

MCTA028-15: Programação Estruturada

Aula 3: Funções (Segunda Parte)

Wagner Tanaka Botelho

wagner.tanaka@ufabc.edu.br / wagtanaka@gmail.com

Universidade Federal do ABC (UFABC)

Centro de Matemática, Computação e Cognição (CMCC)

Introdução

- Uma função é uma **sequência de comandos** que recebe um **nome** e pode ser **chamada** de qualquer **parte do programa**, quantas vezes forem necessárias, durante a sua execução;
- Você sabia que a **Linguagem C** possui muitas **funções** já implementadas? Por exemplo:
 - Funções básicas de entrada e saída, como **scanf()** e **printf()**;
 - Um detalhe, o programador **NÃO** precisa saber o **código** contido dentro das funções:
 - Neste caso, basta saber o seu **nome** e como **utilizá-la**.
- Principais **razões** para criar funções:
 - **Estruturação** dos programas;
 - **Reutilização** de código.

Como Criar Uma Função?

SINTAXE

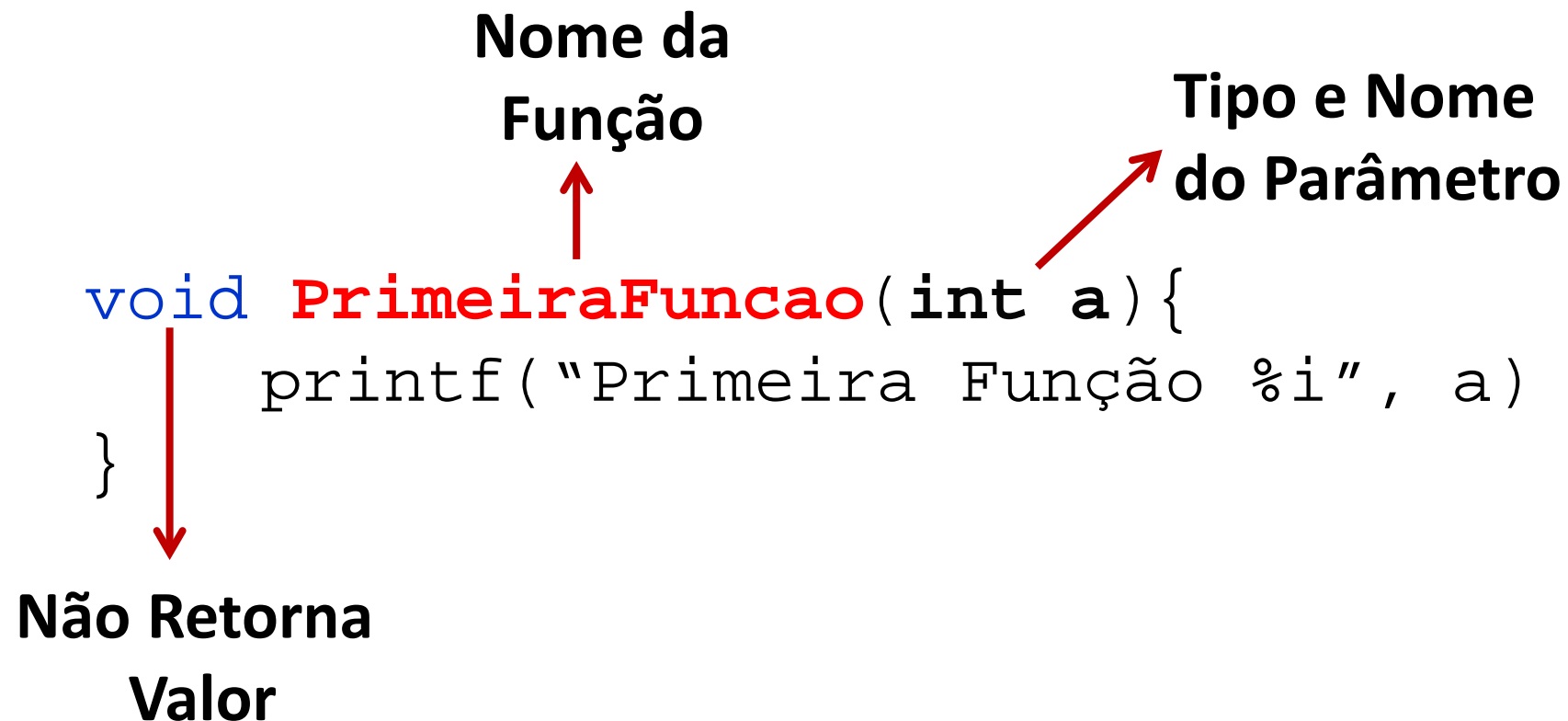
```
tipo_retornado NOME_FUNÇÃO (Lista_De_Parâmetros) {  
    sequência de declarações e comandos  
}
```

