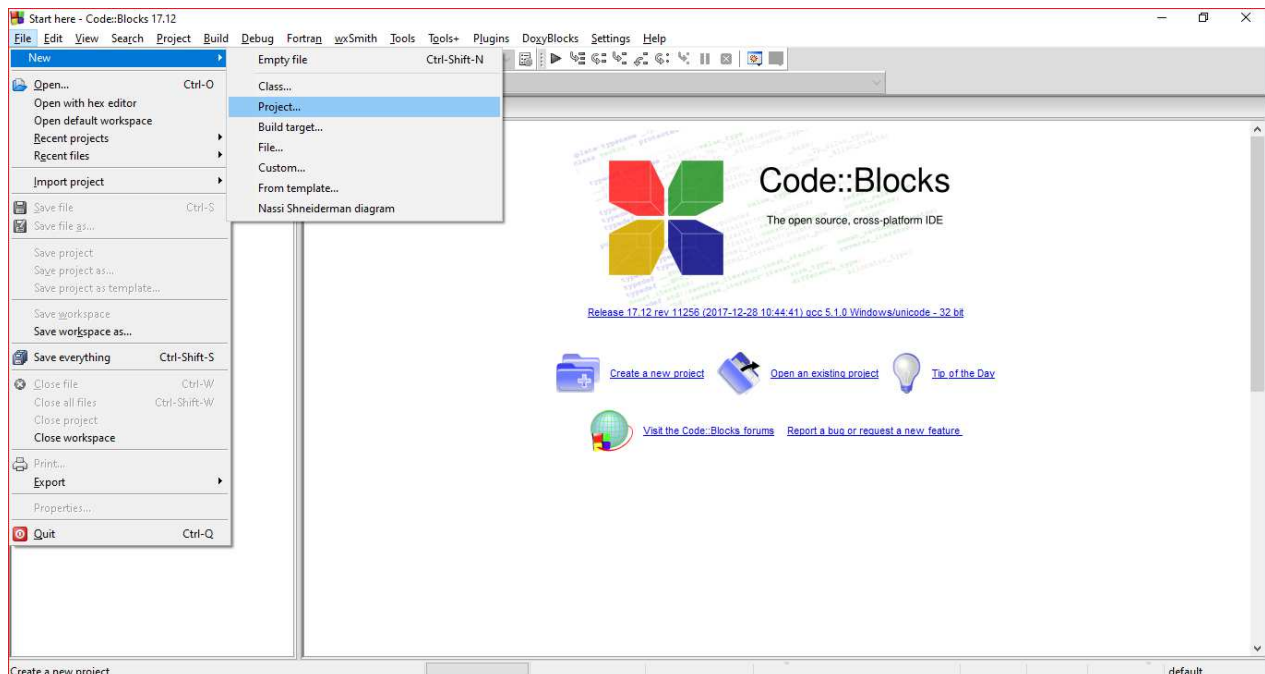


Tipos de Datos Abstractos (TDA) – LISTA.

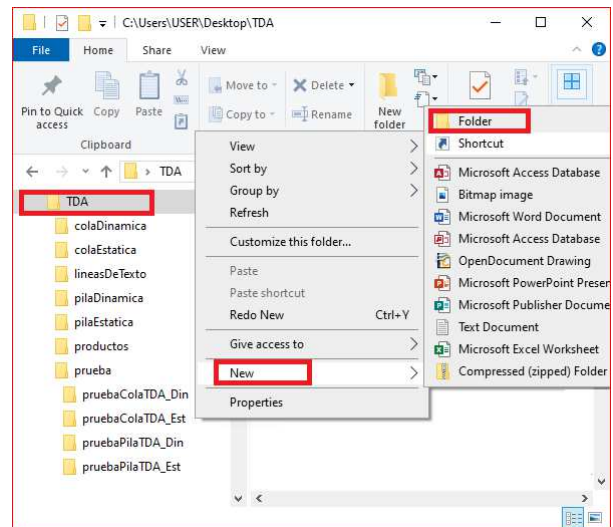
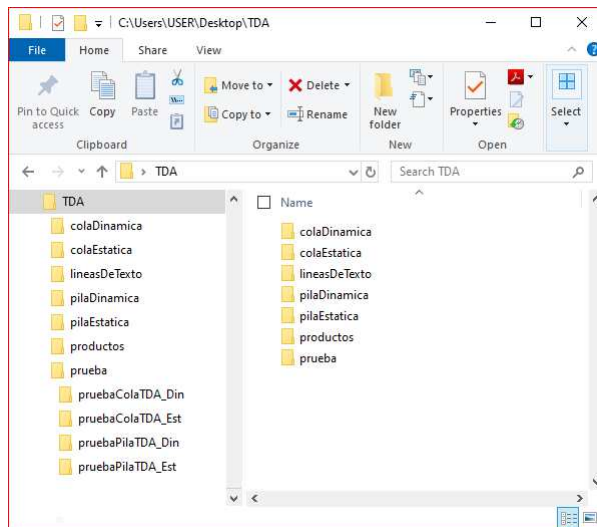
Agregar el proyecto de prueba.

En la carpeta TDA proceder a generar el proyecto "pruebaListaTDA".

Abrir el IDE de Code::Blocks y entrando por el menú desplegable **[File]** / **[New]** / **[Project...]** proceder a generarlo en la sub *carpeta* "TDA\prueba" . . .

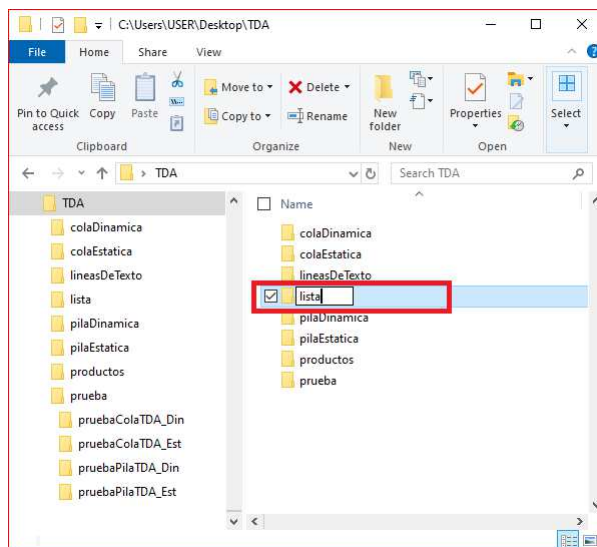


... o de lo contrario, valiéndose del explorador de Windows, proceder a ...

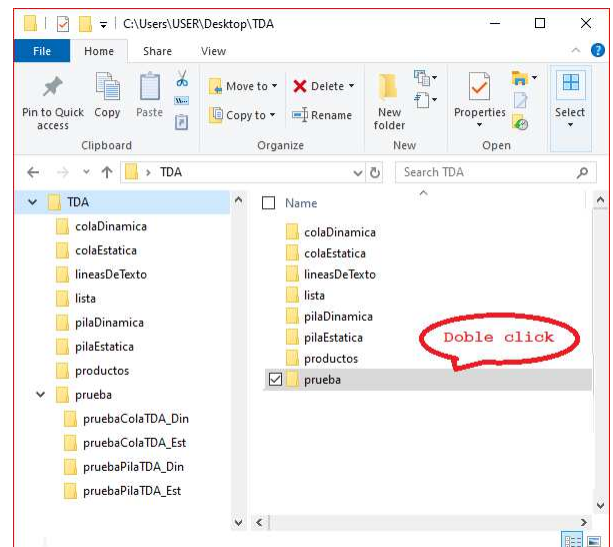


... generar dentro de la carpeta "TDA" ...

