# Práctico 6 - Iteración - Ejemplos avanzados

Programación 1
InCo - Facultad de Ingeniería, Udelar

1. Escriba un programa que calcule el MCD (Máximo Común Divisor) para dos naturales $m$ y $n$ con $m \neq 0$. ¿Qué estructura de control se debe utilizar y por qué?

   Se recuerda que el máximo común divisor se define como:

   $$\text{MCD}(m, n) = \text{máx}\{k : k \text{ es divisor de } m \text{ y } k \text{ es divisor de } n\}$$

   Se sugiere aplicar el *algoritmo de Euclides* que se basa en las siguientes igualdades:

   $$\begin{aligned} \text{MCD}(m, 0) &= m \\ \text{MCD}(m, n) &= \text{MCD}(n, m \bmod n) \end{aligned}$$

   Ejemplos:

   ```
   Ingrese m y n : 42 30
   El MCD es:  6
   ```

   ```
   Ingrese m y n : 132 168
   El MCD es: 12
   ```

   No es posible usar for ya que no conocemos a priori la cantidad de iteraciones. Dada la formulada presentada, se calcula el MCD hasta que $n$ sea 0. La estructura de control adecuada es `while` dado que puede no requerir paso iterativo si al leer $m$ y $n$, ya estamos en el paso base ($n$=0).

   ```
   program Pr6Ej1;
   var
     m,n,aux : integer;
   begin

     write ('Ingrese m y n: ');
     read (m,n);

     while (n <> 0) do
     begin
       aux := m;
       m := n;
       n := aux mod n;
     end;

     writeln ('El MCD es ',m);

   end.
   ```

2. (a) Escriba un programa que lea una letra y una oración y despliegue la cantidad de palabras de la oración que terminan con la letra dada. Se asume que la oración es una secuencia de palabras separadas solo por espacios, y que terminan con el carácter punto (`.`). La oración siempre tiene al menos una palabra.

Ejemplos:

```
Letra: s
Oración: Las cualidades indispensables para un buen cocinero no se conocen.
La oración tiene 3 palabras que terminan con s
```

```
Letra: r
Oración: Si no le pides nada que no pueda hacer jamas hara lo que puede.
La oración tiene 1 palabras que terminan con r
```

```
Letra: o
Oración: No basta tener buen ingenio lo principal es aplicarlo bien.
La oración tiene 4 palabras que terminan con o
```

```
program Pr6Ej2a;
const
   FIN = '.';
   ESPACIO = ' ';
var
   letra, car, car_ant : char;
   cant : integer;
begin
   write ('Letra: ');
   readln (letra);
   write ('Oración: ');

   cant := 0;

   {avanzar hasta comienzo de primera palabra}
   repeat
      read(car)
   until car <> ESPACIO;

   repeat
      {avanzar hasta terminar la palabra}
      repeat
         car_ant := car;
         read(car);
      until (car = ESPACIO) or (car = FIN);

      {control de fin de palabra}
      if (car_ant = letra) then
         cant := cant + 1;

      {avanzar hasta comienzo siguiente palabra}
      while car = ESPACIO do
         read(car);

   until car = FIN;

   writeln ('La oración tiene ', cant, ' palabras que terminan con ', letra);
end.
```

(b) Escriba un programa que despliegue la cantidad de palabras que **comienzan** con la letra leida

```
program Pr6Ej2a;
const
   FIN = '.';
   ESPACIO = ' ';
var
   letra, car : char;
   cant : integer;
begin
   write ('Letra: ');
   readln (letra);
   write ('Oración: ');

   cant := 0;

   {avanzar hasta comienzo de primera palabra}
   repeat
      read(car)
   until car <> ESPACIO;

   repeat
      {control letra al comienzo }
      if car = letra then
         cant:= cant + 1;

      {avanzar hasta terminar la palabra}
      repeat
         read(car);
      until (car = ESPACIO) or (car = FIN);

      {avanzar hasta comienzo siguiente palabra}
      while car = ESPACIO do
         read(car);

   until car = FIN;

   writeln ('La oración tiene ', cant, ' palabras que comienzan con ', letra);
end.
```

