INTRODUCCIÓN A LA PROGRAMACIÓN
PRÁCTICA 4

Objetivos

- ✓ Introducir nuevos elementos en el léxico de un programa: producto de tipos, acciones y funciones.
- ✓ Aprender a definir acciones y funciones en un programa Pascal, introducir los conceptos de parámetro y léxico local.
- ✓ Estudiar las implicaciones de todo lo anterior en programas Pascal resolviendo problemas sencillos.

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 la 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`.

Notación	Pascal
$< c_1 : t_1,..., c_n : t_n >$	**record** $c_1 : t_1$; ... $c_n : t_n$ **end**

Ejemplo: A continuación se muestra la declaración adecuada para definir un nuevo tipo, `Cuadrado`. Este tipo especificará un cuadrado mediante sus cuatro vértices.

```
type

Punto = record
          abscisa, ordenada : real;
      end;
Cuadrado = record
          p1, p2, p3, p4 : Punto;
          end;
```

Ahora, si `p` es una variable de tipo `Punto` y `c` una variable de tipo `Cuadrado`, en el cuerpo del programa podríamos hacer accesos como los siguientes:

```
 (* Escribir el valor de la ordenada del punto "p" *)
 Write (p.ordenada);
 (* Multiplicar por dos el valor de la abscisa del punto "p3" del
cuadrado "c" *)
 c.p3.abscisa := c.p3.abscisa * 2;
```

Problema 1

Reescriba el algoritmo que calcula "*La fecha del día siguiente*", utilizando el producto de tipos para definir el tipo fecha. Recordemos el enunciado: Diseñe un algoritmo y transcríbalo en forma de programa Pascal que lea de la entrada una fecha (día, mes, año) y ofrezca como salida la fecha del día siguiente. Supondremos que el año no es bisiesto.

2. 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:

```
procedure Intercambio ( var a, b : real );
procedure DibujarCuadrado ( c : Cuadrado );
procedure BorrarPantalla;
```

Problema 2

A continuación se presenta un pequeño listado en Pascal que utiliza dos acciones estudiadas en clase. Identifique los siguientes elementos: nombre de las acciones, parámetros *dato*, parámetros *dato_resultado*, parámetros formales y parámetros reales. Las dos acciones se presentan sin precondición ni postcondición, asígneles las que le parezcan adecuadas. Sin ejecutar el programa, indique qué resultado aparecerá en pantalla. Por último, modifique el programa para que la salida sea la siguiente:

```
El valor de las variables x, y, z es: CBA
El valor del número 1 es: 13
El valor del número 2 es:   0
```

```
program  acciones;

var
        b1, b2, decimal1, decimal2: Integer;
        x, y, z: Char;

procedure BinDec ( b1, b2 , b3, b4 : Integer; var dec : Integer );
begin
        dec := b1*1 + b2*2 + b3*4 + b4*8
end;
```

```
procedure Intercambio ( var a, b: Char );
var
        aux:Char;
begin
        aux := a;
        a := b;
        b := aux
end;

begin
        b1 := 0;
        b2 := 1;
        x := 'O';
        y := 'P';
        z := 'Q';
        Intercambio (x, y);
        Intercambio (y, z);
        BinDec (b1, b2, b1, b2, decimal1);
        BinDec (0, 1, b1, b1, decimal2);
        WriteLn ('El valor de las variables x, y, z es: ', x, y, z);
        WriteLn ('El valor del numero 1 es : ', decimal1);
        WriteLn ('El valor del numero 2 es : ', decimal2)
end.
```

Problema 3: Desglose de dinero

Diseñe una acción para realizar la división entera. Esta acción recibe como parámetros el dividendo y el divisor, y devuelve el cociente y el resto. Utilice dicha acción para resolver el problema del "*Desglose de monedas*".

- ✓ Diseñe el algoritmo en notación SP.
- ✓ Implemente el algoritmo en Pascal.

3. 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:

```
function Mayor ( a, b : real ) : real;
function Distancia ( p1, p2 : Punto ) : real;
function ConvertirDS ( d : Duracion ) : integer;
function Seno ( x : real ) : real;
```

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.

Problema 4. El máximo otra vez

En clase de teoría se han estudiado diferentes soluciones para la obtención del máximo de tres números. Se proporciona una función en Pascal que obtiene el máximo de dos números, construya una función, Max3, que haciendo uso de Max2, mediante composición funcional, obtenga el máximo de tres números**.**

```
function  max2 (x, y : integer): integer;
begin
      if x>y then max2 :=x
             else max2 :=y
end;
```

Problema 5. Perímetro de un triángulo

En el listado que se muestra a continuación se define un nuevo tipo de datos, no primitivo, y una función. Identifique: el nombre del tipo, el nombre de la función, las variables de ese tipo, los parámetros reales y formales, y las llamadas a la función. Una vez comprendido qué hace este programa y cómo lo hace. Realice las siguientes tareas:

- ✓ Defina de manera conveniente los tipos de datos `Punto` y `Triángulo`
- ✓ Defina un procedimiento que solicite al usuario, (lea de la entrada), los datos que definen un triángulo. Esta acción debe devolver el triángulo leído mediante un parámetro de la clase resultado
- ✓ Defina una función que calcule la distancia entre dos puntos del plano. Si (*x1*, *y1*) y (*x2*, *y2*) son dos puntos del plano, la distancia *d* entre ellos se calcula mediante la siguiente fórmula:

$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

- ✓ Diseñe un algoritmo que haciendo uso de todo lo anterior calcule el perímetro de un triangulo.

```
program productotipos;

type
      Punto = record    (* definición del tipo punto como producto de tipos *)
            abscisa  : REAL;
            ordenada : REAL
      end;

var
      p1, p2 : Punto;   (* variables de entrada, puntos*)
      distancia : REAL; (* variable de salida, distancia entre los puntos*)

function sqr (x : REAL) : REAL;
(* Pre: x es un número Real *)
begin
        sqr := x*x
 (* Post: Devuelve el cuadrado de valor pasado como parámetro*)
end;
```

```
begin

(* Introducción de datos *)
      Write ('Introduce el valor de abscisa: ');
      ReadLn (p1.abscisa);
      Write ('Introduce el valor de la ordenada: ');
      ReadLn (p1.ordenada);
      Write ('Introduce el valor de abscisa: ');
      ReadLn (p2.abscisa);
      Write ('Introduce el valor de la ordenada: ');
      ReadLn (p2.ordenada);


(* cálculo de la longitud del segmento definido por los dos puntos *)

      distancia:= sqrt (sqr (p1.abscisa – p2.abscisa) +
                        sqr (p1.ordenada - p2.ordenada));

(* mostramos el resultado *)

      Write (distancia:6:2)
end.
```

4. Trabajo personal del alumno

Problema 1.

Supongamos que dado un polinomio de segundo grado $P(x) = ax^2+bx+c$, deseamos encontrar las raíces de este polinomio, es decir, los valores de x tal que $P(x) = 0$. Haremos uso de la conocida expresión:

$$r_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Cuando el radicando es mayor o igual que cero $\left(b^2 - 4ac \geq 0\right)$, las dos raíces reales se calculan directamente según la expresión anterior. Sin embargo, cuando este valor es menor que cero aparecen dos raíces complejas conjugadas con partes imaginarias de distinto signo que pueden calcularse según la expresión siguiente, derivada de la anterior, en la que i representa la unidad imaginaria:

$$r_{1,2} = \frac{-b}{2a} \pm i \cdot \frac{\sqrt{-\left(b^2 - 4ac\right)}}{2a} = \frac{-b}{2a} \pm i \cdot \frac{\sqrt{4ac - b^2}}{2a}$$

Diseñe un programa que dados los valores *a*, *b*, y *c*, calcule las raíces del polinomio. Utilice una acción, o una función, lo que considere mas adecuado para realizar los cálculos. Para comprobar la corrección de éste, a continuación se lista una tabla de valores de *a*, *b*, y *c* con los correspondientes resultados correctos:

a	*b*	*c*	Raíz 1	Raíz 2
1	1	1	-0.5 + 0.866 i	-0.5 - 0.866 i
1	2	1	-1	-1
2	5	2	-2	-0.5
4	0	1	0.5 i	-0.5 i

Problema 2

A continuación se muestra un listado de un programa lleno de errores; errores sintácticos, semánticos y lógicos. Antes de compilar y ejecutar el programa intente descubrirlos y modifique el programa para que sea correcto. El compilador le puede dar mensajes de error, pero muchas veces no se corresponden con la causa principal del error. Aun estando correctamente escrito, puede seguir siendo incorrecto. Compruebe manualmente que los resultados se corresponden a los datos introducidos.

Se insiste en la modificación del programa sobre el papel. Hacer continuos cambios directamente en el programa hasta que funcione es una **pésima** manera de solucionar los problemas y una gran perdida de tiempo que sólo lleva a programas incorrectos dignos del peor de los aficionados.

El enunciado del problema seria el siguiente:

Dados la longitud de los lados de un rectángulo y el radio de un círculo, obtener el área y el perímetro del rectángulo, así como el área y el diámetro del círculo.

```
program geometria;
        Dato1, Dato2: real;     {datos de entrada del círculo o del rectángulo}
        Resul, Resul2: real;    {datos de salida}

prodecure arearec (lado1,lado2 : real;  area, perimetro : real);
begin
        area := lado1*lado2;
        perimetro := 2* (lado1+lado2);
end;

procedure areacir (radio : real; VAR  area, diametro);
begin
        area := radio*radio*3.1416;
        diametro := 2*radio;
end;

begin
        Writeln ('introduce la longitud de un lado ');
        Readlen (dato1);
        Writeln (introduce la  longitud del otro lado '):
        Readlin (dato2);
        Resul1 := arearec (dato1,dato2, area, perimetro)
        Writeln ('introducce el radio del circulo);
        Readn (dato2);
        areacir (dato2, resul1, resul2)
        Writeln ('el area del rectángulo es ' , resul1, ' el perimetro ' resul2 );
        Writeln ('el area del círculo es ', resul2, ' el diametro ', resul2)
end.
```