TADs

TAD Vértice:

Conjunto de datos:

- coordX, real, indica la coordenada en x del vértice.
- coordY, real, indica la coordenada en y del vértice.
- coordZ, real, indica la coordenada en z del vértice.

Comportamiento (operaciones) del objeto:

- Vertice(), constructor de vértices.
- ObtenerCoorX(), retorna el valor de la coordenada x del vértice.
- FijarCoorX(x), asigna x al valor de la coordenada x del vértice.
- ObtenerCoorY(), retorna el valor de la coordenada y del vértice.
- FijarCoorY(y), asigna y al valor de la coordenada y del vértice.
- ObtenerCoorZ(), retorna el valor de la coordenada z del vértice.
- FijarCoorZ(z), asigna z al valor de la coordenada z del vértice.

TAD NodoCTP:

Conjunto de datos:

- dato, Vertice, indica el vértice que almacena este nodo.
- indice, entero, indica el número identificador del vértice que almacena el nodo.
- hijo1, apuntador de un vértice, indica el hijo 1 de el nodo.
- hijo2, apuntador de un vértice, indica el hijo 2 de el nodo.
- hijo3, apuntador de un vértice, indica el hijo 3 de el nodo.
- hijo4, apuntador de un vértice, indica el hijo 4 de el nodo.
- hijo5, apuntador de un vértice, indica el hijo 5 de el nodo.
- hijo6, apuntador de un vértice, indica el hijo 6 de el nodo.
- hijo7, apuntador de un vértice, indica el hijo 7 de el nodo.
- hijo8, apuntador de un vértice, indica el hijo 8 de el nodo.

Comportamiento (operaciones) del objeto:

- NodoCTP(), constructor de NodoCTP.
- fijarDato(dat), asigna dat al valor de el vértice dato.
- obtenerDato(), retorna el vértice dato.
- fijarIndice(ind), asigna ind al valor del índice.
- obtenerIndice(), retorna el valor del índice.
- fijarHijo1(hijo), asigna el valor de hijo al hijo 1.
- obtenerHijo1(), retorna el valor del hijo 1.
- fijarHijo2(hijo), asigna el valor de hijo al hijo 2.
- obtenerHijo2(), retorna el valor del hijo 2.
- fijarHijo3(hijo), asigna el valor de hijo al hijo 3.

- obtenerHijo3(), retorna el valor del hijo 3.
- fijarHijo4(hijo), asigna el valor de hijo al hijo 4.
- obtenerHijo4(), retorna el valor del hijo 4.
- fijarHijo5(hijo), asigna el valor de hijo al hijo 5.
- obtenerHijo5(), retorna el valor del hijo 5.
- fijarHijo6(hijo), asigna el valor de hijo al hijo 6.
- obtenerHijo6(), retorna el valor del hijo 6.
- fijarHijo7(hijo), asigna el valor de hijo al hijo 7.
- obtenerHijo7(), retorna el valor del hijo 7.
- fijarHijo8(hijo), asigna el valor de hijo al hijo 8.
- obtenerHijo8(), retorna el valor del hijo 8.
- VerticeCercano(x,y,z), retorna una cola con el índice y la distancia del nodo mas cercano al punto xyz..
- buscarNodoInd(buscado), retorna el índice de el vértice buscado.

TAD OctreePuntos:

Conjunto de datos:
- raíz, apuntador de un NodoCTP, indica donde inicia el árbol de vértices.
- cantVertices, entero, cantidad de nodos en el árbol.

Comportamiento (operaciones) del objeto:
- OctreePuntos(), constructor del OctreePuntos.
- fijarRaiz(nuevo),asigna el valor de nuevo a la raíz.
- obtenerRaiz(), retorna un apuntador a la raíz del árbol.
- obtenerCantVertices(), retorna la cantidad de nodos en el árbol.
- insertarPunto(x,y,z), inserta un punto con las coordenadas x, y y z.
- listarVertices(), retorna un lista de vértices, donde cada vértice es uno de los vértices que están en los nodos del árbol.
- VerticeCercano(x,y,z), retorna un cola con el vértice más cercano a el punto con coordenadas x y z.
- buscarNodoInd(buscado), retorna el índice de el vértice buscado.

TAD Arista:

Conjunto de datos:
- inicio, apuntador de un vértice, indica donde inicia la arista.
- fin, apuntador de un vértice, indica donde finaliza la arista.

