Estrutura de fila e de pilha

Estruturas de Dados

Bruno Prado

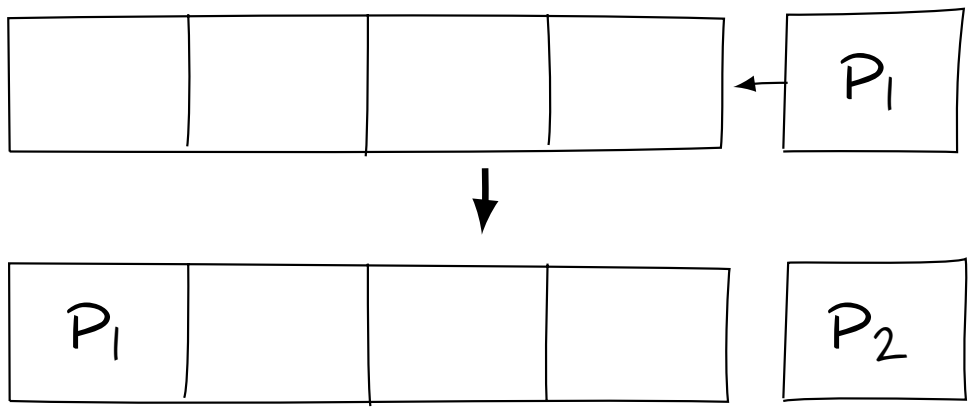
Departamento de Computação / UFS

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

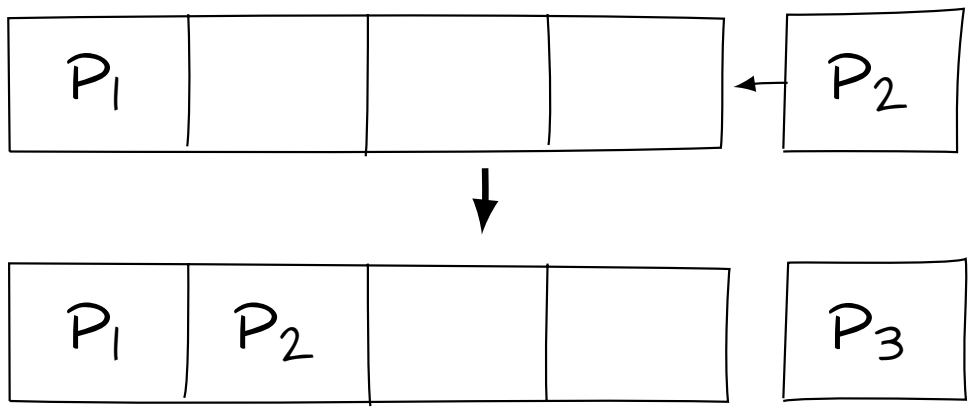
Introdução

- Fila de pessoas
  - Enfileirar (*push_back*)



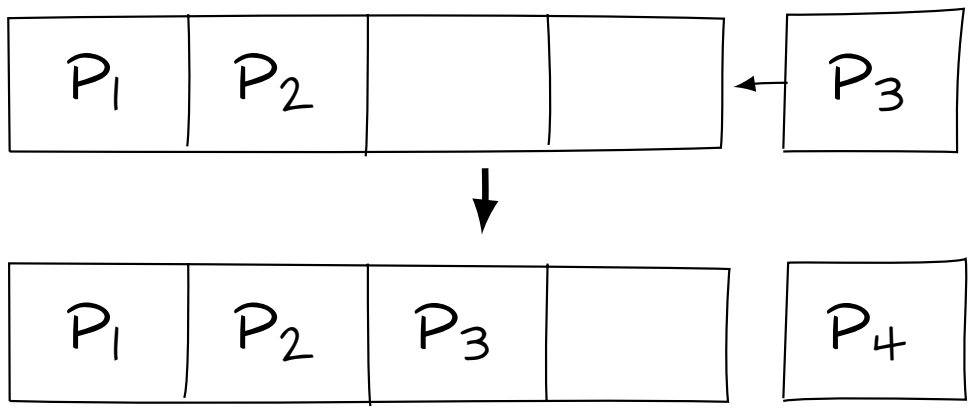
Introdução

- Fila de pessoas
  - Enfileirar (*push_back*)



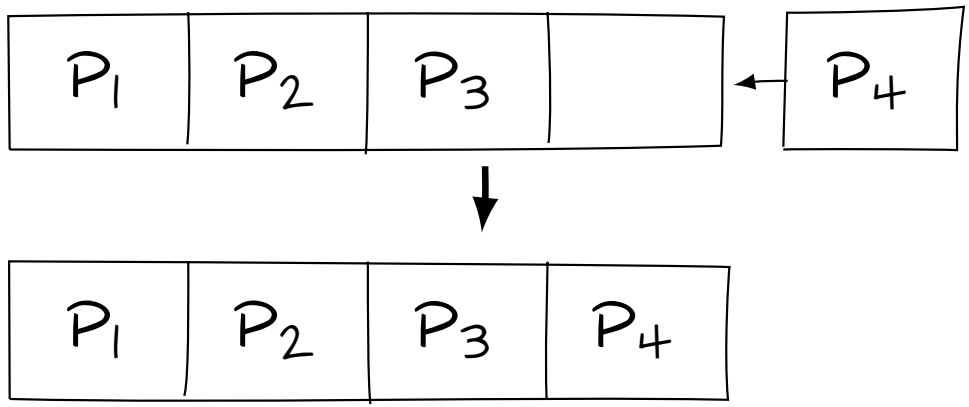
Introdução

- Fila de pessoas
  - Enfileirar (*push_back*)



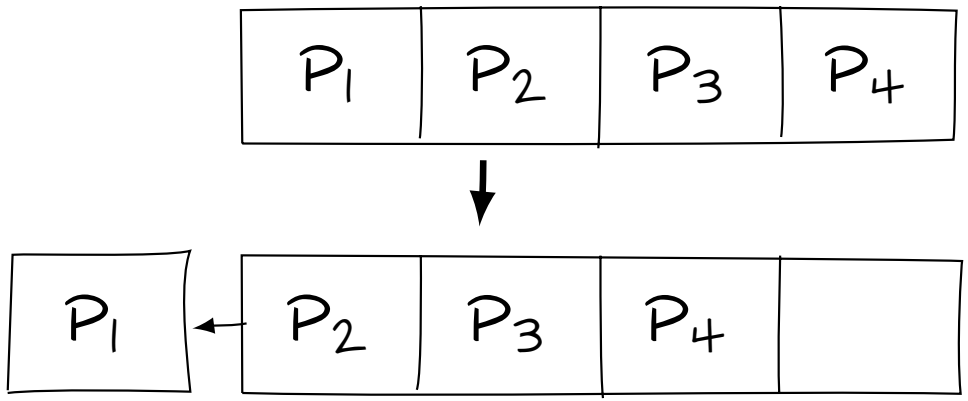
Introdução

- Fila de pessoas
  - Enfileirar (*push_back*)



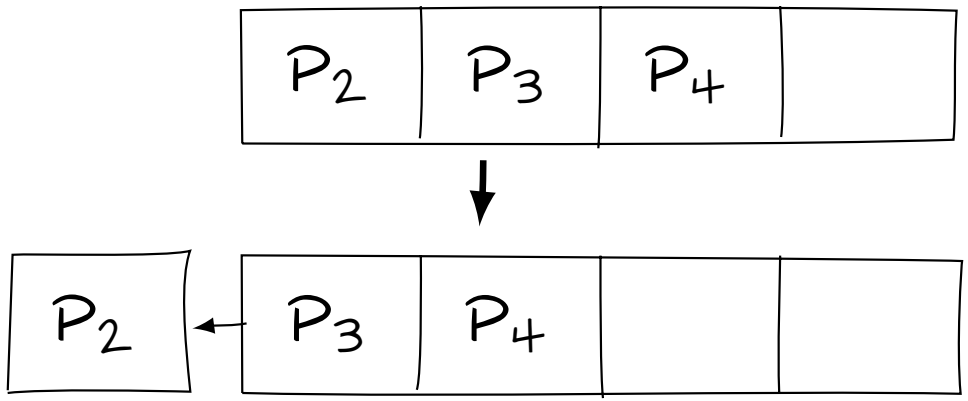
Introdução

- Fila de pessoas
  - Desenfileirando (*pop_front*)



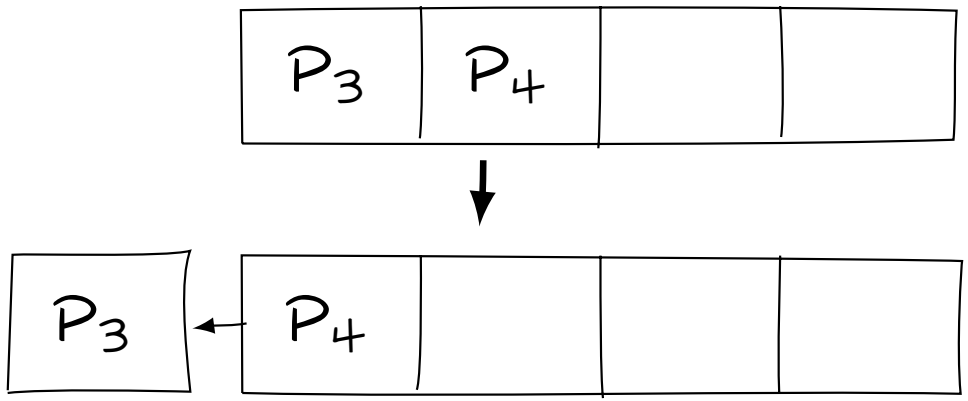
Introdução

- Fila de pessoas
  - Desenfileirando (*pop_front*)



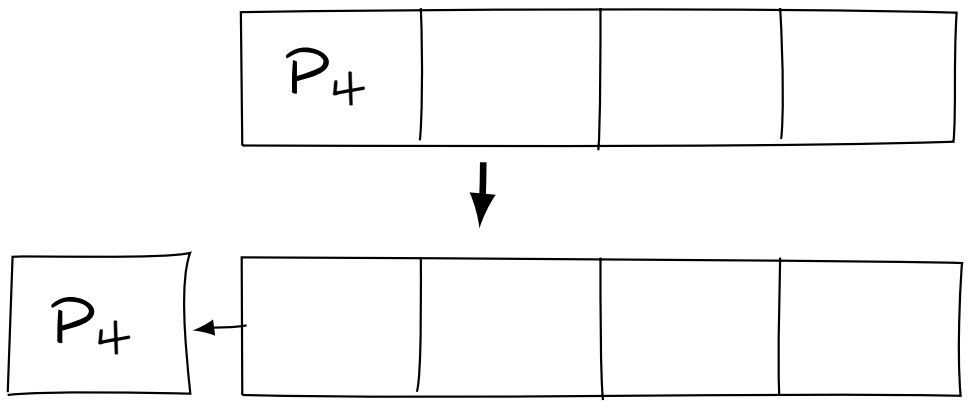
Introdução

- Fila de pessoas
  - Desenfileirando (*pop_front*)



Introdução

- Fila de pessoas
  - Desenfileirando (*pop_front*)

