

# Lista Encadeada: conceito, representação e algoritmos

Algoritmos e Estruturas de Dados I

# Problema

---

- Imaginem a situação da **automação de uma biblioteca**
  - Todos os livros devem ser cadastrados
  - O sistema deve informar se um determinado livro está ou não disponível nas estantes
  - Caso o livro não esteja disponível, o usuário poderá aguardar pela liberação do livro se cadastrando em uma fila de espera
  - Quando o livro for devolvido e liberado, o primeiro da fila deve ser contatado para vir buscá-lo

# Problema

---

## ■ Estatísticas

- 120.000 livros
- 1 fila de espera para cada livro
- No máximo 1000 pessoas ficam esperando por livros da biblioteca
- No máximo 30 pessoas ficam esperando um mesmo livro

# Problema

---

- Como representar/estruturar o problema?

# Soluções

---

- Alternativa 1

- Reservar espaço para 120.000 filas (uma para cada livro), com capacidade para 30 pessoas
- 120.000 vetores 30 elementos
- Espaço reservado para 3.600.000 pessoas
  
- Problema?

# Soluções

---

- Alternativa 1

- Reservar espaço para 120.000 filas (uma para cada livro), com capacidade para 30 pessoas
- 120.000 vetores 30 elementos
- Espaço reservado para 3.600.000 pessoas
  
- Problema?
  - Muito espaço reservado não é utilizado

# Soluções

---

- Alternativa 2
  - Alocar espaço para 1000 elementos
  - Todas as 120.000 filas compartilham o mesmo espaço
  - Problema?

# Soluções

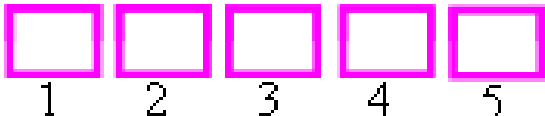
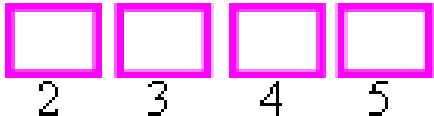
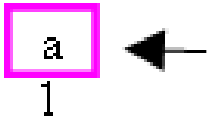
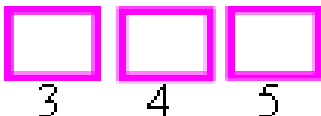
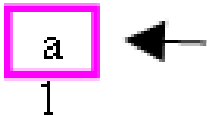
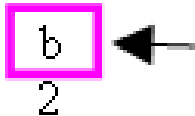
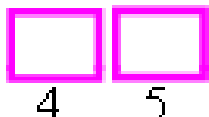
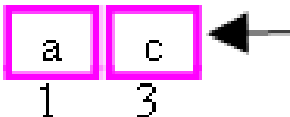
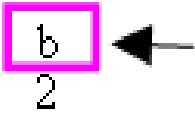
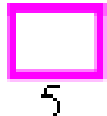
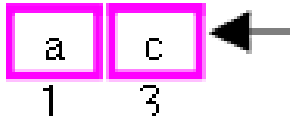
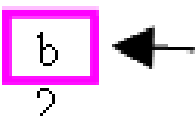

---

- Alternativa 2
  - Alocar espaço para 1000 elementos
  - Todas as 120.000 filas compartilham o mesmo espaço
- Problema?
  - Como 120.000 filas podem compartilhar a memória reservada a elas?


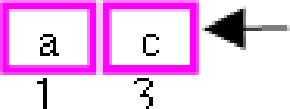

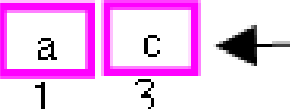
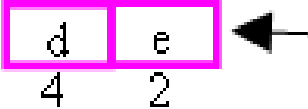
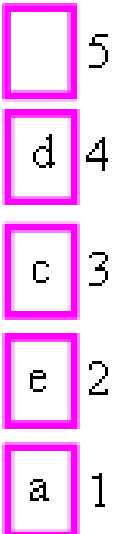
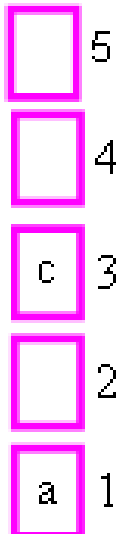
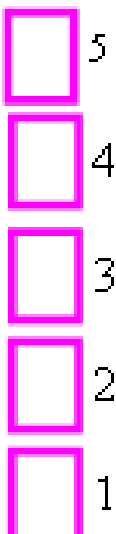
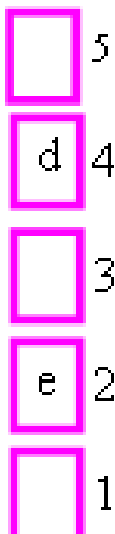
Como várias estruturas podem compartilhar um espaço de memória?



