

# AULA 14

# Pilhas

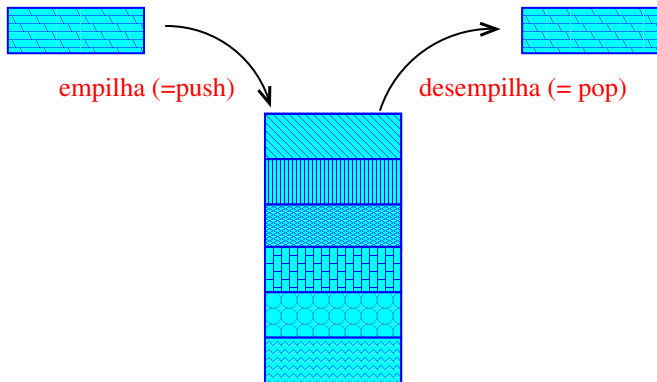
PF 6.1 e 6.3

<http://www.ime.usp.br/~pf/algoritmos/aulas/pilha.html>

[http://en.wikipedia.org/wiki/RPN\\_calculator](http://en.wikipedia.org/wiki/RPN_calculator)

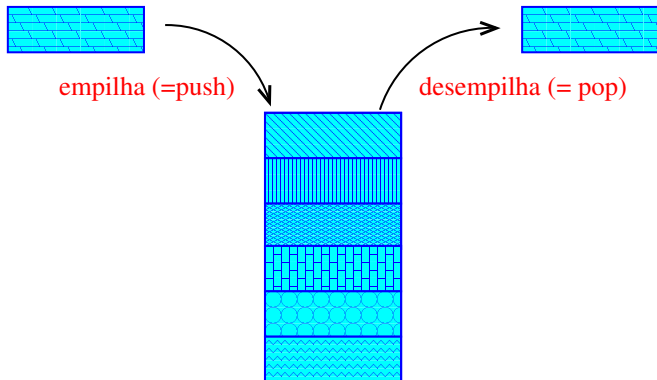
# Pilhas

Uma **pilha** (= *stack*) é uma lista (=sequência) dinâmica em que todas as operações (*inserções*, *remoções* e *consultas*) são feitas em uma mesma extremidade chamada de **topo**.



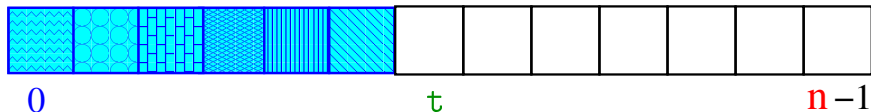
# Pilhas

Assim, o **primeiro** objeto a ser **removido** de uma pilha é o **último** que foi **inserido**. Esta política de manipulação é conhecida pela sigla **LIFO** (= *Last In First Out*)



## Implementação em um vetor

A pilha será armazenada em um vetor  $s[0 \dots n-1]$ .



O índice  $t$  indica o **topo** ( $=top$ ) da pilha.

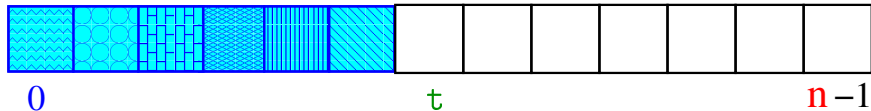
Esta é a **primeira posição vaga** da pilha.

A pilha está **vazia** se “ $t == 0$ ”.

A pilha está **cheia** se “ $t == n$ ”.

## Implementação em um vetor

A pilha será armazenada em um vetor  $s[0 \dots n-1]$ .



Para **remover** (=desempilhar=*pop*) um elemento faça

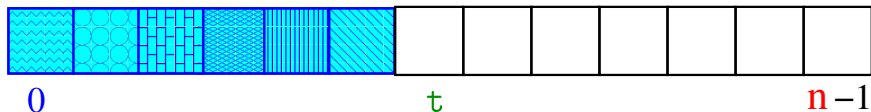
```
x = s[--t];
```

que é equivalente a

```
t -= 1;  
x = s[t];
```

## Implementação em um vetor

A pilha será armazenada em um vetor  $s[0 \dots n-1]$ .