Nome da
Função

Tipo e Nome
do Parâmetro

```
void PrimeiraFuncao(int a) {  
    printf("Primeira Função %i", a)  
}
```

Não Retorna
Valor



Parâmetros da Função

Parâmetros da Função

- Os **parâmetros** são utilizados para **passar** a informação de um **trecho de código** para **dentro** da função;
- Basicamente, são uma **lista de variáveis**, separadas por **vírgula**, em que são especificados o **tipo** e o **nome** de cada variável passada para a função;
- Os parâmetros devem ser declarados da seguinte forma:

SINTAXE

```
tipo_retornado NOME_FUNÇÃO (tipo nome1, tipo nome2, ..., tipo nomeN) {  
    sequência de declarações e comandos  
}
```

Função Sem Lista de Parâmetros

Função Sem Lista de Parâmetros

- Uma função pode **NÃO** possuir nenhum parâmetro;
- Pode-se optar por **duas** soluções:
 - Deixar a lista de parâmetros **VAZIA**:

SINTAXE

```
tipo_retornado NOME_FUNÇÃO () {  
    sequência de declarações e comandos  
}
```

- Colocar **void** entre parênteses:

SINTAXE

```
tipo_retornado NOME_FUNÇÃO (void) {  
    sequência de declarações e comandos  
}
```



```
Ex_08.c x
1  #include <stdio.h>
2
3  void Imprime_Testes() {
4      printf("Testando\n");
5  }
6
7  int main() {
8      Imprime_Testes();
9      Imprime_Testes(5, 10, 30, "TESTE");
10     Imprime_Testes(5, 'a');
11 }
```

```
D:\UFABC\Disciplin
Testando
Testando
Testando
Process returned
```

- Não é especificado **NENHUM** parâmetro:
- Portanto, a função pode ser **CHAMADA** passando-se **valores** para ela;
- O compilador **NÃO** vai verificar se a função é realmente chamada **sem argumentos**, e a função **NÃO** conseguirá ter acesso a esses parâmetros.

Ex_09.c

```
1  #include <stdio.h>
2
3  void Imprime Teste(void){
4      printf("Testando\n");
5  }
6
7  int main(){
8      Imprime_Testes();
9      Imprime_Testes(5, 10, 30, "TESTE");
10     Imprime_Testes(5, 'a');
11 }
```

- Imprime_Testes(void):
 - Com o **void**, **nenhum** parâmetro é esperado;
 - O programa acusará um **ERRO** se o programador **tentar** passar um **valor** para essa função.

