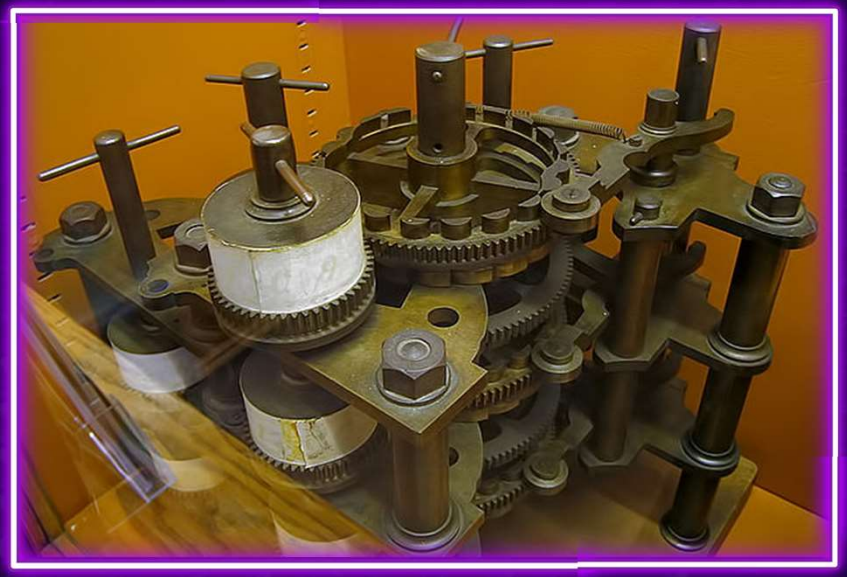
LINGUAGENS DE PROGRAMAÇÃO

O QUE É NECESSÁRIO PARA PROGRAMAR:

Para programar, é necessário ter ferramentas de “desenvolvimento”. Como, por exemplo, computadores, sistemas operacionais, pacotes, softwares.
Antigamente, usavam-se forma de se programar em que não necessariamente precisasse de “código”, porém, no que se entende de “linguagem” necessita desse ambiente de programação.
Algumas linguagens, não necessitam de ambientes “específicos” para serem “desenvolvidas”, por exemplo, JavaScript, aonde um navegador é usado como ambiente. No entanto, algumas linguagens necessitam de um “ambiente” de desenvolvimento. Ao fim, sempre é necessário uma ferramenta para programar e um “ambiente” para rodar a “programação”.



EVOLUÇÃO DAS LINGUAGENS DE PROGRAMAÇÃO

As primeiras máquinas de calcular:
Tudo começou com as primeiras máquinas de calcular, como a máquina analítica desenvolvida por Charles Babbage no século XIX. Essas máquinas utilizavam mecanismos físicos para realizar cálculos complexos, sem a necessidade de uma linguagem de programação formal. No entanto, com o avanço da eletrônica e o surgimento dos primeiros computadores eletrônicos na década de 1940, foi necessário desenvolver uma forma de instruir essas máquinas a realizar tarefas específicas.

```
; Hello World for Intel Assembler (MSDOS)

mov ax,cs
mov ds,ax
mov ah,9
mov dx, offset Hello
int 21h
xor ax,ax
int 21h

Hello:
  db "Hello World!",13,10,"$"
```

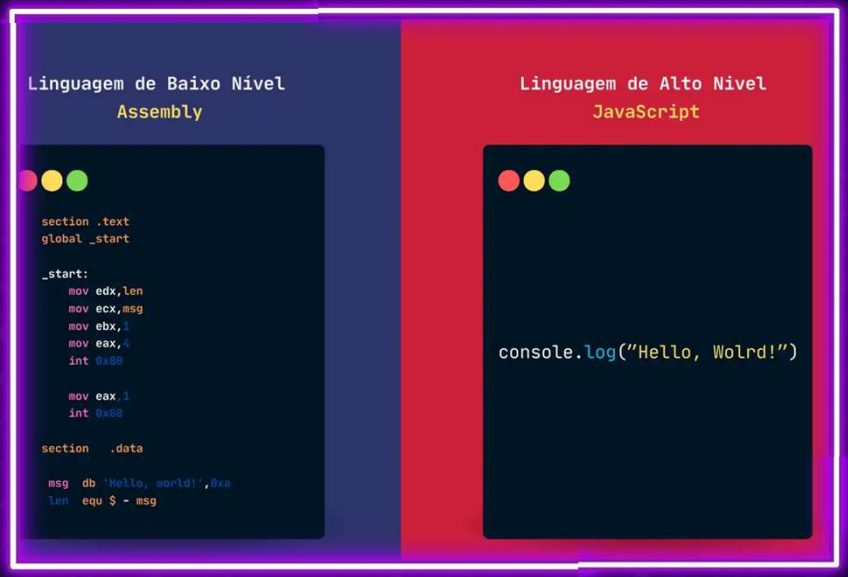
EVOLUÇÃO DAS LINGUAGENS DE PROGRAMAÇÃO

Linguagens de programação de baixo nível:
Foi assim que surgiram as linguagens de programação de baixo nível, como o Assembly. Essas linguagens permitiam que os programadores se comunicassem diretamente com o hardware do computador, escrevendo instruções em um código que representava as operações a serem realizadas. Embora o Assembly fosse eficiente em termos de desempenho, ele era extremamente complexo e exigia um conhecimento profundo da arquitetura do computador.

```
program cilindro
! Calcula a área de um cilindro.
!
! Declara as variáveis e constantes.
implicit none ! Requer que todas as variáveis sejam declaradas
integer :: ierr
real :: raio,altura,area
real , parameter :: pi = 3.14159
do
   ! Pergunta ao usuário o raio e a altura e lê os valores.
   write (*,*) "Entre com o raio e a altura, 'q' para sair."
   read (*,*,iostat=ierr) raio,altura
   !
   ! Se o raio e a altura não puderam ser lidos da entrada, termina o programa.
   if (ierr /= 0) stop "finalizando o programa"
   !
   ! Calcula a área. O sinal ** significa "eleva a uma potência".
   area = 2*pi*(raio**2 + raio*altura)
   !
   ! Escreve as variáveis de entrada (raio, altura) e a saida (área) na tela.
   write (*,"(1x,'raio=',f6.2,5x,'altura=',f6.2,5x,'area=',f6.2)") raio,altura,are
end do
end program cilindro
```

EVOLUÇÃO DAS LINGUAGENS DE PROGRAMAÇÃO

Linguagens de programação de alto nível:
Com o passar do tempo, surgiram as linguagens de programação de alto nível, que buscavam simplificar o processo de programação e torná-lo mais acessível. Uma das primeiras linguagens de alto nível a ganhar popularidade foi o FORTRAN, desenvolvido na década de 1950. Com o FORTRAN, os programadores podiam escrever instruções em uma linguagem mais próxima da linguagem humana, o que facilitou muito o desenvolvimento de programas complexos.



