# Решения на задачите от глава 11 на книгата "Въведение в програмирането със C#"

Предлагаме ви решения на задачите от [книгата "Въведение в програ­мирането със C#"](http://www.introprogramming.info), заедно с анализ на задачата, описание на използва­ните идеи, алгоритми,подходи за решаване и тестове.

## Авторски колектив

**Георги Иванов Петров**

georgi.iv.petrov@gmail.com

**Георги Красимиров Георгиев**

georgievgeorgy@yahoo.com

**Мартин Борисов Василев**

martin\_bv@abv.bg

**Цветомир Йорданов Тодоров**

todorov.ceco@gmail.com

# Решение и описание на задачите

|  |  |
| --- | --- |
| Задача 1. Високосна година | |
| **Условие**  Напишете програма, която прочита от конзолата година и проверява дали е високосна. | |
| **Описание на входа**  Един единствен ред, съдържащ една година. | |
| **Описание на изхода**  True – високосна. False – невисокосна. | |
| **Анализ на задачата**  Потребиртелят въвежда число за година, което се преобразува в **integer** ор функцията **int.Parse**. Създава се нов клас от тип **DateTime**, който се инициализира с годината, въведена от потребителя. След това се извиква **isLeapYear**, който връща **true** ако годината е високосна и **false** ако не е. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  class CheckingForLeapYear  {  static void Main(string[] args)  {  //on the input line is the year as an integer number  int year = int.Parse(Console.ReadLine());  DateTime Date = new DateTime(year);  Console.WriteLine(DateTime.IsLeapYear(year));  // the output is True if its a leap year and False if it isn't  }  } | |
| **Тестовe**  Интересните случаи за тестване са следните:   * 2012 година - Вярно * 2011 година – Грешно * 1 година – Грешно   2314 година - Грешно | |
| **Вход** | **Изход** |
| 2012 | True |
| **Вход** | **Изход** |
| 2011 | False |
| **Вход** | **Изход** |
| 1 | False |
| **Вход** | **Изход** |
| 2314 | False |

|  |  |
| --- | --- |
| Задача 2. Генериране на случайни числа | |
| **Условие**  Напишете програма, която генерира и принтира на конзолата 10 случайни числа в интервала [100, 200]. | |
| **Описание на входа**  Няма | |
| **Описание на изхода**  Изписват се 10 произволни числа на нов ред в диапазона от 100 до 200. | |
| **Анализ на задачата**  Създава се обект **rand** от класа **Random** и последователно 10 пъти се извиква методът **Next()** с два параметъра. Първият задава **минималното произволно** число, което да се върне, а вторият **максималното -1**, което да се върне. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  class RandomNumbers  {  static void Main(string[] args)  {  Random rand = new Random();  for (int i = 0; i < 10; i++)  {  Console.WriteLine(rand.Next(100, 201));  }  }  } | |
| **Тестовe**  Интересните случаи за тестване са следните: | |
| **Вход** | **Изход** |
|  | 125  120  178  152  173  130  182  114  194  200 |

|  |  |
| --- | --- |
| Задача 3. Днешен ден от седмицата | |
| **Условие**  Напишете програма, която извежда на конзолата кой ден от седмицата е днес. | |
| **Описание на входа**  Няма. | |
| **Описание на изхода**  Днешният ден от седмицата. | |
| **Анализ на задачата**  Създава се променлива от тип **DateTime**, която се инициализира с днешния ден. След това той се извежда на конзолата. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  class DayOfTheWeek  {  static void Main(string[] args)  {  //there is no input line just the output of the day of the week  DateTime Date = DateTime.Today;  Console.WriteLine(Date.DayOfWeek);  }  } | |
| **Тестовe** | |
| **Вход** | **Изход** |
|  | Wednesday |

|  |  |
| --- | --- |
| Задача 4. Изминало време от включването на компютъра | |
| **Условие**  Напишете програма, която извежда на стандартния изход броя на дните, часовете и минутите, които са изтекли от включването на ком­пютъра до момента на изпълнението на програмата. За реализацията изпол­звайте класа Environment. | |
| **Описание на входа**  Няма | |
| **Описание на изхода**  Изминалото време от зареждането на операционната система. | |
| **Анализ на задачата**  Създава се обект **datePlusTickCountAdded**, който се инициализира с броя милисекунди, изминали от стартирането на операционната система. Получения период от време се отпечатва на екрана. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  class ElapsedTimeFromSystemStart  {  static void Main(string[] args)  {  // Console.WriteLine((Environment.TickCount));  DateTime datePlusTickCountAdded = DateTime.Now.AddMilliseconds(Environment.TickCount);  Console.WriteLine(datePlusTickCountAdded - DateTime.Now);  }  } | |
| **Тестовe** | |
| **Вход** | **Изход** |
|  | 01:23:41.2339999 |

