From Prog1 - InCo

Laboratorio: SegundaTarea2009

Segunda Tarea. Programación 1

Para obtener una versión apropiada para impresión, siga el vínculo **imprimir**.

En esta página... (ocultar)

1. Introducción

Este documento presenta el problema que deberá resolverse para la aprobación de la segunda tarea del laboratorio del curso 2009.

Se presenta información acerca de: normas, recursos, plazos, una presentación general del problema, las especificaciones del mismo, ejemplos de ejecución y la forma de entrega.

---

2. Información general

El estudiante que no respete alguna de las consideraciones que siguen corre el riesgo de que su trabajo sea invalidado, con la consiguiente pérdida del curso.

Compilador

Todos los programas deben ser compatibles con el compilador del curso (**Free Pascal**). No se aceptarán como válidas aquellas tareas que pudieran funcionar con algún otro compilador Pascal, pero no funcionen con Free Pascal, versión 2.2.2 para windows.

Individualidad

Esta segunda tarea se debe realizar de manera individual.
Para todas las tareas rige el **Reglamento del Instituto de Computación ante Instancias de No Individualidad en los Laboratorios**. A continuación se adjunta un fragmento del mismo; ante cualquier duda se recomienda leer el documento completo (**http://www.fing.edu.uy/inco/cursos/prog1/pm/field.php/Laboratorio/NoIndividualidad**)

> Los laboratorios deben ser realizados únicamente por los integrantes del grupo establecido La realización de los laboratorios es estrictamente individual, sea a nivel unipersonal en el primer caso, o del grupo establecido en el segundo. Se entiende que compartir total o parcialmente cualquier actividad del laboratorio atenta contra la integridad del estudiante universitario y de su formación, y por lo tanto constituye una falta grave. Específicamente no es posible compartir por ninguna vía entre integrantes de grupos distintos las tareas de codificación, digitación, compilación, depuración y documentación de los programas u objetos (o entregas) del laboratorio. Además de que no se pueden compartir actividades del laboratorio, no se pueden compartir los productos de las mismas. Cada grupo es responsable de su trabajo de laboratorio y de que el mismo sea individual, independientemente de las causas que pudiesen originar la no individualidad. A modo de ejemplo y sin ser exhaustivos: utilización de código realizado en cursos anteriores (por otros estudiantes) u otros cursos, perder el código, olvidarse del código en lugares accesibles a otros estudiantes, prestar el código o dejar que el mismo sea copiado por otros estudiantes, dejar la terminal con el usuario abierto al retirarse, enviarse código por mail, utilizar código suministrado por terceros, etc. Asimismo se prohíbe el envío de código al grupo de noticias del curso, dado que el mismo será considerado como una forma de compartir código y será sancionado de la manera más severa posible.

Forma de entrega

Las entregas se realizarán por la *web*. Para ello se deberá acceder a un sitio destinado a tal fin y seguir los pasos que se explicarán en la página correspondiente. Esta página estará disponible cuando comience el plazo de entrega.

Fecha de Entrega

Se debe entregar desde el **9/11** hasta el **13/11**. El plazo vence el 13/11 a las 12 de la noche (medianoche del viernes al sábado)

---

3. Presentación

El buscaminas es un conocido juego de computadora que simula un campo de minas que debe ser *despejado* por el jugador, marcando la posición de todas las minas sin *pisar* ninguna.

Incluímos una descripción de las reglas tomada de la Wikipedia (**http://es.wikipedia.org/wiki/Buscaminas**)

> El juego consiste en despejar todas las casillas de una pantalla que no oculten una mina.
>
> Algunas casillas tienen un número, este número indica las minas que suman todas las casillas circundantes. Así si una casilla tiene el número 3 significa que de las ocho casillas que hay alrededor (si no está en una esquina o borde) hay 3 con minas y 5 sin minas. Si se descubre una casilla sin número indica que ninguna de las casillas vecinas tiene mina y estas se descubren automáticamente.
>
> Si se descubre una casilla con una mina se pierde la partida.

Versión para el entorno KDE

3.1 Acciones Básicas

Existen muchas versiones del buscaminas que pueden diferir en algunos aspectos. La mayoría de ellas incluye los siguientes movimientos por parte del jugador:

- **Descubrir una celda**. Esta acción se realiza sobre una celda oculta. El resultado depende del contenido de la celda:
  - Si la celda contiene una mina el juego se pierde.
  - Si la celda está libre pero hay minas en las celdas circundantes, aparece un número (entre 1 y 8) que indica la cantidad de celdas circundantes que contienen una mina.
  - Si la celda está libre y no hay minas en ninguna de las celdas circundantes, la celda se descubre y también se descubren automáticamente todas las casillas circundantes, aplicando el mismo procedimiento para cada una de ellas. Este procedimiento deja al descubierto toda un área de casillas libres adyacentes.
- **Marcar una celda**. Esta acción se realiza sobre una celda oculta que el jugador supone que contiene una mina; suposición que puede ser correcta o no. Independientemente de ello, el sistema muestra la celda como marcada e incrementa el contador de celdas marcadas.
- **Desmarcar una celda**. Esta acción se realiza sobre una celda marcada. El resultado es desmarcar la celda y dejarla como oculta. y decrementar el número de celdas marcadas.

