

# SSC0501 - Introdução à Ciência de Computação I

Bem Vindos!

```
int main[2][3]:
```

# Macros



# O que são Macros?

**Macros** em C são instruções de **substituição textual** realizadas pelo pré-processador **antes da compilação**.

- São definidas com a diretiva **#define**:

```
#define PI 3.14159  
#define TAMANHO_MAXIMO 100
```

- Não ocupam memória, são substituições diretas no código-fonte.

# Para que servem Macros?

- Definir constantes:

```
#define VELOCIDADE_LUZ 299792458
```

- Criar expressões reutilizáveis:

```
#define QUADRADO(x) ((x) * (x))
```

- Facilitar a leitura e manutenção do código
- Evitar uso de constantes literais no meio do código (números mágicos)
- Escrever **código portátil entre diferentes plataformas**



# Exemplo Prático

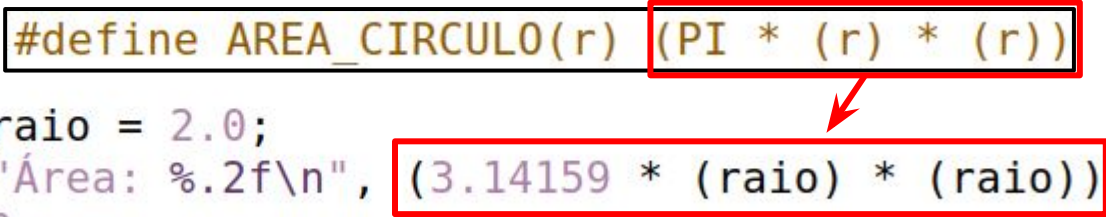
```
1#include <stdio.h>
2
3#define PI 3.14159
4#define AREA_CIRCULO(r) (PI * (r) * (r))
5
6int main()
7{
8    double raio = 2.0;
9    printf("Área: %.2f\n", AREA_CIRCULO(raio));
10    return 0;
11}
```

# Exemplo Prático

```
1 <códigos que estavam no stdio.h>
2
3
4
5
6 int main()
7 {
8     double raio = 2.0;
9     printf("Área: %.2f\n", (3.14159 * (raio) * (raio)));
10    return 0;
11 }
```

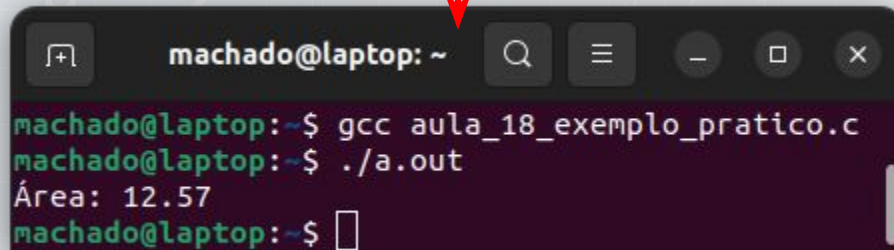
# Exemplo Prático

```
1 <códigos que estavam no stdio.h>
2
3
4
5
6 int main()  #define AREA_CIRCULO(r) (PI * (r) * (r))
7 {
8     double raio = 2.0;
9     printf("Área: %.2f\n", (3.14159 * (raio) * (raio)));
10    return 0;
11 }
```



# Exemplo Prático

```
1 <códigos que estavam no stdio.h>
2
3
4
5
6 int main()
7 {
8     double raio = 2.0;
9     printf("Área: %.2f\n", (3.14159 * (raio) * (raio)));
10    return 0;
11 }
```



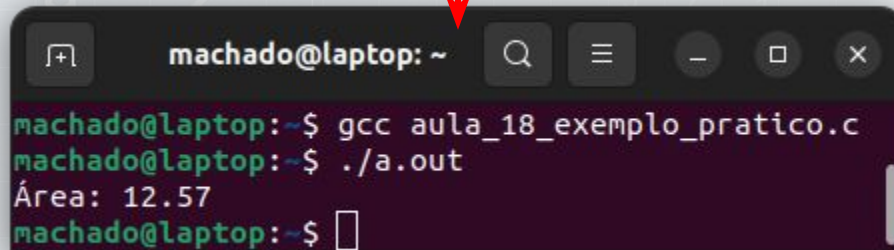
A terminal window titled 'machado@laptop: ~' with standard window controls. It shows the compilation and execution of a C program. A red arrow points from the code above to the terminal.

```
machado@laptop:~$ gcc aula_18_exemplo_pratico.c
machado@laptop:~$ ./a.out
Área: 12.57
machado@laptop:~$
```



