**1. Состав. NET Framework. Структура среды выполнения CLR.**

**.Net framework** состоит из **BCL**(Base class library) и **CLR**(Common language runtime (общеязыковая исполняющая среда)).

**CLR** состоит из

**JIT-компилятора**,

**MSIL**(высокоуровневый ассемблер(байт-код)),

**CLI**(Common language infrastructure(определяет, что должно присутствовать в минимально рабочей версии CLR, какие классы и типы должны быть из BCL))

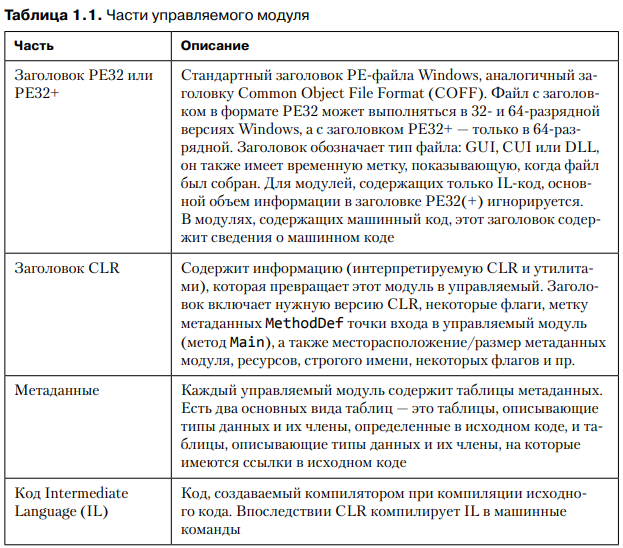
**FCL**(Framework class library) – библиотека объектно-ориентированных классов, интерфейсов и системы типов.

**CTS**(Common type system ) – система общих типов.

**2. Структура управляемого модуля - portable executable (PE). Понятие и исполнение сборки. CIL.**

**Portable Executable** (PE, «переносимый исполняемый») — формат исполняемых файлов, объектного кода, используемый в 32- и 64-разрядных версиях операционной системы Microsoft Windows. Формат PE представляет собой структуру данных, содержащую всю информацию, необходимую PE-загрузчику для отображения файла в память.

**Структура PE**

****

**Сборка:**

1) это абстрактное понятие, для логической группировки одного или нескольких управляемых модулей или файлов ресурсов.   
2) дискретная единица многократно используемого кода внутри CLR.

**3. CTS (Common Type System). Типы данных C#. Ссылочные и типы значений.**

**Система общих типов CTS** определяет способ объявления, использования и управления типами в среде CLR, а также является важной составной частью поддержки межъязыковой интеграции в среде выполнения.

Все типы на платформе .NET делятся на **типы значений и ссылочные типы**.

Система общих типов CTS на платформе .NET поддерживает следующие пять категорий типов:

**Классы, Структуры, Перечисления, Интерфейсы, Делегаты**

**4. Понятие упаковки и распаковки типов. Типы Nullable: преобразование, проверка, null-объединение**

Когда любой значимый **тип** присваивается к ссылочному **типу** данных, значение перемещается из области стека в кучу. Эта операция называется **упаковкой**.

Когда любой ссылочный **тип** присваивается к значимому **типу** данных, значение перемещается из области кучи в стек. Это называется **распаковкой**.

Преобразование к nullable – **неявное**

Преобразование от nullable – **явное**

**проверка на null**

**a = x ?? y**; - если x = null, возвращает y, иначе возвращает x

**classA?.classB?.method()** – method() вызовется тогда, когда classA и classB не равны null

**5. Тип данных String: операции, литералы, пустые и нулевые строки, форматированный вывод.**

Операции: Конкатенация – + или join()

Сравнение – Compare() (-1 если перв. Строка больше, -1 – если меньше и 0 – если равны)

Поиск – IndexOf() StartWith()

Разделение – Split()

Обрезка – Trim()

Вставка – Insert()

Удаление – Remove()

Замена – Replace()

**6. Неявная типизация – назначение и использование.**

Неявная типизация – с помощью **var**

**7. Массивы C# одномерные, прямоугольные и ступенчатые.**

Одномерный – int[] array = {1,2}

Прямоугольный – int[,] array = {{1,1}{2,2}}

Ступенчатый – int[][] array = new int[x][];

**8. Понятие кортежей. Свойства, создание**

Свойства: может соединять разнотипные данные, можно декомпозировать, можно обратиться к любому элементу, можно сравнивать и деконструировать.

