NÚMEROS EN ACCIÓN

Fernando Chamizo Lorente

Universidad Autónoma de Madrid

http://www.uam.es/fernando.chamizo

11 de noviembre de 2010

La ciencia para la sociedad

¿Qué nos viene a la mente cuando se mencionan algunas ciencias?

Biología

Física

$$= \int_1^n \frac{1}{x}\,dx = \sum_{j=1}^{n-1} \int_j^{j+1} \frac{1}{x}\,dx \leqslant \sum_{j=1}^{n-1} \int_j^{j+1} \frac{1}{j}\,dx$$

$$= \frac{(n+1)^2 (n!)^2}{(2n+2)(2n+1)(2n)!} x^{2n+2}$$

$$|g(a)| \quad f(x) = \ln\left(\sqrt{\frac{1+x}{1-x}}\right) \quad \varepsilon$$

Matemáticas

Química

Hay una serie de afirmaciones sobre las Matemáticas muy comunes entre los que son ajenos a ella.

1. Hay que hacer cuentas muy largas
2. No tienen aplicaciones prácticas
3. No queda nada por descubrir
4. Son aburridas

¿Hay que hacer cuentas muy largas?

En realidad en Matemáticas se buscan continuamente trucos para no hacerlas.

Las leyes generales que se aplican en multitud de situaciones se llaman teoremas.
Por ejemplo, hay teoremas (de teoría de grupos) que afirman al mismo tiempo cosas acerca del cubo de Rubik y de propiedades de divisibilidad.

¿Cuántas bolas coloreadas hay en el billar americano?

Sumar las horizontales del triángulo.

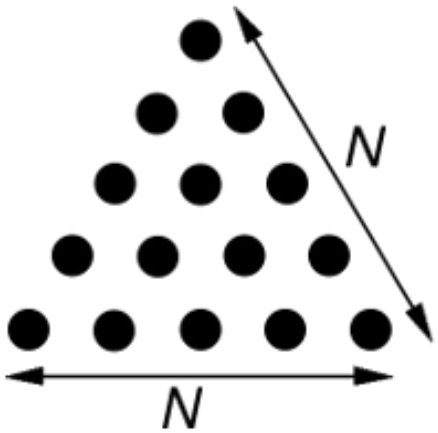
$$S = 1 + 2 + 3 + 4 + 5$$

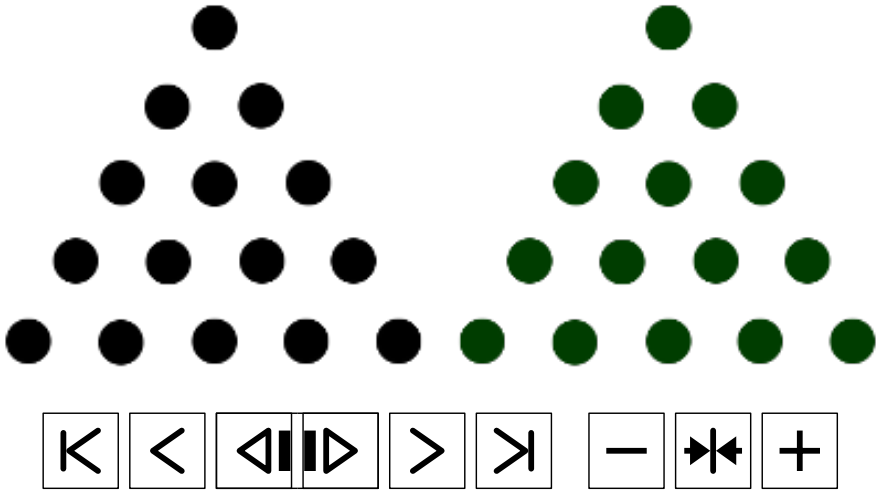
¿Cuántas bolas habría en un triángulo de lado arbitrario?

Abstracción: 5, 10, 100, $\dashrightarrow$ arbitrario$=N$

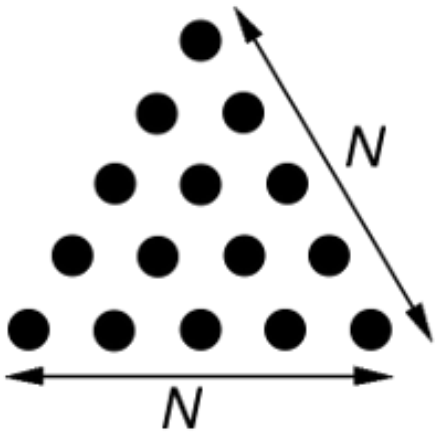
Resultado$=S$

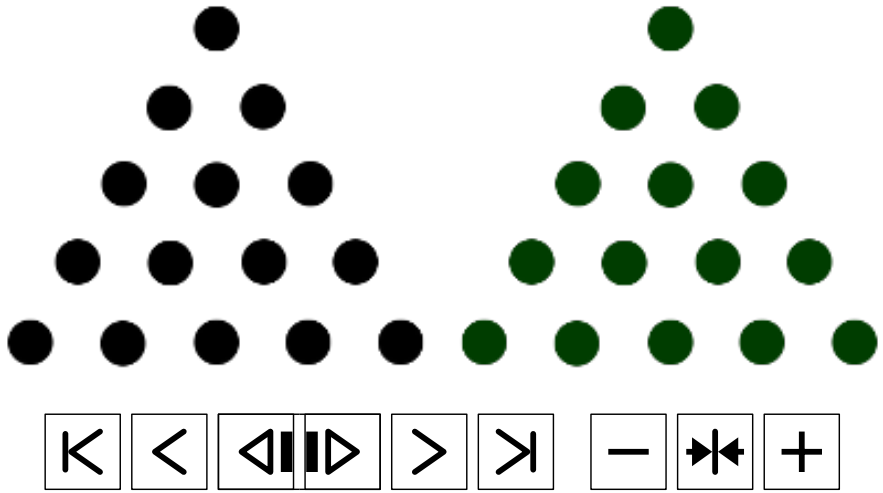
Primer método





Primer método

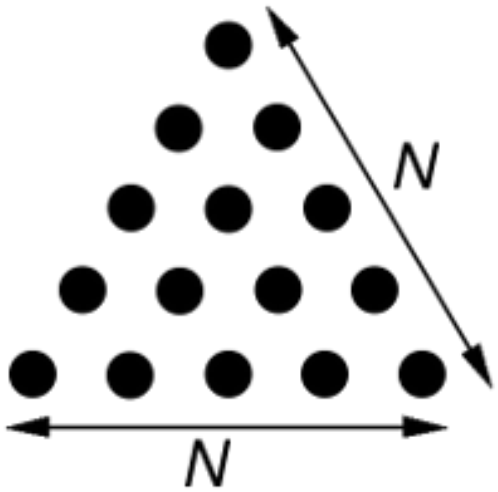


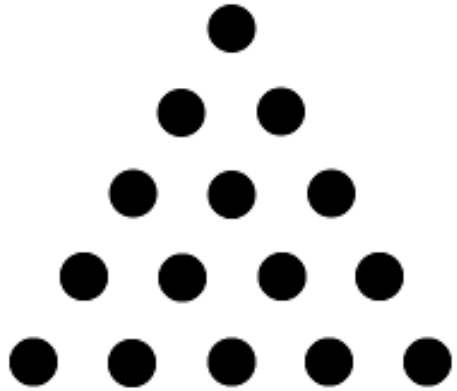


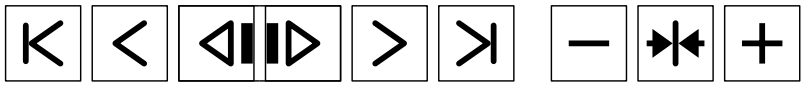
Contar en triángulo $\longleftrightarrow$ Contar en rectángulo (más fácil)

