Simulación de una empresa de

Transporte de automóviles en C++

**Poner qué no tenemos (b2)**



Álvaro Camacho Martín 48083824R Óscar Montesano Martínez 53852484P

TAD´s implementados en el proyecto

# - Pila:

Especificación PILA[VEHICULO] Operaciones:

**{Generadoras}**

Pila():pila ; genera una pila vacía de longitud 0 Apilar(Vehiculo): ; Añade un vehículo a la cima de la pila

**{Modificadoras}**

Parcial Desapilar(): ; Saca el ultimo vehículo añadido a la pila

**{Observadoras}**

Parcial Mostrar(): Vehiculo; Muestra el último vehículo añadido a la pila Es\_vacia(): Bool; Devuelve True si la pila está vacía

verPila(): ; Muestra la pila, vehículo a vehículo empezando por la cima get\_profundidad(): entero; Devuelve la profundidad de la pila

# - Cola:

Especificación COLA[VEHICULO] Operaciones:

**{Generadoras}**

Cola():cola ; genera una cola vacía (con ambos nodos a NULL y con longitud 0) Encolar(Vehiculo): ; Añade un vehículo al final de la cola

**{Modificadoras}**

Parcial Desencolar(): Vehículo; Saca el ultimo vehículo de la cola (y lo devuelve)

**{Observadoras}**

Parcial Inicio(): Vehículo; Muestra el primer vehículo de la cola Get\_longitud(): Entero; Devuelve la longitud de la cola, 0 si es vacía Es\_vacia(): Bool; Devuelve True si la cola es vacía

Mostrar\_cola(): Vehiculos; Va mostrando el primer vehículo de la cola uno detrás de otro.

* Listas:

Especificación LISTA[PEDIDO]

Operaciones:

**{Generadoras}**

Lista():lista ; genera una lista vacía(con ambos nodos a NULL y con longitud 0)

insertar\_derecha(Pedido): ; añade un pedido al final de la lista

insertar\_izquierda(Pedido): ; añade un pedido al inicio de la lista

insertar\_prioridad(Pedido): ; añade un pedido según su prioridad

insertar\_enPosicion(int, Pedido): ; añade un pedido en la posición dada

**{Modificadoras}**

Parcial borrar\_izquierda(): ; elimina el primer pedido de la lista

Parcial borrar\_position(int): ; elimina el pedido de la posición dada

Parcial vaciar\_lista(): ; vacía la lista

**{Observadoras}**

Parcial ver\_primero(): Pedido ; Muestra el primer pedido de la lista

Parcial ver\_ultimo(): Pedido ; Muestra el último pedido de la lista

Parcial ver\_posicion(int): Pedido ; Muestra el pedido de la posición dada

get\_longitud(): int ; devuelve la longitud de la lista. 0 si es vacía

es\_vacia(): bool; devuelve True si la lista es vacía

mostrar(): ; muestra los pedidos de la lista en orden.

* Arboles(ABB):

Especificación ABB[NumConc, zona, listaVehiculos]

Operaciones:

**{Generadoras}**

ABB(): ; genera árbol vacío, sin raíz ni hijos

ABB(NodoABB \*r): ; genera un árbol con raíz r

ABB(NumConc, zona, listaVehiculos, NodoABB \*hIz=NULL, NodoABB \*hDer=NULL): ; genera un árbol con raíz NumConc, zona y listaVehiculos sin hijos.

Insertar(NumConc, zona, listaVehiculos): ; añade un NumConc, zona y lista de vehículos al árbol ordenados por NumConc

**{Modificadoras}**

**{Observadoras}**

Parcial verInOrden(): ; muestra el árbol en inorden

* 1. Para implementar ambos Tipos Abstractos de Datos hemos usado memoria dinámica. Cada TAD consta de dos clases, la suya y el nodo. Un nodo es una celda que contiene un elemento de la estructura de datos además de un puntero que apunta al siguiente nodo. En el caso de Colas, hay además un puntero que apunta al inicio, por donde salen los elementos y otro que apunta al final, por donde entran. La pila solo tiene uno apuntando al nodo que en ese momento sea la cima. Las listas tienen un puntero que apunta al primer elemento y otro al último. Los árboles apuntan a sus hijos izquierdo y derecho.

Las dificultades que nos hemos encontrado han principalmente sido trabajar con punteros, ya que nunca lo habíamos hecho, y aunque en java habíamos trabajado con clases en POO, al haber dos ficheros por clase y tantos métodos nos ha costado bastante más corregir los errores.

Hemos conseguido implementar todo lo que pedía el trabajo menos [RELLENAR]

* 1. Como hemos mencionado anteriormente, contamos con una clase Nodo por cada TAD. Esta toma como ”friend class” a su TAD para que este pueda usarla sin problemas.

