Pilha

Estrutura de Dados

- Na computação, uma Estrutura de Dados é um modo particular de armazenamento e organização de dados em um computador, de modo que possam ser usados eficientemente;

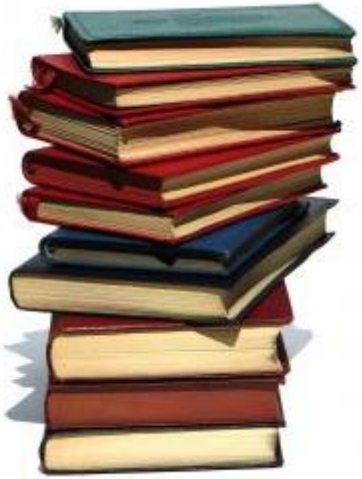
Estrutura de Dados

Dentre as principais Estrutura de Dados existentes, temos:

- Pilha;
- Fila;
- Lista;
- Árvore Binária;
- Grafo; e
- Tabela de Hashing.

Pilha (Stack) – LIFO

- A Pilha ou *Stack* é uma estrutura de dados baseada no princípio LIFO (*Last In*, *First Out*), no qual um dado nela inserido em primeiro será removido dela por último:



Pilha (Stack) – LIFO

Existem duas funções que se aplicam às pilhas:

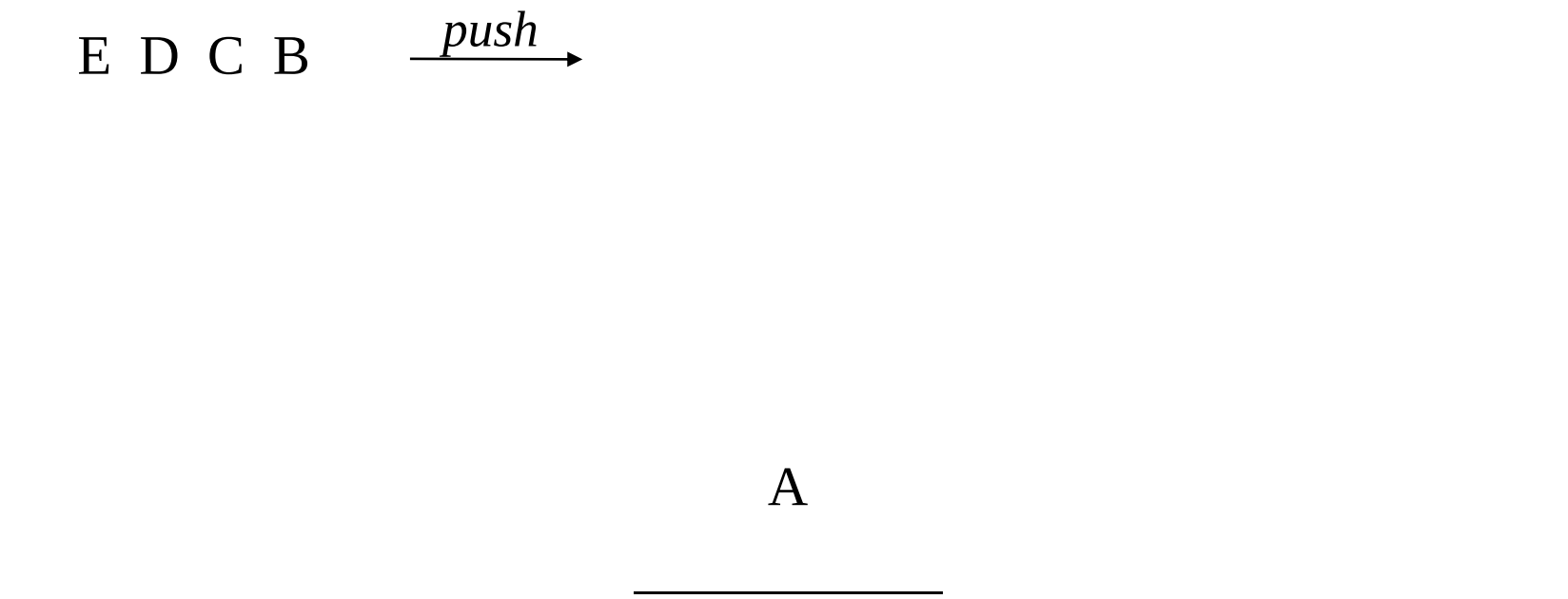
- *push* : Insere um dado no topo da Pilha; e

- *pop* : Remove o item do topo da Pilha.

