UNIVERSIDADE
FEDERAL DO CARIRI

Funções

Professora Dra. Luana Batista da Cruz
luana.batista@ufca.edu.br

Roteiro

01 Funções

02 Exemplos

01

Funções

Definição de funções
Estrutura de uma função
Escopo de variáveis
Exemplos

Funções

- As funções permitem a criação de um programa em módulos. Portanto, uma função é um conjunto de instruções desenhadas para cumprir determinada tarefa e agrupadas em uma unidade com um nome para referenciá-la

- Qualquer sequência de instruções que apareça mais de uma vez no programa é candidata a ser uma função

- Portanto, a existência de funções evita que o programador tenha de escrever o mesmo código repetidas vezes

- O código de uma função é agregado ao programa uma única vez e pode ser executado muitas vezes no decorrer do programa

Funções

- Funções são chamadas, ou invocadas, por uma chamada de função, que especifica o nome da função e oferece informações, como argumentos, de que a função chamada precisa para realizar sua tarefa designada

- **Analogia**: chefe e subordinado. O chefe é a função chamadora, o subordinado é a função chamada. O chefe pede a um subordinado que realize uma tarefa e informe quando ela tiver sido concluída. O chefe não sabe exatamente como o subordinado realiza suas tarefas. O subordinado pode chamar outros para ajudá-lo a realizar as tarefas sem o chefe saber

Funções

- Funções dividem grandes tarefas de computação em tarefas menores, e permitem às pessoas trabalharem sobre o que outras já fizeram, em vez de partir do nada

- Evite reinventar a roda. Quando possível, use as funções da biblioteca-padrão de C em vez de escrever novas funções. Isso pode reduzir o tempo de desenvolvimento do programa

- Funções apropriadas podem frequentemente esconder detalhes de operação de partes do programa que não necessitam conhecê-las

- Você já usou a função printf() sem conhecer detalhes de sua programação

Principais objetivos de uma função

- Dividir e estruturar um algoritmo em partes logicamente coerentes
- Modularizar um programa em partes menores
- Facilitar o teste de trechos de código em separado
- Proporcionar ao programador a reutilização de código, através da criação de bibliotecas de funções personalizadas
- Aumentar a legibilidade e manutenibilidade do programa
- Evitar a repetição de código, substituindo códigos semelhantes por chamadas a uma única função

Estrutura de uma função

```
<tipo_retorno> <nome_função>(<lista_declaração_parâmetro>){
        <corpo_função>
        return <valor_de_retorno>
}
```

- Onde
  - **<tipo_retorno>**: é o tipo do valor que a função retorna; quando a função não retorna nenhum valor utiliza-se a palavra chave void
  - **<nome_função>**: é o identificador que nomeia a função
  - **<lista_declaração_parâmetro>**: é uma lista de variáveis (vazia ou não) separadas por vírgulas onde valores serão dados
  - **<corpo_função>**: descreve o comportamento/comandos da função
  - **<valor_de_retorno>**: valor que será retornado pela função
  - **Toda função** deve começar com uma chave de abertura de bloco { e terminar com uma chave de fechamento de bloco }, delimitando o corpo da função

Uso de funções

- Bibliotecas C/C++ são compostas de funções, de forma a permitir que o programador reaproveite códigos existentes

```
// função que calcula a raiz quadrada
double x = sqrt(y);

// função para gerar números aleatórios
int numero = rand();
```

Escopo de variáveis

- **Variáveis locais**
  - São aquelas declaradas dentro do bloco de uma função ou main
  - Não podem ser usadas ou modificadas por outras funções
  - As variáveis da lista de parâmetros de uma função também são variáveis locais
  - Somente existem enquanto a função onde foi declarada estiver sendo executada

- **Variáveis globais**
  - São declaradas fora de todos os blocos de funções e main
  - São acessíveis em qualquer parte do programa, ou seja, podem ser usadas e modificadas por todas as outras funções
  - Existem durante toda a execução do programa