# Exemplo Prático

```
1#include <stdio.h>
2
3#define PI 3.14159
4#define AREA_CIRCULO(r) (PI * (r) * (r))
5
6int main()
7{
8    double raio = 2.0;
9    printf("Área: %.2f\n", AREA_CIRCULO(raio));
10    return 0;
11}
```



A terminal window titled 'machado@laptop: ~' with search, menu, and window control icons. It shows the compilation and execution of the C program. A red arrow points from the code block above to the terminal.

```
machado@laptop:~$ gcc aula_18_exemplo_pratico.c
machado@laptop:~$ ./a.out
Área: 12.57
machado@laptop:~$
```

# Dica

Evite o uso de valores **literais** (como 3.14, 100, 255, etc.) que aparecem no código **sem contexto ou explicação**:

```
if (velocidade > 100)
{
    printf("Velocidade acima do permitido!\n");
}
```

Por que 100? O que significa? É um número mágico!

# Dica

Evite o uso de valores **literais** (como 3.14, 100, 255, etc.) que aparecem no código **sem contexto ou explicação**:

```
#define LIMITE_VELOCIDADE 100

if (velocidade > LIMITE_VELOCIDADE)
{
    printf("Velocidade acima do permitido!\n");
}
```

Agora:

- O código é **mais legível**
- Se o limite mudar, você altera **apenas a macro**
- Ajuda na **manutenção, documentação implícita e reduz erros**

# Macros e Diferentes Plataformas

As **macros em C** são amplamente utilizadas para escrever **código portátil entre diferentes plataformas**, especialmente com o uso de **diretivas de pré-processamento condicionais** como `#ifdef`, `#ifndef`, `#if`, `#else`, `#endif`.

- O **pré-processador** verifica se a macro `_WIN32`, `__linux__` ou `__APPLE__` está definida (elas são geralmente **predefinidas pelo compilador** conforme o sistema).
- Isso permite **escrever diferentes trechos de código** para cada sistema, mantendo **o mesmo arquivo-fonte**.



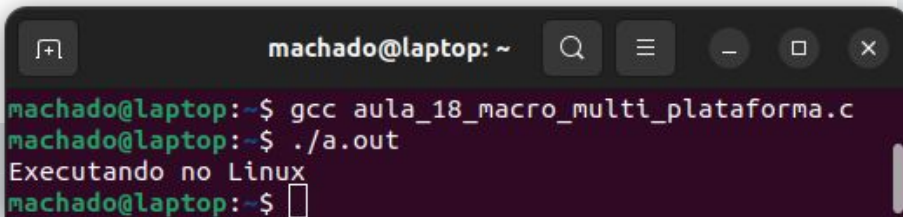
# Macros e Diferentes Plataformas

```
1 #include <stdio.h>
2
3 int main()
4 {
5     #ifdef _WIN32
6         printf("Executando no Windows\n");
7     #elif __linux__
8         printf("Executando no Linux\n");
9     #elif __APPLE__
10        printf("Executando no macOS\n");
11    #else
12        printf("Sistema operacional desconhecido\n");
13    #endif
14
15    return 0;
16 }
```



# Macros e Diferentes Plataformas

```
1 #include <stdio.h>
2
3 int main()
4 {
5     #ifdef _WIN32
6         printf("Executando no Windows\n");
7     #elif __linux__
8         printf("Executando no Linux\n");
9     #elif __APPLE__
10        printf("Executando no macOS\n");
11    #else
12        printf("Sistema operacional desconhecido\n");
13    #endif
14
15    return 0;
16 }
```

