Práctico 5 - Repetición condicional: while y repeat

Programación 1
InCo - Facultad de Ingeniería, Udelar

1. Indique qué se exhibirá en la salida estándar al ejecutar cada uno de los siguientes fragmentos de programa. Suponga que todas las variables son enteras.

   (a)
   ```
   x := 10;
   while x > 0 do
       x := x - 3;
   writeln (x)
   ```

   Salida: `-2`

   (b)
   ```
   suma := 0;
   i := 3;
   while i <= 7 do
   begin
      suma := suma + i;
      i := i + 2
   end;
   writeln (i, suma);
   ```

   Salida: `9 15`

   (c) Asuma que se ingresan los siguientes valores en la entrada estándar: 10 5 12 -5
   ```
   suma := 0;
   read (x);
   while x >= 0 do
   begin
      suma := suma + x;
      read (x)
   end;
   writeln (suma);
   ```

   Salida: 27

   (d) Asuma que se ingresan los siguientes valores en la entrada estándar: 5 6 -3 -4 7 0 5 8 9
   ```
   n := 3;
   suma := 0;
   i := 0;
   while i < n do
   begin
      read (valor);
      if valor > 0 then
   ```

```
      suma := suma + valor
   else
      i := i + 1
end;
writeln (suma, i, valor);
```

Salida: `18 3 0`

2. Indique qué se exhibirá en la salida estándar al ejecutar cada uno de los siguientes fragmentos de programa. Suponga que todas las variables son enteras.

   (a)
   ```
   a := 6;
   b := 5;
   repeat
      a := a + 1
   until a > b;
   writeln (a);
   ```

   Salida: `7`

   (b)
   ```
   i := 0;
   repeat
      writeln (i)
   until i = 0;
   ```

   Salida: `0`

   (c)
   ```
   i := 10;
   j := 5;
   repeat
      i := i - 1;
      j := j + 1;
      writeln (i, j);
   until i < j;
   ```

   Salida:
   ```
   9 6
   8 7
   7 8
   ```

   (d) Asuma que se ingresan los siguientes valores en la entrada estándar: 10 7 1 -1 7 8
   ```
   read(i, j);
   repeat
      read(x);
      i := i - x;
      j := j + x;
      writeln (i, j, x);
   until (i < j) or (x < 0);
   ```

Salida:

```
9 8 1
10 7 -1
```

3. Determine cuáles de los siguientes fragmentos de programa producirán la misma salida al ejecutarlos. Suponga que todas las variables son enteras.

(a)
```
i := 1;
j := 2;
repeat
   write (i, j);
   i := i + 1;
   j := j + 1
until j <= 3;
```

(b)
```
i := 1;
j := 1;
while (i <= 3) and (j <= 2) do
begin
   write (i, j + 1);
   i := i + 1;
   j := j + 1
end;
```

(c)
```
i := 1;
repeat
   write (i, i + 1);
   i := i + 1
until i <= 3;
```

Respuesta: Las opciones (a) y (c)

4. Suponga que se ingresa una secuencia de números enteros positivos que debe ser leída de la entrada estándar. La secuencia contiene al menos un número entero positivo y se indica su fin con el entero -1. Para los siguientes programas responda: ¿cuál es la instrucción de iteración más adecuada para utilizar (for, while, repeat)? ¿Por qué?

(a) Escriba un programa en Pascal que determine la suma y el promedio de todos los enteros positivos leídos. Exhiba ambos resultados en la salida estándar.

No es posible usar `for` ya que no conocemos la cantidad de números que serán ingresados.