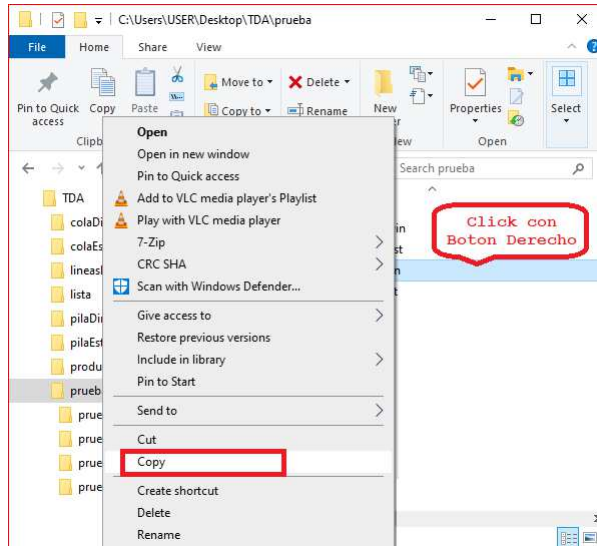
... la subcarpeta ...



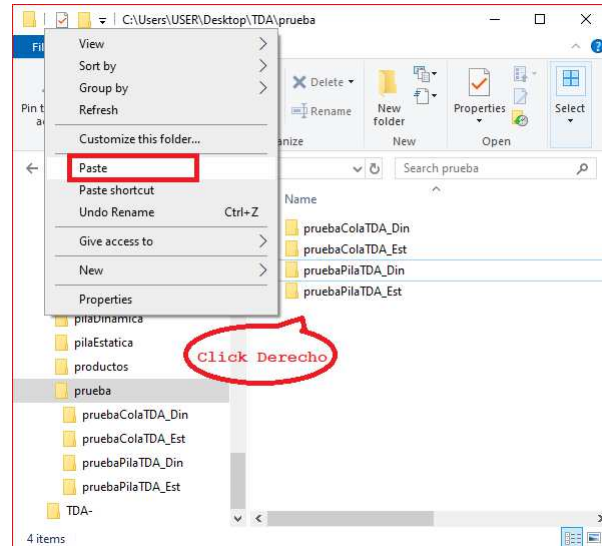
... "lista" ...



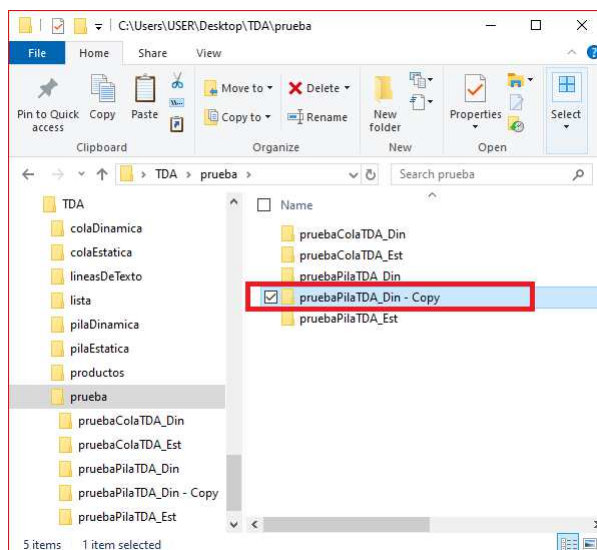
... y en la subcarpeta "TDA\prueba" ...



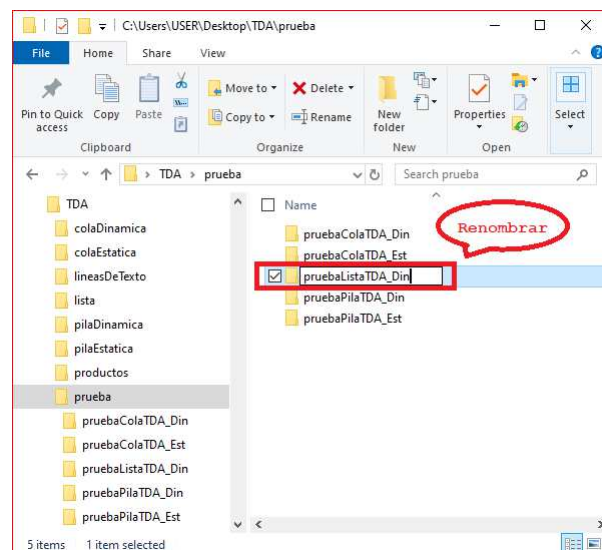
Seleccionar y *copiar*, por ejemplo, la carpeta **"pruebaPilaTDA_Din"**...



... para *pegarla* en la misma carpeta...

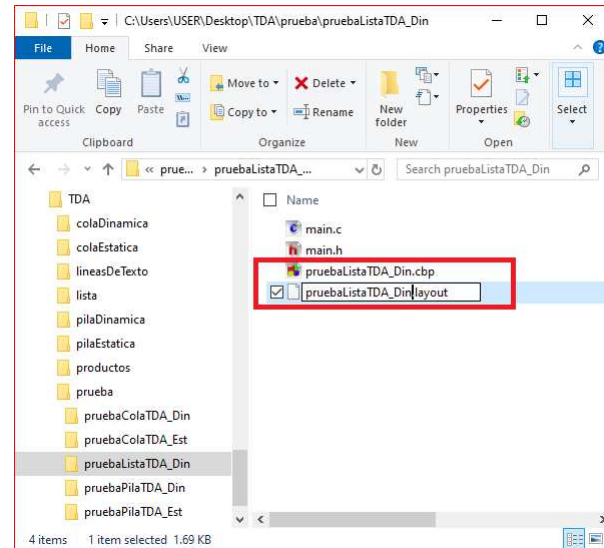
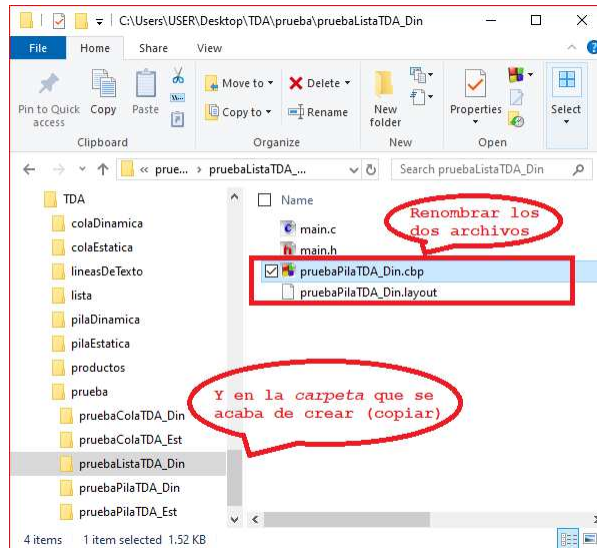


... seleccionar la *carpeta* que se acaba de generar (copiar) para...



... renombrarla como **"pruebaListaTDA_Din"**

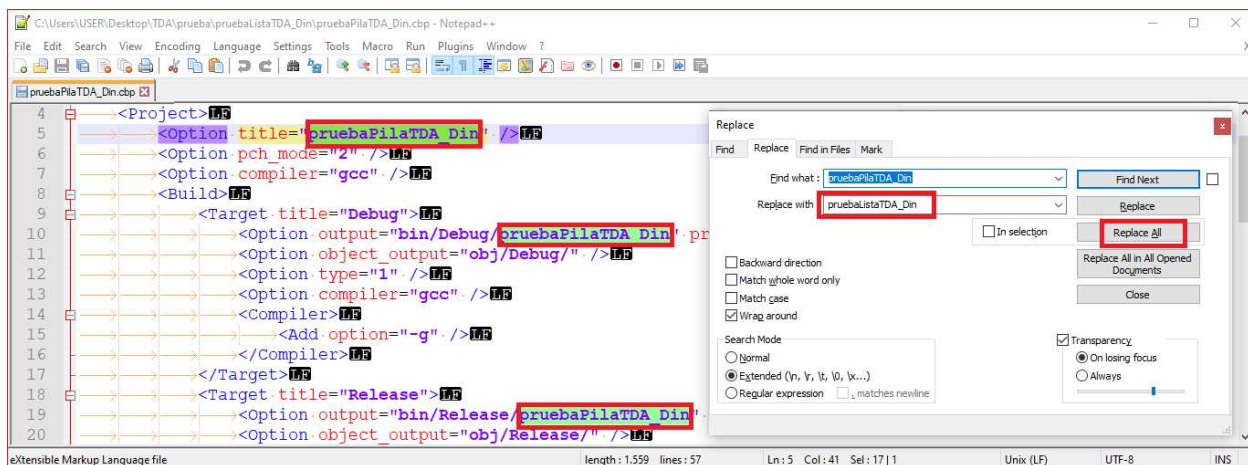
... Y en la carpeta recién generada (copiada), ...



