

# CONSTRUÇÃO DE ALGORITMOS

---

*Bacharelado em Sistemas da Informação*

*Prof. Marco André Abud Kappel*

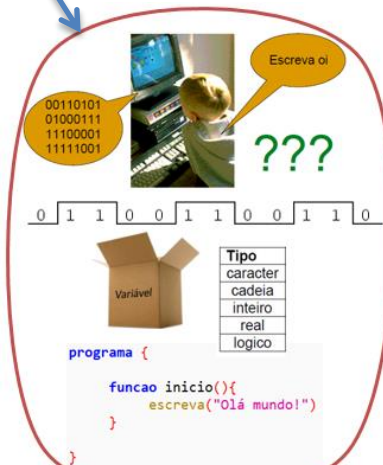
**Aula 5 – Funções**

# Funções

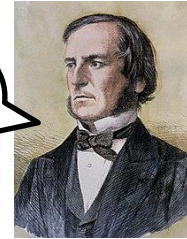
- Anteriormente:



Hoje:  
Funções



falso  
verdadeiro



nao

e

ou

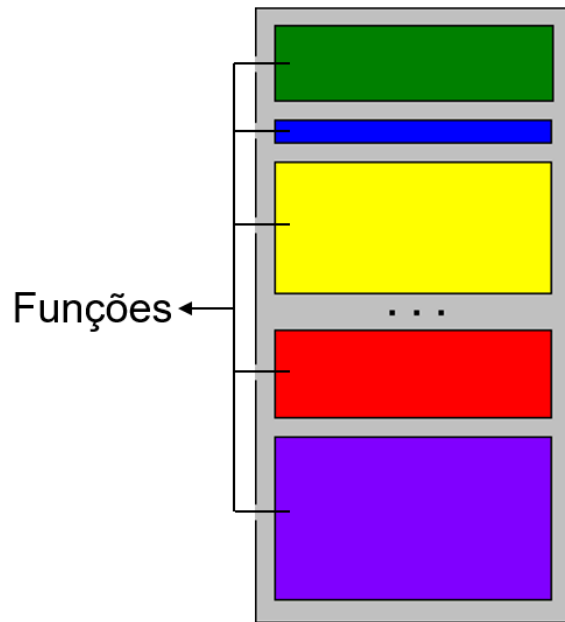
Operador	Operação
==	igual
>	maior
<	menor
>=	maior ou igual
<=	menor ou igual

x	y	x ou y	x e y
falso	falso	falso	falso
falso	verdadeiro	verdadeiro	falso
verdadeiro	falso	verdadeiro	falso
verdadeiro	verdadeiro	verdadeiro	verdadeiro

# Funções

- **Definição**

- Como vimos anteriormente, um **programa** de computador organiza um **conjunto de códigos** divididos em **vários módulos**.
- Cada um destes **módulos** é representado por uma **função**.

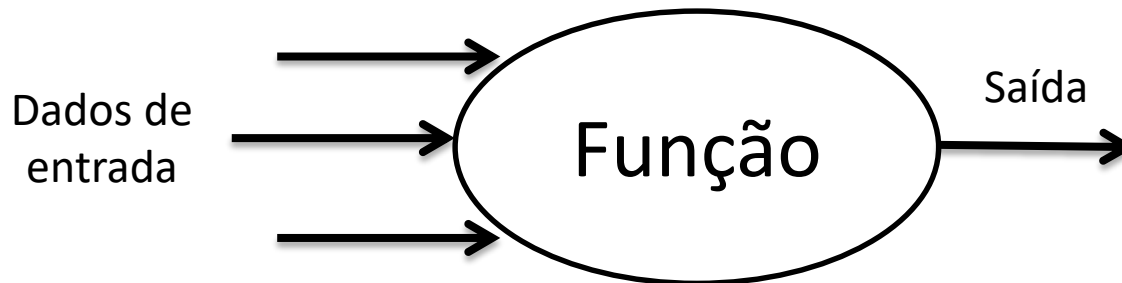


- Cada função é responsável por executar uma **ação própria**, realizando algum processamento útil para o programa como um todo.

