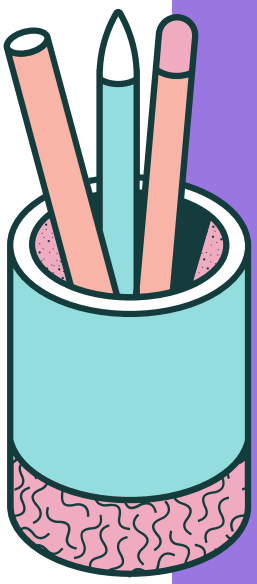
HELLO WORLD!

Aprenda a programar!

Alessandra Diamantino
Caique Figueiredo

CONTEÚDOS



01.

COMO UTILIZAR O PLANO DE ESTUDOS NO TRELLO?

Para utilizar o plano de estudos, assista o vídeo disponível no QR code abaixo e acesse o link que estará em sua descrição.

Ah, não se preocupe, também deixaremos o link aqui :)

https://trello.com/b/HvIS9www/hello-world-plano-de-estudos

Nosso plano de estudos

02.

COMO ACESSAR OS QR CODES?

É muito simples, basta apenas apontar a câmera do seu celular para ele que você será redirecionado(a) para os vídeos!

(caso não consiga pela câmera, existem diversos aplicativos que podem ser baixados para cumprir esse propósito. Também deixaremos o links dos vídeos ao final do livreto, caso seja necessário! Bons estudos :)

Como acessar o Plano de estudos

SEUS PRIMEIROS PASSOS

Primeiros passos com a programação

Os computadores são ferramentas poderosíssimas, porém sem a programação eles não conseguem fazer muita coisa, pois falta o raciocínio lógico e a capacidade de resolver problemas que só os seres humanos possuem. E para resolver este problema, é preciso "conversar" com as máquinas através das famosas **linguagens de programação.** Elas nos possibilitam desenvolver estratégias baseadas em **algoritmos**, que ao serem executados resolvem tarefas pré-estabelecidas.

Atualmente existem diversas linguagens de programação com diferentes propósitos e características, mas todas compartilham da mesma **lógica de programação.**

Todo código escrito por um programador carrega um algoritmo, e este sozinho não consegue fazer nada, porém com a ajuda de uma linguagem de programação é possível realizar a implementação deste algoritmo.

A implementação é o processo de transferir os algoritmos para a linguagem de programação, o resultado desta ação é o programa, um conjunto de instruções escritos em uma linguagem computacional.

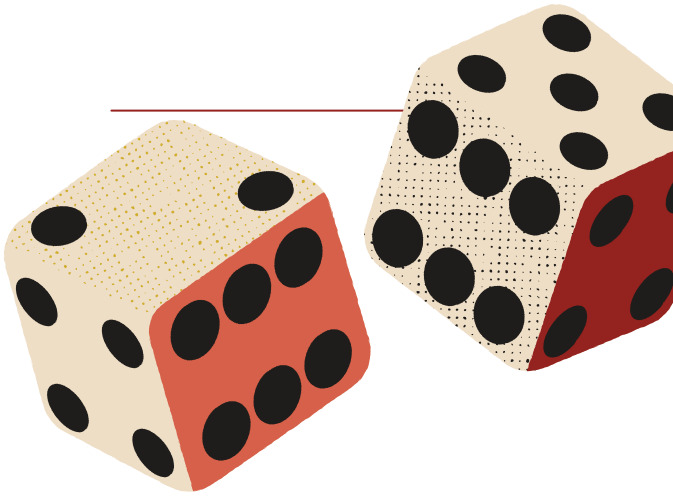
Neste livreto você verá alguns dos conceitos-base da programação utilizando **a linguagem C**, de forma dinâmica e ilustrativa, abstraindo a parte complexa e apresentando o que importa!

Linguagem mais antiga usada até hoje? Temos!

TRANslação da fórmula ou FORTRAN foi criado por John Backus sendo considerada a linguagem de programação mais antiga em uso hoje em dia. A linguagem de programação foi criada para computações científicas, matemáticas e estatísticas de alto nível. FORTRAN ainda hoje é usada em alguns dos supercomputadores mais avançados do mundo.

