

**E-BOOK**  
**E-MAILS DO**  
**MINICURSO**  
**DE**  
**LÓGICA**  
**DE**  
**PROGRAMAÇÃO**

# Sumário

[E-mail 1 - Por que aprender programação?](#)

[E-mail 2 - Criando os seus primeiros programinhas.](#)

[E-mail 3 - Variáveis, constantes e tipos de dados.](#)

[E-mail 4 - Operadores](#)

[E-mail 5 - Tomando decisões!](#)

[E-mail 6 - Resposta do exercício "para-casa" da Aula \(5/10\) Tomando decisões!](#)

[E-mail 7 - Tomando decisões!](#)

[E-mail 8 - Resposta do exercício da Aula \(6/10\)](#)

[E-mail 9 - Sacada! Solução do exercício de ontem sem usar ESCOLHA-CASO ou SE-ENTÃO-SENÃO](#)

[E-mail 10 - Loops Básicos!](#)

[E-mail 11 - Resposta do exercício da Aula \(7/10\)](#)

[E-mail 12 - Loops Pré-definido](#)

[E-mail 13 - Algoritmo números primos: Exercício da Aula \(8/10\)](#)

[E-mail 14 - Vetores e Matrizes](#)

[E-mail 15 - Jogo da Velha: Exercício da Aula \(9/10\)](#)

[E-mail 16 - Funções e Procedimentos](#)

[E-mail 17 - Jogo da Velha com funções e procedimentos](#)

# E-mail 1 - Por que aprender programação?

→ [Clique AQUI para ler a primeira aula!](#)

Talvez você já tenha acessado esta primeira aula, mas estou enviando este e-mail só para garantir que você realmente recebeu o link para a primeira aula do minicurso de lógica de programação.

Nesta aula #1 do **minicurso de lógica de programação** você irá aprender:

1. 4 Motivos para você começar aprender programação AGORA!
2. Se precisa ou não saber inglês para aprender programação.
3. Com qual linguagem de programação você deve começar.

→ [Acessar a primeira aula do minicurso GRÁTIS!](#)

Espero que goste desta primeira aula.

Importante! As demais aulas serão enviadas automaticamente a cada 24 horas. Você receberá a próxima aula amanhã neste mesmo horário.

Bons estudos!

# E-mail 2 - Criando os seus primeiros programinhas.

→ [Clique AQUI para ler a aula #2](#)

Nesta aula #2 do **minicurso de lógica de programação** você irá aprender:

1. O que é um Algoritmo
2. A melhor ferramenta para aprender lógica de programação
3. Criar os seus primeiros programas

→ [Acessar a segunda aula do minicurso GRÁTIS!](#)

Espero que goste desta segunda aula.

# E-mail 3 - Variáveis, constantes e tipos de dados.

[→ Clique AQUI para ler a aula #3](#)

Nesta aula #3 do **minicurso de lógica de programação** você irá aprender dois assuntos básicos, mas muito importantes para você se tornar um bom programador.

Você vai aprender como armazenar dados na memória do computador.

Também aprenderá quais são os tipos de dados que podemos usar nos nossos algoritmos, desde os tipos de dados primitivos até os tipos de dados customizados.

[→ Acessar a terceira aula do minicurso GRÁTIS!](#)

Além disso no final da aula você verá a solução do exercício que eu pedi para você fazer ontem.

E aí, está gostando deste minicurso? Convide os seus amigos para se inscreverem neste minicurso 100% grátis de lógica de programação.

Boa aula!

# E-mail 4 - Operadores

[→ Clique AQUI para ler a aula #4](#)

Chegamos a aula #4 do **minicurso de lógica de programação**.

Hoje é dia de você aprender a usar os três tipos operadores: **aritméticos, lógicos e relacionais**

Um dos grandes motivos dos iniciantes em programação se enrolarem no começo é não entender as expressões, como os valores se relacionam e o resultado esperado de operações.

