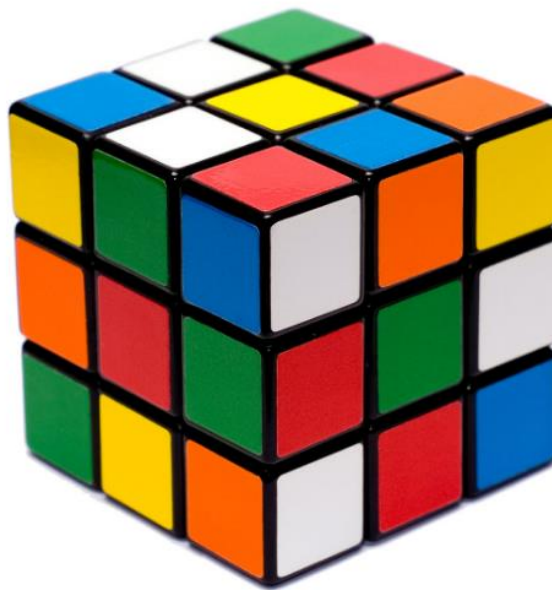


**Abner da Conceição Ferreira  
Giovanna dos Santos Ferreira**

# **Lógica de Programação**

## **A arte de pensar ordenadamente**



**1ª EDIÇÃO  
AMPARO-SP  
2023**

Esta obra está licenciada em Creative Commons.

<a rel="license" href="http://creativecommons.org/licenses/by-nc/4.0/"></a><br /><span xmlns:dct="http://purl.org/dc/terms/" property="dct:title">Lógica de Programação-A arte de pensar ordenadamente</span> de <a xmlns:cc="http://creativecommons.org/ns#" href="https://github.com/abnerdev23/L-gica-de-Programa-o--A-arte-de-pensar-ordenadamente/tree/main" property="cc:attributionName" rel="cc:attributionURL">Abner da Conceição Ferreira e Giovanna dos Santos Ferreira</a> está licenciado com uma Licença <a rel="license" href="http://creativecommons.org/licenses/by-nc/4.0/">Creative Commons - Atribuição-NãoComercial 4.0 Internacional</a>. <br />Baseado no trabalho disponível em <a xmlns:dct="http://purl.org/dc/terms/" href="https://github.com/abnerdev23" rel="dct:source">https://github.com/abnerdev23</a>. <br />Podem estar disponíveis autorizações adicionais às concedidas no âmbito desta licença em <a xmlns:cc="http://creativecommons.org/ns#" href="https://creativecommons.org/licenses/by-nc/4.0/" rel="cc:morePermissions">https://creativecommons.org/licenses/by-nc/4.0/</a>.

This work is licensed under the Creative Commons Atribuição-NãoComercial 4.0 Internacional License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/4.0/>.

**Atribuição-NãoComercial 4.0  
Internacional**



Esta não é uma Licença de Cultura Livre.



## **Dedicatória**

À minha esposa Marta, que sempre me apoiou em todos os meus Projetos, Faculdades e também nas adversidades da vida, pois “ ao lado de um grande homem sempre tem uma grande mulher” e você é esta pessoa que Deus colocou na minha vida, te amo muito .

À minha filha Giovanna que abraçou a carreira de tecnologia e dedicou muitos anos de estudos, finais de semana, abdicando até da vida social para estudar e se especializar e está começando a colher os frutos. Que Deus te abençoe grandemente nesta jornada, pois tecnologia é análoga a carreira de Medicina e de Professor, se estuda para sempre, não pode parar, aliás isto é muito bom, pois mitiga os riscos de Alzheimer quando chegarmos na melhor idade.

À meu filho Artur que está descobrindo novos caminhos, e encontrou um motivo para estudar; solucionando o problema das pessoas com a tecnologia.

Aos meus alunos que sempre me incentivaram a produzir aulas melhores e a continuar estudando e pesquisando.

À todos os professores pelos quais eu passei, e tiveram paciência comigo e acreditaram que aquele menino poderia se tornar “alguém na vida” , me impondo limites, cobrando tarefas e seminários ( que aliás melhoraram minha timidez).

Aos meus pais por me incentivarem sempre a estudar, pois tive poucos brinquedos, mas livros nunca faltaram na casa deles.

Finalmente, seria eu ingrato se não agradecesse a Deus por ter me emprestado inteligência, força de vontade, determinação, resiliência e acima de tudo o desejo de compartilhar com outras pessoas 100% do 1% que sei, mas com certeza de uma forma clara, transparente e traduzida, pois é isto o que falta nos dias atuais para uma educação de qualidade.

*“Os verdadeiros artistas criam coisas reais e  
que serão usadas”.*

*(Steve Jobs)*

## **Sobre os Autores**

### **Abner da C.Ferreira**

Abner Ferreira nascido em Guarulhos-São Paulo em 1973, sempre gostou de estudar mas a virada de chave mesmo foi SENAI, onde se formou em 1989 no curso de Eletricista de Manutenção. Trabalhou na Indústria, Comércio, Empreendedor e autodidata, percebeu que a chave para abrir as portas que precisava estava nos estudos. Graduado em Matemática e Física pela Universidade Guarulhos\_UNG, Graduado em Pedagogia pela Faculdade da Aldeia de Carapicuíba-FALC e Graduando em Engenharia da Computação pela UNIVESP, lecionou em diversos colégios particulares, sindicato dos hotéis de Guarulhos e atualmente é professor efetivo da Secretaria da Educação do Estado de São Paulo. Atuou como supervisor do PIBID do Instituto Federal de São Paulo-pólo Guarulhos onde orientava dez residentes da Graduação em Matemática.

Github: <https://github.com/abnerdev23/>

### **Giovanna dos Santos Ferreira**

Giovanna Ferreira, nascida em Guarulhos-São Paulo em 2003, é graduada em Análise e Desenvolvimento de Sistemas pela UNIP-Universidade Paulista, desenvolveu vários TCCs, resolveu empreender em 2023 recém formada prestando serviços para a YPÊ-Química Amparo, na área da TI, em parceria com uma grande empresa de Tecnologia de Amparo, a Shark It.

Github: <https://github.com/gihdeveloper17>

## **Prefácio**

O objetivo desta obra é facilitar o aprendizado de qualquer linguagem de programação de uma forma divertida e bem visual, pois em qualquer curso o mais importante é a didática e, se o professor ensinar sobre manga nas aulas, posteriormente deverá ter exercícios de manga, trabalhos sobre manga e consequentemente prova sobre manga, o que infelizmente, na maioria das vezes não acontece. Nos apoiamos em grandes autores para escrevermos esta obra, então, não temos aqui a pretensão de forma alguma de excluir ou até mesmo denegrir nenhum curso técnico ou de graduação, mas sim temos o intuito de colaborar com o aprendizado e desenvolvimento não somente de jovens, mas de qualquer pessoa que queira estudar programação.

Bons estudos!

## LISTA DE FIGURAS

Figura 1. Ábaco .....	12
Figura 2. Régua de Cálculos .....	12
Figura 3. Pascaline.....	13
Figura 4. Tear Programável.....	13
Figura 5. Máquina das Diferenças.....	14
Figura 6. George Boole .....	14
Figura 7. Mark I .....	15
Figura 8. Alan Turing .....	15
Figura 9. Enigma .....	15
Figura 10. A Máquina de Turing (A Bomba) .....	16
Figura 11. Steve Jobs.....	17
Figura 12. Bill Gates.....	17
Figura 13. Fluxograma .....	22
Figura 14. Cofre .....	25
Figura 15. MySQL .....	25
Figura 16. MariaDB .....	25
Figura 17. Banco de Dados.....	26
Figura 18. Banco de Dados Não- Relacionais.....	26
Figura 19. Estacionamento.....	27
Figura 20. Estacionamento.....	29
Figura 21. Arduino.....	31
Figura 22. Logo Arduino .....	33
Figura 23. Arduino Uno .....	33
Figura 24. IDE Version 1.8 .....	34
Figura 25. Led .....	35
Figura 26. Sistema Hidráulico .....	35
Figura 27. Resistor 220 ohms. ....	36
Figura 28. Resistor 10 Kohms.....	36
Figura 29. Interface TinkerCad.....	37
Figura 30. Tela de login.....	38
Figura 31. Tela geral .....	38
Figura 32. Tela de circuitos .....	39
Figura 33. Transformação de Decimal em Binário .....	40
Figura 34. Função Escreva .....	41
Figura 35. Tela para o Usuário.....	41
Figura 36. Função Leia.....	42
Figura 37. Tela para o Usuário.....	42
Figura 38. Projeto Entrada Porta Serial.....	45
Figura 39. Circuito .....	47
Figura 40. Soma .....	48
Figura 41. Variável Float .....	49
Figura 42. Resto da Divisão .....	49
Figura 43. Condição .....	50
Figura 44. Barra .....	51
Figura 45. Substituição da Barra por I.....	51
Figura 46. Circuito .....	52
Figura 47. Operador Lógico OU .....	52
Figura 48. Operador Lógico E .....	53
Figura 49. Circuito no TinkerCad.....	54
Figura 50. Código IF (Se).....	55
Figura 51. Código ELSE (Senão) .....	55

Figura 52. Estrutura Switch-case .....	56
Figura 53. Código .....	57
Figura 54. Array.....	59
Figura 55. Código.....	59
Figura 56. Código.....	60
Figura 57. Código.....	60
Figura 58. Pista de Aeroporto.....	61
Figura 59. Circuito Pista de Aterrissagem .....	61
Figura 60. Código Pista de Aterrissagem .....	62
Figura 61. Código Identação .....	63
Figura 62. Circuito While (Enquanto/Faça).....	64
Figura 63. Código While (Enquanto/Faça) .....	64
Figura 64. Código Do-While (Faça/Enquanto).....	65
Figura 65. Monitor Serial .....	66
Figura 66. Circuito .....	66
Figura 67. Função Ligar e Desligar .....	67
Figura 68. Parênteses .....	68
Figura 69. Chaves .....	68
Figura 70. Objeto.....	68
Figura 71. Colchetes .....	69
Figura 72. Array em JavaScript .....	69
Figura 73. Circuito Array em JavaScript.....	70



## SUMÁRIO

<b>INTRODUÇÃO .....</b>	<b>10</b>
<b>1. Uma breve história da computação.....</b>	<b>12</b>
1.1 Gerações de computadores .....	16
<b>2. Introdução a Lógica de Programação .....</b>	<b>18</b>
<b>3. Mas afinal, o que são Algoritmos? .....</b>	<b>20</b>
3.1 Algoritmo Computacional .....	20
3.2 Formas de Representação .....	20
3.3 Descrição Narrativa .....	21
3.4 Fluxograma .....	21
3.5 Pseudocódigo. ....	23
<b>4. Informação x Dados x Conhecimento .....</b>	<b>23</b>
<b>5. Banco de Dados.....</b>	<b>25</b>
5.1 Tipos de Dados.....	26
<b>6. Variáveis e Constantes na programação .....</b>	<b>27</b>
6.1 Critérios para criar o nome de uma variável .....	28
<b>7. Microcontroladores .....</b>	<b>32</b>
7.1 Ambiente de Desenvolvimento Integrado .....	34
<b>8. Alguns componentes eletrônicos.....</b>	<b>34</b>
8.1 Botão de pressão ou PUSH BOTTON.....	36
8.2 Simulador Thinkercad.....	37
<b>9. Como o computador entende os Dados? .....</b>	<b>39</b>
9.1 Entrada e Saída de dados.....	41
9.2 Analogia entre linguagens .....	45
<b>10. Operadores.....</b>	<b>46</b>
10.1 Operadores Aritméticos.....	47
10.2 Operadores Lógicos .....	50
10.3 Estruturas de Decisão/Seleção .....	54
10.4 Estruturas de Repetição .....	57
<b>11. O que são Arrays? .....</b>	<b>58</b>
<b>12. O que é Identação? .....</b>	<b>63</b>
<b>13. Funções na Programação .....</b>	<b>66</b>
<b>14. Qual Linguagem de programação devo escolher? .....</b>	<b>70</b>
<b>CONCLUSÃO .....</b>	<b>72</b>
<b>REFERÊNCIAS.....</b>	<b>73</b>

## INTRODUÇÃO

É muito comum aos iniciantes na área de Tecnologia da Informação-TI, ficarem completamente perdidos em relação a que linguagem escolher, qual é a melhor, qual é a mais fácil ou qual é a mais usada no mercado de trabalho atualmente. Digo isto, pois aconteceu comigo e com vários amigos, que aliás desistiram antes mesmo de conhecerem o mais legal da TI, a área de desenvolvimento.

O objetivo deste curso é de facilitar aos egressos na profissão, para que não desistam, pois é o que mais acontece nas Universidades e cursos de tecnologia, sejam eles cursos técnicos ou cursos profissionalizantes, devido a inversão no ensino da Lógica, linguagem, IDE, e um treinamento específico através de projetos e não de um único TCC, visto que o tempo é muito precioso e as tecnologias se tornam defasadas em apenas alguns meses.

Quando fui estudar Engenharia da Computação, fiquei apavorado quando fui apresentado a linguagem JAVA, não por dizerem que ela era difícil, o que não é verdade, mas por não terem ensinado a Lógica de programação primeiro, que é fundamental para qualquer linguagem. Pois bem, estudei a mesma durante um semestre, fui atrás de livros em bibliotecas virtuais e percebi que tinha algo de muito errado naquela metodologia. Como ensinar uma linguagem para um aluno que nunca vira uma IDE na vida? Como iniciar por uma linguagem que era orientada a objetos sem antes ensinar uma linguagem estruturada e a diferença entre as duas? Chegou então o grande e temido dia da prova impressa de JAVA, estudei muito ( e de forma errada ) para tirar a média apenas, que era nota 5. Será se o problema estava em mim?

Os motivos acima me levaram a me aprofundar no assunto, pois apesar de já programar em C, devido ao arduíno que me abriu a mente para a tecnologia, e a desenvolver uma metodologia, sobre o quê estudar primeiro antes de se aventurar num PYTHON por exemplo sem saber de conceitos fundamentais e nos primeiros exercícios desanimar e abandonar o curso. Isto que aconteceu comigo, acontece todos os dias nas Universidades, Faculdades, cursos livres ou em escolas técnicas. Longe de mim de denegrir a imagem de tais escolas/universidades, mas não sou o primeiro e nem serei o último e o objetivo aqui é desmistificar para que haja avanços significativos na sua nova vida profissional.

Espero do profundamente que as teorias aqui apresentadas, que estudei em várias obras referenciadas no final desta obra, sejam de grande proveito para você, e que possamos abrir portas no “labirinto dos ratos”, que nos impuseram quando crianças, dizendo que programação é só para nerds, só para homens,etc...

Na verdade, programação é para quem tem o desejo, para quem tem vontade de aprender, de ajudar as pessoas, solucionar problemas da sociedade.

É bem verdade que, existem altos salários na TI, mas, você não deve escolher esta profissão por este motivo, pois existem grandes desafios também, um deles é o de se manter sempre atualizado em relação as novas tecnologias, pois senão, ficará desatualizado e com certeza logo ficará fora do mercado de trabalho.

Uma dica bem interessante que funciona com com estuda tecnologia é, se você estuda por livros, estude o tema, abra uma IDE e vá praticar, você vai perceber que se esquecer de digitar um ponto e vírgula no seu código, ele vai “bugar”.Outra coisa é a velocidade, quanto mais treinar, mais rápido ficará naquela tecnologia, com certeza.Agora, você que gosta de assistir vídeos do youtube e odeia ler,tem solução; assista aos vídeos, umas cinco aulas por exemplo, um módulo se for curto e, abra o VSCode e vá praticar, pois ninguém se torna programador apenas assistindo vídeos na internet, mas sim, praticando, fazendo projetos, participando de comunidades, estudando documentações em sites oficiais,etc...

Outra dica muito importante, faça todos os exercícios deste livro, pois estão numa ordem lógica de aprendizagem e nunca se contente apenas com isto, invista em você, compre livros, eles são caros dependendo o ponto de vista, veja; se você pagar R\$200,00 por um livro de HTML e CSS você acha um absurdo não é mesmo? Então, de repente isto é um lanche com seus amigos e você não achou caro.Se seu salário hoje for de R\$1800,00 e pretende aumentá-lo para R\$3000,00 por exemplo, saiba que o investimento é muito baixo em relação ao custo-benefício (16% de investimento e retorno de 66,6% aproximadamente).É claro que, isto não acontece de forma instantânea, mas é rápido em se tratando de tecnologia, depende de você exclusivamente.

A última dica é, utilize o método HBC (Horas-Bunda-Cadeira), mas, em estudo e prática efetivamente, sem celular, redes sociais, distrações,etc... Crie um ambiente de estudo favorável, não na cama de forma confortável, pois dará sono.Separe um lugar da

casa com seu computador, fone de ouvido se necessário para entender a semântica e a sintaxe da linguagem. E, lembre-se: “ Não existe cozinheiro sem chegar perto do fogão”, ou seja ; saia dos videos e vá programar!

## 1. Uma breve história da computação

A história da computação inicia com a necessidade do homem de calcular, pois computar significa calcular, contar. Voltando então na história, a computação nasceu então entre 5000 a 7000 anos atrás aproximadamente, com a invenção do **ábaco** pelos mesopotâmios, conforme mostra a figura 1, posteriormente aprimorado pelos chineses e romanos.

Figura 1. Ábaco



Fonte: Google, 2023.

Em 1863, o padre William Oughtred inventou a **régua de cálculos**, que calculava logaritmos e fazia multiplicações e divisões, figura 2.

