RESOLUCIÓN DE PROBLEMAS Y ALGORITMOS

CLASE 2

Lenguaje de Programación Pascal: constantes, variables, tipos y asignaciones.

Luciano H. Tamargo
http://cs.uns.edu.ar/~lt
Depto. de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur, Bahía Blanca
2016

CONCEPTOS DE LA CLASE PASADA

- Conceptos vistos:
  - Computadora
  - Algoritmo
  - Primitiva
  - Traza de un algoritmo

¿PREGUNTAS?

TEMARIO

- Metodología propuesta.
- Datos constantes o variables.
- Programa en Pascal.
- Declaraciones de constantes y variables.
- Primitiva de asignación.
- Primitivas para interacción con el usuario.

METODOLOGÍA PROPUESTA

- A continuación, se muestra en forma esquemática la **metodología propuesta en esta materia**.
- Esta metodología abarca los **pasos que proponemos se deben transitar**:
  - **desde** abordar el enunciado de un problema que se quiere resolver,
  - **hasta** llegar a tener una aplicación que resuelva ese problema con una computadora.
- Es muy importante tener en cuenta lo siguiente:
  - Aunque esta metodología pueda resultar exagerada para los problemas extremadamente simples que planteamos en las primeras clases, los problemas propuestos irán creciendo en complejidad en muy corto tiempo, y será entonces cuando usar una metodología de trabajo marcará la diferencia.

METODOLOGÍA GENERAL PROPUESTA

PROBLEMA
↓
SOLUCIÓN
↓
ALGORITMO
↓
verificación
↓
PROGRAMA
↓
verificación

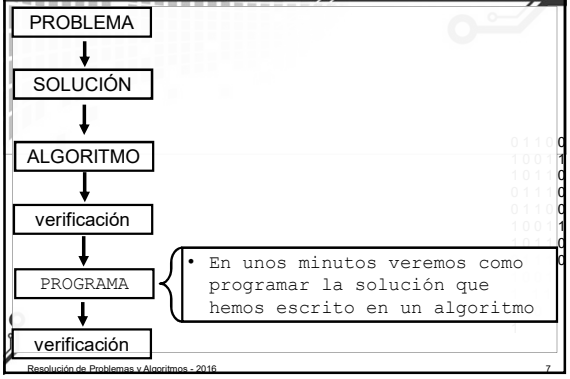
- Un **algoritmo** es la especificación de una secuencia de pasos u operaciones, que al ser ejecutadas permiten resolver un problema.
- Un **algoritmo** debe tener un único punto de inicio y al menos un punto final; y todos sus pasos deben estar expresados con operaciones comprensibles para quién las ejecutará (a las cuales llamamos primitivas).

METODOLOGÍA GENERAL PROPUESTA

PROBLEMA
↓
SOLUCIÓN
↓
ALGORITMO
↓
verificación
↓
PROGRAMA
↓
verificación

- Una **traza** es una **simulación** de la ejecución real de los pasos de un algoritmo, en la cual se lleva cuenta de los movimientos realizados y los cambios que se producen en los elementos o datos involucrados.

METODOLOGIA GENERAL PROPUESTA



Resolución de Problemas y Algoritmos - 2016 7

TEMARIO

- Metodología propuesta.
- Datos constantes o variables.
- Programa en Pascal.
- Declaraciones de constantes y variables.
- Primitiva de asignación.
- Primitivas para interacción con el usuario.

Resolución de Problemas y Algoritmos - 2016 8

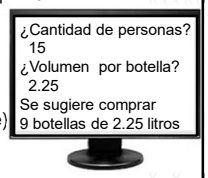
DATOS CONSTANTES O VARIABLES PARA UN PROBLEMA

- En general, los problemas involucran **datos**.
- Estos datos pueden ser:
  - **constantes** (no cambian su valor) o
  - **variables** (cambian su valor).
- Existen acciones primitivas que permiten darle un **valor inicial** o **modificar** el valor de los datos variables.
- A continuación, veremos un ejemplo que involucra datos constantes y variables.

