

Repetición Condicional

InCo - FING

Programación 1

Section 1

La instrucción while

Ejemplo

Leer números naturales de la entrada y calcular su suma. El fin de la entrada está indicado por un número negativo

```
var
  numero,suma : integer;
begin
  (* lectura inicial *)
  readln(numero);

  (* inicializacion *)
  suma:= 0;
```

Ejemplo

Leer números naturales de la entrada y calcular su suma. El fin de la entrada está indicado por un número negativo

```
var
  numero,suma : integer;
begin
  (lectura inicial *)
  readln(numero);

  (inicializacion *)
  suma:= 0;

  while numero >= 0 do
  begin
    (acumulacion *)
    suma:= suma + numero;
    (siguiente lectura *)
    readln(numero)
  end;

  (mostrar resultado *)
  writeln('La suma de los numeros es: ', suma)
end.
```

Sintaxis de la instrucción `while`

`while` e `do` instrucción

Donde:

- `e` es una expresión de tipo boolean
- *instrucción* es cualquier instrucción.
- `while` y `do` son palabras reservadas.

La ejecución de una instrucción:

while e **do** *instr*

procede de la siguiente manera:

- 1 Evaluar la expresión (la condición del `while`)
Si resulta `false`, terminar.
- 2 Ejecutar la instrucción `instr`
Volver al paso 1.

Indentación

```
(* correcto *)  
while condición do  
begin  
    hacer_algo;  
    hacer_otra_cosa  
end
```

```
(* incorrecto *)  
while condición do  
begin  
hacer_algo;  
hacer_otra_cosa  
end
```

```
(* incorrecto *)  
while condición do  
    begin  
        hacer_algo;  
        hacer_otra_cosa  
    end  
end
```

Lectura con centinela

Un **centinela** es un valor especial usado para indicar el final de una lista de datos.

El esquema de la lectura con centinela es como sigue:

```
(* lectura adelantada *)
readln(dato);

{inicialización de acumuladores }
...
(* iteración *)
while "dato no es centinela" do
begin
  (* procesar dato *)
  ...
  (* acumular *)
  ...
  (* siguiente lectura *)
  readln(dato)
end
```

Notar que el centinela no es tratado.

Determinar si un número es primo

- Un número natural es **primo** si es mayor que uno y es divisible únicamente entre sí mismo y la unidad.
- Ejemplos de números primos son: 2, 5, 11, 29, 31, 47.
- Un procedimiento para determinar si un número n es primo es recorrer todo el intervalo entre 2 y $n-1$ buscando si hay divisores de n .

Determinar si un número es primo

- Un número natural es **primo** si es mayor que uno y es divisible únicamente entre sí mismo y la unidad.
- Ejemplos de números primos son: 2, 5, 11, 29, 31, 47.
- Un procedimiento para determinar si un número n es primo es recorrer todo el intervalo entre 2 y $n-1$ buscando si hay divisores de n .
- El procedimiento anterior no es eficiente. Una primera forma de mejorarlo es buscar únicamente en el intervalo entre 2 y $n \div 2$ ya que luego de $n \div 2$ no hay divisores de n .

Determinar si un número es primo

- Un número natural es **primo** si es mayor que uno y es divisible únicamente entre sí mismo y la unidad.
- Ejemplos de números primos son: 2, 5, 11, 29, 31, 47.
- Un procedimiento para determinar si un número n es primo es recorrer todo el intervalo entre 2 y $n-1$ buscando si hay divisores de n .
- El procedimiento anterior no es eficiente. Una primera forma de mejorarlo es buscar únicamente en el intervalo entre 2 y $n \div 2$ ya que luego de $n \div 2$ no hay divisores de n .
- La búsqueda se puede hacer en forma más eficiente como se muestra en el siguiente programa.

Determinar si un número es primo

La solución utilizando for la consideramos **incorrecta**.

```
var
  fin,numero, divisor : integer;
begin
  readln(numero);
  if numero < 2 then
    writeln('El numero ',numero,' no es primo')
  else
    begin
      fin:= trunc(sqrt(numero));
      divisor:= 2;
      while (divisor <= fin) and (numero mod divisor <> 0) do
        divisor:= divisor + 1;

      if divisor <= fin then
        writeln('El numero ',numero,' no es primo')
      else
        writeln('El numero ',numero,' es primo')
      end
    end
end.
```

Factorial de un número

Todo problema resuelto con for se puede resolver con un while.

```
var
  factorial,i,numero : integer;
begin
  readln(numero);
  factorial:= 1;
  i:= 2;
  while i<=numero do
  begin
    factorial:= factorial * i;
    i:= i+1
  end;

  writeln('El factorial es: ', factorial)
end.
```

Esto se considera una mala práctica de programación, la solución apropiada es con for.