Figura 2. Régua de Cálculos



Fonte: Google, 2023.

Em 1642, Pascal criou a primeira calculadora mecânica da história, a **Pascaline** (Figura 3). Ela fazia somas e subtrações.

Figura 3. Pascaline



Fonte: Google, 2023.

Em 1831 Joseph M. Jacquard desenvolveu o **tear programável**, para o corte de tecidos de forma automática, figura 4.

Figura 4. Tear Programável

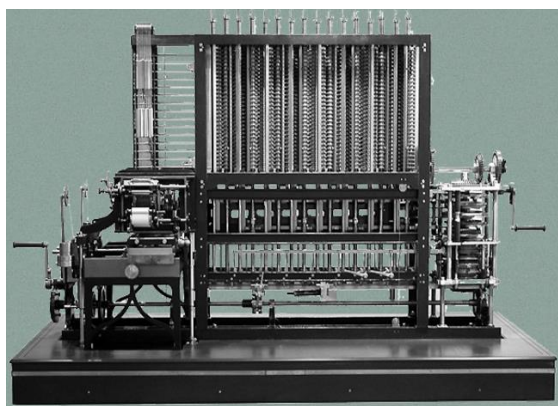


Fonte: Google, 2023.

Em 1801, Charles Babbage criou a **máquina das diferenças**, que resolvia funções de trigonometria, logaritmos e polinômios de forma simples sem energia elétrica, apresentada na figura 5. Aparece então, a primeira mulher na computação, Augusta Ada Byron Lovelace, que encantada pelo Tear programável, é convidada por Babbage para ajudá-lo neste novo projeto. Ada Lovelace se preocupava com a parte da programação

através de símbolos enquanto Babbage com os números e a parte mecânica. Uma coisa interessante, por questões políticas e financeiras, Babbage morreu e seu projeto foi construído posteriormente. Outra curiosidade é que a máquina tinha as características dos computadores atuais; entrada, saída, memória expansível e uma central de processamento, uma tecnologia muito avançada para a época.

Figura 5. Máquina das Diferenças



Fonte: Google, 2023.

Naquela época, a única utilidade que todos viam para o computador, era o de realizar cálculos, mas, Ada Lovelace afirmou que não, o computador daria para trabalhar com símbolos e imagens também, com um propósito geral.

Em 1847, George Boole, desenvolveu a Lógica Moderna ou **Lógica Booleana**, o sistema binário e a tabela-verdade que consiste em apenas dois valores 0 ou 1, que é a forma que o computador entende (figura 6).

Figura 6. George Boole

**George Boole (1815–1864)**



A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1

Fonte: Google, 2023.

Em 1890, seria realizado um censo nos Estados Unidos e Hermann Holerith resolveu desenvolver uma máquina baseada no Tear de Jacquard e na lógica de Boole, deu tão certo que ele criou a Empresa Tabulating Machine Company, que mais tarde se fundiria a mais três empresas e mudaria de nome, passando a se chamar Computing Tabulating Recording Corporation. Hoje conhecida como IBM.

Por volta de 1944 a Universidade de Harvard (EUA) desenvolveram o **Mark I**. fig.7

Figura 7. Mark I



Fonte: Google, 2023.

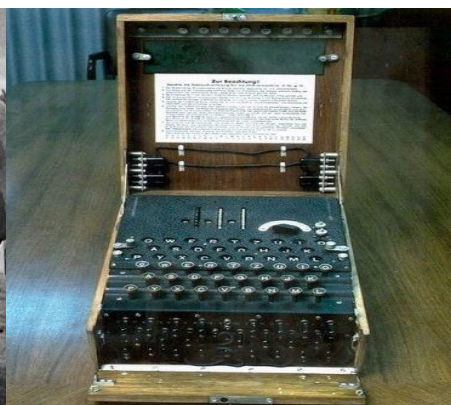
Não poderíamos deixar de citar Alan Turing, que adorava correr fig 8, mas que era um grande matemático que estudava Inteligência Artificial e criptografia, inspirado é claro em John Von Neumann, outro grande matemático. Nesta época, estava acontecendo a 2ª grande guerra mundial (1945) e Turing fora chamado para decifrar a máquina de Hitler, a **Enigma** fig.9. Alan Turing então, que era muito bom em codificar coisas, teria agora que fazer o contrário “descodificar” uma máquina que matava milhões de soldados americanos. Foi então que ele criou uma máquina mais poderosa que a Enigma, denominada de “**A Bomba**” inicialmente e mais tarde rebatizada de **A máquina de Turing** fig.10.

Figura 8. Alan Turing



Fonte: Google, 2023.

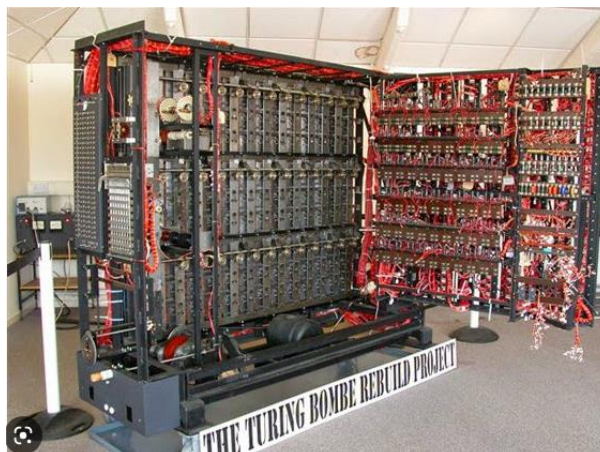
Figura 9. Enigma



Fonte: Google, 2023.



Figura 10. A Máquina de Turing (A Bomba)



Fonte: Google, 2023.

## 1.1 Gerações de computadores

Os computadores estão divididos em quatro gerações, sendo elas:

### 1ª Geração (1946-1959)

Eram gigantescos pesavam 30 toneladas , mediam aproximadamente 25 metros e utilizavam válvulas.

### 2ª Geração (1959-1964)

Com o avanço da tecnologia, surgiu o circuito impresso e o transistor, que reduziu consideravelmente o tamanho dos computadores.

### 3ª Geração (1964-1970)

Surge nesta época o circuito integrado que continha vários transistores embutidos nele. Além do computador diminuir ainda mais de tamanho, ele começa a ficar mais barato.



#### 4ª Geração (1970- até hoje)

Na quarta geração, surgem os microprocessadores que possuem vários circuitos integrados embutidos. Foi desenvolvido o computador Altair que não possuía monitor, nem mouse, pois isso não existia até o momento. Como o lançamento foi um sucesso pois vendeu 10 vezes mais que o esperado, um jovem chamado Bill Gates (Fig 12) resolveu criar uma linguagem para este computador a Altair Basic. Outro jovem chamado Steve Jobs (Fig 11), percebeu que aquele computador não era fácil de ser usado por pessoas comuns e resolveu acrescentar um monitor, para que o usuário enxergasse o que estava acontecendo dentro do computador, deu o nome a ele de APPLE I, por volta de 1979.

Outras empresa fizeram algo parecido mas, sempre incrementando, tais como os computadores LISA (1983) e a MACHINTOSH (1984) que inseriram a interface gráfica atual e o mouse.

O jovem Bill Gates então fundou a Microsoft para papalelo com a APPLE. Como seus programas estavam inferiores aos de Jobs, resolveu fazer uma parceria APPLE-MICROSOFT que seria desfeita em breve e cada um iria para seu lado, mas segundo ele nunca se tornaram inimigos.

Figura 11. Steve Jobs



Figura 12. Bill Gates



## 2. Introdução a Lógica de Programação

Definimos lógica de programação a forma de se ordenar uma sequência de instruções, de forma que faça sentido e que ao final, solucione o problema do cliente. Vejamos na prática como isto funciona.

Carlos está com fome e quer comer miojo, ele vai até o armário e descobre que não tem miojo em casa. Carlos precisa decidir o que fazer, pois a cada minuto, seu estômago “ronca” cada vez mais. Decidido, ele pensa:

Início

- 1) Ir ao mercado;
- 2) Comprar miojo;
- 3) Voltar para casa;
- 4) Colocar 300 ml de água em uma panela para ferver no fogão por 3 minutos;
- 5) Ao iniciar a fervura, abrir o pacote de miojo e colocar na panela;
- 6) Observar o tempo;
- 7) Após 2 minutos, colocar o tempero pronto;
- 8) Aos 3 minutos, desligar e despejar num prato;
- 9) Comer o miojo.

Fim

A questão é a seguinte, esta sequência de instruções, não precisa necessariamente estar nesta ordem. Vejamos esta outra sequência:

Início

- 1) Comprar miojo;
- 2) Ir ao mercado;

- 3) Voltar para casa;
  - 4) Colocar 300 ml de água em uma panela para ferver no fogão por 3 minutos;
  - 5) Ao iniciar a fervura, abrir o pacote de miojo e colocar na panela;
  - 6) Observar o tempo;
  - 7) Após 2 minutos, colocar o tempero pronto;
  - 8) Aos 3 minutos, desligar e despejar num prato;
  - 9) Comer o miojo.
- Fim

Percebemos claramente que algo deu errado neste tipo de pensamento. Existe uma sequência de instruções, claras, porém, estão numa ordem bem confusa.

Primeiro Carlos compra miojo, depois vai ao mercado. Mas, onde ele comprou o miojo se ele foi ao mercado em seguida?

Muito bem, a Lógica de Programação serve exatamente para isto, ordenar de forma clara, e numa ordem que faça total sentido para qualquer um, isso mesmo, eu disse qualquer um, possa realizar a tarefa. Agora, imagine se você não consegue organizar de forma clara, um conjunto de instruções para alguém que nunca fez miojo na vida, se vai conseguir “ensinar” um computador a realizar uma ou mais tarefas. Entende agora o grau de importância de se aprender Lógica de Programação e algoritmos antes de entrar de cabeça em qualquer linguagem de programação?

Existem cursos que ensinam diretamente lógica de programação juntamente com a linguagem de programação. Não que seja proibido, mas veja, o aluno acha que está avançando, mas quando se encontra sozinho para desenvolver uma solução, sem aqueles exercícios prontos, adivinha? Não vai conseguir, e vai ter que voltar e estudar lógica desde o início.

### 3. Mas afinal, o que são Algoritmos?

No módulo anterior estudamos a Lógica de Programação que é organizar um conjunto de instruções de forma que faça sentido. A este conjunto de instruções, damos o nome de algoritmos.

Os algoritmos são utilizados para representar a solução de um problema e são instruções “ensinadas” aos computadores e executadas por eles.

Quando aprendemos a base dos algoritmos, podemos aplicar este conhecimento em qualquer linguagem de programação (C, C#, Java, Javascript, Python, etc...).

Quando temos um problema cotidiano, e conseguimos resolver este problema, através de uma determinada solução, não quer dizer que esta seja a única solução possível. Outra pessoa pode resolver o mesmo problema utilizando outra solução. Essa é a parte mais interessante dos algoritmos pois cada programador pensa de forma diferente e resolverá um mesmo problema com soluções e linguagens de programação diferentes.

#### 3.1 Algoritmo Computacional

Como vimos na história da computação, o computador realiza tarefas, entretanto essas tarefas necessitam de instruções precisas, claras, detalhadas e que ele entenda. Chamamos essas instruções específicas de **Programa**.

#### 3.2 Formas de Representação

Existem três formas de representarmos os algoritmos:

- a) Descrição narrativa;
- b) Fluxograma;
- c) Pseudocódigo

### 3.3 Descrição Narrativa

Na Descrição Narrativa, a sequência de passos é descrita exclusivamente na língua portuguesa. Exemplo:

Média Bimestral de um aluno

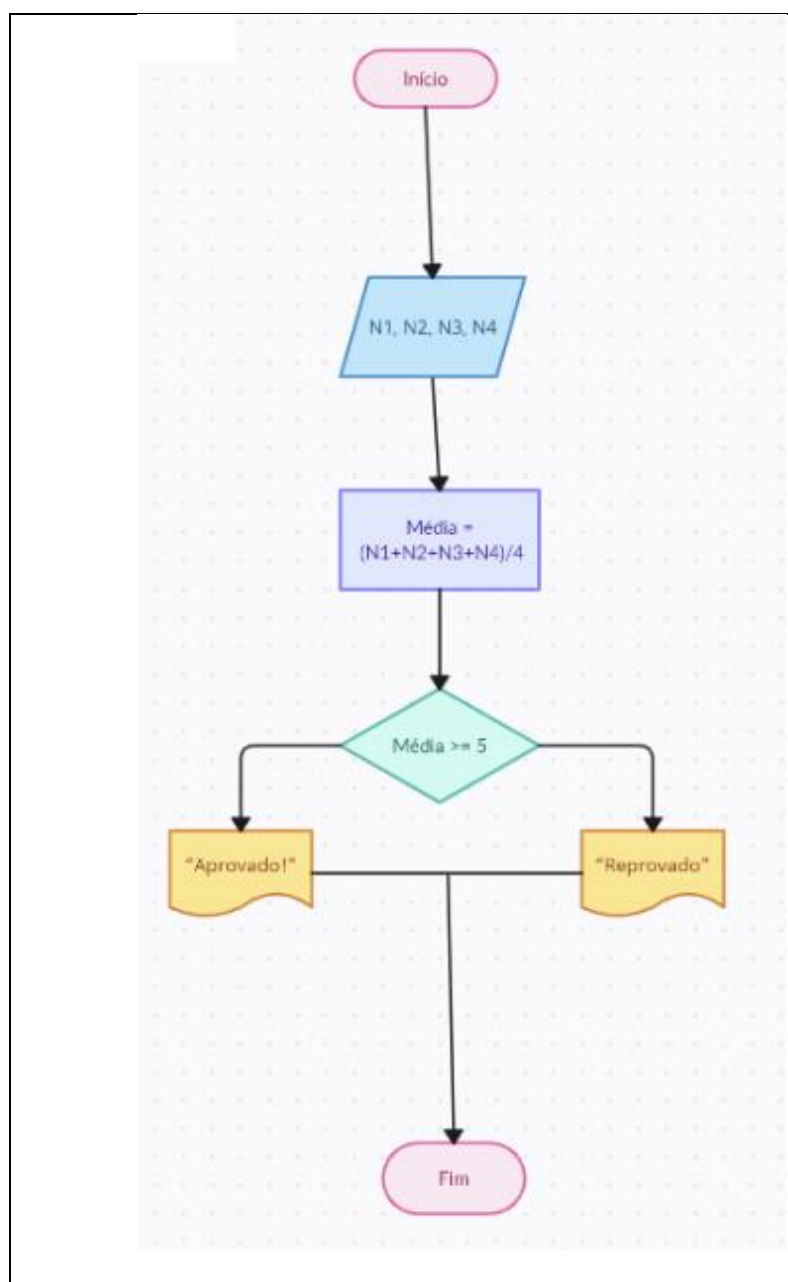
- Digitar as 4 notas de provas;
- Calcular a média aritmética;
- Se a média for maior ou igual a 5, mostrar na tela a mensagem “Aprovado!”;
- Senão, mostrar na tela a mensagem “Reprovado”.

### 3.4 Fluxograma

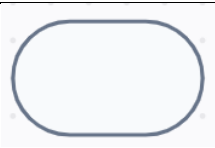

O Fluxograma é a representação gráfica dos algoritmos utilizando formas geométricas. Os símbolos que vamos dar como exemplo abaixo, variam de autor para autor. Outro aspecto importante sobre fluxogramas, é que a visualização e entendimento é bem melhor do que a Descrição Narrativa, entretanto, o uso deles é bem interessante e eficaz quando trabalhamos com pequenos programas, dificultando quando o programa possui muitas linhas de código.


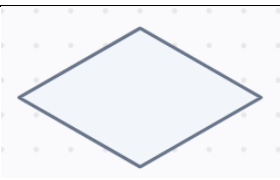
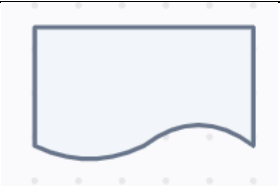
Exemplo:

Figura 13. Fluxograma



Fonte: Autor, 2023.

Formas Geométricas Específicas	
	Início e Fim do Fluxograma
	Entrada de Dados

	Saída de Dados
	Atribuição
	Decisão

### 3.5 Pseudocódigo.

O pseudocódigo é uma forma natural de escrevermos nosso código, se você quiser se aprofundar neste assunto, visite: <https://www.youtube.com/watch?v=6OIADpFlmtc> que ensina sobre o compilador Portugol.

## 4. Informação x Dados x Conhecimento

Os **dados** são elementos a serem armazenados, tratados e processados.

Exemplos:

O rg de uma pessoa :20.554.512-4;

O nome de um funcionário(a): 'Marcela';

O endereço do funcionário(a): 'rua Sabiá';

A profissão de um funcionário na empresa: 'Arquiteto'.

Temperaturas mínima e máxima em Amparo nesta semana: 9° e 23°.

Possibilidade de chuvas em Amparo daqui a três dias: 89%.

Já a **informação** é o conjunto estruturado e organizado dos dados, ou seja, um depende do outro.

Exemplo:

Funcionário:Marcela

Rg: 20.554.512-4

Endereço: Rua Sabiá

Profissão: arquiteta

**Conhecimento** por sua vez, é a aplicação prática da informação por uma pessoa, após o processamento da mesma.

Marcela pretende lavar roupas nesta semana, ainda não tem dia certo, mas tem muita roupa e sua lavanderia é descoberta.