Aponte a câmera do seu celular

TIPOS DE DADOS



O QUE SÃO?

Os computadores conseguem processar informações apenas na forma de dados, que são convertidos em linguagem de máquina através dos **compiladores de código**. Os dados que o computador processa podem ser de vários tipos e estes variam dependendo da linguagem. Aqui falaremos sobre os mais comuns e que são geralmente encontrados nas linguagens mais usadas atualmente.

1 INTEIRO (INT)

O tipo inteiro representa qualquer número inteiro, seja negativo, nulo ou positivo. Sua notação comumente usada é "int".
Exemplos de int: 1, 2, 3, -2, 0, -49;

2 PONTO FLUTUANTE (FLOAT)

O tipo ponto flutuante é um tipo que representa os números reais, ou seja, aqueles com valores que podem não ser exatos. A sua notação depende de alguns fatores, o "float" é a mais comum, porém algumas linguagens usam também "double" quando o número precisa de mais espaço na memória.
Exemplo de float: 3.14, -23.234, 2, 54.2;

3 CARACTERE (CHAR)

O tipo caractere representa caracteres alfanuméricos. Dados deste tipo são assimilados usando aspas para representar que são caracteres e evitar confusão com outros tipos, já que números podem ser considerados e usados como este tipo. Sua notação é "char".
Exemplos de caracteres: "a", "%", "3", "j", "C";

4 VALOR BOOLEANO (BOOL)

O tipo booleano representa valores booleanos, ou seja, valores lógicos que podem ser ou verdadeiro ou falso. Pela sua natureza lógica, o tipo booleano só pode apresentar um de dois valores.
Sua notação mais comum é "bool".
Exemplo de booleanos: false, true.

Aponte a câmera do seu celular

Aponte a câmera
do seu celular

Como armazenar dados?

Use variáveis ou constantes!

Mas... o que são ?

São a forma mais comum de armazenar os dados em tempo de execução na programação. Usaremos ambos sempre que for preciso guardar algum valor para ser usado novamente em outra parte do programa. Ambos servem como armazenamento de dados, mas possuem algumas diferenças importantes.

Variável X Constante

As variáveis se diferenciam por serem maleáveis, elas estão sempre sujeitas às mudanças e podem ter seu valor trocado após a declaração. Enquanto as constantes não podem ter seu valor trocado após a declaração, pois uma vez assimilada ela terá aquele valor até o fim da execução do programa.

Tipos

Tanto as variáveis quanto as constantes possuem um tipo que define qual conteúdo estas armazenarão. Estes tipos podem variar de linguagem para linguagem, mas geralmente possuem alguns comuns como os inteiros, booleanos, caracteres, etc.

Como usar?

Para declarar uma variável será necessário escolher o tipo, o nome e caso necessário o valor inicial da mesma.

inteiro idade = 10;

Para declarar uma constante será necessário escolher o tipo, o nome e o valor que ela armazenará.

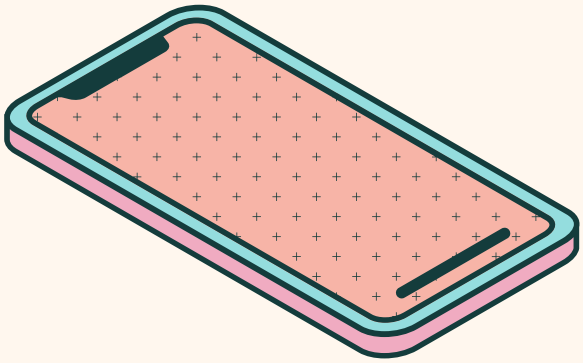
constante flutuante pi = 3,14;

Ambas declarações são parecidas, mas podem variar conforme a linguagem utilizada. Há certas linguagens que não possuem tipagem definida, então não é necessário definir o tipo dos dados enquanto é declarada a variável/constante.

DIFERENÇAS

Uso das variáveis

São usadas quando existe a necessidade de armazenar um dado volátil, incerto, ou seja, que pode variar conforme a execução do programa. Para mudar o conteúdo da variável, basta escrever o seu nome e assimilar os novos dados usando o operador **(=)**.