|  |  |
| --- | --- |
| Задача 5. Хипотенуза на правоъгълен триъгълник | |
| **Условие**  Напишете програма, която по дадени два катета намира хипотенузата на правоъгълен триъгълник. Реализирайте въвеждане на дължините на катетите от стандартния вход, а за пресмятането на хипотенузата използвайте методи на класа Math. | |
| **Описание на входа**  Две страни на триъгълник, представени като цифра с възможна десетична запетая. Всяка на нов ред. | |
| **Описание на изхода**  Число със стойността на третата страна. | |
| **Анализ на задачата**  Потребителят въвежда 2 числа, като първо се записва в променливата **a**, а второто в променливата **b**. Питагоровата теорема гласи, че страната c е корен квадратен от **a\*a+b\*b**. Функцията **Math.Sqrt** връща **корен квадратен** от подаденото като параметър число, което се записва в **c** и показва на кнзолата. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  class FindingTheHypothenuse  {  static void Main(string[] args)  {  //the 2 sides a and b of the triangle are on the 2 input lines  //the output is the hypothenuse c found by the pythagorian theorem  double a = double.Parse(Console.ReadLine());  double b = double.Parse(Console.ReadLine());  double c = Math.Sqrt(a \* a + b \* b);  Console.WriteLine(c);  }  } | |
| **Тестовe**  Интересните случаи за тестване са следните:   * Катети равни, съответно на 5 и 4 * Катети равни, съответно на 1 и 0 * Катети равни, съответно на 0 и 0 | |
| **Вход** | **Изход** |
| 5  4 | 6.40312423743285 |
| **Вход** | **Изход** |
| 1  0 | 1 |
| **Вход** | **Изход** |
| 0  0 | 0 |

|  |  |
| --- | --- |
| Задача 6. Лице на триъгълник | |
| **Условие**  Напишете програма, която пресмята лице на триъгълник по:   * 1. дължините на трите му страни;   2. дължината на една от страните и височината към нея;   3. дължините на две от страните му и ъгъла между тях в градуси. | |
| **Описание на входа**  Потребителят въвежда **a**, **b** или **c** съответно за 3-те варианта на задачата. Ако е избрал **a,** въвежда 3 числа на нов ред за всяка страна. Ако е избрал **b,** въвежда 2 числа на нов ред за дължината на една от страните и височината към нея. Ако е избрал **c,** въвежда 3 числа всяко на нов ред съответно за дължините на две от страните му и ъгъла между тях в градуси. | |
| **Описание на изхода**  Пресметнатото лице на триъгълника. | |
| **Анализ на задачата**  Потребителят избира кой вариант иска да използва за изчисление на лице на триъгълник:   1. дължините на трите му страни; 2. дължината на една от страните и височината към нея; 3. дължините на две от страните му и ъгъла между тях в градуси.   В зависимост избрания вариант, въвежда нужните страни, височина или ъгъл между страни.  Методът **Area()** изчислява лицето и връща стойността му. Той има варианта, които **се различават по броя и вида параметри.** В зависимост от тях, бива извикана **една от трите версии**. Методите **Math.Sqrt()** и **Math.Sin()** връщат съответно корен квадратен или синус от подадения параметър. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  class TriangleArea  {  static double Area(double a, double h)  {  return a \* h / 2;  }  static double Area(double a, double b, double c)  {  double p = (a + b + c) / 2;  return Math.Sqrt(p \* (p - a) \* (p - b) \* (p - c));  }  static double Area(double a, double b, float angle)  {  return a \* b \* Math.Sin(angle \* Math.PI / 180) / 2;  }  static void Main(string[] args)  {  //choice which task to solve - a, b or c  Console.WriteLine("Please, input your choice - a, b, or c:");  string choice = Console.ReadLine();    switch (choice)  {  case "a":  Console.WriteLine("Please, input value for a:");  double a = double.Parse(Console.ReadLine());  Console.WriteLine("Please, input value for b:");  double b = double.Parse(Console.ReadLine());  Console.WriteLine("Please, input value for c:");  double c = double.Parse(Console.ReadLine());  Console.WriteLine(Area(a, b, c));  return;  case "b":  Console.WriteLine("Please, input value for a:");  a = double.Parse(Console.ReadLine());  Console.WriteLine("Please, input value for h:");  double h = double.Parse(Console.ReadLine());  Console.WriteLine(Area(a, h));  return;  case "c":  Console.WriteLine("Please, input value for a:");  a = double.Parse(Console.ReadLine());  Console.WriteLine("Please, input value for b:");  b = double.Parse(Console.ReadLine());  Console.WriteLine("Please, input value for the angle between a and b:");  float angle = float.Parse(Console.ReadLine());  Console.WriteLine(Area(a, b, angle));  return;  default:  Console.WriteLine("Please, input a, b or c as a subtask choice!");  return;  }  }  } | |
| **Тестовe**  Интересните случаи за тестване са следните:   * **Първа Формула**:   + Сбор на две от страните, равен на третата страна(1,3,4) – 0 * **Втора Формула:**   + Задаване на нулева стойност на височината - 0 * **Трета Формула:**   + Задаване на нулева стойност на ъгъла - 0 | |
| **Вход** | **Изход** |
| 1  3  4 | 4 |
| **Вход** | **Изход** |
| 10  0 | 0 |
| **Вход** | **Изход** |
| 5  143  0 | 0 |

