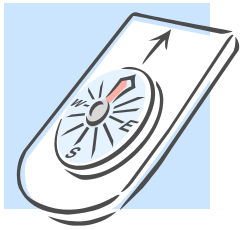
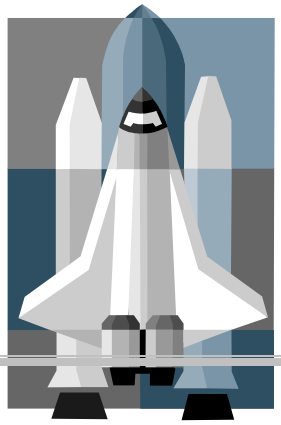
Cuaderno de Prácticas
Fundamentos de Informática para la Ingeniería con FORTRAN 90

Curso 2009 / 2010



Índice de contenidos



Introducción

El presente cuaderno incluye un conjunto de prácticas de programación orientadas a los alumnos de la asignatura Fundamentos de Informática para la Ingeniería de la titulación de Ingeniero de Caminos, Canales y Puertos de la Universidad de Granada.

Los ejercicios aquí propuestos están preparados para un aprendizaje sostenido y fundamentado de las habilidades de programación que se enseñan en la asignatura. Para poder afrontarlos con solvencia, el alumno debe haber estudiado con profundidad los conceptos necesarios para la realización de cada una de las sesiones, explicados previamente en las clases de teoría.

Cada sesión propuesta empieza con un breve recordatorio de los conceptos de programación que se pretenden trabajar con la práctica. Luego sigue un conjunto de ejercicios organizados en cuatro bloques:

- Toma de contacto: uno o varios ejercicios sencillos que permiten utilizar por vez primera los conceptos de programación objetivo de la práctica.
- Corrección de errores: el alumno se enfrenta a la corrección de errores estratégicamente insertados en un listado de código que pretende resolver una tarea concreta.
- El problema: durante las distintas sesiones se propone la implementación incremental de un programa de cierta complejidad. En nuestro caso, el alumno irá construyendo una solución cada vez más sofisticada para jugar al conocido juego de cartas de las *siete y media* contra el ordenador.
- Ejercicios opcionales: un conjunto de ejercicios para el alumno aventajado.

Los objetivos cubiertos por cada una de las sesiones son los siguientes:

- Acceso a las aulas de prácticas y habilidades básicas con el sistema operativo Windows
- Conocimiento del entorno de desarrollo Microsoft Developer Studio – Fortran Power Station
- Estructura general de un programa
- Estructuras condicionales
- Estructuras iterativas
- Vectores y matrices
- Subprogramas
- Ficheros. Formato general de Entradas y Salidas

Tres apéndices se pueden encontrar al final del cuaderno:

- Uno destinado a presentar las capacidades de depuración del entorno de desarrollo.
- Otro con material adicional para una orientación rápida en el entorno de desarrollo.
- Una práctica adicional para aquellos alumnos que deseen ampliar el trabajo desarrollado en el conjunto de las prácticas.

Las Aulas de Prácticas

En este apartado se pretende explicar de forma detallada todos los pasos que el alumno debe seguir para arrancar el entorno Microsoft ® Developer Studio - Fortran Power Station 4.0 para la realización de las prácticas.

No disponemos de unidades de disco de las que no se elimine todo el contenido al arrancar la máquina. Esto implica que al inicio de cada sesión habrá que repetir todo el proceso y, lo que es más importante, que en caso de estar trabajando, si no se ha salvado el trabajo en un disco de 3 ½ (disquete o floppy disk), o en una memoria USB, se perderá todo.

SERÁ OBLIGATORIO QUE CADA ALUMNO LLEVE A LAS PRÁCTICAS SU PROPIO DISCO DE 3 ½ (O MEMORIA USB), PARA GUARDAR SUS FICHEROS, Y SERÁ SU RESPONSABILIDAD EL SALVARLOS CADA CIERTO TIEMPO A FIN DE EVITAR PÉRDIDAS INDESEADAS.

Los pasos que hay que seguir son los siguientes:

- Arrancar el PC e identificarse. Utilizar el usuario que tengamos en la Universidad, añadiendo .docencia al usuario. Por ejemplo, para un usuario cuyo login (nombre de usuario) sea juanpl, tendría que identificarse como:
    - juanpl.docencia
    - su_password
- Tras el proceso de arranque, en el cual la máquina cargará una imagen de Windows XP, hacer doble clic sobre “Inicio”. Mover el ratón hasta el icono llamado “Internet Explorer” y hacer un clic sobre dicho elemento del menú. Una vez se abra el navegador, escribir una de las siguientes direcciones (esperar instrucciones del profesor sobre cuál utilizar) y pulsar ENTER:
    - http://flanagan.ugr.es/fortran/
    - http://decsai.ugr.es/~jbaena/fortran/
    - http://talento.ugr.es/fortran/
- Seguir los pasos necesarios para almacenar el fichero FIPI.exe en el escritorio del ordenador. Cerrar el explorador.
- En el escritorio veremos que hay un nuevo icono con el nombre “FIPI.exe”. Hacer doble clic sobre dicho icono.
- Seguir los pasos que se indican para instalar el programa (si hay dudas, preguntar al profesor).
- Si se han seguido todos los pasos con precisión, ya tendremos abierto y funcionando el entorno para la realización de las prácticas. En caso de apagado de la máquina (porque haya que reiniciarla al colgarse, o porque se vaya la luz, etc.) recordar que hay que repetir todo el proceso, y que los datos de programas que haya hecho el alumno se perderán (a no ser que los salve regularmente en su disquete).

El Sistema Operativo

1 Windows

Windows es un sistema operativo gráfico creado por la empresa Microsoft. Su diseño está orientado a permitir un manejo sencillo e intuitivo del ordenador. Como cualquier sistema operativo está encargado de controlar y dirigir el funcionamiento del ordenador, traduciendo las instrucciones que generan los demás programas en ejecución a un lenguaje que los componentes físicos del ordenador (hardware) puedan comprender.

2 Manejo del ratón

En Windows prácticamente todo ha sido diseñado para ser utilizado con el ratón. Al arrancar aparece en la pantalla un pequeño objeto móvil llamado puntero o cursor, cuya posición está relacionada con el movimiento del ratón. Aunque la mayoría de las operaciones se pueden realizar también desde el teclado, el ratón proporciona el modo más directo, cómodo e intuitivo de interactuar con los elementos que aparecen en la pantalla.

El ratón de Windows utiliza dos botones fundamentales, aunque pueden encontrarse en el mercado una amplia gama de modelos que incorporan elementos adicionales a estos dos botones.

Las operaciones que se pueden hacer con el ratón implican una o varias de las combinaciones siguientes:

- Apuntar: llevar el puntero a un lugar concreto de la pantalla desplazando el ratón sobre la mesa o la alfombrilla.
- Hacer clic: pulsar y soltar un botón del ratón.
- Arrastrar: manteniendo pulsado un botón del ratón, mover éste a un lugar diferente y luego soltar el botón.
- Hacer doble clic: pulsar y soltar el botón dos veces seguidas rápidamente.

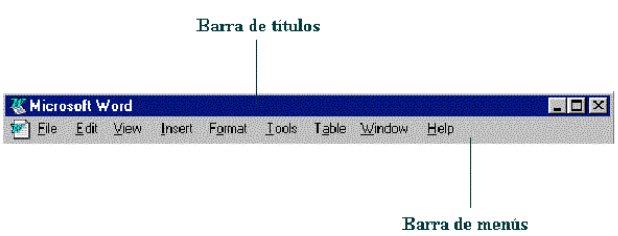
El botón derecho del ratón se utiliza principalmente para activar el menú contextual, un menú que contiene los comandos correspondientes a las operaciones más frecuentes que se pueden realizar con el tipo de objeto sobre el que se ha pulsado.

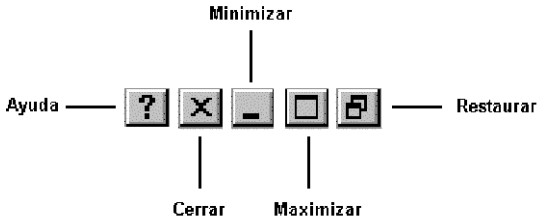
Dependiendo de dónde esté situado y de la tarea que está realizando el ordenador, el puntero puede adoptar diferentes formas: una flecha orientada hacia la izquierda y ligeramente hacia arriba, una barra vertical en forma de I, un reloj de arena ...

3 Manejo de ventanas

Una ventana es una parte de la pantalla dedicada a interactuar con la ejecución de una aplicación o utilidad concreta. Hay muchos tipos de ventanas: ventanas de aplicación, de documentos, de ayuda, cuadros de diálogo, etc…

Algunos de los elementos que suelen aparecer en una ventana son:



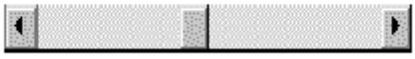


- Barra de títulos: Muestra el nombre de la ventana. Con mucha frecuencia el nombre de la ventana contiene el nombre de la aplicación abierta en ella, seguido del nombre del documento activo.
- Barra de menús: Inmediatamente debajo de la barra de títulos de la mayoría de las ventanas, hay una banda horizontal llamada Barra de Menús que contiene nombres tales como Archivo(File), Edición (Edit) o Ayuda (Help). Haciendo clic en cualquiera de estos nombres se despliega un menú en forma de persiana, es decir se despliega una lista de comandos. Para escoger uno, basta con desplazar el puntero del ratón sobre el comando correspondiente y hacer clic.
- Botón de minimizar (Minimize): Haciendo clic sobre este botón la ventana se reduce y se coloca su nombre en una barra que está en la parte inferior de la pantalla denominada Barra de Tareas (Taskbar).
- Botón de maximizar (Maximize): En este caso, al presionar el botón, la ventana aumenta de tamaño hasta ocupar la totalidad de la pantalla.
- Botón de restaurar (Restore): Una vez maximizada la ventana, el botón de maximizar cambia al de restaurar. Presionando éste, la ventana vuelve al tamaño que poseía antes de ser maximizada.
- Botón de cerrar (Close): Cierra una ventana y la aplicación que está abierta. Suele estar en la esquina superior derecha o bien en la esquina superior izquierda en forma de un pequeño icono correspondiente a la aplicación.
- Botón de Ayuda (Help): Este botón que aparece en la esquina superior derecha de muchas de las cajas de diálogo, sirve para que Windows muestre información acerca de un elemento de la pantalla. Para ello hacer clic sobre el botón y arrastrar el cursor transformado en un signo de interrogación sobre el objeto de la pantalla que se desconoce o del que se desea obtener una breve explicación.

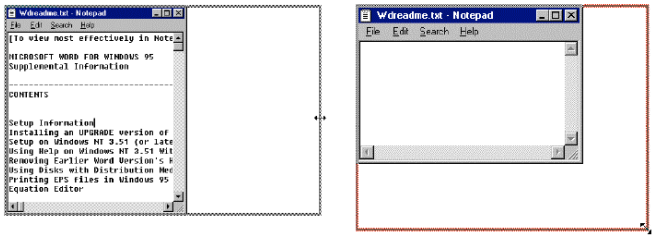
Debe tenerse en cuenta, que cuando se minimiza una ventana, ésta sigue abierta, es decir, la ventana y cualquier programa que se está ejecutando en ella siguen disponibles en la memoria del ordenador y pueden visualizarse nuevamente. Si por el contrario se cierra una ventana, esta desaparece de la memoria del ordenador y para volverla a visualizar es necesario arrancar de nuevo la aplicación.

Si se desplaza el cursor hacia los bordes de una ventana, éste se transforma en una doble fecha (horizontal o vertical, según de qué borde se trate) permitiendo, si se arrastra con el botón del ratón pulsado, modificar el tamaño de dicha ventana. Si el cursor se coloca en alguna de las cuatro esquinas toma la forma de una doble flecha inclinada, permitiendo modificar el tamaño de los dos lados de la ventana que confluyen en esa esquina.

Cuando se reduce una ventana a un tamaño en el que no se puede mostrar toda la información, aparecen unos elementos nuevos denominados Barras de Desplazamiento (Scrolling Bars), que permiten ver todo el contenido de la ventana sin variar su tamaño. Existen dos barras de desplazamiento, horizontal y vertical, aunque no tienen por qué aparecer ambas simultáneamente. La operación de desplazarse por la ventana se puede realizar, bien pulsando en una de las dos flechas de los extremos – pequeño avance -, pulsando en la propia barra - gran avance -, o arrastrando el rectángulo existente entre ellas a voluntad del usuario.



Cuando se trabaja con Windows, a menudo se querrá tener en la pantalla más de una ventana abierta a la vez, y cambiar de una a otra. Para mover una ventana de posición se debe arrastrar el puntero después de haberlo situado sobre la barra de títulos.



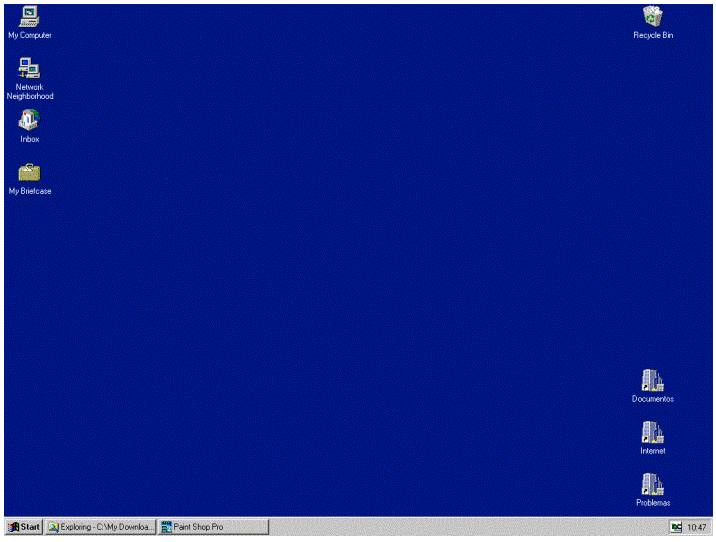
Para conmutar entre dos ventanas se debe tener en cuenta que la ventana activa posee la Barra de Títulos resaltada (de color) mientras que la otra está en un tono menos llamativo (normalmente en color gris). Con Windows es particularmente fácil conmutar ventanas. Para ello hay dos métodos principalmente: haciendo clic sobre la ventana que se desea volver activa o utilizando la Barra de Tareas (TaskBar).

