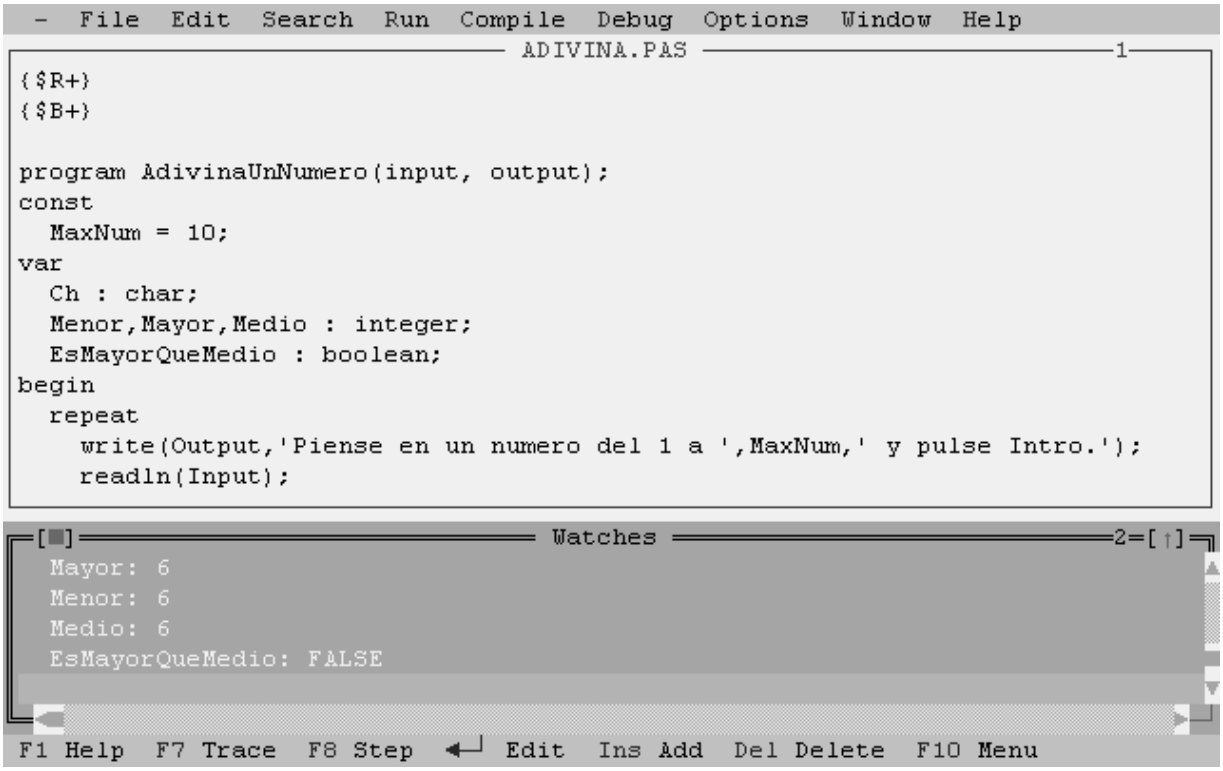
Programación I - Primera Sesión de Prácticas

Introducción a Turbo-Pascal

1. El Entorno de Programación

El entorno de programación de Turbo-Pascal consiste en un editor de programas desde el cual es posible escribir, compilar, ejecutar y depurar programas escritos en Pascal.

Para ejecutar Turbo-Pascal se debe hacer doble-click sobre su icono situado en el escritorio de windows. La apariencia del entorno es la siguiente:



Para alternar entre el modo de pantalla completa y el modo de ventana pulse [Ctrl-Intro]. Se puede apreciar que el entorno consta de las siguientes partes:

- **Menú general**: Mediante el menú es posible acceder a todos las comandos de Turbo-Pascal: Abrir y guardar programas, compilarlos, ejecutarlos, buscar texto, etc. Para acceder al menú se puede usar tanto el ratón como el teclado: Pulsando [F10] se accede a (y se sale de) la barra de menú, desde la cual podemos desplazarnos y desplegar submenús utilizando las teclas del cursor.

- **Barra de estado**: Muestra combinaciones de teclas que permiten acceder de manera rápida a los comandos más usuales.
- **Ventanas**: Sirven para editar programas. Podemos tener abiertas hasta 9 ventanas distintas. Cada una de ellas recibe un número (se puede apreciar en la esquina superior derecha) y pulsando [Alt] junto con el número de la ventana podemos hacer que pase a primer plano. También se puede cambiar su posición y tamaño usando los comandos del menú **Window**.
- **Diálogos**: Sirven para introducir información (el nombre de un fichero, el texto de búsqueda, etc.). Para pasar de un elemento del diálogo a otro pulse el tabulador. Para marcar o desplegar listas use la tecla de espacio, y [Esc] o Intro para salir.

