UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Câmpus Campo Mourão

Ciência da Computação

Algoritmos e Estrutura de Dados 1

Pilha com alocação dinâmica

Prof. Rafael Liberato

liberato@utfpr.edu.br

Objetivos

- Entender o funcionamento de uma Pilha Dinâmica
- Ser capaz de implementar as operações definidas no TAD Pilha manipulando uma estrutura dinâmica de armazenamento.

Roteiro

- TAD Pilha
- Pilha Dinâmica
- Simulação
- Implementação

TAD Pilha

TAD Pilha

```
#define ItemType int

typedef struct{

}Stack;

Stack* createStack();
void initializeStack(Stack* stack);
int push(Stack* stack, ItemType e);
int pop(Stack* stack, ItemType* e);
int top(Stack* stack, ItemType* e);
void printStack(Stack* stack);
int containsStack(Stack* stack, ItemType *e);
int sizeStack(Stack* stack);
int isEmptyStack(Stack* stack);
```

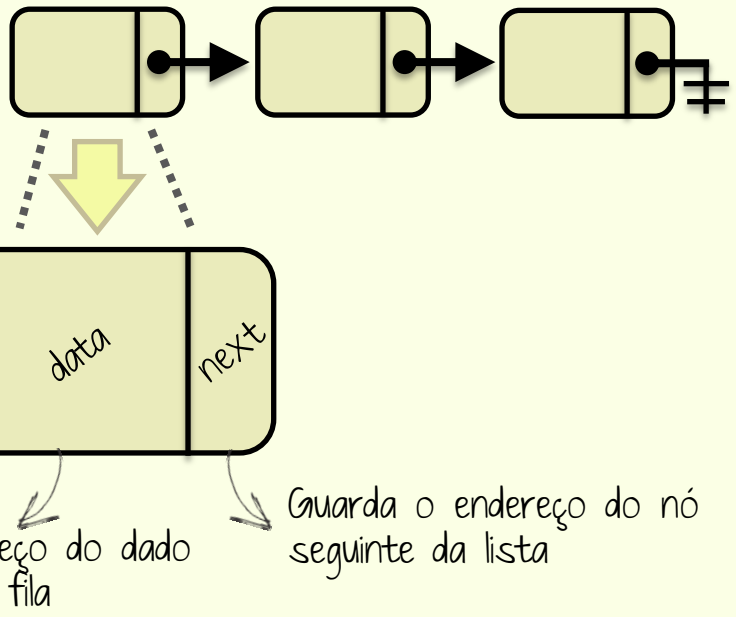
Vamos identificar os atributos que representarão a Pilha dinâmica

Estrutura utilizada para armazenar os dados

Pilha Dinâmica

Pilha Dinâmica

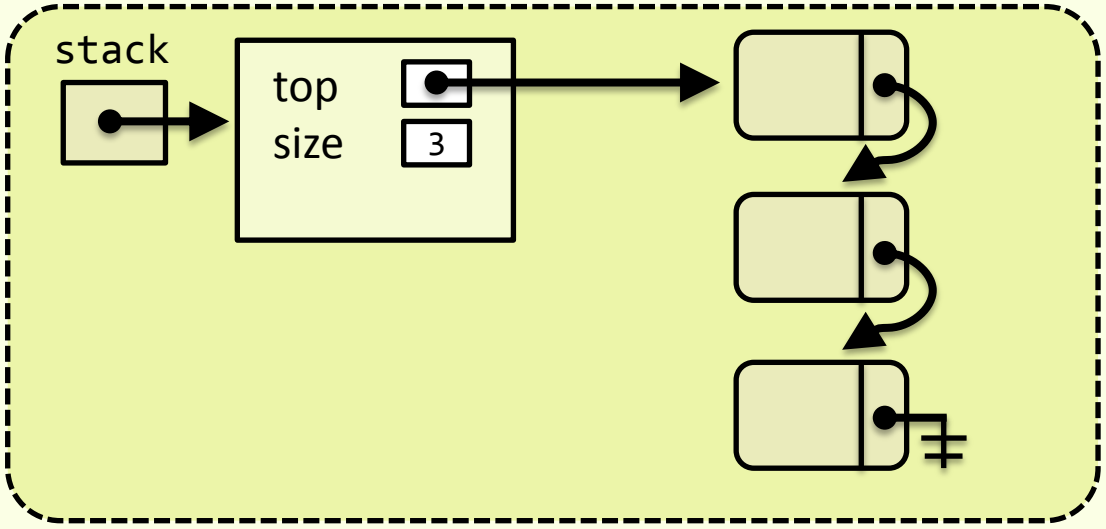
- A **Pilha Dinâmica** utiliza uma estrutura de alocação dinâmica de memória para o armazenamento dos dados
- Portanto, temos que utilizar uma estrutura própria para armazenar e interligar os dados
  - ➔ Um encadeamento de nós




```
typedef struct node{
   ItemType    data;
   struct node *next;
}Node;
```