|  |  |
| --- | --- |
| Задача 7. Упражнение с пространства | |
| **Условие**  Дефинирайте свое собствено пространство Chapter11 и поставете в него двата класа Cat и Sequence, които използвахме в примерите на текущата тема. Направете още едно собствено пространство с име Chapter11.Examples и в него направете клас, който извиква класовете Cat и Sequence. | |
| **Описание на входа**  Няма | |
| **Описание на изхода**  Cat Unnamed said: Miauuuuuu!  The color of cat Unnamed is gray. | |
| **Анализ на задачата**  Дефинират се три класа: **CatAndSequenceCaller**, **Cat** и **Sequence**. Създава се клас от тип **Cat**. Извиква се метода му **SayMiau()** и на конзолата се изписват името и цвета посредством достъп до свойствата **Name** и **Color**.  Класът **Cat** има 2 **private** променливи: **name** и **color**. В конструктора на **Cat** те приемат стойности съответно **Unnamed** и **gray**. Това са стойностите по подразбиране в случай, че конструкторът **Cat** се извика без параметри. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  class CatAndSequenceCaller  {  static void Main()  {  Cat someCat = new Cat();  someCat.SayMiau();  Console.WriteLine("The color of cat {0} is {1}.", someCat.Name, someCat.Color);  }  }  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  class Cat  {  // Field name  private string name;  // Field color  private string color;  public string Name  {  // Getter of the property "Name"  get  {  return this.name;  }  // Setter of the property "Name"  set  {  this.name = value;  }  }  public string Color  {  // Getter of the property "Color"  get  {  return this.color;  }  // Setter of the property "Color"  set  {  this.color = value;  }  }  // Default constructor  public Cat()  {  this.name = "Unnamed";  this.color = "gray";  }  // Constructor with parameters  public Cat(string name, string color)  {  this.name = name;  this.color = color;  }  // Method SayMiau  public void SayMiau()  {  Console.WriteLine("Cat {0} said: Miauuuuuu!", name);  }  }  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  class Sequence  {  // Static field, holding the current sequence value  private static int currentValue = 0;  // Intentionally deny instantiation of this class  private Sequence()  {  }  // Static method for taking the next sequence value  public static int NextValue()  {  currentValue++;  return currentValue;  }  } | |
| **Тестовe** | |
|  | **Изход** |
|  | Cat Unnamed said: Miauuuuuu!  The color of cat Unnamed is gray. |

|  |  |
| --- | --- |
| Задача 8. Дефиниране на обекти и използване на пространства. | |
| **Условие**  Напишете програма, която създава 10 обекта от тип Cat, дава им имена от вида CatN, където N e уникален пореден номер на обекта, и накрая извиква метода SayMiau() на всеки от тях. За реализацията използвайте вече дефинираното пространство Chapter11. | |
| **Описание на входа**  Няма | |
| **Описание на изхода**  Всеки един от 10-те обекта изкарва по един ред на конзолата. | |
| **Анализ на задачата**  Създава се масив от 10 обекта **Cat**. Завърта се цикъл 10 пъти, който създава нов обект **Cat** в масива, като му присвоява име, състоящо се от **Cat + поредна стойност от обекта** **Sequence**. Всеки път методът **Sequence.NextValue()** връща нарастващ номер (**Cat1**, **Cat2**, **Cat3**, …), от който да се образува стринга при създаването на обекта **Cat**.  С оператора **foreach** се прави цикъл, в който за всеки един обект в масива cats, се вика съответния метод **SayMiau().** Той печата по един ред на конзолата. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using task7.UsingClassesCatAndSequence.Chapter11;  class TenCats  {  static void Main(string[] args)  {  Cat[] cats = new Cat[10];  for (int i = 0; i < cats.Length; i++)  {  cats[i] = new Cat("Cat" + Sequence.NextValue(), "purple");  }  foreach (var cat in cats)  {  cat.SayMiau();  }  }  } | |
| **Тестовe** | |
|  | **Изход** |
|  | Cat Cat1 said: Miauuuuuu!  Cat Cat2 said: Miauuuuuu!  Cat Cat3 said: Miauuuuuu!  Cat Cat4 said: Miauuuuuu!  Cat Cat5 said: Miauuuuuu!  Cat Cat6 said: Miauuuuuu!  Cat Cat7 said: Miauuuuuu!  Cat Cat8 said: Miauuuuuu!  Cat Cat9 said: Miauuuuuu!  Cat Cat10 said: Miauuuuuu! |

