# Alocação Dinâmica com Ponteiros em C

## O que é Alocação Dinâmica?

A alocação dinâmica de memória em C permite reservar espaço na memória durante a execução do programa (em tempo de execução), ao invés de definir o tamanho fixo em tempo de compilação.

Ela é especialmente útil quando não sabemos antecipadamente quantos elementos serão necessários, como em vetores de tamanho variável ou estruturas complexas como listas ligadas.

## Funções de Alocação Dinâmica (stdlib.h)

1. malloc() – Aloca um bloco de memória e retorna um ponteiro para o início.

2. calloc() – Aloca um bloco de memória e inicializa os valores com zero.

3. realloc() – Realoca um bloco de memória já alocado para um novo tamanho.

4. free() – Libera um bloco de memória previamente alocado.

## Exemplo 1: Alocação de um vetor com malloc

#include <stdio.h>

#include <stdlib.h>

int main() {

int \*vetor;

int n;

printf("Digite o tamanho do vetor: ");

scanf("%d", &n);

vetor = (int \*) malloc(n \* sizeof(int)); // Alocação

if (vetor == NULL) {

printf("Erro de alocação de memória!\n");

return 1;

}

for (int i = 0; i < n; i++) {

printf("Digite o valor para vetor[%d]: ", i);

scanf("%d", &vetor[i]);

}

printf("Valores armazenados:\n");

for (int i = 0; i < n; i++) {

printf("%d ", vetor[i]);

}

free(vetor); // Libera memória

return 0;

}

## Exemplo 2: Alocação com calloc

int \*vetor = (int \*) calloc(n, sizeof(int));

// Diferente do malloc, calloc inicializa todos os valores com zero.

## Exemplo 3: Reallocação de memória com realloc

vetor = (int \*) realloc(vetor, novo\_tamanho \* sizeof(int));

// realloc permite expandir ou reduzir o bloco de memória

## Cuidados Importantes

- Sempre verificar se a alocação retornou NULL (falha).

- Após o uso, liberar a memória com free().

- Evite vazamentos de memória (memory leaks) não liberando os ponteiros alocados.

- Nunca acessar memória após o free().

## Resumo

- Alocação dinâmica é essencial para programas eficientes e flexíveis.

- malloc, calloc, realloc e free são funções fundamentais.

- Sempre liberar memória para evitar desperdício ou falhas.