Para poder realizar cualquier operación dentro de una ventana ésta debe estar activa. Cuando hay dos o más ventanas superpuestas en la pantalla, la ventana activa siempre aparece en primer plano.

4 El Escritorio (Desktop)

Al arrancar el PC suele aparecer una pantalla principal que puede presentar diferentes configuraciones, dependiendo de cómo se haya instalado Windows. Esta pantalla recibe el nombre de Escritorio o Desktop. En el Escritorio aparecen entre otros, una serie de iconos o símbolos que merece la pena destacar:

- Mi PC (My Computer). Un doble clic en este icono hace aparecer una ventana con dispositivos de almacenamiento (disco duro, una o más unidades de disquete y una unidad de CD-ROM - en caso que existiera -) y/o de impresión a los que se tiene acceso. A través de ellos se puede acceder a toda la estructura de directorios y ficheros.
- Entorno de red (Network Neighborhood). Pulsando dos veces sobre este icono se muestran, en una ventana, los recursos disponibles de la red (si se está conectado a una red y se tiene permiso para moverse sobre ella).
- Papelera de Reciclaje (Recycle Bin). Lugar de almacenamiento temporal de archivos borrados, pero que pueden ser recuperados nuevamente mientras no se haya 'vaciado' la papelera. Para borrar un fichero o directorio basta con arrastrarlo a la papelera.
- Botón de Inicio (Start Button). Pulsando sobre este botón se pueden abrir documentos y programas, ejecutarlos, acceder a la ayuda, etc.



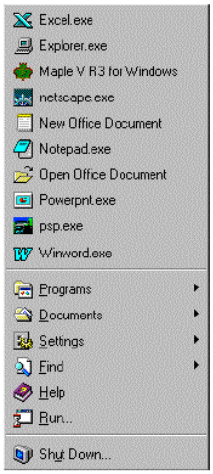
5 El Menú de Inicio (Start menu)

El menú de Inicio (Start) es la estructura más importante de Windows. Según el tipo de instalación seleccionada, se incorporarán en el mismo los programas suministrados con Windows e incluso los programas ya instalados en versiones anteriores. También pueden incorporarse al menú los programas que se instalen posteriormente.

Al pulsar con el botón izquierdo del ratón sobre el botón Inicio (Start), se despliega un menú en forma de persiana que posee una serie de contenidos o elementos. Algunos de estos ítems o elementos tienen un pequeño triángulo en su parte derecha para indicar que al posicionar el cursor sobre éste, se desplegará un nuevo menú. Estos menús se denominan "menús en cascada".

En la tabla siguiente se explica brevemente lo que es cada uno de los elementos que aparecen al desplegar el menú.

Comando	Descripción
Programas (Programs)	Muestra una serie de programas que pueden ser ejecutados.
Documentos (Documents)	Almacena la lista de los documentos abiertos recientemente.
Configuración (Settings)	Presenta una lista de utilidades que permiten modificar los parámetros del sistema (Panel de control, impresoras,…).
Buscar (Find)	Aplicación que permite buscar archivos, directorios u otros ordenadores en la red.
Ayuda (Help)	Se selecciona la ayuda de *Windows*.
Ejecutar (Run)	Ejecuta un programa.
Apagar sistema (Shut Down)	Cierra la sesión *Windows*. Se utiliza antes de apagar el ordenador o para comenzar una nueva sesión.



6 Ejecución y salida de un programa

Para ejecutar un programa a través del menú de inicio se debe presionar con el botón izquierdo del ratón sobre el botón Inicio y seguidamente seleccionar Programas. Se abrirá otro menú sobre el que se puede elegir una nueva opción, correspondiente a un grupo de programas, por ejemplo Accesorios y finalmente se pulsa sobre el programa que se desea ejecutar, por ejemplo WordPad.

También se puede elegir el comando Inicio/Ejecutar (Start/Run) y luego especificar el programa ejecutable correspondiente. Un programa arranca también automáticamente al pulsar dos veces sobre un fichero o documento creado con él o sobre su fichero ejecutable.

Cuando se termine de utilizar la aplicación, para finalizar con el programa y cerrar definitivamente la ventana, hay que presionar, bien sobre el icono de cierre (ilustración 3) o bien pulsar dos veces consecutivas en la parte superior izquierda de la ventana.

7 La Ayuda (Help)

Uno de los componentes más importantes de Windows es la Ayuda o Help, que proporciona en cualquier momento asistencia sobre cualquier tema. También se puede aprender con ella la forma de realizar cualquier tarea concreta. Para acceder a la ayuda hay que pulsar Inicio/Ayuda (Start/Help).

Aparecerá una ventana que contiene tres hojas seleccionables por medio de unas lengüetas, Contenido, Indice y Buscar (Contents, Index y Find), a las que se puede acceder pulsando en los correspondientes separadores de la parte superior. Cada una de estas fichas permite el acceso a numerosos temas, pero con un método distinto en cada caso.

La primera de las hojas citadas, llamada Contenido (Contents), muestra el contenido general disponible sobre un tema concreto. Según se vaya pulsando en cada uno de los temas, irá apareciendo el contenido del mismo en forma de libros, hasta llegar al tema que se desee consultar.

Al pulsar en la lengüeta Indice (Index) aparecerá un lista de temas y subtemas por orden alfabético. Para seleccionar una opción concreta puede moverse a través de ella gracias a las barras de desplazamiento o escribir la palabra o palabras del tema del cual se desea obtener ayuda en el cuadro de la parte superior de la ficha. La información de las ventanas de ayuda suele ser por lo general concisa. Algunas de estas ventanas incluyen botones de acceso directo que llevan directamente a un cuadro de diálogo determinado.

La ficha Buscar (Find) permite localizar ayuda mediante entradas que contienen palabras concretas, funcionando por lo demás de forma similar a la ficha Índice (Index).

Existen además dos tipos de ayudas distintas de las anteriores y que vale la pena destacar: las etiquetas o microayudas que se presentan al colocar el puntero del ratón sobre los botones de la barra de herramientas durante un pequeño espacio de tiempo y la posibilidad de obtener ayuda acerca de una determinada parte de la ventana, en la que aparezca el símbolo de interrogación en la parte superior derecha de la pantalla. En este segundo caso, basta presionar el botón de interrogación y pulsar con el nuevo cursor - en forma de signo de interrogación - sobre cualquier elemento de la ventana o ficha de la que se desee una breve descripción.

8 Buscar archivos (Find)

El comando Buscar (Find) que aparece tras presionar el botón Inicio (Start), permite localizar de una manera rápida y cómoda cualquier archivo o carpeta en algún lugar del árbol de unidades, en lugar de ir mirando de forma manual, una por una, todas las carpetas.

La búsqueda más sencilla de todas, es la búsqueda por Nombre de Archivo. Windows puede encontrar archivos aunque no se conozca el nombre completo. Para ello, tras arrancar la opción Buscar (Find), se presenta un cuadro de diálogo en el que hay tres separadores.

El primero de ellos se utiliza para buscar archivos por nombre completo o mediante un conjunto de caracteres que forman parte del nombre del archivo. Para ello debe introducirse en el cuadro Nombre (Name) el nombre del archivo, y en el cuadro Buscar en (Look in), la unidad en la que se va a realizar la búsqueda y presionar el botón Buscar ahora (Find Now). Si se quiere que Windows localice ficheros en las diferentes subcarpetas habrá que activar la opción Incluir Subcarpetas (Include Subfolders).

Sin embargo, la capacidad de la opción Buscar (Find) de Windows no termina ahí. Hay información adicional que puede ser útil para localizar un determinado fichero, incluso si se desconoce el nombre del mismo. Para este tipo de búsqueda están las lengüetas Fecha de Modificación (Date Modified) y Avanzadas (Advanced). La primera permite localizar archivos entre dos fechas distintas y durante los “n” meses o días anteriores a la fecha actual. Basta introducir para ello los datos apropiados en los cuadros de diálogo.

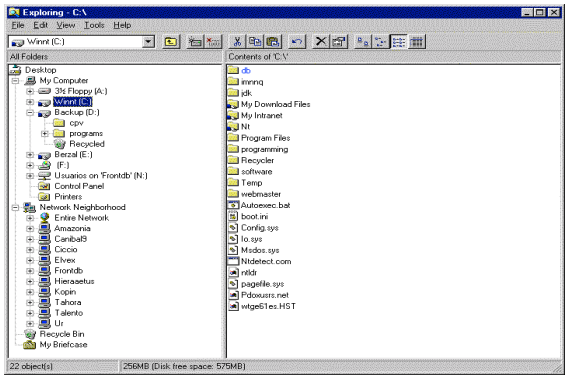
La lengüeta Advanced (Avanzadas) localiza todos los archivos que contengan una determinada cadena de caracteres y posean un determinado tamaño en Kilobytes. De forma similar al caso anterior, basta con introducir los datos en cada cuadro de diálogo y presionar el botón.

Es posible ejecutar o abrir directamente los archivos que aparecen en lista devuelta por el proceso de búsqueda mediante un doble clic.

9 El Explorador de Windows

El Explorador de Windows (Windows Explorer) nos permite movernos y trabajar con todo lo que tenemos en nuestro ordenador. Este programa se puede ejecutar desde el menú de Inicio o accediendo al menú contextual que se abre haciendo clic con el botón derecho del ratón sobre el icono Mi PC (My Computer).

En la figura se muestra el aspecto típico que presenta la ventana del Explorador. Esta ventana está dividida en dos partes:



La parte situada a la izquierda contiene, a modo de estructura en árbol, las unidades de disco y las carpetas (folders) del ordenador. El ordenador y las unidades de disco son, metafóricamente, las raíces y el tronco de un árbol, las carpetas sus ramas, las subcarpetas otras ramas más pequeñas y los archivos sus hojas.

En el panel localizado a la derecha de la ventana del Explorador, aparece el contenido de la carpeta que tengamos seleccionada.

Si al lado del icono de una unidad o carpeta aparece un signo más (+), significa que esta carpeta contiene a su vez una o varias subcarpetas que no se visualizan. Para expandir un nivel más y visualizar su contenido, debe hacerse clic con el ratón sobre el signo más. Una vez expandida la rama del árbol, el signo más cambia por un signo menos y, si pulsamos con el ratón sobre él, se contrae toda la rama. En cada rama expandida del árbol se presentan las subcarpetas sangradas para indicar su subordinación y unidas por líneas que indican cómo están relacionadas entre sí.

El Explorador ofrece muchas posibilidades para el manejo de archivos y directorios, tales como mover, copiar, cambiar el nombre, crear o eliminar directorios y ficheros.

Crear archivos o carpetas

Para crear una nueva carpeta o un archivo se selecciona primero el lugar en el árbol de directorios donde se quiere que aparezca el nuevo documento o carpeta. Posteriormente, en el menú se elige Fichero/Nuevo (File/New) y en el submenú que aparece se decide si va a ser un nuevo documento o una carpeta. Utilizando el botón derecho en la parte derecha de la ventana del Explorador aparece un menú contextual con el que también se puede crear una nueva carpeta o fichero.

Copiar y mover archivos o carpetas

Con el Explorador de Windows también es posible copiar y mover archivos. Para copiar o mover un archivo se puede utilizar la técnica "arrastrar y soltar" (drag and drop). Para ello, el archivo seleccionado se arrastra con el ratón -manteniendo pulsado el botón izquierdo- desde su localización original, hasta el directorio de destino. Ahí se coloca soltando el botón del ratón. Copiar y mover archivos son operaciones diferentes, aunque algo similares. Cuando se mueve, el archivo se traslada desapareciendo de su posición original. Al copiar, el archivo permanece donde estaba y se crea una copia idéntica en el destino. Dentro de una misma unidad o disco, la opción por defecto es mover (para copiar hay que mantener pulsada la tecla CTRL mientras se arrastra el fichero). Entre discos distintos la opción por defecto es copiar (para mover hay que mantener pulsada la tecla SHIFT o MAYÚSCULAS). Se puede saber qué operación se está realizando porque al copiar aparece un signo más (+) junto al icono del fichero que se está arrastrando.

Otra forma de copiar y mover ficheros es utilizar el Portapapeles de Windows (Clipboard). Se elige Cortar (Cut) o Copiar (Copy) según queramos mover o copiar, se va uno al directorio destino, y se pega el contenido del portapapeles en la carpeta deseada (Pegar-Paste). Las operaciones que se pueden realizar con el portapapeles suelen aparecer siempre en un menú llamado Editar o Edición (Edit) y en el menú contextual que aparece cuando pulsamos el botón derecho si tenemos seleccionados uno o más ficheros o directorios. Por otro lado, la combinación de teclas CRTL-X es equivalente a cortar, CRTL-C es equivalente a copiar y CRTL-V a pegar.

Cambiar el nombre y eliminar archivos

Se puede cambiar el nombre o borrar un fichero o directorio por medio de las órdenes Fichero/Cambiar nombre (File/Rename) y Fichero/Eliminar (File/Delete) respectivamente.

