

Bueno hasta ahora llegamos hasta aquí, este es el código fuente que nos queda hasta ahora:

```
#include <stdio.h>
#include <string.h>

main(){
    funcion();
}

funcion(){
    int eleccion=0;
    int numero_de_fichas=0;
    char texto_tipeado[40];
    struct{
        char nombrefich[44];    /* Nombre del fichero */
        unsigned long tamaño; /* El tamaño en bytes */
    } fichas[1000];

    while (eleccion!=5)
    {
        printf("Escoja una opción:\n");
        printf("1.- Añadir datos de un nuevo fichero\n");
        printf("2.- Mostrar los nombres de todos los ficheros\n");
        printf("3.- Mostrar ficheros que sean de mas de un cierto tamaño\n");
        printf("4.- Ver datos de un fichero\n");
        printf("5.- Salir\n");

        gets(texto_tipeado);
        sscanf(texto_tipeado, "%d", &eleccion);
        switch(eleccion){
            case 1:

                if (numero_de_fichas < 1000) {
                    printf("Introduce el nombre del fichero: ");

                    numero_de_fichas++;
                } else
                    printf("Máximo de fichas alcanzado (1000)!\n");
                break;
            case 2:
                printf("Tipeaste 2\n ");
                break;
            case 3:
                printf("Tipeaste 3\n ");
                break;
            case 4:
                printf("Tipeaste 4\n ");
                break;
            case 5:
                printf("Tipeaste 5\n ");
                break;
        }
    }
}
```

```

    default:
        printf("Cualquier huevada!\n");
        break;
    }
}
}

```

Le agregamos la inicializacion que faltaba a la variable **numero_de_fichas**, la ponemos a cero al inicio.

Vemos el código que sigue a continuacion:

```

lea    ecx, [ebp+fichas]
mov     edx, [ebp+numero_de_fichas]
mov     eax, edx
add     eax, eax
add     eax, edx
shl     eax, 4
lea     eax, [ecx+eax]
mov     [esp], eax    ; char *
call    gets

```

Recordemos que **ficha[0].nombrefich** estaba en el primer lugar del array, que ocupaba **44 bytes**.

```

00000000 ,
00000000
00000000 struc_1      struc ; (sizeof=0x30)
00000000 nombrefich    db 44 dup(?)          ; string(C)
00000020 tamaño       dd ?
00000030 struc_1      ends
00000030

```

Si reemplazamos **numero_de_fichas** por **0**.

```

lea     ecx, [ebp+fichas]
mov     edx, [ebp+0]
mov     eax, edx    #EDX vale cero, EAX vale cero
add     eax, eax    #EAX vale cero
add     eax, edx    #EAX vale cero
shl     eax, 4      #EAX vale cero
lea     eax, [ecx+eax]    #EAX vale 0
mov     [esp], eax  ; char *
call    gets

```

Por lo tanto valdrá

`lea eax, [ecx+eax]` es igual a `lea eax, [ecx+0]` y apuntara al primer campo **nombrefich**.

Así que esta parte del código maneja el índice **numero_de_fichas**, para recorrer los diferentes campos **nombrefich** vemos que si **numero_de_fichas =1** , por lo tanto **ficha[1].nombrefich** esta 48 bytes mas adelante, ya que el primer **nombrefich** ocupaba **44** y el dword del **tamaño** 4 bytes mas.

Si reemplazamos **numero_de_fichas** por 1, vemos que el valor final de EAX es 48, así que se sumara a la dirección de la variable **fichas** que esta en ECX, obteniendo la nueva dirección del segundo campo **nombrefich**.

```
lea ecx, [ebp+fichas]
mov  edx, [ebp+1]
mov  eax, edx    #EDX vale uno, EAX vale uno
add  eax, eax    #EAX vale dos
add  eax, edx    #EAX vale tres
shl  eax, 4      # 3 * 16 = 48
lea  eax, [ecx+eax]    #EAX vale 48 y al sumarlo a ECX apuntara al 2do nombrefich
mov  [esp], eax  ; char *
call gets
```

Así que a `gets` según los valores que toma **numero_de_fichas** siempre se le pasara como argumento, el correspondiente campo **nombrefich** para llenarlo al tipear, el código anterior no es necesario transcribirlo pues es solo la conversión del índice **numero_de_fichas** del código fuente al índice para saltar de a 48 bytes en la memoria para hallar el siguiente campo similar.

numero_de_fichas =0 en el código fuente sera **0** (campo **nombrefich 0**)
numero_de_fichas =1 en el código fuente sera **48** (campo **nombrefich 1**)
numero_de_fichas =2 en el código fuente sera **92** (campo **nombrefich 2**)

y así sucesivamente por lo tanto en el código fuente solo deberemos agregar

```
gets(fichas[numero_de_fichas].nombrefich);
```

completemos el case 1

case 1:

```
if (numero_de_fichas < 1000) {
    printf("Introduce el nombre del fichero: ");
    gets(fichas[numero_de_fichas].nombrefich);

    numero_de_fichas++;
} else
    printf("Máximo de fichas alcanzado (1000)!\n");
break;
```

Luego viene un `printf` que nos muestra el mensaje que **Introduzcamos el tamaño**, lo agregamos.

```

mov     dword ptr [esp], offset aIntroduceElTam ; "Introduce el tama"
call    printf

```

case 1:

```

if (numero_de_fichas < 1000) {
    printf("Introduce el nombre del fichero: ");
    gets(fichas[numero_de_fichas].nombrefich);
    printf("Introduce el tamaño en KB: ");
    numero_de_fichas++;
} else
    printf("Máximo de fichas alcanzado (1000)!\n");
break;

```

Luego viene el código para introducir el campo **tamaño** que de la misma forma lo filtra primero con **gets** y luego usa **sscanf** para darle formato, usa la misma variable **texto_tipeado** para guardar lo que tipeamos.

```

lea     eax, [ebp+texto_tipeado]
mov     [esp], eax ; char *
call    gets

```

Así que esto puede traducirse al código fuente como:

case 1:

```

if (numero_de_fichas < 1000) {
    printf("Introduce el nombre del fichero: ");
    gets(fichas[numero_de_fichas].nombrefich);
    printf("Introduce el tamaño en KB: ");
    gets(texto_tipeado);

    numero_de_fichas++;
} else
    printf("Máximo de fichas alcanzado (1000)!\n");
break;

