Anexo 1: Estructuración de código en librerías.

Hasta ahora, todas las funciones que teníamos en el código del programa, estaban declaradas en el mismo archivo .c. Sin embargo, ya podemos intuir que este hecho tiene que cambiar cuando nos enfrentamos a proyectos más y más grandes. En el proyecto de FO os proponemos usar librerías para estructurar y hacer más modular vuestro código.

La idea básica es muy sencilla: Se trata de agrupar en un fichero (la librería) todas las funciones propias que están relacionadas entre sí y utilizar distintos ficheros para cada agrupación. De esta manera el código principal podrá usar todas las funciones sin tenerlo todo en un único archivo excesivamente grande. Así el código quedará dividido en bloques y será más sencillo encontrar errores, ampliar el código, cambiar bloques, etc. La idea es que cada librería contenga funciones más o menos relacionadas entre sí (p.e. todas las funciones que operan sobre una misma estructura de datos) para hacer más sencilla la comprensión del código. Para entender cómo se estructura el código, veremos un ejemplo.

Ejemplo. Supongamos que tenemos el código de un programa que suma y multiplica matrices de números complejos. El código que tenemos escrito es el siguiente:

```
#include<stdio.h>

#define MAX 100

typedef struct
{
     float preal;   /* parte real */
     float pimg;    /* parte imaginaria */

} tcomplejo;

typedef struct
{
      int fil, col;
      tcomplejo matriu[MAX][MAX];
} tmatriz;

void lee_complejo(tcomplejo *c) /*Lectura de un complejo*/
{/*Código de la función*/
}

void escribe_complejo(tcomplejo c) /*Salida por pantalla del complejo
c*/
{/*Código de la función*/
}

tcomplejo suma_complejo(tcomplejo c1, tcomplejo c2)  /*Suma: c1+c2*/
{/*Código de la función*/
}

tcomplejo mul_complejo(tcomplejo c1, tcomplejo c2) /*Multiplica:
c1*c2*/
{/*Código de la función*/
}

void lee_matriz(tmatriz *mat) /*Lectura de la matriz mat*/
{/*Código de la función*/
 /*Usa lee_complejo*/
}
```

```
void escribe_matriz(tmatriz mat) /*Salida por pantalla de la matriz
mat*/
{/*Código de la función*/
 /*Usa escribe_complejo*/
}

void suma_matriz(tmatriz a, tmatriz b, tmatriz *c) /*Suma c=a+b*/
{/*Código de la función*/
 /*Usa suma_complejo*/
}

void multiplica_matriz(tmatriz a, tmatriz b, tmatriz *c)/*Multiplica
c=a*b*/
{/*Código de la función*/
 /*Usa suma_complejo y mul_complejo*/
}

int main()
{
    tmatriz a,b,c;

    printf("Introduce matriz A:\n");
    lee_matriz(&a);
    printf("Introduce matriz B:\n");
    lee_matriz(&b);
    printf("Suma A+B:\n");
    suma_matriz(a,b,&c);
    escribe_matriz(c);
    printf("Mutiplicacion A*B:\n");
    multiplica_matriz(a,b,&c);
    escribe_matriz(c);
}
```

Nuestra intención ahora será agrupar en archivos distintos los tipos de datos y las funciones relacionadas con los complejos y las relacionadas con matrices.

Por un lado tenemos el tipo de dato **tcomplejo** y las funciones **lee_complejo, escribe_complejo**, **suma_complejo** y **mul_complejo,** todas relacionadas con la manipulación de la estructura **tcomplejo** y, por otro lado, tenemos el tipo de dato **tmatriz** y las funciones **lee_matriz**, **escribe_matriz**, **suma_matriz** y **multiplica_matriz** que están relacionadas con la manipulación de la estructura **tmatriz**. Además, tenemos el programa principal que usa estas funciones.

Para estructurar este código en librerías necesitamos 5 archivos distintos:

- *main.c*: contiene la función **main** del programa, y es el bloque principal del código. En nuestro ejemplo sería:

```
#include<stdio.h>
#include"matriz.h"

int main()
{
    tmatriz a,b,c;

    printf("Introduce matriz A:\n");
    lee_matriz(&a);
    printf("Introduce matriz B:\n");
    lee_matriz(&b);
    printf("Suma A+B:\n");
    suma_matriz(a,b,&c);
    escribe_matriz(c);
```

```
    printf("Mutiplicacion A*B:\n");
    multiplica_matriz(a,b,&c);
    escribe_matriz(c);
}
```

- *complejo.h*: es el archivo que contiene los tipos de datos y los prototipos (cabeceras) de todas las funciones relacionadas con la manipulación de los complejos:

```
#ifndef COMPLEJO_H
#define COMPLEJO_H

typedef struct
{
     float preal;   /* parte real */
     float pimg;    /* parte imaginaria */
} tcomplejo;

void lee_complejo(tcomplejo *c);
void escribe_complejo (tcomplejo c) ;
tcomplejo suma_complejo(tcomplejo c1, tcomplejo c2);
tcomplejo mul_complejo(tcomplejo c1, tcomplejo c2);

#endif
```

donde `#ifndef` y `#endif` representan directivas de compilación condicional que permiten incluir o descartar parte del código de un programa si se cumple una determinada condición (macro definida como parámetro). Estas directivas aseguran que el fichero se incluye una única vez en el código fuente, aunque sea incluido como librería en más de un fichero.

- *complejo.c*: archivo que contiene todas las definiciones de las funciones relacionadas con los complejos:

```
#include<stdio.h>
#include "complejo.h"

void lee_complejo(tcomplejo *c)       /*Lectura de un complejo*/
{
     /*Código de la función*/
}

void escribe_complejo(tcomplejo c)     /*Salida por pantalla del c*/
{
     /*Código de la función*/
}

tcomplejo suma_complejo(tcomplejo c1, tcomplejo c2)  /*Suma: c1+c2*/
{
     /*Código de la función*/
}

tcomplejo mul_complejo(tcomplejo c1, tcomplejo c2)/*Multiplica:c1*c2*/
{
     /*Código de la función*/
}
```

- *matriz.h*: es el archivo que contiene los tipos de datos y los prototipos (cabeceras) de todas las funciones relacionadas con matrices:

```
#ifndef MATRIZ_H
#define MATRIZ_H

#include "complejo.h"
#define MAX 100

typedef struct
{
     int fil, col;
     tcomplejo matriu[MAX][MAX];
} tmatriz;

void lee_matriz(tmatriz *mat);
void escribe_matriz(tmatriz mat);
void suma_matriz(tmatriz a, tmatriz b, tmatriz *c);
void multiplica_matriz(tmatriz a, tmatriz b, tmatriz *c);


#endif
```

- *matriz.c*: es el archivo que contiene todas las definiciones de funciones relacionadas con matrices:

```
#include<stdio.h>
#include "matriz.h"

void lee_matriz(tmatriz *mat)   /*Lectura mat*/
{
     /*Código de la función*/
}

void escribe_matriz(tmatriz mat)
/*Salida por pantalla de la matriz mat*/
{
     /*Código de la función*/
}

void suma_matriz(tmatriz a, tmatriz b, tmatriz *c)   /*Suma c=a+b*/
{
     /*Código de la función*/
}

void multiplica_matriz(tmatriz a, tmatriz b, tmatriz *c)
/*Multiplica c = a*b*/
{
     /*Código de la función*/
}
```

Finalmente, lo único que queda es compilar todo el código con el siguiente comando:

gcc* –g** *main.c complejo.c matriz.c* ***-o *ejecutable_matrices*