$$2S = (N+1)N \qquad \Longrightarrow \qquad \boxed{S = \frac{N(N+1)}{2}}$$

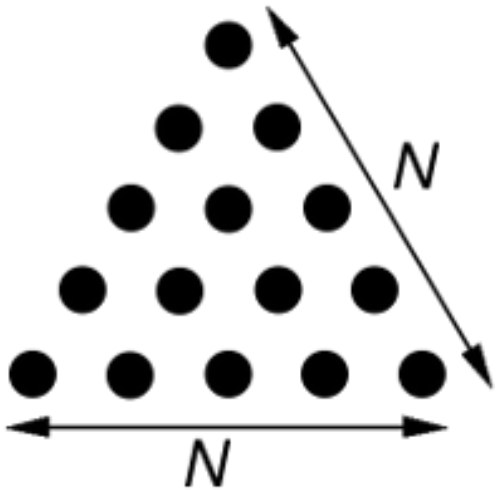
Segundo método

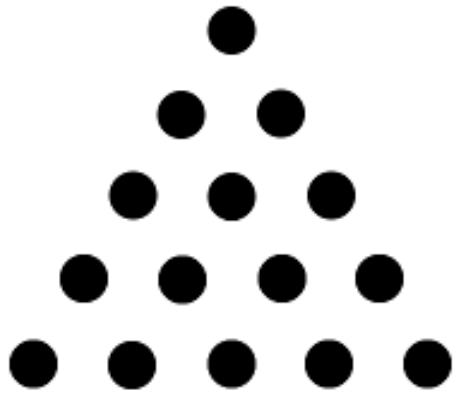


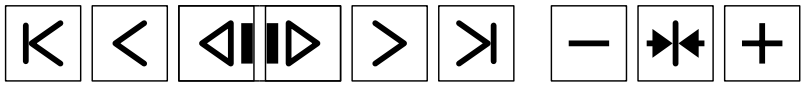




Segundo método







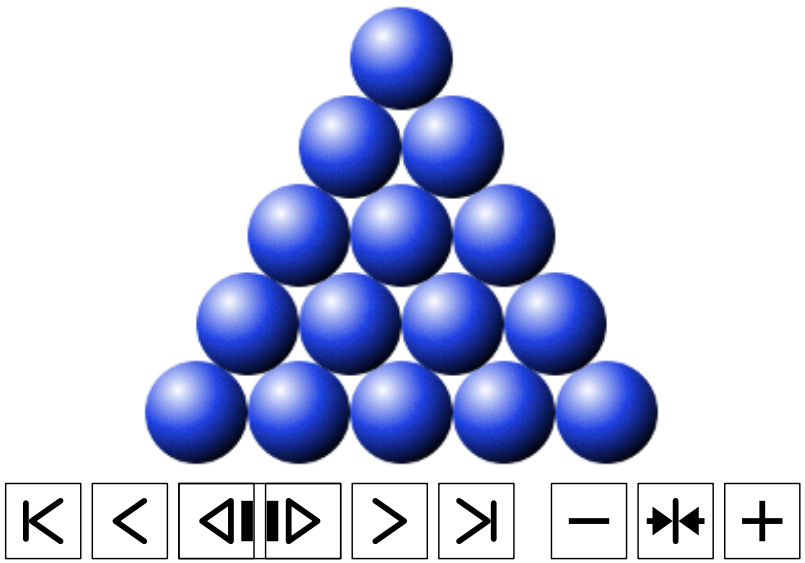
Contar en triángulo $\longleftrightarrow$ Contar parejas

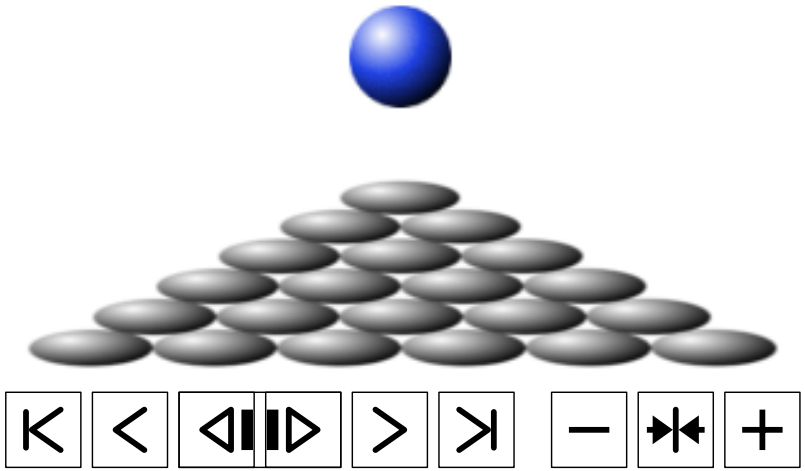
$$2S = (N+1)N \qquad \Longrightarrow \qquad \boxed{S = \frac{N(N+1)}{2}}$$

En tres dimensiones

¿Y si formamos una montaña triangular?

Hay que sumar unos cuantos triángulos





Contar en pirámide $\longleftrightarrow$ Contar triángulos equiláteros

Subtriángulos equiláteros de uno de lado $n+1$.

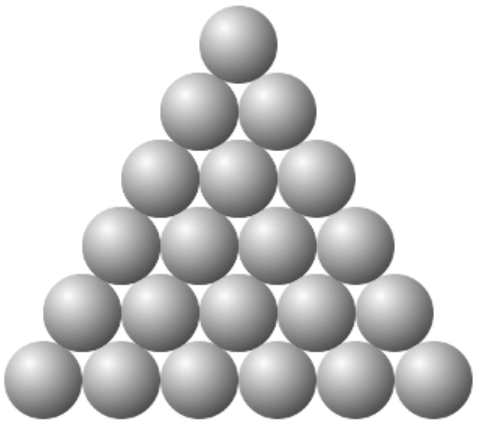
Contar triángulos equiláteros $\longleftrightarrow$ Contar ternas$\leq n+2$

$a < b < c \leq n+2 \longleftrightarrow$ a a la derecha, $b-a$ al noreste (vértice inferior derecho), $c-b$ otra vez al noreste (vértice superior) $c-b$ al sureste (vértice inferior izquierdo)

