

Bueno vamos a tratar de reversear el Ejemplo completo sabemos que es complicado, así que vamos paso a paso, lo primero que debemos ver es lo que hace al ejecutarlo.

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

Tenemos cinco opciones para elegir de 1 a 5, si elegimos 1

```
C:\> C:\Documents and Settings\ricnar\Escritorio\1315-C Y REVERSING (par
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
```

Vemos que nos pide que ingresemos el nombre del fichero y el tamaño y vuelve a salir el mismo menú para elegir otra opción, si elijo 2 me imprime el nombre y el tamaño.

```
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
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
```

Si elijo 3 me pregunta a partir de que tamaño quiero mostrar los ficheros.

```
5.- Salir
3
¿A partir de que tamaño quieres que te muestre?2
Nombre: pepe; Tamaño: 3 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 ta
4.- Ver datos de un fichero
5.- Salir
```

Si elijo 4 me pregunta el nombre de un fichero para darme los datos

```

1 De qué fichero quieres ver todos los datos?pepe
Nombre: pepe; Tamaño: 3 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

```

y si elijo 5 sale del programa.

Evidentemente esto es un switch abramos el programa en IDA.

```

; int __cdecl main(int argc, const ch
_main proc near

var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h

push    ebp
mov     ebp, esp
sub     esp, 8
and     esp, 0FFFFFF0h
mov     eax, 0
add     eax, 0Fh
add     eax, 0Fh
shr     eax, 4
shl     eax, 4
mov     [ebp+var_4], eax
mov     eax, [ebp+var_4]
call    ___chkstk
call    _main
call    sub_4012C1
leave
retn
_main endp

```

Lo primero que vemos es que el main solo tiene una funcion, así que el esquema es similar a los que veníamos viendo.

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

```
main(){
    funcion();
}
```

```
funcion(){

}
```

Podemos entrar y renombrar la funcion.

Lo otro que vemos que hay un llamado a **gets** por lo tanto debemos agregar **#include <string.h>**

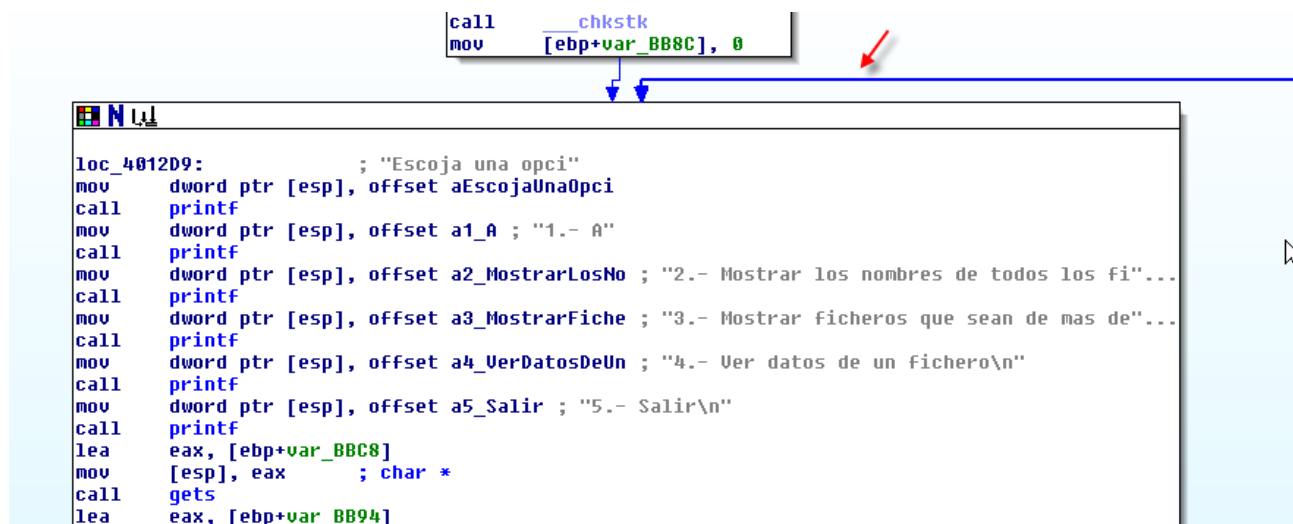
```
#include <stdio.h>
#include <string.h>
```

```
main(){
    funcion();
}
```

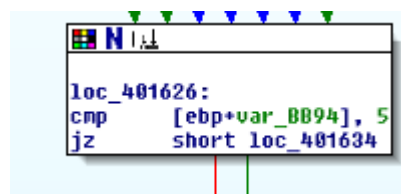
```
funcion(){

}
```

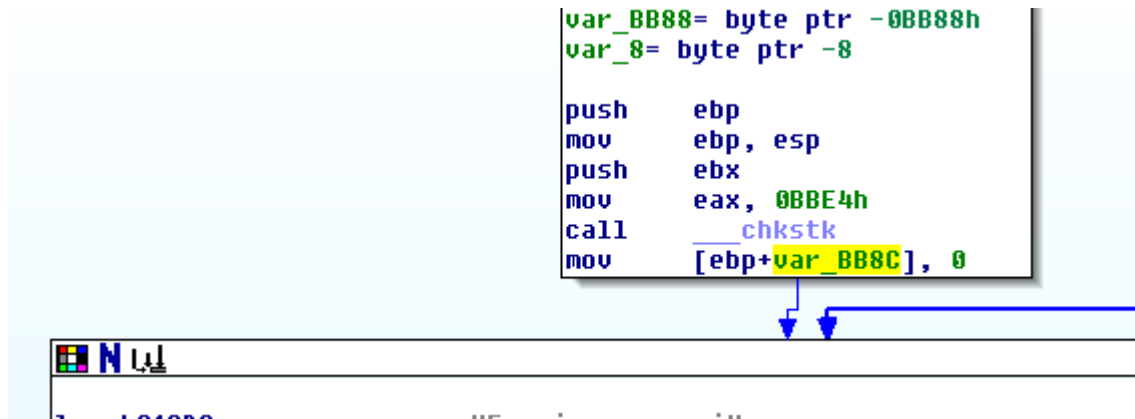
Lo primero que hace es imprimir el mensaje con las opciones que esta aquí, pero vemos la flecha azul que esta volviendo, así que todo el programa esta dentro de un loop



Si vemos de donde viene esa flecha azul y seguimos la misma viene de un **jmp** que antes viene de aquí que es donde se evaluará la condición de salida, la variable para ver si se sale o no del loop es **var_BB94**



y la que inicializa antes de entrar al loop es diferente es **var_BB8C** lo cual hace pensar que no es un **for** el cual tiene la misma variable que se inicializa, incrementa o decrementa y se chequea al final, como aquí no se da el esquema del **for**, suponemos que es un **while** que es mas flexible y no tiene necesariamente que incrementarse algo, en este caso vimos que cuando el usuario tipeaba **5** se salia del programa, así que coincide con la condición de salida, pero la misma depende de lo teclea el usuario no hay contador ni nada por el estilo, así que antes que nada ponemos un **while** y la variable que chequea para la salida la lee posteriormente dentro del mismo ciclo.



```

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