Como conocemos que la secuencia tiene al menos un número, la solución con `repeat` parece ser la más apropiada. No obstante, también se puede resolver con `while`

```
program secuencia;
const
   CENTINELA = -1;
var
   numero , contador, suma : integer;
   promedio : real;
begin
   { inicializacion }
   suma:= 0;
   contador:= 0;

   { primera lectura (no centinela)}
   read(numero);
   repeat
      { acumulacion y conteo }
      suma:= suma + numero;
      contador:= contador + 1;

      {lectura siguiente (posible centinela)}
      read(numero);

   until numero = CENTINELA;

   {se calcula promedio}
   promedio:= suma / contador;


   {mostrar resultados }
   writeln('La suma es: ',  suma:6);
   writeln('El promedio es: ', promedio:8:2)
end.
```

(b) Escriba un programa en Pascal que determine el número más grande y el más pequeño de todos los enteros positivos leídos. Exhiba ambos resultados en la salida estándar.

La instrucción `for` no es apropiada por las razones que se dieron en la parte anterior.

En esta solución se comsume el primer número (que nunca es el centinela) para inicializar las variables `minimo` y `maximo`. La secuencia restante puede ser vacía. Por lo tanto se aplica una *lectura con centinela* con `while` como se vio en las notas teóricas.

```
program secuencia;
const
   CENTINELA = -1;
var
   numero, maximo, minimo   : integer;
begin
   { primera lectura, no centinela}
   read(numero);

   { inicializar maximo y minimo }
   maximo:= numero;
   minimo:= numero;

   { segunda lectura, posible centinela}
   read(numero);
   while numero <> CENTINELA do
   begin
      { actualizar max y min si corresponde}
      if numero < minimo then
         minimo:= numero
      else if numero > maximo then
         maximo:= numero;

      {lectura siguiente, posible centinela }
      read(numero)
   end;

   { mostrar resultados }
   writeln('Maximo y minimo son : ',  maximo:6, ' y ', minimo:6);
end.
```

5. Se desea implementar un programa que calcule el saldo final de una cuenta. Suponga que los datos son leídos de la entrada estándar y que constan de renglones. El primer renglón contiene el saldo inicial de la cuenta. Los siguientes renglones contienen una letra y un valor real para indicar las transacciones (posiblemente ninguna). La letra puede ser la D para efectuar un depósito o la R para efectuar un retiro. El último renglón contiene únicamente la letra X. Escriba un programa en Pascal que determine el saldo exacto de la cuenta después de procesar las transacciones. Incluya mensajes de salida con etiquetas descriptivas para exhibir los valores.

Ejemplos

```
1200.35
D 64.12
R 390.00
R 289.67
D 13.02
R 51.07
X
El saldo final es 546.75
```

```
600.50
X
El saldo final es 600.5
```

```
program saldocuenta;
const
   FIN    = 'X';
   DEPOSITO = 'D';
   RETIRO = 'R';
var
   tipo : char;
   monto, saldo : real;
begin
   { ingresar saldo inicial }
   readln(saldo);

   { leer tipo del primer movimiento }
   read(tipo);
   while tipo <> FIN do
   begin
      readln(monto);
      { se procesa el movimiento }
      case tipo of
        DEPOSITO  : saldo := saldo + monto;
        RETIRO    : saldo := saldo - monto;
      end;
      read(tipo)
   end;
   readln; { Se consume ultimo fin de linea }

   { mostrar resultado final }
   writeln('El saldo final es: ', saldo:8:2)
end.
```

6. Dado un fragmento de texto que debe ser leído de la entrada estándar, todo en una línea, y terminado por el caracter $ (centinela), escriba un programa en Pascal que determine y exhiba las consonantes y vocales que aparecen duplicadas en forma contigua. Asuma que todas las letras ingresadas son minúsculas. Incluya mensajes de salida con etiquetas descriptivas para solicitar y exhibir los valores.