Comportamiento (operaciones) del objeto:
- Arista(), constructor de la arista.
- ObtenerInicio(), retorna el vértice donde inicia la arista.
- FijarInicio(ver), asigna el valor de ver a el valor del vértice donde inicia la arista.

- ObtenerFin(), retorna el vértice donde finaliza la arista.
- FijarFin(ver), asigna el valor de ver a el valor del vértice donde inicia la arista

TAD Cara:

Conjunto de datos:
- tamano, entero largo, indica el tamaño de la cara
- indicesP, vector de enteros, conjunto de índices de los vértices que forman la cara.
- tamano, entero largo ,tamaño de la cara

Comportamiento (operaciones) del objeto:
- Cara(), constructor de la cara.
- ObtenerAristas(), retorna una lista con las aristas de la cara.
- FijarAristas(lista), asigna una lista de aristas a la cara.
- ObtenerTamaño(), retorna el tamaño de la cara.
- FijarTamaño(tam), asigna el valor de tam a el tamaño de la cara.
- ObtenerCantAristas(), retorna el tamaño de la lista de aristas.
- ObtenerIndicesP(), retorna un vector con los índices de los vértices que conforman la cara.
- FijarIndicesP(lista), asigna el vector lista a el vector indicesP.

TAD Objeto:

Conjunto de datos:
- nombre, cadena de caracteres, indica el nombre del objeto.
- l_caras, lista de cara, conjunto de caras que conforman el objeto.
- cantVertices, entero largo, representa la cantidad de vértices del objeto.
- Oc_vertices, OctreePuntos, árbol con el conjunto de vértices que conforman el objeto.
- l_aristas, vectorde arista, conjunto de aristas que conforman el objeto.
- cantAristas, entero largo, cantidad de aristas del objeto.

Comportamiento (operaciones) del objeto:
- Objeto(), constructor del objeto.
- ObtenerNombre(), retorna el nombre del objeto.
- FijarNombre(nom), asigna el valor de nom al nombre del objeto.
- ObtenerCaras(), retorna una lista con las caras del objeto.
- FijarCaras(caras), asigna la lista de caras a la lista de caras del objeto.
- VerticeExtremoMenor(), retorna el apuntador a un vértice extremo menor.
- VerticeExtremoMayor(), retorna el valor a un vértice extremo mayor.
- ObtenerCantVertices(), retorna la cantidad de vértices.
- Datos(), retorna información (vértices, caras y aristas) acerca del objeto.
- ObtenerCantidadAristas(), retorna el valor total de las aristas del objeto.

- FijarVertices(vertices), asigna el árbol vértices a el Oc_vertices
- FijarAristas(aristas), asigna el vector aristas a l_aristas.
- ObtenerAristas(), retorna el vector de aristas del objeto.
- ObtenerVertices(), retorna el arbolde vértices del objeto.
- FijarCaras(caras), asigna la lista caras a l_caras.
- FijarCantidadVertices(ver), asigna ver a cantVertices
- FijarCantidadAristas(ver), asigna ver a cantAristas.
- VerticeCercano(x,y,z), retorna una cola con el índice del vértice y la distancia entre este vértice y el punto x y x.
- coversionAG(), retorna el grafo de la malla.
- calcularCentroInd(), retorna el índice del nodo mas cercano a el centro del grafo.
- rutaCorta(origen,destino), retorna la ruta más corta entre origen y destino.

TAD Sistema

Conjunto de datos:

- l_objetos, lista de objetos, conjunto de objetos del sistema.

Comportamiento (operaciones) del objeto:

- Sistema(), constructor del sistema.
- eliminarObjeto(nombre), elimina el objeto que tiene el nombre.
- buscarObjeto(nombre), retorna un iterador que apunta al objeto si lo encontró.
- DatosPorObjeto(), retorna la información de todos los objetos.
- CargarArchivo(), lee un objeto de un archivo y retorna un mensaje en caso de que se pueda leer, no se pueda leer o el archivo no tenga el fromato de un obejto 3D.
- ObtenerObjetos(), retorna la lista de objetos del sistema.
- GuardarObjeto(nombre,archivo), toma el objeto con nombre nombre y lo guarda en un archivo con el nombre archivo.
- CrearEnvolventeObjeto(objeto), crea una caja donde se puede guardar el objeto, si el resultado es exitoso retorna verdadero.
- CrearEnvolvente(), crea una caja donde se pueden guardar todos los objetos, si el resultado es exitoso retorna verdadero.
- verticeCercanoPuntoxObjeto(nombre,x,y,z), retorna un string mencionando cual es el vértice del objeto con nombre nombre mas cercano a el punto con coordenadas x y z.
- verticeCercanoPuntoxObjetoGeneral(x,y,z),), retorna un string mencionando cual es el vértice mas cercano a el punto con coordenadas x y z, de todos los objetos cargados en el sistema, al igual que la distancia y el nombre del objeto al que pertenece el vértice.