Pilha Dinâmica

- A Pilha Dinâmica é representada pelo endereço do nó que está no topo do encadeamento.
- Também utilizaremos um atributo para guardar a quantidade de elementos contidos na Pilha.




```
typedef struct node{
    ItemType data;
    struct node *next;
}Node;
```

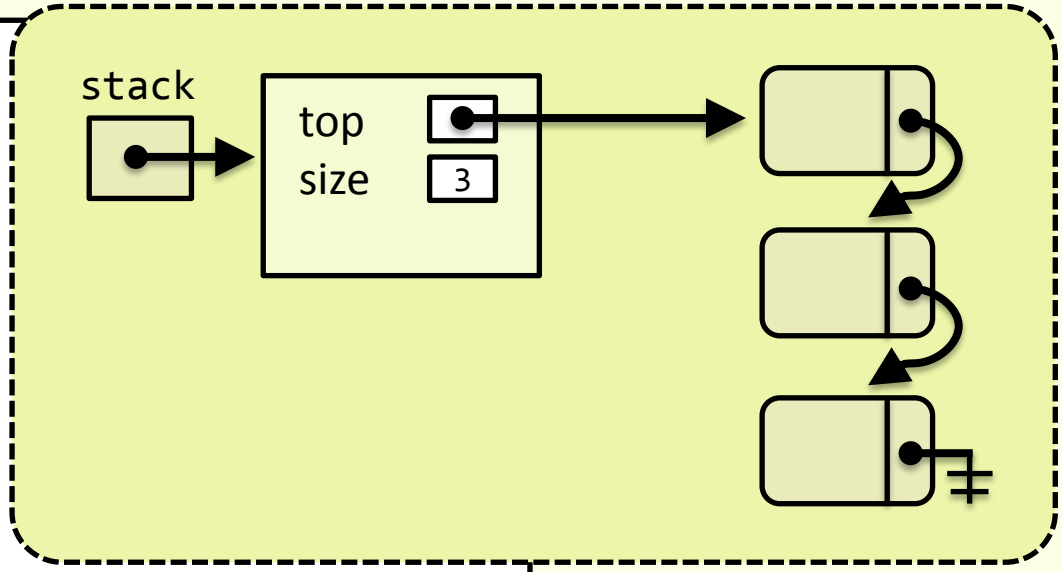
```
typedef struct{
    Node *top;
    int size;
}Stack;
```

Simulação

Simulação

- Utilize a simulação para entender o comportamento das funções e auxiliá-lo na implementação.

