ALUNO: Guilherme Gomes de Brites
MATRICULA: 808721

ESTRUTURA DE DADOS BÁSICAS LINEARES

1)
```
using System;
using System.Collections;

class Fila {
    private ArrayList fila = new ArrayList();

    public void Inserir(object elemento)
    {
        fila.Add(elemento);
    }
    public void Remover()
    {
        if (fila.Count > 0)
        {
            fila.RemoveAt(0);
        }
else
        {
            Console.WriteLine("A fila está vazia. Não é possível remover elementos.");
        }
    }
    public void MostrarElementos()
    {
        Console.WriteLine("Elementos na fila:");
foreach (object elemento in fila)
        {
            Console.WriteLine(elemento);
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        Fila minhaFila = new Fila();

        minhaFila.Inserir("Primeiro elemento");
minhaFila.Inserir("Segundo elemento");
minhaFila.Inserir("Terceiro elemento");

        minhaFila.MostrarElementos();
```

```
        minhaFila.Remover();
minhaFila.MostrarElementos();
    }
}


2)  using
System; using
System.Collec
tions;

class Pilha
{
    private ArrayList pilha = new ArrayList();

    public void Inserir(object elemento)
    {
        pilha.Add(elemento);
    }
    public void Remover()
    {
        if (pilha.Count > 0)
        {
            pilha.RemoveAt(pilha.Count - 1);
        }
else
        {
            Console.WriteLine("A pilha está vazia. Não é possível remover
elementos.");
        }
    }
    public void MostrarElementos()
    {
        Console.WriteLine("Elementos na pilha (do topo para o fundo):");
for (int i = pilha.Count - 1; i >= 0; i--)
        {
            Console.WriteLine(pilha[i]);
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        Pilha minhaPilha = new Pilha();

        minhaPilha.Inserir("Primeiro elemento");
minhaPilha.Inserir("Segundo elemento");
minhaPilha.Inserir("Terceiro elemento");

        minhaPilha.MostrarElementos();

        minhaPilha.Remover();
minhaPilha.MostrarElementos();
    }
```

```
}

3)  using
System;
using System.Collections.Generic;

class Fila {
    private Stack<object> entrada = new Stack<object>();
private Stack<object> saida = new Stack<object>();     public
void Inserir(object elemento)
    {
        entrada.Push(elemento);
    }
    public void Remover()
    {
        if (saida.Count == 0)
        {
            while (entrada.Count > 0)
            {
                saida.Push(entrada.Pop());
            }
        }
        if (saida.Count > 0)
        {
            saida.Pop();
        }
else
        {
            Console.WriteLine("A fila está vazia. Não é possível remover
elementos.");
        }
    }
    public void MostrarElementos()
    {
        if (saida.Count == 0)
        {
            while (entrada.Count > 0)
            {
                saida.Push(entrada.Pop());
            }
        }

        Console.WriteLine("Elementos na fila:");
foreach (object elemento in saida)
        {
            Console.WriteLine(elemento);
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        Fila minhaFila = new Fila();
```

```
        minhaFila.Inserir("Primeiro elemento");
minhaFila.Inserir("Segundo elemento");
minhaFila.Inserir("Terceiro elemento");

        minhaFila.MostrarElementos();

        minhaFila.Remover();
minhaFila.MostrarElementos();     }
}
```

4) using

```
System;
using System.Collections.Generic;

class Pilha
{
    private Queue<object> fila1 = new Queue<object>();
private Queue<object> fila2 = new Queue<object>();

    public void Inserir(object elemento)
    {
        if (fila1.Count == 0)
        {
            fila1.Enqueue(elemento);
while (fila2.Count > 0)
            {
                fila1.Enqueue(fila2.Dequeue());
            }
}         else
{
            fila2.Enqueue(elemento);
while (fila1.Count > 0)
            {
                fila2.Enqueue(fila1.Dequeue());
            }
        }
    }
    public void Remover()
    {
        if (fila1.Count > 0)
        {
            fila1.Dequeue();
        }
        else if (fila2.Count > 0)
        {
            fila2.Dequeue();
        }
else
        {
            Console.WriteLine("A pilha está vazia. Não é possível remover
elementos.");
        }
    }
    public void MostrarElementos()
    {
        Queue<object> filaAtual = fila1.Count > 0 ? fila1 : fila2;
```

```
        Console.WriteLine("Elementos na pilha (do topo para o fundo):");
foreach (object elemento in filaAtual)
        {
            Console.WriteLine(elemento);
        }
    }
}

class Program
{
    static void Main(string[] args)
    {
        Pilha minhaPilha = new Pilha();

        minhaPilha.Inserir("Primeiro elemento");
minhaPilha.Inserir("Segundo elemento");
minhaPilha.Inserir("Terceiro elemento");

        minhaPilha.MostrarElementos();

        minhaPilha.Remover();
minhaPilha.MostrarElementos();
    }
}
```