Otra forma de cambiar el nombre de un fichero es hacer clic en el correspondiente icono y, después, hacer clic de nuevo en su nombre. Aparecerá un cuadro alrededor del nombre resaltado y se verá un cursor intermitente. Entonces se puede escribir el nuevo nombre y pulsar ENTRAR (ENTER/RETURN).

Para eliminar un archivo o carpeta también se puede utilizar la barra de herramientas del Explorador (el botón con forma de aspa) o pulsar la tecla SUPR (DEL). Obviamente, antes hay que seleccionar los archivos que deseamos eliminar. Si la barra de herramientas del Explorador no estuviera visible, se puede activar por medio del comando Ver/Barra de Herramientas (View/Taskbar).

Para seleccionar varios archivos consecutivos se selecciona en el primero de ellos y luego en el último, manteniendo pulsada la tecla SHIFT (MAYÚSCULAS). Para seleccionar varios archivos no consecutivos, se selecciona cada uno de ellos manteniendo pulsada la tecla CTRL. Si en el momento de borrar se tiene pulsada la tecla SHIFT, los ficheros seleccionados se borran directamente, sin pasar por la papelera de reciclaje (Recycle Bin).

Uso de disquetes

Para conservar los ficheros que se manejan en las prácticas es necesario copiarlos en disquetes de tres pulgadas y media. Un disco ha de tener formato para que se pueda almacenar información en él. Para formatear una unidad, se puede utilizar la opción correspondiente que aparece en el menú contextual de la unidad (haciendo clic con el botón derecho del ratón sobre el icono de la unidad).

Con la opción Propiedades [Properties] del mismo menú contextual se puede obtener información general de la unidad de disco: etiqueta, capacidad, espacio utilizado, espacio libre...

10 La Barra de Tareas (Taskbar)

Cada vez que se arranca una nueva aplicación o se abre una ventana, aparece un nuevo botón representando a la aplicación, en una barra situada en la parte inferior de la pantalla, llamada Barra de Tareas (Taskbar). Así, si se tienen varias aplicaciones abiertas, existirá un botón por cada una de ellas. Aquella que esté activa en ese momento se reconocerá porque el botón correspondiente a la aplicación estará presionado y la ventana se mostrará abierta en el Escritorio. Si se desea activar una nueva aplicación, basta con pulsar en el botón correspondiente para que inmediatamente se active ésta en una ventana y la anterior deje de estar activa.

Cuando hay muchas ventanas o programas abiertos, los botones en la barra de tareas son demasiado pequeños para poder mostrar el nombre completo de todas las aplicaciones. Si se desea ver el nombre completo de un programa que aparece en forma de botón, basta con mover el puntero del ratón sobre él y dejarlo quieto durante aproximadamente un segundo. Aparecerá una pequeña etiqueta en la que se mostrará el nombre completo.

11 La Papelera o Recycle Bin

La Papelera de Reciclaje (Recycle Bin) de Windows es una zona para almacenar archivos borrados. De forma predeterminada, todos los archivos borrados van a parar primero a la papelera y ahí esperan su eliminación definitiva. Para enviar archivos a la papelera, una vez seleccionados, se pueden arrastrar éstos desde una unidad o carpeta hasta el icono de la papelera.

Cuando en la papelera se encuentran muchos ficheros y el espacio en disco duro está muy ajustado, pueden presentarse problemas. Hay que tener en cuenta que este almacenamiento intermedio se mantiene más allá de la sesión actual de Windows. Pero en el caso de que se borre un archivo accidentalmente, la papelera se convertirá en un verdadero salvavidas. Sin ninguna dificultad se podrá recuperar el documento que se tiró accidentalmente a la papelera.

Otra posibilidad que ofrece la papelera es la de ajustar su tamaño. Para ello, pulsar con el botón derecho del ratón sobre su icono y elegir en el menú contextual la opción Propiedades (Properties). Debe tenerse en cuenta que, si se envía un archivo a la papelera cuando ésta está llena, Windows eliminará los archivos necesarios, comenzando por los más antiguos, hasta que posea espacio suficiente para el/los nuevo/s archivos enviados.

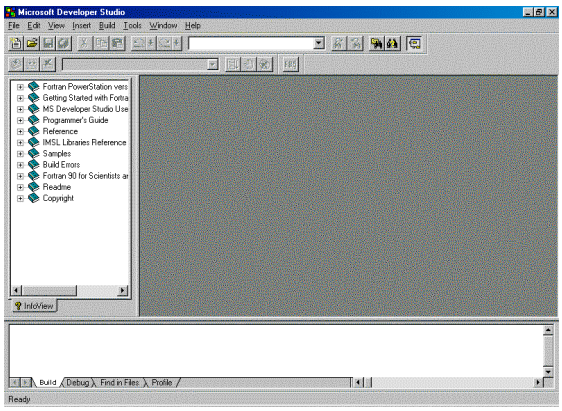
Si se desea recuperar un archivo, se pulsa dos veces sobre el icono de la papelera y se localiza el archivo en la lista. En el menú de archivo (File) se elige entonces la opción Recuperar (Restore). Se puede vaciar la papelera (borrando de forma definitiva su contenido) con la opción Vaciar la Papelera (Empty Recycle Bin).

12 Salir de Windows

Cuando se ha terminado de trabajar con el ordenador y se desea apagarlo, es necesario salir de Windows de una forma ordenada ya que, de lo contrario, se pueden dañar algunos archivos del programa necesarios en una nueva sesión, pudiéndose incluso perder información muy valiosa para el usuario. Para salir de Windows se debe elegir, tras presionar el botón Inicio (Star), la opción Apagar el Sistema (Shut Down the Computer). Aparece un cuadro de diálogo que muestra una serie de opciones para apagar o reiniciar el sistema. Si elegimos apagar y aceptamos, tras unos segundos de espera, Windows presentará un mensaje en pantalla advirtiendo al usuario que ya se está en condiciones de desconectar el ordenador.

El Entorno de Desarrollo

La apariencia del entorno de desarrollo MSDS-FPS4.0 cuando comienza es la siguiente:



1 Tipos de ficheros

El contenido de los distintos tipos de ficheros viene dado por la extensión del nombre (3 caracteres tras el punto).

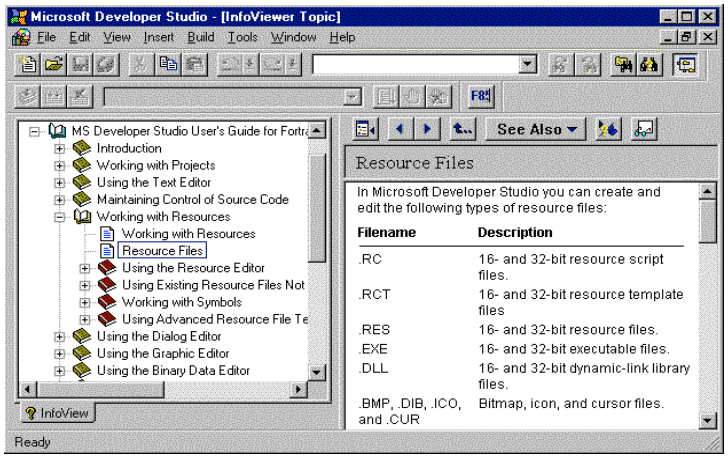
En las prácticas prestaremos especial atención a los ficheros con extensión .F90 (o bien .F o .FOR) que contienen programas; los ficheros con extensión .MDP y .MAK que contienen información sobre proyectos y los ficheros con extensión .EXE que contienen el programa final compilado y enlazado listo para ejecutar. Algunos de los tipos de ficheros que se manejan con el MSDS-FPS4.0 se muestran en la siguiente tabla.

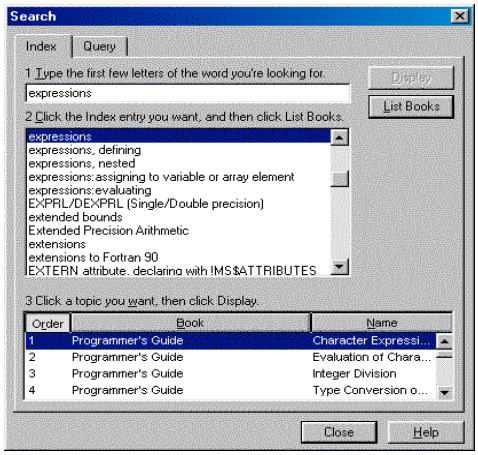
Extensión	Tipo	Descripción / Contenido
MAK	*Project Makefile*	Contiene: • Los nombres de los ficheros fuente que se han de usar para construir el proyecto. • Las opciones necesarias para compilar y enlazar el proyecto.
MDP	*Project Workspace*	Contiene: • Configuración del entorno MSDS-FPS4.0 (apariencia). • Breakpoints marcados. • Referencia al fichero .MAK del proyecto.
EXE	*Ejecutable*	Pseudo-código máquina ejecutable.
F90	*Código fuente F90*	Programa escrito en Fortran 90 (formato libre).
F o FOR	*Código fuente F90*	Programa escrito en Fortran 90 (formato fijo).
ILK	*Fichero auxiliar*	Fichero intermedio que se genera durante la fase de enlazado (link).
OBJ	*Fichero objeto*	Fichero intermedio generado tras la fase de compilación.
PDB	*Fichero auxiliar*	Fichero intermedio que contiene información para poder realizar la depuración del programa.

2 Ayuda on-line

InfoViewer es un sistema de ayuda integrado en MSDS-FPS4.0. A través de él puedes consultar on-line gran cantidad de información sobre el funcionamiento del propio entorno de desarrollo y sobre el lenguaje Fortran 90. Algunos de los apartados que incluye son los siguientes:

- Programmer's guide. Esta es una guía para aprender a programar en Fortran 90.
- Reference. Esta es la guía de referencia del lenguaje Fortran 90. Contiene una especie de diccionario con la descripción y ejemplos de todas las instrucciones existentes en el lenguaje.
- Build Errors. Descripción de todos los errores que podemos encontrarnos tanto en tiempo de compilación como de ejecución.
- Fortran 90 for Scientists and Engineers. Esta es la versión electrónica del libro del mismo nombre escrito por Brian Hahn.





InfoViewer es un sistema en el que la información se clasifica de forma jerárquica. También es posible realizar búsquedas rápidas pulsando Help->Search. Al pulsar esa opción nos aparece una venta que muestra un formulario que se puede utilizar para encontrar información a partir de algún término de búsqueda. En el apartado 1 (Type the first few letters of the word you're looking for) tecleamos la palabra sobre la que deseamos obtener información (puede ser una instrucción de Fortran o cualquier otra cosa relacionada con el lenguaje o con el entorno de desarrollo). Conforme vamos escribiendo en el apartado 1, en el apartado 2 (Click the index entry you want, and then click list books) nos aparecen aquellas palabras que comienzan por lo que hemos escrito y sobre las que nos puede ofrecer algún tipo de ayuda. Cuando aparezca en el apartado 2 lo que estamos buscando pulsamos 2 veces sobre él y en el apartado 3 (Click a topic you want, then click display) nos aparecerán todos aquellos puntos que puedan tener relación con el término de búsqueda. Pulsamos dos veces sobre el que nos interese y obtendremos la información buscada.

El uso de este sistema de ayuda puede ser especialmente útil para averiguar el significado de los distintos errores que se producen (tanto en tiempo de compilación como de ejecución) ya que escribiendo el código de error obtendremos información sobre lo que significa y las posibles causas del mismo.

Como muchos otros sistemas, este entorno de desarrollo de Microsoft incorpora una faceta de ayuda rápida denominada Tips of the day. Para acceder a ella, basta desplegar el menú Help. A través de esta forma de ayuda rápida el usuario puede acceder a consejos que le permiten sacar partido de una forma más eficaz del entorno de desarrollo. El usuario puede configurar el entorno para que, cada vez que lo inicie, se le muestre uno de estos consejos sin necesidad de tener que acceder al menú Help.

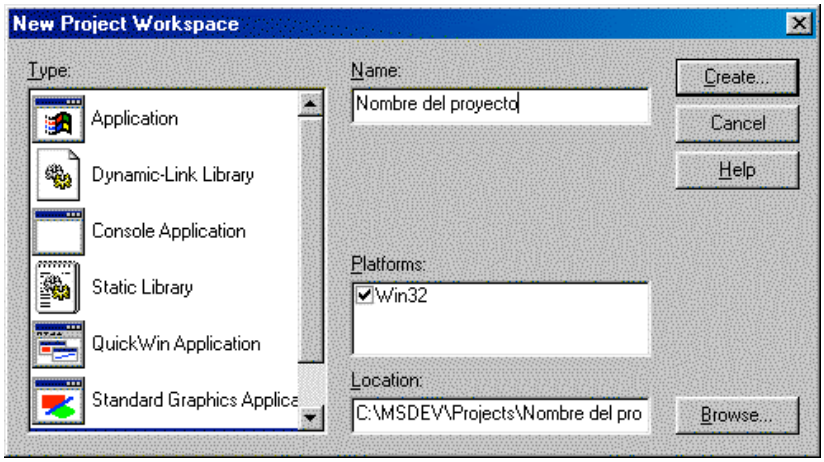
3 Project Workspace

El entorno de desarrollo MSDS-FPS4.0 nos va a permitir abordar proyectos de programación de gran envergadura. Estos grandes programas suelen estar compuestos de distintos módulos, cada uno de los cuales es independiente del resto y se almacena en distintos ficheros. Cuando vamos a compilar el programa global se recoge la información de todos esos ficheros y se generan otros ficheros con el programa listo para ejecutarse. Para organizar todos esos ficheros que componen nuestro programa se utiliza lo que se denomina Project Workspace o simplemente un proyecto.

En la zona denominada FileView disponemos de la información relativa a los ficheros que componen nuestro programa.

