



FIAP

Engenharia de Software

EDGE COMPUTING & COMPUTER SYSTEMS

04 – Tipos de Variáveis



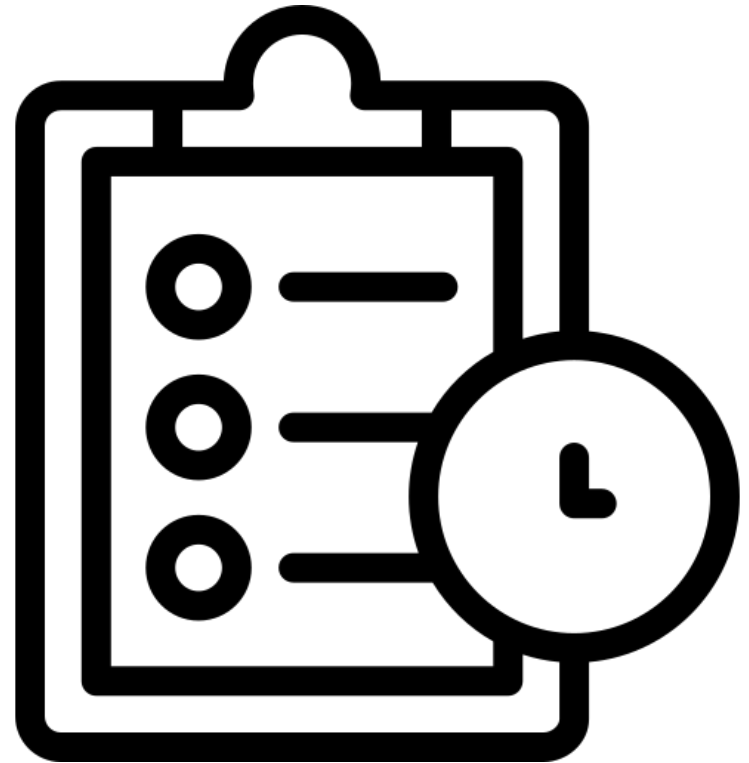
Prof. Airton Y. C. Toyofuku



profairton.toyofuku@fiap.com.br

Agenda

- Tipos de Variáveis;
- Variáveis Numéricas;
- Variáveis Booleanas;
- Variáveis do Tipo Caractere;
- Tipos no **printf**;
- Exercícios;
- Arrays;
- Matrizes;
- Exercícios;
- Strings;
- Exercícios;
- Laboratório;



Tipos de Variáveis

Numérico: Inteiros (int) ou reais (float), podendo ser usado para cálculos matemáticos;
Exemplo: 1, 0, 3.14

Caracteres (char): Símbolos que não contém números, como nomes;
Exemplo 'a', 'b', '\$'

Alfanumérico: combinação de números e letras, podendo conter só letras ou só números, mas não pode executar operações matemáticas;

Lógica / booleana: Verdadeiro ou falso;
Exemplo: "true", "false," "TRUE", "FALSE".
Obs: Algumas linguagens atribuem falso ao valor 0 e verdadeiro a qualquer outro valor diferente de 0

Variáveis Numéricas

Numérico: Inteiros (int) ou reais (float), podendo ser usado para cálculos matemáticos;

Exemplo: 1, 0, 3.14

Tipo	Tamanho	Intervalo
Void	0 Bytes	Nulo
Char	1 byte	-128 a 127
Unsigned Char	1 byte	0 a 255
Int	2 bytes	-32.768 a 32.767
Unsigned Int	2 bytes	0 a 65.535
Long	4 bytes	-2.147.483.648 a 2.147.483.647
Unsigned Long	4 bytes	0 a 4.294.967.295
Float	4 bytes	$3,4 \times 10^{-38}$ a $3,4 \times 10^{38}$
Double	8 bytes	$1,7 \times 10^{-308}$ a $1,7 \times 10^{308}$

Variáveis Numéricas

```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3
4 int main() {
5     printf("tamanho do char: · %ld bytes \n", sizeof(char));
6     printf("tamanho do int: · %ld bytes \n", sizeof(int));
7     printf("tamanho do long: · %ld bytes \n", sizeof(long));
8     printf("tamanho do float: · %ld bytes \n", sizeof(float));
9     printf("tamanho do double: %ld bytes \n", sizeof(double));
10    return 0;
11 }
```

/tmp/HQnFb1614X.o


tamanho do char: · 1 bytes
tamanho do int: · 4 bytes
tamanho do long: · 8 bytes
tamanho do float: · 4 bytes
tamanho do double: 8 bytes



Mas os tipos **Int**
e **Long** não
tinham tamanho
de 2 e 4 bytes,
respectivamente
?