# Funções

- **Definição**

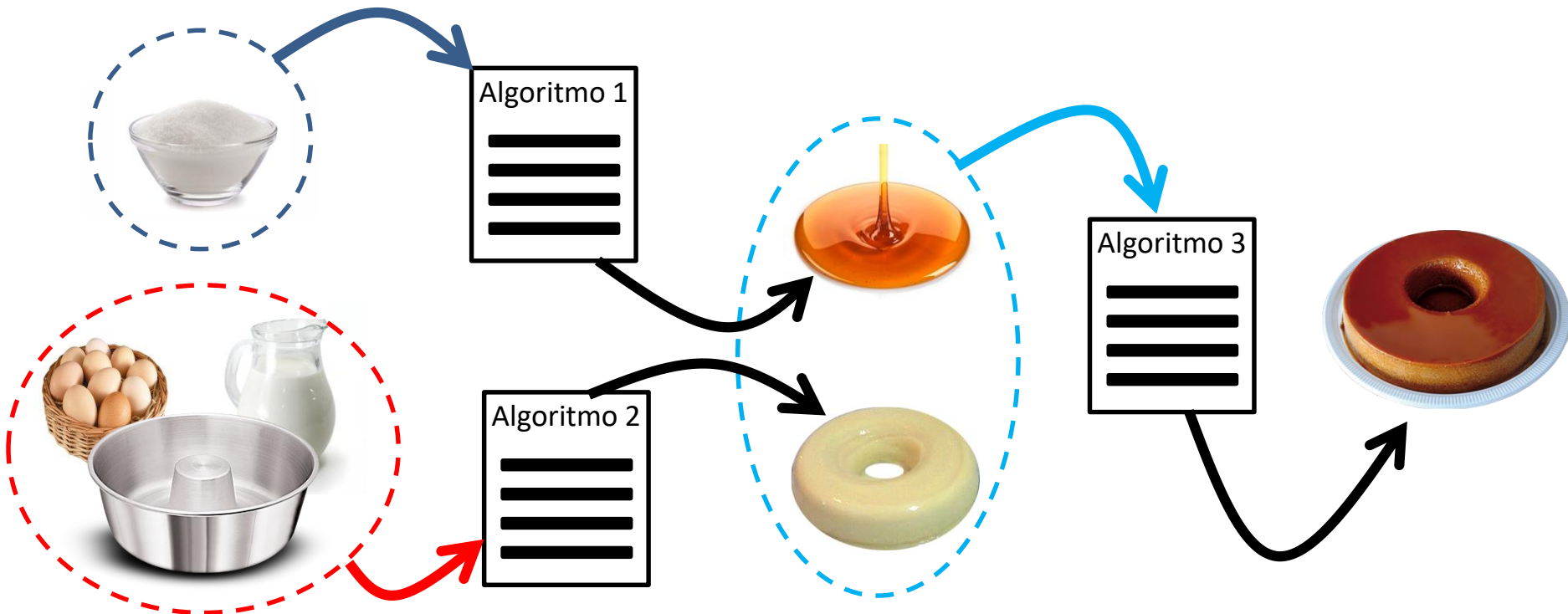
- A função é um dos elementos mais fundamentais na **estrutura** de um programa.
- Facilita a realização da essência da programação: **resolver problemas usando algoritmos**.
- Geralmente, um algoritmo precisa de **dados de entrada** e de um **procedimento** que irá transformá-los, gerando um **dado de saída**.



# Funções

- **Definição**

- Utilizando as funções, é possível **dividir** um problema em vários problemas menores.



# Funções

---

- **Definição**
  - As funções também são chamadas de “**subprogramas**”, pois representam exatamente isso: um pequeno programa, que será **utilizado** por um maior.
  - Já usamos algumas **funções prontas** até aqui:

# Funções

- **Definição**

- As funções também são chamadas de “**subprogramas**”, pois representam exatamente isso: um pequeno programa, que será **utilizado** por um maior.
- Já usamos algumas **funções prontas** até aqui:

```
escreva("Meu nome é ", "Marco")
```

```
leia(x)
```

```
mat.raiz(9,2)
```

- Perceba que não precisamos saber **como** é calculada a raiz, apenas **confiamos** que a função o faz corretamente.

