Vetores

Vetores/Arrays unidimensionais

- Permitem o tratamento de um conjunto de dados com as mesmas características;

- São uma maneira de armazenar vários dados (dezenas, centenas, milhares, milhões...) em um **mesmo nome de variável** através do uso de **índices numéricos.**

Declaração

- A declaração de uma variável do tipo vetor em C# segue o formato:

- `tipo[] nome = new tipo[tamanho];`

- Onde:
  - **`tipo`**: é o tipo de dados dos elementos que serão armazenados;
  - `nome`: é o nome da variável (nome do vetor);
  - **`[tamanho]`**: é um valor numérico que indica quantos elementos estarão armazenados no vetor, do tipo de dado especificado

Declaração

- Exemplo de declaração:

```
//vetor de 100 elementos do tipo inteiro
int[] meuVetor = new int[100];

//Vetor de 50 elementos do tipo double
double[] meuOutrovetor = new double[50];

//vetor de 25 elementos do tipo char
char[] maisUmVetor = new char[25];
```

Manipulação de um vetor

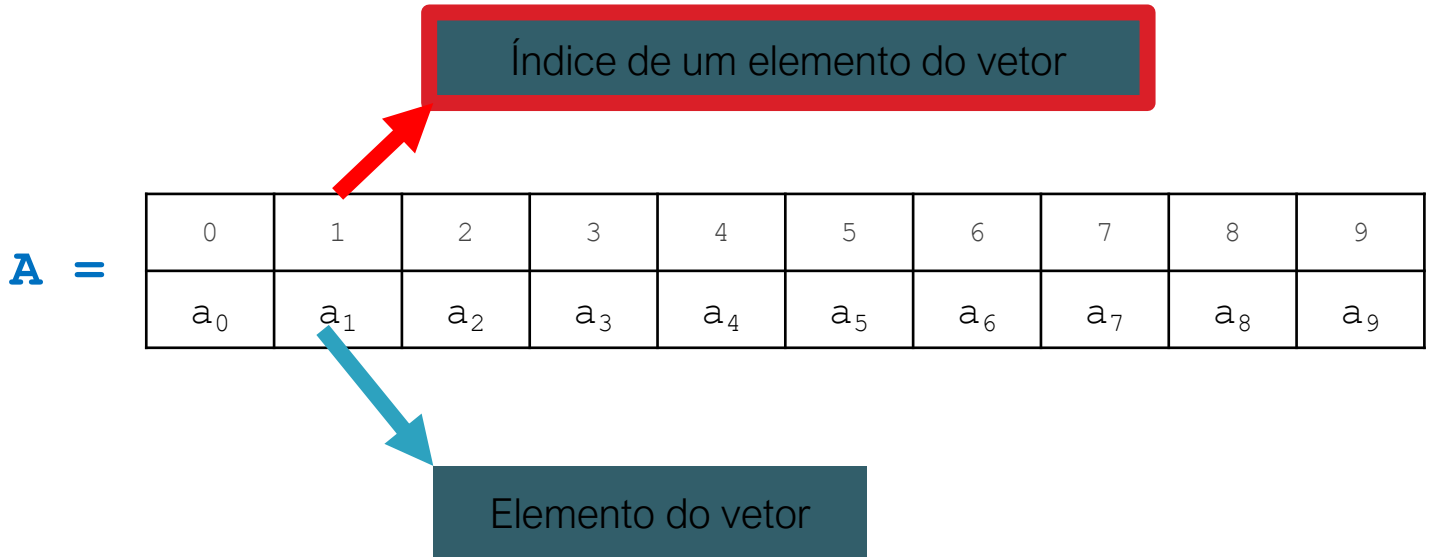
- Os elementos são **referenciados** por índices;

- O **primeiro elemento** do vetor possui sempre o índice **ZERO**;

- O **último elemento** do vetor tem índice igual ao TAMANHO do vetor MENOS 1 **`[tamanho-1];`**

Manipulação de um vetor

- Por exemplo, declaramos o seguinte vetor:
  - int[] A = new int[10];
- Na memória, ele seria armazenado mais ou menos assim:



Manipulação de um vetor

- Para acessar um elemento do vetor, deve-se usar o nome do vetor e o valor do índice do elemento entre colchetes [ ]

- Por exemplo, na figura abaixo, o 4º elemento do vetor é referenciado por `A[3]`, cujo valor é `20`.

```
int[] A = new int[10];
```

