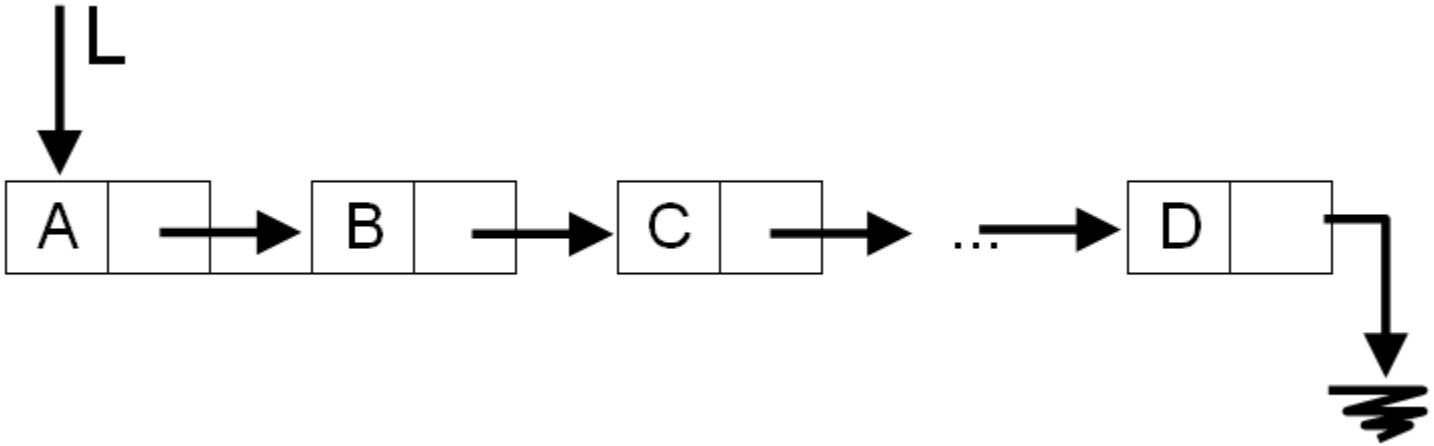
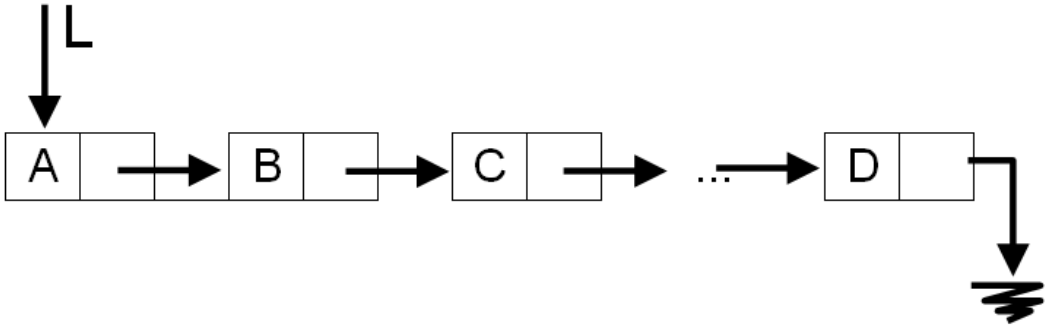
Estruturas de Dados com Jogos



Capítulo 4

Listas Encadeadas

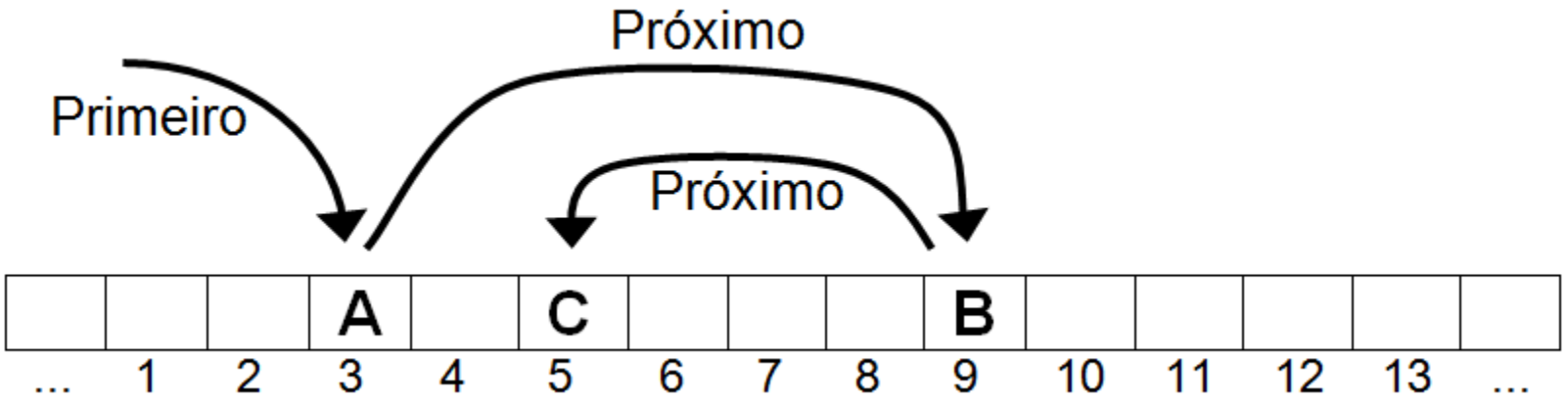
Seus Objetivos neste Capítulo



- Entender o que é Alocação Encadeada de Memória, no contexto do armazenamento temporário de conjuntos de elementos; conhecer o conceito de Listas Encadeadas, e sua representação usual;

- Desenvolver habilidade para implementar Pilhas e Filas através do conceito de Listas Encadeadas;

Alocação Sequencial Versus Alocação Encadeada

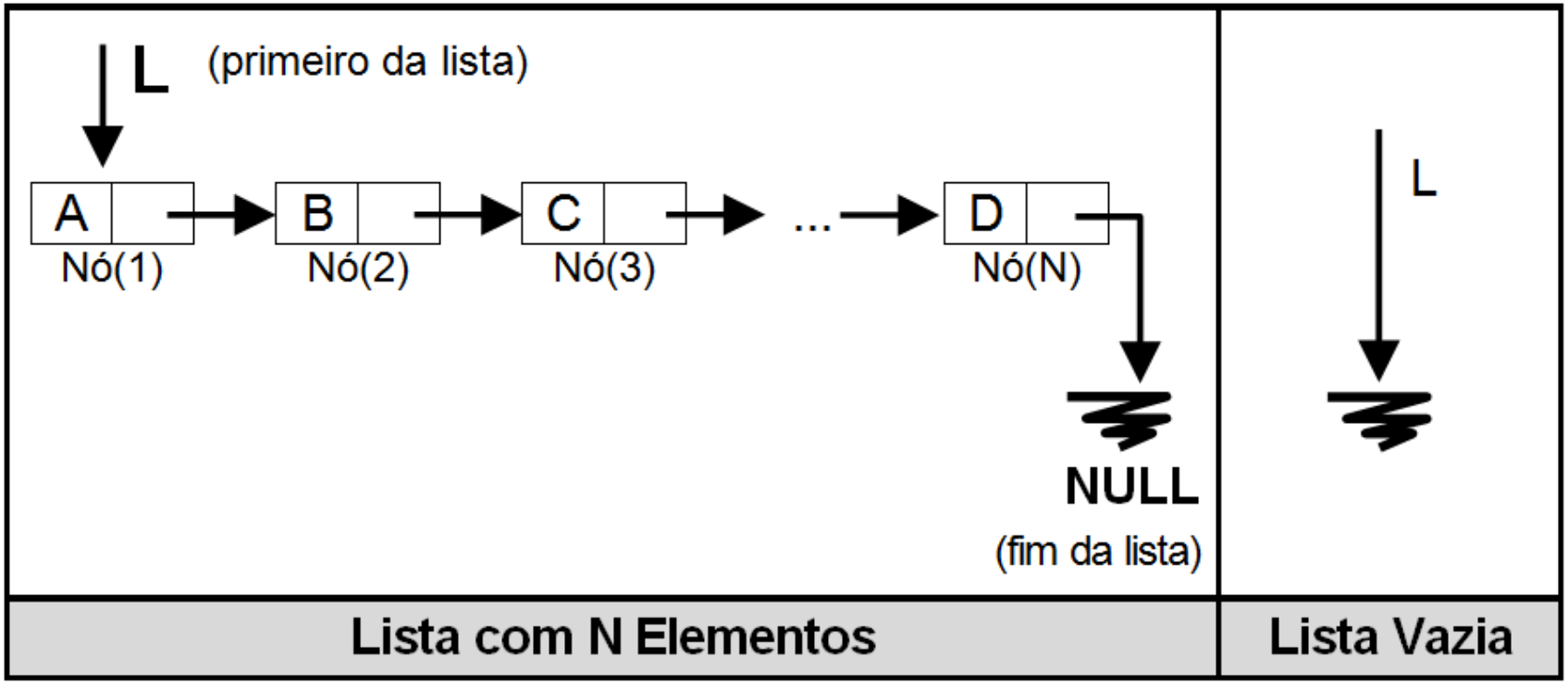


Definição: Alocação Encadeada de Memória para um Conjunto de Elementos

Os elementos não são armazenados, necessariamente, em posições de memória adjacentes;

A ordem dos elementos precisa ser explicitamente indicada: cada elemento do conjunto aponta qual é o próximo na sequencia.

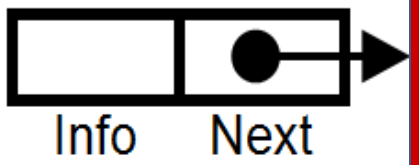
Representação Usual de uma Lista Encadeada



Notação para Manipulação de Listas Encadeadas

Definição de Tipos e Variáveis

```
Defina o tipo Node = Registro {
  Info do tipo Char;  // campo usado para armazenar informação
  Next do tipo ponteiro para Node;  // indica o próximo elemento da Lista
}
```




```
Defina o tipo NodePtr = ponteiro para Node;

Variáveis L, P, P1, P2 do tipo NodePtr;    // variáveis do mesmo tipo que o campo Next
Variável X do tipo Char;                   // X é do mesmo tipo que o campo Info
```

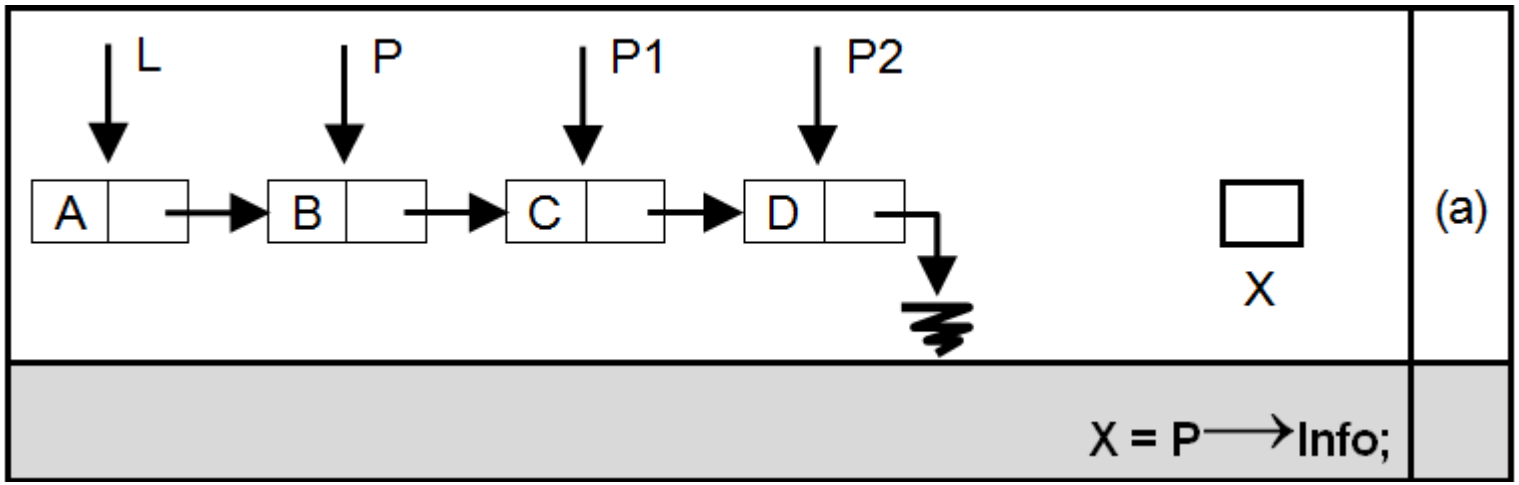
Notação para Manipulação de Listas Encadeadas

Operações

X = P→Info;	X recebe o valor do campo Info do nó apontado por P.
P→Info = X;	O campo Info do Nó apontado por P recebe o valor de X.
P1 = P2;	O ponteiro P1 passa a apontar para onde aponta P2.
P1 = P→Next;	P1 passa a apontar para onde aponta o campo Next do Nó apontado por P.
P→Next = P1;	O campo Next do Nó apontado por P passa a apontar para onde aponta P1.
P = NewNode;	Aloca um Nó e retorna o endereço em P.
DeleteNode(P);	Libera (desaloca) o Nó apontado por P.

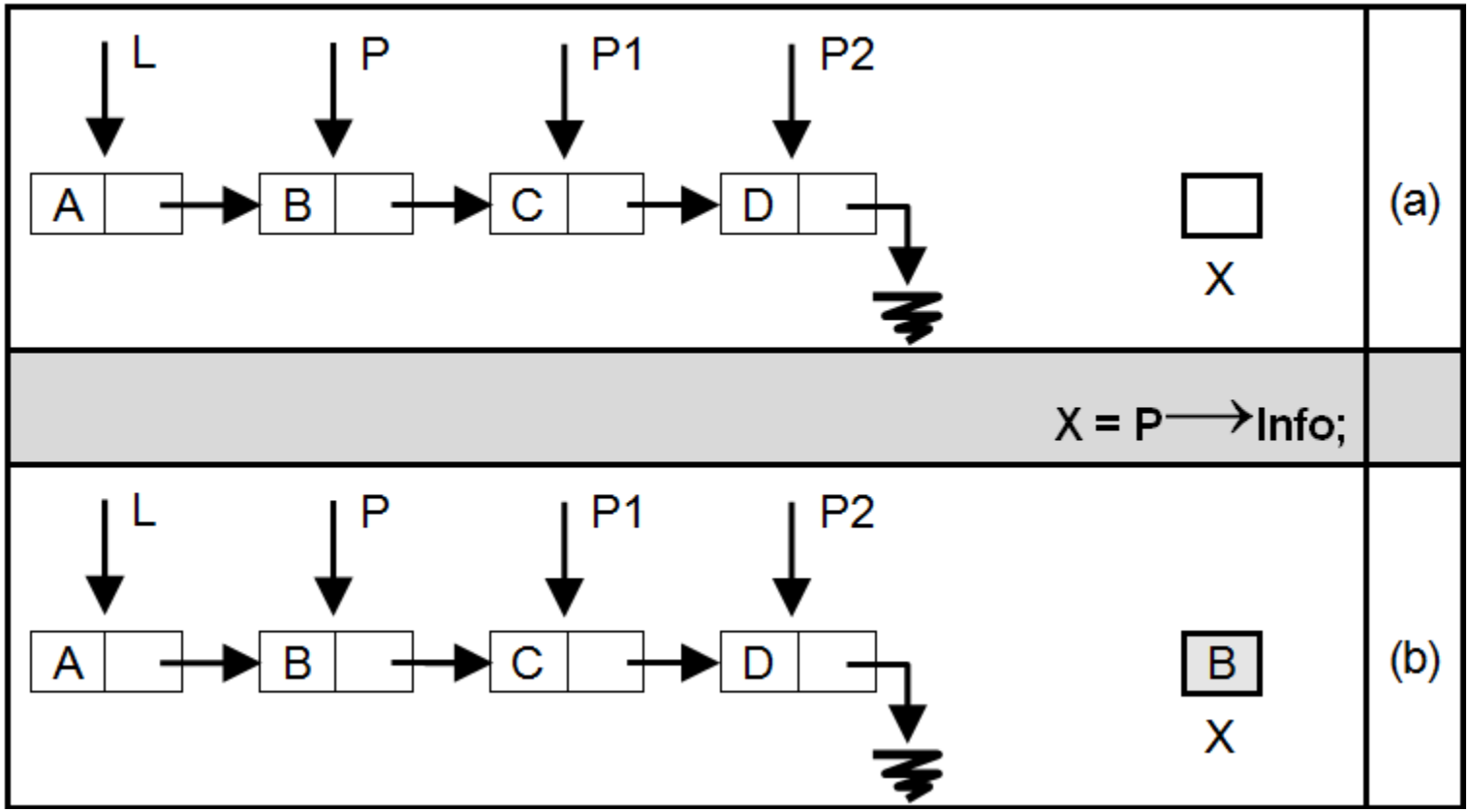
Notação para Manipulação de Listas Encadeadas

