

Capítulo 3

Estructura de Control Iterativa, de Repetición o Ciclos

1. Ciclo Incondicional For
2. Ciclo Condicional While
3. Ciclo Condicional Repeat
4. Ejemplos (While – Repeat - For)
5. Anidación de ciclos.

Estructura de Control Iterativa, de Repetición o Ciclos

Se utilizan para ejecutar sentencias en forma repetitiva.

Si el número de repeticiones se conoce de antemano (ya sea una cantidad fija o variable) se utiliza For.

En caso contrario se debe implementar con las estructuras While o Repeat.

Es común que algunas de las sentencias que se ejecutan dentro del ciclo cuenten ocurrencias (de las repeticiones del ciclo o de determinados eventos) o acumulen sumas (o productos).

- Un contador es una variable numérica entera que se incrementa (decrementa) en uno.
Cont:= Cont + 1;
- Un acumulador es una variable numérica que se incrementa (decrementa) en una cantidad variable.
Acum:=Acum + X;

1. Ciclo Incondicional For

La característica de este ciclo es que se ejecuta una cantidad “conocida” de veces, el control de las repeticiones es asumido por la propia estructura a través de la variable de control de ciclo (VC) que toma valores ordinales, entre un límite inicial y otro final (VI y VF son expresiones ordinales). Dichos valores pueden ser ascendentes o descendentes (incremento 1 o -1)

Según sea la secuencia ascendente o descendente

3.a FOR ascendente (For – to ; VI<=VF)

```
For VC := VI to VF do
  Sentencia;
```

VC: variable de control.
VI: valor inicial
VF: valor final
Sentencia simple o compuesta

La variable VC toma el valor VI y se incrementa en 1 en cada iteración hasta alcanzar el valor VF, Previo a cada iteración se verifica que se cumpla $VC \leq VF$. Estas acciones las realiza la estructura en forma automática.

3.b FOR descendente (For-downto ; VI>=VF)

```
For VC := VI downto VF do
  Sentencia;
```

El cambio que conlleva con respecto al anterior es que en cada iteración decremента en 1 la variable de control (VC)
Previo a cada iteración se verifica que se cumpla $VC \geq VF$.

Consideraciones:

- El valor inicial y el valor final se evalúan al comienzo (una sola vez) y si el valor inicial es mayor (o menor en el downto) que el final no se ejecuta el ciclo.
- La variable de control de ciclo (VC) y los límites inicial y final (VI y VF) NO deben ser alterados dentro del ciclo For.
- La variable de control queda indefinida luego de terminar For.
- Es posible calcular la cantidad de repeticiones a partir de VI, VF y el Paso (1 ó -1).

$$\text{CantRepeticiones} = (\text{VF} - \text{VI}) * \text{Paso} + 1$$

Ej1 –Ingresar una cantidad N de números enteros, calcular el promedio de positivos y negativos, además la cantidad de ceros

```

Program Ej1;
Var
  ContP, ContN, SumP, ContC, N, i : word;
  Num, SumN: integer;
Begin
  ContP:=0; ContN:=0; ContC:=0;
  SumP:=0; SumN:=0;
  Write ('Ingresar cant. nros. '); Readln(N);
  For i:= 1 to N do
    Begin
      Write ('Ingrese un número');
      Readln(Num);
      If Num > 0 then
        Begin
          ContP:= ContP + 1; SumP:= SumP + Num;
        End
      Else
        If Num < 0 then
          Begin
            ContN:= ContN + 1; SumN:= SumN + Num;
          End
        Else
          ContC:=ContC + 1;
    End;
  If ContP <> 0 then
    Writeln('Promedio positivos', SumP DIV ContP)
  Else
    Writeln('no ingresaron numeros positivos');
  If ContN <> 0 then
    Writeln('Promedio negativos', SumN DIV ContN)
  Else
    Writeln('no ingresaron numeros negativos');
  Writeln('ingresaron', ContC, 'ceros');
End.

```

Ej2. Leer N números enteros, calcular e informar el máximo.

(iniciando el máximo con un a) **valor imposible** o b) **con el 1er valor leído**)

```
{inicia el máximo con un valor imposible}
Program E2_a;
Var
  Num, Max: integer;
  N, i : byte;
Begin
  Write ('Ingresar cantidad de numeros:');
  Readln(N);
  Max:= -9999;
  For i:= 1 to N do
    Begin
      Write ('Ingrese un número');
      Readln(Num);
      If Num > Max then
        Max := Num;
      End;
    Writeln('El maximo es', Max);
  End.
```

```
{inicia el máximo con el 1er valor leído}
Program E2_b;
Var
  Num, Max: integer;
  N, i : byte;
Begin
  Write ('Ingresar cantidad de numeros:');
  Readln(N);
  Writeln('Ingrese el 1er. numero:');
  Readln (Max);
  For i:= 2 to N do
    Begin
      Write ('Ingrese un número');
      Readln(Num);
      If Num > Max then
        Max := Num;
      End;
    Writeln('El maximo es', Max);
  End.
```

2. Ciclo Condicional

La característica de estos ciclos es que están “condicionados” por una expresión lógica cuyo resultado determina la continuación o la finalización del ciclo.