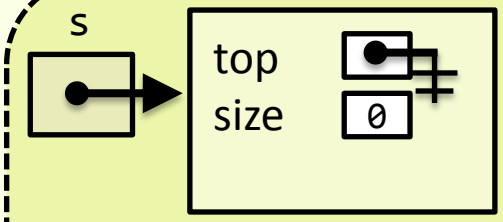
```
#define ItemType int

typedef struct{
    Node *top;
    int size;
}Stack;


Stack* createStack();
void initializeStack(Stack* stack);
int push(Stack* stack, ItemType e);
int pop(Stack* stack, ItemType* e);
int top(Stack* stack, ItemType* e);
void printStack(Stack* stack);
int containsStack(Stack* stack, ItemType *e);
int sizeStack(Stack* stack);
int isEmptyStack(Stack* stack);
```




```
typedef struct node{
    ItemType data;
    struct node *next;
}Node;
```

Simulação

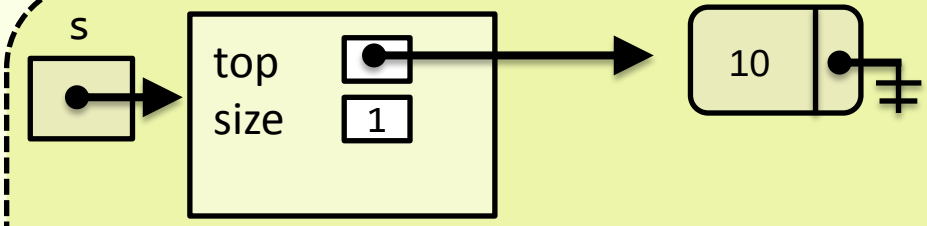
```
Stack* s = createStack();
push(s,10);
push(s,20);
push(s,30);
ItemType removed;
pop(s, &removed);
pop(s, &removed);
pop(s, &removed);
```



PILHA VAZIA

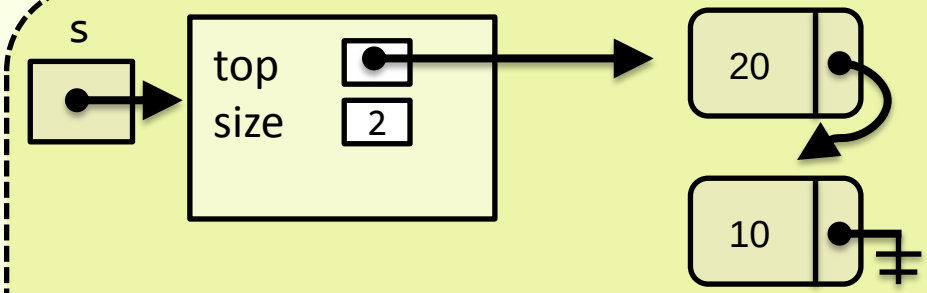
Simulação

```
Stack* s = createStack();
push(s,10);
push(s,20);
push(s,30);
ItemType removed;
pop(s, &removed);
pop(s, &removed);
pop(s, &removed);
```



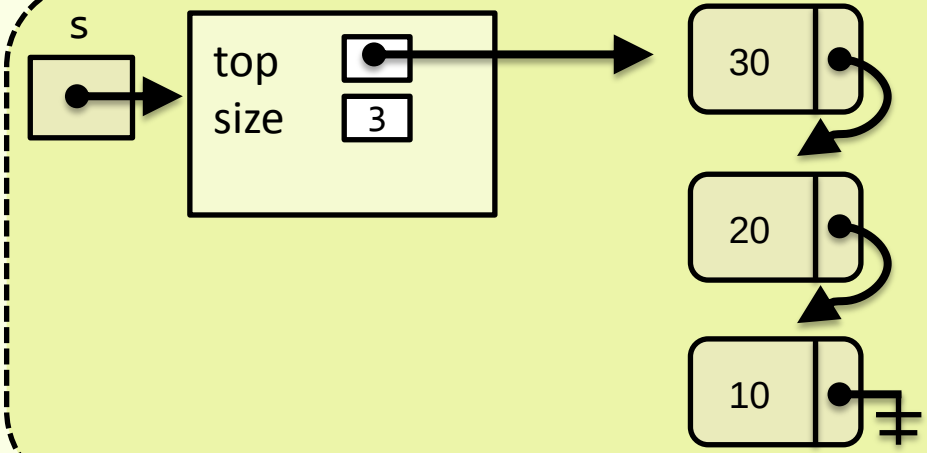
Simulação

```
Stack* s = createStack();
push(s,10);
push(s,20);
push(s,30);
ItemType removed;
pop(s, &removed);
pop(s, &removed);
pop(s, &removed);
```