ALTO NÍVEL VS BAIXO NÍVEL

Alto nível:

- Abstração do Hardware
- Sintaxe Simples e Legível
- Portabilidade
- Riqueza de Bibliotecas e Frameworks
- Facilidade de Depuração e Manutenção
- Suporte a Paradigmas de Programação
- Segurança e Robustez

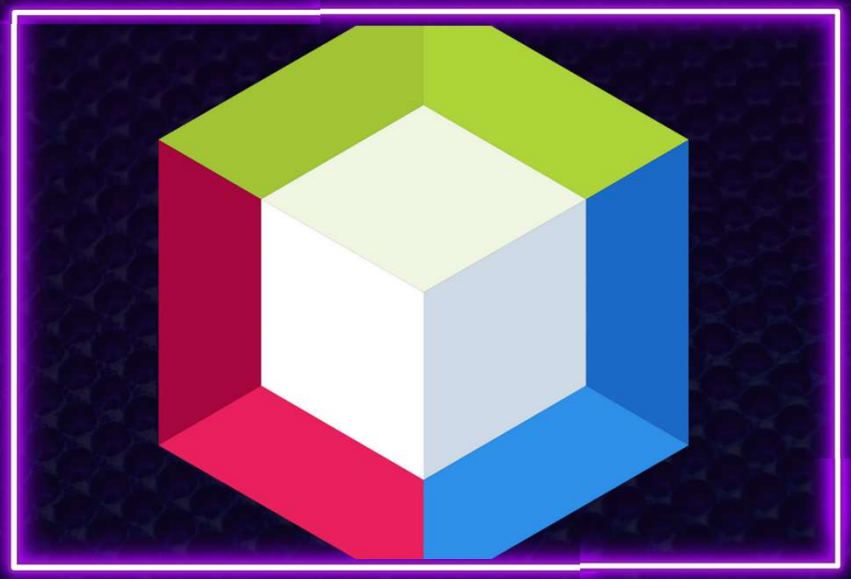
Baixo nível:

- Proximidade com o Hardware
- Sintaxe Complexa e Detalhada
- Alto Desempenho
- Dependência de Plataforma
- Uso em Sistemas Críticos
- Difícil Depuração e Manutenção

obs: O conceito de abstração consiste em esconder os detalhes de algo, no caso, os detalhes desnecessários.

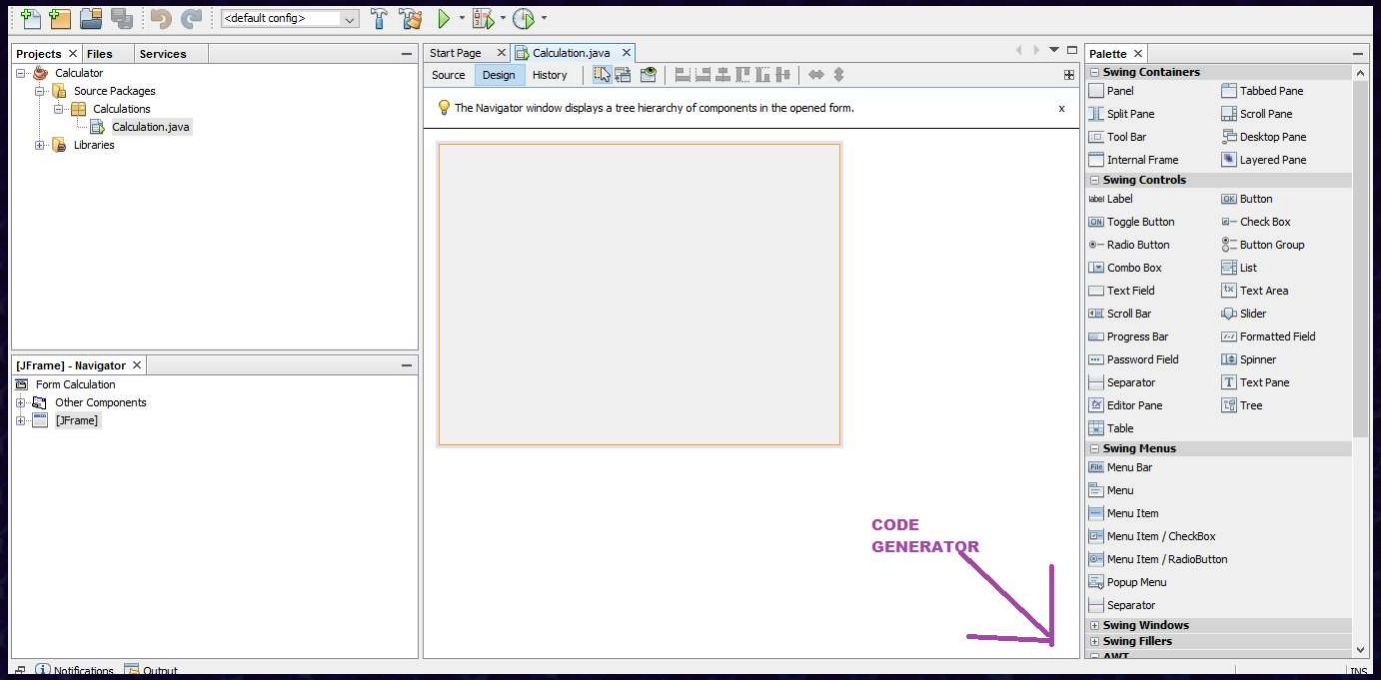
OS TIPOS DE PROGRAMAS:

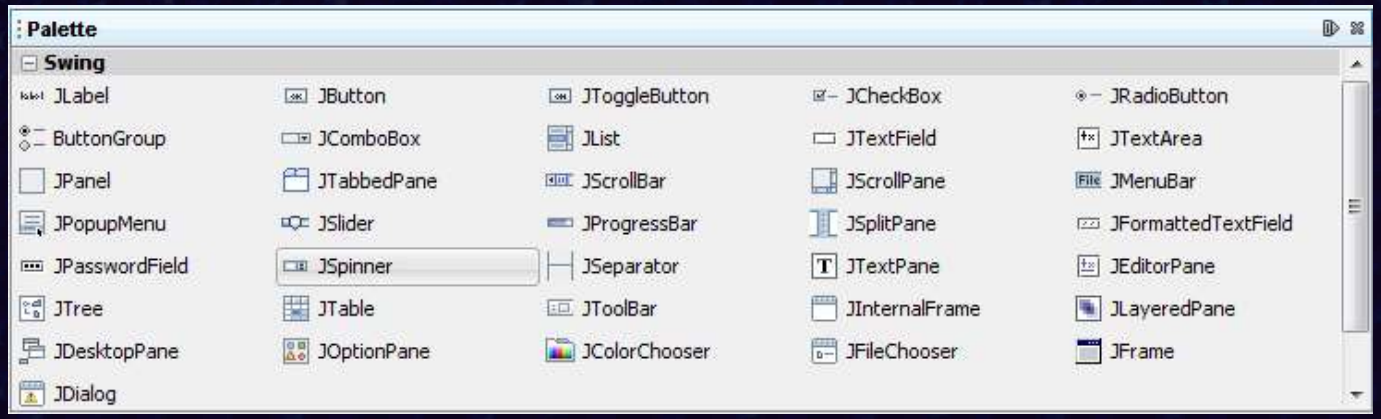
Algumas linguagens de programação (grande parte) necessitam de um ambiente de desenvolvimento (Ambiente de desenvolvimento integrado, conhecido como IDE). Um IDE é uma aplicação que fornece um conjunto. abrangente de ferramentas para facilitar o desenvolvimento de software. O objetivo principal de um IDE é aumentar a produtividade do programador, fornecendo uma interface unificada e integrada para todas as tarefas de desenvolvimento.