Formas de reordenar tres números: 6

$$6P = (N+2)(N+1)N \qquad \Longrightarrow \qquad \boxed{S = \frac{N(N+1)(N+2)}{6}}$$

$$\{a = 1, b = 2, c = 4\} \longrightarrow$$



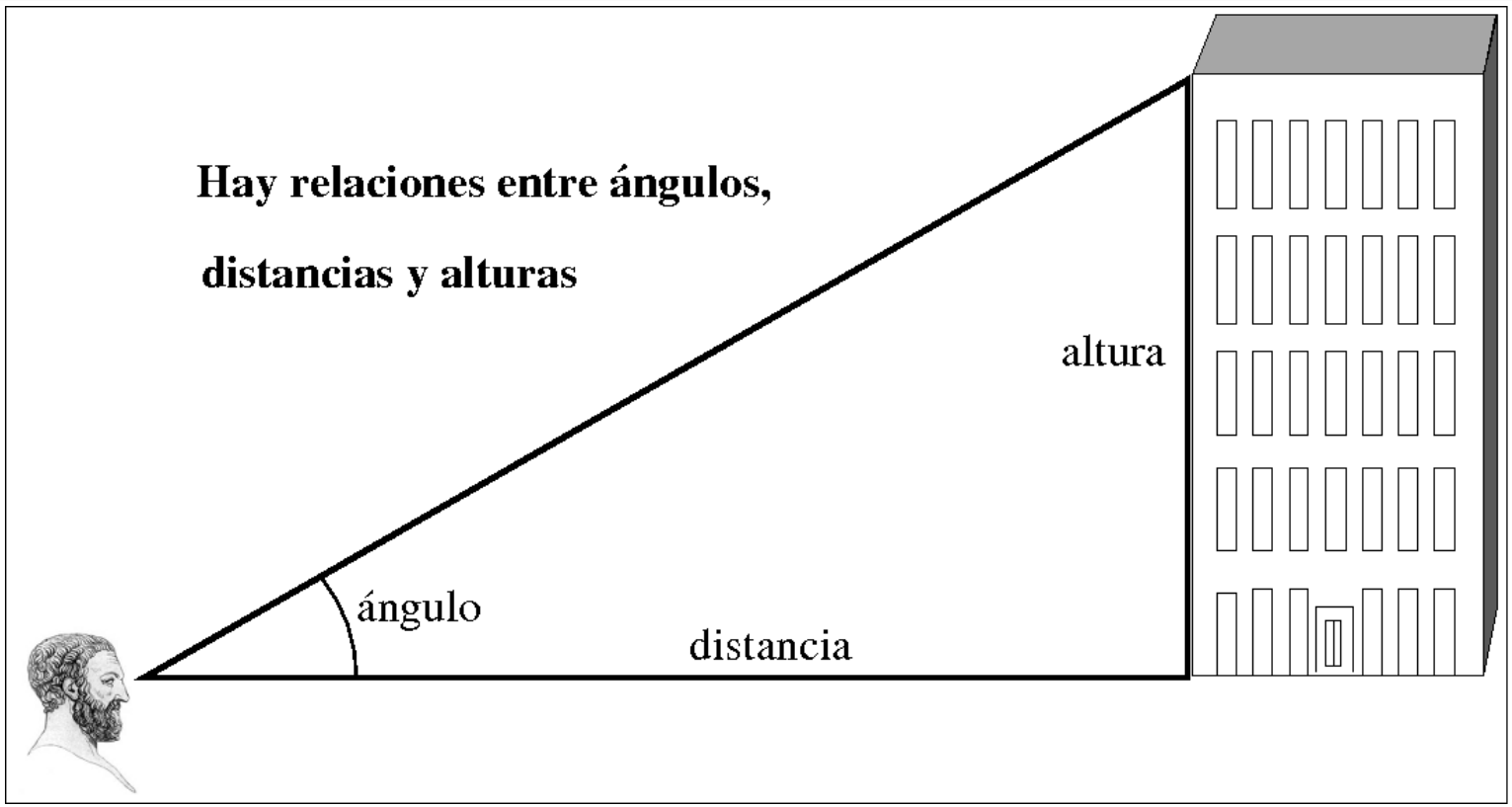
¿Las Matemáticas no tienen aplicaciones?

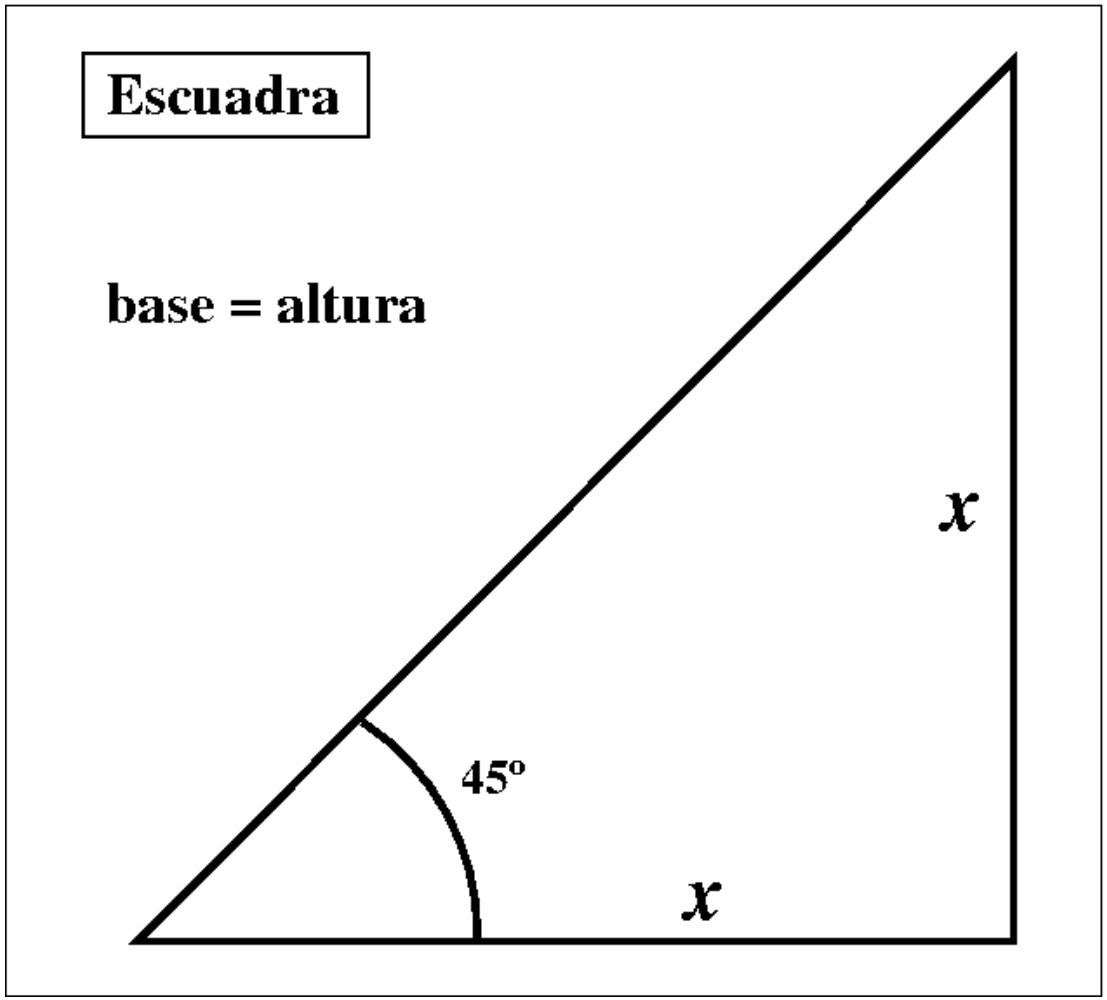
Si abrimos un libro de Física o de Ingeniería está lleno de fórmulas matemáticas.