Pilha (Stack) – LIFO

Entrada	Pilha	Saída
E D C B A		

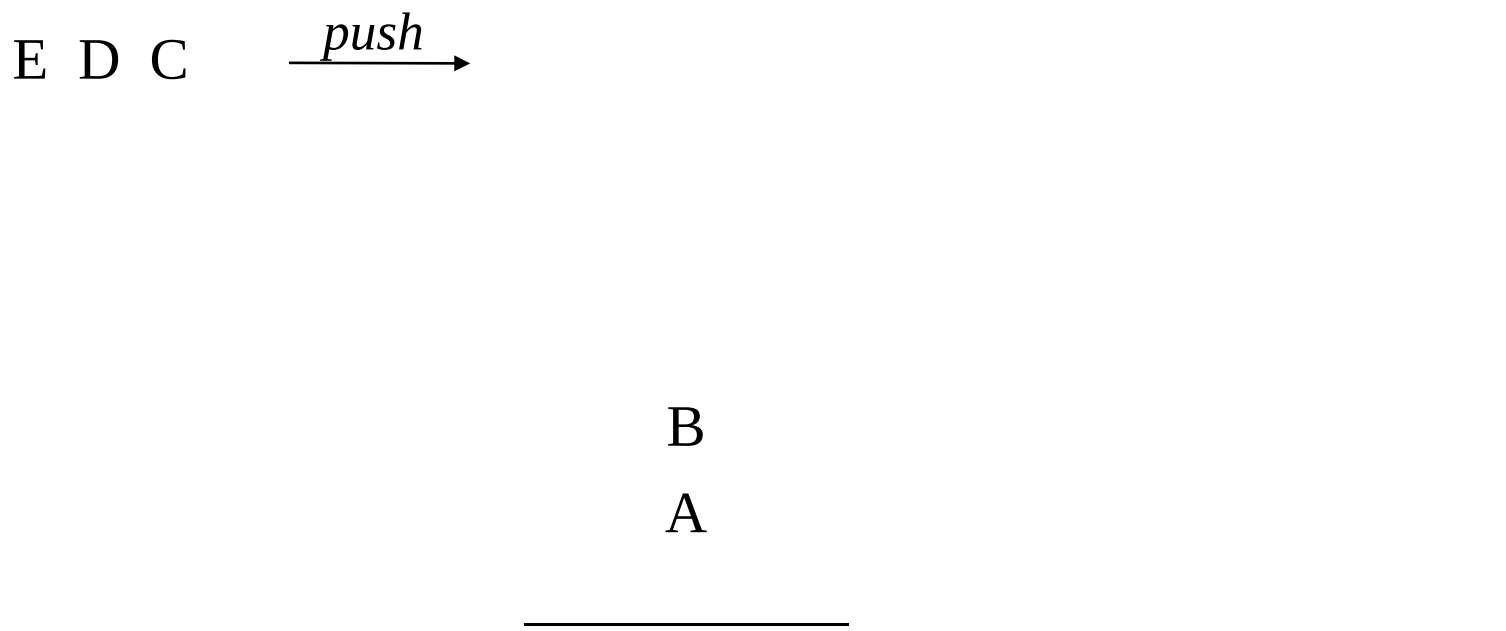
Pilha (Stack) – LIFO



Pilha (Stack) – LIFO

Entrada	Pilha	Saída
E D C B		
	A	

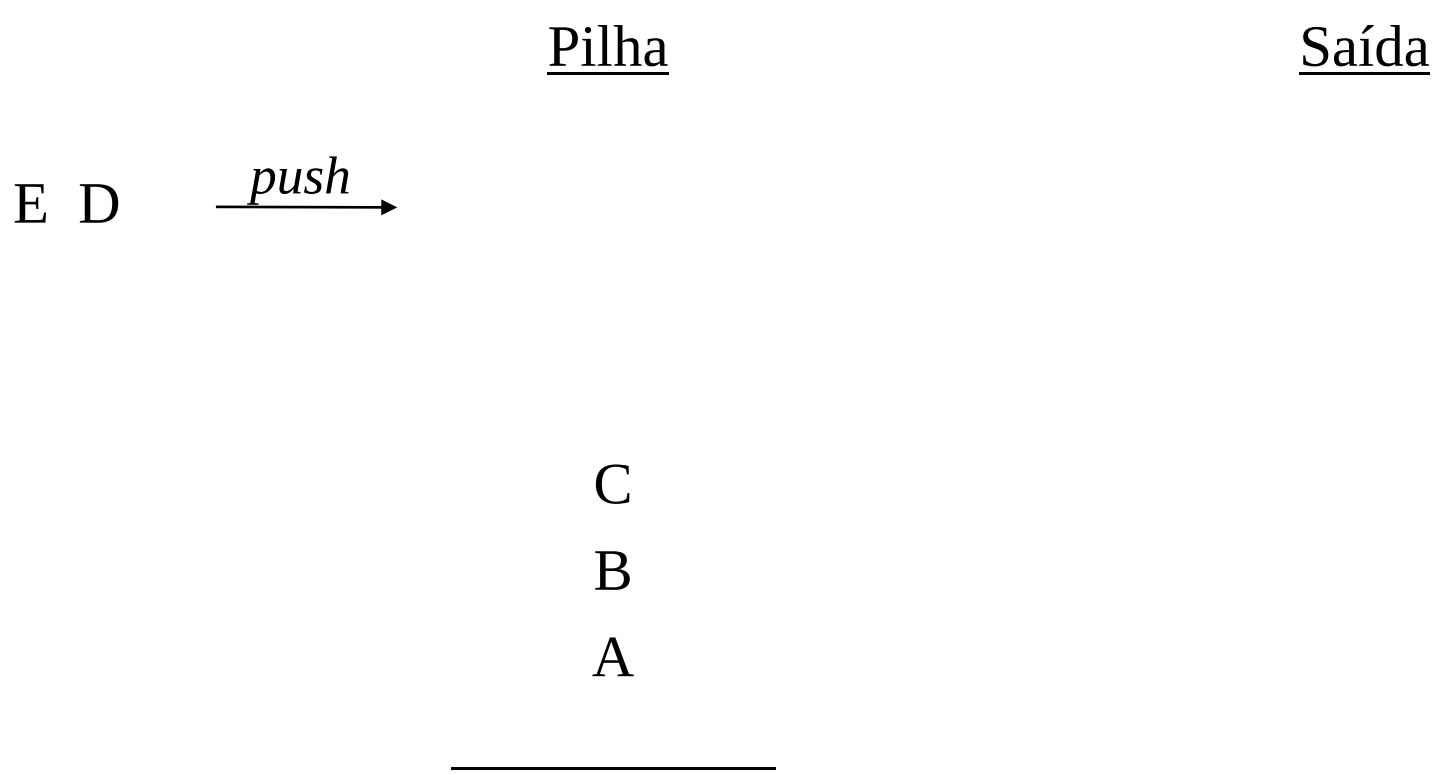
Pilha (Stack) – LIFO



Pilha (Stack) – LIFO

Entrada	Pilha	Saída
E D C		
	B	
	A	

Pilha (Stack) – LIFO



Pilha (Stack) – LIFO

Entrada | Pilha | Saída

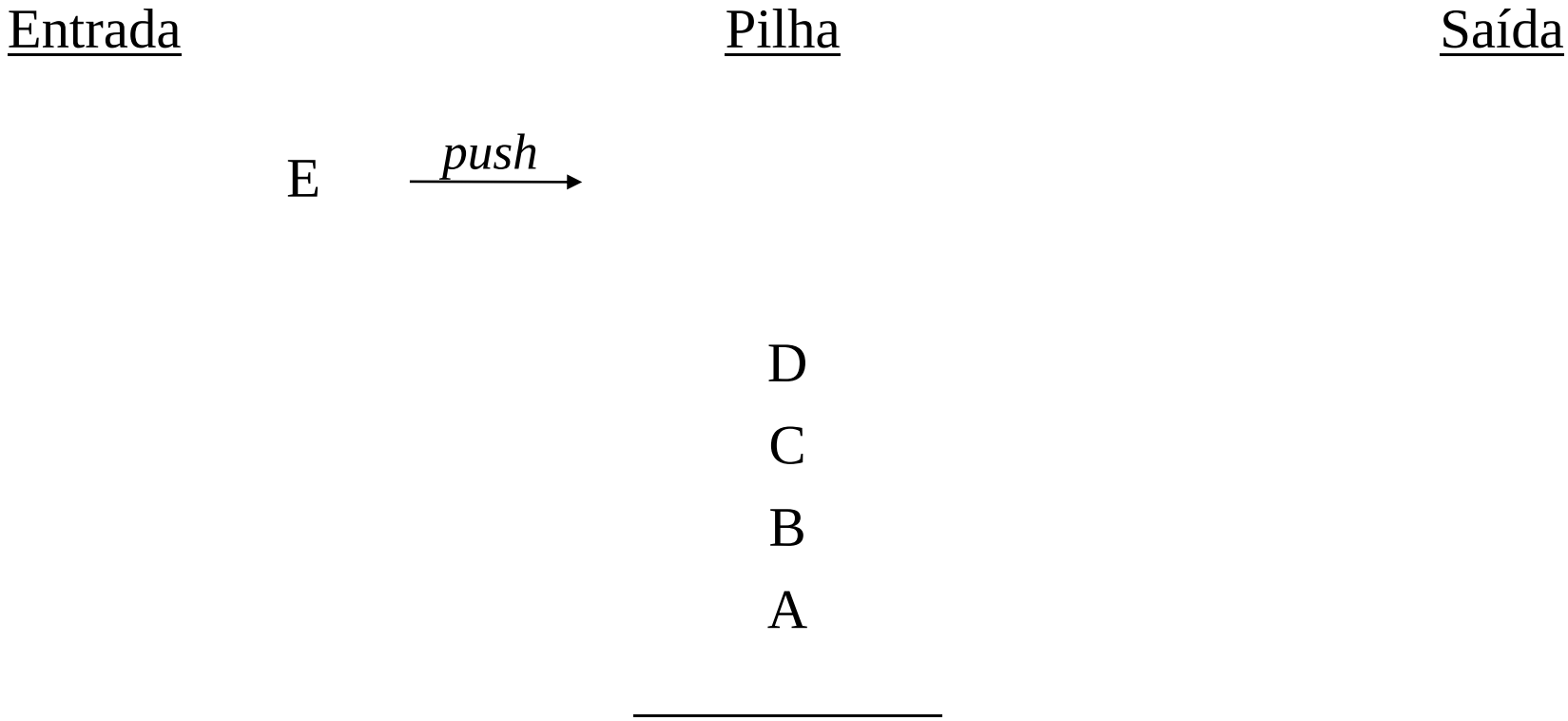
E D

C

B

A

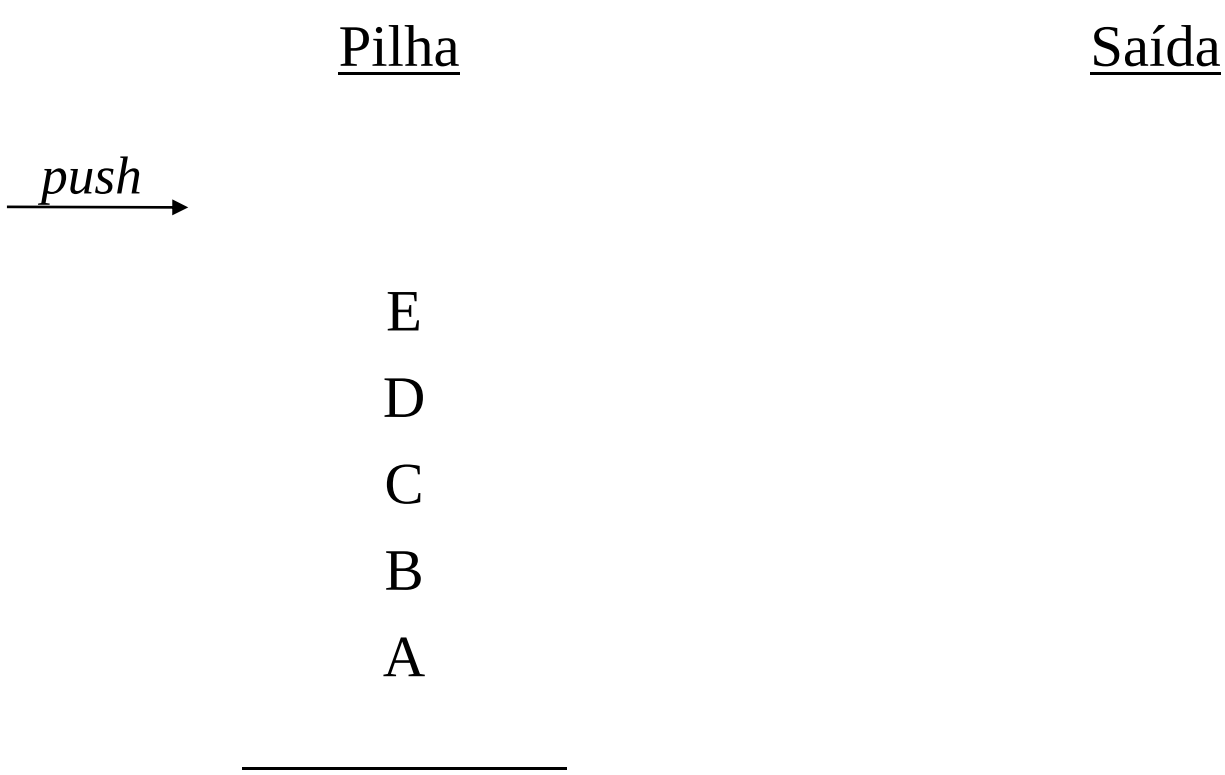
Pilha (Stack) – LIFO



Pilha (Stack) – LIFO

Entrada	Pilha	Saída
E		
	D	
	C	
	B	
	A	

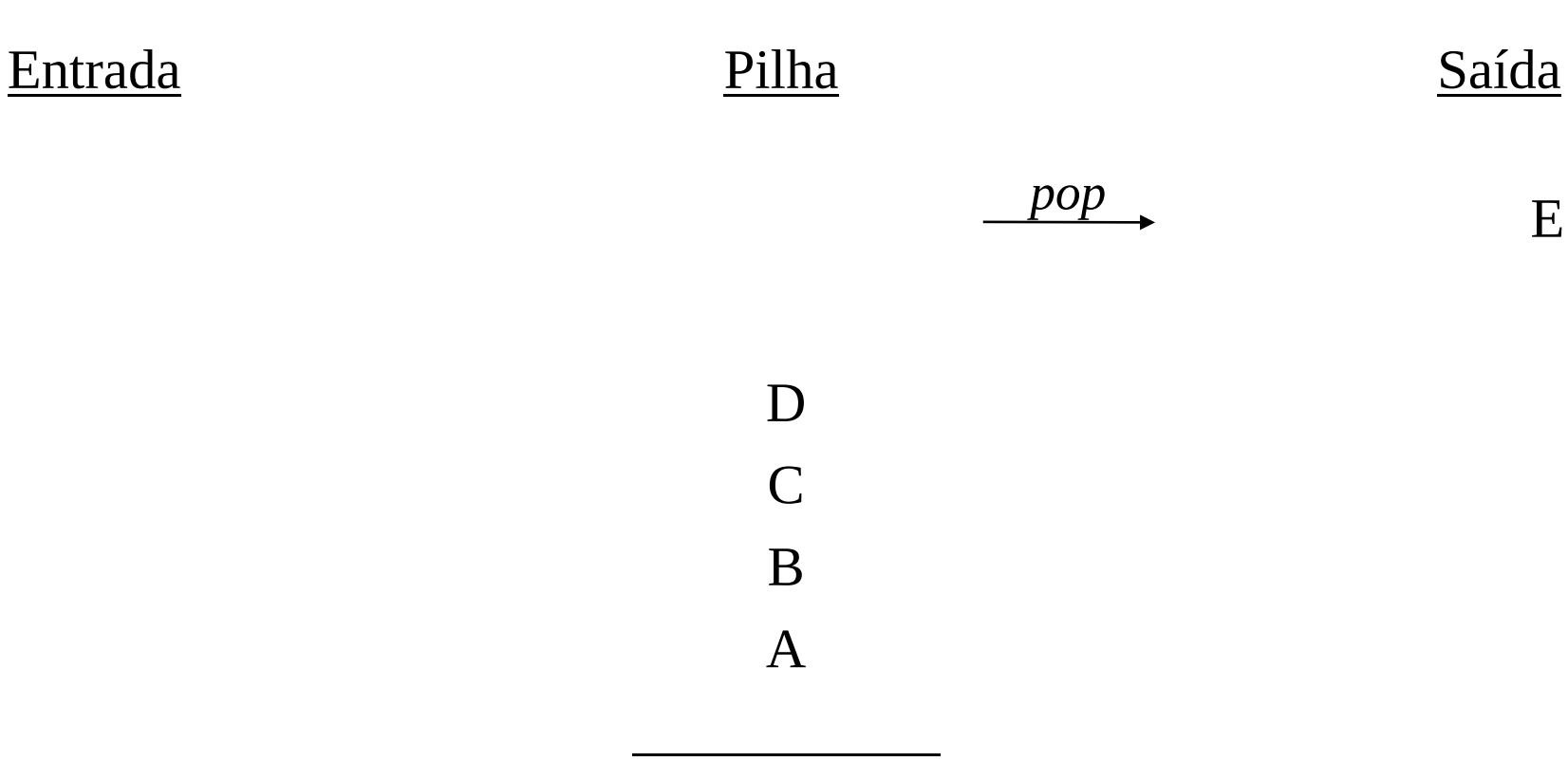
Pilha (Stack) – LIFO



Pilha (Stack) – LIFO

Entrada	Pilha	Saída
	E	
	D	
	C	
	B	
	A	
	_____	

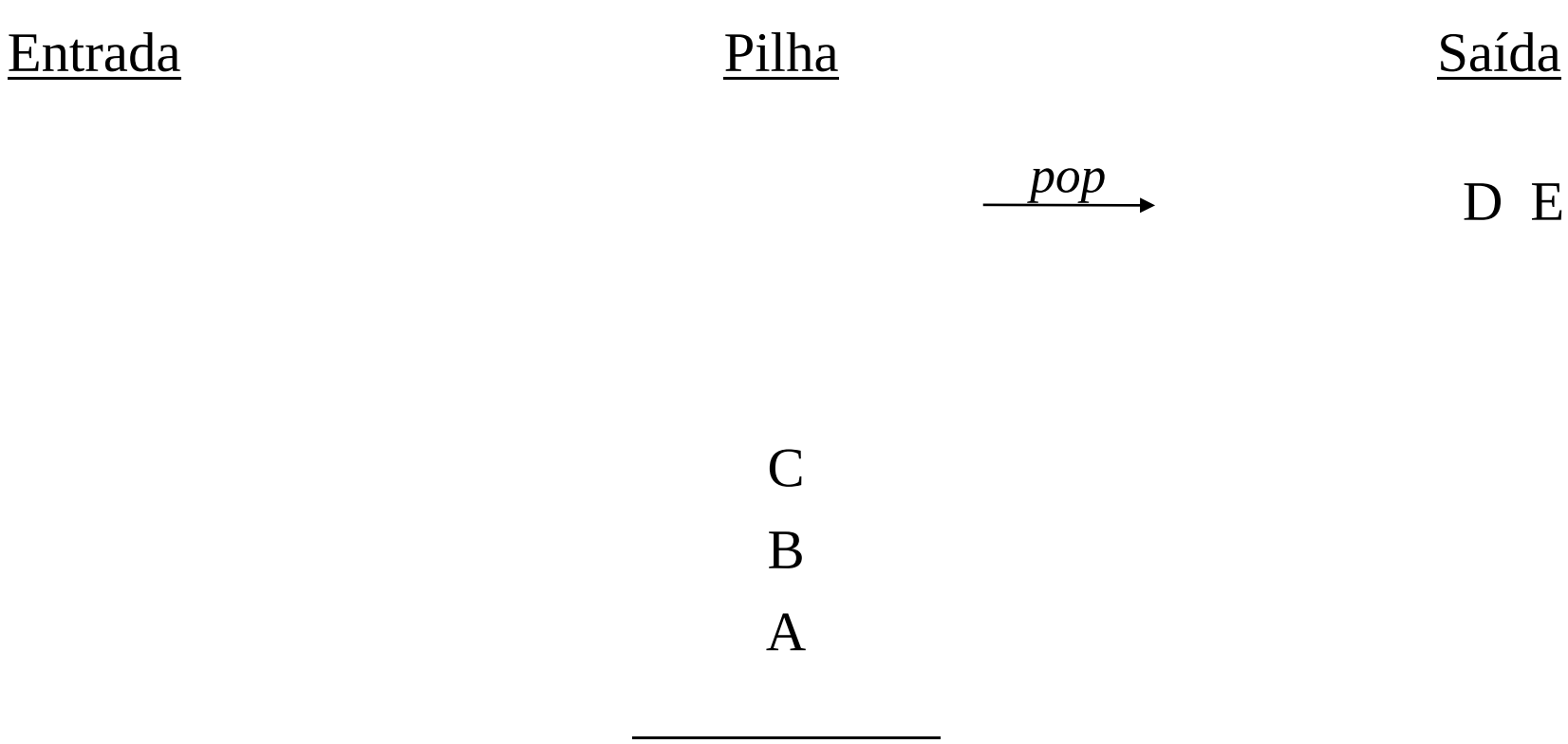
Pilha (Stack) – LIFO



Pilha (Stack) – LIFO

Entrada	Pilha	Saída
		E
	D	
	C	
	B	
	A	

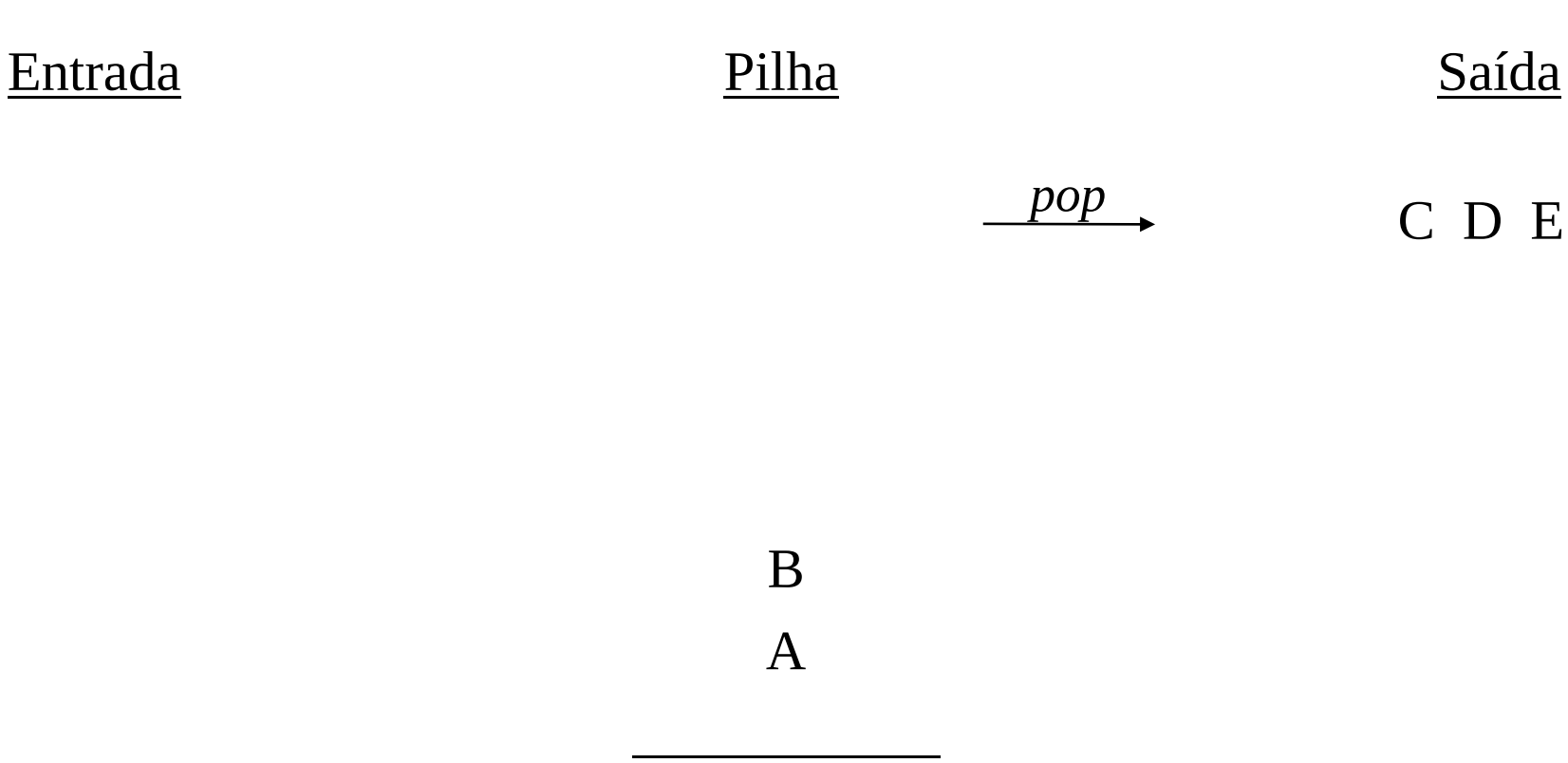
Pilha (Stack) – LIFO



Pilha (Stack) – LIFO

Entrada	Pilha	Saída
		D E
	C	
	B	
	A	

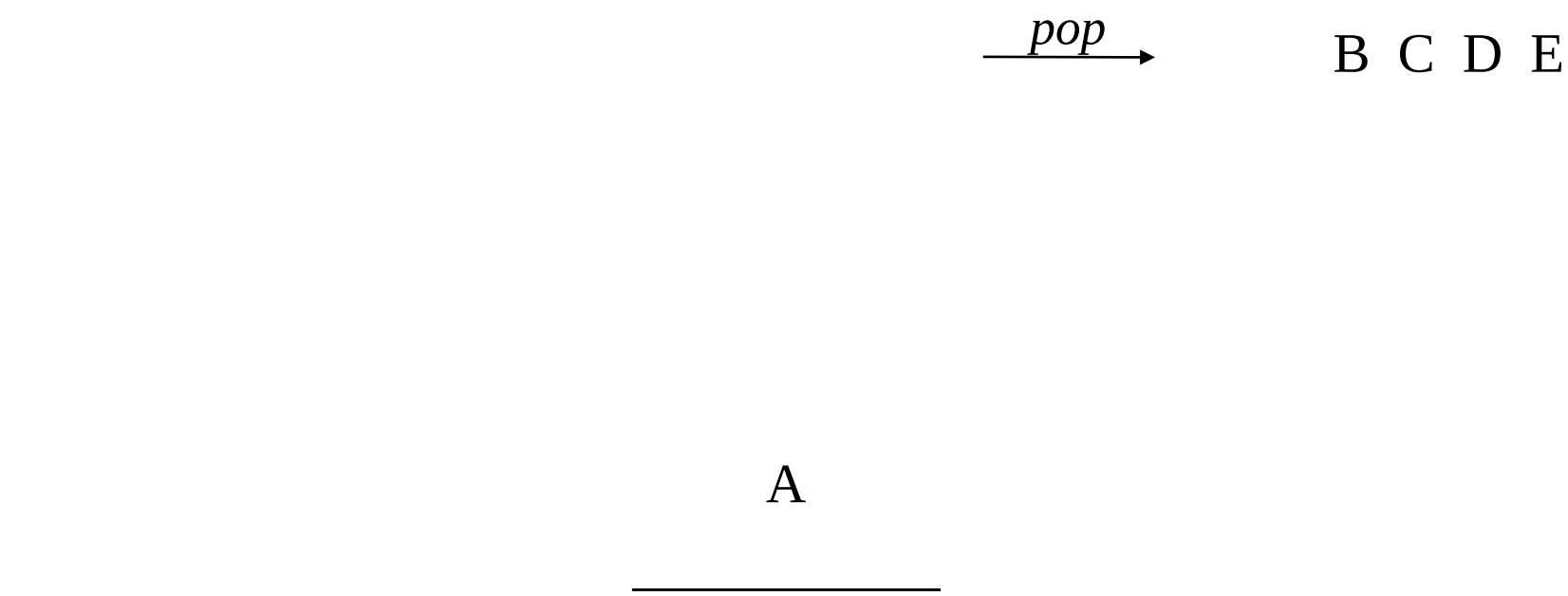
Pilha (Stack) – LIFO



