**Comandos C# no Terminal**

**Csc NomeArquivos.cs = Compila o arquivo, e gera o executável;**

**cls = Apaga a Tela;**

**cd NOME DA PASTA = Entra na pasta;**

**cd .. = Sai da pasta atual;**

**cd ../.. = Volta 2 pasta;**

**mkdir NomePasta = Cria pasta;**

**rmdir NomeArquivo = Deleta o Arquivo (Método 1);**

**Rename NomeAquivo NovoNome = Renomeia o Arquivo;**

**NUL> NomeArquivo = Cria o Arquivo;**

**del NomeArquivo = Deleta o Arquivo (Método 2);**

**csc NomePrograma = Compila;**

**NomePrograma = Executa;**

**Comandos C#**

**Console.Write() =** Escreve na tela;

**Console.WriteLine() =** Escreve na tela com quebra de linha;

**Console.Read() =** Captura do terminal o que foi digitado e os retorna **(Aula08)**;

**Console.ReadLine() =** Captura do terminal o que foi digitado e os retorna  **(Aula08)**;

**Console.Clear() =** Limpa a tela **(Aula16)**;

**:c =** Valor Monetário na impressão;

**:p =** Formato de Porcentagem na impressão;

**.GetLength() =** Tamanho do Conteudo

**.GetValue(POSIÇÃO) =** Captura o elemento na posição

***Aula 08 – Ler Valores do Teclado***

**Console.Read() =** Captura do terminal o que foi digitado e os retorna;

**Console.ReadLine() =** Captura do terminal o que foi digitado e os retorna;

**Int.Parse() =** Converte o conteúdo para inteiro (Forma 1);

**Convert.ToInt32() =** Converte o conteúdo para inteiro (Forma 2);

**Aula 09 – Operações Bitwise**

**<< =** Dobra o valor;

**>> =** Diminui pela metade;

**Aula 11 – Conversões Typecast**

**Typecast: Converter um valor em outro;**

***Aula 16 – Comando GOTO***

**Console.Clear() =** Limpa a tela;

***Aula 22 - Foreach***

**FOREACH() = Pega um elemento de um Array, armazena em uma variavel e executa o comando**;

***Aula 23 – Metodos de Manipulação de Arrays***

***Random = Biblioteca que permite gerar números aleatórios;***

**Random.Next(NumMin, NumMax\*) =** Comando que permite sortear números, o () pode estar vazio, assim ele sorteia qualquer número\*;

**- Array.BinarySearch(Array, Valor) =** Retorna *(INT)* a posição do elemento no Array, caso o elemento no esta no Array, se retorna ***-1\****;

**- Array.Copy (Ar\_origem, Ar\_destino, qtde\_elementos) =** (*VOID)* Copia o conteúdo de um Array pra outro;

**NomeArray.CopyTo(Ar\_destino, a\_partir\_desta\_pos) =** *(VOID*) Copia o conteúdo do Array Citado, para outro Array, apartir de tal posição;

**NomeArray.GetLongLength(dimensão) =** Retorna (*LONG)* a quantidade de elementos do Array/Matriz;

**NomeArray.GetLowerBound(dimensão) =** Retorna *(INT)* o MENOR INDICE do Array/Matriz;

**NomeArray.GetUpperBound(dimensão) =** Retorna *(INT)* o MAIOR INDICE do Array/Matriz;

**NomeArray.GetValue(long índice) =** Retorna *(OBJECT)* o valor da posição a partir de um índice;

**Array.IndexOf(array, valor) =** Retorna *(INT)* a posição do primeiro elemento encontrado;

**Array.LastIndexOf(array, valor) =** Retorna(INT) a posição do último elemento encontrado;

**NomeArray.Reverse(array) =** Retorna(VOID) o valor INVERTIDO do Array;

**NomeArray.SetVaLue(object valor, long pos) =** (VOID) Permite inserir um valor a partir de uma posição;

***Aula 24 - Métodos***

***Métodos:*** *São um bloco de comandos/instruções que podem ser chamados a qualquer momento dentro do programa.*

***Aula 25 – Passagem de Valor e Por Referência***

***Passagem por valor:*** *Cria uma nova posição na memória, diferente da variável;*

***Por Referência:*** *Não cria uma nova posição na memória, utiliza a própria posição da variável;*

***Ref VAR: Semelhante ao Ponteiro em C++, captura exatamente o endereço da memória, no caso o valor da variável;***

***Aula 26 – Argumento OUT***

***- Argumento de saída, é como se fosse um Return***

**%(mode) =**Retorna o resto de uma divisão;

***Aula 27 – Argumento Params***

***- Permite a entrada de zero ou mais argumentos para uma função/metodo;***

***Aula 28 - Classes e Objetos***

***Classes: Tipo de dados composto por membros, onde os membros são as propriedades e métodos(funções);***

***Objetos: Instância(filhos) de uma classe;***

***Aula 29 – Construtores e Destrutores***

***Construtores: - Métodos que iram iniciar as propriedades/variáveis dos objetos instanciados;***

***- Devem ter o mesmo nome de seu Método;***

***- É chamado automaticamente quando instaciado um objeto dessa classe;***

***Destrutores: Métodos que destruirão os Objetos após eles não serem mais utilizados;***

***Aula 31 – Classes Static***

***Classes Static: - Não permitem a instanciação de objetos e o uso de Construtores, todos os seus membros devem ser statics;***

