INF-253 Lenguajes de Programación
Tarea 2: C

Profesor: José Luis Martí Lara, Roberto Diaz Urra
Ayudante Cátedras: Gabriela Acuña Benito, Hugo Sepúlveda Arriaza,
Lucio Fondón Rebolledo
Ayudante Tareas: Gabriel Carmona Tabja, Domingo Benoit Cea,
Héctor Larrañaga Jorquera, Ignacio Ulloa Huenul,
Joaquín Gatica Hernández, Javier Pérez Riveros,
José Runín Basáez, Rafael Aros Soto
Rodrigo Pérez Jamett

8 de octubre de 2021

1. Un problema ambiental

Un informático anónimo no solo era fan de Mario, sino que también era fan de las simulaciones. Él recuerda una simulación clásica que corresponde a situar conejos y zorros en una matriz cuadrada. Los conejos y zorros se podían mover libremente una vez por jugada, además en cada jugada los conejos y zorros se pueden reproducir, y en el caso de que un zorro se junte con un conejo, el conejo es comido por el zorro. Este informático anónimo se volvió loco y decidió reproducir la simulación, pero con más animales y opciones.

2. Mundo

Este mundo en el cual se hará la simulación será representado por una matriz cuadrada de 1000x1000. Cada casilla podrá ser ocupada por un animal. Además, en este mundo se definirán diferentes tipos de animales con distintas características que serán ingresados por consola.

2.1. Animales

Cada animal corresponderá a una estructura que contiene tres atributos básicos (fuerza, velocidad y resistencia) y cada uno tiene asociado a su vez un caracter que le especifica su tipo, si es un entero se almacenará “e”, si un caracter se almacenará “c” y si es un flotante se almacenará “f”. También tiene dos punteros a funciones que serán: reproducción y comerHui.

```
typedef struct Animal{
  void* fuerza;
  char tipo_fuerza;
  void* velocidad;
  char tipo_velocidad;
  void* resistencia;
```

```
  char tipo_resistenca;
  void (*reproduccion)(struct Animal*, struct Animal*, struct Animal*);
  void (*comerHuir)(struct Animal*, struct Animal*);
} Animal;
```

2.2. Comparación de dos atributos iguales

Para comparar los diferentes tipos de atributos, se hará una operación que permitirá tener todos los valores de tipo entero. Si el atributo es entero se considera el valor como tal. Si el atributo es caracter se considera el valor ASCII del caracter y se aplica división entera por 4. Si el atributo es flotante se redondea el valor. Y así se pueden comparar todos los tipos como enteros.

3. Funciones a implementar

Las siguientes funciones serán las que se deben implementar para la tarea.

- CrearAnimal: se debe recibir la variable donde se creará el animal (puntero), donde adentro se le deberá pedir los atributos por consola, por ende deben pedir el tipo que voy a ingresar y el valor. Y también que funciones usar (más adelante se detalla al respecto).
- BorrarAnimal: se debe recibir un animal y eliminarlo.
- MostrarAnimal: se debe recibir un animal y mostrarlo por pantalla, de la siguiente forma:

  ```
  fuerza: el valor
  velocidad: el valor
  resistencia: el valor
  ```

- Reproducir: debe reproducir dos animales entre ellos. Para saber, que función de reproducción usar deben elegir al azar, donde hay un (50 % de elegir la función del primer animal y 50 % de elegir la función del segundo animal, debe mostrar por pantalla cual fue el animal que uso la función).

  OJO: si se elige la función del primer animal se debe pasar como parámetro primero el primer animal y segundo el otro animal, si se elige la función del segundo animal se debe pasar como parámetro primero el segundo animal y segundo el otro animal.

- ComerOHuir: debe realizar la acción de alguno de los animales. Para saber, que función de comerHuir usar deben elegir al azar, donde hay un (50 % de elegir la función del primer animal y 50 % de elegir la función del segundo animal), debe mostrar por pantalla cual fue el animal que uso la función y resultado de está).

