DOCUMENTACION PL2

Roger Borrat Gutiérrez. DNI:09125748S
Iván Martín Calvo. DNI: 03202770C

DETALLES Y JUSTIFICACIÓN DE LA IMPLEMENTACIÓN:

TAD'S IMPLEMENTADOS:

Cola:
espec COLA[AUTOMÓVIL]
usa BOOL
parámetro formal
géneros automóvil
fparametro
géneros cola

Operaciones:

Generadoras:
Cola(): ->cola{Crea una cola}
encolar(): cola, automovil->cola{Añade un automóvil a la cola}

Modificadoras:
desencolar(): cola->cola{Elimina un automóvil de la cola}
~Cola(): cola->nil{Destruye la cola}

Observadoras:
contar(): cola->entero{Devuelve el numero de automóviles de la cola}
inicio(): cola->automovil{Devuelve el automóvil que esta al principio de la cola}
es_vacia(): cola->bool{Devuelve si la cola está vacia o no}

fin espec

Pila:
espec PILA[AUTOMÓVIL]
usa BOOL
parámetro formal
géneros automóvil
fparametro
géneros pila

Operaciones:

Generadoras:
Pila(): ->pila{Crea una pila}
apilar(): pila, automovil->pila{Añade un automóvil a la pila}

Modificadoras:
desapilar(): pila->pila{Elimina un automóvil de la pila}
~Pila(): pila->nil{Destruye la pila}

Observadoras:
contar(): pila->entero{Devuelve el numero de automóviles de la pila}
verCima(): pila->automovil{Devuelve el automóvil que esta al principio de la pila}
esVacia(): pila->bool{Devuelve si la pila está vacia o no}

fin espec

Lista:
espec LISTA[PEDIDO]
usa BOOL
parámetro formal
géneros automóvil
fparametro
géneros lista

Operaciones:
Generadoras:
Lista(): ->lista{Crea una lista}
insertar_derecha(): lista, pedido->lista{Añade un pedido a la lista por la derecha}
insertar_izquierda(): lista, pedido->lista{Añade un pedido a la lista por la izquierda}
insertar_enPosicion(): lista, pedido,int->lista{Añade un pedido a la lista en una posicion dada}

Modificadoras:
borrar_izquierda(): lista, ->lista{Borra el primer pedido}
borrar_posicion(): lista,entero->lista{Borra un el pedido de la lista de una posicion dada}
~Lista(): lista->nil{Destruye la lista}

Observadoras:
get_longitud(): lista->entero{Devuelve el numero de pedidos de la lista}
ver_primero(): lista->pedido{Devuelve el primer pedido de la lista}
ver_ultimo(): lista->pedido{Devuelve el ultimo pedido de la lista}
ver_posicion(): lista entero ->pedido{Devuelve el un pedido dado por posicion de la lista}
es_vacia(): lista->bool{Devuelve si la lista está vacia o no}

fin espec

Lista_Aut:
espec LISTA_AUT[AUTOMOVIL]
usa BOOL

parámetro formal
géneros automóvil
fparametro
géneros lista_aut

Operaciones:
Generadoras:
Lista_Aut(): ->lista_aut{Crea una lista}
insertar_derecha(): lista_aut, automovil->lista_aut{Añade un automovil a la lista por la derecha}
insertar_izquierda(): lista_aut, automovil->lista_aut{Añade un automovil a la lista por la izquierda}
insertar_enPosicion(): lista_aut, automovil,int->lista_aut{Añade un automovil a la lista en una posicion dada}

Modificadoras:
borrar_izquierda(): lista_aut ->lista_aut{Borra el primer automovil}
borrar_posicion(): lista_aut,entero->lista{Borra un el automovil de la lista de una posicion dada}
~Lista(): lista_aut->nil{Destruye la lista}