Turbo-Pascal dispone de una ayuda tanto sobre el entorno como sobre la versión del lenguaje Pascal que se soporta. La ayuda se puede acceder de las siguientes formas:

- Pulsando [F1] se muestra ayuda referente a la parte del entorno donde nos encontremos: La ventana de edición, los comandos de menú, etc. También sirve para obtener información ampliada sobre los errores de sintaxis de nuestros programas.
- Pulsando [Shift-F1] accedemos al índice general de la ayuda.
- Si estamos en la ventana de edición y el cursor se encuentra situado sobre una palabra reservada de Pascal, pulsando [Ctrl-F1] obtenemos ayuda sobre el lenguaje Pascal.
- Por último, pulsando sobre **Contents** en el menú **Help** obtenemos la ventana principal de ayuda.

Nota: La única referencia válida sobre el lenguaje Pascal para la asignatura Programación I es la información proporcionada en las clases teóricas. No debe utilizarse ningún comando, estructura de datos o control o subprogramas de librerías que no sean aquellos contemplados explícitamente en la asignatura, independientemente de que aparezcan en la ayuda de Turbo-Pascal o no. En caso de duda consulte con su profesor.

2. Creación y Compilación de Programas

En este apartado vamos a seguir el proceso de escritura y corrección de un programa en Pascal. El programa simulará el juego de adivinar un número desconocido siguiendo la estrategia de acotar su valor mediante una serie de preguntas cuya respuesta es si o no.

El programa consta de un bucle principal que repite la acción de adivinar un número hasta que el usuario indique que desea terminar el programa. Para adivinar un número se usan tres variables: *menor* y *mayor* nos indican el rango de valores en el que sabemos, en un momento dado, que se encuentra el número a adivinar. Conoceremos el número cuando ambas contengan el mismo valor. Para ir acotando, calcularemos el valor *medio* de los dos y preguntaremos al usuario si el número es mayor que el valor *medio*: Si la respuesta es que sí, podemos descartar el intervalo desde *menor* hasta *medio*, y asignamos a *menor* el valor de *medio*. En caso contrario, se descarta el intervalo desde *medio*+1 hasta *mayor*, y asignamos a *mayor* el valor de *medio*.

A continuación se muestra un programa que implementa el algoritmo anterior. Intencionadamente se han incluido errores tanto de sintaxis como de diseño, para mostrar algunos de los errores más comunes y posibles técnicas para detectarlos y corregirlos. No es, por lo tanto, un programa que pueda servir de ejemplo de programación, sino más bien lo contrario.

Transcriba textualmente el siguiente programa:

```
 1  {$R+}
 2  {$B+}
 3
 4  program AdivinaUnNumero(input, output);
 5  const
 6    MaxNum = 10;
 7  var
 8    Ch : char;
 9    Menor,Mayor,Medio : integer;
10    EsMayorQueMedio : boolean;
11  begin
12    repeat
13      write(Output,'Piense en un numero del 1 a ',MaxNum,' y pulse Intro.');
14      readln(Input);
15      Menor := 1;
16      Mayor := MaxNum;
17      while Menor <> Mayor do
18      begin
19        Medio := (Menor+Mayor)/2;
20        write(output,'¿Es mayor que ',Medio,'? [S/N]');
21        readln(input,Ch);
22        EsMayorQueMedio := (Ch = 'S') or (Ch = 's');
23        if EsMayorQueMedio then
24          Menor := Medio;
25        else
26          Mayor := Medio;
27
28      writeln(output,'Ya esta! Pensaste en el ',Menor);
29      write(output,'¿Otra vez? [S/N]');
30      readln(input,Ch)
31    until (Ch = 'n') and (Ch = 'N')
32  end
```

Nota: Los números de la parte izquierda no forman parte del programa (no deben escribirse en el editor), su función es numerar las líneas del programa para poder hacer referencia a ellas posteriormente. Las líneas 1 y 2 no son comentarios sino directivas de compilación, y es obligatorio que aparezcan siempre como primeras líneas de cualquier programa que escribamos.

Existen tres formas de compilar un programa. Pulsando [Alt-F9] compilamos únicamente el programa que aparece en la ventana actual. Pulsando [F9] compilamos tanto el programa como las unidades que éste utiliza, y pulsando [Ctrl-F9] se compila el programa sólo si ha habido cambios desde la última vez y tras la compilación se ejecuta.