Precisa praticar bastante. É como aprender a fazer contas na escola. Prática!

[→ Acessar a quarta aula do minicurso GRÁTIS!](#)

Boa aula!

# E-mail 5 - Tomando decisões!

→ [Clique AQUI para ler a aula #5](#)

Nesta aula #5 do **minicurso de lógica de programação** vamos aprender a deixar os nossos programas tomando decisões sozinhos.

Você vai aprender a utilizar a estrutura de decisão **SE-ENTÃO-SENÃO**.

Esta é a estrutura de controle mais básica da lógica de programação.

Para aprender bem, é preciso praticar bastante. Ao final desta você terá um exercício prático. Amanhã eu envio a resposta para você conferir, mas é muito importante que você tente resolvê-lo sozinho antes.

→ [Acessar a quinta aula do minicurso GRÁTIS!](#)

Boa aula!

# E-mail 6 - Resposta do exercício "para-casa" da Aula (5/10) Tomando decisões!

Ontem eu enviei pra você a aula #5 do **minicurso de lógica de programação**. No final da aula eu pedi pra você tentar resolver um exercício de lógica para verificar se um aluno foi **aprovado** ou **reprovado** no final do ano.

Você fez? Espero que sim! Teve alguma dificuldade? Bom, abaixo eu mostro como eu escrevi um algoritmo para resolver esse exercício. Compare com o que você fez. Se o seu não deu certo, continue lendo que eu explico cada parte do algoritmo.

Esse é o algoritmo.

---

```
algoritmo "AprovacaoFinalDeAno"
var
    nota1, nota2, nota3, nota4, media: real
inicio
    escreva("Informe a nota (de 0 a 10) do primeiro bimestre: ")
    leia(nota1)
    escreva("Informe a nota (de 0 a 10) do segundo bimestre: ")
    leia(nota2)
    escreva("Informe a nota (de 0 a 10) do terceiro bimestre: ")
    leia(nota3)
    escreva("Informe a nota (de 0 a 10) do quarto bimestre: ")
    leia(nota4)
    media := (nota1 + nota2 + nota3 + nota4) / 4
    escreval("Sua média foi: ", media)

    se media >= 6 entao
        escreva("Você foi APROVADO!")
    senao
        escreva("Você foi REPROVADO!")
```



```
    fimse  
fimalgoritmo
```

---

## Entendendo o algoritmo.

Primeiro eu declarei 5 variáveis do tipo **REAL**. Elas têm que ser do tipo **REAL** porque as notas podem ter valores decimais, por exemplo **5.5**.

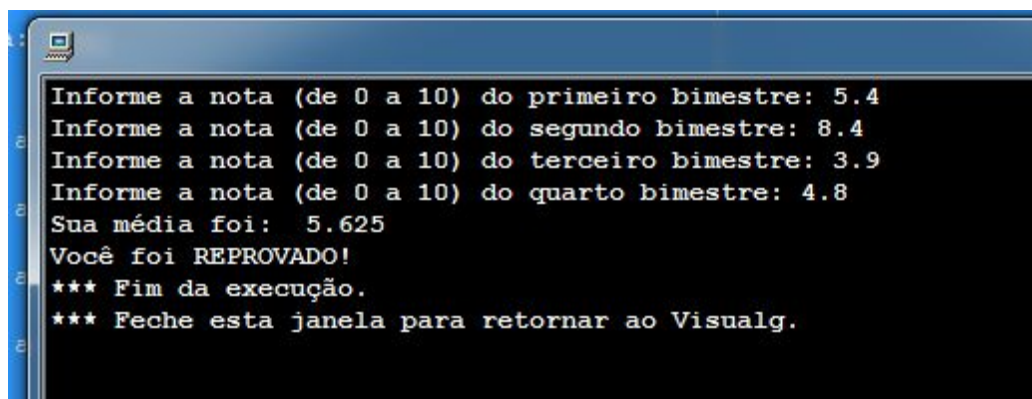
Depois eu escrevi na tela "**Digite a nota (de 0 a 10) do primeiro bimestre:** " e armazenei na variável nota1 o valor que o usuário digitou. Fiz o mesmo para as outras 3 notas.