# Funções

---

- Criação de funções
  - Uma função é constituída por:

```
funcao tipo_de_retorno identificador(tipo1 par1, tipo2 par2, ...){  
    bloco de comandos  
    retorne algum_resultado  
}
```




# Funções

- Criação de funções

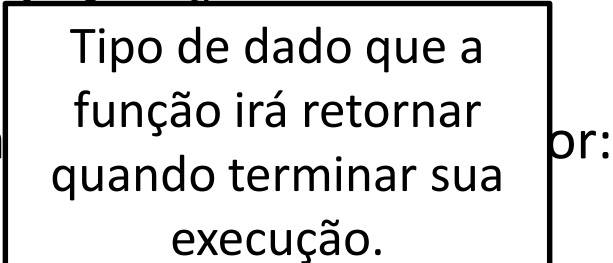
Palavra reservada que indica que será criada uma função.

substituída por:



```
funcao tipo_de_retorno identificador(tipo1 par1, tipo2 par2, ...){  
    bloco de comandos  
    retorne algum_resultado  
}
```

# Funções

- Criação
  - Uma  or:


```
funcao tipo_de_retorno identificador(tipo1 par1, tipo2 par2, ...){  
    bloco de comandos  
    retorne algum_resultado  
}
```

# Funções

- Criação de função

- Uma função

Nome da função, escolhido pelo programador. O nome precisa seguir as regras de um identificador válido.



```
funcao tipo_de_retorno identificador(tipo1 par1, tipo2 par2, ...){  
    bloco de comandos  
    retorne algum_resultado  
}
```

# Funções

- Criação de funções

- Uma função é constituída por:

Cada parâmetro receberá um valor passado na chamada da função, na ordem que aparecerem. Depois, eles funcionarão como variáveis que só existem dentro da função (no seu escopo).

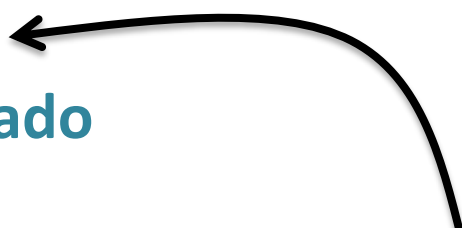
```
funcao tipo_de_retorno identificador(tipo1 par1, tipo2 par2, ...){  
    bloco de comandos  
    retorne algum_resultado  
}
```

Lista de parâmetros de entrada.  
Cada parâmetro precisa ter um tipo e um nome, definidos pelo programador.

# Funções

- Criação de funções
  - Uma função é constituída por:

```
funcao tipo_de_retorno identificador(tipo1 par1, tipo2 par2, ...){  
    bloco de comandos  
    retorne algum_resultado  
}
```




O bloco de comandos pode conter qualquer algoritmo, expressões, criação de variáveis, etc...

# Funções

---

- Criação de funções
  - Uma função é constituída por:

```
funcao tipo_de_retorno identificador(tipo1 par1, tipo2 par2, ...){  
    bloco de comandos  
    retorne algun_resultado  
}
```



Palavra reservada que indica que o valor seguinte será retornado ao escopo da chamada da função.

# Funções

- Criação de funções
  - Uma função é constituída por:

```
funcao tipo_de_retorno identificador(tipo1 par1, tipo2 par2, ...){  
    bloco de comandos  
    retorne algun_resultado  
}
```



Alguma variável ou expressão que resulte em um valor do mesmo tipo indicado no **tipo\_de\_retorno**


# Funções

- Criação de funções

- Também podemos construir funções que apenas executam algum processamento, mas **não retornam nada**.
- Nesse caso, sua declaração seria:

```
funcao tipo_de_retorno identificador(tipo1 par1, tipo2 par2, ...){  
    bloco de comandos  
    retorne algum_resultado  
}
```

Estas funções também podem ser chamadas de “procedimentos”.




```
funcao identificador(tipo1 par1, tipo2 par2, ...){  
    bloco de comandos  
}
```



# Funções

- Criação de funções
  - No seu programa:

```
programa{  
  
    funcao inicio(){  
  
    }  
  
}
```



As funções criadas pelo programador devem ser colocadas aqui, no escopo do programa, mas fora do escopo de outras funções.

# Funções

- Criação de funções

- No seu programa:

```
programa{  
  
    funcao inicio(){  
  
    }  
  
}
```



Perceba que o módulo “inicio” nada mais é que uma função especial, sem retorno e sem argumentos de entrada.

As funções criadas pelo programador devem ser colocadas aqui, no escopo do programa, mas fora do escopo de outras funções.

