```c
/* --------------------------------o---x---o--------------------------------
 *          main.c     prueba del TDA PILA con asignación dinámica de memoria
 * --------------------------------o---x---o-------------------------------- */

#include "main.h"


int main(void)
{
    probarIngresarYMostrarProd();

    probarIngresarYMostrarTexto();

    probarPonerYSacarDePila();

    return 0;
}


void probarIngresarYMostrarProd(void)
{
    tProd   prod;
    int     cant = 0;

    puts("Probando ingresar productos y mostrar productos.\n"
         "======== ======== ========= = ======= =========");
    if(ingresarProducto(&prod))
        mostrarProducto(NULL);
    do
    {
        mostrarProducto(&prod);
        cant++;
    } while(ingresarProducto(&prod));
    fprintf(stdout, "Se mostraron %d productos.\n\n", cant);
}

void probarIngresarYMostrarTexto(void)
{
    char    linea[90];
    int     cant = 0;

    puts("Probando ingresar lineas de texto mostrandolas.\n"
         "======== ======== ====== == ===== ============");
    while(ingresarTexto(linea, sizeof(linea)))
    {
        cant++;
        printf("\"%s\"\n", linea);
    }
    fprintf(stdout, "Se mostraron %d lineas de texto.\n\n", cant);
}

/**
 ** DE NINGUNA MANERA ES ADMISIBLE HACER UNA FUNCION "MONOLITICA" TAN LARGA.
 ** USTED DEBERIA DIVIDIRLA EN VARIAS FUNCIONES.
 ** EN CADA FUNCION PROBAR UNA O DOS PRIMITIVAS.
 ** A ESAS FUNCIONES DEBERIA PASARLES LA PILA (por puntero).
 ** (VER AL FINAL)
 **/
void probarPonerYSacarDePila(void)
{
    tProd   prod,
            otro;
    char    linea[70];
    tPila   pila;
    int     cant;

    crearPila(&pila);

    puts("Probando primitivas de pila con productos.\n"
         "======== ========== == ==== === =========\n"
         "Probando pila llena y poner en pila.");
    mostrarProducto(NULL);
    cant = 0;
    while(!pilaLlena(&pila, sizeof(prod)) && ingresarProducto(&prod))
    {
        if(!ponerEnPila(&pila, &prod, sizeof(prod)))
        {
            fprintf(stderr, "ERROR - inesperado - pila llena.\n");
            puts("no se pudo cargar la informacion y"
                 " habria que tomar alguna decision drastica.");
        }
        mostrarProducto(&prod);
        cant++;
    }
```

```
85      printf("se pusieron %d productos en la pila.\n\n", cant);
86      puts("Probando ver el tope de la pila.");
87      if(verTope(&pila, &otro, sizeof(otro)))
88      {
89          mostrarProducto(NULL);
90          mostrarProducto(&otro);
91      }
92      else
93          puts("La pila estaba vacia.");
94      puts("");
95      cant -= 2;
96      printf("Probando pila vacia y sacar de pila %d productos (mostrandolos.\n",
97             cant);
98      if(pilaVacia(&pila))
99          puts("La pila esta vacia.");
100     else
101         mostrarProducto(NULL);
102     while(cant > 0 && sacarDePila(&pila, &prod, sizeof(prod)))
103     {
104         cant--;
105         mostrarProducto(&prod);
106     }
107     puts("");
108     puts("Probando ver el tope de la pila.");
109     if(verTope(&pila, &otro, sizeof(otro)))
110     {
111         puts("La pila no quedo vacia - en el tope hay ...");
112         mostrarProducto(NULL);
113         mostrarProducto(&otro);
114     }
115     else
116         puts("La pila esta vacia.");
117     puts("");
118     puts("Probando vaciar pila y pila vacia.");
119     vaciarPila(&pila);
120     if(!pilaVacia(&pila))
121         fprintf(stderr, "ERROR - la pila debia estar vacia\n\n");
122     else
123         printf("Vaciar pila funciona!\n\n");
124     puts("");
125
