# INTRODUCCIÓN A LA PROGRAMACIÓN
# PRÁCTICA 2

## Objetivos

- Identificación de las variables y constantes que aparecen en los problemas.
- Consolidar los conceptos de: tipo, secuenciación y estado de un programa.
- Utilización de las sentencias de asignación y entrada salida.

Esta práctica se entrega con un anexo que contiene la información necesaria para implementar en FreePascal los algoritmos que se proporcionan en pseudocódigo. Para mas información puede consultar el manual disponible con el compilador FCP o la página: *http://www.freepascal.org/*

## 1. Intercambio circular de tres variables

Dadas tres variables A, B, C escribir la secuencia de instrucciones necesarias para intercambiar entre sí sus valores del modo siguiente: B toma el valor de A, C toma el valor de B, A toma el valor a C.

## Solución

La sentencia de asignación sustituye el valor que en ese instante contiene la variable. Por tanto, es necesario introducir una variable auxiliar si queremos preservar los valores de las tres variables. La solución del problema sería la misma para variables de cualquier tipo, en este caso utilizaremos variables de tipo carácter.

**Léxico**

```
    a, b, c: carácter;        (datos de entrada y de salida);
    aux: carácter;            (variable auxiliar);
```

**Algoritmo**

```
    Leer (a,b,c);
{e1: las variables contienen los valores introducidos por teclado}
    aux ←b;
{e2: a, b, y c no se han modificado, aux contiene el mismo valor que b}
    b ←a;
{e3: a y c no se han modificado, b contiene el mismo valor que a}
    a ← c;
{e4: c no modificado, a y b contienen el valor  inicial de c y de a respectivamente}
    c ← aux;
{e5: los valores de a , b y c se han desplazado de forma circular}
    Escribir (a, b, c);
Fin.
```

- ✓ Realice una traza del algoritmo
- ✓ Si se intercambia el orden de las sentencias de asignación, ¿qué ocurriría?
- ✓ Codifique el algoritmo en Pascal de forma que informe por pantalla de los estados intermedios de las variables.

## *2. Desglose de dinero*

Escriba un algoritmo que dada una cantidad de dinero la desglose según los billetes habituales en España (usaremos los de 5, 10, 20, 50, 100 y 500 euros). Se desea encontrar el número mínimo de billetes. No hay limitación en el número de billetes disponibles.

## *Solución*

Se debe seguir una estrategia similar a la aplicada en el ejercicio de clase de obtener las horas, los minutos y los segundos a partir de una cantidad de segundos. Primero se intentarán obtener los billetes de 500, luego los de 100, los de 50, los de 20, los de 10 y por último los de 5. Se indicará también el resto que debe ser contabilizado en monedas.

**Léxico**

```
cantidad: entero;
resto: entero;
quinientos, cien, cincuenta, veinte, diez, cinco: entero;
```

**Algoritmo**

```
Leer (cantidad);
quinientos ← cantidad DIV 500;
resto ←  cantidad MOD 500;
cien ←  resto DIV 100;
resto ←  resto MOD 100;
cincuenta ←  resto DIV 50;
resto ←  resto MOD 50;
veinte ← resto DIV 20;
resto  ← resto  MOD 20;
diez ← resto DIV 10;
resto  ← resto MOD 10;
cinco  ← resto DIV 5;
resto  ← resto MOD 5;
Escribir ( quinientos, “de quinientos euros”);
Escribir (cien, “de cien euros”);
Escribir (cincuenta, “de cincuenta euros”);
Escribir (veinte, “de veinte euros”);
Escribir (diez, “de diez euros”);
Escribir (cinco, “ de cinco euros”);
Escribir (resto, “ en monedas”);
```

**FIN**.

- ✓ Realice una traza del algoritmo
- ✓ Si se intercambia el orden de las sentencias de asignación, ¿qué ocurriría?
- ✓ Codifique el algoritmo en Pascal de forma que informe por pantalla de los estados intermedios de las variables.

