Pilha

Função- como controlar os retornos?

Início
1. imprima "x"
2. call A
3. fim

Inicio_SubA
4. call B
5. call C
6. fim_sub

Inicio_SubB
7. imprimia "y"
8. call C
9. fim_sub

Inicio_SubC
10. imprima "z"
11. leia "w"
12. fim_sub

Exemplos

- pilha de pratos
- pilha de livros
- pilha de pratos do bandejão

Problema

Função principal chama função A, que chama a função B, que chama a função C. Qual o próximo comando a ser executado quando C termina?

Solução

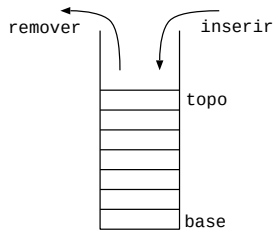
A cada chamada de sub-rotina, armazenar o endereço de retorno. Mas guardar em qual estrutura de dados?
⇒ PILHA

Operações com pilha ($\mathcal{O}(1)$?)

- `cria_pilha(s)`
- `pilha_vazia(s)`
- `pilha_cheia(s)`
- `empilha(s, x)` ou `push(s, x)`
- `desempilha(s)` ou `pop(s)`
- `elem_topo(s)`

Pilha (LIFO - Last In, First Out)

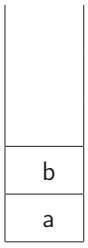
Lista linear em que inserção, eliminação e acesso de elementos só ocorrem em uma das extremidades (TOPO).



- `cria_pilha(s)` – cria uma pilha vazia
- `pilha_vazia(s)` – retorna `V` se pilha vazia
- `pilha_cheia(s)` – retorna `V` se pilha cheia
- `empilha(s, x)` ou `push(s, x)` – insere x no topo da pilha
- `desempilha(s)` ou `pop(s)` – remove o elemento da pilha retornando-o como valor da função
- `elem_topo(s)` – acessa o elemento do topo da pilha sem removê-lo

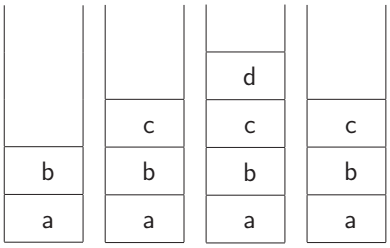
Operações `push` e `pop`

- pilha `s` com 2 elementos



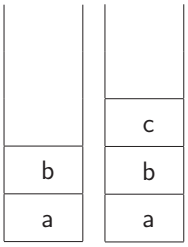
Operações `push` e `pop`

- `pop(s)`



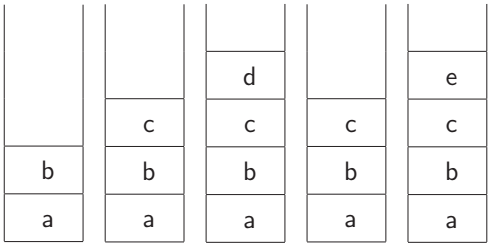
Operações `push` e `pop`

- `push(s, 'c')`



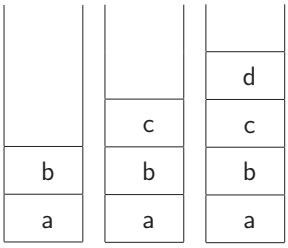
Operações `push` e `pop`

- `push(s, 'e')`



Operações `push` e `pop`

- `push(s, 'd')`



Operações

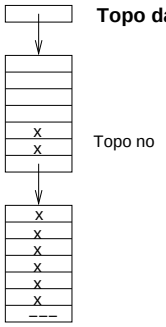
- Como identificar o segundo elemento a partir do topo da pilha mantendo a pilha inalterada?
- Como identificar o elemento da base da pilha?
- Como identificar o elemento da base da pilha mantendo a pilha inalterada?
- Como inverter os elementos de uma pilha?

Funções

Como resolver o problema de chamada e retorno de funções com o uso de pilha?

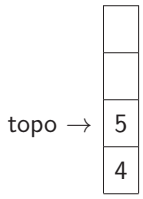
Implementação de Pilha

- mista – utiliza listas encadeadas e vetores



Implementação de Pilha

- estática ou sequencial – utiliza vetores



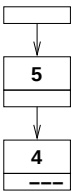
Pilha estática x dinâmica

- Problema da implementação de listas encadeadas: necessidade de movimentações de itens em inserções e remoções.

  – não ocorre em pilhas!
- Alocação estática vantajosa na maioria das situações.
- Alocação dinâmica interessante para pilhas cujo tamanho não pode ser antecipado, ou é muito variável.

