Escuela Superior de Ingeniería y Tecnología
Universidad de La Laguna

Grado en Ingeniería Informática
Algoritmos y Estructuras de Datos Avanzadas

Curso 2020-2021

Práctica 4: Mundo Hormigas Modificado

Objetivo

El objetivo de esta práctica es utilizar el polimorfismo dinámico y el manejo de excepciones en tipos de datos definidos por el usuario en un programa en lenguaje C++.

Entrega

Esta práctica se entregará en la sesión de laboratorio realizada entre las fechas del 22 al 26 de marzo. Durante esta sesión se podrán solicitar modificaciones sobre la práctica. En el actual escenario de presencialidad adaptada, la entrega se realizará en tareas separadas para la rotación con asistencia presencial y la rotación con asistencia online.

Enunciado

A partir de la implementación de la práctica 3, **Mundo Hormigas**, en esta práctica se pide realizar las siguientes modificaciones:

1. En la práctica anterior se añadieron las siguientes operaciones a la clase **Hormiga**:

   a. El método privado `Girar(Color)`, que a partir de la dirección guardada y el color de la celda que ocupa, aplica la regla para el cambio de dirección.

   b. El método privado `Desplazar()`, que realiza el cambio de posición de la hormiga.

   c. El método público `Actualizar(Mundo&)`, que es invocado desde el bucle que controla el paso del tiempo en el objeto **Universo**. Este método pregunta al objeto **Mundo** por el color de la celda que ocupa la hormiga, le pide al objeto **Mundo** que actualice el color de dicha celda, e invoca a los métodos `Girar()` y `Desplazar()`.

   En esta práctica se pide que el método `Actualizar()` pase a ser un método virtual, y que los métodos `Girar()` y `Desplazar()` pasen a ser métodos nulos. De esta forma la clase **Hormiga** se convierte en una clase abstracta que utilizaremos como clase base para derivar una jerarquía de diferentes tipos de hormigas con distintas reglas de actualización. Implementar, al menos, dos tipos de hormigas:

   - Tipo a. Tiene el comportamiento de la Hormiga de Langton.
   - Tipo b. Tiene un comportamiento inverso a la Hormiga de Langton Modificada.

2. Se mantienen las dos versiones alternativas de implementación del objeto **Mundo**, un mundo finito y un mundo infinito. La clase **Mundo** es la clase base de las clases derivadas que implementan los mundos alternativos.

   La responsabilidad de cualquiera de las clases derivadas de la clase **Mundo** es gestionar la malla bidimensional de celdas que contiene los colores utilizados por las reglas de actualización de los objetos **Hormiga**. Para la implementación de la malla en el objeto **MundoInfinito** se utilizará la clase genérica `Vector<T>`, indicada en la práctica 3, que admite la indexación de sus elementos con valores negativos.

3. En la clase genérica `Vector<T>` se utiliza el mecanismo de manejo de excepciones para indicar la situación de error que se produce al intentar acceder a una posición del vector fuera del rango de valores de índices válidos. La clase `Vector<T>` genera un objeto derivado de la clase `std::exception` y lanza la excepción. En el objeto **Mundo** se captura la excepción y se realiza la acción que corresponda para tratarla:

   a. Si se trata del objeto **MundoFinito** actualiza la posición de la hormiga para que vuelva a estar posicionada sobre la malla.

   b. Si se trata del objeto **MundoInfinito** crea una nueva malla con las dimensiones actualizadas para que la hormiga quede bien posicionada.

4. El programa principal solicita al usuario que elija el tipo de mundo, así como sus dimensiones. También pregunta el número de hormigas que coexisten en el mundo, y para cada una de ellas solicita el tipo a instanciar y la posición que ocupa inicialmente. Se crea el objeto **Mundo** y la lista de punteros a la clase base **Hormiga**, de forma que para cada hormiga se aplique la regla de actualización que corresponde a su tipo. El objeto **Mundo** y la lista de objetos **Hormiga** se pasan como parámetros al constructor del objeto **Universo** para que realice la simulación.