# Estrutura Abstrata de Dados

# PILHA

## Implementação com ligações simples

## (usando ponteiros e memória dinâmica)

# Conceitos gerais

## Elementos

- Os elementos duma **Pilha** (*Stack*) são processados por ordem inversa à da chegada
  - o **último** elemento a **entrar** é o **primeiro** a **sair** (**LIFO** - "Last In First Out")

## Operações

- Qualquer operação a realizar sobre uma Pilha será efetuada no seu **topo**
  - **inserção** de um (novo) elemento
    - acrescenta um elemento ao topo da Pilha (e torna-se o topo)
  - **remoção** de um elemento
    - retira o elemento que se encontra no topo da Pilha
    - passa o elemento que está antes, caso exista, a ser o topo de Pilha
    - só é possível de realizar se a Pilha não estiver vazia
  - **pilha vazia**
    - acontece quando é removido o último elemento da Pilha
    - só é possível realizar a operação de inserção na Pilha

# Operações básicas (únicas) sobre uma EAD PILHA

## Criar uma Pilha (vazia)

S = **criarPilha** ()

## Verificar se uma Pilha está vazia

**pilhaVazia** (S)

## Inserir um elemento numa Pilha (coloca no topo da Pilha)

S = **push** (X, S)

## Remover um elemento de uma Pilha (o que está no topo)

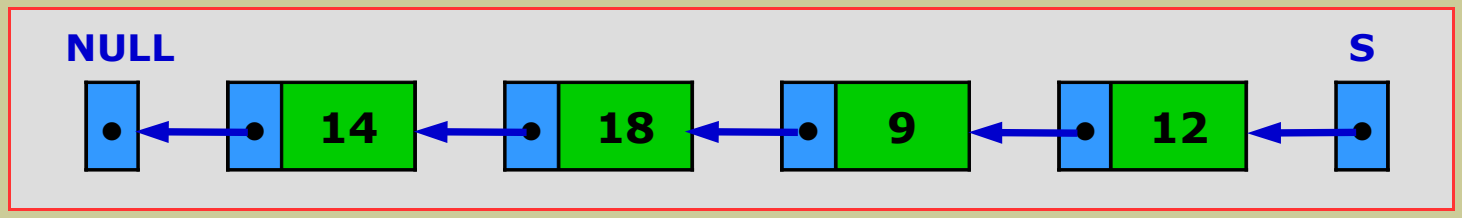
S = **pop** (S)

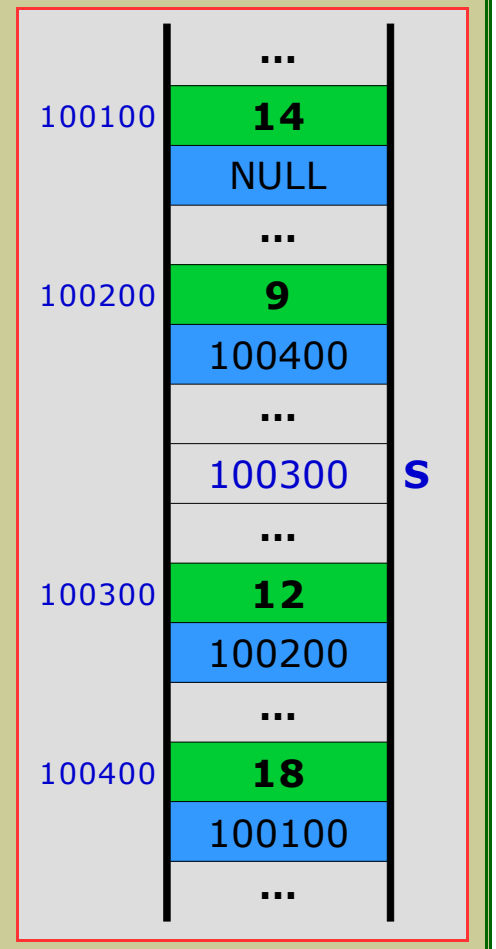
## Consultar uma Pilha (devolve o elemento do topo)

X = **topo** (S)

