Переменные и типы данных

№ урока: 3 **Курс:** C# Starter

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Рассмотрение понятия переменной, константы и типа данных.
Рассмотрение арифметических операторов и операторов сравнения.

Изучив материал данного занятия, учащийся сможет:

- Применять переменные и константы.
- Понимать, когда и какие типы использовать при создании переменной.
- Выполнять арифметические операции над значениями переменных.
- Сравнивать значения переменных.
- Выполнять форматирование строк.

Содержание урока

1. Константы.
2. Преобразование типов (Casting).
3. Арифметические операторы.
4. Математические функции.
5. Инкремент и Декремент.
6. Операции сравнения.
7. Присвоение с действием.
8. Локальные области видимости.
9. Ключевые слова в качестве идентификаторов.
10. Правила именования локальных переменных в .Net (Code convention).
11. Понятие переноса (`carry / auxiliarycarry`) и переполнения (`overflow`).
12. Проверка переполнения (`checked`).
13. Отсутствие проверки переполнения (unchecked)
14. Комбинирование (`checked/unchecked`)
15. Конкатенация
16. Форматирование строк
17. Флаги форматирования строк.
18. Использование оператора `sizeof()`.
19. Неявно типизированные локальные переменные.
20. Сравнение значений разных типов.

Резюме

- Переменная (`Variable`) – это область памяти, которая хранит в себе некоторое значение, которое можно изменить.
- Инициализация переменной – это первое присвоение ей значения. Все последующие присвоения новых значений этой переменной, не считаются инициализацией.
- Технически, имена переменных могут начинаться со знака «_» – нижнее подчеркивание и с любого алфавитного символа. (Имена не могут начинаться с цифр и других символов.)
- Для именования локальных переменных в C#, рекомендуется использовать соглашение camel Casing. Чтобы выделить слова в идентификаторе, первые буквы каждого слова (кроме первого) сделайте заглавными. Например: `myAge`, `myName`.
- Язык C# чувствительный к регистру (**casesensitivity**) Например: `MyName` и `myName` – это разные имена.

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

- Не используйте символы подчеркивания, дефисы и любые другие неалфавитно-цифровые символы для разделения слов в идентификаторе.
- Не используйте венгерскую нотацию. Суть венгерской нотации сводится к тому, что имена идентификаторов предваряются заранее оговорёнными префиксами, состоящими из одного или нескольких символов. Например: `string sClientName; int iSize;`
- Имена переменных должны быть понятны и передавать смысл каждого элемента.
- В редких случаях, если у идентификатора нет точного семантического значения, используйте общие названия. Например: `value`, `item`.
- При создании переменной, используйте название-псевдоним, когда это возможно, а не полное имя типа.
- Джеффри Рихтер никогда не использует псевдонимов типов, считая, что их использование приводит к запутанному коду (это не рекомендация, а мнение Джеффри).
- Константа (**Constant**) – это область памяти, которая хранит в себе некоторое значение, которое нельзя изменить.
- Правила использования констант:
  1) Константам необходимо присваивать значение непосредственно в месте создания;
  2) Попытка присвоения константе нового значения приводит к ошибке уровня компиляции;
- Преобразование типа (Casting или Type conversion) – это преобразование значения переменной одного типа в значение другого типа. (Преобразование не следует путать с приведением типов – Cast) Выделяют явное (explicit) и неявное (implicit) преобразование типов.
- Неявное преобразование типа (безопасное) – преобразование меньшего типа в больший или целого типа в вещественный. Является безопасным, так как не происходит потеря точности.
- Явное преобразование типа (опасное) – преобразование большего типа в меньший или вещественного типа в целый. Является опасным, так как происходит потеря точности результата без округления.
- Возможно неявное преобразование значения константы большего типа в меньший, при инициализации переменной значением константы, если значение константы не превышает максимально допустимого значения переменной.
- Возможно явное преобразование значения константы вещественного типа в целый тип, при инициализации переменной значением константы, если значение константы не превышает максимально допустимого значения переменной.
- Если значение константы превышает максимально допустимый диапазон значения переменной, такое преобразование невозможно и приведет к ошибке.
- Оператор присвоения (=) сохраняет значение своего правого операнда в месте хранения (переменной) обозначенной в левом операнде. Операнды должны быть одного типа (или правый операнд должен допускать явное преобразование в тип левого операнда).
- Если **после знака присвоения** идет выражение с вычислением или передачей каких-либо значений, то данная операция **выполняется справа налево**. Для повышения приоритета операции можно использовать круглые скобки ( ).
- **Только четыре операции гарантируют порядок вычислений слева направо: ,, ?:, && и ||**
- Язык C# предоставляет большой набор операторов, которые представляют собой символы, определяющие операции, которые необходимо выполнить с выражением. К операторам, которые выполняют арифметические операции можно отнести операторы:
  + (сложения),
  – (вычитания),
  * (умножения),
  / (деления),
  % (получения остатка от деления)
- Язык C# предоставляет большой набор математических функций для выполнения различных вычислений.
- `Math.Sqrt()` – математическая функция которая извлекает квадратный корень. В аргументных скобках указываем значение числа, из которого хотим извлечь квадратный корень.
- `Math.Pow()` – возведения числа в степень. В аргументных скобках через запятую указываем два аргумента (первый – число, которое хотим возвести в степень, второй – степень, в которую мы хотим возвести число).