# Funções

- Criação de funções
  - No seu programa:

```
programa{  
  
    funcao inicio(){  
  
    }  
  
    funcao real calculaMedia(real X, real Y){  
        real resultado  
        resultado = (X + Y)/2.0  
        retorne resultado  
    }  
  
}
```


# Funções

- Criação de funções

- No seu programa:

```
programa{  
  
    funcao inicio(){  
  
    }  
  
    funcao real calculaMedia(real X, real Y){  
        real resultado  
        resultado = (X + Y)/2.0  
        retorne resultado  
    }  
  
}
```

Função que recebe dois números reais, calcula a média aritmética deles, coloca o resultado na variável local “resultado” e retorna o valor que está na variável para quem invocou a função.



# Funções

- Criação de funções

- No seu programa:

```
programa{  
  
    funcao inicio(){  
  
    }  
  
    funcao real calculaMedia(real X, real Y){  
        real resultado  
        resultado = (X + Y)/2.0  
        retorne resultado  
    }  
  
}
```

Função que recebe dois números reais, calcula a média aritmética deles, coloca o resultado na variável local “resultado” e retorna o valor que está na variável para quem invocou a função.



Importante! Perceba que a variável “resultado” existe apenas no escopo dessa função.

- Criação de funções
  - Para chamar a função criada:

```
programa{  
  
    funcao inicio(){  
        real media  
        real a = 5.0  
        real b = 10.0  
  
        media = calculaMedia(a,b)  
  
        escreva(media)  
    }  
  
    funcao real calculaMedia(real X, real Y){  
        real resultado  
        resultado = (X + Y)/2.0  
        retorne resultado  
    }  
  
}
```

- Criação de funções

- Para chamar a função criada:

O escopo de cada função é definido pelas chaves que envolvem o bloco de comandos.

```
programa{
```

```
funcao inicio(){
```

```
    real media
```

```
    real a = 5.0
```

```
    real b = 10.0
```

```
    media = calculaMedia(a,b)
```

```
    escreva(media)
```

```
funcao real calculaMedia(real X, real Y){
```

```
    real resultado
```

```
    resultado = (X + Y)/2.0
```


```
    retorne resultado
```

```
}
```

- Criação de funções
  - Para chamar a função criada:

```
programa{  
  
    funcao inicio(){  
        real media  
        real a = 5.0  
        real b = 10.0  
  
        media = calculaMedia(a,b)  
  
        escreva(media)  
    }  
  
    funcao real calculaMedia(real X, real Y){  
        real resultado  
        resultado = (X + Y)/2.0  
        retorne resultado  
    }  
  
}
```

Declaração de três variáveis no escopo da função “inicio”. Estas variáveis só existem neste escopo.



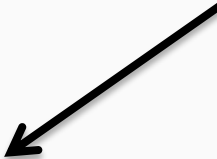


- Criação de funções

- Para chamar a função criada:

```
programa{  
  
    funcao inicio(){  
        real media  
        real a = 5.0  
        real b = 10.0  
  
        media = calculaMedia(a,b)  
  
        escreva(media)  
    }  
  
    funcao real calculaMedia(real X, real Y){  
        real resultado  
        resultado = (X + Y)/2.0  
        retorne resultado  
    }  
  
}
```

Chamada da função  
“calculaMedia”.



- Criação de funções

- Para chamar a função criada:

O resultado da função será atribuído à variável “media”.

Chamada da função “calculaMedia”.

```
inicio(){  
    media  
    a = 5.0  
    b = 10.0  
  
    media = calculaMedia(a,b)  
  
    escreva(media)  
}  
  
funcao real calculaMedia(real X, real Y){  
    real resultado  
    resultado = (X + Y)/2.0  
    retorne resultado  
}  
}
```

- Criação de funções

- Para chamar a função criada:

O resultado da função será atribuído à variável “media”.

Chamada da função “calculaMedia”.

O valores que estão nas variáveis “a” e “b” serão copiados para os parâmetros “X” e “Y”, nesta ordem.

```
inicio(){  
    media  
    a = 5.0  
    b = 10.0  
    real  
    media = calculaMedia(a,b)  
    escreva(media)  
}  
  
funcao real calculaMedia(real X, real Y){  
    real resultado  
    resultado = (X + Y)/2.0  
    retorne resultado  
}  
}
```

- Criação de funções

- Para chamar a função criada:

O resultado da função será atribuído à variável “media”.

Chamada da função “calculaMedia”.

