Programação em Portugol Studio

Luciana Ferreira Baptista
Ronildo Aparecido Ferreira
Monica Martos de Rezende Campos Aguirre

Jundiaí
2023

Sumário

1. Um pouco da história

Portugol Studio é uma ferramenta baseada em C e PHP, em que estudantes de programação podem criar algoritmos totalmente em português. Possui uma sintaxe fácil com diversos exemplos e material de apoio à aprendizagem.

Desenvolvida pela Universidade do Vale do Itajaí (UNIVALI) pelo estudante Luiz Fernando Noschang, sua concepção teve início em 2007 com o intuito de facilitar a transição dos estudantes para linguagens de programação profissionais.

Seu lançamento ocorreu em 7 de julho de 2009 sob licença GPL 3.0 e em setembro de 2011 foi lançada sua versão com uma interface para uso da linguagem na versão 1.0 Portugol Studio, sendo utilizada exclusivamente por estudantes dos cursos de ciências da computação da UNIVALI, outras universidades, como a Universidade Federal do ABC, começaram a utilizar à partir da versão 2.0.

Na Copa Rio Info de Algoritmos (CRIA) foi utilizada a versão 2.5 do Portugol Studio como linguagem principal para a resolução dos desafios.

O Portugol Studio possui uma interface didática e simplificada, contendo poucos botões, apresentando somente o necessário, diminuindo a complexidade de uso da IDE e permitindo acesso rápido às funcionalidades.

Seu sistema de ajuda, navegável e sequencial permite que o usuário aprenda a linguagem e como programar dentro da ferramenta, a Árvore Estrutural, permite identificar e compreender os elementos existentes no código, com um sistema de busca por nome dos elementos, o Editor de código, possuiu destaque de sintaxe, conclusão de código para bibliotecas, dobramento de código e troca de temas, o Depurador, permite executar o programa passo a passo, podendo verificar por onde a execução passa, seu Inspetor de variáveis, permite a visualização dos valores das variáveis do código durante a execução ou depuração e Suporte para desenvolvimento de Jogos, bibliotecas facilitam a criação de programas mais complexos com o uso de interface gráfica.

2. Instalação

Comece baixando o instalador no site (Figura 1):

https://univali-lite.github.io/Portugol-Studio/

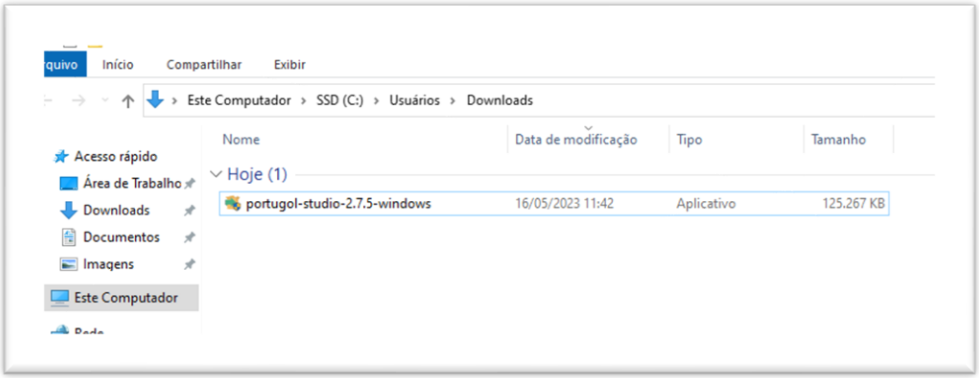
Figura 1-Site do Portugol Studio

Fonte: https://univali-lite.github.io/Portugol-Studio/

Clique em Download para baixar o instalador para Sistema Windows. Se necessário, clique em Outras Versões para instalação em outros sistemas operacionais.

Logo em seguida, entre na pasta Downloads e clique no executável portugol-studio-2.7.5-windows conforme Figura 2.

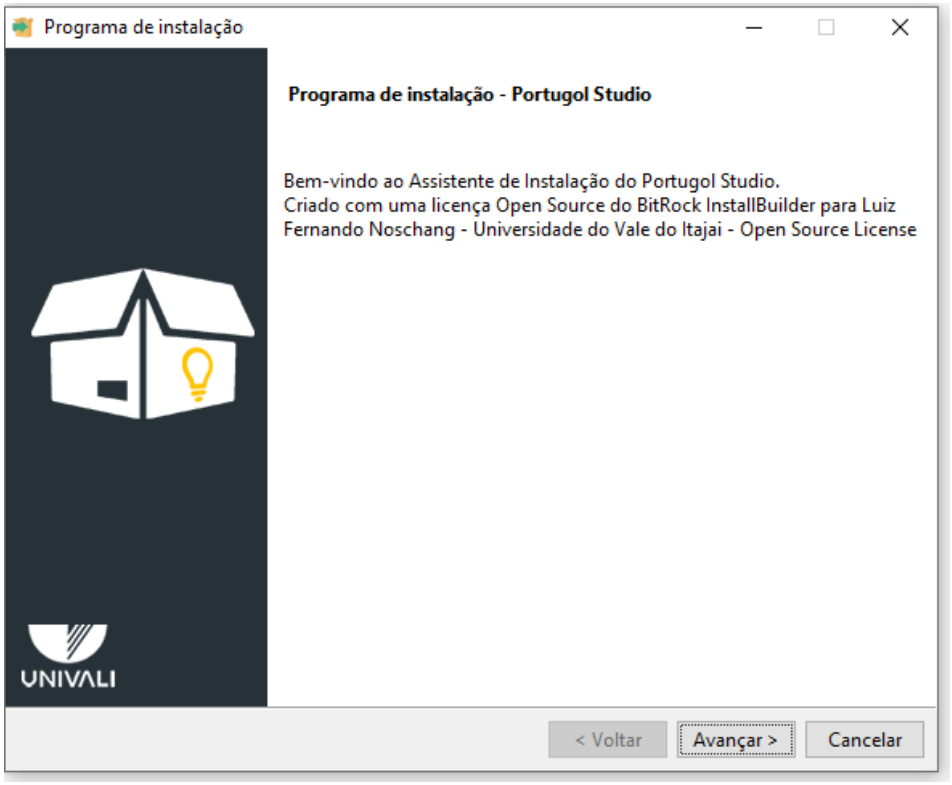
Figura 2-Executável do Portugol Studio



Fonte: Próprios autores, 2023

A instalação irá iniciar e na tela de Programa de Instalação – Portugol Studio clique em Avançar para dar seguimento conforme Figura 3.

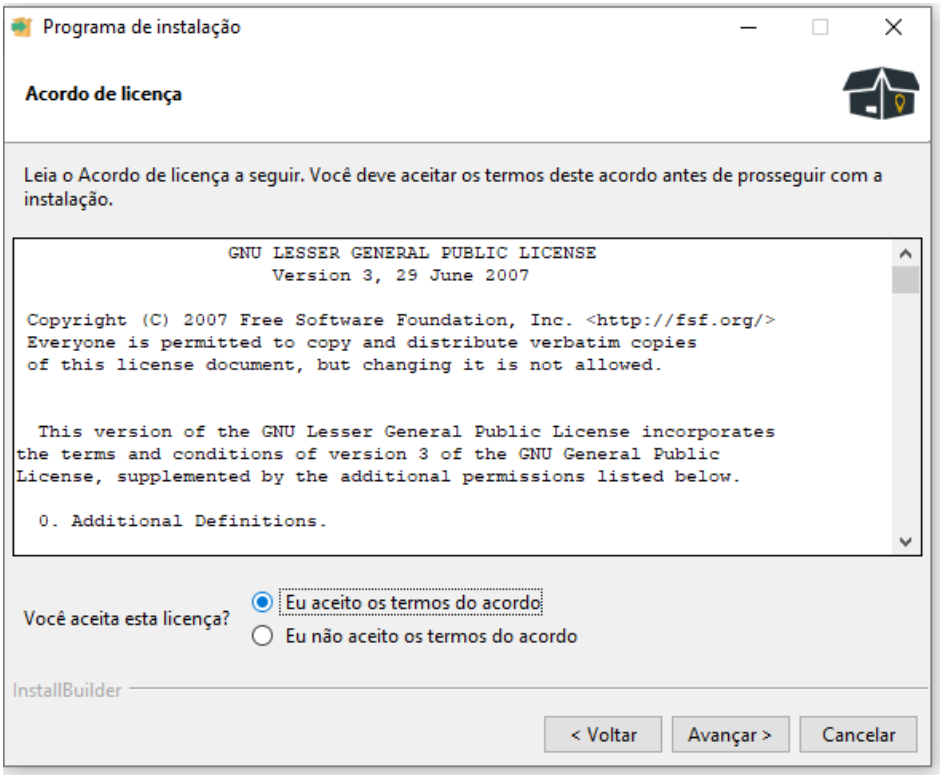
Figura 3-Tela de instalação do Portugol Studio