A =

0	1	2	3	4	5	6	7	8	9
5	10	15	20	25	30	35	40	45	50

Exemplo 1 – leitura de valores

```
static void Main(string[] args)
{
    int i;
    int[] b = new int[5];
    Console.WriteLine("Digite o valor 0: ");
    b[0] = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite o valor 1: ");
    b[1] = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite o valor 2: ");
    b[2] = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite o valor 3: ");
    b[3] = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite o valor 4: ");
    b[4] = int.Parse(Console.ReadLine());

    for (i = 0; i < 5; i++)
    {
        Console.WriteLine("Valor índice "+i+" é igual a "+b[i]);
    }
}
```

Exemplo 2 – leitura de valores v2

```
static void Main(string[] args)
{
    int i;
    int[] b = new int[5];
    Console.WriteLine("------ - Leitura dos valores:-------");
    for (i = 0; i < 5; i++)
    {
        Console.WriteLine("Digite o valor " + i + ":");
        b[i] = int.Parse(Console.ReadLine());
    }


    for (i = 0; i < 5; i++)
    {
        Console.WriteLine("Valor índice " + i + " é igual a " + b[i]);
    }
}
```

Exemplo 3 – porque usar vetores?

```
static void Main(string[] args)
{
    double[] notas = new double[50];
    double media;
    int i;

    Console.WriteLine("-------Leitura dos valores:-------");
    for (i = 0; i < 50; i++)
    {
        Console.WriteLine("Digite a nota do aluno "+i+":");
        notas[i] = double.Parse(Console.ReadLine());
    }

    media = 0;
    Console.WriteLine("-------Calculo da media sendo feito:-------");
    for (i = 0; i < 50; i++)
    {
        media += notas[i];
    }
    media = media / 50;

    Console.WriteLine("-------Mostra a media:-------");
    Console.WriteLine("A media de notas é igual "+media);

}
```

Exemplo 3v2 – aprimorado

```
static void Main(string[] args)
{
    double[] notas = new double[50];
    double media = 0;
    int i;

    Console.WriteLine("-------Leitura dos valores:-------");
    for (i = 0; i < 50; i++)
    {
        Console.WriteLine("Digite a nota do aluno " + i + ":");
        notas[i] = double.Parse(Console.ReadLine());
        media += notas[i];
    }
    media = media / 50;

    Console.WriteLine("-------Mostra a media:-------");
    Console.WriteLine("A media de notas é igual " + media);

}
```

Exemplo 4 – maior valor do vetor

```
static void Main(string[] args)
{
    int[] vetor = new int[10];
    int i, maior = -1000000;
    for (i = 0; i < 10; i++)
    {
        Console.WriteLine("Digite o valor "+i);
        vetor[i] = int.Parse(Console.ReadLine());
        if (vetor[i] > maior)
        {
            maior = vetor[i];
        }
    }
    Console.WriteLine("O maior valor do vetor é = "+maior);

}
```

Exemplo 5 – vetor A e vetor B

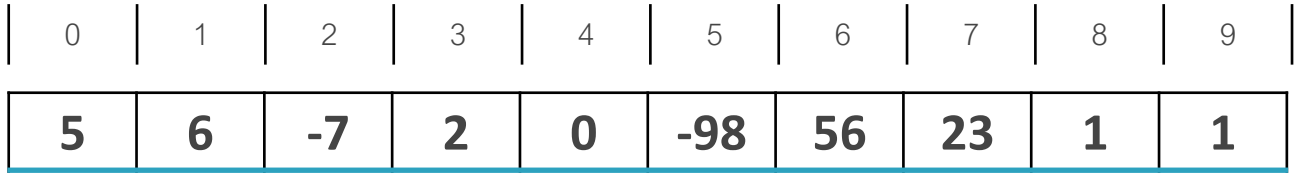
```
static void Main(string[] args)
{
    int[] vetorA = new int[15], vetorB = new int[15];
    int i;
    Console.WriteLine("-------Leitura dos elementos de A-------");
    for (i = 0; i < 15; i++)
    {
        Console.WriteLine("Digite o valor " + i);
        vetorA[i] = int.Parse(Console.ReadLine());
        if (vetorA[i] % 2 == 0)
        {
            vetorB[i] = vetorA[i] * 5;
        }
        else
        {
            vetorB[i] = vetorA[i] + 5;
        }

    }
    for (i = 0; i < 15; i++)
    {
        Console.WriteLine("VetorB["+i+"] ="+vetorB[i]);
    }
}
```

