

# Funções

# Introdução

A modularização de código diminui a quantidade de linhas repetidas, pois permite reaproveitamento de código...

... tornando o programa mais compacto e de mais fácil manutenção

Até aqui já utilizamos funções em C: *printf( )* e *scanf( )*

Essas funções são “chamadas” e são passados argumentos (que também chamamos de parâmetros)

Agora vamos aprender a criar nossas próprias funções

# Exemplo 1

Ambas as funções recebem possuem um tipo, que define o que será retornado

Ambas recebem um parâmetro

As variáveis usadas dentro de uma função são chamadas variáveis locais...

... só existem dentro das funções

... e não alteram nenhuma variável fora delas

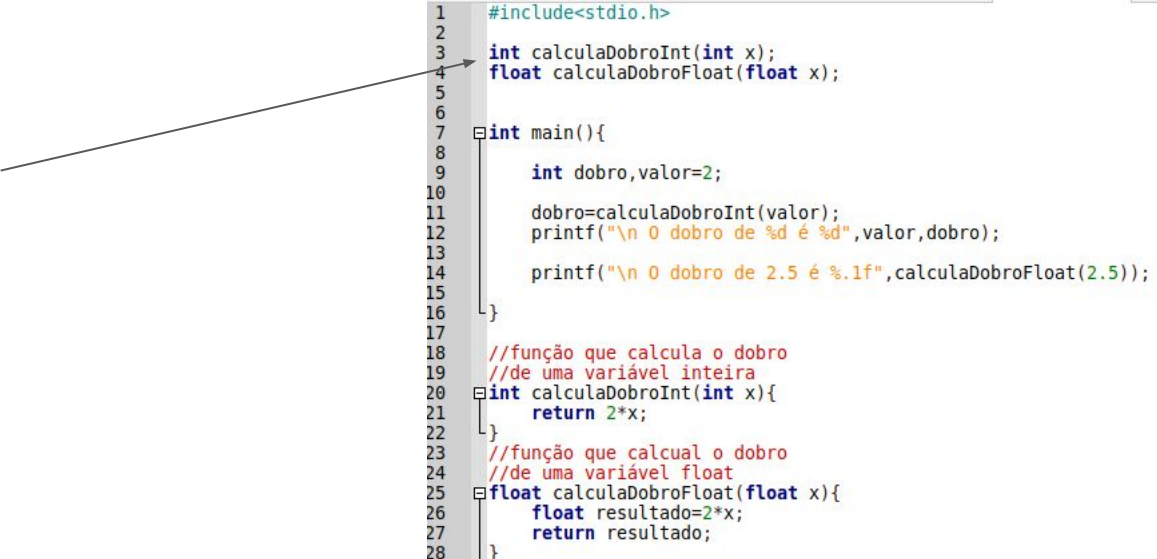
... mesmo que tenham o mesmo nome

```
1  #include<stdio.h>
2
3  //função que calcula o dobro
4  //de uma variável inteira
5  int calculaDobroInt(int x){
6      return 2*x;
7  }
8  //função que calcula o dobro
9  //de uma variável float
10 float calculaDobroFloat(float x){
11     float resultado=2*x;
12     return resultado;
13 }
14
15 int main(){
16
17     int dobro,valor=2;
18
19     dobro=calculaDobroInt(valor);
20     printf("\n O dobro de %d é %d",valor,dobro);
21
22     printf("\n O dobro de 2.5 é %.1f",calculaDobroFloat(2.5));
23
24 }
```

## Exemplo 2

Quando temos muitas funções, fica mais organizado deixamos ela no fim do programa, após a função principal

Para tanto, é necessário criar protótipos das funções



```
1  #include<stdio.h>
2
3  int calculaDobroInt(int x);
4  float calculaDobroFloat(float x);
5
6
7  int main(){
8
9      int dobro,valor=2;
10
11      dobro=calculaDobroInt(valor);
12      printf("\n O dobro de %d é %d",valor,dobro);
13
14      printf("\n O dobro de 2.5 é %.1f",calculaDobroFloat(2.5));
15
16  }
17
18  //função que calcula o dobro
19  //de uma variável inteira
20  int calculaDobroInt(int x){
21      return 2*x;
22  }
23
24  //função que calcula o dobro
25  //de uma variável float
26  float calculaDobroFloat(float x){
27      float resultado=2*x;
28      return resultado;
29  }
```

# Exemplo 3

Uma função pode ter dois ou mais parâmetros...

Também pode não receber parâmetros

```
1  #include<stdio.h>
2
3  int soma(int x,int y);
4
5  void boasVindas();
6
7  int main(){
8
9      boasVindas();
10     int num1=1,num2=2,resultado;
11
12     resultado=soma(num1,num2);
13     printf("\n A soma de %d e %d é %d",num1,num2,resultado);
14
15 }
16
17 int soma(int x, int y){
18     return x+y;
19 }
20
21 void boasVindas(){
22     printf("\n*****");
23     printf("\n*****Bem vindo*****");
24     printf("\n*****\n");
25 }
```

```
*****
*****Bem vindo*****
*****

A soma de 1 e 2 é 3
```

# Exemplo 4

Pode-se passar um vetor como parâmetro

Passa-se o ponteiro para o vetor

```
1  #include<stdio.h>
2
3  float media(int numNotas, float *notas);
4
5  int main(){
6      int numNotas=3;
7      float notas[]={10,5,8}, resultado;
8
9      resultado=media(numNotas, notas);
10     printf("\n A média é: %.2f", resultado);
11
12 }
13 float media(int numNotas, float *notas){
14     float soma=0;
15     for(int i=0; i<numNotas; i++){
16         soma=soma+notas[i];
17     }
18     return soma/numNotas;
19 }
```

## Exemplo 5 (não será cobrado em avaliação)

Uma função pode chamar ela mesma (recursividade)

Na linha 11, a função fatorial formará a sequência:  $5*4*3*2*1$

```
1  #include<stdio.h>
2
3  long fatorial(long n);
4
5
6
7  int main(){
8
9      long num,resultado;
10     num=5;
11     resultado=fatorial(num);
12     printf("\n O fatorial de %ld é %ld",num,resultado);
13
14 }
15
16
17 long fatorial(long n){
18     if (n>1)
19         return n*fatorial(n-1);
20
21     else
22         return 1;
23
24 }
```

# Exemplo 6 (não será cobrado em avaliação)

Quando se passa o endereço de uma variável (parâmetro por referência)...

... a variável é alterada

```
1  #include<stdio.h>
2
3  int parametroCopia(int n);
4  int parametroReferencia(int *n);
5
6
7
8  int main(){
9
10     int n=5,resultado;
11
12     resultado=parametroCopia(n);
13     printf("\n O valor de n é %d e o resultado é %d",n,resultado);
14
15     //passa endereço de n como parâmetro
16     resultado=parametroReferencia(&n);
17     printf("\n O valor de n é %d e o resultado é %d",n,resultado);
18
19 }
20
21 //nesta função é feita a cópia do valor de n
22 int parametroCopia(int n){
23     n=2*n;
24     return n;
25 }
26
27
28 //nesta função a operação ocorre no endereço de n
29 int parametroReferencia(int *n){
30     *n=2*(*n);
31     int resultado=*n;
32     return resultado;
33 }
34 }
```