```c
/* ---------------------------------o---x---o---------------------------------
 *          cola.h      TDA COLA con asignación estática de memoria
 * ---------------------------------o---x---o---------------------------------*/

#ifndef COLA_H_
#define COLA_H_


#include <string.h>
#include <stdlib.h>


#define     TAM_COLA        300

typedef struct
{
    char        cola[TAM_COLA];
    unsigned    pri,
                ult,
                tamDisp;
} tCola;


void crearCola(tCola *p);
int  colaLlena(const tCola *p, unsigned cantBytes);
int  ponerEnCola(tCola *p, const void *d, unsigned cantBytes);
int  verPrimeroCola(const tCola *p, void *d, unsigned cantBytes);
int  colaVacia(const tCola *p);
int  sacarDeCola(tCola *p, void *d, unsigned cantBytes);
void vaciarCola(tCola *p);

#endif


/** ------------------------------------------------------------------------- **/
/* ---------------------------------o---x---o---------------------------------
 *          cola.h      TDA COLA con asignación estática de memoria
 * ---------------------------------o---x---o---------------------------------*/

#include "cola.h"

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


void crearCola(tCola *p)
{
    p->pri      = TAM_COLA - 70;
    p->ult      = TAM_COLA - 70;
    p->tamDisp  = TAM_COLA;
}

int  colaLlena(const tCola *p, unsigned cantBytes)
{
    return p->tamDisp < cantBytes + sizeof(unsigned);
}

int  ponerEnCola(tCola *p, const void *d, unsigned cantBytes)
{
    unsigned    ini,
                fin;

    if(p->tamDisp < sizeof(unsigned) + cantBytes)
        return 0;
    p->tamDisp -= sizeof(unsigned) + cantBytes;
    if((ini = minimo(sizeof(cantBytes), TAM_COLA - p->ult)) != 0)
        memcpy(p->cola + p->ult, &cantBytes, ini);
    if((fin = sizeof(cantBytes) - ini) != 0)
        memcpy(p->cola, ((char *)&cantBytes) + ini, fin);
    p->ult = fin ? fin : p->ult + ini;
    if((ini = minimo(cantBytes, TAM_COLA - p->ult)) != 0)
        memcpy(p->cola + p->ult, d, ini);
    if((fin = cantBytes - ini) != 0)
        memcpy(p->cola, ((char *)d) + ini, fin);
    p->ult = fin ? fin : p->ult + ini;
    return 1;
}

int  verPrimeroCola(const tCola *p, void *d, unsigned cantBytes)
{
    unsigned    tamInfo,
                ini,
                fin,
                pos = p->pri;

```

```c
    if(p->tamDisp == TAM_COLA)
        return 0;
    if((ini = minimo(sizeof(unsigned), TAM_COLA - pos)) !=0)
        memcpy(&tamInfo, p->cola + pos, ini);
    if((fin = sizeof(unsigned) - ini) != 0)
        memcpy(((char *)&tamInfo) + ini, p->cola, fin);
    pos = fin ? fin : pos + ini;
    tamInfo = minimo(tamInfo, cantBytes);
    if((ini = minimo(tamInfo, TAM_COLA - pos)) != 0)
        memcpy(d, p->cola + pos, ini);
    if((fin = tamInfo - ini) != 0)
        memcpy(((char *)d) + ini, p->cola, fin);
    return 1;
}

int  colaVacia(const tCola *p)
{
    return p->tamDisp == TAM_COLA;
}

int  sacarDeCola(tCola *p, void *d, unsigned cantBytes)
{
    unsigned    tamInfo,
                ini,
                fin;

    if(p->tamDisp == TAM_COLA)
        return 0;
    if((ini = minimo(sizeof(unsigned), TAM_COLA - p->pri)) != 0)
        memcpy(&tamInfo, p->cola + p->pri, ini);
    if((fin = sizeof(unsigned) - ini) != 0)
        memcpy(((char *)&tamInfo) + ini, p->cola, fin);
    p->pri = fin ? fin : p->pri + ini;
    p->tamDisp += sizeof(unsigned) + tamInfo;
    tamInfo = minimo(tamInfo, cantBytes);
    if((ini = minimo(tamInfo, TAM_COLA - p->pri)) != 0)
        memcpy(d, p->cola + p->pri, ini);
    if((fin = tamInfo - ini) != 0)
