

C Y REVERSING (parte 8) por Ricnar

ESTRUCTURAS

Hemos utilizado en la parte anterior en IDA estructuras para solucionar parcialmente, la poca posibilidad que al menos yo conozco en el mismo para manejar arrays o tablas de varias dimensiones.

Ahora veremos el tema estructuras en C códigos de ejemplos y reversearemos los mismos.

Una estructura es una agrupación de datos, los cuales no necesariamente son del mismo tipo. Se definen con la palabra “**struct**”.

Para acceder a cada uno de los datos, tanto si queremos leer su valor como si queremos cambiarlo, se debe indicar el nombre de la variable o instancia y el del dato (o campo) separados por un punto:

Veamos un ejemplo:

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

```
main(){
```

```
    funcion1();  
    getchar();  
}
```

```
funcion1(){
```

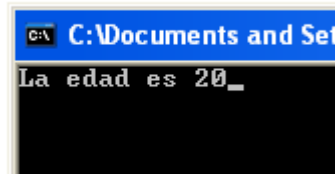
```
    struct mia  
    {  
        char inicial;  
        int edad;  
        float nota;  
    } persona;
```

```
    persona.inicial = 'J';  
    persona.edad = 20;  
    persona.nota = 7.5;  
    printf("La edad es %d", persona.edad);
```

```
}
```

Vemos resaltado la definición de la estructura llamada **mia**, y sus campos, al final del mismo se coloca el nombre de la instancia en este caso **persona**, el nombre **mia** puede no estar si no vamos a realizar mas de una instancia de este mismo tipo, en el caso de que lo coloquemos podremos instanciar mas variables con la misma estructura, la cual tiene tres campos un **char** llamado **inicial**, un **int** llamado **edad** y un **float** llamado **nota**, luego vemos como se inicializan los tres campos y luego como se imprime **persona.edad**.

Si lo ejecutamos vemos.



Si quisiéramos realizar otra instancia en mia, podríamos agregar este código:

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

```
main(){
```

```
    funcion1();  
    getchar();  
}
```

```
funcion1(){
```

```
    struct mia  
    {  
        char inicial;  
        int edad;  
        float nota;  
    } persona, pepe;
```

```
    persona.inicial = 'J';  
    persona.edad = 20;  
    persona.nota = 7.5;  
    printf("La edad es %d\n", persona.edad);
```

```
    pepe.edad=30;  
    printf("La edad de pepe es %d\n", pepe.edad);  
}
```

Esta seria una posibilidad en la declaración de las estructuras, poner las instancias separadas por comas en este caso le agregamos la instancia **pepe**, pero en el caso que quisiéramos agregar otra instancia mas adelante en el programa, podríamos hacerlo así.

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

```
main(){
```

```

funcion1();
getchar();
}

funcion1(){

struct mia
{
    char inicial;
    int  edad;
    float nota;
} persona;

persona.inicial = 'J';
persona.edad = 20;
persona.nota = 7.5;
printf("La edad es %d\n", persona.edad);

struct mia pepe;
pepe.edad=30;
printf("La edad de pepe es %d\n", pepe.edad);
}

```

Vemos que ahora quitamos **pepe** de la definición de estructura **mia**, pero lo agregamos posteriormente como otra instancia, como vemos en la instrucción resaltada y haciendo uso del nombre de la estructura para saber que se agrega a ella, si no tuviera nombre no podríamos adicionarle mas instancias al vuelo que las definidas originalmente.

Veamos el caso mas sencillo en el IDA

```

main(){

funcion1();
getchar();
}

funcion1(){

struct mia
{
    char inicial;
    int  edad;
    float nota;
} persona;

persona.inicial = 'J';
persona.edad = 20;
persona.nota = 7.5;
printf("La edad es %d", persona.edad);
}