  OJO: si se elige la función del primer animal se debe pasar como parámetro primero el primer animal y segundo el otro animal, si se elige la función del segundo animal se debe pasar como parámetro primero el segundo animal y segundo el otro animal.

- Comparar: Debe recibir dos animales. La comparación se hará de la siguiente forma, se comparará cada atributo de los animales respectivamente (fuerza con fuerza, velocidad con velocidad, resistencia con resistencia) usando lo explicado en la sección 2.2. Luego, si el primer animal tiene más atributos mayores se retorna 0, sino se retorna 1.
- BorrarMundo: debe borrar todos los animales del mundo y el mundo mismo.
- MostrarMundo: debe mostrar cada animal dentro del mundo.

Las siguientes funciones corresponden a funciones que los Animales deben tener para reproducirse y comerHuir, como se indicó más arriba. Estas cinco funciones son para que vean que su programa funciona, teóricamente hablando se le podría pasar cualquier otra función para reproducción o comerHuir.

- ReproduccionSimple: debe recibir dos animales (a1 y a2) y el animal a hijo a crear. La reproducción se hará de la siguiente forma, se comparará los animales gracias a la función comparar de más arriba. Luego, si el valor que retorna es 0 entonces el animal hijo recibe todo del animal a1, sino el animal hijo recibe todo del animal a2.
- ReproduccionCruzada: debe recibir dos animales (a1 y a2) y el animal a hijo a crear. La reproducción se hará de la siguiente forma, se comparará los animales gracias a la función comparar de más arriba. Luego, si el valor que retorna es 0, entonces el animal hijo recibe la fuerza, velocidad y función de reproducción del animal a1 y recibe la resistencia y función de comerHuir del animal a2. Si el valor que retorna no es 0, entonces el animal hijo recibe la resistencia y función de comerHuir del animal a1 y recibe la fuerza, velocidad y función de reproducción del animal a2.
- ComerSiempre: debe recibir dos animales (a1 y a2) se debe tomar la fuerza del animal a1 y compararla con la resistencia del animal a2, si la fuerza es mayor a la resistencia, el animal a2 se muere, sino el animal a1 se muere.
- HuirSiempre: debe recibir dos animales (a1 y a2) se debe tomar la velocidad del animal a1 y compararla con la velocidad del animal a2. Si la velocidad de a1 es mayor a la velocidad de a2, el animal a1 vive y se mueve a la izquierda de la casilla actual, si está ocupada se ubicará en la casilla de arriba, si está ocupada se situará en la casilla de la derecha, si está ocupada le queda la última opción que es la casilla de abajo, sino hay casilla disponible el animal a1 morirá. Si la velocidad del animal a2 es mayor o igual a la velocidad del animal a1, entonces se muere definitivamente.
- ComerAleatorio: debe recibir dos animales (a1 y a2) se debe tomar algún atributo del animal a1 al azar y compararlo con un atributo del animal a2 al azar, luego se comparan los atributos obtenidos y el que tenga mayor atributo se come al otro animal. En el caso de empate, el animal a2 se come a1 siempre.

4. Definición de funciones a realizar

A continuación, se definen los headers de las funciones a realizar:

```
void CrearAnimal(Animal* a);
void Borrar(Animal* a);
void MostrarAnimal(Animal* a);
void Reproducir(Animal* a1, Animal* a2, Animal* hijo);
void ComerOHuir(Animal* a1, Animal* a2);
int Comparar(Animal* a1, Animal* a2);
void BorrarMundo(Animal** Mundo);
void MostrarMundo(Animal** Mundo);
void ReproduccionSimple(Animal* a1, Animal* a2, Animal* hijo);
void ReproduccionCruzada(Animal* a1, Animal* a2, Animal* hijo);
void ComerSiempre(Animal* a1, Animal* a2);
void HuirSiempre(Animal* a1, Animal* a2);
void ComerAleatorio(Animal* a1, Animal* a2);
```