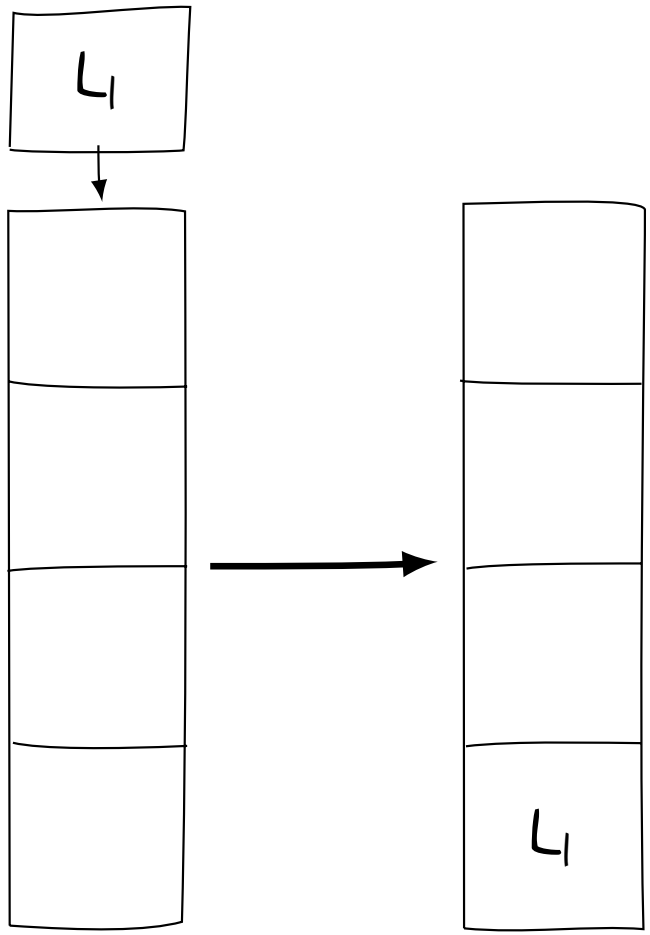


Introdução

- 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

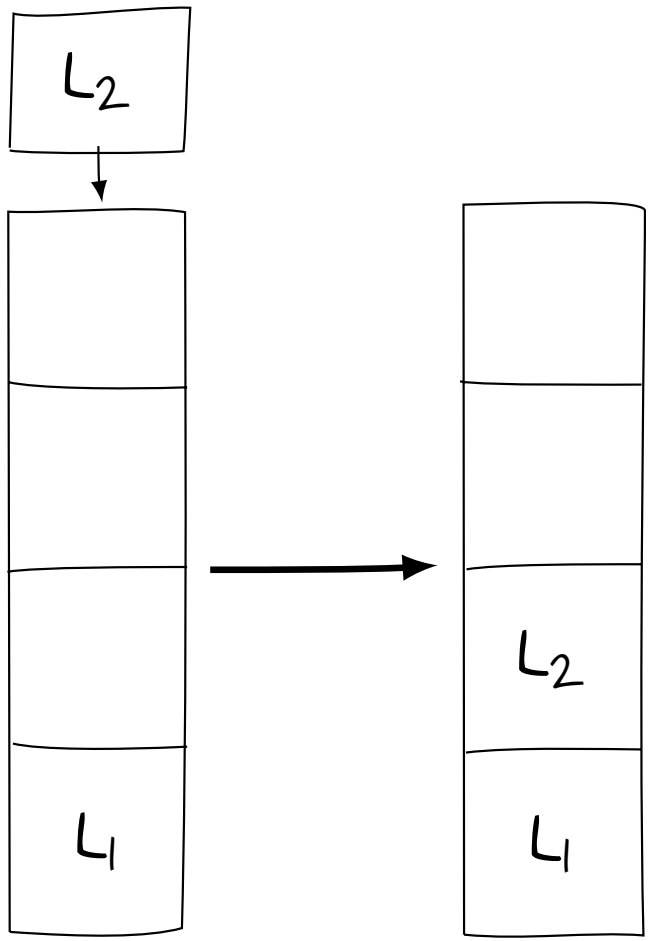
Introdução

- Pilha de livros
  - Empilhando (*push*)



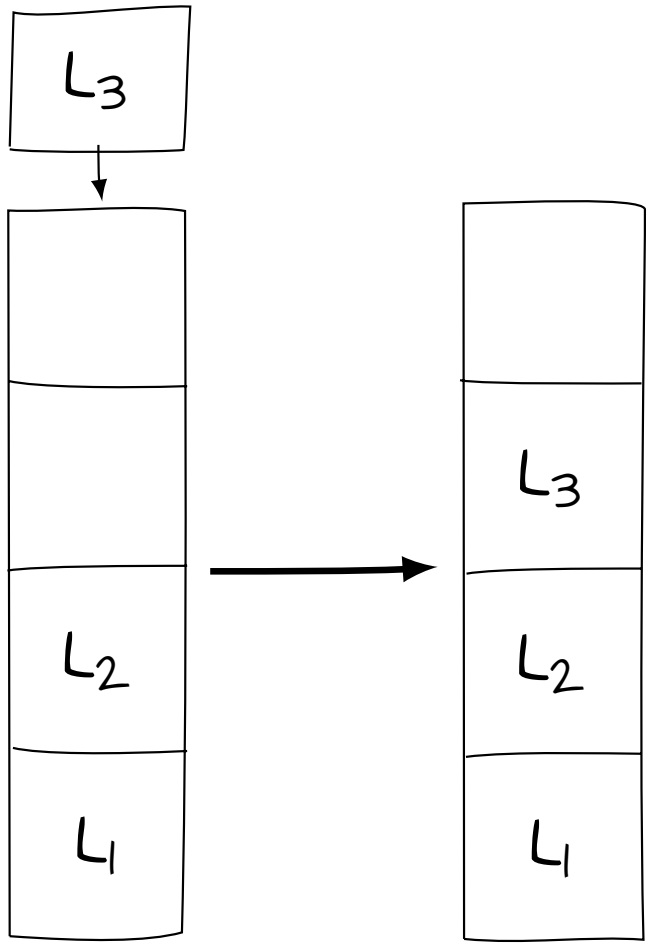
Introdução

- Pilha de livros
  - Empilhando (*push*)



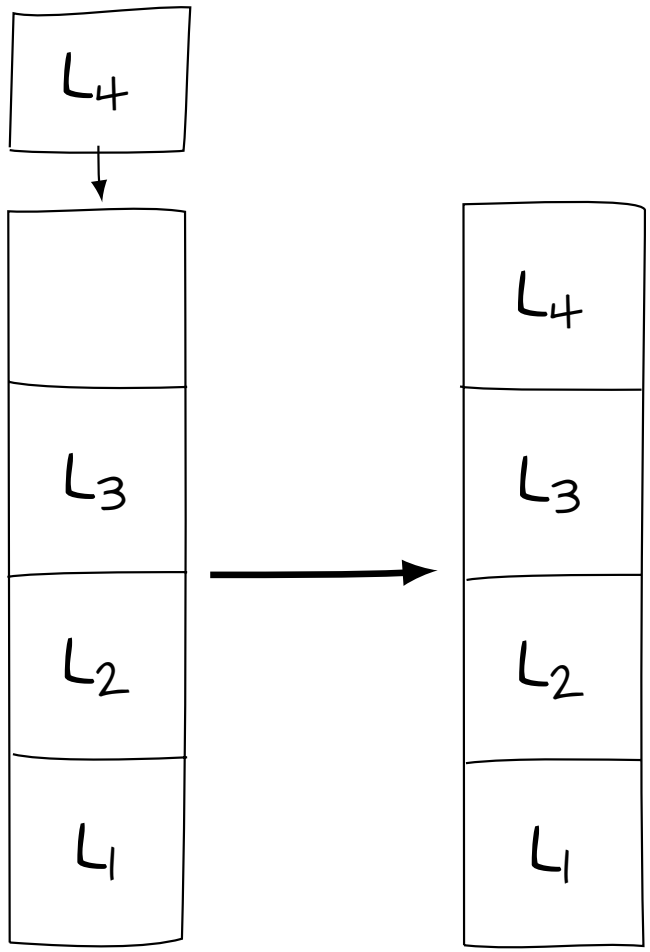
Introdução

- Pilha de livros
  - Empilhando (*push*)



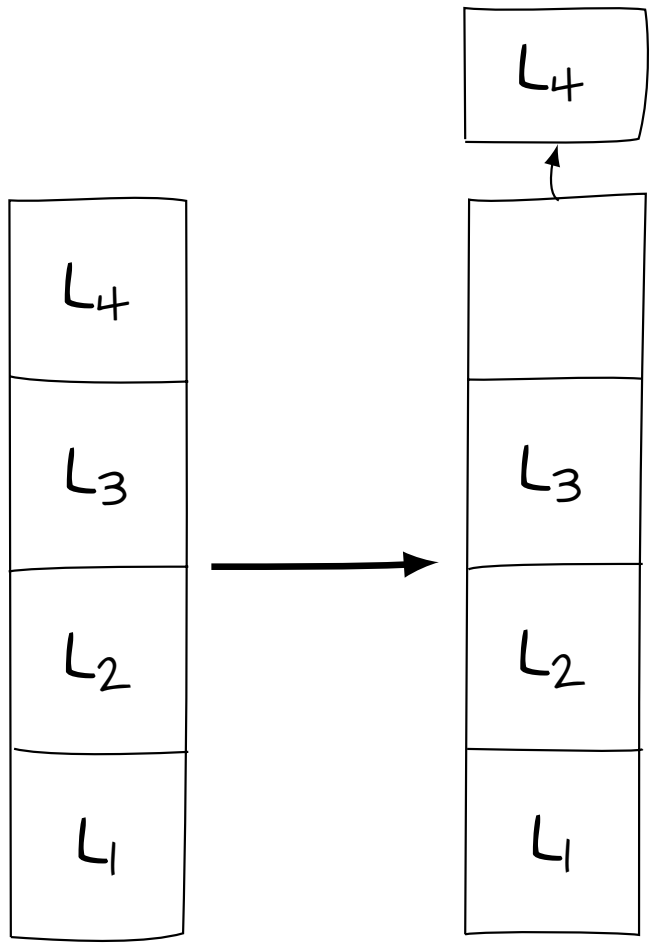
Introdução

- Pilha de livros
  - Empilhando (*push*)



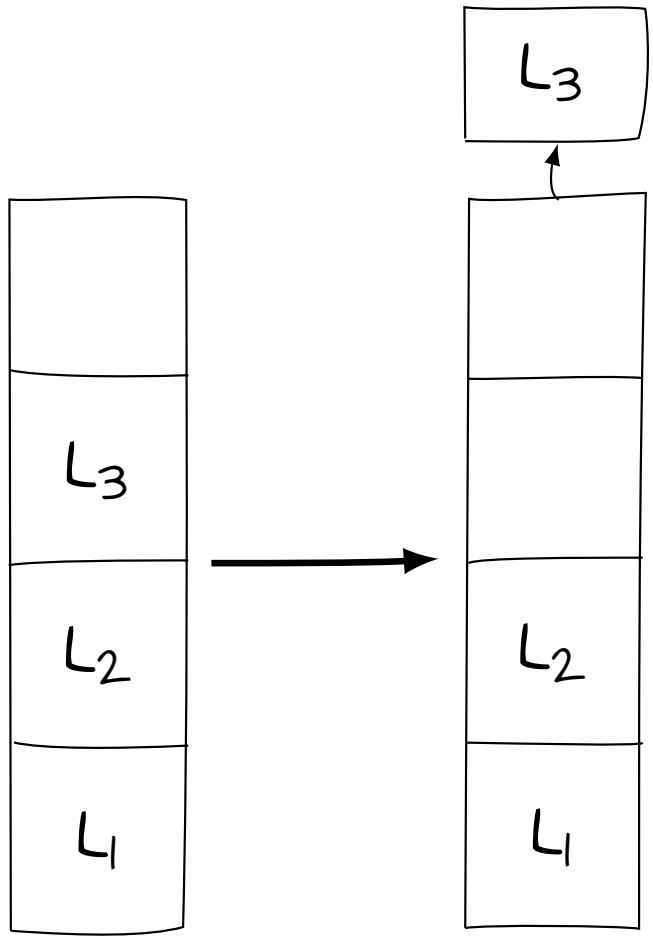
Introdução

- Pilha de livros
  - Desempilhando (*pop*)