... renombrar ambos archivos del proyecto ...

... como "pruebaListaTDA_Din.cbp" y "pruebaListaTDA_Din.layout".

Editar ambos archivos con "Notepad++".

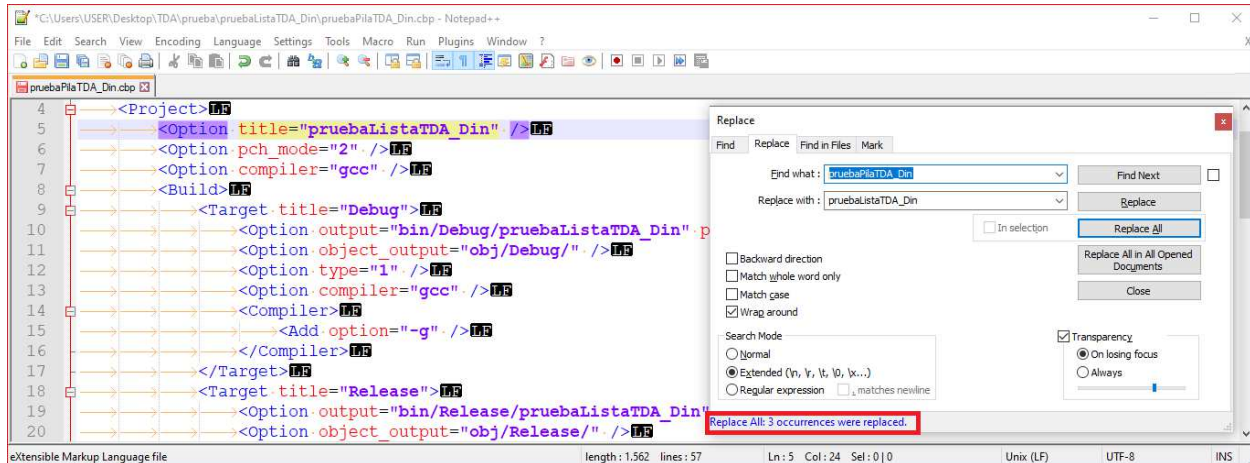


Reemplazar "pruebaPilaTDA_Din" por "pruebaListaTDA_Din" (3 ocurrencias).

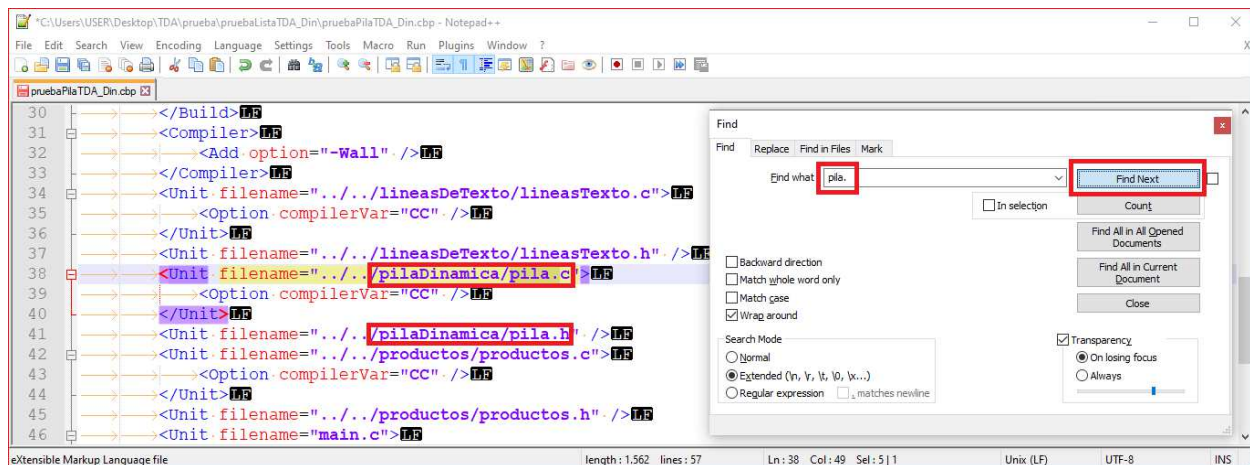


UNLaM

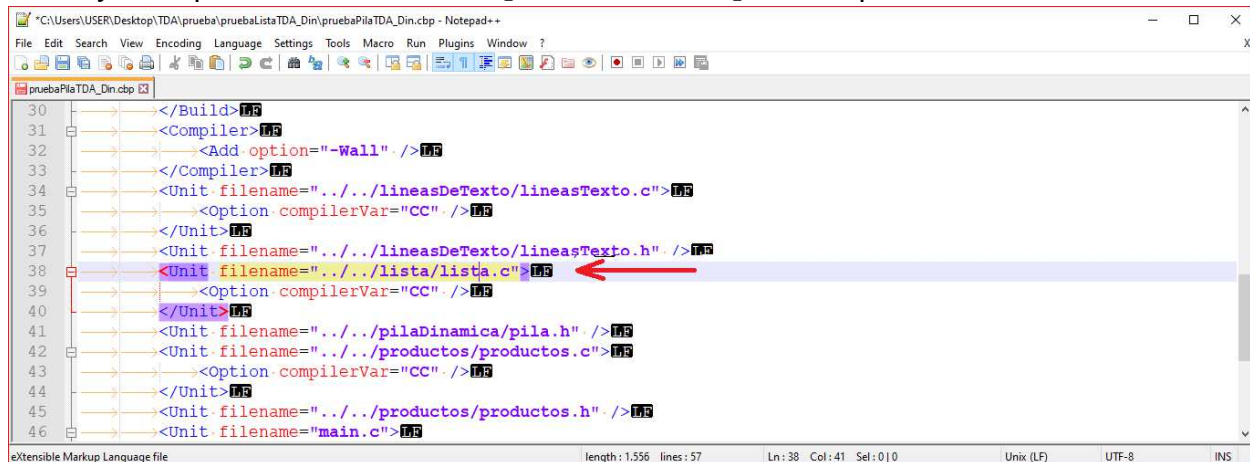
Dto. Ingeniería e Investigaciones Tecnológicas



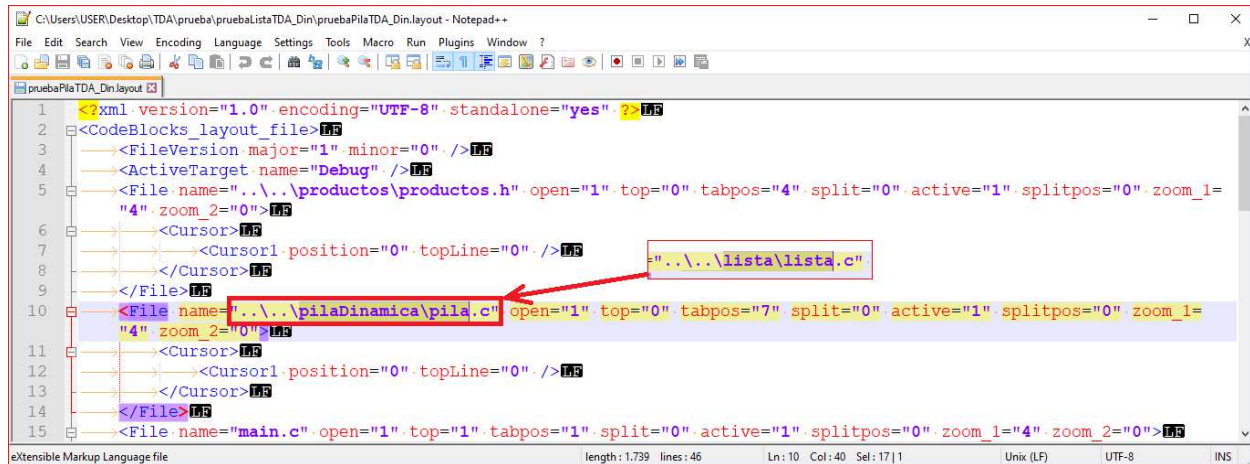
... buscar dentro del archivo la palabra "pila.c" ...