Então por que
quando
rodamos o
código, eles
mostram **4 e 8**
bytes?

Por que tudo
depende da
arquitetura do
computador, do
tipo de
linguagem e do
tipo de
compilador!



Dependendo do compilador, também podemos definir variáveis **inteiras de tamanhos diferentes**:

Tipo	Descrição
int8	Inteiro com sinal de 1 byte
int16	Inteiro com sinal de 2 bytes
int32	Inteiro com sinal de 4 bytes
int64	Inteiro com sinal de 8 bytes
uint8	Inteiro sem sinal de 1 byte
uint16	Inteiro sem sinal de 2 bytes
uint32	Inteiro sem sinal de 4 bytes
uint64	Inteiro sem sinal de 8 bytes

Variáveis Booleanas

Lógica / booleana: Verdadeiro ou falso;

Exemplo: “true”, “false,” “TRUE”, “FALSE”.

Obs: Algumas linguagens atribuem falso ao valor 0 e verdadeiro a qualquer outro valor diferente de 0

Tipo	Tamanho na RAM	Intervalo
Bool	1 byte	0 ou 1 (False ou True)

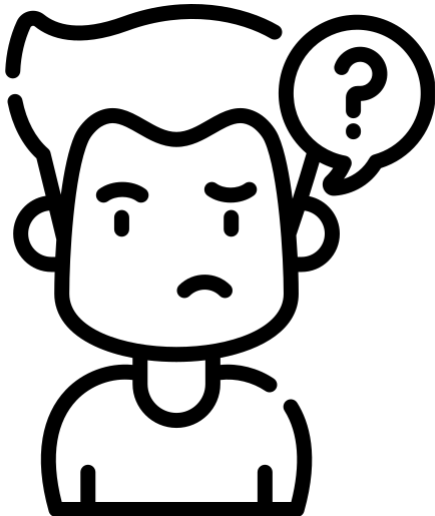
```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3 #include <stdbool.h>
4
5 int main() {
6     bool Verdade = true;
7     bool Falso = false;
8     printf("tamanho do bool: · %ld bytes \n", sizeof(bool));
9     printf("tamanho do true: · %ld \n", Verdade);
10    printf("tamanho do false: · %ld \n", Falso);
11    return 0;
12 }
```

```
/tmp/HQnFbl6l4X.o
tamanho do bool: · 1 bytes
tamanho do true: · 1
tamanho do false: · 0
```


Variáveis do tipo Caractere

Caracteres (char): Símbolos que não contém números, como nomes;
Exemplo 'a', 'b', '\$'

Tipo	Tamanho	Intervalo
Char	1 byte	-128 a 127
Unsigned Char	1 byte	0 a 255



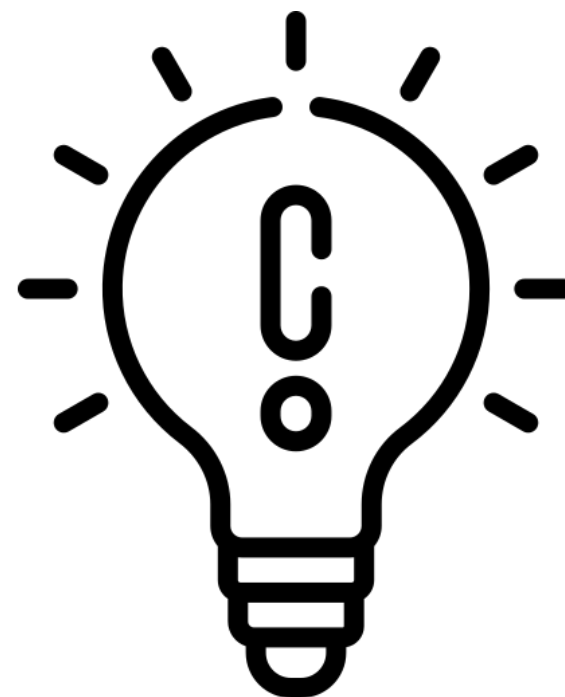
MAS ESPERA AI
DE NOVO!