EXERCICIOS LISTA LINEAR

1)

```
using
System;
using System.Collections.Generic;

class ListaFlexivel<T>
{
    private List<T> lista = new List<T>();

    public void Inserir(T elemento)
    {
        lista.Add(elemento);
    }
    public void Remover(T elemento)
    {
        lista.Remove(elemento);
    }
    public void MostrarElementos()
    {
        Console.WriteLine("Elementos na lista:");
foreach (T elemento in lista)
        {
            Console.WriteLine(elemento);
        }
    }
    public T CalcularSoma()
    {
```

```
        dynamic soma = default(T);
foreach (T elemento in lista)
        {
            if (elemento is int || elemento is double || elemento is
float)             {
                soma += elemento;
            }
        }
        return soma;
    }
}
class Program
{
    static void Main(string[] args)
    {
        ListaFlexivel<int> minhaLista = new ListaFlexivel<int>();

        minhaLista.Inserir(5);
minhaLista.Inserir(10);
minhaLista.Inserir(15);

        minhaLista.MostrarElementos();

        var soma = minhaLista.CalcularSoma();
        Console.WriteLine($"Soma dos elementos na lista: {soma}");
    }
}

2)  using
System;
using System.Collections.Generic;

class ListaFlexivel<T> where T : IComparable<T>
{
    private List<T> lista = new List<T>();

    public void Inserir(T elemento)
    {
        lista.Add(elemento);
    }
    public void Remover(T elemento)
    {
        lista.Remove(elemento);
    }
    public void MostrarElementos()
    {
        Console.WriteLine("Elementos na lista:");
foreach (T elemento in lista)
        {
            Console.WriteLine(elemento);
        }
    }
    public T EncontrarMaiorElemento()
    {
        if (lista.Count == 0)
        {
```

```
            throw new InvalidOperationException("A lista está vazia. Não
é possível encontrar o maior elemento.");
        }

        T maiorElemento = lista[0];
foreach (T elemento in lista)
        {
            if (elemento.CompareTo(maiorElemento) > 0)
            {
                maiorElemento = elemento;
            }
}
        return maiorElemento;
    }
}
class Program
{
    static void Main(string[] args)
    {
        ListaFlexivel<int> minhaLista = new ListaFlexivel<int>();

        minhaLista.Inserir(5);
minhaLista.Inserir(10);
minhaLista.Inserir(15);

        minhaLista.MostrarElementos();

        var maiorElemento = minhaLista.EncontrarMaiorElemento();
        Console.WriteLine($"O maior elemento na lista é:
{maiorElemento}");
    }
}

3)   using
System;
using System.Collections.Generic;

class ListaFlexivel<T>
{
    private List<T> lista = new List<T>();

    public void Inserir(T elemento)
    {
        lista.Add(elemento);
    }
    public void Remover(T elemento)
    {
        lista.Remove(elemento);
    }
    public void MostrarElementos()
    {
        Console.WriteLine("Elementos na lista:");
foreach (T elemento in lista)
        {
            Console.WriteLine(elemento);
        }
```

```
    }
    public void InverterOrdem()
    {
        lista.Reverse();
    }
}

class Program
{
    static void Main(string[] args)
    {
        ListaFlexivel<int> minhaLista = new ListaFlexivel<int>();

        minhaLista.Inserir(5);
minhaLista.Inserir(10);
minhaLista.Inserir(15);

        Console.WriteLine("Ordem original:");
minhaLista.MostrarElementos();

        minhaLista.InverterOrdem();

        Console.WriteLine("Ordem invertida:");
minhaLista.MostrarElementos();
    }
}

4)  using
System;
using System.Collections.Generic;

class ListaFlexivel<T>
{
    private List<T> lista = new List<T>();

    public void Inserir(T elemento)
    {
        lista.Add(elemento);
    }
    public void Remover(T elemento)
    {
        lista.Remove(elemento);
    }
    public void MostrarElementos()
    {
        Console.WriteLine("Elementos na lista:");
foreach (T elemento in lista)
        {
            Console.WriteLine(elemento);
        }
    }
    public int ContarParesEMultiplosDeCinco()
    {
        int contador = 0;
        foreach (T elemento in lista)
        {
```

```
            if (elemento is int && (int)(object)elemento % 2 == 0 &&
(int)(object)elemento % 5 == 0)
            {
                contador++;
            }
        }
        return contador;
    }
}
class Program
{
    static void Main(string[] args)
    {
        ListaFlexivel<int> minhaLista = new ListaFlexivel<int>();

        minhaLista.Inserir(10);
minhaLista.Inserir(15);         minhaLista.Inserir(20);
minhaLista.Inserir(25);

        minhaLista.MostrarElementos();

        int contador = minhaLista.ContarParesEMultiplosDeCinco();
Console.WriteLine($"Número de elementos pares e múltiplos de cinco na
lista: {contador}");
    }
}

5)
using System;

class ListaLinearSimples<T>
{
    private class No
    {
        public T Dado;
public No Proximo;

        public No(T dado)
        {
            Dado = dado;
            Proximo = null;
        }
    }
    private No primeiro;

    public void Inserir(T dado)
    {
        No novoNo = new No(dado);

        if (primeiro == null)
        {
            primeiro = novoNo;
        }
else
        {
```

```
            No atual = primeiro;
while (atual.Proximo != null)
            {
                atual = atual.Proximo;
            }
            atual.Proximo = novoNo;
        }
    }
    public void Remover(T dado)
    {
        if (primeiro == null)
        {
            return;
        }
        if (primeiro.Dado.Equals(dado))
        {
            primeiro = primeiro.Proximo;
return;
        }

        No atual = primeiro;
        while (atual.Proximo != null && !atual.Proximo.Dado.Equals(dado))
        {
            atual = atual.Proximo;
        }
        if (atual.Proximo != null)
        {
            atual.Proximo = atual.Proximo.Proximo;
        }
    }
    public void MostrarElementos()
    {
        Console.WriteLine("Elementos na lista:");
No atual = primeiro;         while (atual !=
null)
        {
            Console.WriteLine(atual.Dado);
atual = atual.Proximo;
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        ListaLinearSimples<int> minhaLista = new
ListaLinearSimples<int>();

        minhaLista.Inserir(5);
minhaLista.Inserir(10);
minhaLista.Inserir(15);

        minhaLista.MostrarElementos();

        minhaLista.Remover(10);
```

```
        minhaLista.MostrarElementos();
    }
}
```