3.2 Acciones complementarias

Varias versiones del buscaminas también incorporan las siguientes facilidades:

- **Despejar circundantes de una celda**. Esta acción se realiza sobre una celda ya descubierta. El resultado es descubrir todas las celdas circundantes que no estén marcadas. Por "*descubrir*" entendemos ejecutar la acción *descubrir una celda* explicada más arriba. La acción se lleva a cabo, solamente si el número que contiene la celda es igual a la cantidad de celdas circundantes marcadas.
- **Marcar circundantes de una celda**. Esta acción se realiza sobre una celda ya descubierta. El resultado es marcar todas las celdas circundantes no descubiertas. La acción se lleva a cabo, solamente si el número que contiene la celda es igual a la suma de la cantidad de celdas circundantes ya marcadas más las celdas circundantes no descubiertas sin marcar.
- **Descubrir una celda segura**. Es una ayuda que solicita el jugador. El resultado es que una celda sin bomba se descubre. En general la solicitud de esta ayuda es penalizada de alguna manera por ejemplo sumando segundos (en esta tarea no tendremos en cuenta el tiempo empleado por el jugador)

3.3 Terminación

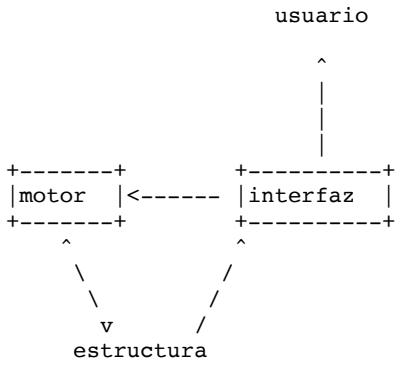
El juego termina cuando se da alguna de las dos siguientes situaciones:

- Se descubre una celda que contiene una mina. En este caso el jugador pierde el juego.
- Se descubrieron todas las celdas que no contienen minas. No importa si se marcaron o no todas las celdas con minas. En este caso el jugador gana el juego.

---

4. Arquitectura del sistema

El sistema que implementa el juego se construirá de acuerdo con la siguiente arquitectura:



Este es un modelo simple donde tenemos dos capas o módulos: la interfaz y el motor.

La **interfaz** se encarga de realizar el diálogo con el usuario (el jugador), capturar sus entradas e invocar las operaciones asociadas. La interfaz también maneja la imagen del juego en pantalla actualizándola cada vez que hay cambios. En resumen, se encarga de todo lo que tenga que ver con entrada y salida. La interfaz no realiza ninguna modificación directa sobre la estructura de datos.

El motor es el módulo que trata con la estructura de datos que representa el juego en un estado determinado. Este módulo estará compuesto por todos los subprogramas necesarios para ejecutar las acciones del jugador y reflejar estas en la estructura de datos. El motor no realiza ninguna operación de entrada y salida.

En esta tarea, el estudiante implementará solamente el motor. La interfaz será provista por los docentes. En las siguientes secciones explicamos los detalles del módulo a ser implementado.

---

5. La estructura

La estructura de datos que representa el juego de buscaminas es la siguiente:

```
type
  TipoEstadoJuego = (jugando,ganado,perdido);
  TipoJuego = record
      estado       : TipoEstadoJuego;
      tablero      : TipoTablero;
      bombas,                   (* cantidad de bombas en el tablero *)
      marcadas,                 (* cantidad de celdas marcadas *)
      descubiertas : integer;   (* cantidad de celdas descubiertas *)
  end;
```

El *juego* es un registro con los siguientes campos:

- `estado`: Contiene el estado del juego en un momento dado.
- `tablero`: el tablero con las celdas
- `bombas`: cantidad de bombas en total en el tablero
- `marcadas`: cantidad de bombas ya marcadas por el usuario
- `descubiertas`: cantidad de celdas que están descubiertas

El tablero

El tablero se respresenta con la siguiente estructura:

```
RangoFila    = 1..MAX_FILAS;
RangoColumna = 1..MAX_COLUMNAS;
TipoTablero = record
   celdas  : array[RangoFila,RangoColumna] of TipoCelda;
   topeFila    : RangoFila;
   topeColumna : RangoColumna;
end;
```

Esta estructura es una matriz con dos *topes* (ver *array con tope*). De esta manera se pueden representar tableros de diferentes tamaños. Las celdas válidas de la matriz son aquellas cuyas coordenadas (i,j) son tales:

- `1<= i <= topeFila`
- `1<= j <= topeColumna`

Las constantes `MAX_FILAS` y `MAX_COLUMNAS` se suponen definidas con valores apropiados. Estos valores están definidos en la interfaz, de manera que el estudiante no necesita conocerlos.