Escopo de variáveis

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// declaração de variáveis globais

int funcao(variáveis locais de parâmetros){

   // declaração das variáveis locais da função

   return;
}

int main(){

    // declaração das variáveis locais da main()

    system("PAUSE");
    return 0;
}
```

Escopo de variáveis

- **Importante**
  - Mesmo que as variáveis possuam o mesmo nome na main e em uma função, o compilador enxerga como variáveis distintas. Portanto, as variáveis irão se comportar como variáveis diferentes

  - Pode-se declarar variáveis globais, para serem utilizadas em todo o programa. Porém, seu uso não é uma boa prática de programação, devendo ser usado apenas quando estritamente necessário

Passagem de parâmetros

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

Exemplo de conversão de temperaturas

- Fahrenheit e Celsius são duas escalas usadas para medir a temperatura
- Desenvolveremos um programa para converter as temperaturas em Celsius para temperaturas equivalentes em Fahrenheit
- A fórmula para conversão é

$$F = 1.8 \times C + 32$$

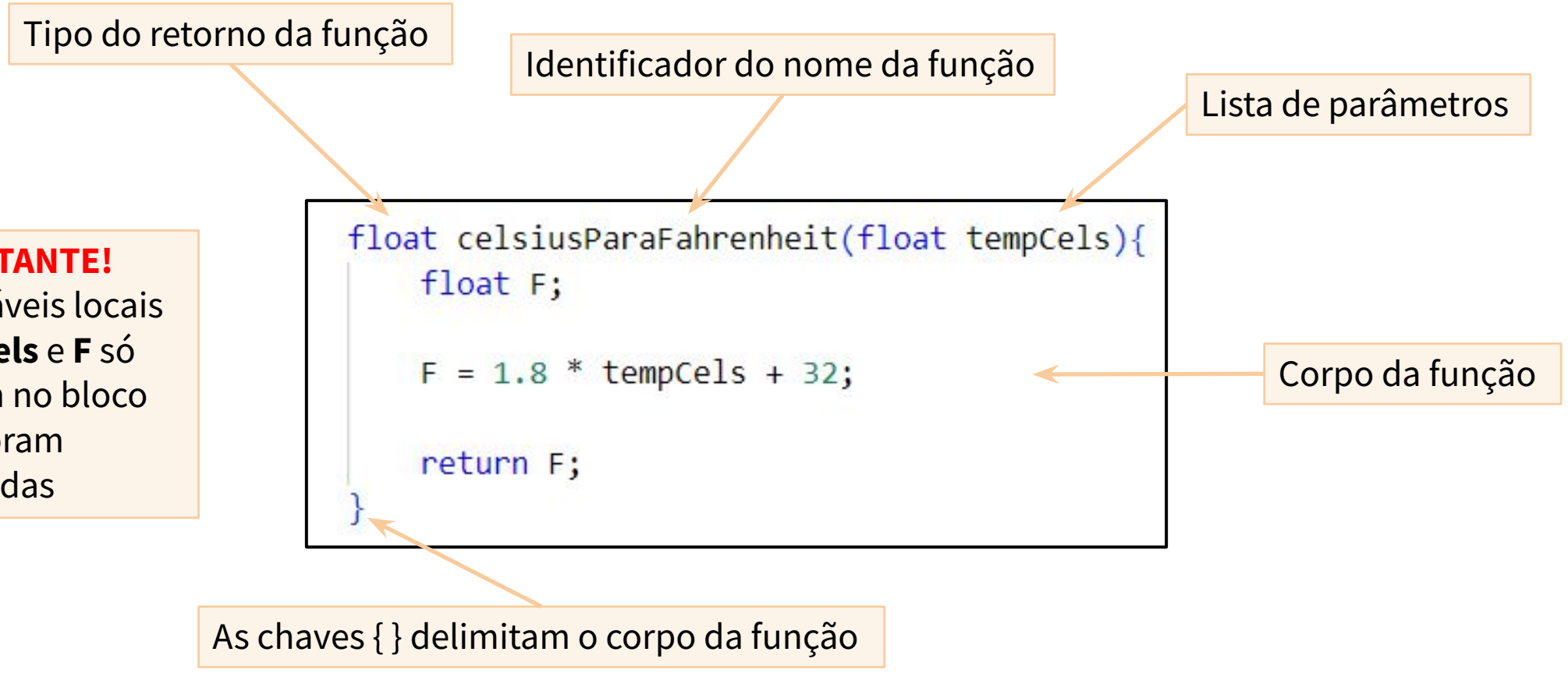
- Onde *C* é a temperatura em Celsius e *F* é a temperatura correspondente em Fahrenheit

Exemplo de conversão de temperaturas

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    float tempC, tempF;

    printf("Conversao Celsius para Fahrenheit\n");
    printf("(valor menor que -273.15 encerra o programa)\n\n");
    printf("Temperatura em Celsius: ");
    scanf("%f", &tempC);

    if (tempC >= -273.15) {
        tempF = 1.8 * tempC + 32;
        printf("%.2f graus Celsius = %.2f graus Fahrenheit.\n", tempC, tempF);
    }

    system("PAUSE");
    return 0;
}
```