Acessando e Atualizando o Campo *Info* de um Nó



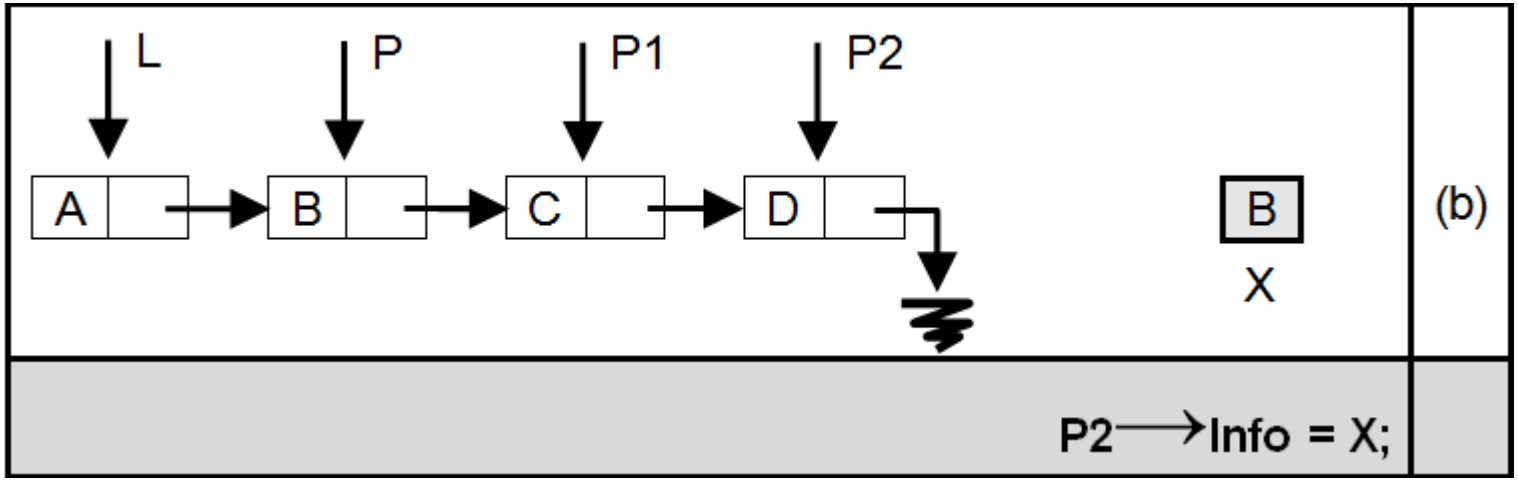
Notação para Manipulação de Listas Encadeadas

Acessando e Atualizando o Campo *Info* de um Nó



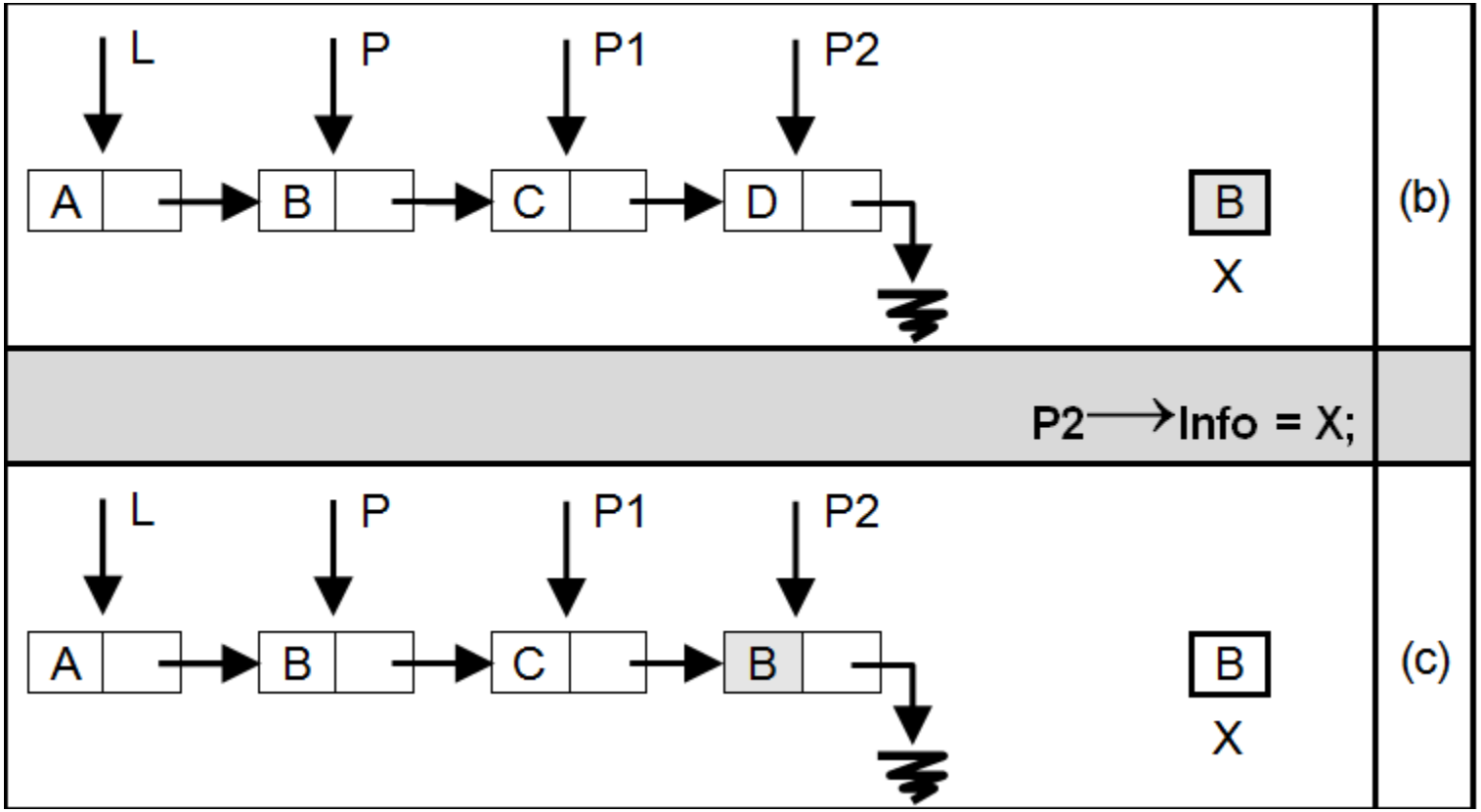
Notação para Manipulação de Listas Encadeadas

Acessando e Atualizando o Campo *Info* de um Nó



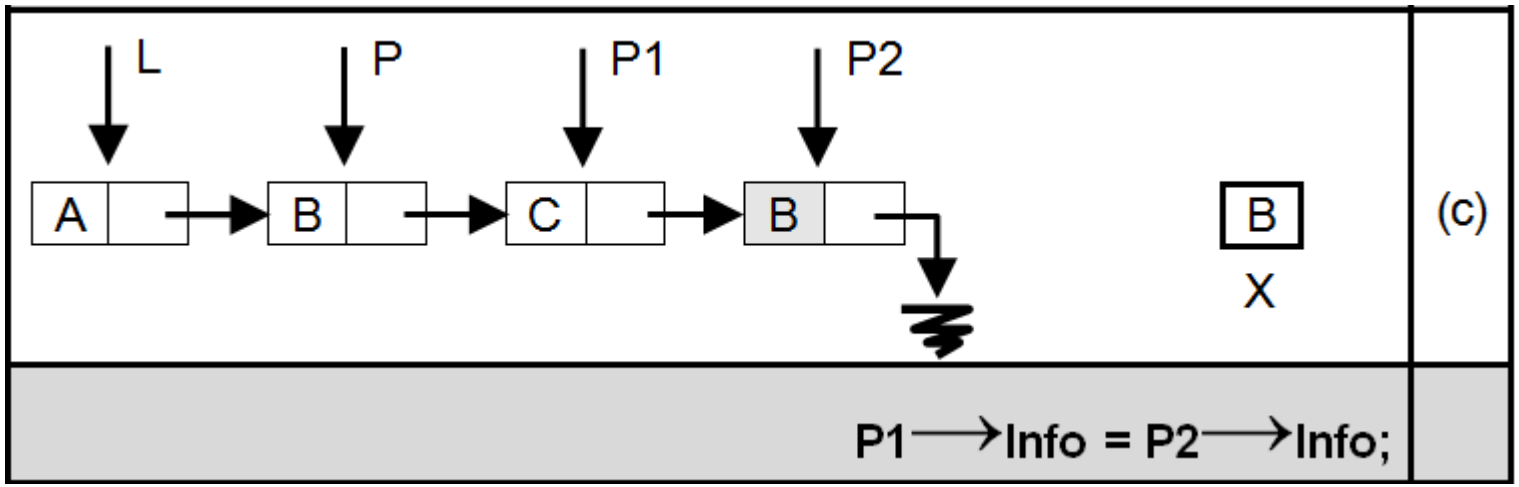
Notação para Manipulação de Listas Encadeadas

Acessando e Atualizando o Campo *Info* de um Nó



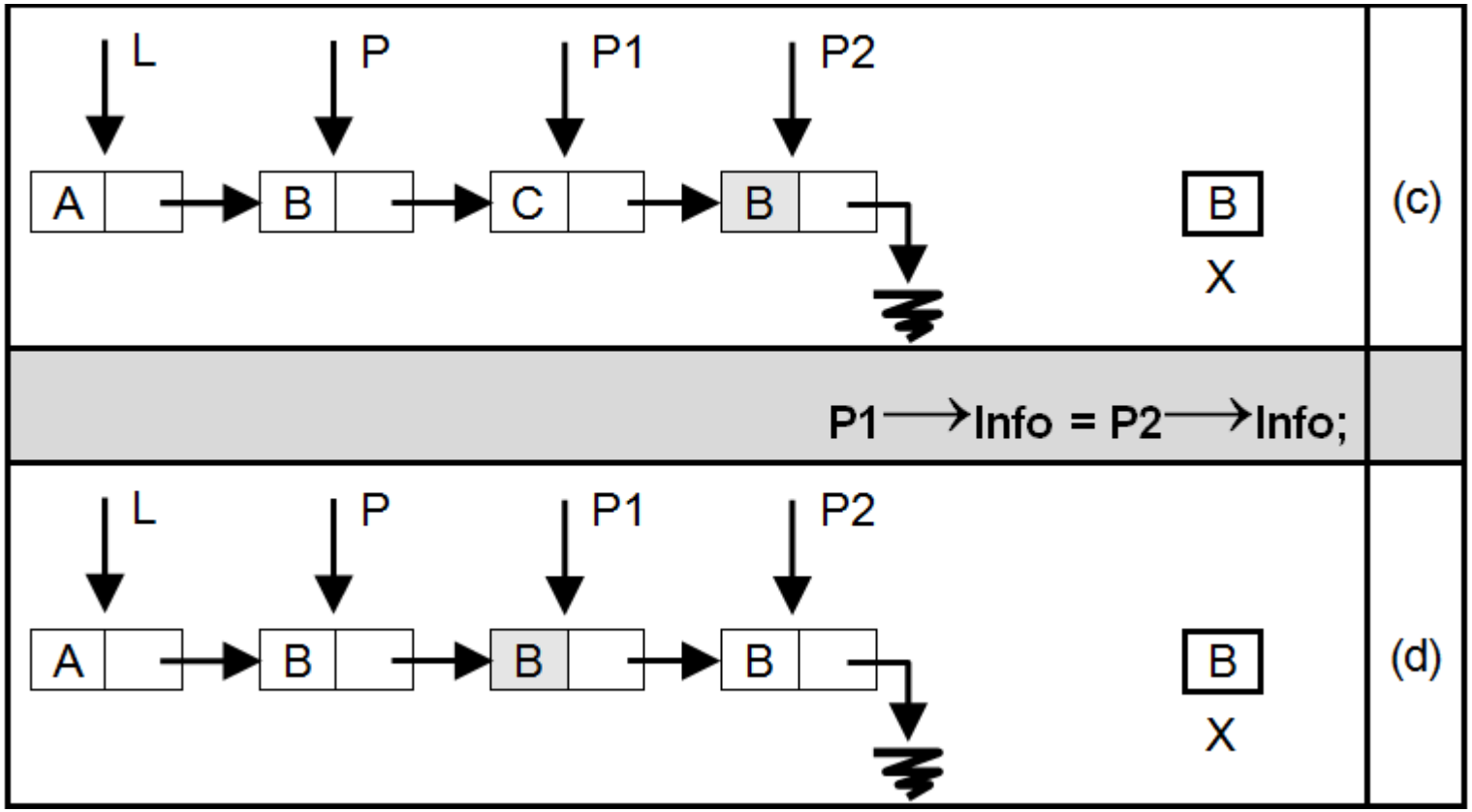
Notação para Manipulação de Listas Encadeadas