Ejemplo

```
Ingrese un texto: llama al chico que lee$
Las consonantes y vocales duplicadas son: ll ee
```

```
program duplicadas;
const
  FIN = '$';
var
  anterior, siguiente : char;
begin
   {etiqueta asociada a la entrada}
   write('Ingrese un texto: ');

   { se lee primer caracter antes de la iteración}
   read(anterior);

   { se exhibe etiqueta asociada a la salida}
   write('Las consonantes y vocales duplicadas son: ');

   while anterior <> FIN do
   begin
      read(siguiente);
      if ('a' <= anterior) and (anterior <= 'z')  {letra minúscula}
         and
         (anterior = siguiente)   {repetida}
      then
         {Se exhibe caracter duplicado}
         write(anterior, anterior, ' ')
      else
         anterior := siguiente  { Se actualiza anterior}
   end;
   writeln; {cambio de linea final}
end.
```

7. Escriba un programa en Pascal que determine si un número n es primo o no, siendo n un entero positivo leído de la entrada estándar. Exhiba un mensaje de salida indicando el resultado. Incluya mensajes de salida con etiquetas descriptivas para solicitar los valores.

Ejemplo

```
Ingrese un entero positivo: 3
Es primo
```

```
program NumeroPrimo;
var
   fin,numero, divisor : integer;
begin
   write('Ingrese un entero positivo: ');
   readln(numero);
   if numero < 2 then
      { caso 1 y 0: no son primos }
      writeln('No es primo')
   else { numero > 2 }
   begin
      fin:= trunc(sqrt(numero));
      divisor:= 2;
      { busco divisores desde 2 hasta la raíz del numero }
      while (divisor <= fin) and (numero mod divisor <> 0) do
         divisor:= divisor + 1;

      if divisor <= fin then
         { encontré un divisor }
         writeln('No es primo')
      else
         { no encontré divisor }
         writeln('Es primo')
   end
end.
```

8. Se desea implementar un programa que realice las funciones de una calculadora simple. Los datos de entrada son una secuencia de enteros sin signo y los operadores +, *, / y -, seguida de un signo =. Cada entero de la entrada está seguido por un operador salvo el último que está seguido por el símbolo =. Los operadores se aplican en el orden en que aparecen sin importar la precedencia. Si bien se ingresa el operador de la división con el símbolo /, el comportamiento debe ser el de DIV. Asuma que se ingresa al menos un número.

   (a) Escriba un programa en Pascal que resuelva lo pedido, suponiendo que no hay espacios en la entrada.

Ejemplo
4+3/2*8-4= 20

```
program calculadora;
VAR
  resultado, operando : integer;
  operador : char;
begin
  (* Se lee el primer entero *)
   read(resultado); {primer numero}
   read(operador);  {primer operador}
   while operador <> '=' do
   begin
      { entra a la iteración con el operador ya leido}
      read(operando);
      case operador of
        '+' : resultado := resultado + operando;
        '-' : resultado := resultado - operando;
        '*' : resultado := resultado * operando;
        '/' : resultado := resultado div operando;
      end;
      { siguiente operador}
      read(operador);
   end;
  writeln(resultado);
end.
```

(b) Escriba un programa en Pascal que resuelva lo pedido, suponiendo que cada entero está separado del símbolo que lo sigue por 0 o más espacios.

Ejemplo

```
4    + 3/   2 *       8-    4 =
20
```

Observación: La lectura de un entero involucra un *salteo de espacios* que la operación `read` ejecuta automáticamente. Por lo tanto, no es necesario consumir espacios previo a la lectura de un entero.

```
program calculadora;
const
   espacio = ' ';
var
  resultado, operando : integer;
  operador : char;
begin
   { Se lee el primer entero }
   read(resultado); {primer numero}

   { leer operador salteando espacios }
   repeat
      read(operador)
   until operador <> espacio;

   while operador <> '=' do
   begin      { entra a la iteración con el operador ya leido }

      { este read consume eventuales espacios previos }
      read(operando);

      { se procesa la operacion }
      case operador of
        '+' : resultado := resultado + operando;
        '-' : resultado := resultado - operando;
        '*' : resultado := resultado * operando;
        '/' : resultado := resultado div operando;
      end;

      { siguiente operador }
      repeat
         read(operador)
      until operador <> espacio;
   end;
  writeln(resultado);
end.
```

9. Escriba un programa en Pascal que determine y exhiba la desviación estándar de n números reales positivos. Los números deben ser leídos de la entrada estándar, donde se ingresará un número negativo al final como centinela. La desviación estándar de un conjunto de números $x_1, x_2, ..., x_n$ se define como la raíz cuadrada de la expresión $s/n - a^2$ donde a es el promedio de los valores x_i $((x_1 + x_2 + ... + x_n)/n)$ y s es la suma de los cuadrados de los valores x_i $({x_1}^2 + {x_2}^2 + ... + {x_n}^2)$. Incluya mensajes de salida con etiquetas descriptivas para solicitar y/o exhibir los valores.