Pilha (Stack) – LIFO

Entrada	Pilha	Saída
		C D E
	B	
	A	
	_____	

Pilha (Stack) – LIFO



Pilha (Stack) – LIFO

Entrada	Pilha	Saída
		B C D E
	A	
	_____	

Pilha (Stack) – LIFO

Entrada | Pilha | Saída

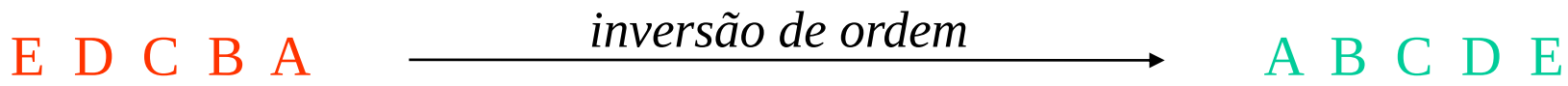
pop →

A B C D E

Pilha (Stack) – LIFO

Entrada	Pilha	Saída
		A B C D E

Pilha (Stack) – LIFO



Pilha (Stack) – LIFO

Exemplos de utilização de Pilhas:

- Chamada de Subprogramas (métodos, funções etc.);

- Inversão de ordem de uma coleção;

- Análise de Expressões e Sintaxe (Lógicas e Matemáticas).

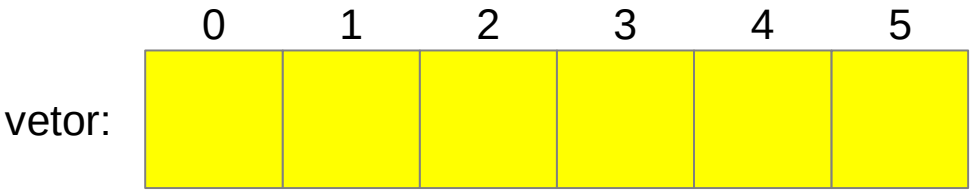
Pilha (Stack) – LIFO

Outras funções que se aplicam às pilhas:

- *size* : Informa o tamanho da referida Pilha; e

- *top* : Informa o elemento no topo da Pilha, sem retirá-lo.

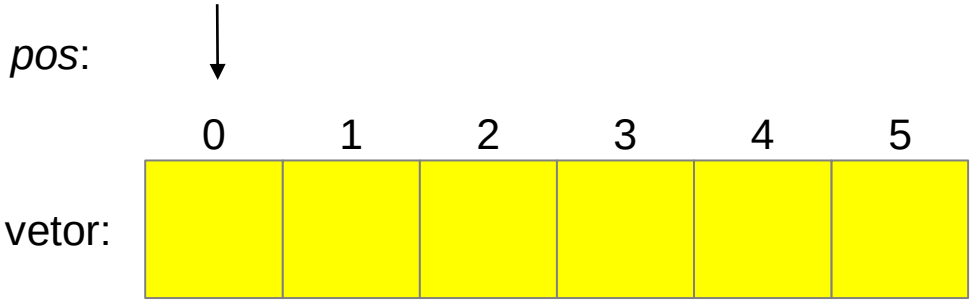
Pilha (Stack) – LIFO



int vetor[ ] = new vetor[6];

Vetor com o tamanho máximo necessário

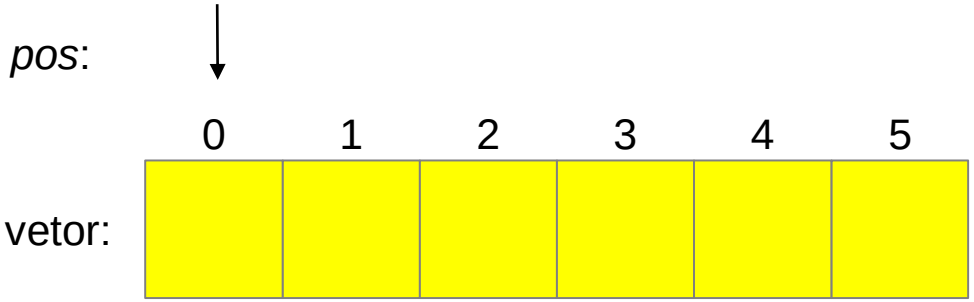
Pilha (Stack) – LIFO




```
int pos;
:
:
pos = 0;
```

