**C# Tips and Tricks**

**1.Минимални и максимални стойности на типовете данни:**

* **sbyte [-128 …127]: signed 8-bit [-27 … 27-1]**
* **byte [0 … 255]: unsigned 8-bit [0 … 28-1]**
* **short [-32 768 … 32 767]: signed 16-bit [-215 … 215-1]**
* **ushort [0 … 65 535]: unsigned 16-bit [0 … 216-1]**
* **int [-2 147 483 648 … 2 147 483 647]: signed 32-bit [-231 … 231-1]**
* **uint [0 … 4 294 967 295]: unsigned 32-bit [0 … 232-1]**
* **long [-9 223 372 036 854 775 808 … 9 223 372 036 854 775 807]: signed 64-bit [-263 … 263-1]**
* **ulong [0 … 18 446 744 073 709 551 615]: unsigned 64-bit [0 … 264-1]**
* **Floating-point types are:**
  + **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**
* **The default value of floating-point types:**
  + **Is 0.0F for the float type**
  + **Is 0.0D for the double type**
* **There is a special decimal floating-point real number type in C#:**
  + **decimal (±1,0 × 10-28 to ±7,9 × 1028)**
    - **128-bits, precision of 28-29 digits**
  + **Used for financial calculations**
  + **Almost no round-off errors**
  + **Almost no loss of precision**
* **The default value of decimal type is:**
  + **0.0M (M is the suffix for decimal numbers)**
* **BigInteger represents an arbitrarily large signed integer.**
  + **From the System.Numerics namespace.**
  + **Reference needs to be added.**
* **Needs special methods for simple actions like adding, subtracting, multiplying, powering etc.**

**2.За прочитане на няколко елемента и разделянето им по space ползвам:**

**for (int i = 0; i < N; i++)**

**{**

**var input = Console.ReadLine().Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries).ToList();**

**string siteName = input.First();**

**long siteVisits = long.Parse(input.Skip(1).First());**

**decimal pricePerVisit = decimal.Parse(input.Last());**

**}**

**3.За отпечатване на елементи от списък и масив ползвам:**

**Console.WriteLine(string.Join("\n", nameOfAllSites));**

**Като сепаратора в кавичките може да е различен,в този случай всеки елемент е на нов ред!**

**4.За включване на BigInteger-Reference/add/System.Numerics след това пиша BigInteger и го включвам с Ctrl+.(или Alt+Enter).**

**5.Полезно е да ползвам и степен с BigInteger:**

**BigInteger securityToken = BigInteger.Pow(securityKey, N);**

**Като 1-вото число е числото,което ще повдигаме на степен,а 2-рото,колко пъти ще го повдигаме.Друг полезен метод за повдигане на степен е:**

**BigInteger securityToken = 1;**

**for (int i = 0; i < N; i++)**

**{**

**securityToken \*= securityKey;**

**}**

**Като тук числото може да ми бъде от всякакъв тип.Последният метод за повдигане на число на степен е с използването на Math.Pow:**

**double value4 = Math.Pow(5, 2);**

**той връща double!!!**

**6.Хитър начин за направа на while loop докато се върти и да намалява е:**

**var amount=int.Parse(Console.ReadLine());**

**while(amount-- > 0)**

**{**

**}**

**Така цикъла хем се върти,хем намалява;**

**7.Използване на TimeSpan за време:**

**long secondsAll = secondsLeaving + secondsWithOutDays;**

**TimeSpan time = TimeSpan.FromSeconds(secondsAll);**

**string totalTime = time.ToString(@"hh\:mm\:ss");**

**Console.Write("Time Arrival: ");**

**Console.WriteLine(totalTime);**

**8.Метод за намиране на всички цифри,малки и големи букви и тяхната сума в string:**

**var allLetters = new string**

**(text.Where(x=>char.IsLetter(x)&&char.IsLower(x)).ToArray());**

**int sumLetters = 0;**

**for (int i = 0; i < allLetters.Length; i++)**

**{**

**sumLetters += allLetters[i];**

**}**

**9.Метод за намиране на сбора на chars в string:**

**string pattern = @"(\<.+?\>)";**

**string input = Console.ReadLine();**

**MatchCollection matches = Regex.Matches(input, pattern);**

**if (matches.Count == 0)**

**{**

**Console.WriteLine("Better luck next time");**

**return;**

**}**

**foreach (Match m in matches)**

**{**

**var allDigits =new string(m.ToString().Where(x => char.IsDigit(x)).ToArray());**

**var digitsToChar = allDigits.ToCharArray();**

**var sum = digitsToChar.Sum(x => int.Parse(x.ToString()));**

**Console.WriteLine($"Found {sum} carat diamond");**

**}**

**Ако ми трябват букви или символи само замествам char.IsLetter IsSymbol,ако не ги parsn-a c# отчита кода на ASCII таблицата,което е много удобно за букви и символи.**