Uso das constantes

São usadas quando o valor a ser armazenado **não deve e não muda** durante o tempo de execução. O conteúdo, então, é mantido constante durante toda a execução do programa.

LÓGICA DE PROGRAMAÇÃO

O seu maior amigo/inimigo

A definição de lógica de programação pode ser um pouco dura às vezes, então resumiremos em apenas uma frase: é o pensamento expresso no código com objetivo definido e lógico.

Para alcançar algum objetivo, é preciso traçar um caminho, uma rota que guiará ao sucesso. Na programação chamamos de algoritmo, e para contrui-los é preciso usar os conceitos de lógica.

A lógica usa de afirmações, de fatos absolutos para compor um raciocínio que leva a um resultado. No contexto computacional a lógica é muito importante: ela fundamenta o funcionamento dos computadores através da lógica booleana. Na programação usamos da lógica para construir algoritmos eficientes.

Existem pensamentos lógicos que ajudam a compreensão do conceito de lógica, são afirmações que levam ao entendimento de algum fato e podem ser entendidos facilmente:

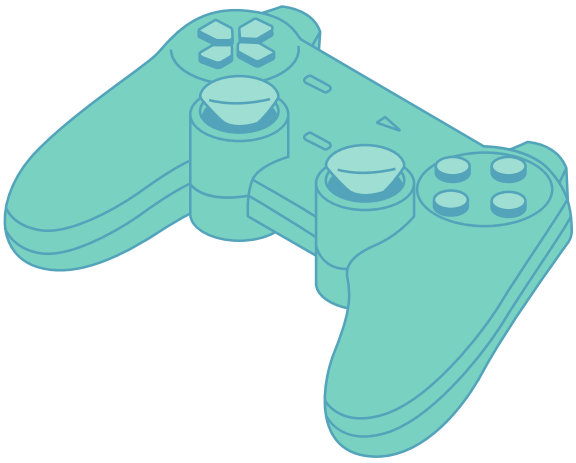
- Todo mamífero é um animal.
- Todo cavalo é um mamífero.
- Logo, todo cavalo é um animal.

A partir de conceitos definidos do que é ser um animal ou um mamífero, é realizada uma análise que conclui na afirmação final de que todos cavalos são animais.

Para usar estes conceitos na programação usaremos expressões booleanas, operadores lógicos e estruturas de controle para escrever e analisar condições e definir a lógica de programação.

Aponte a câmera do seu celular

ESTRUTURAS DE CONTROLE



Os operadores são responsáveis por comparar variáveis ou condições, são geralmente usados em estruturas de controle para verificar alguma regra definida pelo algoritmo.

Aponte a câmera do seu celular

Apenda agora sobre estruturas de controle:

As estruturas de controle são responsáveis por orientar o fluxo do programa, definindo qual caminho o algoritmo deve seguir com base em uma condição. Conforme o algoritmo encontra essas estruturas, ele pode executar diferentes blocos de código dependendo do resultado das **expressões lógicas**.

As condições são baseadas em expressões lógicas resultando em verdadeiro ou falso, podendo conter diversas comparações entre **variáveis** e outros dados. A expressão lógica pode também conter apenas uma variável do tipo booleana, que armazena um valor falso ou verdadeiro.

É possível utilizar operadores relacionais que definem um tipo de comparação que será feita sobre dois valores impostos na condição da estrutura.

Os operadores relacionais são:

- Igual a (=)
- Maior que (>)
- Menor que (<)
- Maior ou igual a (≥)
- Menor ou igual a (≤)
- Diferente de (<>)

Existem ainda os operadores lógicos, que condicionam o valor da expressão para suprir duas condições ou inverter o valor da mesma.

Operadores lógicos:

- NÃO (!) — Inverte o valor da expressão/variável booleana;
- OU (||) — divide a expressão lógica em duas, o valor será verdadeiro caso qualquer uma das expressões for verdadeira;
- E (&&) — Divide a expressão lógica em duas, o valor será verdadeiro caso ambas expressões forem verdadeiras.