Resolución de Problemas y Algoritmos - 2016 9

PROBLEMA PROPUESTO: BOTELLAS

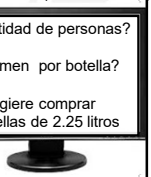
- **Crear una aplicación** para calcular cuantas botellas de gaseosa comprar para una reunión.
  - La aplicación debe interrogar al usuario, y utilizar la experiencia de expertos que recomiendan 1.25 litros por persona, y comprar el entero siguiente al resultado.
  - Por ejemplo, si el resultado es 8.33 comprar 9 botellas.
- **Datos involucrados:**
  - litros por persona (constante)
  - cantidad personas (variable)
  - volumen botella (variable)
  - cantidad botellas a comprar (variable)



- La solución se obtiene con Álgebra.
- Antes de hacer el algoritmo hagamos una tabla con algunos ejemplos para 1, 2, 3, o 10 personas.

ALGORITMO

- Podemos usar primitivas para mostrar texto en pantalla, leer datos de un dispositivo de entrada, asignar un valor a un dato y además, operadores matemáticos:
  - suma, multiplicación, división y eliminación de decimales.



- **Algoritmo** botellas:
  - mostrar-pantalla ¿Cantidad personas?
  - leer (personas)
  - mostrar-pantalla ¿Volumen por botellas? (litros)
  - leer (volumen)
  - asignar a cantidad, el resultado (sin decimales) de multiplicar (personas x 1.25) dividido por volumen, más 1 botella extra.
  - mostrar-pantalla Se sugiere comprar
  - mostrar-pantalla cantidad botellas de volumen litros.

Resolución de Problemas y Algoritmos - 2016 11

TEMARIO

- Metodología propuesta.
- Datos constantes o variables.
- Programa en Pascal.
- Declaraciones de constantes y variables.
- Primitiva de asignación.
- Primitivas para interacción con el usuario.

Resolución de Problemas y Algoritmos - 2016 12

METODOLOGIA GENERAL PROPUESTA

PROBLEMA
↓
SOLUCIÓN
↓
ALGORITMO
↓
verificación
↓
PROGRAMA
↓
verificación

- En lo que resta la clase de hoy, veremos lo necesario como para implementar el algoritmo anterior en el lenguaje de programación Pascal.
- En las clases siguientes seguiremos viendo más elementos, conceptos y primitivas que ofrece el lenguaje Pascal para programar.

CONCEPTO: LENGUAJE DE PROGRAMACIÓN

- Un **lenguaje de programación** es un lenguaje artificial creado para expresar procesos que pueden ser llevados a cabo por computadoras.
- Un lenguaje de programación se utiliza para crear **programas de computadoras**, y de esta manera, implementar algoritmos que controlen el comportamiento de una máquina y resuelvan tareas específicas.
- Un lenguaje de programación está definido por un **conjunto de símbolos**, **reglas sintácticas** (que definen su estructura) y **reglas semánticas** (que definen el significado de sus elementos).

EJEMPLO MUY SIMPLE: APLICACION DE CÁLCULO ÁREA CÍRCULO

PROBLEMA → Calcule el área de un circulo
SOLUCIÓN → Usar la fórmula "Pi por radio al cuadrado"
ALGORITMO → PROGRAMA

En algún lenguaje de programación

```
Ingrese Radio: 1
El área es
3.1416
```


EJEMPLO MUY SIMPLE

PROBLEMA → Calcule el área de un circulo
SOLUCIÓN → Usar la fórmula "Pi por radio al cuadrado"
ALGORITMO → PROGRAMA

En algún lenguaje de programación

Algoritmo Área Círculo
mostrar "Ingrese radio"
leer (**radio**)
Asignar a **área círculo** el resultado de
Pi x **radio** x **radio**
mostrar El área es: área círculo

```
Ingrese Radio: 1
El área es
3.1416
```


