1.    Write a program, which reads from the console a year and **checks if it is a leap year**.

using System;

public class Demo

{

public static void Main()

{

Console.WriteLine("Enter the year");

int year = Int32.Parse(Console.ReadLine());

if (DateTime.IsLeapYear(year))

{

Console.WriteLine("Leap Year!");

}

else

{

Console.WriteLine("Not a Leap Year!");

}

}

}

2.    Write a program, which generates and prints on the console **10 random numbers** in the range [100, 200].

using System;

class GFG

{

public static void Main()

{

Random rand1 = new Random();

Console.WriteLine("Printing 10 random numbers between 100 and 200");

for (int i = 1; i <= 10; i++)

Console.WriteLine( rand1.Next(100, 200));

}

}

3.    Write a program, which prints, on the console **which day of the week is today**.

using System;

public class Example3

{

static void Main(string[] args)

{

Console.Write("Enter year: ");

int year = Int32.Parse(Console.ReadLine());

Console.Write("Enter month: ");

int month = Int32.Parse(Console.ReadLine());

Console.Write("Enter day: ");

int day = Int32.Parse(Console.ReadLine());

DateTime ditaesotme = new DateTime(year,month,day);

Console.WriteLine(ditaesotme.ToString("ddddd"));

}

}

4.    Write a program, which prints on the standard output the **count of days, hours, and minutes, which have passes since the computer is started** until the moment of the program execution. For the implementation use the class **Environment**.

using System;

namespace Detyra\_4

{

class Program

{

static void Main(string[] args)

{

var mo = new ManagementObject(@"\root\cimv2:Win32\_OperatingSystem@"

);

var lastBootUp = ManagementDateTimeConverter.ToDateTime(mo["LastBootupTime"

].Tostring());

Console.WriteLine(DateTime.Now.ToUniversalTime() - lastBootUp.TouniversalTime());

}

}

}

5.    Write a program which by given two sides **finds the hypotenuse of a right triangle**. Implement entering of the lengths of the sides from the standard input, and for the calculation of the hypotenuse use methods of the class **Math**.

using System;

namespace Detyra\_5

{

class Program

{

static void Main(string[] args)

{

Console.Write("First side: ");

int a = Int32.Parse(Console.ReadLine());

Console.Write("Second side: ");

int b = Int32.Parse(Console.ReadLine());

Console.Write("Hypotenuse is: " + Math.Sqrt(Math.Pow(a, 2) + Math.Pow(b, 2)));

}

}

}

6.    Write a program which **calculates the area of a triangle** with the following given:

-     three sides;

-     side and the altitude to it;

-     two sides and the angle between them in degrees.

using System;

namespace Detyra6

{

class Program

{

public static void byThreeSides()

{

Console.Write("\nEnter a: ");

float a = float.Parse(Console.ReadLine());

Console.Write("Enter b: ");

float b = float.Parse(Console.ReadLine());

Console.Write("Enter c: ");

float c = float.Parse(Console.ReadLine());

float p = (a + b + c) / 2;

Console.WriteLine("S is: {0}", (float)(Math.Sqrt(p \* (p - a) \* (p - b) \* (p - c))));

Console.ReadLine();

}

public static void bySideAltitude()

{

Console.Write("\nEnter a: ");

float a = float.Parse(Console.ReadLine());

Console.Write("Enter h(a): ");

float b = float.Parse(Console.ReadLine());

Console.WriteLine("S is: {0}", (a \* b) / 2);

Console.ReadLine();

}

public static void byTwoSidesAngle()

{

Console.Write("\nEnter a: ");

float a = float.Parse(Console.ReadLine());

Console.Write("Enter b: ");

float b = float.Parse(Console.ReadLine());

Console.Write("Enter sine: ");

float c = float.Parse(Console.ReadLine());

Console.WriteLine("S is: {0}", (a \* b \* Math.Sin(c)) / 2);

Console.ReadLine();

}

static void Main(string[] args)

{

byte choice;

do

{

Console.Clear();

Console.WriteLine("Calculate the area of a triangle by given:");

Console.WriteLine("1.Three side lengths.");

Console.WriteLine("2.Length of one side and it's altitude.");

Console.WriteLine("3.Length of two sides and the angle between them.");

Console.WriteLine("4.Exit.");

Console.Write("Enter choice from 1 to 4: ");

choice = System.Convert.ToByte(Console.ReadLine());

switch (choice)

{

case 1: byThreeSides(); break;

case 2: bySideAltitude(); break;

case 3: byTwoSidesAngle(); break;

}

} while (choice != 4);

}

}

}

7.    Define your own namespace **CreatingAndUsingObjects** and place in it two classes **Cat** and **Sequence**, which we used in the examples of the current chapter. Define one more namespace and make a class, which calls the classes **Cat** and **Sequence**, in it.