Matrizes

Matrizes/Vetores Bidimensionais

- Um vetor possui índices para uma determinada posição:

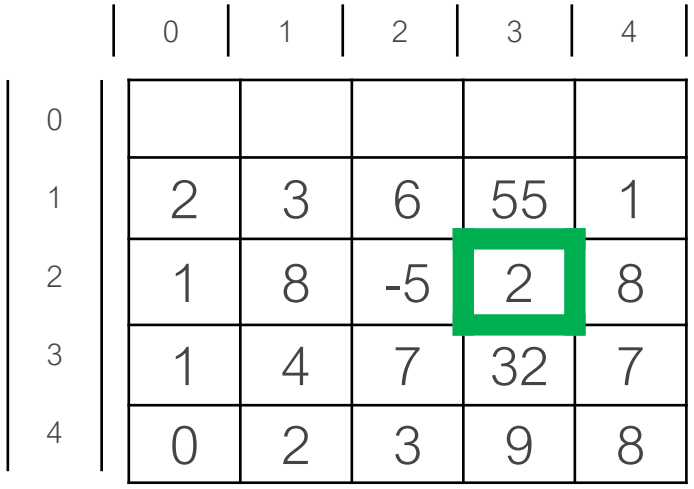
  - `int[] vet = new int[10];`



Matrizes/Vetores Bidimensionais

- Uma matriz possui índices para linhas e colunas:

  - `int[,] mat = new int[5,5];`



Ou seja, o elemento marcado a esquerda é referenciado por: `mat[2,3]`

Propriedades:

- Os índices sempre iniciam em zero.

  - Por exemplo, para uma matriz com **`L`** linhas e **`C`** colunas:
    - os índices das linhas variam de **`[0, L-1]`**
    - os índices das colunas variam de **`[0, C-1]`**

  - Logo, uma matriz com 5 linhas e 10 colunas:
    - os índices das linhas variam de **`[0, 4]`**
    - os índices das colunas variam de **`[0, 9]`**

Declaração:

- A **declaração** de uma variável do tipo Matriz na linguagem C# segue o formato:

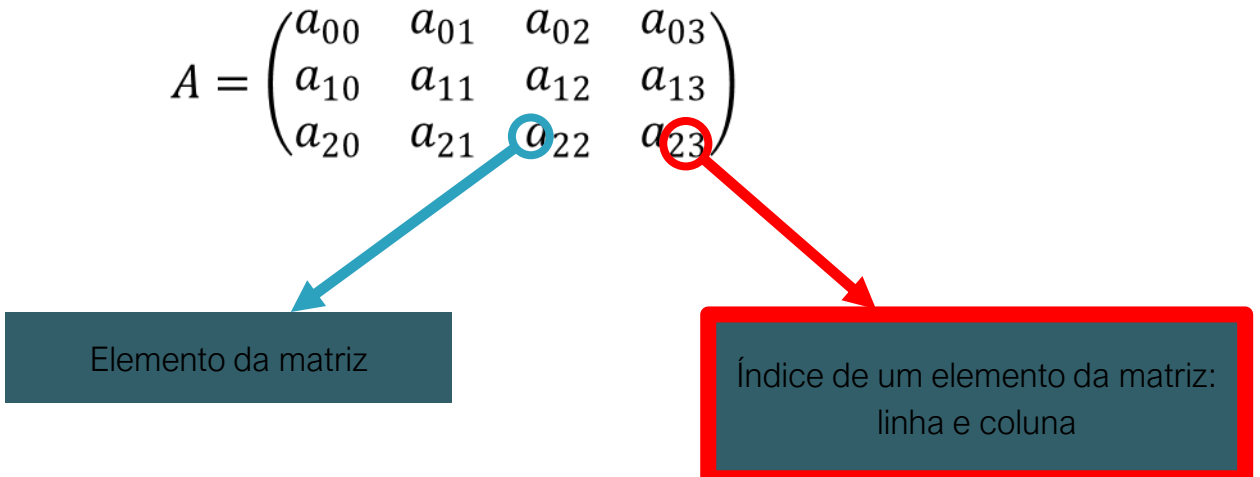
  - `tipo[,] nome = new tipo[qtde_linhas,qtde_colunas];`

- onde:
  - **`tipo`** é o tipo de dados dos elementos que serão armazenados;
  - **`nome`** é o nome da variável (nome da matriz)
  - **`qtde_linhas`** é um valor numérico que indica quantas linhas estarão armazenadas na matriz, do tipo de dado especificado.
  - **`qtde_colunas`** é um valor numérico que indica quantas linhas estarão armazenadas na matriz, do tipo de dado especificado.

