

# PROGRAMACIÓN 2

## Curso 2010

### Ambiente de trabajo

#### Índice

<b>1</b>	<b><i>Introducción .....</i></b>	<b><i>2</i></b>
<b>2</b>	<b><i>Ambiente de trabajo.....</i></b>	<b><i>2</i></b>
<b>3</b>	<b><i>Instalación .....</i></b>	<b><i>2</i></b>
<b>4</b>	<b><i>Trabajar con proyectos.....</i></b>	<b><i>3</i></b>
<b>4.1</b>	<b><i>¿Qué es un proyecto? .....</i></b>	<b><i>3</i></b>
<b>4.2</b>	<b><i>Crear un proyecto.....</i></b>	<b><i>3</i></b>
<b>4.3</b>	<b><i>Agregar módulos de definición al proyecto.....</i></b>	<b><i>11</i></b>
<b>4.4</b>	<b><i>Agregar módulos de implementación al proyecto.....</i></b>	<b><i>14</i></b>
<b>4.5</b>	<b><i>Generar el ejecutable del proyecto.....</i></b>	<b><i>15</i></b>
<b>4.6</b>	<b><i>Ejecutar el programa .....</i></b>	<b><i>18</i></b>
<b>4.7</b>	<b><i>Trabajar con Librerías dentro del proyecto .....</i></b>	<b><i>19</i></b>
<b>4.8</b>	<b><i>Usar el <i>Debugger</i>.....</i></b>	<b><i>20</i></b>
<b>5</b>	<b><i>Errores de Compilación.....</i></b>	<b><i>25</i></b>

# 1 Introducción

En este documento se describe el ambiente de trabajo que se deberá instalar y utilizar durante los laboratorios del curso de Programación 2.

## 2 Ambiente de trabajo

Para implementar los programas existen dos opciones:

### 1. Trabajar por línea de comandos

En este caso se trabaja directamente con el compilador del lenguaje a través de una consola de comandos. En el caso de Modula2 y en un sistema operativo Windows, se trabaja con una ventana DOS donde se ejecutan directamente los comandos que ofrece el propio compilador.

En esta situación el ambiente de trabajo esta formado por el compilador del lenguaje y la ventana DOS.

### 2. Trabajar en un entorno de desarrollo

En este caso la interacción con el compilador se hace indirectamente a través de este aplicativo que llamamos *entorno de desarrollo*. El entorno de desarrollo es un programa que suele integrar varias funcionalidades necesarias para implementar programas, creado con el fin de hacer más ágil y amigable esta tarea a los implementadores. Las funcionalidades más comunes son la edición de código (escribir los módulos de definición e implementación en este caso), compilación, debbuging y ejecución de programas. El entorno de desarrollo hace uso del compilador del lenguaje, y lo ejecuta de la misma forma que se haría manualmente desde la línea de comandos.

El ambiente de trabajo en este caso, esta formado por el compilador y el entorno de desarrollo.

El estudiante puede elegir cualquiera de las dos opciones para trabajar durante el curso.

## 3 Instalación

La información respecto a cuál es el compilador oficial del curso, así como la información y guía de cómo instalar el compilador y entorno de desarrollo que se deberá usar en el curso se encuentra en:

<http://www.fing.edu.uy/inco/cursos/prog2/info/index.html>

### IMPORTANTE

La ruta donde se instale el compilador **NO** debe contener espacios en blancos ni tildes.

## 4 Trabajar con proyectos

A continuación se presentan conceptos básicos para trabajar con proyectos en Modula2. Las funcionalidades descritas en cada sección corresponden al entorno de desarrollo *XDS*, que es el nombre del entorno de desarrollo oficial del curso. Vale la pena mencionar que las mismas funcionalidades pueden ser resueltas directamente por línea de comandos, en cuyo caso se puede encontrar una guía básica en:

<http://www.fing.edu.uy/inco/cursos/prog2/info/instala.html#Sec3>

### 4.1 ¿Qué es un proyecto?

Un programa en el lenguaje Modula2 está generalmente compuesto por varios archivos, donde cada uno de ellos es llamado *módulo*. Generalmente se utiliza la palabra proyecto para designar el conjunto de módulos que componen un programa.

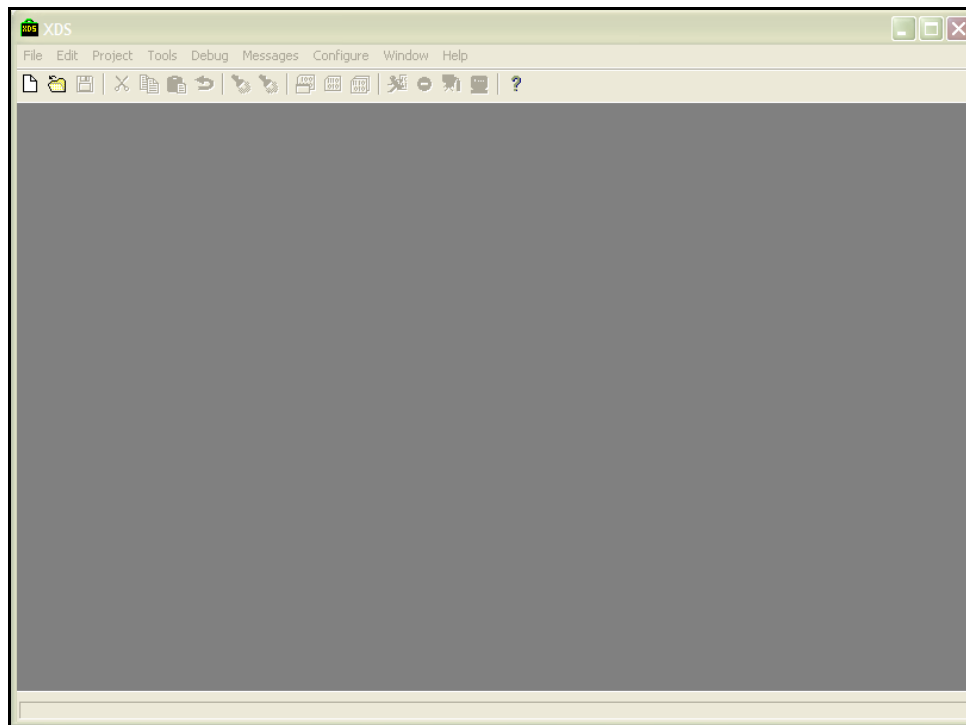
Un proyecto en Modula2 esta compuesto por dos clases de módulos: los *módulos de definición* y los *módulos de implementación*. Por cada módulo de implementación existe un correspondiente módulo de definición, con la excepción del *módulo principal*, que es el que contiene el programa principal del proyecto. Este último no tiene módulo de definición asociado. Los módulos de definición son archivos con extensión *.def* y los de implementación y principal tienen extensión *.mod*.

### 4.2 Crear un proyecto

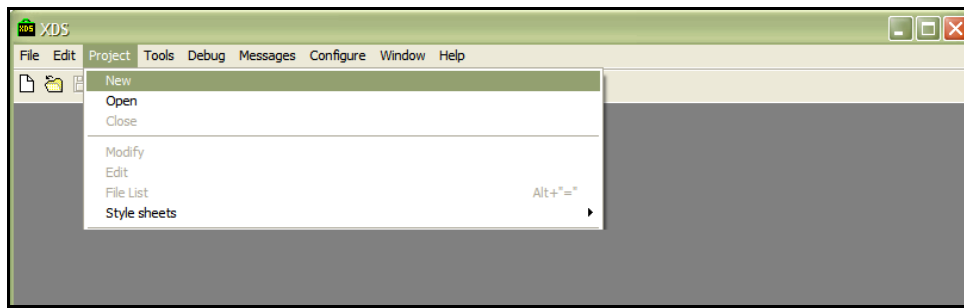
Considere que se desea crear un proyecto, de nombre *proyecto1*, compuesto por un módulo principal que será llamado *principal.mod* y un módulo de implementación, con su correspondiente módulo de definición, los cuales serán llamados *archivo1.mod* y *archivo1.def* respectivamente.

Lo primero que se debe realizar es abrir el entorno de desarrollo, que de aquí en más en este documento será nombrado *XDS*, el cual se inicia en Windows desde *Inicio/Programas/Native XDS-x86/XDS Environment* o un camino similar dependiendo del directorio elegido durante la instalación.

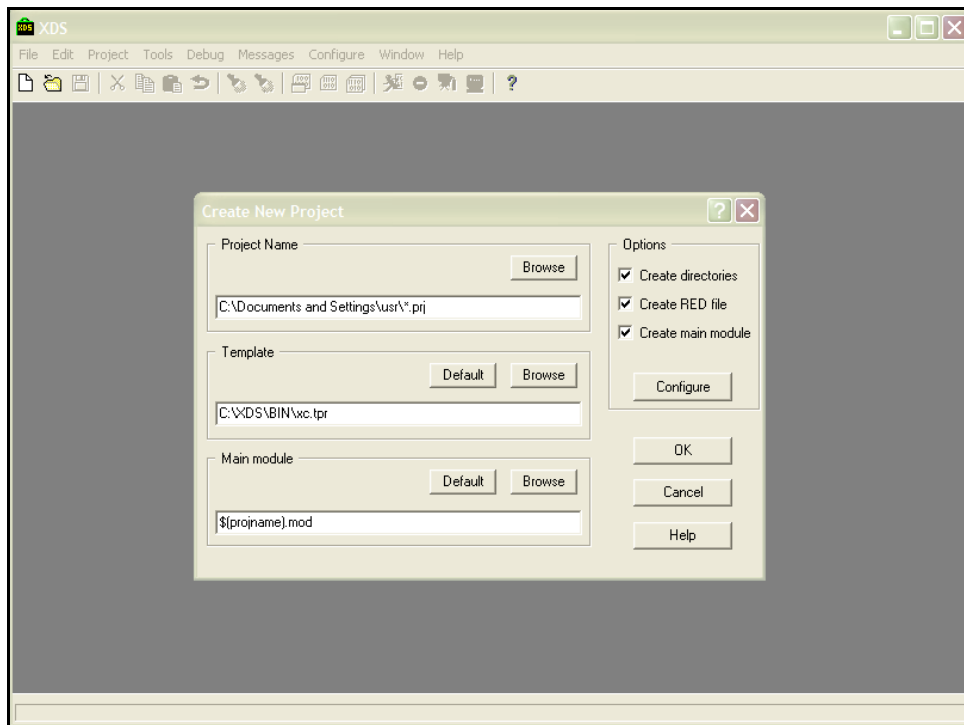
Al abrir XDS se despliega la siguiente pantalla:



Para crear el proyecto, se debe ir al menú y elegir *Project/New* como se muestra a continuación:

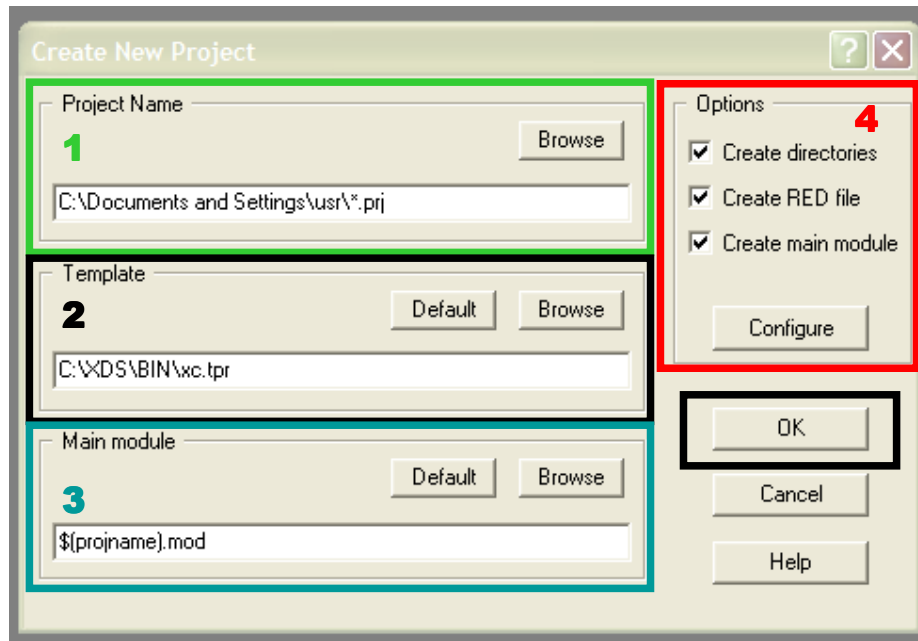


Se desplegará una pantalla como la siguiente:



### 4.2.1 Descripción de las opciones para su creación

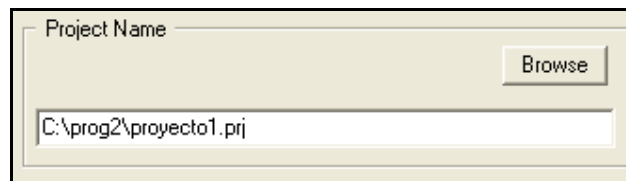
A continuación se describen cada una de las opciones que se muestran en la pantalla anterior. Para esto, se agrupan las opciones por recuadros de colores numerados como se muestra a continuación.



### **Recuadro 1 - Project Name**

En esta cuadro de texto se configura cual será el directorio de trabajo para el proyecto a crear y el nombre que se le dará al mismo. En otras palabras, se debe indicar la ruta donde se desea que el proyecto sea creado y cual será su nombre. Para esto, se puede editar directamente el cuadro de texto indicando la ruta y nombre del proyecto, o se puede usar el botón *Browse*, para elegir a través del explorador el directorio deseado y a continuación se completa con el nombre del proyecto.

Para el ejemplo de este instructivo, el proyecto será creado en el directorio *C:\prog2*. El nombre del proyecto, como ya se mencionó es *proyecto1*. El cuadro de texto se verá como se muestra a continuación:



### **IMPORTANTE**

- 1 - La ruta elegida para el directorio de trabajo **NO** debe contener espacios en blancos ni tildes.
- 2 - La ruta del directorio de trabajo debe ser una ruta válida. En el caso del ejemplo, debe estar creado el directorio *prog2*.

### **Recuadro 2 – Template**

En este cuadro de texto se indica el template a usar para determinado archivo de configuración que necesita el entorno de desarrollo. El archivo *xc.tpr* que se muestra en el ejemplo, es el template por defecto y ya está ubicado en el directorio *C:\XDS\BIN*, pero este podría ser diferente si fue elegido otro directorio para la instalación de XDS. **No se debe modificar este cuadro de texto.**

### **Recuadro 3 – Main Module**

En este cuadro de texto se indica cual es el nombre que se le dará al módulo principal del proyecto. Por defecto el módulo principal tendrá el nombre de la variable *projname*, la cual almacena el nombre que le fue dado al proyecto en el “*Recuadro 1 – Project name*”.

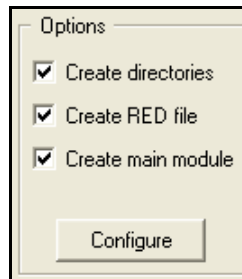
Si se desea que el nombre del módulo principal sea diferente al nombre del proyecto, entonces se debe editar el cuadro de texto y sustituir  $\$(projname)$  por el nombre deseado.

También puede suceder que ya se cuente con un archivo que se quiera usar como módulo principal del proyecto, en cuyo caso se puede usar el botón *Browse* para seleccionar dicho archivo, o editar el cuadro de texto para escribir la ruta donde se encuentra el mismo.

En el caso del ejemplo que se está trabajando en este documento, la variable *projname* tiene el valor *proyecto1*, y por tanto, el nombre que tendrá el módulo principal del proyecto será *proyecto1.mod*.

#### **Recuadro 4 - Options**

Aquí se indica al entorno de desarrollo las opciones que se desean configurar para la creación del proyecto.

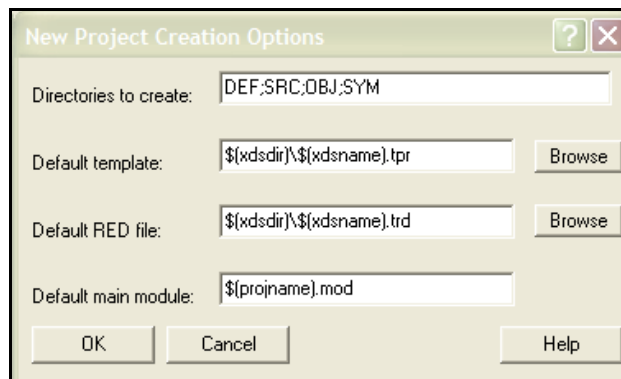


#### **Descripción de los check boxes:**

- *Create directories*: al marcar esta opción se indica al XDS que cree determinados directorios que permiten ordenar los distintos tipos de archivos que componen el proyecto (archivos de definición, archivos de implementación, archivos con código objeto, etc.). **Es recomendable dejar marcada esta opción.**
- *Create RED file*: al marcar esta opción se indica al XDS que cree automáticamente el archivo de extensión RED. Este es un archivo de configuración que usa el compilador de Modula2 para buscar las rutas de los directorios donde se encuentran los archivos que componen el proyecto. **Es recomendable dejar marcada esta opción.** Mas adelante se verá en detalle el contenido y utilidad de este archivo.
- *Create main module*: al marcar esta opción se indica al XDS que cree el módulo principal del proyecto. Lo va a crear con el nombre que se le indicó en el “Recuadro 3 – Main Module”. Si en el recuadro 3 se indicó que use como módulo principal del proyecto un archivo que ya existe, o si el módulo principal aún no existe pero se quiere generar en otro momento, entonces este check box no debe ser marcado.

En el ejemplo que se trabaja en este documento, **no** será marcada esta opción.

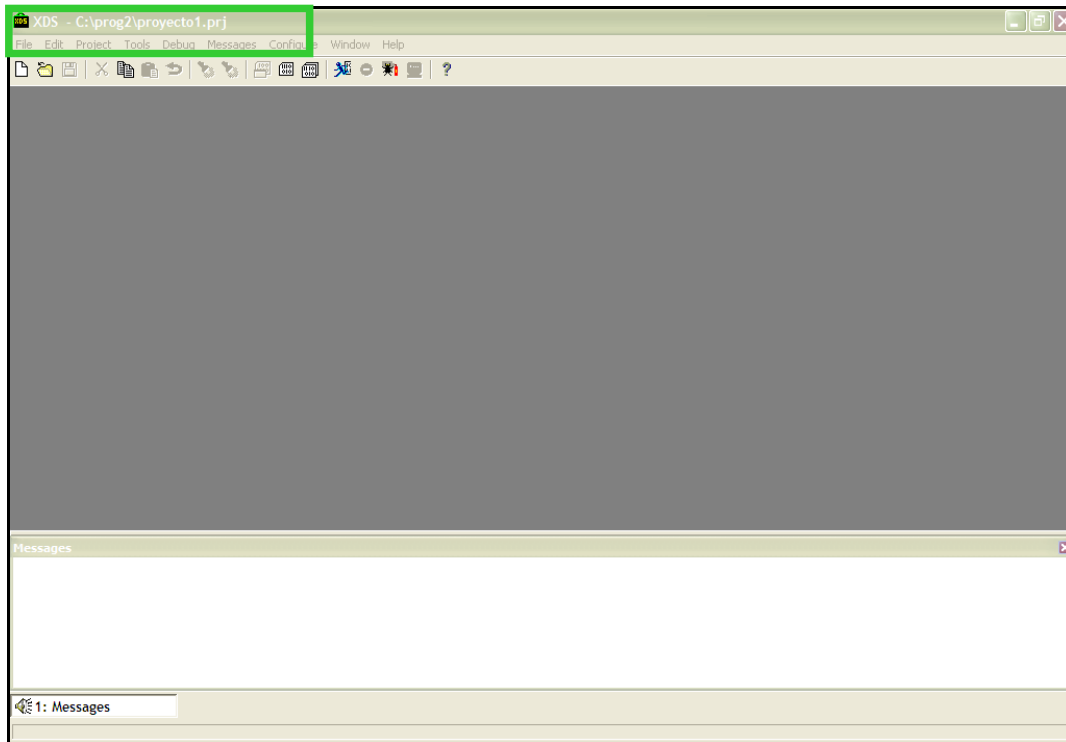
- *Botón Configure*: al presionar este botón se despliega la siguiente pantalla:



donde es posible cambiar parámetros de configuración. No será necesario modificar estos datos. Para volver a la pantalla anterior sin modificar ningún valor, se presiona el botón *Cancel*.

