

Introdução à Lógica de programação

Por Micael Mota.

O objetivo desta apostila é te preparar para aprender lógica de programação de maneira simples e te ajudar a construir um raciocínio lógico mais eficiente para lidar com os próximos desafios de entrar na área de tecnologia.

Eu quero que você tenha uma base realmente sólida para não cometer certos erros que vão te impedir de avançar. Então se você quer realmente trabalhar com tecnologia e fazer a diferença, não cometa o erro de desvalorizar essa etapa inicial.

Para fins didáticos iremos nos aprofundar somente em temas diretamente conectados com Lógica de programação e faremos isso de uma maneira leve, pois nesse primeiro momento, para a maioria das pessoas, é assustador a quantidade de informação e termos técnicos.

O que é lógica de programação?

Como todo bom resolvidor de problemas, vou começar dividindo essa pergunta em duas:

1. O que é lógica?

Você calça primeiro o sapato e depois veste a meia? Com certeza não. Isso é lógica. E ela já vem de fábrica no nosso cérebro, na nossa forma de raciocinar, você só precisa treiná-la.

Quando eu era pequeno e estava aprendendo a amarrar os cadarços me estressei bastante porque não sabia fazer direito, não conseguia apertar o suficiente, eles sempre soltavam. Aí um dia minha irmã mais velha chegou em casa com uma sapatilha que não tinha cadarço, fiquei pensando como seria bom ter um sapato que não tivesse cadarço também pra eu não precisar amarrá-los.

No outro dia enquanto estava me arrumando para a escola, na hora colocar os sapatos, me veio um pensamento do nada dizendo “eu não preciso amarrar esse troço”, e aí simplesmente coloquei os cadarços de fora para dentro dos furos e nunca mais perdi tempo na minha vida amarrando os sapatos de novo.

Meu cérebro fez isso, sozinho. E eu tenho certeza que você já passou por algo parecido, um dia que deu um estalo e você percebeu que poderia fazer algo melhor e mais rápido. É como descobrir um atalho e economizar horas de um engarrafamento gigantesco.

Então a dica aqui é estar atento a todo momento e fazer disso um hábito, para que você consiga mais estalos com cada vez mais frequência, e principalmente quando estiver programando.

Fique atento!

2. O que é programação?

Imagine um programa de televisão. O Jornal Nacional, por exemplo. Geralmente começa com a musiquinha tocando, os apresentadores falam “Boa noite” e contam as principais manchetes, depois da última manchete eles falam a frase “a partir de agora no Jornal Nacional”, e aí a música termina com aquela parte que todo mundo se conhece.



Perceba que todos os programas de tv seguem um certo roteiro. Tem alguém que escreve esse roteiro, geralmente é chamado de roteirista ou diretor.

Programação é a arte de escrever programas.

- Mas o que é um programa de computador?

Sabe o Google, Facebook, o Instagram, o Youtube? Todos esses sites e aplicativos que você acessa pelo seu celular, inclusive seu celular, seu computador, seu microondas, a televisão, o controle remoto da televisão e todos os outros equipamentos eletrônicos que você usa no dia-a-dia. Eles têm ou são um programa. Eles são programados.

Um programa é simplesmente uma sequência de passos executados em uma ordem específica para realizar uma ação e/ou gerar um resultado. Na nossa área, essa sequência de passos é definida através de um código (algoritmo - o código fonte).

Agora junte os dois. Você sabe que primeiro deve se vestir a meia e depois calçar o sapato (Lógica) e sabe que existem vários passos disponíveis para realizar uma ação (programa).

Lógica de programação é a forma como você organiza qual passo você deve dar em cada momento para realizar a ação que você deseja.

Em outras palavras, você vai aprender alguns comandos (os passos) e vai escrever códigos (o roteiro/programa) usando lógica para definir qual comando vai ser executado primeiro, qual vai ser depois, e assim sucessivamente até atingir o seu objetivo.

