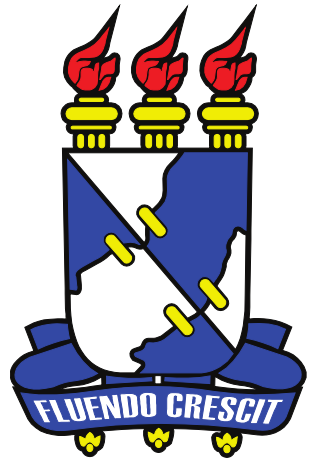


Estrutura de Dados I

Estrutura de fila e de pilha

Bruno Prado

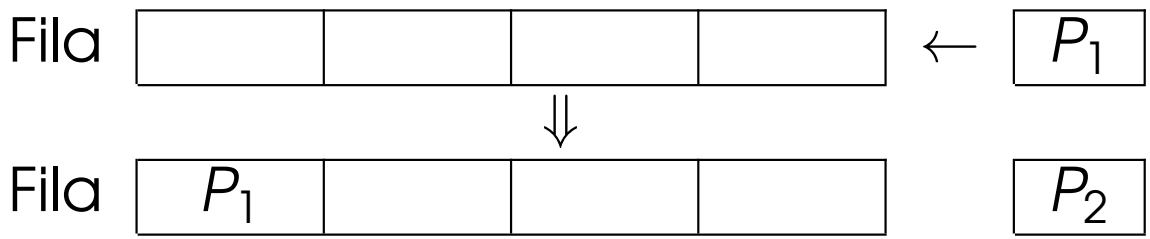
Departamento de Computação / UFS

Introdução

- O que é uma fila?
  - É uma estrutura de dados First-In First-Out (FIFO)
  - Duas operações principais: enfileirar e desenfileirar
  - A restrição imposta é que o primeiro elemento inserido é o primeiro a ser removido

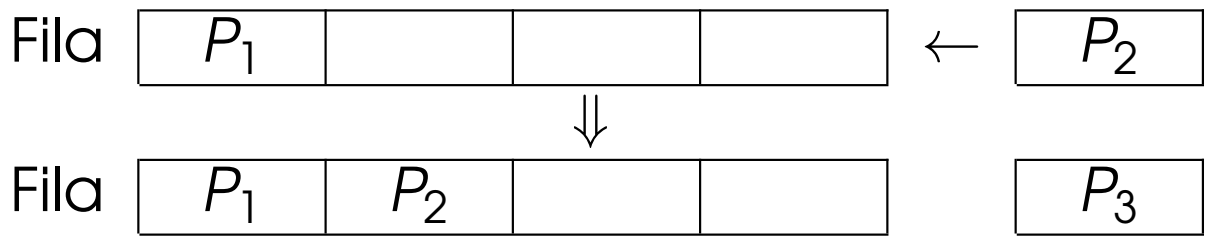
Introdução

- Pensando em pessoas
  - Enfileirar (push_back)



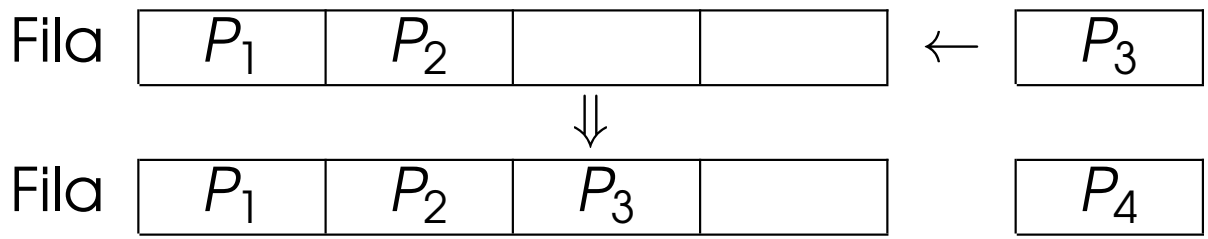
Introdução

- Pensando em pessoas
  - Enfileirar (push_back)



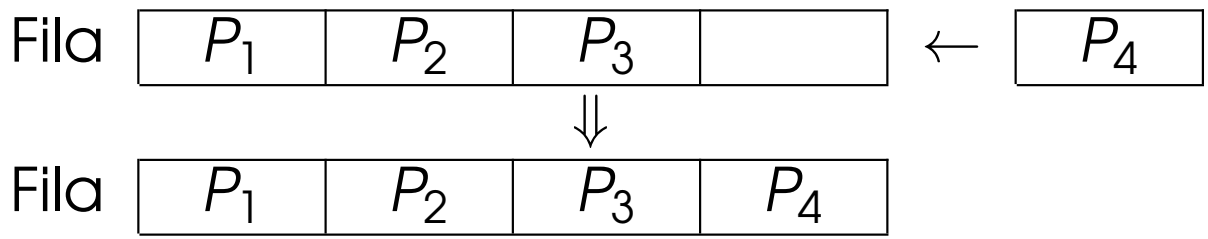
Introdução

- Pensando em pessoas
  - Enfileirar (push_back)



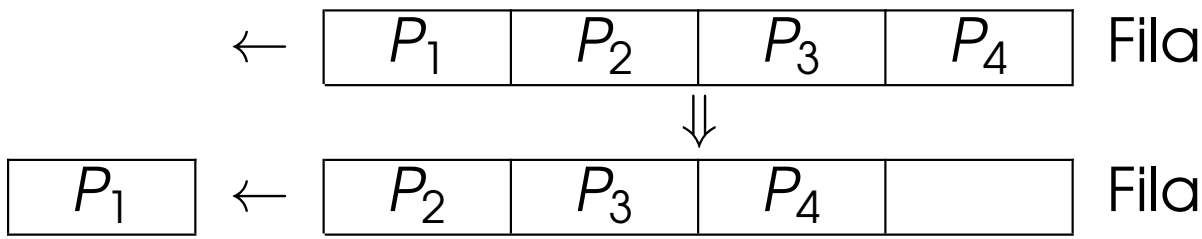
Introdução

- Pensando em pessoas
  - Enfileirar (push_back)



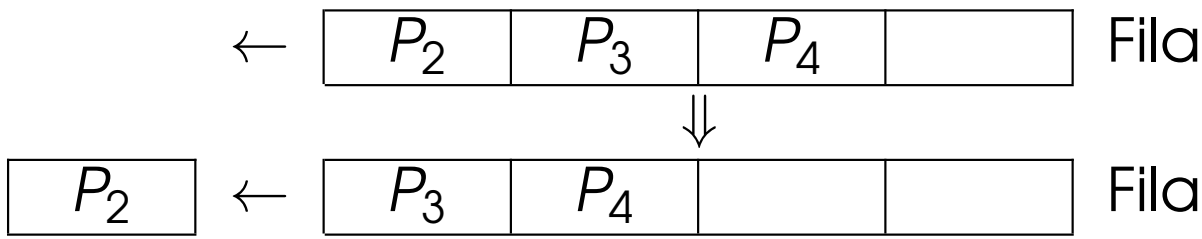
Introdução

- Pensando em pessoas
  - Desenfileirando (pop_front)



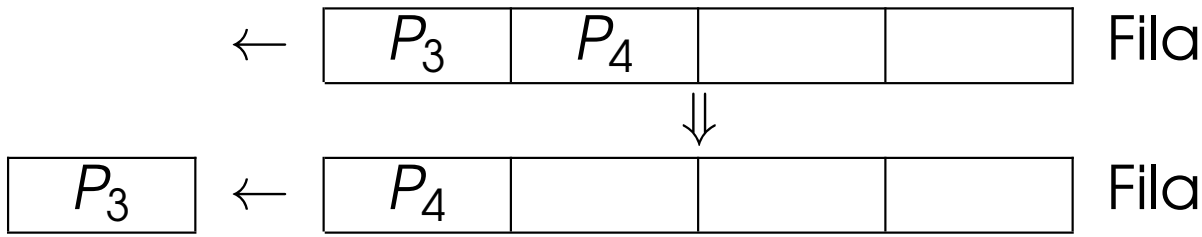
Introdução

- Pensando em pessoas
  - Desenfileirando (pop_front)



