Aula 08: Funções (Parte II)

Introdução a Programação

Túlio Toffolo & Puca Huachi
http://www.toffolo.com.br

BCC201 – 2019/2
Departamento de Computação – UFOP

Aula Anterior

- Introdução a Funções
- Aula prática

Aula de Hoje

1. Revisão de função
2. Exercícios da aula prática
3. Passagem de parâmetro
4. Introdução a ponteiros

Aula de Hoje

1. Revisão de função
2. Exercícios da aula prática
3. Passagem de parâmetro
4. Introdução a ponteiros

Protótipo

Definição Geral de uma Função

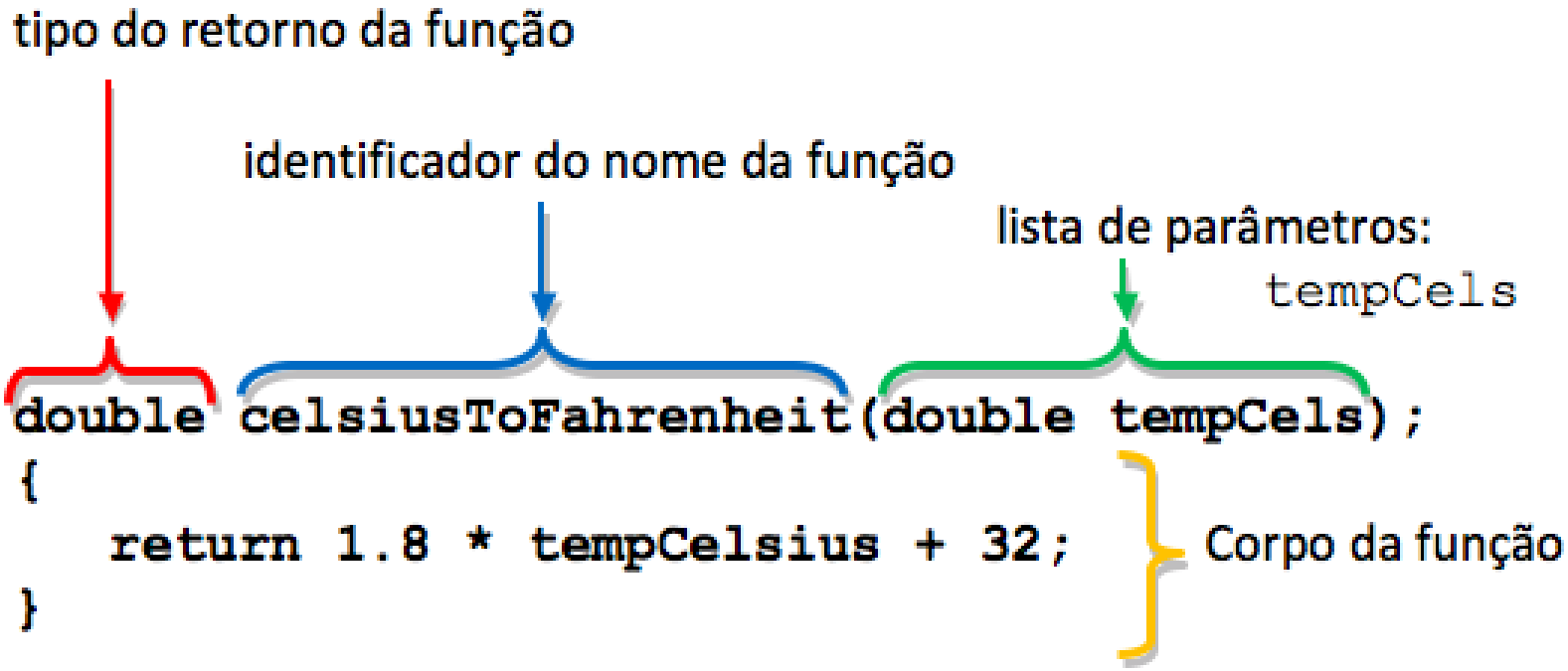
```
1 <tipo_retorno> <nome_função>(<lista_declaração_parâmetro>)
2 {
3     <corpo_função>
4 }
```

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, possivelmente vazia, de declarações separadas por vírgulas, dos parâmetros da função.
- `<corpo_função>`: descreve o comportamento da função.