# Compartilhamento de memória

| banco de memória  | pilha x   | pilha y   | pilha z   | operação  |
|---|---|---|---|-----------|
|     |   |   |   |           |
|    |    |   |   | push(x,a) |
|    |    |    |   | push(y,b) |
|  |  |  |   | push(x,c) |
|  |  |  |  | push(z,d) |

# Compartilhamento de memória

|  |   |   |   |           |
|--|---|---|---|-----------|
|                           |                  |   |                  | pop(y,E)  |
|  |                  |   |                  | push(z,e) |
| <p>banco de memória</p>  | <p>pilha x</p>  | <p>pilha y</p>  | <p>pilha z</p>  |           |

# Compartilhamento de memória

---

## ■ Perguntas

- Como saber qual é o topo de uma pilha dessas  $(x, y, z)$ ?
- Como saber qual elemento vem logo abaixo do topo (o próximo na seqüência)?

# Compartilhamento de memória

---

## ■ Perguntas

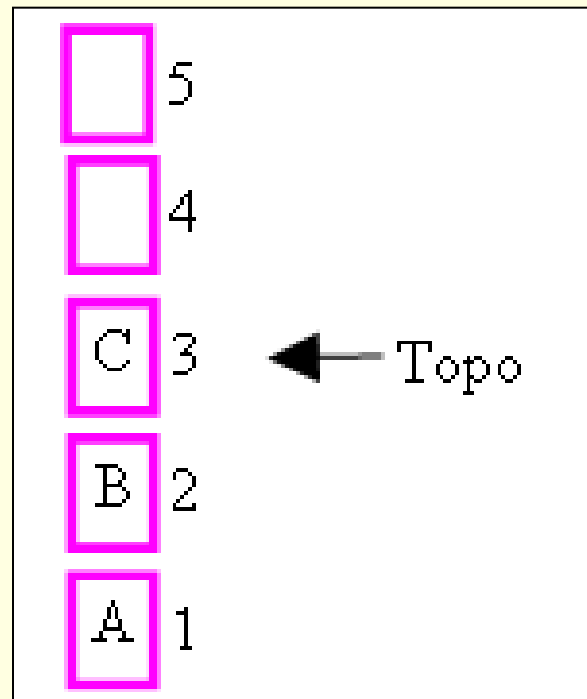
- Como saber qual é o topo de uma pilha dessas  $(x, y, z)$ ?
- Como saber qual elemento vem logo abaixo do topo (o próximo na seqüência)?

Alocação encadeada!

