6.4.—Transformaciones básicas de datos

Los datos formateados correctamente, al igual que el código, son muy importantes para comprender el comportamiento de un programa. Como ya sabemos IDA toma la información de varias fuentes y utiliza algunos algoritmos para poder determinar la forma más apropiada para formatear los datos en un desensamblado. Algunos ejemplos servirán para enseñarnos como son seleccionados los formatos de datos.

1. El tipo de datos y su tamaño pueden inferir en la forma en que se utilizarán los registros. Una instrucción preparada para cargar un registro con 32 bit de la memoria infiere que dicha ubicación de memoria tenga un tipo de dato de 4-byte, sin embargo no podemos ser capaces de saber si es un entero de 4-byte o un puntero de 4-byte.

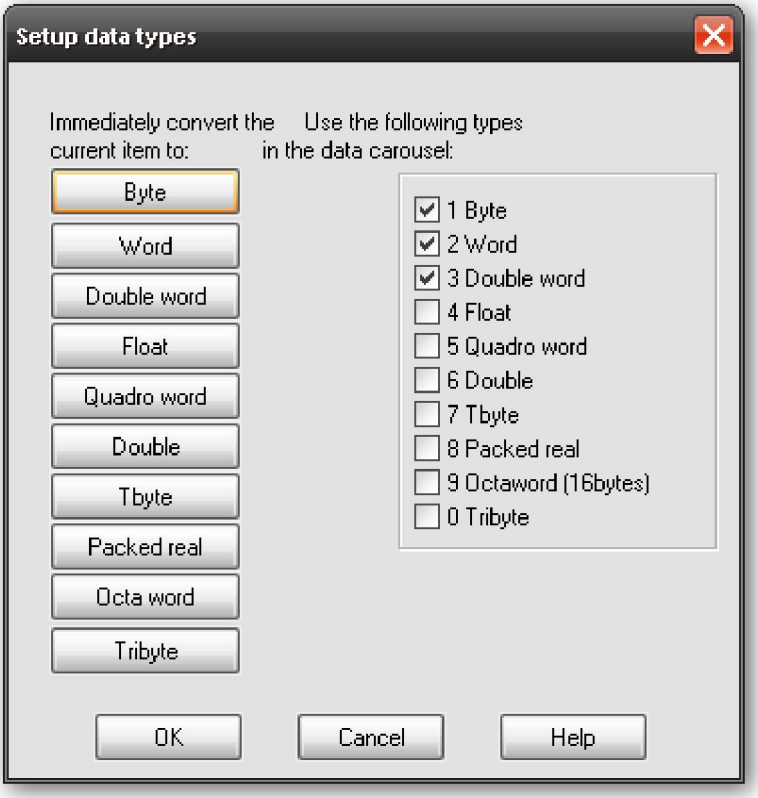
2. Los prototipos de función pueden utilizarse para asignar tipos de datos a los parámetros de función. IDA proporciona una gran librería de prototipos de función para este propósito. El análisis se realiza en los parámetros pasados a las funciones en un intento de ligar un parámetro con la ubicación de memoria. Si se encuentra tal relación, entonces se puede aplicar un tipo de dato asociado a dicha ubicación. Veamos, consideremos una función con un solo parámetro el cual es un puntero a una **CRITICAL_SECTION**, tipo de dato **Windows API**. Si IDA puede determinar la dirección pasada en la llamada a esta función. Entonces IDA puede habilitar esta dirección como un objeto de **CRITICAL_SECTION**.

3. El análisis de una secuencia de bytes nos puede revelar los tipos de datos probables. Esto es precisamente lo que sucede cuando se examina un binario para saber su contenido de cadenas. Cuando se encuentra una larga secuencia de caracteres ASCII, es lógico asumir que representa un conjunto de caracteres.

En las siguientes secciones hablaremos sobre ciertas transformaciones básicas que pueden ejecutarse en los desensamblados.