8.    Write a program which creates 10 objects of type **Cat**, gives them names **CatN**, where **N** is a unique serial number of the object, and in the end call the method **SayMiau()** for each of them. For the implementation use the namespace **CreatingAndUsingObjects**.

9.    Write a program, which **calculates the count of workdays between the current date and another given date**after the current (inclusive). Consider that workdays are all days from Monday to Friday, which are not public holidays, except when Saturday is a working day. The program should keep a list of predefined public holidays, as well as a list of predefined working Saturdays.

using System;

namespace Detyra\_9

{

class Program

{

static void Main(string[] args)

{

DateTime[] holidays = new DateTime[]

{

new DateTime(2014, 12, 24),

new DateTime(2014, 12, 25),

new DateTime(2015, 01, 01),

new DateTime(2015, 01, 02),

new DateTime(2015, 03, 02),

new DateTime(2015, 03, 03),

new DateTime(2015, 04, 10),

new DateTime(2015, 04, 13),

new DateTime(2015, 05, 01),

new DateTime(2015, 05, 06),

new DateTime(2015, 09, 21),

new DateTime(2015, 09, 22),

new DateTime(2015, 12, 24),

new DateTime(2015, 12, 25),

};

DateTime[] workSaturdays = new DateTime[]

{

new DateTime(2015, 01, 24),

new DateTime(2015, 03, 21),

new DateTime(2015, 09, 12),

new DateTime(2015, 12, 12),

};

int workingDays = 0;

Console.Write("Enter end date (YYYY/MM/DD): ");

DateTime endDate = System.Convert.ToDateTime(Console.ReadLine());

DateTime now = DateTime.Now;

do

{

now = now.AddDays(1);

if ((now.DayOfWeek >= DayOfWeek.Monday) && (now.DayOfWeek <= DayOfWeek.Friday))

workingDays++;

foreach (var i in holidays)

if (i.Date == now.Date)

workingDays--;

foreach (var i in workSaturdays)

if (i.Date == now.Date)

workingDays++;

} while (now.Date != endDate.Date);

Console.WriteLine("{0} working days.", workingDays);

}

}

}

10.   You are given a **sequence of positive integer numbers** given as string of numbers separated by a space. Write a program, which **calculates their sum**. Example: "**43 68 9 23 318**"  **461**.

using System;

namespace Detyra10

{

class Program

{

static void Main(string[] args)

{

int result = 0;

Console.Write("Enter numbers: ");

string inputNumbers = Console.ReadLine();

string[] splitNumbers = inputNumbers.Split(' ');

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

result += Convert.ToInt32(splitNumbers[i]);

Console.WriteLine("Result is: {0}", result);

}

}

}

11.   Write a program, which **generates a random advertising message** for some product. The message has to consist of laudatory phrase, followed by a laudatory story, followed by author (first and last name) and city, which are selected from predefined lists. For example, let’s have the following lists:

-     **Laudatory phrases**: {"The product is excellent.", "This is a great product.", "I use this product constantly.", "This is the best product from this category."}.

-     **Laudatory stories**: {"Now I feel better.", "I managed to change.", "It made some miracle.", "I can’t believe it, but now I am feeling great.", "You should try it, too. I am very satisfied."}.

-     **First name** of the author: {"Dayan", "Stella", "Hellen", "Kate"}.

-     **Last name** of the author: {"Johnson", "Peterson", "Charls"}.

-     **Cities**: {"London", "Paris", "Berlin", "New York", "Madrid"}.