AS ESTRUTURAS DE CONTROLE

1
if

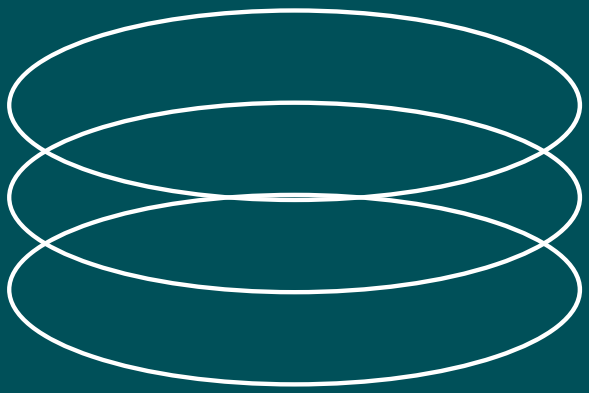
O comando IF serve para alterar o fluxo de execução de um programa baseado no valor, verdadeiro ou falso, de uma expressão lógica.

2
else

Caso a expressão definida na declaração do IF não seja solucionada, há a opção de utilizar o ELSE, que executa um bloco de código quando a condição do IF é falsa.

3
else if

É possível encadear um IF após um ELSE, criando o ELSE IF que será executado caso a condição do primeiro IF for falsa e a condição do IF que vem após o ELSE for verdadeira.



4
switch case

Diferente das estruturas com IF/ELSE, o SWITCH define um comportamento diferente para cada resultado presente em uma variável, dando mais liberdade para o programador definir não apenas condições booleanas mas também com base em vários tipos de dados.

EXEMPLOS

1
if

```
if (exp) {
      comando1; // executado se "exp" for verdadeira
}

comando2; // executado independente da condição, pois está fora do bloco de controle do if
```

2
else

```
if (falso){

    // código não será executado pois a
    //  expressão é falsa

}else{
    // código do else será executado

 }
```

3
else if

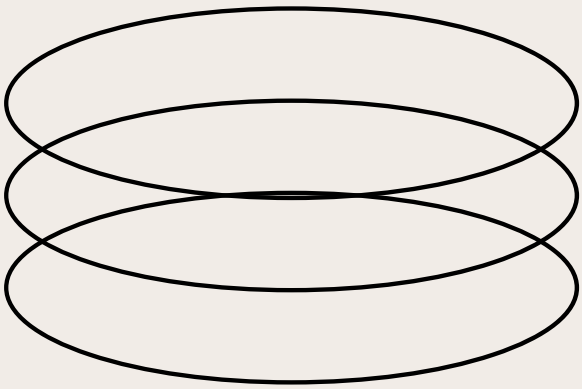
```
if (falso){

    // código não será executado pois a
    //  expressão é falsa

}else if (verdadeiro){

    // código será executado

 }
```



4
switch case

```
switch(condição){
 case valor1: {
 // código a ser executado
 }
 case valor1: {
 // código a ser executado
 }
 default: {
 // código a ser executado
 }
}
```

IF TERNÁRIO

O IF ternário é uma variação do IF que é geralmente usada quando a condição e os resultados são simples e podem ser simplificados. É útil para simplificar o código e diminuir a quantidade de linhas. Mesmo sendo um pouco diferente, ele funciona do mesmo jeito do IF normal.

condição ? : casoVerdadeiro : casoFalso

// O primeiro elemento é a condição, os outros dois são os comandos a serem executados em ambos os casos booleanos.

BÔNUS

ANTES DO SOFTWARE PODER SER REUTILIZÁVEL ELE PRIMEIRO TEM DE SER UTILIZÁVEL!

- Ralph Johnson

Foque em aprender os conceitos básicos e ir evoluindo gradualmente. Não adianta dominar os maiores *frameworks* do mercado e não saber escrever um algoritmo simples.

Vamos falar sobre:

ESTRUTURAS DE REPETIÇÃO

Estruturas de repetição são responsáveis por repetir uma parte do código uma quantidade de vezes definida até satisfazer a necessidade do algoritmo.