Para crear un nuevo proyecto podemos pulsar en "File->New->Project Workspace...". Tras esta operación nos aparece una ventana similar a la de la figura. En esta ventana especificamos cual será el nombre del proyecto (que será usado para nombrar al fichero final que contenga el programa ejecutable), el directorio en el que está toda la información relativa a él (location) y el tipo de proyecto (en la parte izquierda de la ventana.

La elección de un tipo de proyecto nos va a permitir definir, por ejemplo, la apariencia de nuestro programa.



Algunos de estos tipos se describen a continuación:

- Console Application. Permite crear programas que hacen uso de la consola MS-DOS, es decir sin interfaz gráfica y únicamente con posibilidades de escribir texto.
- QuickWin Application. Permite crear aplicaciones sencillas que hacen uso de una interfaz gráfica. Podemos disponer de varias ventanas para mostrar diferentes cosas.
- Standard Graphics Application. Es similar a la opción QuickWin salvo que aquí sólo se puede hacer uso de una única ventana para entrada/salida de datos (texto y gráficos).
- Application. Permite crear aplicaciones que hacen uso de todas las posibilidades que ofrece Windows. Las opciones QuickWin y Standard Graphics están bastante limitadas en muchos aspectos.

La creación de un proyecto es obligatoria aunque vayamos a crear un programa pequeño que únicamente necesite hacer uso de un fichero.

Los proyectos se guardan en ficheros con extensión .mdp. En estos ficheros se almacenan los nombres de todos los ficheros pertenecientes al proyecto así como el tipo de información que contienen y las relaciones entre ellos.

4 Cómo comenzar a escribir un nuevo programa

Podemos optar por dos modalidades para comenzar a escribir un nuevo programa. Con la primera de ellas (Opción A) nos vamos a desentender del concepto de proyecto y vamos a poder realizar programas sin ningún tipo de capacidad gráfica.

Con la segunda opción (Opción B) tenemos la posibilidad extra de crear programas que tienen la apariencia de una aplicación de tipo Windows y que además tienen la posibilidad de generar gráficos simples.

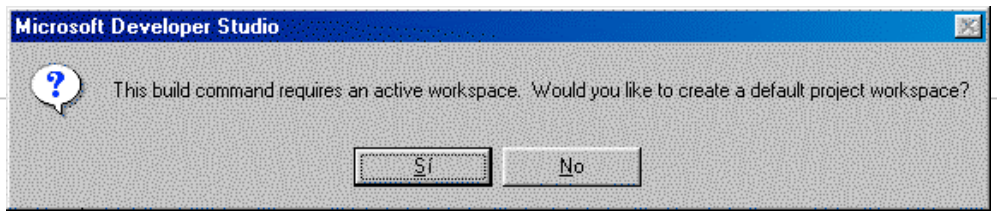
Opción A

La forma más simple de hacer un programa (en modo texto) sin preocuparnos demasiado sobre el concepto de proyecto es la siguiente:

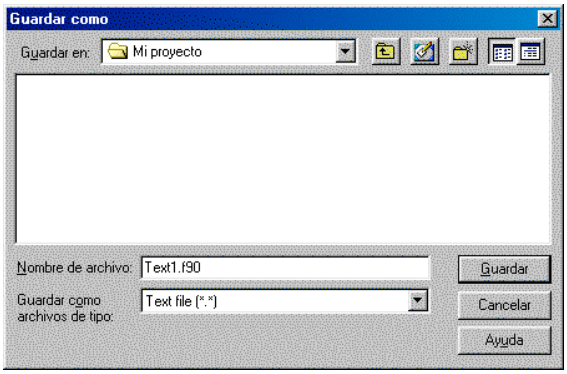
1. Pulsamos el botón New Source File o bien seleccionamos File->New...->Text file. Tras realizar esta operación aparece una ventana en blanco en la zona de trabajo.

2. A continuación podemos escribir nuestro programa en esa ventana que ha aparecido.

3. Una vez que hemos escrito el programa podemos optar por guardarlo en un fichero de disco o por pasar directamente a compilarlo. Cuando pasamos a la opción de compilar (Build->Compile o bien Build->Build), independientemente de si hemos guardado en disco el programa o no, nos aparecerá un mensaje como el de la figura. Este mensaje nos indica que se necesita de la existencia de un proyecto para poder compilar el programa. Si pulsamos "Si" se creará un proyecto por defecto para poder continuar.



4. A continuación, y únicamente en el caso de que no hayamos guardado en un fichero nuestro programa, se nos mostrará la ventana de título "Guardar como" en la que indicaremos donde queremos almacenar dicho programa.



5. Pasado este punto se procederá a la compilación y ejecución del programa (en el caso de que no haya errores). A partir de este momento cuando volvamos a compilar el programa no se nos volverá a preguntar ni por el proyecto ni por el nombre del fichero para guardar el programa (se usa el mismo que se usó la primera vez que lo guardamos).

6. La ejecución del programa, una vez compilado correctamente, se puede hacer de dos formas:

   a. Mediante la opción Execute del menú Build. En este caso se abre una venta de ejecución desde el entorno de desarrollo, se ejecuta el programa y, una vez terminada la ejecución, pulsando una tecla se cierra la ventana.

   b. Desde el explorador de windows (o una ventana de MS-Dos), activando la ejecución del archivo .exe correspondiente.

Opción B

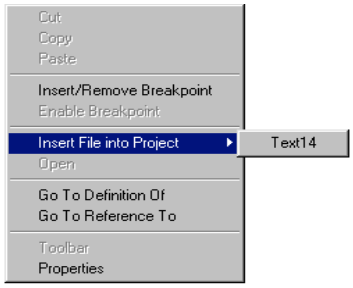
La segunda opción para empezar a escribir un nuevo programa se diferencia de la anterior en que lo primero que hacemos es definir el tipo de proyecto y después escribimos el programa.

1. Seleccionamos “File->New...->New project...”. Nos aparecerá la ventana en la que seleccionaremos el tipo de proyecto que vamos a hacer, el nombre del proyecto y la localización del mismo. Si vamos a hacer un programa que no tenga necesidades gráficas podemos seleccionar Console Application.

2. A continuación pulsamos “File->New...->Text file” o bien “New source file”. En la ventana que nos aparece escribimos el programa.

3. Guardamos el programa en un fichero en disco.

4. Antes de pasar a compilar/ejecutar hemos de añadir el fichero que hemos creado al proyecto. Esto se hace situándonos en la ventana donde hemos escrito el programa y pulsando el botón derecho del ratón. Nos aparecerá un menú desplegable como el que vemos a la derecha. En ese menú seleccionamos “Insert File into Project” y marcamos el fichero que se corresponde con nuestro programa.



5. Finalmente ya podemos compilar y ejecutar el programa (de la misma manera que indicábamos antes).

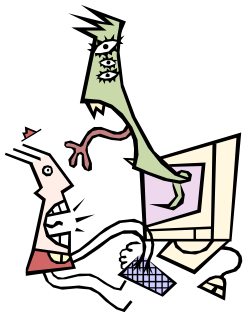
5 Edición de programas ya almacenados

Para editar un programa existente en un fichero de disco tenemos también diversas alternativas:

- El fichero existe como parte de un proyecto. Basta con que carguemos el fichero del proyecto (extensión .mdp).
- El fichero no forma parte de ningún proyecto. En este caso tenemos dos alternativas similares a las que se vieron en la sección de creación de nuevos programas. En cualquiera de los dos esquemas basta con sustituir la fase de creación de un nuevo documento en blanco (New Source File o File->New...->Text file) por la apertura de un fichero desde disco (File->Open).

6 Edición de un nuevo programa

Después de haber estado compilando y ejecutando un programa (o mas bien un proyecto), para poder pasar a editar un nuevo programa o para escribir desde cero un programa distinto al que tenemos en curso, no basta con que cerremos la ventana del programa con el que estamos trabajando **sino que tenemos que cerrar el proyecto completo con el que estamos trabajando (File->Close Workspace) de forma explícita.**



Primeros Pasos con Fortran 90

1 Recuerda

Antes de empezar con los ejercicios que componen esta primera sesión de prácticas, recordemos algunos aspectos de sintaxis que nos van a ser de especial interés.

La estructura general de un programa Fortran responde a los siguientes puntos:

- Nombre del programa (seguido de instrucciones de tipo IMPLICIT).
- Declaración de tipos definidos por el usuario. Declaración y/o inicialización de variables y constantes.
- Sentencias ejecutables.
- Subprogramas.
- Fin del programa.

En esta primera práctica vamos a realizar nuestros primeros programas Fortran que serán muy sencillos, basados exclusivamente en la estructura de control secuencial. En prácticas sucesivas iremos complicando las estructuras que utilicemos al construir nuestros programas, aumentando la potencia de los mismos.

Algunas cuestiones de sintaxis que debes tener en cuenta son las siguientes:

Inicio y fin del programa


```
PROGRAM <nombre_del_programa>
…
END
```

Prohibición de la declaración implícita


```
IMPLICIT NONE
```

Declaración de variables


```
TIPO <lista_atributos>::<lista_variables>
```

Comentarios

```
! Texto del comentario
```

Asignación

```
<variable>=<expresión>
```

Salida de datos por pantalla

```
PRINT*, <lista_expresiones>
```

Entrada de datos por teclado

```
READ*, <lista_variables>
```

2 Toma de contacto: programas en Fortran 90

1) Realiza un programa en Fortran 90 que te salude.

```
1   PROGRAM HolaMundo
2
3   PRINT*, "Hola, bienvenido a Fortran 90"
4
5   END
```

2) Realiza un programa en Fortran 90 que calcule y muestre en pantalla las siguientes expresiones (sin utilizar variables):

$$2^3,\ 2.0^3,\ 2^{3.0},\ 2^{-3},\ 2.0^{-3}$$

Por ejemplo: para calcular la primera expresión (2^3) escribiríamos:

```
1   PROGRAM Elevar2a3
2
3   PRINT*, "2 elevado a 3 vale:", 2**3
4
5   END
```

3) Haz ahora un programa que calcule a^b, para cualesquiera valores *a* y *b* introducidos por el usuario.

3 Corrige los errores

4) A continuación tienes el código de un programa cuya misión es: dada una cantidad arbitraria de horas, minutos y segundos, mostrar dicha cantidad con los segundos y minutos normalizados al rango [0,59]. Por ejemplo, dada la cantidad 20h 78m 145s el programa escribirá en pantalla 21h 20m 25s. Copia el programa (sin los números de línea) en el editor de F-90, y guárdalo con el nombre que te indique el profesor.

Este programa contiene errores sintácticos. Estos errores se deben a que el programa no cumple las reglas sintácticas que define el lenguaje Fortran 90. En gran parte de las ocasiones se dan en instrucciones mal escritas o en expresiones incorrectas (operadores mal usados, número incorrecto de paréntesis, ...). Este tipo de errores se detectan en tiempo de compilación. Compila ahora el programa y mira los errores en la ventana de compilación.

```
1   PROGRAM Conversion_con_errores
2
3   ! Variables
4   INTEGER :: h, m, s
5   INTEGER :: h, mok, sok
6
7   ! Introduccion de datos
8   PRIN*,'Dame las horas, min y seg '
9   READ*, h, m, s
10
11  ! Ajuste de las horas
12  sok = mod(s,60)
13  m = m + s/60
14  mok = mod(m,60)
15  hok = h + m/60
16
17  ! Salida de datos
18  PRINT*,”El tiempo normalizado es:”
19  PRINT*,hok,'h',mok,' m y',sok,' s
20
21  END
```

Veamos la descripción de los errores del programa:

- Se ha declarado dos veces la variable h:
- En la introducción de datos se ha escrito mal la instrucción `PRINT`:

```
p1_4.f90(--):error FOR2934: lexical error: Unrecognized statement

p1_4.f90(--): error FOR3852: syntax error
```

Los errores de este tipo se pueden localizar fácilmente de forma visual ya que el entorno de desarrollo tiene la capacidad de usar colores para identificar los distintos ítems del programa (verde para escribir los comentarios, azul para las palabras reservadas y para aquellas funciones intrínsecas que le son conocidas, …).

