



## Resolución de Problemas y Algoritmos


### Clase 2 Lenguaje de Programación Pascal: datos, tipos, asignaciones y expresiones





**Dr. Alejandro J. García**

<http://cs.uns.edu.ar/~ajg>




Departamento de Ciencias e Ingeniería de la Computación  
Universidad Nacional del Sur  
Bahía Blanca - Argentina

### Conceptos de la clase pasada

Conceptos de la clase anterior:

- Computadora
- Algoritmo
- Primitiva
- Trazo de un algoritmo

¿Preguntas?



Resolución de Problemas y Algoritmos Dr. Alejandro J. García 2

### Herramientas a disposición de los alumnos

Herramientas brindadas por la cátedra:

1. clase teórica
2. clase práctica
3. ejercicios en los prácticos
4. horarios de laboratorio
5. evaluación en máquina y parciales (obligatorio)

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 3

### Objetivos de la materia

El objetivo principal es que los alumnos adquieran la capacidad de desarrollar programas de computadoras para resolver problemas de pequeña escala.

El desarrollo de un programa se concibe como un proceso que abarca varias etapas:

1. La interpretación adecuada del enunciado a través del cual se plantea el problema.
2. El diseño de un algoritmo que modela la resolución del problema.
3. La implementación del algoritmo en un lenguaje de programación imperativo.
4. La verificación de la solución.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 4

### Etapas

El objetivo principal es que los alumnos adquieran la capacidad de desarrollar programas de computadoras para resolver problemas de pequeña escala.

El desarrollo de un programa se concibe como un proceso que abarca varias etapas:

1. La interpretación adecuada del enunciado a través del cual se plantea el problema.
2. El diseño de un algoritmo que modela la resolución del problema.
3. La implementación del algoritmo en un lenguaje de programación imperativo.
4. La verificación de la solución.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 5

### Metodología general propuesta

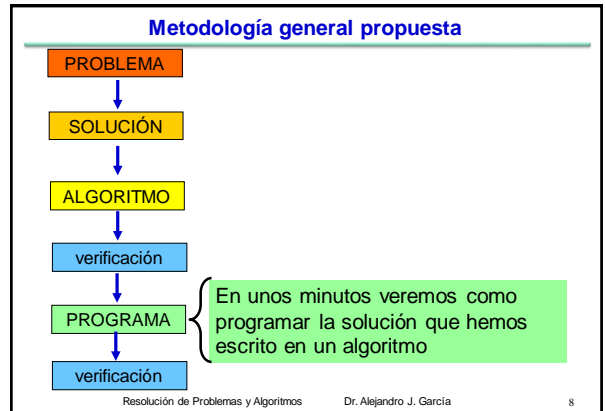
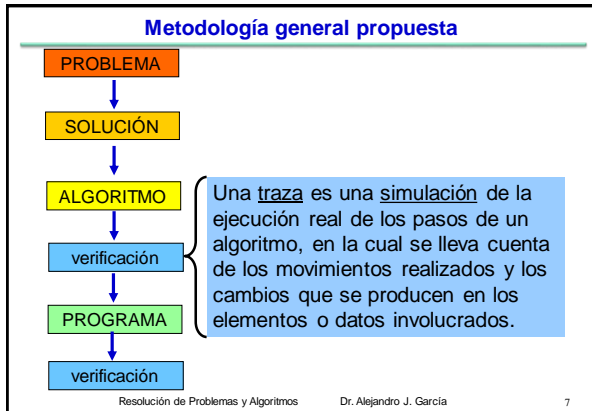


**Definición:** Un *algoritmo* es una secuencia de pasos u operaciones, que cuando se los ejecuta, producirá el resultado esperado y terminará luego de una cantidad finita de tiempo.