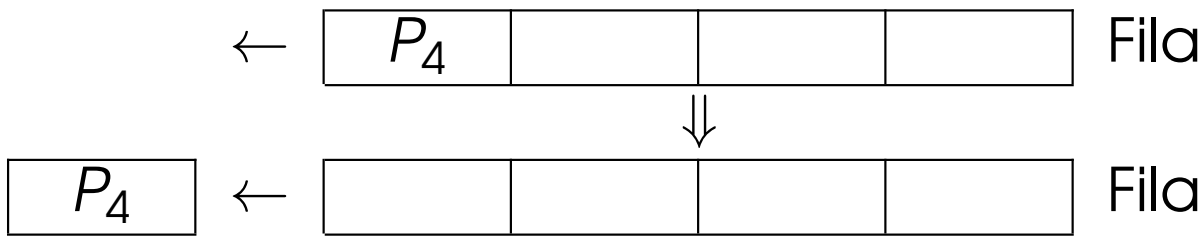
Introdução

- Pensando em pessoas
  - Desenfileirando (pop_front)



Introdução

- Pensando em pessoas
  - Desenfileirando (pop_front)

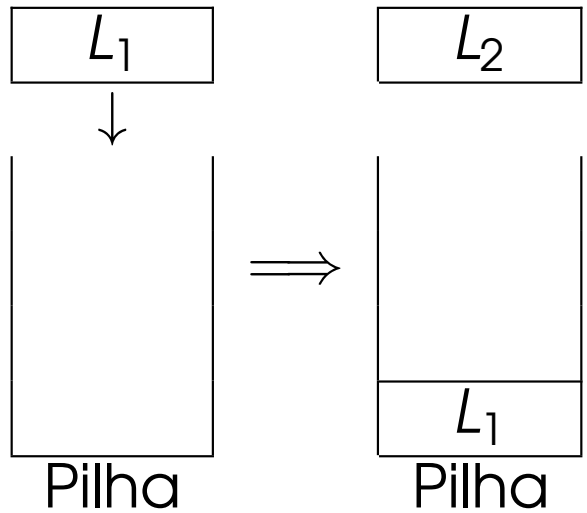


Introdução

- O que é uma pilha?
  - É uma estrutura de dados Last-In First-Out (LIFO)
  - Duas operações principais: empilhar e desempilhar
  - A restrição imposta é que o último elemento inserido é o primeiro a ser removido

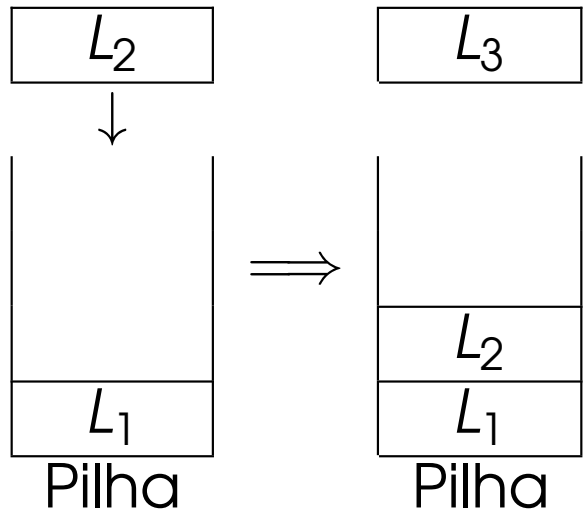
Introdução

- Pensando em livros
  - Empilhando (push)



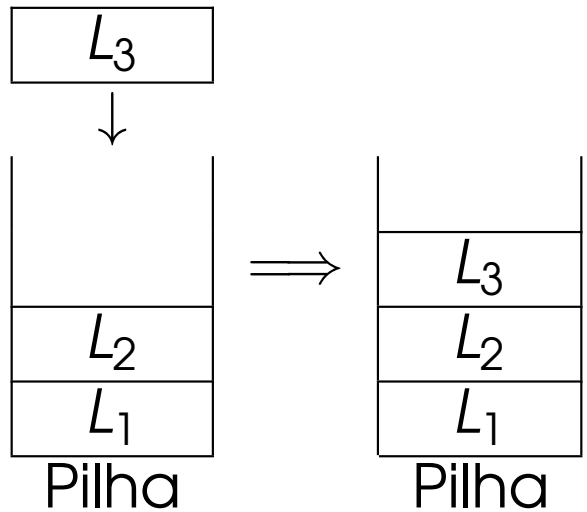
Introdução

- Pensando em livros
  - Empilhando (push)



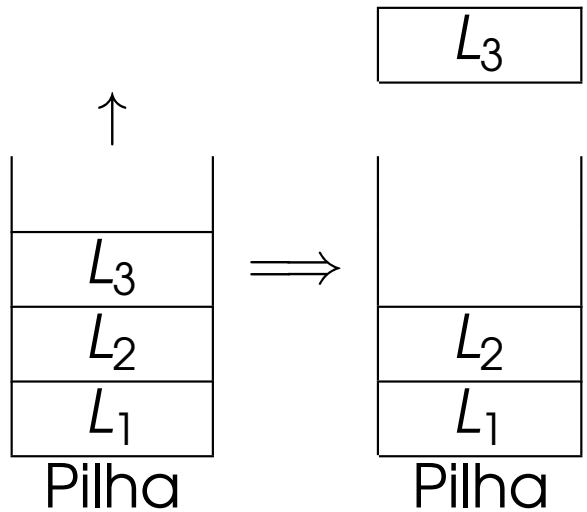
Introdução

- Pensando em livros
  - Empilhando (push)



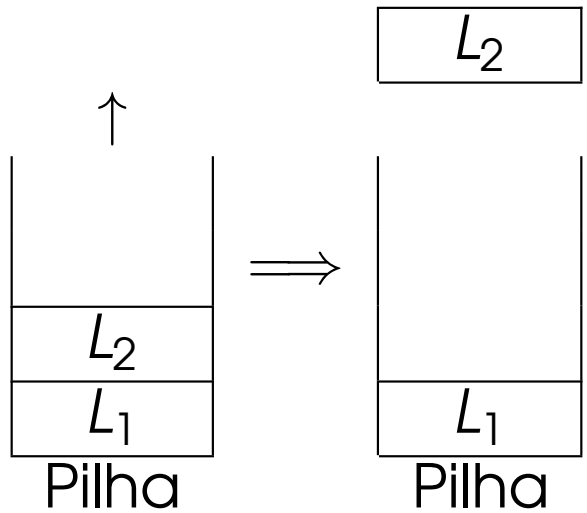
Introdução

- Pensando em livros
  - Desempilhando (pop)



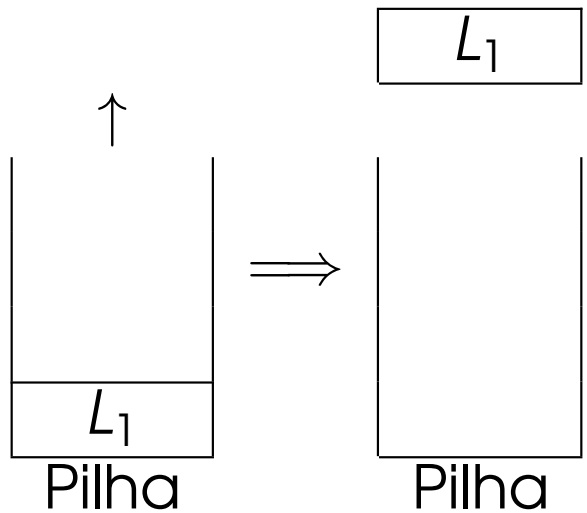
Introdução

- Pensando em livros
  - Desempilhando (pop)



Introdução

- Pensando em livros
  - Desempilhando (pop)



Fila

- Técnicas de implementação
  - Vetor
  - Estrutura de lista

Fila

- Técnicas de implementação
  - Vetor
  - Estrutura de lista
- Funções auxiliares
  - Verificar se está vazia (empty)
  - Verificar a quantidade de elementos na fila (size)
  - Acessar o elemento inicial da fila (front)
  - Acessar o elemento final da fila (back)

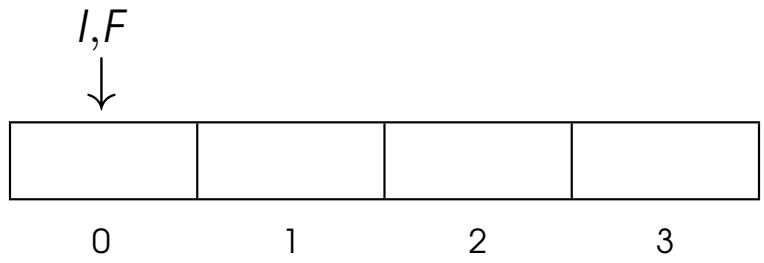
Fila

- Implementação em C
  - Vetor
  - Definição da estrutura

```
// Estrutura de fila
typedef struct fila {
        // Capacidade do vetor
        size_t capacidade;
        // Inicio do vetor
        size_t inicio;
        // Tamanho utilizado
        size_t tamanho;
        // Vetor
        int* E;
} fila;
```