Veja abaixo um programa que geralmente executamos todo dia chamado "Tomar banho":

Início

1. Entrar no banheiro;
2. Tirar a roupa que está usando;
3. Ligar o chuveiro;
4. Se molhar, ensaboar, enxaguar;
5. Desligar o chuveiro;
6. Se secar;
7. Vestir a roupa limpa;
8. Sair do banheiro.

Fim

Agora pense um pouco e faça o seguinte exercício:

- Escreva cada passo que você deve executar caso fosse tomar um banho exatamente agora, detalhando a maior quantidade de passos possível.

Pensar em cada passo que você vai executar detalhadamente te dará uma visão completa de algo que você faz hoje no modo automático, sem nem pensar. Provavelmente você vai identificar vários passos que podem ser excluídos ou otimizados, como por exemplo lembrar de levar a toalha sempre, porque com certeza você já esqueceu algum dia.

Dica: tente fazer isso com coisas comuns no dia-a-dia, você vai aprender e ganhar muito tempo identificando essas pequenas coisas desnecessárias que fazemos durante o dia.

Princípios básicos

Definindo um computador

Daqui para frente chamaremos todos os dispositivos (notebooks, celulares, tablets, etc) de computadores.

Todos eles possuem uma estrutura básica:

- Unidades de entrada
Mouse e teclados, touch screens, microfones, etc
- Unidades de processamento
Processadores, placas de vídeo
- Unidades de armazenamento
HD, cartões de memória, memória ram
- Unidades de Saida
Monitor, caixas de som

Nós, humanos, interagimos com os computadores através das unidades de entrada e saída.

Linguagem de Programação

Uma linguagem de programação é um método padronizado para escrever programas de computadores. Toda linguagem possui um conjunto específico de comandos (instruções) definidos por algumas regras (de sintaxe e semântica) que devemos seguir para escrever os códigos.

Você não precisa saber inglês para aprender a programar, mas a maioria das linguagens possui comandos em inglês, por isso eu recomendo fortemente que você comece a estudar inglês a partir de agora. Sugiro começar utilizando o aplicativo Duolingo, porque eu mesmo comecei aprender usando ele e com apenas 5 minutos por dia eu consegui aprender o bastante de inglês para facilitar muito na hora de aprender a minha primeira linguagem de programação.

Lembre-se: você vai precisar saber inglês se quiser ir longe na área de tecnologia.

Analogia do amigo

Escrever um programa de computador é como escrever uma mensagem pedindo um favor para seu amigo. Imagine que você pede a ele para pegar o carregador do seu celular, se for aquele amigo que te visita sempre e conhece bem o seu quarto ele vai entrar e pegar facilmente. Mas se for um amigo não muito próximo, talvez você precise falar como abrir a porta, como ligar a luz, em qual gaveta está, qual a cor... e ele ainda vai demorar um tempo tentando encontrar o carregador.

No nosso caso, o computador é pior que o amigo não muito próximo, porque além de não te conhecer e não conhecer seu quarto, ele não pensa, não raciocina e **só faz exatamente o que você mandar**. Então você precisa dizer cada passo que ele tem de executar um a um e na ordem correta.

A quantidade de detalhes que você vai precisar dar para ele depende da linguagem de programação que você escolhe utilizar. Existem centenas de linguagens diferentes e a maioria delas segue um certo padrão e são bem parecidas. A boa notícia é que se você aprender lógica muito bem, depois que você aprender uma, vai ser bem mais fácil aprender qualquer outra.

Verdadeiro ou Falso

Aprendemos o que significa algo ser verdadeiro ou falso nos primeiros anos de idade quando ainda somos uma criança. Geralmente quando acontece algo como no quadrinho ao lado.

Nesse momento aprendemos um dos conceitos mais importantes das nossas vidas: Validade. O valor da verdade. Validade é um valor que podemos atribuir às coisas. Que pode ser verdadeiro ou falso.



Um é oposto do outro. Se algo é verdadeiro, então não é falso. E se é falso, então não é verdadeiro.