Estava “mexendo” no celular, viu exatamente na hora em que o ligou que estava 18° de temperatura, mas não ligou muito.Ao continuar pesquisando no Instagram, Facebook, chegou num site de notícias locais que tinha a seguinte informação:

Temperaturas mínima e máxima em Amparo nesta semana: 9° e 23°.

Possibilidade de chuvas em Amparo daqui a três dias: 89%.

Marcela então pensou que se iria chover daqui a três dias, ela teria que tomar uma decisão rápida, ou esperaria passar as chuvas para lavar suas roupas, ou lavá-las imediatamente para que desse tempo de secá-las.Foi o que ela fez, lavou tudo e resolveu seu problema.

Marcela viu os **dados**, no celular, mas como estes estavam soltos, desconectados, não faziam muito sentido.Em seguida ela viu a **Informação**, mas de nada adianta ter acesso a mesma se não houver uma ação, a aplicação prática daquilo em sua vida; lavar a roupa suja.Este último é o **Conhecimento**.



## 5. Banco de Dados

Para guardamos valores, dinheiro, documentos, escrituras, diamantes,ouro, devemos ter um local seguro, idôneo e não corrompido.Este lugar, que é chamado de cofre (Fig 14), deve ter uma senha que somente o dono daquela conta possa acessar, e mesmo que este cofre ou conta corrente esteja em um Banco particular ou público, os gerenciadores do mesmo não podem ter acesso a esta senha, somente aos valores que ali se encontram.

Figura 14. Cofre



Fonte: <https://pt.dreamstime.com>, 2023.

Muito bem, no mundo digital os **dados** são estes valores, e seguem as mesmas regras, devem ser armazenados em um local seguro.Este local se chama Banco de Dados.

Existem diversos Sistemas que gerenciam os Bancos de Dados, estes sistemas tem o nome de Sistema de Gerenciamento de Banco de Dados-**SGBD**.

Alguns exemplos de SGBD são mostrados nas Figuras 15, 16 e 17 logo abaixo:

Figura 15. MySQL



Fonte: Google, 2023

Figura 16. MariaDB



Fonte: Google, 2023

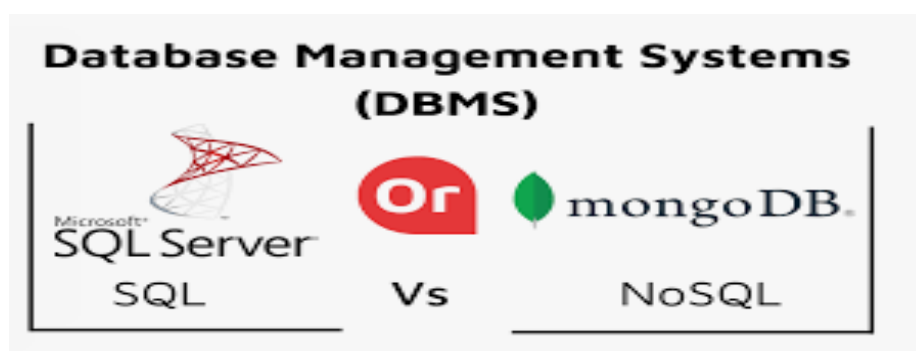
Figura 17. Banco de Dados



Fonte: Google, 2023

Os SGBDs são divididos em duas classes, os **Relacionais** (Sql) e os **Não-Relacionais**(NoSql) (Fig 18.). O Banco de Dados que iremos utilizar aqui no nosso curso é o Mongo DB (NoSql) e lá aprofundaremos mais este assunto.

Figura 18. Banco de Dados Não- Relacionais



Fonte: Google, 2023

## 5.1 Tipos de Dados

Dados são as informações tratadas e armazenadas no computador. Existem 3 tipos de dados:

- Literais;
- Numéricos;
- Lógicos.

Os **dados Literais** são as letras, dígitos e caracteres especiais.

Exemplos: **string** ou **char** dependendo da Linguagem de programação

Os **dados Numéricos** são os números que podem ser o conjunto dos números inteiros ( negativos,zero,positivos) ou o conjunto dos números Reais ( que engloba números naturais, inteiros, decimais, frações, etc...). Exemplos: **int, float, const.**

Os **dados Lógicos** são usados para **Verdadeiro ou Falso.**

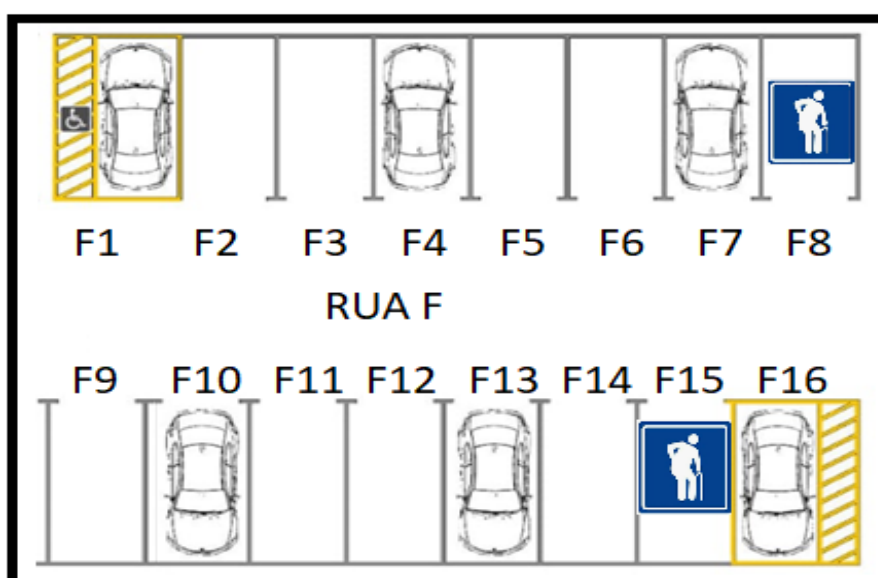
## 6. Variáveis e Constantes na programação

**Variáveis** são espaços reservados na memória do computador. Este espaço reservado tem um endereço e pode ser alterado durante a execução do programa, ou seja, pode variar.

Já a **Constante**, é um espaço reservado na memória, possui um endereço mas ao contrário da variável, não pode ser alterada durante a execução do programa, será como o próprio nome diz, constante até o final.

Imaginemos um estacionamento com cada vaga demarcada e com o nome de cada rua conforme a Figura 19 abaixo.

Figura 19. Estacionamento



Fonte: Autor, 2023

Todas as vagas são para carros, porém existem duas vagas destinadas a idosos e duas vagas destinadas a deficientes físicos, que são os tipos de vagas que são reservadas

por lei a estas pessoas. Outra coisa a ser levada em consideração é que estas vagas são para veículos pequenos, não cabendo então uma caminhonete, uma moto, um ônibus ou até mesmo um caminhão. Estes, exigem **outro tipo de vaga**.

Analogamente é a memória do computador, quando criamos uma variável, estamos reservando um espaço para guardar ali dentro algum dado.

As variáveis devem ter:

- a. Um nome;
- b. O tipo de dado;
- c. A Informação.

### 6.1 Critérios para criar o nome de uma variável

Existem alguns critérios universais para a criação do nome uma variável, vejamos quais são eles.

- a. Os nomes devem começar sempre por uma letra.

Exemplos:

Media, Telefone. telefone, salario.

- b. **Não** devem conter caracteres especiais.

Exemplos:

%porcentagem, &media, s@lario.

- c. **Não** devem conter espaços em branco.

Exemplos:

Data de nascimento, media de salario, endereco do cliente.

Correto: data\_de\_nascimento, media\_salario, end\_cliente.

- d. **Não** devem conter hífen entre as palavras. No lugar do hífen devemos utilizar underline.

Exemplos:

Data-de-nascimento, media-salario.

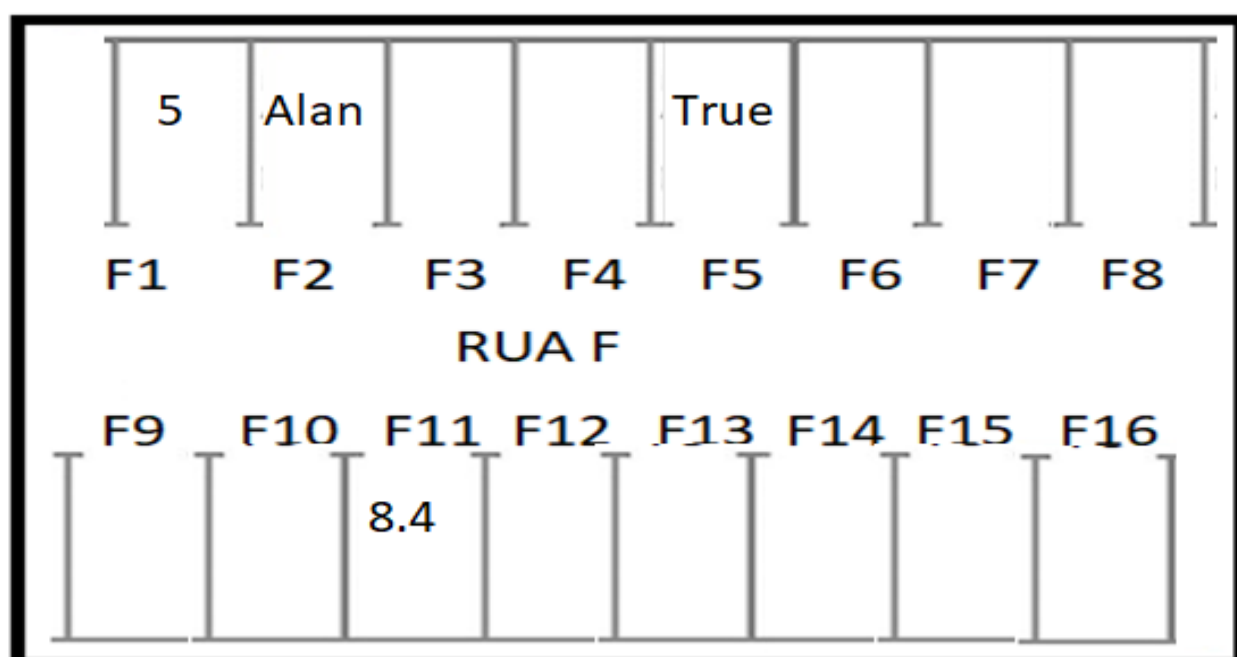
Correto: data\_nascimento, media\_salario.

Obs:As Linguagens de programação são sensíveis (case-sensitive), ou seja, se você colocar o nome de uma variável de media, com “m” minúsculo , quando você chamar esta variável por Media com “M” maiúsculo, o computador não entenderá, pois se trata de outra variável.

e. O  **sinal de igual (= )** em Linguagem de programação é utilizado para **atribuir um valor a uma variável**(colocar um dado dentro dela ).Quando precisamos utilizar o sinal de igual em operações, usamos o igual dobrado (==).Não confunda.

Observe na Figura 20 abaixo do estacionamento e imagine agora que no lugar dos carros, colocaremos alguns dados.

Figura 20. Estacionamento



Fonte: Autor, 2023

Preciso colocar o número 5 dentro da variável F1.Ela deve ter um nome, o tipo e a informação correto?

1 int F1 = 5 ;

O número 5 é do tipo inteiro, portanto na **sintaxe** ( falaremos sobre este assunto mais a frente) desta Linguagem de Programação inteiro é representado por **int**.O nome

desta variável é **F1** e a informação a ser armazenada é o próprio número **5**.

Neste caso vemos: A variável F1 recebe (=) o número 5.

Vejamos como serão guardados os outros valores:

Alan, True (verdadeiro), 8.4

Perceba que Alan é um nome formado por caracteres, então será do tipo **string**.

True que significa verdadeiro, é do tipo **lógico (booleano)**

A variável F11 receberá um número que não é inteiro, este número é racional, um número com vírgula. Neste caso e nesta linguagem utilizaremos o **float** para identificar o seu tipo.

2 string F2 = "Abner"; // ( aqui pode ser aspas simples ou duplas dependendo da linguagem)

3 bool F5 = TRUE;

4 float F11 = 8.4;

Viu como não é tão difícil, é só "avisar" o computador o tipo de dado, utilizar o código correto (sintaxe) e utilizar a lógica que dará certo.

Você deve ter percebido que foram utilizados o ponto e virgula (;) ao final de cada linha, um número no início de cada linha que representa a própria linha para que possamos nos localizar se houver algum erro, e que no comentário da linha 2, eu digitei barra barra (//), que serve exatamente para este fim se quisermos comentar algo naquela linha sem alterar o programa.

Se quisermos comentar algo mais extenso, devemos utilizar barra-asterico, asterisco- barra ( /\* \*/ ). Exemplo de código com comentários:

```
/* Ligar Led com botão
Autor: Abner da C. Ferreira
Data: 12/03/2015
*/

int led_A = 11; // led ligado na porta digital 11

int botao_A = 9; // botão conectado ao pino digital 9
```

```

int val = 0;    // variável para guardar o valor lido

void setup() {

    pinMode(led_A, OUTPUT); // determina a porta digital 11 como saída

    pinMode(botao_A, INPUT); // determina a porta digital 9 como entrada

}

void loop() {

    val = digitalRead(botao_A); // lê o botão A

    digitalWrite(led_A, val); // liga o LED depois de ter lido o botão A

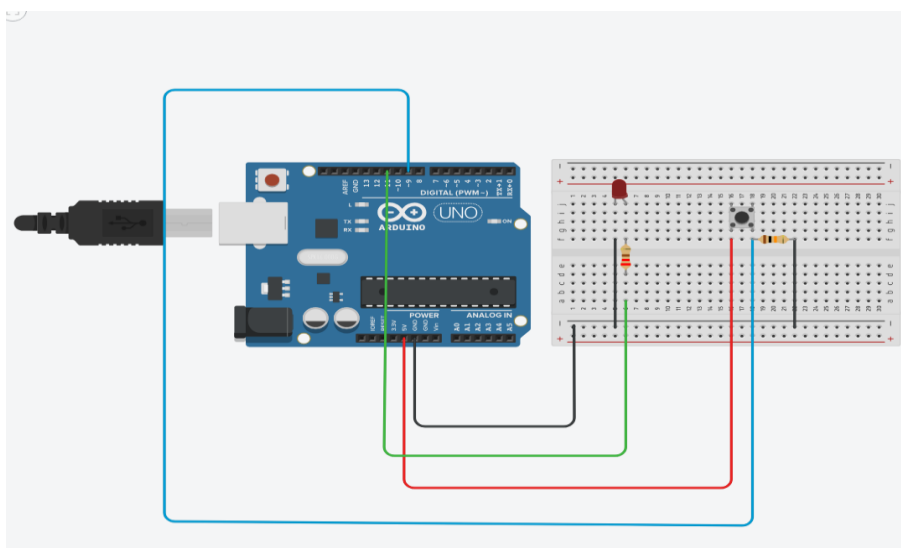
}

```

Todos os comentários que estão na cor azul, não irão interferir de nenhuma forma no código, pois na hora de rodar o programa irá “pular” tais comentários.

Segue abaixo o esquema de ligação no **arduino** na Figura 21 do nosso código :

Figura 21. Arduino



Fonte: Autor, 2023.

Agora que já entendemos o que são dados, variáveis, constante e seus tipos, vamos conversar um pouco sobre microcontroladores.

## 7. Microcontroladores

Mas você deve estar se perguntando porque estamos estudando microcontroladores num curso de Linguagem de Programação, Lógica, Node, etc?

Então, quando fui estudar Linguagem de Programação tive muita dificuldade pois, tudo era muito abstrato, ou pelo menos quase tudo, tinha uns comandos meio malucos e esse tal de “imprima na tela”, “escreva na tela”, eu ficava meio perdido. Não consegui avançar muito pois a **didática** (forma de ensinar) não era boa. Fiz um curso do SENAI de eletricista de manutenção em 1998 que me ajudou e me ajuda até hoje na compreensão de lógica de programação e de IoT (Internet Of Things-Internet das coisas), o que é um circuito fechado, interruptores, motores, etc... Mas o “pulo do gato” veio em 2005 quando criaram um negócio na Itália chamado **Arduino** (Figuras 22 e 23), uma plataforma robusta, reprogramável e que se ligava a outros módulos, de código aberto (open source) que permite não somente que você copie os códigos pois ele é livre para ser estudado, mas também os criadores permitem que você copie e fabrique seu próprio arduino (só não pode usar a marca registrada do infinito), de resto, você pode brincar a vontade usando a imaginação. Eu particularmente conheci o arduino em meados de 2013, quando estudava robótica educacional através do Lego Mindstorms, que é excelente, mas muito cara. Então como ensinar robótica nos colégios particulares se a plataforma Lego custava R\$3000,00 aproximadamente e teriam que ser vários kits. E na escola pública, quem pagaria por isto? Os alunos? O professor? O Estado? O dono do colégio?

Pensando nestas perguntas e como acabara de conhecer o arduino, juntando meus conhecimentos de Física, Eletricista, Eletrônico e Matemático, resolvi me aprofundar neste universo de faça você mesmo (que hoje é conhecido como Movimento Maker) e tomei ali uma decisão, ensinar robótica, programação e princípios de eletrônica na rede pública sozinho assim que tivesse oportunidade. Esta oportunidade veio quando me mudei de Guarulhos para Amparo interior de São Paulo, onde meu projeto foi aceito e leciono robótica e matemática em um colegio de tempo integral.