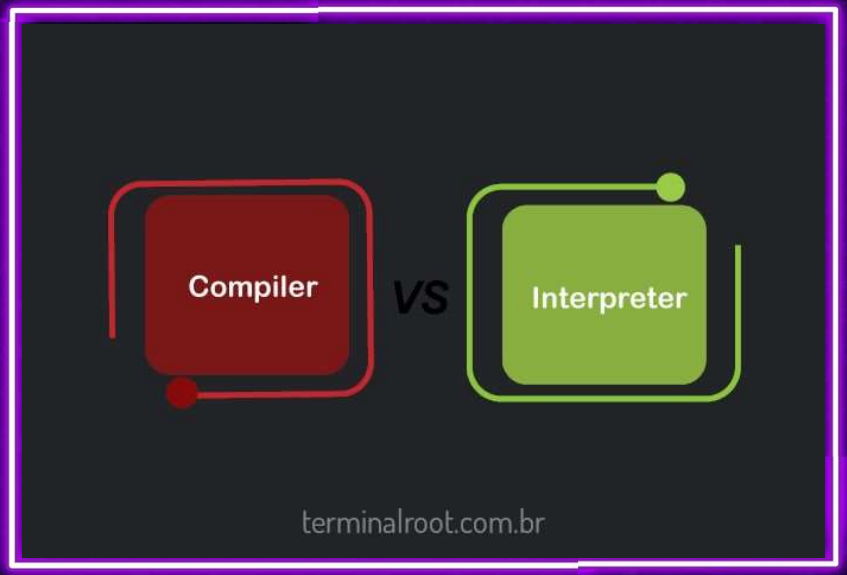


IDE:

- Editor de Código
- Depurador (Debugger)
- Compilador/Interpretador
- Gerenciamento de Projetos
- Controle de Versão Integrado
- Ferramentas de Construção (Build Tools)
- Emuladores e Simuladores


Projects
Files
Services
Calculator
Source Packages
Calculations
Calculation.java
Libraries
[JFrame] - Navigator
Form Calculation
Other Components
[JFrame]
Start Page
Calculation.java
Source
Design
History
The Navigator window displays a tree hierarchy of components in the opened form.
CODE
GENERATOR
Palette
Swing Containers
Panel
Split Pane
Tool Bar
Internal Frame
Tabbed Pane
Scroll Pane
Desktop Pane
Layered Pane
Swing Controls
Label
Toggle Button
Radio Button
Combo Box
Text Field
Scroll Bar
Progress Bar
Password Field
Separator
Editor Pane
Table
Button
Check Box
Button Group
List
Text Area
Slider
Formatted Field
Spinner
Text Pane
Tree
Swing Menus
Menu Bar
Menu
Menu Item
Menu Item / CheckBox
Menu Item / RadioButton
Popup Menu
Separator
Swing Windows
Swing Fillers
Notifications
Output


Palette
Swing
JLabel
JButton
JToggleButton
JCheckBox
JRadioButton
ButtonGroup
JComboBox
JList
JTextField
JTextArea
JPanel
JTabbedPane
JScrollBar
JScrollPane
JMenuBar
JPopupMenu
JSlider
JProgressBar
JSplitPane
JFormattedTextField
JPasswordField
JSpinner
JSeparator
JTextPane
JEditorPane
JTree
JTable
JToolBar
JInternalFrame
JLayeredPane
JDesktopPane
JOptionPane
JColorChooser
JFileChooser
JFrame
JDialog



COMPILAR VS INTERPRETAR

É preciso ter em mente a diferença entre compilação e interpretação, já que as linguagens operam de diferente formas: Pode-se dizer que compiladores “compilam” um código(lendo todo o código e transformando espécie de código objeto que contém instruções para a execução no microprocessador.) e um interpretador “interpreta”(Não tem o processo de traduzir todo o programa em um arquivo para conseguir rodar, e sim, ele inclui cada instrução da sua linguagem de alto nível no código de máquina.) cada linha do código.

COMPILADORES

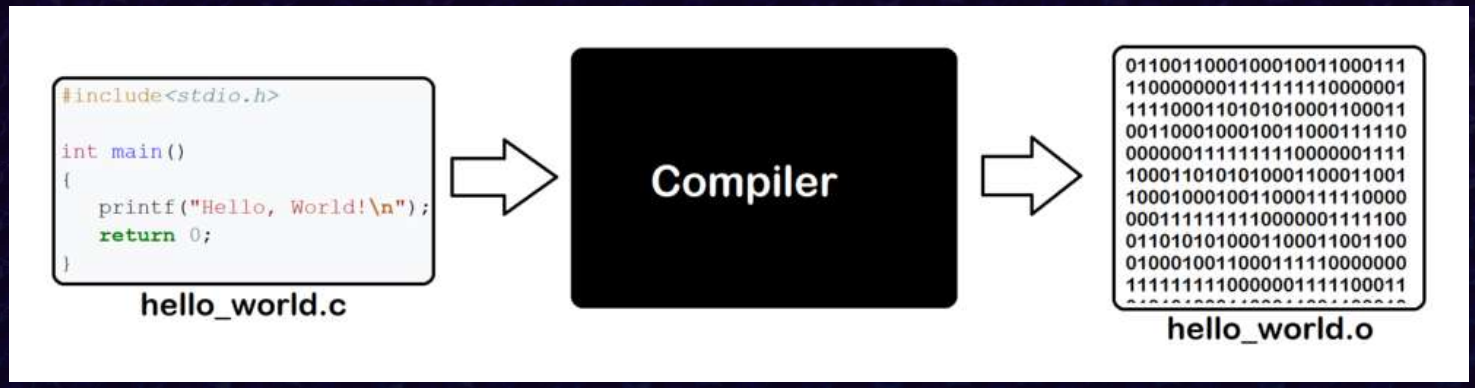
Um compilador é um programa que traduz o código escrito em uma linguagem de alto nível para código de máquina. Em resumo, traduz o código legível para os seres humanos e converte para a linguagem binária que o processador entende. Além disso, o compilador deve obedecer as regras de sintaxe da linguagem, no caso, se ocorrer algum erro você deve corrigi-los.

Vantagens dos compiladores:

- o código já é traduzido para a linguagem de máquina, com isso, se obtém um tempo menor de execução;
- a confiabilidade do código final, já que os processos do compilador são bem severos e conseguem validar e otimizar muito bem o código.