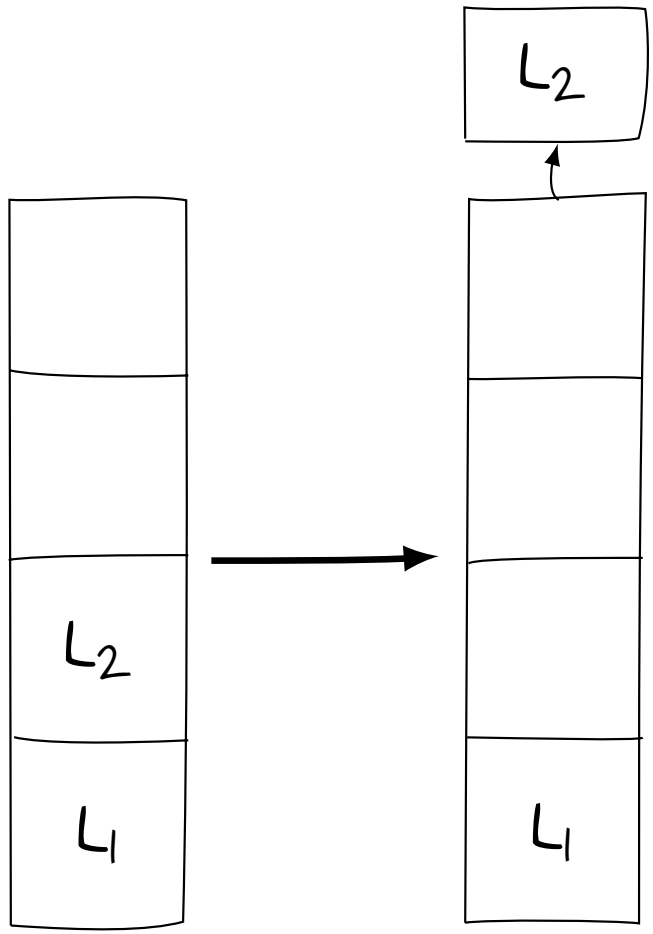
Introdução

- Pilha de livros
  - Desempilhando (*pop*)



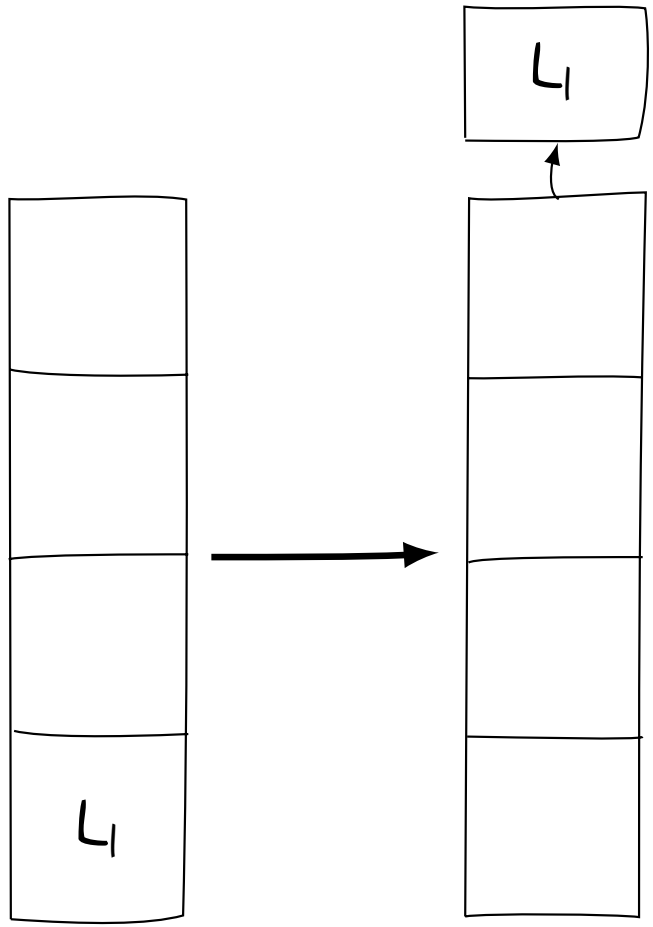
Introdução

- Pilha de livros
  - Desempilhando (*pop*)



Introdução

- Pilha de 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*)

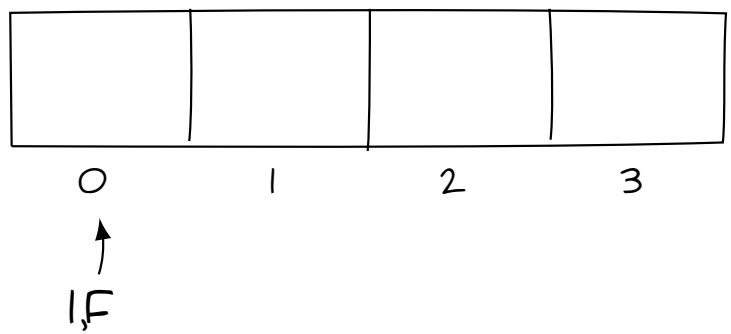
Fila

- Implementação em C
  - Definição da estrutura com vetor

```
1  // Padrão de tipos por tamanho
2  #include <stdint.h>
3  // Estrutura de fila
4  typedef struct fila {
5      // Capacidade alocada do vetor
6      size_t capacidade;
7      // Índice de inicio do vetor (I)
8      size_t inicio;
9      // Índice de fim do vetor (F)
10     size_t fim;
11     // Tamanho da fila
12     size_t tamanho;
13     // Vetor sem sinal de 32 bits
14     uint32_t* vetor;
15 } fila;
```

Fila

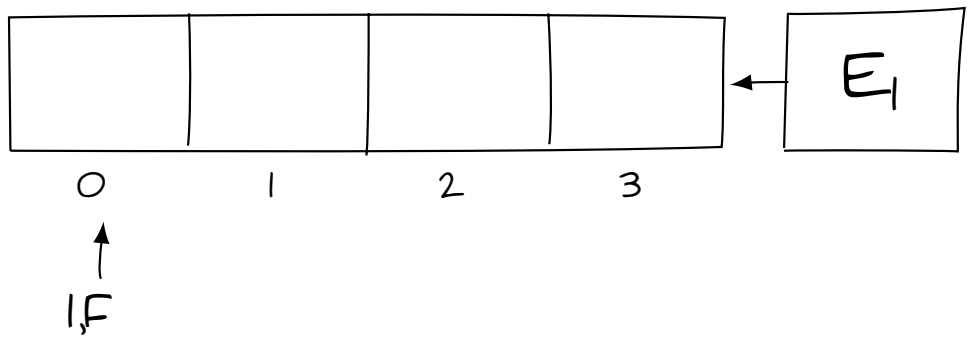
- Implementação com vetor
  - Inicialização da estrutura
  - É feita a alocação do vetor com capacidade 4 e com índice de início e de tamanho com valor 0, indicando que não possui nenhum elemento



Capacidade = 4, Início = 0, Fim = 0, Tamanho = 0

Fila

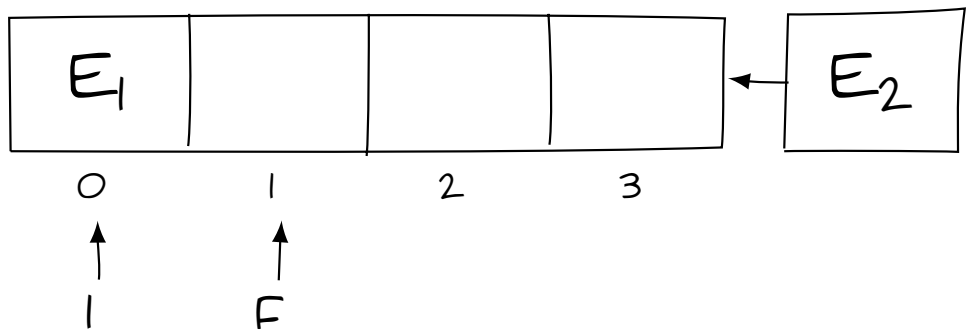
- Implementação com vetor
  - Enfileirando os elementos
  - É feito o incremento do tamanho da fila e a posição $(Inicio + Fim) \bmod Capacidade$ recebe o valor do elemento E_1



Capacidade = 4, Início = 0, Fim = 0, Tamanho = 0

Fila

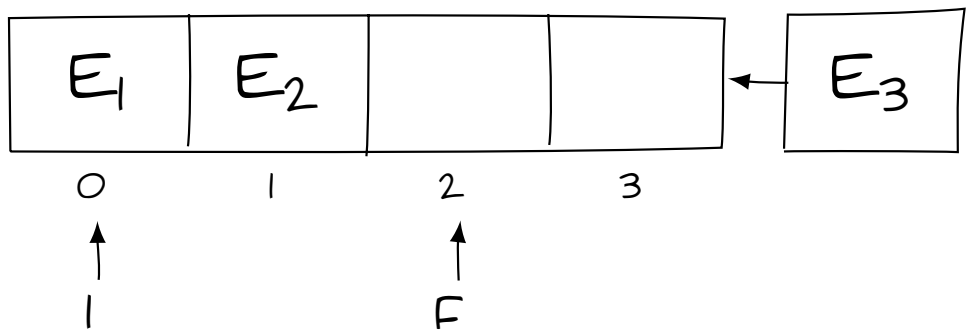
- Implementação com vetor
  - Enfileirando os elementos
  - É feito o incremento do tamanho da fila e a posição $(Inicio + Fim)$ mod $Capacidade$ recebe o valor do elemento E_2



Capacidade = 4, Início = 0, Fim = 1, Tamanho = 1

Fila

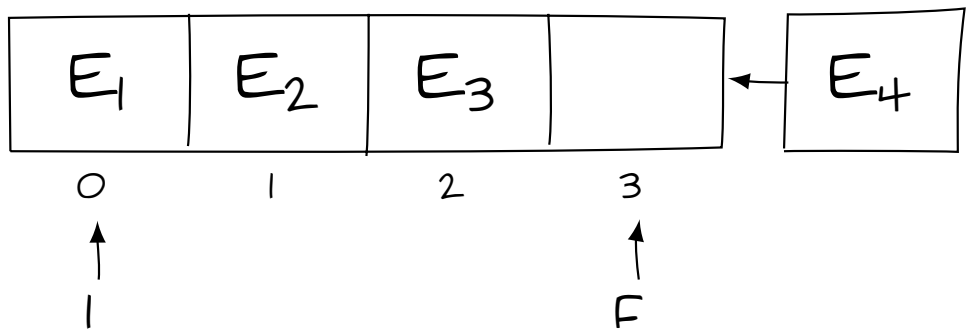
- Implementação com vetor
  - Enfileirando os elementos
  - É feito o incremento do tamanho da fila e a posição $(Inicio + Fim)$ mod $Capacidade$ recebe o valor do elemento E_3



Capacidade = 4, Início = 0, Fim = 2, Tamanho = 2

Fila

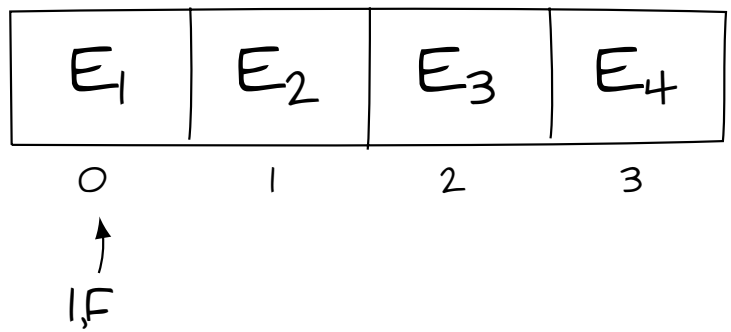
- Implementação com vetor
  - Enfileirando os elementos
  - É feito o incremento do tamanho da fila e a posição $(Inicio + Fim)$ mod $Capacidade$ recebe o valor do elemento E_4