Inicializa indicador de posição

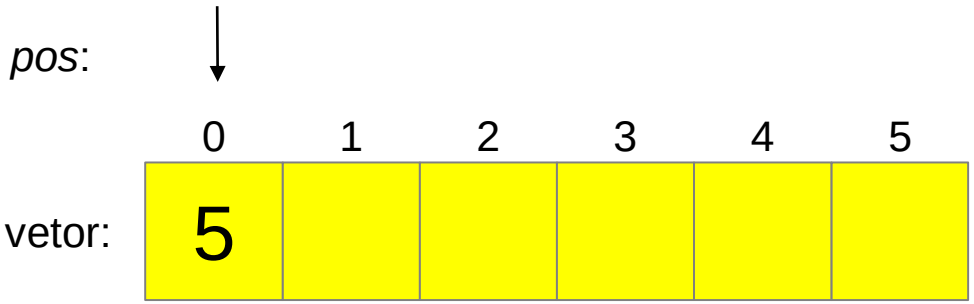
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(5);
```

Verifica se a pilha está cheia

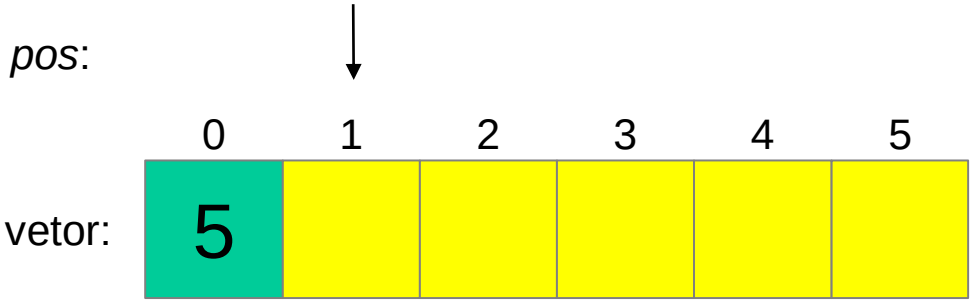
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(5);
```

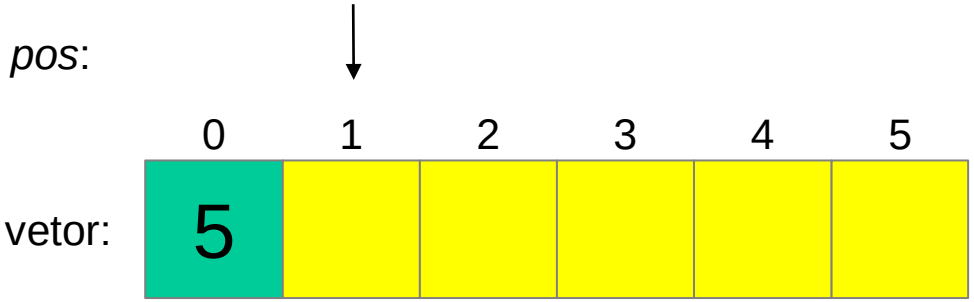
Não está cheia! Coloca o elemento 5 na pilha

Pilha (Stack) – LIFO



Incrementa indicador de posição

Pilha (Stack) – LIFO

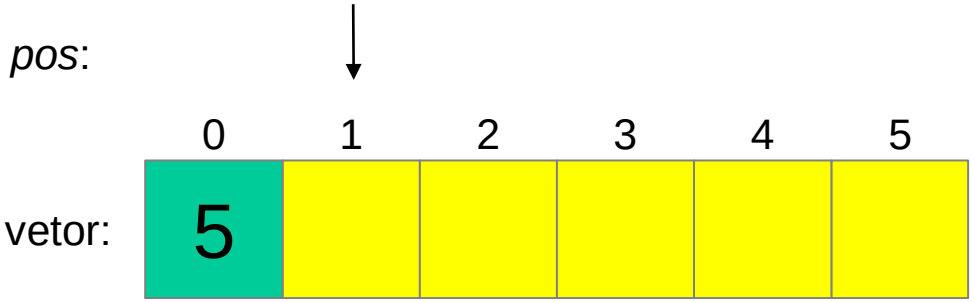



```
int topo;
:
:
topo = top( );

//topo: 5
```

Consulta o topo da pilha

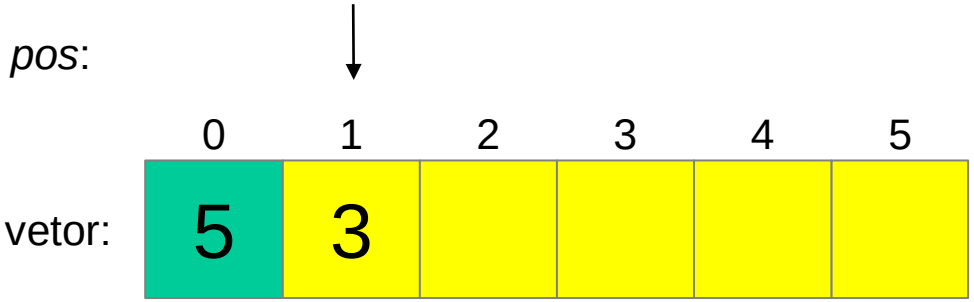
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(3);
```

Verifica se a pilha está cheia

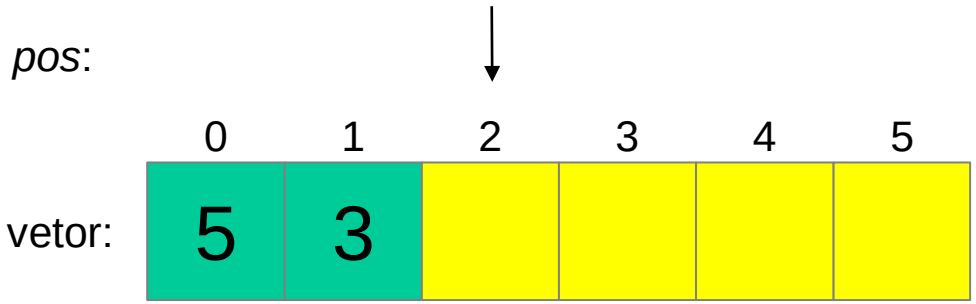
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(3);
```

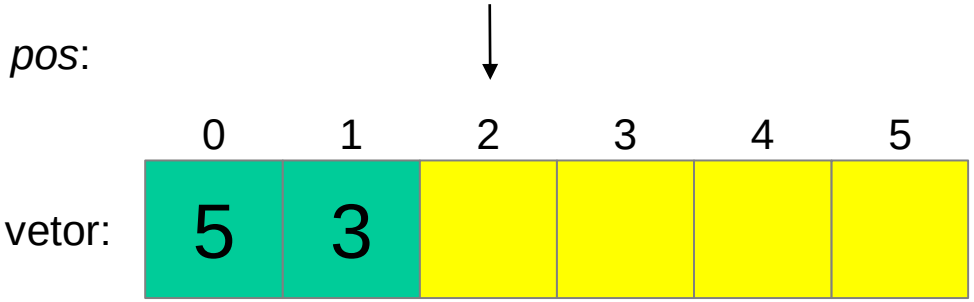
Não está cheia! Coloca o elemento 3 na pilha

Pilha (Stack) – LIFO



Incrementa indicador de posição

Pilha (Stack) – LIFO

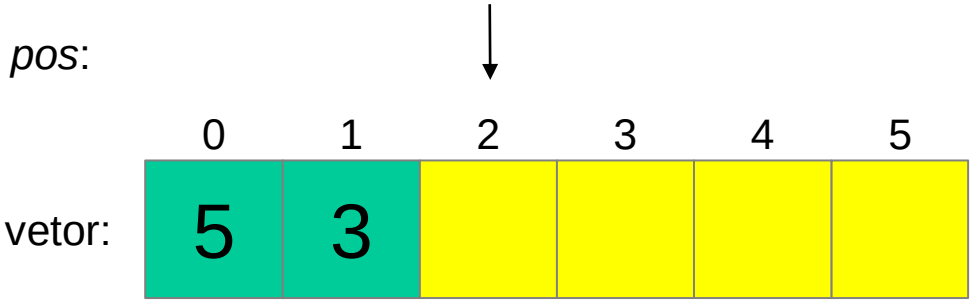



```
int topo;
:
:
topo = top( );

//topo: 3
```

Consulta o topo da pilha

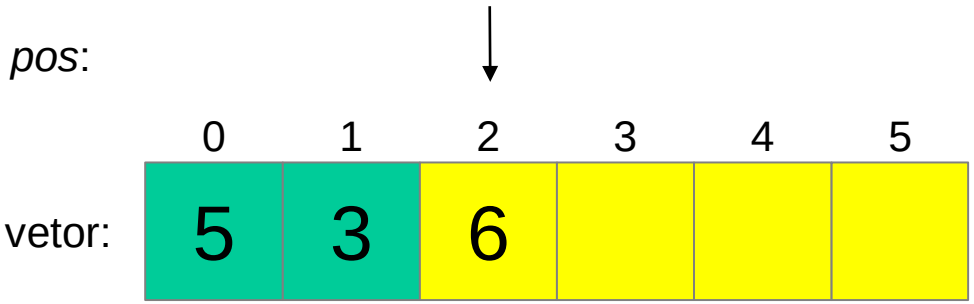
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(6);
```

Verifica se a pilha está cheia

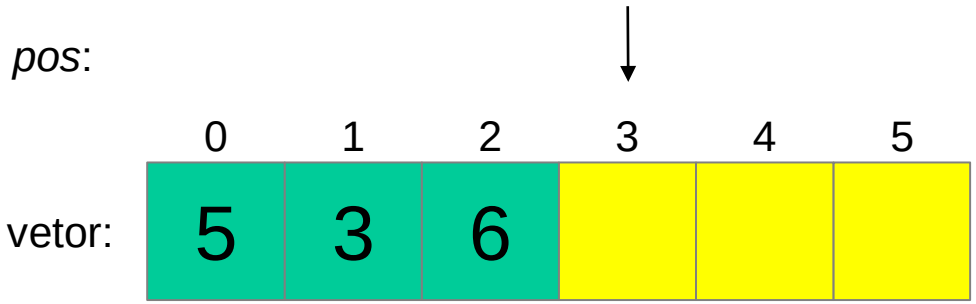
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(6);
```

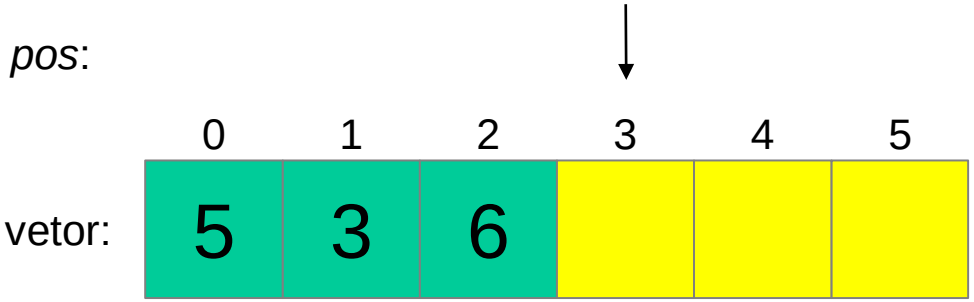
Não está cheia! Coloca o elemento 6 na pilha

Pilha (Stack) – LIFO



Incrementa indicador de posição

Pilha (Stack) – LIFO

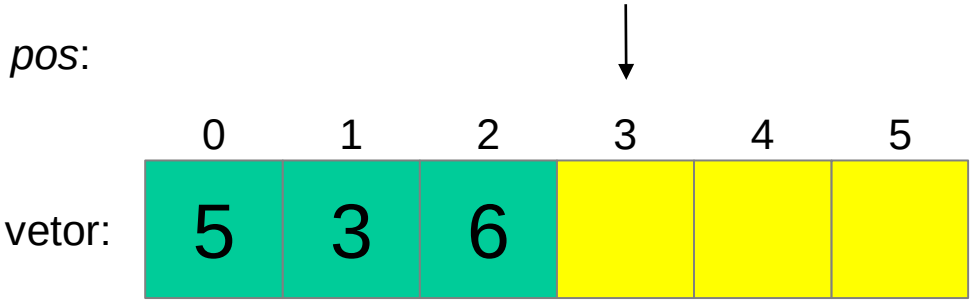



```
int topo;
:
:
topo = top( );

//topo: 6
```

Consulta o topo da pilha

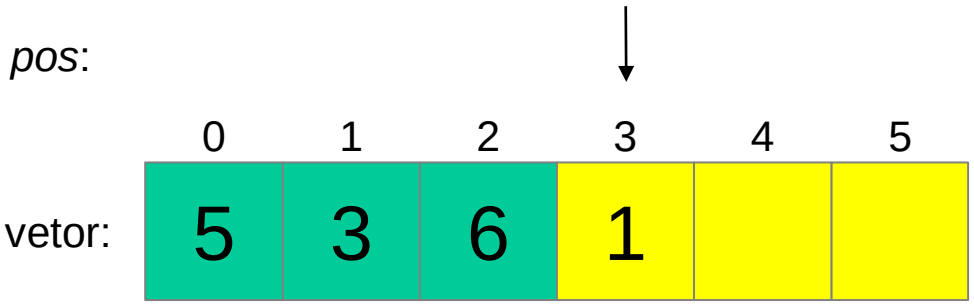
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(1);
```

Verifica se a pilha está cheia

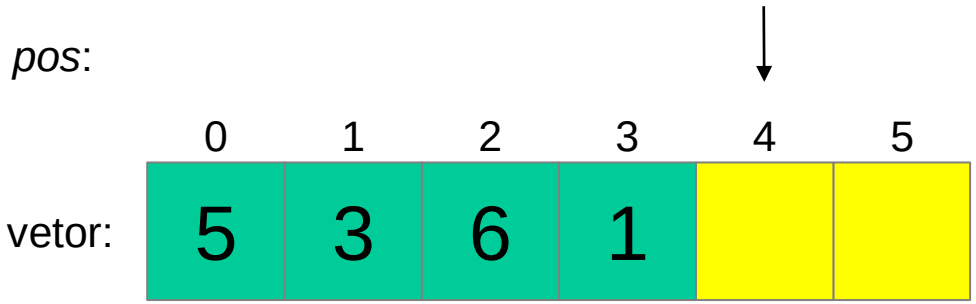
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(1);
```

