

Facultad de Informática – Universidad Complutense
1º curso de los Grados
Fundamentos de la programación 2
Convocatoria Extraordinaria
Curso 2021–2022

Examen de junio de 2022

Tiempo disponible: 3 horas

Gran Hermano es una creación de George Orwell para una novela distópica llamada 1984, escrita entre 1947 y 1948.

El poder del Gran Hermano se basa en someter a la población a un estricto control, mediante una vigilancia total hasta de los más íntimos pensamientos. Para ello, existe un organismo llamado la Policía del Pensamiento que supervisa la correcta utilización de la *neolengua*. La *neolengua* es una adaptación del inglés que reduce y transforma el léxico con fines represivos, con la idea de que lo que no forma parte del lenguaje no puede ser pensado.

EJERCICIO 1 (3 PUNTOS)

Para vigilar la adhesión de la población a la *neolengua*, la Policía del Pensamiento cuenta con escuchas en todas partes (los hogares, los lugares de trabajo, los centros de ocio y las universidades) y penaliza el uso de palabras prohibidas.

En cada estancia, para poder vigilar la mayor superficie posible, se reparten escuchas de la siguiente forma: la estancia se parcela dividiéndose en una cuadrícula y se sitúan las escuchas en celdas escogidas de la cuadrícula. Cada escucha es capaz de cubrir su propia celda y las colindantes. La cobertura de lo que recoge la escucha es máxima sobre la celda en la que está situada (3 puntos), en las 8 más próximas baja a 2 y una fila o columna dos lugares más allá es 1. En la siguiente figura se puede ver un ejemplo de una estancia y la cobertura de la celda con una escucha en la posición (4,4) y en sus celdas colindantes:

	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	1	1	1	1	1	0	0
3	0	1	2	2	2	1	0	0
4	0	1	2	3	2	1	0	0
5	0	1	2	2	2	1	0	0
6	0	1	1	1	1	1	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

La fiabilidad de lo escuchado en cada celda de la estancia es la suma de las puntuaciones de todas las escuchas que tienen influencia sobre esa celda.

Se pide lo siguiente:

1. **(0,75 puntos)** Un subprograma que reparta aleatoriamente las escuchas en las celdas de una estancia:

```
void repartirEscuchas(tEstancia estancia, tListaCoord& escuchas);
```

El número de escuchas a repartir será una constante de programa ($N_ESCUCHAS = 10$). No puede haber más de una escucha por celda, por lo que si una celda ya está ocupada se ocupará la siguiente celda libre recorriendo de izquierda a derecha y de arriba abajo (al llegar a la esquina inferior derecha se volverá a subir a la superior izquierda). El subprograma devolverá una lista que contenga las coordenadas de todas las escuchas colocadas ($tListaCoord$).

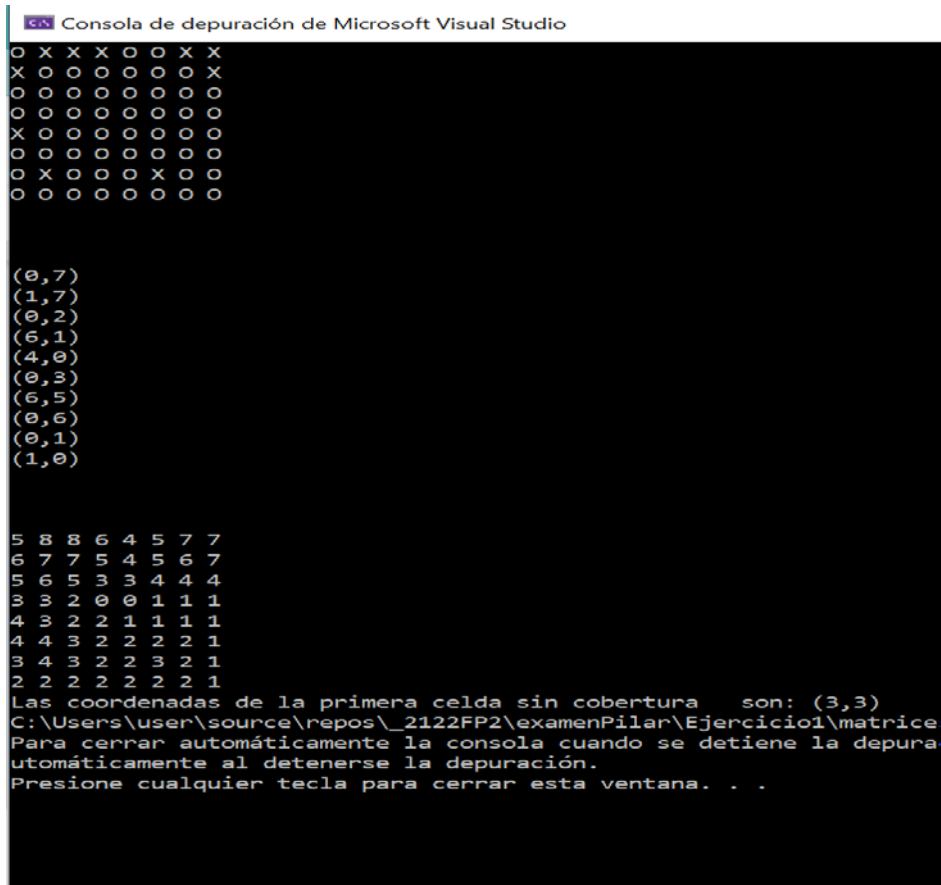
2. **(1,5 puntos)** Un subprograma que, dada una estancia y una lista de coordenadas con las escuchas, calcule la puntuación (puntos) de cada celda, una vez repartidas las escuchas.

```
void calcularCobertura(tEstancia estancia, const tListaCoord& escuchas);
```