Fonte: Tela da instalação do portugol-studio-2.7.5-windows.exe, 2023

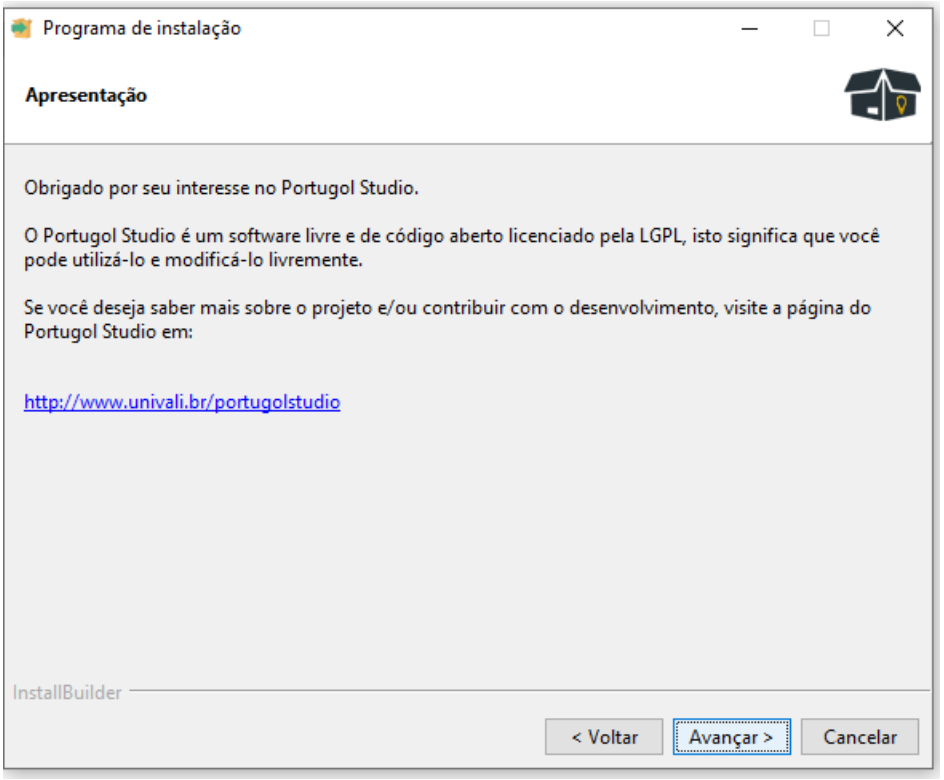
Na tela de Acordo de Licença (Figura 4), selecione a opção "Eu aceito os Termos de Acordo" e logo em seguida clique em Avançar até a Figura 9.

Figura 4-Clique aceitar termos do Portugol Studio



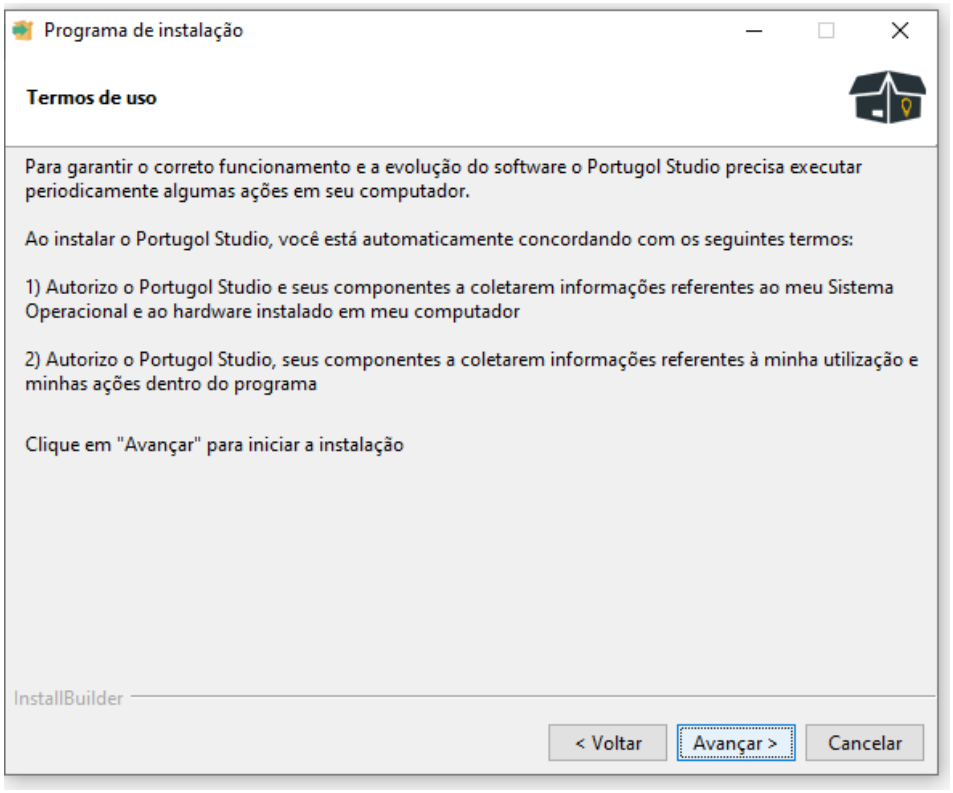
Fonte: Tela da instalação do portugol-studio-2.7.5-windows.exe, 2023

Figura 5-Tela de Apresentação do Portugol Studio



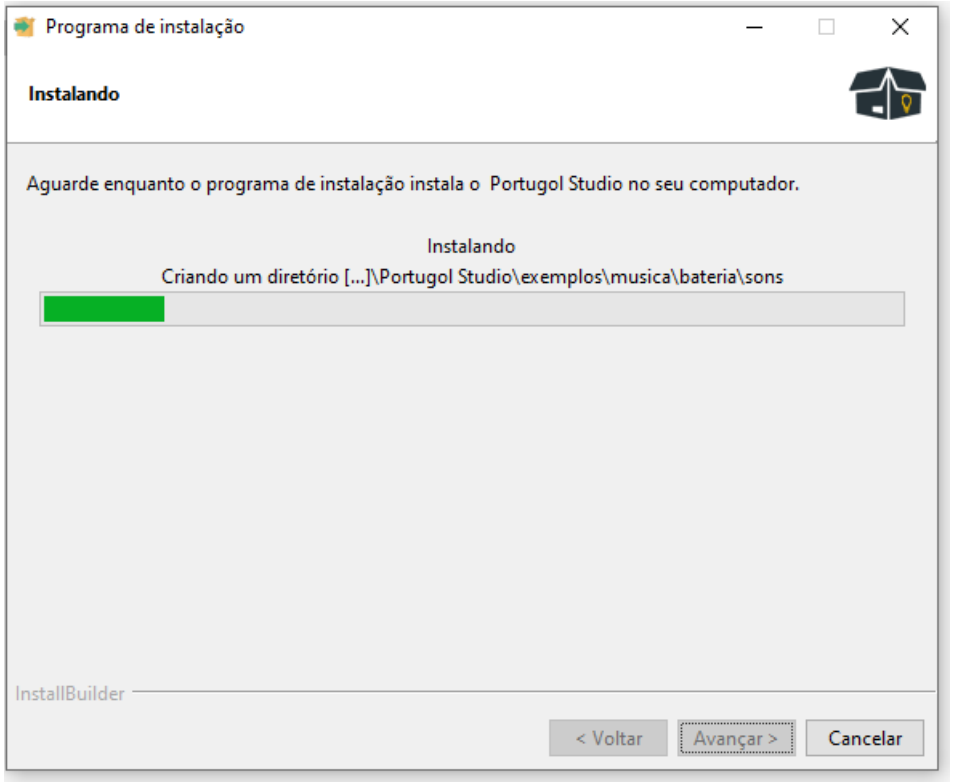
Fonte: Tela da instalação do portugol-studio-2.7.5-windows.exe, 2023

