TDA PILA
IMPLEMENTACION CON NODOS

75.41 - ALGORITMOS Y PROGRAMACIÓN II

FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

LUCAS PANDOLFO - 2DO CUATRIMESTRE 2019

FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

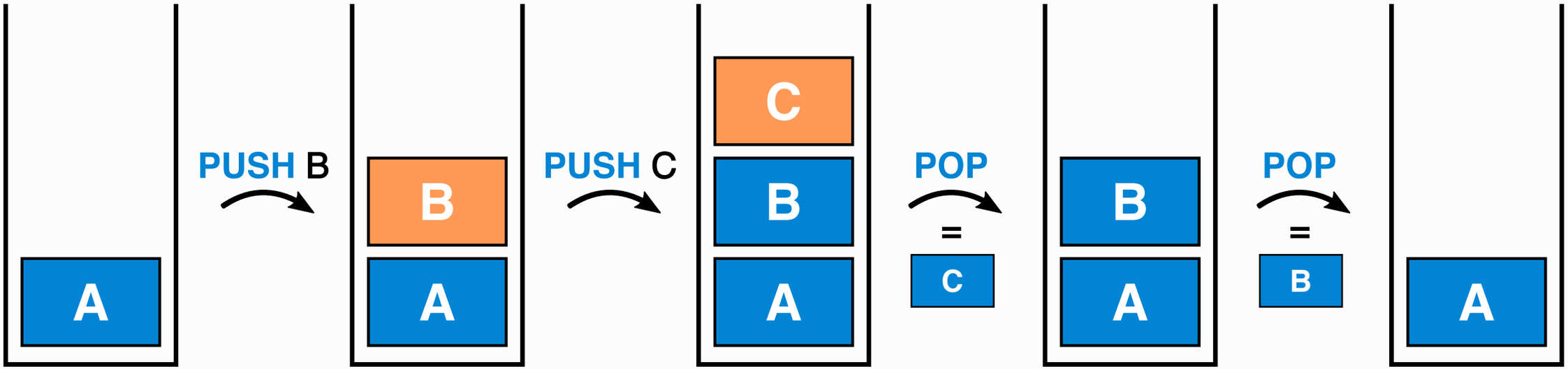
¿QUÉ ES UNA PILA?

¿QUÉ ES UNA PILA?

Estructura de datos **LIFO** (**L**ast **I**n **F**irst **O**ut).

PUSH agrega un elemento en el tope de la pila.

POP quita el elemento en el tope de la pila, si existe.

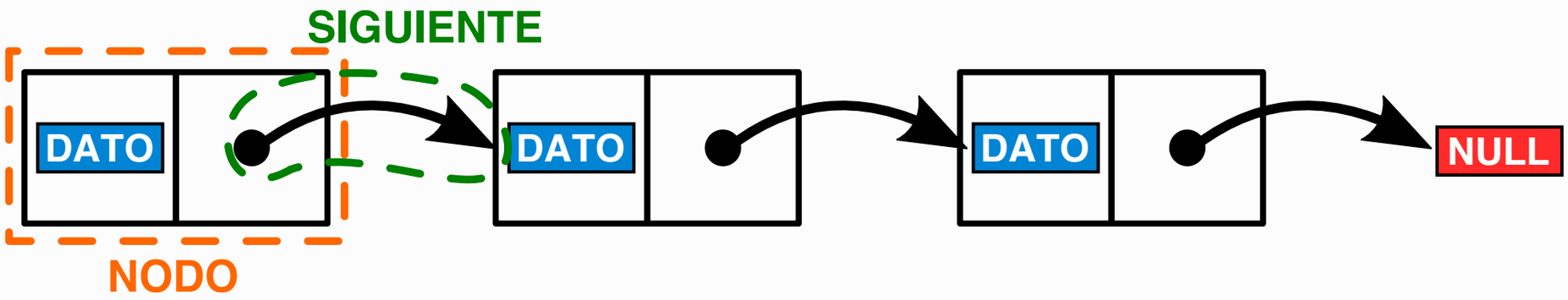


FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

FUNCIONAMIENTO E IMPLEMENTACIÓN (NODOS ENLAZADOS)

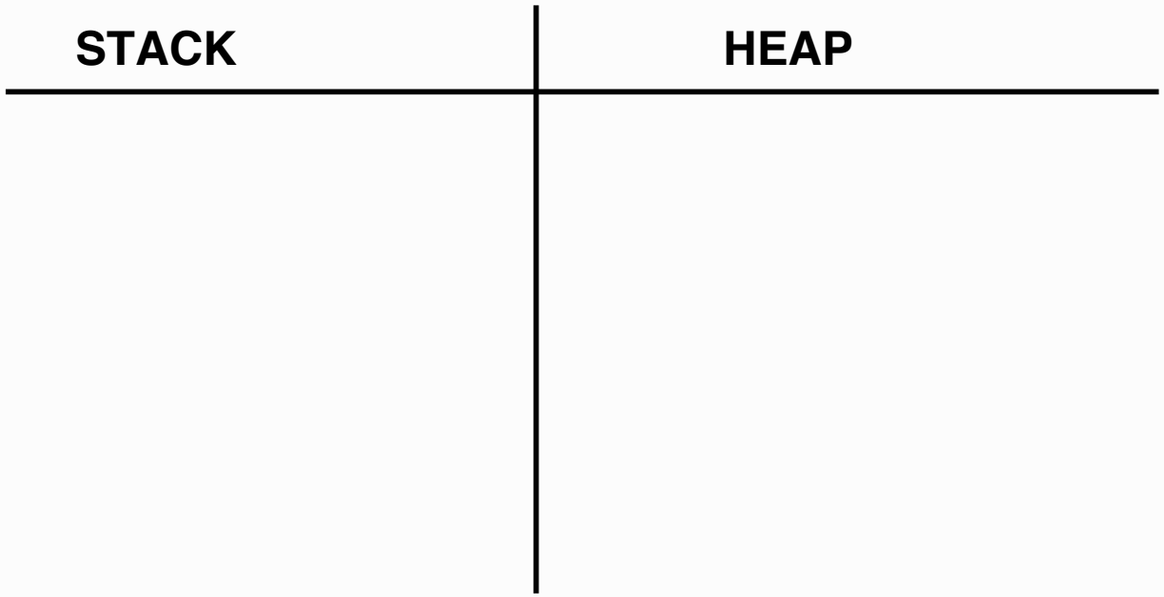
NODOS

```
typedef struct Nodo{
   void* dato;
   struct Nodo* siguiente;
} Nodo;
```



CREACIÓN

```c
typedef struct pila {
  nodo_t* primer_nodo;
} pila_t;

pila_t* pila_crear(){
  pila_t* pila = malloc(sizeof(pila_t));
  pila->primer_nodo = NULL;
  return pila;
}
```

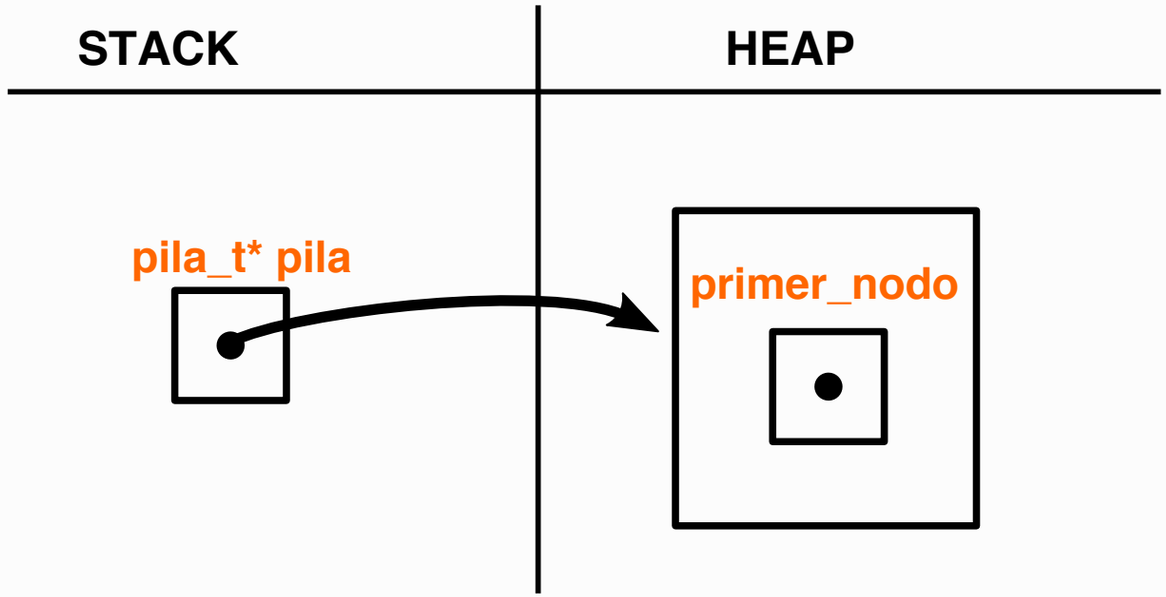


CREACIÓN

```c
typedef struct pila {
  nodo_t* primer_nodo;
} pila_t;

pila_t* pila_crear(){
  pila_t* pila = malloc(sizeof(pila_t));
  pila->primer_nodo = NULL;
  return pila;
}
```

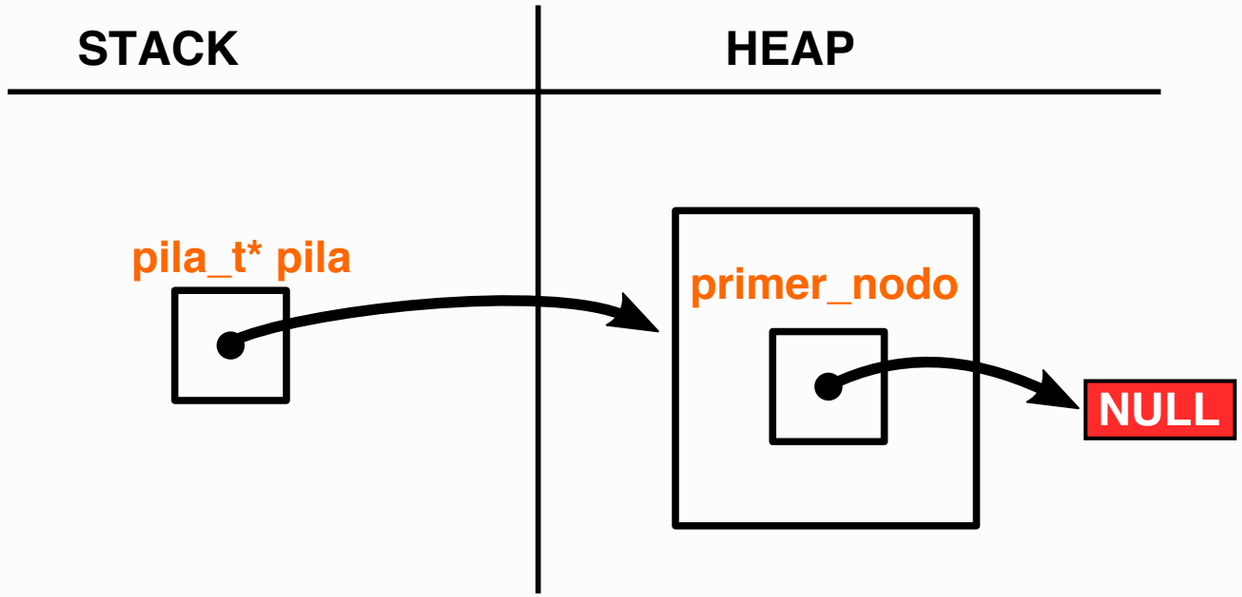
CREACIÓN

```c
typedef struct pila {
  nodo_t* primer_nodo;
} pila_t;

pila_t* pila_crear(){
  pila_t* pila = malloc(sizeof(pila_t));
  pila->primer_nodo = NULL;
  return pila;
}
```



