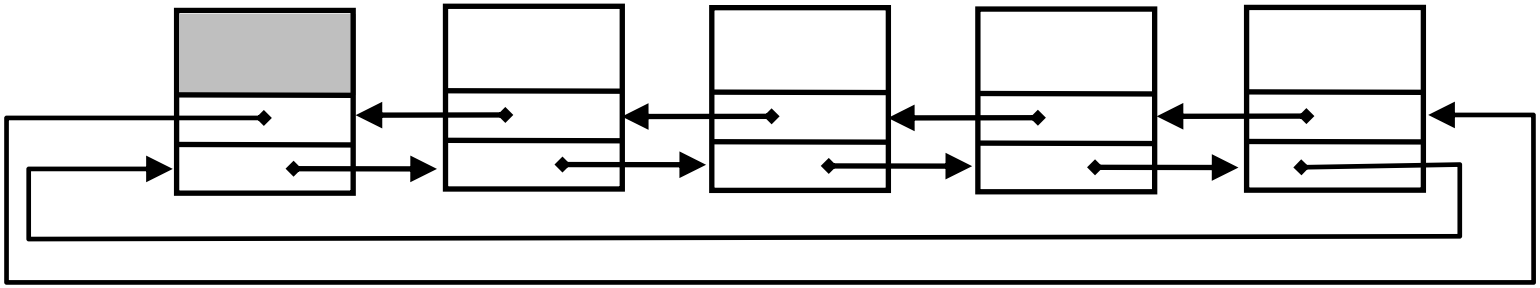
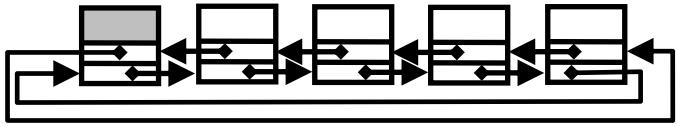
Estruturas de Dados com Jogos



Capítulo 7

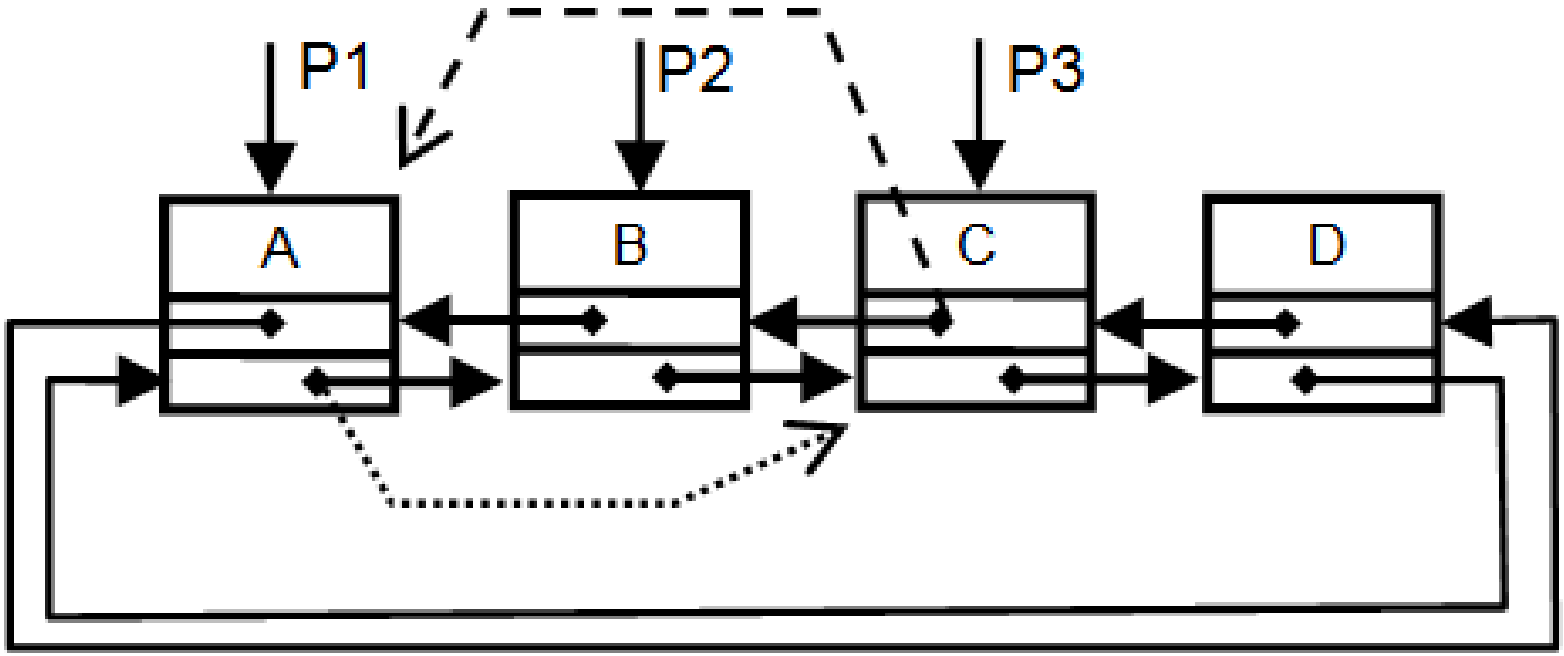
Generalização de Listas Encadeadas

Seus Objetivos neste Capítulo

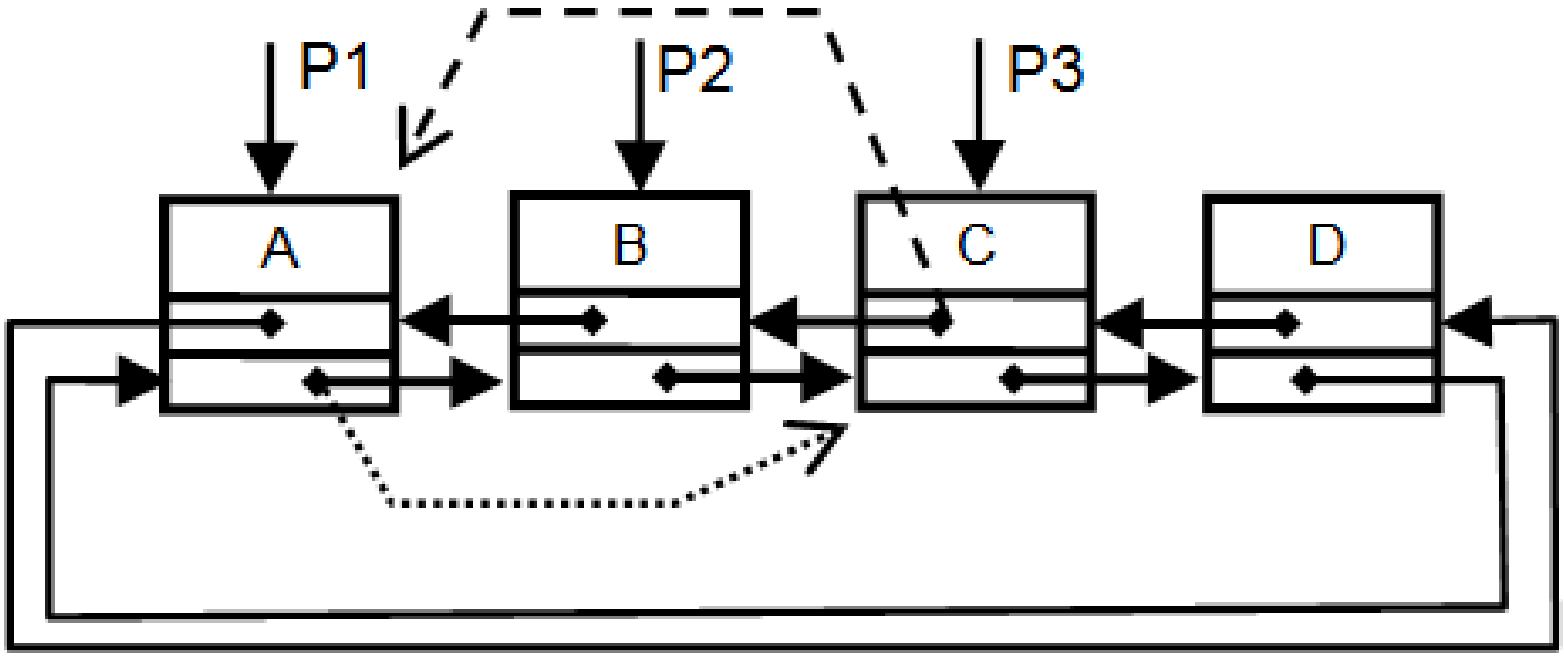


- Estudar técnicas complementares para a implementação de Listas Encadeadas: Encadeamento Duplo, Nó Header, e Primitivas de Baixo Nível;
- Ganhar experiência na elaboração de algoritmos sobre Listas Encadeadas, implementando Pilhas, Filas, Listas Cadastrais e Filas de Prioridades utilizando combinações das técnicas complementares estudadas;
- Conhecer conceitos relativos a generalização de Listas Encadeadas: Listas Multilineares, Listas de Listas, e Listas Genéricas Quanto ao Tipo do Elemento;
- Entender que é possível conceber sua própria estrutura encadeada, para atender a necessidades específicas de determinada aplicação.

Listas Duplamente Encadeadas

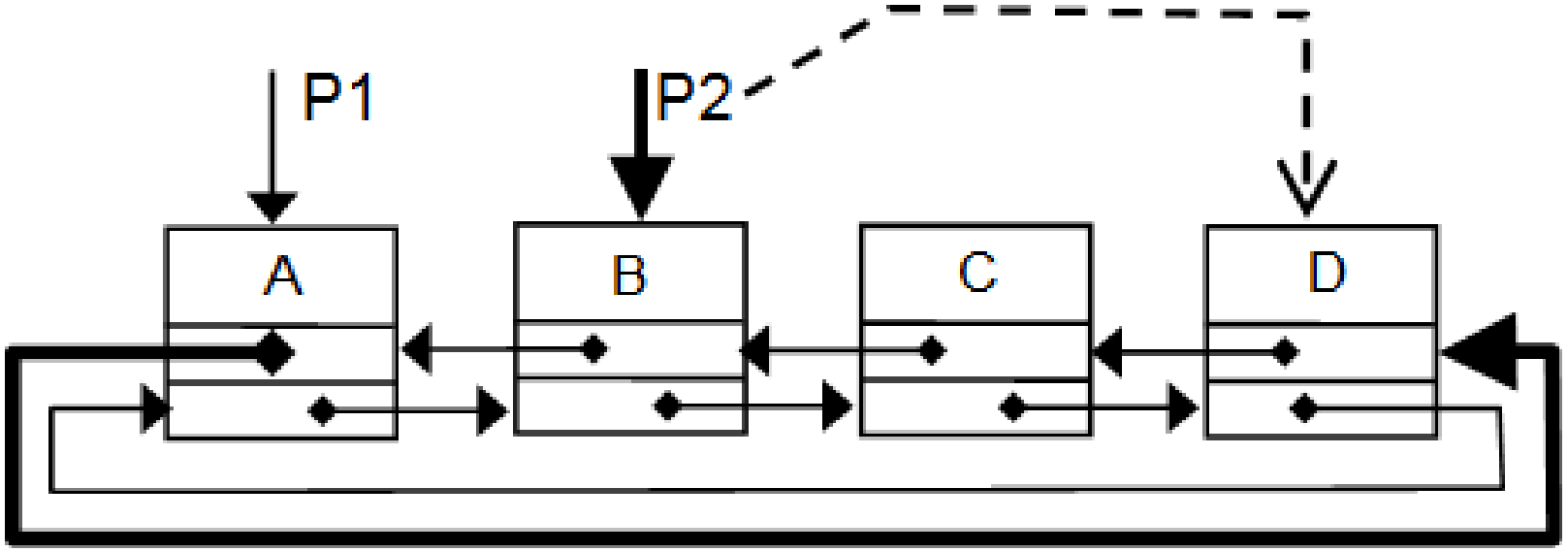


Listas Duplamente Encadeadas

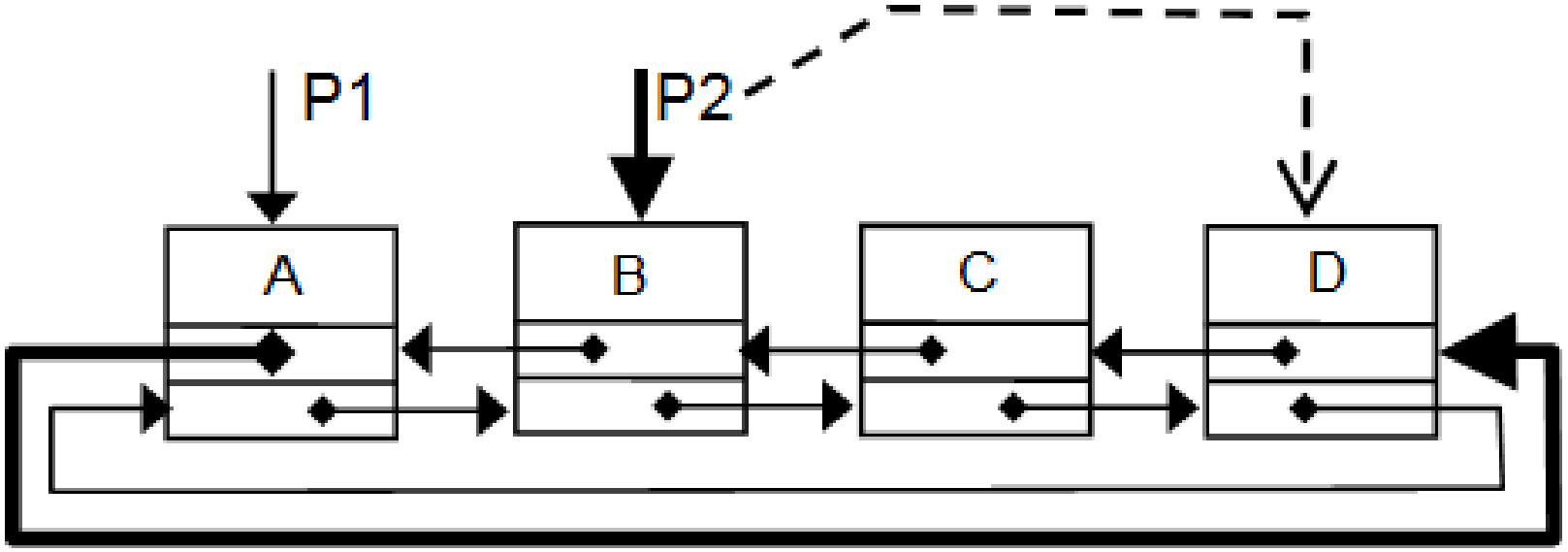


P1→Dir = P3; P3→Esq = P2→Esq;

