Laboratorio 2010

Principios de Programación

Introducción

Este documento presenta el problema que deberá resolverse para la aprobación del laboratorio del curso 2010.

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.

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 (turbo C). No se aceptarán como válidas aquellas tareas que pudieran funcionar con algún otro compilador distinto. El SO ha utilizar será Windows XP.

El compilador fue entregado a la casilla del grupo vespertino: tecnogrup2010@gmail.com el dia 14 de abril y esta instalado en los salones. Se encuentra para bajarlo en el siguiente link http://www.fing.edu.uy/tecnoinf/cursos/prinprog/material/varios/turboc.zip

Individualidad

Esta tarea se debe realizar de manera individual
Para todas las tareas rige el Reglamento del Instituto de Computación de Facultad de Ingeniería(Udelar) 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.

Descripción del Juego Batalla Naval.

La batalla naval es un conocido juego, donde dos jugadores disponen de dos tableros cada uno. En uno de los tableros el jugador dispone en las celdas de este sus distintos tipos de barcos, que el otro jugador deberá tratar de ir encontrando en cada turno del juego. En el otro tablero será usado por el jugador para ir registrando las zonas del mapa del jugador contrario que vaya descubriendo en cada turno(en cada ataque). Cuando un jugador descubra todos los barcos enemigos, será el ganador del juego.

¿Que hay que implementar?

Al comenzar cada partida, cada jugador debe primero que nada debe ubicar sus barcos en su mapa. Para esto cada jugador debe disponer de un archivo de estrategias creado previamente, que con ayuda de librerías brindadas por los docentes, podrá cargar en memoria en una estructura de datos adecuada para su utilización (esto se describe más abajo en este documento). Luego de que cada jugador dispuso todos sus barcos (hay distintos tipos de barcos que ocupan distinta cantidad de celdas, todas ellas consecutivas horizontalmente o verticalmente).Para este laboratorio habrá tres tipos de barcos, nos referiremos a ellos cómo barcos de tipo a, tipo b, y tipo c. Tanto los barcos de tipo a y b ocuparan dos celdas consecutivas, y los de tipo c tres. Utilizaremos los símbolos ‘a’,’b’,’c’ para identificar estos tipos de barcos en el tablero. Es decir, por ejemplo si un barco de tipo a esta ubicado en el mapa en las coordenadas (0,1) y (0,0), en el tablero en esas posiciones se encontrara el símbolo ‘a’. El juego se desarrolla por turnos. En el momento que cada jugador dispone de su turno, elije una posición en el mapa enemigo, que no haya sido seleccionada antes para atacarlo. En el momento del ataque puede ocurrir que no de a ningún blanco (llamémosle a esta situación “agua”), o que acierte a una sección de un barco enemigo. Si un barco enemigo es atacado completamente (son descubiertas todas las celdas que este ocupa), el programa debe notificar al jugador que hizo el ataque, que ha hundido completamente un barco. El juego finaliza cuando uno de los dos jugadores haya hundido todos los barcos enemigos. Este jugador será el ganador de la partida.

Nuestro tablero tendrá una cantidad de filas y de columnas igual a **TAM_TABLERO,** tal que 4< **TAM_TABLERO** <10.Las filas y columnas se enumeran empezando desde cero hasta **TAM_TABLERO-1.**

Estructuras Principales.

La estructura de datos que se utiliza en el juego en está primera entrega es la siguiente:

```
typedef enum tipo_celda={OCULTA='*',AGUA='|',BUQUE1='B',
BUQUE2='C',PORTA_AVIONES='V' };
```

Información General del laboratorio

El laboratorio presentado, esta pensado para ser implementado en 3 etapas, con sus respectivos entregables cada una de ellas.

Se pide

Primera entrega

En esta primera entrega, se deberá implementar una serie de funciones y procedimientos, que luego podrán o no ser reutilizados, en las sucesivas etapas del laboratorio. Los estudiantes deberán entregar en un archivo llamado batnav1_nroCI.cpp (donde nroCI es el número de cedula del estudiante sin puntos ni guiones), la implementación de un conjunto de funciones y procedimientos descritos a continuación (cinco en total), junto con una función main que contenga código que sirva para probar el correcto funcionamiento de cada una de las funciones implementadas.

A continuación se describen las funciones a implementar:

- **Inicializar tablero**

void inicializarTablero(tipo_celda tablero[][TAM_TABLERO],char inicial),

Los parámetros que recibe son:

- `tablero`: Matriz de caracteres que contiene los elementos que se pueden encontrar en un tablero de la batalla naval.
- Inicial: símbolo inicial con que se inicializaran todas las celdas del tablero.