***- Quando se cria propriedades Static, todos os elementos da classe, apontaram pra esta propriedade;***

***Aula 32 – This***

***This: É uma referência que faz a chamada para o Método;***

***Aula 34 – Herança:***

***Herança: É quando um uma Classe Filha(Derivada), Herda membros de uma de uma Classe Pai, chamada de Classe Base;***

***Aula 36 – Protected***

***Private: Restringe o acesso ao membro, somente para a classe usando métodos dentro da própria classe***

***Protected: Permite o acesso pelas classes derivadas, mas somente pelas classes;***

***Aula 37 – Herança, Ordem de Execução dos Construtores***

***- A ordem sempre será da Base para seus derivados;***

***Aula 38 – Métodos Virtuais***

***Métodos Virtuais: São Métodos que tem o mesmo NOME da sua Classe Base, com Classes Diferentes;***

***Aula 39 – Classes e Métodos Abstratos***

***Classe Abstrata: Classe de Referência/Base para outras classes estanciadas, não permitindo a instanciação de novos objetos, só servem para ser herdadas;***

***- Caso se crie métodos não abstratos, devem ser implementados;***

***- Caso se crie métodos abstratos, não podem ser implementados;***

***Aula 40 – Classe Sealed***

***Classe Sealed : Classe Sealed não podem ser herdadas;***

***Aula 42 – Indexadores de Classes***

***Indexador: Membro de uma classe, que permite aos Objetos dessa classe, seram indexados como Arrays;***

***Aula 43 – Interfaces***

***Interfaces: Só complementam métodos, ou o protótipo dos métodos;***

***Aula 44 – Struct***

***Struct: - Estrutura que permite armazenar tipos de dados diferentes, e não se pode herdar;***

***- Struct é tratada como tipo Value, já a Classe é tratada como tipo Reference;***

***Aula 48 – Recursividade***

***Recursividade: Função que chama ela mesma;***

***Aula 50 – Delegates***

***Delegates: É um objeto que faz referência a um Método, acessa o endereço de memória do método;***

***Aula 54 – Namespaces***

***Namespace: Controla o escopo, organiza os elementos;***

***Aula 55 – Dictionary***

***Dictionary: Dupla de valores, formadas por key-value, precisa de importar a biblioteca “ using System.Collection.Generic ”;***

***NomeDictionary.Add() = Adiciona elementos ao Dictionary;***

***NomeDictionary.Count = Diz o tamanho do Dictionary;***

***NomeDictionary.Clear() = Limpa todo o Dictionary;***

***NomeDictionary.ContainKey(CHAVE) = Verifica se tem a CHAVE no Dictionary, retornando TRUE / FALSE;***

***NomeDictionary.ContainValue(VALOR) = Verifica se tem o VALOR no Dictionary, retornando TRUE / FALSE;***

***NomeDictionary.Remove(CHAVE) = Remove a CHAVE do Dictionary;***

***Aula 56 – LinkedList / Lista Dupllamente Encadeada***

***LinkedList: Cada elemento é um nó da lista, que tem um link pro elemento posterior e anterior. Precisa importar a biblioteca “ using System.Collection.Generic ”;***

***NomeLinkedList.AddFirst(“VALOR”) = Insere o elemento no inicio;***

***NomeLinkedList.AddLast(“VALOR”) = Insere o elemento no final;***

***NomeLinkedList.FindLast(“VALOR”) = Procura o último valor;***

***NomeLinkedList.AddAfter(LinkedListNode, “VALOR”) = Insere o elemento após o elemento;***

***NomeLinkedList.AddBefore(“VALOR”) = Insere o elemento antes do elemento;***

***NomeLinkedList.Find(“VALOR”) = Verifica se o valor existe, retorna TRUE /NULL;***

***NomeLinkedList.RemoveFirst (“VALOR”) = Remove o primeiro elemento;***

***NomeLinkedList.RemoveLast(“VALOR”) =Remove o último elemento;***

***Aula57 – List – Parte 1***

***List: Seria um “Array Tunado” . Precisa importar a biblioteca “ using System.Collection.Generic ”;***

***NomeList(PARA RECEBER).AddRange(NomeList para ser copiado para outra list) = Copia uma List e insere em outra;***

***NomeList.Contain(“VALOR”) = Verifica se o elemento existe na List, retorna TRUE / FALSE;***

***NomeList”ParaCopia”.CopyTo(List“Receber os Elementos”, POSIÇÃO) = Copia os elementos de uma List para outra, a partir de uma posição;***

***Aula 58 – List – Parte 2***

***NomeList.Insert(POSIÇÃO, “VALOR”) = Insere o elemento na posição desejada;***

***NomeList.LastIndexOf (“VALOR”) = Retorna a posição do ultimo elemento pesquisado;***

***NomeList.RemoveAt() = Remove o elemento na posição indicada;***

***NomeList.Reverse() =Inverte a ordem da List;***

***NomeList.Sort() = Ordena a List;***

***NomeList.Count = Diz o tamanho da List;***

***Aula 59 – Queue / Fila***

***Queue: Primeiro elemento a entrar, é o primeiro a sair;***

***NomeQueue.Enqueue(“VALOR”) = Insere o elemento no final;***

***NomeQueue.Dequeue() =Retorna o valor do primeiro elemento, e o remove;***

***NomeQueue.Peek() =Retorna o valor do primeiro elemento, e NÃO o remove;***