```c
/* --------------------o---x---o--------------------
 *           main.h
 * --------------------o---x---o--------------------*/
#ifndef MAIN_H_

#define MAIN_H_


#include <stdio.h>

#include "productos/productos.h"
#include "pilaEstatica/pila.h"
#include "pilaDinamica/pila.h"


void probarIngresarYMostrarProd(void);

void probarPonerYSacarDePila(void);

#endif
/* --------------------o---x---o--------------------
 *           main.c
 * --------------------o---x---o--------------------*/
#include "main.h"


int main(void)
{
    printf("Tamano tProd: %d\n", sizeof(tProd));
    probarIngresarYMostrarProd();

    probarPonerYSacarDePila();

    return 0;
}


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

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

void probarPonerYSacarDePila(void)
{
    tProd    prod;
    tPila    pila;
    int      result,
             llena;

    crearPila(&pila);
    llena = pilaLlena(&pila, sizeof(prod));
    if(!llena)
    {
        result = ingresarProducto(&prod);
        puts("Procediendo a poner en pila");
        mostrarProducto(NULL);
    }
    while(result && !llena)
    {

```

```
        if(!ponerEnPila(&pila, &prod, sizeof(prod)))CRLF
        {CRLF
            fprintf(stderr, "ERROR - inesperado: pila llena\n");CRLF
            puts("no se pudo cargar la informacion");CRLF
        }CRLF
        mostrarProducto(&prod);CRLF
        llena = pilaLlena(&pila, sizeof(prod));CRLF
        if(!llena)CRLF
            result = ingresarProducto(&prod);CRLF
        elseCRLF
            puts("Se lleno la pila");CRLF
    }CRLF
CRLF
    puts("\nMostrando el tope de la pila");CRLF
    if(!pilaVacia(&pila))CRLF
    {CRLF
        tProd otro;CRLF
        verTope(&pila, &otro, sizeof(otro));CRLF
        mostrarProducto(&otro);CRLF
    }CRLF
    elseCRLF
        puts("La pila estaba vacia");CRLF
CRLF
    puts("\nProcediendo a sacar de la pila y mostrar");CRLF
    if(pilaVacia(&pila))CRLF
        puts("La pila está vacía");CRLF
    elseCRLF
        mostrarProducto(NULL);CRLF
    while(sacarDePila(&pila, &prod, sizeof(prod)))CRLF
        mostrarProducto(&prod);CRLF
    puts("");CRLF
}CRLF
CRLF
/* ---------------------o---x---o---------------------CRLF
 *          productos.hCRLF
 * ---------------------o---x---o--------------------- */CRLF
#ifndef PRODUCTOS_H_CRLF
#define PRODUCTOS_H_CRLF
CRLF
#include <stdio.h>CRLF
CRLF
CRLF
typedef structCRLF
{CRLF
    char    codProd[11],CRLF
            descrip[46];CRLF
} tProd;CRLF
CRLF
int ingresarProducto(tProd *d);CRLF
CRLF
void mostrarProducto(const tProd *d);CRLF
CRLF
CRLF
#endifCRLF
/* ---------------------o---x---o---------------------CRLF
 *          productos.cCRLF
 * ---------------------o---x---o--------------------- */CRLF
#include "productos.h"CRLF
CRLF
CRLF
int ingresarProducto(tProd *d)CRLF
{CRLF
    static const tProd productos[] = {CRLF
        ///1234567890     123456789 123456789 123456789 123456789 12345CRLF
        { "clavoro3/4", "Clavo de oro 24 kilates de 3/4 de pulgada" },CRLF
        { "martillo3K", "Martillo bolita con saca clavos de 3 kilos"},CRLF
        { "alamyeso1",  "Alambre de yeso de un milimetro de espesor" },CRLF