- verticeCercanoxEnvolvente(nombre), retorna un string con cual es el vértice mas cercano a cada vértice del envolvente del objeto, al igual que la distancia.

TAD vecino

Conjunto de datos:
- dato, plantilla, representa el nodo al que apunta el vecino.
- Precio, flotante, conto para llegar al nodo que apunta el vecino.

Comportamiento (operaciones) del objeto:
- Vecino(), constructor.
- insertarDato(nuevo), asigna el valor de nuevo a dato.
- obtenerDato(), retorna el valor de dato.
- fijarPrecio(nuevo), asigna el valor de nuevo aprecio.
- obtenerPrecio(), retorna el valor de precio.

TAD nodoGrafo

Conjunto de datos:
- dato, plantilla, dato que guarda el nodo del grafo
- vecinos, lista de tipo vecino, lista de vecinos del nodo
- yaPase, boleano, sirve para saber si ya se visito el nodo o no.
- Índice, entero, índice del nodoGrafo

Comportamiento (operaciones) del objeto:
- nodoGrafo(), constructor del nodo
- insertarDato(datoNuevo), recibe datoNuevo y asigna el valor de este a dato
- obtenerDato(), retorna el valor de dato.
- obtenerVecinos(), retorna la lista de vecinos del nodo
- insertarVecino(nodo), recibe un nodo de tipo vecino y lo asigna al la lista de vecinos.
- yaPaso(), retorna si ya se visito el nodo o no.
- fijarYaPase(estado), asigna el valor de estado a yaPase.
- buscarVecino(veci), retorna un iterador con el nodo veci
- eliminarVecino(nodo), elimina el vecino nodo.
- cantVecinos(), retorna la cantidad de vecinos
- obtenerIndice(), retorna el índice del nodoGrafo
- fijarIndice(i), inserta el valor de i a índice.

TAD grafo

Conjunto de datos:
- g, mapa, mapa donde se encuentran todo los nodos del grafo.

- cantVer, entero, cantidad de vértices del grafo.
- Dirigido, booleano, el grafo es dirigido o no.

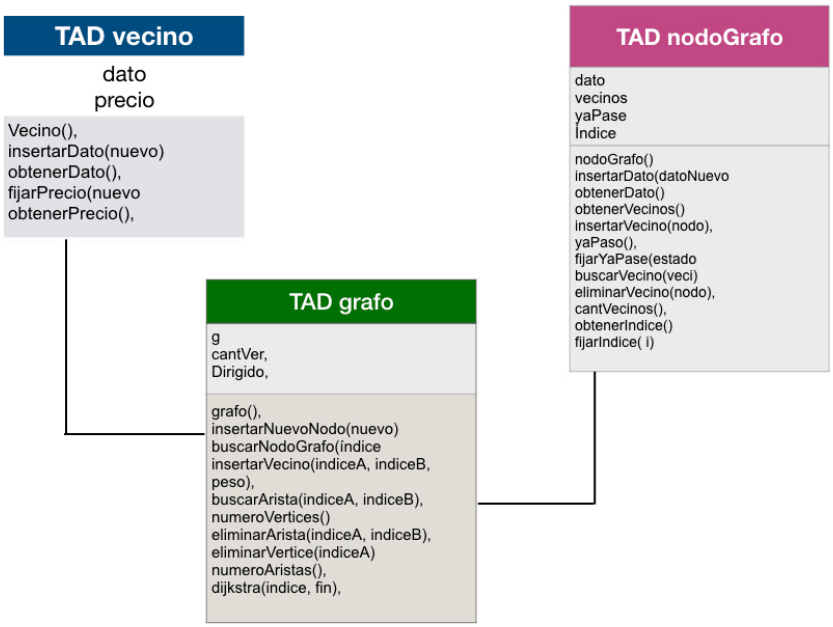
Comportamiento (operaciones) del objeto:

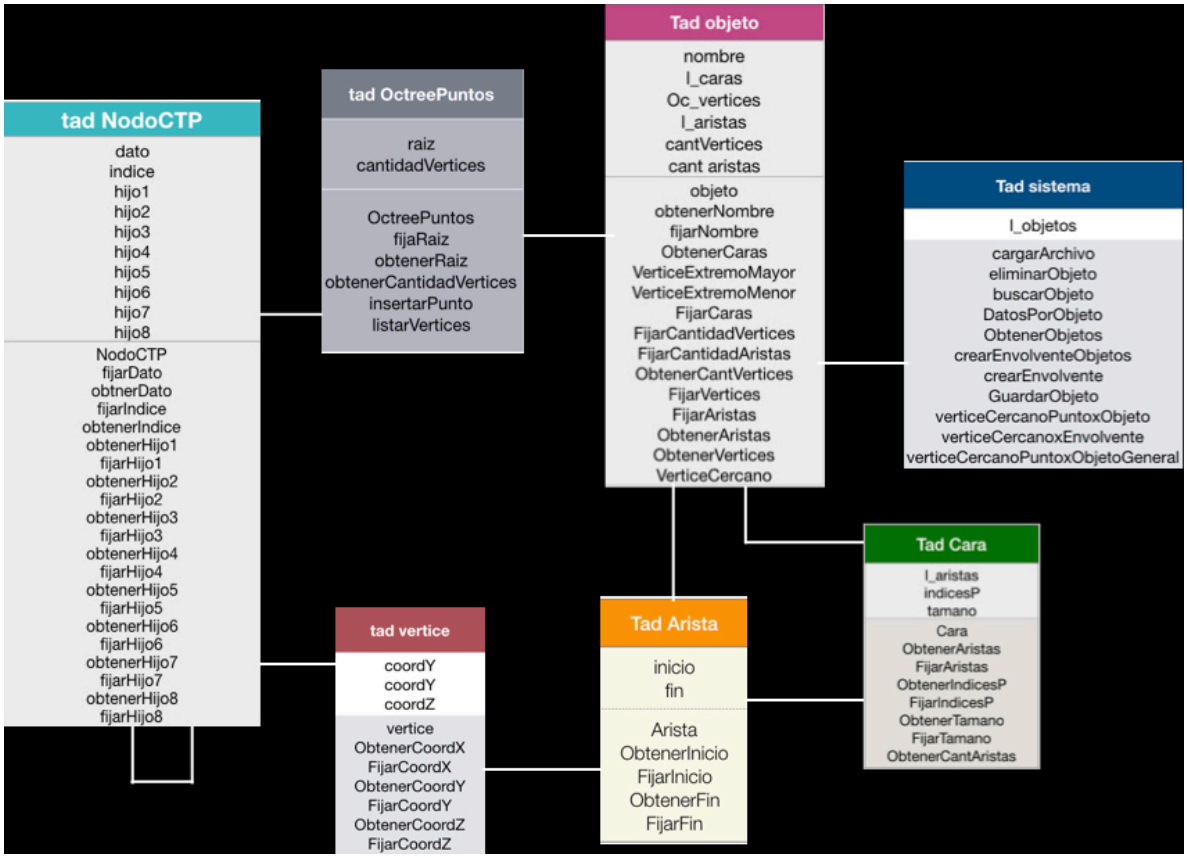
- grafo(), constructor de el grafo.
- insertarNuevoNodo(nuevo), inserta el nodo nuevo al mapa.
- buscarNodoGrafo(índice), retorna el nodo al que le pertenece el índice índice.
- insertarVecino(indiceA, indiceB, peso), crea una arista entre el nodo con indiceA y el nodo con indiceB.
- buscarArista(indiceA, indiceB), busca si existe una arista entre el nodo con indiceA, y el nodo con el indiceB
- numeroVertices(), retorna la cantidad de nodos.
- eliminarArista(indiceA, indiceB), elimina la arista entre el nodo con indiceA y el nodo con indiceB.
- eliminarVertice(indiceA), elimina el nodo con indiceA
- numeroAristas(), retorna la cantidad de aristas que tiene el grafo.
- dijkstra(indice, fin), retorna un string con el camino de dijkstra entre el nodo con índice índice y el nodo con índice fin.

Análisis y diseño.

Tokenizar(linea, separador): recibe una cadena de caracteres y un separador, retorna una cola con los datos de la cadena de caracteres.