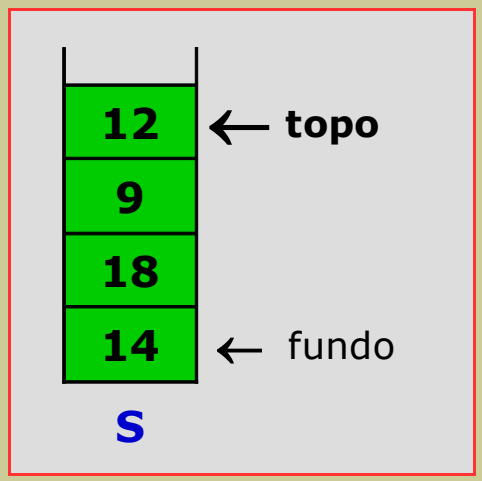
# Representação gráfica

## EAD Pilha




## Analogia



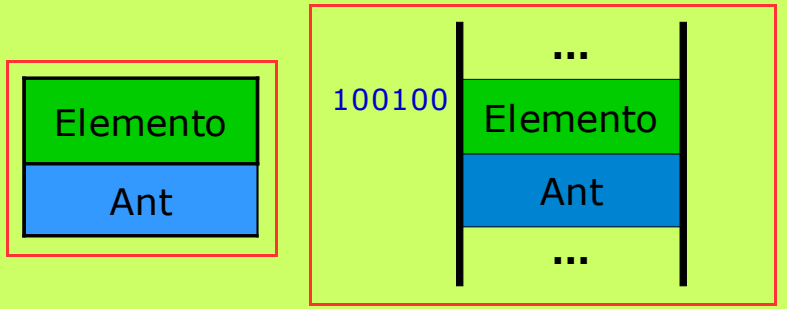

## Identificador S

- É um **ponteiro** para o **topo** da **Pilha**

# Nodos e ligações

## Definição de um nodo de uma AED PILHA (em linguagem C)

```
struct NodoPilha {
    DadosPilha Elemento;
    struct NodoPilha *Ant;
};
typedef struct NodoPilha *PNodoPilha;
```




- Cada nodo aponta para o nodo anterior da Pilha
- O nodo do fundo da Pilha aponta para NULL
- A memória para os nodos é
  - atribuída quando um elemento é inserido na Pilha
  - libertada quando um elemento é removido da Pilha.
- Pilha vazia: quando o topo da Pilha é um ponteiro nulo (NULL)
- Pilha cheia: quando não há memória para alocar um novo elemento

# Operações sobre a estrutura DadosPilha

## Mostrar os dados de um elemento do tipo DadosPilha

void **mostrarElementoPilha** (DadosPilha X)

- operação que mostra/guarda os dados/informação do elemento X do tipo DadosPilha

## Criar um elemento do tipo DadosPilha

DadosPilha **criarElementoPilha** ()

- operação que cria um elemento do tipo DadosPilha com dados vindos de fonte adequada

## Comparar dois elementos do tipo DadosPilha

int **CompararElementosPilha** (DadosPilha X, DadosPilha Y)

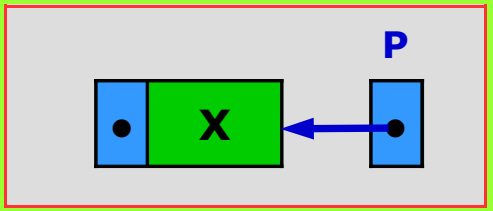
- operação que compara dois elementos do tipo DadosPilha (segundo o campo chave)
- devolve: **-1** (X < Y), **0** (X = Y) ou **1** (X > Y)

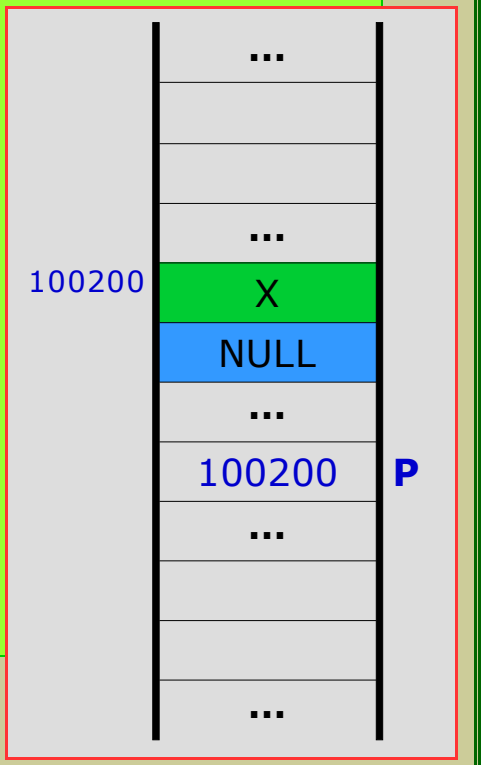
# Operações básicas sobre um nodo

## Criar um nodo de uma Pilha

Operação: criar um novo nodo de uma Pilha
Entrada: a informação propriamente dita (elemento X), que fará parte de um nodo
Saída: um ponteiro para um **nodo** com informação (elemento X) e ligação a NULL

```
PNodoPilha criarNodoPilha (DadosPilha X)
{
  PNodoPilha  P;
  P = (PNodoPilha) malloc (sizeof(struct NodoPilha));
  if (P == NULL)
     return NULL;
  P→Elemento = X;
  P→Ant = NULL;
  return P;
}
```

