FaculdadeBarretos
www.faculdadebarretos.com.br

Algoritmos e Estruturas de Dados

(**Pilhas** - Implementação e Uso)

Prof. Me. Diogo Tavares da Silva
contato: *diogotavares@unibarretos.com.br*

Nas últimas aulas aprendemos...

- Sobre as estruturas de listas encadeadas
  - seus métodos
  - suas variações

A estrutura de Pilha (*Stack*)

- É uma subclasse da estrutura de lista
  - Na prática é uma lista encadeada...
  - PORÉM
    - Possui regras de inserção e remoção
      - **Último a entrar é o primeiro a sair**
        - **Last In - First Out (LIFO)**
    - POR QUE?
      - Para manter seu propósito e funcionalidade

A estrutura de Pilha

- A estrutura de dados de pilha é uma organização lógica de um comportamento natural no mundo real
  - Ex:
    - Pilha de pratos
    - Pilha de cartas
    - Um caminhão para descarregar
    - Um pacote de bolachas
    - etc.

A estrutura de Pilha

- Na computação são amplamente utilizadas
  - propósito:
    - Preservar os dados e acessá-los na ordem inversa a que foram obtidos
    - Preservar o histórico de ocorrências permitindo a reversão de eventos.

A estrutura de Pilha

- Na computação são amplamente utilizadas
  - Exemplo:
    - Ações de Desfazer e Refazer
    - Avançar e Voltar páginas em um navegador
    - Pilha de execução de um programa
      - pilha de recursão
    - Algoritmos de árvore e de grafos
    - Pilha de retorno do sistema android
      - https://developer.android.com/guide/components/activities/tasks-and-back-stack?hl=pt-br

A estrutura de Pilha

- Pilha de retorno do sistema android
  - https://developer.android.com/guide/components/activities/tasks-and-back-stack?hl=pt-br

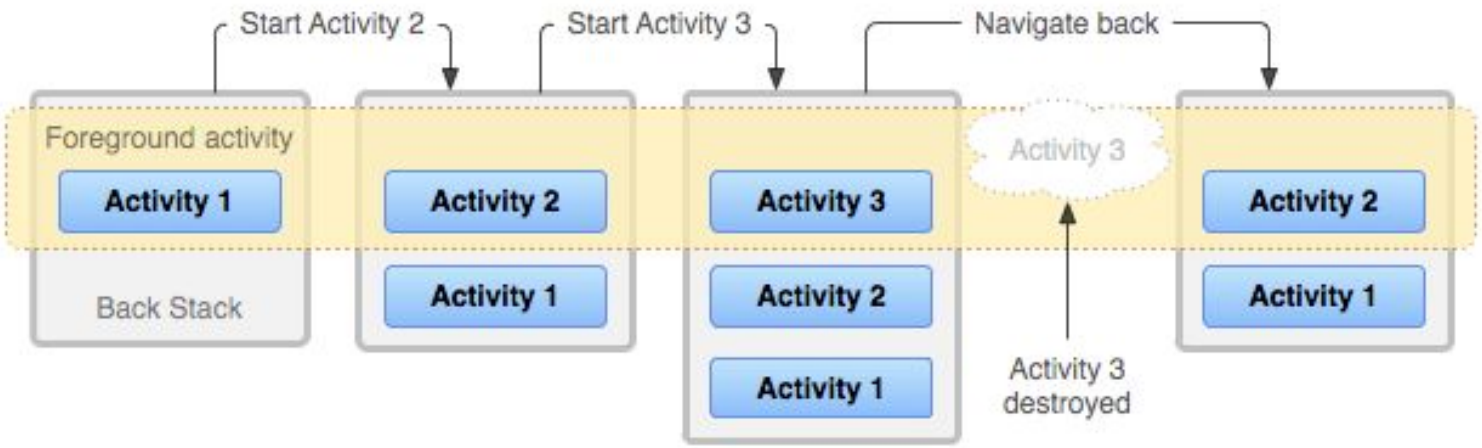
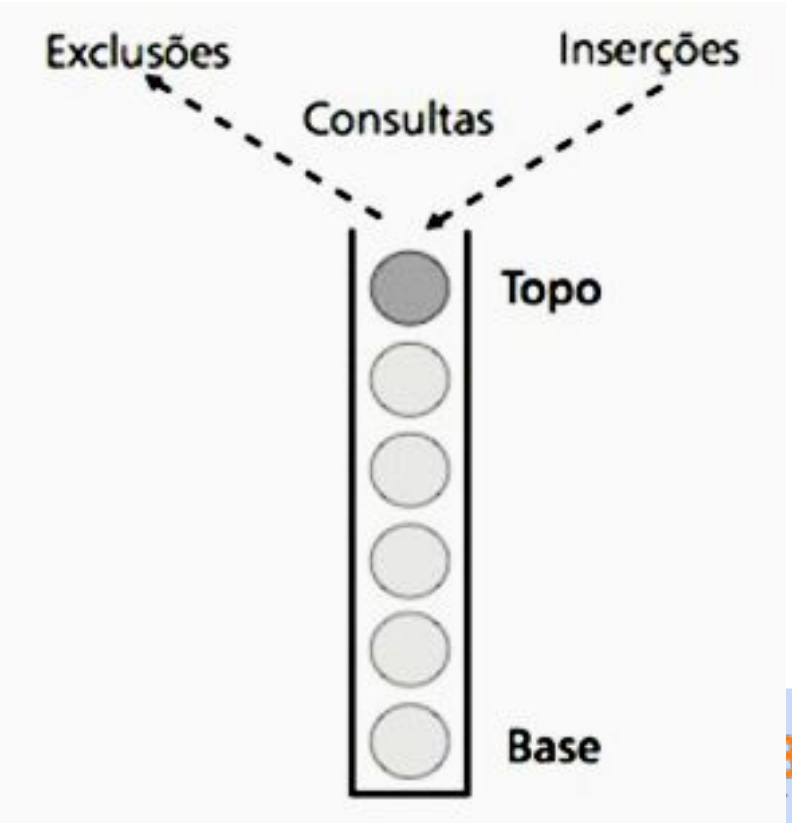


Figura 1. Uma representação de como cada nova atividade em uma tarefa adiciona um item à pilha de retorno. Quando o usuário pressiona o botão **Voltar**, a atividade atual é destruída e a anterior é retomada.

A estrutura de Pilha

- Na prática:
  - Pilha é uma lista encadeada onde a as inserções, remoções e consultas são realizadas apenas por uma extremidade da estrutura: O topo



Operações sobre pilhas

- Devem sempre respeitar a disciplina de acesso
  - Somente pelo topo!
    - Se acessa, insere, remove ou altera nós abaixo do topo **NÃO É Pilha!**
      - Cuidado na prova coleguinhas...

Operações sobre pilhas

- Deste modo, as seguintes operações são comuns para a estrutura de pilha:
  - Criar a pilha vazia
  - Verificar se a pilha está vazia
  - Inserir um novo nó no topo da pilha (*PUSH*)
  - Excluir o nó do topo da pilha (*POP*)
  - Consultar e/ou modificar o nó que está no topo da pilha (*TOP*)
  - Destruir a pilha, liberando as posições reservadas para ela.

