Змінні та типи даних

|  |  |  |
| --- | --- | --- |
| **№ Уроку:** | 2 | **Курс:** |

C#

|  |
| --- |
| **Засоби навчання :** |

Компьютер з програмою Visual Studio

**Огляд, мета і призначення уроку**

Розглянемо концепцію змінної, константи і типів даних.

Розглянемо арефметичні оператори та оператори порівняння.

**Вивчивши матеріал цього занняття, студент зможе:**

* Використовувати змінні та константи.
* Розуміти, коли і які типи використовувати при створенні змінної.
* Виконувати арефметичні операції над значеннями змінних.
* Порівнювати значення змінних.
* Виконувати форматування стрічок.

**Зміст уроку**

1. Розглянемо приклад: константи.
2. Розглянемо приклад: перетворення типів (Casting).
3. Розглянемо приклад: арифметичні оператори.
4. Розглянемо приклад: математичні функціїї.
5. Розглянемо приклад: інкремент та декремент.
6. Розглянемо приклад: операції порівняння.
7. Розглянемо приклад: привласнення з дією.
8. Розглянемо приклад: локальні області видимості.
9. Розглянемо приклад: ключові слова у якості ідентифікаторів.
10. Поняття переносу і переповнення (overflow).
11. Розглянемо приклад: перевірка переповнення (checked).
12. Розглянемо приклад: відсутність перевірки переповнення (unchecked).
13. Розглянемо приклад: комбінування (checked / unchecked).
14. Розглянемо приклад: конкатенація.
15. Розглянемо приклад: форматування стрічок.
16. Розглянемо приклад: прапорці форматування стрічок.
17. Розглянемо приклад: використання оператора sizeof().
18. Розглянемо приклад: неявно типізавані локальні змінні.
19. Розглянемо приклад: порівняння значень різних типів.

**Резюме**

* Змінна (Variable) - це область пам'яті, яка зберігає у собі певне значення, яке можна змінити
* Ініціалізація змінної – це перше присвоєння їй значення. Всі наступні присвоєння нових значень для цієї змінної не вважається Ініціалізацією.
* Технічно, імена змінних можуть починатись із знака «\_» нижнього підкреслення і будь-якого алфавітного символу. (Імена не можуть починатися з цифр та інших символів.)
* Для іменування локальних змінних в C #, рекомендується використовувати правило camel Casing. Щоб виділити слова в ідентифікаторах, перші літери кожного слова (окрім першого) необхідно писати великими. Наприклад: myAge, myName.
* Мова C# чутлива до регістрів. Наприклад: myName і MyName – це різні імена для змінних.
* Не використовуйте символи підкреслення, дефіси або будь-які інші символи не алфавітні символи для розділення слів ідентифікатора.
* Не використовуйте угорську нотацію. Суть угорської нотації полягає в тому, що імена ідентифікаторів мають префікс із одного або декількох символів обговорених наперед. string sClientName; Int isize;
* Імена змінних повинні бути чіткими і передавати значення кожного елементу.
* В інших випадках, якщо ідентифікатор немає точної семантичної цінності, використовуйте загальні імена. value , item.
* При створенні змінної використовуйте її псевдонім, коли це можливо, а не повне ім'я типу.
* Константа -це область пам'яті, яка містить певне значення, яке не можна змінити.
* Правила для використання Константант:
  + 1. Константам необхідно присвоювати значення безпосередньо у місці створення;
    2. Спроба присвоїти нове значення константі призводить до помилки рівня компіляції;
* Перетворення типу (Casting або Type conversion) - це перетворення значення змінної одного типу в значення іншого типу. Виділяють явне (explicit) і неявне (implicit) перетворення типів.
* Неявне перетворення типу (безпечний) – це перетворення меншого типу в більший або цілого числа у чило з плаваючою точкою.
* Явне перетворення типу (небезпечна) - це перетворення більшого типу у менший або чило з плаваючою точкою у ціле число. Це небезпечно, так як існує ймовірність втрати точності результату без заокруглення.
* Можливо неявне перетворення значення константи більшого типу в менший, при ініціалізації змінної значенням константи, якщо значення константи не перевищує максимально допустимого значення змінної.
* Можливо явне перетворення значення константи дійсного(з плаваючою точкою) типу в цілий тип, при ініціалізації змінної значенням константи, якщо значення константи не перевищує максимально допустимого значення змінної.
* Якщо значення константи перевищує максимально допустимий діапазон значення змінної, таке перетворення неможливе і призведе до помилки.
* Оператор присвоєння (=) зберігає значення свого правого операнда в місці зберігання (змінної) позначеної в лівому операнді. Операнди повинні бути одного типу (або правий операнд повинен допускати явне перетворення в тип лівого операнда).
* Якщо після знака присвоєння йде вираз з обчисленням або передачею будь-яких значень, то дана операція виконується справа-наліво. Для підвищення пріоритету операції можна використовувати круглі дужки ().
* Тільки чотири операції гарантують порядок обчислень зліва направо: ,,?:, && і ||
* Мова C # надає великий набір операторів, які представляють собою символи, які у свою чергу визначають операції. До операторів, які виконують арифметичні операції можна віднести такі:

+ (додавання),

- (віднімання),

\* (множення),

/ (ділення),

% (отримання залишку від ділення)

* Мова C# забезпечує великий набір математичних функцій для виконання різних обчислень.
* Math.Sqrt () - математична функція яка повертає квадратний корінь числа. У дужках вказуємо значення числа, з якого хочемо отримати квадратний корінь.
* Math.Pow () - приведення числа в степінь. У дужках через кому вказуємо два аргументи (перший - число, яке хочемо привести до степеня, другий - степінь, до якої ми хочемо привести число).
* Операції множення, ділення, отримання залишку від ділення мають більший пріоритет, ніж додавання і віднімання, тому виконуються в першу чергу.
* При отриманні результату залишку від ділення - знак результату не скорочується і відповідає значенню першого операнда (діленого).
* Якщо в правій частині виразу виконувались операції ділення між цілими числами, то результат буде приведений компілятором до цілого типу, навіть якщо результат записати у змінну дійсного(з плаваючою точкою) типу або привести весь вираз до дійсного типу.
* Оператор інкремента (++) збільшує свій операнд на 1. Оператор інкремента може перебувати як перед операндом, так і після нього: ++ variable або variable ++.
* Префіксна операція збільшення - результатом виконання цієї операції є значення операнда після його збільшення.
* Постфіксна операція збільшення - результатом виконання цієї операції є значення операнда перед його збільшенням.
* Оператор декремента (- -) зменшує свій операнд на 1. Оператор декремента може перебувати як перед операндом, так і після нього: - - variable або variable - -.
* Префіксна операція декремента - результатом виконання цієї операції є значення операнда після його декремента.
* Постфіксна операція декремента - результатом цієї операції є використання значення операнда до його декремента.
* До операцій порівняння можна віднести операції:

> більше,

> = більше або дорівнює,

< менше,

<= менше або дорівнює.

* До операцій перевірки на рівність можна віднести операції:

== дорівнює,

! = не дорівнює.