O site oficial do arduino é **arduino.cc**, lá você pode estudar e se aprofundar ainda mais pois existe uma documentação imensa. Claro que vou deixar também os melhores canais do youtube como indicação, mas saibam que estudaram por lá.



Percebi que quanto mais eu estudava programação pelo arduino que é bastante concreto, pois é visual, eu vejo claramente o que está acontecendo quando eu programo, que eu entendia melhor a Lógica de Programação. Então comecei a associar por analogia a Linguagem de Programação do arduino, que é baseada em C e C++ com uma outra Linguagem que aliás é muito temida por muitos programadores, mas, que é mal compreendida pois faltam conceitos básicos de Lógica de Programação. Sim, estou falando do JAVA.

E é por este motivo que vamos iniciar nossos estudos em Lógica de Programação, fazendo, programando, acertando, errando, corrigindo e copiando, sim, pois no início é um tal de copia e cola da internet que vou te falar, mas, quem nunca, não é mesmo?

O problema é que você deve aprender a programar utilizando a sintaxe correta da linguagem (o código certo), pois cada linguagem tem a sua própria sintaxe.

Os microcontroladores são dispositivos pequenos, plataformas que podem ser programadas e reprogramadas, contém um chip e memória. Possuem também as portas de entrada e de saída que são as mesmas. Quando queremos que uma porta ou pino seja de entrada, programamos esta como **INPUT** e quando queremos que ela seja saída, como **OUTPUT**. Ele tem uma filosofia muito importante que é chamada de **OPEN SOURCE**, ou seja, de Código Aberto. Isto quer dizer que você pode copiar ou melhorar qualquer código que esteja tanto na plataforma oficial quanto na internet pois é autorizado, inclusive copiar e reproduzir o hardware, isto mesmo copiar o arduino. Mas, tem uma ressalva, a marca registrada abaixo não pode ser copiada e reproduzida de forma alguma, ok?

Este dispositivo funciona com uma tensão de entrada entre 5V a 12V, mas o ideal mesmo é 9V, para trabalhar bacaninha.

Figura 22. Logo Arduino



Fonte: Google, 2023

Figura 23. Arduino Uno

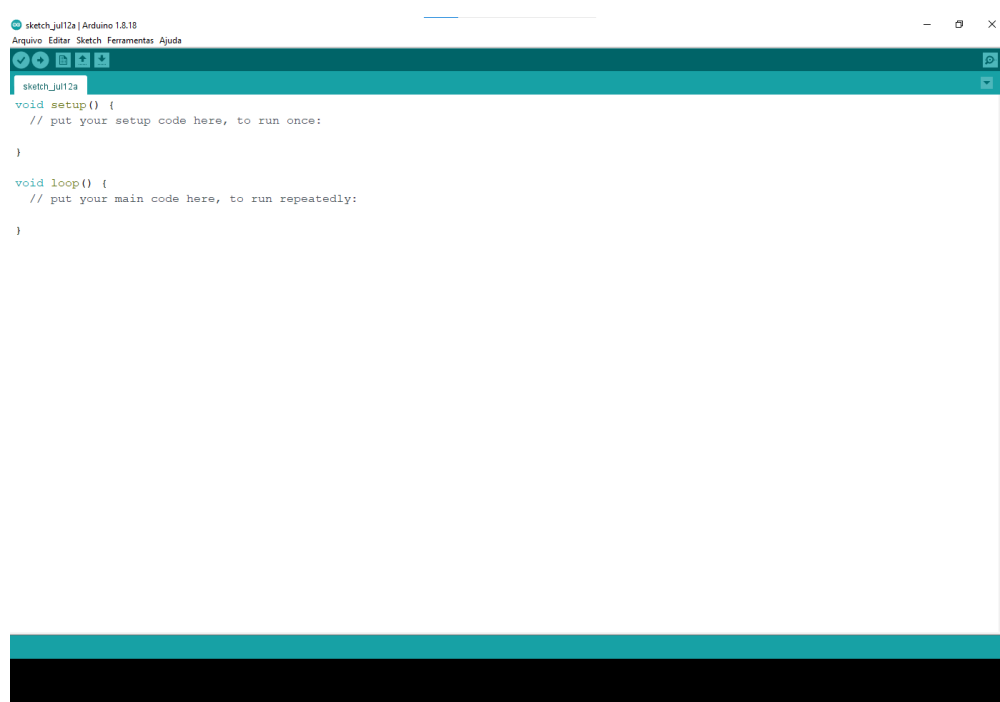


Fonte: UsinaInfo

## 7.1 Ambiente de Desenvolvimento Integrado

Toda Linguagem de programação deve ser escrita em algum lugar concorda? Pois então, este lugar se cham Ambiente de Desenvolvimento Integrado a **IDE** (*Integrate Development Integrate*) (Fig 24). Na IDE você consegue desenvolver tudo pois existem vários recursos para auxiliá-lo, inclusive você baixa gratuitamente no site [arduino.cc](https://www.arduino.cc). Um dos recursos é escolher qual arduino será utilizado, fazer download, fazer upload, escrever código, compilar código e etc...

Figura 24. IDE Version 1.8



Fonte: Autor, 2023.

## 8. Alguns componentes eletrônicos

Nesta apostila faremos algumas montagens utilizando não somente o arduino, mas também leds, resistores e botões (push-botton). Cada um deles tem sua função específica.

O Led (Fig 25.) é uma pequena lâmpada que de tensão que varia de 1,5V a 3V, dependendo de sua cor é claro.

Figura 25. Led

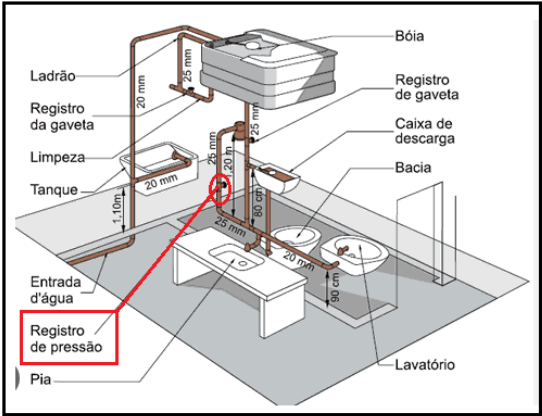


LEDs		
Cor do LED	Tensão em Volts (V)	Corrente em Milliampères (mA)
Vermelho	1,8V - 2,0V	20 mA
Amarelo	1,8V - 2,0V	20 mA
Laranja	1,8V - 2,0V	20 mA
Verde	2,0V - 2,5V	20 mA
Azul	2,5V - 3,0V	20 mA
Branco	2,5V - 3,0V	20 mA

Fonte: Google, 2023.

Já os resistores funcionam como “ redutores” de tensão/corrente para não queimarmos nossos leds, pois em cada porta do arduino sai 5 volts.O resistor então que utilizaremos muito aqui é chamado de resistor do Led (Fig 26.), varia entre **220 ohms e 330 ohms**.

Figura 26. Sistema Hidráulico

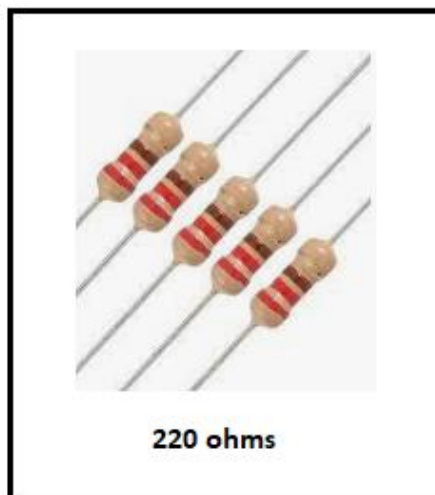


Fonte: Google, 2023.

O registro de água serve para abrir ou fechar a água, mas também conseguimos diminuir a pressão da água se estiver muito forte.

O resistor faz exatamente isto, porém ele possui um valor fixo.

Figura 27. Resistor 220 ohms.

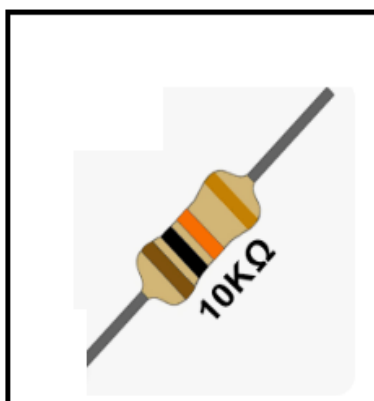


Fonte: Google, 2023.

### 8.1 Botão de pressão ou PUSH BOTTON

Temos outro componente que será usado , o nome dele é push bottton ou simplesmente botão de pressão. Ele também necessita de um resistor para ser ligado ao arduino. Este resistor (Fig 28.) tem um valor de 10Kohms, isso mesmo 10 000 ohms. Portanto, muita atenção, pois as vezes seu código não funcionará pois utilizou o resistor errado, 100Kohms por exemplo no lugar de 10Kohms.

Figura 28. Resistor 10 Kohms



Fonte: Google, 2023.

## 8.2 Simulador Thinkercad

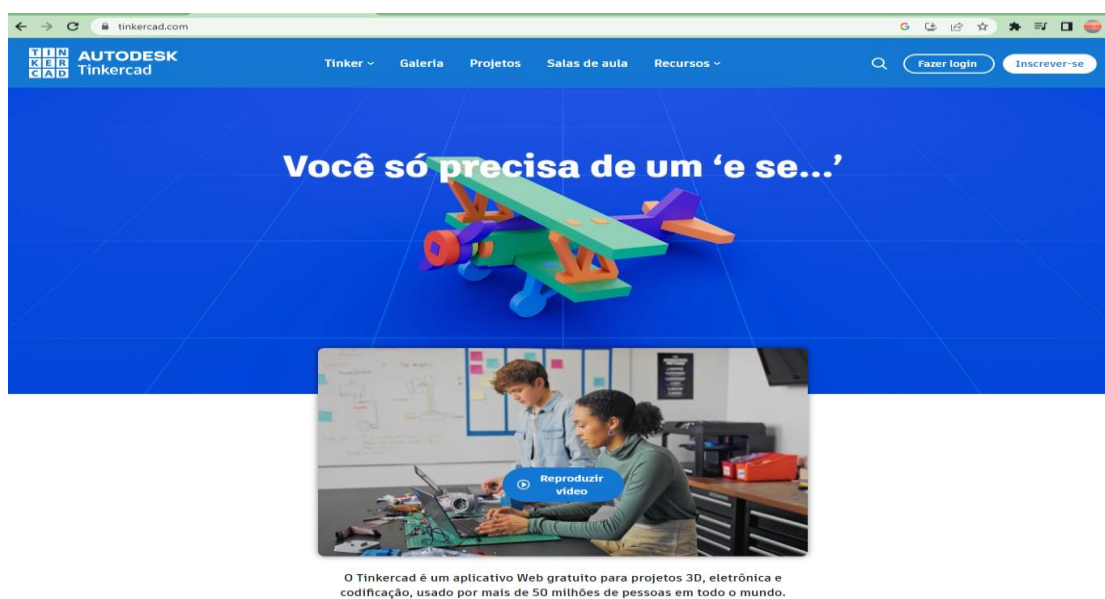
O Thinkercad é uma ferramenta muito útil a qualquer pessoa que deseja desenvolver algum projeto na área de programação, circuitos eletrônicos ou até mesmo criar uma modelagem para uma impressora 3D.

O interessante desta ferramenta, é que como a maioria dos componentes eletrônicos são muito caros e nem sempre temos eles a mão, podemos criar a vontade, testar sem nos preocuparmos de não queimar nada.

O site desta ferramenta é <https://www.tinkercad.com>, lá você se cadastra com um email, desenvolve seus projetos e olha que legal, eles ficam salvos lá para uso futuro.

Vejamos sua interface na Figura 29 abaixo:

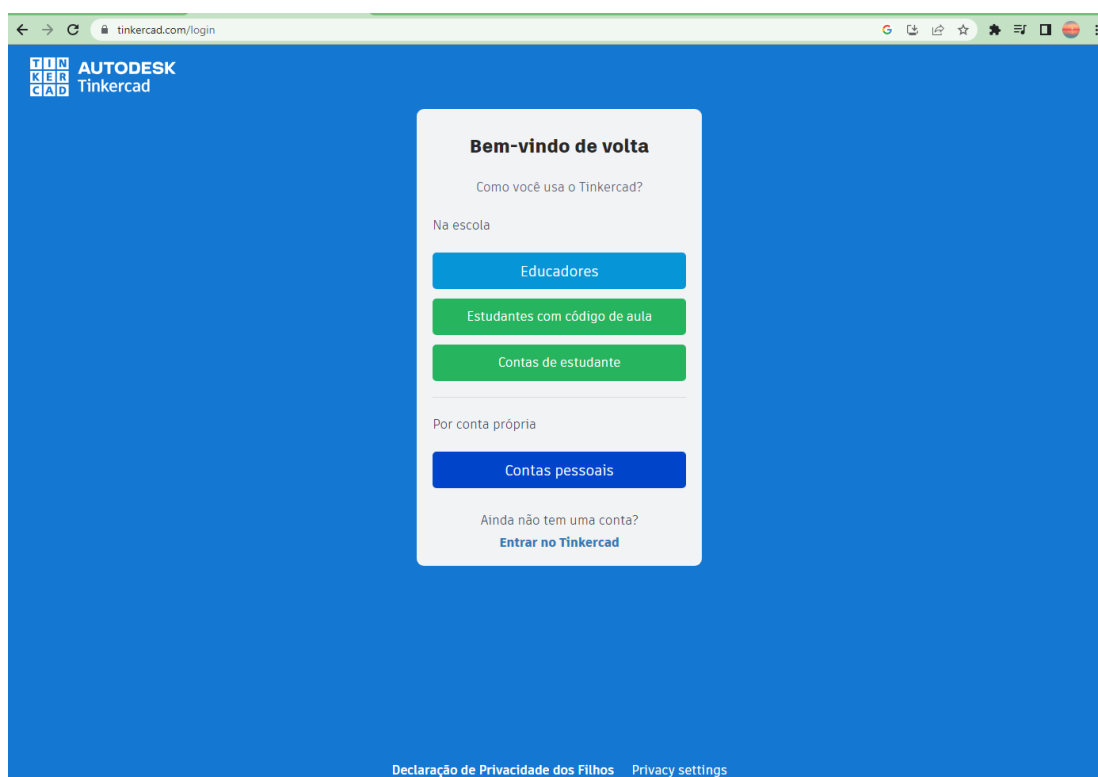
Figura 29. Interface TinkerCad



Fonte: Autor, 2023.

Abaixo está a Tela de Login na Figura 30:

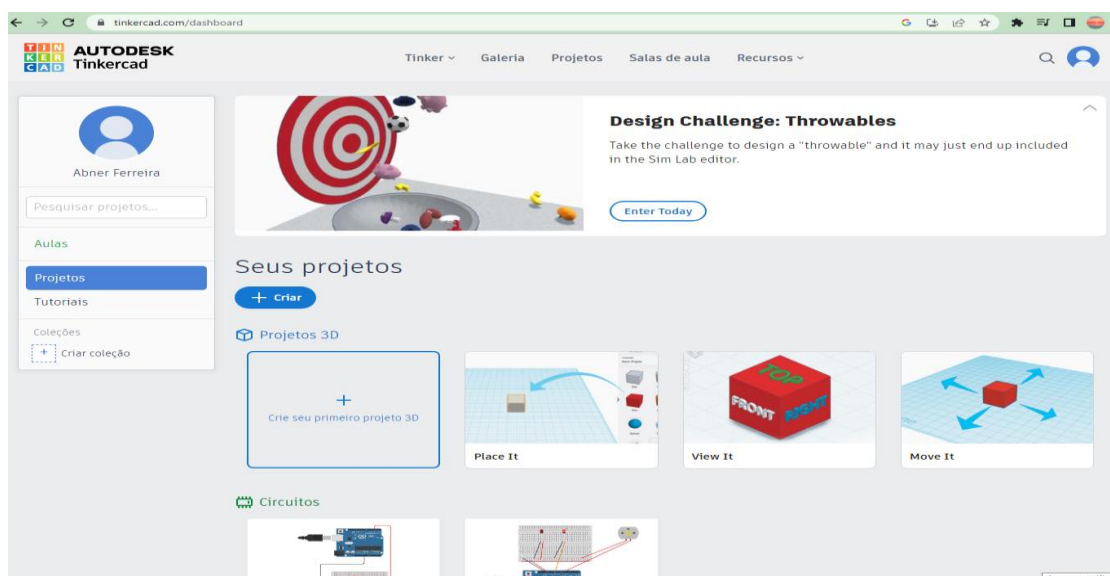
Figura 30. Tela de login



Fonte: Autor, 2023.