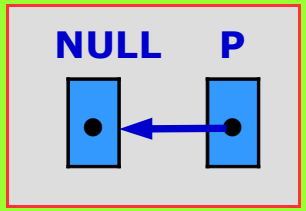


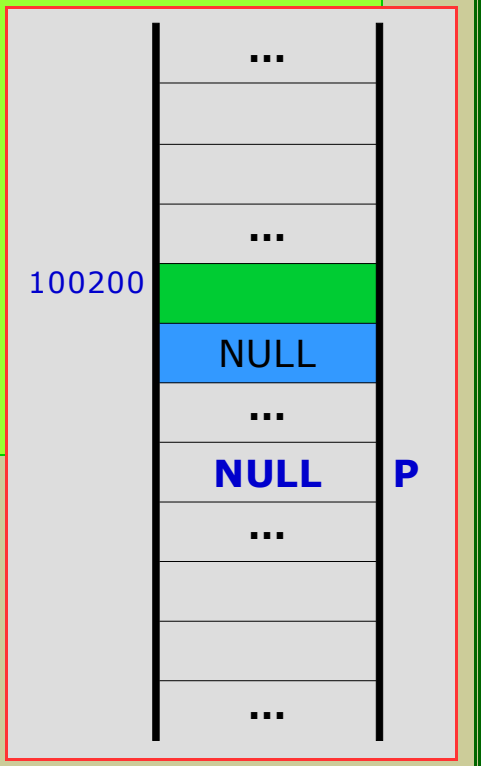



# Libertar/destruir um nodo de uma Pilha

Operação: libertar todos os nodos de uma Pilha
Entrada: um ponteiro para o nodo que se pretende destruir
Saída: o ponteiro a apontar para NULL

```
PNodoPilha libertarNodoPilha (PNodoPilha P)
{
  P→Anterior = NULL;
  free(P);
  P = NULL;
  return P;
}
```



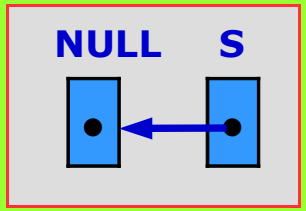



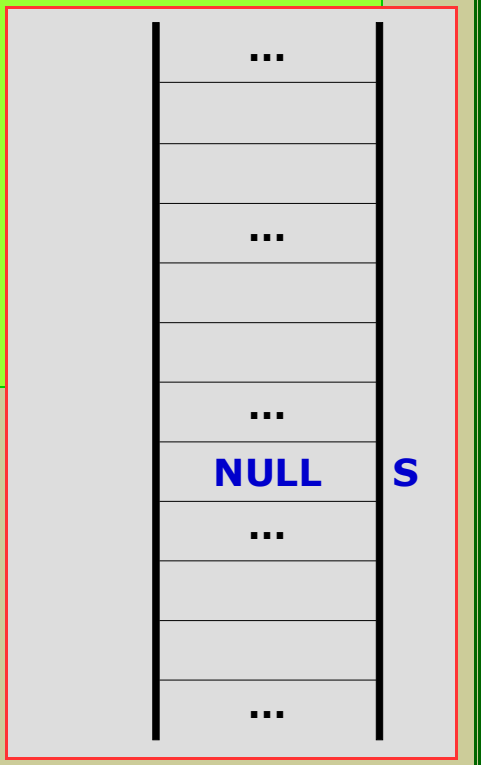
# Operações básicas (únicas) sobre uma Pilha

## Criar uma Pilha

Operação: criar uma Pilha vazia (**criarPilha**)
Entrada: nada
Saída: devolve uma Pilha S vazia (sem elementos)

```
PNodoPilha criarPilha ()
{
  PNodoPilha S;
  S = NULL;
  return S;
}
```



## Verificar se uma Pilha está vazia

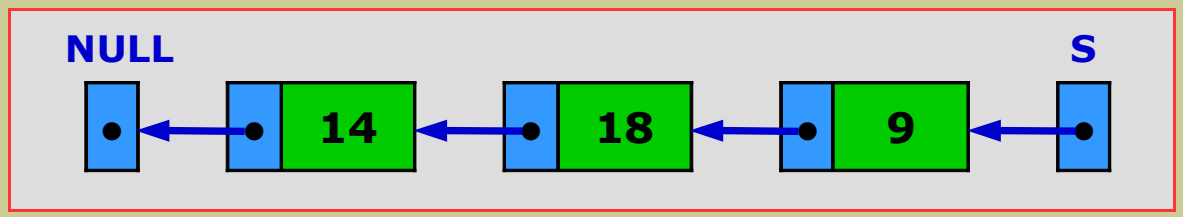
Operação: verificar se uma Pilha está vazia (**pilhaVazia**)
Entrada: uma Pilha S
Saída: 1 (se a Pilha S está vazia) ou 0 (se a Pilha S não está vazia)

