1. Перечислите свойства внутренних и вложенных классов?

*Ответ:*

**Вложенные классы.**

В языке C# любой класс в своей реализации может содержать объявление другого класса. Класс, который объявляется в пределах фигурных скобок другого класса, называется **вложенным классом, или внутренним**.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**1) Вложенный тип имеет доступ ко всем членам, которые доступны вмещающему типу**. Он **может получать доступ к закрытым и защищенным членам вмещающего типа**, включая любые наследуемые защищенные члены.

**2) Вложенные классы, объявленные как public, позволяют создавать экземпляры коду, внешнему по отношению к охватывающему классу.**

3) Пользователю разрешено **создавать объекты вложенного класса во внешнем классе.**

4) **Вложенный класс может получить доступ к статическому члену, объявленному во внешнем классе.**

**5) По умолчанию вложенный класс является private.**

6) Разрешено **наследовать вложенный класс от внешнего класса**.

Изображение выглядит как текст

Автоматически созданное описание

2. Что такое статический класс и какие у него свойства?

*Ответ:*

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Класс считается статическим, если при его создании, перед ключевым словом **class** указывается ключевое слово **static**. Определение класса статическим, означает то, что этот **класс является набором только статических сущностей** (полей, методов, свойств и т.п.), и, **внимание**, **создавать объекты такого класс нельзя.**

Кроме этого, на статический класс, накладываются как минимум следующие ограничения:

* **статический класс не может служить базовым для других классов**, а также сам **не может быть наследником какого-либо производного класса;**
* класс **не может иметь перегруженных операций.**

Изображение выглядит как текст, табличка, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

3. Каково назначение перегрузки операторов?

*Ответ:* **Оператор — это символ, который сообщает компилятору о необходимости выполнения некоторых математических или логических действий.**

В языке программирования С# процесс, **который позволяет определять назначение оператора по отношению к создаваемому классу, называется перегрузка операторов**. Он **расширяет область применения оператора в классе**. **Поведение оператора можно менять и контролировать**.

**Перегрузка операторов**  — **один из способов реализации**[**полиморфизма**](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), заключающийся **в возможности одновременного существования в одной**[**области видимости**](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D1%8C_%D0%B2%D0%B8%D0%B4%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8)**нескольких различных вариантов применения операторов**, имеющих одно и то же имя, но различающихся типами параметров, к которым они применяются.

Иногда возникает потребность описывать и применять к созданным программистом типам данных операции, по смыслу эквивалентные уже имеющимся в языке.  Когда одинаковые по смыслу операции применяются к операндам различных типов, их вынужденно приходится называть по-разному. Невозможность применять для разных типов функции с одним именем приводит к необходимости выдумывать различные имена для одного и того же, что создаёт путаницу, а может и приводить к ошибкам.

Изображение выглядит как текст, снимок экрана, экран

Автоматически созданное описание

Изображение выглядит как текст, внутренний, монитор, снимок экрана

Автоматически созданное описание

Перегрузка операторов тесно связана с перегрузкой методов. Для перегрузки оператора служит ключевое слово **operator**, определяющее операторный метод, который, в свою очередь, определяет действие оператора относительно своего класса.

4. Как используется ключевое слово operator?

*Ответ:* **Оператор — это символ, который сообщает компилятору о необходимости выполнения некоторых математических или логических действий.**

Перегрузка операторов тесно связана с перегрузкой методов. Для перегрузки оператора служит ключевое слово **operator**, определяющее операторный метод, который, в свою очередь, **определяет действие оператора относительно своего класса.**

**Существуют две формы операторных методов (operator):** одна - для **унарных** операторов, другая - для **бинарных**. Ниже приведена общая форма для каждой разновидности этих методов:

Изображение выглядит как текст

Автоматически созданное описание

**Здесь вместо op подставляется перегружаемый оператор**, например + или /, а ***возвращаемый\_тип* обозначает конкретный тип значения, возвращаемого указанной операцией**. Это значение может быть любого типа, но зачастую оно указывается такого же типа, как и у класса, для которого перегружается оператор.

Изображение выглядит как текст

Автоматически созданное описание

5. Какие операции нельзя перегружать в C#?

*Ответ:*

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

6. Можно ли перегрузкой отменить очередность выполнения операции?

*Ответ:* При перегрузке операторов также следует помнить, что **мы не можем изменить приоритет оператора или его ассоциативность**, **мы не можем создать новый оператор** или изменить логику операторов в типах, который есть по умолчанию в .NET.

