Algoritmos e Estrutura de Dados

1. Arrays em C

Um array é uma sequência de elementos do mesmo tipo. Exemplo de um vetor de inteiros e cálculo da média:

```c
#include <stdio.h>

int main() {
    int notas[5] = {7, 8, 6, 9, 10};
    int soma = 0;
    for(int i = 0; i < 5; i++) {
        soma += notas[i];
    }
    float media = soma / 5.0;
    printf("Média das notas = %.2f\n", media);
    return 0;
}
```

2. Mini-Projeto: Cadastro de Pessoas

Cadastro de pessoas usando **struct**:

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

struct Pessoa {
    char nome[50];
    int idade;
};

int main() {
    struct Pessoa lista[3];

    for(int i = 0; i < 3; i++) {
        printf("Digite o nome da pessoa %d: ", i+1);
        scanf("%s", lista[i].nome);
        printf("Digite a idade: ");
        scanf("%d", &lista[i].idade);
```

```
    }

    printf("\nLista de pessoas:\n");
    for(int i = 0; i < 3; i++) {
        printf("%s tem %d anos\n", lista[i].nome, lista[i].idade);
    }

    return 0;
}
```

3. Lista Encadeada Simples

Lista encadeada armazena elementos conectados por ponteiros:

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

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

void imprimir(No *inicio) {
    No *atual = inicio;
    while(atual != NULL) {
        printf("%d -> ", atual->valor);
        atual = atual->prox;
    }
    printf("NULL\n");
}

int main() {
    No *inicio = malloc(sizeof(No));
    inicio->valor = 1;
    inicio->prox = malloc(sizeof(No));
    inicio->prox->valor = 2;
    inicio->prox->prox = NULL;

    imprimir(inicio);
    return 0;
}
```

4. Pilha (LIFO)

Pilha segue a regra Último a Entrar, Primeiro a Sair (LIFO):

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

typedef struct Pilha {
    int valor;
    struct Pilha *prox;
} Pilha;

Pilha* push(Pilha *topo, int valor) {
    Pilha *novo = malloc(sizeof(Pilha));
    novo->valor = valor;
    novo->prox = topo;
    return novo;
}

Pilha* pop(Pilha *topo) {
    if(topo == NULL) return NULL;
    Pilha *temp = topo;
    topo = topo->prox;
    free(temp);
    return topo;
}

void imprimir(Pilha *topo) {
    while(topo != NULL) {
        printf("%d\n", topo->valor);
        topo = topo->prox;
    }
}

int main() {
    Pilha *pilha = NULL;
    pilha = push(pilha, 10);
    pilha = push(pilha, 20);
    pilha = push(pilha, 30);
    imprimir(pilha);
    pilha = pop(pilha);
    imprimir(pilha);
    return 0;
}
```

5. Fila (FIFO):

Fila segue a regra Primeiro a Entrar, Primeiro a Sair (FIFO):

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

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

typedef struct {
    No *inicio, *fim;
} Fila;

void enfileirar(Fila *f, int valor) {
    No *novo = malloc(sizeof(No));
    novo->valor = valor;
    novo->prox = NULL;
    if(f->fim != NULL)
        f->fim->prox = novo;
    f->fim = novo;
    if(f->inicio == NULL)
        f->inicio = novo;
}

void desenfileirar(Fila *f) {
    if(f->inicio != NULL) {
        No *temp = f->inicio;
        f->inicio = f->inicio->prox;
        if(f->inicio == NULL) f->fim = NULL;
        free(temp);
    }
}

void imprimir(Fila *f) {
    No *atual = f->inicio;
    while(atual != NULL) {
        printf("%d ", atual->valor);
```

```
        atual = atual->prox;
    }
    printf("\n");
}

int main() {
    Fila f = {NULL, NULL};
    enfileirar(&f, 5);
    enfileirar(&f, 10);
    enfileirar(&f, 15);
    imprimir(&f);
    desenfileirar(&f);
    imprimir(&f);
    return 0;
}
```