Практическая работа № 2.  
Регулярные выражения

**Цель практической работы**

Закрепление теоретических знаний по обработке текстов с помощью регулярных выражений.

**Постановка задачи**

**Регулярные выражения** представляют эффективный и гибкий метод по обработке больших текстов, позволяя в то же время существенно уменьшить объемы кода по сравнению с использованием стандартных операций со строками. Основная функциональность регулярных выражений в .NET сосредоточена в пространстве имен System.Text.RegularExpressions. А центральным классом при работе с регулярными выражениями является класс Regex.

Например, у нас есть текст "Бык тупогуб, тупогубенький бычок, у быка губа бела была тупа" и нам надо найти в нем все словоформы «туп»:

using System.Text.RegularExpressions;

var str = "Бык тупогуб, тупогубенький бычок, у быка губа бела была тупа";

var regex = new Regex(@"туп\w\*"); //Задаем шаблон:

var matches = regex.Matches(str);

if (matches.Count > 0)

{

foreach (Match match in matches)

Console.WriteLine(match.Value);

}

else Console.WriteLine("Совпадений не найдено");

В данном случае последовательность начинается с «туп», \w означает любой символ слова (любые буквы, а также знаки и цифры, если они являются частью слова, например, «А.13» будет считаться словом и все её символы будут подходить по шаблону \w).

Знак звездочки «\*» означает, что выражение левее него может повторяться 0 раз и более. В данном случае это относится к \w.

Класс символов соответствует любому из набора символов. Классы символов включают языковые элементы, перечисленные в следующей таблице.

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс** | **Описание** | **Шаблон** | **Ответ** |
| [ группа\_символов ] | Единичный символ из группы (чувствителен к регистру) | [ae] | "a" в "gray" "a", "e" в "lane" |
| [^ группа\_символов ] | НЕ единичный символ из группы (инверсия предыдущего примера) | [^aei] | "r", "g", "n" в "reign" |
| [ первый - последний ] | Диапазон символов | [A-Z] | "A", "B" в "AB123" |
| . (Знак точки) | Любой единичный символ | a.e | "ave" в "nave" "ate" в "water" |
| \p{ кодировка } | Любой единичный символ заданной кодировки | \p{Lu}  \p{IsCyrillic} | "C", "L" в "City Lights"  "Д", "Ж" в "ДЖem" |
| \s | Любой символ пробела | \w\s | "D " в "ID A1.3" |

Квантификаторы определяют количество экземпляров символа, группы или класса символов, которое должно присутствовать во входных данных, чтобы было зафиксировано совпадение. В следующей таблице перечислены квантификаторы, поддерживаемые .NET:

|  |  |  |  |
| --- | --- | --- | --- |
| **Квантор** | **Описание** | **Шаблон** | **Ответ** |
| \* | Предыдущий элемент повторяется 0 и более раз | a.\*c | "abcbc" в "abcbc" |
| + | 1 и более раз | "be+" | "bee" в "been", "be" в "bent" |
| ? | 0 или 1 раз | "rai?" | "rai" в "rain" |
| { *n* } | n раз | ",\d{3}" | ",043" в "1,043.6", ",876", ",543", и ",210" в "9,876,543,210" |
| { *n* ,} | n и более раз | "\d{2,}" | "166", "29", "1930" |
| { *n* , *m* } | от n до m раз включительно | "\d{3,5}" | "166", "17668" "19302" в "193024" |

**В жадном режиме** (по умолчанию) квантификатор повторяется столько раз, сколько это возможно.

Движок регулярного выражения пытается получить максимальное количество символов, соответствующих .+, а затем сокращает это количество символ за символом, если остаток шаблона не совпадает. Далее рассмотрим не жадные кванторы.

|  |  |  |  |
| --- | --- | --- | --- |
| **Квантор** | **Описание** | **Шаблон** | **Ответ** |
| \*? | Предыдущий элемент повторяется 0 и более раз | a.\*?c | "abc" в "abcbc" |
| +? | 1 и более раз | "be+?" | "be" в "been", "be" в "bent" |
| ?? | 0 или 1 раз | "rai??" | "ra" в "rain" |
| { *n* }? | n раз | ",\d{3}?" | ",043" в "1,043.6", ",876", ",543", и ",210" в "9,876,543,210" |
| { *n* ,}? | n и более раз | "\d{2,}?" | "166", "29", "1930" |
| { *n* , *m* }? | от n до m раз включительно | "\d{3,5}?" | "166", "17668"  "193", "024" в "193024" |