7. Истинно ли следующее утверждение: операция >= может быть перегружена.

*Ответ:* **Истина**. Эта операция сравнения может быть перегружена. **C# требует совместной перегрузки подобных операций, т.е <= и >=.**

Изображение выглядит как текст

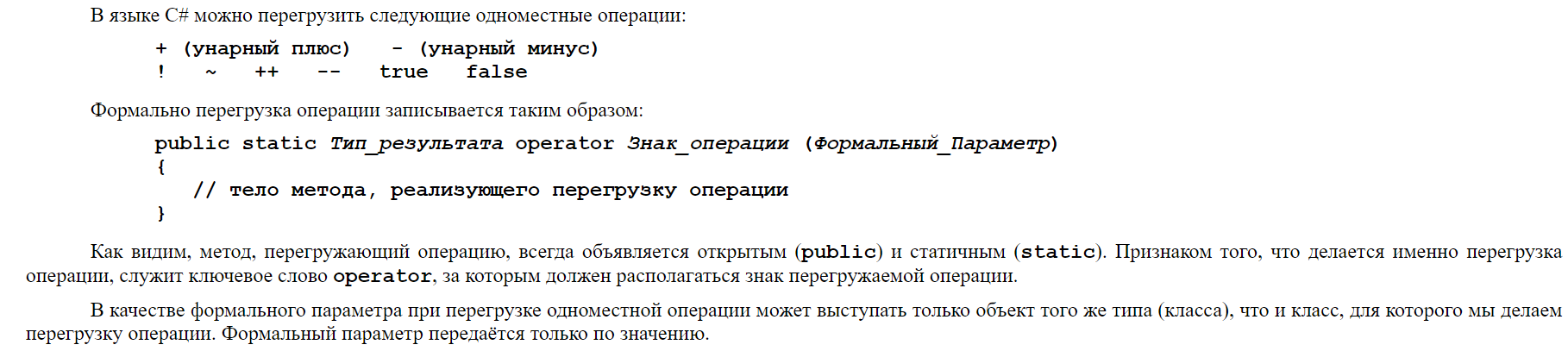
Автоматически созданное описание

8. Сколько аргументов требуется для определения перегруженной унарной операции?

*Ответ:* Унарные операторы перегружаются таким же образом, как и бинарные. Главное отличие заключается, конечно, в том, что **у них имеется лишь один операнд.** **При перегрузке унарной операции функция-оператор не имеет параметров**.

Изображение выглядит как текст

Автоматически созданное описание



Изображение выглядит как текст

Автоматически созданное описание

9. Можно ли перегружать операцию []?

*Ответ:* **Нет**, но функционал этого оператора предоставляют **индексаторы.**

Изображение выглядит как текст

Автоматически созданное описание

В теле индексатора определены **два аксессора** (т.е. **средства доступа к данным**): **get** и **set**. Аксессор подобен методу, за исключением того, что в нем не объявляется тип возвращаемого значения или параметры. **Аксессоры вызываются автоматически при использовании индексатора, и оба получают индекс в качестве параметра.**

**Следует особо подчеркнуть, что индексатор совсем не обязательно должен оперировать массивом**. Его основное **назначение** — **предоставить пользователю функциональные возможности, аналогичные массиву.**

Изображение выглядит как текст

Автоматически созданное описание

где

* *type* – тип элемента индексатора;
* *index* – индекс или позиция элемента в массиве, по которому осуществляется доступ к элементу;
* *type\_index* – тип индекса (позиции) индексатора. Как правило это тип int. Однако, допускается наличие других типов (например double, char);
* *this* – ссылка на данный класс, в котором реализуется индексатор;
* *get*, *set* – аксессоры соответственно для чтения (get) значения элемента и записи (set) значения в элемент.

10. Можно ли перегружать операцию ->?

*Ответ:* Оператор относится к категории **разыменование и адресация**.