Na sequência eu calculei a média das 4 notas e armazenei o resultado na variável "media". Importante colocar o parênteses para somar as notas ANTES de dividir por 4.

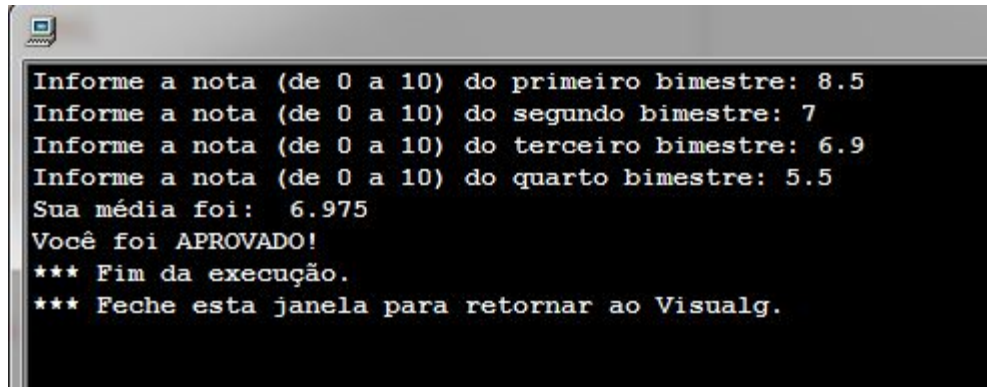
Agora que vem a parte da decisão, o SE-ENTÃO-SENÃO.

Eu verifiquei se a média é MAIOR OU IGUAL a 6. Se SIM **ENTÃO** imprimi na tela a mensagem informando que o aluno foi aprovado. **SENÃO** imprimi a mensagem informando que o aluno foi reprovado.

Veja abaixo o resultado da execução do algoritmo no **Visualg**, quando a média era menor que 6 e quando foi maior.



```
Informe a nota (de 0 a 10) do primeiro bimestre: 5.4  
Informe a nota (de 0 a 10) do segundo bimestre: 8.4  
Informe a nota (de 0 a 10) do terceiro bimestre: 3.9  
Informe a nota (de 0 a 10) do quarto bimestre: 4.8  
Sua média foi: 5.625  
Você foi REPROVADO!  
*** Fim da execução.  
*** Feche esta janela para retornar ao Visualg.
```



```
Informe a nota (de 0 a 10) do primeiro bimestre: 8.5
Informe a nota (de 0 a 10) do segundo bimestre: 7
Informe a nota (de 0 a 10) do terceiro bimestre: 6.9
Informe a nota (de 0 a 10) do quarto bimestre: 5.5
Sua média foi: 6.975
Você foi APROVADO!
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

Viu como foi simples? Se você teve dificuldades para resolver, não se preocupe. No início parece difícil mesmo. Mas como sempre digo, **é preciso praticar!**

Se conseguiu resolver sem dificuldades **ótimo**, mas continue praticando.

Amanhã vou te enviar a aula #6 do **minicurso de lógica de programação** fique atento à sua caixa de entrada!

Até lá!

# E-mail 7 - Tomando decisões!

→ [Clique AQUI para ler a aula #6](#)

Nesta aula #6 do **minicurso de lógica de programação** vamos aprender uma estrutura elegante para fazer nossos programas tomar decisões quando temos muitas opções.

Você também vai aprender a diferença de um HUB e um SWITCH.

A estrutura de controle de fluxo que vamos estudar hoje é a **ESCOLHA-CASO**.

Você vai perceber a diferença gritante na legibilidade do código entre o SE-ENTÃO-SENÃO e o ESCOLHA-CASO quando temos muitas opções de escolha.

E como sempre digo, para aprender bem, é preciso praticar bastante. Ao final desta você terá outro exercício prático. Amanhã eu envio a resposta para você conferir, mas é muito importante que você tente resolvê-lo sozinho antes de ver a minha resolução.

→ [Acessar a sexta aula do minicurso GRÁTIS!](#)

Boa aula!

# E-mail 8 - Resposta do exercício da Aula (6/10)

Ontem eu enviei pra você a aula #6 do **minicurso de lógica de programação**. No final da aula eu pedi pra você resolver um exercício de lógica para informar a posição de uma letra no alfabeto usando a estrutura ESCOLHA-CASO.

Espero que você tenha tentado fazer sozinho.

Abaixo a solução deste exercício usando a estrutura ESCOLHA-CASO.

---