        { "rem-vid15",  "Remache de vidrio de 1,5 milimetros" },CRLF
        { "plom-telgo", "Plomada de poliestireno expandido" },CRLF
        { "limagoma17", "Lima de goma de 17 pulgadas"} };CRLF
    static int posi = 0;CRLF
CRLF
    if(posi == sizeof(productos) / sizeof(tProd))CRLF
```

```
    {CRLF
        posi = 0;CRLF
        return 0;CRLF
    }CRLF
    *d = productos[posi];CRLF
    posi++;CRLF
CRLF
    return 1;CRLF
}CRLF
CRLF
void mostrarProducto(const tProd *d)CRLF
{CRLF
    if(d)CRLF
        fprintf(stdout,CRLF
                "%-*s %-*s ...\n",CRLF
                sizeof(d->codProd) - 1, d->codProd,CRLF
                sizeof(d->descrip) - 1, d->descrip);CRLF
    elseCRLF
        fprintf(stdout,CRLF
                "%-*.*s %-*.*s ...\n",CRLF
                sizeof(d->codProd) - 1, sizeof(d->codProd) - 1,CRLF
                    "Cod. Producto",CRLF
                sizeof(d->descrip) - 1, sizeof(d->descrip) - 1,CRLF
                    "Descripcion del producto");CRLF
}CRLF
CRLF
/* ---------------------o---x---o----------------------CRLF
 *          pila.h      ESTÁTICACRLF
 * ---------------------o---x---o---------------------- */CRLF
#ifdef ESTATICACRLF
CRLF
#ifndef PILA_H_CRLF
#define PILA_H_CRLF
CRLF
/// pila ESTÁTICACRLF
CRLF
#include <string.h>CRLF
CRLF
#define minimo( X , Y )     ( ( X ) <= ( Y ) ? ( X ) : ( Y ) )CRLF
CRLF
CRLF
#define     TAM_PILA        340CRLF
CRLF
typedef structCRLF
{CRLF
    char        pila[TAM_PILA];CRLF
    unsigned    tope;CRLF
} tPila;CRLF
CRLF
void crearPila(tPila *p);CRLF
int  pilaLlena(const tPila *p, unsigned cantBytes);CRLF
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes);CRLF
int  verTope(const tPila *p, void *d, unsigned cantBytes);CRLF
int  pilaVacia(const tPila *p);CRLF
int  sacarDePila(tPila *p, void *d, unsigned cantBytes);CRLF
void vaciarPila(tPila *p);CRLF
CRLF
#endifCRLF
CRLF
#endifCRLF
/* ---------------------o---x---o----------------------CRLF
 *          pila.c      ESTÁTICACRLF
 * ---------------------o---x---o---------------------- */CRLF
#ifdef ESTATICACRLF
CRLF
/// pila ESTÁTICACRLF
CRLF
#include "pila.h"CRLF
CRLF
CRLF
void crearPila(tPila *p)CRLF
{CRLF
    p->tope = TAM_PILA;CRLF
```

```
}CRLF
CRLF
int pilaLlena(const tPila *p, unsigned cantBytes)CRLF
{CRLF
    return p->tope < cantBytes + sizeof(unsigned);CRLF
}CRLF
CRLF
int ponerEnPila(tPila *p, const void *d, unsigned cantBytes)CRLF
{CRLF
    if(p->tope < cantBytes + sizeof(unsigned))CRLF
        return 0;CRLF
    p->tope -= cantBytes;CRLF
    memcpy(p->pila + p->tope, d, cantBytes);CRLF
    p->tope -= sizeof(unsigned);CRLF
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));CRLF
    return 1;CRLF
}CRLF
CRLF
int verTope(const tPila *p, void *d, unsigned cantBytes)CRLF
{CRLF
    unsigned tamInfo;CRLF
CRLF
    if(p->tope == TAM_PILA)CRLF
        return 0;CRLF
    memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));CRLF
    memcpy(d, p->pila + p->tope + sizeof(unsigned),CRLF
           minimo(cantBytes, tamInfo));CRLF
    return 1;CRLF
}CRLF
CRLF
int pilaVacia(const tPila *p)CRLF
{CRLF
    return p->tope == TAM_PILA;CRLF
}CRLF
CRLF
int sacarDePila(tPila *p, void *d, unsigned cantBytes)CRLF
{CRLF
    unsigned tamInfo;CRLF
