## Luis Fernando Muñoz A00046396

## Santiago del Campo A00137608

## Sebastian Arango A00130532

## Fase 1: Identificación del problema

**Síntomas y necesidades**

**-**Las grandes ciudades presentan un aumento del parque automotor.

-No es posible que todos los autos de la ciudad circulen por sus calles.

-Se presenta una falta de espacios para el estacionamiento de los automóviles.

-Una institución educativa adquirió nuevos terrenos con miras a que se utilicen como estacionamiento para los estudiantes y sus colaboradores.

-La institución educativa requiere de un programa que le permita a la directivas conocer un poco más sobre el funcionamiento del nuevo parqueadero comprado.

**Definición del problema**

Una institución educativa se enfrenta al problema de falta de parqueaderos y para darle solución ha decidido comprar un terreno aledaño para que los autos de sus colaboradores y estudiantes parqueen en él, por consiguiente, se necesita de un programa que le permita a las directivas de este ente universitario conocer un poco acerca del funcionamiento del parqueadero.

**Listado de requerimientos**

**Requerimientos funcionales**

|  |  |
| --- | --- |
| Nombre | R1 – Calcular cantidad de carros que se tuvieron que mover al final de la jornada |
| Resumen | Calcula la cantidad de carros que se tuvieron que mover al final de la jornada |
| Entradas | |
| Número de bahías que tendrá habilitadas el parqueadero. | |
| Capacidad máxima permitida por bahía. | |
| Número de autos que entrarán al parqueadero durante la jornada. | |
| Serie de códigos vehiculares en el orden en que entraron durante la mañana y los códigos en el orden en que fueron retirados los vehículos. | |
| Resultados | |
| Carros que tuvieron que ser movidos en total en cada bahía para que estos pudieran salir. | |

**Requerimientos no funcionales**

-La determinación de la bahía en la que está un carro, se debe de hacer de la forma más eficiente posible.

## Fase 2: Recopilación de información

**Definiciones**

**Pila:** Es un tipo de datos abstracto que sirve como una colección de elementos con dos operaciones principales: apilar, retirar.

**Cola:** Es una estructura de datos, caracterizada por ser una secuencia de elementos en la que la operación de inserción push se realiza por un extremo y la operación de extracción pop por el otro.

**Tabla Hash:** Es un tipo de datos abstracto que sirve como una colección de elementos en la cual se puede ejecutar las operaciones de inserción, consulta y de eliminación en tiempo constante.

## Fase 3: Búsqueda de soluciones creativas

(1)

Se procede a utilizar la técnica de relación forzada entre una pila y una cola, con el objetivo de modelar el funcionamiento de una bahía en la solución del problema. La parte ovalada de la bahía será la cola, y la parte recta donde se estacionan los carros será la pila. Se puede utilizar una tabla Hash para la rápida búsqueda de la placa de un carro en alguna de las bahías, esto hace que se tenga un control general del parqueadero.

Cada una de estas estructuras serán implementadas con componentes de listas enlazadas.

(2)

Por lluvia de ideas, se propone modelar la bahía como dos listas secuenciales, una donde se guardarán los carros estacionados, y en la otra los carros que tendrán que ir por la vía ovalada para ser reacomodados en la primera.

El parqueadero será una lista secuencial de bahías.

(3)

Se propone una estructura similar a la de la idea (1) pero con cambios internos en las estructuras de datos a utilizar, donde la pila, cola y el hashmap se implementaran con arreglos de tamaño estático.

## Fase 4: Transición de la formulación de ideas a los diseños preliminares

(1)

-Permite un modelo intuitivo de la solución con estructuras como la pila y la cola.

-Permite un acceso rápido a las bahías de los carros mediante la tabla hash

(2)

-Permite facilidad de implementación por contener estructuras de datos familiares a los programadores

(3)

-Mal manejadas, se puede incurrir en errores de sobrellenado en las estructuras utilizadas en esta solución.

## Fase 5: Evaluación y selección de la mejor solución

**Criterios de evaluación**

­ **Criterio A.** La solución ofrece estructuras de datos que permiten acceder a la bahía en la que un determinado carro se encuentra en tiempo:

[2] O(1)  
­ [1] O(n)  
­**Criterio B.** El modelo de las bahías en la solución es:

­ [2] Intuitivo y de fácil entendimiento para los programadores  
­ [1] Rebuscado y tedioso de codificar para los programadores

**Criterio C.** El nivel de aprendizaje por implementar la solución es:

[3]Alto

[2]Intermedio

[1]Bajo

**Criterio D.** La solución ofrece estructuras de datos de tamaño

[2] Dinámico

[1] Estático

**Criterio E.** Las operaciones de búsqueda son de tiempo.

[2]O(1)

[1]O(n)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Soluciones | Criterio A | Criterio B | Criterio C | Criterio D | Criterio E | Sumatoria |
| (1) | 2 | 2 | 3 | 2 | 2 | 11 |
| (2) | 1 | 1 | 1 | 2 | 1 | 5 |
| (3) | 2 | 2 | 3 | 1 | 2 | 10 |