```
algoritmo "Posição da letra no alfabeto"
var
    letra : CARACTERE
    posicao : INTEIRO
inicio
    ESCRIVA("Digite uma letra: ")
    LEIA(letra)
    ESCOLHA letra
        CASO "a"
            posicao := 1
        CASO "b"
            posicao := 2
        CASO "c"
            posicao := 3
        CASO "d"
            posicao := 4
        CASO "e"
            posicao := 5
        CASO "f"
            posicao := 6
        CASO "g"
            posicao := 7
        CASO "h"
            posicao := 8
```

```

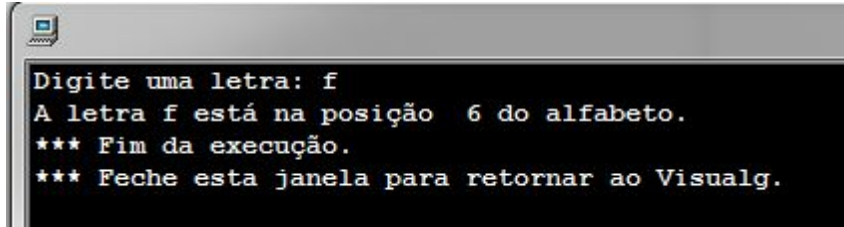
CASO "i"
    posicao := 9
CASO "j"
    posicao := 10
CASO "k"
    posicao := 11
CASO "l"
    posicao := 12
CASO "m"
    posicao := 13
CASO "n"
    posicao := 14
CASO "o"
    posicao := 15
CASO "p"
    posicao := 16
CASO "q"
    posicao := 17
CASO "r"
    posicao := 18
CASO "s"
    posicao := 19
CASO "t"
    posicao := 20
CASO "u"
    posicao := 21
CASO "v"
    posicao := 22
CASO "w"
    posicao := 23
CASO "x"
    posicao := 24
CASO "y"
    posicao := 25
CASO "z"
    posicao := 26

FIMESCOLHA

ESCREVA("A letra ", letra, " está na posição ", posicao, " do alfabeto.")

```

Aqui um resultado da execução deste algoritmo.



É possível implementar um algoritmo com a estrutura SE-ENTÃO-SENÃO, mas ficaria bem maior. Veja um início deste algoritmo.

---

**algoritmo** "Posição da letra no alfabeto com SE"

**var**

letra : CARACTERE

posicao : INTEIRO

**inicio**

ESCREVA("Digite uma letra: ")

LEIA(letra)

**SE** letra = "a" **ENTÃO**

posicao := 1

**SENÃO**

**SE** letra = "b" **ENTÃO**

posicao := 2

**SENÃO**

**SE** letra = "c" **ENTÃO**

posicao := 3

**SENÃO**

**SE** letra = "d" **ENTÃO**

posição := 4

**SENÃO**

**SE** letra = "e" **ENTÃO**

posicao := 5

**SENÃO**

**SE** ....

.....

**FIMSE**