**10.Бърз начин за сплитване по дадени елементи:**

**var placeholders = Console.ReadLine()**

**.Split("{}".ToCharArray(), StringSplitOptions.RemoveEmptyEntries).ToArray();**

**Мога да изброя всички елементи в двойните кавички и да ги сплитна(по-лесно е отколкото с char[])**

**11.Бърз начин за мачване и replace на текст е:**

**string pattern = @"([A-Za-z]+)(.+)\1";**

**string text = Console.ReadLine();**

**var placeholders = Console.ReadLine()**

**.Split("{}".ToCharArray(), StringSplitOptions.RemoveEmptyEntries).ToArray();**

**MatchCollection matches = Regex.Matches(text, pattern);**

**int i = 0;**

**foreach (Match item in matches)**

**{**

**string newValue = item.Groups[1] + placeholders[i++] + item.Groups[1];**

**text = text.Replace(item.ToString(), newValue);**

**}**

**Console.WriteLine(text);**

**12.Бърз начин за принтиране на елементи от лист или масив,всеки на нов ред е:**

**Console.WriteLine("Broadcasts:");**

**if (listOfBroadcasts.Count==0)**

**{**

**Console.WriteLine("None");**

**}**

**else**

**{**

**listOfBroadcasts.ForEach(x => Console.WriteLine(x));**

**}**

**13.Метод за Reverse na string:**

**string recepCode = matchMessage.Groups[1].Value;**

**recepCode = string.Join("", recepCode.ToCharArray().Reverse().ToArray());**

**14.Бърз начин за добавяне на елементи в дадено Dictionary е:**

**Dictionary<char, int> chars = new Dictionary<char, int>();**

**foreach (char c in currentWord)**

**{**

**if (chars.ContainsKey(c)) chars[c]++;**

**else chars.Add(c, 1);**

**}**

**15.Начин за разделяне и парсване на цифри от конзолата,които не са разделени:**

**var inputKey = Console.ReadLine().ToCharArray()**

**.Select(x => int.Parse(x.ToString())).ToList();**

**16.Ако имам много неща в едно Dictionary,а накрая трябва да изпринтирам малка част от тях,най-добре е да си направя ново Dictionary за резултата и да елиминирам елементите,които не ми трябват, иначе мога да сбъркам с LINQ(Hornet Armada).**

**17.Метода Substring:**

**int startIndexFirst = firtTeamName.IndexOf(key);**

**int lastIndexFirst = firtTeamName.LastIndexOf(key);**

**firtTeamName = firtTeamName**

**.Substring(startIndexFirst + key.Length, lastIndexFirst - startIndexFirst - key.Length);**

**firtTeamName = string.Join("", firtTeamName.ToUpper().ToCharArray().Reverse().ToArray());**

**Ползва се когато имам даден string между други 2 stringa и искам да го извадя.Вземат се стартовия индекс и последният индекс на stringowete.Зад Football League(2016/06/12)**