```

Luego bien este código, la parte resaltada es similar a la forma como maneja el índice de 48 el 48 solo que ahora le suma 2ch, o sea que en este caso sera el índice que barre los campos **tamaño**.

```

lea     ecx, [ebp+fichas]
mov     edx, [ebp+numero_de_fichas]
mov     eax, edx
add     eax, eax
add     eax, edx
shl     eax, 4
lea     eax, [ecx+eax]

```

```
add    eax, 2Ch
```

En este caso el primer campo **tamaño** se encuentra a partir del byte 44, así que cuando el numero de fichas vale cero, **EAX** sale valiendo **cero** como antes pero al final le suma 2ch o 44 decimal lo cual lo lleva a la dirección del primer campo tamaño.

numero_de_fichas = 0 en el código fuente sera **44** (campo tamaño 0)

numero_de_fichas = 1 en el código fuente sera **92** (campo tamaño 1)

numero_de_fichas = 2 en el código fuente sera 136 (campo tamaño 2)

```
add    eax, 2Ch
mov     [esp+8], eax
mov     dword ptr [esp+4], offset aLd ; "%ld"
lea     eax, [ebp+texto_tipeado]
mov     [esp], eax ; char *
call    scanf
```

Entonces ya tenemos los argumentos para scanf el puntero al texto_tipeado anteriormente, el formato **%ld** y la dirección usando **&** ya que no es una api de string, del campo tamaño.

```
scanf(texto_tipeado, "%ld", &fichas[numero_de_fichas].tamano);
```

Luego de eso se incrementa la variable **numero_de_fichas** y se sale del case 1 yendo a la comparación de si **eleccion** es 5 para salir sino vuelve a looppear.

```
lea     eax, [ebp+texto_tipeado]
mov     [esp], eax ; char *
call    scanf
lea     eax, [ebp+numero_de_fichas]
inc     dword ptr [eax]
jmp     loc_401626
```

```
loc_401626:
cmp     [ebp+eleccion], 5
jz      short loc_401634
```

Vemos que la opción 1 ya funciona como corresponde, va pidiendo y almacenando los datos.

```
C:\Documents and Settings\ricnar\Escritorio\1315-C Y REVERSING (parte
Escoja una opción:
1.- Añadir datos de un nuevo fichero
2.- Mostrar los nombres de todos los ficheros
3.- Mostrar ficheros que sean de mas de un cierto tamaño
4.- Ver datos de un fichero
5.- Salir
1
Introduce el nombre del fichero: pepe
Introduce el tamaño en KB: 3
Escoja una opción:
1.- Añadir datos de un nuevo fichero
2.- Mostrar los nombres de todos los ficheros
3.- Mostrar ficheros que sean de mas de un cierto tamaño
4.- Ver datos de un fichero
5.- Salir
1
Introduce el nombre del fichero: jose
Introduce el tamaño en KB: 5
Escoja una opción:
1.- Añadir datos de un nuevo fichero
2.- Mostrar los nombres de todos los ficheros
3.- Mostrar ficheros que sean de mas de un cierto tamaño
4.- Ver datos de un fichero
5.- Salir
```

El código completo hasta ahora es así:

```
#include <stdio.h>
#include <string.h>

main(){
    funcion();
}

funcion(){
    int eleccion=0;
    int numero_de_fichas=0;
    char texto_tipeado[40];
    struct{
        char nombrefich[44]; /* Nombre del fichero */
        unsigned long tamaño; /* El tamaño en bytes */
    } fichas[1000];

    while (eleccion!=5)
    {
        printf("Escoja una opción:\n");
        printf("1.- Añadir datos de un nuevo fichero\n");
        printf("2.- Mostrar los nombres de todos los ficheros\n");
        printf("3.- Mostrar ficheros que sean de mas de un cierto tamaño\n");
        printf("4.- Ver datos de un fichero\n");
        printf("5.- Salir\n");

        gets (texto_tipeado);
        sscanf(texto_tipeado, "%d", &eleccion);
        switch(eleccion){
            case 1:
```

