

SSC0501 - Introdução à Ciência de Computação I

Bem Vindos!

Funções com Ponteiros!

Relembrando Funções...

Funções

O que é um função?

```
int matrix[3][3]:
```



5	9	3
2	6	8
4	0	7

9	2	3
7	8	5
2	2	8

2	2	0	8	6	1
+	9	6	5	3	0
1	2	8	5	2	7

Funções

O que é um função?

Uma **função** é um bloco de código que realiza **uma tarefa específica** e pode ser **reutilizado** ao longo do programa.

```
int matrix[3][3]:
```

5	9	3
2	6	8
4	0	7

9	2	3
7	8	5
2	2	8

2	2	0	8	6	1
4	9	6	5	3	0
1	2	8	5	2	0

Funções

O que é um função?

Uma **função** é um bloco de código que realiza **uma tarefa específica** e pode ser **reutilizado** ao longo do programa.

Como é a sintaxe de declaração de uma função?

Funções

O que é um função?

Uma **função** é um bloco de código que realiza **uma tarefa específica** e pode ser **reutilizado** ao longo do programa.

Como é a sintaxe de declaração de uma função?

```
tipo nome_da_funcao(tipo_param1 nome_param1, tipo_param2 nome_param2)
{
    // corpo da função
    return valor; // (se o tipo for diferente de void)
}
```

Funções

Exemplo:

```
int soma(int a, int b)
{
    int resultado = a + b;
    return resultado;
}
```

Exemplo de chamada:

```
int resultado = soma(3, 4); // resultado recebe 7
```


Funções

Lembre-se que podemos **pré-declarar** uma função, assim o compilador “entende” que a função existe e será utilizada ao longo do código. Assim podemos definir a função mais ao final do código sem problemas:

```
int soma(int a, int b);
```

Exemplo:

```
int soma(int a, int b)
{
    int resultado = a + b;
    return resultado;
}
```

Exemplo de chamada:

```
int resultado = soma(3, 4); // resultado recebe 7
```

Funções

Lembre-se que podemos **pré-declarar** uma função, assim o compilador “entende” que a função existe e será utilizada ao longo do código. Assim podemos definir a função mais ao final do código sem problemas:

```
int soma(int a, int b);
```

Exemplo:

```
int soma(int a, int b)
{
    int resultado = a + b;
    return resultado;
}
```

A pré declaração utiliza a mesma assinatura da função, porém insere-se um “;” ao final.

Exemplo de chamada:

```
int resultado = soma(3, 4); // resultado recebe 7
```

Relembrando Ponteiros...

Ponteiros em C

O que é um ponteiro?

```
int matrix[3][3]:
```



5	9	3
2	6	8
4	0	7

9	2	3
7	8	5
2	2	8

2	2	0	8	6	1
+	9	6	5	3	0
1	2	8	5	2	

Ponteiros em C

O que é um ponteiro?

- Um **ponteiro** é uma variável que **guarda o endereço** de outra variável.

```
int matrix[3][3]:
```

5	9	3
2	6	8
4	0	7

9	2	3
7	8	5
2	2	8

2	2	0	8	6	1
4	9	6	5	3	0
1	2	8	5	2	

Ponteiros em C

O que é um ponteiro?

- Um **ponteiro** é uma variável que **guarda o endereço** de outra variável.

Qual a Sintaxe de declaração?

Ponteiros em C

O que é um ponteiro?

- Um **ponteiro** é uma variável que **guarda o endereço** de outra variável.

Qual a Sintaxe de declaração?

- tipo *nome_ponteiro;

Ponteiros em C

O que é um ponteiro?

- Um **ponteiro** é uma variável que **guarda o endereço** de outra variável.

Qual a Sintaxe de declaração?

- tipo *nome_ponteiro;

Quais são os operadores?

Ponteiros em C

O que é um ponteiro?

- Um **ponteiro** é uma variável que **guarda o endereço** de outra variável.

Qual a Sintaxe de declaração?

- tipo *nome_ponteiro;

Quais são os operadores?

- O operador & obtém o **endereço** de uma variável.
- O operador * acessa o **conteúdo** do endereço apontado.

