USO DE UN LENGUAJE DE PROGRAMACIÓN DE ALTO NIVEL PARA LA IMPLEMENTACIÓN DE SOLUCIONES COMPUTACIONALES EMPLEANDO ARREGLOS

Maria C. Torres Madroñero
Profesora asociada
Departamento de Ciencias de la Computación y la Información

Objetivos

- Diseñar soluciones computacionales empleando una estructura de datos lineal de memoria fija, el arreglo, empleando las operaciones básicas de agregar, eliminar, buscar y ordena.

Recursos requeridos

- PC
- IDE para JAVA o Python – el estudiante deberá seleccionar un lenguaje de programación para el desarrollo de las prácticas de laboratorio

Actividades preliminares al laboratorio

- Lectura de la guía

Marco teórico

ARREGLOS DE OBJETOS

Los **arreglos** son una colección de variables todas del mismo tipo. Cada variable o celda de un arreglo tiene un **índice**, el cual permite referenciar el valor determinado en la cerda. Cada valor almacenado en un arreglo se llama **elemento** del arreglo. Un arreglo puede ser usado para almacenar una colección de objetos.

EJEMPLO DE COLECCIÓN DE OBJETOS: PUNTAJES DE JUGADORES

Supongamos que deseamos almacenar el nombre del jugador y su puntaje en un video juego. Para esto podemos definir la clase GameEntry como se indica en el siguiente diagrama de clase:

GameEntry
-nombre: String -score: int
+GameEntry(String n, int s) +getNombre( ): String +getScore( ): int +toString(): String

Ahora supongamos que deseamos mantener una colección con los puntajes más altos, vamos a denominar esta clase como ScoreBoard. Un tablero de puntajes está limitado a un determinado número de puntajes, una vez el límite es alcanzado, un nuevo puntaje ingresa a la colección solo si es mayor al puntaje más bajo en el tablero.

- Usaremos un arreglo para administrar las instancias de GameEntry. El arreglo es declarado con una capacidad máxima, y todas las entradas inicialmente están vacías.

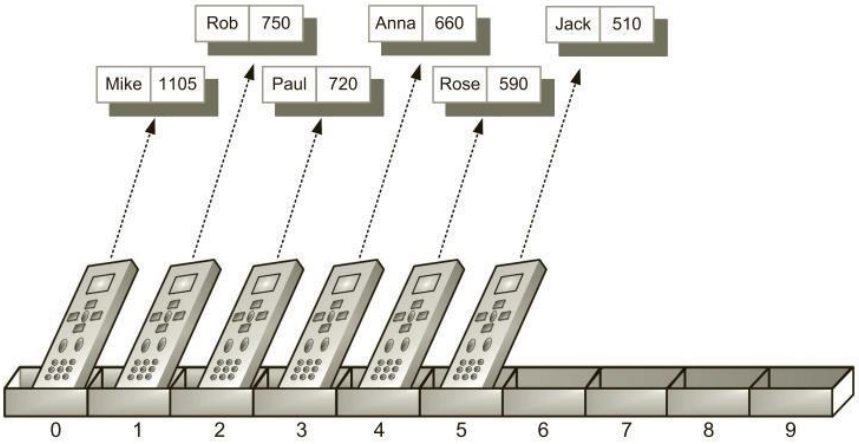

Figura 1. Arreglo de GameEntry

- Para agregar un nuevo puntaje: si el tablero no está lleno, un nuevo puntaje puede ser agregado
- Si está lleno, el puntaje se agrega solo si es mayor al menor puntaje almacenado
- Los puntajes se almacenan en orden descendente