(c) Escriba un programa que despliegue la cantidad de palabras que **contienen la letra una sola vez**.

```
program Pr6Ej2a;
const
   FIN = '.';
   ESPACIO = ' ';
var
   letra, car : char;
   contLetra , contPalabra : integer;
begin
   write ('Letra: ');
   readln (letra);
   write ('Oración: ');

   contPalabra := 0;

   {avanzar hasta comienzo de primera palabra}
   repeat
      read(car)
   until car <> ESPACIO;

   repeat
      { comienza palabra  }
      contLetra:= 0;

      { avanzar hasta terminar la palabra}
      repeat
         if car = letra then
            contLetra:= contLetra + 1;
         read(car);
      until (car = ESPACIO) or (car = FIN);

      {actualizo contador de palabras}
      if contLetra = 1 then
         contPalabra:= contPalabra + 1;

      {avanzar hasta comienzo siguiente palabra}
      while car = ESPACIO do
         read(car);

   until car = FIN;

   writeln ('La oración tiene ', contPalabra, ' palabras que tienen ', letra, ' una sola vez');
end.
```

3. Escriba un programa que evalúe un polinomio para un valor dado.

   El usuario ingresa los siguientes datos:

   - un número real $v$
   - una secuencia de enteros no negativos $a_n, a_{n-1}, \ldots, a_1, a_0$ que son los coeficientes del polinomio. El final de la secuencia está indicado por el centinela `-1`. Siempre se ingresará al menos **un número** antes del centinela.

   El polinomio en cuestión es:

$$\mathcal{P}(x) := a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

   y el programa debe desplegar en la salida el valor $\mathcal{P}(v)$.

   Para calcular el resultado se sugiere utilizar la regla de *Ruffini-Horner*:

$$\mathcal{P}(v) = (\cdots((a_n v + a_{n-1})v + a_{n-2})v + \cdots + a_1)v + a_0$$

   ¿Qué estructuras de control corresponde utilizar para resolver el problema? Explique su respuesta.

Ejemplo:

```
Ingrese x: 3
Ingrese los coeficientes: 2 5 4 -1
El valor del polinomio evaluado en  3.00 es 37.00
```

No es posible usar for ya que no conocemos a priori la cantidad de coeficientes que el usuario ingresará.

La solución que se presenta inicializa el acumulador con el primer coeficiente y luego itera con `while` por el resto de los coeficientes.

Otra solución correcta puede inicializar el acumulador en 0 e iterar con `repeat` por todos los coeficientes.

```
program Pr6Ej3;
const
   CENTINELA = -1;
var
  x, total : real;
  coef : integer;
begin

  write ('Ingrese x: '); readln(x);
  write ('Ingrese los coeficientes: ');

  { se inicializa total con el primer coeficiente }
  read (total);

  {segunda lectura (posible centinela)}
  read(coef);
  while coef <> CENTINELA do
  begin
      {acumulacion segun Horner}
      total := total * x + coef;

      {siguiente lectura (posible centinela)}
      read(coef);
  end;

  {mostrar resultado}
  writeln ('El valor del polinomio evaluado en ', x:5:2, ' es ', total:5:2);
end.
```

4. Escriba un programa que lea un entero positivo $n$ y despliegue la raíz cuadrada de los $n$ primeros naturales primos.

Ejemplo:

```
Ingrese n: 4
1: Primo: 2 Raiz Cuadrada:  1.41
2: Primo: 3 Raiz Cuadrada:  1.73
3: Primo: 5 Raiz Cuadrada:  2.24
4: Primo: 7 Raiz Cuadrada:  2.65
```

(a) Escriba una solución que utilice *iteración condicional* (*while* o *repeat*).

```
program Pr6Ej4a;
var
  cant_primos, num, fin, divisor, iter : integer;
begin

  write ('Ingrese n: '); readln (cant_primos);

  num := 0;
  iter := 0;

  repeat
    num := num + 1;
    divisor := 2;
    fin := trunc(sqrt(num));
    while (divisor <= fin) and (num mod divisor <> 0) do
      divisor := divisor + 1;
    if divisor > fin then begin
      writeln(iter,': Primo: ', num, ' Raiz Cuadrada: ', sqrt(num):5:2);
      iter := iter + 1;
    end;
  until iter = cant_primos;
end.
```

