Estrutura de Dados I

Pilhas

Prof. Dr. Marcelo Otone Aguiar

Universidade Federal do Espírito Santo - UFES

26 de janeiro de 2026

Conteúdo

- Pilhas
- Pilha estática
- Pilha dinâmica

Pilhas

- Estrutura de dados linear **restrita**.
- Baseada na política **LIFO**.
- Muito usada em programação e sistemas.

LIFO =*Last In, First Out*
Equivale à = Último a entrar, primeiro a sair

Pilhas

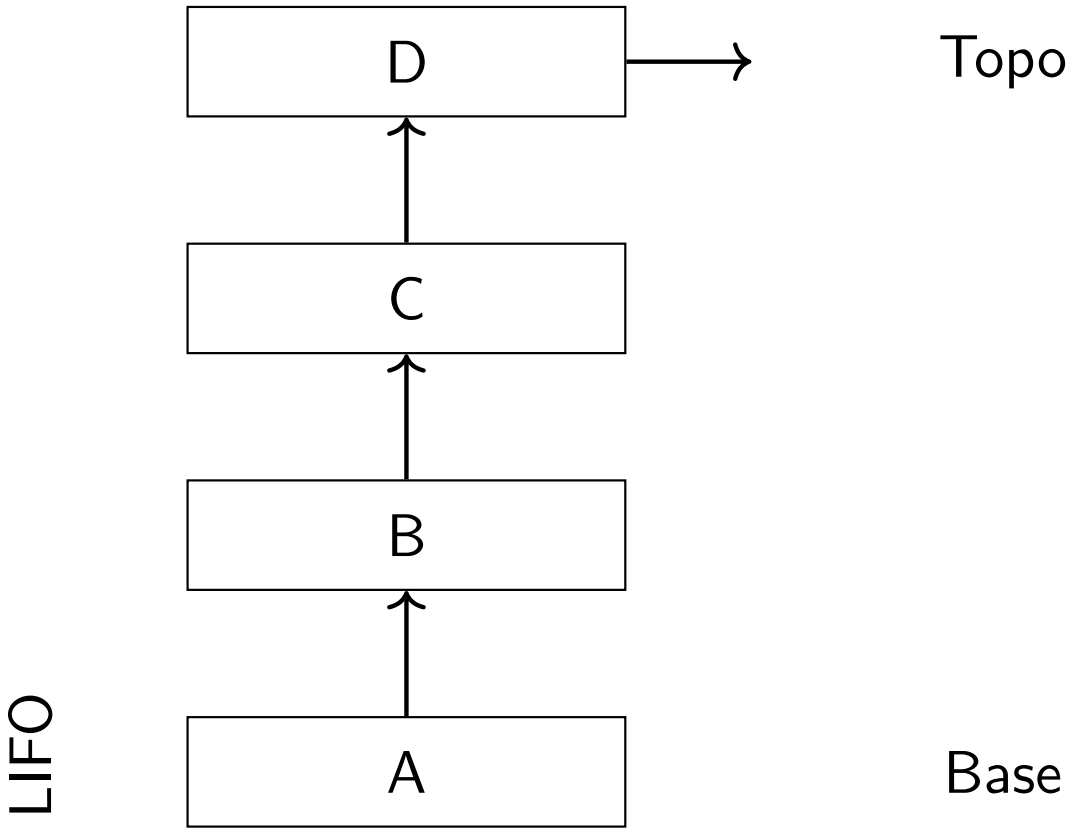
Uma pilha é uma estrutura de dados em que:

- A inserção ocorre **somente no topo**
- A remoção ocorre **somente no topo**
- **Não é permitido** acesso direto a elementos intermediários

Analogia:

- Pilha de pratos
- Pilha de livros
- Histórico de navegação

Pilhas



Operações Básicas da Pilha

- **Push:** insere um elemento no topo
- **Pop:** remove o elemento do topo
- **Top (ou Peek):** acessa o topo sem remover

Exemplos:

- Push(p, x) //Insere x no topo
- Pop(p) //Remove o topo
- Top(p) //Retorna o topo sem removê-lo da pilha

Implementações de Pilhas

Uma pilha pode ser implementada de duas formas:

Estática:

- Uso de vetor
- Tamanho fixo
- Pode causar desperdício ou *overflow*

Dinâmica:

- Lista encadeada
- Tamanho variável
- Uso mais eficiente da memória

Pilha Estática - Implementação

- **PilhaEstatica.h:** definir
  - Os protótipos das funções
  - O tipo de dado armazenado na pilha
  - O ponteiro **pilha**
  - Tamanho do vetor usado na pilha
- **PilhaEstatica.c:** definir
  - O tipo de dados **pilha**
  - Implementar as suas funções.

