

**Novática**, revista fundada en 1975 y decana de la prensa informática española, es el órgano oficial de expresión y formación continua de **ATI** (Asociación de Técnicos de Informática). **Novática** edita también **Upgrade**, revista digital de **CEPIS** (Council of European Professional Informatics Societies), en lengua inglesa, y es miembro fundador de **UPENET** (UPGRADE European Network)

<<http://www.ati.es/novatica/>>  
<<http://www.upgrade-cepis.org/>>

ATI es miembro fundador de **CEPIS** (Council of European Professional Informatics Societies) y es representante de España en **IFIP** (International Federation for Information Processing); tiene un acuerdo de colaboración con **ACM** (Association for Computing Machinery), así como acuerdos de vinculación o colaboración con **AdaSpain**, **AIZ** y **ASTIC**.

### CONSEJO EDITORIAL

Antoni Carbonell Noqueas, Francisco López Crespo, Julián Marcelo Cocho, Celestino Martín Alonso, Josep Molas i Bertrán, Roberto Moya Quiles, César Pérez Chirinos, Mario Piattini Velthuis, Fernando Píera Gómez (Presidente del Consejo), Miquel Sarries Griñó, Asunción Yturbe Herranz

#### Coordinación Editorial

Rafael Fernández Calvo <rfcalvo@ati.es>

#### Composición y autoedición

Jorge Llorens <lllorens@ati.es>

#### Traducciones

Grupo de Lengua e Informática de ATI <<http://www.ati.es/gl/lengua-informatica/>>

#### Administración

Tomás Brunete, María José Fernández, Enric Camarero, Felicidad López

### SECCIONES TÉCNICAS: COORDINADORES

#### Administración Pública electrónica

Gumerindo García Arribas, Francisco López Crespo (MAP)

<gumerindo.garcia@map.es>, <flc@ati.es>

#### Argumentarios

Jordi Tubella (DAC-UPC) <jordit@ac.upc.es>

Victor Vilalta Yulera (Univ. de Zaragoza) <victor@unizar.es>

#### Audiencia ATIC

Marina Tourino, Manuel Palao (ASIA)

<marinatourino@marinatourino.com>, <manuel@palao.com>

#### Base de datos

Corat Calero Muñoz, Mario G. Piattini Velthuis

(Escuela Superior de Informática, UCLM)

<Corat.Calero@uclm.es>, <mpiattini@inf-cr.uclm.es>

#### Hardware y Periféricos

Isabel Hernando Collazas (Fac. Derecho de Donostia, UPV) <ihernando@legaltek.net>

Isabel Davara Fernández de Marcos (Davara & Davara) <idavara@davara.com>

#### Seguridad e Inseguridad

José María López (Univ. de Zaragoza) <jlopez@unizar.es>

Cristóbal Pareja Flores (DSIP-UCM) <cpajef@dsip.ucm.es>

#### Gestión del Conocimiento

Joan Baiget Solé (Cap Gemini Ernst & Young) <jbaiget@ati.es>

#### Informática y Filosofía

Josep Corco (UC) <jcorco@unijca.edu>

Esperanza Marcos (ESCET-URJC) <cuca@escet.urjc.es>

#### Informática Gráfica

Miquel Chover Selles (Universitat Jaume I de Castellón) <chover@lsi.uji.es>

Roberto Vivó (Eurographics, sección española) <rvivo@dsic.upv.es>

#### Ingeniería del Software

Javier Dolado Costa (UPV) <dolado@si.ehu.es>

Luis Fernández (PRIS-El-UEM) <lufern@dpris.esi.uem.es>

#### Inteligencia Artificial

Federico Barber, Vicente Botti (DSIC-UPV)

