для структур;

30. Что нужно написать в блоке `catch (Exception ex)`, чтобы выбросить исключение `ex` сохранив оригинальный стек исключения?

- ☐ `throw`;
- ☐ `throw ex`;
- ☐ `throw new Exception(ex.Message, ex)`;
- ☐ `throw new ex`;
- ☐ Данная операция не поддерживается

31. Выберите верные утверждения о именованных и необязательных аргументах:

- ☐ Необязательные аргументы должны находиться строго после обязательных;
- ☐ Необязательные аргументы не могут быть использованы в индексаторах;
- ☐ Необязательные аргументы не могут быть использованы в делегатах;
- ☐ При вызове метода, значение именованного аргумента можно задать по его имени, без указания других необязательных аргументов, даже если они идут до него;
- ☐ В качестве значения для необязательных аргументов могут быть выбраны только константы и значения по умолчанию;

32. Какие механизмы условной компиляции предусмотрены в C#?

- ☐ Директива #endif
- ☐ Директива #elseif
- ☐ Атрибут Conditional
- ☐ Директива #define
- ☐ Такой механизм не предусмотрен

33. Что выведет на консоль следующий код?

```
[StructLayout(LayoutKind.Explicit)]
struct A
{
    [FieldOffset(0)]
    public int V1;

    [FieldOffset(0)]
    public int V2;
}

public static void Main()
{
    var a = new A();
    a.V1 = 1;
    Console.WriteLine(a.V2);
}
```

- ☐ 0
 - ☐ 1
 - ☐ -2147483648
 - ☐ Код выдаст ошибку компиляции
 - ☐ Код выдаст ошибку выполнения
-

34. Что выведет на консоль следующий код?

```
string x = "x";  
string a = "xx";  
string b = "xx";  
string c = x + x;  
Console.Write("{0} {1};", a == b,  
ReferenceEquals(a, b));  
Console.Write("{0} {1};", a == c,  
ReferenceEquals(a, c));
```

- ☐ True False; True False;
- ☐ True True; True False;
- ☐ True False; True True;
- ☐ True True; True True;
- ☐ True False; False False;

35. Выберите верные утверждения о событиях:

- ☐ С помощью переопределения методов доступа add и remove можно перехватить подписку на событие
- ☐ Событие может быть вызвано только в том классе, где оно объявлено
- ☐ Тип события должен быть EventHandler или EventHandler
- ☐ События представляют собой реализацию паттерна (шаблона проектирования) Посетитель (Visitor)
- ☐ Событие может быть объявлено как readonly

36. Что поможет избавиться от утечки памяти:

```
public class A
{
    public event Action Event;
    public void Handler() {}
}
public void Method( A global )
{
    var local = new A();
    global.Event += local.Handler;
    local.Event += global.Handler;
}
```

- ☐ Добавить отписку от global.Event
- ☐ Добавить отписку от local.Event
- ☐ Убрать подписку на global.Event
- ☐ Убрать подписку на local.Event
- ☐ В данном коде нет утечек памяти

37. Выберите конструкции, внутри которых нельзя использовать yield return:

- ☐ В try, если есть блок catch
- ☐ В try, если есть блок finally
- ☐ В блоке finally
- ☐ В блоке using
- ☐ В анонимных методах

38. Какой результат выведет данный код:

```
var arr = new[]{1, 2, 3};  
var list = new List<Func<int>>();  
foreach (var q in arr)  
    list.Add(() => q);  
  
var s = 0;  
foreach (var q in list)  
    s += q();  
Console.WriteLine(s);
```

- ☐ 6
- ☐ 9
- ☐ 9 в C# 4.0, 6 в C# 5.0
- ☐ 9 в .Net 4.0, 6 в .Net 4.5
- ☐ Ошибку времени выполнения

39. Какой результат выведет данный код:

```
var arr = new[] {1, 2, 3};  
var list = new List<Func<int>>();  
for (int i = 0; i < arr.Length; i++)  
    list.Add(() => arr[i]);  
  
foreach (var q in list)  
    Console.Write(q());
```

- ☐ 123
- ☐ 333
- ☐ 333 в C# 4.0, 123 в C# 5.0
- ☐ 333 в .Net 4.0, 123 в .Net 4.5
- ☐ Ошибку времени выполнения

40. Выберите наиболее быстрый способ скопировать одномерный массив double'ов в другой массив:

- ☐ Array.Copy
- ☐ Array.ConstrainedCopy
- ☐ Buffer.BlockCopy
- ☐ Поэлементно в цикле foreach
- ☐ Поэлементно в цикле for

41. Выберите наиболее быстрый способ обеспечить потокобезопасность при инициализации объекта:

- ☐ Использовать конструкцию lock
- ☐ Использовать класс AutoResetEvent
- ☐ Использовать класс Interlocked
- ☐ Использовать класс Mutex
- ☐ Использовать класс Monitor

42. Отметьте правильные утверждения о настраиваемых атрибутах:

- ☐ Атрибуты носят информационный характер и сами по себе никак не влияют на процесс компиляции и выполнения кода
- ☐ Атрибуты могут быть применены к сборке
- ☐ Атрибут нельзя применить к возвращаемому значению
- ☐ Атрибут не может быть значимым типом
- ☐ Если имя атрибута заканчивается на слово Attribute, то при использовании его можно не писать (вместо BAttribute писать B)

43. Отметьте правильные утверждения об анонимных типах:

- ☐ Анонимный тип может быть типом возвращаемого значения
- ☐ После объявления, свойства анонимного типа доступны только для чтения
- ☐ При объявлении анонимного типа всегда обязательно указывать имена всех его свойств
- ☐ Метод Equals для анонимных типов возвращает true только если это один и тот же объект
- ☐ Компилятор создаёт один анонимный тип для разных экземпляров, если они объявлены в передлах одной сборки, и содержат одинаковый набор свойств (имена, типы и последовательность объявления)

44. Какой тип данных наиболее подходящий для организации массива бит:

- ☐ List
- ☐ bool[]
- ☐ int
- ☐ BitVector32
- ☐ BitArray

45. Какие из приведённых коллекций позволяют производить операцию вставки за $O(\log N)$ и быстрее:

- ☐ Dictionary
- ☐ SortedDictionary
- ☐ SortedList
- ☐ List
- ☐ HashSet