Vetores, Matrizes e Funções Recursivas em C: Fundamentos de Estruturas de Dados

Uma jornada pelos elementos fundamentais da programação estruturada em C, explorando como armazenar, manipular e processar dados de forma eficiente usando técnicas que são a base de algoritmos avançados.

Parte 1: Fundamentos de Vetores em C

Vetores são estruturas que armazenam sequências de elementos do mesmo tipo em posições contíguas de memória, formando uma coleção indexada.

1

Declaração

`int vetor[5];` - Cria um vetor de inteiros com 5 posições

2

Acesso

`vetor[0] = 10;` - Atribui o valor 10 à primeira posição (índice 0)

3

Inicialização

`int numeros[5] = {10, 20, 30, 40, 50};`

Manipulação de Vetores: Exemplos Práticos

Preenchimento com Laço

```
for(i=0; i<5; i++) {
    vetor[i] = i+1;
}
```

Leitura de Dados

```
for(i=0; i<5; i++) {
    printf("Digite o valor %d: ",
i+1);
    scanf("%d", &vetor[i]);
}
```

Busca do Maior Elemento

```
int maior = vetor[0];
int posicao = 0;
for(i=1; i<5; i++) {
    if(vetor[i] > maior) {
        maior = vetor[i];
        posicao = i;
    }
}
```

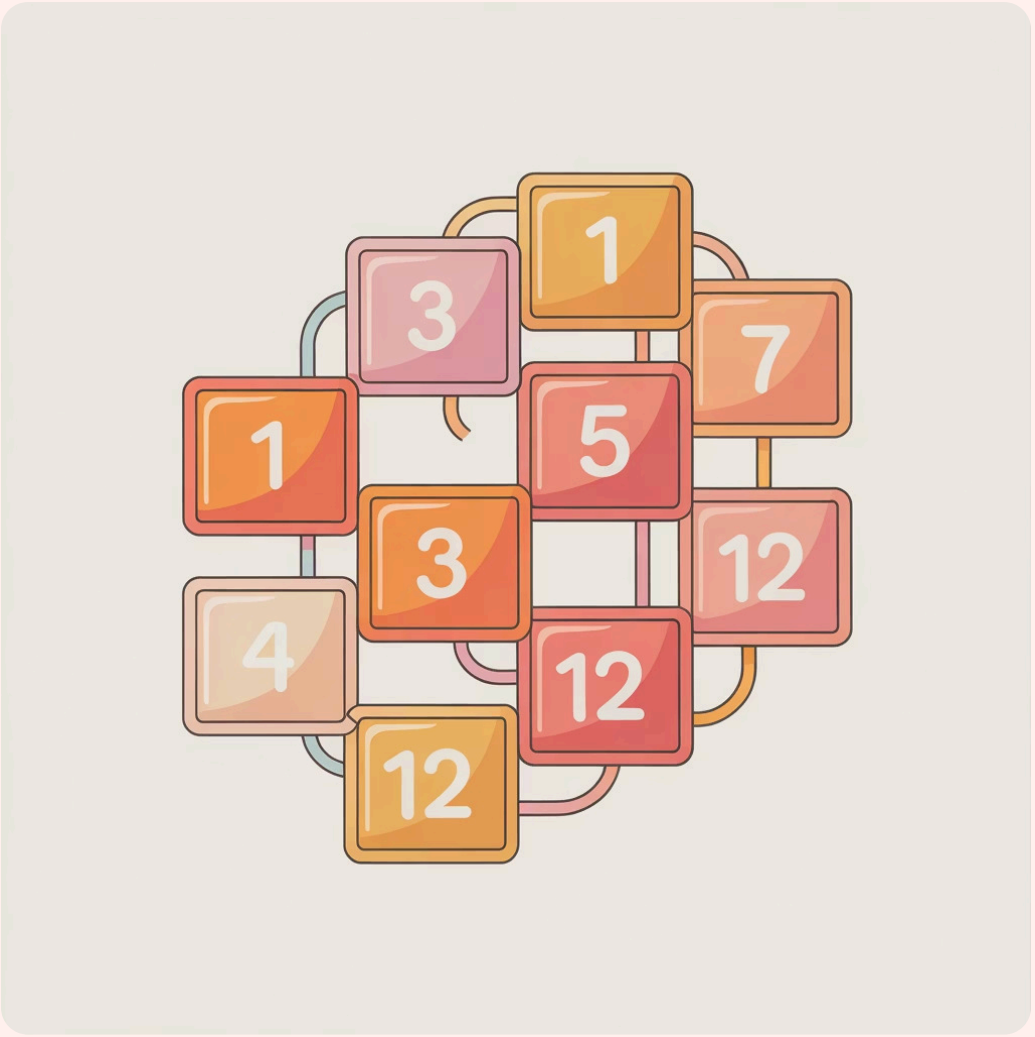
A manipulação eficiente de vetores é essencial para operações como ordenação, busca binária e processamento de dados.