6)

```
using System;

class ListaLinearSimples<T>
{
    private class No
    {
        public T Dado;
public No Proximo;

        public No(T dado)
        {
            Dado = dado;
            Proximo = null;
        }
    }
    private No primeiro;

    public void Inserir(T dado)
    {
        No novoNo = new No(dado);
novoNo.Proximo = primeiro;          primeiro
= novoNo;
    }
    public void Remover(T dado)
    {
        if (primeiro == null)
        {
            return;
        }
        if (primeiro.Dado.Equals(dado))
        {
            primeiro = primeiro.Proximo;
return;
        }

        No atual = primeiro;
        while (atual.Proximo != null && !atual.Proximo.Dado.Equals(dado))
        {
            atual = atual.Proximo;
        }
        if (atual.Proximo != null)
        {
            atual.Proximo = atual.Proximo.Proximo;
        }
    }
    public void MostrarElementos()
    {
        Console.WriteLine("Elementos na lista:");
No atual = primeiro;          while (atual !=
null)
```

```
        {
            Console.WriteLine(atual.Dado);
atual = atual.Proximo;
        }
    }
}  class
Program {
    static void Main(string[] args)
    {
        ListaLinearSimples<int> minhaLista = new
ListaLinearSimples<int>();

        minhaLista.Inserir(5);
minhaLista.Inserir(10);
minhaLista.Inserir(15);

        minhaLista.MostrarElementos();

        minhaLista.Remover(10);

        minhaLista.MostrarElementos();
    }
}

7)  using
System;
class Fila<T>
{
    private class No
    {         public T
Dado;         public No
Proximo;

        public No(T dado)
{
            Dado = dado;
            Proximo = null;
        }
    }
    private No inicio;
private No fim;

    public void Enfileirar(T dado)
    {
        No novoNo = new No(dado);

        if (inicio == null)
        {             inicio
= novoNo;             fim =
novoNo;
        }
else         {
            fim.Proximo = novoNo;
fim = novoNo;
        }
    }
```

```
    public T Desenfileirar()
    {
        if (inicio == null)
        {
            throw new InvalidOperationException("A fila está vazia.");
}

        T dado = inicio.Dado;
inicio = inicio.Proximo;

        if (inicio == null)
        {
            fim = null;
        }
        return dado;
    }
    public void MostrarElementos()
    {
        Console.WriteLine("Elementos na fila:");
No atual = inicio;          while (atual != null)
        {
            Console.WriteLine(atual.Dado);
atual = atual.Proximo;
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        Fila<int> minhaFila = new Fila<int>();

        minhaFila.Enfileirar(5);
minhaFila.Enfileirar(10);
minhaFila.Enfileirar(15);

        minhaFila.MostrarElementos();

        int desenfileirado = minhaFila.Desenfileirar();
        Console.WriteLine($"Elemento desenfileirado: {desenfileirado}");

        minhaFila.MostrarElementos();
    }
}

8)
using System;

class Pilha<T>
{
    private class No
    {
        public T Dado;
public No Proximo;

        public No(T dado)
```

```
        {
            Dado = dado;
            Proximo = null;
        }
    }
    private No topo;

    public void Empilhar(T dado)
    {
        No novoNo = new No(dado);
novoNo.Proximo = topo;          topo
= novoNo;
    }
    public T Desempilhar()
    {
        if (topo == null)
        {
            throw new InvalidOperationException("A pilha está vazia.");
}

        T dado = topo.Dado;
topo = topo.Proximo;
return dado;
    }
    public void MostrarElementos()
    {
        Console.WriteLine("Elementos na pilha:");
No atual = topo;          while (atual != null)
        {
            Console.WriteLine(atual.Dado);
atual = atual.Proximo;
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        Pilha<int> minhaPilha = new Pilha<int>();

        minhaPilha.Empilhar(5);
minhaPilha.Empilhar(10);
minhaPilha.Empilhar(15);

        minhaPilha.MostrarElementos();

        int desempilhado = minhaPilha.Desempilhar();
        Console.WriteLine($"Elemento desempilhado: {desempilhado}");

        minhaPilha.MostrarElementos();
    }
}
```