Cada paso debe estar definido sin ambigüedad, y las operaciones deben ser comprensibles por el que las ejecutará. Debe haber un único punto de comienzo y al menos un punto final.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 6

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente: **“Resolución de Problemas y Algoritmos. Notas de Clase”. Alejandro J. García. Universidad Nacional del Sur. (c)1998-2014.**



### Datos constantes, datos variables y expresiones

- En general, en los algoritmos hay **datos** con los cuales se obtiene un resultado.
- Estos datos pueden ser:
  - **constantes** (no cambian) o
  - **variables** (cambian su valor).
- Los algoritmos pueden usar acciones (primitivas) para modificar datos variables.  
(¿Qué datos constantes o variables había en los problemas realizados en la clase anterior?)

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 9

### Problema propuesto: calcular pintura

Se ha construido un edificio con aulas nuevas. Se desea tener una aplicación que calcule cuántos litros de pintura se necesitan para pintar las paredes de cualquiera de las aulas. Las aulas son rectangulares y tienen diferente largo y ancho. Sin embargo, todas tienen la misma altura (2,60m) y dos puertas iguales de 1,60m x 2m. Aunque las ventanas son todas iguales (1m x 2m), cada aula tiene una cantidad diferente. La pintura viene en latas de 4 litros y cubre por litro 8m<sup>2</sup>.

- Identificar incógnita, datos constantes y variables.
- Dividir el problema en partes
- Hacer un ejemplo con valores particulares

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 10

### Identificar incógnita, datos constantes y variables.

Se ha construido un edificio con aulas nuevas. Se desea tener una aplicación que calcule cuántos litros de pintura se necesitan para pintar las paredes de cualquiera de las aulas. Las aulas son rectangulares y tienen diferente largo y ancho. Sin embargo, todas tienen la misma altura (2,60m) y dos puertas iguales de 1,60m x 2m. Aunque las ventanas son todas iguales (1m x 2m), cada aula tiene una cantidad diferente. La pintura viene en latas de 4 litros y cubre por litro 8m<sup>2</sup>.

- Incongnita:** ¿Cuántos litros de pintura para un aula?
- Datos constantes:** alto = 2,60m; puerta = 3,20m<sup>2</sup>; ventanas = 2m<sup>2</sup>; cubrelitro = 8m<sup>2</sup>; litros lata = 4
- Datos variables:** ancho, largo y cant\_ventanas

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 11

### Dividir el problema en partes (P)

- Datos constantes:** alto = 2,60m; puerta = 3,20m<sup>2</sup>; ventanas = 2m<sup>2</sup>; cubrelitro = 8m<sup>2</sup>; litros lata = 4
- Datos variables:** ancho, largo y cant\_ventanas
- Incongnita:** ¿Cuántos litros?
- Diseño (dividido en partes):**
  - calcular la superficie total (con puertas, etc.)
  - calcular superficie a no pintar (depende de la cantidad de ventanas)
  - calcular superficie a pintar (total menos lo que no se pinta)
  - calcular cuantos litros se necesitan.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 12

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente: "Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c)1998-2014.

### Algoritmo general (p)

- **Datos constantes:** alto = 2,60m; puerta = 3,20m<sup>2</sup> ; ventana = 2m<sup>2</sup>; cubrelitro = 8m<sup>2</sup>; litros lata = 4
- **Datos variables:** ancho; largo y cant\_ventanas

**Algoritmo general:**

- el total es  $2 \times (\text{ancho} \times \text{alto}) + 2 \times (\text{largo} \times \text{alto})$
- no\_pintar es  $2 \times \text{puerta} + \text{ventana} \times \text{cant\_ventanas}$
- sup\_a\_pintar es total - no\_pintar
- cant\_litros es  $\text{sup\_a\_pintar} / \text{cubrelitro}$

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 13

### Verificar con una traza y un ejemplo (p)

- **Datos constantes:** alto = 2,60m; puerta = 3,20m<sup>2</sup> ; ventanas = 2m<sup>2</sup>; cubrelitro = 8m<sup>2</sup>; litros lata = 4
- **(Ejemplo)** valores particulares para las variables:
- ancho = 5 largo = 10 y cant\_ventanas = 2
- **Con estos valores:**
  - Total es  $2 \times (\text{ancho} \times \text{alto}) + 2 \times (\text{largo} \times \text{alto}) = 2 \times (5 \times 2,60) + 2 \times (10 \times 2,60) = 26 + 52 = 78$
  - a no pintar es  $6,40 + 2 \times 2 = 10,40$
  - a pintar es  $78 - 10,40 = 67,60$
  - litros a usar  $67,60 / 8 = 8,45$


Resolución de Problemas y Algoritmos Dr. Alejandro J. García 14

### El lenguaje de programación Pascal

El lenguaje de programación **Pascal** fue creado en 1969 por el científico de la computación **Niklaus Wirth** (1934-).