Fila

- Implementação com vetor
  - Inicialização da estrutura
  - É feita a alocação do vetor com capacidade 4 e com índice de início e de tamanho com valor 0, indicando que não possui nenhum elemento

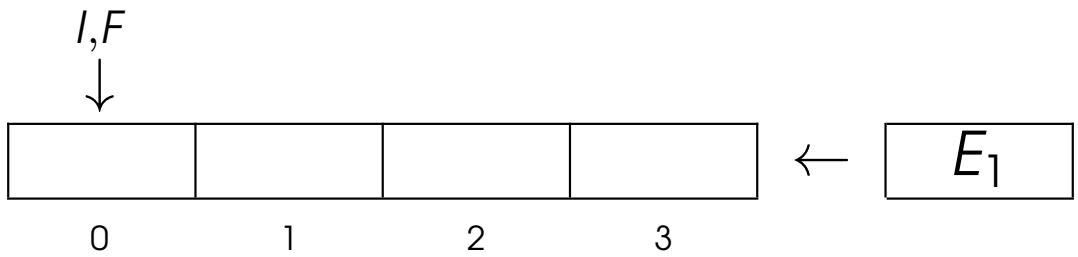


Fila

$$Capacidade = 4$$
$$Inicio = 0$$
$$Tamanho = 0$$

Fila

- Implementação com vetor
  - Enfileirando os elementos
  - É feito o incremento do tamanho da fila e a posição (*Inicio* + *Tamanho*) mod *Capacidade* recebe o valor do elemento E_1

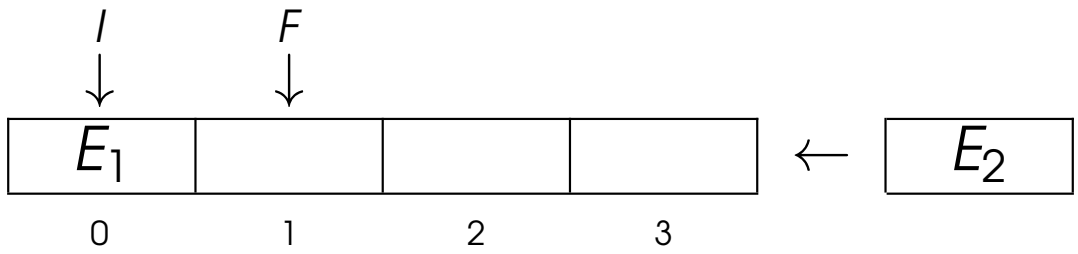


Fila

$$Capacidade = 4$$
$$Inicio = 0$$
$$Tamanho = 0$$

Fila

- Implementação com vetor
  - Enfileirando os elementos
  - É feito o incremento do tamanho da fila e a posição (*Inicio* + *Tamanho*) mod *Capacidade* recebe o valor do elemento E_2

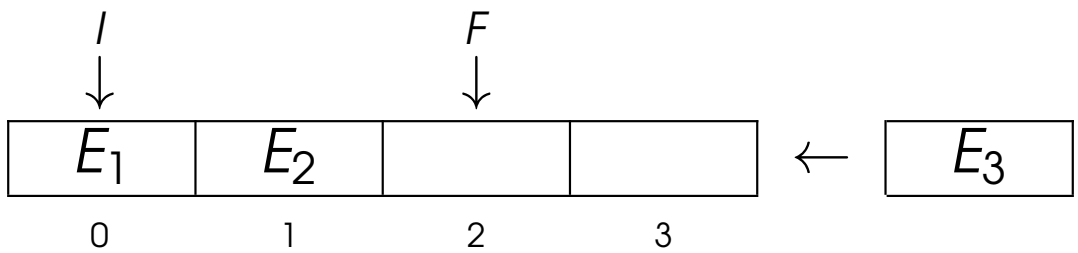


Fila

$$Capacidade = 4$$
$$Inicio = 0$$
$$Tamanho = 1$$

Fila

- Implementação com vetor
  - Enfileirando os elementos
  - É feito o incremento do tamanho da fila e a posição (*Inicio* + *Tamanho*) mod *Capacidade* recebe o valor do elemento E_3

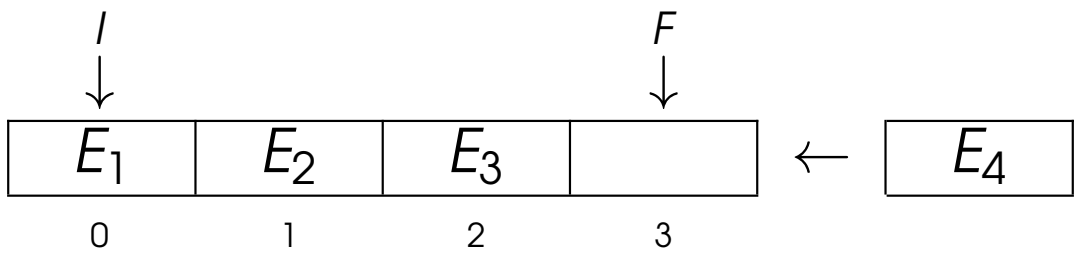


Fila

$$Capacidade = 4$$
$$Inicio = 0$$
$$Tamanho = 2$$

Fila

- Implementação com vetor
  - Enfileirando os elementos
  - É feito o incremento do tamanho da fila e a posição (*Inicio* + *Tamanho*) mod *Capacidade* recebe o valor do elemento E_4

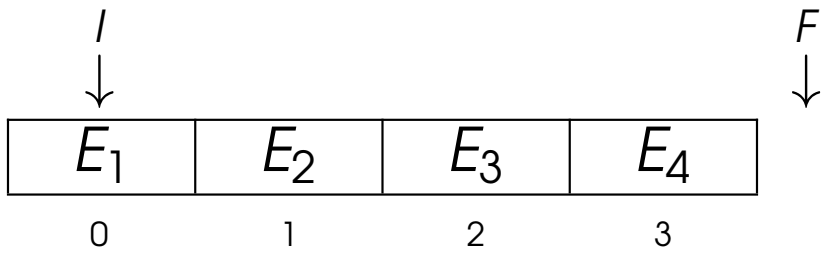


Fila

$$Capacidade = 4$$
$$Inicio = 0$$
$$Tamanho = 3$$

Fila

- Implementação com vetor
  - Todas as posições do vetor estão ocupadas, condição que pode ser verificada através da comparação do tamanho e da capacidade

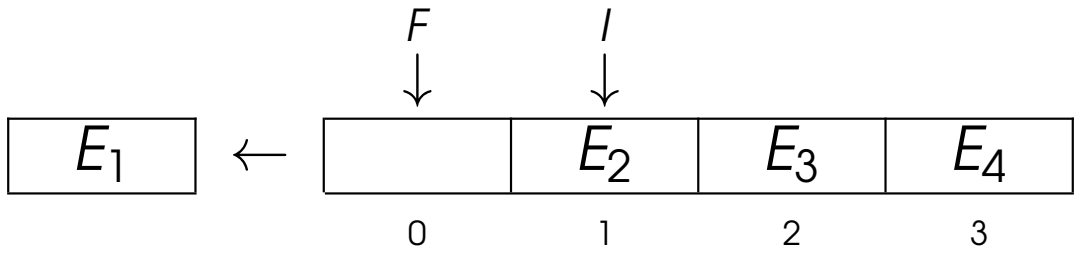


Fila

$$Capacidade = 4$$
$$Inicio = 0$$
$$Tamanho = 4$$

Fila

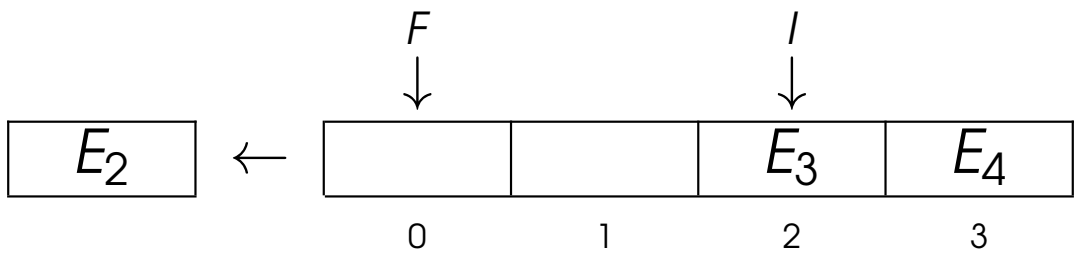
- Implementação com vetor
  - Desenfileirando os elementos
  - O elemento E_1 do início da fila é removido, com o incremento do *Inicio* $+ 1$ mod *Capacidade* e decremento do *Tamanho* $- 1$