Código em C - Estrutura da Pilha

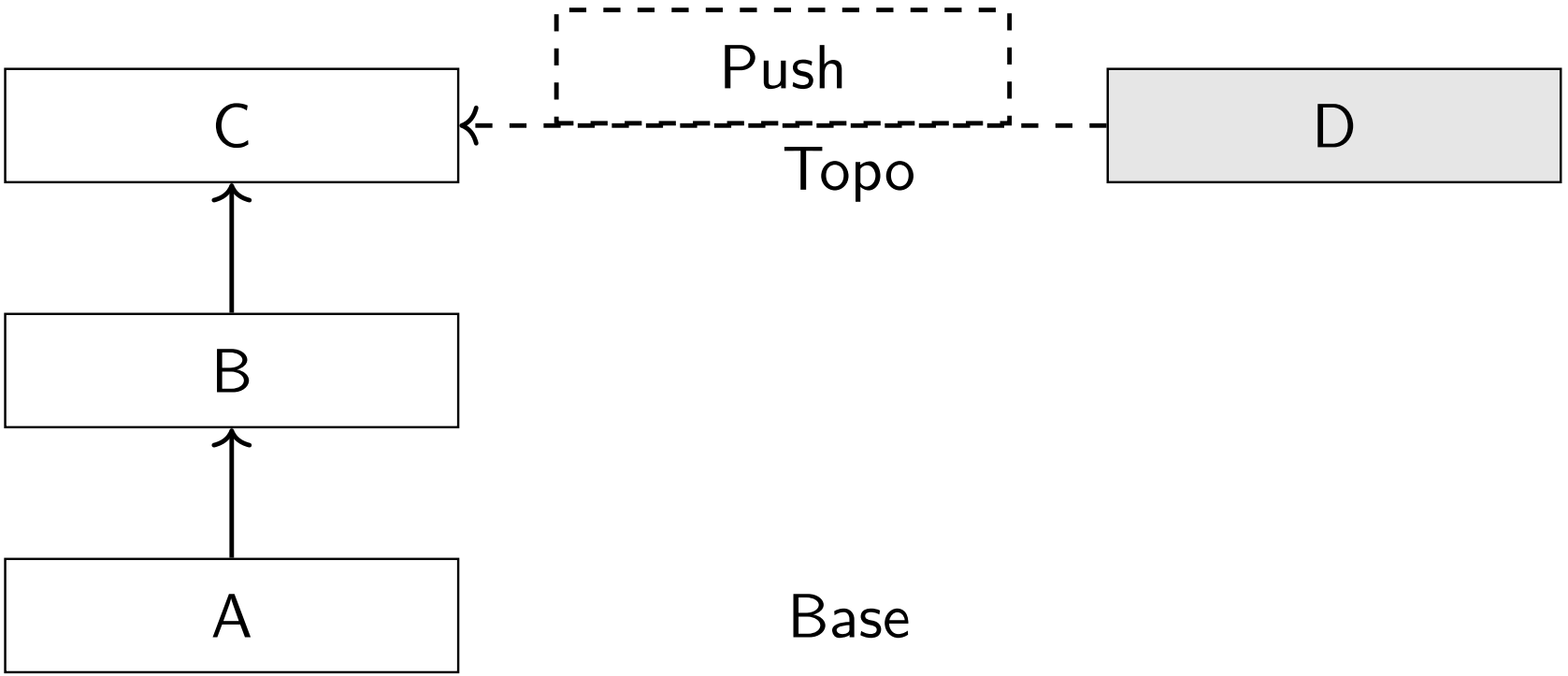
```
1  //Arquivo PilhaEstatica.h
2  #define MAX 100
3  struct aluno {
4    int matricula;
5    char nome[30];
6    float n1, n2, n3;
7  };
8  typedef struct pilha Pilha;
9
10 //Arquivo PilhaEstatica.c
11 struct pilha {
12   int qtd;
13   struct aluno dados[MAX];
14 };
15
16 //Programa principal
17 Pilha *pi;
```

