«ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ» ՀԻՄՆԱԴՐԱՄ

ՎԱՆԱՁՈՐԻ ՄԱՍՆԱՃՅՈՒՂ

Ամփոփիչ քննության հարցաշար

1. Ո՞ր օպերացիոն համակրգի համար է նախատեսված .Net framework-ը:

.NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент .NET Core поддерживается на большинстве современных ОС Windows, MacOS, Linux. Используя различные технологии на платформе .NET, можно разрабатывать приложения на языке C# для самых разных платформ - Windows, MacOS, Linux, Android, iOS, Tizen.

1. CLR -

Название среды — общеязыковая среда выполнения (Common Language Runtime, CLR) — говорит само за себя: это среда выполнения, которая подходит для разных языков программирования. Основные возможности CLR (управление памятью, загрузка сборок, безопасность, обработка исключений, синхронизация) доступны в любых языках программирования, использующих эту среду.

1. MSIL or IL

-код иногда называют управляемым (managed code),потому что CLR управляет его выполнением. IL — язык более высокого уровня по сравнению с большинством других машинных языков. Он позволяет работать с объектами и имеет команды для создания и инициализации объектов, вызова виртуальных методов и непосредственного манипулирования элементами массивов. В нем даже есть команды инициирования и перехвата исключений для обработки ошибок. IL можно рассматривать как объектно-ориентированный машинный язык.

1. .Net framework-ը ստեղծել է…

2002 փետրվար

1. Ինչ է նշանակում JIT հապավումը.

Для выполнения какого-либо метода его IL-код должен быть преобразован в машинные команды. Этим занимается JIT-компилятор (Just-In-Time) среды CLR.Функции JITCompiler известен вызываемый метод и тип, в котором он определен. JITCompiler ищет в метаданных соответствующей сборки IL-код вызываемого метода. Затем JITCompiler проверяет и компилирует IL-код в машинные команды, которые сохраняются в динамически выделенном блоке памяти. После этого JITCompiler возвращается к структуре внутренних данных типа, созданной средой CLR, и заменяет адрес вызываемого метода адресом блока памяти, содержащего готовые машинные команды. В завершение JITCompiler передает управление коду

в этом блоке памяти.

1. *Ունենք հետևյալ կոդը թարգմանված MSIL-ի` if(k>7) {հրահանգներ 1;} else { հրահանգներ 2;}: Ինչպ՞ս կվարվի JIT-ը:*

JIT-компилятор может определить, что некоторое условие на том компьютере,

на котором он выполняется, всегда оказывается ложным. Допустим, метод содержит следующий фрагмент:

if (numberOfCPUs > 1) {

...

}

Если компьютер оснащен всего одним процессором, то JIT-компилятор не будет генерировать машинные команды для указанного фрагмента. В этом случае машинный код оптимизируется для конкретной машины, а следовательно, занимает меньше места и быстрее выполняется.

1. Ո՞ր լեզուներով կարելի է գրել ծրագիր .Net միջավայրում:

C# , VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET.

1. .Net միջավայրով կարելի է ստեղծել`

Классические приложения(WinForms и WPF) Универсальная платформа (Windows) Кроссплатформенные мобильные приложения(Visual Studio и Xamarin) Веб-приложения (ASP.NET)

1. .Net-ի պրոեկտում բոլոր դասերը …

Поддерживает все классы в иерархии классов .NET и предоставляет низкоуровневые службы для производных классов. Является исходным базовым классом для всех классов .NET и корнем иерархии типов.

1. Միջանկյալ լեզվի թարգմանված կոդը որևէ բարձր կարգի լեզվից …Компилятор
2. C#-հիմնային տիպերն են`

bool byte sbyte short ushort int uint long ulong float double decimal char string object

1. Ինչպես հայտարել անունների տարածք MySpace.

