LENGUAJES DE PROGRAMACIÓN

Solución al Trabajo Práctico - Convocatoria extraordinaria de 2024

Ejercicio 1

En una autopista de peaje hay 10 puntos de entrada y salida llamados A, B, C, D, E, F, G, H, I y J, que están ubicados consecutivamente a lo largo de la autopista, y que permiten el acceso a cualquiera de los dos sentidos en que puede circularse por ella. La distancia entre puntos consecutivos es la siguiente:

Distancia entre	**km**
A y B	24
B y C	37
C y D	13
D y E	12
E y F	28
F y G	19
G y H	36
H e I	49
I y J	29

El precio del peaje se calcula en función de los km recorridos por el cliente. Los primeros 50 km recorridos se cobran a 0.05 euros/km. Los siguientes 50 km se cobran a 0.04 euros/km. Los km recorridos a partir de 100 km se cobran a 0.02 euros/km. A continuación se muestran tres ejemplos.

- *Ejemplo 1.* Un cliente entra a la autopista en B y sale en A. Ha recorrido 24 km. El precio es: 24 km $\times$ 0.05 euros/km = 1.20 euros.
- *Ejemplo 2.* Un cliente entre en D y sale en H. Ha recorrido 12+28+19+36 = 95 km. El precio es: 50 km $\times$ 0.05 euros/km + 45 km $\times$ 0.04 euros/km = 4.30 euros.

– *Ejemplo 3.* Un cliente entra en J y sale en A. Ha recorrido $24+37+13+12+28+19+36+49+29 = 247$ km. El precio es: 50 km $\times$ 0.05 euros/km + 50 km $\times$ 0.04 euros/km + 147 km $\times$ 0.02 euros/km = 7.44 euros.

Escriba un programa en C++ que escriba un mensaje en la consola solicitando al usuario que introduzca el punto de entrada y el punto de salida. El programa debe leer los valores introducidos por consola. Si no corresponden con puntos de entrada y salida, el programa debe escribir un mensaje en la consola indicándolo y terminar. En caso contrario, debe calcular los km recorridos y el precio del peaje, escribirlos en la consola y terminar. El precio debe escribirse en formato fijo, con dos dígitos de precisión.

Muestre en la memoria el resultado obtenido al ejecutar su programa seis veces, para los seis pares de puntos siguientes:
(1) A y B; (2) D y H; (3) J y A; (4) C e I; (5) J y B; (6) F y A.

Solución al Ejercicio 1

```cpp
#include <iostream>
#include <iomanip>
#include <vector>
#include <string>
#include <algorithm>

std::vector<std::string> vPuntos =
                        {"A","B","C","D","E","F","G","H","I","J"};
std::vector<int> vKm = {24,37,13,12,28,19,36,49,29};

int isPuntoValido(std::string P) {
   for (unsigned int i=0; i<vPuntos.size(); i++)
      if (vPuntos[i] == P) return i;
   return -1;
}

double getPrecio(int km) {
   const double Tarifa1=0.05, Tarifa2=0.04, Tarifa3=0.02;
   return  Tarifa1 * std::min( km, 50 ) +
           Tarifa2 * std::min( std::max(km-50,0), 50 ) +
           Tarifa3 * std::max( km-100, 0 );
}

int main() {
   std::string P1, P2;
   std::cout << "Punto de entrada: ";
   std::cin >> P1;
   int indP1 = isPuntoValido(P1);
   if ( indP1 == -1 ) {
      std::cerr << "Error: el punto " << P1 << " no existe";
      return 0;
   }
   std::cout << "Punto de salida: ";
   std::cin >> P2;
   int indP2 = isPuntoValido(P2);
   if ( indP2 == -1 ) {
      std::cerr << "Error: el punto " << P2 << " no existe";
      return 0;
   }
   double distancia_km = 0;
   for (int i=std::min(indP1,indP2); i<std::max(indP1,indP2); i++)
      distancia_km += vKm[i];
   std::cout << "Recorrido: " << distancia_km << " km\nPrecio: "
             << std::fixed << std::setprecision(2)
             << getPrecio(distancia_km) << " euros\n";
    return 0;
}
```