- La última instrucción `PRINT` no cierra las comillas de la última cadena de caracteres:

```
p1_4.f90(--):error FOR2934: lexical error: Unterminated character literal

p1_4.f90(--):error FOR3852: syntax error
```

5) Ahora tienes otra versión del mismo programa con distintos errores. En este caso los errores que vas a encontrar son de dos tipos: sintácticos y semánticos. Los sintácticos se detectan siempre en tiempo de compilación. Los semánticos no siempre se pueden detectar durante la compilación. Un error semántico se da cuando el programa es correcto desde el punto de vista sintáctico (es decir, no incumple ninguna regla de sintaxis) pero los resultados no son correctos. En algunos casos el compilador nos avisa de que hay algunas cosas que, aunque son sintácticamente correctas podrían ser

errores potenciales: a este tipo de avisos se les denomina *warnings*. Aunque un programa con warnings se compila correctamente y se puede ejecutar, es conveniente que se repasen todos y cada uno de esos warnings para evitar posibles errores futuros. Modifica el programa anterior para que sea igual al de la figura y guárdalo con el nombre que indique el profesor

```
1   PROGRAM Conversion_con_mas_errores
2
3   ! Variables
4   INTEGER :: h o, m
5   INTEGER :: hok, mok, sok
6
7   ! Introduccion de datos
8   PRINT *,'Dame las horas, min y seg '
9   READ *, ho, m, s
10
11  ! Ajuste de las horas
12  sok = mod(s,60)
13  m = m + s/60
14  mok = mod(m,60)
15  hok = h + m/60
16
17  ! Salida de datos
18  PRINT *,'El tiempo normalizado es:'
19  PRINT *,hok,' h',mok,' m y',sok,' s'
```

Dependiendo del proceso de corrección de errores que sigas podrás encontrarte con algunos de los siguientes problemas:

- En la declaración de variables se declara una variable de tipo entero llamada “h o” (el espacio en blanco no puede formar parte de un identificador).
- En la lectura de valores se lee una variable llamada “ho” que no se ha declarado en ningún sitio.
- Al hacer "hok=h+m/60" no sabemos que resultado puede salir ya que la variable “h” no se ha inicializado en ningún momento. Esto se arregla solucionando el error anterior.
- Usamos una variable h sin haberle asignado previamente ningún valor:

  ```
  p1_05.f90(43): warning FOR4265: symbol H referenced but not set
  ```

  Esto no es un error como tal ya que no estamos infringiendo ninguna regla sintáctica pero es muy posible que sea una fuente de errores en tiempo de ejecución. Los “*warnings*” que nos da el compilador nos indican que hay algo que, aunque la sintaxis es correcta, puede provocar resultados no deseables durante la ejecución del programa y, por tanto, conviene repararlo antes de seguir adelante.
- Al no declarar la variable s se asume la declaración implícita de tipos y entiende que es de tipo real. La función MOD sólo admite argumentos del mismo tipo (normalmente enteros, o reales) y de ahí que tengamos el siguiente error:

  ```
  p1_05.f90(34): error FOR2436: arguments of differing data type/kind in intrinsic MOD
  ```
- Cuando hemos declarado una variable de forma explícita y no hacemos uso de ella durante la ejecución del programa, el compilador nos avisa con un warning de lo que está ocurriendo:

  ```
  p1_05.f90(43): warning FOR4270: unused symbol O
  ```

  Este warning puede deberse principalmente a:

  - Inicialmente habíamos pensado en usar una variable y la hemos puesto en la declaración de variables pero durante el desarrollo del programa no la hemos necesitado y no la hemos quitado de la zona de declaraciones. En este caso no pasa nada, simplemente se quita de la declaración de variables y ya está.
  - En el lugar donde teníamos pensado usar la variable hemos cometido algún error y hemos puesto el nombre de otra variable. Esto si es grave y además es difícil de detectar ya que, puesto que el compilador no sabe que es lo que queremos hacer con cada una de las variables no puede detectar donde hemos cometido el fallo.

Recuerda que Fortran asigna tipo entero a las variables cuyo nombre empieza por i, j, k, l, m, n y tipo real al resto salvo que explícitamente se declaren de otro tipo. En este caso las variables “ho” y “h”, que se usan en el algoritmo, no han sido

declaradas y por tanto se asume que son de tipo real. **Para evitar errores de este tipo lo adecuado es usar siempre la instrucción IMPLICIT NONE en todos nuestros programas.**

4 El problema

6) A lo largo de las prácticas vamos a construir una versión del juego de las siete y media con una baraja española de 40 cartas (del 1 al 7 y las tres figuras, para cada uno de los cuatro palos: oros, espadas, copas y bastos).

Recordemos que en este juego cada carta tiene un valor dependiendo del número que tiene asociado. La forma de saber ese valor es la siguiente: Si el número de la carta está entre uno y siete, el valor de esa carta es el número que tiene asociado; si el número de la carta es diez, once o doce (la carta es una sota, un caballo o un rey), el valor de esa carta es un medio.

El objetivo del juego es conseguir puntos hasta acercarse lo más posible a siete y medio, pero sin pasarse. La puntuación de cada jugador es la suma de los valores de sus cartas.

El juego consiste en que la banca (es así como se denomina al jugador que reparte las cartas, en nuestro caso, el ordenador) reparte una carta a cada jugador. Después de mirar su carta, cada jugador, comenzando por el que está situado inmediatamente a la derecha de la banca, ha de decidir si quiere otra carta o no, dependiendo del valor que tiene su primera carta.

Si un jugador elige continuar pidiendo cartas, recibirá cartas hasta que se plante, es decir, hasta que considere que su puntuación sea suficientemente buena o hasta que sea eliminado por haberse pasado de 7'5 puntos.

Cuando una de esas dos cosas ocurre, la banca pregunta al siguiente jugador si quiere otra carta y se repite la situación. Cada jugador puede mantener una de sus cartas boca abajo. Puede cambiar la carta oculta cada vez que reciba una nueva carta, pero sólo puede tener una carta boca abajo al mismo tiempo. Si tienes todas tus cartas boca arriba, la siguiente carta te será dada boca abajo.

Si por el contrario tienes una de tus cartas boca abajo, la siguiente carta vendrá dada boca arriba.

Un jugador que consigue siete y media tiene que mostrar todas sus cartas cuando reciba la última.

Cuando todos los jugadores que no han sido eliminados han terminado de pedir cartas, tienen que mostrar sus cartas y el que tenga la puntuación más alta gana el juego.

Ahora mismo no disponemos de los conocimientos necesarios para implementar este juego en su totalidad, aunque lo lograremos conforme avancemos en el desarrollo de este cuaderno. Iremos realizando algunas versiones del programa más sencillas que iremos complicando práctica a práctica. Vamos a empezar centrándonos en la generación de números aleatorios con funciones intrínsecas de Fortran 90, que nos va a permitir repartir cartas "al azar".

Consulta en la ayuda del compilador el funcionamiento de las funciones intrínsecas RANDOM_SEED y RANDOM_NUMBER y, con ellas, elabora un programa que genere varias cartas de forma aleatoria y se las indique al usuario (de momento, nuestras cartas son un número del 1 al 40).

Piensa en las limitaciones de esta forma de generar las cartas. Para valorar una jugada hay que sumar los valores asociados a las cartas y ver cómo de cerca se queda de siete y medio. ¿Cómo podrías establecer una correspondencia entre el valor generado aleatoriamente y el valor real de la carta?

5 Ejercicios opcionales

7) Seguramente ya conoces que la energía cinética de un cuerpo en movimiento se calcula a partir de la expresión:

$$E_c = \frac{1}{2} m v^2$$

donde *m* es la masa del cuerpo y *v* es la velocidad con que se mueve.

La energía potencial de un cuerpo sometido a la acción del campo gravitatorio terrestre se calcula igualmente con la ayuda de la siguiente fórmula:

$$E_p = m*g*h$$

donde m es la masa del cuerpo, g es la aceleración de la gravedad y h es la altura a la que se encuentra.

Elabora un programa que solicite la masa y la altura a la que se encuentra un determinado cuerpo suspendido en el aire y que, tras imprimir la expresión y el valor de la energía potencial y cinética, calcule la velocidad con la que impactará en el suelo.

AYUDA: Recuerde que la energía no se crea ni se destruye, sólo se transforma.

8) Complete el modelo del ejercicio anterior para que calcule también el tiempo que tardará en impactar con el suelo. Recuerde que la ecuación que rige espacio recorrido en un movimiento uniformemente acelerado es:

$$s = v_0 t + \frac{1}{2} a t^2$$

Estructuras Condicionales

1 Recuerda

De nuevo, antes de empezar con los ejercicios que componen esta primera sesión de prácticas, recordemos algunos aspectos de sintaxis que nos van a ser de especial interés.

En esta sesión de prácticas vamos a dedicarnos a trabajar con la estructura de control condicional, en sus diferentes variantes. Recuerda que las estructuras condicionales permiten alterar el flujo de ejecución normal de un programa y permiten elegir entre varias alternativas para seguir la ejecución. A continuación refrescamos la sintaxis de las estructuras que permiten hacer esto en Fortran 90.

La estructura condicional IF-THEN

```
IF (<Condición>) THEN
  <BloqueSiCondicionTRUE>
ENDIF
```

La estructura condicional IF lógico

```
IF (<Condición>) <una_instrucción_Si_TRUE>
```

La estructura condicional IF-THEN-ELSE

```
IF (<Condición>) THEN
  <BloqueSiCondicionTRUE>
ELSE
  <BloqueSiCondicionFALSE>
ENDIF
```

La estructura condicional IF-THEN-ELSEIF

```
IF (<Condición1>) THEN
  <BloqueSiCondicion1TRUE>
ELSEIF (<Condición2>)
  <BloqueSiCondicion1TRUE>
…
ELSEIF (<CondiciónN>)
  <BloqueSiCondicionNTRUE>
ELSE
  <BloqueSiNingunaCondicionTRUE>
ENDIF
```

La estructura multicondicional SELECT-CASE


```
SELECT CASE (<expr>)
  CASE (<selector1>)
    <BloqueSiExpr==Selector1>
  CASE (<selector2>)
    <BloqueSiExpr==Selector2>
…
  CASE (<selectorN>)
    <BloqueSiExpr==SelectorN>
  CASE DEFAULT
    <BloqueSiExprDistintoSelectores>
END SELECT
```

2 Toma de contacto: condicionales en Fortran

1) A continuación tienes el listado de un programa que puede calcular la raíz cuadrada, el logaritmo o el cuadrado de un número. Pruébalo y estudia el uso que se hace de las estructuras condicionales. Prueba a introducir valores negativos y positivos seleccionando distintas operaciones sobre dichos valores. ¿Qué ocurre si seleccionas una opción distinta de 1, 2 ó 3? Mejora el programa, de forma que se controle esto último y se utilicen de forma más eficiente las posibilidades de la estructura condicional.

```
1   PROGRAM Condicionales
2
3   REAL :: x,resultado
4   INTEGER :: opcion
5   LOGICAL :: error=.FALSE.
6
7   PRINT*, "Escribe un número real"
8   READ*, x
9
10  PRINT*, "Ahora dime que deseas calcular"
11  PRINT*, "  1 - Su raiz cuadrada"
12  PRINT*, "  2 - Su logaritmo neperiano"
13  PRINT*, "  3 - Su cuadrado"
14  READ*, opcion
15
16  IF (opcion.EQ.1) THEN
17    IF (x<0) THEN
18      error=.TRUE.
19    ELSE
20      resultado = SQRT(x)
21    ENDIF
22  ENDIF
23  IF (opcion.EQ.2) THEN
24    IF (x<=0) THEN
25      error=.TRUE.
26    ELSE
27      resultado = LOG(x)
28    ENDIF
29  ENDIF
30
31  IF (opcion.EQ.3) resultado = x**2
32
33  IF (error.EQV..TRUE.) THEN
34   PRINT*,"No es posible realizar esa operación"
35  ELSE
36   PRINT*,"El resultado es : ",resultado
37  ENDIF
38
39  END
```

3 Corrige los errores

2) El siguiente programa acepta como entrada la nota de un alumno dada en forma de número entero entre 0 y 10 y devuelve un mensaje indicando la nota en forma de texto (suspenso, aprobado, notable, ...). El programa tiene algunos errores. Búscalos y corrígelos.

   ¿Sería posible hacer uso de una estructura condicional sin ELSE IF para resolver este problema? Si la respuesta es afirmativa hazlo.

```
1   PROGRAM PonerNotas
2   ! Variables
3   INTEGER :: la_nota
4
5   ! Captar datos de entrada
6   PRINT*,"¿Que nota has obtenido?"
7   READ*,la_nota
8
9   IF (la_nota<0) .OR. (la_nota>10) THEN
10    PRINT*,"La nota no es válida"
11  ELSE  IF (la_nota<5) THEN
12    PRINT*,"Estas suspenso"
13  ELSE  IF (la_nota.LT.7) THEN
14    PRINT*,"Estas aprobado"
15  ELSE  IF (la_nota<9) THEN
16    PRINT*,"Tienes un notable"
17  ELSE  IF (la_nota=9) THEN
18    PRINT*,"Tienes un sobresaliente"
19  ELSE
20    PRINT*,"Enhorabuena, tienes matricula"
21  ENDIF
22
23  END
```

Ahora, en lugar de aceptar como entrada un número entero entre 0 y 10 nos planteamos aceptar un número real (es decir, que sean válidas notas con decimales de precisión). Modifica el programa original para que funcione de esta forma. ¿Sería posible en este caso hacer uso de la estructura condicional SELECT-CASE? Si se puede, hazlo.

4 El problema

3) Volvamos a nuestro juego de las siete y media. En la sesión anterior tuvimos que tener una "idea feliz" para resolver la asignación entre número de carta y valor. Con el uso de condicionales se simplifica esta tarea.

   Mejora el programa de forma que se establezca una correspondencia entre el número generado y una carta de un palo de la baraja y esto se le comunique de forma apropiada al usuario.

5 Ejercicios opcionales

4) Retome el ejercicio de física de la práctica anterior. Complete el programa de forma que permita al usuario elegir entre las siguientes opciones de un menú:

   - Simular la caída del objeto: el cuerpo está suspendido en el aire y se suelta para que caiga.
   - Simular el lanzamiento hacia arriba del objeto: el cuerpo se encuentra en el suelo y se lanza hacia arriba.

Si el usuario opta por la primera opción se le preguntará la masa del objeto y la altura a la que se encuentra y se le permitirá elegir entre conocer el tiempo que tardará en caer o la velocidad de impacto en el suelo, ofreciéndole la respuesta adecuada en cada caso.

Si el usuario opta por la segunda opción se le preguntará la masa del objeto y la velocidad con que inicia su movimiento de ascensión y se le permitirá elegir entre conocer el tiempo que tardará en subir o la altura que alcanzará el objeto, ofreciéndole la respuesta adecuada en cada caso.

5) Escribe un programa en Fortran 90 que lea tres números enteros por teclado y que devuelva el máximo de los tres. Para ello considera distintas posibilidades:

   - Usa la función intrínseca MAX.
   - Usa condicionales simples (sin parte ELSE).
   - Usa condicionales completas (IF-THEN-ELSE) y anidamiento.

   Para probar el programa utiliza los siguientes casos de ejecución:

   - Introduce los tres números enteros separados por comas.
   - Introduce los tres números enteros separados por ENTER.
   - Introduce algo que no sea un número entero (números reales, letras,...) y observa el resultado. Puedes obtener alguno de los siguientes mensajes:

     ```
     run-time error F6101: READ(CON) - invalid INTEGER

     run-time error F6504: READ(CON) - invalid number in input
     ```

   ¿Qué significan cada uno de los mensajes?