Пространства имен часто используются в программировании на C# двумя способами. Первый способ — .NET использует пространства имен для упорядочения множества ее классов следующим образом: System.Console.WriteLine("Hello World!"); System является пространством имен, а Console — это класс в нем. Можно использовать ключевое слово using, и тогда полное имя не потребуется. См. следующий пример:using System; Console.WriteLine("Hello");Console.WriteLine("World!");

1. Ինչ ավելացնենք կոդում, որպեսզի prt.Write(“Where is Alias”); հրահանգը տպի փակագծերի ներսի տեքստը`

Console.Write(“Where is Alias”)

1. Ինչ կանեք, եթե շփոթելով այլ լեզվի հետ int բառի փոխարեն C#-ի կոդի բազմաթիվ տեղերում գրել եք Integer,

ոչ

1. Կարելի է արդյոք մաթեմատիկական ֆունկցիաները կանչել զուտ ֆունկցիայի անունը նշելով. Օրինակ Sin(0.22), այլ ոչ թե Math.Sin(0.22);

ոչ

1. Որն է դասերի լռելյայն(default) հասանելիության մոդիֆիկատորը c#-ում:

Internal

1. Որն է դասերի անդամների լռելյայն(default) հասանելիության մոդիֆիկատորը c#-ում:

private

1. Ի՞նչ հասանելիության մոդիֆիկատոր կարող է ունենալ հատկանիշի get բլոկը::

Internal,protected,private

1. Նշեք բոլոր սխալները հետևյալ կոդում` public int D {private get; protected set;}:

public int D { get { return D; } private set { D = value; } }

1. int[,] nums2 = new int[2, 3]; ինչ արժեք ունի nums.Length-ը:

6

1. int[][] nums = new int[2][];nums[0]= new int[2] { 1, 2 }; nums[1] = new [3] { 1, 2, 3 }; որքան է nums.Length-ը:

2

1. enum Fingers : byte {thumb, index, middle, ringo, baby}; Console.Write((int)Fingers.ringo); ի՞նչ կտպվի:

3

1. enum Fingers {thumb, index, middle, ringo, baby}; Console.Write((double)Fingers.ringo+4.5); ի՞նչ կտպվի:

7.5

1. Ունենք void qar(int x) {x\*=x;} ֆունկցիան և կոդը` int k=5; qar(k); Console.Write(“x=” + k); ի՞նչ կտպվի:

x=5 որովհետև void մեթոդ է qar-ը և ոչինչ չի վերադարձնում։

1. Ինչ պետք է խմբագրենք void qar(int x) {x\*=x;} ֆունկցիային, որպեսզի տրված պարամետրը բարձրանա քառակուսի:

int qar(int x) {return x \*= x; }

1. Կարելի է արդյոք c#-ուն ֆունկցիայից ստանալ մի քանի տարբեր արժեքներ:

այո օգտագործելով ref out կամ վերադարձնելով Tuple<,>

1. Նշվածներից որոնք չեն արժեքային տիպեր` String, double, enum, array, struct, delegate; class;

String,delegate; class,array

1. Նշվածներից որոնք չեն հղումային տիպեր` String, double, bool; enum, struct, array, delegate; class;

double, bool,enum, struct

1. Ունենք struct MyStr{public int k;} և կոդ ` myStr str = new myStr(); str.k=66; myStr str2 = str; str2.k = 77; Console.WriteLine(str2.k+" " +str.k); Ի՞նչ կտպվի: 77 77
2. Ունենք class MyStr{public int k;} և կոդ ` myStr str = new myStr(); str.k=66; myStr str2 = str; str2.k = 77; Console.WriteLine(str2.k+" " +str.k); Ի՞նչ կտպվի: 77 77
3. Ինչ դեպքում է հնարավոր նույն ֆունկցիան նկարագրել 3 պարամետրով, բայց կանչել 2-ով:

Public void Test(int k, bool m,string s=null);

Public void Test(int? k=null, bool m,string s);

1. sealed- մեթոդը`

Модификатор sealed можно использовать для метода или свойства, которое переопределяет виртуальный метод или свойство в базовом классе. Это позволяет классам наследовать от вашего класса, запрещая им при этом переопределять определенные виртуальные методы или свойства.

1. sealed- բառը դասը հայտարարելիս

Если класс объявлен с модификатором **sealed**, то от этого класса нельзя наследовать и создавать производные классы.

1. Կարելի՞ է արդյոք ժառանգել ստատիկ դասը:

Статические классы запечатаны и поэтому не могут быть унаследованы. Они не могут наследовать ни от какого класса, кроме Object.

1. Կարող՞ է արդյոք ստատիկ դասը ժառանգել այլ դասից:

Статические классы запечатаны и поэтому не могут быть унаследованы. Они не могут наследовать ни от какого класса, кроме Object.