Se eu falo pra você: "Eu tenho 22 anos", você nem percebe, mas a primeira coisa que você faz é um julgamento de validade da minha afirmação. Sua mente vai inicialmente pensar: "ele **pode ter** 22 anos, ou ele **pode não ter** 22 anos". E a partir daí, em milésimos de segundos começam outros julgamentos, como por exemplo: "ele aparenta ter mais, ou menos" ou "realmente, você parece ter essa idade".

Perceba que para qualquer afirmação feita, não há uma terceira opção, ou é verdade, ou é mentira.

E se eu faço uma afirmação que é verdade, então não é verdade que essa afirmação não é verdade.

Ficou confuso? Pera que vou te dar um exemplo:

Imagina que você falou para um amigo: “eu vou me casar” e realmente você vai se casar. Mas aí seu amigo fala - “é mentira kkkkk tu nunca vai achar uma namorada”. Seu amigo está negando a sua afirmação. Mas o que está fazendo não é válido, porque você vai sim se casar.

Ou seja, **não é verdade** que **você não vai se casar**, por que **você vai sim se casar**.

Agora Imagine que seu amigo fala “Se eu chegar antes das 11:00 nós vamos almoçar juntos”. Aí você responde: “Isso se eu estiver com fome” porque você sempre faz um lanche às 10:30.

Vamos analisar mais fundo esse caso:

1. Se seu amigo chegar antes das 11:00 e você estiver com fome quando ele chegar
 A B
você vão almoçar juntos
 C

Ou seja, se **A e B** forem verdade então **C** também será

2. Do outro lado

- Se seu amigo não chegar antes das 11:00 ou você não estiver com fome quando ele chegar
 $Não A$ $Não B$
você não vão almoçar juntos
 $Não C$

Ou seja, se **A ou B não** forem verdade então **C** também **não** será

Observe que no segundo caso, basta que uma das condições não seja verdade para que o resultado também não seja. Isso significa que se seu amigo chegar atrasado **ou** você não estiver com fome vocês não irão almoçar juntos.

Preste atenção nas palavras “e” e “ou”. São chamadas conectivos e elas estabelecem relações entre condições para algo acontecer.

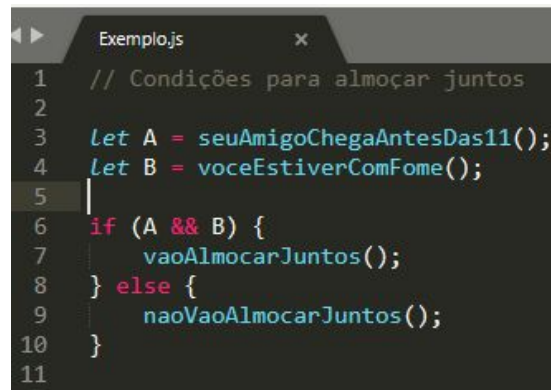
O “e” faz com que as duas condições relacionadas precisem ser obrigatoriamente verdade para prosseguir. Se uma delas não acontece, impede o processo.

O “ou” faz com que se pelo menos uma das condições forem verdade, possamos prosseguir. Nesse caso, o processo só é impedido de prosseguir caso as duas condições não sejam verdade.

Parabéns, você acabou de aprender os princípios de Lógica clássica!

Os computadores usam essas validações de verdade em todo o processo de funcionamento. A todo momento eles estão testando a validade das informações que eles têm na memória como condição para executar as suas ações.

Vamos entender mais sobre isso nos próximos capítulos.



```
Exemplo.js x
1 // Condições para almoçar juntos
2
3 let A = seuAmigoChegaAntesDas11();
4 let B = voceEstiverComFome();
5
6 if (A && B) {
7   vaoAlmocarJuntos();
8 } else {
9   naoVaoAlmocarJuntos();
10 }
11
```

Como os computadores entendem

Eu disse que o computador é pior que um amigo distante, porque ele não te conhece, não pensa e só faz exatamente o que você mandar. E você já sabe que a forma de mandar o computador fazer as coisas é através das linguagens de programação. Mas como funciona esse negócio de escrever o código e o computador entender e executar?