Удобен для обмена значениями переменных, возврата из функции нескольких значений

**Отличие System.ValueTuple и System.Tuple**

Типы System.ValueTuple являются типами значений. Типы System.Tuple являются ссылочными типами.

Типы System.ValueTuple являются изменяемыми. Типы System.Tuple являются неизменяемыми.

Элементами данных типов System.ValueTuple являются поля. Элементами данных типов System.Tuple являются свойства.

**9. Принципы объектно-ориентированного программирования.**

* *Абстракция*. Моделирование требуемых атрибутов и взаимодействий сущностей в виде классов для определения абстрактного представления системы.
* *Инкапсуляция*. Скрытие внутреннего состояния и функций объекта и предоставление доступа только через открытый набор функций.
* *Наследование*. Возможность создания новых абстракций на основе существующих.
* *Полиморфизм*. Возможность реализации наследуемых свойств или методов отличающимися способами в рамках множества абстракций.

**10. Класс. Элементы класса. Свойства и индексаторы.**

Класс — модель для создания объектов определённого типа, описывающая их структуру (набор полей и их начальное состояние) и определяющая алгоритмы (функции или методы) для работы с этими объектами

Свойства — это методы класса. Доступ к ним осуществляется так же, как если бы они были полями этого класса. Свойство может защитить поле класса от изменений (независимо от объекта).

Индексаторы – свойства, которые индексируют экземпляры класса.

public T this[int i]

{

get => arr[i];

set => arr[i] = value;

}

**11. Класс. Константы. Поля только для чтения. Инициализаторы класса.**

Инициализотор: class = new class{field1, field2};

**12. Спецификаторы доступа C#. Видимость типов. Доступ к членам типов.**

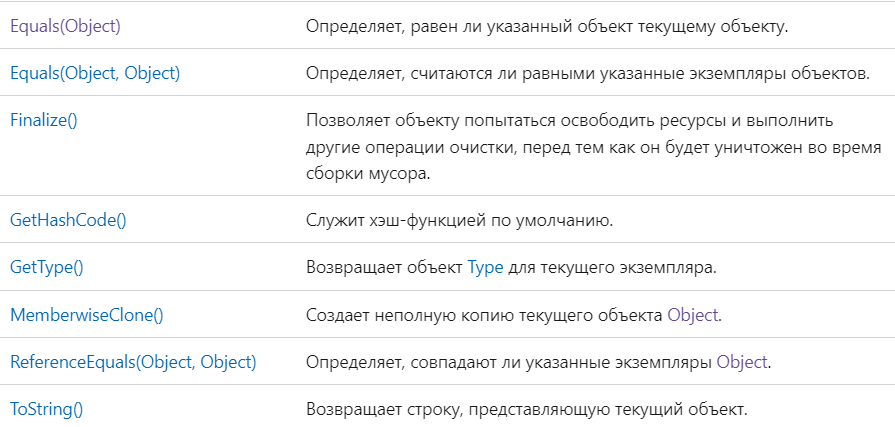


Доступ к классам, структурам: если класс не вложен – public, internal. Если класс вложен – public, internal, private

**13. Класс. Конструкторы и их свойства. Деструкторы**

**14. Класс и методы System.Object.**

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.



**15. Статические методы и статические конструкторы класса.**

**16. Статические классы. Методы расширения и правила их определения.**

**17. Анонимные типы.**

var user = new { Name = "Tom", Age = 34 };

свойства анонимных типов доступны только для чтения

Иногда возникает задача использовать один тип в одном узком контексте или даже один раз. Создание класса для подобного типа может быть избыточным.

**18. Модификаторы параметров - ref , out, params. Необязательные и именованные аргументы.**

**19. Перегрузка методов и операторов. Правила перегрузки операторов.**

20. Операции преобразования типа. Явная и неявная форма. Ограничения.

Явная форма – explicit неявная форма – implicit

public static implicit operator Counter(int x)

{

return new Counter { Seconds = x };

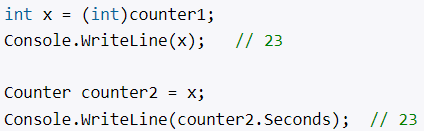
}

public static explicit operator int(Counter counter)

{

return counter.Seconds;

}



**21. Вложенные типы. Вложенные объекты**

**22. Правила наследования C#.**

модификатор наследующего класса должен быть таким же, как и у наследуемого или строже.

Запрещено множественное наследование классов.