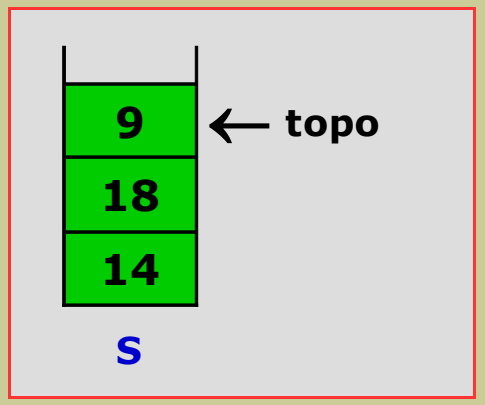
```
int pilhaVazia (PNodoPilha S)
{
  if (S == NULL)
     return 1;
  else
     return 0;
}
```

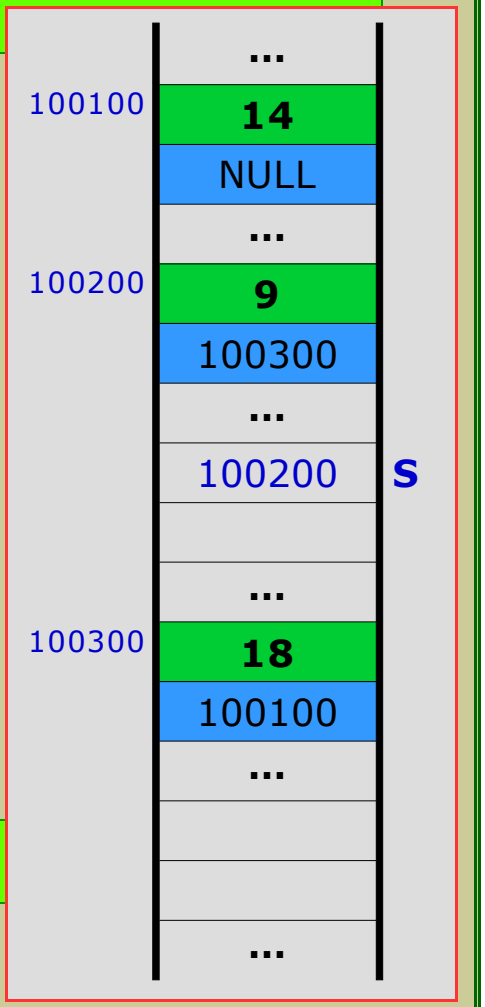
# Inserir um elemento numa Pilha

- Operação **push**: S = push(X, S)

Pilha S antes da operação



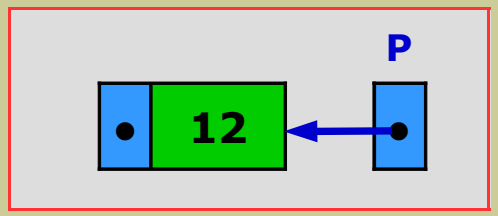

Inserir na Pilha S o elemento **12**

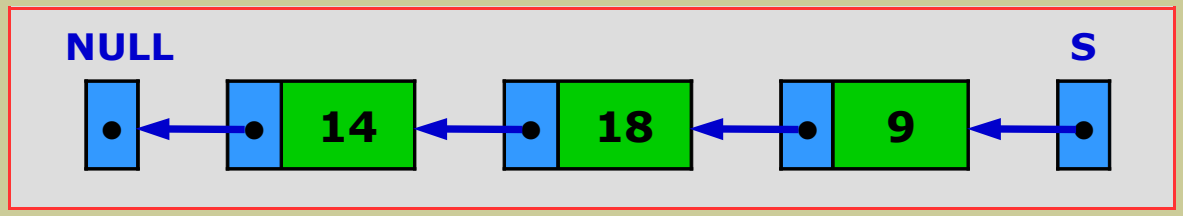
# Inserir um elemento numa Pilha

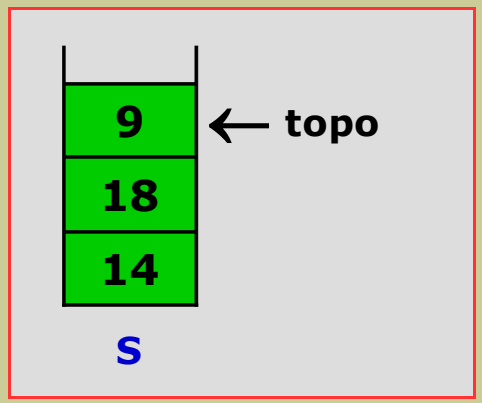
- Operação **push**: S = push(X, S)