O valores que estão nas variáveis “a” e “b” serão copiados para os parâmetros “X” e “Y”, nesta ordem.

```
inicio(){  
    media  
    a = 5.0  
    b = 10.0
```

```
media = calculaMedia(a,b)
```

```
escreva(media)
```


```
}  
  
funcao real calculaMedia(real X, real Y){  
    real resultado  
    resultado = (X + Y)/2.0  
    retorne resultado  
}
```



**Atenção!** Os tipos de dados enviados como entrada devem ser compatíveis com os declarados aqui, nesta ordem!

- Criação de funções
  - Para chamar a função criada:

```
programa{  
  
    funcao inicio(){  
        real media  
        real a = 5.0  
        real b = 10.0  
  
        media = calculaMedia(a,b)  
  
        escreva(media)  
    }  
  
    funcao real calculaMedia(real X, real Y){  
        real resultado  
        resultado = (X + Y)/2.0  
        retorne resultado  
    }  
}
```



Aqui dentro, podemos usar os parâmetros “X” e “Y” livremente, até chegar no resultado.

- Criação de funções

- Para chamar a função criada:

```
programa{  
  
    funcao inicio(){  
        real media  
        real a = 5.0  
        real b = 10.0  
  
        media = calculaMedia(a,b)  
  
        escreva(media)  
    }  
  
    funcao real calculaMedia(real X, real Y){  
        real resultado  
        resultado = (X + Y)/2.0  
        retorne resultado  
    }  
}
```

The diagram illustrates the execution flow of the code. A purple arrow originates from the function call `calculaMedia(a,b)` in the `inicio` function and points to the `retorne resultado` statement in the `calculaMedia` function. Another purple arrow starts from the `retorne resultado` statement and points back to the assignment `media =` in the `inicio` function, indicating that the return value is stored in the `media` variable. A black arrow points from a text box to the `retorne resultado` statement.

Apenas o valor que está na variável resultado irá retornar. Seu valor será copiado para exatamente o mesmo local em que a função foi chamada.

- **Exemplo:**
  - Implemente o seguinte programa e veja o seu funcionamento:

```
programa{  
  
    funcao inicio(){  
        real media  
        real a = 5.0  
        real b = 10.0  
  
        media = calculaMedia(a,b)  
  
        escreva(media)  
    }  
  
    funcao real calculaMedia(real X, real Y){  
        real resultado  
        resultado = (X + Y)/2.0  
        retorne resultado  
    }  
}
```

# Funções

- **Exemplos**

- Escreva uma função para retornar o dobro de um número fornecido como parâmetro. Teste sua função em um módulo “inicio”.
- Escreva uma função para retornar a área de um triângulo, sendo a sua base e a sua altura fornecidos como parâmetros da função. Teste sua função em um módulo “inicio”.

$$\textit{Área} = \frac{\textit{base} \times \textit{altura}}{2}$$

- Escreva uma função que recebe um número inteiro e retorna um valor lógico indicando se o número é par. Teste sua função em um módulo “inicio”.



# Funções

- **Passagem de parâmetros**

- Até aqui, vimos que a **passagem de parâmetros** para as funções é feita de uma maneira bem simples: os valores são **copiados** do lugar original para uma nova variável, dentro do escopo da função.

```
media = calculaMedia(a,b)
```

```
funcao real calculaMedia(real X, real Y){  
    real resultado  
    resultado = (X + Y)/2.0  
    retorne resultado  
}
```

Perceba que “X” e “a” não precisam nem ter o mesmo nome, são variáveis completamente diferentes.

# Funções

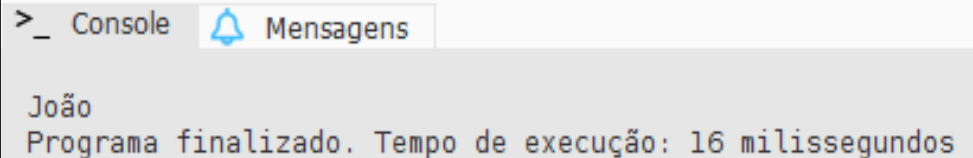
- Passagem de parâmetros
  - O que será impresso na tela?

```
programa{  
  
    funcao inicio(){  
        cadeia meuNome = "João"  
        minhaFuncao(meuNome)  
        escreva(meuNome)  
    }  
  
    funcao minhaFuncao(cadeia meuNome){  
        meuNome = "José"  
    }  
}
```