<fvbotti\_fbarber@dsic.upv.es>

#### Información Persona-Computador

Julio Abascal González (FI-UPV) <julio@si.ehu.es>

Jesús Lora Vidal (Univ. de Lleida) <jesus@sup.udl.es>

#### Internet

Alonso Álvarez García (TID) <alonso@ati.es>

Llorens Paredes Casas (Iindra) <pages@ati.es>

#### Lengua e Informática

M. del Carmen Ugarte (IBM) <cugarte@ati.es>

#### Lenguajes Informáticos

Andrés Martín López (Univ. Carlos III) <amarin@it.uc3m.es>

J. Ángel Velázquez (ESCET-URJC) <a.velazquez@escet.urjc.es>

#### Librerías e Informática

Alfonso Escobedo (IR-UPV, de La Laguna) <aescolan@ull.es>

#### Lingüística computacional

Xavier Gómez Guinovart (Univ. de Vigo) <xgg@uvigo.es>

Manuel Palomar (Univ. de Alicante) <mpalomar@lsl.ua.es>

#### Mundo estudiantil

Adolfo Vázquez Rodríguez (Rama de Estudiantes del IEEE-UCM)

<a.vazquez@ieee.org>

#### Profesión Informática

Rafael Fernández Calvo (ATI) <rfcalvo@ati.es>

Miquel Sarries Griñó (Ayto. de Barcelona) <msarries@ati.es>

#### Redes y servicios telemáticos

Luis Calvario Colomo (DCIM-UPV) <lcalvario@dcim.upv.es>

Josep Saló Parela (DAC-UPC) <parela@ac.upc.es>

#### Seguridad

Javier Arellito Bertolin (Univ. de Deusto) <jarellito@eside.deusto.es>

Javier López Muñoz (ETSI Informática-UMA) <jlm@lcc.uma.es>

#### Sistemas de Tiempo Real

Alejandro Alonso, Juan Antonio de la Puente

(DI-UPM) <jalonso.juante@ati.es>

#### Software Libre

Jesús M. González Barahona, Pedro de las Heras Quirós

(GSVC-URJC) <frob.gheras@gsvc.escet.urjc.es>

#### Tecnología de Objetos

Jesús García Molina (DIS-UM) <jmolina@correo.um.es>

Gustavo Rossi (LFLIA-UNLP, Argentina) <gustavo@sol.info.unlp.edu.ar>

#### Tecnología para la Educación

Juan Manuel Dodero Beardo (UC3M) <dodero@inf.uc3m.es>

Francisco Riviere (PalmCAT) <friviere@wanadoo.es>

#### Tecnología y Empresa

Pablo Hernández Medrano (Bluemat) <pablohm@bluemat.biz>

#### TIC para la Sanidad

Valentín Masero Vargas (DI-UNEX) <vmasero@unex.es>

#### TIC y Turismo

Andrés Aguayo Maldonado, Antonio Guevara Plaza (Univ. de Málaga)

<aguayo.guevara@lcc.uma.es>

#### Las opiniones expresadas por los autores son responsabilidad exclusiva de los mismos.

**Novática** permite la reproducción de todos los artículos, salvo los marcados con © o copyright, debiéndose en todo caso citar su procedencia y enviar a **Novática** un ejemplar de la publicación.

#### Coordinación Editorial, Redacción Central y Redacción ATI Madrid

Padilla 66, 3º dcha., 28006 Madrid

Tel. 914029391; fax 913093685 <novatica@ati.es>

#### Composición, Edición y Redacción ATI Valencia

Av. del Reino de Valencia 23, 46005 Valencia

Tel./fax 963330392 <secretal@ati.es>

#### Administración y Redacción ATI Cataluña

Via Laietana 41, 1º, 14 08003 Barcelona

Tel. 934125235; fax 934127713 <secretgen@ati.es>

#### Redacción ATI Andalucía

Isaac Newton, s/n, Ed. Sadiel,

Isla Cartuja 41092 Sevilla. Tel./fax 954460779 <secretand@ati.es>

#### Redacción ATI Aragón

Lagasca 9, 3-B, 50006 Zaragoza.

