Lenguaje de Programación 1

Pilas

Algoritmos y Estructura de Datos.
Una perspectiva en C - Capitulo 10
Aguilar & Martínez

Pilas - Stack

Contenido

1. Concepto de Pilas
2. Pila implementado con arreglos
3. Pila implementado como una lista enlazada
4. Evaluación de expresiones aritméticas mediante pilas

Conceptos Claves

Concepto de pila

Pila basadas en array

Pila basada en Lista enlazadas

Introducción

- Se estudia la estructura de datos Pila utilizada frecuentemente en los programas más usuales y en la vida diaria.
- La organización de una pila es muy conocida: una estructura de datos que almacena y recupera sus elementos atendiendo a un estricto orden.
- Las pilas se conocen también como estructuras LIFO (Last-in, first-out).
- Todas las inserciones y retiradas de elementos se realiza por el mismo extremo denominado cima de la pila.

Concepto de Pila

- Una **pila** (*stack*) es una colección ordenada de elementos a los que sólo se puede acceder por un único lugar o extremo de la pila.
- Los elementos de la pila se añaden o quitan de la misma sólo por su parte superior (**cima**) de la pila.
- Ejemplo: pila de platos, pila de libros, organización de la memoria libre, y otros

Operaciones

INSERTAR O PONER (*PUSH*)

Añade un elemento en la cima de la Pila

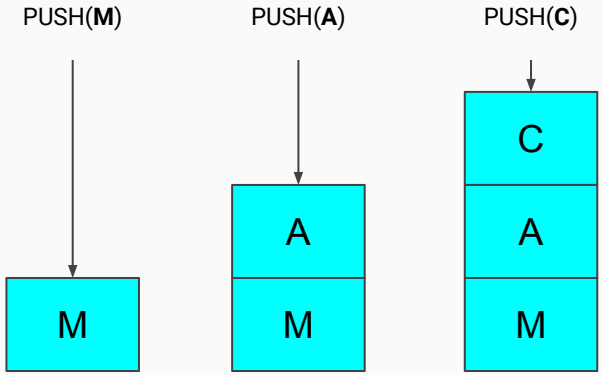
QUITAR O RETIRAR (*POP*)

Elimina o Saca un elemento de la Pila

Operaciones

INSERTAR O PONER (*PUSH*)

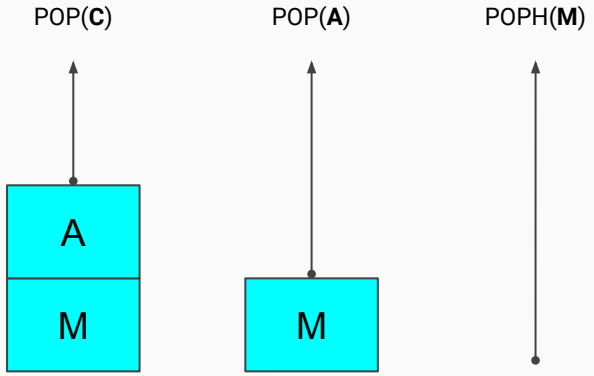
Añade un elemento en la cima de la Pila



Entrada MAC

QUITAR O RETIRAR (*POP*)

Elimina o Saca un elemento de la Pila



Salida CAM

Implementaciones

Mediante arrays

- Dimensión de la pila es fija
- La pila puede estar **vacía** o **llena**

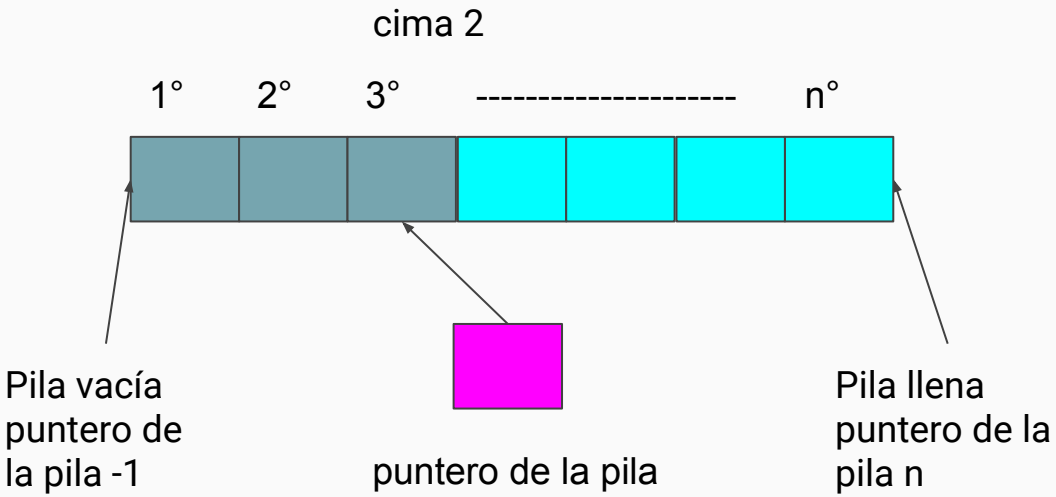
Mediante listas enlazadas

- Dimensión de la pila es dinámica
- El límite es el tamaño de la memoria RAM
- La pila puede estar vacía

Pilas mediante arrays

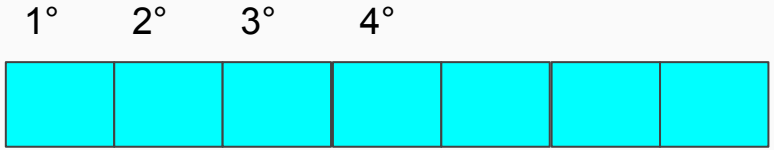
Componentes

1. Arreglo de n elementos
2. Contador de elementos o puntero a la pila



Ejemplo básico de funcionamiento

Pila vacía
PunteroPila = -1



Push (50)
PunteroPila = 0

Push (25)
PunteroPila = 1

Pop (25)
PunteroPila = 0

Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. `pilaLlena()` - Verificar Pila Llena
4. `insertarPila()`
5. `quitarPila()`
6. `limpiarPila` - Limpiar Pila
7. `cima()`- Devuelve el valor de cima sin quitar

```
/*archivo pilaarray.h*/

#define TAMPILA 100

struct PILA{
    TipoDato listaPila[TAMPILA];
    int cima;
}

typedef struct PILA Pila;

/*Operaciones sobre la Pila*/
void crearPila(Pila * pila);
void insertarPila(Pila * pila, TipoDato elemento);
TipoDato quitarPila(Pila * pila);
void limpiarPila(Pila * pila);
TipoDato cimaPila(Pila * pila);
int pilaLlena(Pila  pila);
int pilaVacia(Pila  pila);
```

Funciones

1. `creaPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. `pilaLlena()` - Verificar Pila Llena
4. `insertarPila()`
5. `quitarPila()`
6. `limpiarPila` - Limpiar Pila
7. `cima()`- Devuelve el valor de cima sin quitar

/*archivo pilaarray.c*/

```
/*Crea una Pila nueva*/
void crearPila(Pila * pila){
    pila -> cima = -1;
}
```

```
/*Verifica si la Pila está vacía*/
int pilaVacia(Pila pila){
    return pila.cima == -1;
}
```

```
/*Verifica si la Pila está llena*/
int pilaLlena(Pila pila){
    return pila.cima == TAMPILA - 1;
}
```

```
/*Vacía la Pila*/
int limpiarPila(Pila * pila){
    pila -> cima = -1;
}
```

Funciones

1. `creaPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. `pilaLlena()` - Verificar Pila Llena
4. `insertarPila()`
5. `quitarPila()`
6. `limpiarPila` - Limpiar Pila
7. `cima()`- Devuelve el valor de cima sin quitar

/*archivo pilaarray.c*/

```
/*Insertar dato en una Pila*/
void insertarPila(Pila * pila, TipoDato elemento){
    if(pilaLlena(*pila)){
        puts("Desbordamiento de pila\n")
        exit(1);
    }
    pila -> cima++;
    pila -> listaPila[pila -> cima] = elemento;
}
```

```
/*Quita dato de una Pila*/
TipoDato quitarPila(Pila * pila){
    TipoDato temporal;
    if(pilaVAcia(*pila)){
        puts("Pila Vacia\n")
        exit(1);
    }
    temporal = pila -> listaPila[pila -> cima];
    pila -> cima--;
    return temporal;
}
```

Funciones

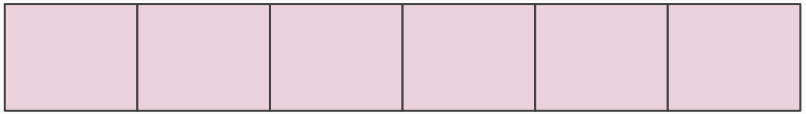
1. `creaPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. `pilaLlena()` - Verificar Pila Llena
4. `insertarPila()`
5. `quitarPila()`
6. `limpiarPila` - Limpiar Pila
7. `cima()`- Devuelve el valor de cima sin quitar

/*archivo pilaarray.c*/

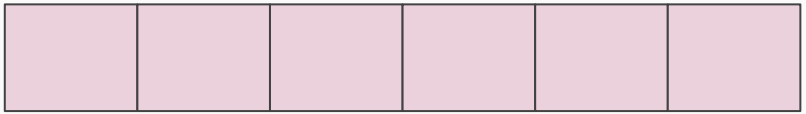
```
/*Devuelve el elemento de top de la Pila*/
TipoDato cimaPila(Pila pila){
    if(pilaVacia(pila)){
        puts("Pila Vacía\n")
        exit(1);
    }
    return pila.listaPila[pila.cima]
}
```