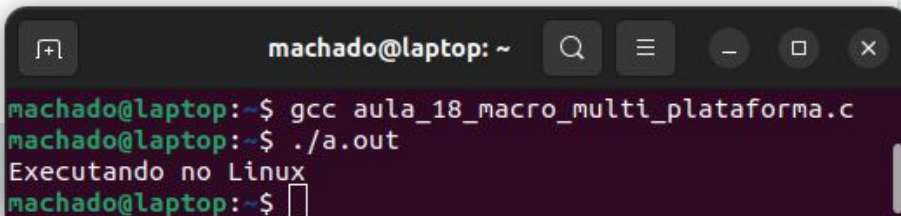


A terminal window titled 'machado@laptop: ~' with standard window controls. It shows the compilation and execution of the program. The command 'gcc aula\_18\_macro\_multi\_plataforma.c' is executed, followed by './a.out'. The output is 'Executando no Linux', which matches the platform detected by the code. The prompt returns to 'machado@laptop:~\$'.

```
machado@laptop: ~
machado@laptop:~$ gcc aula_18_macro_multi_plataforma.c
machado@laptop:~$ ./a.out
Executando no Linux
machado@laptop:~$
```

# Macros e Diferentes Plataformas

```
1 <linhas de codigos copiadas do stdio.h>
2
3 int main()
4 {
5
6
7
8     printf("Executando no Linux\n");
9
10
11
12
13
14
15     return 0;
16 }
```



A terminal window with a dark background and light text. The title bar shows 'machado@laptop: ~'. The command prompt shows the user is in the home directory. The user enters 'gcc aula\_18\_macro\_multi\_plataforma.c' and the compiler outputs 'a.out'. Then the user enters './a.out' and the program outputs 'Executando no Linux'.

```
machado@laptop: ~
machado@laptop:~$ gcc aula_18_macro_multi_plataforma.c
machado@laptop:~$ ./a.out
Executando no Linux
machado@laptop:~$
```

# Cuidados com Macros

- Substituição textual = **sem verificação de tipo**

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

# Cuidados com Macros

- Substituição textual = **sem verificação de tipo**
- **Use parênteses** para evitar erros:
  - Não use:

```
#define QUADRADO(x) x*x
```

# Cuidados com Macros

- Substituição textual = **sem verificação de tipo**
- **Use parênteses** para evitar erros:

- **Não use:**

```
#define QUADRADO(x) x*x
```

- **Use:**

```
#define QUADRADO(x) ((x) * (x))
```



# Cuidados com Macros

```
1 #include <stdio.h>
2
3 #define DOBR0(x) x * 2
4
5 int main()
6 {
7     int resultado = DOBR0(3 + 1); // Esperado: (3 + 1) * 2 = 8
8     printf("Resultado: %d\n", resultado);
9     return 0;
10 }
```

# Cuidados com Macros

O que realmente acontece?

```
1 #include <stdio.h>
2
3 #define DOBR0(x) x * 2
4
5 int main()
6 {
7     int resultado = DOBR0(3 + 1); // Esperado: (3 + 1) * 2 = 8
8     printf("Resultado: %d\n", resultado);
9     return 0;
10 }
```

# Cuidados com Macros

O que realmente acontece?

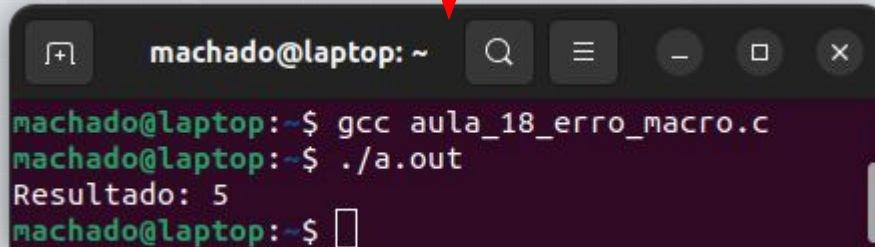
```
1 #include <stdio.h>
2
3 #define DOBR0(x) x * 2
4
5 int main()
6 {
7     int resultado = DOBR0(3 + 1); // Esperado: (3 + 1) * 2 = 8
8     printf("Resultado: %d\n", resultado);
9     return 0;
10 }
```