EJEMPLO MUY SIMPLE

PROBLEMA → Calcule el área de un circulo
SOLUCIÓN → Usar la fórmula "Pi por radio al cuadrado"
ALGORITMO → Algoritmo Área Círculo
PROGRAMA

En algún lenguaje de programación

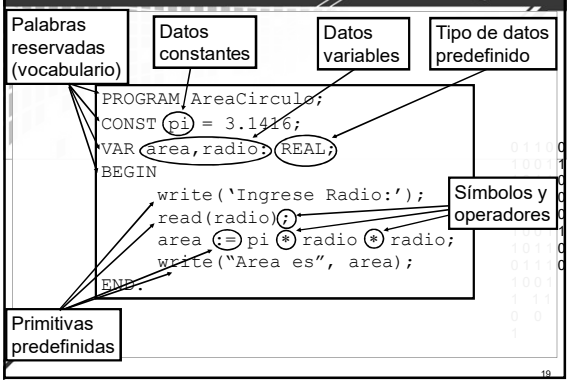
```
Ingrese Radio: 1
El área es
3.1416
```

```
PROGRAM AreaCirculo;
CONST pi = 3.1416;
VAR area,radio: REAL;
BEGIN
     write ('Ingrese Radio:');
     read(radio);
     area := pi * radio * radio;
     write("Area es", area);
END.
```

ALGUNOS ELEMENTOS DE UN PROGRAMA EN PASCAL

```
PROGRAM AreaCirculo;
CONST pi = 3.1416;
VAR area,radio: REAL;
BEGIN
     write ('Ingrese Radio:');
     read(radio);
     area := pi * radio * radio;
     write("Area es", area);
END.
```

ALGUNOS ELEMENTOS DE UN PROGRAMA EN PASCAL



19

IDENTIFICADORES DE PASCAL

- Un programa en Pascal puede tener **tres clases de identificadores** (nombres que identifican algo):
  - Reservados (ya tiene significado y no puede cambiarse)
  - Predefinidos (ya tiene significado preestablecido pero puede cambiarse)
  - Definidos por el programador (el significado lo establece el programador)
- Pascal no es sensible a mayúsculas y minúsculas: radio, RADIO y Radio son el mismo identificador.

Resolución de Problemas y Algoritmos - 2016 20

EJEMPLOS DE IDENTIFICADORES RESERVADOS

PROGRAM	identifica que lo que sigue es un programa en Pascal
CONST	identifica la declaración de los datos constantes
VAR	identifica la declaración de datos variables
BEGIN	identifica el comienzo del bloque ejecutable
END	identifica el final del bloque ejecutable

Identificadores reservados
El programador no puede cambiar su significado.

```
PROGRAM AreaCirculo;
CONST pi = 3.1416;
VAR area,radio: REAL;
BEGIN
     write ('Ingrese Radio:');
     read(radio);
     area := pi * radio * radio;
     write("Area es", area);
END.
```

EJEMPLOS DE IDENTIFICADORES PREDEFINIDOS

REAL	identifica a un tipo de dato predefinido.
write	identifica a una primitiva predefinida, la cual permite mostrar datos en una consola de pantalla.
read	identifica a una primitiva predefinida, la cual permite recuperar (leer) valores de un dispositivo de entrada y asignarlos a una variable

Identificadores predefinidos
Ya tienen un significado pre-establecido, pero el programador podría cambiarlo si quisiera.

```
PROGRAM AreaCirculo;
CONST pi = 3.1416;
VAR area,radio: REAL;
BEGIN
     write ('Ingrese Radio:');
     read(radio);
     area := pi * radio * radio;
     write("Area es", area);
END.
```

IDENTIFICADORES RESERVADOS EN PASCAL

- Listado de las palabras reservadas de Pascal (las que serán utilizadas en RPA están **resaltadas**).

and	**end**	nil	set
array	**file**	**not**	**then**
begin	**for**	**of**	**to**
case	**function**	**or**	**type**
const	goto	packed	**until**
div	**if**	**procedure**	**var**
do	in	**program**	**while**
downto	label	record	with
else	**mod**	**repeat**	

- Observe que REAL, write, read no están entre las palabras reservadas.