Desvantagens dos compiladores:

- você não pode mudar o código sem ter que recompilar todo o programa;
- mesmo com diversas otimizações, o compilador acaba sofrendo com o seu tempo de finalização do código;


#include<stdio.h>
int main()
{
printf("Hello, World!\n");
return 0;
}
hello_world.c
Compiler
0110011000100010011000111
1100000001111111110000001
1111000110101010001100011
0011000100010011000111110
0000001111111110000001111
1000110101010001100011001
1000100010011000111110000
0001111111110000001111100
0110101010001100011001100
0100010011000111110000000
1111111110000001111100011
hello_world.o

INTERPRETADORES

O interpretador também é um programa, porém, ele não tem o processo de traduzir todo o programa em um arquivo para conseguir rodar, e sim, ele inclui cada instrução da sua linguagem de alto nível no código de máquina. Com isso, o papel de tradução é feito de uma maneira muito mais dinâmica e que consegue incluir alguns tipos de código, tais como, scripts, códigos pré-compilados e o próprio código fonte.

Vantagens dos interpretadores:

- são fáceis de usar, especialmente para iniciantes;
- a execução é linha por linha, sendo assim fica mais fácil de encontrar o erro;
- nenhum código intermediário, por tanto, utiliza a memória de maneira mais inteligente.

Desvantagens dos interpretadores:

- o tempo de execução é muito grande, devido a forma de que se é traduzido;
- para ser executado deve-se ter um programa correspondente ao interpretador;
- menos seguro.

Figure: Interpreter

```
ss Pessoa

 private String nome;
 private int idade;

 public boolean equals(Object o)
 {
     if (this == o)
          return true;
     else if (o == null || getClass() != o.getClass())
 return false;
     else
     {
         Pessoa p = (Pessoa) o;
         return nome.equals(p.nome) && idade == p.idade
     }
 }
```

TIPAGEM

Tipagem Forte:

Em uma linguagem com tipagem forte, os tipos de dados são rigidamente aplicados e a conversão automática entre tipos incompatíveis é limitada.
Isso significa que, em uma operação, você precisa garantir que os tipos de dados sejam compatíveis. Por exemplo, se você tentar somar uma string com um número, uma linguagem com tipagem forte geralmente gerará um erro.
Exemplos de linguagens com tipagem forte incluem Java, C#, Pye Haskell.

```
let numero = 5

console.log(numero);

numero = "Alisson Chagas"

console.log(numero)
```

TIPAGEM

Tipagem Fraca:

Em contraste, em uma linguagem com tipagem fraca, a conversão automática entre tipos é mais flexível. A linguagem pode tentar converter automaticamente tipos de dados diferentes para permitir operações.
Isso pode levar a comportamentos imprevisíveis, especialmente para programadores inexperientes, já que operações inesperadas podem ocorrer silenciosamente.
Exemplos de linguagens com tipagem fraca incluem JavaScript, PHP e Python (em algumas situações, como operações entre diferentes tipos de dados).

C

- Área de Foco: Desenvolvimento de sistemas operacionais, software de sistema, drivers de dispositivos, sistemas embarcados.
- Tipo de execução: C é uma linguagem compilada.
- Tipagem: C é uma linguagem fortemente tipada.
- Nível da linguagem: C é considerada uma linguagem de baixo nível, oferecendo controle direto sobre o hardware do computador.
- Particularidades: C é conhecida por sua eficiência e portabilidade, sendo amplamente utilizada para desenvolvimento de sistemas de software de alto desempenho.

COBOL

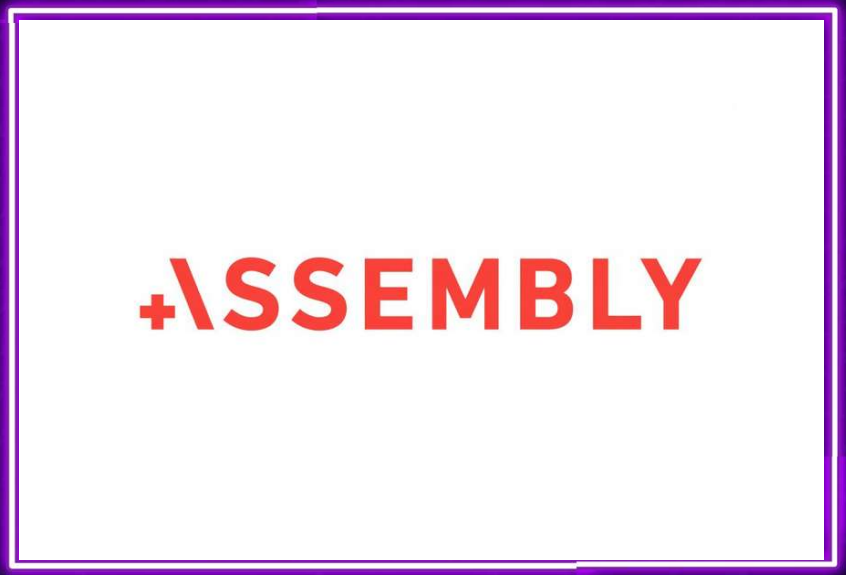
COBOL (Common Business Oriented Language):

- Área de Foco: Aplicações empresariais, especialmente em sistemas de processamento de dados de grande escala.
- Tipo de execução: COBOL é geralmente compilada.
- Tipagem: COBOL é uma linguagem fortemente tipada.
- Nível de linguagem: COBOL é considerada uma linguagem de alto nível.
- Particularidades: COBOL é amplamente utilizado em sistemas de mainframe e em setores como finanças, governo e saúde, devido à sua robustez e capacidade de processar grandes volumes de dados.

FORTRAN

Fortran (Formula Translation):

- Área de Foco: Computação científica, matemática, engenharia, física.
- Tipo de execução: Fortran é geralmente compilada.
- Tipagem: Fortran pode ser considerada uma linguagem fortemente tipada.
- Nível de linguagem: Fortran é uma linguagem de alto nível, especialmente projetada para cálculos numéricos e científicos.
- Particularidades: Fortran é uma das linguagens mais antigas ainda em uso, conhecida por sua eficiência em cálculos numéricos e modelagem matemática.



ASSEMBLY

Assembly:

- Área de Foco: Programação de baixo nível, controle direto do hardware do computador.
- Tipo de Compilação: Assembly é uma linguagem de baixo nível e é geralmente montada.
- Tipagem: Assembly pode ser considerada uma linguagem fracamente tipada.
- Nível: Assembly é uma linguagem de muito baixo nível, próxima da linguagem de máquina.
- Particularidades: O código Assembly é altamente específico para a arquitetura do processador e é usado quando é necessário um controle preciso do hardware.

BASIC

PROGRAMMIN
G LANGUAGE

BASIC

BASIC (Beginner's All-purpose Symbolic Instruction Code):

- Área de Foco: Educação, desenvolvimento rápido de protótipos, aplicativos simples.
- Tipo de Compilação: BASIC pode ser tanto interpretada quanto compilada, dependendo da implementação.
- Tipagem: A tipagem em BASIC pode variar dependendo da versão e implementação, mas geralmente é considerada uma linguagem fracamente tipada.
- Nível: BASIC é considerada uma linguagem de alto nível.
- Particularidades: BASIC é conhecida por sua simplicidade e facilidade de aprendizado, sendo frequentemente usada como uma linguagem introdutória para programação.