# Alocação sequencial vs. encadeada

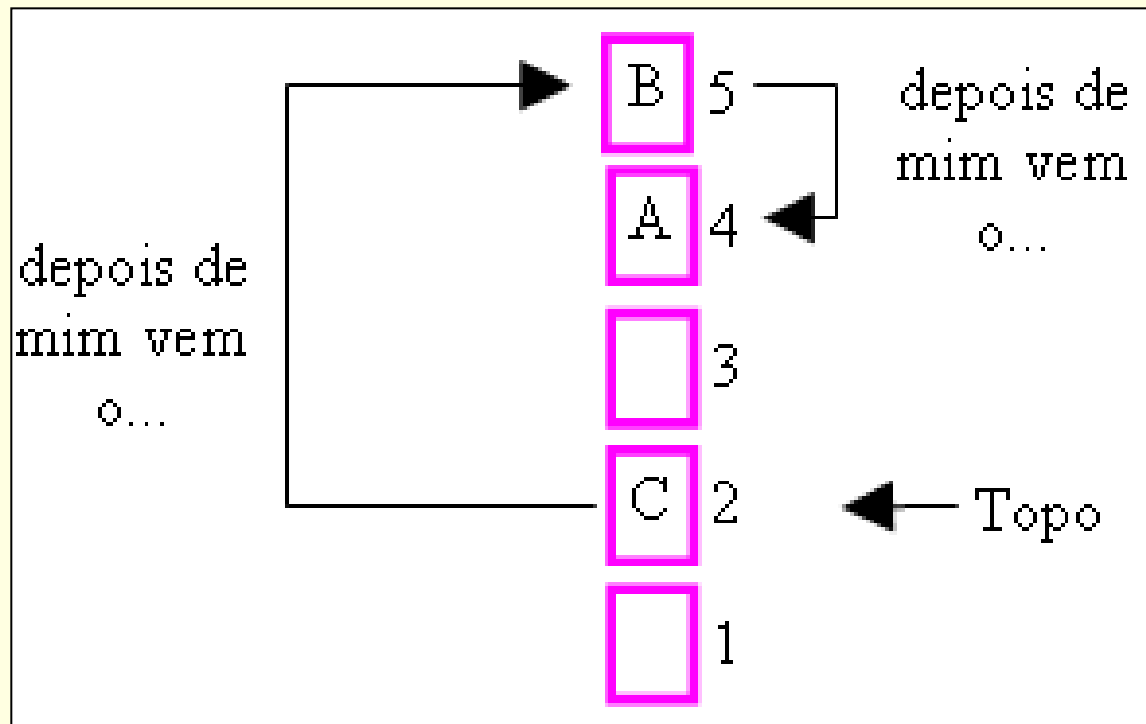
---

- **Alocação sequencial**: elementos são alocados em sequência; sequência “física”



# Alocação seqüencial vs. encadeada

- **Alocação encadeada**: elementos não estão necessariamente em posições adjacentes de memória; sequência “lógica” ou “virtual”

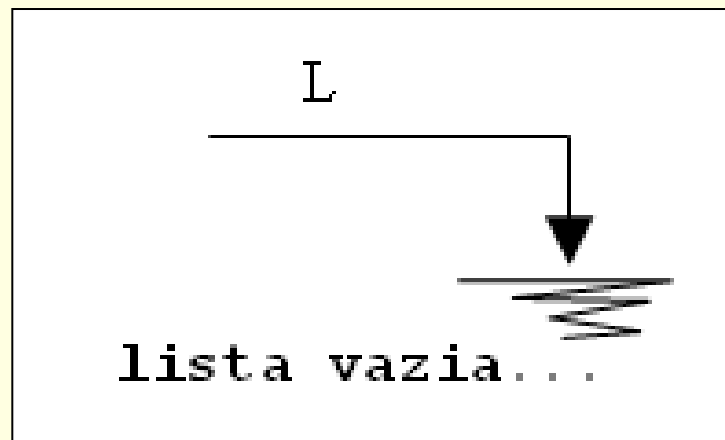
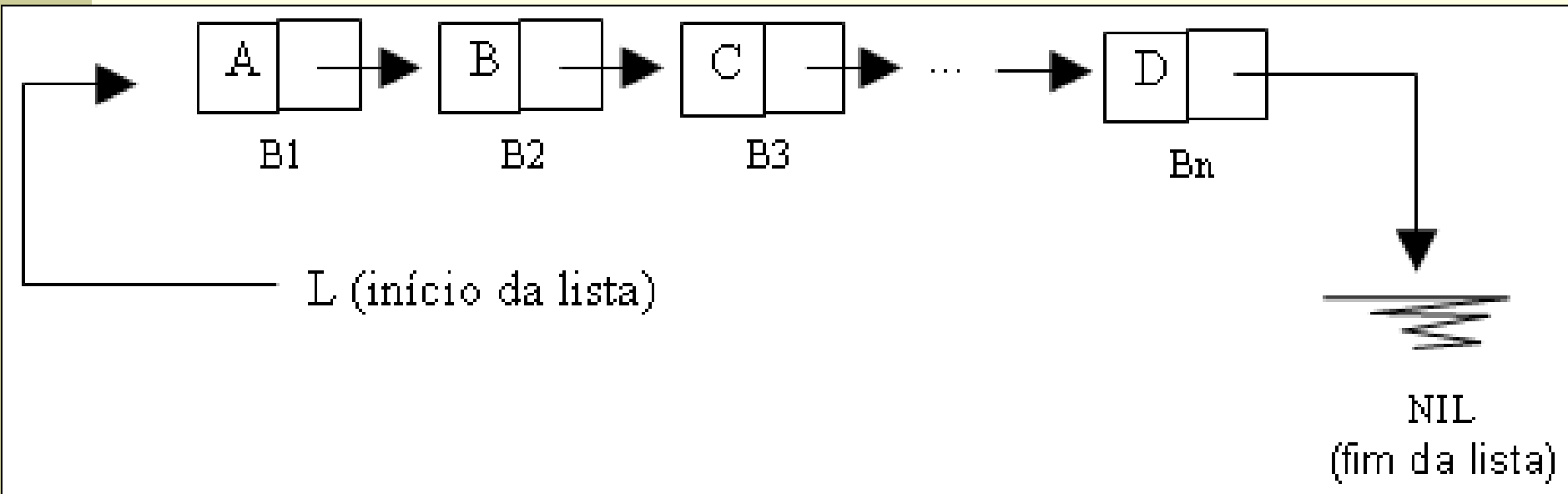