Figura 6-Termos de uso do Portugol Studio



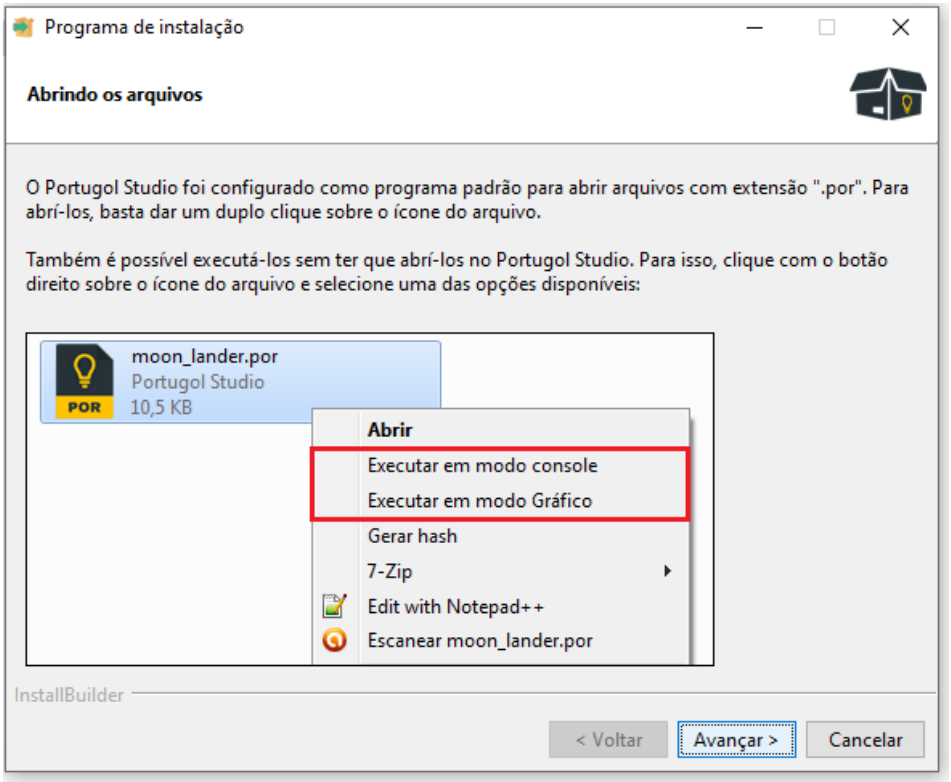
Fonte: Tela da instalação do portugol-studio-2.7.5-windows.exe, 2023

Figura 7-Instalando Portugol Studio



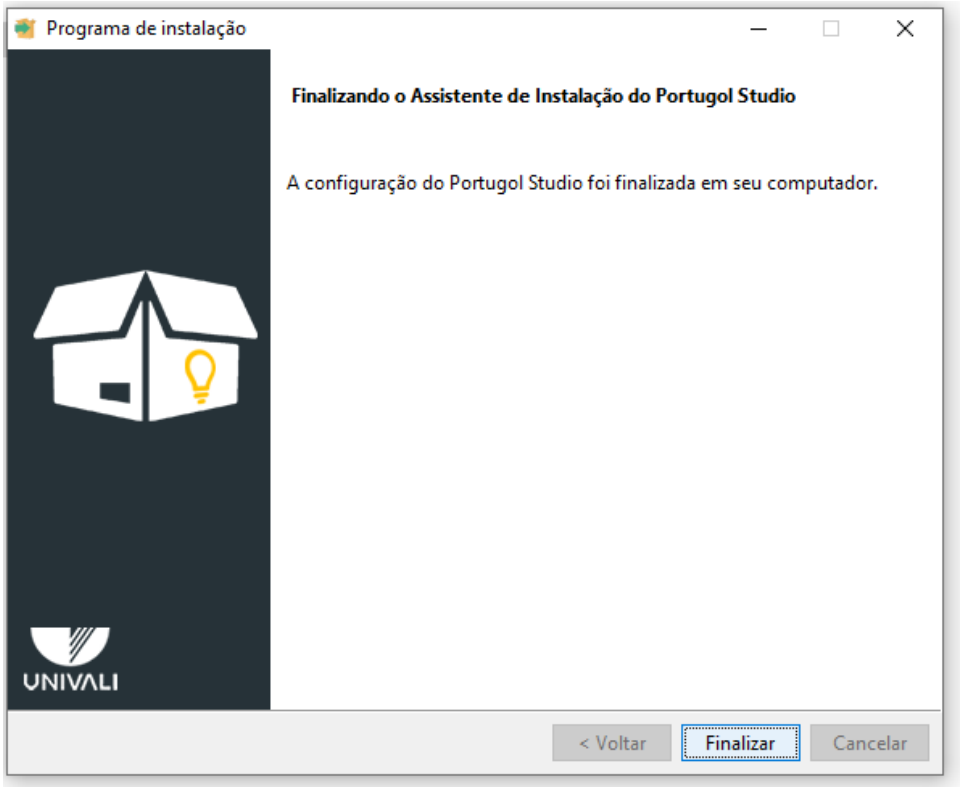
Fonte: Tela da instalação do portugol-studio-2.7.5-windows.exe, 2023

Figura 8-Abrindo Arquivos do Portugol Studio



Fonte: Tela da instalação do portugol-studio-2.7.5-windows.exe, 2023

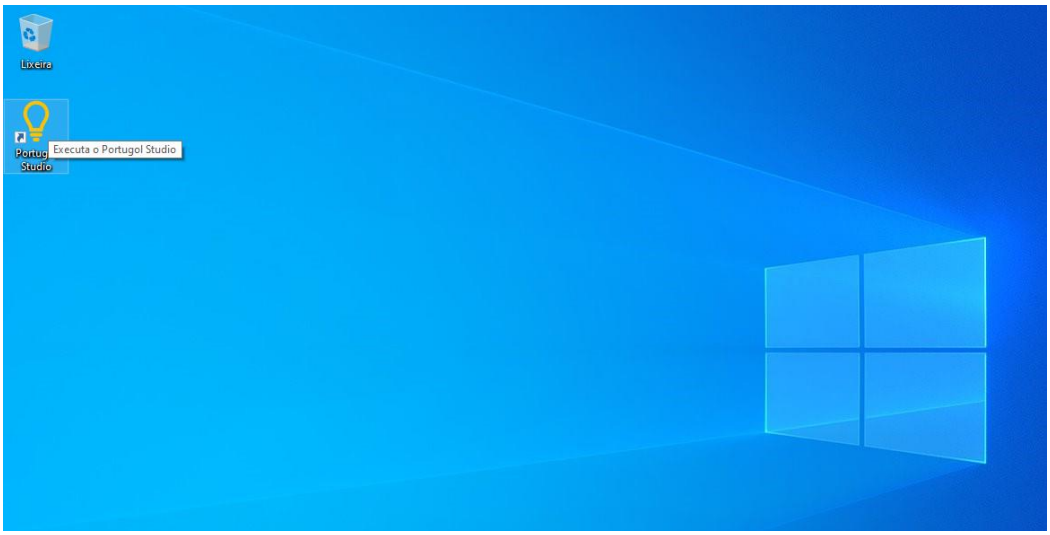
Figura 9-Finalizando Instalação do Portugol Studio



Fonte: Tela da instalação do portugol-studio-2.7.5-windows.exe, 2023

Após a instalação é criado um ícone na Área de Trabalho, conforme apresentado na Figura 10.

Figura 10-Ícone do Portugol Studio na Área de Trabalho



Fonte: Próprios autores, 2023

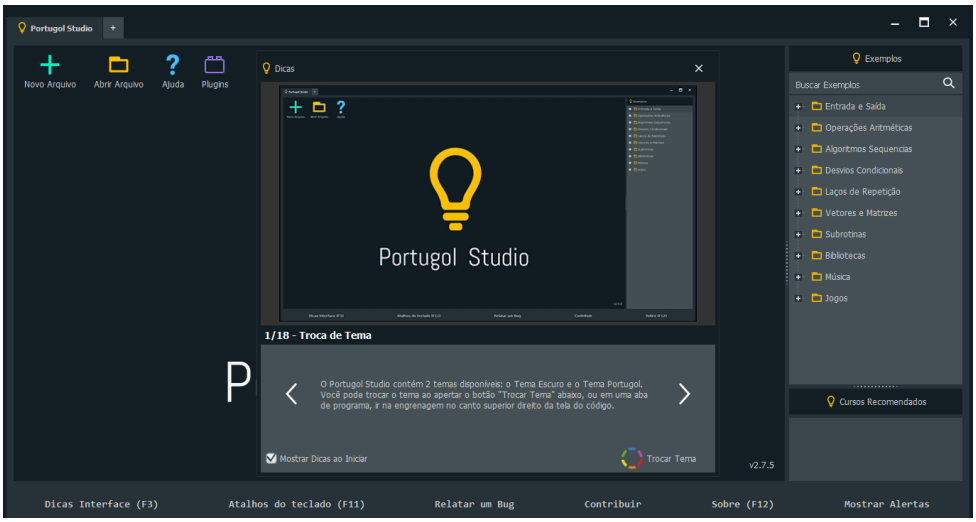
Ao clicar no ícone do Portugal Studio, ele é iniciado (Figura 11).

