Curso C Reversing
Solución Ejercicio 12 (by asOlOt)

Como es menester, abrimos con el IDA el ejemplo que Ricardo nos ha preparado, en este ejemplo se trata de trabajar con punteros (la verdadera piedra de toque del Lenguaje C).

```
:004012B0        call    ___chkstk
:004012B5        call    ___main
:004012BA        mov     eax, 40A00000h
:004012BF        mov     [ebp+var_4], eax
:004012C2        mov     [esp+28h+var_28], 4
:004012C9        call    malloc
:004012CE        mov     [ebp+var_8], eax
:004012D1        mov     edx, [ebp+var_8]
:004012D4        mov     eax, 40D66666h
:004012D9        mov     [edx], eax
:004012DB        mov     [esp+28h+var_28], 4
:004012E2        call    malloc
:004012E7        mov     [ebp+var_C], eax
:004012EA        mov     edx, [ebp+var_C]
:004012ED        mov     eax, [ebp+var_8]
:004012F0        fld     dword ptr [eax]
:004012F2        fadd    [ebp+var_4]
:004012F5        fstp    dword ptr [edx]
:004012F7        mov     eax, [ebp+var_C]
```

Lo primero que vemos, que vamos a trabajar en la misma función Main, y que vamos a trabajar con la FPU (ya nos advierte Ricardo que vamos a trabajar con Floats).

Para saber a qué números floats corresponden podemos correr el programa y ver en la FPU que valores cogen, veremos que son: 5.0 y 6.7

Visto los valores, podemos renombrar las variables, Tambien notamos que reservamos memoria con el malloc, para almacenar un total de 4 bytes.

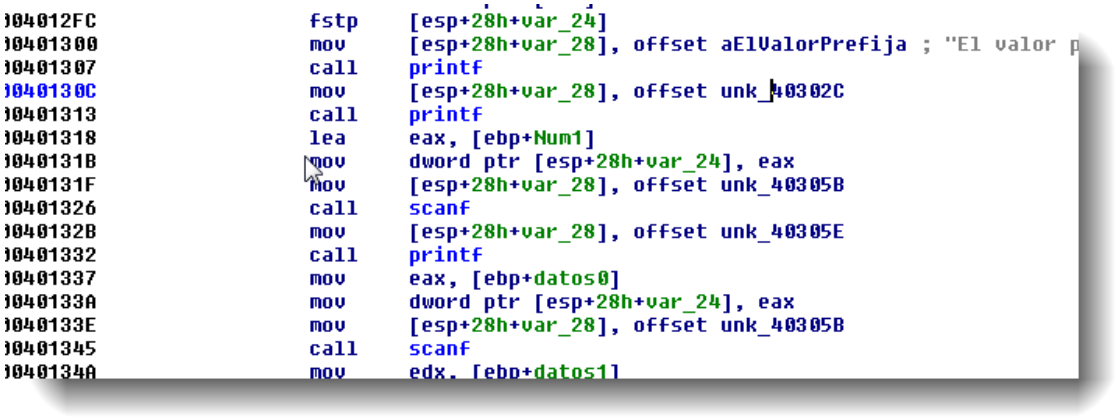
```
004012B5        call    ___main
004012BA        mov     eax, 5.0
004012BF        mov     [ebp+Num1], eax
004012C2        mov     [esp+28h+var_28], 4
004012C9        call    malloc
004012CE        mov     [ebp+datos0], eax
004012D1        mov     edx, [ebp+datos0]
004012D4        mov     eax, 6.7
004012D9        mov     [edx], eax
004012DB        mov     [esp+28h+var_28], 4
004012E2        call    malloc
004012E7        mov     [ebp+datos1], eax
004012EA        mov     edx, [ebp+datos1]
004012ED        mov     eax, [ebp+datos0]
004012F0        fld     dword ptr [eax]
004012F2        fadd    [ebp+Num1]
004012F5        fstp    dword ptr [edx]
```

Pasando a C este pequeño trozo quedaría a asi:

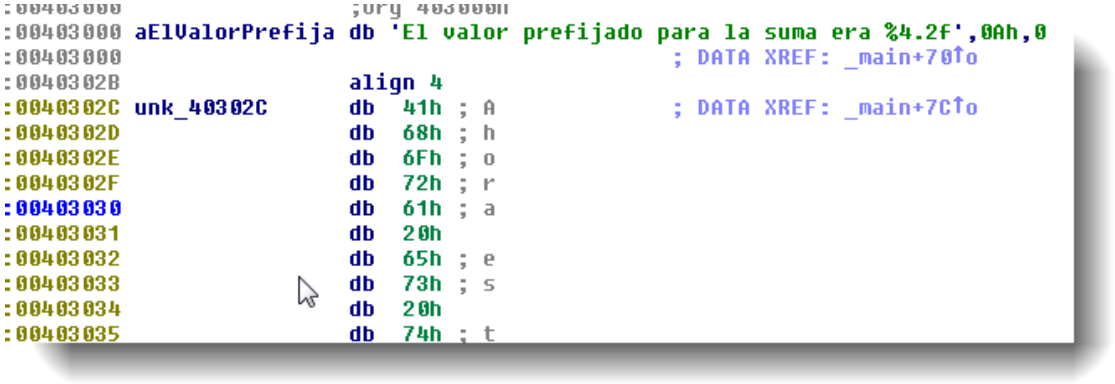
```
main()
{
        float Num1;
        float* datos0;
         float* datos1;

        num1 = 5.0;
        datos0 = (float *) malloc(sizeof(float));

        *datos0 = 6.7;
        datos1 = (float *) malloc(sizeof(float));

        *datos1 = *datos0 + num1;
}
```

Solamente hemos asignado al campo datos0 el valor de 6.7, y luego lo hemos sumado al número 5.0 guardando el resultado en la memoria reservada apuntada por datos1.

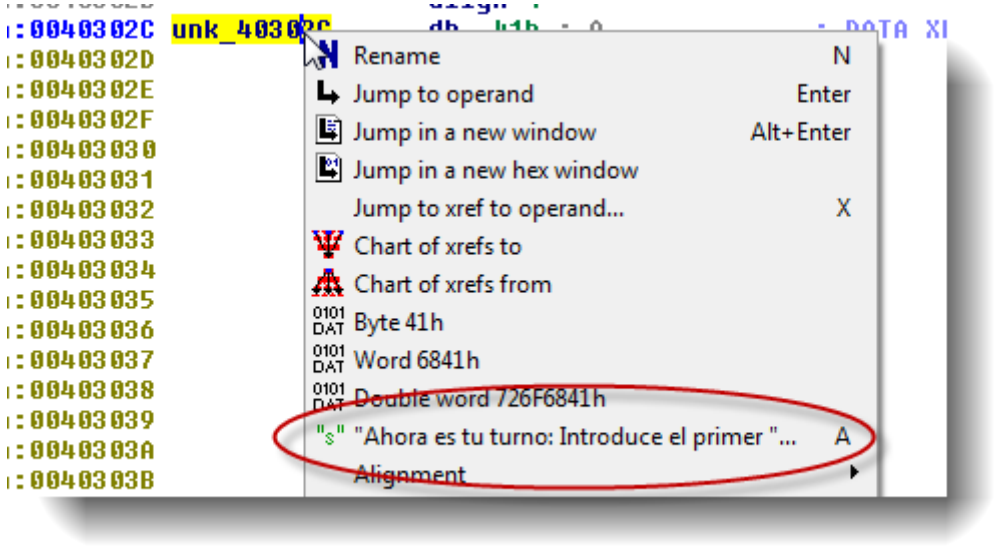
Luego vemos como sigue el código, en este caso imprime el resultado de la suma:



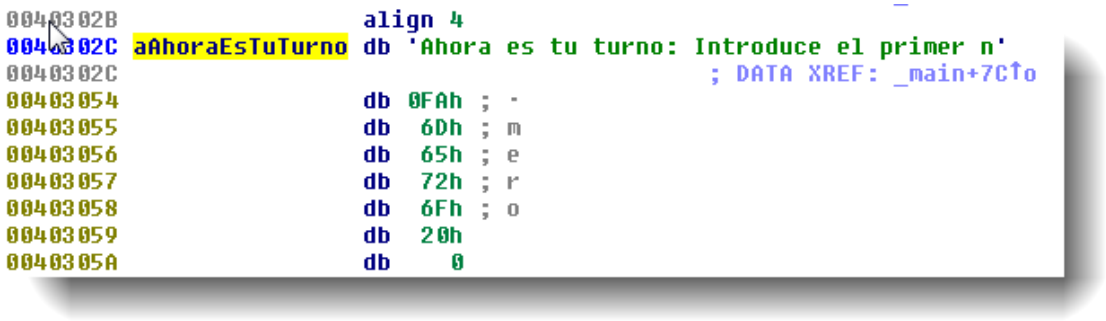
La cadena que se imprimirá la vemos en la siguiente captura:



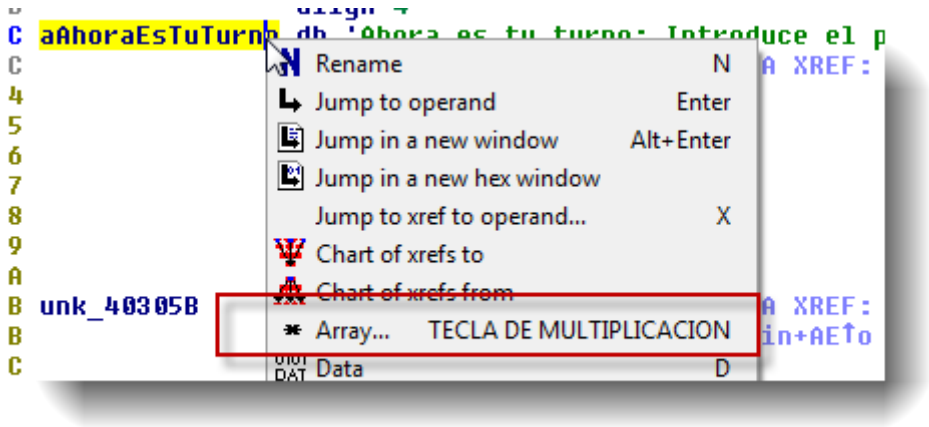
La vemos claramente “El valor prefijado…..”, lo vemos bien pero en los siguiente printf no ocurre lo mismo, por ejemplo en lA dirección 04032C, no vemos la cadena alineada correctamente, esto lo solucionamos clickando el botón derecho encima de la dirección deseada y vemos la cadena formada por los bytes que siguen, si clickamos se nos presenta de una forma más legible:



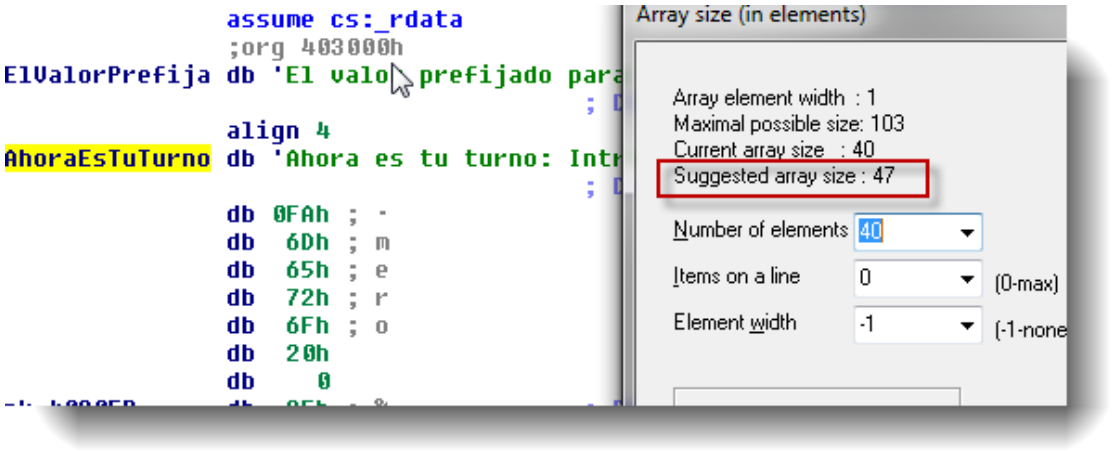
Y nos queda:



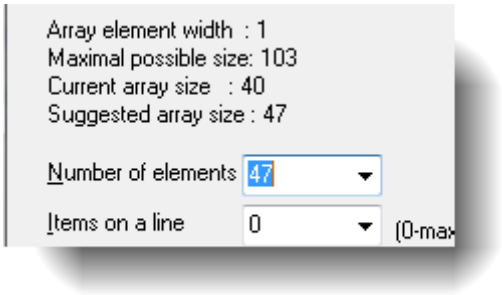
Opps, hay un carácter (0FAh), que no lo reconoce como parte de la frase, pero sabemos que si lo es, de hecho se han quedado 7 bytes fuera de la frase (incluido el null terminated). Para arreglar esto clickamos encima de la dirección y seleccionamos el array:



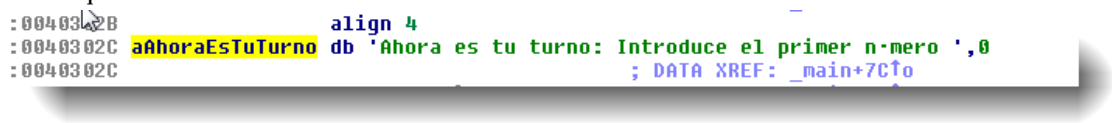
Ya nos dice que el tamaño aconsejable son 47 bytes:



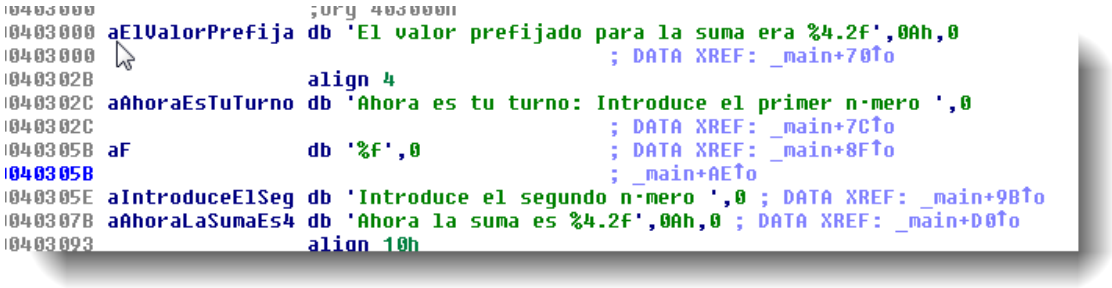
Como el tamaño actual son 40 bytes y nos faltaban 7 por incluir, hacemos caso a lo que nos sugiere y le metemos 47 bytes de tamaño:



Y nos queda:



Hacemos lo mismo con las cadenas que vemos que le siguen:



Para la cadena %f, nos colocamos encima y apretamos la tecla ‘a’, y ya se nos queda en el formato correcto.
Ahora nos es más fácil seguir el código anterior:

```
004012FC                fstp    [esp+28h+var_24]
00401300                mov     [esp+28h+var_28], offset aElValorPrefija ; "El valor
00401307                call    printf
0040130C                mov     [esp+28h+var_28], offset aAhoraEsTuTurno ; "Ahora es
00401313                call    printf
00401318                lea     eax, [ebp+Num1]
0040131B                mov     dword ptr [esp+28h+var_24], eax
0040131F                mov     [esp+28h+var_28], offset aF ; "%f"
00401326                call    scanf
0040132B                mov     [esp+28h+var_28], offset aIntroduceElSeg ; "Introduc
00401332                call    printf
00401337                mov     eax, [ebp+datos0]
0040133A                mov     dword ptr [esp+28h+var_24], eax
0040133E                mov     [esp+28h+var_28], offset aF ; "%f"
00401345                call    scanf
0040134A                mov     edx, [ebp+datos1]
0040134D                mov     eax, [ebp+datos0]
00401350                fld     dword ptr [eax]
00401352                fadd    [ebp+Num1]
00401355                fstp    dword ptr [edx]
```

