Introducción a la programación - Ejercicios Tema 3

DISIA - Facultad de Informática UCM (2009-2010)
Ing. Técnica en Informática de Gestión - 1º B
Profesores: Federico Peinado Gil y Pablo Moreno Ger

Ejercicio 1.- En las construcciones siguientes ¿para qué valores de la variable *RangoI* se ejecutará la sentencia de asignación?

```
a) if not (RangoI / 4 <= 12) then
     total := cantidad * 4

b) if ((RangoI > 1) and (RangoI  < 5)) or (RangoI = 8) then
     valor := 0.15

c) if RangoI > 1 then
     if (RangoI < 5) or (RangoI = 8) then
               valor := 0.15
```

Ejercicio 2.- Rescribe las sentencias siguientes utilizando, exclusivamente, construcciones if-then-else completas.

```
a) if x >= 0 then y := x;
    if x < 0 then y := -x;

b) if disponible > precio then
     writeln ('Disponible es mayor que precio');
    if disponible = precio then
     writeln('Disponible es igual que precio');
    if disponible < precio then
     writeln('Disponible es menor que precio');

c) if a <= 150 then
     c := 50 + a;
   if (a > 150) and (a <= 250) then
     c:= 50 + 1.18 * a;
   if a > 250 then
      c :=  50 + 1.18 * a + 1.03 * a;
```

Ejercicio 3.- Indica qué presenta por pantalla el siguiente fragmento de código:

```
numero:= 3;
respuesta := not True and False;
if numero mod 2 > 0 then
     if respuesta or (numero / 5 > 3) then
               writeln('Primera instrucción')
else
     writeln('Segunda instrucción');
writeln('Tercera instrucción')
```

Ejercicio 4.- Escribe un programa que solicite un carácter por teclado y si es una letra minúscula muestre por pantalla la letra en mayúsculas. En cualquier otro caso, el programa mostrará un mensaje indicando que el carácter introducido no era una letra minúscula.

Ejercicio 5.- Escribe un programa que lea tres números enteros y devuelva el mayor de los tres.

Ejercicio 6.- Escribe un programa que lea una temperatura introducida a través de teclado y muestre por pantalla la actividad más apropiada para dicha temperatura teniendo en cuenta los siguientes criterios.

ACTIVIDAD	TEMPERATURA IDONEA
Natación	temp > 30
Tenis	20 < temp <= 30
Golf	10 < temp <= 20
Esquí	5 < temp <= 10
Parchís	temp <= 5

Ejercicio 7.- Debido a la pertinaz sequía sufrida años atrás en algunos puntos de la geografía española, se decidió poner en práctica un sistema de cobro de agua que penalizara el consumo excesivo de la forma que se indica en la tabla siguiente:

Consumo (m^3)	Euros/ m^3
Primeros 100	0.15
De 100 a 500	0.20
De 500 a 1000	0.35
Más de 1000	0.80

Escribe un programa que lea de teclado los metros cúbicos consumidos y presente en pantalla el coste de agua total. Tener en cuenta que en la tabla se indica lo que hay que cobrar por los m^3 que se encuentran en el intervalo. Así, si hemos consumido 750 m^3 deberíamos pagar:

100 0.15 + 400 * 0.20 + 250 * 0.35 = 182.5 euros

Ejercicio 8.- ¿Cuál es el valor de *a* que se muestra por pantalla tras ejecutarse los siguientes trozos de código, suponiendo las variables *a* y *J* de tipo integer, *I* de tipo boolean y *K* de tipo char?

a)
```
a := 0;
   for I := false to true do
    for J := 8 downto 2 do
              for K := ‘a’ to ‘e’ do
                        a := a + 1;
   writeln(a);
```

b)
```
a := 0;
   for I := true to false do
    for J := 8 downto 2 do
              for K := ‘a’ to ‘e’ do
                        a := a + 1;
   writeln(a);
```

c)
```
a := 0;
   for I := false to true do
    for J := 8 downto 2 do
              for K := ‘e’ downto ‘a’ do
                        a := a + 1;
   writeln(a);
```

d)
```
a := 0;
   for I := false downto true do
    for J := 8 downto 2 do
              for K := ‘a’ to ‘e’ do
                        a := a + 1;
   writeln(a);
```

e)
```
a := 0;
    for I := false to true do
    for K := ‘a’ to ‘e’ do
              for J := 8 downto 2 do
                        a := a + 1;
    writeln(a);
```

f)
```
a:= 0;
   for I:= false to true do
       for J:= 8 downto 2 do
          while K <= 'e' do
               begin
                    a:= a + 1;
                    K:= chr(ord(K) + 1);
               end;
   writeln(a);
```

g)
```
a:= 0;
    K:= 'a';
    for I:= false to true do
       for J:= 8 downto 2 do
          while K <= 'e' do
               begin
                    a:= a + 1;
                    K:= chr(ord(K) + 1);
               end;
    writeln(a);
```

Ejercicio 9.- Escribe un programa que lea diez números enteros y devuelva el mayor de todos ellos.

