**Aula 00: Introdução**

Aqui vamos estudar o começo de algoritmos para entrarmos no mundo da programação, já aviso que esse curso é inteiramente feito e produzido com o **Portugol Studio**, e nas próximas aula veremos como instalar essa ferramenta em nossa máquina. Porém, teremos um bônus a quem chegar no final, que será um complemento desse curso com **Java**, e também o processo para instalar a ferramenta em nossa máquina.

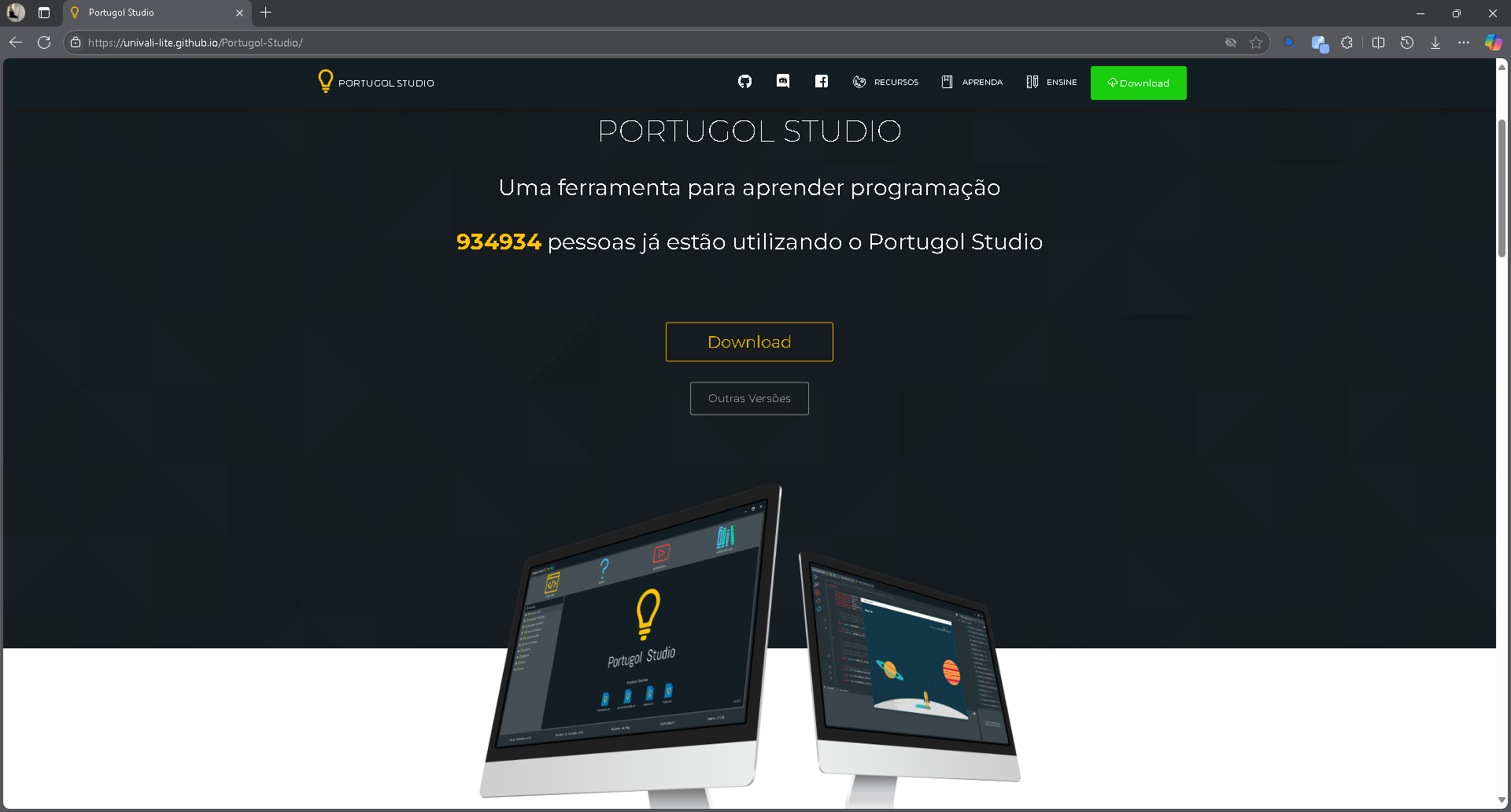
Esse curso é completo com aulas teóricas e práticas em torno do assunto de algoritmos, podendo ter exercícios e desafios que serão inteiramente realizados pelo aluno e não terá demonstração de resolução. Cada novo exercício e ou desafio, deverá ser feito do zero sem o reaproveitamento do código anterior. Essa prática, nos leva a fixar melhor o aprendizado a as nuances das linguagens utilizadas.