* Результатом виконання операцій порівняння і перевірки на рівність, нерівність завжди буде або false або true.
* Для визначених типів значень оператора рівності (==) повертає значення true, якщо значення його операндів збігаються, в іншому випадку - значення false. Для типу string оператор == порівнює значення стрічок.
* Оператор нерівності (! =) повертає значення false, якщо його операнди рівні, в іншому випадку - значення true.
* Оператор порівняння "менше або дорівнює" (<=) повертає значення true, якщо перший операнд менше або дорівнює другому, в іншому випадку повертається значення false.
* Оператор порівняння "менше" (<) повертає значення true, якщо перший операнд менше другого, в іншому випадку повертається значення false.
* Оператор порівняння "більше" (>) повертає значення true, якщо перший операнд більше другого, в іншому випадку повертається значення false.
* Оператор порівняння "більше або дорівнює" (> =) повертає значення true, якщо перший операнд більше або дорівнює другому, в іншому випадку повертається значення false.
* Всі арифметичні операції, що здійснюються над двома значеннями типу (byte, sbyte, short, ushort) в якості результату, повертають значення типу int.
* Для типів int, uint, long і ulong, не відбувається перетворення типу результату арифметичних операцій.
* Локальна область - ділянку коду, всередині класу або блоку, який обмежений фігурними дужками.
* Область видимості змінної - частина тексту програми, в якій ім'я можна явно використовувати. Найчастіше область видимості збігається з областю дії.
* Змінна створена всередині локальної області називається локальною змінної, область її дії - від відкриваючої дужки локальної області до її закінчення (закриваючої дужки) блоку, включаючи всі вкладені локальні області.
* Змінна рівня класу називається глобальною змінною або полем.
* У коді можна створювати локальні області і в двох різних локальних областях зберігати однойменні змінні.
* Якщо в коді є локальні області, то забороняється зберігати однойменні змінні за межами локальних областей. І навпаки, якщо за межами локальних областей вже створені змінні з якимось ім'ям, то в локальних областях цього рівня забороняється створювати однойменні змінні.
* Ключові слова- це попередньо визначені зарезервовані ідентифікатори, які мають спеціальні значення для компілятора. Їх не можна використовувати в програмі в якості ідентифікаторів, якщо тільки вони не містять префікс @.
* Символ @, який використовується в ідентифікаторі змінної, вказує компілятору, що це слово необхідно трактувати як ідентифікатор, а не як ключове слово С # або його команду.
* Символ @ не є частиною ідентифікатора, тому, @myVariable - це те ж саме, що і myVariable.
* Оператори C # можуть виконуватися в перевіряючому або неперевіряючому контексті. У перевіряючому контексті арифметичне переповнення викличе виняток (помилку).
* В неперевіряючому контексті арифметичне переповнення буде проігноровано, а результат успішний. Для таких дій використовуються такі конструкції:

сhecked- вказівка перевіряючому контексту;

unchecked- вказівка не перевіряючому контексту;

* Перевірка переповнень застосовується в наступних випадках:

1) Якщо виконуються операції над цілими числами з використанням наступних ареф операційй (++, -, +, -, \*, /).

2) Якщо виконуються явні числові перетворення між цілими типами даних.

* Конкатенація - зчеплення стрічок або значень змінних типу string, для отримання стрічки більшого розміру за допомогою операції +.
* Для форматування числових результатів і виведення їх на екран можна використовувати метод Console.Write () або Console.WriteLine (), який викликає метод string.Format ().
* Формат задається за допомогою прапорців форматування. Прапор форматування може мати наступну форму: Axx, де A - прапор формату (визначає тип формату), а xx - описувач точності (кількість відображуваних цифр або десяткових знаків форматованого результату). Наприклад: Console.WriteLine ( "{0: F2}", 99.935);
* Існують наступні прапорці форматування стрічок:

C або c - Валюта (Currency);

D або d - Десяткове число (Decimal);

E або e – Науковий формат (Scientific, exponential)

F або f - Формат з фіксованим значенням після коми (Fixed-point)

G або g - Загальні (General)

Nіліn - Number (Number)

X або x - Шіснадцятковий формат (Hexadecimal)

р -відсотковий (Percent)

* Оператор sizeof () - дозволяє отримати розмір значення в байтах для зазначеного типу.
* Оператор sizeof () можна застосовувати тільки до типів: (byte, sbyte, short, ushort, int, uint, long, ulong, float, double, decimal, char, bool).
* Повернені оператором sizeof () значення мають тип int.
* В C # 3.0 з'явилася можливість в області метода створювати змінні, які можуть мати неявний тип var. Такі змінні називають неявнотипізованими локальними змінними. Таким способом можна «доручити» компілятору визначити тип ваших змінних, якщо ви не знаєте точно результат.
* Правила використання неявнотипізованих локальних змінних:

Можна створити тільки в локальних областях;

Повинні бути проініціалізовані безпосередньо в місці створення;

Не допускають множинного оголошення;

Константи не можуть бути неявно типізовані;

**Закріплення матеріалу**

* Що таке змінна?
* Де і з для чого використовуються змінні?
* Які основні типи даних?
* Які типи підходять для зберігання значень із плаваючою точкою?
* У якому форматі слід встановити значення для змінних ітрчкового формату?
* Що таке константа?
* Коли використовуються константи?
* Що таке перетворення значень тупу?
* Які правила використання перетворення значень при роботі з константами?
* Яка різниця між явним і неявним перетворенням значення типу?
* Що таке переповнення і як його можна контролювати в програмах?
* Що таке конкатенація?
* Що робить оператор sizeof()?
* Що таке інкремент і декремент?
* Які обмеження застосовуються до неініціалізованих локальних змінних?
* Чи можна використовувати в операціях порівняння, два значення різних типів даних?
* Що таке неявно типізована локальна змінна?

**Дз**

Завдання1

Використовуючи VisualStudio створіть проект за шаблоном ConsoleApplication.

Створіть дві цілочисельні змінні і виведіть на екран результати всіх арифметичних операцій над цими двома змінними.

Завдання 2

Є 3 змінні типу int x = 10, y = 12, и z = 3;

Виконайте і розрахуйте результат наступних операцій для цих змінних, на листочку а потім програмно:

* x += y - x++ \* z**;**
* z = --x – y \* 5**;**
* y /= x + 5 % z**;**
* z = x++ + y \* 5**;**
* x = y - x++ \* z**;**

Завдання3

Використовуючи Visual Studio, створіть проект за шаблоном Console Application.

Обчисліть середнє арифметичне трьох цілочисельних значень і виведіть його на екран.

З якою проблемою ви зіткнулися? Який тип змінних краще використовувати для коректного відображення результату?

Завдання4

Використовуючи Visual Studio, створіть проект за шаблоном Console Application.

Створіть константу з ім'ям -pi (число π «пі»), створіть змінну радіус з ім'ям - r. Використовуючи формулу πR^2, обчисліть площу круга і виведіть результат на екран.

Завдання5

Використовуючи Visual Studio, створіть проект за шаблоном Console Application.

Напишіть програму розрахунку обсягу - V обєму і площі поверхні -S циліндра.

Обсяг V циліндра радіусом - R і висотою - h, обчислюється за формулою: V = πR^2 \* h

Площа S поверхні циліндра обчислюється за формулою: S = 2πR^2 + 2πR^2 = 2πR\*(R + h)

Результати розрахунків виведіть на екран.

Завдання6

Використовуючи Visual Studio, створіть проект за шаблоном Console Application.

Перевірте, чи можна створити змінні з наступними іменами:

uberflu? , \_Identifier, \ u006fIdentifier, & myVar, myVariab1le

Завдання7

Зайдіть на сайт MSDN.

Використовуючи пошукові механізми MSDN, знайдіть самостійно опис теми по кожному наприкладу, який був розглянутий на уроці, так, як це представлено нижче, в розділі «Рекомендовані ресурси», опису даного уроку. Збережіть посилання і дайте їм короткий опис.

**Рекомендовані ресурси**

MSDN: Оператори с #

http://msdn.microsoft.com/ru-ru/library/6a71f45d. aspx

MSDN: Змінні і константа.

http://msdn.microsoft.com/ru-ru/library/wew5ytx4.aspx

MSDN: sizeof ().

http://msdn.microsoft.com/ru-ru/library/eahchzkf.aspx

MSDN: Ключові слова с #

http://msdn.microsoft.com/ru-ru/library/x53a06bb.aspx