

Arquitetura de Computadores, Algoritmos, Lógica de Programação e Circuitos Digitais

Danilo Rios
10/10/2025

Função

- Até então a função recebe parâmetros e passamos valores/variáveis para cada parâmetro
 - Isso chamamos de passar por valor
 - Ou seja, passamos uma cópia do valor fixo ou do valor da variável para a função
 - A variável que é enviada por parâmetro não é afetada

```
1  #include <stdio.h>
2
3  int retornaMais2(int valor) {
4      valor = valor + 2;
5      return valor;
6  }
7
8  int main() {
9
10     int valor = 10;
11     int valorRetornado = 0;
12
13     printf("valor = %d - valorRetornado = %d\n", valor, valorRetornado);
14
15     valorRetornado = retornaMais2(valor);
16
17     printf("valor = %d - valorRetornado = %d", valor, valorRetornado);
18
19     return 0;
20 }
```

```
1  #include <stdio.h>
2
3  int retornaMais2(int valor) {
4      valor = valor + 2;
5      return valor;
6  }
7
8  int main() {
9      int valor = 10;
10     int valorRetornado = 0;
11     printf("valor = %d - valorRetornado = %d", valor, valorRetornado);
12     printf("-----\n");
13     printf("Process exited after 0.191 seconds with return value 0\n");
14     printf("Press any key to continue . . .\n");
15     valorRetornado = retornaMais2(valor);
16
17     printf("valor = %d - valorRetornado = %d", valor, valorRetornado);
18
19     return 0;
20 }
```

Função

- Outro modo é passar o parâmetro por referência
- O que significa isso?

Função

- Outro modo é passar o parâmetro por referência
- O que significa isso?
 - Passar o endereço de memória para a função
 - Então alterações no valor da variável vão refletir fora da função

```
1  #include <stdio.h>
2
3  int retornaMais2(int *valor) {
4      *valor = *valor + 2;
5      return *valor;
6  }
7
8  int main() {
9
10     int valor = 10;
11     int valorRetornado = 0;
12
13     printf("valor = %d - valorRetornado = %d\n", valor, valorRetornado);
14     printf("valor = %p - valorRetornado = %p\n", &valor, &valorRetornado);
15
16     valorRetornado = retornaMais2(&valor);
17
18     printf("valor = %d - valorRetornado = %d\n", valor, valorRetornado);
19     printf("valor = %p - valorRetornado = %p\n", &valor, &valorRetornado);
20
21     return 0;
22 }
```

```
1  #include <stdio.h>
2
3  int retornaMais2(int *valor) {
4      *valor = *valor + 2;
5      return *valor;
6  }
7
8  int main()
9  {
10     int valor = 10 - valorRetornado = 0;
11     int valorRetornado = 0;
12     valor = 12 - valorRetornado = 12;
13     valorRetornado = 0;
14     printf("valor = %d - valorRetornado = %d\n", valor, valorRetornado);
15     printf("valor = %p - valorRetornado = %p\n", &valor, &valorRetornado);
16     valorRetornado = retornaMais2(&valor);
17
18     printf("valor = %d - valorRetornado = %d\n", valor, valorRetornado);
19     printf("valor = %p - valorRetornado = %p\n", &valor, &valorRetornado);
20
21     return 0;
22 }
```

C:\Users\Danilo\Desktop\Arq x + v

valor = 10 - valorRetornado = 0
valor = 000000000062FE4C - valorRetornado = 000000000062FE48
valor = 12 - valorRetornado = 12
valor = 000000000062FE4C - valorRetornado = 000000000062FE48

Process exited after 0.5067 seconds with return value 0
Press any key to continue . . .

