

# ARQUITETURA DE COMPUTADORES



## AULA 06



AGENDA

## Representação de dados

**Objetivo: Conhecer a representação de dados no formato interno dos sistemas computacionais**

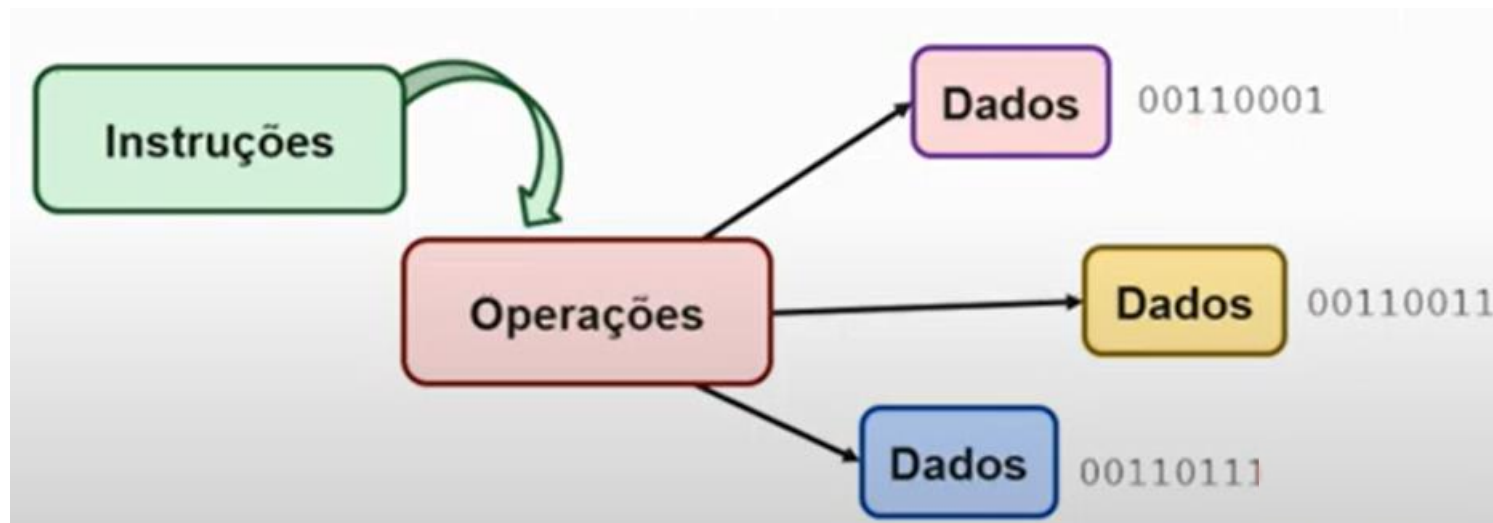
- Formas de representação
- Tipos de dados



## PROCESSAMENTO DE DADOS

Um computador funciona por meio da execução sistemática de instruções que o orientam a realizar algum tipo de operação sobre valores (numéricos, alfanuméricos ou lógicos). Esses valores são genericamente conhecidos como **dados**.

(MONTEIRO, 2007)





## PROCESSO DE CONVERSÃO DE DADOS

Os dados são convertidos internamente em um código de armazenamento no formato binário.





# ARQUITETURA DE COMPUTADORES



## PROCESSO DE CONVERSÃO DE DADOS

### Notação simbólica

```
number = 10

Se number > 0 então
    writeln("is positive")
senão
    writeln("is negative")
```

Linguagem natural  
Linguagem de alto nível  
Alto nível de abstração

Linguagem de máquina  
Linguagem de baixo nível  
Baixo nível de abstração

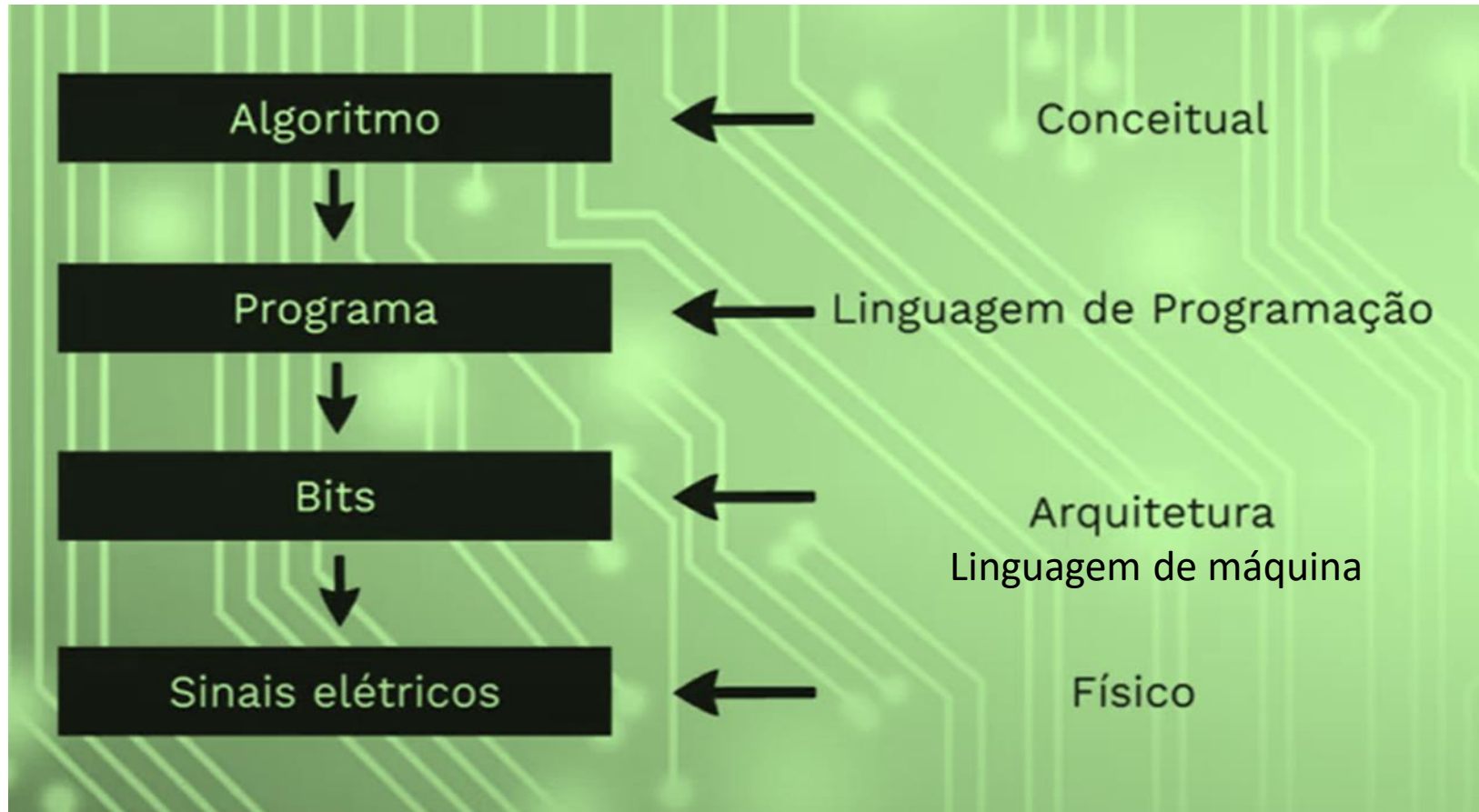
```
01010010 (82) = 10 em binário
00000010 (2) = 0 em binário
00000101 (5) = positivo em binário
00000011 (3) = negativo em binário
```