   ¿Qué diferencia hay entre ellos?

6) El servicio de ferrocarriles establece descuentos en el precio de los billetes de acuerdo a los siguientes criterios:

   - Según la edad: Los menores de 4 años no pagan. Los que tienen entre 4 y 7 (ambos inclusive) pagan el 50% del billete. Los mayores de 65 pagan el 40%.
   - Los estudiantes pagan el 45% del billete.
   - Los miembros de familia numerosa pagan el 70%.

   Hacer un programa que tome como datos de entrada: el precio del billete, la edad del cliente, si es o no estudiante y si es miembro de familia numerosa. El programa ha de imprimir el precio a pagar (teniendo en cuenta que los descuentos no son acumulativos). En el caso de que se den varias de las condiciones para un mismo cliente se aplicará sólo aquella que sea más beneficiosa para él.

Estructuras Iterativas

1 Recuerda

La mayoría de los programas que "merece la pena" que realice un ordenador implican tareas que siguen un patrón que se repite. Los bucles nos permiten implementar dichas tareas con facilidad. El objetivo de un bucle es realizar un conjunto de operaciones varias veces de forma consecutiva. Existen distintos tipos de bucle; Fortran 90 incluye de forma natural los dos siguientes:

El bucle DO indexado


```
DO <varcont>=<exprini>,<exprfin>,<exprinc>
  <Bloque>
END DO
```

El bucle DO WHILE


```
DO WHILE(<Condición>)
  <Bloque>
END DO
```

2 Toma de contacto: bucles en Fortran 90

1) Calcular la siguiente función, para un *n* dado:

$$\sum_{k=1}^{n} \frac{3+k^2}{4+7k}$$

```
1   PROGRAM Bucle
2   IMPLICIT NONE
3
4   ! Variables
5   INTEGER :: n, k
6   REAL :: suma
7
8   ! Introduccion de datos
9   PRINT*,'Dame el valor de n:'   ;  READ*,n
10
11  ! Calculos
12  suma = 0
13  DO k=1,n
14    suma=suma+(3.0 + k**2)/(4+7*k)
15  ENDDO
16
17  ! Salida del resultado
18  PRINT *,'El resultado es ',suma
19  END
```

2) La sucesión de Fibonacci tiene la siguiente forma: $s_0=0$, $s_1=1$, $s_n=s_{n-1}+sn_{-2}$.

Elabore un programa que solicite un número al usuario y le diga como respuesta si el número aparece en la sucesión de Fibonacci y, en su caso, en qué posición.

3 Corrige los errores

3) Verifica si el siguiente programa calcula el máximo común divisor y mínimo común múltiplo de dos números. Si es incorrecto repáralo y explica donde está el error.

```
1   PROGRAM MCD_y_MCM
2   IMPLICIT NONE
3
4   ! Variables
5   INTEGER :: m, n, i, tope
6   INTEGER :: mcd, mcm
7
8   ! Introduccion de datos (con filtro)
9   m=0
10  n=0
11
12  DO WHILE (n .le. 0 .or. m .le.0)
13    PRINT*,'Dame dos numeros positivos'
14    READ*,n, m
15  ENDDO
16
17  ! Calculo del MCD
18  mcd = 1
19  tope = min(m,n)
20  DO i=2,tope
21    IF(mod(n,i).eq.0.or.mod(m,i).eq.0) THEN
22      mcd = i
23    ENDIF
24  ENDDO
25
26  ! Calculo del MCM
27  mcm = max (n,m)
28  DO WHILE(mod(mcm,m).ne.0.and.mod(mcm,n).ne.0)
29    mcm = mcm + 1
30  ENDDO
31
32  ! Salida de datos
33  PRINT *,'El MCD de ',n,' y ',m,' es ',mcd
34  PRINT *,'El mcm de ',m,' y ',n,' es ',mcm
35  END
```

4 El problema

4) Con los bucles vamos a poder dar un salto de calidad en nuestro objetivo de implementar el juego de las siete y media. Para empezar ya podemos dotarlo de un funcionamiento cíclico en el que ordenador y jugadores van tomando cartas mientras lo decidan.

El jugador debe poder elegir cuándo quiere parar. Para el ordenador, de momento, se considera una estrategia muy sencilla: dejará de pedir cartas si tiene una puntuación superior a 5. Más adelante intentaremos convertir al ordenador en un jugador "inteligente" de siete y media.

Fíjate también que, gracias a los bucles, ya no es necesario repetir el código que implementa cada turno del juego.

5 Ejercicios opcionales

5) Retoma el ejercicio de física de prácticas anteriores y complétalo de forma que permita al usuario fijar un intervalo de tiempo Δt, y muestre la simulación de la caída del objeto indicando la altura y velocidad que tiene cada Δt segundos hasta que llega al suelo. Haz lo mismo para el caso del lanzamiento hacia arriba del objeto.

6) Vamos a hacer un programa que juegue con nosotros a un clásico. Se trata de ser el más rápido "adivinando" el número que tiene en mente el oponente. Cuando se juega entre dos personas el juego consiste en lo siguiente: cada una de las dos personas piensa un número entre 1 y 100 (sin decírselo al oponente). Alternativamente, cada jugador intenta averiguar el número que pensó el contrario diciendo un número (bien al azar o bien siguiendo alguna estrategia) y el adversario le contesta si el número que tiene pensado en secreto es mayor o menor que el que le han dicho. El primero de los dos jugadores que acierte el número que pensó el contrario gana el juego.

   Veamos un ejemplo:

   Jugador 1: Piensa el 67.
   Jugador 2: Piensa el 22.

   Turno 1:
   Jugador 1: Dice el 43.
   Jugador 2: Contesta que su número es menor.
   Jugador 2: Dice el 20.
   Jugador 1: Contesta que su número es mayor.
   Turno 2:
   Jugador 1: Dice el 19.
   Jugador 2: Contesta que su número es mayor.
   Jugador 2: Dice el 60.
   Jugador 1: Contesta que su número es mayor.
   Turno 3:
   Jugador 1: Dice el 25.
   Jugador 2: Contesta que su número es menor.
   Jugador 2: Dice el 80.
   Jugador 1: Contesta que su número es menor.
   Turno 4:
   Jugador 1: Dice el 22.
   Jugador 2: Contesta que acertó.
   Jugador 2: Dice el 70.
   Jugador 1: Contesta que su número es menor.
   Finaliza el juego ganando el jugador 1.
   I

   mplementa un programa en Fortran 90 que juegue contra un oponente humano a este juego. Piensa en alguna estrategia que permita que el ordenador acierte el número en el menor número de jugadas posible.

   Nota: Para que el ordenador sea capaz de "inventarse" números puedes hacer uso de las subrutinas RANDOM_NUMBER y RANDOM_SEED que ya conocemos.

7) Escribir un programa que determine si un número es primo o no.

Cadenas y matrices

1 Recuerda

Las estructuras de datos que hemos utilizado hasta ahora son muy simples. Con el uso de valores estructurados homogéneos como los vectores y las matrices o, para el manejo de texto, las cadenas de caracteres, aumenta nuestra potencia de gestión de datos.

Declaración de una matriz

```
<TIPO>, DIMENSION(L1,…,Ln) ::<variables>
```

Declaración de una cadena de caracteres

```
CHARACTER (LEN=<valor>) ::<variables>
```

Acceso de lectura

```
<vector>[índice]
<cadena>(índice:índice)
```

Asignación

```
<vector>[índice]=<expresión>
<cadena>(índice:índice)=<expresión>
```

2 Toma de contacto: vectores y matrices

1) Crea un programa que lea por teclado una serie de números, los almacene en un vector y calcule la media, la varianza y la desviación típica.

2) Haz un programa que lea una cadena de caracteres y que escriba la misma cadena eliminando todas las vocales.

3 Corrige los errores

3) A continuación tienes un programa que nos dice si una cadena es o no un palíndromo, es decir, si la cadena se lee igual hacia delante que hacia atrás (sin tener en cuenta signos de puntuación y espacios en blanco). Ejemplos de palíndromos: "Ana", "Dábale arroz a la zorra el abad", "Sara Baras", etc. Corrige los errores que tiene.

```
1   PROGRAM Palindromo
2   IMPLICIT NONE
3   CHARACTER:: H, G  ;  LOGICAL:: Igual  ;  INTEGER:: longitud, pos
4
5   PRINT*, ('Introducir la cadena') ;   READ*, H
6   ! Mostrar la cadena que se ha introducido
7   ! FORTRAN rellena al final con espacios
8   PRINT*,'La cadena tiene longitud:'
9   PRINT*,'>',H,'<'
10  PRINT*,"LEN(h)      = ",len(h)
11  PRINT*,"LEN_TRIM(h) = ",len_trim(h)
12  PRINT*
13  !Eliminamos espacios intermedios
14  pos = index(trim(h),' ')
15
16  DO WHILE (pos.ne.0)
17   H = H(:pos-1) // H(pos+1:)
18   pos = index(trim(h),' ')
19  ENDDO
20
21  PRINT*,'La cadena sin espacios es:'
22  PRINT*,'>',trim(H),'<'
23  PRINT*
24
25  ! Invertimos la cadena H en otra cadena G
26  G = ''       ;            longitud=len(trim(h))
27  DO j=longitud,1,-1
28    G(longitud-j+1:longitud-j+1) = H(j:j)
29  ENDDO
30
31  PRINT*,'La cadena invertida es:'
32  PRINT*,'>',trim(G),'<'
33  PRINT*
34
35  ! Comparamos las dos cadenas
36  IF (H.EQ.G) THEN
37   PRINT *, 'La cadena SI es un palindromo'
38  ELSE
39   Print *, 'La cadena NO es un palindromo'
40  ENDIF
41  END
```

4 El problema

4) Los vectores nos van a permitir arreglar uno de los problemas que tiene nuestro juego: no controlamos que al repartir se repitan cartas. Un vector nos puede servir para almacenar las cartas que ya se han repartido y evitar darlas dos veces.

Usando vectores, retoma el programa del juego y resuelve el problema mencionado. Puedes seguir dos estrategias:

- Almacenar las cartas que ya se han repartido y utilizar un método de búsqueda para comprobar que la carta generada aleatoriamente no se ha repartido ya.
- Implementar la baraja como un vector y "barajarla" antes de repartir. Una vez barajada, repartir cartas es avanzar en el vector. Una forma de barajar puede ser la siguiente: elegir una carta al azar entre 1 y 40, e

intercambiarla con la primera. Luego elegimos al azar alguna de las cartas que están en las posiciones entre 2 y 40 y la intercambiamos con la segunda, y así sucesivamente.

5 Ejercicios opcionales

5) Implementa una versión sencilla del juego del ahorcado. El programa funcionará de la siguiente manera:
   - Dispondrá de un pequeño repertorio de palabras de entre las que elegir.
   - En cada ejecución, elegirá de forma pseudoaleatoria una de esas palabras, sin que la sepa el usuario.
   - Presentará al usuario un carácter "_" por cada letra que tenga la palabra.
   - Pedirá al usuario que le dé una letra:
   - Si la letra ya fue dicha con anterioridad, el programa avisa al usuario y le dice que tiene poca memoria.
   - Si la letra está en la palabra y no ha sido dicha con anterioridad, sustituirá los "_" correspondientes.
   - Si la letra no está en la palabra, avisará del error al usuario.
   - El número de intentos que tiene el usuario para adivinar la palabra se fija al principio de la ejecución del programa.

6) Haz un programa que lea un vector de cadenas de caracteres desde teclado (por ejemplo, nombres de personas) y que lo ordene haciendo uso de alguno de los métodos de ordenación que se han visto en clase.

7) Haz un programa que implemente una versión sencilla del juego de los barquitos.

Subprogramas

1 Recuerda

Sabemos resolver problemas de bastante complejidad con los elementos de programación que hemos trabajado hasta este momento. Ahora se trata de mejorar el aspecto de nuestros programas y facilitar la reutilización del código que hemos implementado para resolverlos.