|  |  |
| --- | --- |
| Задача 9. Брой работни дни между две дати. | |
| **Условие**  Напишете програма, която пресмята броя работни дни между днешната дата и дадена друга дата след днешната (включително). Работните дни са всички дни без събота и неделя, които не са официални празници, като по изключение събота може да е работен ден, когато се отработват почивни дни около празниците. Програмата трябва да пази списък от предварително зададени официални празници, както и списък от предварително зададени работни съботи. | |
| **Описание на входа**  Година, месец и ден в цифри, всяка на нов ред. Това е първоначалната дата.  Година, месец и ден в цифри, всяка на нов ред. Това е крайната дата. | |
| **Описание на изхода**  Едно число, показващо броя работни дни между двете въведени дати. | |
| **Анализ на задачата**  Потребителят въвежда **началните** и **крайните** година, месец и ден. Създават се две инстанции на класа **DateTime**, като **start** съдържа **началната** дата, а **end** – **крайната**.  Променливата **days** приема **общия брой дни** между двете дати, а **week** – **броя на седмиците**.  Ако има уикенд по средата на седмицата от началото, **days** се намалява, за да компенсира това. Следва цикъл, започващ от **началната година** и свършващ в **последната**. За всяка година се **отбелязват празниците** в конкретен месец и ден. За всеки празник между **началната** и **крайната** дата, който **не е събота или неделя**, **days** се намалява.  Накрая **days** се намалява с **2** за дните във **всяка седмица**, за да **компенсира** и **събота и неделя** и крайният резултат се отпечатва на конзолата. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  class TimeSpanBetweenDates  {  static void Main(string[] args)  {  // on the six input lines should be first the starting year month and day(integers)  // and then respectively the ending year month and day  // the output is the timespan between them and should be noted that the user should add the holidays listed by month and day  //(3,3); (5,1); (5, 6); (5, 24); (9, 6); (9, 22); (11, 1); (12, 24); (12, 25); (12, 26);  int StartingYear = int.Parse(Console.ReadLine());  int startMonth = int.Parse(Console.ReadLine());  int startDay = int.Parse(Console.ReadLine());  DateTime start = new DateTime(StartingYear, startMonth, startDay);  int EndingYear = int.Parse(Console.ReadLine());  int endMonth = int.Parse(Console.ReadLine());  int endDay = int.Parse(Console.ReadLine());  DateTime end = new DateTime(EndingYear, endMonth, endDay);  // basic data  int days = (int)(end - start).TotalDays;  int weeks = days / 7;  // check for a weekend in a partial week from start.  if (7 - (days % 7) <= (int)start.DayOfWeek)  days--;  if (7 - (days % 7) <= (int)start.DayOfWeek)  days--;  // check for holidays  for (int i = StartingYear; i <= EndingYear; i++)  {  int year = i;  DateTime[] holidays = new DateTime[10];  holidays[0] = new DateTime(year, 3, 3);  holidays[1] = new DateTime(year, 5, 1);  holidays[2] = new DateTime(year, 5, 6);  holidays[3] = new DateTime(year, 5, 24);  holidays[4] = new DateTime(year, 9, 6);  holidays[5] = new DateTime(year, 9, 22);  holidays[6] = new DateTime(year, 11, 1);  holidays[7] = new DateTime(year, 12, 24);  holidays[8] = new DateTime(year, 12, 26);  holidays[9] = new DateTime(year, 12, 25);  foreach (DateTime dt in holidays)  {  if ((dt > start) && (dt < end) && (dt.DayOfWeek != DayOfWeek.Saturday) && (dt.DayOfWeek != DayOfWeek.Sunday))  days--;  }  }  // lose the weekends  days -= weeks \* 2;  Console.WriteLine(days);  }  } | |
| **Тестовe**  Интересните случаи за тестване са следните:   * Примерен случай   Стартова дата: 1.1.1 до 1.1.1 | |
| **Вход** | **Изход** |
| 2000  12  21  2012  3  21 | 2854 |
| **Вход** | **Изход** |
| 1  1  1  1  1  1 | 0 |

