# LÓGICA DE PROGAMAÇÃO

SUMÁRIO

---



---



---



---

# 1 Introdução à programação

## 1.1 O que é programação

Programação é o processo de desenvolvimento de software, aplicativos, sistemas e outras soluções computacionais. É uma disciplina que utiliza linguagens de programação para escrever instruções que serão entendidas e executadas pelos computadores. Essas instruções definem como o computador deve realizar tarefas específicas, como manipular dados, realizar cálculos, interagir com o usuário, entre outros. A programação é uma habilidade fundamental para a era digital e é uma das principais ferramentas para a resolução de problemas complexos.

## 1.2 História da programação

A história da programação tem suas raízes na década de 1940, quando os primeiros computadores foram inventados. Naquela época, as pessoas programavam diretamente os computadores, inserindo fios e interruptores para realizar tarefas específicas.
No fim da década de 1940, a linguagem de programação Assembly foi introduzida, permitindo que os programadores escrevessem instruções em um formato mais acessível e fácil de entender.
Na década de 1950, surgiram outras linguagens de programação, como o Fortran (Formula Translation) e o COBOL (Common Business Oriented Language). Estas linguagens eram mais fáceis de usar e permitiam a criação de programas mais sofisticados.
Na década de 1960, o conceito de programação estruturada foi introduzido, permitindo que os programas fossem escritos de maneira mais organizada e fácil de entender.
Na década de 1970, surgiram as primeiras linguagens de programação orientadas a objetos, como o Smalltalk. Estas linguagens permitiam a criação de programas mais complexos, com a utilização de objetos e classes.
A partir daí, a evolução da programação continuou, com a introdução de novas linguagens, ferramentas e tecnologias, como o Java, o Python e o JavaScript, entre outros. Atualmente, a programação é uma das disciplinas mais importantes da tecnologia da informação e é fundamental para a criação de soluções computacionais avançadas.

### a. Conceitos básicos de programação.

**Variáveis**: um local na memória do computador onde um valor pode ser armazenado e recuperado.

**Tipos de dados**: os tipos de dados incluem números inteiros, números de ponto flutuante, caracteres e strings.

**Operadores**: são os símbolos usados para realizar operações matemáticas e lógicas, como adição, subtração, comparação, entre outras.

**Condicionais**: permitem que o programa tome decisões com base em certas condições, como se um número é par ou ímpar.

**Laços**: permitem que o programa execute uma série de instruções repetidamente, até que uma determinada condição seja atendida.

**Funções**: são blocos de código reutilizáveis que realizam uma tarefa específica.

**Arrays**: são estruturas de dados que armazenam uma coleção de valores.

**Objetos**: são estruturas de dados que representam entidades do mundo real, como um carro, uma pessoa, entre outros.

## 2. Lógica de programação

### 2.1 Estruturas de controle

As estruturas de controle são estruturas usadas para controlar o fluxo de execução de um programa. Algumas das estruturas de controle mais comuns incluem:

- **Condicionais**: permitem que o programa tome decisões com base em certas condições, como se um número é par ou ímpar.

Em Portugol, as estruturas de controle condicionais são:

✓ **Se**: usada para avaliar uma condição e executar uma série de instruções caso a condição seja verdadeira.
✓ **Senão se**: usada para avaliar múltiplas condições e executar a primeira série de instruções cuja condição seja verdadeira.
✓ **Senão**: usada para executar uma série de instruções caso nenhuma das condições avaliadas pelo se ou senão se sejam verdadeiras.

Aqui está um exemplo de uso dessas estruturas em Portugol:

```
int idade = 25;

se (idade >= 18) então
  // se a idade for maior ou igual a 18
  escreva("Você é maior de idade");
senão se (idade >= 12) então
  // se a idade for maior ou igual a 12 e menor que 18
  escreva("Você é adolescente");
senão
  // se a idade for menor que 12
  escreva("Você é criança");
fim_se
```

- **Laços**: permitem que o programa execute uma série de instruções repetidamente, até que uma determinada condição seja atendida.