Generalmente la expresión lógica utiliza operadores relacionales para enfrentar una variable a un valor preestablecido. Se utilizan operadores lógicos que potencian el control que se ejerce sobre el ciclo.

2a. Ciclo While

<pre>..... ; While Expresión lógica do Sentencia; ;</pre>	<pre>..... ; While Expresión lógica do Begin Sentencia₁; Sentencia₂; Sentencia_n; End;</pre>
---	--

Se evalúa la expresión lógica, si es verdadera ejecuta la sentencia (simple o compuesta) y vuelve a evaluar la expresión. Si es falsa el ciclo finaliza y continua la ejecución de la sentencia que está físicamente abajo del ciclo.

Notar que:

- Evalúa la expresión lógica al comienzo, por lo tanto, si la primera vez resulta falsa no entra al ciclo y no se ejecuta ni una vez.
- alguna de las variables que forman parte de la expresión lógica deben modificarse durante la ejecución del ciclo, para que la expresión pase de resultar verdadera a resultar falsa y así detener el ciclo. En caso contrario, el ciclo resulta infinito.

Ejemplos que utilizan la estructura While (no se conoce la cantidad de repeticiones).

2.a.1-El ciclo se detiene cuando una variable toma por lectura un valor "imposible". Ejer. 3 y 4

Ej3. Leer un conjunto de números enteros cuya cantidad no se conoce de antemano (se sabe que son distintos de cero), calcular el promedio de positivos y negativos.

```

Program Ej3;
Var
ContP, ContN, SumP: word; Num, SumN: integer;
Begin
ContP:=0; ContN:=0; SumP:=0; SumN:=0;
Write ('Ingresar un numero:'); Readln(Num);
While Num <> 0 do
  Begin
    If Num > 0 then
      Begin
        ContP:= ContP +1; SumP:= SumP + Num;
      End
    Else
      Begin
        ContN:= ContN +1; SumN:= SumN + Num;
      End
    Write('Ingresar un numero:'); Readln(num);
  End;
If ContP <> 0 then
  Writeln('Promedio positivos', SumP DIV ContP)
Else
  Writeln('no ingresaron numeros positivos');
If ContN <> 0 then
  Writeln('Promedio negativos', SumN DIV ContN)
Else
  Writeln('no ingresaron numeros negativos');
End.
  
```

Notar que a diferencia del ejercicio 1, el cero es un valor de corte.....no se pueden contar los ceros....

Ej4. Leer un conjunto de números enteros cuya cantidad no se conoce de antemano, calcular el promedio de positivos y negativos, además la cantidad de ceros.

```

Program Ej4;
Var
ContP, ContN, SumP, ContC : word; Num, SumN: integer; Rers: char;
Begin
ContP:=0; ContN:=0; ContC:=0; SumP:=0; SumN:=0;
Write ('Ingresa nro. (S/N)'); Readln(Res);
While Res = 'S' do
  Begin
    Readln(Num);
    If Num > 0 then
      Begin
        ContP:= ContP +1; SumP:= SumP + Num;
      End
    Else
      If Num < 0 then
        Begin
          ContN:= ContN +1; SumN:= SumN + Num;
        End
      Else
        ContC:=ContC + 1;
    Write ('Ingresa nro. (S/N)'); Readln(Res);
  End;
  
```

El cero es incluido como valor posible, el fin del ciclo depende de la respuesta del

2.a.2-El ciclo debe detenerse cuando una variable toma por cálculo un resultado límite.

Ej5. Leer un conjunto de números reales y sumarlos, detener el proceso cuando la suma supere un cierto valor X, ingresado por teclado.

Mostrar la cantidad de números sumados y la suma obtenida.

```

Program Ej5;
Var
  Num, X, Sum: real;
  Cont: word;
Begin
  Sum:=0;  Cont:= 0;
  Writeln('Ingresar la cota de la suma:');  Readln(X);
  While Sum <=X do
    Begin
      Writeln('Ingresar un numero:');    Readln(Num);
      Sum:= Sum + Num;    Cont:= Cont +1;
    End;
  Writeln('Son', Cont, 'numeros');
  Writeln('Su suma es', Sum:8:2);
End.

```

2.a.3-El control del ciclo puede potenciarse utilizando operadores lógicos para armar condiciones compuestas.

Ej6. Leer y sumar un conjunto de números reales, detener el proceso cuando la suma supere un cierto valor X, o la cantidad de número leídos sea mayor a M (X y M ingresados por teclado).

Mostrar la cantidad de números sumados y la suma obtenida.

```

Program Ej6;
Var
  Num, X, Sum: real;
  Cont: word;
Begin
  Sum:=0;  Cont:= 0;
  Writeln('Ingresar la cota de la suma y de los numeros:');
  Readln(X, M);
  While (Sum <=X) and (Cont <= M) do
    Begin
      Writeln('Ingresar un numero:');    Readln(Num);
      Sum:= Sum + Num;    Cont:= Cont +1;
    End;
  Writeln('Son', Cont, 'numeros');
  Writeln('Su suma es', Sum:8:2);
End.