|  |  |
| --- | --- |
| Задача 10. Сума на числа, записани като символен низ | |
| **Условие**  Дадена е последователност от цели положителни числа, записани едно след друго като символен низ, разделени с интервал. Да се напише програма, която пресмята сумата им. Пример: "43 68 9 23 318" 🡪 461. | |
| **Описание на входа**  На един ред се въвежда поредица от числа, отделени с интервал**.** | |
| **Описание на изхода**  На един ред се изписва сумата от въведените числа. | |
| **Анализ на задачата**  Потребителят въвежда поредицата от числа, отделени с интервал и те се подават на метода **StringToSum()** като символен низ **string**.  В този метод първо се извиква метода **Trim()**, който премахва излишни интервали в началото и края на символния низ. След това се създава масива **nums**, който се инициализира чрез метода **Split()** върху първоначалния низ. Резултатът е поредица (масив) от символни низове, отделени по критерий празно място между тях. В цикъл всеки отделен символен низ се преобразува в **integer** чрез метода **Parse()** и се добавя в променливата **sum**. Накрая тя бива върната и отпечатана на конзолата. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  class StringToSum  {  static int StringToSum(string s)  {  int sum = 0;  s = s.Trim();  string[] nums = s.Split(' ');  for (int i = 0; i < nums.Length; i++)  {  sum += int.Parse(nums[i]);  }  return sum;  }  static void Main(string[] args)  {  string s = Console.ReadLine(); //"43 68 9 23 318";  Console.WriteLine(StringToSum(s));  }  } | |
| **Тестовe**  Интересните случаи за тестване са следните:   * Примерен случай: 43 68 8 23 318 - 461   Поредица с отрицателни числа : 123 – 32 554 +54 -3 65 77 - 838 | |
| **Вход** | **Изход** |
| 43 68 8 23 318 | 461 |
| **Вход** | **Изход** |
| 123 – 32 554 +54 -3 65 77 | 838 |

|  |  |
| --- | --- |
| Задача 11. Случайно рекламно съобщение | |
| **Условие**  Напишете програма, която генерира случайно рекламно съобщение за някакъв продукт. Съобщенията трябва да се състоят от хвалебст­вена фраза, следвани от хвалебствена случка, следвани от автор (първо и второ име) и град, които се избират от предварително подготвени списъци. Например, нека имаме следните списъци:   * Хвалебствени фрази: {"Продуктът е отличен.", "Това е страхотен продукт.", "Постоянно ползвам този продукт.", "Това е най-добрият продукт от тази категория."}. * Хвалебствени случки: {"Вече се чувствам добре.", "Успях да се променя.", "Той направи чудо.", "Не мога да повярвам, но вече се чувствам страхотно.", "Опитайте и вие. Аз съм много доволна."}. * Първо име на автор: {"Диана", "Петя", "Стела", "Елена", "Катя"}. * Второ име на автор: {"Иванова", "Петрова", "Кирова"}. * Градове: {"София", "Пловдив", "Варна", "Русе", "Бургас"}.   Тогава програма би могла да изведе следното случайно-генерирано рекламно съобщение:   |  | | --- | | Постоянно ползвам този продукт. Опитайте и вие. Аз съм доволна. -– Елена Петрова, Варна | | |
| **Описание на входа**  Няма. | |
| **Описание на изхода**  Произволно съобщение, комбинация от фраза, събитие, първо име, фамилия и град. Техните стойности са заложени в самото приложение. | |
| **Анализ на задачата**  Създават се няколко **private** **static** стринга, съдържащи фрази, събития, първи имена на автори, фамилии на автори и градове на авторите.  Създава се нов обект **rnd** от клас **Random**. Създава се нов обект **CommercialMessage** от класа **StringBuilder**.  Посредством метода **GenerateString(string),** се избират произволна фраза, събитие, първо име, фамилия и град. Всички те се обединяват и изкарват на конзолата.  Методът **GenerateString(string)** приемта като параметъра списък от string-ове и връща произволен от тях посредством метода **rnd.Next().** | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  class RandomCommercialMessage  {  private static string[] PraisingPhrases = new string[]  {  "Продуктът е отличен.",  "Това е страхотен продукт.",  "Постоянно ползвам този продукт.",  "Това е най-добрият продукт от тази категория."  };  private static string[] PraisingEvents = new string[]  {  "Вече се чувствам добре.",  "Успях да се променя.",  "Той направи чудо.", "Не мога да повярвам, но вече се чувствам страхотно.",  "Опитайте и вие. Аз съм много доволна."  };  private static string[] AuthorsFirstName = new string[] { "Диана", "Петя", "Стела", "Елена", "Катя" };  private static string[] AuthorsLastName = new string[] { "Иванова", "Петрова", "Кирова" };  private static string[] AuthorsCity = new string[] { "София", "Пловдив", "Варна", "Русе", "Бургас" };  private static Random rnd = new Random();  static void Main(string[] args)  {  StringBuilder CommercialMessage = new StringBuilder();  string Phrase = GenerateString(PraisingPhrases);  CommercialMessage.Append(Phrase + " ");  string Event = GenerateString(PraisingEvents);  CommercialMessage.Append(Event + " ");  string FirstName = GenerateString(AuthorsFirstName);  string LastName = GenerateString(AuthorsLastName);  string City = GenerateString(AuthorsCity);  CommercialMessage.Append(" -- " + FirstName + " " + LastName + ", " + City);  string finalMessage = CommercialMessage.ToString();  Console.WriteLine(finalMessage);  }  //From each array of strings i generate a random one for the random message and the output is that message  //The arrays should be the ones listed in the task  private static string GenerateString(string[] availableStrings)  {  int randomIndex = rnd.Next(availableStrings.Length);  string randomString = availableStrings[randomIndex];  return randomString;  }  } | |
| **Тестовe** | |
| **Вход** | **Изход** |
|  | Продуктът е отличен. Той направи чудо. -- Диана Петрова, Варна |
| **Вход** | **Изход** |
|  | Това е страхотен продукт. Той направи чудо. -- Стела Иванова, Русе |
| **Вход** | **Изход** |
|  | Това е най-добрият продукт от тази категория. Вече се чувствам добре. – Стела Иванова, София |
| **Вход** | **Изход** |
|  | Постоянно ползвам този продукт. Опитайте и вие. Аз съм много доволна. -- Петя Кирова, Русе |