9) using System; class ListaOrdenada

```
<T> where T :
IComparable<T
>
{
    private class No
    {
        public T Dado;
public No Proximo;

        public No(T dado)
        {
            Dado = dado;
            Proximo = null;
        }
    }
    private No primeiro;

    public void InserirOrdenado(T dado)
    {
        No novoNo = new No(dado);

        if (primeiro == null || dado.CompareTo(primeiro.Dado) <= 0)
        {
            novoNo.Proximo = primeiro;
primeiro = novoNo;              return;
        }

        No atual = primeiro;
while (atual.Proximo != null &&
dado.CompareTo(atual.Proximo.Dado) > 0)
        {
            atual = atual.Proximo;
        }
        novoNo.Proximo = atual.Proximo;
atual.Proximo = novoNo;
    }
    public void MostrarElementos()
    {
        Console.WriteLine("Elementos na lista ordenada:");
No atual = primeiro;         while (atual != null)
        {
            Console.WriteLine(atual.Dado);
atual = atual.Proximo;
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        ListaOrdenada<int> minhaLista = new ListaOrdenada<int>();

        minhaLista.InserirOrdenado(10);
minhaLista.InserirOrdenado(5);         minhaLista.InserirOrdenado(15);
minhaLista.MostrarElementos();
    }
```

```
}

10)  using
System;  class
Deque<T> {
    private class No     {
public T Dado;
public No Proximo;

        public No(T dado)
{
            Dado = dado;
            Proximo = null;
        }
    }      private No
inicio;     private No
fim;

    public void AdicionarInicio(T dado)
    {
        No novoNo = new No(dado);
if (inicio == null)         {
inicio = novoNo;             fim
= novoNo;         }         else
{
            novoNo.Proximo = inicio;
inicio = novoNo;
        }
    }
    public void AdicionarFim(T dado)
    {
        No novoNo = new No(dado);
if (fim == null)         {
inicio = novoNo;             fim
= novoNo;         }         else
{              fim.Proximo =
novoNo;              fim = novoNo;
        }
    }
    public T RemoverInicio()
    {
        if (inicio == null)
        {
            throw new InvalidOperationException("O deque está vazio.");
}

        T dado = inicio.Dado;
inicio = inicio.Proximo;

        if (inicio == null)
        {
            fim = null;
        }
        return dado;
    }
    public T RemoverFim()
    {
```

```
        if (fim == null)
        {
            throw new InvalidOperationException("O deque está vazio.");
}

        T dado = fim.Dado;

        if (inicio == fim)
        {
            inicio = null;
fim = null;
        }
else
        {
            No atual = inicio;
while (atual.Proximo != fim)
            {
                atual = atual.Proximo;
            }
            atual.Proximo = null;
fim = atual;
        }
        return dado;
    }
    public void MostrarElementos()
    {
        Console.WriteLine("Elementos no deque:");
No atual = inicio;          while (atual != null)
        {
            Console.WriteLine(atual.Dado);
atual = atual.Proximo;
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        Deque<int> meuDeque = new Deque<int>();
meuDeque.AdicionarInicio(5);          meuDeque.AdicionarFim(10);
meuDeque.AdicionarFim(15);

        meuDeque.MostrarElementos();

        int removidoInicio = meuDeque.RemoverInicio();
Console.WriteLine($"Elemento removido do início:
{removidoInicio}");

        int removidoFim = meuDeque.RemoverFim();
        Console.WriteLine($"Elemento removido do fim: {removidoFim}");

        meuDeque.MostrarElementos();
    }
}
```