

🏠 Home (/) » SCC0222 (/offerings/view/1601) » [5 - Alocação Dinâmica] Notação Polonesa Reversa

## [5 - Alocação Dinâmica] Notação Polonesa Reversa

**Disciplina:** SCC0222 - Laboratório de Introdução à Ciência da Computação I

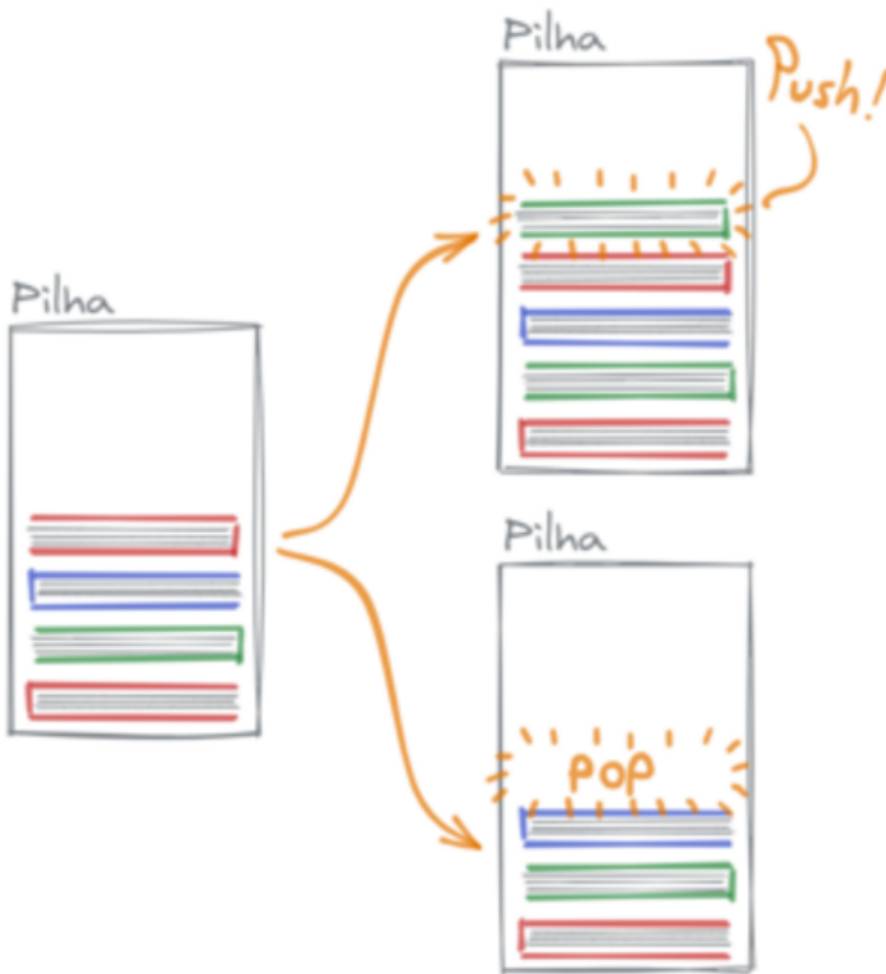
**Prazo de Entrega:** 23/07/2021 23:59:59 Fechado