Em seguida é apresentada a Tela Geral na Figura 31 abaixo:

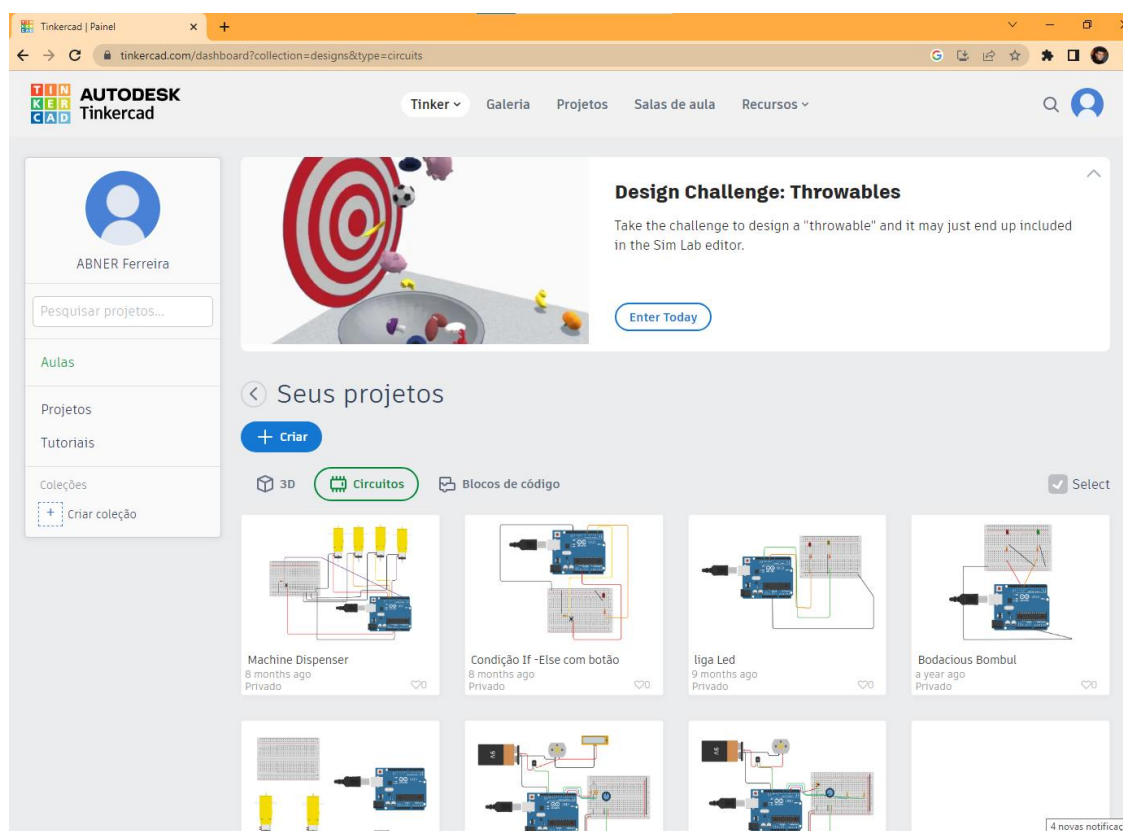
Figura 31. Tela geral



Fonte: Autor, 2023.

E por fim visualizamos a Tela de Circuitos como exibida na Figura 32:

Figura 32.Tela de circuitos



Fonte: Autor, 2023.

Agora, que você já conhece o simulador, vamos realizar o seu cadastro.

## 9. Como o computador entende os Dados?

O computador só entende números binários, que é uma linguagem de baixo nível. Isto quer dizer que se digitarmos o número 47 (sistema decimal), o computador vai entender **101111**, pois serão realizadas divisões sucessivas pelo número dois, quando o resto der zero você para e pega todos os restos da direita para a esquerda “subindo”, como mostrado na Figura 33 do exemplo abaixo.

Figura 33. Transformação de Decimal em Binário

$$\begin{array}{r}
 47 \overline{) 2} \\
 1^{\circ} \text{ resto } \text{---} \textcircled{1} \quad 23 \overline{) 2} \\
 2^{\circ} \text{ resto } \text{---} \textcircled{1} \quad 11 \overline{) 2} \\
 3^{\circ} \text{ resto } \text{---} \textcircled{1} \quad 5 \overline{) 2} \\
 4^{\circ} \text{ resto } \text{---} \textcircled{1} \quad 2 \overline{) 2} \\
 5^{\circ} \text{ resto } \text{---} \textcircled{0} \textcircled{1} \text{---} \text{Último quociente}
 \end{array}$$

Fonte: Google, 2023.

Se quisermos transformar um número binário em sistema decimal, o que fazer?

Por exemplo o número 10100?

Faça uma tabela e escreva/digite o binário em cada quadradinho na segunda linha. Em seguida, da direita para a esquerda, escreva na primeira linha. Eleve o primeiro quadradinho ao número 1, o segundo ao número 2 e assim sucessivamente até o último quadradinho. Resolva as potências somente dos binários que o número é 1. Some tudo e encontrará o resultado.

$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
1	0	1	0	0
16	não	4	não	não

$$16 + 4 = 20$$

O número procurado é 20.



## 9.1 Entrada e Saída de dados

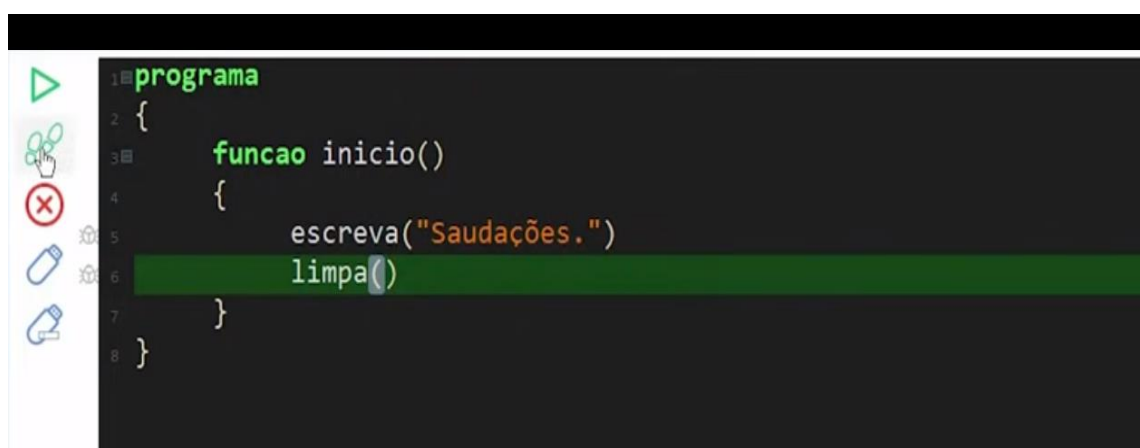
Como você pode ver, o computador só entende números binários, ou seja 0 ou 1, Alto ou Baixo, HIGH ou LOW, ligado ou desligado e, funciona basicamente da seguinte forma; Entrada de dados, processamento destes dados e saída dos mesmos.

No pseudocódigo existem duas formas disto acontecer; o Leia e o Escreva.

O Escreva é quando algo é mostrado(impresso) na tela(monitor) ao usuário.

Exemplo de função escreva na Figura 34:

Figura 34. Função Escreva



Fonte: Autor, 2023.

O que será mostrado na tela para o usuário (Fig 35.):

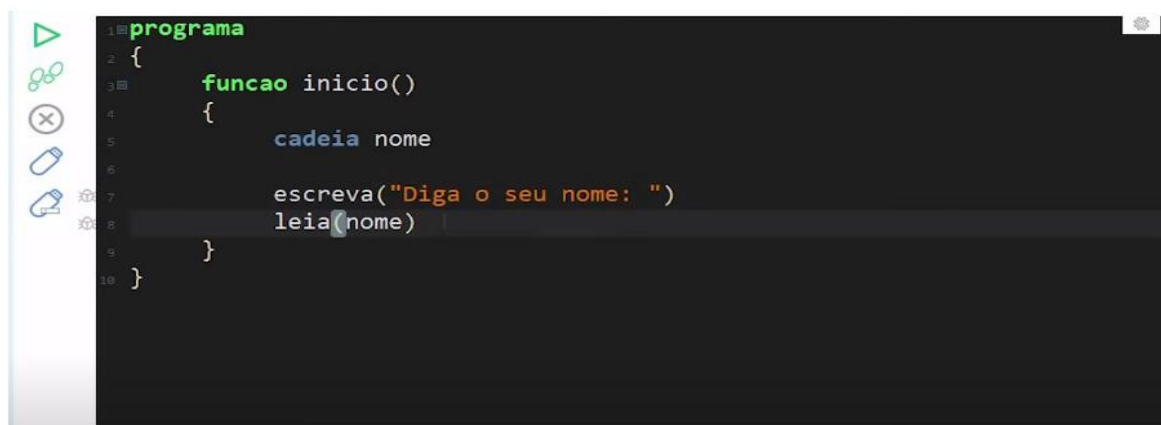
Figura 35. Tela para o Usuário



Fonte: Autor, 2023.

Exemplo de função Leia na Figura 36:

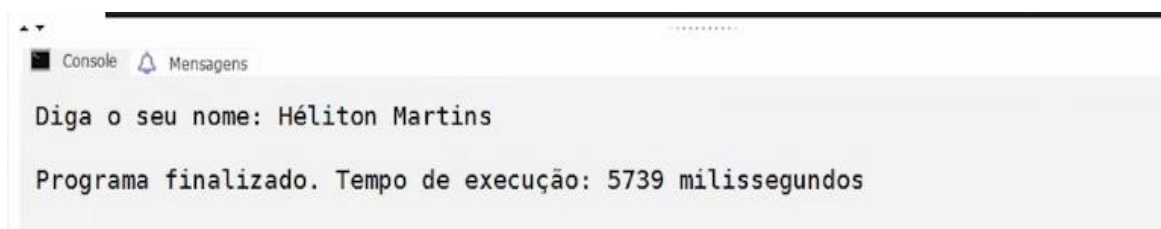
Figura 36. Função Leia



Fonte: Autor, 2023.

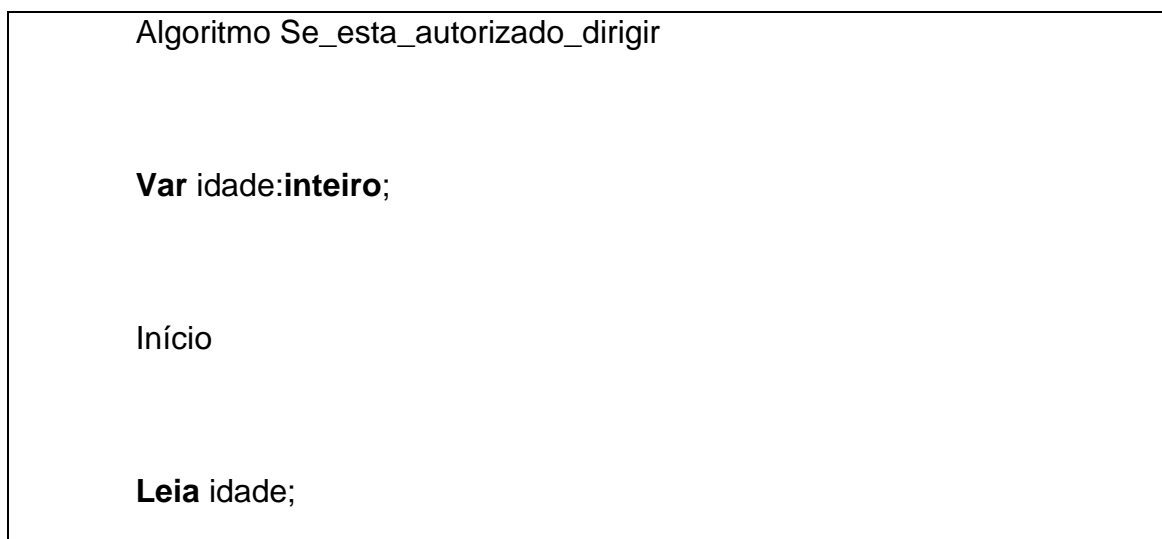
Veja o que aparece na tela para o usuário (Fig 37.):

Figura 37. Tela para o Usuário



Fonte: Autor, 2023.

Outro exemplo:



```
If idade >=18
```

```
  Escreva “ Pode seguir viagem”;
```

```
Else
```

```
  Escreva “ Terei que apreender seu veículo e chamar seus pais”;
```

```
Fim
```

No arduino, podemos utilizar uma tela para “conversarmos” com o programa, ou seja entrar com os dados mais exatamente. O nome desta tela é **Monitor Serial**.

Vamos entrar agora no simulador Thinkercad.

No projeto da Figura 38 abaixo, vamos ligar um led ao arduino e todas as vezes que digitarmos o número 1, o programa entenderá que é para ligar o mesmo, caso contrário, se eu digitar o número 0, o led será desligado.

É um programa muito simples, porém estes conceitos serão muito importantes futuramente e não podemos pular de forma alguma.

```
/*
```

```
  Entrada de dados no arduino pela porta serial
```

```
  Autor: Abner da Conceição Ferreira
```

```
  data:13/08/2014
```

```
*/
```

```
//declarando a variável led na porta 3
```

```
#define led 3
```

```
int valor_lido; // valor lido será um número inteiro teclado
```

```
void setup()//neste bloco definimos quem vai fazer o quê

{

    Serial.begin(9600);

    pinMode(led, OUTPUT);

}


void loop() /*neste bloco acontece

a repetição, os comandos automatizados*/

{

    if(Serial.available()>0) /*aqui é analisado se

    existe um valor digitado maior que 0 */

    {

        valor_lido = Serial.read (); //aqui é feita a entrada dos dados

    }

    if (valor_lido == '1') // se o valor for 1

    {

        digitalWrite(led, HIGH); // o led acende

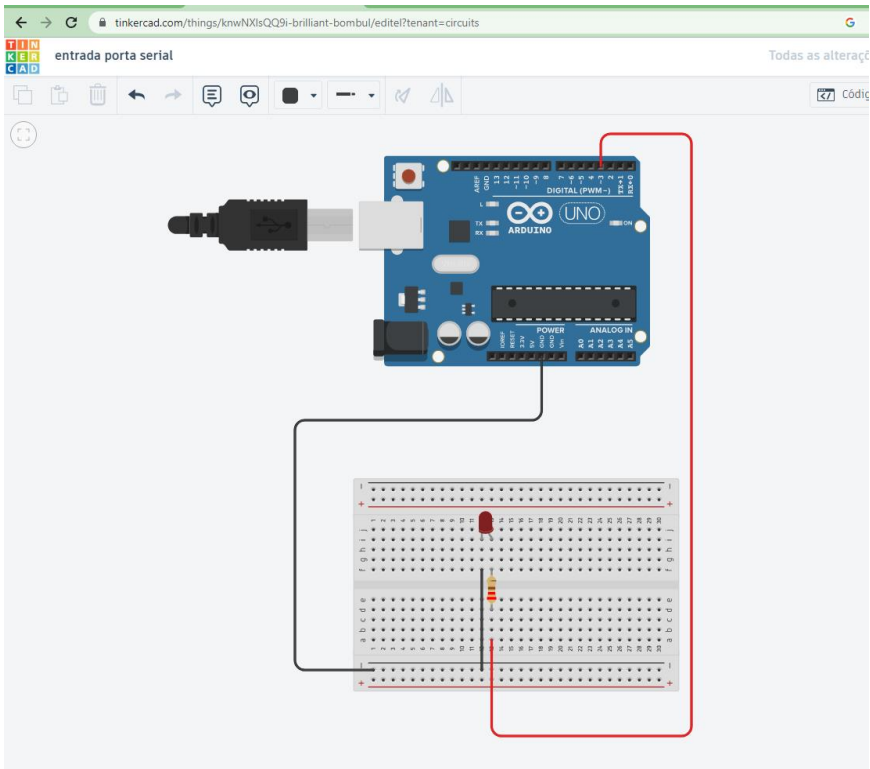
    } else{ // senão

        digitalWrite(led, LOW); // o led apaga

    }

}
```

Figura 38. Projeto Entrada Porta Serial



Fonte: Autor, 2023.

OBS: Perceba o uso de comentários ( // ou /\* \*/) como é de extrema importância, se daqui a 3 anos você precisar abrir este código e realizar uma implementação, se estiver comentado (documentado) , você não terá problemas e se lembrará de absolutamente tudo.Entretanto , se tiver preguiça e não comentar nada, confiando na sua memória, terá sérios problemas e seu trabalho será enorme.

9.2 Analogia entre linguagens

Linguagem	Ação a ser executada	
(*)Pseudocódigo	LEIA	ESCREVA
C	scanf	printf

Arduino	Serial.read  digitalRead(na porta digital)  analogRead(na porta analógica)	Serial.println (no monitor serial)  digitalWrite (na porta digital)  analogWrite (na porta analogica)
Javascript	prompt  input	alert  console.log

(\*)Lembrando que pseudocódigo não é uma linguagem de programação, e sim, um método de se escrever um código numa linguagem natural de fácil entendimento.

## 10. Operadores

Operadores são diversos símbolos utilizados nas Linguagens de Programação que representam cálculos, ordem dos dados e atribuições.