Acessando e Atualizando o Campo *Info* de um Nó



Notação para Manipulação de Listas Encadeadas

Acessando e Atualizando o Campo *Info* de um Nó

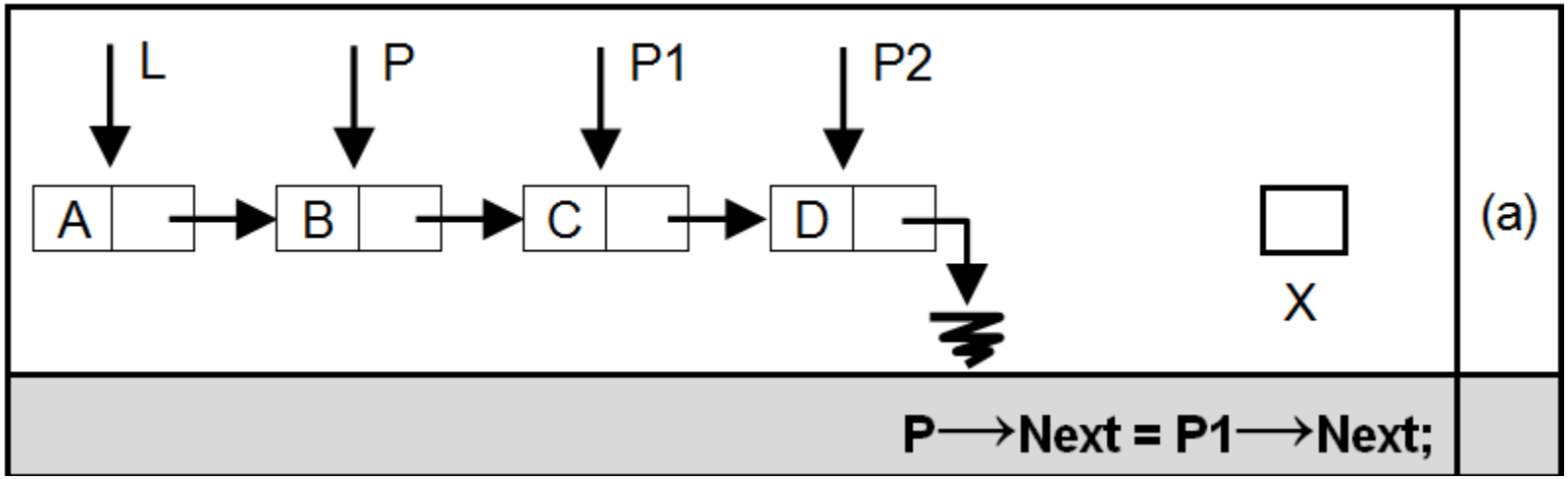


Notação para Manipulação de Listas Encadeadas

Operações

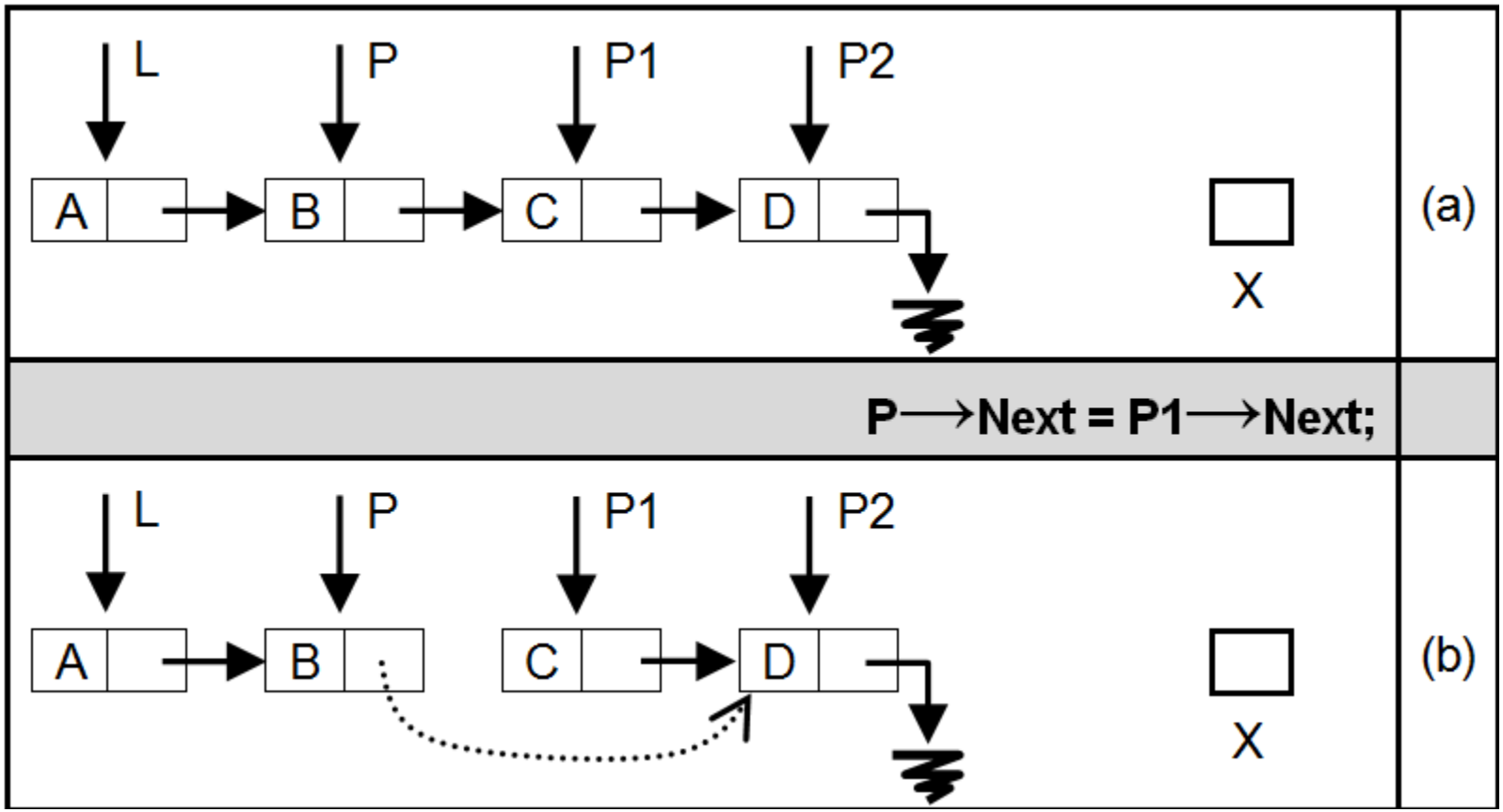
X = P→Info;	X recebe o valor do campo Info do nó apontado por P.
P→Info = X;	O campo Info do Nó apontado por P recebe o valor de X.
P1 = P2;	O ponteiro P1 passa a apontar para onde aponta P2.
P1 = P→Next;	P1 passa a apontar para onde aponta o campo Next do Nó apontado por P.
P→Next = P1;	O campo Next do Nó apontado por P passa a apontar para onde aponta P1.
P = NewNode;	Aloca um Nó e retorna o endereço em P.
DeleteNode(P);	Libera (desaloca) o Nó apontado por P.

Notação para Manipulação de Listas Encadeadas



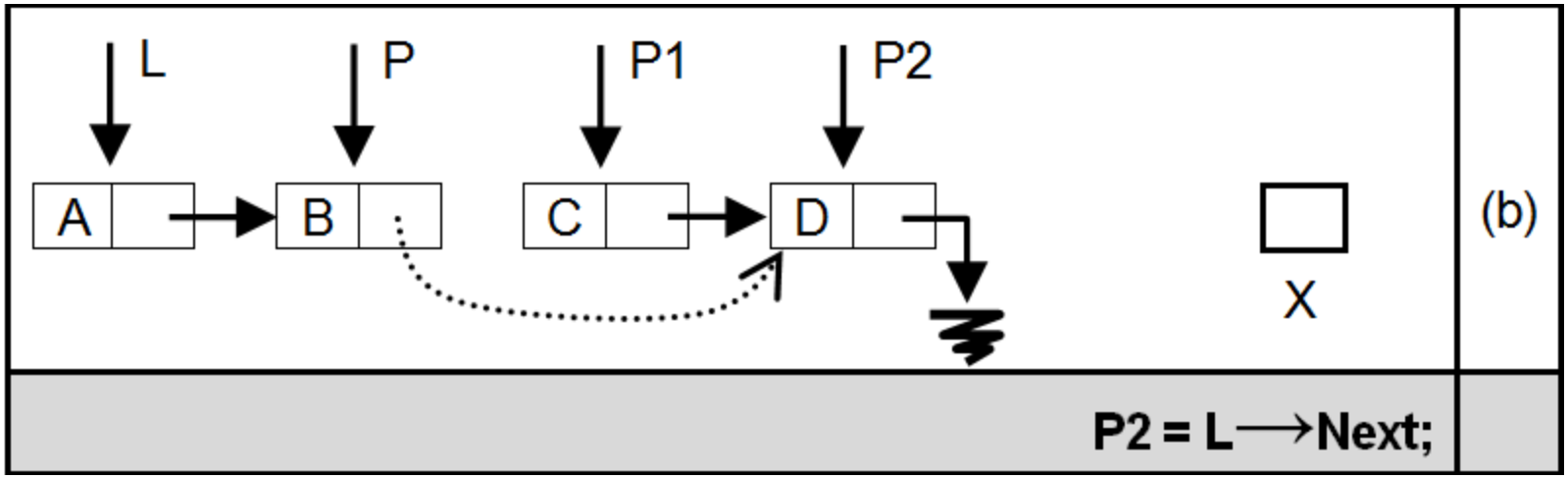
Acessando e Atualizando o Campo *Next* de um Nó

Notação para Manipulação de Listas Encadeadas



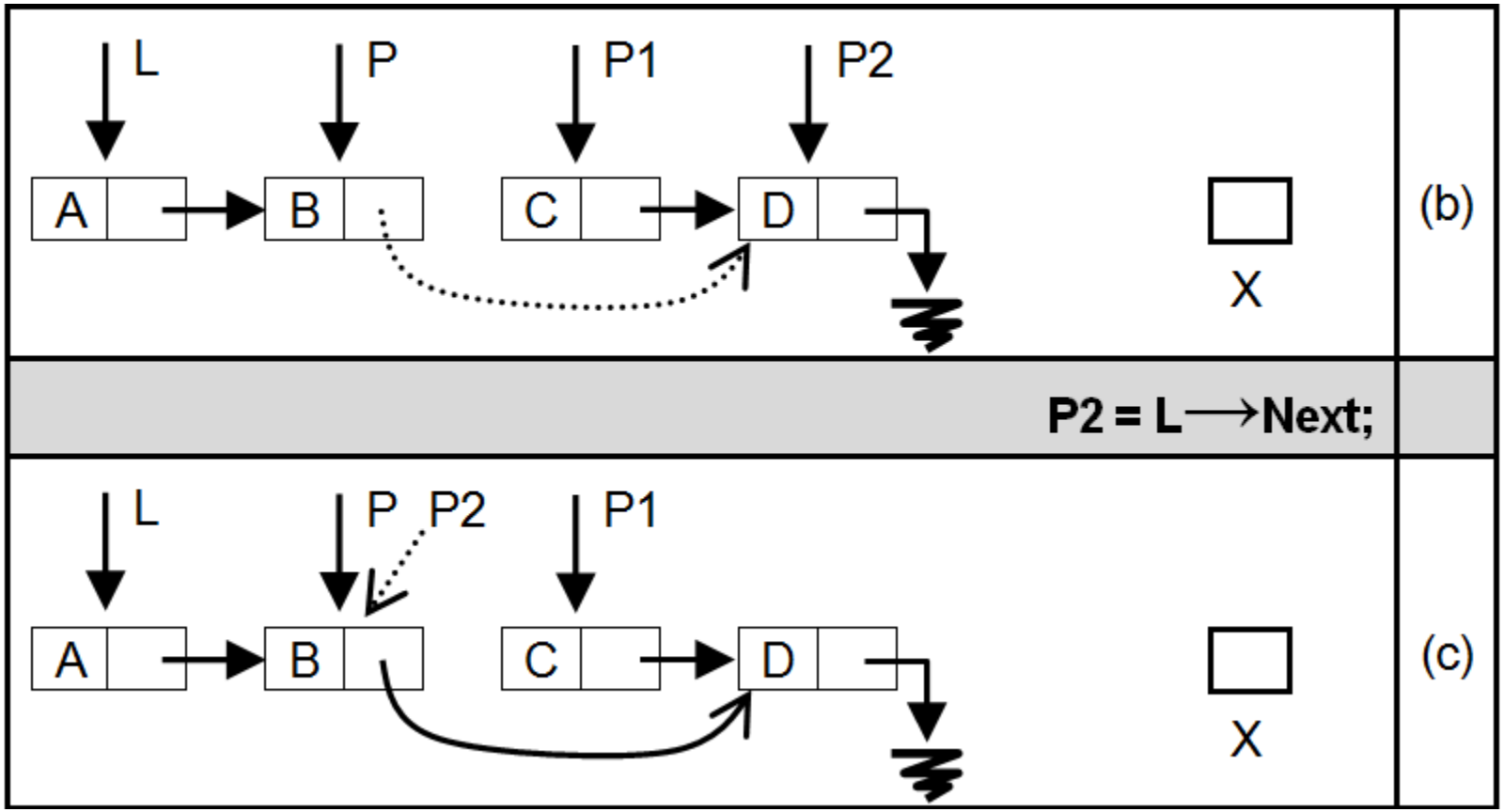
Acessando e Atualizando o Campo *Next* de um Nó

Notação para Manipulação de Listas Encadeadas



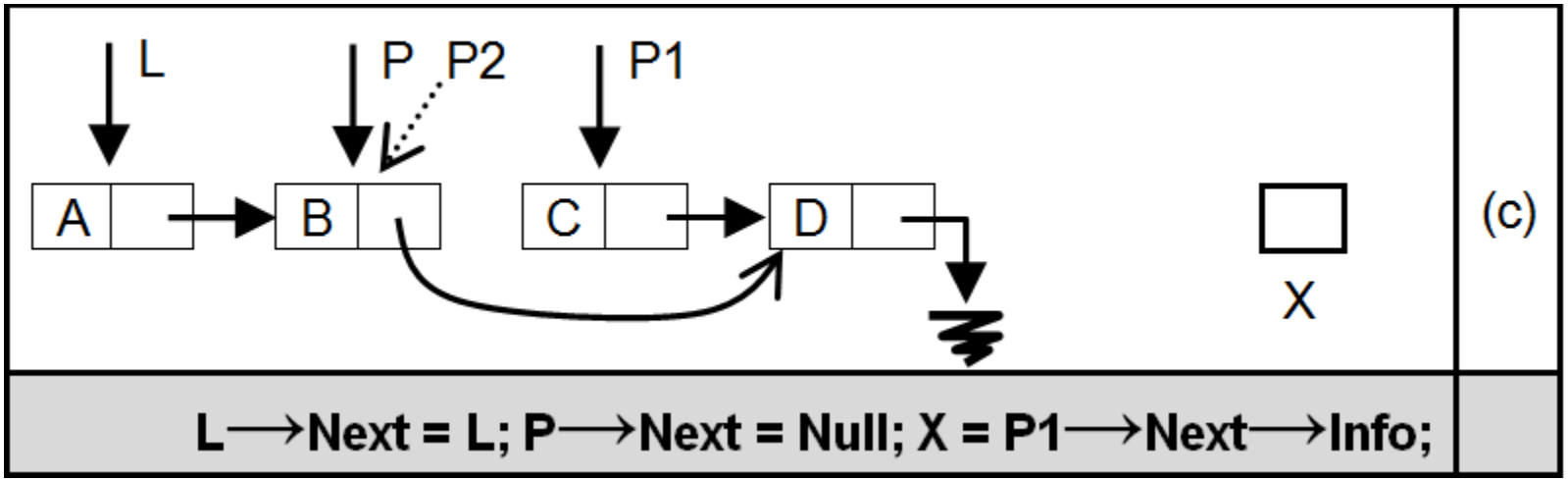
Acessando e Atualizando o Campo *Next* de um Nó

Notação para Manipulação de Listas Encadeadas



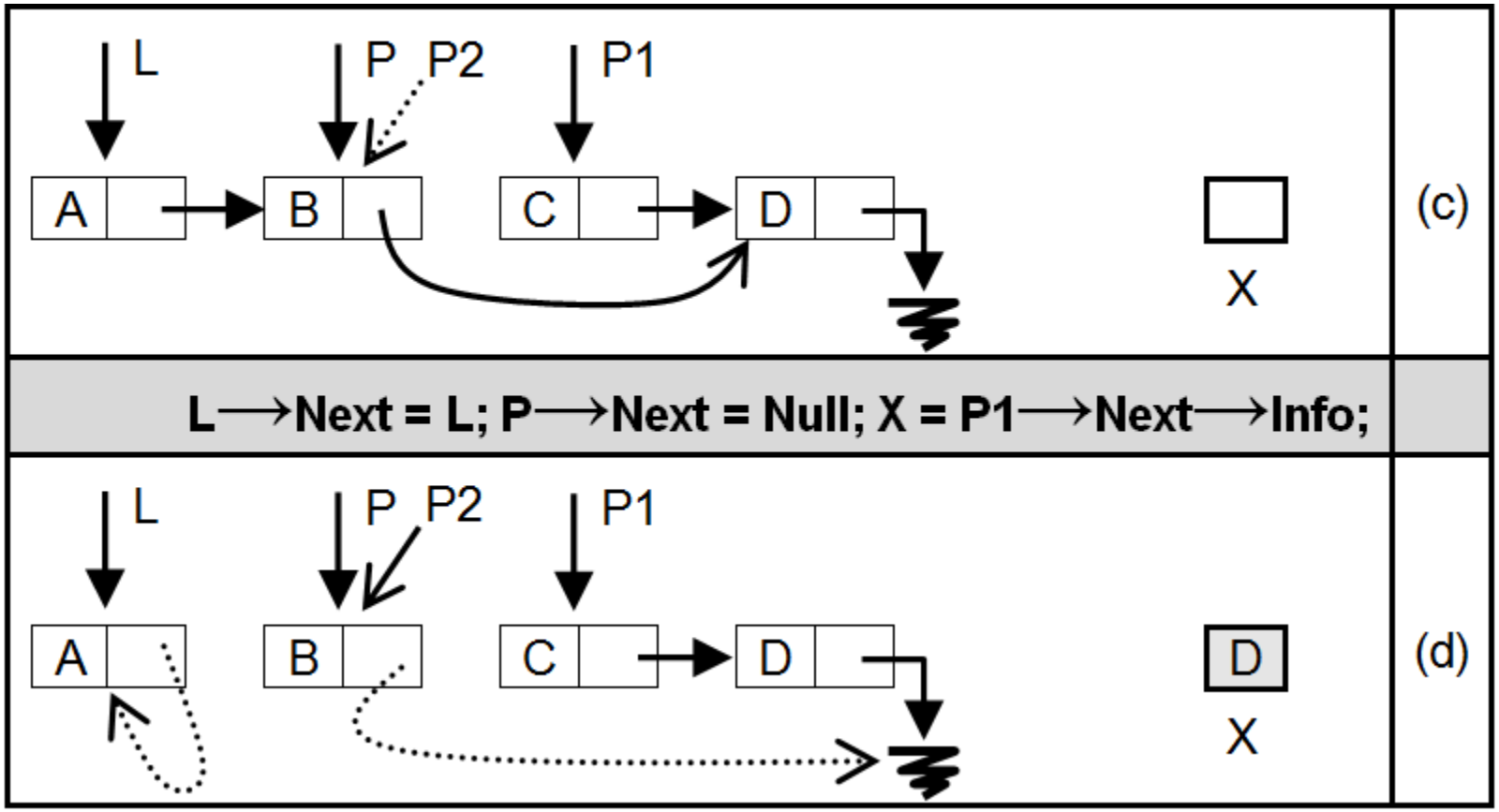
Acessando e Atualizando o Campo *Next* de um Nó