Tel./fax 976235181 <secretara@ati.es>

#### Redacción ATI Asturias-Cantabria

<gg-astucant@ati.es>

#### Redacción ATI Castilla-La Mancha

<gg-clmancha@ati.es>

#### Redacción ATI Galicia

Recinto Ferial s/n, 36340 Silleda (Pontevedra)

Tel. 986581413; fax 986580162 <secretgal@ati.es>

#### Subscripción y Ventas

<<http://www.ati.es/novatica/interes.html>>, o en ATI Cataluña o ATI Madrid

#### Publicidad

Padilla 66, 3º dcha., 28006 Madrid

Tel. 914029391; fax 913093685 <novatica.publicidad@ati.es>

#### Imprenta

9 impresos S.A. Juan de Austria 66, 08005 Barcelona.

**Distribución legal:** B.15.154-1975 -- ISSN. 0211-2124. CODEN NOVACB

**Portada:** Antonio Crespo Folix / © ATI 2004

**Diseño:** Fernando Agresta / © ATI 2004

### en resumen

**TPS o el software como proceso**

Rafael Fernández Calvo

> 02

## monografía

**Tecnología de Proceso Software**

(En colaboración con **Upgrade**)

Editores invitados: *Francisco Ruiz González, Gerardo Canfora*

**Presentación. La Tecnología de Proceso Software y la mejora de la gestión de los proyectos y de la calidad de los productos**

*Francisco Ruiz González, Gerardo Canfora*

> 03

**Procesos Software: características, tecnología y entornos**

*Francisco Ruiz González, Gerardo Canfora*

> 05

**Cuestiones clave y nuevos retos en la Tecnología de Proceso Software**

*Jean-Claude Derniame, Flavio Oquendo*

> 09

**Una taxonomía de los Servicios de Entornos de Ingeniería de Software: la futura norma ISO/IEC 15940**

*Dan Hyung Lee, Juan Garbajosa Sopena*

> 14

**Software libre y de código abierto: ¿un nuevo modelo para el desarrollo de software?**

*Alfonso Fuggetta*

> 18

**Aplicación de los principios básicos de la Ingeniería de Modelos al campo de la Ingeniería de Procesos**

*Jean Bézivin, Erwan Breton*

> 22

**Lenguajes de Modelización de Procesos de Software basados en UML**

*Pere Botella i López, Xavier Franch Gutiérrez, Josep M. Ribó Balust*

> 27

**Soporte a los Procesos Software en un Entorno de Ingeniería del Software orientado a Procesos**

*Hans-Ulrich Kobiak*

> 32

**Gestión de proyectos distribuidos con GENESIS**

*Lerina Aversano, Andrea De Lucia, Matteo Gaeta,*

*Pierluigi Ritrovato, Maria Luisa Villani*

> 38

**Medición de los Procesos Software**

*Félix García Rubio, Francisco Ruiz González, Mario Piattini Velthuis*

> 43

## contribución invitada

**Más allá de Internet: la Red Universal Digital**

*Fernando Sáez Vacas*

> 48

## secciones técnicas

**Administración Pública electrónica**

**Agència Catalana de Certificació: la certificación digital en la Administración Pública catalana**

*Josep Llopis Méndez*

> 52

**Ingeniería del Software**

**Diseño basado en componentes: alternativas en la etapa de partición**

*Arantza Irastorza Goñi, Arturo Jaime Elizondo, Oscar Díaz García*

> 55

**Desarrollo de Sistemas de Inspección Visual Automatizada a partir de la descripción de un Patrón Arquitectural Genérico**