Números binários

# ARQUITETURA DE COMPUTADORES



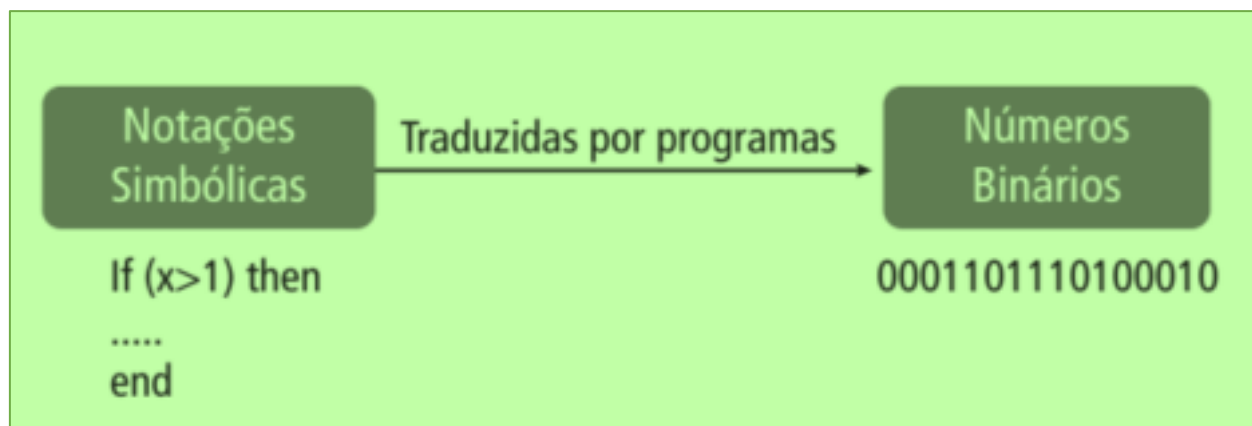
## PROCESSO DE CONVERSÃO DE DADOS





## PROCESSO DE CONVERSÃO DE DADOS

Qualquer que tenha sido a linguagem de programação utilizada para escrever o programa, ela deverá ser convertida para **código-objeto** (**código binário**) e, em seguida, para o código executável (conjunto de **códigos de máquina**), o qual é gerado pelo **compilador** da linguagem.



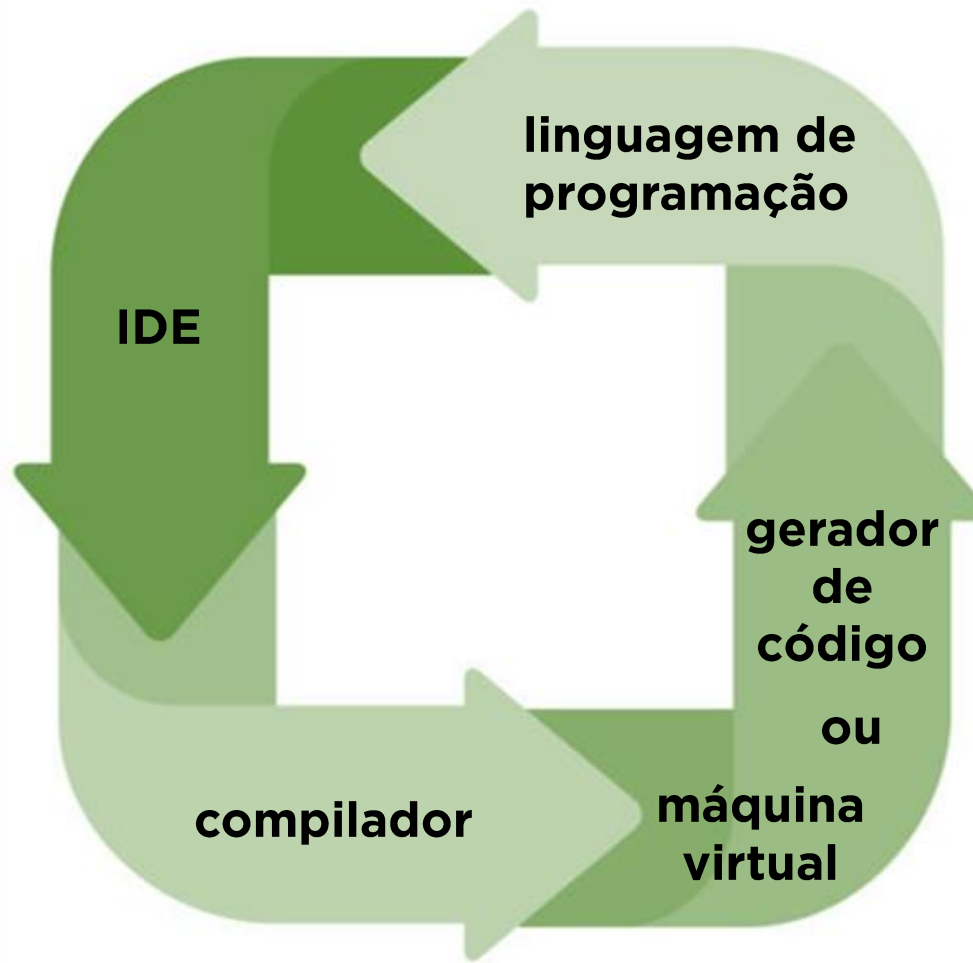
Essa conversão também inclui dados, que deverão ser alterados de modo a estarem em uma forma apropriada para utilização pela ULA (ex.: números inteiros ou fracionários).