Definição de funções

Exemplo: Definição da função `celsiusToFahrenheit()`

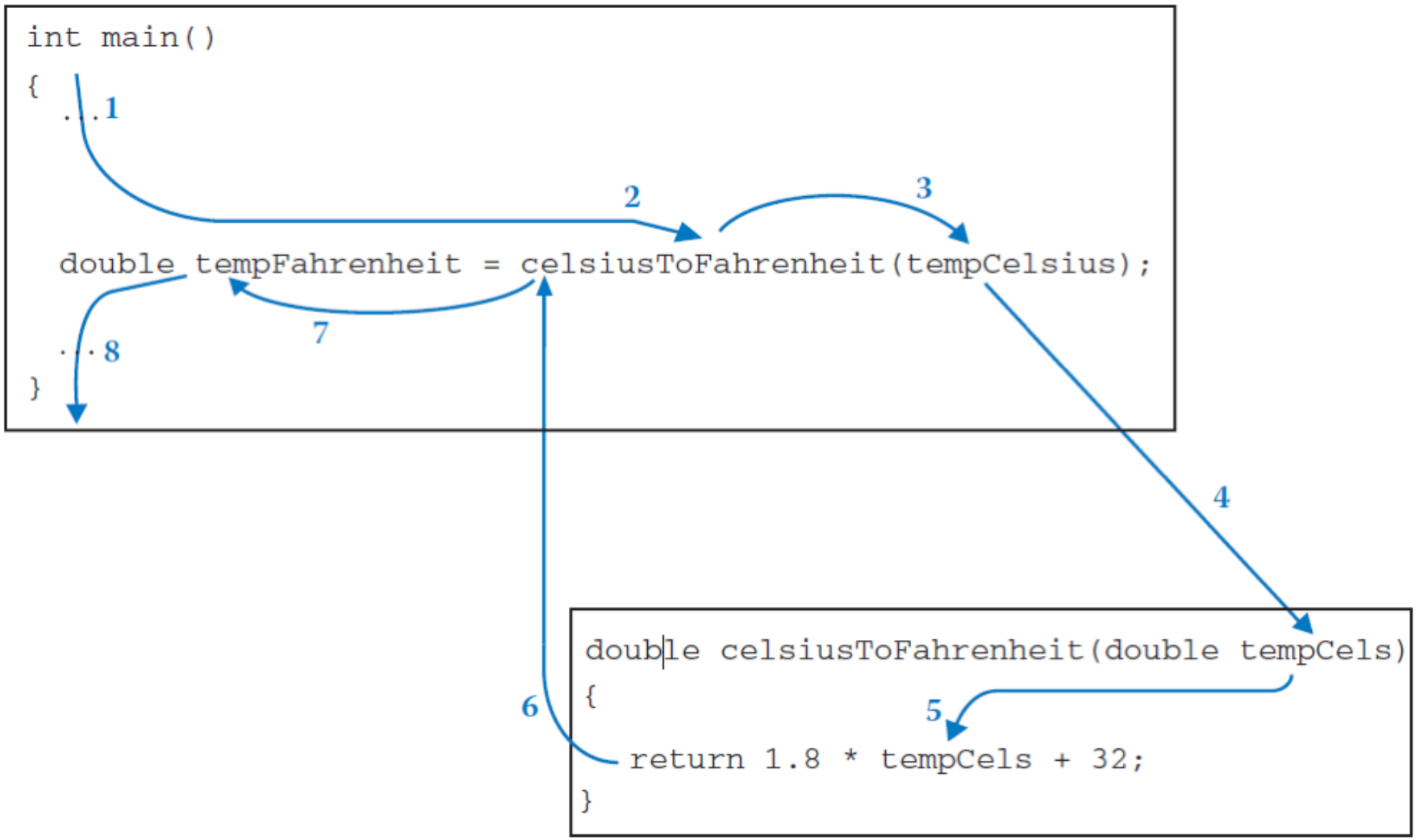


Escopo de variáveis

As variáveis só existem no bloco onde foram declaradas.

Obs. Mesmo que as variáveis possuam o mesmo nome na `main()` e em uma função `funcao()`, o compilador enxerga como variáveis distintas.

