Resolución de Problemas y Algoritmos

Clase 2
Lenguaje de Programación Pascal: constantes, variables, tipos y asignaciones.

Niklaus Wirth

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

¿Preguntas?

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.

Metodología general propuesta

PROBLEMA
↓
SOLUCIÓN
↓
ALGORITMO
↓
verificación
↓
PROGRAMA
↓
verificación

En unos minutos veremos como programar la solución que hemos escrito en un algoritmo

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.

Problema propuesto: botellas

Crear una aplicación para calcular cuantas botellas de gaseosa comprar para una reunión. La aplicación debe interrogar al usuario como figura en el ejemplo, 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.

¿Cantidad de personas? 15
¿Volumen por botella? (litros) 2.25
Se sugiere comprar 9 botellas de 2.25 litros

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.

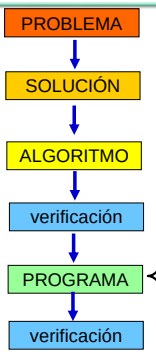
Algoritmo

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.

¿Cantidad de personas? 15
¿Volumen por botella? (litros) 2.25
Se sugiere comprar 9 botellas de 2.25 litros

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

Metodología general propuesta



En lo que resta la clase de hoy, veremos lo necesario como para implementar el algoritmo anterior en el lenguaje de programación Pascal.

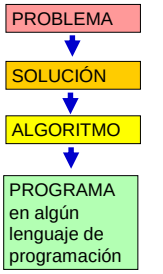
En las clases siguientes seguiremos viendo más elementos, conceptos y primitivas que ofrece el lenguaje Pascal para programar.

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: aplicación de cálculo área círculo



Calcule el área de un círculo

Usar la fórmula “Pi por radio al cuadrado”

Ejemplo muy simple

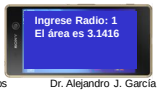
PROBLEMA → Ejemplo: Calcule el área de un círculo

SOLUCIÓN → Usar la fórmula "Pi por radio al cuadrado"

ALGORITMO →

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**

PROGRAMA en algún lenguaje de programación



Ejemplo muy simple

PROBLEMA → Ejemplo: Calcule el área de un círculo

SOLUCIÓN → Usar la fórmula "Pi por radio al cuadrado"

ALGORITMO → Algoritmo Área Círculo

PROGRAMA en el lenguaje 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

Dato constante

Datos variables

Tipo de datos predefinido

Palabras reservadas (vocabulario)

Primitivas predefinidas

Símbolos y operadores

```
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 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.

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.

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:
Radio
Pi
x23
es_nro_par
UNprogramA
SueldoNeto

No son válidos:
La cantidad
program
%mas
23i
es-nro-par
Primo(i)
área
año

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

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.

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.

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;

Dos 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.

Ejemplos:

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**

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

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 "**=**", 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.

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: **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 un 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 derecha y se obtiene un valor,
2) luego se **modifica el valor** de la variable, **perdiéndose el valor anterior.**

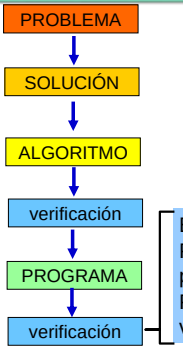
```
PROGRAM Asigna1;
CONST c = 2.3;
VAR a, b: REAL;
BEGIN
  a:= 1.5;
  a:= 2 + 1/2;
  b:= 2 * a + c;
  b:= a * -1;
END.
```

modifica el valor de a

usa el valor de a

modifica el valor de b

Metodología general propuesta



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 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

Primitivas predifinidas para interacción con el usuario

Procedimientos predefinidos de Pascal:

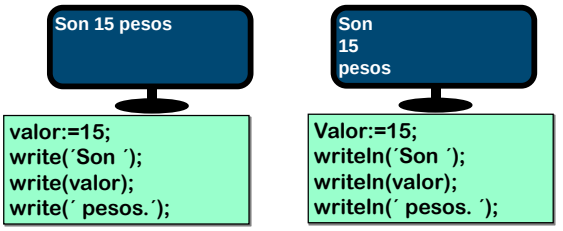
WRITE: muestra valores en la pantalla
WRITELN: muestra valores en pantalla y baja de línea (LN)

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)



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.
```

2.500000E+0002.500000E+000
presione Enter

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 (asi 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.
```

2.500 -2.5
presione Enter

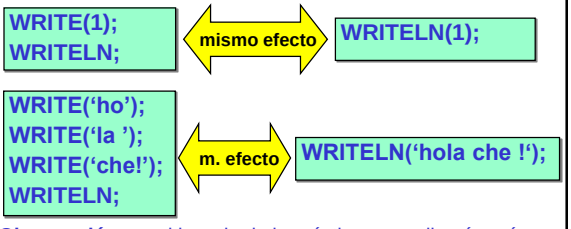
El formato a:10:3 indica que mostrar el valor de a en 10 lugares con 3 decimales.

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)



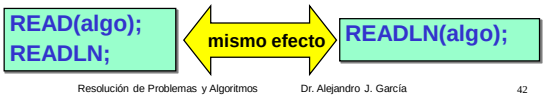
Observación: en el horario de la práctica se explicarán más detalles sobre estas primitivas (no se lo pierda) ☺

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



Implementación 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.

Implementación 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 otro 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 otro 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

Continuará

Información adicional

El lenguaje de programación Pascal

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

Wirth en 1984

Pascal fue concebido como un lenguaje pequeño y eficiente, con la intención de fomentar buenas prácticas de programación.

Actualmente ya casi no se lo usa en la industria del software. Sin embargo, Pascal sigue siendo elegido como primer lenguaje para enseñar programación imperativa, por su simpleza y fomentar buenos hábitos de programación.

El lenguaje fue llamado así en honor al matemático Blas Pascal (1623-1662) quien fue pionero de la computación.

Niklaus Wirth

Niklaus Wirth realizó una gran tarea como científico en computación y nuestra comunidad le debe mucho.

- Su artículo "*Desarrollo de un programa por refinamiento sucesivo"* es considerado un texto clásico en la ingeniería del software.
- Su libro: "*Algoritmos + Estructuras de datos = Programas"*, recibió un amplio reconocimiento.
- Fue el jefe de diseño de los lenguajes : Euler, Algol W, Pascal, Modula, Modula-2 y Oberon.
- Recibió el Premio Turing por el desarrollo de estos lenguajes de programación en 1984.
- Se jubiló en 1999. (¡Gracias Wirth!)

Información sobre Pascal

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)

Material en la página de la materia:
http://cs.uns.edu.ar/~wmg/rpalz/

En línea: http://en.wikipedia.org/wiki/Pascal_language