


# Curso C Reversing

## Solucion Ejercicio 11 (by as0l0t)

Vemos el comienzo del programa con el IDA:



```
004012AA mov     [ebp+var_C], eax
004012AD mov     eax, [ebp+var_C]
004012B0 call    __chkstk
004012B5 call    __main
004012BA mov     [esp+18h+var_18], offset aIngresoElValor ; "Ingreso el valor de a:"
004012C1 call    printf
004012C6 lea     eax, [ebp+var_4]
004012C9 mov     [esp+18h+var_14], eax
004012CD mov     [esp+18h+var_18], offset aD ; "%d"
004012D4 call    scanf
004012D9 mov     [esp+18h+var_18], offset aIngresoElVal_0 ; "Ingreso el valor de b:"
004012E0 call    printf
004012E5 lea     eax, [ebp+var_8]
004012E8 mov     [esp+18h+var_14], eax
004012EC mov     [esp+18h+var_18], offset aD ; "%d"
004012F3 call    scanf
004012F8 mov     eax, [ebp+var_8]
004012FD mov     [esp+18h+var_14], eax

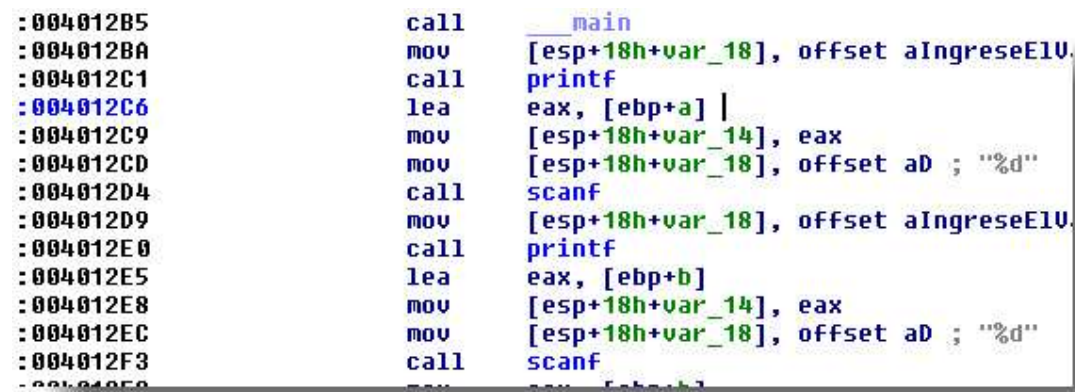
;org 403000h
030000 aIngresoElValor db 'Ingreso el valor de a:',0Ah,0 ; DATA XREF:
030010 aD          db '%d',0 ; DATA XREF: _main+3Df
030018          ; DATA XREF: _main+5Cf
03001B aIngresoElVal_0 db 'Ingreso el valor de b:',0Ah,0 ; DATA XREF:
```

Esta vez en el mismo Main tenemos la recogida de las variables, como ya se ha visto en el curso este trozo seria:

```
printf("Ingreso el valor de a:\n");
scanf("%d", &a);

printf("Ingreso el valor de b:\n");
scanf("%d", &b);
```

En var\_4 se guardaría la **a** y en var\_8 la **b**, las renombramos:



```
:004012B5 call    __main
:004012BA mov     [esp+18h+var_18], offset aIngresoElV
:004012C1 call    printf
:004012C6 lea     eax, [ebp+a]
:004012C9 mov     [esp+18h+var_14], eax
:004012CD mov     [esp+18h+var_18], offset aD ; "%d"
:004012D4 call    scanf
:004012D9 mov     [esp+18h+var_18], offset aIngresoElV
:004012E0 call    printf
:004012E5 lea     eax, [ebp+b]
:004012E8 mov     [esp+18h+var_14], eax
:004012EC mov     [esp+18h+var_18], offset aD ; "%d"
:004012F3 call    scanf
:004012FD mov     [esp+18h+var_14], eax
```

Seguimos con el código:

```

:004012F3      call     scanf
:004012F8      mov     eax, [ebp+b]
:004012FB      mov     [esp+18h+var_14], eax
:004012FF      mov     eax, [ebp+a]
:00401302      mov     [esp+18h+var_18], eax
:00401305      call    sub_401328
:0040130A      mov     eax, [ebp+b]
:0040130D      mov     [esp+18h+var_14], eax
:00401311      mov     eax, [ebp+a]
:00401314      mov     [esp+18h+var_18], eax
:00401317      call    sub_40134C
:0040131C      call    _getch
:00401321      mov     eax, 0
:00401326      leave
:00401327      retn
:00401327      main
:00401327      endp

```

Vemos que tanto la a como la b, sirven de argumento a dos funciones (o procedimientos, jeje).