Ponteiros em C

Exemplo de utilização:

```
int x = 10;
int *p = &x;    // p aponta para x

printf("%p\n", p);    // imprime endereço de x
printf("%d\n", *p);   // imprime conteúdo de x (10)
```


Ponteiros em C

Porque utilizar ponteiros?

```
int matrix[3][3]:
```



5	9	3
2	6	8
4	0	7

5	9	3
2	6	8
4	0	7

9	2	3
7	8	5
2	2	8

2	2	0	8	6	1
4	9	6	5	3	0
1	2	8	5	2	

Ponteiros em C

Porque utilizar ponteiros?

- Permite **alterar variáveis fora da função (passagem por referência!)**.
- Útil em **alocação dinâmica**, vetores, strings, structs e **parâmetros por referência**.
- Necessário para trabalhar com **endereços de memória diretamente**.

Funções: Passagem por Valor vs. Referência

Funções: Passagem por Valor vs. Referência

Passagem por Valor:

```
void dobrar(int x) {  
    x = x * 2;  
}  
  
int main() {  
    int a = 10;  
    dobrar(a);  
    printf("%d\n", a); // Qual o valor impresso?  
}
```

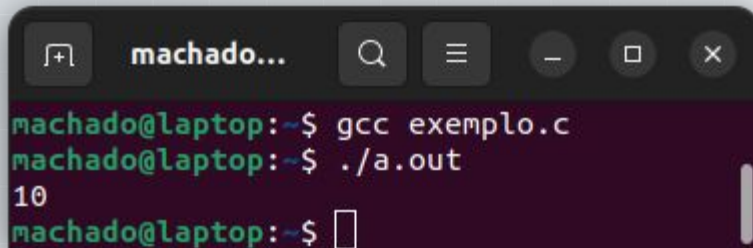
Passagem por Referência:

```
void dobrar(int *x) {  
    *x = (*x) * 2;  
}  
  
int main() {  
    int a = 10;  
    dobrar(&a);  
    printf("%d\n", a); // Qual o valor impresso?  
}
```

Funções: Passagem por Valor vs. Referência

Passagem por Valor:

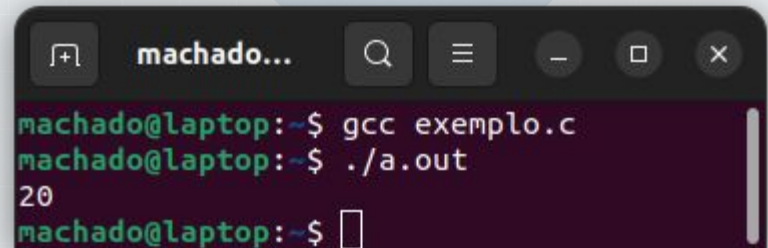
```
void dobrar(int x) {  
    x = x * 2;  
}  
  
int main() {  
    int a = 10;  
    dobrar(a);  
    printf("%d\n", a); // Qual o valor impresso?  
}
```



```
machado...  
machado@laptop:~$ gcc exemplo.c  
machado@laptop:~$ ./a.out  
10  
machado@laptop:~$
```

Passagem por Referência:

```
void dobrar(int *x) {  
    *x = (*x) * 2;  
}  
  
int main() {  
    int a = 10;  
    dobrar(&a);  
    printf("%d\n", a); // Qual o valor impresso?  
}
```



```
machado...  
machado@laptop:~$ gcc exemplo.c  
machado@laptop:~$ ./a.out  
20  
machado@laptop:~$
```


Funções: Passagem por Valor vs. Referência

Passagem por Valor:

```
void dobrar(int x) {  
    x = x * 2;  
}  
  
int main() {  
    int a = 10;  
    dobrar(a);  
    printf("%d\n", a); // Qual o valor impresso?  
}
```

Passagem por Referência:

```
void dobrar(int *x) {  
    *x = (*x) * 2;  
}  
  
int main() {  
    int a = 10;  
    dobrar(&a);  
    printf("%d\n", a); // Qual o valor impresso?  
}
```

Conclusão:

- Com valor, a função altera apenas a cópia.
- Com ponteiro, a função altera a variável original.

Função retornando ponteiro

Por que retornar um ponteiro?

- Quando a função cria ou localiza algo e deseja devolver seu endereço.