Listas Duplamente Encadeadas

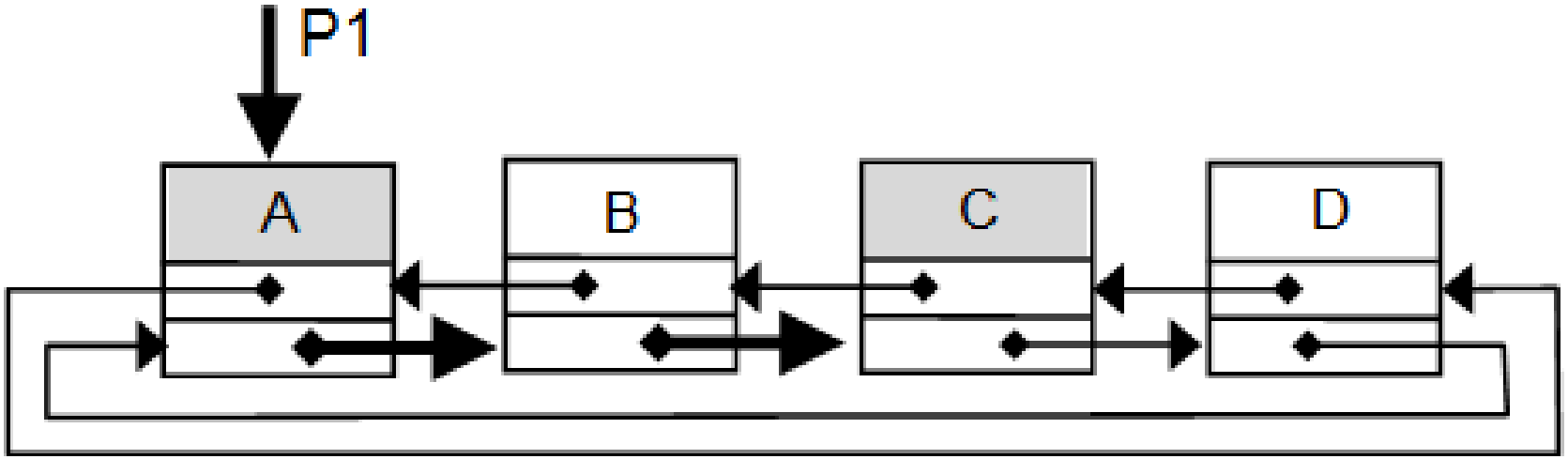


Listas Duplamente Encadeadas

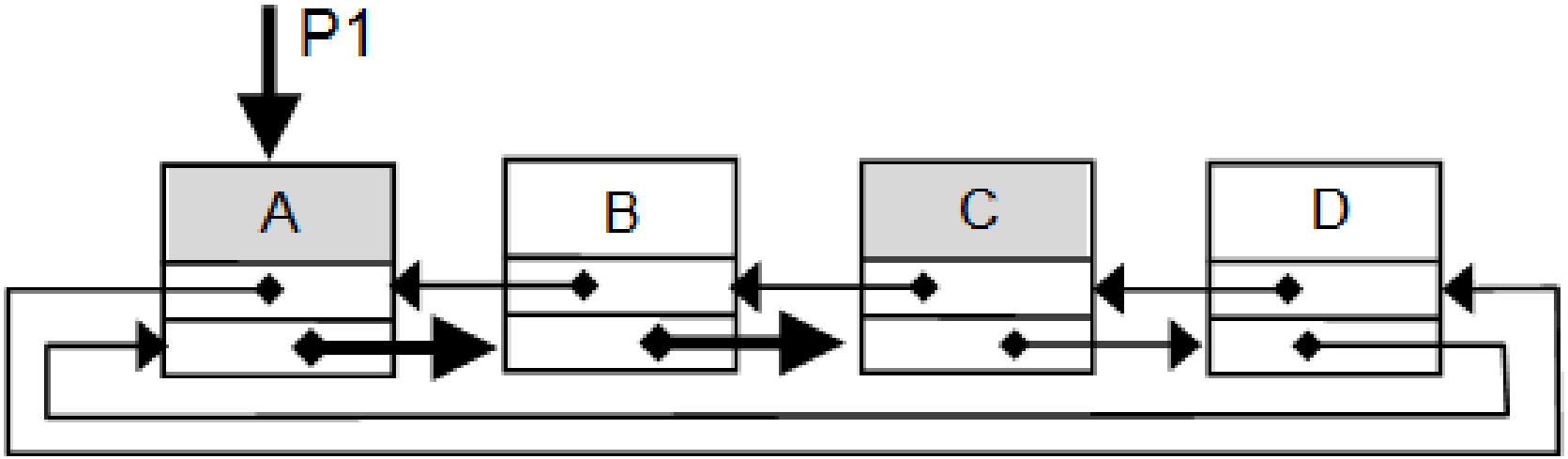


P2 = P1→Esq;

Listas Duplamente Encadeadas

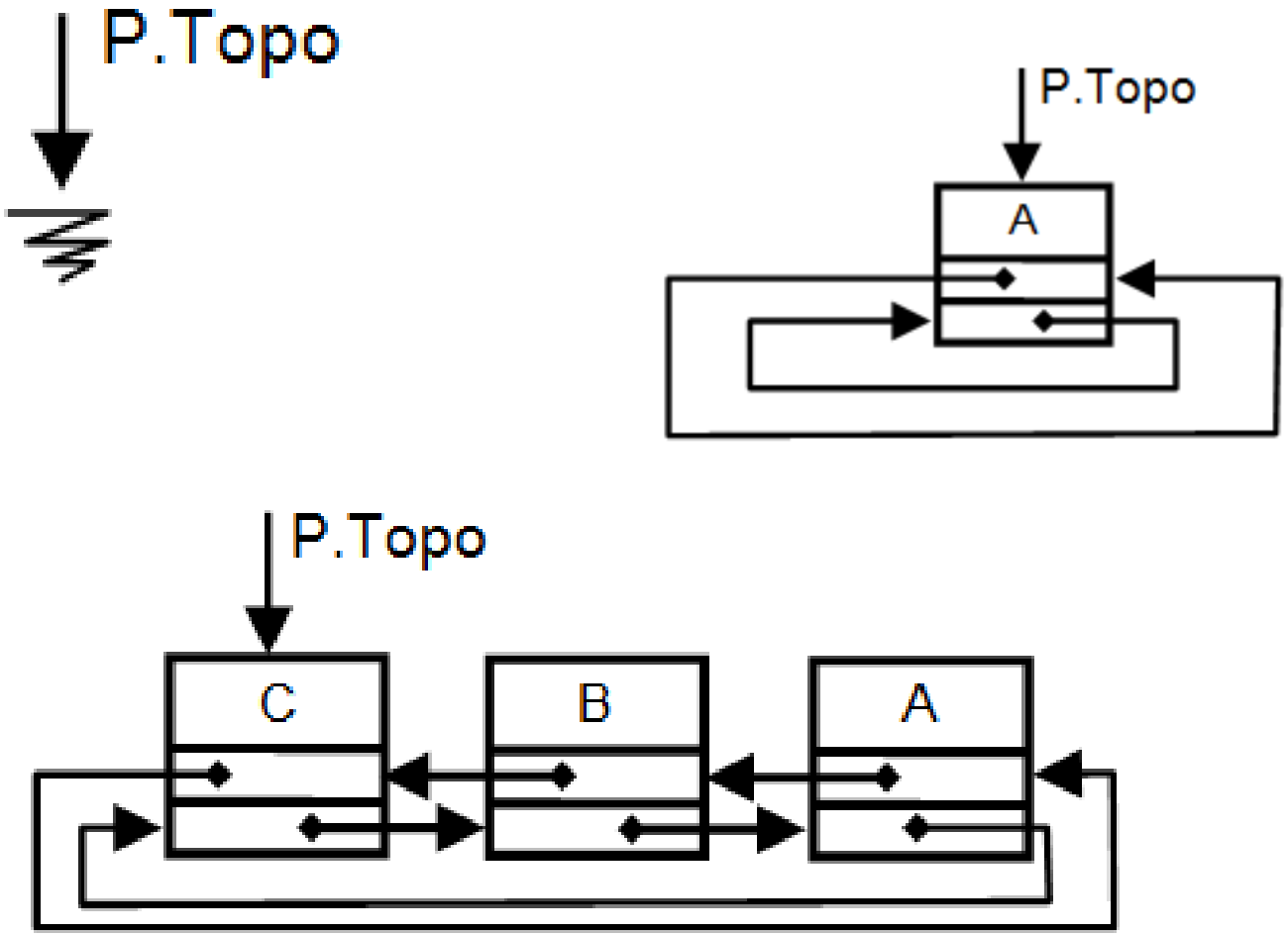


Listas Duplamente Encadeadas

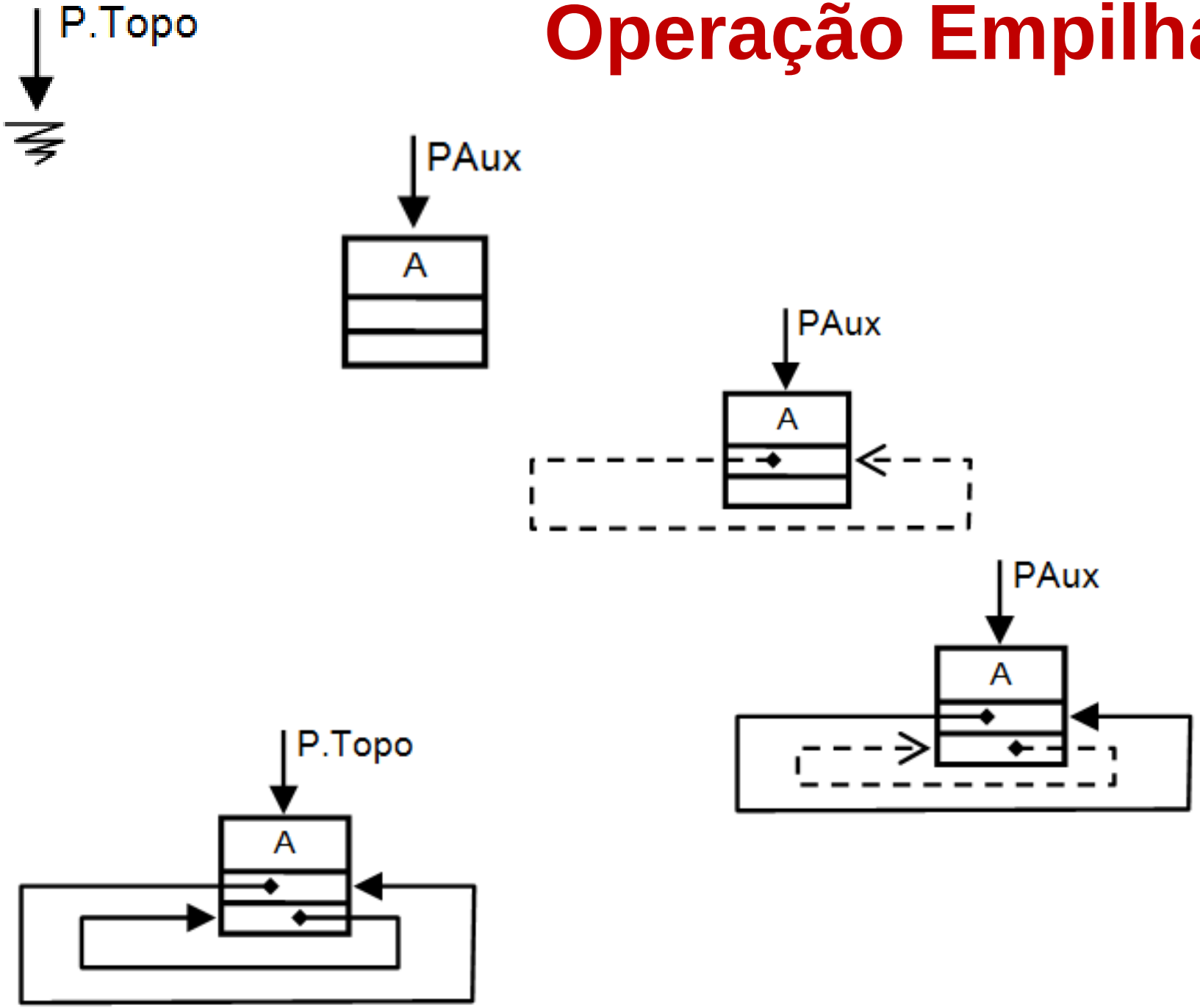


P1→Info = P1→Dir→Dir→Info;

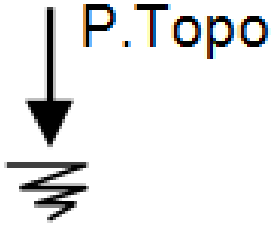
Pilha como uma Lista Circular Duplamente Encadeada



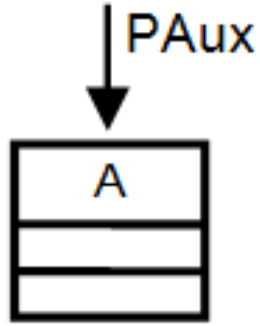
Operação Empilha

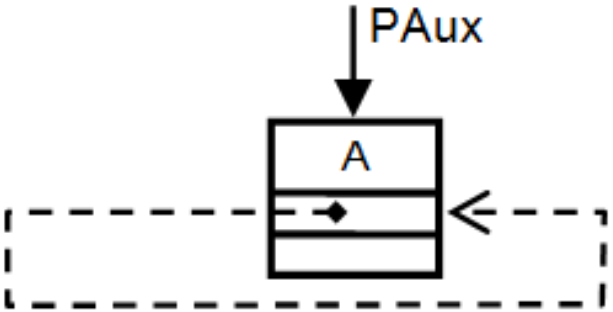


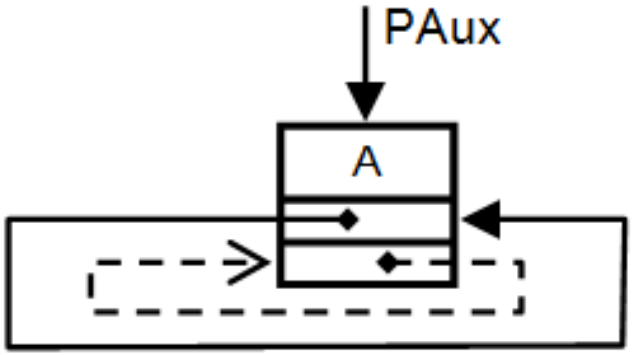
Operação Empilha

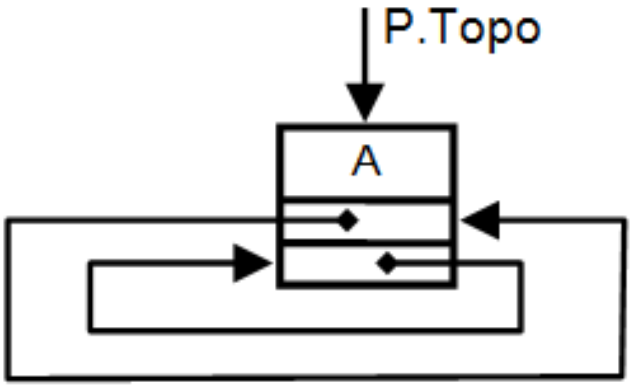


PAux = NewNode;
PAux→Info = X;