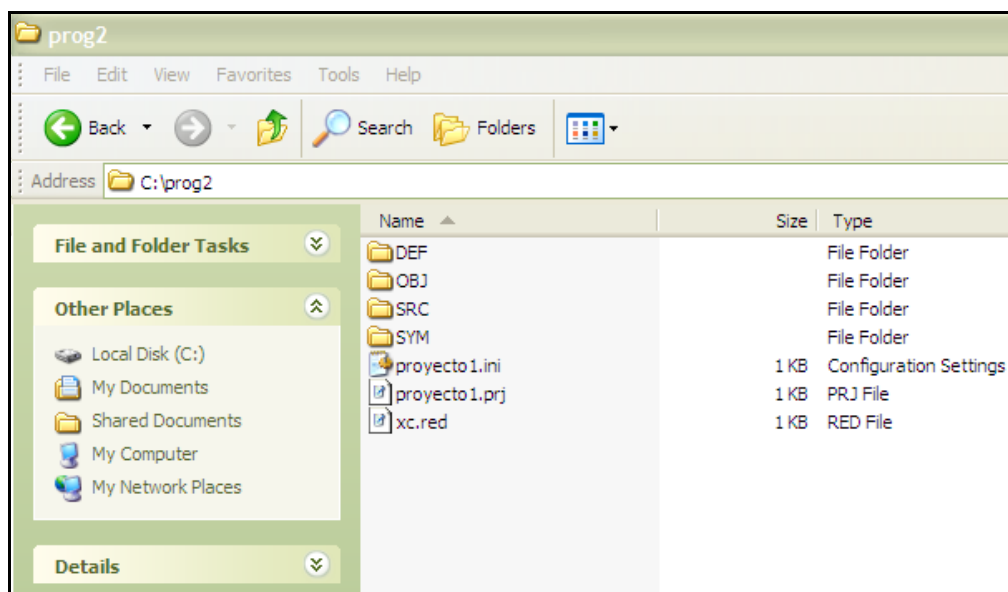
#### Recuadro 5 Botón OK

Cuando se presiona el botón *OK* de la ventana anterior, se crea efectivamente el proyecto, y el mismo se abre en el ambiente XDS, como se muestra a continuación:



El nombre y ruta del proyecto creado se visualiza en la propia pantalla del XDS, tal como lo indica el recuadro verde colocado sobre la imagen anterior.

A su vez, en el directorio de trabajo especificado en el **Recuadro 1 - Project Name**, se crea la siguiente estructura:



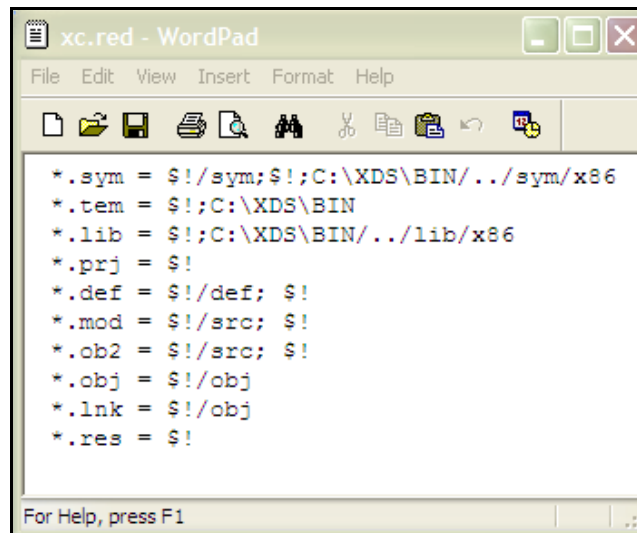
**Descripción del contenido del directorio creado:****1- Directorios:**

- *DEF*: directorio donde se almacenan los archivos con extensión *.def*, correspondientes a los módulos de definición que forman parte del proyecto.
- *OBJ*: directorio donde se almacenan los archivos con extensión *.obj*, los cuales contienen el código objeto de cada módulo de implementación del proyecto. Se generan automáticamente al compilar cada uno de estos.
- *SRC*: directorio donde se almacenan los archivos con extensión *.mod*, correspondientes a los módulos de implementación que forman parte del proyecto.
- *SYM*: directorio donde se almacenan los archivos con extensión *.sym* que se generan automáticamente cuando se compilan los módulos de definición del proyecto.

**2- Archivos:**

- *Proyecto1.ini*: archivo que contiene los nombres de los módulos del proyecto que se encontraban abiertos la última vez que se trabajó en el XDS. Cuando se “abre” el XDS, éste abre cada uno de los módulos que figuran en este archivo, con el fin de “retomar” en el último estado en que se cerró el proyecto.
- *Proyecto1.prj*: archivo que define al proyecto. En él se especifica cuáles son los módulos que componen el proyecto. Este archivo es generado y mantenido automáticamente por XDS a medida que se crea y manipula el proyecto.
- *xc.red*: es el archivo de configuración del proyecto. En él se especifican las rutas de acceso a cada uno de los archivos que lo componen. Es usado por Modula2 para saber donde encontrar cada uno de los módulos que componen el proyecto. Este archivo es generado y mantenido automáticamente por XDS a medida que se crea y manipula el proyecto.

Para el ejemplo que se está trabajando en este documento, este archivo tiene la siguiente forma:



```
*.sym = $!/sym;$!;C:\XDS\BIN\..\sym/x86
*.tem = $!;C:\XDS\BIN
*.lib = $!;C:\XDS\BIN\..\lib/x86
*.prj = $!
*.def = $!/def; $!
*.mod = $!/src; $!
*.ob2 = $!/src; $!
*.obj = $!/obj
*.lnk = $!/obj
*.res = $!
```

donde se indica que, por ejemplo, los archivos de definición, “\*.def”, se deben buscar primero en *\$!/def* y a continuación en *\$!* en caso de no encontrarlo en el primer lugar. *\$!* representa el directorio de trabajo, que en este caso es *C:\prog2*. De esta manera, el archivo especifica que los archivos de definición del proyecto se encuentran en *C:\prog2DEF* o en *C:\prog2*. El razonamiento es análogo con el resto de las extensiones.



En el caso de los archivos con extensión *.sym*, *.tem*, y *.lib* se especifican además las rutas donde se encuentran las librerías y/o archivos de definición de Modula2, dado que en la mayoría de los proyectos que se construyen se usan utilidades provistas por el propio lenguaje, como puede ser el uso de módulos para la manipulación de cadenas de caracteres (Strings) o para manejo de la entrada/salida estándar.

#### NOTA

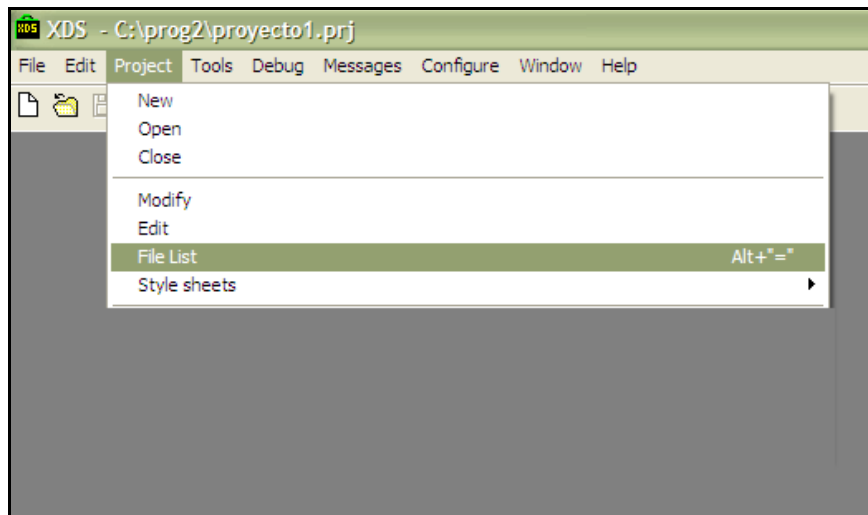
XC es el nombre de la utilidad de XDS que se encarga de manipular los proyectos y programas en Módulo2 (compilar, linkeditar, ejecutar). Cuando se usa XDS por línea de comandos, se invoca directamente a esta utilidad<sup>1</sup>. Cuando se usa el entorno de desarrollo que se describe en este documento, esta utilidad es usada por el propio entorno, es decir, es transparente para el usuario. Esta es invocada cuando se ejecutan determinadas funcionalidades del entorno, como ser *Compile*, *Make*, *Build all*, descriptas más adelante.

Cuando se invoca a la funcionalidad *xc*, sea de la forma que sea, ésta trata de ubicar el archivo *xc.red*. Primero lo busca en el directorio donde se este ejecutando; sino lo encuentra entonces lo busca en el directorio donde se encuentra la utilidad *xc* (C:\XDS\BIN o una ruta similar dependiendo de cuál fue la ruta elegida para la instalación de XDS).

Por lo anterior, es importante que el archivo *xc.red* se encuentre en el directorio de trabajo, dado que es en éste (y no en el ubicado en el directorio C:\XDS\BIN) donde se encuentran las rutas de los módulos que componen el proyecto con el cual se quiere trabajar.