Fluxo de execução



Aula de Hoje

1. Revisão de função
2. Exercícios da aula prática
3. Passagem de parâmetro
4. Introdução a ponteiros

Exercícios da aula prática

Questão 01

Crie um programa para converter valores em diferentes unidades de **ângulo** e **temperatura**. Ao ser executado, seu programa deve imprimir um menu para escolha entre ângulo e temperatura.

Ao optar por uma das opções, outro menu é exibido, desta vez perguntando qual a unidade do valor de origem. Em seguida o programa deve solicitar o valor e imprimir os valores convertidos.

Utilize o comando `switch` para implementar os menus e crie **funções** para coverter os valores. Note que seu programa terá pelo menos oito funções.

Exercícios da aula prática

Questão 01

Exemplos de execução:

```
### CONVERSOR DE UNIDADES ###

1) Angulo
2) Temperatura

Digite uma opcao: 1

Qual a unidade de origem?

1) Graus
2) Radianos

Selecione uma opcao: 1

Digite o valor em Graus: 180
Valor em radianos: 3.141593
```

```
### CONVERSOR DE UNIDADES ###

1) Angulo
2) Temperatura

Digite uma opcao: 2

Qual a unidade de origem?

1) Celsius
2) Fahrenheit
3) Kelvin

Selecione uma opcao: 1

Digite o valor em Celsius: 100
Valor em Fahrenheit: 212.00
Valor em Kelvin: 373.15
```

Exercícios da aula prática

Conversão de graus Celsius para Fahrenheit e vice-versa:

```
1  double celsiusToFahrenheit(double celsius) {
2      return (celsius * 9.0/5.0) + 32;
3  }
4
5  double fahrenheitToCelsius(double fahrenheit) {
6      return (fahrenheit - 32) * 5.0/9.0;
7  }
```

Conversão de graus Celsius para Kelvin e vice-versa:

```
1  double celsiusToKelvin(double celsius) {
2      return celsius + 273.15;
3  }
4
5  double kelvinToCelsius(double kelvin) {
6      return kelvin - 273.15;
7  }
```

Exercícios da aula prática

Conversão de Fahrenheit para Kelvin e vice-versa:

```
1  // podemos usar funções definidas anteriormente!
2  double fahrenheitToKelvin(double fahrenheit) {
3      return celsiusToKelvin(fahrenheitToCelsius(fahrenheit));
4  }
5
6  double kelvinToFahrenheit(double kelvin) {
7      return celsiusToFahrenheit(kelvinToCelsius(kelvin));
8  }
```

Conversão de graus para radianos e vice-versa:

```
1  double degreeToRadian(double degree) {
2      const double PI = 3.141593;
3      return degree * PI/180.0;
4  }
5
6  double radianToDegree(double rad) {
7      const double PI = 3.141593;
8      return rad * 180.0/PI;
9  }
```

Aula de Hoje

1. Revisão de função
2. Exercícios da aula prática
3. Passagem de parâmetro
4. Introdução a ponteiros

Passagem de Parâmetros

Os parâmetros formais (variáveis locais, declaradas como parâmetro da função chamada) são inicializados com o valor dos parâmetros reais (variáveis passadas como parâmetro).

- **Passagem por valor** – O valor dos parâmetros formais, **se** alterados durante a execução da função **não** acarretarão em nenhuma modificação no valor dos parâmetros reais (variáveis da função chamadora).

Observação: Todos os exemplos mostrados até o momento utilizam passagem de parâmetro **por valor**.

Exemplo

Fazer uma função em C para trocar dois números.

- A função recebe dois valores e retorna esses valores trocados.
- Problema: Como retornar dois valores?

Passagem de Parâmetro por Valor

Declaração da função:

```
1  void troca1 (int a, int b)
2  {
3     int temp = a;
4     a = b;
5     b = temp;
6  }
```

Chamada da função:

```
1  ...
2  c = 4; d = 5;
3  printf("c = %d, d = %d\n", c, d);
4  troca1(c, d);
5  printf("c = %d, d = %d\n", c, d);
6  ...
```

Passagem de parâmetro

Saída do programa:

```
1 ...
2 c = 4, d = 5
3 c = 4, d = 5
4 ...
```

- O programa cria uma cópia das variáveis c e d para as variáveis a e b, respectivamente.
- As variáveis possuem **escopo diferentes** e são independentes. Os valores de c e d permanecem os mesmos, pois nada foi passado de volta para a unidade chamadora.