6.4.1.— Especificar el tamaño de los datos

Una forma simple de modificar un dato es ajustar su tamaño. IDA nos proporciona un número distinto de especificaciones para los datos, **tamaño/tipo**. Las especificaciones más normales que utilizaremos son **db**, **dw** y **dd**, las cuales representan respectivamente **1, 2 y 4 byte** de datos. La primera forma para poder cambiar el tamaño de un elemento dato es realizando la acción **Options > Setup Data Types**, y se nos mostrará el diálogo, figura abajo.



En este diálogo tenemos dos partes distintas. En el lado izquierdo vemos una columna de botones utilizados para cambiar directamente el tamaño del elemento actualmente seleccionado. En el lado derecho vemos una columna de checkbox utilizados para configurar lo que IDA llama **data carousel**. Observa que a cada botón de la izquierda le corresponde un checkbox de la derecha. El data carousel no es más que una lista iterativa de tipos de dato la cual sólo contiene los tipos de los checkbox seleccionados. Modificar el contenido del data carousel no tiene ningún efecto inmediato en la vista de IDA. Lo que ocurre es cada tipo de dato seleccionado en el data carousel se listará en un menú de contexto que aparecerá haciendo click derecho en el elemento dato seleccionado. De esta forma, es fácil formatear un dato con algún tipo listado en el data carousel. Si miramos la figura arriba, y observamos los tipos seleccionados cuando hagamos click derecho en el dato nos dará la opción de formatearlo como dato **byte, Word o double-word.**

El nombre de **data carousel** viene determinado por el atajo asociado a él, **D**. Cuando pulsas D, el elemento de la dirección seleccionada es reformateado con el siguiente tipo de dato de la lista del data carousel. Con los tres elementos de la lista, especificados previamente, un elemento con formato **db** cambiaría a **dw**, un elemento con formato **dw** cambiaría a **dd** y un elemento con formato **dd** cambiaría a **db** completando el circuito del carrusel (tiovivo). Utilizar el atajo de datos, **D**, en un elemento que no sea dato, sino como código, produce que dicho elemento se formatee como el primer tipo de dato de la lista del carrusel, en este caso **db.**

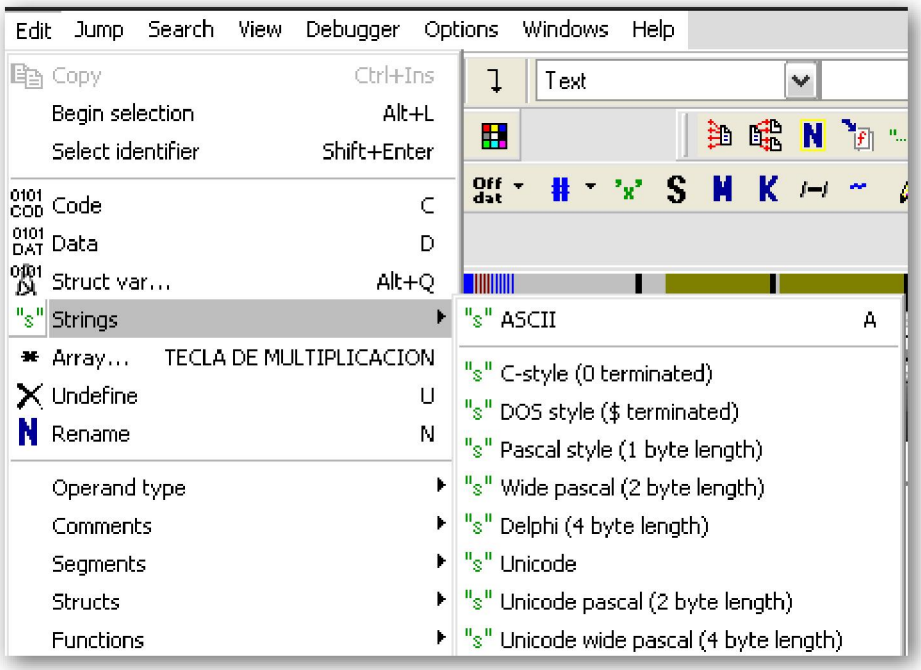
Conmutar los distintos tipos, causa que los elementos de datos se agranden, se encojan o permanezcan con el mismo tamaño. Si el tamaño de un elemento queda igual sólo podremos ver el cambio del elemento por la manera que será formateado el dato. Si por ejemplo, reducimos el tamaño de un elemento de **dd (4 byte)** a **db (1 byte)**, cada byte extra, en este caso **3**, quedan indefinidos. Si incrementamos el tamaño de un elemento,