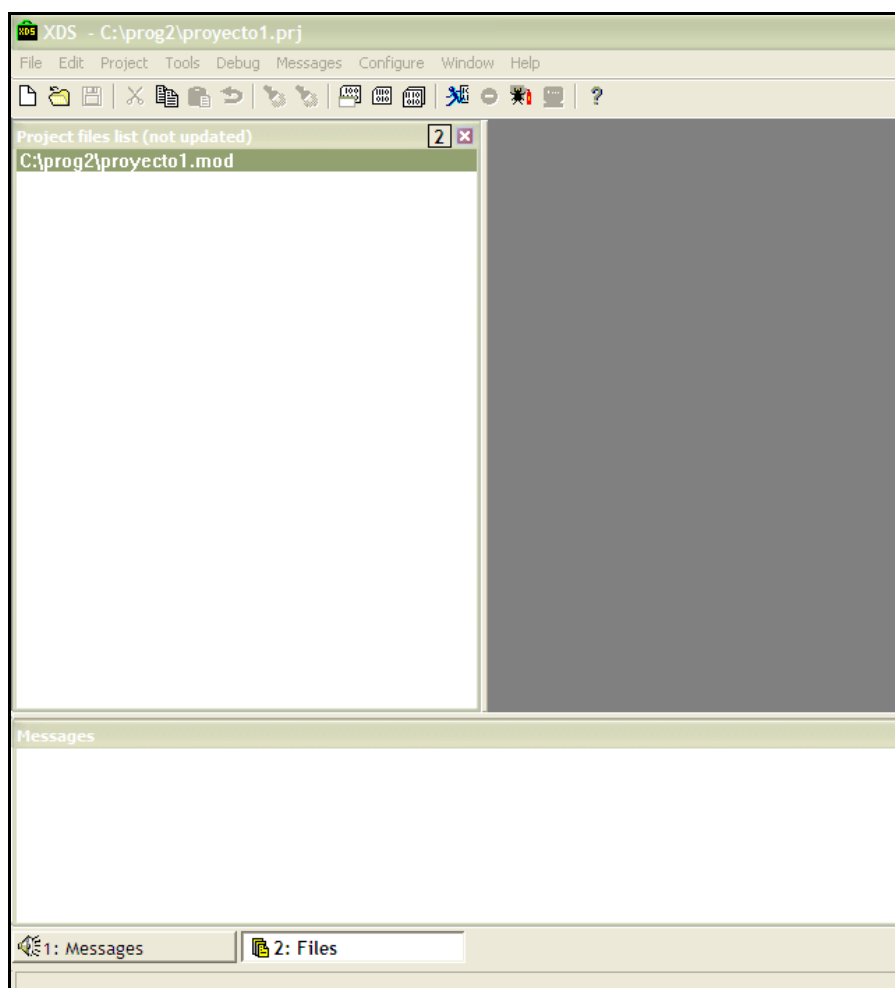
## 4.2.2 Visualización de los componentes del proyecto

XDS cuenta con una funcionalidad que permite visualizar en la pantalla del proyecto todos los módulos que lo componen. Para acceder se debe ejecutar *Project/File List* desde el Menú de XDS como se muestra en la siguiente imagen:



Con lo cual la pantalla del proyecto pasa visualizarse como se muestra a continuación:

<sup>1</sup> Por instrucciones para trabajar por línea de comandos, ver <http://www.fing.edu.uy/inco/cursos/prog2/info/instala.html#Intro>



A medida que se agregan módulos al proyecto, estos son listados en el panel. Para editar uno de estos archivos alcanza con hacer doble click sobre el nombre del archivo que se desea editar. Dicho archivo es abierto en el panel que aparece en color gris oscuro en la figura anterior.

## 4.3 Agregar módulos de definición al proyecto

Básicamente, existen dos formas de agregar módulos al proyecto:

- Los módulos de definición ya existen, y se desea agregarlos al proyecto.
- Los módulos de definición no existen, y se desea generar los mismos en el propio ambiente de desarrollo.

Se describen a continuación instrucciones básicas para trabajar con ambas opciones.

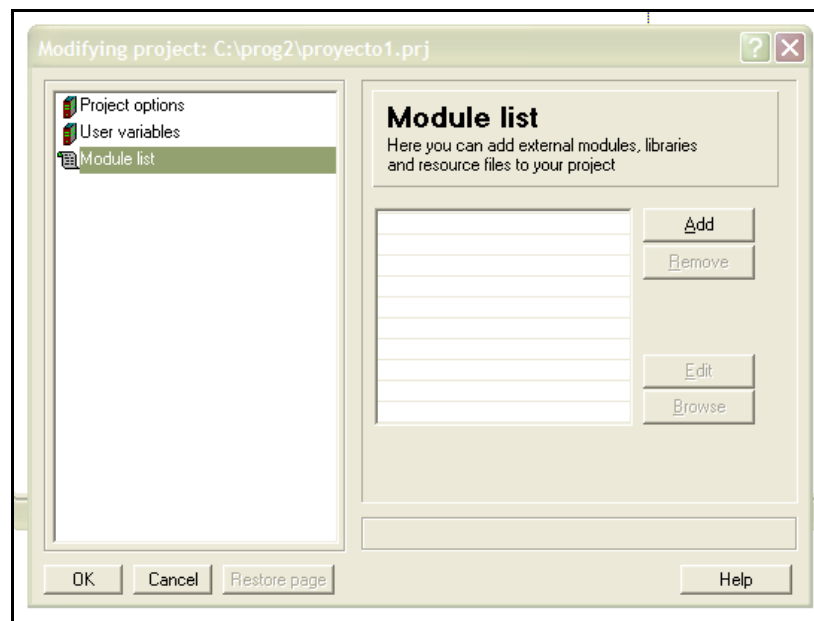
### 4.3.1 Los módulos de definición ya existen y se desea agregarlos al proyecto

Esta es la forma de agregar archivos de definición al proyecto que será usada en los laboratorios del curso de Programación 2.

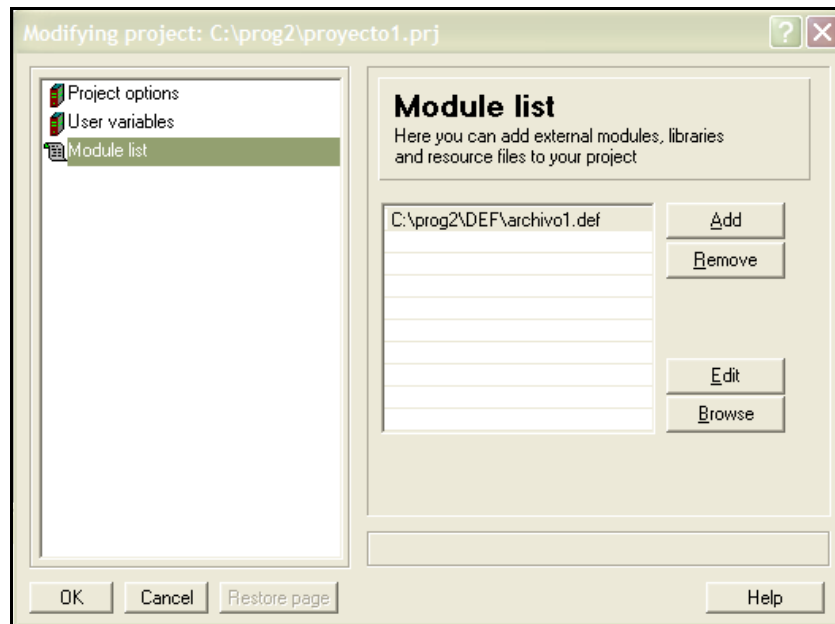
Para explicar como se agregan archivos de definición, se va a retomar el ejemplo que se ha venido desarrollando en este instructivo. Es necesario agregar entonces, el módulo *archivo1.def* al proyecto.

Para esto se deben seguir tres pasos:

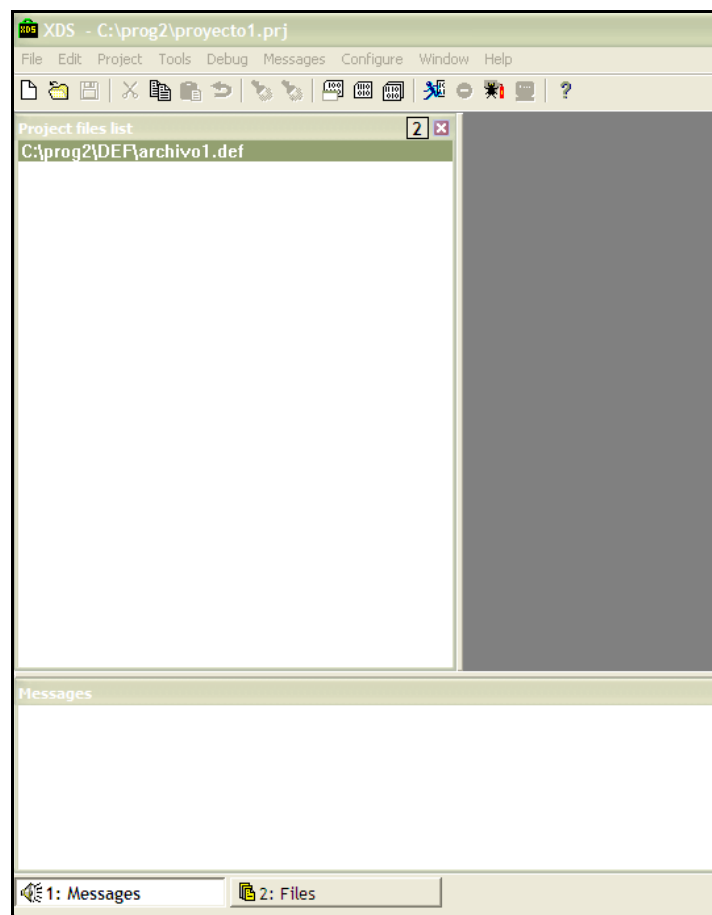
- 1- Copiar los archivos de definición en el directorio *DEF* dentro del directorio de trabajo. En el caso del ejemplo, *archivo1.def* se deben copiar en *C:\prog2\DEF*.
- 2- Indicar que cada uno de los archivos copiados es parte del proyecto. Para esto se debe ir al Menú y seleccionar *Project/Modify/Module List* con lo cual se despliega la siguiente pantalla:



Luego, mediante el botón *Add*, y con la ayuda del botón *Browse*, buscar y seleccionar cada módulo de definición copiado en el directorio *DEF*. En el ejemplo, se busca *archivo1.def* en *C:\prog2\DEF*. Al terminar presionar *OK*. La pantalla se actualiza y se visualiza de la siguiente forma:



- 3- Actualizar la lista de visualización de módulos del proyecto. Para esto se debe ir al Menú y seleccionar *Tools/Update File List*. Para el ejemplo, la pantalla se visualizará como se muestra en la siguiente imagen:



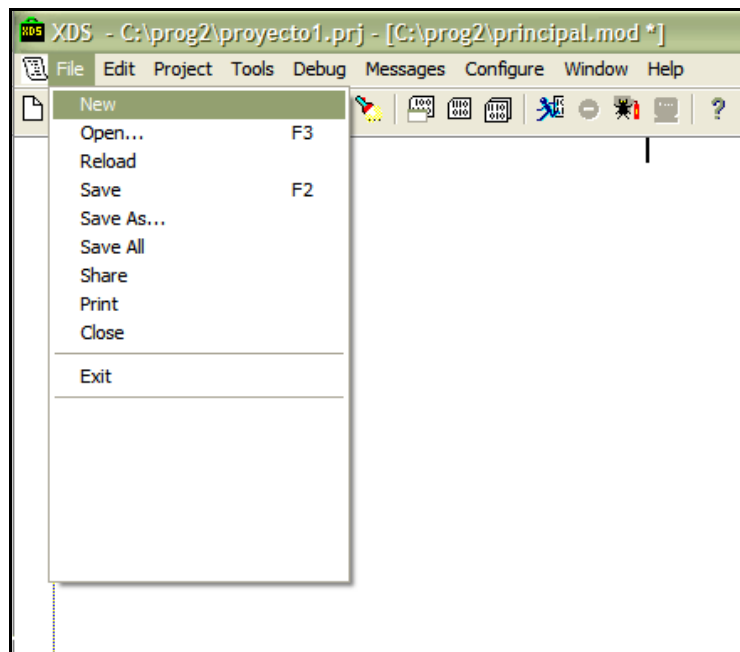
**NOTA**

La actualización de la lista de visualización de módulos solo se actualizará correctamente cuando los nuevos módulos agregados al proyecto compilen sin errores. En este ejemplo, la lista **no** se hubiera actualizado como se muestra en la imagen si el archivo *archivo1.def* estuviera vacío o tuviera algún otro error de compilación. Si esto hubiera pasado, debería corregir los errores y repetir este paso.

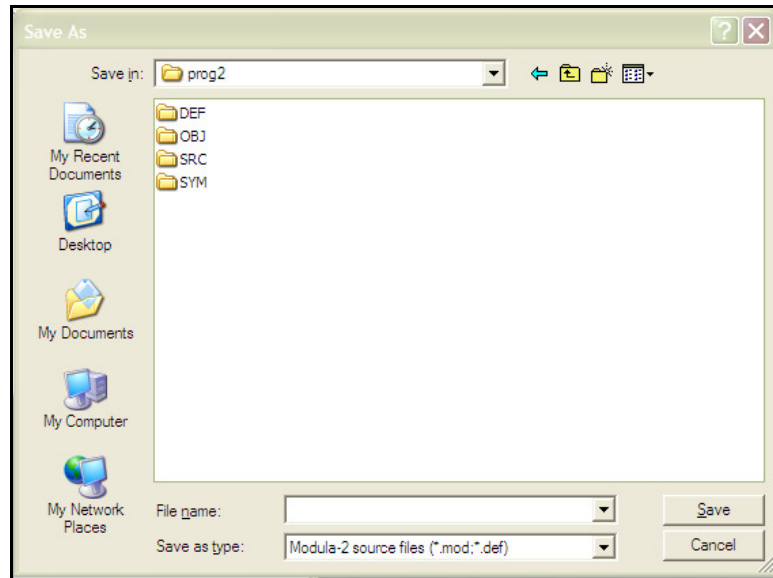
### 4.3.2 Los módulos de definición no existen, se desean generar en el propio ambiente de desarrollo

En este caso no existe previamente un archivo de definición, sino que se va a generar en el propio entorno de desarrollo. Para hacer esto se deben ejecutar los siguientes 3 pasos:

- 1- Crear el archivo de definición. Para hacer esto, ir al Menú y seleccionar *File/New* como se muestra a continuación



Esto genera un nuevo archivo que debe ser almacenado. Para hacer esto, se debe acceder al Menú y seleccionar *File/Save As*, donde se muestra un cuadro de diálogo como el siguiente:



Se debe seleccionar la carpeta *DEF* (recordar que es donde se almacenan los archivos de definición del proyecto). Se le da un nombre al archivo y se elige la extensión *.def*. Se presiona el botón *Save* y con esto el archivo ya es creado.

#### IMPORTANTE

El nombre elegido para el archivo **NO** debe contener espacios en blancos ni tildes.

- 2- Indicar que el archivo recién creado es parte del proyecto. Para esto se debe ir al Menú y seleccionar *Project/Modify/Module List*, de la misma manera que fue explicado en el paso 2 para el caso donde existen previamente los archivos de definición. En este caso, se selecciona el archivo de definición recién creado.
- 3- Actualizar la lista de visualización de módulos del proyecto. Para esto se debe ir al Menú y seleccionar *Tools/Update File List*, de la misma manera que fue explicado en el paso 3 para el caso donde existen previamente los archivos de definición.

## 4.4 Agregar módulos de implementación al proyecto

El proceso para agregar módulos de implementación al proyecto es análogo al proceso descrito en la sección anterior para el caso de módulos de definición.

Al igual que con los archivos de definición existen dos formas de agregar archivos de implementación al proyecto:

- Los módulos de implementación ya existen, y se desea agregarlos al proyecto.

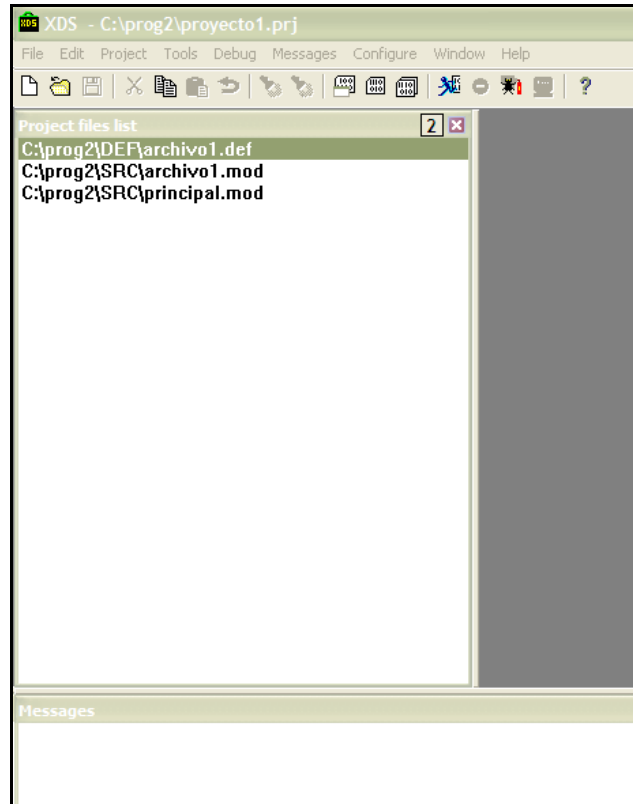
Los pasos a seguir son los mismos que los explicados en la sección de 4.3, con la salvedad que los archivos deben ser copiados en el directorio *SRC*.

- Los módulos de implementación no existen, y se desea generar los mismos en el propio entorno de desarrollo.

Para los laboratorios del curso, esta es la forma más común por la cual se agregan archivos de implementación al proyecto.

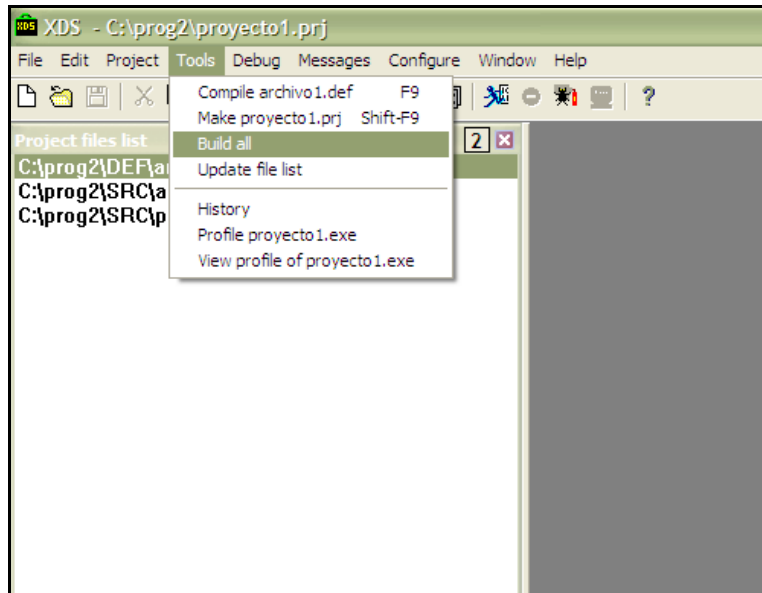
Los pasos a seguir son los mismos que los explicados en la sección de 4.3, con la salvedad que los archivos deben ser salvados en el directorio *SRC*.

Continuando con el ejemplo de este instructivo, se crean, almacenan y agregan al proyecto: *archivo1.mod* y *principal.mod* que son los módulos de implementación del ejemplo. Luego de esto, el proyecto se visualiza de la siguiente forma:

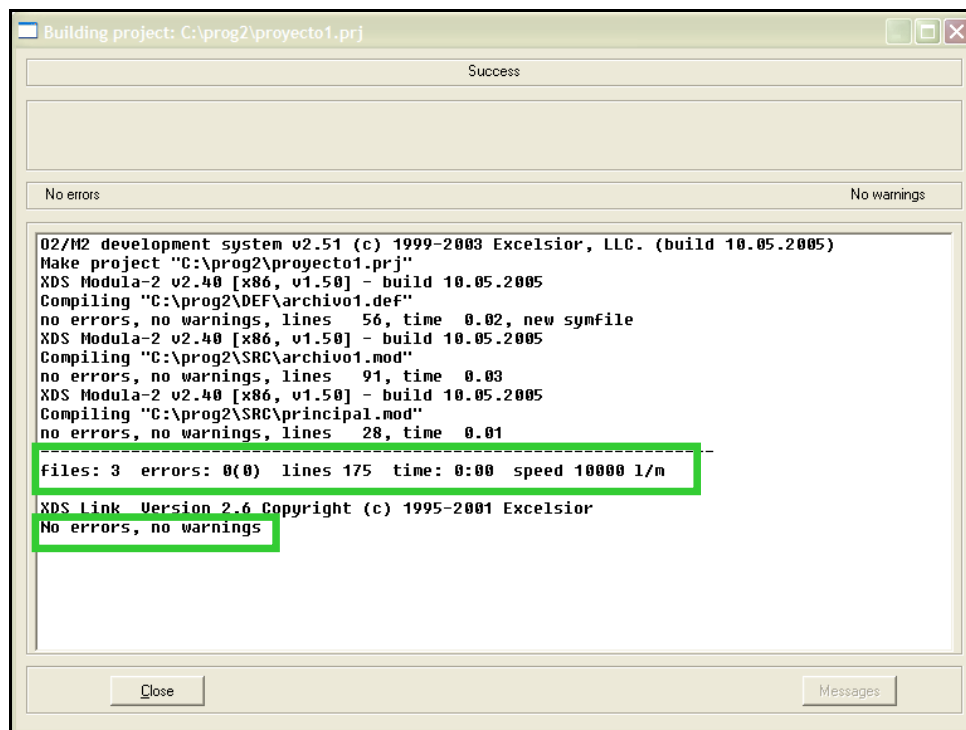


## 4.5 Generar el ejecutable del proyecto

Para generar el ejecutable del programa que se está construyendo, es necesario compilar los módulos y linkeditarlos. XDS provee una funcionalidad que hace ambas acciones a la misma vez. La funcionalidad se llama *Build all* y se encuentra en el Menú, opción *Tools/Build all* como se muestra en la siguiente imagen:



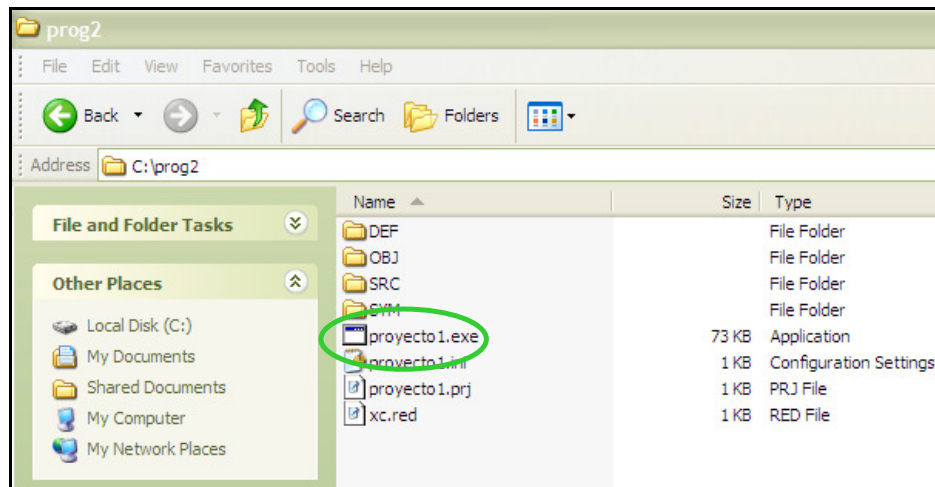
Si la generación del ejecutable es exitosa (no ocurren errores de compilación y/o linking) se muestra una pantalla como la siguiente:



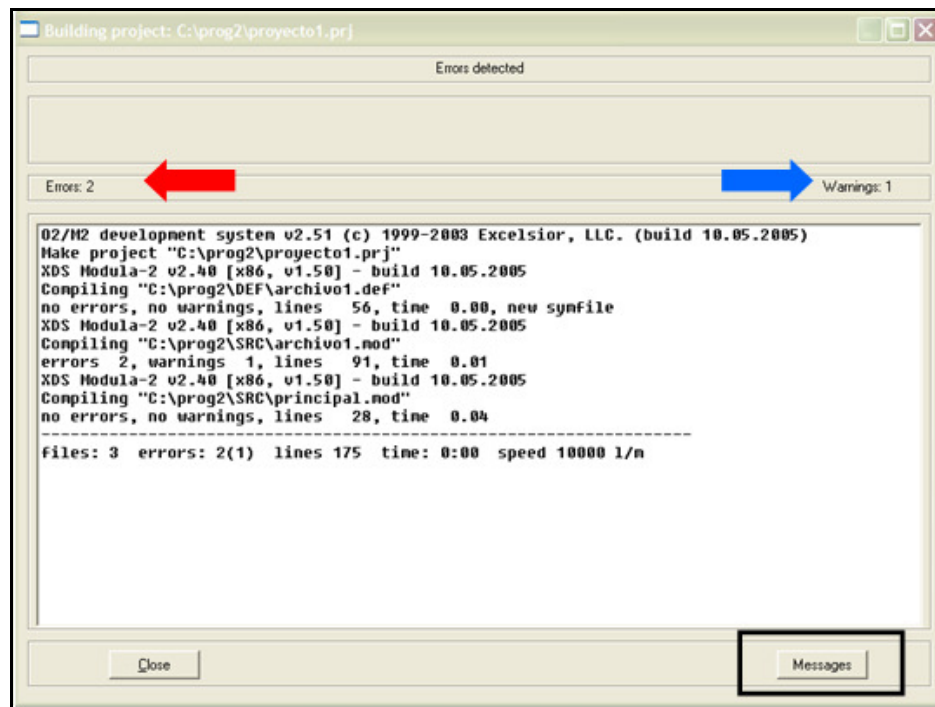
Los recuadros verdes en la imagen anterior señalan las líneas de texto de la salida que indican que la tarea fue exitosa.

A su vez, se genera el ejecutable del proyecto en el directorio de trabajo, el cual tendrá el mismo nombre del proyecto. En el caso del ejemplo, se generará el ejecutable *proyecto1.exe* en el directorio *C:\prog2* como se puede apreciar a continuación:

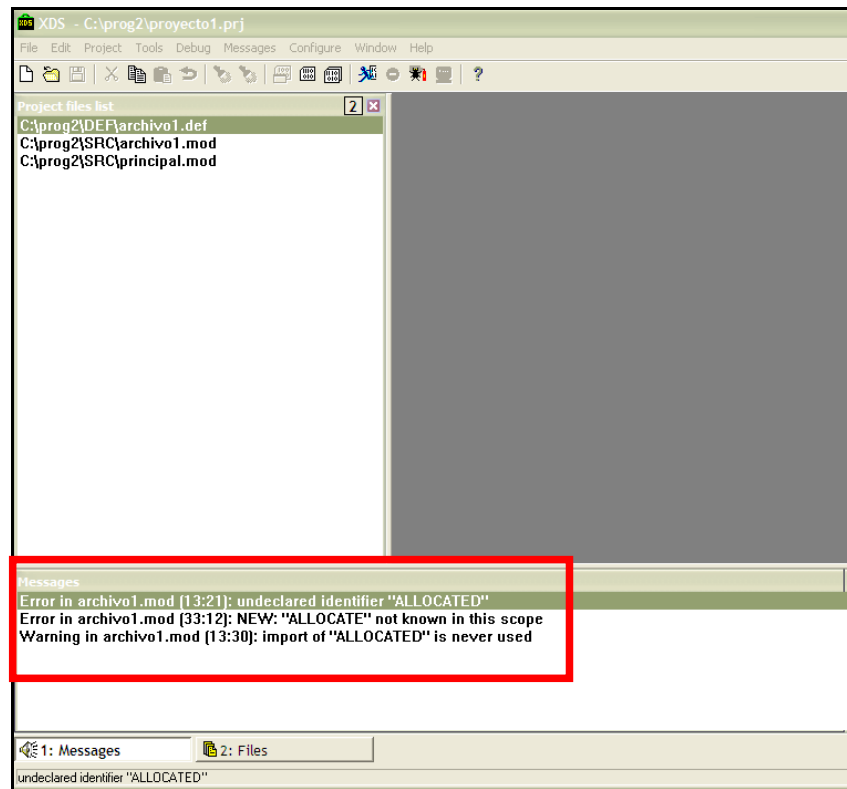




Si ocurren errores, ya sea de compilación y/o linking, se muestra una pantalla que indica cuales y cuántos son los errores ocurridos y de que tipo. Para ejemplificar lo anterior, se modificó uno de los archivos del ejemplo para que genere un error. El resultado es la pantalla que se muestra abajo:



La flecha roja indica la cantidad de errores encontrados, mientras que lo indicado por la flecha azul indica la cantidad de advertencias que encuentra el compilador. Para visualizar cuales son los errores y advertencias, se presiona el botón *Messages*, el cual despliega en la pantalla del proyecto el panel indicado con el recuadro rojo en la siguiente imagen:



En este panel se encuentra una descripción de cada error encontrado. Si se da un doble click sobre uno de los errores, se abre el módulo en el cual es detectado ese error. Esto ayuda a la tarea de corrección de errores.

Una vez realizadas las modificaciones necesarias, se repite el proceso de generación. Este ciclo se realiza tantas veces como sea necesario hasta que todos los errores son corregidos, y se genera el ejecutable de forma exitosa.

#### NOTA

El panel indicado con el recuadro rojo en la imagen anterior, puede ser visualizado en cualquier momento mientras se trabaja en el proyecto, no solo cuando se esta generando el ejecutable y ocurren errores. Esto se logra a través del Menú, con la opción *Messages/Show*.

