**Inserção em Vetor não ordenado**

#include <stdio.h> // Biblioteca padrão para funções de entrada e saída

#define TAMANHO 10 // Definindo o tamanho máximo do vetor

int main() {

int vetor[TAMANHO]; // Declarando um vetor de inteiros com espaço para 10 elementos

int n = 0; // Variável que guarda quantos elementos já inserimos (inicialmente vazio)

// Inserindo valores

vetor[n++] = 10; // Inserimos 10 na posição 0 e incrementamos n

vetor[n++] = 30; // Inserimos 30 na posição 1 e incrementamos n

vetor[n++] = 20; // Inserimos 20 na posição 2 e incrementamos n

printf("Elementos do vetor:\n"); // Mensagem para identificar os dados que serão exibidos

for (int i = 0; i < n; i++) { // Laço para percorrer os elementos já inseridos

printf("Posição %d: %d\n", i, vetor[i]); // Exibe a posição e o valor correspondente

}

return 0; // Finaliza o programa

}

**Pesquisa Linear em vetor não ordenado**

#include <stdio.h> // Biblioteca padrão para funções de entrada e saída

#define TAMANHO 10 // Define o tamanho máximo do vetor

int main() {

int vetor[TAMANHO] = {10, 30, 20}; // Inicializa o vetor com 3 valores

int n = 3; // Indica que temos 3 elementos no vetor

int valor = 30; // Valor que queremos buscar

int encontrado = -1; // Variável para armazenar a posição do valor (inicialmente -1 = não encontrado)

// Pesquisa Linear: percorre o vetor do começo ao fim

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

if (vetor[i] == valor) { // Se encontrar o valor desejado

encontrado = i; // Guarda a posição onde encontrou

break; // Para o laço (não precisa procurar mais)

}

}

// Mostrando o resultado da pesquisa

if (encontrado != -1) { // Se encontrou o valor

printf("Valor %d encontrado na posição %d.\n", valor, encontrado);

} else { // Se não encontrou

printf("Valor %d não encontrado.\n", valor);

}

return 0; // Finaliza o programa

}

**Exclusão de um elemento no vetor não ordenado**

#include <stdio.h> // Biblioteca padrão para entrada e saída

#define TAMANHO 10 // Define o tamanho do vetor

int main() {

int vetor[TAMANHO] = {10, 30, 20}; // Vetor inicializado com 3 valores

int n = 3; // Número de elementos no vetor

int valor = 30; // Valor que queremos excluir

int posicao = -1; // Variável para armazenar a posição do valor a excluir (inicialmente -1 = não encontrado)

// Pesquisa para encontrar a posição do valor

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

if (vetor[i] == valor) { // Se o valor encontrado for igual ao procurado

posicao = i; // Armazena a posição

break; // Para o laço

}

}

// Verifica se encontrou o valor para excluir

if (posicao != -1) {

// Desloca os elementos após a posição encontrada uma casa para a esquerda

for (int i = posicao; i < n - 1; i++) {

vetor[i] = vetor[i + 1]; // Move o próximo valor para a posição atual

}

n--; // Decrementa o número de elementos no vetor

printf("Valor %d excluído.\n", valor); // Informa que a exclusão foi feita

} else {

printf("Valor %d não encontrado.\n", valor); // Informa que o valor não foi encontrado

}

// Exibe o vetor atualizado

printf("Elementos do vetor após exclusão:\n");

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

printf("Posição %d: %d\n", i, vetor[i]);