126     puts("Probando primitivas de pila con lineas de texto.\n"
127          "======== ========== == ==== === ====== == =====\n"
128          "Probando pila llena y poner en pila.");
129     cant = 0;
130     while(!pilaLlena(&pila, sizeof(prod)) && ingresarTexto(linea, sizeof(linea)))
131     {
132         if(!ponerEnPila(&pila, linea, strlen(linea) + 1))
133         {
134             fprintf(stderr, "ERROR - inesperado - pila llena.\n");
135             puts("no se pudo cargar la informacion y"
136                  " habria que tomar alguna decision drastica.");
137         }
138         printf("\"%s\"\n", linea);
139         cant++;
140     }
141     printf("se pusieron %d lineas de texto en la pila.\n\n", cant);
142     printf("Probando sacar de pila con las lineas de texto.\n");
143     cant = 0;
144     while(sacarDePila(&pila, linea, sizeof(linea)))
145     {
146         cant++;
147         printf("\"%s\"\n", linea);
148     }
149     printf("Se sacaron y mostraron %d lineas de texto\n\n", cant);
150     puts("ATENCION: se mostro el uso de una pila, en la que se apilaron"
151          " productos, se pro\n"
152          "baron todas las primitivas.  Una vez que se la dejo vacia "
153          "se apilaron lineas\n"
154          "de texto y luego se desapilaron.  Lo mas importante de esto "
155          "no es utilizar la\n"
156          "misma pila, lo que mas importa es que con las mismas primiti"
157          "vas se pueden api-\n"
158          "lar distintos tipos de objetos, incluyendo lineas de texto d"
159          "e distinto tamano.\n");
160 }
161
162
163 /**
164 int _probarLlenaYApilar(tPila *pila)
165 {
166     tProd   prod;
167     int     cant = 0;
168
```

```
169     puts("Probando pila llena y poner en pila.");
170     mostrarProducto(NULL);
171     while(!pilaLlena(pila, sizeof(prod)) && ingresarProducto(&prod))
172     {
173         if(!ponerEnPila(pila, &prod, sizeof(prod)))
174         {
175             fprintf(stderr, "ERROR - inesperado - pila llena.\n");
176             puts("no se pudo cargar la informacion y"
177                  " habria que tomar alguna decision drastica.");
178         }
179         mostrarProducto(&prod);
180         cant++;
181     }
182     return cant;
183 }
184
185 void _probarVerTope(tPila *pila)
186 {
187     tProd   prod;
188
189     puts("Probando ver el tope de la pila.");
190     if(verTope(pila, &prod, sizeof(prod)))
191     {
192         mostrarProducto(NULL);
193         mostrarProducto(&prod);
194     }
195     else
196         puts("La pila estaba vacia.");
197     puts("");
198 }
199
200 void _probarVaciaYDesapilarN(tPila *pila, int canti)
201 {
202     tProd   prod;
203
204     printf("Probando pila vacia y sacar de pila %d productos (mostrandolos.\n",
205            canti);
206     if(pilaVacia(pila))
207         puts("La pila esta vacia.");
208     else
209         mostrarProducto(NULL);
210     while(canti > 0 && sacarDePila(pila, &prod, sizeof(prod)))
211     {
212         canti--;
213         mostrarProducto(&prod);
214     }
215     puts("");
216 }
217
218 void probarPonerYSacarDePila_2(void)
219 {
220     tPila   pila;
221     int     cant;
222
223     crearPila(&pila);
224
225     puts("Probando primitivas de pila con productos.\n"
226          "======== ========== == ==== === =========\n");
227     cant = _probarLlenaYApilar(&pila);
228     printf("se pusieron %d productos en la pila.\n\n", cant);
229
230     _probarVerTope(&pila);
231
232     _probarVaciaYDesapilarN(&pila, cant - 2);
233
234 TERMINE DE DESARROLLAR LAS RESTANTES
235     puts("Probando ver el tope de la pila.");