### 3. Binario a decimal

Escriba un algoritmo que, para un número binario de cuatro cifras, imprima su valor en base 10. Los datos del problema se representan mediante enteros (0 o 1). Por ejemplo, 1, 1, 0, 1.

- ✓ ¿Qué variables necesita?
- ✓ Diseñe el algoritmo en SP
- ✓ Codifique el algoritmo en Pascal

## Trabajo personal del alumno

### Problema 1

La empresa de bajo conste JetLag, desea obtener una serie de información para su departamento de contabilidad. En cada vuelo, el asistente de cabina anota los kilómetros recorridos y los litros de combustible utilizados. Además, por cada kilómetro recorrido el avión tiene unos gastos fijos de mantenimiento. La empresa desea conocer, por cada viaje: el coste total del viaje, los kilómetros recorridos por litro de combustible y el coste por kilómetro. Diseñe un algoritmo que facilite el trabajo a los empleados de contabilidad.

- ✓ Analice cuáles son los datos de entrada del algoritmo
- ✓ Determine cuál debe ser la salida del algoritmo
- ✓ Diseñe el algoritmo en notación SP y realice una traza a mano
- ✓ Implemente el algoritmo en Pascal

### Problema 2

Para calcular la nota final de un alumno se tiene en cuenta tres calificaciones: la obtenida en el examen, la nota del trabajo realizado en casa, y el resultado de un test cada una de ellas ponderada al 50%, 30% y 20%, respectivamente. Realice un programa en Pascal que dadas las tres notas muestre la nota final. Siga los pasos sugeridos en el problema uno.

### Problema 3

A una empresa de transporte le han encargado el traslado de cerdos ibéricos, de todos es conocido que si este tipo de cerdo sufre estrés la calidad de su carne baja considerablemente. Por ello deben ser transportados en contenedores especiales. Estos contenedores están disponibles en cuatro tamaños, que pueden albergar 50, 25, 5, y 1 gorrinos, respectivamente. Escriba un programa que dado el número de animales a trasportar, calcule el número de contenedores de cada tipo que se necesitan, usando el menor número posible.

# Anexo

## Tipos primitivos y operadores básicos en Pascal

La tabla de la figura indica los tipos de datos de Pascal que corresponden a los tipos básicos de la notación:

| Notación | Pascal |
|---|---|
| Entero | integer |
| Real | real |
| Booleano | boolean |
| Carácter | char |

Como ya señalamos, estos tipos de datos son llamados *tipos escalares* porque su dominio consiste en entidades indivisibles que no pueden descomponerse en partes más elementales. Las versiones de Pascal pueden proporcionan variantes de los tipos real y entero, en función de que éstos se consideren con mayor o menor precisión o longitud, con signo o sin signo, etcétera. Para un mayor detalle sobre estas variantes se remite al *manual de referencia del lenguaje* del compilador que, como ya se ha mencionado se encuentra en la dirección *http://www.freepascal.org/*

Los operadores de los tipos básicos de se expresan de la misma forma en Pascal, que en la notación SP, excepto los que aparecen a continuación:

| Notación | Pascal |
|---|---|
| × | * |
| ≠ | <> |
| Y | and |
| O | or |
| NO | not |
| Predecesor | pred |
| Sucesor | succ |
| Carácter | chr |
| Código | ord |

Los formatos para la representación de los números enteros y reales en la memoria de un ordenador sólo permiten representar un subconjunto de los enteros y de los reales. En el caso de los enteros, aunque la representación es exacta, el entero más alto y más bajo que puede representarse depende del número de bits empleado por el compilador. En el caso del FreePascal es el siguiente:

| Tipo | Rango | Tamaño en bytes |
|---|---|---|
| Shortint | -128 .. 127 | 1 |
| Integer | -32768 .. 32767 | 2 |
| Longint | -2147483648 .. 2147483647 | 4 |

Como la memoria de un ordenador tiene un tamaño finito no es posible representar con exactitud los números reales en un ordenador, sino que cada valor del tipo real *representará* a

un intervalo de la recta real que contiene infinitos reales. Por tanto, los cálculos con valores reales serán aproximados, al contrario de lo que ocurre con los enteros, y dependiendo del ordenador existirá un rango y una precisión. La precisión determina la anchura del intervalo de los reales que es asociado a cada *representante* de un número real, a mayor precisión menor anchura. El rango determina cuáles son los valores extremos que podemos representar, es decir, cuál es el infinito del tipo real.