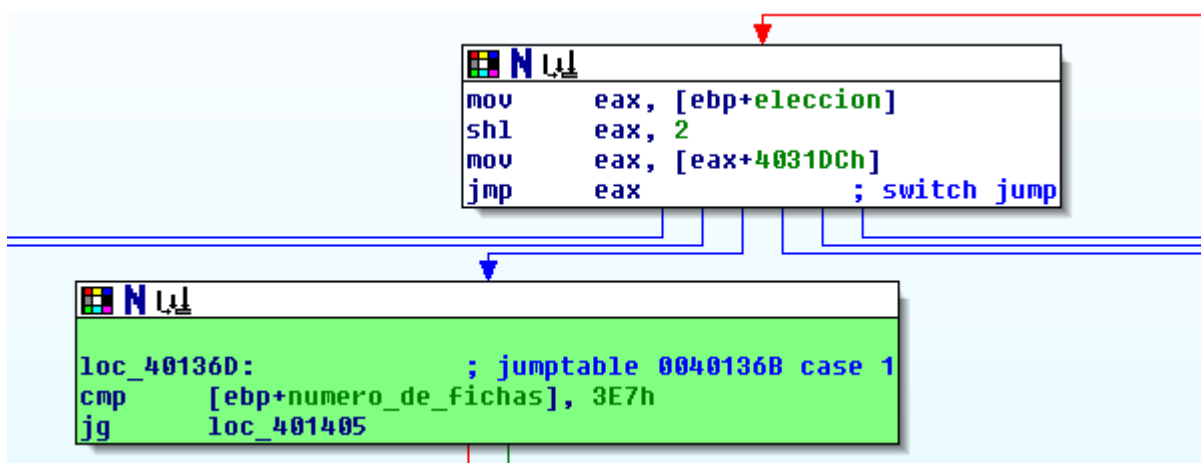
```

if (numero_de_fichas < 1000) {
    printf("Introduce el nombre del fichero: ");
    gets(fichas[numero_de_fichas].nombrefich);
    printf("Introduce el tamaño en KB: ");
    gets(texto_tipeado);
    sscanf(texto_tipeado,"%ld",&fichas[numero_de_fichas].tamanio);

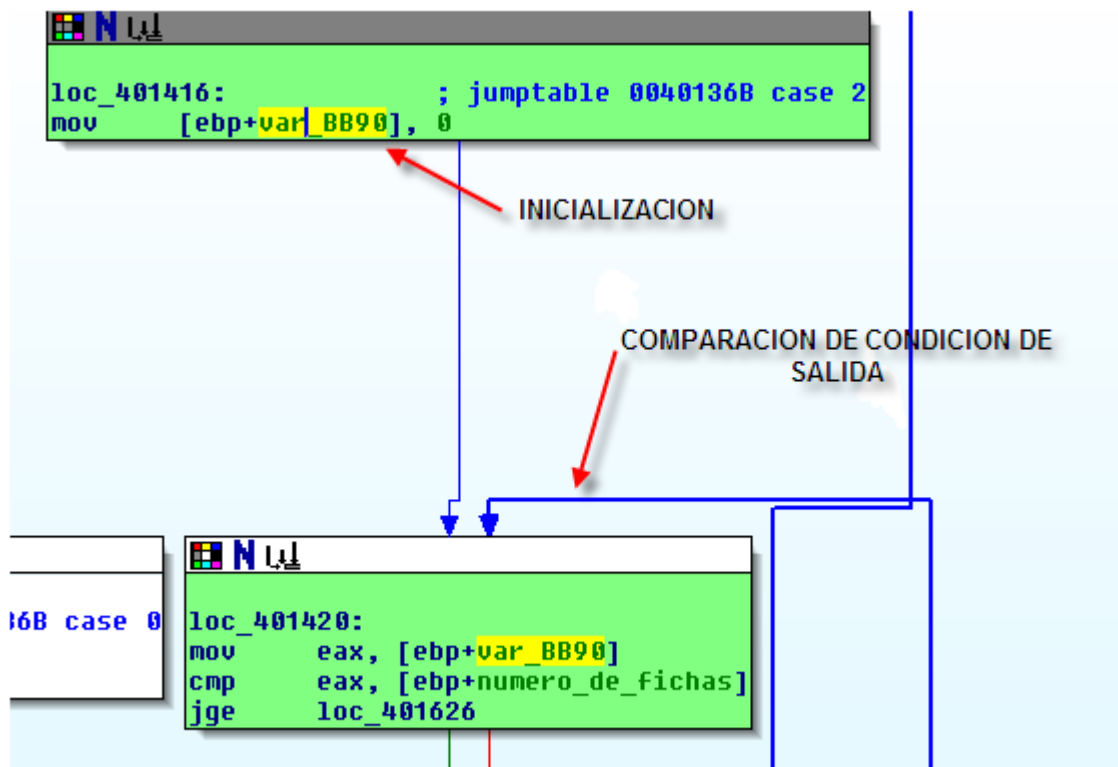
    numero_de_fichas++;
} else
    printf("Máximo de fichas alcanzado (1000)!\n");
    break;
case 2:
    printf("Tipeaste 2\n ");
    break;
case 3:
    printf("Tipeaste 3\n ");
    break;
case 4:
    printf("Tipeaste 4\n ");
    break;
case 5:
    printf("Tipeaste 5\n ");
    break;
default:
    printf("Cualquier huevada!\n");
    break;
}
}
}

```

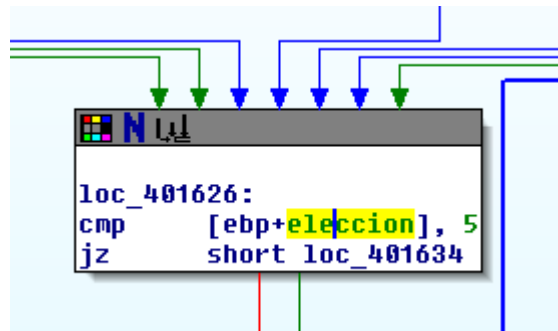
Para no repetir al análisis volvemos al switch y allí donde dice **switch jump** buscamos la flecha de las 6 salidas que va al **case 2**.



Vemos que el case 2 inicializa una nueva variable a cero, y luego hay un loop, como tenemos una inicialización inicial, un loop, una comparación de esa misma variable para salir del mismo, y un incremento de la misma.



Aquí vemos la salida si esa variable es mayor o igual que el numero de fichas actual, va por el camino de la flecha verde a la salida del case.



También dentro del ciclo la misma variable se va incrementando aquí.

```

mov     edx, [ebp+var_BB90]
mov     eax, edx
add     eax, eax
add     eax, edx
shl     eax, 4
lea     edx, [ebp+var_8]
add     eax, edx
sub     eax, 0BB60h
mov     eax, [eax+0Ch]
mov     [esp+8], eax
lea     ecx, [ebp+fichas]
mov     edx, [ebp+var_BB90]
mov     eax, edx
add     eax, eax
add     eax, edx
shl     eax, 4
lea     eax, [ecx+eax]
mov     [esp+4], eax
mov     dword ptr [esp], offset aNombreSTama ; "Nombre: %s; Tama"
call    printf
lea     eax, [ebp+var_BB90]
inc     dword ptr [eax]
jmp     short loc_401420

```

INCREMENTO

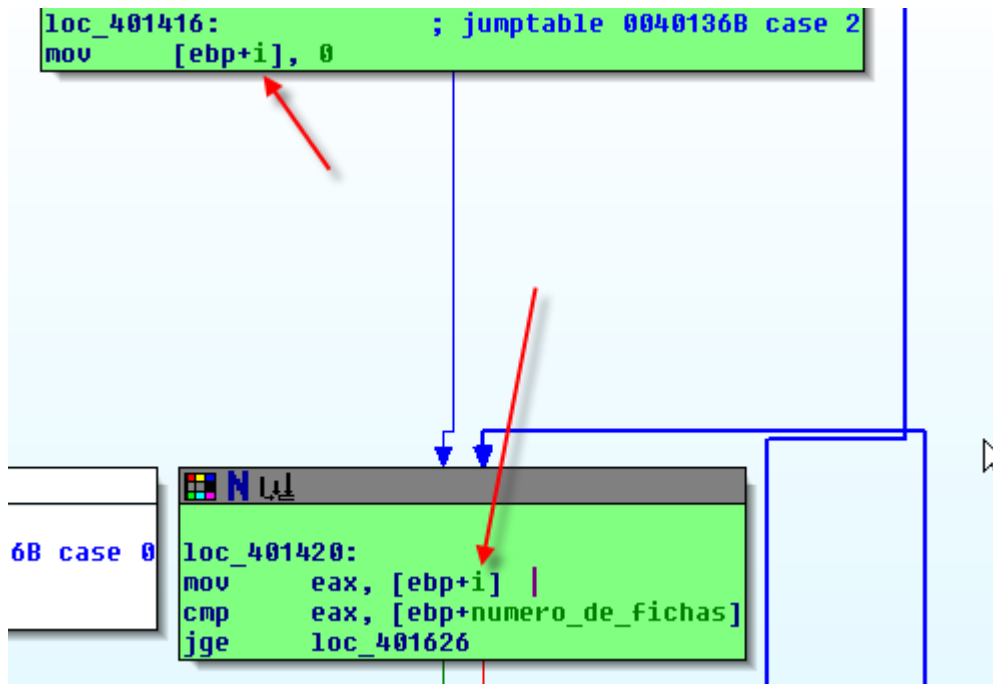
Así que como aquí en el case 2 el mismo texto nos dice que va a imprimir todos los campos nombres y tamaño, pues esta es una variable temporal para dentro de un for recorrerlos a todos desde cero hasta el valor máximo de fichas actual.

case 2:

for (i=0; i<numero_de_fichas; i++)

break;

Así que allí tenemos el for por lo tanto a la variable que será el contador la llamamos **i**, la renombramos.