Por exemplo, para efetivar uma soma, a ULA executa, passo a passo, uma série de micro-operações (um algoritmo): verificar o sinal dos números, verificar o tipo do número, etc.



## PROCESSO DE CONVERSÃO DE DADOS

O que é preciso para se fazer um programa de computador?







## PROCESSO DE CONVERSÃO DE DADOS

**O que é preciso para se fazer um programa de computador?**

- ❑ uma **linguagem de programação**: regras léxicas e sintáticas para se escrever o programa



## PROCESSO DE CONVERSÃO DE DADOS

**O que é preciso para se fazer um programa de computador?**

- ☐ uma **linguagem de programação**: regras léxicas e sintáticas para se escrever o programa
- ☐ uma **IDE**: Ambiente Integrado de Desenvolvimento, software para editar e testar o programa



## PROCESSO DE CONVERSÃO DE DADOS

**O que é preciso para se fazer um programa de computador?**

- ☐ uma **linguagem de programação**: regras léxicas e sintáticas para se escrever o programa
- ☐ uma **IDE**: Ambiente Integrado de Desenvolvimento, software para editar e testar o programa
- ☐ um **compilador**: software para transformar o código fonte em código objeto



## PROCESSO DE CONVERSÃO DE DADOS

O que é preciso para se fazer um programa de computador?

- ❑ uma **linguagem de programação**: regras léxicas e sintáticas para se escrever o programa
- ❑ uma **IDE**: Ambiente Integrado de Desenvolvimento, software para editar e testar o programa
- ❑ um **compilador**: software para transformar o código fonte em código objeto

### Relembrando:

**Código fonte**: é aquele escrito pelo programador em linguagem de programação. Não é entendido pelo computador nem pelo sistema operacional. Ele tem que ser convertido em um código que possa ser executável (COMPILAÇÃO).





## PROCESSO DE CONVERSÃO DE DADOS

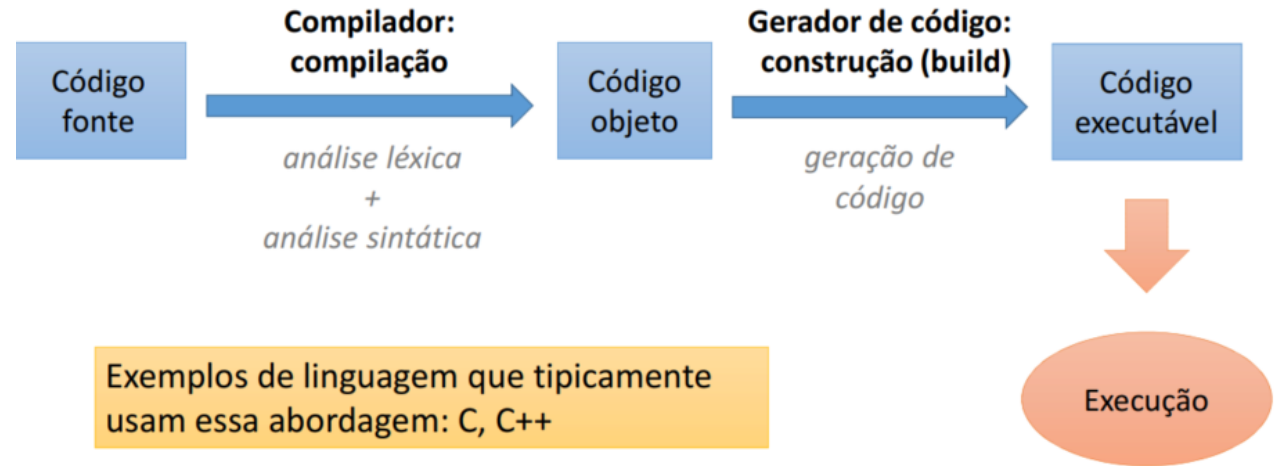
O que é preciso para se fazer um programa de computador?

- ☐ uma **linguagem de programação**: regras léxicas e sintáticas para se escrever o programa
- ☐ uma **IDE**: Ambiente Integrado de Desenvolvimento, software para editar e testar o programa
- ☐ um **compilador**: software para transformar o código fonte em código objeto
- ☐ um **gerador de código** ou **máquina virtual**: software que permite que o programa seja executado (permite o código objeto ser executado).

