# TRILHA DE KOTLIN





# **DADOS ALGORITMOS**







# **DADOS ALGORITMOS**

Dados nada mais que é algo do mundo real que pode ser representado computacionalmente. Por exemplo, os números que pertencem ao conjunto dos números inteiros, os números que pertencem ao conjunto dos números reais, letras, caracteres especiais, acentuação, pontuação, palavras, etc.

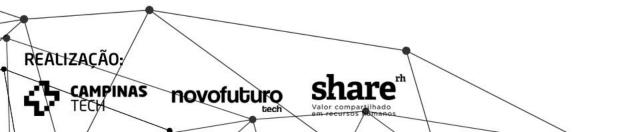






Para que seja possível armazenar e manipular dados no computador é necessário representá-los internamente de alguma forma. Nós seres humanos, representamos nossos números usando um sistema que chamamos de sistema decimal (ou sistema na base 10).

Esse sistema, que se originou do fato de utilizarmos os 10 dedos das mãos para realizarmos nossas contas, possui 10 diferentes dígitos para representar as infinitas quantidades e valores que desejamos (0 1 2 3 4 5 6 7 8 e 9).







Nos caso dos computadores digitais, a notação que é utilizada possui apenas 2 algarismos ou dígitos para representar uma quantidade desejada, o **0** e o **1**. Esse sistema de representação é chamado de sistema binário (ou sistema na base 2) e utiliza a noção de ligado/desligado, ou verdadeiro/falso, ou finalmente 0/1 .







Pelo fato de um número precisar de muitos algarismos para ser expresso no sistema binário, outras formas de representação auxiliares também são utilizadas nos computadores, como por exemplo:

a **representação pelo sistema hexadecimal** (ou sistema na base 16)

que utiliza 16 dígitos (0 1 2 3 4 5 6 7 8 9 A B C D E F), e a **representação no sistema octal** (ou sistema na base 8) que utiliza 8 dígitos (0 1 2 3 4 6 7 8).







### Referência para entendimento

Decimal Base-10	Binary Base-2	Octal Base-8	Hexa Decimal Base-16
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	В
12	1100	14	С
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10







O **ASCII** é um código que foi proposto por Robert W. Bemer como uma solução para unificar a representação de caracteres alfanuméricos em computadores. Antes de 1960 cada computador utilizava uma regra diferente para representar estes caracteres e o código ASCII nasceu para se tornar comum entre todas as máquinas.







O nome **ASCII** vem do inglês *American Standard Code for Information Interchange* ou "Código Padrão Americano para o Intercâmbio de Informação". Ele é baseado no alfabeto romano e sua função é padronizar a forma como os computadores representam letras, números, acentos, sinais diversos e alguns códigos de controle.







No **ASCII** existem apenas **95 caracteres** que podem ser impressos, eles são numerados de **32 a 126 sendo os caracteres** de **0 a 31 reservados para funções de controle**. Ou seja, funções de computador. Alguns caracteres acabaram caindo em desuso pois eram funções específicas para computadores da época como o Teletype (máquinas de escrever eletro-mecânicas), fitas de papel perfurado e impressoras de cilindro.







Tabela ASCII

ASCII control characters				
00	NULL	(Null character)		
01	SOH	(Start of Header)		
02	STX	(Start of Text)		
03	ETX	(End of Text)		
04	EOT	(End of Trans.)		
05	ENQ	(Enquiry)		
06	ACK	(Acknowledgement)		
07	BEL	(Bell)		
08	BS	(Backspace)		
09	HT	(Horizontal Tab)		
10	LF	(Line feed)		
11	VT	(Vertical Tab)		
12	FF	(Form feed)		
13	CR	(Carriage return)		
14	SO	(Shift Out)		
15	SI	(Shift In)		
16	DLE	(Data link escape)		
17	DC1	(Device control 1)		
18	DC2	(Device control 2)		
19	DC3	(Device control 3)		
20	DC4	(Device control 4)		
21	NAK	(Negative acknowl.)		
22	SYN	(Synchronous idle)		
23	ETB	(End of trans. block)		
24	CAN	(Cancel)		
25	EM	(End of medium)		
26	SUB	(Substitute)		
27	ESC	(Escape)		
28	FS	(File separator)		
29	GS	(Group separator)		
30	RS	(Record separator)		
31	US	(Unit separator)		
127	DEI	(Doloto)		