236     if(verTope(&pila, &otro, sizeof(otro)))
237     {
238         puts("La pila no quedo vacia - en el tope hay ...");
239         mostrarProducto(NULL);
240         mostrarProducto(&otro);
241     }
242     else
243         puts("La pila esta vacia.");
244     puts("");
245     puts("Probando vaciar pila y pila vacia.");
246     vaciarPila(&pila);
247     if(!pilaVacia(&pila))
248         fprintf(stderr, "ERROR - la pila debia estar vacia\n\n");
249     else
250         printf("Vaciar pila funciona!\n\n");
251     puts("");
252
```

```
    puts("Probando primitivas de pila con lineas de texto.\n"
         "======== ========== == ==== === ====== == =====\n"
         "Probando pila llena y poner en pila.");
    cant = 0;
    while(!pilaLlena(&pila, sizeof(prod)) && ingresarTexto(linea, sizeof(linea)))
    {
        if(!ponerEnPila(&pila, linea, strlen(linea) + 1))
        {
            fprintf(stderr, "ERROR - inesperado - pila llena.\n");
            puts("no se pudo cargar la informacion y"
                 " habria que tomar alguna decision drastica.");
        }
        printf("\"%s\"\n", linea);
        cant++;
    }
    printf("se pusieron %d lineas de texto en la pila.\n\n", cant);
    printf("Probando sacar de pila con las lineas de texto.\n");
    cant = 0;
    while(sacarDePila(&pila, linea, sizeof(linea)))
    {
        cant++;
        printf("\"%s\"\n", linea);
    }
    printf("Se sacaron y mostraron %d lineas de texto\n\n", cant);
    puts("ATENCION: se mostro el uso de una pila, en la que se apilaron"
         " productos, se pro\n"
         "baron todas las primitivas.  Una vez que se la dejo vacia "
         "se apilaron lineas\n"
         "de texto y luego se desapilaron.  Lo mas importante de esto "
         "no es utilizar la\n"
         "misma pila, lo que mas importa es que con las mismas primiti"
         "vas se pueden api-\n"
         "lar distintos tipos de objetos, incluyendo lineas de texto d"
         "e distinto tamano.\n");
}
 **/
/* ---------------------------------o---x---o---------------------------------
 *          main.h     prueba del TDA PILA con asignación dinámica de memoria
 * ---------------------------------o---x---o--------------------------------- */

#ifndef MAIN_H_
#define MAIN_H_

#include <stdio.h>


#include "../productos/productos.h"
#include "../lineasDeTexto/lineasTexto.h"
#include "../pilaDinamica/pila.h"


void probarIngresarYMostrarProd(void);

void probarIngresarYMostrarTexto(void);

void probarPonerYSacarDePila(void);


#endif // MAIN_H_

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

#include "main.h"


int main(void)
{
    probarIngresarYMostrarProd();

    probarIngresarYMostrarTexto();

    probarPonerYSacarDePila();

    return 0;
}


void probarIngresarYMostrarProd(void)
{
    tProd   prod;
    int     cant = 0;

```

```c
    puts("Probando ingresar productos y mostrar productos.\n"
         "======== ======== ========= = ======= =========");
    if(ingresarProducto(&prod))
        mostrarProducto(NULL);
    do
    {
        mostrarProducto(&prod);
        cant++;
    } while(ingresarProducto(&prod));
    fprintf(stdout, "Se mostraron %d productos.\n\n", cant);
}

void probarIngresarYMostrarTexto(void)
{
    char    linea[90];
    int     cant = 0;

    puts("Probando ingresar lineas de texto mostrandolas.\n"
         "======== ======== ====== == ===== ============");
    while(ingresarTexto(linea, sizeof(linea)))
    {
        cant++;
        printf("\"%s\"\n", linea);
    }
    fprintf(stdout, "Se mostraron %d lineas de texto.\n\n", cant);
}