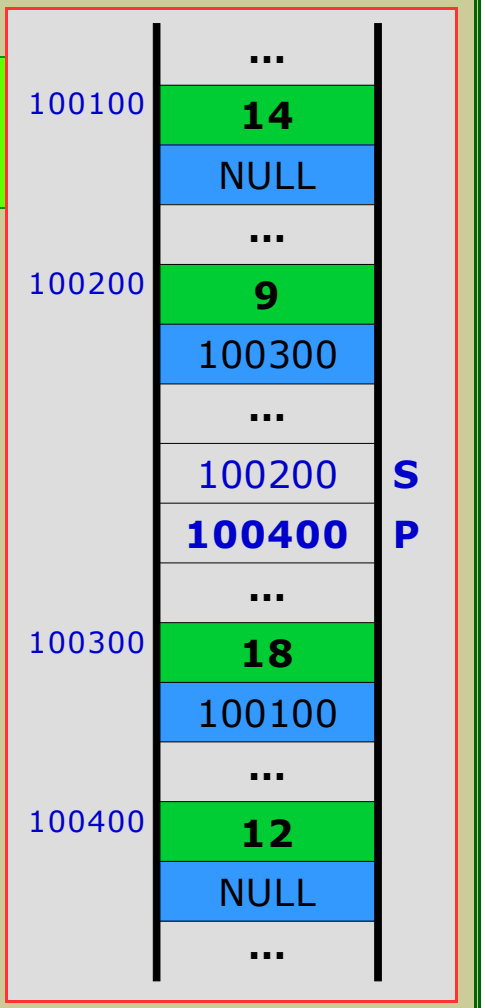
**Passo 1**

**criar** um nodo com o elemento **12** (novo), apontado por **P**



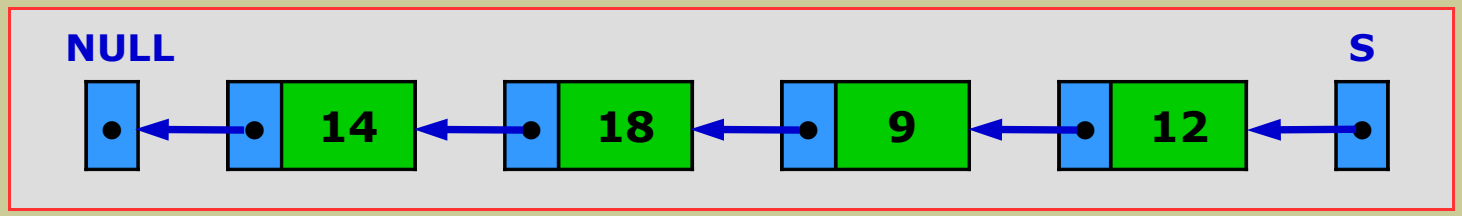



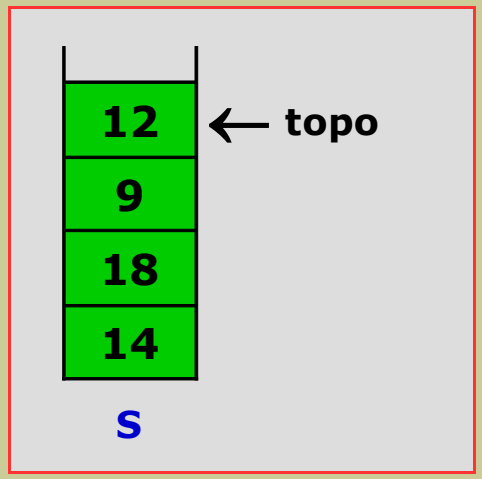
# Inserir um elemento numa Pilha

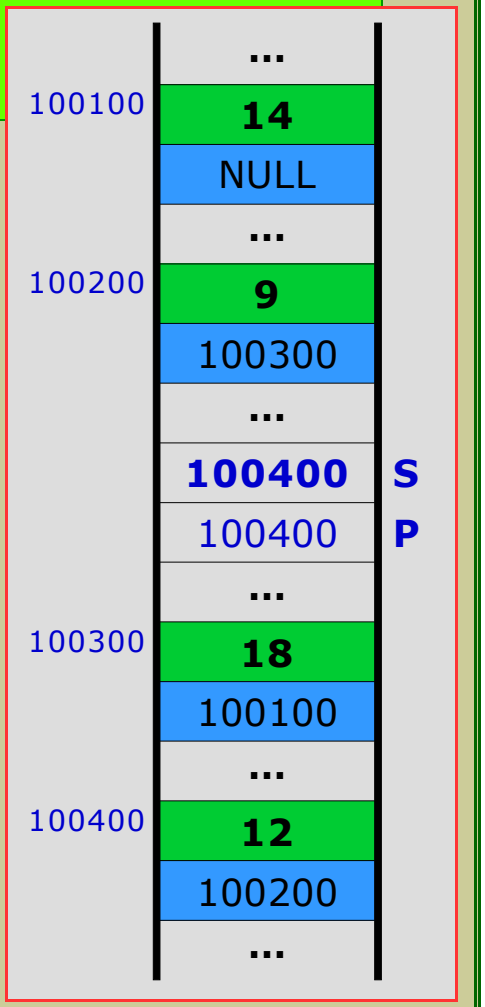
- Operação **push**: S = push(X, S)