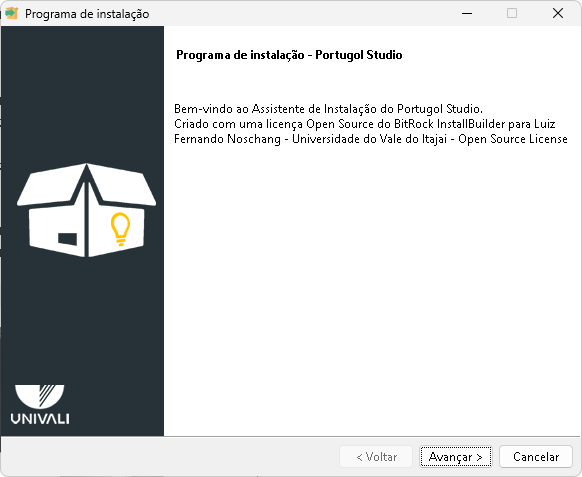
**Aula 01: Primeiro Algoritmo**

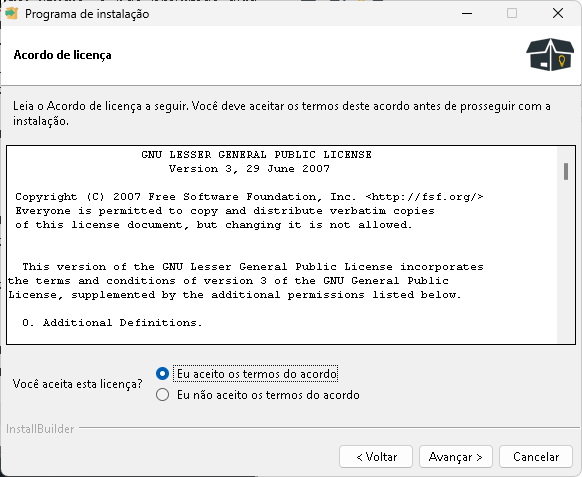
Antes de começarmos, vamos instalar e baixar a ferramenta que iremos trabalhar nesse curso, sem ela fica praticamente inviável ter algum tipo de evolução no curso, ou até mesmo programação.

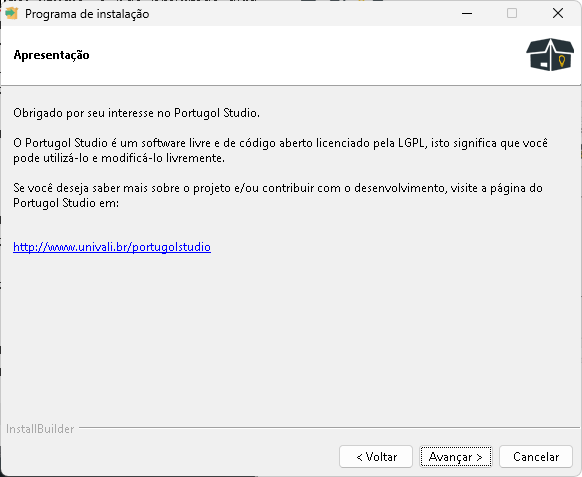
E como dito anteriormente, vamos aprender primeiramente usando o Portugal Studio, que nada mais é que um editor de código (IDE), como tantos outros no mercado, porém com sua linguagem “nativa” em português. A ideia de usar esse editor é deixar mais fácil e prático todo o entendimento de algoritmos, e logo após isso partir par Java que tem os mesmos, quase os mesmos comandos, porém em inglês.

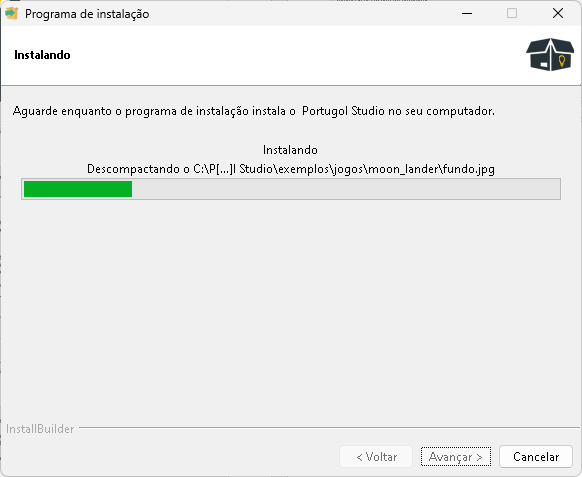
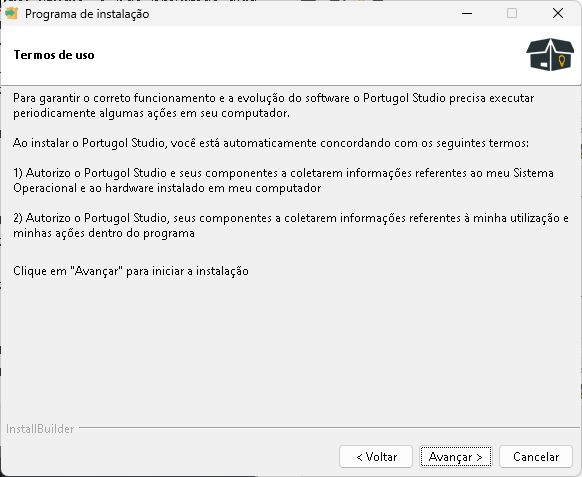
Primeiramente vamos acessar o site para baixar o Portugol Studio, que é <https://univali-lite.github.io/Portugol-Studio/>, e vamos clicar em “download” e vamos salvar o arquivo, e logo em seguida vamos abrir esse arquivo e prosseguir para a instalação, seguindo todos os passos das imagens a seguir:

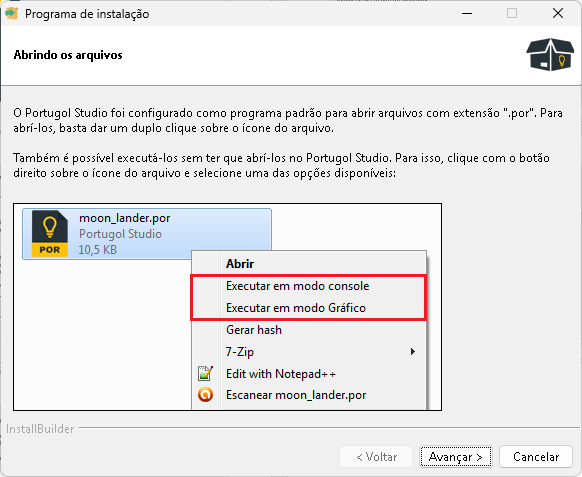












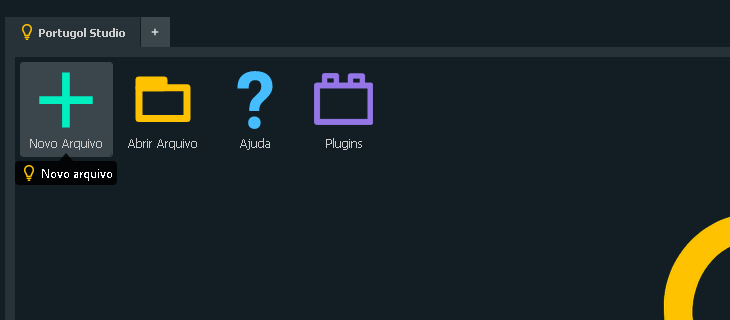
Logo após essa tela, já teremos terminado o processo de instalação, e podemos ver em nossa área de trabalho o ícone do nosso editor de código



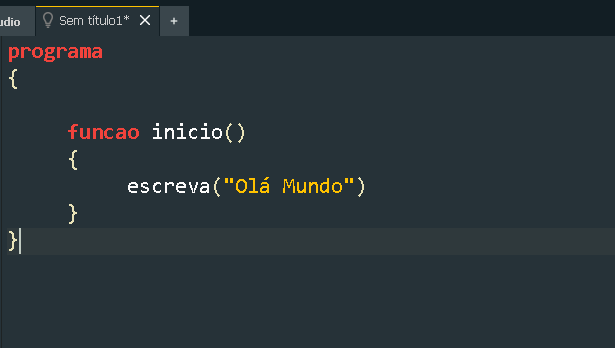
Sendo assim, já podemos iniciar e realizar o nosso primeiro exercício, que temos aqui.



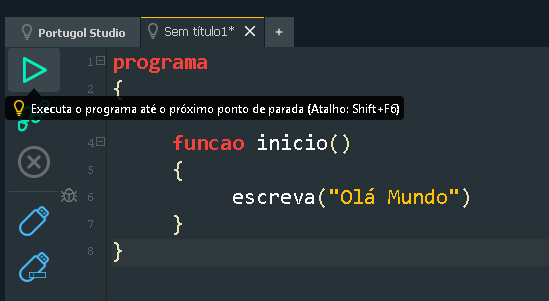
O primeiro passo, com o Portugol aberto, vamos selecionar a opção “novo arquivo”.



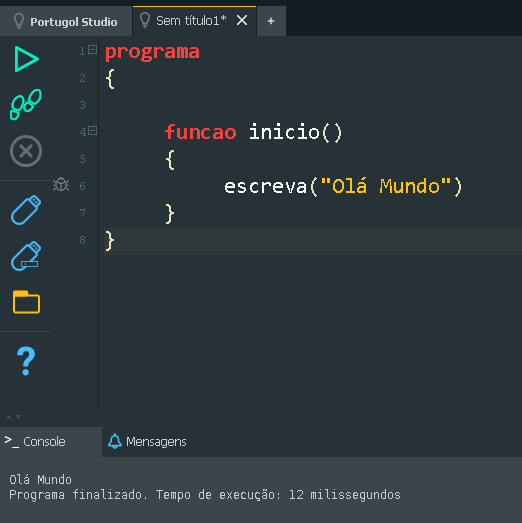
Vamos perceber que nosso novo arquivo já vem com um primeiro código “pronto”, e assim podemos observar a estrutura e que a sua linguagem simples está em português.



E como podemos executar esse código? Bom, no próprio editor temos uma opção a esquerda que podemos fazer isso, parecia com o botão “play” de algum player de música.



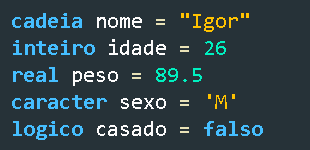
E como qualquer outro editor de código, podemos ver o resultado do nosso código sendo exibido no “console”.



**Aula 02: Tipos Primitivos**

Tipos primitivos, nada mais são que os tipos de dados básicos que nossos programas podem receber e trabalhar. No caso do Portugal eles são:

* **cadeia**: todo o tipo de dado entre aspas duplas “”, que contenha mais de um caractere, será do tipo **cadeia** de caracteres. No Portugol, damos esse nome para esse tipo primitivo.
* **inteiro**: aqui, todo valor numérico inteiro, que não contenha pontos ou virgulas serão denominados.
* **real**: todo valor numérico “quebrado”, com pontos se separação, será denominado um valor **real**.
* **caracter**: todo caractere único dentro de aspas simples ‘’ será denominado caráter.
* **logico**: podemos chamar de resultado ou valor lógico, podendo ser verdadeiro ou falso. Não existe outro tipo de resultado ou valor aceito para o tipo primitivo logico.