$$Capacidade = 4$$
$$Inicio = 1$$
$$Tamanho = 3$$

Fila

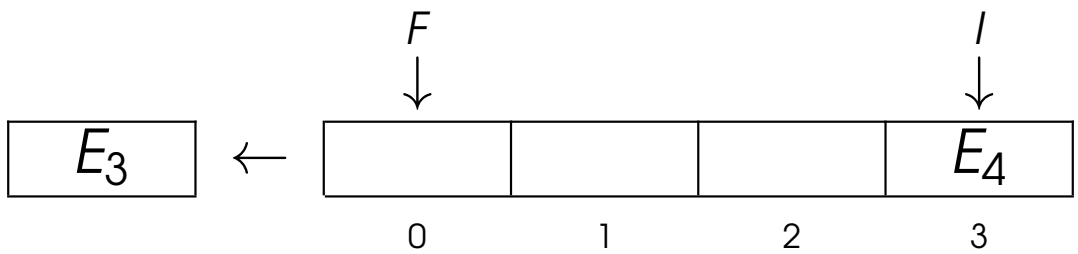
- Implementação com vetor
  - Desenfileirando os elementos
  - O elemento E_2 do início da fila é removido, com o incremento do $Inicio + 1$ mod $Capacidade$ e decremento do $Tamanho - 1$



$$Capacidade = 4$$
$$Inicio = 2$$
$$Tamanho = 2$$

Fila

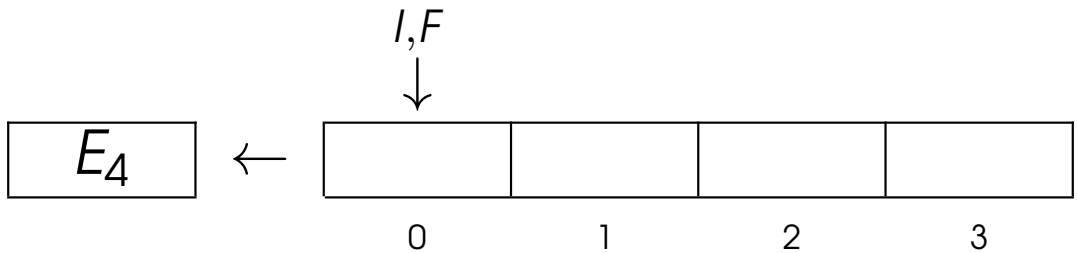
- Implementação com vetor
  - Desenfileirando os elementos
  - O elemento E_3 do início da fila é removido, com o incremento do $Inicio + 1$ mod $Capacidade$ e decremento do $Tamanho - 1$



$$Capacidade = 4$$
$$Inicio = 3$$
$$Tamanho = 1$$

Fila

- Implementação com vetor
  - Desenfileirando os elementos
  - Todos os elementos foram removidos, a fila está vazia com índices de início e de tamanho com valor 0



$$Capacidade = 4$$
$$Inicio = 0$$
$$Tamanho = 0$$

Fila

- Implementação em C
  - Estrutura de lista
  - Definição das estruturas

```
// Estrutura de elemento
typedef struct elemento {
      // Valor do elemento
      int E;
      // Ponteiro
      elemento* P;
} elemento;
```

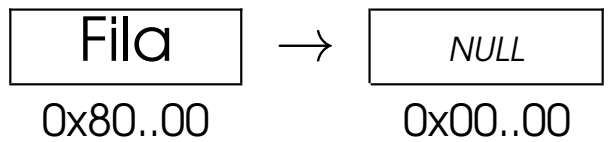
Fila

- Implementação em C
  - Estrutura de lista
  - Definição das estruturas

```
// Estrutura de fila
typedef struct fila {
        // Elemento inicial
        elemento* inicial;
        // Elemento final
        elemento* final;
} fila;
```

Fila

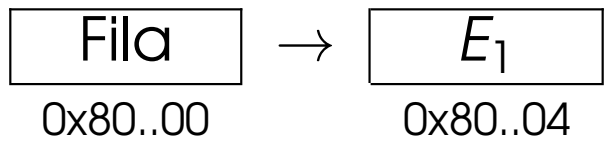
- Implementação com lista encadeada
  - Inicialização das estruturas
  - A fila está vazia com referências nulas para elementos inicial e final



$$Inicial = 0x00..00$$
$$Final = 0x00..00$$

Fila

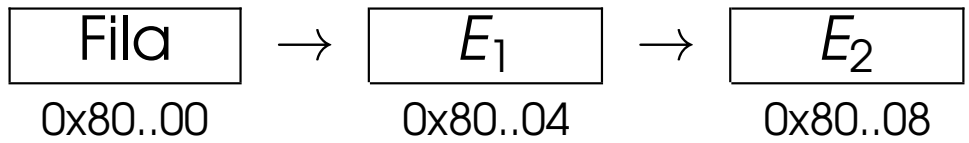
- Implementação com lista encadeada
  - Enfileirando os elementos
  - É feita a alocação do elemento E_1 e a sua inserção na fila, ajustando os ponteiros inicial e final



$$Inicial = 0x80..04$$
$$Final = 0x80..04$$

Fila

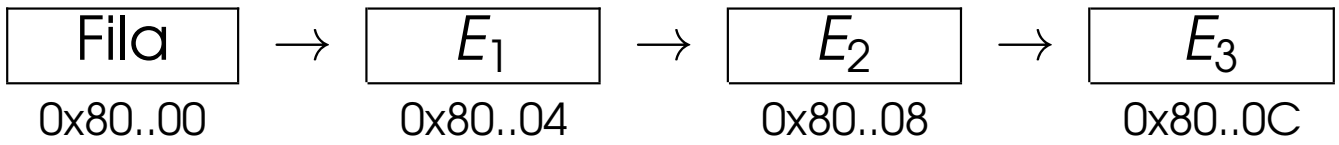
- Implementação com lista encadeada
  - Enfileirando os elementos
  - É feita a alocação do elemento E_2 e utilizando a referência do final da fila é realizada a sua inserção



$$Inicial = 0x80..04$$
$$Final = 0x80..08$$

Fila

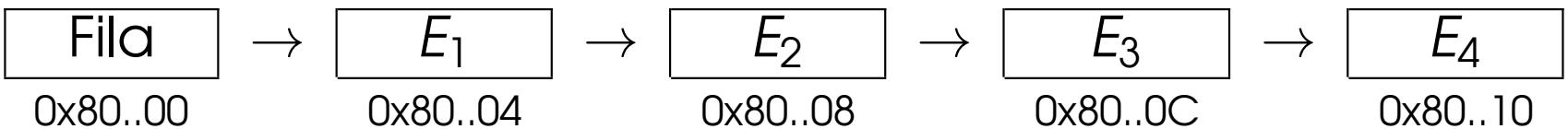
- Implementação com lista encadeada
  - Enfileirando os elementos
  - É feita a alocação do elemento E_3 e utilizando a referência do final da fila é realizada a sua inserção



$$Inicial = 0x80..04$$
$$Final = 0x80..0C$$

Fila

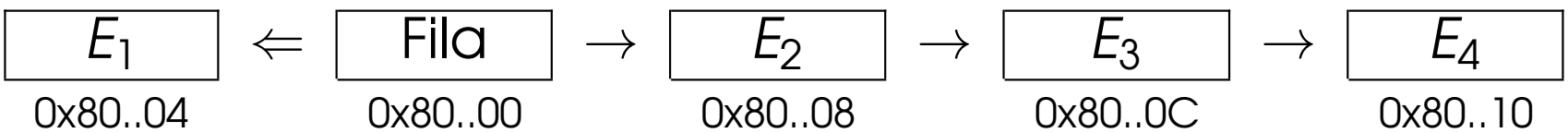
- Implementação com lista encadeada
  - Enfileirando os elementos
  - É feita a alocação do elemento E_4 e utilizando a referência do final da fila é realizada a sua inserção



