

ESTRUTURA DE DADOS I

Ramon Lummertz

FUNÇÕES

FUNÇÕES

```
pow(a,b);  
  
printf("Ola");  
  
main();
```

```
System.out.println("Mensagem");
```

FUNÇÕES

Uma função é um pedaço de código que faz alguma tarefa específica e pode ser chamado de qualquer parte do programa quantas vezes desejarmos.

Clareza do código: separando pedaços de código da função `main()`, podemos entender mais facilmente o que cada parte do código faz. Além disso, para procurarmos por uma certa ação feita pelo programa, basta buscar a função correspondente. Isso torna muito mais fácil o ato de procurar por erros.

Reutilização: muitas vezes queremos executar uma certa tarefa várias vezes ao longo do programa. Repetir todo o código para essa operação é muito trabalhoso, e torna mais difícil a manutenção do código: se acharmos um erro nesse código, teremos que corrigí-lo em todas as repetições do código. Chamar uma função diversas vezes contorna esses dois problemas.

Independência: uma função é relativamente independente do código que a chamou. Uma função pode modificar variáveis globais ou ponteiros, mas limitando-se aos dados fornecidos pela chamada de função. A ideia de uma função está, naturalmente, permitir você encapsular um ideia ou operação, dando um nome a ela, então chamar que operação de várias partes do resto de seu programa simplesmente usando o seu nome. Estando corretamente projetado e estruturado o programa, em uma situação ideal, deverá ser possível modificar as funções sem efeito no resto do programa.

DEFININDO UMA FUNÇÃO

Uma função pode necessitar de alguns dados para que possa realizar alguma ação baseada neles. Esses dados são chamados parâmetros da função. Além disso, a função pode retornar um certo valor, que é chamado valor de retorno. Os parâmetros (e seus tipos) devem ser especificados explicitamente, assim como o tipo do valor de retorno.

```
[tipo de retorno da função] [nome da função] (1º parâmetro, 2º  
parâmetro, ...)  
{  
    //código  
}
```

Para o nome da função e dos parâmetros valem as mesmas regras que foram dadas para os nomes de variáveis. Não podemos usar o mesmo nome para funções diferentes em um programa. Todas as funções devem ser definidas antes da função main, ou deve ser feito o protótipo da função, que veremos mais adiante. O código deve estar obrigatoriamente dentro das chaves e funciona como qualquer outro bloco

VALOR DE RETORNO

Frequentemente, uma função faz algum tipo de processamento ou cálculo e precisa retornar o resultado desse procedimento. Em C, isso se chama valor de retorno e pode ser feito com a instrução `return`. Para poder retornar um valor, precisamos especificar seu tipo (`char`, `int`, `float`, `double` e variações). Para efetivamente retornar um valor, usamos a instrução `return` seguida do valor de retorno, que pode ou não vir entre parênteses. Um exemplo bem simples de função que retorna um valor inteiro:

```
int tres()
{
    return 3;  // poderia também ser return (3);
}
```

O tipo de retorno, além dos tipos normais de variáveis (`char`, `int`, `float`, `double` e suas variações), pode ser o tipo especial `void`, que na verdade significa que não há valor de retorno.

Nota Muitos livros dizem que a função `main` tem tipo de retorno `void`, o que não está correto. Segundo o padrão da linguagem C, a função `main` deve ter retorno do tipo `int`. Compiladores como o `gcc` darão mensagens de erro caso a função `main()` não seja definida corretamente.

PARÂMETROS

Como já foi dito, um parâmetro é um valor que é fornecido à função quando ela é chamada. É comum também chamar os parâmetros de argumentos, embora argumento esteja associado ao valor de um parâmetro.

```
int funcao (int a, int b)
float funcao (float preco, int quantidade)
double funcao (double angulo)
```

Para especificar que a função não usa nenhum parâmetro, a lista de parâmetros deve conter apenas a palavra-chave void. No entanto, ela é frequentemente omitida nesses casos. Portanto, você poderia escrever qualquer uma destas duas linhas:

```
void funcao (void)
void funcao ()
```

CHAMANDO UMA FUNÇÃO

Uma função pode ser chamada dentro de outra função.
Para isso basta chamar a função pelo e passar os parâmetros que a mesma precisa.
Importante lembrar que funções que retornar um valor devem ser atribuídas em uma variável.

QUANDO UMA FUNÇÃO ENCERRA

Uma função encerra sua execução quando:
o fim do seu código é atingido;
ou
um comando return é encontrado e executado.