Figura 11-Iniciando Portugol Studio

Fonte: Tela da execução do portugol-studio.jar, 2023

Na tela inicial do Portugol Studio são apresentadas dicas, através da janela "Dicas", que pode ficar oculta ao desmarcar a opção "Mostrar Dicas ao Iniciar" (Figura 12).

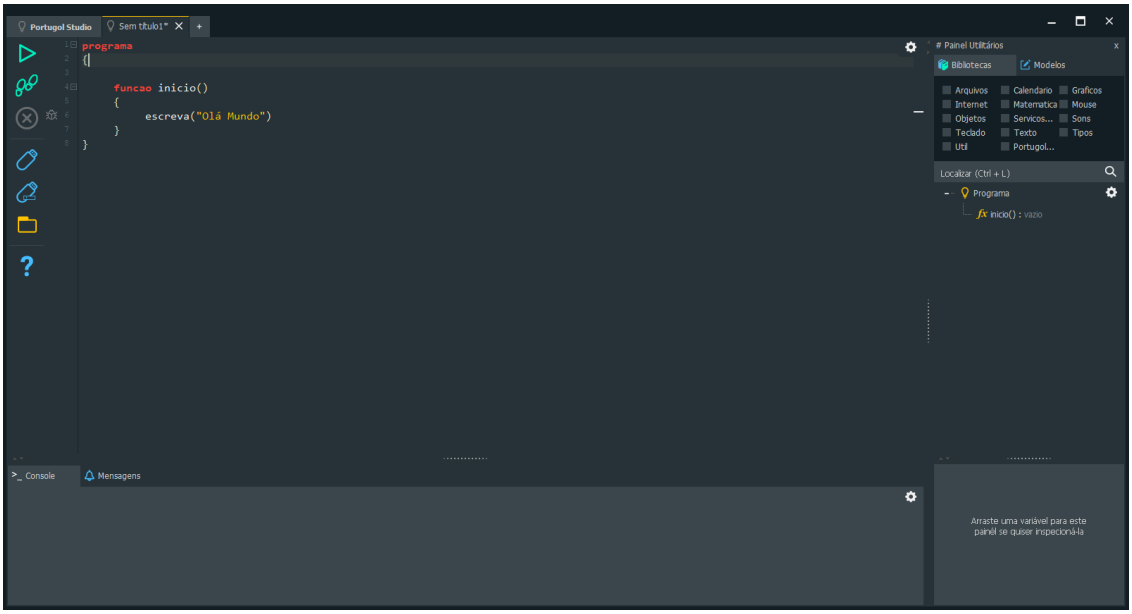
Figura 12-Tela Inicial Portugol Studio



Fonte: Tela da execução do portugol-studio.jar, 2023

Ao clicar em Novo Arquivo, a tela para programa no Portugol Studio é apresentada (Figura 13).

Figura 13-Tela para programar no Portugol Studio



Fonte: Tela da execução do portugol-studio.jar, 2023

Nesta tela, há alguns botões:

 Executa o programa até o próximo ponto de parada

 Executa o programa passo a passo

 Interrompe a execução / depuração do programa atual

 Salva o programa atual no computador em uma pasta escolhida pelo usuário

 Salva uma nova cópia do programa atual no computador em uma pasta escolhida pelo usuário

 Abre um arquivo .por

 Abre a ajuda com sintaxe e bibliotecas

3. Começando a programar

Comentários

Comentários são trechos de texto dentro do código fonte de um programa que não são executados pelo computador. Eles são usados para fornecer informações, explicações ou anotações para os programadores que estão lendo o código. Comentários são essenciais para tornar o código mais compreensível e documentar a lógica por trás das instruções.

// comentário de linha

/* ... */ comentário de bloco (mais de uma linha)

Tipos de Dados

Tipos de dados definem o tipo de valor que uma variável ou expressão pode armazenar ou representar.

Os tipos de dados mais comuns incluem:

Tipo cadeia: Para um texto ou muitos caracteres.

cadeia nome_da_cadeia

Tipo caracter: Para um único caracter qualquer.

caracter nome_do_caracter

Tipo inteiro: Para valores inteiros.

inteiro nome_do_inteiro

Tipo real: Para valores com ponto flutuante.

real nome_do_real

Tipo lógico: Para informções do tipo verdadeiro e falso.

logico nome_do_lógico

Variáveis

Variável é o nome utilizado para definir um ou mais valores que são manipulados pelos programas durante a sua operação.

O nome "variável" é utilizado por ser um tipo de conteúdo que pode apresentar diferentes valores enquanto o sistema está em execução.

tipo_de_dado nome_variável.

tipo_de_dado nome_variável1, nome_variável2

tipo_de_dado nome_variável = valor_inicial

Há regras para nomear variáveis:

1. Formado apenas por letras, números e sublinhado ("_");
2. Deve começar com uma letra;
3. Não pode ser igual a palavras reservadas do Portugol Studio.

Constantes

Constantes são utilizadas quando um tipo reservado na memória não pode ter seu valor alterado durante a execução do programa.

const tipo_de_dado NOME_CONSTANTE = valor

Operadores

Operadores são símbolos especiais usados para realizar operações entre variáveis e valores.

Alguns tipos de operadores:

Operadores Aritméticos

Realizam operações matemáticas:

+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (resto da divisão inteira)

Operadores Relacionais

Comparam valores, resultando em verdadeiro ou falso:

<	Menor
>	Maior
<=	Menor ou igual
>=	Maior ou igual
==	Igual
!=	Diferente

Operações Lógicas

Realizam operações lógicas, como "E", "OU" e "NÃO":

nao	(não - negação) primeiro a ser executado
e	segundo a ser executado
ou	terceiro a ser executado

Comandos para interação

leia(variável_1,variável_n)

Usado para receber entradas de dados do usuário. Ele permite que o programa espere o usuário digitar um valor, que será lido pelo programa e armazenado em uma variável.

escreva(texto_ou_variável_1,texto_ou_variável_n)

Usado para exibir informações na tela. Ele permite que você mostre mensagens, valores de variáveis ou qualquer outra informação que deseje exibir para o usuário.

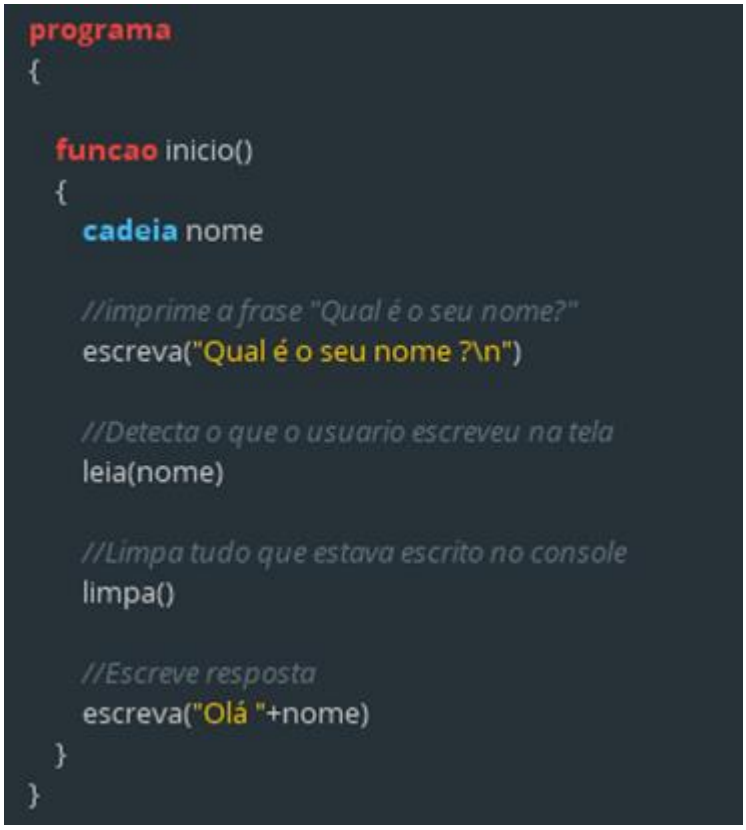
Use também:

- "**\n**": insere uma nova linha
- "**\t**": insere espaços maiores entre os dados

limpa()

Usado para limpar a tela do console, removendo qualquer conteúdo anterior exibido.