Resolución de Problemas y Algoritmos - 2016 23

IDENTIFICADORES DEFINIDOS POR EL PROGRAMADOR

- Son nombres que identifican a **elementos creados por el programador.**
- El significado es dado por el programador.
  - Deben comenzar obligatoriamente con una letra, y sólo involucran letras, números y el guion bajo "_" (underscore)
  - No pueden utilizarse las vocales con acentos, ni la letra ñ.
  - No pueden ser igual a una palabra reservada.

Son válidos:	**No son válidos:**
Radio	La cantidad
Pi	program
x23	%mas
es_nro_par	23i
UNprogramA	es-nro-par
SueldoNeto	Primo(i)
_Saldo	área
	año

Importante: no afecta si usamos mayúsculas o minúsculas. Ej: CANTIDAD, canTIDAD, y CaNtIdAd son el mismo identificador.

24

EL SÍMBOLO ; (PUNTO Y COMA)

- El símbolo "; " (punto y coma) se utiliza en Pascal como separador de sentencias.
- Pueden haber una o más sentencias en el mismo renglón.
- Además el "; " antes del END es optativo.

```
PROGRAM AreaCirculo;
CONST pi = 3.1416; {aproximación de pi}
VAR area,radio: REAL;
BEGIN {cálculo del área}
     write ('Ingrese Radio:');
     read(radio);
     area := pi * radio * radio;
     write("Area es", area);
END.
```

Todo texto entre llaves { } son comentarios del programador los cuales son ignorados durante la ejecución del programa.

El punto (.) indica el final del programa.

- Es muy importante que un programa sea fácil de leer.
- Hablaremos más sobre esto en las clases siguientes.

PARTES DE UN PROGRAMA

```
PROGRAM AreaCirculo;
```
Encabezado

```
CONST pi = 3.1416;
VAR area,radio: REAL;
```
Bloque de declaraciones

```
BEGIN
     write ('Ingrese Radio:');
     read(radio);
     area := pi * radio * radio;
     write("Area es", area);
END.
```
Bloque ejecutable

- Las instrucciones son ejecutadas de arriba hacia abajo y de izquierda a derecha.

TEMARIO

- Metodología propuesta.
- Datos constantes o variables.
- Programa en Pascal.
- Declaraciones de constantes y variables.
- Primitiva de asignación.
- Primitivas para interacción con el usuario.

DECLARACION DE DATOS VARIABLES Y CONSTANTES

```
CONST pi = 3.1416;
VAR area,radio: REAL;
```
Bloque de declaraciones

Definición de Constantes `CONST`

- Tienen un **valor fijo** asociado
- Se definen por un **nombre** (identificador) y tienen **implícitamente** asociado un **tipo de dato** dado por el valor elegido
- Ejemplo:

```
CONST pi = 3.1416;
      cant_de_meses = 12;
```

Definición de Variables `VAR`

- Su **valor es variable**
- Se definen por un **nombre** (identificador) y un **tipo de dato** asociado
- Ejemplo:

```
VAR radio: REAL;
```


EJEMPLOS DE TIPOS DE DATOS PREDEFINIDOS

- **Tipo de Dato**: define el **conjunto de valores** posibles que puede tomar una variable, y **las operaciones** que pueden aplicarse.
- Ejemplo:

Tipo de dato predefinido: REAL

- Es un **subconjunto** de los números reales. Se usa punto para separar la parte entera de la decimal.
  Ejemplos de valores: 3.1416 0.00001 128.5 3.0
- **Operaciones**: + - * / (y otras que mostraremos la próxima clase). Por ejemplo, trunc(R) es una función predefinida que dado un valor real R retorna un entero que corresponde a la parte entera de R.