# Listas encadeadas

---

- Definição: uma **lista encadeada L**, com **n blocos de memória  $B_1, B_2, \dots, B_n$**  é definida pelas seguintes características:
  - Cada bloco de memória  $B_i$ , ou cada “nó” da lista, tem pelo menos dois campos:
    - Informação a ser armazenada
    - Indicação do próximo elemento da lista
  - Os blocos de memória não estão necessariamente em sequência física na memória
  - O acesso aos elementos da lista ocorre através de um indicador do início da lista (o primeiro elemento); o acesso aos demais elementos ocorre através da indicação de quem é o próximo na sequência
  - O último nó da lista indica um endereço inválido, chamado NIL ou NULL

# Representação





# Lista

---

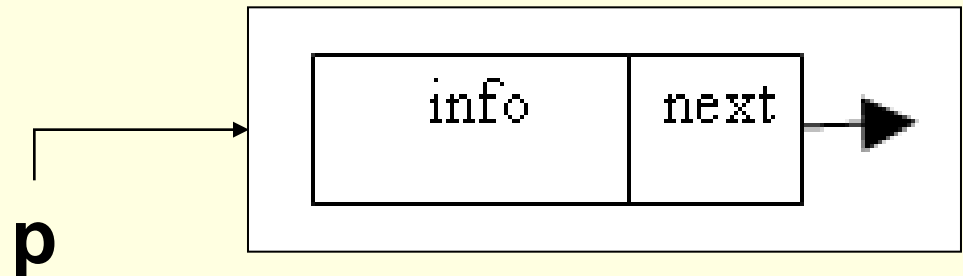
- Lista encadeada e dinâmica
  - Uma das representações mais interessantes e flexíveis que há
  - Aplicável para diversos problemas

# Possível declaração

```
struct no {  
    char info;  
    struct no *next;  
}
```

*/\* no programa: \*/*

```
struct no *p;  
p=(struct no*) malloc(sizeof(struct no));
```



# Implementação da lista encadeada

---

- Dependendo das exigências, indicadores do início e do fim da lista podem ser necessários
  - Acesso e manipulação dos elementos da lista

```

#include <stdio.h>

struct no {
    char info;
    struct no *next;
};

struct no *ini, *fim, *p;

int main(void) {
    ini=NULL;
    fim=NULL;

    p=(struct no*) malloc(sizeof(struct no));
    p->info='a';
    p->next=NULL;
    ini=p;
    fim=p;

```

```

p=(struct no*) malloc(sizeof(struct no));
p->info='b';
p->next=NULL;
fim->next=p;
fim=p;

```

```

p=ini;
while (p!=NULL) {
    printf("%c ",p->info);
    p=p->next;
}

```

```

p=ini;
while (p!=NULL) {
    ini=ini->next;
    free(p);
    p=ini;
}

```

```

system("pause");
return 0;

```

```

}

```

## Exemplo

Qual o resultado da execução desse programa?