```

                                FIMSE
                        FIMSE
                FIMSE
        FIMSE
FIMSE
        ESCREVA("A letra ", letra, " está na posição ", posicao, " do alfabeto.")
finalgoritmo
```

---

**Surpresa!** Se você acompanhou esse exercício até aqui e está gostando do minicurso, eu vou te ensinar como fazer todo o trabalho dessa estrutura ESCOLHA-CASO neste exercício, com apenas **UMA LINHA de código!**

Mas só vou mandar essa sacada amanhã. Para que você tende descobrir sozinho.

Fazer um algoritmo para mostrar a ordem de uma letra no alfabeto sem usar as estruturas SE-ENTÃO-SENÃO e ESCOLHA-CASO.

Amanhã vou te mostrar como fazer isso. Mas fica o desafio, **tende descobrir antes que eu te envie e resposta.**

(Dica: Tabela ASCII)

Forte abraço e até amanhã!

# E-mail 9 - Sacada! Solução do exercício de ontem sem usar ESCOLHA-CASO ou SE-ENTÃO-SENÃO

Olá nobre aluno(a). Lembra que prometi te enviar um algoritmo que resolva o exercício de ontem sem usar nenhuma estrutura de controle de fluxo.

Ficou curioso? Descobriu como fazer isso?

Aí vai! A malandragem é a seguinte...

Na computação, todos caracteres tem um correspondente numérico para que este caractere possa ser armazenado na forma de bits.

Existe uma tabela chamada **Tabela ASCII** para sabermos qual o número de uma letra. E as letras do alfabeto estão em sequência nesta tabela.

Veja a baixo uma parte da tabela ASCII e identifique o valor numérico do caractere "a".

Decimal	Caractere	Decimal	Caractere
32	espaço	80	P
33	!	81	Q
34	"	82	R
35	#	83	S
36	\$	84	T
37	%	85	U
38	&	86	V
39	'	87	w
40	(	88	X



41	)	89	Y
42	*	90	Z
43	+	91	[
44	,	92	
45	-	93	]
46	.	94	^
47	/	95	_
48	0	96	`
49	1	97	a
50	2	98	b
51	3	99	c
52	4	100	d
53	5	101	e
54	6	102	f
55	7	103	g
56	8	104	h
57	9	105	i
58	:	106	j
59	;	107	k
60	<	108	l
61	=	109	m
62	>	110	n
63	?	111	o
64	@	112	p
65	A	113	q

66	B	114	r
67	C	115	s
68	D	116	t
69	E	117	u
70	F	118	v
71	G	119	w
72	H	120	x
73	I	121	y
74	J	122	z
75	K	123	{
76	L	124	
77	M	125	}
78	N	126	~
79	O	127	DEL

Viu que o valor do caractere "a" é 97 e que as outras letras estão na sequência?  
b = 98, c = 99, d = 100, ...

Agora ficou fácil, só precisamos descobrir o valor da letra que o usuário digitou e subtrair 96.

Para descobrir o valor ASCII de um caractere no Visualg, podemos utilizar a função ASC, passando como parâmetro a letra que o usuário digitou.

**Importante!** Vamos falar mais sobre **funções** e **procedimento** na aula 10. Por hora só saiba que uma função executa uma tarefa pra gente. (Talvez a função ASC use a estrutura ESCOLHA-CASO internamente para retornar o número ASCII da letra.)

A função ASC(caracter) retorna o número da tabela ASCII da letra que passamos como parâmetro.

Logo, o nosso algoritmo ficaria assim.

---