Su correspondencia en C:

printf("El valor prefijado para la suma era %4.2f\n", *datos1);

printf("Ahora es tu turno: Introduce el primer n·mero ");

scanf("%f", &num1);

printf("Introduce el segundo n·mero ");

scanf("%f", datos0);

****datos1= *datos0 + num1;***

(Para introducir el caractar 250 (0fah), teclamos el numero 250 con la tecla Alt apretada, cuando soltemos la tecla Alt, tendremos tipeado el carácter ·).

Nos pide que introduzcamos los dos números y la suma de ambos la deja en la memoria apuntada en datos1.

Ya se finaliza el código, sacando la suma actual por la pantalla, y liberando la memoria reservada.

```
0401355                 fstp    dword ptr [edx]
0401357                 mov     eax, [ebp+datos1]
040135A                 fld     dword ptr [eax]
040135C                 fstp    [esp+28h+var_24]
0401360                 mov     [esp+28h+var_28], offset aAhoraLaSumaEs4 ; "Aho
0401367                 call    printf
040136C                 mov     eax, [ebp+datos0]
040136F                 mov     [esp+28h+var_28], eax
0401372                 call    free
0401377                 mov     eax, [ebp+datos1]
040137A                 mov     [esp+28h+var_28], eax
040137D                 call    free
0401382                 call    getchar
0401387                 call    getchar
040138C                 call    getchar
0401391                 leave
0401392                 retn
0401392 _main           endp
```

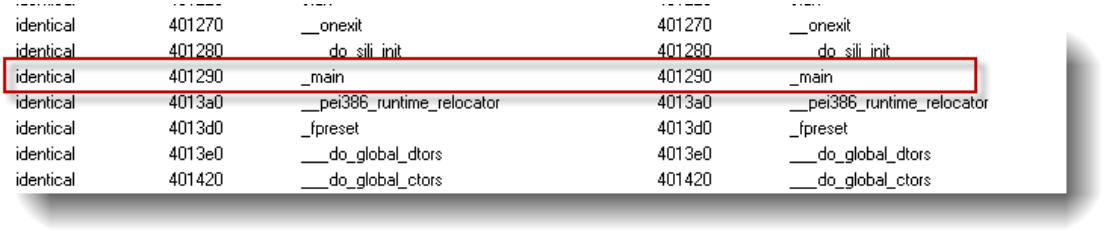
Su correspondencia en C:

```
        printf("Ahora la suma es %4.2f\n", *datos1);

        free(datos0);
        free(datos1);

        getchar();
        getchar();
        getchar();
```

Pues nada la función completa seria:

```
main()
{
        float num1;
        float* datos0;
         float* datos1;

        num1 = 5.0;
        datos0 = (float *)malloc(sizeof(float));

        *datos0 = 6.7;
        datos1 = (float *)malloc(sizeof(float));

        *datos1 = *datos0 + num1;

        printf("El valor prefijado para la suma era %4.2f\n", *datos1 );
        printf("Ahora es tu turno: Introduce el primer n·mero ");

        scanf("%f", &num1);

        printf("Introduce el segundo n·mero ");

        scanf("%f", datos0);

        *datos1= *datos0 + num1;
        printf("Ahora la suma es %4.2f\n", *datos1);

        free(datos0);
        free(datos1);

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

Compilamos y probamos con el turbodiff, haber si tenemos suerte…



identical	401270	__onexit	401270	__onexit
identical	401280	__do_sili_init	401280	__do_sili_init
identical	401290	_main	401290	_main
identical	4013a0	__pei386_runtime_relocator	4013a0	__pei386_runtime_relocator
identical	4013d0	_fpreset	4013d0	_fpreset
identical	4013e0	___do_global_dtors	4013e0	___do_global_dtors
identical	401420	___do_global_ctors	401420	___do_global_ctors

Pues esta vez hemos tenido suerte.

Hasta la próxima.

asOlOt, 2010 CLS