Em Portugol, as estruturas de controle de laços são:

✓ **Para**: usada para repetir uma série de instruções um número fixo de vezes.
✓ **Enquanto**: usada para repetir uma série de instruções enquanto uma determinada condição seja verdadeira.

```
// Leitura do número
int numero;
escreva("Digite um número: ");
leia(numero);

// Laço com estrutura condicional
int soma = 0;
para (int i = 1; i <= numero; i = i + 1) faca
  se (i % 2 == 0) então
    soma = soma + i;
  fim_se
fim_para

// Exibição do resultado
escreva("A soma dos números pares até ", numero, " é ", soma);
```

Este programa solicita ao usuário que digite um número e, em seguida, realiza a soma dos números pares até o número informado, usando um laço com uma estrutura condicional dentro dele.

- **Saltos:** permitem que o programa pule para uma determinada posição no código, evitando a execução de instruções desnecessárias.

As principais estruturas de controle de saltos em Portugol são:

✓ **Pare**: usado para interromper a execução de um laço.
✓ Retorne: usado para retornar o controle para o chamador de uma função ou subrotina.

```
var
   numero: inteiro

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

   se (numero > 0) entao
      escreva("O número é positivo.")
      pare
   fimse

   se (numero < 0) entao
      escreva("O número é negativo.")
      pare
   fimse

   escreva("O número é zero.")

fimalgoritmo
```

Neste exemplo, usamos a estrutura de controle se para verificar se o número digitado pelo usuário é positivo, negativo ou zero. Se o número for positivo ou negativo, o programa exibe uma mensagem apropriada e para a execução com a instrução pare. Se nenhum dos dois primeiros se for verdadeiro, o programa exibirá a mensagem "O número é zero."

- Interrupções: permitem que o programa interrompa a execução de uma rotina e retorne ao ponto de chamada.

As estruturas de controle de interrupção são utilizadas para interromper o fluxo normal de execução de um programa e saltar para uma seção específica do código. Algumas das principais estruturas de controle de interrupção incluem:

- ✓ Instrução pare: interrompe a execução do programa imediatamente.
- ✓ Instrução sair: interrompe a execução de um laço e pula para a próxima instrução após o laço.
- ✓ Instrução continue: interrompe a execução atual de um laço e passa para a próxima iteração do laço.

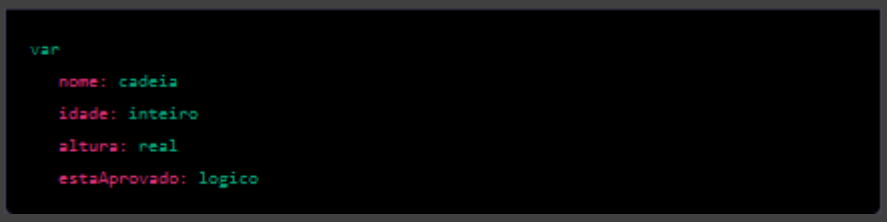
```
algoritmo "Exemplo de Estrutura de Controle de Interrupção com Sair"
var
   numero, soma: inteiro

inicio
   soma = 0
   para i de 1 ate 10 passo 1 faca
      escreva("Digite um número: ")
      leia(numero)

      se (numero < 0) entao
         escreva("Números negativos não são permitidos. Tente novamente.")
         continue
      fimse

      se (numero = 0) entao
         escreva("A soma foi interrompida.")
         sair
      fimse

      soma = soma + numero
   fimpara

   escreva("A soma dos números é: ", soma)

fimalgoritmo
```

Neste exemplo, usamos a estrutura de controle de interrupção ‘sair’ dentro de um laço ‘para’ para interromper a execução do laço se o usuário digitar zero. Se o usuário digitar um número negativo, o programa exibirá uma mensagem pedindo para o usuário tentar novamente e pulará para a próxima iteração do laço com a instrução ‘continue’. Ao final, o programa exibe a soma dos números digitados pelo usuário, excluindo quaisquer números negativos ou zeros.

## 2.2 Variáveis e constantes

As variáveis e constantes são usadas para armazenar valores em um programa.