Pilha Estática - Implementação

As funções listadas a seguir para uma **Pilha** podem ser implementadas utilizando as funções que já conhecemos de **lista estática**:

- Criação da pilha
- Liberar a pilha da memória
- Obter o tamanho da pilha
- Verificar se a pilha está cheia
- Verificar se a pilha está vazia

Inserção no topo (*Push*)



Inserção no topo (*Push*)

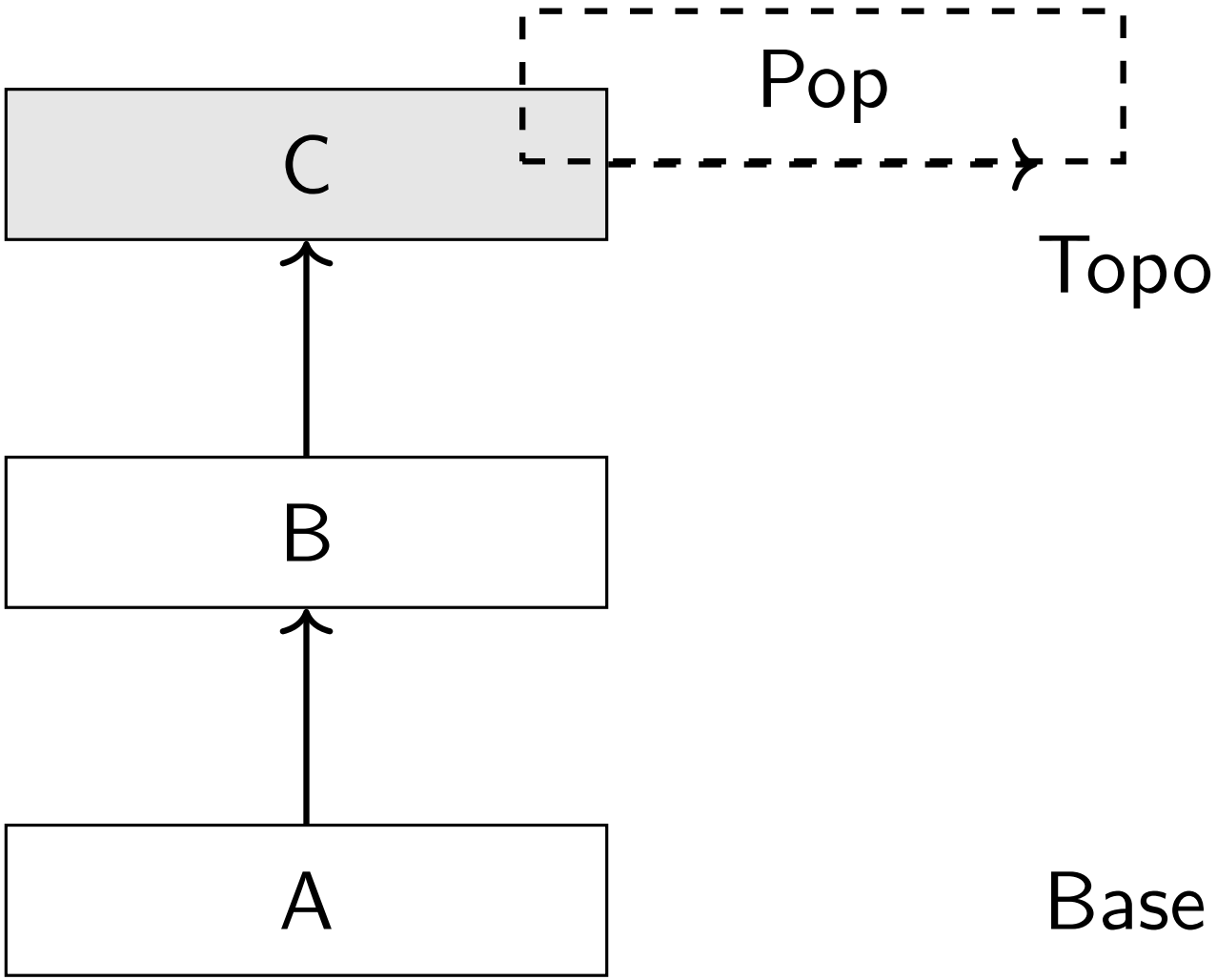
- Em **Pilhas estáticas** a inserção é sempre no *final*, que equivale a inserir no topo
- Inserir no **topo da pilha**, equivale à função de **inserir no final** da lista sequencial estática
- Também existe o caso em que a inserção é feita em uma pilha vazia
- Não se pode inserir em uma pilha cheia.