División entera por restas

Se puede calcular la división entera solo usando restas:

```
var
    dividendo, resto,
    divisor, cociente : integer;
begin
    readln(dividendo,divisor);
    resto:= dividendo;
    cociente:= 0;
    while resto >= divisor do
        begin
            resto:= resto - divisor;
            cociente:= cociente + 1
        end;
    writeln('El resto es: ', resto);
    writeln('El cociente es: ', cociente)
end.
```

Banderas booleanas

Leer números de la entrada y terminar cuando aparecen 2 números consecutivos iguales o cuando aparece un negativo.

Calcular la cantidad de pares.

```
var
  numero,cantidadPares,anterior : integer;
  repetidos                      : boolean;
begin
  readln(numero);
  repetidos:= false;
  cantidadPares:= 0;
  while not repetidos and (numero > 0) do
  begin
    if numero mod 2 = 0 then
      cantidadPares:= cantidadPares + 1;
      anterior:= numero;
      readln(numero);
      repetidos:= numero = anterior
    end;

    writeln('La cantidad es: ', cantidadPares)
  end.
```

Condición compuesta

Advinar un número en menos de 10 intentos:

```
const
  NUMERO = 39; (* algun valor *)
var
  intentos, numLeido : integer;
begin
  Write('Numero: ');
  readln(numLeido);
  intentos:= 1;
  while (numLeido <> NUMERO) and (intentos < 10) do
  begin
    writeln('Incorrecto. ', intentos:2, ' intentos');

    Write('Numero: ');
    readln(numLeido);
    intentos:= intentos + 1
  end;

  if numLeido = NUMERO then
    writeln('Correcto. ', intentos:2, ' intentos')
  else
    writeln('Incorrecto. No quedan mas intentos')
  end.
```

Section 2

La instrucción repeat

Sintaxis de la instrucción `repeat`.

```
repeat instrucción { ; instrucción } until expresión
```

Donde

- *expresión* es una expresión de tipo boolean
- El cuerpo de un `repeat` puede ser una secuencia de instrucciones, sin necesidad de delimitarlas por un `begin` y un `end`.
- `repeat` y `until` son palabras reservadas.

Semántica del repeat.

La ejecución de una instrucción:

```
repeat
  instrucción;
  ...
  instrucción
until expresión
```

procede de la siguiente manera:

- 1 Se ejecuta el cuerpo (siempre al menos una vez)
- 2 Se evalúa la expresión. Si es `false` se vuelve al paso 1. Si es `true`, termina.

Relación con `while`

La instrucción:

```
repeat  
  S  
until C
```

es equivalente a:

```
S;  
while not C do  
begin  
  S  
end
```

Procesamiento de palabras

- Deseamos contar las palabras contenidas en un texto.
- Las palabras están separadas por espacios.
- El final del texto se señala con un carácter de punto '.'
- El punto está precedido por al menos un espacio.

Esquema del procesamiento

```
const
    ESPACIO = ' ';
    FIN      = '.';
var
    caracter      : Char;
    cantidadPalabras : Integer;

begin
    cantidadPalabras:= 0;
    [avanzar comienzo de palabra]
    while [quedan palabras] do
    begin
        [leer-palabra]
        cantidadPalabras:= cantidadPalabras + 1

        [avanzar siguiente palabra]
    end; {while}

    writeln('La cantidad de palabras es: ',
            cantidadPalabras)
end.
```

Versión final

```
(* inicializar contadores *)
cantidadPalabras:= 0;
(* saltar espacios *)
repeat
  read(caracter)
until caracter <> ESPACIO;

while caracter <> FIN do
begin
  (* leer palabra *)
  repeat
    read(caracter);
  until caracter = ESPACIO;
  (* incrementar contador *)
  cantidadPalabras:= cantidadPalabras + 1;

  (* avanzar siguiente palabra *)
  repeat
    read(caracter)
  until caracter <> ESPACIO
end; {while}
```