Capacidade = 4, Início = 0, Fim = 3, Tamanho = 3

Fila

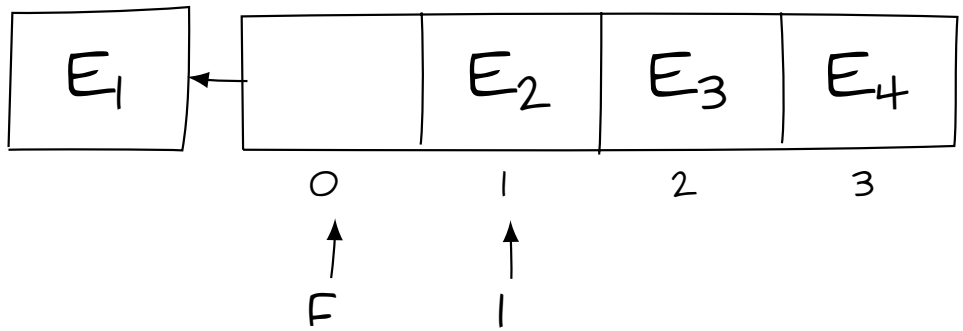
- Implementação com vetor
  - Todas as posições do vetor estão ocupadas, condição que pode ser verificada através da comparação da capacidade e do tamanho



Capacidade = 4, Início = 0, Fim = 0, Tamanho = 4

Fila

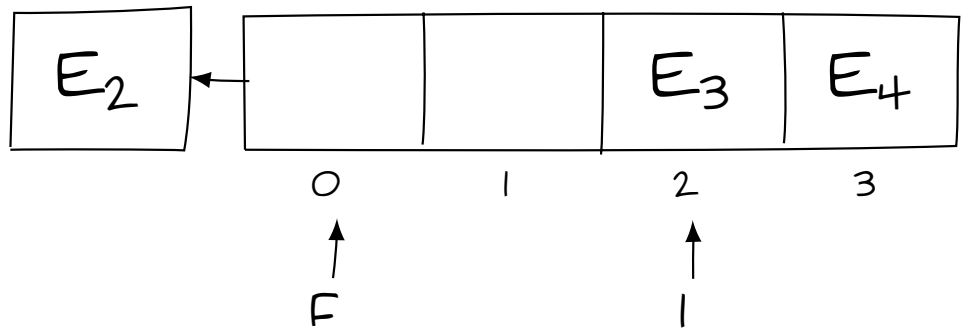
- Implementação com vetor
  - Desenfileirando os elementos
  - O elemento E_1 do início da fila é removido, com o incremento do $(Inicio + 1) \bmod Capacidade$



Capacidade = 4, Início = 1, Fim = 0, Tamanho = 3

Fila

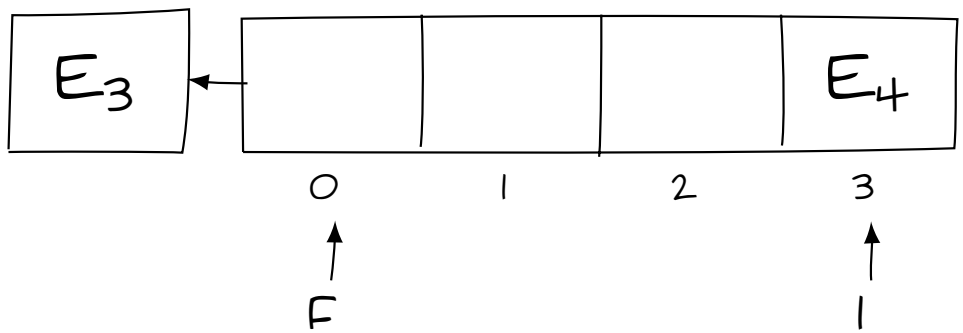
- Implementação com vetor
  - Desenfileirando os elementos
  - O elemento E_2 do início da fila é removido, com o incremento do $(Inicio + 1) \bmod Capacidade$



Capacidade = 4, Início = 2, Fim = 0, Tamanho = 2

Fila

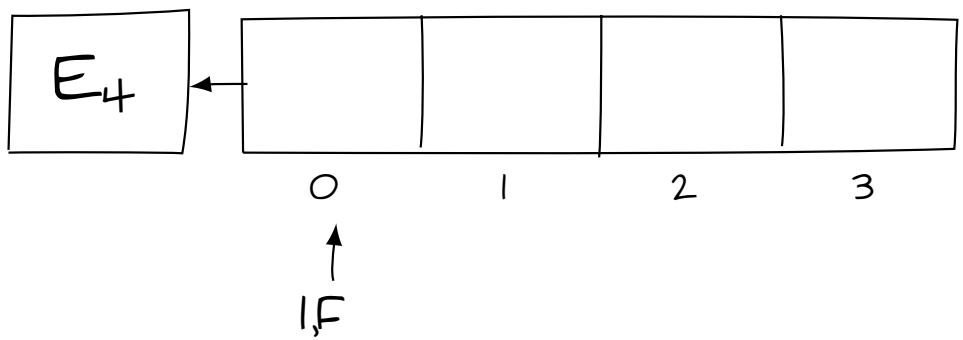
- Implementação com vetor
  - Desenfileirando os elementos
  - O elemento E_3 do início da fila é removido, com o incremento do $(Inicio + 1) \bmod Capacidade$



Capacidade = 4, Início = 3, Fim = 0, 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, Início = 0, Fim = 0, Tamanho = 0

Fila

- Implementação em C
  - Definição da estrutura com lista

```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 // Estrutura de elemento
4 typedef struct elemento {
5     // Ponteiro para próximo elemento
6     elemento* P;
7     // Valor do elemento
8     uint32_t valor;
9 } elemento;
```

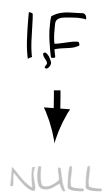
Fila

- Implementação em C
  - Definição da estrutura com lista

```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 // Estrutura de fila
4 typedef struct fila {
5     // Ponteiro inicial
6     elemento* I;
7     // Ponteiro final
8     elemento* F;
9 } fila;
```