Bom, atualmente a nossa vida está bem mais fácil, porque hoje nós usamos programas para fazer outros programas. Esses programas são chamados de compiladores. Eles transformam o nosso código em um programa executável pelos computadores.

O que eles fazem é basicamente criar uma lista de passos extremamente detalhada para o computador executar.

Lembra do programa Tomar-banho que eu pedi para você escrever da forma mais detalhada possível? Os compiladores fazem isso de forma muuuuito mais detalhada, passo a passo, linha a linha. Traduzindo o nosso código para uma linguagem que o computador entende.

Você deve estar perguntando “Sim, mas que lista e que linguagem é essa que eles entendem?”.

Basicamente existe um componente chamado processador que é responsável por processar essas instruções (os passos). Os fabricantes dos processadores disponibilizam um conjunto de comandos que podem ser utilizados para controlar o computador. Então o que os compiladores fazem é transformar os comandos que a gente escreve nos comandos disponibilizados pelo fabricante, ou seja, os comandos que os processadores entendem.

Binários

Você provavelmente já deve ter ouvido falar dos números binários ou visto alguma imagem como essa ao lado.

Neste capítulo vamos aprender o que são e qual a relação deles com os computadores e com a lógica.



A gente aprende na escola os números decimais, certo? São dez números (de 0 à 9) que usamos para representar os valores.

“O sistema binário ou de base 2 é um sistema de numeração posicional em que todas as quantidades se representam com base em dois números, ou seja, zero e um. [Wikipédia](#)”

Os computadores utilizam o sistema binário para todas as suas operações. Isso porque nos níveis mais baixos, nos circuitos digitais, placas mães e em todos os componentes de um computador que você pode tocar (ou seja, hardware) esse é o protocolo de comunicação utilizado. Portanto, no final das contas, tudo no computador é 1 e 0 passando de um lado para outro e fazendo as coisas acontecerem.

Basicamente “0” e “1” são usados para representar o estado dos componentes e circuitos, “0” significa que **não há tensão** elétrica ativa, e “1” significa que há tensão elétrica ativa.

Mas por que eu estou lhe falando isso?

Porque como eu disse no capítulo anterior, os computadores fazem validações de verdade a todo momento e os códigos que iremos fazer quando estivermos programando vão dizer para o computador fazer essas validações.

Acho que é importante você aprender como funciona o sistema binário de numeração, como converter de decimal para binário e vice-versa, mas para não sairmos do foco não vou me aprofundar mais sobre isso. Mas vamos assumir de agora em diante o seguinte:

O número 1 tem valor de verdade verdadeiro

O número 0 tem valor de verdade falso

Dissecando as linguagens

Neste capítulo você vai aprender a estrutura base de todas as linguagens de programação.

Palavras reservadas

Eu falei que toda linguagem possui um conjunto específico de comandos (instruções) que podemos utilizar, lembra? Então... Esse termo é muito comum, e próxima vez quando você ouvir a expressão “Palavras reservadas” saiba que estamos falando simplesmente de uma palavra que pertence ao conjunto de comandos da linguagem em questão.

Variáveis

Você vai ouvir muito sobre elas e elas podem te enlouquecer se você não ler esta parte aqui. Nós usamos variáveis nos códigos o tempo inteiro para processar os dados e, ao pé da letra, variável é um valor que pode variar.

Mais tecnicamente, uma variável é uma representação (ou referência) de um espaço de memória onde guardamos os dados que o nosso programa vai utilizar durante a execução.

“Como assim espaço de memória, Micael?”

Sabe quando vc vai comprar um computador ou um celular e pergunta quantos gigas ele tem? Geralmente quando perguntamos isso estamos falando da memória ram. Essa informação é importante porque é justamente a memória ram que os programas utilizam para fazer os processamentos dos dados.

Além disso, memória ram geralmente é um recurso escasso, por isso os nossos códigos devem sempre otimizar o uso dela.

