C# Referências:

**Básico:**

|  |
| --- |
| **C#:** |
| Console.WriteLine() - Exibe valores na tela. |
| .Readline () - Espera o imput do usuário. |
| .Read() - Retorna o valor ASCII de um =caracter. |
| .ReadKey() - Para o programa aguardar sem finalizar. |
| ConvertToInte32(Console.ReadLine()). |
| .Equals() - Compara se valor e tipo são iguais. |
| .GetHashCode - Mostra o código de uma variável/objeto. |
| void Dobrar(ref int y).... calc.Dobrar(ref x) -> Para alterar também o valor original da variável passada no parâmetro. |
| .IsReadOnly – Verifica se é apenas leitura |
| .AsReadOnly – Converte em apenas leitura. |
| Atributo estático de classe é compartilhado por todos. |
| **Data Types:** |
| sbyte: -128/127(8bit/1byte) - byte 0/255(8bit) - short -32.768-32.767(16bits/2bytes) - ushort 0/65.535 |
| int - -2bi/2bi(32bits/4bytes) / long - -9bi/9bi(64bits/8bytes) - unint 0/4bi - ulong 0/18bi |
| float 4bytes - 6 a 9 dígitos (colocar F/f no final do valor) / double 8bytes - 15 a 17 dígitos / decimal 16bytes - 28 a 29 dígitos (colocar M/m no final do valor) |
| bool - 8 bit (true or false) / char - 16bits, um único caractere Unicode entre aspas simples (Ex: 'A', 'b') |
| string - Sequência de caracteres, 16bits pra cada. |
| **Math:** |
| Math.Max(x, y...) - Retorna o valor máximo no parâmetro (Math.Min() faz o contrário. |
| Math.Sqrt(x) - Retorna a raiz quadrada de um valor. |
| Math.Abs(x) - Retorna um valor negativo pro positivo. / |
| Math.Round(x) - Retorna um número fracionado pro seu inteiro mais próximo (arredonda). |
| Math.pi / Cos.(x) / Sin(x) / Tan(x) / Pow(x, y) = x elevado a y / Exp(x) = Exponencial |

**Data:**

|  |
| --- |
| DateTime data = new DateTime(1993, 02, 03); //Data personalizada. |
| DateTime data2 = DateTime.Now; //Data atual. |
| Data = dateAndTime.ToShortDateString()/dateAndTime.ToLongDateString() //Versão curta ou longa da hora. |
| Data = dateAndTime.ToShortTimeString()/dateAndTime.ToLongTimeString() //Versão curta ou longa de horas. |

**Array:**

|  |
| --- |
| Array.Reverse() // Inverte o array. |
| Array.Sort(); // Reordena o array. |
| [,] – Array BiDimensional / [,,] Array MultiDimensional. |

**ArrayList:**

|  |
| --- |
| Lista.add()// |
| Lista.Insert(index, value)// |
| Lista.AddRange(ArrayValue)//Insere um array no final do outro. |
| Lista.InsertRange(index, arrayValue) // |
| Lista.Remove(value);// |
| Lista.RemoveAt(index);// |
| Lista.RemoveRange(indexInicio, indexFim) |
| Lista.Contains(“Rafael”) //bool, verifica se contém o valor. |
| Lista.Clear() //Remove tudo. |

**Random:**

|  |  |  |
| --- | --- | --- |
| Random aleat = new Random() | //Instanciar classe Random |  |
| Console.WriteLine(aleat.Next()) | // Numero aleatorio padrão |  |
| Console.WriteLine(aleat.Next(20)) | // Numero aleatório com o máximo no parametro. |  |
| Console.WriteLine(aleat.Next(10, 20)) | // Aleatorio entre os numeros do parametro. |  |
| Console.WriteLine(aleat.NextDouble()) | //Aleatorio 0.0 <> 1.0 |  |
| Console.WriteLine(aleat.NextInt64()) | //Aleatorio 0 <> 9bi |  |

**Classe/Obj:**

|  |
| --- |
| Pessoa p1 = new Pessoa(); //Instanciou classe Pessoa. |
| Classes/Métodos não ”static” precisam ser instanciados. |
| Classe/Método “abstract” só podem ser instanciados pelas classes herdeiras. |
| Método “void” não gera “return”. |
| Argumento nomeado. Ex: (Destinatário(Nome: “Rafael”, Cel: xxx-xx)); |
| Parametro padrão. Ex: (Email(assunto = “Assunto padrão”) |

POO:

|  |
| --- |
| Public - Acessado de qualquer parte do programa. |
| Private - Acessado apenas dentro da própria classe. |
| Protected - Acessado pela classe e pela herança (classes filho). |
| Abstract - Não pode ser instanciado. Mas pode ser herdado. |
| Internal - Usado apenas no mesmo projeto (assembly, DLL). |
| Partial class - A mesma classe pode ser trabalhada em mais de um arquivo no projeto. |
| Sealded - Impede que a classe seja herdada. Ou um Override seja sobrescrito. |
| Override – Sobrescreve Virtual e Abstract. |