São geralmente usadas para evitar a repetição de um código ou para percorrer uma lista ou sequência de elementos a fim de encontrar um valor específico.

01.

WHILE

O WHILE define um bloco de comando que será executado **enquanto** a condição passada pelo programador for verdadeira.
Essa condição será verificada antes de cada execução e então o bloco de código será executado. É importante definir no bloco de execução uma forma de parar a execução do loop caso alguma condição seja satisfeita para não gerar um looping infinito.

02.

DO WHILE

O DO WHILE é parecido com o WHILE, a diferença está na forma como ele lida com a primeira verificação. Enquanto o WHILE verifica a condicional primeiro, antes de executar, o Do WHILE executa o código uma vez antes de verificar a condição imposta. Ambos possuem lógica bem parecida, mas podem gerar resultados diferentes dependendo do contexto em que são usados.

03.

FOR

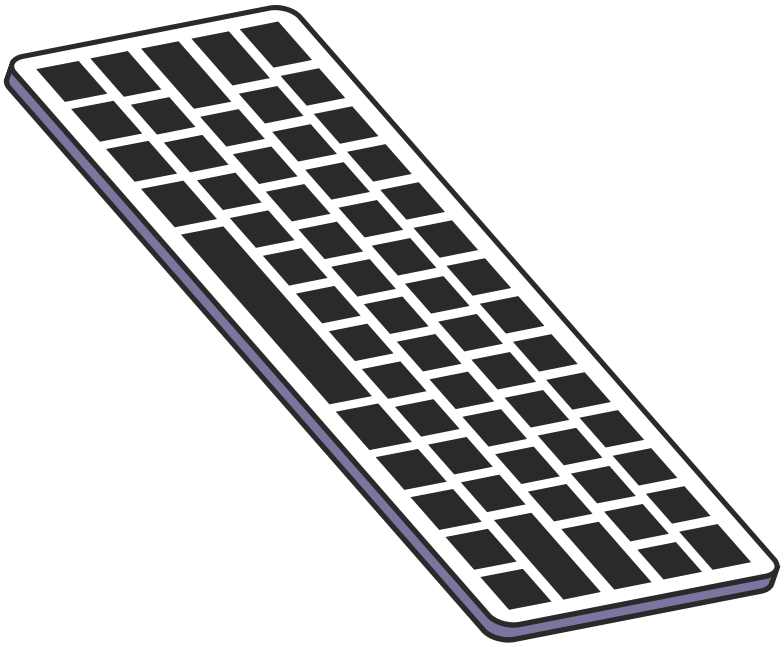
A estrutura de repetição for é uma das mais usadas na programação e apesar de sua sintaxe ser geralmente mais complexa que a do WHILE, o seu funcionamento é bem simples.

A sua sintaxe base é composta por uma estrutura conhecida como “para/até/faça”. **Para** uma variável de valor inicial **até** valor final **faça**.
O primeiro campo é definido o valor inicial da variável a ser usada como contador, o segundo campo é definido a condição para terminar o loop, geralmente usando a própria variável, e o terceiro campo é a operação que será feita para alterar o valor dessa variável, dando prosseguimento ao loop.

EXEMPLOS DE CÓDIGO:

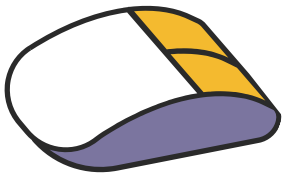
WHILE

```
while(condição) {
  //código a ser executado
}
```



DO WHILE

```
do {
  // Código a ser executado
} while(condição)
```



FOR

```
int i;
for (i = 0; i < n; i++) {
  //código a ser executado
}
```

Aponte a câmera do seu celular

ESTRUTURAS DE DADOS:

vetores, matrizes e registros.

Estruturas de dados são conjuntos de elementos de mesmo tipo, agrupados e que podem ser acessados através de uma única variável.

Existem alguns tipos comuns de estrutura de dados, os mais usados são os vetores, as matrizes e os registros.

Os vetores e as matrizes são bem parecidos em declaração e comportamento, ambos obedecem a um mesmo princípio de estruturação com algumas diferenças na organização dos dados.