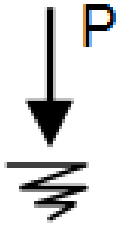




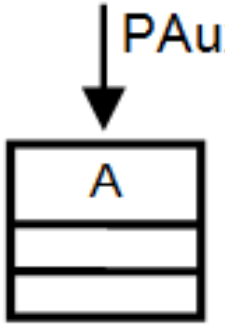




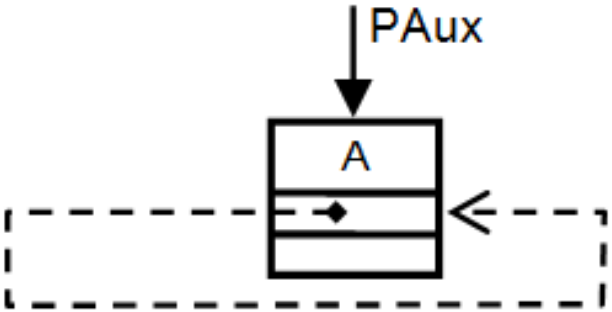
Operação Empilha

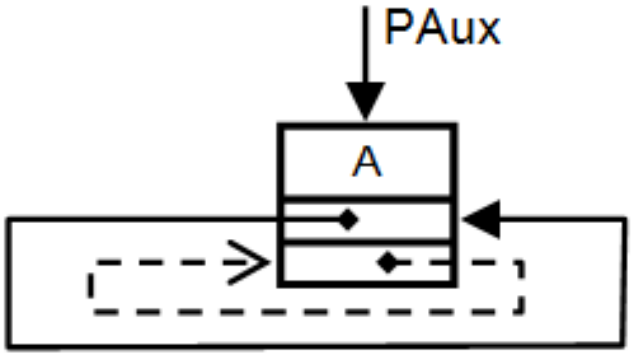


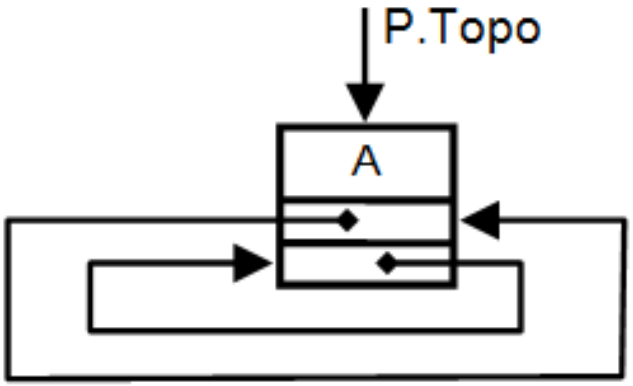
PAux = NewNode;
PAux→Info = X;



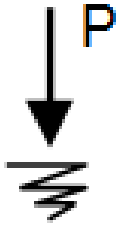
PAux→Esq = Paux;

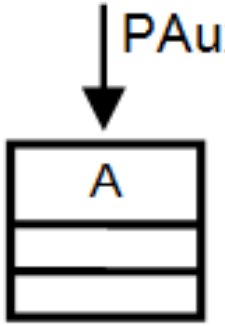




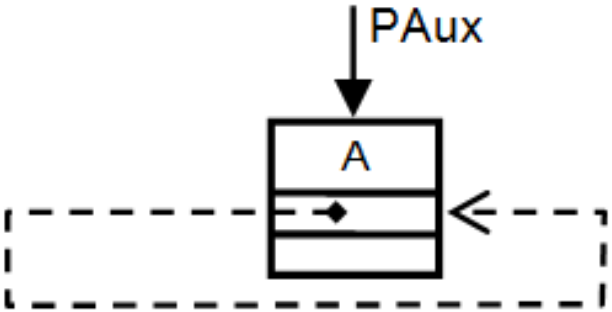


Operação Empilha

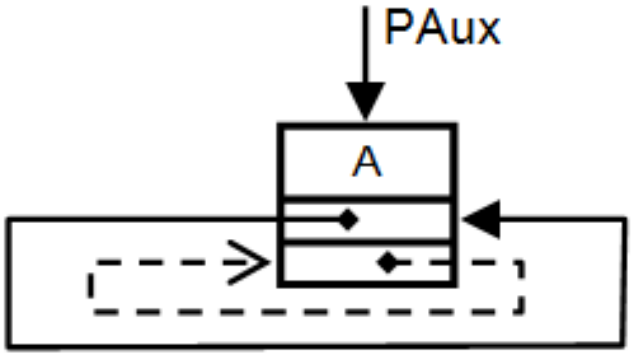




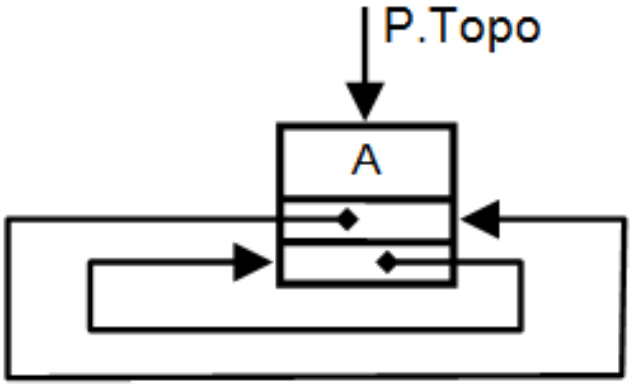
PAux = NewNode;
PAux→Info = X;



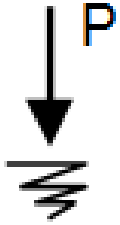
PAux→Esq = Paux;

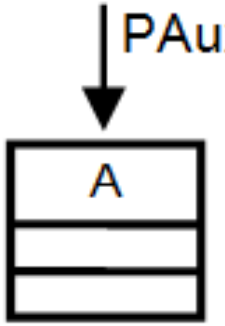


PAux→Dir= Paux;

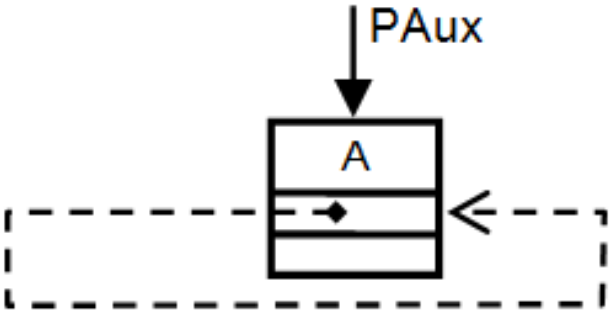


Operação Empilha

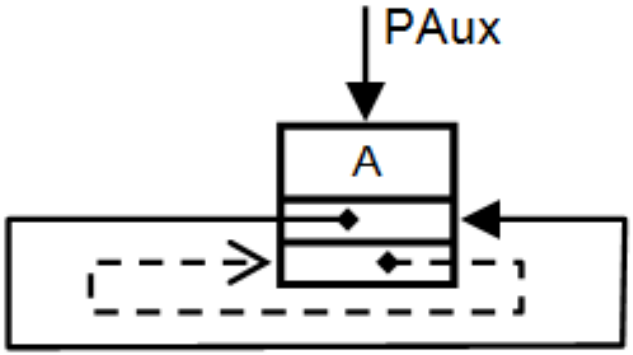




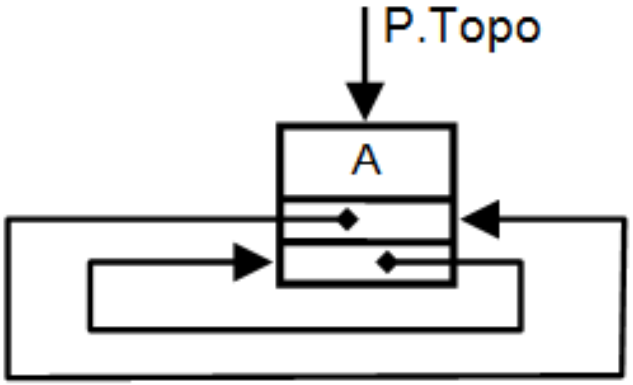
PAux = NewNode;
PAux→Info = X;



PAux→Esq = Paux;

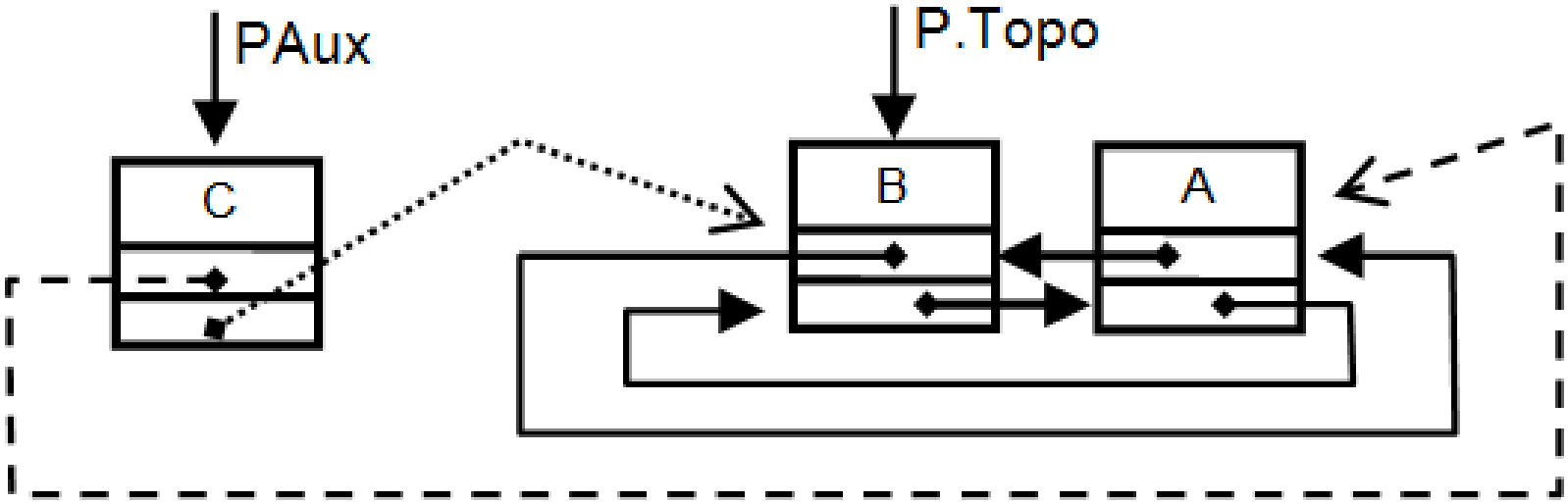


PAux→Dir= Paux;



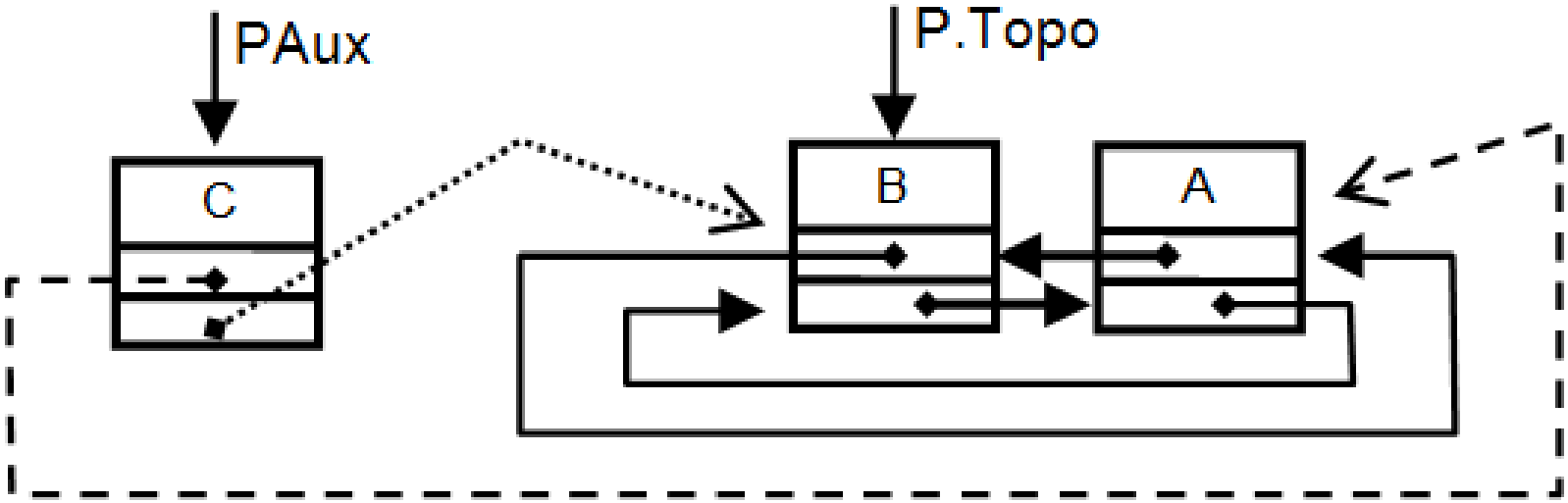
P.Topo = Paux;

Empilha



PAux = NewNode; PAux→Info = X;