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

- **Операции умножения, деления, получения остатка от деления имеют больший приоритет, чем сложения и вычитания**, поэтому выполнятся в первую очередь.
- При получении результата остатка от деления – знак результата не сокращается и соответствует значению первого операнда (делимого).
- Если в правой части выражения выполнялись операции деления между целыми числами, то результат будет приведен компилятором к целому типу, даже если результат записать в переменную вещественного типа или привести все выражение к вещественному типу.
- Оператор **инкремента** (++) увеличивает свой операнд на 1. Оператор инкремента может находиться как перед операндом, так и после него: ++variable или variable++.
- **Префиксная операция увеличения** – результатом выполнения этой операции является использование значения операнда после его увеличения.
- **Постфиксная операция увеличения** – результатом выполнения этой операции является использование значения операнда перед его увеличением.
- Оператор **декремента** (--) уменьшает свой операнд на 1.Оператор декремента может находиться как перед операндом, так и после него: --variable или variable--.
- Префиксная операция декремента – результатом выполнения этой операции является использования значения операнда после его декремента.
- Постфиксная операция декремента – результатом этой операции является использование значения операнда до его декремента.
- К операциям сравнения можно отнести операции:
  > больше,
  >= больше или равно,
  < меньше,
  <= меньше или равно.
- К операциям проверки на равенство можно отнести операции:
  == равно,
  != не равно.
- Результатом выполнения операций сравнения и проверки на равенство неравенство всегда будет либо `false` или `true`.
- Для предопределенных типов значений оператор равенства **(==)** возвращает значение `true`, если значения его операндов совпадают, в противном случае – значение `false`.Для типа `string` оператор **==** сравнивает значения строк.
- Оператор неравенства **(! =)** возвращает значение `false`, если его операнды равны, в противном случае – значение `true`.
- Оператор сравнения "меньше или равно" **(<=)** возвращает значение `true`, если первый операнд меньше или равен второму, в противном случае возвращается значение `false`.
- Оператор сравнения "меньше" **(<)** возвращает значение `true`, если первый операнд меньше второго, в противном случае возвращается значение `false`.
- Оператор сравнения "больше" **(>)** возвращает значение `true`, если первый операнд больше второго, в противном случае возвращается значение `false`.
- Оператор сравнения "больше или равно" **(> =)** возвращает значение `true`, если первый операнд больше или равен второму, в противном случае возвращается значение `false`.
- Все арифметические операции, производимые над двумя значениями типа (`byte, sbyte, short, ushort`) в качестве результата, возвращают значение типа `int`.
- Для типов `int, uint, long и ulong`, не происходит преобразования типа результата арифметических операций.
- **Локальная область** – участок кода, внутри класса или блок, который ограничен фигурными скобками.
- **Область видимости переменной** – часть текста программы, в которой имя можно явно использовать. Чаще всего область видимости совпадает с областью действия.
- Переменная созданная внутри локальной области называется **локальной переменной**, область ее действия – от открывающей скобки локальной области до ее окончания (закрывающей скобки) блока, включая все вложенные локальные области.
- Переменная уровня класса называется **глобальной переменной или полем.**