EJEMPLOS DE TIPOS DE DATOS PREDEFINIDOS

- **Tipo de Dato**: define el **conjunto de valores** posibles que puede tomar una variable, y **las operaciones** que pueden aplicarse.
- Ejemplo:

Tipo de dato predefinido: INTEGER

- Es un **subconjunto** de los números enteros.
- **Operaciones**: + - * div (y otras que mostraremos la próxima clase).

DECLARACION DE CONSTANTES Y VARIABLES EN PASCAL

- Para usar datos en Pascal, hay que "declararlos":
  - Declaración de constantes: se escribe la palabra reservada CONST, y luego nombre y valor de cada constante usando el símbolo "=", por ejemplo:

```
CONST
 pi = 3.141592;
 e = 2.718281828;
```

Se separa una de otra con punto y coma (;)

  - Declaración de variables: se escribe la palabra reservada VAR, y luego los nombres y tipos de dato de cada variable usando el símbolo " : ", por ejemplo:

```
VAR
 contador: INTEGER;
 precio1, precio2, precio3: REAL
```

Puedo declarar varias variables del mismo tipo separándolas con coma. Con punto y coma separo una declaración de otra.

CONSTANTES, VARIABLES Y TIPOS

```
PROGRAM AreaCirculo;
CONST pi = 3.1416;
VAR area,radio: REAL;
```

Valor fijo y tipo fijo (real)

Conjunto de valores que puede tomar y operaciones que puede usar

  - Al declarar una constante se le asigna un valor que no puede cambiar durante la ejecución del programa.
  - Al declarar una variable **no se le asigna ningún valor** (solamente se indica que tipo de valores puede tener).
- Durante la ejecución del programa los valores de las variables se almacenan en la memoria de la computadora.
- La **declaración de una variable** indica que debe **reservarse un espacio** de memoria para esa variable.

TEMARIO

- Metodología propuesta.
- Datos constantes o variables.
- Programa en Pascal.
- Declaraciones de constantes y variables.
- Primitiva de asignación.
- Primitivas para interacción con el usuario.

LOS VALORES DE LAS VARIABLES

```
BEGIN
     write ('Ingrese Radio:');
     read(radio);
     area := pi * radio * radio;
     write("Area es", area);
END.
```

Bloque ejecutable

- Para darle valor a una variable se puede utilizar:
  1. La **primitiva predefinida** `read` que lee de un dispositivo de entrada un valor y lo asocia a la variable que tiene como parámetro. Ej: `read` (radio).
  2. La **primitiva de asignación**, la cual se representa con el símbolo `:=` (dos puntos igual).
- **Muy importante**: consideramos que es erróneo asumir que al declarar una variable, esta ya tiene un valor inicial como cero u otro valor. **Una variable sin valor es un error de programación**.

PASCAL: PRIMITIVA DE ASIGNACIÓN

- La primitiva de asignación en Pascal
  - Permite dar o cambiar el valor a una variable.
  - Se expresa con el símbolo :=
    A la **izquierda** del := debe ir obligatoriamente un identificador de **variable** y a la **derecha una expresión**.
  - Por ejemplo, si radio es una variable de tipo REAL, la siguiente asignación permite darle el valor "5.23" a radio y se lee *"a la variable radio le asigno el valor 5.23"*. *Si radio ya tenía valor, el valor anterior se pierde y es reemplazado con 5.23.*

```
radio:= 5.23;
```

- Hay una gran diferencia entre "saldo=10" y "saldo:=10"
  - saldo:=10 significa *"le doy el valor 10 a saldo"*
  - saldo=10 significa *"¿es saldo igual a 10?"*