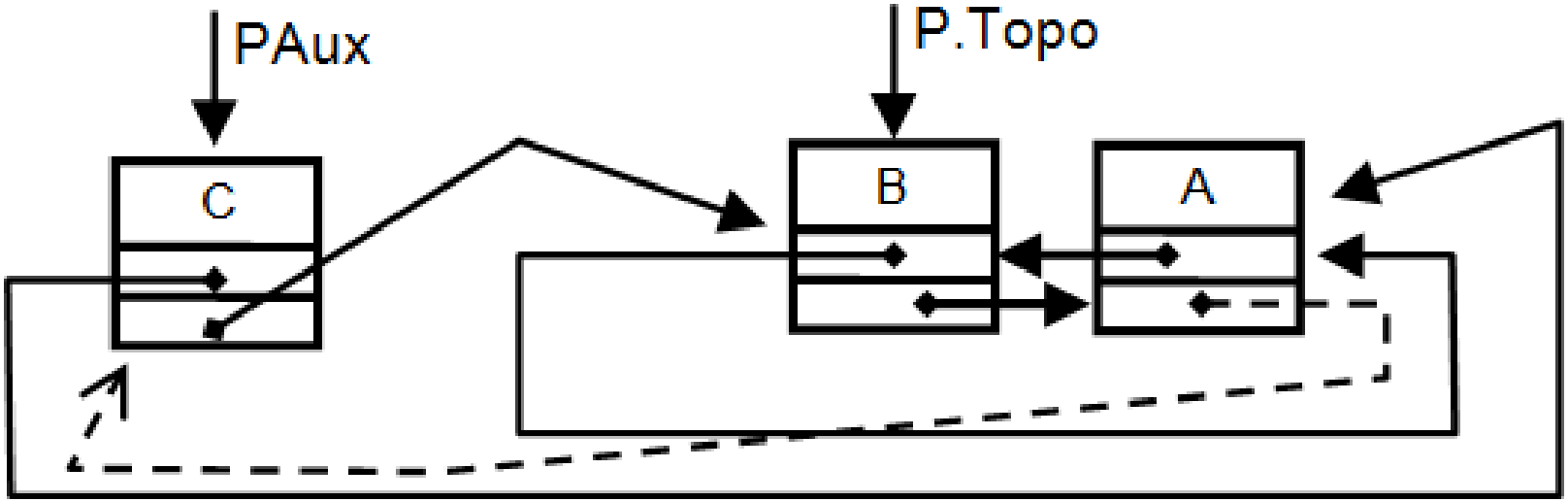
Empilha



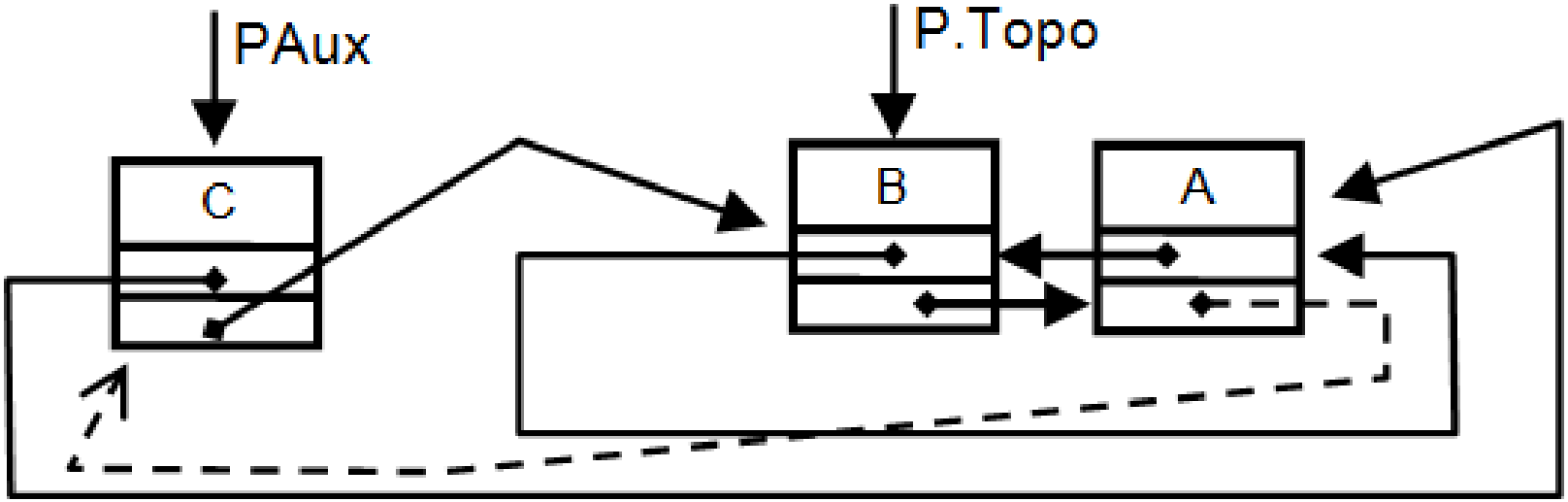
PAux = NewNode; PAux→Info = X;

PAux→Dir = P.Topo; PAux→Esq = P.Topo→Esq;

Empilha

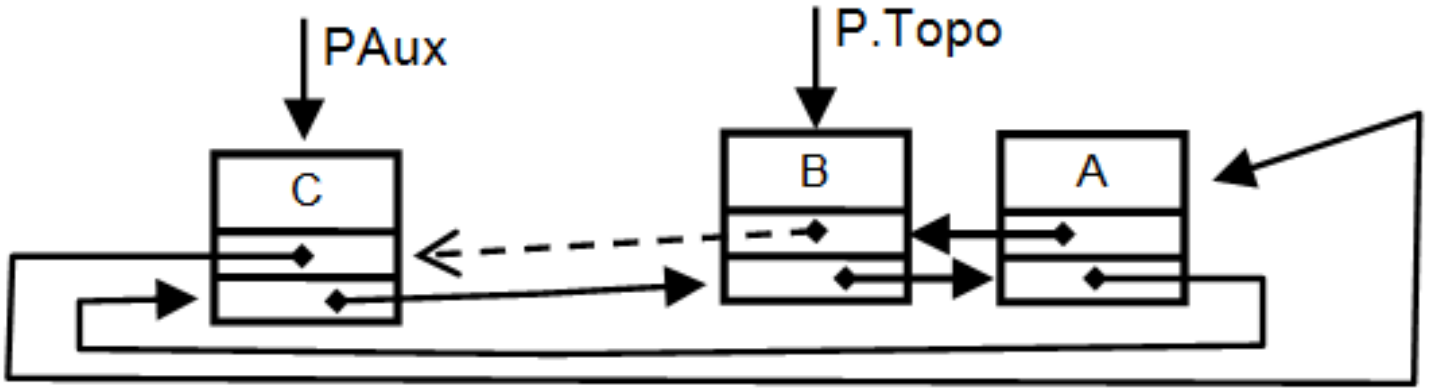


Empilha

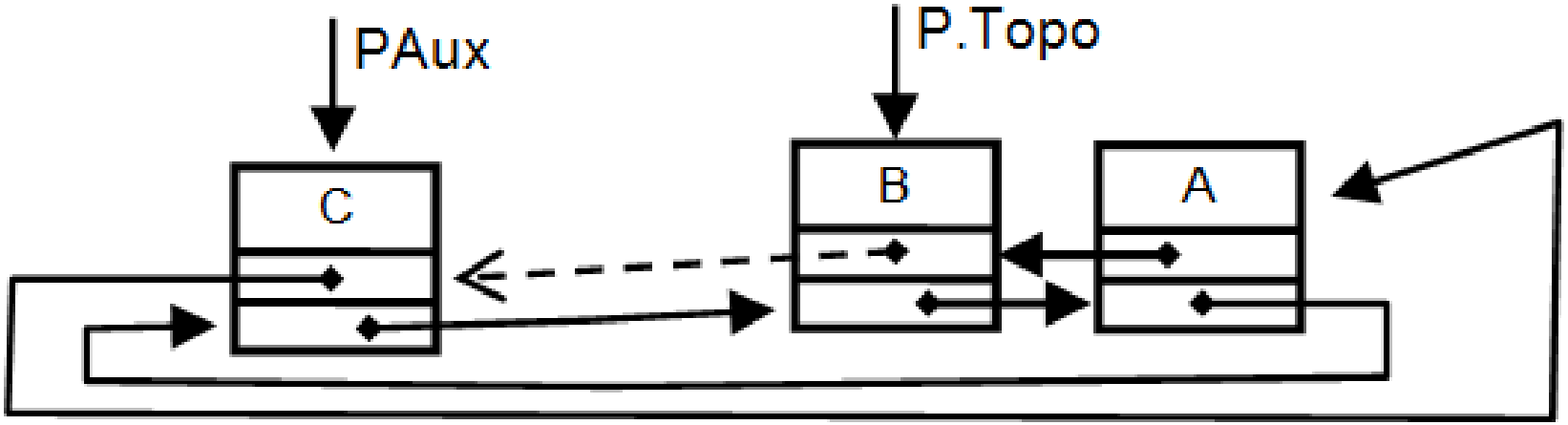


P.Topo→Esq→Dir = PAux;

Empilha

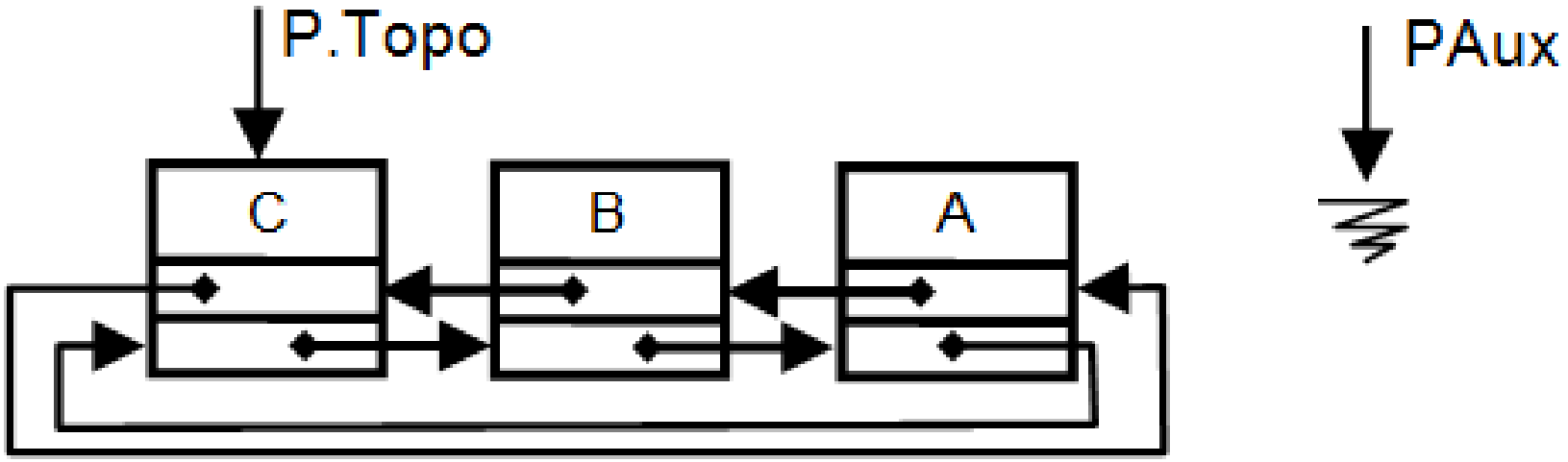


Empilha



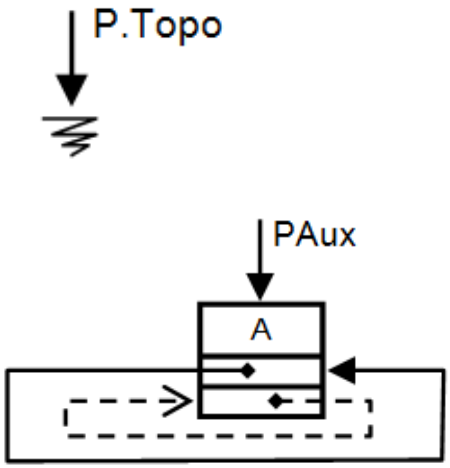
P.Topo→Esq = PAux;

Empilha



P.Topo = PAux; PAux = Null;

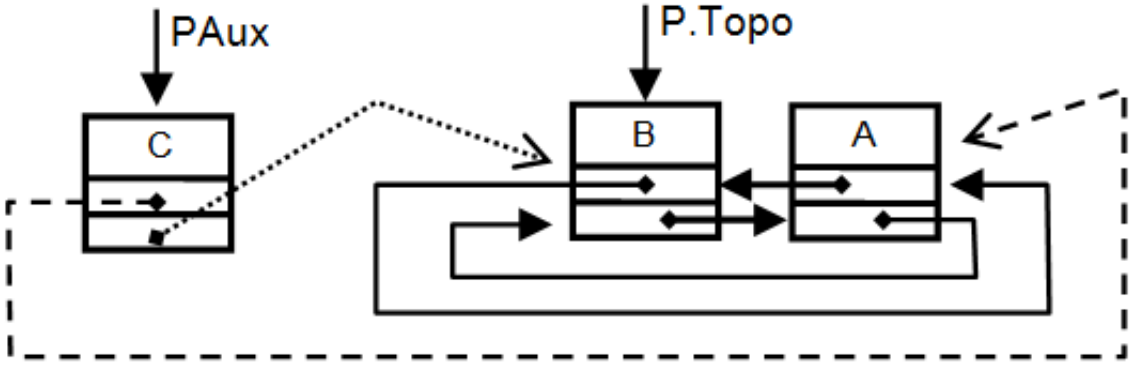
Empilha



Empilha (parâmetro por referência **P** do tipo Pilha, parâmetro **X** do tipo Char, parâmetro por referência **DeuCerto** do tipo Boolean) {

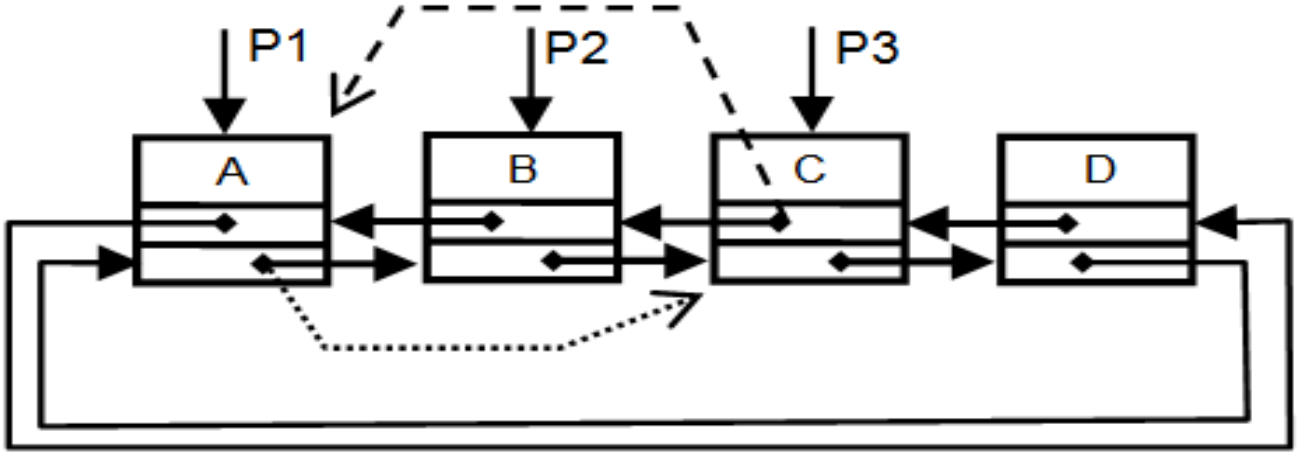
Variável PAux do tipo NodePtr;

```
Se (Cheia(P)==Verdadeiro)   // se P estiver cheia, não empilha
Então     DeuCerto = Falso;
Senão { DeuCerto = Verdadeiro;
            Se (Vazia(P)==Verdadeiro)
            Então {...}      /* Caso 1 - Pilha Vazia */
            Senão {...}      /* Caso 2 - Pilha Não Vazia */
           } // senão
} fim Empilha
```