Los subprogramas nos permiten construir y mantener programas grandes dividiéndolos en módulos independientes más sencillos de implementar. De esta forma podemos generalizar nuestros algoritmos y facilitamos su reutilización. En Fortran hay dos tipos de subprogramas:

Definición de una subrutina

```
SUBROUTINE <nombre> (<parámetros_formales>)
  <Declaración de parámetros formales>
  <Declaración de variables locales>
  …
  <Sentencias ejecutables>
END SUBROUTINE <nombre>
```

Llamada a una subrutina

```
CALL <nombre> (<parámetros_actuales>)
```

Definición de una función

```
FUNCTION <nombre> (<parámetros_formales>)
  <Declaración de parámetros formales>
  <Declaración del tipo devuelto>
  <Declaración de variables locales>
  …
  <Sentencias ejecutables>
  <Asignación del resultado>
END FUNCTION <nombre>
```

Llamada a una función

```
<variable>=<nombre>(<parámetros_actuales>)
```

2 Toma de contacto: subprogramas en Fortran

1) Dado el siguiente programa, construye la subrutina *tricalc* y la función *seguir_o_no*. La subrutina *tricalc* tiene 4 argumentos, el primero es un dato real de entrada y en los otros tres devolverá, en este orden, la raíz cuadrada, el cuadrado y el cubo del dato de entrada. La función *seguir_o_no* mostrará un mensaje en pantalla preguntándole al usuario si desea repetir los cálculos o no; en caso afirmativo devolverá el valor de verdad .TRUE. y en caso negativo devolverá .FALSE. .

```
1   PROGRAM Calculos
2
3   REAL :: x,q,w,b    ;    LOGICAL :: seguir=.true.
4
5   DO WHILE (seguir)
6    PRINT *, "Dime un número"     ;           READ *, x
7
8    CALL tricalc(x,q,w,b)
9    PRINT *, "  Su raiz cuadrada es: ",q
10   PRINT *, "  Su cuadrado es: ",w
11   PRINT *, "  Su cubo es: ",b
12
13   seguir = seguir_o_no()
14  ENDDO
15
16  CONTAINS
17
18   (......)
19
20  END
```

3 Corrige los errores

2) A continuación tienes un subprograma que calcula el factorial de un número. Corrige los errores y haz un programa principal que lea un número por teclado, y que imprima su factorial. Ten en cuenta que el factorial sólo está definido para enteros mayores o iguales que cero. ¿Cuál es el mayor número del que es posible calcular el factorial con este programa? ¿Por qué?

```
1   FUNCTION factorial(x)
2
3   INTEGER :: x     ;    INTEGER :: i
4
5   IF (x<=0) THEN
6    factorial=-1
7   ELSE
8    factorial=0
9    DO i=2,x
10      factorial=i
11   ENDDO
12  ENDIF
13  END FUNCTION factorial
```

4 El problema

3) El paso siguiente es modularizar nuestro programa para que resulte más sencillo su mantenimiento y para que pueda reutilizarse parte del código. Piensa qué módulos pueden resultar de interés. Por ejemplo, puedes hacer un módulo que implemente el barajado de las cartas; otro que calcule el valor de una jugada; … Para ampliar, puedes considerar generalizar los módulos que haces para que permitan implementar de forma sencilla otras versiones del juego con reglas parecidas, por ejemplo el juego de blackjack con una baraja francesa.

En el blackjack la suma del valor numérico de las cartas recibidas deberá alcanzar los 21 puntos o aproximarse lo más posible a esa cantidad, consiguiendo al mismo tiempo una puntuación mayor que la del ordenador. Si el valor de la suma de las cartas sobrepasa los 21 puntos, se habrá perdido la apuesta. En este juego el As tiene un valor de 1 u 11 puntos (el jugador determinará el valor del As, dependiendo del juego que tenga y tomando en cuenta el valor que más le convenga). Las cartas que tienen figuras o caras (es decir, J, Q, K) valen 10 puntos y el resto de las cartas (del 2 al 9)

representan el valor real; es decir, el que tienen impreso. El palo de las cartas (corazones, diamantes, picas y tréboles) tampoco tiene ninguna importancia en este juego.

5 Ejercicios opcionales

4) Elabora un programa modular al que se le dé como entrada un polinomio y permita al usuario trabajar con el siguiente menú de opciones:
   - Evaluar el polinomio en un punto.
   - Mostrar su derivada.
   - Evaluar la derivada en un punto.
5) Haz un subprograma que reciba como entrada un número y devuelva un valor que nos indique si es o no primo. A continuación haz un programa que, haciendo uso del subprograma anterior, devuelva la descomposición en factores primos de un número cualquiera.
6) Revisa el código de programas anteriores y considera posibles modularizaciones de los mismos.

Apéndice A: Depuración

1 MSDSFPS4.0 - Ejemplo de uso del depurador

Con este ejercicio vas a hacer uso de una herramienta que te permitirá detectar los errores de programación con más facilidad: el depurador. El depurador (debugger) nos va a permitir ejecutar instrucción a instrucción nuestro programa. Cada vez que se ejecuta una instrucción tenemos la oportunidad de analizar los valores que tienen las variables de nuestro programa e incluso modificar sus valores desde el depurador.

Esta herramienta permite también ejecutar bloques de varias instrucciones. Supongamos que tenemos un programa con un error y que sabemos que el error se encuentra aproximadamente en la segunda mitad del programa.

Podemos indicarle al depurador que ejecute normalmente la primera parte del programa y que se detenga cuando llegue al trozo donde creemos que está el error para ejecutar instrucción a instrucción a partir de ese punto. Para ello usaremos los *breakpoints* (puntos de ruptura).

A continuación tienes una tabla con los comandos más utilizados durante la depuración:

Tecla	Comando	Descripción
Alt + F5	Stop debugging	Detener el proceso de depuración por completo.
Mayús. + F5	Restart	Finalizar el proceso de depuración en curso y comenzar desde el principio.
F5	Go	Comenzar/continuar el proceso de depuración (hasta encontrar algún punto de ruptura o hasta llegar al final del programa).
F7	Run to cursor	Comenzar/continuar el proceso de depuración hasta llegar a la instrucción en la que tenemos situado el cursor y detenerse justo antes de ejecutarla.
Mayús. + F7	Step out	Si estamos en mitad del proceso de depuración y estamos ejecutando instrucciones dentro de un subprograma, al ejecutar este comando se completa la ejecución del subprograma y el depurador se detiene justo después de terminar el subprograma.
F8	Step into	Ejecuta la instrucción siguiente. Si esta instrucción es un subprograma, la siguiente instrucción a ejecutar será la primera del subprograma. Es decir: continuamos la depuración dentro del subprograma. Si aún no habíamos empezado a depurar entonces este comando comienza la depuración y nos sitúa en la primera instrucción ejecutable de nuestro programa.
F10	Step over	Ejecuta la instrucción siguiente. Si esta instrucción es un subprograma entonces, en lugar de pasar a la depuración del subprograma, lo ejecuta por completo (como si fuese una única instrucción) y pasamos a la instrucción que haya a continuación de la llamada al subprograma. Es decir, se ejecuta el subprograma completo (salvo que haya puntos de ruptura en su interior) como si fuese una única instrucción.
F9	Insert/Remove breakpoint	Pone un punto de ruptura en la línea del programa donde tenemos situado el cursor. Si ya existía un punto de ruptura en esa línea entonces lo elimina.

Considera el siguiente programa:

```
1   PROGRAM Depurame
2    REAL :: x          ;       INTEGER :: y, z=5
3    PRINT *, "Dime un número"    ;    READ *, x
4    y = 1
5    IF (x<0) THEN
6       PRINT *, "El número es negativo"
7       y = 2
8    ELSE IF (x<10) THEN
9       PRINT*, "El número está entre 0 y 10"
10      y = 3
11   ELSE
12      PRINT*, "Numero mayor o igual que 10"
13      y = 4
14   ENDIF
15   z = y + 2          ;     PRINT *, "z vale ", z
16  END
```

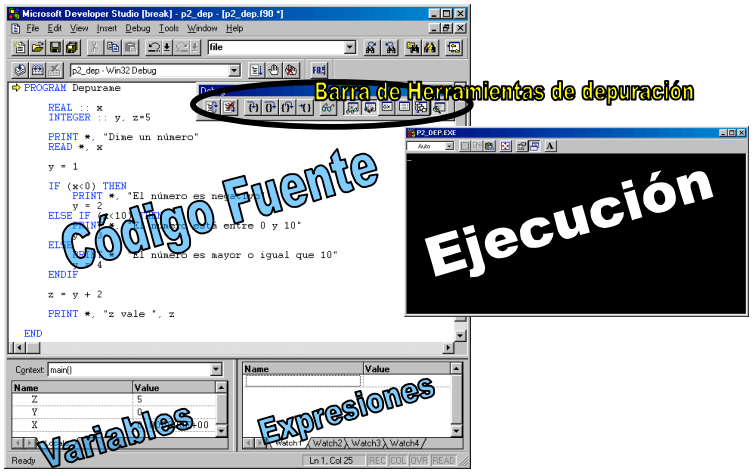
Vamos a realizar una depuración del mismo para entender el funcionamiento del depurador.

Comenzaremos la depuración pulsando F8. Tras haber compilado y enlazado el programa nos aparecerá una ventana de tipo DOS donde tendremos el resultado de la ejecución de nuestro programa como de costumbre aunque, aparentemente, no

veremos (aún) ningún resultado. Inmediatamente después esta ventana será ocultada por el compilador en el que podremos distinguir 3 zonas (ver figura):

- El código fuente en la mitad superior.
- Visualización de variables en la mitad inferior izquierda.
- Visualización de expresiones en la mitad inferior derecha. Esta última ventana no nos interesa por ahora.

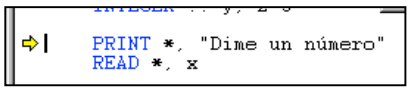
En la ventana del código fuente podemos ver una flecha amarilla en la primera línea. Esta flecha nos indica, en todo momento, cual es la siguiente instrucción que hay que ejecutar. En este caso, aunque la instrucción `PROGRAM` no es ejecutable, nos indica, de forma simbólica, que estamos a punto de comenzar la ejecución (y depuración) de nuestro programa.



En la ventana de variables podemos ver una tabla en la que tenemos todas las variables que hemos declarado en nuestro programa junto con su valor actual. Podemos observar que al comienzo de la ejecución las variables se inicializan a cero salvo que se inicialicen de forma explícita (como es el caso de la variable `z` que inicializamos a 5).

Además de estas zonas también nos ha aparecido una barra de herramientas con el título “Debug”. Estos botones nos ofrecen un acceso rápido a algunos comandos del depurador.

- Comenzamos la depuración pulsando de nuevo F8. La flecha amarilla se sitúa en la línea donde aparece la primera instrucción ejecutable. Esto nos indica que esa será la siguiente instrucción a ejecutar. Podemos ver la ventana de ejecución y observaremos que aún permanece vacía.

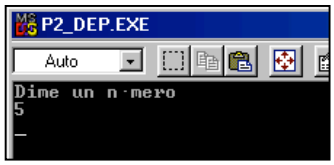


- Pulsamos de nuevo F8 y ejecutamos la instrucción `PRINT`.

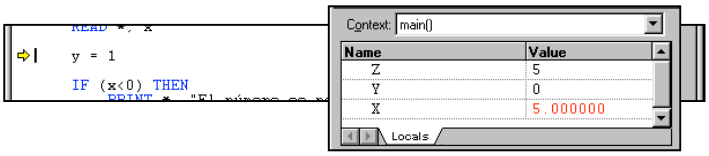
Ahora podemos ver la ventana de ejecución y observar el resultado de haber ejecutado el `PRINT`. Vemos escrito el mensaje “Dime un número”.

Hasta el momento la ventana de Variables permanece intacta.

- Si pulsamos de nuevo F8 y ejecutamos la instrucción `READ` en apariencia no ocurre nada y parece que el depurador no responde ya que la flecha amarilla no avanza. Esto no es ningún error. La ejecución de la instrucción `READ` implica realizar una entrada de datos por teclado y eso es justo lo que está esperando el depurador. Si vamos a la ventana de ejecución vemos que el cursor parpadeante ha avanzado a la línea siguiente al mensaje del `PRINT`. Está esperando a que introduzcamos algún dato que será procesado por la instrucción `READ`. Podemos introducir, por ejemplo, un número 5. Tras hacer esto el depurador toma el control de nuevo y hace avanzar la flecha amarilla.

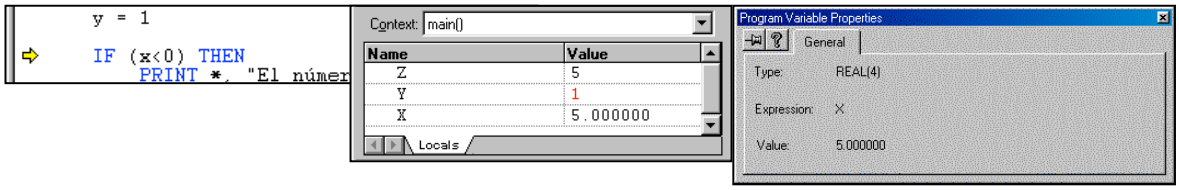


Pero no sólo eso sino que también podemos ver, en la ventana de variables, como se ha cambiado el valor de la variable `x`. En esta ventana aparecerá en color rojo la última variable que haya cambiado su valor.



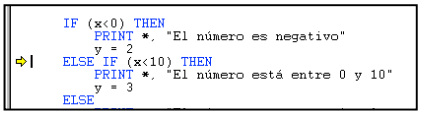
- Desde la ventana de variables también podemos consultar el tipo del que ha sido declarada una variable. Para eso situamos el puntero del ratón sobre la variable en cuestión y pulsamos el botón derecho. Nos aparece un menú desplegable y seleccionamos "properties". Obtendremos algo similar a la ventana siguiente.

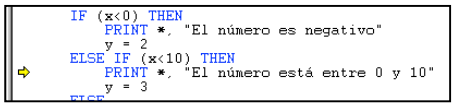
- Si seguimos la depuración (F8), ejecutaremos la instrucción de asignación `y=1` y vemos el efecto en la ventana de variables. El valor de la variable `Y` aparece en rojo.



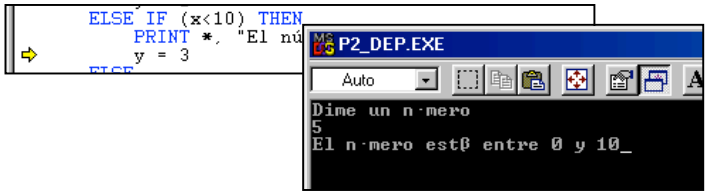
- La siguiente instrucción que vamos a ejecutar es una condicional. Tras pulsar F8 la flecha amarilla se sitúa en la línea `ELSEIF`. Lo que ha ocurrido es que se ha evaluado la condición (`x<0`) y se ha llegado a la conclusión de que es falsa (en la ventana de variables vemos que `x` vale 5) y por tanto se ha pasado a evaluar la siguiente condición de la estructura `IF-THEN-ELSEIF`.

- Puesto que la condición de la línea `ELSEIF` es cierta (`x<10`) tras ejecutarla marcamos como siguiente línea ejecutable el `PRINT` que hay dentro de ese bloque. Hasta este punto no habíamos producido ningún nuevo cambio sobre la ventana de ejecución ya que desde el último `READ` no ha habido más instrucciones de E/S.

