Lista de Laboratório – Funções, Ponteiros e Aritmética de Ponteiros em C

Os alunos do segundo semestre do curso superior de Sistemas de Informação estão estudando linguagem C, funções com parâmetros por valor e referência, ponteiros e aritmética de ponteiros. Esta lista contém 10 exercícios de nível avançado para serem realizados em laboratório, visando reforçar o uso de ponteiros nas operações sobre vetores, strings e estruturas.

1. Normalização de Vetor (somente aritmética de ponteiros)

Implemente uma função que normaliza um vetor de números reais dividindo cada elemento pela soma total. A função deve usar apenas aritmética de ponteiros (sem v[i]). Leia n, os n valores, normalize e imprima.

2. Mínimo, Máximo e Média por Referência

Implemente uma função estatisticas(double *v, int n, double *min, double *max, double *media) que percorre o vetor apenas uma vez, usando ponteiros, e devolve mínimos, máximos e média por referência.

3. Compactação de Vetor (Remover Negativos In-place)

Implemente compactar(int *v, int n) que remove números negativos sem usar vetor auxiliar, usando dois ponteiros. Retorne o novo tamanho.

4. Matrizes com Ponteiro para Ponteiro

Implemente funções para ler, imprimir e transpor uma matriz int usando ponteiro para ponteiro, com aritmética de ponteiros, sem vetor auxiliar.

5. Função split de String

Implemente split(char *str, char **tokens, int max_tokens) que substitui espaços por '\0' e armazena ponteiros para o início de cada palavra, sem usar strtok.

6. Ordenação de Vetor de Structs

Dada a struct Aluno {char nome[50]; float nota;}, implemente ordenar_alunos(Aluno *inicio, Aluno *fim) usando ponteiros para ordenar por nota (decrescente), sem usar [].

7. max_element Genérico com void*

Implemente max_element(void *base, int n, int size, int (*cmp)(const void*,const void*)) usando ponteiros genéricos e teste com vetores de int e double.

8. Espelhamento de String In-place

Implemente inverter_string(char *str) usando dois ponteiros para inverter a string in-place, sem usar [].

9. Soma Parcial (Prefix Sum)

Implemente soma_parcial(const int *entrada, int *saida, int n) usando apenas aritmética de ponteiros para calcular a soma acumulada.

10. Rotação de Vetor à Direita

Implemente rotacionar_direita(int *v, int n, int k) sem vetor auxiliar, usando ponteiros e a técnica das três inversões.