### Primera función:

Nos Centramos en la primera:

```

:00401328  sub_401328  proc near          ; CODE XREF: _main+75↑p
:00401328
:00401328  var_18      = dword ptr -18h
:00401328  var_14      = dword ptr -14h
:00401328  var_4       = dword ptr -4
:00401328  arg_0       = dword ptr 8
:00401328  arg_4       = dword ptr 0Ch
:00401328
:00401328      push     ebp
:00401329      mov      ebp, esp
:0040132B      sub      esp, 18h
:0040132E      mov      eax, [ebp+arg_4]
:00401331      add      eax, [ebp+arg_0]
:00401334      mov      [ebp+var_4], eax
:00401337      mov      eax, [ebp+var_4]
:0040133A      mov      [esp+18h+var_14], eax
:0040133E      mov      [esp+18h+var_18], offset aElValorDeLaSuma ; "El valor de la suma es %d:
:00401345      call    printf
:0040134A      leave
:0040134B      retn
:0040134B  sub_401328  endp

```

Es un función cortita, y como vemos al final nos muestra la cadena:

“El valor de la suma es %d”

Si nos fijamos tenemos los arg\_0 y arg\_4 que son los a y b que hemos puesto en la pila antes de la llamada. Por lo que es obvio que esta función se encarga de sumar a + b y mostrar el resultado en la pantalla.

En var\_4 se guarda el resultado de la suma, así que también la renombraremos:

```

00401328 ; int __cdecl Sumar(int a, int b)
00401328 Sumar      proc near          ; CODE XREF: _main+751p
00401328
00401328 var_18      = dword ptr -18h
00401328 var_14      = dword ptr -14h
00401328 suma       = dword ptr -4
00401328 a          = dword ptr 8
00401328 b          = dword ptr 0Ch
00401328
00401328      push    ebp
00401329      mov     ebp, esp
0040132B      sub     esp, 18h
0040132E      mov     eax, [ebp+b]
00401331      add     eax, [ebp+a]
00401334      mov     [ebp+suma], eax
00401337      mov     eax, [ebp+suma]
0040133A      mov     [esp+18h+var_14], eax
0040133E      mov     [esp+18h+var_18], offset aElValorDeLaSuma ;
00401345      call    printf
0040134A      leave
0040134B      retn
0040134B Sumar      endp

```

Como se ve también hemos puesto el tipo de la función (con set function type), realmente no es necesario todo esto para trasladar el código a C, es bastante simple, pero de lo que se trata es de practicar con el IDA, y eso es lo que hacemos, así que renombraremos siempre las variables y las funciones para coger soltura. Esta función en C sería:

```

Sumar(int a, int b)
{
    int suma;
    suma= a + b;
    printf("El valor de la suma es %d:\n\n", suma);
}

```

Observar el doble retorno de carro, tal y como se ve en el valor de la cadena:

```

aElValorDeLaSuma db 'El valor de la suma es %d:', 0Ah ; D
db 0Ah, 0

```

## Segunda función:

Ahora vamos a por la segunda función:

```

:0040134C
:0040134C sub_40134C      proc near                ; CODE XREF: _main+87↑p
:0040134C
:0040134C var_8           = dword ptr -8
:0040134C arg_0          = dword ptr  8
:0040134C arg_4          = dword ptr  0Ch
:0040134C
:0040134C      push      ebp
:0040134D      mov       ebp, esp
:0040134F      sub       esp, 8
:00401352      mov       eax, [ebp+arg_0]
:00401355      cmp       eax, [ebp+arg_4]
:00401358      jnz       short loc_401368
:0040135A      mov       [esp+8+var_8], offset aSonIguales ; "Son iguales\n\n"
:00401361      call      printf
:00401366      jmp       short locret_40138A
:00401368 ; -----
:00401368
:00401368 loc_401368:      ; CODE XREF: sub_40134C+C↑j
:00401368      mov       eax, [ebp+arg_0]
:0040136B      cmp       eax, [ebp+arg_4]
:0040136E      jle       short loc_40137E
:00401370      mov       [esp+8+var_8], offset aElValorDeAEsMa ; "El valor de a
:00401377      call      printf