Projeto da estrutura do tipo Pilha

- Duas abordagens:
  - Criar um tipo específico
  - Usar um ponteiro que aponta para o topo
    - Assim como o ponteiro na lista que aponta para a cabeça

Projeto da estrutura do tipo Pilha

- Duas abordagens:
  - Criar um tipo específico
  - Usar um ponteiro que aponta para o topo
    - Assim como o ponteiro na lista que aponta para a cabeça

- Vamos usar a primeira abordagem!

Projeto da estrutura nó de pilha

- Deve conter a informação armazenada e um ponteiro para o próximo nó
  - Ou seja, um nó comum de lista.

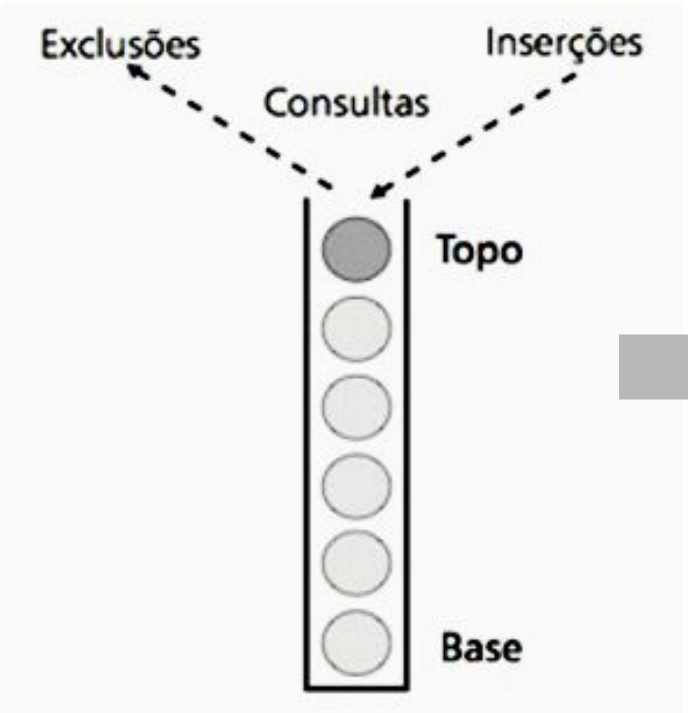
```
struct nodo{
   int info; //informação
   struct nodo *prox; //ponteiro para o próximo conteúdo da pilha
};
typedef struct nodo Nodo;
```

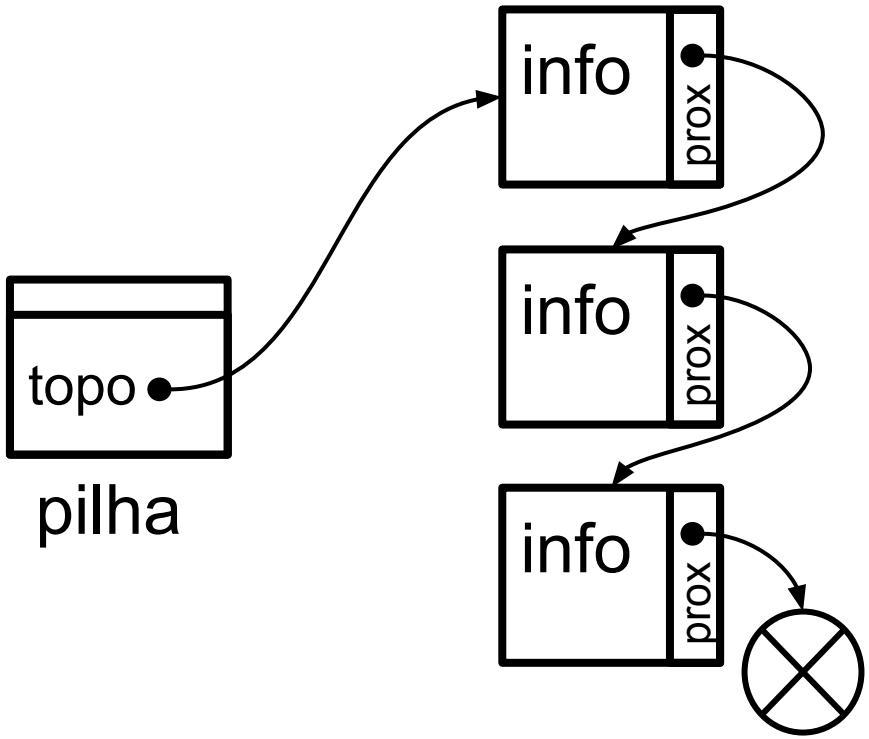
Projeto da estrutura de Pilha

- Após construir o nó:

```
struct pilha{
   Nodo *topo; //nó para o topo da pilha
}
typedef struct pilha Pilha;
```

Projeto da estrutura de Pilha



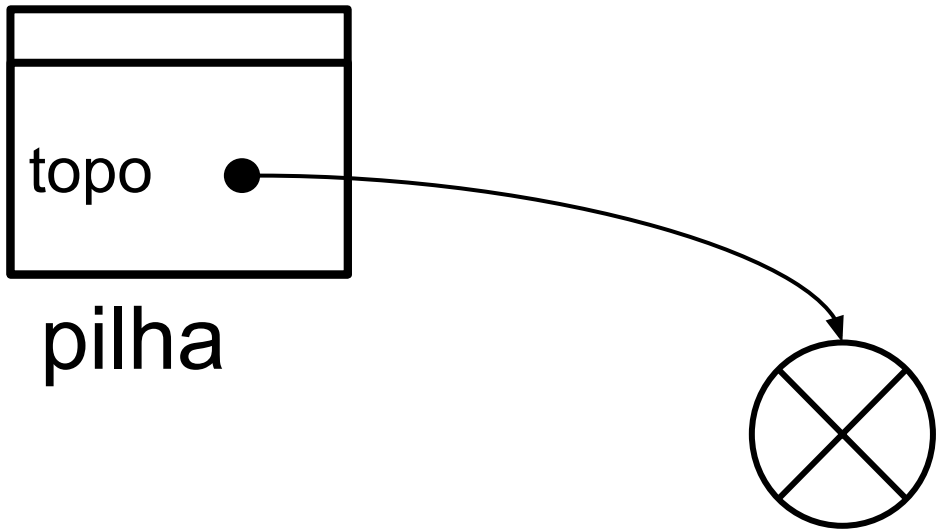


Criação da Pilha vazia

- Para criar a pilha vazia é necessário criar uma função ***criaPilha()***
  - Aloca o espaço da estrutura de pilha na memória
  - Inicializa o ponteiro "*topo*" apontando para NULL
    - A lista começa vazia.
  - Retorna um ponteiro para a estrutura de pilha criada