Notação para Manipulação de Listas Encadeadas



Acessando e Atualizando o Campo *Next* de um Nó

Notação para Manipulação de Listas Encadeadas



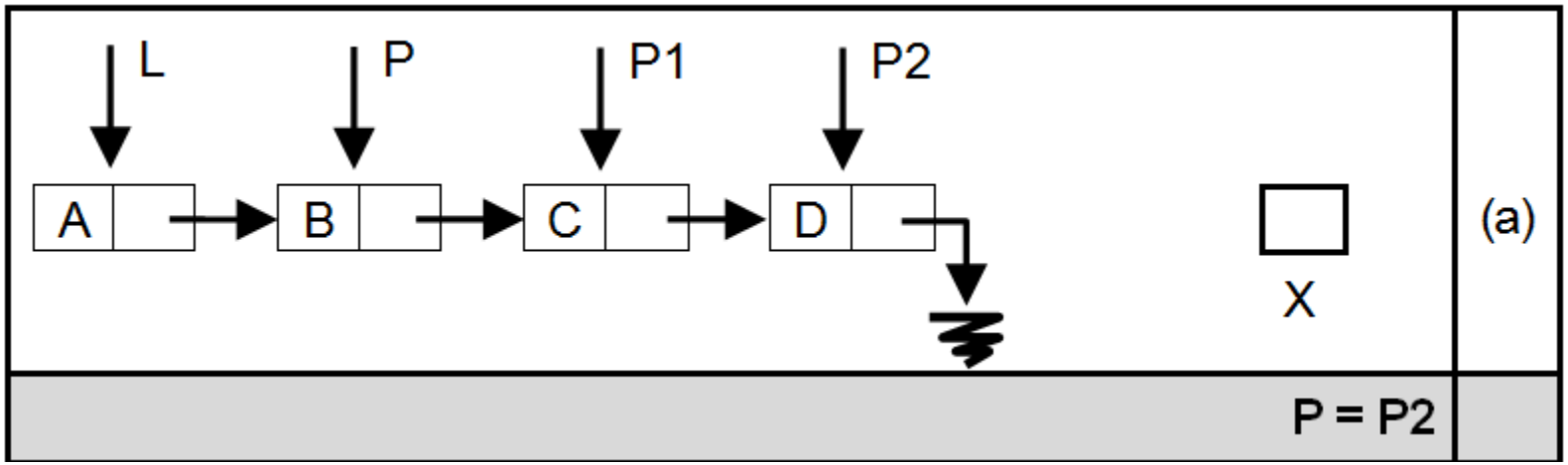
Acessando e Atualizando o Campo *Next* de um Nó

Notação para Manipulação de Listas Encadeadas

Operações

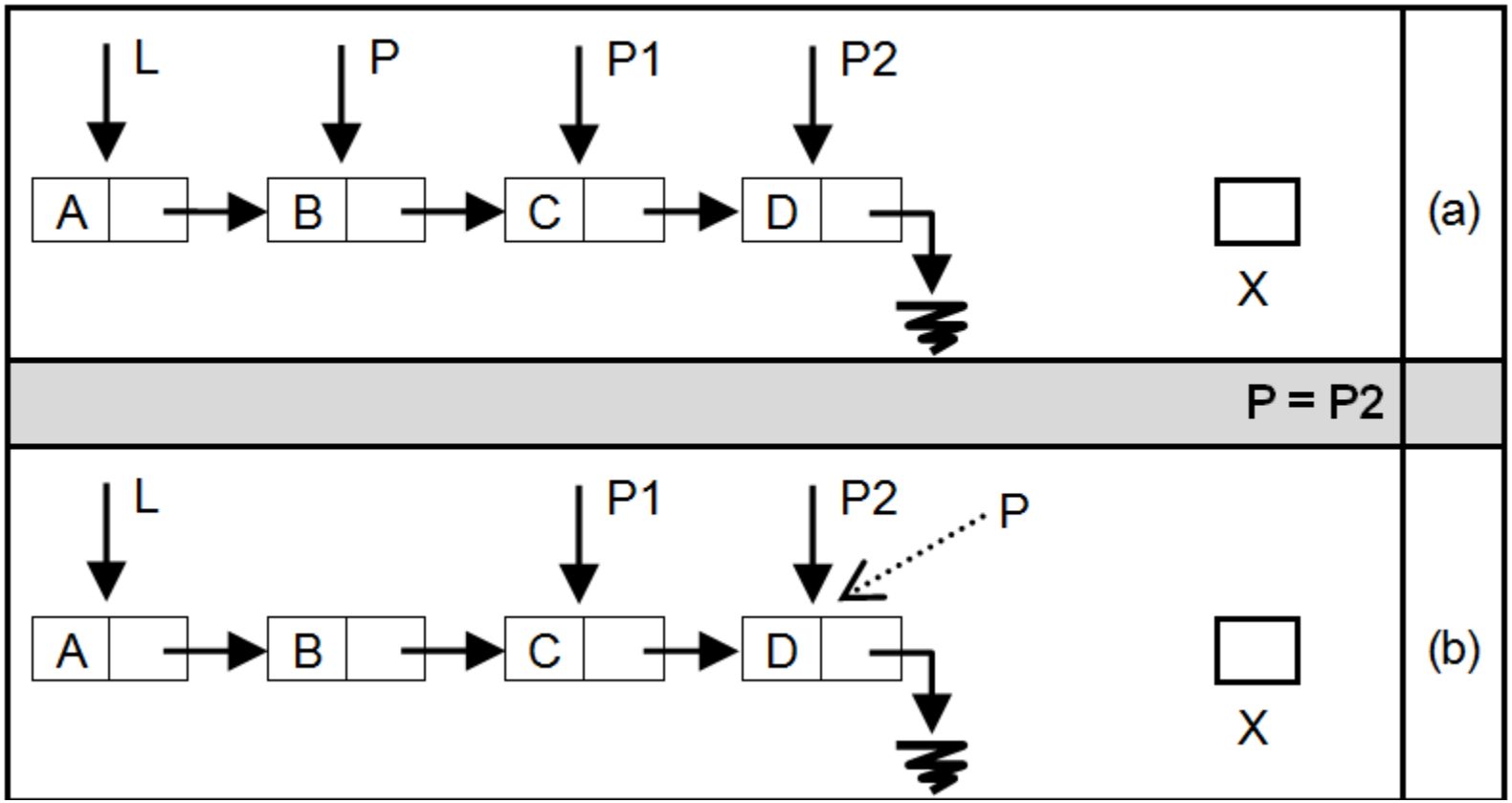
X = P→Info;	X recebe o valor do campo Info do nó apontado por P.
P→Info = X;	O campo Info do Nó apontado por P recebe o valor de X.
P1 = P2;	O ponteiro P1 passa a apontar para onde aponta P2.
P1 = P→Next;	P1 passa a apontar para onde aponta o campo Next do Nó apontado por P.
P→Next = P1;	O campo Next do Nó apontado por P passa a apontar para onde aponta P1.
P = NewNode;	Aloca um Nó e retorna o endereço em P.
DeleteNode(P);	Libera (desaloca) o Nó apontado por P.

Notação para Manipulação de Listas Encadeadas



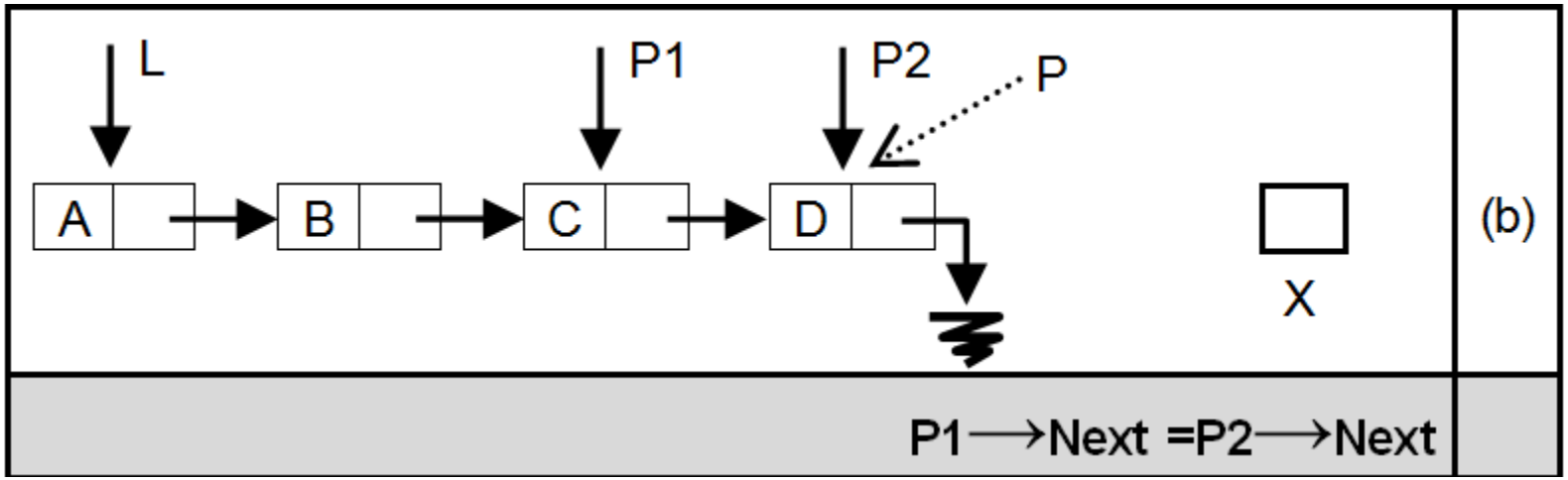
Movendo Ponteiros

Notação para Manipulação de Listas Encadeadas



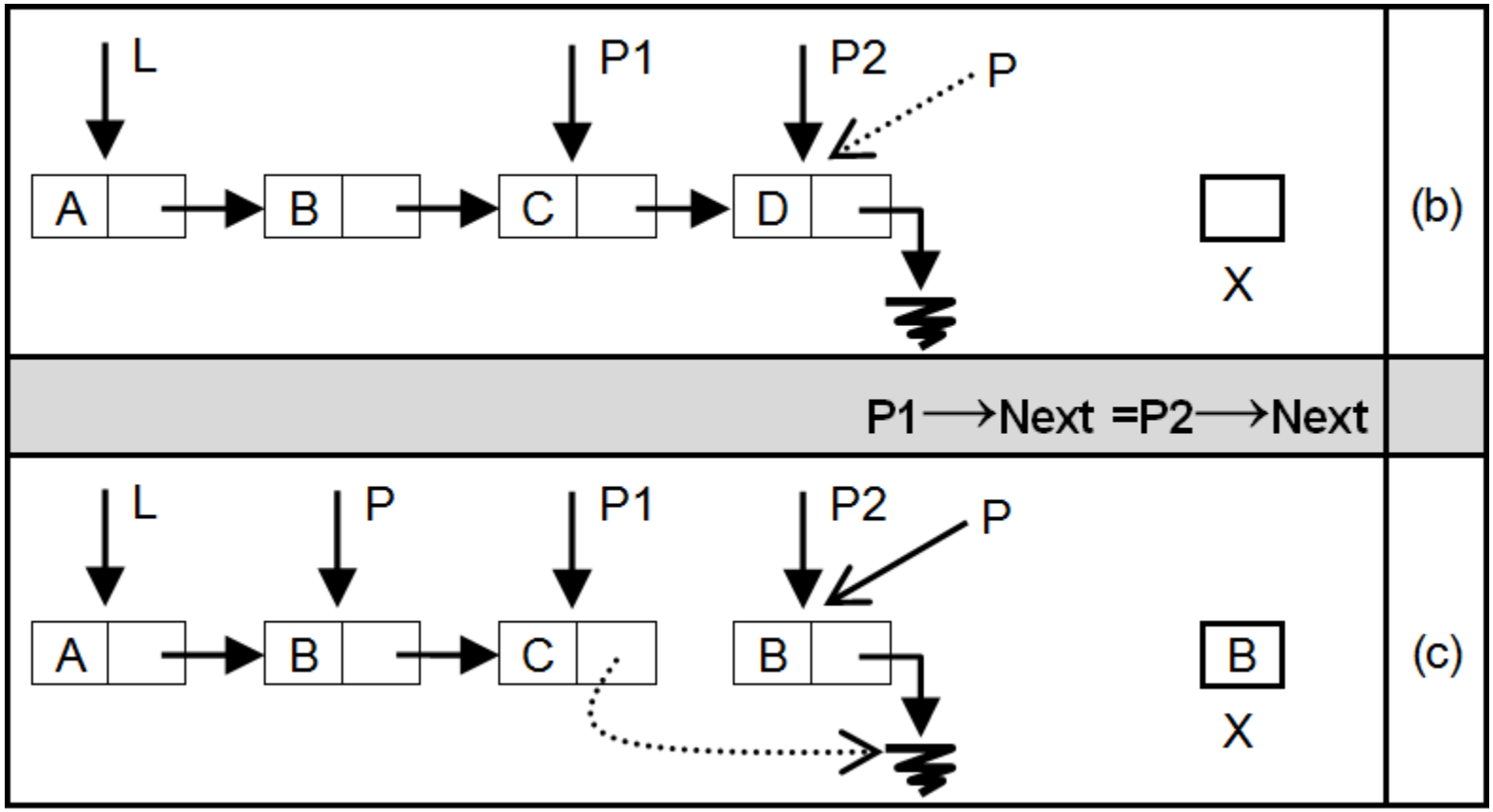
Movendo Ponteiros

Notação para Manipulação de Listas Encadeadas



Movendo Ponteiros

Notação para Manipulação de Listas Encadeadas



Movendo Ponteiros

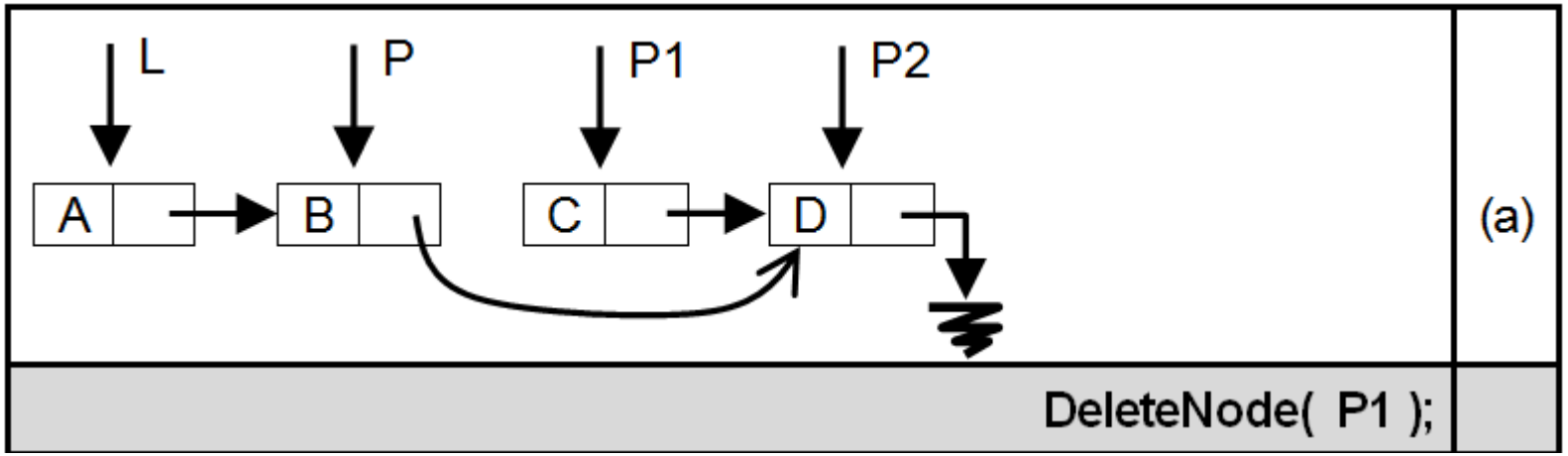
Notação para Manipulação de Listas Encadeadas

Operações

X = P→Info;	X recebe o valor do campo Info do nó apontado por P.
P→Info = X;	O campo Info do Nó apontado por P recebe o valor de X.
P1 = P2;	O ponteiro P1 passa a apontar para onde aponta P2.
P1 = P→Next;	P1 passa a apontar para onde aponta o campo Next do Nó apontado por P.
P→Next = P1;	O campo Next do Nó apontado por P passa a apontar para onde aponta P1.
P = NewNode;	Aloca um Nó e retorna o endereço em P.
DeleteNode(P);	Libera (desaloca) o Nó apontado por P.