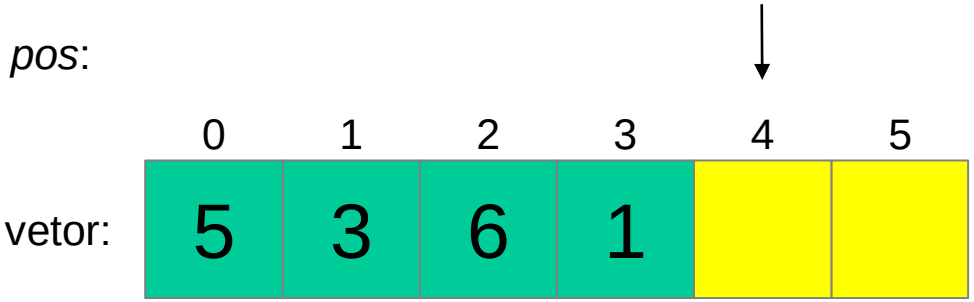
Não está cheia! Coloca o elemento 1 na pilha

Pilha (Stack) – LIFO



Incrementa indicador de posição

Pilha (Stack) – LIFO

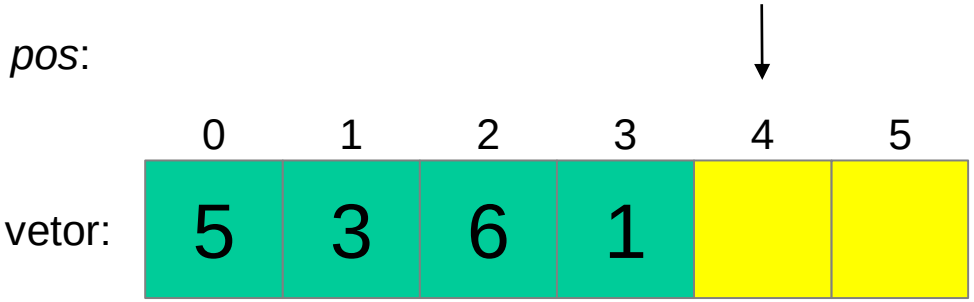



```
int topo;
:
:
topo = top( );

//topo: 1
```

Consulta o topo da pilha

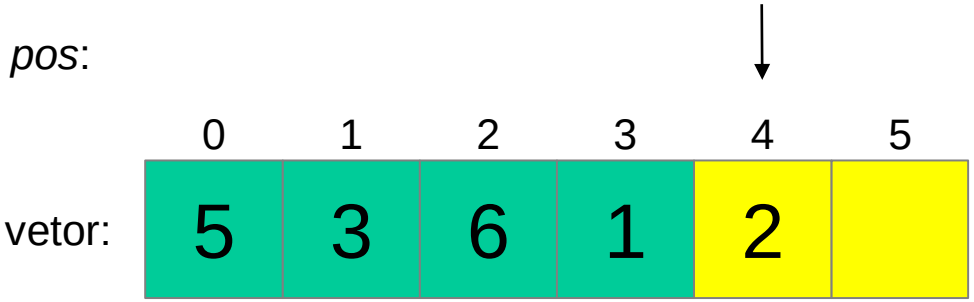
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(2);
```

Verifica se a pilha está cheia

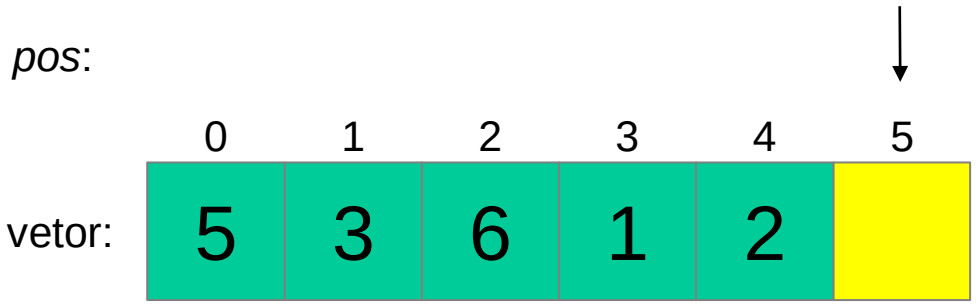
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(2);
```

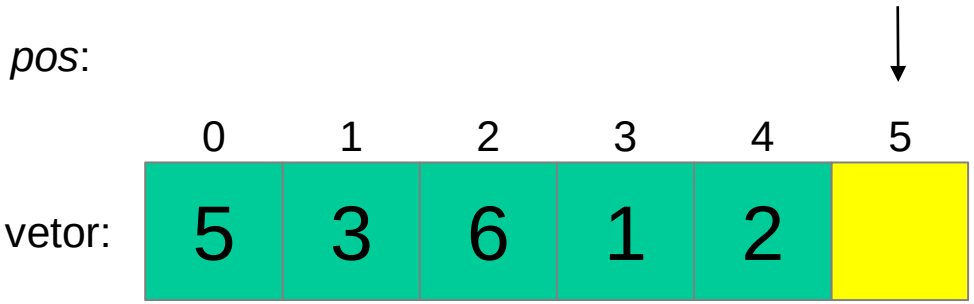
Não está cheia! Coloca o elemento 2 na pilha

Pilha (Stack) – LIFO



Incrementa indicador de posição

Pilha (Stack) – LIFO

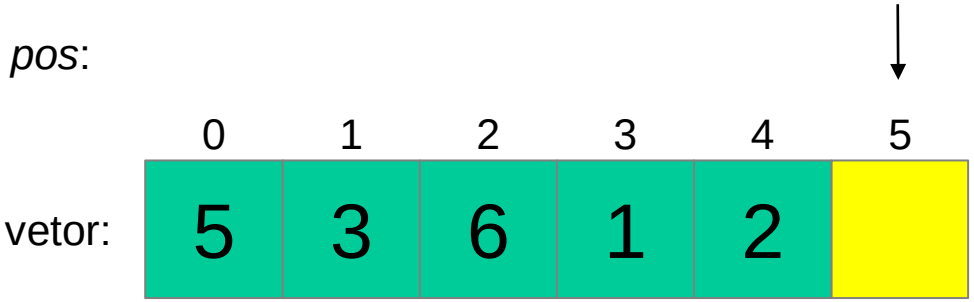



```
int topo;
:
:
topo = top( );

//topo: 2
```

Consulta o topo da pilha

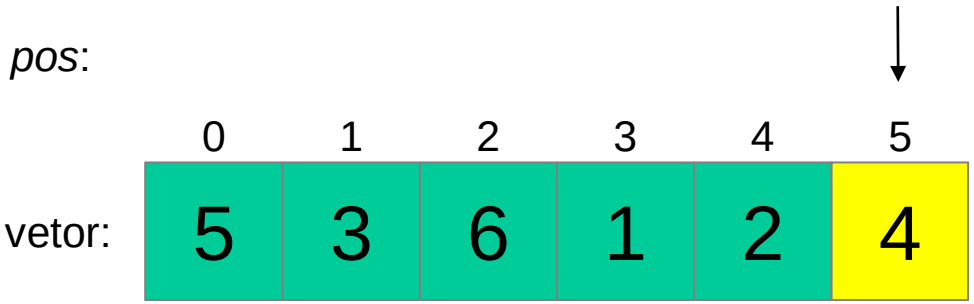
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(4);
```

Verifica se a pilha está cheia

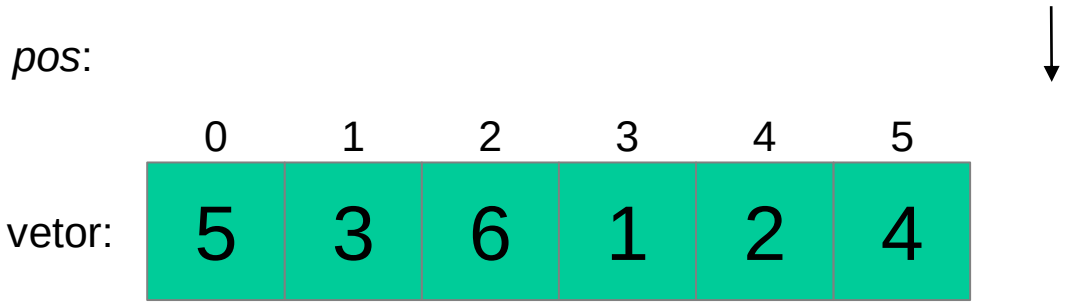
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(4);
```

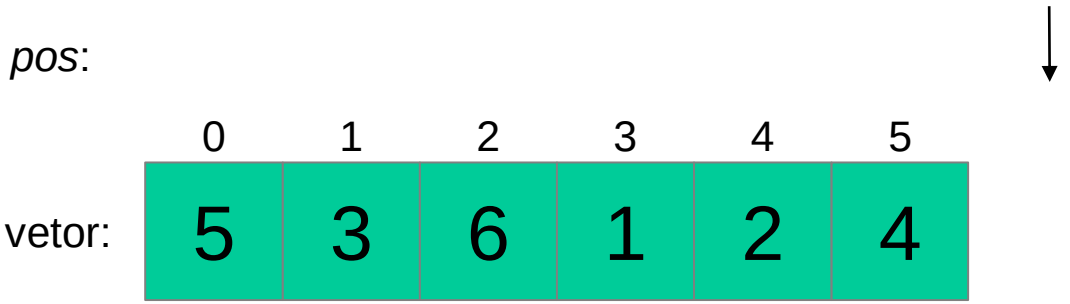
Não está cheia! Coloca o elemento 4 na pilha

Pilha (Stack) – LIFO



Incrementa indicador de posição

Pilha (Stack) – LIFO

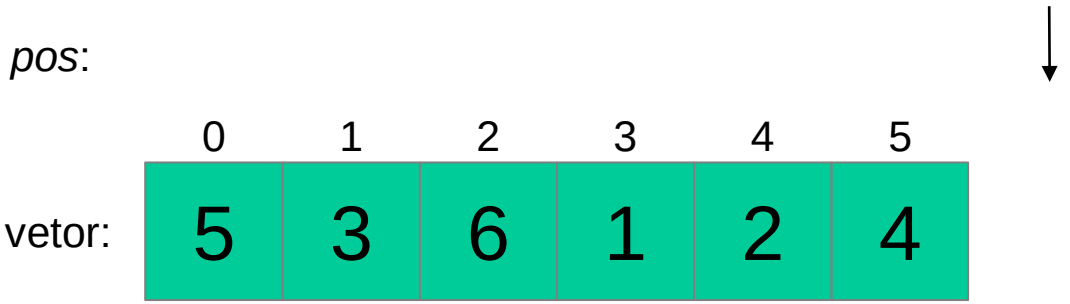



```
int topo;
:
:
topo = top( );

//topo: 4
```

Consulta o topo da pilha

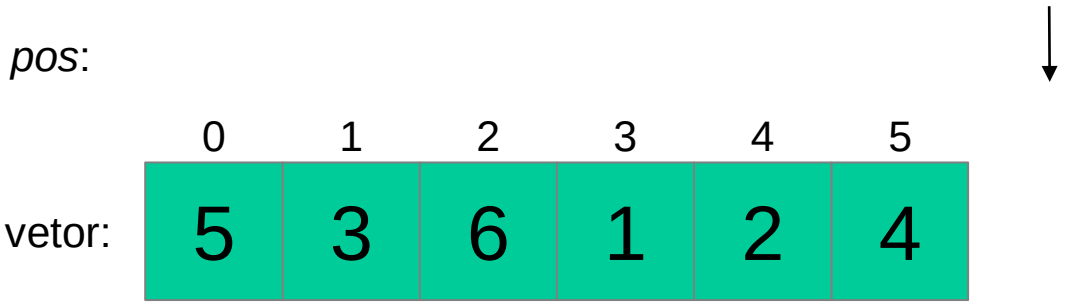
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(7);
```

Verifica se a pilha está cheia

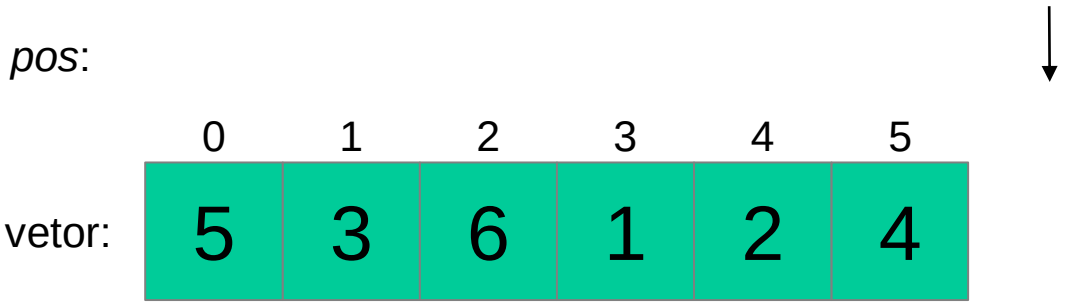
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(7);
```

