**Вопрос 1** **Какой оператор нельзя использовать в принимаемых параметрах лямбды**

params

int

IComparable

[] (массив)

ref

string

**Ответ:** ref

**Вопрос 2** **Что произойдет в результате выполнения данного кода на C#, если он не содержит синтаксическую ошибку.**

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

**class ObservableList {**

**List<int> list = new List<int>();**

**EventHandler**

**public event EventHandler Changed;**

**public void Add(int value) {**

**list.Add(value);**

**OnChanged();**

**}**

**protected virtual void OnChanged()**

**{**

**Changed(this, EventArgs.Empty);**

**}**

**public int this[int index]**

**{ get { return list[index]; } }**

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

**}**

**class Program {**

**static void Main(){**

**ObservableList l = new ObservableList();**

**l.Add(1);**

**l.Add(2);**

**l.Add(3);**

**for (var i = 0; i < l.Count;++i)**

**Console.WriteLine(l[i]);**

**}**

**}**

321

пустая строка

123

NullReferenceException

InvalidOperationException

1

**Ответ:** NullReferenceException

**Вопрос 3** **Что произойдет в результате выполнения данного кода на C#, если он не содержит синтаксическую ошибку.**

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

**static class Extension{**

**static public void Say(this A inst, string name){**

**Console.WriteLine("Hello, {0}!", name);**

**}**

**}**

**class A{**

**public void Say(string name){**

**Console.WriteLine("Hi, {0}!", name);**

**}**

**}**

**public static void Main(){**

**A inst = new A();**

**inst.Say("Anny");**

**}**

нельзя перегружать метод Say, т.к. его вариант в классе Extension является статическим

Hello, Anny!

нельзя создать метод расширения для пользовательского типа данных

нельзя создать метод расширения с таким именем

нельзя создать метод расширения с такой сигнатурой

Hi, Anny!

**Ответ:** Hi, Anny!

**Вопрос 4** **Как установить для переменной шаблонного типа значение по-умолчанию**

написать метод, возвращающий значение по-умолчанию

T a = 0;

T a = default(T);

язык такое не поддерживает

T a = null;

добавить статическое поле, содержащее значение по-умолчанию

**Ответ:** T a = default(T)

**Вопрос 5** **Какой интерфейс достаточен для того, чтобы коллекция поддерживала цикл foreach**

IList

IEnumerator

IEnumerable

ICollection

IIterable

IComparable

**Ответ:** IEnumerable

**Вопрос 6** **Выберите все верные утверждения о лямбдах**

если лямбда имеет только один входной параметр, скобки можно не ставить, во всех остальных случаях они обязательны

переменная, объявленная в лямбде видима ниже в методе, где объявлена лямбда

иногда компилятору бывает трудно или даже невозможно вывести типы входных параметров. В этом случае типы можно указать в явном виде

отсутствие входных параметров задаётся пустыми скобками

лямбда выполняется сразу после ее объявления

в лямбде можно использовать оператор goto для перехода в другой метод

**Ответ:** если лямбда имеет только один входной параметр, скобки можно не ставить, во всех остальных случаях они обязательны   
иногда компилятору бывает трудно или даже невозможно вывести типы входных параметров. В этом случае типы можно указать в явном виде   
отсутствие входных параметров задаётся пустыми скобками

**Вопрос 7** **Какая коллекция BCL из перечисленных может хранить объекты любого типа и скорость доступа ко всем элементам которой одинакова**

ListDictionary

Dictionary<K,V>

HashMap<K,V>

ArrayList

StringDictionary

Vector

**Ответ:** ArrayList

**Вопрос 8** **Какие ключевые слова используются для определения пользовательского кода подписки и отписки от обработчика события**

clear

remove

add

set

insert

delete

**Ответ:** remove   
add

**Вопрос 9** **Что произойдет в результате выполнения данного кода на C#, если он не содержит синтаксическую ошибку.**

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

**struct Point{**

**public int x;**

**public int y;**

**}**

**class Program {**

**static void Main(){**

**Point p = new Point();**

**p.x = 5;**

**var list = new List<Point>(10);**

**list.Add(p);**

**list[0].x = 90;**

**Console.WriteLine(list[0].x);**

**}**

**}**

9

переменные, y не инициализирована

5

нельзя использовать Point, как параметр типа - List ссылочный тип, а Point тип значения