**23. Сокрытие имен при наследовании. Обращение к срытым членам**

Скрытие через new, обращение через base

**24. Использование операций is и as**

Is возвращает булевое значение, а as – ссылку

25. Полиморфизм. Виртуальные методы, свойства и индексаторы. Правила переопределения.

**26. Понятие раннего и позднего связывания.**

**27. Абстрактные классы и методы. Бесплодные классы.**

**28. Структур в C#.**

**29. Интерфейсы. Свойства интерфейсов. Реализация интерфейсов.**

**30. Явная и неявная реализация интерфейсов. Работа с объектами через интерфейсы.**

**31. Ковариантность интерфейсов. Контравариантность интерфейсов**

Ковариантность: позволяет использовать более конкретный тип, чем заданный изначально

Контравариантность: позволяет использовать более универсальный тип, чем заданный изначально

**32. Стандартные интерфейсы .NET. Назначение и применение.**

ICloneble – глубокое клонирование

IComparable – Сравнение объектов

IEnumerator, IEnumerable – для коллекций

**33. Исключительные ситуации. Генерация и повторная генерация исключений.**

**34. Исключительные ситуации. Варианты обработки исключений. Фильтры исключений**

**35. Обобщения (generics). Свойства обобщений.**

**36. Концепция ограничений обобщений. Статические члены обобщений.**

**37. Делегаты. Определение, назначение и варианты использования. Обобщенные делегаты.**

**38. Анонимные функции. Лямбда-выражения.**

**39. Обобщённые делегаты .NET. Action, Func, Predicate**

public delegate void Action<in T>(T obj) – 16 параметров

delegate bool Predicate<in T>(T obj); - 1 параметр

TResult Func<in T, out TResult>(T arg) – 16 параметров и 1 out

**40. События и делегаты.**

**41. Стандартные коллекции .NET. Типы коллекций.**

Классы System.Collections.Generic - обобщенные

Классы System.Collections.Concurrent -обобщенные потокобезопасные

Классы System.Collections – коллекци object

**42. Стандартные интерфесы коллекций.**

ICollection

IList

IDictionary

**43. IEnumerable и IEnumerator**

IEnumerator – перечислитель(контейнер)

public interface IEnumerable

{

IEnumerator GetEnumerator();

}

IEnumerable – перечисляемый

public interface IEnumerator

{

bool MoveNext(); // перемещение на одну позицию вперед в контейнере элементов

object Current {get;} // текущий элемент в контейнере

void Reset(); // перемещение в начало контейнера

}

**44. LINQ to Objects. Синтаксис. Форма. Возврат результата. Грамматика выражений запросов. Отложенные и неотлаженные операции.**

**45. LINQ to Objects. Операции Where, Select, Take, OrderB, Join, GroupBy**

**46. Рефлексия. System Type.**

**47. Классы для работы с файловой системой.**

**48. Синтаксическая конструкция using. Чтение и запись файлов. Потоковые классы.**

**49. Классы адаптеры потоков.**

**50. Сериализация. Форматы сериализации.**

**51. Сериализация контрактов данных. интерфейс ISerializable.**

52. Атрибуты. Создание собственного атрибута. -

53. Процесс. Домен приложений. Поток выполнения.

54. Создание потоков , классы приоритетов. Состояния потоков

55. Синхронизация потоков. Lock. Monitor. Мutex. Semaphore

56. Библиотека параллельных задач TPL. Класс Task. Состояние задачи.

57. Способы создания Task. Возврат результата. Отмена выполнения задач. Продолжения.

58. Параллелизм при императивной обработке данных. Класс Parallel

59. Асинхронные методы. async и await

60. Проектирование отношений. Агрегация, композиция и ассоциация

61. Антипаттерны проектирования. Рефакторинг. Методы рефакторинга.

62. Чистый код. Требования к именам, функциям, форматированию.

63. Чистый код. Требования к классам и объектам. Закон Деметры. DTO. Избыточный код.

64. Паттерны проектирования. Классификация. Порождающие Abstarct Factory, Factory Method

65. Порождающие паттерны проектирования :Builder, Object pool, Prototype.

66. Порождающие паттерны проектирования: Singleton – 4 реализации, Lazy initialization,

67. Структурные паттерны: Adapter, Decorator

68. Структурные паттерны: Composite, Facade

69. Структурные паттерны: Proxy, Bridge

70. Паттерны поведения: Chain of Responsibility, Command

71. Паттерны поведения: Iterator, Mediator, Visitor

72. Паттерны поведения: Memento, Observer, Null Object, Strategy

73. Принципы проектирования SOLID.