**Apuntes prácticas IP**

1. *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.



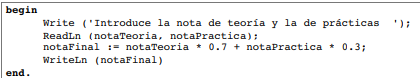
**Secuencia de instrucciones (sentencia compuesta)**

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

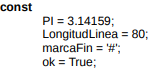
léxico, es un bloque que finaliza con un punto. Un ejemplo de secuencia de instrucciones que es un programa principal sería:



1. *Tipos primitivos y operadores básicos en Pascal*

**Declaraciones de constantes variables**

Una declaración de constantes va encabezada por la palabra reservada const y consta de una o más declaraciones del tipo =, donde denota el identificador de la constante y denota el valor. Un ejemplo sería:

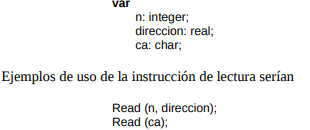


Una declaración de variables va encabezada por la palabra reservada var y consta de una o más declaraciones del tipo: , donde denota el identificador de la variable y denota el tipo de dato. Un ejemplo sería:

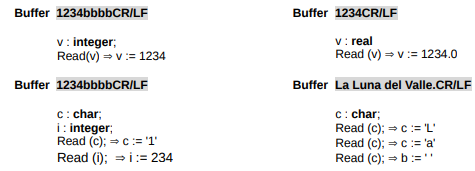


**Entrada y salida de datos**

Existe un conjunto de procedimientos predefinidos que corresponden a las acciones Leer y Escribir de la notación. Para introducir valores por teclado, tenemos el procedimiento Read que puede tener cualquier número de argumentos. Las variables deben ser de tipo integer, real, o char (o string en el caso de Borland Pascal y otros como el FreePascal). Dada la siguiente declaración de variables:



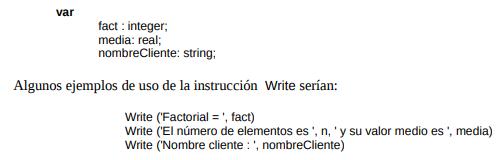
Cuando en la ejecución de un programa Pascal se ejecuta una instrucción Read, los valores que se asignan a las variables son los que se encuentren en el buffer del teclado, una memoria asociada al teclado. Si el buffer se encuentra vacío, no se ejecutará la siguiente instrucción hasta que el usuario no introduzca un valor para cada variable. El final de una entrada de valores se produce cuando se pulsa la tecla de retorno de carro, lo cual supone añadir al buffer los caracteres ASCII de final de línea (CR y LF si estamos trabajando en DOS o Windows). Cuando se leen variables numéricas se pueden introducir varios valores a la vez separados por uno o más espacios, los espacios anteriores al primer valor son ignorados. En el caso de una lectura de valores enteros o reales, la conversión de la secuencia de caracteres que representan al número al correspondiente valor numérico se realiza de forma automática; si la secuencia está mal formada, por ejemplo, incluye caracteres que no son dígitos, se produce un error y la detención del programa. A continuación, vemos cuatro ejemplos de lectura, indicando el contenido del buffer y la asignación de valores a unas variables cuya declaración también se indica. Una b en el contenido del buffer denota un carácter espacio al final, después de los datos, y CR/LF denota los códigos del carácter/los caracteres de final de línea.



En la última lectura, está claro que tan sólo se leen tres caracteres, por lo que no se limpia el buffer, todavía queda la cadena “Luna del Valle.CR/LF”, y se supone que sucesivas lecturas leerán esos caracteres. Sin embargo, en los otros tres casos se podría pensar que el buffer 6 queda vacío al finalizar la lectura, pero no es así, ya que en el primer caso permanecen cuatro caracteres espacio y los de final de línea (CR/LF); en la lectura del valor real permanecen los caracteres de retorno de carro; y en el siguiente caso sucede igual que en el primero. En estos tres casos, una nueva lectura de una variable de tipo carácter le asignaría el carácter espacio o CR/LF (en el caso de sistemas Windows realmente son dos caracteres, como hemos dicho anteriormente, en sistemas Unix y derivados, como Linux, es únicamente un carácter, el de salto de línea LF), y si se tratase de la lectura de una variable numérica se produciría un error. Por ello, Pascal proporciona el procedimiento Readln para entrada de datos, cuyo efecto es tomar del buffer la información que se requiere y limpiarlo. Supuesta la declaración de variables del inicio de este apartado, si se ejecuta Readln (n, direccion) se recogerán del buffer dos valores numéricos separados por espacios y se asignarán a las dos variables n y direccion, y luego se limpiará. Normalmente, como es lógico, las lecturas se realizan con Readln. Es recomendable que cada instrucción de lectura en un programa vaya precedida de una instrucción de escritura que indique al usuario la naturaleza de los valores que debe introducir, por ejemplo:

**Write ('Introduce número de figuras: '); Readln (n)**

Para visualizar datos por pantalla, Pascal proporciona el procedimiento Write que tiene como argumento una lista de expresiones de tipo integer, real, char y cadenas de caracteres. Sean las declaraciones de variables



Finalmente, cabe señalar que no se pueden efectuar operaciones de entrada/salida con variables o expresiones que no sean entero, real carácter y cadena de caracteres. Esto contrasta con lo que hacíamos en los algoritmos escritos con la notación, donde utilizábamos “Leer” y “Escribir” para cualquier tipo, y así convenía que fuese, porque en el nivel de un algoritmo no se debe prestar atención a estos detalles que corresponden al lenguaje de programación.

**La instrucción de asignación**

El operador de asignación en Pascal es := . Sea v una variable y expr una expresión, la

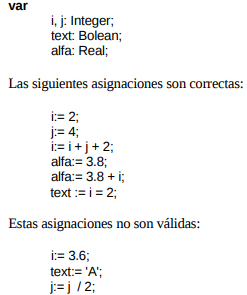
asignación v  expr se expresaría en Pascal como v := expr. Pascal incluye los mismos tipos de

expresión que la notación SP.

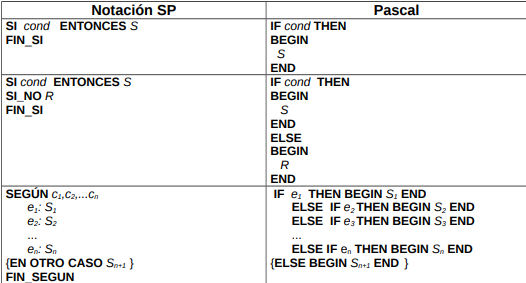
Dado que el resultado de una instrucción de asignación es que la variable v, toma el valor del

resultado de evaluar la expresión **exp**, sólo una variable puede estar en el lado izquierdo de una

sentencia de asignación. La variable y la expresión deben ser del mismo tipo, salvo una excepción, el resultado de una expresión entera se puede asignar a una variable real, no en sentido contrario.

Dada la siguiente declaración:

1. *Análisis de casos en Pascal*

****

En la tabla, **cond**, indica una condición, es decir una expresión booleana cuyo resultado es verdadero o falso; **S** y **R** denotan un bloque de instrucciones. La semántica de la instrucción IF es igual a la estudiada en clase para la composición SI.

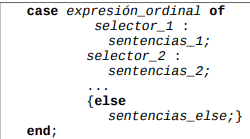
**NOTA:** Cuando sólo hay una sentencia en un caso ésta puede ir sola, sin necesidad de estar encerrada entre un par **BEGIN-END**. Esto solo es necesario cuando hay más de una sentencia que debe ejecutarse en un caso. Como la mayoría de los lenguajes de programación, Pascal dispone de una estructura de selección múltiple llamada case. Aunque esta estructura aparentemente es similar al SEGÚN de la notación algorítmica, presenta importantes diferencias que se enumeran a continuación:

• Sólo es posible efectuar selecciones en función del valor de una única expresión, a diferencia del SEGÚN en el que podíamos poner tantas como fuese preciso.

• La expresión en función de la cual se efectúa la selección ha de ser necesariamente de un tipo ordinal.

• Los valores con los que se compara esta expresión deben ser estáticos. Esto quiere decir que podemos comparar con valores literales, constantes o expresiones cuyo valor sea conocido en tiempo de compilación, y no con valores de expresiones cuyo valor dependa, por ejemplo, de variables.

La sintaxis de la estructura case es la siguiente:

****

La sentencia **case** evalúa la expresión\_ordinal que va a continuación de la palabra **case** y después trataría de encajar el valor obtenido con el de alguno de los selectores. Si el valor encaja con el selector\_i entonces se ejecuta la secuencia de sentencias asociadas a este selector (sentencias\_i). Si no encaja con ningún selector, se ejecutarían las sentencias asociadas a la cláusula **else**, si esta cláusula existe (sentencias\_else). Esta última cláusula, no tiene porqué aparecer siempre.