Ejercicio 10.- Escribe un programa que lea caracteres por teclado hasta que el usuario introduzca un *. Los caracteres se solicitan e introducen uno a uno. El programa debe contar el número de espacios en blanco, guiones, letras mayúsculas y letras minúsculas introducidas y cuando finaliza la ejecución mostrará cuántos caracteres de cada tipo han sido introducidos.

Modifica el programa de modo que se introduzca una cadena de caracteres terminada con un salto de línea.

Ejercicio 11.- Escribe un programa que invierta números enteros positivos. El programa actuará de forma cíclica, solicitando un número en cada pasada y, si no es un número negativo, lo invertirá. El programa finaliza cuando se introduce un número negativo.

En este caso, se entiende por invertir el dar la vuelta a los dígitos que componen el número (*hallar su imagen especular*), esto es, el inverso de 3952 es 2593.

Ejercicio 12.- Dada una secuencia de caracteres introducidos por teclado y terminada en salto de línea, escribe un programa que cuente el número de caracteres anteriores a la aparición de la primera *a*. El programa devolverá un mensaje si en la secuencia de entrada no existe ninguna *a*.

Ejercicio 13.- Escribe un programa que solicite dos números enteros positivos y calcule su producto usando sólo sumas.

Ejercicio 14.- Examen febrero 97-98, 3 puntos. Implementa un programa que lea de teclado secuencias de caracteres terminadas en fin de línea y, para cada secuencia, cuente y muestre por pantalla el número de blancos, letras (mayúsculas y minúsculas) y dígitos existentes entre una pareja de signos de admiración. Si sólo aparece el signo de admiración inicial, el recuento se realizará hasta el final de la secuencia. Hay que tener en cuenta que puede no haber ninguna pareja de signos de admiración. El programa solicitará secuencias de caracteres hasta que se introduzca un fin de línea (<eoln>) como secuencia. Un ejemplo de ejecución del programa es el siguiente:

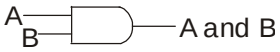
```
Entrada:  Esto es ¡una prueba! de secuencia de entrada<eoln>
Salida:   Blancos: 1    Letras: 9    Dígitos: 0
Entrada:  Esto es la prueba de secuencia ¡de entrada 2<eoln>
Salida:   Blancos: 2    Letras: 9    Dígitos: 1
Entrada:  Esto es la prueba de secuencia de entrada 3<eoln>
Salida:   Blancos: 0    Letras: 0    Dígitos: 0
Entrada:  <eoln>
Fin del programa
```

Ejercicio 15.- Examen febrero 2000-01, 2 puntos. El cuadrado de un número entero es igual a la suma de tantos números impares consecutivos (desde la unidad) como unidades tiene el número. Es decir, 3^2 es igual a 1+3+5 y $(-5)^2$ es igual a 1+3+5+7+9.

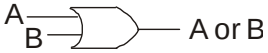
Implementa un programa que, de forma iterativa, haga lo siguiente (en cada iteración): solicite un número entero y muestre por pantalla su cuadrado calculado utilizando el algoritmo indicado. El programa deberá finalizar cuando se introduzca el valor 0.

Ejercicio 16.- Modelado de circuitos lógicos. Las expresiones booleanas pueden utilizarse para modelar circuitos lógicos. Las puertas AND, OR y NOT (también conocidas estas últimas como *inversores*) son componentes electrónicos de dichos circuitos. Las entradas a estas puertas son pulsos de corriente aplicados a las líneas de entrada de las puertas. Las salidas son pulsos de corriente sobre las líneas que salen de las puertas.