En esta primera sesión no vamos a generar ficheros ejecutables, por lo que elegimos compilar a memoria: Pulsamos [Alt-C] para desplegar el menú **Compile** y si la opción **Destination** tiene el valor **Disk** pulsamos [D] para cambiarla; Si ya tuviera el valor **Memory** salimos del menú pulsando [Esc] o [F10].

Pulsamos [Alt-F9] para compilar el programa y obtenemos el error **type mismatch**, situándose el cursor en la línea 19, donde se ha detectado. Si pulsamos [F1] obtenemos información ampliada sobre las posibles causas del error. En este caso lo que sucede es que intentamos asignar un valor de tipo real (la media de *menor* y *mayor*) a una variable de tipo entero. Debemos obtener un valor entero, y para ello sustituimos el operador de división real (/) por el de cociente (div).

Si compilamos otra vez, obtenemos **error in statement** en la línea 25. Esto sucede por el punto y coma situado tras la asignación "menor := medio". En Pascal, a diferencia de otros lenguajes (por ejemplo, C), el punto y coma no se usa para finalizar sentencias, sino para separar una sentencia de otra. Por eso, aquí Pascal entiende que se termina la sentencia condicional, y no entiende la aparición

de **else**, que no puede comenzar ninguna sentencia válida. Eliminamos el punto y coma y volvemos a compilar.

Ahora obtenemos el error "**; expected**"en la línea 31. Aparentemente, el error es no haber finalizado con punto y coma la sentencia anterior, lo que parece estar en contradicción con el párrafo anterior, pues no hay que separar esa sentencia de ninguna otra. Sin embargo, si ponemos un punto y coma al final de la línea 30 y volvemos a compilar, obtenemos **error in statement** en la línea 31, lo que demuestra que la existencia o no del punto y coma al final de la línea 30 no es la causa del error sintáctico.

El error de la línea 31 se produce porque no es correcto en ese punto que intentemos cerrar la estructura **repeat**. ¿Por qué? Si nos fijamos en el programa, veremos que nos falta el **end** de cierre de la sentencia compuesta que iniciamos en la línea 18. Si ahora escribimos "**end;**" en la línea 27, y eliminamos el innecesario punto y coma que añadimos en la línea 30, resolvemos ese problema.

Lo anterior en un claro ejemplo de que no siempre la línea donde se detecta un error es la misma que en la que está el error.

Por ultimo, al compilar nos aparece ahora el error **unexpected end of file**, situándose el cursor en la línea siguiente al end final del programa. Lo que sucede en este caso es que nos hemos olvidado de poner el punto final del programa. Si añadimos un punto al final de la línea 32 y volvemos a compilar no obtendremos ningún error. El programa quedaría así:

```
 1  {$R+}
 2  {$B+}
 3
 4  program AdivinaUnNumero(input, output);
 5  const
 6    MaxNum = 10;
 7  var
 8    Ch : char;
 9    Menor,Mayor,Medio : integer;
10    EsMayorQueMedio : boolean;
11  begin
12    repeat
13      write(Output,'Piense en un numero del 1 a ',MaxNum,' y pulse Intro.');
14      readln(Input);
15      Menor := 1;
16      Mayor := MaxNum;
17      while Menor <> Mayor do
18      begin
19        Medio := (Menor+Mayor) div 2;
20        write(output,'¿Es mayor que ',Medio,'? [S/N]');
21        readln(input,Ch);
22        EsMayorQueMedio := (Ch = 'S') or (Ch = 's');
23        if EsMayorQueMedio then
24          Menor := Medio
25        else
26          Mayor := Medio;
27      end;
28      writeln(output,'Ya esta! Pensaste en el ',Menor);
29      write(output,'¿Otra vez? [S/N]');
30      readln(input,Ch)
31    until (Ch = 'n') and (Ch = 'N')
32  end.
```

3. Depuración de programas

El hecho de que en la compilación de un programa no se detecte ningún error tan sólo significa que es sintácticamente correcto, pero de ningún modo que no existan errores ni, por supuesto, que realice correctamente la tarea para la que ha sido diseñado. Veremos que todavía existen errores en el programa anterior y examinaremos algunas de las herramientas de que dispone Turbo-Pascal para detectarlos.

Si disponemos de un disquete pulsamos la tecla [F2] para salvar el programa, y escribimos "**a:\adivina.pas**" (no olvidar introducir antes el disquete en la disquetera). Pulsamos retorno, volviendo al editor del programa. Es conveniente siempre salvar el trabajo realizado antes de ejecutar un programa, ya que pueden existir errores tan graves que impidan regresar de manera normal al entorno.