Toda vez que você cria uma variável você está separando um espaço na memória ram do computador e reservando para armazenar dados do seu programa.

A forma de criar variáveis é diferente em algumas linguagens de programação. Mas podemos separar em duas categorias: as linguagens que separam cada tipo de variável (tipadas) e as linguagens que tratam todas variáveis da mesma maneira (não tipadas).

Vamos aos exemplos:

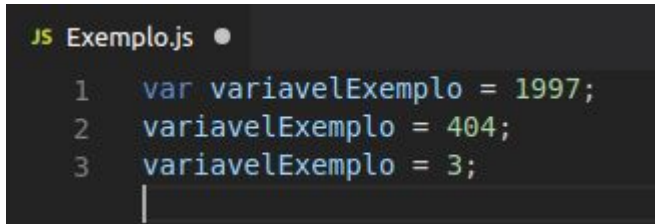
Imagina que você vai fazer uma festa de aniversário na sua casa e chamou vários amigos. Daí a festa tava bacana, seus amigos ficaram até a noite e você os convidou para dormir. Vamos supor que você tem 20 quartos na casa. O que você faria agora?

Seja lá como for, você vai organizar dizendo para cada amigo em que quarto ele vai dormir. Você pode colocar qualquer amigo em qualquer quarto e trocar de quarto quando quiser.

Nesse exemplo a memória ram é a sua casa, as variáveis são os quartos, os dados são os seus amigos.

Outro bom exemplo é o do armário. Quando a gente cria uma variável e insere um dado nela, é como se a gente abrisse uma gaveta e colocasse o um objeto lá dentro. E acessar uma variável é como se a gente abrisse a gaveta e olhasse o objeto.

Na imagem ao lado, na linha 1 a variável é criada. Esse é o momento em que o computador separa o espaço de memória e reserva para o nosso programa inserindo o valor 1997.



```
JS Exemplo.js ●
1  var variavelExemplo = 1997;
2  variavelExemplo = 404;
3  variavelExemplo = 3;
```

Nas linha 2 e 3 o que acontece simplesmente é a alteração do valor da variável. É como se a gente tivesse tirando um amigo do quarto e colocando o outro. Ou seja, no final da execução desse programinha, o amigo que está o quarto é o “3”.

O trecho de código do exemplo foi feito com a linguagem Javascript. Mas não se preocupe com isso agora, porque a forma de escrever (o que chamamos de sintaxe) varia de linguagem para linguagem, mas o conceito é o mesmo em todas.

Geralmente vamos usar o sinal de igual “=” quando criarmos ou acessarmos variáveis.

Podemos criar uma variável com qualquer nome (na foto o nome que usei foi variavelExemplo), mas geralmente as linguagens definem alguma regra, como por exemplo, não começar com números ou caracteres especiais.

Simbólos especiais

Eu chamo de símbolos especiais alguns caracteres que representam um comando por si só. Por exemplo o sinal de igual “=” é usado na maioria das linguagens para indicar o comando de atribuição de valor as variáveis.

Temos um coleguinha também chamado ponto-e-vírgula que vai te quebrar o juízo quando você estiver aprendendo C ou Java por exemplo. Na maioria das linguagens o “;” é utilizado para sinalizar o final de um comando. O problema é que a gente sempre esquece esse bendito e aí o código não compila. Então presta atenção nisso e lembre-se sempre dele.

Ah.. e não se sinta estúpido quando isso acontecer com você. Meu professor, Fábio Martinez, que tem mais de 15 anos programando até hoje esquece o bendito, imagina você que tá começando agora. Só preste atenção e sempre cheque isso quando seu código não compilar (caso a linguagem que você estiver utilizando precise dele).

Outros símbolos especiais são os símbolos matemáticos, que são usados para fazer operações matemáticas, mas dependendo da linguagem podem ser usados para outras ações também.

Lembra do **e** e do **ou**? Geralmente os símbolos usados para representar eles são o “&” e “|”.