Figura 14-Exemplo de "leia", "escreva" e "limpa"



Fonte: Próprios autores, 2023

Exercícios

1) O Haras Cavalos Brilhantes compra cavalos com frequência. Ler a quantidade de cavalos adquiridos recentemente, calcular a quantidade de ferraduras necessárias para os novos cavalos e exibir a quantidade de novos cavalos e sua respectiva quantidade de ferraduras.

2) Ler o nome e a idade do usuário. Calcular a idade em meses e dias. Exibir o nome e a idade em anos, meses e dias.

3) Ler a base e altura de um triângulo. Calcular e exibir a área do triângulo sabendo que área = base * altura / 2.

4) Ler 4 notas, calcular a média ponderada com os pesos 1, 2, 3 e 4 respectivamente e exibir as notas e o resultado da média.

5) O motorista de aplicativo abastece o tanque do seu carro com um determinado valor em reais. Ler o preço do litro de combustível e o valor que pretende abastecer. Calcular a quantidade de litro no abastecimento e exibir os dados lidos e o valor calculado.

6) Ler o peso de uma pessoa em quilos, calcular e mostrar o peso em gramas.

7) Ler a base menor, a base maior e a altura. Calcular e mostrar a área de um trapézio: (base menor + base maior) + altura / 2.

8) O vendedor recebe seu salário fixo acrescido de comissões de vendas, calculada a partir do percentual do valor de suas vendas. Ler o salário fixo do vendedor, o valor de suas vendas e o percentual sobre as vendas. Calcular e exibir o salário final do vendedor.

9) Ler o peso de um boi e o percentual de engorda. Calcular e exibir o novo peso do boi.

10) O caixa do supermercado recebe uma certa quantidade de moedas por dia. Ler a quantidade de moedas recebidas de acordo com cada um dos valores 1, 5, 10, 25 e 50 centavos, e ainda moedas de 1 real. Calcular e exibir o valor recebido de cada um dos tipos de moeda e a soma total em moedas.

4. Estruturas Condicionais

As estruturas condicionais são blocos de código que permitem que o programa tome decisões com base em condições específicas. Elas permitem que você execute diferentes blocos de código dependendo de se uma determinada condição seja verdadeira ou falsa. Em geral, as estruturas condicionais mais comuns são:

Estruturas de Decisão Simples

```
se (condição)
{
        instruções a serem executadas se verdadeiro
}
```

Figura 15-Exemplo de "se"

```
programa
{
    funcao inicio()
    {
        inteiro numero

        escreva("Digite um número: ")
        leia(numero)

        se (numero>0)
        {
            escreva("O número é positivo")
        }
    }
}
```

Fonte: Próprios autores, 2023

```
se (condição)
{
        instruções a serem executadas se verdadeiro
}
senao
{
        instruções a serem executadas se falso
}
```

Figura 16-Exemplo de "se - senao"

```
programa
{
    funcao inicio()
    {
        inteiro numero

        escreva("Digite um número: ")
        leia(numero)

        se (numero>0)
        {
            escreva("O número é positivo")
        }
        senao
        {
            escreva("O número é negativo")
        }
    }
}
```

Fonte: Próprios autores, 2023

```
se (condição)
{
    instruções a serem executadas se verdadeiro
}
senao se (condição)
{
    instruções a serem executadas se falso para a condição anterior
    e se verdadeiro para esta condição
}
```

Figura 17-Exemplo de "se – senao se"

```
programa
{
	funcao inicio()
	{
		real nota

		escreva("Digite a nota obtida pelo aluno: ")
		leia(nota)

		se (nota>=6.0)
		{
			escreva("Você está Aprovado.")
		}
		senao se (nota>=4.0)
		{
			escreva("Você está em Recuperação.")
		}
	}
}
```

Fonte: Próprios autores, 2023

se (condição**)**
{
instruções a serem executadas se verdadeiro
}
senao se (condição**)**
{
instruções a serem executadas se falso para a condição anterior e se verdadeiro para esta condição
}
senao
{
instruções a serem executadas se falso para a condição anterior e se falso para esta condição
}

Figura 18-Exemplo de "se – senao se - senao"

```
programa
{
    funcao inicio()
    {
        real nota

        escreva("Digite a nota obtida pelo aluno na olimpíada: ")
        leia(nota)

        se (nota>=9.0)
        {
            escreva("Excelente! Você está classificado para a Final.")
        }
        senao se (nota>=7.0)
        {
            escreva("Parabéns! Você está classificado para a Semifinal.")
        }
        senao se (nota>=5.0)
        {
            escreva("Muito bem! Você está classificado para Repescagem.")
        }
        senao
        {
            escreva("Que pena. Você não atingiu a nota mínima para continuar.")
        }
    }
}
```

Fonte: Próprios autores, 2023

Estrutura de Decisão Múltipla

```
escolha (variável)
{
        caso valor1
                instruções a serem executadas para o valor1
                pare
        caso valor2
                instruções a serem executadas para o valor2
                pare
        caso valorN
                instruções a serem executadas para o valorN
                pare
        caso contrario
                instruções a serem executadas para nenhum dos valores
                anteriores
}
```

Figura 19-Exemplo de "escolha - caso"

```
programa
{
    funcao inicio()
    {
        inteiro opcao

        escreva("1- Paraná \n")
        escreva("2- Santa Catarina \n")
        escreva("3- Rio Grande do Sul \n")

        escreva("Digite uma opção: ")
        leia(opcao)

        limpa()

        escolha (opcao)
        {
            caso 1:
                escreva ("Paraná")
                pare
            caso 2:
                escreva ("Santa Catarina")
                pare
            caso 3:
                escreva ("Rio Grande do Sul")
                pare
            caso contrario:
                escreva ("Opção Inválida.")
        }
        escreva("\n")
    }
}
```

Fonte: Próprios autores, 2023

Exercícios

1) Ler um ano e verificar se o ano é bissexto. Um ano é bissexto se for divisível por 4, mas não por 100, exceto se for divisível por 400. Exibir o ano lido, informando se é ou não é bissexto.

2) Ler três números, verificar se formam um triângulo e, se sim, exibir se é um triângulo equilátero, isósceles ou escaleno.

3) Ler o valor de uma compra e calcular o desconto, de acordo com o valor total da compra: se for menor que R$100, não há desconto; se for entre R$100 e R$500, o desconto é de 10%; acima de R$500, o desconto é de 20%. Exibir o valor após aplicado o desconto.

4) Ler o tempo de permanência de um veículo em um estacionamento. Calcular e exibir o preço a ser pago, considerando o tempo de permanência do veículo. As primeiras 2 horas custam R$ 2,00 cada, e cada hora adicional custa R$1,00.

5) Ler o preço do etanol e da gasolina. Sugerir o tipo de combustível a ser utilizado em um carro, com base no preço: se o preço do etanol for até 70% do preço da gasolina, é recomendado usar álcool; caso contrário, usar gasolina. Exibir o resultado sugerido.

6) Ler a idade do passageiro. Determinar o preço de uma passagem de acordo com a idade do passageiro: até 2 anos (gratuita), de 3 a 12 anos (meia tarifa) e acima de 12 anos (tarifa completa). Exibir o tipo de tarifa que deve ser aplicada.

7) Ler um caractere. Verificar se o caractere lido é uma vogal. Exibir o caractere lido com a informação se é ou não uma vogal.

8) Ler três números inteiros e exibir o maior desses três números.

9) Ler um número, verificar e exibir se o número é um quadrado perfeito, ou seja, se a raiz quadrada dele é um número inteiro.

10) Ler a classificação indicativa de um filme. Identificar e exibir a categoria desse filme, com base na classificação indicativa: até 10 anos (infantil), de 11 a 14 anos (infantojuvenil), de 15 a 17 anos (juvenil) e acima de 17 anos (adulto).