Ejecutamos el programa, pulsando [Ctrl-F9] y respondemos a las preguntas suponiendo que hemos pensado en el valor 1. Aparentemente, el programa funciona correctamente:

```
Piense en un numero del 1 a 10 y pulse Intro.
¿Es mayor que 5? [S/N] n
¿Es mayor que 3? [S/N] n
¿Es mayor que 2? [S/N] n
¿Es mayor que 1? [S/N] n
Ya esta! Pensaste en el 1
¿Otra vez? [S/N] _
...
```

Pero si escribimos "**s**" y volvemos a repetir el proceso pensando ahora en el 2, obtenemos lo siguiente:

```
¿Otra vez? [S/N] s
Piense en un numero del 1 a 10 y pulse Intro.
¿Es mayor que 5? [S/N] n
¿Es mayor que 3? [S/N] n
¿Es mayor que 2? [S/N] n
¿Es mayor que 1? [S/N] s
¿Es mayor que 1? [S/N] s
¿Es mayor que 1? [S/N] s
...
```

Podemos detener el programa pulsando [Ctrl-C]. Observamos que en este caso el programa no detecta que el número pensado es el 2. Para resolver este tipo de problemas es necesario examinar el programa y comprobar si hemos cometido algún error al traducir el algoritmo a Pascal o incluso si el algoritmo que diseñamos es erróneo.

Una técnica que suele ser útil en estos casos es obtener una **traza** del programa. Turbo-Pascal permite que ejecutemos paso a paso las sentencias de nuestro programa, viendo en cada momento los valores de las variables que seleccionemos. Vamos a usar esa técnica para resolver el problema con que nos hemos encontrado:

Pulsamos [Alt-W] para desplegar el menú **Window**, y pulsamos el comando **Watch**. Vemos que nos aparece una ventana titulada **Watches**, que es donde se van a mostrar los valores de las variables. Es conveniente que cambiemos el tamaño de la ventana donde se muestra el programa para que no oculte a la ventana **Watches**:

- Con el ratón: Pulsamos en la ventana del programa, la cual pasa a primer plano ocultando la ventana "**Watches**". Movemos el ratón a la esquina inferior derecha de la ventana, pulsando y moviendo el ratón (sin dejar de pulsarlo) vemos que la ventana cambia de tamaño. Cuando haya alcanzado el tamaño deseado dejamos de pulsar el ratón.

- Con el teclado: Pulsamos [Alt-1] para que la ventana del programa pase a primer plano. Pulsamos [Ctrl-F5] para indicar que queremos mover o cambiar las dimensiones de la ventana actual (el borde de la ventana adquiere un color verde) y a partir de ese momento las teclas del cursor [←] [↑] [→] [↓] sirven para desplazar la ventana, y manteniendo pulsada la tecla *shift* ([⇑]) estas mismas teclas sirven para cambiar su tamaño. Cuando hayamos terminado pulsamos [Esc].

Para incluir una variable en la ventana "**Watches**" pulsamos [Ctrl-F7], escribimos su nombre y pulsamos Intro. En este caso añadimos las variables *Menor*, *Mayor*, *Medio* y *EsMayorQueMedio*.

Para iniciar la ejecución paso a paso, pulsamos [F7]. Vemos que cada pulsación provoca la ejecución de una sentencia, que aparece remarcada. Cuando se ejecuta una sentencia **readln** aparece la pantalla de salida, permitiéndonos introducir el valor adecuado. Si en un momento dado queremos pasar de la pantalla de salida a Turbo-Pascal, o viceversa, pulsamos [Alt-F5].

Ejecutamos de esta forma el programa, observando como cambian los valores de las variables. En la tabla siguiente vemos los valores obtenidos al llegar a la línea 27 tras sucesivas iteraciones:

Menor	Mayor	Medio	EsMayorQueMedio
1	5	5	False
1	3	3	False
1	2	2	False
1	2	1	True
1	2	1	True

Podemos observar que el problema reside en que no se limita adecuadamente el rango en que se puede encontrarse el valor buscado, puesto que si sabemos que el valor es mayor estrictamente (no mayor o igual) que medio, entonces el límite inferior no será *medio*, sino *medio*+1. La solución pasa por cambiar la asignación de la línea 24, "*menor* := *medio*" por "*menor* := *medio*+1". Pulsamos [Ctrl-F2] para detener la ejecución y realizamos el cambio.

