INTRODUCCIÓN A LA PROGRAMACIÓN
PRÁCTICA 1

Objetivos

- Familiarización con el entorno de programación.
- Ser capaces de editar, compilar y ejecutar de programas con el editor Geany.
- Conocer la estructura de un programa Pascal.

Introducción

El lenguaje Pascal fue diseñado por Niklaus Wirth (Instituto Tecnológico de Zurich) con el objetivo de apoyar la enseñanza de la programación siguiendo los principios de la programación estructurada. Pascal se diseñó siguiendo la línea marcada por los lenguajes Algol-60 y Algol-W, y el primer compilador estuvo disponible en 1970. En 1975 se publicó el libro que sirvió como manual de referencia durante muchos años, "*Pascal. User Manual and Report*" [Jensen 1975]. Durante la década de los ochenta Pascal gozó de mucha popularidad, sobre todo en el ámbito universitario, y la mayoría de universidades lo eligieron para el primer curso de programación. Existen dos definiciones estándar (ISO y ANSI/IEEE), aunque sin duda la versión que más contribuyó a la difusión de Pascal fue *TurboPascal* de Borland.

Nosotros vamos a trabajar con una versión denominada **Free Pascal** que, como su nombre indica, es un software de libre distribución. En los ordenadores del laboratorio ya está instalada. Para instalarla en casa la podemos descargar desde la dirección: *http://www.freepascal.org/*

1. Creación y compilación de un programa fuente

Vamos a utilizar el sistema operativo Linux para hacer nuestras prácticas. Cuando encendamos el ordenador del laboratorio, nos aparecerá un menú en modo texto (el menú de GRUB) para que seleccionemos el sistema operativo que deseamos arrancar. Si en unos segundos no realizamos ninguna selección, arrancará el sistema por defecto, que en este caso es Windows. Debemos seleccionar el sistema operativo **Ubuntu**.

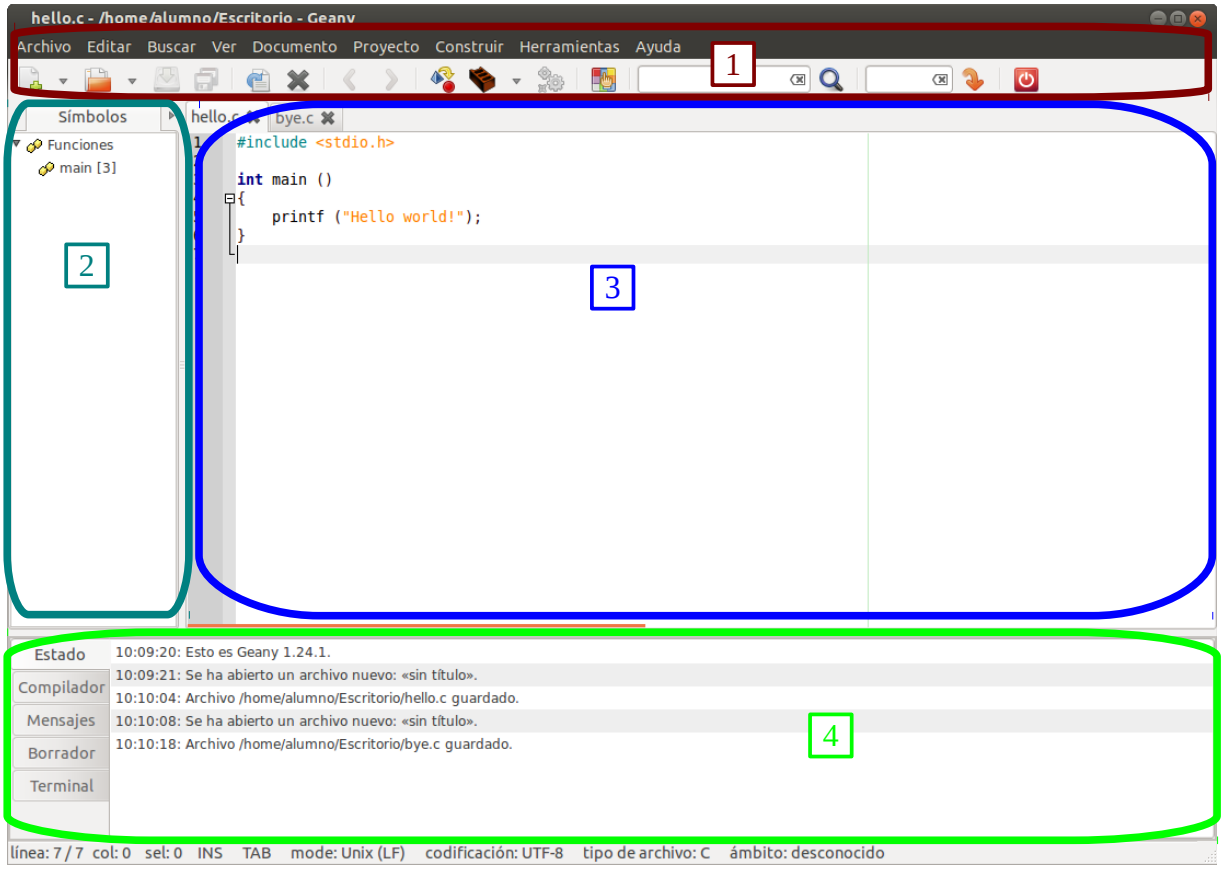
Cuando el sistema arranque nos aparecerá la pantalla de entrada en la que debemos autenticarnos para utilizar el sistema:

Usuario: `alumno`
Contraseña: `alumno`

Tras esto, nos encontraremos en el entorno de escritorio Gnome, que es el que por omisión utiliza Ubuntu.

Free Pascal dispone de un entorno de desarrollo propio. A pesar de que este entorno es suficiente para los propósitos de nuestra asignatura, presenta inconvenientes en el funcionamiento de su interfaz que a veces pueden resultar molestos. Por tanto, utilizaremos otro editor distinto, el Geany, desde el que podremos cubrir de manera sencilla y cómoda todos los pasos en el proceso de elaboración de un programa Pascal.

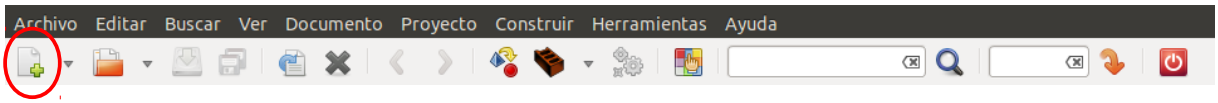
Podemos iniciar Geany seleccionándolo en Aplicaciones → Programación → Geany. Tras ello, nos aparecerá una ventana como la siguiente:



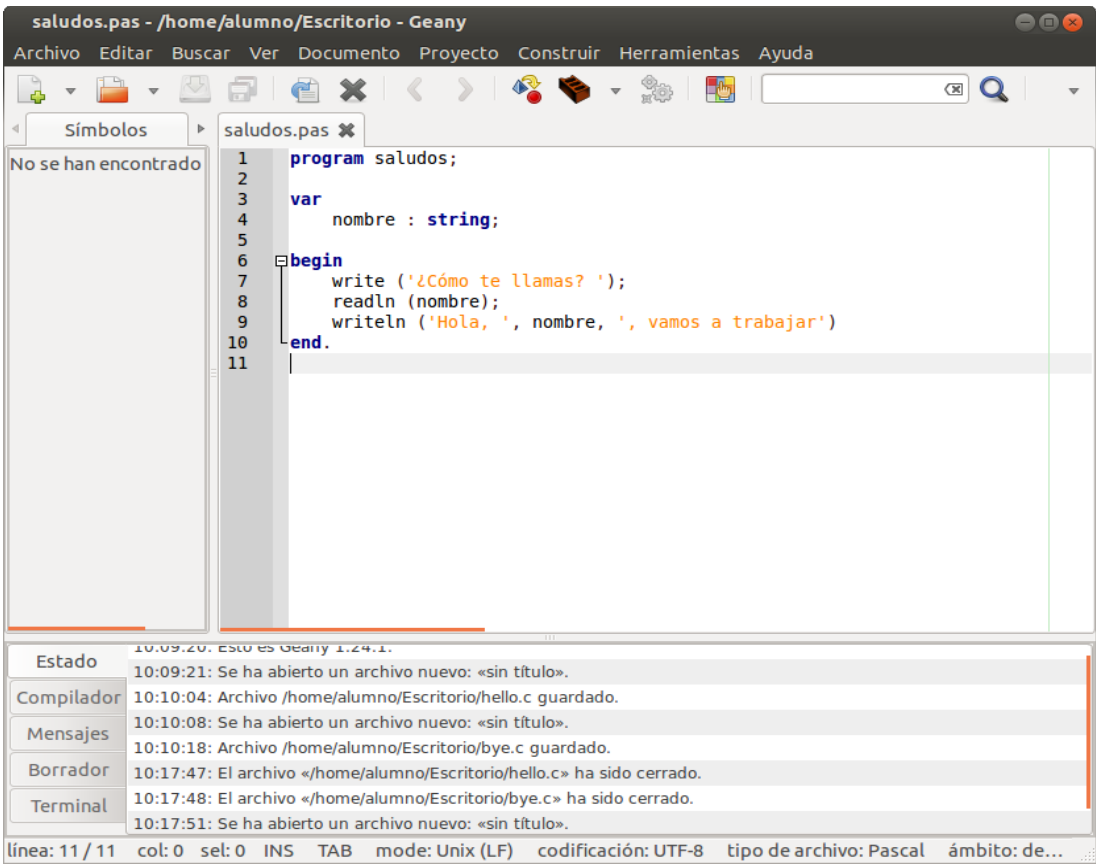
En la ventana del editor Geany podemos distinguir cuatro áreas:

1. El área superior, donde encontramos la barra de menú y botones de acceso rápido.
2. El área izquierda, donde se nos muestra entre otras cosas un resumen de los símbolos que utiliza y define el archivo que estamos editando.
3. El área derecha, que es el área de edición. Aquí es donde escribiremos los programas. Geany recuerda los archivos que se estuviesen editando la última vez que se ejecutó, de modo que es posible que al abrirlo nos encontremos con algún(unos) archivo(s) abierto(s), como en el ejemplo. Si fuera el caso, cerraríamos este(os) archivo(s) haciendo click en el aspa roja a la derecha de la(s) pestaña(s) del área de edición.
4. El área inferior. Es el área de mensajes y estado. A través de las distintas pestañas del lado izquierdo (Estado, Compilador, etc.) podemos recibir información del editor o del compilador, cuando lo utilicemos.

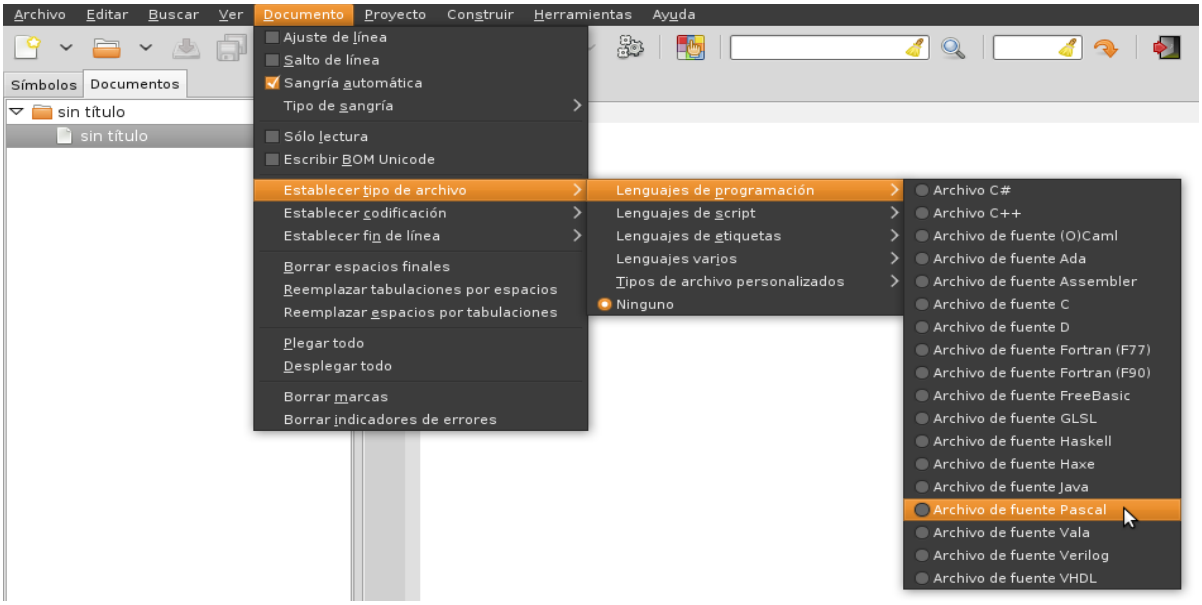
Después de cerrar los archivos que estuviesen abiertos anteriormente, crearemos un nuevo archivo. Para ello podemos utilizar la opción “Nuevo” desde el menú “Archivo”, o bien podemos hacer click directamente en el acceso rápido de creación de nuevo archivo:



Se abrirá entonces una nueva pestaña en el área de edición para crear un nuevo archivo, que inicialmente tendrá como nombre “sin título”. En esta pestaña introduciremos el siguiente programa:

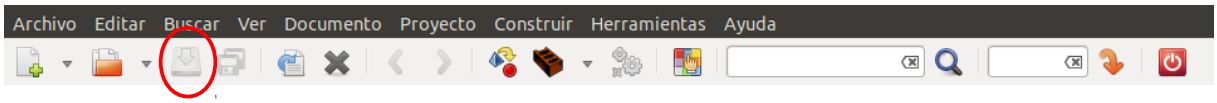


Como puede observarse en este ejemplo, Geany es sensible a la sintaxis de Pascal, así como de otros muchos lenguajes. Sin embargo, inicialmente no se nos resaltará esta sintaxis mientras Geany no sepa cuál es el tipo del archivo. Podemos instruirle sobre cuál es el tipo de sintaxis que debe resaltar indicándoselo manualmente mediante el menú “Documento” → “Establecer tipo de archivo” → “Lenguajes de programación” → “Archivo de fuente Pascal” (véase imagen).



Si no utilizamos esta opción, Geany también detectará automáticamente el tipo de archivo a partir del nombre que le pongamos cuando lo guardemos, como veremos más adelante. Tampoco será necesario hacer esto cuando abramos un archivo existente cuyo nombre indique el tipo del que se trata.

Una vez escrito nuestro programa, lo guardaremos en el disco. Para ello podremos utilizar la opción "Guardar" dentro del menú "Archivo", o bien podemos utilizar el botón de acceso rápido de guardar archivo actual:

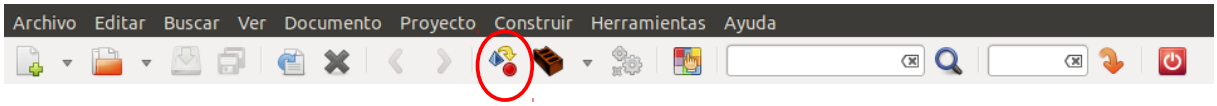


En cualquiera de los casos nos aparecerá una ventana de diálogo solicitándonos el lugar donde queremos guardar el archivo. Tomaremos las siguientes precauciones a la hora de guardar el archivo:

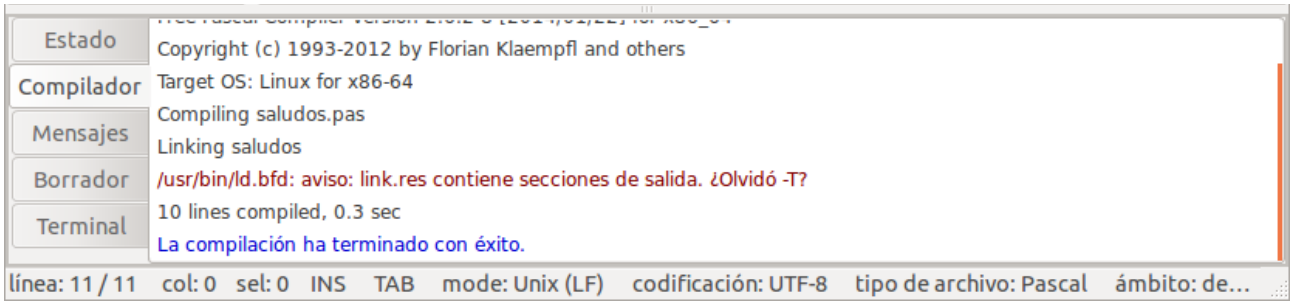
- Utilizaremos un nombre que termine en `.pas` para guardarlo. De este modo indicaremos que se trata de un archivo fuente Pascal.
- No utilizaremos espacios blancos ni caracteres no anglosajones en el nombre del archivo ni en ninguna carpeta de la trayectoria donde se encuentre el archivo.
- Procuraremos que el nombre coincida con el identificador que nombra al programa dentro del texto del programa, terminado en `.pas`. En el ejemplo, por tanto, un nombre adecuado para el archivo que contendrá nuestro programa sería `saludos.pas`.