Ejercicio 2

Un sistema evoluciona pasando sucesivamente por diferentes estados que se han etiquetado mediante la letra S seguida de un número entero mayor que cero denominado el *índice* del estado. La descripción de la evolución del sistema se encuentra en un fichero de texto llamado *evolución.txt*. El contenido de éste podría ser:

```
S4 S3 S4 S5 S9 S3 S4 S6 S3
```

lo cual indicaría que el sistema inicialmente se encuentra en el estado S_4 y va pasando sucesivamente por los estados S_3, S_4, S_5, S_9, etc. El conocimiento de los instantes de tiempo en los que se producen las transiciones entre los sucesivos estados no es de interés para este problema.

Para cada fichero de evolución puede calcularse una matriz de probabilidad $\mathbf{Q}$, tal que el elemento de la matriz ubicado en la fila i y la columna j es la probabilidad de que se produzca la transición desde el estado S_j al estado S_i.

$$\mathbf{Q} = \begin{pmatrix} P(S_1 \to S_1) & P(S_2 \to S_1) & \cdots & P(S_n \to S_1) \\ P(S_1 \to S_2) & P(S_2 \to S_2) & \cdots & P(S_n \to S_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(S_1 \to S_n) & P(S_2 \to S_n) & \cdots & P(S_n \to S_n) \end{pmatrix}$$

El tamaño de la matriz es $n \times n$, siendo S_n el estado con mayor índice presente en el fichero. Es posible que alguno de los estados S_1, S_2, ..., S_{n-1} no se encuentre en el fichero, y que consiguientemente la matriz $\mathbf{Q}$ tenga filas y columnas con todos sus elementos igual a cero. Obsérvese que los elementos de la matriz deben ser no negativos y que la suma de los elementos de cada columna debe ser igual a 1.

Escriba un programa en C++ que lea un fichero de texto llamado *evolucion.txt* donde está registrada la evolución del sistema y que escriba en la consola la correspondiente matriz de probabilidad $\mathbf{Q}$. Los elementos de la matriz deben escribirse en forma de fracción. A continuación se muestra un ejemplo. Supongamos que contenido del fichero *evolucion.txt* es el siguiente:

```
S4 S3 S4 S5 S2 S1 S4 S3 S1 S5 S3 S1 S3 S4 S4 S2 S3 S3 S3 S1
S5 S3 S1 S5 S1 S3 S1 S3 S4 S1 S3 S4 S2 S2 S1 S2 S1 S1 S3 S2
S1 S1 S5 S4 S2 S3 S1 S3 S1
```

Analizando el fichero se observa que el estado con mayor índice en que se ha encontrado el sistema es S_5. Por ello, el tamaño de la matriz $\mathbf{Q}$ es 5×5. Veamos en

detalle cómo calcular la columna 3 de la matriz, la cual describe la transición desde el estado S_3. El número de veces que se produce cada transición desde el estado S_3 es:

Transición	**Número de veces**
$S_3 \rightarrow S_1$	7
$S_3 \rightarrow S_2$	1
$S_3 \rightarrow S_3$	2
$S_3 \rightarrow S_4$	4
$S_3 \rightarrow S_5$	0

El número total de transiciones desde el estado S_3 ha sido $7 + 1 + 2 + 4 + 0 = 14$. Si el sistema se encuentra en el estado S_3, la probabilidad de que el siguiente estado sea S_1 es $7/14$, de que sea S_2 es $1/14$ y así sucesivamente.

$$
\begin{aligned}
P\left(S_3 \rightarrow S_1\right) &= 7/14 \\
P\left(S_3 \rightarrow S_2\right) &= 1/14 \\
P\left(S_3 \rightarrow S_3\right) &= 2/14 \\
P\left(S_3 \rightarrow S_4\right) &= 4/14 \\
P\left(S_3 \rightarrow S_5\right) &= 0
\end{aligned}
$$