Criação da Pilha vazia

- função ***criaPilha()***



Criação da Pilha vazia

- Função ***criaPilha()***

```
/*
FUNÇÃO: criaPilha
RESUMO: Cria uma nova pilha vazia
PARAM: void
RETORNO: Pilha* (ponteiro para a pilha);
*/
Pilha* criaPilha(){
    Pilha* p = new Pilha;
    p->topo = NULL;
    return p;
}
```

Função de teste *pilhaVazia()*

- Mais uma vez, criaremos uma função para verificar se a pilha está vazia ou não
- Por essa razão, é conveniente criar uma função que testa se a lista está vazia
  - **int pilhaVazia(Pilha* p)**
    - **retorna 1, se lista vazia**
    - **retorna 0, caso contrário**

Criação da função *pilhaVazia()*

*//lembre-se de inserir TODOS os protótipos de funções no **pilha.h***

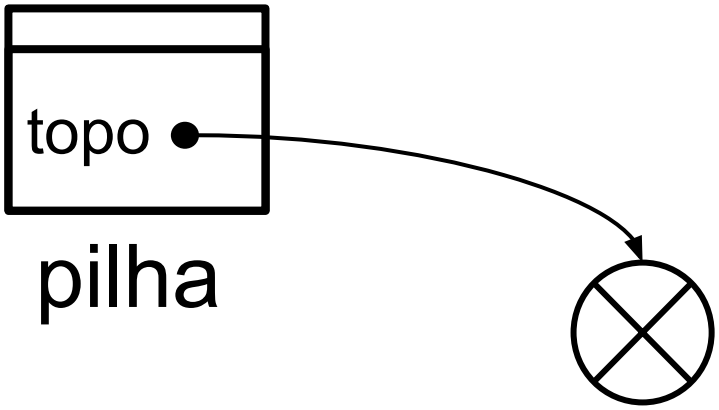
```
/*
FUNÇÃO: vazia
RESUMO: verifica se a pilha está vazia
PARAM: Pilha* (ponteiro para a pilha)
RETORNO: int (1 se está vazia, 0 se não está)
*/
int vaziaPilha(Pilha* pilha){
    if( pilha->topo == NULL)
        return 1;
    else return 0;
}
```

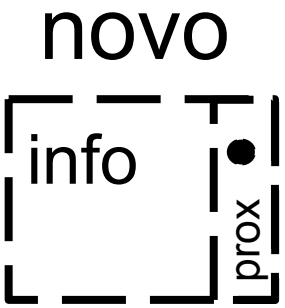
Função de empilhar *(PUSH)*

- A função de empilhar (*push*) deve fazer uma inserção no topo da pilha.
  - Análogo a inserir no início da lista
    1. Cria-se no novo nó
    2. novo -> prox recebe pilha->topo
    3. pilha->topo recebe novo

Função de empilhar (push)