Vetores: Estrutura Linear Simples

Vetores são a mais fundamental estrutura de dados em programação. Cada elemento ocupa um espaço de memória contíguo e pode ser acessado diretamente através de seu índice.

Características:

- Acesso aleatório em O(1)
- Tamanho fixo na declaração
- Armazenamento sequencial



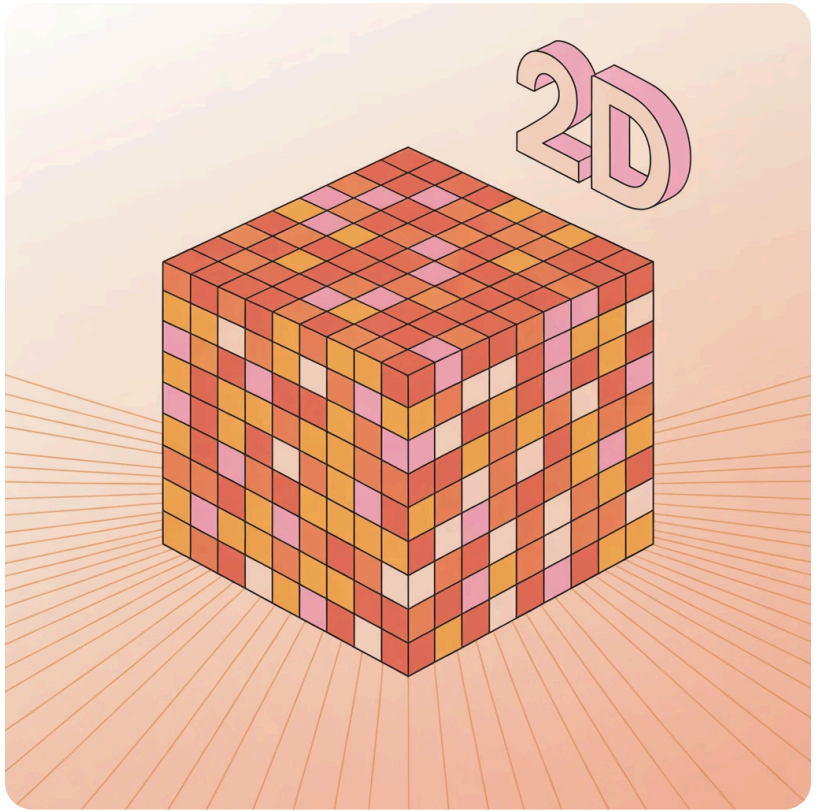
Parte 2: Matrizes em C - Conceitos e Declaração

Matrizes são estruturas bidimensionais que organizam dados em linhas e colunas, formando uma tabela de valores.

```
// Declaração de matriz 3x4
int matriz[3][4];

// Acesso ao elemento (linha 0, coluna 1)
matriz[0][1] = 15;

// Cada linha pode ser tratada como um vetor
int *linha1 = matriz[1];
```



Em memória, as matrizes são armazenadas linearmente, normalmente por linhas (*row-major order*).

Operações Básicas com Matrizes

Preenchimento com Loops Aninhados

```
for(i=0; i<3; i++) {
    for(j=0; j<4; j++) {
        matriz[i][j] = i*4 + j;
    }
}
```

Exibição de Matriz em Formato Tabular

```
for(i=0; i<3; i++) {
    for(j=0; j<4; j++) {
        printf("%3d ", matriz[i][j]);
    }
    printf("\n");
}
```

Operações com Matrizes

Soma de matrizes, multiplicação, transposição e outras operações matemáticas são implementadas usando loops aninhados para percorrer cada elemento.