Si vamos al print que esta en el siguiente bloque dentro del for, vemos que la segunda parte es similar a como había manejado el indice en el case 1, pivotando con la variable i, va recorriendo desde 0 saltando de 48 en 48, para obtener los campos **nombrefich**.

```
lea    ecx, [ebp+fichas]
mov    edx, [ebp+i]
mov    eax, edx
add    eax, eax
add    eax, edx
shl    eax, 4
lea    eax, [ecx+eax]
mov    [esp+4], eax
```

Como antes:

i=0 en el código fuente sera **0** (campo nombrefich 0)
i=1 en el código fuente sera **48** (campo nombrefich 1)
i=2 en el código fuente sera **92** (campo nombrefich 2)

ahora para hallar el tamaño esta vez en vez de usar un indice similar y sumarle 2ch como antes como el compilador nos quiere complicar la vida lo hace en forma diferente.

```

mov     edx, [ebp+i]
mov     eax, edx
add     eax, eax
add     eax, edx
shl     eax, 4
lea     edx, [ebp+var_8]
add     eax, edx
sub     eax, 0BB60h
mov     eax, [eax+0Ch]
mov     [esp+8], eax

```

Aquí vemos una **var_8** que esta justo a continuacion del array pegada al final del mismo y de allí hace una resta, veamos.

```

-0000BBCE      db ? ; undefined
-0000BBCD      db ? ; undefined
-0000BBCC      var_BBCC      dd ?
-0000BBC8      texto_tipeado  db 52 dup(?)
-0000BB94      eleccion      dd ?
-0000BB90      i              dd ?
-0000BB8C      numero_de_fichas dd ?
-0000BB88      fichas        struc_1 1000 dup(?)
-00000008      var_8         db ?
-00000007      db ? ; undefined
-00000006      db ? ; undefined

```

```

mov     edx, [ebp+i]
mov     eax, edx
add     eax, eax
add     eax, edx
shl     eax, 4
lea     edx, [ebp+var_8]
add     eax, edx
sub     eax, 0BB60h
mov     eax, [eax+0Ch]
mov     [esp+8], eax

```

la parte resaltada es igual que antes de 0, 48, etc pero esta vez en vez de sumarle 2c toma la variable **var_8** y le resta BB60, que debe ser lo mismo que sumarle 2c desde el inicio jeje

```

-0000BB88 fichas      struc_1 1000 dup(?)
-00000008 var_8      db ?

```

Como el array esta en el offset BB88 y la **var_8** en el offset 8 quiere decir que el largo del array es BB80 hexa, si a eso le resto desde el final BB60 sera igual a 20h al cual le suma 0ch a continuacion

```

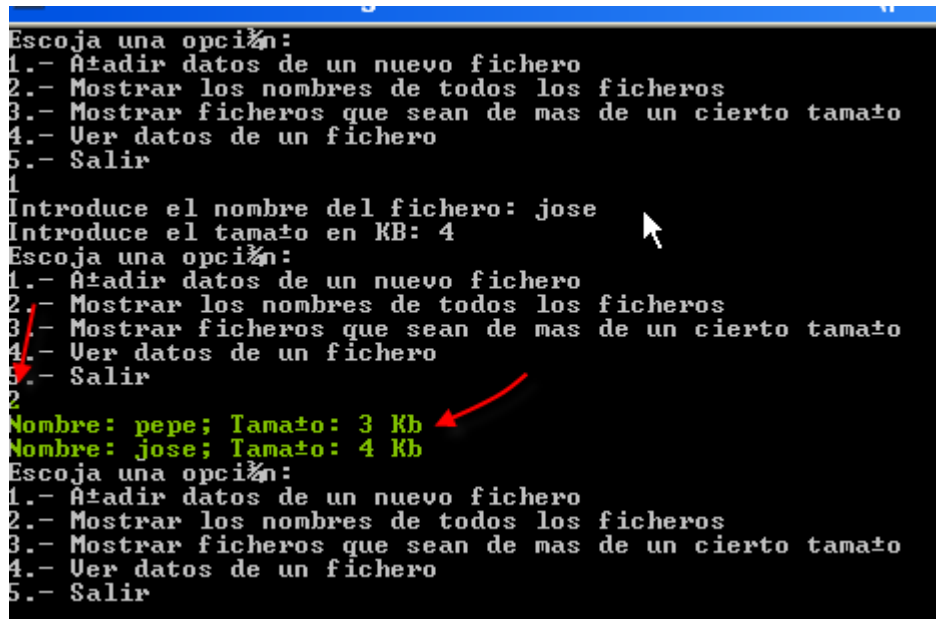
mov     eax, [eax+0Ch]
mov     [esp+8], eax

```

con lo cual es lo mismo que sumar **2ch**

así que recorrerá todos los campos **nombrefich** y **tamania** y los imprimira con printf