```

Es importante que alguna de las variables que intervienen en la expresión lógica que controla el ciclo While, cambien su valor dentro del ciclo, en caso contrario el ciclo es infinito (no se detiene). Se generaría un ciclo infinito si las siguientes sentencias faltaran en los ejemplos anteriores:

```
Readln(num);    Sum:= Sum + Num;    Cont:= Cont +1;
```

- Si el ciclo debe repetirse una determinada cantidad de veces que se conoce de antemano, se requiere evaluar si un contador de repeticiones ha alcanzado dicha cantidad.
- En dicho caso la implementación del While es equivalente al funcionamiento de la estructura For

```
VC:= VI;  
While VC<= VF do  
  begin  
    Sentencia;  
    VC := VC + 1;  
  End;
```



```
For VC:= VI to VF do  
  Sentencia;
```

- En este caso se recomienda el uso de la estructura For

2.a.4. Ciclo de lectura de datos desde archivo

Un caso particular es el ciclo que lee datos de un archivo, la función booleana eof() que controla si se alcanzó o no el fin de archivo constituye la expresión lógica que controla el fin o la continuación del ciclo, respectivamente.

Tanto la entrada como la salida estándar, usan los archivos del sistema **input** y **output** por defecto, o sea que no es necesario especificar el teclado y la pantalla respectivamente.

Si se requiere lectura o escritura sobre archivo es necesario indicar el nombre del archivo (y en el caso de que este no se encuentre en la misma ubicación del programa, también indicar unidad y carpeta que lo contiene), como así también la operación a realizar sobre el mismo (leer o escribir).

Consideraciones previas para lectura o escritura desde un almacenamiento secundario con archivos de texto (*text*)

Si el programador desea grabar/recuperar datos desde un **archivo de texto**, debe declarar la variable de tipo **text**. Estos archivos son secuenciales, es decir se acceden en forma progresiva desde el comienzo al final, esta característica los hace lentos y se recomienda su uso con información que será procesada en su totalidad.

Un archivo de texto es utilizado para leer o grabar, en forma excluyente o sea no admite ambas operaciones simultáneamente.

Los datos en un archivo de texto se almacenan usando el código ASCII, en el cual cada carácter es representado por un simple byte. Debido a que los archivos de texto utilizan el código ASCII, pueden ser creados, inspeccionados y modificados utilizando un editor.

Los datos se almacenan como cadenas de caracteres, es decir, los números se almacenan con su representación ASCII y no su representación numérica.

Esto implica una conversión automática en el almacenamiento y la recuperación para los datos numéricos.

Si bien las dos características anteriores los hacen lentos y restringidos, son útiles porque son estándar, reconocidos en cualquier ámbito.

Luego de declarar la variable de tipo archivo en la sección

Var

NombreArch:text;

Deben utilizarse las siguientes sentencias, de cada uno se describe la sintaxis y su función:

Sentencia	Sintaxis	Función
assign	assign(nombre archivo Pascal, nombre archivo en el disco)	Enlaza la variable de tipo text con el archivo físico en el disco. Necesario para lectura/ escritura.
reset	reset(nombre del archivo Pascal)	Prepara al archivo para la lectura, ubicándose al comienzo.
read/ readln	read/readln(nombre archivo Pascal, lista de variables)	Se leen del archivo los valores y se almacenan en las variables indicadas en la lista.
rewrite	rewrite(nombre del archivo Pascal)	Prepara al archivo para la escritura, ubicándose al comienzo.
write/ writeln	write/writeln(nombre del archivo Pascal, lista de variables)	Se graban en el archivo los valores contenidos en las variables de la lista.
close	close(nombre del archivo Pascal)	Cierra el archivo.
eof	eof(nombre de archivo Pascal)	Función booleana que detecta el fin del archivo

Ha de tenerse cuidado con la sentencia **rewrite**, pues si el archivo no existe, se crea uno nuevo, pero si ya existe, *su contenido se pierde al sobrescribir los nuevos valores*.

Ej7 - Ingresar números reales desde un archivo Numeros.txt, están grabados uno por línea. Calcular el porcentaje de negativos, de positivos y de ceros.

Program Ej7;

Var

ContP, ContN, ContC, Total : word; Num: integer;

Arch: text;

Begin

ContP:=0; ContN:=0; ContC:=0;

Assign (Arch,'Numeros.txt'); {vincula el archivo Numeros.txt con la variable Arch}

Reset (Arch); {prepara el archivo para lectura}

While **not Eof(Arch)** do {controla que no haya alcanzado el fin de archivo}

Begin

Readln(Arch, Num);

If Num > 0 then

ContP:= ContP +1

Else

If Num < 0 then

ContN:= ContN +1

Else

ContC:=ContC + 1;

End;

Close (Arch); { cierra el archivo}

Total:= ContP + ContN + ContC;

Writeln('Porcentaje positivos', ContP * 100/Total:8:2);

Writeln('Porcentaje negativos', ContN * 100/Total:8:2);

Writeln('Porcentaje Nulos', ContC * 100/Total:8:2)

End.

