PRÁCTICA DE LA ASIGNATURA
LABORATORIO DE PROGRAMACIÓN II
Curso 2008/09

Ingeniería Informática
Ingeniería Técnica en Informática de Sistemas
Ingeniería Técnica en Informática de Gestión

Roberto Rodríguez Echeverría (rre@unex.es)
Encarna Sosa Sánchez (esosa@unex.es)
José María Conejero (chemacm@unex.es)

PRISONBREAK REDUX

Entrega 2

Objetivos de la entrega

En esta segunda fase del proyecto, el objetivo es construir un prototipo del sistema final, que incluya principalmente la definición estructural del mismo. Este prototipo debe incluir la siguiente funcionalidad:

1. Creación de la prisión.
    1. Creación de cada planta según las características especificadas (dimensiones)
    2. Distribución de las celdas de cada planta mediante el algoritmo de Kruskal
    3. Mejora de la distribución mediante la introducción de “atajos”
2. Puertas y reparto de llaves
    1. Creación de la puerta de cada planta y configuración de la misma
    2. Reparto de las llaves en cada planta mediante la detección de los “cuellos de botella” de cada planta
3. Cálculo de la ruta de escape de un preso
4. Lectura del fichero de inicio: configuración inicial del sistema
5. Salida del sistema al fichero de registro según el formato especificado
6. Juegos de pruebas unitarias
    1. Juego de pruebas de cada unidad del sistema siguiendo un esquema de integración incremental

Creación de la prisión

Algoritmo de Kruskal. Este algoritmo parte de una estructura base de la planta (ver ilustración 1) en la que todas las celdas están aisladas (tienen paredes por todos lados) y su funcionamiento básico consiste en ir derribando paredes para crear pasadizos entre las celdas. El resultado del algoritmo será una distribución de celdas parecida a la mostrada en la ilustración 4. A continuación se explica de manera detallada el algoritmo:

- Inicialmente cada celda de la planta está marcada con un valor numérico diferente (coincidente con el identificador de la celda, tal y como se muestra en la ilustración 1).
- El algoritmo consiste en intentar derribar todas las paredes del laberinto, teniendo en cuenta que no puede derribarse una pared si separa dos celdas con la misma marca. En caso de separar dos celdas con marcas diferente, la pared puede ser derribada y las celdas que quedan conectadas deben ser marcadas con el mismo identificador. Si una de estas celdas estaba conectada con otras celdas (tenían la misma marca), todas las celdas conectadas por el camino creado quedarán marcadas con el mismo valor numérico (ver ejemplo en ilustración 3, marca 13). Mientras que, en caso de separar dos celdas con marcas iguales, la pared no se derriba. Evidentemente, las paredes exteriores de la planta no se pueden derribar en este proceso.
- La selección de una pared para ser derribada se debe hacer aleatoriamente tomando como base el conjunto de paredes que existen en la planta. Hay que tener en cuenta que cada vez que se seleccione una pared, sea o no derribada finalmente, se elimina de este conjunto para no poder ser seleccionada de nuevo. Inicialmente, las paredes deben almacenarse en el conjunto siguiendo el orden de los identificadores de celda, es decir, primero se almacenan las paredes de la celda 0, después las paredes de la celda 1, hasta haber almacenados las paredes de la última celda de la planta. Además, de las 4 paredes posibles que puede tener una celda, se almacenan siguiendo el orden N, E, S, O (Norte, Este, Sur, Oeste).
- Este proceso se repite mientras existan paredes cuyo derribo no ha sido aún valorado.
- Al final, todas las celdas compartirán la misma marca.

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29
30	31	32	33	34	35

Ilustración 1: Estructura base para laberinto

0	1	2	3	4	5
6	7	8	9	10	11
12	13	8	15	16	17
18	20	20	21	22	23
24	25	26	27	28	29
30	31	32	33	34	35

Ilustración 2: Planta tras derribar dos paredes

0	1	2	3	10	5
6	7	13	9	10	11
12	13	13	15	16	17
18	20	20	21	22	23
24	31	26	27	28	29
30	31	32	33	35	35

Ilustración 3: Proceso de derribo de paredes en la planta

33	33	33	33	33	33
33	33	33	33	33	33
33	33	33	33	33	33
33	33	33	33	33	33
33	33	33	33	33	33
33	33	33	33	33	33

Ilustración 4: Resultado final del algoritmo

Para la generación de atajos dentro de una planta, se derribarán n paredes de la planta elegidas de forma aleatoria, siendo n el 5% del número total de celdas de la planta (si hay 100 celdas en la planta, hay que derribar 5 paredes). Al igual que en el algoritmo de Kruskal, las paredes exteriores de la planta no se pueden derribar en este proceso. El proceso de selección de la pared a tirar es el siguiente:

1. Se elige una celda aleatoriamente
2. Se busca un vecino no accesible (existe pared entre la celda elegida (CE) y el vecino), comprobando los vecinos, como muestra la tabla 1, en el siguiente orden: Norte, Sur, Oeste y Este.