Conversão de temperaturas usando função

- Definição da função celsiusParaFahrenheit()



IMPORTANTE!
As variáveis locais **tempCels** e **F** só existem no bloco onde foram declaradas

Conversão de temperaturas usando função

```c
#include <stdio.h>
#include <stdlib.h>

// protótipo da função
float celsiusParaFahrenheit(float tempCels);

int main(){
    float tempC, tempF;

    printf("Conversao Celsius para Fahrenheit\n");
    printf("(valor menor que -273.15 encerra o programa)\n\n");
    printf("Temperatura em Celsius: ");
    scanf("%f", &tempC);

    if (tempC >= -273.15) {
        tempF = celsiusParaFahrenheit(tempC);
        printf("%.2f graus Celsius = %.2f graus Fahrenheit.\n", tempC, tempF);
    }

    system("PAUSE");
    return 0;
}
```

Conversão de temperaturas usando função

```
24  float celsiusParaFahrenheit(float tempCels){
25      float F;
26
27      F = 1.8 * tempCels + 32;
28
29      return F;
30  }
```

ou

```
24  float celsiusParaFahrenheit(float tempCels){
25      return 1.8 * tempCels + 32;
26  }
```

Conversão de temperaturas usando função

- **Eliminando o protótipo da função**
  - Se a função for escrita antes da instrução de sua chamada, seu protótipo não será obrigatório

Conversão de temperaturas usando função

- **Eliminando o protótipo da função**

```c
#include <stdio.h>
#include <stdlib.h>

// definição da função
float celsiusParaFahrenheit(float tempCels){
    return 1.8 * tempCels + 32;
}

int main(){
    float tempC, tempF;

    printf("Conversao Celsius para Fahrenheit\n");
    printf("(valor menor que -273.15 encerra o programa)\n\n");
    printf("Temperatura em Celsius: ");
    scanf("%f", &tempC);

    if (tempC >= -273.15) {
        tempF = celsiusParaFahrenheit(tempC);
        printf("%.2f graus Celsius = %.2f graus Fahrenheit.\n", tempC, tempF);
    }

    system("PAUSE");
    return 0;
}
```

Exemplo: função que retorna valor

- Função que recebe dois valores e retorna o maior valor

```
4   // protótipo da função maior entre 2 números
5   int maior(int a, int b);
```

Exemplo: função que retorna valor

- Função que recebe dois valores e retorna o maior valor

```
4   // definição da função maior entre 2 números
5   int maior2(int a, int b){
6       int maior;
7
8       if (a > b)
9           maior = a;
10      else
11          maior = b;
12
13      return maior;
14  }
```

```
int main(){
    int n1, n2, maior;

    printf("Digite o primeiro numero: ");
    scanf("%d", &n1);
    printf("Digite o segundo numero: ");
    scanf("%d", &n2);

    maior = maior2(n1, n2);
    printf("O maior numero eh: %d\n", maior);

    system("PAUSE");
    return 0;
}
```

Exemplo: função que não retorna valor

- Função que imprime um número como moeda

```
4   // protótipo da função que imprime um número como moeda
5   void printComoMoeda(float m);
```

Exemplo: função que não retorna valor

- Função que imprime um número como moeda

```
7   // definição da função
8   void printComoMoeda(float m){
9       printf("R$ %.2f", m);
10  }
```

```
12  int main(){
13      float moeda = 2;
14
15      printComoMoeda(moeda);
16
17      system("PAUSE");
18      return 0;
19  }
```

Exemplo: função sem parâmetro e retorno

- Função que não possui parâmetro e não retornar nenhum valor

```
4   // protótipo da função que toca um beep
5   void beep(void);
```

Exemplo: função sem parâmetro e retorno

- Função que não possui parâmetro e não retornar nenhum valor

```
7   // definição da função
8   void beep(){
9       printf("\a");
10  }
```

```
12  int main(){
13      int n;
14
15      printf("Digite um número positivo: ");
16      scanf("%d", n);
17
18      if(n < 0)
19          beep();
20
21      system("PAUSE");
22      return 0;
23  }
```

