Методы

№ урока: 8 **Курс:** C# Starter

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Рассмотрение работы методов.

Изучив материал данного занятия, учащийся сможет:

- Понимать работу методов.
- Понимать работу рекурсии.
- Понимать перегрузку методов.
- Использовать методы с опциональными и именованными параметрами.

Содержание урока

1. Рассмотрение примера: Перегрузки методов.
2. Рассмотрение примеров: Методы с опциональными параметрами.
3. Рассмотрение примеров: Рекурсивный вызов метода.

Резюме

- **Метод** – это именованная часть программы, которая может вызываться из других частей программы столько раз, сколько необходимо.
- **Сигнатура метода** – часть общего объявления метода, позволяющая идентифицировать функцию среди других. В C#, в сигнатуру метода входит Идентификатор метода, тип и количество формальных аргументов. В CIL, в сигнатуру может входить также и возвращаемое значение метода, так как в CIL, перегруженные методы могут отличаться типами возвращаемых значений, что недопустимо в C#.
- **Перегрузка методов** – создание одноименного метода, но с другим типом и/или количеством аргументов. Перегруженные метода могут отличаться наличием `ref`/ `out` аргументов, но только в том случае, если они имеются только у одного метода. Если единственное различие между методами заключается в том, что один метод принимает аргумент `ref`, а другой – `out` , они не могут быть перегружены
- Старайтесь ограничивать количество параметров примерно семью.
- Минимизируйте число возвратов из каждого метода. Тяжело понять логику метода, когда при анализе нижних строк приходится помнить о возможных выходах в верхних строках.
- **Рекурсия** – вызов функции (процедуры) из неё же самой, непосредственно (простая рекурсия) или через другие функции (сложная рекурсия). Например, функция A вызывает функцию B, а функция B – функцию A. Количество вложенных вызовов функции или процедуры называется глубиной рекурсии.
- Одна из ключевых задач рекурсивного метода – предотвращение бесконечной рекурсии.
- Реализация рекурсивных вызовов функций, опирается на механизм стека вызовов — адрес возврата и локальные переменные функции записываются в стек, благодаря чему каждый следующий рекурсивный вызов этой функции пользуется своим набором локальных переменных и за счёт этого работает корректно
- На каждый рекурсивный вызов требуется некоторое количество оперативной памяти, и при чрезмерно большой глубине рекурсии может наступить переполнение стека вызовов. Вследствие этого, обычно рекомендуется избегать рекурсивных программ, которые приводят (или в некоторых условиях могут приводить) к слишком большой глубине рекурсии.
- Не используйте рекурсию для вычисления факториалов и чисел Фибоначчи.

CyberBionic systematics

CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

- Так как, методы – это конструкции для выполнения действий, рекомендуется их называть глагольными фразами или глаголами.
- Старайтесь именовать методы в соответствии с задачами, которые они выполняют, а не в соответствии с деталями реализации.
- Для именования методов в C#, рекомендуется использовать соглашение Pascal Casing. Чтобы выделить слова в идентификаторе, первые буквы каждого слова (кроме первого) сделайте заглавными. Например: `WriteLine`, `GetType`.
- Язык C# чувствительный к регистру (case sensitivity) Например: `GetType` и `getType` – это разные имена.
- Не используйте символы подчеркивания, дефисы и любые другие неалфавитно-цифровые символы для разделения слов в идентификаторе.
- Описывайте все, что метод выполняет.
- Избегайте невыразительных и неоднозначных глаголов или фраз.
- Для именования метода-функции рекомендуется использовать описание возвращаемого значения. Например: `CuttentColor()`
- В C# 4.0 появились именованные и необязательные аргументы. Именованные аргументы позволяют указать аргумент для определенного параметра, связав аргумент с именем параметра, а не с позицией параметра в списке параметров. Необязательные аргументы позволяют опускать аргументы для некоторых параметров. Оба подхода можно применять к методам, индексаторам, конструкторам и делегатам.
- При использовании именованных и необязательных аргументов аргументы вычисляются в том порядке, в котором они указаны в списке аргументов, а не в списке параметров.
- При совместном использовании именованных и необязательных параметров разработчик может задавать аргументы лишь для некоторых параметров из списка необязательных параметров.
- Именованные аргументы освобождают разработчика от необходимости помнить или уточнять порядок следования параметров при вызове методов. Параметр для каждого из аргументов можно задать с помощью имени параметра.
- Именованный аргумент можно поместить после позиционных аргументов
- При создании метода, можно указать, что параметры являются обязательными или необязательными. При каждом вызове необходимо указывать аргументы для всех обязательных параметров, но можно опустить аргументы для необязательных параметров.
- В рамках определения каждого необязательного параметра задается его значение по умолчанию. Если для параметра не передается аргумент, используется значение по умолчанию. Значения по умолчанию должны быть константами.
- Необязательные параметры определяются в конце списка параметров после всех обязательных параметров. Если вызывающий объект задает аргумент для какого-либо из последующих необязательных параметров, он должен задать аргументы для всех предшествующих необязательных параметров. Разделенные запятыми пустые позиции в списке аргументов не поддерживаются.
- Метод Main является точкой входа консольного приложения C# или приложения Windows. (Для библиотек и служб не требуется метод `Main` в качестве точки входа). При запуске приложения метод Main является первым вызываемым методом.
- В программе C# возможна только одна точка входа. Если в наличие имеется больше одного класса, который имеет метод `Main`, то необходимо скомпилировать программу с параметром компилятора /main, чтобы указать, какой метод Main нужно использовать в качестве точки входа.
- Метод `Main` является точкой входа EXE-программы, в которой начинается и заканчивается управление программой.
- Метод `Main` объявлен внутри класса или структуры. Метод `Main` должен быть статичным и не иметь атрибута `public`.
- `Main` может иметь возвращаемый тип либо `void`, либо `int`. Метод `Main` может быть объявлен с параметром `string []`, который содержит аргументы командной строки, или без него. При использовании Visual Studio для создания приложений Windows Forms, можно

