Aula 12 - Aplicações com Pilha

Aplicações com pilha

- Inversa – saída deve ser a entrada na ordem inversa 12345 ⇒ 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
```

Algoritmo: Infixa ⇒ 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.

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.

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	ABC*-	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∧*/+

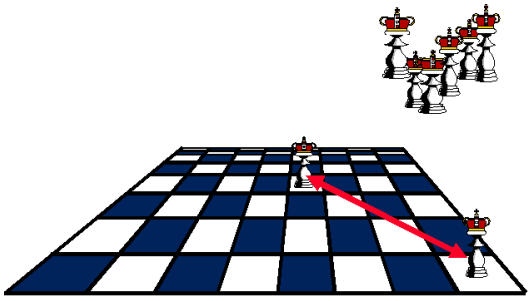
N-Rainhas

8 rainhas e um tabuleiro de xadrez

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

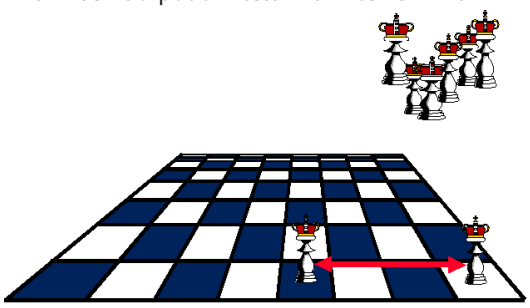
N-Rainhas

2 rainhas não podem estar na mesma diagonal



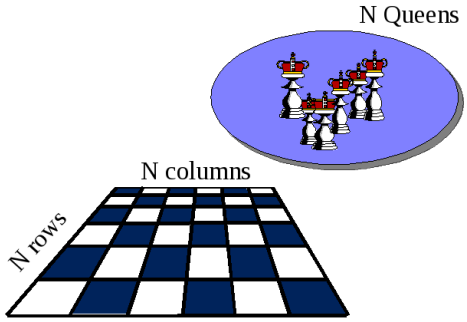
N-Rainhas

2 rainhas não podem estar na mesma linha



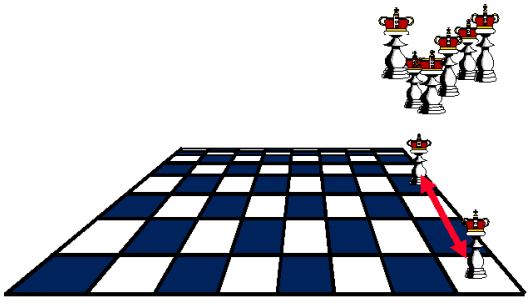
N-Rainhas

O número de rainhas pode variar



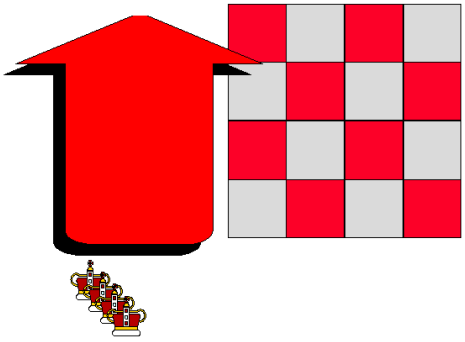
N-Rainhas

2 rainhas não podem estar na mesma coluna



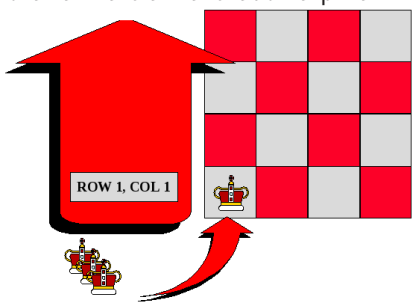
N-Rainhas

Pilha para marcar onde as rainhas são colocadas



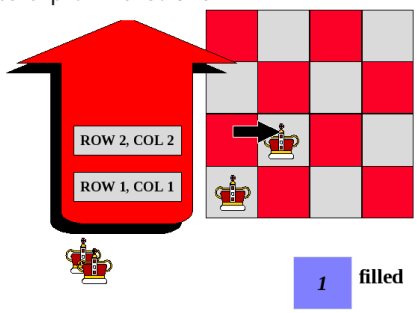
N-Rainhas

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



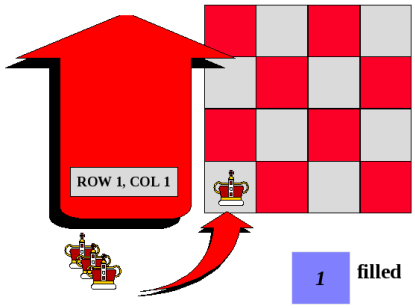