	1	
3	**CE**	4
	2	

Tabla 1: Orden de selección de vecinos

2. Si se encuentra vecino no accesible y se puede tirar la pared, sin crear espacios vacíos, se tira la pared. Un espacio vacío se forma cuando 4 celdas están unidas sin existir ninguna pared entre ellas. Situaciones en las que se crearían espacios vacíos:

NO N	N NE
O **CE**	**CE** **E**

Tabla 2: Tirar pared entre CE y vecino Norte

O **CE**	**CE** E
SO **S**	**S** **SE**

Tabla 3: Tirar pared entre CE y vecino Sur

NO **N**	**O** **CE**
O **CE**	**SO** **S**

Tabla 4: Tirar pared entre CE y vecino Oeste

N **NE**	**CE** **E**
CE **E**	**S** **SE**

Tabla 5: Tirar pared entre CE y vecino Este

1. Si no hay vecino accesible o la pared no se puede tirar, se vuelve al paso 1

Reparto de llaves en las celdas de una planta

La generación de llaves se realizará de la misma manera que para la primera entrega. De manera que, dado un número de llaves para cada planta, en la puerta de cada planta se insertarán las llaves con código impar. Por otro lado se generará un juego completo de llaves en el que se repitan las de código impar para repartir por las celdas de la planta. El número de llaves de cada planta se especificará en el fichero de configuración inicial.

El algoritmo de reparto tiene como objetivo depositar las llaves en las celdas que son más frecuentadas por los presos en sus rutas de escape. Dada una planta, se deben calcular todas las rutas de escape de la misma y obtener una relación de las celdas que más aparecen en las diferentes rutas ordenadas de más a menos frecuente. Una vez se dispone de esta relación, se irán depositando las llaves de 5 en 5 empezando por la celda más frecuentada. El reparto termina cuando no haya más llaves que repartir o más celdas en las que depositar llaves. Por ejemplo, si el número de llaves a repartir es de 45, sólo las 9 celdas más frecuentadas de la planta contendrán llaves.

Cálculo de la ruta de escape de un preso

En esta entrega no es necesario implementar la especificación de preso, pero sí hay que implementar el algoritmo que calcula la ruta de escape de un preso. El algoritmo debe realizar un recorrido en profundidad desde la celda de entrada a la de salida del grafo formado por la distribución de celdas en cada planta. A partir de este recorrido debe generar una secuencia de orientaciones (N, S, E, O) que le permita al preso, posteriormente, moverse desde la celda de entrada de la planta a la celda de salida de la misma.

Cuando el algoritmo se encuentre con una celda que conecta con varias debe ir eligiéndolas usando la siguiente secuencia: Norte, Este, Sur y Oeste. Sin embargo, no siempre empieza a comprobar por la primera posición (Norte), sino que se basará en la orientación elegida en el movimiento anterior, empezando por la siguiente a ésa. Por ejemplo, supongamos que el preso se encuentra en la celda 46 y que esta celda permite seguir hacia el Este (47) o hacia el Sur (56). Supongamos, además, que el último movimiento se ha producido de la celda 45 a la 46, eligiendo la orientación Este. El algoritmo comenzaría a comprobar su secuencia en la posición Sur (siguiente a la Este). Como la opción Sur es una opción posible, el preso pasaría a la celda 56 (Sur) y no a la 47 (Este). **Modificado - 20/04/2009**

Fichero de inicio

La configuración inicial del sistema (plantas, presos, llaves, etc.) se cargará al comienzo de la ejecución del sistema mediante la lectura de un fichero de texto que contiene esta configuración en un formato concreto. Cada línea del fichero de configuración definirá los detalles de una determinada entidad del sistema a simular. Así, cada línea tendrá un primer token que define el tipo de entidad (PLANTA, PRESO, etc.) y después una lista de tokens con los detalles de configuración de esa entidad. Cada uno de estos tokens estará separado por un carácter #.

Pueden existir además líneas adicionales de comentario que no serán tenidas en cuenta en el proceso (comienzan con --).