PRIMITIVA DE ASIGNACIÓN

- En una asignación: variable := expresión
  1) **primero** se evalúa la expresión de la derecha y se obtiene un valor,
  2) **luego** se modifica el valor de la variable, perdiéndose el valor anterior.

```
PROGRAM Asigna1;
CONST c = 2.3;
VAR a, b: REAL;
BEGIN
  a:= 1.5;
  a:= 2 + 1/2;
  b:= 2 * a + c;
  b:= a * -1;
END.
```

modifica el valor de **a**

usa el valor de **a**

modifica el valor de **b**

METODOLOGIA GENERAL PROPUESTA

PROBLEMA
↓
SOLUCIÓN
↓
ALGORITMO
↓
verificación
↓
PROGRAMA
↓
verificación

- En una **traza** de un programa en Pascal se llevará cuenta de los pasos y cambios realizados.
- Principalmente de cambios en los valores de las variables.

PRIMITIVA DE ASIGNACIÓN

- En una asignación: variable := expresión
  1) **primero** se evalúa la expresión de la derecha y se obtiene un valor,
  2) **luego** se modifica el valor de la variable, perdiéndose el valor anterior.

```
PROGRAM Asigna1;
CONST c = 2.3;
VAR a, b: REAL;
BEGIN
  a:= 1.5;
  a:= 2 + 1/2;
  b:= 2 * a + c;
  b:= a * -1;
END.
```

Traza de los valores almacenados en memoria para cada variable:

a	b
?	?

PRIMITIVA DE ASIGNACIÓN

- En una asignación: variable := expresión
  1) **primero** se evalúa la expresión de derecha y se obtiene un valor,
  2) **luego** se modifica el valor de la variable, perdiéndose el valor anterior.

```
PROGRAM Asigna1;
CONST c = 2.3;
VAR a, b: REAL;
BEGIN
  a:= 1.5;
  a:= 2 + 1/2;
  b:= 2 * a + c;
  b:= a * -1;
END.
```

Traza de los valores almacenados en memoria para cada variable:

a	b
?	?
1.5	?

PRIMITIVA DE ASIGNACIÓN

- En una asignación: variable := expresión
  1) **primero** se evalúa la expresión de derecha y se obtiene un valor,
  2) **luego** se modifica el valor de la variable, perdiéndose el valor anterior.

```
PROGRAM Asigna1;
CONST c = 2.3;
VAR a, b: REAL;
BEGIN
  a:= 1.5;
  a:= 2 + 1/2;
  b:= 2 * a + c;
  b:= a * -1;
END.
```

Traza de los valores almacenados en memoria para cada variable:

a	b
?	?
1.5	?
2.5	?

PRIMITIVA DE ASIGNACIÓN

- En una asignación: variable := expresión
  1) **primero** se evalúa la expresión de derecha y se obtiene un valor,
  2) **luego** se modifica el valor de la variable, perdiéndose el valor anterior.

```
PROGRAM Asigna1;
CONST c = 2.3;
VAR a, b: REAL;
BEGIN
  a:= 1.5;
  a:= 2 + 1/2;
  b:= 2 * a + c;
  b:= a * -1;
END.
```

Traza de los valores almacenados en memoria para cada variable:

a	b
?	?
1.5	?
2.5	?
2.5	7.3

PRIMITIVA DE ASIGNACIÓN

- En una asignación: variable := expresión
  1) **primero** se evalúa la expresión de derecha y se obtiene un valor,
  2) **luego** se modifica el valor de la variable, perdiéndose el valor anterior.

```
PROGRAM Asigna1;
CONST c = 2.3;
VAR a, b: REAL;
BEGIN
  a:= 1.5;
  a:= 2 + 1/2;
  b:= 2 * a + c;
  b:= a * -1;
END.
```