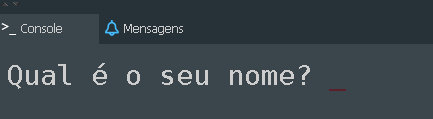


**Aula 03: Entrada de Dados**

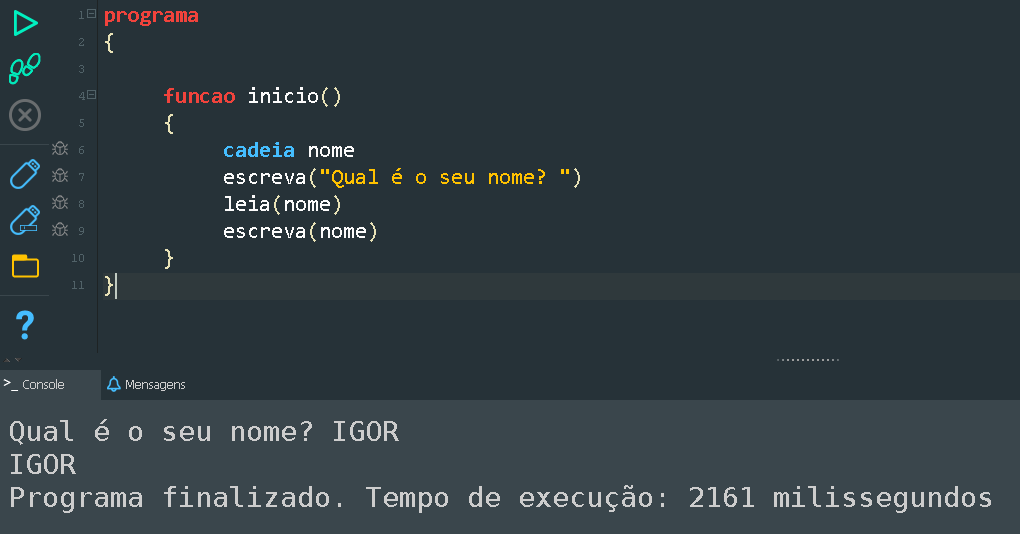
Nessa aula vamos aprender como fazer para que os dados sejam inseridos pelo usuário, e não mais inseridos diretamente no código, fazendo assim que nosso programa seja mais interativo.

No Portugol usamos o comando **LEIA** para inserir dados no nosso programa, e é bem simples. Vamos primeiramente retirar o valor da nossa cadeira de caractere e deixar apenas o nome da variável. Logo em seguida vamos chamar o comando leia, mas ainda assim precisamos dizer para esse comando qual variável que ele vai *imputar* valor, sendo assim entre parênteses vamos colocar o nome da nossa variável, **nome**.

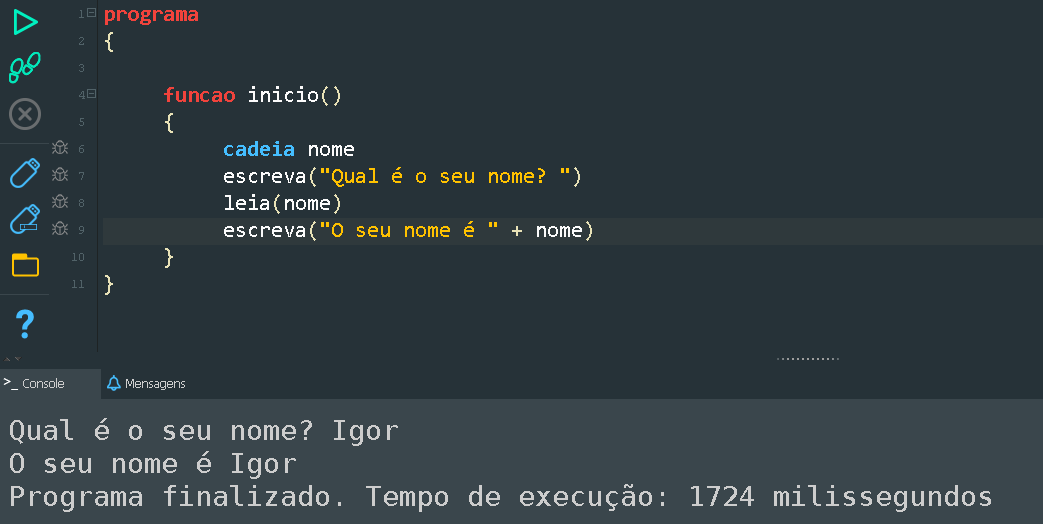
E como sabemos que esse valor foi atribuído a nossa variável nome? Vamos fazer com que o programa escreva o valor na tela, com o nosso comando **escreva**, e entre parênteses colocamos o nome da variável e assim podemos executar e ver o resultado.



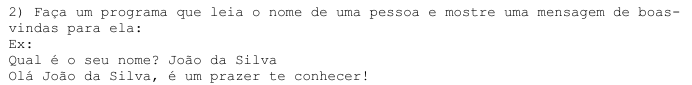
Podemos notar que temos na tela do console, o nosso texto de pergunta, juntamente com um hífen piscando, e exatamente ali que iremos digitar o nosso “nome” e logo em seguida podemos pressionar o enter para continuar o nosso programa. E o resultado será o seguinte:

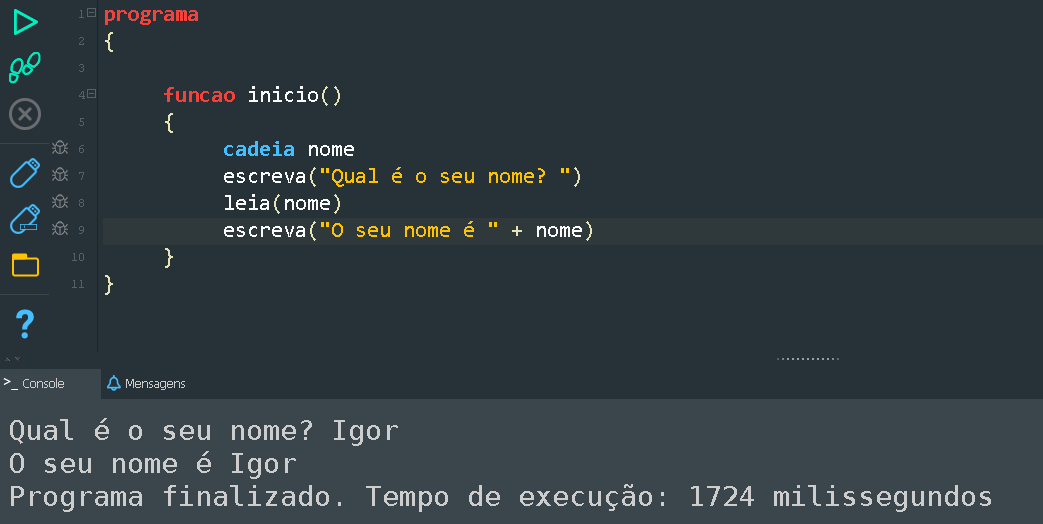


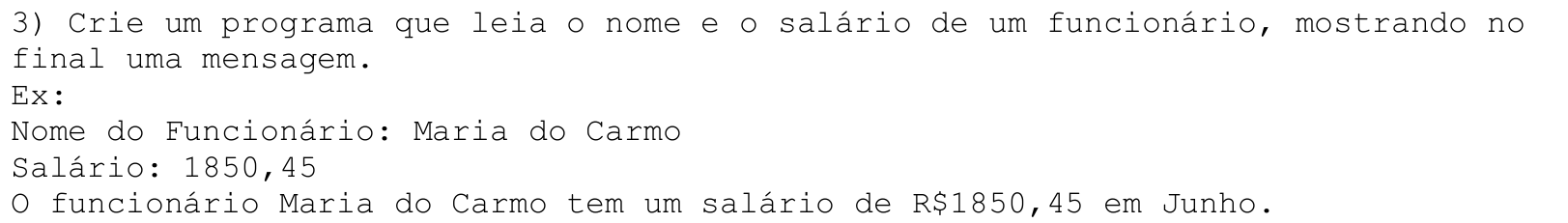
Certo, mas o resultado está muito simples, e para complementar essa mensagem podemos escrever um texto dentro de aspas duplas e com ajuda do sinal de + podemos realizar uma *interpolação*, que nada mais é que a junção de um texto com uma variável. Segue o exemplo:

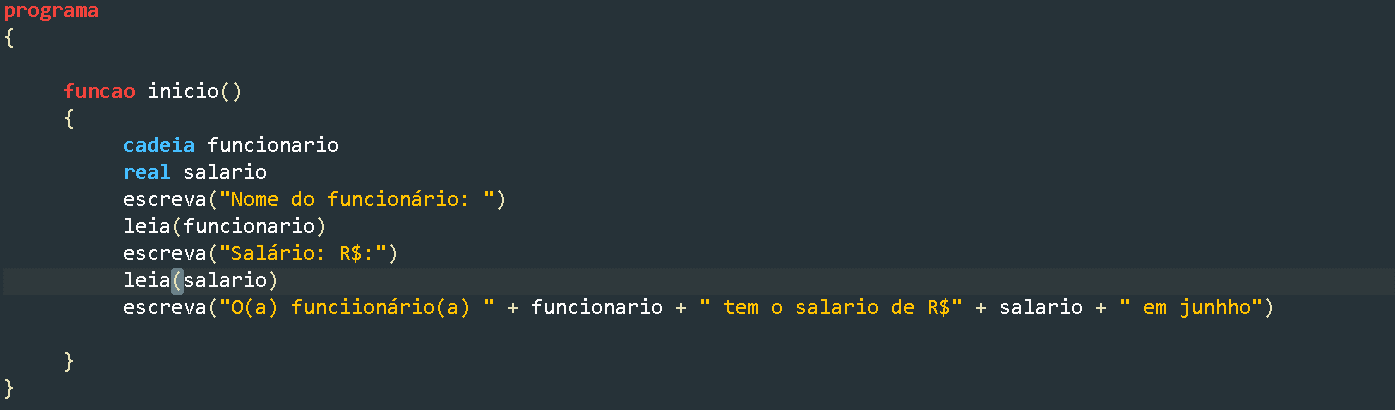


Sendo assim, podemos dar continuidade e praticar junto com o nosso **exercício 2**, e se o resultado for o mesmo que o esperado pelo exercício, podemos também realizar o **exercício 3**.









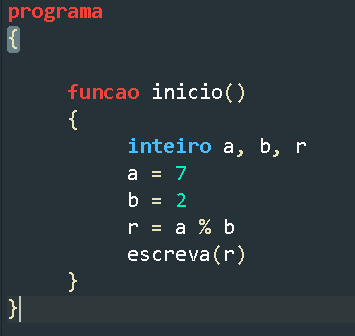
**Aula 04: Operadores Aritméticos**

Nessa aula temos que entender quais são os operadores aritméticos, e como usá-los em nossos programas. São eles:

* + soma
* - subtração
* \* multiplicação
* / divisão
* % resto

Aqui conhecemos quase todos os operadores, porém na programação temos o resto, que será o resto de uma divisão, ou seja, vamos dividir 7 por 2. O primeiro passo é declarar o 3, e nesse caso, a subtração entre os valores resta 1 e para continuarmos essa conta, precisamos adicionar uma virgula, e é ai que o **resto** entra em cena, o resto seria tudo que sobra da operação antes de solucionarmos a divisão, sendo assim, a divisão restou o valor 1, e esse é o valor do **resto**.

Temos que levar em consideração outro fator importante que é a **divisão**, pois se não declararmos o tipo correto do valor dela, uma divisão que resulta em um valor *quebrado* pode terminar com um valor **inteiro**. Então sempre declare o resultado de uma divisão como **real**, e para não ocorrer erros no processo, declare não só o resultado, mas todos os valores como **real**.