Привязки (или атомарные утверждения нулевой ширины) указывают положение в строке, где должно быть найдено соответствие. При использовании привязки в выражении поиска обработчик регулярных выражений не проходит по строке и не потребляет символы; он ищет соответствия только в заданном местоположении. Например, ^ указывает, что соответствие должно начаться в начале строки. Таким образом, регулярное выражение ^http: находит соответствие для http, только если этот элемент находится в начале строки. В таблице ниже перечислены привязки, поддерживаемые регулярными выражениями в .NET.

|  |  |  |  |
| --- | --- | --- | --- |
| **Привязка** | **Описание** | **Шаблон** | **Ответ** |
| ^ | Начало строки или строк в случае многострочного вывода. | ^\d{3} | "901" в "901-333-" |
| $ | Конец строки или строк в случае многострочного вывода. | -\d{3}$ | "-333" в "-901-333" |
| \A | Начало строки. | \A\d{3} | "901" в "901-333-" |
| \Z | Конец строки или до последнего символа новой строки. | -\d{3}\Z | "-333" в "-901-333" |
| \z | Конец строки. | -\d{3}\z | "-333" в "-901-333" |
| \G | Непрерывное соответствие. | \G\(\d\) | "(1)", "(3)", "(5)"  в "(1)(3)(5)[7](9) |
| \b | На границе слова | \b\w+\s\w+\b | "them theme", "them them" в "them theme them them" |

**Параметр RegexOptions.**

Regex regex = new Regex(@"туп(\w\*)", RegexOptions.IgnoreCase);

Regex regex = new Regex(@"туп(\w\*)", RegexOptions.Compiled | RegexOptions.IgnoreCase);

**Compiled**: при установке этого значения регулярное выражение компилируется в сборку, что обеспечивает более быстрое выполнение

**CultureInvariant**: при установке этого значения будут игнорироваться региональные различия

**IgnoreCase**: при установке этого значения будет игнорироваться регистр

**IgnorePatternWhitespace**: удаляет из строки пробелы и разрешает комментарии, начинающиеся со знака #

**Multiline**: указывает, что текст надо рассматривать в многострочном режиме. При таком режиме символы "^" и "$" совпадают, соответственно, с началом и концом любой строки, а не с началом и концом всего текста

**RightToLeft**: приписывает читать строку справа налево

**Singleline**: при данном режиме символ "." соответствует любому символу, в том числе последовательности "\n", которая осуществляет переход на следующую строку

**Пример** поиска номера в формате 111-111-1111:

string s = "456-435-2318";

Regex regex = new Regex(@"\d{3}-\d{3}-\d{4}");

Перепишем пример с номером телефона и явно укажем, какие символы там должны быть:

string s = "456-435-2318";

Regex regex = new Regex("[0-9]{3}-[0-9]{3}-[0-9]{4}");

С помощью операции | можно задать альтернативные символы, например:

Regex regex = new Regex(@"(2|3){3}-[0-9]{3}-\d{4}");

Первые три цифры могут содержать только двойки или тройки. Такой шаблон будет соответствовать, например, строкам "222-222-2222" и "323-435-2318". А вот строка "235-435-2318" уже не подпадает под шаблон, так как одной из трех первых цифр является цифра 5.

**Проверка** на соответствие строки формату, метод **IsMatch**.

Нередко возникает задача проверить корректность данных, введенных пользователем. Это может быть проверка электронного адреса, номера телефона, Класс Regex предоставляет статический метод IsMatch, который позволяет проверить входную строку с шаблоном на соответствие:

var strings = new[] { "example@mail.ru", "examplemail.ru", "example@mailru" };

Regex regex = new Regex(@"\w+@\w+\.\w+");

foreach (string s in strings)

if (regex.IsMatch(s))

Console.WriteLine(s);

**Замена символов, метод Replace.**

Класс Regex имеет метод Replace, который позволяет заменить строку, соответствующую регулярному выражению, другой строкой:

string text = "Мама мыла раму. ";

string pattern = @"\s+";

string target = " ";

Regex regex = new Regex(pattern);

string result = regex.Replace(text, target);

Console.WriteLine(result);

Данная версия метода Replace принимает два параметра: строку с текстом, где надо выполнить замену, и сама строка замены. Так как в качестве шаблона выбрано выражение "\s+ (то есть наличие одного и более пробелов), метод Replace проходит по всему тексту и заменяет несколько подряд идущих пробелов одинарными.

**Задание на практическую работу**

1. Создать форму регистрации.
2. Добавить проверку введенных данных на соответствие шаблонам с помощью регулярных выражений.
3. Регистрационная форма обязательно должна включать:

* Логин,
* Пароль,
* Электронная почта.

1. Регистрационная форма должна включать как минимум 3 из следующих параметров на выбор:

* ФИО,
* Дата рождения,
* Пол,
* Страна проживания,
* Номер телефона,
* Серия и номер паспорта,
* Почтовый индекс.

1. Защита работы включает демонстрацию работы программы, описание возможных входных данных.