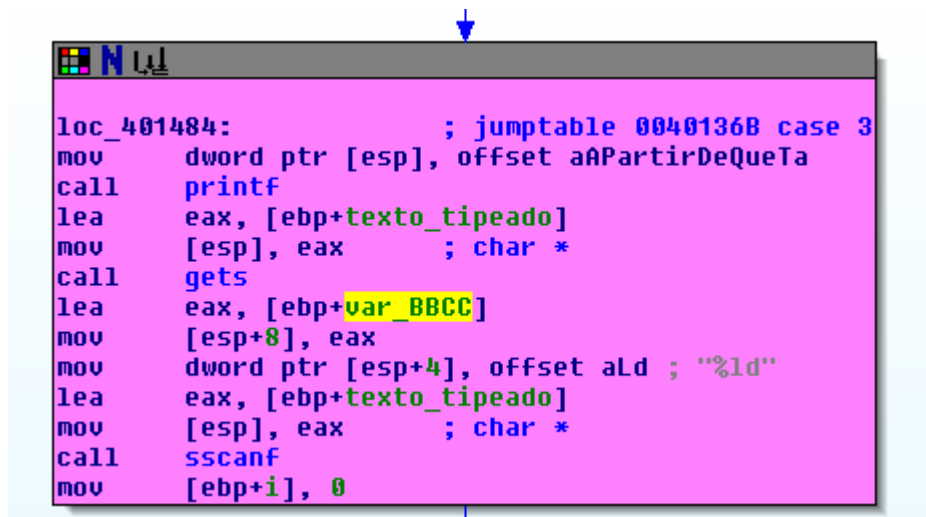
```
printf("Nombre: %s; Tamaño: %ld Kb\n",
      fichas[i].nombrefich , fichas[i].tamanio);
```



```
Escoja una opción:
1.- Añadir datos de un nuevo fichero
2.- Mostrar los nombres de todos los ficheros
3.- Mostrar ficheros que sean de mas de un cierto tamaño
4.- Ver datos de un fichero
5.- Salir
1
Introduce el nombre del fichero: jose
Introduce el tamaño en KB: 4
Escoja una opción:
1.- Añadir datos de un nuevo fichero
2.- Mostrar los nombres de todos los ficheros
3.- Mostrar ficheros que sean de mas de un cierto tamaño
4.- Ver datos de un fichero
5.- Salir
2
Nombre: pepe; Tamaño: 3 Kb
Nombre: jose; Tamaño: 4 Kb
Escoja una opción:
1.- Añadir datos de un nuevo fichero
2.- Mostrar los nombres de todos los ficheros
3.- Mostrar ficheros que sean de mas de un cierto tamaño
4.- Ver datos de un fichero
5.- Salir
```

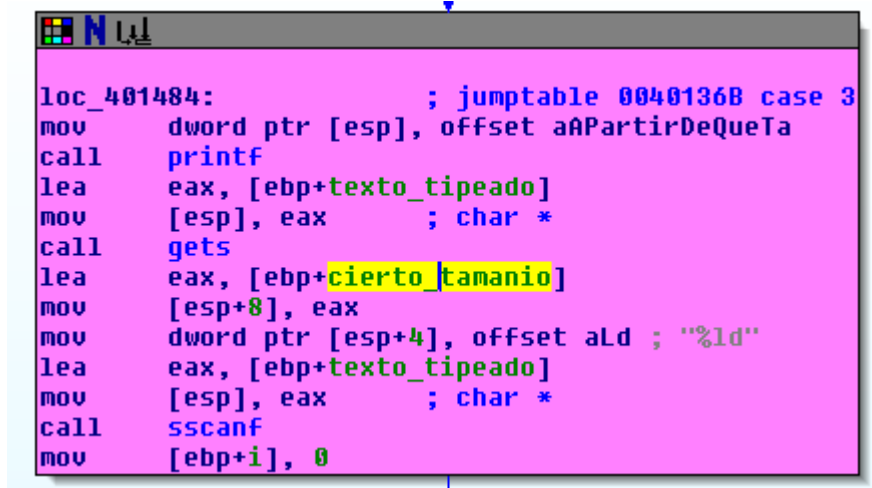
Vemos que la opción 2 también funciona bien imprimiendo los nombres y tamaños que hayamos tipeado en la opción 1.

El caso 3 nos dice que mostrara los nombres de los ficheros que sean mas de un cierto tamaño.



```
loc_401484:                ; jumtable 0040136B case 3
mov     dword ptr [esp], offset aAPartirDeQueTa
call    printf
lea     eax, [ebp+texto_tipeado]
mov     [esp], eax          ; char *
call    gets
lea     eax, [ebp+var_BBCC]
mov     [esp+8], eax
mov     dword ptr [esp+4], offset aLd ; "%ld"
lea     eax, [ebp+texto_tipeado]
mov     [esp], eax          ; char *
call    sscanf
mov     [ebp+i], 0
```

Vemos que el análisis es similar para la entrada usa **gets** y almacena la string tipeada en la variable **texto_tipeado** y luego filtra la misma usando **sscanf** y **%ld**, y guarda el resultado en una variable llamada **var_BBCC** que la renombraremos a **cierto_tamanio**



```
loc_401484:                ; jumtable 00401368 case 3
mov     dword ptr [esp], offset aAPartirDeQueTa
call    printf
lea     eax, [ebp+texto_tipeado]
mov     [esp], eax          ; char *
call    gets
lea     eax, [ebp+cierto_tamano]
mov     [esp+8], eax
mov     dword ptr [esp+4], offset aLd ; "%ld"
lea     eax, [ebp+texto_tipeado]
mov     [esp], eax          ; char *
call    sscanf
mov     [ebp+i], 0
```

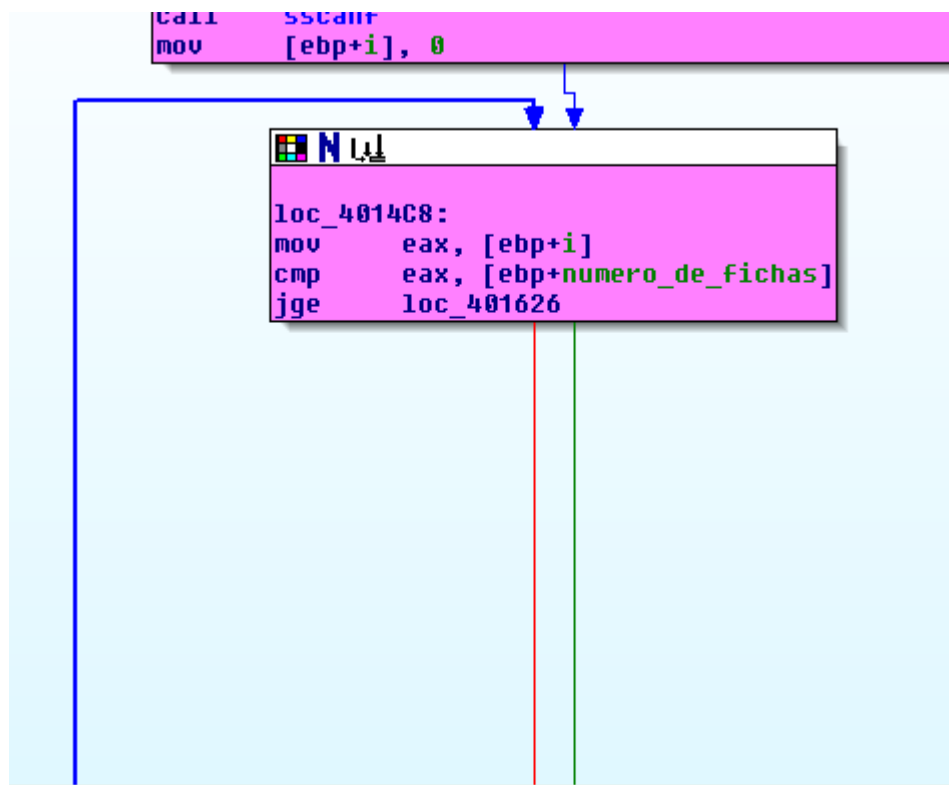
Todo ello corresponde al código fuente siguiente.