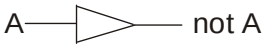
En el caso de una puerta AND, se produce un pulso de salida cuando hay pulsos en las dos líneas de entrada (A y B). Si interpretamos la existencia de pulso en una línea como TRUE y la ausencia de pulso como FALSE, el valor de la salida de una puerta AND sería equivalente al correspondiente del operador booleano *and* con argumentos A y B:



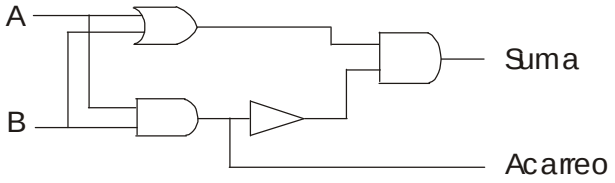
Por su parte, en una puerta OR basta con que haya pulso en una de sus líneas de entrada para que se produzca un pulso de salida. En consecuencia:



Una puerta NOT produce pulso de salida sólo cuando hay pulso de entrada. De ahí que:



Teniendo en cuenta lo anterior, escribe un programa en Pascal que implemente el comportamiento de un semi-sumador binario



El programa deberá solicitar al usuario que introduzca por teclado los valores de las líneas de entrada, A y B, y controlar que dichos valores sean válidos (dígitos binarios). Si se introducen valores no válidos, el programa deberá continuar pidiendo los datos hasta que lo sean.

Ejercicio 17.- Simulaciones que implican aleatoriedad. Una simulación consiste en obtener un modelo de un proceso dinámico y utilizar ese modelo para estudiar el comportamiento de dicho proceso. El comportamiento de procesos deterministas se puede modelar mediante una o más ecuaciones. Sin embargo, el proceso sometido a estudio puede involucrar aleatoriedad (por ejemplo, el número de objetos producidos por una máquina que tienen algún defecto, la llegada de autocares a una estación, el desalojo de las personas que hay en un edificio, etc.). Los programas que simulan este segundo tipo de procesos necesitan hacer uso de algún tipo de generador de números aleatorios.

Un generador de números aleatorios produce, de manera arbitraria, un número dentro de un rango fijo. El lanzamiento de una moneda, el lanzamiento de un dado o un giro de ruleta son mecanismos generadores de números aleatorios por excelencia.

Ahora bien, resulta poco práctico que el usuario lance una moneda al aire cada vez que un programa necesita un número aleatorio. Los programas van a hacer uso de generadores de números *pseudoaleatorios*: algoritmos deterministas que permiten obtener secuencias de números que parecen aleatorios. La mayoría de estos generadores siguen una distribución uniforme: la secuencia resultante tiende a distribuirse uniformemente a lo largo del rango considerado.

En este ejercicio vamos a abordar dos temas distintos: la generación de números pseudoaletorios y su aplicación a sencillos programas de simulación.

GENERANDO NÚMEROS PSEUDOALEATORIOS

Hay muchos algoritmos para generar números pseudoaleatorios. Uno de los más simples y usados es el *algoritmo de Congruencia lineal* propuesto por D.H. Lehmer en 1948, según el cual

$$r_k = (x \cdot r_{k-1} + y) \bmod \max$$

donde r_k es el número aleatorio generado, y *max*, *x* e *y* son constantes. Para un cierto número r_{k-1}, la ecuación anterior genera un número entero aleatorio, r_k, comprendido entre *0* y *max-1*. r_k sirve como *semilla* del siguiente número aleatorio generado dentro de la misma ejecución del programa. Es decir, cada número puede usarse con dos propósitos: como número aleatorio y como semilla para crear otro número aleatorio.

Como la secuencia se genera por medio de un algoritmo, siempre se obtiene la misma secuencia si las condiciones iniciales son iguales. Es decir, fijados *max*, *x* e *y* por el algoritmo, si el programa empieza siempre con el mismo dato (semilla del generador), se obtiene el mismo resultado una y otra vez en diferentes ejecuciones del programa. Una secuencia que siempre es igual no es satisfactoria. Necesitamos una forma de hacer que la semilla del generador (la semilla inicial) sea aleatoria. Una manera consiste en pedir al usuario que introduzca un número que actúe como semilla del generador al principio del programa. Otra forma consiste en utilizar un número que pueda ser suministrado por el sistema y que sea diferente cada vez que se ejecute el programa (por ejemplo, el tiempo marcado por el reloj de la máquina).

a) Implementa un programa que, usando el algoritmo de congruencia lineal, genere y muestre por pantalla una secuencia de números pseudoaleatorios de longitud determinada por el usuario.
La longitud de la secuencia puede ser fijada al comienzo del programa, mediante un valor proporcionado por el usuario, o se puede optar por preguntarle al usuario si desea seguir generando números cada vez que se genere uno y se muestre. Asimismo, es el usuario quien proporciona la semilla del generador al principio del programa.

