UTN – FRBA – Algoritmos y Estructura de Datos – Examen Final – 20/02/2015

Apellido y nombre:_____ Legajo: _____ Cursó con Prof: _____

Cantidad de hojas entregadas:_____ Nota:_____ Evaluó Prof: _____

- Si luego de la lectura del examen, durante la resolución tiene alguna duda, escriba hipótesis de trabajo, las cuales también serán evaluadas.
- Los puntos que solicitan codificación puede ser respondidos en C, ó C++, pero debe indicar el lenguaje utilizado.
- En C y C++ prototipo refiere a la declaración de la función, es decir tipo de dato retornado, nombre de la función, y tipos de los parámetros.

Sistema para el Seguimiento de Jugadores de Fútbol

Temas evaluados: Resolución de problemas, estructuras de datos, archivos, listas, y lenguaje de programación.

Contexto

Usted es parte de un equipo que desarrolla aplicaciones para la *FIFA*, y es el responsable de procesar las estadísticas referida a los jugadores clubes o selecciones a las que pertenecen.

Problema

Mostrar por la salida estándar un listado de jugadores cuyos idJugador esté comprendido en el rango [***desde, hasta***] y que hayan convertido más de **N** goles. Por cada jugador seleccionado se debe informar su **idJugador**, la **cantidad** de goles convertidos, y el **minuto promedio** en el que los convirtió. No se conoce la representación en archivos de los goles, pero la función ***LeerGol*** abstrae la representación y lectura de los datos. El listado resultante debe estar ordenado por idJugador en forma ascendente.

Restricciones

- Para representar los goles por jugador debe usar un arreglo de **estructuras con punteros a listas enlazadas**.
- La estructura sí debe tener la **cantidad de goles**, pero **no el id del jugador**; de ser necesario, puede tener otros campos.
- Invoque pero no desarrolle la función ***InicializarArreglo***.
- Invoque pero no desarrolle la función ***bool LeerGol(unFlujo, unGol).*** Si pudo leer de *unFlujo*, devuelve los datos leídos en el parámetro de salida *unGol*, y retorna *true*. Si no puede leer retorna *false*. **Gol** se declara como: **struct Gol{unsigned jugadorId, partidoId, minuto;};**
- Para manejar estructuras enlazadas, **debe usar funciones de biblioteca**.
- Los id de jugadores son valores pares y comienzan según la constante **IDMIN**.
- La cantidad de jugadores la define la constante **CANTJUGADORES**

Se pide

1. Justifique. Dado el problema planteado, ¿cuál es la estructura de datos más eficiente del punto de vista de la velocidad para cargar los minutos de cada gol de cada jugador? Informe en caso de no tener que cumplir la restricción de incluir la cantidad de goles en la estructura *qué ventajas y desventajas observa*
2. Codifique la declaración de las estructuras.

Diagrame o codifique las siguientes funciones, considerando todas las restricciones expuestas.

3. ***GetÍndice(unId)*** que dado un id de un jugador retorna el índice que le corresponde en el arreglo.
4. ***GetId(unÍndice)*** que realiza la operación inversa. y=2*x+b
5. ***CargarDatosEnMemoria(unArreglo, unFlujo)***
6. ***ListarJugadores(unArreglo, idDesde, idHasta, cantidadMinimaDeGoles)***.

UTN – FRBA – Algoritmos y Estructura de Datos – Examen Final – 20/02/2015

Apellido y nombre:_____ Legajo: _____ Cursó con Prof: _____

Cantidad de hojas entregadas:_____ Nota:_____ Evaluó Prof: _____

- Si luego de la lectura del examen, durante la resolución tiene alguna duda, escriba hipótesis de trabajo, las cuales también serán evaluadas.
- Los puntos que solicitan codificación puede ser respondidos en C, ó C++, pero debe indicar el lenguaje utilizado.
- En C y C++ prototipo refiere a la declaración de la función, es decir tipo de dato retornado, nombre de la función, y tipos de los parámetros.

Punto 1.

Justifique. Dado el problema planteado, ¿cuál es la estructura de datos más eficiente del punto de vista de la velocidad para cargar los minutos de cada gol de cada jugador? Informe en caso de no tener que cumplir la restricción de incluir la cantidad de goles en la estructura qué ventajas y desventajas observa.

R. Un array de pilas. Array ya que por el índice es posible encontrar una Posicion Unica y Predecible con la que poder hacer acceso directo. Los goles se pueden poner en una pila ya que al no requerir orden no es necesario recorrer para insertar en una lista ordenada, menos aun recorrer hasta el final si se optara por una cola

Ventaja: Menos memoria, sin necesidad de actualizar la cantidad de goles.

Desventaja: Al no tener la cantidad de goles, se requiere recorrer cada lista de goles para determinar si debe ser incluido o no en el listado.

Punto 2.

```
struct Nodo{unsigned minuto; Nodo *sgte;};
struct Jugador{unsigned cantidad; Nodo *losGoles;};
Jugador losJugadores[CANTJUGADORES];
```

Punto 3.

Diagrame o codifique GetÍndice(unId) que dado un id de un jugador retorna el índice que le corresponde en el arreglo.

```
unsigned GetÍndice(unsigned unId){
  return (unId-IDMIN)/2;
}
```

Punto 4.

*Diagrame o codifique GetId(unÍndice) que realiza la operación inversa. y=2*x+b*

```
unsigned GetId(unsigned unÍndice){
  return 2*unÍndice + IDMIN;
}
```

Punto 5.

Diagrame o codifique CargarDatosEnMemoria(unArreglo, unFlujo)

```
// Recordar bool LeerGol(FILE *, Gol &). Gol: struct Gol{unsigned jugadorId, partidoId, minuto;};
// El hecho de disponer de la función nos permite abstraernos del flujo
// La estructura más adecuada, por lo ya señalado es un array de pilas

void CargarDatosEnMemoria(Jugador a[], FILE *unFlujo){
  InicializarArreglo(Jugador a[]); // se pide invocar la función sin desarrollar
  Gol gol;
  while( LeerGol(unFlujo, gol) ){
    i=GetÍndice(gol.jugadorId);
    ++a[i].cantidadDeGoles;
    Push(a[i].minutos, gol.minuto);
  }
}
```

Punto 6.

Diagrame o codifique ListarJugadores(unArreglo, idDesde, idHasta, cantidadMinimaDeGoles).

```
void ListarJugadores(Jugador a[], unsigned idDesde, unsigned idHasta, unsigned minimosGoles){
  unsigned i = GetÍndice(idDesde);
  for(j=idDesde; j<idHasta; j+=2, i++)
    if(a[i].goles>minimosGoles){
      cout << GetId(i) << " "
        << a[i].cantidad << " "
        << GetPromedio(a[i].minutos) << endl;
    }
}
```