3. **(0,75 puntos)** Un subprograma que devuelva las coordenadas de la primera celda sin cobertura, empezando por la esquina superior izquierda y yendo de izquierda a derecha y de arriba a abajo.

```
bool sinCobertura(tEstancia estancia, tCoord& celda);
```

Por último, implementa un main con el que pruebes los distintos subprogramas haciendo lo siguiente: repartir las escuchas en la estancia, visualizar la estancia indicando qué celdas tienen escucha y cuáles no, visualizar la lista de coordenadas de las escuchas repartida, calcular la cobertura en cada celda, visualizar la distribución de la cobertura en la estancia y mostrar las coordenadas de la primera celda sin cobertura. Aquí puedes ver un ejemplo de ejecución.



```
Consola de depuración de Microsoft Visual Studio
O X X X O O X X
X O O O O O O X
O O O O O O O O
O O O O O O O O
X O O O O O O O
O O O O O O O O
O X O O O X O O
O O O O O O O O

(0,7)
(1,7)
(0,2)
(6,1)
(4,0)
(0,3)
(6,5)
(0,6)
(0,1)
(1,0)

5 8 8 6 4 5 7 7
6 7 7 5 4 5 6 7
5 6 5 3 3 4 4 4
3 3 2 0 0 1 1 1
4 3 2 2 1 1 1 1
4 4 3 2 2 2 2 1
3 4 3 2 2 3 2 1
2 2 2 2 2 2 2 1
Las coordenadas de la primera celda sin cobertura son: (3,3)
C:\Users\user\source\repos\_2122FP2\examenPilar\Ejercicio1\matrice
Para cerrar automáticamente la consola cuando se detiene la depuración.
Presione cualquier tecla para cerrar esta ventana. . .
```

EJERCICIO 2 (7 PUNTOS).

El sistema universal de escuchas está enfocado a identificar quién ha pronunciado palabras prohibidas. Para ello se recogen los siguientes datos por cada celda de la ciudad (`tCelda`) en la se han recogido palabras que no pertenecen a la neolengua:

- `idCelda`: Código numérico (almacenado en un entero largo) que representa la identificación de las celdas donde se han escuchado palabras prohibidas (la ciudad completa es una retícula de celdas).
- `cobertura`: entero que recoge el nivel de cobertura en esa celda. Este número guarda relación con la fiabilidad de lo escuchado.
- `listaPersonas`: Lista de personas que estaban dentro de esa celda. Cada persona se identifica mediante un código alfanumérico (un `string`). La lista de personas (de tamaño variable) se almacenará en un array dinámico (`tListaPersonas`).

La lista con la información de aquellas celdas de la ciudad en las que se han escuchado palabras prohibidas se guarda en un array de punteros a datos dinámicos, con una dimensión máxima de `N_CELDAS`, pero que puede contener un número inferior de datos (`tListaCeldas`).

Por otro lado, la Policía del Pensamiento va registrando en una lista a los ciudadanos que han estado en estas celdas. A los ciudadanos de `tListaCeldas` se les penaliza con una puntuación igual a la cobertura de la celda en la que estaban. Con estos ciudadanos se conforma una nueva lista (`tListaPeligrosos`), soportada por un array estático que puede contener un número variable de ciudadanos (máximo número de elementos `CIUDADANOS = 100`).

El programa de vigilancia básica de la Policía del Pensamiento a desarrollar debe incluir los siguientes módulos y operaciones:

- **En el módulo `tCelda.h`:**

1. **(0,75 puntos)** Tipos de datos:

- `tListaCeldas`: Almacena la información recogida en todas las celdas de la ciudad (es un array de punteros a datos dinámicos, con un número variable de datos cuya dimensión máxima es `N_CELDAS = 100`).
- `tCelda`: Contiene la información de cada celda. En concreto, los enteros `idCelda`, `cobertura` y la lista con los ciudadanos que estaban en ella `listaPersonas` (un array dinámico de `strings -idCiudadano-` de capacidad ampliable, que cuenta inicialmente con `N_PERSONAS = 10` y que se va duplicando conforme se necesita).