Note que no Portugal podemos declarar o tipo da variável, e como ela ainda não irá receber valor, podemos declarar várias variáveis ao mesmo tempo, apenas separando com uma vírgula. Após isso vamos chamá-las para assim atribuir valor.

Entendido como funciona o processo para declarar valores e como são os operadores aritméticos, faça do exercício 4 até o 11, e somente após a resolução deles avançaremos a próxima aula.

**Aula 05: Cálculo de Porcentagem**

Nessa aula não teremos muitas novidades, apenas vamos entender como funciona a lógica para o cálculo de porcentagem, podemos realizar da seguinte maneira, vamos imaginar que um produto custe R$100 reais, e vamos dar a ele 5% de desconto, então podemos realizar a operação das seguintes maneiras:

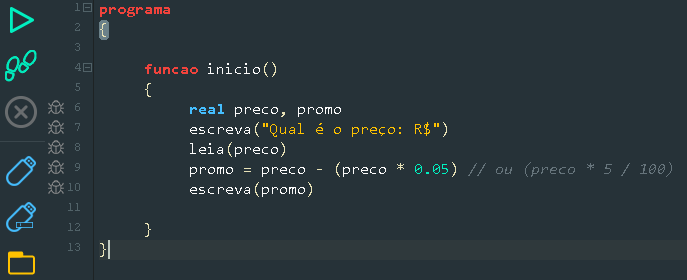
* 100 – (100 \*0.05) = 95.00

Aqui temos o preço menos o valor resultante da multiplicação para descobrir a porcentagem de 5% de 100, vamos só entender que o 0.05 corresponde aos 5%, se fossemos buscas 10% seria 0.10, se fossemos buscar 28$ seria 0.28, e assim por diante. Podemos também realizar da seguinte maneira:

* 100 – (100 \* 5 /100)

E da mesma forma aqui temos também uma multiplicação para descobrir a porcentagem, porém temos uma divisão de 5 dividido por 100, e aqui vamos descobrir a porcentagem de 5% do valor.

Para entendermos melhor vamos executar o exercício 12 da nossa lista de exercícios.



Em seguida podemos dar continuidade a nossa folha de exercícios, notando que o exercício 16 é um **desafio**, então ele é naturalmente mais complexo que os vistos até aqui.

**Aula 06: Condições Simples e Compostas**

Uma condição simples é caso uma determinada condição for atendida ela irá executar o bloco do código, se não for atendida nada irá acontecer. No caso de uma condição composta, vamos ter o argumento se a condição for verdadeira ou for falsa, assim o programa terá ações diferentes para qualquer resultado apresentado.



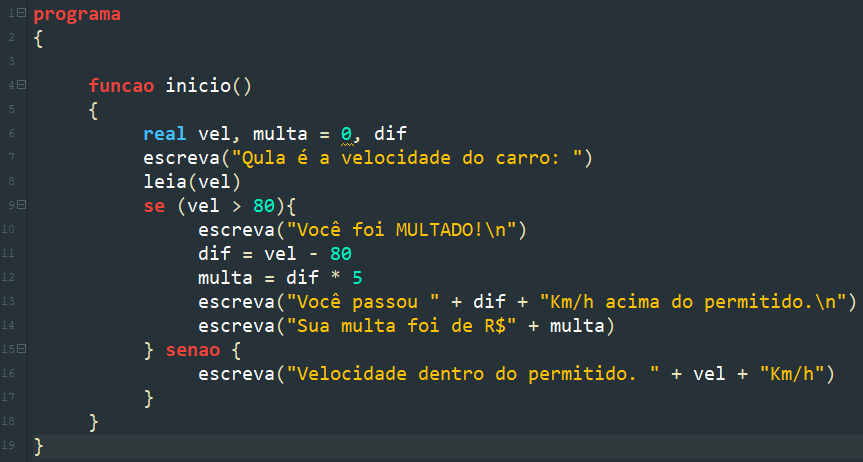
Um exemplo disso seria realizando um simples cálculo de “idade”, aonde **SE** (**condição simples**) for **maior que** ( **>** ) teremos uma mensagem aparecendo, **SENAO** (**condição composta**) vamos ter outra mensagem exibida na tela.

Para finalizarmos a explicação, temos que apresentar também os operadores relacionais, pois eles que realizam a lógica de nossas condições, e são eles:

* **>** maior que
* **<** menor que
* **>=** maior ou igual
* **<=** menor ou igual
* **==** igual (algumas linguagens utilizam **===** para definir o valor *idêntico*)
* **!=** diferente

Uma última observação para finalizar, é que ao declararmos um tipo de variável podemos declarar várias ao mesmo tempo, apenas com o nome dela e isso já a declara sem valor, ou com valor 0, e podemos declarar um valor dessa variável apenas separando elas por uma vírgula.

Seguindo tudo que apresentamos até aqui, vamos resolver o **exercício 17** e logo sem seguida podem fazer até o **exercício 25** que é o desafio.



**Aula 07: Operadores Lógicos e Relacionais**

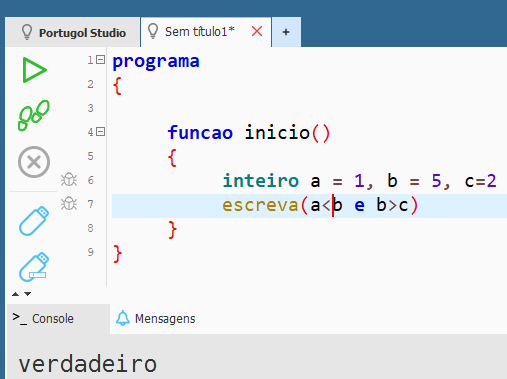
Operadores relacionais já explicamos acima como eles funcionam, mas agora vamos trazer os operadores Lógicos, e demonstrar em poucos exemplos como que eles funcionam.