Exercícios

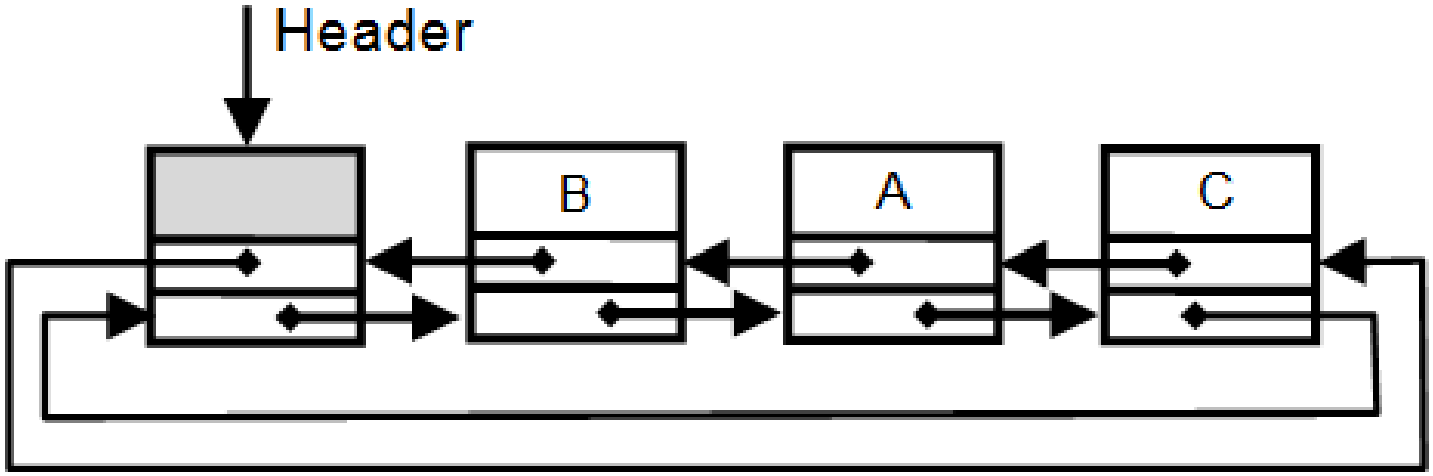
Lista Duplamente Encadeada

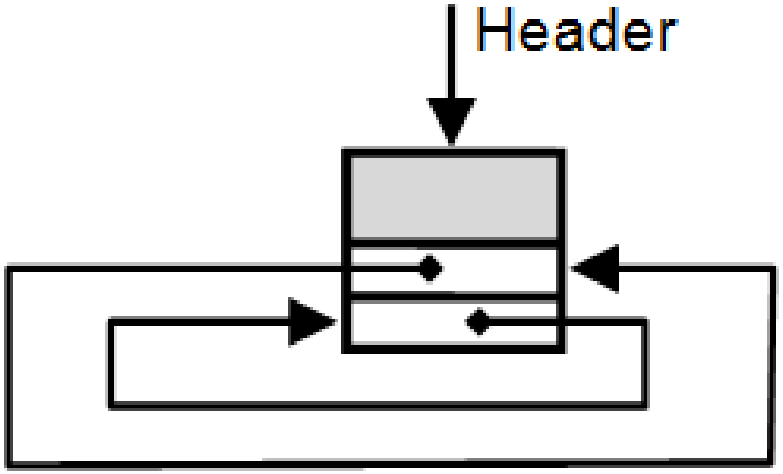


Exercícios 7.2 a 7.5 – Pilha (Desempilha, Cria, Vazia, Cheia), Fila, Lista Cadastral.

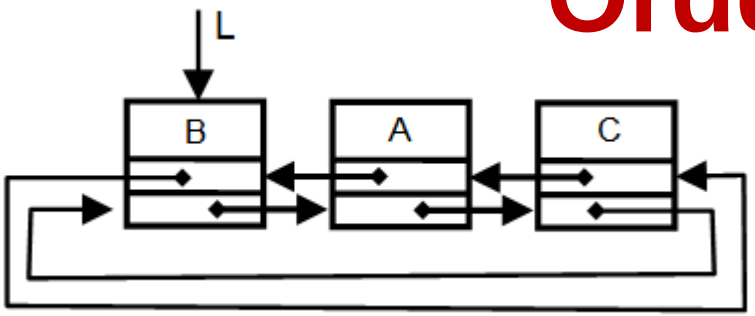
Todas as Operações. Identificar Casos. Visualizar Passo a Passo.

Listas Com Nó Header



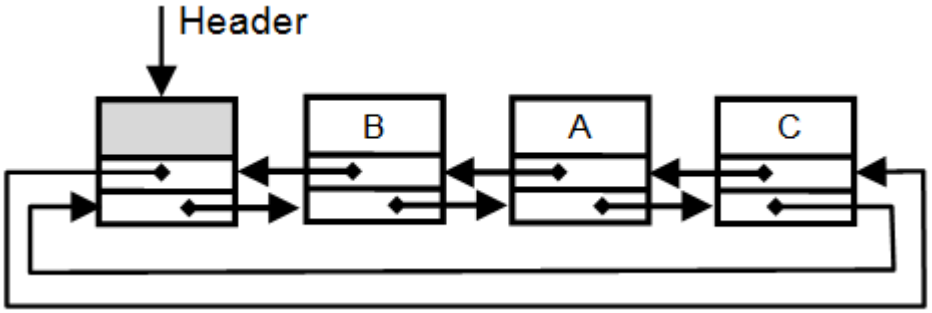


Busca em uma Lista Não Ordenada Sem Header




```
Se (L == Null)          // Lista Vazia
Então    AchouX = Falso;
Senão { P = L→Dir // P é um ponteiro auxiliar que percorre a Lista
          Enquanto (( P→Info != X )  E ( P != L ))
                    P = P→Dir;
          Se (P→Info == X)
          Então AchouX = Verdadeiro;
          Senão AchouX = Falso;
        } // senão
```

Busca em uma Lista Não Ordenada Com Header



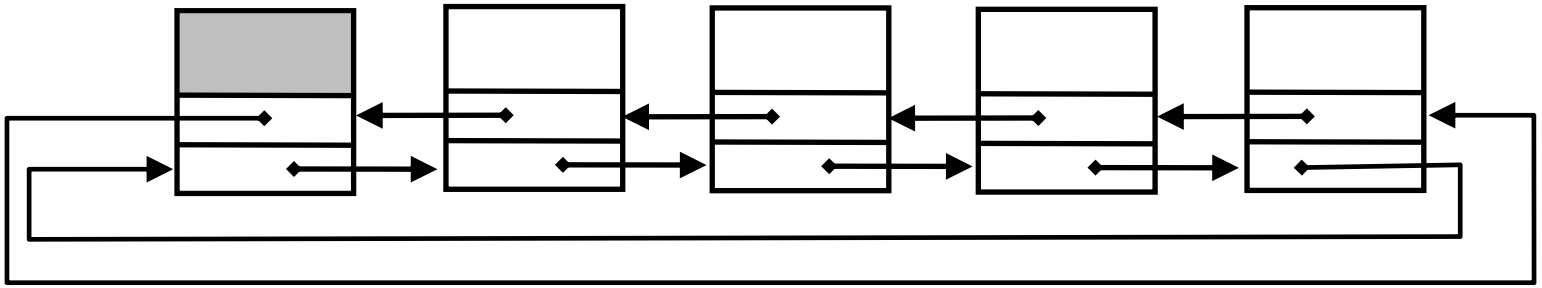

```
P = Header→Dir;                    // P é ponteiro auxiliar
Header→Info = X;                   // apenas para auxiliar na busca

Enquanto (P→Info != X)             // E ( P != L ))
          P = P→Dir;

Se  (P != Header)
Então  AchouX = Verdadeiro;        // achamos um X de verdade
Senão AchouX = Falso;              // achamos um X "de mentira"
```

Exercícios

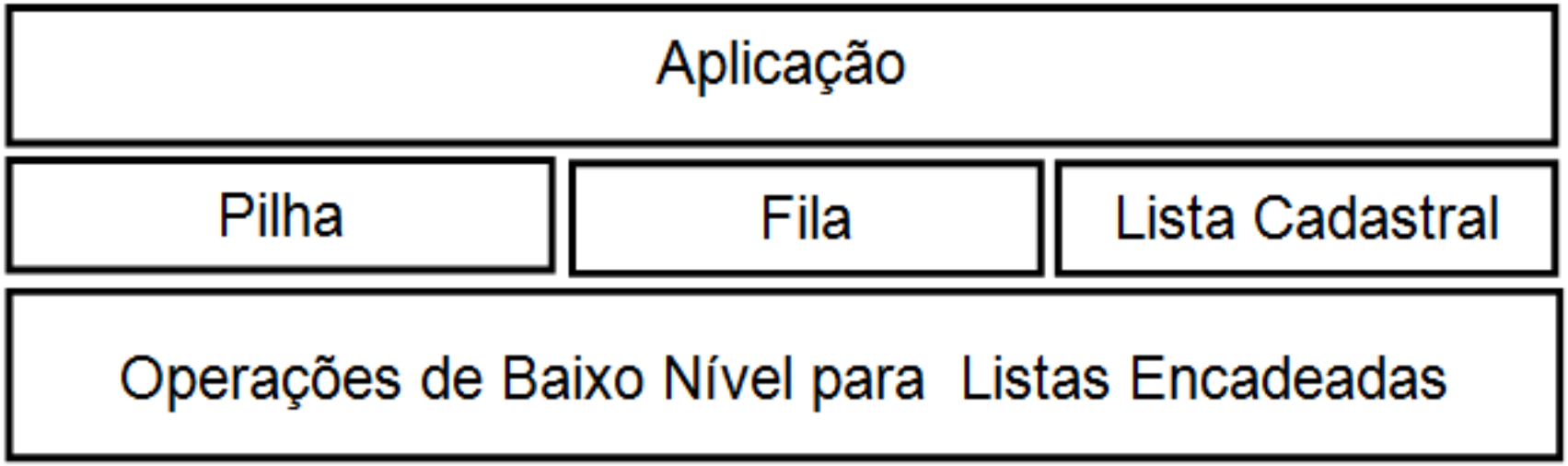
Listas Com Nó Header



Exercícios 7.6 e 7.7 – Fila e Lista Cadastral.

Todas as Operações. Identificar Casos. Visualizar Passo a Passo.

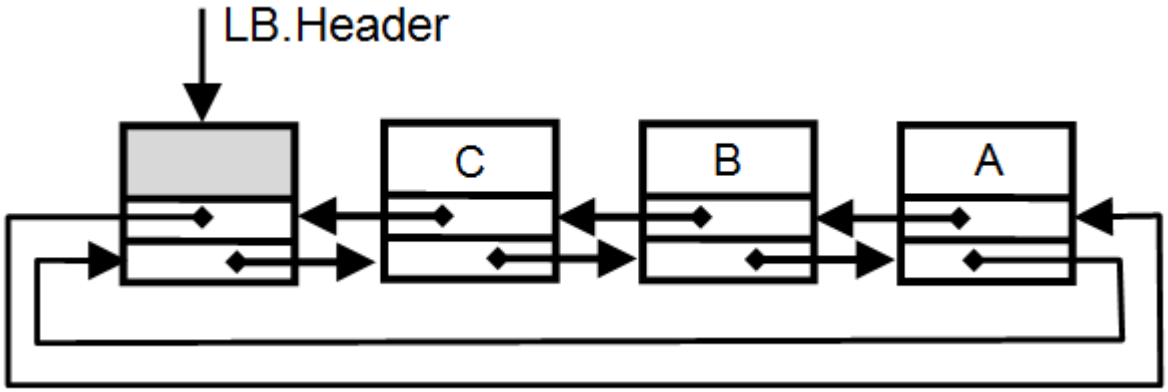
Operações de Baixo Nível para Listas Encadeadas



Operações de Baixo Nível para Listas Encadeadas

Remove_P (LB, P, Ok)
InsereADireitaDeP (LB, P, X,Ok)
EstáNaLista (LB, X, P)
Char Info_de_P (LB, P, Ok)
Vazia (LB)
Cheia(LB)
Cria (LB)
PegaOPrimeiro(LB, X, TemElemento)
PegaOPróximo(LB, X, TemElemento)

Pilha Implementada a Partir de Operações de Baixo Nível

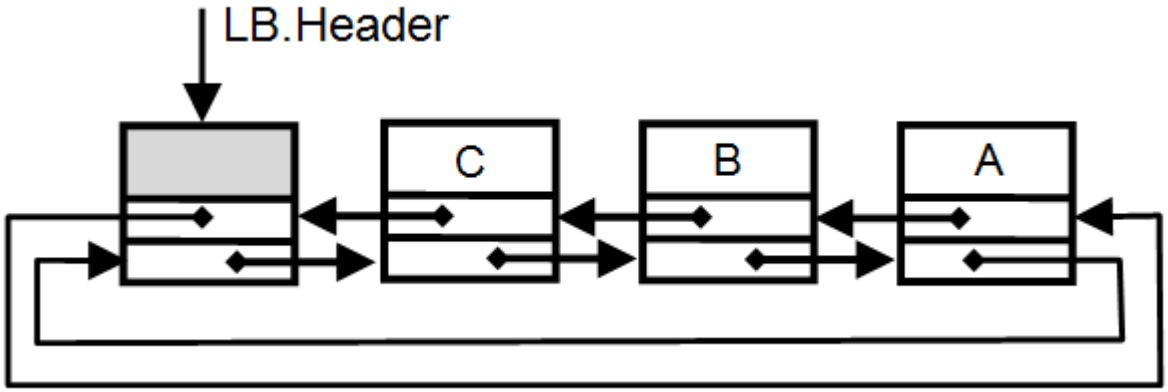


InsereADireitaDeP (LB, P, X,Ok)

Empilha (parâmetro por referência **LB** do tipo ListaBásica, parâmetro **X** do tipo Char, parâmetro por referência **DeuCerto** do tipo Boolean) {

} // fim Empilha

Pilha Implementada a Partir de Operações de Baixo Nível

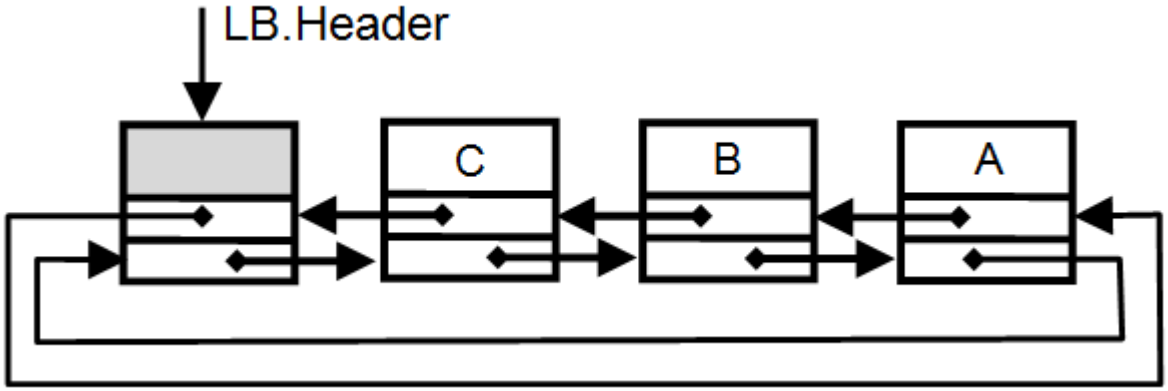


InsereADireitaDeP (LB, P, X,Ok)

Empilha (parâmetro por referência **LB** do tipo ListaBásica, parâmetro **X** do tipo Char, parâmetro por referência **DeuCerto** do tipo Boolean) {

InsereADireitaDeP (**LB**, **LB.Header**, X, DeuCerto);
/* DeuCerto é atualizado por InsereADireitaDeP */
} // fim Empilha