### Contexto

A Notação Polonesa Inversa ([https://pt.wikipedia.org/wiki/Nota%C3%A7%C3%A3o\\_polonesa\\_inversa](https://pt.wikipedia.org/wiki/Nota%C3%A7%C3%A3o_polonesa_inversa)) (ou RPN na sigla de Reverse Polish Notation) é uma notação onde os operadores ficam em posição pós-fixada, ou seja, após o operando. A notação mais comum, é a infixa, nela temos exemplos como  $2 + 2$  onde o operador  $+$  está entre os operandos. Na posição pós-fixada o operador vem depois e assim, o exemplo anterior ficaria  $2\ 2\ +$ . É um pouco estranho à primeira vista, mas as vantagens dessa notação estranha é que ela dispensa a necessidade de parênteses! Na notação infixa temos toda essa história de precedência dos operadores e de que devemos sempre calcular os parênteses primeiros, depois as multiplicações e divisões e só por último calculamos as somas e subtrações. Não na Notação Polonesa Inversa. Nela o operador é simplesmente aplicado nos valores mais a esquerda calculando da esquerda para a direita. Podemos ver um exemplo disso em  $(2 + 3) * 5$ , onde os parênteses claramente importam. Mas em RPN temos  $5\ 2\ 3\ +\ *$ , para calcular, vamos lendo da esquerda para a direita até encontrar um operador. Quando encontramos, somamos os dois valores à esquerda e substituímos pelo valor resultante. No exemplo anterior a primeira operação é a  $+$ , então fazemos  $2\ 3\ + = 5$  e substituímos, ficando  $5\ 5\ *$ , agora podemos multiplicar e obter o resultado final,  $25$ . Sem os parênteses teríamos em notação infixa  $2 + 3 * 5 = 17$  e em RPN  $2\ 3\ 5\ * + = 17$ .

### A pilha (stack)

A pilha é uma estrutura de dados muito útil para a computação, com ela, é possível fazer muitas coisas diferentes usando uma estrutura relativamente simples. Estruturas de dados servem para armazenar uma coleção de dados de uma forma específica e possuem algumas operações bem definidas. Em particular, a pilha funciona como uma pilha de livros: você pode adicionar livros em cima da pilha e remover de cima dela, mas não de baixo ou de qualquer outro lugar. Existem duas operações fundamentais em pilhas, o `push` (empilha), que adiciona um valor no topo da pilha, e o `pop` (desempilha) que tira o valor do topo da pilha para que ele possa ser usado para algo.



Para implementar uma pilha, precisamos ter 2 coisas a todo momento:

1. Um ponteiro que aponta para o vetor alocado dinamicamente que armazena os valores da pilha.
2. A posição considerada como "topo" da pilha.

Nesse problema, implementaremos uma pilha de valores `double`. No início do programa, podemos inicializar o ponteiro de pilha como `NULL` e colocar o valor 0 na variável de topo da pilha, sinalizando que o próximo valor a ser empilhado, será inserido no índice 0 do vetor.

Quando formos adicionar um valor ao topo da pilha, precisamos alocar mais espaço para o novo elemento, colocar o novo elemento no índice do topo da pilha e incrementar o valor da variável topo.

Para desempilhar, basta tirar o valor **logo abaixo** do índice indicado pela variável topo (já que ela indica onde um novo elemento seria adicionado), decrementar a variável topo e realocar novamente para o novo tamanho da pilha.

