1. **Преимущества и недостатки платформы Microsoft .NET**:
    a. **Преимущества**: кроссплатформенность, высокая производительность, поддержка множества языков, большое количество библиотек.
    b. **Недостатки**: сложности с переносом старых приложений, необходимость изучения новых технологий.
2. **Что такое BCL(FCL)?**:
    a. **BCL (Base Class Library)**: набор библиотек, предоставляющий базовые функциональные возможности для работы с .NET.
    b. **FCL (Framework Class Library)**: включает в себя BCL и дополнительные библиотеки для более специфичных задач.
3. **Плюсы и минусы языка программирования C#**:
    a. **Плюсы**: строгая типизация, простота синтаксиса, поддержка параллельного и асинхронного программирования.
    b. **Минусы**: меньше кроссплатформенных возможностей по сравнению с некоторыми другими языками, требовательность к ресурсам.
4. **Назначение рефлекторов и дотфускаторов**:
    a. **Рефлектор**: позволяет изучать и изменять структуру кода на этапе выполнения.
    b. **Дотфускатор**: усложняет код для защиты от реверс-инжиниринга.
5. **Разновидности типов данных в C#**:
    a. Простые типы (int, float, double, char)
    b. Сложные типы (классы, структуры, массивы)
6. **Ключевое слово var и его применение**:
    a. Позволяет компилятору автоматически определять тип переменной на основании присваиваемого значения.
7. **Структурные и ссылочные типы**:
    a. **Структурные**: хранятся в стеке, копируются при присваивании.
    b. **Ссылочные**: хранятся в куче, передаются по ссылке.
8. **Ввод, вывод в консольном приложении**:
    a. Использование `Console.ReadLine()` для ввода и `Console.WriteLine()` для вывода.
9. **Явное и неявное приведение типов**:
    a. **Явное**: выполняется вручную, используя операторы приведения.
    b. **Неявное**: выполняется автоматически, если нет потери данных.
10. **Операторы языка C#**:
    a. **Арифметические**: +, -, *, /, %.
    b. **Логические**: &&, ||, !.
    c. **Сравнения**: ==, !=, <, >, <=, >=.
11. **Условные операторы в C#**:
    a. `if`, `else if`, `else`, `switch`.
12. **Циклические операторы в C#**:
    a. `for`, `while`, `do-while`, `foreach`.
13. **Использование массивов в C#**:
    a. Объявление: `int[] array = new int[10];`
    b. Доступ к элементам: `array[0] = 1;`
14. **Ключевое слово null и его применение**:
    a. Обозначает отсутствие значения у ссылочной переменной.
15. **Строки в C#**:
    a. Использование класса `String` и его методов (например, `Substring`, `IndexOf`).

16. **Различие между структурами и классами**:
    a. **Структуры**: значимые типы, хранятся в стеке.
    b. **Классы**: ссылочные типы, хранятся в куче.
17. **Перегрузка методов**:
    a. Создание нескольких методов с одинаковым именем, но разными параметрами.
18. **Перегрузка операторов**:
    a. Позволяет определить новые реализации стандартных операторов для пользовательских типов.
19. **Использование ключевого слова this**:
    a. Указывает на текущий экземпляр класса.
20. **Конструкторы и деструкторы**:
    a. **Конструкторы**: инициализируют объекты.
    b. **Деструкторы**: выполняют очистку перед удалением объектов.
21. **Исключения. try, catch, throw, finally**:
    a. Обработка ошибок с помощью блоков `try`, `catch`, `finally`, выброс исключений `throw`.
22. **Применение конструкций checked и unchecked**:
    a. Контроль переполнения арифметических операций.
23. **Пространства имен в программах C#**:
    a. Группировка классов и других типов данных для предотвращения конфликтов имен.
24. **Свойства. get, set**:
    a. Определение методов доступа к полям класса.
25. **Особенности наследования в C#**:
    a. Поддержка одиночного наследования, возможность использования интерфейсов.
26. **Виртуальные методы**:
    a. Методы, которые могут быть переопределены в производных классах.
27. **Абстрактный класс**:
    a. Класс, который не может быть инстанцирован и служит основой для других классов.
28. **Делегаты. Цели и применение**:
    a. Ссылки на методы, используемые для передачи методов в качестве параметров.
29. **Интерфейсы. Цели и применение**:
    a. Определяют контракт для классов, которые их реализуют.
30. **Перечисления. Цели и применение**:
    a. Набор именованных констант, используемых для представления фиксированных значений.
31. **Стандартные интерфейсы**:
    a. Встроенные интерфейсы, такие как `IComparable`, `IEnumerable`.
32. **Использование анонимных и лямбда-функций**:
    a. Анонимные методы: `delegate`.
    b. Лямбда-выражения: `x => x * x`.
33. **Методы расширения (Extension)**:
    a. Добавляют новые методы в существующие типы без их изменения.
34. **Назначение и использование Generic**:
    a. Позволяют создавать классы и методы, работающие с любыми типами.
35. **Коллекции. Generic-коллекции**:
    a. Использование коллекций, таких как `List<T>`, `Dictionary<TKey, TValue>`.
36. **Сборщик мусора**:

    a. Управляет автоматической очисткой неиспользуемых объектов из памяти.
37. **Работа с файлами**:
    a. Использование классов `FileStream`, `StreamWriter`, `StreamReader`, `BinaryWriter`, `BinaryReader`.
38. **Работа с директориями**:
    a. Использование классов `Directory`, `DirectoryInfo`, `FileInfo`.
39. **Регулярные выражения**:
    a. Использование класса `Regex` для работы с шаблонами поиска и замены текста.
40. **LINQ, цели назначение, применение**:
    a. Язык интегрированных запросов для работы с коллекциями данных.
41. **Атрибуты, сериализация**:
    a. Использование атрибутов для добавления метаданных к коду, сериализация для сохранения объектов в файл.