Pilha Implementada a Partir de Operações de Baixo Nível

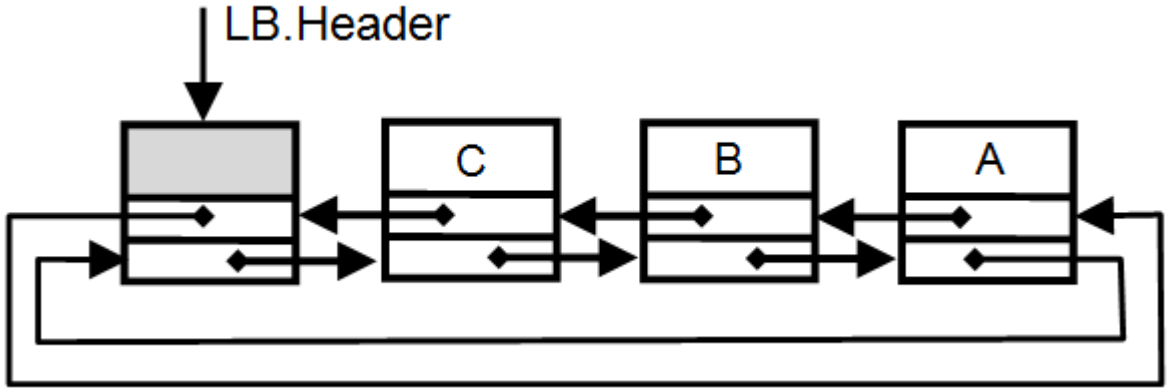


Char Info_de_P (LB, P, Ok)
Remove_P (LB, P, Ok)

Desempilha (parâmetro por referência **LB** do tipo ListaBásica, parâmetro por referência **X** do tipo Char, parâmetro por referência **DeuCerto** do tipo Boolean) {

} // fim Desempilha

Pilha Implementada a Partir de Operações de Baixo Nível



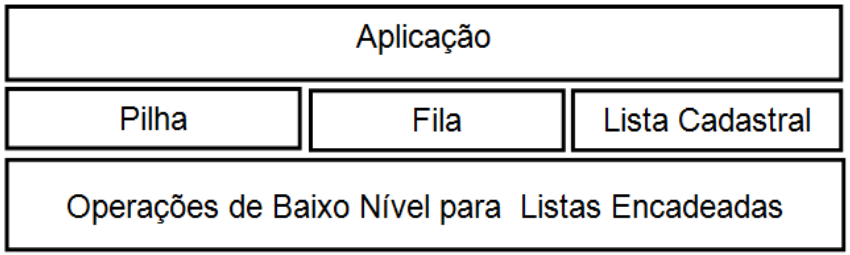
Char Info_de_P (LB, P, Ok)
Remove_P (LB, P, Ok)

Desempilha (parâmetro por referência **LB** do tipo ListaBásica, parâmetro por referência **X** do tipo Char, parâmetro por referência **DeuCerto** do tipo Boolean) {

X = Info_de_P(LB, **LB.Header→Dir**, DeuCerto);
Remove_P (LB, **LB.Header→Dir**, DeuCerto);
/* DeuCerto é atualizado por Remove_P */
} // fim Desempilha

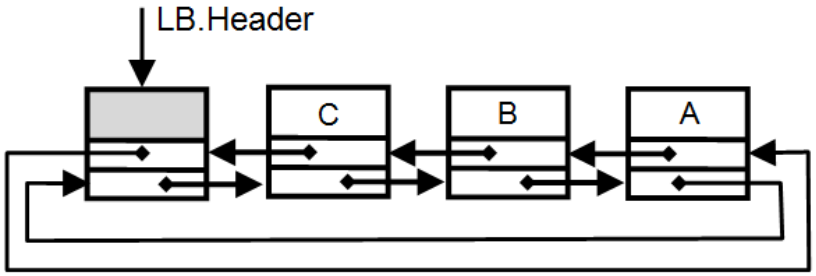
Exercícios

Operações de Baixo Nível para Listas Encadeadas



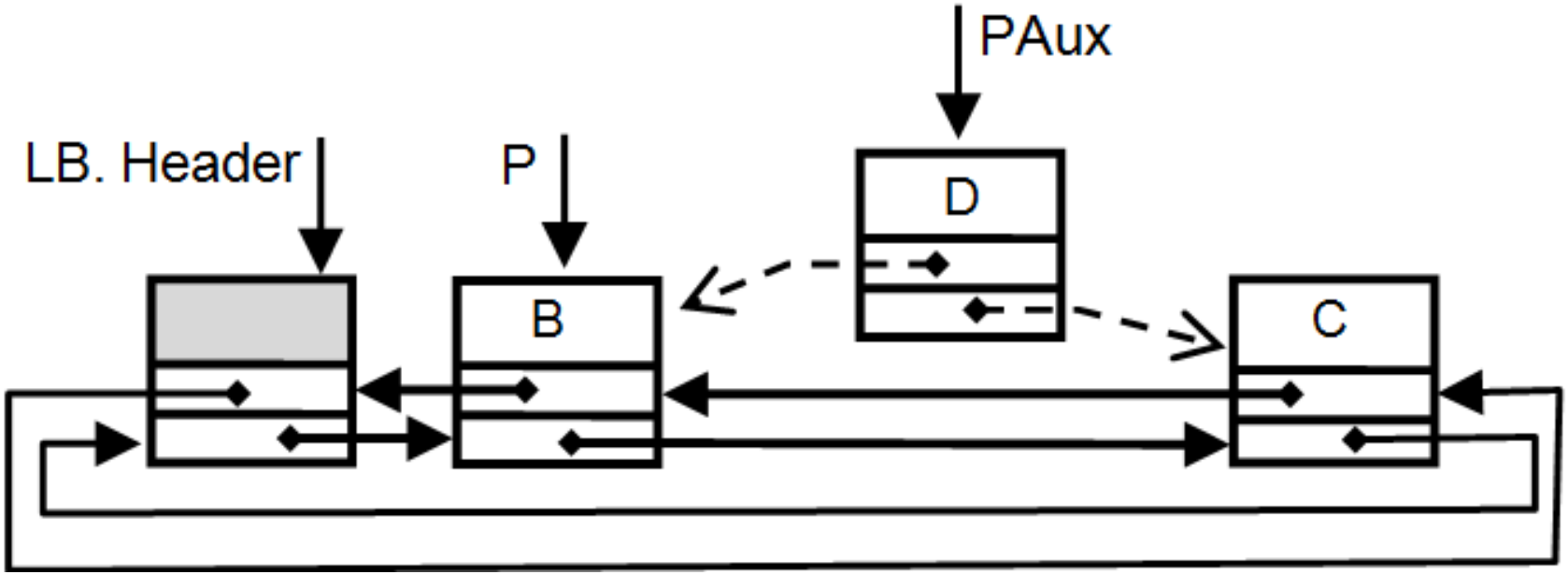
Exercícios 7.9 e 7.10 – Fila e Lista Cadastral Implementadas a partir das Operações de Baixo Nível.

Todas as Operações. Identificar Casos. Visualizar Passo a Passo. Considere Implementação por Lista Duplamente Encadeada com Header.



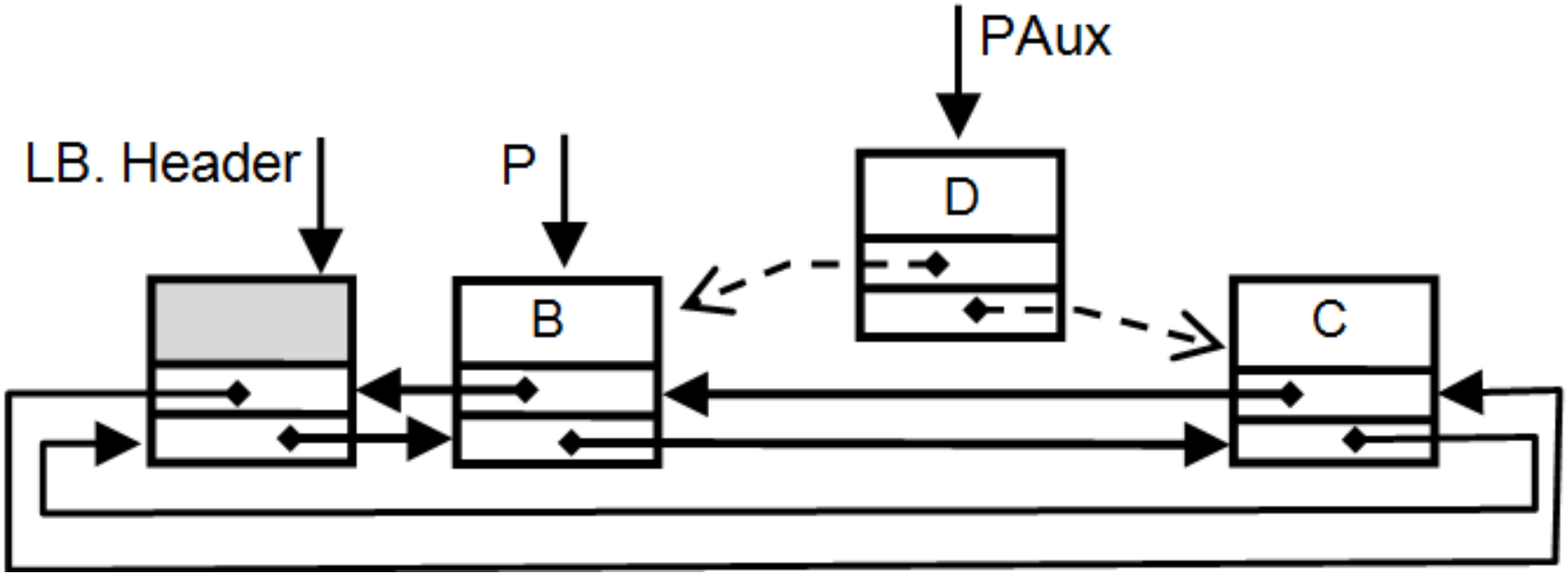
Implementando as Operações de Baixo Nível

InsereADireitaDeP (LB, P, X,Ok)



Implementando as Operações de Baixo Nível

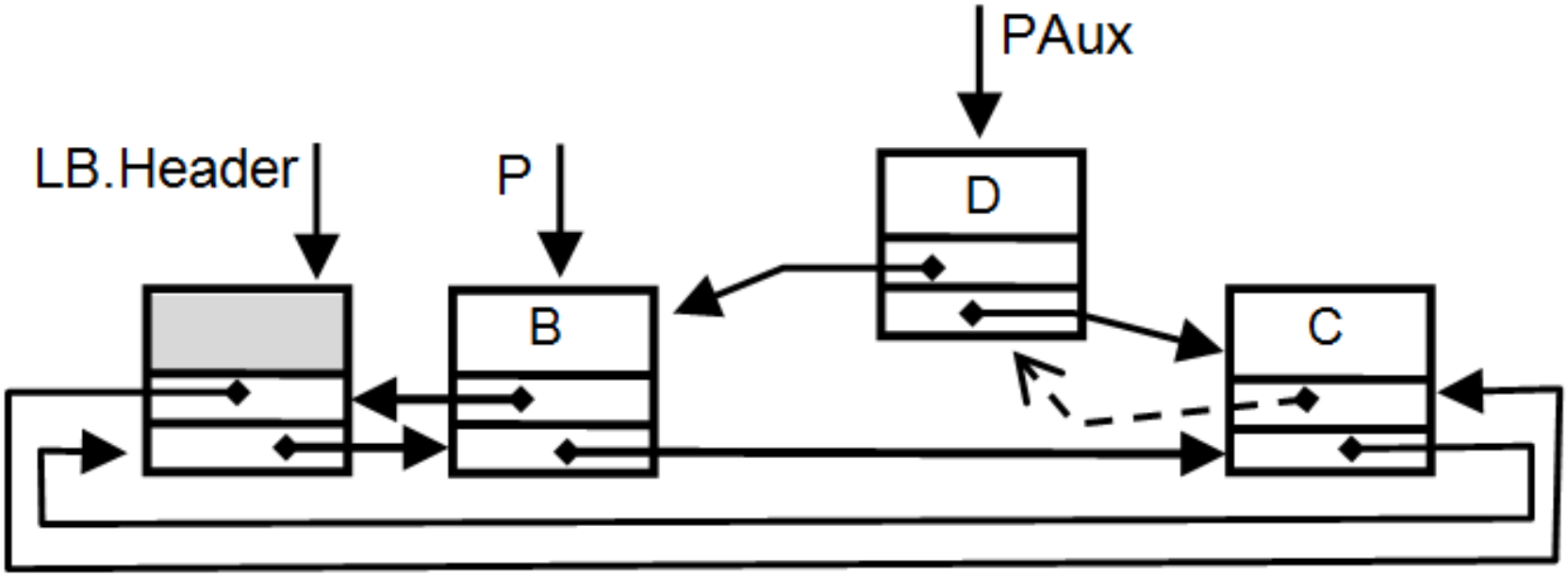
InsereADireitaDeP (LB, P, X,Ok)



PAux→Dir = P→Dir; PAux→Esq = P;

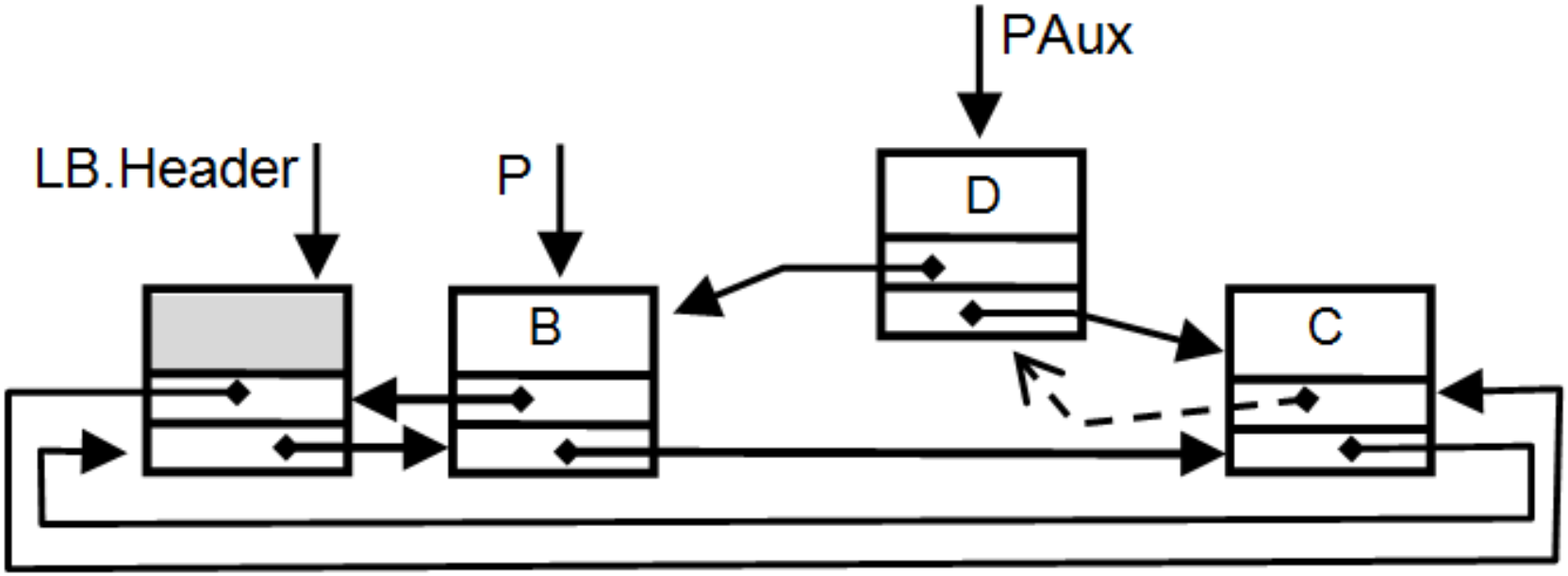
Implementando as Operações de Baixo Nível