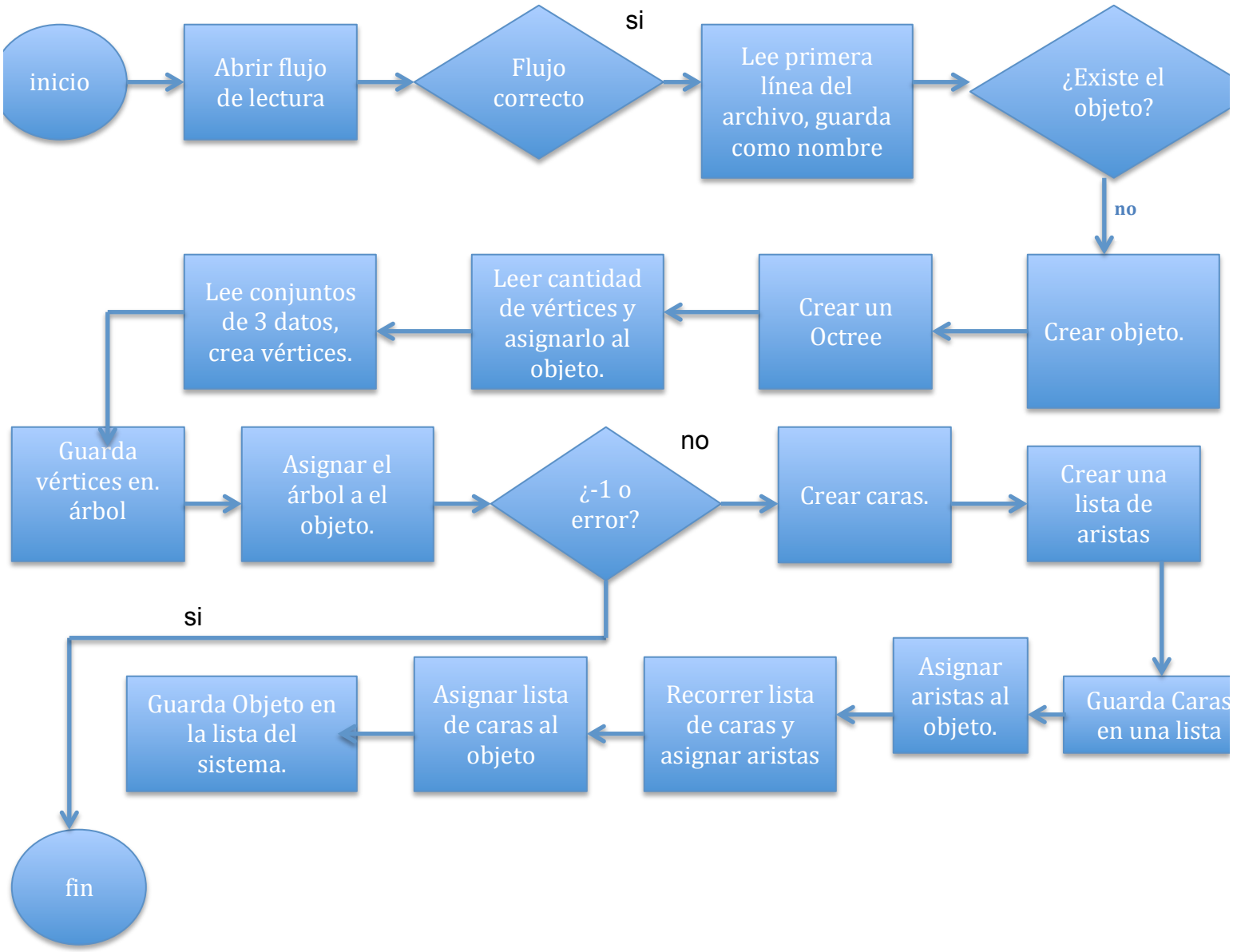
Esquemáticos.





