Exercícios

João Pedro Epaminondas do Carmo

Básico

Exercício 1 -

```
#include <stdio.h>

int main() {
   int vetor[5];

   printf("Digite 5 números inteiros:\n");
   for(int i = 0; i < 5; i++) {
      printf("Número %d: ", i+1);
      scanf("%d", &vetor[i]);
   }

   printf("\nValores digitados:\n");
   for(int i = 0; i < 5; i++) {
      printf("%d ", vetor[i]);
   }

   return 0;
}
```

Exercício 2 -

```
#include <stdio.h>

int encontrarMaior(int vetor[], int tamanho) {
   int maior = vetor[0];
   for(int i = 1; i < tamanho; i++) {
      if(vetor[i] > maior) {
         maior = vetor[i];
      }
   }
   return maior;
}

int main() {
   int vetor[6] = {10, 45, 23, 67, 89, 34};

   int maior = encontrarMaior(vetor, 6);
   printf("O maior valor no vetor é: %d\n", maior);
```

```
    return 0;
}
```

Exercício 3 -

```
#include <stdio.h>

struct Aluno {
    char nome[40];
    float nota;
};

int main() {
    struct Aluno aluno;

    printf("Digite o nome do aluno: ");
    scanf("%39s", aluno.nome);

    printf("Digite a nota do aluno: ");
    scanf("%f", &aluno.nota);

    printf("\nDados do aluno:\n");
    printf("Nome: %s\nNota: %.2f\n", aluno.nome, aluno.nota);

    return 0;
}
```

Exercício 4 -

```
#include <stdio.h>

struct Produto {
    char nome[40];
    float preco;
};

int main() {
    struct Produto produtos[3];

    for(int i = 0; i < 3; i++) {
        printf("Digite o nome do produto %d: ", i+1);
        scanf("%39s", produtos[i].nome);

        printf("Digite o preço do produto %d: ", i+1);
        scanf("%f", &produtos[i].preco);
```

```
    }

    printf("\nProdutos cadastrados:\n");
    for(int i = 0; i < 3; i++) {
        printf("Produto %d: %s - R$%.2f\n", i+1, produtos[i].nome, produtos[i].preco);
    }

    return 0;
}
```

Exercício 5 -

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

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

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

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

int main() {
    No* inicio = NULL;

    inserirInicio(&inicio, 10);
    inserirInicio(&inicio, 20);
    inserirInicio(&inicio, 30);

    imprimirLista(inicio);
```

```
    return 0;
}
```

Exercício 6 -

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

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

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

int pop(No** topo) {
    if(*topo == NULL) {
        printf("Pilha vazia!\n");
        return -1;
    }

    No* temp = *topo;
    int valor = temp->valor;
    *topo = (*topo)->proximo;
    free(temp);

    return valor;
}

int main() {
    No* topo = NULL;

    push(&topo, 10);
    push(&topo, 20);
    push(&topo, 30);

    printf("Elemento removido: %d\n", pop(&topo));
    printf("Elemento removido: %d\n", pop(&topo));
```

```
    return 0;
}
```

Exercício 7 -

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

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

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

void enqueue(Fila* fila, int valor) {
    No* novo = (No*)malloc(sizeof(No));
    novo->valor = valor;
    novo->proximo = NULL;

    if(fila->fim != NULL) {
        fila->fim->proximo = novo;
    } else {
        fila->inicio = novo;
    }
    fila->fim = novo;
}

int dequeue(Fila* fila) {
    if(fila->inicio == NULL) {
        printf("Fila vazia!\n");
        return -1;
    }

    No* temp = fila->inicio;
    int valor = temp->valor;

    fila->inicio = fila->inicio->proximo;
    if(fila->inicio == NULL) {
        fila->fim = NULL;
    }
```