Char e Unsigned
Char não eram
VARIÁVEIS
NUMÉRICAS?

Por definição,
SIM! Mas
lembrem da
Tabela ASCII?

Variáveis do tipo Caractere

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]



Cada símbolo (Caractere) está mapeado em 8 bits, ou 1 byte.

Por isso, podemos usar o tipo **CHAR** ou **UNSIGNED CHAR** como um **VALOR NUMÉRICO**, ou como um **SÍMBOLO** da tabela ASCII.

Variáveis do tipo Caractere

```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3 #include <stdbool.h>
4
5 int main() {
6     char variavel = 'a';
7     printf("tamanho do char: %ld bytes \n", sizeof(char));
8     printf("caractere em variavel: %c \n", variavel);
9     printf("valor numerico da variavel: %ld \n", variavel);
10    return 0;
11 }
```

```
/tmp/HQnFb1614X.o
tamanho do char: 1 bytes
caractere em variavel: a
valor numerico da variavel: 97
```

Decimal	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d

Perceba que
agora usamos o
%c para mostra
a letra 'a'

Isso porque o
printf precisa
saber que tipo
de variável ele
vai mostrar

Logo:
%ld para
números e %c
para caracteres

Tipos de printf

Como vimos, precisamos dizer pro **printf** qual o tipo de variável ele vai mostrar. A tabela a seguir serve como uma **cola** de **alguns** dos principais tipos aceitos pelo printf.

Tipo	Símbolo	Descrição
Char	%c	Mostra o caractere
Char	%d	Mostra o valor numérico do caractere
Int	%d	Mostra o valor numérico da variável
Long	%ld	Mostra o valor numérico da variável
Int / Long / Char	%x	Mostra o valor numérico em Hexadecimal
Float / Double	%f	Mostra o valor numérico da variável
Float / Double	%.nf	Mostra o valor numérico da variável com n casas decimais.
String	%s	Mostra a frase ou palavra contido na variável

Exercícios

1. Crie uma variável booleana chamada "isTrue" e atribua o valor true.
2. Crie uma variável inteira chamada "idade" e atribua o valor 25.
3. Crie uma variável do tipo caractere chamada "letra" e atribua o valor 'A'.
4. Declare uma variável float chamada "altura" e atribua o valor 1.75.
5. Imprima o valor da variável "isTrue" usando printf.
6. Imprima o valor da variável "idade" usando printf.
7. Imprima o valor da variável "letra" usando printf.
8. Imprima o valor da variável "altura" usando printf.
9. Declare uma variável inteira chamada "numero1" e atribua o valor 10.
10. Declare uma variável inteira chamada "numero2" e atribua o valor 5.
11. Imprima a soma dos valores das variáveis "numero1" e "numero2".
12. Imprima a diferença entre os valores das variáveis "numero1" e "numero2".
13. Imprima o produto dos valores das variáveis "numero1" e "numero2".
14. Imprima o resultado da divisão inteira entre os valores das variáveis "numero1" e "numero2".
15. Imprima o resultado da divisão real entre os valores das variáveis "numero1" e "numero2".
16. Declare uma variável do tipo caractere chamada "caractere1" e atribua o valor 'X'.
17. Declare uma variável do tipo caractere chamada "caractere2" e atribua o valor 'Y'.
18. Imprima a concatenação dos valores das variáveis "caractere1" e "caractere2".
19. Imprima a formatação de uma frase utilizando printf, por exemplo: "O valor da variável idade é: 25".
20. Utilize printf para exibir uma mensagem de boas-vindas personalizada, incluindo o nome do usuário.