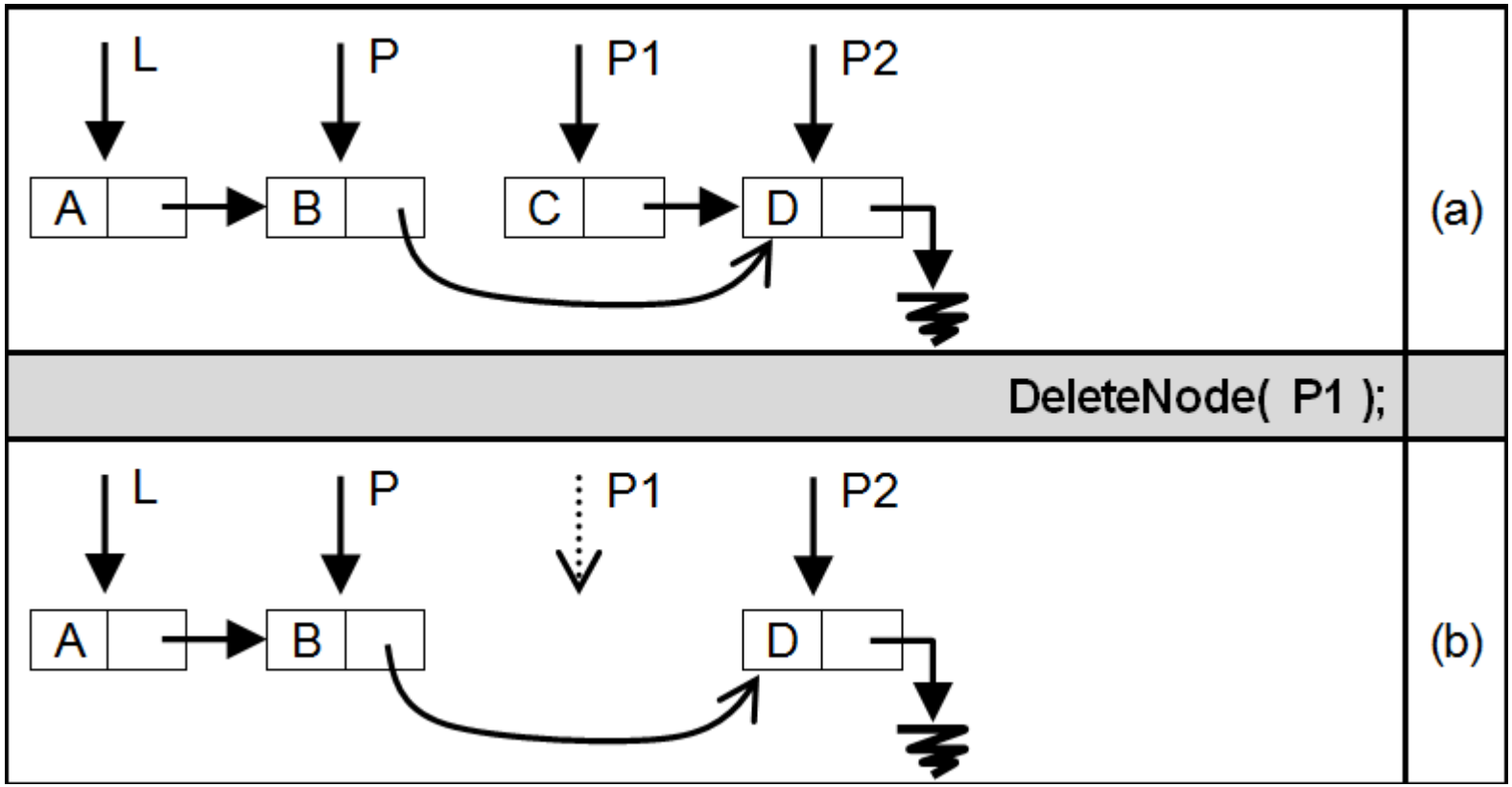
Notação para Manipulação de Listas Encadeadas

Alocando e Desalocando Nós



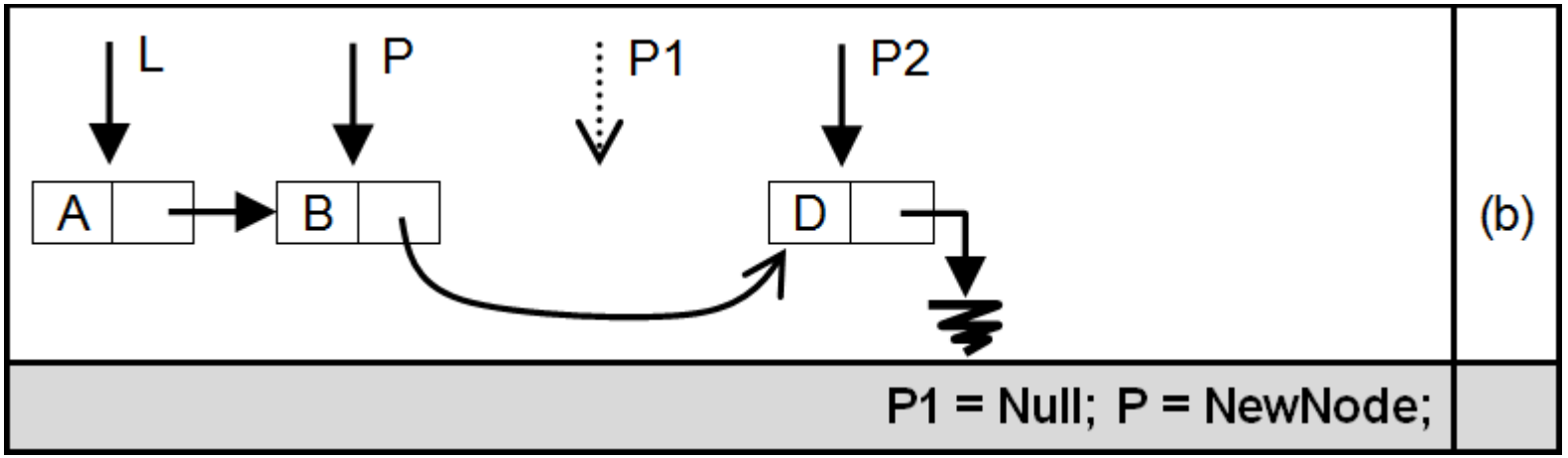
Notação para Manipulação de Listas Encadeadas

Alocando e Desalocando Nós



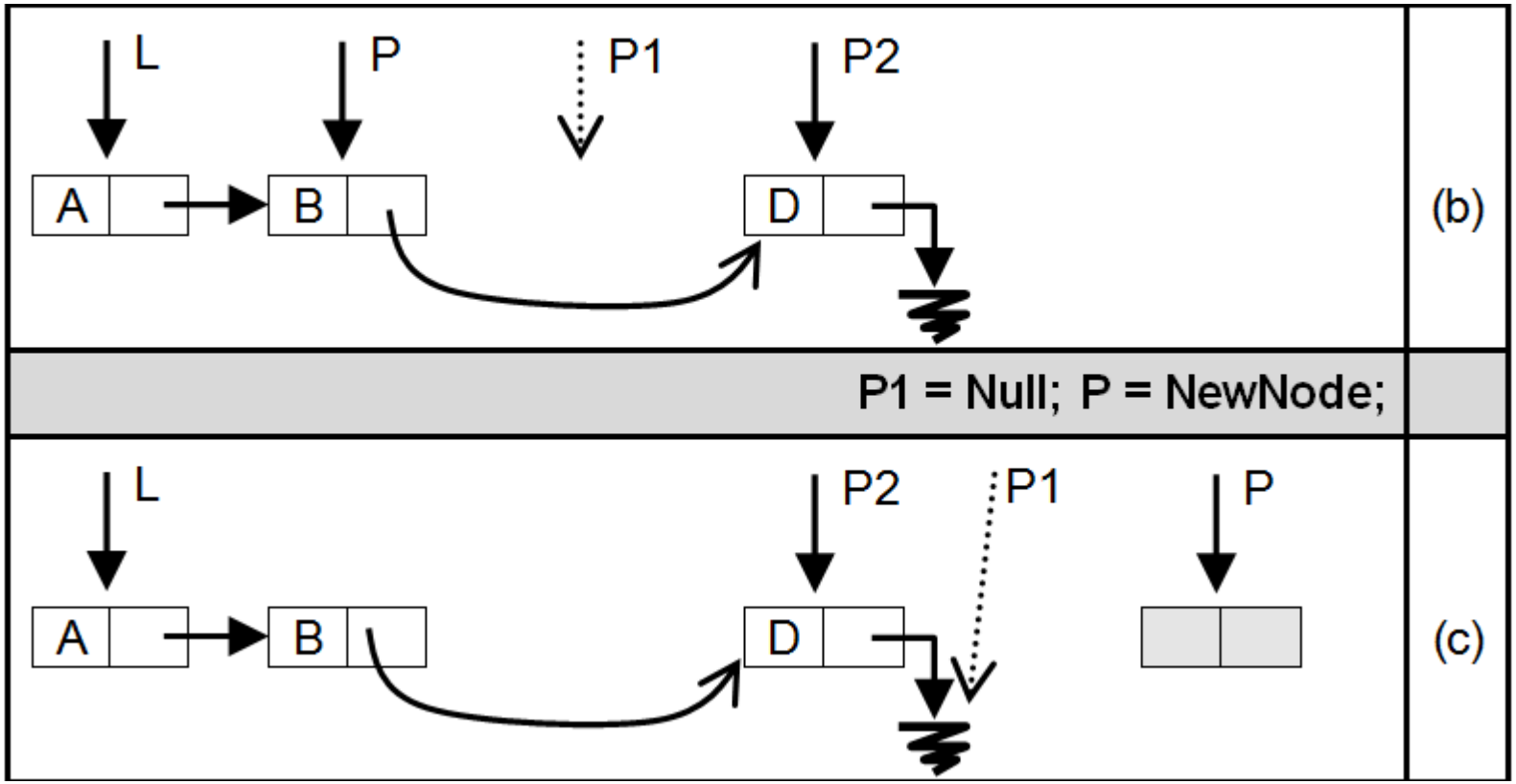
Notação para Manipulação de Listas Encadeadas

Alocando e Desalocando Nós



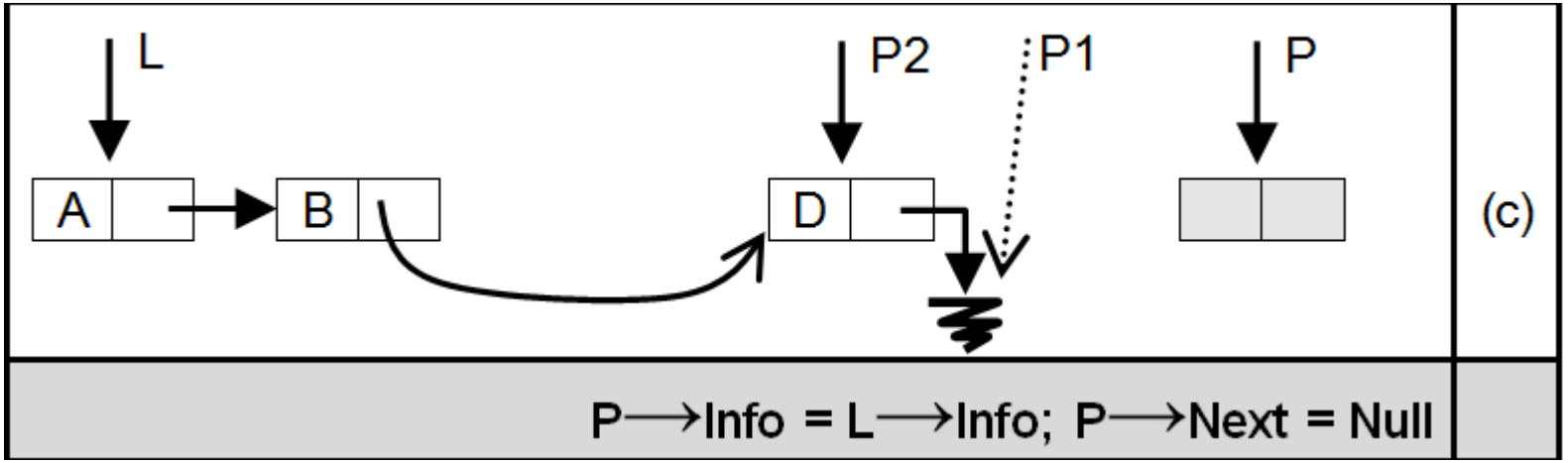
Notação para Manipulação de Listas Encadeadas

Alocando e Desalocando Nós



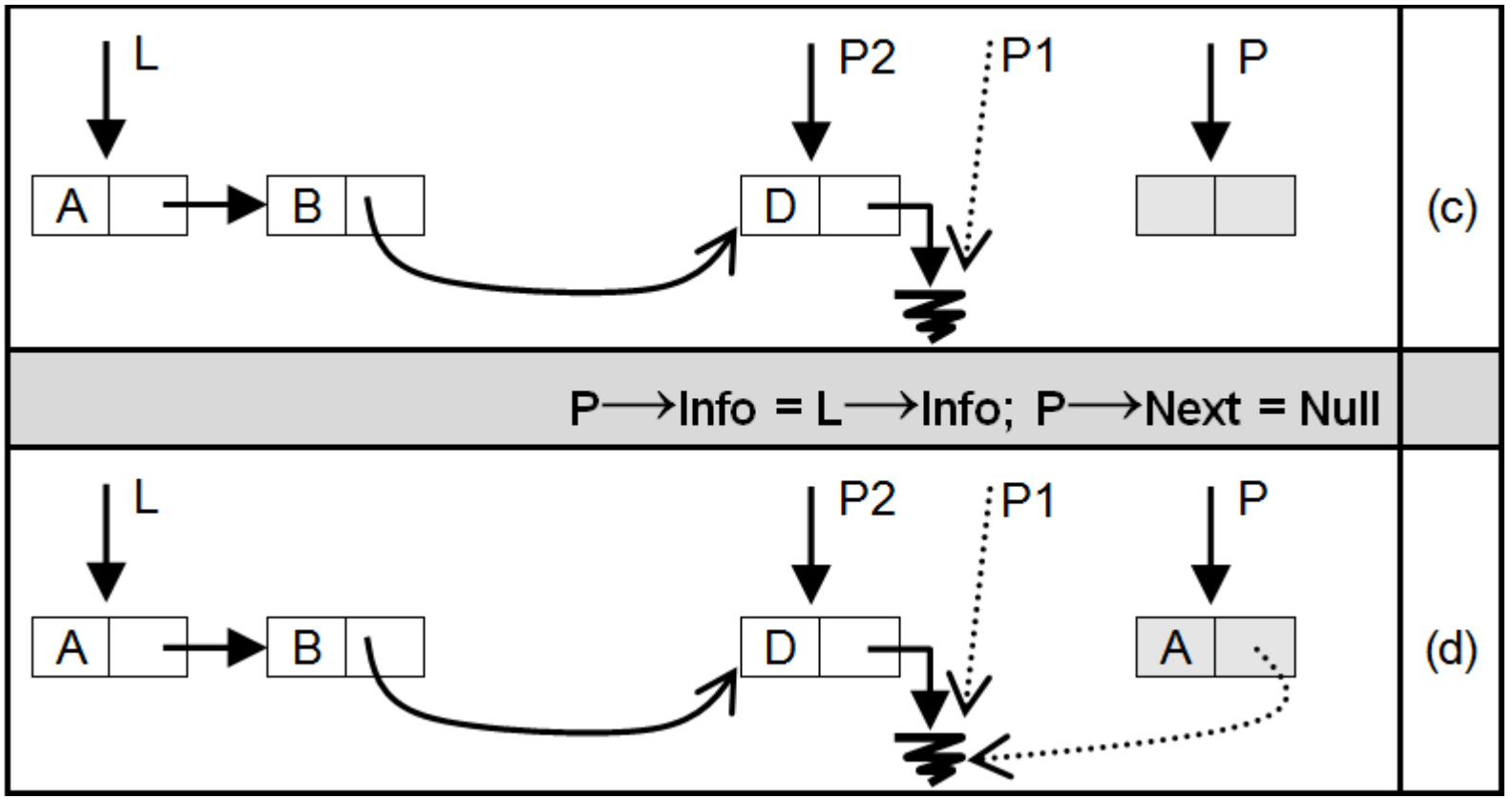
Notação para Manipulação de Listas Encadeadas