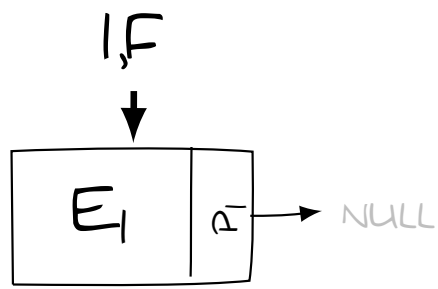
Fila

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



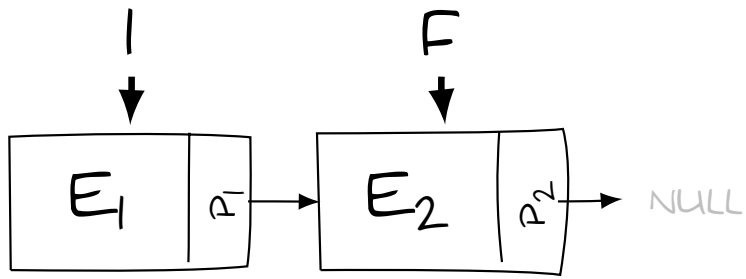
Fila

- 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



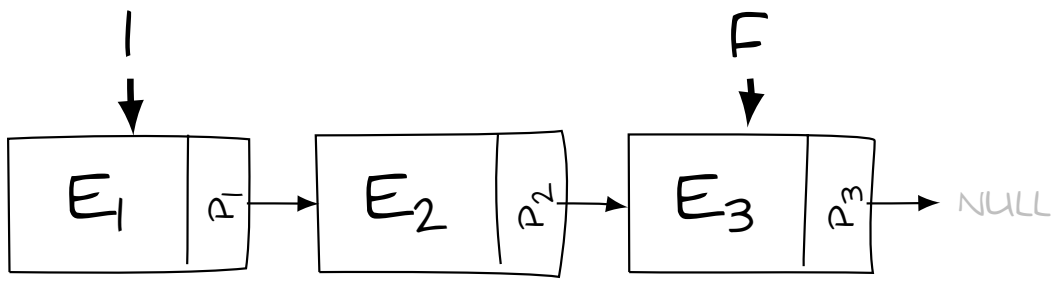
Fila

- 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



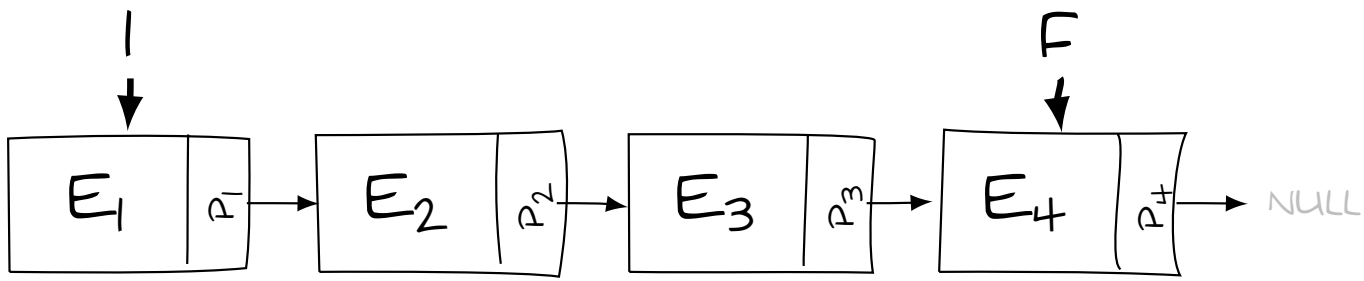
Fila

- 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



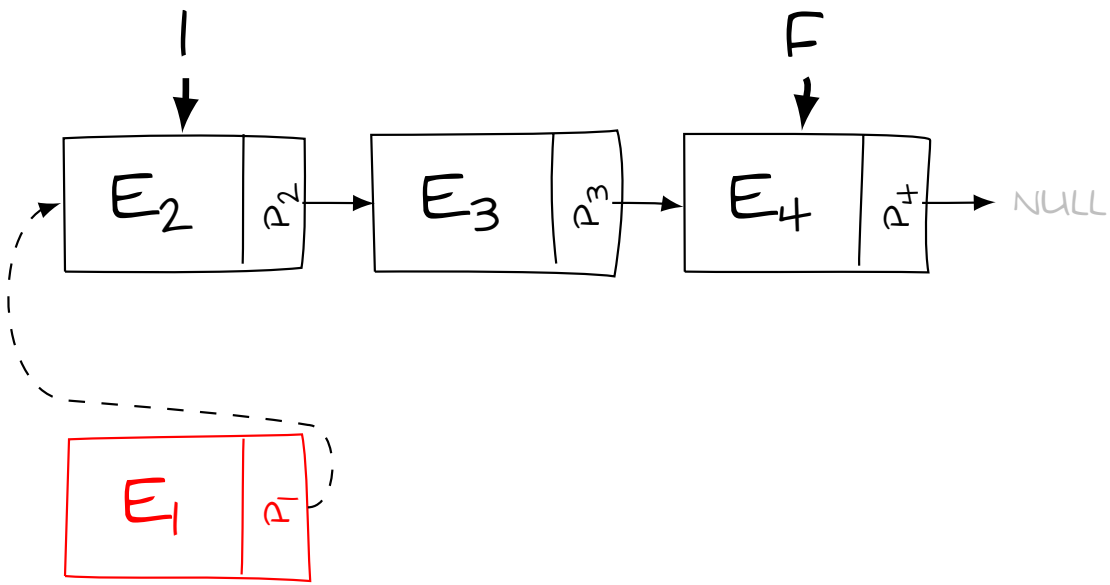
Fila

- 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



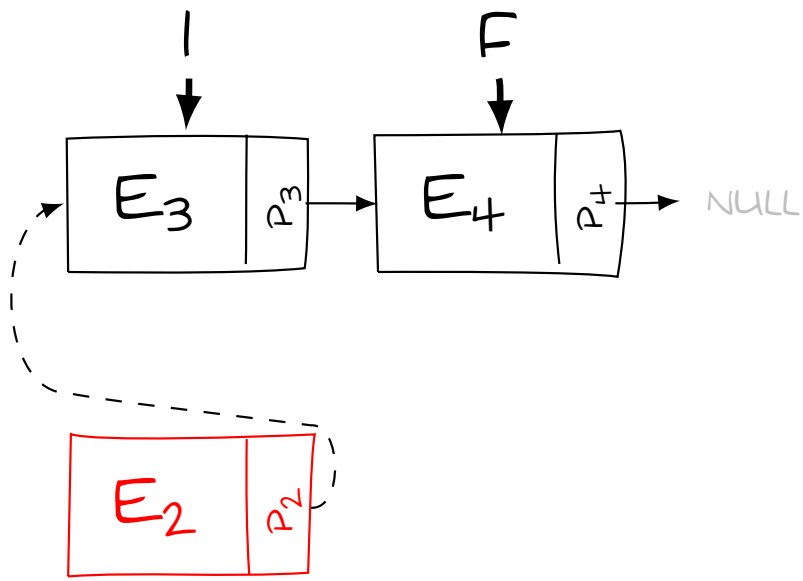
Fila

- 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



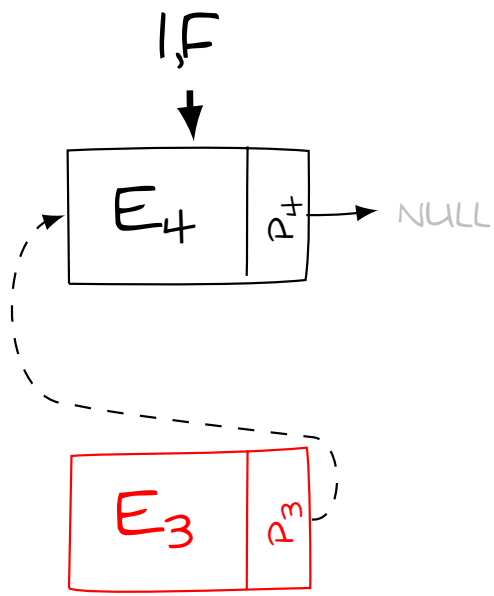
Fila

- 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



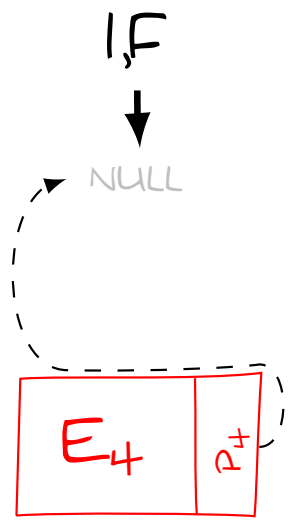
Fila

- 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



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

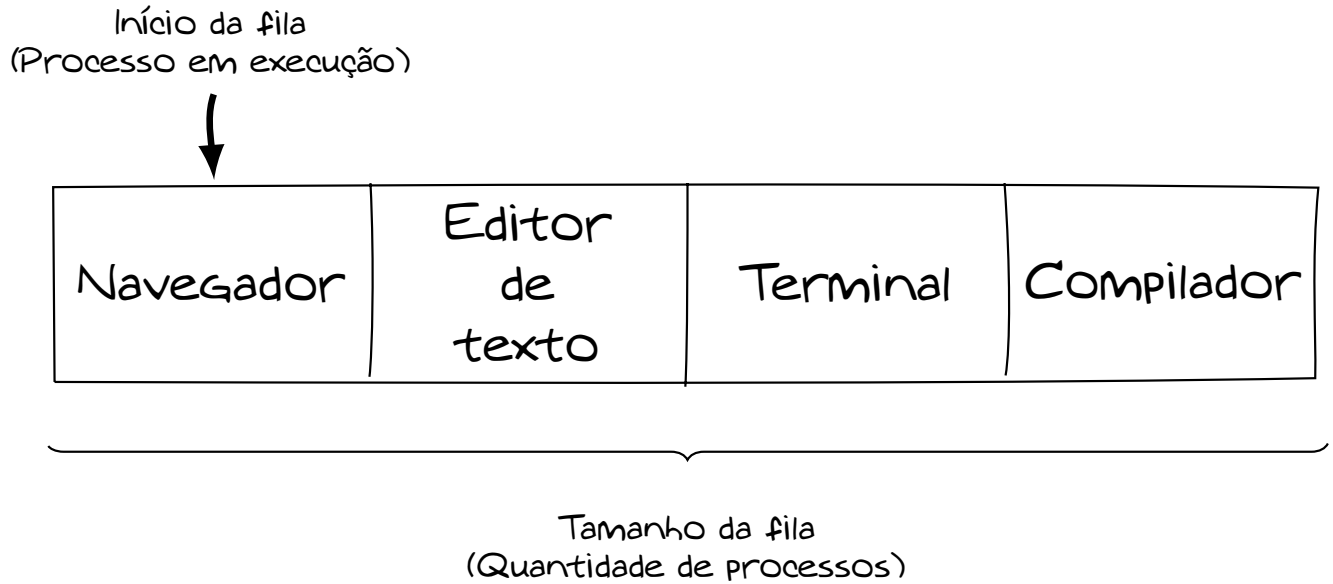


Fila

- Análise de complexidade
  - Enfileirar: $\Theta(1)$
  - Desenfileirar: $\Theta(1)$

Aplicações

- Escalonamento de processos



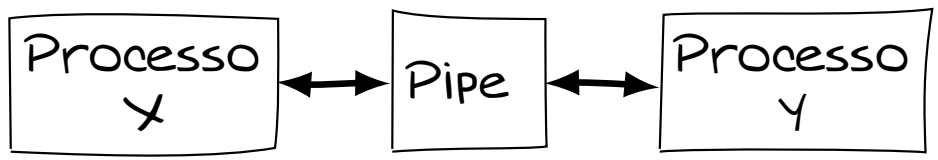
Aplicações

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

Aplicações

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

  - Interface de comunicação (*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*)

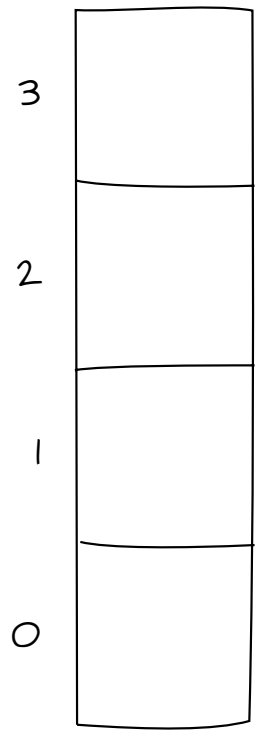
Pilha

- Implementação em C
  - Definição da estrutura com vetor

```
1  // Padrão de tipos por tamanho
2  #include <stdint.h>
3  // Estrutura de pilha
4  typedef struct pilha {
5      // Capacidade alocada do vetor
6      size_t capacidade;
7      // Topo da pilha
8      size_t topo;
9      // Vetor sem sinal de 32 bits
10     uint32_t* vetor;
11 } pilha;
```

Pilha

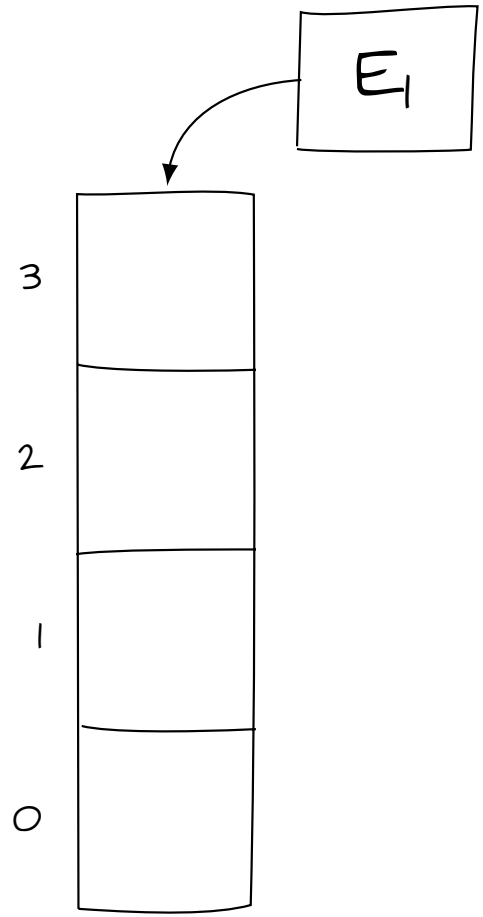
- Implementação com vetor
  - Inicialização da estrutura
  - É feita a alocação do vetor com capacidade 4 e com índice de topo negativo, indicando que não possui nenhum elemento



Capacidade = 4, Topo = -1

Pilha

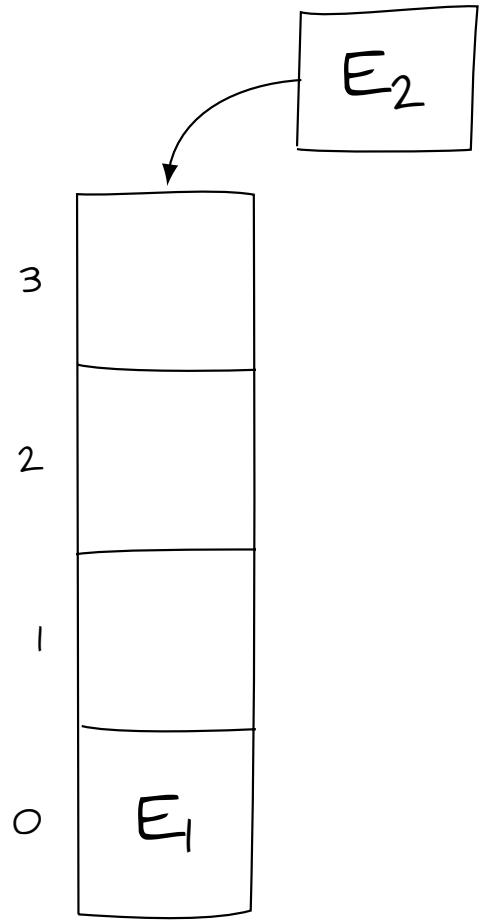
- Implementação com vetor
  - Empilhando os elementos
  - É feito o incremento do topo da pilha e a posição recebe o valor do elemento E_1



Capacidade = 4, Topo = -1

Pilha

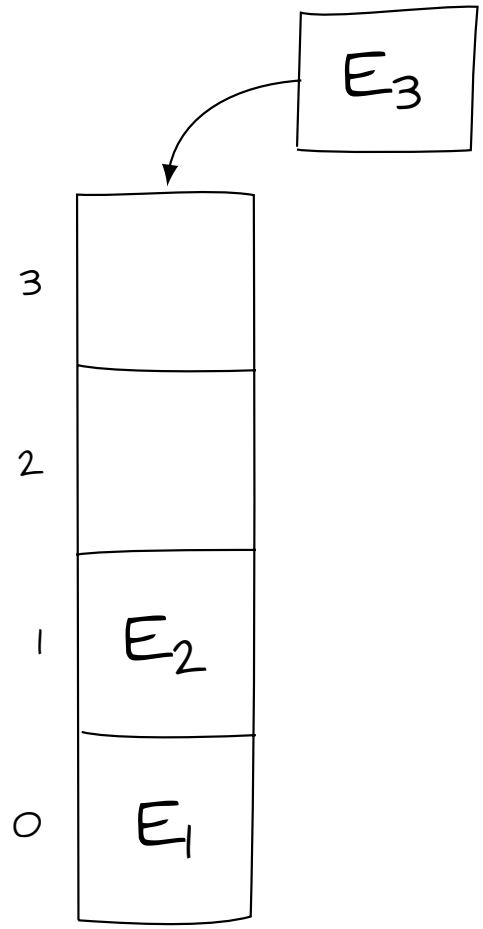
- Implementação com vetor
  - Empilhando os elementos
  - É feito o incremento do topo da pilha e a posição recebe o valor do elemento E_2



