

# El rescate perfecto

Algoritmos y programación II - 2C2022

Trabajo práctico Nº3

Grupal



## Introducción

---

A la mayoría de la gente le gustan los animales y hay muchas personas dispuestas a cuidarlos y darles un hogar. Después de mucho ahorrar, se logró abrir una reserva para todos los animales que lo necesiten. Pero al tener tantos es difícil llevar un registro de todo...

## Enunciado

---

Luego de crear la reserva y rescatar algunos animales los rescatistas se dieron cuenta que debían hacer unos pequeños cambios:

Una vez leído el archivo se deberán guardar todos los animales en un **diccionario** implementado con un **árbol B de 3 vías** con el nombre del animal como clave y sus datos como valor, el mismo deberá ser hecho por completo por los alumnos. **NO** está permitido el uso de STL.

El menú inicial mantendrá las mismas opciones que para el trabajo practico anterior y agregaremos una nueva:

1. Listar animales.
2. Rescatar animal.
3. Buscar animal.
4. Cuidar animales.
5. Adoptar animal.
6. Cargar combustible.
7. Guardar y salir.

### Listar animales

Se deberán listar todos los animales con su nombre, edad, tamaño, especie en forma completa<sup>1</sup>, personalidad, hambre e higiene.




### Rescatar animal

Ahora la reserva consiguió un auto y puede ir a buscar los animales que necesitan ser rescatados. El auto comenzará con 100 de combustible y se le cargará 5 por cada acción del menú principal que sea seleccionada.

El mapa se representará en una matriz de 8x8, cada casilla de éste puede ser de distinto tipo:

- Montaña
- Precipicio
- Camino
- Tierra

Dependiendo el tipo de casillero el costo de pasar por ese casillero variará.

-  Montaña: costo de 5 de combustible.
-  Precipicio: costo de 40 de combustible.
-  Camino: costo de 1 de combustible.

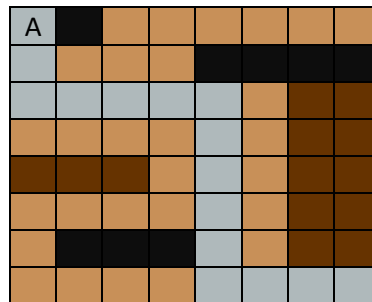
---

<sup>1</sup> Se debe poner la palabra completa en lugar del caracter.

■ Tierra: costo de 2 de combustible

### Mapa

El mapa del juego seguirá el siguiente diseño:



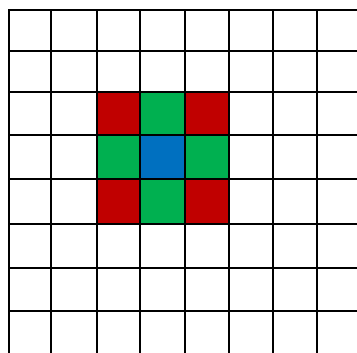
El auto siempre comenzará en la esquina superior izquierda y se generarán de forma aleatoria (tanto sus características como posición) 5 animales que pueden ser rescatados, para rescatarlos el auto se deberá desplazar a su posición y solo podrá hacerlo si cuenta con el combustible suficiente.

NOTA: los animales se mostrarán en el mapa con la letra que los represente, por ejemplo, si es un perro se mostrará una P en la posición del mapa en la que se encuentre.

### Camino mínimo

Para moverse de un lugar a otro, siempre se sigue el camino por el cual se gasta la menor cantidad de energía posible.

El auto puede moverse a casillas aledañas, pero no puede desplazarse en diagonal en un sólo movimiento, es decir:



- Auto
- Movimiento permitido
- Movimiento no permitido

Para moverse de un lugar A a un lugar B, el auto deberá tener el combustible suficiente para arribar a destino, en caso contrario, el auto no se desplazará y el usuario podrá elegir si realizar otra acción o moverse a otro destino. Para encontrar el camino de menor consumo de energía se deberá utilizar lo visto en clase sobre grafos.

**NOTA:** Deberá notarse claramente cuál fue el camino seguido por el auto.

### Buscar animal

Se deberá solicitar el nombre del animal al usuario y validar que el mismo este entre los animales de la reserva. Si el animal existe se le mostrará la información de este.

En caso de que no exista un animal con el nombre ingresado se le deberá mostrar un error acorde al usuario.

### Cuidar animales

Si un animal llega a 0 de higiene o 100 de hambre se escapará de la reserva y en caso de que 3 animales se escapen la misma se verá clausurada y la partida terminará.