```
algoritmo "Posição da letra no alfabeto"
var
    letra : CARACTERE
    posicao : INTEIRO
inicio
    ESCREVA("Digite uma letra: ")
    LEIA(letra)

    posicao := ASC(letra) - 96
    ESCREVA("A letra ", letra, " está na posição ", posicao, " do alfabeto.")
fimalgoritmo
```

---

O resultado é o mesmo do algoritmo que fizemos ontem.

Gostou dessa sacada?

Na nossa próxima aula vamos aprender a usar as estruturas de repetição.

Fique atento ao seu e-mail que enviarei a próxima aula amanhã!

Até mais ...

# E-mail 10 - Loops Básicos!

→ [Clique AQUI para ler a aula #7](#)

Nesta aula #7 do **minicurso de lógica de programação** vamos aprender uma estrutura MUITO utilizada na programação.

As **estruturas de repetição** são muito utilizadas em desenvolvimento de softwares. Entender como elas funcionam é muito importante para resolver problemas que precisam executar tarefas repetidas vezes. Acredite, existem muitos!

E como sempre digo, para aprender bem, é preciso praticar bastante. Ao final desta aula você também terá mais um exercício prático pra fazer. Amanhã eu envio a resposta para você conferir, mas é muito importante que você tente resolvê-lo sozinho antes de ver a minha resolução.

→ [Acessar a sétima aula do minicurso GRÁTIS!](#)

Boa aula!

# E-mail 11 - Resposta do exercício da Aula (7/10)

Olá! Ontem eu enviei pra você a aula #7 do **minicurso de lógica de programação**. No final da aula eu pedi pra você resolver um exercício criando uma algoritmo capaz de fazer multiplicação de qualquer número positivo.

Espero que você tenha tentado fazer sozinho heim! Se não fez, tente fazer primeiro pra depois olhar a resposta aqui neste e-mail.

Se você se lembrar, eu mostrei esse algoritmo na primeira aula deste minicurso. Lembra?

Abaixo o meu algoritmo de multiplicação entre números positivos.

---

```
algoritmo "Multiplicação"
var
    numero1, numero2, resultado, contador: INTEIRO
inicio
    ESCREVA("Informe o primeiro número: ")
    LEIA(numero1)
    ESCREVA("Informe o segundo número: ")
    LEIA(numero2)
    contador <- 0
    ENQUANTO ( contador < numero2 ) FACA
        resultado <- resultado + numero1
        contador <- contador + 1
    FIMENQUANTO
    ESCREVA("Resultado: ", resultado)
fimalgoritmo
```

---

Aqui um resultado da execução deste algoritmo.

---

```
Início da execução
Informe o primeiro número: 6
```

Informe o segundo número: 8

Resultado: 48

Fim da execução.

---

Gostou do exercício?

Na próxima aula você vai aprender uma outra estrutura de repetição muito usada na programação.

Vou mandar a aula por e-mail, aguarde.

Por hora tente praticar um pouco mais fazendo algoritmos que utilizam loops

Até a próxima aula!

# E-mail 12 - Loops Pré-definido

[→ Clique AQUI para ler a aula #8](#)

Nesta aula **#8** do **minicurso de lógica de programação** vamos aprender a estrutura de repetição mais utilizada na programação.

A estrutura **PARA-FAÇA!**

Na aula de hoje você vai entender como ela funciona e porque ela é a mais usada.

Ah! Claro que tem um super exercício pra você resolver também. Amanhã eu envio a resposta para você conferir, mas é muito importante que você tente resolvê-lo sozinho antes de ver a minha resolução. Combinado?

Clica aí pra ler a aula de hoje.

[→ Acessar a oitava aula do minicurso GRÁTIS!](#)

Boa aula!

# E-mail 13 - Algoritmo números primos:

## Exercício da Aula (8/10)

Opa! Ontem eu enviei pra você a aula **#8 do minicurso de lógica de programação**. No final da aula eu pedi pra você resolver um exercício.

Fazer um algoritmo para dizer se um determinado número é primo ou não.

E aí, conseguiu fazer? Espero que você tenha tentado e conseguido fazer sozinho!

Se não conseguiu, tudo bem. Abaixo você vai ver o algoritmo que eu fiz para este problema.

---

**Algoritmo** "NumeroPrimo"

**Var**

contador : INTEIRO

numero : INTEIRO

eprimo : LOGICO

**Inicio**

ESCREVA("Informe um número para verificar se ele é primo: ")

LEIA(numero)

eprimo := VERDADEIRO

**PARA** contador DE 2 ATÉ numero-1 **FAÇA**

**SE** (numero MOD contador) = 0 **ENTAO**

eprimo := FALSO

**FIMSE**

**FIMPARA**

**SE** eprimo = VERDADEIRO **ENTAO**

ESCREVA("O número ", numero, " é primo!")

**SENAO**

ESCREVA("O número ", numero, " NÃO é primo!")

**FIMSE**

**Fimalgoritmo**

---



Neste algoritmo eu faço um loop de 2 até o número imediatamente anterior ao número que estou verificando se é primo. Por quê? Evidentemente o número primo é divisível (resto 0) por 1 e por ele mesmo. Ele não deve ser divisível por qualquer outro número.

Caso algum número entre 2 e "numero-1" seja capaz de dividir o número verificado com resto zero (SE (numero MOD contador) = 0 ENTÃO ...), significa que ele não é primo (eprimo = FALSO).

Aqui um resultado da execução deste algoritmo.

---