N-Rainhas

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



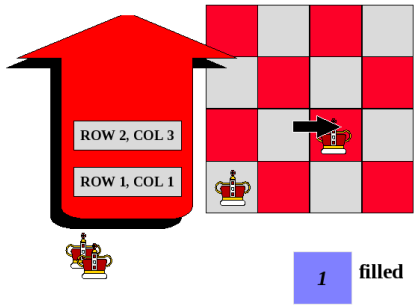
N-Rainhas

Inteiro para guardar num rainhas no tabuleiro



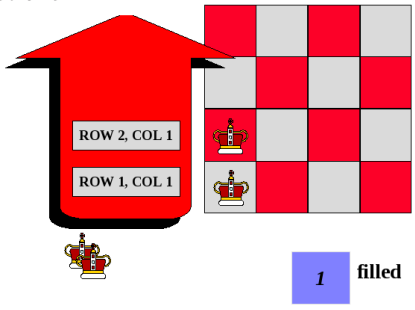
N-Rainhas

Outro conflito ⇒ novo deslocamento



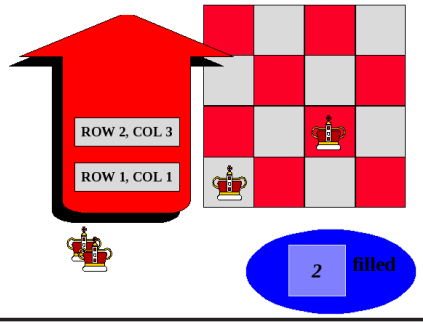
N-Rainhas

Nova rainha é inicialmente colocada na primeira coluna



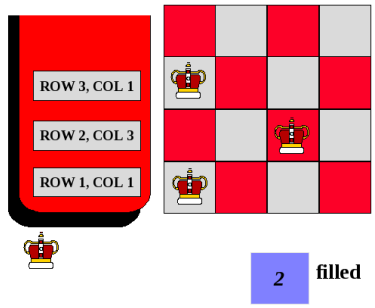
N-Rainhas

Sem conflitos



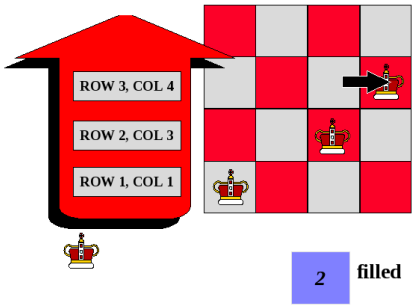
N-Rainhas

Nova rainha



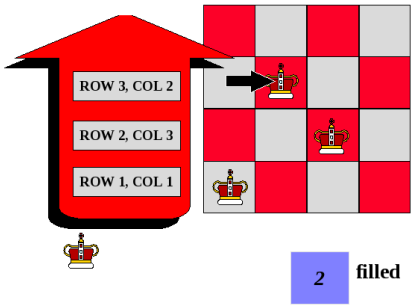
N-Rainhas

Conflito



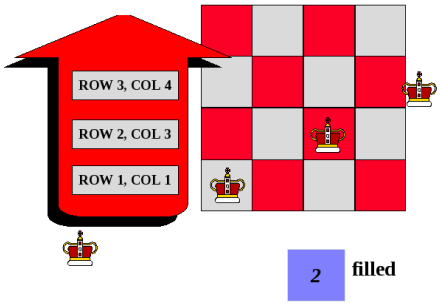
N-Rainhas

Conflito



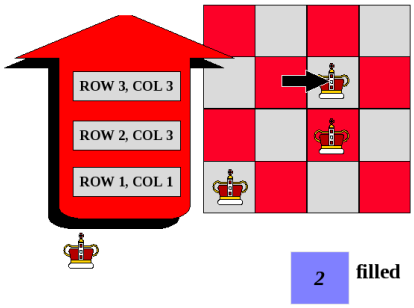
N-Rainhas

Não há onde colocar?!?



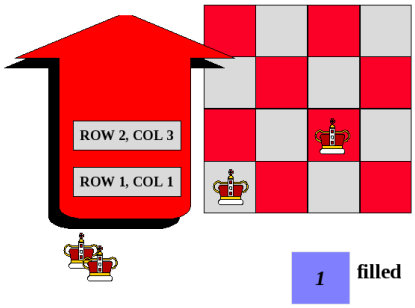
N-Rainhas

Conflito



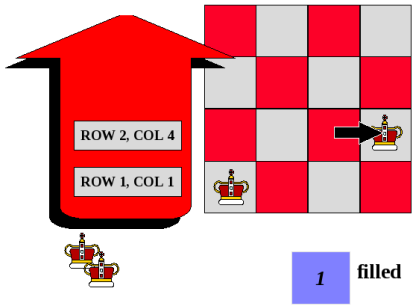
N-Rainhas

Remove da pilha e decrementa o inteiro



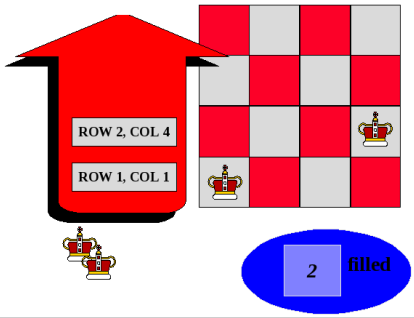
N-Rainhas

Desloca a rainha



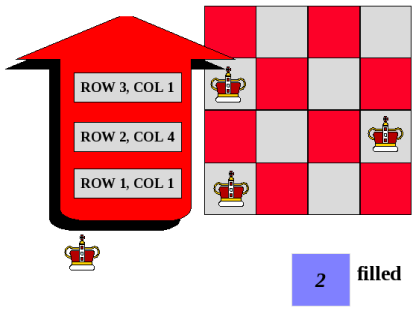
N-Rainhas

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".

Bibliografia

- Michael Main and Walter Savitch, *Data Structures and Other Objects Using C++*, 2. edição, Addison Wesley, 2004.