

Funções e Métodos (Procedimentos) Linguagem C#

Subalgoritmos

São trechos de algoritmos que efetuam determinada tarefa

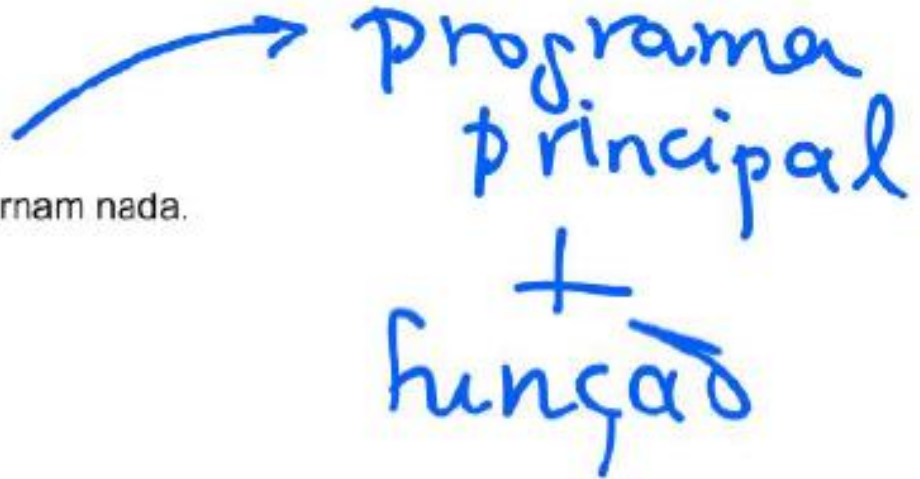
Ao invés de escrever um algoritmo grande, escrevem-se vários algoritmos menores, os quais, em conjunto, resolvem o problema proposto.

É especialmente indicado usá-los quando uma determinada tarefa efetuada em diversos lugares no mesmo algoritmo.

São declarados no início do algoritmo e podem ser chamados em qualquer ponto após a sua declaração.

Podem ser de dois tipos:

- Funções - Retornam algum valor.
- Procedimentos (Subrotinas) - retornam nada.



Programa principal
+
função

Um subalgoritmo é um trecho de programa que contém início e fim, executa um determinado conjunto de instruções e possui um identificador, por meio do qual pode ser chamado em qualquer parte do algoritmo, como se fosse um comando.

Elementos de um subalgoritmo

Os elementos de um subalgoritmo são o corpo e o cabeçalho.

No corpo, são definidas as instruções, ou seja, as ações que o subalgoritmo vai executar cada vez que for chamado dentro do algoritmo principal.

No cabeçalho, definimos o nome, os parâmetros, as variáveis locais e o tipo. Veja a descrição de cada um desses elementos a seguir.

Nome – é o identificador pelo qual o subalgoritmo será chamado no algoritmo principal.

Parâmetros – são os dados que permitem as relações entre o subalgoritmo e o algoritmo principal. Ou seja, são os dados que o subalgoritmo precisa receber para executar suas instruções e os dados que ele retorna quando termina de executar. Por exemplo, para realizar uma soma, os parâmetros recebidos são os números que serão somados. Ao efetuar a soma retornamos o resultado dessa soma.

Variáveis locais – são as variáveis declaradas no subalgoritmo e que só podem ser utilizadas dentro dele.

Tipo – os subalgoritmos podem ser de dois tipos: funções ou procedimentos. A diferença entre ambos é que uma função retorna sempre a um único valor. Um procedimento não retorna valores, mas os recebe e modifica.

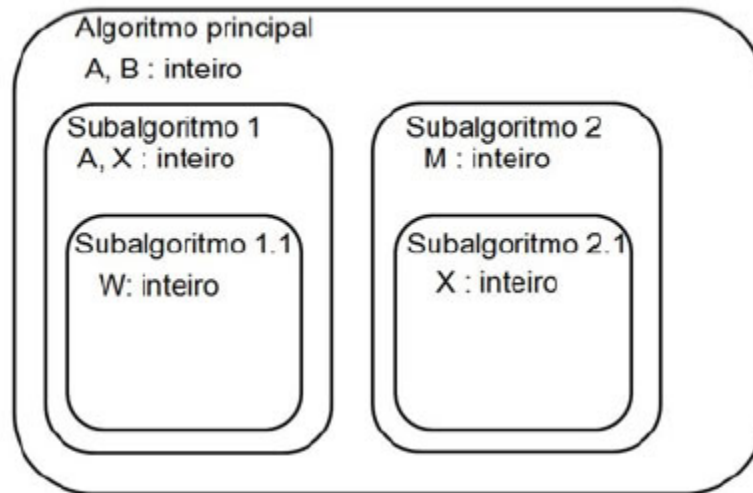
Escopo de variáveis

As variáveis que criamos no início de um algoritmo podem ser utilizadas em qualquer lugar dentro do algoritmo. Se desejarmos utilizar tais variáveis dentro de um subalgoritmo, por exemplo, podemos fazer isso sem problemas. Essas variáveis são as variáveis globais.