**Passo 2**

**inserir** o nodo com o elemento **12** (novo) na Pilha S



## Inserir um elemento numa Pilha

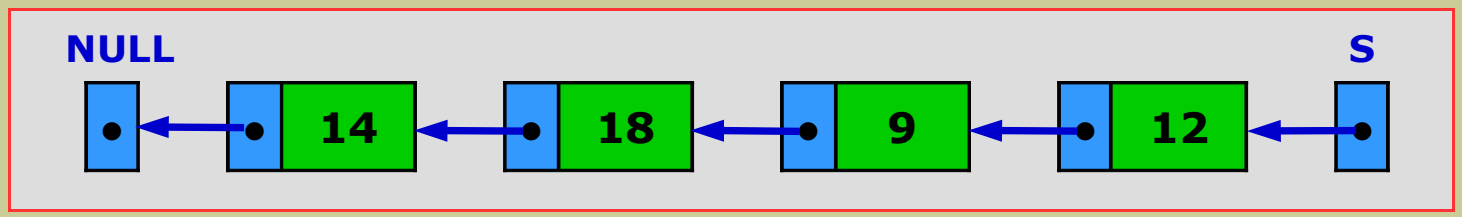
Operação: inserir um elemento no topo de uma Pilha (**push**)
Entrada: um elemento X a inserir e uma Pilha S
Saída: a Pilha S atualizada (com mais um elemento)

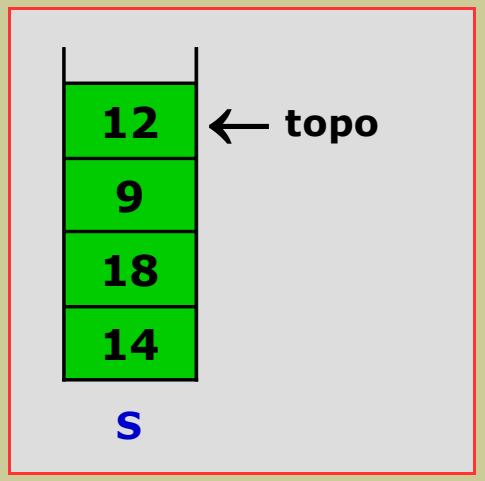
```
PNodoPilha push (DadosPilha X, PNodoPilha S)
{
  PNodoPilha  P;
  P = criarNodoPilha(X);      // passo 1
  if (P == NULL)
     return S;
  P→Ant = S;    // passo 2
  S = P;
  return S;
}
```

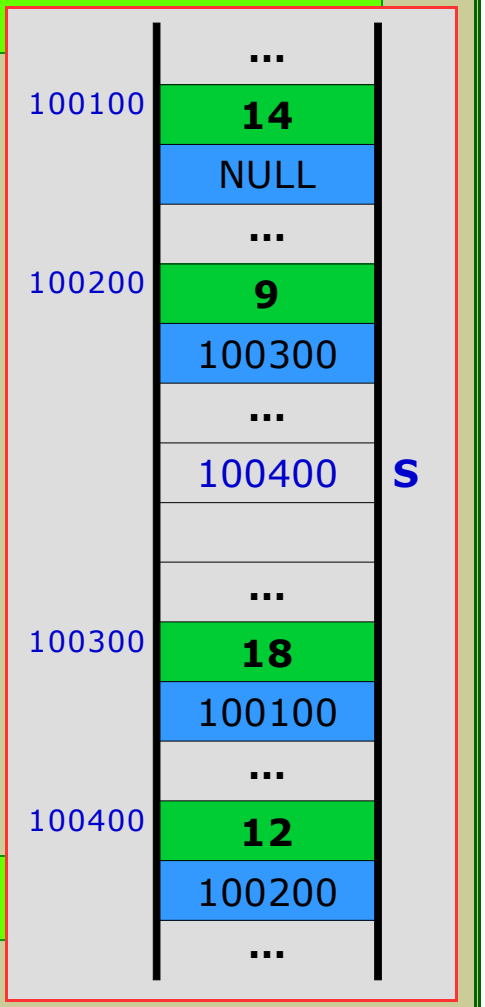
# Remover um elemento de uma Pilha

- Operação **pop**: S = pop(S)

Pilha S antes da operação



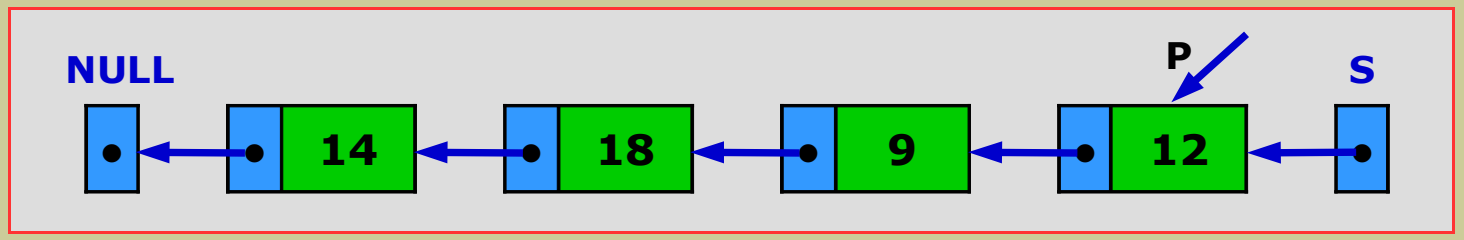

Remover elemento da Pilha S (remove o elemento do topo)

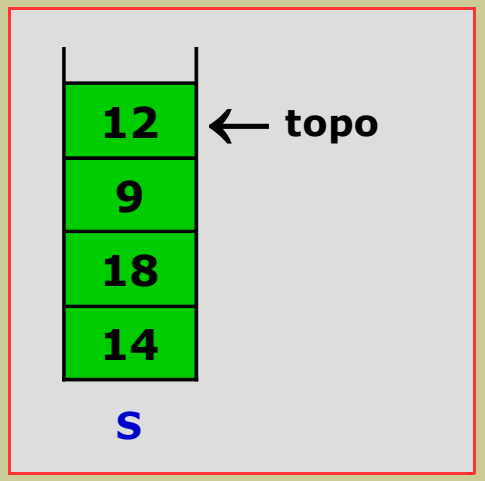
# Remover um elemento de uma Pilha

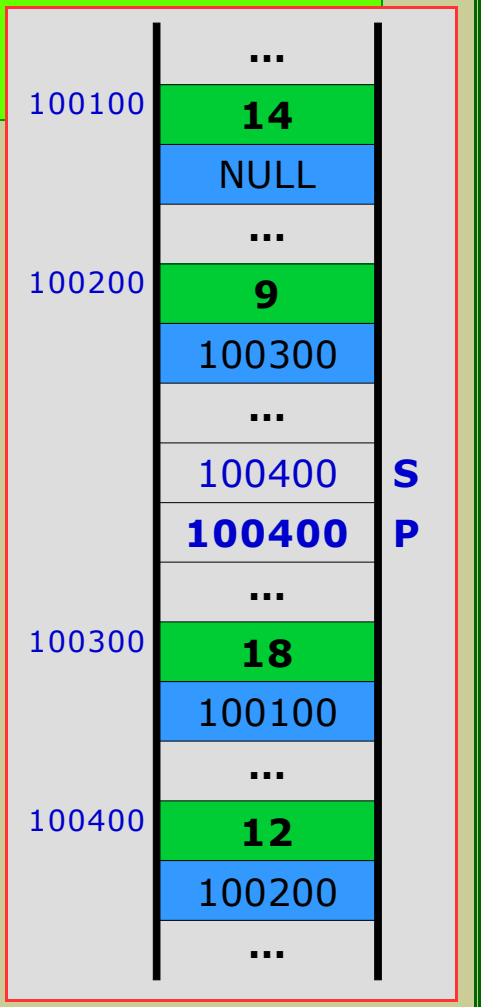
- Operação **pop**: S = pop(S)

**Passo 1**

**marcar** o nodo do topo da Pilha S com o ponteiro **P**



# Remover um elemento de uma Pilha

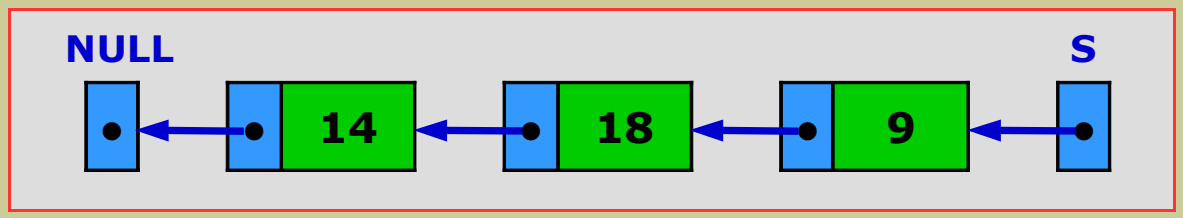
- Operação **pop**: S = pop(S)