/**
 ** DE NINGUNA MANERA ES ADMISIBLE HACER UNA FUNCION "MONOLITICA" TAN LARGA.
 ** USTED DEBERIA DIVIDIRLA EN VARIAS FUNCIONES.
 ** EN CADA FUNCION PROBAR UNA O DOS PRIMITIVAS.
 ** A ESAS FUNCIONES DEBERIA PASARLES LA PILA (por puntero).
 ** (VER AL FINAL)
 **/
void probarPonerYSacarDePila(void)
{
    tProd   prod,
            otro;
    char    linea[70];
    tPila   pila;
    int     cant;

    crearPila(&pila);

    puts("Probando primitivas de pila con productos.\n"
         "======== ========== == ==== === =========\n"
         "Probando pila llena y poner en pila.");
    mostrarProducto(NULL);
    cant = 0;
    while(!pilaLlena(&pila, sizeof(prod)) && ingresarProducto(&prod))
    {
        if(!ponerEnPila(&pila, &prod, sizeof(prod)))
        {
            fprintf(stderr, "ERROR - inesperado - pila llena.\n");
            puts("no se pudo cargar la informacion y"
                 " habria que tomar alguna decision drastica.");
        }
        mostrarProducto(&prod);
        cant++;
    }
    printf("se pusieron %d productos en la pila.\n\n", cant);
    puts("Probando ver el tope de la pila.");
    if(verTope(&pila, &otro, sizeof(otro)))
    {
        mostrarProducto(NULL);
        mostrarProducto(&otro);
    }
    else
        puts("La pila estaba vacia.");
    puts("");
    cant -= 2;
    printf("Probando pila vacia y sacar de pila %d productos (mostrandolos.\n",
           cant);
    if(pilaVacia(&pila))
        puts("La pila esta vacia.");
    else
        mostrarProducto(NULL);
    while(cant > 0 && sacarDePila(&pila, &prod, sizeof(prod)))
    {
        cant--;
        mostrarProducto(&prod);
    }
    puts("");
    puts("Probando ver el tope de la pila.");
```

```
421         if(verTope(&pila, &otro, sizeof(otro)))
422         {
423             puts("La pila no quedo vacia - en el tope hay ...");
424             mostrarProducto(NULL);
425             mostrarProducto(&otro);
426         }
427         else
428             puts("La pila esta vacia.");
429         puts("");
430         puts("Probando vaciar pila y pila vacia.");
431         vaciarPila(&pila);
432         if(!pilaVacia(&pila))
433             fprintf(stderr, "ERROR - la pila debia estar vacia\n\n");
434         else
435             printf("Vaciar pila funciona!\n\n");
436         puts("");
437 
438         puts("Probando primitivas de pila con lineas de texto.\n"
439              "======== ========== == ==== === ====== == =====\n"
440              "Probando pila llena y poner en pila.");
441         cant = 0;
442         while(!pilaLlena(&pila, sizeof(prod)) && ingresarTexto(linea, sizeof(linea)))
443         {
444             if(!ponerEnPila(&pila, linea, strlen(linea) + 1))
445             {
446                 fprintf(stderr, "ERROR - inesperado - pila llena.\n");
447                 puts("no se pudo cargar la informacion y"
448                      " habria que tomar alguna decision drastica.");
449             }
450             printf("\"%s\"\n", linea);
451             cant++;
452         }
453         printf("se pusieron %d lineas de texto en la pila.\n\n", cant);
454         printf("Probando sacar de pila con las lineas de texto.\n");
455         cant = 0;
456         while(sacarDePila(&pila, linea, sizeof(linea)))
457         {
458             cant++;
459             printf("\"%s\"\n", linea);
460         }
461         printf("Se sacaron y mostraron %d lineas de texto\n\n", cant);
462         puts("ATENCION: se mostro el uso de una pila, en la que se apilaron"
463              " productos, se pro\n"
464              "baron todas las primitivas.  Una vez que se la dejo vacia "
465              "se apilaron lineas\n"
466              "de texto y luego se desapilaron.  Lo mas importante de esto "
467              "no es utilizar la\n"
468              "misma pila, lo que mas importa es que con las mismas primiti"
469              "vas se pueden api-\n"
470              "lar distintos tipos de objetos, incluyendo lineas de texto d"
471              "e distinto tamano.\n");
472     }
473 
474 
475     /**
476     int _probarLlenaYApilar(tPila *pila)
477     {
478         tProd   prod;
479         int     cant = 0;
480 
481         puts("Probando pila llena y poner en pila.");
482         mostrarProducto(NULL);
483         while(!pilaLlena(pila, sizeof(prod)) && ingresarProducto(&prod))
484         {
485             if(!ponerEnPila(pila, &prod, sizeof(prod)))
486             {
487                 fprintf(stderr, "ERROR - inesperado - pila llena.\n");