Para esta entrega las únicas entidades que aparecerán en el fichero de configuración son las plantas que forman parte de la prisión a simular. Los campos que forman la especificación de una planta son: identificador, ancho, alto, celda entrada, celda salida, número de identificadores de llaves a generar y altura de control del ABB de la cerradura. A continuación se muestran varios ejemplos:

PLANTA#0#10#5#3#49#20#4#
Planta con identificador 0, con 10 celdas de ancho y 5 de alto, cuya entrada está en la celda 3 y salida en la 49, y en la que se van a generar 20 llaves (sin incluir repeticiones). La altura de control del ABB de la cerradura es 4 (condición de apertura).

--Ejemplo de comentario
Cualquier línea que comience por los caracteres –- representa un comentario en el fichero de configuración y por tanto no será tenida en cuenta en el proceso del concurso.

Un ejemplo de fichero de inicio podría quedar de la siguiente manera:

```
--Plantas de la prision
--Planta baja
PLANTA#0#10#5#3#49#20#4#
--Planta intermedia
PLANTA#1#6#6#0#18#10#3#
--Planta alta
PLANTA#2#10#10#0#99#30#4#
```

Con el objetivo de centrar el trabajo en los puntos más interesantes, se proporcionará el código necesario para el procesamiento básico de este fichero de inicio. Este código deberá ser extendido por cada uno para contemplar la creación de cada una de las posibles entidades de su sistema.

Fichero de registro

Para que quede constancia de todo lo que ocurra en el sistema, se ha decidido registrar en un fichero de texto todos los datos de la simulación. Para esta entrega sólo se tendrán en cuenta las acciones de creación de la prisión (estructura) y el cálculo de las rutas de escape.

Estructura de la prisión
En primer lugar, al inicio de la simulación, para cada planta se guardará en este registro el estado de la distribución de sus celdas antes de generar los atajos. En el fichero se guardará una representación de la planta similar al que se muestra a continuación:
(planta:0)

```
 _ _ _ _ _ _ _ _ _ _
| | |_ _  | |    _  |
|  _ _|   |_  |_ _| |
|  _   _|_ _      | |
|  _|_ _  |  _| |_| |
|  _ _|   |   | |   |
|_ _|  _|_|_|_|_|_|_|
|    _|_      |  _ _|
| | | | | |_|_  |  _|
| |    _ _   _|_  | |
|_|_|_|_ _ _ _|_ _|
```

Posteriormente, se guardará una representación de la planta una vez generados los atajos.

```
 _ _ _ _ _ _ _ _ _ _
| | |_ _  | |    _  |
|  _ _|    _  |_ _| |
|  _   _|  _      | |
|  _|_ _  |  _| |_| |
|  _ _|       | |   |
|  _|  _|_|_| |_|_|_|
|    _|_      |  _ _|
| | | | | |_|_  |  _|
| |    _ _   _|_  | |
|_|_|_|_ _ _ _|_ _|
```

Después, se mostrará cómo han quedado repartidas las llaves en las celdas de la planta, mostrando sólo las celdas que contienen llaves ordenadas por su frecuencia. Por ejemplo, si tenemos 30 llaves para la planta anterior:

(celda:10: 0 1 1 2 3)
(celda:20: 3 4 5 5 6)
(celda:66: 7 7 8 9 9)
(celda:76: 10 11 11 12 13)
(celda:77: 13 14 15 15 16)
(celda:87: 17 17 18 19 19)
(celda:88: 20 21 21 22 23)
(celda:98: 23 24 25 25 26)
(celda:22: 27 27 28 29 29)

Modificado - 14/04/2009

Finalmente, se indicará la secuencia de orientaciones que componen la ruta de escape para esa planta. Por ejemplo, una posible ruta de escape de la celda 0 a la 99 en la distribución anterior sería:

(ruta: S S E E S E E S O S O S O S S E E E N N E E S E S E S E)

Esta información se repetirá para cada una de las plantas de la prisión.

Para ver un ejemplo completo de fichero de registro, se recomienda usar los generados por el ejecutable demo proporcionado por los profesores.