Muestre en la memoria el resultado obtenido al ejecutar su programa con el fichero *evolucion.txt* mostrado a continuación.

```
S4 S3 S4 S5 S2 S1 S4 S3 S1 S5 S3 S1 S3 S4 S4 S2 S3 S3 S3 S1 S2 S3 S2
S5 S3 S1 S5 S1 S3 S1 S3 S4 S1 S3 S4 S2 S2 S1 S2 S1 S1 S3 S2 S3 S3 S2
S1 S1 S5 S4 S2 S3 S1 S3 S1 S5 S2 S8 S4 S3 S3 S3 S5 S5 S3 S7 S6 S8 S7
S6 S7 S8 S8 S6 S7 S5 S6 S7 S7 S7 S7 S8 S7 S6 S5 S4 S3 S2 S1 S2 S2 S3
S2 S1 S1 S1 S2 S3 S3 S3 S4 S4 S4 S4 S3 S3 S2 S3 S5 S6 S7 S7 S5 S6 S5
S7 S8 S8 S8 S7 S6 S6 S6 S7 S6 S6 S6 S7 S6 S6 S6 S7 S6 S6 S6 S6 S5 S4
S5 S2 S1 S4 S3 S1 S5 S3 S2 S8 S4 S3 S3 S3 S5 S1 S5 S1 S5 S6 S5 S4 S3
S6 S7 S8 S3 S1 S3 S4 S3 S3 S4 S4 S5 S5 S5 S5 S5 S6 S7 S8 S8 S8 S7 S5
S6 S7 S8 S6 S5 S4 S5 S6 S5 S6 S7 S8 S8 S8 S8 S7 S7 S6 S6 S6 S5 S4 S3
S3 S1 S3 S4 S6 S7 S8 S7 S5 S5 S5 S3 S3 S4 S4 S3 S2 S1 S1 S1 S2 S4 S5
S7 S6 S6 S6 S7 S6 S6 S6 S7 S8 S8 S6 S7 S8 S6 S8 S7 S6 S8 S6 S7 S5 S4
S2 S1 S4 S3 S1 S5 S3 S2 S3 S1 S3 S1 S1 S3 S4 S4 S2 S3 S3 S3 S4 S3 S4
S2 S3 S1 S3 S1 S2 S1 S4 S3 S1 S5 S3 S1 S3 S4 S4 S2 S3 S3 S3 S2 S3 S5
S6 S7 S8 S8 S6 S7 S5 S6 S7 S7 S7 S7 S8 S7 S6 S5 S4 S3 S2 S1 S2 S2 S6
S5 S6 S7 S2 S1 S4 S3 S1 S5 S2 S3 S1 S3 S1 S3 S1 S3 S4 S4 S2 S3 S3 S2
S6 S7 S8 S6 S5 S4 S5 S6 S5 S6 S7 S8 S8 S8 S8 S7 S7 S6 S6 S6 S5 S4 S5
S4 S5 S6 S7 S8 S8 S7 S7 S8 S7 S8 S7 S7 S8 S8 S7 S7 S6 S6 S7
```

Solución al Ejercicio 2

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
#include <fstream>

const char nombreFich[] = "evolucion.txt";

unsigned getIndice(std::string S) {
   if (S[0] != 'S' || S.size() < 2)
      return -1;
   std::stringstream ss;
   for (unsigned i=1; i<S.size(); i++) {
      if (S[i] < '0' || S[i] > '9')
         return -1;
      ss << S[i];
   }
   unsigned indice;
   ss >> indice;
   return indice;
}