CREACIÓN

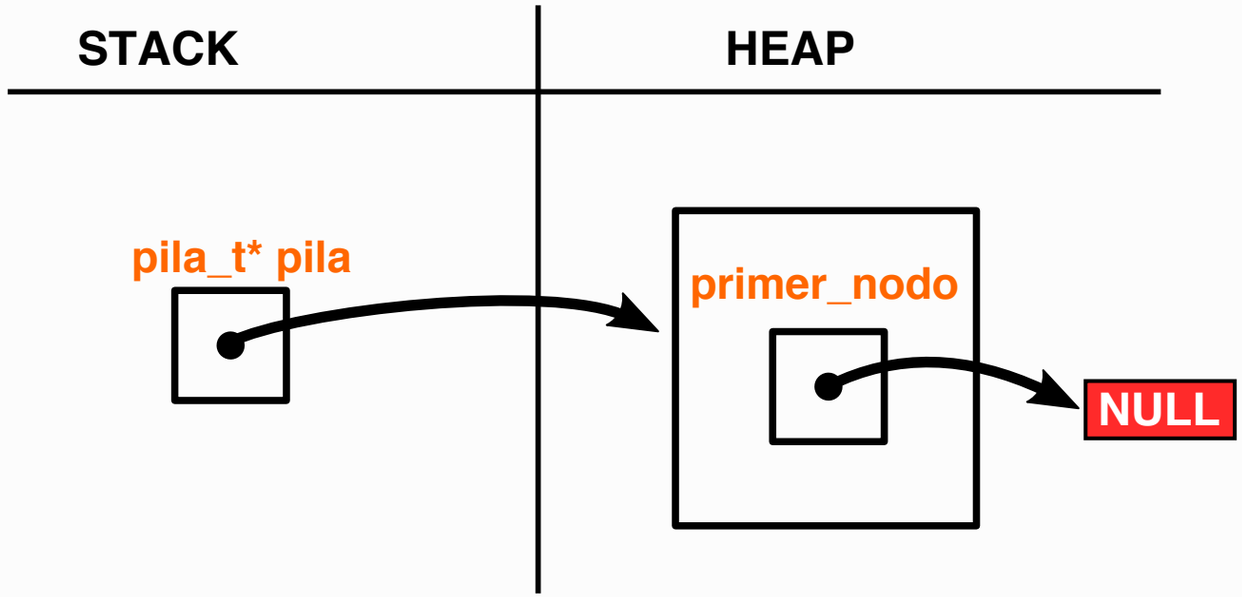
```c
typedef struct pila {
  nodo_t* primer_nodo;
} pila_t;

pila_t* pila_crear(){
  pila_t* pila = malloc(sizeof(pila_t));
  pila->primer_nodo = NULL;
  return pila;
}
```



CREACIÓN

```c
typedef struct pila {
  nodo_t* primer_nodo;
} pila_t;

pila_t* pila_crear(){
  pila_t* pila = malloc(sizeof(pila_t));
  pila->primer_nodo = NULL;
  return pila;
}
```

¿QUÉ FALTA? ಠ_ಠ

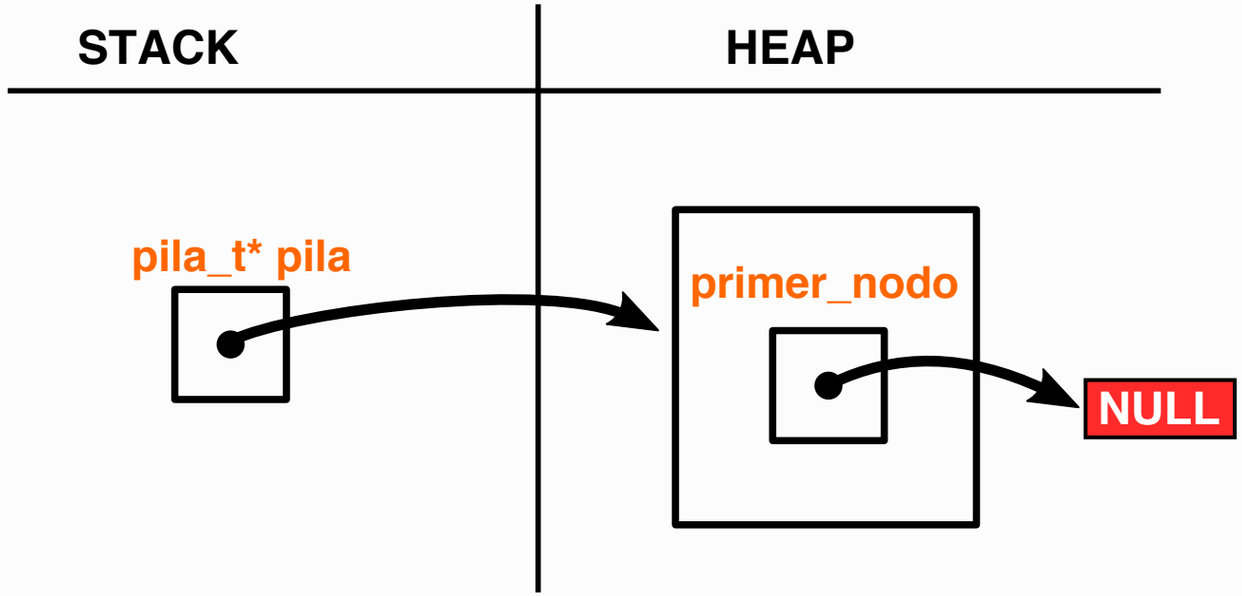


CREACIÓN

```c
typedef struct pila {
  nodo_t* primer_nodo;
} pila_t;

pila_t* pila_crear(){
  pila_t* pila = malloc(sizeof(pila_t));
  pila->primer_nodo = NULL;
  return pila;
}
```

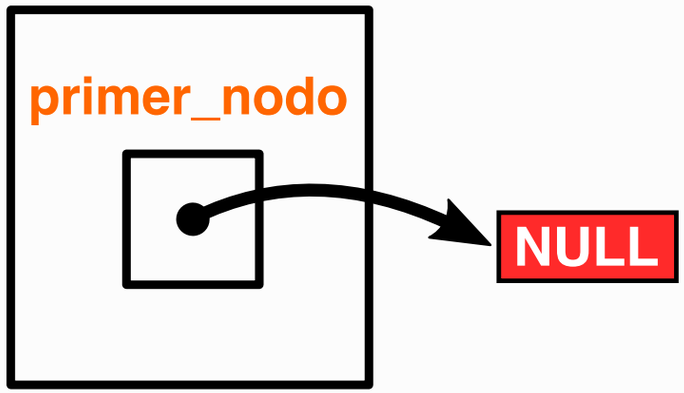
¿QUÉ FALTA? ಠ_ಠ

VERIFICAR NULL



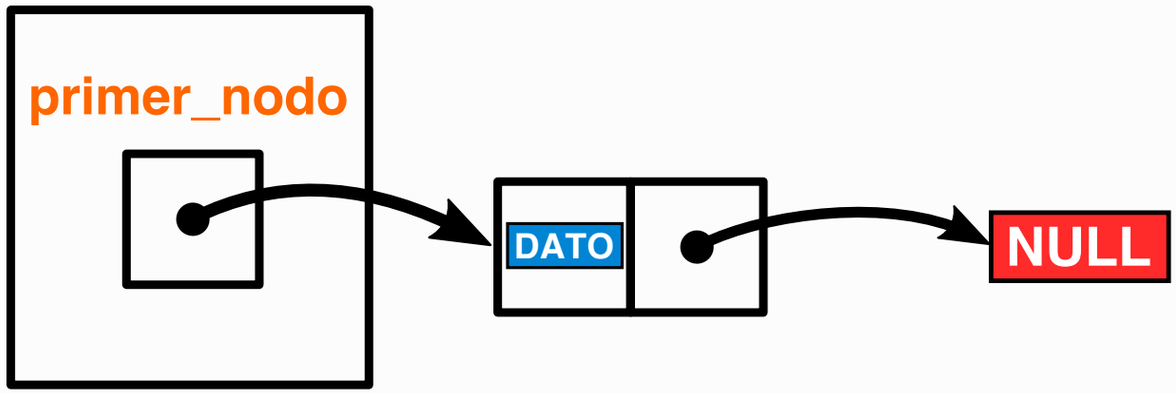
¿PUSH?

PUSH(DATO), PUSH(DATO), PUSH(DATO)



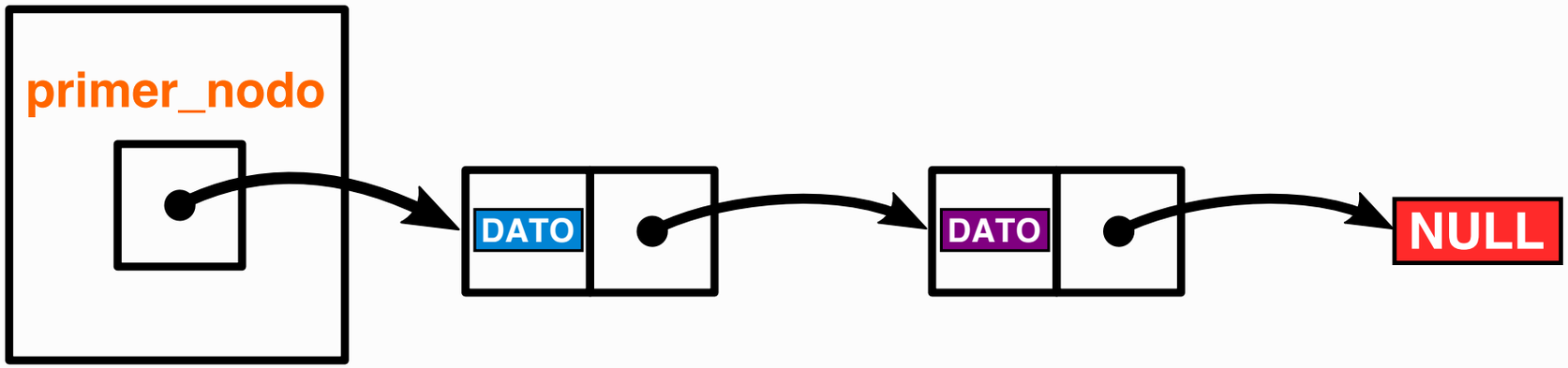
¿PUSH?

PUSH(DATO), PUSH(DATO), PUSH(DATO)



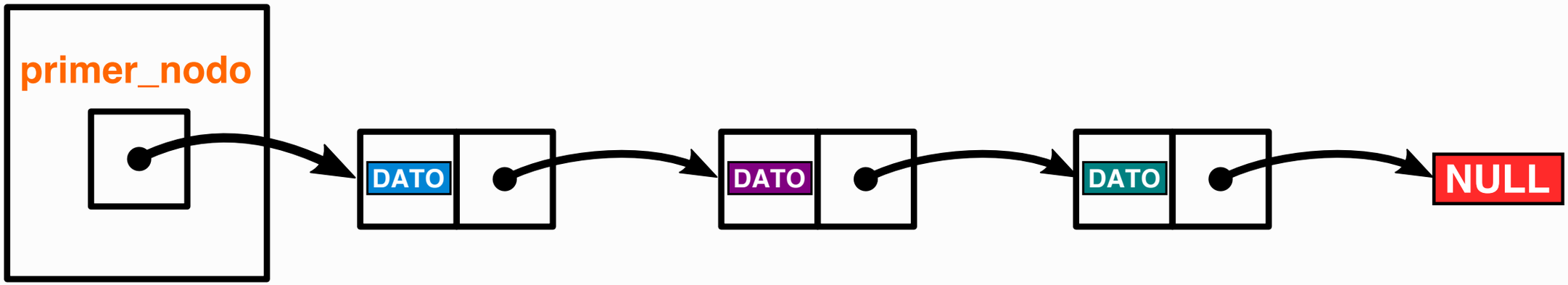
¿PUSH?