Os tipos de dados usados nos operadores são :

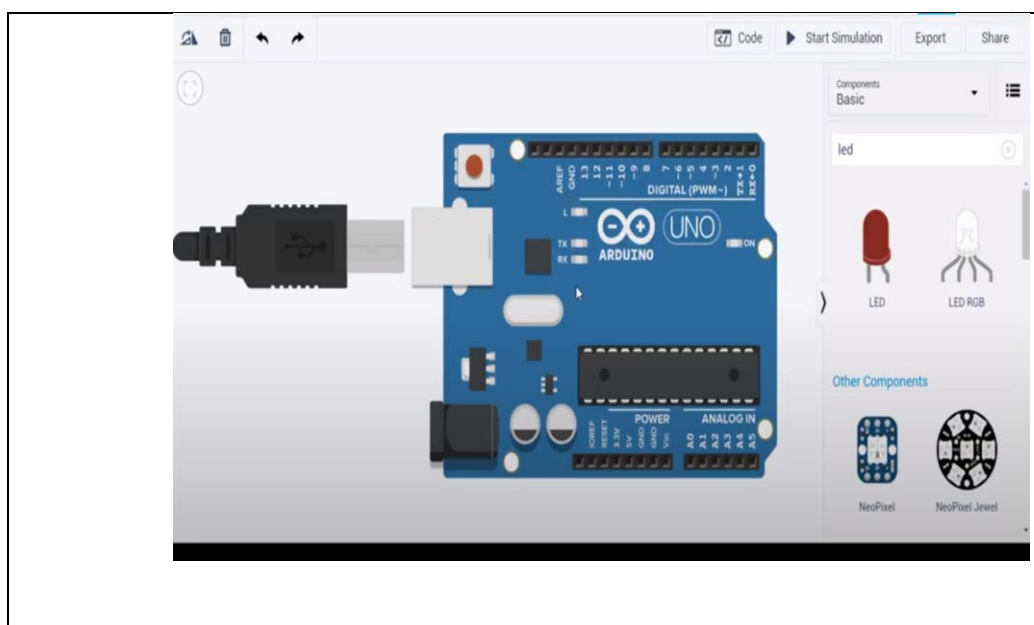
ARITMÉTICOS		ATRIBUIÇÃO	
DIVISÃO	/	SIMPLES	=
ADIÇÃO	+	COM ADIÇÃO	+=
SUBTRAÇÃO	-	COM SUBTRAÇÃO	-=
MULTIPLICAÇÃO	*	COM MULTIPLICAÇÃO	*=
RESTO	%	COM DIVISÃO	/=
INCREMENTO	++	COM MÓDULO	%=
DECREMENTO	--		

LÓGICOS		RELACIONAIS	
Conjunção	and/∧/e	IGUAL	==
DISJUNÇÃO	or/ou/∨	MAIOR	>
NEGAÇÃO	not/não	MENOR	<
		DIFERENTE	!= ou <>
		MAIOR OU IGUAL	>=
		MENOR OU IGUAL	<=

### 10.1 Operadores Aritméticos

Vamos abrir agora o nosso simulador TinkerCad e montar o circuito da Figura 39 abaixo:

Figura 39. Circuito

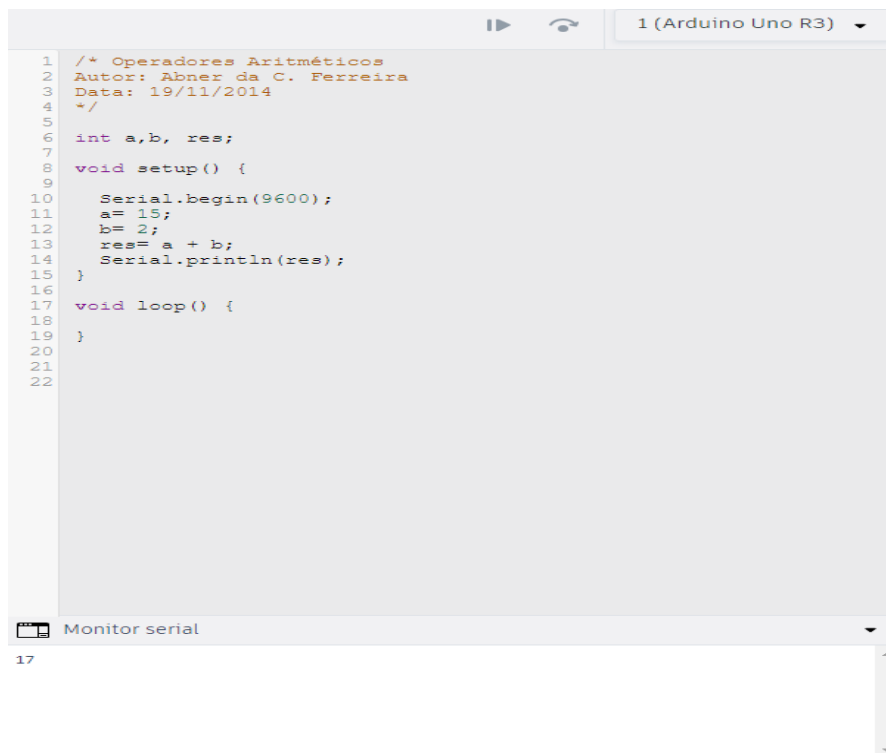


Fonte: <https://www.youtube.com/>, 2018.

Nesse caso vamos realizar as 4 operações básicas da matemática que são: adição, subtração, multiplicação e divisão.

Exemplo: Precisamos somar  $15 + 2$ , vejamos na Figura 40 como fica:

Figura 40. Soma



```

1  /* Operadores Aritméticos
2  Autor: Abner da C. Ferreira
3  Data: 19/11/2014
4  */
5
6  int a,b, res;
7
8  void setup() {
9
10     Serial.begin(9600);
11     a= 15;
12     b= 2;
13     res= a + b;
14     Serial.println(res);
15 }
16
17 void loop() {
18
19 }
20
21
22

```

Monitor serial

Fonte: Autor,2023.

Se quisermos realizar a operação  $15 - 2$ , é só alterar na variável “res”.

Exemplo:

$res = a - b;$

E terá como resultado o número 13.

Na multiplicação  $15 \times 2$ , ficará  $a * b$  que terá como resultado 30.

E por fim na divisão que seria  $15 : 2$ , ficará  $a / b$  que terá como resultado 7.

Por que será que isso aconteceu, sendo que o resultado deveria ser 7.5?

Porque as variáveis que estão sendo utilizadas no início do programa (escopo) são do tipo inteiro (int), mas na verdade deveriam ser do tipo decimal (float). Vejamos no exemplo da Figura 41 abaixo:



Figura 41.Variável Float

```

1  /* Operadores Aritméticos
2  Autor: Abner da C. Ferreira
3  Data: 19/11/2014
4  */
5
6  float a,b, res;
7
8  void setup() {
9
10     Serial.begin(9600);
11     a= 15;
12     b= 2;
13     res= a / b;
14     Serial.println(res);
15 }
16
17 void loop() {
18
19 }
20
21
22

```

Monitor serial

7.50

Fonte: Autor, 2023.

Nos operadores aritméticos se quisermos descobrir se um número é par ou ímpar, devemos dividir o primeiro número por 2. Se o resto for 0 o primeiro número é par, senão será ímpar. A sintaxe que devemos usar nesse caso é o %. Vejamos no exemplo da Figura 42 abaixo:

Figura 42.Resto da Divisão

```

1  /* Operadores Aritméticos
2  Autor: Abner da C. Ferreira
3  Data: 19/11/2014
4  */
5
6  float a,b, res;
7
8  void setup() {
9
10     Serial.begin(9600);
11     a= 15;
12     b= 2;
13     res= int (a) % int (b);
14     Serial.println(res);
15 }
16
17 void loop() {
18
19 }
20
21
22

```

Monitor serial

1.00

Fonte: Autor, 2023.

No exemplo acima o resultado deu 1, ou seja, o primeiro número (15) é ímpar.

Mas, e se quisermos que exista uma condição de que se o resto for par imprima na tela “este número é par”, senão, imprima na tela “este número é ímpar”.

Vejamos na Figura 43:

Figura 43. Condição



```

1  /* Operadores Aritméticos
2  Autor: Abner da C. Ferreira
3  Data: 19/11/2014
4  */
5
6  float a,b, res;
7
8  void setup() {
9
10     Serial.begin(9600);
11     a= 15;
12     b= 2;
13     res= int (a) % int (b);
14     if( (res)== 0){
15
16         Serial.println ("Este numero e par");
17
18     } else {
19
20         Serial.println ("Este numero e impar");
21
22     }
23
24 }
25
26
27 }
28
29 void loop() {
30
31 }
32
33
34

```

Monitor serial

Este numero e impar

Fonte: Autor, 2023.

Os comando if e else veremos mais adiante com profundidade.

Obs: incremento e decremento falaremos mais adiante quando estudarmos as Estruturas de Repetição.

## 10.2 Operadores Lógicos

### ( OR-OU-||)

Quando temos que trabalhar com operadores lógicos, devemos ter em mente que duas situações serão analisadas e será tomada uma decisão, realizada uma instrução. No

caso do operador **OU**, que em inglês é **OR**, sua sintaxe é **||**. Mas, veja que interessante, vários cursos e professores pelos quais passei, nunca haviam me ensinado onde se localizava esta barra no teclado. Vejamos na prática então, pois de nada adianta saber codificar e não se localizar no teclado. Este símbolo **|** fica do lado esquerdo da letra Z, como indicado na Figura 44 abaixo, então é só segurar o SHIFT e digitar duas vezes que está pronto.

Figura 44. Barra



Fonte: Google, 2023.

E nem pense em substituir o **||** pela letra **I** maiúscula que não dará certo. Observe o exemplo da Figura 45 abaixo:

Figura 45. Substituição da Barra por I

```

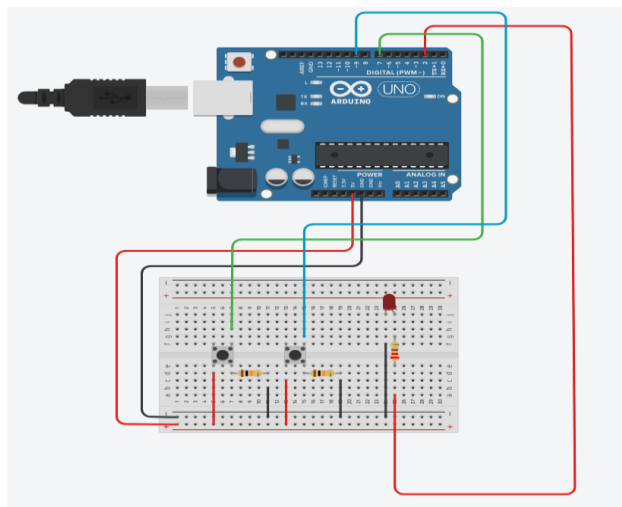
9 void loop()
0 {
1
2  if (digitalRead(botao_A) II digitalRead(botao_B))
3  {
4
5      digitalWrite(ledPin, HIGH);
6      delay(1000);
7      digitalWrite(ledPin, LOW);
8  }
9  }

```

Fonte: Autor, 2023.

Problema: Foi solicitado para que montássemos um circuito com um led e que este tenha dois botões. Quando for pressionado botão A **OU** o botão B, o led acenderá. Vamos ao código no exemplo da Figura 46 e 47 abaixo:

Figura 46. Circuito



Fonte: Autor, 2023.

Figura 47. Operador Lógico OU

```

1  /*OPERADORES LÓGICOS-OR-OU-||
2  AUTOR: ABNER DA C.FERREIRA
3  DATA: 03/03/2015
4
5  */
6
7
8  int ledPin = 2;
9  int botao_A = 7;
10 int botao_B = 9;
11
12
13 void setup() {
14   pinMode(botao_A, INPUT_PULLUP);
15   pinMode(botao_B, INPUT_PULLUP);
16   pinMode(ledPin, OUTPUT);
17 }
18
19 void loop()
20 {
21
22   if (digitalRead(botao_A) || digitalRead(botao_B))
23   {
24
25     digitalWrite(ledPin, HIGH);
26     delay(1000);
27     digitalWrite(ledPin, LOW);
28   }
29 }

```

Fonte: Autor, 2023.

### ( AND-E-&&)

Problema: Foi solicitado para que montássemos um circuito com um led (usaremos o mesmo circuito), e que este tenha dois botões. Quando for pressionado botão A E o botão B, o led acenderá. Vamos ao código da Figura 48:

Figura 48. Operador Lógico E

```
1  /*OPERADORES LÓGICOS-E-AND-&&
2  AUTOR: ABNER DA C.FERREIRA
3  DATA: 03/03/2015
4
5  */
6
7
8  int ledPin = 2;
9  int botao_A = 7;
10 int botao_B = 9;
11
12
13 void setup() {
14   pinMode(botao_A, INPUT_PULLUP);
15   pinMode(botao_B, INPUT_PULLUP);
16   pinMode(ledPin,OUTPUT);
17 }
18
19 void loop()
20 {
21
22   if (digitalRead(botao_A) && digitalRead(botao_B))
23   {
24
25     digitalWrite(ledPin, HIGH);
26     delay(1000);
27     digitalWrite(ledPin, LOW);
28   }
29 }
```

Fonte: Autor, 2023.

Veja que o circuito é o mesmo, só teremos uma dificuldade aqui no simulador, não teremos como testar apertando os dois botões ao mesmo tempo, mas, se você realizar esta montagem, dará certo.

### 10.3 Estruturas de Decisão/Seleção

As estruturas de decisão são comandos que avaliam condições lógicas, comparando e validando resultados. Temos as seguintes estruturas if-else e switch-case.

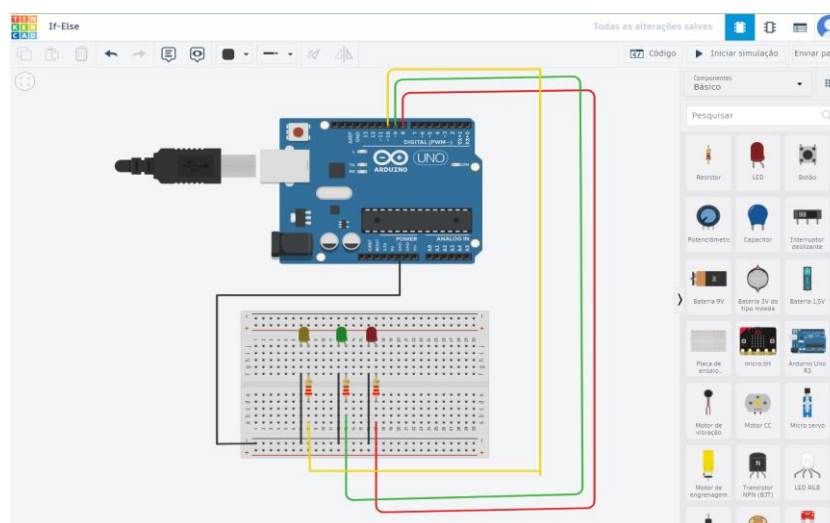
#### Se e senão (IF/ELSE)

Problema: Preciso acender três Leds; um amarelo, um verde e um vermelho, digitando no teclado 1 para acender o led amarelo, 2 para acender o led verde e 3 para acender o led vermelho.

Se, nenhum desses três números forem digitados, os leds deverão continuar apagados.

Vamos montar o circuito da Figura 49 abaixo utilizando o TinkerCad.

Figura 49. Circuito no TinkerCad



Fonte: Autor, 2023.

Vamos utilizar resistores de 220 ohms, para não queimar os leds.

Vejamos o código na Figura 50:

Figura 50. Código IF (Se)



```

1  /* Estruturas de Decisão If-Else
2  Autor: Abner da C. Ferreira
3  Data: 02/02/2015
4  */
5  String texto;
6
7
8  void setup() {
9
10     Serial.begin(9600);
11     pinMode(8,OUTPUT);
12     pinMode(9,OUTPUT);
13     pinMode(10,OUTPUT);
14 }
15
16 void loop() {
17     while (Serial.available () > 0)
18     {
19
20         texto = Serial.readString ();
21         digitalWrite(8,LOW);
22         digitalWrite(9,LOW);
23         digitalWrite(10,LOW);
24
25         if (texto == "1") {
26             digitalWrite(10,HIGH);
27         }
28
29         if (texto == "2") {
30             digitalWrite(9,HIGH);
31         }
32
33         if (texto == "3") {
34             digitalWrite(8,HIGH);
35         }
36     }
37 }
38
39
40
41
42
43
44
45
46

```

Fonte: Autor, 2023.

Já no código da Figura 51 abaixo utilizamos o comando Else, que significa Senão. Se não acontecer o que está descrito no bloco do “if”, deverá acontecer o que está dentro do “else”.

Exemplo: Se o número não for par, ele só poderá ser ímpar.

Figura 51. Código ELSE (Senão)



```

1  /* Operadores Aritméticos
2  Autor: Abner da C. Ferreira
3  Data: 19/11/2014
4  */
5
6  float a,b, res;
7
8  void setup() {
9
10     Serial.begin(9600);
11     a= 15;
12     b= 2;
13     res= int (a) % int (b);
14     if( (res)== 0){
15
16         Serial.println ("Este numero e par");
17
18     } else {
19
20         Serial.println ("Este numero e impar");
21
22     }
23
24 }
25
26
27
28
29 void loop() {
30
31 }
32
33
34

```

Monitor serial

Este numero e impar

Fonte: Autor, 2023.