b) La calidad de un generador de números pseudoaleatorios es una medida de la dificultad que existe para distinguir la secuencia obtenida de una realmente aleatoria. Los valores de las constantes del algoritmo son, en principio, arbitrarios pero ¿afectan a la calidad de los números generados? Responde a la pregunta tras analizar las secuencias generadas en las siguientes situaciones:
   - semilla = 1; x = 2; y = 3; max = 5
   - semilla = 2; x = 2; y = 3; max = 5
   - semilla = 3; x = 4; y = 2; max = 10
   - semilla = 3; x = 11; y = 7; max = 10
   - semilla = 3; x = 11; y = 7; max = 1000
   - semilla = 0; x = 11; y = 7; max = 1000

c) Los valores x = 40, y = 3641 y max = 729 trabajan razonablemente bien con implementaciones concretas en las que el mayor número entero admitido sea 32761 o mayor. Con estos valores, el generador produce 729 números antes de repetir uno. Pero no son muchos los programas que requieren un número entero elegido aleatoriamente en el rango de 0 a 728. Normalmente son rangos más pequeños.
Modifica el programa del apartado A para que se generen números comprendidos entre *0* y *num-1* (*num* es un valor entero proporcionado por el usuario). ¿Qué modificaciones habría que realizar si se desean generar números entre *1* y *num*? ¿Y si, con los valores indicados en el apartado actual para las constantes, se quieren generar números reales entre *0* y *1*?

EJEMPLOS DE SIMULACIÓN

Vamos a ilustrar el uso de los generadores de números pseudoaleatorios en el desarrollo de los programas que se proponen a continuación.

Lanzamiento de dados

Se trata de implementar un programa que sea capaz de simular el lanzamiento de dos dados y mida la frecuencia relativa de aparición de un cierto valor a lo largo de un determinado número de tiradas. El valor cuya frecuencia de aparición se desea medir y el número de tiradas a realizar son proporcionados por el usuario al principio del programa.

Sobre el programa implementado, realiza las modificaciones oportunas de manera que el usuario pueda decidir tras cada experimento (grupo de tiradas) si desea o no realizar otro.

Piedra, papel o tijeras

Este es un juego muy antiguo en el que intervienen dos personas. Cada jugador hace su elección entre las tres alternativas existentes (piedra, papel o tijeras) y el ganador se determina atendiendo a las siguientes reglas:

- La piedra gana a las tijeras (puede golpearlas hasta romperlas).
- Las tijeras ganan al papel (pueden cortarlo).
- El papel gana a la piedra (puede envolverla).
- Si las dos elecciones son la misma se produce un empate.

Escribe un programa que permita a la máquina jugar contra un humano. Máquina y humano harán su elección y la primera será la que decida quién gana.

Ejercicio 18.- Escribir un programa que calcule la suma de los pares que hay entre dos números dados como entrada.

Ejercicio 19.- Escribir un programa que muestre las tablas de multiplicar para todos los números entre dos números introducidos por el usuario. Es decir, si el usuario solicita las tablas entre 2 y 5, el programa mostrará las tablas completas del 2, el 3, el 4 y el 5.

Ejercicio 20.- Escribir un programa que dibuje un tablero de damas, enmarcado con asteriscos, utilizando para las casillas blancas el carácter cuyo código ASCII es el 219 y para las negras el espacio. Para este ejercicio sólo está permitido usar una sentencia IF-THEN-ELSE.

Ejercicio 21.- Escribir un programa que determine si un número entero positivo es primo ó no. Los números 1, 2 y 3 son primos; un número entero n mayor que 3 es primo si no tiene ningún divisor entero entre 2 y la raíz de n.

Ejercicio 22.- Escribir un programa que lea un número entero y pregunte por su raíz cuadrada tantas veces como sea necesario hasta que el usuario acierte. Deberá indicar en cada intento si hubo acierto, si se paso o si se quedó corto. Tómese alguna decisión con respecto a la precisión de la respuesta que debe dar el usuario.

Ejercicio 23.- Dado un número impar mostrar un cuadrado de tamaño n con la siguiente forma:

Si n=3	Si n=5	Si n=7
111	11111	1111111
121	12221	1222221
111	12321	1233321
	12221	1234321
	11111	1233321
		1222221
		1111111