PUSH(DATO), PUSH(DATO), PUSH(DATO)



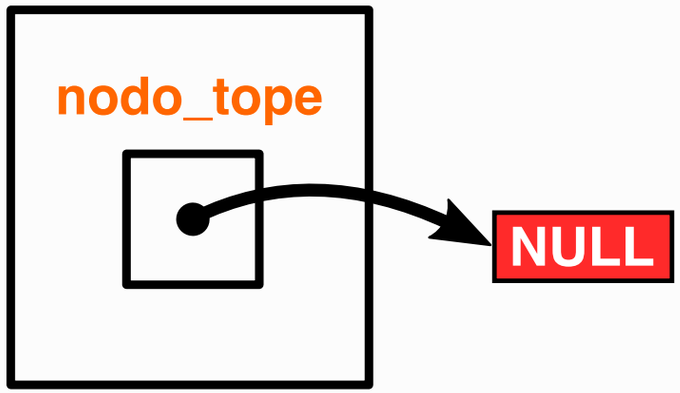
¿PUSH?

PUSH(DATO), PUSH(DATO), PUSH(DATO)



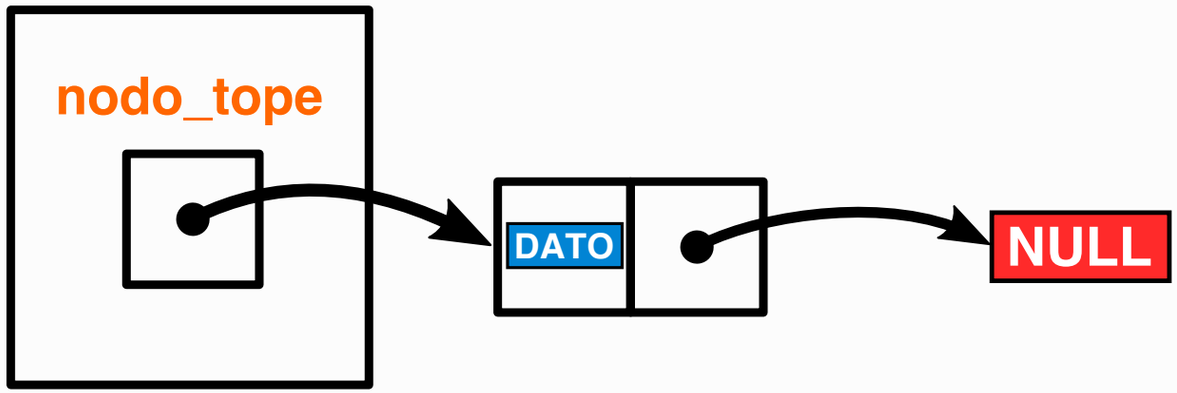
PUSH

PUSH(DATO), PUSH(DATO), PUSH(DATO)



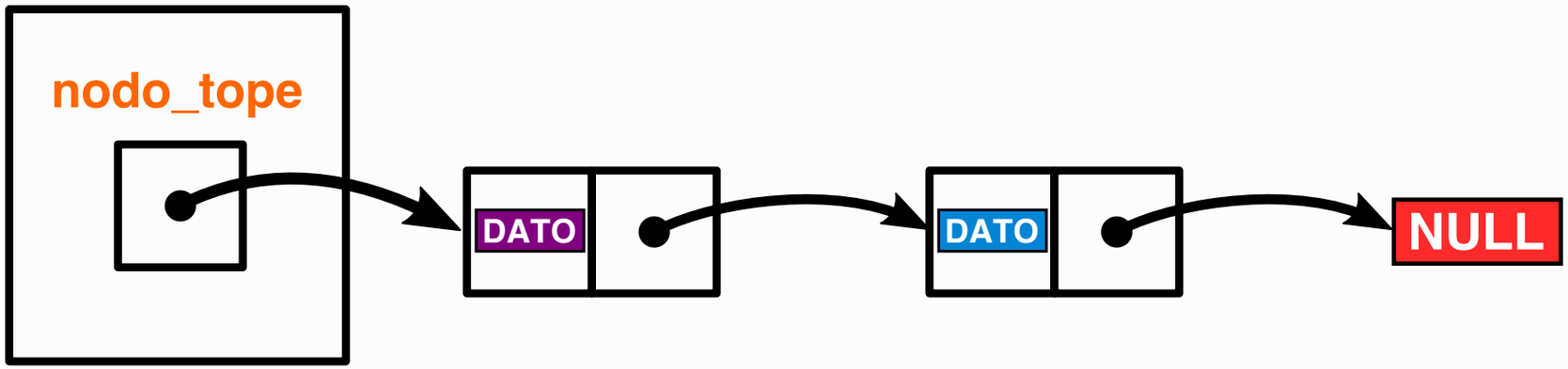
PUSH

PUSH(DATO), PUSH(DATO), PUSH(DATO)



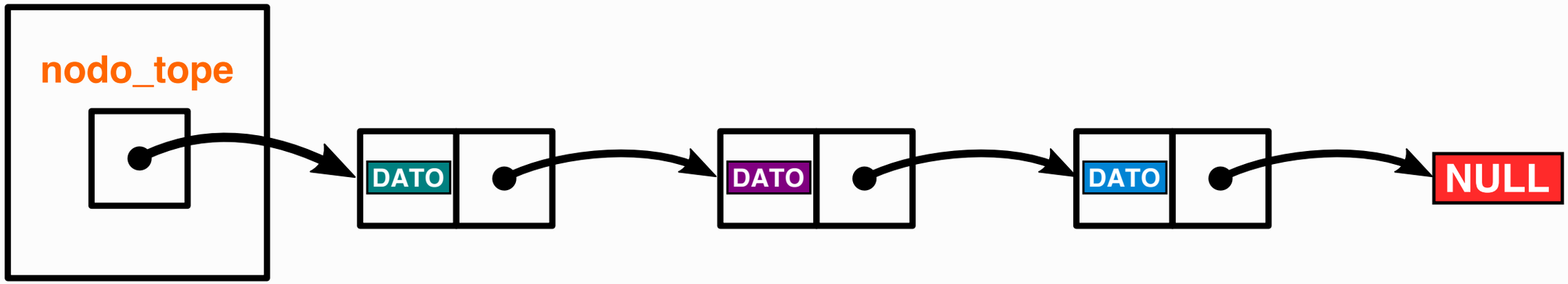
PUSH

PUSH(DATO), PUSH(DATO), PUSH(DATO)



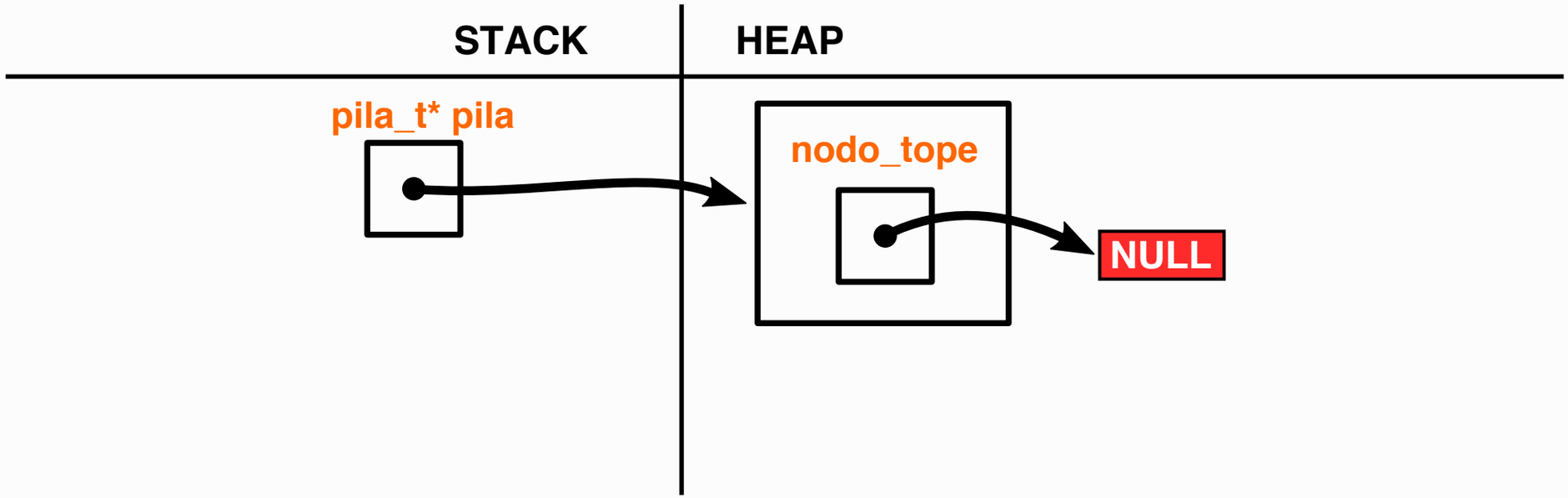
PUSH

PUSH(DATO), PUSH(DATO), PUSH(DATO)



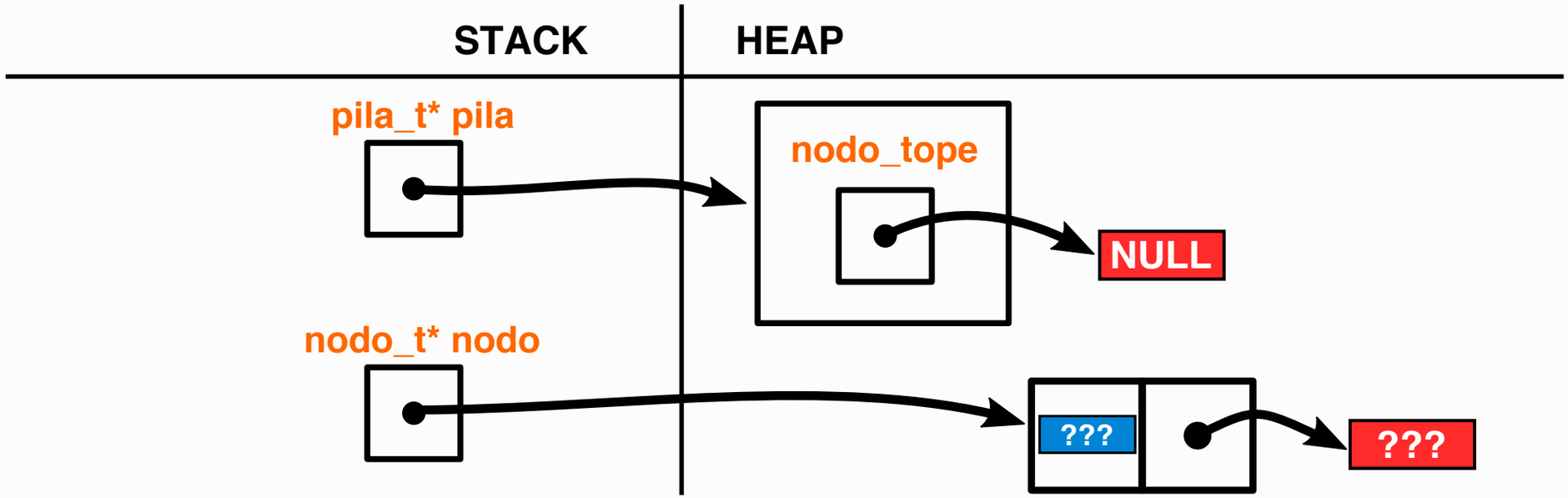
PUSH

```c
int pila_apilar(pila_t* pila, void* elemento){
  nodo_t* nodo = malloc(sizeof(nodo_t));
  nodo->siguiente = pila->nodo_tope;
  nodo->elemento = elemento;
  pila->nodo_tope = nodo;
  return 0;
}
```