Una vez escrito y guardado, vamos a compilarlo. Podemos hacerlo de tres formas:

1. Utilizando la opción "Compilar" dentro del menú "Construir".
2. Pulsando la tecla de acceso rápido F8 cuando esté seleccionada la ventana de edición del programa.
3. Haciendo click en el botón de acceso rápido para compilar:



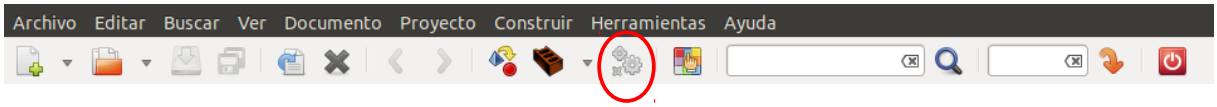
Si todo ha ido bien (si nuestro programa no tiene errores sintácticos), en el área de mensajes se activará la pestaña "Compilador" y se nos mostrará el mensaje de que la compilación ha terminado con éxito:



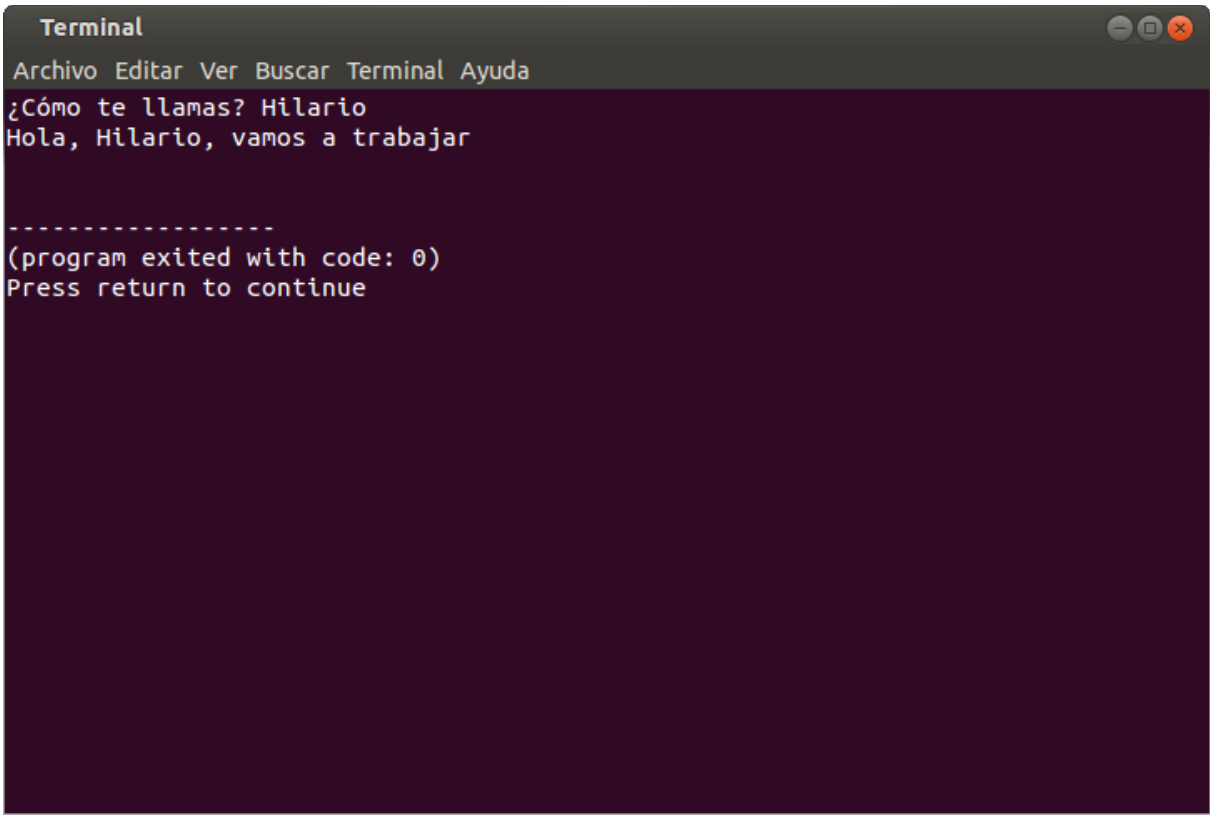
En caso contrario se nos mostrarán los errores y las posiciones donde éstos se encuentran. En la barra inferior de la ventana se nos muestra en todo momento la situación del cursor dentro del archivo (línea y columna). De este modo nos resulta fácil localizar el lugar donde se encuentra el error.