IDA se quejará si los byte a continuación del elemento están definidos y te preguntará, de forma indirecta, si quieres que IDA interprete el siguiente elemento de manera que pueda expandir el elemento actual. El mensaje que te mostrará en estos casos será, **“Directly convert to data?”** Este mensaje significa que IDA puede interpretar un número suficiente de los elementos subsiguientes con lo cual puede realizar tú solicitud. Por ejemplo, cuando convertimos un dato **byte** (**db**) a un dato **double-word** (**dd**), los tres bytes adicionales deben cogerse para formar el nuevo elemento de dato.

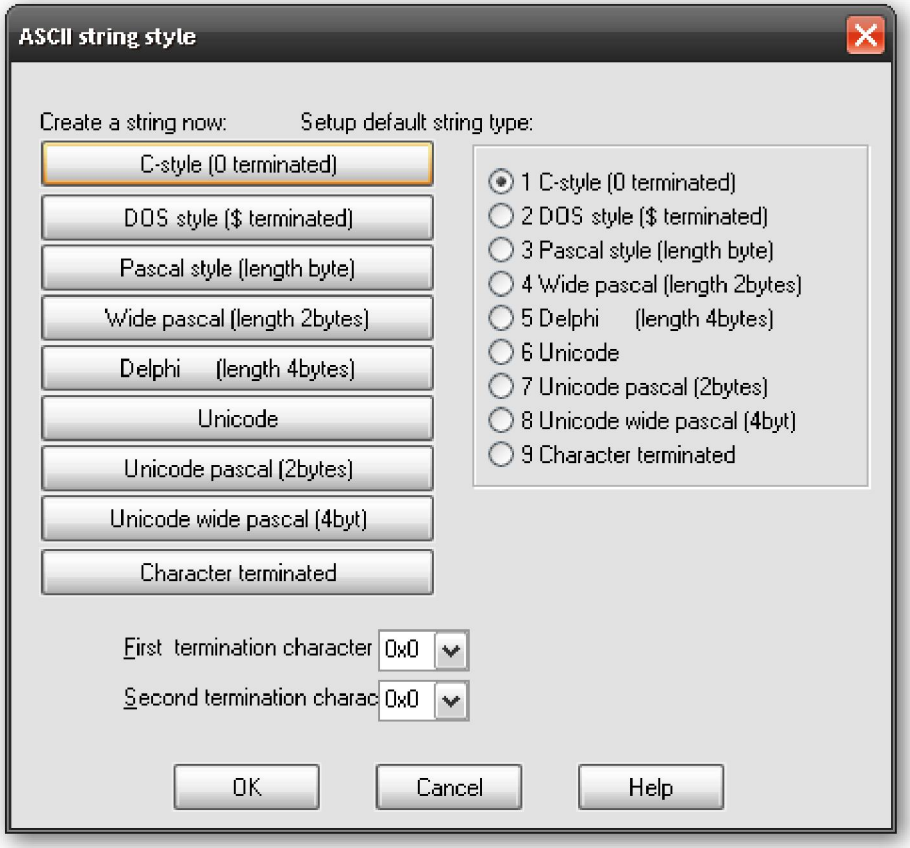
Los tipos de datos y sus tamaños pueden especificarse en cualquier ubicación donde se describe como dato, incluidas las variables de pila. Para cambiar el tamaño de las variables de la pila ya distribuidas, abre la **vista detallada del stack frame** haciendo doble clic en la variable que deseas modificar; entonces cambia el tamaño de la variable como lo harías con cualquier otra variable.

