INF-253 Lenguajes de Programación
Tarea 2: C

Profesor: Esteban Daines - Roberto Diaz
Ayudante Cátedras: Sebastian Godinez
Ayudante Tareas: Gabriel Valenzuela - Monserrat Figueroa

02 de Octubre de 2018

1. Objetivos

El alumno analizará e implementará operaciones básicas para el desarrollo de algoritmos genéticos. Esto lo realizará mediante el uso de **listas, punteros a funciones y punteros a void**, entre otros elementos del lenguaje.

2. Algoritmos Genéticos

Los algoritmos genéticos son un método de resolución de problemas aplicados principalmente en inteligencia artificial. Estos simulan la herencia del código genético de los seres vivos y su modificación por el proceso de selección natural, lo que conlleva a la evolución de las especies; en inteligencia artificial, ayudan a encontrar una solución óptima para un problema en base a los mecanismos antes mencionados. Para este caso, se intentará buscar la mejor solución de un problema acotado.

3. A implementar

En esta ocasión se ha decidido utilizar un algoritmo genético para mejorar la calidad de los datos presentes en diferentes listas. Para esto, se deben implementar las siguientes funciones:

- void* generarSolucion(int n): crea una lista enlazada de largo n compuesta por elementos al azar, y devuelve el puntero correspondiente. El detalle de estos elementos se detallan en la **Sección 4**.
- void* copiar(void* Lista): crea una copia de la lista ingresada y retorna el puntero correspondiente.
- void borrar(void* Lista): libera el espacio de memoria asignado a una lista.

- void imprimirSolucion(void* Lista): muestra por consola los elementos de la lista en el formato **(dato,tipo)**.
- void cruceMedio(void* Lista1, void* Lista2): función que intercambia la sección de los primeros $\frac{n}{2}$ elementos de la Lista 1 por los de la lista 2. En caso de ser de largo impar, $\frac{n}{2}$ se debe truncar. Ambas listas serán del mismo largo.
- void cruceIntercalado(void* Lista, void* Lista2): intercambia los elementos que se encuentren en posiciones pares entre las listas.
- void mutacionRand(void* Lista): selecciona un elemento aleatoriamente y lo reemplaza por otro valor al azar (que puede ser del mismo o distinto tipo).
- void mutacionTipo(void* Lista): selecciona un elemento aleatoriamente y lo reemplaza por otro valor al azar del mismo tipo. En el caso de los bit, éste debe cambiarse obligatoriamente.
- int evaluacionLista(int (*fun)(void*), void* Lista): aplica una función de evaluación sobre cada uno de los nodos, la cual evalúa la calidad de éste último. Se debe retornar la suma de todos los puntajes obtenidos.
- void genetico(void (*muta)(void*), void (*cruce)(void*), int n, int iteraciones): aplica el algoritmo genético haciendo uso todas las funciones mencionadas. Se deben crear dos listas de elementos, a las cuales se les deben realizar las siguientes operaciones:
  - Evaluar la calidad de cada lista padre.
  - Aplicar el cruce entre ambas listas padres para obtener dos listas hijas, las cuales deben ser evaluadas. Si la evaluación de ambos hijos es mayor a la de los padres, los hijos reemplazan a sus respectivos padres.
  - Aplicar mutación sobre los hijos obtenidos del cruzamiento, los cuales deben ser nuevamente evaluados. Si alguno de los hijos mutados mejoró su calidad con la mutación, debe reemplazar a su padre.

  Las operaciones descritas debe ejecutarse tantas veces como se indica en el número de *iteraciones*. Al finalizar los anterior, se deben mostrar ambas listas resultantes por pantalla y el puntaje de cada una.
- int main(): debe solicitar los datos necesarios para la ejecución del algoritmo, y al terminar debe dar la opción de repetir el proceso nuevamente.

4. Consideraciones

- La estructura de los nodos debe ser:

```
struct tNodo{
    void* dato;
    char tipo; //'i' 'c' 'b'
}
```

  Donde *i* (entero) pueden ser números del 0 al 9, *c* (caracter) pueden ser letras de la A a la F y *b* (bit) puede ser 0 o 1.

- Se hará entrega del archivo *genetico.h*, el cual contiene la definición de todas las funciones solicitadas. Este archivo no puede ser modificado.

- Las funciones deben ser desarrolladas e implementadas en el archivo *genetico.c*, junto con sus propias definiciones de las listas a utiliza.

- Cualquier tipo de función auxiliar no se debe agregar en el header entregado.

5. Sobre Entrega

- La revisión se efectuará sobre el sistema operativo CentOS presente en los computadores del LDS.

- **Se debe trabajar en grupos de DOS personas.**

- La entrega debe realizarse en tarball (tar.gz) y debe llevar el nombre: "Tarea2LP_RolIntegrante-1_RolIntegrante-2.tar.gz". **Tareas que no cumplan esta regla llevarán descuento**.

- El archivo README.txt debe contener nombre y rol de los integrantes del grupo e instrucciones **claras** para la compilación y utilización de su programa. Aquí pueden agregar también distintos supuestos que hayan utilizado para el desarrollo (conversado previamente con los ayudantes). **Tareas que no cumplan esta regla llevarán descuento**.

- Debe estar presente el archivo **makefile** para que se efectúe la revisión.

- Su código no debe presentar warnings y debe estar comentado de tal forma que sea comprensible. Éstos deben venir antes del código que comentan y, en caso de funciones, debe especificar los parámetros, los valores de retorno y una breve descripción de lo que hace. **Tareas que no cumplan esta regla llevarán descuento**.

- Todas las dudas deben ser hechas por la plataforma *Moodle*.

- La entrega será mediante *Moodle* y el plazo máximo de entrega es hasta el Martes 23 de Octubre a las 23:55.

- Por cada día o fracción de atraso se descontarán **20 puntos**. Si la fracción es menor a 20 minutos, el descuento será de sólo **5 puntos**.
- Las copias serán evaluadas con nota 0.

6. Calificación

La escala a utilizar para la revisión de la tarea es la siguiente:

- Código (60 Puntos)
  - Orden (5 Puntos)
  - Implementación (15 Puntos)
  - Punteros Void (10 Puntos)
  - Puntero Funciones (10 Puntos)
  - Listas (10 Puntos)
  - Estructuras (10 Puntos)
- Funciones (40 Puntos)
  - generarSolucion (4 Puntos)
  - copiar (3 Puntos)
  - borrar (3 Puntos)
  - cruceMedio (5 Puntos)
  - cruceIntercalado (5 Puntos)
  - mutacionRand (4 Puntos)
  - mutacionTipo (4 Puntos)
  - evaluacion (3 Puntos)
  - imprimirSolucion (2 Puntos)
  - genetico (5 Puntos)
  - main (2 Puntos)

Los descuentos que pueden ocurrir son:

- No compila (100 puntos)
- No usar punteros a funciones y void (100 puntos)
- Warnings (5 Puntos c/u)
- No respetar reglas de entrega (40 puntos c/u)