Krzysztof Kadowski, kkadowski@gmail.com

Przedmiot:	Programowanie aplikacji desktopowych i mobilnych
Klasa:	II TPM
Temat:	Regex w C#

1. Wstęp

Regular Expressions (Regex) to potężne narzędzie do przetwarzania tekstu, pozwalające na wyszukiwanie, dopasowywanie i manipulowanie tekstem na podstawie wzorców. W C# Regex jest dostępny dzięki przestrzeni nazw System.Text.RegularExpressions.

2. Podstawowa składnia Regex

Znaki specjalne:

- . - dowolny znak (poza nową linią)
- ^ - początek linii
- $ - koniec linii
- * - zero lub więcej wystąpień poprzedniego znaku
- + - jedno lub więcej wystąpień poprzedniego znaku
- ? - zero lub jedno wystąpienie poprzedniego znaku
- \d - dowolna cyfra (0-9)
- \w - dowolny znak alfanumeryczny (litera, cyfra, podkreślnik)
- \s - dowolny biały znak (spacja, tabulator, nowa linia)
- [abc] - dowolny znak z zestawu a, b, c
- [^abc] - dowolny znak spoza zestawu a, b, c
- (abc) - grupa, pozwala na łączenie wielu wyrażeń

Modyfikatory:

- `*` - 0 lub więcej wystąpień.
- `+` - 1 lub więcej wystąpień.
- `?` - 0 lub 1 wystąpienie.
- `{n}` - Dokładnie n wystąpień.
- `{n,}` - Co najmniej n wystąpień.
- `{n,m}` - Od n do m wystąpień.

3. Użycie Regex w C#

- Tworzenie obiektu Regex

```
using System.Text.RegularExpressions;

Regex regex = new Regex(@"pattern");
```

Podstawowe metody Regex

- IsMatch(string input) – sprawdza, czy tekst pasuje do wzorca.
- Match(string input) – zwraca pierwsze dopasowanie.
- Matches(string input) – zwraca wszystkie dopasowania.
- Replace(string input, string replacement) – zastępuje dopasowane fragmenty tekstu.

4. Przykłady zastosowań

Przykład 1. Sprawdzanie czy wpisany kod pocztowy jest poprawny *[kod_01.cs]*

```
using System;
using System.Text.RegularExpressions;
using System.Text;
class Program
{
    static void Main()
    {
        Console.OutputEncoding = System.Text.Encoding.UTF8;
        string pattern = @"^\d{2}-\d{3}$"; // Wzorzec dla kodu pocztowego XX-XXX
        string input = "00-123"; // Przykładowy kod pocztowy

        bool isValid = Regex.IsMatch(input, pattern);

        if (isValid)
        {
            Console.WriteLine("Kod pocztowy jest poprawny.");
        }
        else
        {
            Console.WriteLine("Kod pocztowy jest niepoprawny.");
        }
    }
}

```

Przykład 2. Sprawdzanie poprawności adresu email *[kod_02.cs]*

```
using System;
using System.Text.RegularExpressions;

class Program
{
    static void Main()
    {
        string emailPattern = @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$";
        string[] testEmails = { "test@example.com", "invalid-email",
            "user@sub.domain.co", "user.name+tag@domain.com" };

        foreach (var email in testEmails)
        {
            bool isValid = Regex.IsMatch(email, emailPattern);
            Console.WriteLine($"{email} is {(isValid ? "valid" : "invalid")}");
        }
    }
}
```

Przykład 3. Wyodrębnianie numerów telefonu *[kod_03.cs]*

```cs
using System;
using System.Text.RegularExpressions;

class Program
{
    static void Main()
    {
        string text = "Kontakt: 123-456-7890, 987-654-3210";
        string pattern = @"\d{3}-\d{3}-\d{4}";

        Regex regex = new Regex(pattern);
        MatchCollection matches = regex.Matches(text);

        Console.WriteLine("Znalezione numery telefonu:");
        foreach (Match match in matches)
        {
            Console.WriteLine(match.Value);
        }
    }
}
```

Przykład 4. Zamiana daty z formatu YYYY-MM-DD na DD-MM-YYYY *kod_04.cs]*

```cs
using System;
using System.Text.RegularExpressions;

class Program
{
    static void Main()
    {
        string date = "2024-09-04";
        string pattern = @"(\d{4})-(\d{2})-(\d{2})";
        string replacement = "$3-$2-$1";

        Regex regex = new Regex(pattern);
        string newDate = regex.Replace(date, replacement);

        Console.WriteLine($"Nowy format daty: {newDate}");
    }
}
```

Przykład 5. Usuwanie zbędnych spacji *[kod_05.cs]*

```
using System;
using System.Text.RegularExpressions;

class Program
{
    static void Main()
    {
        string text = "  C#   jest    fajnym     jezykiem    !   ";
        string pattern = @"\s+";

        Regex regex = new Regex(pattern);
        string result = regex.Replace(text, " ");

        Console.WriteLine($"Tekst bez zbędnych spacji: '{result.Trim()}'");
    }
}
```

Przykład 6. Zamiana tekstu *[kod_06.cs]*

```
using System;
using System.Text.RegularExpressions;
using System.Text;
class Program
{
    static void Main()
    {
        Console.OutputEncoding = System.Text.Encoding.UTF8;
        string text = "Słowo: kot, słowo: pies, słowo: ptak.";
        string replacedText = Regex.Replace(text, @"kot|pies|ptak", "zwierzak");
        Console.WriteLine(replacedText);
    }
}
```

Przykład 7. Dzielenie tekstu na podstawie separatorów *[kod_07.cs]*

```
using System;
using System.Text.RegularExpressions;
using System.Text;
class Program
{
    static void Main()
    {
        Console.OutputEncoding = System.Text.Encoding.UTF8;
        string csv = "jabłko,banan,winogrono";
        string[] fruits = Regex.Split(csv, @",");

        foreach (string fruit in fruits)
            {
                Console.WriteLine(fruit);
            }
    }
}
```

Zadania

Zadanie 1. Napisz regex, który znajduje wszystkie daty w formacie dd-mm-yyyy w liście dat. *[odp_01.cs]*

Zadanie 2. Napisz regex do weryfikacji numeru PESEL.*[odp_02.cs]*

Zadanie 3. Stwórz funkcję, która zamienia wszystkie wystąpienia wielokrotnych spacji w jednym ciągu na pojedynczą spację. *[odp_03.cs]*

Zadanie 4. Napisz wyrażenie regularne, które dopasowuje wszystkie słowa zaczynające się od dużej litery w zdaniu. *[odp_04.cs]*

Zadanie 5. Stwórz Regex do wyodrębniania wszystkich adresów URL z tekstu. *[odp_05.cs]*

Zadanie 6. Zaimplementuj funkcję, która waliduje numery kart kredytowych (np. w formacie XXXX-XXXX-XXXX-XXXX). *[odp_06.cs]*

Algorytm Luhna walidacji kart kredytowych:

- *Usuwamy myślniki z numeru karty za pomocą Replace("-", "").*
- *Algorytm Luhna jest używany do sprawdzenia poprawności numeru karty:*
  - *Liczby są analizowane od prawej do lewej.*
  - *Co druga liczba jest mnożona przez 2. Jeśli wynik jest większy od 9, odejmujemy 9.*
  - *Sumujemy wszystkie liczby.*
  - *Jeśli suma jest podzielna przez 10, numer karty jest poprawny.*