Ao criarmos um subalgoritmo, porém, criamos variáveis específicas para utilizarmos em seu interior. Tais variáveis são inicializadas no momento em que o subalgoritmo está sendo executado e são válidas somente em seu interior. Elas não são visualizadas fora dos limites do subalgoritmo. São as variáveis locais.

A essa visibilidade das variáveis (se ela é local ou global), damos o nome de escopo de variáveis. Trata-se da abrangência de uma variável, ou seja, em que limites do algoritmo ela é visível e pode ser utilizada.

subalgoritmo



Uma variável local pode ter o mesmo nome de uma variável global. Porém, uma vez declaradas em contextos diferentes, elas são distintas.

Uma função é um subalgoritmo que é chamado dentro do algoritmo através da citação de seu nome (identificador) e deve retornar um único valor.

As funções, assim como as variáveis, devem ser declaradas antes de serem inicializadas. Essa declaração deve estar posicionada no espaço após a declaração das variáveis e antes do início da execução do algoritmo. Assim:

```
Algoritmo "<nome do algoritmo>"
Var
    <declaração de variáveis do algoritmo>
```

```
<declaração da função>
```

```
Início
    <corpo do algoritmo>
finalgoritmo
```

A sintaxe da criação de uma função é a seguinte:

```
//cabeçalho, com o nome, os parâmetros e o tipo de retorno
Função <identificador> (<lista de parâmetros>:<tipo dos parâmetros>): <tipo de retorno>
//declaração das variáveis da função
Var
    <declaração das variáveis locais da função>
//início da execução da função
Início
    <instruções>
//valor a ser retornado no algoritmo, ao chamar a função
    retorne <valor de retorno>
fimfuncao
```

Funções	Procedimentos
As funções retornam o seu valor de forma explícita, por meio do comando <i>retorne</i> .	Os procedimentos não retornam valor. Não existe comando <i>retorne</i> .
As chamadas às funções ocorrem sempre em expressões ou instruções de atribuição.	Os procedimentos são chamados em comandos isolados, com as instruções de entrada e saída de dados (<i>leia</i> e <i>escreva</i>), e nunca em expressões ou atribuições.

Os procedimentos devem ser criados e declarados na mesma posição que as funções: após a declaração de variáveis dos algoritmos e antes do início da execução do mesmo. Veja:

Uma função é um subalgoritmo que é chamado dentro do algoritmo através da citação de seu nome (identificador) e deve retornar um único valor.

```
algoritmo "Função soma"
var
  a,b, soma: inteiro //declaração das variáveis globais

funcao fsoma(a1, b1: inteiro):inteiro //declaração da função
var
  resultado : inteiro //declaração de variáveis locais
Início
  resultado<-a1+b1 //instruções
  retorne resultado //valor de retorno
fimfuncao

início
  leia (a)
  leia (b)
  soma <- fsoma(a,b) //chamada da função
  escreva ("soma: ", soma)

finalgoritmo
```

Função criada

Corpo do algoritmo principal

Um procedimento é um subalgoritmo que é chamado dentro do algoritmo através da citação de seu nome (identificador) e deve alterar os valores dos parâmetros recebidos.

Os procedimentos devem ser criados e declarados na mesma posição que as funções: após a declaração de variáveis dos algoritmos e antes do início da execução do mesmo. Veja:

```
Algoritmo "<nome do algoritmo>"
Var
    <declaração de variáveis do algoritmo>

<declaração do procedimento>

Início
    <corpo do algoritmo>
finalgoritmo
```

Veja o algoritmo abaixo, que utiliza um procedimento que calcula a área de um triângulo:

```
algoritmo "Procedimento triangulo"
var
altura, base, área : real //declaração das variáveis globais

procedimento ptriangulo (var h,b,ar : real) //declaração do
procedimento
inicio
    ar <- (b*h)/2 //instruções
fimprocedimento

inicio
    escreval ("Digite o valor da base: ")
    leia (base)
    escreval ("Digite o valor da altura: ")
    leia (altura)
    ptriangulo(altura,base, área) //chamada ao procedimento
    escreval (area,"m²")

finalgoritmo
```