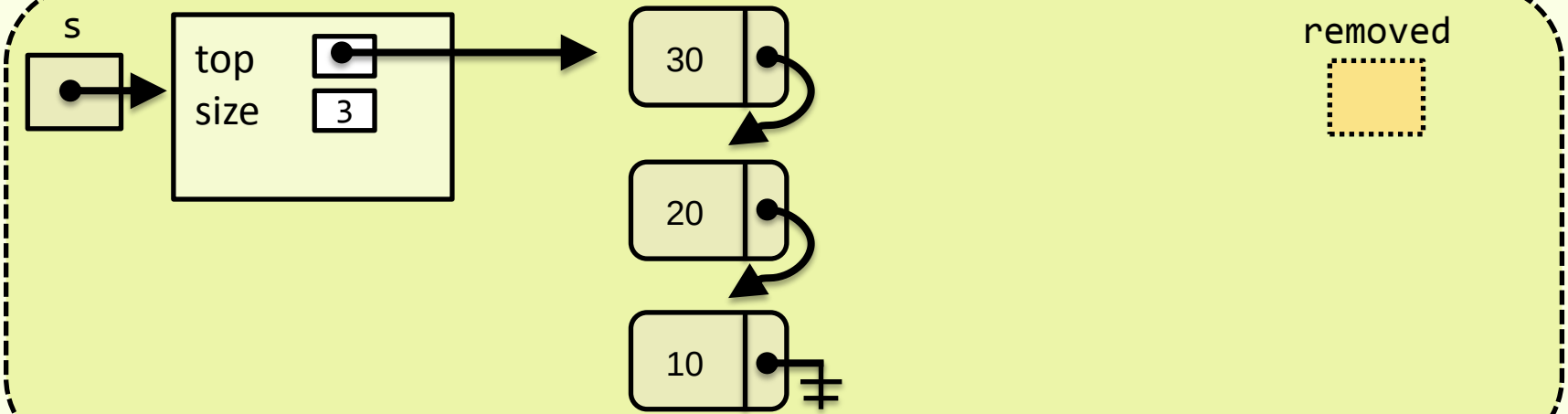
Simulação

```
Stack* s = createStack();
push(s,10);
push(s,20);
push(s,30);
ItemType removed;
pop(s, &removed);
pop(s, &removed);
pop(s, &removed);
```



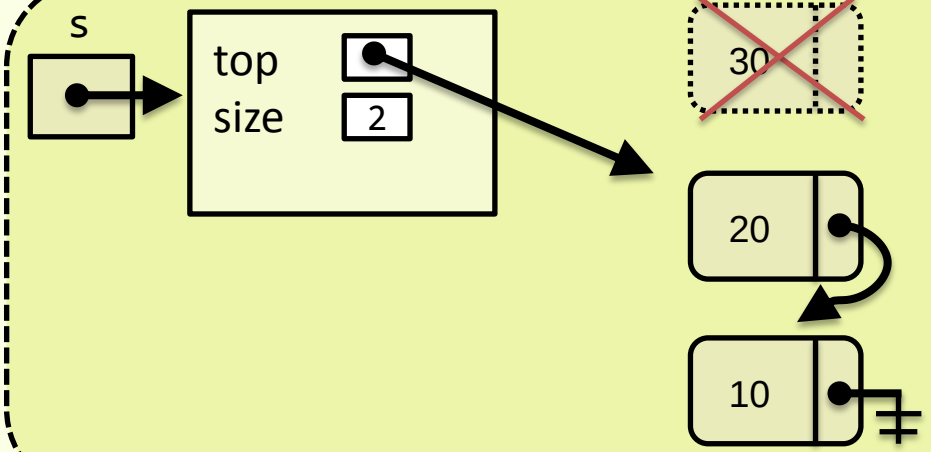
Simulação

```
Stack* s = createStack();
push(s,10);
push(s,20);
push(s,30);
ItemType removed;
pop(s, &removed);
pop(s, &removed);
pop(s, &removed);
```