# ARQUITETURA DE COMPUTADORES



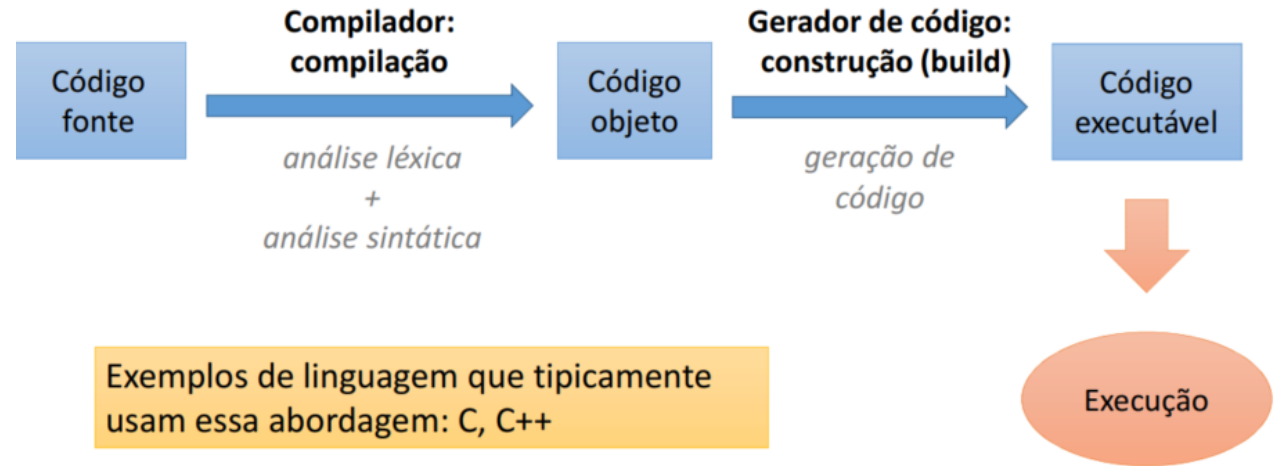
## COMPILAÇÃO



# ARQUITETURA DE COMPUTADORES



## COMPILAÇÃO



## INTERPRETAÇÃO

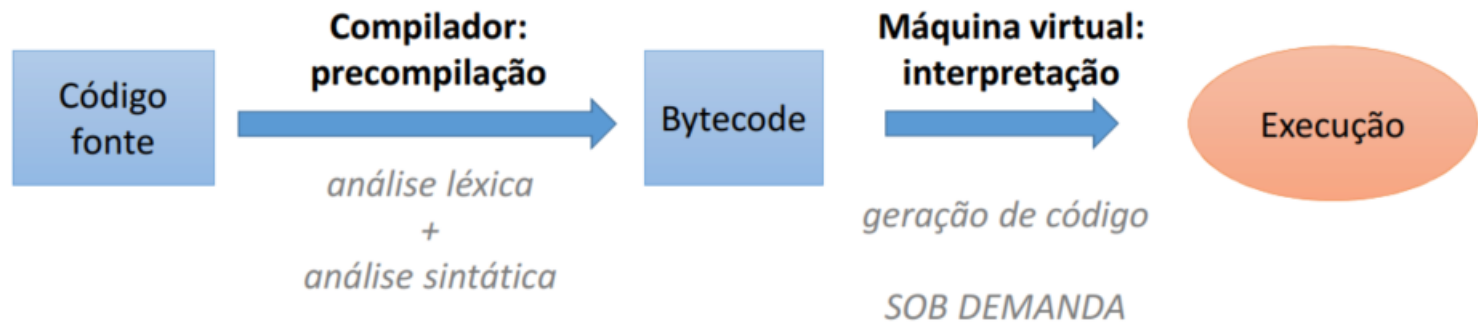


Exemplos de linguagem que tipicamente usam essa abordagem: PHP, JavaScript, Python, Ruby

# ARQUITETURA DE COMPUTADORES



## ABORDAGEM HÍBRIDA



Exemplos de linguagem que tipicamente usam essa abordagem: Java (JVM), C# (Microsoft .NET Framework)

Maquina virtual que executa

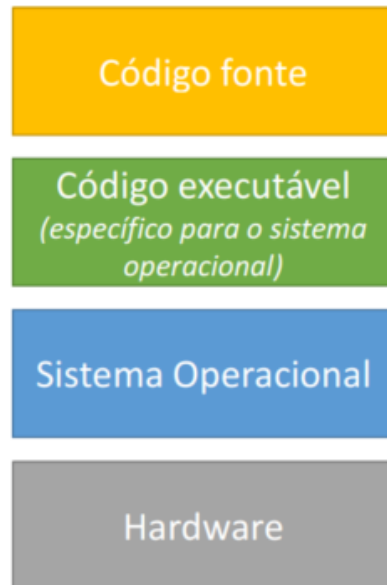
Maquina virtual que executa



# ARQUITETURA DE COMPUTADORES

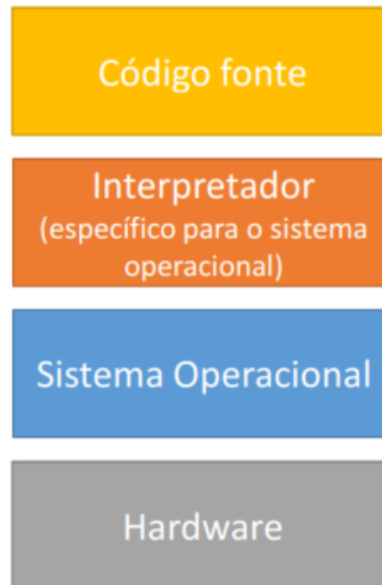


## COMPILAÇÃO



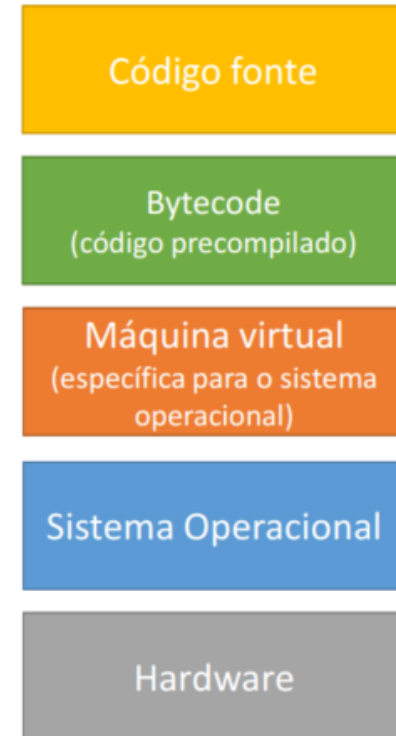
C / C++

## INTERPRETAÇÃO



PHP, Python, JavaScript

## ABORDAGEM HÍBRIDA



Java, C#



## REPRESENTAÇÃO DE DADOS

As **diferentes formas de representação** e respectivos algoritmos de realização das operações matemáticas são muito úteis, pois cada uma tem uma aplicação mais vantajosa que a outra.



