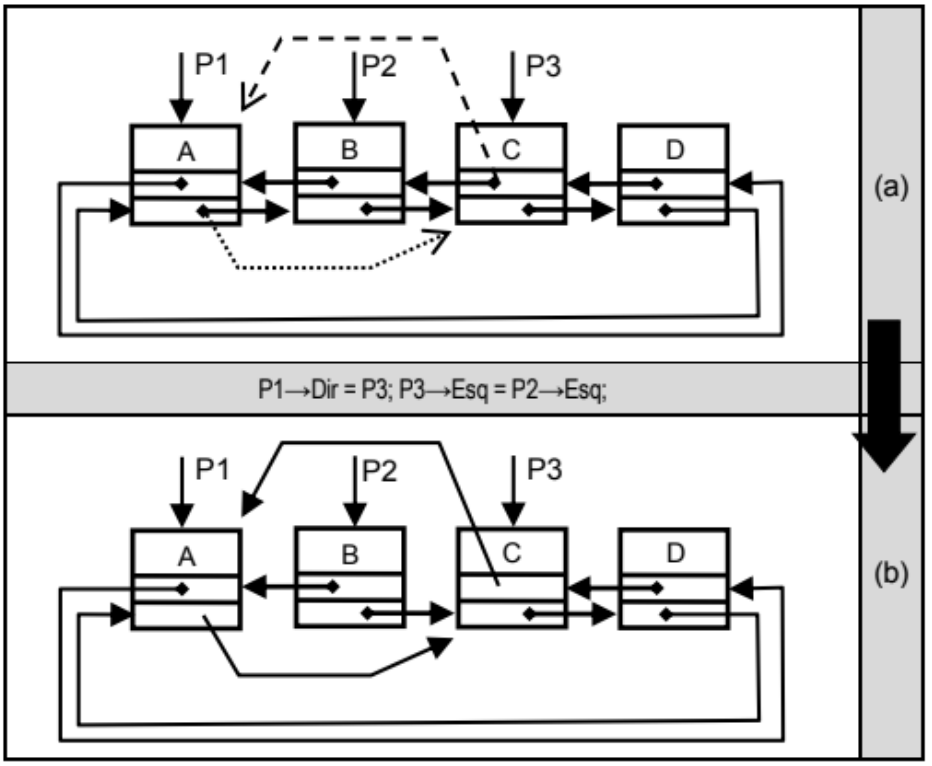
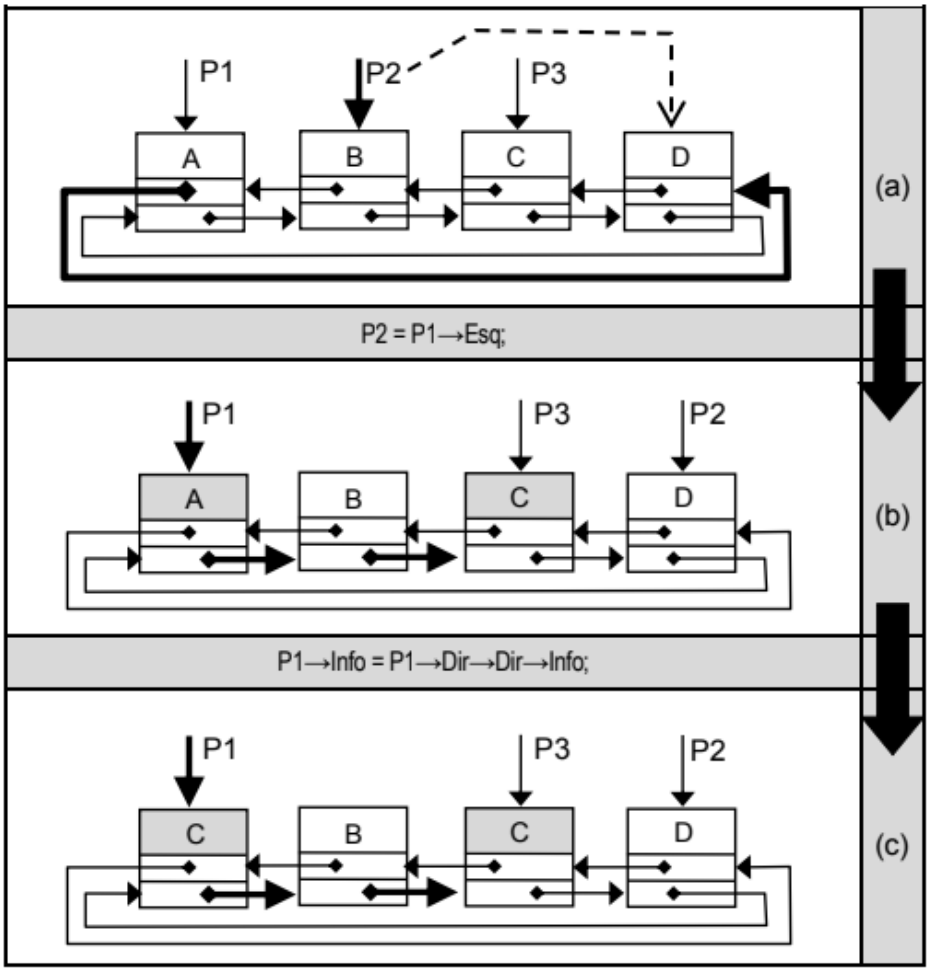
Nome: Matheus Goulart Ranzani
RA: 800278