- função ***push()*** ou ***empilha()***

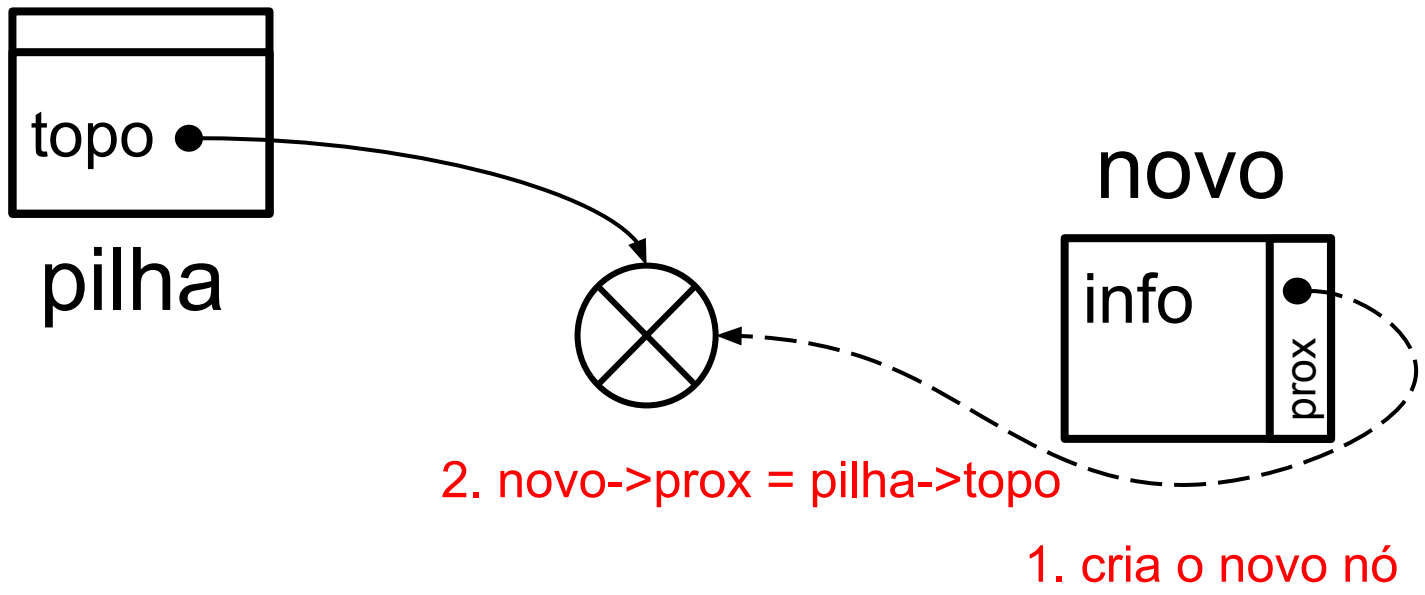




1. cria o novo nó

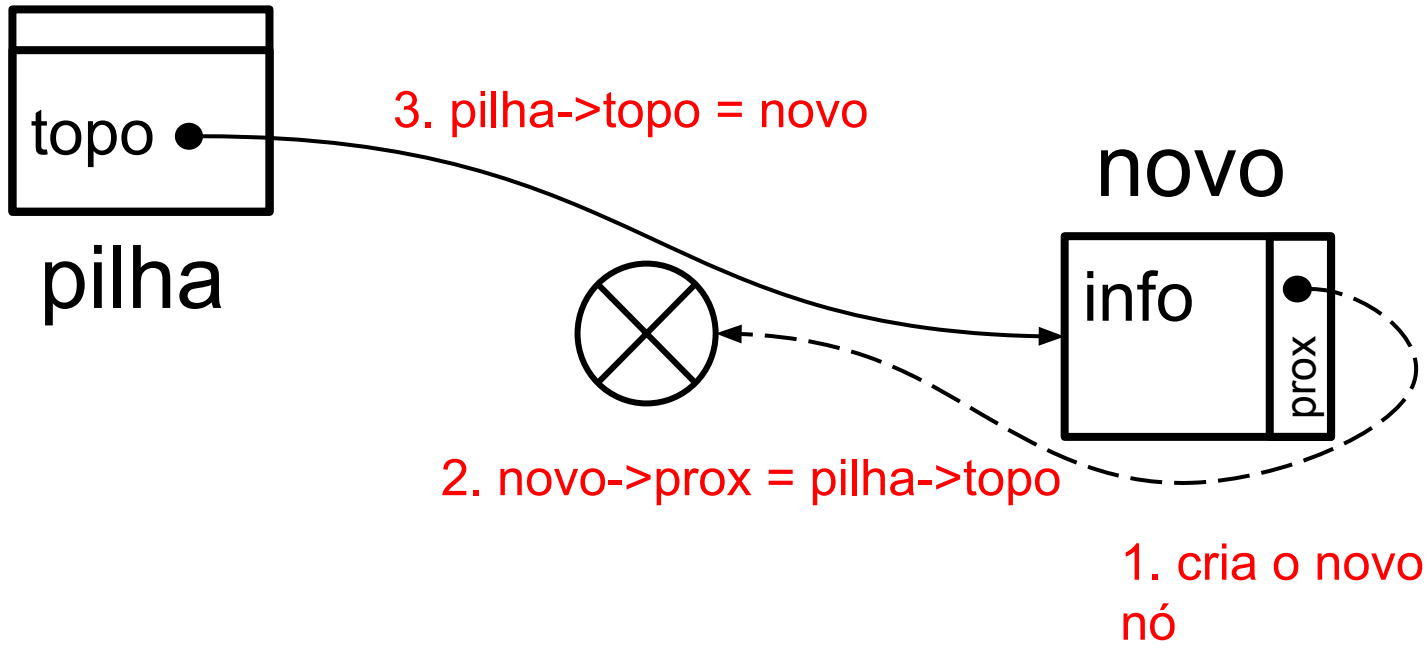
Função de empilhar (push)

- função ***push()*** ou ***empilha()***



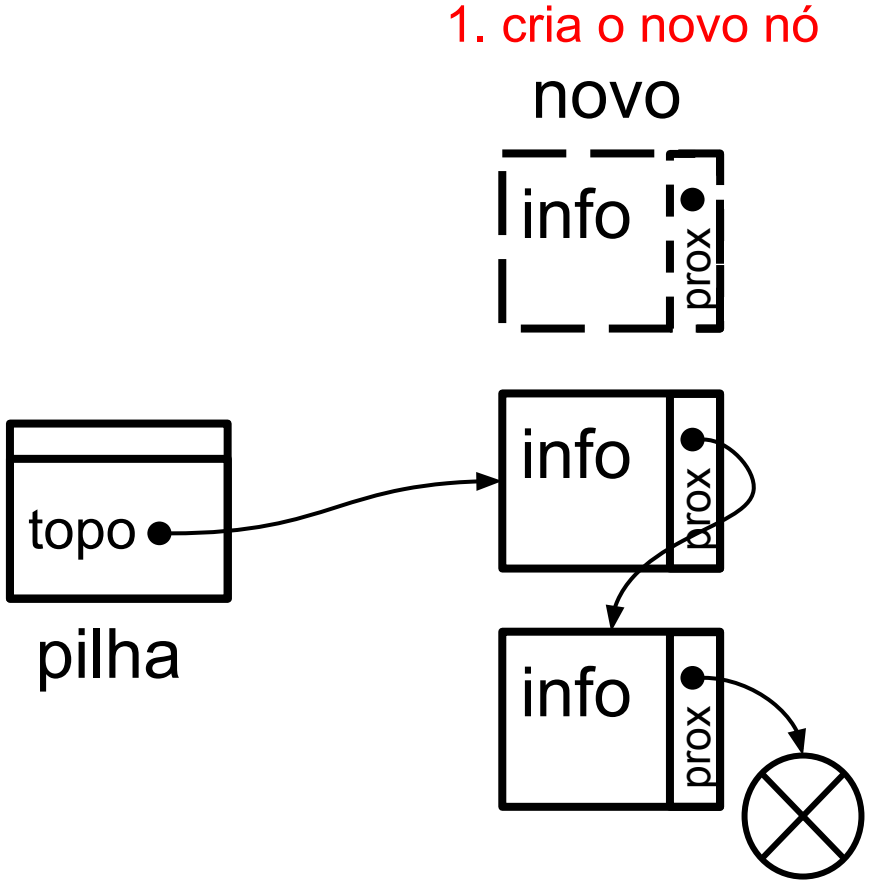
Função de empilhar (push)

- função ***push()*** ou ***empilha()***



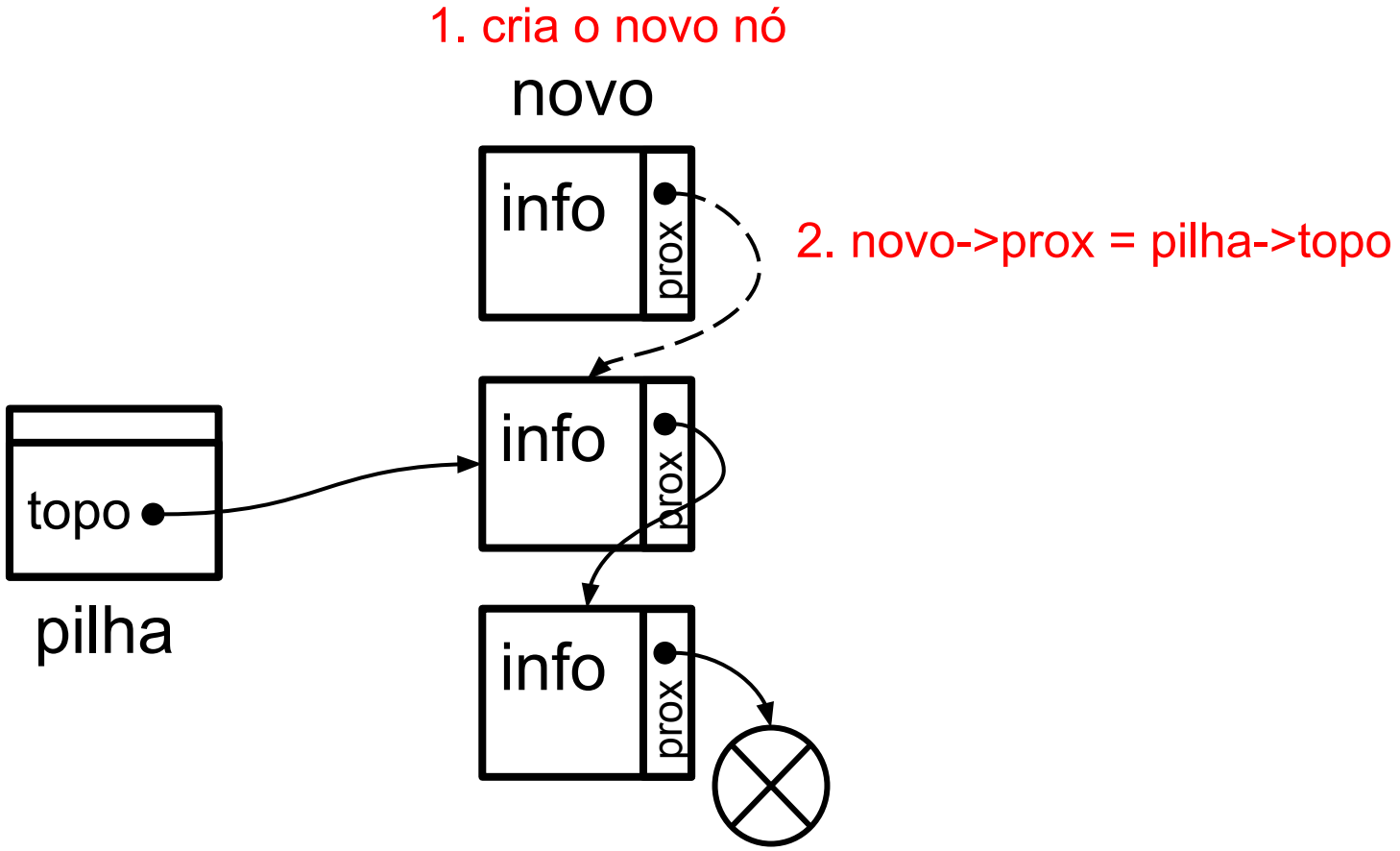
Função de empilhar (push)

