EIE
Escuela de Ingeniería Eléctrica

Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

UNIVERSIDAD DE COSTA RICA

`IE-0217` Estructuras abstractas de datos y algoritmos para ingeniería

Laboratorio 1: Juego de ecología en C++ (II)

M. Sc. Ricardo Román Brenes - `ricardo.roman@ucr.ac.cr`

II-2018

Tabla de contenidos

---

1. Enunciado

A la simulación del laboratorio anterior, agregue la siguiente funcionalidad:

- Agregue una clase `Vegetal` que represente diferentes tipos de plantas. Haga que la clase Pasto herede de esta y además cree al menos 3 clases más como por ejemplo `Uva`, `Zanahoria`, `Roble` que también heredan de `Vegetal`. Cada planta tendrá su cantidad máxima de energía, y podrá ser comida o no por alguno tipo de `Animal`.
- Agregue una clase `Animal` que agrupe los diferentes tipos de fauna. Haga que las clase `Conejo`, `Oveja`, `Lobo` y `Zorro` hereden de esta. Modifíquelas como sea necesario para realizar este procedimiento.
- Cada `Celda` ahora tendrá entonces un atributo de tipo `Animal` y otro de tiempo `Planta` en vez de utilizar varios atributos de cada uno de los tipos de animales previos, esto con el fin de utilizar la herencia y el polimorfismos de las objetos.
- Existen nuevas reglas de interacción entre animales y plantas:
  1. Los Conejos recupera el doble de energía al comer Zanahoria que al comer Pasto, y no pueden comer Robles.
  2. Las Ovejas no puede comer Robles.

3. Los Zorros pueden comer Uvas y recuperan la energía que normalmente daría el Pasto a un herbívoro.
4. Los Ratones puede comer todo menos robles y ganan la misma cantidad de energía, como si comieran pasto.
5. Los Robles ganan una cuarta parta de energía por día que el pasto comienzan con la cuarta parte de su energía máxima. Cuando alcanzan su energía máxima, hacen que el terreno sea no transitable.
6. Las Uvas y Zanahorias crecen a la mitad de la velocidad del Pasto.

- Agregue el código necesario para que su simulación puede procesar archivos de configuración con toda esta nueva información.
- Agregue e implemente métodos virtuales puros a sus clases para obtener la posición en la que se encuentran y su estado.
- Sobrecargue el operador $<$ para cuando un animal come a otro.
- Sobrecargue el operador $\hat{}$ para cuando un animal come plantas.
- Sobrecargue el operador $\tilde{}$ para cuando un animal muere.
- Sobrecargue el operador $*$ para cuando un animal se reproduce.

Todas las reglas anteriores aplican, a menos que alguna de las nuevas la reescriba.

Al inicio y al final de la simulación es necesario que imprima en pantalla un reporte de los animales disponibles en el mapa. Para esto construya una **función** que reporte, el identificador, la especie, el sexo y las coordenadas del Animal.

En cada día de ejecución de la simulación imprima el estado del juego de una forma clara en pantalla.

Documente su trabajo utilizando la herramienta *doxygen* y cree un diagrama de clases UML de su trabajo; así como los archivos README e INSTALL.

Utilice un Makefile para facilitar la construcción y prueba de su programa.

Recuerde utilizar todas las capacidades vistas de C++ (a menos que sea necesario otra cosa (C)).

2. Consideraciones

- Haga grupos de hasta 3 personas.
- Genere un reporte en LaTeX que incluya su código, su abordaje para la solución y sus conclusiones.
- Suba su código y documentación (doxygen, README, INSTALL) al git respectivo de su grupo y el directorio del laboratorio.
- Cada estudiante debe subir el reporte a Schoology.
- Recuerde que por cada día tardío de entrega se le rebajaran puntos de acuerdo con la formula: 4^d, donde $d > 1$ es la cantidad de días tardíos.