```
    free(temp);
    return valor;
}

int main() {
    Fila fila = {NULL, NULL};

    enqueue(&fila, 10);
    enqueue(&fila, 20);
    enqueue(&fila, 30);

    printf("Elemento removido: %d\n", dequeue(&fila));
    printf("Elemento removido: %d\n", dequeue(&fila));

    return 0;
}
```

Intermediário

Exercício 1 -

```
#include <stdio.h>

int main() {
    int vetor[10];
    int tamanho = 10;

    printf("Digite 10 números inteiros:\n");
    for(int i = 0; i < 10; i++) {
        printf("Número %d: ", i+1);
        scanf("%d", &vetor[i]);
    }

    // Remover pares
    for(int i = 0; i < tamanho; i++) {
        if(vetor[i] % 2 == 0) {
            // Deslocar elementos
            for(int j = i; j < tamanho-1; j++) {
                vetor[j] = vetor[j+1];
            }
            tamanho--;
            i--; // Verificar novamente a mesma posição
        }
    }
```

```
    printf("\nVetor sem números pares:\n");
    for(int i = 0; i < tamanho; i++) {
        printf("%d ", vetor[i]);
    }

    return 0;
}
```

Exercício 2 -

```
#include <stdio.h>

struct Aluno {
    char nome[40];
    float nota1;
    float nota2;
};

int main() {
    struct Aluno alunos[5];

    for(int i = 0; i < 5; i++) {
        printf("Digite o nome do aluno %d: ", i+1);
        scanf("%39s", alunos[i].nome);

        printf("Digite a nota 1 do aluno %d: ", i+1);
        scanf("%f", &alunos[i].nota1);

        printf("Digite a nota 2 do aluno %d: ", i+1);
        scanf("%f", &alunos[i].nota2);
    }

    printf("\nAlunos com média >= 7.0:\n");
    for(int i = 0; i < 5; i++) {
        float media = (alunos[i].nota1 + alunos[i].nota2) / 2;
        if(media >= 7.0) {
            printf("%s - Média: %.2f\n", alunos[i].nome, media);
        }
    }

    return 0;
}
```

Exercício 3 -

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

```c
#include <stdlib.h>

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

void inserirOrdenado(No** inicio, int valor) {
    No* novo = (No*)malloc(sizeof(No));
    novo->valor = valor;

    // Caso lista vazia ou novo valor menor que o primeiro
    if(*inicio == NULL || valor < (*inicio)->valor) {
        novo->proximo = *inicio;
        *inicio = novo;
        return;
    }

    No* atual = *inicio;
    while(atual->proximo != NULL && atual->proximo->valor < valor) {
        atual = atual->proximo;
    }

    novo->proximo = atual->proximo;
    atual->proximo = novo;
}

void imprimirLista(No* inicio) {
    No* atual = inicio;
    printf("Lista ordenada: ");
    while(atual != NULL) {
        printf("%d ", atual->valor);
        atual = atual->proximo;
    }
    printf("\n");
}

int main() {
    No* inicio = NULL;

    inserirOrdenado(&inicio, 30);
    inserirOrdenado(&inicio, 10);
    inserirOrdenado(&inicio, 20);
    inserirOrdenado(&inicio, 5);
```

```
    imprimirLista(inicio);

    return 0;
}
```

Exercício 4 -

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

typedef struct No {
    char letra;
    struct No* proximo;
} No;

void push(No** topo, char letra) {
    No* novo = (No*)malloc(sizeof(No));
    novo->letra = letra;
    novo->proximo = *topo;
    *topo = novo;
}

char pop(No** topo) {
    if(*topo == NULL) {
        return '\0';
    }

    No* temp = *topo;
    char letra = temp->letra;
    *topo = (*topo)->proximo;
    free(temp);

    return letra;
}

int ehPalindromo(char* palavra) {
    No* pilha = NULL;
    int len = strlen(palavra);

    // Empilhar todas as letras
    for(int i = 0; i < len; i++) {
        push(&pilha, tolower(palavra[i]));
```

```
    }

    // Verificar desempilhando
    for(int i = 0; i < len; i++) {
        if(tolower(palavra[i]) != pop(&pilha)) {
            return 0;
        }
    }

    return 1;
}

int main() {
    char palavra[100];

    printf("Digite uma palavra: ");
    scanf("%s", palavra);

    if(ehPalindromo(palavra)) {
        printf("'%s' é um palíndromo!\n", palavra);
    } else {
        printf("'%s' não é um palíndromo.\n", palavra);
    }

    return 0;
}
```