```

```

main(){
    funcion();
}

```

```

funcion(){
    int eleccion;

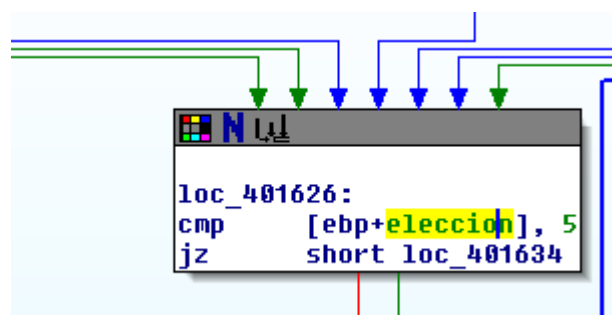
    while (eleccion!=5)
    {

    }

}

```

Así que agregamos un **while** y podemos renombrar la variable que compara a la salida como **elección** por supuesto sera un **int**.



```

function_investigada proc near
var_BBCC= dword ptr -0BBCCCh
var_BB88= byte ptr -0BB88h
eleccion= dword ptr -0BB94h
var_BB90= dword ptr -0BB90h
var_BB8C= dword ptr -0BB8Ch
var_BB88= byte ptr -0BB88h
var_8= byte ptr -8

push    ebp
mov     ebp, esp
push    ebx
mov     eax, 0BBE4h
call    _chkstk
mov     [ebp+var_BB8C], 0

```

Bueno una vez dentro del while imprime el mensaje que muestra las diferentes opciones del menú, vemos en el IDA que es una seguidilla de **printf**.

```

loc_4012D9:                ; "Escoja una opci"
mov     dword ptr [esp], offset aEscojaUnaOpci
call    printf
mov     dword ptr [esp], offset a1_A ; "1.- A"
call    printf
mov     dword ptr [esp], offset a2_MostrarLosNo ; "2.- Mostrar los nombres de todos los fi"...
call    printf
mov     dword ptr [esp], offset a3_MostrarFiche ; "3.- Mostrar ficheros que sean de mas de"...
call    printf
mov     dword ptr [esp], offset a4_VerDatosDeUn ; "4.- Ver datos de un fichero\n"
call    printf
mov     dword ptr [esp], offset a5_Salir ; "5.- Salir\n"
call    printf

```

Así que hacemos las llamadas a **printf** pasándoles las mismas strings para que se construya el mismo menú de opciones, colocamos **\n** para que haya salto de línea al final de cada **printf**.

```

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

```

```

main(){
    funcion();
}

```

```

funcion(){
    int eleccion;

```

```

    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");
    }
}

```

```

lea    eax, [ebp+var_BBC8]
mov     [esp], eax      ; char *
call    gets
lea     eax, [ebp+eleccion]
mov     [esp+8], eax
mov     dword ptr [esp+4], offset aD ; "%d"
lea     eax, [ebp+var_BBC8]
mov     [esp], eax      ; char *
call    sscanf

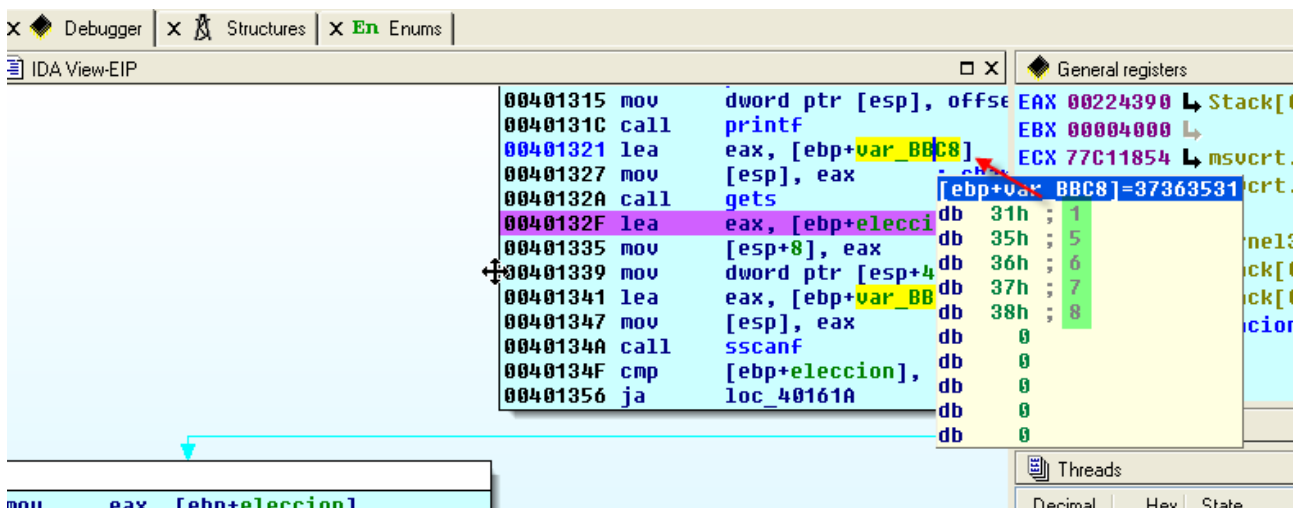
```

Bueno aquí la cuestión es que usa **gets** para obtener la string que tipea el usuario y **sscanf** que es igual a **scanf** pero toma como entrada una variable en vez del teclado, o sea lo hace así para filtrar la entrada, como vimos que printf hace un format string del tipo:

printf("El número 50 multiplicado por 2 vale %d\n", 50*2);

donde 100 es un **int** y lo transforma a la string "100", ahora **scanf** usando **%d** hace lo opuesto de una string que tipeamos por ejemplo "100" halla el valor numérico 100 y **sscanf** hace lo mismo pero en vez de tomar como entrada lo que tipea el usuario, toma como entrada una variable con una string o array de caracteres.

Para que se entienda mas puse un breakpoint al volver de gets y tipee **15678** en la consola



Ahí vemos que al volver de **gets** la variable tiene esos bytes que tipee, ahora sacara el valor numérico.

```

00401328 call    gets
0040132F lea     eax, [ebp+eleccion]
00401335 mov     [esp+8], eax
00401339 mov     dword ptr [esp+4], offset aD ; "%d"
00401341 lea     eax, [ebp+var_BBC8]
00401347 mov     [esp], eax      ; char *
0040134A call    sscanf

```

Al llegar a **sscanf** vemos los tres argumentos la dirección de la variable **elección** la cual obtiene mediante **lea** y la inicializa aquí en **sscanf**, el segundo argumento es el formato **%d** y el tercero la string a la cual le aplicara el formato que es **var_BBC8** que tiene lo tipeado por mi **15678**.