Caso de uso: Verificar si una cadena de caracteres es `palíndromo`

zorra arroz



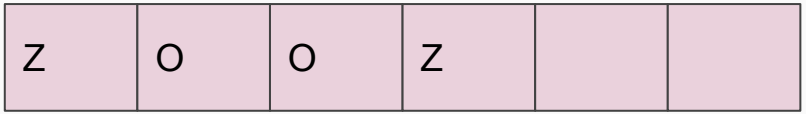
palabra



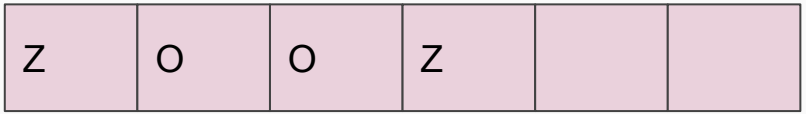
listaPila

Caso de uso: Verificar si una cadena de caracteres es `palíndromo`

ZOOZ



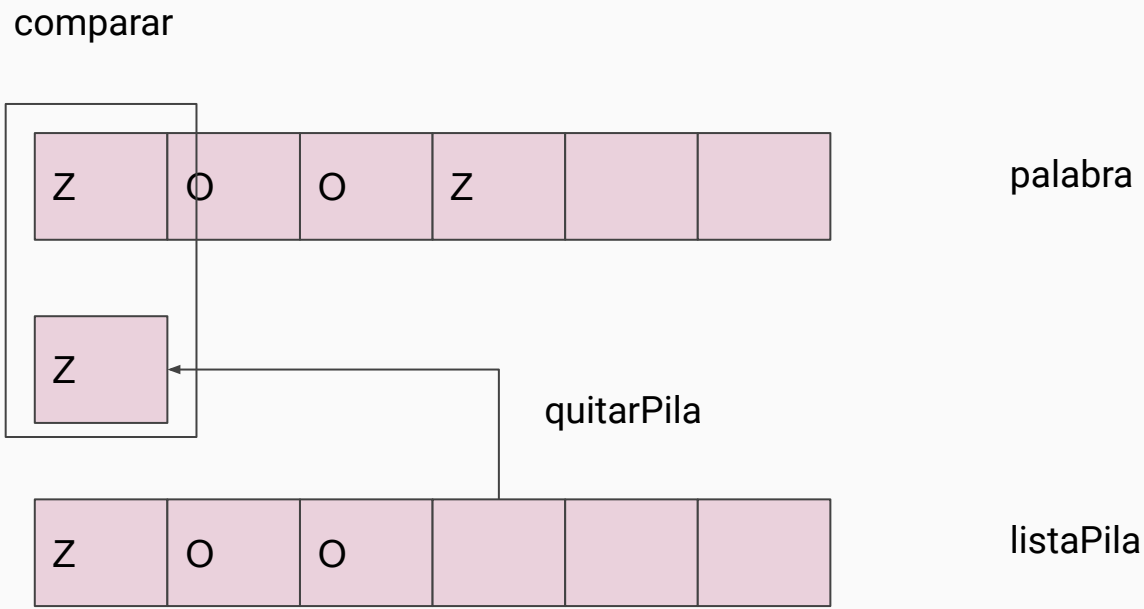
palabra



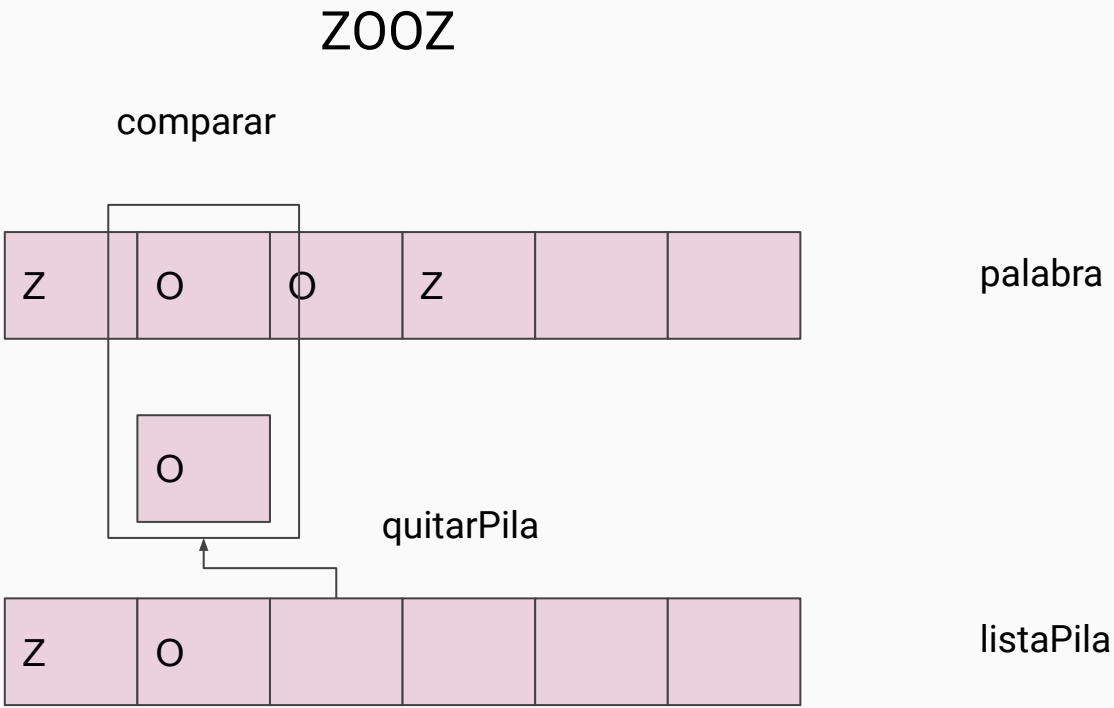
listaPila

Caso de uso: Verificar si una cadena de caracteres es `palíndromo`

ZOOZ



Caso de uso: Verificar si una cadena de caracteres es `palíndromo`



Caso de uso: Verificar si una cadena de caracteres es **palíndromo**

```
/*Archivo paldromo.c*/
typedef char TipoDato;  /*definimos tipo de dato a manejar*/

#include <stdio.h>
#include "pilaarray.h" /*usamos las funciones de pila*/
#include <ctype.h>

void main(void){
      char palabra[100], ch;
      Pila pila;
      int j, esPaldro;

      crearPila(&pila);
      /*se lee la palabra o frase*/
      do{
            puts("\n Palabra a comprobar");
            for (j = 0; ch = getchar() != '\n'){
                  palabra[j++] = ch;
                  insertarPila(&pila, ch);
            }
            palabra[j] = '\0';
            esPaldro = 1;
            for (j = 0; esPaldro && !pilaVacia(pila); ){
                  esPaldr = palabra[j++] ==  quitarPila(&pila);
            }
            limpiarPila(&pila);
```

Caso de uso: Verificar si una cadena de caracteres es palíndromo

```
/*Archivo paldromo.c*/
typedef char TipoDato;  /*definimos tipo de dato a manejar*/

#include <stdio.h>
#include "pilaarray.h" /*usamos las funciones de pila*/
#include <ctype.h>

void main(void){
      char palabra[100], ch;
      Pila pila;
      int j, esPaldro;

      crearPila(&pila);
      /*se lee la palabra o frase*/
      do{
            puts("\n Palabra a comprobar");
            for (j = 0; (ch = getchar()) != '\n'; ){
                  palabra[j++] = ch;
                  insertarPila(&pila, ch);
            }
            palabra[j] = '\0';
            esPaldro = 1;
            for (j = 0; esPaldro && !pilaVacia(pila); ){
                  esPaldr = palabra[j++] ==  quitarPila(&pila);
            }
            limpiarPila(&pila);
```

```
/*Crea una Pila nueva*/
void crearPila(Pila *
pila){
      pila -> cima = -1;
}
```

Caso de uso: Verificar si una cadena de caracteres es palíndromo

```
/*Archivo paldromo.c*/
typedef char TipoDato;  /*definimos tipo de dato a manejar*/

#include <stdio.h>
#include "pilaarray.h" /*usamos las funciones de pila*/
#include <ctype.h>

void main(void){
      char palabra[100], ch;
      Pila pila;
      int j, esPaldro;

      crearPila(&pila);
      /*se lee la palabra o frase*/
      do{
            puts("\n Palabra a comprobar");
            for (j = 0; (ch = getchar()) != '\n'; ){
                  palabra[j++] = ch;
                  insertarPila(&pila, ch);
            }
            palabra[j] = '\0';
            esPaldro = 1;
            for (j = 0; esPaldro && !pilaVacia(pila); ){
                  esPaldr = palabra[j++] ==  quitarPila(&pila);
            }
            limpiarPila(&pila);
```

```
/*Insertar dato en una Pila*/
void insertarPila(Pila * pila, TipoDato
elemento){
      if(pilaLlena(*pila)){
            puts("Desbordamiento de pila\n")
            exit(1);
      }
      pila -> cima++;
      pila -> listaPila[pila -> cima] = elemento;
}
```

Caso de uso: Verificar si una cadena de caracteres es palíndromo

```
/*Archivo paldromo.c*/
typedef char TipoDato;  /*definimos tipo de dato a manejar*/

#include <stdio.h>
#include "pilaarray.h" /*usamos las funciones de pila*/
#include <ctype.h>

void main(void){
      char palabra[100], ch;
      Pila pila;
      int j, esPaldro;

      crearPila(&pila);
      /*se lee la palabra o frase*/
      do{
            puts("\n Palabra a comprobar");
            for (j = 0; (ch = getchar()) != '\n'; ){
                  palabra[j++] = ch;
                  insertarPila(&pila, ch);
            }
            palabra[j] = '\0';
            esPaldro = 1;
            for (j = 0; esPaldro && !pilaVacia(pila); ){
                  esPaldr = (palabra[j++] ==  quitarPila(&pila));
            }
            limpiarPila(&pila);
```

```
/*Quita dato de una Pila*/
TipoDato quitarPila(Pila * pila){
      TipoDato temporal;
      if(pilaVAcia(*pila)){
            puts("Pila Vacia\n")
            exit(1);
      }
      temporal = pila -> listaPila[pila ->
cima];
      pila -> cima--;
      return temporal;
}
```

