Download .NET: <https://dotnet.microsoft.com/dowload>

Documentação do .NET: <https://docs.microft.com/en-us/dotnet/>

.NET foundation: <https://dotnetfoundation.org/>

Comando para criar um projeto local via execução console:

* dotnet new console –n “nome do projeto”

Para executar uma build de teste se todo o projeto está feito de maneira correta:

* dotnet build

Para executar o arquivo:

* dotnet run

O que é C#?

R: É uma linguagem que foi criada no final dos anos 90 teve sua versão 1.0 entre 2001 e 2002, e vem evoluindo agora na versão 5.0, mas no meu projeto estou usando 3.1. É uma linguagem Orientada a objeto e fortemente tipada, sintaxe similar ao C, C++ ou Java, Suporta conceitos e encapsulamento, herança e polimorfismo (OO).

Os programas C# são executados no .NET, atualmente o compilador é o Roslyn que inclui:

* CLR (Common Language Runtime)
* Conjunto unificado de bibliotecas de classes

Como funciona o C#?

R: O código fonte em C# é compilado em uma linguagem intermediária (IL), O código e os recursos de IL são armazenados no disco em um arquivo executável chamado assembly, geralmente com uma extensão:

* .exe
* .dll

Inclui uma extensa biblioteca com milhares de classes que fornecem uma ampla variedade de funcionalidades úteis, desde entrada e saída de arquivos, manipulação de cadeias de caracteres, análise XML, etc...

Também fornece outros serviços:

* Garbage Collector- Uma ferramenta que trabalha coletando lixo, verifica objetos que não estão sendo utilizados na memoria e aloca esses objetos para o lixo liberando memória em disco.
* Exceptin Handler- Serve para receber e controlar as exceções que ocorrem de erros dentro do código.
* Resources Manager- Gerenciador de Recursos

Estrutura do programa?

R: principais conceitos organizacionais em C# são:

* Programas- Programas consistem em um ou mais arquivos, os programas declaram tipos, que contêm membros e podem ser organizados em namespaces.
* Namespaces- Gerenciam o portador da classe principal e faz referência de caminho.
* Tipos- Classes e interfaces exemplos de tipos
* Membros-Campos, Métodos, Propriedades e eventos são exemplos de membros

Tipo de Valor- contêm diretamente seus dados, as variáveis têm sua própria cópia dos dados e não é possível que as operações afetem outra variável (exceto no caso das variáveis de parâmetros ref e out).

* Numéricos: sbyte, short, int, log, byte, ushort, uint, ulong
* Caracteres Unicode: char
* Boleano: bool

Enum, struct e tipos nullable

Tipos de classe: class, object, string

Tipos de arrays: int[], etc...

* Array – é uma estrutura de dados que contem um numero x de elementos, todos do mesmo tipo que são acessados através de índices computados.
* Classes- Um classe é uma estrutura de dados que combina estado (campos) e ações (métodos).
* Objetos são instâncias de uma classe- As classes suportam herança e polimorfismo, mecanismos pelos quais as classes derivadas podem estender e especializar as classes base.

**public class Ponto{**

**public int x, y;**

**public Ponto(int x, int y){**

**this.x;**

**this.y;**

**}**

**}**

**Instâncias de classes (objetos) são criadas usando o operador new, que aloca memória para uma nova instância, chama um construtor para inicializar a instância e retorna uma referência à instância:**

**Ponto p1=new Ponto(0,0);**

**Ponto p2= new Ponto(10,20);**

* **Os membros de uma classe podem ser estáticos ou membros da instância.**
* **Membros estáticos pertencem à classe e membros de instância pertencem ao objeto.**
* **Constantes, variáveis, métodos, propriedades, construtores, etc... São membros**
* **Herança- É uma declaração de classe que pode especificar uma classe base, herdando os membros public, internal e protected da classe base.**
* **Métodos- É um membro que determina uma ação que pode ser executada por um objeto ou classe. Podem agregar uma lista de parâmetros, que representam valores ou referências de variáveis passadas para o método, assim como podem ter um tipo de retorno, como também especificar o tipo do valor calculado e retornando pelo método.**

**Structs**

**São estruturas de dados que podem conter membros de dados e membros ação, são do tipo de valor e não requerem alocação heap. Uma variável de um tipo struct armazena diretamente os dados da estrutura.**

**Structs não aceitam herança determinada pelo desenvolvedor.**

**Interfaces**

**Funciona como um contrato que pode ser implementado através de classes e structs, normalmente pode conter métodos, propriedades eventos e indexadores. As interfaces podem empregar herança múltipla**

**Enums**

**É um tipo de valor diferente com um conjunto de valores nomeados, Quando for definir um tipo que pode ter um conjunto de valores discretos. Normalmente usam um dos tipos de valor integral como armazenamento subjacente. Fornecendo significado semântico aos valores discretos.**

**Cada tipo de enum possui um tipo integral correspondente chamado tipo subjacente do tipo enum. Um tipo de enumeração que não declara explicitamente um tipo subjacente tem um tipo subjacente int.**

**Orientação a objetos em .NET**

**O que são os paradigmas de programação- São modos diferentes de estruturar a execução de um código.**

**Cada paradigma surgiu por necessidades diferentes. Sendo assim que um paradigma pode oferecer técnicas apropriadas para uma aplicação específica. Entre elas estão à programação orientada a objetos (POO) e a Programação estruturada (PE).**