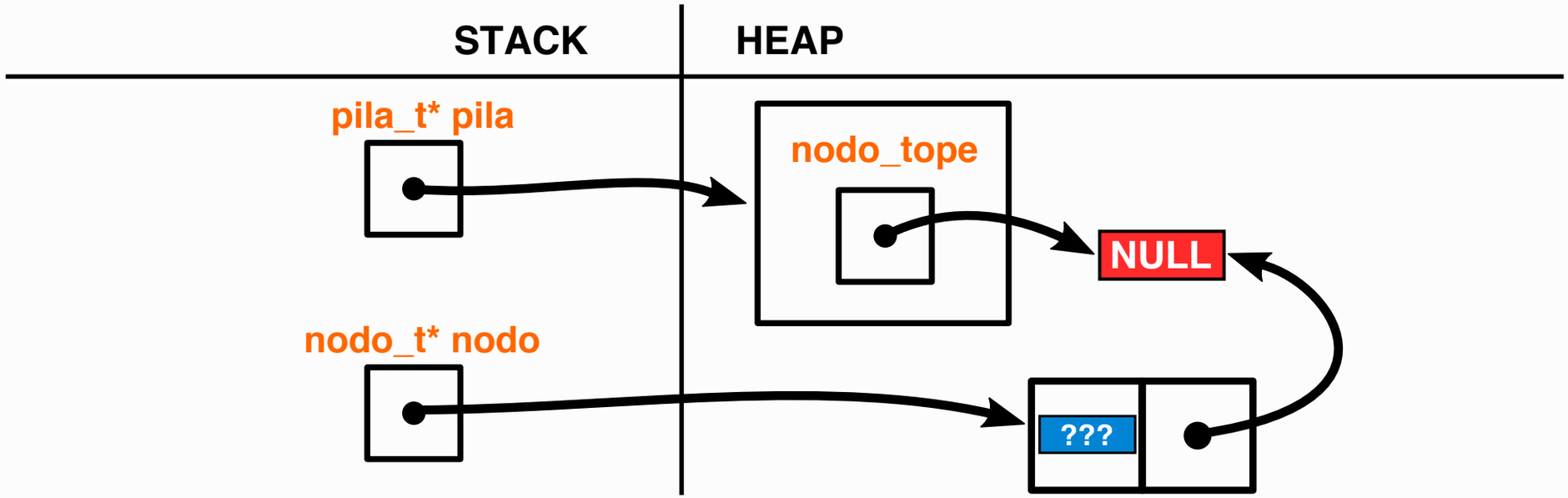
PUSH

```c
int pila_apilar(pila_t* pila, void* elemento){
  nodo_t* nodo = malloc(sizeof(nodo_t));
  nodo->siguiente = pila->nodo_tope;
  nodo->elemento = elemento;
  pila->nodo_tope = nodo;
  return 0;
}
```



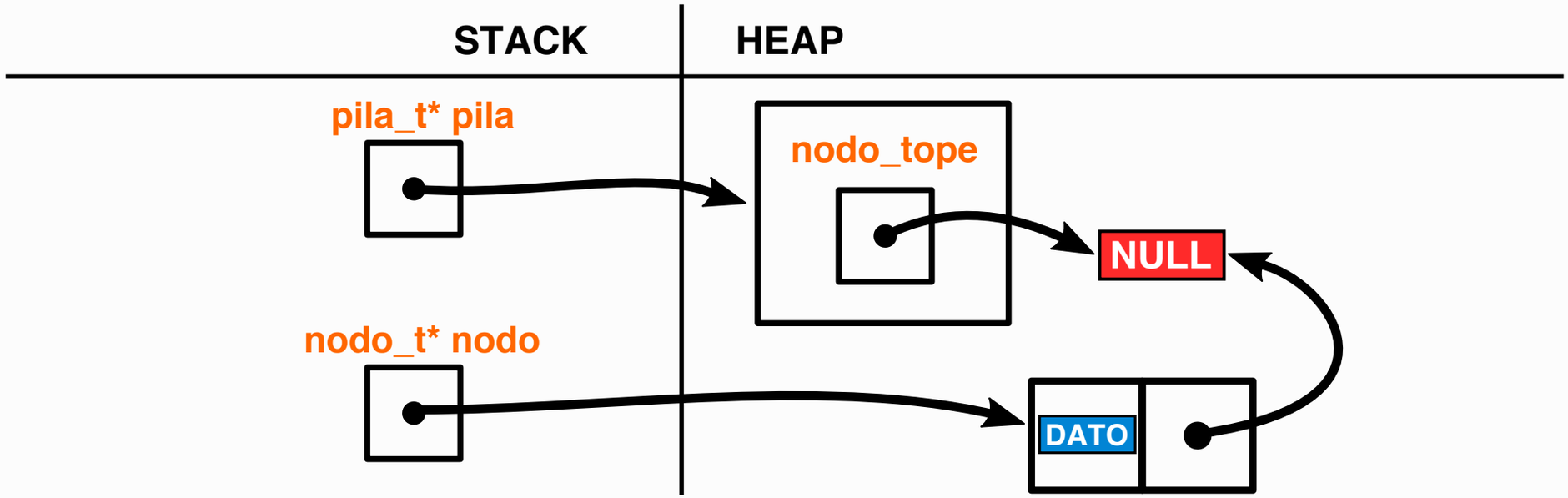
PUSH

```c
int pila_apilar(pila_t* pila, void* elemento){
  nodo_t* nodo = malloc(sizeof(nodo_t));
  nodo->siguiente = pila->nodo_tope;
  nodo->elemento = elemento;
  pila->nodo_tope = nodo;
  return 0;
}
```



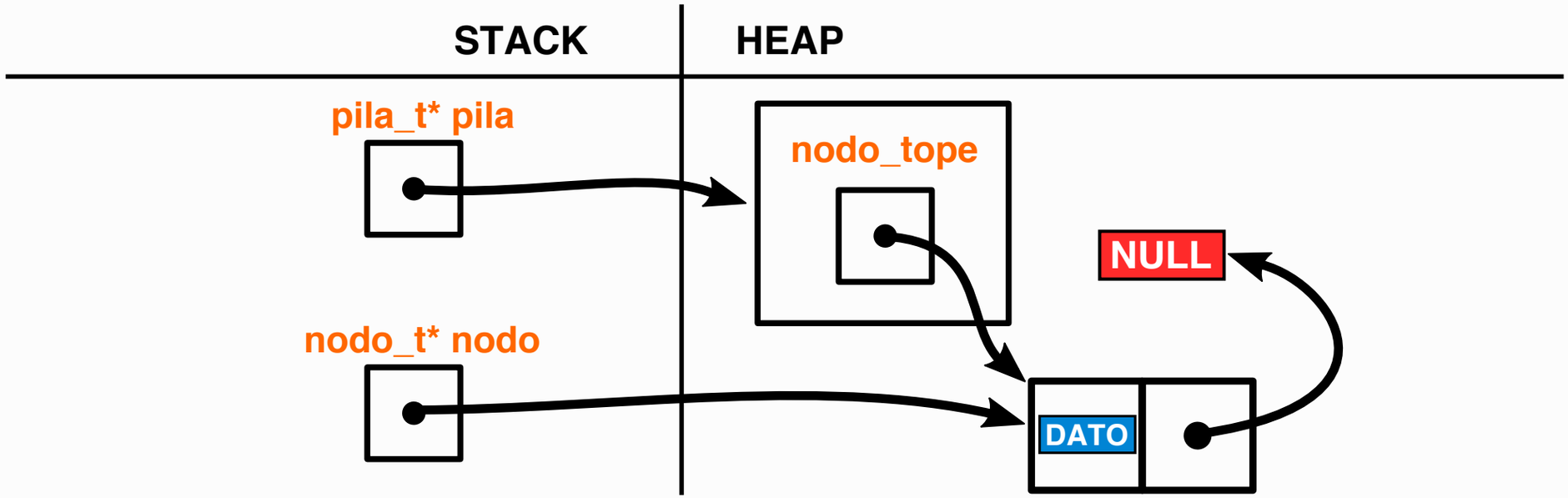
PUSH

```c
int pila_apilar(pila_t* pila, void* elemento){
  nodo_t* nodo = malloc(sizeof(nodo_t));
  nodo->siguiente = pila->nodo_tope;
  nodo->elemento = elemento;
  pila->nodo_tope = nodo;
  return 0;
}
```



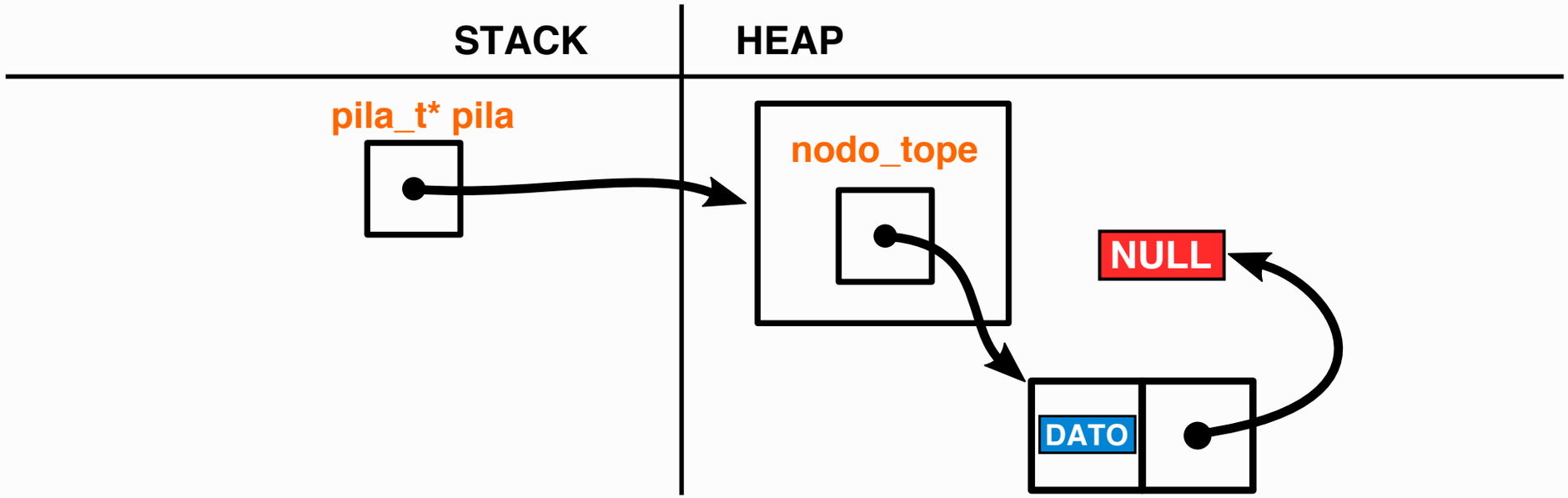
PUSH

```c
int pila_apilar(pila_t* pila, void* elemento){
  nodo_t* nodo = malloc(sizeof(nodo_t));
  nodo->siguiente = pila->nodo_tope;
  nodo->elemento = elemento;
  pila->nodo_tope = nodo;
  return 0;
}
```



