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

8º ATIVIDADE – PONTEIROS

1. Faça um programa em linguagem C que implemente um **"kit de utilidades matemáticas"** para comparar, modificar e calcular valores inteiros armazenados em uma **struct**.

Struct obrigatória:

- **Struct Numeros:**

- (int) a;
- (int) b;

Funções obrigatórias:

- void maxValor(Numeros *n*, int **resultado*): Calcula e armazena, em resultado, o maior valor entre a e b da struct Numeros.
- void minValor(Numeros *n*, int **resultado*): Calcula e armazena, em resultado, o menor valor entre a e b da struct Numeros.
- int **maxValorPonteiro*(Numeros *n*): Retorna o **endereço do maior valor** entre a e b.
- int **minValorPonteiro*(Numeros *n*): Retorna o **endereço do menor valor** entre a e b.
- void aplicarAumentoPercentual(int **valor*, float **percentual*, float **resultado*): Calcula e armazena em resultado o valor original acrescido de um aumento percentual. Esta função deve ser chamada separadamente para a e b da struct Numeros. O parâmetro valor deve receber o endereço de a ou b (ex: &n.a ou &n.b).
- void aplicarDescontoPercentual(int **valor*, float **percentual*, float **resultado*): Calcula e armazena em resultado o valor original reduzido de um desconto percentual. Esta função deve ser chamada separadamente para a e b da struct Numeros, conforme necessário. O parâmetro valor deve receber o endereço de a ou b (ex: &n.a ou &n.b).

- *void *somarValores(Numeros n, int *soma): Soma os valores de a e b dentro da struct Numeros.*

Exemplo de entrada e saída:

Valores iniciais:

a = 20

b = 10

Percentual = 10.0

Maior valor (maxValor): 20

Menor valor (minValor): 10

Maior valor (maxValorPonteiro): : 0x7ffd42a3c4ac "endereço de memória de 20"

Menor valor (minValorPonteiro): 0x7ffd42a3c4b0 "endereço de memória de 10"

Aplicando aumento de 10.00% em a:

(Chamada: aplicarAumentoPercentual(&n.a, 10.0, &resultado))

Resultado: 22.00

Aplicando desconto de 10.00% em a:

(Chamada: aplicarDescontoPercentual(&n.a, 10.0, &resultado))

Resultado: 18.00

Somando valores de a e b:

Resultado da soma: 30

2. Faça um programa em linguagem C que simule um **duelo entre dois personagens** utilizando structs, ponteiros, ponteiros para ponteiros e funções. O jogo deve permitir que os personagens se ataquem alternadamente, aplicando o cálculo de dano (ataque - defesa). O combate segue até que um dos personagens tenha **vida igual ou menor que zero**, sendo declarado derrotado.

Além do combate, o programa deve:

- Permitir a **troca de atributos** (ataque e defesa) entre os personagens.
- Mostrar qual personagem possui o **maior ataque**.
- Exibir o status dos personagens durante a execução.

Struct obrigatória:

- **Struct Personagem:**

- (char) nome;
- (int) vida;
- (int) ataque;
- (int) defesa;

Funções obrigatórias:

- void atacar(Personagem *atacante, Personagem *defensor): Aplica o dano (ataque - defesa) ao defensor.
- void statusPersonagem(Personagem *p): Exibe o status (nome, vida, ataque, defesa) de um personagem.
- void trocarAtributos(Personagem *p1, Personagem *p2): Troca os valores de ataque e defesa entre dois personagens.
- int* maiorAtaque(Personagem *p1, Personagem *p2): Retorna o ponteiro para o atributo 'ataque' do personagem com maior ataque.
- int verificarVencedor(Personagem *p1, Personagem *p2): Verifica se um personagem foi derrotado (vida <= 0) e retorna 1 se houver um vencedor, 0 caso contrário.

Exemplo de entrada e saída:

Status inicial dos personagens:

Nome: Guerreiro | Vida: 100 | Ataque: 30 | Defesa: 10

Nome: Mago | Vida: 80 | Ataque: 25 | Defesa: 5

Guerreiro ataca Mago!

Mago ataca Guerreiro!

