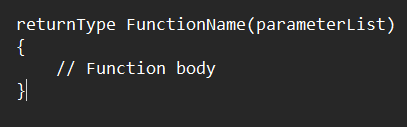
**Функции**

Функции в C# — это основные блоки кода, которые выполняют определенные задачи. Они помогают структурировать код, делают его более читаемым и поддерживаемым, а также позволяют повторно использовать логику в различных частях программы.

1. Определение функции

Функция в C# определяется с помощью ключевого слова returnType, имени функции и списка параметров.

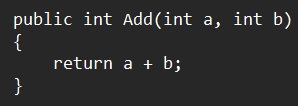


- returnType — тип данных, который функция возвращает. Если функция ничего не возвращает, используется ключевое слово void.

- FunctionName — имя функции.

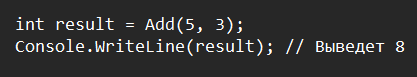
- parameterList — список параметров, передаваемых функции (необязательный).

**Пример простой функции**

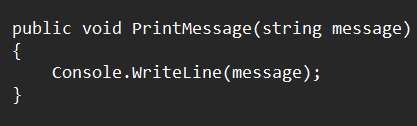


Эта функция принимает два целых числа a и b, складывает их и возвращает результат.

**Вызов функции**

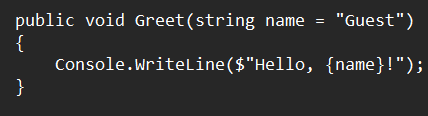


**Функция без возвращаемого значения**



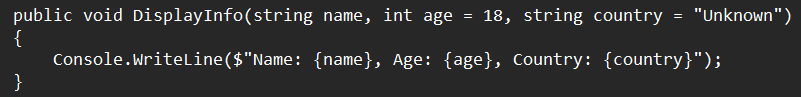
Эта функция выводит строку на консоль и не возвращает значения (void).

**Функция с параметрами по умолчанию**

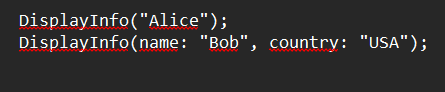


Если при вызове функции параметр name не указан, будет использовано значение по умолчанию "Guest".

**Функция с опциональными и именованными параметрами**

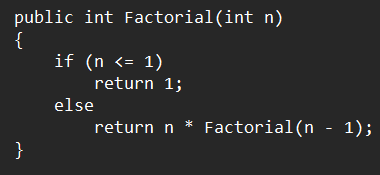
****

Можно вызывать функцию, указывая только необходимые параметры или использовать именованные параметры для большей ясности.



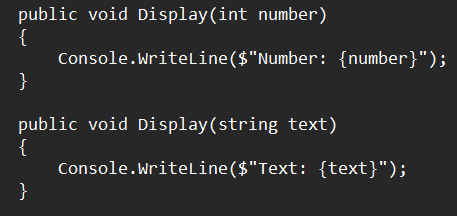
**Рекурсивная функция**

Рекурсивная функция вызывает саму себя. Пример функции для вычисления факториала:



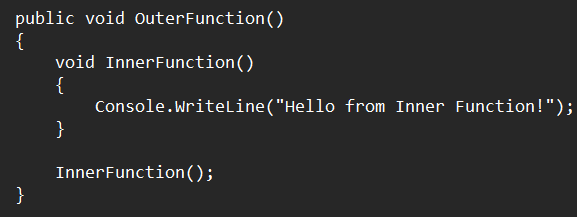
**Перегрузка функций**

C# поддерживает перегрузку функций, что означает, что в одном классе можно иметь несколько функций с одним и тем же именем, но с разными параметрами.



**Локальные функции**

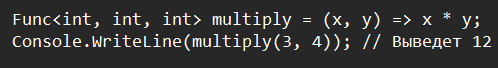
В C# можно определять функции внутри других функций:



Локальные функции полезны, когда функция нужна только внутри другого метода и не должна быть доступна за его пределами.

**Анонимные функции и лямбда-выражения**

Анонимные функции — это функции без имени, которые можно использовать для кратковременных задач.



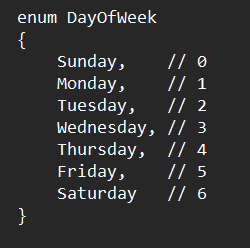
Лямбда-выражения часто используются с делегатами и функциональными конструкциями. (пример 1 - 8).

**Перечисления Enum**

Перечисления (enum) в C# — это специальные типы данных, представляющие собой набор именованных констант. Они позволяют определять набор значений, которые могут быть использованы в программе, что делает код более понятным и легким для поддержки.

**Определение перечисления (enum)**

Перечисление создается с помощью ключевого слова enum. Каждому элементу перечисления автоматически присваивается целочисленное значение, начиная с 0, если явно не указано другое.



**Использование перечисления**

Перечисление используется для хранения значений, которые ограничены определенным набором. В примере ниже мы создадим консольное приложение, которое использует перечисление для выбора дня недели. (пример 9).

**Перечисление с пользовательскими значениями**

По умолчанию элементы перечисления имеют целочисленные значения, начинающиеся с 0. Однако вы можете задать свои значения для элементов. (пример 10)

**Комбинирование перечислений с атрибутом [Flags]**

Атрибут [Flags] позволяет комбинировать значения перечисления с помощью битовых операций, как это делается с флагами. (пример 11).

**Приведение типов и работа с перечислениями**

В C# можно явно приводить значения из enum в int и обратно, что бывает полезно для выполнения различных операций. (пример 12).

Перечисления (enum) в C# позволяют работать с фиксированными наборами значений, обеспечивая более безопасный и понятный код. Их можно использовать для упрощения логики программы, улучшения читабельности кода, а также для работы с флагами и комбинациями значений.

**Классы и объекты. Введение в ООП**

Классы и объекты являются основными элементами объектно-ориентированного программирования (ООП) в C#. Классы позволяют определять структуру и поведение объектов, а объекты являются экземплярами классов.

Класс в C# — это шаблон или чертеж для создания объектов. Он определяет свойства (переменные) и методы (функции), которые будут использоваться объектами.

Объект — это экземпляр класса. Когда создается объект, выделяется память для всех его переменных и методов, и эти переменные и методы становятся доступными для использования.

**Создание класса**

Определение класса начинается с ключевого слова class, за которым следует имя класса и его тело, заключенное в фигурные скобки { }. (пример 13).

**Создание объектов и использование класса**

Объект создается с помощью ключевого слова new, за которым следует вызов конструктора класса. (пример 14).

**Конструкторы классов**

Конструктор — это специальный метод, который вызывается при создании объекта класса. Он используется для инициализации полей объекта. Конструкторы могут быть перегружены, что позволяет создавать несколько вариантов инициализации объекта.

**Инкапсуляция**

Инкапсуляция — это принцип ООП, который предполагает сокрытие деталей реализации и предоставление доступа к данным только через методы класса. В C# это достигается с помощью модификаторов доступа, таких как private, public, protected. (пример 16).

**Свойства (Properties)**

Свойства — это удобный способ для доступа к приватным полям класса. Они выглядят как обычные поля, но предоставляют возможности контроля за доступом к данным. (пример 17).

**Статические поля и методы**

Статические поля и методы принадлежат самому классу, а не конкретному объекту. Они могут быть вызваны без создания экземпляра класса. (пример 18).