Caso de uso: Verificar si una cadena de caracteres es palíndromo

```
/*Archivo paldromo.c*/
typedef char TipoDato;  /*definimos tipo de dato a manejar*/

#include <stdio.h>
#include "pilaarray.h" /*usamos las funciones de pila*/
#include <ctype.h>

void main(void){
      char palabra[100], ch;
      Pila pila;
      int j, esPaldro;

      crearPila(&pila);
      /*se lee la palabra o frase*/
      do{
            puts("\n Palabra a comprobar");
            for (j = 0; (ch = getchar()) != '\n'; ){
                  palabra[j++] = ch;
                  insertarPila(&pila, ch);
            }
            palabra[j] = '\0';
            esPaldro = 1;
            for (j = 0; esPaldro && !pilaVacia(pila); ){
                  esPaldr = (palabra[j++] ==  quitarPila(&pila));
            }
            limpiarPila(&pila);
```

```
/*Vacía la Pila*/
int limpiarPila(Pila * pila){
      pila -> cima = -1;
}
```

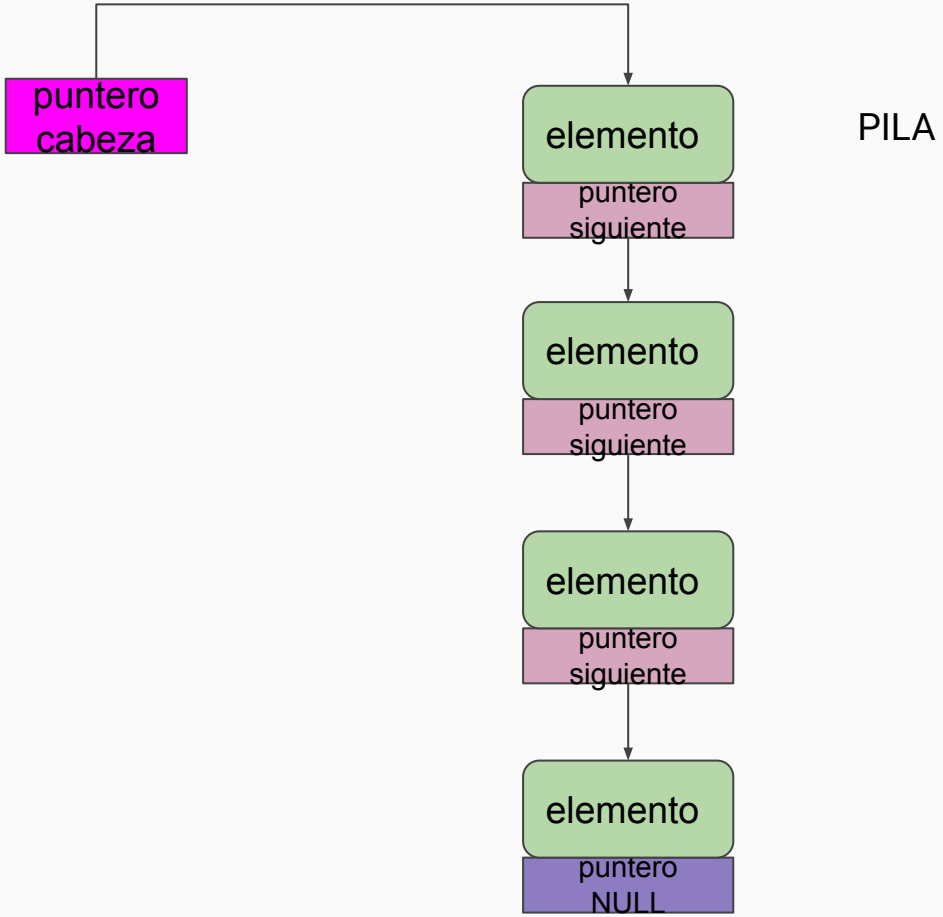
Caso de uso: Verificar si una cadena de caracteres es palíndromo

```
/*Archivo paldromo.c*/
typedef char TipoDato;  /*definimos tipo de dato a r

#include <stdio.h>
#include "pilaarray.h" /*usamos las funciones de
#include <ctype.h>

void main(void){
      char palabra[100], ch;
      Pila pila;
      int j, esPaldro;

      crearPila(&pila);
      /*se lee la palabra o frase*/
      do{
            puts("\n Palabra a comprobar");
            for (j = 0; ch = getchar() != '\n'){
                  palabra[j++] = ch;
                  insertarPila(&pila, ch);
            }
            palabra[j] = '\0';
            esPaldro = 1;
            for (j = 0; esPaldro && !pilaVacia(pila); ){
                  esPaldr = palabra[j++] ==  quitarPila(&pila);
            }
            limpiarPila(&pila);
            if (esPaldro){
                  printf("\n La palabra %s
                              es palindrome", palabra);
            }else{
                  printf("\n La palabra %s
                              no es palindo", palabra);
            }

            printf("\n ¿ Otra palabra ?: ");
            scanf("%c%*c", &ch);
      }while(tolower(ch) == 's');

}
```