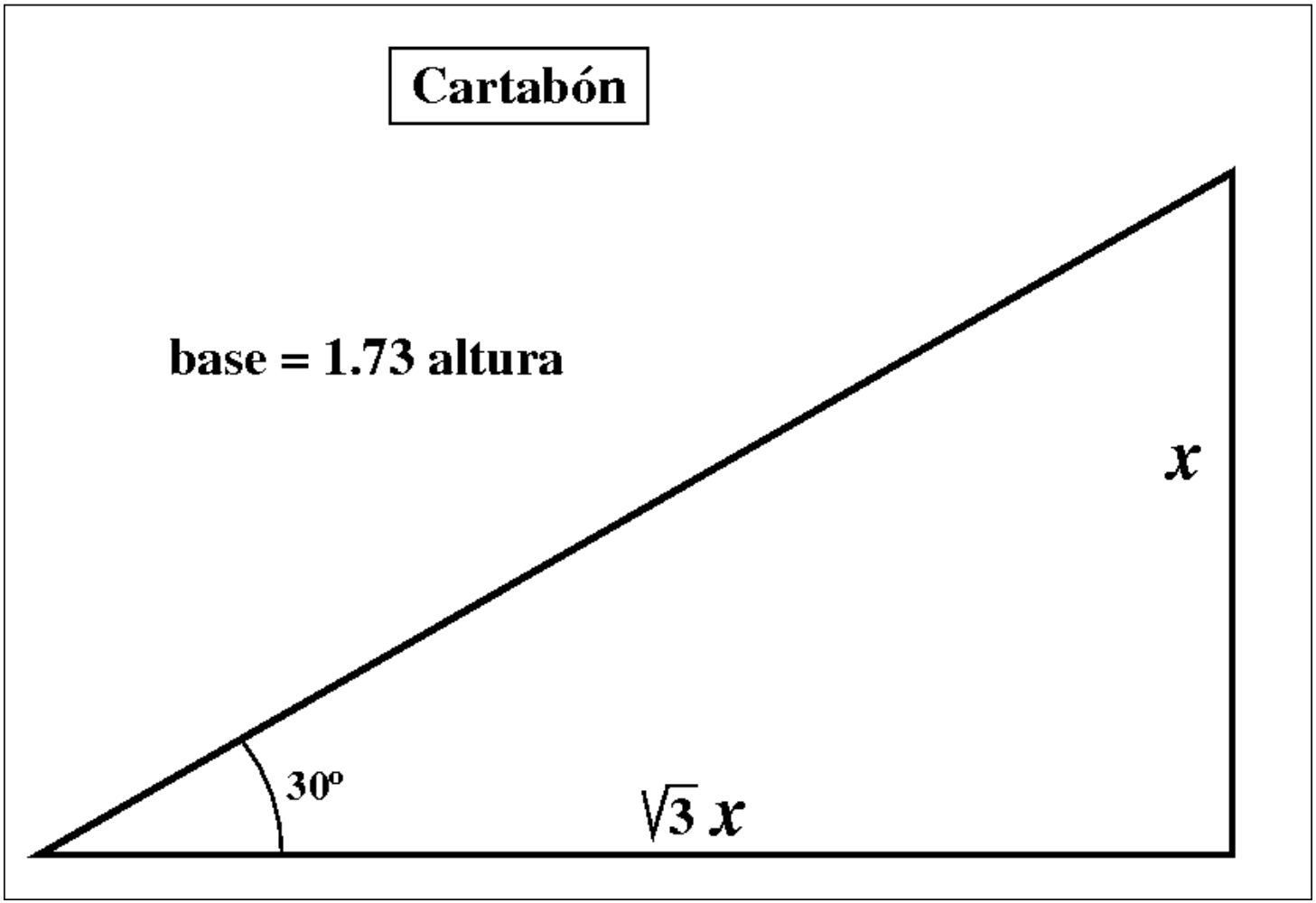
Hay aplicaciones simples (pero poderosas) de las Matemáticas, como la trigonometría que permite calcular la distancia a las estrellas, y otras mucho más profundas.

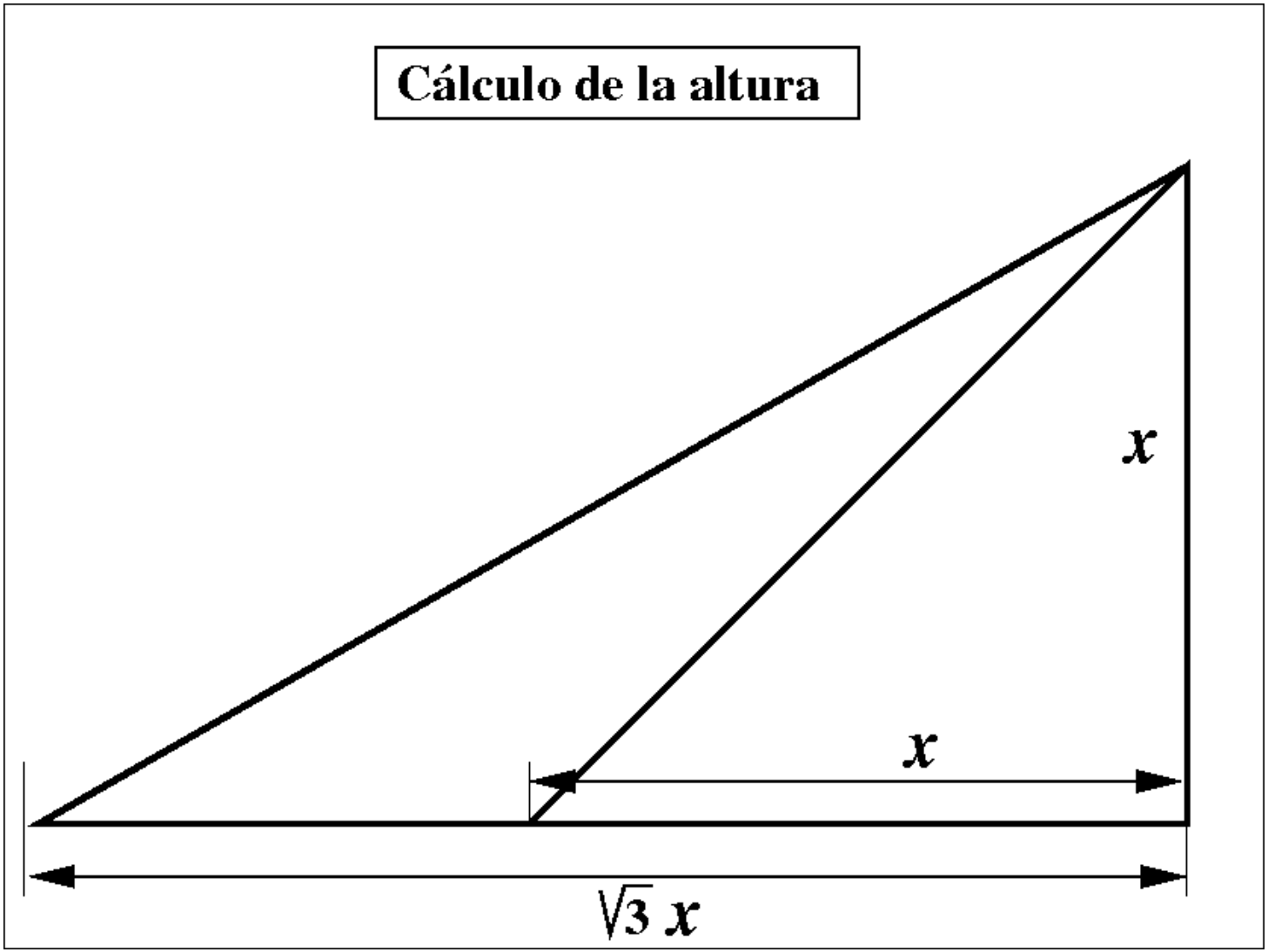
Las Matemáticas están en muchos artilugios cotidianos pero escondidas a los ojos del usuario.
Uno de los escondites preferidos de las Matemáticas es el *software*.
Se puede hacer fotos al *hardware* pero las líneas de código normalmente son secretas y además habitualmente no nos dirían demasiado.

