

# Lista de Exercício 02- Linguagem C

Disciplina: Linguagem e Técnicas de Programação  
Professor: Dr. Camilo Barreto  
Curso: Grande Área de Computação  
UNIUBE  
Valor: 15 pts  
Entrega: 05/12/25

## Instruções de Entrega do Exercício:

→ Exercício Individual! ←

O exercício deverá ser entregue **de forma DIGITAL**, seguindo **a estruturação de pastas e arquivos abaixo**.

- Envio pelo sistema **Estudos Autônomos**;
- Você deve seguir rigorosamente a estrutura de pastas e arquivos:

```
exercicio-final-seu-nome (pasta)
├─ questoes_teoricas   (pasta)
│   └─ responder em texto (texto)
├─ q1                  (pasta)
│   └─ codigo_q1.cpp   (código)
├─ q2                  (pasta)
│   └─ codigo_q2.cpp   (código)
└─ ...
```

### Como Compactar?

- Crie a pasta **exercicio-final-seu-nome**.
- Dentro dela, crie uma pasta para as questões teóricas: **questoes\_teoricas**
  - Em um arquivo de texto, responda todas elas apenas informando a letra correta. Ex: 01-B, 02-C, 03-D, ...
- Crie as pastas referentes as questões do enunciado. Ex: **q1, q2, q3, etc**.
- Coloque **codigo\_q1.cpp** dentro de **q1, q2, ...** (cada questão com seu arquivo).
- Clique com o botão direito na pasta **exercicio-final-seu-nome** → **Enviar para** → **Pasta compactada (zipada)**.

**Somente serão aceitos trabalhos que estejam de acordo com estas orientações.**

## Questões Teóricas:

**01** – Qual palavra reservada encerra a execução do bloco `case`, evitando que o fluxo continue no próximo `case` ?

- ☐ A. `continue`
- ☒ B. `break`
- ☐ C. `exit`
- ☐ D. `stop`

**02** – Em um `switch(expr)`, os rótulos `case` devem ser:

- ☒ A. Expressões constantes inteiras/ `char` (ou valores de `enum`).
- ☐ B. `string` (cadeias de caracteres arbitrárias).
- ☐ C. Valores de ponto flutuante.

☐ D. Variáveis calculadas em tempo de execução.

**03** – Sobre o rótulo `default` no `switch`, assinale a correta:

- ☐ A. É obrigatório e precisa vir primeiro.
- ☒ B. É opcional e pode aparecer em qualquer posição dentro do `switch`.
- ☐ C. É obrigatório se houver mais de três `case`.
- ☐ D. Só é executado quando todos os `case` possuem `break`.

**04** – Se um `case` não termina com `break`, o que ocorre?

- ☐ A. Erro de compilação.
- ☒ B. O fluxo “cai” no próximo `case` (comportamento de *fall-through*) até encontrar um `break` ou o fim do `switch`.
- ☐ C. O `switch` reinicia do primeiro `case`.
- ☐ D. O programa encerra.

---

**05** – Na expressão `for (inicialização; condição; incremento)`, a condição representa:

- ☐ A. O passo de incremento/decremento do contador.
- ☒ B. O teste lógico que determina se o laço continua executando.
- ☐ C. A limpeza de recursos ao fim do laço.
- ☐ D. A quantidade fixa de iterações.

**06** – Sobre o **escopo** da variável `i` em `for (int i = 0; ...) { ... }`, marque a correta:

- ☐ A. `i` existe em todo o arquivo.
- ☒ B. `i` existe apenas dentro do bloco do `for` (escopo local ao laço).
- ☐ C. `i` existe em todas as funções.
- ☐ D. `i` existe somente até o primeiro `break`.

**07** – No cabeçalho do `for`, usar `i++` ou `++i` na parte de incremento:

- ☐ A. É inválido utilizar `++i`.
- ☐ B. Duplica o incremento quando se usa `i++`.
- ☒ C. Ambas as formas são válidas e, no cabeçalho do `for`, produzem o mesmo efeito sobre o número de iterações.
- ☐ D. `++i` faz o incremento antes do corpo e altera a ordem das etapas do laço.