Logs & others		
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++		
File	Line	Message
=== Build: Debug in Aula03 (compiler: GNU GCC Compiler) ===		
In function 'main':		
D:\UFABC\Di...	9	error: too many arguments to function 'Imprime_Testes'
D:\UFABC\Di...	3	note: declared here
D:\UFABC\Di...	10	error: too many arguments to function 'Imprime_Testes'
D:\UFABC\Di...	3	note: declared here
=== Build failed: 2 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===		

Retorno da Função

Retorno da Função

- É a forma como uma função **devolve** o resultado (se ele existir) da sua execução para quem a **chamou**;
- Considerando a **sintaxe**:

SINTAXE
<pre>tipo_retornado NOME_FUNÇÃO (Lista_De_Parâmetros) { sequência de declarações e comandos }</pre>

- **tipo_retornado**:
 - Estabelece o **tipo de valor** que a função vai devolver para quem a chamar.
- Uma função pode retornar **qualquer** tipo válido:
 - Tipos **básicos predefinidos** (**int**, **char**, **float**, **double**, **void** e ponteiros).
 - Tipos **definidos pelo programador**: **struct**, **array** (indiretamente) etc.

Sem Retorno de Valor

Ex_10.c X

```
1  #include <stdio.h>
2
3  void Imprime_Teste(int qtd) {
4      int cont=0;
5
6      while (cont<qtd) {
7          printf("UFABC\n");
8          cont = cont + 1;
9      }
10 }
11
12 void main() {
13     Imprime_Teste(3);
14 }
```

 D:\UFABC\Disciplinas\2021-

UFABC

UFABC

UFABC

Process returned 0 (0x0)

Press any key to continue

Uma função também pode NÃO retornar um valor. Para isso, basta colocar o tipo `void` como valor retornado.

Com Retorno de Valor

Ex_11.c X

```
1  #include <stdio.h>
2
3  int Calcular Soma(int x, int y){
4      int soma = 0;
5
6      soma = x + y;
7
8      return soma;
9  }
10
11 void main(){
12     int x, y, result;
13
14     printf("Digite x:\n");
15     scanf("%d", &x);
16
17     printf("Digite y:\n");
18     scanf("%d", &y);
19
20     result = Calcular_Soma(x, y);
21     printf("Soma eh = %d\n", result);
22 }
```

IMPORTANTE!
Os tipos são IGUAIS!!

➤ Se a função não for do tipo **void**, ela deverá retornar um valor:

- O comando **return** é utilizado para retornar esse valor para o programa;;
- A expressão da cláusula **return** tem de ser **compatível** com o **tipo de retorno** declarado para a função.

D:\UFABC\Disciplinas\2021-2025'

```
Digite x:
3
Digite y:
4
Soma eh = 7
```

```
Process returned 12 (0xC)
Press any key to continue.
```


Ex_12.c X

```
1  #include <stdio.h>
2
3  int Maior_Menor(int x, int y){
4
5      if(x > y){
6          return x;
7      }
8      else{
9          return y;
10     }
11 }
12
13 void main(){
14     int x, y, result;
15
16     printf("Digite x:\n");
17     scanf("%d", &x);
18
19     printf("Digite y:\n");
20     scanf("%d", &y);
21
22     result = Maior_Menor(x, y);
23     printf("Maior numero eh = %d\n", result);
24 }
```

Uma função pode ter mais de uma declaração return.

D:\UFABC\Disciplinas\2021-202

```
Digite x:
4
Digite y:
9
Maior numero eh = 9

Process returned 20 (0x14)
Press any key to continue.
```

Eu



Main -> Como resolver
 $(a+b)*c$ e imprimir o
resultado na tela??

 $a=2$ e $b=4$

6

 6 e $c=5$

30

Somar

Multiplicar



Carlos



Maria

Referências

- Slides do Prof. Luiz Rozante;
- SALES, André Barros de; AMVAME-NZE, Georges. Linguagem C: roteiro de experimentos para aulas práticas. 2016;
- BACKES, André. Linguagem C Completa e Descomplicada. Editora Campus. 2013;
- SCHILDT, Herbert. C Completo e Total. Makron Books. 1996;
- DAMAS, Luís. Linguagem C. LTC Editora. 1999;
- DEITEL, Paul e DEITEL, Harvey. C Como Programar. Pearson. 2011.