Operadores Lógicos são aqueles que tem apenas dois resultados esperados, Verdadeiro ou Falso (true ou false) e assim podem ajudar em algumas validações em nosso código. Os operadores lógicos, independente da linguagem tem o mesmo conceito e são os seguintes:

* E: Operador lógico aonde os valores apresentados têm que conter o mesmo resultado para resultar em VERDADEIRO, se um deles for FALSO, o resultado será falso. Se os dois forem FALSO, também resultará como FALSO. Como nos exemplos abaixo:



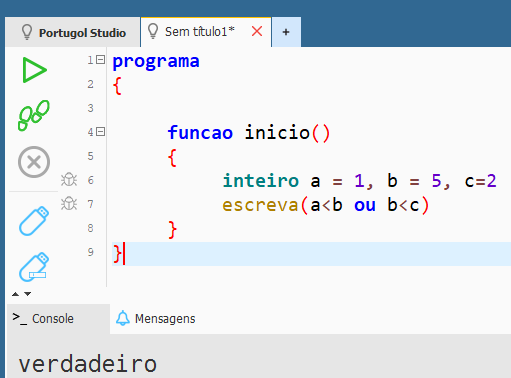
O resultado acima se dá como FALSO, porque um dos valores não foi dado como VERDADEIRO.



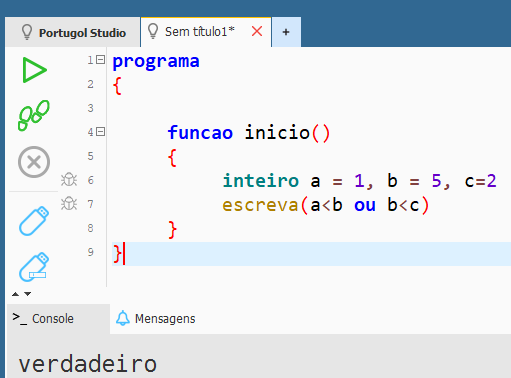
Resultado verdadeiro, onde ambos os valores são VERDADEIROS.

resultado falso, pois os dois valores resultam como FALSO.

* OU: operador lógico, onde podemos ter um resultado ou outro para VERDAEIRO. Ou seja, se um dos valores apresentados resultarem em VERDADEIRO, mesmo que o outro seja FALSO o resultado será verdadeiro. Agora, se os dois forem FALSO, o resultado será FALSO. Como nos exemplos abaixo:



No código acima, podemos perceber que um dos resultados é FALSO e o outro é VERDADEIRO, e mesmo assim o resultado é VERDADEIRO.



Agora podemos perceber, que como explicado o resultado para FALSO é apenas quando os dois valores são FALSO.

* NAO: pode ser entendido como operador logico de NEGAÇÃO, ou seja, ele irá negar uma afirmação e o resultado será invertido. Como no exemplo abaixo:

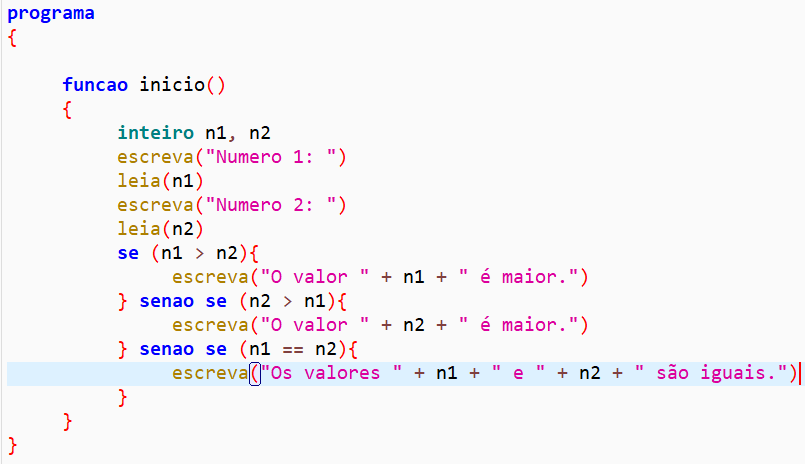


Aqui temos certeza de que o resultado seria FALSO, mas o operador de negação NAO, irá alterar o resultado para o contrário.

**Aula 08: Condições Aninhadas**

Condições aninhadas são aquelas que tem mais do um resultado ou outro, e para conseguirmos declarar mais do que uma condição podemos usar o **senao se**, para assim podermos ter mais de um resultado esperado.

Um exemplo que podemos usar aqui é o exercício 26, como na imagem abaixo:



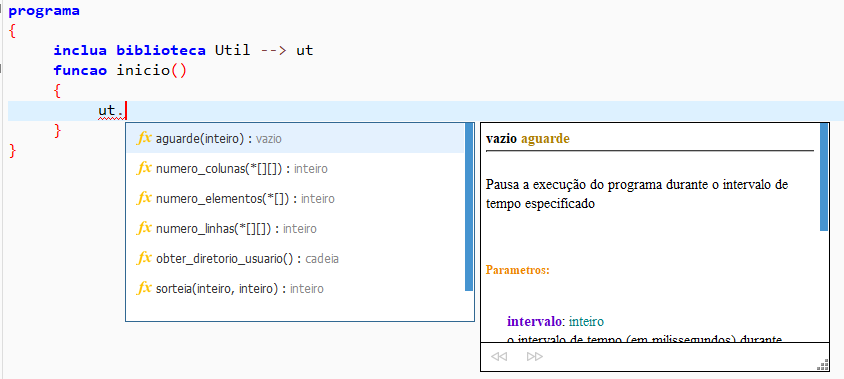
E tendo assim esse conhecimento, podemos resolver os exercícios seguintes do 26 até o 30 que é o nosso **desafio**.