Una vez que el programa ha sido compilado con éxito podemos ejecutarlo. Para ello, al igual que en el caso de la compilación, tenemos tres opciones:

1. Utilizar la opción “Ejecutar” dentro del menú “Construir”.
2. Pulsar la tecla de acceso rápido F5 cuando esté seleccionada la ventana de edición del programa.
3. Hacer click en el botón de acceso rápido para ejecutar:



En cualquiera de los casos se abrirá una ventana de terminal donde se ejecutará el programa. En esta ventana será donde se nos mostrará su salida y donde tendremos que introducir las entradas necesarias. Al terminar la ejecución, esta ventana podría tener un aspecto como el siguiente:



Nótese que únicamente las dos primeras líneas corresponden realmente a la ejecución de nuestro programa. El resto son líneas de información añadidas para indicarnos que el programa ha terminado de una forma normal (o no) y se nos pide que pulsemos intro para cerrar la ventana de ejecución. Al hacerlo, esta ventana desaparecerá.

2. Modificación de un programa fuente

Cree un nuevo fichero e introduzca el siguiente código, compile y ejecute el programa.

```
program segundo;
var
    numero : Integer;
begin
    writeln ('Este es el segundo programa');
    write ('Escribe un número: ');
    read (numero);
    writeln ('La variable numero ha tomado el valor ', numero)
end.
```

Añada las instrucciones siguientes, e intente comprender qué hace cada una de ellas.

```
program segundo;
var
    numero1 : Integer;
    numero2 : Integer;
begin
    writeln ('Este es el segundo programa');
    write ('Escribe un número: ');
    read (numero1);
    writeln ('La variable numero1 ha tomado el valor ', numero1);
    write ('Escribe otro número: ');
    read (numero2);
    writeln ('La variable numero2 ha tomado el valor ', numero2);
    writeln ('Y la suma de ambas es: ', numero1 + numero2)
end.
```

Modifique el programa anterior de forma adecuada para que se lean dos variables enteras de la entrada estándar (teclado). El programa debe mostrar como resultado la suma, la resta y la división de ambos valores.

Abra un nuevo archivo, edite, compile y ejecute el programa correspondiente al algoritmo 2.1 del libro de texto.

```
program mediaponderada;

var
      notaTeoria : Real;
      notaPractica : Real;
      notaFinal : Real;
begin
      Write ('Introduce la nota de teoría y la de prácticas  ');
      ReadLn (notaTeoria, notaPractica);
      notaFinal := notaTeoria * 0.7 + notaPractica * 0.3;
      WriteLn (notaFinal)
end.
```

Anexo: Estructura de un programa Pascal

En la notación estudiada en clase, un algoritmo está formado por dos partes: léxico y algoritmo (o secuencia de instrucciones). Un programa Pascal tiene además una cabecera, la primera línea, que establece el nombre del programa. Por tanto, las tres partes de un programa Pascal serían:

Cabecera

En ella el programador indica el nombre del programa tras la palabra reservada program. Un ejemplo de cabecera sería:

```
program  Nota_media;
```

El nombre es un *identificador* y, como cualquier identificador de Pascal, será una secuencia de caracteres formada por letras y dígitos, cuyo primer elemento debe ser una letra. Los nombres de constantes, variables, procedimientos, funciones y tipos también serán identificadores.

Léxico

De la misma forma que en la notación, en el léxico se declaran las constantes, tipos, variables, procedimientos y funciones del programa, si las hay. La estructura de los procedimientos y funciones es la misma que la de un programa, de modo que se habla del programa principal y de los subprogramas que engloba. En el último ejemplo, el léxico sería el que aparece en el recuadro.

```
var
      notaTeoria : Real;
      notaPractica : Real;
      notaFinal : Real;
```

Secuencia de Instrucciones (sentencia compuesta)

En Pascal se denomina *bloque* a una secuencia de instrucciones delimitada por las palabras reservadas begin y end. Las instrucciones de la secuencia van separadas por punto y coma. La última instrucción no necesita ir seguida de este separador. El programa principal, que sigue al léxico, es un bloque que finaliza con un punto. Un ejemplo de secuencia de instrucciones que es un programa principal sería:

```
begin
      Write ('Introduce la nota de teoría y la de prácticas  ');
      ReadLn (notaTeoria, notaPractica);
      notaFinal := notaTeoria * 0.7 + notaPractica * 0.3;
      WriteLn (notaFinal)
end.
```