Simulação

```
Stack* s = createStack();
push(s,10);
push(s,20);
push(s,30);
ItemType removed;
pop(s, &removed);
pop(s, &removed);
pop(s, &removed);
```

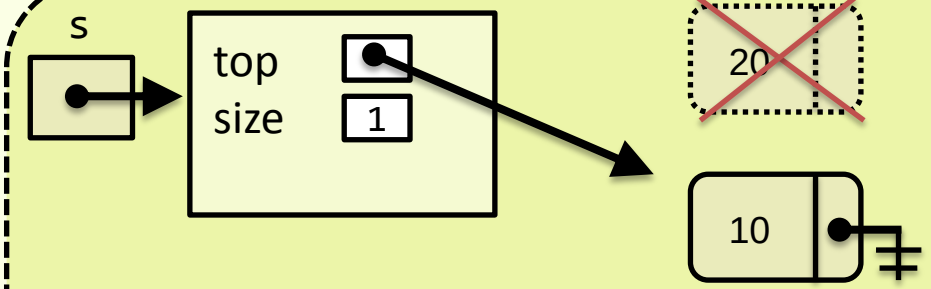


removed

30

Simulação

```
Stack* s = createStack();
push(s,10);
push(s,20);
push(s,30);
ItemType removed;
pop(s, &removed);
pop(s, &removed);
pop(s, &removed);
```

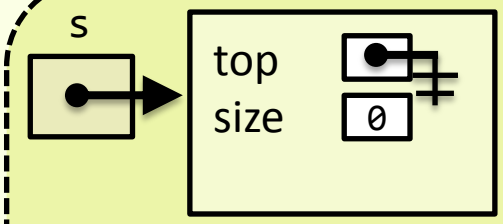


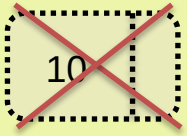
removed

20

Simulação

```
Stack* s = createStack();
push(s,10);
push(s,20);
push(s,30);
ItemType removed;
pop(s, &removed);
pop(s, &removed);
pop(s, &removed);
```





removed

10

PILHA VAZIA

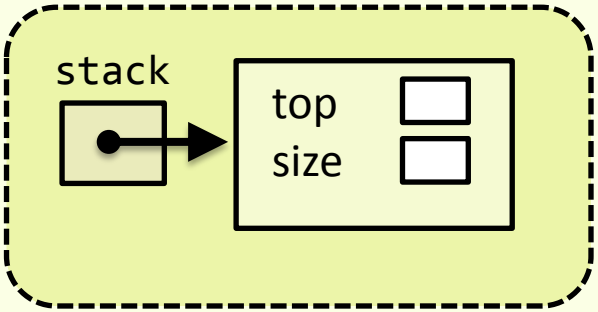
Implementação

Implementação

- A partir dessa simulação é possível extrair o comportamento das funções sobre os atributos da **Pilha dinâmica**

```
Stack* createStack();
void initializeStack(Stack* stack);
int push(Stack* stack, ItemType e);
int pop(Stack* stack, ItemType* e);
int top(Stack* stack, ItemType* e);
void printStack(Stack* stack);
int containsStack(Stack* stack, ItemType *e);
int sizeStack(Stack* stack);
int isEmptyStack(Stack* stack);
```

```
typedef struct node{
    ItemType data;
    struct node *next;
}Node;

typedef struct{
    Node *top;
    int size;
}Stack;
```



Implementação

LET'S DO IT