InsereADireitaDeP (LB, P, X,Ok)



Implementando as Operações de Baixo Nível

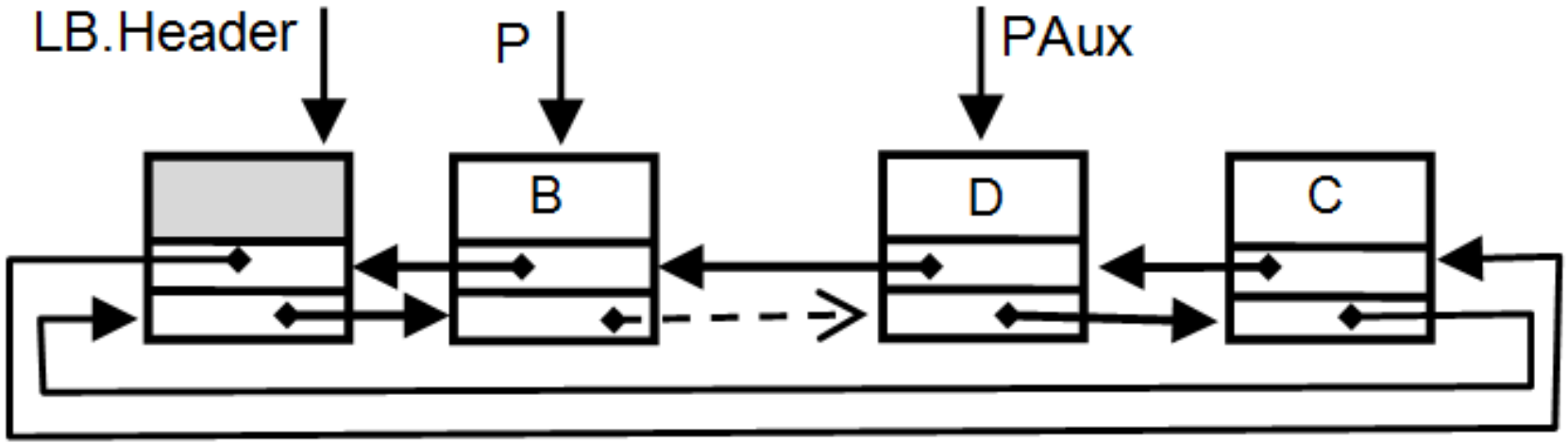
InsereADireitaDeP (LB, P, X,Ok)



P→Dir→Esq = PAux;

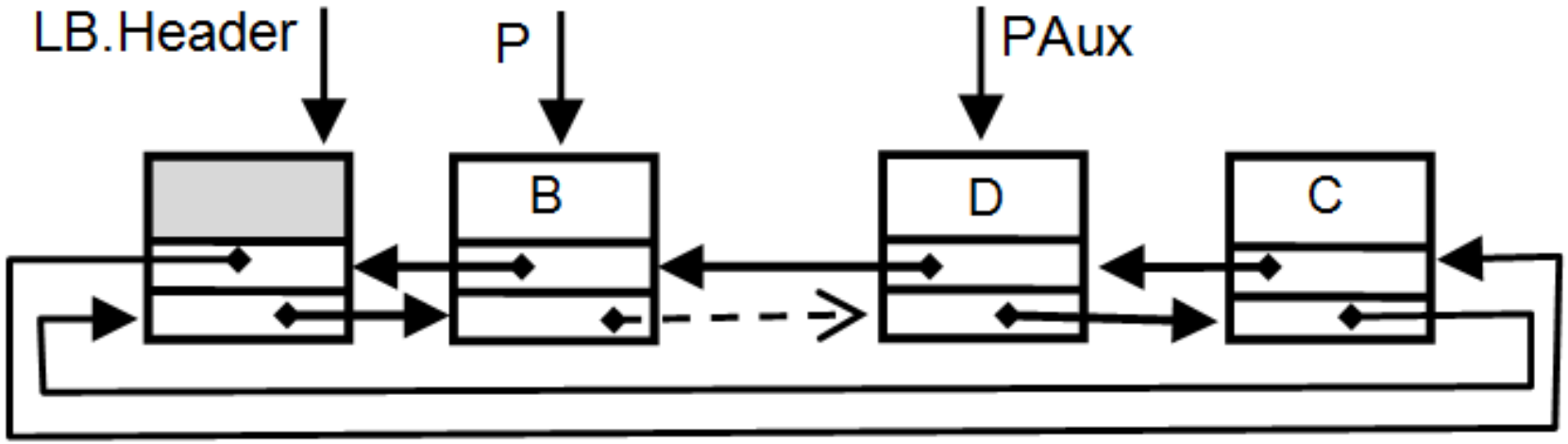
Implementando as Operações de Baixo Nível

InsereADireitaDeP (LB, P, X,Ok)



Implementando as Operações de Baixo Nível

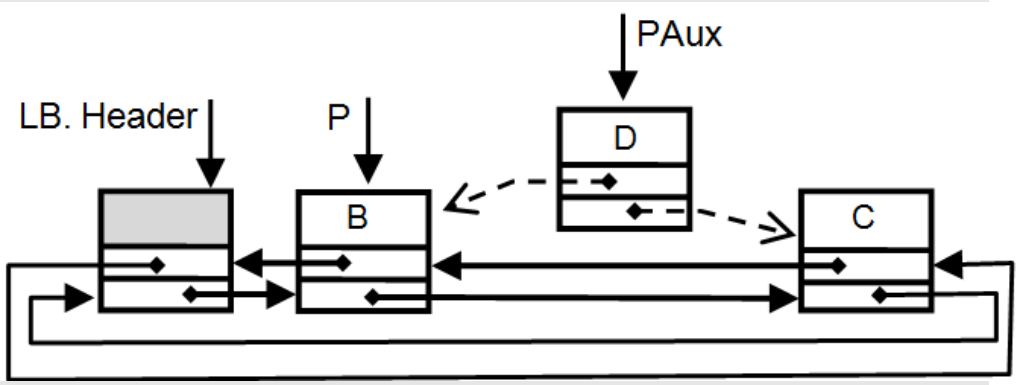
InsereADireitaDeP (LB, P, X,Ok)



P→Dir = PAux;

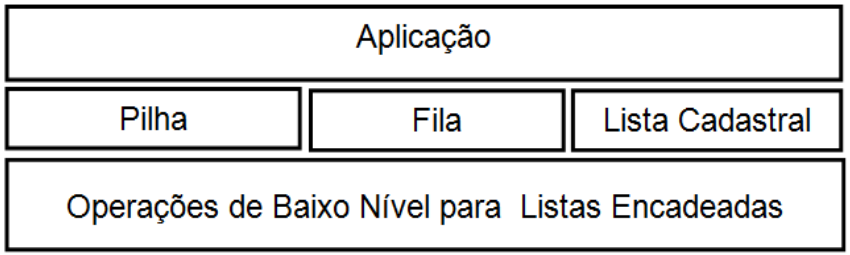
Implementando as Operações de Baixo Nível

InsereADireitaDeP (parâmetro por referência **LB** do tipo ListaBásica, parâmetro **P** do tipo NodePtr, parâmetro por referência **DeuCerto** do tipo Boolean) {
/* Insere um novo Nó a direita do Nó apontado por P, passado como parâmetro */
Variável auxiliar NovoNó do tipo NodePtr; // NodePtr = ponteiro para Nó
Se (Cheia(LB)==Verdadeiro)
Então DeuCerto = Falso;
Senão { DeuCerto = Verdadeiro;
PAux = NewNode;
PAux→Info = X;
PAux→Dir = P→Dir;
PAux→Esq = P;
P→Dir→Esq = PAux;
P→Dir = PAux;
Aux = Null;
} // fim Senão
} // fim Insere_a_Direita_de_P



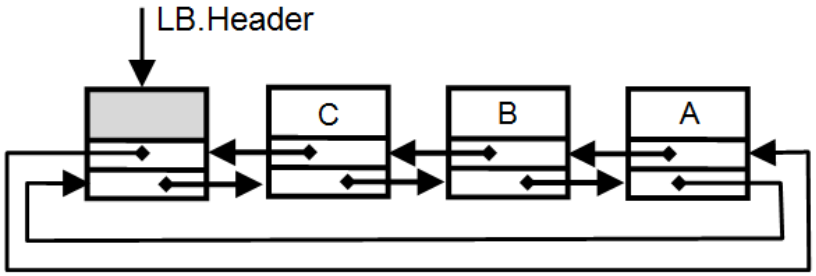
Exercícios

Implementar Primitivas de Baixo Nível

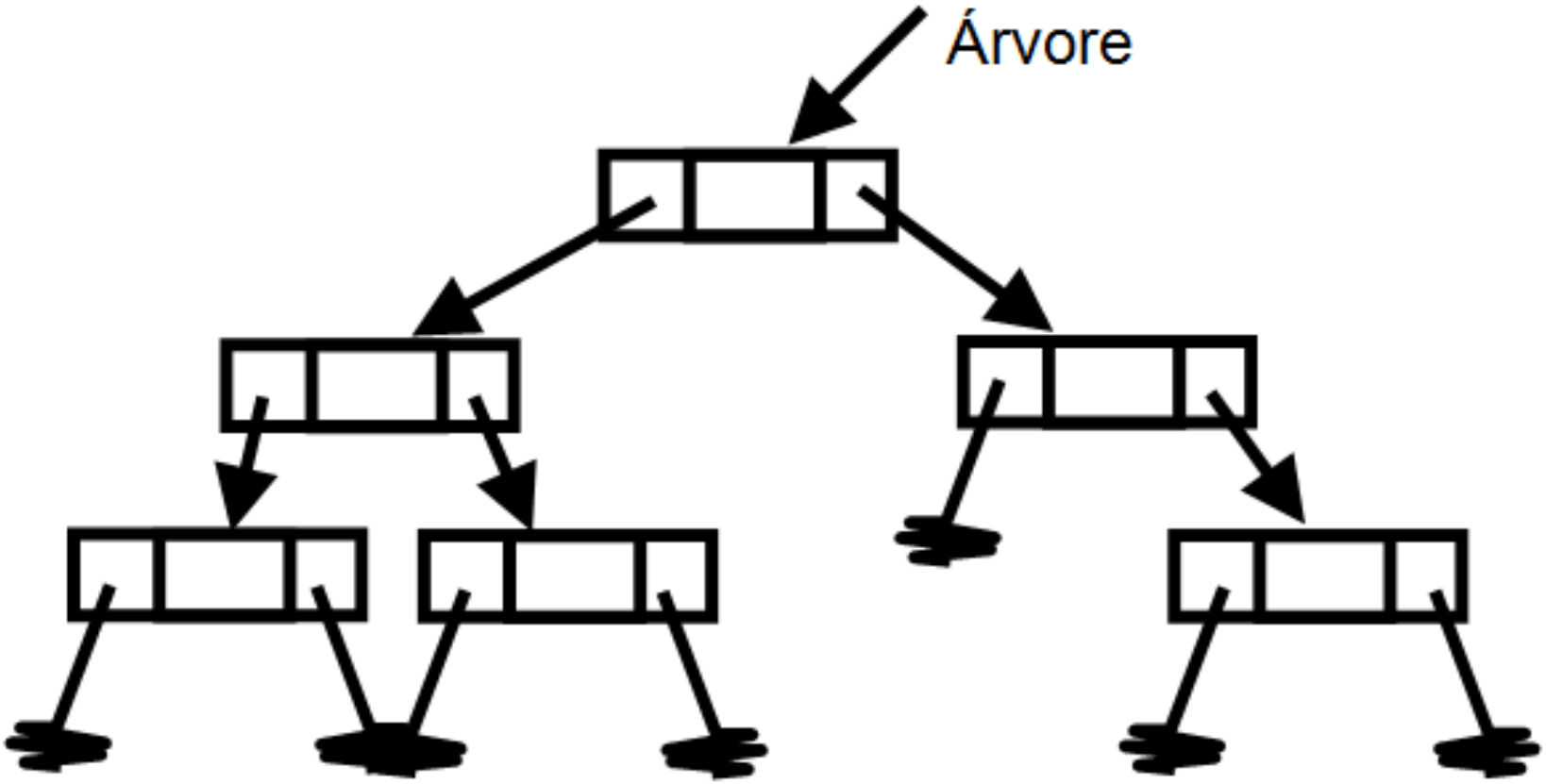


Exercício 7.12 Remove_P, EstáNaLista e Info_de_P

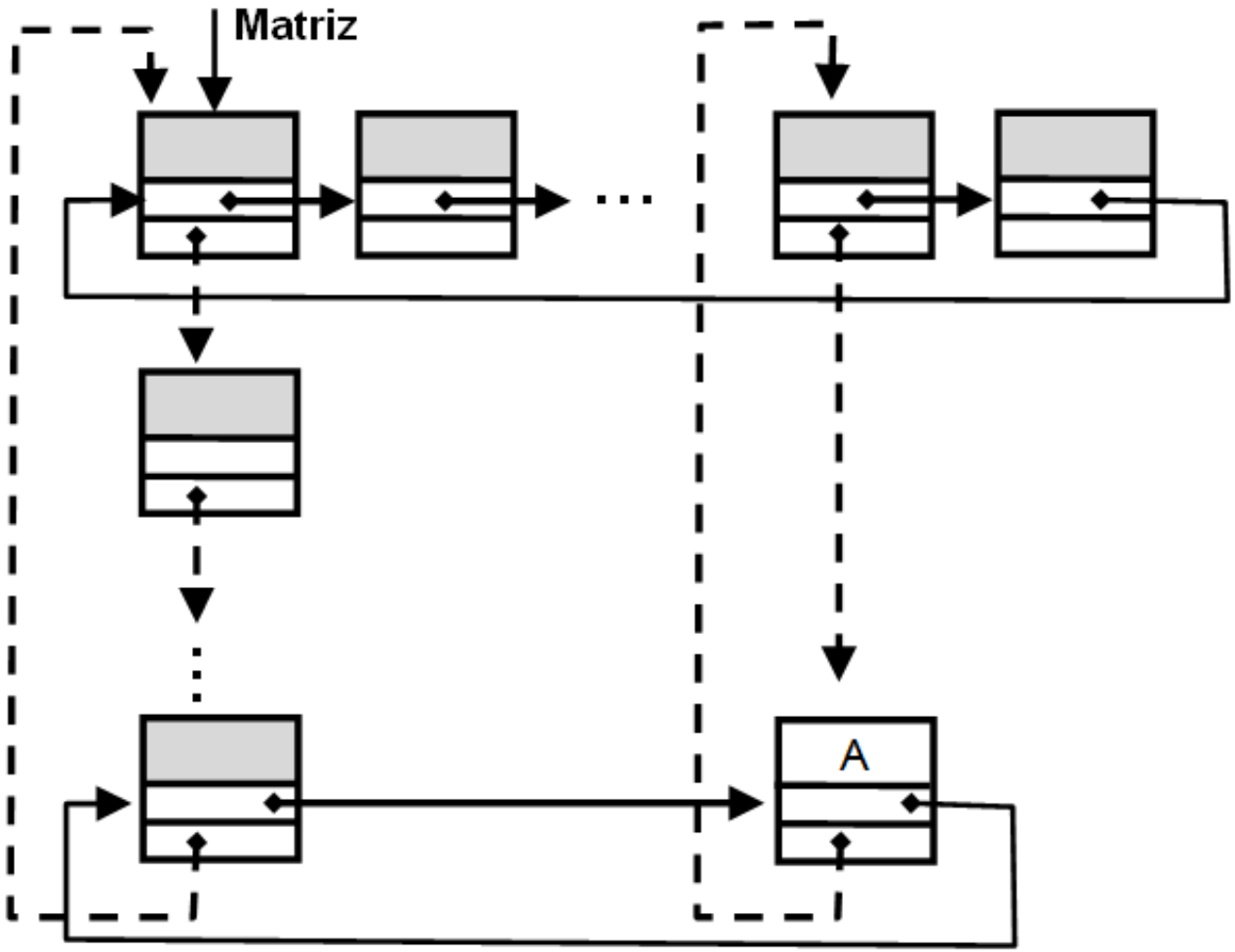
Identificar Casos. Visualizar Passo a Passo. Considere Implementação por Lista Duplamente Encadeada com Header.



