UNOESC
Campus de Videira

RESUMO D MATÉRIA

UNOESC
Campus de Videira

Arrays (Vetores)

- Coleção de elementos em memória contínua.
- Tamanho fixo.
- Acesso rápido pelo índice → O(1).
- **Exemplo:**

```
int numeros[5] = {1, 2, 3, 4, 5};
```

Matrizes (Arrays 2D)

- Arrays com linhas e colunas. Tamanho fixo.
- Usadas para tabelas, imagens, jogos (ex: xadrez).
- **Exemplo:**

```
int matriz[3][3] = { {1,2,3}, {4,5,6}, {7,8,9} };
```

UNOESC
Campus de Videira

Listas

- Coleções dinâmicas (podem crescer).Tamanho fixo.

Tipos:

- Array List → parecido com vector (C++) ou ArrayList (Java).
- Lista Encadeada (Linked List) → elementos ligados por ponteiros.
- Fácil de inserir e remover, mas buscar pode ser lento.
- **Exemplo:**

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct No {
    int valor;
    struct No *proximo;
} No;

void inserirInicio(No **lista, int valor) {
    No *novo = malloc(sizeof(No));
    novo->valor = valor;
    novo->proximo = *lista;
    *lista = novo;
}

void mostrarLista(No *lista) {
    while (lista != NULL) {
        printf("%d -> ", lista->valor);
        lista = lista->proximo;
    }
    printf("NULL\n");
}

int main() {
    No *lista = NULL;
    inserirInicio(&lista, 10);
    inserirInicio(&lista, 20);
    inserirInicio(&lista, 30);

    mostrarLista(lista);
    return 0;
}
```

UNOESC
Campus de Videira

Pilha (Stack – LIFO)

- Estrutura LIFO → Último a entrar, primeiro a sair. Tamanho fixo.

Operações:

- push() → adicionar
- pop() → remover
- peek() → ver o topo
- Usada em: desfazer ações (Ctrl+Z), chamadas de função, verificação de parênteses.
- **Exemplo:**

```c
#include <stdio.h>
#define MAX 5

int pilha[MAX];
int topo = -1;

void push(int valor) {
    if (topo < MAX - 1) {
        pilha[++topo] = valor;
    }
}

int pop() {
    if (topo >= 0) {
        return pilha[topo--];
    }
    return -1; // erro
}

int main() {
    push(10);
    push(20);
    push(30);

    printf("Removendo: %d\n", pop());
    printf("Removendo: %d\n", pop());
}
```

UNOESC
Campus de Videira

Árvores

- Estrutura em forma de hierarquia (pai → filhos).

Tipos:

- Árvore Binária
- Árvore de Busca Binária (BST) → ordenada
- Árvore AVL → auto-balanceada
- Heap → para filas de prioridade
- Usadas em busca, ordenação, sistema de arquivos, inteligência artificial (árvores de decisão).
- **Exemplo:**

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct No {
    int valor;
    struct No *esq, *dir;
} No;

No* novoNo(int valor) {
    No *novo = malloc(sizeof(No));
    novo->valor = valor;
    novo->esq = novo->dir = NULL;
    return novo;
}

int main() {
    No *raiz = novoNo(10);
    raiz->esq = novoNo(5);
    raiz->dir = novoNo(20);

    printf("Raiz: %d, Esquerda: %d, Direita: %d\n", raiz->valor, raiz->esq->valor, raiz->dir->val
}
```

UNOESC
Campus de Videira

Algoritmos de Busca

Busca Linear:

- Sequencial
- O(n)
- Não Precisa tar ordenado

Busca Binária:

- Divisão
- O(log n)
- Precisa tar Ordenado
- **Exemplo:**

```c
int buscaBinaria(int arr[], int n, int alvo) {
    int esquerda = 0, direita = n - 1;
    while (esquerda <= direita) {
        int meio = (esquerda + direita) / 2;
        if (arr[meio] == alvo) return meio;
        if (arr[meio] < alvo) esquerda = meio + 1;
        else direita = meio - 1;
    }
    return -1; // não encontrado
}
```

UNOESC
Campus de Videira

Fila (Queue – FIFO)

Estrutura FIFO → Primeiro a entrar, primeiro a sair. Operações:

Operações:

- enqueue() → inserir
- dequeue() → remover
- Utilizada em: impressoras, agendamento de tarefas, algoritmos BFS, etc.
- **Exemplo:**

```c
#include <stdio.h>
#define MAX 5

int fila[MAX];
int inicio = 0, fim = 0;

void enqueue(int valor) {
    if (fim < MAX) {
        fila[fim++] = valor;
    }
}

Perguntar ao ChatGPT

int dequeue() {
    if (inicio < fim) {
        return fila[inicio++];
    }
    return -1;
}

int main() {
    enqueue(10);
    enqueue(20);
    enqueue(30);

    printf("Atendido: %d\n", dequeue());
    printf("Atendido: %d\n", dequeue());
}
```

UNOESC
Campus de Videira

Tabelas Hash (Hash Tabeles)

- Versão otimizada dos mapas.
- Usa função hash para localizar dados rapidamente.
- Busca média: O(1).
- Usada em bancos de dados, cache, compiladores.
- **Exemplo:**

```c
#include <stdio.h>

int hash(int chave) {
    return chave % 10;  // posições de 0 a 9
}

int main() {
    int tabela[10] = {0};
    int chave = 123;

    tabela[hash(chave)] = 50; // salva na posição calculada
    printf("Valor encontrado: %d\n", tabela[hash(123)]);
}
```

UNOESC
Campus de Videira

Mapas (Dicionários)

- Armazena pares chave → valor.Tamanho fixo.
- Busca rápida usando a chave.
- **Exemplo:**

```c
#include <stdio.h>
#include <string.h>

typedef struct {
    char chave[20];
    int valor;
} Mapa;

int main() {
    Mapa pessoa;
    strcpy(pessoa.chave, "idade");
    pessoa.valor = 25;

    printf("%s = %d\n", pessoa.chave, pessoa.valor);
}
```

UNOESC
Campus de Videira

Bubble Sort

- Compara pares de elementos e vai “subindo” os maiores para o final, como bolhas.
- **Exemplo:**

```c
for(int i = 0; i < n - 1; i++){
    for(int j = 0; j < n - i - 1; j++){
        if(arr[j] > arr[j+1]){
            int temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
        }
    }
}
```

UNOESC
Campus de Videira

Selection Sort

- Procura o menor elemento e coloca na primeira posição, depois na segunda, e assim por diante.
- **Exemplo:**

```c
for(int i = 0; i < n - 1; i++){
    int menor = i;
    for(int j = i + 1; j < n; j++){
        if(arr[j] < arr[menor]) menor = j;
    }
    int temp = arr[i];
    arr[i] = arr[menor];
    arr[menor] = temp;
}
```

UNOESC
Campus de Videira

Insertion Sort

- Insere os elementos um por um em ordem, como organizar cartas na mão.
- **Exemplo:**

```c
for(int i = 1; i < n; i++){
    int chave = arr[i];
    int j = i - 1;
    while(j >= 0 && arr[j] > chave){
        arr[j+1] = arr[j];
        j--;
    }
    arr[j+1] = chave;
}
```

UNOESC
Campus de Videira

Merge Sort *(Dividir e Conquistar)*

- Divide o array no meio, ordena as partes separadas e depois junta.
- **Exemplo:**

```
void mergeSort(int arr[], int inicio, int fim) {
    if(inicio < fim){
        int meio = (inicio + fim) / 2;
        mergeSort(arr, inicio, meio);
        mergeSort(arr, meio + 1, fim);
        merge(arr, inicio, meio, fim);
    }
}
```

UNOESC
Campus de Videira

Quick Sort *(Muito rápido)*

- Escolhe um pivô e separa os menores à esquerda e os maiores à direita.
- **Exemplo:**

```c

void quickSort(int arr[], int inicio, int fim){
    if(inicio < fim){
        int pivo = partition(arr, inicio, fim);
        quickSort(arr, inicio, pivo - 1);
        quickSort(arr, pivo + 1, fim);
    }
}
```

UNOESC
Campus de Videira

Heap Sort *(Usa uma árvore Heap)*

- Transforma o array em heap (estrutura de árvore) e ordena removendo a raiz várias vezes.
- **Exemplo:**

```c
void heapSort(int arr[], int n){
    for(int i = n/2 - 1; i >= 0; i--) heapify(arr, n, i);
    for(int i = n - 1; i > 0; i--){
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;
        heapify(arr, i, 0);
    }
}
```