*Cristina Vicente Chicote, Carlos Fernández Andrés, Pedro Sánchez Palma*

> 63

**Profesión informática**

**TIC y Sociedad de la Información: propuestas para una nueva etapa**

*Josep Molas i Bertrán*

> 66

**Referencias autorizadas**

> 68

## sociedad de la información

**Personal y transferible**

**Protección de datos personales y Seguridad del Estado**

*Ofelia Tejerina Rodríguez*

> 73

**programar es crear**

**Diseño de suelos (CUPCAM 2003, problema G, solución)**

*Juan Céspedes Prieto, Antonio Fernández Anta, Ángel Herranz Nieva*

> 74

## asuntos interiores

**Coordinación editorial / Programación de Novática**

**Normas de publicación para autores / Socios Institucionales**

> 76

> 77

**Monografía del próximo número: "Criptografía"**

Juan Céspedes Prieto<sup>1</sup>, Antonio Fernández Anta<sup>1</sup>, Ángel Herranz Nieva<sup>2</sup>  
<sup>1</sup> Universidad Rey Juan Carlos; <sup>2</sup> Facultad de Informática, Universidad Politécnica de Madrid

<cespedes.anto@gsync.esct.urjc.es>  
 <aherranz@fi.upm.es>

# Problema G: Diseño de suelos

El enunciado de este problema apareció en el número 170 de *Novática* (julio-agosto 2004, p. 72). Es el problema G de los planteados en el I Concurso Universitario de la Comunidad Autónoma de Madrid (CUPCAM 2003)

## 1. Resumen del problema

El problema presentado en el número anterior pedía encontrar el número de todos los posibles diseños distintos que se pueden elegir para un suelo dado un determinado conjunto de baldosas. Este problema tiene una fuerte componente combinatoria. Disponemos de sólo 4 tipos de baldosas y el diseño debe ser simétrico.

## 2. Contando diseños

Si tuviéramos tanta cantidad de baldosas de cada tipo como quisiéramos, bastaría con contar todos los posibles diseños simétricos. Para ello podemos observar que en un diseño simétrico basta fijar una mitad del mismo para tener el diseño completo. Claramente, si la longitud es impar la fila central debe formar parte de esa 'mitad' que hay que fijar. Entonces, para el caso concreto de una habitación de dimensiones  $l \times w$ , el número de baldosas a fijar es  $\lceil l/2 \rceil \times w$ . Como cada baldosa debe ser de uno de 4 posibles tipos, el número total de diseños ( $D_{tot}$ ) se calcula fácilmente así:

$$D_{tot}(l,w) = 4^{\lceil l/2 \rceil w}. \quad (1)$$

Sin embargo, tenemos limitado el número de baldosas de cada tipo ( $t_i$ ). Eso impide que podamos usar la fórmula 1 directamente. Sin embargo sí se podría plantear el generar todos estos casos y no contar aquéllos que no cumplan con las restricciones en número de baldosas de algún tipo. Como en el caso peor nos dicen que la superficie es de 33 pies cuadrados, y éste se puede dar con una habitación de  $33 \times 1$ , en el caso peor estaríamos hablando de  $4^{33}$  casos, que es un número enorme y hace inviable esta solución.

## 3. Contando con más cuidado

Por lo tanto, vamos a tener que contar con más cuidado y tener en cuenta el número de baldosas disponible de cada tipo. Primero debemos observar que cuando consideramos solamente la mitad del diseño, como hacemos arriba, fijar una baldosa a un tipo en realidad implica consumir dos baldosas de ese tipo, excepto si la longitud es impar y estamos fijando la fila central. Por ello vamos a separar la superficie de la habitación en dos partes, la fila central y el resto.