5. Estruturas de Repetição

As estruturas de repetição, também conhecidas como loops, são usadas para executar um bloco de código repetidamente enquanto uma condição é verdadeira. Essas estruturas são usadas para automatizar tarefas que precisam ser repetidas várias vezes. As estruturas de repetição mais comuns:

Com variável de controle

Usada para executar um bloco de código várias vezes, com um controle explícito sobre o número de iterações. Ela é especialmente útil quando você sabe exatamente quantas vezes deseja repetir uma ação.

```
para (tipo variável=valor_inicial; condição; incremento/decremento)
{
    instruções a serem executadas enquanto a condição for verdadeira
}
```

Figura 20-Exemplo de "para"

```
programa
{
    funcao inicio()
    {
        const inteiro QTDE = 10

        real nota, soma=0.0, media

        para (inteiro i=1; i<=QTDE; i++)
        {
            escreva("Digite a "+i+"ª nota: ")
            leia(nota)
            soma += nota
        }

        media = soma / QTDE

        escreva("\n Média das notas: "+media)
    }
}
```

Fonte: Próprios autores, 2023

Pré-condição

Usada quando o número de iterações não é conhecido antecipadamente e depende da satisfação de uma condição. Executa um bloco de código enquanto uma condição específica for verdadeira.

```
enquanto (condição)
{
    instruções a serem executadas enquanto a condição for verdadeira
}
```

Figura 21-Exemplo de "enquanto"

```
programa
{
    funcao inicio()
    {
        caracter sair = 'N'

        enquanto (sair != 'S')
        {
            escreva("Digite <S> para Sair: ")
            leia(sair)
        }
    }
}
```

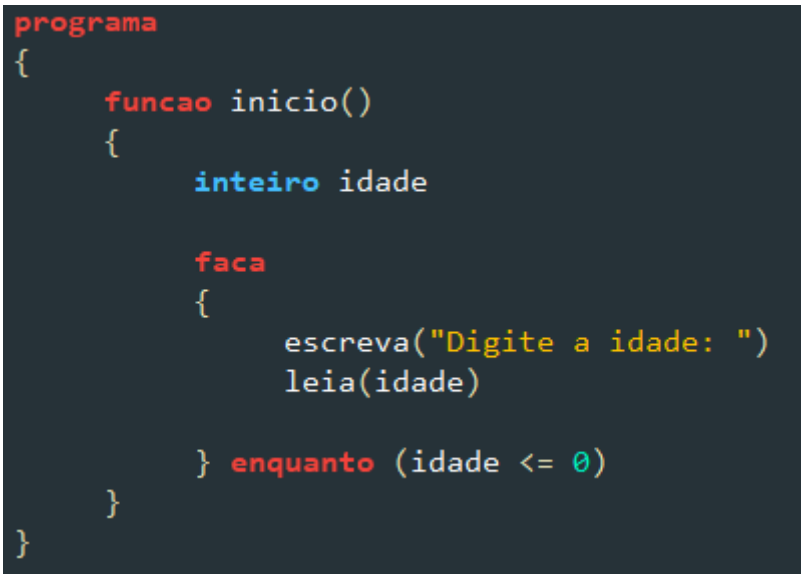
Fonte: Próprios autores, 2023

Pós-condição

Semelhante à "Pré-condição", mas garante que o bloco de código seja executado pelo menos uma vez, mesmo se a condição for falsa desde o início.

```
faca
{
    Instruções executadas pelo menos uma vez e enquanto a condição
      for verdadeira
} enquanto (condição)
```

Figura 22-Exemplo de "faca - enquanto"



Fonte: Próprios autores, 2023

Cada uma dessas estruturas de repetição tem sua aplicação adequada, dependendo da situação. O "para" é ideal quando você sabe o número exato de iterações, o "enquanto" é útil quando a condição é avaliada antes da execução do bloco, e o "faca-enquanto" é útil quando o bloco precisa ser executado pelo menos uma vez, independentemente da condição.

Exercícios

1) Ler um número inteiro. Verificar e exibir se o número é primo.

2) Ler um número inteiro, calcular e exibir separadamente as potências de 2^0 até 2^(esse número).

3) Ler vários números até que seja digitado um número negativo. Calcular e exibir a soma desses números.

4) Ler 10 números. Verificar e exibir o menor e maior número dessa sequência.

5) Ler um número para a parada final e outro número que representa um múltiplo. Exibir os múltiplos do número lido de 1 até o número final lido.

6) Ler vários números até que a soma desses números seja maior que 100. Exibir a multiplicação dos números lidos.

7) Ler a quantidade de pessoas e ler a altura dessas pessoas. Calcular e exibir a altura média de todas as pessoas.

8) Ler um número. Exiba a soma de todos os números pares e ímpares de zero até o número lido.

9) Ler um número entre 50 e 100. Exibir a sequência de Fibonacci até esse número.

10) Simule um jogo de adivinhação. O programa deve gerar um número aleatório e o jogador precisa acertar o número. Cada vez que o jogador informar o número, deve exibir se o palpite é muito alto, muito baixo ou correto. Quando for correto, exibir a quantidade de palpites.

6. Funções

Funções são blocos de código independentes que podem ser reutilizados para realizar uma tarefa específica. Elas permitem dividir um programa em partes menores e mais gerenciáveis, facilitando a organização e a manutenção do código. Funções podem receber parâmetros como entrada, executar um conjunto de instruções e retornar um valor de saída.

```
funcao tipo nome_função (tipo parametro1, tipo paramentroN)
{
        retorne conteúdo, somente se não retornar vazio
}
```

Figura 23-Exemplo de declaração de função

```
programa
{
    funcao inicio()
    {

    }

    //Declaração de função sem retorno e sem parâmetros
    funcao exibeLinha()
    {

    }

    //Declaração de função com retorno do tipo cadeia e sem parâmatros
    funcao cadeia horario()
    {
        retorne "16:44"
    }

    //Declaração de função sem retorno e com parâmetro do tipo inteiro
    funcao exibeAsteriscos(inteiro qtdeAsterisco)
    {

    }

    //Declaração de função com retorno do tipo real e com parâmetros do tipo real
    funcao real soma(real numero1, real numero2)
    {
        retorne 7.1
    }
}
```

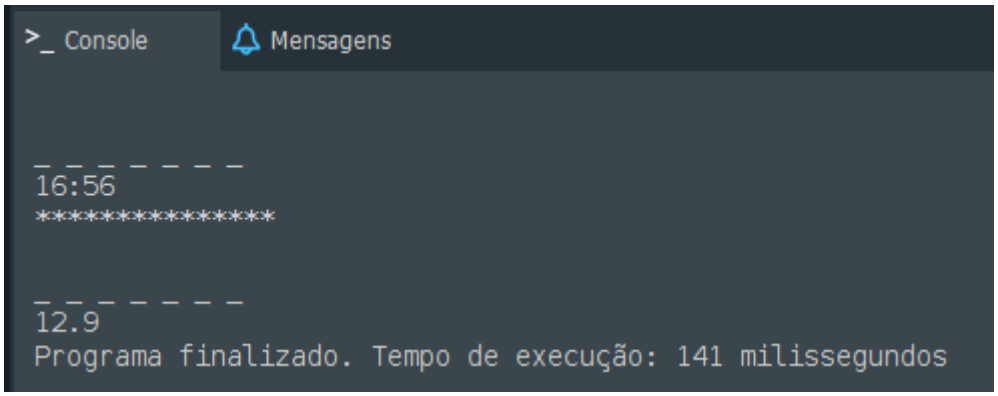
Fonte: Próprios autores, 2023

Figura 24-Exemplo de funções

```
programa
{
	inclua biblioteca Calendario

	funcao inicio()
	{
		//Invocação de todas as funções declaradas neste programa
		exibeLinha()
		escreva(horario())
		exibeAsteriscos(15)
		escreva(soma(3.1,9.8))
	}


	//Declaração de função sem retorno e sem parâmetros
	funcao exibeLinha()
	{
		escreva("\n_ _ _ _ _ _ _\n")
	}

	//Declaração de função com retorno do tipo cadeia e sem parâmatros
	funcao cadeia horario()
	{
		retorne Calendario.hora_atual(falso)+":"+Calendario.minuto_atual()
	}

	//Declaração de função sem retorno e com parâmetro do tipo inteiro
	funcao exibeAsteriscos(inteiro qtdeAsterisco)
	{
		escreva("\n")
		para (inteiro i=1; i<=qtdeAsterisco; i++)
		{
			escreva("*")
		}
		escreva("\n")
	}

	//Declaração de função com retorno do tipo real e com parâmetros do tipo real
	funcao real soma(real numero1, real numero2)
	{
		exibeLinha() //Invocação de função
		retorne numero1 + numero2
	}
}
```

Fonte: Próprios autores, 2023

Figura 25- Console do programa que apresenta exemplo de funções



Fonte: Tela da execução do portugol-studio.jar, 2023

Exercícios

1) Ler um número inteiro. Exibir a soma dos números ímpares até o número lido. Criar uma função que retorne verdadeiro para números ímpares.

