**Заяц Александр Владимирович:**

1. Что такое Git? Для чего он используется?

Git – система контроля версий, используемая для отслеживания изменений в коде. Git используется для облегчения процесса совместной разработки, ведения истории изменений, отслеживания и управления версиями проекта.

1. Что такое управляемый/неуправляемый код?

Код, выполняющийся под управлением среды выполнения, называется управляемым кодом. И наоборот, код, выполняемый вне среды выполнения, называется неуправляемым кодом.

1. Что такое **CLR**?

Common Language Runtime (общеязыковая исполняющая среда) - исполняющая среда для байт-кода CIL (MSIL), в который компилируются программы, написанные на .NET-совместимых языках программирования.

1. Что такое коллекция? Какие типы коллекций вы знаете в C#?

Структура данных, которая позволяет хранить и управлять набором объектов.

Коллекции без типизации:

- ArrayList

- Hashtable

- Queue

- Stack

Коллекции с типизацией:

- List<T>

- Dictionary<TKey, TValue>

- Queue<T>

- Stack<T>

- HashSet<T>

1. Что такое исключение? Как обработать исключение в C#?

Исключение – событие, возникающее в процессе выполнения программы, которое нарушает ее нормальное функционирование. Обработка исключения указывает на действие, которое необходимо выполнить при возникновении ошибки, чтобы предотвратить принудительное завершение программы. Для обработки исключения используется конструкция try-catch-finally.

1. Опишите конструкцию **try-catch-finally**, что вы о ней знаете.

При использовании конструкции try-catch-finally вначале исполняются все конструкции в блоке try. Если в этом блоке не возникло исключений, то после его выполнения начинается выполняться блок finally и затем конструкция try-catch-finally завершает свою работу. Если в блоке try возникает исключение, то обычный порядок исполнения останавливается, и среда CLR начинает искать блок catch который может обработать данное исключение. Если нужный блок catch найден, то он выполняется, и после его завершения выполняется блок finally. Если нужный блок catch не найден, то при возникновении исключения программа аварийно завершит ее выполнение.

1. Если сделать **return** из **catch** выполнится ли **finally**?

Да

1. Перечислите **SOLID** принципы.

Single Responsibility Principle – принцип единственной ответственности

Open Closed Principle - принцип открытости-закрытости

Liskov Substitution Principle – принцип подстановки Барбары Лисков

Interface Segregation Principle – принцип разделения интерфейсов

Dependency Inversion Principle – принцип инверсии зависимостей

1. Что такое статический класс? В чем его особенности?

Статические классы объявляются с модификатором static и могут содержать только статические поля, свойства и методы. Статические классы имеют следующие ограничения:

- Нельзя создавать экземпляр класса, используя ключевое слово new

- Не разрешается использовать не статические члены этого же класса

- Он не поддерживает наследование

1. В чем отличие **delegate**, **Action**, **Func** и **Predicate**?

Delegate - это базовый тип, который представляет ссылку на метод с определенной сигнатурой (списком параметров и типом возвращаемого значения). Делегаты позволяют передавать методы как аргументы или возвращать их из других методов. Делегаты могут указывать на методы с любой сигнатурой и требуют явного объявления типа делегата.

Action - это встроенный обобщенный делегат, который представляет метод без возвращаемого значения (void). Он может принимать от 0 до 16 параметров.

Func - это встроенный обобщенный делегат, который представляет метод с возвращаемым значением. Последний тип в обобщении указывает на тип возвращаемого значения, а остальные - на типы параметров. Может принимать от 0 до 16 параметров.

Predicate - это встроенный обобщенный делегат, который представляет метод, возвращающий bool.

1. Может ли интерфейс объявить событие?

Да

1. Для чего в **.NET** используется конструкция **using(…){…}**, и во что она разворачивается?

Конструкция using(…){…} в .NET используется для обеспечения корректного освобождения неуправляемых ресурсов, таких как файловые потоки, соединения с базами данных, другие объекты, которые реализуют интерфейс IDisposable. Она гарантирует, что метод Dispose() будет вызван автоматически, даже если в блоке using произойдет исключение.

Конструкция

using (var resource = new SomeDisposableResource())

{

}

Разворачивается в

var resource = new SomeDisposableResource();

try

{

}

finally

{

if (resource != null)

{

((IDisposable)resource).Dispose();

}

}

1. Синхронизация потоков, какие виды блокировок вы знаете.

