Pedro Inácio Rodrigues Pontes

Prática 9: Regex

Belo Horizonte, Brasil

2025

1 Introdução

O objetivo deste projeto é desenvolver duas aplicações distintas em C# que demonstrem conceitos fundamentais de Regex. A primeira aplicação consiste em um programa de console que implementa um sistema de validação de senhas com critérios de segurança rigorosos, solicitando ao usuário a digitação repetida de senhas até que uma senha considerada "forte"seja fornecida. O sistema deve verificar se a senha inserida atende aos seguintes requisitos: possuir entre 7 e 16 caracteres, conter pelo menos uma letra minúscula (a-z), uma letra maiúscula (A-Z), um dígito (0-9) e um caractere especial dentre os seguintes símbolos:

```
! @ # $ % ^ & * ( ) + = _ - { } [ ] : ; " ' ? < > , .
```

Para essa validação, será utilizada a expressão regular:

```
@"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)
(?=.*[!@#$%^&*()+=_\-{}\[\]:;""'?<>,.]).{7,16}$"
```

cujo funcionamento e estrutura serão detalhadamente explicados. A segunda aplicação envolve o processamento de dados em formato JSON contendo informações sobre os Prêmios Nobel, com foco específico na extração e listagem dos primeiros nomes (firstname) dos ganhadores do prêmio de economia (categoria "economics"). Ambas as aplicações demonstram práticas essenciais de desenvolvimento em C, incluindo validação de entrada de dados, uso de expressões regulares, manipulação de arquivos JSON e tratamento adequado de erros, proporcionando uma base sólida para o entendimento de conceitos fundamentais de programação orientada a objetos e processamento de dados estruturados.

2 Desenvolvimento

2.1 Validação de Senhas

O programa de validação de senhas implementa um sistema que utiliza uma expressão regular para verificar critérios de segurança rigorosos, executando um loop contínuo que solicita senhas ao usuário até que uma senha considerada forte seja fornecida. O código utiliza a biblioteca System.Text.RegularExpressions e define uma variável estática para armazenar a senha digitada pelo usuário.

```
using System.Text.RegularExpressions;
class Program
{
    static string? Password;
    public static void Main(string[] args)
    {
```

```
        string regexPattern =
           @"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)
        (?=.*[!@#$%^&*()+=_\-{}\[\]:;""'?<>,.]).{7,16}$";
        while (true)
        {
            Console.WriteLine("Digite a senha a ser
               verificada:");
            Password = Console.ReadLine();
            if (Regex.IsMatch(Password, regexPattern))
            {
                Console.WriteLine("Senha forte!");
                return;
            }
            Console.WriteLine("Senha fraca, tente
               novamente");
        }
    }
}
```

A expressão regular utilizada é composta por várias partes que trabalham em conjunto para garantir todos os critérios de segurança estabelecidos. O símbolo circunflexo no início da expressão indica o começo da string, enquanto o símbolo de cifrão ao final indica o término, garantindo que toda a string seja validada do início ao fim. Os lookaheads positivos, representados pela sintaxe (?=...), são elementos fundamentais que verificam condições específicas sem consumir caracteres da string, permitindo que múltiplas validações sejam aplicadas simultaneamente sobre a mesma entrada. O primeiro lookahead (?=.*[a-z]) verifica se existe pelo menos uma letra minúscula em qualquer posição da string, seguido por (?=.*[A-Z]) que confirma a presença de pelo menos uma letra maiúscula. O terceiro lookahead `(?=.*\d)` assegura que há pelo menos um dígito numérico, enquanto o quarto lookahead verifica a existência de pelo menos um caractere especial da lista definida entre os colchetes. Por fim, a expressão `.{7,16}` corresponde a qualquer caractere repetido entre 7 e 16 vezes, definindo assim o comprimento mínimo e máximo aceitável para a senha. Este mecanismo de lookaheads é essencial pois permite que todos os critérios sejam verificados independentemente da ordem em que os caracteres aparecem na senha digitada pelo usuário.

2.2 Nome dos Vencendores do Nobel de Economia

A segunda aplicação desenvolvida realiza o processamento de dados JSON contendo informações sobre os Prêmios Nobel, com o objetivo específico de extrair os primeiros nomes dos ganhadores do prêmio de economia. O programa utiliza técnicas avançadas de manipulação