int main() {
   std::ifstream inFich(nombreFich, std::ios::in);
   if (!inFich) {
      std::cerr << "Error al abrir el fichero\n";
      return 0;
   }
   std::string S;
   std::vector<unsigned> vS;
   while (inFich >> S) {
      unsigned indice = getIndice(S);
      if (indice < 0) {
         std::cerr << "Error. Estado no valido: " << S << "\n";
         return 0;
      }
      vS.push_back(indice);
   }
   inFich.close();
   unsigned indMax = 0;
   for (unsigned i=0; i<vS.size(); i++)
      if (vS[i] > indMax)
         indMax = vS[i];
   std::vector< std::vector<unsigned> >
                     Q(indMax, std::vector<unsigned> (indMax, 0));
   for (unsigned i=1; i<vS.size(); i++)
      Q[vS[i]-1][vS[i-1]-1]++;
   std::vector<unsigned> vCol(indMax, 0);
   for (unsigned i=0; i<indMax; i++)
      for (unsigned j=0; j<indMax; j++)
         vCol[j] += Q[i][j];
```

```
52    for (unsigned i=0; i<indMax; i++) {
53       for (unsigned j=0; j<indMax; j++)
54          if ( Q[i][j] )
55             std::cout << Q[i][j] << "/" << vCol[j] << "\t";
56          else
57             std::cout << Q[i][j] << "\t";
58       std::cout << "\n";
59    }
60    return 0;
61 }
```

Ejercicio 3

Dos tuberías están conectadas a un depósito de agua. A través de la primera tubería se introduce agua en el depósito y a través de la segunda se extrae agua del depósito. El caudal que circula por cada una de las tuberías es registrado en sendos ficheros de texto llamados *caudalEntrada.txt* y *caudalSalida.txt*, en los cuales se indica el instante de tiempo en el cual cambia el caudal y el nuevo valor del mismo. El caudal se mantiene constante entre los sucesivos instantes de cambio.

En los ficheros *caudalEntrada.txt* y *caudalSalida.txt*, cada línea describe un evento de cambio del caudal. En la primera columna se indica el instante de tiempo (en segundos) en que se produce el evento y en la segunda columna el nuevo caudal (en m^3/s). Los eventos están ordenados en orden cronológico. La primera línea de ambos ficheros corresponde al instante $t = 0$. Veamos a continuación un ejemplo.

El siguiente contenido del fichero *caudalEntrada.txt*

```
 0.0   1.2
10.0   0.3
23.2   0
27.8   0.9
```

indica que entre el instante $t = 0$ y el instante $t = 10$ s el caudal de entrada ha sido 1.2 m^3/s; entre el instante $t = 10$ s y el instante $t = 23.2$ s ha sido 0.3 m^3/s; entre el instante $t = 23.2$ s y el instante $t = 27.89$ s ha sido 0 m^3/s; y a partir del instante $t = 27.8$ s ha sido 0.9 m^3/s.

El siguiente contenido del fichero *caudalSalida.txt*

```
 0.0  0.2
 4.2  1.4
14.8  0.2
```

indica que entre el instante $t = 0$ y el instante $t = 4.2$ s el caudal de salida ha sido 0.2 m^3/s; entre el instante $t = 4.2$ s y el instante $t = 14.8$ s ha sido 1.4 m^3/s; y a partir del instante $t = 14.8$ s ha sido 0.2 m^3/s.

Escriba un programa en C++ que realice las acciones siguientes:

1. Mediante un mensaje escrito en la consola, solicitar al usuario que introduzca por consola el volumen de líquido (en m^3) que contiene el depósito en el instante $t = 0$. Leer dicho valor de consola y almacenarlo en una variable llamada `V0`. Si el valor leído es menor que cero, mostrar un mensaje en la consola indicándolo y terminar.

2. Mediante un mensaje escrito en la consola, solicitar al usuario que introduzca por consola el instante de tiempo (en segundos) para el cual se quiere calcular el volumen de agua contenido en el depósito. Leer dicho valor de consola y almacenarlo en una variable llamada `tFin`. Si el valor leído es menor que cero, mostrar un mensaje en la consola indicándolo y terminar.

3. Usando la información contenida en los ficheros *caudalEntrada.txt* y *caudalSalida.txt*, calcular el volumen de agua contenido en el depósito en el instante `tFin`, sabiendo que en el instante inicial ($t = 0$) hay un volumen `V0` de agua en el depósito.

   Si se produce error al abrir alguno de los ficheros, el programa debe indicarlo mediante un mensaje escrito en la consola y terminar.

4. Terminar.

Muestre en la memoria el resultado de ejecutar su programa para los siguientes valores introducidos por consola: `V0=10` y `tFin=120`; y los ficheros siguientes:

Fichero *caudalEntrada.txt*