- função ***push()*** ou ***empilha()***



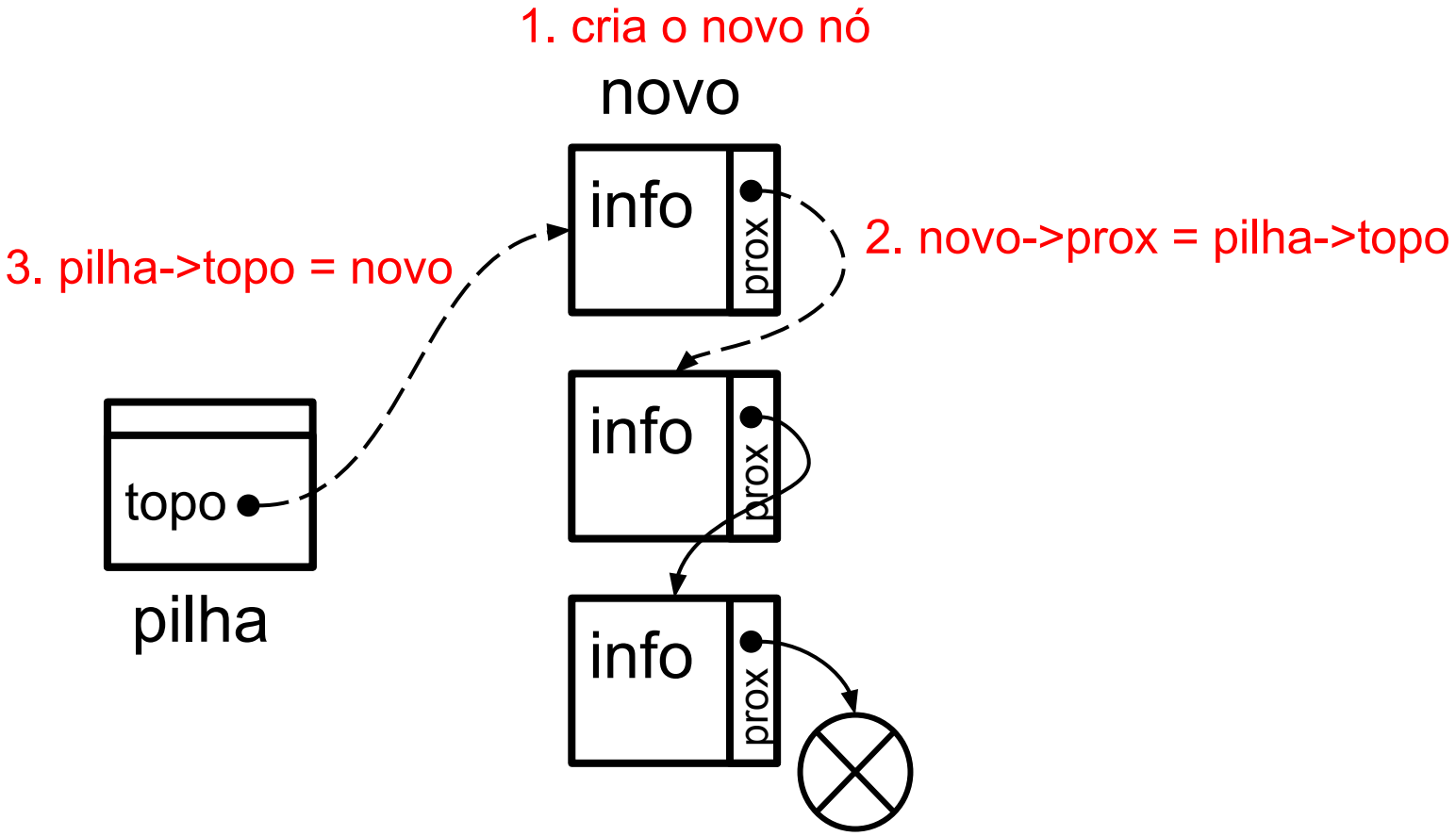
Função de empilhar (push)

- função ***push()*** ou ***empilha()***



Função de empilhar (push)

- função ***push()*** ou ***empilha()***



Função de empilhar *(PUSH)*

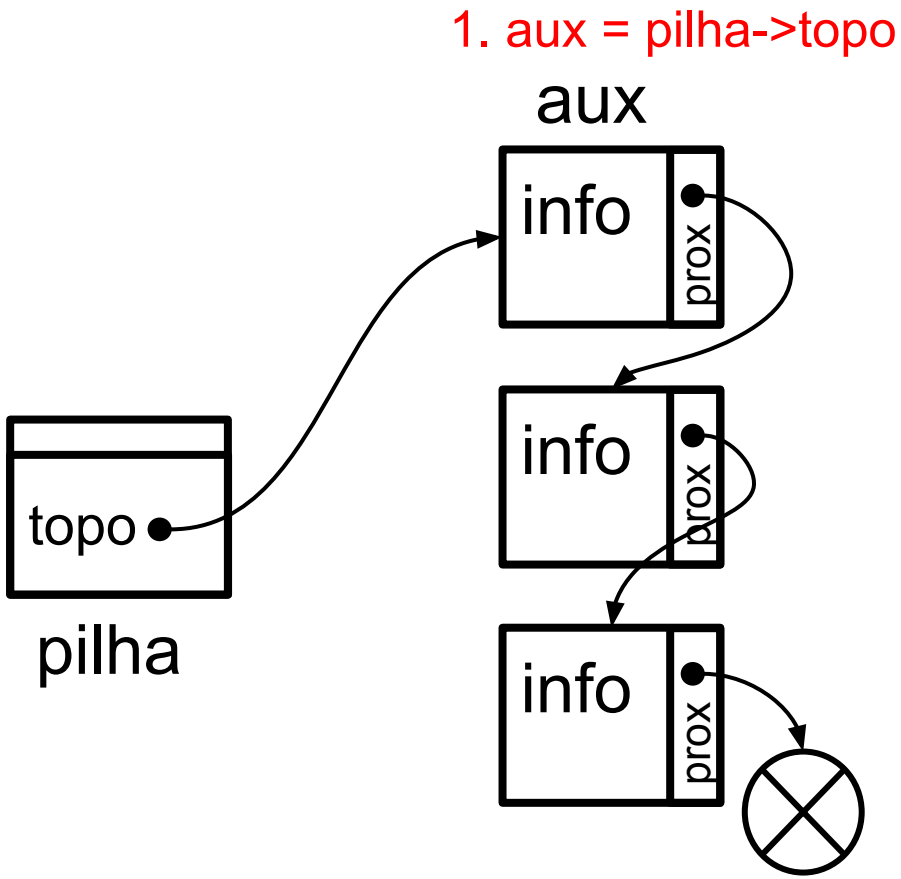
```
/*
FUNÇÃO: push
RESUMO: empilha novo elemento na pilha
PARAM: Pilha* (ponteiro para a pilha), int (valor a ser empilhado);
RETORNO: Pilha* (ponteiro para a pilha);
*/
Pilha* push(Pilha* pilha, int valor){
    Node* novo = new Node;
    novo->info = valor;
    novo->prox = pilha->topo;
    pilha->topo = novo;
    return pilha;
}
```