El resultado es en mi caso **3d3eh** que es el valor hexa de **15678**, lo mismo que si hubiéramos tipeado "100", en la variable elección tendríamos **64h** que es **100** decimal.

Realmente podría haber usado **scanf** solamente pero bueno así es el código jeje, así que la **var_BBC8** debe ser un array de caracteres de cierto tamaño, para guardar lo que el usuario tipee, veamos las variables.

0000BBCC	var_BBCC	dd ?	
0000BBC8	var_BBC8	db ?	
0000BBC7		db ?	; undefined
0000BBC6		db ?	; undefined
0000BBC5		db ?	; undefined
0000BBC4		db ?	; undefined
0000BBC3		db ?	; undefined
0000BBC2		db ?	; undefined
0000BBC1		db ?	; undefined
0000BBC0		db ?	; undefined
0000BBBF		db ?	; undefined
0000BBBE		db ?	; undefined
0000BBBD		db ?	; undefined
0000BBBC		db ?	; undefined
0000BBBB		db ?	; undefined
0000BBBA		db ?	; undefined
0000BBB9		db ?	; undefined
0000BBB8		db ?	; undefined
0000BBB7		db ?	; undefined
0000BBB6		db ?	; undefined
0000BBB5		db ?	; undefined
0000BBB4		db ?	; undefined
0000BBB3		db ?	; undefined
0000BBB2		db ?	; undefined
0000BBB1		db ?	; undefined

Si seguimos mirando hacia abajo la siguiente variable es **elección** que no se puede pisar ni pertenece a este array, así que ya podemos definir apretando asterisco en **var_BBC8** y tomando todo el espacio disponible hasta la variable elección.

The screenshot shows the 'Struct field size' dialog box. The 'Array element width' is set to 1, and the 'Array size' is set to 52. The 'Items on a line' is 0, and the 'Alignment' is -1. The 'Use "dup" construct' checkbox is checked. The 'Signed elements' and 'Display indexes' checkboxes are unchecked. The 'OK', 'Cancel', and 'Help' buttons are at the bottom.

Es importante que la **var_BBC8** este definida como de un solo byte ya que es un array de caracteres y cada campo es un byte.

-0000BBBD		db ? ; undefined
-0000BBBC	var_BBCC	dd ?
-0000BBC8	var_BBCC	db 52 dup(?)
-0000BB94	eleccion	dd ?
-0000BB90	var_BB90	dd ?
-0000BB8C	var_BB8C	dd ?
-0000BB88	var_BB88	db ?
-0000BB87		dh ? : undefined

Renombremos a **texto_tipeado**

-0000BBBD		db ? ; undefined
-0000BBBC	var_BBCC	dd ?
-0000BBC8	texto_tipeado	db 52 dup(?)
-0000BB94	eleccion	dd ?
-0000BB90	var_BB90	dd ?
-0000BB8C	var_BB8C	dd ?
-0000BB88	var_BB88	db ?
-0000BB87		db ? ; undefined

Ya tenemos dos variables renombradas y bien definidas, debemos agregar al código fuente que estamos armando las llamadas a **gets** y **scanf** y inicializar elección a cero porque sino al llegar al **while** no tendrá valor inicial y dará error.

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

```
main(){
    funcion();
}
```

```
funcion(){
    int eleccion=0;
    char texto_tipeado[40];
```

```
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);
    scanf(texto_tipeado, "%d", &eleccion);
```

```

}
```

Este ultimo ejemplo al menos ya compila y si tipeamos 5 sale así que vamos bien ahora tenemos

que hacer el switch.

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

main(){
funcion()
}

funcion(){
    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");
    int eleccion;
    char texto_tipeado[40];
    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:
                printf("Tipeaste 1\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("Opción no válida\n ");
                break;
        }
    }
}
```

Allí vemos en el IDA nos dice que hay un switch de 6 casos, son las cinco opciones que tenemos para tipear de 1 a 5 y la restante es la opción por default por si tipeamos cualquier cosa diferente que no este entre 1 y 5, así que agreguemos el switch a nuestro código.

```
lea     eax, [ebp+texto_tipeado]
mov     [esp], eax ; char *
call    scanf
cmp     [ebp+eleccion], 5 ; switch 6 cases
ja      loc_40161A ; jumtable 0040136B case 0
```

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

```
main(){
    funcion();
}
```

```
funcion(){
    int eleccion=0;
    char texto_tipeado[40];
```

```
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:
            printf("Tipeaste 1\n ");
            break;
```

```

        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;
}

```

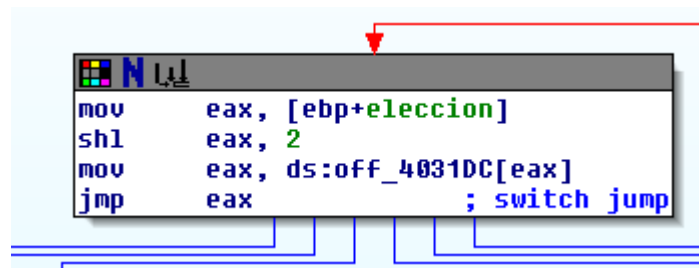
```