CRLF
    if(p->tope == TAM_PILA)CRLF
        return 0;CRLF
    memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));CRLF
    p->tope += sizeof(unsigned);CRLF
    memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));CRLF
    p->tope += tamInfo;CRLF
    return 1;CRLF
}CRLF
CRLF
void vaciarPila(tPila *p)CRLF
{CRLF
    p->tope = TAM_PILA;CRLF
}CRLF
CRLF
#endifCRLF
CRLF
/* --------------------o---x---o--------------------CRLF
 *          pila.h      DINÁMICACRLF
 * --------------------o---x---o--------------------*/CRLF
#ifdef DINAMICACRLF
CRLF
#ifndef PILA_H_CRLF
#define PILA_H_CRLF
CRLF
/// pila DINÁMICACRLF
#include <stdlib.h>CRLF
#include <string.h>CRLF
CRLF
#define minimo( X , Y )     ( ( X ) <= ( Y ) ? ( X ) : ( Y ) )CRLF
CRLF
typedef struct sNodoCRLF
{CRLF
    void           *info;CRLF
    unsigned        tamInfo;CRLF
```

```
    struct sNodo    *sig;CRLF
} tNodo;CRLF
typedef tNodo *tPila;CRLF
CRLF
void crearPila(tPila *p);CRLF
int  pilaLlena(const tPila *p, unsigned cantBytes);CRLF
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes);CRLF
int  verTope(const tPila *p, void *d, unsigned cantBytes);CRLF
int  pilaVacia(const tPila *p);CRLF
int  sacarDePila(tPila *p, void *d, unsigned cantBytes);CRLF
void vaciarPila(tPila *p);CRLF
CRLF
#endifCRLF
CRLF
#endifCRLF
/* --------------------o---x---o--------------------CRLF
 *          pila.c      DINÁMICACRLF
 * --------------------o---x---o-------------------- */CRLF
#ifdef DINAMICACRLF
CRLF
CRLF
/// pila DINÁMICACRLF
CRLF
#include "pila.h"CRLF
CRLF
void crearPila(tPila *p)CRLF
{CRLF
    *p = NULL;CRLF
}CRLF
CRLF
int  pilaLlena(const tPila *p, unsigned cantBytes)CRLF
{CRLF
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));CRLF
    void   *info = malloc(cantBytes);CRLF
CRLF
    free(aux);CRLF
    free(info);CRLF
    return aux == NULL || info == NULL;CRLF
}CRLF
CRLF
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)CRLF
{CRLF
    tNodo *nue;CRLF
CRLF
    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||CRLF
       (nue->info = malloc(cantBytes)) == NULL)CRLF
    {CRLF
        free(nue);CRLF
        return 0;CRLF
    }CRLF
    memcpy(nue->info, d, cantBytes);CRLF
    nue->tamInfo = cantBytes;CRLF
    nue->sig = *p;CRLF
    *p = nue;CRLF
    return 1;CRLF
}CRLF
CRLF
int  verTope(const tPila *p, void *d, unsigned cantBytes)CRLF
{CRLF
    if(*p == NULL)CRLF
        return 0;CRLF
    memcpy(d, (*p)->info, minimo(cantBytes, (*p)->tamInfo));CRLF
    return 1;CRLF
}CRLF
CRLF
int  pilaVacia(const tPila *p)CRLF
{CRLF
    return *p == NULL;CRLF
}CRLF
CRLF
int  sacarDePila(tPila *p, void *d, unsigned cantBytes)CRLF
{CRLF
    tNodo *aux = *p;CRLF
```

```
366 CRLF
367     if(aux == NULL)CRLF
368         return 0;CRLF
369     *p = aux->sig;CRLF
370     memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));CRLF
371     free(aux->info);CRLF
372     free(aux);CRLF
373     return 1;CRLF
374 }CRLF
375 CRLF
376 void vaciarPila(tPila *p)CRLF
377 {CRLF
378     while(*p)CRLF
379     {CRLF
380         tNodo *aux = *p;CRLF
381 CRLF
382         *p = aux->sig;CRLF
383         free(aux->info);CRLF
384         free(aux);CRLF
385     }CRLF
386 }CRLF
387 CRLF
388 #endifCRLF
389 CRLF
390
```