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 коде можно создавать локальные области и в двух разных локальных областях хранить одноименные переменные.
- Если в коде имеются локальные области, то запрещается хранить одноименные переменные за пределами локальных областей. И наоборот, если за пределами локальных областей уже созданы переменные с каким-то именем, то в локальных областях этого уровня запрещается создавать одноименные переменные.
- **Ключевые слова** – это предварительно определенные зарезервированные идентификаторы, имеющие специальные значения для компилятора. Их нельзя использовать в программе в качестве идентификаторов, если только они не содержат префикс @.
- Символ @, который используется в идентификаторе переменной, указывает компилятору, что это слово необходимо трактовать как идентификатор, а не как ключевое слово C# или его команду.
- Символ @ не является частью идентификатора, поэтому, `@myVariable` – это тоже самое, что и `myVariable`.
- Операторы C# могут выполняться в проверяемом или непроверяемом контексте. В проверяемом контексте арифметическое переполнение вызовет исключение (ошибку).
- В непроверяемом контексте арифметическое переполнение будет проигнорировано, а результат усечен. Для таких действий используются следующие конструкции:
  - ✓ `Checked` – указание проверяемого контекста;
  - ✓ `Unchecked` – указание непроверяемого контекста;
- Проверка переполнений применяется в следующих случаях:
  1) Если используются выражения, использующие предопределенные операторы в целых типах с операциями (++, --, +, -, *, /).
  2) Если выполняются явные числовые преобразования между целыми типами данных.
- **Конкатенация – сцепление строк** или значений переменных типа string, для получения строк большего размера с помощью операции +.
- Для форматирования числовых результатов и вывода их на экран можно использовать метод `Console.Write()` или `Console.WriteLine()`, который вызывает метод `string.Format()`.
- Формат задается с помощью флагов форматирования. Флаг форматирования может иметь следующую форму: Axx, где A – флаг формата (определяет тип формата), а xx – описатель точности (количество отображаемых цифр или десятичных знаков форматированного результата). Например: `Console.WriteLine("{0:F2}", 99.935);`
- Существуют следующие **флаги форматирования** строк:
  - ✓ C или c – валюта (Currency);
  - ✓ D или d – десятичное число (Decimal);
  - ✓ E или e – научный формат (Scientific, exponential)
  - ✓ F или f – формат с фиксированным значением после запятой (Fixed-point)
  - ✓ G или g – общие (General)
  - ✓ N или n – Number (Number)
  - ✓ X или x – шестнадцатеричный формат (Hexadecimal)
  - ✓ P или p –процентный (Percent)
- Оператор `sizeof()` - позволяет получить размер значения в байтах для указанного типа.
- Оператор `sizeof()` можно применять только к типам: (`byte, sbyte, short, ushort, int, uint, long, ulong, float, double, decimal, char, bool`).
- Возвращаемые оператором `sizeof()` значения имеют тип `int`.
- В C# 3.0 появилась возможность в области метода создавать переменные, которые могут иметь неявный тип `var`. Такие переменные называют **неявно типизированными локальными переменными.** Таким способом можно «поручить» компилятору определить тип ваших переменных, если вы не знаете точно результат.
- Правила использования **неявно типизированных локальных переменных:**
  - ✓ Можно создать только в локальных областях;
  - ✓ Должны быть проинициализированы непосредственно в месте создания;
  - ✓ Не допускают множественного объявления;
  - ✓ Константы не могут быть неявно типизированными

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

Закрепление материала

- Что такое переменная?
- Где и для чего используются переменные?
- Назовите основные типы данных.
- Какие типы данных подходят для хранения значений чисел с плавающей запятой?
- В каком формате должны задаваться значения для строковых переменных?
- Что такое константа?
- В каких случаях используются константы?
- Что такое преобразование значений типов (Casting)?
- Какие существуют правила использования преобразования значений при работе с константами?
- В чем разница явного неявного преобразования значения типа?
- Что такое конкатенация?
- Что такое переполнение и как его контролировать в программах?
- Что делает оператор `sizeof()`?
- Что такое инкремент и декремент?
- Какие ограничения применяются к неинициализированным локальным переменным?
- Можно ли использовать в операциях сравнения, два значения разных типов, данных?
- Что такое неявно типизированная локальная переменная?

Дополнительное задание

Задание
Используя VisualStudio, создайте проект по шаблону ConsoleApplication.
Создайте две целочисленные переменные и выведите на экран результаты всех арифметических операций над этими двумя переменными.

Самостоятельная деятельность учащегося

Задание 1
Имеется 3 переменные типа `int x = 10, y = 12, и z = 3;`
Выполните и рассчитайте результат следующих операций для этих переменных:

- `x += y - x++ * z;`
- `z = --x - y * 5;`
- `y /= x + 5 % z;`
- `z = x++ + y * 5;`
- `x = y - x++ * z;`

Задание 2
Используя Visual Studio, создайте проект по шаблону Console Application.
Вычислите среднее арифметическое трех целочисленных значений и выведите его на экран.
С какой проблемой вы столкнулись? Какой тип переменных лучше использовать для корректного отображения результата?

Задание 3
Используя Visual Studio, создайте проект по шаблону Console Application.
Создайте константу с именем -pi (число π «пи»), создайте переменную радиус с именем – r. Используя формулу πR^2, вычислите площадь круга и выведите результат на экран.

Задание 4
Используя Visual Studio, создайте проект по шаблону Console Application.
Напишите программу расчета объема – V и площади поверхности – S цилиндра.
Объем V цилиндра радиусом – R и высотой – h, вычисляется по формуле: $V = \pi R^2 h$
Площадь S поверхности цилиндра вычисляется по формуле: $S = 2\pi R\mathbf{h} + 2\pi R^2 = 2\pi R(R+h)$

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

Результаты расчетов выведите на экран.

Задание 5
Используя Visual Studio, создайте проект по шаблону Console Application.
Проверьте, можно ли создать переменные со следующими именами:
`uberflu?, _Identifier,  \u006fIdentifier, &myVar, myVariab1le`

Задание 6
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: Операторы C#
http://msdn.microsoft.com/ru-ru/library/6a71f45d.aspx

MSDN: Переменные и константы.
http://msdn.microsoft.com/ru-ru/library/wew5ytx4.aspx

MSDN: `sizeof()`.
http://msdn.microsoft.com/ru-ru/library/eahchzkf.aspx

MSDN: Ключевые слова C#
http://msdn.microsoft.com/ru-ru/library/x53a06bb.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