```

}

Como vemos siempre IDA nos muestra todas variables sueltas

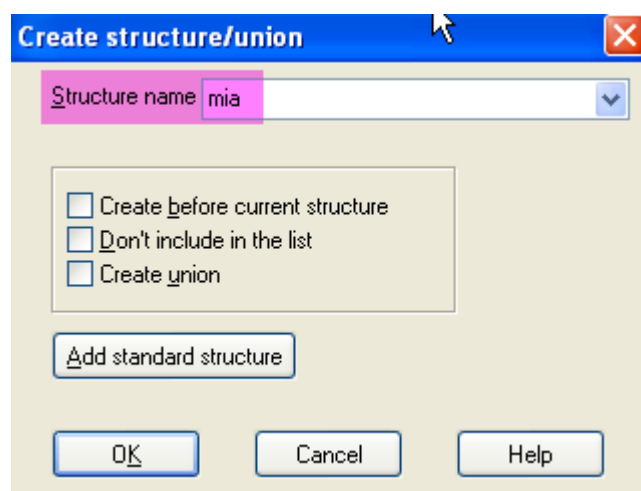
```
sub_4012C6 proc near
var_18= byte ptr -18h
var_14= dword ptr -14h
var_10= dword ptr -10h

push    ebp
mov     ebp, esp
sub     esp, 28h
mov     [ebp+var_18], 4Ah
mov     [ebp+var_14], 14h
mov     eax, 40F00000h
mov     [ebp+var_10], eax
mov     eax, [ebp+var_14]
mov     [esp+4], eax
mov     dword ptr [esp], offset aLaEdadEsD ; "La edad es %d"
call    printf
leave
retn
sub_4012C6 endp
```

En este caso al campo de la estructura llamado **inicial** que era un **char** aquí lo muestra como una variable char suelta, a los otros dos campos **int**, llamados **edad** y **nota** también.

Bueno debemos ir a la ventana estructuras y crear una estructura acorde a los tres campos que tenemos en ella, sabemos que un array no es pues el mismo no puede contener diferentes tipos de datos, podrían ser solo **int** o solo **char**, por ejemplo, pero en este caso hay un campo **char** y dos **int** así que no tenemos duda al reversear que es una estructura.

En la ventana de estructuras apretamos INSERT y ponemos el nombre de la misma en este caso **mia** y damos OK.



```

00000000 ; Ins/Del : create/delete structure
00000000 ; D/A/*   : create structure member (data/ascii/array)
00000000 ; N       : rename structure or structure member
00000000 ; U       : delete structure member
00000000 ; -----
00000000
00000000 mia          struc ; (sizeof=0x0)
00000000 mia          ends
00000000

```

Ahora hay que agregar un **char** en este caso apretamos la D y sale ya directamente un campo **db**.

```

IDA View-A | Hex View-A | Structures | Enums
00000000 ; Ins/Del : create/delete structure
00000000 ; D/A/*   : create structure member (data/a
00000000 ; N       : rename structure or structure m
00000000 ; U       : delete structure member
00000000 ; -----
00000000
00000000 mia          struc ; (sizeof=0x1)
00000000 field_0        db ?
00000001 mia          ends
00000001

```

El nombre de este campo es inicial así que lo renombramos.

```

00000000 ; Ins/Del : create/delete structure
00000000 ; D/A/*   : create structure member (
00000000 ; N       : rename structure or struc
00000000 ; U       : delete structure member
00000000 ; -----
00000000
00000000 mia          struc ; (sizeof=0x1)
00000000 inicial        db ?
00000001 mia          ends
00000001

```

Ahora vamos a **ends** y apretamos D de nuevo pero ahora luego que aparece el nuevo campo sera **db**, vamos alli y apretando D nuevamente hasta cambiar a **dd**.

Debería quedar así ya que edad y nota son dwords.

```

00000000 ; Ins/Del : create/delete structure
00000000 ; D/A/*   : create structure member (data/ascii/a
00000000 ; N       : rename structure or structure member
00000000 ; U       : delete structure member
00000000 ; -----
00000000
00000000 mia          struc ; (sizeof=0x9)
00000000 inicial        db ?
00000001 edad          dd ?
00000005 nota          dd ?
00000009 mia          ends
00000009

