```c
/* ---------------------o---x---o---------------------
 *          pila.h     PILA DINÁMICA CIRCULAR
 * ---------------------o---x---o--------------------- */

#ifndef PILA_H_
#define PILA_H_

/// pila DINÁMICA CIRCULAR
#include <stdlib.h>
#include <string.h>

#define minimo( X , Y )      ( ( X ) <= ( Y ) ? ( X ) : ( Y ) )

typedef struct sNodo
{
    void           *info;
    unsigned        tamInfo;
    struct sNodo   *sig;
} tNodo;
typedef tNodo *tPila;

void crearPila(tPila *p);
int  pilaLlena(const tPila *p, unsigned cantBytes);
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes);
int  verTope(const tPila *p, void *d, unsigned cantBytes);
int  pilaVacia(const tPila *p);
int  sacarDePila(tPila *p, void *d, unsigned cantBytes);
void vaciarPila(tPila *p);

#endif


/** ************************************************************************** */
/* ---------------------o---x---o---------------------
 *          pila.c     PILA DINÁMICA CIRCULAR
 * ---------------------o---x---o--------------------- */

/// pila DINÁMICA CIRCULAR
#include "pila.h"
#include <stdio.h>

void crearPila(tPila *p)
{
    *p = NULL;
}

int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}

int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;
    unsigned nueMio;
    unsigned nueMioSig;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nueMio=(nue);///MIOOOOO
    //nueMioSig=nue->sig;
    if(*p == NULL)
    {
        nue->sig = nue;
        nueMioSig=nue->sig;
        *p = nue;
    }
    else
    {
        nue->sig = (*p)->sig;
        (*p)->sig = nue;
        nueMioSig=nue->sig;
    }
```

```
85      printf("t:%p, p-sig:%p \nnue:%x nue-sig:%x \n", *p, (*p)->sig,nueMio,nueMioSig);
86      return 1;
87  }
88
89  int  verTope(const tPila *p, void *d, unsigned cantBytes)
90  {
91      if(*p == NULL)
92          return 0;
93      memcpy(d, (*p)->sig->info, minimo(cantBytes, (*p)->sig->tamInfo));
94      return 1;
95  }
96
97  int  pilaVacia(const tPila *p)
98  {
99      return *p == NULL;
100 }
101
102 int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
103 {
104     tNodo *aux;
105
106     if(*p == NULL)
107         return 0;
108     aux = (*p)->sig;
109     memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
110     if(aux == *p)
111         *p = NULL;
112     else
113         (*p)->sig = aux->sig;
114
115     printf(" tope %p, p->sig: %p \n", *p, (*p)->sig);
116
117     free(aux->info);
118     free(aux);
119
120
121
122     return 1;
123 }
124
125 void vaciarPila(tPila *p)
126 {
127     while(*p)
128     {
129         tNodo *aux = (*p)->sig;
130
131         if(*p == aux)
132             *p = NULL;
133         else
134             (*p)->sig = aux->sig;
135         free(aux->info);
136         free(aux);
137     }
138 }
139
140
141
```