```
  0.0   1.2
 10.0   0.3
 23.2   0
 27.8   0.9
 32.5   2.3
 44.9   1.6
 55.2   0.7
 60.0   0
 70.0   0
 75.9   0.5
 85.9  11.7
 97.2   0.4
110.9   2.2
130.7   1.2
147.8   0.3
160.0   0
```

Fichero *caudalSalida.txt*

```
  0.0   1.2
 10.0   0.3
 23.2   0
 27.8   0.9
 39.4   0.15
 50.0   0.8
 70     0.3
 75.9   0.6
100.0   0
110.0   2.3
120.4   6.8
122.4   0.1
150.0   0
```

Solución al Ejercicio 3

```
#include <iostream>
#include <vector>
#include <fstream>

const char nombreFich_Qin[]  = "caudalEntrada.txt";
const char nombreFich_Qout[] = "caudalSalida.txt";

struct Evento {
   double tiempo, caudal;
};

int main() {
   std::cout << "Volumen inicial de liquido (m3): ";
   double V0;
   std::cin >> V0;
   if (V0<0) {
      std::cerr << "Error. Valor no valido";
      return 0;
   }

   std::cout << "Instante de tiempo (s): ";
   double tFin;
   std::cin >> tFin;
   if (tFin<0) {
      std::cerr << "Error. Valor no valido";
      return 0;
   }

   std::ifstream inFich1(nombreFich_Qin, std::ios::in);
   if (!inFich1) {
      std::cerr << "Error al abrir el fichero " << nombreFich_Qin;
      return 0;
   }
   Evento evento;
   std::vector<Evento> vEv_Qin;
   while (inFich1 >> evento.tiempo >> evento.caudal)
      if ( evento.tiempo <= tFin )
         vEv_Qin.push_back(evento);
      else
         break;
   inFich1.close();

   std::ifstream inFich2(nombreFich_Qout, std::ios::in);
   if (!inFich2) {
      std::cerr << "Error al abrir el fichero " << nombreFich_Qout;
      return 0;
   }
   std::vector<Evento> vEv_Qout;
   while (inFich2 >> evento.tiempo >> evento.caudal)
      if ( evento.tiempo <= tFin )
         vEv_Qout.push_back(evento);
      else
         break;
   inFich2.close();
```