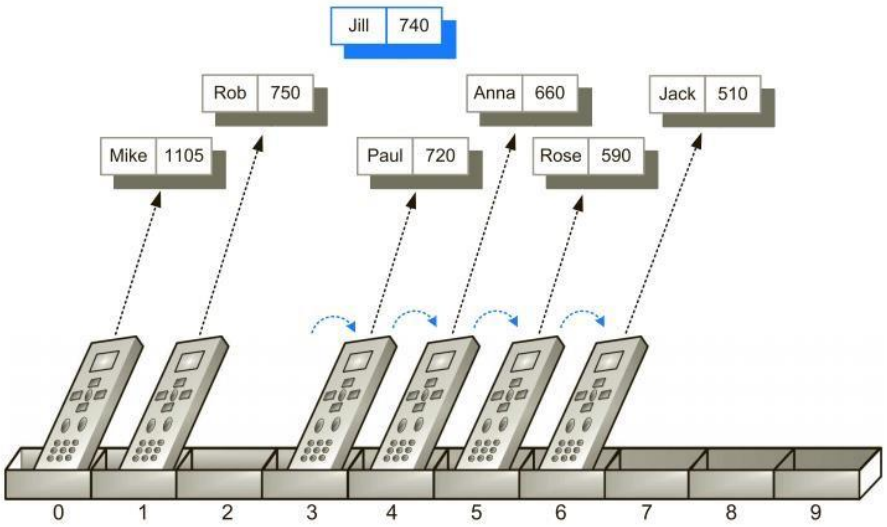

Figura 2. Movimiento de elementos para insertar un nuevo puntaje

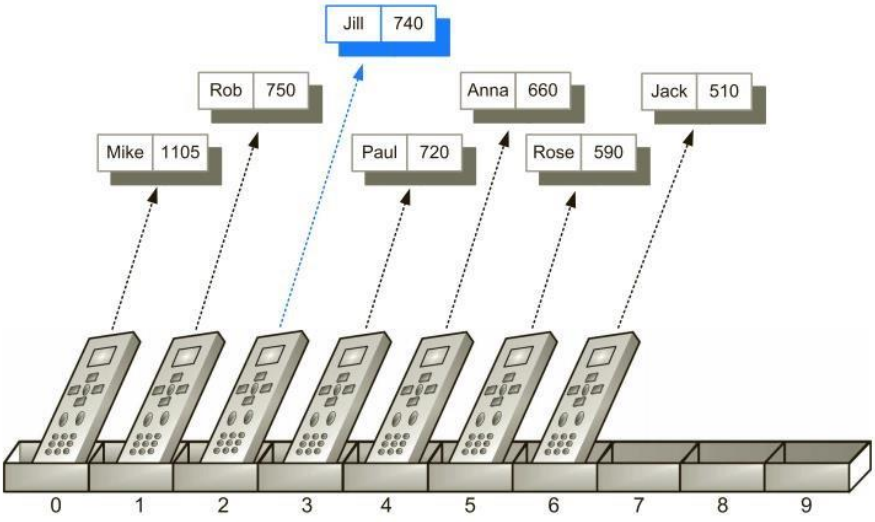

Figura 3. Arreglo de GameEntry despues de insertar nuevo puntaje

- Para eliminar un puntaje, todos los puntajes más bajos se deben desplazar para llenar el espacio vacío