Implementação de Pilha

- dinânica ou encadeada – utiliza listas encadeadas



Aplicações com pilha

- Inversa – saída deve ser a entrada na ordem inversa 12345 $\Rightarrow$ 54321

```
cria_pilha(s)
leia num
enquanto pilha_cheia(s) != V faça
  push(s, num)
  leia num
fim enquanto
enquanto pilha_vazia(s) != V faça
  x <- pop(s)
  imprima x
fim enquanto
```

Notação polonesa

prefixa	infixa	posfixa
+AB	A+B	AB+

- notação tradicional é ambígua – obriga o pré-estabelecimento de regras de prioridade
- parênteses alteram a ordem de precedência
- conversão e avaliação de expressões:
  - melhor compreensão da utilidade de pilha
  - algoritmos concisos e robustos para cálculos envolvendo expressões matemáticas

Os prefixos "pre", "pos" e "in" referem-se à posição relativa do operador em relação aos dois operandos. Na notação prefixa (ou notação polonesa), o operador precede os dois operandos. Na notação posfixa (ou polonesa reversa), o operador aparece após os operandos.

Algoritmo: Infixa $\Rightarrow$ Posfixa

Percorrer a nova expressão infixa da esq p/ dir e para cada símbolo encontrado:

1. se operando, copiá-lo para a expressão posfixa (saída)
2. se operador α, enquanto a pilha não estiver vazia e houver operador no seu topo com prioridade maior ou igual a α, desempilha e copia-o para saída. Empilha α.
3. se '(', empilha-o
4. se ')', desempilha e copia-o para saída até encontar um '('

No final, a pilha deve ficar vazia.

Esta aplicação ilustra os diferentes tipos de pilhas e as diversas operações e funções definidas a partir delas. O exemplo é, em si mesmo, um relevante tópico de ciência da computação.

Exercícios

1. $A - B * C$
2. $A * (B - C)$
3. $A - B + C$
4. $(A - B)/(C + D) * E$
5. $A \wedge B * C - D + E/F/(G - H)$
6. $((A + B) * C - (D - E)) \wedge (F - G)$
7. $A + B/(C * D \wedge E)$

	Prefixa	Posfixa
1	-A*BC	*A-BC
2	*A-BC	ABC-*
3	+-ABC	AB-C+
4	*/-AB+CDE	AB-CD+/E*
5	+-*∧ABCD//EF-GH	AB∧C*D-EF/GH-/+
6	∧-*+ABC-DE-FG	AB+C*DE–FG-∧
7	+A/B*C∧DE	ABCDE∧*/+

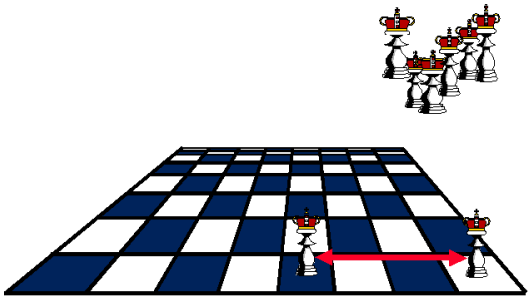
N-Rainhas

8 rainhas e um tabuleiro de xadrez

Como colocar as rainhas no tabuleiro sem que uma possa atacar a outra?