```

Como sabemos los arg\_0 y arg\_4 corresponden a la **a** y la **b** respectivamente. Lo primero que hace la función es comparar a y b si no son iguales los vuelve a comparar y salta si **a** no es mayor que **b**.

Si seguimos:

```

:0040136E      jle       short loc_40137E
:00401370      mov       [esp+8+var_8], offset aElValorDeAEsMa ; "E
:00401377      call      printf
:0040137C      jmp       short locret_40138A
:0040137E ; -----
:0040137E
:0040137E loc_40137E:      ; CODE XREF: sub_40134C+22↑j
:0040137E      mov       [esp+8+var_8], offset aElValorDeBEsMa ; "E
:00401385      call      printf
:0040138A
:0040138A locret_40138A:    ; CODE XREF: sub_40134C+1F↑j
:0040138A      ; sub_40134C+30↑j
:0040138A      leave
:0040138B      retn
:0040138B sub_40134C      endp
:0040138B

```

Luego como no queda otra posibilidad, nos escribe que el valor de **b** es mayor que **a**. Sale de la función sin valor de retorno. Antes de pasarla a C la renombramos adecuadamente:

```

:00401340
:0040134C ; int __cdecl Comparar(int a, int b)
:0040134C Comparar      proc near                               ; CODE XREF
:0040134C
:0040134C var_8          = dword ptr -8
:0040134C a              = dword ptr  8
:0040134C b              = dword ptr  0Ch
:0040134C
:0040134C      push      ebp
:0040134D      mov       ebp, esp
:0040134F      sub       esp, 8
:00401352      mov       eax, [ebp+a]
:00401355      cmp       eax, [ebp+b]
:00401358      jnz       short loc_401368
:0040135A      mov       [esp+8+var_8], offset aSon
:00401361      call      printf
:00401366      jmp       short locret_40138A
:00401368 ; -----
:00401368
:00401368 loc_401368:                                           ; CODE XREF
:00401368      mov       eax, [ebp+a]
:0040136B      cmp       eax, [ebp+b]
:0040136E      jle       short loc_40137E
:00401370      mov       [esp+8+var_8], offset aElU
:00401377      call      printf
:0040137C      jmp       short locret_40138A
:0040137E ; -----
:0040137E

```

Y su código en C:

```

Comparar(int a, int b)
{
    if( a == b)
        printf("Son iguales\n\n");
    else if( a > b)
        printf("El valor de a es mayor que el de b\n\n");
    else
        printf("El valor de b es mayor que el de a\n\n");
}

```

### Finalizando el programa:

Como ya sabemos todo, creamos el programa completo, como bien nos explica Ricardo en la parte 11 del curso, es mejor colocar las funciones antes de ser llamadas, (declararlas aun sería mejor), así que ponemos las funciones antes del main():

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

```
Sumar(int a, int b)
```

```
{
    int suma;
    suma= a + b;
    printf("El valor de la suma es %d\n\n", suma);
}
```

```
Comparar(int a, int b)
```

```
{
    if( a == b)
        printf("Son iguales\n\n");
    else if( a > b)
        printf("El valor de a es mayor que el de b\n\n");
    else