```

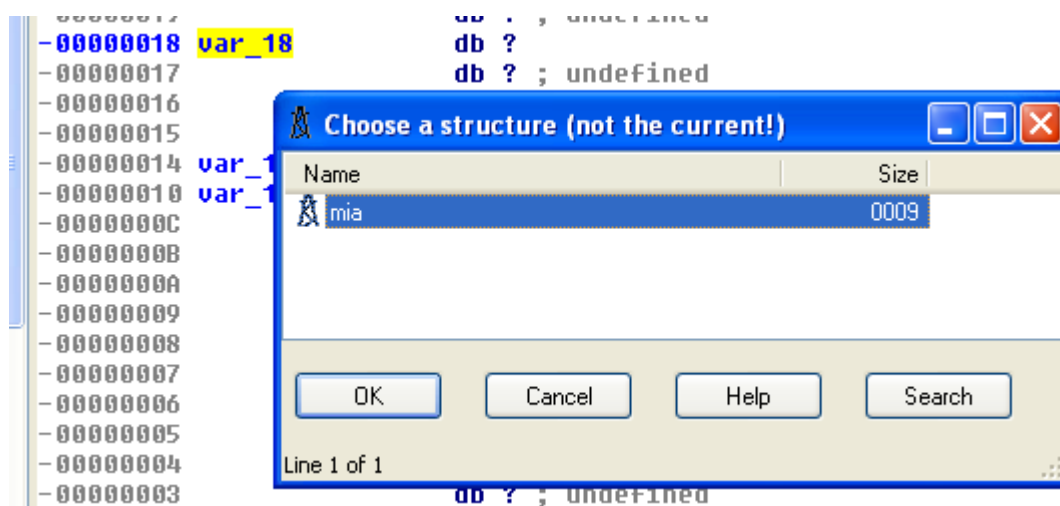
Ya tenemos la estructura creada ahora vayamos a nuestras variables.

```

-00000017          dd ? ; undefined
-00000018 var_18    db ?
-00000017          db ? ; undefined
-00000016          db ? ; undefined
-00000015          db ? ; undefined
-00000014 var_14    dd ?
-00000010 var_10    dd ?
-0000000C          db ? ; undefined
-0000000B          db ? ; undefined
-0000000A          db ? ; undefined
-00000009          db ? ; undefined
-00000008          db ? ; undefined
-00000007          db ? ; undefined
-00000006          db ? ; undefined
-00000005          db ? ; undefined
-00000004          db ? ; undefined
-00000003          db ? ; undefined
-00000002          db ? ; undefined
-00000001          db ? ; undefined
+00000000 s        db 4 dup(?)
+00000004 r        db 4 dup(?)
+00000008

```

Marcamos la variable superior y apretamos ALT mas Q.



Elegimos la estructura mia que esta allí y apretamos OK.

```

00000019          db ? ; undefined
00000018 var_18      mia ?
0000000F          db ? ; undefined
0000000E          00000000 ; Ins/Del : create/delete structure
0000000D          00000000 ; D/A/* : create structure member (dat
0000000C          00000000 ; N : rename structure or structur
0000000B          00000000 ; U : delete structure member
0000000A          00000000 ; -----
00000009          00000000
00000008          00000000 mia          struc ; (sizeof=0x9)
00000007          00000000 inicial      db ?
00000006          00000001 edad        dd ?
00000005          00000005 nota        dd ?
00000004          00000009 mia          ends
00000003          00000009
00000002          db ? ; undefined
00000001          db ? ; undefined
00000000          db 4 dup(?)

```

Listo vemos que al lado de **var_18** donde dice el tipo de datos, ahora dice **mia** y que si ponemos el mouse sobre **var_18** nos muestra como es la estructura que tiene 9 bytes de largo ya que tiene un char de un byte y dos dwords de 4 bytes cada uno.

Ahora renombramos **var_18** con el nombre de nuestra instancia de la estructura, la llamamos **persona** como en el código fuente.

```

-00000019          db ? ; undefined
-00000018 persona  mia ?
-0000000F          db ? ; undefined
-0000000E          db ? ; undefined
-0000000D          db ? ; undefined
-0000000C          db ? ; undefined
-0000000B          db ? ; undefined
-0000000A          db ? ; undefined
-00000009          db ? ; undefined
-00000008          db ? ; undefined
-00000007          db ? ; undefined
-00000006          db ? ; undefined
-00000005          db ? ; undefined
-00000004          db ? ; undefined
-00000003          db ? ; undefined
-00000002          db ? ; undefined
-00000001          db ? ; undefined
+00000000          s          db 4 dup(?)
+00000004          r          db 4 dup(?)
.00000000

```

Si vemos el código vemos que para que sea perfecto hemos cometido una falla.

```

004012C6 persona= mia ptr -18h
004012C6
004012C6 push ebp
004012C7 mov ebp, esp
004012C9 sub esp, 28h
004012CC mov [ebp+persona.inicial], 4Ah
004012D0 mov [ebp+persona.edad+3], 14h
004012D7 mov eax, 40F00000h
004012DC mov [ebp+persona.nota+3], eax
004012DF mov eax, [ebp+persona.edad+3]
004012E2 mov [esp+4], eax
004012E6 mov dword ptr [esp], offset aLaEdadEsD ;
004012ED call printf
004012F2 leave
004012F3 retn
004012F3 sub_4012C6 endp
004012F3