```
Início da execução
Informe um número para verificar se ele é primo: 53
O número 53 é primo!
Fim da execução.
```

---

O que você achou? Gostou da minha resolução. O seu algoritmo pode ter sido diferente. Há várias formas de se fazer algoritmos. Não precisa estar igual o meu. Precisa funcionar corretamente. ;)

Bom por hoje é só.

Vou mandar próxima aula sobre arrays por e-mail, fique atento à tua caixa de entrada.

Até lá!

# E-mail 14 - Vetores e Matrizes

[→ Clique AQUI para ler a aula #9](#)

Nesta aula #9 do **minicurso de lógica de programação** vamos aprender uma estrutura de dados básica.

Existem várias estruturas de dados no mundo da programação, mas a mais conhecida e também a mais utilizada é o **array**

E este é o assunto desta nona aula do nosso minicurso grátis de lógica de programação.

Além de aprender como funcionam os arrays, você verá uma aplicação para ele e também um desafio para você resolver.

Você será testado sobre todo o conteúdo do minicurso. Portanto eu te desafio a resolver este problema. Leia a aula toda para ver que desafio é esse.

[→ Acessar a nona aula do minicurso GRÁTIS!](#)

Boa aula!

# E-mail 15 - Jogo da Velha: Exercício da Aula (9/10)

Olá! Estava ansioso por este e-mail?

Na aula **#9** eu prometi te enviar o meu algoritmo do jogo da velha.

Não é um algoritmo facinho, mas já estamos chegando ao final deste minicurso e sei que você tem capacidade de criar este algoritmo sozinho.

Mas tinha que quebrar a cabeça um pouquinho. Neste exercício, que passei como um **desafio** para você, é necessário utilizar várias estruturas, operadores, variáveis, etc. Tudo que já aprendemos nas aulas anteriores.

Então vamos ao meu algoritmo do jogo da velha, mas antes vamos lembrar as regras:

1 - As jogadas do jogo da velha deverão ser armazenadas numa matriz (3x3) de caractere, chamada "tabuleiro", cada posição desta matriz armazenará um dos valores: " ", "\_", "X" ou "O". Abaixo uma representação gráfica desta matriz.