## REPRESENTAÇÃO DE DADOS

As **diferentes formas de representação** e respectivos algoritmos de realização das operações matemáticas são muito úteis, pois cada uma tem uma aplicação mais vantajosa que a outra.

Cabe ao programador a escolha da forma a ser utilizada pelo sistema, conforme a própria plataforma de desenvolvimento, podendo a representação ser explícita ou implícita.

- ☐ **Explícita (fortemente tipada)**: quando o programador define as variáveis e constantes em seu programa.
- ☐ **Implícita (fracamente tipada)**: quando é deixado para que o compilador faça a escolha.





## TIPOS DE DADOS

**Definem para o sistema como cada dado deverá ser manipulado, pois conforme citado anteriormente, cada tipo de dado recebe um tratamento diferenciado pelo processador**

**Exemplo:**

**VAR X = INTEGER;      TIPO INTEIRO**

**VAR X = REAL;        TIPO PONTO FLUTUANTE**

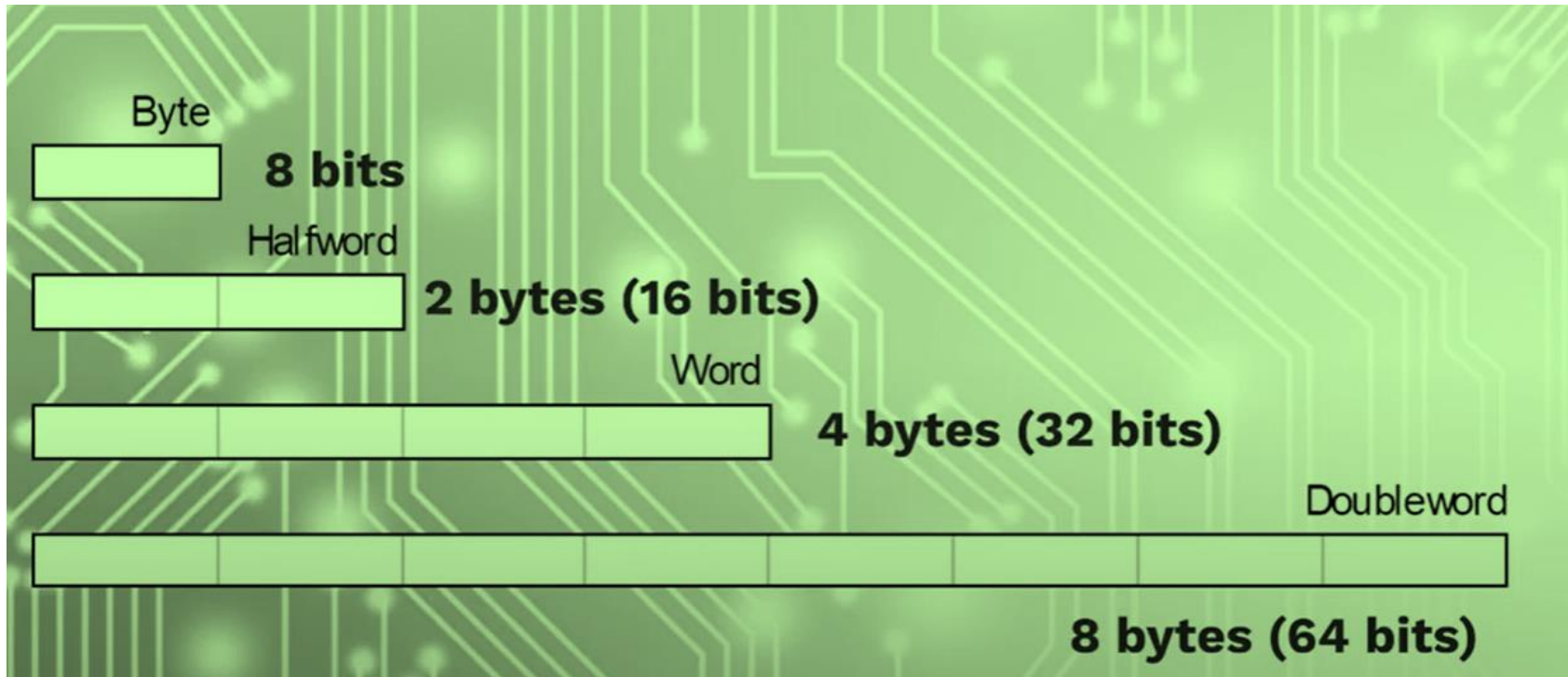


# ARQUITETURA DE COMPUTADORES



## TIPOS DE DADOS

DEFINIDOS PELO TAMANHO E PELA ESTRUTURA UTILIZADA PARA SUA CRIAÇÃO PARA SUA UTILIZAÇÃO





## TIPOS DE DADOS

### Escalares



**Caracteres**

**1011  
001**

**Números**



**Dados Lógicos**

### Estruturados



**Estáticos**



**Dinâmicos**



## TIPOS DE DADOS

De modo geral, as seguintes formas de dados são mais utilizadas nos programas atuais de computadores (**formas primitivas**, entendidas pelo hardware):

- ❑ **TIPO CARACTERE** (dados sob forma de caractere);
- ❑ **TIPO LÓGICO** (dados sob forma lógica);
- ❑ **TIPO NUMÉRICO** (dados sob forma numérica).



## TIPOS DE DADOS

De modo geral, as seguintes formas de dados são mais utilizadas nos programas atuais de computadores (**formas primitivas**, entendidas pelo hardware):

- ❑ **TIPO CARACTERE** (dados sob forma de caractere);
- ❑ **TIPO LÓGICO** (dados sob forma lógica);
- ❑ **TIPO NUMÉRICO** (dados sob forma numérica).

Outras formas mais complexas são permitidas em certas linguagens modernas (como **tipo REGISTRO**, **tipo ARRAY**, **tipo INDEX**, **tipo POINTER** etc.). No entanto, durante o processo de compilação, os dados acabam sendo convertidos finalmente nas formas primitivas já mencionadas, para que o hardware possa executá-las.