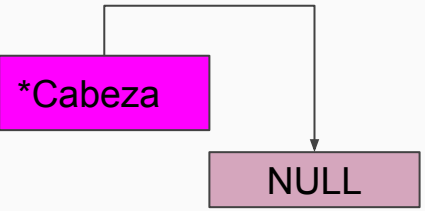
Pilas mediante listas enlazadas

Componentes

1. Lista simplemente enlazada

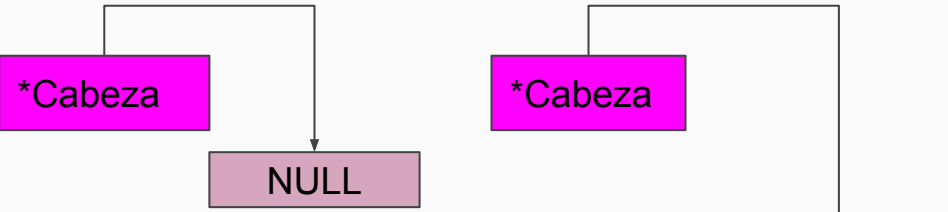


Ejemplo básico de funcionamiento `PUSH`

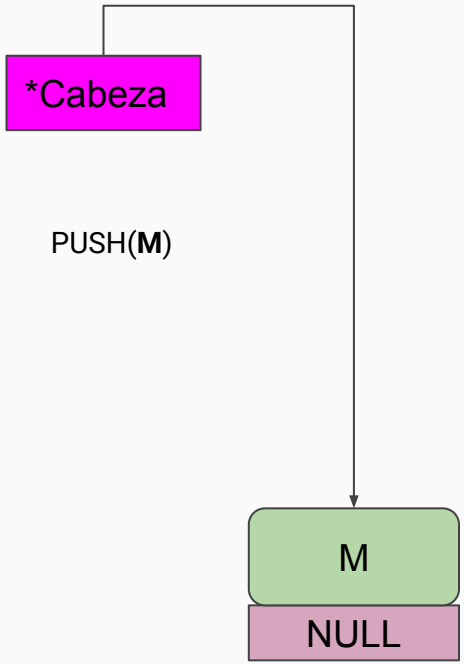


Entrada MAC

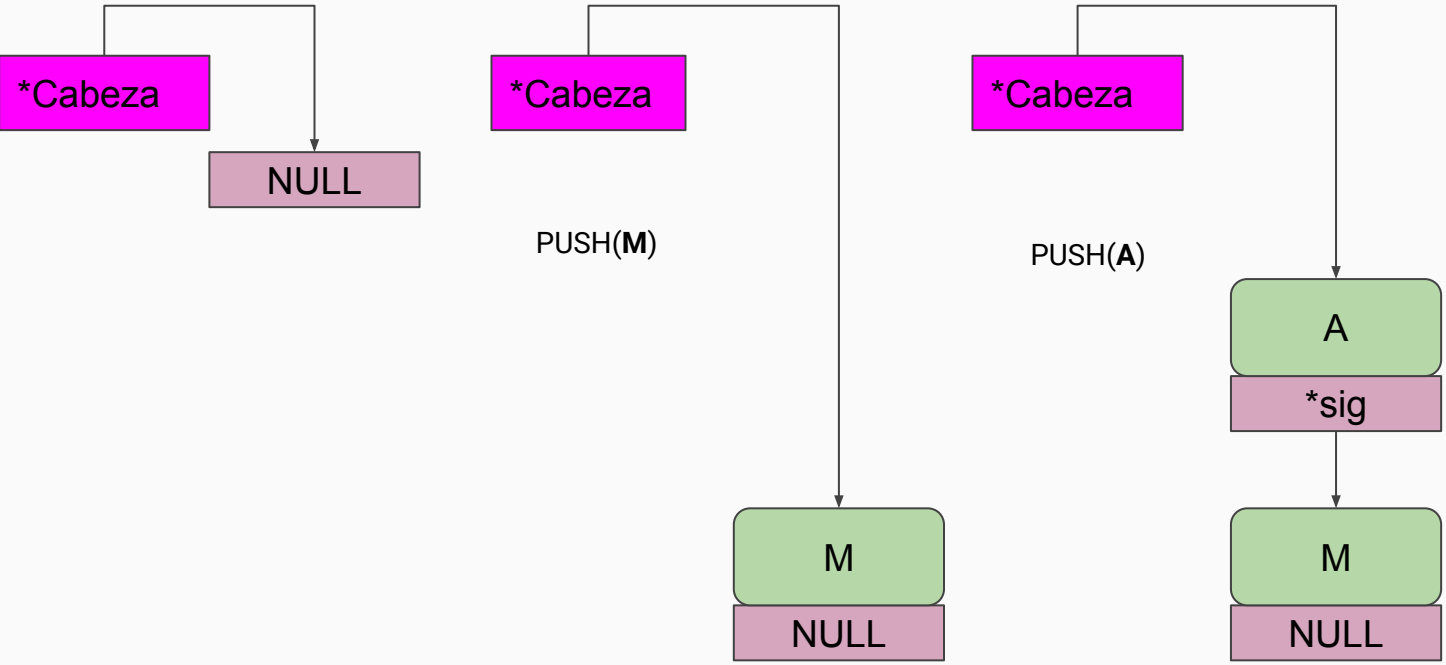
Ejemplo básico de funcionamiento `PUSH`



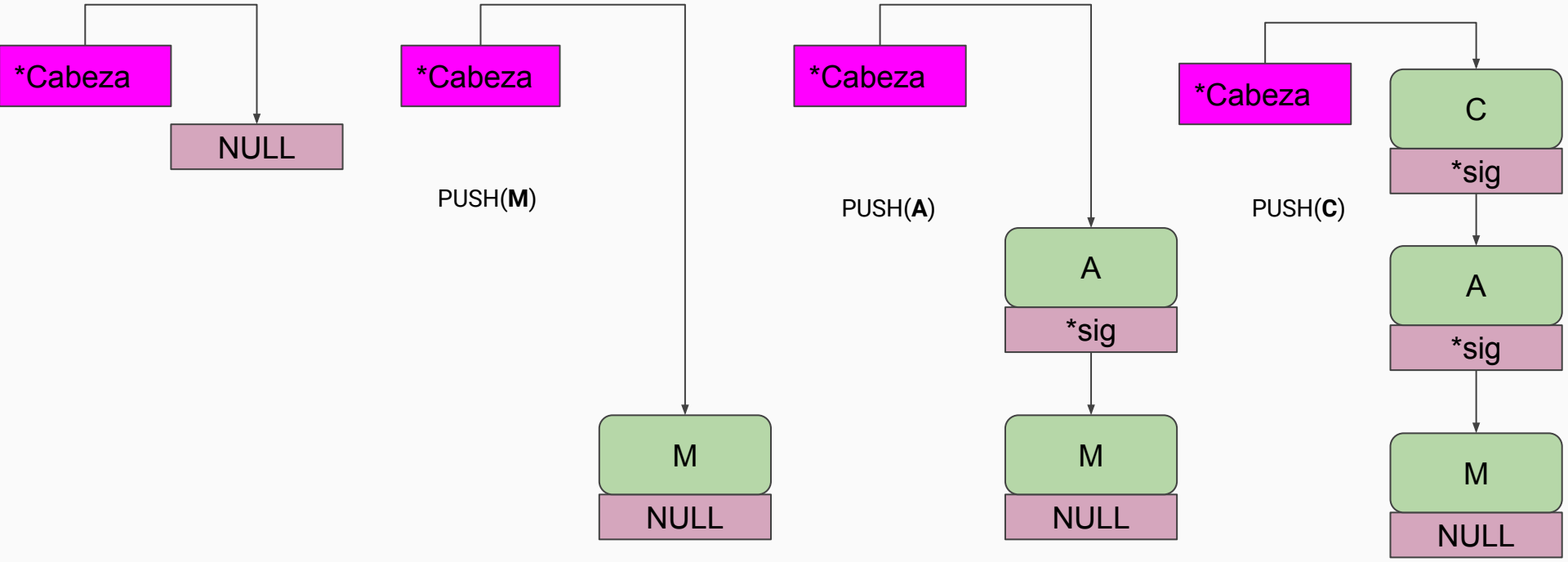
PUSH(**M**)