}
}

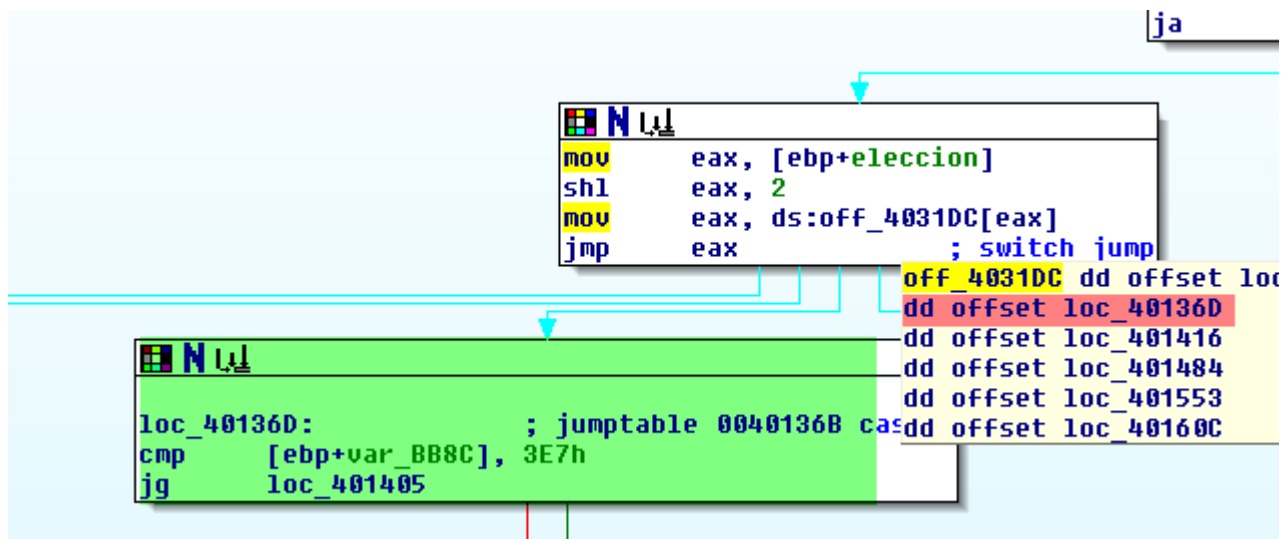
```

Si lo compilamos vemos que si elegimos **1** nos dice **Tipeaste 1**, y así sucesivamente, si pones cualquier valor de que no este entre **1** y **5** te sale la opción por default “**Cualquier huevada**”, y al apretar 5 se sale.

Así que el esquema general esta ahora falta ver que hace en cada case, empecemos por el case 1.



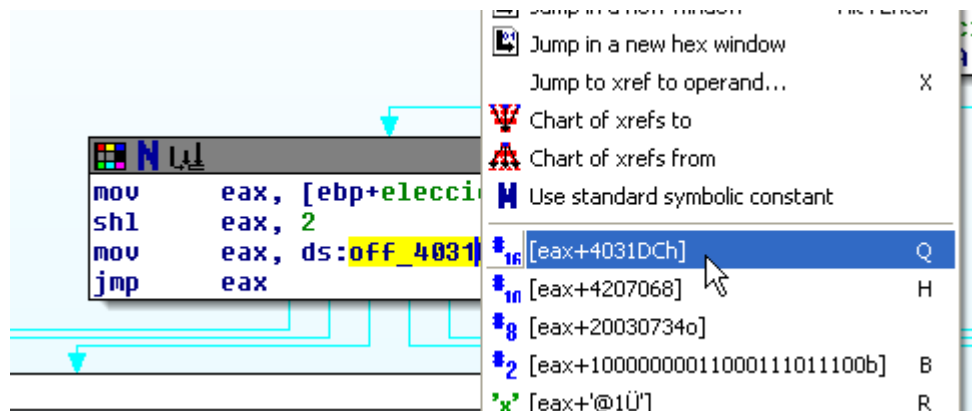
Bueno esto es parte del mecanismo interno del switch vemos que agarra el valor que tipeamos y lo multiplica por cuatro con **SHL EAX,2** de esta forma como la tablita de direcciones va de cuatro en cuatro podrá recorrerla con nuestro indice.



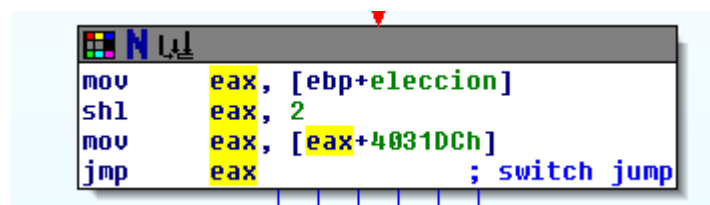
Allí vemos que la base de la tabla esta en **4031DC**.

mov eax, ds:off_4031DC[eax]

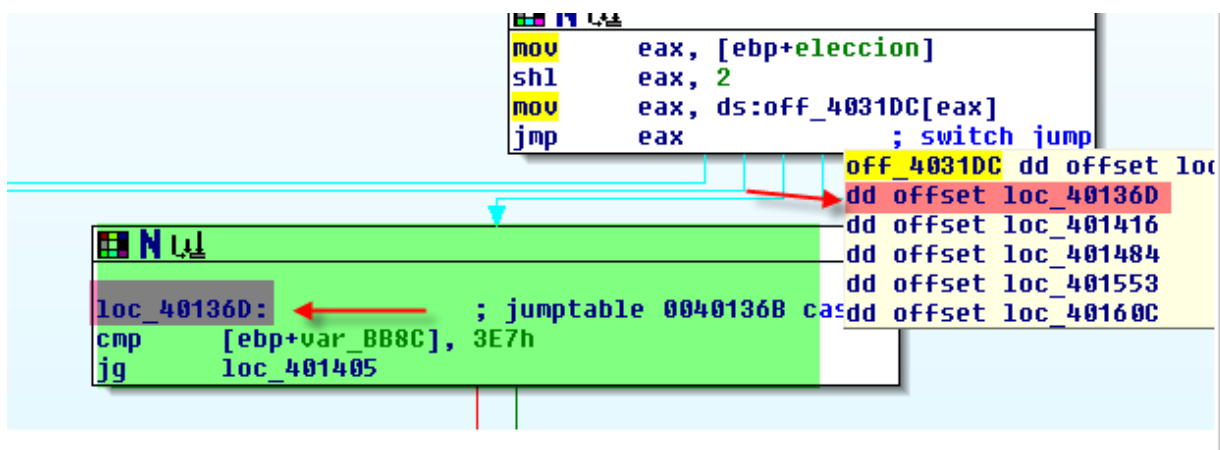
Si no nos gusta la forma en que nos muestra la instrucción hacemos click derecho y buscamos la mas confortable.



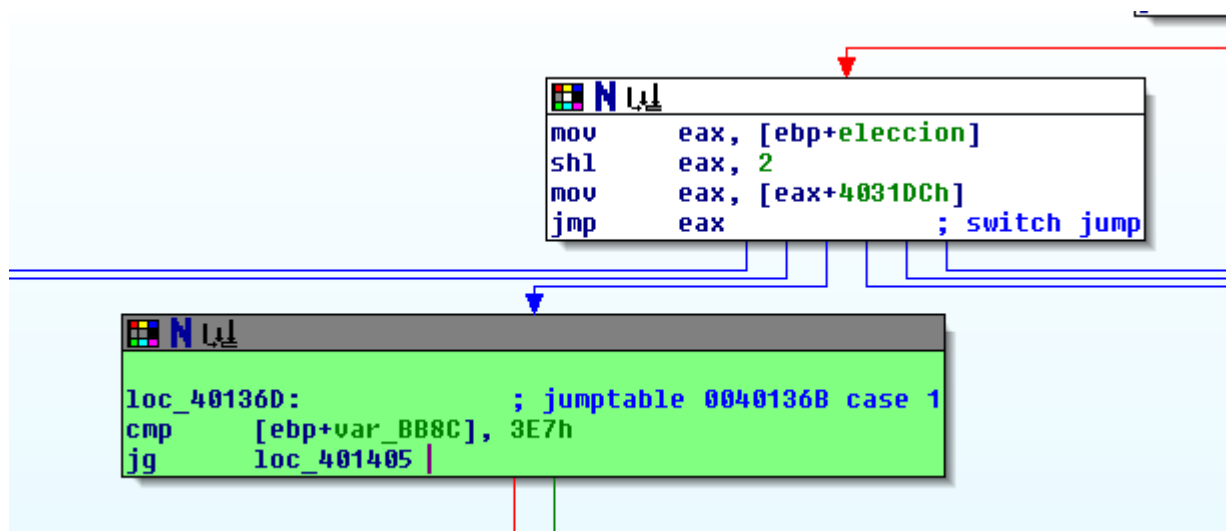
Elegimos esa.



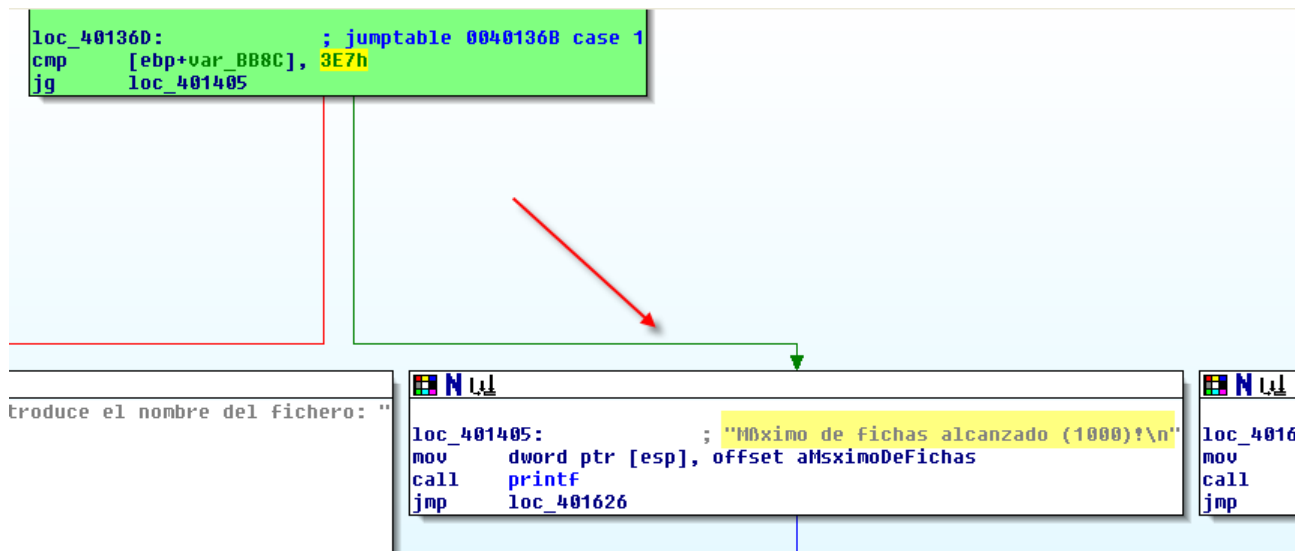