(b) Escriba una solución que utilice una repetición *for*.

```
program Pr6Ej4b;
var
  cant_primos, num, fin, divisor, iter : integer;
begin

  write ('Ingrese n: '); readln (cant_primos);

  num := 0;
  for iter:= 1 to cant_primos do
  begin
    {itero hasta encontrar un primo}
    repeat
      num := num + 1;
      divisor := 2;
      fin := trunc(sqrt(num));
      while (divisor <= fin) and (num mod divisor <> 0) do
        divisor := divisor + 1;
    until divisor > fin;
    writeln(iter,': Primo: ', num, ' Raiz Cuadrada: ', sqrt(num):5:2);
  end;
end.
```

5. Escriba un programa que lea un entero positivo `K` y despliegue una secuencia de cuadrados de enteros positivos consecutivos. La secuencia comienza con el entero 1 y termina cuando la diferencia entre un cuadrado y el anterior sea mayor que `K`.

Ejemplo:

```
Ingrese k: 40
Secuencia de cuadrados:
1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361 400 441
```

(a) Escriba una solución que utilice iteración condicional (*while* o *repeat*).

```
program Pr6Ej5a;
var
  k, cuad1, cuad2,num : integer;
begin

  write ('Ingrese k: '); readln (k);

  num := 1;
  cuad2 := sqr(num);
  writeln ('Secuencia de cuadrados:');
  write (cuad2);
  repeat
    cuad1 := cuad2;
    num := num +1;
    cuad2:= sqr(num);
    write (' ', cuad2);
  until (cuad2- cuad1) > k;
end.
```

(b) Escriba una solución que utilice una iteración *for*. **Sugerencia:** note que la diferencia entre cuadrados consecutivos es siempre un número impar. A partir de esta observación, intente predecir el largo de la secuencia en función de `K`.

```
program Pr6Ej5b;
var
  k, iter : integer;
begin

  write ('Ingrese k: '); readln (k);
  writeln ('Secuencia de cuadrados:');
  for iter := 1 to (k+3) DIV 2 do
    write (' ', sqr(iter));
end.
```

6. La compañía Riky Gamusino:

   - vende actualmente 20 gamusinos al mes
   - obtiene un beneficio de $30 por gamusino.
   - gasta una cierta suma al mes en publicidad
   - tiene costos de funcionamiento de $100 al mes que no dependen del volumen de ventas.
   - si dobla la cantidad gastada en publicidad, las ventas se incrementan en un 50 por ciento.

   Escriba un programa que lea el gasto inicial en publicidad y despliegue una tabla que determine las ventas y el beneficio neto en función de tal gasto. La tabla comienza con el gasto inicial en publicidad leido de la entrada. Cada nueva fila dobla la cantidad gastada en publicidad. La tabla finaliza cuando el beneficio neto empieza a declinar. La salida debe incluir las cantidades hasta la primera vez que el beneficio neto disminuye.

Ejemplo:

```
Ingrese el gasto inicial en publicidad: 200
Gastado Publicidad   Ventas   Beneficio Neto
      200              20         300
      400              30         400
      800              45         450
     1600              67         310
```

```
program Pr6Ej6;
const
  COSTO_FUNC = 100;
  BENEF_GAM = 30;
  VENTAS_MES = 20;
var
  ventas, benef_neto_actual, benef_neto_ant, publ : integer;

begin

  write ('Ingrese el gasto inicial en publicidad: '); readln(publ);

  writeln('Gastado Publicidad   Ventas   Beneficio Neto');

  ventas := VENTAS_MES;

  benef_neto_actual :=  ventas*BENEF_GAM - COSTO_FUNC - publ;
  writeln(publ,'   ', ventas,'   ',benef_neto_actual);

  repeat
    benef_neto_ant := benef_neto_actual;
    publ := publ * 2;
    ventas := trunc(ventas * 1.50);
    benef_neto_actual :=  ventas*BENEF_GAM - COSTO_FUNC - publ;
    writeln(publ,'   ', ventas,'   ',benef_neto_actual);
  until benef_neto_ant > benef_neto_actual;

end.
```

7. Escriba un programa que lea dos enteros positivos $m$ y $n$ (entre 1 y 10), y exhiba para cada entero en el rango su correspondiente tabla de multiplicar. En caso de que $m$ sea mayor que $n$ no se exhibirá nada. ¿Qué estructuras de control corresponde utilizar para resolver el problema? Explique su respuesta.