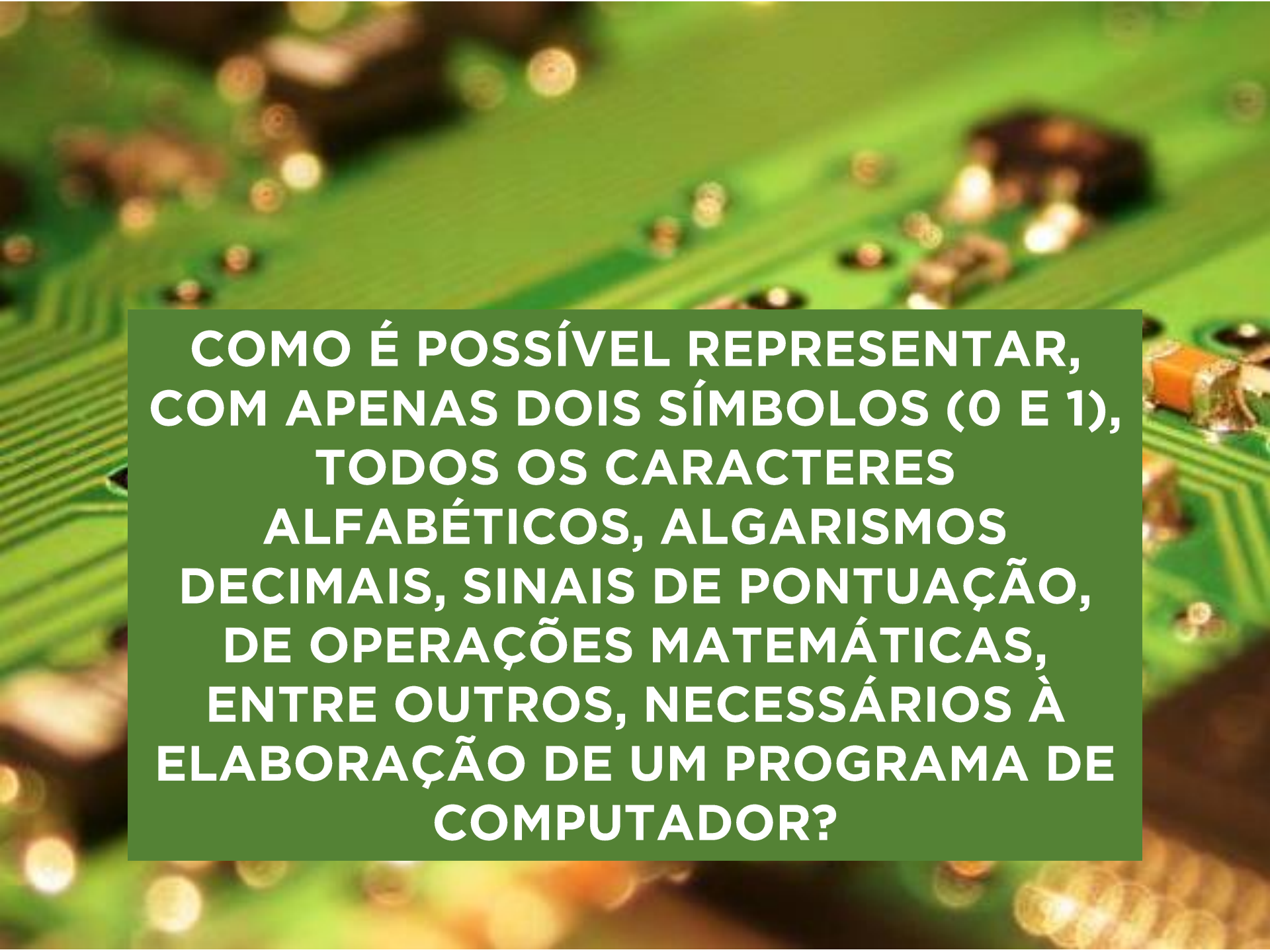





## TIPOS DE DADOS

**A representação interna de informações em um computador é realizada através de uma correspondência entre o símbolo da informação e o grupo de algarismos binários (bits). Cada símbolo (caractere, número ou símbolo) possui uma identificação específica.**

**Exemplo: Símbolo “A”  $\Rightarrow$  Algarismos binários “10101101”**



**COMO É POSSÍVEL REPRESENTAR,  
COM APENAS DOIS SÍMBOLOS (0 E 1),  
TODOS OS CARACTERES  
ALFABÉTICOS, ALGARISMOS  
DECIMAIS, SINAIS DE PONTUAÇÃO,  
DE OPERAÇÕES MATEMÁTICAS,  
ENTRE OUTROS, NECESSÁRIOS À  
ELABORAÇÃO DE UM PROGRAMA DE  
COMPUTADOR?**



**PELA UTILIZAÇÃO DO MÉTODO  
CHAMADO DE **CODIFICAÇÃO**, PELO  
QUAL CADA SÍMBOLO DA NOSSA  
LINGUAGEM TEM UM  
CORRESPONDENTE GRUPO DE BITS  
QUE IDENTIFICA UNIVOCAMENTE O  
REFERIDO SÍMBOLO (CARACTERE).**



## TIPOS DE DADOS

**Portanto, existem alguns padrões de codificação previamente definidos.**

### Códigos Caracteres

BCD – *Binary Code Decimal*

Utiliza 6 *bits*/caracteres, codificando 64 caracteres.

EBCDIC – *Extended Binary Coded Decimal Interchange Code*

Exclusivo da IBM, utilizando 8 *bits* para codificar 256 caracteres.

ASCII – *American Standart Code for Information Interchange*

ASCII – usado pelos demais fabricantes. Utiliza oito *bits*/caractere em sua versão estendida, codificando 256 caracteres.

UNICODE

Código que utiliza 16 *bits*/símbolo, podendo representar 65.536 símbolos diferentes. Pretende codificar em um único código os símbolos de todas as linguagens conhecidas no mundo. Está sendo desenvolvido por um consórcio desde 1991 ([www.unicode.org](http://www.unicode.org)).