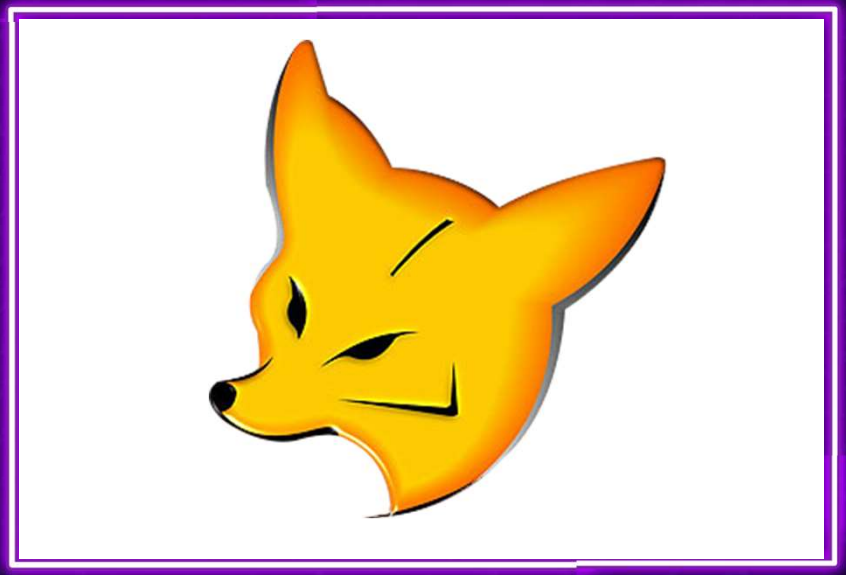
CLIPPER

Clipper:

- Área de Foco: Desenvolvimento de aplicativos de banco de dados e sistemas de gestão empresarial.
- Tipo de Compilação: Clipper é uma linguagem compilada.
- Tipagem: Clipper é considerada uma linguagem fracamente tipada.
- Nível: Clipper é considerada uma linguagem de alto nível.
- Particularidades: Clipper foi amplamente utilizado nos anos 80 e 90 para o desenvolvimento de sistemas de banco de dados baseados em arquivos DBF.

PASCAL

- Área de Foco: Ensino de programação, desenvolvimento de software educacional, aplicações científicas.
- Tipo de execução: Pascal é uma linguagem compilada.
- Tipagem: Pascal é geralmente considerada uma linguagem fortemente tipada.
- Nível da linguagem: Pascal é considerada uma linguagem de alto nível.
- Particularidades: Pascal foi projetada para incentivar a boa prática de programação e é conhecida por sua clareza e estruturação.

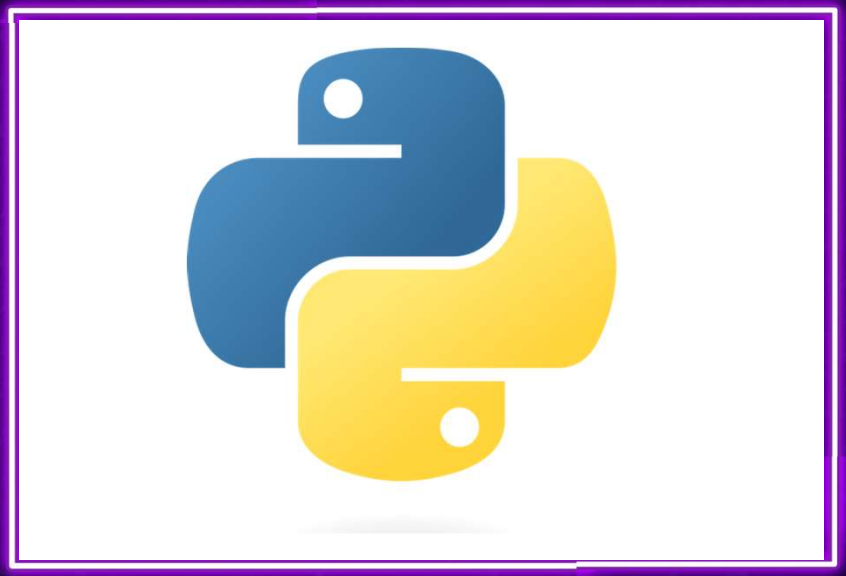


FOXPRO

- Área de Foco: Desenvolvimento de aplicativos de banco de dados e sistemas de gestão empresarial.
- Tipo de execução: FoxPro é uma linguagem interpretada.
- Tipagem: FoxPro é considerada uma linguagem fracamente tipada.
- Nível da linguagem: FoxPro é considerada uma linguagem de alto nível.
- Particularidades: FoxPro foi popular para o desenvolvimento de aplicativos de banco de dados baseados em DBF, mas foi descontinuado pela Microsoft em favor de tecnologias mais modernas.

VISUAL BASIC

- Área de Foco: Desenvolvimento de aplicativos de desktop, aplicações web, automação de escritório.
- Tipo de execução: Visual Basic é uma linguagem compilada.
- Tipagem: Visual Basic é considerada uma linguagem fracamente tipada.
- Nível da linguagem: Visual Basic é considerada uma linguagem de alto nível.
- Particularidades: Visual Basic foi amplamente utilizado para desenvolvimento rápido de aplicativos de desktop baseados no ambiente de desenvolvimento da Microsoft, mas foi substituído pelo Visual Basic .NET.



PYTHON

- Área de Foco: Desenvolvimento web, ciência de dados, automação, inteligência artificial, aprendizado de máquina.
- Tipo de execução: Python é uma linguagem interpretada.
- Tipagem: Python é dinamicamente tipada, o que significa que as variáveis não precisam ser declaradas com um tipo específico.
- Nível da linguagem: Python é considerada uma linguagem de alto nível.
- Particularidades: Python é conhecida por sua sintaxe simples e legível, bem como pela vasta gama de bibliotecas disponíveis para diversas aplicações.

JAVA

- Área de Foco: Desenvolvimento de aplicativos corporativos, aplicações web, aplicativos móveis (Android), sistemas embarcados.
- Tipo de execução: Java é compilada para bytecode e interpretada pela JVM (Java Virtual Machine).
- Tipagem: Java é fortemente tipada.
- Nível da linguagem : Java é considerada uma linguagem de alto nível.
- Particularidades: Java é conhecida por sua portabilidade, segurança e robustez, sendo uma das linguagens mais populares para o desenvolvimento de aplicativos corporativos.

JAVASCRIPT

- Área de Foco: Desenvolvimento web (front-end e back-end), aplicações web interativas.
- Tipo de execução: JavaScript é uma linguagem interpretada.
- Tipagem: JavaScript é uma linguagem fracamente tipada, permitindo flexibilidade no uso de variáveis.
- Nível da linguagem: JavaScript é considerada uma linguagem de alto nível.
- Particularidades: JavaScript é a linguagem de programação padrão da web, sendo amplamente utilizada para criar interatividade em páginas da web, desenvolvimento de aplicativos web e servidores web (Node.js). Ela suporta programação assíncrona e é conhecida por sua integração fácil com HTML e CSS, permitindo a criação de interfaces de usuário dinâmicas e responsivas.