... y reemplazar manualmente "pilaDinamica/pila.c" por "lista/lista.c" ...



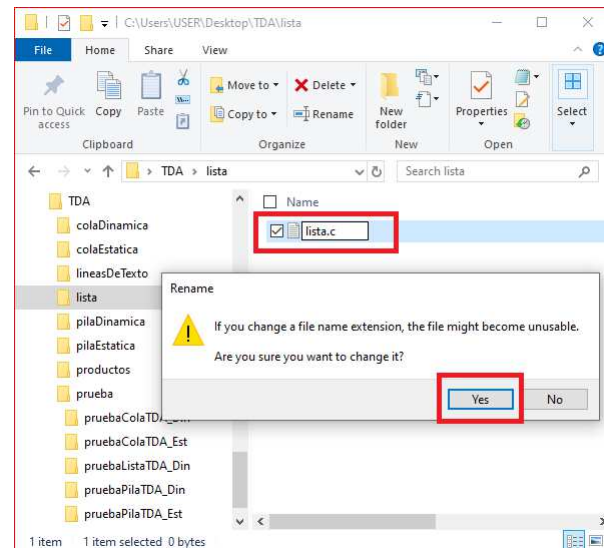
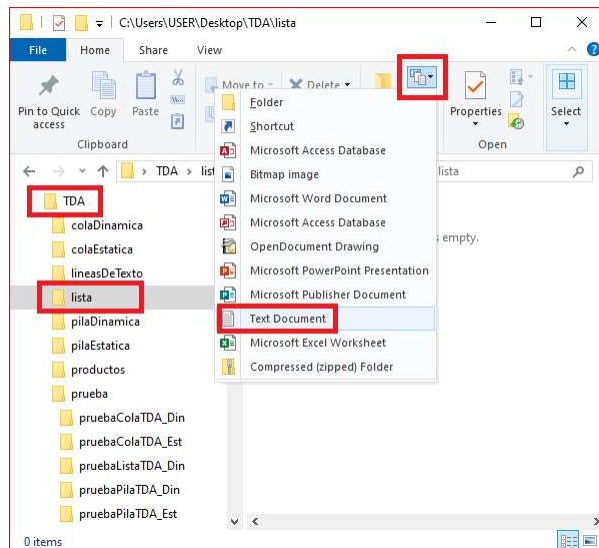
... grabar y cerrar el archivo para editar: "pruebaListaTDA_Din.depend" ...



```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <CodeBlocks_layout_file>
3   <FileVersion major="1" minor="0" />
4   <ActiveTarget name="Debug" />
5   <File name="..\..\productos\productos.h" open="1" top="0" tabpos="4" split="0" active="1" splitpos="0" zoom_1=
6     "4" zoom_2="0" />
7   <Cursor position="0" topLine="0" />
8   <Cursor position="0" topLine="0" />
9   <File name="..\..\pilaDinamica\pila.c" open="1" top="0" tabpos="7" split="0" active="1" splitpos="0" zoom_1=
10     "4" zoom_2="0" />
11   <Cursor position="0" topLine="0" />
12   <Cursor position="0" topLine="0" />
13   <File name="main.c" open="1" top="1" tabpos="1" split="0" active="1" splitpos="0" zoom_1="4" zoom_2="0" />
14
15
  
```

... reemplazar : "pilaDinamica\pila.c" con "lista\lista.c" grabar el archivo ...
 ... luego, en la carpeta "TDA\lista\" ...



... crear un nuevo archivo de texto ...

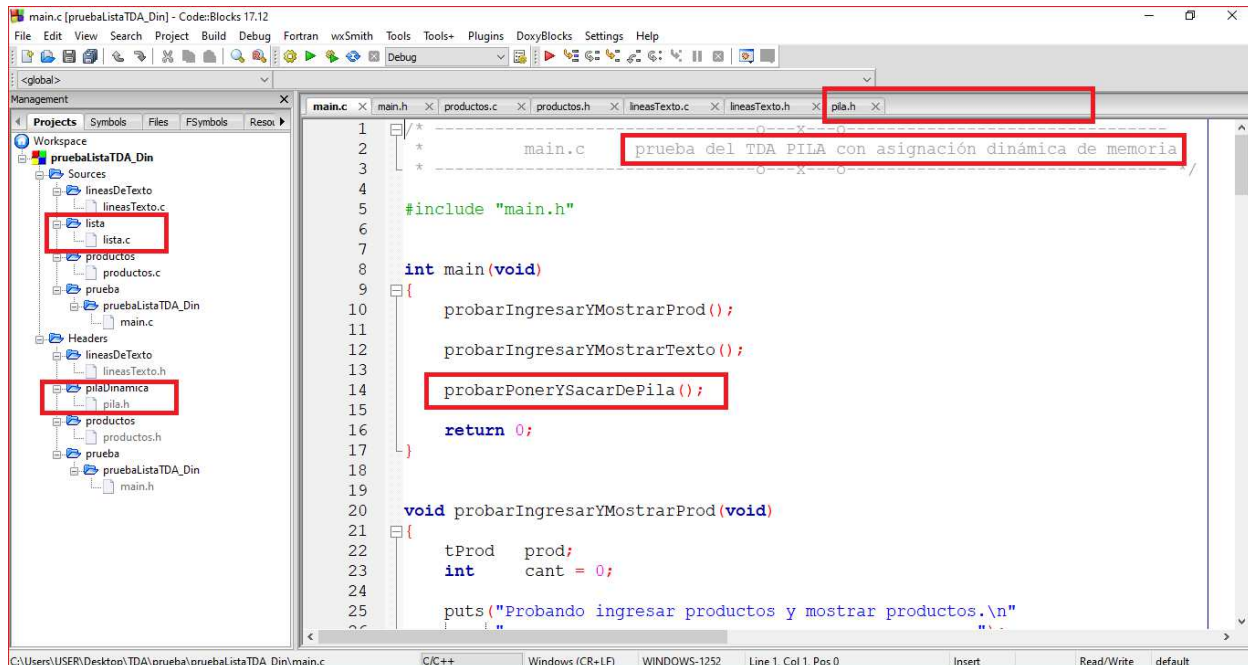
... con el nombre "lista.c" ...

... ir al proyecto en "TDA\prueba\pruebaListaTDA_Din" y editarlo con el IDE ...