2) Ler a base e o expoente. Criar uma função para efetuar o cálculo para potência utilizando estrutura de repetição. Exibir o resultado do cálculo.

3) Ler o sexo e a altura. Calcular e exibir o peso ideal. Criar uma função para o cálculo quando o sexo for masculino utilizando a fórmula 72,7 x altura - 58 e outra função para o sexo feminino utilizando a fórmula 62,1 x altura - 44,7.

4) Ler dois números inteiros e efetuar os seguintes cálculos: adição, subtração, multiplicação e divisão. Criar função para cada um dos cálculos. Exibir o resultado de todos os cálculos.

5) Ler uma palavra e exibir se a palavra é um palíndromo (lê-se igual de trás para frente). Criar uma função para retornar verdadeiro ou falso na verificação do palíndromo.

6) Ler um número e exibir o dia da semana. Criar uma função que determine o dia da semana correspondente ao número.

7) Ler 5 números e exibir a ordem desses números. Criar uma função que retorne ordem crescente, ordem decrescente ou não ordenado na verificação da ordenação.

8) Ler o raio de 5 círculos e exibir a soma da área de todos os círculos. Criar uma função para calcular a área de um círculo.

9) Ler uma palavra e exibir essa palavra substituindo as vogais por *. Criar uma função para realizar essa substituição.

10) Ler um nome completo de uma pessoa e exibir a quantidade de vogais e consoantes presentes nesse nome. Criar uma função para a verificação das vogais e outra função para a verificação das consoantes.

7. Vetores

Também conhecido como array unidimensional, são estruturas de dados que permitem armazenar vários elementos do mesmo tipo em uma única variável. Os elementos são acessados por meio de um índice que começa em zero e vai até o tamanho do vetor menos um. Vetores são úteis para armazenar e manipular conjuntos de valores de maneira eficiente.

O índice é um valor inteiro, e o primeiro elemento do vetor é encontrado no índice 0 (zero), enquanto o último elemento está localizado no índice equivalente a dimensão-1, como demonstrado na Figura 26.

Figura 26-Representação de Vetor em linha

índice:	0	1	2				n-1
conteúdo:	**Caio**	**Ivo**	**Léo**				**Kim**

Fonte: Próprios autores, 2023

tipo_de_dado nome_variável_vetor[dimensão]

A dimensão de um vetor pode ser implícita quando valores são atribuídos durante sua declaração. Nessa situação, a dimensão será determinada automaticamente com base na quantidade de valores separados por vírgula:

tipo_de_dado nome_variável_vetor[] = {valores_iniciais}

Figura 27-Ler 4 notas, calcular e exibir a média aritmética

```
programa
{
    funcao inicio()
    {
        real nota[4], media=0.0
        para (inteiro i=0; i<4; i++)
        {
            escreva("Digite a "+(i+1)+"ª nota: ")
            leia(nota[i])
            media += nota[i]
        }
        escreva("Média: ",media/4)
    }
}
```

Fonte: Próprios autores, 2023

No Portugol Studio, há um exemplo de uso de vetor em:

Bibliotecas → Util → Tamanho do Vetor (Figura 28).

Figura 28-Exemplo vetor.por do Portugol Studio

```
programa
{
    inclua biblioteca Util --> u

    funcao inicio()
    {
        // Cria um vetor com 5 elementos
        inteiro vet[] = { 4, 2, 9, 3, 8 }

        // Experimente substituir o vetor acima por este e veja que
        // o programa consegue percorrer normalmente o novo vetor

        // inteiro vet[] = { 4, 2, 9, 3, 8, 6, 5, 6, 2, 3, 9, 1 }

        inteiro elementos = u.numero_elementos(vet)

        escreva("O vetor possui ", elementos, " elementos:\n\n")

        // Utilizamos o valor obtido anteriormente para percorrer
        // o vetor e exibir seus valores
        para (inteiro elemento = 0; elemento < elementos; elemento++)
        {
            se (elemento == 0)
            {
                escreva("{ ")
            }

            escreva(vet[elemento])

            se (elemento < elementos - 1)
            {
                escreva(", ")
            }

            se (elemento == elementos - 1)
            {
                escreva(" }")
            }
        }

        escreva("\n")
    }
}
```

Fonte: Tela da execução do portugol-studio.jar, 2023

Exercícios

1) Ler 10 números, armazenando-os em um vetor. Substituir todos os números negativos por zero. Exibir todos os números do vetor.

2) Ler 15 números, armazenando-os em um vetor. Contar e exibir quantos números pares e ímpares existem no vetor.

3) Ler 20 letras, armazenando-as em um vetor. Ler uma letra qualquer e exibir quantas vezes essa letra é encontrada no vetor.

4) Ler 10 nomes, armazenando-os em um vetor. Classificar o vetor em ordem decrescente. Exibir o conteúdo do vetor em ordem crescente.

5) Ler 20 números, armazenando-os em um vetor. Encontre e exiba a posição (índice) do primeiro valor negativo desse vetor.

6) Ler 15 números, armazenando-os em um vetor. Verificar e exibir a posição (índice) do menor e do maior elemento do vetor.

7) Ler 10 números, armazenando-os em um vetor. Inverter o sinal dos elementos desse vetor, armazenando-os em outro vetor. Exibir a soma de todos os elementos positivos dos dois vetores.

8) Ler 20 letras, armazenando-as em um vetor. Conte a quantidade de cada vogal, armazenando-as em outro vetor de 5 elementos. Exiba o conteúdo do segundo vetor.

9) Ler 10 números, armazenando os números ímpares em um vetor e os números pares em outro vetor. Classifique os vetores em ordem crescente e exiba a soma entre os elementos dos dois vetores, armazenando-a em um terceiro vetor. Exiba todos os vetores.

10) Ler 10 números, armazenando-os em um vetor. Exibir o terceiro menor valor presente nesse vetor.

8. Matrizes

Matrizes são estruturas de dados bidimensionais que consistem em linhas e colunas. Elas permitem armazenar informações em uma grade organizada. Cada elemento de uma matriz é identificado por um par de índices, um para a linha e outro para a coluna. Matrizes são usadas para lidar com conjuntos de dados multidimensionais, como tabelas ou imagens.

Figura 29-Representação de Matriz

	Coluna 0	Coluna 1	Coluna 2		Coluna n-1
Linha 0	**1.0**	**7.5**	**5.2**		**4.1**
Linha 1	**4.6**	**8.0**	**9.6**		**5.2**
Linha 2	**7.7**	**8.5**	**5.7**		**4.7**
Linha n-1	**9.3**	**5.2**	**7.3**		**3.4**

Fonte: Próprios autores, 2023

tipo_de_dado nome_variável_matriz[dimensão_linha] [dimensão_coluna]

A dimensão de uma matriz pode ser inferida se valores já estiverem atribuídos durante sua declaração. Nesse cenário, a dimensão será determinada com base na quantidade de valores presentes em linhas e colunas, separados por vírgula:

tipo_de_dado nome_variável_matriz[] [] =
{valores_inicias},{valores_inicias},...}

Figura 30-Ler 4 notas de 2 disciplinas, calcular e exibir a média aritmética

```
programa
{
    funcao inicio()
    {
        real nota[2][4]
        real media[2]

        para (inteiro i=0; i<2; i++)
        {
            media[i] = 0.0
        }

        para (inteiro linha=0; linha<2; linha++)
        {
            para (inteiro coluna=0; coluna<4; coluna++)
            {
                escreva("Digite a "+(coluna+1)+"ª nota da "+(linha+1)+"ª disciplina: ")
                leia(nota[linha][coluna])
                media[linha] += nota[linha][coluna]
            }
            escreva("\n")
        }

        para (inteiro i=0; i<2; i++)
        {
            escreva("Média da "+(i+1)+"ª disciplina: ",media[i]/4,"\n")
        }
    }
}
```

Fonte: Próprios autores, 2023

Exercícios

1) Ler números para serem armazenados em uma matriz 4x4. Calcular e exibir a soma de todos os elementos.

2) Ler números para serem armazenados em uma matriz 2x2. Verificar e exibir se é uma matriz simétrica.

3) Ler números para serem armazenados em uma matriz 3x3 e ler um número escalar. Multiplicar e exibir os elementos da matriz pelo número escalar.

4) Ler números para serem armazenados em duas matrizes 2x2. Verificar e exibir se as duas matrizes são iguais.

5) Ler números para serem armazenados em uma matriz 3x3. Calcular e exibir a soma de das linhas pelas colunas da matriz.