Ejemplo:

```
Ingrese m: 5
Ingrese n: 7
>>> Tabla 5 <<<
1*5 = 5
2*5 = 10
...
10*5 = 50
>>> Tabla 6 <<<
1*6 = 6
...
10*6 = 60
>>> Tabla 7 <<<
1*7 = 7
...
10*7 = 70
```

Dado que se conoce a priori la cantidad de iteraciones, la estructura de control adecuada es `for`.

```
program Pr6Ej7;
var
  n, m, tabla, num : integer;
begin
  write('Ingrese m: '); readln(m);
  write('Ingrese n: '); readln(n);

  for tabla := m to n do
  begin
    writeln('>>> Tabla ', tabla,' <<<');
    for num:=1 to 10 do
      writeln(num,'*',tabla,' = ', num*tabla);
    writeln();
  end;

end.
```

8. Escriba un programa que lea dos enteros positivos $m$ y $n$, y exhiba todos los números primos en el rango. En caso de que $m$ sea mayor que $n$, no se exhibirá ninguno.

   ¿Qué estructuras de control corresponde utilizar para resolver el problema? Explique su respuesta.

   Ejemplo:

   ```
   Ingrese dos enteros positivos: 10 31
   Los numeros primos entre 10 y 31 son: 11 13 17 19 23 29 31
   ```

La solución usa dos estructuras de control. La primera es un `for` dado que se recorrerá los números entre $m$ y $n$. La segunda es un `while`, dado que se determinará si el número que estoy analizando es primo o no. La iteración para cuando no hay más divisores o encuentro un divisor que divide al número en cuestión.

```
program Pr6Ej8;
var
  n, m, fin : integer;
  num, divisor : integer;
begin

  write('Ingrese dos numeros enteros positivos: '); readln(m, n);

  write('Los numeros primos entre ', m, ' y ', n, ' son: ');
  for num := m to n do
  begin

    divisor := 2;
    fin := trunc(sqrt(num));
    while (divisor <= fin) and (num mod divisor <> 0) do
      divisor := divisor + 1;
    if divisor > fin then
      write(num, ' ');
  end;

end.
```

9. Escriba un programa que lea dos enteros positivos $m$ y $n$ y exhiba todos los *primos gemelos* en el rango. Los *primos gemelos* son parejas de números primos con una diferencia entre sí de exactamente 2 unidades. En caso de que $m$ sea mayor que $n$, no se exhibirá salida alguna.

   ¿Qué estructuras de control corresponde utilizar para resolver el problema? Explique su respuesta.

Ejemplo:

```
Ingrese dos enteros positivos: 10 31
Los primos gemelos entre 10 y 31 son:
11 y 13
17 y 19
29 y 31
```

La solución usa dos estructuras de control. La primera es un `for` dado que se recorrerá los números entre $m$ y $n$. La segunda es un `while`, dado que se determinará si el número que estoy analizando es primo o no. La iteración para cuando no hay más divisores o encuentro un divisor que divide al número en cuestión.

```
program Pr6Ej9;
var
  n, m, fin : integer;
  num, divisor, primo_anterior : integer;
begin

  write('Ingrese dos numeros enteros positivos: ');readln(m, n);

  primo_anterior := 2;
  writeln('Los numeros primos entre ', m, ' y ', n, ' son: ');
  for num := m to n do
  begin
    divisor := 2;
    fin := trunc(sqrt(num));
    while (divisor <= fin) and (num mod divisor <> 0) do
      divisor := divisor + 1;
    if divisor > fin then
    begin
      if (num - primo_anterior = 2) then
        writeln(primo_anterior, ' y ',num);
      primo_anterior := num;
    end;
  end;

end.
```