```
55
56    double V = V0;
57    for (unsigned i=1; i<vEv_Qin.size(); i++)
58       V += vEv_Qin[i-1].caudal *
59                (vEv_Qin[i].tiempo - vEv_Qin[i-1].tiempo);
60    V += vEv_Qin[vEv_Qin.size()-1].caudal *
61             (tFin - vEv_Qin[vEv_Qin.size()-1].tiempo);
62    for (unsigned i=1; i<vEv_Qout.size(); i++)
63       V -= vEv_Qout[i-1].caudal *
64                (vEv_Qout[i].tiempo - vEv_Qout[i-1].tiempo);
65    V -= vEv_Qout[vEv_Qout.size()-1].caudal *
66             (tFin - vEv_Qout[vEv_Qout.size()-1].tiempo);
67
68    std::cout << "Volumen (m3): " << V << std::endl;
69    return 0;
70 }
```

Ejercicio 4

Consideremos un autómata celular bidimensional compuesto por $N \times N$ células iguales, dispuestas formando una retícula cuadrada. Las filas se numeran, de arriba a abajo, de la 1 a la N; y las columnas se numeran, de izquierda a derecha, de la 1 a la N. El vecindario de cada célula lo constituyen 4 células: la situada inmediatamente a su izquierda, a su derecha, arriba y abajo. Las células ubicadas en las diagonales no forman parte del vecindario. El espacio se considera cíclico en ambas dimensiones: a la derecha de la columna N de la retícula está la columna 1, y debajo de la fila N está la fila 1.

Cada célula tiene dos posibles estados: "viva" o "muerta". El estado de todas las células se actualiza simultáneamente, en instantes de tiempo que llamaremos $t = 1$, $t = 2$, etc., aplicando la regla siguiente: se cuenta el número de células vecinas vivas y si este número es par (0, 2, 4) el nuevo estado de la célula es "muerta", y es "viva" en caso contrario.

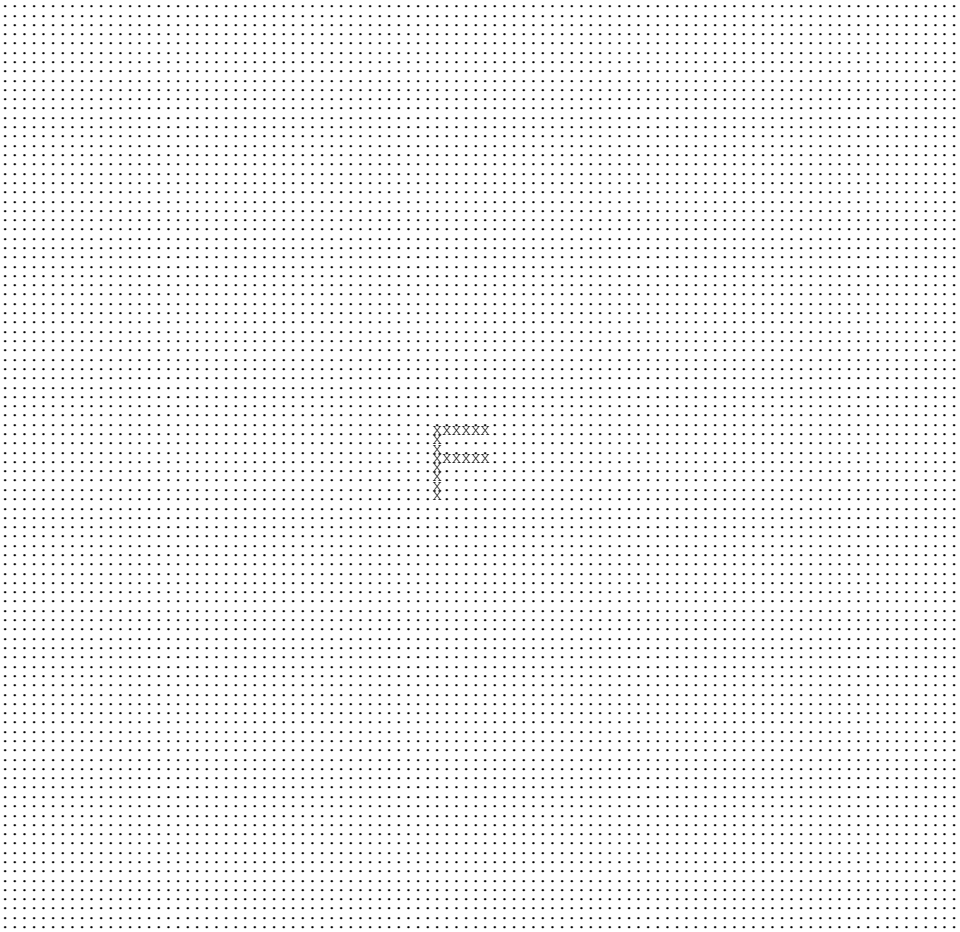
El estado inicial (en $t = 0$) de un autómata celular $N \times N$ está escrito en un fichero de texto llamado *patronInicial.txt* como N palabras (una por fila) de N caracteres: 'X' indica que la célula está viva y '.' que está muerta. En la página siguiente se muestra un ejemplo: el estado inicial de un autómata celular 100×100.

Escriba un programa en C++ que realice las acciones siguientes.

1. Leer, del fichero *patronInicial.txt*, el estado inicial del autómata.
2. Comprobar que el número de filas y columnas del autómata es el mismo, y que éste (N) es al menos 3. Si no se satisface alguna de estas condiciones, el programa debe indicarlo mediante un mensaje en la consola y terminar.
3. Escribir, en un fichero de texto llamado *evolucionCA.txt*, el estado del autómata en los instantes $t = 0$ (estado inicial), $t = 1$, $t = 2$, $t = 3$, ..., $t = 50$.
4. Terminar.

Ejecute su programa, empleando el fichero *patronInicial.txt* cuyo contenido se muestra en la página siguiente. Incluya en la memoria el listado del fichero *evolucionCA.txt* generado por su programa. Hágalo de manera que se muestre un estado por página, de forma similar a como se muestra el estado inicial en el enunciado del ejercicio.

Fichero *patronInicial.txt*



Solución al Ejercicio 4