Arrays

São Estruturas de **DADOS**;

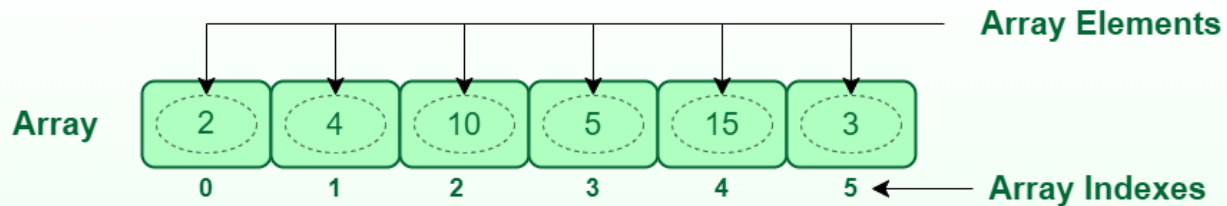
O acesso aos elementos da coleção é feito **através de seu índice**;

Permitem armazenar uma coleção de valores, mas **sempre do mesmo tipo**;

O primeiro elemento tem **índice 0**, o segundo tem índice 1 e assim por diante **até n-1**, onde n é o tamanho do array;

A declaração de um array é feita indicando **seu Tipo, seu Nome e seu Tamanho entre []**:
`Int vetor[10];`

Pode se alterar o valor de um elemento acessando seu índice e atribuindo um novo valor:
`Vetor[5] = 32;`



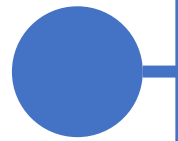
Arrays

```
1 #include <stdio.h>
2
3 int main() {
4     int a [10];
5     char b [] = {'a','b','c','d','e'};
6     int c [] = {12, 13, 14, 15, 16};
7     printf("b0: %c \n", b[0]);
8     printf("b3: %c \n", b[3]);
9
10    printf("a0: %d \n", a[0]);
11    printf("a3: %d \n", a[3]);
12
13    a[0] = c[0] + c[1];
14    a[3] = 2 * a[0];
15
16    printf("a0: %d \n", a[0]);
17    printf("a3: %d \n", a[3]);
18    return 0;
19 }
```

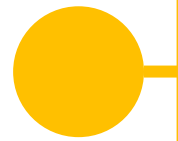
/tmp/HQnFb1614X.o

b0: a
b3: d
a0: 0
a3: 0
a0: 25
a3: 50

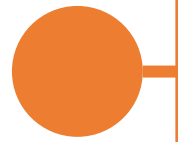
Matrizes



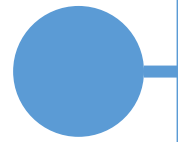
Estruturas de dados que permitem armazenar uma coleção de elementos de mesmo tipo, organizados em linhas e colunas.



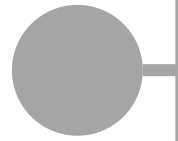
O acesso aos elementos da coleção é feito **através de seu índice da linha e da coluna**;



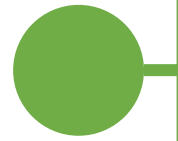
É uma tabela retangular, onde cada elemento é acessado por um índice que representa sua posição na matriz.



Pode se alterar o valor de um elemento acessando seu índice e atribuindo um novo valor:
`Matriz[1][5] = 32;`



A declaração é feita indicando **seu Tipo, seu Nome e seu Tamanho da Linha e Tamanho da Coluna**:
`Int matriz[6][6];`



São úteis para armazenar dados tabulares, como matrizes numéricas, imagens, mapas e muito mais.

	Column 0	Column 1	Column 2
Row 0	<code>x[0][0]</code>	<code>x[0][1]</code>	<code>x[0][2]</code>
Row 1	<code>x[1][0]</code>	<code>x[1][1]</code>	<code>x[1][2]</code>
Row 2	<code>x[2][0]</code>	<code>x[2][1]</code>	<code>x[2][2]</code>

Matrizes

```
1  #include <stdio.h>
2
3  int main() {
4      int matriz[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
5
6      // Imprimir a matriz
7      printf("Matriz:\n");
8      for (int i = 0; i < 3; i++) {
9          for (int j = 0; j < 3; j++) {
10             printf("%d ", matriz[i][j]);
11         }
12         printf("\n");
13     }
14
15     return 0;
16 }
```

/tmp/HQnFb1614X.