case 3:

```
printf("¿A partir de que tamaño quieres que te muestre?");
gets(texto_tipeado);
sscanf(texto_tipeado, "%ld", &cierto_tamano);
```

```
break;
```

La variable **cierto_tamano** sera un **unsigned long** al igual que tamaño, así que la declaramos al inicio.



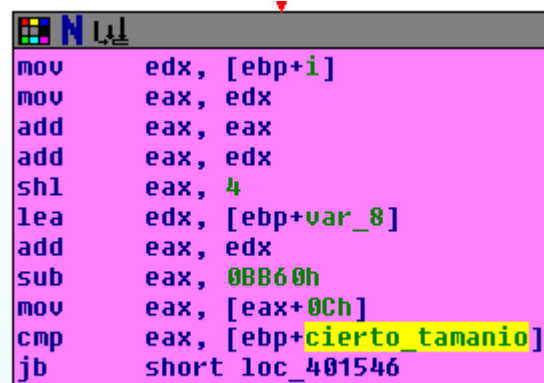
Luego usando la misma variable i como contador, se realiza un for, donde i se inicializa a cero, se compara con el numero de fichas actual y se incrementa a la salida tal cual el case anterior.

case 3:

```
printf("¿A partir de que tamaño quieres que te muestre?");  
gets(texto_tipeado);  
sscanf(texto_tipeado, "%ld", &cierto_tamano);  
for (i=0; i<numero_de_fichas; i++)
```

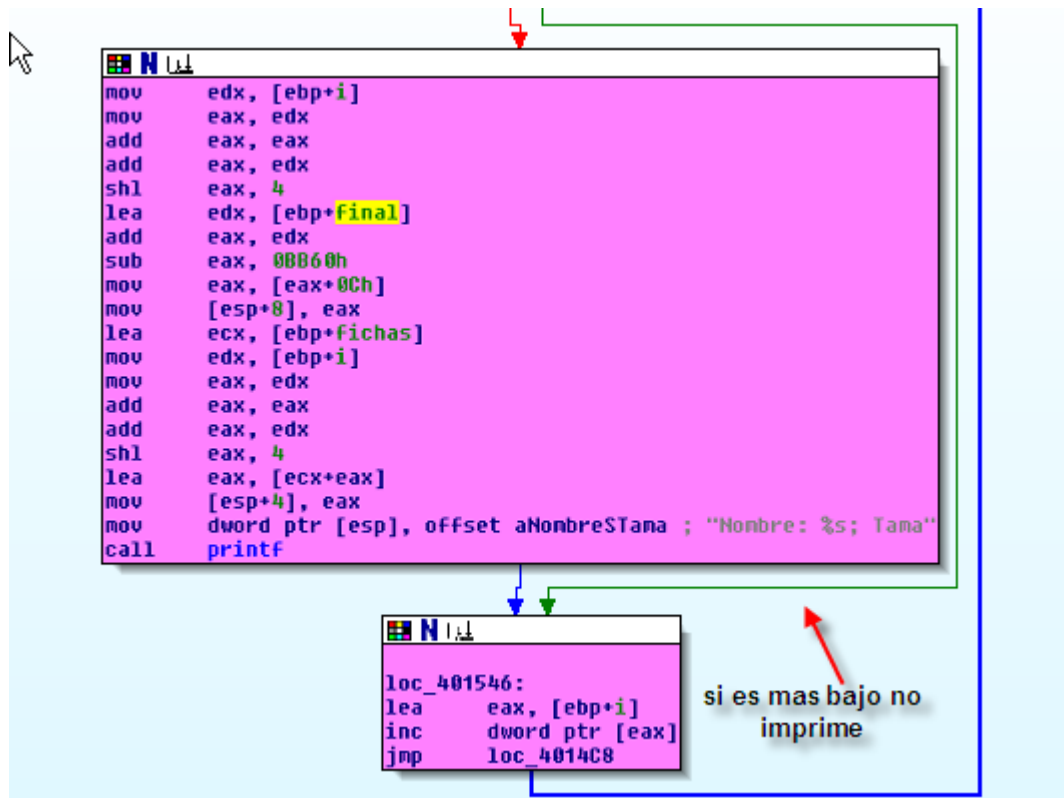
```
break;
```

Veamos que hace dentro del for



```
mov     edx, [ebp+i]  
mov     eax, edx  
add     eax, eax  
add     eax, edx  
shl     eax, 4  
lea     edx, [ebp+var_8]  
add     eax, edx  
sub     eax, 0BB60h  
mov     eax, [eax+0Ch]  
cmp     eax, [ebp+cierto_tamano]  
jb      short loc_401546
```

Al igual que en el caso 2, usa la variable var_8 para hallar el campo tamaño restando desde el final del array, renombraremos **var_8** como **final**, luego de obtener el tamaño lo compara con la variable **cierto_tamano** y si es mas baja que esa sigue la flecha verde evitando ir a imprimir con print f los valores, o sea que el case 3 es igual al case 2 solo que imprime los que son mas grandes que el valor ingresado en **cierto_tamano**.