5. Programa General

Su programa general debe generar un menú que permita crear animales, tener la opción de avanzar una iteración en el tiempo y una opción para mostrar el mundo. Cada animal, será creado en un lugar **NO OCUPADO** de la matriz, donde se debe pedir por pantalla en cuales coordenadas será creado. También al crear animal se debe preguntar por cual función usar, para eso se ingresará un entero por función y tendrá el siguiente entero asignado:

- Funciones de reproducción
    1. ReproduccionSimple
    2. ReproduccionCruzada
- Funciones de comerHuir
    1. ComerSiempre
    2. HuirSiempre
    3. ComerAleatorio

Avanzar una iteración en el tiempo, provocará que todos los animales se muevan una casilla, donde puede ser arriba, abajo, izquierda y derecha, siendo estas cuatro opciones equiprobables (25 % cada una). Si llega la situación donde dos animales aparezcan en la misma casilla se debe hacer lo siguiente: primero deben reproducirse, el hijo que nazca de esa relación debe situarse a la izquierda de la casilla actual, si está ocupada se ubicará en la casilla de arriba, si está ocupada se situará en la casilla de la derecha, si está ocupada le queda la última opción que es la casilla de abajo, sino hay casilla disponible el hijo se morirá. Luego, de la reproducción los padres deben combatir entre ellos.

También debe existir una opción para terminar el programa y al seleccionar esta opción debe liberar toda la memoria pedida.

6. Archivos a Entregar

Resumiendo lo anterior, los archivos mínimos a entregar son:

- README.txt
- Animal.c
- Animal.h
- simulacion.c
- MAKEFILE

El archivo Animal.c debe contener las funciones implementadas relacionadas al manejo y control del Animal.

El archivo simulacion.c debe ejecutar el menu y llamar a las funciones necesarias.

7. Sobre Entrega

- El código debe venir **indentado y ordenado**.
- Las funciones deberán ir comentadas, explicando clara y brevemente lo que realiza, los parámetros que recibe y los que devuelve (en caso de que devuelva algo). Se deja libertad al formato del comentario.

- Debe estar presente el archivo MAKEFILE para que se efectúe la revisión, este debe compilar **TODOS** los archivos.
- Se debe trabajar de forma individual.
- **La entrega debe realizarse en tar.gz y debe llevar el nombre: Tarea2LP_RolIntegrante.tar.gz**
- **El archivo README.txt debe contener nombre y rol del alumno e instrucciones detalladas para la compilación y utilización de su programa.**
- La entrega será vía aula y el plazo máximo de entrega es hasta el **15 de octubre de 2021 a las 23:55 hora aula**.
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.
- Si el makefile no está bien realizado, la tarea no se revisará.
- Se utilizará Valgrind para detectar los leak de memoria.

8. Calificación

Todos comienzan con nota máxima y se les irá descontando por cada punto omitido (el puntaje que aparece al lado es el máximo de puntos que se les descontará en caso de incumplimiento):

- Transimperativer
  - CrearAnimal (7 pts)
  - BorrarAnimal (7 pts)
  - MostrarAnimal (3 pts)
  - Reproducir (4 pts)
  - Comparar (5 pts)
  - ComerOHuir (4 pts)
  - BorrarMundo (5 pts)
  - MostrarMundo (5 pts)
  - ReproduccionSimple (9 pts)
  - ReproduccionCruzada (9 pts)
  - ComerSiempre (9 pts)
  - HuirSiempre (9 pts)
  - ComerAleatorio (9 pts)
- Main (15 puntos)
- Descuentos
  - Código no ordenado (-10 puntos)
  - Código no compila (-100 puntos)
  - Warning (c/u -5 puntos)
  - Falta de comentarios (-30 puntos MAX)

- Por cada día de atraso se descontarán 20 puntos (La primera hora de atraso serán solo 10 puntos).
- Porcentaje de leak de memoria ((1-5) % -5 puntos (6- 50 %) -15 puntos (51-100 %) -25 puntos)

En caso de existir nota negativa esta será reemplazada por un 0.