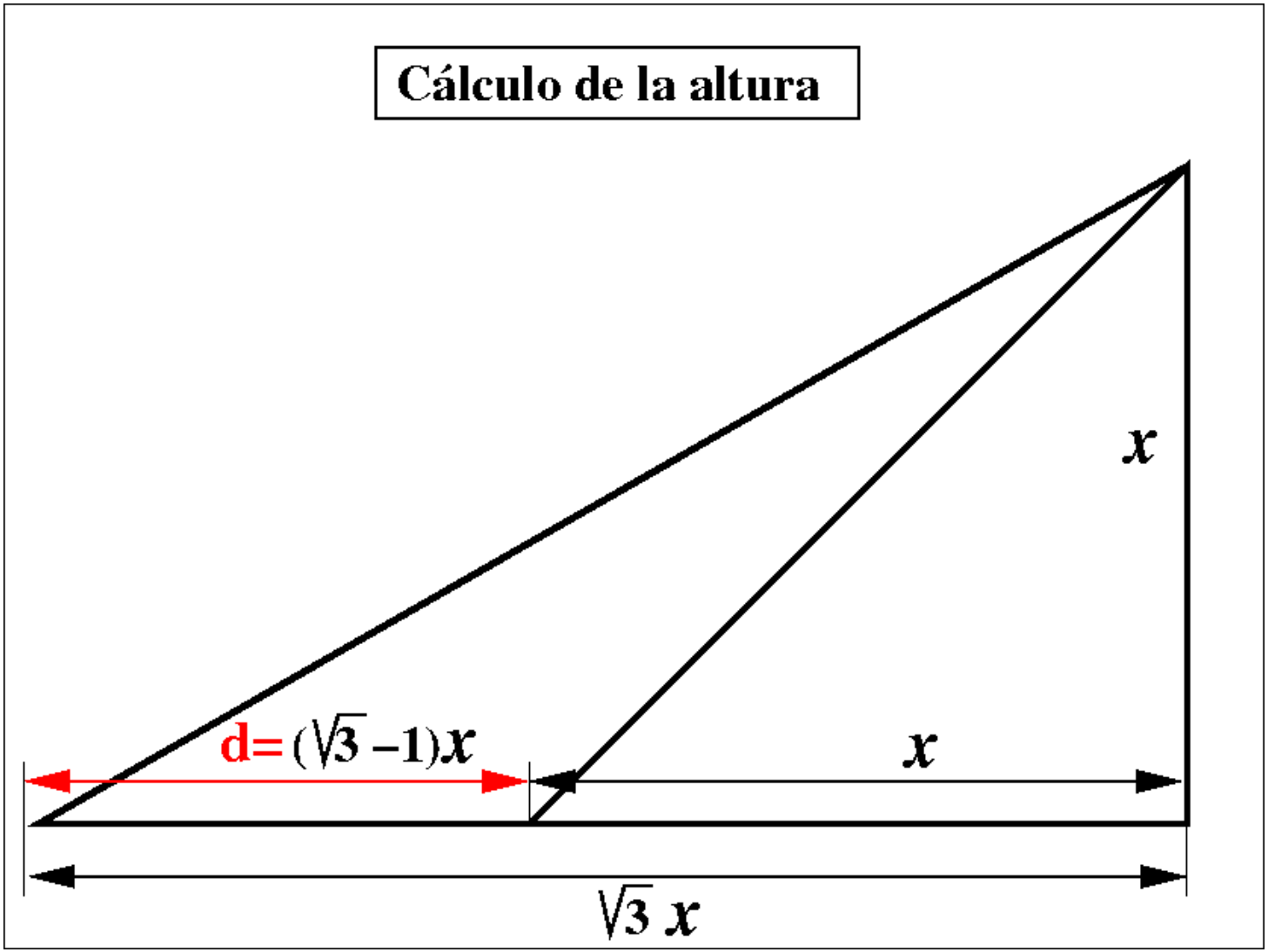
Los autores clásicos cuentan que Tales de Mileto (para muchos el primer matemático) utlilizó la geometía en el siglo VI a. C. para medir la altura de las pirámides y la distancia a los barcos desde la costa