Código em C - Inserção no topo (*Push*)

```c
//Arquivo PilhaEstatica.c
int push(Pilha* pi, struct aluno al){
  if (pi == NULL)
      return 0;
  if (pilha_cheia(pi))
      return 0;
  pi->dados[pi->qtd] = al;
  pi->qtd++;
  return 1;
}
```

```c
//Programa principal
int x = push(pi, dados_aluno);
```

Remoção no topo (*Pop*)



Remoção no topo (*Pop*)

- Em **Pilhas estáticas** a remoção é sempre no *final*, que equivale a remover no topo
- Remover no **topo da pilha**, equivale à função de **remover no final** da lista sequencial estática
- Não se pode remover em uma pilha vazia.

Código em C - Remoção no topo (*Pop*)

```c
//Arquivo PilhaEstatica.c
int pop(Pilha* pi){
  if (pi == NULL)
      return 0;
  if (pi->qtd == 0)
      return 0;
  pi->qtd--;
  return 1;
}
```

```c
//Programa principal
int x = pop(pi);
```

Consulta no topo (*Top*)

- Em **Pilhas estáticas** a consulta é sempre no *final*, que equivale a consultar no topo
- Consultar no **topo da pilha**, equivale à consultar o último elemento da lista sequencial estática.

Código em C - Consulta no topo (*Top*)

```c
//Arquivo PilhaEstatica.c
int top(Pilha* pi, struct aluno *al){
  if (pi == NULL || pi->qtd == 0)
      return 0;

  *al = pi->dados[pi->qtd-1];
  return 1;
}
```

```c
//Programa principal
int x = top(pi, &dados_aluno);
```

Pilha Dinâmica - Implementação

- **PilhaDin.h:** definir
  - Os protótipos das funções
  - O tipo de dado armazenado na pilha
  - O ponteiro **pilha**
- **PilhaDin.c:** definir
  - O tipo de dados **pilha**
  - Implementar as suas funções.