Fonte: Adaptada de Monteiro (2007)





## TIPOS DE DADOS

**Portanto, existem alguns padrões de codificação previamente definidos.**

A utilização de padrões de codificação (ex.: ASCII, Unicode) é o método primário de introdução de informações no computador. As demais formas de representação de informação (tipos de dados) surgem no decorrer do processo de compilação ou interpretação do programa.

O padrão de codificação mais utilizado pela indústria de computadores é o **ASCII**. A codificação correspondente a esse padrão já é parte do *hardware* (armazenado em uma memória do tipo ROM) e é definida pelo próprio fabricante.



# ARQUITETURA DE COMPUTADORES



## ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	



<https://www.youtube.com/watch?v=uIEkazak3JQ>

<https://www.youtube.com/watch?v=rjBuymn8Gm8>



## TIPOS DE DADOS

### TIPO CARACTERE (Character Data Types);

- ☐ Caractere (**Char**): Armazena um único caractere, como letras, números ou símbolos.
- ☐ Cadeia de Caracteres (**String**): Armazena uma sequência de caracteres, como palavras ou frases.



## TIPOS DE DADOS

### TIPO CARACTERE (Character Data Types);

- ❑ **Caractere (Char):** Armazena um único caractere, como letras, números ou símbolos.

O tipo de dado caractere é usado para representar um único caractere alfanumérico.

Geralmente, é representado entre aspas simples ou aspas duplas.

Exemplos de caracteres são: 'A', 'b', '5', '@'



## TIPOS DE DADOS

### TIPO CARACTERE (Character Data Types);

- ❑ **Caractere (Char):** Armazena um único caractere, como letras, números ou símbolos.

```
var  
    letra: caractere  
  
inicio  
    letra <- 'A'  
    escreva(letra)  
finalgoritmo
```







## TIPOS DE DADOS

### TIPO CARACTERE (Character Data Types);

- ❑ **Cadeia de Caracteres (**String**):** Armazena uma sequência de caracteres, como palavras ou frases.

O **tipo de dado string** é usado para representar uma sequência de caracteres.

É usado para armazenar palavras, frases ou qualquer texto.

As **strings** são geralmente representadas entre aspas simples ou aspas duplas.

Exemplos de strings são: "Olá, mundo!", "Marcel Rios", "37".

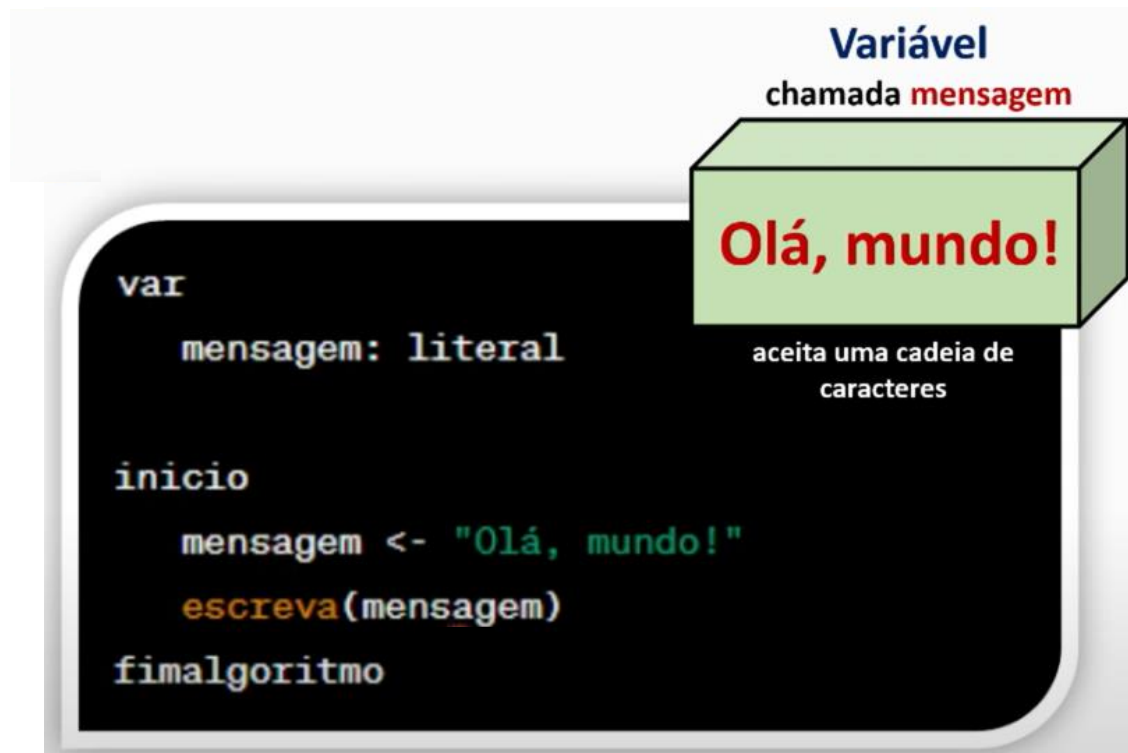




## TIPOS DE DADOS

### TIPO CARACTERE (Character Data Types);

- ❑ Cadeia de Caracteres (**String**): Armazena uma sequência de caracteres, como palavras ou frases.





## TIPOS DE DADOS

### TIPO LÓGICO (Boolean Data Types);

- ❑ **Booleano (Boolean):** armazena valores falsos (false) (Bit 0) ou verdadeiros (true) (Bit 1)

O **tipo de dado booleano** é usado para representar um valor lógico, que pode ser verdadeiro (true) ou falso (false).

É usado principalmente em expressões condicionais e lógicas.

Exemplos de booleanos são: true, false.



## TIPOS DE DADOS

### TIPO LÓGICO (Boolean Data Types);

- ❑ **Booleano (Boolean):** armazena valores falsos (false) (Bit 0) ou verdadeiros (true) (Bit 1)





## Operadores lógicos e suas funções

### Porta Lógica

### Definição

(N) AND O operador lógico AND é definido de modo que o resultado da operação com ele será VERDADE se e somente se todas as variáveis de entrada forem VERDADE ( $=1$ ). Caso contrário, o resultado será FALSO ( $=0$ ).