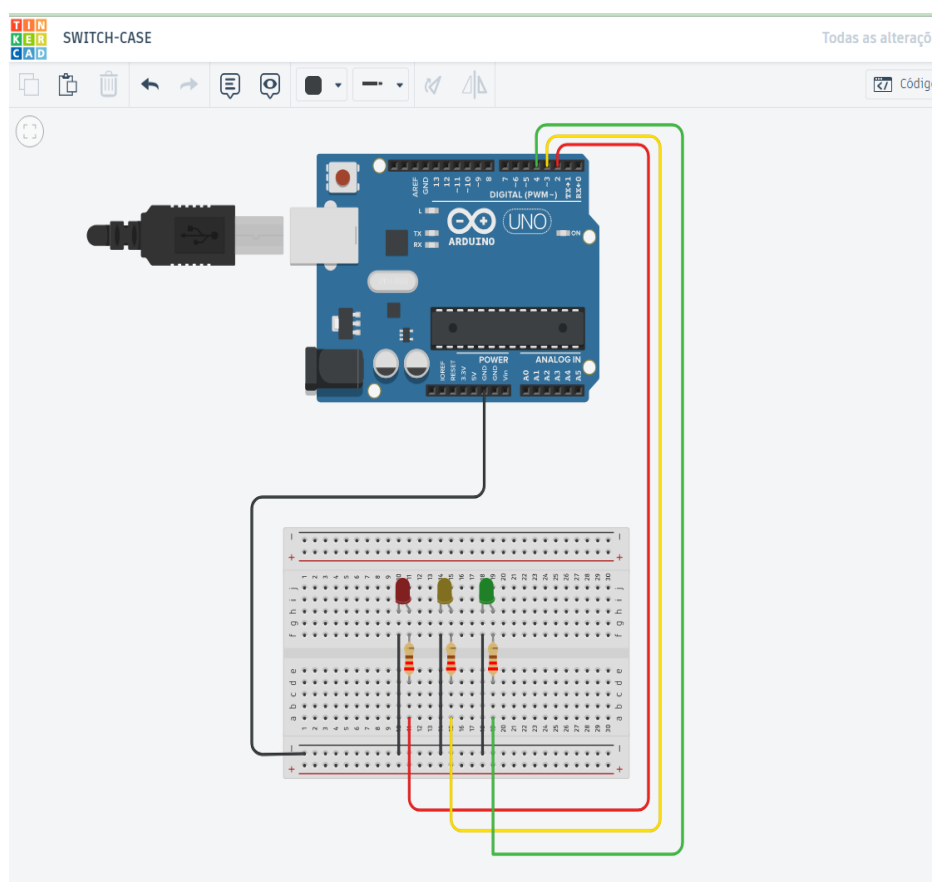
## Escolha-Caso (SWITCH/CASE)

Quando começamos a utilizar muito repetidamente a estrutura d “if”, devemos repensar e utilizar a estrutura SWITCH-CASE ,pois além de ser mais prática, torna o código mais limpo e menor.

Problema: Quero acender 3 leds, digitando no teclado do meu computador.O primeiro Led será vermelho, o segundo amarelo e o terceiro será verde.Os códigos que vou digitar são as letras “a”,”b”, e “c” respectivamente, ou seja, do tipo **char**.

Monte o circuito da Figura 52 abaixo e vamos entender a estrutura Switch-case.

Figura 52. Estrutura Switch-case



Fonte: Autor, 2023.

Abaixo na Figura 53 encontra-se o código desse circuito:



Figura 53. Código

```

1  /*Estrutura de decisão switch/case
2  autor:Abner da C.Ferreira
3  data: 04/01/2014
4  */
5
6
7  void setup()
8  {
9      Serial.begin (9600);//velocidade do monitor serial
10
11     pinMode(2, OUTPUT);//porta funcionando como saída
12     pinMode(3, OUTPUT);
13     pinMode(4, OUTPUT);
14 }
15
16 void loop(){
17
18     char valorDigitado=Serial.read();//valor digitado é lido e armazena
19     Serial.println(valorDigitado);//valor lido mostrado na tela
20     delay(1000);//tempo de 1 segundo na mensagem
21
22
23     switch (valorDigitado)//caso o valor digitado seja
24     {
25         case 'a':// a letra "a"
26             digitalWrite(2,HIGH);//ligar o led vermelho
27             digitalWrite(3,LOW);// desligar este
28             digitalWrite(4,LOW);// desligar este
29             break;
30
31         case 'b':
32             digitalWrite(2,LOW);
33             digitalWrite(3,HIGH);
34             digitalWrite(4,LOW);
35             break;

```

Fonte: Autor, 2023.

Esta estrutura é muito utilizada na programação como MENU de Opções.

Quando não é digitado nenhuma das opções, podemos se quisermos digitar **default**. No caso acima, quando nada é digitado aparece na tela a mensagem "digite o código correto para acender os leds".

## 10.4 Estruturas de Repetição

As estruturas de repetição são comandos que um bloco de instruções seja executado várias vezes até que a condição pré-estabelecida seja satisfeita. No jargão dos programadores, estas estruturas são chamadas de loops ou laços de repetição. Temos as seguintes estruturas for, while, do-while.

## For (Para/Faça)

Este comando deve ser usado quando conhecemos o número exato de repetições. A variável utilizada neste caso deve ser do tipo **inteiro** ou **literal**. A **condição** pré-estabelecida é validada no início de cada loop, enquanto a condição for verdadeira as instruções dentro do laço serão executadas, no momento em que esta condição for falsa, o laço será encerrado.

No comando **for** usaremos o incremento (++) ou o decremento (--), que é somente somar 1 ou subtrair 1 da variável.

Exemplos de:

incremento `i++` ou `i=i+1` ou `n= n+1`

decremento `i - -` ou `i= i-1` ou `n=n-1`

Antes de irmos para o exemplo, não tem como falar do comando **FOR** sem falar de **ARRAYS**.

## 11. O que são Arrays?

Arrays são vários valores dentro de uma única variável. Na verdade é uma **Matriz**, também conhecido como **Vetor**, com valores(dados) que podem ser de um único tipo ou de vários tipos string, boolean, integer, float, etc .Estes dados guardados no array podem ser acessados através de um índice.

No exemplo abaixo preciso guardar 5 valores do tipo inteiro no **array media**. No array é obrigatório o uso de colchetes [ ]. Já para declarar os valores, estes devem estar dentro de chaves { }.

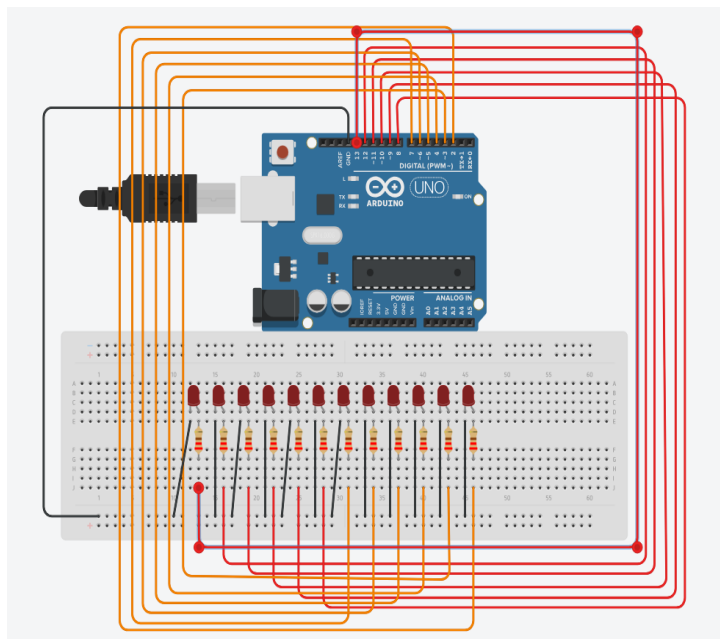
```
int media [ 5 ] = { 40,10,8,9,87 };
```

Agora, você não é obrigado a declarar a quantidade de valores dentro dos colchetes, pode deixar em branco que também funciona. Observe.

```
int media [ ] = { 40,10,8,9,87 };
```

Vejamos o exemplo das Figura 54 e 55 abaixo:

Figura 54. Array



Fonte: Autor, 2023.

Figura 55. Código

```

1  /*
2  ARRAYS-MATRIZES E LAÇOS DE REPETIÇÃO-FOR
3  AUTOR:ABNER DA C.FERREIRA
4  DATA:20/01/2015
5  */
6
7  int portas[]={13,12,11,10,9,8,7,6,5,4,3,2}; //array
8
9
10 void setup()
11 {
12
13
14     for (int n=0;n<=13;n=n+1)
15     {
16         pinMode(portas[n],OUTPUT);
17     }
18 }
19
20
21
22 void loop()
23 {
24     /*AQUI TEMOS UMA FUNÇÃO,
25     MAIS A FRENTE VEREMOS COMO FUNCIONA */
26     ligarleds();
27     // desligarleds();
28 }
29
30
31 void ligarleds()
32 {
33     digitalWrite(portas[0],HIGH);delay(100);
34     digitalWrite(portas[1],HIGH);delay(100);
35     digitalWrite(portas[2],HIGH);delay(100);
36     digitalWrite(portas[3],HIGH);delay(100);
37     digitalWrite(portas[4],HIGH);delay(100);
38     digitalWrite(portas[5],HIGH);delay(100);
39     digitalWrite(portas[6],HIGH);delay(100);
40     digitalWrite(portas[7],HIGH);delay(100);
41     digitalWrite(portas[8],HIGH);delay(100);
42     digitalWrite(portas[9],HIGH);delay(100);
43     digitalWrite(portas[10],HIGH);delay(100);
44     digitalWrite(portas[11],HIGH);delay(100);
45     digitalWrite(portas[12],HIGH);delay(100);
46     digitalWrite(portas[13],HIGH);delay(100);
47 }

```

Fonte: Autor, 2023.

Observe que na linha 14 do código na Figura 56, o `n=n+1` foi substituído por `n++` que é a mesma coisa, aliás é bem mais prático. Funciona da mesma forma.

Figura 56. Código

```

3  AUTOR:ABNER DA C.FERREIRA
4  DATA:20/01/2015
5  */
6
7  int portas[]={13,12,11,10,9,8,7,6,5,4,3,2}; //array
8
9
10 void setup()
11 {
12
13
14     for (int n=0;n<=13;n++) //comando FOR
15     {
16         pinMode(portas[n],OUTPUT); //portas funcionando como saída
17     }
18 }
19
20

```

Fonte: Autor, 2023.

Veja que se eu souber utilizar o comando **FOR**, posso reduzir e deixar meu código mais limpo. Deletei da linha 33 até a linha 43 e substitui pelo comando da Figura 57 abaixo. O resultado será o mesmo.

Figura 57. Código

```

31
32 void ligarleds() //laço de repetição
33 {
34     for (int n=0;n<=13;n++)
35     {
36         digitalWrite(portas[n],HIGH);
37         delay(100);
38     }
39 }
40

```

Fonte: Autor, 2023.

Ainda no comando **FOR**, e ainda falando de **ARRAYS**, será se conseguimos reproduzir a sinalização de uma pista de aeroporto (Figura 58) utilizando o exemplo de Montagem acima? Como ficaria o código?

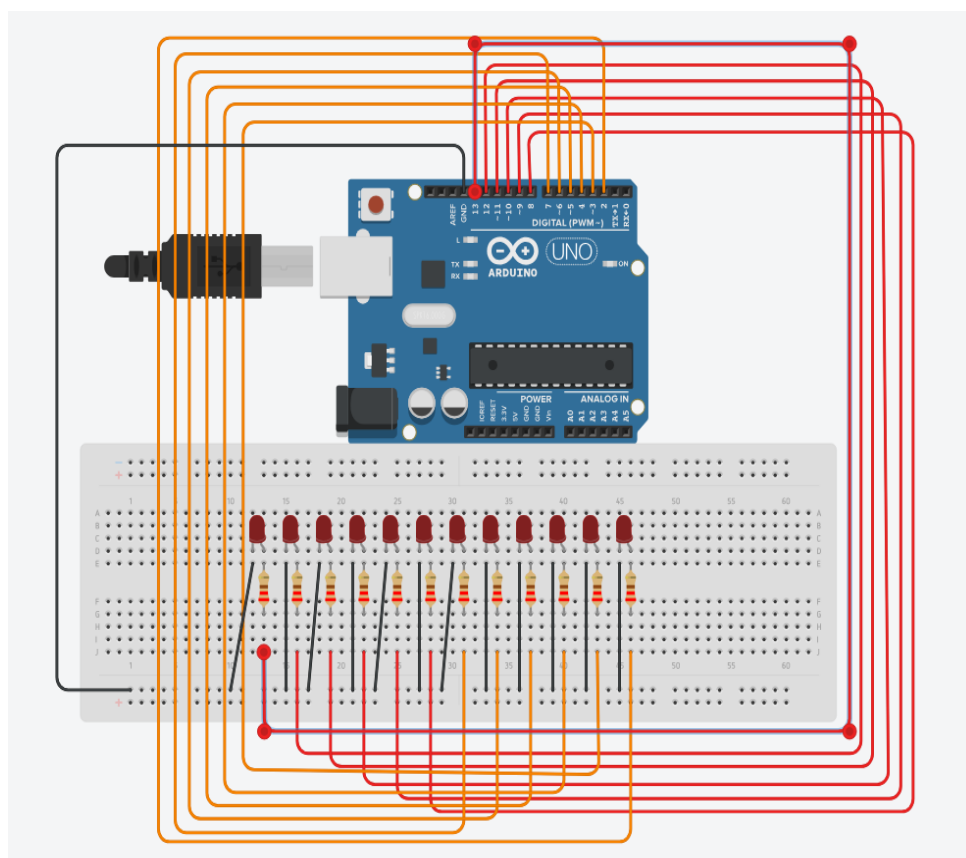
Figura 58. Pista de Aeroporto



Fonte: Autor, 2023.

Problema: Utilizando 12 leds apenas (como nas Figura 59 abaixo), simule uma pista de aterrissagem de aeroporto, que acende e apaga sem parar.

Figura 59. Circuito Pista de Aterrissagem



Fonte: Autor, 2023.

A Figura 60 representa o código dessa pista de aterrissagem:

Figura 60. Código Pista de Aterrissagem

```

3  AUTOR:ABNER DA C.FERREIRA
4  DATA:20/01/2015
5  */
6
7  int portas[]={13,12,11,10,9,8,7,6,5,4,3,2};//array
8
9
10 void setup()
11 {
12
13
14     for (int n=0;n<=13;n++)//comando FOR
15     {
16         pinMode(portas[n],OUTPUT);//portas funcionando como saída
17     }
18 }
19
20
21
22 void loop()
23 {
24     /*AQUI TEMOS UMA FUNÇÃO,
25     MAIS A FRENTE VEREMOS COMO FUNCIONA */
26     ligarleds();
27     desligarled();
28
29
30 }
31
32 void ligarleds()//laço de repetição
33 {
34     for (int n=0;n<=13;n++)
35     {
36         digitalWrite(portas[n],HIGH);
37         delay(100);
38     }
39 }
40
41 void desligarled(){
42
43     for (int n=13;n>=0;n--)
44     {
45         digitalWrite(portas[n],LOW);
46         delay(100);
47     }
48
49 }

```

## 12. O que é Identação?

A Figura 61 abaixo representa um código sobre o que é Identação:

Figura 61. Código Identação

```

3  AUTOR:ABNER DA C.FERREIRA
4  DATA:20/01/2015
5  */
6
7  int portas[]={13,12,11,10,9,8,7,6,5,4,3,2};//array
8
9
10 void setup()
11 {
12
13
14     for (int n=0;n<=13;n++)//comando FOR
15     {
16         pinMode(portas[n],OUTPUT);//portas funcionando como saída
17     }
18 }
19
20
21
22 void loop()
23 {
24     /*AQUI TEMOS UMA FUNÇÃO,
25     MAIS A FRENTE VEREMOS COMO FUNCIONA */
26     ligarleds();
27     desligarled();
28 }
29
30
31
32 void ligarleds()//laço de repetição
33 {
34     for (int n=0;n<=13;n++)
35     {
36         digitalWrite(portas[n],HIGH);
37         delay(100);
38     }
39 }
40
41 void desligarleds(){
42     for (int n=13;n>=0;n--)
43     {
44         digitalWrite(portas[n],LOW);
45         delay(100);
46     }
47 }
48
49

```

Damos o nome destes alinhamentos de **Identação**.  
Serve para que o programador consiga ler o código com  
mais clareza e identificar possíveis bugs

A **identação** é como se fossem  
os parágrafos de um texto.

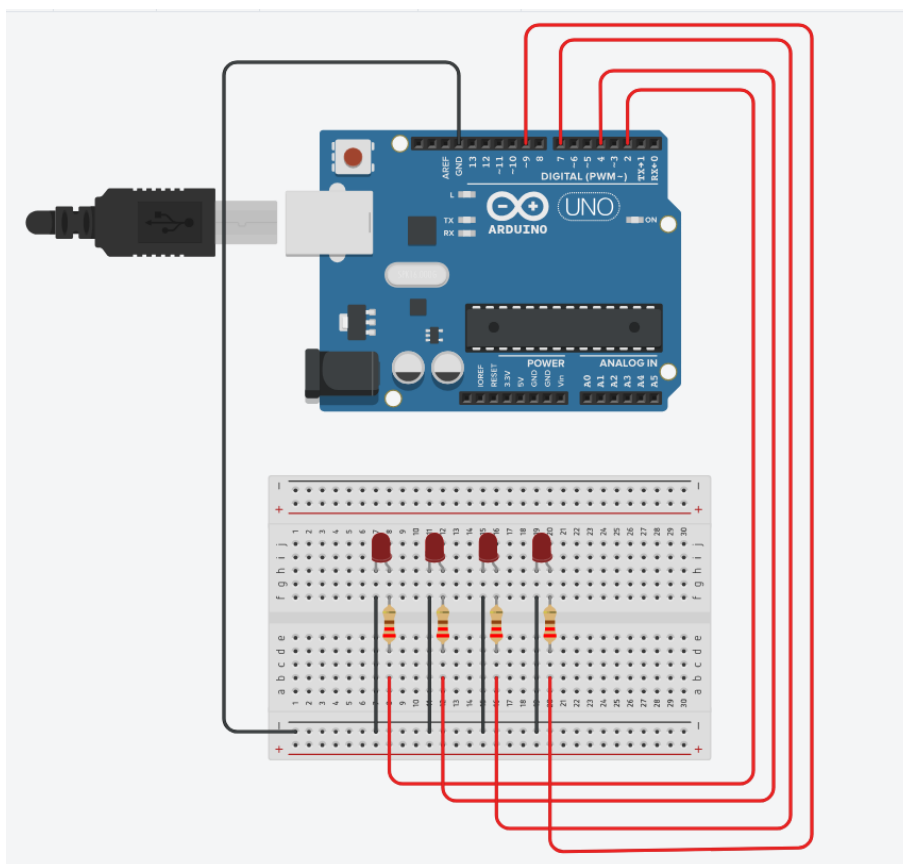
Fonte: Autor, 2023.