**18.Метод за reverse на елементи от лист по даден старт и брой:**

**var listOfElements = Console.ReadLine()**

**.Split(" ".ToCharArray(), StringSplitOptions.RemoveEmptyEntries)**

**.ToList();**

**int elementsLength = listOfElements.Count();**

**int lastIndexOfElements = elementsLength - 1;**

**string input = Console.ReadLine();**

**if (input == "end")**

**break;**

**string[] command = input.Split().ToArray();**

**string operation = command.First();**

**if (operation == "reverse")**

**{**

**int startIndex = int.Parse(command.Skip(2).First());**

**int count = int.Parse(command.Last());**

**long lastIndexReversed = startIndex + count;**

**if (startIndex < 0 || startIndex > lastIndexOfElements**

**|| count < 0 || count > elementsLength)**

**{**

**Console.WriteLine("Invalid input parameters.");**

**continue;**

**}**

**if (lastIndexReversed > elementsLength)**

**{**

**Console.WriteLine("Invalid input parameters.");**

**continue;**

**}**

**listOfElements.Reverse(startIndex, count);**

**}**

**19.Метод за sort на елементи от лист по даден старт и брой:**

**else if (operation == "sort")**

**{**

**int startIndex = int.Parse(command.Skip(2).First());**

**int count = int.Parse(command.Last());**

**long lastIndexSort = startIndex + count;**

**if (startIndex < 0 || startIndex > lastIndexOfElements**

**|| count < 0 || count > elementsLength)**

**{**

**Console.WriteLine("Invalid input parameters.");**

**continue;**

**}**

**if (lastIndexSort > elementsLength)**

**{**

**Console.WriteLine("Invalid input parameters.");**

**continue;**

**}**

**listOfElements.Sort(startIndex, count, null);**

**}**

**20.Методи за RollLeft and RollRight:**

**else if (operation == "rollLeft")**

**{**

**int count = int.Parse(command.Skip(1).First());**

**int timesRow = count % elementsLength;**

**if (count<0)**

**{**

**Console.WriteLine("Invalid input parameters.");**

**continue;**

**}**

**for (int i = 0; i < timesRow; i++)**

**{**

**listOfElements.Add(listOfElements.ElementAt(0));**

**listOfElements.RemoveAt(0);**

**}**

**}**

**else if(operation=="rollRight")**

**{**

**int count = int.Parse(command.Skip(1).First());**

**int timesRow = count % elementsLength;**

**if (count < 0)**

**{**

**Console.WriteLine("Invalid input parameters.");**

**continue;**

**}**

**listOfElements.Reverse();**

**for (int i = 0; i < timesRow; i++)**

**{**

**listOfElements.Add(listOfElements.ElementAt(0));**

**listOfElements.RemoveAt(0);**

**}**

**listOfElements.Reverse();**

**}**

**21.Метод за намиране на дните в даден месец и година:**

decimal pricePercapsule = decimal.Parse(Console.ReadLine());

var orderDate = Console.ReadLine().Split('/').ToArray();

int capsulesCount = int.Parse(Console.ReadLine());

var day = orderDate.First();

var month = int.Parse(orderDate.Skip(1).First());

var year = int.Parse(orderDate.Last());

decimal countDays =DateTime.DaysInMonth(year, month);

priceCoffee = countDays \* pricePercapsule \* capsulesCount;

**Датата се чете като string и се ползва опцията DateTime.DaysInMonth**

**22.Метод за позлване на TimeSpan и дата към string:**

var leavingTime = Console.ReadLine().Split(new char[] { ':' }).ToArray();

long stepsToHome = long.Parse(Console.ReadLine());

long secondsForStep = long.Parse(Console.ReadLine());

long allTimeInSeconds = stepsToHome \* secondsForStep;

int hours = int.Parse(leavingTime[0]);

int minutes = int.Parse(leavingTime[1]);

int seconds = int.Parse(leavingTime[2]);

int secondsLeaving = seconds + (60 \* minutes) + (3600 \* hours);

long secondsWithOutDays = allTimeInSeconds % (24\*3600);

long secondsAll = secondsLeaving + secondsWithOutDays;

TimeSpan time = TimeSpan.FromSeconds(secondsAll);

string totalTime = time.ToString(@"hh\:mm\:ss");

Console.Write("Time Arrival: ");

Console.WriteLine(totalTime);

**23.Стекове и опашки-определения,функции и начини на употреба:**

**1.**

**Stack<T> - структура от данни която пази елементи.Ако искаме да извадим елемент от структурата,то изваденият ще бъде последният вкаран елемент.**

**Пример: 1,2,3,34,56565,5 5 ще бъде изваденият елемент**

**Stack<T>.Push – добавяме елемент към стека(добавя се на последно място)**

**Stack<T>.Pop – изтриваме елемент от стека(последният добавен)**

**Stack<T>.Peek – показва последният добавен елемент в стека,но не го маха!**

**Stack<T>.Count – брой на елементите в стека**

**Stack<T>.ToArray – превръща стека в масив като реда на вкараните елементи се запазва**

**Stack<T>.Contains(n) – true or false взависимост дали стека съдържа n**

**Stack<T>.Clear – изтрива всички елементи от стека**

**Stack<T>.TrimExcess – прави стека в дължина,колкото са елементите(маха всички празни елементи)**

**Хитър начин за вадене на елементи от стек,докато не остане нито един е:**

**while (stack.Count != 0)**

**{**

**Console.Write(stack.Pop());**

**}**

**2.**

**Queue<T> - структура от данни която пази елементи.Ако искаме да извадим елемент от структурата,то изваденият ще бъде първият вкаран елемент.**

**Пример: 1,2,3,34,56565,5 1 ще бъде изваденият елемент**

**Queue<T>.Enqueue – добавяме елемент към опашката**

**Queue <T>.Dequeue – изтриваме елемент от опашката(първият добавен)**

**Queue <T>.Peek – показва последният добавен елемент в опашката,но не го маха!**

**Queue <T>.Count – брой на елементите в опашката**

**Queue <T>.ToArray – превръща опашката в масив като реда на вкараните елементи се запазва**

**Queue <T>.Contains(n) – true or false взависимост дали опашката съдържа n**

**Queue <T>.Clear – изтрива всички елементи от опашката**

**Queue <T>.TrimExcess – прави опашката с дължина,колкото са елементите(маха всички празни елементи)**