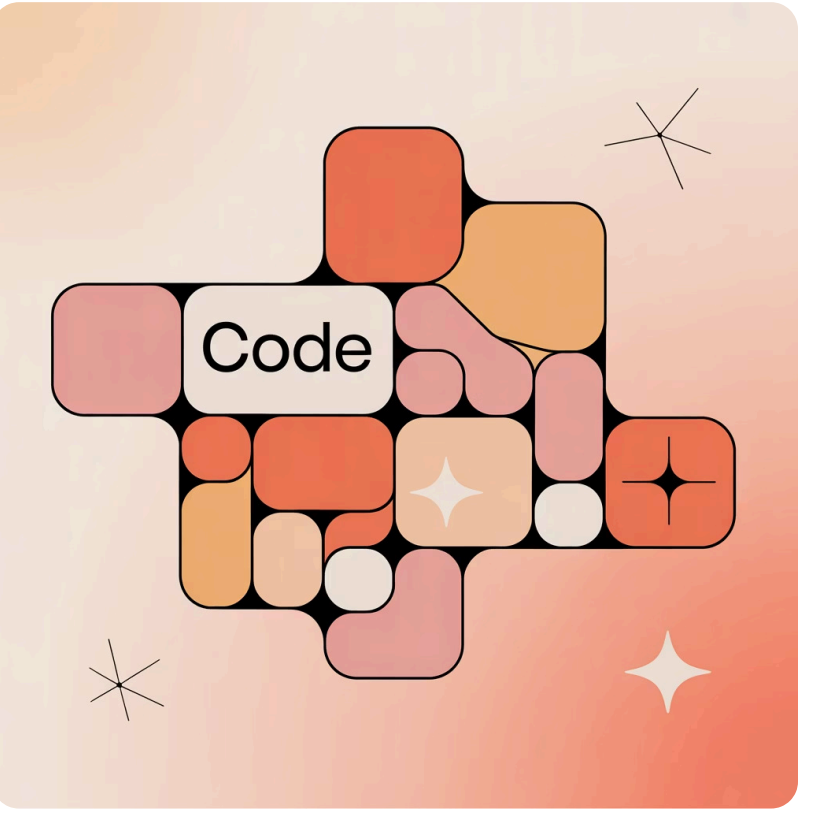
Matrizes são amplamente utilizadas em processamento de imagens, cálculos científicos e representação de grafos.

Funções em C

Funções em C são blocos de código que executam uma tarefa específica, permitindo modularizar programas e promover a reutilização. Elas melhoram a organização, legibilidade e manutenção do código, encapsulando lógicas complexas em unidades menores e gerenciáveis.

Compreender funções é essencial para construir software robusto e eficiente.

- **Modularidade:** Divide o programa em partes gerenciáveis.
- **Reutilização:** Evita duplicação de código.
- **Organização:** Melhora a estrutura e clareza do projeto.
- **Manutenção:** Facilita correções e atualizações.



Exemplos de Funções em C

Para ilustrar a versatilidade das funções em C, vejamos exemplos práticos de como elas recebem parâmetros, retornam valores e interagem com estruturas como vetores.

1

Soma de Valores e Retorno

```
int somar(int a, int b) {
    return a + b;
}

// Uso:
int resultado = somar(5, 3); // resultado = 8
```

Aqui, a função `somar` recebe dois inteiros, executa uma operação e retorna o resultado. Isso demonstra o fluxo básico de entrada e saída de dados.

2

Passando Posição de Vetor (Referência)

```
void dobrarValor(int *valor) {
    *valor *= 2;
}

// Uso:
int meuVetor[] = {10, 20, 30};
dobrarValor(&meuVetor[1]); // meuVetor agora é {10, 40, 30}
```

Ao passar o endereço de memória (ponteiro) de um elemento, a função pode modificar o valor original. Isso é fundamental para manipular vetores e outras estruturas de dados.

A compreensão desses mecanismos de passagem de parâmetros é crucial para escrever código eficiente e sem efeitos colaterais indesejados.

Parte 3: Funções Recursivas em C

Funções recursivas são aquelas que chamam a si mesmas para resolver um problema dividindo-o em casos menores do mesmo tipo.