Como alterar o valor da variável dentro da função

Nós já utilizamos uma função que faz isso...

```c
int main()
{
    int x;
    scanf("%d", &x); // passamos o endereço de memória de x: &x

    if (x % 2 == 0)
        printf("%d é um número par!\n", x);
    else
        printf("%d é um número ímpar!\n", x);
}
```

Como alterar o valor da variável dentro da função

Como receber um endereço de memória na função?

- A chamada abaixo vai funcionar?

```
1 troca1(&c, &d); // agora passando endereços de memória
```

- **Não**. Pois a função recebe dois valores do tipo `int`, não dois endereços de memória.
- Para receber um endereço de memória, temos que usar um **ponteiro**!

```
1 int *ponteiro; // essa variável armazena um endereço de memória!
```

Passagem de parâmetro

Declaração da função:

```
1  void troca2 (int *a, int *b)
2  {
3      int temp = *a;
4      *a = *b;
5      *b = temp;
6  }
```

Chamada da função:

```
1  ...
2  c = 4; d = 5;
3  printf("c = %d, d = %d\n", c, d);
4  troca2(&c, &d);
5  printf("c = %d, d = %d\n", c, d);
6  ...
```

Passagem de parâmetro por referência

Saída do programa:

```
1 ...
2 4, 5
3 5, 4
4 ...
```

- A chamada de `troca2()` recebe **ponteiros** para os parâmetros:
  - `a = &c`, *i.e.*, ponteiro `a` aponta para o endereço de `c`
  - `b = &d`, *i.e.*, ponteiro `b` aponta para o endereço de `d`
- Assim, os valores de c e d são de fato trocados (mesmo as variáveis tendo escopo diferentes), pois foi passado para `troca2()` o endereço desses parâmetros.

Aula de Hoje

1. Revisão de função
2. Exercícios da aula prática
3. Passagem de parâmetro
4. Introdução a ponteiros

Ponteiros

Um ponteiro (apontador ou *pointer*) é um tipo especial de variável que armazena um **endereço de memória**

- Ponteiros são declarados utilizando o caractere especial `*`:

```
1  int *pi;     // pi é um ponteiro do tipo int
2  char *pc;    // pc é um ponteiro do tipo char
3  float *pf;   // pf é um ponteiro do tipo float
4  double *pd; // pd é um ponteiro do tipo double
```

- Vários podem ser declarados em uma única linha:

```
1  int *p1, *p2, *p3;
```

Ponteiros

O **conteúdo** da memória apontada por um ponteiro se refere ao valor armazenado no endereço de memória para o qual o ponteiro aponta.

- Este conteúdo (valor) pode ser alterado usando o operador *
- Exemplo:

```
1  int main()
2  {
3      int x = 10, y = 0;
4      int *px = &x;
5   → y = *px; // y recebe o conteúdo do endereço apontado por px
6      printf("y = %d\n", y);
7      return 0;
8  }
```

- O que será impresso?

```
1  y = 10
```

Ponteiros

Exemplo:

```c
int main()
{
    int x = 0;
    int *px;
    px = &x;
    *px = 99;
    printf("x = %d\n", x);
    return 0;
}
```

O que será impresso?

```
x = 99
```

Ponteiros

Exemplo:

```c
int main()
{
    int x = 100;
    int *px = &x;
    printf("valor de x    = %d\n", x);
    printf("endereço de x = %p\n", &x); // %p: formato para ponteiro
    printf("endereço de x = %p\n", px); // %p: formato para ponteiro
    printf("valor de x    = %d\n", *px);
    return 0;
}
```

Exemplo de saída (computador com 64 bits):

```
valor de x    = 100
endereço de x = 0x7ffedfc1e378
endereço de x = 0x7ffedfc1e378
valor de x    = 100
```

Tipos de ponteiros

Há vários tipos de ponteiros:

- Ponteiros para caracteres
- Ponteiros para inteiros
- Ponteiros para vetores
- Ponteiros para ponteiros para inteiros
- etc...

Você especifica o tipo de ponteiro!

Perguntas?