90

не удалось изменить возвращаемое значение

**Ответ:** не удалось изменить возвращаемое значение

**Вопрос 10** **Что произойдет в результате выполнения данного кода на C#, если он не содержит синтаксическую ошибку.**

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

**class Counter{**

**static int value;**

**static Counter(){ value = 1; }**

**public Action Show = delegate(){**

**Console.Write(value);**

**};**

**public Counter(){ ++value; }**

**}**

**public static void Main(){**

**Counter c = new Counter();**

**Counter c1 = new Counter();**

**Counter c2 = new Counter();**

**c.Show();**

**c1.Show();**

**c2.Show();**

**}**

111

234

123

444

222

неверный синтаксис вызова делегата

**Ответ:** 444

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

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

**class Program {**

**static bool IsInited<T>(T a) { return a == null; }**

**static void Main(){**

**Console.WriteLine(IsInited(0));**

**}**

**}**

true

нельзя сравнивать с null, т.к. тип значечения не известен

false

False

True

InvalidOperationException

**Ответ:** False

**Вопрос 12** **Чем отличается код 1 от кода 2**

**Код 1**

**EventHandler changed;**

**event EventHandler Changed {**

**add { changed += value; }**

**remove { changed -= value; }**

**}**

**Код 2**

**event EventHandler Changed;**

во втором случае используется блокировка на время подписывания и отписывания от события

первый вариант используется для добавления событий в структуры, а второй - классы

это аналогичный по исполнению код

первый вариант используется для подписывания к делегату таким образом, чтобы у программиста явно была на него ссылка

первый случай используется для классов, а второй для интерфейсов

подписывание и отписывание от события происходит в первом случае, а во втором не происходит (ISomeInterface.Changed = null)

**Ответ:** во втором случае используется блокировка на время подписывания и отписывания от события

**Вопрос 13** **Что произойдет в результате выполнения данного кода на C#, если он не содержит синтаксическую ошибку.**

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

**static void Main(string[] args){**

**System.Collections.Generic.Dictionary<int, int> d =**

**new System.Collections.Generic.Dictionary<int, int>();**

**d.Add(34, 20);**

**d.Add(4, 10);**

**d.Add(18, 50);**

**Console.WriteLine(d[5]);**

**}**

10

50

KeyNotFoundException

IndexOutOfRangeException

словари не поддерживают индексаторы

20

**Ответ:** KeyNotFoundException

**Вопрос 14** **Назовите какой тип у первого параметра метода расширения**

тип объекта для которого написан этот метод

int

зависит от того, какой тип напишет программист

object

string

string[]

**Ответ:** тип объекта для которого написан этот метод

**Вопрос 15** **Что произойдет в результате выполнения данного кода на C#, если он не содержит синтаксическую ошибку.**

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

**class MyComparer<T> where T : class{**

**public bool IsEquals(T a, T b) { return a == b; }**

**}**

**class Program {**

**static void Main(){**

**MyComparer<string> f = new MyComparer<string>();**

**string s1 = "target";**

**System.Text.StringBuilder sb =**

**new System.Text.StringBuilder("target");**

**string s2 = sb.ToString();**

**Console.WriteLine(f.IsEquals(s1, s2));**

**}**

**}**

1

False

true

True

false

нельзя сравнивать шаблонные типы

**Ответ:** False

**Вопрос 16** **Что произойдет в результате выполнения данного кода на C#, если он не содержит синтаксическую ошибку.**

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

**delegate int MyDelegate(Object o);**

**MyDelegate ddd = new MyDelegate((h) => { Console.WriteLine("1"); return 5; });**

**ddd += (h) => { Console.WriteLine("2"); return 5; };**

**ddd += (h) => { Console.WriteLine("3"); return 5; };**

**Console.WriteLine(ddd(null));**

1235

123

555

5

неверный синтаксис создания делегата

пропущен тип данных перенной h

**Ответ:** 1235

**Вопрос 17** **укажите все методы, что может вызывать делегат**

**delegate int MyDelegate(Object o);**

void A(string s){ Console.Write("Hello, delegates!"); }

int E(int s){ Console.Write("Hello, delegates!"); return 0; }

short C(){ Console.Write("Hello, delegates!"); return 12; }

Int32 F(Object s){ Console.Write("Hello, delegates!"); return 15; }

bool D(Object s){ Console.Write("Hello, delegates!"); return true;}

int B(object s){ Console.Write("Hello, delegates!"); return 0; }

**Ответ:** Int32 F(Object s){ Console.Write("Hello, delegates!"); return 15; }   
int B(object s){ Console.Write("Hello, delegates!"); return 0; }