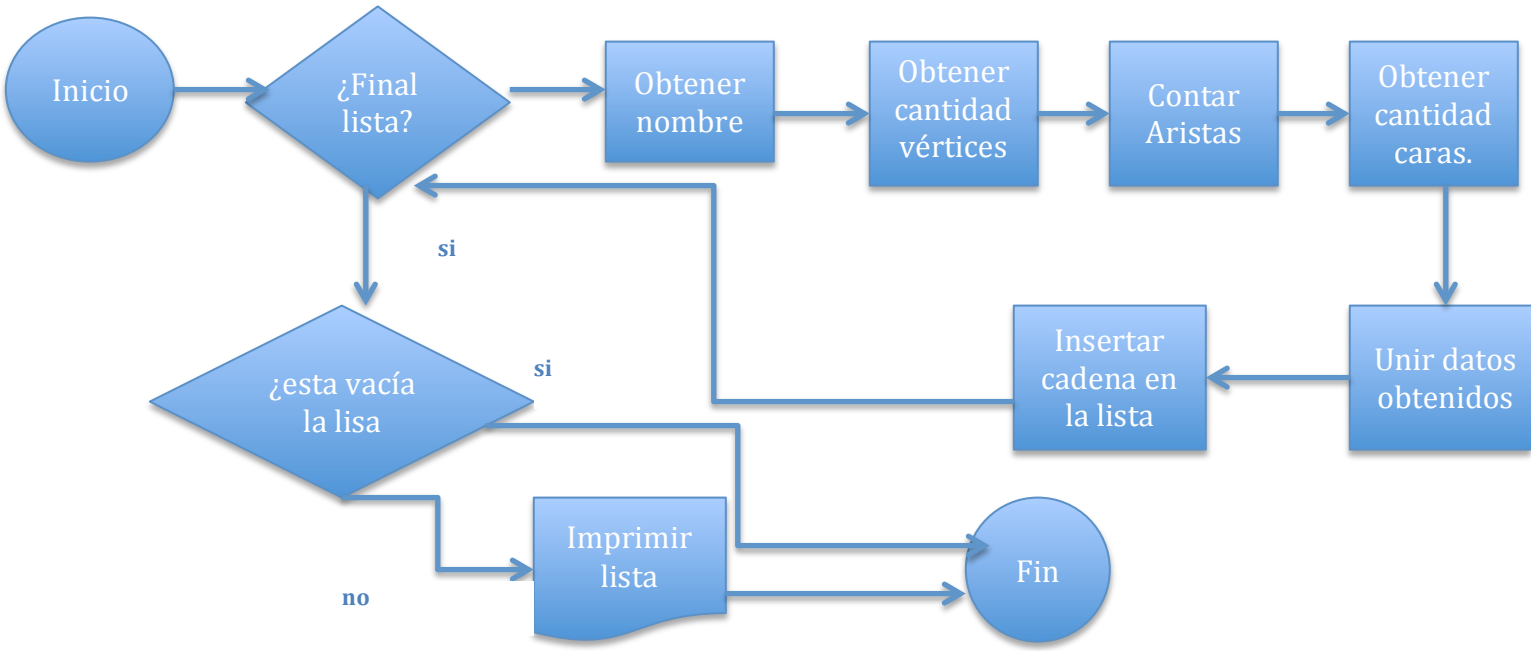
Comando cargar nombre:

1. Abrir flujo de lectura
2. Verificar que el flujo este correcto.
   En caso de que el flujo sea correcto:
3. Leer la primera línea del archivo y guardarla como nombre.
4. Buscar si ya existe el objeto
   En caso de que no exista:
5. Crear un objeto con el nombre que tiene el archivo.
6. Crear un Octree.
7. Leer la cantidad de vértices y asignarlo al objeto antes creado.
8. Leer de a 3 datos.
9. Crear un vértice con los tres datos leidos.
10. Agregarlos al árbol de vértices con su respectivo índice.
11. Asignar el árbol al objeto.
12. Lee el archivo hasta un fallo o un -1.
13. Crea una cara, le asigna tamaño.
14. Crea una lista de aristas.
15. Guarda la cara en una lista de caras.
16. Asigna las aristas al objeto
17. Recorre la lista de caras y le asigna las aristas correspondientes.
18. Asigna la lista de caras a la lista de caras del objeto.
19. Guarda el objeto en la lista de objetos del sistema.