Si están grabados en una única línea, cambia por
Read (Arch, Num);

Ej8 - En el archivo Clima.txt se ha grabado información climática, en cada línea:

Fecha (dd/mm/aaaa) máxima mínima

Se pide leer el archivo y grabar en otro archivo Amplitud.txt las fechas que registren amplitud térmica mayor a 10.

```
program Ej8;
var
  fecha:string[10];
  max, min:real;
  ArchE, ArchS:text; {se declaran dos archivos, se usara uno de entrada y otro de salida}
Begin
  Assign (ArchE,'Clima.txt'); Reset (ArchE);           {prepara el archivo para lectura}
  Assign (ArchS,'Amplitud.txt'); Rewrite(ArchS); {prepara el archivo para escritura}
  While not eof(ArchE) do
    Begin
      Readln(ArchE, fecha, max, min);           {lee en una línea una medición }
      If max - min > 10 then
        Writeln(ArchS, fecha);                 {si la amplitud es mayor a 10 graba la fecha}
      End;
    Close (ArchE); Close (ArchS);               {cierra los archivos}
  End.
```

Diferencias en la lectura de variables carácter o cadena y numérica

Imaginemos una flecha (puntero) como el indicador del próximo carácter a ser leído, o en el caso de escritura, la posición del próximo carácter a grabar. Cuando se abre un archivo la flecha se posiciona al comienzo y luego de cada operación de lectura o escritura avanza a la próxima ubicación.

Consideremos los caracteres

- <eoln> fin de línea (tipeando enter) internamente CTRL-M
- <eof> fin de archivo (close) internamente CTRL-Z

- ✓ Si la variable es **numérica**, reconoce un dígito o signo como comienzo, ignora blancos y <eoln> hasta encontrarlo. En el caso de variable real permite un punto. Cualquier otro carácter deviene en un error de ejecución. La secuencia leída finaliza al encontrar un blanco o <eoln>.
- ✓ Si la variable es **carácter o cadena** comienza a reconocer desde el carácter que indica la flecha.
 - Si es **carácter** toma el primero
 - Si es **cadena** toma tantos caracteres como la longitud de la misma o hasta que encuentre un <eoln>.

- ✓ En la lista de variables a leer pueden aparecer variables de distintos tipos, debe tenerse en cuenta la modalidad de lectura de cada uno.

Además, mientras que la sentencia Read avanza hasta el primer carácter no leído (donde queda posicionado), la sentencia Readln avanza al primer carácter de la línea siguiente, pudiendo haber quedado caracteres sin procesar.

Consideremos el caso de un archivo que contiene la siguiente información:

```
Arch =      1234.56 -789 345.67<eoln>
           w 456<eoln><eof>
```


Y las siguientes declaraciones de variables

X,W : real;
N,M : integer;
C1,C2: char;
Cad: string[5];

A continuación y siempre posicionado al principio del archivo, se muestra el resultado de diferentes órdenes de lectura

	X	W	N	M	C1	C2	Cad
Read (Arch, X, N, C1)	1234.56		-789		' '		
Readln(Arch, X,N); Read(Arch, C1)	1234.56		-789		'w'		
Read(Arch, Cad, N, M, C1, C2)			56	-789	' '	'3'	'1234.'
Readln(Arch,X,Cad,W);Read(Arch,C1, N)	1234.56	345.67	456		'w'		' -789'

2.b. Ciclo Repeat

REPEAT

Sentencia/s;

UNTIL expresión lógica;

Se ejecutan las sentencias que contiene la estructura y luego evalúa la condición, si es falsa repite el ciclo hasta que la condición sea verdadera, cuando el ciclo se detiene continúa la ejecución de la sentencia que esta físicamente abajo del mismo.

Notar que:

- Evalúa la condición después de ejecutar las sentencias que contiene, por lo tanto se ejecuta al menos una vez.
- alguna de las variables que forman parte de la expresión lógica deben modificarse durante la ejecución del ciclo, para que la expresión pase de resultar falsa a resultar verdadera y así detener el ciclo.
- A diferencia del ciclo While que itera mientras la condición es verdadera, el Repeat itera si la condición resulta falsa (o sea se detiene cuando es verdadera).
- Para codificar una estructura While como una estructura Repeat, basta con negar la expresión lógica. Pero no es totalmente equivalente ya que la primera puede no ejecutarse nunca y la segunda se ejecuta al menos una vez.
- Las consideraciones hechas para la estructura While sobre las expresiones lógicas que controlan las repeticiones se aplican a la estructura Repeat (valor imposible por lectura, valor límite por cálculo, contador alcanza una cantidad establecida)

Se reprograman los ejemplos vistos anteriormente.

Ej9. Leer un conjunto de números reales y sumarlos, detener el proceso cuando la suma supere un cierto valor X, ingresado por teclado. Mostrar la cantidad de números sumados y la suma obtenida.

```

Program Ej9;
Var
  Num, X, Sum: real;
  Cont: word;
Begin
  Sum:=0;
  Cont:= 0;
  Writeln('Ingresar la cota de la suma:');
  Readln(X);
  Repeat
    Writeln('Ingresar un numero:');
    Readln(Num);
    Sum:= Sum + Num;
    Cont:= Cont +1;
  Until Sum > X ;
  Writeln('Son', Cont, ' numeros');
  Writeln('Su suma es', Sum:8:2);
End.

```

Ej10 - Leer un mes (1 .. 12) luego indicar a que trimestre pertenece (1 .. 4) . Validar la lectura hasta que ingrese un mes correcto.

```

Program Ej10;
var
  Mes:byte;
Begin
  Repeat
    writeln('ingrese mes');
    readln(Mes);
  Until (1<=Mes) and (Mes<=12);
  Writeln ('pertenece al trimestre ', (Mes-1) div 3 +1)
End.