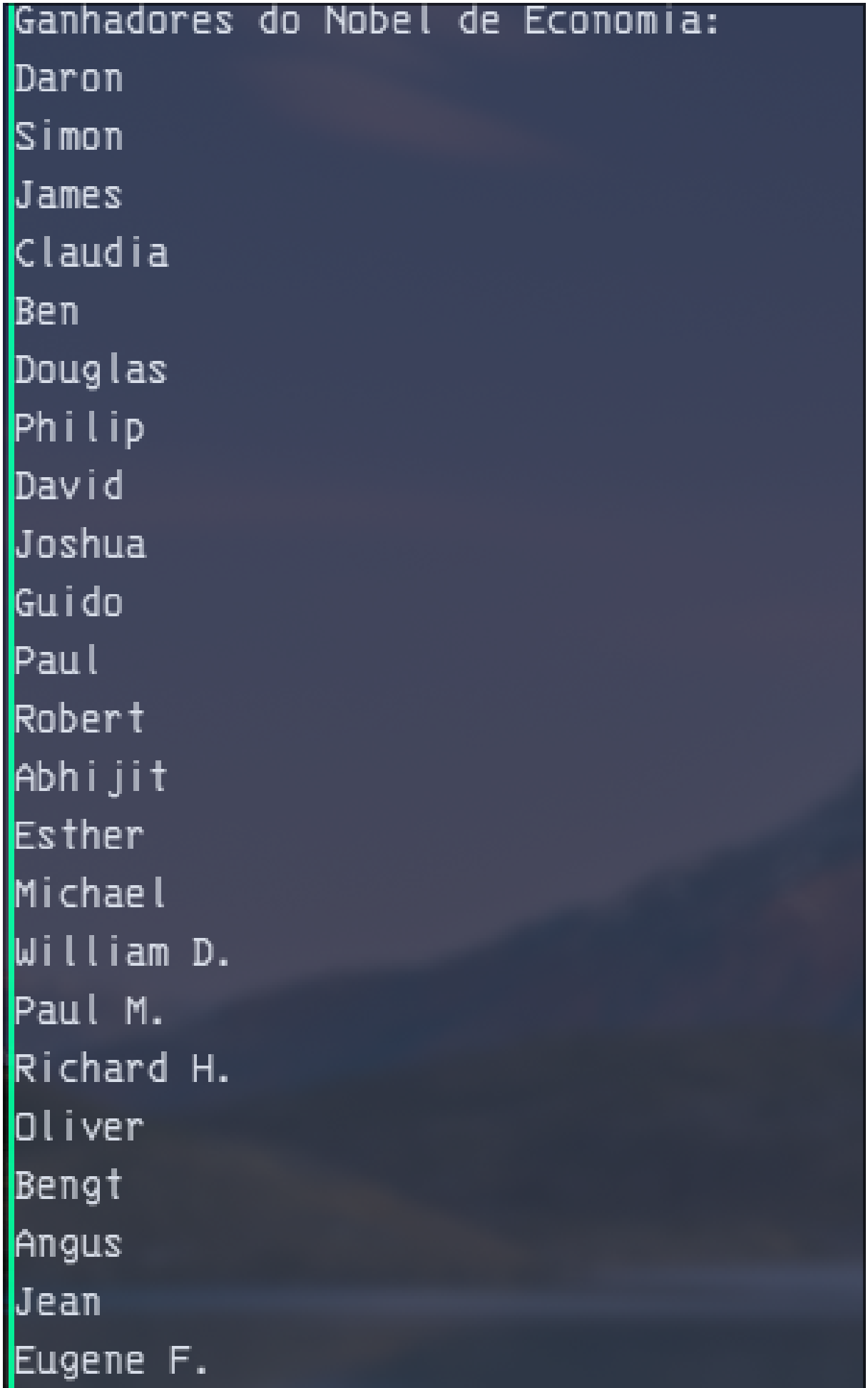
de strings através de expressões regulares para analisar a estrutura hierárquica do arquivo JSON e localizar informações específicas dentro de categorias determinadas.

```
using System.Text.RegularExpressions;
string json = File.ReadAllText("prize.json");
string pattern =
   @"""category"":""economics"".*?""laureates"":\s*\[(.*?)\]";
MatchCollection matches = Regex.Matches(json, pattern);
List<string> names = new List<string>();
foreach (Match match in matches)
{
    string contentLaureates = match.Groups[1].Value;
    string namePattern = @"""firstname"":""([^""]+)""";
    MatchCollection namesMatches =
       Regex.Matches(contentLaureates, namePattern);
    foreach (Match nameMatch in namesMatches)
    {
        string name = nameMatch.Groups[1].Value;
        names.Add(name);
    }
}
Console.WriteLine("Ganhadores do Nobel de Economia:");
foreach (string name in names)
{
    Console.WriteLine(name);
}
```

O programa inicia carregando todo o conteúdo do arquivo prize.json na memória através do método File.ReadAllText. A primeira expressão regular utilizada busca especificamente por seções do JSON que contenham a categoria "economics", utilizando o padrão que localiza a palavra "category"seguida de "economics"e posteriormente captura todo o conteúdo da seção "laureates"correspondente. A expressão utiliza grupos de captura, onde a sintaxe (.*?) dentro dos parênteses captura de forma não-gulosa todo o conteúdo entre os colchetes que definem o array de laureados. Para cada correspondência encontrada, o programa extrai o conteúdo capturado pelo primeiro grupo e aplica uma segunda expressão regular específica para localizar os primeiros nomes. O padrão utilizado para extrair os nomes busca pela chave "firstname"seguida de dois pontos e aspas, capturando qualquer sequência de caracteres que não sejam aspas através da classe de caracteres negativa [^"]. Esta abordagem garante que apenas o valor do primeiro nome seja extraído, excluindo as aspas delimitadoras. O programa utiliza uma estrutura de dados List para armazenar todos os nomes encontrados e posteriormente os

exibe no console de forma organizada, precedidos por um cabeçalho identificativo.