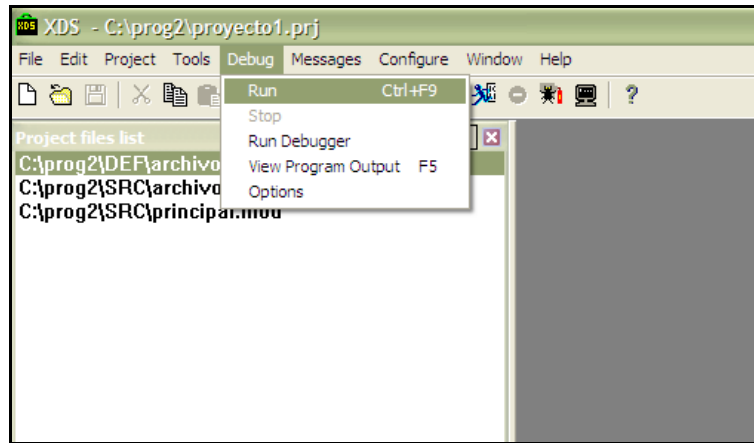
## 4.6 Ejecutar el programa

La ejecución del programa se puede realizar de dos formas diferentes:

- desde el propio entorno de desarrollo
- desde línea de comandos

### 4.6.1 Ejecución desde el entorno de desarrollo

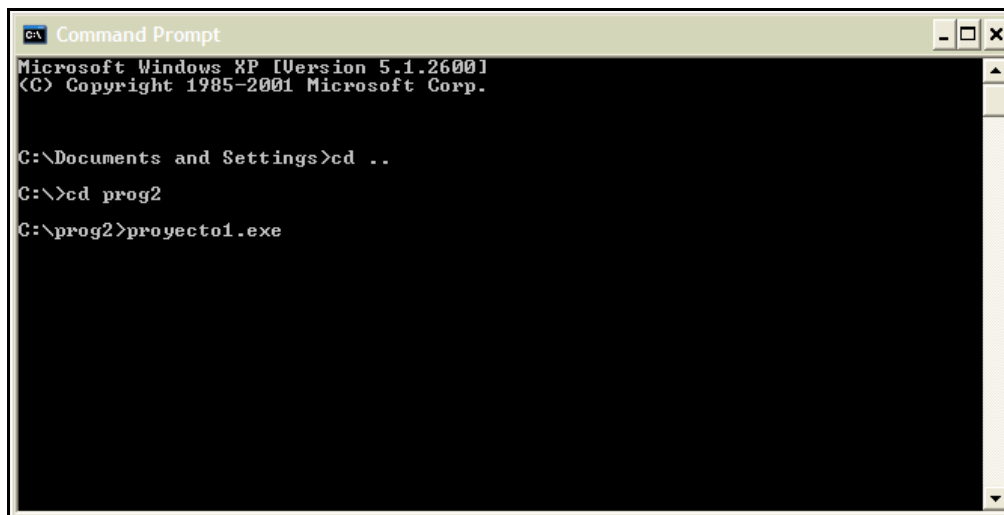
Para ejecutar el programa se debe ir al Menú y seleccionar *Debug/Run*. A continuación, se abre una consola DOS y se ejecuta el programa realizado. En el caso del ejemplo que se esta desarrollando en este instructivo se ejecuta *proyecto1.exe*



## 4.6.2 Ejecución desde línea de comando

Otra forma para ejecutar el programa generado es la siguiente:

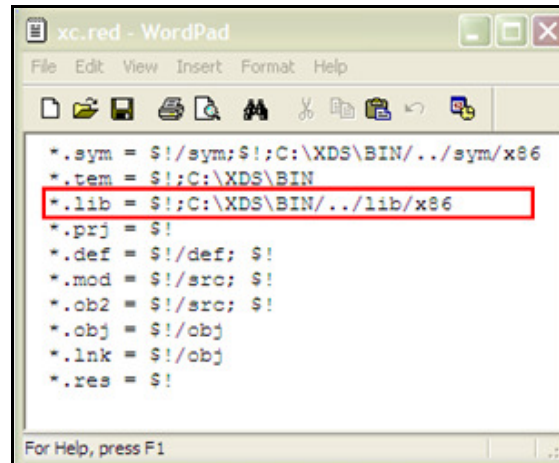
- abrir una consola DOS
- posicionarse en el directorio de trabajo, que es donde se generó el archivo ejecutable. En el caso del ejemplo, se debe posicionar en el directorio *C:\prog2*.
- una vez en el directorio de trabajo, se ejecuta el programa simplemente escribiendo el nombre del archivo ejecutable y a continuación se presiona *enter*, como se muestra en la siguiente imagen:



## 4.7 Trabajar con Librerías dentro del proyecto

En algunas oportunidades es necesario trabajar con librerías, las cuales deben ser incluidas en el proyecto. Las librerías en Modula2 son archivos con extensión *.lib*. Al igual que con los archivos de implementación y definición del proyecto, los módulos de librería deben ser copiados a un directorio, y posteriormente indicar al proyecto que cuenta con un módulo de este tipo.

Si se observa el archivo de configuración del proyecto que se usa como ejemplo en este instructivo, el cual se vuelve a mostrar en la siguiente imagen, se le indica al compilador que los módulos de extensión *.lib* los encontrará en el directorio de trabajo (\$!) o en el directorio de las librerías del propio compilador. Esto quiere decir que para que el compilador encuentre las librerías que se deseen agregar, es necesario copiarlas en el directorio de trabajo.



Asimismo, debe incluirse en el proyecto el módulo de definición de la librería. Esto se realiza de la misma manera que se agrega cualquier módulo de definición al proyecto, tal como se explicó en la sección 4.3.

Como ejemplo, suponga que es necesario trabajar con la librería *Lib1.lib* en el ejemplo que se viene desarrollando en este documento. Para agregar tal librería al proyecto, y de acuerdo a lo dicho anteriormente, se deben los siguientes pasos:

1. Copiar *Lib1.def* al directorio *C:\prog2\DEF*
2. Copiar *Lib1.lib* al directorio de trabajo del proyecto, en este caso *C:\prog2*
3. Indicarle al proyecto que ahora cuenta con estos dos nuevos archivos. Esto se hace a través de la funcionalidad del Menú *Project/Modify*, de manera análoga a como se explicó en la sección 4.3.

## 4.8 Usar el *Debugger*

### 4.8.1 Características generales de un Debugger

El *debugger* es un programa que permite detectar errores de programación en otros programas, en particular los errores que se producen en tiempo de ejecución.

En la mayoría de los casos, cuando se inicia un *debugger*, este inicia el programa a examinar. El programa examinado se ejecuta normalmente hasta que el *debugger* detiene su ejecución, permitiendo al usuario examinar la instancia actual del programa.

En general<sup>2</sup>, un debugger puede detener la ejecución del programa examinado por alguno de los siguientes motivos:

- A pedido del usuario.
- En una sentencia determinada del programa, marcada por el usuario mediante lo que se denomina un *breakpoint*.
- En un momento determinado cuando se cumplen determinadas condiciones.
- En una sentencia determinada del programa la cual cumple ciertas condiciones indicadas por el usuario, por lo que se denomina un *breakpoint* condicional.

Durante la interrupción el usuario puede:

- Examinar y modificar la memoria y las variables del programa.
- Examinar el contenido de los registros del procesador.
- Examinar el *stack* de llamadas del programa hasta la situación actual.

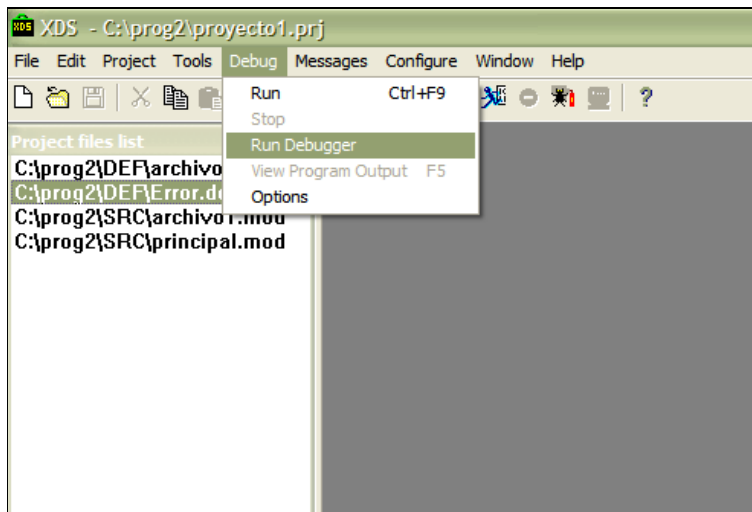
<sup>2</sup> Un debugger depende de la arquitectura y sistema operativo en el que se ejecuta, por lo cual algunas funcionalidades pueden cambiar de un sistema a otro. En este instructivo se mencionan las funcionalidades más comunes.

- Cambiar el punto de ejecución del programa, de manera que al reanudar la ejecución, continúe en un punto diferente al punto en el que fue detenido.
- Ejecutar el programa sentencia a sentencia.
- Ejecutar partes determinadas del código, como puede ser el interior de una función.

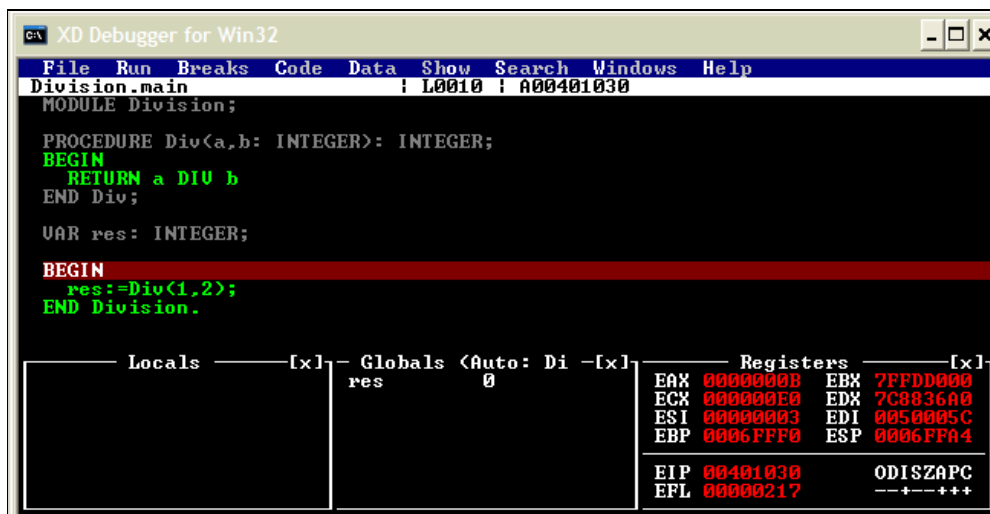
## 4.8.2 Debugger incluido en XDS

XDS contiene un programa *debugger* que se puede ejecutar desde el propio ambiente de desarrollo XDS o por línea de comandos. En esta sección se proveerán nociones básicas para trabajar con el debugger a través de XDS. Para hacerlo por línea de comandos se recomienda leer el manual de usuario que se encuentra en el archivo `../XDS/PDF/xd.pdf`, el cual se genera durante la instalación del compilador.