Tipos de Dados

Tipo	Descrição		
Byte	1 byte	-128 a 127	Números inteiros
Short	2 bytes	-32768 a 32767	Números inteiros
int	4 bytes	-2147483648 a 2147483647	Números inteiros
Long	8 bytes	-922337203685477808 a 922337203685477807	Números inteiros
float	4 bytes	Aproximadamente 3.40282347E+38	Números reais
double	8 bytes		
Char	16 bits	0 a 256	Caracteres
boolean	Possuem valores true e false		Boolean

VARIÁVEIS

Variáveis são elementos fundamentais em programação que servem para armazenar e manipular dados. Elas atuam como espaços de memória nomeados que podem conter valores de diferentes tipos, como números, texto, booleanos, entre outros. As variáveis são essenciais para que os programas possam armazenar informações temporárias ou persistentes e realizar operações sobre esses dados.

```
public static void main(String[] args) {
    int num0;

    int num1 = 2;

    float num2 = 3;

    double num3 = 4;

    String nome = "Artigo Blog";
```

VARIÁVEIS

Ao criar uma variável, é necessário especificar um nome para ela, que servirá como uma referência para acessar seu conteúdo. Além disso, é comum declarar o tipo de dado que a variável irá armazenar, embora em algumas linguagens de programação, como JavaScript e Python, isso não seja estritamente necessário, devido à tipagem dinâmica.

As variáveis podem ser inicializadas com um valor específico no momento da criação ou podem ser atribuídas posteriormente ao longo da execução do programa. Elas podem ser atualizadas com novos valores e seu conteúdo pode ser utilizado em cálculos, comparações, controle de fluxo e muitas outras operações dentro do programa.

```
public static void main(String[] args) {
    int num0;

    int num1 = 2;

    float num2 = 3;

    double num3 = 4;

    String nome = "Artigo Blog";
```

VARIÁVEIS

Tipos de dados:

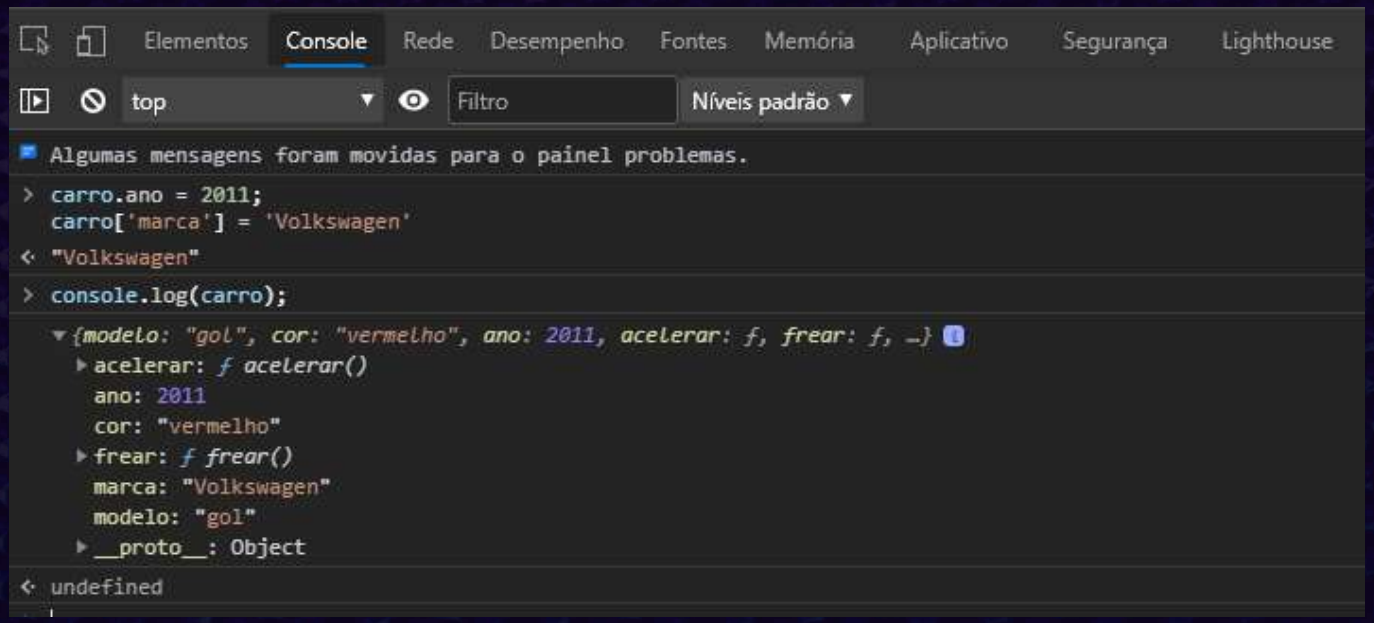
- Inteiro (Integer): Representa números inteiros, positivos ou negativos, sem casas decimais. Exemplos incluem -10, 0, 42.
- Ponto Flutuante (Float/Double): Representa números reais (números com casas decimais). Exemplos incluem 3.14, -0.5, 2.71828.
- Caractere (Character/Char): Representa um único caractere alfanumérico. Exemplos incluem 'a', 'B', '7'.
- String: Representa uma sequência de caracteres. Pode conter letras, números, símbolos e espaços. Exemplos incluem "Olá, mundo!", "12345", "python".

```
let arrayUm = [6, 5, 4, 3, 2, 1, 0];

arrayUm.join('');
// '6543210'

arrayUm.join('/');
// '6/5/4/3/2/1/0'
```

VARIÁVEIS

Tipos de dados:

- Array: Coleção ordenada de elementos do mesmo tipo de dados, acessados por um índice numérico. Exemplos incluem [1, 2, 3, 4, 5], ["maçã", "banana", "laranja"].
- Nulo (Null): Representa a ausência de valor. Pode ser usado para indicar que uma variável não tem valor atribuído.
- Objeto: Em linguagens orientadas a objetos, um objeto é uma instância de uma classe que contém dados (atributos) e métodos (funções). Exemplos incluem objetos de classes como "Carro", "Pessoa", "Conta Bancária".


Elementos
Console
Rede
Desempenho
Fontes
Memória
Aplicativo
Segurança
Lighthouse
top
Filtro
Níveis padrão
Algumas mensagens foram movidas para o painel problemas.
carro.ano = 2011;
carro['marca'] = 'Volkswagen'
"Volkswagen"
console.log(carro);
{modelo: "gol", cor: "vermelho", ano: 2011, acelerar: ƒ, frear: ƒ, …}
acelerar: ƒ acelerar()
ano: 2011
cor: "vermelho"
frear: ƒ frear()
marca: "Volkswagen"
modelo: "gol"
__proto__: Object
undefined

-	Subtração
+	Adição
*	Multiplicação
/	Divisão
%	Resto da divisão (módulo)

OPERADORES

Operadores Aritméticos:

São usados para realizar operações matemáticas básicas, como adição, subtração, multiplicação, divisão e resto da divisão (módulo).

Exemplos: + (adição), - (subtração), * (multiplicação), / (divisão), % (módulo).

Operadores de Atribuição:

São usados para atribuir valores a variáveis.

Exemplos: = (atribuição simples), += (atribuição com adição), -= (atribuição com subtração), *= (atribuição com multiplicação), /= (atribuição com divisão), %= (atribuição com módulo).