Está cheia! Não coloca o elemento 7 na pilha

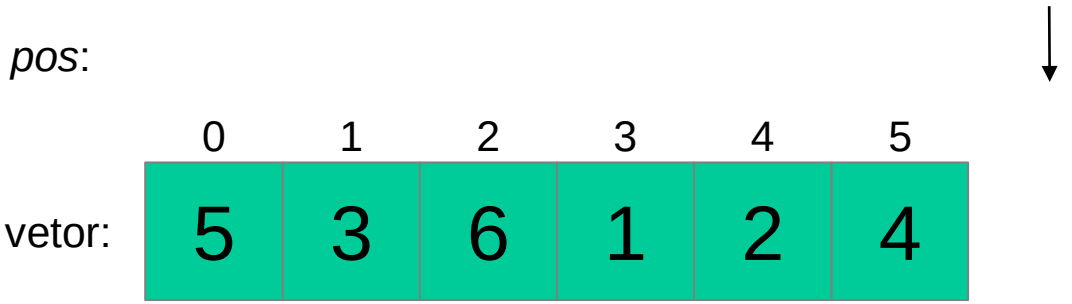
Pilha (Stack) – LIFO




```
int n;
:
:
if ( size( ) <= 0 )
    System.out.println("Pilha Vazia");
else
    n = pop( );
```

Verifica se a pilha está vazia

Pilha (Stack) – LIFO

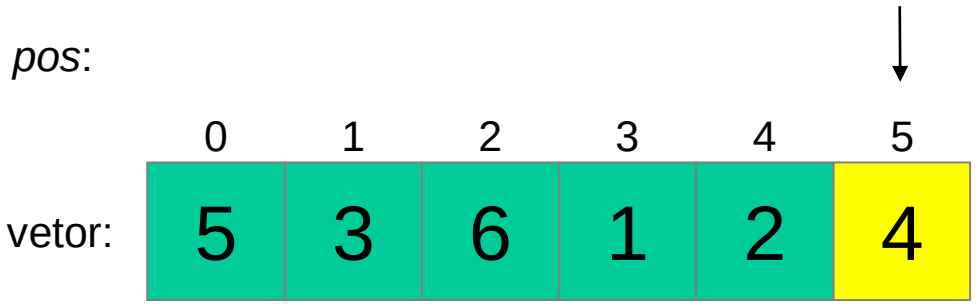



```
int n;
:
:
if ( size( ) <= 0 )
    System.out.println("Pilha Vazia");
else
    n = pop( );

    //n: 4
```

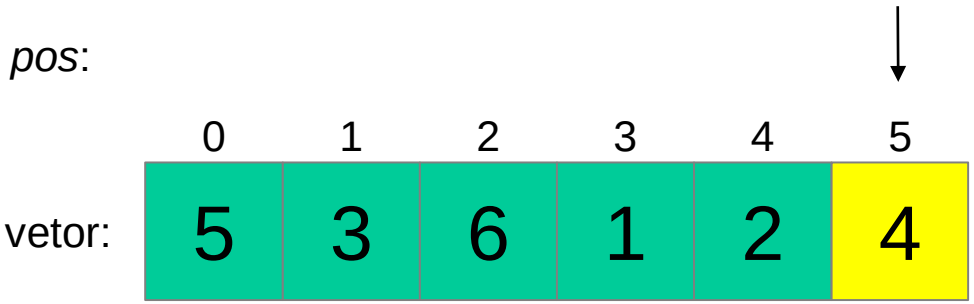
Não está vazia! Retira o topo da pilha

Pilha (Stack) – LIFO



Decrementa indicador de posição

Pilha (Stack) – LIFO

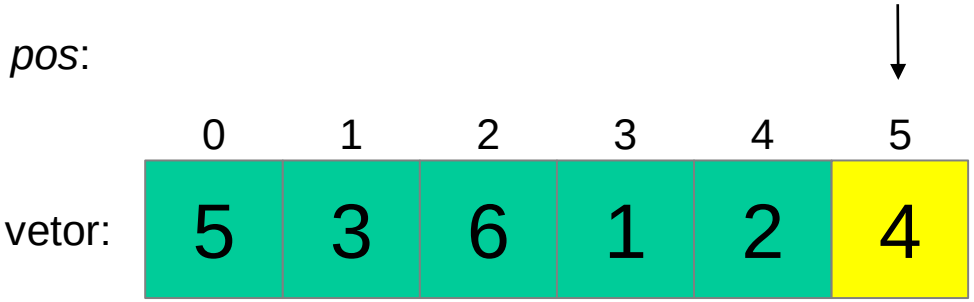



```
int topo;
:
:
topo = top( );

//topo: 2
```

Consulta o topo da pilha

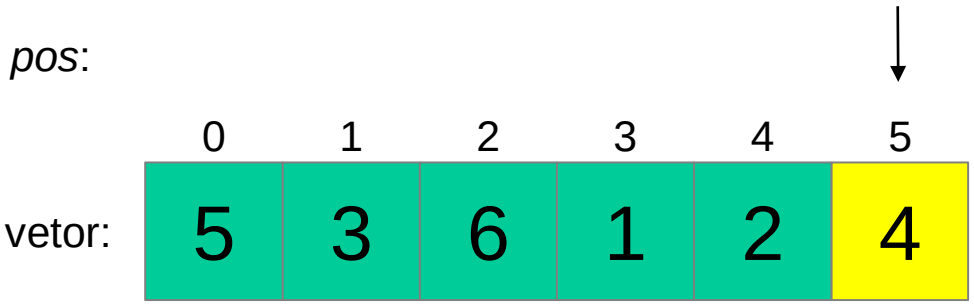
Pilha (Stack) – LIFO




```
int n;
:
:
if ( size( ) <= 0 )
    System.out.println("Pilha Vazia");
else
    n = pop( );
```

Verifica se a pilha está vazia

Pilha (Stack) – LIFO

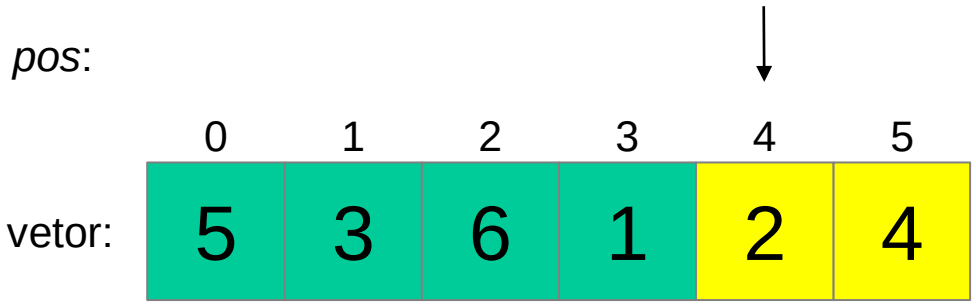



```
int n;
:
:
if ( size( ) <= 0 )
    System.out.println("Pilha Vazia");
else
    n = pop( );

    //n: 2
```

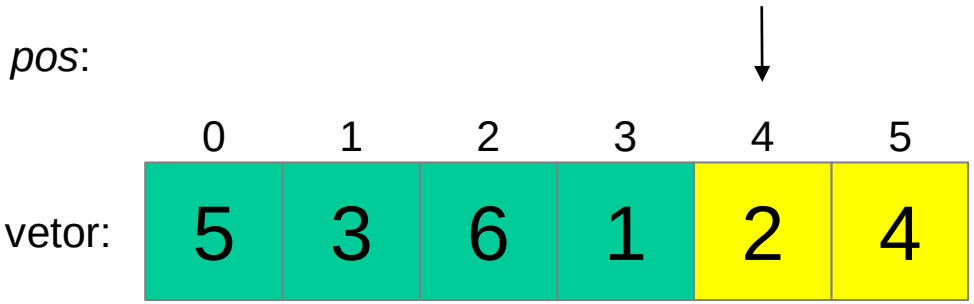
Não está vazia! Retira o topo da pilha

Pilha (Stack) – LIFO



Decrementa indicador de posição

Pilha (Stack) – LIFO

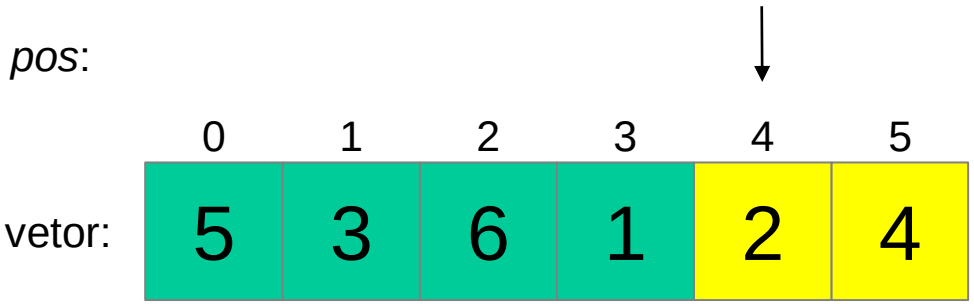



```
int topo;
:
:
topo = top( );

//topo: 1
```

Consulta o topo da pilha

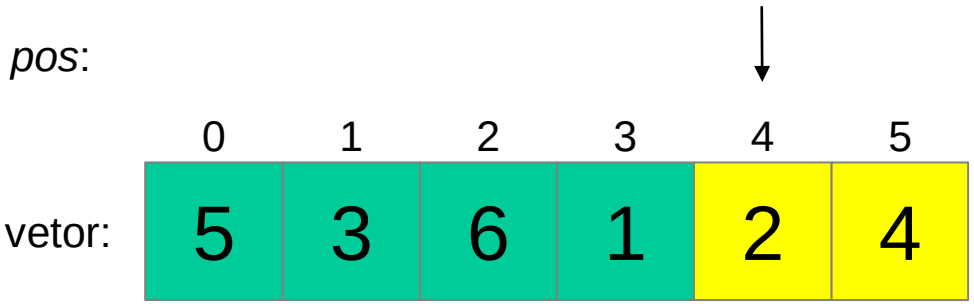
Pilha (Stack) – LIFO




```
int n;
:
:
if ( size( ) <= 0 )
    System.out.println("Pilha Vazia");
else
    n = pop( );
```

Verifica se a pilha está vazia

Pilha (Stack) – LIFO

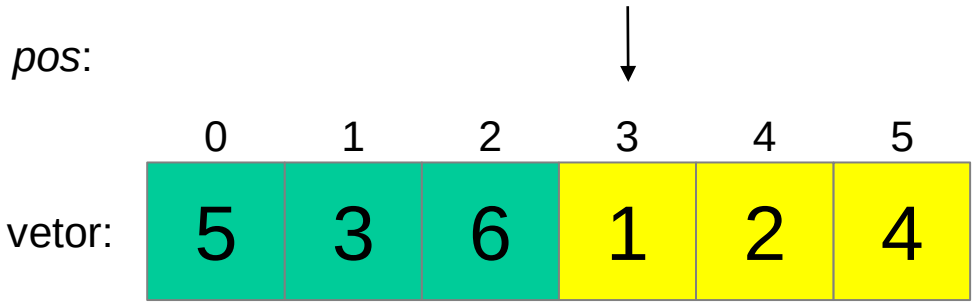



```
int n;
:
:
if ( size( ) <= 0 )
    System.out.println("Pilha Vazia");
else
    n = pop( );

    //n: 1
```

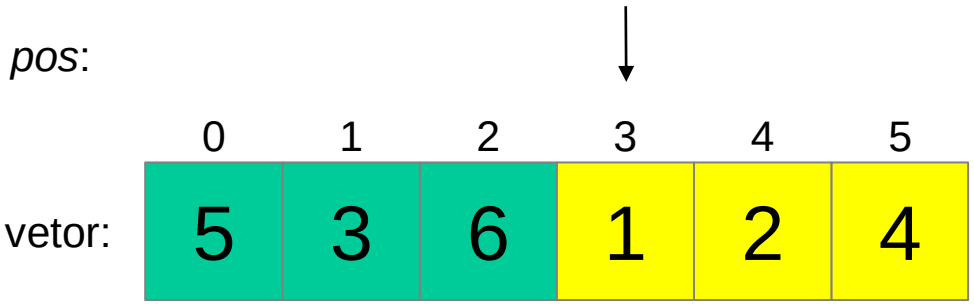
Não está vazia! Retira o topo da pilha

Pilha (Stack) – LIFO



Decrementa indicador de posição

Pilha (Stack) – LIFO

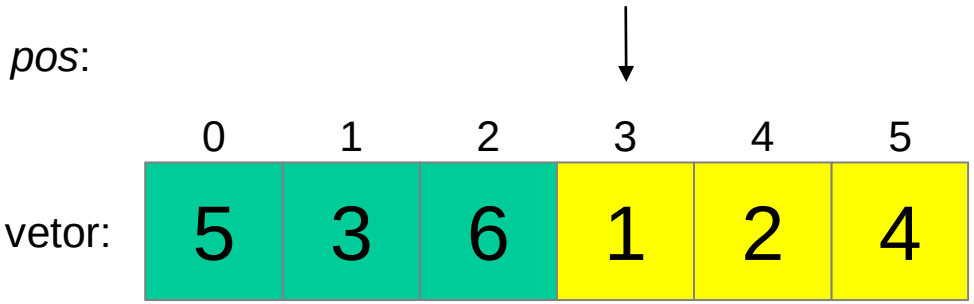



```
int topo;
:
:
topo = top( );

//topo: 6
```

Consulta o topo da pilha

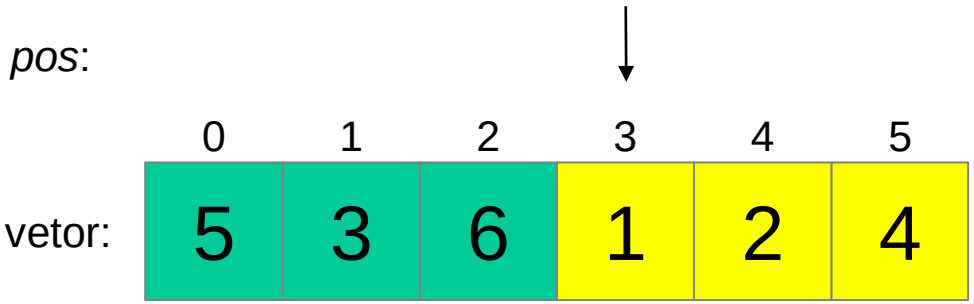
Pilha (Stack) – LIFO




```
int n;
:
:
if ( size( ) <= 0 )
    System.out.println("Pilha Vazia");
else
    n = pop( );
```

Verifica se a pilha está vazia

Pilha (Stack) – LIFO

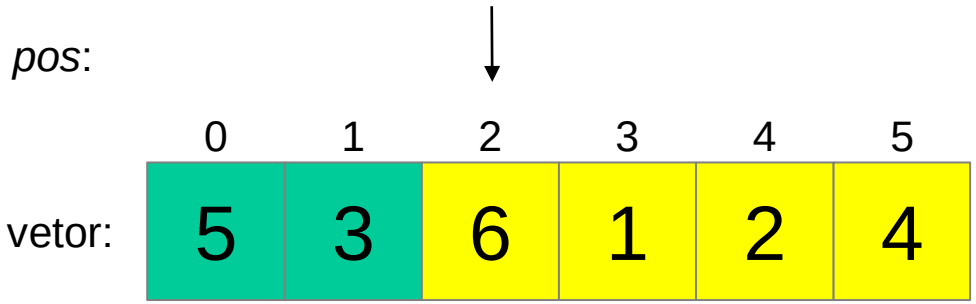



```
int n;
:
:
if ( size( ) <= 0 )
    System.out.println("Pilha Vazia");
else
    n = pop( );

    //n: 6
```

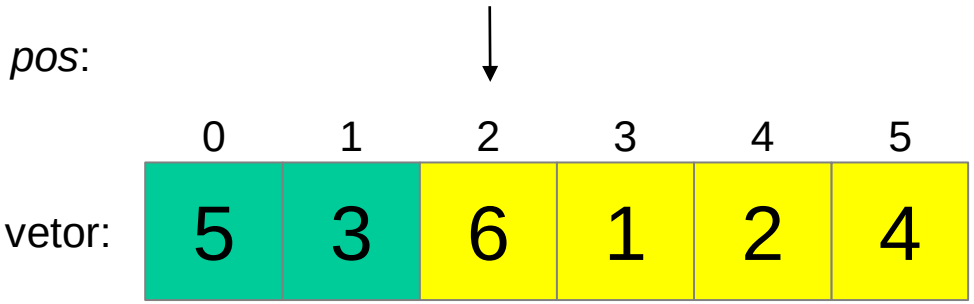
Não está vazia! Retira o topo da pilha

Pilha (Stack) – LIFO



Decrementa indicador de posição

Pilha (Stack) – LIFO

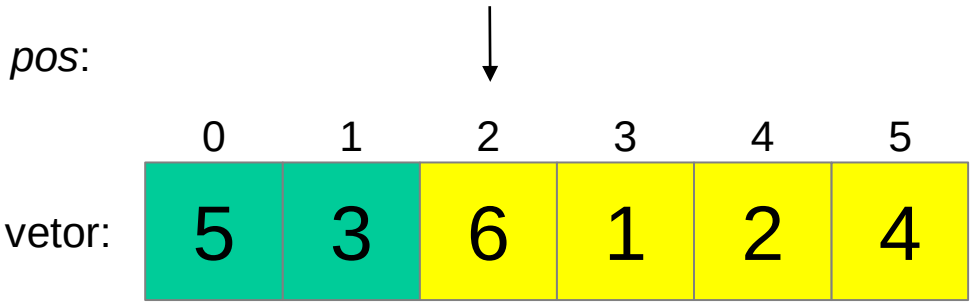



```
int topo;
:
:
topo = top( );

//topo: 3
```

Consulta o topo da pilha

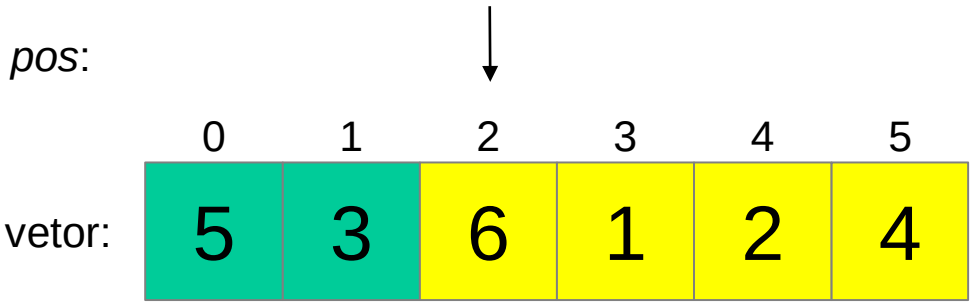
Pilha (Stack) – LIFO




```
int n;
:
:
if ( size( ) <= 0 )
    System.out.println("Pilha Vazia");
else
    n = pop( );
```

Verifica se a pilha está vazia

Pilha (Stack) – LIFO

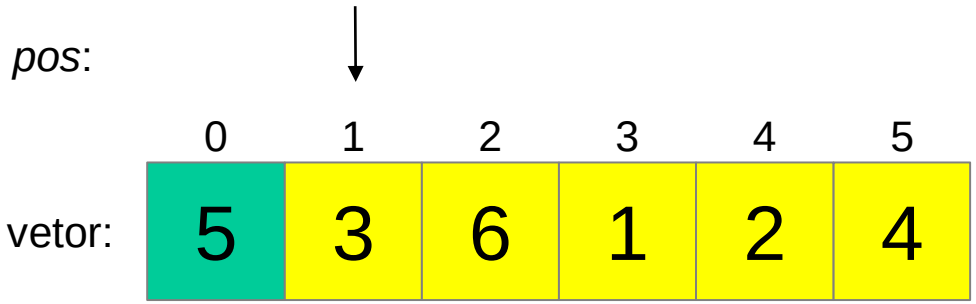



```
int n;
:
:
if ( size( ) <= 0 )
    System.out.println("Pilha Vazia");
else
    n = pop( );

    //n: 3
```

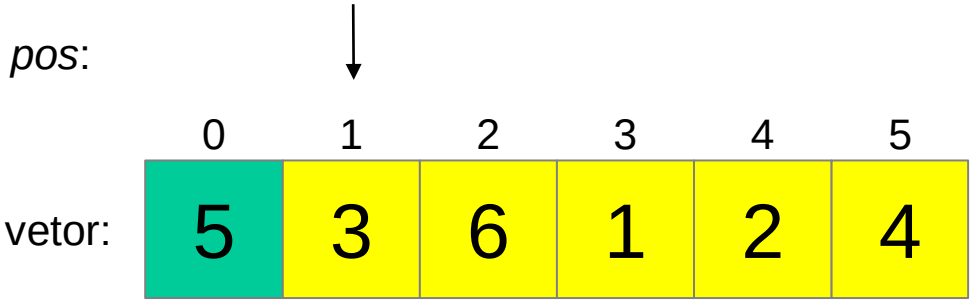
Não está vazia! Retira o topo da pilha

Pilha (Stack) – LIFO



Decrementa indicador de posição

Pilha (Stack) – LIFO

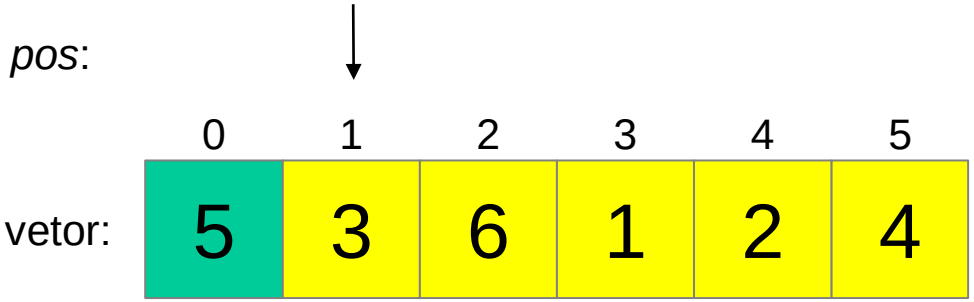



```
int topo;
:
:
topo = top( );

//topo: 5
```

Consulta o topo da pilha

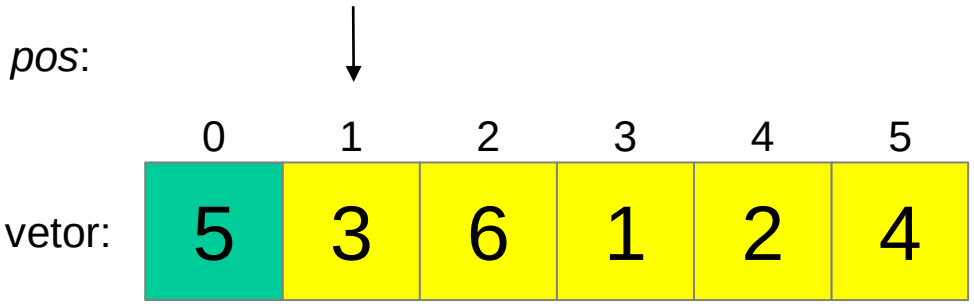
Pilha (Stack) – LIFO




```
int n;
:
:
if ( size( ) <= 0 )
    System.out.println("Pilha Vazia");
else
    n = pop( );
```

Verifica se a pilha está vazia

Pilha (Stack) – LIFO

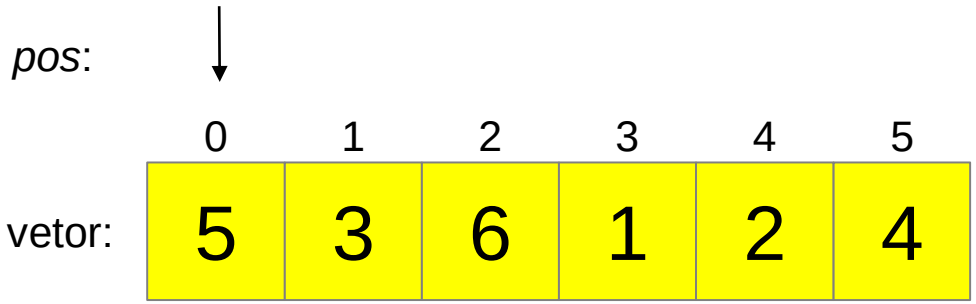



```
int n;
:
:
if ( size( ) <= 0 )
    System.out.println("Pilha Vazia");
else
    n = pop( );

    //n: 5
```

