1. Время имеет формат часы:минуты. И часы, и минуты состоят из двух цифр, пример: 09:00. Напишите регулярное выражение для поиска времени в строке. – пример правильных выражений: Завтрак в 09:00. – пример неправильных выражений: 37:98, 24:00.

class Program

{

static void Main(string[] args)

{

string pattern = @"(2[0-3]|[0-1]\d):[0-5]\d";

string text = "09:00, 21:37, 26:40";

Regex newReg = new Regex(pattern);

MatchCollection matches = newReg.Matches(text);

foreach (Match mat in matches)

{

Console.WriteLine("Значение найденного обьекта {0}", mat.Value);

}

}

}

2. Написать регулярное выражение, определяющее является ли заданная строка правильным MAC-адресом. – пример правильных выражений: aE:dC:cA:56:76:54. – пример неправильных выражений: 01:23:45:67:89:Az.

class Program

{

static void Main(string[] args)

{

string pattern = string.Format (@"{0}:{0}:{0}:{0}:{0}:{0}", "[0-9ABCDEFabcdef]{2}");

string text = "aE:dC:cA:56:76:54. – пример неправильных выражений: 01:23:45:67:89:Az.";

Regex newReg = new Regex(pattern);

MatchCollection matches = newReg.Matches(text);

foreach (Match mat in matches)

{

Console.WriteLine("Значение найденного обьекта {0}", mat.Value);

}

}

}

3. Написать регулярное выражение, определяющее является ли данная строчка валидным E-mail адресом согласно RFC под номером 2822. – пример правильных выражений: user@example.com, root@localhost – пример неправильных выражений: bug@@@com.ru, @val.ru, Just Text2.

class Program

{

static void Main(string[] args)

{

string pattern = @"(\b\w+@[a-zA-Z\_]+\b)";

string text = "user@example.com, root@localhost – пример неправильных выражений: bug@@@com.ru, @val.ru, Just Text2.";

Regex newReg = new Regex(pattern);

MatchCollection matches = newReg.Matches(text);

foreach (Match mat in matches)

{

Console.WriteLine("Значение найденного обьекта {0}", mat.Value);

}

}

}

4. Написать регулярное выражение, определяющее является ли данная строчка датой в формате dd/mm/yyyy. Начиная с 1600 года до 9999 года. – пример правильных выражений: 29/02/2000, 30/04/2003, 01/01/2003. – пример неправильных выражений: 29/02/2001, 30-04-2003, 1/1/1899.

class Program

{

static void Main(string[] args)

{

string text = "29/02/2021, 29/02/2000, 31/12/9999";

Regex reg = new Regex (@"((0[1-9]|1\d|2[0-8])/(0[1-9]|1[0-2])/\d{4})|((29/(0[1-9]|1[0-2])/ ([02468][048]00|[13579][26]00|\d\d0[48]|\d\d[2468][048]|\d\d[13579][26])

)|(29/(0[13-9]|1[0-2])/\d{3}[1235679]))|(30/(0[13-9]|1[0-2])/\d{4})|

(31/(0[13578]|10|12)/\d{4})", RegexOptions.IgnorePatternWhitespace);

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

Console.WriteLine("Значение найденного обьекта {0}", mat.Value);

}

}

}

5. Есть текст с процентами. Извлечь из него значения процентов. Значения процентов брать с точностью до тясячных. – пример правильных выражений: 23.78 %. – пример неправильных выражений: 22 ps, 0.0002 %.

class Program

{

static void Main(string[] args)

{

string text = "65,367%";

Regex reg = new Regex(@"^(?:[0-9]\d{0,2}|0)?(?:\,\d{0,3})?%", RegexOptions.IgnorePatternWhitespace);

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

Console.WriteLine("Значение найденного обьекта {0}", mat.Value);

}

}

}

6. Проверить, надежно ли составлен пароль. Пароль считается надежным, если он состоит из 8 или более символов. Где символом может быть английская буква, цифра и знак подчеркивания. Пароль должен содержать хотя бы одну заглавную букву, одну маленькую букву и одну цифру. – пример правильных выражений: C00l\_Pass, SupperPas1. – пример неправильных выражений: Cool\_pass, C00l.

class Program

{

static void Main(string[] args)

{

string text = "Cool\_pass";

Regex reg = new Regex(@"^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[^\da-zA-Z]).{9,}$$", RegexOptions.IgnorePatternWhitespace);

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

Console.WriteLine("Значение найденного обьекта {0}", mat.Value);

}

}

}

Задание 2.

1.

public static class StringExtension

{

public static string RegexTime(this string text)

{

string pattern = @"(2[0-3]|[0-1]\d):[0-5]\d";

Regex reg = new Regex(pattern);

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

return (mat.Value);

}

return "0";

}

}

class Program

{

static void Main(string[] args)

{

string text = "09:00";

string i = text.RegexTime();

Console.WriteLine(i);

}

}

2.

public static class StringExtension

{

public static string RegexMAC(this string text)

{

string s1 = string.Format (@"{0}:{0}:{0}:{0}:{0}:{0}", "[0-9ABCDEFabcdef]{2}");

Regex reg = new Regex(s1, RegexOptions.IgnorePatternWhitespace);

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

return (mat.Value);

}

return "0";

}

}

class Program

{

static void Main(string[] args)

{

string text = "aE:dC:cA:56:76:54. – пример неправильных выражений: 01:23:45:67:89:Az.";

string i = text.RegexMAC();

Console.WriteLine(i);

}

}

3. public static class StringExtension

{

public static string RegexMail(this string text)

{

string pattern = @"(\b\w+@[a-zA-Z\_]+\b)";

Regex reg = new Regex(pattern);

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

return (mat.Value);

}

return "0";

}

}

class Program

{

static void Main(string[] args)

{

string text = "user@example.com";

string i = text.RegexMail();

Console.WriteLine(i);

}

}

4.

public static class StringExtension

{

public static string RegexDate(this string text)

{

Regex reg = new Regex(@"((0[1-9]|1\d|2[0-8])/(0[1-9]|1[0-2])/\d{4})|((29/(0[1-9]|1[0-2])/ ([02468][048]00|[13579][26]00|\d\d0[48]|\d\d[2468][048]|\d\d[13579][26])

)|(29/(0[13-9]|1[0-2])/\d{3}[1235679]))|(30/(0[13-9]|1[0-2])/\d{4})|

(31/(0[13578]|10|12)/\d{4})", RegexOptions.IgnorePatternWhitespace);

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

return (mat.Value);

}

return "0";

}

}

class Program

{

static void Main(string[] args)

{

string text = "29/02/2000";

string i = text.RegexDate();

Console.WriteLine(i);

}

}

5.

public static class StringExtension

{

public static string RegexPercent(this string text)

{

Regex reg = new Regex(@"^(?:[0-9]\d{0,2}|0)?(?:\,\d{0,3})?%", RegexOptions.IgnorePatternWhitespace);

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

return (mat.Value);

}

return "0";

}

}

class Program

{

static void Main(string[] args)

{

string text = "0,366%, 0595%";

string i = text.RegexPercent();

Console.WriteLine(i);

}

}

6.

public static class StringExtension

{

public static string RegexPassword(this string text)

{

Regex reg = new Regex(@"^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[^\da-zA-Z]).{9,}$$", RegexOptions.IgnorePatternWhitespace);

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

return (mat.Value);

}

return "0";

}

}

class Program

{

static void Main(string[] args)

{

string text = "C0ol\_pass";

string i = text.RegexPassword();

Console.WriteLine(i);

}

}