Diagramas da Lista Duplamente Encadeada





Direita = Anterior e Esquerda = Próximo

Código do arquivo lista_duplamente_encadeadal.h separado em três imagens

Imagem 1

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

// Declaração do Nó
typedef struct Node {
    int info;
    struct Node *anterior, *proximo;
} Node;

// Declaração do Ponteiro de Nó
typedef Node *NodePtr;

// Declaração da Lista Duplamente Encadeada
typedef struct Lista {
    NodePtr inicio, fim;
    int tamanho;
} Lista;

// Função que retorna um ponteiro para uma Lista vazia
Lista *cria() {
    Lista *l;

    l = (Lista *) malloc(sizeof(Lista));

    l->inicio = NULL;
    l->fim = NULL;
    l->tamanho = 0;

    return l;
}

// Função que libera a memória alocada pela Lista
void libera(Lista *l) {
    if (l != NULL) {
        NodePtr aux = l->inicio;

        while (aux != NULL) {
            l->inicio = l->inicio->proximo;
            free(aux);
            aux = l->inicio;
        }

        free(l);
    }
}
```

Imagem 2

```c
// Função que insere um elemento na Lista em ordem crescente
int insere(Lista *l, int x) {
    assert(l != NULL);

    NodePtr p = malloc(sizeof(NodePtr));

    p->info = x;
    p->proximo = NULL;
    p->anterior = NULL;

    NodePtr aux = l->inicio;
    NodePtr anterior = NULL;

    while (aux != NULL && x > aux->info) {
        anterior = aux;
        aux = aux->proximo;
    }

    // Verificação se o elemento já está na Lista
    if (aux != NULL && x == aux->info) {
        free(p);

        return 0;
    }

    // Inserção no início, Lista vazia ou Lista com elementos
    if (anterior == NULL) {
        p->proximo = l->inicio;

        if (l->inicio != NULL) {
            l->inicio->anterior = p;
        } else {
            l->fim = p;
        }

        l->inicio = p;
    } else { // Inserção no meio ou no fim da Lista
        p->proximo = anterior->proximo;
        anterior->proximo = p;

        if (p->proximo != NULL) {
            p->proximo->anterior = p;
        } else {
            l->fim = p;
        }

        p->anterior = anterior;
    }

    return 1;
}
```

Imagem 3

```c
// Função que retira o elemento recebido da Lista, caso ele exista
int retira(Lista *l, int x) {
    assert(l ≠ NULL);

    NodePtr p = l→inicio;
    NodePtr anterior = NULL;

    while (p ≠ NULL && x > p→info) {
        anterior = p;
        p = p→proximo;
    }

    if (p == NULL || p→info ≠ x) {
        return 0;
    }

    if (anterior == NULL) { // Remoção no início da Lista
        l→inicio = l→inicio→proximo;

        if (l→inicio ≠ NULL) {
            l→inicio→anterior = NULL;
        } else {
            l→fim = NULL;
        }

        free(p);
    } else {
        if (p→proximo == NULL) { // Remoção no fim da Lista
            l→fim = p→anterior;
            anterior→proximo = NULL;

            free(p);
        } else { // Remoção no meio da Lista
            anterior→proximo = p→proximo;
            p→proximo→anterior = anterior;

            free(p);
        }
    }

    return 1;
}
```

Código do arquivo usa_lista_duplamente_encadeada.c separado em duas imagens

Imagem 1

```c
// Inclui a Lista Duplamente Encadeada
#include "lista_duplamente_encadeada.h"

#include <stdio.h>

// Função que imprime os elementos da Lista
void imprime(Lista *l) {
    assert(l != NULL);

    NodePtr p = l->inicio;

    printf("\nImprimindo a Lista já ordenada:\n");

    while (p != NULL) {
        printf("%d ", p->info);
        p = p->proximo;
    }

    printf("\n");
}

// Função que imprime os elementos da Lista em ordem inversa
void imprime_inverso(Lista *l) {
    assert(l != NULL);

    NodePtr p = l->fim;

    while (p != NULL) {
        printf("%d ", p->info);
        p = p->anterior;
    }

    printf("\n");
}

// Função que retorna o tamanho atual da Lista
int tamanho_lista(Lista *l) {
    int tamanho = 0;
    NodePtr p = l->inicio;

    while (p != NULL) {
        p = p->proximo;

        tamanho++;
    }

    return tamanho;
}
```

Imagem 2

```c
// Função principal que utiliza a Lista Duplamente Encadeada
int main() {
    Lista *l = cria();

    printf("Adicionando os elementos 8, 3, 19, 21, 2 à Lista...\n");
    insere(l, 8);
    insere(l, 3);
    insere(l, 19);
    insere(l, 21);
    insere(l, 2);
    imprime(l);
    printf("Tamanho atual da Lista: %d\n", tamanho_lista(l));

    printf("\nImprimindo a Lista na ordem inversa\n");
    imprime_inverso(l);

    printf("\nRetirando o elemento 2 da Lista...\n");
    retira(l, 2);
    imprime(l);
    printf("Tamanho atual da Lista: %d\n", tamanho_lista(l));
    printf("\nRetirando o elemento 21 da Lista...\n");
    retira(l, 21);
    imprime(l);
    printf("Tamanho atual da Lista: %d\n", tamanho_lista(l));
    printf("\nRetirando o elemento 8 da Lista...\n");
    retira(l, 8);
    imprime(l);
    printf("Tamanho atual da Lista: %d\n", tamanho_lista(l));

    libera(l);

    return 0;
}
```

Imagem da execução do programa