Alocando e Desalocando Nós



Notação para Manipulação de Listas Encadeadas

Alocando e Desalocando Nós

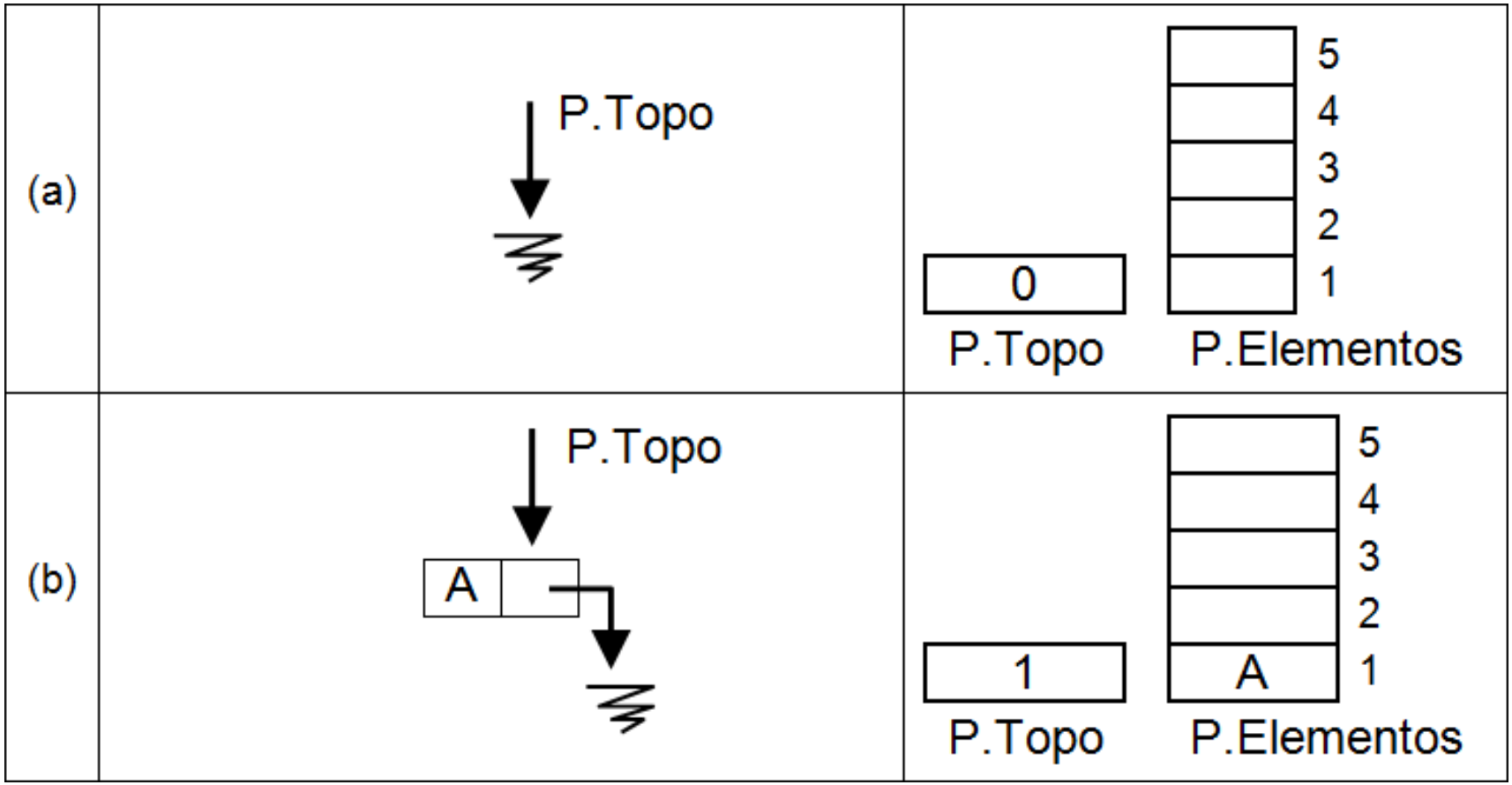


Notação para Manipulação de Listas Encadeadas

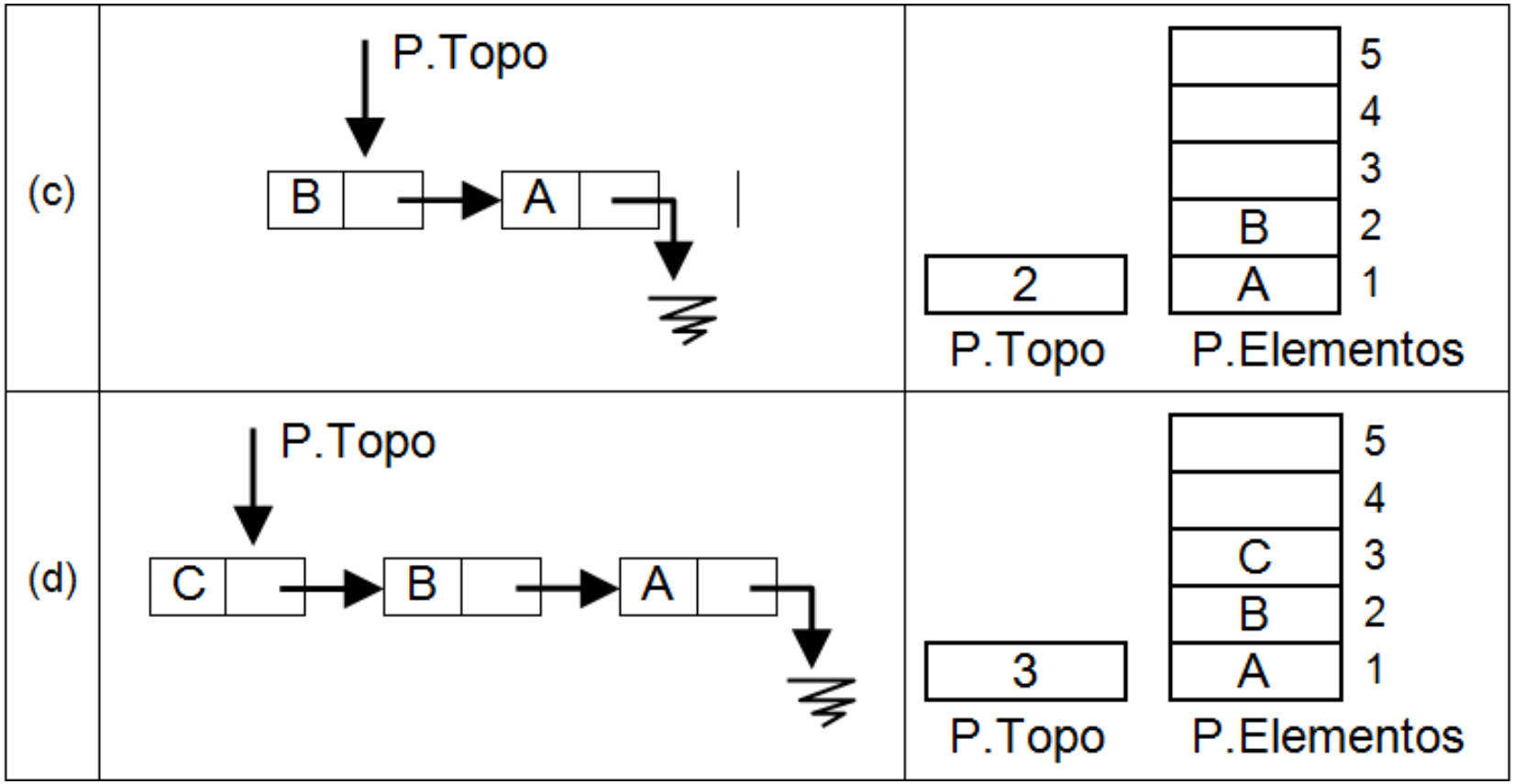
Operações

X = P→Info;	X recebe o valor do campo Info do nó apontado por P.
P→Info = X;	O campo Info do Nó apontado por P recebe o valor de X.
P1 = P2;	O ponteiro P1 passa a apontar para onde aponta P2.
P1 = P→Next;	P1 passa a apontar para onde aponta o campo Next do Nó apontado por P.
P→Next = P1;	O campo Next do Nó apontado por P passa a apontar para onde aponta P1.
P = NewNode;	Aloca um Nó e retorna o endereço em P.
DeleteNode(P);	Libera (desaloca) o Nó apontado por P.

Pilha como uma Lista Encadeada



Pilha como uma Lista Encadeada

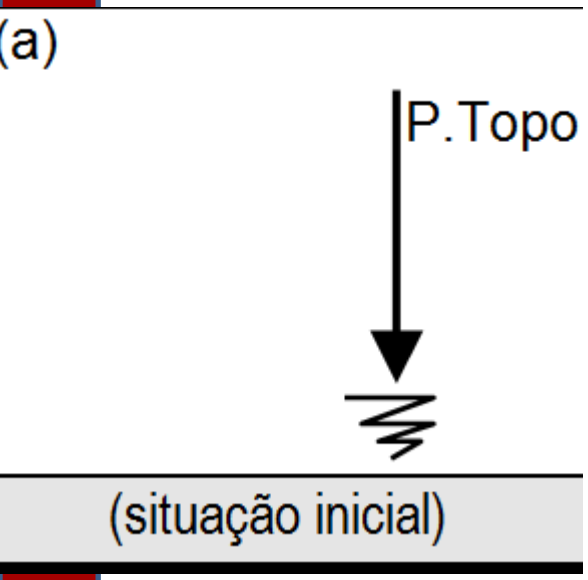


Exercício 4.1
Operação Empilha

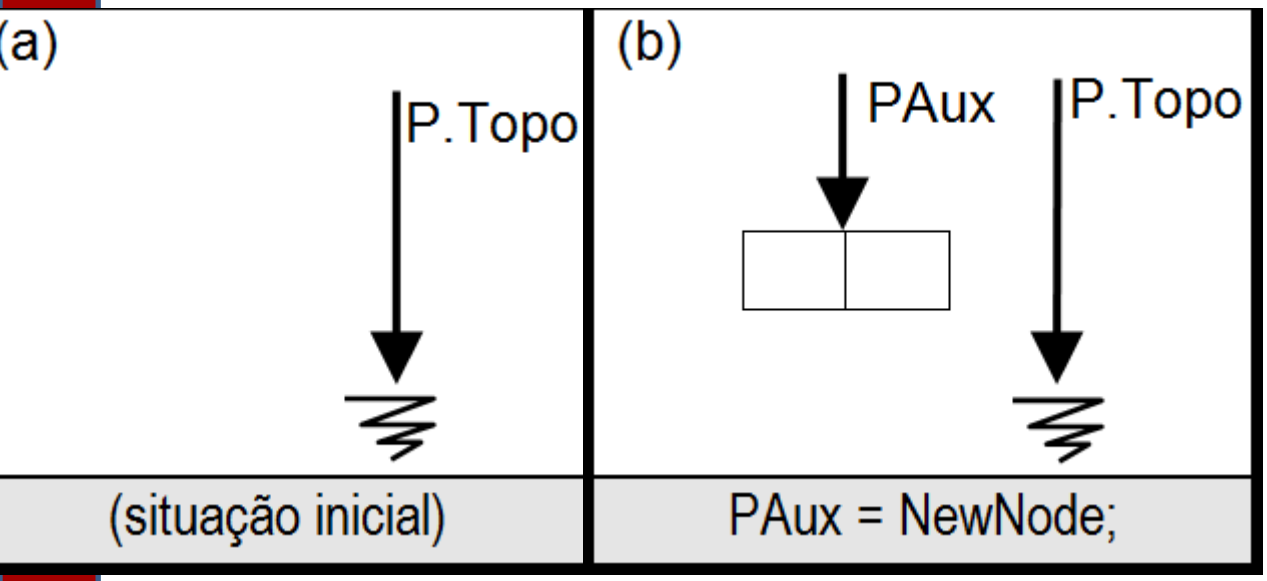
Empilha (parâmetro por referência **P** do tipo Pilha, parâmetro **X** do tipo Char, parâmetro por referência **DeuCerto** do tipo Boolean);

/* Empilha o elemento X na Pilha P. O parâmetro DeuCerto deve indicar se a operação foi bem sucedida ou não */