## 

## 

## Fase 6: Preparación de informes y especificaciones

**Definición de los TAD**

|  |
| --- |
| **TAD TablaHash** |
| **Representación: (H(K),T,L,V)**  **Donde H(k) es una función del dominio de todas las llaves (L) hasta un subconjunto de los naturales [0,1,2...n-1].**  **T es un arreglo de tamaño n que guarda Valores (V).** |
| **Invariantes:**  **0<n**  **n E N** |
| **Operaciones:**  **CrearTablaHash Entero X TipoValor X TipoLlave ----> TablaHash**  **EstaVacio TablaHash ----> booleano**  **Insertar L X V X TablaHash ----> booleano**  **Eliminar L X TablaHash ----> V**  **Buscar L X TablaHash ----> V** |

|  |
| --- |
| **CrearTablaHash(n,TV,TL)** |
| **Pre: TRUE** |
| **Post: (H(k),T,L,V), |T|=n, V=TV, L=TL** |

|  |
| --- |
| **EstaVacio(hs)** |
| **Pre:hs=(H(k),T,L,V)** |
| **Post:**  **TRUE si NumeroDeElementos(T)=0**  **FALSE si NumeroDeElementos(T)!=0** |

|  |
| --- |
| **Insertar(Ll,Val,hs)** |
| **Pre: hs.H(Ll) está definido** |
| **Post:**  **True si se insertó**  **False en caso contrario** |

|  |
| --- |
| **Eliminar(Ll,hs)** |
| **Pre: hs.T[hs.H(Ll)] Debe de tener un valor** |
| **Post:**  **hs.T[hs.H(Ll)] Ya no tiene valor** |

|  |
| --- |
| **Buscar(Ll,hs)** |
| **Pre: hs.H(Ll) está definido** |
| **Post: hs.T[hs.H(Ll)]** |

|  |
| --- |
| **TAD Cola** |
| **Representación:**  **cola: <a0,a1...an>, a\_i E V**  **V es el tipo de elemento** |
| **Invariantes:**  **n E N** |
| **Operaciones:**  **CrearCola TipoValor ----> Cola**  **EstaVacio Cola ----> booleano**  **Insertar Cola X V ----> booleano**  **Retirar Cola ----> V** |

|  |
| --- |
| **CrearCola(tv)** |
| **Pre: TRUE** |
| **Post: cola | cola.V=tv** |

|  |
| --- |
| **EstaVacio(c)** |
| **Pre:TRUE** |
| **Post:**  **True si c no tiene elementos**  **False de lo contrario** |

|  |
| --- |
| **Insertar(c,val)** |
| **Pre: val E V** |
| **Post: val esta en c, c.a0=val** |

|  |
| --- |
| **Retirar(c)** |
| **Pre: EstaVacio(c)=FALSE** |
| **Post: a0 ya no pertenece a c** |

|  |
| --- |
| **TAD Parqueadero** |
| **Representación:**  **lista: <a1,a2...an>, a\_i es una Bahia** |
| **Invariantes:**  **n>0** |
| **Operaciones:**  **CrearParqueadero \_\_\_\_\_\_\_\_ ----> Parqueadero**  **Insertar Bahia X V ----> void**  **obtenerVahiaDeVehiculo String ---->Bahia** |

|  |
| --- |
| **CrearParqueadero()** |
| **Pre: TRUE** |
| **Post: lista de bahias** |

|  |
| --- |
| **Insertar(B,V)** |
| **Pre: val E V** |
| **Post: vehiculo V se aloja en Bahia B** |

|  |
| --- |
| **obtenerVahiaDeVehiculo(vehiculo a)** |
| **Pre: val E V** |
| **Post: Bahia donde se encuentra el vehículo a** |

|  |
| --- |
| **TAD Bahía** |
| **Representación: <Cola,Pila>** |
| **Invariantes:**  **Se llena la pila primero.**  **Solo se hace uso de la cola como ayuda para extraer un carro de la Bahía.** |
| **Operaciones:**  **CrearBahia \_\_\_\_\_\_ ---->Parqueadero**  **ingresaVehiculo String ---->Boolean**  **saleVehiculo String ---->Boolean** |

|  |
| --- |
| **CrearBahia()** |
| **Pre: True** |
| **Post: una nueva bahia** |

|  |
| --- |
| **ingresaVehiculo(String a)** |
| **Pre: a, es la placa del vehículo nuevo** |
| **Post:ingresa un nuevo vehículo a la bahia** |

|  |
| --- |
| **saleVehiculo (String a)** |
| **Pre: a, es la placa del vehiculo que sale** |
| **Post:un vehículo abandona la bahia** |