$$Inicial = 0x80..04$$
$$Final = 0x80..10$$

Fila

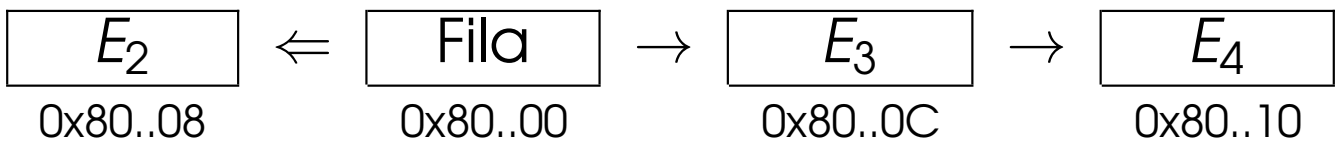
- Implementação com lista encadeada
  - Desenfileirando os elementos
  - O elemento E_1 referenciado pelo ponteiro inicial é removido da fila e o próximo elemento da sequência E_2 passa a ser o inicial



$$Inicial = 0x80..08$$
$$Final = 0x80..10$$

Fila

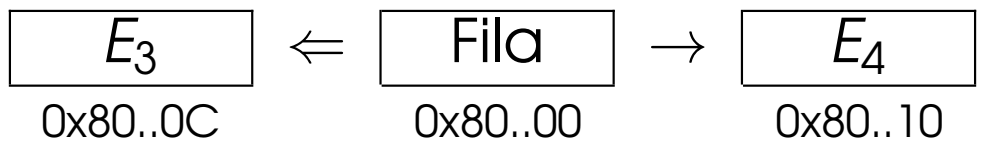
- Implementação com lista encadeada
  - Desenfileirando os elementos
  - O elemento E_2 referenciado pelo ponteiro inicial é removido da fila e o próximo elemento da sequência E_3 passa a ser o inicial



$$Inicial = 0x80..0C$$
$$Final = 0x80..10$$

Fila

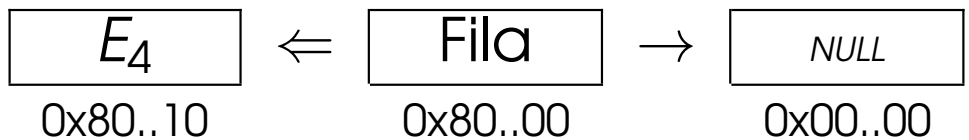
- Implementação com lista encadeada
  - Desenfileirando os elementos
  - O elemento E_3 referenciado pelo ponteiro inicial é removido da fila e o próximo elemento da sequência E_4 passa a ser o inicial



$$Inicial = 0x80..10$$
$$Final = 0x80..10$$

Fila

- Implementação com lista encadeada
  - Desenfileirando os elementos
  - O elemento E_4 referenciado pelo ponteiro inicial é removido da fila e por não ter mais nenhum elemento armazenado, ambas as referências são anuladas



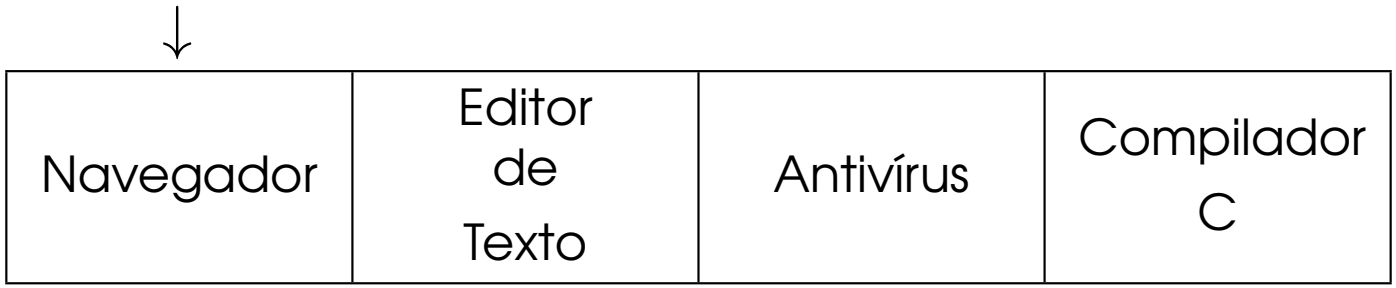
$$Inicial = 0x00..00$$
$$Final = 0x00..00$$

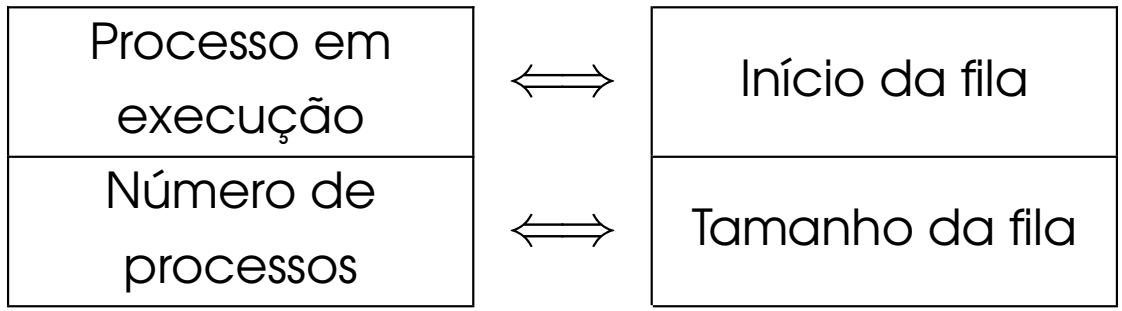
Fila

- Análise de complexidade
  - Espaço $\Theta(n)$
  - Tempo
    - Enfileirar $O(1)$
    - Desenfileirar $O(1)$

Aplicações

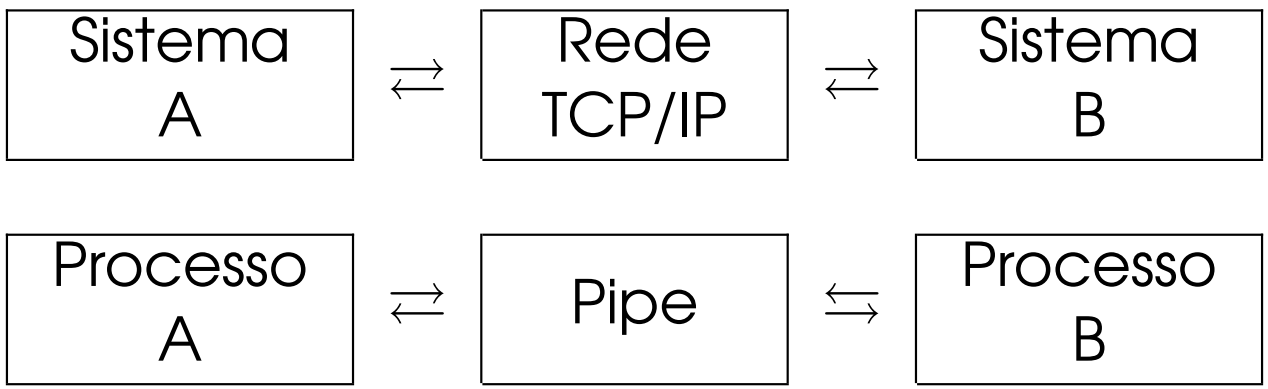
- Escalonamento de processos





Aplicações

- Troca de informações entre processos ou sistemas
  - Rede TCP/IP
  - Interface pipe



Pilha

- Técnicas de implementação
  - Vetor
  - Estrutura de lista

Pilha

- Técnicas de implementação
  - Vetor
  - Estrutura de lista
- Funções auxiliares
  - Verificar se está vazia (empty)
  - Verificar a quantidade de elementos na pilha (size)
  - Acessar o elemento do topo da pilha (top)

Pilha

- Implementação em C
  - Vetor
  - Definição da estrutura

```
// Estrutura de pilha
typedef struct pilha {
        // Capacidade do vetor
        size_t capacidade;
        // Topo da pilha
        size_t topo;
        // Vetor
        int* E;
} pilha;
```

Pilha

- Implementação com vetor
  - Inicialização da estrutura
  - É feita a alocação do vetor com capacidade 4 e com índice de topo negativo, indicando que não possui nenhum elemento

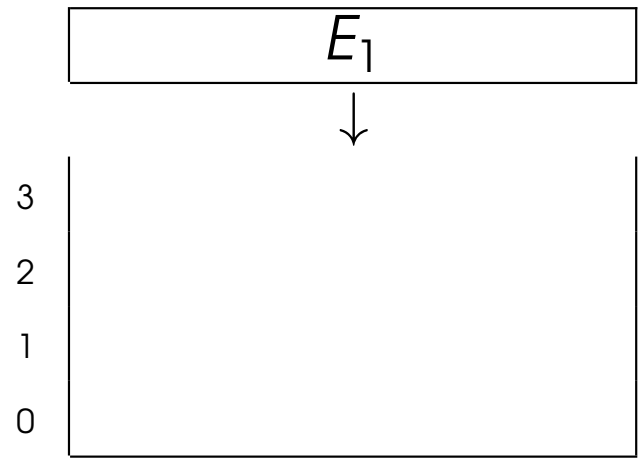
Pilha

$Capacidade = 4$

$Topo = -1$

Pilha

- Implementação com vetor
  - Empilhando os elementos
  - É feito o incremento do topo da pilha e a posição recebe o valor do elemento E_1