Operadores de Atribuição

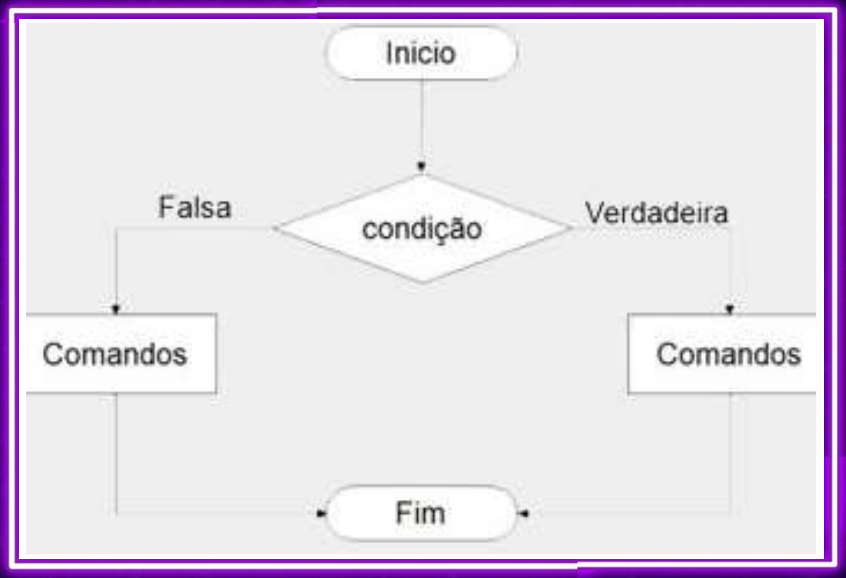
Operador	Descrição
=	Atribuição Simples
+=	Acréscimo
-=	Decréscimo
*=	Auto-multiplicação
/=	Auto-divisão
%=	Auto-módulo

Operador	Descrição
==	Igual a
===	Mesmo valor e tipo
!=	Diferente
!==	Diferente em valor e tipo
<	Menor que
>	Maior que
<=	Menor ou igual a
>=	Maior ou igual a

OPERADORES

Operadores de Comparação (Relacionais):
São usados para comparar valores e retornar um resultado verdadeiro ou falso (booleano).
Exemplos: == (igual a), != (diferente de), < (menor que), > (maior que), <= (menor ou igual a), >= (maior ou igual a).

Operadores Lógicos:
São usados para realizar operações lógicas entre valores booleanos.
Exemplos: && (e lógico), || (ou lógico), ! (não lógico).

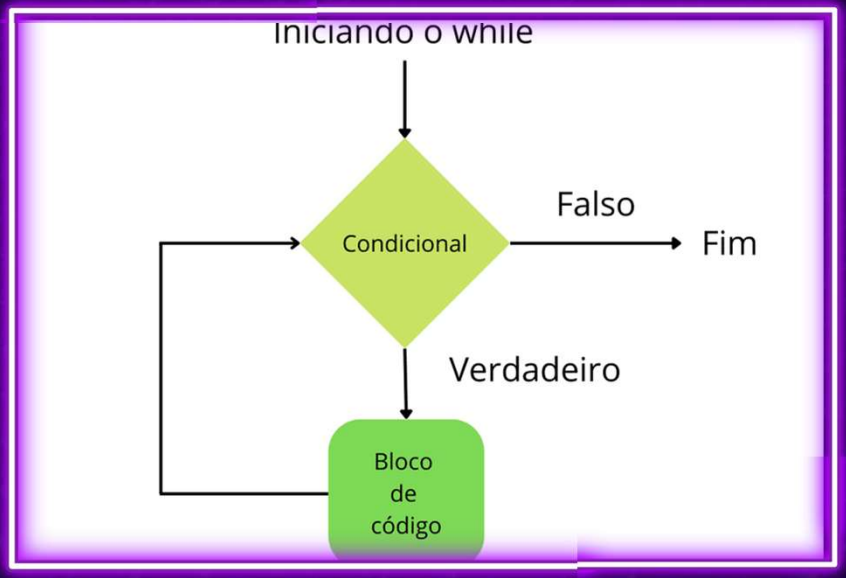


ESTRUTURAS DE CONTROLE

Estruturas de Decisão (Condicional):

If-else: Permite executar um bloco de código se uma condição for verdadeira e outro bloco de código se a condição for falsa.

Switch-case: Permite avaliar uma expressão e executar diferentes blocos de código com base nos valores resultantes.



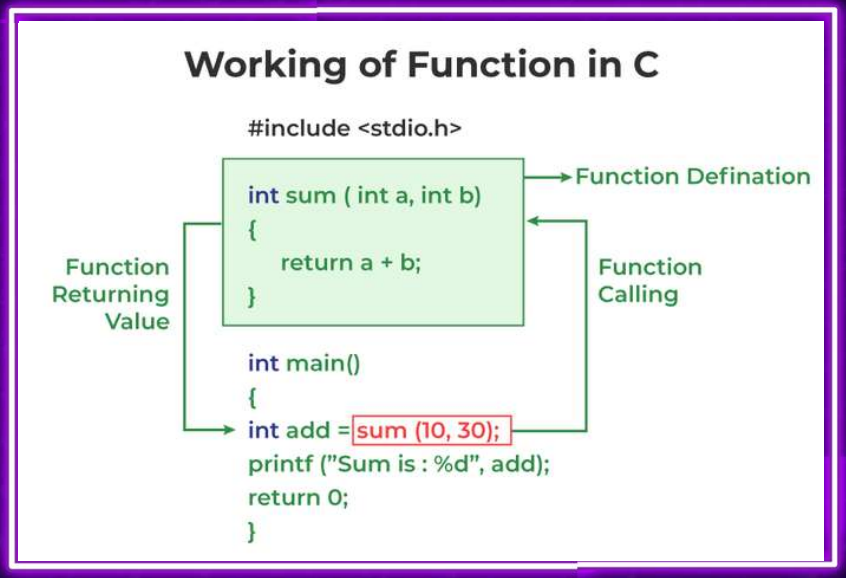
ESTRUTURAS DE CONTROLE

Estruturas de Repetição (Loop):

For: Executa um bloco de código um número específico de vezes, geralmente com um contador.

While: Executa um bloco de código repetidamente enquanto uma condição específica for verdadeira.

Do-While: Similar ao while, mas garante que o bloco de código seja executado pelo menos uma vez antes de verificar a condição de parada.