```

Vemos la instancia **persona** del tipo mia bien creada pero el indice vemos que no guarda en **persona.edad** el valor sino en **persona.edad + 3** y esto es por lo siguiente, a veces hay que tener cuidado pues las variables originales no estaban pegadas como nosotros armamos la estructura si las vemos

```


-00000019          dd ? ; undefined
-00000018 var_18    db ?
-00000017          db ? ; undefined
-00000016          db ? ; undefined
-00000015          db ? ; undefined
-00000014 var_14    dd ?
-00000010 var_10    dd ?
-0000000C          db ? ; undefined
-0000000B          db ? ; undefined
-0000000A          db ? ; undefined
-00000009          db ? ; undefined
-00000008          db ? ; undefined
-00000007          db ? ; undefined
-00000006          db ? ; undefined
-00000005          db ? ; undefined
-00000004          db ? ; undefined
-00000003          db ? ; undefined
-00000002          db ? ; undefined
-00000001          db ? ; undefined
+00000000 s        db 4 dup(?)
+00000004 r        db 4 dup(?)
+00000008

```

Vemos que luego de la primera variable que era un **byte** y que es el primer campo hay tres bytes sin usar y luego si el segundo campo, así que si nosotros creamos la estructura acorde a esto debemos meter 3 bytes de relleno, que aunque no usemos, mantendrán los indices en forma correcta, volvamos a la definición de la estructura mia.


La borramos y la creamos nuevamente ahora luego del primer campo **inicial** que es un byte agregamos tres campos mas de byte con cualquier nombre y luego los otros dos campos dwords **edad** y **nota**.

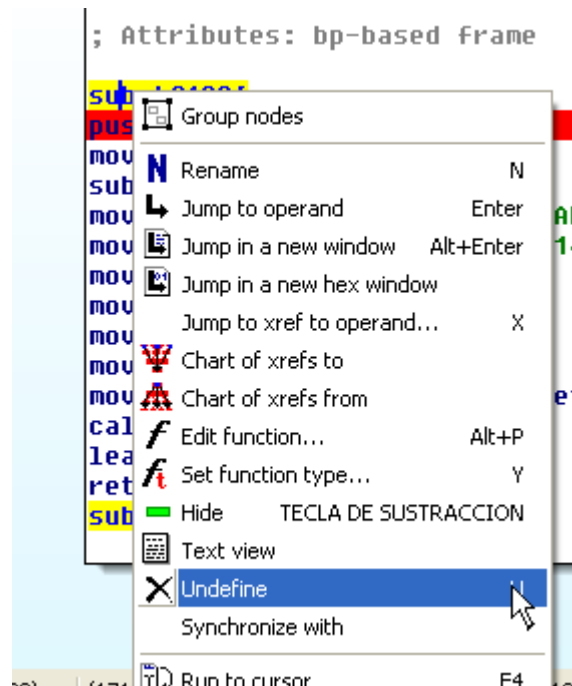
```
00000000
00000000 mia          struc ; (sizeof=0xC)
00000000 inicial      db ?
00000001 b            db ?
00000002 c            db ?
00000003 d            db ?
00000004 edad         dd ?
00000008 nota         dd ?
0000000C mia          ends
AAAAAAAA
```



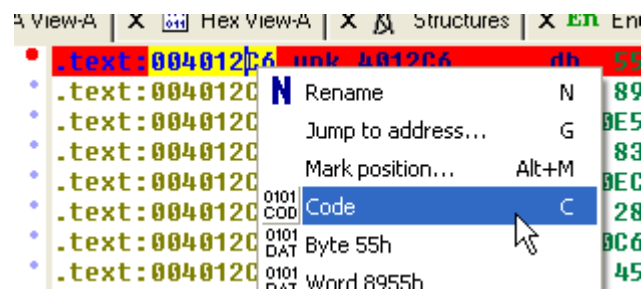
Volvemos a la definición de las variables si no nos aparece ninguna variable, lo cual es lo mas posible y para que quede el código como original, en la funcion nuestra, apretamos click derecho UNDEFINE

```
sub_4012C6 proc near
push    ebp
mov     ebp, esp
sub     esp, 28h
mov     byte ptr [ebp-18h], 4Ah
mov     dword ptr [ebp-14h], 14h
mov     eax, 40F00000h
mov     [ebp-10h], eax
mov     eax, [ebp-14h]
mov     [esp+4], eax
mov     dword ptr [esp], offset aLaEdadEsD ; "La edad es %d"
call    printf
leave
retn
sub_4012C6 endp
```