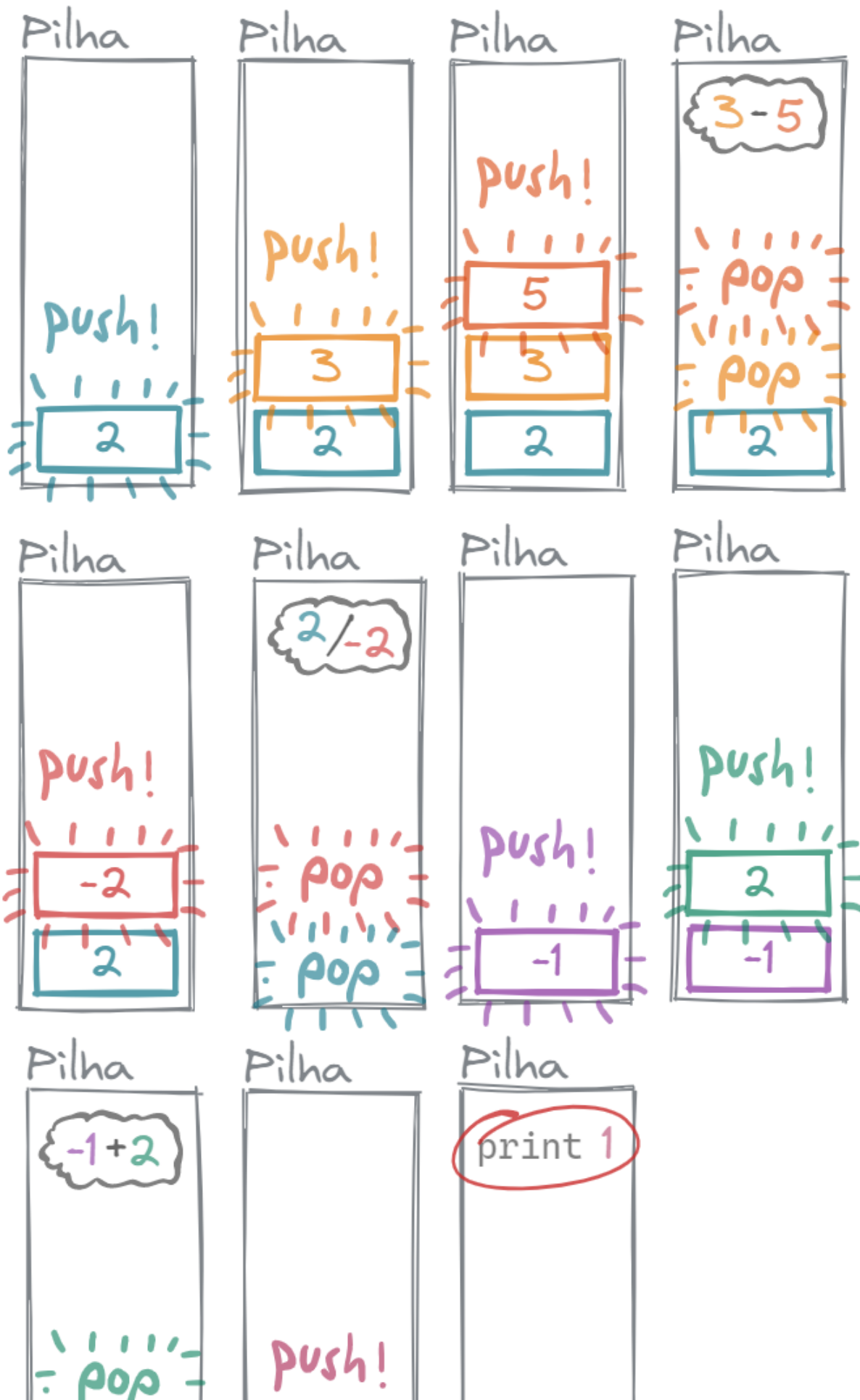
**Nota:** De maneira geral, não é muito eficiente ficar usando um `realloc` para cada operação na pilha, mas para os propósitos desse exercício, isso pode ser feito.

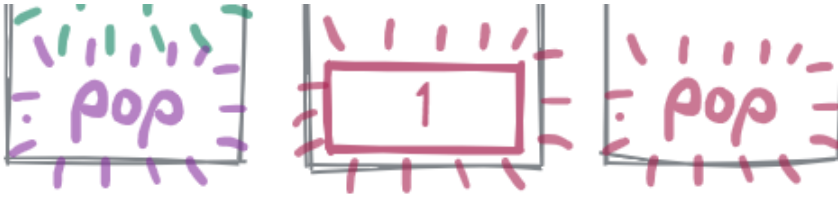
## Descrição

Implemente um programa em C que leia como entrada uma sequência de números ou operações até o final da entrada ( `EOF` ) e imprima o resultado dessas operações de acordo com as regras da Notação Polonesa Inversa. O programa deverá implementar a estrutura de dados pilha para funcionar. Para fazer essa calculadora, podemos simplesmente ir empilhando todos os números lidos da esquerda para a direita num pilha e, quando lemos uma operação, desempilhamos os dois valores do topo da pilha e fazemos essa operação com eles, que serão justamente os dois últimos valores lidos e os dois valores imediatamente a esquerda do operador. Depois de fazermos a operação, colocamos o resultado no topo da pilha para poder ser usado por outros operadores e continuamos a leitura.