A diferença está na disposição dos dados. Enquanto os vetores guardam dados de forma linear, como em uma lista, as matrizes guardam os dados de forma bidimensional, como em uma tabela.

Estas estruturas geralmente possuem um tipo de dado definido, o que faz com que todos os dados tenham que seguir o tipo especificado na declaração da estrutura.

Já os registros se comportam de forma diferente. São estruturas de dados que podem armazenar variáveis de diferentes tipos dentro de sua composição. Este tipo de estrutura da mais liberdade ao programador para que ele possa criar estruturas específicas com tipos de dados específicos e assim evitar a criação de variáveis repetitivas para casos semelhantes.

Aponte a câmera do seu celular

Vetores e matrizes são declarados de forma parecida às variáveis, é definido um tipo, um nome e caso necessário os dados.

A diferença está nos colchetes que indicam que a variável criada será uma estrutura de dados. Nos colchetes deve ser colocada a quantidade de posições que a estrutura terá.

Para diferenciar dos vetores, as matrizes usam quatro colchetes (dois pares) e cada par fica responsável por uma quantidade de posições. É como uma tabela, onde declaramos a quantidade de linhas e colunas, cada célula representará uma posição marcada pela posição em linha e coluna.

FICA DE OLHO

As posições nas estruturas de controle como vetores e matrizes começam sempre no 0, então a primeira posição estará ligada ao numeral 0, a segunda ao numeral 1 e assim por diante.

Mesmo declarando um vetor de 5 posições a contagem dos índices sempre começarão em 0, logo o vetor declarado terá as seguintes posições: vetor[0], vetor[1], vetor[2], vetor[3], vetor[4].

EXEMPLOS

Veja na prática como essas estruturas se comportam!

Vetores:

Float notas[6];

Aqui declaramos o vetor de nome “notas” do tipo float (números decimais) e com 6 posições;

notas [3] = 15.4;

Assimilamos um valor numérico decimal para a posição 3 do vetor.

int nota1 = notas[3];

Aqui estamos resgatando os dados do vetor na posição 3.

Matrizes:

String mapa[6][4];

Aqui declaramos o vetor “mapa” do tipo string (texto) e com 6×4 posições;

mapa[2][3] = “Ifmg”;

Assimilamos um valor em texto para a posição 2×3 do vetor;

Para facilitar, podemos considerar o primeiro colchete como linha e o segundo coluna, assim falamos que o dado está na linha 2 e coluna 3 da matriz;

string If = mapa[2][3];

Aqui estamos resgatando os dados da matriz na posição 2×3

Registros:

```
struct Aluno {
    int idade;
    double notas[];
};
```

Diferente das outras estruturas de dados, os registros precisam de ser definidos antes de inicializados, como se fosse uma planta para construção de uma casa.

O Registro é montado através da palavra reservada **struct** acompanhada do nome e do corpo do registro delimitado pelas chaves.

No corpo é declarado os dados que comporão o registro.

```
Aluno aluno1;
```

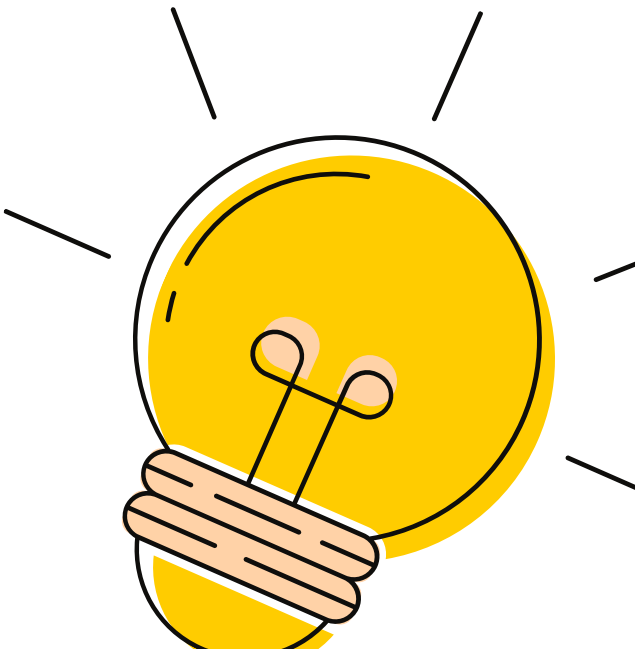
Após a definição do registro é criado uma variável que armazenará dados conforme a estrutura definida anteriormente. É possível criar vários registros do mesmo tipo **Aluno** e cada um poderá conter dados diferentes, a única semelhança será a estrutura dos dados.

```
aluno1.idade = 15;
```