```
1  2  3
1  __|__|__
2  __|__|__
3   |  |  |
```

2 - A cada jogada o programa deverá mostrar na tela a situação atual do "tabuleiro".

Por exemplo:

```
1  2  3
```

1 \_|\_|\_

2 \_|\_X\_|\_

3 O | | O

3 - Terão dois jogadores no jogo. O programa deve solicitar o nome dos dois jogadores antes de começar o jogo. A cada jogada o programa deverá perguntar separadamente as posições horizontal e vertical da jogada, nesta ordem.

4 - Quando um jogador vencer o jogo, o programa deve apresentar imediatamente o vencedor e a situação do “tabuleiro”.

**Estou te enviando em anexo** o meu algoritmo, para visualizá-lo e executá-lo abra-o no VisuAlg.

Para baixar o VisuAlg Acesse:  
<http://dicasdeprogramacao.com.br/download-visualg/>

Abra o algoritmo, entenda, execute-o, veja porque usei cada estrutura PARA-FAÇA, REPIRA-ATÉ, SE-ENTÃO-SENÃO, cada variável, operadores lógicos (E e OU) etc.

Se tiver dúvida, acesse as aulas anteriores onde falo sobre cada um desses assuntos.

Aguarde a próxima aula de número #10 deste minicurso. Vamos ver como podemos melhorar este algoritmo do jogo da velha utilizando **funções** e **procedimentos**.

Até lá!

Gustavo

# E-mail 16 - Funções e Procedimentos

[→ Clique AQUI para ler a aula #10](#)

Nesta aula #10 do **minicurso de lógica de programação** vamos aprender o que são funções e procedimentos e como usá-los para melhorar os nossos algoritmos.

Além de ver algumas funções e procedimentos que são nativas em qualquer linguagem de programação, vamos aprender a criar as nossas próprias funções e procedimentos.

Você poderá aplicar isso no Jogo da Velha que te mandei ontem.

Amanhã eu vou te mandar o mesmo algoritmo utilizando funções e procedimentos, mas tente você mesmo melhorá-lo.

[→ Acessar a décima aula do minicurso GRÁTIS!](#)

Boa aula!

# E-mail 17 - Jogo da Velha com funções e procedimentos

Na última aula **#10** deste minicurso gratuito de lógica de programação, eu prometi enviar um algoritmo do Jogo da Velha melhorado, usando Funções e Procedimentos.

**Estou te enviando o novo algoritmo em anexo**, para visualizá-lo e executá-lo abra-o no VisuAlg.

Para baixar o Visualg Acesse: <http://dicasdeprogramacao.com.br/download-visualg/>

Abra o algoritmo, entenda, execute-o.

O primeiro "procedimento" que podemos ver é o `imprimeTabuleiro`, este procedimento é responsável por mostrar na tela a situação atual do Jogo da Velha. Repare que antes fazíamos isso duas vezes no algoritmo, uma em cada jogada e uma no final. Agora não tem mais essa duplicação no código, a gente só chama o procedimento "imprimeTabuleiro".

---

```
procedimento imprimeTabuleiro
inicio
    //Apresenta a situação atual do tabuleiro
    escreval("Neste momento o tabuleiro está assim:")
    escreval("  1   2   3")
    escreval("1 _", tabuleiro[1,1], "_|_", tabuleiro[1,2], "_|_", tabuleiro[1,3],
    "_")
    escreval("2 _", tabuleiro[2,1], "_|_", tabuleiro[2,2], "_|_", tabuleiro[2,3],
    "_")
    escreval("3  ", tabuleiro[3,1], " | ", tabuleiro[3,2], " | ", tabuleiro[3,3], "
    ")
fimprocedimento
```

---

Além disso eu tirei uma complexidade do código criando uma função chamada "verificaVencedor" que é responsável por identificar se alguém ganhou o jogo, dessa forma o fluxo principal do algoritmo fica mais legível.

Este foi o final da aula #10 do minicurso de lógica de programação. Mas não acabou, você deu o primeiro passo que é aprender lógica de programação e está pronto para evoluir mais. **Lógica de programação deve ser praticada para, de fato, aprender.**

Vou continuar enviando conteúdo para você, fique atento à sua caixa de entrada.

Se você gostou deste minicurso, [indique-o](#) para os seus amigos, é gratuito. Acredito que todos devem aprender programação.

Até mais.

Gustavo