Указания к работе

**1. Обязательные поля**

**1.1 Логин**

**Требования:**

* Разрешены **только латинские буквы (a-z, A-Z) и цифры (0-9)**.
* **Длина:** от 5 до 20 символов.
* **Не может начинаться с цифры**.
* **Запрещены спецсимволы** (кроме подчеркивания \_).

**Примеры:**

✅ User123, Admin\_Test

❌ 123User, !Login, User@

**1.2 Пароль**

**Требования:**

* **Минимум 8 символов.**
* **Обязательно:**
  + хотя бы **1 заглавная буква (A-Z)**,
  + хотя бы **1 цифра (0-9)**,
  + хотя бы **1 спецсимвол** (!@#$%^&\* и т. д.).
* **Запрещены пробелы.**
* **Не может состоять только из цифр.**

**Примеры:**  
✅ Pass123!, Secure#2024

❌ password, 12345678, Pass 123

**1.3 Электронная почта**

**Требования:**

* **Стандартный формат:** username@domain.tld.
* **Допустимые символы:**
  + Латиница (a-z, A-Z), цифры (0-9), точки ., подчеркивание \_, дефис -.
* **Домен:**
  + Минимум **2 символа** после точки (.com, .ru).
  + Запрещены **disposable-email** (типа temp-mail.org).

**Примеры:**

✅ user@example.com, [john.doe@mail.ru](mailto:john.doe@mail.ru)

❌ user@.com, user@temp-mail.org, user@site

**2. Дополнительные поля (выбрать 3-5)**

**2.1 ФИО**

**Требования:**

* **Кириллица или латиница** (в зависимости от системы).
* **Формат:**
  + Иванов Иван Иванович (3 слова, пробелы).
  + Допустимы дефисы (Анна-Мария).
* **Запрещены:** цифры, спецсимволы (кроме дефиса и пробела).

**Примеры:**

✅ Петров Петр, Smith John

❌ Иван123, John!Doe

**2.2 Дата рождения**

**Требования:**

* **Формат:** DD.MM.YYYY (например, 01.01.2000).
* **Минимальный возраст:** от 13 лет.
* **Максимальный возраст:** до 120 лет.
* **Проверка на реальность даты** (например, 30.02.2000 — недопустимо).

**Примеры:**

✅ 15.05.1995, 01.12.2010

❌ 32.01.2005, 01.01.1800

**2.3 Номер телефона**

**Требования:**

* **Формат:**
  + Международный: +7 (XXX) XXX-XX-XX (РФ).
  + Альтернативный: 8 XXX XXX-XX-XX.
* **Только цифры** (после очистки от скобок и дефисов).
* **Длина:** 10–12 цифр (включая код страны).

**Примеры:**  
✅ +7 (912) 345-67-89, 89123456789

❌ +7 912 345 67 89, 123-45-67

**2.4 Серия и номер паспорта**

**Требования (для РФ):**

* **Формат:** XX XX XXXXXX (4 цифры серии + 6 цифр номера).
* **Проверка на валидность:**
  + Серия не может быть 0000.
  + Номер не может быть 000000.

**Примеры:**

✅ 12 34 567890

❌ 00 00 000000, 1234 567890

**2.5 Почтовый индекс**

**Требования (для РФ):**

* **Ровно 6 цифр.**
* **Не может начинаться с**0**.**
* **Допустимые диапазоны:**
  + 101000–901999 (основные индексы).

**Примеры:**

✅ 123456, 654321

❌ 012345, 1234567

**Пример разработки приложения**

**Шаг 1: Создание проекта Windows Forms (.NET 8)**

1. Откройте Visual Studio
2. Создайте новый проект:
   * Выберите "Windows Forms App" (.NET 8.0)
   * Назовите проект "RegistrationFormApp"
3. Убедитесь, что выбран .NET 8.0 в выпадающем списке

**Шаг 2: Дизайн формы регистрации**

1. Откройте Form1.cs в конструкторе форм
2. Добавьте следующие элементы:
   * Label "Логин"
   * TextBox для ввода логина (назовите txtLogin)
   * Button "Зарегистрироваться" (назовите btnRegister)
   * Label для вывода сообщений об ошибках (назовите lblError, установите ForeColor в Red)

**Шаг 3: Код формы (Form1.cs)**

using System.Text.RegularExpressions;

namespace RegistrationFormApp

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void btnRegister\_Click(object sender, EventArgs e)

{

if (ValidateLogin(txtLogin.Text))

{

lblError.Text = "Регистрация успешна!";

lblError.ForeColor = Color.Green;

// Здесь можно добавить логику сохранения данных

}

}

private bool ValidateLogin(string login)