Código em C - Estrutura da Pilha

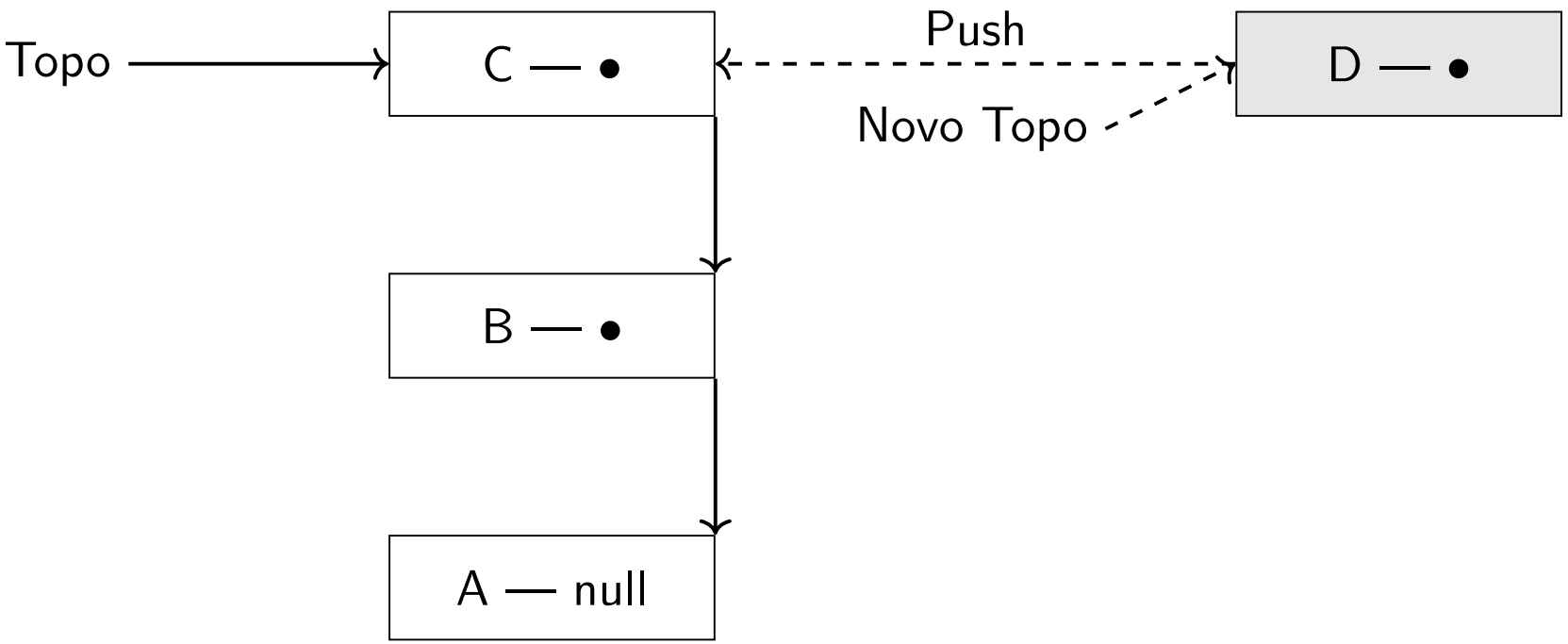
```
1  //Arquivo PilhaDin.h
2  struct aluno {
3    int matricula;
4    char nome[30];
5    float n1, n2, n3;
6  };
7  typedef struct elemento* Pilha;
8
9  //Arquivo PilhaDin.c
10 struct elemento {      //estrutura que controla a pilha
11   struct aluno dados;
12   struct elemento *prox;
13 };
14 typedef struct elemento Elem;
15
16 //Programa principal
17 Pilha *pi; //ponteiro para ponteiro
```

Pilha Dinâmica - Implementação

As funções listadas a seguir para uma **Pilha** podem ser implementadas utilizando as funções que já conhecemos de **lista simplesmente encadeada**:

- Criação da pilha
- Liberar a pilha da memória
- Obter o tamanho da pilha
- Verificar se a pilha está vazia

Inserção no topo (*Push*)



Inserção no topo (*Push*)