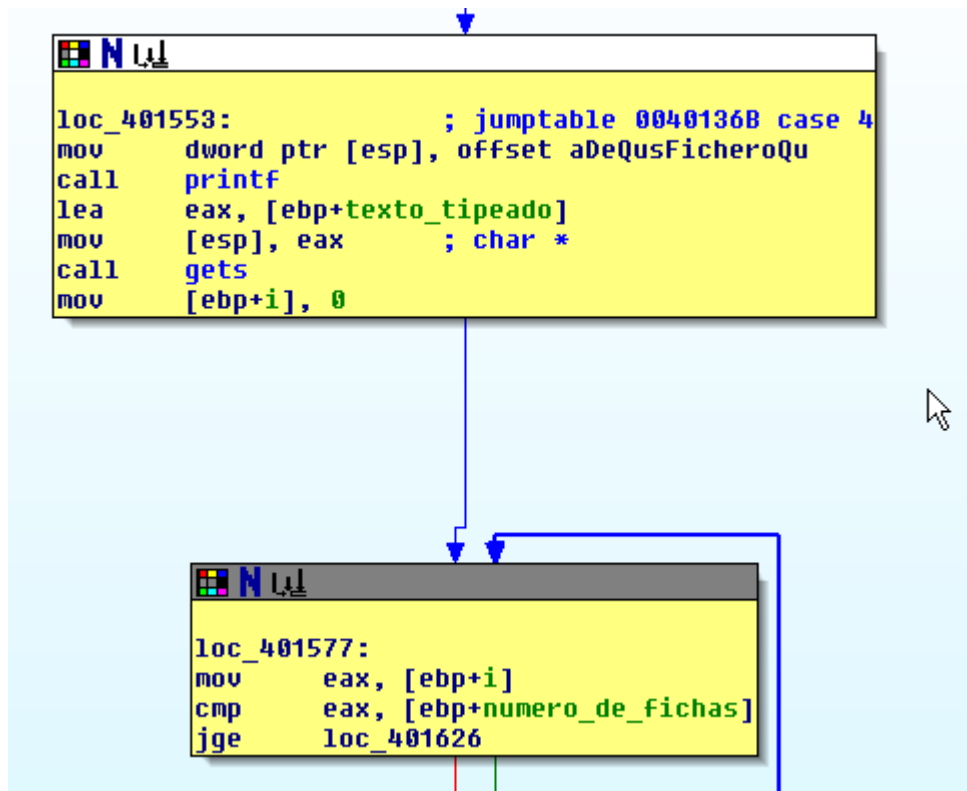
case 3:

```

printf("¿A partir de que tamaño quieres que te muestre?");
gets(texto_tipeado);
sscanf(texto_tipeado, "%ld", &cierto_tamano);
if (fichas[i].tamanio >= cierto_tamano)
    printf("Nombre: %s; Tamaño: %ld Kb\n",
        fichas[i].nombrefich , fichas[i].tamanio);
break;

```

El caso 4 nos pregunta de que fichero queremos ver todos los datos, lo primero que hace es un print con este mensaje.



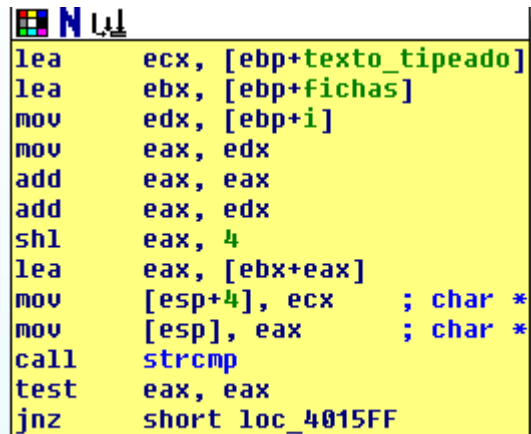
Luego un gets para ingresar el nombre en la variable texto tipeado y luego un for usando i como contador nuevamente:

case 4:

```
printf("¿De qué fichero quieres ver todos los datos?");  
gets(texto_tipeado);  
for (i=0; i<numero_de_fichas; i++)
```

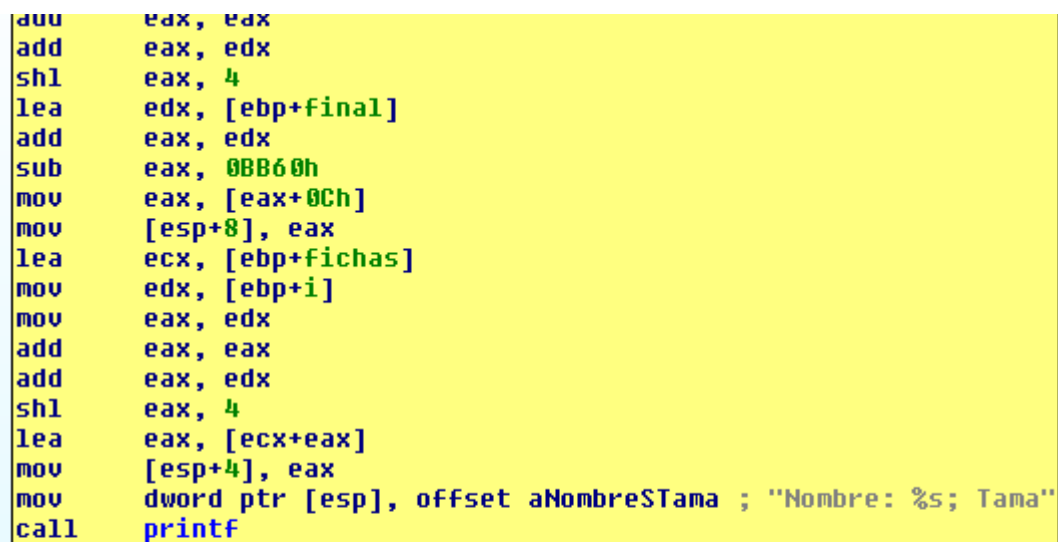

break;

Hasta aquí nada nuevo

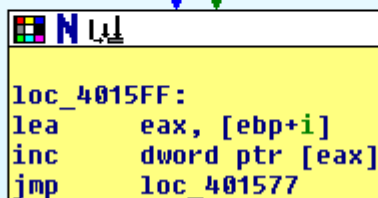


```
lea    ecx, [ebp+texto_tipeado]
lea    ebx, [ebp+fichas]
mov     edx, [ebp+i]
mov     eax, edx
add     eax, eax
add     eax, edx
shl     eax, 4
lea     eax, [ebx+eax]
mov     [esp+4], ecx    ; char *
mov     [esp], eax      ; char *
call    strcmp
test    eax, eax
jnz     short loc_4015FF
```