Função de desempilhar *(POP)*

- A função de desempilhar (*pop*) deve fazer uma remoção do elemento no topo da pilha retornando seu conteúdo.
  - Análogo a remover no início da lista
  - Se a pilha não é vazia
    1. aux recebe pilha->topo
       1.1. valor = aux->info;
    2. pilha->topo recebe (pilha->topo)->prox
    3. liberar(aux)
    4. retornar valor;

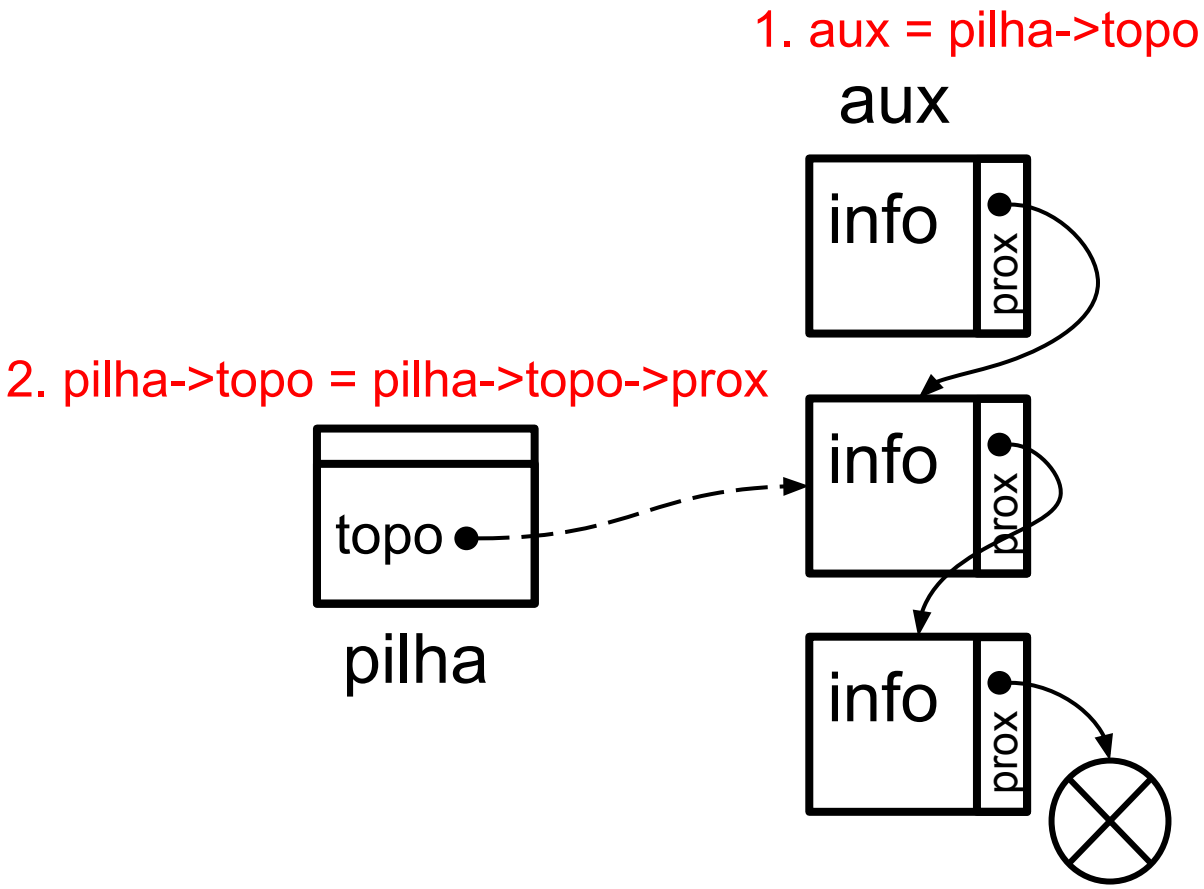
Função de desempilhar (pop)