Matriz:

1 2 3

4 5 6

7 8 9

Exercícios

1. Crie um programa em C que declare um array de inteiros com 5 elementos e preencha-o com valores digitados pelo usuário.
2. Escreva um programa em C que declare um array de caracteres e inicialize-o com uma palavra de sua escolha. Em seguida, imprima o conteúdo do array.
3. Crie um programa em C que declare uma matriz de inteiros 3x3 e preencha-a com valores aleatórios. Em seguida, imprima a matriz.
4. Escreva um programa em C que leia uma matriz de inteiros 3x3 digitada pelo usuário e calcule a soma de todos os elementos.
5. Crie um programa em C que declare uma matriz de floats 2x2 e preencha-a com valores fornecidos pelo usuário. Em seguida, imprima a matriz.
6. Escreva um programa em C que leia uma matriz de floats 3x3 digitada pelo usuário e calcule a média dos elementos.
7. Crie um programa em C que declare um array de inteiros com 10 elementos e encontre o valor mínimo e máximo no array. Imprima esses valores.
8. Escreva um programa em C que declare um array de inteiros com 5 elementos e calcule a soma de todos os elementos pares.
9. Crie um programa em C que leia uma matriz de inteiros 3x3 digitada pelo usuário e verifique se a matriz é simétrica.
10. Escreva um programa em C que declare uma matriz de caracteres 2x2 e preencha-a com letras do alfabeto. Imprima a matriz.
11. Crie um programa em C que declare um array de inteiros com 6 elementos e ordene o array em ordem crescente.
12. Escreva um programa em C que leia duas matrizes de inteiros 2x2 digitadas pelo usuário e calcule a soma das matrizes.
13. Crie um programa em C que declare um array de floats com 5 elementos e calcule a média dos elementos.
14. Escreva um programa em C que declare uma matriz de inteiros 3x3 e preencha-a com valores fornecidos pelo usuário. Em seguida, imprima a matriz transposta.
15. Crie um programa em C que leia uma matriz de inteiros 3x3 digitada pelo usuário e verifique se todos os elementos são positivos.
16. Escreva um programa em C que declare uma matriz de caracteres 3x3 e preencha-a com valores aleatórios. Em seguida, imprima a matriz.
17. Crie um programa em C que declare um array de inteiros com 7 elementos e inverta a ordem dos elementos.
18. Escreva um programa em C que leia duas matrizes de inteiros 2x2 digitadas pelo usuário e calcule o produto matricial.
19. Crie um programa em C que declare uma matriz de floats 4x4 e preencha-a com valores aleatórios. Em seguida, imprima a soma de cada linha.
20. Escreva um programa em C que leia uma matriz de inteiros 3x3 digitada pelo usuário e verifique se existe algum elemento repetido.



Strings

Alfanumérico: combinação de números e letras, podendo conter só letras ou só números, mas não pode executar operações matemáticas;

Estruturas de dados que permitem **armazenar um texto;**

A declaração pode ser feita de duas formas, dependendo do compilador:
Char nomeDaString [tamanho]
String nomeDaString [tamanho]

A atribuição de uma string pode ser feita de uma só vez:
nomeDaString = "Isso é uma String!"