Capacidade = 4, Topo = 0

Pilha

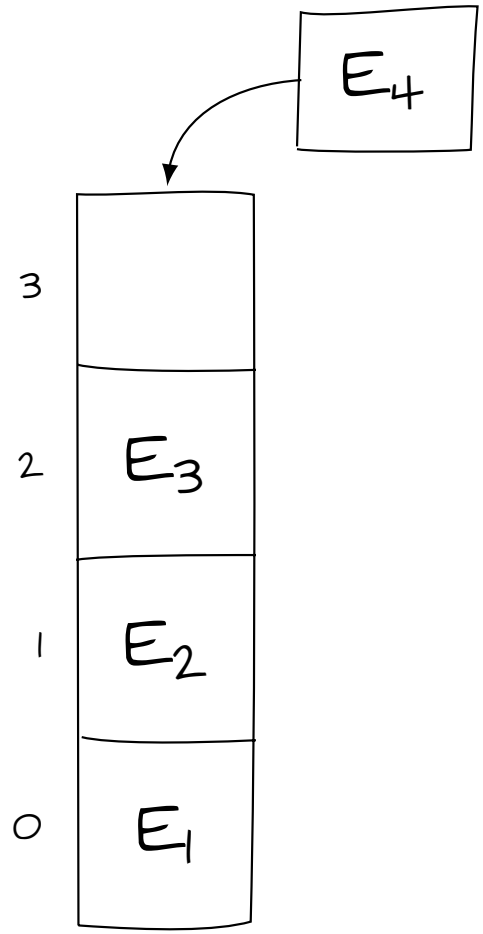
- Implementação com vetor
  - Empilhando os elementos
  - É feito o incremento do topo da pilha e a posição recebe o valor do elemento E_3



Capacidade = 4, Topo = 1

Pilha

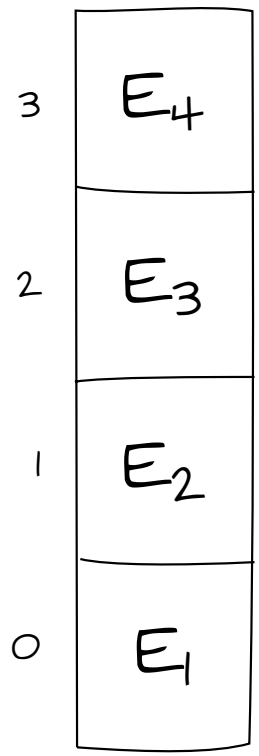
- Implementação com vetor
  - Empilhando os elementos
  - É feito o incremento do topo da pilha e a posição recebe o valor do elemento E_4



Capacidade = 4, Topo = 2

Pilha

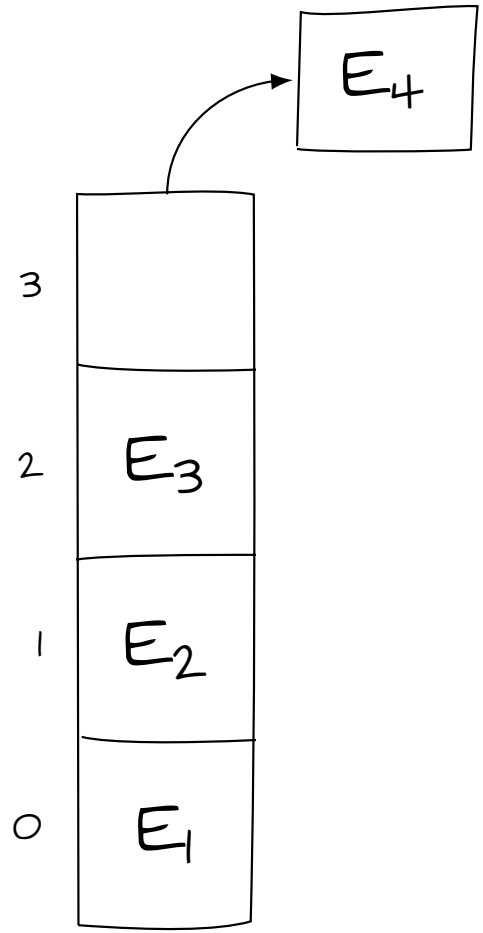
- Implementação com vetor
  - Todas as posições do vetor estão ocupadas, caso seja feita uma operação de empilhamento é preciso realizar a realocação do vetor e atualizar sua capacidade



Capacidade = 4, Topo = 3

Pilha

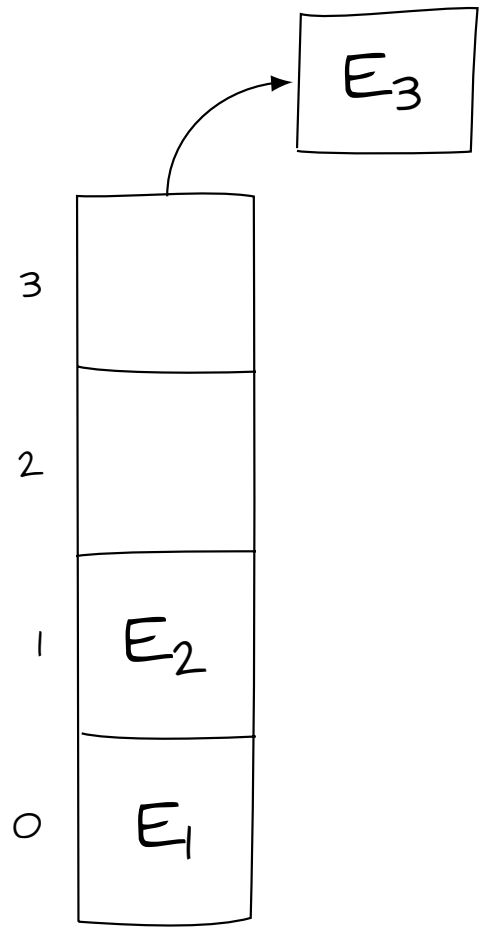
- Implementação com vetor
  - Desempilhando os elementos
  - O elemento E_4 é removido da pilha e o valor do topo é decrementado



Capacidade = 4, Topo = 2

Pilha

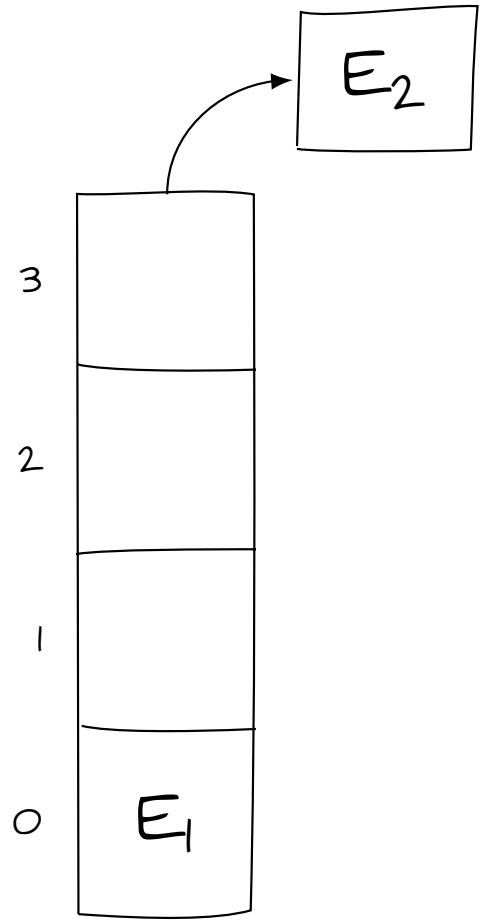
- Implementação com vetor
  - Desempilhando os elementos
  - O elemento E_3 é removido da pilha e o valor do topo é decrementado



Capacidade = 4, Topo = 1

Pilha

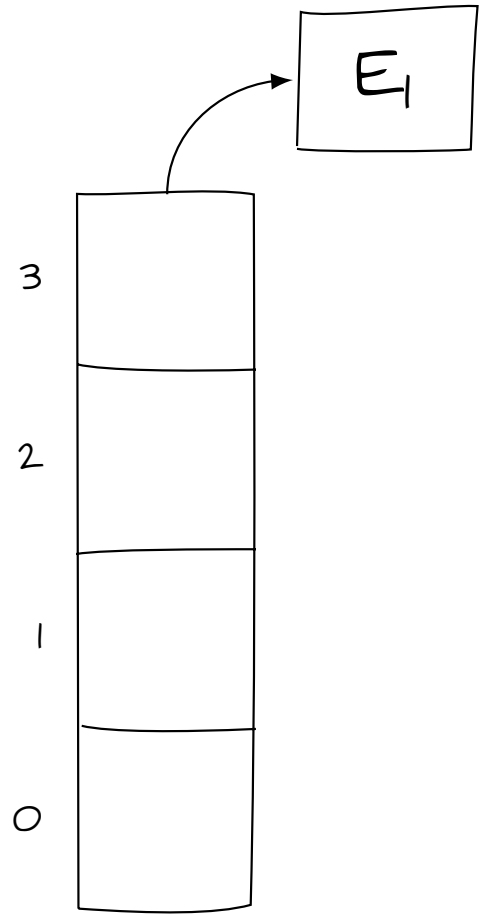
- Implementação com vetor
  - Desempilhando os elementos
  - O elemento E_2 é removido da pilha e o valor do topo é decrementado



Capacidade = 4, Topo = 0

Pilha

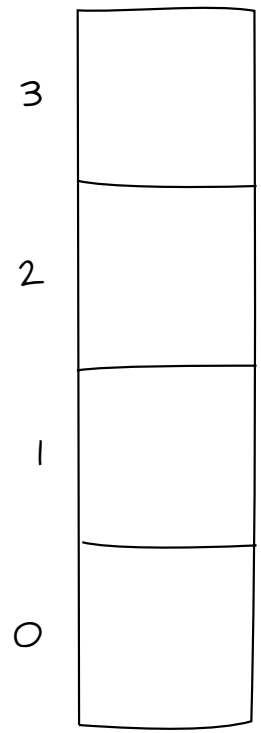
- Implementação com vetor
  - Desempilhando os elementos
  - O elemento E_1 é removido da pilha e o valor do topo é decrementado



Capacidade = 4, Topo = -1

Pilha

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



Capacidade = 4, Topo = -1

Pilha

- Implementação em C
  - Definição da estrutura com lista

```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 // Estrutura de elemento
4 typedef struct elemento {
5     // Ponteiro para próximo elemento
6     elemento* P;
7     // Valor do elemento
8     uint32_t valor;
9 } elemento;
```

Pilha

- Implementação em C
  - Definição da estrutura com lista

```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 // Estrutura de pilha
4 typedef struct pilha {
5     // Ponteiro para topo da pilha
6     elemento* T;
7 } pilha;
```

Pilha

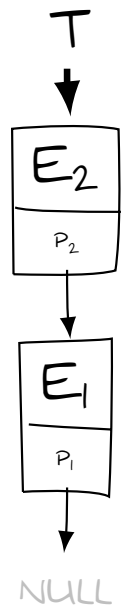
- Implementação com lista encadeada
  - Inicialização da estrutura
  - A pilha está vazia com o topo da pilha com referência nula