Elementos:

- Os elementos são **referenciados** por **índices** de linha e coluna.

- Na ilustração a seguir, há a representação de uma matriz de elementos inteiros, com 3 linhas e 4 colunas, declarada como `int[,] A = new int[3,4]`.

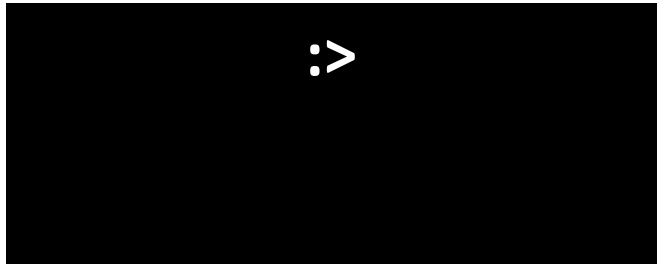


Manipulação:

- Para acessar um elemento da matriz, deve-se usar o **nome da matriz** e os **valor dos índices da linha e da coluna** do elemento, entre colchetes;

```
double[,] A = new double[2,2];
        A[0,0] = 9.0;
        A[0,1] = 8.0;
        A[1,0] = 7.0;
        A[1,1] = 7.0;
```

$$A = \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix}$$

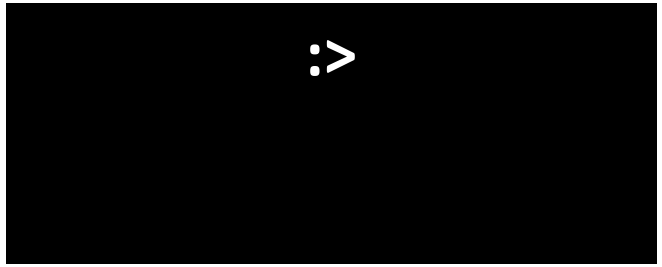


Manipulação:

- Para acessar um elemento da matriz, deve-se usar o **nome da matriz** e os **valor dos índices da linha e da coluna** do elemento, entre colchetes;

```
double[,] A = new double[2,2];
        A[0,0] = 9.0;
        A[0,1] = 8.0;
        A[1,0] = 7.0;
        A[1,1] = 7.0;
```

$$A = \begin{pmatrix} 9.0 & 8.0 \\ 7.0 & 7.0 \end{pmatrix}$$

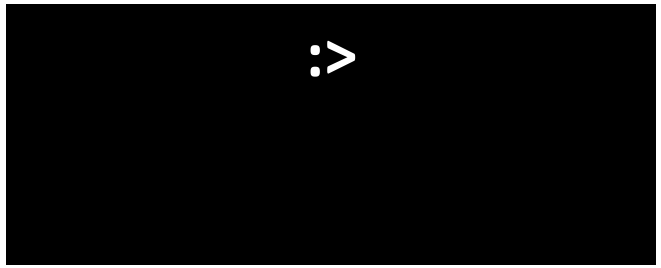


Manipulação:

- Para acessar um elemento da matriz, deve-se usar o **nome da matriz** e os **valor dos índices da linha e da coluna** do elemento, entre colchetes;

```
double[,] A = new double[2,2];
          A[0,0] = 9.0;
          A[0,1] = 8.0;
          A[1,0] = 7.0;
          A[1,1] = 7.0;
quant=A[0,0]+A[0,1]+A[1,0]+A[1
                ,1];
   Console.WriteLine(quant);
```

$$A = \begin{pmatrix} 9.0 & 8.0 \\ 7.0 & 7.0 \end{pmatrix}$$



Manipulação:

- Para acessar um elemento da matriz, deve-se usar o **nome da matriz** e os **valor dos índices da linha e da coluna** do elemento, entre colchetes;

```
double[,] A = new double[2,2];
          A[0,0] = 9.0;
          A[0,1] = 8.0;
          A[1,0] = 7.0;
          A[1,1] = 7.0;
quant=A[0,0]+A[0,1]+A[1,0]+A[1
                  ,1];
   Console.WriteLine(quant);
```

$$A = \begin{pmatrix} 9.0 & 8.0 \\ 7.0 & 7.0 \end{pmatrix}$$

```
:>31.0000
```

Manipulação:

- Para acessar um elemento da matriz, deve-se usar o **nome da matriz** e os **valor dos índices da linha e da coluna** do elemento, entre colchetes;

```
double[,] A = new double[2,2];
        A[0,0] = 9.0;
        A[0,1] = 8.0;
        A[1,0] = 7.0;
        A[1,1] = 7.0;
quant=A[0,0]+A[0,1]+A[1,0]+A[1,1];
    Console.WriteLine(quant);
        media = quant/4;
    Console.WriteLine(media);
```

$$A = \begin{pmatrix} 9.0 & 8.0 \\ 7.0 & 7.0 \end{pmatrix}$$

```
:>31.0000
:>7.75000
```

Manipulação através de um laço:

- Em um vetor, utilizamos um **`for`** para realizar a leitura de elementos;
- Em matrizes, utilizaremos um **`for`** dentro de um outro **`for`**;

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \end{pmatrix}$$

– Assim, para listar todos os elementos da matriz na tela, pode-se usar os seguinte código:

```
int i, j;
int[,] A = new int[3, 4];
for (i = 0; i < 3; i++)
{ //laço que percorre as linhas da matriz
    for (j = 0; j < 4; j++)
    { //laço que percorre as colunas da matriz
        Console.WriteLine("A[" + i + "," + j + "] = " + A[i, j]);
    }
}
```

Manipulação:

- para fazer a leitura dos elementos de uma matriz, utiliza-se a função **`Console.ReadLine()`**:

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \end{pmatrix}$$

  - para fazer a leitura dos elementos de uma matriz:

```
int i, j;
int[,] A = new int[3, 4];
for (i = 0; i < 3; i++)
{ //laço que percorre as linhas da matriz
    for (j = 0; j < 4; j++)
    { //laço que percorre as colunas da matriz
        Console.WriteLine("Digite o elemento A["+i +","+j +"]");
        A[i, j] = int.Parse(Console.ReadLine());
    }
}
```

Arquivos

Arquivos

- As operações de input e output na plataforma .NET são gerenciadas pela classe abstrata Stream que esta no namespace System.IO.

- Se você conhece os fundamentos da programação orientada a objetos deve saber que uma classe abstrata não pode ser instanciada diretamente.

Arquivo

- Escrita
  - A classe StreamWriter implementa um TextWriter para a escrita de caractereres em um fluxo (stream) usando uma determinada codificação.

Arquivo

```
static void Main(string[] args)
{

    try //tento os comandos abaixo:
    {
        StreamWriter sw = new StreamWriter("C:\\arquivo\\teste.txt");  //Passa o caminho do arquivo e o nome do arquivo para o Construtor StreamWriter
        sw.WriteLine("Estou escrevendo um texto aqui..."); //Escreve uma linha de texto no StreamWriter
        sw.WriteLine("E posso ter uma segunda linha!"); //Escreve outra linha de texto no StreamWriter
        sw.Close(); //Fecha o arquivo
    }
    catch (Exception e) //caso ocorra algum erro, entro e apresento o erro (Exceção)
    {
        Console.WriteLine("Exceção: " + e.Message);
    }
    finally // garante que o código definido aqui sempre será executado mesmo com exceção (útil para limpar os recursos que foram alocados).
    {
        Console.WriteLine("Bloco finally");
    }

}
```

Arquivo

- Leitura
  - Para a Leitura, as operações utilizam a classe StreamReader, que é um tipo de TextReader e que StreamWriter é um tipo de TextWriter.
  - Assim, um StreamReader obtém os seus dados de um fluxo ou stream que pode se representado por dados que estão na memória, em um arquivo ou vindo de uma porta serial, vídeo ou capturado em um dispositivo.

Arquivo

```
static void Main(string[] args)
{
    String line;
    try
    {
        //Passa o caminho do arquivo e o nome do arquivo para o construtor StreamReader
        StreamReader sr = new StreamReader("C:\\arquivo\\teste.txt");
        line = sr.ReadLine(); //Lê a primeira linha do arquivo

        while (line != null)//Contiinua lendo enquando não encontra o final do arquivo
        {

            Console.WriteLine(line); //Apresenta a linha lida no Console
            line = sr.ReadLine(); //E efetua a leitura da próxima linha do arquivo
        }
        sr.Close(); //Fecha o arquivo
        Console.ReadLine();
    }
    catch (Exception e)
    {
        Console.WriteLine("Exceçãp: " + e.Message);
    }
    finally
    {
        Console.WriteLine("Executando o bloco finally");
    }
}
```

```
static void Main(string[] args)
{
    String line;
    try
    {
        //Passa o caminho do arquivo e o nome do arquivo para o construtor StreamReader
        StreamReader sr = new StreamReader("C:\\arquivo\\teste.txt");
        line = sr.ReadLine(); //Lê a primeira linha do arquivo

        while (line != null)//Contiinua lendo enquando não encontra o final do arquivo
        {

            Console.WriteLine(line); //Apresenta a linha lida no Console
            line = sr.ReadLine(); //E efetua a leitura da próxima linha do arquivo
        }
        sr.Close(); //Fecha o arquivo
        Console.ReadLine();
    }
    catch (Exception e)
    {
        Console.WriteLine("Exceçãp: " + e.Message);
    }
    finally
    {
        Console.WriteLine("Executando o bloco finally");
    }
}
```