O acesso aos elementos individuais da coleção é feito **através de seu índice no array;**

O ultimo elemento deve ser **"\0"** ou **NULL**, para indicar o fim da string

Na prática, é um **array alfanumérico.**

Strings

```
1 #include <stdio.h>
2 #include <string.h>
3 int main() {
4     char frase[18] = "Isso eh uma frase";
5
6     printf("A frase é: %s \n", frase);
7     printf("O tamanho da frase é de %ld caracteres \n", strlen(frase));
8     printf("O tamanho da estrutura de dados é de %ld bytes \n", sizeof
        (frase));
9     printf("A letra no indice 5 da frase é '%c' \n", frase[5]);
10    return 0;
11 }
```

/tmp/HQnFb1614X.o
A frase é: Isso eh uma frase
O tamanho da frase é de 17 caracteres
O tamanho da estrutura de dados é de 18 bytes
A letra no indice 5 da frase é 'e'

▶ Não existe uma documentação oficial sobre as bibliotecas disponíveis em C;

▶ Mas podemos encontrar manuais e documentos de forma muito fácil na internet;
Um exemplo: https://www.tutorialspoint.com/c_standard_library/string_h.htm

Exercícios

1. Crie um programa em C que leia uma string digitada pelo usuário e imprima o seu comprimento.
2. Escreva um programa em C que leia duas strings digitadas pelo usuário e concatene-as em uma terceira string.
3. Crie um programa em C que leia uma string digitada pelo usuário e verifique se é um palíndromo (ou seja, se pode ser lida da mesma forma da esquerda para a direita e vice-versa).
4. Escreva um programa em C que leia uma string digitada pelo usuário e conte quantas vogais ela possui.
5. Crie um programa em C que leia uma string digitada pelo usuário e verifique se é um número válido (contendo apenas dígitos numéricos).
6. Escreva um programa em C que leia uma string digitada pelo usuário e converta todas as letras minúsculas para maiúsculas.
7. Crie um programa em C que leia uma string digitada pelo usuário e remova os espaços em branco.
8. Escreva um programa em C que leia uma string digitada pelo usuário e substitua todas as ocorrências de uma determinada letra por outra letra fornecida.
9. Crie um programa em C que leia uma string digitada pelo usuário e verifique se é um palíndromo ignorando espaços em branco e diferenças de maiúsculas/minúsculas.
10. Escreva um programa em C que leia uma string digitada pelo usuário e inverta a ordem dos caracteres.



Laboratório – LED Chase Effect

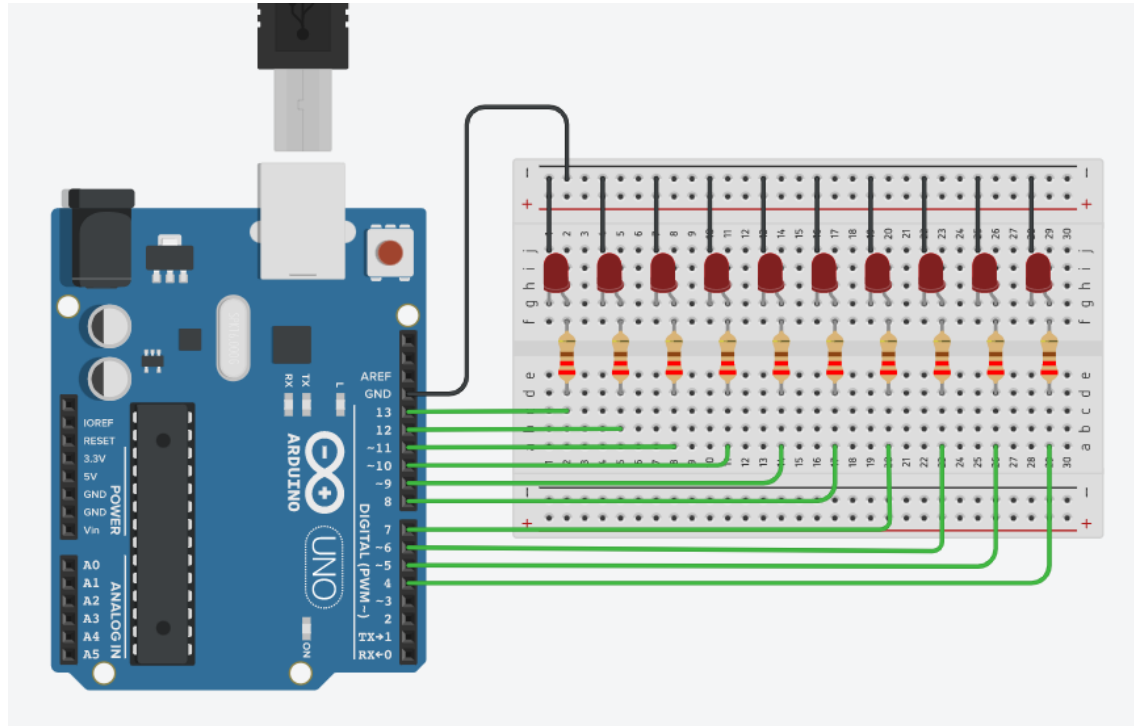
Neste projeto vamos fazer com que cada Led acenda na sequência, apagando o anterior, causando o efeito de perseguição.