PUSH

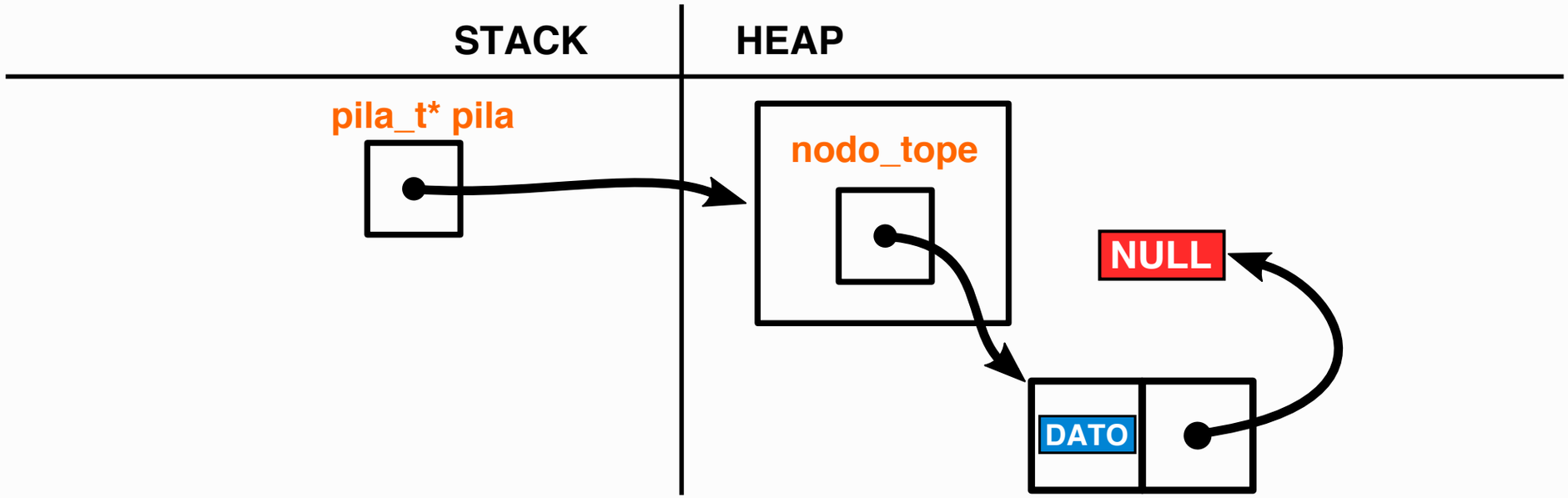
```c
int pila_apilar(pila_t* pila, void* elemento){
  nodo_t* nodo = malloc(sizeof(nodo_t));
  nodo->siguiente = pila->nodo_tope;
  nodo->elemento = elemento;
  pila->nodo_tope = nodo;
  return 0;
}
```



PUSH

```c
int pila_apilar(pila_t* pila, void* elemento){
  nodo_t* nodo = malloc(sizeof(nodo_t));
  nodo->siguiente = pila->nodo_tope;
  nodo->elemento = elemento;
  pila->nodo_tope = nodo;
  return 0;
}
```

¿QUÉ FALTA? ಠ_ಠ

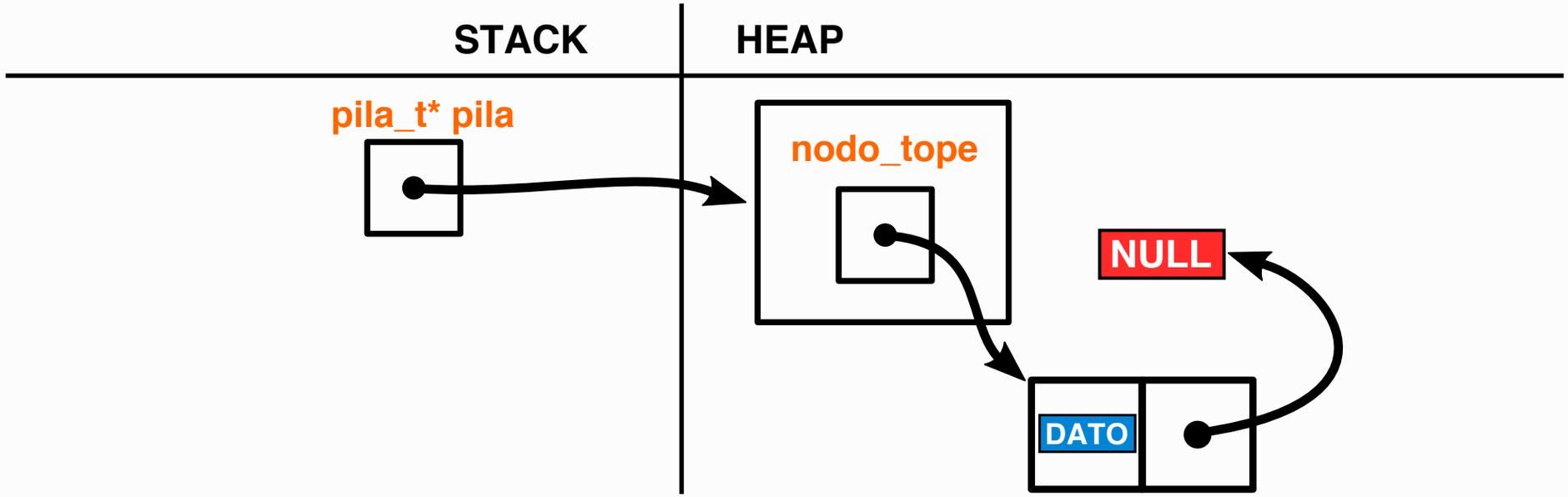


PUSH

```c
int pila_apilar(pila_t* pila, void* elemento){
  nodo_t* nodo = malloc(sizeof(nodo_t));
  nodo->siguiente = pila->nodo_tope;
  nodo->elemento = elemento;
  pila->nodo_tope = nodo;
  return 0;
}
```

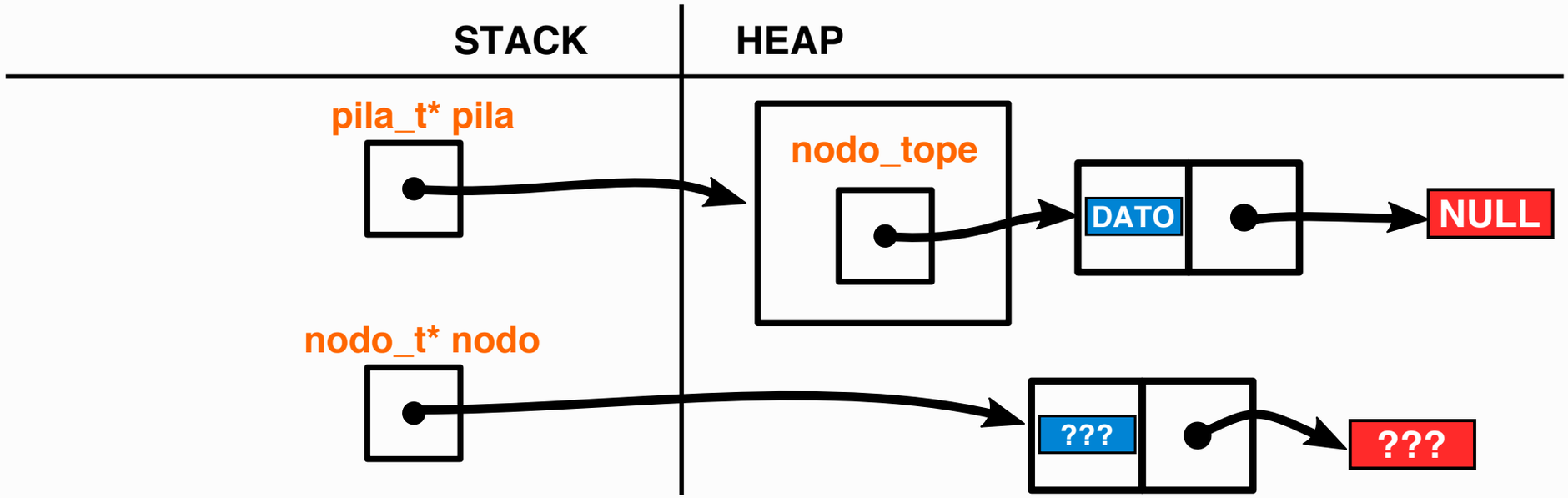
VERIFICAR NULL

¿QUÉ FALTA? ʚ_ɞ



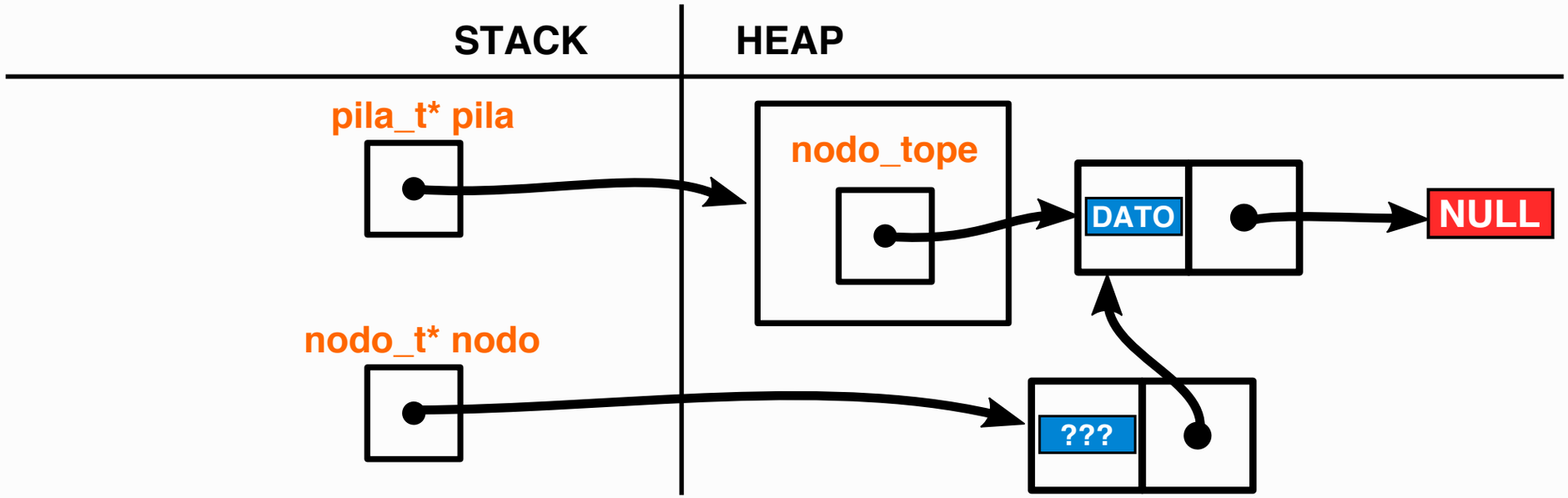
PUSH

```
int pila_apilar(pila_t* pila, void* elemento){
  nodo_t* nodo = malloc(sizeof(nodo_t));
  nodo->siguiente = pila->nodo_tope;
  nodo->elemento = elemento;
  pila->nodo_tope = nodo;
  return 0;
}
```



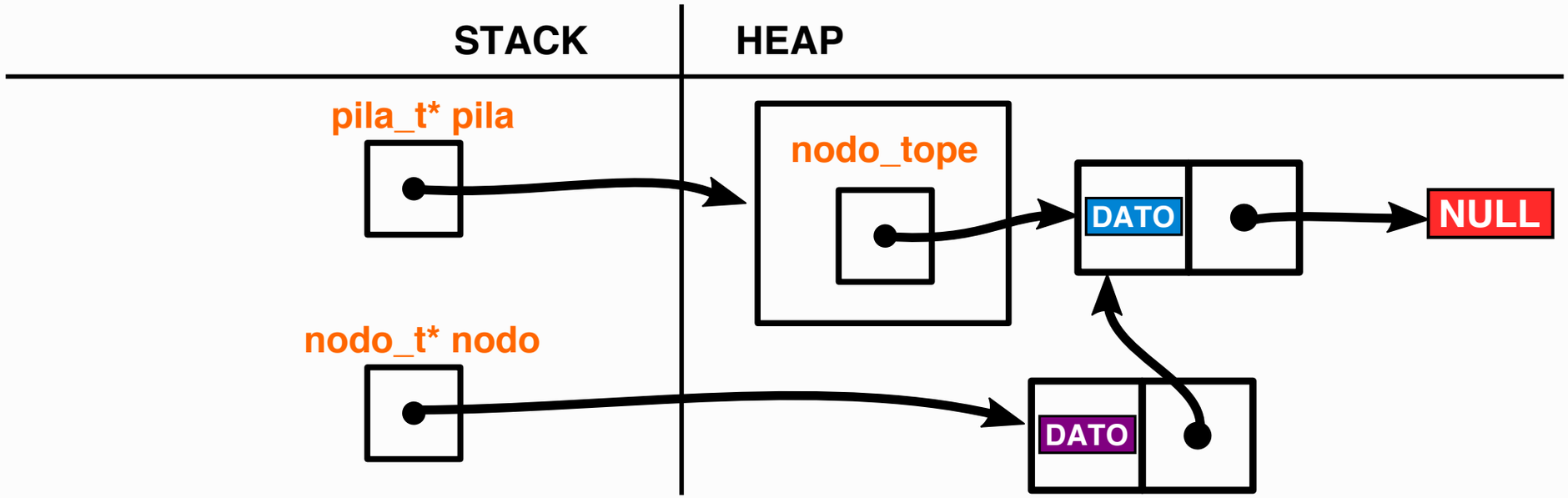
PUSH

```c
int pila_apilar(pila_t* pila, void* elemento){
  nodo_t* nodo = malloc(sizeof(nodo_t));
  nodo->siguiente = pila->nodo_tope;
  nodo->elemento = elemento;
  pila->nodo_tope = nodo;
  return 0;
}
```