|  |  |
| --- | --- |
| Задача 12. Изчисление на числен израз | |
| **Условие**  \* Напишете програма, която изчислява стойността на даден числен израз, зададен като стринг. Численият израз се състои от:   * реални числа, например 5, 18.33, 3.14159, 12.6; * аритметични оператори: +, -, \*, / (със стандартните им приоритети); * математически функции: ln(x), sqrt(x), pow(x,y); * скоби за промяна на приоритета на операциите: ( и ). | |
| **Описание на входа**  Потребителят въвежда на един ред числов израз, съдържащ произволна комбинация от числа и всички позволени върху тях оператори от условието. | |
| **Описание на изхода**  Три реда. Първият показва входния символен низ от числа и оператори върху тях. Вторият показва Reversed Polish notation на същия символен низ, а третият – финалния резултат след изчислението. | |
| **Анализ на задачата**  След като потребителят въведе числения израз, той се отпечатва на конзолата и се подава на метода **ShuntingYard().**  Методът **ShuntingYard()** има за цел да го преобразува в **Reverse Polish notation**. Целта е изразът да се преподреди в удобна форма, в която всеки две числа, върху които ще бъде извършена някаква операция се подреждат едно след друго, а операторът, който ще я извърши, се подрежда непосредствено след тях.  Завърта се цикъл за всяка позиция от входния **string**. Пропускат се празните символи.  Ако символът е **число**, той се добавя в StringBuilder обекта **number**. Ако е и последният символ, number се преобравува с **string**, добавя му се празен символ и се добавя в **опашката numbers**.  Ако символът не е число и е предпоследния символ, на **масива** **number** се добавя **празен символ** и добавя в опашката **numbers**. В зависимост от това дали символът започва с **p**, **s** или **запетая**, се създава нов **string** от следващия елемент и резултатът се добавя в **стека operators**. Методът **IsOpp()** връща стойност true ако символът се окаже оператор.  Тъй като операторите имат различен приоритет, този приоритет се пази в **hash таблицата precedence** в числов вид. По този начин стойността на текущия оператор се записва в променливата **value1** чрез метода **TryGetValue()**.  След това докато в **стека operators** има елементи, стойността на последно постъпилия такъв в стека се записва в **value2** и ако се окаже, че **value1 < value2**, последния оператор се изважда от **стека operators** и се добавя в **опашката numbers**.  Ако се окаже, че символът не е нито число, нито оператор, а **отворена скоба**, той директно се добавя в **стека operators**. В случай, обаче, че е **затворена скоба**, всички оператори от **operators** до **отворена скоба** се изваждат от там и се **добавят в опашката numbers**.  Накрая всички **останали оператори** се **изваждат от стека operators** и се **добавят към опашката numbers**. Резултатът е **опашката numbers, в която всички елементи са подредени в Reversed Polish notation.** Тук методът **ShuntingYard()** приключва, изпълнението на кода се връща в **Main()** метода, а на екрана се изписва върнатия **string**, под формата на **Reversed Polish notation**.  Резултатът се разделя на база празно място между елементите и се подава като **свързан списък** на метода **ReversedPolish()**. Неговата цел е да **изчисли крайния резултат**.  В метода **ReversedPolish()** за всеки елемент от свързания списък се проверява какъв тип операция е. **Обръща се внимание само на типовете операции, а не на самите числа**. За всеки тип операция се взимат **двата елемента преди операцията**, извършва се **действието** върху тях според оператора и се записва в променливата **temp**. След това текущия елемент приема стойността на **temp**, а **двете числа** премахват от **свързания списък** и функцията **ReversedPolish() се извиква рекурсивно за останалите вътре елементи**. **Дъното** на рекурсията се постига когато **няма останали повече оператори**, а всички те са **земестени** на място от получен резултат между двете числа и **оператора**, **действащ върху тях**. Крайният резултат от рекурсията е **едно число** и то се извежда на конзолата. | |