Função

- No lugar de passar o endereço da variável, pode ser um ponteiro
 - É a mesma coisa

```
1  #include <stdio.h>
2
3  int retornaMais2(int *valor) {
4      *valor = *valor + 2;
5      return *valor;
6  }
7
8  int main() {
9
10     int valor = 10;
11     int *ponteiroValor = &valor;
12     int valorRetornado = 0;
13
14     printf("valor = %d - valorRetornado = %d\n", valor, valorRetornado);
15     printf("valor = %p - valorRetornado = %p\n", &valor, &valorRetornado);
16
17     valorRetornado = retornaMais2(ponteiroValor);
18
19     printf("valor = %d - valorRetornado = %d\n", valor, valorRetornado);
20     printf("valor = %p - valorRetornado = %p\n", &valor, &valorRetornado);
21
22     return 0;
23 }
```

```
1 #include <stdio.h>
2
3 int retornaMais2(int *valor) {
4     *valor = *valor + 2;
5     return *valor;
6 }
7
8 int main()
9 {
10     int valor = 10 - valorRetornado = 0
11     int *p_valor = 000000000062FE44 - valorRetornado = 000000000062FE40
12     int valorRetornado = 12 - valorRetornado = 12
13     valorRetornado = 000000000062FE44 - valorRetornado = 000000000062FE40
14     printf("valorRetornado = %d\n", valorRetornado);
15     printf("Process exited after 0.5128 seconds with return value 0\n", valorRetornado);
16     printf("Press any key to continue . . .\n");
17     valorRetornado = retornaMais2(p_valor);
18
19     printf("valor = %d - valorRetornado = %d\n", valor, valorRetornado);
20     printf("valor = %p - valorRetornado = %p\n", &valor, &valorRetornado);
21
22     return 0;
23 }
```

Função

- Como que funciona para passar um array por parâmetro? É possível?

Função

- Como que funciona para passar um array por parâmetro? É possível?
 - Sim, é possível
 - Não é por passagem por valor:
 - Cópia da informação
 - int, float, double, char... ok copiar
 - Um array unidimensional de double com 1000 posições
 - 8000 Bytes é muita memória para duplicar
 - Passagem por referência:
 - Array é passado por referência e então sempre a alteração reflete fora da função

Função

- Atenção!
- Passar parâmetro por referência significa passar o endereço de memória
- Ao passar o endereço de memória do array, não existe mais a informação do tamanho do array e deve ser passada essa segunda informação por valor

Função

```
1  #include <stdio.h>
2
3  void imprimirArray(int *array, int tamanho) {
4      int i;
5      for(i=0;i<tamanho;i++) {
6          printf("%d\n", *(array + i));
7      }
8  }
9
10 int main() {
11
12     int array[] = {0,1,2,3,4,5};
13     int tamanho = 6;
14
15     printf("Passando array:\n");
16     imprimirArray(array, tamanho);
17
18     printf("Passando &array:\n");
19     imprimirArray(&array, tamanho);
20
21     return 0;
22 }
```

Função

```
1 #include <stdio.h>
```

```
2
```

```
3 void imprimeArray(int *array, int tamanho) {
```

C:\Users\Danilo\Desktop\Arq X

+

▼

Passando array:

0

1

2

3

4

5

Passando &array:

0

1

2

3

4

5

Process exited after 0.1879 seconds with return value 0

Press any key to continue . . .

```
21     return 0;
```

```
22 }
```


Função

```
1  #include <stdio.h>
2
3  void imprimirArray(int array[], int tamanho) {
4      int i;
5      for(i=0;i<tamanho;i++) {
6          printf("%d\n", array[i]);
7      }
8  }
9
10 int main() {
11
12     int array[] = {0,1,2,3,4,5};
13     int tamanho = 6;
14
15     printf("Passando array:\n");
16     imprimirArray(array, tamanho);
17
18     printf("Passando &array:\n");
19     imprimirArray(&array, tamanho);
20
21     return 0;
22 }
```

Função

```
1  #include <stdio.h>
2
3  ... {
4
5  Passando array:
6  0
7  1
8  2
9  3
10 4
11 5
12
13 Passando &array:
14 0
15 1
16 2
17 3
18 4
19 5
20
21 -----
22 Process exited after 0.19 seconds with return value 0
23 Press any key to continue . . .
24
25 return 0;
26 }
```