# Funções

- Passagem de parâmetros
  - O que será impresso na tela?

```
programa{  
  
    funcao inicio(){  
        cadeia meuNome = "João"  
        minhaFuncao(meuNome)  
        escreva(meuNome)  
    }  
  
    funcao minhaFuncao(cadeia meuNome){  
        meuNome = "José"  
    }  
}
```



The screenshot shows a console window with two tabs: 'Console' and 'Mensagens'. The 'Console' tab is active, displaying the output of the program. The first line is 'João', which is the result of the 'escreva(meuNome)' statement in the 'inicio' function. The second line is 'Programa finalizado. Tempo de execução: 16 milissegundos', which is the final status message.

```
>_ Console Mensagens  
  
João  
Programa finalizado. Tempo de execução: 16 milissegundos
```

# Funções

- **Passagem de parâmetros**
  - O que será impresso na tela?



Apesar de terem o **mesmo identificador**, a variável “meuNome” e o parâmetro “meuNome” estão em **escopos diferentes**, ou seja, ocupam diferentes endereços de memória.

```
programa{
```

```
funcao inicio(){  
  cadeia meuNome = "João"  
  minhaFuncao(meuNome)  
  escreva(meuNome)  
}
```

```
funcao minhaFuncao(cadeia meuNome){  
  meuNome = "José"  
}
```

Endereço	Valor
00012	“João”
...	...
00314	“José”

> \_ Console

Mensagens

João

Programa finalizado. Tempo de execução: 16 milissegundos

# Funções

- **Passagem de parâmetros**
  - O que será impresso na tela?

Essa atribuição altera apenas a variável que existe no escopo interno da função. A outra continua inalterada.

```
programa{  
  
    funcao inicio(){  
        cadeia meuNome = "João"  
        minhaFuncao(meuNome)  
        escreva(meuNome)  
    }  
  
    funcao minhaFuncao(cadeia meuNome){  
        meuNome = "José"  
    }  
}
```

Endereço	Valor
00012	"João"
...	...
00314	"José"

> \_ Console Mensagens

João  
Programa finalizado. Tempo de execução: 16 milissegundos

- **Passagem de parâmetros**

- Pense agora no seguinte programa. O que será escrito na tela se o usuário inserir 2 e 5?

```
programa{  
  
    funcao inicio(){  
        real x, y  
        escreva("Escreva o valor de x:" )  
        leia(x)  
        escreva("Escreva o valor de y: ")  
        leia(y)  
        troca(x, y)  
        escreva("O valor final de x é ", x, "\n")  
        escreva("O valor final de y é ", y, "\n")  
    }  
  
    funcao troca(real a, real b){  
        real aux  
        aux = a  
        a = b  
        b = aux  
    }  
  
}
```

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x: ")
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

}

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
program{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x: ")
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}

funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]



- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x: ")
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

Escreva o valor de x:

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:")
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

Escreva o valor de x:2

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x: " )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}

funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y:
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}

funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
program{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x: ")
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

<b>Identificador</b>	<b>Endereço</b>	<b>Valor</b>
x	000121	2
y	002122	5
a	001345	2
b	005120	5
aux	001001	

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x: ")
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]



- **Passagem de parâmetros**

```
program{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
O valor final de x é 2.0
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real a, real b){
    real aux
    aux = a
    a = b
    b = aux
}
```