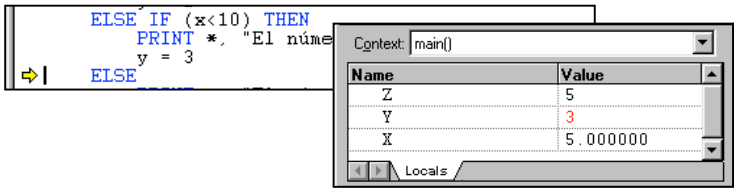




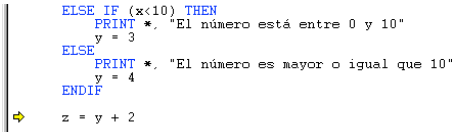
- Tras ejecutar la instrucción `PRINT` vemos los cambios en la ventana de ejecución.



- Después de ejecutar la asignación nos queda la flecha apuntando a `ELSE`. Indicando que se ha acabado la ejecución condicional. El resultado que produce la ejecución del `ELSE` es devolver el control del programa a la siguiente instrucción tras la estructura condicional.

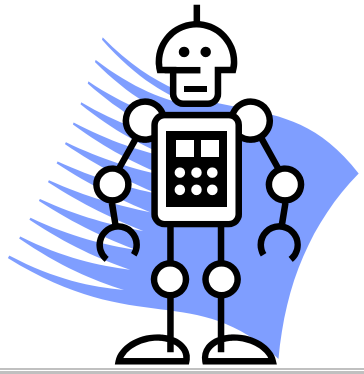


- Si ejecutamos ahora vemos que el control salta a la asignación `z=y+2`.



Una vez ejecutada la asignación vemos que z toma el valor 5 en la ventana de variables. En este caso concreto z ya valía 5 antes de asignarle su nuevo valor.

- Finalmente, después del último PRINT, nos quedará esta ventana de ejecución.



Apéndice B: Mejorando el juego

1 Una estrategia para ganar

Buena parte de nuestro esfuerzo a lo largo de este cuaderno de prácticas ha estado orientado a construir una versión del juego de las siete y media. Sin embargo, pese a este esfuerzo, el ordenador es un oponente previsible, con una técnica de juego bastante sencilla (por encima del 5 no arriesga). Vamos a dedicar este apartado a presentar algunas ideas que nos pueden permitir implementar un comportamiento más "inteligente" por parte de la computadora.

Lo que tenemos que discutir es una estrategia o regla de decisión que permita determinar la forma de actuar del ordenador en función de las cartas que tenemos (las nuestras y las que podemos ver de los demás).

Una regla de decisión podría ser tan simple como la que ya hemos implementado (seguir mientras no superemos un cinco) o puede ser más sofisticada teniendo en cuenta, por ejemplo, la teoría de la probabilidad (seguir mientras la probabilidad de ganar sea superior a 0.7).

Una conocida regla de decisión para este juego basada precisamente en el uso de probabilidades es la siguiente:

Sea p_1 la probabilidad de ganar si no se pide otra carta y p_2 la probabilidad de pasarse de siete y media si se pide otra carta. Se define la siguiente función $f(p_1,p_2)$ por casos:

$$f(p_1,p_2)=\begin{cases} Parar & si\ \{p_1 \geq A\} ó\ \{p_1 \in [C,A)\ y\ p_2 \geq B\} \\ Continuar & en\ caso\ contrario \end{cases}$$

donde A, B y C son valores entre 0 y 1.

Esta regla de decisión quiere decir básicamente que pararemos de pedir cartas si nuestra probabilidad de ganar es suficientemente alta ($p_1 \geq A$) o si la probabilidad de pasarse es demasiado alta ($p_2 \geq B$) y la probabilidad de ganar está en un intervalo controlado ($p_1 \in [C, A)$. El riesgo que toma la máquina depende de los valores A, B y C elegidos.

Si queremos jugar a las "Siete y media", utilizando la regla de decisión anterior tenemos que calcular algunas probabilidades. Estas probabilidades son p_1 y p_2 y las calculamos usando las herramientas básicas del cálculo de probabilidades y la información que tenemos en cada momento del juego:

- Las cartas que nuestros oponentes tienen boca arriba.
- Las cartas que nuestros oponentes tienen boca abajo.

1) Como ejercicio final de estas prácticas, puedes completar la implementación del juego de las siete y media incorporando este tipo de estrategias para adoptar la decisión que toma el ordenador a la hora de pedir carta o no pedirla. ¡Refresca tus conocimientos de estadística y atrévete con el reto!

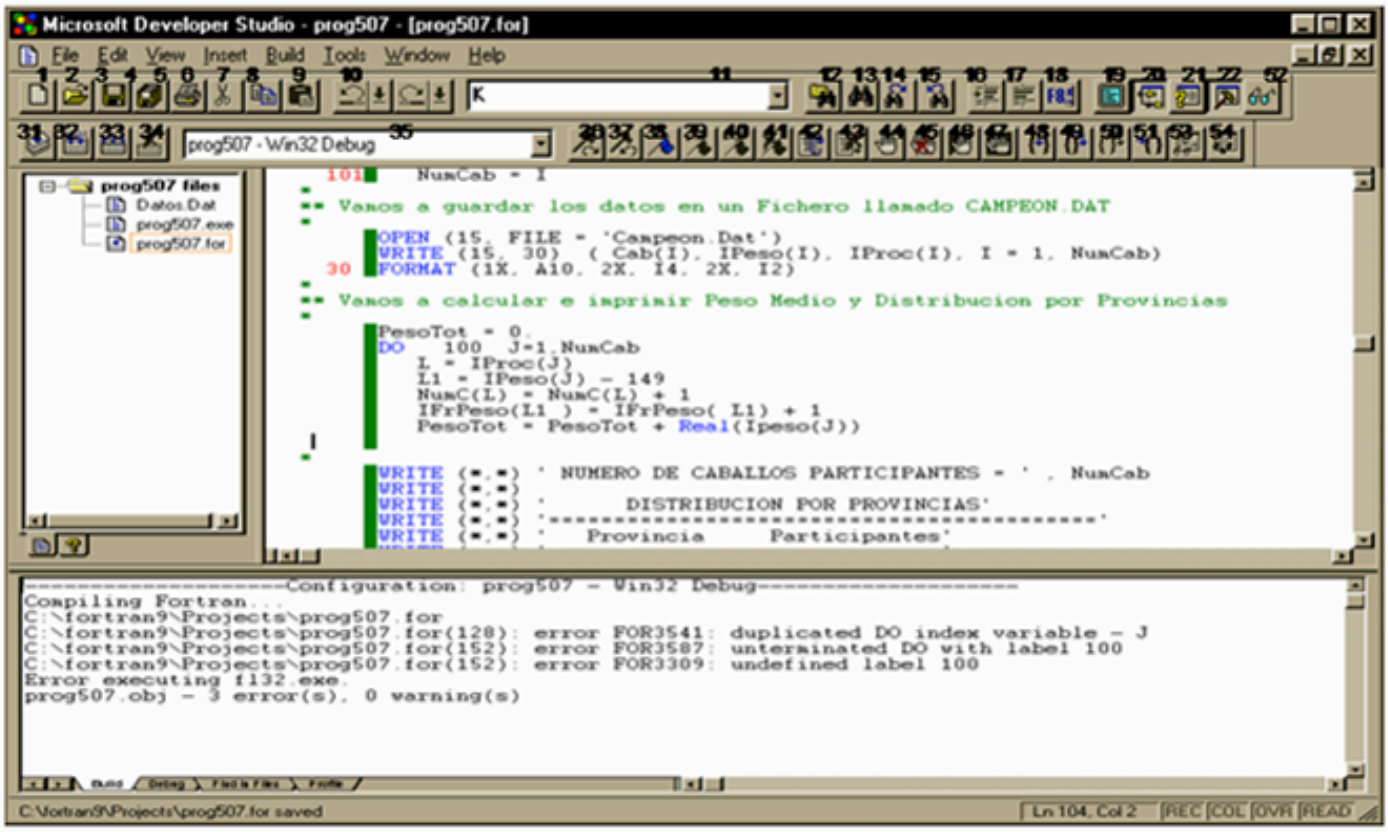
Apéndice C: Material adicional

1 Decálogo de indicaciones rápidas para MDS

1	F90PS Distingue entre Formato Fijo (Casi F-77) y Formato Libre (F-90). Formato Libre no tiene en cuenta lo de las columnas 7-72, etc, pero además, no acepta 'C' ni '*' como símbolo de línea de comentarios (solo acepta '!' , pero se puede poner en cualquier columna). Un fichero no se puede compilar en las dos formas, Para diferenciarlo, los de formato fijo han de llevar la extensión *.f o *.for , mientras que los de formato libre llevan por fuerza, *.f90
2	Para entrar en el entorno integrado, se carga MS Developer Studio , y se entra directamente a trabajar. Hay que manejar los conceptos de Area de Proyectos (WorkSpace), Area de Compilación (OutputSpace) y Area Edición (FileSpace). Son 3 zonas de la ventana de Trabajo, que se pueden usar en modo normal, y para ver la ayuda.
3	Los Programas se pueden crear como ficheros en cualquier editor, incluido este Area Edición. Para compilarlos, es necesario incluirlos en un proyecto (se debe poner el mismo nombre al fichero *.f90 y al proyecto, que le da nombre al ejecutable (*.exe). Los proyectos disponibles en el directorio actual, se pueden ver en el Area Proyectos, y el que esté en negrita, es el activo por defecto. El Area-Proyectos se guardan en ficheros *.mdp Para abrir un Area Proyecto, Fichero→ Open WorkSpace (Si el Area esta creada, se abren los proyectos contenidos en la misma). Para crear un proyecto nuevo, se hace en Insert → Project , y se incorpora al Area-Proyectos actual. Para abrir ficheros de programa, File→Open, y para crearlos nuevos, File→New. Para añadir ficheros (por ejemplo *.f90 o de datos) a un proyecto, se hace con Insert→File into Project .
4	Para compilar y ejecutar, usar los apartados del Menú Build. Si al compilar hay errores, estos se listan en el Area de Compilación. Habrá que corregirlos en el Area Edición, y volver a compilar. Para errores difíciles de corregir, se puede usar la opción Build→Debug , que permite poner marcas de parada en la ejecución y rastrear el programa paso a paso. Si se intenta compilar sin haber creado un Area-Proyecto, nos dice (en inglés) que debemos crearla, y hay que decirle que SI. Para crear el ejecutable hay que usar la opción Build . Cuando se ejecuta el programa, se abre una ventana del DOS para la ejecución, y cuando se termina, se cierra esa ventana.
5	En Area-Proyectos, pinchando en signo + previo a un proyecto, se puede ver los ficheros contenidos en el mismo, y seleccionar el que queramos para pasarlo al Area-Edición. Podemos tener varios ficheros en el Area-Edición, y cambiar de un fichero a otro, con el menú Windows .
6	Poniendo el cursor en una palabra (p.e., en un tipo de error en Area-Compilación), y pulsando F1, se obtiene una información adicional sobre la palabra (sobre el tipo de error existente, y como corregirlo).
7	Es conveniente dar un "paseo" por las ayudas, y leerse alguna vez el contenido de: Help → Technical Support y Help → Tip of the Day. Cuando se tengan problemas, buscar en Help → Search o en Help → Contents
8	Los menús File, Edit, Insert, Build (y Debug) son los que usaremos para crear, compilar y ejecutar programas en Fortran-90. Los menús de Edición (File, Edit , Insert) funcionan de forma parecida a otros programas como el Word, Power Point, etc.. En las fotocopias que siguen, hay una descripción más detallada de la forma de hacer la compilación y la depuración (cuando haga falta) (Menús Build, Debug).
9	Los Menús, View y Windows nos permiten adecuar la presentación de las ventanas a nuestro gusto. Con View → ToolsBar , podemos poner en la ventana aquellos botones que más se utilizan (en fotocopias siguientes se incluyen estos botones). El menú Help nos permite ver el manual del Entorno, y un manual de Fortran 90, para resolver problemas concretos.
10	Dar un repaso a las fotocopias donde se indican los botones clásicos que se usarán en este curso, y los apartados de los menús que también se van a usar de forma habitual.

2 F90PS – Entorno Integrado

1, 2, 3	CREAR, ABRIR y GUARDAR FICHERO
4	GUARDAR TODOS FICHEROS
5	IMPRIMIR FICHERO
6,7, 8	CORTAR, COPIAR y PEGAR SELECCIÓN
9, 10	DESHACER, REHACER ULTIMA ACCION
11	VISUALIZADOR CADENA BUSQUEDA
12, 13	BUSCAR en UN0 / VARIOS FICHEROS
14, 15	BUSCAR SIGUIENTE /ANTERIOR OCUR
16,17	PONER/QUITAR INDENTACION PARRA.
18	EDITOR FORMATOS
19	PANTALLA COMPLETA
20,21,22	AREAS PROYECTO, AYUDA, COMPILACION
31,32,33,34	COMPILA, LINK, LINK TODO, PARAR LINK
35	VISUALIZA PROYECTO POR DEFECTO
36, 37	IR AL PROXIMO / ANTERIOR ERROR
38,39,40,41	MANEJAR MARCAS en el TEXTO
42,43	REINICIA/PARA RASTREO ERRORES (DEBUG)
44,45,46,47	MANEJO PUNTOS PARADA en RASTREO ERR.
48,49, 50, 51	MANEJO DE PASOS en RASTREO ERR.
52	CALCULO RAPIDO EXPRESIONES
53, 54	VENTANAS EJECUCION y de VARIABLES



3 Menús del entorno desplegados