488                 puts("no se pudo cargar la informacion y"
489                      " habria que tomar alguna decision drastica.");
490             }
491             mostrarProducto(&prod);
492             cant++;
493         }
494         return cant;
495     }
496 
497     void _probarVerTope(tPila *pila)
498     {
499         tProd   prod;
500 
501         puts("Probando ver el tope de la pila.");
502         if(verTope(pila, &prod, sizeof(prod)))
503         {
504             mostrarProducto(NULL);
```

```
505         mostrarProducto(&prod);
506     }
507     else
508         puts("La pila estaba vacia.");
509     puts("");
510 }
511
512 void _probarVaciaYDesapilarN(tPila *pila, int canti)
513 {
514     tProd   prod;
515
516     printf("Probando pila vacia y sacar de pila %d productos (mostrandolos.\n",
517            canti);
518     if(pilaVacia(pila))
519         puts("La pila esta vacia.");
520     else
521         mostrarProducto(NULL);
522     while(canti > 0 && sacarDePila(pila, &prod, sizeof(prod)))
523     {
524         canti--;
525         mostrarProducto(&prod);
526     }
527     puts("");
528 }
529
530 void probarPonerYSacarDePila_2(void)
531 {
532     tPila   pila;
533     int     cant;
534
535     crearPila(&pila);
536
537     puts("Probando primitivas de pila con productos.\n"
538          "======== ========== == ==== === =========\n");
539     cant = _probarLlenaYApilar(&pila);
540     printf("se pusieron %d productos en la pila.\n\n", cant);
541
542     _probarVerTope(&pila);
543
544     _probarVaciaYDesapilarN(&pila, cant - 2);
545
546 TERMINE DE DESARROLLAR LAS RESTANTES
547     puts("Probando ver el tope de la pila.");
548     if(verTope(&pila, &otro, sizeof(otro)))
549     {
550         puts("La pila no quedo vacia - en el tope hay ...");
551         mostrarProducto(NULL);
552         mostrarProducto(&otro);
553     }
554     else
555         puts("La pila esta vacia.");
556     puts("");
557     puts("Probando vaciar pila y pila vacia.");
558     vaciarPila(&pila);
559     if(!pilaVacia(&pila))
560         fprintf(stderr, "ERROR - la pila debia estar vacia\n\n");
561     else
562         printf("Vaciar pila funciona!\n\n");
563     puts("");
564
565     puts("Probando primitivas de pila con lineas de texto.\n"
566          "======== ========== == ==== === ====== == =====\n"
567          "Probando pila llena y poner en pila.");
568     cant = 0;
569     while(!pilaLlena(&pila, sizeof(prod)) && ingresarTexto(linea, sizeof(linea)))
570     {
571         if(!ponerEnPila(&pila, linea, strlen(linea) + 1))
572         {
573             fprintf(stderr, "ERROR - inesperado - pila llena.\n");
574             puts("no se pudo cargar la informacion y"
575                  " habria que tomar alguna decision drastica.");
576         }
577         printf("\"%s\"\n", linea);
578         cant++;
579     }
580     printf("se pusieron %d lineas de texto en la pila.\n\n", cant);
581     printf("Probando sacar de pila con las lineas de texto.\n");
582     cant = 0;
583     while(sacarDePila(&pila, linea, sizeof(linea)))
584     {
585         cant++;
586         printf("\"%s\"\n", linea);
587     }
588     printf("Se sacaron y mostraron %d lineas de texto\n\n", cant);
```

```
589         puts("ATENCION: se mostro el uso de una pila, en la que se apilaron"
590              " productos, se pro\n"
591              "baron todas las primitivas.  Una vez que se la dejo vacia "
592              "se apilaron lineas\n"
593              "de texto y luego se desapilaron.  Lo mas importante de esto "
594              "no es utilizar la\n"
595              "misma pila, lo que mas importa es que con las mismas primiti"
596              "vas se pueden api-\n"
597              "lar distintos tipos de objetos, incluyendo lineas de texto d"
598              "e distinto tamano.\n");
599     }
600      **/
601     /* ---------------------------------o---x---o---------------------------------
602      *          main.h     prueba del TDA PILA con asignación estática de memoria
603      * ---------------------------------o---x---o--------------------------------- */
604
605     #ifndef MAIN_H_
606     #define MAIN_H_
607
608     #include <stdio.h>
609
610
611     #include "../productos/productos.h"
612     #include "../lineasDeTexto/lineasTexto.h"
613     #include "../pilaEstatica/pila.h"
614
615     void probarIngresarYMostrarProd(void);
616
617     void probarIngresarYMostrarTexto(void);
618
619     void probarPonerYSacarDePila(void);
620