6.4.2.—Trabajar con cadenas (Strings)

Por suerte, IDA reconoce un gran número de formatos de cadena. Por defecto IDA busca y formatea las cadenas con estilo C y terminadas en cero. Para convertir un dato a una cadena, utilizaremos la acción **Edit > String** y especificaremos el estilo de cadena.

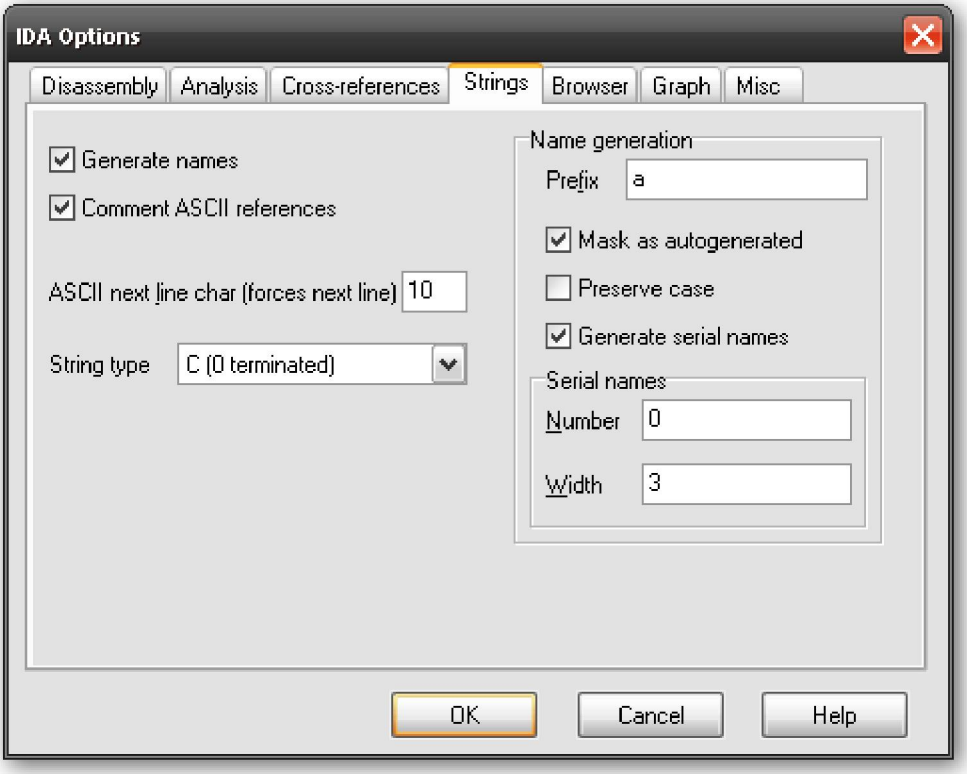


Dos diálogos son los responsables de la configuración de las cadenas de datos. El primero es el mostrado, figura abajo, al cual podemos acceder realizando la acción **Options > ASCII String Style**, sin embargo en esta ocasión ASCII es un término algo inapropiado, ya que comprende más opciones de estilos.



Similar al diálogo de configuración de tipo de datos, los botones de la izquierda se utilizan para crear una cadena, en la ubicación seleccionada, con el estilo especificado. La cadena sólo se creará si el dato de la ubicación actual se conforma con el formato de cadena especificado. Con la opción de cadena, **Character terminated**, al final del diálogo pueden especificarse hasta dos opciones de terminación con carácter. Las opciones de la derecha del diálogo se utilizan para especificar el estilo de cadena por defecto asociado con el atajo **A**.