# Cuidados com Macros

O que realmente acontece?

```
1 #include <stdio.h>
2
3 #define DOBR0(x) x * 2
4
5 int main()
6 {
7     int resultado = DOBR0(3 + 1); //
8     printf("Resultado: %d\n", resultado);
9     return 0;
10 }
```

Esperado:  $(3 + 1) * 2 = 8$



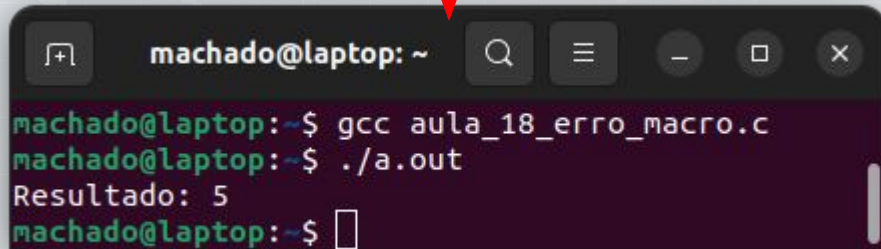
A terminal window titled 'machado@laptop: ~' with standard window controls. It shows the compilation and execution of a C program. The command 'gcc aula\_18\_erro\_macro.c' is executed, followed by './a.out'. The output is 'Resultado: 5', which is highlighted in green. The prompt 'machado@laptop:~\$' is shown at the bottom.

```
machado@laptop:~$ gcc aula_18_erro_macro.c
machado@laptop:~$ ./a.out
Resultado: 5
machado@laptop:~$
```

# Cuidados com Macros

O que realmente acontece?

```
1 #include <stdio.h>
2
3 #define DOBR0(x) x * 2
4
5 int main()
6 {
7     int resultado = 3 + 1 * 2;
8     printf("Resultado: %d\n", resultado);
9     return 0;
10 }
```



```
machado@laptop: ~  
machado@laptop:~$ gcc aula_18_erro_macro.c  
machado@laptop:~$ ./a.out  
Resultado: 5  
machado@laptop:~$
```

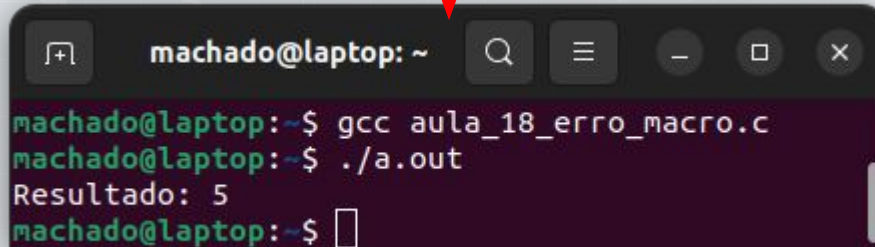


# Cuidados com Macros

O que realmente acontece?

Multiplica primeiro

```
1 #include <stdio.h>
2
3 #define DOBR0(x) x * 2
4
5 int main()
6 {
7     int resultado = 3 + 1 * 2;
8     printf("Resultado: %d\n", resultado);
9     return 0;
10 }
```



```
machado@laptop: ~
machado@laptop:~$ gcc aula_18_erro_macro.c
machado@laptop:~$ ./a.out
Resultado: 5
machado@laptop:~$
```

# Macros

- **Macros** (#define) são instruções de substituição textual feitas **antes da compilação**.
- São úteis para:
  - Definir **constantes** (sem uso de memória)
  - Criar **expressões simples e reutilizáveis**
  - Escrever código **portável entre plataformas**
- **Mas exigem cuidado!**
  - Não fazem **verificação de tipo**
  - Podem gerar **erros sutis** se mal usadas (ex: falta de parênteses)
- Boa prática:
  - Use #define para **constantes simples**
  - Prefira **const** ou **enum** para maior segurança e legibilidade
  - Evite macros para funções complexas

# Enum

```
int main[3][3]:
```

# O que é enum?

- enum (enumeração) permite **definir um conjunto de constantes inteiras nomeadas**.
- Torna o código **mais legível e mais seguro** do que usar números mágicos.
- Exemplo de definição:

```
enum DiaSemana { DOM, SEG, TER, QUA, QUI, SEX, SAB };
```