De forma mais específica, se a próxima entrada for um número, seu programa deverá empilhar esse número na stack. Se for uma operação o seu programa deverá desempilhar o valor do topo da stack que será o lado direito da operação e desempilhar outro em seguida que será o valor esquerdo da operação. Por fim, após chegar no final da entrada, seu programa deverá retirar o valor que restou no topo da stack e imprimir ele. As seguintes operações são possíveis: `+`, `-`, `*`, `/`.

Aqui vai um exemplo de como seria computado o caso `2 3 5 - / 2 + :`





## Detalhes de implementação

É necessária a utilização de alocação dinâmica e da estrutura stack. A não utilização de qualquer um desses recursos resultará em perda de pontos na nota.

A saída deve conter 6 casas decimais de precisão.

## Dicas de implementação

Você pode definir a pilha, e variável topo como globais se isso for mais fácil.

Nesse programa, não há como prever se a próxima entrada é um número ou uma operação. Isso pode ser complicado já que não é possível determinar de imediato se precisamos dar `scanf` num número ou num char (para o operador). Entretanto, corrigir isso é relativamente fácil, basta consumir um único caractere da entrada padrão. Se esse caractere for um espaço, ignore, se for um operador, faça a operação, e se for um dígito existem duas possibilidades:

1. Criar sua própria função que lê o resto dos dígitos e adiciona o dígito já lido no começo do número;
2. ou, usar o `ungetc` da seguinte forma: digamos que o caractere lido para a verificação esteja armazenado numa variável `caractere` e seja um dígito, então podemos "devolver" o caractere lido a entrada padrão com `ungetc(caractere, stdin)`, e em seguida podemos usar o `scanf` normalmente. Ou seja, como nós "devolvemos" o caractere a entrada padrão, o `scanf` vai ler ele novamente.

**Nota:** `stdin` se refere a entrada padrão de textos (*standard input*).

## Formato da saída

A saída deve ser composta da palavra "Resultado: " seguido do resultado da conta em RPN fornecida na entrada com 6 casas decimais de precisão. Ao final da saída, imprima um `'\n'`.

## Exemplos

### Exemplo 1:

Entrada: 51 48 +  
Saída:  
Resultado: 99.000000

### Exemplo 2:

Entrada: 75 42 + 61 92 52 / \* /  
Saída:  
Resultado: 1.084105

### Exemplo 3:

Entrada: 16.68 17.69 + 9.19 1.86 \* 13.87 + + 14.95 - 6.36 / 2.26 +  
Saída:  
Resultado: 10.181918

## Quaisquer dúvidas ou problemas a relatar

Envie uma mensagem para o monitor Gabriel Dertoni via telegram @GabrielDertoni (<https://t.me/GabrielDertoni>) ou no Discord da disciplina (<https://discord.gg/9gQ5YQfFsA>). Se preferir, também pode enviar um email para o professor Leonardo [leonardop@usp.br](mailto:leonardop@usp.br) (<mailto:leonardop@usp.br>) ou para o Gabriel [gab.dertoni@usp.br](mailto:gab.dertoni@usp.br) (<mailto:gab.dertoni@usp.br>).