Exercício 5 -

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

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

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

void enqueue(Fila* fila, int id) {
    No* novo = (No*)malloc(sizeof(No));
    novo->id = id;
```

```
    novo->proximo = NULL;

    if(fila->fim != NULL) {
        fila->fim->proximo = novo;
    } else {
        fila->inicio = novo;
    }
    fila->fim = novo;
}

int dequeue(Fila* fila) {
    if(fila->inicio == NULL) {
        return -1;
    }

    No* temp = fila->inicio;
    int id = temp->id;

    fila->inicio = fila->inicio->proximo;
    if(fila->inicio == NULL) {
        fila->fim = NULL;
    }

    free(temp);
    return id;
}

int main() {
    Fila fila = {NULL, NULL};

    // Inserir 5 clientes
    for(int i = 1; i <= 5; i++) {
        enqueue(&fila, i);
        printf("Cliente %d chegou na fila.\n", i);
    }

    printf("\nAtendendo clientes:\n");
    int cliente;
    while((cliente = dequeue(&fila)) != -1) {
        printf("Atendendo cliente %d\n", cliente);
    }

    return 0;
}
```

Avançado

Desafio 1 -

```
#include <stdio.h>

void ordenarDecrescente(int vetor[], int tamanho) {
   for(int i = 0; i < tamanho-1; i++) {
      for(int j = i+1; j < tamanho; j++) {
         if(vetor[i] < vetor[j]) {
            int temp = vetor[i];
            vetor[i] = vetor[j];
            vetor[j] = temp;
         }
      }
   }
}

int main() {
   int vetor[10];

   printf("Digite 10 números inteiros:\n");
   for(int i = 0; i < 10; i++) {
      printf("Número %d: ", i+1);
      scanf("%d", &vetor[i]);
   }

   ordenarDecrescente(vetor, 10);

   printf("\nVetor ordenado decrescente:\n");
   for(int i = 0; i < 10; i++) {
      printf("%d ", vetor[i]);
   }

   return 0;
}
```

Desafio 2 -

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

typedef struct Aluno {
   char nome[40];
```

```
    float nota;
    struct Aluno* proximo;
} Aluno;

void inserirOrdenado(Aluno** inicio, char nome[], float nota) {
    Aluno* novo = (Aluno*)malloc(sizeof(Aluno));
    strcpy(novo->nome, nome);
    novo->nota = nota;

    // Caso lista vazia ou novo nome vem antes do primeiro
    if(*inicio == NULL || strcmp(nome, (*inicio)->nome) < 0) {
        novo->proximo = *inicio;
        *inicio = novo;
        return;
    }

    Aluno* atual = *inicio;
    while(atual->proximo != NULL && strcmp(nome, atual->proximo->nome) > 0) {
        atual = atual->proximo;
    }

    novo->proximo = atual->proximo;
    atual->proximo = novo;
}

void imprimirLista(Aluno* inicio) {
    Aluno* atual = inicio;
    printf("Lista de alunos:\n");
    while(atual != NULL) {
        printf("Nome: %s, Nota: %.2f\n", atual->nome, atual->nota);
        atual = atual->proximo;
    }
}

int main() {
    Aluno* inicio = NULL;

    inserirOrdenado(&inicio, "Carlos", 8.5);
    inserirOrdenado(&inicio, "Ana", 9.0);
    inserirOrdenado(&inicio, "Bruno", 7.5);

    imprimirLista(inicio);

    return 0;
```

```
}
```