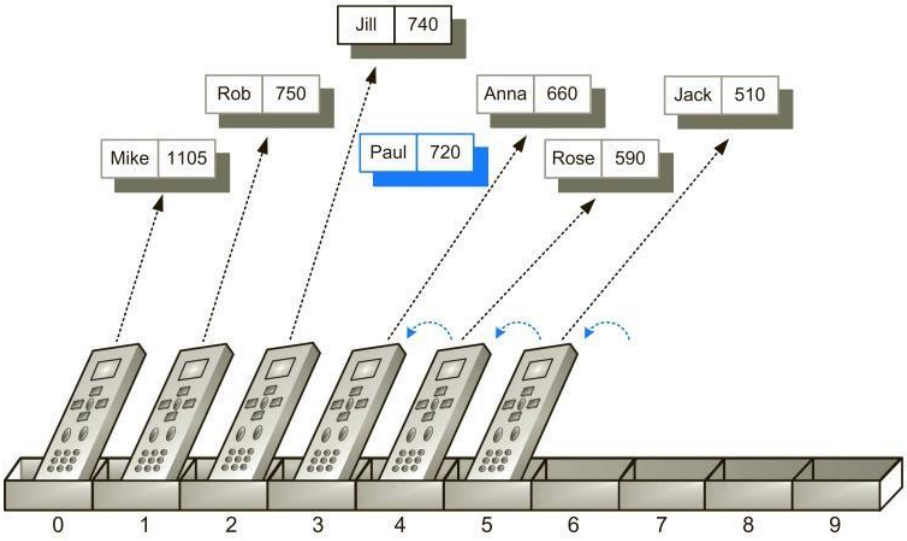

Figura 4. Proceso para eliminar un puntaje

A continuación se presenta la clase ScoreBoard:

ScoreBoard
-numEntries: int -board: GameEntry[ ]

+ScoreBoard(int capacity) +add(GameEntry e) +remove(int i): GameEntry +toFile()

La clase ScoreBoard tiene dos atributos: numEntries mantiene cuantas posiciones del arreglo contienen elementos GameEntry, y board es el arreglo, donde cada posición almacena un GameEntry. El constructor recibe como parametro la capacidad maxima del arreglo. Note que el numEntries es diferente a la capacidad o longitud del arreglo: inicialmente, el numEntries será igual a 0 y la longidud del arreglo será igual a la capacidad; solo si el arreglo está lleno, la capacidad será igual a numEntries. El metodo add() ingresa un nuevo puntaje GameEntry al arreglo, remove() elimina el elemento de la posición i y retorna el objeto, y toFile() guarda la información del tablero en un archivo de texto.

A continuación, se describen cada uno de los seudocodigos:

```
//Seudocodigo para el constructor
scoreBoard(int capacity)
1.  board = new GameEntry[capacity]
2. numEntries = 0
```

```
//Seudocodigo para agregar un nuevo puntaje
add(GameEntry e)
1.  int newScore = e.getScore()
2.  //Se determinar si el nuevo puntaje debe ingresar
3.  if (numEntries<board.length() | | newScore>board[numEntries-1].getScore())
4.      numEntries++
5.  //Realizar desplazamientos
6.  int j = numEntries-1
7.  while(j>0 && board[j-1].getScore()<newScore)
8.      board[j] = board[j-1]
9.      j--
10.     board[j] = e
```

```
//Seudocodigo para eliminar un puntaje
Remove(int I)
1.  if (i<0 | | I>=numEntries)
2.     return Null
3.  else
4.      GameEntry temp = board[i]  //Realizar desplazamiento
5.      for(int j = I, j<numEntries-1, j++)
6.          board[j] = board[j+1]
7.       board[numEntries-1] = Null
8.       numEntries--
9.       return temp
```

Este es un ejemplo de como podemos usar arreglos (ya sean estáticos o dinámicos) para administrar colecciones de objetos. Dependiendo del problema, las operaciones que vamos a realizar sobre la colección pueden variar, sin embargo, las operaciones más comunes son agregar un nuevo elemento, eliminar, buscar, ordenar e incluso copiar la información a un archivo. Los arreglos son una de las estructuras de datos más comunes para administrar colecciones de variables, de ahí la importancia de conocer su funcionamiento.

Actividades

Problema 1 Construcción de una colección de usuarios – agenda

En esta actividad vamos a utilizar la clase Usuario elaborada en el laboratorio 2 (ver diagrama de clase)

Usuario	
- nombre: String - id: long - fecha_nacimiento: Fecha - ciudad_nacimiento: String	- tel: long - email: String - dir: Direccion
+ Usuario() + Usuario(String n, long id) + setNombre(String n) + setId(long id) + setFecha_nacimiento(Fecha f) + setCiudad_nacimiento(String c) + setTel(long t) + setEmail(String e) + setDir(Direccion d)	+ getNombre():String + getId():long + getFecha_nacimiento():Fecha + getCiudad_nacimiento():String + getTel():long + getEmail():String + getDir():Direccion + toString(): String