Ahora el programa adivina el número correctamente, pero surge un último problema: Independientemente de lo que escribamos tras la pregunta "¿Otra vez? [S/N]", el programa vuelve a repetir el proceso de adivinar un número. Parece evidente que el problema reside en la condición de salida del bucle principal (línea 31). En estos casos donde sabemos el lugar del posible error, lo más cómodo es poner un punto de interrupción en esa línea del programa y evaluar los valores de las variables y las expresiones en ese punto. Veamos la manera de hacerlo en Turbo-Pascal:

- Nos situamos en la línea 31 y pulsamos [Ctrl-F8]. Vemos que la línea se resalta en color rojo. Eso significa que hemos establecido un punto de interrupción en esa parte del programa: Cuando la ejecución llegue a esa línea, se detendrá el programa, permitiéndonos examinar el valor de las variables en ese punto.
- Ejecutamos el programa pulsando [Ctrl-F9]. Respondemos "n"a la pregunta "¿Otra vez? [S/N]". Vemos que en ese punto se detiene (temporalmente) el programa y volvemos al editor. Ahora nos interesa ver el valor de la variable *ch* y el resultado de la expresión "(ch='n') and (ch='N')". Pulsamos [Ctrl-F4] y obtenemos el diálogo **Evaluate and Modify**, el cual nos permite evaluar expresiones.
- Escribimos "ch" y pulsamos Intro: Vemos que el valor de la expresión es "n", el carácter que hemos introducido. Escribimos ahora "(ch='n') and (ch='N')" y pulsamos retorno: Vemos que el resultado es **false**, cuando el valor adecuado debería ser **true**, puesto que la expresión anterior establece la condición de salida del bucle.

Por lo tanto la expresión anterior es errónea. Podemos ver fácilmente que un carácter no puede ser simultáneamente igual a "n" y a "N", que es lo que significa la expresión anterior. La expresión

correcta es "(ch='n') **or** (ch='N')" Para comprobarlo lo escribimos en el diálogo, obteniendo el valor adecuado, **true**.

Pulsamos [Esc] para salir del diálogo, y [Ctrl-F2] para detener definitivamente la ejecución. Para eliminar el punto de interrupción nos situamos sobre la línea donde se encuentra y volvemos a pulsar [Ctrl-F8]. Si ahora realizamos el cambio y volvemos a ejecutar el programa, veremos que se ejecuta correctamente.

4. Un caso especial: Uso de read y readln

Para terminar, vamos a examinar un problema relativo al uso de las sentencias **read** y **readln** cuando se leen caracteres. Este problema surge con relativa frecuencia y puede resultar difícil de encontrar si no se conocen bien las características que diferencian a ambas sentencias de lectura.

Pulse [Alt-S] para desplegar el menú **Search** y pulse el comando **Replace..**.

Escriba **readln** en la primera casilla y **read** en la segunda. Asegúrese de que se encuentran marcadas las opciones **Whole words only** y **Prompt on replace**, sitúese sobre el botón **Change all** y pulse Intro.

Turbo-Pascal se sitúa sobre cada aparición de **readln** y nos pregunta si deseamos cambiarlo por **read**. Responderemos "No" en la primera aparición y "Yes" en las siguientes.

Ejecutamos el programa. En la primera pregunta respondemos "si". La salida del programa es la siguiente:

```
Piense en un numero del 1 a 10 y pulse Intro.
¿Es mayor que 5? [S/N] si
¿Es mayor que 8? [S/N]¿Es mayor que 7?[S/N]¿Es
mayor que 6? [S/N]Ya esta! Pensaste en el 6
¿Otra vez? [S/N]
```

El programa no espera a que introduzcamos las respuestas a las tres siguientes preguntas. Lo que sucede es que cuando escribimos "si" y pulsamos Intro, enviamos tres caracteres al programa: [s], [i], y dos caracteres que corresponden a la tecla de retorno. Cada uno de esos cuatro caracteres se interpreta como entrada a cada una de las cuatro preguntas (esto se aprecia mejor volviendo a ejecutar el programa paso a paso y trazando la variable **ch**).

Cuando utilizamos **readln** esto no sucede porque la lectura se efectúa utilizando todos los caracteres hasta el final de la línea, es decir incluyendo los dos caracteres correspondientes a la tecla retorno. En el caso anterior, también se leerían cuatro caracteres, pero solo se devolvería al programa el primero, descartando los tres restantes.

Como norma general, si debemos leer caracteres de teclado es preferible usar **readln**, salvo que el problema que debamos resolver requiera que el programa obtenga todos los caracteres introducidos, incluyendo los que indican el salto de línea.