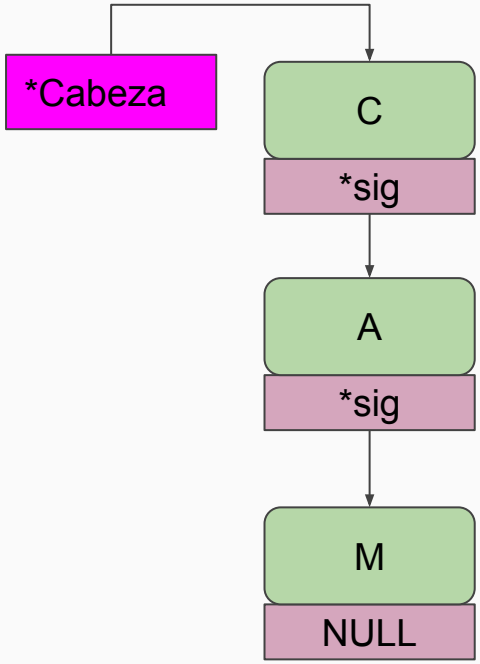
Ejemplo básico de funcionamiento `PUSH`



Ejemplo básico de funcionamiento `PUSH`

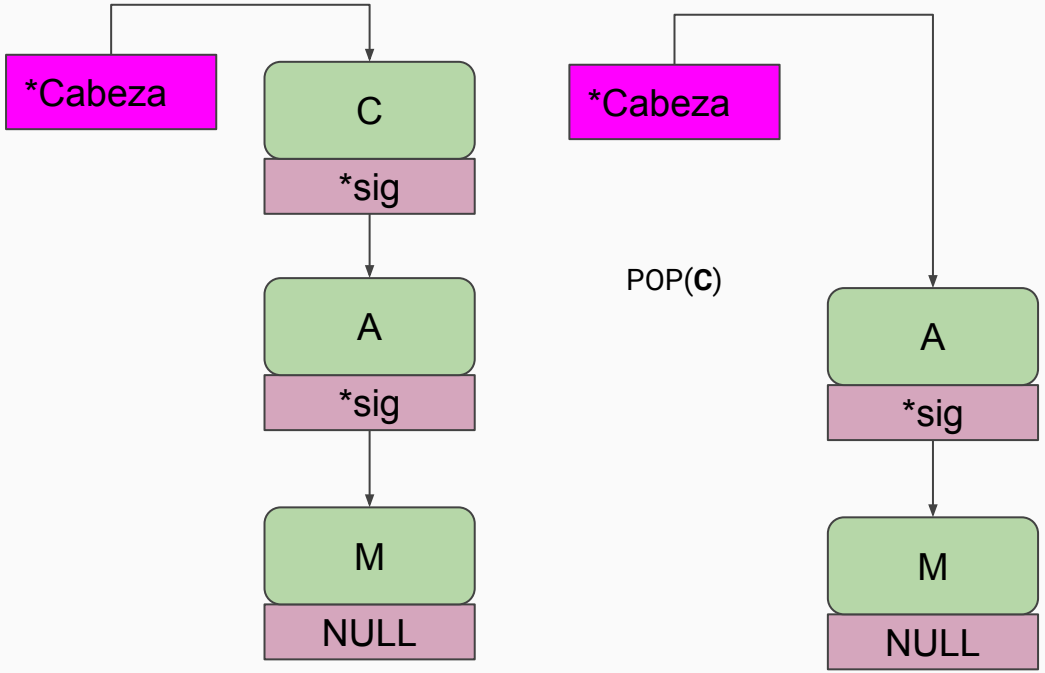


Ejemplo básico de funcionamiento `POP`



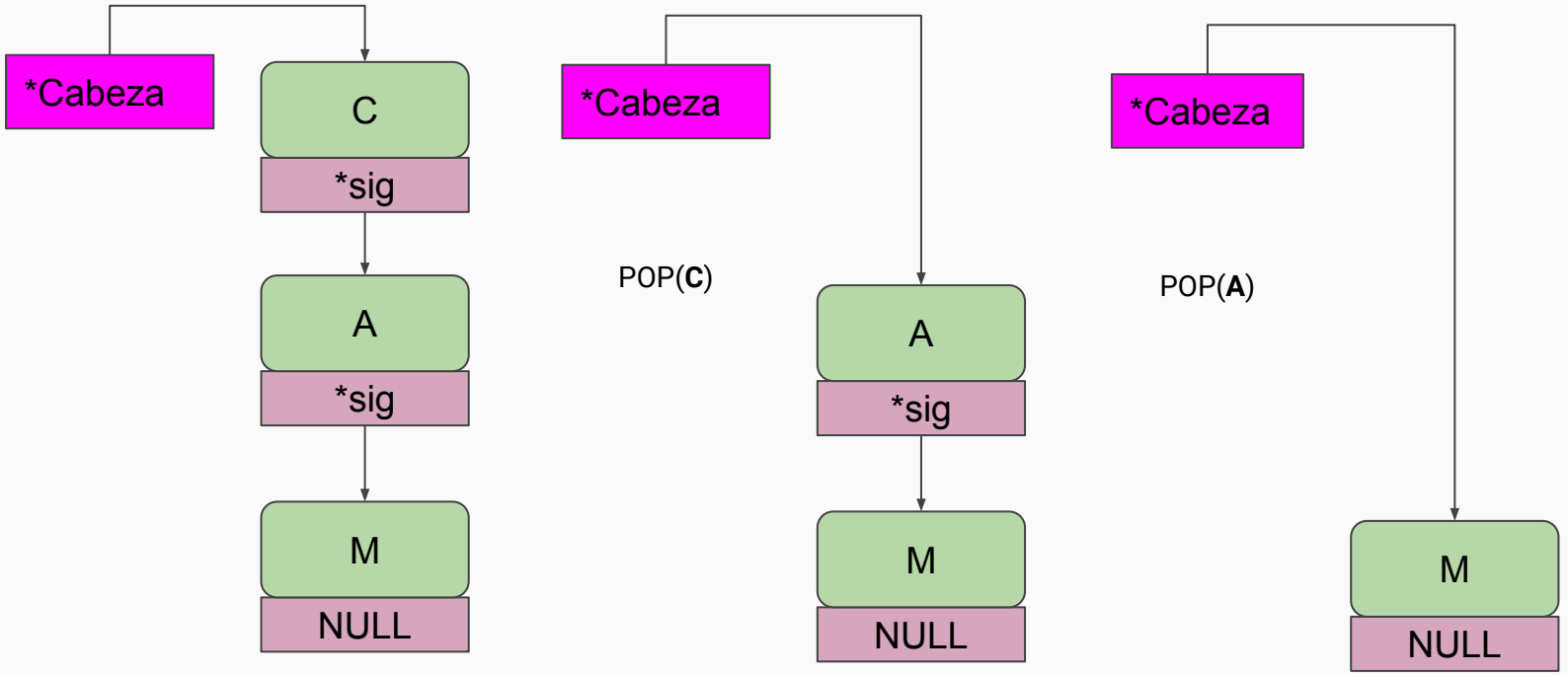
Salida CAM

Ejemplo básico de funcionamiento `POP`



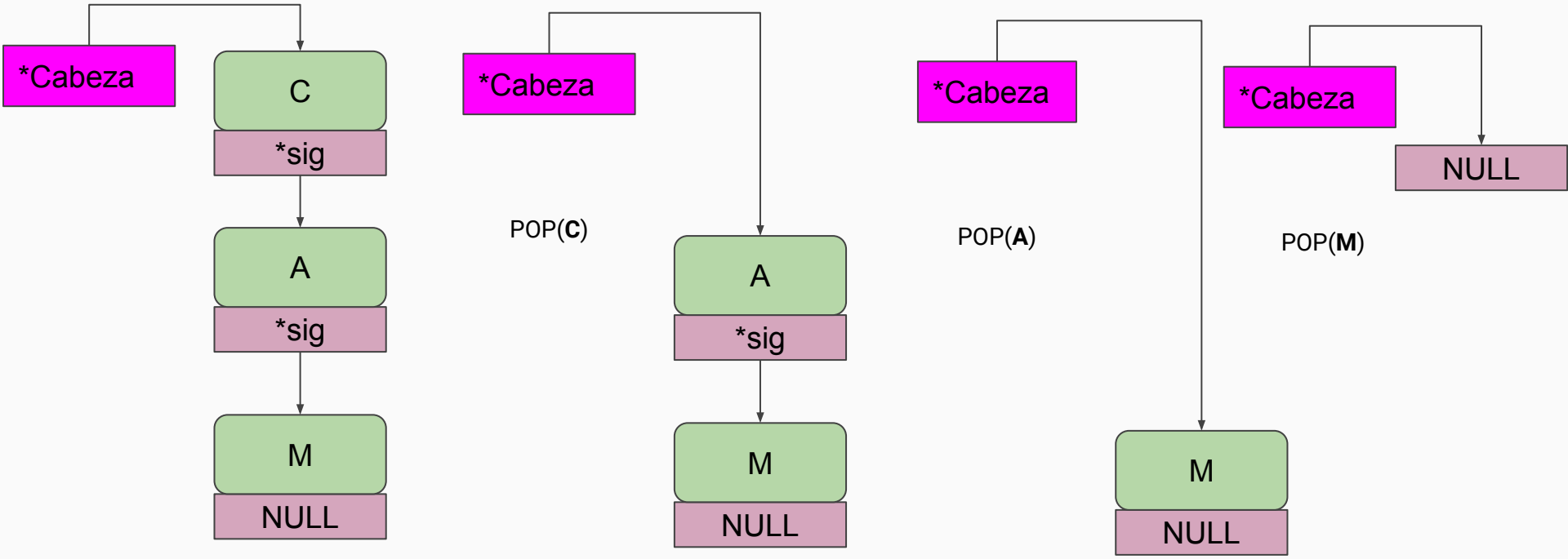
Salida CAM

Ejemplo básico de funcionamiento `POP`



Salida CAM

Ejemplo básico de funcionamiento `POP`



Salida CAM

Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

```
/*archivo pilaLista.h*/


struct nodo{
    TipoDato elemento;
    struct nodo * siguiente;
}
typedef struct nodo Nodo;

/*Operaciones sobre la Pila*/
void crearPila(Nodo ** pila);
void insertarPila(Nodo ** pila, TipoDato elemento);
TipoDato quitarPila(Nodo ** pila);
void limpiarPila(Nodo ** pila);

int cimaPila(Nodo * pila);
int pilaVacia(Nodo * pila);
```

Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

```
/*archivo pilaLista.c*/
```

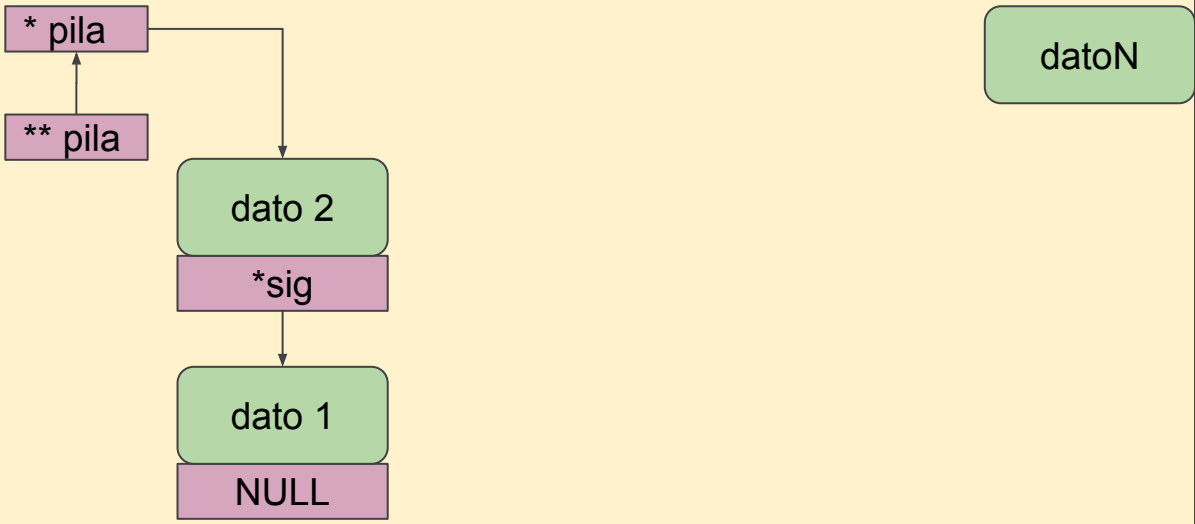
```
/*Crea una Pila nueva*/
void crearPila(Nodo ** pila){
     *pila = NULL;
}
```

```
/*Verifica si la pila esta vacia*/
int  pilaVacia(Nodo * pila){
     return *pila;
}
```

Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

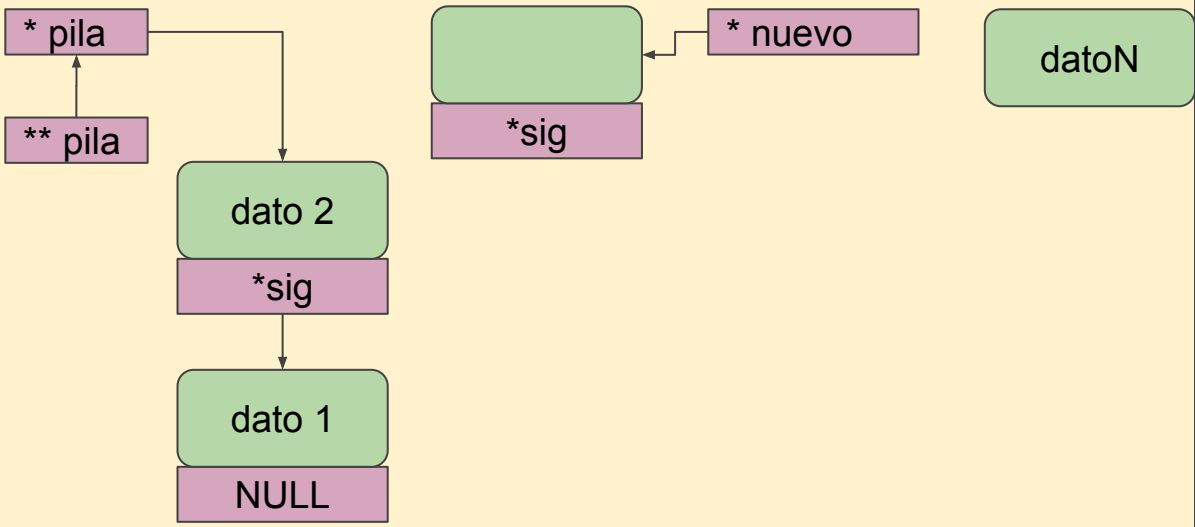
```
/*archivo pilaLista.c*/

/*Insertar elemento en una Pila*/
void insertarPila(Nodo ** pila, TipoDato datoN){
     Nodo * nuevo;

     nuevo = (Nodo *) malloc (sizeof(Nodo));
     nuevo -> elemento = datoN;
     nuevo -> siguiente = *pila;
     (*pila) = nuevo;
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

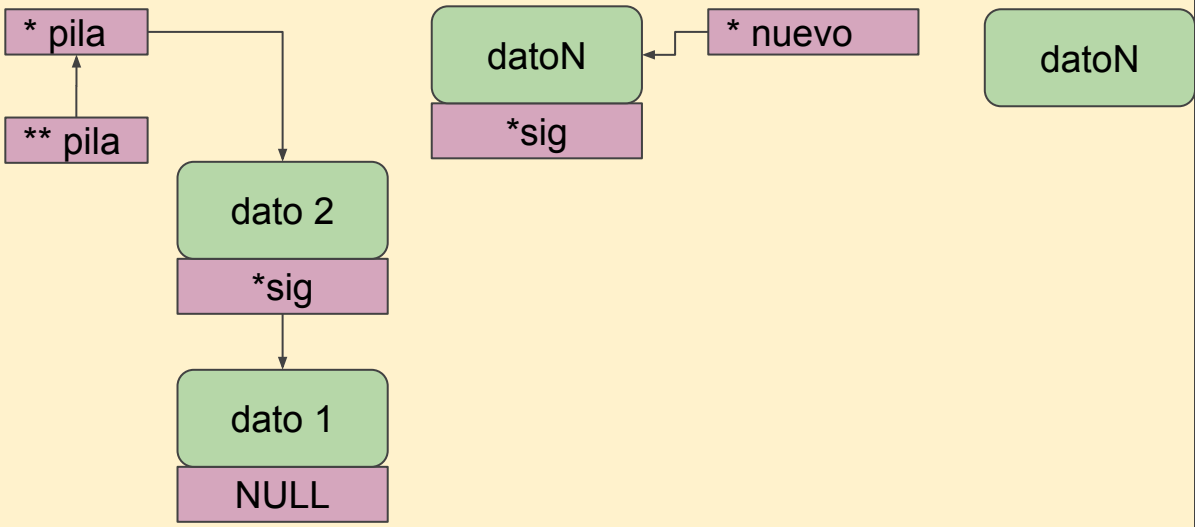
```
/*archivo pilaLista.c*/