El segundo diálogo, figura abajo, se utiliza para configurar las operaciones con cadenas, se accede a él con la acción **Options > General dialog**, en donde la pestaña **Strings** nos permite unas opciones de configuración adicionales. Al igual que en el anterior puedes especificar el tipo de la cadena por defecto, utilizando la lista desplegable. Además de distintas opciones para generar su nombre y visualizar la cadena de caracteres, sin influir su tipo. El área **Name generation** a la derecha del diálogo, sólo es visible cuando tienes seleccionada la opción **Generate names**. En el caso en que la generación de nombres esté deshabilitada, las variables de cadena son tomadas como **dummy names** utilizando el prefijo **asc_** .



Cuando tenemos habilitada la generación de nombres, las opciones de generación de nombres, controlan cómo IDA genera los nombres para las variables de cadena. Cuando no tenemos seleccionado **Generate serial names**, que es por defecto, el prefijo especificado se combina con los caracteres tomados de la cadena sin exceder el máximo especificado en **name lenght**. Un ejemplo de cómo aparecerá una cadena:

```
.rdata:00402069 aThisIsACharact db 'This is a Character array',0
```

El título es utilizado en el nombre, y todo carácter no admitido para nombrarlo, como un espacio, se omitirán cuando se forme el nombre. La opción **Mark as autogenerated**, produce que los nombres generados se muestren en un color distinto, azul oscuro por defecto, en contraposición de los nombres especificados por el usuario en azul por defecto. La opción siguiente **Preserve case**, obliga a utilizar los caracteres tal como aparecen en la cadena para formar el nombre, antes que utilizar el título para ello. Y ya para finalizar **Generate serial names**, causa que IDA realice una secuencia de nombres adjuntándoles un sufijo numérico, compuesto por el valor introducido en **Number**. Además el número de dígitos que se generarán en el sufijo se controla con la opción **Width**. Si miramos, figura arriba, tal como lo tenemos configurado los nombres que nos generaría deberían ser **a000**, **a001**, **a002**.

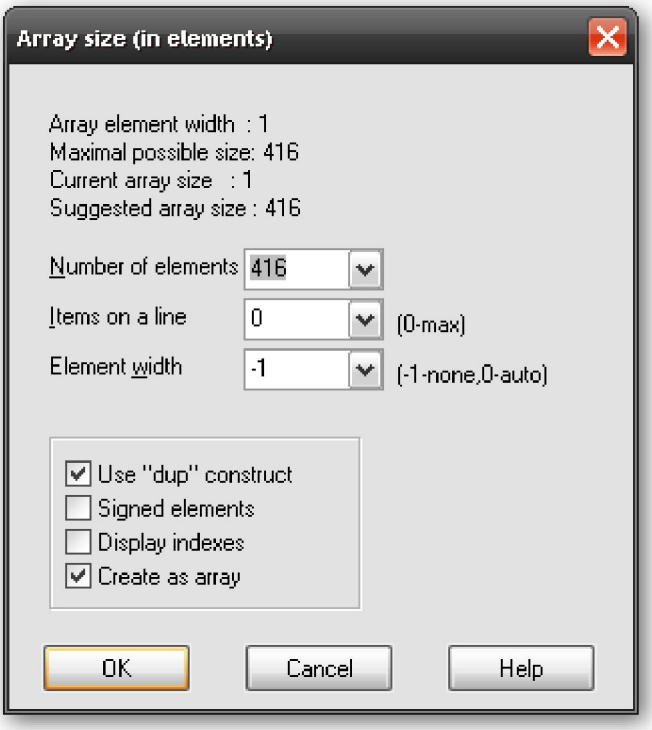
6.4.3.—Especificar Arrays (Conjuntos)