Este procedimiento debe:

- Fijar un valor dado en el parámetro inicial, en todas las celdas del tablero.

- **Imprimir Tablero**

void imprimirTablero(tipo_celda tablero[][TAM_TABLERO]),

Los parámetros que recibe son:

- `tablero`: Matriz de caracteres que contiene los elementos que se pueden encontrar en un tablero de la batalla naval.

Este procedimiento debe :

- Imprimir en pantalla el contenido de tablero, utilizando la primer columna y la primer fila para mostrar los números de fila y columnas respectivamente. Por ejemplo:

Esta instancia del tablero muestra que en la fila uno columnas 0 y 1 hay ubicado un barco de tipo a, el la fila 5, columnas 3,4, y 5 hay un barco de tipo c, y el la columna 5 filas 7 y 8 un hay un barco de tipo b, las restantes celdas del tablero no han sido descubiertas, tener en cuenta que las celdas ocultas se representan median le símbolo de '*', y si se descubre una celda que no contiene barcos, esta celda la identificamos con el símbolo '|' (agua).

- **Obtener Tipo elemento en una posición dada**

tipo_celda obtenerTipoElemento(tipo_celda tablero[][TAM_TABLERO],int fila,int columna),

Los parámetros que recibe son:

- `tablero`: Matriz de caracteres que contiene los elementos que se pueden encontrar en un tablero de la batalla naval.
- Fila,columna: posiciones entre 0 y TAM_TABLERO-1, del tipo de elemento que se quiere conocer, contenido en la celda en la fila y columna respectivas

Esta función debe:

- Retornar el tipo contenido en la matriz, en la fila y columnas pasadas cómo parámetros.

- **Asignar Tipo elemento en una posición dada**

void asignarTipoElemento(tipo_celda tablero[][TAM_TABLERO],int fila,int columna, tipo_celda tipoNuevo),

Los parámetros que recibe son:

- `tablero`: Matriz de caracteres que contiene los elementos que se pueden encontrar en un tablero de la batalla naval.
- Fila, columna: posiciones entre 0 y TAM_TABLERO-1, del tipo de elemento que se quiere almacenar en la celda en la fila y columna respectivas
- tipoNuevo: Es el tipo de elementos de juego (agua,oculta,etc…) que queremos que haya en la posición (fila,columna) de la matriz.

Este procedimiento debe:

- establecer el nuevo tipo de contenido en la matriz, en la fila y columnas pasadas cómo parámetros.

- **Obtener la cantidad de celdas adyacentes consecutivas dado un tipo de celda dados una posición y un sentido**

bool cantidadCeldasAdyacentes(tipo_celda tablero[][TAM_TABLERO],int fila,int columna, tipo_celda tipo, bool sentido, int &cant1, int &cant2),

Los parámetros que recibe son:

- `tablero`: Matriz de caracteres que contiene los elementos que se pueden encontrar en un tablero de la batalla naval.
- fila, columna: posiciones entre 0 y TAM_TABLERO-1, del tipo de elemento que se quiere almacenar en la celda en la fila y columna respectivas
- tipo: Es el tipo de elementos de juego (agua,oculta,etc…) que queremos saber si se encuentra en la posición (fila,columna) de la matriz, y a partir de la posición donde buscaremos la cantidad de celdas adyacentes que haya del mismo tipo.
- Sentido: En caso de ser verdadero, indica que se quiere contar la cantidad de celdas adyacentes en la fila pasada cómo parámetro a partir de la posición (fila,columna) tanto a la derecha (esté valor se devolverá en el parámetro por referencia cant1) cómo a la izquierda(cant2), si es falso indica que se quiere contar la cantidad de celdas adyacentes en la columna pasada cómo parámetro a partir de la posición (fila, columna) tanto hacia arriba-posiciones menores de

columna- (esté valor se devolverá en el parámetro por referencia cant1) cómo hacia abajo-valores mayores de columna-(cant2).
- Cant1, y cant2: utilizados según la descripción anterior.

Esta función debe:

- Retornar verdadero si el tipo de celda tipo pasado cómo parámetro es el que se encuentra en la matriz en la posición (fila, columna)
- Devolver en los parámetros variables cant1,cant2 la cantidad de celdas consecutivas del tipo de celda, tipo pasado cómo parámetro a partir de la posición (fila, columna) a la derecha(cant1) y a la izquierda(cant2) si el parámetro sentido es trae, de lo contrario la cantidad de celdas consecutivas del tipo de celda, tipo pasado cómo parámetro, a partir de la posición (fila, columna) hacia arriba(cant1) y hacia abajo(cant2).

Forma de entrega

Se indicara por medio del docente responsable la forma de entrega

Fecha de entrega

10/05/2010