Fue llamado así en honor al matemático Blas Pascal (1623-1662) uno de los pioneros de la computación.

Wirth también fue creador de Algol-w, Modula, Euler y Oberon, y ganador del premio Turing en 1984 (foto).



Wirth en 1984

Biblioteca Central de la UNS: <http://bc.uns.edu.ar>

1 - Reporte Original de Jensen y Wirth  
 2 - "Programación en Pascal" de Peter Grogono (1986)  
 En línea: [http://en.wikipedia.org/wiki/Pascal\\_language](http://en.wikipedia.org/wiki/Pascal_language)

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 15

### Concepto: lenguaje de programación

Un **lenguaje de programación** es un lenguaje artificial creado para expresar procesos que pueden ser llevados a cabo por computadoras.

Está definido por un conjunto de **símbolos** y **reglas sintácticas y semánticas** que definen su estructura y el significado de sus elementos.

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.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 16

### Conceptos: programa – código fuente

Un **programa de computadoras** (o simplemente **programa**) es una secuencia de instrucciones escritas en un *lenguaje de programación*.

El texto de un programa se conoce como **código fuente**.

Al proceso por el cual se escribe, se prueba, se depura, y se mantiene el código fuente de un programa de computadoras se lo llama **programación**.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 17

### Ejemplo muy simple

**PROBLEMA**

↓

**SOLUCIÓN**

↓

**ALGORITMO**

↓

**PROGRAMA**  
en algún lenguaje de programación (por ejemplo: Pascal)

**Ejemplo: Calcule el área de un círculo**

Usar la fórmula "Pi por radio al cuadrado"

Algoritmo ÁreaCírculo  
Área es 3.1416 x Radio x Radio

```

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

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 18

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:  
 "Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c)1998-2014.

### Algunos elementos de un programa en Pascal

```

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

Resolución de Problemas y Algoritmos Dr. Alejandro J. García

### Pascal: Palabras Reservadas

Las **palabras reservadas** son aquellas que ya tienen un significado en el lenguaje Pascal, y el programador sólo puede usarlas con ese significado.

Algunos ejemplos:

```

PROGRAM
CONST
VAR
BEGIN
END
    
```

**Importante:**  
no afecta si usamos mayúsculas o minúsculas.  
Ej: **PROGRAM**, **proGRAM**, y **program** son la misma palabra reservada.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 20

### Pascal: Identificadores definidos por el programador

Los **identificadores** son nombres que identifican a elementos creados por el programador. No pueden ser igual a una palabra reservada. Deben comenzar obligatoriamente con una letra, y sólo involucran letras, números y el guión bajo “\_” (underscore)

<p><b>Son válidos:</b> Radio Pi x23 es_nro_par UNprogramA SueldoNeto</p>	<p><b>No son válidos:</b> La cantidad program %mas 23i es-nro-par Primo(i)</p>	<p><b>Importante:</b> no afecta si usamos mayúsculas o minúsculas. Ej: CANTIDAD, canTIDAD, y CaNtIdAd son el mismo identificador.</p>
--	--	---

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 21

### Pascal: Variables y constantes (P)

**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 litros: REAL;**

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 22

### Pascal: tipos de datos predefinidos

**Tipo de Dato:** define el **conjunto de valores posibles que puede tomar una variable, y también define las operaciones que puede usarse.**

**Tipo de dato predefinido: INTEGER**  
Es un subconjunto de los números enteros.  
Operaciones: + - \* div (y otras que mostraremos luego)

**Tipo de dato predefinido: REAL**  
Es un subconjunto de los números reales.  
Se usa punto para separar la parte entera de la decimal.  
Ejemplos: 3.1416 0.00001 128.5  
Operaciones: + - \* / (y otras que mostraremos luego)