Observadoras:
get_longitud(): lista_aut->entero{Devuelve el numero de automoviles de la lista}
ver_primero(): lista_aut->automovil{Devuelve el primer automovil de la lista}
ver_ultimo(): lista_aut->automovil{Devuelve el ultimo automovil de la lista}
ver_posicion(): lista_aut, entero ->pedido{Devuelve el un automovil dado por posicion de la lista}
es_vacia(): lista_aut->bool{Devuelve si la lista está vacia o no}

fin espec

<u>ABB:</u>
espec ABB[CONCESIONARIO]
usa BOOL
parámetro formal
géneros automóvil
fparametro
géneros abb

Operaciones:
Generadoras:
ABB(): ->ABB{Crea un arbol vacio}
ABB():concesionario ->ABB{Crea un arbol con un concesionario especifico como raiz}
insertar(): ABB, concesionario->ABB{Añade un concesionario al arbol}

Modificadoras:
~ABB(): ABB->nil{Destruye el arbol}

Observadoras:
verInOrden(): ABB->{Muestra el arbol ordenado por orden}

DIFICULTADES ENCONTRADAS:

Hemos tenido dificultades con las opciones 9 y 14 del menú ya que no somos capaces de hacerlas funcionar correctamente, por algún motivo que somos incapaces de razonar falla la función desencolar en la línea del delete(). Sin embargo, las funciones que dependen de las mismas como son la 11 y la 15 respectivamente funcionan con total normalidad. A estos no hemos podido encontrar solución.

También tuvimos problemas con los métodos que recorrían listas, hasta que nos dimos cuenta de que las posiciones en las listas comenzaban en la 1 y terminaban es la de la longitud de la lista.

En las clases lista y lista_aut la función borrar_posicion , nos daba problemas, por lo que muchos métodos que dependen de esta también, finalmente, nos dimos cuenta que esta función no hacía longitud– como sí hacían las otras funciones para borrar, al añadir esto ya no daba problemas.

En la opción 17, conseguimos que borre los automóviles que comparten modelo, con el modelo seleccionado, pero no hemos logrado arreglar la función borrar_vacios para que borre los nodos que tengan la lista de automóviles vacia.

MÉTODOS MÁS DESTACADOS:

Queremos destacar los métodos “generar_automoviles_ModCol()” , “genear_pedidos()” , “simulacion_completa()” y “zonas_a_arbol()” todas ellas son una parte crucial en el desarrollo de la simulación. Por una parte, generar_automoviles_ModCol() nos permite generar NV automóviles de cada modelo y color y añadirlos a la cola de fábrica lo que facilita la generación masiva de automóviles. Por otra parte, generar_pedidos() generar aleatoriamente al menos $NC*2^2$ pedidos de los concesionarios y los inserta en la lista de pedidos. simulacion_completa() distribuye los automoviles de la cola de fabrica a lo largo de las zonas de reparto dependiendo de los pedidos que haya en la lista de pedidos, si coincide con algun pedido se enviara a dicha zona, si no se desechará (se envian de 1 en 1, cuando se llenan los 2 camiones que tienen capacidad NP se envian a la cola de zona). Finalmente zonas_a_arbol() saca los automoviles de las cola_de_zona y los envia al arbol binario donde los alamacenaremos segun su numero de concesionario, si comparten concesionario compartiran nodo y se almacenarán en la lista que tiene cada nodo.

COMPORTAMIENTO DEL PROGRAMA:

Al ejecutar el programa, lo primero que saldrán serán las diferentes opciones. El usuario escoge una opción de las indicadas en el menú, resaltando que para que sea una buena ejecución será necesario ejecutar las opciones 1 o 2, y las opciones 5 o 6 independientemente de su orden, esto con el fin de que al avanzar la simulación haya tanto pedidos como automóviles que hagan posible el avance de la simulación, ya sea parcial (opción 9 o 10) o total (opcion 11). Posteriormente moverlos al árbol de almacenamiento de concesionarios con el 14 o 15. Las demás opciones se ven en el menú y sirven desde borrar los automóviles disponibles en la cola fabrica, como general o mostrar datos de diferentes estructuras (las colas, pilas, listas o árbol). También se incluyen funciones que permiten la eliminación de elementos del árbol. La opción 20 finalizará el programa.

BIBLIOGRAFÍA:

Contenido (blackboard.com)
Y diferentes paginas relacionadas con C++ que ahora no conseguimos recordar.