Felipe Barroso de Castro
RA: 2311292
Curso: Engenharia de Software

Lista Encadeada Simples

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

// Estruturas definidas
typedef struct no {
    int dado;
    struct no *prox;
} No;

typedef struct lista {
    No *inicio;
    No *fim;
    unsigned int length;
} Lista;
```

```c
// Função para inserir no início
void inserirInicio(Lista *lista, int valor) {
    No *novo_no = (No *)malloc(sizeof(No));
    if (novo_no == NULL) {
        printf("Erro ao alocar memória\n");
        return;
    }
    novo_no->dado = valor;
    novo_no->prox = lista->inicio;
    lista->inicio = novo_no;

    if (lista->fim == NULL) {
        lista->fim = novo_no;
    }
    lista->length++;
}
```

```c
// Função para inserir no fim
void inserirFim(Lista *lista, int valor) {
    No *novo_no = (No *)malloc(sizeof(No));
    if (novo_no == NULL) {
        printf("Erro ao alocar memória\n");
        return;
    }
    novo_no->dado = valor;
    novo_no->prox = NULL;

    if (lista->fim != NULL) {
        lista->fim->prox = novo_no;
    } else {
        lista->inicio = novo_no;
    }

    lista->fim = novo_no;
    lista->length++;
}
```

```c
// Função para inserir de forma ordenada
void inserirOrdenado(Lista *lista, int valor) {
    No *novo_no = (No *)malloc(sizeof(No));
    if (novo_no == NULL) {
        printf("Erro ao alocar memória\n");
        return;
    }
    novo_no->dado = valor;

    if (lista->inicio == NULL || lista->inicio->dado >= valor) {
        novo_no->prox = lista->inicio;
        lista->inicio = novo_no;
        if (lista->fim == NULL) {
            lista->fim = novo_no;
        }
    } else {
        No *atual = lista->inicio;
        while (atual->prox != NULL && atual->prox->dado < valor) {
            atual = atual->prox;
        }
        novo_no->prox = atual->prox;
        atual->prox = novo_no;
        if (novo_no->prox == NULL) {
            lista->fim = novo_no;
        }
    }
    lista->length++;
}
```

```c
// Função para deletar do início
void deletarInicio(Lista *lista) {
    if (lista->inicio == NULL) {
        printf("Lista está vazia\n");
        return;
    }

    No *temp = lista->inicio;
    lista->inicio = lista->inicio->prox;
    free(temp);

    if (lista->inicio == NULL) {
        lista->fim = NULL;
    }
    lista->length--;
}
```

Lista Duplamente Encadeada

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

// Estruturas definidas
typedef struct no {
    char dado;
    struct no *ant;
    struct no *prox;
} No;

typedef struct listaDEnc {
    No *inicio;
    No *fim;
    unsigned int length;
} ListaDEnc;
```

```c
// Função para deletar do início
void deletarInicio(ListaDEnc *lista) {
    if (lista->inicio == NULL) {
        printf("Lista está vazia\n");
        return;
    }

    No *temp = lista->inicio;
    lista->inicio = lista->inicio->prox;

    if (lista->inicio != NULL) {
        lista->inicio->ant = NULL;
    } else {
        lista->fim = NULL;
    }

    free(temp);
    lista->length--;
}

// Função para deletar do fim
void deletarFim(ListaDEnc *lista) {
    if (lista->fim == NULL) {
        printf("Lista está vazia\n");
        return;
    }

    No *temp = lista->fim;
    lista->fim = lista->fim->ant;

    if (lista->fim != NULL) {
        lista->fim->prox = NULL;
    } else {
        lista->inicio = NULL;
    }

    free(temp);
    lista->length--;
}

```

Funções de Pilha

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

// Estruturas definidas
typedef struct no {
    unsigned int dado;
    struct no *prox;
} No;

typedef struct pilha {
    No *topo;
    unsigned int length;
} Pilha;
```

```c
// Função para retornar o valor do topo
int topo(Pilha *pilha) {
    if (pilha->topo == NULL) {
        return -1;
    }
    return pilha->topo->dado;
}
```

```c
// Função para empilhar
void empilha(Pilha *pilha, unsigned int dado) {
    No *novo_no = (No *)malloc(sizeof(No));
    if (novo_no == NULL) {
        printf("Erro ao alocar memória\n");
        return;
    }
    novo_no->dado = dado;
    novo_no->prox = pilha->topo;
    pilha->topo = novo_no;
    pilha->length++;
}
```

```c
// Função para desempilhar
int pop(Pilha *pilha) {
    if (pilha->topo == NULL) {
        return -1;
    }

    No *temp = pilha->topo;
    int dado = temp->dado;
    pilha->topo = pilha->topo->prox;
    free(temp);
    pilha->length--;

    return dado;
}

```

Funções de Fila

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

// Estruturas definidas
typedef struct no {
    int dado;
    struct no *prox;
} No;

typedef struct fila {
    No *inicio;
    No *fim;
    unsigned int length;
} Fila;
```

```c
// Função para enfileirar
void enfileira(Fila *fila, int valor) {
    No *novo_no = (No *)malloc(sizeof(No));
    if (novo_no == NULL) {
        printf("Erro ao alocar memória\n");
        return;
    }
    novo_no->dado = valor;
    novo_no->prox = NULL;

    if (fila->fim != NULL) {
        fila->fim->prox = novo_no;
    } else {
        fila->inicio = novo_no;
    }

    fila->fim = novo_no;
    fila->length++;
}
```

```c
// Função para desenfileirar
int desenfileira(Fila *fila) {
    if (fila->inicio == NULL) {
        return -1;
    }

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

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

    free(temp);
    fila->length--;

    return valor;
}
```