1. դասի readonly անդամը`

Поля для чтения можно инициализировать при их объявлении либо на уровне класса, либо инициилизировать и изменять в конструкторе. Инициализировать или изменять их значение в других местах нельзя, можно только считывать их значение.

Поле для чтения объявляется с ключевым словом readonly

1. դասի const անդամը`

Константа должна быть проинициализирована при определении

После определения значение константы не может быть изменено

Константы предназначены для описания таких значений, которые не должны изменяться в программе. Для определения констант используется ключевое слово const.

При использовании констант надо помнить, что объявить мы их можем только один раз и что к моменту компиляции они должны быть определены.Константы должны быть определены во время компиляции, а поля для чтения могут быть определены во время выполнения программы.Соответственно инициализировать константу можно устанновить только при ее определении. Поле для чтения можно инициализировать либо при его определении, либо в конструкторе класса. Константы не могут использовать модификатор static, так как уже неявно являются статическими. Поля для чтения могут быть как статическими, так и не статическими.

1. new void fun(int x)… որն է new բառի իմաստը

Անտանելի վատ գրած օրինակ,,,,եթե ունենք երկու կլաս

class Person

{

public string Name { get; set; }

public Person(string name)

{

Name = name;

}

public void Display()

{

Console.WriteLine(Name);

}

}

class Employee : Person

{

public string Company { get; set; }

public Employee(string name, string company)

: base(name)

{

Company = company;

}

public new void Display()

{

Console.WriteLine($"{Name} work in {Company}");

}

}

Եթե ստեղծենք

Person tom = new Employee("Tom", "Microsoft"); օբեկտը ապա

tom.Display(); կտպի Tom այլ ոչ թե Tom work in Microsoft.

1. Ո՞րն է է abstract բառի նպատակը դասերի հայտարարման ժամանակ`

Кроме обычных классов в C# есть абстрактные классы. Абстрактный класс похож на обычный класс. Он также может иметь переменные, методы, конструкторы, свойства. Единственное, что при определении абстрактных классов используется ключевое слово abstract.

Но главное отличие состоит в том, что мы не можем использовать конструктор абстрактного класса для создания его объекта.

Кроме обычных свойств и методов абстрактный класс может иметь абстрактные члены классов, которые определяются с помощью ключевого слова abstract и не имеют никакого функционала. В частности, абстрактными могут быть:

Методы Свойства Индексаторы События

Абстрактные члены классов не должны иметь модификатор private. При этом производный класс обязан переопределить и реализовать все абстрактные методы и свойства, которые имеются в базовом абстрактном классе. При переопределении в производном классе такой метод или свойство также объявляются с модификатором override (как и при обычном переопределении виртуальных методов и свойств). Также следует учесть, что если класс имеет хотя бы одный абстрактный метод (или абстрактные свойство, индексатор, событие), то этот класс должен быть определен как абстрактный.

1. Ո՞րն է abstract և virtual մեթոդների տարբերությունը:

Виртуальные методы имеют реализацию и предоставляют производным классам возможность переопределения.

Абстрактные методы не предоставляют реализацию и заставляют производные классы переопределять метод.

1. ինչ է նշանակում this(a), երբ կարող է օգտագործվել:

Ключевое слово this представляет ссылку на текущий экземпляр класса.Для квалификации элементов, скрытых одинаковыми именами

Для передачи другим методам объекта в качестве параметра

Для объявления индексаторов

У статических функций-членов нет указателя this, так как они существуют только на уровне класса и не являются частями объектов. Использование this в статическом методе приводит к ошибке.

1. Երբ կարիք կա կանչելու բազային կոնստրուկտորը:

Каждый раз, когда создается класс или структура, вызывается конструктор. Конструкторы имеют имя, совпадающее с именем класса или структуры, и обычно инициализируют члены данных нового объекта.Конструктор, который не принимает никаких параметров, называется конструктором без параметров. Конструкторы без параметров вызываются всякий раз, когда создается экземпляр объекта с помощью оператора new, а аргументы в new не передаются. Если класс не является статическим, компилятор C# выделяет классам без конструкторов открытый конструктор без параметров, позволяющий создавать экземпляры классов.