Para las colas, tomamos dos punteros a NodoCola, primero y ultimo. Si la cola es vacía apuntan a NULL. Cada vez que queremos insertar un elemento creamos un nuevo NodoCola; Si la cola es vacía tanto primero como ultimo apuntarán a él. Si no, apuntamos el siguiente de ultimo al nuevo nodo y este lo convertimos en ultimo. Finalmente incrementamos la longitud de la cola en uno.

Para las pilas, al crearla la cima apunta a null y la profundidad es 0. A medida que vamos apilando o desapilando vehículos la profundidad varía. Al insertar un vehículo este nodo se convierte en cima y apunta al de debajo.

Para las listas, al crearla, el puntero primero y último apuntan a NULL y su longitud es 0. Por cada pedido añadido la longitud aumenta. Al insertar un pedido en la izquierda, el puntero primero apuntará al nuevo nodo, y al insertarlo en la derecha, el puntero último apuntará al nuevo nodo. Si se inserta en una posición, se crea un puntero auxiliar, y se recorre la lista hasta llegar a la posición, donde se cambia el puntero del siguiente al nuevo nodo, y el del nuevo al siguiente.

Para los árboles ABB, tenemos una raíz y dos punteros, uno a cada hijo izquierdo y derecho. Al crear el árbol vacío la raíz es NULL y no hay punteros a los hijos. Si se crea el árbol con una raíz, tenemos un puntero a cada hijo (subárboles). Para insertar un elemento en orden, se compara. Si el elemento es menor o igual a la raíz, se crea un nodo para insertarlo en el árbol como hijo izquierdo.

* + 1. Cabe destacar que hemos implementado cuatro métodos nuevos en gestor para generar nv, ns, nc y np de manera aleatoria y entre los límites establecidos en el enunciado. Se pueden activar en el método main quitando los comentarios, y de esta forma serán aleatorios para cada simulación.

Otro método interesante sería crearVehiculo en la clase gestor, el cual recibe como parámetro nv y genera ese número de coches; creamos un objeto coche auxiliar con el constructor, el cual se vale de otros métodos implementados por nosotros los cuales generan sus atributos de manera aleatoria. Los distintos valores que pueden tomar los atributos se encuentran en vectores y se pueden cambiar fácilmente. Hemos implementado métodos para pedir los datos de los vehículos o pedidos que se desean introducir y así evitar errores a la hora de trabajar con los datos. Hemos creado un método en listas para poder insertar un pedido según su prioridad, ya que nos encontramos con el problema de que, si metíamos los pedidos prioritarios por la izquierda, los primeros prioritarios en salir serían los últimos que se han añadido, cuando los primeros en salir deberían ser los primeros que se añadieron.

* + 1. El programa comienza con un menú de 20 opciones. Para comenzar la simulación crearemos un número de coches, ya sea aleatorio (con nuestro método generar nv, o estableciéndolo en el main) o determinado por el usuario(opción 2), de forma que se crean vehículos de todas las combinaciones posibles. También generaremos un número de pedidos, de forma aleatoria o seleccionando nosotros el número de pedidos y sus características (opciones 5 y 6). Una vez haya coches en las colas (Almacenes de fábrica) podremos observarlos presionando 3. Para eliminar los coches de la fábrica basta con pulsar 4. Con la opción 7 veremos la lista de pedidos ordenados según su prioridad y con la opción 8 podremos borrarlos todos. En el paso 9 salen NS vehículos de la fábrica y llegan a la zona de reparto si coinciden con algún pedido que se haya hecho. Se elimina de la lista de pedidos y de la fábrica y se lleva a la zona. En la opción 10 podremos elegir cuántos vehículos salen de la fábrica. Para finalizar la simulación y sacar todos los vehículos posibles de la fábrica, seleccionaremos la opción 11 del menú. En la opción 12 veremos los datos de las zonas y de los camiones y registros. Podremos añadir al árbol un vehículo con los datos que queramos con la opción 13. Con la 14 se añadirán al árbol los datos de los vehículos de una de las zonas y con la 15 añadiríamos los datos de los de todas las zonas. Con la 16 podremos ver los datos de los vehículos ordenados por su número de concesionario. En el punto 17 eliminamos del árbol los datos de los automóviles de un modelo dado. Se pueden ver los automóviles entregados en un concesionario que se desee con la opción 18. La opción 19 permite ver el concesionario y zona en que se ha entregado un automóvil dando un número de bastidor. Se sale del menú pulsando 0.
  1. Bibliografía: Apuntes de la asignatura.