

```

1  /*-----o--x--o-----CRLE
2  *.....main.hCRLE
3  /*-----o--x--o-----*/CRLE
4  #ifndef MAIN_H_CRLE
5  #define MAIN_H_CRLE
6  CRLE
7  #include <stdio.h>CRLE
8  CRLE
9  CRLE
10 #include "productos/productos.h"CRLE
11 #include "colaEstatica/cola.h"CRLE
12 #include "colaDinamica/cola.h"CRLE
13 CRLE
14 CRLE
15 void probarIngresarYMostrarProd(void);CRLE
16 CRLE
17 void probarPonerYSacarDeCola(void);CRLE
18 CRLE
19 #endifCRLE
20 /*-----o--x--o-----CRLE
21 *.....main.cCRLE
22 /*-----o--x--o-----*/CRLE
23 #include "main.h"CRLE
24 CRLE
25 CRLE
26 int main(void)CRLE
27 {CRLE
28     printf("%d %d\n", sizeof(unsigned), sizeof(tProd));CRLE
29     probarIngresarYMostrarProd();CRLE
30     CRLE
31     probarPonerYSacarDeCola();CRLE
32     CRLE
33     return 0;CRLE
34 }CRLE
35 CRLE
36 CRLE
37 void probarIngresarYMostrarProd(void)CRLE
38 {CRLE
39     tProd prod;CRLE
40     int result;CRLE
41     int cant = 0;CRLE
42     CRLE
43     puts("Probando ingresar productos y mostrar productos");CRLE
44     result = ingresarProducto(&prod);CRLE
45     if(result)CRLE
46         mostrarProducto(NULL);CRLE
47     while(result)CRLE
48     {CRLE
49         mostrarProducto(&prod);CRLE
50         result = ingresarProducto(&prod);CRLE
51         cant++;CRLE
52     }CRLE
53     printf(stdout, "Se mostraron %d productos.\n\n", cant);CRLE
54 }CRLE
55 CRLE
56 CRLE
57 void probarPonerYSacarDeCola(void)CRLE
58 {CRLE
59     tProd prod;CRLE
60     tCola cola;CRLE
61     int result;CRLE
62     int llena;CRLE
63     CRLE
64     crearCola(&cola);CRLE
65     llena = colaLlena(&cola, sizeof(tProd));CRLE
66     if(!llen)CRLE
67     {CRLE
68         result = ingresarProducto(&prod);CRLE
69         puts("Procediendo a poner en cola");CRLE

```

```

70     ....mostrarProducto(NULL);CRLF
71     ...}CRLF
72     ....while(result && !llena)CRLF
73     ...{CRLF
74     CRLF
75     .....if(!ponerEnCola(&cola, &prod, sizeof(tProd)))CRLF
76     .....{CRLF
77     .....fprintf(stderr, "ERROR - inesperado: cola llena\n");CRLF
78     .....puts("no se pudo cargar la informacion");CRLF
79     .....}CRLF
80     .....mostrarProducto(&prod);CRLF
81     .....llenar = colaLlena(&cola, sizeof(tProd));CRLF
82     .....if(!llenar)CRLF
83     .....result = ingresarProducto(&prod);CRLF
84     .....elseCRLF
85     .....puts("Se lleno la cola");CRLF
86     ....}CRLF
87     CRLF
88     ....puts("\nMostrando el primero de la cola");CRLF
89     ....if(!colaVacia(&cola))CRLF
90     ....{CRLF
91     .....tProd otro;CRLF
92     .....verPrimeroCola(&cola, &otro, sizeof(tProd));CRLF
93     .....mostrarProducto(&otro);CRLF
94     ....}CRLF
95     ....elseCRLF
96     .....puts("La cola estaba vacia");CRLF
97     CRLF
98     ....puts("\nProcediendo a sacar de la cola y mostrar");CRLF
99     ....if(colaVacia(&cola))CRLF
100    .....puts("La cola está vacía");CRLF
101    ....elseCRLF
102    .....mostrarProducto(NULL);CRLF
103    ....while(sacarDeCola(&cola, &prod, sizeof(tProd)))CRLF
104    .....mostrarProducto(&prod);CRLF
105    ....puts("");CRLF
106    }CRLF
107    CRLF
108    /*-----o--x--o-----CRLF
109    *.....productos.hCRLF
110    *-----o--x--o-----*/CRLF
111    #ifndef PRODUCTOS_H_CRLF
112    #define PRODUCTOS_H_CRLF