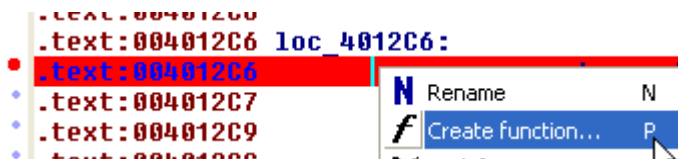




Después en la misma línea click derecho CODE



y al final click derecho CREATE FUNCION



Este es un truco que siempre funciona y restaura una función a como era originalmente sin tener que cargar todo de nuevo ni reiniciar si nos mandamos alguna macana.

```

sub_4012C6 proc near

var_18= byte ptr -18h
var_14= dword ptr -14h
var_10= dword ptr -10h

push    ebp
mov     ebp, esp
sub     esp, 28h
mov     [ebp+var_18], 4Ah
mov     [ebp+var_14], 14h
mov     eax, 40F00000h
mov     [ebp+var_10], eax
mov     eax, [ebp+var_14]
mov     [esp+4], eax
mov     dword ptr [esp], offset aLaEdadEsD ; "La edad es %d"
call    printf
leave

```

Apretando la barra espaciadora se cambia a modo gráfico, ahora vamos a las variables y le asignamos a **var_18** con ALT mas Q la estructura mia.

```

-0000001A          db ? ; undefined
-00000019          db ? ; undefined
-00000018  var_18   mia ?
-0000000C          db ? ; undefined
-0000000B          db ? ; undefined
-0000000A          db ? ; undefined
-00000009          db ? ; undefined
-00000008          db ? ; undefined
-00000007          db ? ; undefined
-00000006          db ? ; undefined
-00000005          db ? ; undefined
-00000004          db ? ; undefined
-00000003          db ? ; undefined
-00000002          db ? ; undefined
-00000001          db ? ; undefined
+00000000  s       db 4 dup(?)
+00000004  r       db 4 dup(?)
+00000008
+00000008 ; end of stack variables

```

Ahora veamos el código, renombrando **var_18** como **persona**.

```

sub_4012C6 proc near

persona= mia ptr -18h

push    ebp
mov     ebp, esp
sub     esp, 28h
mov     [ebp+persona.inicial], 4Ah
mov     [ebp+persona.edad], 14h
mov     eax, 40F00000h
mov     [ebp+persona.nota], eax
mov     eax, [ebp+persona.edad]
mov     [esp+4], eax
mov     dword ptr [esp], offset aLaEdadEsD
call    printf
leave
retn

```

Vemos que los índices quedaron correctos y inicializa correctamente **persona.inicial** con 4ah que podemos hacer click derecho y cambiarlo al carácter **J**, luego **persona.edad** con **14h** y **persona.nota** con el valor float que corresponde a **7.5** que también cambiaremos marcándolo y yendo a EDIT-OPERAND TYPE-NUMBR-FLOATING POINT.

```

; Attributes: bp-based frame

sub_4012C6 proc near

persona= mia ptr -18h

push    ebp
mov     ebp, esp
sub     esp, 28h
mov     [ebp+persona.inicial], 'J'
mov     [ebp+persona.edad], 14h
mov     eax, 7.5
mov     [ebp+persona.nota], eax
mov     eax, [ebp+persona.edad]
mov     [esp+4], eax
mov     dword ptr [esp], offset aLaEdadEsD ; "La edad es
call    printf
leave
retn
sub 4012C6 endp

```

Ahora si que quedo perfecto jeje.

No quiero meter mil ejemplos todos juntos en una sola parte para ir de a poquito con este tema en la parte siguiente veremos mas sobre estructuras y ejemplos mas complicados pero quiero que digieran bien antes esta parte para luego continuar sin atorarse.

Hasta la parte siguiente:
Ricardo Narvaja