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 23

### Declaración de constantes y variables en Pascal (I)

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 “=”

```

CONST Pi = 3.1416;
      e = 2.718281828;
    
```

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

**Declaración de variables:** se escribe la palabra reservada **VAR**, y luego **nombres** y **tipos de dato** de cada variable usando el símbolo “:”

```

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

Puedo declarar varias variables del mismo tipo separándolas con coma, y uso punto y coma luego del tipo.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 24

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente: **“Resolución de Problemas y Algoritmos. Notas de Clase”. Alejandro J. García. Universidad Nacional del Sur. (c)1998-2014.**

### Constantes, Variables y Tipos

```
PROGRAM pintura_aula;
CONST alto = 2.60; cubrelitro = 8;
VAR cant_ventanas:INTEGER; largo: REAL;
BEGIN
END.
```

**Valores fijos (y tipo fijo)**

**Tipo de valores que puede tomar**

- 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 (sólo se indica que tipo de valores puede tener)
- Muy importante:** 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.
- Comience a construir el programa siguiendo el algoritmo desarrollado para "calcular pintura".

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 25

### Ejemplo propuesto

- Datos constantes:** alto = 2,60m; puerta = 3,20m<sup>2</sup> ; area\_ventana = 2m<sup>2</sup>; cubrelitro = 8m<sup>2</sup>;
- Datos variables:** ancho; largo (número real) y cant\_ventanas (número entero)

```
PROGRAM pintura_aula;
CONST alto = 2.60; puerta = 3.20; area_ventana = 2;
      cubrelitro = 8;
VAR cant_ventanas:INTEGER;
      ancho, largo: REAL;
BEGIN
END.
```

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

**Puedo declarar varias variables del mismo tipo, separándolas con coma**

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 26

### Pascal: primitiva de Asignación

Para dar un valor inicial o cambiar el valor de una variable, Pascal tiene una **primitiva** llamada **asignación**.

- Se expresa con el símbolo **:=** que se suele leer "dos puntos igual", y está formado por dos caracteres ":" seguido "=".
- A la **izquierda** del := debe ir un **obligatoriamente un identificador de variable** y a la **derecha una expresión**.

```
VAR ancho: REAL;

      ancho := 5.23
```

- Permite darle el valor "5.23" a la variable ancho y se lee "a la variable ancho le asigno el valor 5.23"

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 27

### Primitiva de asignación

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?"

En una asignación: **variable := expresión**

- primero** se evalúa la **expresión** se obtiene un valor,
- y luego** se modifica el valor de la **variable**

Un dato sin valor a la derecha de ":= " es un **ERROR**

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 28

### Pascal: Primitiva de Asignación

- Para separar asignaciones en una secuencia se utiliza el símbolo **;** (punto y coma)
- Por ejemplo si un programa tiene las variables:

```
VAR ancho, largo, total: REAL;
```

- Se puede escribir la siguiente secuencia de asignaciones que se ejecutará de arriba hacia abajo:

```
ancho := 5;
largo:= 10;
total := 2*(ancho*2.60) + 2*(largo*2.60)
```

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 29

### Programa en Pascal para el algoritmo propuesto

```
PROGRAM pintura_aula;
CONST alto = 2.60; puerta = 3.20; area_ventana = 2;
      cubrelitro = 8; litrosLata = 4;
VAR ancho, largo, total, a_no_pintar, a_pintar: REAL;
      cant_litros:REAL; cant_ventanas : INTEGER;
BEGIN
ancho := 5; largo:= 10;
cant_ventanas :=2;
total := 2*(ancho*alto)+2*(largo*alto);
a_no_pintar:= 2*puerta + area_ventana*cant_ventanas ;
a_pintar := total - a_no_pintar;
cant_litros:= a_pintar / cubrelitro;
END.
```

**Traza por favor**

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 30

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente: "Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c)1998-2014.

### Pascal: primitiva de Asignación

**<identificador de variable> := <expresión compatible>**

- 1) **primero** se evalúa la **expresión** de la derecha de :=, se obtiene un valor,
- 2) **y luego** se modifica el valor de la **variable** a la izquierda del símbolo := (perdiéndose el viejo valor)

**VAR** saldo, consumo: integer;