(N) OR O resultado da operação será VERDADE ( $=1$ ) se um operando (ou variável lógica) ou o outro for verdadeiro. Basta que apenas um dos operandos seja verdadeiro. Caso contrário, o resultado será FALSO ( $=0$ ). Operadores lógicos OR também são largamente utilizados em lógica digital ou na definição de condições em comandos de decisão de certas linguagens de programação.

NOT É definido de modo a produzir na saída um resultado de valor oposto (ou inverso) ao da variável de entrada. É usado apenas com uma única variável. Desse modo, se a variável tem o valor 0 (FALSO), o resultado da operação NOT sobre essa variável será 1 (VERDADE), e se a variável for igual a 1 (VERDADE), então o resultado do NOT será 0 (FALSO).

XOR O operador lógico XOR (EXCLUSIVE-OR) ou OU EXCLUSIVO é definido de modo a prover um resultado VERDADEIRO se apenas uma das variáveis ou operadores for VERDADEIRA. Sendo  $X=A \text{ XOR } B$ , o resultado X será VERDADE se exclusivamente (daí o nome OU EXCLUSIVO) A OU B for VERDADE. Caso ambos sejam "VERDADE" ou ambos "FALSO", então o resultado será FALSO.





## TIPOS DE DADOS

### **TIPO NUMÉRICO** (Numeric Data Types);

**Como os computadores são elementos binários, a forma mais eficiente de representar números deve ser “binária”, isto é, converter o número diretamente de decimal para seu correspondente valor binário. Deste modo a ULA poderá executar as operações mais rapidamente.**

**Alguns problemas devem ser considerados em relação à representação do tipo numérico, como:**

- **a representação do sinal do número,**
- **a representação do ponto ou vírgula do número e**
- **o tamanho dos registradores da UCP, os quais limitam o tamanho dos números que poderão ser representados pelo hardware.**





## TIPOS DE DADOS

### TIPO NUMÉRICO (Numeric Data Types);

- ❑ **Inteiro (Integer):** Armazena números inteiros positivos ou negativos, como 1, -5, 1000, etc.
- ❑ **Real ou Ponto Flutuante (Floating-Point):** Armazena números com casas decimais, como 3.14, -0.005, 123.456, etc.



## TIPOS DE DADOS

### TIPO NUMÉRICO (Numeric Data Types);

- ❑ **Inteiro (Integer):** Armazena números inteiros positivos ou negativos, como 1, -5, 1000, etc.

O tipo de dado inteiro é usado para representar números inteiros, ou seja, números sem parte decimal.

Pode ser representado em diferentes tamanhos, como 8 bits, 16 bits, 32 bits ou 64 bits, dependendo da linguagem. Exemplos de inteiros são: -10, 0, 42



## TIPOS DE DADOS

### TIPO NUMÉRICO (Numeric Data Types);

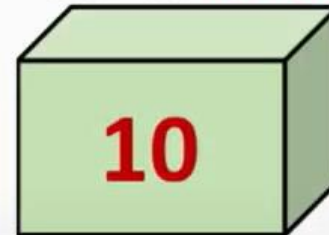
- ❑ **Inteiro (Integer):** Armazena números inteiros positivos ou negativos, como 1, -5, 1000, etc.

```
var  
    numero: inteiro
```

```
inicio  
    numero <- 10  
    escreva(numero)  
fimalgoritmo
```

**Variável**

chamada **numero**



apenas números  
inteiros



## TIPOS DE DADOS

### TIPO NUMÉRICO (Numeric Data Types);

- ❑ **Real ou Ponto Flutuante (Floating-Point):** Armazena números com casas decimais, como 3.14, -0.005, 123.456, etc.

O **tipo de dado ponto flutuante** é usado para representar números com parte decimal.

É útil quando precisamos de valores fracionários.

Os tipos de ponto flutuante mais comuns são **float** e **double**, que diferem em precisão e faixa de valores que podem representar. Exemplos de ponto flutuante são: 3.14, -0.5, 1.8e10 (notação científica).



## TIPOS DE DADOS

### TIPO NUMÉRICO (Numeric Data Types);

- ❑ **Real ou Ponto Flutuante (Floating-Point):** Armazena números com casas decimais, como 3.14, -0.005, 123.456, etc.

```
var  
    valor: real  
  
inicio  
    valor <- 3.14  
    escreva(valor)  
fimalgoritmo
```







## TIPOS DE DADOS

**“Definem para o sistema como cada dado deverá ser manipulado, pois conforme citado anteriormente, cada tipo de dado recebe um tratamento diferenciado pelo processador”**

# ARQUITETURA DE COMPUTADORES



REVISANDO...



## REVISANDO...

**A representação de dados esclarece o formato de representação dos diversos tipos de dados que o computador pode receber como entrada - seja a partir da execução de programas ou a partir de dispositivos de entrada e saída.**

**Os principais tipos de dados compreendidos pelo computador, também chamados de tipos primitivos de dados, são: caráter, lógico e numérico. A partir desses tipos básicos foram desenvolvidos alguns tipos considerados mais complexos e que não são compreendidos diretamente pelo *hardware* (ex.: vetor, índice, registro, ponteiros), sendo necessário para isso que tais tipos sejam convertidos para formatos primitivos equivalentes.**

**A tabela ASCII é o padrão de codificação utilizado para a conversão de caracteres, números e símbolos inseridos no computador via dispositivos de entrada e saída (ex.: teclado). Assim, cada caractere possui um código binário único que o representa.**

A close-up, macro photograph of a green printed circuit board (PCB). The board is populated with various electronic components, including several black integrated circuits (chips) and smaller surface-mount components. Some components have markings like "29U15", "102", and "E37E". The green color of the PCB is vibrant, and the intricate circuit patterns are visible. The background is softly blurred, showing more components and the overall texture of the board.

**ATÉ A PRÓXIMA!**