- função ***pop()*** ou ***desempilha()***



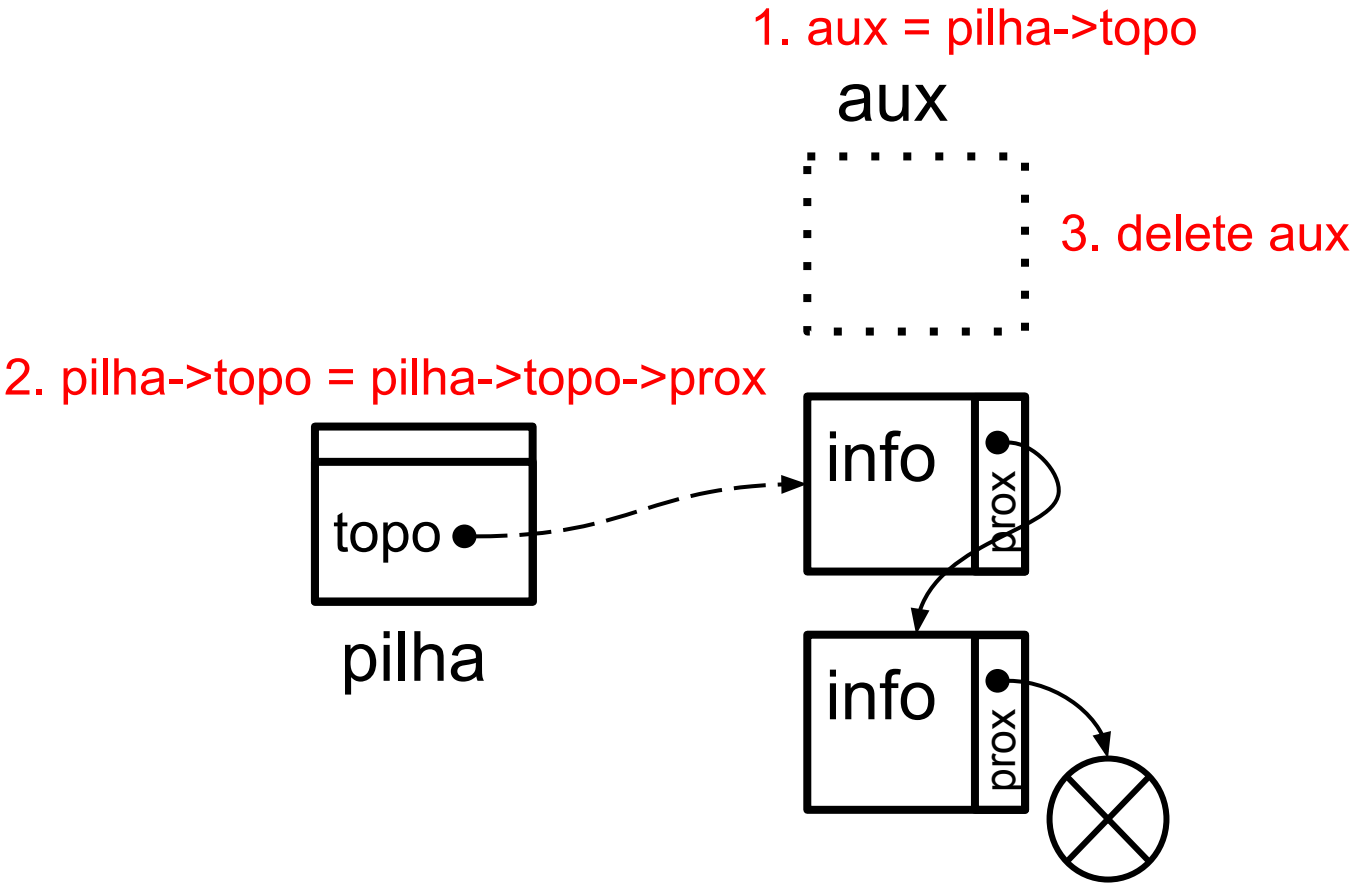
Função de desempilhar (pop)

- função ***pop()*** ou ***desempilha()***



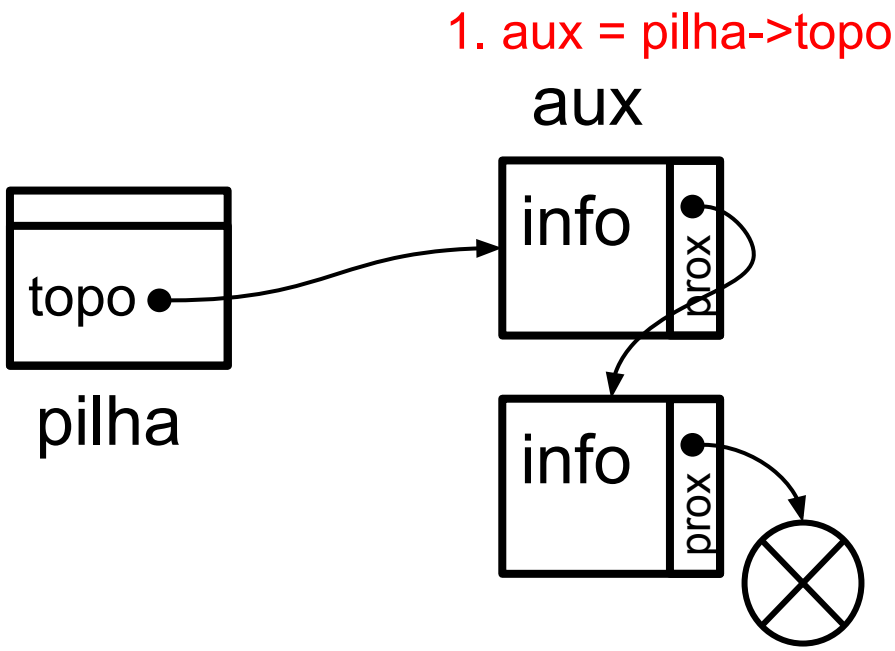
Função de desempilhar (pop)

- função ***pop()*** ou ***desempilha()***



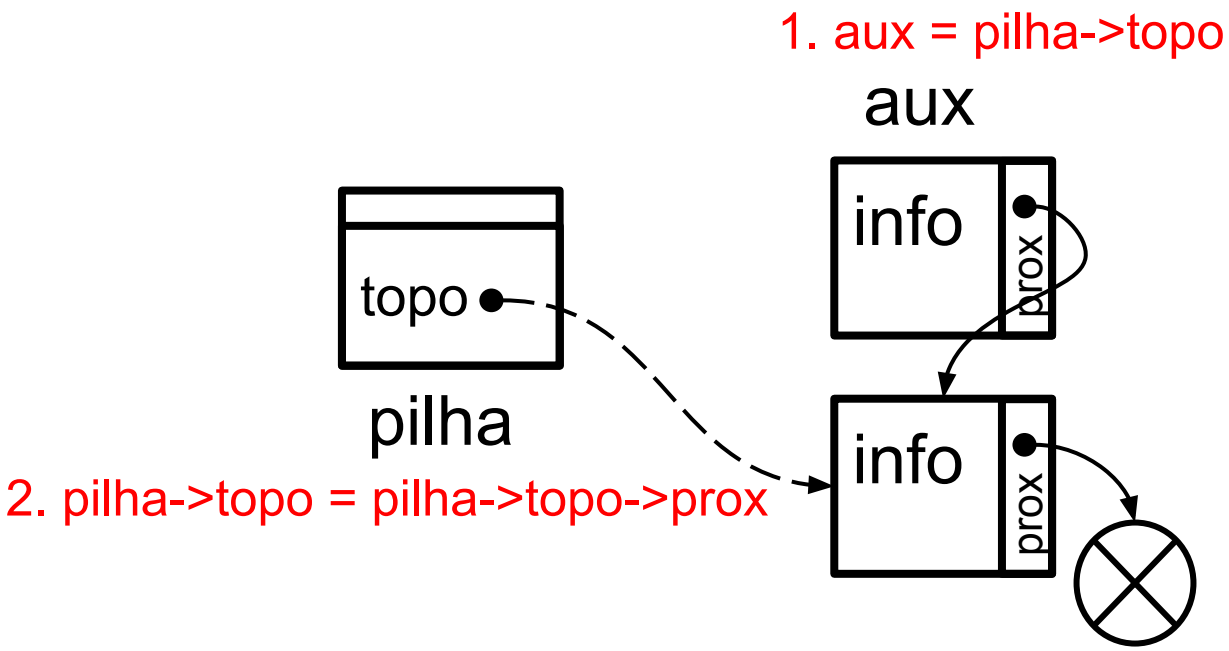
Função de desempilhar (pop)