Exemplo: uso de função em argumentos

- Chamada as funções usadas com argumento de outras funções

```
4   // protótipo da função que soma dois números
5   int soma(int m, int n);
6
7   // protótipo da função que soma o quadrado de dois números
8   int somaQuadrado(int a, int b);
```

Exemplo: uso de função em argumentos

- Chamada as funções usadas com argumento de outras funções

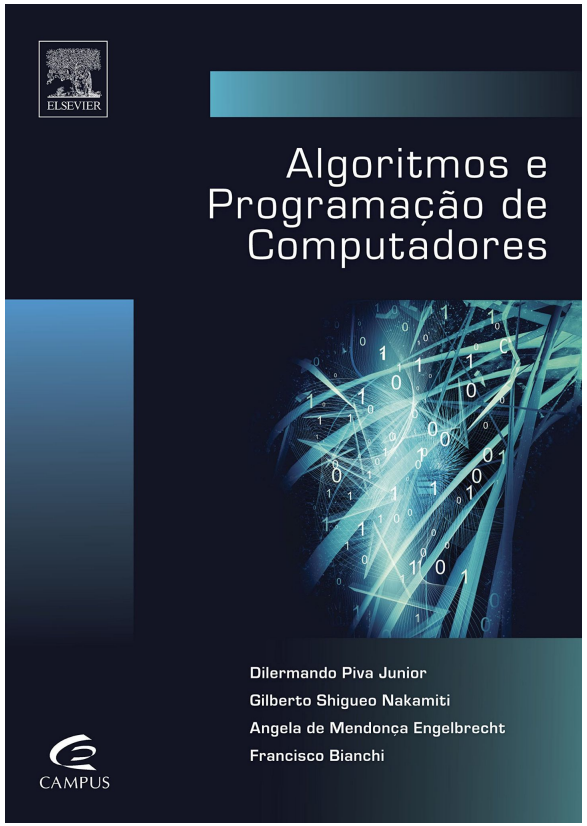
```c
// definição da função soma
int soma(int m, int n){
    return m + n;
}

// definição da função somaQuadrado
int somaQuadrado(int a, int b){
    return soma(pow(a, 2), pow(b, 2));
}
```

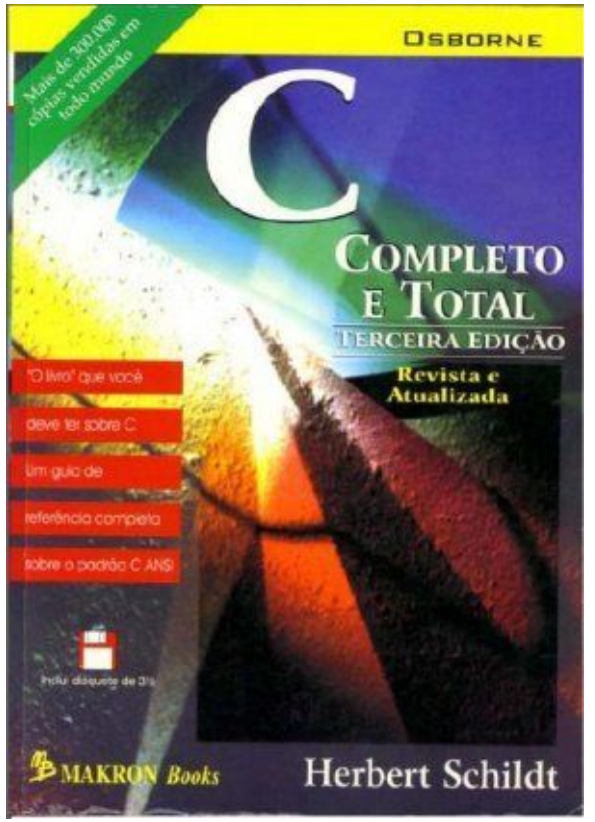
```c
int main(){
    int n1 = 2, n2 = 3, res;
    res = somaQuadrado(n1, n2);

    printf("Resultado: %d\n", res);

    system("PAUSE");
    return 0;
}
```

Resumindo..

- Definição de funções
- Principais objetivos de uma função
- Estrutura de uma função
- Escopo de variáveis
- Exemplos

Referências





PIVA, D. J. et al. **Algoritmos e programação de computadores**. Rio de Janeiro, RJ: Elsevier, 2012.

SCHILDT, Herbert. **C completo e total**. Makron, 1997.