```

saldo := 10;
consumo := 8;
Saldo := saldo - consumo;
Saldo := saldo + 30;
        
```

El **tipo** del resultado de la **expresión** tiene que ser compatible con el tipo de la variable que se quiere modificar (esto se verá en detalle en otra clase).

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 31

### Pascal: primitiva de asignación

- Si un dato aparece a la **izquierda** del símbolo “:=” el valor que contiene el dato se **modificará**, perdiéndose definitivamente el anterior.
- Si un dato aparece a la **derecha** de “:=” el valor que contiene el dato se **utiliza** para calcular el resultado de la expresión (no se modifica). Error: no tiene valor
- Un dato sin valor a la derecha de “:=” es un **ERROR**.
- Realice una traza de estas dos secuencias.

```

PROGRAM pintura_aula;
VAR saldo,carga:INTEGER;
BEGIN
saldo := 10; carga := 30;
Saldo := saldo + carga
END.
        
```

```

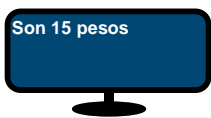
PROGRAM pintura_aula;
VAR saldo,gasto:INTEGER;
BEGIN
saldo := 10;
Saldo := saldo - gasto
END.
        
```

**¡MAL!**

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 32


### 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) ©

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 33

### 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;
        
```

↔  
 m. efecto

```

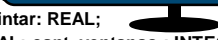
WRITELN('hola che !');
        
```

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 34

```

PROGRAM pintura_aula;
CONST alto = 2.60;
cubrelitro=8; puerta=3.20;
VAR ancho, largo, a_no_pintar: REAL;
cant_litros , a_pintar :REAL; cant_ventanas : INTEGER;
BEGIN
ancho := 5; largo:= 10; cant_ventanas :=2;
writeln(' Para un aula de ', largo, 'x', ancho,' con ',
cant_ventanas,' ventanas');
a_no_pintar:= 2*puerta + 2 * cant_ventanas ;
a_pintar := 2*(ancho*alto)+2*(largo*alto)-a_no_pintar;
Writeln( 'voy a pintar: ', a_pintar,' m2');
cant_litros:= a_pintar / cubrelitro;
Write(' Litros a usar: '); writeln(cant_litros);
END.
        
```

Para un aula de 10 x 5 con 2 ventanas voy a pintar 67.60 m2  
Litros a usar: 8.45



Resolución de Problemas y Algoritmos Dr. Alejandro J. García 35

### 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;
READ(cant_ventanas);
READLN(ancho);
READ(largo,ancho,cant_ventanas)
        
```

```

READ(algo);
READLN;
        
```

↔  
 mismo efecto

```

READLN(algo);
        
```

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 36

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:  
 “Resolución de Problemas y Algoritmos. Notas de Clase”. Alejandro J. García. Universidad Nacional del Sur. (c)1998-2014.

```
PROGRAM pintura_aula;
CONST alto = 3; puerta = 3.20;
ventanas = 2; cubrelitro = 8;
VAR
  ancho, largo, a_no_pintar: REAL;
  cant_litros , a_pintar :REAL;
  cant_ventanas:INTEGER;
BEGIN
  writeln('Ingrese ancho del aula');
  readln(ancho);
  writeln('Ingrese largo del aula'); readln(largo);
  writeln('¿Cuántas ventanas?'); readln(cant_ventanas);
  a_no_pintar:= 2*puerta + 2 * cant_ventanas ;
  a_pintar := 2*(ancho*alto)+2*(largo*alto)-a_no_pintar;
  cant_litros:= a_pintar / cubrelitro;
  writeln('Litros a usar =', cant_litros);
END.
```

Ingrese ancho del aula  
5  
Ingrese largo del aula  
10  
¿Cuántas ventanas?  
2  
Litros a usar = 8.45

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 37

Continuará

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 38

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:  
"Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c)1998-2014.