La celda

La celda se representa con la siguiente estructura:

```
TipoEstadoCelda = (oculta,marcada,descubierta);
TipoCelda = record
   estado : TipoEstadoCelda;
   case tieneBomba :Boolean of
      True  : ();
      False : (bombasCircundantes :integer)
   end;
```

La información que tiene una celda es la siguiente:

- `estado` : indica si la celda fue marcada, descubierta o continúa oculta (cuando decimos *oculta* nos referimos a *oculta y sin marca*).
- `tieneBomba`: es un booleano que vale `true` si y sólo si la celda contiene una mina.
- `bombasCircundantes`: indica la cantidad de bombas que hay en las celdas circundantes. Este valor puede ser 0.

6. Los subprogramas

El motor del juego está constituido por un conjunto de subprogramas que trabajan sobre la estructura definida. En varios de los subrogramas se utiliza un parámetro que representa la posición de una celda dentro del tablero. Para ello necesitamos definir el siguiente tipo:

```
TipoPosicion = record
     fila   : RangoFila;
     columna:  RangoColumna;
end;
```

Cualquiera de los subprogramas pedidos más abajo retorna sin ejecutar nada cuando recibe una posición *no válida*.

A continuación se describen los subprogramas que constituyen el motor del buscaminas:

Inicialización

Este subprograma es invocado al comienzo del juego para generar el tablero inicial.

```
procedure IniciarJuego(var juego: TipoJuego;
                           cuantas_filas: RangoFila; cuantas_columnas: RangoColumna; cuantas_bombas: integer);
```

Los parámetros que recibe son:

- `cuantas_filas`: cantidad de filas que va a tener el tablero
- `cuantas_columnas` : cantidad de columnas que va a tener el tablero
- `cuantas_bombas`: cantidad de bombas que va a tener el tablero

Este procedimiento debe retornar el parámetro `juego` configurado para iniciar el juego:

- Todas las celdas quedan ocultas.
- Las bombas se colocan al azar en tantas celdas como lo indique el parámetro `cuantas_bombas`. La colocación al azar se realiza utilizando la función `random` (ver apéndice **Números Randómicos**)
- Las celdas sin bombas deben tener el campo `bombasCircundantes` con el valor apropiado.
- Los demás campos del juego (estados, contadores, topes, etcétera) se deben inicializar apropiadamente

En caso que el parámetro `cuantas_bombas` tenga un valor negativo o sea mayor que el número de celdas, el estudiante puede decidir el comportamiento del procedimiento.

Descubrir una celda

```
procedure Descubrir(var juego: TipoJuego; posicion: TipoPosicion);
```

Esta operación realiza sobre el parámetro `juego` la acción de *descubrir una celda* tal como se describe más arriba. La celda a descubrir es la indicada por el parámetro `posicion`. En caso de que la celda ya esté descubierta, o la posición no sea válida, el procedimiento no hace nada.

Tener en cuenta que cuando se descubre una celda con 0 bombas circundantes, se produce automáticamente el descubrimiento de las celdas adyacentes para las cuales a su vez se aplica el mismo criterio. Esto puede provocar el descubrimiento de toda un área de celdas adyacentes, situación seguramente muy apreciada por el jugador.

En las versiones clásicas del buscaminas, esta operación se realiza con un *clic* con el botón principal sobre una celda no descubierta.

Marcar y Desmarcar

```
procedure Marcar(var juego: TipoJuego; posicion: TipoPosicion);
procedure DesMarcar(var juego: TipoJuego; posicion: TipoPosicion);
```

Estos dos procedimientos corresponden a las acciones de marcado y desmarcado de una celda tal como se describió en la sección Acciones Básicas.

En las versiones clásicas del buscaminas, esta operación se realiza con un *clic* con el botón secundario del ratón sobre una celda no descubierta.

Despejar Celdas Circundantes

```
procedure DespejarCircundantes(var juego: TipoJuego; posicion: TipoPosicion);
```

Este procedimiento corresponde a la acción *Despejar Circundantes de una celda* tal como se explicó en la sección Acciones Complementarias.