3 Resultados



```
George A.
A. Michael
Joseph E.
James J.
Daniel L.
Robert
Amartya
Robert C.
Myron
James A.
William
Robert E.
John C.
```

Figura 1 – Vencedores do Nobel de Economia

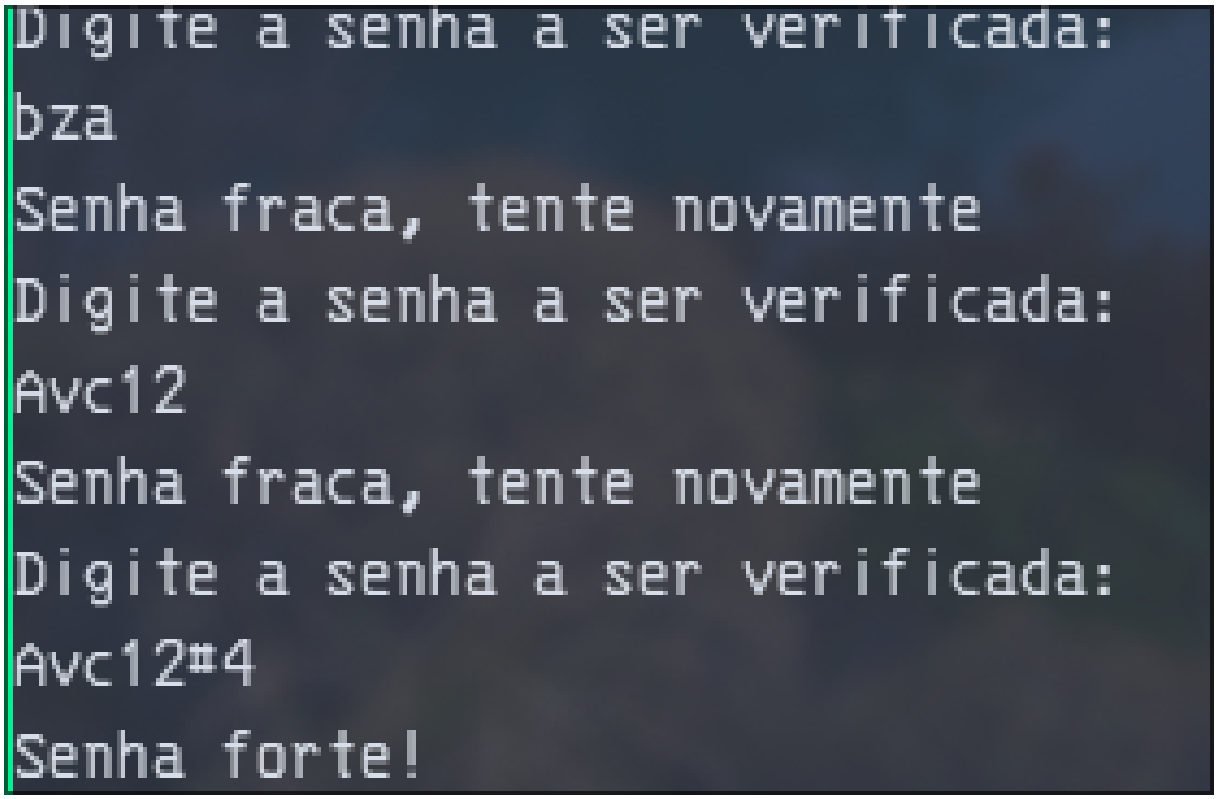


Figura 2 – Verificação de Senhas

4 Conclusão

O desenvolvimento das duas aplicações em C demonstrou a eficácia das expressões regulares na validação de dados e no processamento de informações estruturadas. O programa de validação de senhas implementou com sucesso critérios rigorosos de segurança através de lookaheads positivos, garantindo que todas as condições sejam verificadas simultaneamente. A aplicação de processamento JSON mostrou como expressões regulares podem ser utilizadas para extrair dados específicos de arquivos estruturados, permitindo a localização e captura precisa dos primeiros nomes dos ganhadores do Nobel de Economia. Ambos os projetos evidenciaram a importância do tratamento adequado de strings e da aplicação de técnicas de programação robustas para manipulação de dados, consolidando conhecimentos fundamentais sobre processamento de arquivos e uso de expressões regulares em aplicações práticas de console.