Consideraciones

- El nombre del fichero de inicio se especificará como un parámetro del ejecutable, según la sintaxis: <nombre_del_programa> <nombre_fichero>

Ejemplo: `./prisionbreak inicio.txt`

- El ejecutable debe admitir **dos modos de ejecución: normal y pruebas**. Por defecto, el ejecutable está en modo normal y realiza la ejecución de la implementación del sistema. El modo de ejecución se puede cambiar a pruebas mediante un parámetro opcional del ejecutable. En modo pruebas, el sistema ejecuta las pruebas unitarias definidas para el sistema en lugar de la funcionalidad del mismo. El parámetro que indica el modo de pruebas aparece al final de los parámetros de ejecución y contiene la cadena “test”. Se trata de un parámetro opcional, por lo tanto, puede aparecer o no. Si no aparece, la ejecución se realizará en modo normal. La sintaxis de ejecución del sistema quedará: <nombre_del_programa> <nombre_fichero> [test]

Ejemplo: `./prisionbreak inicio.txt test`

- La salida por pantalla no tiene un formato predefinido, cada alumno puede hacerla como quiera.

- El fichero de registro debe denominarse **registro.log.**

- El contenido del fichero de **registro** resultante debe ser **contrastado** con el proporcionado por los profesores **de forma automática**, usando diff, kompare o meld.

- El proyecto **no** debe contener **ficheros fuente que no formen parte del ejecutable final.**

- El programa entregado debe ejecutar de principio a fin **sin pedir ninguna tecla.**

- Se debe usar el mecanismo de **excepciones** para gestionar los errores del sistema.

- Es **obligatorio el uso del grafo** como estructura de soporte de la distribución de las celdas en una planta.

- Para la generación de números aleatorios se debe utilizar una clase creada para tal efecto por los profesores. Esta clase se publicará dentro de los materiales de esta entrega.

- Se deben seguir las recomendaciones apuntadas en enunciados anteriores.

- Opcionalmente, como adelanto de trabajo, se puede implementar el comportamiento y gestión de presos dentro del prototipo. Se recomienda, no obstante, que se aborde esta tarea una vez terminadas completamente las tareas establecidas para esta segunda entrega.

SEGUNDA PRUEBA DE LA EVALUACIÓN CONTINUA (EC2)

De cara a la realización final de la práctica, la segunda prueba se centrará en la realización de algoritmos sobre un grafo. La entrega previa a esta modificación presenta los siguientes requisitos:

- La implementación de las clases necesarias para satisfacer las especificaciones del sistema establecidas en este enunciado y en los anteriores.
- Uso común de las estructuras de datos de la STL.
- Manejo de errores basado en excepciones. Definición y uso de, al menos, 5 tipos de excepción diferentes.
- La implementación de la estructura de datos grafo con los algoritmos necesarios para la entrega.
- Uso del patrón de diseño Singleton.
- Implementación de las pruebas unitarias correspondientes al prototipo.
- Generación de la documentación de análisis y diseño correspondientes al prototipo.

La implementación debe seguir los conceptos de Programación Orientada a Objetos vistos en clase y debe gestionar correctamente la memoria dinámica.

Cada alumno deberá realizar la **entrega de esta segunda fase del proyecto antes de presentarse a la modificación**. El código de la entrega contendrá **exclusivamente el código necesario para satisfacer las especificaciones** de la misma y nada más. Para realizar esta entrega se habilitará una **tarea en el aula virtual**. Posteriormente, el alumno **realizará la prueba de modificación sobre el mismo código que haya entregado** en el aula virtual.

Requisitos sobre el código de la EC2:

- El código debe estar perfectamente documentado con la notación doxygen y debe generar la documentación correctamente.
- El código debe estar escrito siguiendo un único estilo de programación.
- En la documentación interna del código debe indicarse:

  Nombre y Apellidos:

Asignatura:
Grupo:
Número de Entrega:
Curso:

Antes de la prueba se entregará:

a) Una carpeta cuyo nombre sea igual al identificador de correo-e del alumno (sin "@alumnos.unex.es"). Dentro de esta carpeta se incluirán solamente los ficheros fuente que formen parte del ejecutable final **(*)**.

Al final de la prueba se entregará:

a) Una carpeta cuyo nombre sea igual al identificador de correo-e del alumno (sin "@alumnos.unex.es"). Dentro de esta carpeta se incluirán solamente los ficheros fuente que formen parte del ejecutable final **(*)**.

En ambos caso, la carpeta creada será comprimida en un fichero en formato **ZIP** con el mismo nombre. La entrega de dicho fichero comprimido se realizará a través de una actividad propuesta en Avuex, de forma que cada alumno subirá a la plataforma virtual dicho fichero. Es muy importante que cada alumno suba su archivo con su cuenta en Avuex.

(*) Nota: deben eliminarse de los proyectos entregados aquellos ficheros que no son necesarios para su corrección, ejemplo: ficheros objeto y fichero ejecutable.