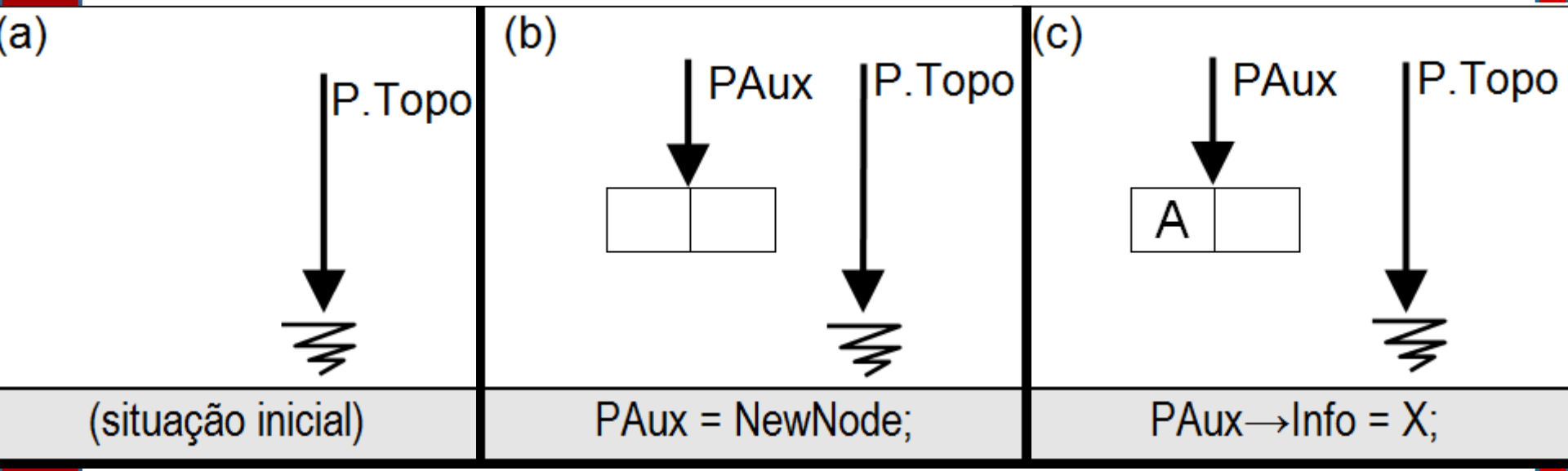
Exercício 4.1 Operação Empilha



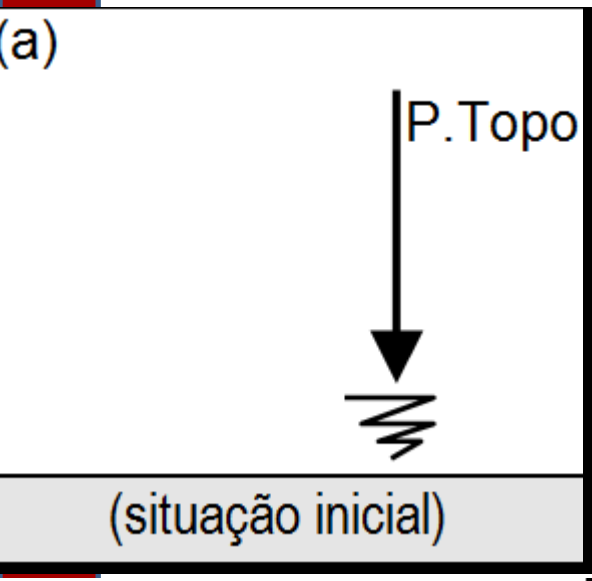
Exercício 4.1 Operação Empilha

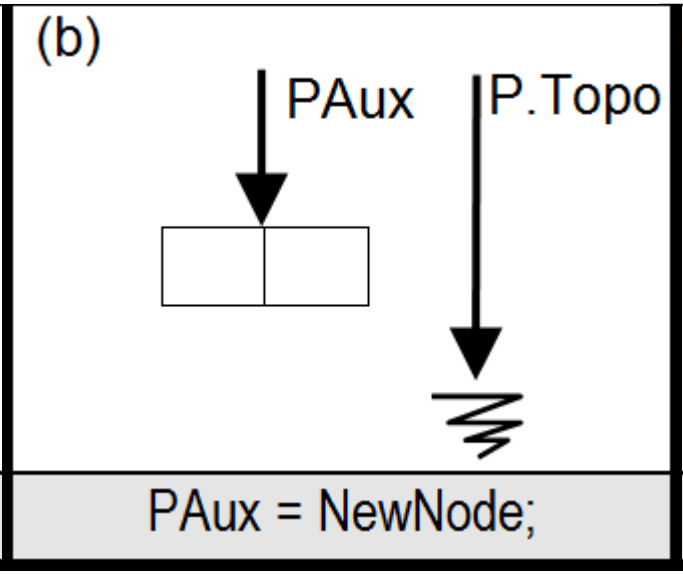


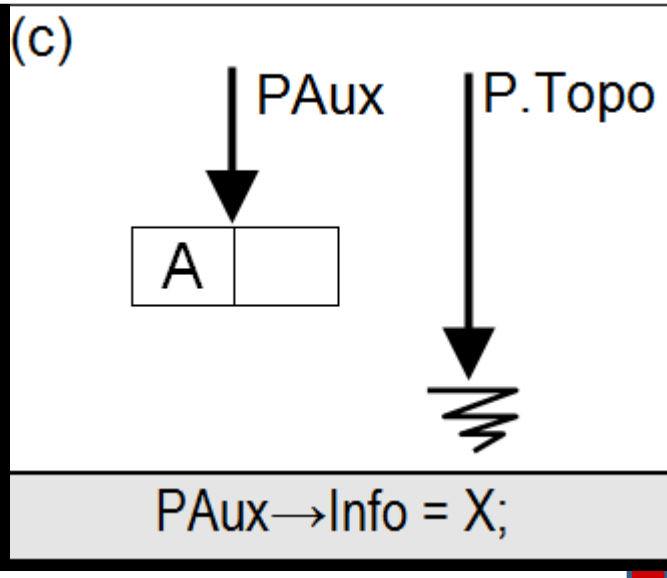
Exercício 4.1 Operação Empilha

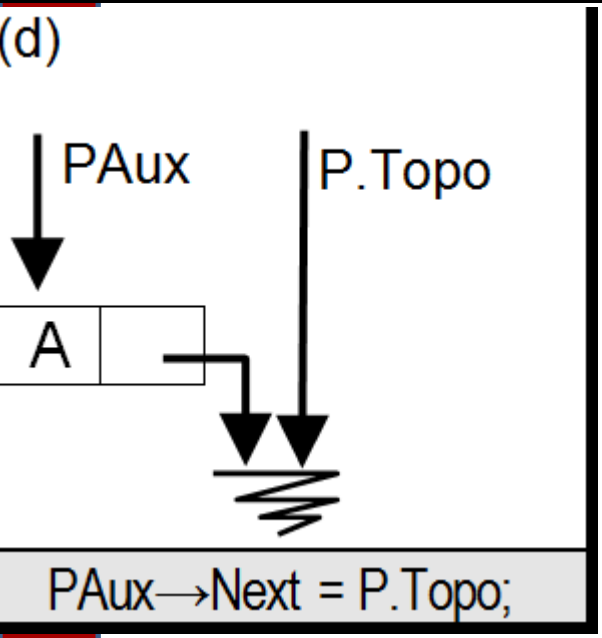


Exercício 4.1 Operação Empilha

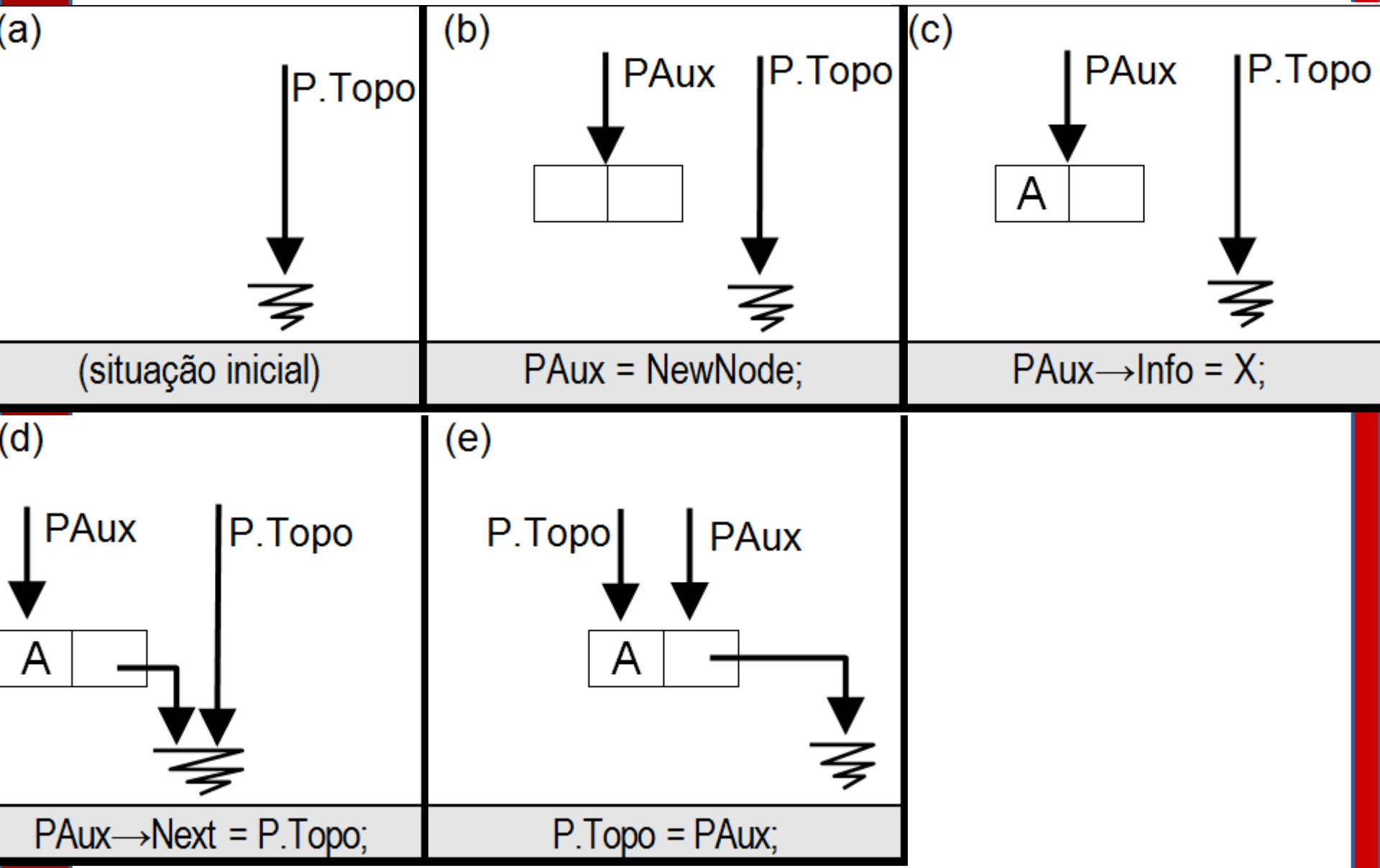




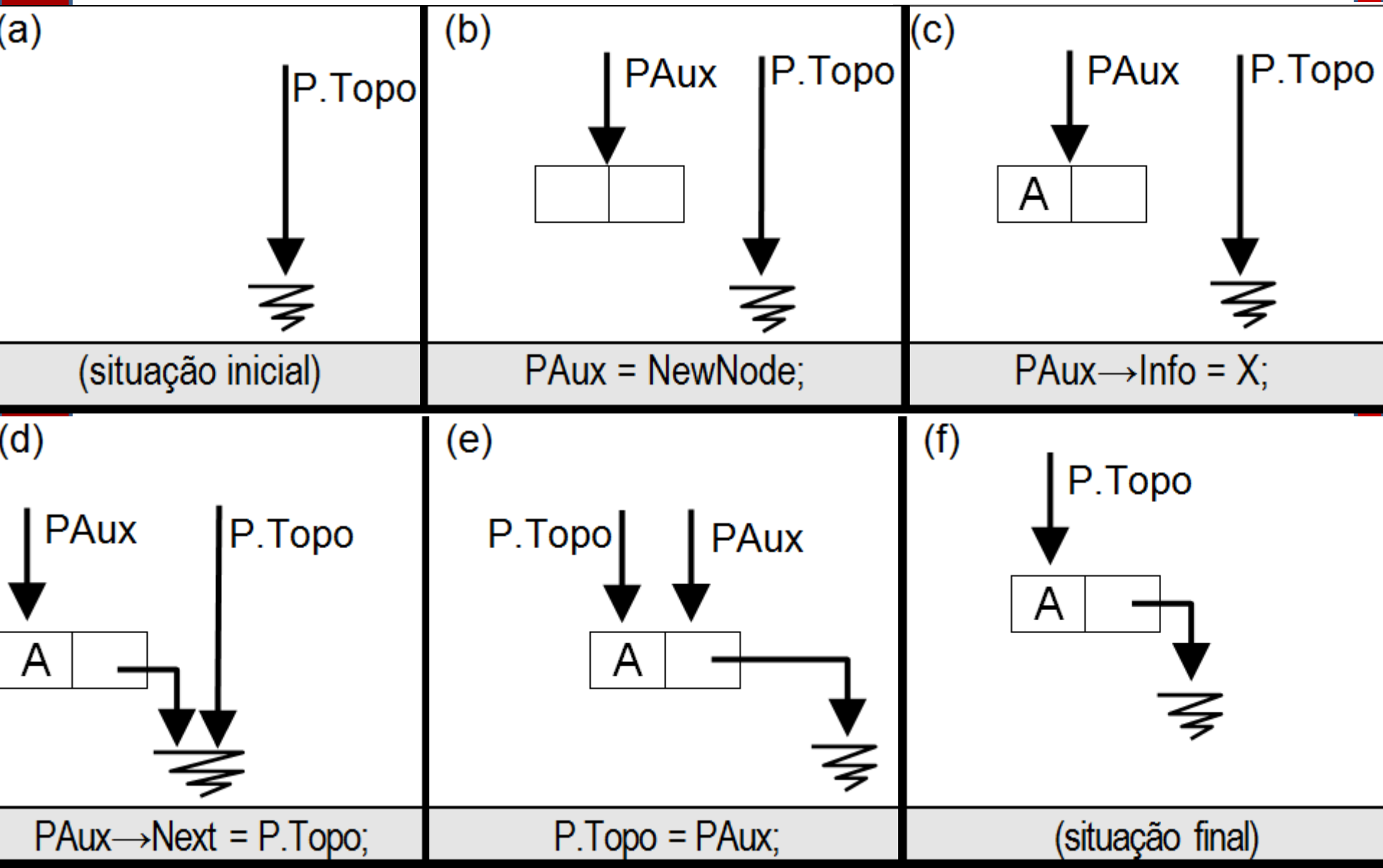




Exercício 4.1 Operação Empilha



Exercício 4.1 Operação Empilha



Exercício 4.1 Operação 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;
/* PAux é uma variável auxiliar do tipo NodePtr. O tipo Pilha, nesta implementação encadeada, contém o campo P.Topo, que também é do tipo NodePtr, ou seja, um ponteiro para Nó */

```
Se (Cheia(P)== Verdadeiro)                  // se a Pilha P estiver cheia...
Então    DeuCerto = Falso;                  // ... não podemos empilhar
Senão {  PAux = NewNode;                    // aloca um novo Nó
PAux→Info = X;                              // armazena o valor de X no novo Nó
PAux→Next = P.Topo;    // o próximo deste novo nó será o elemento do topo
P.Topo = PAux;                              // o topo da pilha passa a ser o novo Nó
         DeuCerto = Verdadeiro;};           // a operação deu certo
} // fim do procedimento Empilha
```

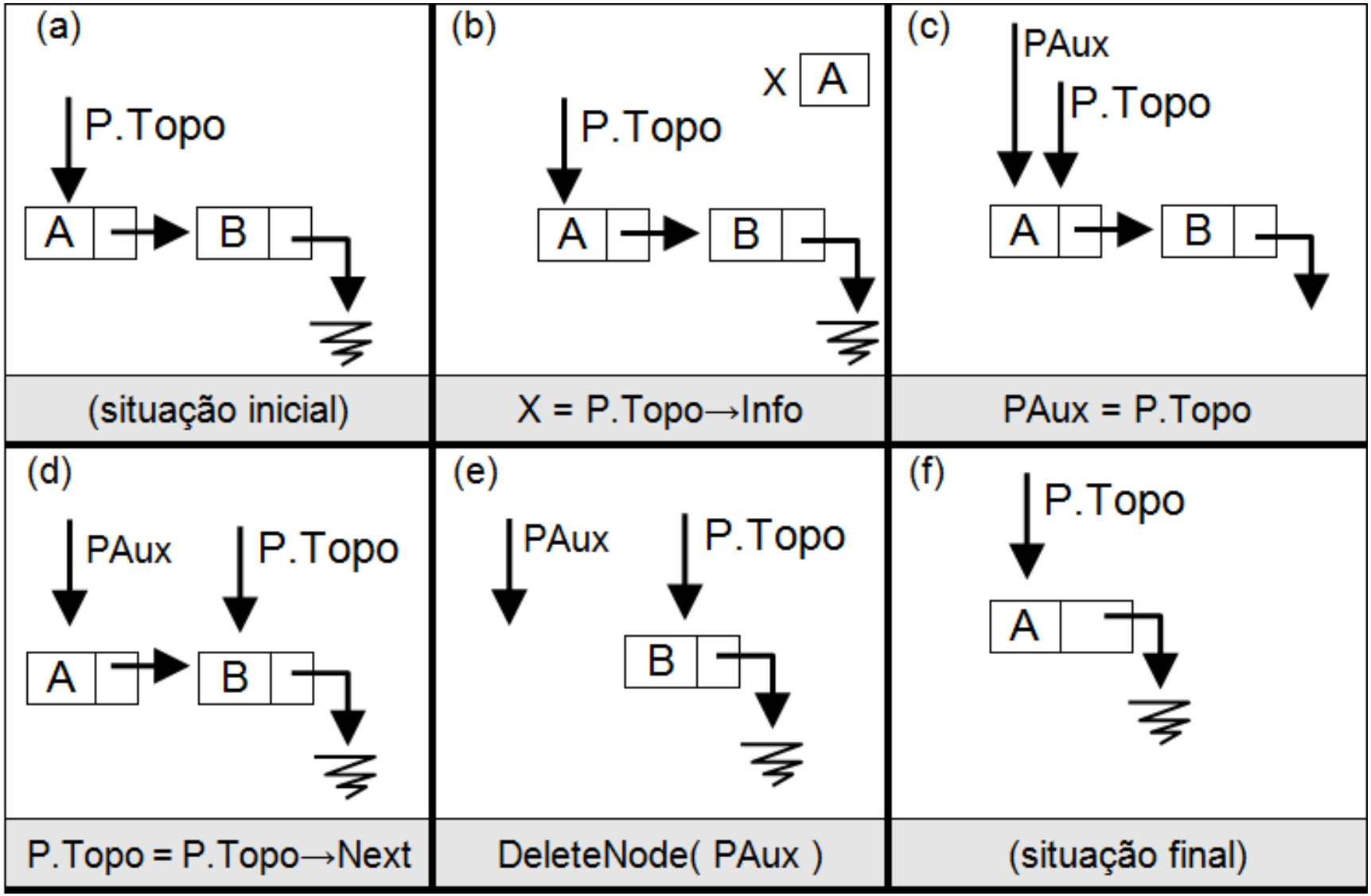
Exercícios

Exercício 4.2 Executar passo a Passo a Operação Empilha Partindo de Situação Inicial com Um Elemento na Pilha

Exercícios 4.3, 4.4, 4.5 e 4.6 Operações Desempilha, Cria, Vazia e Cheia

Exercício 4.7 Executar passo a Passo a Operação Desempilha Partindo de Situação Inicial com Um Único Elemento na Pilha

Exercício 4.2 Operação Desempilha



Exercícios 4.4 a 4.6

```
Cria (parâmetro por referência P do tipo Pilha) {
/* Cria a Pilha P, inicializando a Pilha como vazia - sem nenhum elemento. */
P.Topo = Null; // P.Topo passa a apontar para Null. Isso indica que a Pilha está vazia.
} // fim do Cria

Boolean Vazia (parâmetro por referência P do tipo Pilha) {
/* Retorna Verdadeiro se a Pilha P estiver sem nenhum elemento; Falso caso contrário */
Se (P.Topo == Null)
Então Retorne Verdadeiro;    // a Pilha P está vazia
Senão Retorne Falso;         // a Pilha P não está vazia
} // fim do Vazia

Boolean Cheia (parâmetro por referência P do tipo Pilha) {
/* Nesta implementação conceitual, a operação cheia retorna sempre Falso */
Retorne Falso;    // nesta implementação conceitual, a Pilha nunca estará cheia.
} // fim do Cheia
```