{

// Регулярное выражение для проверки логина

string pattern = @"^[a-zA-Z][a-zA-Z0-9\_]{4,19}$";

if (string.IsNullOrWhiteSpace(login))

{

lblError.Text = "Логин не может быть пустым";

lblError.ForeColor = Color.Red;

return false;

}

if (!Regex.IsMatch(login, pattern))

{

lblError.Text = "Логин не соответствует требованиям";

lblError.ForeColor = Color.Red;

return false;

}

return true;

}

private void txtLogin\_TextChanged(object sender, EventArgs e)

{

lblError.Text = string.Empty;

}

}

}

**Шаг 4: Код дизайнера формы (Form1.Designer.cs)**

namespace RegistrationFormApp

{

partial class Form1

{

private System.ComponentModel.IContainer components = null;

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

private void InitializeComponent()

{

this.label1 = new System.Windows.Forms.Label();

this.txtLogin = new System.Windows.Forms.TextBox();

this.btnRegister = new System.Windows.Forms.Button();

this.lblError = new System.Windows.Forms.Label();

this.SuspendLayout();

// label1

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(20, 20);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(47, 20);

this.label1.TabIndex = 0;

this.label1.Text = "Логин:";

// txtLogin

this.txtLogin.Location = new System.Drawing.Point(80, 17);

this.txtLogin.Name = "txtLogin";

this.txtLogin.Size = new System.Drawing.Size(250, 27);

this.txtLogin.TabIndex = 1;

this.txtLogin.TextChanged += new System.EventHandler(this.txtLogin\_TextChanged);

// btnRegister

this.btnRegister.Location = new System.Drawing.Point(80, 60);

this.btnRegister.Name = "btnRegister";

this.btnRegister.Size = new System.Drawing.Size(150, 35);

this.btnRegister.TabIndex = 2;

this.btnRegister.Text = "Зарегистрироваться";

this.btnRegister.UseVisualStyleBackColor = true;

this.btnRegister.Click += new System.EventHandler(this.btnRegister\_Click);

// lblError

this.lblError.AutoSize = true;

this.lblError.ForeColor = System.Drawing.Color.Red;

this.lblError.Location = new System.Drawing.Point(80, 110);

this.lblError.Name = "lblError";

this.lblError.Size = new System.Drawing.Size(0, 20);

this.lblError.TabIndex = 3;

// Form1

this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 20F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(350, 150);

this.Controls.Add(this.lblError);

this.Controls.Add(this.btnRegister);

this.Controls.Add(this.txtLogin);

this.Controls.Add(this.label1);

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

this.Name = "Form1";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Регистрация";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private Label label1;

private TextBox txtLogin;

private Button btnRegister;

private Label lblError;

}

}

**Шаг 5: Объяснение регулярного выражения**

Регулярное выражение для проверки логина:

^[a-zA-Z][a-zA-Z0-9\_]{4,19}$

Разберем его по частям:

* ^ - начало строки
* [a-zA-Z] - первым символом должна быть буква (латиница)
* [a-zA-Z0-9\_] - последующие символы: буквы, цифры или подчеркивание
* {4,19} - количество символов после первого (итого от 5 до 20)
* $ - конец строки

**Шаг 6: Дополнительные улучшения**

1. **Подсказки при наведении**:

*// В конструкторе формы*

toolTip1.SetToolTip(txtLogin,

"Логин должен:

- Начинаться с буквы

- Содержать только A-Z, a-z, 0-9 и \_

- Быть длиной 5-20 символов");

1. **Проверка в реальном времени**:

private void txtLogin\_TextChanged(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtLogin.Text))

{

lblError.Text = "";

return;

}

string pattern = @"^[a-zA-Z][a-zA-Z0-9\_]{0,19}$";

if (!Regex.IsMatch(txtLogin.Text, pattern) || txtLogin.Text.Length < 5)

{

lblError.Text = "Некорректный логин";

lblError.ForeColor = Color.Red;

}

else

{

lblError.Text = "Логин корректный";

lblError.ForeColor = Color.Green;

}

}

1. **Добавление иконки предупреждения**:

// Добавьте PictureBox (picWarning) рядом с полем ввода

private void UpdateValidationUI(bool isValid)

{

picWarning.Visible = !isValid;

if (isValid)

{

txtLogin.BackColor = SystemColors.Window;

}

else

{

txtLogin.BackColor = Color.LightPink;

}

}

**Шаг 7: Тестирование формы**

Проверьте следующие случаи:

1. Корректные логины:
   * "User123" (должно быть принято)
   * "Admin\_Test" (должно быть принято)
2. Некорректные логины:
   * "123User" (начинается с цифры)
   * "!Login" (содержит спецсимвол)
   * "Us" (слишком короткий)
   * "VeryLongLoginNameThatExceedsLimit" (слишком длинный)

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»