```

Ej11 - Leer dos números reales, y luego utilizar en forma repetitiva un **menú** para obtener uno o más de los siguientes resultados: suma, resta, producto o división.

```

Program Ej11;
var
  x, y: real;
  op: char;
Begin
  writeln('ingrese dos números distintos de cero'); readln(x, y);
  Repeat
    writeln('S - Suma');
    writeln('R - Resta');
    writeln('P - Producto');
    writeln('D - Division');
    writeln('F - Finalizar');
  
```

menú

```

    writeln('Ingrese opcion');
    Readln(op);
    Case op of
        'S': writeln( x + y :8:2);
        'R': writeln( x - y :8:2);
        'P': writeln( x * y :8:2);
        'D': writeln( x / y :8:2);
    End;
    Until op = 'F';
End.

```

Los dos últimos ejemplos muestran las aplicaciones más comunes de la estructura Repeat.

Ej12- Implementar un juego donde la computadora genera al azar un número entre 1 y 50 y el usuario debe adivinarlo en a lo sumo 10 intentos.

```

Program Ej12;
Var
    Num, Aleatorio, Cont: byte;
Begin
    Randomize;
    Aleatorio:=Random(50)+1;
    Cont:= 0;
    Repeat
        Writeln('Ingresar un numero:');
        Readln(Num);
        Cont:= Cont +1;
    Until (Num=Aleatorio) or (Cont=10) ;
    IF Num=Aleatorio then
        Writeln('Gano')
    else
        Writeln('Agotó los 10 intentos');
    End.

```

3. Ejemplos (While – Repeat - For)


Ej13 - En un estudio de mercado (para el posicionamiento de un producto), se encuestó una cantidad no conocida de personas. Por cada una se obtuvo una de estas tres posibles respuestas:

- N - No lo conoce
- C - Lo conoce y lo consume
- A - Lo conoce y no lo consume

Se pide desarrollar una solución que ingrese las respuestas (F es fin de datos), calcule e informe:

- ✓ % de personas que no lo consumen, sobre el total de encuestados
- ✓ % de personas que lo consumen, sobre el total de personas que lo conocen
- ✓ % de personas que no lo conocen, sobre el total de encuestados

Ejemplo: C C N C N A C N F

ContN= 3		NoConsumen	= (3 + 1)*100/8
ContC= 4		ConsumenConocen	= 4*100/(4 + 1)
ContA= 1		NoConocen	= 3*100/8

```

Program Ej13;
Var
  ContT, ContC, ContN, ContA : word;
  Res : char;
Begin
  ContC:= 0;
  ContN:= 0;
  ContA:= 0;
  Writeln('C- lo conoce y consume ; N – no lo conoce ; A - lo conoce y no lo consume');
  Readln(Res);
  Res:= UPCASE (Res);
  While Res <> 'F' do
    begin
      Case Res of
        'C' : ContC:= ContC + 1;
        'N' : ContN:= ContN + 1;
        'A' : ContA:= ContA + 1;
      End;
      Writeln('C- lo conoce y consume ; N – no lo conoce ; A - lo conoce y no lo consume; F – fin');
      Readln(Res);
      Res:= UPCASE (Res);
    End;
  ContT:= ContC + ContN + ContA;
  writeln('% de personas que no lo consumen', (ContN + ContA)/ContT * 100 :8:2);
  writeln ('% de personas que lo consumen', ContC/(ContC + ContA) * 100 :8:2); (*)
  writeln ('% de personas que no lo conocen', ContN /ContT * 100 :8:2);
end.

```

(*) Si la encuesta solo registra datos de personas que no lo conocen, esta sería una división por cero. Por lo tanto, debería controlarse esta situación utilizando una estructura alternativa (If).

Ej14 – Ingresar desde un archivo un conjunto de números naturales, mostrar los que tienen los factores primos (de 1 sólo dígito, 2, 3, 5, 7) coincidentes con los del primero leído.
 Ejemplo: **12, 18, 21, 36, 42, 15, 54** → sus factores son solo 2 y 3

```

Program Ej14;
Var
  Fact2, Fact3, Fact5, Fact7 : Boolean;
  N: word;
  Arch : text;
Begin
  Assign (Arch,'Datos.txt');
  Reset (Arch);
  Read(Arch, N);
  {lee y establece los divisores del primer número}
  Fact2:= N mod 2 = 0;
  Fact3:= N mod 3 = 0;
  Fact5:= N mod 5 = 0;
  Fact7:= N mod 7 = 0;

```

```

While not eof(Arch) do
  Begin
    Read(Arch,N);
    If (Fact2 = (N mod 2 = 0)) and (Fact3 = (N mod 3 = 0)) and (Fact5 = (N mod 5 = 0)) and
                                                (Fact7 = (N mod 7 = 0)) then
      Writeln(N);
    End;
  Close(Arch);
End.

```

5. Procesamiento de secuencias de caracteres

El reconocimiento de ciertos “**patrones**” sobre las secuencias de caracteres tiene múltiples aplicaciones.

Se presentan varios ejemplos para procesar secuencias de caracteres **terminadas con un punto y almacenadas en una línea de un archivo de texto**.

```

XXXXXXXX XXXX X XXX XX.
XXXXXXXX XXXX X XXX XX .