6) Ler números para serem armazenados em uma matriz 3x3. Verificar e exibir se é uma matriz de identidade.

7) Ler números para serem armazenados em uma matriz 4x4. Verificar e exibir as posições (linha e coluna) do menor e do maior elemento da matriz.

8) Ler números para serem armazenados em uma matriz 3x3. Verificar e exibir se é uma matriz triangular superior.

9) Ler números para serem armazenados em uma matriz 4x4. Verificar e exibir a quantidade de números negativos e positivos por linha.

10) Ler números para serem armazenados em uma matriz 5X5. Verificar e exibir o maior valor da diagonal principal.

Para realizar os exercícios

Antes de começar a programar, organize o exercício para compreendê-lo melhor e assim passar para o Portugol Studio.

A seguir, alguns exemplos:

1 – SOMAR NÚMEROS

DESCRIÇÃO:

Ler dois números inteiro e exibir a soma deles.

INFORMAÇÕES ADICIONAIS:

S = A + B, onde S, A, B pertencem ao conjunto de números inteiros.

EXEMPLOS DE ENTRADA/SAÍDA:

ENTRADA	SAÍDA
Digite o primeiro valor: 2 Digite o segundo valor: 3	
	A soma de 2 com 3 é 5

Digite o primeiro valor: 13 Digite o segundo valor: 2	
	A soma de 13 com 2 é 15

2 – DIVISÃO DE NÚMEROS

DESCRIÇÃO:

Ler dois números reais e exibir a divisão deles.

INFORMAÇÕES ADICIONAIS:

S = A / B, onde S, A, B pertencem ao conjunto de números reais.

EXEMPLOS DE ENTRADA/SAÍDA:

ENTRADA	SAÍDA
Digite o primeiro valor: 2.8 Digite o segundo valor: 2.2	
	A soma de 2.8 com 2.2 é 1,272727272727273

Digite o primeiro valor: 15 Digite o segundo valor: 2	
	A soma de 15 com 2 é 7.5

3 – SOMAR NÚMEROS

DESCRIÇÃO:

Um automóvel percorre uma certa distância em velocidade constante. Ler a velocidade e a distância percorrida e calcular o tempo gasto neste percurso.

INFORMAÇÕES ADICIONAIS:

Tempo = Distância / Velocidade, onde Tempo, Distância e Velocidade pertencem ao conjunto de números reais. A distância em km (quilômetros), a velocidade em km/h (quilômetros por hora) e o tempo em h (hora)

EXEMPLOS DE ENTRADA/SAÍDA:

ENTRADA	SAÍDA
Digite a distância percorrida (km): 200 Qual a velocidade (km/h): 100	
	O tempo gasto para percorrer a distância de 200 km com uma velocidade constante de 100 km/h é de 2 horas

Digite a distância percorrida (km): 235 Qual a velocidade (km/h): 85	
	O tempo gasto para percorrer a distância de 235 km com uma velocidade constante de 85 km/h é de 2,764705882352941 horas

9. Biblioteca

A "Biblioteca" no Portugol Studio se refere a um conjunto de funções pré-definidas e recursos que podem ser utilizados para ampliar a funcionalidade do seu programa. Essas funções são agrupadas em categorias específicas, permitindo que você acesse facilmente diferentes funcionalidades sem precisar implementar tudo do zero.

O Portugol Studio oferece diversas bibliotecas internas que cobrem várias áreas, como matemática, manipulação de strings, operações de tempo, entre outras. Essas bibliotecas podem ser incluídas no seu código para estender as capacidades do seu programa.

Aqui estão alguns exemplos de bibliotecas do Portugol Studio:

- **biblioteca Matematica:** Fornece funções matemáticas, como raiz(), potencia(), seno(), cosseno(), etc.
- **biblioteca Texto:** Oferece funções para manipulação de textos, como numero_caracteres(), substituir(), caixa_alta(), entre outros.
- **biblioteca Util:** Inclui funções relacionadas sorteio de números, pausas, vetores, como sorteia(), aguarde(), numero_elementos(), entre outras.

Essas bibliotecas facilitam o desenvolvimento de programas mais complexos, pois você pode aproveitar as funcionalidades pré-implementadas em vez de criar tudo manualmente. Para usar uma biblioteca, você precisa importá-la no início do seu programa.

A seguir serão apresentados alguns exemplos de programas que usam as bibliotecas.

Figura 31-Raiz de Um Número

```
programa
{
    inclua biblioteca Matematica --> mat

    funcao inicio()
    {
        real numero
        real raiz

        numero = 4.0
        raiz = mat.raiz(numero, 2.0) // Obtém a raíz quadrada do número

        escreva("A raíz quadrada de ", numero , " é: ", raiz, "\n")

        numero = 27.0
        raiz = mat.raiz(numero, 3.0) // Obtém a raíz cúbica do número

        escreva("A raíz cúbica de ", numero , " é: ", raiz, "\n")
    }
}
```

Fonte: portugol-studio.jar, 2023

Figura 32-Potência Entre Dois Números

```
programa
{
    inclua biblioteca Matematica --> mat

    funcao inicio()
    {
        real base, quadrado, cubo, resultado

        escreva("Informe um número: ")
        leia(base)

        // Eleva o número informado ao quadrado
        quadrado = mat.potencia(base, 2.0)
        escreva("\n", base, " ao quadrado é igual a: ", quadrado)

        // Eleva o número informado ao cubo
        cubo = mat.potencia(base, 3.0)
        escreva("\n", base, " ao cubo é igual a: ", cubo, "\n")
    }
}
```

Fonte: portugol-studio.jar, 2023

Figura 33-Tamanho do Texto

```
programa
{
    inclua biblioteca Texto --> tx

    funcao inicio()
    {
        cadeia nome
        inteiro tamanho

        escreva("Digite seu nome: ")
        leia(nome)

        // Obtém o número de caracteres armazenados na variável
        tamanho = tx.numero_caracteres(nome)

        escreva("Seu nome possui ", tamanho, " caracteres\n")

    }
}
```

Fonte: portugol-studio.jar, 2023

Figura 34-Caixa Alta

```
programa
{
    inclua biblioteca Texto --> tx

    funcao inicio()
    {
        cadeia nome, bem_vindo

        escreva("Digite seu nome em letras minúsculas: ")
        leia(nome)

        // Transforma os caracteres da cadeia em caracteres maiúsculos
        nome = tx.caixa_alta(nome)

        // Transforma os caracteres da cadeia em caracteres maiúsculos
        bem_vindo = tx.caixa_alta("bem vindo(a)")

        limpa()
        escreva(bem_vindo, " ", nome, "!!\n")
    }
}
```

Fonte: portugol-studio.jar, 2023

Figura 35-Sorteio de Um Número

```
programa
{
    inclua biblioteca Util --> u

    funcao inicio()
    {
        inteiro valor_inicial
        inteiro valor_final
        inteiro valor_sorteado
        inteiro sorteios

        escreva("Informe um valor inicial: ")
        leia(valor_inicial)

        escreva("Informe um valor final: ")
        leia(valor_final)

        escreva("Informe quantos valores deseja sortear: ")
        leia(sorteios)

        para (inteiro i = 1; i <= sorteios; i++)
        {
            // Sorteia um número entre os valores informados, incluindo
            // o próprio valor inicial e final
            valor_sorteado = u.sorteia(valor_inicial, valor_final)

            escreva("\nSorteio nº ", i, ": ", valor_sorteado)
        }

        escreva("\n")
    }
}
```

Fonte: portugol-studio.jar, 2023

Figura 36-Aguardar Intervalo

```
programa
{
    inclua biblioteca Util --> u

    funcao inicio()
    {
        para (inteiro contador = 10; contador >= 0; contador--)
        {
            limpa()
            escreva("Lançando o foguete em: ", contador)

            /* Faz com que o programa espere 1 segundo antes de fazer a próxima contagem.
             *
             * O intervalo que o programa deve aguardar é informado sempre em milissegundos.
             * Como 1 segundo equivale a 1000 milissegundos, neste caso, informamos o valor
             * 1000.
             *
             * Para fazer com que o programa aguarde apenas meio segundo, podemos informar o
             * valor 500, isto também fará com que a contagem seja mais rápida.
             */
            u.aguarde(1000)
        }

        limpa()
        escreva("O foguete foi lançado!!\n")
    }
}
```

Fonte: portugol-studio.jar, 2023

10. Referências

PORTUGOL STUDIO. **Portugol Studio**. Disponível em: <http://lite.acad.univali.br/portugol/>. Acesso em 11 mar. 2023.