Ahora se ve mas lindo si EAX vale 0, multiplicado por cuatro seguirá valiendo cero e ira al contenido de **4031DC** que es el caso cero, si EAX vale 1 que es el caso que estamos mirando al multiplicar por 4 valdrá 4, así que sera el contenido de **[4031dc +4]** o sea el segundo valor de la tablita de switch **40136d**.



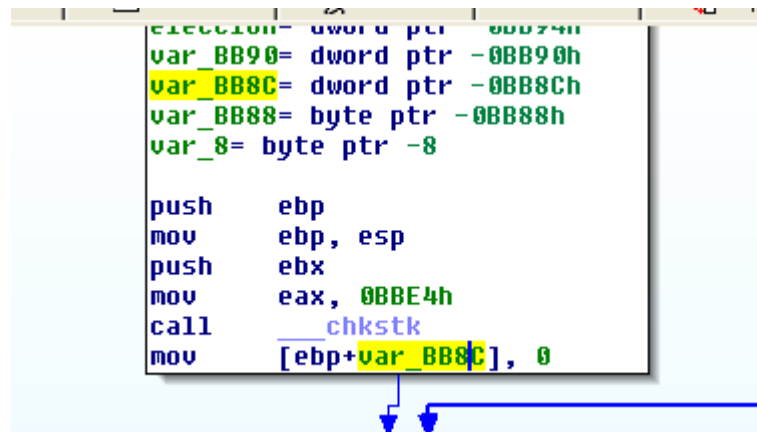
Así que el bloque verde es el **caso 1**, igual IDA nos dice, así que aunque no hagamos todo este análisis sabemos que es el caso 1 jeje.



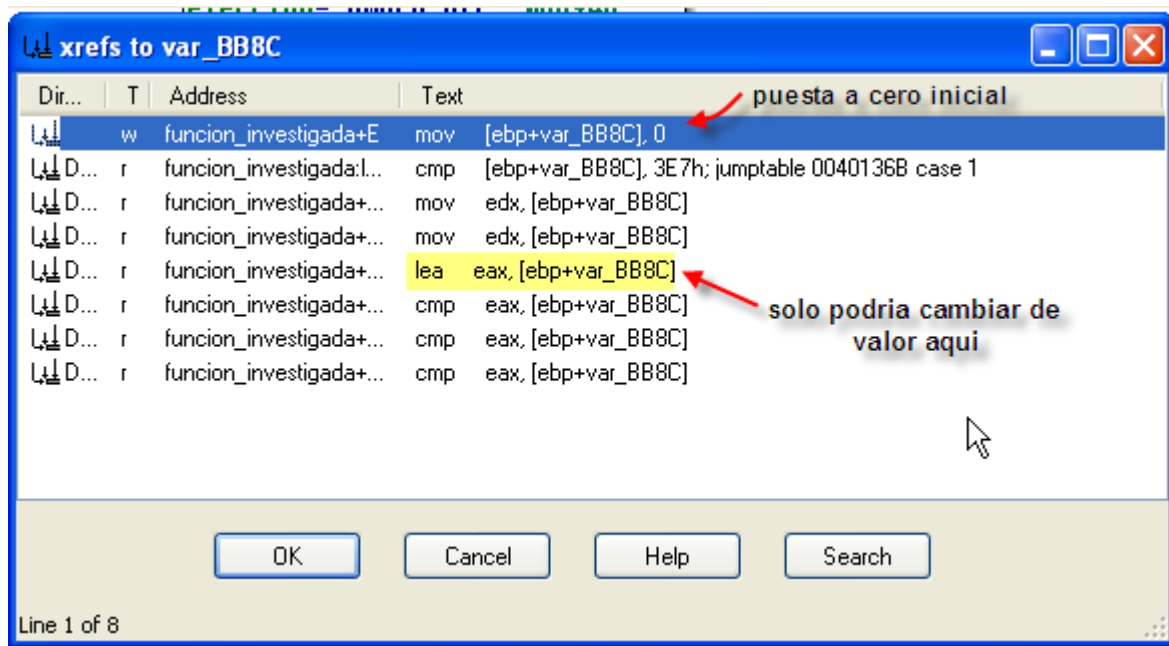
Allí vemos que en el caso 1 lo primero que hace es comparar una variable que aun no usamos **var_BB8C** con **3e7h** que es **999** decimal y si es mas grande o sea si vale **1000** va por la flecha verde.



Vemos que si vale 1000 nos manda a un printf que dice **Máximo numero de fichas alcanzado** (1000), así que esta variable sera el numero de fichas.



Vemos que dicha variable al iniciar se inicializa a cero, y luego si apretamos X veremos donde puede cambiar de valor.



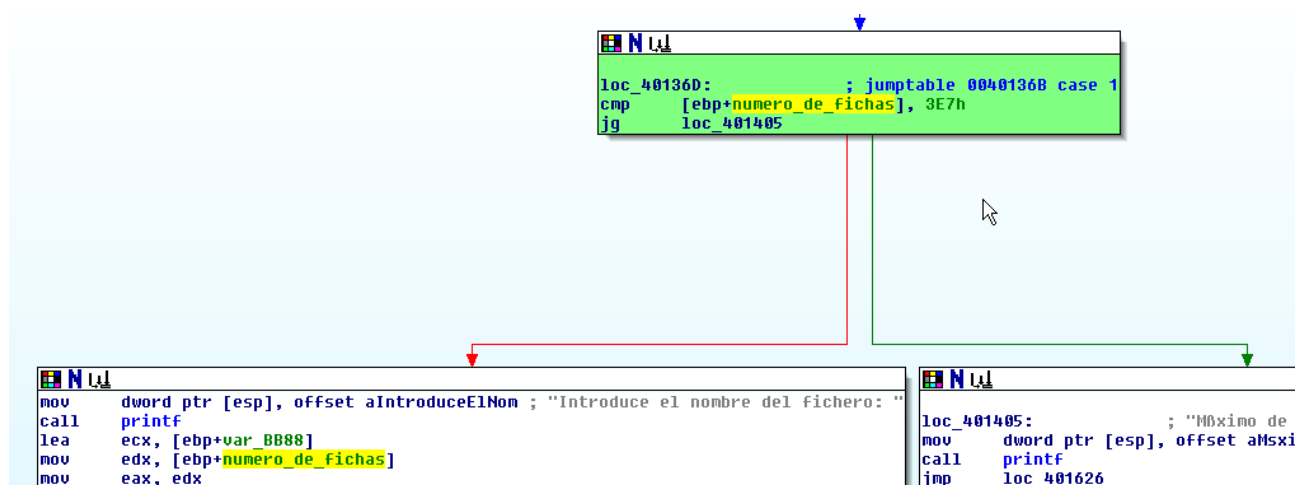
Vemos que dicha variable fuera de la inicializacion inicial, se compara o se lee, pero solo podría cambiar de valor en el **lea**, pues allí obtiene la dirección de la variable y podría trabajar con ella, vayamos allí.