Vamos a crear una Agenda que permita administrar un registro de usuarios almacenados de acuerdo con el orden de de registro. Para esto vamos a construir la clase Agenda de acuerdo con el siguiente diagrama de clases:

Agenda
- registro: Usuario[ ] - no_reg: int
+ Agenda(int capacity) + agregar(Usuario u): Boolean + buscar(int id): int + eliminar(int id): Boolean + toFile() + import()

A continuación, la descripción de cada uno de los atributos y métodos:

- Atributo registro: es un arreglo de objetos, donde cada posición almacena un Usuario
- Atributo no_reg: es un entero que mantiene el número de posiciones llenas o Usuarios almacenados en el arreglo registro
- Constructor Agenda(int capacity): inicializa el arreglo registro con una longitud dada por el parámetro de entrada capacity; adicionalmente inicializa el atributo no_reg en cero.
- Método agregar(Usuario u): recibe un objeto de tipo usuario. Este método, primero verifica si el usuario u ya existe en la agenda, invocando el método buscar(). Si el usuario ya se encuentra en la agenda retorna Falso indicando que no fue posible agregar nuevamente el usuario. Si el usuario no existe, lo agrega en la siguiente posición disponible sin exceder la capacidad del arreglo. Una vez ingresado el u en el registro se retorna TRUE indicando que si fue posible agregar el nuevo usuario.

- Método buscar(int id): este método busca un usuario en el registro que tenga el número de identificación id ingresado como parámetro. Si el método no encuentra el usuario regresa -1, en caso contrario retorna la posición del arreglo registro donde se encuentra el usuario. El método buscar debe implementarse completamente, sin hacer uso de los métodos existentes en JAVA o Python.
- Método eliminar(int id): este método busca el usuario con número de identificación id ingresado como parámetro empleando el método buscar(). Si no lo encuentra retorna FALSE, en caso contrario, debe eliminar el usuario, realizar los desplazamientos respectivos para llenar la posición del arreglo, y retornar TRUE al finalizar el proceso.
- Método toFile(): copia todos los datos de los usuarios almacenados en el registro en un archivo de texto agenda.txt. Cada línea del archivo de texto es un usuario.
- Método import(): lee un archivo agenda.txt donde cada línea de texto corresponde a un usuario; este método agrega cada usuario del archivo al arreglo registro.

Para la prueba de la clase Agenda realice las siguientes actividades:
1. Presente un programa que:
a. Inicialice 5 usuarios en una agenda con toda su información. La información de los usuarios debe ser configurada desde la clase principal (no por consola)
b. Busque un usuario por su número de id e imprima en pantalla la posición del arreglo donde se encuentra almacenado
c. Emplee el método toFile() para almacenar todos los usuarios de la agenda en un archivo Agenda.txt

2. Presente un programa (en un main diferente al anterior) que:
a. Emplee el método import() para inicializar los usuarios de una agenda con los datos almacenados en Agenda.txt, todos los usuarios deben estar completamente configurados de acuerdo con estos datos.
b. Imprimir en pantalla los 5 usuarios leído empleando el método toString() de la clase Usuario
c. Eliminar un usuario dado su número de id
d. Emplee el método toFile() para almacenar todos los usuarios actualizados de la agenda en un archivo Agenda2.txt

Instrucciones de entrega

- La solución de los problemas debe desarrollarse en JAVA o Python. Los estudiantes tendrán la libertad de seleccionar el lenguaje de programación y plataforma para presentar la solución de los problemas.
- La solución debe emplear librerías nativas y se invita a los estudiantes a no usar código descargado de internet. Los laboratorios están diseñados para practicar los fundamentos teóricos; entre más código escriba el estudiante más fácil será su comprensión de los temas de clase.
- La solución se puede presentar en grupos de hasta 3 estudiantes.
- La solución de los problemas debe entregarse y sustentarse en el aula de clase o en la hora de asesoría a estudiantes. No se reciben soluciones por correo electrónico.