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

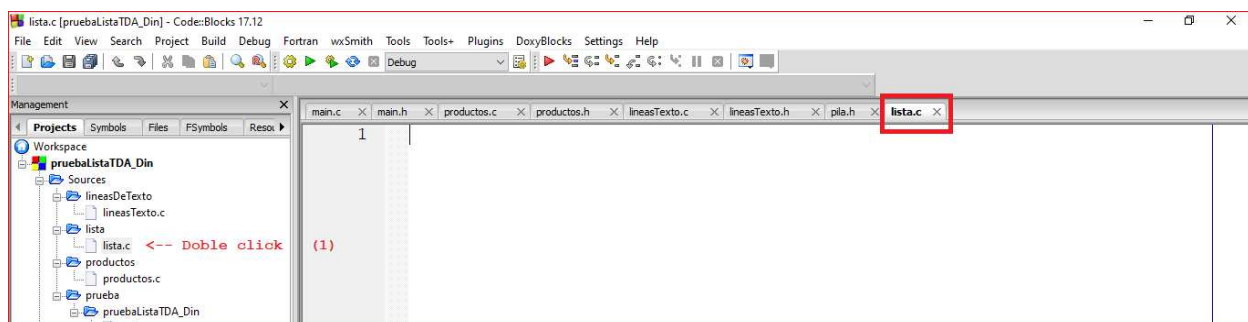


Notar que al expandir en el panel izquierdo ("Workspace") los componentes del proyecto, está el archivo "lista.c", aunque en el panel derecho no lo ha abierto. Hay que hacerle "doble click" para abrirlo ⁽¹⁾. Se verá que está vacío

En cambio el archivo "pila.h", permanece en el proyecto (panel izquierdo) y está disponible para su edición (panel derecho). Hay que eliminarlo del proyecto ⁽²⁾.

En cuanto a "main.c" (y luego en "main.h") se modificará en el comentario "TDA PILA" por "TDA LISTA" ⁽³⁾ y además "probarPonerYSacarDePila();" por "probarPonerYSacarDeLista();" ⁽⁴⁾.

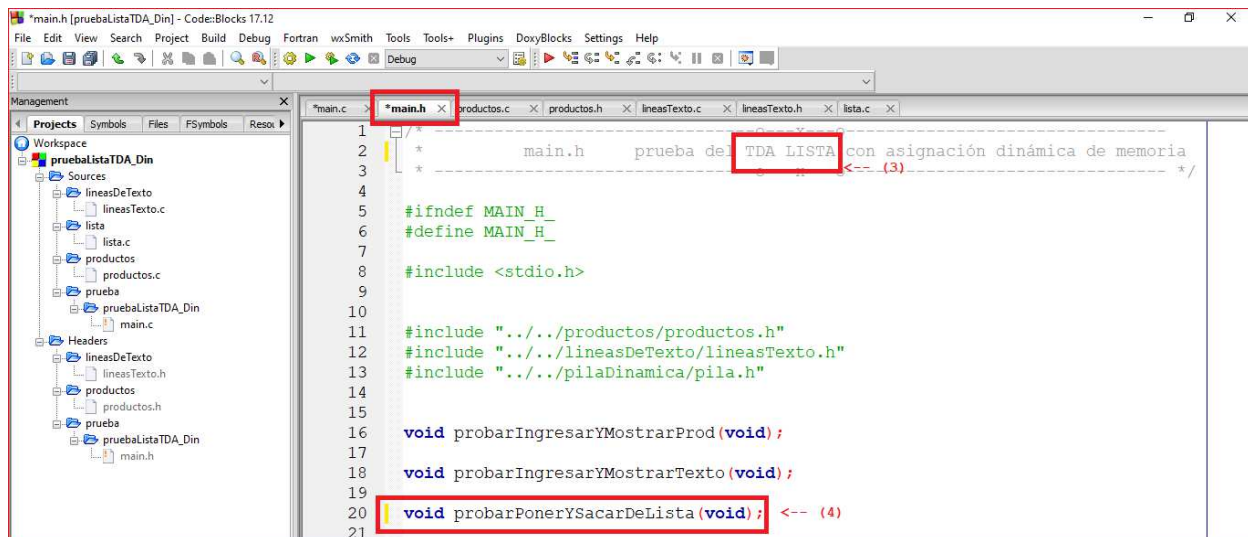
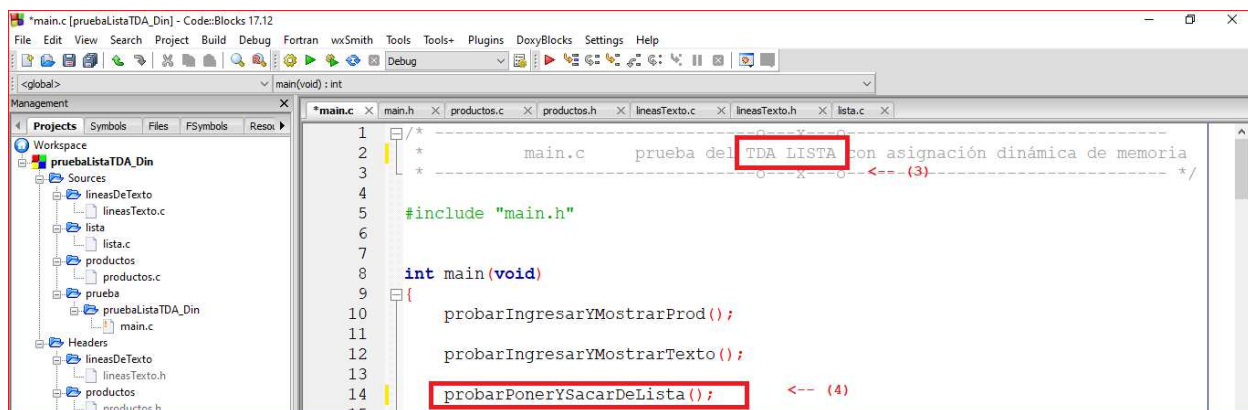
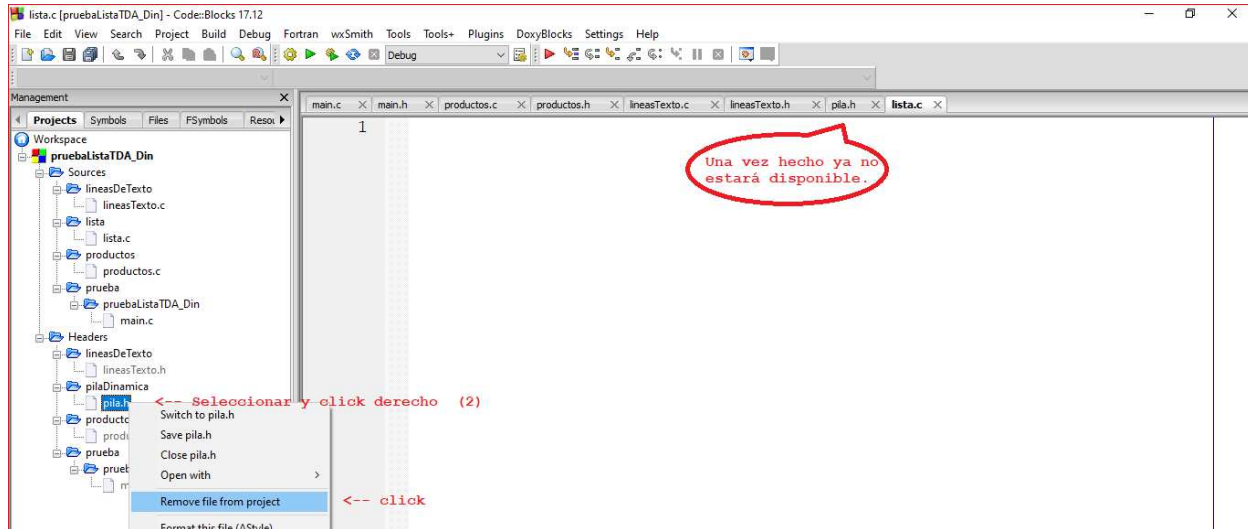
En las siguientes figuras están numerados los pasos indicados.