Status após a rodada:

Nome: Guerreiro | Vida: 85 | Ataque: 30 | Defesa: 10

Nome: Mago | Vida: 55 | Ataque: 25 | Defesa: 5

...

Guerreiro ataca Mago!

Mago foi derrotado!

Guerreiro é o VENCEDOR!

Trocando atributos de ataque e defesa...

Status após troca de atributos:

Nome: Guerreiro | Vida: 85 | Ataque: 25 | Defesa: 5

Nome: Mago | Vida: 0 | Ataque: 30 | Defesa: 10

O personagem com maior ataque agora é: Mago (30)

3. Considere o código:

```
1  #include <stdio.h>
2
3  int main() {
4      int numero = 50;
5      int *ponteiro = &numero;
6
7      printf("Valor de numero: %d\n", numero);
8      printf("Endereco de numero: %p\n", (void*)&numero);
9      printf("Valor de ponteiro: %p\n", (void*)ponteiro);
10     printf("Conteudo apontado por ponteiro: %d\n", *ponteiro);
11
12     *ponteiro = 100;
13
14     printf("\nNovo valor de numero: %d\n", numero);
15
16     return 0;
17 }
```

Preencha as tabelas:

Estado inicial:

Endereço	Variável	Conteúdo
#1000	Int numero	
#1004	Int *ponteiro	

Estado após ***ponteiro = 100**:

Endereço	Variável	Conteúdo
#1000	Int numero	
#1004	Int *ponteiro	

4. Considere o código:

```
1  #include <stdio.h>
2
3  typedef struct {
4      int codigo;
5      float preco;
6  } Produto;
7
8  void atualizar_produto(Produto **pp) {
9      (**pp).codigo += 100;
10     (*pp)->preco *= 1.1;
11 }
12
13 int main() {
14     Produto p;
15     p.codigo = 25;
16     p.preco = 99.90;
17
18     Produto *ptr = &p;
19
20     printf("Antes: Código=%d, Preço=%.2f\n", p.codigo, p.preco);
21     atualizar_produto(&ptr);
22     printf("Depois: Código=%d, Preço=%.2f\n", ptr->codigo, ptr->preco);
23
24     return 0;
25 }
```

Preencha as tabelas:

Estado inicial:

Endereço	Variável	Conteúdo
#2000	p.codigo	
#2004	p.preco	
#2008	ptr	

Estado após `atualizar_produto(&ptr)`:

Endereço	Variável	Conteúdo
#2000	p.codigo	
#2004	p.preco	
#2008	ptr	

5. Crie um programa em linguagem C que receba do usuário uma quantidade de tempo expressa em segundos e utilize ponteiros para converter esse valor em horas, minutos e segundos. O programa deve exibir o resultado da conversão no formato X horas, Y minutos e Z segundos.

Utilize o seguinte protótipo de função para realizar a conversão:

```
void converterTempo(int totalSegundos, int *horas, int *minutos, int *segundos)
```

Exemplo:

Entrada: 34567 segundos

Saída: 9 horas, 36 minutos e 7 segundos.

6. Em seguida, utilize uma função para contar quantas vezes uma temperatura específica aparece no vetor, utilizando ponteiros para a contagem.

Protótipo da função:

```
void contaTemperaturas(float temperaturas[], int n, float temp, int *count);
```

Especificações:

- O programa deve ler 10 temperaturas e armazená-las em um vetor.
 - O programa deve então solicitar ao usuário uma temperatura para ser procurada.
 - O programa deve exibir a quantidade de vezes que a temperatura solicitada aparece no vetor.
7. Implemente uma função que calcule área da superfície e o volume de uma esfera de **raio R**, utilizando ponteiros. **Essa função deve obedecer ao protótipo:**

```
void calc_esfera(float R, float *area, float *volume)
```

A área da superfície e o volume são dados, respectivamente, por:

$$\mathbf{\acute{A}rea} = 4 \times \pi \times R^2$$

$$\mathbf{Volume} = (4/3) \times \pi \times R^3$$

Exemplo:

Entrada: Valor do raio = 6

Saída: Área: 452,38

Volume: 904,77