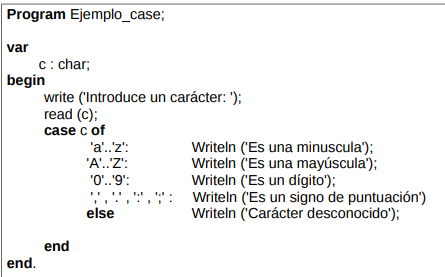
Los selectores pueden estar compuestos por los siguientes elementos:

• Literales o valores constantes del tipo de la expresión\_ordinal

• Intervalos del tipo de la expresión\_ordinal, en la forma: límite\_inferior .. límite\_superior

• Varios elementos de los tipos anteriores separados por comas

En el siguiente cuadro, podemos ver un ejemplo de utilización de esta sentencia

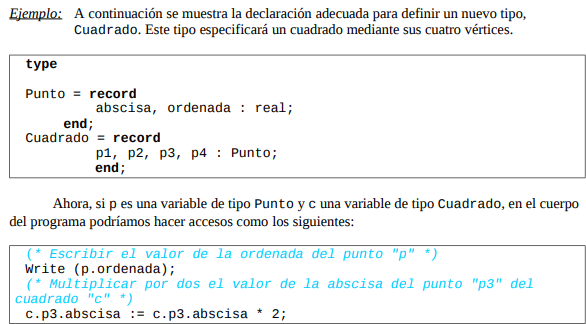
****

1. *Productos de tipos*

Además de los tipos escalares, Pascal proporciona constructores de tipos que nos permiten definir tipos estructurados, entre ellos está el constructor producto de tipos o registro. Todos los tipos definidos por el usuario deberán situarse en el léxico del programa. Al igual que las constantes van encabezadas por la palabra **const**, y las variables por el término var, todos los tipos definidos por el programador se situarán en una sección que comienza con la palabra reservaba **type**. Un tipo, producto de tipos, se define en Pascal mediante el par constructor **record – end**, dentro del cual se listan los campos que componen el producto de tipos de igual modo que si se tratase de una declaración de variables.

El acceso a los campos de variables de un tipo producto de tipos se hace del mismo modo que vimos en clase de teoría, es decir, mediante el operador punto de la forma: variable.campo.

****

****

1. *Acciones y parámetros en Pascal*

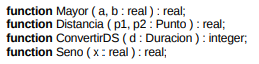
Al igual que en la notación empleada en las clases de teoría, las acciones y las funciones en el leguaje Pascal se declaran en la zona destinada al léxico. La estructura de los procedimientos y funciones es la misma que la de un programa, de modo que se habla del programa principal y de los subprogramas que engloba.

En la cabecera de un procedimiento, la palabra reservada **procedure** va seguida del nombre y de una lista de parámetros, si los hay, encerrados entre paréntesis. Las declaraciones de parámetros son iguales a las declaraciones de variables que hemos visto, pero ahora el identificador denota a un parámetro y no a una variable. En Pascal existen dos tipos de paso de parámetro: por valor y por referencia, que corresponden respectivamente a los pasos de parámetro dato y dato-resultado de la notación. Una declaración de parámetro debe ir precedida de la palabra reservada **var** para indicar un paso por referencia. Por defecto, si la declaración del parámetro no va precedida por **var**, se supone que se trata de un paso por valor. Ejemplos de declaraciones de procedimientos serían:



1. *Funciones en Pascal*

Una función se declara igual que un procedimiento, sólo que la declaración comienza con la palabra reservada **function**, y debe acabar indicando el tipo de dato del valor de retorno, separado del resto de la declaración por dos puntos. Ejemplos de declaraciones de funciones serían:



Como es lógico, los parámetros de una función deben ser de tipo paso por valor, ya que una función debe calcular un valor a partir de los argumentos, pero no utiliza argumentos con paso por referencia para retornar valores. En Pascal estándar, sólo se permite que las funciones devuelvan valores de tipos escalares. Por tanto, un programador no podría construir una función que devolviese, por ejemplo, un valor de tipo Cuadrado, y en su lugar, tendría que construir un procedimiento que incluyese un parámetro paso por referencia de este tipo. El compilador que nosotros usamos no tiene esta restricción.