Exercício 4.3

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

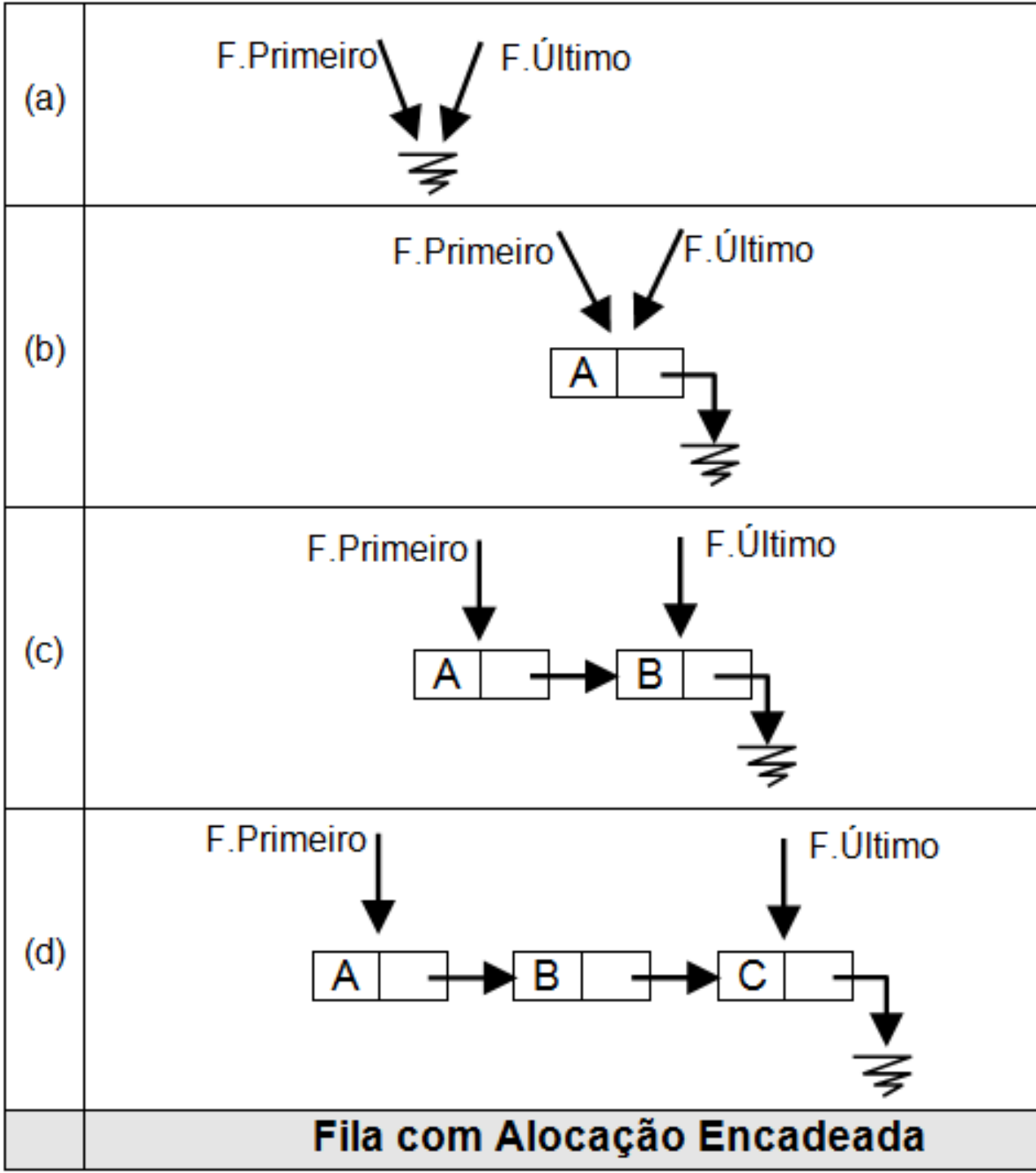
/* Se a Pilha P estiver vazia o parâmetro DeuCerto deve retornar Falso. Caso a Pilha P não estiver vazia, a operação Desempilha deve retornar o valor do elemento do Topo da Pilha no parâmetro X. O Nó em que se encontra o elemento do Topo deve ser desalocado, e o Topo da Pilha deve então ser atualizado para o próximo elemento */

Variável PAux do tipo NodePtr; // ponteiro para Nó, auxiliar

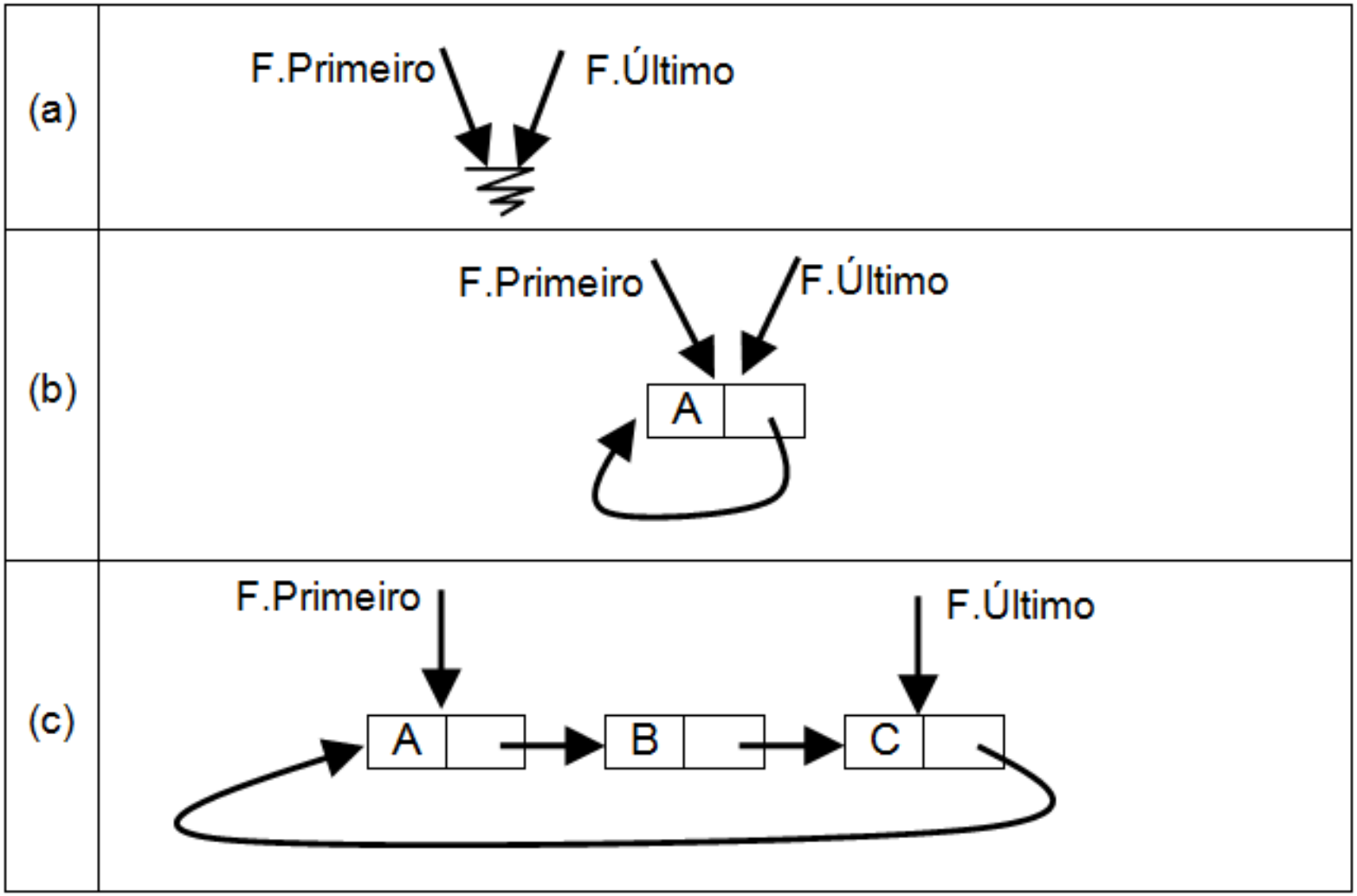
```
Se  (Vazia( P ) == Verdadeiro)
Então      DeuCerto = Falso;
Senão {    DeuCerto = Verdadeiro;
 X  = P.Topo→Info;
            PAux = P.Topo ;      // aponta PAux para P.Topo, para guardar este endereço
                                 // antes de mudar P.Topo de lugar
            P.Topo = P.Topo→Next;  // o topo da Pilha avança para o próximo elemento.
            DeleteNode( PAux ); // desaloca o Nó que armazenava o elemento do topo
          }
} // fim do Desempilha
```

Exercícios 4.8 a 4.12 - Fila Implementada como Lista Encadeada

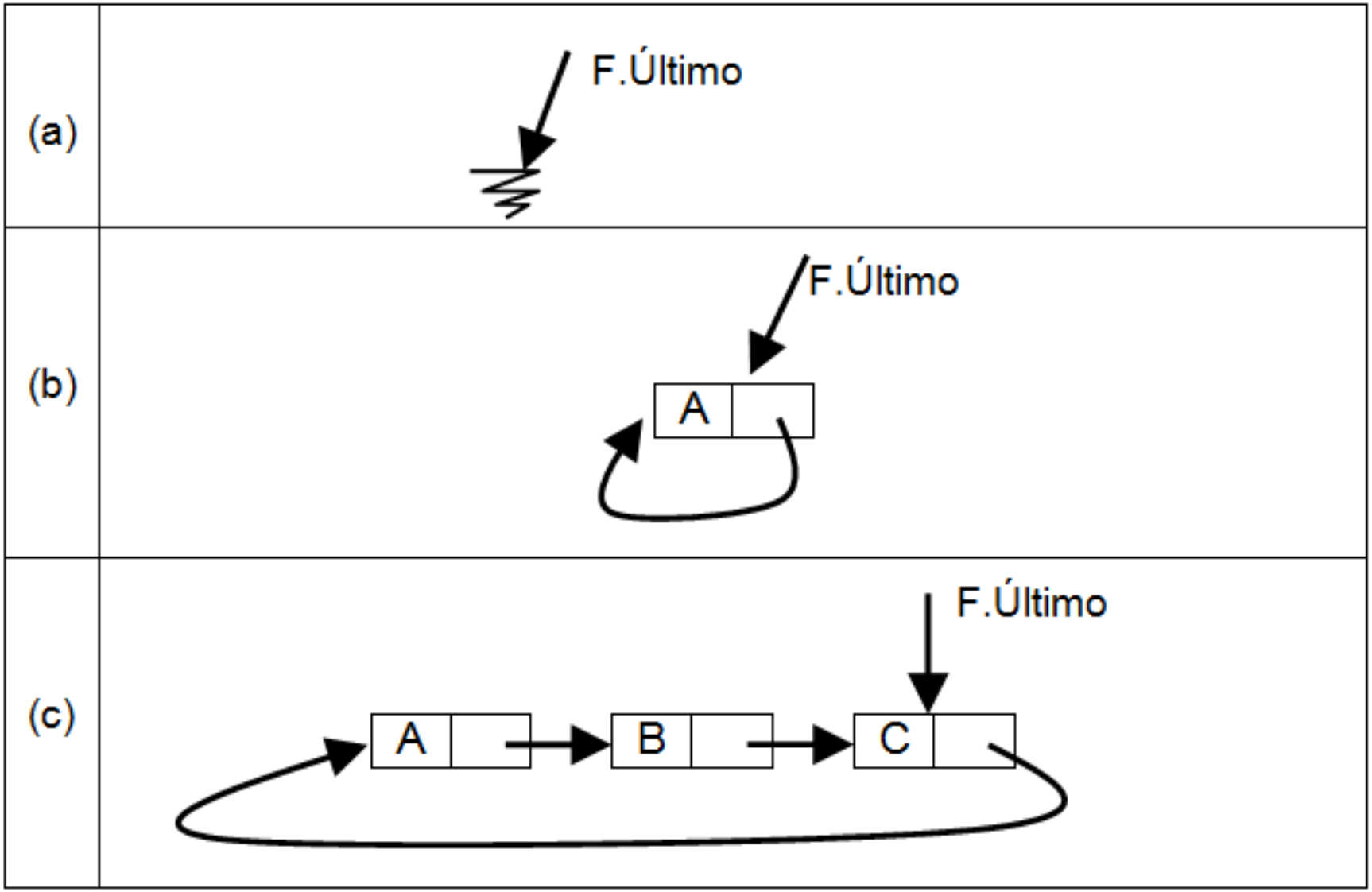
Solução Com Dois Ponteiros



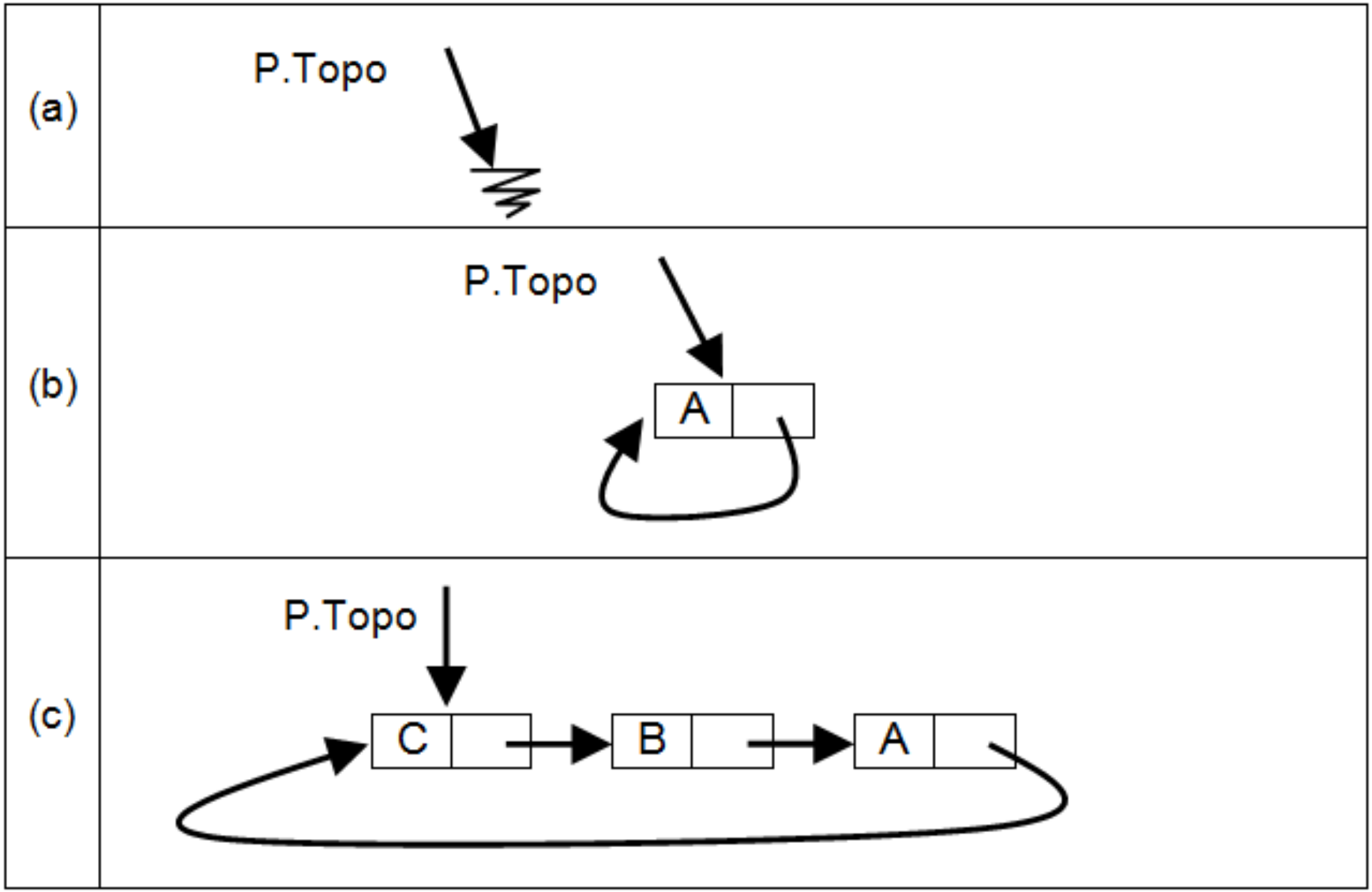
Exercício 4.13- Fila Implementada como Lista Encadeada Circular



Exercício 4.14- Fila como Lista Encadeada Circular com Só 1 Ponteiro



Exercício 4.15- Pilha como Lista Encadeada Circular



Exercício 4.18
Avanço de Projeto

Após fazer uma reflexão sobre as vantagens e desvantagens da Alocação Sequencial e da Alocação Encadeada, reflita sobre qual técnica de implementação parece ser mais adequada às características dos jogos que você está desenvolvendo no momento: Alocação Sequencial ou Alocação Encadeada de Memória?