**Aula 09/10: Bibliotecas Adicionais/Jogo da Adivinhação (desafio 32)**

Para continuarmos com nossos exercícios, é preciso entender o conceito de bibliotecas em Portugol. Não é nada tão complicado, mas são “complementos” a linguagem que fazem com que algumas ações complexas se tornem mais fáceis de serem realizadas. A primeira que vamos ver, é a biblioteca UTIL, e como ela funciona?

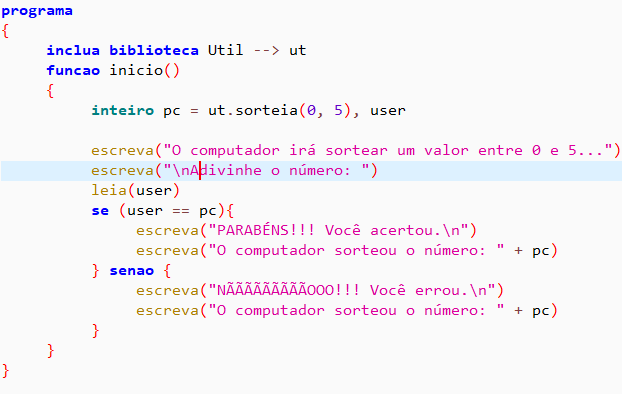
Acima de “funcao" vamos escrever **inclua biblioteca Util**, e para utilizá-la em nosso código, basta chama-la e colocar um sinal de ponto na frente para que assim possa carregar suas funcionalidades de maneira completa.

Uma maneira de deixar mais prático invocar a biblioteca, é usando o comando “-->” para dar um apelido a ela.

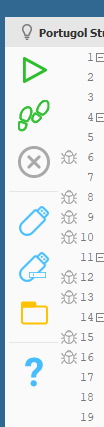
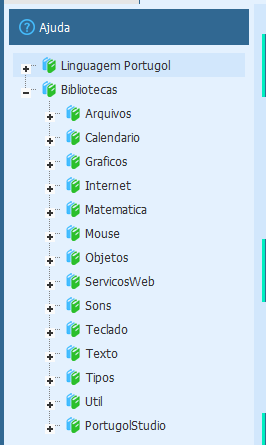


No exemplo demos o “apelido” de ut, mas se quisermos deixar sem o “apelido”, irá funcionar da mesma maneira.

Feito isso, vamos então a resolução do desafio 32 para entender de fato como realizar o uso de uma biblioteca.



Para fazer bom uso e entender melhor como usar uma biblioteca, dentro do Portugol procure pela “AJUDA” e depois abra “Bibliotecas”. Como nas figuras a seguir:

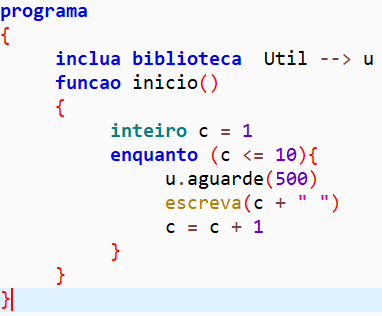


**Aula 11: Como vai ser a prova**

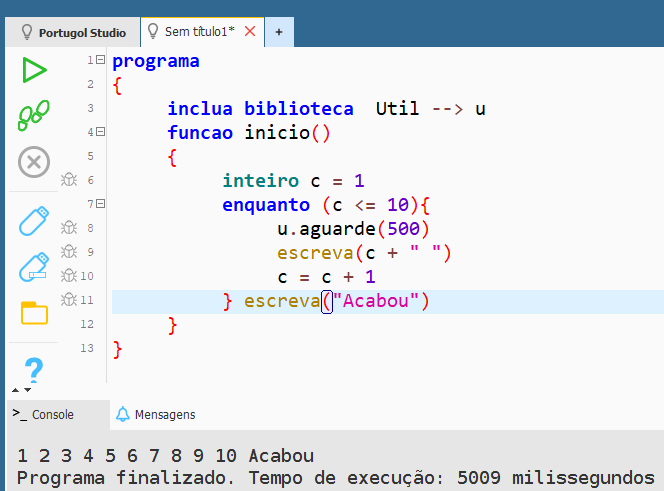
Aula destinada a alunos com o curso realizado de maneira presencial.

**Aula 12: Repetição enquanto (parte 1)**

Vamos entender como que funciona uma estrutura de repetição, usando o comando **enquanto**, e assim também aplicando uma validação. Na situação apresentada abaixo, temos um problema precisamos que o contador declarado como **c** pare sua contagem quando receber o valor determinado na validação.



Para acompanhar a contagem é simples, basta colocar dentro do bloco da estrutura **enquanto** (*dentro dos* ***{ }***), o comando escreva e nele colocar a nossa variável c. Podemos também, colocar um texto, pois a estrutura funciona da seguinte maneira, enquanto **c** for menor que **10**, tudo dentro da estrutura é feito novamente, até que o valor determinado seja alcançado. Quando for alcançado o código continua se houver outros comandos após a estrutura de repetição **enquanto**.



Dessa maneira podemos obter a contagem, e por meio de uma mensagem ao final que “acabou”, e acredito ser bem explicativo que a mensagem apenas foi mostrada porque o bloco de repetição acabou, assim nosso código pode prosseguir para o comando seguinte.