2. **(1 punto)** Un subprograma para cargar la información de la lista de celdas de la ciudad a partir de un fichero de texto (`celdas.txt`):

```
bool cargar(tListaCeldas& listaCeldas)
```

El fichero de texto contiene la siguiente información:

- En la primera línea: el número de celdas de las que se han recogido datos (`n_celdas`).
- En las siguientes `n_celdas` líneas: un entero largo con el `idCelda`, un entero con la cobertura del celda, una lista con un número indeterminado de `strings` sin espacios (`idCiudadano`) representando a los ciudadanos que estaban en la celda. La lista de ciudadanos acaba con el centinela “xxx”.

Por ejemplo, el siguiente fichero contiene la información recogida en 3 celdas:

```
3
654646 5 id01 id02 id03 id07 id10 id12 id18 id21 id19 id23 id35 id48 id57 xxx
897982 3 id05 id10 id12 xxx
154654 9 id01 id02 id03 id07 id10 id12 id18 id21 id19 id23 id24 id29 id31 id35 id48
id57 xxx
```

A la hora de cargar el fichero, ten en cuenta que, si hay más ciudadanos que la dimensión inicial prevista, se expandirá el array dinámico al doble de su capacidad.

3. **(0,5 puntos)** Un subprograma para borrar la memoria dinámica que se ha reservado.

- En el módulo `tPeligrosos.h`:

1. **(0,25 puntos)** Tipos de datos:

- `tPeligrosos`: Almacena la lista de los ciudadanos peligrosos. Estará soportada por un array estático que puede estar parcialmente lleno (máximo número de elementos `CIUDADANOS = 100`). Cada elemento (`tCiudadano`) contiene la identificación del ciudadano (`idCiudadano`) y la penalización que acumula (puntuación). Se mantendrá ordenado por la identificación (`idCiudadano`).

2. **(0,5 puntos)** Un subprograma que cargue la lista inicial de ciudadanos peligrosos a partir de un fichero de texto (`peligrosos.txt`):

```
bool cargar(tPeligrosos& listaPeligrosos)
```

El fichero, ordenado por `idCiudadano`, contiene en cada línea el identificador del ciudadano y los puntos que tiene acumulados. Termina con el centinela "xxx".

```
id01 15
id02 19
id03 5
id07 9
id14 16
id19 1
id23 7
id34 5
xxx
```

3. **(3 puntos)** Un subprograma que, tomando de la lista de celdas a los ciudadanos que estaban presentes en ellas, actualice la información de `tPeligrosos`:

```
void actPeligrosos(tPeligrosos& listaPeligrosos, const tListaCeldas& listaCeldas);
```

Si el ciudadano no estaba todavía en la lista, se le insertará en su posición correspondiente (dándole los puntos de peligrosidad que tiene la cobertura de la celda). Si ya estaba, se le sumará la cobertura de la celda a su puntuación de peligrosidad.

- Para localizar a un ciudadano en la lista se empleará un subprograma auxiliar `buscarCiudadano` que utilice la búsqueda óptima **sabiendo que el array está ordenado**.
- Para insertar debes implementar un subprograma auxiliar `insertarCiudadano`.

4. **(0,5 puntos)** Un subprograma que sobrescriba el fichero de `peligrosos.txt` con la información actualizada de la lista, una vez se ha ejecutado el subprograma `actPeligrosos`.

(0,5 puntos) Implementa la función main donde se pueda probar todo: cargar la lista de celdas del fichero `celdas.txt`, cargar la lista de ciudadanos peligrosos del fichero `peligrosos.txt`, actualizar la lista de peligrosos con la información contenida en la lista de celdas y escribir en el fichero `peligrosos.txt` la información actualizada en la lista.

El siguiente ejemplo muestra cómo queda el fichero `peligrosos.txt` una vez se ha actualizado con la información de `celdas.txt`.

FICHEROS INICIALES:

`celdas.txt`

```
3
654646 5 id01 id02 id03 id07 id10 id12 id18 id21 id19 id23 id35 id48 id57 xxx
897982 3 id05 id10 id12 xxx
154654 9 id01 id02 id03 id07 id10 id12 id18 id21 id19 id23 id24 id29 id31 id35 id48 id57 xxx
```

`peligrosos.txt`

```
id01 15
id02 19
id03 5
id07 9
id14 16
id19 1
id23 7
id34 5
xxx
```

FICHERO FINAL DESPUÉS DE LA EJECUCIÓN:

`peligrosos.txt`

```
id01 29
id02 33
id03 19
id05 3
id07 23
id10 17
id12 17
id14 16
id18 14
id19 15
id21 14
id23 21
id24 9
id29 9
id31 9
id34 5
id35 14
id48 14
id57 14
xxx
```