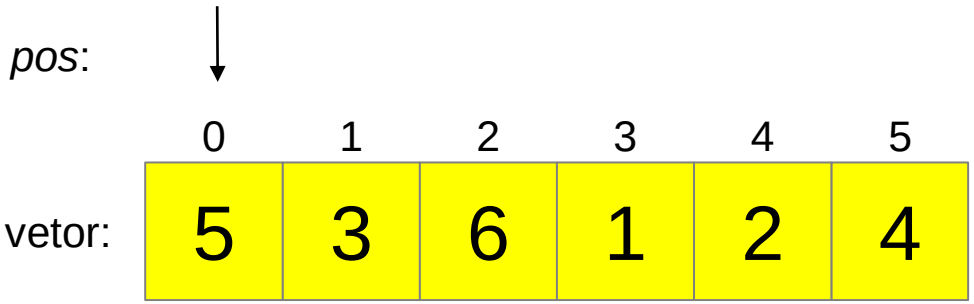
Não está vazia! Retira o topo da pilha

Pilha (Stack) – LIFO



Decrementa indicador de posição

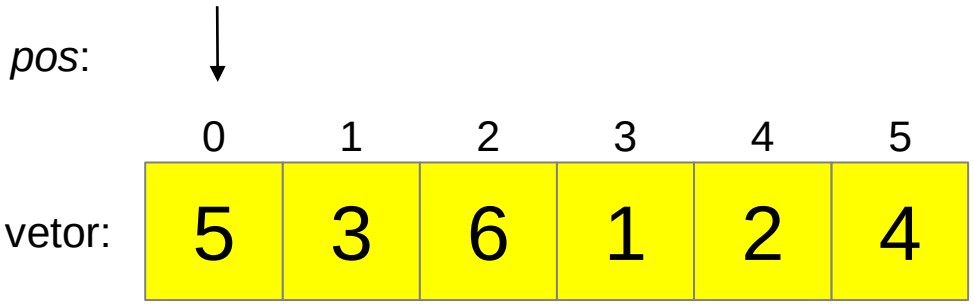
Pilha (Stack) – LIFO




```
int n;
:
:
if ( size( ) <= 0 )
    System.out.println("Pilha Vazia");
else
    n = pop( );
```

Verifica se a pilha está vazia

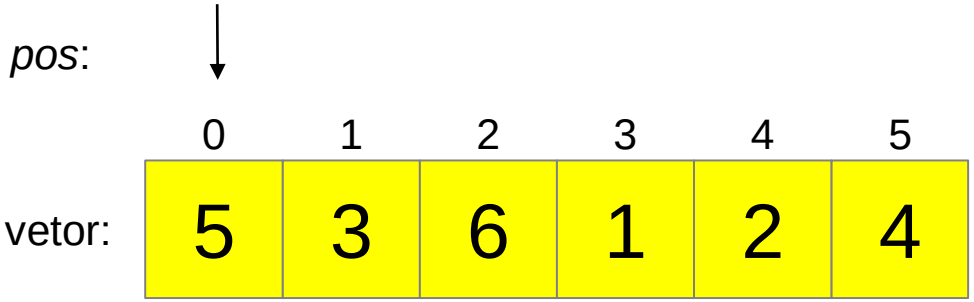
Pilha (Stack) – LIFO




```
int n;
:
:
if ( size( ) <= 0 )
    System.out.println("Pilha Vazia");
else
    n = pop( );
```

Está vazia! Não retira o topo da pilha

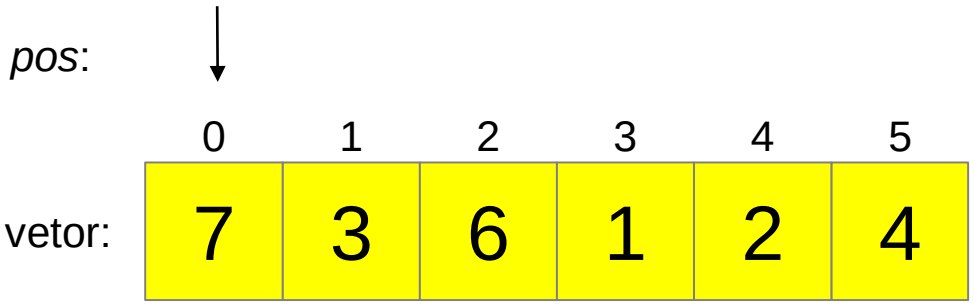
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(7);
```

Verifica se a pilha está cheia

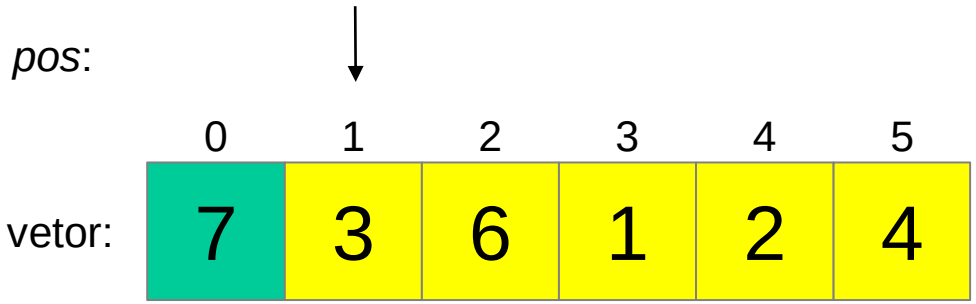
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(7);
```

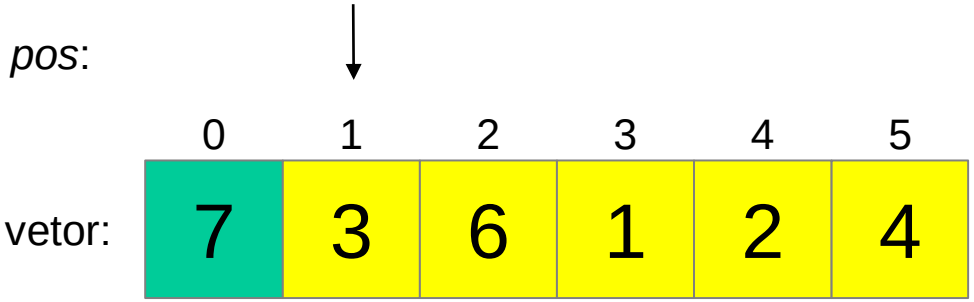
Não está cheia! Coloca o elemento 7 na pilha

Pilha (Stack) – LIFO



Incrementa indicador de posição

Pilha (Stack) – LIFO

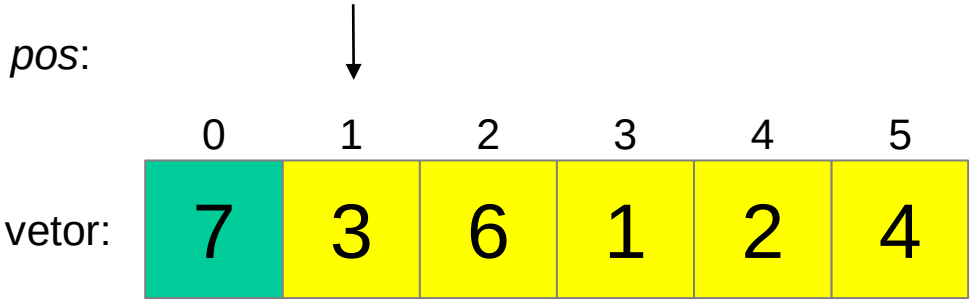



```
int topo;
:
:
topo = top( );

//topo: 7
```

Consulta o topo da pilha

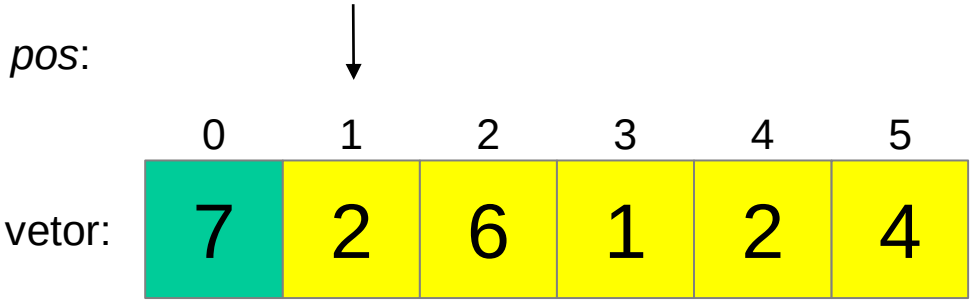
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(2);
```

Verifica se a pilha está cheia

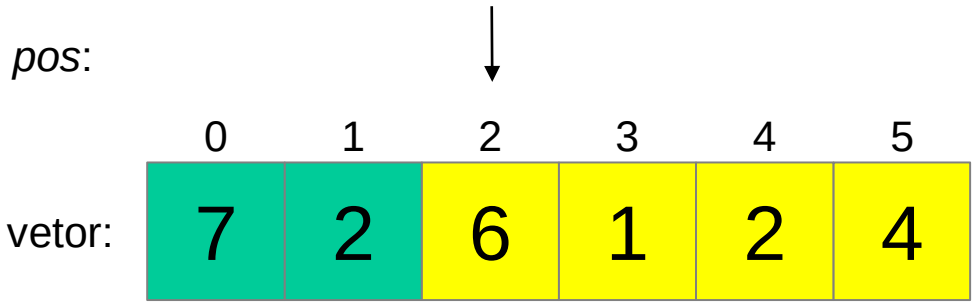
Pilha (Stack) – LIFO




```
if ( size( ) >= vetor.length( ) )
    System.out.println("Pilha Cheia");
else
    push(2);
```

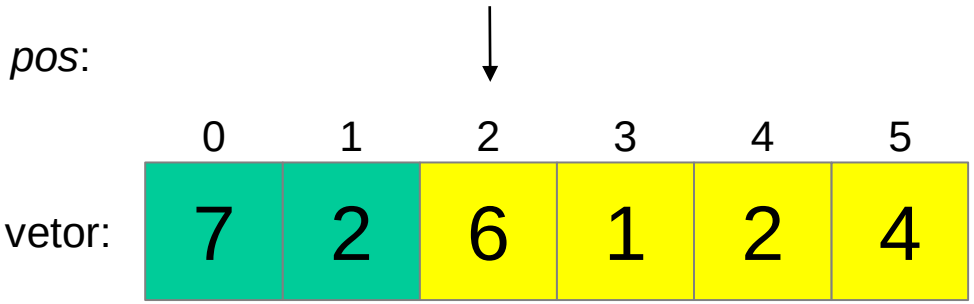
Não está cheia! Coloca o elemento 2 na pilha

Pilha (Stack) – LIFO



Incrementa indicador de posição

Pilha (Stack) – LIFO

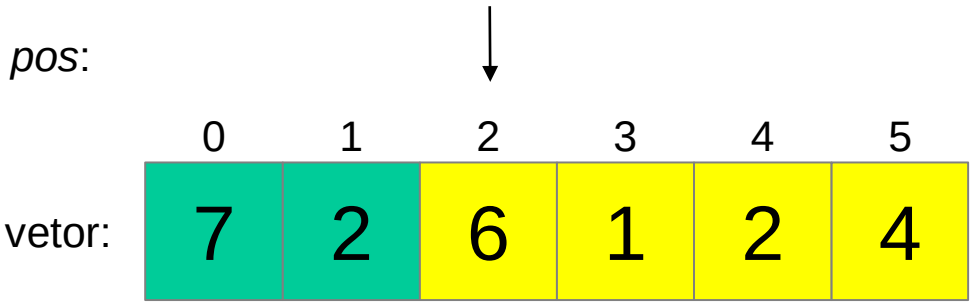



```
int topo;
:
:
topo = top( );

//topo:2
```

Consulta o topo da pilha

Pilha (Stack) – LIFO



Repete qualquer operação:
push, pop, size e top,
conforme necessidade...

Insere ou retira elementos da pilha...

Pilha (Stack) – LIFO

Exemplo de códigos escrito em Java:

Codificação dos Métodos

Pilha (Stack) – LIFO

```
public   static   int      iTAM = 10;
public   static   int      iPilha[ ] = new   int[iTAM];
public   static   int      iPos = 0;

:
:

public   static   int  size( )
{
     return   iPos;
}

public   static   int  top( )
{
     return   iPilha[iPos-1];
}
```

Métodos *size( )* e *top( )*

Pilha (Stack) – LIFO

```
public   static    int       iTAM = 10;
public   static    int       iPilha[ ] = new   int[iTAM];
public   static    int       iPos = 0;
:
:

public   static    int   push( int iC)
{
     if( iPos >= iTAM)     return   0;
     return   iPilha[ iPos++ ] = iC;
}

public   static    int   pop()
{
     if( iPos == 0)     return   0;
     return   iPilha[ --iPos ];
}
```

Método *push( )* e *pop( )*

Pilha (Stack) – LIFO

FIM