}

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
O valor final de x é 2.0
O valor final de y é 5.0
```

Programa finalizado. Tempo de execução: 461852 milissegundos

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

# Funções

- **Passagem de parâmetros**

- A função “troca” não mexeu com nenhum valor das **variáveis originais**.
- Isso aconteceu porque passamos os argumentos da forma mais simples, chamada de **passagem por valor**.
- Se quisermos alterar um valor de uma variável que está **fora do escopo** da função, precisamos fazer uma passagem de argumentos **por referência**.

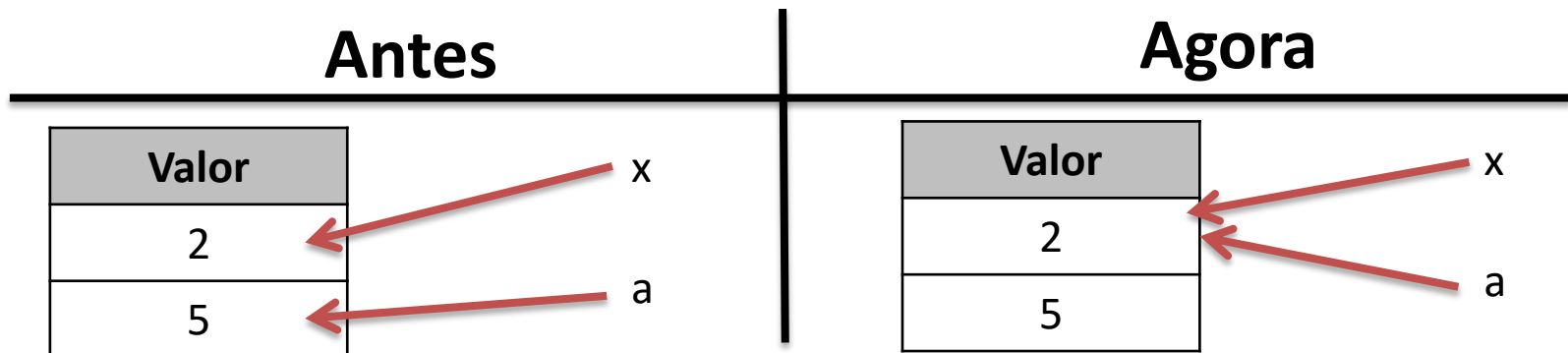
- Para isso, basta fazer a seguinte **alteração** na declaração da função:

```
funcao troca(real &a, real &b){  
    real aux  
    aux = a  
    a = b  
    b = aux  
}
```

# Funções

- **Passagem de parâmetros**

- Quando colocamos o **&** antes do identificador do parâmetro, estamos indicando que ele não receberá uma **cópia** do valor passado, mas uma **referência** à posição de memória da variável passada.
- Agora, quando fizermos alguma alteração em “a”, estamos, na verdade, alterando o conteúdo que está na **posição de memória** a que “a” se refere.



- **Passagem por referência**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x: ")
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}

funcao troca(real &a, real &b){
    real aux
    aux = a
    a = b
    b = aux
}
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]



- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x: ")
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real &a, real &b){
    real aux
    aux = a
    a = b
    b = aux
}
```

}

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
program{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x: ")
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real &a, real &b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

Escreva o valor de x:

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:")
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real &a, real &b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

Escreva o valor de x:2

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}

funcao troca(real &a, real &b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y:
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}

funcao troca(real &a, real &b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real &a, real &b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x: ")
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real &a, real &b){  
    real aux  
    aux = a  
    a = b  
    b = aux  
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
program{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real &a, real &b){  
    real aux  
    aux = a  
    a = b  
    b = aux  
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]



- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x: ")
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real &a, real &b){  
    real aux  
    aux = a  
    a = b  
    b = aux  
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x: ")
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real &a, real &b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real &a, real &b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real &a, real &b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x:" )
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real &a, real &b){
    real aux
    aux = a
    a = b
    b = aux
}
```

> Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
O valor final de x é 2.0
```

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

- **Passagem de parâmetros**

```
programa{
```

```
funcao inicio(){
    real x, y
    escreva("Escreva o valor de x: ")
    leia(x)
    escreva("Escreva o valor de y: ")
    leia(y)
    troca(x, y)
    escreva("O valor final de x é ", x,
    escreva("O valor final de y é ", y,
}
```

```
funcao troca(real &a, real &b){
    real aux
    aux = a
    a = b
    b = aux
}
```

>\_ Console  Mensagens

```
Escreva o valor de x:2
Escreva o valor de y: 5
O valor final de x é 5.0
O valor final de y é 2.0
```

Programa finalizado. Tempo de execução: 5363 milissegundos

Para entender, vamos ver o que ocorre na memória durante a execução:

[illegible]

# Funções

- **Exercícios**

1. Escreva uma função para calcular o comprimento  $C$  de uma circunferência, sendo o valor de seu raio  $R$  fornecido como parâmetro da função. Teste sua função em um módulo “início”.

$$C = 2\pi R$$

2. Escreva uma função que recebe como parâmetro uma temperatura  $F$  dada na escala Fahrenheit e retorna a temperatura equivalente em Celsius. Teste sua função em um módulo “início”.

$$C = (F - 32) \times \frac{5}{9}$$

# Funções

- **Exercícios**

3. Faça uma função que recebe como parâmetros o valor da hora (número inteiro) e o valor dos minutos (número inteiro) de um horário. Esta função deverá calcular e retornar o horário convertido em minutos.