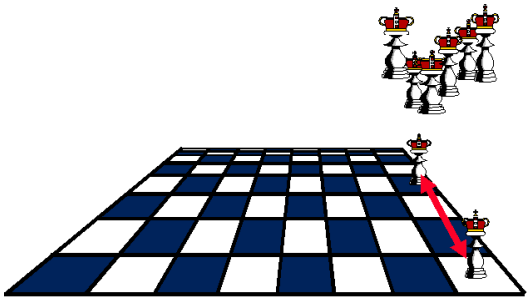
N-Rainhas

2 rainhas não podem estar na mesma linha



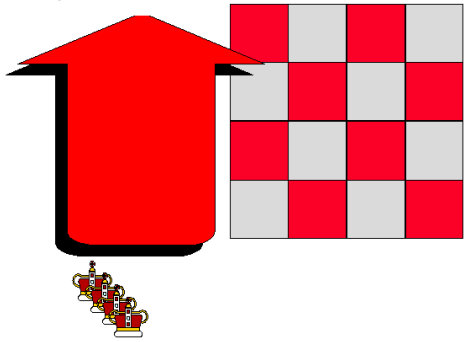
N-Rainhas

2 rainhas não podem estar na mesma coluna



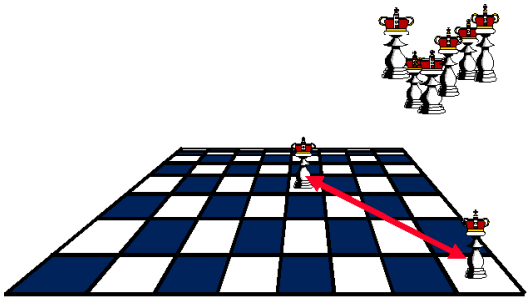
N-Rainhas

Pilha para marcar onde as rainhas são colocadas



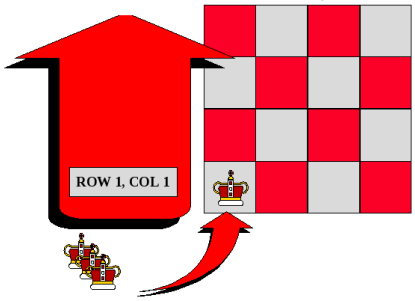
N-Rainhas

2 rainhas não podem estar na mesma diagonal



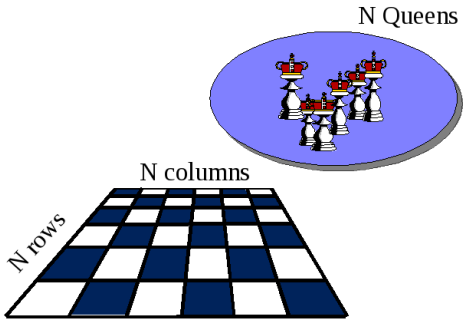
N-Rainhas

Ao colocar uma rainha no tabuleiro, a posição da nova rainha é armazenado na pilha.