Then the program would print randomly generated advertising message like the following:

|  |
| --- |
| I use this product constantly. You should try it, too. I am very satisfied. -- Hellen Peterson, Berlin |

using System;

namespace Detyra\_11

{

class Program

{

static void Main(string[] args)

{

Random rnd = new Random();

string[] firstNames = new string[] { "Diana", "Petia", "Stela", "Elena", "Katia" };

string[] secondNames = new string[] { "Ivanova", "Petrova", "Kirova", "Popova" };

string[] cities = new string[] { "Sofia", "Plovdiv", "Varna", "Ruse", "Dolno konare" };

string[] reviews1 = new string[] { "This product is excelent.", "This is a great product.", "I use this product all the time.", "This is the best product of this category." };

string[] reviews2 = new string[] { "Now I feel better.", "I changed.", "It was a miracle.", "I can't believe it. Now I feel better.", "Try it yourself. I'm very greatful." };

Console.WriteLine("{0} {1}\n-- {2} {3}, {4}.", reviews1[rnd.Next(4)], reviews2[rnd.Next(5)], firstNames[rnd.Next(5)], secondNames[rnd.Next(4)], cities[rnd.Next(5)]);

}

}

}

12.   \* Write a program, which calculates the value of a given numeral expression given as a string. The numeral expression consists of:

-     real numbers, for example **5**, **18.33**, **3.14159**, **12.6**;

-     arithmetic operations: **+**, **-**, **\***, **/**(with their standard priorities);

-     mathematical functions: **ln(x)**, **sqrt(x)**, **pow(x,** **y)**;

-     brackets for changing the priorities of the operations: **(** and **)**.

Note that the numeral expressions have priorities, for example the expression **-1** **+** **2** **+** **3** **\*** **4** **-** **0.5** **=** **(-1)** **+** **2** **+** **(3** **\*** **4)** **-** **0.5** **=** **12.5**.

using System;

namespace Detyra\_12

{

class Program

{

public static string equation;

public static string[] equationArr = new string[50];

public static string[] actions = new string[50];

public static string[] numbers = new string[50];

public static bool change = false;

public static double result = 0;

public static double temp = 0;

public static byte depth = 0;

public static byte brakets = 0;

public static void removeNulls()

{

numbers = numbers.Where(x => !string.IsNullOrEmpty(x)).ToArray();

actions = actions.Where(x => !string.IsNullOrEmpty(x)).ToArray();

}

static void Main(string[] args)

{

Console.WriteLine("Enter equation: ");

equation = Console.ReadLine();

equationArr = equation.Split(' ');

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

{

if (change) actions[i] = equationArr[i];

else numbers[i] = equationArr[i];

change = change ? false : true;

}

removeNulls();

do

{

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

if (i == 0)

if (actions[i] == "\*" || actions[i] == "/")

{

if (actions[i] == "\*")

temp = Convert.ToDouble(numbers[0]) \* Convert.ToDouble(numbers[1]);

else temp += Convert.ToDouble(numbers[0]) / Convert.ToDouble(numbers[1]);

numbers[0] = temp.ToString();

actions[i] = numbers[1] = null;

temp = 0;

removeNulls();

}

else

do

{

if (actions[i] == "\*" || actions[i] == "/")

{

if (actions[i] == "\*") temp = Convert.ToDouble(numbers[i]) \* Convert.ToDouble(numbers[i + 1]);

else temp += Convert.ToDouble(numbers[i]) / Convert.ToDouble(numbers[i + 1]);

numbers[i] = temp.ToString();

actions[i] = numbers[i + 1] = null;

temp = 0;

removeNulls();

if (depth > 0) depth--;

}

foreach (var s in actions)

if (s == "\*" || s == "/") depth++;

} while (depth != 0);

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

Console.WriteLine(actions[i]);

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

Console.WriteLine(numbers[i]);

Console.WriteLine("Continue {0} to +-", result);

Console.ReadLine();

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

if (i == 0)

if (actions[i] == "+" || actions[i] == "-")

{

if (actions[i] == "+")

numbers[0] = (Convert.ToDouble(numbers[0]) + Convert.ToDouble(numbers[1])).ToString();

else numbers[0] = (Convert.ToDouble(numbers[0]) - Convert.ToDouble(numbers[1])).ToString();

actions[i] = numbers[1] = null;

removeNulls();

i--;

}

else

do

{

if (actions[i] == "+" || actions[i] == "-")

{

if (actions[i] == "+")

numbers[i] = (Convert.ToDouble(numbers[i]) + Convert.ToDouble(numbers[i + 1])).ToString();

else numbers[i] = (Convert.ToDouble(numbers[i]) - Convert.ToDouble(numbers[i + 1])).ToString();

actions[i] = numbers[i + 1] = null;

removeNulls();

if (depth > 0) depth--;

}

foreach (var s in actions)

if (s == "+" || s == "-") depth++;

} while (depth != 0);

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

Console.WriteLine("actions {0}", actions[i]);

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

Console.WriteLine("numbers {0}", numbers[i]);

} while (numbers.Length > 1);

Console.WriteLine("Result is {0}", numbers);

}

}

}