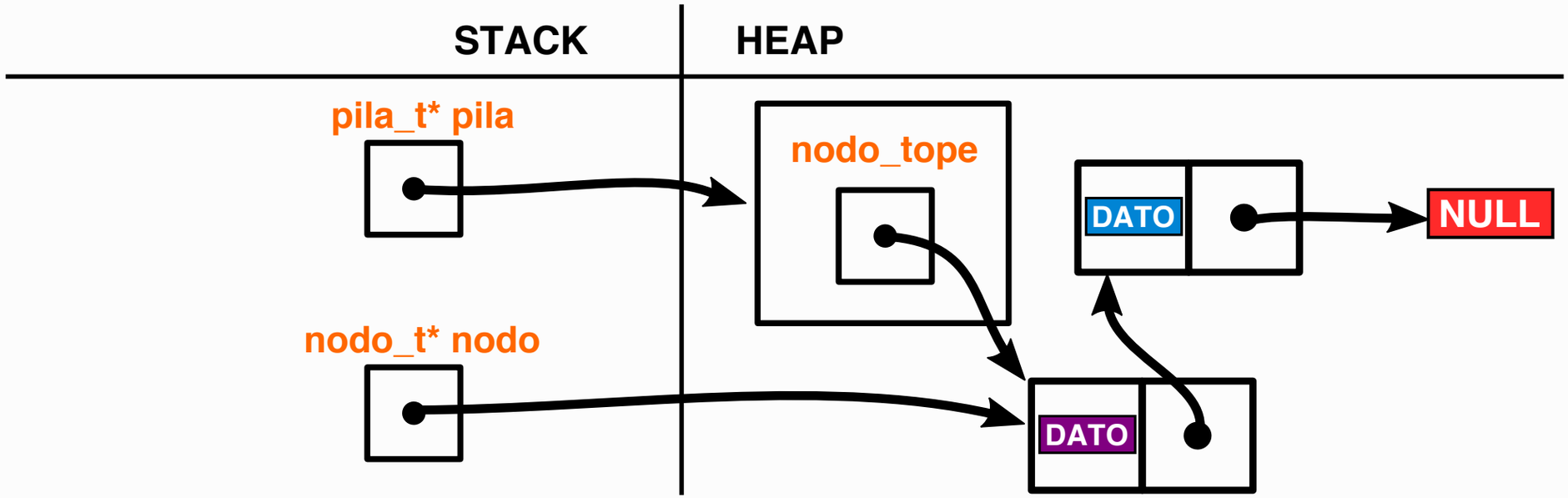
PUSH

```c
int pila_apilar(pila_t* pila, void* elemento){
  nodo_t* nodo = malloc(sizeof(nodo_t));
  nodo->siguiente = pila->nodo_tope;
  nodo->elemento = elemento;
  pila->nodo_tope = nodo;
  return 0;
}
```



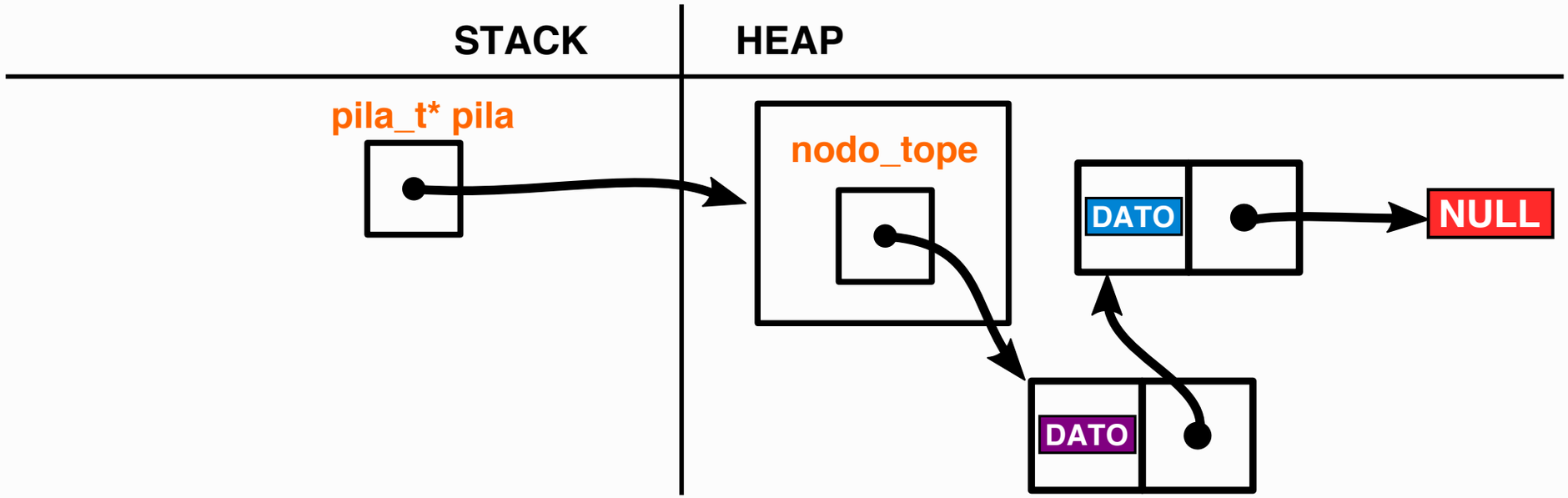
PUSH

```c
int pila_apilar(pila_t* pila, void* elemento){
  nodo_t* nodo = malloc(sizeof(nodo_t));
  nodo->siguiente = pila->nodo_tope;
  nodo->elemento = elemento;
  pila->nodo_tope = nodo;
  return 0;
}
```



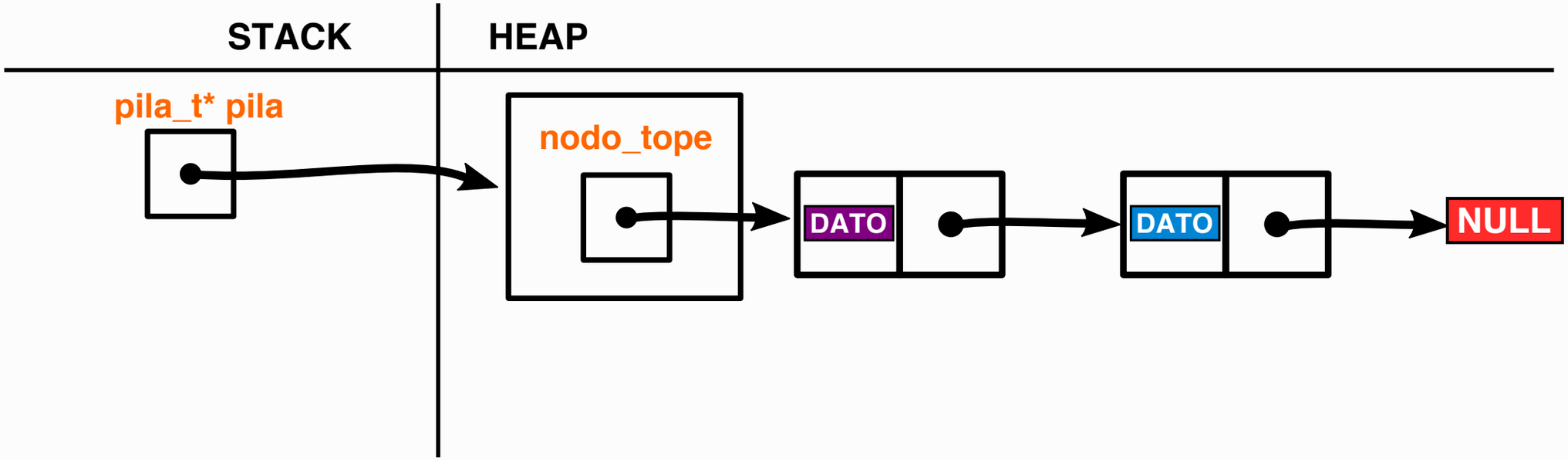
PUSH

```
int pila_apilar(pila_t* pila, void* elemento){
  nodo_t* nodo = malloc(sizeof(nodo_t));
  nodo->siguiente = pila->nodo_tope;
  nodo->elemento = elemento;
  pila->nodo_tope = nodo;
  return 0;
}
```



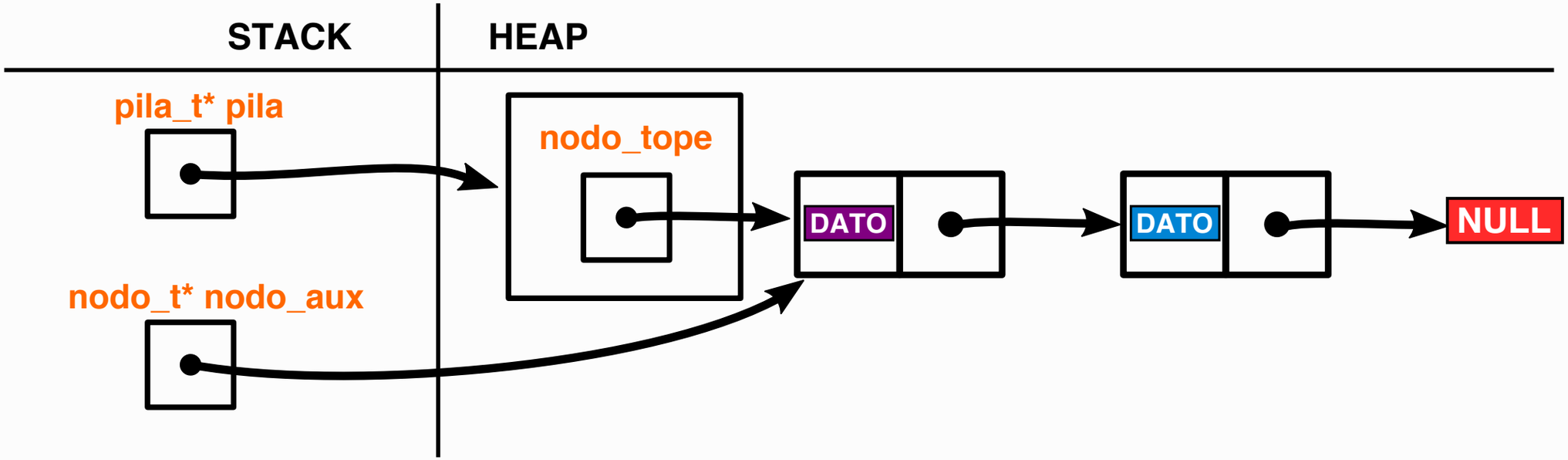
POP

```
int pila_desapilar(pila_t* pila){
   nodo_t* nodo_aux = (pila->nodo_tope);
   pila->nodo_tope = nodo_aux->siguiente;
   free(nodo_aux);
   return 0;
}
```