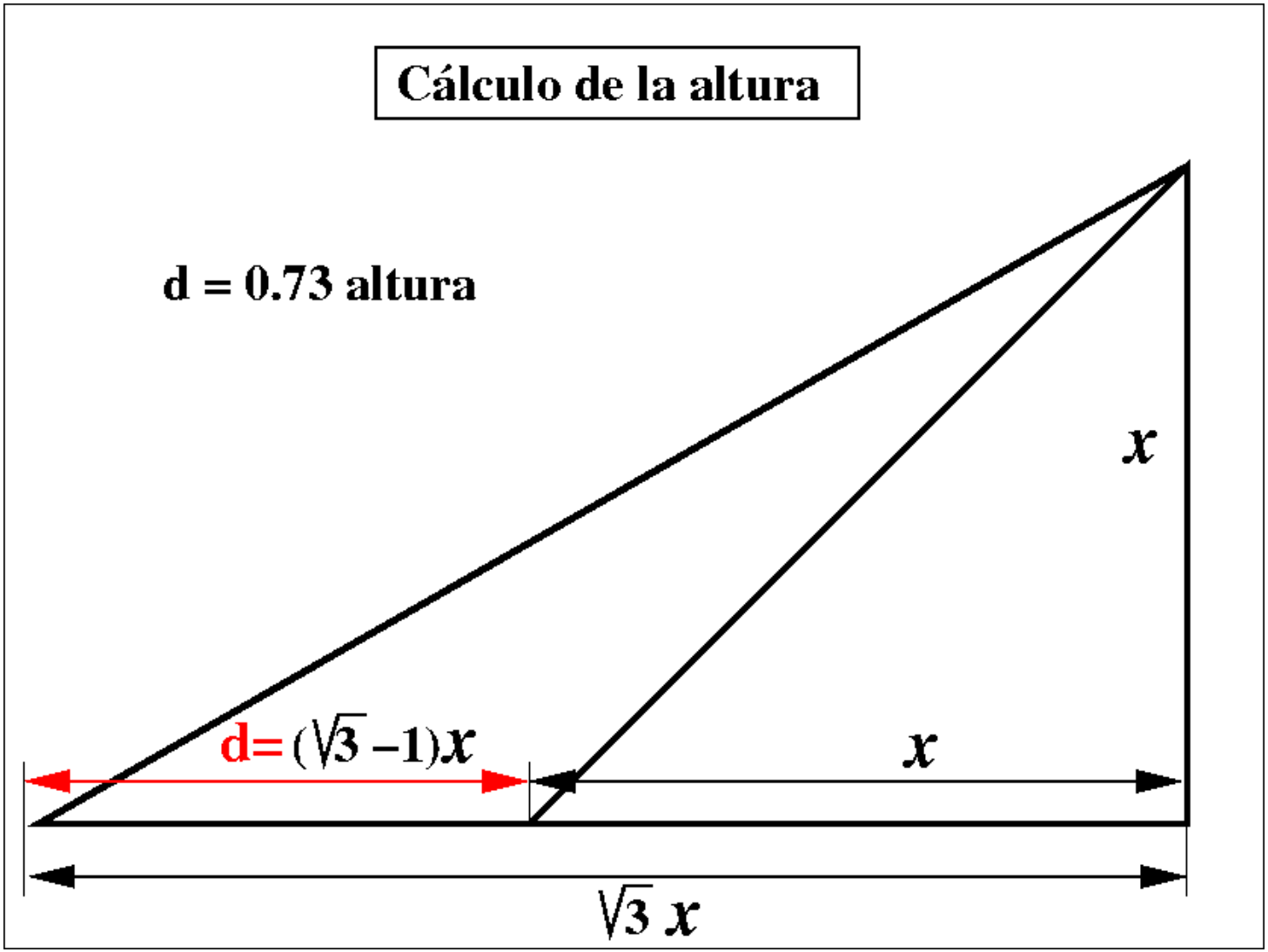


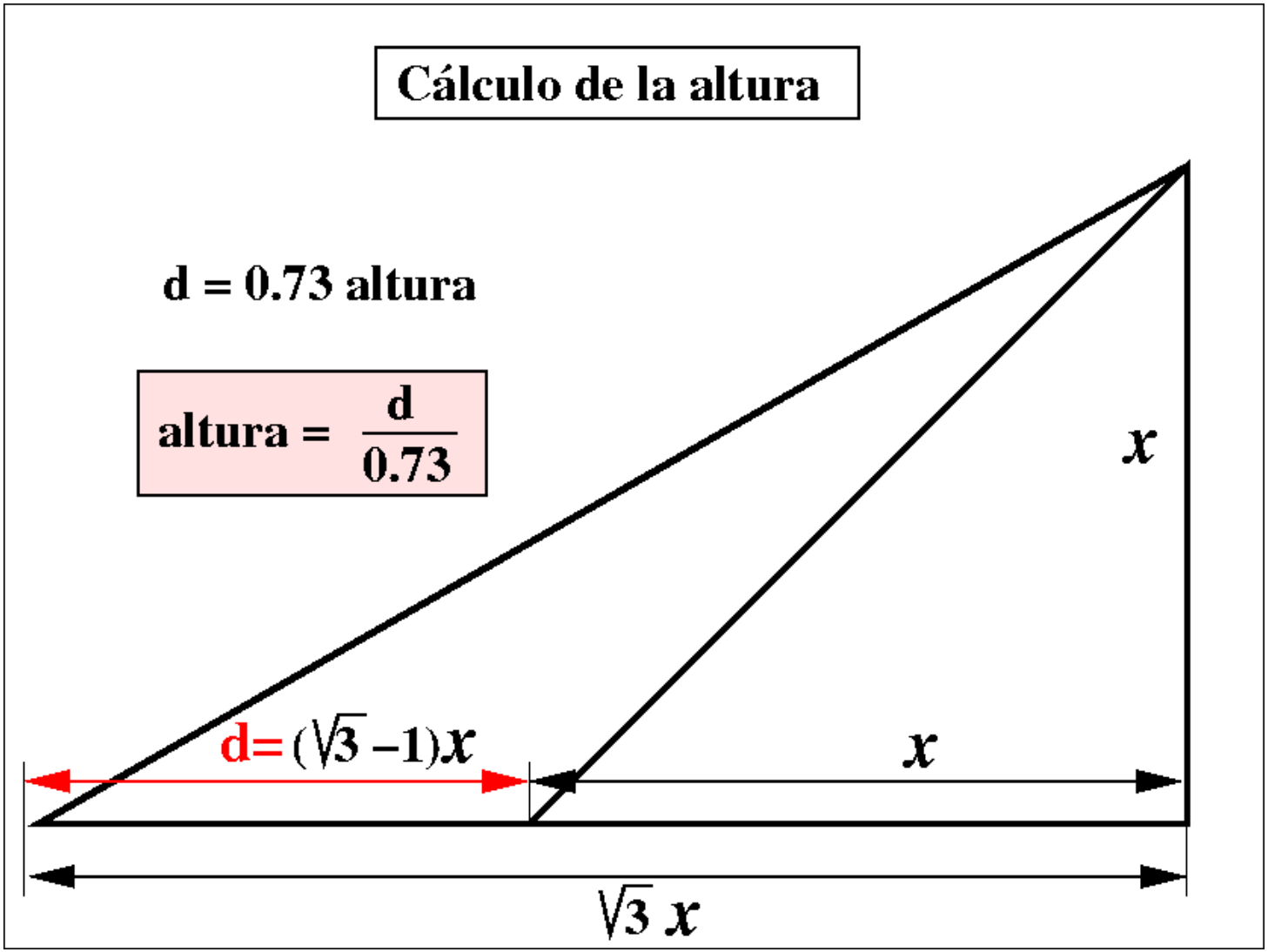

Escuadra
base = altura
x
45°
x


Cartabón
base = 1.73 altura
x
30º
√3 x


Cálculo de la altura
x
x
√3 x


Cálculo de la altura
x
d=(√3 −1)x
x
√3 x


Cálculo de la altura
d = 0.73 altura
x
d= (√3 –1)x
x
√3 x


Cálculo de la altura
d = 0.73 altura
altura = d / 0.73
x
d= (√3 –1)x
x
√3 x

Números y letras

Cuando escribimos un mensaje, leemos un libro electrónico, escribimos un documento, las letras internamente están representadas por números

El código ASCII (en inglés, las siglas de *código estándar americano para el intercambio de información*) es un convenio para realizar esta asignación

Ceros y unos

Internamente para un ordenador los números son almacenados en base dos: con ceros y unos

$$\begin{array}{rcccccccc} 97 & = & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ & & 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \end{array}$$