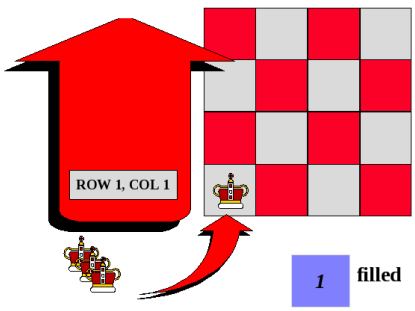
N-Rainhas

O número de rainhas pode variar



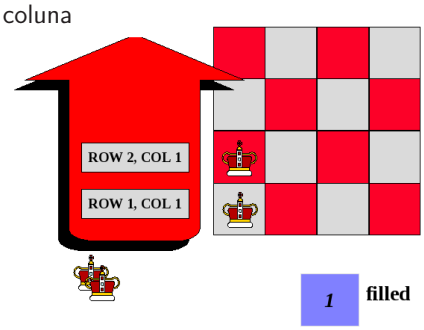
N-Rainhas

Inteiro para guardar num rainhas no tabuleiro



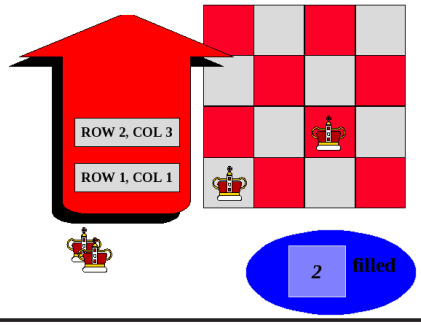
N-Rainhas

Nova rainha é inicialmente colocada na primeira coluna



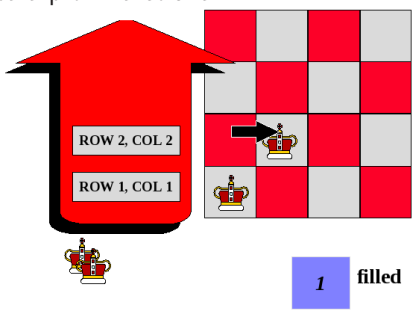
N-Rainhas

Sem conflitos



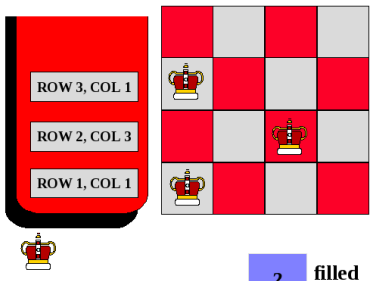
N-Rainhas

Se há conflito com outras rainhas, então desloca para próxima coluna



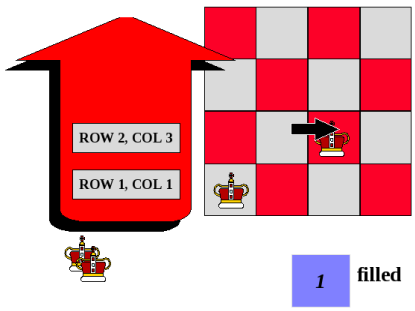
N-Rainhas

Nova rainha



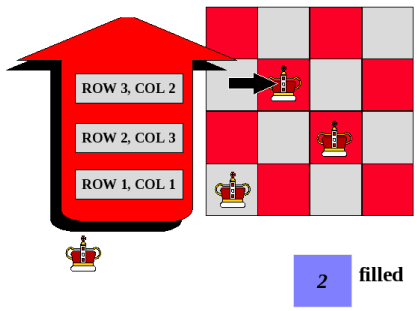
N-Rainhas

Outro conflito ⇒ novo deslocamento



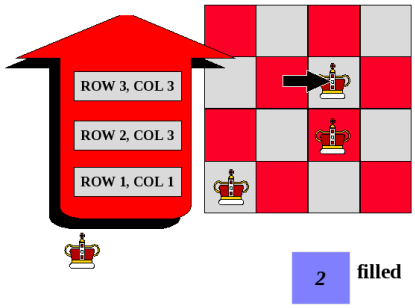
N-Rainhas

Conflito



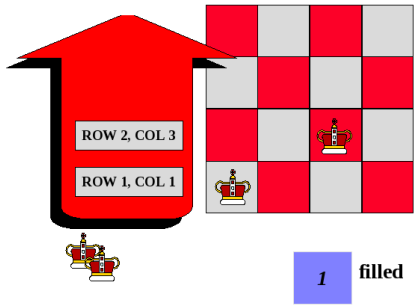
N-Rainhas

Conflito



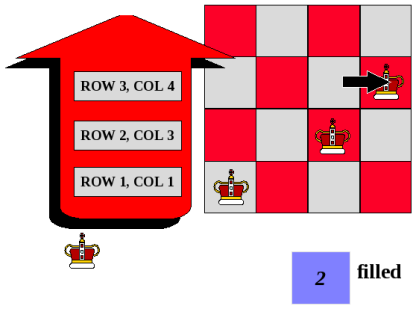
N-Rainhas

Remove da pilha e decrementa o inteiro



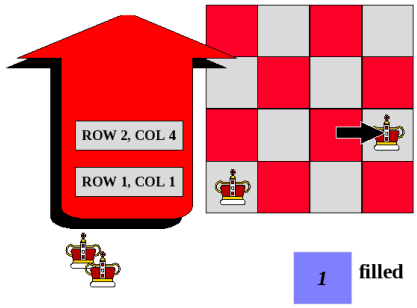
N-Rainhas

Conflito



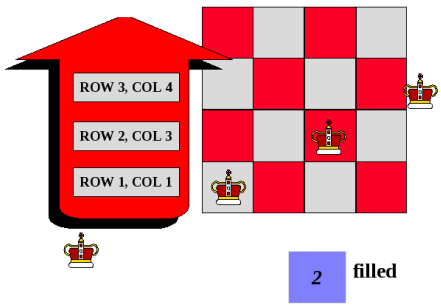
N-Rainhas

Desloca a rainha



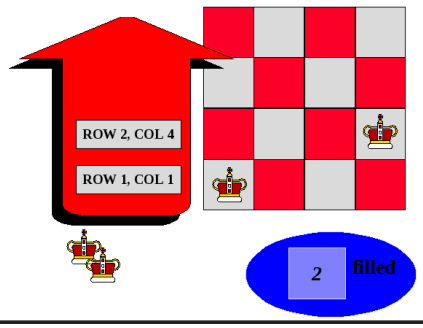
N-Rainhas

Não há onde colocar?!?



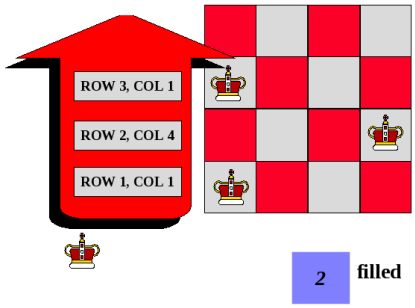
N-Rainhas

Sem conflitos



N-Rainhas

Insere novamente na terceira coluna



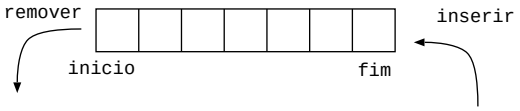
Exercícios

Utilize pilha na implementação de:

- conversão de decimal para binário
- verificação de parênteses balanceados
- conversão da notação infixa para posfixa
- avaliação da expressão na forma posfixa
- edição de texto: suponha "#" o caractere correspondente a operação apagar. Então a string "abc#d##e" é na verdade a string "ae".

Fila

- conjunto ordenado de itens no qual todas as inserções são realizadas em um extremo (fim, rear ou tail) e todas as remoções e acessos são realizados no outro extremo (início, front ou head) da lista
- FIFO (First In, First Out)



Em fila, existe uma ordem linear que é a ordem de chegada. Filas são utilizadas quando desejamos processar itens de acordo com a ordem "primeiro-que-chega, primeiro-atendido".

Exemplo

Fila de metrô. As pessoas vão chegando...

Exemplo

Fila de metrô. As pessoas vão chegando...

Exemplo

Fila de metrô. As pessoas vão chegando...

Exemplo

Fila de metrô. As pessoas vão chegando...

Exemplo

Fila de metrô. As pessoas vão chegando...

Exemplo

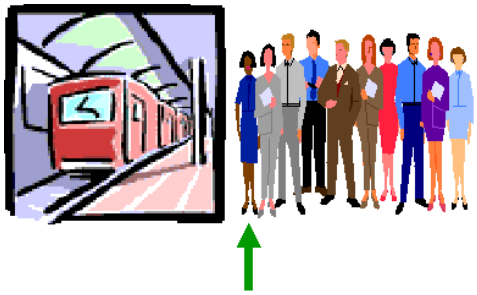
Inserção sempre ocorre no final (`insere_fila`)

Exemplo

Fila de metrô. As pessoas vão chegando...