113    CRLF
114    #include <stdio.h>CRLF
115    CRLF
116    CRLF
117    typedef structCRLF
118    {CRLF
119    ....char codProd[11],CRLF
120    .....descrip[46];CRLF
121    } tProd;CRLF
122    CRLF
123    int ingresarProducto(tProd *d);CRLF
124    CRLF
125    void mostrarProducto(const tProd *d);CRLF
126    CRLF
127    CRLF
128    #endifCRLF
129    /*-----o--x--o-----CRLF
130    *.....productos.cCRLF
131    *-----o--x--o-----*/CRLF
132    #include "productos.h"CRLF
133    CRLF
134    CRLF
135    int ingresarProducto(tProd *d)CRLF
136    {CRLF
137    ....static const tProd productos[] = {CRLF
138    .....////1234567890....123456789..123456789..123456789..123456789..12345

```

```

139     {"clavoro3/4", "Clavo de oro 24 kilates de 3/4 de pulgada"},
140     {"martillo3K", "Martillo bolita con saca clavos de 3 kilos"},
141     {"alamyeso1", "Alambre de yeso de un milimetro de espesor"},
142     {"rem-vid15", "Remache de vidrio de 1,5 milímetros"},
143     {"plom-telgo", "Plomada de poliestireno expandido"},
144     {"limagoma17", "Lima de goma de 17 pulgadas"}];
145 static int posi = 0;
146
147 if (posi == sizeof(productos) / sizeof(tProd))
148 {
149     posi = 0;
150     return 0;
151 }
152 *d = productos[posi];
153 posi++;
154
155 return 1;
156 }
157
158 void mostrarProducto(const tProd *d)
159 {
160     if (d)
161         fprintf(stdout,
162             "%-*s-%*s...\n",
163             sizeof(d->codProd) - 1, d->codProd,
164             sizeof(d->descrip) - 1, d->descrip);
165     else
166         fprintf(stdout,
167             "%-*s-%*s...\n",
168             sizeof(d->codProd) - 1, sizeof(d->codProd) - 1,
169             "Cod. Producto",
170             sizeof(d->descrip) - 1, sizeof(d->descrip) - 1,
171             "Descripcion del producto");
172 }
173
174 /*-----o--x--o-----
175  * cola.h ESTÁTICA
176  *-----o--x--o-----*/
177 #ifdef ESTATICA
178
179 #ifndef COLA_H
180 #define COLA_H
181
182
183 #include <string.h>
184 #include <stdlib.h>
185
186 #define minimo(X, Y) ((X) <= (Y) ? (X) : (Y))
187
188 #define TAM_COLA 300
189
190 typedef struct
191 {
192     char cola[TAM_COLA];
193     unsigned pri,
194     ult,
195     tamDisp;
196 } tCola;
197
198
199 void crearCola(tCola *p);
200 int colaLlena(const tCola *p, unsigned cantBytes);
201 int ponerEnCola(tCola *p, const void *d, unsigned cantBytes);
202 int verPrimeroCola(const tCola *p, void *d, unsigned cantBytes);
203 int colaVacía(const tCola *p);
204 int sacarDeCola(tCola *p, void *d, unsigned cantBytes);
205 void vaciarCola(tCola *p);
206
207 #endif

```

```

208 CRLE
209 #endifCRLE
210 /*-----o--x--o-----CRLE
211 *-----cola.c-----ESTÁTICA-----CRLE
212 *-----o--x--o-----*/CRLE
213 #ifdef ESTÁTICACRLE
214 CRLE
215 CRLE
216 #include "cola.h"CRLE
217 CRLE
218 CRLE
219 void crearCola(tCola *p) CRLE
220 {CRLE
221     p->pri = TAM_COLA - 70;CRLE
222     p->ult = TAM_COLA - 70;CRLE
223     p->tamDisp = TAM_COLA;CRLE
224 }CRLE
225 CRLE
226 int colaLlena(const tCola *p, unsigned cantBytes) CRLE
227 {CRLE
228     return p->tamDisp < cantBytes + sizeof(unsigned);CRLE
229 }CRLE
230 CRLE
231 int ponerEnCola(tCola *p, const void *d, unsigned cantBytes) CRLE
232 {CRLE