Arquivo

```
class Arquivo
{
    string nome, mensagem;
    StreamWriter sw;
    StreamReader sr;
    public Arquivo(string nome)
    {
        this.nome = nome;
    }

    public void criaAbreArquivo()
    {
        sw = new StreamWriter("C:\\Arquivo\\" + nome + ".txt", true, Encoding.UTF8);

    }

    public void lerArquivo()
    {
        string linha;
        sr = new StreamReader("C:\\Arquivo\\" + nome + ".txt");
        linha = sr.ReadLine();
        while (linha != null)
        {
            Console.WriteLine(linha);
            linha = sr.ReadLine();
        }
        sr.Close();
    }

    public void gravaMensagem(string mensagem)
    {
        sw.WriteLine(mensagem);
    }

    public void fechaArquivo()
    {
        sw.Close();
    }

}
```

Arquivo

```
class Arquivo
{
    string nome, mensagem;
    StreamWriter sw;
    StreamReader sr;
    public Arquivo(string nome)
    {
        this.nome = nome;
    }

    public void criaAbreArquivo()
    {
        sw = new StreamWriter("C:\\Arquivo\\" + nome + ".txt", true, Encoding.UTF8);

    }
```

Arquivo

```
public void lerArquivo()
{
    string linha;
    sr = new StreamReader("C:\\Arquivo\\" + nome + ".txt");
    linha = sr.ReadLine();
    while (linha != null)
    {
        Console.WriteLine(linha);
        linha = sr.ReadLine();
    }
    sr.Close();
}
```

Arquivo

```
public void gravaMensagem(string mensagem)
{
    sw.WriteLine(mensagem);
}

public void fechaArquivo()
{
    sw.Close();
}
```

Arquivo

```
static void Main(string[] args)
{
    string nome, mensagem;
    Console.WriteLine("Digite o nome do arquivo: ");
    nome = Console.ReadLine();
    Arquivo a = new Arquivo(nome);
    int op;
    while (true)
    {
        Console.WriteLine("Digite uma operação:\n1 - para escrever no arquivo\n2 - para apresentar o texto do arquivo\n3 - para finalizar a execução ");
        op = int.Parse(Console.ReadLine());
        if (op == 1)
        {
            a.criaAbreArquivo();
            Console.WriteLine("Digite a mensagem para ser armazenada: \nPara sair, digite SAIR");
            while (true)
            {
                mensagem = Console.ReadLine();
                if (mensagem.Equals("SAIR"))
                {
                    a.fechaArquivo();
                    break;
                }
                else
                {
                    a.gravaMensagem(mensagem);
                }
            }
        }
        else if (op == 2)
        {
            a.lerArquivo();
        }
    }
}
```

```
static void Main(string[] args)
{
    string nome, mensagem;
    Console.WriteLine("Digite o nome do arquivo: ");
    nome = Console.ReadLine();
    Arquivo a = new Arquivo(nome);
    int op;
    while (true)
    {
        Console.WriteLine("Digite uma operação:\n1 - para escrever no arquivo\n2 - para apresentar o texto do arquivo\n3 - para finalizar a execução ");
        op = int.Parse(Console.ReadLine());
        if (op == 1)
        {
            a.criaAbreArquivo();
            Console.WriteLine("Digite a mensagem para ser armazenada: \nPara sair, digite SAIR");
            while (true)
            {
                mensagem = Console.ReadLine();
                if (mensagem.Equals("SAIR"))
                {
                    a.fechaArquivo();
                    break;
                }
                else
                {
                    a.gravaMensagem(mensagem);
                }
            }
        }
        else if (op == 2)
        {
            a.lerArquivo();
        }
    }
}
```

Atividades

- Vetores
- Matrizes
- Arquivos