O ponto aqui é você saber que sempre em todas as linguagens vai ter algum conjunto de símbolos especiais. E aí não tem jeito, você vai precisar aprender como que ele funciona naquela linguagem para encontrar a melhor forma de utilizar ele.

Operações matemáticas

Essa é uma parte bem simples de entender. Eu ousaria dizer que todas as linguagens tratam isso de maneira igual, mas né... não conheço todas as linguagens então não posso afirmar isso. Mas geralmente as linguagens tratam as operações matemática com os mesmos símbolos de uma calculadora e obedecendo a precedência matemática.

Na maioria das vezes gente vai usar os símbolos de adição “+”, subtração “-”, multiplicação “*” e divisão “/”. Não tem segredo. Se você sabe o básico de matemática (aritmética), você vai entender fácil como escrever programas usando essas operações.

Tem algumas operações especiais que geralmente não prestamos muito atenção na escola, mas por exemplo, se quisermos saber o resto da divisão de um número pelo outro, geralmente vamos utilizar o símbolo “%”.

Toda vez que você ver por exemplo: `variavelExemplo = 10 % 3;` lembre-se que essa operação é o resto da divisão. Ou seja, o valor da `variavelExemplo` vai ser 1, porque se você dividir 10 por 3, dá resto 1 certo?

Condições

No nosso dia-a-dia fazemos milhões de desvios condicionais e nem percebemos. Sério, acredite! Mas vamos focar nos que nós percebemos, por exemplo quando combinamos com nosso amigo de almoçar juntos caso ele chegue cedo e a gente esteja com fome.

As condições são usadas para fazer desvios. Como na vida real, os programas verificam se certas condições são verdadeiras para realizar alguma ação.

Em outras palavras, desvios condicionais são simplesmente momentos em que os programas executam certo trecho de código dependendo de alguma condição ser verdadeira ou não.

Se a gente fosse programar em português (e podemos porque tem uma linguagem chamada portugol) a gente escreveria por exemplo:

```
se (meuAmigoChegarCedo e euEstiverComFome) {  
    vamosAlmocarJuntos = verdadeiro;  
} senão {  
    vamosAlmocarJuntos = falso;  
}
```

Basicamente o que acontece é que na hora da execução, caso a condição seja verdadeira, o código dentro desse bloco do “**se**” vai executar, caso a condição seja falsa, o código dentro do bloco do “**senão**” não executa.

Geralmente também podemos “encadear” esses comandos de condições. Ou seja, caso uma condição seja falsa, a gente verifica outra. E podemos fazer isso quantas vezes quisermos.

Essa é a minha parte favorita das linguagens, porque as vezes a gente escreve de uma forma, e aí no outro dia olhamos o código e percebemos “caramba eu precisava verificar só essa condição pra isso funcionar” ou então “vish, tem uma condição que isso não vai funcionar correto”.

Essa é a parte onde a sua habilidade de lógica vai ficar mais evidente. Por isso, vamos treinar bastante isso na nossa jornada.

Repetição

Em lógica de programação utilizamos a expressão “Laço de repetição” ou em inglês “Loops”. Que são comandos que nós utilizamos quando precisamos que um certo trecho de código seja executado várias vezes.

O que você diz quando quer que alguém faça alguma coisa várias vezes? Depende de quantas vezes é necessário repetir e da condição de parada, certo?

Por exemplo, imagina que você está ensinando uma criança atravessar a rua. Provavelmente você vai ensinar a ela as cores do semáforo e falar: “enquanto o sinal estiver vermelho você espera, quando ficar verde você atravessa”.

A condição para ela **parar** de esperar é o sinal ficar verde. Preste atenção que a palavra **enquanto** significa que ela deve repetir a ação de esperar até que a condição de parada seja atendida. E é literalmente como se a gente dissesse: “Computador, execute tal ação **enquanto** essa **condição** aqui for **verdade**”;

Todo laço de repetição tem a mesma base lógica: uma ação que deve ser executada e uma condição de parada. As linguagens geralmente disponibilizam alguns comandos mais elaborados para facilitar o uso. Mas todos seguem o mesmo princípio.