**Há três tipos de estruturas básicas para navegar pelo código: sequência (Comando executados de cima para baixo, linha a linha de forma sequencial), seleção (sequência que será apenas executada se uma condição for satisfeita) e repetição (Sequencias de devem ser executadas repetidamente até uma condição for satisfeita), uma das características principais de programação estrutural é a interpretação linha por linha.**

**O POO uma de suas características é que um procedimento de objeto pode acessar, e geralmente modificar, os campos de dados do objeto com o qual eles estão associados e cada um é pode receber e processar dados. O POO contém os campos que são os atributos e o procedimentos que são os métodos.**

**Existem quatro pilares para considerar uma linguagem á POO sendo elas: Abstração, Encapsulamento, Herança e Polimorfismo.**

**Tudo em .Net é POO, mesmo os tipos de dados mais simples são considerados objetos, já estes também contém métodos e propriedades.**

**O que são Classes?**

**Pode ser considerado um molde para um objeto, contendo dentro de si as principais informações para sua criação. Define os atributos e métodos para a sua criação. Define os atributos e métodos comuns que serão compartilhados por um objeto.**

**O que são Objetos?**

**Tudo aquilo em geral que possui um atributo, comportamento e um estado.**

**Uma observação sobre a classe é que ela passa a ser considerada uma classe, caso ela tenha a criação de um objeto.**

**O que seria Visibilidade?**

**A visibilidade é importante para a proteção do código e suas funcionalidades, define quem pode alterar cada dado dos trechos de código em três principais níveis:**

**Pública (representado pelo símbolo “+”)**

**Privada (representado pelo símbolo “-”)**

**Protegida (representado pelo símbolo “#”)**

**public – Sem limitações de acesso**

**protected internal – Acesso limitado à própria classe. Às classes derivadas e ao próprio Assembly.**

**protected – Acesso limitado à própria classe, às classes derivadas**

**internal – Acesso limitado ao próprio assembly**

**private – Acesso limitado à própria classe.**

**Visibilidade está relacionada diretamente ao encapsulamento de atributos e métodos que impede o chamado vazamento de escopo, onde um atributo ou método é visível por alguém que não deveria vê-lo, como outro objeto ou classe.**

**Tipos por valor**

**A atribuição de uma variável a outra copia o conteúdo, criando efetivamente outra copia da variável. Essas variáveis não podem conter o valor null.**

**Os tipos de inteiros (Integers) sempre terão o mesmo significado, independente da implementação.**

**Sendo eles: int, char (usa 16 bits por caractere), long int, double (suporta 15 dígitos após a vírgula), float (suporta 7 dígitos após a vírgula),**

**Tipos por referencia**

**Um tipo por referencia armazena uma referência a seus dados. Os tipos de referência seus dados. Os tipos de referência podem incluir: Duas variáveis podem conter a referência a um mesmo objeto, Operações feitas em um objeto podem afetam o outro objeto.**

**Tipos de valor por referência são as matrizes, que exige que todos seus elementos sejam de tipos de valor, assim como as Strings, classes e arrays.**

**Tipos ponteiro**

**É um tipo de dado cujo valor se refere diretamente a um outro valor alocado em outra área da memória, através de seu endereço.**

**Métodos o que são?**

**Determinam o comportamento dos objetos de uma classe e são capazes de controlar o estado da instancia. São funções que realizam tarefas. Eles podem ou não retornar valores e podem ou não receber parâmetros.**

**Tipos de Métodos**

**Métodos assessores- Os famosos Getters que solicitam acesso a informações de um determinado objeto sem dar acesso diretamente a ele, colocando ali uma barreira de proteção de dados.**

**Métodos modificadores- Melhor dizendo Setters enviam solicitações de alteração de uma determinada informação de um objeto sem que se altere diretamente o mesmo.**

**Métodos Construct- construct é inicializador ou da forma para a classe. É nele que indicamos os valores dos campos de uma classe. Esses valores normalmente podem ser internos (no código) ou externos (quando passados por parâmetros).**

**As regras para definir um struct são: O construtor deve ter o mesmo nome da classe, Não ter tipo de retorno, Ser executado somente uma vez no momento da criação do objeto, Não pode ser chamado diretamente, Deve ser “public”.**

**Propriedades- Consistem em um par de métodos “get” e “set” que respectivamente servem para recuperar e atribuir o valor a um campo. Geralmente, para cada método existe uma variável dentro da classe que armazena o valor da propriedade.**

**Eventos- São mensagens que a classes dispara em determinadas situações. Para o evento ocorrer, se faz necessário que um método seja escrito para ser executado quando ocorrer o evento. A classe apenas se informa do método em tempo de execução.**

**Para que o evento funcione são necessárias estruturas chamadas Delegate, uma variável que guarda o endereço de uma função. Assim, quando o evento acontecer, essa variável fará a chamada da função associada a ela.**

**SQL Server**

**Relacionais: SQL Server, Oracle, MySQL**

**Não Relacionais: MongoDB, Neo4j, Firebase**

**Atomicidade: Garantir que todos os dados estão corretos, para inseri-los dentro de uma tabela do banco de dados.**

**Consistência dos dados: Existe uma preocupação com a inserção e verificação dos dados constante, mas esse detalhe apenas se aplica nos bancos de dados relacionais.**

**Server=localhost\SQLEXPRESS;Database=master;Trusted\_Connection=True;**