```

Una secuencia vacía contiene solo un punto.

La lectura se realiza carácter a carácter.

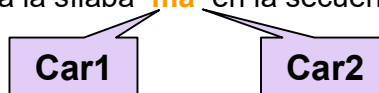
Uno de los problemas típicos es reconocer “**parejas**” de caracteres, lo que requiere tener **dos caracteres consecutivos** de la secuencia en dos variables.

Ej 15. Determinar cuántas veces se encuentra la sílaba ‘ma’ en la secuencia.

```

Program Ej15;
Var
  Arch: Text;
  Car1, Car2: char;
  ContMa: word;
Begin
  ASSIGN (Arch, 'Secuen.txt'); RESET(Arch);
  ContMa := 0;
  READ(Arch, Car1); {lee el primer caracter de la pareja}
  While Car1 <> '.' do
    Begin
      READ(Arch, Car2); {lee el segundo caracter de la pareja}
      If (Car1 = 'm') and (Car2 = 'a') then
        ContMa := ContMa + 1;
      Car1 := Car2; {el segundo carácter será el primero de la próxima pareja}
    End;
  CLOSE(Arch);
  Writeln ('El total de <ma> es', ContMa);
End.

```



Ej 16. Si se requiere contar palabras en una secuencia, consideremos que una **palabra** es una **serie de caracteres distintos de blanco antecedida por uno o más blancos** (salvo la primera).

```

XXXXX  XXXXXX  XXX  X  XX.

```


Tomando la **idea** del algoritmo anterior para **contar parejas**, la pareja formada por **Car1 = blanco y Car2 <> blanco** indica el **comienzo** de una **palabra**.

```

Program CuentaComienzoPalabras;
Var Arch: Text;
    Car1, Car2: char;
    ContPal: word;
Begin
    ASSIGN (Arch, 'Secuen.txt'); RESET(Arch);
    ContPal := 0;
    Car1 := ' '; {pone blanco en el 1er caracter de la 1ra pareja, para detectar la 1ra palabra}
    While Car1 <> '.' do begin
        READ(Arch, Car2);
        If (Car1 = ' ') and (Car2 <> ' ') and (Car2 <> '.') then {es comienzo de palabra}
            ContPal := ContPal + 1;
        Car1 := Car2;
    End;
    CLOSE(Arch);
    Writeln('El total de palabras es', ContPal);
End.

```

¿Qué sucede cuándo hay más de un espacio entre palabras?

Ej 17. Otra posibilidad para **contar palabras** es detectar el **final** de una **palabra** como una pareja **xxxxx** **x** **xxxxxx** **x** **xxx** **x** **x** **x** **xx** **x**.  Car1 <> blanco y (Car2 = blanco o Car2 = '.')

```

Program CuentaFinPalabras;
Var Arch: Text;
    Car1, Car2: char;
    ContPal : word;
Begin
    ASSIGN (Arch, 'Secuen.txt'); RESET(Arch);
    ContPal := 0;
    Read(Arch, Car1); {lee el primer caracter de la pareja}
    While Car1 <> '.' do
        begin
            READ(Arch, Car2); {lee el segundo caracter de la pareja}
            If (Car1 <> ' ') and ( (Car2 = ' ') or (Car2 = '.') ) then {evalúa si es fin de palabra}
                ContPal := ContPal + 1;
            Car1 := Car2; {asigna el primer caracter de la pareja}
        End;
    CLOSE(Arch);
    Writeln('El total de palabras es', ContPal);
End.

```

¿Qué sucede cuándo hay más de un espacio entre palabras?

Ej 18. Contar las palabras que terminen con una letra que se ingresa por teclado.

xxxxx **x** **xxxxxx** **a** **xxx** **x** **a** **a** **xa** **x**.  Car1 = 'a' y (Car2 = blanco o Car2 = '.')

```

Program CuentaFinLetra;
Var Arch: Text;
    Car1, Car2, Letra: char;
    ContLetra : word;

```

Begin

ASSIGN (Arch, 'Secuen.txt'); RESET(Arch);

Write('Ingrese letra de terminación');

Readln(Letra);

ContLetra := 0;

Read(Arch, Car1);

While **Car1 <> '.'** do

Begin

READ(Arch, Car2);

If **(Car1 = Letra) and ((Car2 = ' ') or (Car2 = '.'))** then {evalúa si es fin de palabra}

ContLetra := ContLetra + 1;

{y si termina con Letra}

Car1:= Car2;

End;

CLOSE(Arch);

Writeln ('El total de palabras que termina con ', Letra, ' es ', ContLetra);

End.

6. Ciclos Anidados

- ✓ Los **ciclos secuenciales** son independientes, después que finaliza el primer ciclo, comienza el segundo.

Program CiclosSecuenciales;

Var

Letra: Char;

Nro:byte;

Begin

For Nro:= 1 to 5 do

Write (Nro:3);

Writeln;

For Letra:= 'a' to 'z' do

Write (letra:3);

End.

1 2 3 4 5

a b c d e f g h i j k l m n o p q r s t u v w x y z

Total de repeticiones

5 + 26

- ✓ Los **ciclos anidados** son dependientes, por cada iteración del ciclo externo se ejecuta íntegramente el ciclo interno.