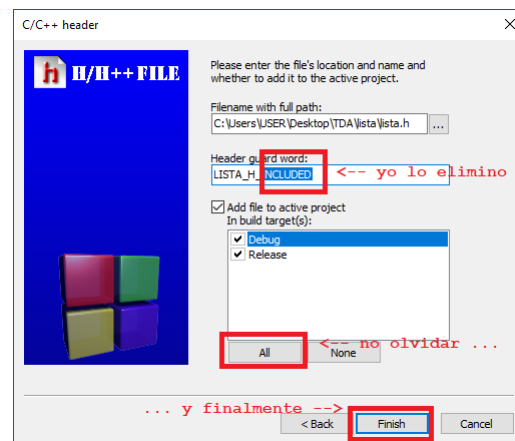
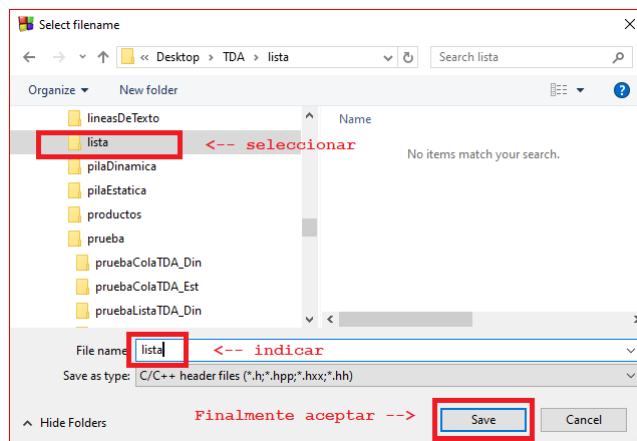
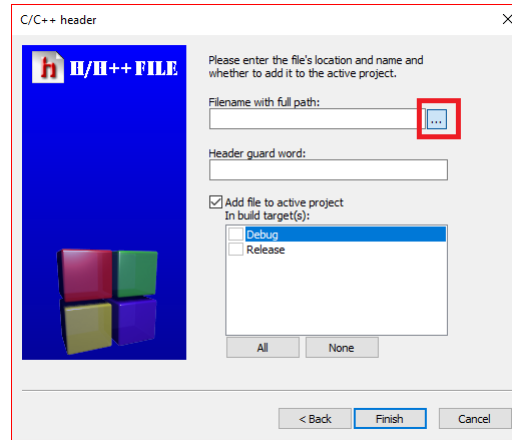
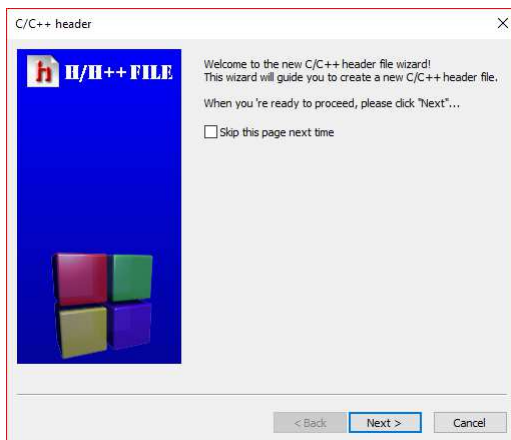
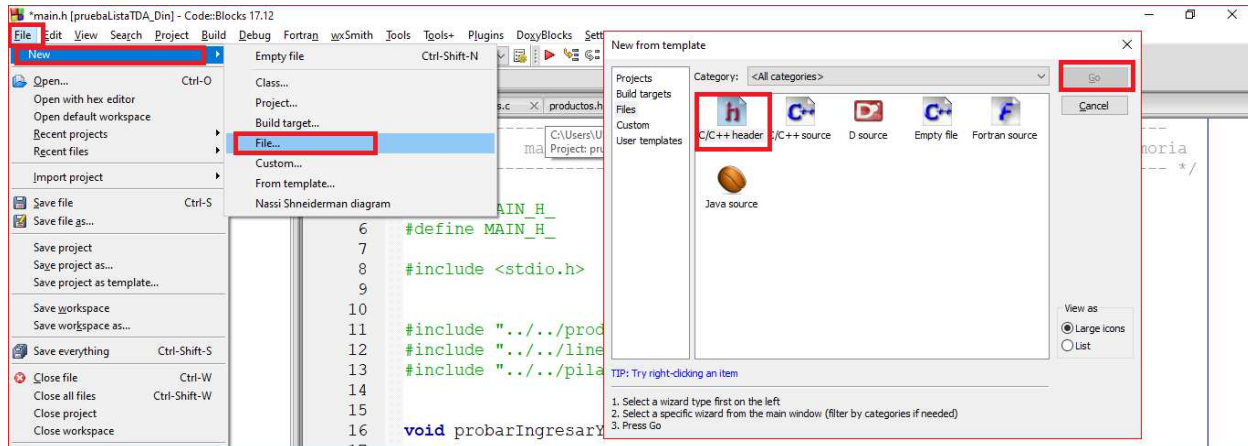


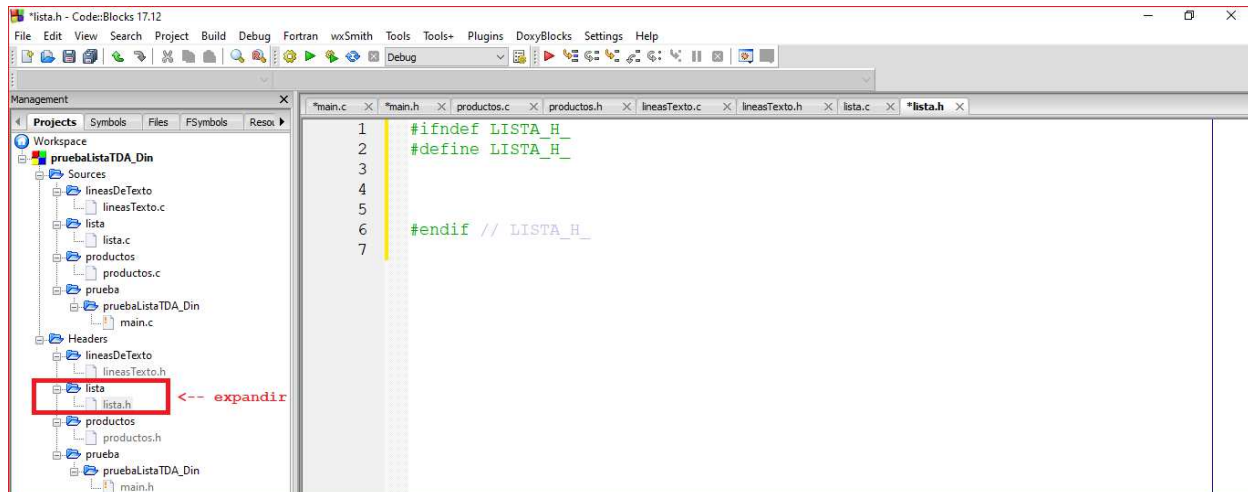
UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