```

add    eax, 20h
mov     [esp+8], eax
mov     dword ptr [esp+4], offset aLd ;
lea     eax, [ebp+texto_tipeado]
mov     [esp], eax ; char *
call    scanf
lea     eax, [ebp+var_BB8C]
inc     dword ptr [eax]
jmp     loc_401626

```

Allí al final del **caso 1** que sirve para crear nuevas fichas se incrementa, así que sabemos que esa variable es obviamente el numero de fichas, que su máximo es 999 y se incrementa cada vez que se agrega una nueva ficha cuando se va al campo 1, la renombramos.

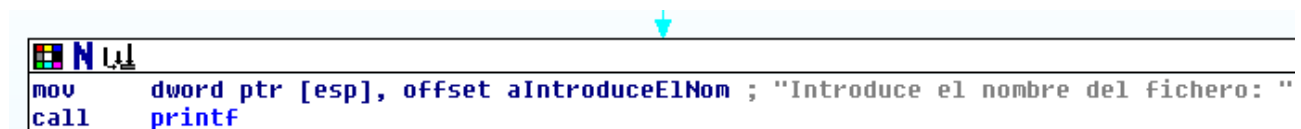


Así que el caso 1 por ahora es algo así, si es menor a 1000 entra y luego de hacer varias cosas que aun no vimos incrementa la variable, si no va al **else** y imprime el cartel de que se alcanzó el

máximo.

```
if (numero_de_fichas < 1000) {  
    printf("Introduce el nombre del fichero: ");  
  
    numero_de_fichas++;  
} else  
    printf("Máximo de fichas alcanzado (1000)!\n");
```

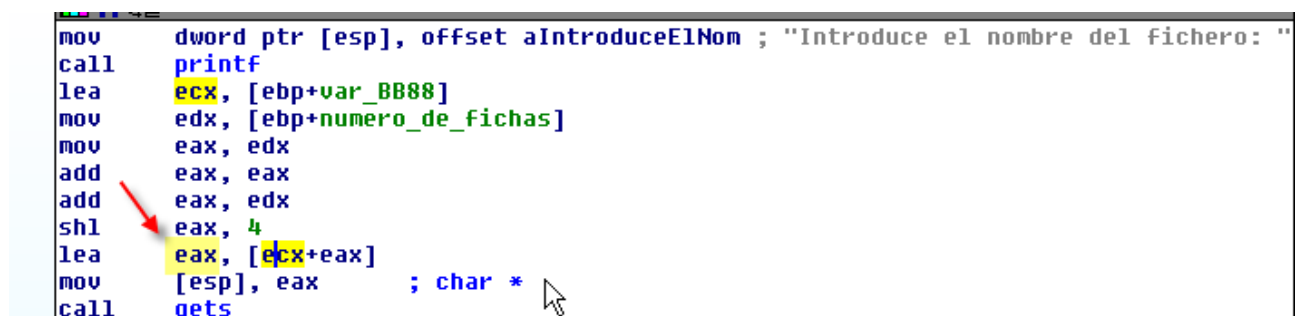
Lo primero que hay es la impresión del mensaje siguiente:



```
mov     dword ptr [esp], offset aIntroduceElNom ; "Introduce el nombre del fichero: "  
call    printf
```

Así que lo agregamos al código

```
if (numero_de_fichas < 1000) {  
    printf("Introduce el nombre del fichero: ");  
  
    numero_de_fichas++;  
} else  
    printf("Máximo de fichas alcanzado (1000)!\n");
```



```
mov     dword ptr [esp], offset aIntroduceElNom ; "Introduce el nombre del fichero: "  
call    printf  
lea     ecx, [ebp+var_BB88]  
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
```

Vemos que va a introducir el nombre del primer fichero, así que debemos armar la estructura, la misma es un array de 1000 donde cada campo es una estructura que tiene dos datos el tamaño que es un **long** y el nombre que es un array de caracteres de un tamaño que aun debemos especificar seria algo así.