/*Insertar elemento en una Pila*/
void insertarPila(Nodo ** pila, TipoDato dato){
     Nodo * nuevo;

     nuevo = (Nodo *) malloc (sizeof(Nodo));
     nuevo -> elemento = dato;
     nuevo -> siguiente = *pila;
     (*pila) = nuevo;
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

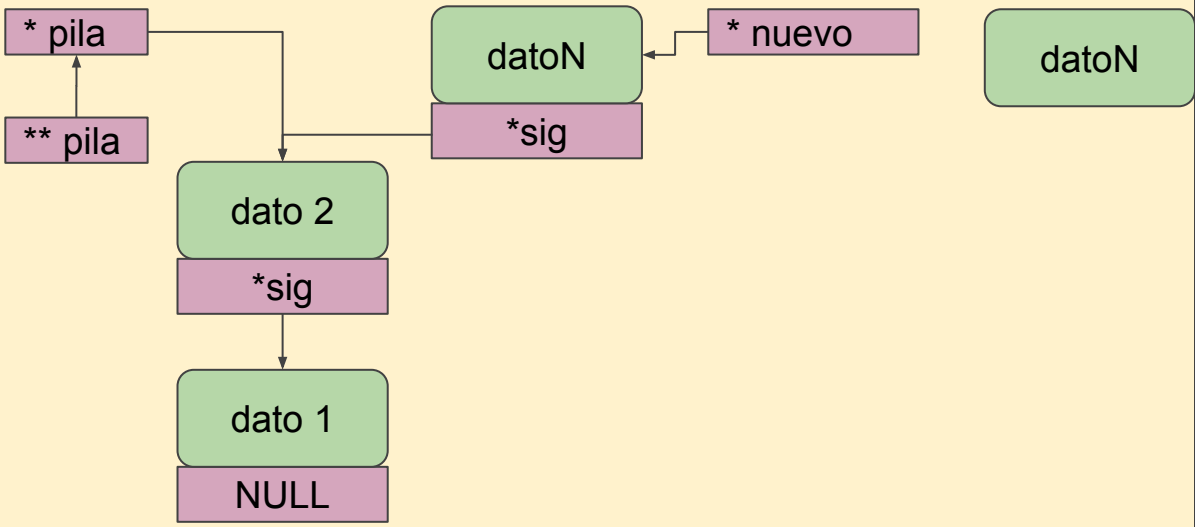
```
/*archivo pilaLista.c*/

/*Insertar elemento en una Pila*/
void insertarPila(Nodo ** pila, TipoDato dato){
     Nodo * nuevo;

     nuevo = (Nodo *) malloc (sizeof(Nodo));
     nuevo -> elemento = dato;
     nuevo -> siguiente = *pila;
     (*pila) = nuevo;
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

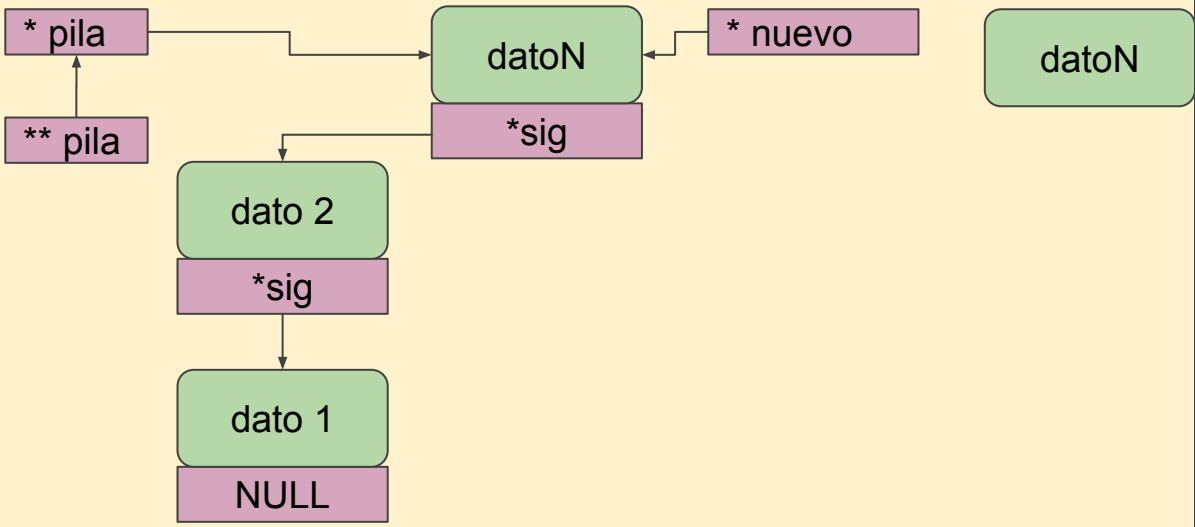
```
/*archivo pilaLista.c*/

/*Insertar elemento en una Pila*/
void insertarPila(Nodo ** pila, TipoDato dato){
     Nodo * nuevo;

     nuevo = (Nodo *) malloc (sizeof(Nodo));
     nuevo -> elemento = dato;
     nuevo -> siguiente = *pila;
     (*pila) = nuevo;
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

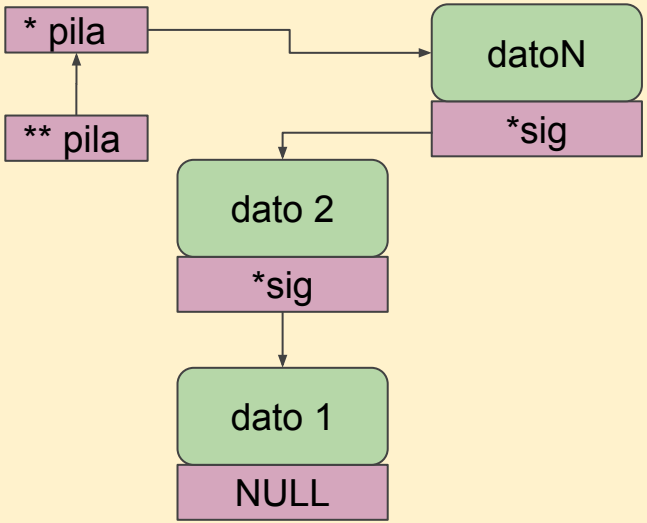
```
/*archivo pilaLista.c*/

/*Insertar elemento en una Pila*/
void insertarPila(Nodo ** pila, TipoDato dato){
     Nodo * nuevo;

     nuevo = (Nodo *) malloc (sizeof(Nodo));
     nuevo -> elemento = dato;
     nuevo -> siguiente = *pila;
     (*pila) = nuevo;
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

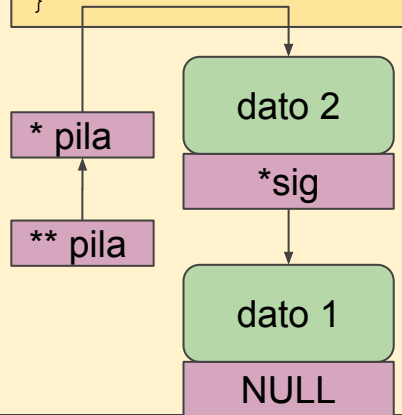
```
/*archivo pilaLista.c*/

/*Insertar elemento en una Pila*/
void insertarPila(Nodo ** pila, TipoDato dato){
     Nodo * nuevo;

     nuevo = (Nodo *) malloc (sizeof(Nodo));
     nuevo -> elemento = dato;
     nuevo -> siguiente = *pila;
     (*pila) = nuevo;
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