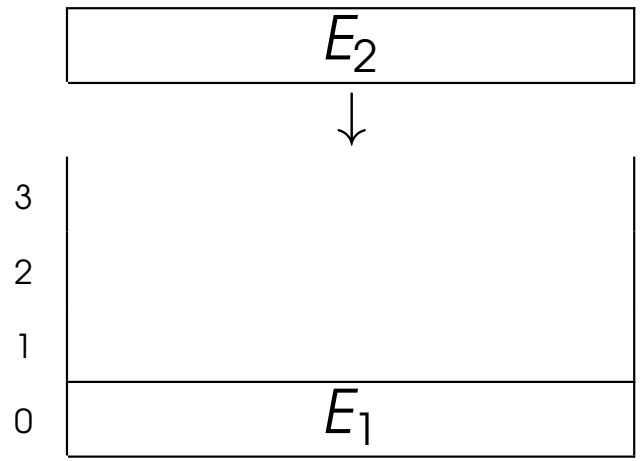
Pilha

$Capacidade = 4$

$Topo = -1$

Pilha

- Implementação com vetor
  - Empilhando os elementos
  - É feito o incremento do topo da pilha e a posição recebe o valor do elemento E_2

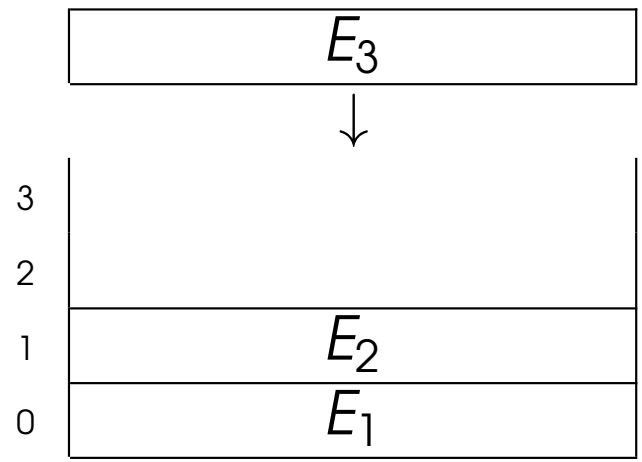


$Capacidade = 4$

$Topo = 0$

Pilha

- Implementação com vetor
  - Empilhando os elementos
  - É feito o incremento do topo da pilha e a posição recebe o valor do elemento E_3



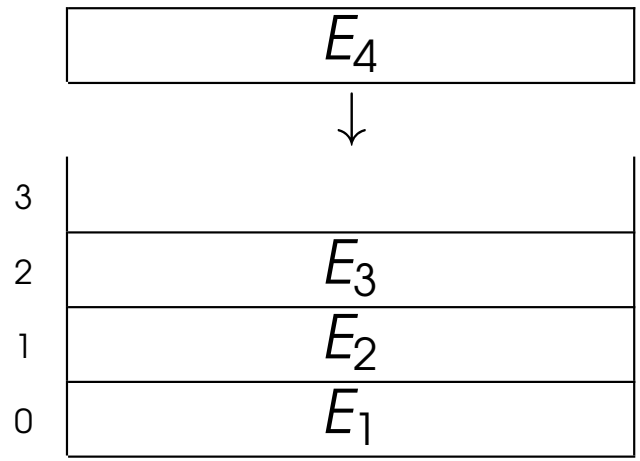
Pilha

$Capacidade = 4$

$Topo = 1$

Pilha

- Implementação com vetor
  - Empilhando os elementos
  - É feito o incremento do topo da pilha e a posição recebe o valor do elemento E_4



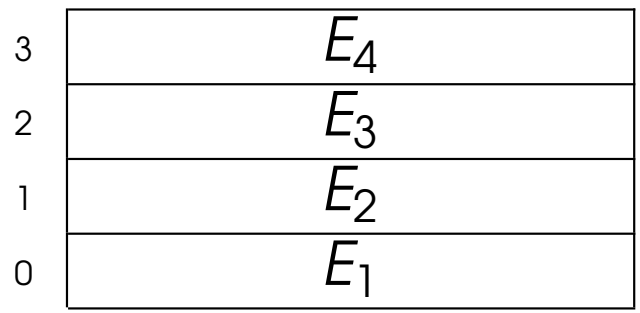
Pilha

$Capacidade = 4$

$Topo = 2$

Pilha

- Implementação com vetor
  - Todas as posições do vetor estão ocupadas, caso seja feita uma operação de empilhamento é preciso realizar a realocação do vetor e atualizar sua capacidade



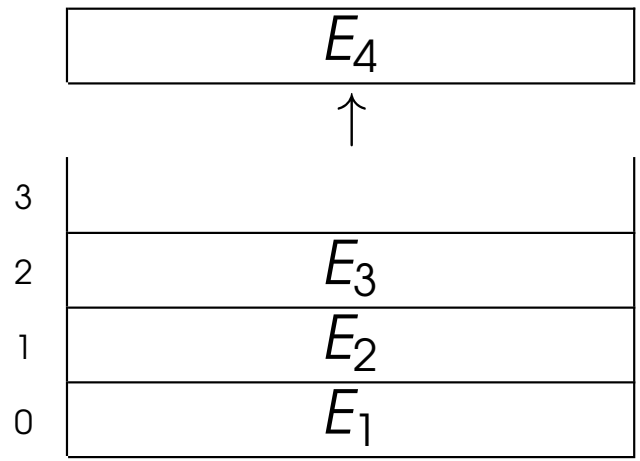
Pilha

$Capacidade = 4$

$Topo = 3$

Pilha

- Implementação com vetor
  - Desempilhando os elementos
  - O elemento E_4 é removido da pilha e o valor do topo é decrementado



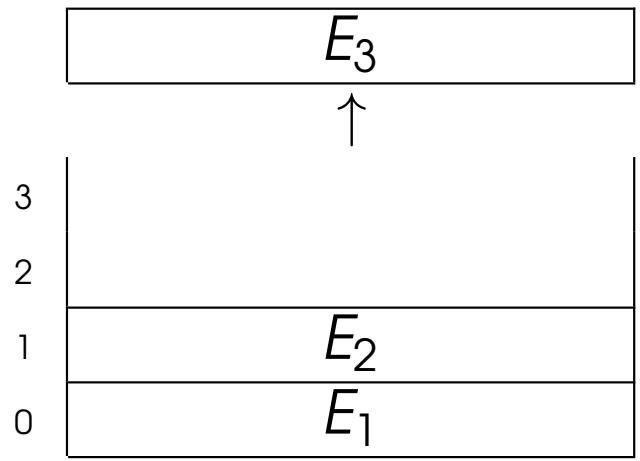
Pilha

$Capacidade = 4$

$Topo = 2$

Pilha

- Implementação com vetor
  - Desempilhando os elementos
  - O elemento E_3 é removido da pilha e o valor do topo é decrementado



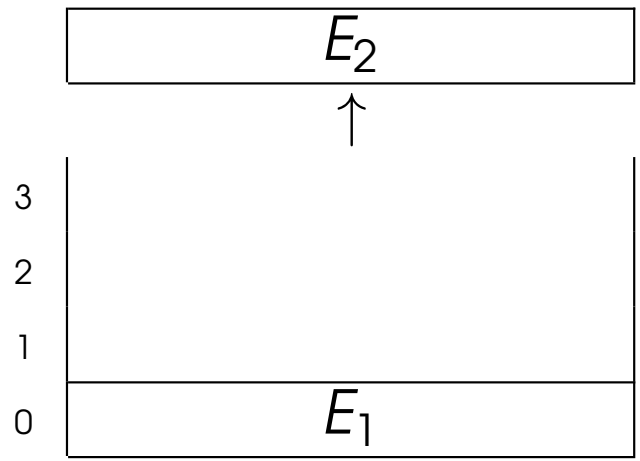
Pilha

$Capacidade = 4$

$Topo = 1$

Pilha

- Implementação com vetor
  - Desempilhando os elementos
  - O elemento E_2 é removido da pilha e o valor do topo é decrementado