| **Решение (сорс код)** | |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  class CalculateExpression  {  static Dictionary<string, int> precedence = new Dictionary<string, int>()  { { "+", 2 }, { "-", 2 }, { "\*", 4 }, { "/", 4 }, { "ln", 8 }, { "pow", 8 }, { "sqrt", 8 } };  static Stack<string> operators = new Stack<string>();  static Queue<string> numbers = new Queue<string>();  static bool IsNumber(char next)  {  switch (next)  {  case '1': return true;    case '2': return true;    case '3': return true;    case '4': return true;    case '5': return true;    case '6': return true;    case '7': return true;    case '8': return true;    case '9': return true;    case '0': return true;    case '.': return true;    default: return false;  }  }  static bool IsOpp(char next)  {  switch (next)  {  case '+': return true;    case '-': return true;    case '/': return true;    case '\*': return true;    default: return false;  }  }  static string ShuntingYard(String s)  {  // optimization for negative numbers  if (s.Contains(", -"))  s = s.Replace(", -", ", 0-");  if (s.Contains(",-"))  s = s.Replace(",-", ",0-");  if (s.Contains("( -"))  s = s.Replace("( -", "( 0-");  if (s.Contains("(-"))  s = s.Replace("(-", "(0-");  // Console.WriteLine(s);  StringBuilder number = new StringBuilder("");  for (int i = 0; i < s.Length; i++)  {  char next = s.ElementAt(i);  if (next == ' ')  continue;  if (IsNumber(next))  {  number.Append(next);  if (i == s.Length - 1)  {  number.Append(" ");  string num = number.ToString();  numbers.Enqueue(num);  number.Clear();  }  }  else  {  if (number.Length > 0 || i == s.Length - 1)  {  number.Append(" ");  string num = number.ToString();  numbers.Enqueue(num);  number.Clear();  }  if (next == 'l')  {  string fn = "" + s.ElementAt(i) + s.ElementAt(i + 1);  operators.Push(fn);  }  else if (next == 'p')  {  string fn = "" + s.ElementAt(i) + s.ElementAt(i + 1) + s.ElementAt(i + 2);  operators.Push(fn);  }  else if (next == 's')  {  string fn = "" + s.ElementAt(i) + s.ElementAt(i + 1) + s.ElementAt(i + 2) + s.ElementAt(i + 3);  operators.Push(fn);  }  else if (next == ',')  {  while (operators.Peek() != "(")  {  numbers.Enqueue(operators.Pop() + " ");  }  }  else if (IsOpp(next))  {  int value1;  int value2;  precedence.TryGetValue(next.ToString(), out value1);  while (operators.Count > 0)  {  precedence.TryGetValue(operators.Peek(), out value2);  if (value1 <= value2 && next.ToString() != operators.Peek())  {  numbers.Enqueue(operators.Pop() + " ");  }  else  break;  }  string opp = next.ToString();  operators.Push(opp);  continue;  }  else if (next == '(')  {  operators.Push(next.ToString());  }  else if (next == ')')  {  while (operators.Peek() != "(")  {  numbers.Enqueue(operators.Pop() + " ");  }  if (operators.Peek() == "(")  {  operators.Pop();  }  if (operators.Count > 0 && (operators.Peek() == "ln" || operators.Peek() == "pow" || operators.Peek() == "sqrt"))  {  numbers.Enqueue(operators.Pop() + " ");  }  }  }  }  while (operators.Count > 0)  {  numbers.Enqueue(operators.Pop() + " ");  }  StringBuilder result = new StringBuilder(s.Length);  foreach (var item in numbers)  {  result.Append(item);  }  return result.ToString();  }  static string