Variáveis são espaços na memória do computador que podem conter valores que podem ser alterados durante a execução do programa. Em Portugol, para declarar uma variável, é necessário especificar o seu tipo (inteiro, real, lógico, etc.) e um nome. Por exemplo:




Neste exemplo, estamos declarando quatro variáveis: uma variável de cadeia de caracteres chamada ‘nome’, uma variável inteira chamada ‘idade’, uma variável real chamada ‘altura’ e uma variável lógica chamada ‘estaAprovado’. Todas as variáveis têm seus respectivos tipos especificados após o sinal de dois pontos (:). Depois de declaradas, essas variáveis podem ser usadas em outras partes do programa para armazenar valores.

Em Portugol, existem vários tipos de variáveis, incluindo:

- Inteiro (inteiro): Armazena números inteiros, como -1, 0, 1, 2, etc.
- Real (real): Armazena números de ponto flutuante, como -1.5, 0.0, 3.14, etc.
- Cadeia de caracteres (cadeia): Armazena strings, ou sequências de caracteres, como "Olá, mundo!"
- Lógico (logico): Armazena valores lógicos, verdadeiro (verdadeiro) ou falso (falso).
- Caractere (caractere): Armazena um único caractere, como 'A' ou 'B'.

Além disso, em algumas implementações de Portugol, é possível usar tipos de dados adicionais, como Arrays, registros e tipos enumerados. A escolha do tipo de variável a ser usado depende da natureza dos dados que precisam ser armazenados e do problema que você está tentando resolver.

Uma constante é um valor que não muda durante a execução de um programa. Em outras palavras, uma constante é um valor que é definido uma vez e nunca muda ao longo da vida do programa.

As constantes são úteis em várias situações, como:

- Quando precisamos usar um valor fixo várias vezes ao longo do programa, como o valor de Pi (π) ou a gravidade.

- Quando queremos garantir que um valor não seja alterado acidentalmente durante a execução do programa.

Em Portugol, as constantes são declaradas usando a palavra-chave constante seguida do tipo, nome e valor da constante. Por exemplo:

```
constante PI: real = 3.14159265
```

Neste exemplo, estamos declarando uma constante real chamada PI com o valor 3.14159265. Depois de ser definida, essa constante pode ser usada em todo o programa sem que o seu valor mude.

## 2.3 Operadores matemáticos e lógicos

Em Portugol, os operadores matemáticos são os mesmos utilizados na matemática:

- Adição (+)
- Subtração (-)
- Multiplicação (*)
- Divisão (/)
- Módulo (ou resto da divisão) (%)
- Potenciação (elevado a) (^)

Os operadores lógicos em Portugol são representados da seguinte forma:

- E (&&)
- Ou (||)
- Negação (!)

Também existem os operadores de comparação:

- Igual (==)
- Diferente (!=)
- Maior que (>)
- Menor que (<)
- Maior ou igual (>=)
- Menor ou igual (<=)

A ordem de precedência, também conhecida como ordem de operações, é uma convenção utilizada em matemática e programação para determinar a ordem na qual as operações devem ser realizadas em uma expressão. Essa ordem é definida pelos seguintes grupos:

1. Parênteses: operações dentro de parênteses são realizadas primeiro
2. Potenciação: operações de potenciação são realizadas em seguida
3. Multiplicação: divisão e módulo: essas operações são realizadas antes das operações de adição e subtração
4. Adição e subtração: essas operações são realizadas por último

Por exemplo, na expressão matemática 2 + 3 * 4, a multiplicação deve ser realizada primeiro de acordo com a ordem de precedência, resultando em 2 + 12 = 14. Se a expressão fosse (2 + 3) * 4, a adição dentro dos parênteses seria realizada primeiro, resultando em 5 * 4 = 20.

Em Portugol, a ordem de precedência dos operadores matemáticos é a mesma da matemática. Para modificar a ordem de precedência, parênteses podem ser utilizados para indicar quais operações devem ser realizadas primeiro.

## 3. Sintaxe básica de Portugol

### 3.1 Declaração de variáveis

Em Portugol, a sintaxe básica para declarar uma variável é a seguinte:

```
tipo nome_da_variavel
```

Onde tipo é o tipo de dados da variável, e nome_da_variavel é o nome que você deseja dar à variável.