|  |
| --- |
| **TAD Pila** |
| **Representación:**  **[A1 A2 A3 A4 … An)**  **[ = intervalo de estabilidad (cerrado)**  **An) = Objeto vulnerable a ser quitado de la pila, así como ser tapado por uno nuevo** |
| **Invariantes:**  **n E N** |
| **Operaciones:**  **crearPila: ->Pila**  **poner: Pila x Object ->Pila**  **estaVacia: Pila ->Booleano**  **tope: Pila ->Object**  **quitarDePila: Pila ->Pila** |

|  |
| --- |
| **crearPila()** |
| **“Crea una pila vacía”** |
| **pre: True** |
| **post: pila = {Tope: <>}** |

|  |
| --- |
| **poner(pila, object)** |
| **“añade un nuevo ítem al tope de la lista”** |
| **pre: pila = [A1 A2 A3 A4 … An)** |
| **post: : pila = A1 A2 … An object** |

|  |
| --- |
| **estaVacia(pila)** |
| **“Determina si la pila está vacía”** |
| **pre: pila = [A1 A2 A3 A4 … An)** |
| **post: False si pila.tope lanza excepción**  **True en caso contrario** |

|  |
| --- |
| **quitarDePila(pila)** |
| **“Si la pila no está vacía, se quita el objeto en el tope de la pila”** |
| **pre: pila = [A1 A2 A3 A4 … An)** |
| **post: pila = A1 A2 … An-1** |

|  |
| --- |
| **tope(pila)** |
| **“Retorna el objeto que está en el tope de la pila, no cambia en nada la pila”** |
| **pre: pila = [A1 A2 A3 A4 … An)** |
| **post: An**  **null si pila.tope lanza excepción** |

**Diseño de pruebas**

**Cola Enlazada**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clase | Método | Escenario | Entrada | Resultados |
| ColaEnlazada | queue() | **Escenario1** | 2 | Agregado Exitosamente |
| ColaEnlazada | queue() | **Escenario1** | 2000000000 | Agregado Exitosamente |
| ColaEnlazada | queue() | **Escenario1** | null | Agregado Exitosamente |
| ColaEnlazada | deQueue() | **Escenario2\_1** |  | 2 |
| ColaEnlazada | deQueue() | **Escenario2\_2** |  | 1 |
| ColaEnlazada | deQueue() | **Escenario2\_3** |  | 10 |
| ColaEnlazada | deQueue() | **Escenario2\_4** |  | 3 |
| ColaEnlazada | deQueue() | **Escenario2** |  | null |

**Pila Enlazada**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clase | Método | Escenario | Entrada | Resultados |
| PilaEnlazada | push() | **Escenario1** | 60 | Agregado Exitosamente |
| PilaEnlazada | push() | **Escenario1** | 456466464 | Agregado Exitosamente |
| PilaEnlazada | push() | **Escenario1** | 000 | Agregado Exitosamente |
| PilaEnlazada | pop() | **Escenario2\_1** | void | 10 |
| PilaEnlazada | pop() | **Escenario2\_2** | void | 0 |
| PilaEnlazada | pop() | **Escenario2\_3** | void | 1 |
| PilaEnlazada | pop() | **Escenario2\_4** | void | 3 |
| PilaEnlazada | pop() | **Escenario1** | void | null |

**Tabla Hash Encadenada**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clase | Método | Escenario | Entrada | Resultados |
| TablaHashEncadenada | insert() | **Escenario1** | **llave=**”sdklfsd”  **valor=**2 | Agregado Exitosamente |
| TablaHashEncadenada | insert() | **Escenario1** | **llave=**”sklfsmlmlkmlmlmlmlmmmmklmlkmlmlmlmlmlmlkmlmlkmlmklkmlkmklmlkmlkmlkmlkmlkmlkmklmlkmlmdsklfsmlmlkmlmlmlmlmmmmklmlkmlmlmlmlmlmlkmlmlkmlmklkmlkmklmlkmlkmlkmlkmlkmlkmklmlkmlmd”  **valor=**8 | Agregado Exitosamente |
| TablaHashEncadenada | insert() | **Escenario1** | **llave=**null  **valor=**46 | nullPointerException, no agregado |
| TablaHashEncadenada | find() | **Escenario2** | “Automovil” | 2 |
| TablaHashEncadenada | find() | **Escenario2** | “l” | 15 |
| TablaHashEncadenada | find() | **Escenario2** | “li” | 10654 |
| TablaHashEncadenada | find() | **Escenario2** | “movil” | 46 |
| TablaHashEncadenada | find() | **Escenario2** | “ka” | null |
| TablaHashEncadenada | delete() | **Escenario2** | “Automovil” | Eliminado exitosamente |
| TablaHashEncadenada | delete() | **Escenario2** | "sklfsmlmlkmlmlmlmlmmmmklmlkmlmlmlmlmlmlkmlmlkmlmklkmlkmklmlkmlkmlkmlkmlkmlkmklmlkmlmd" | Eliminado exitosamente |
| TablaHashEncadenada | delete() | **Escenario2** | null | nullPointerException, no eliminado |