Para isso, vamos implementar os conceitos de array vistos na aula.

Material necessário:

- 1 Arduino;
- 10 Resistores de 220 ohms
- 10 Leds Vermelho;
- 1 Protoboard;
- Jumpers cables.



Link: [Projeto 04 – LED Chase Effect](#)

Laboratório Desafio - Interactive Traffic Lights

Vamos melhorar o **Semáforo** desenvolvido na aula anterior, adicionado uma nova feature para pedestres.

1 – Desenhe um fluxograma de como o semáforo deve funcionar, seguindo o seguinte racional:

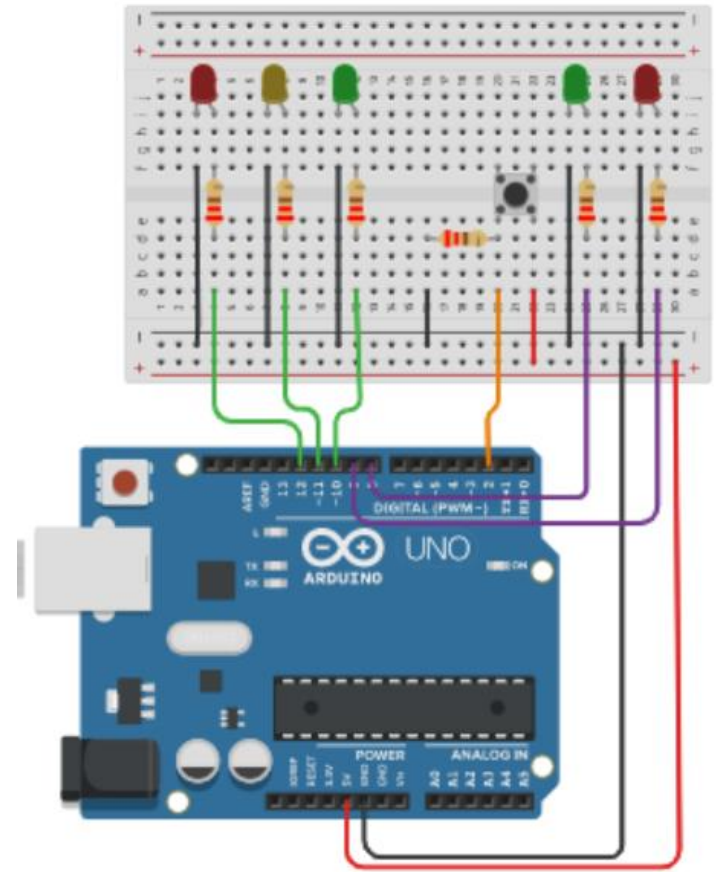
- O Semáforo opera normalmente para carros;
- Quando apertar o botão, o semáforo para carros deve fechar, e o de pedestres abrir por 5 segundos;
- Lembre de antes do vermelho, o led amarelo deve ser acionado!

2 – Implemente o projeto no TinkerCad, e tente incluir os conceitos de array aprendidos nessa aula.

3 – Monte o projeto na prática

Material necessário:

- 1 Arduino;
- 6 Resistores de 220 ohms
- 2 Led Vermelho;
- 1 Led Amarelo;
- 2 Led Verde;
- 1 Protoboard;
- Jumpers cables.



Link: [Projeto 05 – Interactive Traffic Lights](#)

Copyright © 2023

Prof. **Airton** / Prof. **Fabio** / Prof. **Lucas** / Prof. **Yan**

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).