

NExT

Nova Experiência de Trabalho

Tatyane Calixto
tscs@cesar.org.br

Erick Simões
esm@cesar.org.br



C.E.S.A.R



Treinamento Preparatório

DESAFIO

Propósito

O objetivo do desafio é avaliar a compreensão e domínio dos conceitos fundamentais de lógica de programação, resolução de algoritmos e boas práticas de programação. A aprovação no desafio é critério para a inscrição no NExT.

Datas e horários

O desafio será lançado dia 11/08 (quinta-feira) e pode ser submetido até o dia 13/08 (sábado), às 23:59.

Submissão do desafio

O desafio será postado no Classroom como um formulário, com questões de múltipla escolha e de envio de arquivo. Antes de enviar, certifique-se que as informações pessoais foram corretamente preenchidas.

Treinamento Preparatório

DESAFIO

? Qualquer dúvida sobre o desafio, entre em contato com *Maurício* (mto@cesar.org.br) e *Erick* (erick.simoes@cesar.school)

Critérios

São critérios de avaliação das questões práticas:

- **Corretude:** o código é executado corretamente, sem erros e com a saída apropriada;
- **Implementação:** o código possui o uso adequado de recursos;
- **Indentação:** o código está indentado corretamente, segundo as boas práticas de programação.

Resultado

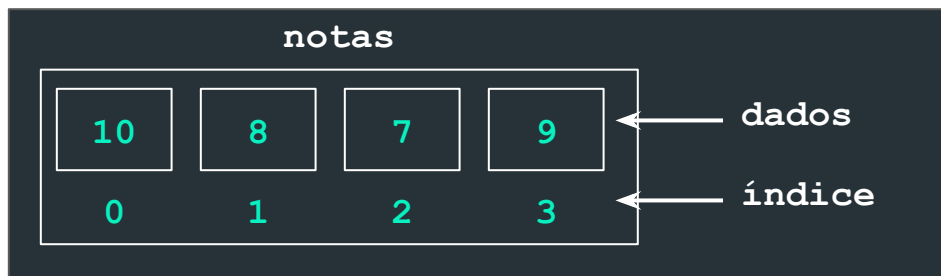
Na comunicação, que será divulgada no dia 22/08 por email, terá apenas o resultado "aprovado" ou "não aprovado", sem a nota.



REVISÃO

vetor

O VETOR É UMA...



...ESTRUTURA DE DADOS

DECLARAÇÃO

colchetes para o tamanho

chaves para os elementos

```
inteiro notas[4]
caracter vogais[] = {'a', 'e', 'i', 'o', 'u'}
real alturas[3] = {1.8, 1.5, 2.0}
```

ACESSAR E MODIFICAR UM VALOR

```
caracter vogais[] = {'a', 'e', 'i', 'o', 'u'}

escreva(vogais[1]) // e
vogais[1] = 'E'
escreva(vogais[1]) // E
```

“VARRER” O VETOR

```
real alturas[] = {1.8, 1.5, 2.0}  
  
para (inteiro i = 0; i < 3; i++) {  
    escreva (alturas[i], "\n")  
}
```



FUNÇÃO

sub-rotina

Função

O uso de estruturas de repetição garante, até certo ponto, a redução de **código duplicado**. Contudo, ainda é comum que tenhamos que repetir uma série de comandos:

```
inteiro idades[] = {18, 15, 20, 40, 21}
```

```
para (inteiro i = 0; i < 5; i++) {  
    escreva(idades[i], " - ")  
}
```

```
escreva("\n")
```

```
idades[2] = 19 // o valor 20 é substituído por 19
```

```
para (inteiro i = 0; i < 5; i++) {  
    escreva(idades[i], " - ")  
}
```

Função

INTRODUÇÃO

Função

Para esses casos, podemos usar **funções**.

De fato, já usamos uma série de funções implementadas pela linguagem:

```
escreva("Olá, tudo bem?")  
leia(numero)  
Texto.numero_caracteres(nome)
```

Função

INTRODUÇÃO

Função

Na verdade, todo o código que escrevemos está inserido dentro de uma **função**:

```
funcao inicio () {  
    cadeia nome  
    escreva("Digite seu nome: ")  
    leia(nome)  
}
```

O interessante é que podemos criar nossas próprias funções!