Traza de los valores almacenados en memoria para cada variable:

a	b
?	?
1.5	?
2.5	?
2.5	7.3
2.5	-2.5

PRIMITIVA DE ASIGNACIÓN

- En una asignación: variable := expresión
  1) **primero** se evalúa la expresión de derecha y se obtiene un valor,
  2) **luego** se modifica el valor de la variable, perdiéndose el valor anterior.

```
PROGRAM Asigna1;
CONST c = 2.3;
VAR a, b: REAL;
BEGIN
  a:= 1.5;
  a:= 2 + 1/2;
  b:= 2 * a + c;
  b:= a * -1;
END.
```

Traza de los valores almacenados en memoria para cada variable:

a	b
?	?
1.5	?
2.5	?
2.5	7.3
2.5	-2.5

TEMARIO

- Metodología propuesta.
- Datos constantes o variables.
- Programa en Pascal.
- Declaraciones de constantes y variables.
- Primitiva de asignación.
- Primitivas para interacción con el usuario.

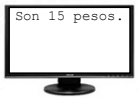
PRIMITIVAS PREDIFINIDAS PARA INTERACCIÓN CON EL USUARIO

- Procedimientos predefinidos de Pascal:
  - WRITE: muestra valores en la pantalla
  - WRITELN: muestra valores en pantalla y baja de línea (LN)
  - READ: obtiene (lee) valores ingresados por teclado
  - READLN: (read-line) idem a read pero espera por un ENTER

- **Observación**: en el horario de la práctica se explicarán más detalles sobre estas primitivas (no se lo pierda).

PRIMITIVAS PARA MOSTRAR EN PANTALLA

- WRITE: muestra valores en la pantalla
- WRITELN: muestra valores en pantalla y baja de línea (LN)



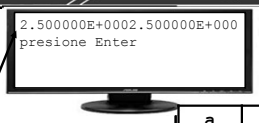

```
valor:=15;
write(´Son ´);
write(valor);
write(´ pesos.´);
```


```
Valor:=15;
writeln(´Son ´);
writeln(valor);
writeln(´ pesos. ´);
```

- **Observación**: en el horario de la práctica se explicarán más detalles sobre estas primitivas (no se lo pierda) .

FORMATEO DE SALIDA EN PANTALLA CON WRITE

```
PROGRAM Asigna1;
CONST c = 2.3;
VAR a, b: REAL;
BEGIN
  a:= 1.5;
  a:= 2 + 1/2;
  b:= 2 * a + c;
  b:= a * -1;
  writeln(a,b);
  writeln('presione Enter');readln;
END.
```



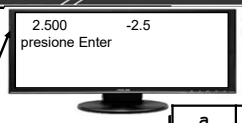
Si no se especifica ningún formato, muestra reales en notación científica.

a	b
?	?
1.5	?
2.5	?
2.5	7,3
2.5	-2.5

- El **readln** final espera un ENTER del usuario, evita que termine el programa y se cierre la consola (así puedo ver los resultados) .

FORMATEO DE SALIDA EN PANTALLA CON WRITE

```
PROGRAM Asigna1;
CONST c = 2.3;
VAR a, b: REAL;
BEGIN
  a:= 1.5;
  a:= 2 + 1/2;
  b:= 2 * a + c;
  b:= a * -1;
  writeln(a:10:3,b:20:1);
  writeln('presione Enter');readln;
END.
```



El formato **a:10:3** indica que mostrar el valor de *a* en 10 lugares con 3 decimales.

a	b
?	?
1.5	?
2.5	?
2.5	7,3
2.5	-2.5

Observación: en el horario de práctica se explicarán más detalles sobre estas primitivas (no se lo pierda).

EQUIVALENCIAS Y DIFERENCIAS

- WRITE: muestra valores en la pantalla
- WRITELN: muestra valores en pantalla y baja de línea (LN)

```
WRITE(1);
WRITELN;
```

mismo efecto

```
WRITELN(1);
```

```
WRITE('ho');
WRITE('la ');
WRITE('che!');
WRITELN;
```

mismo efecto

```
WRITELN('hola che!');
```

- **Observación**: en el horario de la práctica se explicarán más detalles sobre estas primitivas (no se lo pierda).