Generalização de Listas Encadeadas

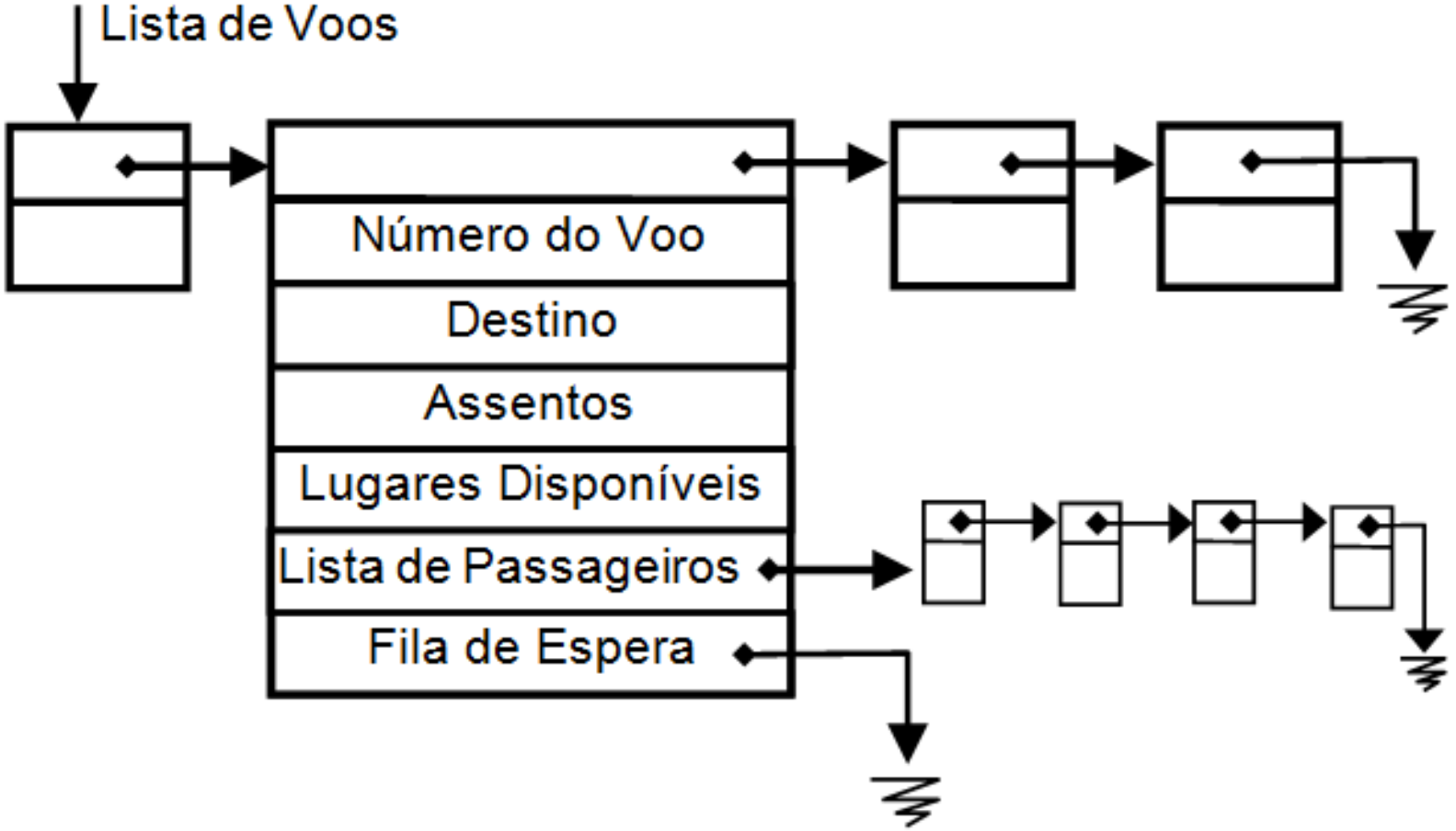


Generalização de Listas Encadeadas



Matrizes Esparsas

Generalização de Listas Encadeadas



Estruturas Aninhadas

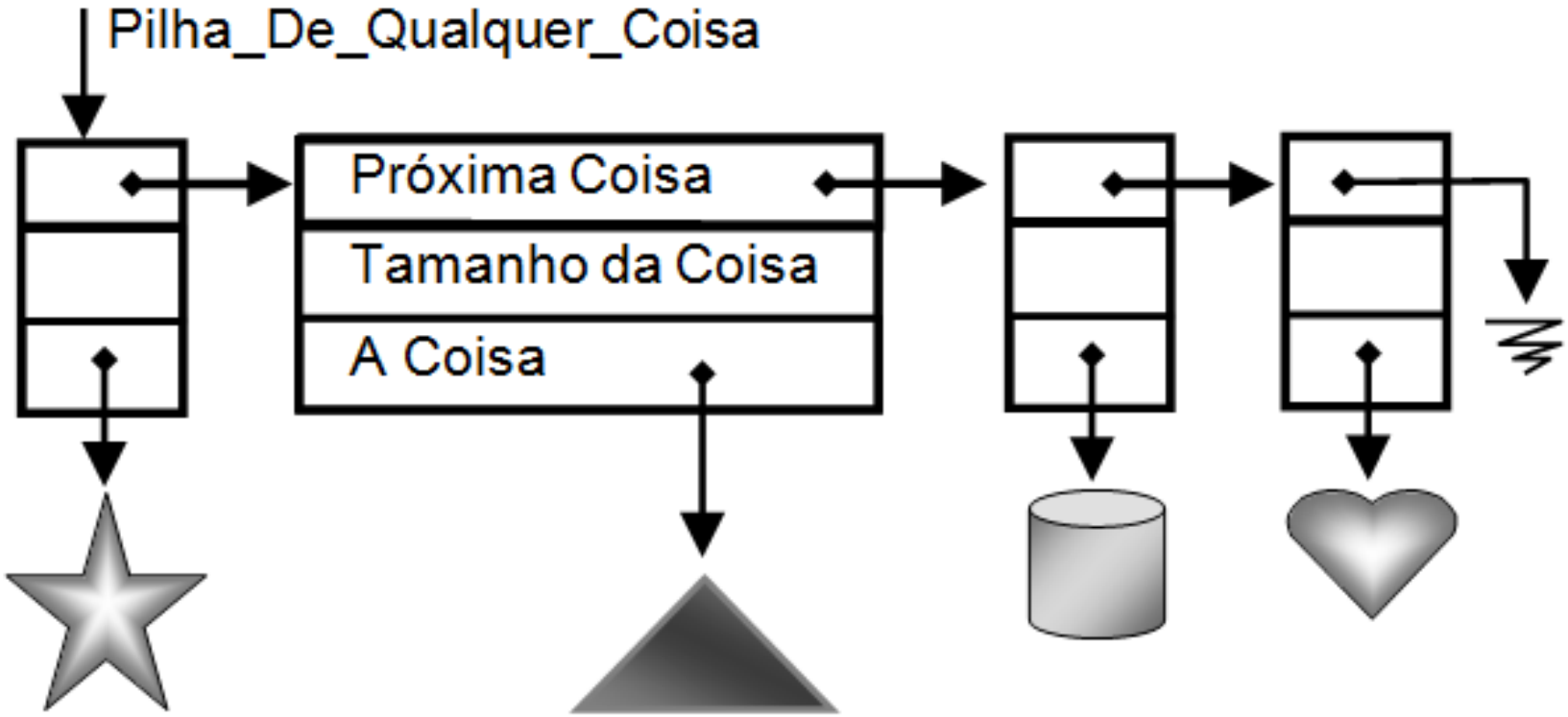
Generalização de Listas Encadeadas

```
/* definição da Classe Node */
template<class TipoDoElemento> class Node {
// class Node usa  template TipoDoElemento

private:
TipoDoElemento Info;  // o tipo do elemento será definido posteriormente
Node<TipoDoElemento>* Next;
...
/* definição da Classe Pilha */
template<class TipoDoElemento> class Pilha {
private:
Node<TipoDoElemento> * Topo;
...
/* no momento de criar as Pilhas, indicamos o tipo dos elementos */
Pilha<int> * p1 = new Pilha<int>();                 // cria uma Pilha de Inteiros - P1
Pilha<Carta> * p2 = new Pilha<Carta>();             // cria uma Pilha de Cartas - P2
```

Estruturas Genéricas Quanto ao Tipo do Elemento: Templates

Generalização de Listas Encadeadas



Elementos de Tipos Diferentes em uma Mesma Estrutura

Exercícios

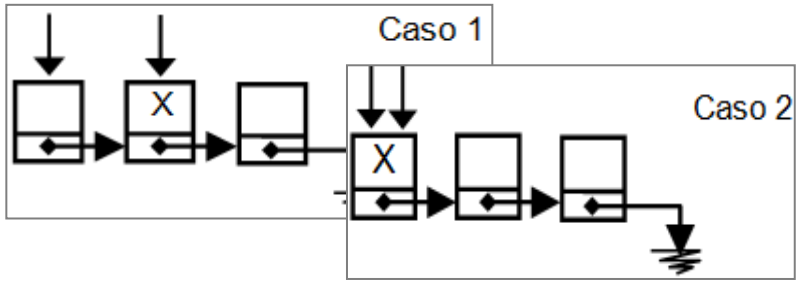
Exercício 7.17 "Pilhas, Filas e Listas Cadastrais possuem uma funcionalidade bem definida, e podem ser implementadas através de Listas Encadeadas ou através de outras técnicas". Neste contexto, qual a diferença entre uma Lista Cadastral e uma Lista Encadeada?

Exercício 7.13 Fila de Prioridades: Fila com Atendimento Prioritário a Idosos
Fila de espera comdois critérios: (1) a idade do indivíduo que está na Fila, em anos; (2) a ordem de chegada. O primeiro a ser atendido será sempre o individuo com idade mais alta. Se houver mais que um individuo com a mesma idade, será respeitado o segundo critério, que é a ordem de chegada. Projete uma estrutura para implementar esta Fila com Atendimento Prioritário a Idosos. Implemente a funcionalidade definida com a técnica de implementação que considerar mais adequada.

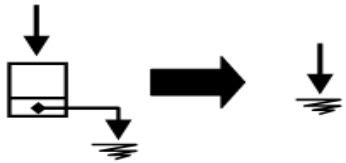
Exercício 7.14 Lista Cadastral em C++

Passos Para Construir uma Boa Solução

Passo 1: Identificar Casos e Desenhar a Situação Inicial.



Passo 2: Desenho da Situação Final.



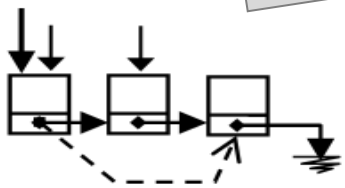
Passo 3: Algoritmo para Tratar Cada Caso, Separadamente.

Para Tratar Caso 2:
- DeleteNode(P);
- L=Null;

Passo 4: Faça um Algoritmo Geral do Modo Mais Simples.

```
Se Caso 1
Então [tratar Caso 1]
Senão Se Caso 2
        Então [tratar Caso 2]
        Senão Se Caso 3...
```

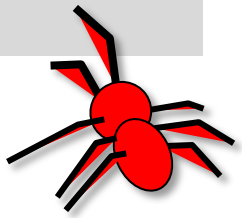
Passo 5: Testar Cada Caso, Alterando o Desenho Passo a Passo.



Avanço de Projeto

Projete Sua Própria Estrutura!

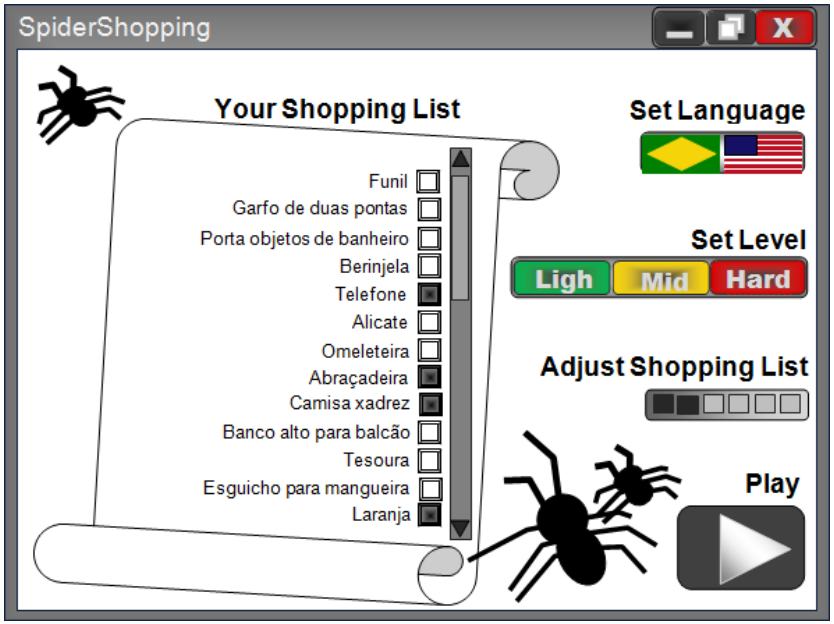
As Listas Encadeadas, e suas variações, constituem uma técnica poderosa e flexível para armazenamento temporário de conjuntos de elementos. É possível adaptar ou combinar diversas técnicas para projetar estrutura encadeada que melhor atenda as peculiaridades de sua aplicação.



(Exercício 6.30) Técnica de Implementação

Qual combinação de técnicas parece ser mais adequada às características do jogo referente ao Desafio 3 que você desenvolverá?

Comece a Desenvolver Seu Jogo Agora!



Faça um jogo com a sua cara! Distribua para os seus amigos!

Estruturas de Dados com Jogos

Aprender a programar pode ser divertido!