Função

INTRODUÇÃO

Função

Uma função é uma **sub-rotina**, ou seja, um bloco de código dentro do programa que pode ser “acionado” sempre que precisamos dele para um objetivo específico.

Função

INTRODUÇÃO

```
funcao exibir_nome() {  
    escreva ("Predo\n")  
}
```

nome da função

parâmetros
(se houver)

corpo

Função

DETALHES

Função

Detalhes importantes:

- Cada função deve se ater a realizar **uma atividade específica**;
- Suas funções devem ser escritas fora da **função início**, mas dentro do **programa**.

Vantagens:

- Reduz a quantidade de linhas de código;
- Facilita o entendimento do programa (modularização);
- Facilita a manutenção.

Função

Para que uma função seja executada, ela precisa ser chamada.

Uma função implementada assim:

```
funcao exibir_nome () {  
    escreva ("Predo\n")  
}
```

deve ser chamada assim:

```
exibir_nome ()
```


Função

CHAMADA

Função - Parâmetros

Quando uma função precisa de alguma informação para realizar sua atividade, essas informações podem ser passadas por parâmetro.

```
funcao exibir_nome (cadeia nome) {  
    escreva("Oi, meu nome é ", nome, "\n")  
}
```



Agora essa função deve ser chamada assim:

```
exibir_nome("Jorge")
```

Função

PARÂMETROS

Função - Parâmetros

Uma função pode ter uma quantidade qualquer de parâmetros de quaisquer tipos.

```
funcao idade (inteiro nasc, inteiro atual, cadeia nome) {  
    inteiro idade_usuario = atual - nasc  
    escreva(nome, " tem ", idade_usuario, " anos\n")  
}
```

Para executar essa função:

```
idade(2003, 2021, "Mar")
```

Função

PARÂMETROS



EXEMPLOS

função

EXEMPLO 1:

Crie uma função que exibe os valores de um vetor de inteiros.
Use para mostrar o antes e depois de uma manipulação de seus dados.

Função

EXEMPLOS

EXEMPLO 1:

@CESAR 2022 | Todos os Direitos Reservados



Função

EXEMPLOS

```
funcao inicio() {  
    inteiro numeros[] = {10, 20, 30, 40, 50}  
  
    exhibe_vetor(numeros, 5)  
  
    numeros[3] = 80  
  
    exhibe_vetor(numeros, 5)  
}  
  
funcao exhibe_vetor(inteiro num[], inteiro t) {  
    para(inteiro i = 0; i < t; i++) {  
        escreva(num[i], " ")  
    }  
    escreva("\n")  
}
```



FUNÇÃO

o retorno

Função - Retorno

@CESAR 2022 | Todos os Direitos Reservados



Além de manipular e exibir os dados recebidos como parâmetro, uma função também pode retornar informação.

Função

RETORNO

```
funcao inteiro soma(inteiro num1, inteiro num2) {  
    retorne num1 + num2  
}
```

*tipo que vai ser
retornado*

*comando que encerra a
função e retorna o valor*

```
valor = soma(10, 5)
```

Função - Retorno

É importante saber que:

- Quando o comando **retorne** é alcançado, a execução do código volta para o ponto da chamada da função;
- Qualquer comando escrito depois do **retorne** não será executado.

Função

RETORNO



EXEMPLOS

função

EXEMPLO 2:

O programa deve calcular a média de um aluno, pedindo a entrada de duas notas N1 e N2. A N1 tem peso 2 e a N2 tem peso 3.

```
funcao inicio() {  
  
    escreva(media_ponderada(10, 9))  
}  
  
funcao real media_ponderada(real n1, real n2) {  
    retorne (n1*2 + n1*3) / (2 + 3)  
}
```

Função

EXEMPLOS

Breakout Time!

Resolva os desafios da lista de exercícios com sua sala no breakout room.

A lista possui exercícios em duas categorias:

- Exercícios fundamentais;
- Exercícios de aprofundamento.

Se precisar de ajuda, chame uma das pessoas monitoras ou professoras.



Lista de Exercícios 09



C . E . S . A . R

Pessoas impulsionando inovação.
Inovação impulsionando negócios.

Tatyane Calixto
tscs@cesar.org.br

Erick Simões
esm@cesar.org.br

e a melhor equipe de monitores
da CESAR School 