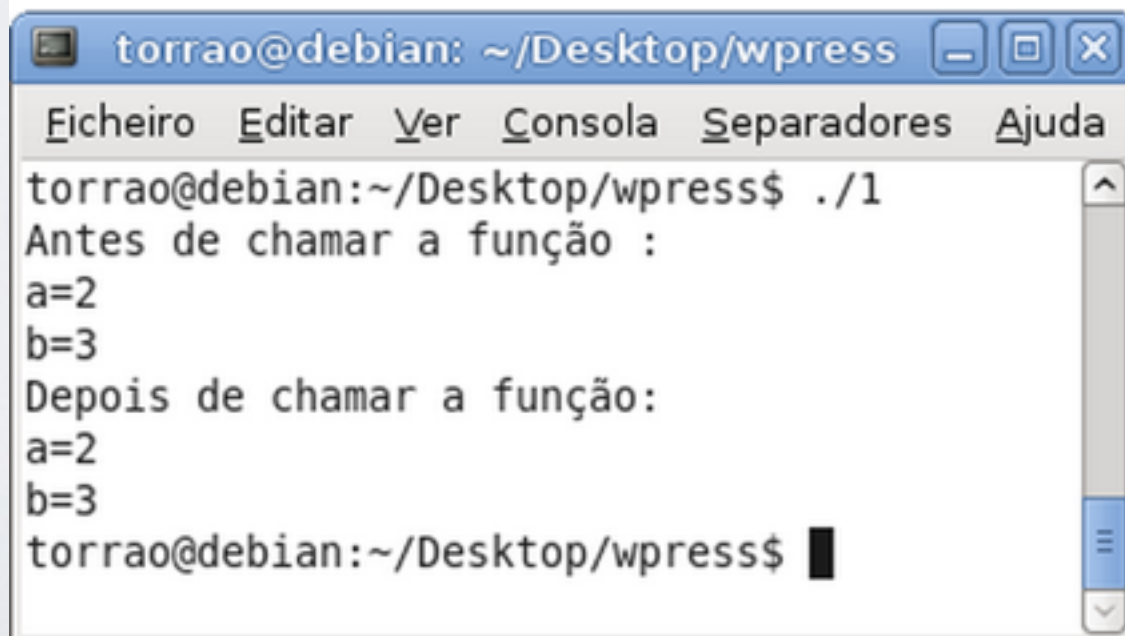
VALOR X REFERÊNCIA

Passagem de parâmetros por valor: A função recebe uma cópia da variável que é fornecida quando é invocada. Todas as alterações feitas dentro da função não vão afectar os valores originais.

Passagem de parâmetros por referência: Neste caso o que é enviado para a função é uma referência às variáveis utilizadas, e não uma simples cópia, pelo que as alterações realizadas dentro da função irão certamente alterar os valores contidos nessas variáveis.

EXEMPLO EM VALOR

```
1  #include<stdio.h>
2  void troca(int a, int b){
3      int temp;
4      temp=a;
5      a=b;
6      b=temp;
7  }
8  int main(){
9      int a=2,b=3;
10     printf("Antes de chamar a função :\na=%d\nb=%d\n",a,b);
11     troca(a,b);
12     printf("Depois de chamar a função:\na=%d\nb=%d\n",a,b);
13     return 0;
14 }
```