**Вопрос 18** **Выберите все верные правила для лямбда-выражений**

возвращаемое значение для лямбда-выражения может быть только значимым типом

- возвращаемое значение лямбда-выражения (если таковое имеется) должно быть неявно преобразуемым в возвращаемый тип делегата

каждый входной параметр в лямбда-выражении должен быть неявно преобразуемым в соответствующий параметр делегата

лямбда-выражение должно содержать то же число параметров, что и тип делегата

размер тела лямбда-выражения не может превышать 50 строк

в лямбда-выражении нельзя вызывать на исполнение потоки

**Ответ:** - возвращаемое значение лямбда-выражения (если таковое имеется) должно быть неявно преобразуемым в возвращаемый тип делегата   
каждый входной параметр в лямбда-выражении должен быть неявно преобразуемым в соответствующий параметр делегата   
лямбда-выражение должно содержать то же число параметров, что и тип делегата

**Вопрос 19** **Какая коллекция BCL может хранить множество значений при едином ключе из перечисленных**

StringDictionary

NameValueCollection

Dictionary<K,V>

Hashtable

ListDictionary

StringCollection

**Ответ:** NameValueCollection

**Вопрос 20** **Каким образом можно изменить код, так, чтобы организовать сравнение параметров шаблонного типа (оператор >)**

**class Program {**

**static void SwapIfGreater<T>**

**(ref T lhs, ref T rhs)**

**{**

**T temp;**

**if (lhs > rhs)**

**{**

**temp = lhs;**

**lhs = rhs;**

**rhs = temp;**

**}**

**}**

**static void Main(){**

**int a=32, b=34;**

**SwapIfGreater(ref a, ref b);**

**}**

**}**

добавить ограничение типа, а именно IComparable<T>

это нельзя сделать

добавить ограничение типа, а именно struct

использовать сраврение с помощью битовых операций

добавить ограничение типа, а именно IEquatable<T>

использовать класс из стандартной библиотеки TypeComparer

**Ответ:** добавить ограничение типа, а именно IComparable<T>

**Вопрос 21** **Выберите все верные правила, на основе которых создаются классы с неограниченными параметрами типа, наример, Dictionary<TKey, TValue>**

допускается использование оператора / (деление)

допускается сравнение со значением null.

параметром типа не может быть объект перечисления

их можно преобразовывать в и из объекта System.Object или явно преобразовывать в любой тип интерфейса

параметром типа не может быть объект структуры

операторы != и == не могут использоваться из-за отсутствия гарантии того, что конкретный аргумент типа будет поддерживать эти операторы

**Ответ:** допускается сравнение со значением null.   
их можно преобразовывать в и из объекта System.Object или явно преобразовывать в любой тип интерфейса   
операторы != и == не могут использоваться из-за отсутствия гарантии того, что конкретный аргумент типа будет поддерживать эти операторы

**Вопрос 22** **Выберите все верные утверждения, объясняющие для чего необходимы делегаты**

для вызова метода, который будет известен только во время работы программы

для создания событий (оператор event)

для создания интерфейсов

для шаблонных типов

для работы расширяющих методов

для вызова некоторого метода в другом потоке

**Ответ:** для вызова метода, который будет известен только во время работы программы   
для создания событий (оператор event)   
для вызова некоторого метода в другом потоке

**Вопрос 23** **При первом построении универсального типа с типом значения в качестве параметра среда выполнения создает специализированный универсальный тип с поставляемым параметром или параметрами, которые замещаются в соответствующих местах кода MSIL. Сколько разновидностей универсального класса Stack<T> создаст CLR**

**class FIO{**

**public string FirstName{get;set;}**

**public string SecondName{get;set;}**

**public string MiddleName{get;set;}**

**}**

**class Program {**

**static void Main(){**

**System.Collections.Generic.Stack<int> diametersOfPlanets**

**= new System.Collections.Generic.Stack<int>();**

**System.Collections.Generic.Stack<long> massPlanets**

**= new System.Collections.Generic.Stack<long>();**

**System.Collections.Generic.Stack<FIO> theNamesOfTheDiscoverersOfThePlanets**

**= new System.Collections.Generic.Stack<FIO>();**

**System.Collections.Generic.Stack<string> theNamesOfThePlanets**

**= new System.Collections.Generic.Stack<string>();**

**}**

**}**

ни одного

2

6

4

5

3

**Ответ:** 3

**Вопрос 24** **Что означает ограничение типа new()**

тип может иметь только один конструктор

для типа должен быть перегружен оператор () (оператор "круглые скобки")