Uno de los inconvenientes de los listados de desensamblado es causado por los lenguajes de alto nivel, los cuales proporcionan muy pocas pistas sobre el tamaño de los **arrays**. En un listado de desensamblado, especificar un array puede requerir una gran cantidad de espacio si cada elemento del conjunto se especifica en una línea de desensamblado. En el siguiente listado de ejemplo vemos las declaraciones de datos que contendrá la variable nombrada como **unk_402060**. El hecho de que sólo el primer elemento del listado es referenciado por cualquiera de las instrucciones, sugiere que puede ser el primer elemento de un conjunto. Ya que antes de referenciarse

directamente, los elementos de un conjunto se referencian entre ellos utilizando un índice de numeración tomando como Offset el primer elemento del conjunto.

```
.rdata:00402060 unk_402060      db    0                 ; DATA XREF: sub_401350+8↑o
.rdata:00402060                                         ; sub_401350+18↑o
.rdata:00402061                 db    0
.rdata:00402062                 db    0
.rdata:00402063                 db    0
.rdata:00402064                 db    0
.rdata:00402065                 db    0
.rdata:00402066                 db    0
.rdata:00402067                 db    0
.rdata:00402068                 db    0
.rdata:00402069                 db    0
.rdata:0040206A                 db    0
```

Para el caso, IDA nos proporciona siempre, el poder agrupar definiciones de datos consecutivas definiéndolas como un simple conjunto. Para crear un conjunto (array), seleccionaremos el primer elemento del array, en nuestro caso **unk_402060** y realizaremos la acción **Edit > Array** con lo cual se nos mostrará el dialogo siguiente.



Si en una ubicación dada hemos definido un elemento dato, entonces la opción **Array** se mostrará haciendo click derecho en el elemento. El tipo del conjunto que se creará vendrá dictaminado por el tipo de dato asociado al elemento seleccionado como primer elemento del conjunto. En nuestro caso, figura arriba, crearemos en conjunto de bytes.

Observación: Antes de crear un conjunto, tenemos que asegurarnos de seleccionar el tamaño apropiado de los elementos de dicho conjunto, cambiando el tamaño de su primer elemento con el tamaño adecuado.

Los campos del diálogo para crear un conjunto los describiremos a continuación:

Array element Widht
Este valor indica el tamaño de cada elemento del conjunto, un byte en este caso, y viene dictaminado por el tamaño del valor del dato seleccionado cuando se abre el diálogo.

Maximum possible size
Este valor es calculado automáticamente como el número máximo de elementos, no de bytes, que podrán incluirse en el conjunto antes de encontrar otro elemento dato ya definido. Especificar un tamaño más grande que el mostrado, es posible, pero requiere que los siguientes datos sean indefinidos a fin de que sean absorbidos en el conjunto.

Number of elements
Aquí es donde podemos especificar el tamaño exacto del conjunto. El tamaño total ocupado por el conjunto se puede calcular de la siguiente forma; **Número de elementos * Tamaño de elemento del conjunto.**

Items on a line
Especifica el número de elementos que se mostrarán en cada línea de desensamblado. Esta opción puede utilizarse para reducir el espacio requerido para mostrar el conjunto.

Element width
Este valor sólo es para propósito de formato y controla el ancho de la columna cuando se muestran varios elementos en una misma línea.

Use “dup” construct
Esta opción causa el número especificado de repeticiones de un elemento dato con idéntico valor, para que luego sean agrupados.

Signed elements
Dicta si los datos mostrados son valores con signo o sin signo.

Display indexes
Produce el indexado del conjunto para mostrarlo como comentario corriente. Es apropiado cuando necesites localizar un valor de dato específico en un conjunto grande.

Create as array
Esta opción parece contravenir el propósito del diálogo, normalmente está seleccionada. No seleccionarla tiene como meta, simplemente especificar cierto número de elementos consecutivos sin agruparlos como un conjunto.

Si aceptamos las opciones especificadas en, figura arriba, nos daría como resultado una declaración de un array, la cual se podría interpretar como un array de **bytes (db)** llamado **byte_402060** constituido por el valor **0** repetido **416 (1A0h)** veces.

El efecto de esta acción, es que **416** líneas de desensamblado han sido condensadas en **una simple línea**, en gran medida debido a la utilización de **dup**. En el siguiente escrito hablaremos de la creación de conjuntos en el **stack frame**.

Performance Bigundill@