1. delegate-ը հնարավորություն կտա`

Делегаты представляют такие объекты, которые указывают на методы. То есть делегаты - это указатели на методы и с помощью делегатов мы можем вызвать данные методы.

1. event-ը

События сигнализируют системе о том, что произошло определенное действие. И если нам надо отследить эти действия, то как раз мы можем применять события.

События объявляются в классе с помощью ключевого слова event, после которого указывается тип делегата, который представляет событие:

delegate void AccountHandler(string message);

event AccountHandler Notify;

В данном случае вначале определяется делегат AccountHandler, который принимает один параметр типа string. Затем с помощью ключевого слова event определяется событие с именем Notify, которое представляет делегат AccountHandler. Название для события может быть произвольным, но в любом случае оно должно представлять некоторый делегат.

Определив событие, мы можем его вызвать в программе как метод, используя имя события:

Notify("Произошло действие");

Поскольку событие Notify представляет делегат AccountHandler, который принимает один параметр типа string - строку, то при вызове события нам надо передать в него строку.Однако при вызове событий мы можем столкнуться с тем, что событие равно null в случае, если для его не определен обработчик. Поэтому при вызове события лучше его всегда проверять на null. Например, так:

if(Notify !=null) Notify("Произошло действие");

1. class MyCl {public int К{get; set}; public static MyCl operator++(MyCl mc){return new MyCl {K=mc.K+1} , վերբեռնել պրեֆիկսային ++;

Հն՞

1. Նշվածներից որ օպերատորները չեն վերբեռնվում c#-ում. պրեֆիքսային --; >, []; ?:, ~; % ? []
2. Հնարավոր է արդյոք վերբեռնել [] օպերատորը c#-ում: Доступ к элементам не считается перегружаемым оператором
3. Բացառիկ իրավիճակների մշակման կառույցներում պարտադի՞ր է finally բլոկի առկայությունը: պարտադիր չէ
4. Հնարավո՞ր է արդյոք հայեցողաբար գեներացնել բացառություն:

Базовым для всех типов исключений является тип Exception. Этот тип определяет ряд свойств, с помощью которых можно получить информацию об исключении.

InnerException: хранит информацию об исключении, которое послужило причиной текущего исключения

Message: хранит сообщение об исключении, текст ошибки

Source: хранит имя объекта или сборки, которое вызвало исключение

StackTrace: возвращает строковое представление стека вызывов, которые привели к возникновению исключения

TargetSite: возвращает метод, в котором и было вызвано исключение

Но также есть более специализированные типы исключений, которые предназначены для обработки каких-то определенных видов исключений. Их довольно много, я приведу лишь некоторые:

DivideByZeroException: представляет исключение, которое генерируется при делении на ноль

ArgumentOutOfRangeException: генерируется, если значение аргумента находится вне диапазона допустимых значений

ArgumentException: генерируется, если в метод для параметра передается некорректное значение

IndexOutOfRangeException: генерируется, если индекс элемента массива или коллекции находится вне диапазона допустимых значений

InvalidCastException: генерируется при попытке произвести недопустимые преобразования типов

NullReferenceException: генерируется при попытке обращения к объекту, который равен null (то есть по сути неопределен)

1. Ինչ պետք է անել, ստեղծելու սեփական բացառույան տիպ

Кроме обычных типов фреймворк .NET также поддерживает обобщенные типы (generics), а также создание обобщенных методов. Чтобы разобраться в особенности данного явления, сначала посмотрим на проблему, которая могла возникнуть до появления обобщенных типов. Посмотрим на примере. Допустим, мы определяем класс для представления банковского счета. К примеру, он мог бы выглядеть следующим образом:

class Account

{

public int Id { get; set; }

public int Sum { get; set; }

}

Класс Account определяет два свойства: Id - уникальный идентификатор и Sum - сумму на счете.

1. Equals մեթոդը

Определяет, равен ли указанный объект текущему объекту.

Параметры

obj

Object

Объект, сравниваемый с текущим объектом.

Возвращаемое значение

Boolean

Значение true, если указанный объект равен текущему объекту; в противном случае — значение false.

1. String ss=”my@ name@ is@ Jon” ; ինչպես հեռացնել բոլոր @-երը

string ss= "my@ name@ is@ Jon";

ss=ss.Replace("@", "");

2020թ.