Program CiclosAnidados;

Var

Letra: Char;

Nro:byte;

Begin

For Nro:= 1 to 5 do

begin

Write (Nro:3);

For Letra:= 'a' to 'z' do

Write (letra:3);

Writeln;

End;

End.

1 a b c d e f g h i j k l m n o p q r s t u v w x y z

2 a b c d e f g h i j k l m n o p q r s t u v w x y z

3 a b c d e f g h i j k l m n o p q r s t u v w x y z

4 a b c d e f g h i j k l m n o p q r s t u v w x y z

5 a b c d e f g h i j k l m n o p q r s t u v w x y z

Total de repeticiones

5 * 26

Ej19 – Se tiene un conjunto de números primos y por cada uno de ellos N números no primos, este agrupamiento se mantiene hasta que ingresa un cero. O sea se repite el siguiente esquema:

- P (número primo ó 0 que indica fin de datos)
- N (cantidad entera)
- y a continuación N números no primos.

Se pide ingresar dichos datos, calcular e informar:

- Para cada primo porcentaje de múltiplos entre los N no primos.
- Número primo con mayor porcentaje de múltiplos.
- Cantidad de primos sin múltiplos.

Ejemplo: a.

11, 5, 24, <u>44</u> , 34, <u>121</u> , 98	→ 40%		b. 5 c. 1
3, 3, 35, 25, 14	→ 0%		
5, 4, <u>15</u> , <u>100</u> , <u>30</u> , <u>85</u>	→ 100%		
2, 8, <u>4</u> , 63, <u>18</u> , 27, <u>32</u> , 21, <u>6</u> , 81	→ 50%		
0			

Program Ej19;

Var

P, N, NP, ContM, ContSM, i, Max, MaxP : word;

Porc. Real;

Begin

ContSM:= 0; Max:=0;

Writeln('Ingrese un número primo'); Readln(P);

While P <> 0 do

begin

Writeln('Ingrese la cantidad de no primos'); Readln(N);

ContM:= 0;

For i := 1 to N do

Begin

Writeln('Ingrese un número no primo'); Readln(NP);

If NP mod P = 0 then

ContM:= ContM + 1;

End;

Porc:= ContM*100/N;

Writeln('el % de múltiplos de ', P, 'es', Porc: 8:2);

If ContM = 0 then

ContSM:= ContSM + 1

Else

If Porc > Max then

Begin

Max:= Porc; MaxP:= P;

End;

Writeln('Ingrese un número primo'); Readln(P);

End;

writeln (ContSM, 'numeros primos no registraron multiplos');

if Max <> 0 then

writeln ('El numero primo con mas múltiplos es', MaxP)

else

writeln ('ningun numero primo tuvo múltiplos ');

end.

UNMDP

Programación I

Ej20 – N camiones tiene asignados una cantidad no conocida de bultos para ser cargados (no necesariamente podrán ser cargados en su totalidad). Cada camión tiene una capacidad en Tns. y cada bulto un peso en kgs. Los datos descriptos están organizados de la siguiente forma para cada uno de los N camiones:

- Capacidad del camión
- y a continuación el peso de cada bulto en kgs. (0 es fin de bulto para dicho camión)

Se pide ingresar dichos datos, calcular e informar:

- a. Para cada camión si pudo cargar todos los bultos.
- b. Peso total de los bultos que quedaron sin poder cargarse (en todos los camiones)
- c. Cuantos camiones completaron su capacidad en un 90 % o más (sin importar si quedaron o no bultos sin cargar).

Program Ej20;

Var

N, Cont90, i, : word;

Suma, Capacidad, Bulto, SinCargar: Real;

Todos:boolean;

Begin

Cont90:= 0; SinCargar:=0;

Writeln('Ingrese cantidad de camiones'); Readln(N);

For i := 1 to N do

begin

Writeln('Ingrese capacidad del camión'); Readln(Capacidad);

Suma:= 0;

Todos:= True;

Writeln('Ingrese peso del bulto'); Readln(Bulto);

Bulto:= Bulto/1000;

While Bulto > 0 do

Begin

If Suma + Bulto >= Capacidad then

Suma:= Suma + Bulto

Else

Begin

SinCargar:= SinCargar + Bulto;

Todos:= False;

End;

Writeln('Ingrese peso del bulto'); Readln(Bulto);

Bulto:= Bulto/1000;

End;

If Todos then

Writeln('Se cargaron todos los bultos para la capacidad', Capacidad)

Else

Writeln('No se cargaron todos los bultos para la capacidad', Capacidad);

If Suma >= 0.9 * Capacidad then {completó su capacidad en un 90% o más}

Cont90:= Cont90 + 1;

End;

Writeln (Cont90, 'camiones completaron el 90% o más de su capacidad');

Writeln ('quedaron ', SinCargar , 'Tns.');

end.

N=4

2	800	1000	0		
1	1200	450	500	0	
1.5	800	900	100	600	0
2	200	1500	0		

Ej21-- Un archivo contiene información sobre el rendimiento académico de un grupo de alumnos. En cada línea se tiene :

- Matrícula (numérica de 4 dígitos)
- Apellido y nombre (sin blancos intermedios)
- Promedio (numérico)
- Historia académica

El diseño de la línea, cuya longitud no se conoce y puede exceder los 255 caracteres, es el siguiente:

```
9999      xxxxxxxxxxxx      99.99      xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx<eoln>
```

Tampoco se conoce la cantidad de blancos que se encuentran separando cada dato.

