CTRL +  ← / →  – прескача през дума

CTRL + Shift + ← / →  – селектира цяла дума

CTRL + Backspace – изтрива цяла дума зад курсора

CTRL + Del – изтрива цяла дума пред курсора

Shift + ↑ / ↓ – селектира цял ред в зависимост от положението на курсора

Home – праща курсора до началото на реда

End – праща курсора до края на реда

Shift + Home – селектира всичко до началото на реда

Shift + End – селектира всичко до края на реда

CTRL + Home – праща курсора до началото на файла

CTRL + Shift + Home – селектира всичко до началото на файла

CTRL + End – праща курсора до края на файла

CTRL + Shift + End – селектира всичко до края на файла

CTRL + Z – undo

Ctrl + "." и Enter – сменя името на променливата навсякъде

Alt + мишката надолу за коментар на повече редове

F9 – посравя брекпойнт

Ctrl+ . + enter = добавя using Sistem…

class Car – върху Car + ctrl + . -> може да rename-нем навсякъде по кода

Маркиран текст + Ctrl+E+C и Ctrl+K+C = коментар

Ctrl+E+U и Ctrl+K+U = маха коментар

Ctrl+K+D = подравнява кода

Ctrl+. Върху текст подсказва коя библютека ни липсва

cr TAB --> Console.ReadLine();

int TAB --> int x = int.Parse(Console.ReadLine()); (съответно за long, string, decimal, double итн)

ca TAB --> var[] name = new var[size]; - за създаване на масив от даден тип (var) с определен размер (size)

cl TAB --> List<var> name = new List<var>(); - за лист

str TAB --> string name = Console.ReadLine();

stw TAB --> string name = "value";

cwc TAB --> foreach (var item in collection)  - за принтиране на колекция

{

Console.WriteLine(item);

}

**Integer Types:**

**sbyte (-128 to 127): signed 8-bit**

**byte (0 to 255): unsigned 8-bit**

**short (-32,768 to 32,767): signed 16-bit**

**ushort (0 to 65,535): unsigned 16-bit**

**int (-2,147,483,648 to 2,147,483,647): signed 32-bit**

**uint (0 to 4,294,967,295): unsigned 32-bit**

**long (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807): signed 64-bit**

**ulong (0 to 18,446,744,073,709,551,615): unsigned 64-bit**

**float (±1.5 × 10−45 to ±3.4 × 1038): 32-bits, precision of 7 digits**

**double (±5.0 × 10−324 to ±1.7 × 10308): 64-bits, precision of 15-16 digits**

**decimal (±1,0 × 10-28 to ±7,9 × 1028): 128-bits, precision of 28-29 digits**

-     **\'** – единична кавичка

-     **\"** – двойна кавичка

-     **\\**– лява наклонена черта

-     **\n** – нов ред

-     **\t**– отместване (табулация)

-     **\uXXXX**– символ, зададен с Unicode номера си, примерно **\u03A7**.

- @ - string a = @"The ""use"" of quotations causes difficulties.";

I.Логически оператори:

1.Логическо отрицание: сменя стойността на аргумента:

!true = false

!false = true

2. && (и) – Конюнкция – поне единият, ако е равен на false и целият резултат е false (връща истина, когато и двете операнди са истина):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| True | && | False | = | False |
| False | && | True | = | False |
| True | && | True | = | True |
| False | && | False | = | False |

3. || (или) – Дезункция – поне единият от двете операнди е true, разглежда първата операнда и ако тя е true и резулата е true (връща истина, когато поне едната операнда е истина):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| True | || | False | = | True |
| False | || | True | = | True |
| True | || | True | = | True |
| False | || | False | = | False |

4. ^ (xor) - Изключващо или – върща истина, ако двата операнда са различни:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| True | ^ | False | = | True |
| False | ^ | True | = | True |
| True | ^ | True | = | False |
| False | ^ | False | = | False |

II.Побитови оператори – променят битовете на едно число. Прави се „и“, „или“ или “xor” за всеки от битовете:

1. ~ - Тилдата обръща всички битове, т.е. навсякъде, където има 1 го прави 0 и обратното:

* &, | и ^ - са същите, но за всеки бит се разглеждат
* << и >> Побитово изместване с по два бита на ляво или надясно

b<<2 – 00000101 става 00010100

b>>2 – 00000101 става 00000001

III. С Convert.ToString(35,2).PadLeft(32,‘0‘); - може да видим 35 в двойчна бройна система, а ако е 16 на мястото на 2, може да видим 35 в шестнадесетична бройна система. С PadLeft може да кажем колко символа искаме да заема нашият стринг (в случая 32) и ако случайно стрига е по-малко на брой, тогава ислам да ми запълва със символа нула.

IV. Оператори за сравнение:

+=,-=,|=,<<=…

X |= 3 🡪 x = x |3 🡪 тове е равно на x или 3

X <<= 3 🡪 x = x <<3 🡪 тове е равно на x побитово измесено с 3

или обобщено: v1 0= v2; 🡪 v1 = v1 o v2;

* „? :“ - оператора ис: дали даден обект е от даден клас
* ?? – за избягване на дефолтни стойности (пример:вземи еди коя си стойност или 5)

X = null;

Y = x ?? 5;

Cw(y);//резултат 5

V.Инплицитно(автоматично) и експлицитно(изрично) прехвърляне на типове:

- Инплицитно кастване : int i = 5; long l = i;

IV. % - дели с остатък

* **How to get the bit at position p in a number n?**

**int p = 5;**

**int n = 35; // 00000000 00100011**

**int mask = 1 << p; // 00000000 00100000**

**int nAndMask = n & mask; // 00000000 00100000**

**int bit = nAndMask >> p; // 00000000 00000001**

**Console.WriteLine(bit); // 1**

* **How to set the bit at position p to 0?**

**int p = 5;**

**int n = 35; // 00000000 00100011**

**int mask = ~(1 << p); // 11111111 11011111**

**int result = n & mask; // 00000000 00000011**

**Console.WriteLine(result); // 3**

* **How to set the bit at position p to 1?**

**int p = 4;**

**int n = 35; // 00000000 00100011**

**int mask = 1 << p; // 00000000 00010000**

**int result = n | mask; // 00000000 00110011**

**Console.WriteLine(result); // 51**

* **How to print a binary number to the console?**

**Console.WriteLine(**

**Convert.ToString(result, 2).PadLeft(32, '0'));**

**// 00000000000000000000000000110011**

char symbol = '\u0019';

Console.WriteLine((int)symbol); - това е експлицитно кастване

string firstName = "Evlogi";

string lastName = "Ivanov";

string fullName = string.Format("{0} {1}", firstName, lastName);

Console.WriteLine(fullName); - конкатенация на стрингове