Daqui em diante todo mundo se bate para entender. São coisas simples mas que precisam de uma abstração maior. Você precisa realmente estimular a sua imaginação para entender esses 3 últimos itens.

Vetores e Matrizes

Se você entendeu a minha explicação de variáveis e de espaço de memória você vai entender o que são vetores e matrizes com as frases:

- Um vetor é uma linha no armário.
- Uma matriz é o armário inteiro.



Vetores e matrizes são estruturas usadas quando precisamos trabalhar com vários dados do mesmo tipo. Inclusive na maioria das vezes utilizamos junto com laços de repetição.

Por exemplo, imagina que você quer fazer um programa para calcular a média das idades de todas as pessoas da sua rua. Vamos supor que tenha 500 pessoas na sua rua.

Você pode criar 500 variáveis para fazer isso, mas vai perder um tempão escrevendo.

O que a gente faz é criar um vetor, que geralmente o comando é bem parecido com o de criar uma variável, e usamos um laço de repetição para armazenar os 500 dados e fazer o cálculo.

PS: Se você for espertinho(a) vai saber que não precisamos do vetor para esse exemplo que eu dei, mas nesse caso aqui, o foco é entender o conceito de vetores e matrizes.

Eu falei que quando a gente cria uma variável a gente reserva um espaço na memória do computador para o nosso programa. E quando a gente cria um vetor ou uma matriz é a mesma coisa, a diferença é que a gente pode reservar vários espaços de memória de uma só vez.

As matrizes também podem ter várias dimensões. Eu costumo dizer que uma matriz é um armário de armário. É como se você abrisse uma gaveta do armário e lá dentro tivesse outro armário. Entendeu?

Só para você já ir se habituando, geralmente a os comando para trabalhar com vetores são parecidos com no exemplo abaixo:

```
var vetorDeNumeros[500]; => Isso seria a gente criando um vetor de 500 posições
vetorDeNumeros[1] = 10; => Isso seria a gente colocando o valor 10 na gaveta 1
```

Versão 1.0

```
vetorDeNumeros[3] = 15; => Isso seria a gente colocando o valor 15 na gaveta 3
```

Mas foque agora só em entender a abstração e o que significa, porque isso varia de linguagem para linguagem e quando você estiver programando mesmo, tudo isso vai se encaixar na sua cabeça.

Funções

Uma função é um trecho de código separado para executar uma ação específica. Elas podem receber e retornar valores. E assim como os laços de repetição, as funções também são usadas para não repetirmos o mesmo código várias vezes, mas só que de uma maneira diferente.

Pense agora na sua rotina diária, geralmente temos um horário para acordar, para dormir, para comer, etc... Mas tem vezes que as coisas saem da rotina e a gente fica meio desorganizado. Agora porque que eu tô falando isso?

Pra mim essa foi a forma mais simples de entender uma função:

A lógica para criarmos uma função é quando sentimos a necessidade de executar algum trecho de código do nosso programa em um momento qualquer fora da rotina. É nesse momento que a gente pensa “hum... isso deveria ser uma função”.

Mas vamos ao exemplo do dia a dia. Imagina que um entregador vai todo dia levar o seu almoço no mesmo horário. Ou seja, ele pode estar em um laço de repetição executando um trecho de código para lhe entregar o almoço. Mas aí, um dia você precisa ir no banco no horário do almoço e quer comer mais tarde. Ou seja, precisa que o trecho de código do laço de repetição seja executado fora da rotina. Esse é o momento de você criar a função “Entregar almoço” e de agora em diante vai poder executar ela quando quiser.

Bom, agora que você leu até aqui, espero que eu tenha te ajudado a entender o que acontece por trás dos códigos e programas e tenha te dado uma visão geral de onde você está se metendo. Agora é a hora de você aprender a sua primeira linguagem de programação.