Entonces, si tenemos que diseñar una superficie de área  $s$ , y hemos fijado un reparto de sus baldosas entre los cuatro tipos, donde  $b_i$  ( $0 \leq b_i \leq t_i$ ) es el número de baldosas elegido de tipo  $i$  tal que

$$\sum_{i=1}^4 b_i = s,$$

podemos usar la fórmula de las permutaciones con repetición para calcular el número de posibles diseños ( $D_{lim}$ ) con ese reparto:

$$D_{lim}(s,b_1,b_2,b_3,b_4) = \frac{s!}{\prod_{i=1}^4 b_i!} \quad (2)$$

Entonces, sumando estos valores para todos los posibles repartos tendríamos el número total de diseños para esa superficie. Nótese que para un área de  $s$  el número máximo de posibles repartos se puede calcular como

**Nota del Editor:** con la solución de este problema de programación concluimos la serie de los planteados en el I Concurso Universitario de la Comunidad Autónoma de Madrid (CUPCAM 2003), serie iniciada en el número 166 (noviembre-diciembre 2003) y que hemos podido ofrecer a nuestros lectores gracias al generoso y continuado esfuerzo de un grupo de docentes informáticos encabezado por nuestro colaborador **Cristóbal Pareja Flores**. A todos ellos expresamos nuestro más sincero agradecimiento.

$$\sum_{i=0}^s \sum_{j=i}^s s-j+1 \leq (s+1)^3, \quad (3)$$

que para  $s$  pequeño es una cantidad razonable.

## 4. Programando

Este análisis da lugar al programa que presentamos. En el mismo se distingue si la longitud de la habitación es un número par o impar. En el primer caso simplemente busca todos los posibles repartos para la mitad de la superficie, teniendo en cuenta, como hemos dicho, que cada posición en realidad consume dos de las baldosas disponibles, y se suma el número de diseños con cada reparto. Sin embargo, si la longitud es impar, se multiplica el número de diseños para la fila central por el número de diseños total para el resto de la habitación. De nuevo, sólo se considera la mitad del resto.

```
#include <stdio.h>
#define MIN(a,b) ((a<b)?(a):(b))
#define MAX(a,b) ((a>b)?(a):(b))

long int fact(int x)
{ int i;
  long int r=1;

  for (i=2; i<=x; i++) r *= i;
  return r;
}

int even_length(size,t1,t2,t3,t4)
{ int size,t1,t2,t3,t4;
  { int b1,b2,b3;
    long int count=0, fsize;

    fsize=fact(size);
    for (b1=0;
         b1<=MIN(size,t1);
         b1++)
      for (b2=0;
           b2<=MIN(size-b1,t2);
           b2++)
        for (b3=0;
             b3<=MIN(size-b1-b2,t3);
             b3++)
          if (t4 >= size-b1-b2-b3)
            count +=
              fsize
              / (fact(b1)
                 *fact(b2)
                 *fact(b3)
                 *fact(size-b1-b2-b3));
    return count;
  }

int odd_length(length,width,
               t1,t2,t3,t4)
{ int length,width,t1,t2,t3,t4;
  { int b1,b2,b3,size;
    long int count=0, fwidth;

    fwidth=fact(width);
    size=(length-1)*width/2;
    for (b1=0;
         b1<=MIN(width,t1);
         b1++)
      for (b2=0;
           b2<=MIN(width-b1,t2);
           b2++)
        for (b3=0;
             b3<=MIN(width-b1-b2,t3);
             b3++)
          if (t4 >= width-b1-b2-b3)
            count +=
              (fwidth
               / (fact(b1)
```

```

        *fact(b2)
        *fact(b3)
        *fact(width-b1-b2-b3))
    * even_length(
        size,
        (t1-b1)/2,
        (t2-b2)/2,
        (t3-b3)/2,
        (t4-(width-b1-b2-b3))
        / 2);
    return count;
}

main() {
    long int l, w, t1, t2, t3, t4;
    long int d;

    scanf("%d", &l);
    while (l != 0) {
        scanf("%d %d %d %d %d",
            &w, &t1, &t2, &t3, &t4);
        if ((l % 2) == 0) {
            d = even_length(l*w/2,
                t1/2,
                t2/2,
                t3/2,
                t4/2);
        }
        else {
            d = odd_length(l,
                w,
                t1,
                t2,
                t3,
                t4);
        }
        printf("Número de diseños: %d\n",
            d);
        scanf("%d", &l);
    }
}