**Операция -> не подлежит перегрузке.**

Изображение выглядит как стол

Автоматически созданное описание

11. Приведите пример оператора приведения типа

*Ответ:* Иногда объект определенного класса требуется использовать в выражении, включающем в себя данные других типов. В одних случаях для этой цели оказывается пригодной перегрузка одного или более операторов, а в других случаях — **обыкновенное преобразование типа класса в целевой тип**. Для подобных ситуаций в C# предусмотрена специальная разновидность операторного метода, называемая **оператором преобразования**. **Такой оператор преобразует объект исходного класса в другой тип.**

Изображение выглядит как текст

Автоматически созданное описание

В языке C# предусмотрены два ключевых слова, **explicit** и

**implicit**, которые можно использовать для управления тем, как типы должны реагировать на попытку преобразования.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

class A{}

class B{}

class C{

public static implicit operator A(C c){return new A();}

public static explicit operator B(C c){return new B();}

}

//............

C c=new C();

A a1=c; //Нормально - есть неявный оператор A

A a2=(A)c; //Тоже нормально - неявный оператор может вызываться явно

B b1=c; //Ошибка - нет неявного ператора пробразования C->B

B b2=(B)c; //Нормально - есть явное преобразование C->B

12. Что такое метод расширения? Как и где его можно использовать?

*Ответ:* **Методы расширения позволяют "добавлять" методы в существующие типы без создания нового производного типа, перекомпиляции и иного изменения первоначального типа**. Методы расширения представляют собой разновидность статического метода, но вызываются так же, как методы экземпляра в расширенном типе.

**Методы расширения (extension methods) позволяют добавлять новые методы в уже существующие типы без создания нового производного класса**. Эта функциональность бывает особенно полезна, когда нам хочется добавить в некоторый тип новый метод, но сам тип (класс или структуру) мы изменить не можем, поскольку у нас нет доступа к исходному коду.

**Для того, чтобы создать метод расширения, вначале надо создать статический класс, который и будет содержать этот метод**. В данном случае это класс StringExtension. Затем объявляем статический метод. Суть нашего метода расширения - подсчет количества определенных символов в строке.

class Program

{

    static void Main(string[] args)

    {

        string s = "Привет мир";

        char c = 'и';

        int i = s.CharCount(c);

        Console.WriteLine(i);

        Console.Read();

    }

}

public static class StringExtension

{

    public static int CharCount(this string str, char c)

    {

        int counter = 0;

        for (int i = 0; i<str.Length; i++)

        {

            if (str[i] == c)

                counter++;

        }

        return counter;

    }

}

**Собственно метод расширения - это обычный статический метод, который в качестве первого параметра всегда принимает такую конструкцию: this имя\_типа название\_параметра**, то есть в нашем случае this string str. Так как наш метод будет относиться к типу string, то мы и используем данный тип.

**Применение методов расширения очень удобно, но при этом надо помнить, что метод расширения никогда не будет вызван, если он имеет ту же сигнатуру, что и метод, изначально определенный в типе.**

Также следует учитывать, что методы расширения действуют на уровне пространства имен. То есть, если добавить в проект другое пространство имен, то метод не будет применяться к строкам, и в этом случае надо будет подключить пространство имен метода через директиву using.

Пример.

*Например, вы могли бы знать, является ли определенная строка числом или нет. Обычный подход заключается в том, чтобы определить функцию, а затем вызывать ее каждый раз, и как только вы получите много таких функций, вы поместите их вместе в служебный класс, например:*

Изображение выглядит как текст

Автоматически созданное описание

НО!

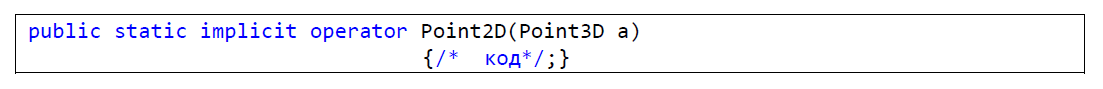
Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

13. Пусть дан фрагмент кода определения оператора преобразования типа. Определить форму преобразования.



*Ответ:* **Форма преобразования неявная, на что указываем слово implicit.**

14. Выберите верное утверждение. Метод расширения может:

1) получать доступ к public членам расширяемого класса   
2) получать доступ к protected членам расширяемого класса   
3) получать доступ к internal членам расширяемого класса   
4) быть объявлен в любом классе   
5) быть без параметров

Изображение выглядит как текст

Автоматически созданное описание

15. Выберите из списка неверное правило перегрузки операторов для C#.

1) префиксные операции ++ и – – перегружаются парами   
2) операции сравнения перегружаются парами: == и != ; < и >;<= и >=   
3) перегруженные операции обязаны возвращать значения   
4) должны объявляться как protected   
5) true и false можно перегружать

Изображение выглядит как текст

Автоматически созданное описание