Exemplo

Remoção sempre no início (`remove_fila`)

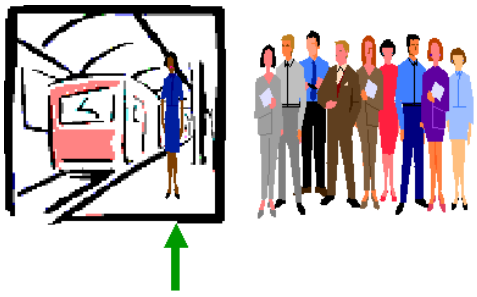


Exemplo

Inserção sempre ocorre no final (`insere_fila`)

Exemplo

Remoção sempre no início (`remove_fila`)

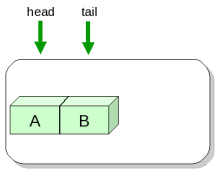


Exemplo

Remoção sempre no início (`remove_fila`)

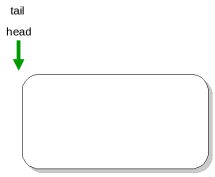
Outro exemplo

- inicialmente vazia
- inserir caixa A
- inserir caixa B



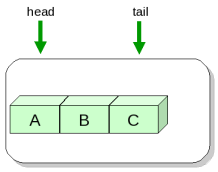
Outro exemplo

- inicialmente vazia



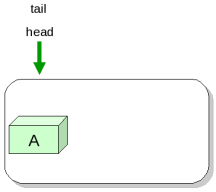
Outro exemplo

- inicialmente vazia
- inserir caixa A
- inserir caixa B
- inserir caixa C



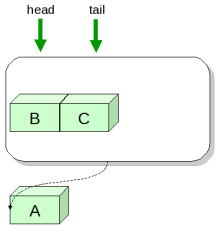
Outro exemplo

- inicialmente vazia
- inserir caixa A



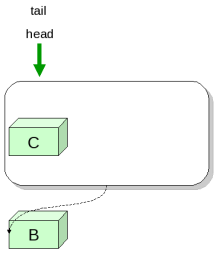
Outro exemplo

- inicialmente vazia
- inserir caixa A
- inserir caixa B
- inserir caixa C
- remover caixa



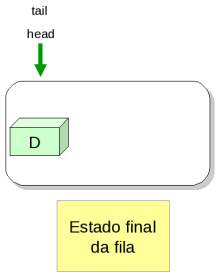
Outro exemplo

- inicialmente vazia
- inserir caixa A
- inserir caixa B
- inserir caixa C
- remover caixa
- remover caixa



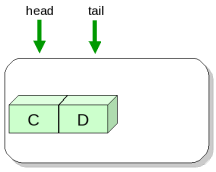
Outro exemplo

- inicialmente vazia
- inserir caixa A
- inserir caixa B
- inserir caixa C
- remover caixa
- remover caixa
- inserir caixa D
- remover caixa



Outro exemplo

- inicialmente vazia
- inserir caixa A
- inserir caixa B
- inserir caixa C
- remover caixa
- remover caixa
- inserir caixa D

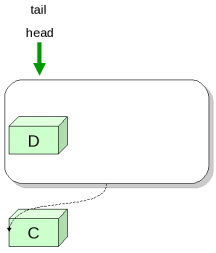


Operações com Fila ($\mathcal{O}(1)$?)

- `cria_fila(q)`
- `fila_vazia(q)`
- `fila_cheia(q)`
- `insere_fila(q, x)`
- `remove_fila(q)`
- `libera_fila(q)`

Outro exemplo

- inicialmente vazia
- inserir caixa A
- inserir caixa B
- inserir caixa C
- remover caixa
- remover caixa
- inserir caixa D
- remover caixa

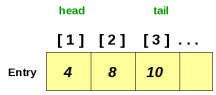


Implementação

- estática – vetor
  - primeiro elemento sempre na primeira posição do vetor – lento
  - dois índices (ini e fim) que sempre crescem.
- dinâmica – lista ligada

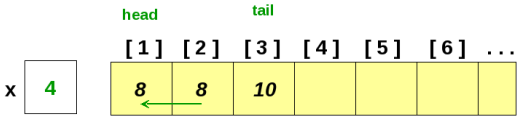
Vetor

Primeira elemento na primeira posição:



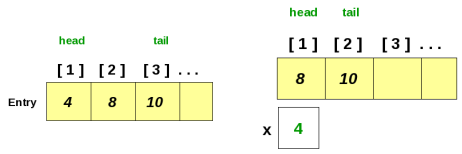
Vetor

Movimentos no vetor:



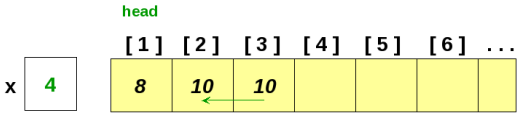
Vetor

Primeira elemento na primeira posição:



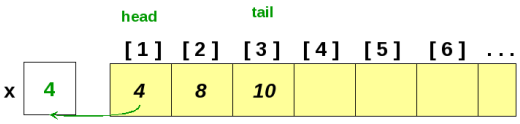
Vetor

Movimentos no vetor:



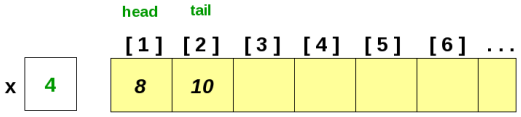
Vetor

Movimentos no vetor:



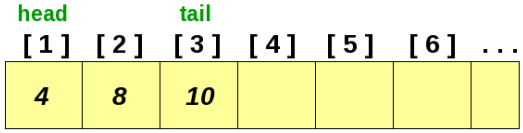
Vetor

Movimentos no vetor:

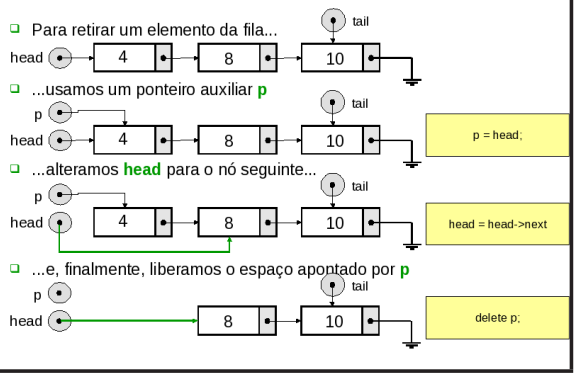


Vetor

Com dois índices: head e tail

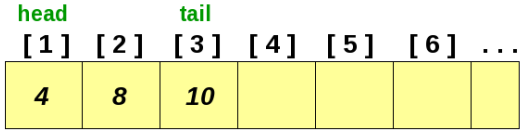


Lista Encadeada - Remoção

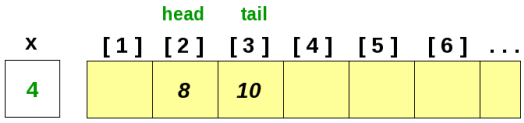


Vetor

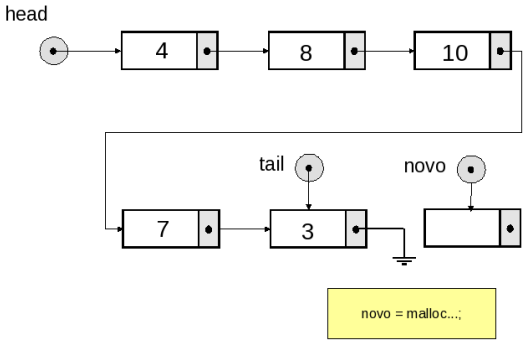
Com dois índices: head e tail



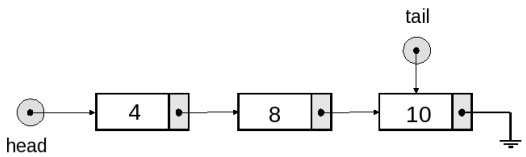
Após remoção:



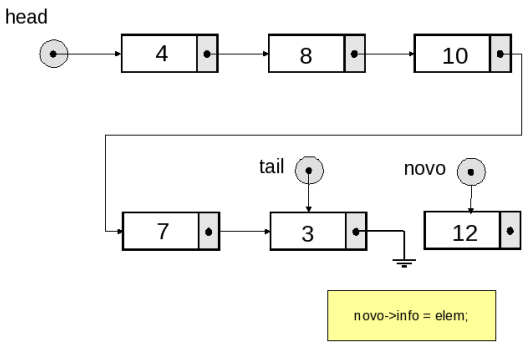
Lista Encadeada - Inserção



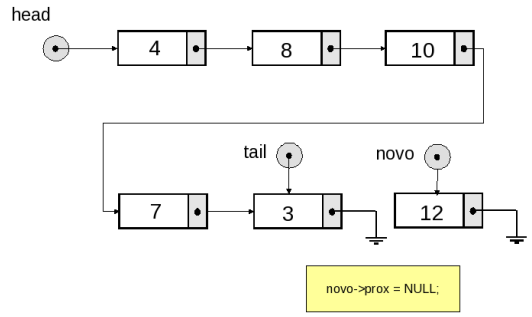
Lista Encadeada



Lista Encadeada - Inserção

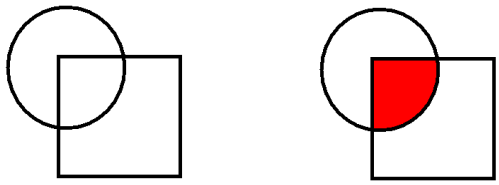


Lista Encadeada - Inserção

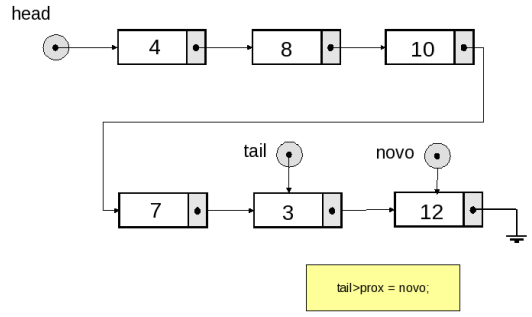


Aplicações

- Coloração de Regiões

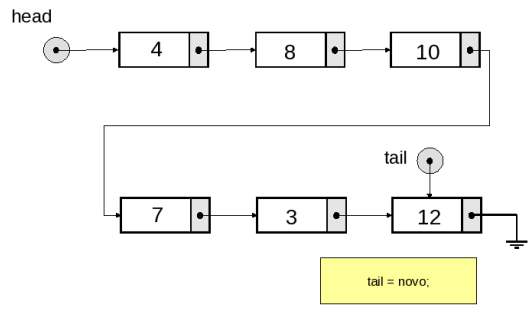


Lista Encadeada - Inserção



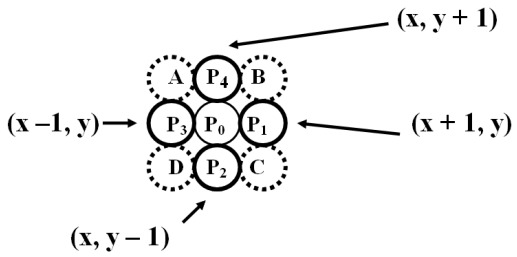
- Algoritmos para colorir regiões de desenhos representados sob a forma de matrizes de pontos
- Região de um desenho: conjunto de pontos conectados entre si e que têm a mesma cor
- Dois pontos Pi e Pj estão conectados entre si se, e somente se, partindo de Pi, ao incrementar (ou decrementar) sua abscissa (ou ordenada), chega-se ao ponto Pj

Lista Encadeada - Inserção



Coloração de Regiões

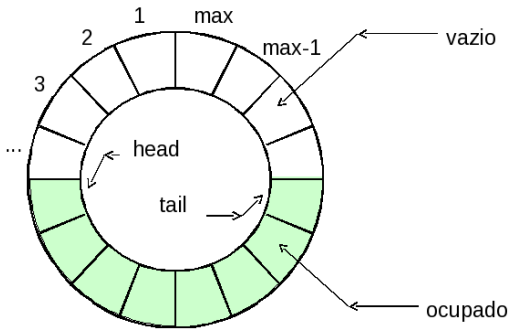
- Pontos conectados a P_0



Coloração de Regiões

1. determinar um ponto inicial P0 de cor C0 pertencente à região R
2. determinar uma nova cor C1 para a região R
3. inserir P0 numa fila q, inicialmente vazia
4. enquanto a fila q não esvaziar
   (a) remover um ponto P da fila q
   (b) inserir em q todos os pontos conectados a P, cuja cor seja C0
   (c) alterar a cor de P para C1

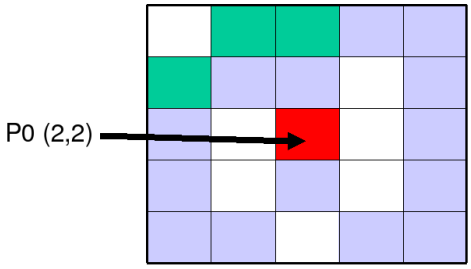
Fila Circular



Armazena os elementos na fila como se fosse um círculo (primeiro elemento do vetor vem logo depois do seu último). Assim, um novo elemento não será incluído em uma fila circular apenas se realmente não houver espaço suficiente nessa fila.

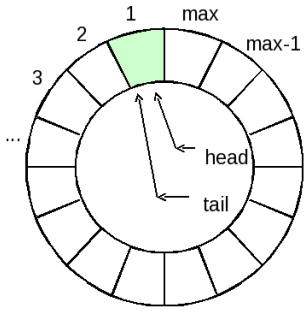
Coloração de Regiões

- P_0 (2,2) inicialmente branco
- nova cor – vermelho



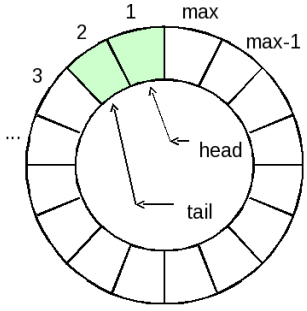
Fila Circular

Inserindo o primeiro elemento:



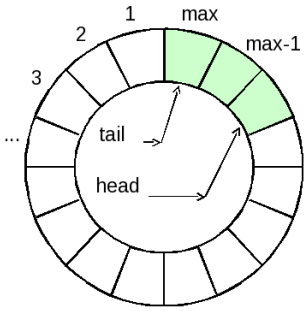
Fila Circular

Inserindo o segundo elemento:

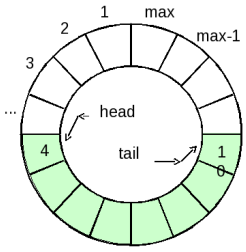


Fila Circular

Após várias inserções e remoções:



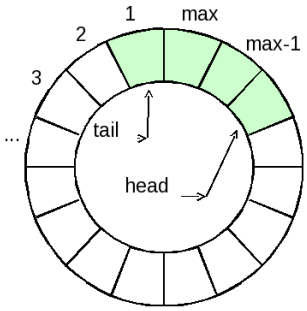
"Desenrolando" o vetor



[1]	[2]	...	head	...	tail	...	[max-1]	[max]
			4		*10*			

Fila Circular

Inserindo mais um elemento:



Fila Circular

- `ini = 2, fim = 4`

4	50
3	40
2	30
1	
0	

Detalhe de Implementação

```
if (i == max)
  i = 1; // ou 0 em C
else
 i ++;
```

Melhor

```
i = (i % max) + 1; // ou sem somar 1 em C
```

Fila Circular

- `ini = 2, fim = 0`

4	50	4	50
3	40	3	40
2	30	2	30
1		1	
0		0	60

Fila Circular

- `ini = 2, fim = 1`

4	50	4	50	4	50
3	40	3	40	3	40
2	30	2	30	2	30
1		1		1	70
0		0	60	0	60

Fila Circular

- `ini = 2, fim = 1` ?!?!?

4	50	4	50	4	50	4	
3	40	3	40	3	40	3	
2	30	2	30	2	30	2	
1		1		1	70	1	
0		0	60	0	60	0	

Implementação

- sacrificar uma posição do vetor
- criar uma variável `tam` que indica o número de elementos na fila
- preencher a posição vazia com um caracter especial (por exemplo, ε)
- verificar a última operação (inserção ou remoção)

Fila com Prioridades

- ordem intrínseca dos elementos determina os resultados das suas operações básicas
- tipos:
  1. ascendente – somente o MENOR elemento pode ser removido
  2. descendente – somente o MAIOR elemento pode ser removido
- pilha – ordenados pelo instante no tempo representando a ordem de inserção

Uma fila com prioridades é uma estrutura de dados na qual a ordem intrínseca dos elementos determina os resultados das suas operações básicas (em particular, a operação Remove).

1) Fila com prioridades ascendente: é uma coleção de elementos na qual novos elementos podem ser inseridos normalmente (como em filas sem prioridade) e da qual somente o MENOR elemento pode ser removido.

2) Fila com prioridade descendente: é uma coleção de elementos na qual novos elementos podem ser inseridos normalmente (como em filas sem prioridade) e da qual somente o MAIOR elemento pode ser removido.

Elementos com mesmo valor (ou mesma prioridade) são retirados na ordem em que foram inseridos, seguindo a regra para filas "normais".

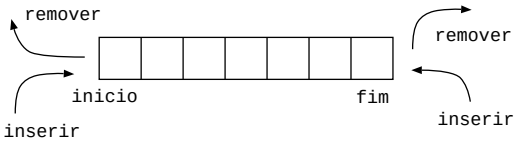
Aplicação: transmissão assíncrona com prioridades (cancelamento de job de impressão, ...).

Normalmente, os elementos de uma fila com prioridades são compostos de dois campos: o dado propriamente dito + prioridade do dado.

Uma pilha pode ser vista como sendo uma fila com prioridades descendente cujos elementos estão ordenados pelo instante no tempo representando a ordem de inserção.

Fila Dupla (Deque)

- inserção, remoção e acesso podem ser realizados nos dois extremos
- exemplo: estacionamento de trens



A estrutura de dados *deque* provê meios para inserir e remover itens de ambas as extremidades. A palavra *deque* vem do inglês *double-ended queue*.

Exercício

Considere a implementação de filas usando arranjos "circulares". Escreva uma função `FuraFila(Fila* pFila, elem_t x)` que insere um item na primeira posição da fila. O detalhe é que seu procedimento deve ser $O(1)$, ou seja, não pode movimentar os outros itens da fila. Observe que este neste caso, estaremos desrepeitando o conceito de FILA – primeiro a entrar é o primeiro a sair.

Exercício

Existem partes de sistemas operacionais que cuidam da ordem em que os programas devem ser executados. Por exemplo, em um sistema de computação de tempo compartilhado ("time-shared") existe a necessidade de manter um conjunto de processos em uma fila, esperando para serem executados. Escreva um programa que seja capaz de ler uma série de solicitações para:

- Incluir novos processos na fila de processo;
- Retirar da fila o processo com o maior tempo de espera;
- Imprimir o conteúdo da lista de processo em determinado momento.

Assuma que cada processo é representado por um registro composto por um número identificador do processo.

Bibliografia

- Michael Main and Walter Savitch, *Data Structures and Other Objects Using C++*, 2. edição, Addison Wesley, 2004.
- José Augusto Baranauskas, notas de aula da disciplina *Algoritmos e Estruturas de Dados I*, Departamento de Física, FFCLRP-USP, 2007.