**Tratamento de erros:**

|  |
| --- |
| Exception() - Classe base de todas as exceções. |
| "throw new exception() - Chamada. |
| NotImplementedException() - Um metodo não está implementado. |
| ArgumentException() - |
| NullReferenceException() - Indica eferencia nula. |
| FormatException() - Formatação inválida. |
| IndexOutOfRangeException() - Indice fora do intervalo. |
| OverFlowException() - Excedeu capacidade de processamento. |
| FileNotFoundException() - Arquivo não localizado. |
| InvalidCastException() - Não foi possível fazer conversão de valores. |
| StackOverFlowException() - Capacidade da pilha excedida. |
| Ex manual: throw new Exception("Lançamento manual de excessão"). |
| .Message - Fornece detalhes. |
| .StackTrace - Rastreamento de pilha para determinar origem do erro. Inclui nome do arquivo e linha. |
| .Source - Define o nome do aplicativo ou objeto que causou o erro. |
| .HelpLink - Contém URL para arquivo de ajuda que fornece informações sobre a exceção. |
| .InnerException - Criar e preservar exceções durante o tratamento. |

**Generics:**

|  |
| --- |
| **Mais utilizados:** |
| Collection<T> |
| List < T > ---Versão genérica e tipada de ArrayList. |
| Dictionary<Tkey, Tvalue> --- Chave/Valor (Chave é única). |
| Queue<T> --- Fila |
| Stack<T> ---- Pilha. |
| HashSet<T> --- Lista com valores distintos, sem repetir. |

**Arquivos/Diretórios/Stream:**

|  |
| --- |
| ***Classes para tratamento de arquivos e diretórios:*** |
| File - FileInfo - Directory - DirectoryInfo - Path - FileStream - StreamReader - StreamWriter |
|  |
| **FILE:** |
| ***Métodos de manipulação::*** |
| Create - Delete - Copy - Move - (Nomes auto explicativos) |
| Exists - Verifica se o arquivo existe. |
|  |
| ***Método leitura e escrita:*** |
| ReadAllText - Abre, lê, retorna uma string e fecha. |
| ReadAllBytes - Lê arquivo binário e retorna array de bytes. |
| WriteAllText - Cria arquivo, grava uma string em arq de texto e fecha. |
| WriteAllBytes - Grava array de bytes em arq binario. |
| AppendAllText - Abre arquivo, anexa string e fecha (cria arquivo se não existir um). |
|  |
| ***Outros métodos:*** |
| ReadAllLines - Lê as linhas do arquivo |
| GetLastwriteTime -Data / Hora última modificação. |
| GetLastAccesTime - Data / Hora último acesso. |
| ***Criar arquivos – Caminho e Nome::*** |
| EX: string caminho = @"C:\MeusArquivos\arquivo.txt"; ("@" permite uso normal do "\"). |
| EX2: string caminho = "C:\\MeusArquivos\\arquivo.txt"; (Sem "@"). |
|  |
| FILESTREAM: |
| Formas de acesso: |
| FileStream fs = new FileStream(CaminhoArquivo, FileMode.Open); |
| FileStream fs1 = File.Open(CaminhoArquivo, FileMode.Open) |
| Entre outras... |
|  |
| Formas de fechamento pós acesso: |
| using (FileStream fs = New FileStream(CaminhoArquivo, FileMode.Open)  {  //Operações  }; |
| using FileStream fs = new FileStream(CaminhoArquivo, FileMode.Open); |
| Close() - (em cada bloco de codigo).\*/ |
| **Chamada comum:**  **fs = new FileStream(caminhoArquivo, FileMode.Open, FileAccess.Read); leitor = new StreamReader(fs);** |
| **Chamada simplificada:**  **leitor = File.OpenText(caminhoArquivo);** |

**LINQ:**

|  |
| --- |
| ------------------PRINCIPAIS---------------- |
| ***Elemento:*** |
| Fisrt() / FirstOrDefault(); //Retorna o primeiro elemento da coleção. |
| ElementAt() / ElementAtOrDefault(); //Retorna o elemento de acordo com o index. |
| Single() / SingleOrDefault(); //Retorna se tem apenas um certo elemento na coleção. Sem réplicas. |
| Last() / LastOrDefault(); //Retorna o último elemento da coleção. |
|  |
| ***Quantificação:*** |
| Contains(); //Retorna se tem o elemento na coleção. |
| Any() //Retorna se tem pelo menos um elemento na coleção(bool). |
| All(); //Retorna se todos os elementos satisfazem a condição(bool). |
|  |
| ***Agregação:*** |
| Count(); //No lugar do Lenght. |
| ***Particionamento:*** |
| Skip(); //Pula quantidade de elementos da coleção. |
| Take(); //Pega quantidade de elementos da coleção. |
| (Obs: Usar ambos em conjunto para paginação. Ou direto Take com range (x..y)). |
|  |
| ***Filtros:*** |
| Where(); //Condição para filtrar os elementos da coleção. (Ex .Where(x => x < 10 || x > 0)). |
| DICA: |
| Usar.ToList() quando manipular as listas. |

RESUME:

Array: Vetor (lista reta) ou Matriz(Lista com linhas e colunas) de um tipo.

ArrayList:

Mesmo que Array, porém comporta vários tipos.

Condicional/Loop :

Condições lógicas para resolução de operações (If, Else, While, Switch, etc...).

POO (Programação Orientada a Objetos):

Uso de Classes; Objetos; Métodos; Interfaces; Herança; agregação... etc

Para abstração e resolução de problemas.

Generics:

Tipos genéricos para versatilizar o uso de tipos num mesmo método/classe/etc.

Programação Assíncrona:

Altera o fluxo do bloco selecionado para uma outra Thread. Enquanto a principal mantém seu trabalho sem precisar ser interrompida.

LINQ:

Métodos de Consulta/Filtro com uso de Expressão Lambda.