PRIMITIVA PARA LA LECTURA O INGRESO DE DATOS

- READ: obtiene (lee) valores ingresados por teclado
- READLN: (read-line) idem a read pero espera por un ENTER
- Ambas tienen como argumentos una o varias variables que pueden ser de diferentes tipos.

```
…
VAR
  cant_ventanas:integer;
  ancho,largo: real;
BEGIN
…
  READ(cant_ventanas);
  READLN(ancho);
  READ(largo, ancho, cant_ventanas)
…
```

```
READ(algo);
READLN;
```

mismo efecto

```
READLN(algo);
```


IMPLEMENTACION DEL ALGORITMO "BOTELLAS"

- A continuación se incluye la **implementación en Pascal** del algoritmo desarrollado antes para calcular la cantidad de botellas a comprar para una reunión de personas.
- Se incluye además en forma de tabla la traza de los cambios en los valores de las variables para tres casos de prueba.
  - **Caso de prueba 1:** una reunión con 10 personas y comprando botellas de 2 litros y cuarto.
  - **Caso de prueba 2:** una reunión con 10 personas y comprando botellas de 1 litro.
  - **Caso de prueba 3:** una reunión con 4 personas y comprando botellas de 1 litro.
- Observe que los casos 1 y 2 tienen la misma cantidad de personas y cambia el volumen de la botella.
- Los casos 2 y 3 mismo volumen y cambian personas.
- Además, el 3 da entero el calculo intermedio.

IMPLEMENTACION DEL ALGORITMO "BOTELLAS"

```
PROGRAM botellas;
{Calcula la cantidad de botellas a comprar para una reunión,
considerando un consumo de 1.25 lit. por persona}
CONST
  litros_por_persona = 1.25;
VAR
  cant_botellas, personas: INTEGER;
  comprar, volumen: real;
BEGIN
 write('Cantidad de Personas?');
 readln (personas);
 write('Volumen de las botellas? (litros)');
 readln (volumen);
 comprar := (personas * litros_por_persona)/ volumen;
 cant_botellas := trunc(comprar) + 1;
 writeln('La sugerencia es comprar ', cant_botellas,
         'botellas de ', volumen:0:2, 'litros');
 writeln ('Pulse ENTER para finalizar');
 readln;
END.
```

TRAZA PARA UN CASO DE PRUEBA

- Caso de prueba 1: una reunión con 10 personas y comprando botellas de 2 litros y cuarto.
- En este caso, se espera que el programa sugiera 6 botellas, que es el entero siguiente a 5.55.

Traza de los valores almacenados en memoria para cada variable del programa "botellas":

Personas	Volumen	Comprar	Cant_botellas
?	?	?	?
10	?	?	?
10	2.25	?	?
10	2.25	5.55	?
10	2.25	5.55	6

TRAZA PARA UN CASO DE PRUEBA

- Caso de prueba 2: una reunión con 10 personas y comprando botellas de 1 litro.
- En este caso, se espera que el programa sugiera 13 botellas, que es el entero siguiente a 12.5.

Traza de los valores almacenados en memoria para cada variable del programa "botellas":

Personas	Volumen	Comprar	Cant_botellas
?	?	?	?
10	?	?	?
10	1	?	?
10	1	12.5	?
10	1	12.5	13

TRAZA PARA UN CASO DE PRUEBA

- Caso de prueba 3: una reunión con 4 personas y comprando botellas de 1 litro.
- En este caso, se espera que el programa sugiera 6 botellas, que es el entero siguiente a 5. Observar que aunque el resultado ya era un número entero, se compra una botella más.

Traza de los valores almacenados en memoria para cada variable del programa "botellas":

Personas	Volumen	Comprar	Cant_botellas
?	?	?	?
4	?	?	?
4	1	?	?
4	1	5	?
4	1	5	6

Resolución de Problemas y Algoritmos - 2016 55