Простые методы блокировки потока:

- Sleep

- Join

Объекты синхронизации – служат для упорядочивания доступа потоков к ресурсу:

- Lock (Mutex)

- Monitor

- Semaphore

1. Коротко опишите алгоритм работы сборщика мусора

Когда CLR запускает сборку мусора, она сначала приостанавливает все потоки в процессе, затем переходит к этапу уборки под названием маркировка, в котором определяется какие объекты переживут уборку мусора, а какие нет. Следующей фазой является сжатие, в которой CLR помещает все «немусорные» объекты вниз памяти, чтобы они занимали смежный блок памяти. Затем идет фаза очищения памяти и обновление поколений, в котором выжившие объекты перемещаются в следующее поколение.

1. Сколько существует поколений в **GC**?

3 поколения

1. Перечислите ограничения обобщенных типов и их назначение.

where T : struct – тип должен быть значимым

where T : class – тип должен быть ссылочным

where T : new() – должен иметь публичный конструктор без параметров

where T : BaseClass – должен быть наследником указанного класса

where T : IInterface – должен реализовать указанный интерфейс

1. Отличаются ли **IEnumerable<T>** и **IQueryable<T>**? Если да, в чем отличие?

Отличаются, IEnumerable<T> - это интерфейс, который предоставляет возможность последовательного перебора элементов коллекции. Все операции (фильтрация, сортировка и т.д.) выполняются в памяти на стороне клиента. IQueryable<T> - это интерфейс, который позволяет выполнять запросы к данным на стороне источника данных (например, база данных). Все операции (фильтрация, сортировка и т.д.) преобразуются в выражение (например, SQL-запрос) и выполняются на стороне сервера.

1. Отличаются ли **Object.GetType()** и **typeof()**? Если да, в чем отличие?

Отличаются, Object.GetType() вызывается на экземпляре объекта, а typeof() используется для получения типа на этапе компиляции, он работает с именем типа, а не с экземпляром объекта.

1. Можно ли реализовать в одном классе два интерфейса описывающих методы с одинаковой сигнатурой.

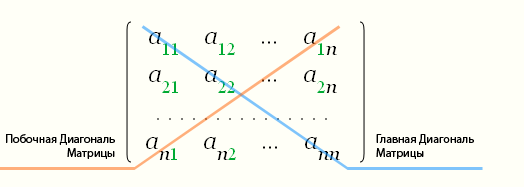
Можно

1. Отличаются ли Асинхронность и Параллельность? Если да, в чем отличие?

Отличаются. Асинхронность говорит о порядке исполнения кода. Если вызываемая функция не возвращает значение сразу, а отдаёт управление вызывающему коду с обещанием выдать значение позже, то эта функция асинхронная. Параллельность говорит о том, что в машине физически происходит несколько процессов одновременно.

Задача 1:

Объявить двумерный массив для элементов типа **int** (минимум 4х4). Инициализировать его вручную или random. Присвоить значение 1 всем элементам массива, которые лежат выше главной диагонали.



Задача 2:

Разработайте метод расширения для типа **int**, возвращающий значение в виде строки.

Пример: Возвращает «1234» в виде «один два три четыре».

Задача 3:

Объявить коллекцию элементов типа **int**. Инициализировать его вручную или random. При помощи **LINQ** отсортировать коллекцию по возрастанию и вернуть вторую половину коллекции (округляя вверх если число элементов нечётное) отсортированную по убыванию, где каждый элемент будет возведён в квадрат.

Задача 4:

Разработайте класс содержащий произвольный публичный метод и событие. После 10 вызовов метода у любых экземпляров разработанного класса необходимо вызвать событие и информировать об этом пользователя. Обработку события произвести либо при помощи анонимного метода, либо при помощи лямбда-выражения.   
(Пояснение: если будет напр. 3 экземпляра и для первого экземпляра вызовется метод 2 раза, для второго 7 раз и для третьего 1, то при следующем вызове этого метода, вне зависимости от конкретного экземпляра, должно вызываться событие у экземпляра для которого произошёл вызов)

Задача 5:

Используя обобщения разработайте несколько интерфейсов, первый - для сохрания/чтения последовательности объектов произвольного типа в/из источник данных, второй - для CRUD операций (взаимодействие с последовательностью объектов в памяти). Разработайте несколько классов, реализующих интерфейсы, разработанные ранее. Вид последовательности элементов и файл для хранения выберите самостоятельно при создании реализации интерфейса. Не забывайте про принципы ООП.