Se pide procesarlo, listando los Apellidos y nombres de los alumnos con promedio mayor a 8 y al final del proceso informar la matrícula correspondiente al promedio mas alto.

Program Alumnos;

Var

Max, Promedio : real;

MaxMatri, Matricula : word;

ApelNom: string;

Car: char;

Arch: text;

Begin

Assign (Arch, 'Alumnos.txt'); Reset(Arch);

Max:= 0;

While Not Eof(Arch) **do**

Begin

Read(Arch, Matricula);

Read(Arch, Car) ;

While (Car = ' ') **do**

Read(Arch, Car) ; *{avanza ignorando blancos hasta encontrar el apellido}*

ApelNom:= '';

While Car <> ' ' **do** *{copia el apellido y nombre en ApelNom}*

Begin

ApelNom:= ApelNom +Car; ReadArch, Car) ;

End;

Read(Arch, Promedio);

If Promedio > 8 **then** *{evalúa si debe listar}*

Writeln (ApelNom, Promedio:8:2);

If Promedio > Max **then** *{evalúa si es máximo promedio}*

Begin

Max:= Promedio; MaxMatri:= Matricula;

End;

Readln(Arch);

End;

Writeln(MaxMatri, 'Es la matrícula de mayor promedio');

Close(Arch);

End.

Ej 22. A partir de un archivo de entrada que contiene secuencias de caracteres que forman palabras, separadas por uno o más blancos, escribir en un archivo de salida las palabras que tienen longitud creciente.

Ejemplo: La casa blanca de la colina fue derribada ayer. → La casa blanca derribada

La cantidad de caracteres de cada palabra debe ser comparada con la longitud de la última palabra grabado, si es mayor se graba.

Solución: no es suficiente determinar el comienzo y fin de cada palabra, pues luego de evaluar y comparar su longitud, se decide si se graba en el archivo de salida. Si no se tuvo la precaución de almacenar en una string, los caracteres que la conforman, al momento de grabar estos ya no están en memoria.

Al recorrer la secuencia, el algoritmo detecta el primer carácter distinto de blanco y lo almacena en una string junto a los siguientes caracteres distintos de blanco, el primer carácter igual a blanco o a punto indica fin de palabra. Se evalúa si la longitud es mayor a UltPal (la de la última grabada) y si es así se graba actualizando UltPal.

Program GrabaCreciente;

Type

St20=string[20];

Var

ArchEnt, ArchSal : Text;

Car: char;

i , UltPal : word;

Pal:St20;

Begin

ASSIGN (ArchEnt, ' Secuen.txt'); RESET(ArchEnt);

ASSIGN (ArchSal, ' Crecient.txt'); REWRITE(ArchSal);

UltPal:= 0;

Read(ArchEnt, Car);

{ lee el primer caracter}

While Car <> ' ' **do**

If Car <> ' ' **then**

{evalúa si es el comienzo de palabra}

Begin

i:= 0;

While (Car <> ' ') and (Car <> '.') **do**

{arma la palabra en Pal}

Begin

i:= i + 1;

Pal:= Pal + Car;

READ(ArchEnt, Car) ;

End;

If i > UltPal **then**

{evalúa si es mas larga que la ultima palabra grabada}

Begin

Write (ArchSal, Pal ' '); *{graba Pal en el archivo de salida y luego un blanco}*

UltPal:= i; *{actualiza la longitud de la ultima palabra copiada}*

End;

End

else

Read(ArchEnt, Car);

{si es blanco lee otro caracter}

CLOSE(ArchEnt); CLOSE(ArchSal);

End.

Ej 23. En un archivo se han grabado palabras finalizando con un punto, cada palabra está conformada por letras y dígitos (3ab4c3Hd).

Se pide a partir del archivo descripto generar otro cambiando las parejas “DigitoLetraminuscula” por las repeticiones de la Letra según indique el dígito.

Ejemplos: **3**ab4**C3**hd ab**4c2**hd. → **aa**ab4**Ch**hd ab**cccc**hd.
0ab4**C3**hd ab**4c0**hd. → b4**Ch**hd ab**cccc**d

Program Cambia;

Var

ArchO, ArchD: Text;

Car1, Car2: char;

c: char;

Begin

ASSIGN (ArchO, 'Origen.txt'); RESET(ArchO);

ASSIGN (ArchD, 'Destino.txt'); REWRITE(ArchD);

READ(ArchO, Car1); {lee el primer caracter de la pareja}

While Car1 <> '.' do

Begin

READ(ArchO, Car2); {lee el segundo caracter de la pareja}

If (Car1 in ['0'..'9']) and (Car2 in ['a'..'z'])then

begin

For c:= '1' to Car1 do

Write (ArchD, Car2);

READ(ArchO, Car2);

end

Else

Write (ArchD, Car1);

Car1 := Car2; {el segundo caracter ser el primero de la próxima pareja}

End;

CLOSE(ArchO); CLOSE(ArchD);

RESET(ArchD); {se abre para ver los resultados}

While Not eof(ARCHD) do

Begin

Read (ArchD, Car1); Write (Car1);

End;

CLOSE(ArchD);

End.