### While (Enquanto/Faça)

Este comando deve ser usado quando **não conhecemos** o número exato de repetições. **Primeiro ele verifica a condição, depois ele faz** o que foi programado.

Vejamos o exemplo nas Figura 62 e 63 abaixo:

Figura 62. Circuito While (Enquanto/Faça)



Fonte: Autor, 2023.

Figura 63. Código While (Enquanto/Faça)

```

1  /*
2  Estrutura de repetição- While (Enquanto-Faça)
3  Autor: Abner da C.Ferreira
4  data: 02/02/2015
5
6  */
7
8  int ledA=2;
9  int ledB=4;
10 int ledC=7;
11 int ledD=9;
12 int num_vezes=1;
13
14 void setup()
15 {
16   pinMode(ledA,OUTPUT);
17   pinMode(ledB,OUTPUT);
18   pinMode(ledC,OUTPUT);
19   pinMode(ledD,OUTPUT);
20 }
21 void loop()
22 {
23   while(num_vezes<=7)/*Enquanto não se repetir 7 vezes,
24   acenda e apague os leds */
25   {
26     {
27       digitalWrite(ledA,HIGH);
28       delay(150);
29       digitalWrite(ledA,LOW);
30       delay(150);
31       digitalWrite(ledB,HIGH);
32       delay(150);
33       digitalWrite(ledB,LOW);
34       delay(150);
35       digitalWrite(ledC,HIGH);
36       delay(150);
37       digitalWrite(ledC,LOW);
38       delay(150);
39       digitalWrite(ledD,HIGH);
40       delay(150);
41       digitalWrite(ledD,LOW);
42       delay(150);
43       num_vezes++;
44     }
45   }
46 }

```

Fonte: Autor, 2023.



No código acima, **Enquanto** a condição (`Serial.available()>0`) for maior que zero, ou seja, existir no teclado números maiores que zero, mas que sejam 1,2 ou 3 , ele executa o bloco de instruções abaixo.

While-1º avalia a condição, depois executa as instruções.

Este comando é interessante para a validação de senha num sistema, pois **enquanto** o usuário não digitar a senha correta “ 1234” vai ficar se repetindo na tela “ senha incorreta, digite novamente”.

### Do-While (Faça/Enquanto)

Este comando deve ser usado para testar a **condição de validação** do laço apenas **no final** do comando, realizando as instruções dentro do laço pelo menos uma vez. **Primeiro ele faz** o que foi programado, **depois ele verifica** a condição.

Problema: Pretende-se realizar uma contagem de 3 em 3, iniciando do 0. Se esta contagem der 24, um led deve acender. Veja as Figura 64, 65 e 66:

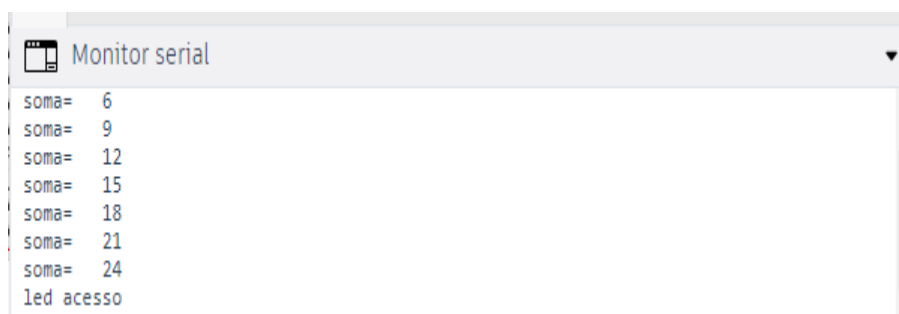
Figura 64. Código Do-While (Faça/Enquanto)

```

1  /*Estrutura de Repetição Do-While
2  Autor:Abner da Conceição Ferreira
3  Data: 22/02/2015
4  */
5  int led=2;
6  int soma=0;
7
8  void setup()
9  {
10   pinMode(led,OUTPUT);
11   Serial.begin(9600);
12   do {
13     soma=soma+3;//primeiro ele executa a instrução
14     Serial.print("soma= ");
15     Serial.println(soma);
16     delay(500);
17
18   }while (soma<24);//depois verifica a condição e acende o led
19   Serial.println("led acesso");
20
21
22
23   }
24
25  void loop()
26  {
27     if (soma ==24)
28     {
29       {
30         digitalWrite(led,HIGH);
31       }
32     }
33   }
34 }
35
36

```

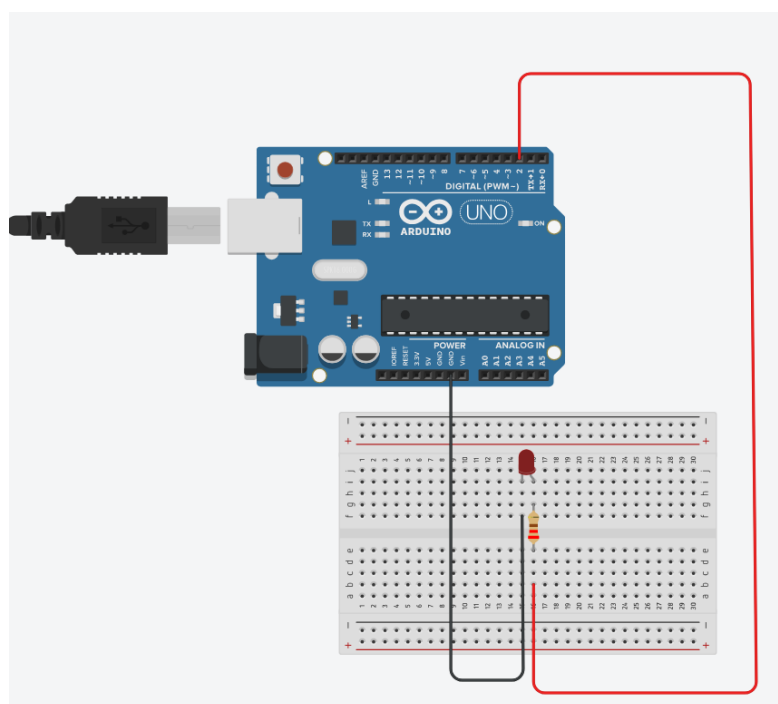
Figura 65. Monitor Serial



```
Monitor serial
soma= 6
soma= 9
soma= 12
soma= 15
soma= 18
soma= 21
soma= 24
led acesso
```

Fonte: Autor, 2023.

Figura 66. Circuito



Fonte: Autor, 2023.

### 13. Funções na Programação

Quando um cliente solicita uma solução para um determinado problema, e iniciamos nosso código, percebemos que se este só tem aproximadamente de 20 a 30 linhas, dá para enterdermos tranquilamente, é de fácil compreensão e quem entende de código também vai entender do que se trata. Entretanto, quando este código vai crescendo, ficando mais complexo, 100, 200, 400 linhas, o negócio fica mais complicado, pois você percebe que existem certos trechos que você repetiu muitas vezes a mesma coisa, a mesma instrução,

ou o conjunto de instruções. O código ficou “monstro”. Mas, e se, ao invés de ficarmos repetindo este conjunto de instruções, criarmos isto uma única vez e todas as vezes que precisarmos usar aquele conjunto de instruções, utilizarmos apenas uma palavrinha “mágica” chamando (invocando) tudo o que preciso, não é mais inteligente? Mais econômico? Mais legal?

Pois bem, o nome desta palavrinha “mágica” é a tal da função ( function ). A função nada mais é do que um mini-programa, mini-código dentro do código, que pode ou não retornar alguma coisa.

Uma função precisa ter um **identificador**, um **tipo de retorno** e os **parâmetros de entrada**. Na entrada, podemos ter apenas uma entrada, várias entradas ou nenhuma entrada.

No exemplo da Figura 67 abaixo foram criadas duas funções: `ligarled()` e `desligarled()`, que foram invocadas mais abaixo.

Figura 67. Função Ligar e Desligar

```

3  AUTOR:ABNER DA C.FERREIRA
4  DATA:20/01/2015
5  */
6
7  int portas[]={13,12,11,10,9,8,7,6,5,4,3,2}; //array
8
9
10 void setup()
11 {
12
13
14     for (int n=0;n<=13;n++) //comando FOR
15     {
16         pinMode(portas[n],OUTPUT); //portas funcionando como saída
17     }
18 }
19
20
21 void loop()
22 {
23     /*AQUI TEMOS UMA FUNÇÃO,
24     */
25     //função 1 e função 2
26     ligarleds();
27     desligarled();
28 }
29
30
31 void ligarleds() //laço de repetição função 1
32 {
33     for (int n=0;n<=13;n++)
34     {
35         digitalWrite(portas[n],HIGH); //retorno é ligar os leds
36         delay(100);
37     }
38 }
39
40 void desligarleds() //função 2
41 {
42     for (int n=13;n>=0;n--)
43     {
44         digitalWrite(portas[n],LOW); //retorno é desligar os leds
45         delay(100);
46     }
47 }
48
49

```

O que pode ser feito aqui é infinito, você pode criar diversas funções; `ligarMotores1_3()`, `ligarMotores2_4`, `ligarSirene()`, `desligar_Sirene`, etc... Ou seja, simplificar seu código reduzindo muitas linhas.

No exemplo abaixo, um led acenderá gradativamente e em seguida apagará lentamente. Para isto usaremos uma função que daremos o nome de **brilho()**. Perceba que uma função leva seu nome e logo a frente um conjunto de parênteses, mas, cuidado para não se confundir quando avançar nos seus estudos.

() utilizados para funções (Fig 68.):

Figura 68. Parênteses

```

32 void ligarleds() //laço de repetição
33 {
34     for (int n=0;n<=13;n++)
35     {
36         digitalWrite(portas[n],HIGH);
37         delay(100);
38     }
39 }
40

```

Fonte: Autor, 2023.

{ } utilizados para conjunto de instruções a serem realizadas, logo após a função (Fig 69.)

Figura 69. Chaves

```

1 void brilho(int intensidade)
2 {
3     analogWrite(led, intensidade);
4     delay(500);
5 }

```

Fonte: Autor, 2023.

Entretanto, { } em Javascript significa Objeto (Fig 70.), que você estudará mais adiante em POO, Programação Orientada a Objetos.

Figura 70. Objeto

```

var carro = {
  marca: "Ford",
  modelo: "Ka",
  getDetalhes: function () {
    return this.marca + ' - ' + this.modelo;
  }
}

```

Fonte: Autor, 2023.

[ ] utilizados para arrays (Fig 71.):

Figura 71. Colchetes

```

3  AUTOR:ABNER DA C.FERREIRA
4  DATA:20/01/2015
5  */
6
7  int portas[]={13,12,11,10,9,8,7,6,5,4,3,2}; //array
8
9
1
2  /* Funções
3  Autor:ABNER DA CONCEIÇÃO FERREIRA
4  data:16/06/2015
5  */
6
7  int led = 6;
8
9  // Declaração da Função
10
11 void brilho(int intensidade)
12 {
13     analogWrite(led, intensidade);
14     delay(500);
15 }
16
17 void setup()
18 {
19     pinMode(led, OUTPUT);
20 }
21
22 void loop()
23 {
24     // funções sendo invocadas com cada valor de seu brilho
25     brilho(5);
26     brilho(55);
27     brilho(105);
28     brilho(155);
29     brilho(205);
30     brilho(255);
31     brilho(205);
32     brilho(155);
33     brilho(105);
34     brilho(55);
35     brilho(5);
36 }

```

Fonte: Autor, 2023.

Array [ ] em Javascript (Figuras 72 e 73):

Figura 72. Array em JavaScript

```

var array = [{nome:'joao', cidade: 'RJ'}, {nome:'maria', cidade: 'SP'}];
console.log(array[1].cidade); // retorna "SP"

```

Fonte: Autor, 2023.



FrontEnd=Html5 +CSS3

Perceba que reduziu o estudo de uma Linguagem de Programação e isto é importantíssimo em tecnologia, tempo. E o mais interessante, é que se chegamos até este ponto de estudos, o nome desta nova carreira é FullStack.

FullStack = Javascript + Node + Mongo DB + Html5 +CSS3

Mas, deixemos este assunto para o curso de Node JS.

## CONCLUSÃO

Chegamos ao final deste curso de Lógica de Programação, espero que tenhamos atendido suas expectativas e que este seja o início de uma brilhante carreira, pois na Tecnologia, quando se é bom no que faz, você não procura emprego, mas sim, ele é que o procura, portanto, esteja disposto a estudar, siga as dicas de seus professores, não se contente somente com este curso, avance em Node Js, Javascript, HTML5, CSS3 e colherá excelentes resultados, e nunca se esqueça de método Estudar,Praticar,Praticar, pois, existem duas coisas na vida que se você investir o retorno será certo e real; imóveis e nos seus estudos.Seja feliz em sua transição de carreira ou início de nova carreira. Um abraço a todos!

Os autores



## REFERÊNCIAS

BACELAR, Guilherme Espírito Santo . **Apostila de Eletrônica Digital**. Ano 2016.

**Disponível em:** <https://wiki.sj.ifsc.edu.br/>

**Acesso em:** 08/06/2023.

CARDOSO, Guilherme. **Ada Lovelace: A Primeira Programadora da História**. Ano 2020.

**Disponível em:** <http://www.ime.unicamp.br/~apmat/ada-lovelace/>

**Acesso em:** 07/07/2023.

**Char**. Ano 2016.

**Disponível em:** <https://cdn.arduino.cc/reference/pt/language/variables/data-types/>

**Acesso em:** 10/06/2023.

CORRÊA, Iran Carlos Stalliviere. **Historia do Ábaco**. Ano 2016.

**Disponível em:** <https://igeo.ufrgs.br/museudetopografia/images/acervo/artigos>

**Acesso em:** 17/06/2023.

**How to Wire and Program a Button**. Ano 2015.

**Disponível em:** <https://docs.arduino.cc/built-in-examples/digital/Button#code>

**Acesso em:** 04/07/2023.

IMPACTA, Redação. **Você sabe o que é Visual Studio?**. Ano 2017.

**Disponível em:** <https://www.impacta.com.br/blog/voce-sabe-o-que-e-visual-studio/>

**Acesso em:** 03/05/2023.

MARQUES, Rafael. **O que é HTML? Entenda de forma descomplicada**. Ano 2022.

**Disponível em:** <https://www.homehost.com.br/blog/tutoriais/o-que-e-html/>

**Acesso em:** 08/05/2023.

MARTINI, Sidnei. **Historia da Computação**. Ano 2020.

**Disponível em:** <https://edisciplinas.usp.br/pluginfile.php/>

**Acesso em:** 19/05/2023.

NOLETO, Cairo. **CRUD: as 4 operações básicas do banco de dados!**. Ano 2021.

**Disponível em:** <https://blog.betrybe.com/tecnologia/crud-operacoes-basicas/> **Acesso em:** 19/11/2022

Oracle. **O que é um Banco de Dados?**. Ano 2022.

**Disponível em:** <https://www.oracle.com/br/database/what-is-database/>

**Acesso em:** 18/05/2023.

NOLETO, Cairo. **Framework: o que é, como ele funciona e para que serve?**. Ano 2020.

**Disponível em:** <https://blog.betrybe.com/>

**Acesso em:** 03/05/2023.

PEREIRA, Fabio. **Microcontroladores PIC: Programação em C**. 2. ed. São Paulo:

Editora Erica, 2003.

**Disponível em:** <https://materialpublic.imd.ufrn.br/curso/disciplina/2/61/1/4>

**Acesso em:** 20/05/2023.

PROGRAMMING, HM. **{Portugol Studio} #2 - Entrada e Saída de Dados.** Youtube. Ano 2016.

**Disponível em:** <https://www.youtube.com/watch?v=qTta8m3CTEo>

**Acesso em:** 13/06/2023.

QUANTICOS, Universos. **Lógica Paraconsistente: Máquina de Turing.** Ano 2017.

**Disponível em:** <https://universosquanticos.wordpress.com/>

**Acesso em:** 19/07/2023.

ROSA, Daniel Lemos da. **O Que é Arduino?.** Ano 2017.

**Disponível em:** <https://www.usinainfo.com.br/blog/o-que-e-arduino/>

**Acesso em:** 14/05/2023.

ROVEDA, Ugo. **JAVASCRIPT: O QUE É, PARA QUE SERVE E COMO FUNCIONA O JS?.** Ano 2021.

**Disponível em:** <https://kenzie.com.br/blog/javascript/>

**Acesso em:** 19/05/2023.

SÖDERBY, Karl. **Getting Started with Arduino.** Ano 2023.

**Disponível em:** <https://docs.arduino.cc/learn/starting-guide/getting-started-arduino>

**Acesso em:** 09/06/2023.

THOMSEN, Adilson. **Monitore sua planta usando Arduino.** Ano 2016.

**Disponível em:** <https://www.makerhero.com/blog/>

**Acesso em:** 14/06/2023.