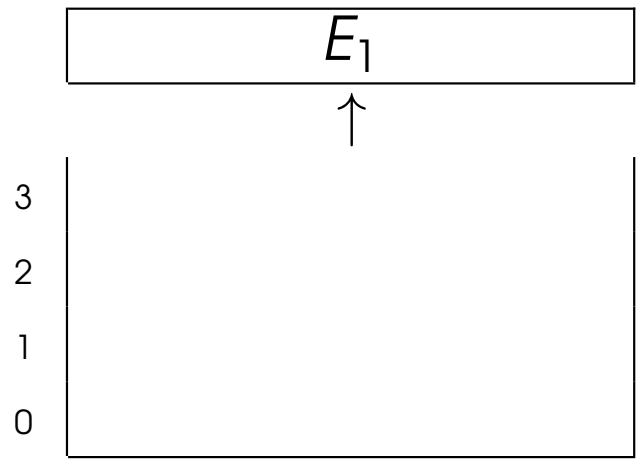


$Capacidade = 4$

$Topo = 0$

Pilha

- Implementação com vetor
  - Desempilhando os elementos
  - O elemento E_2 é removido da pilha e o valor do topo é decrementado



Pilha

$Capacidade = 4$

$Topo = -1$

Pilha

- Implementação com vetor
  - O índice de topo negativo indica que a pilha não possui nenhum elemento armazenado

Pilha

$Capacidade = 4$

$Topo = -1$

Pilha

- Implementação em C
  - Estrutura de lista
  - Definição das estruturas

```
// Estrutura de elemento
typedef struct elemento {
        // Valor
        int E;
        // Ponteiro
        elemento* P;
} elemento;
```

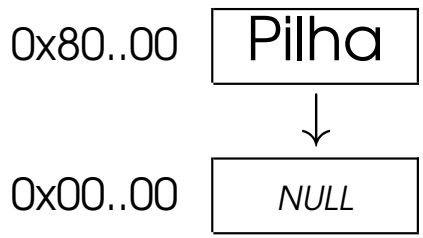
Pilha

- Implementação em C
  - Estrutura de lista
  - Definição das estruturas

```
// Estrutura de pilha
typedef struct pilha {
        // Topo da pilha
        elemento* topo;
} pilha;
```

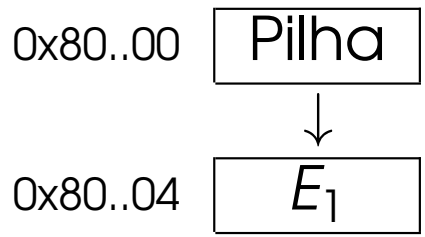
Pilha

- Implementação com lista encadeada
  - Inicialização das estruturas
  - A pilha está vazia com o topo da pilha com referência nula



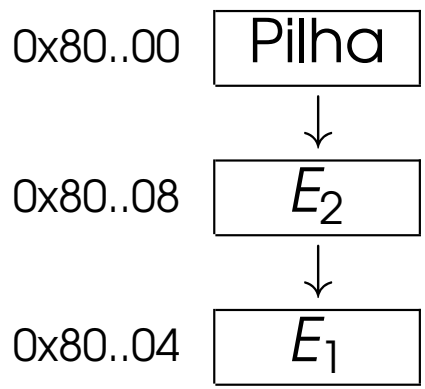
Pilha

- Implementação com lista encadeada
  - Empilhando os elementos
  - O elemento E_1 é empilhado e o topo da pilha é ajustado



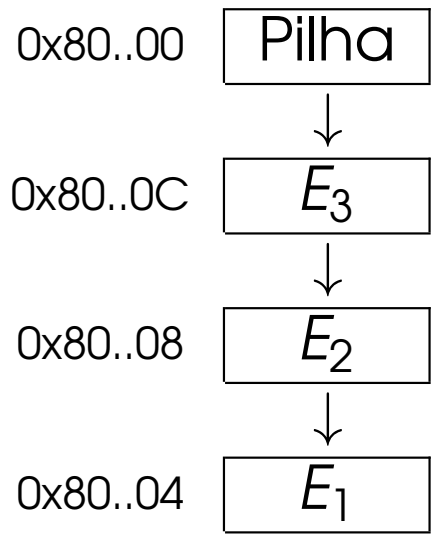
Pilha

- Implementação com lista encadeada
  - Empilhando os elementos
  - O elemento E_2 é empilhado e o topo da pilha é ajustado



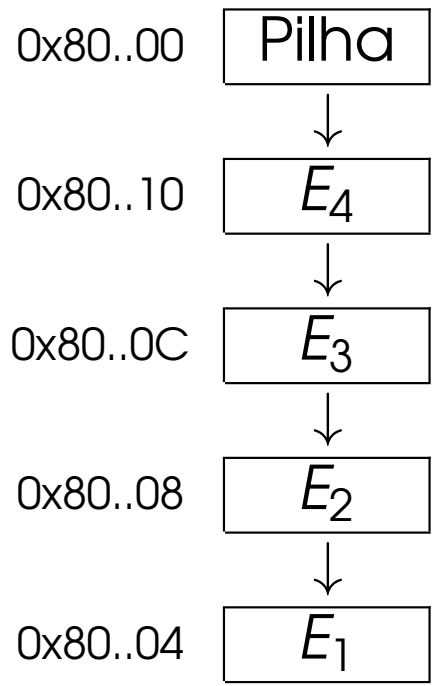
Pilha

- Implementação com lista encadeada
  - Empilhando os elementos
  - O elemento E_3 é empilhado e o topo da pilha é ajustado



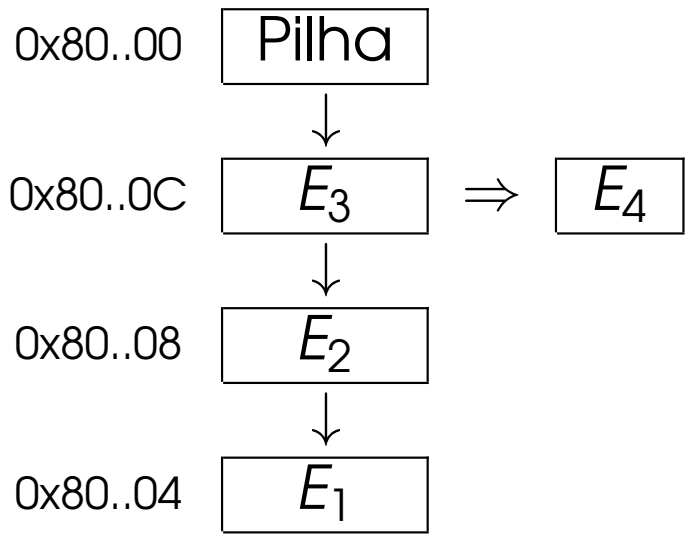
Pilha

- Implementação com lista encadeada
  - Empilhando os elementos
  - O elemento E_4 é empilhado e o topo da pilha é ajustado



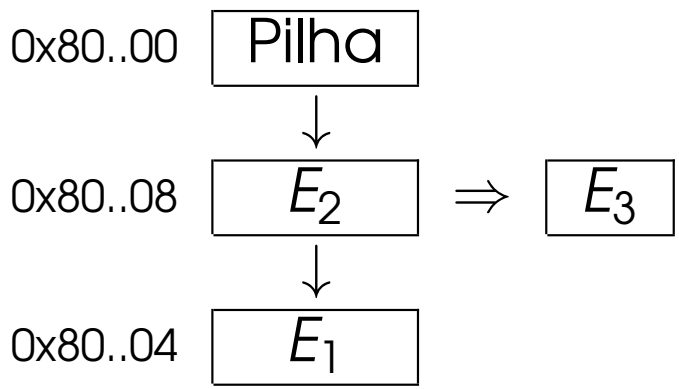
Pilha

- Implementação com lista encadeada
  - Desempilhando os elementos
  - O elemento E_4 é desempilhado e o topo da pilha é ajustado



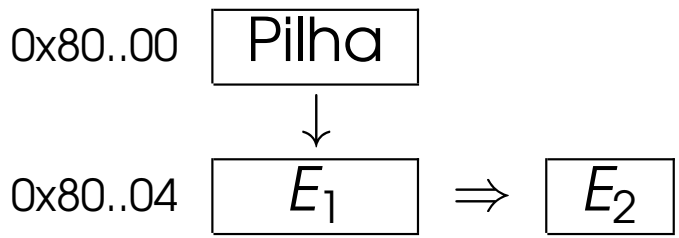
Pilha

- Implementação com lista encadeada
  - Desempilhando os elementos
  - O elemento E_3 é desempilhado e o topo da pilha é ajustado