CyberBionic systematics

CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

добавить параметр вручную или использовать класс `Environment` для получения аргументов командной строки. Параметры считываются как аргументы нулевого индекса командной строки

- Если значение, возвращаемое методом Main, не используется, то указание в качестве возвращаемого типа `void` несколько упрощает код. Однако возврат целого значения позволяет программе передавать информацию о своем состоянии другим программам и скриптам, которые вызывают исполняемый файл.
- Нулевое возвращаемое значение из метода `Main` указывает на успешное выполнение программы.

Закрепление материала

- Что такое метод?
- Чем отличаются функции и процедуры?
- Что такое перегрузка метода?
- Что такое рекурсия и какие виды рекурсии вы знаете?
- Что такое опциональные параметры?
- Что такое метод предикат?
- Какие правила именования применимы к методам?
- Можно ли выполнить перегрузку метода Main?

Дополнительное задание

Задание
Используя Visual Studio, создайте проект по шаблону Console Application.
Создайте метод с именем `Calculate`, который принимает в качестве параметров три целочисленных значения и возвращает значение каждого аргумента деленного на 5.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные конструкции и понятия, рассмотренные на уроке.

Задание 2
Используя Visual Studio, создайте проект по шаблону Console Application.
Представьте, что вы реализуете программу для банка, которая помогает определить, погасил ли клиент кредит или нет. Допустим, ежемесячная сумма платежа должна составлять 100 грн. Клиент должен выполнить 7 платежей, но может платить реже большими суммами. Т.е., может двумя платежами по 300 и 400 грн. закрыть весь долг.
Создайте метод, который будет в качестве аргумента принимать сумму платежа, введенную экономистом банка. Метод выводит на экран информацию о состоянии кредита (сумма задолженности, сумма переплаты, сообщение об отсутствии долга).

Задание 3
Имеется N клиентов, которым компания производитель должна доставить товар. Сколько существует возможных маршрутов доставки товара, с учетом того, что товар будет доставлять одна машина?
Используя Visual Studio, создайте проект по шаблону Console Application.
Напишите программу, которая будет рассчитывать и выводить на экран количество возможных вариантов доставки товара. Для решения задачи, используйте факториал N!, рассчитываемый с помощью рекурсии. Объясните, почему не рекомендуется использовать рекурсию для расчета факториала. Укажите слабые места данного подхода.

Задание 4
Используя Visual Studio, создайте проект по шаблону WindowsForms.
Напишите программу, в который пользователь в textBox1 и textBox2 вводит данные и выбирает при помощи нескольких radioButton тип операции, которую требуется выполнить: остаток от деления,

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

возведение в степень, конкатенация, деление. Результат выводится в textBox3 при нажатии на кнопку «Вычислить»

Задание 5
Зайдите на сайт MSDN.
Используя поисковые механизмы MSDN, найдите самостоятельно описание темы по каждому примеру, который был рассмотрен на уроке, так, как это представлено ниже, в разделе «Рекомендуемые ресурсы», описания данного урока. Сохраните ссылки и дайте им короткое описание.

Рекомендуемые ресурсы

MSDN: Методы (Руководство по программированию на C#)
http://msdn.microsoft.com/ru-ru/library/ms173114(v=vs.90).aspx

MSDN: Именованные и необязательные аргументы (Руководство по программированию на C#)
http://msdn.microsoft.com/ru-ru/library/dd264739.aspx

MSDN: Передача параметров (Руководство по программированию в C#)
http://msdn.microsoft.com/ru-ru/library/0f66670z.aspx

`Main`() и аргументы командной строки (Руководство по программированию на C#)
http://msdn.microsoft.com/ru-ru/library/acy3edy3.aspx

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com