```
struct{  
    char nombrefich[x]; /* Nombre del fichero */  
    unsigned long tamano; /* El tamaño en bytes */  
} fichas[1000];
```

Si completamos el código con lo que hay seria así

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

```

main(){
    funcion();
}

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

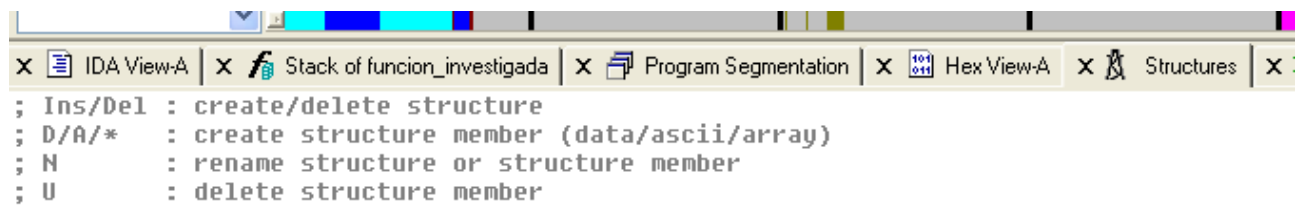
    while (eleccion!=5)
    {
        printf("Escoja una opción:\n");
        printf("1.- Añadir datos de un nuevo fichero\n");
        printf("2.- Mostrar los nombres de todos los ficheros\n");
        printf("3.- Mostrar ficheros que sean de mas de un cierto tamaño\n");
        printf("4.- Ver datos de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

```


Ahora nos queda crear la estructura.



La misma solo tiene dos campos un array de caracteres y un float que es un dword para el tamaño, sabemos que al inicio va el array de caracteres con el nombre y luego el otro campo es un float con el tamaño, pero aquí lo mas difícil es hallar el tamaño máximo del array de caracteres del nombre eso lo analizaremos a continuación.

Veamos esto

```
lea    ecx, [ebp+var_BB88]
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
```

Para simplificar como la primera vez que entramos el numero de fichas vale cero, lo podemos resumir a esto.

```
lea     ecx, [ebp+var_BB88]
lea     eax, [ecx]
mov     [esp], eax    ; char *
call    gets
```

Por supuesto como **var_BB88** es el inicio del array de estructuras se toma como base y ahí estará guardando en el sin ningún límite el nombre que tipea el usuario.

```
mov     dword ptr [esp], offset aIntroduceElTam ; "Introduce el tama"
call    printf
lea     eax, [ebp+texto_tipeado]
mov     [esp], eax    ; char *
call    gets
```

Luego reusa la variable **texto_tipeado** para introducir el tamaño.

```
lea     ecx, [ebp+var_BB88]
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
mov    [esp+8], eax
mov    dword ptr [esp+4], offset aLd ; "%ld"
lea    eax, [ebp+texto_tipeado]
mov    [esp], eax    ; char *
call   scanf

```

Esta parte final como numero de fichas vale cero se reduce a esto.

```

lea    ecx, [ebp+var_BB88]
lea    eax, [ecx]
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

```

Vemos que toma la dirección inicial y le suma **2ch** o sea **44** decimal para crear una variable allí y guardar el tamaño

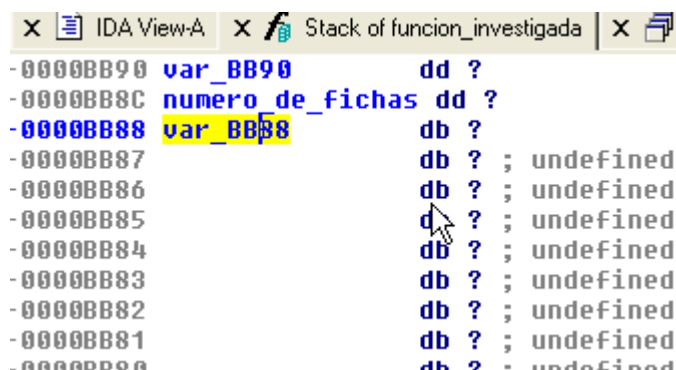
O sea que creamos la estructura de esa forma 44 bytes para **nombrefich** y un dword para **tamanio**.

Ahora vamos a la variable apretamos ALT mas Q y le asignamos la estructura esta.

```

----- ; -----
00000000 ;
00000000
00000000 struc_1      struc ; (sizeof=0x30)
00000000 nombrefich    db 44 dup(?)
0000002C tamanio      dd ?
00000030 struc_1      ends
00000030

```