        memcpy(((char *)d) + ini, p->cola, fin);
    p->pri = fin ? fin : p->pri + ini;
    return 1;
}

void vaciarCola(tCola *p)
{
    p->ult = p->pri;
    p->tamDisp = TAM_COLA;
}


/** ************************************************************************ **/
/* ---------------------------------o---x---o--------------------------------
 *          main.h     prueba del TDA COLA con asignación estática de memoria
 * ---------------------------------o---x---o-------------------------------- */

#ifndef MAIN_H_
#define MAIN_H_

#include <stdio.h>


#include "../../productos/productos.h"
#include "../../lineasDeTexto/lineasTexto.h"
#include "../../colaEstatica/cola.h"


void probarIngresarYMostrarProd(void);

void probarIngresarYMostrarTexto(void);

void probarPonerYSacarDeCola(void);

#endif


/** ************************************************************************ **/
/* ---------------------------------o---x---o--------------------------------
 *          main.c     prueba del TDA COLA con asignación estática de memoria
 * ---------------------------------o---x---o-------------------------------- */

#include "main.h"


int main(void)
```

```
169 {
170     probarIngresarYMostrarProd();
171
172     probarIngresarYMostrarTexto();
173
174     probarPonerYSacarDeCola();
175
176     return 0;
177 }
178
179
180 void probarIngresarYMostrarProd(void)
181 {
182     tProd   prod;
183     int     cant = 0;
184
185     puts("Probando ingresar productos y mostrar productos.\n"
186          "======== ======== ========= = ======= =========");
187     if(ingresarProducto(&prod))
188         mostrarProducto(NULL);
189     do
190     {
191         mostrarProducto(&prod);
192         cant++;
193     } while(ingresarProducto(&prod));
194     fprintf(stdout, "Se mostraron %d productos.\n\n", cant);
195 }
196
197 void probarIngresarYMostrarTexto(void)
198 {
199     char    linea[90];
200     int     cant = 0;
201
202     puts("Probando ingresar lineas de texto mostrandolas.\n"
203          "======== ======== ====== == ===== ============");
204     while(ingresarTexto(linea, sizeof(linea)))
205     {
206         cant++;
207         printf("\"%s\"\n", linea);
208     }
209     fprintf(stdout, "Se mostraron %d lineas de texto.\n\n", cant);
210 }
211
212 int _probarLlenaYEncolar(tCola *cola)
213 {
214     tProd   prod;
215     int     cant = 0;
216
217     puts("Probando cola llena y poner en cola.");
218     mostrarProducto(NULL);
219     while(!colaLlena(cola, sizeof(prod)) && ingresarProducto(&prod))
220     {
221         if(!ponerEnCola(cola, &prod, sizeof(prod)))
222         {
223             fprintf(stderr, "ERROR - inesperado - cola llena.\n");
224             puts("no se pudo cargar la informacion y"
225                  " habria que tomar alguna decision drastica.");
226         }
227         mostrarProducto(&prod);
228         cant++;
229     }
230     return cant;
231 }
232
233
234 void _probarVerTope(tCola *cola)
235 {
236     tProd   prod;
237
238     puts("Probando ver el primero de la cola.");
239     if(verPrimeroCola(cola, &prod, sizeof(prod)))
240     {
241         mostrarProducto(NULL);
242         mostrarProducto(&prod);
243     }
244     else
245         puts("La cola estaba vacia.");
246     puts("");
247 }
248
249
250 void _probarVaciaYDesacolarN(tCola *cola, int canti)
251 {
252     tProd   prod;
```

```
253
254         printf("Probando cola vacia y sacar de cola %d productos (mostrandolos.\n",
255             canti);
256         if(colaVacia(cola))