Así que recorre todos los nombres de las fichas y usa **strcmp** para comparar con lo que tipeamos.



```
add     eax, eax
add     eax, edx
shl     eax, 4
lea     edx, [ebp+final]
add     eax, edx
sub     eax, 0BB60h
mov     eax, [eax+0Ch]
mov     [esp+8], eax
lea     ecx, [ebp+fichas]
mov     edx, [ebp+i]
mov     eax, edx
add     eax, eax
add     eax, edx
shl     eax, 4
lea     eax, [ecx+eax]
mov     [esp+4], eax
mov     dword ptr [esp], offset aNombreSTama ; "Nombre: %s; Tama"
call    printf
```



```
loc_4015FF:
lea     eax, [ebp+i]
inc     dword ptr [eax]
jmp     loc_401577
```

Si no son iguales toma el camino de la flecha verde y no hace nada incrementa la variable i y vuelve a repetir el for, si son iguales tal cual hizo antes imprime el nombre y el tamaño.

case 4:

```
printf("¿De qué fichero quieres ver todos los datos?");
gets(texto_tipeado);
for (i=0; i<numero_de_fichas; i++)
    if (strcmp(fichas[i].nombrefich, texto_tipeado) == 0)
```

```

printf("Nombre: %s; Tamaño: %ld Kb\n",
      fichas[i].nombrefich, fichas[i].tamanio);
break;

```

```

C:\Documents and Settings\ricnar\Escritorio\1315-C Y REVERSING
Escoja una opción:
1.- Añadir datos de un nuevo fichero
2.- Mostrar los nombres de todos los ficheros
3.- Mostrar ficheros que sean de mas de un cierto tamaño
4.- Ver datos de un fichero
5.- Salir
1
Introduce el nombre del fichero: jose
Introduce el tamaño en KB: 2
Escoja una opción:
1.- Añadir datos de un nuevo fichero
2.- Mostrar los nombres de todos los ficheros
3.- Mostrar ficheros que sean de mas de un cierto tamaño
4.- Ver datos de un fichero
5.- Salir
4
De qué fichero quieres ver todos los datos?pepe
Nombre: pepe; Tamaño: 1 Kb
Escoja una opción:
1.- Añadir datos de un nuevo fichero
2.- Mostrar los nombres de todos los ficheros
3.- Mostrar ficheros que sean de mas de un cierto tamaño
4.- Ver datos de un fichero
5.- Salir

```

Vemos que solo nos queda la opción 5 para salir y la opción default si se tipea algo que no este entre 1 y 5.

```

loc_40160C:                ; jumtable 0040136B case 5
mov     dword ptr [esp], offset aFinDelPrograma
call    printf
jmp     short loc_401626

```

El caso 5 solo imprime el mensaje **Fin del programa** y el caso default solo imprime **Opcion desconocida..**

```

5 loc_40161A:                ; jumtable 0040136B case 0
mov     dword ptr [esp], offset a0pci
call    printf

```

```

case 5:
    printf("Fin del programa\n");
    break;
default:
    printf("Opción desconocida!\n");
    break;

```

Así que el código final completo quedaría así:

```
#include <stdio.h>
#include <string.h>

main(){
    funcion();
}

funcion(){
    int eleccion=0;
    int i=0;
    int numero_de_fichas=0;
    char texto_tipeado[40];
    unsigned long cierto_tamano;
    struct{
        char nombrefich[44]; /* Nombre del fichero */
        unsigned long tamano; /* El tamaño en bytes */
    } fichas[1000];

    while (eleccion!=5)
    {
        printf("Escoja una opción:\n");
        printf("1.- Añadir datos de un nuevo fichero\n");
        printf("2.- Mostrar los nombres de todos los ficheros\n");
        printf("3.- Mostrar ficheros que sean de mas de un cierto tamaño\n");
        printf("4.- Ver datos de un fichero\n");
        printf("5.- Salir\n");

        gets(texto_tipeado);
        sscanf(texto_tipeado, "%d", &eleccion);
        switch(eleccion){
            case 1:

                if (numero_de_fichas < 1000) {
                    printf("Introduce el nombre del fichero: ");
                    gets(fichas[numero_de_fichas].nombrefich);
                    printf("Introduce el tamaño en KB: ");
                    gets(texto_tipeado);
                    sscanf(texto_tipeado,"%ld",&fichas[numero_de_fichas].tamano);

                    numero_de_fichas++;
                } else
                    printf("Máximo de fichas alcanzado (1000)!\n");
                break;
            case 2:
                for (i=0; i<numero_de_fichas; i++)
                    printf("Nombre: %s; Tamaño: %ld Kb\n",
                        fichas[i].nombrefich , fichas[i].tamano);
                break;
```

```
printf("¿A partir de que tamaño quieres que te muestre?");
gets(texto_tipeado);
sscanf(texto_tipeado, "%ld", &cierto_tamanio);
if (fichas[i].tamanio >= cierto_tamanio)
    printf("Nombre: %s; Tamaño: %ld Kb\n",
        fichas[i].nombrefich , fichas[i].tamanio);
break;
```

```
printf("¿De qué fichero quieres ver todos los datos?");
gets(texto_tipeado);
for (i=0; i<numero_de_fichas; i++)
if (strcmp(fichas[i].nombrefich, texto_tipeado) == 0)
    printf("Nombre: %s; Tamaño: %ld Kb\n",
        fichas[i].nombrefich, fichas[i].tamano);
break;
```

```
printf("Fin del programa\n");
break;
```

```
printf("Opción desconocida!\n");  
break;
```

$$\left. \begin{array}{l} \} \\ \} \end{array} \right\}$$

```
printf("Nombre: %s; Tamano: %ld Kb\n", nombre, tamano);
```

C:\Documents and Settings\ricnar\Escritorio\1315-C Y REVERSING (parte 9)

Nombre: 2; Tamano: 3 Kb
Escoja una opcion:
1.- Aniadir datos de un nuevo fichero
2.- Mostrar los nombres de todos los ficheros
3.- Mostrar ficheros que sean de mas de un cierto tamano
4.- Ver datos de un fichero
5.- Salir
4
De que fichero quieres ver todos los datos?pepe
Escoja una opcion:
1.- Aniadir datos de un nuevo fichero
2.- Mostrar los nombres de todos los ficheros
3.- Mostrar ficheros que sean de mas de un cierto tamano
4.- Ver datos de un fichero
5.- Salir
4
De que fichero quieres ver todos los datos?jose
Nombre: jose; Tamano: 2 Kb
Escoja una opcion:
1.- Aniadir datos de un nuevo fichero
2.- Mostrar los nombres de todos los ficheros
3.- Mostrar ficheros que sean de mas de un cierto tamano
4.- Ver datos de un fichero
5.- Salir

Disculpen que lo hice a las apuradas pero queria terminar esto para poder continuar con las partes siguientes.

Ricardo Narvaja