Pilha

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

Pilha

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

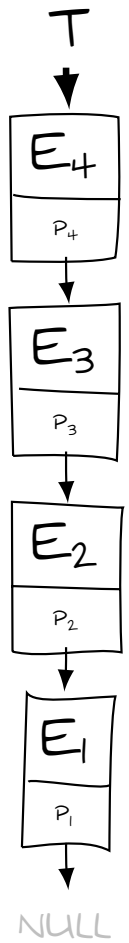


Pilha

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

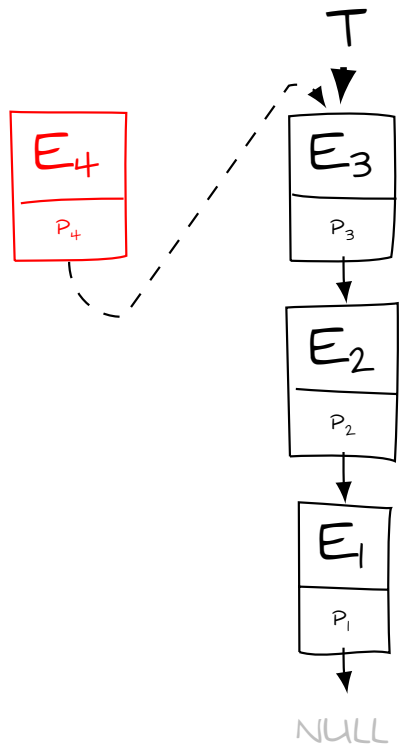
Pilha

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



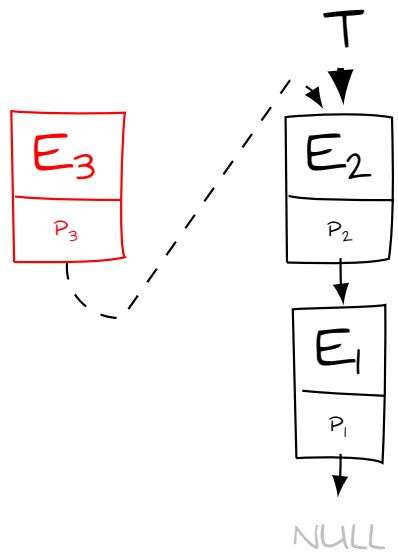
Pilha

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



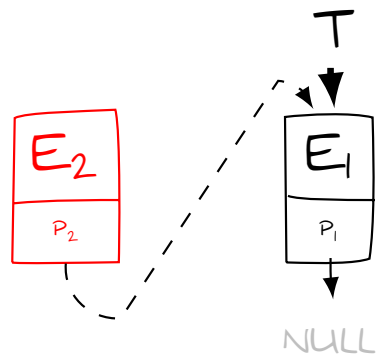
Pilha

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



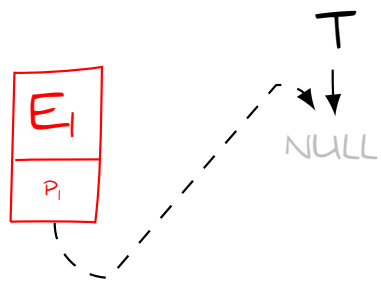
Pilha

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



Pilha

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

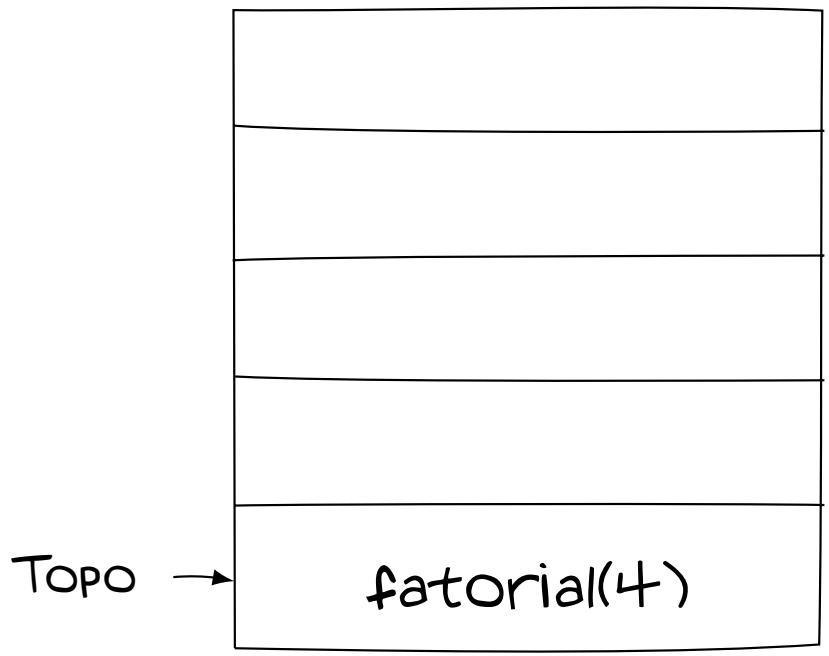


Pilha

- Análise de complexidade
  - Empilhar: $\Theta(1)$
  - Desempilhar: $\Theta(1)$

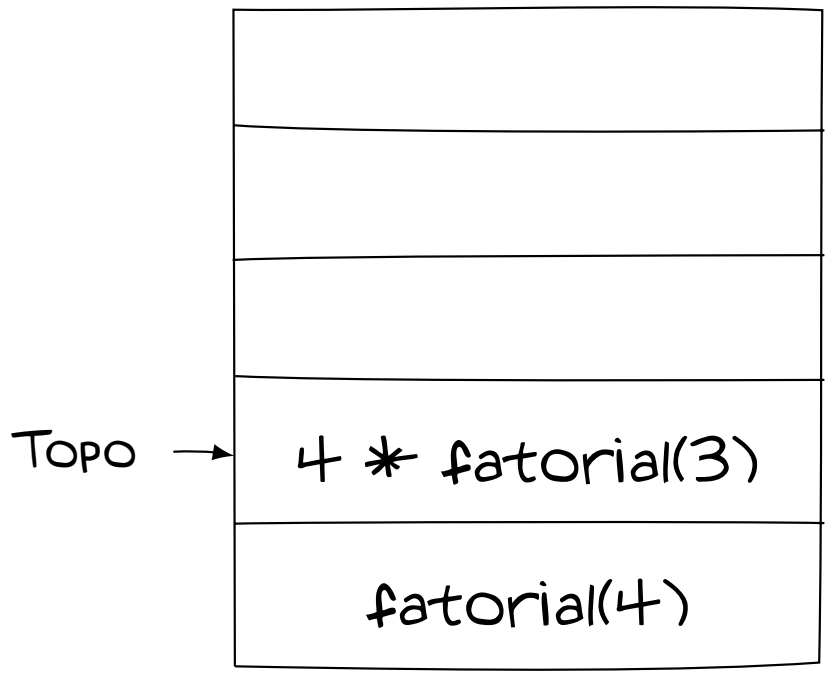
Aplicações

- Suporte a recursão
  - Função recursiva **uint64_t** fatorial(**uint32_t** n)



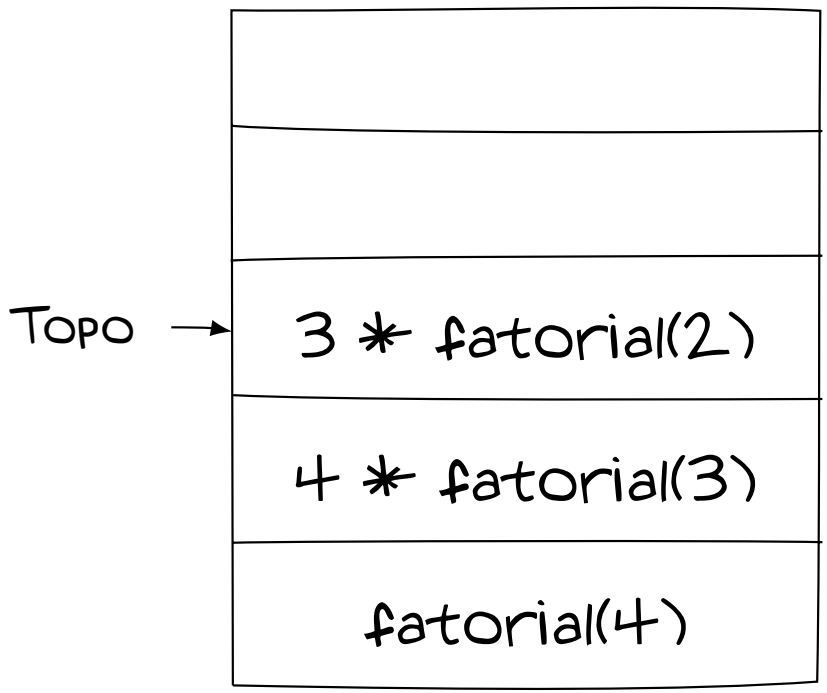
Aplicações

- Suporte a recursão
  - Função recursiva **uint64_t** fatorial(**uint32_t** n)



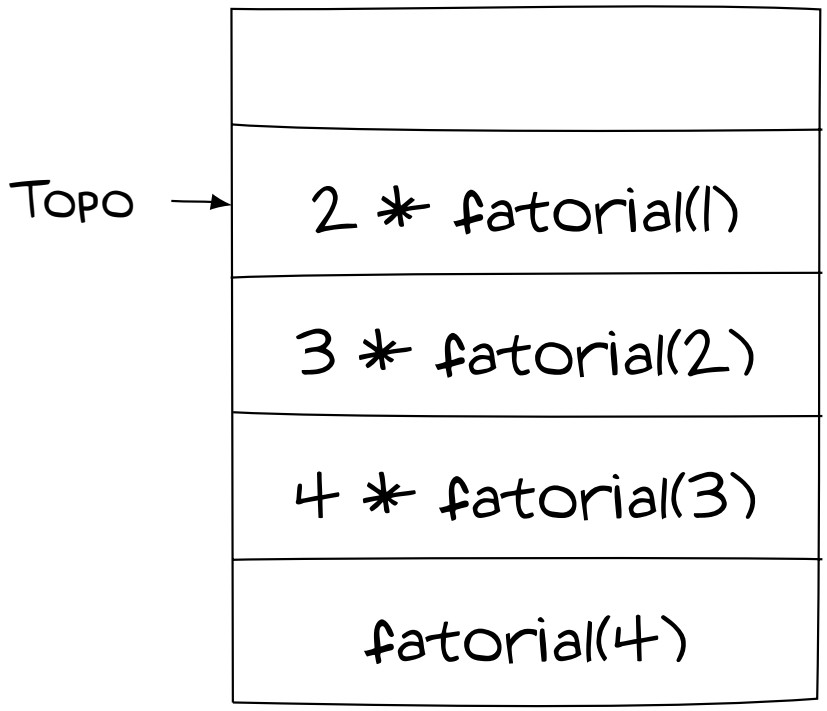
Aplicações

- Suporte a recursão
  - Função recursiva **uint64_t** fatorial(**uint32_t** n)



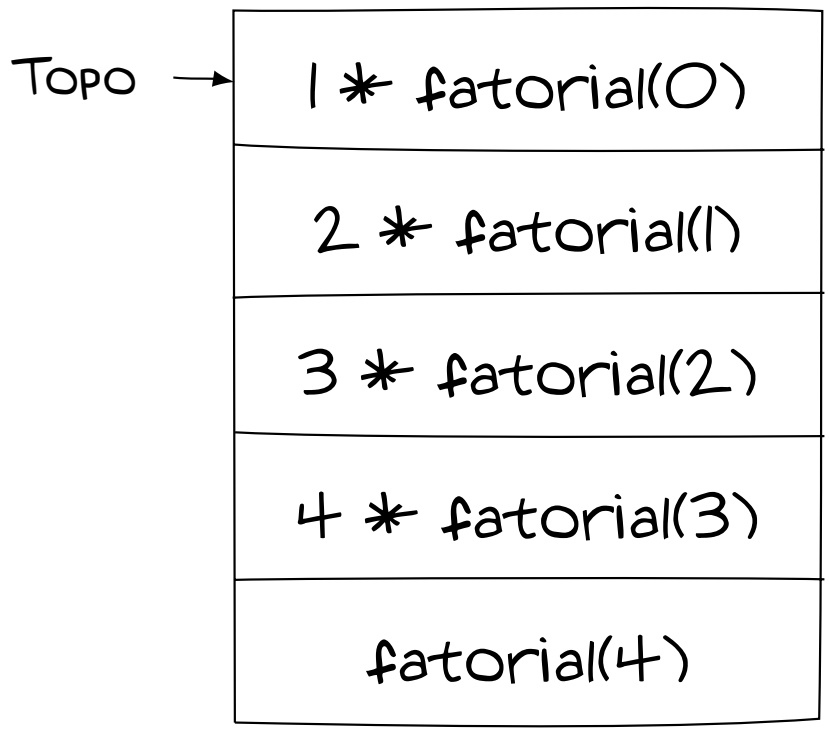
Aplicações

- Suporte a recursão
  - Função recursiva **uint64_t** fatorial(**uint32_t** n)