```

```

        printf("El valor de b es mayor que el de a\\nn");
    }
    main()
    {
        int a, b;

        printf("Ingrese el valor de a:\\n");
        scanf("%d", &a);

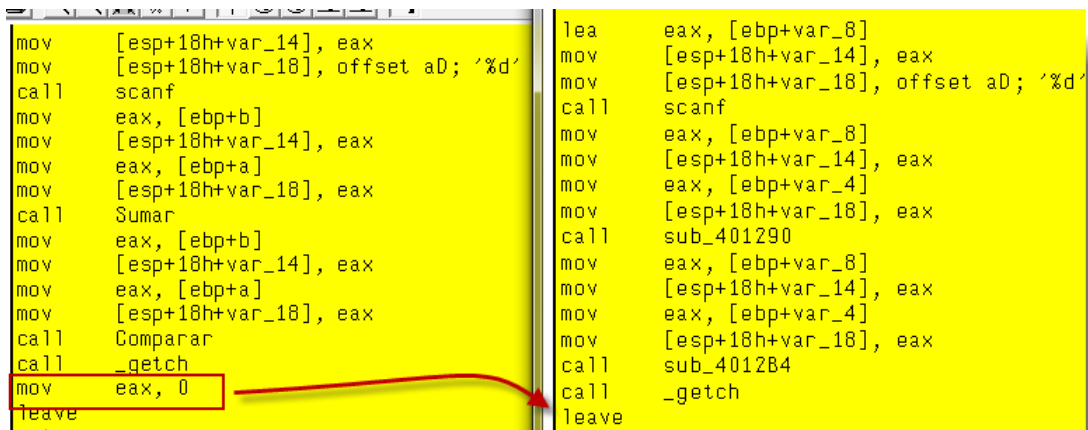
        printf("Ingrese el valor de b:\\n");
        scanf("%d", &b);

        Sumar(a, b);
        Comparar(a, b);

        getch();
    }

```

Lo compilamos y comprobamos con el turbodiff:



mov [esp+18h+var_14], eax	lea eax, [ebp+var_8]
mov [esp+18h+var_18], offset a0; '%d'	mov [esp+18h+var_14], eax
call scanf	mov [esp+18h+var_18], offset a0; '%d'
mov eax, [ebp+b]	call scanf
mov [esp+18h+var_14], eax	mov eax, [ebp+var_8]
mov eax, [ebp+a]	mov [esp+18h+var_14], eax
mov [esp+18h+var_18], eax	mov eax, [ebp+var_4]
call Sumar	mov [esp+18h+var_18], eax
mov eax, [ebp+b]	call sub_401290
mov [esp+18h+var_14], eax	mov eax, [ebp+var_8]
mov eax, [ebp+a]	mov [esp+18h+var_14], eax
mov [esp+18h+var_18], eax	mov eax, [ebp+var_4]
call Comparar	mov [esp+18h+var_18], eax
call _getch	call sub_4012B4
mov eax, 0	call _getch
leave	leave

Valla se nos ha pasado que el main() de Ricardo se coloca `eax = 0` al final, es un Return 0

```

main()
{
    int a, b;

    printf("Ingrese el valor de a:\\n");
    scanf("%d", &a);

    printf("Ingrese el valor de b:\\n");
    scanf("%d", &b);

    Sumar(a, b);
    Comparar(a, b);

    getch();

    return 0;
}

```

Lo volvemos a comparar con el original:

identical	401930	fprintf	401930	fprintf
identical	401940	SetUnhandledExceptionFilter	401940	SetUnhandledExceptionFilter
identical	401950	ExitProcess	401950	ExitProcess
identical	401990	__sjl_init_ctor	401990	__sjl_init_ctor
identical	401290	sub_401290	401328	Sumar
identical	4012b4	sub_4012b4	40134c	Comparar
identical	4010f1	sub_4010f1_undefined	4010f1	sub_4010f1_undefined

Identico.

Saludos...

PD: Hey, vemos que las funciones ,en el programa de ejemplo, están declaradas después del Main(), ejem... habrá que releer la parte 11, sobre la mejor posición de declarar las funciones.... Jeje.

```

11326             leave
11327             retn
11327 _main         endp
11328
11328 ; ===== S U B R O U T I N E =====
11328
11328 ; Attributes: bp-based frame
11328
11328 ; int __cdecl Sumar(int a, int b)
11328 Sumar         proc near                ; CODE XREF
11328
11328 var_18        = dword ptr -18h
11328 var_14        = dword ptr -14h
11328 suma         = dword ptr -4
11328 a            = dword ptr 8
11328 b            = dword ptr 0Ch
11328
11328             push    ebp
11329             mov     ebp, esp
11328             sub     esp, 18h
1132E             mov     eax, [ebp+b]
11331             add     eax, [ebp+a]
11334             mov     [ebp+suma], eax
11327             mov     eax, [ebp+suma]

```