[Esconder Descrição](#)**Este exercício aceita os seguintes tipos de arquivos:**☐ C[Baixar Casos de Teste \(/Exercises/downloadCases/20581\)](/Exercises/downloadCases/20581)

Novo Envio

[G \(/Exercises/exportExerciseToGoogleCalendar/20581\)](/Exercises/exportExerciseToGoogleCalendar/20581)

O exercício está fechado

**23/07/2021 23:59:59**[Fechado](#)

Meu Último Envio

[Download \(/Commits/download/1422559\)](/Commits/download/1422559)

status

**Finalizado**

compilado

**Sim**

casos corretos

**50/50**

pontuação

**10.00**

Caso	Status	Tempo de CPU	Tam. de Memória Utilizado	Mensagem
Caso 1	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 2	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 3	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 4	Correto	0.0011 s	-1 Kb	Resposta Correta
Caso 5	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 6	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 7	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 8	Correto	0.0013 s	-1 Kb	Resposta Correta
Caso 9	Correto	0.0013 s	-1 Kb	Resposta Correta

Caso	Status	Tempo de CPU	Tam. de Memória Utilizado	Mensagem
Caso 10	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 11	Correto	0.0011 s	-1 Kb	Resposta Correta
Caso 12	Correto	0.0011 s	-1 Kb	Resposta Correta
Caso 13	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 14	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 15	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 16	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 17	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 18	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 19	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 20	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 21	Correto	0.0014 s	-1 Kb	Resposta Correta
Caso 22	Correto	0.0014 s	-1 Kb	Resposta Correta
Caso 23	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 24	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 25	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 26	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 27	Correto	0.0024 s	-1 Kb	Resposta Correta
Caso 28	Correto	0.0011 s	-1 Kb	Resposta Correta
Caso 29	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 30	Correto	0.0012 s	-1 Kb	Resposta Correta
Caso 31	Correto	0.0023 s	-1 Kb	Resposta Correta
Caso 32	Correto	0.0022 s	-1 Kb	Resposta Correta
Caso 33	Correto	0.0024 s	-1 Kb	Resposta Correta
Caso 34	Correto	0.0024 s	-1 Kb	Resposta Correta
Caso 35	Correto	0.0022 s	-1 Kb	Resposta Correta
Caso 36	Correto	0.0023 s	-1 Kb	Resposta Correta
Caso 37	Correto	0.0022 s	-1 Kb	Resposta Correta
Caso 38	Correto	0.0025 s	-1 Kb	Resposta Correta
Caso 39	Correto	0.0022 s	-1 Kb	Resposta Correta

Caso	Status	Tempo de CPU	Tam. de Memória Utilizado	Mensagem
Caso 40	Correto	0.0024 s	-1 Kb	Resposta Correta
Caso 41	Correto	0.0023 s	-1 Kb	Resposta Correta
Caso 42	Correto	0.0024 s	-1 Kb	Resposta Correta
Caso 43	Correto	0.0024 s	-1 Kb	Resposta Correta
Caso 44	Correto	0.0024 s	-1 Kb	Resposta Correta
Caso 45	Correto	0.0024 s	-1 Kb	Resposta Correta
Caso 46	Correto	0.0025 s	-1 Kb	Resposta Correta
Caso 47	Correto	0.0024 s	-1 Kb	Resposta Correta
Caso 48	Correto	0.0024 s	-1 Kb	Resposta Correta
Caso 49	Correto	0.0022 s	-1 Kb	Resposta Correta
Caso 50	Correto	0.0023 s	-1 Kb	Resposta Correta

### Detalhes dos Casos de Teste

Selecione um caso de teste...



### Histórico de Entregas

Data	Status	Corretos	Notas	Ações
10/07/2021 14:42:38	Finalizado	50/50	10.00	<a href="/Commits/download/1422559">Download (/Commits/download/1422559)</a> <a href="/commits/details/1422559">Detalhes (/commits/details/1422559)</a>
10/07/2021 14:35:19	Finalizado	50/50	10.00	<a href="/Commits/download/1422547">Download (/Commits/download/1422547)</a> <a href="/commits/details/1422547">Detalhes (/commits/details/1422547)</a>