```

**5. Desbordamiento**

Tras ejecutar el programa anterior con distintos casos de prueba nos encontramos con la siguiente situación:

```

$ ./tiles
7 4 20 20 20 20
Número de diseños: -762568
0
$

```

Esto indica que nuestro programa sufre desbordamientos. Puesto que la superficie de la habitación es como máximo de 33 pies cuadrados, existe la posibilidad de que nuestra solución acabe intentando calcular el factorial de 33 (86833176188118864955 1819440128e7!) algo que garantizaría resultados incorrectos incluso utilizando mayor precisión (ver la definición de la macro ULLONG\_MAX en limits.h).

A continuación mostramos un programa que supera este problema precomputando los números combinatorios necesarios (*triángulo de Pascal*) y en el que se ha introducido una variación en el cálculo del número de diseños con un algoritmo recursivo sobre el número de huecos sin cubrir y el tipo de baldosa (evitando un código dependiente de los tipos de baldosas).

```

#include <stdio.h>
#define MAX_AREA 33
#define MAX_TIPOS 4

/*
 * Índice 0 indica huecos por cubrir.
 * Índices mayores que 0 indican
 * baldosas disponibles de un tipo.
 */
typedef unsigned int
    disponibles_t[MAX_TIPOS+1];
/* Números combinatorios */
unsigned C[MAX_AREA+1][MAX_AREA+1];

/* Triángulo de pascal */
void pascal()
{
    unsigned n, m;
    for (n = 0; n <= MAX_AREA; n++) {
        C[n][0] = 1;
        for (m = 1; m < n; m++) {
            C[n][m] =
                C[n-1][m-1] + C[n-1][m];
        }
    }
}

```

```

    }
    C[n][m] = 1;
}
}

/* Diseños */
long unsigned D(unsigned tipo,
    disponibles_t disp)
{
    unsigned huecos = disp[0];
    long unsigned dibujos;

    if (huecos == 0) {
        /* Suelo cubierto */
        dibujos = 1;
    }
    else if (tipo == 0) {
        /* Suelo imposible de cubrir */
        dibujos = 0;
    }
    else {
        int usadas;

        dibujos = 0;
        for (usadas = 0;
            usadas <= disp[tipo]
            && usadas <= huecos;
            usadas++) {
            disp[0] = huecos - usadas;
            dibujos +=
                D(tipo - 1, disp)
                * C[huecos][usadas];
        }
        return dibujos;
    }
}

```

```

int main() {
    unsigned l, w, d1, d2, d3, d4;
    disponibles_t disp;

    pascal();

    scanf("%d", &l);
    while (l != 0) {
        scanf("%d %d %d %d %d",
            &w, &d1, &d2, &d3, &d4);
        disp[0] = l * w / 2;
        disp[1] = d1 / 2;
        disp[2] = d2 / 2;
        disp[3] = d3 / 2;
        disp[4] = d4 / 2;
        if ((l % 2) == 0) {
            printf("\
Número de diseños: %lu\n",
                D(MAX_TIPOS, disp));
        }
        else {
            fprintf(stderr,
                "\
No funciona para longitud impar:\n");
            return 1;
        }
        scanf("%d", &l);
    }
    return 0;
}

```

- Como se puede ver, se han dejado al lector un par de ejercicios interesantes:
- Añadir al programa la capacidad de calcular los números de diseños para longitudes impares sin introducir un código dependiente de los tipos de baldosas.
  - Evitar tener que recalcular números combinatorios utilizando la propiedad  $C_{n,m} = C_{m,n}$ .

**Agradecimientos**

Queremos terminar con unas palabras de agradecimiento a Cristobal Pareja Flores por todas sus aportaciones a estos comentarios.

**Nota**

<sup>1</sup> Calculado en el intérprete de Haskell Hugs mediante la expresión `let f n = if n==1 then 1 else n*f(n-1) in f 33`