621     #endif // MAIN_H_
622
623     /* ---------------------------------o---x---o---------------------------------
624      *          productos.c     ingreso sintético de productos
625      * ---------------------------------o---x---o--------------------------------- */
626
627     #include "productos.h"
628
629     int ingresarProducto(tProd *d)
630     {
631         static const tProd productos[] = {
632             ///1234567890    123456789 123456789 123456789 123456789 12345
633             { "clavoro3/4", "Clavo de oro 24 kilates de 3/4 de pulgada" },
634             { "martillo3K", "Martillo bolita con saca clavos de 3 kilos"},
635             { "alamyeso1",  "Alambre de yeso de un milimetro de espesor" },
636             { "rem-vid15",  "Remache de vidrio de 1,5 milimetros" },
637             { "plom-telgo", "Plomada de poliestireno expandido" },
638             { "limagoma17", "Lima de goma de 17 pulgadas"} };
639         static int posi = 0;
640
641         if(posi == sizeof(productos) / sizeof(tProd))
642         {
643             posi = 0;
644             return 0;
645         }
646         *d = productos[posi];
647         posi++;
648
649         return 1;
650     }
651
652     void mostrarProducto(const tProd *d)
653     {
654         if(d)
655             fprintf(stdout,
656                     "%-*s %-*s ...\n",
657                     sizeof(d->codProd) - 1, d->codProd,
658                     sizeof(d->descrip) - 1, d->descrip);
659         else
660             fprintf(stdout,
661                     "%-*.*s %-*.*s ...\n",
662                     sizeof(d->codProd) - 1, sizeof(d->codProd) - 1, "Cod. Producto",
663                     sizeof(d->descrip) - 1, sizeof(d->descrip) - 1,
664                         "Descripcion del producto");
665     }
666
667
668
669
670
671
672
```

```
673
674
675  /* ---------------------------------o---x---o---------------------------------
676   *          productos.h     ingreso sintético de productos
677   * ---------------------------------o---x---o--------------------------------- */
678
679  #ifndef PRODUCTOS_H_
680  #define PRODUCTOS_H_
681
682  #include <stdio.h>
683
684
685  typedef struct
686  {
687      char    codProd[11],
688              descrip[46];
689  } tProd;
690
691
692  int ingresarProducto(tProd *d);
693
694  void mostrarProducto(const tProd *d);
695
696
697  #endif // PRODUCTOS_H_
698  /* ---------------------------------o---x---o---------------------------------
699   *          lineasTexto.c     ingreso sintético de lineas de texto
700   * ---------------------------------o---x---o--------------------------------- */
701
702  #include "..\lineasDeTexto\lineasTexto.h"
703
704
705  int ingresarTexto(char *linea, int tamLinea)
706  {
707      static const char *texto[] = {
708              "Se necesita un amigo - Fragmento del Poema de Vinicius de Moraes",
709              "",
710              "Debe tener un ideal, y miedo de perderlo,",
711              "y en caso de no ser asi,",
712              "debe sentir el gran vacio que esto deja.",
713              "Tiene que tener resonancias humanas,",
714              "su principal objetivo debe ser el del amigo.",
715              "Debe sentir pena por las personas tristes",
716              "y comprender el inmenso vacio de los solitarios.",
717              "Se busca un amigo para gustar",
718              "de los mismos gustos,",
719              "que se conmueva cuando es tratado de amigo.",
720              "",
721              NULL };
722      static int posi = 0;
723
724      if(texto[posi] == NULL)
725      {
726          posi = 0;
727          return 0;
728      }
729      *linea = '\0';
730      strncat(linea, texto[posi], tamLinea - 1);
731      posi++;
732      return 1;
733  }
734  /* ---------------------------------o---x---o---------------------------------
735   *          lineasTexto.h     ingreso sintético de lineas de texto
736   * ---------------------------------o---x---o--------------------------------- */
737
738  #ifndef LINEASTEXTO_H_
739  #define LINEASTEXTO_H_
740
741  #include <string.h>
742
743
744  int ingresarTexto(char *linea, int tamLinea);
745
746
747  #endif // LINEASTEXTO_H_
748  /* ---------------------------------o---x---o---------------------------------
749   *          pila.c     TDA PILA con asignación dinámica de memoria
750   * ---------------------------------o---x---o--------------------------------- */
751
752  #include "pila.h"
753
754
755  #define minimo( X , Y )      ( ( X ) <= ( Y ) ? ( X ) : ( Y ) )