233     unsigned ini,CRLE
234     fin;CRLE
235 CRLE
236     if(p->tamDisp < sizeof(unsigned) + cantBytes)CRLE
237         return 0;CRLE
238     p->tamDisp -= sizeof(unsigned) + cantBytes;CRLE
239     if((ini = minimo(sizeof(cantBytes), TAM_COLA - p->ult)) != 0)CRLE
240         memcpy(p->cola + p->ult, &cantBytes, ini);CRLE
241     if((fin = sizeof(cantBytes) - ini) != 0)CRLE
242         memcpy(p->cola, ((char *) &cantBytes) + ini, fin);CRLE
243     p->ult = fin ? fin : p->ult + ini;CRLE
244     if((ini = minimo(cantBytes, TAM_COLA - p->ult)) != 0)CRLE
245         memcpy(p->cola + p->ult, d, ini);CRLE
246     if((fin = cantBytes - ini) != 0)CRLE
247         memcpy(p->cola, ((char *) d) + ini, fin);CRLE
248     p->ult = fin ? fin : p->ult + ini;CRLE
249     return 1;CRLE
250 }CRLE
251 CRLE
252 int verPrimeroCola(const tCola *p, void *d, unsigned cantBytes) CRLE
253 {CRLE
254     unsigned tamInfo,CRLE
255     ini,CRLE
256     fin,CRLE
257     pos = p->pri;CRLE
258 CRLE
259     if(p->tamDisp == TAM_COLA)CRLE
260         return 0;CRLE
261     if((ini = minimo(sizeof(unsigned), TAM_COLA - pos)) != 0)CRLE
262         memcpy(&tamInfo, p->cola + pos, ini);CRLE
263     if((fin = sizeof(unsigned) - ini) != 0)CRLE
264         memcpy(((char *) &tamInfo) + ini, p->cola, fin);CRLE
265     pos = fin ? fin : pos + ini;CRLE
266     tamInfo = minimo(tamInfo, cantBytes);CRLE
267     if((ini = minimo(tamInfo, TAM_COLA - pos)) != 0)CRLE
268         memcpy(d, p->cola + pos, ini);CRLE
269     if((fin = tamInfo - ini) != 0)CRLE
270         memcpy(((char *) d) + ini, p->cola, fin);CRLE
271     return 1;CRLE
272 }CRLE
273 CRLE
274 int colaVacia(const tCola *p) CRLE
275 {CRLE
276     return p->tamDisp == TAM_COLA;CRLE

```

```

277 }CRLE
278 CRLE
279 int sacarDeCola(tCola *p, void *d, unsigned cantBytes)CRLE
280 {CRLE
281     unsigned tamInfo,CRLE
282     ini,CRLE
283     fin;CRLE
284 CRLE
285     if(p->tamDisp == TAM_COLA)CRLE
286         return 0;CRLE
287     if((ini = minimo(sizeof(unsigned), TAM_COLA - p->pri)) != 0)CRLE
288         memcpy(&tamInfo, p->cola + p->pri, ini);CRLE
289     if((fin = sizeof(unsigned) - ini) != 0)CRLE
290         memcpy(((char *)&tamInfo) + ini, p->cola, fin);CRLE
291     p->pri = fin ? fin : p->pri + ini;CRLE
292     tamInfo = minimo(tamInfo, cantBytes);CRLE
293     p->tamDisp += sizeof(unsigned) + tamInfo;CRLE
294     if((ini = minimo(tamInfo, TAM_COLA - p->pri)) != 0)CRLE
295         memcpy(d, p->cola + p->pri, ini);CRLE
296     if((fin = tamInfo - ini) != 0)CRLE
297         memcpy(((char *)d) + ini, p->cola, fin);CRLE
298     p->pri = fin ? fin : p->pri + ini;CRLE
299     return 1;CRLE
300 }CRLE
301 CRLE
302 void vaciarCola(tCola *p)CRLE
303 {CRLE
304     p->ult = p->pri;CRLE
305     p->tamDisp = TAM_COLA;CRLE
306 }CRLE
307 CRLE
308 #endifCRLE
309 CRLE
310 /*-----o-x-o-----CRLE
311 *-----cola.h-----DINÁMICACRLE
312 *-----o-x-o-----*/CRLE
313 #ifdef DINAMICACRLE
314 CRLE
315 #ifndef COLA_H_CRLE
316 #define COLA_H_CRLE
317 CRLE
318 #include <stdlib.h>CRLE
319 #include <string.h>CRLE
320 CRLE
321 #define minimo(X, Y) ((X) <= (Y) ? (X) : (Y))CRLE
322 CRLE
323 typedef struct sNodoCRLE
324 {CRLE
325     void *info;CRLE