Procedimento criado

Corpo do algoritmo principal

Modularização

- Modularização é uma técnica de programação que se caracteriza pela divisão de um programa em sub-programas

Funções e Procedimentos em C#

- Na linguagem C#, definiremos função como uma rotina (bloco de comandos), com objetos próprios (variáveis, constantes, arquivos...), que realiza uma tarefa específica e retornando um valor (numérico, literal, lógico, endereço, etc.)
- Uma função que não retorna nenhum valor (retorno nulo) é chamada, em algumas linguagens, de procedimento, e na linguagem C#, em particular, de função com retorno do tipo **void**

Funções e Procedimentos em C#

- Notação

```
Static tipo_de_retorno nome_função (declaração  
de parâmetros) {  
    declaração dos objetos locais  
    bloco de comandos  
    return valor_de_retorno  
}
```

Funções e Procedimentos em C#

- Exemplo - Calcular o Cubo de um Número

```
public static int cubo(int L) { return L * L * L; }
```

```
static void Main(string[] args) {  
    Console.WriteLine(cubo(2));  
}
```

Funções e Procedimentos em C#

- Exemplo - Calcular o Fatorial de um Número

```
public static int Fatorial(int n) {  
    int Fat = 1;  
    int i = n;  
    while (i > 0){  
        Fat = Fat * i;  
        i--;  
    }  
    return Fat;  
}
```

Funções e Procedimentos em C#

- Exemplo - Calcular o Máximo entre dois números

```
static int maximo(int a, int b){  
    if(a>b)  
        return a;  
    else  
        return b;  
}
```


Passagem de Parâmetros por Valor e Referência

- Passagem de Parâmetros por Valor
 - Nesse tipo de passagem de parâmetro uma cópia do valor é efetuada pelo C#
 - Uma alteração em seus valores, nas respectivas sub-rotinas, não acarreta alteração nas variáveis externas a eles associadas, pois ocupam endereços de memória diferentes daquelas.

Passagem de Parâmetros por Valor e Referência

- Passagem de Parâmetros por Valor
 - Exemplo

```
static int somaUM(int a, int b) {  
    a++;  
    b++;  
    return a + b;  
}  
static void Main(string[] args){  
    int x = 3, y = 4;  
    somaUM(x, y);  
}
```

Passagem de Parâmetros por Valor e Referência

- Passagem por Referência
 - Os parâmetros por referência não ocupam um novo espaço de memória, mas apenas criam um “apelido” para a variável associada, e por isso são utilizados quando é grande a quantidade de dados a ser passado para a função.
 - Este tipo de parâmetro pode ser usado também para retornar valores de uma sub-rotina.
 - A linguagem C#, denota os parâmetros por referência com o termo **ref** (“referência”) antes da identificação dos tipos de dados

Passagem de Parâmetros por Valor e Referência

- Passagem por Referência

- Exemplo

```
static void Ordena(ref double a, ref double b){  
    double x;  
    if (a>b){ x = a;  a = b; b = x; }  
    return;  
}  
static void Main(string[] args)      {  
    double  x = 5, y = 4;  
    Ordena(ref x, ref y);  
    Console.WriteLine(x);  
    Console.WriteLine(y);  
}
```

Exercícios

1. Crie uma função que leia 3 números e, para cada um, imprimir o dobro de um número. Usar uma função que retorne valor.

2. Crie um algoritmo que receba 3 notas e calcule a sua média, utilizando uma função.

3. Crie um algoritmo (usando subalgoritmos) que leia uma temperatura dada em Farenheit e converta-a para Celsius.

Sabemos que:

$$tc = 5*(tf-32)/9$$

4. Crie um subalgoritmo para ler três (3) valores a, b e c reais e calcular e resolver a equação de segundo grau, usando a Fórmula de Báscara:

$$ax^2 + bx + c = 0 \text{ Sendo que } \text{delta} = b^2 - 4 \cdot a \cdot c$$

$$\text{se } \text{delta} > 0 \Rightarrow \text{duas raizes: } x_1 = (-b + \text{raiz}(\text{delta})) / 2 \cdot a$$

$$x_2 = (-b - \text{raiz}(\text{delta})) / 2 \cdot a$$

$$\text{Se } \text{delta} = 0 \Rightarrow \text{uma raiz igual a } x_1 = -b / 2 \cdot a$$