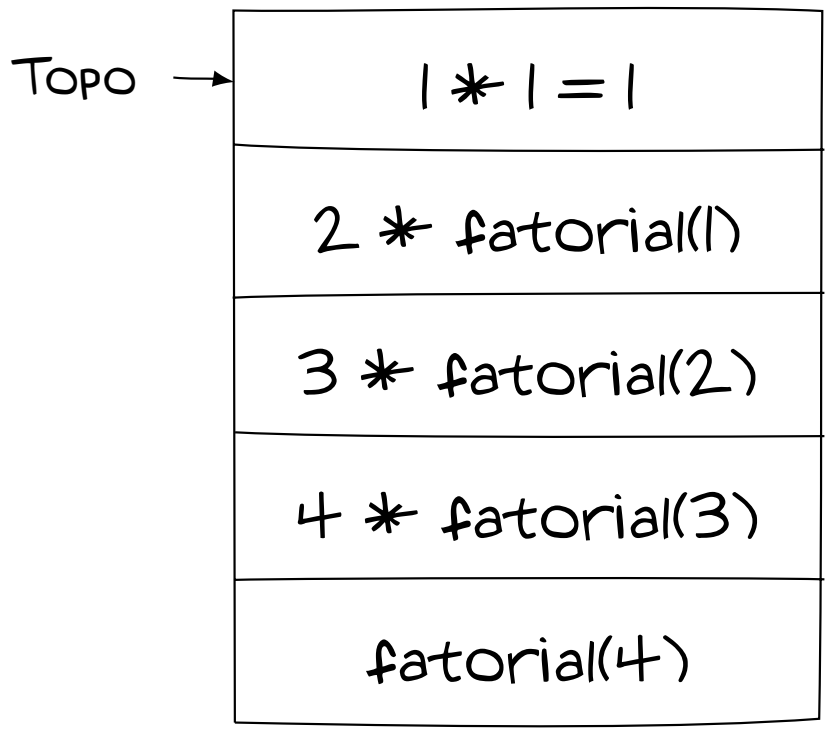
Aplicações

- Suporte a recursão
  - Função recursiva **uint64_t** fatorial(**uint32_t** n)



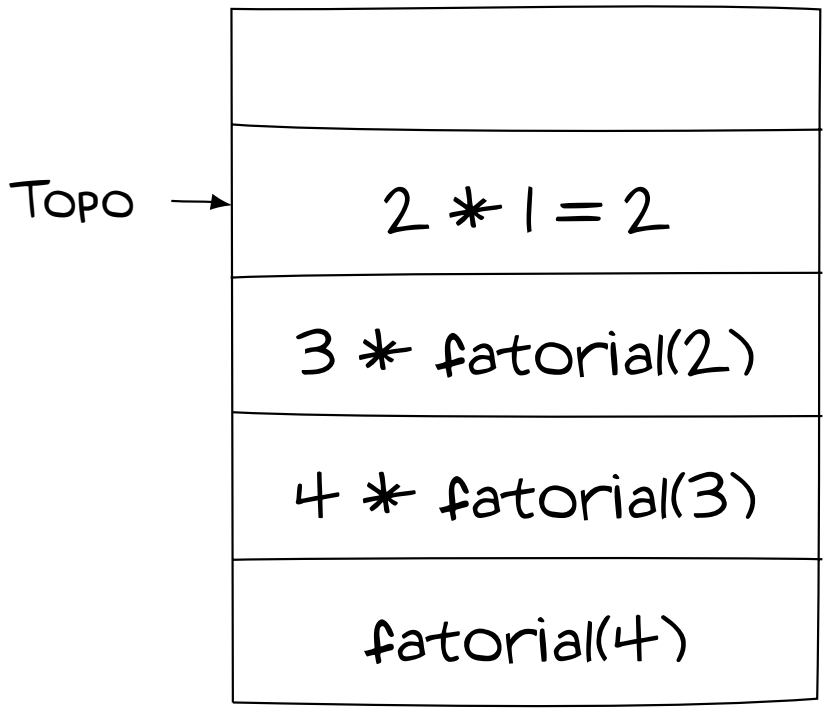
Aplicações

- Suporte a recursão
  - Função recursiva **uint64_t** fatorial(**uint32_t** n)



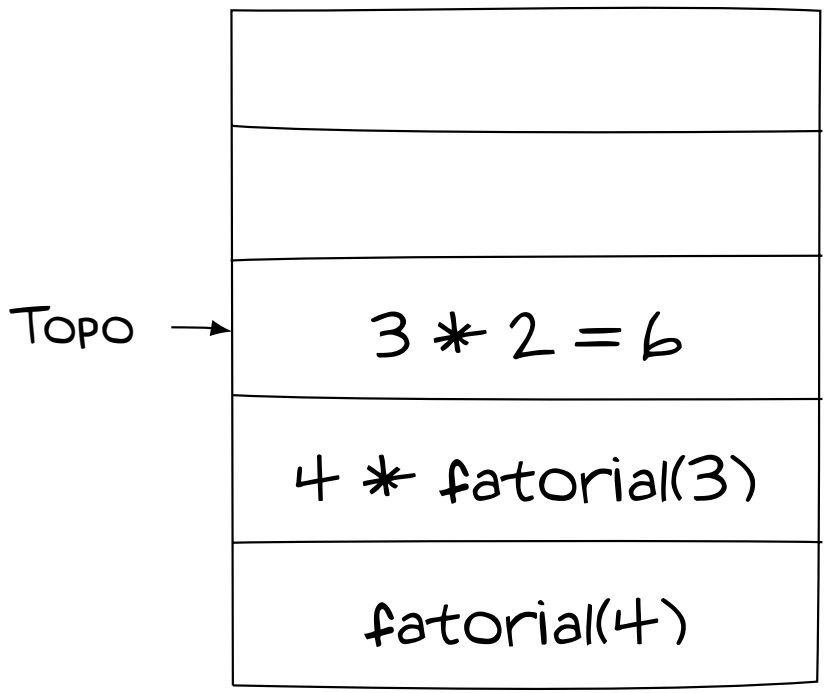
Aplicações

- Suporte a recursão
  - Função recursiva **uint64_t** fatorial(**uint32_t** n)



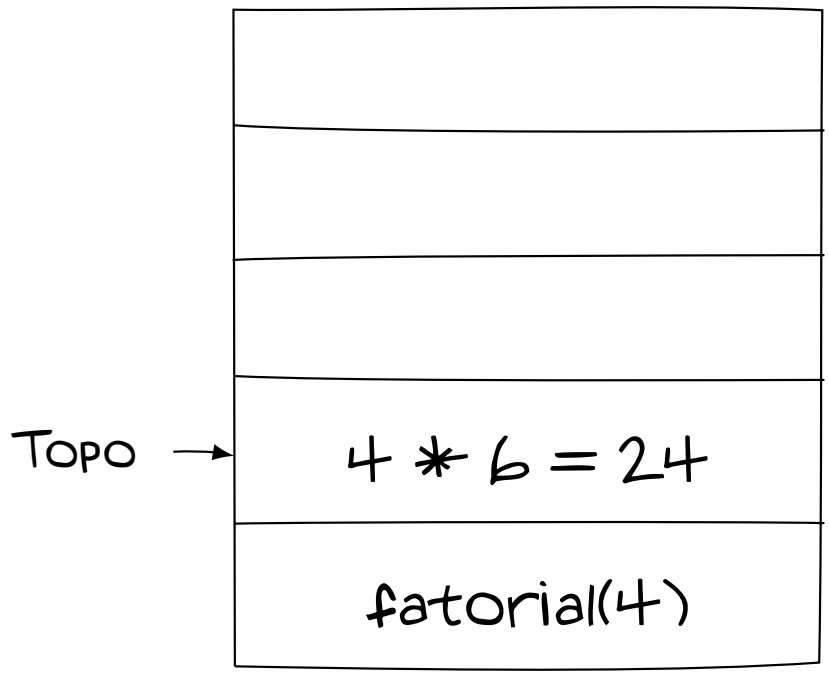
Aplicações

- Suporte a recursão
  - Função recursiva **uint64_t** fatorial(**uint32_t** n)



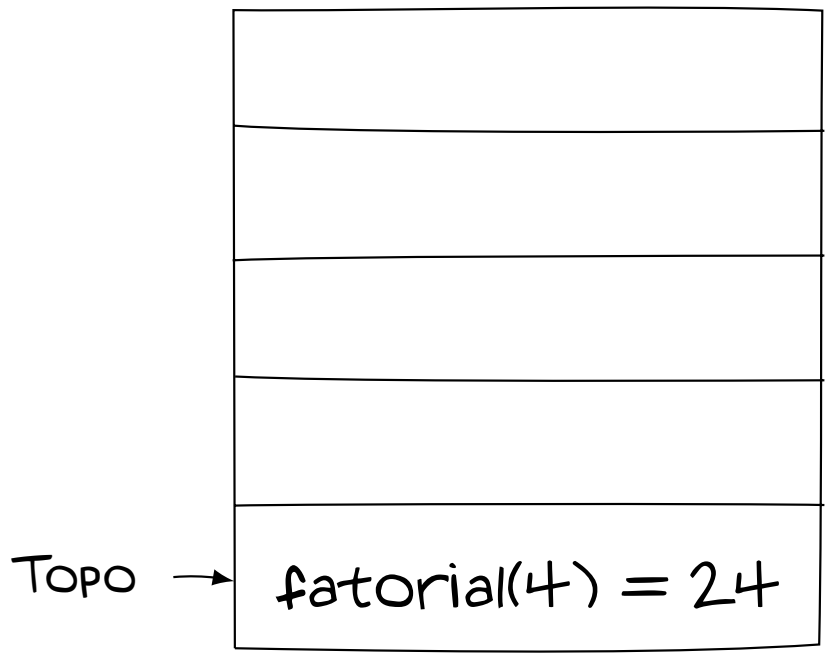
Aplicações

- Suporte a recursão
  - Função recursiva **uint64_t** fatorial(**uint32_t** n)



Aplicações

- Suporte a recursão
  - Função recursiva **uint64_t** fatorial(**uint32_t** 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*]

```
1  2
2  jatodetinta
3  laser
4  6
5  sigaa␣2
6  bbbbb␣7
7  documento␣3
8  abc␣2
9  xyz␣5
10 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

```
1  [jatodetinta] sigaa-2p
2  [laser] bbbbb-7p
3  [jatodetinta] documento-3p, sigaa-2p
4  [jatodetinta] abc-2p, documento-3p, sigaa-2p
5  [jatodetinta] xyz-5p, abc-2p, documento-3p, sigaa-2p
6  [laser] aaaaa-6p, bbbbb-7p
7  25p
8  aaaaa-6p
9  xyz-5p
10 bbbbb-7p
11 abc-2p
12 documento-3p
13 sigaa-2p
```