- função ***pop()*** ou ***desempilha()***



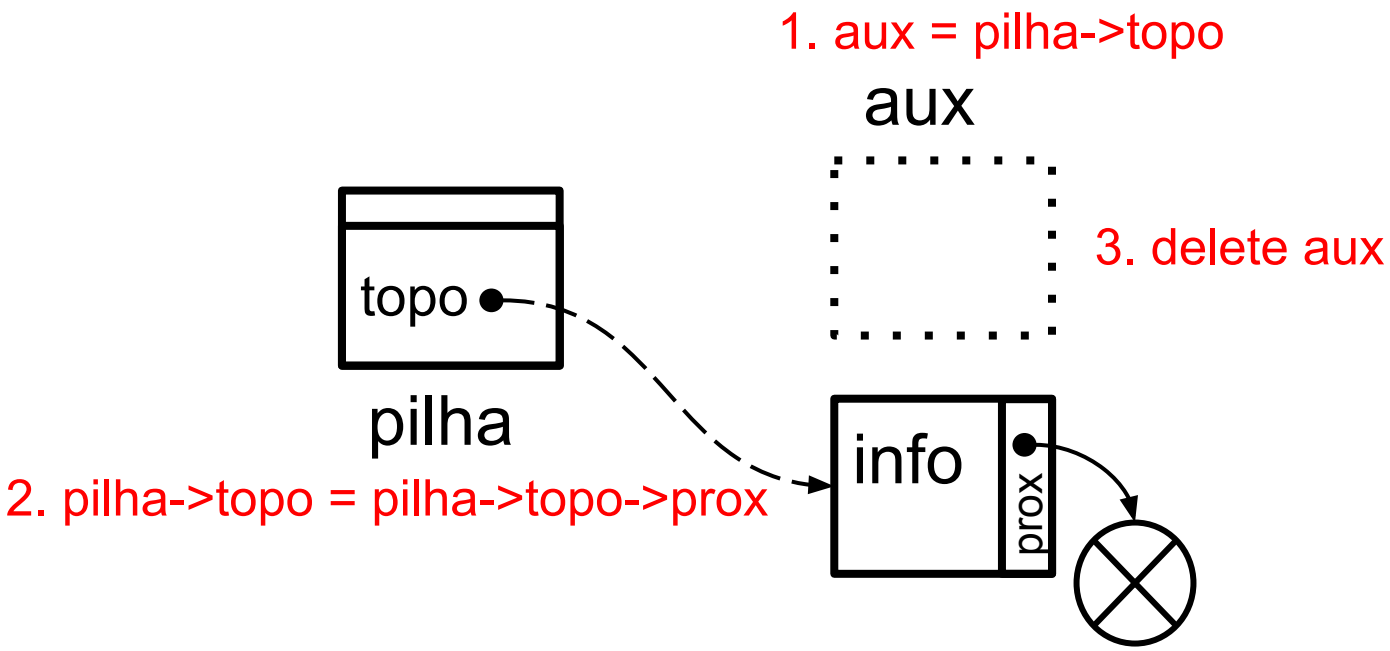
Função de desempilhar (pop)

- função ***pop()*** ou ***desempilha()***



Função de desempilhar (pop)

- função ***pop()*** ou ***desempilha()***



Função de desempilhar *(POP)*

```
/*
FUNÇÃO: pop
RESUMO: desempilha o elemento no topo da pilha e retorna seu valor
PARAM: Pilha* (ponteiro para a pilha), int (valor a ser desempilhado);
RETORNO: Pilha* (ponteiro para a pilha);
*/
int pop(Pilha* pilha){
    Node* aux = pilha->topo;
    int valor;
    if(!vaziaPilha(pilha)){
        valor = aux->info;
        pilha->topo = (pilha->topo)->prox;
        delete aux;
    } else cout << "pilha vazia!" << endl;
    return valor;
}
```

Função de consultar topo *(TOP)*

- A função de consultar o conteúdo do topo (*top*) deve retornar o conteúdo do topo da pilha sem alterá-lo.
  - Se a pilha não é vazia
    1. retornar = (pilha->topo)->info;

Função consultar topo *(TOP)*

```
/*
FUNÇÃO: top
RESUMO: informa o valor do primeiro elemento da pilha
PARAM: Pilha* (ponteiro para a pilha)
RETORNO: int (valor da informação do nó armazenado no topo)
*/
int top(Pilha* pilha){
    if (!vaziaPilha(pilha))
        return (pilha->topo)->info;
    else cout << "pilha vazia!";
}
```

Função de destruir pilha *(destroiPilha)*

- Respeitar as regras de pilha SEMPRE!
  - removendo pelo topo até esvaziar
- Enquanto a pilha não é vazia
  - pop(pilha);

Operações com pilhas

- Pronto! As funções básicas sobre pilhas foram implementadas!
- E agora?
- Precisamos aprender a utilizar pilhas para resolver problemas
  - **TUDO CONSISTE EM EMPILHAR E DESEMPILHAR ELEMENTOS**

Operações com pilhas

- Imprimir o conteúdo de uma pilha?
- Buscar um elemento de uma pilha?
- Buscar e remover um elemento na pilha?
- Processar todo conteúdo de uma pilha?
- Fazer um avançar e voltar de uma página web?

TODAS AS OPERAÇÕES SE BASEIAM EM EMPILHAR E DESEMPILHAR...

Imprimir o conteúdo de uma pilha

- Como imprimir todo o conteúdo da pilha se...
  - só podemos acessar o topo da pilha
  - quando desempilhamos, perdemos os elementos do topo?
  - como remover, guardar e preservar a ordem dos elementos pra reinserí-los de novo?
    - Usando outra pilha como auxiliar!

Imprimir o conteúdo de uma pilha

- Se pilha principal não está vazia:
  - Criamos uma pilha auxiliar vazia
  - Enquanto pilha principal não estiver vazia
    - Imprimo o conteúdo do topo
    - desempilho da principal e empilho na auxiliar
  - Depois, precisamos empilhar de volta:
    - Enquanto auxiliar não estiver vazia
      - Desempilho da auxiliar e empilho na principal

Função imprimir pilha (printPilha) (adicionar na biblioteca!)

```
void printPilha(Pilha* pilha){
   if(!pilhaVazia(pilha)){
      Pilha *aux = criaPilha();
      while(!pilhaVazia(pilha)){
         cout << "(" << top(pilha) << ") -> ";
         push(aux,pop(pilha));
      }
      cout << "|X|" << endl; //pra marcar o final
      //agora reempilhamos na principal
      while(!pilhaVazia(aux)){
         push(pilha,pop(aux));
      }
   }//fim if
}//fim func
```

Atividade:

1. **Crie uma função que busque um elemento de uma pilha.**
2. **Cria uma função que busque e remova um elemento de uma pilha.**
3. **Crie uma função que remova todos os números pares de uma pilha**
4. **Crie uma função que remova todos os valores repetidos de uma pilha.**
5. Crie um novo nó de pilha pra armazenar um endereço web (ex: “www.google.com”) e crie um pograma que insira e remova endereços e implemente o **voltar** e **avançar** usando pilhas.