$$\text{Se } \text{delta} < 0 \Rightarrow \nexists \text{ raizes reais}$$

5. Crie um subalgoritmo que verifique se um número é primo ou não.

6. Crie um subalgoritmo que leia um número inteiro n e retorne na tela os seus divisores positivos:

Ex: $n = 28 \Rightarrow$ deve retornar: 1, 2, 4, 7, 14, 28

7. Crie um subalgoritmo que leia um número inteiro n e calcule o seu fatorial

Ex: $n = 7 \Rightarrow 7! = 1 * 2 * 3 * 4 * 5 * 6 * 7 = 5040$

```

algoritmo "Subalgoritmo - Fatorial"
var
    num: inteiro

funcao fatorial(n:inteiro):real
var
    fat,i: inteiro
inicio
    fat <- 1
    para i de 1 ate n faca
        fat <- fat * i
    fimpara
    retorne fat
fimfuncao
inicio
    escreval("FATORIAL DE UM NÚMERO - USANDO SUB-ALGORITMO")
    escreval("Digite um n° inteiro positivo: ")
    leia(num)
    escreval(num,"! = ",fatorial(num))
fimalgoritmo

```

8. Faça um sub-algoritmo que recebe por parâmetro o tempo de duração de um intervalo de tempo expresso em segundos e retorne esse parâmetro em horas, minutos e segundos.

```

algoritmo "Subalgoritmo - Conversão de Tempo"
var
    t: inteiro

procedimento converte(tempo:inteiro):inteiro
var
    h,m,s:inteiro
inicio
    h <- tempo \ 3600
    m <- (tempo % 3600) \ 60
    s <- (tempo % 3600) % 60
    escreval(tempo," seg. = ",h," h, ",m," min. e ",s," seg.")
fimprocedimento

inicio
    escreval("Digite um intervalo de tempo, em segundos: ")
    leia(t)
    converte(t)
fimalgoritmo

```

9. Elabore uma subrotina que receba como parâmetro a altura (alt) e o sexo de uma pessoa e retorne o seu peso ideal. Para homens, calcular o peso ideal, usando a fórmula:

$$\text{peso ideal} = 72.7 * \text{alt} - 58$$

e para as mulheres:

$$\text{peso ideal} = 62.1 * \text{alt} - 44.7$$

```

algoritmo "Subalgoritmo - Peso Ideal"
var
    altura: real
    sexo:inteiro

procedimento pesoideal(h:real; s:inteiro):real
var
    peso:real
inicio
    se (s = 0) entao
        peso <- 72.7 * h - 58
    senao
        peso <- 62.1 * h - 44.7
    fimse
    escreval("Peso ideal da pessoa = ",peso:3:1," kg")
fimprocedimento

inicio
    escreval("PESO IDEAL DE UMA PESSOA")
    escreval("Digite a altura da pessoa: ")
    leia(altura)
    escreval("Digite o sexo da pessoa: ([0] - Masc. | [1] - Fem.) ")
    leia(sexo)
    pesoideal(altura;sexo)
fimalgoritmo

```


10. Escreva uma função que recebe por parâmetro um valor inteiro e positivo N e retorna o valor de S.

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \dots + \frac{1}{N}.$$

```

algoritmo "Subalgoritmo - Soma de Frações"
var
    n: inteiro

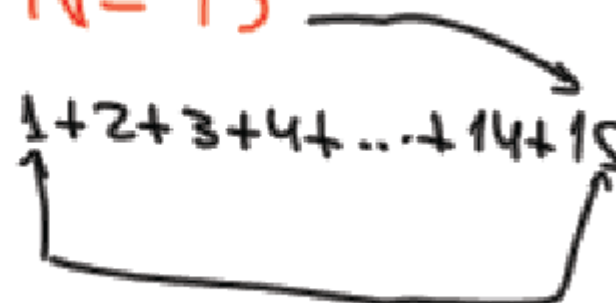
procedimento somafrac(num:inteiro):real
var
    i:inteiro
    s:real
inicio
    s <- 0
    para i de 1 ate num faca
        s <- s + 1/i
    fimpara
    escreval("Soma da série, com ",num," termos, vale: ",s:4:2)
fimprocedimento

inicio
    escreval("Digite o nº de termos da série: ")
    leia(n)
    somafrac(n)
fimalgoritmo

```

11. Crie uma sub-rotina que leia um número inteiro e positivo N como parâmetro e retorne a soma dos números inteiros existentes entre o número 1 e N.