ReversedPolish(List<string> token)  {  for (int i = 0; i < token.Count; i++)  {  string temp = "";  switch (token[i])  {  case "+":  temp = (double.Parse(token[i - 1]) + double.Parse(token[i - 2])).ToString();  token[i] = temp;  token.RemoveRange(i - 2, 2);  ReversedPolish(token);  break;  case "-": temp = (double.Parse(token[i - 2]) - double.Parse(token[i - 1])).ToString();  token[i] = temp;  token.RemoveRange(i - 2, 2);  ReversedPolish(token);  break;  case "/": temp = (double.Parse(token[i - 2]) / double.Parse(token[i - 1])).ToString();  token[i] = temp;  token.RemoveRange(i - 2, 2);  ReversedPolish(token);  break;  case "\*": temp = (double.Parse(token[i - 2]) \* double.Parse(token[i - 1])).ToString();  token[i] = temp;  token.RemoveRange(i - 2, 2);  ReversedPolish(token);  break;  case "ln": temp = (Math.Log(double.Parse(token[i - 1]))).ToString();  token[i] = temp;  token.RemoveRange(i - 1, 1);  ReversedPolish(token);  break;  case "sqrt": temp = (Math.Sqrt(double.Parse(token[i - 1]))).ToString();  token[i] = temp;  token.RemoveRange(i - 1, 1);  ReversedPolish(token);  break;  case "pow": temp = (Math.Pow(double.Parse(token[i - 2]), double.Parse(token[i - 1]))).ToString();  token[i] = temp;  token.RemoveRange(i - 2, 2);  ReversedPolish(token);  break;  default:  break;  }  }  return token[0];  }  static void Main(string[] args)  {  string s = Console.ReadLine(); //"(3+5.3) \* 2.7 - ln(22) / pow(2.2, -1.7) ";    Console.WriteLine("The string: " + s);  Console.WriteLine();  string result = ShuntingYard(s);  Console.WriteLine("Reversed Polish Notation: " + result);  Console.WriteLine();  string[] array = result.Split(' ');  List<string> token = array.ToList<string>();  Console.WriteLine("Result: " + ReversedPolish(token));  Console.WriteLine();  }  } | |
| **Тестовe**  Интересните случаи за тестване са следните:   * Примерен случай: -1 + 2 + 3 \* 4 – 0.5 =(-1) + 2 + (3\*4) – 0.5 = 12.5 * Поредица с отрицателни числа : 123 – 32 554 +54 -3 65 77 – 838 = -68116 * 5! – 5! = 1\*2\*3\*4\*5-5\*4\*3\*2\*1 = 0 * ln 5 = 1.6094379124341 * sqrt(14.34) = 3.7868192457523 * pow(3,5.12)=277.243149722385 * ln(sqrt25) = ln(5) = 1.6094379124341 * pow(5,sqrt25)= pow(5,5)=3125 * ln1\*sqrt(673332)\*pow(5,0)-1 | |
|  | **Изход** |
| -1 + 2 + 3 \* 4 – 0.5 | The string: -1+2+3\*4-0.5  Reversed Polish Notation: 0 1 - 2 3 4 \* + + 0.5 -  Result: 12.5 |
| **Вход** | **Изход** |
| (-1) + 2 + (3\*4) – 0.5 | The string: (-1) + 2 + (3\*4) - 0.5  Reversed Polish Notation: 0 1 - 2 3 4 \* + + 0.5 -  Result: 12.5 |
| **Вход** | **Изход** |
| 123 – 32 554 +54 -3 65 77 - 838 | The string: 123 - 32 554 +54 -3 65 77 - 838  Reversed Polish Notation: 123 32554 - 54 + 36577 838 - -  Result: -68116 |
| **Вход** | **Изход** |
| 1\*2\*3\*4\*5 - 5\*4\*3\*2\*1 | The string: 1\*2\*3\*4\*5 - 5\*4\*3\*2\*1  Reversed Polish Notation: 1 2 3 4 5 \* \* \* \* 5 4 3 2 1 \* \* \* \* -  Result: 0 |
| **Вход** | **Изход** |
| ln5 | The string: ln5  Reversed Polish Notation: 5 ln  Result: 1.6094379124341 |
| **Вход** | **Изход** |
| sqrt(14.34) | The string: sqrt(14.34)  Reversed Polish Notation: 14.34 sqrt  Result: 3.7868192457523 |
| **Вход** | **Изход** |
| pow(3,5.12) | The string: pow(3,5.12)  Reversed Polish Notation: 3 5.12 pow  Result: 277.243149722385 |
| **Вход** | **Изход** |
| ln(sqrt25) | The string: ln(sqrt25)  Reversed Polish Notation: 25 sqrt ln  Result: 1.6094379124341 |
| **Вход** | **Изход** |
| pow(5,sqrt25) | The string: pow(5,sqrt25)  Reversed Polish Notation: 5 25 sqrt pow  Result: 3125 |
| **Вход** | **Изход** |
| ln1\*sqrt(673332)+pow(5,0)-1 | The string: ln1\*sqrt(673332)+pow(5,0)-1  Reversed Polish Notation: 1 ln 673332 sqrt \* 5 0 pow + 1 -  Result: 0 |