Con estas consideraciones, se deduce que no es adecuado hacer comprobaciones de igualdad donde intervengan variables reales, debido al carácter aproximado de los cálculos. Así puede suceder que 3*(1/3) sea distinto de 1. Conviene, por tanto, en vez de comprobar la igualdad de dos valores reales, comprobar que el valor absoluto de su diferencia es menor que una constante cercana a cero, tanto como lo requiera la magnitud de los valores. Como curiosidad y como muestra evidente de este problema, se puede intentar calcular en el compilador Pascal el resultado de las dos expresiones 1.0 – 0.2 – 0.8 y 1.0 – 0.8 – 0.2. En principio, ambas deberían dar idéntico resultado, el valor cero, pero no sucede así.

El tipo char presenta como dominio el conjunto finito y ordenado de caracteres. Internamente a cada carácter se le hace corresponder un código numérico de acuerdo con alguna normalización. Como indicamos en el primer capítulo, los dos códigos más extendidos son ASCII y Unicode. En Pascal encontramos las funciones Ord y Chr que corresponden a las funciones Código y Carácter de la notación. Dado un carácter cualquiera, Ord devuelve su código, y dado un entero en un intervalo de enteros, que dependerá de la instalación y del juego de caracteres que se emplee, Chr devuelve el carácter asociado a ese código. Por ejemplo, para el código ASCII, Chr (66) retorna 'B' y Ord ('B') retorna 66.

Los literales numéricos y carácter se forman igual que en la notación. También pueden representarse literales reales en *notación científica*, es decir, como un número real seguido de un exponente de base diez, como por ejemplo 1.0e+3 = 1000.0, o 3.455e–5 = 0.00003455. Los literales booleanos Verdad y Falso se expresan, respectivamente, como true y false.

## *Declaraciones de constantes, variables*

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

```
const
        PI = 3.14159;
        LongitudLinea = 80;
        marcaFin = '#';
        ok = True;
```

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

```
var
        numeroCaracteres : integer;
        ca, letra : char;
        longitudMedia: real;
```

## *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 FreePacal). Dada la siguiente declaración de variables:

```
var
    n: integer;
    direccion: real;
    ca: char;
```

Ejemplos de uso de la instrucción de lectura serían

```
Read (n, direccion);
Read (ca);
```

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.

**Buffer** **1234bbbbCR/LF**

```
v : integer;
Read(v) ⇒ v := 1234
```

**Buffer** **1234CR/LF**

```
v : real
Read (v) ⇒ v := 1234.0
```

**Buffer** **1234bbbbCR/LF**

```
c : char;
i : integer;
Read (c); ⇒ c := '1'
Read (i);  ⇒ i := 234
```

**Buffer** **La Luna del Valle.CR/LF**

```
c : char;
Read (c); ⇒ c := 'L'
Read (c); ⇒ c := 'a'
Read (c); ⇒ b := ' '
```

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

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

```
var
        fact : integer;
        media: real;
        nombreCliente: string;
```

Algunos ejemplos de uso de la instrucción Write serían:

```
Write ('Factorial = ', fact)
Write ('El número de elementos es ', n, ' y su valor medio es ', media)
Write ('Nombre cliente : ', nombreCliente)
```

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:

```
var
    i, j: Integer;
    text: Bolean;
    alfa: Real;
```

Las siguientes asignaciones son correctas:

```
i:= 2;
j:= 4;
i:= i + j + 2;
alfa:= 3.8;
alfa:= 3.8 + i;
text := i = 2;
```

Estas asignaciones no son válidas:

```
i:= 3.6;
text:= 'A';
j:= j  / 2;
```