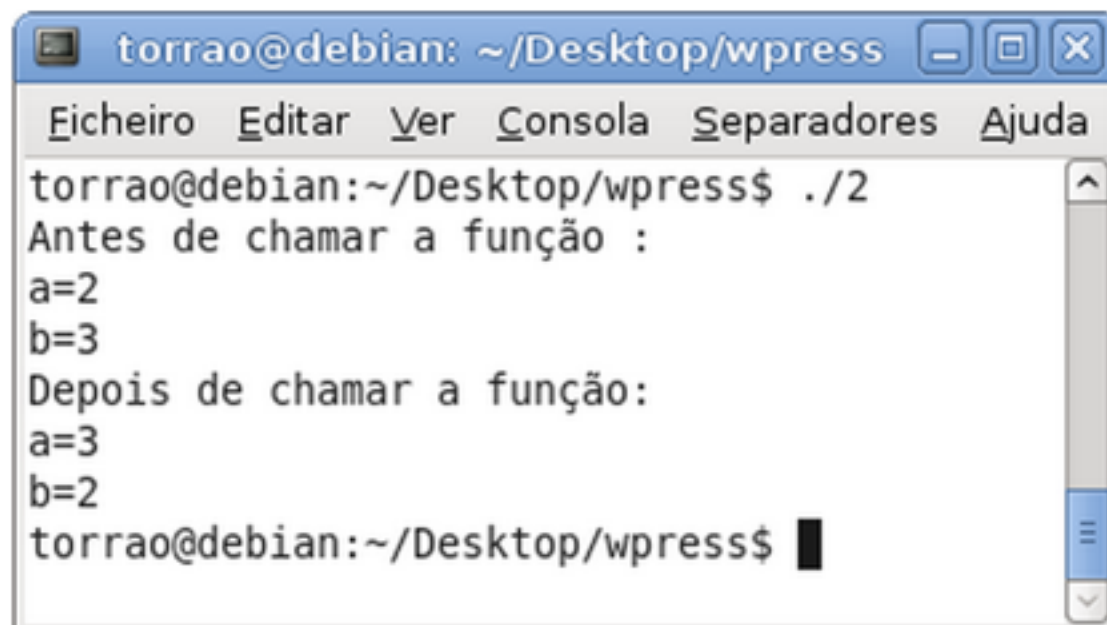


The screenshot shows a terminal window titled "torrao@debian: ~/Desktop/wpress". The window has a menu bar with "Ficheiro", "Editar", "Ver", "Consola", "Separadores", and "Ajuda". The terminal output shows the execution of the program: the user runs "./1", and the program prints "Antes de chamar a função :", "a=2", "b=3", "Depois de chamar a função:", "a=2", and "b=3". The prompt "torrao@debian:~/Desktop/wpress\$" is visible at the bottom.

```
torrao@debian: ~/Desktop/wpress
Ficheiro  Editar  Ver  Consola  Separadores  Ajuda
torrao@debian:~/Desktop/wpress$ ./1
Antes de chamar a função :
a=2
b=3
Depois de chamar a função:
a=2
b=3
torrao@debian:~/Desktop/wpress$
```

EXEMPLO EM C REFERÊNCIA

```
1  #include<stdio.h>
2  void troca(int *a, int *b){
3      int temp;
4      temp=*a;
5      *a=*b;
6      *b=temp;
7  }
8  int main(){
9      int a=2,b=3;
10     printf("Antes de chamar a função :\na=%d\nb=%d\n",a,b);
11     troca(&a,&b);
12     printf("Depois de chamar a função:\na=%d\nb=%d\n",a,b);
13     return 0;
14 }
```



The screenshot shows a terminal window titled "torrao@debian: ~/Desktop/wpress". The window has a menu bar with "Ficheiro", "Editar", "Ver", "Consola", "Separadores", and "Ajuda". The terminal output shows the execution of the program: "torrao@debian:~/Desktop/wpress\$./2", followed by the output of the first printf statement: "Antes de chamar a função :", "a=2", and "b=3". Then, the output of the second printf statement is shown: "Depois de chamar a função:", "a=3", and "b=2". The prompt "torrao@debian:~/Desktop/wpress\$" is visible at the bottom.

```
torrao@debian: ~/Desktop/wpress
Ficheiro  Editar  Ver  Consola  Separadores  Ajuda
torrao@debian:~/Desktop/wpress$ ./2
Antes de chamar a função :
a=2
b=3
Depois de chamar a função:
a=3
b=2
torrao@debian:~/Desktop/wpress$
```


EX: EM JAVA POR VALOR

```
1  /**
2   * Created by ramon on 23/08/16.
3   */
4  public class FuncaoValor {
5
6      public static void main(String[] args) {
7          int a;
8          a=5;
9          System.out.println("valor original:"+ a);
10         troca(a);
11         System.out.println("valor original:"+ a);
12     }
13
14     public static void troca(int b){
15         b=10;
16     }
17 }
18
```

```
▶ /Library/Java/JavaVirtualMachines/jdk1.8.0_51.jdk/Contents/Home/bin/java ...
valor original:5
valor original:5

Process finished with exit code 0
```


EX: EM JAVA POR REFERÊNCIA

```
1  /**
2   * Created by ramon on 23/08/16.
3   */
4  class Obj{
5      int a;
6  }
7
8  public class FuncaoRef {
9      public static void main(String[] args) {
10         Obj v = new Obj();
11         v.a=5;
12         System.out.println("Valor original "+v.a);
13         troca(v);
14         System.out.println("Valor original "+v.a);
15     }
16
17     public static void troca(Obj b){
18         b.a=10;
19     }
20 }
21
22
```

```
▶ /Library/Java/JavaVirtualMachines/jdk1.8.0_51.jdk/Contents/Home/bin/java ...
Valor original 5
Valor original 10
Process finished with exit code 0
```

EXERCÍCIOS

- 1 - Criar uma função que some os elementos de um vetor recebidos por parâmetro. Fazer a chamada no main. - A função deve retornar a soma do valor
- 2 - Fazer uma função que conte o numero de vogais numa string passada por parâmetro. Fazer a chamada no main. A função deve retornar a quantidade de vogais.
- 3 - Escreva uma uma função para mostrar se o numero é par. Se for par a função deve retornar 0, caso contrario 1.
- 4 - Fazer uma função que conte o numero de consoantes numa string passada por parâmetro. Fazer a chamada no main. A função deve alterar a variável nConsoantes que foi declarada no main.
- 5 - Faça uma função que some dois números, entretanto a função deve garantir que o primeiro argumento é um numero par e o segundo um numero impar