- Os nomes definidos recebem valores inteiros **automáticos e sequenciais** a partir de 0:
  - Equivalente a: DOM = 0, SEG = 1, TER = 2, ...
- Também é possível **definir esses valores manualmente**:

```
enum Status { OK = 0, ERRO = 1, DESCONHECIDO = -1 };
```



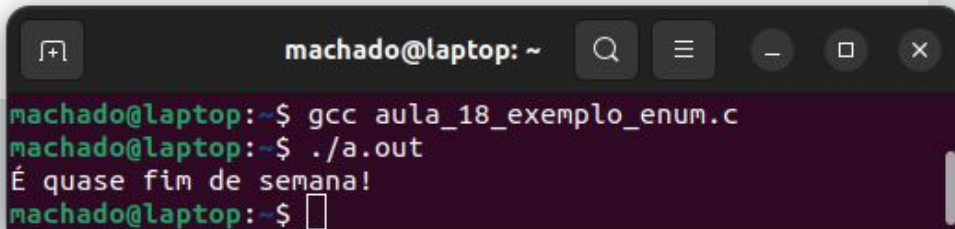
# Enum - Exemplo

```
1 #include <stdio.h>
2
3 enum DiaSemana { DOM, SEG, TER, QUA, QUI, SEX, SAB };
4
5 int main()
6 {
7     enum DiaSemana hoje = SEX;
8
9     if (hoje == SEX || hoje == SAB)
10    {
11        printf("É quase fim de semana!\n");
12    }
13    return 0;
14 }
```



# Enum - Exemplo

```
1 #include <stdio.h>
2
3 enum DiaSemana { DOM, SEG, TER, QUA, QUI, SEX, SAB };
4
5 int main()
6 {
7     enum DiaSemana hoje = SEX;
8
9     if (hoje == SEX || hoje == SAB)
10    {
11        printf("É quase fim de semana!\n");
12    }
13    return 0;
14 }
```



```
machado@laptop: ~
machado@laptop:~$ gcc aula_18_exemplo_enum.c
machado@laptop:~$ ./a.out
É quase fim de semana!
machado@laptop:~$
```

# Limitações

- O compilador ainda trata os valores como **inteiros**.
- Não é possível imprimir o nome ("DOM" ou "SEX") diretamente - só o valor numérico.

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

- Para exibir como string, você precisa usar switch ou um vetor de strings.

|   |   |   |
|---|---|---|
| 5 | 9 | 3 |
| 2 | 6 | 8 |
| 4 | 0 | 7 |

|   |   |   |
|---|---|---|
| 9 | 2 | 3 |
| 7 | 8 | 5 |
| 2 | 2 | 3 |

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 2 | 2 | 0 | 8 | 6 | 0 |
| 1 | 9 | 6 | 5 | 0 | 0 |
| 1 | 2 | 8 | 5 | 2 | 0 |

# 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 | <a href="#">Ver Detalhes</a> <a href="#">Remover Exercício</a> |
| 2   | 1.01 Maior número               | Não Entregue | 0/6            | 0     | 0        | 0/46          | 19/03/2025 23:59:59 | <a href="#">Ver Detalhes</a> <a href="#">Remover Exercício</a> |
| 3   | 1.02 Par ou ímpar               | Não Entregue | 0/7            | 0     | 0        | 0/46          | 19/03/2025 23:59:59 | <a href="#">Ver Detalhes</a> <a href="#">Remover Exercício</a> |
| 4   | 1.03 Positivo, negativo ou zero | Não Entregue | 0/7            | 0     | 0        | 0/46          | 19/03/2025 23:59:59 | <a href="#">Ver Detalhes</a> <a href="#">Remover Exercício</a> |
| 5   | 1.04 Maior de três números      | Não Entregue | 0/7            | 0     | 0        | 0/46          | 19/03/2025 23:59:59 | <a href="#">Ver Detalhes</a> <a href="#">Remover Exercício</a> |

# SSC0501 - Introdução à Ciência de Computação I

Obrigado pela atenção!!