}

return 0; // Finaliza o programa

}

**Array estático**

#include <stdio.h> // Biblioteca padrão para entrada e saída

int main() {

int vetor[5]; // Declara um vetor de 5 inteiros

int n = 0; // Variável para controlar o número de elementos inseridos

vetor[n++] = 10; // Insere 10 na posição 0 e incrementa n

vetor[n++] = 40; // Insere 40 na posição 1 e incrementa n

vetor[n++] = 20; // Insere 20 na posição 2 e incrementa n

printf("Vetor não ordenado (Array Estático):\n"); // Exibe mensagem antes de mostrar o vetor

for (int i = 0; i < n; i++) { // Percorre os elementos já inseridos

printf("Posição %d: %d\n", i, vetor[i]); // Mostra o índice e o valor armazenado

}

return 0; // Encerra o programa

}

**Array dinâmico (malloc)**

#include <stdio.h> // Biblioteca padrão para entrada e saída

#include <stdlib.h> // Biblioteca para alocação dinâmica de memória

int main() {

int \*vetor; // Declara um ponteiro para inteiro

int n = 0; // Número de elementos atualmente no vetor

int tamanho = 5; // Tamanho máximo inicial para o vetor

vetor = malloc(tamanho \* sizeof(int)); // Aloca memória para 5 inteiros

vetor[n++] = 15; // Insere 15 na posição 0

vetor[n++] = 25; // Insere 25 na posição 1

vetor[n++] = 5; // Insere 5 na posição 2

printf("Vetor não ordenado (Array Dinâmico):\n"); // Mensagem inicial

for (int i = 0; i < n; i++) { // Percorre os elementos inseridos

printf("Posição %d: %d\n", i, vetor[i]); // Exibe a posição e o valor

}

free(vetor); // Libera a memória alocada dinamicamente

return 0; // Finaliza o programa

}

**Lista encadeada**

#include <stdio.h> // Biblioteca padrão de entrada e saída

#include <stdlib.h> // Biblioteca para funções de alocação dinâmica

// Definição da estrutura do nó

typedef struct No {

int valor; // Valor armazenado no nó

struct No \*proximo; // Ponteiro para o próximo nó

} No;

// Função para inserir um novo nó no início da lista

void inserir\_inicio(No \*\*inicio, int valor) {

No \*novo = malloc(sizeof(No)); // Aloca memória para o novo nó

novo->valor = valor; // Define o valor do novo nó

novo->proximo = \*inicio; // Faz o novo nó apontar para o antigo primeiro nó

\*inicio = novo; // Atualiza o início da lista

}

// Função para imprimir a lista

void imprimir(No \*inicio) {

No \*atual = inicio; // Começa pelo primeiro nó

printf("Lista: ");

while (atual != NULL) { // Enquanto houver nós

printf("%d -> ", atual->valor); // Imprime o valor

atual = atual->proximo; // Avança para o próximo nó

}

printf("NULL\n"); // Fim da lista

}

int main() {

No \*lista = NULL; // Inicializa a lista como vazia

// Inserções

inserir\_inicio(&lista, 30);

inserir\_inicio(&lista, 20);

inserir\_inicio(&lista, 10);

imprimir(lista); // Imprime a lista encadeada

return 0;

}

**Inserção em vetor ordenado**

#include <stdio.h> // Biblioteca padrão de entrada e saída

#define TAM 10 // Tamanho máximo do vetor

int main() {

int vetor[TAM] = {10, 20, 30, 50}; // Vetor já preenchido com 4 elementos ordenados

int n = 4; // Número atual de elementos no vetor

int novo = 25; // Valor que queremos inserir

int i = n - 1; // Começamos pelo último elemento preenchido

// Etapa 1: Encontrar onde o novo valor deve ser inserido

while (i >= 0 && vetor[i] > novo) { // Enquanto o valor atual for maior que o novo

vetor[i + 1] = vetor[i]; // Empurra o valor para a direita

i--; // Move para o elemento anterior

}

vetor[i + 1] = novo; // Insere o novo valor na posição correta

n++; // Atualiza o número de elementos

// Etapa 2: Mostrar o vetor atualizado

printf("Vetor após inserção: ");

for (i = 0; i < n; i++) { // Percorre o vetor atualizado

printf("%d ", vetor[i]); // Imprime cada elemento

}

printf("\n");

return 0; // Finaliza o programa

}

**Pesquisa Linear em Vetor Ordenado**

#include <stdio.h> // Biblioteca padrão de entrada e saída

#define TAM 6 // Define o tamanho do vetor

int main() {

int vetor[TAM] = {5, 10, 15, 20, 30, 50}; // Vetor já ordenado

int valor = 20; // Valor que queremos procurar

int i; // Variável de controle do laço

int encontrado = -1; // Variável para guardar a posição encontrada (-1 = não encontrado)

