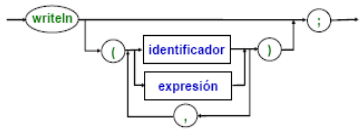
Resolución de Problemas y Algoritmos

Lenguaje Pascal

Entrada y Salida

Sentencia Writeln

Diagrama Sintáctico:



Muestra el contenido de cada uno de los parámetros en la misma línea *y luego avanza a la próxima línea*.

Ejemplo:

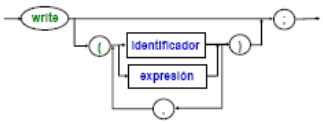
```
Ancho:= 3;  Alto:= 2;
writeln('El tamaño es', Ancho, ' por ', Alto,' :' );
writeln(Ancho * Alto, ' cm cuadrados ');
```

Se mostrará:

```
El tamaño es 3 por 2 :
6 cm cuadrados
```

Sentencia Write

Diagrama Sintáctico:



Muestra el contenido de cada uno de los parámetros en la misma línea.

Ejemplo:

```
Ancho:= 3;  Alto:= 2;
write('El tamaño es', Ancho, ' por ', Alto,' :' );
write(Ancho * Alto, ' cm cuadrados ');
```

Se mostrará:

```
El tamaño es 3 por 2 : 6 cm cuadrados
```

Write / Writeln

• **Usualmente, se utiliza el writeln para mostrar resultados. Sin embargo, si la cantidad de información es mucha es conveniente distribuirla en varios write para mayor claridad.**

Ejemplo:

```
writeln(P1,P2,P3,P4,P5,P6,P7,P8);
```

es equivalente a

```
write(P1,P2,P3);
write(P4,P5,P6);
writeln(P7,P8);
```

Write/writeln con formato

• **Generalmente, se necesita mostrar la información con un formato específico o en forma tabular.**

Formato:

```
write(<ParámetroEntero>:fw);
write(<ParámetroReal>:fw:dp);
```

Enteros con formato

• **Para formatear un valor Entero se utiliza un ancho de campo (fw) que especifica la cantidad de lugares que ocupará el número.**

***Obs**: el signo (-) ocupa 1 lugar.*

• **Si la cantidad de dígitos a mostrar es igual a fw se muestra el valor sin modificación.**

Ej.: `write(123:3);` *Muestra:* `123`

• **Si la cantidad de dígitos a mostrar supera a fw se agregan los lugares necesarios.**

Ej.: `write(123:1);` *Muestra:* `123`

• **Si la cantidad de dígitos a mostrar es inferior a fw se agregan espacios, justificando el número a la derecha.**

Ej.: `write(123:6);` *Muestra:* `...123`

Reales con formato

• **Para formatear un valor Real además del ancho de campo (fw) se utiliza un valor para especificar la cantidad de dígitos decimales a ser mostrado (dp).**

• **El ancho de campo (fw) indica la cantidad de dígitos en la parte entera, más los dígitos en la parte decimal, más el lugar del punto decimal (.)**

• **La parte decimal se redondea y la parte entera mantiene la cantidad de dígitos (igual que enteros).**

Ejemplo**: Considere X un valor Real y el formato **:5:1

Valor de X	Salida
-99.42	-99.4
0.123	→→0.1
-9.53	→-9.5
99.999	100.0

***Ejercicio:** Como sería mostrado el valor **-15.564** almacenado en X utilizando los siguientes formatos*

```
X:8:4    X:8:3    X:8:2    X:8:1    X:8:0    X:8
-15.5640
        .-15.564
                ..-15.56
                        ...-15.6
                                .....-16
                                    -1.6E+001
```

Sentencias Read y Readln

❖ **Tecla ENTER**

•En las computadoras, la tecla **ENTER** tiene asociados 2 caracteres cuyos códigos ASCII son 13 y 10

ASCII 13: retorno de carro *(CR:carriage return)*

ASCII 10: nueva línea *(LF: line feed)*

•Los caracteres 13 y 10 son de control y al imprimirlos en pantalla producen un efecto en lugar de mostrar algo visible.

❖ BUFFER de lectura

•Todos los datos ingresados por teclado se almacenan en una zona llamada buffer de lectura (o simplemente buffer).

•Todos los caracteres ingresados quedan retenidos en el buffer hasta que son leídos por los programas (en el caso de Pascal por read y readln).

•El buffer tiene un puntero (↑), el cual señala al elemento que está siendo leído por el programa.

Read: comportamiento del buffer

•Al ejecutar un READ, *el programa se suspende* y todo lo ingresado por teclado se almacena en el buffer de lectura.

•Al presionar la tecla ENTER *la ejecución continúa* y el procedimiento READ lee del buffer los valores que necesita para asociar a las variables.

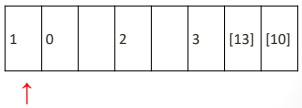
Ejemplo:

```
program ejemplo_read;
var
      a,b,c,d: integer;
begin
      write('Ingrese valores enteros:');
      read(a);   read(b);   read(c);
      readln;
      d := a + b + c;
      writeln('El valor de D es ',d);
      readln;
end.
```

Al ejecutar la primera primitiva READ, el programa se suspenderá y esperará a que ingresemos datos por teclado y ENTER.

Supongamos que ingresamos 10 2 3 y ENTER.

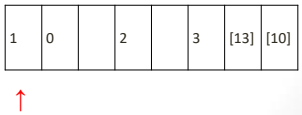
En el buffer se guardan todos los caracteres ingresados, incluido los espacios en blanco y el [13][10] (*ENTER*).



Una vez que se presiona ENTER, la primitiva READ comienza a leer del buffer en la posición que indica el puntero (↑).

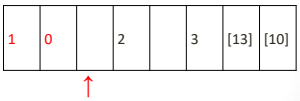
En el ejemplo, el espacio en blanco indica que termina el número y empieza otro.

Mientras READ va leyendo del buffer, el puntero se desplaza indicando el próximo carácter a leer.

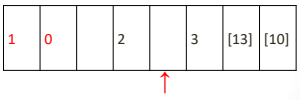


Como la variable A es de tipo integer, cuando el read lee un espacio en blanco almacena el número leído en la variable A. Lo mismo para la variable B y C.

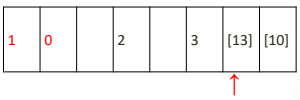
A recibe el valor 10



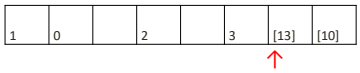
A recibe el valor 10
B recibe el valor 2



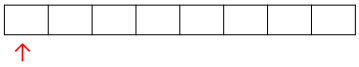
A recibe el valor 10
B recibe el valor 2
C recibe el valor 3



- Al ejecutarse las 3 primitivas READ del programa ejemplo, en el buffer aún quedan caracteres.



- La primitiva READLN saca un ENTER del buffer y vuelve el puntero al principio, dejando el buffer vacío preparado para una nueva entrada de datos.



Ejemplo:

```
program ejemplo_read_2;
var
      a,b,c: char;
begin
      write('Ingrese caracteres:');
      read(a);   read(b);   read (c);
      readln;
      writeln('Los tres primeros caracteres
      ingresados son: ',a, b,c);
      readln;
end.
```

Ahora las variables son de tipo **char**.

El read lee el primer carácter (***1***) y lo almacena en la variable A. Lo mismo para la variable B (***0***) y C (***espacio en blanco***).

1	0		2		3	[13]	[10]

↑

Otra vez, la primitiva READLN saca un ENTER del buffer y vuelve el puntero al principio, dejando el buffer vacío (es decir, se *descartan* los caracteres que no fueron leídos).