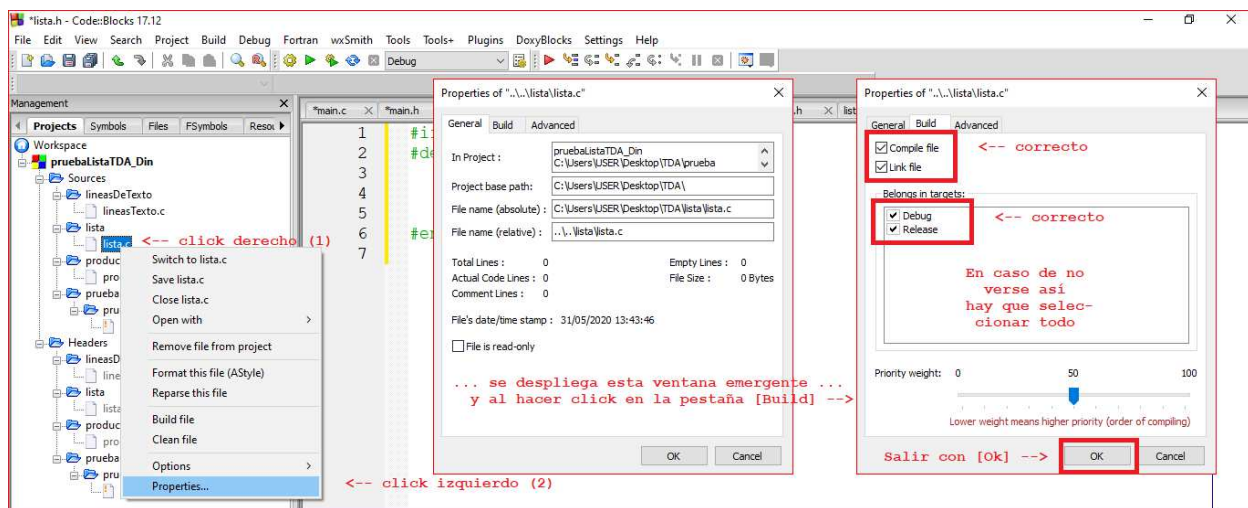


Agregar al proyecto el fuente "lista.h" ...





Como a "lista.c" se lo puso "de contrabando" en el proyecto . . .



. . . controlar que las "propiedades" del archivo para el IDE son las de un archivo que se va a compilar.



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

Completar los comentarios de "lista.h" además de declarar (algunas) las primitivas.

```

1  /* -----X----- */
2  /*      lista.h      declaración y primitivas del TDA LISTA
3  /*      implementada en lista dinámica simplemente enlazada
4  /* -----X----- */
5
6  #ifndef LISTA_H_
7  #define LISTA_H_
8
9
10 #include <stdlib.h>
11 #include <string.h>
12
13
14 #define SIN_MEM      1
15 #define CLA_DUP      2
16 #define TODO_BIEN    0
17
18
19 typedef struct sNodo
20 {
21     void      *info;
22     unsigned  tamInfo;
23     struct sNodo *sig;
24 } tNodo;
25 typedef tNodo *tLista;
26
27
28 void crearLista(tLista *p);
29
30 int  listaVacia(const tLista *p);
31
32 int  listaLlena(const tLista *p, unsigned cantBytes);
33
34 void vaciarLista(tLista *p);
35
36 int  ponerAlComienzo(tLista *p, const void *d, unsigned cantBytes);
37
38 int  sacarPrimeroLista(tLista *p, void *d, unsigned cantBytes);
39
40 int  verPrimeroLista(const tLista *p, void *d, unsigned cantBytes);
41
42 int  ponerAlFinal(tLista *p, const void *d, unsigned cantBytes);
43
44 int  sacarUltimoLista(tLista *p, void *d, unsigned cantBytes);
45
46 int  verUltimoLista(const tLista *p, void *d, unsigned cantBytes);
47
48
49 #endif // LISTA_H_
50
51

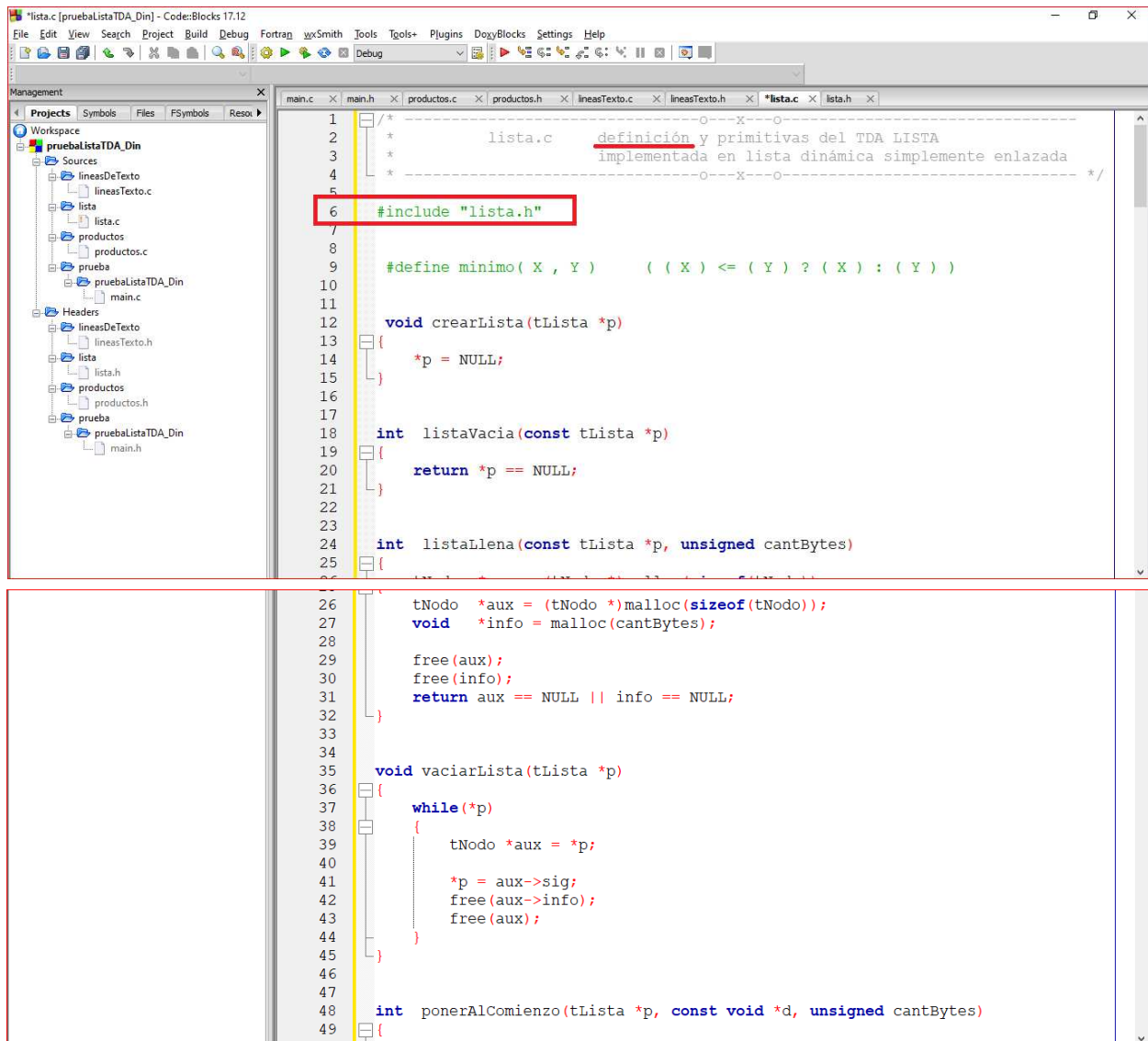
```



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

... y completar las definiciones en "lista.c" ...



```
1  /*-----X-----*/
2  *      lista.c  definición y primitivas del TDA LISTA
3  *      *
4  *      *      implementada en lista dinámica simplemente enlazada
5  *      *      *
6  #include "lista.h"
7
8
9  #define minimo( X , Y )      ( ( X ) <= ( Y ) ? ( X ) : ( Y ) )
10
11
12 void crearLista(tLista *p)
13 {
14     *p = NULL;
15 }
16
17
18 int listaVacia(const tLista *p)
19 {
20     return *p == NULL;
21 }
22
23
24 int listaLlena(const tLista *p, unsigned cantBytes)
25 {
26     tNodo *aux = (tNodo *)malloc(sizeof(tNodo));
27     void *info = malloc(cantBytes);
28
29     free(aux);
30     free(info);
31     return aux == NULL || info == NULL;
32 }
33
34
35 void vaciarLista(tLista *p)
36 {
37     while(*p)
38     {
39         tNodo *aux = *p;
40
41         *p = aux->sig;
42         free(aux->info);
43         free(aux);
44     }
45 }
46
47
48 int ponerAlComienzo(tLista *p, const void *d, unsigned cantBytes)
49 {
50     tNodo *aux;
```




UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

```

50     tNodo *nue;
51
52     if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
53        (nue->info = malloc(cantBytes)) == NULL)
54     {
55         free(nue);
56         return 0;
57     }
58     memcpy(nue->info, d, cantBytes);
59     nue->tamInfo = cantBytes;
60     nue->sig = *p;
61     *p = nue;
62     return 1;
63 }
64
65
66 int sacarPrimerLista(tLista *p, void *d, unsigned cantBytes)
67 {
68     tNodo *aux = *p;
69
70     if(aux == NULL)
71         return 0;
72     *p = aux->sig;
73     memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
74     free(aux->info);
75     free(aux->info);
76     free(aux);
77     return 1;
78 }
79
80 int verPrimerLista(const tLista *p, void *d, unsigned cantBytes)
81 {
82     if(*p == NULL)
83         return 0;
84     memcpy(d, (*p)->info, minimo(cantBytes, (*p)->tamInfo));
85     return 1;
86 }
87
88
89 int ponerAlFinal(tLista *p, const void *d, unsigned cantBytes)
90 {
91     tNodo *nue;
92
93     while(*p)
94         p = (*p)->sig;
95     if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
96        (nue->info = malloc(cantBytes)) == NULL)
97     {
98         free(nue);
99         return 0;
100     }
101     memcpy(nue->info, d, cantBytes);
102     nue->tamInfo = cantBytes;
103     nue->sig = NULL;
104     *p = nue;
105     return 1;
106 }
107
108
109 int sacarUltimoLista(tLista *p, void *d, unsigned cantBytes)
110 {
111     if(*p == NULL)
112         return 0;
113     while((*p)->sig)
114         p = (*p)->sig;
115     memcpy(d, (*p)->info, minimo(cantBytes, (*p)->tamInfo));
116     free((*p)->info);
117     free(*p);
118     *p = NULL;
119     return 1;
120 }
121

```



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

```

121
122
123 int verUltimoLista(const tLista *p, void *d, unsigned cantBytes)
124 {
125     if(*p == NULL)
126         return 0;
127     while((*p)->sig)
128         p = &(*p)->sig;
129     memcpy(d, (*p)->info, minimo(cantBytes, (*p)->tamInfo));
130     return 1;
131 }
132
133

```

Compilar (sólo generar el `.obj`, no el ejecutable) entrando por el menú desplegable **[Build] / [Compile current file]** (o pulsando **[Ctrl] + [Shift] + [F9]**) y comprobar que no se introdujo ningún error.

```

114     p = &(*p)->sig;
115     memcpy(d, (*p)->info, minimo(cantBytes, (*p)->tamInfo));
116     free((*p)->info);
117     free(*p);
118     *p = NULL;
119     return 1;
120 }
121
122
123 int verUltimoLista(const tLista *p, void *d, unsigned cantBytes)
124 {
125     if(*p == NULL)
126         return 0;
127     while((*p)->sig)
128         p = &(*p)->sig;
129     memcpy(d, (*p)->info, minimo(cantBytes, (*p)->tamInfo));
130     return 1;
131 }
132
133

```

Build file: Debug in pruebaListaTDA_Din (compiler: GNU GCC Compiler)

```

mingw32-gcc.exe -Wall -g -c C:\Users\USER\Desktop\TDA\lista\lista.c -o obj\Debug\lista\lista.o
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))

```



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

En "main.h" aún falta . . .

```

1  /*
2  *      main.h      prueba del TDA LISTA con asignación dinámica de memoria
3  *
4  *      -----
5
6  #ifndef MAIN_H
7  #define MAIN_H
8
9  #include <stdio.h>
10
11 #include "../productos/productos.h"
12 #include "../lineasDeTexto/lineasTexto.h"
13 #include "."
14 #include "../lineasDeTexto/lineasTexto.h"
15 #include "../lista/lista.h"
16 #include "../productos/productos.h"
17
18 void pro

```

. . . y en "main.c" hay que eliminar todo el código de prueba del TDA PILA y comenzar a probar algunas primitivas . . .

```

43  "===== ";
44  while(ingresarTexto(linea, sizeof(linea)))
45  {
46      cant++;
47      printf("\n%s\n", linea);
48  }
49  fprintf(stdout, "Se mostraron %d lineas de texto.\n", cant);
50  }
51
52  Se eliminó todo el código
53  de las funciones con que
54  se hicieron las pruebas
55  del TDA PILA.
56
57  void probarPonerYSacarDeLista(void)
58  {
59      tLista lista;
60
61      crearLista(&lista);
62      if(listaLlena(&lista, 1))
63          puts("Que pasa? No hay lugar ni para un Byte? No hay memoria???");
64      if(listaVacía(&lista))
65          puts("Obviamente que la lista está vacía! Recien creada!");
66      vaciarLista(&lista);
67  }

```

Se agregó una prueba mínima del uso del TDA LISTA.



Nuevamente compilar y verificar que no hay ningún error (¿recuerda el *atajo* [Ctrl] + [Shift] + [F9]?) ...

```
50 }
51
52
53
54
55 void probarPonerYSacarDeLista(void)
56 {
57     tLista lista;
58
59     crearLista(&lista);
60     if(listaLlena(&lista, 1))
61         puts("Que pasa? No hay lugar ni para un Byte? No hay memoria???");
62     if(listaVacia(&lista))
63         puts("Obviamente que la lista está vacía! Recien creada!");
64     vaciarLista(&lista);
65 }
66
67
```

Build log: Build file: Debug in pruebaListaTDA_Din (compiler: GNU GCC Compiler)

mingw32-gcc.exe -Wall -g -o C:\Users\USER2\Desktop\TDA\prueba\pruebaListaTDA_Din\main.o -c C:\Users\USER2\Desktop\TDA\prueba\pruebaListaTDA_Din\main.c

Process terminated with status 0 (0 minute(s), 0 second(s))

0 error(s), 0 warning(s) (0 minute(s), 0 second(s))

¡Manos a la obra!

Haga uso y pruebe el funcionamiento de las primitivas que ya tiene escritas . . .

La ejecución de lo hecho hasta ahora es . . .

```
C:\Users\USER\Desktop\TDA\prueba\pruebaListaTDA_Din\bin\Debug\pruebaListaTDA_Din.exe
Probando ingresar productos y mostrar productos.
=====
Cod. Produ Descripcion del producto ...
clavoro3/4 Clavo de oro 24 kilates de 3/4 de pulgada ...
martillo3K Martillo bolita con saca clavos de 3 kilos ...
alamyeso1 Alambre de yeso de un milimetro de espesor ...
rem-vid15 Remache de vidrio de 1,5 milímetros ...
plom-telgo Plomada de poliestireno expandido ...
limagoma17 Lima de goma de 17 pulgadas ...
Se mostraron 6 productos.

Probando ingresar lineas de texto mostrandolas.
=====
"Se necesita un amigo - Fragmento del Poema de Vinicius de Moraes"
""
"Debe tener un ideal, y miedo de perderlo,"
"y en caso de no ser asi,"
"debe sentir el gran vacio que esto deja."
"Tiene que tener resonancias humanas,"
"su principal objetivo debe ser el del amigo."
"Debe sentir pena por las personas tristes"
"y comprender el inmenso vacio de los solitarios."
"Se busca un amigo para gustar"
"de los mismos gustos,"
"que se conmueva cuando es tratado de amigo."
""
Se mostraron 13 lineas de texto.

Obviamente que la lista esta vacia! Recien creada!

Process returned 0 (0x0)   execution time : 0.022 s
```

No pierda más tiempo, póngase a practicar . . .