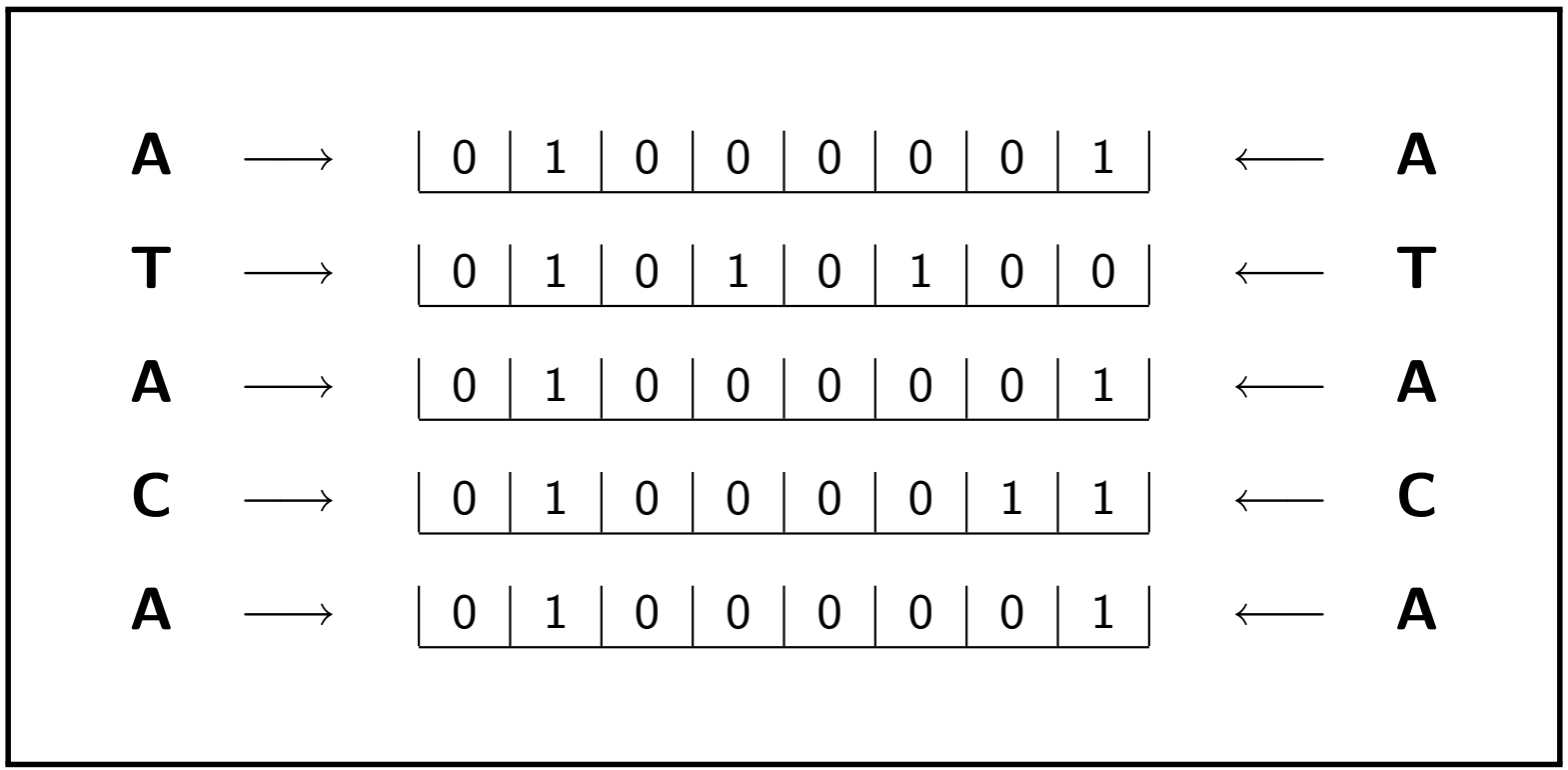
E	$\longrightarrow$	69	$\longrightarrow$	0	1	0	0	0	1	0	1
s	$\longrightarrow$	115	$\longrightarrow$	0	1	1	1	0	0	1	1
t	$\longrightarrow$	116	$\longrightarrow$	0	1	1	1	0	1	0	0
u	$\longrightarrow$	117	$\longrightarrow$	0	1	1	1	0	1	0	1
d	$\longrightarrow$	100	$\longrightarrow$	0	1	1	0	0	1	0	0
i	$\longrightarrow$	105	$\longrightarrow$	0	1	1	0	1	0	0	1
a	$\longrightarrow$	97	$\longrightarrow$	0	1	1	0	0	0	0	1

Un cambio en unos pocos bits puede hacer que un mensaje sea equívoco

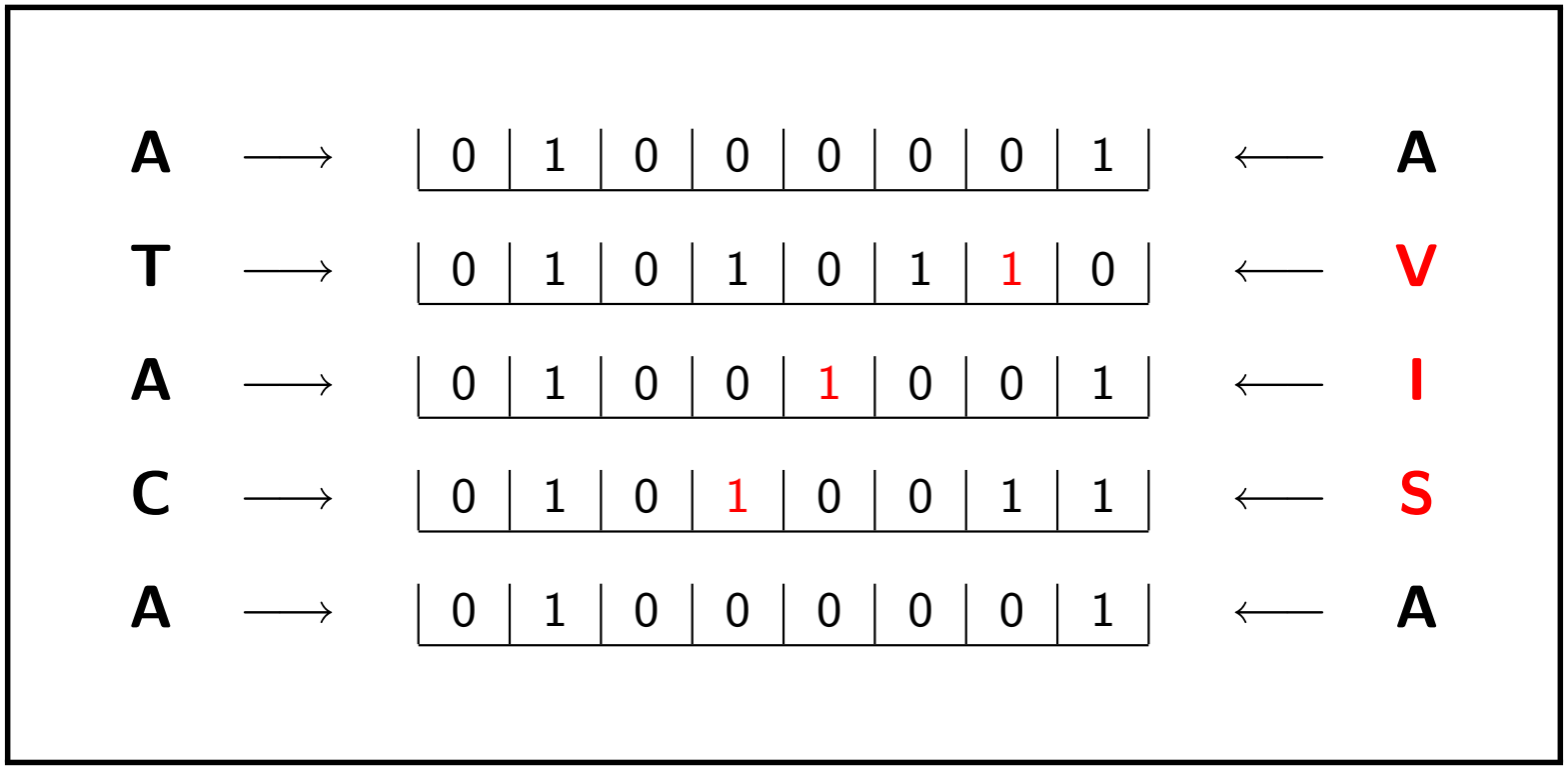
En el caso del software un solo bit erróneo en una parte principal corrompería totalmente el programa y lo haría inservible

Los canales habituales de comunicación están sometidos a interferencias esporádicas imprevisibles

Un cambio de tres bits entre cuarenta



Un cambio de tres bits entre cuarenta



Un CD sin estrenar puede tener cientos de miles de errores de grabación. En un CDRom 50 errores por cada 2 kilobytes se considera una tasa buena [dato por comprobar]

Cerca del 15 % de un CDRom está ocupado por información redundante para corregir errores

¿Sirve de algo repetir las cosas?