Se mostrará un nuevo menú con las siguientes opciones:

1. Elegir individualmente.
2. Regresar al inicio

Si se elige la opción 1 se deberá mostrar los datos de cada animal individualmente y preguntar si desea, bañar, alimentar, saltar al mismo o volver al inicio. **Si el animal ya fue adoptado no debe aparecer.**

Al alimentar a cualquier animal su hambre volverá a 0 y deberá decir su nombre y lo que comió.

Al bañar a cualquier animal su higiene volverá a 100 y si no requiere un baño deberá decirlo.

Para volver al menú inicial se deberá seleccionar la opción 4.

### Adoptar animal

Se deberá preguntar el espacio disponible para el animal (en metros cuadrados) y mostrar todos los animales de la lista que podrían ser adoptados, es decir, si tengo 5 metros cuadrados disponibles debo mostrar todos los animales diminutos y pequeños con su nombre, edad, especie y personalidad.

La reserva busca fomentar la adopción de animales mayores por lo que a partir de ahora, se deberán mostrar los animales que cumplan con las condiciones **ordenados por edad de mayor a menor**. Para ello se deberá usar alguno de los métodos de ordenamiento vistos en clase.

Se le deberá pedir al usuario que elija entre alguno de los animales o si desea cancelar la adopción. Si un animal es adoptado deberá ser marcado como adoptado, pero no es necesario removerlo del árbol.

**NOTA IMPORTANTE:** En caso de decidir eliminarlo y hacerlo correctamente se darán 5 puntos extra. En caso de decidir no eliminar dicho animal deberá aparecer con estado ADOPTADO al momento de listar y buscarlo, pero NO aparecerá a la hora de cuidar de los animales.

### Cargar combustible

Al seleccionar esta opción nos deberá decir el combustible actual y preguntar cuanto combustible deseamos agregar, no podemos tener más de 100 de combustible.

### Guardar y salir

Al terminar de usar el programa se deberá actualizar<sup>2</sup> el archivo con los valores correspondientes.

---

<sup>2</sup> Si los valores cambiaron se reescribirá el archivo manteniendo el formato y nombre de este y notificando si surge algún error a la hora de abrir o escribir en el mismo.

## UML

Se deberá incluir como mínimo un UML de las clases creadas y sus relaciones entre sí.

## Aclaraciones

---

- ✚ El trabajo es de carácter **grupal**. Los grupos estarán compuestos por **4** integrantes.
- ✚ Se deberá usar Github, Gitlab o alguna otra plataforma en la que puedan generar un repositorio al cual subir su código y permita ver las estadísticas de dicho repositorio. **Se tendrá en cuenta la participación de cada individuo en el repositorio**. Además, el repositorio deberá ser privado.
- ✚ El trabajo no cuenta con reentrega.
- ✚ Los animales deben guardarse en una lista.
- ✚ Se debe hacer uso de herencia y polimorfismo.
- ✚ Los archivos están bien formados.
- ✚ No recorrer más de una vez los archivos.
- ✚ Se deben validar los datos ingresados por el usuario y mostrar errores que sean fáciles de comprender por cualquier persona.
- ✚ No se deben subir archivos de configuración de los IDEs (**solo subir .cpp y .h**).
- ✚ El trabajo debe compilar con los flags **-Wall -Werror -Wconversion**.
- ✚ No se permiten usar bibliotecas de templates como por ejemplo STL.

## ¿Qué se evaluará?

---

- ✚ Compilación
- ✚ Funcionalidad
- ✚ Manejo de archivos
- ✚ Usabilidad (que tan sencillo es de utilizar para un usuario)
- ✚ Eficiencia espacial
- ✚ Eficiencia temporal
- ✚ Correcto uso de memoria dinámica.
- ✚ Herencia y polimorfismo.
- ✚ Correcto uso de programación orientada a objetos.
- ✚ Árboles.
- ✚ Grafos.
- ✚ Buenas prácticas de programación (nombres descriptivos, indentación, etc.)
- ✚ Modularización.
- ✚ Precondiciones y postcondiciones.
- ✚ UML.

## *Normas de entrega*

---

Se deberá subir al campus un único archivo comprimido (.zip) en la sección TPs.  
Este archivo deberá tener un nombre formado de la siguiente manera:

**NombreGrupo\_TP3**

El nombre del grupo debe ser una única palabra, por ejemplo:

dinamicos\_TP3.zip

Deberá contener solo los archivos fuente. Es decir, solo .cpp y .h. NO subir los archivos de configuración de sus IDEs. (por ejemplo: CMakeList y cmake-build para Clion, .vscode para VisualStudioCode).

La fecha de **entrega** vence el **jueves 24/11/2022** a las 23.55hs.

**Puntaje:** 80 puntos.