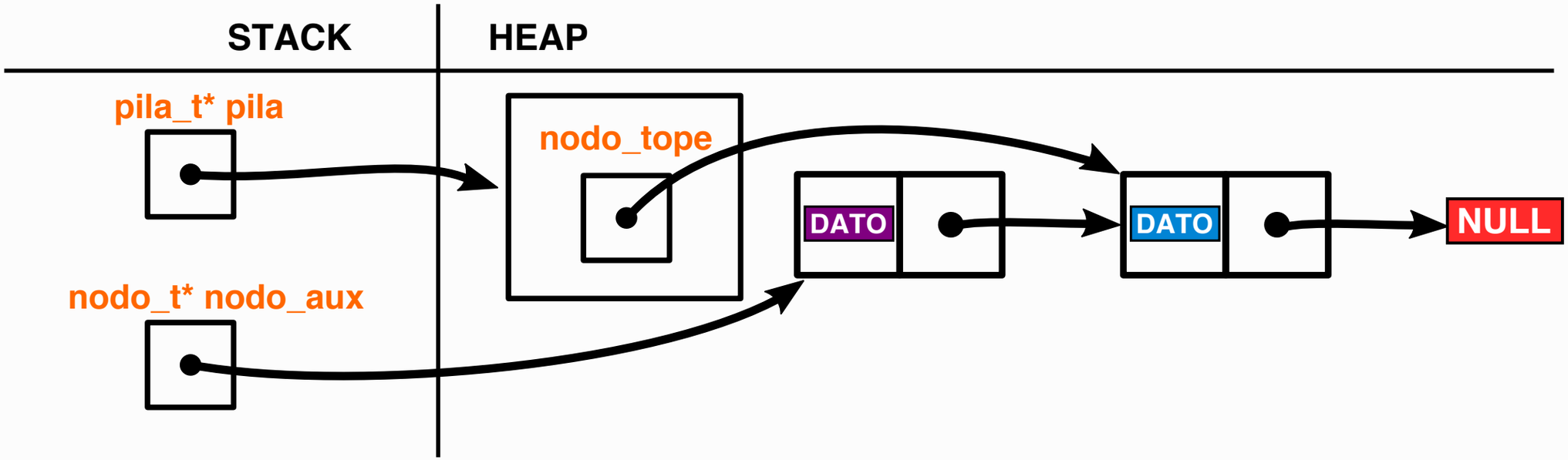
POP

```
int pila_desapilar(pila_t* pila){
  nodo_t* nodo_aux = (pila->nodo_tope);
  pila->nodo_tope = nodo_aux->siguiente;
  free(nodo_aux);
  return 0;
}
```



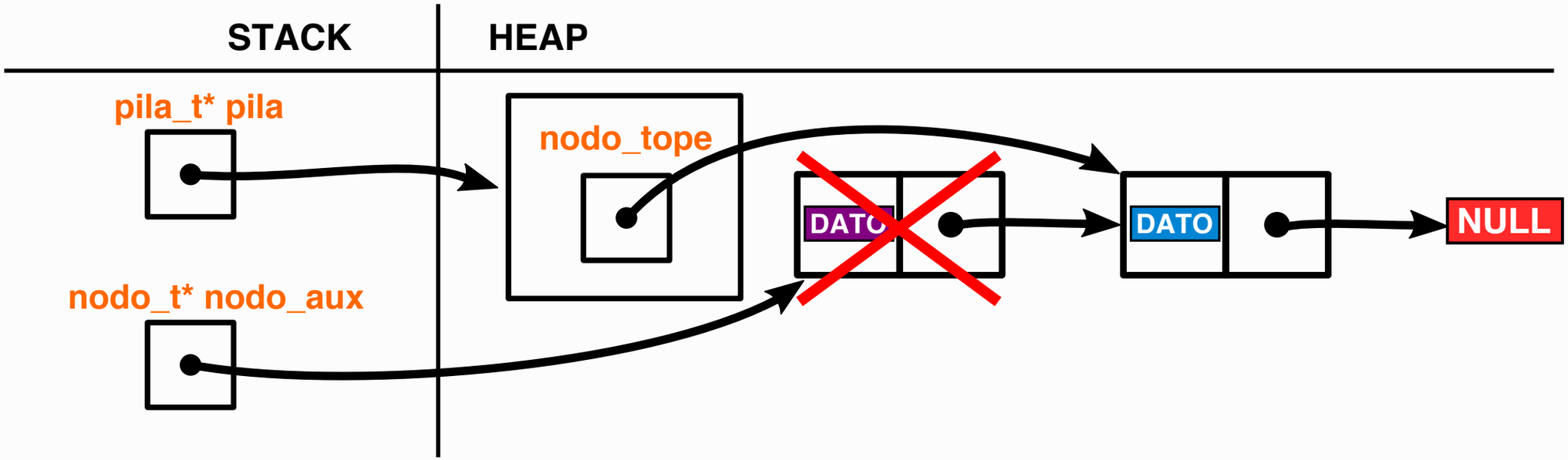
POP

```
int pila_desapilar(pila_t* pila){
   nodo_t* nodo_aux = (pila->nodo_tope);
   pila->nodo_tope = nodo_aux->siguiente;
   free(nodo_aux);
   return 0;
}
```



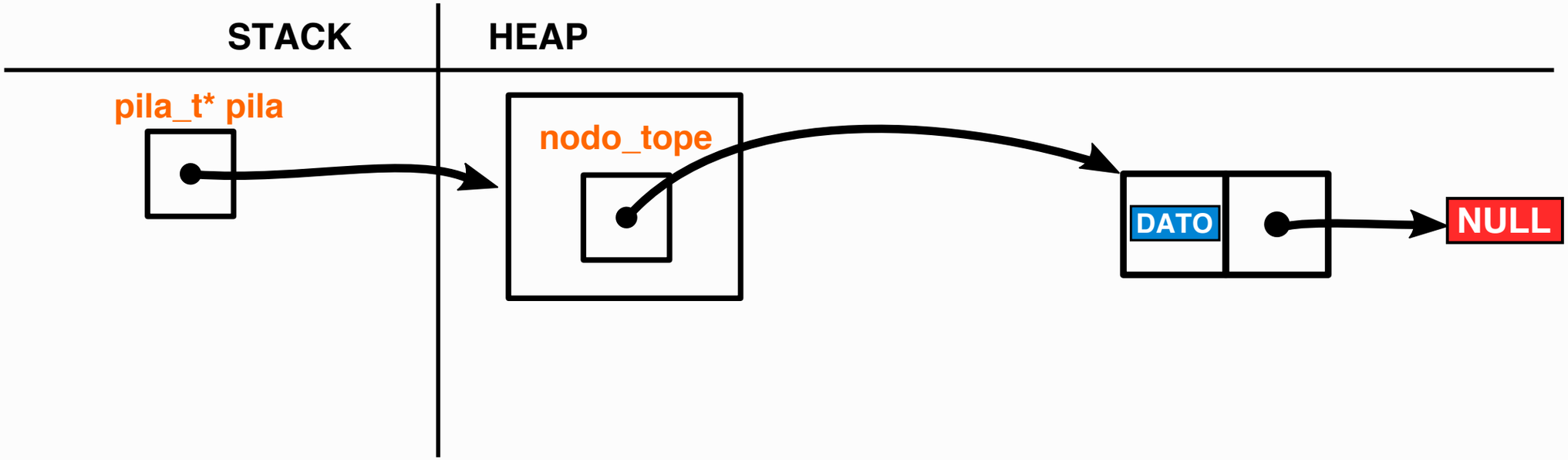
POP

```
int pila_desapilar(pila_t* pila){
   nodo_t* nodo_aux = (pila->nodo_tope);
   pila->nodo_tope = nodo_aux->siguiente;
   free(nodo_aux);
   return 0;
}
```



POP

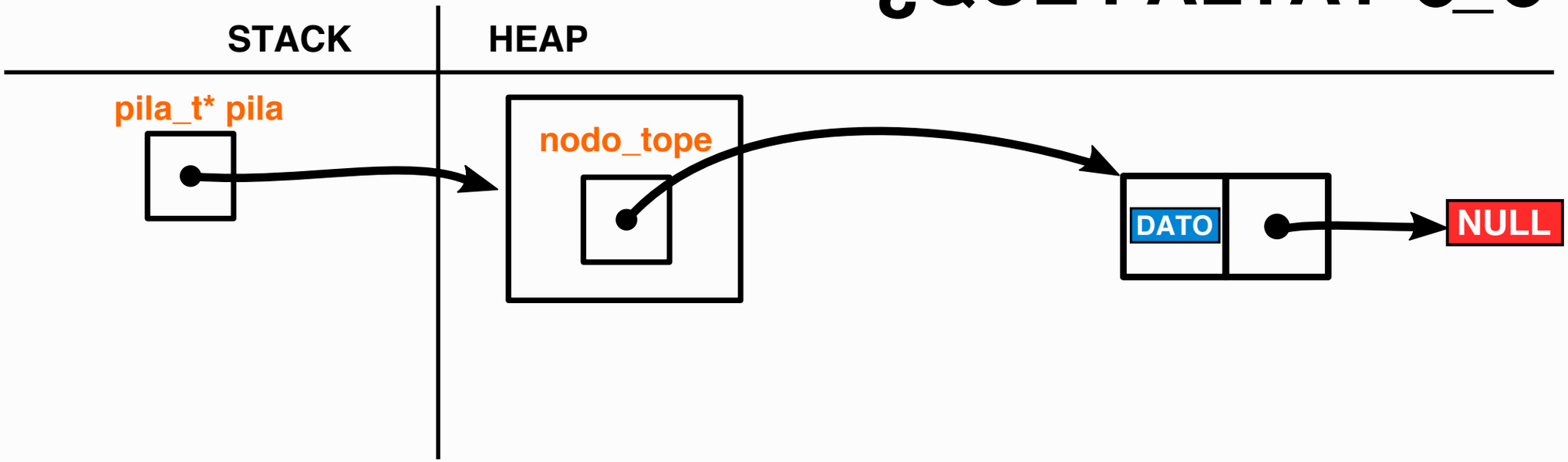
```
int pila_desapilar(pila_t* pila){
   nodo_t* nodo_aux = (pila->nodo_tope);
   pila->nodo_tope = nodo_aux->siguiente;
   free(nodo_aux);
   return 0;
}
```



POP

```c
int pila_desapilar(pila_t* pila){
  nodo_t* nodo_aux = (pila->nodo_tope);
  pila->nodo_tope = nodo_aux->siguiente;
  free(nodo_aux);
  return 0;
}
```

¿QUÉ FALTA? ಠ_ಠ

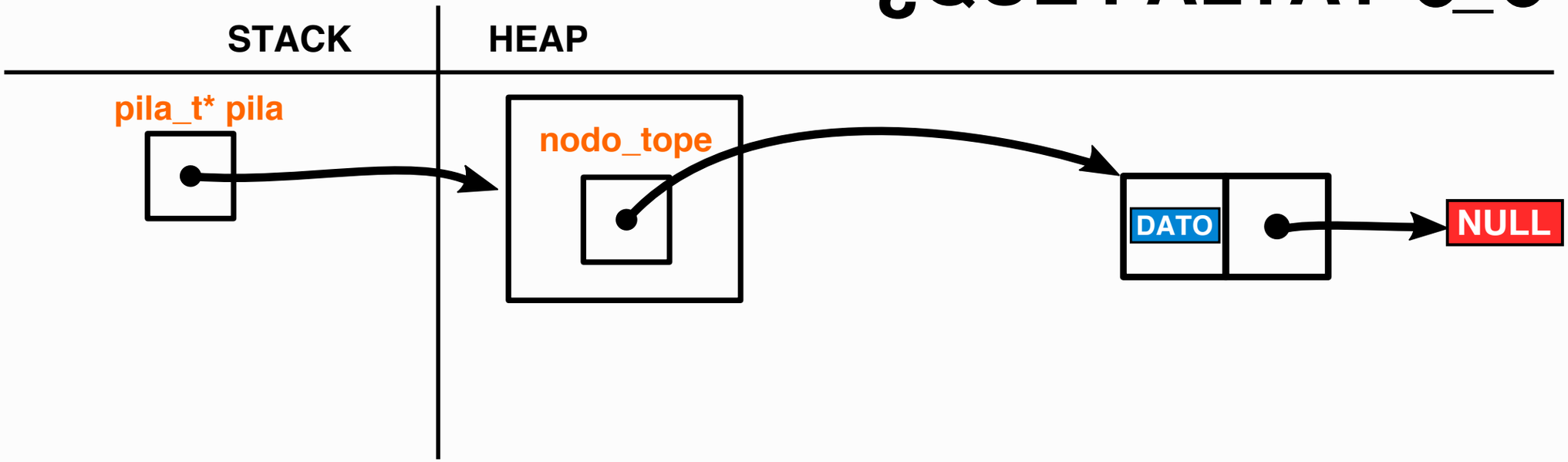


POP

```
int pila_desapilar(pila_t* pila){
   nodo_t* nodo_aux = (pila->nodo_tope);
   pila->nodo_tope = nodo_aux->siguiente;
   free(nodo_aux);
   return 0;
}
```