326     unsigned tamInfo;CRLE
327     struct sNodo *sig;CRLE
328 } tNodo;CRLE
329 CRLE
330 typedef structCRLE
331 {CRLE
332     tNodo *pri,CRLE
333     *ult;CRLE
334 } tCola;CRLE
335 CRLE
336 void crearCola(tCola *p);CRLE
337 int colaLlena(const tCola *p, unsigned cantBytes);CRLE
338 int ponerEnCola(tCola *p, const void *d, unsigned cantBytes);CRLE
339 int verPrimeroCola(const tCola *p, void *d, unsigned cantBytes);CRLE
340 int colaVacía(const tCola *p);CRLE
341 int sacarDeCola(tCola *p, void *d, unsigned cantBytes);CRLE
342 void vaciarCola(tCola *p);CRLE
343 CRLE
344 #endifCRLE
345 CRLE

```

```

346 #endifCRLE
347 /*-----o--x--o-----CRLE
348 *-----cola.c-----DINÁMICA-----CRLE
349 *-----o--x--o-----*/CRLE
350 #ifndef DINAMICA-----CRLE
351 -----CRLE
352 #include "cola.h"-----CRLE
353 -----CRLE
354 -----CRLE
355 void crearCola(tCola *p)-----CRLE
356 {-----CRLE
357     p->pri = NULL;-----CRLE
358     p->ult = NULL;-----CRLE
359 }-----CRLE
360 -----CRLE
361 int colaLlena(const tCola *p, unsigned cantBytes)-----CRLE
362 {-----CRLE
363     tNodo *aux = (tNodo *) malloc(sizeof(tNodo));-----CRLE
364     void *info = malloc(cantBytes);-----CRLE
365     free(aux);-----CRLE
366     free(info);-----CRLE
367     return aux == NULL || info == NULL;-----CRLE
368 }-----CRLE
369 -----CRLE
370 int ponerEnCola(tCola *p, const void *d, unsigned cantBytes)-----CRLE
371 {-----CRLE
372     tNodo *nue = (tNodo *) malloc(sizeof(tNodo));-----CRLE
373     -----CRLE
374     if(nue == NULL || (nue->info = malloc(cantBytes)) == NULL)-----CRLE
375     {-----CRLE
376         free(nue);-----CRLE
377         return 0;-----CRLE
378     }-----CRLE
379     memcpy(nue->info, d, cantBytes);-----CRLE
380     nue->tamInfo = cantBytes;-----CRLE
381     nue->sig = NULL;-----CRLE
382     if(p->ult)-----CRLE
383         p->ult->sig = nue;-----CRLE
384     else-----CRLE
385         p->pri = nue;-----CRLE
386     p->ult = nue;-----CRLE
387     return 1;-----CRLE
388 }-----CRLE
389 -----CRLE
390 int verPrimeroCola(const tCola *p, void *d, unsigned cantBytes)-----CRLE
391 {-----CRLE
392     if(p->pri == NULL)-----CRLE
393         return 0;-----CRLE
394     memcpy(d, p->pri->info, minimo(cantBytes, p->pri->tamInfo));-----CRLE
395     return 1;-----CRLE
396 }-----CRLE
397 -----CRLE
398 int colaVacia(const tCola *p)-----CRLE
399 {-----CRLE
400     return p->pri == NULL;-----CRLE
401 }-----CRLE
402 -----CRLE
403 int sacarDeCola(tCola *p, void *d, unsigned cantBytes)-----CRLE
404 {-----CRLE
405     tNodo *aux = p->pri;-----CRLE
406     if(aux == NULL)-----CRLE
407         return 0;-----CRLE
408     p->pri = aux->sig;-----CRLE
409     memcpy(d, aux->info, minimo(aux->tamInfo, cantBytes));-----CRLE
410     free(aux->info);-----CRLE
411     free(aux);-----CRLE
412     if(p->pri == NULL)-----CRLE
413         p->ult = NULL;-----CRLE
414     return 1;-----CRLE

```

```
415 }CRLE
416 CRLE
417 void vaciarCola (tCola *p) CRLE
418 {CRLE
419     while (p->pri) CRLE
420     {CRLE
421         tNodo *aux = p->pri; CRLE
422         p->pri = aux->sig; CRLE
423         free (aux->info); CRLE
424         free (aux); CRLE
425     }CRLE
426     p->ult = NULL; CRLE
427 }CRLE
428 CRLE
429 #endifCRLE
430 CRLE
431
```