ASCII printable characters					
32	space	64	@	96	
33	!	65	A	97	a
34		66	В	98	b
35	#	67	C	99	C
36	\$	68	D	100	d
37	96	69	E	101	е
38	&	70	F	102	f
39		71	G	103	g
40	(	72	Н	104	h
41	)	73	1	105	i
42	*	74	J	106	i
43	+	75	K	107	k
44	,	76	L	108	1
45	-	77	M	109	m
46		78	N	110	n
47	1	79	0	111	0
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	S
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	V
55	7	87	W	119	W
56	8	88	X	120	X
57	9	89	Y	121	У
58	:	90	Z	122	Z
59	;	91	]	123	{
60	<	92	Ĭ	124	Ì
61	=	93	1	125	}
62	>	94	٨	126	~
63	?	95	_		









O "a" minúsculo que você digita no teclado é representado na tabela pelo decimal 97, mas internamente o computador trabalha isso em binário, ou seja, com 8 bits 01100001 (97 em decimal).

Faça esse teste: crie um arquivo de texto e coloque nele apenas a letra **a**. Depois de salvá-lo, o seu sistema operacional vai reconhecer que ele ocupa apenas **1 byte (8 bits)**:







Na década de 60 notaram a necessidade de normalizar a tabela para usar os 8 bits disponíveis (1 byte), que era o sistema mais usado na época. Criaram uma versão estendida da ASCII e então ela passou representar até 256 caracteres. Além dos caracteres da versão original, a estendida dispõe de alguns novos símbolos e caracteres gráficos:

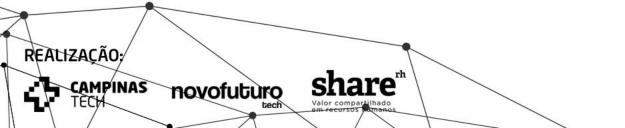






Tabela Extendida ASCII

Extended ASCII characters							
128	Ç	160	á	192	L	224	Ó
129	ű	161	ĺ	193		225	ß
130	é	162	Ó	194	т	226	Ô
131	â	163	ú	195	-	227	Ò
132	ä	164	ñ	196	_	228	ō
133	à	165	Ñ	197	+	229	Õ
134	å	166	a	198	ā	230	μ
135	ç	167	0	199	Ã	231	þ
136	ê	168	3	200	L	232	Þ
137	ë	169	®	201	F	233	Ú
138	è	170	7	202	1	234	Û
139	ï	171	1/2	203	TF	235	Ù
140	Î	172	1/4	204	T	236	ý
141	ì	173	i	205	=	237	Ý
142	Ä	174	««	206	#	238	-
143	Å	175	33	207		239	
144	É	176	1	208	ð	240	≡
145	æ	177	-	209	Đ	241	±
146	Æ	178		210	Ê	242	_
147	ô	179	T	211	Ë	243	3/4
148	Ö	180	+	212	È	244	1
149	ò	181	Á	213	1	245	8
150	û	182	Â	214	ĺ	246	÷
151	ù	183	À	215	Î	247	,
152	ÿ	184	©	216	Ϊ	248	0
153	Ö	185	4	217	7	249	**
154	Ü	186		218	Г	250	
155	Ø	187	j	219		251	1
156	£	188	Ţ	220		252	3
157	Ø	189	¢	221	Ī	253	2
158	×	190	¥	222		254	
159	f	191	٦	223		255	nbsp







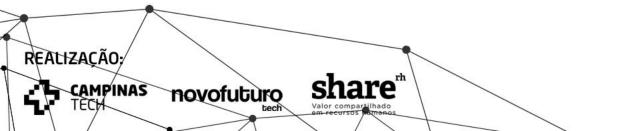
O problema, na tabela **ASCII** é que ela **não consegue representar os símbolos de outras línguas** como a hebraica, arábica, grega, japonesa







Foi então que no início dos anos 90 surgiu o **Unicode**, bem no inicio da internet, momento crucial em que a informática passou a necessitar de um padrão universal de representação de texto, para que o tráfego das informações ficasse inteligível em todos os computadores, não importando o idioma de escrita.