756
```

```c

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;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
    (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}

int  verTope(const tPila *p, void *d, unsigned cantBytes)
{
    if(*p == NULL)
        return 0;
    memcpy(d, (*p)->info, minimo(cantBytes, (*p)->tamInfo));
    return 1;
}

int  pilaVacia(const tPila *p)
{
    return *p == NULL;
}

int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
{
    tNodo *aux = *p;

    if(aux == NULL)
        return 0;
    *p = aux->sig;
    memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
    free(aux->info);
    free(aux);
    return 1;
}

void vaciarPila(tPila *p)
{
    while(*p)
    {
        tNodo *aux = *p;

        *p = aux->sig;
        free(aux->info);
        free(aux);
    }
}

/* ---------------------------------o---x---o---------------------------------
 *          pila.h     TDA PILA con asignación dinámica de memoria
 * ---------------------------------o---x---o--------------------------------- */

#ifndef PILA_H_
#define PILA_H_

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


typedef struct sNodo
{
```

```
841         void            *info;
842         unsigned         tamInfo;
843         struct sNodo    *sig;
844     } tNodo;
845     typedef tNodo *tPila;
846
847     void crearPila(tPila *p);
848     int  pilaLlena(const tPila *p, unsigned cantBytes);
849     int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes);
850     int  verTope(const tPila *p, void *d, unsigned cantBytes);
851     int  pilaVacia(const tPila *p);
852     int  sacarDePila(tPila *p, void *d, unsigned cantBytes);
853     void vaciarPila(tPila *p);
854
855     #endif
856
857     /* ------------------------------o---x---o------------------------------
858      *          pila.c     ESTÁTICA
859      * ------------------------------o---x---o------------------------------ */
860
861     #include "pila.h"
862
863
864     void crearPila(tPila *p)
865     {
866         p->tope = TAM_PILA;
867     }
868
869     int  pilaLlena(const tPila *p, unsigned cantBytes)
870     {
871         return p->tope < cantBytes + sizeof(unsigned);
872     }
873
874     int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
875     {
876         if(p->tope < cantBytes + sizeof(unsigned))
877             return 0;
878         p->tope -= cantBytes;
879         memcpy(p->pila + p->tope, d, cantBytes);
880         p->tope -= sizeof(unsigned);
881         memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
882         return 1;
883     }
884
885     int  verTope(const tPila *p, void *d, unsigned cantBytes)
886     {
887         unsigned tamInfo;
888
889         if(p->tope == TAM_PILA)
890             return 0;
891         memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
892         memcpy(d, p->pila + p->tope + sizeof(unsigned),
893             minimo(cantBytes, tamInfo));
894         return 1;
895     }
896
897     int  pilaVacia(const tPila *p)
898     {
899         return p->tope == TAM_PILA;
900     }
901
902     int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
903     {
904         unsigned tamInfo;
905
906         if(p->tope == TAM_PILA)
907             return 0;
908         memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
909         p->tope += sizeof(unsigned);
910         memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
911         p->tope += tamInfo;
912         return 1;
913     }
914
915     void vaciarPila(tPila *p)
916     {
917         p->tope = TAM_PILA;
918     }
919
920     /* ------------------------------o---x---o------------------------------
921      *          pila.h     ESTÁTICA
922      * ------------------------------o---x---o------------------------------ */
923
924     #ifndef PILA_H_
```

```
925  #define PILA_H_
926
927
928  #include <string.h>
929
930  #define minimo( X , Y )     ( ( X ) <= ( Y ) ? ( X ) : ( Y ) )
931
932
933  #define     TAM_PILA        340
934
935  typedef struct
936  {
937      char        pila[TAM_PILA];
938      unsigned    tope;
939  } tPila;
940
941  void crearPila(tPila *p);
942  int  pilaLlena(const tPila *p, unsigned cantBytes);
943  int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes);
944  int  verTope(const tPila *p, void *d, unsigned cantBytes);
945  int  pilaVacia(const tPila *p);
946  int  sacarDePila(tPila *p, void *d, unsigned cantBytes);
947  void vaciarPila(tPila *p);
948
949  #endif
950
951  SUB
```