Методы

№ урока: 7 **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 отличие процедуры от функции.

Содержание урока

1. Обзор методов.
2. Различия между процедурами и функциями.
3. Рассмотрение примеров: Работа методов.
4. Рассмотрение управляющей структуры `return`.
5. Использование сторожевых операторов.
6. Рассмотрение примера: Использование сторожевого оператора, для защиты номинального варианта.
7. Рассмотрение примера: Использование `ref` и `out` аргументов в методах

Резюме

- **Метод** – это именованная часть программы, которая может вызываться из других частей программы столько раз, сколько необходимо.
- **Метод – это функция или процедура, выполняющая одну задачу**.
- О функциях и процедурах. В некоторых языках программирования (например, в Паскале) функции и процедуры (подпрограммы, не возвращающие значения) чётко разграничены синтаксисом языка. В языке C#, – процедуры являются частным случаем (подмножеством) функций, возвращающими значение типа void — пустое значение.
- **Функция** – это метод, возвращающий значение, **процедура** – это метод который значения не возвращает. Все методы в C#, технически являются функциями, но логически, методы которые возвращают – `void`, являются процедурами.
- Определение метода задает имена и типы любых необходимых параметров. Когда код вызова вызывает метод, он передает в него конкретные значения, называемые аргументами, для каждого параметра. Аргументы должны быть совместимыми с типом параметра, но имя аргумента (если таковое имеется), используемое в коде вызова, не обязательно должно совпадать с именем параметра, определенного в методе.
- Принято различать сигнатуру вызова и сигнатуру реализации метода. Сигнатура вызова обычно составляется по синтаксической конструкции вызова метода с учётом имени данной функции, последовательности фактических типов аргументов в вызове и типе результата. В сигнатуре реализации обычно участвуют некоторые элементы из синтаксической конструкции объявления функции: спецификатор области видимости функции, её имя и последовательность формальных типов аргументов.
- **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#.
- **Семантика метода** – это описание того, что данный метод делает. Семантика метода включает в себя описание того, что является результатом вычисления метода, как и от чего этот результат

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

зависит. Обычно результат выполнения зависит от значений аргументов метода. К понятию семантики можно отнести как тело метода и возвращаемое значение, так и `out` аргументы.

- Вызов метода объекта очень похож на обращение к полю. После имени объекта ставится точка, затем имя метода и скобки. В скобках перечисляются аргументы, разделенные запятыми.
- Самая важная причина создания методов – это снижение сложности программ.
- Одна из главных задач, которые решают методы, – избежать дублирования кода. Или другими словами, методы открывают возможность повторного использования кода.
- Методы реализуют идею сокрытия информации. Один раз, создав метод, вы его используете, не думая о его внутренней работе.
- Использование методов приводит к минимизации кода, облегчению сопровождения программ и снижению числа ошибок.
- Использование методов, формирует понятную промежуточную абстракцию.
- Выделение фрагмента кода в отдельный, удачно названый метод, является одним из способов документирования целей данного метода.
- Методы позволяют выполнять оптимизацию кода в одном месте. Это облегчает профилирование кода, направленное на определение неэффективных фрагментов.
- **Методы-предикаты** – методы, которые возвращают логическое значение, называют.
- **Связность** (как мера независимости или самостоятельности) на уровне методов, характеризует **соответствие выполняемых в методе операций единой цели**. Некоторые программисты связность, называют – силой метода (strength).
- Очень важно, чтобы каждый метод эффективно решал только одну задачу и больше ничего не делал. Например: метод `Console.WriteLine("Hello")` – имеет четко определенную цель.
- Рекомендуемый вид связности: Функциональная связность – самый сильный вид связности. В этом случае метод выполняет только одну операцию.
- Приемлемый вид связности: Последовательная связность (sequentialcohesion), вид связности, когда метод содержит наборы операций, которые выполняются в строго определенном порядке, используют данные предыдущих этапов и не формируют в целом единую функцию.
- Приемлемый вид связности: Коммуникационная связность (communicationalcohesion), вид связности, когда выполняемые в методе операции используют одни и те же данные и не связанны между собой другим образом. В таком случае рекомендуется разделить операции на два метода.
- Приемлемый вид связности: Временная связность (temporalcohesion), вид связности, когда метод пытается объединить в себе операции, которые должны выполняться в один интервал времени.
- Плохой вид связности: Процедурная связность (proceduralcohesion), вид связности, когда операции в методе выполняются в определенном порядке. Для достижения лучшей связности рекомендуется поместить разные операции в отдельные методы.
- Плохой вид связности: Логическая связность (logicalcohesion), вид связности, когда метод включает несколько операций, а выбор операции осуществляется на основании передаваемого в качестве аргумента флага. Вид связности называется логическим, потому что, операции метода объединены только управляющей логикой метода: оператором `if` или рядом блоков `case`.
- Плохой вид связности: Случайная связность (coincidentalcohesion), вид связности, когда каких-либо отношений между выполняемыми в методе операциями нет. Этот вариант еще называют – «отсутствием связности» или «хаотичной связностью».
- Стремитесь создавать методы с функциональной связностью – это возможно почти всегда.
- Оператор `return` – это управляющая структура, которая позволяет программе в нужный момент завершить работу метода. В результате метод завершается через нормальный канал выхода, возвращая управление вызывающему методу.
- Используйте `return`, если это повышает читабельность метода.
- Используйте `return`, как сторожевой оператор досрочного выхода.
- Методы могут возвращать значения вызывающим их объектам. Если тип возвращаемого значения, указываемый перед именем метода, не равен `void`, для возвращения значения используется ключевое слово `return`.