Para *inserir* (=empilhar=*push*) um elemento faça

```
s[t++] = x;
```

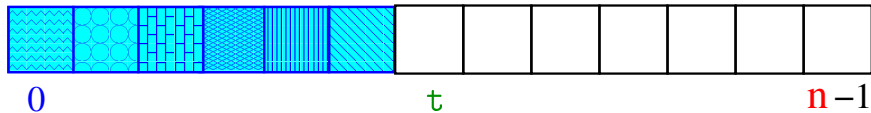
que é equivalente a

```
s[t] = x;
```

```
t += 1;
```

## Implementação em um vetor

A pilha será armazenada em um vetor  $s[0 \dots n-1]$ .



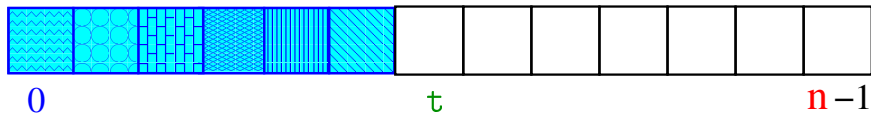
Para **consultar** um elemento, sem removê-lo, faça

```
x = s[t-1];
```



## Implementação em um vetor

A pilha será armazenada em um vetor  $s[0 \dots n-1]$ .



Tentar **desempilhar** de uma pilha que está **vazia** é um erro chamado *stack underflow*

Tentar **empilhar** em uma pilha **cheia** é um erro chamado *stack overflow*

## Notação polonesa

Usualmente os operadores são escritos **entre** os operandos como em

$$(A + B) * D + E / (F + A * D) + C$$

Essa é a chamada **notação infixa**.

Na **notação polonesa** ou **posfixa** os operadores são escritos **depois** do operandos

$$A B + D * E F A D * + / + C +$$

## Notação polonesa

**Problema:** Traduzir para **notação posfixa** a expressão infixa armazenada em uma cadeia de caracteres **inf**.

infixa	posfixa
$A+B*C$	$ABC*+$
$A*(B+C)/D-E$	$ABC+*D/E-$
$A+B*(C-D*(E-F)-G*H)-I*3$	$ABCDEF-* -GH*-*+I3*-$
$A+B*C/D*E-F$	$ABC*D/E*+F-$
$A+(B-(C+(D-(E+F))))$	$ABCDEF+-+--+$
$A*(B+(C*(D+(E*(F+G))))$	$ABCDEFG+*+*+*$

# Simulação

`inf` = expressão `infixa`

`s` = pilha

`posf` = expressão `posfixa`

# Simulação

inf [0 . . i-1]	s [0 . . t-1]	posf [0 . . j-1]
(	(	
(A	(	A
(A*	(*	A
(A*(	(*	A
(A*(B	(*	AB
(A*(B*	(*	AB
(A*(B*C	(*	ABC
(A*(B*C+	(*	ABC*
(A*(B*C+D	(*	ABC*D
(A*(B*C+D)	(*	ABC*D+
(A*(B*C+D))		ABC*D+*

# Pilha de execução

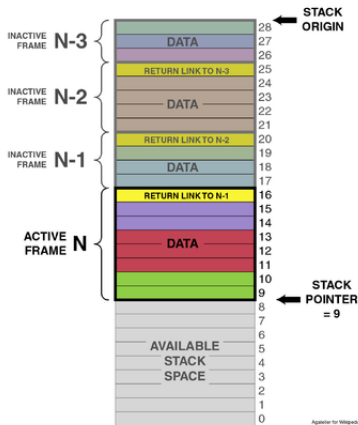
PF 6.5

<http://www.ime.usp.br/~pf/algoritmos/aulas/pilha.html>

[http://en.wikipedia.org/wiki/Call\\_stack](http://en.wikipedia.org/wiki/Call_stack)

# Pilha de execução

Para executar um programa, o computador utiliza uma **pilha de execução** (= *execution stack* = *call stack*).