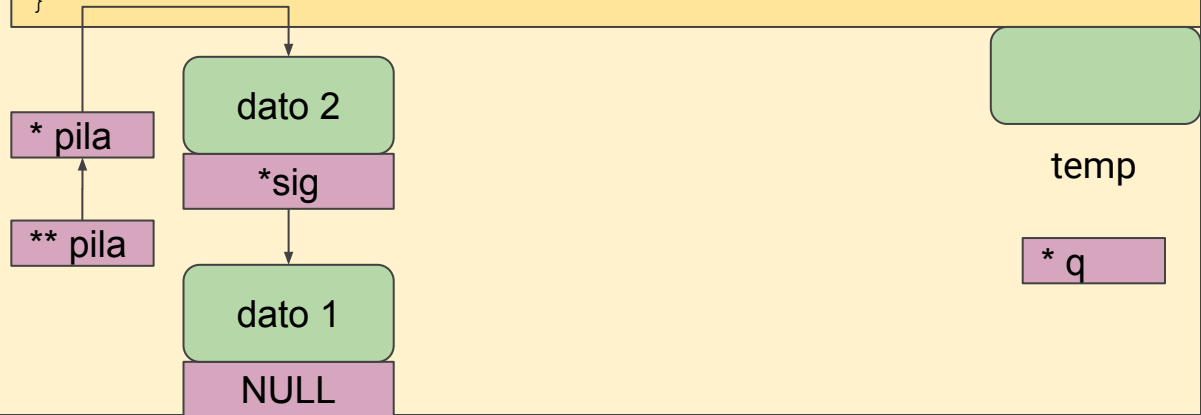
```
/*archivo pilaLista.c*/
/*Quitar elemento en una Pila*/
TipoDato quitarPila(Nodo ** pila){
     TipoDato tem;
     Nodo * q;

     if(pilaVacia(*pila)){
          puts("Pila Vacia\n");
          exit(1);
     }
     tem = (*pila) -> elemento;
     q = *pila:
     (*pila) = (*pila) -> siguiente;
     free(q);

     return tem;
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

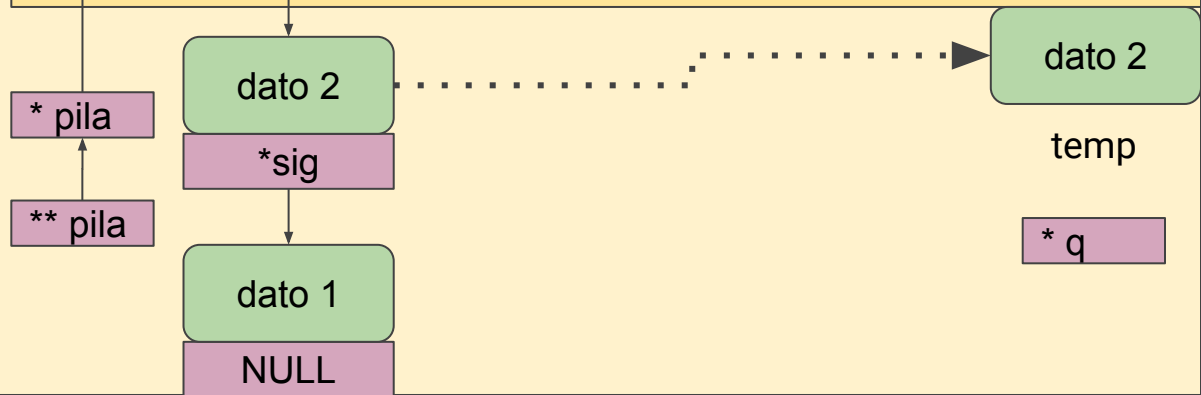
```
/*archivo pilaLista.c*/
/*Quitar elemento en una Pila*/
TipoDato quitarPila(Nodo ** pila){
      TipoDato tem;
      Nodo * q;

      if(pilaVacia(*pila)){
            puts("Pila Vacia\n");
            exit(1);
      }
      tem = (*pila) -> elemento;
      q = *pila:
      (*pila) = (*pila) -> siguiente;
      free(q);

      return tem;
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

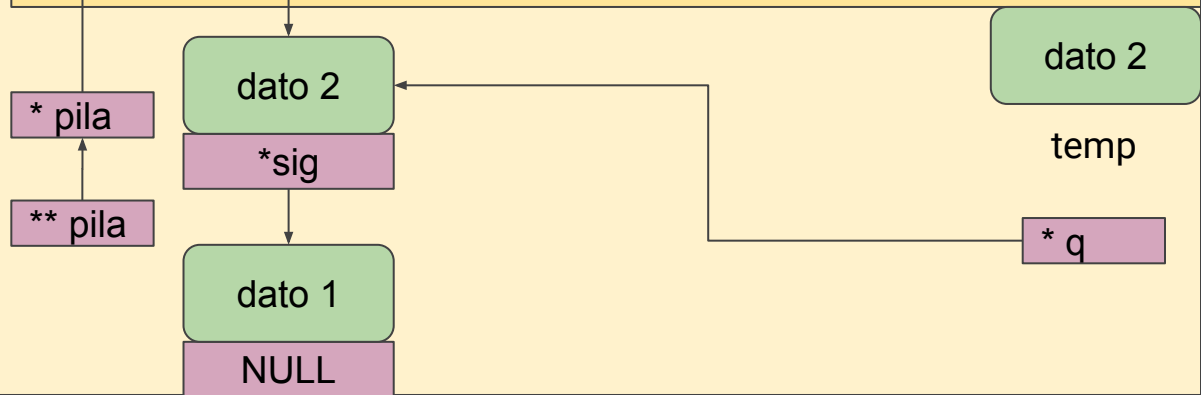
```
/*archivo pilaLista.c*/
/*Quitar elemento en una Pila*/
TipoDato quitarPila(Nodo ** pila){
      TipoDato tem;
      Nodo * q;

      if(pilaVacia(*pila)){
            puts("Pila Vacia\n");
            exit(1);
      }
      tem = (*pila) -> elemento;
      q = *pila;
      (*pila) = (*pila) -> siguiente;
      free(q);

      return tem;
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

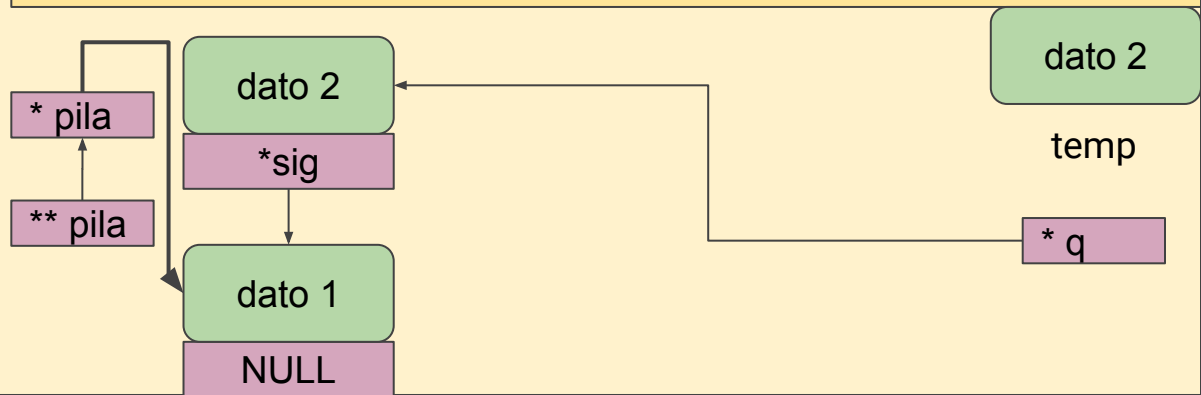
```
/*archivo pilaLista.c*/
/*Quitar elemento en una Pila*/
TipoDato quitarPila(Nodo ** pila){
      TipoDato tem;
      Nodo * q;

      if(pilaVacia(*pila)){
            puts("Pila Vacia\n");
            exit(1);
      }
      tem = (*pila) -> elemento;
      q = *pila;
      (*pila) = (*pila) -> siguiente;
      free(q);