Ejemplo

```
25.0 23.0 22.0 21.0 17.0 9.0 6.0 5.0 -1.0
La desviación estándar es 7.60
```

```
program DesviacionEstandar;
var
   sumaCuadrados, suma, numero, resultado : real;
   cantidad : integer;
begin
   { inicializar sumas }
   sumaCuadrados:= 0;
   suma:= 0;
   cantidad:= 0;

   { lectura inicial }
   read(numero);
   repeat
      { acumular  sumas }
      suma:= suma + numero;
      sumaCuadrados:= sumaCuadrados + sqr(numero);
      cantidad:= cantidad + 1;

      { siguiente número }
      read(numero)
   until numero < 0;

   { calculo final }
   resultado:= sqrt(sumaCuadrados / cantidad - sqr(suma / cantidad));

   { mostrar resultado}
   writeln('La desviacion estandar es: ', resultado:6:2)
end.
```

10. Todo número natural positivo num tiene una descomposición única de la forma $num = 2^n \times val$, donde val es un número natural impar y $n >= 0$. Escriba un programa en Pascal que lea de la entrada estándar un entero positivo num, calcule y exhiba los correspondientes valores de val y n. Incluya mensajes de salida con etiquetas descriptivas para solicitar los valores.

Ejemplos
`Ingrese un número natural positivo: 12` `12 = 2^2 * 3`
`Ingrese un número natural positivo: 36` `36 = 2^2 * 9`
`Ingrese un número natural positivo: 7` `7 = 2^0 * 7`
`Ingrese un número natural positivo: 8` `8 = 2^3 * 1`

```
program factorizacion2;
var
  num, n, k: integer;
begin
  { Lectura del numero }
  write('Ingrese un natural positivo: ');
  readln(num);
  { Guardo el valor original }
  k := num;
  { Inicializo exponente }
  n := 0;
  { Calculo factores }
  while num mod 2 = 0 do
  begin
     num := num div 2;
     n := n + 1
  end;
  { mostrar resultado }
  writeln(k:3, ' = 2^', n:1, ' * ', num:3)
end.
```

11. Escriba un programa en Pascal que lea de la entrada estándar dos enteros n y b, calcule y exhiba la parte entera del logaritmo de n en base b. Dicho resultado es un entero k que cumple lo siguiente: $b^k <= n < b^{k+1}$. El algoritmo solo puede realizar divisiones y sumas. Asuma que los valores ingresados cumplen que $n > 0$ y $b > 1$. Incluya mensajes de salida con etiquetas descriptivas para solicitar y/o exhibir los valores.

Ejemplos
Ingrese numero: 1024 Ingrese base: 2 Resultado: 10
Ingrese numero: 1020 Ingrese base: 2 Resultado: 9
Ingrese numero: 8 Ingrese base: 111 Resultado: 0
Ingrese numero: 300 Ingrese base: 5 Resultado: 3

```
program logaritmo;
var
   numero, base,
   resto, contador : integer;
begin
   { ingresar datos }
   write('Ingrese numero: ');
   readln(numero);
   write('Ingrese base: ');
   readln(base);

  { inicializar }
   resto:= numero;
   contador:= 0;

   { divido hasta tener un resto menor a la base }
   while resto >=  base do
   begin
      resto:= resto div base;
      contador:= contador + 1
   end;

   { la cantidad de divisiones es el resultado }
   { despliego resultados }
   writeln('Resultado: ', contador)
end.
```