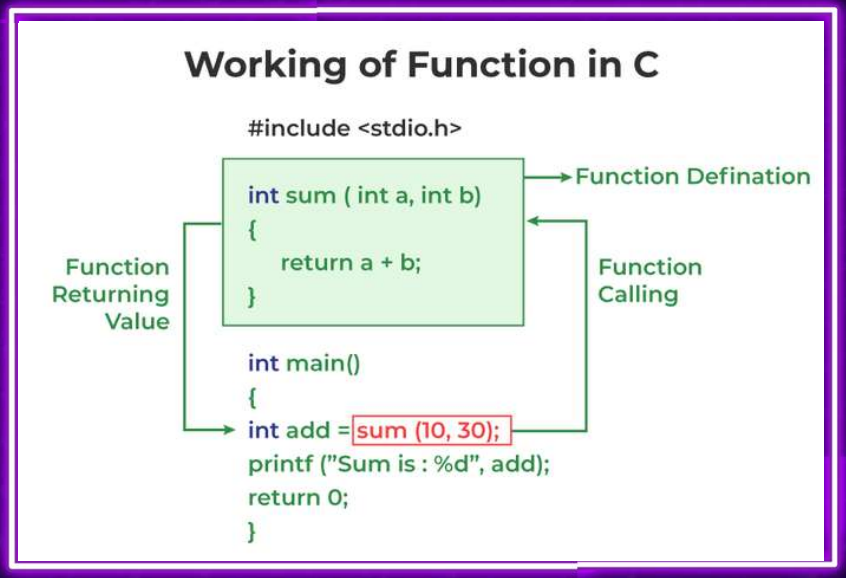


FUNÇÕES

Funções são blocos de código que realizam uma tarefa específica e podem ser reutilizadas em diferentes partes de um programa. Elas ajudam a tornar o código mais modular, legível e fácil de manter.

Características das Funções

- Nome: Cada função tem um nome único que identifica o que ela faz.
- Parâmetros: Funções podem receber valores de entrada chamados parâmetros.
- Corpo da Função: O conjunto de instruções que define o que a função faz.
- Valor de Retorno: O resultado que a função devolve após a execução.



MÓDULOS

Módulos são arquivos que contêm definições de funções, classes e variáveis que podem ser reutilizadas em outros arquivos. Eles permitem organizar o código de forma lógica, separando diferentes funcionalidades em diferentes arquivos.

Características dos Módulos

- Importação: Para usar um módulo em um programa, ele deve ser importado.
- Encapsulamento: Módulos encapsulam funcionalidades específicas, tornando o código mais organizado e modular.
- Namespace: Cada módulo tem seu próprio namespace, evitando conflitos de nomes.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Programação Orientada a Objetos (POO) é um paradigma de programação que organiza o design do software em torno de objetos, em vez de funções e lógica. Um objeto pode ser considerado como uma instância de uma classe, que é uma estrutura que define atributos (dados) e métodos (comportamentos) que os objetos daquela classe podem ter.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classe:

Uma classe é um "molde" ou "template" que define as propriedades (atributos) e os comportamentos (métodos) que os objetos criados a partir dessa classe terão.

Objeto:

Um objeto é uma instância de uma classe. Ele representa uma entidade concreta que pode possuir atributos específicos e comportamentos definidos pela classe.

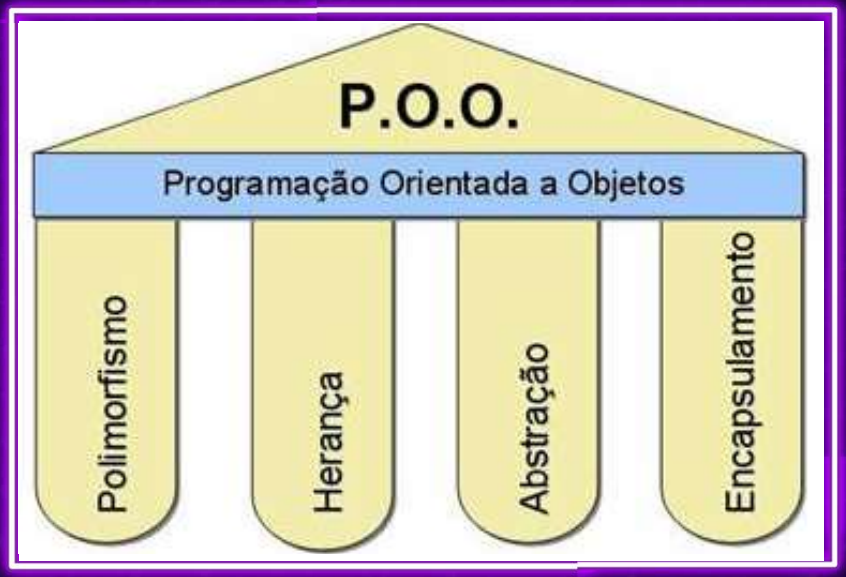
PROGRAMAÇÃO ORIENTADA A OBJETOS

Encapsulamento:

O encapsulamento é o princípio de esconder os detalhes internos de um objeto, expondo apenas o necessário. Isso é feito através de modificadores de acesso como private, protected e public (em linguagens que suportam esses modificadores explicitamente).

Herança:

A herança permite que uma classe (subclasse) herde atributos e métodos de outra classe (superclasse), promovendo a reutilização de código.



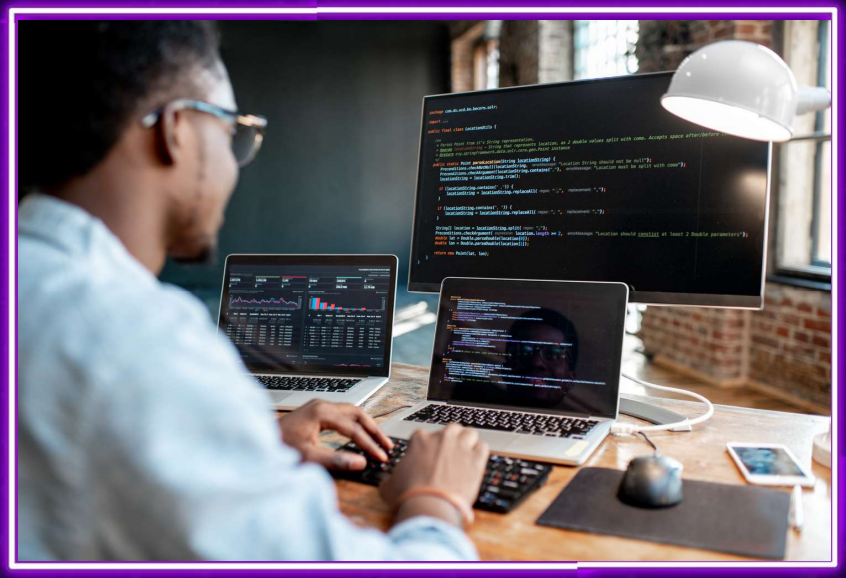
PROGRAMAÇÃO ORIENTADA A OBJETOS

Polimorfismo:

Polimorfismo permite que objetos de diferentes classes sejam tratados como objetos de uma classe comum. Em outras palavras, é a capacidade de diferentes classes implementarem métodos com o mesmo nome, mas comportamentos distintos.

Abstração:

Abstração é o conceito de simplificar sistemas complexos, focando nos aspectos essenciais e ignorando os detalhes menos importantes. Em POO, isso é geralmente feito através de classes abstratas e interfaces (ou métodos abstratos, dependendo da linguagem).



PROFISSIONAL DE PROGRAMAÇÃO

O programador é uma das profissões que mais crescem no mundo e conta com ótimas remunerações. A área de atuação é bem ampla e é considerada uma das profissões do futuro. Mas, afinal, o que faz um programador?

Um profissional de programação, pode ser responsável por Desenvolvimento de Software, Manutenção de Software, Teste de Software, Gerenciamento de Projetos, Suporte Técnico e etc... Pode-se, ao grosso modo, também categorizar em “Back End” e “Front End”.