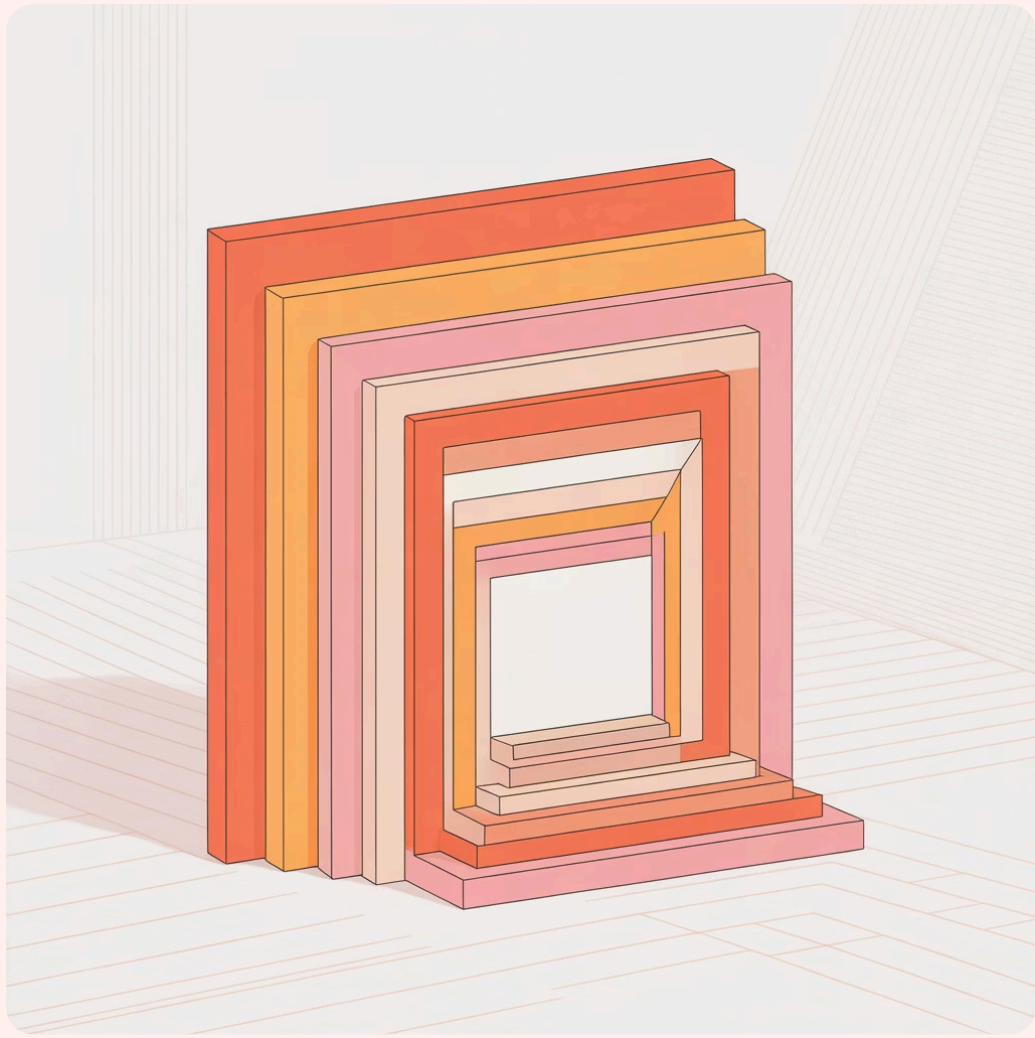
Estrutura Básica

```
tipo_retorno funcao(parametros) {
    // Caso base (condição de parada)
    if(condicao_terminal) {
        return valor;
    }

    // Chamada recursiva com problema menor
    return funcao(parametros_reduzidos);
}
```

Onde Usar Recursão

- Problemas naturalmente recursivos (fatorial, Fibonacci)
- Percurso em estruturas hierárquicas (árvores, grafos)
- Algoritmos "dividir e conquistar" (mergesort, quicksort)
- Backtracking (labirintos, quebra-cabeças)
- Problemas de subconjuntos e permutações



Exemplos de Funções Recursivas em C

Cálculo de Fatorial

```
int fatorial(int n) {
    // Caso base
    if(n <= 1) return 1;

    // Chamada recursiva
    return n * fatorial(n-1);
}
```

Computa n! = n × (n-1) × ... × 2 × 1

Sequência de Fibonacci

```
int fibonacci(int n) {
    if(n <= 1) return n;
    return fibonacci(n-1) +
fibonacci(n-2);
}
```

Gera a sequência: 0, 1, 1, 2, 3, 5, 8, 13...

Busca Binária Recursiva

```
int buscaBinaria(int v[], int inicio,
int fim, int x) {
 if(inicio > fim) return -1;

 int meio = (inicio + fim)/2;

 if(v[meio] == x) return meio;
 if(v[meio] > x)
 return buscaBinaria(v, inicio,
meio-1, x);
 else
 return buscaBinaria(v, meio+1,
fim, x);
}
```