```

IDA View-A  Stack of funcion_investigada
-0000BB90 var_BB90      dd ?
-0000BB8C numero_de_fichas dd ?
-0000BB88 var_BB88     db ?
-0000BB87 db ? ; undefined
-0000BB86 db ? ; undefined
-0000BB85 db ? ; undefined
-0000BB84 db ? ; undefined
-0000BB83 db ? ; undefined
-0000BB82 db ? ; undefined
-0000BB81 db ? ; undefined
-0000BB80 db ? ; undefined

```

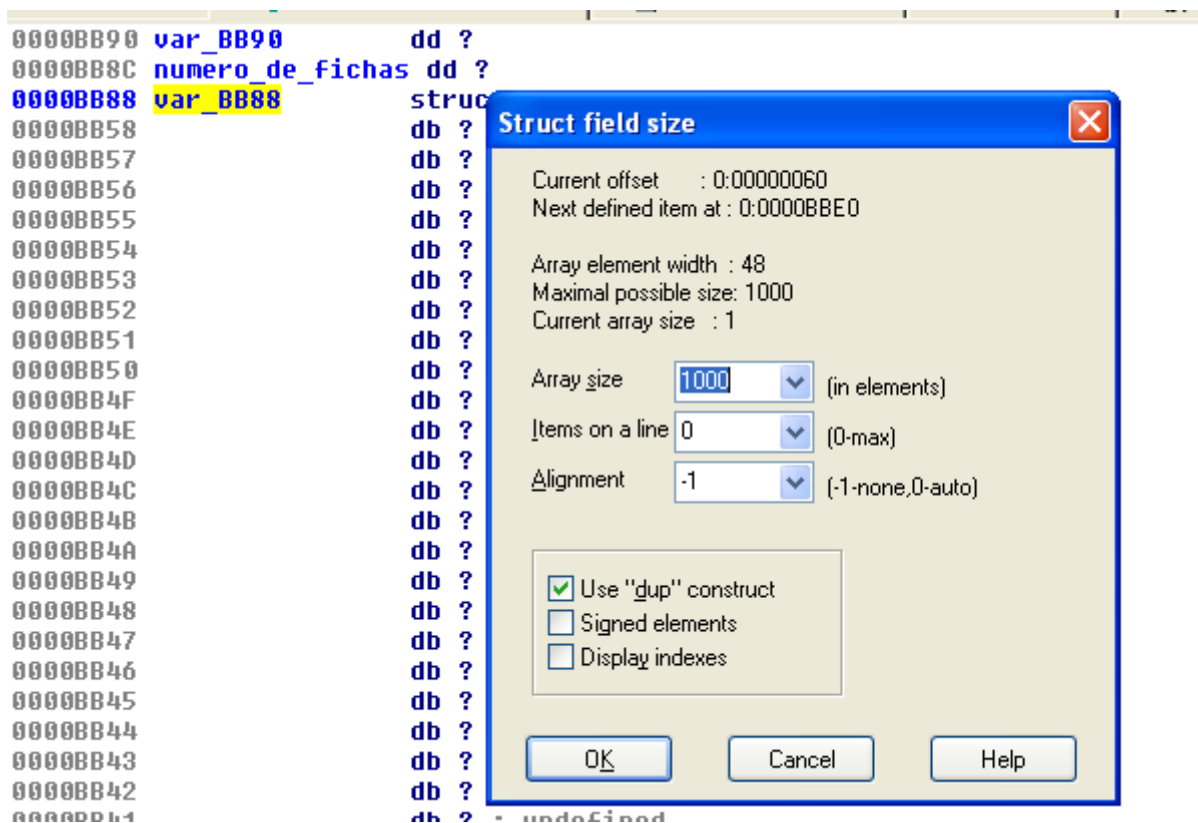
Quedara asi

```

0000BB8C numero_de_fichas dd ?
0000BB88 var_BB88 struc_1 ?
0000BB58 db ? ; undefined
0000BB57 db ? ; undefined
0000BB56 db ? ; undefined
0000BB55 db ? ; undefined
0000BB54 db ? ; undefined
0000BB53 db ? ; undefined
0000BB52 db ? ; undefined
0000BB51 db ? ; undefined
0000BB50 db ? ; undefined
0000BB4F db ? ; undefined
0000BB4E db ? ; undefined

```

Ahora falta hacer el array de 1000 de estas estructuras, aprieto asterisco.



Dice que puede meter justo 1000 así que justo entrara allí,

```

x IDA View-A x Stack of funcion_investigada x Program
-0000BB90 var_BB90 dd ?
-0000BB8C numero_de_fichas dd ?
-0000BB88 fichas struc_1 1000 dup(?)
-0000BB08 var_8 db ?
-0000BB07 db ? ; undefined
-0000BB06 db ? ; undefined
-0000BB05 db ? ; undefined

```

Ahora quedo mejor.

Para comprobar pongo un breakpoint allí, tipeo nombre pepe y tamaño 33..

```

IDA View-EIP
.text:004013D5 shl     eax, 4
.text:004013D8 lea     eax, [ecx+eax]
.text:004013DB add     eax, 2Ch
.text:004013DE mov     [esp+8], eax
.text:004013E2 mov     dword ptr [esp+4], offset aLd
.text:004013EA lea     eax, [ebp+texto_tipeado]
.text:004013F0 mov     [esp], eax
.text:004013F3 call    sscanf
.text:004013F8 lea     eax, [ebp+numero_de_fichas]
.text:004013FE inc     dword ptr [eax]
.text:00401400 jmp     loc_401626
.text:00401405 ; -----

```

y cuando para hago doble click en la variable fichas y veo el nombre pepe, si allí hago ALT mas Q y le asigno la estructura.

```

Stack[000004E0]:002243CB db 0
Stack[000004E0]:002243CC db 0
Stack[000004E0]:002243CD db 0
Stack[000004E0]:002243CE db 0
Stack[000004E0]:002243CF db 0
Stack[000004E0]:002243D0 db 70h ; p
Stack[000004E0]:002243D1 db 65h ; e
Stack[000004E0]:002243D2 db 70h ; p
Stack[000004E0]:002243D3 db 65h ; e
Stack[000004E0]:002243D4 db 0
Stack[000004E0]:002243D5 db 0
Stack[000004E0]:002243D6 db 0
Stack[000004E0]:002243D7 db 0
Stack[000004E0]:002243D8 db 0
Stack[000004E0]:002243D9 db 0
Stack[000004E0]:002243DA db 0
Stack[000004E0]:002243DB db 0
Stack[000004E0]:002243DC db 0
Stack[000004E0]:002243DD db 0

```

```

004E0]:002243CF dd 0
004E0]:002243D0 db 70h, 65h, 70h, 65h, 28h dup(0) ; nombrefich
004E0]:002243D0 dd 21h ; tamaño
004E0]:00224400 db 0

```

Allí veo los dos campos el **nombrefich** y el **tamaño** que es 21h o 33 decimal.

Toqueteando un poco hice que me lo muestre así

```

004E0]:002243CE dd 0
004E0]:002243CF db 0
004E0]:002243D0 struct_1 <'pepe', 21h>
004E0]:00224400 db 0
004E0]:00224401 db 0

```

si seguimos agregando mas fichas

```

004E0]:002243CF db 0
004E0]:002243D0 struc_1 <'pepe', 21h>
004E0]:00224400 db 72h ; r
004E0]:00224401 db 69h ; i
004E0]:00224402 db 63h ; c
004E0]:00224403 db 68h ; k
004E0]:00224404 db 79h ; y
004E0]:00224405 db 0
004E0]:00224406 db 0

```

```

004E0]:002243CF db 0
004E0]:002243D0 struc_1 <'pepe', 21h>
004E0]:00224400 struc_1 <'ricky', 2Ch>
004E0]:00224430 db 0

```

Quedo lindo jeje asi que para acceder a los campos tenemos para el primer inscripto **ficha[0].nombrefich** y **ficha[0].tamanio** y luego **ficha[x].nombrefich** y **ficha[x].tamanio** para los restantes.

Como este ejercicio es muy largo lo voy a sacar en dos partes hasta aquí la parte 1, ya creamos el esquema general y armamos el array de estructuras nos falta completar el resto de los casos.

Hasta la parte 2 de la solución del ejercicio
ricnar