# Pilha de execuções

Uma **pilha de execução** é usada para armazenar:

- ▶ **variáveis locais** das funções “ativas”;
- ▶ **parâmetros** das funções “ativas”;
- ▶ **endereço de retorno** para o ponto em que as funções foram chamadas;
- ▶ **cálculo** de expressões;



## Exemplo

```
int G (int a, int b) {
    int x;
    x = a + b;
    return x;
}
int F (int i, j, k) {
    int x;
    x = /*2*/ G (i, j) /*3*/;
    x = x + k;
    return x;
}
int main (void) {
    int i, j, k, y;
    i = 111; j = 222; k = 444;
    y = /*1*/ F (i, j, k) /*4*/;
    printf ("%d\n", y);
    return EXIT_SUCCESS;
}
```

?
?
?
?
?
?
?
?
?
?
?
?
?
?
?



Saida

	?
	?
	?
	?
	?
	?
	?
	?
	?
	?
y	?
k	?
j	?
i	?
end. retorno	010101010



Saida

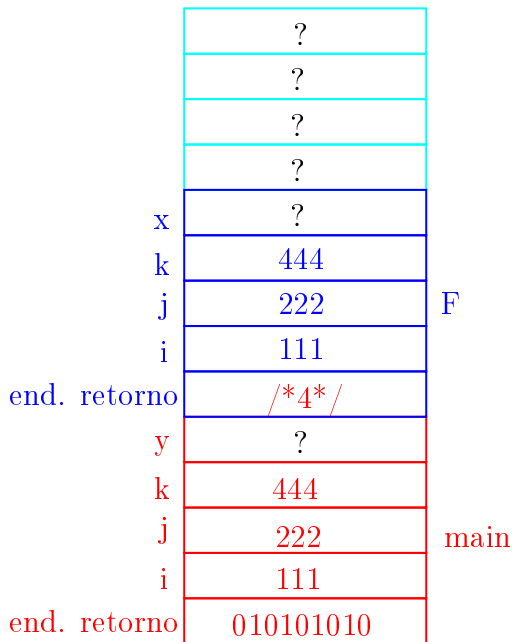
main

	?
	?
	?
	?
	?
	?
	?
	?
	?
	?
y	?
k	444
j	222
i	111
end. retorno	010101010



Saida

main



Saida

x	?	G
b	222	
a	111	
end. retorno	/*3*/	
x	?	
k	444	F
j	222	
i	111	
end. retorno	/*4*/	
y	?	
k	444	main
j	222	
i	111	
end. retorno	010101010	



Saida

x	333	G
b	222	
a	111	
end. retorno	/*3*/	
x	?	F
k	444	
j	222	
i	111	
end. retorno	/*4*/	
y	?	main
k	444	
j	222	
i	111	
end. retorno	010101010	



Saida

	?
	?
	?
	?
x	333
k	444
j	222
i	111
end. retorno	/*4*/
y	?
k	444
j	222
i	111
end. retorno	010101010

F

main



Saida



	?
	?
	?
	?
x	777
k	444
j	222
i	111
end. retorno	/*4*/
y	?
k	444
j	222
i	111
end. retorno	010101010

F

main



Saida

	?
	?
	?
	?
	?
	?
	?
	?
	?
y	777
k	444
j	222
i	111
end. retorno	010101010

main



Saida

?
?
?
?
?
?
?
?
?
?
?
?
?
?
?

777

Saida

## Exercício

Qual o **erro** na função a seguir?

```
CelPixel*
inserir(CelPixel *l, int a, int b)
{
    CelPixel* novo;
    novo.x = a;
    novo.y = b;
    novo.prox = l;
    return &novo;
}
```