Por exemplo, para declarar uma variável do tipo inteiro chamada idade, você pode escrever o seguinte código:

```
inteiro idade
```

Além disso, você também pode inicializar a variável ao declará-la, atribuindo um valor a ela. Por exemplo:

```
inteiro idade = 20
```

Isso irá declarar uma variável do tipo inteiro chamada idade e inicializá-la com o valor 20.

Também é possível declarar múltiplas variáveis do mesmo tipo em uma única linha, separando os nomes das variáveis por vírgulas. Por exemplo:

```
inteiro idade, peso, altura
```

Isso irá declarar três variáveis do tipo inteiro chamadas idade, peso e altura.

Outra forma de declarar variáveis é através de parâmetros de uma função ou procedimento. Por exemplo:

```
funcao calcularIdade(inteiro anoNascimento)

{
```

```
  inteiro idade = 2023 - anoNascimento

  escreva("Sua idade é: ", idade)

}
```

Neste exemplo, declaramos uma função chamada calcularIdade que recebe um parâmetro do tipo inteiro chamado anoNascimento. Dentro da função, declaramos uma variável do tipo inteiro chamada idade e inicializamos ela com o resultado de uma operação matemática. Em seguida, usamos a função escreva para exibir o valor da variável idade na tela.

Conclusão:

A sintaxe básica para declarar uma variável em Portugol consiste em especificar o tipo de dados da variável e o nome que você deseja dar a ela. É possível inicializar a variável ao declará-la e também é possível declarar múltiplas variáveis do mesmo tipo em uma única linha. Além disso, as variáveis também podem ser declaradas como parâmetros de funções ou procedimentos.

## 3.2 Comentários no código

Comentários são trechos de código que são ignorados pelo interpretador da linguagem e servem para explicar o que o código está fazendo ou para adicionar informações relevantes sobre o código. Em Portugol, os comentários são escritos com o uso do caractere // ou entre /* e */.

Comentários de linha única são escritos com //. Por exemplo:

```
// Este é um comentário de linha única
```

Comentários de múltiplas linhas são escritos entre /* e */. Por exemplo:

```
/*

Este é um comentário de

múltiplas linhas

*/
```

Comentários são muito úteis para explicar o que um trecho de código faz, ou para fornecer informações adicionais que possam ajudar outras pessoas que leiam o código. É uma boa prática de programação sempre incluir comentários claros e concisos em seu código.

## 3.3 Comandos de entrada e saída de dados

Em Portugol, os comandos de entrada e saída de dados são usados para permitir a interação do programa com o usuário. Os comandos básicos de entrada e saída em Portugol são leia e escreva, respectivamente.

O comando leia é usado para ler dados digitados pelo usuário a partir do teclado e atribuí-los a variáveis. Sua sintaxe básica é:

```
leia(nome_da_variavel)
```

Por exemplo, para ler um número inteiro digitado pelo usuário e armazená-lo na variável idade, podemos usar o seguinte comando:

```
leia(idade)
```

O comando escreva é usado para exibir valores na tela. Sua sintaxe básica é:

```
escreva(valor1, valor2, ...)
```

Por exemplo, para exibir uma mensagem na tela, podemos usar o seguinte comando:

```
escreva("Olá, mundo!")
```

Também é possível exibir valores de variáveis, usando o nome da variável como parâmetro do comando escreva. Por exemplo, para exibir o valor da variável idade, podemos usar o seguinte comando:

```
escreva(idade)
```

É importante lembrar que o comando escreva pode receber um ou mais valores como parâmetro, que serão exibidos na tela separados por um espaço em branco. Para exibir valores separados por outros caracteres, é possível concatenar strings usando o operador +. Por exemplo:

python

```
escreva("Meu nome é " + nome + " e tenho " + idade + " anos.")
```

Este comando exibirá uma mensagem contendo o valor das variáveis nome e idade, separadas por um texto fixo e por outros caracteres.

Conclusão:

Em Portugol, os comandos de entrada e saída de dados são leia e escreva, respectivamente. O comando leia é usado para ler dados digitados pelo usuário e atribuí-los a variáveis, enquanto o comando escreva é usado para exibir valores na tela. É

importante lembrar que o comando escreva pode receber um ou mais valores como parâmetro e que é possível concatenar strings para exibir valores separados por outros caracteres.