4. Faça um programa que leia três números inteiros que serão fornecidos pelo usuário, calcule e exiba a soma e a média dos números. Este programa deverá utilizar as seguintes funções desenvolvidas por você:

a) `calculaSoma`: recebe como parâmetros os 3 números inteiros, calcula e retorna a soma.

b) `calculaMedia`: recebe como parâmetros os 3 números inteiros, calcula e exibe a média usando a função feita em (a).



## • Exercícios

5. Escreva uma função que recebe um número inteiro e retorna verdadeiro se ele corresponde a um mês.

6. Na matemática, quando estudamos equações do segundo grau da forma  $Ax^2 + Bx + C = 0$ , onde  $x$  é a variável a ter seu valor conhecido e  $A$ ,  $B$  e  $C$  são os coeficientes da equação, aprendemos que precisamos inicialmente obter o valor de delta ( $\Delta = B^2 - 4AC$ ). Crie uma função que recebe os coeficientes e retorna o valor de  $\Delta$ . Em seguida, teste sua função em uma função início.

7. Faça um **procedimento** que calcula a **soma** e o **produto** de 2 números inteiros. O procedimento recebe como parâmetro dois números inteiros e coloca os valores da soma e do produto nas variáveis cujos endereços são fornecidos na chamada.

8. Escreva uma função que receba como parâmetros a quantidade de minutos falados no telefone e o valor do minuto. A função deverá retornar o valor a ser pago pelo cliente.

## • Exercícios

9. Uma conta de caderneta de poupança foi aberta por um cliente com um depósito de R\$ 500,00. O cliente gostaria de saber o valor que ele terá na conta daqui a 12 meses, considerando que a conta é remunerada em 0,6% de juros fixos ao mês, sobre o valor inicial. Crie uma função que receba o valor atual depositado na poupança e o número de meses o qual se deseja calcular a previsão. A função deve retornar o valor na poupança após o número de meses indicado. Use esta função para tirar a dúvida do cliente.

10. Crie uma função que recebe um número e retorna o seu último dígito.

11. Um número é INTERESSANTE se ele for menor do que 10 vezes o seu último dígito. Crie uma função que recebe um inteiro e retorna verdadeiro se o número é INTERESSANTE.

12. Um número é CHATO se ele não for interessante e corresponder a um possível mês. Crie uma função que recebe um inteiro e retorna verdadeiro se o número é CHATO .

## • Exercícios

13. Sabe-se que o volume de uma caixa de lados  $a$ ,  $b$  e  $c$  é dado por:

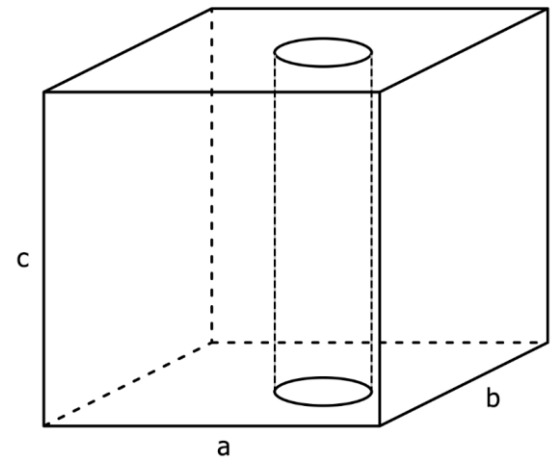
$$V_{caixa} = a \times b \times c$$

e que o volume de um cilindro de raio  $r$  e altura  $h$  é dado por:

$$V_{cilindro} = \pi \times h \times r^2$$

Pede-se:

- Escreva uma função para calcular e retornar o volume de uma caixa de lados  $a$ ,  $b$  e  $c$ . Esta função deverá receber como parâmetro os lados  $a$ ,  $b$  e  $c$ .
- Escreva uma função para calcular e retornar o volume de um cilindro de raio  $r$  e altura  $h$ . Esta função deverá receber como parâmetro o raio e a altura.
- Usando as funções dos itens anteriores, escreva um programa para calcular o volume de uma caixa com um furo cilíndrico, conforme desenho ao lado.



# Funções

---

FIM