Para comenzar la ejecución del debugger, se debe ir al Menú, opción *Debug/RunDebugger* como se muestra en la siguiente imagen:



El cual inicia en una pantalla aparte, como se muestra a continuación:



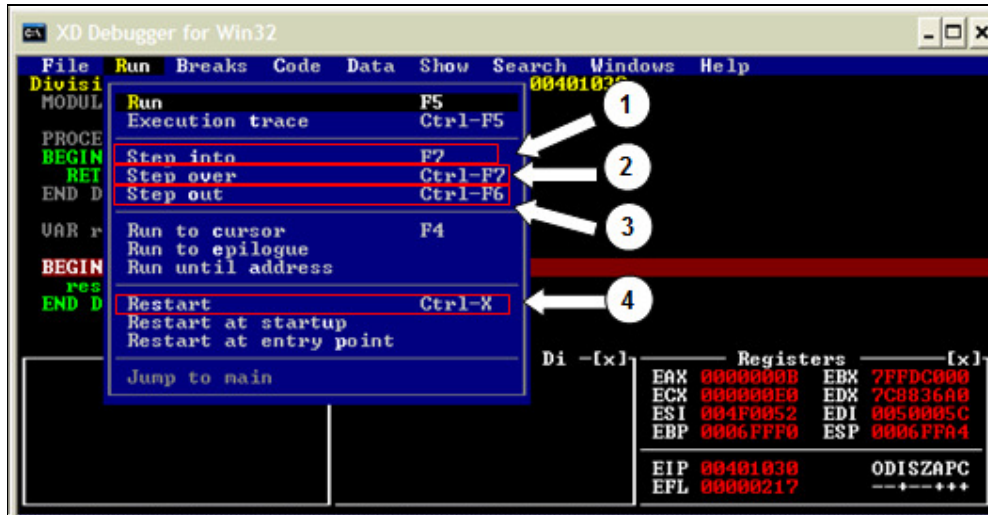
El código que se muestra en la pantalla cuando se ejecuta el debugger, corresponde al código del programa principal del programa que se esta examinando. Cuando inicia, el debugger se posiciona en la primer sentencia del programa principal, listo para que se le indique qué hacer. La barra roja indica donde se encuentra el control del programa.

En el caso del ejemplo que se muestra en la figura anterior, el código corresponde a un programa muy simple que ejecuta una división entre los enteros 1 y 2.

Se describen a continuación los comandos más relevantes para el uso del debugger.

## Descripción de los comandos más usados

### a. Step into, Step over y Step out



Como ya se dijo, una de las funcionalidades más importantes que tiene un debugger es la de poder controlar la ejecución del programa examinado. Por lo general, cuando se quiere encontrar un error en tiempo de ejecución, es útil poder ejecutar el programa sentencia por sentencia para poder determinar en cuál ocurre el problema. Para esto se usan comúnmente los comandos que se muestran recuadrados en la figura anterior y que se describen a continuación:

#### Recuadro 1 – Step into

Si se ejecuta este comando sobre una sentencia que es una invocación a función o procedimiento, el control de la ejecución “entra” en el código de la función o procedimiento invocado. Si se ejecuta sobre otro tipo de sentencia, el comportamiento es el mismo que se obtiene cuando se ejecuta el comando *Step over*.

#### Recuadro 2 – Step over

La ejecución de este comando produce la ejecución de la sentencia sobre la cual se aplica (la sentencia marcada con la barra roja).

#### Recuadro 3 – Step out

Cuando se ejecuta este comando, el control de la ejecución vuelve al invocador del código que esta siendo ejecutado por el debugger. En particular, cuando se ejecuta sobre una sentencia del programa principal provoca la finalización de la ejecución del programa examinado.

#### Recuadro 4 – Restart

Este comando le indica al debugger que vuelva a iniciar la ejecución del programa examinado desde el principio.

#### **Ejemplo:**

Considere el siguiente fragmento de código:

1. MODULE Ejemplo;
2. FROM STextIO IMPORT WriteString;
3. PROCEDURE Div(a,b: INTEGER): INTEGER;
4. BEGIN
5. RETURN a DIV b

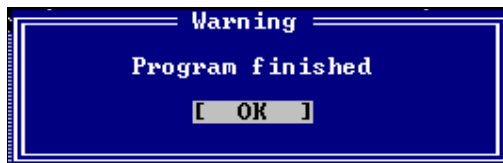
6. END Div;
7. VAR res: INTEGER;
8. BEGIN
9. res:=Div(1,4);
10. WriteString("Fin programa muy simple");
11. END Ejemplo.

Considere ahora que se desea examinar este programa en tiempo de ejecución, para lo cual se ejecuta el debugger de XDS. Al iniciar se muestra la siguiente pantalla:

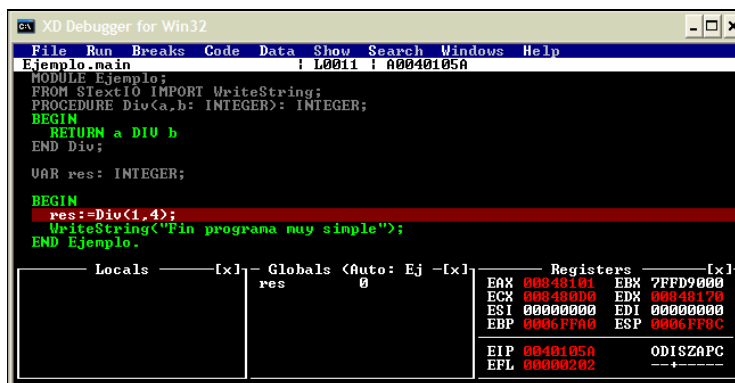


Si en este escenario se ejecuta:

- *Step over*: se ejecuta la sentencia 8 y el control se transfiere a la sentencia 9.
- *Step into*: se ejecuta la sentencia 8 y el control se transfiere a la sentencia 9, dado que la sentencia BEGIN no es una invocación a función o procedimiento.
- *Step out*: finaliza la ejecución del programa, mostrando un mensaje como el siguiente:



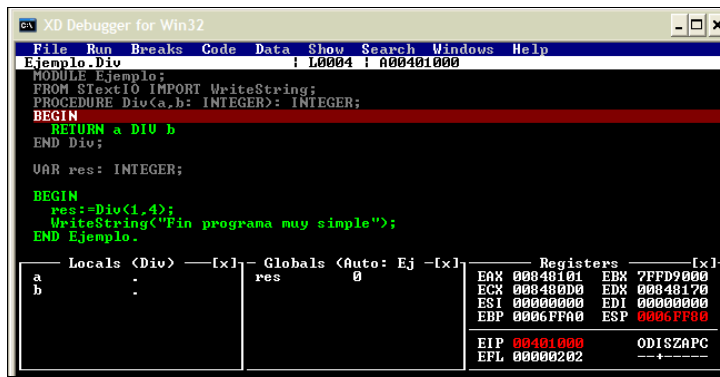
Suponga que se ejecuta *Step over*. La pantalla del debugger mostrará lo siguiente:



Ahora, si en este escenario se ejecuta:

- *Step over*: se ejecuta la sentencia 9 y el control se transfiere a la sentencia 10.
- *Step into*: el control se transfiere a la primer sentencia del procedimiento Div, o sea la sentencia 4.
- *Step out*: finaliza la ejecución del programa, de igual manera que se mostró antes.

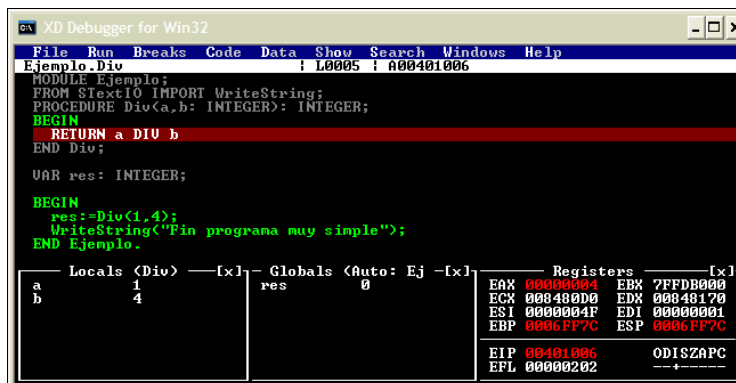
Si se elige *Step into*, la pantalla del debugger mostrará:



Si ahora se ejecuta:

- *Step over*: se ejecuta la sentencia 4 y el control se transfiere a la sentencia 5.
- *Step into*: se ejecuta la sentencia 4 y el control se transfiere a la sentencia 5.
- *Step out*: se ejecuta el resto del código del procedimiento Div y el control retorna a la sentencia 10.
- *Restart*: el programa se inicia de nuevo, el control vuelve a la sentencia 8.

Cada vez que el debugger detiene la ejecución del programa examinado es posible conocer el valor que tienen cargado en ese momento las variables del programa (entre otras cosas). Por ejemplo, en la imagen anterior donde el control del programa se encuentra en la sentencia 4, el debugger muestra que la variable *res* tiene cargado el valor 0, y que las variables *a* y *b*, locales al procedimiento *Div*, no tiene aun un valor asignado. Si en esta situación se ejecutara *Step over*, se ejecutaría la sentencia 4 y el control avanzaría a la sentencia 5. En este nuevo momento las variables *a* y *b* ya tendrán cargados los nuevos valores, 1 y 4 respectivamente, tal cual como se muestra en la siguiente imagen:



Es posible seguir examinando el programa hasta encontrar el error que generó la necesidad de usar el debugger.

## b. Breakpoint

El *breakpoint* es otra forma de controlar la ejecución del programa examinado. Es una señal que le indica al debugger que interrumpa la ejecución del programa en un punto determinado. Esta funcionalidad se encuentra en el menú, en *Breaks/Breakpoint*, como muestra la siguiente imagen:





Para usarla, se elige la sentencia donde se quiere detener la ejecución del programa y luego se selecciona *Breaks/Breakpoint*. A continuación, por ejemplo, se le indica al debugger que ejecute el programa automáticamente (opción del menú: *Run/Run*) y este se ejecutara hasta encontrar la sentencia que tiene el breakpoint, donde interrumpirá la ejecución.

## 5 Errores de Compilación

En el siguiente link se encuentran los códigos de los mensajes de error que muestra el compilador, así como la descripción asociada para cada uno de ellos. Esta información es de utilidad para corregir los errores de compilación de los programas que se generen. Esta información se encuentra en:

<http://www.fing.edu.uy/inco/cursos/prog2/info/docXDS/xs06.html#170>