En las versiones clásicas del buscaminas, esta operación se realiza con un *clic* con el botón principal del ratón sobre una celda descubierta.

Marcar Celdas Circundantes

```
procedure MarcarCircundantes(var juego: TipoJuego; posicion: TipoPosicion);
```

Este procedimiento corresponde a la acción *Marcar Circundantes de una Celda* tal como se explicó más arriba.

Esta operación no está implementada en las versiones clásicas del buscaminas.

Descubrir Celda Segura

```
procedure DescubrirSegura(var juego: TipoJuego);
```
Este procedimiento corresponde a la acción *Descubrir Celda Segura* tal como se explicó más arriba.

Este operación no está implementada en las versiones clásicas del buscaminas.

7. Se pide

Escribir un archivo con todos los subprogramas que forman el *motor* del juego.

> Los cabezales de los subprogramas deben **coincidir exactamente** con los que aparecen en esta letra. Si el estudiante realiza algún cambio se considerará que el subprograma no fue implementado.

El lunes 19/10 se publicará la interfaz implementada por el equipo docente, junto con instrucciones de cómo compilar y ejecutar el programa compuesto por la interfaz y el motor.

Se puede utilizar todo lo visto en las clases teóricas y prácticas.

Para la corrección de las tareas, se compilarán con la versión 2.2.2 para windows. La compilación y la ejecución se realizarán en línea de comandos. El comando de compilación se invocará de la siguiente manera:

```
fpc -Co -Cr programa.pas
```

Si trabaja con el IDE, asegúrese de configurarlo para que compile de la misma manera que el comando anterior (habilitación de *range checking* e *Integer Overflow Checking*).

> No está permitido utilizar facilidades de Free Pascal que no forman parte del estándar y no se dan en el curso. Así por ejemplo, no se pueden utilizar ninguna de las palabras siguientes: `uses`, `crlscr`, `gotoxy`, `crt`, `readkey`, `longint`, `string`, `break`, etcétera.

En esta tarea como en todos los problemas de este curso, se valorará además de la lógica correcta, la utilización de un buen estilo de programación de acuerdo a los criterios impartidos en el curso. De esta manera, se hará énfasis en buenas prácticas de programación que lleven a un código legible, bien documentado y mantenible, tales como:

- indentación adecuada
- utilización correcta y apropiada de las estructuras de control
- código claro y legible
- algoritmos razonablemente eficientes
- utilización de comentarios que documenten y complementen el código
- utilización de constantes simbólicas
- nombres mnemotécnicos para variables, constantes, etcétera.

8. Apéndices

Números randómicos

En free pascal se pueden generar números al azar utilizando la función `random`. Esta función recibe un parámetro entero positivo y retorna un número aleatorio mayor o igual que 0 y menor que el parámetro recibido.

El procedimiento `randomize` debe invocarse una sola vez, previamente al cualquier invocación de la función `random`. El único objetivo de esta operación es inicializar el generador de números aleatorios.

A continuación se muestra un ejemplo donde se generan 10 números al azar entre 1 y 6, como si fueran lanzamientos de un dado.

```
{ inicialización }
randomize;
for i:= 1 to 10 do
begin
  { sorteo }
   numero:= random(6) + 1;

   writeln(numero);
end;
```

Tener en cuenta que la función `random` puede repetir números.

Guía para descubrir celda

El procedimiento `Descubrir` tiene algunas complejidades algorítmicas que requieren pensarlo con cuidado. En este apéndice damos algunas sugerencias para su implementación.

Si la celda a descubir contiene una bomba el juego termina. Si la celda a descubrir no contiene bomba y la cantidad de circundantes con bomba es mayor que cero, se descubre sólo esta celda y continúa el juego.

La situación más compleja surge cuando la celda no contiene bomba y la cantidad de circundantes con bomba es igual a 0. Para este caso sugerimos un algoritmo que permite explorar exhaustivamente el área circundante a descubrir.

Se utiliza una lista de posiciones de celdas que llamaremos la *lista de pendientes*. En esta estructura guardamos un conjunto de posiciones de celdas que aún nos resta procesar en el proceso de descubrir toda el área circundante. La implementación de esta estructura la dejamos a consideración del estudiante.

El algoritmo es el siguiente:

- Poner el estado de la celda inicial como `descubierta`
- Guardar posición de esta celda en lista de pendientes.
- Repetir mientras la lista de pendientes no sea vacía
    1. Elegir una celda cualquiera de la lista de pendientes. Quitarla de la lista
    2. Para cada una de sus celdas circundantes aún no descubiertas (pueden ser hasta 8) hacer lo siguiente:
        i. Poner el estado en `descubierta`
        ii. Si el número de circundantes con bomba es 0, agregar la posición de esta celda a la lista de pendientes

---