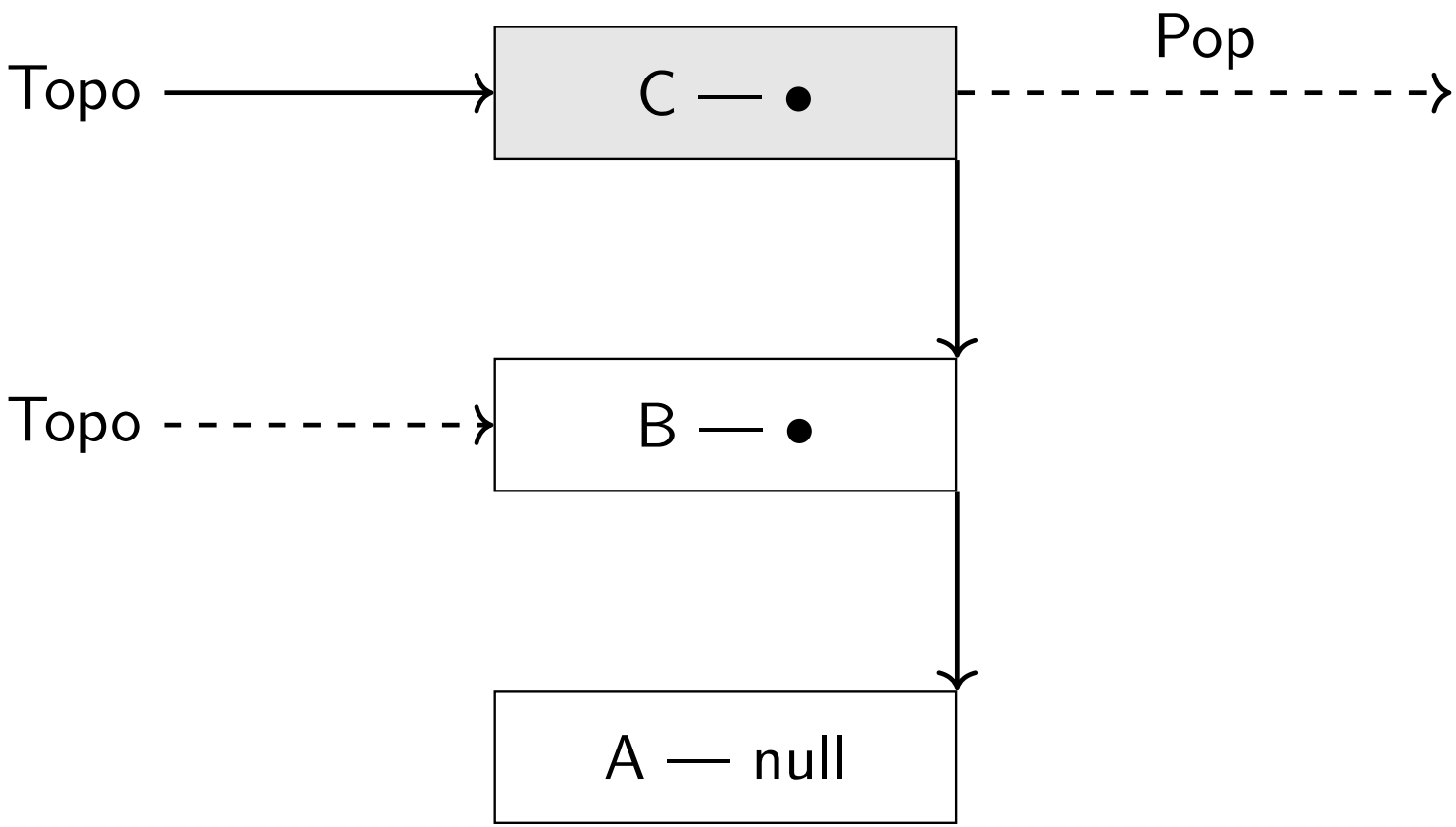
- Em **Pilhas dinâmicas** a inserção é sempre no *início*, que equivale a inserir no topo
- Inserir no **topo da pilha**, equivale à função de **inserir no início** da lista simplesmente encadeada
- Também existe o caso em que a inserção é feita em uma pilha vazia.

Código em C - Inserção no topo (*Push*)

```c
int push(Pilha* pi, struct aluno al){
  if (pi == NULL)
      return 0;
  Elem* no = (Elem*) malloc(sizeof(Elem));
  if (no == NULL)
      return 0;
  no->dados = al;
  no->prox = (*pi);
  *pi = no;
  return 1;
}
```

```c
//Programa principal
int x = push(pi, dados_aluno);
```

Remoção no topo (*Pop*)



Remoção no topo (*Pop*)

- Em **Pilhas dinâmicas** a remoção é sempre no *início*, que equivale a remover no topo
- Remover no **topo da pilha**, equivale à função de **remover no início** da lista simplesmente encadeada
- Não se pode remover em uma pilha vazia.

Código em C - Remoção no topo (*Pop*)

```c
int pop(Pilha* pi){
  if (pi == NULL)
      return 0;
  if (*pi == NULL) //pilha vazia
      return 0;
  Elem *no = *pi;
  *pi = no->prox;
  free(no);
  return 1;
}
```

```c
//Programa principal
int x = pop(pi);
```

Consulta no topo (*Top*)

- Em **Pilhas dinâmicas** a consulta é sempre no *início*, que equivale a consultar no topo
- Consultar no **topo da pilha**, equivale à consultar o primeiro elemento da lista simplesmente encadeada.

Código em C - Consulta no topo (*Top*)

```c
int top(Pilha* pi, struct aluno al){
  if (pi == NULL)
       return 0;
  if ((*pi) == NULL) //pilha vazia
       return 0;
  *al = (*pi)->dados;
  return 1;
}
```

```c
//Programa principal
int x = top(pi, &dados_aluno);
```