```
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <algorithm>
#include <sstream>

const char nombre_inFich[]  = "patronInicial.txt";
const char nombre_outFich[] = "evolucionCA.txt";
const char VIVA   = 'X';
const char MUERTA = '.';
const unsigned minSIZE = 3;
const unsigned Nclicks = 50;

std::string logCA(std::vector<std::string> &CA) {
   std::stringstream ss;
   for (unsigned i=0; i<CA.size(); i++)
      ss << CA[i] << "\n";
   return ss.str();
}

bool checkSizeCA(std::vector<std::string> &CA) {
   const unsigned N = CA.size();
   if ( N < minSIZE ) return false;
   for (unsigned i=0; i<N; i++)
      if ( CA[i].size() != N ) return false;
   return true;
}

std::vector<std::string> nuevoEstado(std::vector<std::string> &CA) {
   const unsigned N = CA.size();
   std::vector<std::string> CAnew(N, std::string(N, MUERTA));
   for (unsigned i=0; i<N; i++)
      for (unsigned j=0; j<N; j++) {
         int numVivas = 0;
         if ( CA [(i==0)   ? N-1 : i-1] [j] == VIVA ) numVivas++;
         if ( CA [(i==N-1) ? 0   : i+1] [j] == VIVA ) numVivas++;
         if ( CA [i]   [(j==N-1) ? 0 : j+1] == VIVA ) numVivas++;
         if ( CA [i]   [(j==0) ? N-1 : j-1] == VIVA ) numVivas++;
         if ( numVivas%2 ) CAnew[i][j] = VIVA;
      }
   return CAnew;
}

int main() {
   std::ifstream inFich(nombre_inFich, std::ios::in);
   if (!inFich) {
      std::cerr << "Error al abrir el fichero";
      return 0;
   }
   std::string fila;
```

```
52    std::vector<std::string> CA;
53    while (inFich >> fila)
54      CA.push_back(fila);
55    inFich.close();
56    if (!checkSizeCA(CA)) {
57      std::cerr << "Error. CA no valido";
58      return 0;
59    }
60
61    std::stringstream ss;
62    ss << "t = 0\n" << logCA(CA);
63    for (unsigned i=1; i<=Nclicks; i++) {
64      CA = nuevoEstado(CA);
65      ss << "\nt = " << i << "\n" << logCA(CA);
66    }
67
68    std::ofstream outFich(nombre_outFich,
69                          std::ios::out | std::ios::trunc);
70    if (!outFich) {
71       std::cerr << "Error al abrir el fichero";
72       return 0;
73    }
74    outFich << ss.str();
75    outFich.close();
76    return 0;
77 }
```