Acessamos o valor idade **dentro** da variável **aluno1** e atribuímos um número a ela do tipo **int**, como definido na declaração do registro.

```
int idade = aluno.idade;
```

Aqui estamos acessando os dados do registro **aluno** na posição **idade**.



E AGORA: FUNÇÕES!

...são de todos os tipos :)

O que são essas funções?

São pedaços de códigos organizados em blocos separados que executam uma atividade e podem ser chamados a qualquer momento dentro do programa.

TIPOS DE *FUNÇÕES*

1
funções sem retorno

São funções do tipo void que não retornam nenhum valor ao serem chamadas, geralmente servem para executar uma determinada atividade.

```
void ligarMotor(){
}
```

2
funções com retorno

Funções com retorno são aquelas que possuem definição de tipo em sua declaração e retornam valores desse tipo quando são chamadas.

```
int somar(int primeiroValor, int
segundoValor){
   return primeiroValor + segundoValor;
}
```

3
funções sem parâmetro

Funções sem parâmetro não precisam de nenhum valor para serem chamadas. Geralmente são usadas para obter algum valor especificado anteriormente no programa.

```
int obterValorDaConstanteX(){
  return x;
}
```

4
funções com parâmetro

Funções com parâmetro são as mais comumente utilizadas, compostas em sua declaração com um parâmetro que deve ser preenchido ao chamar a função.

```
int numeroEpar(int numero){
   return numero % 2 == 0 ? true : false;
}
```

Aponte a câmera
do seu celular

Índice com todos os QR codes:

Aqui estarão listados todos os vídeos sobre os tópicos deste livreto, nosso canal do YouTube e o Plano de Estudos!

Canal do YouTube

https://www.youtube.com/channel/UCABZPP10O5hEd1DAwbaj3Cg

Plano de Estudos

https://trello.com/b/HvIS9wvw/hello-world-plano-de-estudos

Como utilizar o plano de estudos no Trello?

https://www.youtube.com/watch?v=pDlBOR0wYhE

Seus primeiros passos

https://www.youtube.com/watch?v=XwZqYU9XmbQ&t

Tipos de dados

https://www.youtube.com/watch?v=IGCmeSA95Gk&t

Como armazenar dados na programação

https://www.youtube.com/watch?v=aB7bLGYoN8I

Lógica de programação

https://www.youtube.com/watch?v=HyKpgC7iNe0&t

Estruturas de controle

https://www.youtube.com/watch?v=6HPMSgL0SfM&t

Estruturas de repetição.

https://www.youtube.com/watch?v=VegD7zwRaoM&t

Estruturas de dados

https://www.youtube.com/watch?v=Nf5Xf5mUhk8

Funções

https://www.youtube.com/watch?v=R7ADc-ZVsZ0

Referências utilizadas

LEMOS, Karina; FIGUEIREDO, Rosinei. **Apostila de Introdução a Programação**. São João Evangelista, f. 87, 2019. 81 p.

LESTAL, Justin. História das linguagens de programação. DevSkiller. 2020. Disponível em: https://devskiller.com/pt/historia-da-programacao-idiomas/. Acesso em: 3 ago. 2022.

EU FAÇO PROGRAMA. As Melhores Frases sobre Programação – Edição 2021. Eu faço programas. 2021. Disponível em: https://eufacoprogramas.com/as-melhores-frases-sobre-programacao/. Acesso em: 10 ago. 2022.

Hello World!

2022

Autores
Alessandra Diamantino
Caique Figueiredo

Supervisão
Eduardo Trindade