Desafio 3 -

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

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

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

int pop(No** topo) {
    if(*topo == NULL) {
        return -1;
    }

    No* temp = *topo;
    int valor = temp->valor;
    *topo = (*topo)->proximo;
    free(temp);

    return valor;
}

int main() {
    No* topo = NULL;
    int valor, soma = 0;

    printf("Digite 5 valores inteiros:\n");
    for(int i = 0; i < 5; i++) {
        printf("Valor %d: ", i+1);
        scanf("%d", &valor);
        push(&topo, valor);
    }

    printf("\nDesempilhando e somando:\n");
    while((valor = pop(&topo)) != -1) {
```

```
        printf("Valor removido: %d\n", valor);
        soma += valor;
    }

    printf("Somatório: %d\n", soma);

    return 0;
}
```

Desafio 4 -

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

typedef struct Paciente {
    char nome[40];
    int idade;
    struct Paciente* proximo;
} Paciente;

typedef struct Fila {
    Paciente* inicio;
    Paciente* fim;
} Fila;

void enqueue(Fila* fila, char nome[], int idade) {
    Paciente* novo = (Paciente*)malloc(sizeof(Paciente));
    strcpy(novo->nome, nome);
    novo->idade = idade;
    novo->proximo = NULL;

    if(idade >= 60) {
        // Inserir no início (prioridade)
        novo->proximo = fila->inicio;
        fila->inicio = novo;
        if(fila->fim == NULL) {
            fila->fim = novo;
        }
    } else {
        // Inserir no final
        if(fila->fim != NULL) {
            fila->fim->proximo = novo;
        } else {
            fila->inicio = novo;
        }
```

```
        fila->fim = novo;
    }
}

void imprimirFila(Fila* fila) {
    Paciente* atual = fila->inicio;
    printf("Ordem de atendimento:\n");
    while(atual != NULL) {
        printf("Paciente: %s, Idade: %d\n", atual->nome, atual->idade);
        atual = atual->proximo;
    }
}

int main() {
    Fila fila = {NULL, NULL};
    char nome[40];
    int idade;

    for(int i = 0; i < 5; i++) {
        printf("Digite o nome do paciente %d: ", i+1);
        scanf("%39s", nome);

        printf("Digite a idade do paciente %d: ", i+1);
        scanf("%d", &idade);

        enqueue(&fila, nome, idade);
    }

    imprimirFila(&fila);

    return 0;
}
```

Desafio 5 -

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

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

void inserirFinal(No** inicio, int valor) {
```

```
    No* novo = (No*)malloc(sizeof(No));
    novo->valor = valor;

    if(*inicio == NULL) {
        novo->anterior = novo;
        novo->proximo = novo;
        *inicio = novo;
    } else {
        No* ultimo = (*inicio)->anterior;

        novo->proximo = *inicio;
        novo->anterior = ultimo;

        ultimo->proximo = novo;
        (*inicio)->anterior = novo;
    }
}

void removerValor(No** inicio, int valor) {
    if(*inicio == NULL) return;

    No* atual = *inicio;
    do {
        if(atual->valor == valor) {
            if(atual == atual->proximo) { // Único nó
                free(atual);
                *inicio = NULL;
                return;
            }

            atual->anterior->proximo = atual->proximo;
            atual->proximo->anterior = atual->anterior;

            if(atual == *inicio) {
                *inicio = atual->proximo;
            }

            free(atual);
            return;
        }
        atual = atual->proximo;
    } while(atual != *inicio);
}
```

```
void imprimirCircular(No* inicio) {
    if(inicio == NULL) return;

    No* atual = inicio;
    printf("Lista circular: ");
    do {
        printf("%d ", atual->valor);
        atual = atual->proximo;
    } while(atual != inicio);
    printf("\n");
}

int main() {
    No* inicio = NULL;

    inserirFinal(&inicio, 10);
    inserirFinal(&inicio, 20);
    inserirFinal(&inicio, 30);

    imprimirCircular(inicio);

    removerValor(&inicio, 20);
    imprimirCircular(inicio);

    return 0;
}
```