Função

- No final do semestre passado falamos da função main “completa” que recebe parâmetros de entrada
- Hoje entendemos como funciona

```
1  #include <stdio.h>
2
3  int main(int argc, char *argv[]) {
4      //argc = quantidade de argumentos
5      //argv = valores recebidos
6
7      return 0;
8  }
```

Função

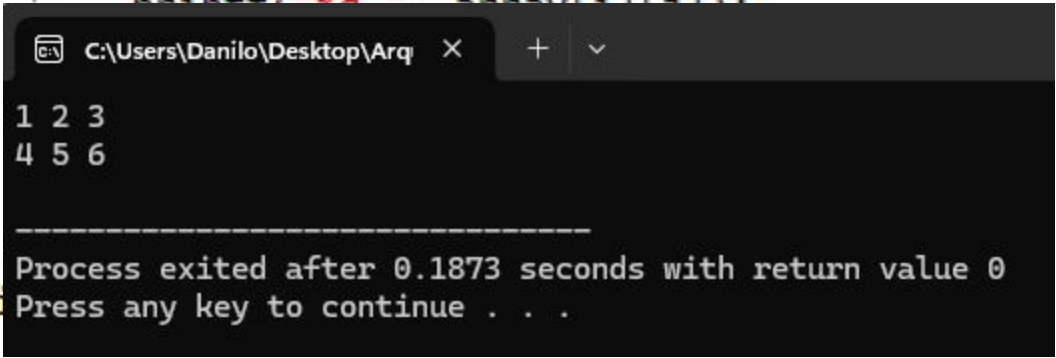
- Como deve ser o código?
 - Declara e preenche um array bidimensional 2x3 com as informações
 - 1, 2, 3
 - 4, 5, 6
 - Chama a função imprimirArray passando o array bidimensional por parâmetro
 - A função imprimirArray deve imprimir cada elemento da linha separado por espaço e pular de linha

Fu

```
1  #include <stdio.h>
2
3  void imprimirArray(int linha, int coluna, int array[][coluna]) {
4      int i, j;
5      for(i=0;i<linha;i++) {
6          for(j=0;j<coluna;j++) {
7              printf("%d ", array[i][j]);
8          }
9          printf("\n");
10     }
11 }
12
13 int main() {
14     int array[2][3] = {{1,2,3},{4,5,6}};
15
16     imprimirArray(2, 3, array);
17
18     return 0;
19 }
20
```

Fu

```
1  #include <stdio.h>
2
3  void imprimirArray(int linha, int coluna, int array[][coluna]) {
4      int i, j;
5      for(i=0;i<linha;i++) {
6          for(j=0;j<coluna;j++) {
7              printf("%d ", array[i][j]);
8          }
9      }
10 }
11
12
13 int main()
14 {
15     int array[2][3] = {{1,2,3},{4,5,6}};
16
17     imprimirArray(2, 3, array);
18
19     return 0;
20 }
```



C:\Users\Danilo\Desktop\Arq x + v

```
1 2 3
4 5 6

-----
Process exited after 0.1873 seconds with return value 0
Press any key to continue . . .
```

Perguntas?

Exercício 8

- **Obs.: Não será aceito exercício com loop infinito ou chamar a main()**
- Criar um programa que:
 - Cria um array unidimensional com 5 posições
 - Pedir para a pessoa digitar 5 números e guardar no array unidimensional
 - Criar um array bidimensional 5x5
 - Automaticamente atribuir valores para cada posição
 - O valor deve ser o valor da coluna no array unidimensional multiplicado pela linha (1 até 5)
 - Ex.: Array unidimensional: 1, 2, 3, 4, 5
Ex.: Array bidimensional: 1, 2, 3, 4, 5
2, 4, 6, 8, 10
3, 6, 9, 12, 15
4, 8, 12, 16, 20
5, 10, 15, 20, 25
 - Chamar a função `imprimirInformacoes` que recebe como parâmetros o array unidimensional e o array bidimensional e imprime conforme o exemplo anterior
- Entrega até: 24/10

Obrigado!

Até a próxima aula!