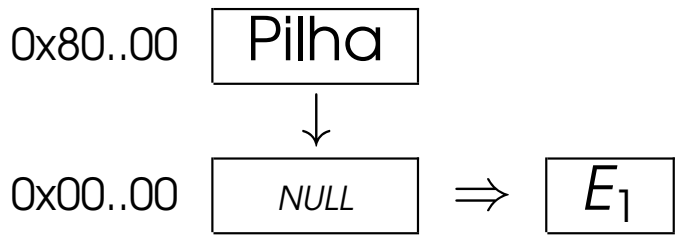
Pilha

- Implementação com lista encadeada
  - Desempilhando os elementos
  - O elemento E_2 é desempilhado e o topo da pilha é ajustado



Pilha

- Implementação com lista encadeada
  - Desempilhando os elementos
  - A pilha está vazia com o topo da pilha com referência nula

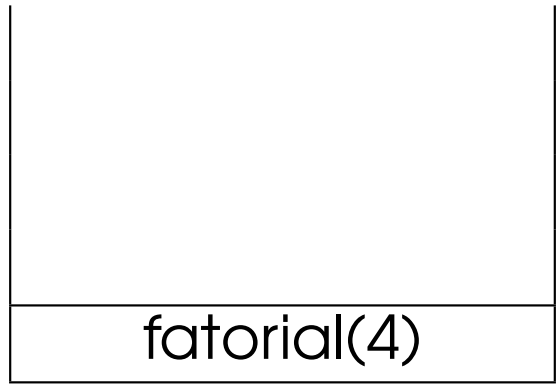


Pilha

- Análise de complexidade
  - Espaço $\Theta(n)$
  - Tempo
    - Empilhar $O(1)$
    - Desempilhar $O(1)$

Aplicações

- Suporte a recursão
  - Função recursiva **unsigned int** fatorial(**unsigned int** n)



Aplicações

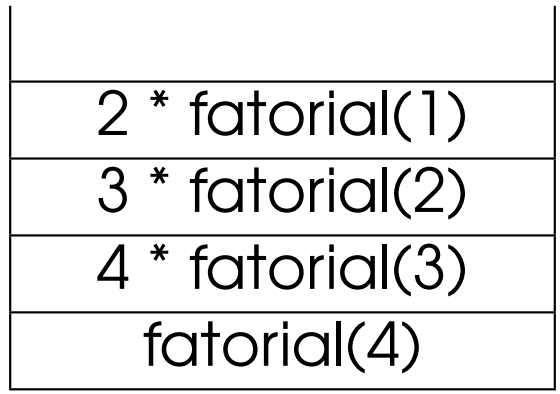
- Suporte a recursão
  - Função recursiva **unsigned int** fatorial(**unsigned int** n)

Aplicações

- Suporte a recursão
  - Função recursiva **unsigned int** fatorial(**unsigned int** n)

Aplicações

- Suporte a recursão
  - Função recursiva **unsigned int** fatorial(**unsigned int** n)



Aplicações

- Suporte a recursão
  - Função recursiva **unsigned int** fatorial(**unsigned int** n)

1 * fatorial (0)
2 * fatorial(1)
3 * fatorial(2)
4 * fatorial(3)
fatorial(4)

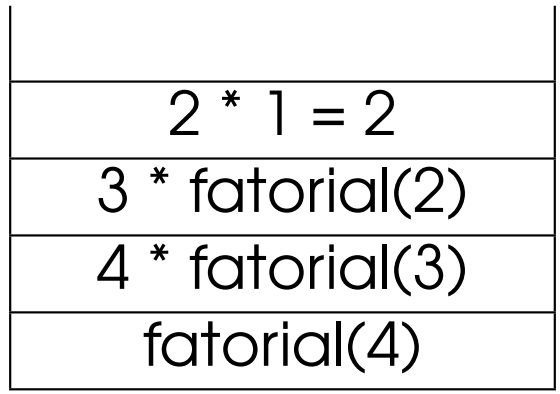
Aplicações

- Suporte a recursão
  - Função recursiva **unsigned int** fatorial(**unsigned int** n)

1 * 1 = 1
2 * fatorial(1)
3 * fatorial(2)
4 * fatorial(3)
fatorial(4)

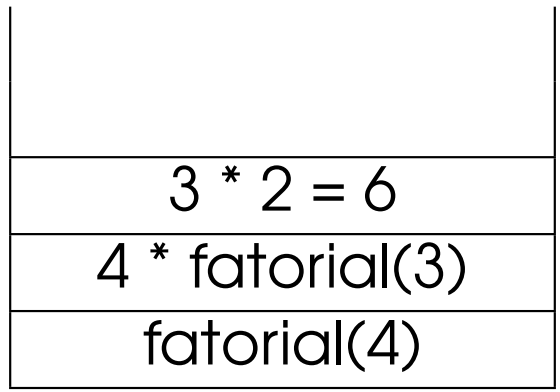
Aplicações

- Suporte a recursão
  - Função recursiva **unsigned int** fatorial(**unsigned int** n)



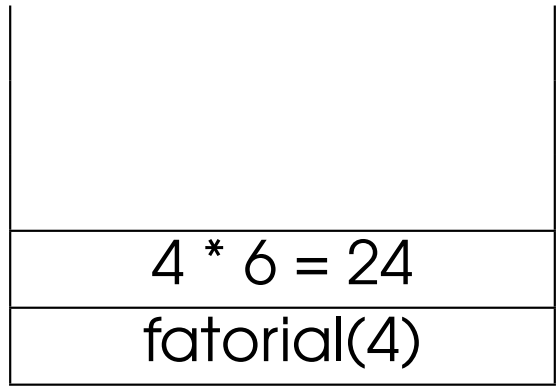
Aplicações

- Suporte a recursão
  - Função recursiva **unsigned int** fatorial(**unsigned int** n)



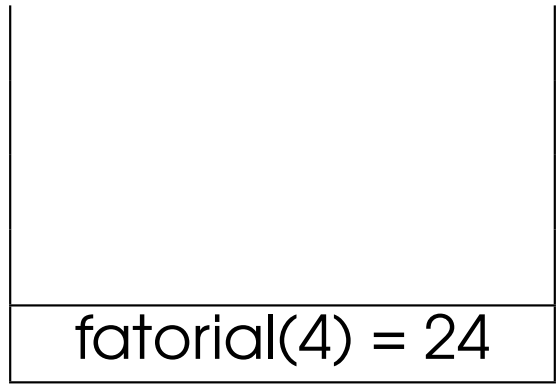
Aplicações

- Suporte a recursão
  - Função recursiva **unsigned int** fatorial(**unsigned int** n)



Aplicações

- Suporte a recursão
  - Função recursiva **unsigned int** fatorial(**unsigned int** n)



Exercício

- A empresa de tecnologia Poxim Tech está desenvolvendo um sistema de impressão centralizado para otimizar a utilização das impressoras e reduzir os custos com manutenção e reposição de suprimentos
  - Todos os documentos enviados para impressão são organizados por ordem de chegada, sendo despachados para uma impressora que estiver ociosa
  - Os nomes dos arquivos e das impressoras possuem até 50 caracteres, sendo limitados a letras e números
  - A quantidade de páginas do documento determina quanto tempo será utilizado na impressora alocada, assumindo que todas as impressoras possuem a mesma velocidade de impressão
  - Após cada impressão ser concluída, as folhas impressas de todas as impressoras são automaticamente recolhidas e empilhadas para serem entregues

Exercício

- Formato do arquivo de entrada
  - [#*n*]
  - [*Impressora* 1]
  - ...
  - [*Impressora n*]
  - [#*m*]
  - [*Documento* 1] [#*Páginas* 1]
  - ...
  - [*Documento m*] [#*Páginas m*]

```
2
jatodetinta
laser
6
sigaa 2
bbbbb 7
documento 3
abc 2
xyz 5
aaaaa 6
```

Exercício

- Formato do arquivo de saída
  - É exibido o nome da impressora alocada e o histórico de impressão dos documentos na pilha
  - Após as alocações são listados o total de páginas e os documentos em ordem de impressão

```
[jatodetinta] sigaa-2p
[laser] bbbbb-7p
[jatodetinta] documento-3p, sigaa-2p
[jatodetinta] abc-2p, documento-3p, sigaa-2p
[jatodetinta] xyz-5p, abc-2p, documento-3p,
sigaa-2p
[laser] aaaaa-6p, bbbbb-7p
25p
aaaaa-6p
xyz-5p
bbbbb-7p
abc-2p
documento-3p
sigaa-2p
```