- Útil para:

- Retornar elementos de um vetor
- Criar valores dinamicamente (malloc).
- Buscar elementos em estruturas

```
int matrix[3][3]:
```

Função retornando ponteiro

Exemplo:

```
typedef struct {
    char nome[50];
    float nota;
} Aluno;

Aluno* melhorAluno(Aluno turma[], int n) {
    if (n == 0) return NULL;

    Aluno *melhor = &turma[0];
    for (int i = 1; i < n; i++) {
        if (turma[i].nota > melhor->nota)
            melhor = &turma[i];
    }
    return melhor;
}
```

Função retornando ponteiro

Exemplo:

```
int main()
{
    Aluno turma[3] = {
        {"Ana", 8.0},
        {"Bruno", 9.2},
        {"Clara", 7.5}
    };

    Aluno *top = melhorAluno(turma, 3);
    if (top != NULL)
        printf("Melhor aluno: %s (%.1f)\n", top->nome, top->nota);

    return 0;
}
```

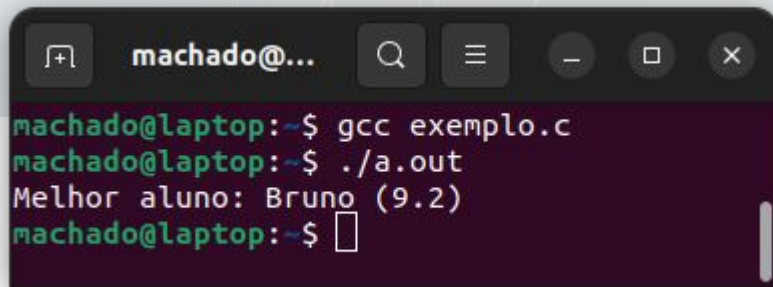
Função retornando ponteiro

Exemplo:

```
int main()
{
    Aluno turma[3] = {
        {"Ana", 8.0},
        {"Bruno", 9.2},
        {"Clara", 7.5}
    };

    Aluno *top = melhorAluno(turma, 3);
    if (top != NULL)
        printf("Melhor aluno: %s (%.1f)\n", top->nome, top->nota);

    return 0;
}
```

A terminal window with a dark background and light-colored text. The window title is "machado@...". The terminal shows the following commands and output: "gcc exemplo.c", "./a.out", "Melhor aluno: Bruno (9.2)", and a prompt character. The window has standard Linux window controls (minimize, maximize, close) and a search icon.

```
machado@laptop:~$ gcc exemplo.c
machado@laptop:~$ ./a.out
Melhor aluno: Bruno (9.2)
machado@laptop:~$
```


Função retornando ponteiro

Vantagens:

- Permite **acessar e modificar diretamente** o elemento encontrado.
- Reutilizável em diversas situações (ex: encontrar menor nota, nota média etc).

Função retornando ponteiro

Lembre-se:

- **Nunca retorne ponteiros para variáveis locais:**

```
int* erro()  
{  
    int x = 10;  
    return &x; // inválido! x será destruído no fim do bloco (escopo)  
}
```

Prática!

Código Matricula: R4SM

<https://runcodes.icmc.usp.br/offerings/view/83>

[run.codes]



Menu Professor ▾

matheus.m.santos@icmc.usp.br ▾

Hora do Servidor: 14/03/2023 18:46:06

Home > SSC0501

SSC0501 - Introdução à Ciência de Computação I

Professores/Monitores

Professores: Matheus Machado dos Santos
Turma: 2025101
Universidade: USP
Ativa até: 21/07/2025



Código de Matrícula

R4SM



Novo Exercício



Enviar E-mail



Ver Notas



Exportar Tabela de Notas

Exercícios

No.	Exercício	Status	Casos Corretos	Nota	Entregas	Participantes	Prazo de Entrega	Ações
1	Hello World	Finalizado	1/1	10.00	78	43/46	14/03/2025 21:00:00	Ver Detalhes Remover Exercício
2	1.01 Maior número	Não Entregue	0/6	0	0	0/46	19/03/2025 23:59:59	Ver Detalhes Remover Exercício
3	1.02 Par ou ímpar	Não Entregue	0/7	0	0	0/46	19/03/2025 23:59:59	Ver Detalhes Remover Exercício
4	1.03 Positivo, negativo ou zero	Não Entregue	0/7	0	0	0/46	19/03/2025 23:59:59	Ver Detalhes Remover Exercício
5	1.04 Maior de três números	Não Entregue	0/7	0	0	0/46	19/03/2025 23:59:59	Ver Detalhes Remover Exercício

SSC0501 - Introdução à Ciência de Computação I

Obrigado pela atenção!!