O **Unicode** faz parte do ISO 10646 e é o padrão de representação de texto mais usado. *Uni* em latim significa **um**, Unicode remete à ideia de um **padrão universal de codificação de caracteres.** 







Ele foi projetado com base no código ASCII, com a diferença que ele é capaz de codificar quase todos os caracteres e símbolos presentes nas mais variadas línguas do mundo ele consegue representar. Existem diversos sistemas de codificação Unicode e versao(junho 2020) eles conseguem representar mais de 143 mil caracteres. Unicode nos permite usar 8, 16 e até 32 bits (mais de 2 bilhões de caracteres possíveis).







Quando salvar um caracter com formato unicode:

Ele vai ocupar 4 bytes, ou seja, estará usando 32 bits.

Unicode mapeia cada caractere para um código específico chamado de **code point** que tem o formato

U+<código hexadecimal>

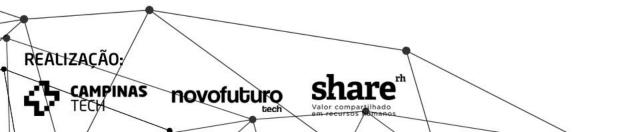
numa faixa de U+0000 a U+10FFFF.







Unicode	ASCII (decimal)	Caractere
U+0021	33	!
U+0024	36	\$
U+0026	38	&
U+002A	42	*



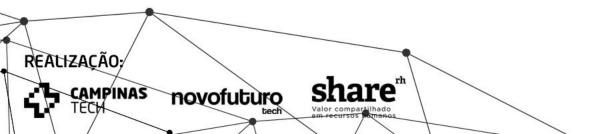




#### UTF-8

O encoding da família Unicode mais amplamente usado e difundido em sistemas operacionais, web e linguagens de programação é o UTF-8, que foi co-criado no início dos anos 90 Rob Pike (Desenvolvedor de linguagem).

Além disso, UTF-8 é o encoding padrão adotado pela W3C para a Web e ele é 100% compatível com a tabela ASCII.







#### UTF-8

Ele é um padrão de encoding variável e pode usar de 1 até 4 bytes, a depender do caractere que se está representando. Ele resolve o problema de não se fixar 32 bits para todos os caracteres, o que naturalmente faria um texto escrito em inglês/Português ocupar um espaço muito maior do que ele poderia ocupar.







#### UTF-8

No seu HTML quando você indica:

#### <meta charset="utf-8">

Você está dizendo pro navegador que sua página utiliza esse encoding.







#### UTF-8

Nas IDEs (integrated development environment) já vem por padrão configurado com UTF-8

Verificar no INTELLIJ







### **TIPOS DE DADOS PRIMITIVOS**

Tratando-se de tipos de dados primitivos, são constatados quatro formatos:

- Texto
- Inteiro
- Real
- lógico







### **TEXTO**

Os dados do tipo texto representam uma sequência de um ou mais caracteres. (0...9), alfabéticos (a...z,A...Z) e especiais (! @ # \$ % & \*)

Normalmente são colocados entre aspas duplas ("").

Exemplos de dado do tipo texto:

Nome (Marcos)

Cidade (Santo André)

Obs: espaços também contam como caracteres.







# **INTEIRO**

São representados por valores numéricos, tanto negativos, quando positivos (sem casas decimais).

Exemplos de dado do tipo inteiro:

Idade (18)

Materia (5)

Pote (3)

. . .