Ex: $N = 15$

$$S = 1 + 2 + 3 + 4 + \dots + 14 + 15$$


```

algoritmo "SubAlgoritmo - Soma de n números"
var
    n: inteiro

funcao soma(num:inteiro):inteiro
var
    s,i:inteiro
inicio
    s <- 0
    para i de 1 ate num faca
        s <- s + i
    fimpara
    retorne s
fimfuncao
inicio
    escreval("Digite um n° inteiro: ")
    leia(n)
    escreval("A soma dos números entre 1 e ",n," vale: ",soma(n))
fimalgoritmo

```

12. Faça um procedimento que recebe 3 valores inteiros por parâmetro e retorne-os ordenados em ordem crescente.

Se $(n1 < n2)$ e $(n2 < n3) \Rightarrow n1, n2, n3$
Se $(n1 < n3)$ e $(n3 < n2) \Rightarrow n1, n3, n2$
Se $(n2 < n1)$ e $(n1 < n3) \Rightarrow n2, n1, n3$
Se $(n2 < n3)$ e $(n3 < n1) \Rightarrow n2, n3, n1$
Se $(n3 < n1)$ e $(n1 < n2) \Rightarrow n3, n1, n2$
Se $(n3 < n2)$ e $(n2 < n1) \Rightarrow n3, n2, n1$

```

algoritmo "Subalgoritmo - Ordena 3 números"
var
    n1,n2,n3: inteiro

procedimento ordena(num1,num2,num3:inteiro):inteiro
inicio
    se (num1 < num2) e (num2 < num3) entao
        escreval(num1,num2,num3)
    fimse
    se (num1 < num3) e (num3 < num2) entao
        escreval(num1,num3,num2)
    fimse
    se (num2 < num1) e (num1 < num3) entao
        escreval(num2,num1,num3)
    fimse
    se (num2 < num3) e (num3 < num1) entao
        escreval(num2,num3,num1)
    fimse
    se (num3 < num1) e (num1 < num2) entao
        escreval(num3,num1,num2)
    fimse
    se (num3 < num2) e (num2 < num1) entao
        escreval(num3,num2,num1)
    fimse
fimprocedimento

inicio
    escreval("Subalgoritmo que ordena 3 números distintos")
    escreval("Digite um 1º nº inteiro: ")
    leia(n1)
    escreval("Digite um 2º nº inteiro: ")
    leia(n2)
    escreval("Digite um 3º nº inteiro: ")
    leia(n3)
    escreval
    escreval("Números Ordenados: ")
    escreval
    ordena(n1,n2,n3)
fimalgoritmo

```

13. Crie uma subrotina que cheque se três valores inteiros distintos A, B e C formam um triângulo e classificá-los:

OBS: A, B e C formam um triângulo $\Leftrightarrow A < (B+C)$ e $B < (A+C)$ e $C < (A+B)$

Se $(A = B)$ e $(B = C) \Rightarrow$ Triângulo Equilátero

Se $(A = B)$ ou $(B = C)$ ou $(A = C) \Rightarrow$ Triângulo Isósceles

Se $(A \neq B)$ e $(B \neq C)$ e $(A \neq C) \Rightarrow$ Triângulo Escaleno

```

algoritmo "Classifica Triângulos Funções"
var
    ladoa,ladob,ladoc: inteiro
procedimento classifica(a,b,c:inteiro):inteiro
inicio
    se (a < (b+c)) e (b < (a+c)) e (c < (a+b)) entao
        escreva(a," ", "b," e "c," formam um triângulo ")
        se ((a=b) e (b=c)) entao
            escreva(" equilátero")
        fimse
        se ((a=b) ou (b=c) ou (a=c)) entao
            escreva(" isósceles")
        fimse
        se ((a<>b) e (b<>c) e (a<>c)) entao
            escreva(" escaleno")
        fimse
    senao
        escreval(a," ", "b," e "c," não formam um triângulo!!! ")
    fimse
fimprocedimento
inicio
    escreval("Digite o 1º número:")
    leia(ladoa)
    escreval("Digite o 2º número:")
    leia(ladob)
    escreval("Digite o 3º número:")
    leia(ladoc)
    classifica(ladoa,ladob,ladoc)
fimalgoritmo

```


14. Crie uma subrotina que leia dois vetores A e B, de 5 posições e crie e mostre os vetores SOMA e DIF, que são as somas dos respectivos termos de A e B e a diferença entre os termos de A e B, respectivamente:

15. Faça um programa que simule um calculador de 4 operações.

Seu programa deverá:

- a) ler dois números e o operador;
- b) chamar procedimentos com passagem de parâmetros para efetuar os cálculos;
- c) escrever o resultado ao final.

Microsoft S2B Students to Business

www.programas2b.com.br

C# ADO.NET e ASP.NET

VB e C# => www.macoratti.net

Java

www.caelum.com.br => Apostila Java

www.globalcode.com.br => Apostila Java