**08** – Sobre a sintaxe do `for` em C, assinale a correta:

- ☐ A. Apenas a condição é opcional.
- ☐ B. Apenas a inicialização é obrigatória.
- ☒ C. As três expressões são opcionais; `for(;;)` é um laço infinito válido.
- ☐ D. A expressão de incremento não pode conter mais de uma expressão.

---

**09** – No `while (condição) { ... }`, quando a condição é testada?

- ☒ A. Antes de cada iteração.
- ☐ B. Somente depois da primeira iteração.
- ☐ C. Apenas na primeira vez.
- ☐ D. Somente quando ocorre `break`.

**10** – Sobre `while(1) { ... }`, é correto afirmar:

- ☐ A. É inválido em C.
- ☒ B. Cria um laço infinito e normalmente se sai com `break` / `return`.
- ☐ C. Executa uma única vez.
- ☐ D. Acontece um erro durante a execução.

11 – Analise o código `while (x < 10); { x++; }` e assinale a correta:

- ☐ A. Não há problema.
- ☒ B. O `;` após o `while` cria um laço vazio e o bloco `{ x++; }` fica fora do `while` em uma execução infinita.
- ☐ C. O compilador remove automaticamente o `;`.
- ☐ D. O código sempre decrementa `x`.

12 – Sobre usar **atribuição** na condição, analise o trecho abaixo e assinale a alternativa correta:

```
int x = 3;
while ( (x = x - 1) > 0 ) {
    /* corpo do laço */
}
```

- ☐ A. É proibido usar `=` na condição do `while`.
- ☒ B. Em C, a atribuição tem valor (o resultado atribuído a `x`), portanto a expressão é válida; é preciso apenas não confundir `=` com `==`.
- ☐ C. O laço nunca executa porque `x = x - 1` não altera `x`.
- ☐ D. O compilador interpreta `=` como `==` dentro de condições.

13 – Em C, o índice do **primeiro** elemento de um vetor é:

- ☐ A. `1`
- ☒ B. `0`
- ☐ C. `-1`
- ☐ D. Depende do compilador

14 – Sobre vetores em C, assinale a alternativa correta:

- ☒ A. Um vetor armazena valores do mesmo tipo em posições contíguas de memória.
- ☐ B. Um vetor pode misturar `int`, `float` e `char` na mesma declaração.
- ☐ C. O tamanho do vetor é definido automaticamente a cada acesso.
- ☐ D. O nome do vetor deve ser sempre `array`.

15 – Dado `int v[5] = {10, 20, 30, 40, 50};`, qual é o índice do último elemento e como você o acessa?

- ☐ A. Índice 5, acesso `v[5]`
- ☒ B. Índice 4, acesso `v[4]`
- ☐ C. Índice 0, acesso `v[0]`
- ☐ D. Índice 1, acesso `v[1]`

16 – Sobre acessar `v[5]` em `int v[5];` (índice fora do limite):

- ☐ A. É permitido e retorna `0`.
- ☒ B. É comportamento indefinido (erro lógico).
- ☐ C. Lança exceção automática.
- ☐ D. O compilador ajusta para o último índice válido.

17 – Protótipo correto de uma função que recebe `int` e retorna `float`:

- ☒ A. `float f(int x);`
- ☐ B. `prot float f(int);`
- ☐ C. `void f(float, int);`
- ☐ D. `int f(void);`

18 – Em C, a passagem de argumentos é:

- ☐ A. Por referência, sempre.

- ☒ B. Por valor; para permitir alteração da variável original usa-se ponteiros.
- ☐ C. Por cópia dupla e revertida.
- ☐ D. Definida pelo compilador em tempo de execução.

19 – Considere `int soma(int a, int b) { return a + b; }`. Assinale a correta:

- ☒ A. O tipo de retorno é `int` e a assinatura é `soma(int, int)`.
- ☐ B. O tipo de retorno é `void` e a assinatura é `soma(a, b)`.
- ☐ C. O tipo de retorno é `int` e a assinatura é `soma(int)`.
- ☐ D. O tipo de retorno é `int` e não existe conceito de assinatura em C.

20 – Sobre funções com `void`:

- ☒ A. `void` indica que a função **não retorna valor**.
- ☐ B. `void` é sinônimo de `int`.
- ☐ C. Funções `void` podem usar `return 1;` livremente.
- ☐ D. Funções `void` não podem receber parâmetros.

21 – Sobre protótipos e definições de funções em C, assinale a alternativa correta:

- ☒ A. O protótipo informa ao compilador o tipo de retorno e os parâmetros, permitindo chamar a função antes da sua definição.
- ☐ B. O protótipo deve repetir o corpo da função entre chaves.
- ☐ C. Protótipos só podem ser declarados dentro da função `main`.
- ☐ D. Sem protótipo, o compilador sempre assume que a função retorna `void`.

22 – Para inicializar a semente do gerador antes de usar `rand()`:

- ☒ A. `srand( time(NULL) );`
- ☐ B. `rand( time(NULL) );`
- ☐ C. `seed( time(0) );`
- ☐ D. `srand();`

23 – Para obter um inteiro no intervalo **[0, 9]** de forma simples:

- ☒ A. `rand() % 10`
- ☐ B. `rand(10)`
- ☐ C. `1 + rand() % 10`
- ☐ D. `rand() % 9 + 1`

24 – Dada a definição `struct Pessoa { char nome[50]; int idade; };`, como declarar uma variável do tipo?

- ☒ A. `struct Pessoa p;`
- ☐ B. `Pessoa p;`
- ☐ C. `struct p Pessoa;`
- ☐ D. `peessoa p;`

25 – Como acessar o campo `idade` da variável `struct Pessoa p;`?

- ☐ A. `p->idade`
- ☒ B. `p.idade`
- ☐ C. `idade.p`
- ☐ D. `Pessoa.idade`

26 – Dada `typedef struct { int x, y; } Ponto;`, escolha uma declaração e inicialização válidas:

- ☒ A. `Ponto p = {10, 20};`
- ☐ B. `struct Ponto p = {10, 20};`

- ☐ C. `Ponto p = (10, 20);`
- ☐ D. `Ponto p; p.x,y = 10,20;`

27 – Dada `struct Pessoa { char nome[50]; int idade; };`, qual alternativa declara uma variável `p` e **define** sua idade como 25?

- ☒ A. `struct Pessoa p; p.idade = 25;`
- ☐ B. `struct Pessoa p; p->idade = 25;`
- ☐ C. `Pessoa p; p.idade = 25;`
- ☐ D. `struct Pessoa p; Pessoa.idade = 25;`

## Questões Práticas:

### 28 — Menu: Dia da Semana

Crie um programa que exiba um menu numerado (1 a 7) para o usuário escolher um dia da semana e imprima o nome correspondente (1=Domingo, 2=Segunda, ..., 7=Sábado). Se o usuário digitar um número fora desse intervalo, mostre “Opção inválida”.

- **Requisitos:** usar `switch` com rótulos `case` e um `default`; mensagens claras para o usuário.
- **Entrada (exemplo):**

Digite um número (1 a 7): 3

- **Saída (exemplo):**

Dia selecionado: Terça-feira

- **Dicas:** leia um `int` com `scanf("%d", &op);`; use `default` para tratar valores inválidos.

### 29 — Menu: Mini-Calculadora

Mostre um menu: 1) Somar 2) Subtrair 3) Multiplicar 4) Dividir. Leia a opção e depois dois números. Execute a operação escolhida e mostre o resultado.

- **Requisitos:** `switch` para decidir a operação; tratar divisão por zero; mensagens antes de cada leitura.
- **Entrada (exemplo):**

Escolha a operação (1-Somar, 2-Subtrair, 3-Multiplicar, 4-Dividir): 4  
Digite o primeiro número: 8  
Digite o segundo número: 0

- **Saída (exemplo):**

Erro: divisão por zero.

- **Dicas:** após ler a opção, use `if` dentro do `case` da divisão para verificar o divisor.

### 30 — Menu: Lanchonete

Exiba um cardápio com códigos de produtos:

`100 - Coxinha (R$ 6,00)` , `101 - Pão de queijo (R$ 4,50)` , `102 - Suco (R$ 5,00)` , `0 - Finalizar` .

O usuário pode digitar repetidamente um código para **acumular** o total da compra e finalizar com `0`. Ao final, mostre o total.

- **Requisitos:** usar `switch` para somar ao total conforme o código; permitir múltiplas entradas; finalizar ao digitar `0`; mensagem antes de **cada** leitura.
- **Entrada (exemplo):**

Digite o código do produto (0 para finalizar): 100  
Digite o código do produto (0 para finalizar): 102  
Digite o código do produto (0 para finalizar): 101  
Digite o código do produto (0 para finalizar): 0

- **Saída (exemplo):**

Total: R\$ 15,50

- **Dicas:** use um laço para repetir a leitura do código; a **decisão de preço** deve ser feita com `switch`.

### 31 — Controle de Despesas Mensais

Leia `N` (quantidade de despesas) e em seguida **N** valores (cada um representando um gasto). Calcule e mostre o **total** e a **média**.

- **Requisitos:** usar `for` para somar; mensagens antes de cada leitura.
- **Entrada (exemplo):**

Digite a quantidade de despesas: 4  
Digite a despesa 1: 120.50  
Digite a despesa 2: 80.00  
Digite a despesa 3: 45.90  
Digite a despesa 4: 200.00

- **Saída (exemplo):**

Total=446.40, Média=111.60

- **Dicas:** use acumulador `float total = 0;` e divida por `N` ao final.

### 32 — Relatório de Produção (7 dias)

Leia a quantidade produzida em cada um dos **7 dias** e exiba: total produzido na semana e o **maior valor diário**.

- **Requisitos:** laço `for` de 1 a 7; acompanhar total e o maior; mensagens antes de cada leitura.
- **Entrada (exemplo):**

Digite a produção do dia 1: 10  
Digite a produção do dia 2: 12  
Digite a produção do dia 3: 7  
Digite a produção do dia 4: 15  
Digite a produção do dia 5: 9  
Digite a produção do dia 6: 0  
Digite a produção do dia 7: 20

- **Saída (exemplo):**

Total=73 | Pico diário=20

- **Dicas:** inicialize o “maior” com a primeira leitura.

### 33 — Pesquisa de Satisfação (contagem)

Leia `N` respostas de clientes (valores inteiros: 1=Ruim, 2=Regular, 3=Bom, 4=Ótimo). Conte quantas respostas de cada tipo e apresente o resultado.

- **Requisitos:** `for` para processar `N` entradas; 4 contadores; mensagens antes de cada leitura.
- **Entrada (exemplo):**

```
Digite a quantidade de respostas: 6
Digite a resposta 1 (1=Ruim,2=Regular,3=Bom,4=Ótimo): 3
Digite a resposta 2 (1=Ruim,2=Regular,3=Bom,4=Ótimo): 4
Digite a resposta 3 (1=Ruim,2=Regular,3=Bom,4=Ótimo): 2
Digite a resposta 4 (1=Ruim,2=Regular,3=Bom,4=Ótimo): 3
Digite a resposta 5 (1=Ruim,2=Regular,3=Bom,4=Ótimo): 1
Digite a resposta 6 (1=Ruim,2=Regular,3=Bom,4=Ótimo): 4
```

- **Saída (exemplo):**

```
Ruim=1 Regular=1 Bom=2 Ótimo=2
```

- **Dicas:** valide entradas fora de 1..4, se desejar, e ignore ou peça nova digitação.

### 34 — Jogo da Adivinhação

O programa deve sortear um número de **0 a 100**. O usuário tenta adivinhar; após cada palpite, informe se é **maior** ou **menor** que o número sorteado. Quando acertar, mostre quantas tentativas foram necessárias.

- **Requisitos:** usar `rand()` e `srand(time(NULL))` ; laço `while` até acertar; mensagens antes de cada palpite.
- **Entrada/saída (exemplo):**

```
Digite seu palpite: 50
Maior!
Digite seu palpite: 70
Menor!
Digite seu palpite: 65
Menor!
Digite seu palpite: 63
Acertou em 4 tentativas!
```

- **Dicas:** gere o número com `rand() % 101` .

### 35 — Menu até “Sair com S/s”

Mostre um pequeno menu informativo. Após executar a ação escolhida (pode ser apenas imprimir uma mensagem), pergunte “Deseja sair? (S/N)”. Repita enquanto a resposta for `N` / `n` .

- **Requisitos:** `while` controlado por um `char` ; encerrar apenas se `S` ou `s` ; mensagens antes de cada leitura.
- **Entrada/saída (exemplo):**

```
Escolha uma opção: 1
Executando opção 1...
Deseja sair? (S/N): n
Escolha uma opção: 2
Executando opção 2...
Deseja sair? (S/N): S
Encerrando...
```

- **Dicas:** ao ler `char` , use `scanf(" %c", &resp);` (note o espaço antes de `%c` ou use `fflush(stdin)`).

### 36 — Soma até Sentinela

Leia inteiros e some-os enquanto o usuário **não** digitar `0` . Ao digitar `0` , exiba a soma.

- **Requisitos:** laço `while` com sentinela `0` ; mensagens antes de cada leitura.
- **Entrada (exemplo):**

```
Digite um número (0 para finalizar): 5
Digite um número (0 para finalizar): 7
Digite um número (0 para finalizar): -2
Digite um número (0 para finalizar): 0
```

- **Saída (exemplo):**

```
Soma=10
```

- **Dicas:** leia, teste sentinela e acumule até que seja `0`.

### 37 — Estatísticas de Notas

Leia quantidade de notas `N` e depois `N` notas (vetor `float`). Calcule **média**, **maior** e **menor** e mostre os resultados.

- **Requisitos:** armazenar notas em vetor; usar `for` para percorrer; mensagens antes de cada leitura.
- **Entrada (exemplo):**

```
Digite a quantidade de notas (N até 50): 5
Digite a nota 1: 7.5
Digite a nota 2: 8.0
Digite a nota 3: 6.0
Digite a nota 4: 10.0
Digite a nota 5: 9.0
```

- **Saída (exemplo):**

```
Média=8.10, Maior=10.00, Menor=6.00
```

- **Dicas:** você pode calcular tudo em um único laço, mantendo acumulador, maior e menor.

### 38 — Substituir Negativos por Zero

Leia `N` inteiros em um vetor. Substitua todos os valores **negativos** por `0` e mostre o vetor atualizado na mesma ordem.

- **Requisitos:** usar `for` para varrer o vetor e **alterar** valores; mensagens antes de cada leitura.
- **Entrada (exemplo):**

```
Digite a quantidade de valores: 6
Digite o valor 1: 5
Digite o valor 2: -1
Digite o valor 3: 3
Digite o valor 4: -7
Digite o valor 5: 0
Digite o valor 6: 8
```

- **Saída (exemplo):**

```
Vetor atualizado: 5 0 3 0 0 8
```

- **Dicas:** `if (v[i] < 0) v[i] = 0;`.

### 39 — Inverter Vetor “in place”

Leia `N` inteiros (até 100) e armazene em um vetor. Inverta a **ordem** dos elementos **no próprio vetor** e imprima o resultado.

- **Requisitos:** usar `for` até a metade do vetor, trocando `v[i]` com `v[N-1-i]` ; mensagens antes de cada leitura.
- **Entrada (exemplo):**

```
Digite a quantidade de valores: 5
Digite o valor 1: 1
Digite o valor 2: 2
Digite o valor 3: 3
Digite o valor 4: 4
Digite o valor 5: 5
```

- **Saída (exemplo):**

```
Vetor invertido: 5 4 3 2 1
```

- **Dicas:** use uma variável temporária para realizar a troca.

## 40 — Calculadora com Funções

Implemente quatro funções com **parâmetros e retorno** (tipos primitivos):

```
float somar(float a, float b) , float subtrair(float a, float b) , float multiplicar(float a, float b) , float dividir(float a, float b) .
```

No `main` , leia dois números e uma opção de operação e chame a função correspondente.

- **Requisitos:** protótipos; tratamento de divisão por zero; mensagens antes de cada leitura.
- **Entrada (exemplo):**

```
Escolha a operação (1-Somar, 2-Subtrair, 3-Multiplicar, 4-Dividir): 3
Digite A: 2.5
Digite B: 4
```

- **Saída (exemplo):**

```
Resultado=10.0
```

- **Dicas:** use `switch` para decidir qual função chamar.

## 41 — Média Ponderada com Função

Crie uma função `float media_ponderada(float n1, float n2, float n3, float p1, float p2, float p3)` que retorna a média ponderada. No `main` , leia as 3 notas e 3 pesos e exiba o resultado.

- **Requisitos:** todos os parâmetros e retorno do tipo `float` ; mensagens antes de cada leitura.
- **Entrada (exemplo):**

```
Digite a nota 1: 7
Digite o peso 1: 2
Digite a nota 2: 8
Digite o peso 2: 3
Digite a nota 3: 10
Digite o peso 3: 5
```

- **Saída (exemplo):**

```
Média ponderada=8.9
```

- **Dicas:** lembre de dividir pela soma dos pesos.

## 42 — Máximo e Mínimo com Funções

Implemente duas funções: `int max3(int a, int b, int c)` e `int min3(int a, int b, int c)`. No `main`, leia três inteiros, chame as funções e imprima o maior e o menor.

- **Requisitos:** apenas tipos primitivos; retornar o valor calculado; mensagens antes de cada leitura.
- **Entrada (exemplo):**

```
Digite o valor A: 7
Digite o valor B: 2
Digite o valor C: 9
```

- **Saída (exemplo):**

```
Maior=9, Menor=2
```

- **Dicas:** compare usando `if` encadeados.

### 43 — Cadastro de Aluno

Defina uma `struct Aluno` com `nome` (string), `ra` (inteiro), `email` (string) e `curso` (string). Leia os dados de **um** aluno e imprima um “cartão” com as informações organizadas.

- **Requisitos:** declarar a `struct`; declarar uma variável do tipo; ler e imprimir os campos; mensagens antes de cada leitura.
- **Entrada (exemplo):**

```
Digite o nome: João Silva
Digite o RA: 123456
Digite o email: joao@exemplo.com
Digite o curso: Sistemas de Informação
```

- **Saída (exemplo):**

```
Aluno: João Silva (RA 123456)
Email: joao@exemplo.com
Curso: Sistemas de Informação
```

- **Dicas:** para campos de texto, use um tamanho fixo (ex.: `char nome[60];`).

### 44 — Retângulo: Área e Perímetro

Crie uma `struct Retangulo` com `largura` e `altura` (`float`). Leia os valores, calcule e mostre **área** e **perímetro**.

- **Requisitos:** declarar a `struct`; ler; calcular `área = largura * altura` e `perímetro = 2*(largura + altura)`; mensagens antes de cada leitura.
- **Entrada (exemplo):**

```
Digite a largura: 5.0
Digite a altura: 2.5
```

- **Saída (exemplo):**

```
Área=12.50, Perímetro=15.00
```

- **Dicas:** formate a saída com duas casas decimais (`%.2lf`).

### Bom trabalho!

Quanquer dúvida envie um email para o professor: [camilo.junior@uniube.br](mailto:camilo.junior@uniube.br)