      return tem;
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

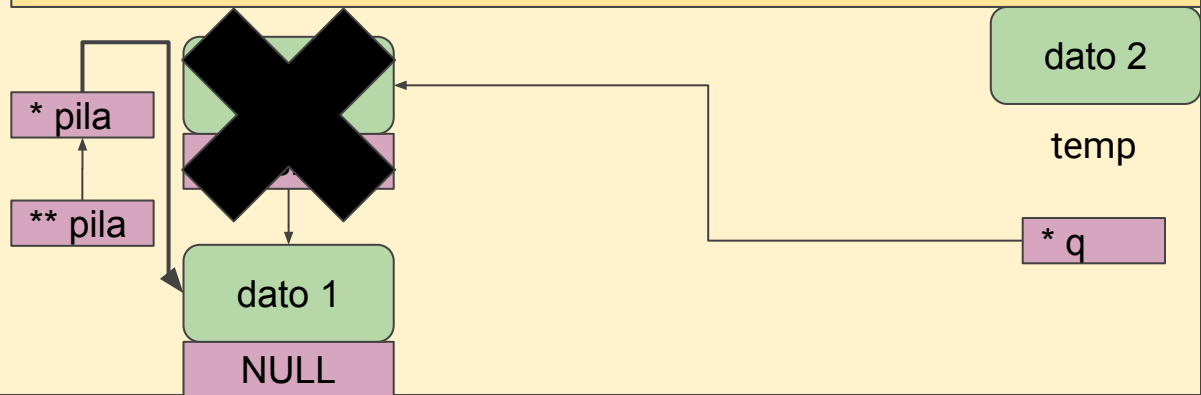
```
/*archivo pilaLista.c*/
/*Quitar elemento en una Pila*/
TipoDato quitarPila(Nodo ** pila){
     TipoDato tem;
     Nodo * q;

     if(pilaVacia(*pila)){
          puts("Pila Vacia\n");
          exit(1);
     }
     tem = (*pila) -> elemento;
     q = *pila;
     (*pila) = (*pila) -> siguiente;
     free(q);

     return tem;
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

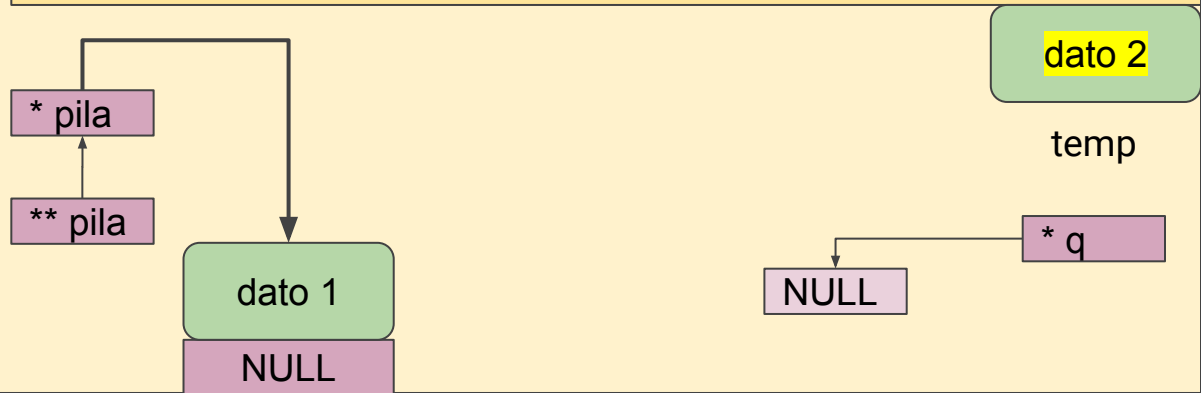
```
/*archivo pilaLista.c*/
/*Quitar elemento en una Pila*/
TipoDato quitarPila(Nodo ** pila){
     TipoDato tem;
     Nodo * q;

     if(pilaVacia(*pila)){
          puts("Pila Vacia\n");
          exit(1);
     }
     tem = (*pila) -> elemento;
     q = *pila;
     (*pila) = (*pila) -> siguiente;
     free(q);

     return tem;
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

```
/*archivo pilaLista.c*/
/*Quitar elemento en una Pila*/
TipoDato quitarPila(Nodo ** pila){
     TipoDato tem;
     Nodo * q;

     if(pilaVacia(*pila)){
          puts("Pila Vacia\n");
          exit(1);
     }
     tem = (*pila) -> elemento;
     q = *pila;
     (*pila) = (*pila) -> siguiente;
     free(q);

     return tem;
}
```



Funciones

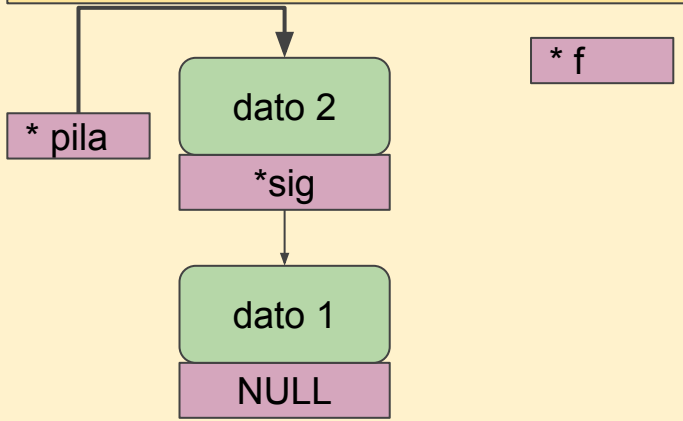
1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

```
/*archivo pilaLista.c*/
/*obtener elemento top de una Pila*/
TipoDato cima(Nodo * pila){
     if(pilaVacia(pila)){
          puts("Pila Vacia\n");
          exit(1);
     }

     return pila -> elemento;
}
```

Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

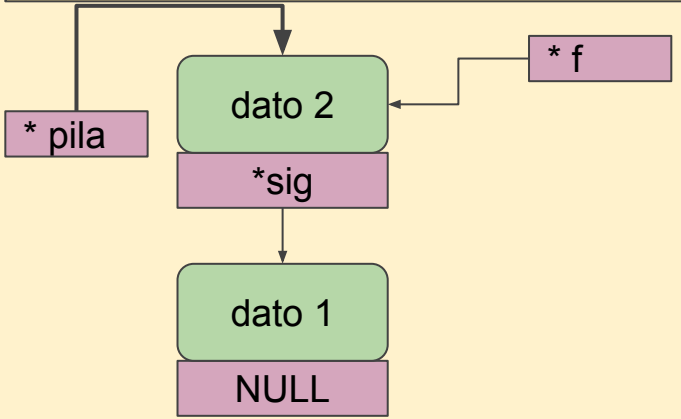
```
/*archivo pilaLista.c*/
/*Elimina elemento top de una Pila*/
void suprimirNodo(Nodo ** pila){
     if(!pilaVacia(*pila)){
          Nodo * f;
          f = *pila;
          (*pila) = (*pila) -> siguiente;
          free(f);
     }
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

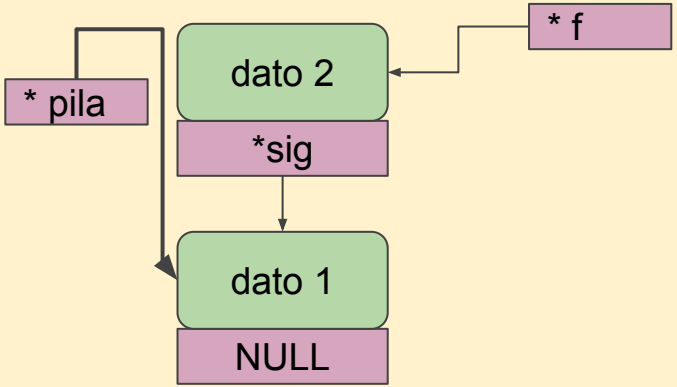
```
/*archivo pilaLista.c*/
/*Elimina elemento top de una Pila*/
void suprimirNodo(Nodo ** pila){
     if(!pilaVacia(*pila)){
          Nodo * f;
          f = *pila;
          (*pila) = (*pila) -> siguiente;
          free(f);
     }
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

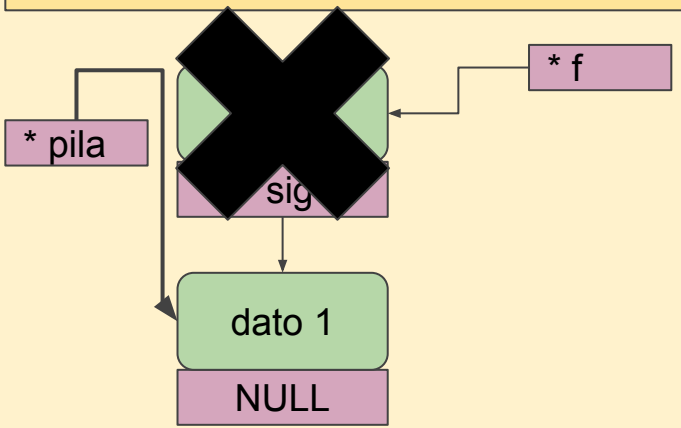
```
/*archivo pilaLista.c*/
/*Elimina elemento top de una Pila*/
void suprimirNodo(Nodo ** pila){
     if(!pilaVacia(*pila)){
          Nodo * f;
          f = *pila;
          (*pila) = (*pila) -> siguiente;
          free(f);
     }
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

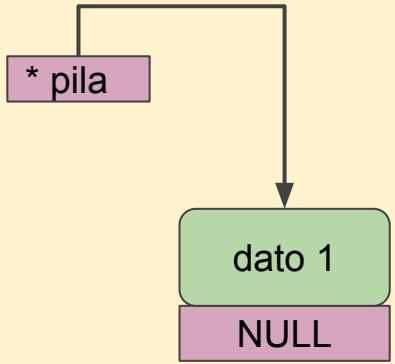
```
/*archivo pilaLista.c*/
/*Elimina elemento top de una Pila*/
void suprimirNodo(Nodo ** pila){
     if(!pilaVacia(*pila)){
          Nodo * f;
          f = *pila;
          (*pila) = (*pila) -> siguiente;
          free(f);
     }
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

```
/*archivo pilaLista.c*/
/*Elimina elemento top de una Pila*/
void suprimirNodo(Nodo ** pila){
     if(!pilaVacia(*pila)){
          Nodo * f;
          f = *pila;
          (*pila) = (*pila) -> siguiente;
          free(f);
     }
}
```



Funciones

1. `crearPila()` - Crear Pila
2. `pilaVacia()`- Verificar Pila Vacía
3. ~~`pilaLlena()` - Verificar Pila Llena~~
4. `insertarPila()`
5. `quitarPila()`
6. `cima()`- Devuelve el valor de cima sin quitar
7. `suprimirNodo()` - elimina elemento top
8. `limpiaPila()` - vacia la pila

```
/*archivo pilaLista.c*/
```

```
/*Elimina elemento top de una Pila*/
void suprimirNodo(Nodo ** pila){
     if(!pilaVacia(*pila)){
          Nodo * f;
          f = *pila;
          (*pila) = (*pila) -> siguiente;
          free(f);
     }
}
```

```
/*Elimina todos los elementos de la Pila*/
void limpiarPila(Nodo ** pila){
     while(!pilaVacia(*pila)){
          suprimirNodo(pila);
     }
}
```

La Pila es una estructura de datos LIFO

¡Aguije!