Transmitido

Estudia Estudia Estudia

01000101 01110011 01110100 01110101 01100100 01101001 01100001
01000101 01110011 01110100 01110101 01100100 01101001 01100001
01000101 01110011 01110100 01110101 01100100 01101001 01100001

¿Sirve de algo repetir las cosas?

Transmitido

Estudia Estudia Estudia

01000101 01110011 01110100 01110101 01100100 01101001 01100001
01000101 01110011 01110100 01110101 01100100 01101001 01100001
01000101 01110011 01110100 01110101 01100100 01101001 01100001

Si es raro que haya más de un error en tres bits entonces repetir tres veces permite corregir errores

¿Sirve de algo repetir las cosas?

Transmitido

Estudia Estudia Estudia

01000101 01110011 01110100 01110101 01100100 01101001 01100001
01000101 01110011 01110100 01110101 01100100 01101001 01100001
01000101 01110011 01110100 01110101 01100100 01101001 01100001

Si es raro que haya más de un error en tres bits entonces repetir tres veces permite corregir errores

Recibido

01000101 01111011 01110100 01110101 01100100 01101001 01100001
01000101 01110011 01110100 01110001 01100100 01101001 01100001
01000101 01110011 01110100 01110101 01100100 01101001 01000011

¿Sirve de algo repetir las cosas?

Transmitido

Estudia Estudia Estudia

01000101 01110011 01110100 01110101 01100100 01101001 01100001
01000101 01110011 01110100 01110101 01100100 01101001 01100001
01000101 01110011 01110100 01110101 01100100 01101001 01100001

Si es raro que haya más de un error en tres bits entonces repetir tres veces permite corregir errores

Recibido

01000101 01111011 01110100 01110101 01100100 01101001 01100001
01000101 01110011 01110100 01110001 01100100 01101001 01100001
01000101 01110011 01110100 01110101 01100100 01101001 01000011

01000101 01110011 01110100 01110101 01100100 01101001 01100001

¿Queda algo por hacer en Matemáticas?

Sí, muchas cosas. Las Matemáticas nunca se acaban.

Ejemplo:

Los números *perfectos* son los que coinciden con la suma de los divsores menores que él

$$6 = 1 + 2 + 3, \qquad 28 = 1 + 2 + 4 + 7 + 14.$$

¿Hay algún número perfecto que sea impar?

Hay multitud de problemas de enunciado elemental que nadie sabe resolver

¿Son aburridas las Matemáticas?

Hay gente que se aburre con las Matemáticas igual que hay gente que se aburre con la música clásica.
Si te divierten o no es algo que debes responder tú.

Si esta charla ha despertado en ti cierto interés por las Matemáticas quizá te guste revisar la bibliografía que acompaña a este fichero

El fichero PDF de esta presentación se puede descargar desde
`http://www.uam.es/fernando.chamizo`

Bibliografía para aprender más:

- A. Córdoba. *La saga de los números*. Crítica 2006.
- W. Dunham. *Euler. El maestro de todos los matemáticos*. Nivola 2000.
- M. Gardner. *Ruedas, vida y otras diversiones matemáticas*. RBA 2008.
- T. Gowers. *Matemáticas una breve introducción*. Alianza Editorial 2008.
- G.H. Hardy. *Apología de un matemático*. Nivola 1999.
- S. Lang. *El placer estético de las matemáticas*. Alianza Universidad 1992.
- I. Stewart. *De aquí al infinito*. Crítica 1998.
- I. Stewart. *Locos por las matemáticas*. Crítica 2006.

Se acompaña el código y una explicación sobre los programas empleados en la charla. Están escritos en Sage. Si alguien los quiere probar y no desea instalar este *software*, puede hacerlo registrándose en

`http://www.sagemath.org/` (Try Sage Online)

Programas en C y la explicación matemática de este y otros códigos correctores sencillos se pueden encontrar en el capítulo 4 de
`http://www.uam.es/fernando.chamizo/libreria/fich/APaplic08.pdf`

Programa 1

Muestra la codificación ASCII de cada letra de un texto dando su expresión decimal y binaria. También muestra los bits agrupados de cuatro en cuatro y los cuenta.

```
import string

@interact
def _(texto = 'Estudia'):
    for let in texto:
print let, '->', str(ord(let)).rjust(3), '->', (bin(ord(let))[2:]).rjust(8,'0')
    print '--------------------------------------------------'
    for let in texto:
print (bin((ord(let)&240)>>4)[2:]).rjust(4,'0'), (bin(ord(let)&15)[2:]).rjust(4,'0'),
    print ' ('+str(8*len(texto))+' bits)'
```

Ejemplo del programa 1

texto Hola

```
H ->  72 -> 01001000
o -> 111 -> 01101111
l -> 108 -> 01101100
a ->  97 -> 01100001
--------------------------------------------------------
0100 1000 0110 1111 0110 1100 0110 0001  (32 bits)
```

Programa 2

Transforma un texto en una tira de bloques de 7 bits más larga que su codificación ASCII.

```
import string
def codif(ceyun):
v=[int('1101001',2),int('0101010',2),int('1001100',2),int('1110000',2)]
num=int(ceyun,2)
j=0
for k in range(0,4):
if int(ceyun,2)&(2^k): j=(j).__xor__(v[k])
print (bin(j)[2:]).rjust(7,'0'),

@interact
def _(texto = 'Estudia'):
    for let in texto:
codif( (bin((ord(let)&240)>>4)[2:]).rjust(4,'0') )
        codif( (bin(ord(let)&15)[2:]).rjust(4,'0') )
    print ' ('+str(14*len(texto))+' bits)'
```

Ejemplo del programa 2

texto Hola

```
1001100 1110000 1100110 1111111 1100110 0111100 1100110 1101001  (56
bits)
```

Programa 3

Corrige un error por bloque de una tira de bits

```
import string
_cad = ''
def corrige(num):
        global _cad
j = 0
for k in range(0,7):
if num&(2^k): j = (j).__xor__(7-k)
        print (bin(num)[2:]).rjust(7,'0'),
if j != 0:
            num = (num).__xor__(2^(7-j))
            print '*'+str(j)+'*',
        else: print '***',
        _cad += (bin(((num&16)>>1)+(num&7))[2:]).rjust(4,'0')
print (bin(num)[2:]).rjust(7,'0'),'=>', _cad[-4:],
        if len(_cad) == 8:
            print ' -> ', chr(int(_cad,2)),
            _cad = ''
        print ''

@interact
def _(tira = '1001100 1110000 1100110 1111111 1100110 0111100 1100110 1101001'):
    for item in tira.split():
        corrige(Integer(int(item,2)))
```

Ejemplo del programa 3

tira 1001100 1010000 1101110 1111110 1110110 0111100 0100110 1111001

```
1001100 *** 1001100 => 0100
1010000 *2* 1110000 => 1000  ->  H
1101110 *4* 1100110 => 0110
1111110 *7* 1111111 => 1111  ->  o
1110110 *3* 1100110 => 0110
0111100 *** 0111100 => 1100  ->  l
0100110 *1* 1100110 => 0110
1111001 *3* 1101001 => 0001  ->  a
```