257             puts("La cola esta vacia.");
258         else
259             mostrarProducto(NULL);
260         while(canti > 0 && sacarDeCola(cola, &prod, sizeof(prod)))
261         {
262             canti--;
263             mostrarProducto(&prod);
264         }
265         puts("");
266     }
267
268
269     int _probarVaciarColaYColaVacia(tCola *cola)
270     {
271         puts("Probando vaciar cola y cola vacia.");
272         vaciarCola(cola);
273         if(!colaVacia(cola))
274             return 0;  // fprintf(stderr, "ERROR - la cola debia estar vacia\n\n");
275         printf("Vaciar cola funciona!\n\n");
276         puts("");
277         return 1;
278     }
279
280
281     void _probarLlenaYEncolarTexto(tCola *cola)
282     {
283         char    linea[70];
284         int     cant = 0;
285
286         puts("Probando cola llena y poner en cola Texto.");
287         while(!colaLlena(cola, sizeof(linea)) &&
288             ingresarTexto(linea, sizeof(linea)))
289         {
290             if(!ponerEnCola(cola, linea, strlen(linea) + 1))
291             {
292                 fprintf(stderr, "ERROR - inesperado - cola llena.\n");
293                 puts("no se pudo cargar la informacion y"
294                     " habria que tomar alguna decision drastica.");
295             }
296             printf("\"%s\"\n", linea);
297             cant++;
298         }
299         printf("se pusieron %d lineas de texto en la cola.\n\n", cant);
300         printf("Probando sacar de cola con las lineas de texto.\n");
301     }
302
303
304     void _probarSacarDeColaTexto(tCola *cola)
305     {
306         char    linea[70];
307         int     cant = 0;
308
309         while(sacarDeCola(cola, linea, sizeof(linea)))
310         {
311             cant++;
312             printf("\"%s\"\n", linea);
313         }
314         printf("Se sacaron y mostraron %d lineas de texto\n\n", cant);
315     }
316
317
318     void probarPonerYSacarDeCola(void)
319     {
320         tCola   cola;
321         int     cant;
322
323         crearCola(&cola);
324
325         puts("Probando primitivas de cola con productos.\n"
326             "======== ========== == ==== === =========\n");
327         cant = _probarLlenaYEncolar(&cola);
328         printf("se pusieron %d productos en la cola.\n\n", cant);
329
330         _probarVerTope(&cola);
331
332         _probarVaciaYDesacolarN(&cola, cant - 2);
333
334         _probarVerTope(&cola);
335
336         if(_probarVaciarColaYColaVacia(&cola) != 1)
```

```
337             fprintf(stderr, "ERROR - inesperado, la cola NO QUEDO vacia\n");
338
339         puts("Probando primitivas de cola con lineas de texto.\n"
340             "======== ========== == ==== === ====== == =====\n");
341
342         _probarLlenaYEncolarTexto(&cola);
343
344         _probarSacarDeColaTexto(&cola);
345         cant = 0;
346
347         puts("ATENCION: se mostro el uso de una cola, en la que se pusieron"
348             " productos, se pro\n"
349             "baron todas las primitivas.  Una vez que se la dejo vacia "
350             "se encolaron lineas\n"
351             "de texto y luego se desencolaron.  Lo mas importante de esto "
352             "no es utilizar la\n"
353             "misma cola, lo que mas importa es que con las mismas primiti"
354             "vas se pueden enco-\n"
355             "lar distintos tipos de objetos, incluyendo lineas de texto d"
356             "e distinto tamano.\n");
357     }
358
359
360     /** ------------------------------------------------------------------------ **/
361
```