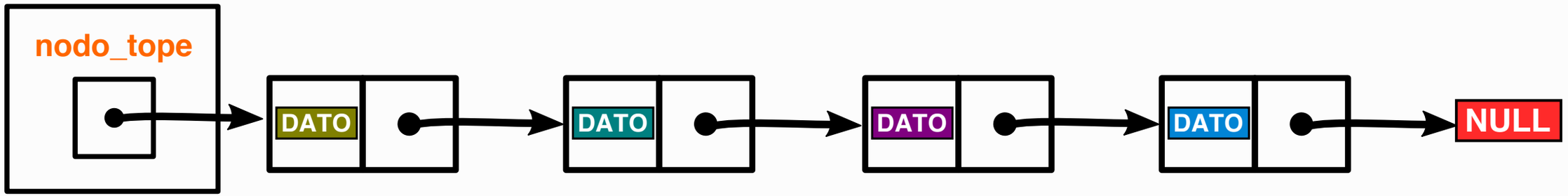
VERIFICAR NULL

¿QUÉ FALTA? ಠ_ಠ



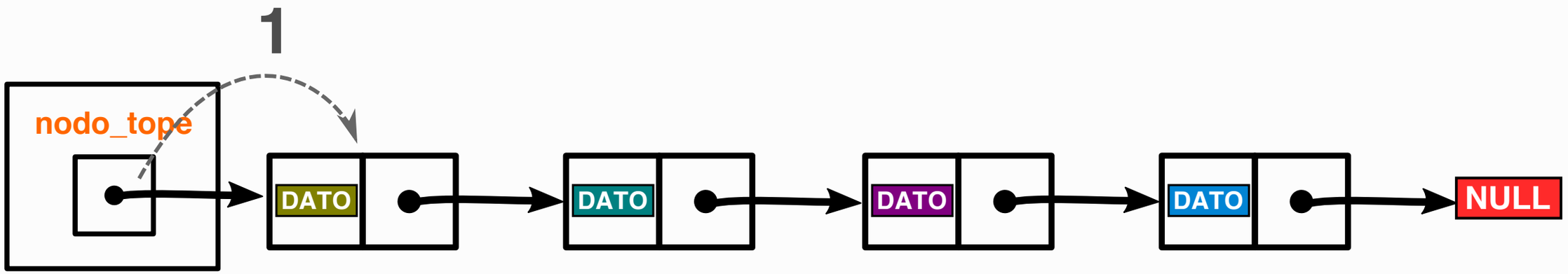
TAMAÑO, TOPE, VACÍA

¿CUÁNTOS ELEMENTOS HAY?



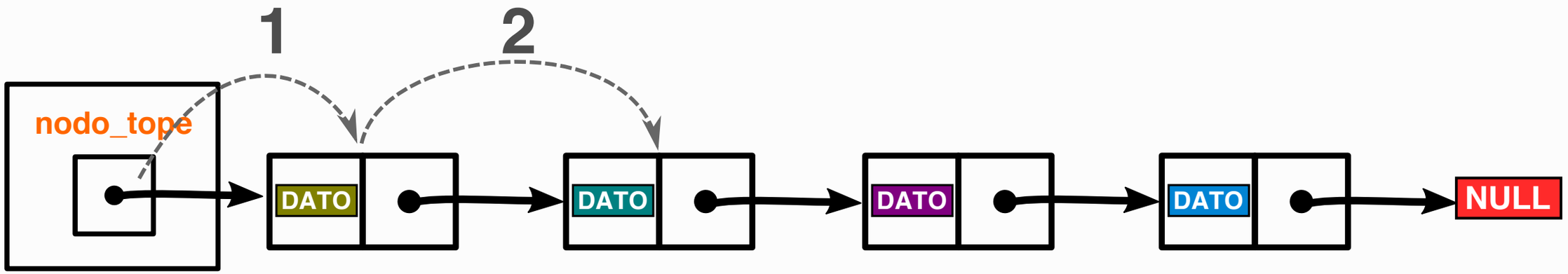
TAMAÑO, TOPE, VACÍA

¿CUÁNTOS ELEMENTOS HAY?



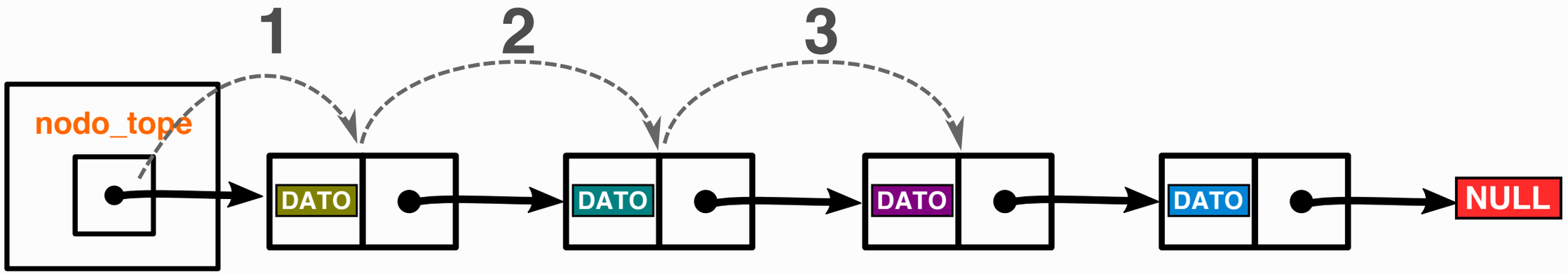
TAMAÑO, TOPE, VACÍA

¿CUÁNTOS ELEMENTOS HAY?



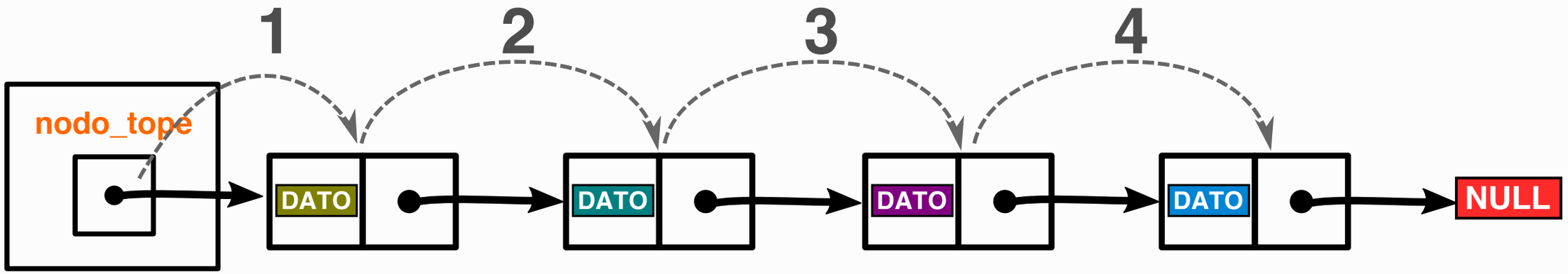
TAMAÑO, TOPE, VACÍA

¿CUÁNTOS ELEMENTOS HAY?



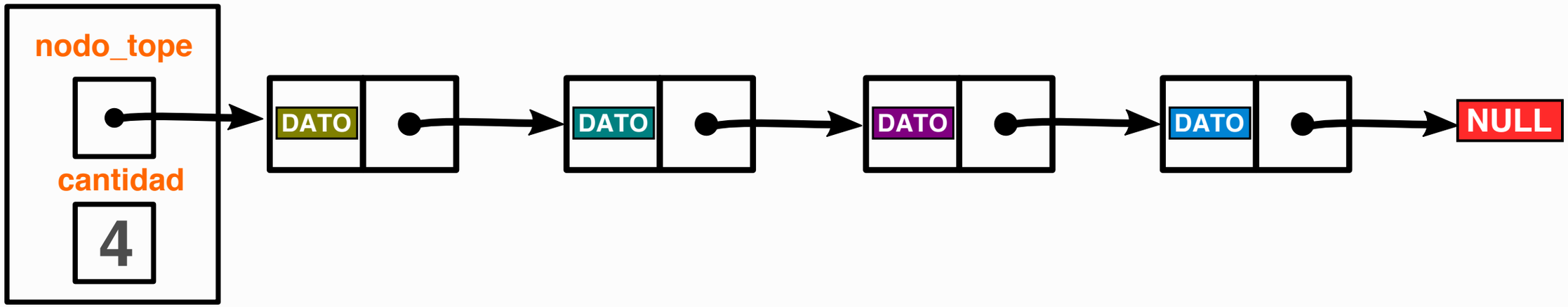
TAMAÑO, TOPE, VACÍA

¿CUÁNTOS ELEMENTOS HAY?



TAMAÑO, TOPE, VACÍA

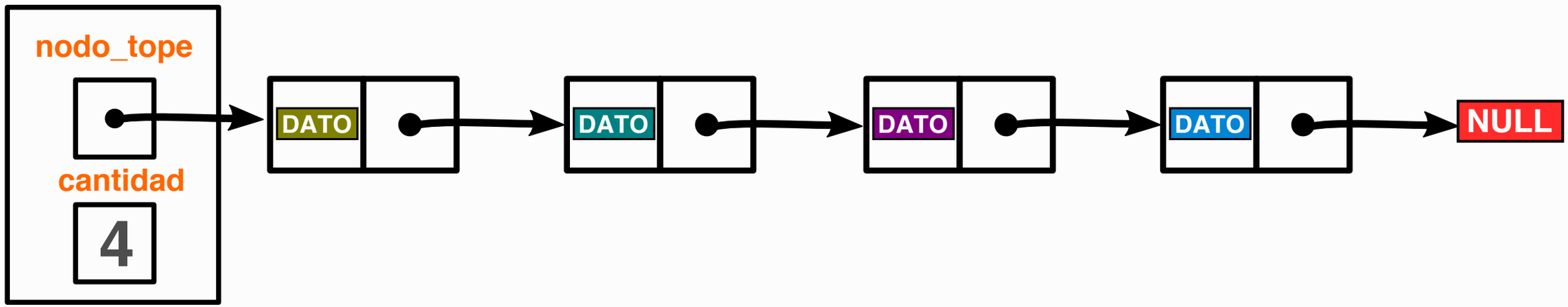
¿CUÁNTOS ELEMENTOS HAY?



TAMAÑO, TOPE, VACÍA

```c
bool pila_vacia(pila_t* pila){
  if(!pila)
    return true;
  return (pila->nodo_tope == NULL);
}

void* pila_tope(pila_t* pila){
  if(!pila || pila_vacia(pila))
    return NULL;
  return pila->nodo_tope->elemento;
}
```

```c
int pila_cantidad(pila_t* pila){
  if(!pila)
    return 0;
  return pila->cantidad;
}
```



DESTRUIR

```
void pila_destruir(pila_t* pila){
  if(!pila)
    return;
  while (!pila_vacia(pila) && pila_desapilar(pila)==0);
  free(pila);
}
```

FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

¿PREGUNTAS?

FIN

FACULTAD
DE INGENIERIA
Universidad de Buenos Aires