// Vamos percorrer o vetor

for (i = 0; i < TAM; i++) {

printf("Comparando com posição %d: %d\n", i, vetor[i]); // Mostra a posição atual e o valor

if (vetor[i] == valor) { // Se encontramos o valor

encontrado = i; // Armazenamos a posição

printf("Valor %d encontrado na posição %d\n", valor, i);

break; // Paramos a busca

} else if (vetor[i] > valor) { // Se o valor atual é maior que o procurado

printf("Parando: valor %d não está no vetor.\n", valor);

break; // Podemos parar porque o vetor é ordenado

}

}

// Se não encontrou o valor

if (encontrado == -1) {

printf("Valor %d não encontrado no vetor.\n", valor);

}

return 0; // Finaliza o programa

}

**Exclusão em Vetor Ordenado**

#include <stdio.h> // Biblioteca padrão de entrada e saída

#define TAM 10 // Tamanho máximo do vetor

int main() {

int vetor[TAM] = {5, 10, 20, 25, 30}; // Vetor ordenado

int n = 5; // Número atual de elementos

int valor = 20; // Valor que queremos excluir

int i, j; // Variáveis para controle dos laços

int encontrado = -1; // Guarda a posição onde o valor foi encontrado

// Etapa 1: Procurar o valor no vetor

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

printf("Comparando com posição %d: %d\n", i, vetor[i]); // Mostra cada comparação

if (vetor[i] == valor) { // Se encontramos o valor

encontrado = i; // Guardamos a posição

printf("Valor %d encontrado na posição %d\n", valor, i);

break; // Paramos a busca

} else if (vetor[i] > valor) { // Se o valor atual for maior, não precisa continuar

printf("Valor não encontrado (interrompido pela ordenação).\n");

break;

}

}

// Etapa 2: Se encontramos, removemos o valor

if (encontrado != -1) {

// Deslocar todos os elementos seguintes para a esquerda

for (j = encontrado; j < n - 1; j++) {

vetor[j] = vetor[j + 1]; // Puxa o elemento seguinte para a posição atual

}

n--; // Decrementa o número de elementos

printf("Valor %d removido do vetor.\n", valor);

} else {

printf("Valor %d não encontrado no vetor.\n", valor);

}

// Mostrar o vetor atualizado

printf("Vetor após exclusão: ");

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

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

}

printf("\n");

return 0; // Finaliza o programa

}

**Pesquisa Binária (busca rápida)**

#include <stdio.h> // Biblioteca padrão para entrada e saída

#define TAM 6 // Tamanho do vetor

int main() {

int vetor[TAM] = {5, 10, 15, 20, 25, 30}; // Vetor já ordenado

int valor = 20; // Valor que queremos encontrar

int inicio = 0; // Posição inicial da busca

int fim = TAM - 1; // Posição final da busca (último índice)

int meio; // Variável para calcular o meio

int encontrado = -1; // Se encontrar, guardamos a posição; começa com -1

// Enquanto ainda tiver elementos para buscar

while (inicio <= fim) {

meio = (inicio + fim) / 2; // Calcula o meio do vetor

printf("Verificando posição %d: %d\n", meio, vetor[meio]); // Mostra a posição e valor atual

if (vetor[meio] == valor) { // Se achou o valor

encontrado = meio; // Guarda a posição

printf("Valor %d encontrado na posição %d.\n", valor, meio);

break; // Para o laço (não precisa mais buscar)

} else if (vetor[meio] < valor) { // Se o valor no meio é menor

inicio = meio + 1; // Continua buscando na metade direita

} else { // Se o valor no meio é maior

fim = meio - 1; // Continua buscando na metade esquerda

}

}

// Depois da busca

if (encontrado == -1) { // Se não encontrou

printf("Valor %d não encontrado no vetor.\n", valor);

}

return 0; // Finaliza o programa

}