### **REAL**

Os dados do tipo real são valores numéricos, tanto positivos, quanto negativos que utilizam casas decimais,

Exemplos, podemos citar:

Salário (R\$1.100,00)

Valor Produto(R\$ 350,00)

Valor Total Circulo(8,3565)







# LÓGICO

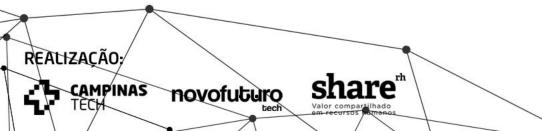
Por último, mas não menos importante, o tipo lógico. Este tipo de dado normalmente é representado por alternativas:

SIM ou NÃO, VERDADEIRO ou FALSO.

Dados do tipo lógico também podem são chamados de Booleano.

Exemplo de dado lógico que é frequentemente utilizado em programação é: Genero (Masculino ou Feminino)

Produto Vendido (verdadeiro/ falso ou sim/nao)







# **VARIÁVEIS E CONSTANTES**

Dentro de um algoritmo podemos encontrar basicamente duas classes diferentes de dados, os dados constantes e os variáveis.

Variável quando seu valor se altera conforme a necessidade da lógica do algoritmo.







# **VARIÁVEIS E CONSTANTES**

Constante quando seu valor não se altera ao longo do tempo em que o algoritmo é executado, ou seja, permanece o mesmo desde o início até o final da execução. Já um dado que pode ter seu valor alterado durante a execução do programa é tido como uma variável. Por exemplo, no cálculo da área de uma circunferência (A =  $\pi$  r2 ), o valor de  $\pi$  é constante, pois é sempre igual a 3.1416..., e o raio r é variável, pois pode assumir diferentes valores a cada cálculo da área. Seguindo a mesma lógica, a área A calculada para diferentes r também é variável.







```
var marca, modelo: caractere;
```

ano: inteiro;

preco: **real**;

vendido: lógico;

PI: real;

MAXIMO:inteiro;

**TECLA:** caractere







REGRA	EXEMPLO
Inicie sempre por um caractere alfabético, nunca por um número.	Nome (correto) - 1nome (errado)
Não utilize caracteres especiais como ", ( ) / *; +.	Nome (M); N*B
Não coloque espaços em branco ou hífen entre nomes.	salario-bruto
Utilize, se necessário, underline	salario_bruto
Crie suas variáveis com nomes sugestivos.	Se vai guardar salário de funcionários, dê à variável o nome <i>salario</i> .







# **ATRIBUIÇÃO**

Após realizada a declaração de um identificador, é possível iniciar a manipulação dos dados que esse identificador irá representar a partir da atribuição de valores ao mesmo. Esse processo de atribuir ou alterar o valor dos dados de um identificador é chamado de atribuição e é representado pelo símbolo ← quando estivermos trabalhando com identificadores variáveis, e pelo símbolo = quando estivermos trabalhando com identificadores constantes . A atribuição de valores a uma variável é feita da seguinte forma:



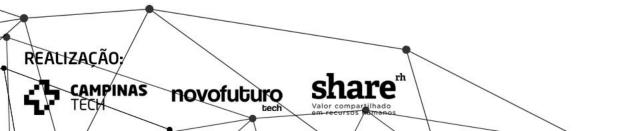




# **ATRIBUIÇÃO**

<identificador da variável> ← <valor do mesmo tipo da variável>

<identificador da variável> ← <operações cujo resultado é o mesmo da varialvel>







Como pode ser visto acima, no lado esquerdo do operador ← será colocado o nome da variável que irá receber o valor, e do lado direito o valor que será armazenado na mesma. A seguir são apresentados alguns exemplos de atribuições de valores a variáveis:

TipoVeiculo ← motocicleta;

Aceleracao  $\leftarrow$  15.52;

Massa  $\leftarrow$  12.3;

Forca ← Massa \* Aceleracao;

Usado  $\leftarrow$  F;







Uma varíavel pode armazenar apenas um único valor por vez, sendo que sempre que um novo valor é atribuído a variável o valor anterior que estava armazenado na mesma é perdido. Por exemplo, consideremos o algoritmo a seguir:

Algoritmo Atribuições de valores a uma variável.

var numero: inteiro

numero ← 222

numero ← 1000

numero ← 23







REALIZAÇÃO:

CAMPINAS novofuturo

share the valor compartified of the valor co

PATROCÍNIO:

