
	<b>MINISTÉRIO DA EDUCAÇÃO</b> <b>UNIVERSIDADE FEDERAL DO PIAUÍ – UFPI</b> <b>CAMPUS SENADOR HELVÍDIO NUNES DE BARROS – PICOS</b>		
<b>Curso: Sistemas de Informação</b>		<b>Período: 2º</b>	<b>Ano/Semestre: 2025.1</b>
<b>Disciplina: Algoritmos e Programação II</b>			<b>Professor: José Denes Lima Araújo</b>

## 6º ATIVIDADE – PONTEIRO, PONTEIRO de PONTEIRO e VETOR e PONTEIRO

1. Dado o código abaixo diga quais valores serão impressos.

```
int main(){
    int a,b,*p,**p1;
    p=&a;
    a=20;
    p1=&p;
    b=50;
    a=b+(*p);
    b=(**p1)+a;
    printf("a= %d e b= %d",a,b);
    return 0;
}
```

endereço	variável	conteúdo
#119		
#120		
#121		
#122		
#123		
#124		
#125		
#126		
#127		
#128		
#129		

2. Qual os valores das variáveis a e b ao final do programa?

```
int main(){  
    int a = 10, b = 20, *ptr1, *ptr2;  
    ptr1 = &a;  
    *ptr1 = *ptr1 + b;  
    *ptr2 = *ptr1 - *ptr2;  
    ptr1 = ptr2;  
    *ptr1 = a + *ptr1;  
    printf("a = %d, b = %d\n", a, b);  
    return 0;  
}
```

endereço	variável	conteúdo
#119		
#120		
#121		
#122		
#123		
#124		
#125		
#126		
#127		
#128		
#129		

3. Crie um programa que leia as notas de 5 alunos em um vetor. Utilize um ponteiro para substituir todas as notas que são menores que 6 por 6, para garantir a aprovação mínima. Exiba o vetor de notas corrigidas.

**Exemplo:**

**Entrada:**

Notas: 3 8 5 7 4

**Saída:**

Notas corrigidas: 6 8 6 7 6

4. Implemente uma função em linguagem C que receba como parâmetro um **ponteiro para char**, representando uma **palavra**, e ordene os caracteres dessa palavra em **ordem alfabética crescente**.
- A ordenação deve ser feita **utilizando ponteiros para acessar e trocar os caracteres**, sem o uso direto de notação de índice (ou seja, evite `str[i]`).
  - O usuário deve digitar uma palavra, e o programa deve exibir a palavra com seus caracteres já ordenados.

**Exemplo:**

**Entrada:** “palavra”

**Saída:** “aaalprv”

5. Implemente uma função em linguagem C que receba como parâmetros dois **ponteiros para ponteiros de inteiro** (`int **p1` e `int **p2`) e troque os endereços armazenados nesses ponteiros para ponteiro, ou seja, faça com que após a chamada da função, o primeiro ponteiro aponte para o endereço que o segundo apontava antes, e vice-versa.

**Especificações:**

- A função **não deve alterar os valores inteiros apontados**, apenas os endereços armazenados nos ponteiros para ponteiro.
- Você deve testar essa função no main criando duas variáveis inteiras, dois ponteiros que apontam para essas variáveis, e dois ponteiros para esses ponteiros.
- Após a troca, mostre que os ponteiros para ponteiros trocaram as referências dos ponteiros para inteiro

### Exemplo:

```
int main(){
    int a = 10, b = 20;
    int *ptrA = &a;
    int *ptrB = &b;
    int **pptrA = &ptrA;
    int **pptrB = &ptrB;
    printf("***pptrA = %d  **pptrB = %d\n", **pptrA, **pptrB); // **pptrA = 10; **pptrB = 20
    trocar_end(pptrA, pptrB);
    printf("***pptrA = %d  **pptrB = %d\n", **pptrA, **pptrB); // **pptrA = 20; **pptrB = 10
}
```

6. Desenvolva um programa em linguagem C para gerenciar um pequeno catálogo de produtos de uma loja. Cada produto possui nome e preço

Utilize a seguinte estrutura para representar um produto:

```
typedef struct {
    char nome[30];
    float preco;
} Produto;
```

**Implemente as seguintes funcionalidades utilizando ponteiros:**

- void aplicarDesconto(Produto \*p, float \*percentual);
  - Esta função deve aplicar um desconto percentual sobre o preço de um produto.
  - O novo preço deve ser calculado dentro da função e atualizado diretamente no campo preco da struct.
  - O percentual é passado por ponteiro (por exemplo, 10 significa 10%).

- `void buscarMaisCaro(Produto *v, int n, int *indiceMaisCaro);`
  - Esta função deve receber um vetor de produtos e o número total de elementos.
  - Ela deve identificar o índice do produto com o maior preço e retornar esse índice por meio do parâmetro `indiceMaisCaro`.

No **main** faça:

- Declare e preencha manualmente um vetor com 5 produtos.
- Solicite ao usuário um percentual de desconto e aplique-o ao segundo produto da lista.
- Exiba os dados do produto mais caro utilizando a função `buscarMaisCaro`.

**7.** Desenvolva um programa em linguagem C para gerenciar livros. Cada livro possui as seguintes informações:

- título: uma string de até 50 caracteres;
- ISBN: uma string de até 13 caracteres (identificador único);
- ano: um número inteiro que representa o ano de publicação.

**Especificações:**

- Permitir o cadastro de  $n$  livros, onde  $n$  é um valor fornecido pelo usuário.
- Não permitir o cadastro de dois livros com o mesmo ISBN.
- Permitir a exibição de todos os livros cadastrados.
- Permitir a remoção de um livro pelo ISBN.
- A variável que armazena o total de livros cadastrados deve ser local, não global.
- Nenhuma lógica deve estar no **main**.

**O programa deve ter as seguintes funções obrigatórias:**

- `void CadastrarLivro(Livro *livros, int *totalLivros, int maxLivros);`

- void MostrarLivros(Livro \*livros, int totalLivros);
- void RemoverLivro(Livro \*livros, int \*totalLivros);

8. Implemente uma função em C que receba uma string contendo uma frase e retorne a quantidade de palavras presentes nela.

**Requisitos:**

- Considere que as palavras são separadas por um ou mais espaços em branco.
- A função deve percorrer a string utilizando apenas ponteiros (sem usar índices).
- Deve contar corretamente o número de palavras, ignorando espaços múltiplos consecutivos.
- Use o protótipo a seguir:

```
int contarPalavras(char *str);
```

**Exemplo:**

**Entrada:** “ Olá mundo, isso é um teste”

**Saída:** 6 palavras

9. Implemente uma função em linguagem C que receba uma string e remova todas as vogais (maiúsculas e minúsculas) do seu conteúdo, armazenando o resultado em uma nova string.

**Requisitos:**

- A função deve percorrer a string original usando ponteiros.
- A nova string (sem vogais) deve ser montada usando também apenas ponteiros.
- A string resultante deve conter apenas as consoantes.
- Use o seguinte protótipo de função:

```
void removerVogais(char *entrada, char *saida);
```

**Exemplo:**

**Entrada:** "Algoritmos 2"

**Saída:** "lgrtms 2"

- 10.** Desenvolva um programa em linguagem C que leia dois vetores de números reais, ambos de tamanho  $n$ , sendo que  $n < 200$ . O programa deve calcular o **produto escalar entre esses dois vetores e verificar se eles são ortogonais**.

Dois vetores são considerados **ortogonais se o produto escalar entre eles for zero** (ou próximo de zero, devido à precisão dos números reais).

**Para calcular o produto escalar de dois vetores de tamanho  $n$ :** Multiplica-se os elementos correspondentes dos vetores e depois soma-se todos esses resultados. **Exemplo:**  $*(v1 + i) * *(v2 + i) + *(v1 + i) * *(v2 + i) + \dots + *(v1 + (n - 1)) * *(v2 + (n - 1))$ .

Devido às limitações da representação de números de ponto flutuante em computadores, pode acontecer de um valor que "matematicamente" deveria ser zero não ser exatamente zero. **Por isso, considere que um número é zero se estiver dentro da margem de erro ( $EPS = 0.001$ ).** Regra para considerar zero: **produtoEscalar  $< EPS$  e produtoEscalar  $> -EPS$ .**

Faça uma função que calcula o produto escalar usando ponteiros e aritmética de ponteiros. Manipulação dos vetores utilizando ponteiros **(sem usar índice [] nas operações principais)**.

**Exemplo 1:**

**Entrada:**

Digite o tamanho dos vetores: 3

Digite os valores do primeiro vetor:

1 1 1

Digite os valores do segundo vetor:

0 0 0

**Saída:**

Os vetores sao ortogonais.

**Exemplo 2:**

**Entrada:**

Digite o tamanho dos vetores: 3

Digite os valores do primeiro vetor:

1 2 -1

Digite os valores do segundo vetor:

2 -1 0

**Saída:**

Os vetores sao ortogonais.

**Exemplo 3:**

**Entrada:**

Digite o tamanho dos vetores: 4

Digite os valores do primeiro vetor:

1 2 3 4

Digite os valores do segundo vetor:

4 3 2 1

**Saída:**

Os vetores nao sao ortogonais.

- 11.** Implemente um programa em linguagem c que crie uma função para inverter a ordem das palavras em uma string, mantendo a ordem dos caracteres dentro de cada palavra.

OBS: Modifique a string in-place (sem usar strings auxiliares). Use apenas ponteiros, sem índices [].

**Exemplo:**



**Entrada:**

“Ponteiros em c sao poderosos”

**Saída:**

“poderosos sao c em Ponteiros”

- 12.** Dado o código abaixo, explique o que será impresso e preencha o diagrama de memória.

```
7      int main(){
8          int a = 1, b = 2, c = 3;
9          int *p1 = &a, *p2 = &b, *p3 = &c;
10         int **pp1 = &p1, **pp2 = &p2, **pp3 = &p3;
11         **pp1 = *p2 + **pp3;
12         *pp1 = *pp3;
13         **pp2 = **pp1 + *p3;
14         printf("%d %d %d", a, b, c);
15         return 0;
16     }
```

endereço	variável	conteúdo
#119		
#120		
#121		
#122		
#123		
#124		
#125		
#126		
#127		
#128		
#129		

- 13.** Faça, em linguagem c, uma struct Fracao com campos: denominador e numerador. Faça um vetor do tipo Fracao de tamanho  $n > 0$  e uma função para simplificar as frações no vetor usando ponteiros.

```
typedef struct {  
    int numerador;  
    int denominador;  
} Fracao;  
  
void simplificar(Fracao *frac);
```

**Exemplo:**

**Entrada:**

Digite a quantidade de fracoes: 3

Digite o numerador da fracao 1: 8

Digite o denominador da fracao 1: 12

Digite o numerador da fracao 2: 50

Digite o denominador da fracao 2: 100

Digite o numerador da fracao 3: 7

Digite o denominador da fracao 3: 13

**Saída:**

Fracao 1 simplificada: 2/3

Fracao 2 simplificada: 1/2

Fracao 3 simplificada: 7/13

**14.** Implemente um programa em c para validar senhas:

**Requisitos:**

- A senha deve ter entre 8 e 20 caracteres;
- Deve conter pelo menos uma letra maiúscula, um número, um caractere especial.
- Função para validar a senha: A função deve percorrer a string usando **apenas ponteiros** (sem índices). Ela retorna 1 se a senha for válida e 0 caso contrário.

**Exemplo:**

**Entrada:**

Digite a senha para validar: Abcdef1@

**Saída:**

Senha VALIDA.

- 15.** Implemente uma função em linguagem C que receba como parâmetro um ponteiro para uma matriz quadrada de inteiros (de tamanho  $n \times n$ ) e seu tamanho  $n$ , e que troque a diagonal principal pela diagonal secundária da matriz utilizando apenas aritmética de ponteiros (sem usar índices []). No main, leia a matriz do usuário, aplique a função e exiba a matriz resultante.

**Exemplo:**

**OBS:** Azul = principal; Amarelo = secundária; Verde = principal e secundária.

**Entrada:**

1 2 3

4 5 6

7 8 9

**Saída:**

3 2 1

4 5 6

9 8 7

- 16.** Crie um programa em C que leia uma string (palavra) e utilize ponteiros para contar e exibir o número de consoantes presentes nessa palavra. O programa não deve usar índices para percorrer a string, apenas ponteiros. Considere letras maiúsculas e minúsculas e descarte caracteres que não sejam letras (A-Z, a-z).

**Exemplo:**

**Entrada:**

Programação

**Saída:**

6