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

- В результате выполнения инструкции с ключевым словом `return`, после которого указано значение нужного типа, вызвавшему метод объекту будет возвращено это значение.
- Ключевое слово `return` останавливает выполнение метода.
- Если тип возвращаемого значения `void`, инструкцию `return` без значения все равно можно использовать для завершения выполнения метода.
- Если ключевое слово `return` отсутствует, выполнение метода завершится, когда будет достигнут конец его блока кода.
- Для возврата значений методами с типом возвращаемого значения отличным от `void` необходимо обязательно использовать ключевое слово `return`.
- Чтобы использовать возвращаемое методом значение в вызываемом методе, вызов метода можно поместить в любое место кода, где требуется значение соответствующего типа.
- Возвращаемое значение метода можно присвоить переменной.
- Методы, которые возвращают логическое значение, называют методами предикатами.
- Параметры принято разделять на формальные параметры и фактические параметры.
- Рекомендуется передавать параметры в метод в следующей последовательности: сначала входные параметры `in`, потом изменяемые `ref` и в конце выходные `out`.
- Язык C# не поддерживает использования ключевого слова `in` для входных параметров, все параметры, кроме `ref` и `out` параметров, являются `in`-параметрами по умолчанию.
- Ключевое слово `ref` используется для передачи аргументов в метод по ссылке. В результате все изменения параметра в методе будут отражены в переменной при передаче элемента управления обратно в вызывающий метод.
- Для работы с параметром `ref` вызывающий метод должен явно использовать ключевое слово `ref` при передаче аргумента в метод.
- Аргумент, передаваемый в параметр `ref`, сначала следует инициализировать. В этом заключается отличие от `out`, аргументы которого не требуют явной инициализации перед передачей.
- Ключевое слово `out` используется для передачи аргументов по ссылке. Оно похоже на ключевое слово `ref`, за исключением того, что `ref` требует инициализации переменной перед ее передачей. Для работы с параметром `out` определение метода и вызывающий метод должны явно использовать ключевое слово `out`.
- Несмотря на то что переменные, передаваемые в качестве аргументов `out`, могут не инициализироваться перед передачей, вызываемый метод должен присвоить значение перед возвратом метода.
- Объявление метода `out` используется тогда, когда необходимо, чтобы метод возвращал несколько значений
- Ключевые слова `ref` и `out` приводят к разным результатам во время выполнения, однако во время компиляции они не считаются частью сигнатуры метода. Поэтому, если единственное различие между методами заключается в том, что один метод принимает аргумент `ref`, а другой – `out`, они не могут быть перегружены.
- Опасно использовать передаваемые в метод параметры, как рабочие переменные. Для такой цели создайте локальные переменные.
- 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.
- 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.

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

- Описывайте все, что метод выполняет.
- Избегайте невыразительных и неоднозначных глаголов или фраз.
- Для именования метода-функции рекомендуется использовать описание возвращаемого значения. Например: `CuttentColor()`
- Все числовые типы содержат метод тип.`Parse()` , который преобразует строковое или численное представление переменной в эквивалентный перечислимый тип, по своему принципу действия эквивалентен `Convert.To…` ()

Закрепление материала

- Что такое метод?
- Чем отличаются функции и процедуры?
- Что делает оператор `return`?
- Что такое сигнатура метода?
- Что такое семантика метода?
- В чем разница между передачей параметров по значению и по ссылке?
- В чем разница между `ref` и `out`?
- Что такое метод предикат?
- Какие правила именования применимы к методам?

Дополнительное задание

Задание
Используя Visual Studio, создайте проект по шаблону Console Application.
Создайте метод с именем `Calculate`, который принимает в качестве параметров три целочисленных аргумента и выводит на экран среднее арифметическое значений аргументов.

Самостоятельная деятельность учащегося

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

Задание 2
Используя Visual Studio, создайте проект по шаблону Console Application.
Создайте четыре метода для выполнения арифметических операций, с именами: `Add` – сложение, `Sub` – вычитание, `Mul` – умножение, `Div` – деление. Каждый метод должен принимать два целочисленных аргумента и выводить на экран результат выполнения арифметической операции соответствующей имени метода. Метод деления `Div`, должен выполнять проверку попытки деления на ноль.
Требуется предоставить пользователю возможность вводить с клавиатуры значения операндов и знак арифметической операции, для выполнения вычислений.

Задание 3
Используя Visual Studio, создайте проект по шаблону Console Application.
Напишите программу, которая будет выполнять конвертирование валют.
Пользователь вводит:
сумму денег в определенной валюте.
курс для конвертации в другую валюту.
Организуйте вывод результата операции конвертирования валюты на экран.

Задание 4
Используя Visual Studio, создайте проект по шаблону Console Application.
Напишите метод, который будет определять:
1) является ли введенное число положительным или отрицательным.
2) Является ли оно простым (используйте технику перебора значений).

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

(**Простое число** – это натуральное число, которое делится на 1 и само на себя. Чтобы определить простое число или нет, следует найти все его целые делители. Если делителей больше 2-х, значит оно не простое.)
3) Делится ли на 2, 5, 3, 6, 9 без остатка.

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

MSDN: Оператор return (Справочник по C#)
http://msdn.microsoft.com/ru-ru/library/1h3swy84(v=VS.90).aspx

MSDN: ref (Справочник по C#)
http://msdn.microsoft.com/ru-ru/library/14akc2c7.aspx

MSDN: Модификатор параметров out (справочник по C#)
http://msdn.microsoft.com/ru-ru/library/ee332485.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