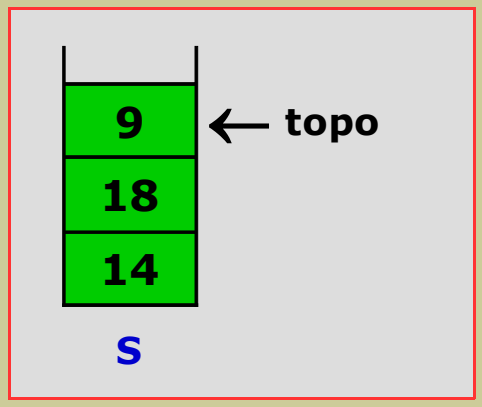
**Passo 2**

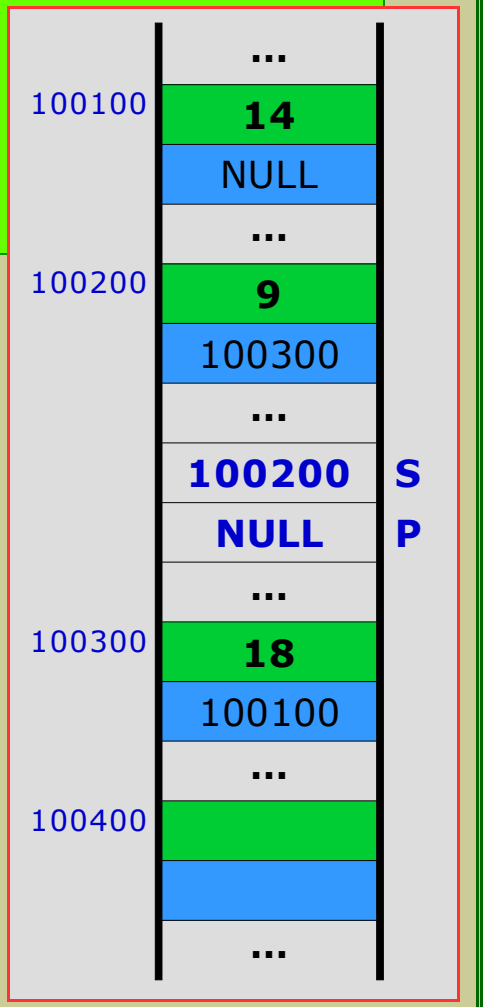
**remover** o topo de S, que passa a ser o nodo anterior

**Passo 3**

**libertar** o nodo com o topo anterior de S (P)



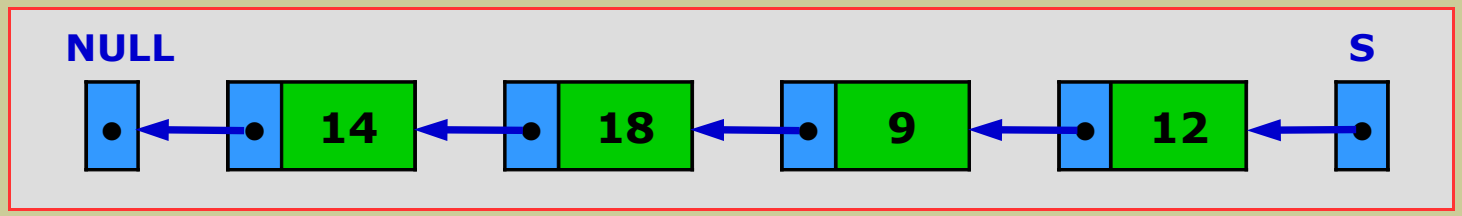
## Remover um elemento de uma Pilha

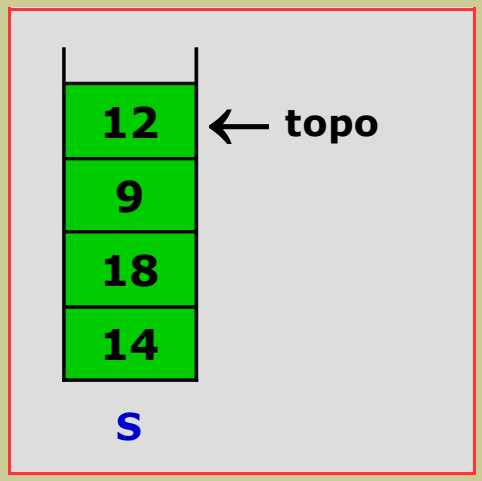
Operação: remove o elemento do topo de uma Pilha (**pop**)
Entrada: uma Pilha S
Saída: a Pilha S atualizada (sem o elemento do topo)

```
PNodoPilha pop (PNodoPilha S)
{
  PNodoPilha P;
  P = S;
  S = S→Ant;
  P = libertarNodoPilha(P);
  return S;
}
```

# Consultar uma Pilha

- Representação gráfica




topo(S) = **12** (S→Elemento)

## Consultar uma Pilha

Operação: consulta o elemento do topo de uma Pilha (**topo**)
Entrada: uma Pilha S
Saída: o elemento que está no topo da Pilha S (do tipo DadosPilha)

```
DadosPilha topo (PNodoPilha S)
{
  return S→Elemento;
}
```