при создании объекта типа, необходимо сразу использовать оператор new, т.е. A a = new A();

для типа должен быть перегружен оператор new

тип должен иметь конструктор по-умолчанию

параметр типа должен быть ссылочным типом (классом)

**Ответ:** тип должен иметь конструктор по-умолчанию

**Вопрос 25** **Выберите все верные правила наследования generic классов**

при реализации generic виртуальных методов, производный необобщенный класс должен конкретизировать параметр типа

если от generic класса наследуется необобщенный класс, необходимо явно реализовать деструктор

наследник generic класса должен реализовать все абстрактные методы, если такие есть

если от generic класса наследуется другой generic класс, в нем необходимо учитывать ограничения типа, указанные в базовом классе

если от generic класса наследуется необобщенный класс, - наследник должен конкретизировать параметр типа

если от generic класса наследуется другой generic класс, в нем необходимо создать статический контруктор

**Ответ:** при реализации generic виртуальных методов, производный необобщенный класс должен конкретизировать параметр типа   
если от generic класса наследуется другой generic класс, в нем необходимо учитывать ограничения типа, указанные в базовом классе   
если от generic класса наследуется необобщенный класс, - наследник должен конкретизировать параметр типа

**Вопрос 26** **Что произойдет в результате выполнения данного кода на C#, если он не содержит синтаксическую ошибку.**

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

**static void Main(string[] args)**

**{**

**Action<int> a = new Action<int>(**

**(p) => {**

**for (int i = 0; i < p; ++i)**

**Console.WriteLine("Hello, Delegates!");**

**});**

**a = (p) => { Console.WriteLine("Give me {0}!", p); };**

**a(5);**

**}**

Hello, Delegates!  
Hello, Delegates!  
Hello, Delegates!  
Hello, Delegates!  
Hello, Delegates!  
Give me 5!

Give me 5!

неверный синтаксис создания делегата

тип Action не объявлен

Hello, Delegates!  
Give me 5!

пропущен тип данных перенной h

**Ответ:** Give me 5!

**Вопрос 27** **В каком случае НЕ следует использовать обобщенную коллекцию, например List<T>**

если коллекция будет иметь большой размер (количество элементов)

если необходимо хранить только ссылочные типы

если коллекция хранит различные типы данных

если необходимо хранить ссылки на большие объекты

на данный момент есть generic-аналоги всех необобщенных коллекций, поэтому лучше использовать только обобщенные коллекции

если необходимо хранить только значимые типы

**Ответ:** если коллекция хранит различные типы данных

**Вопрос 28** **что такое делегат в С#**

объект, инкапсулирующий интерфейсные ссылки

указатель на метод

указатель на класс

объект, инкапсулирующий метод с определенной сигнатурой

указатель на структуру

объект, инкапсулирующий поле класса

**Ответ:** объект, инкапсулирующий метод с определенной сигнатурой

**Вопрос 29** **Почему хранение в необобщенной коллекции ссылочных типов не приводит к снижению производительности**

ссылочные типы занимают меньше памяти

происходит упаковка и распаковка

ссылочные типы оптимизированы для хранения их в коллекциях

не происходит упаковка и распаковка

необобщенные коллекции оптимизированы для хранения ссылочных типов

ссылочные типы быстрее работают

**Ответ:** не происходит упаковка и распаковка

**Вопрос 30** **почему не рекомендуется применять оператор == для сравнения переменных шаблонного типа**

может оказаться, что значимый тип сравнивается с null

может оказаться, что ссылочный тип со значимым сравнимаются

операция сравнения происходит долго, т.к. среда исполнения должна первоначально определить какие типы сравниваются

создается много версий класса, для корректного сравнения ссылочных и значимых типов данных

не известно перегружен ли оператор == для этих типов

компиллятор сравнивает эти объекты только по ссылке

**Ответ:** компиллятор сравнивает эти объекты только по ссылке