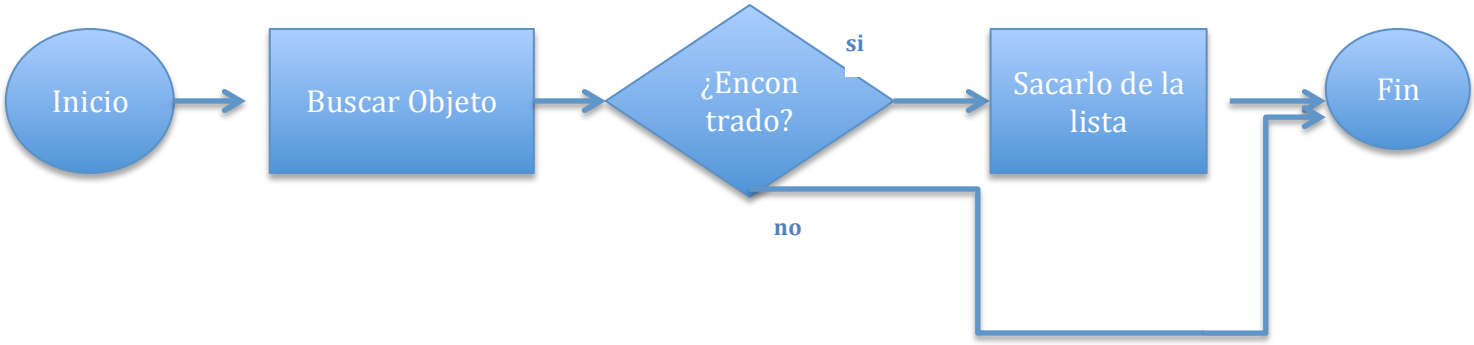
Comando listado:

1. Verificar que la lista de objetos no este vacía.
2. Obtener nombre del objeto.
3. Obtener cantidad de vértices.
4. Contar aristas del objeto.
5. Obtener cantidad de caras.
6. Unir todos los datos en una cadena de caracteres.
7. Colocar la cadena de caracteres en una lista.
8. Repetir los pasos anteriores con cada uno de los objetos.
9. Imprimir la lista.



Comando descargar nombre_objeto:

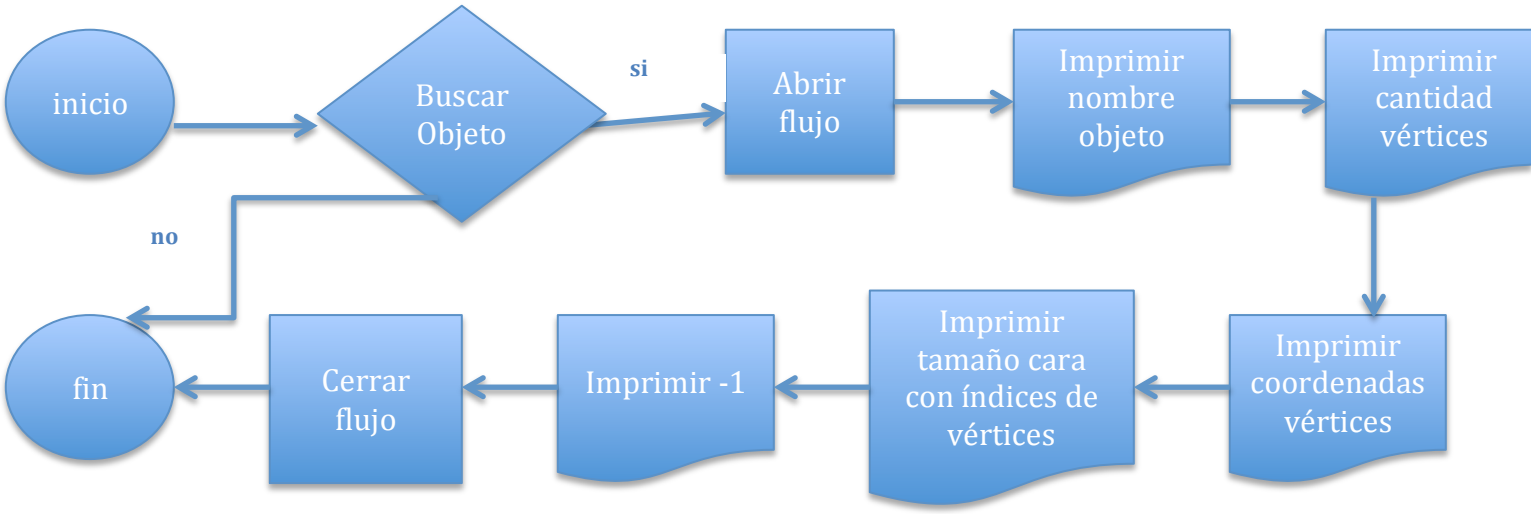
1. Buscar objeto en la lista de objetos.
2. En caso de ser encontrado, sacarlo de la lista.



Comando guardar nombre_objeto nombre_archivo:

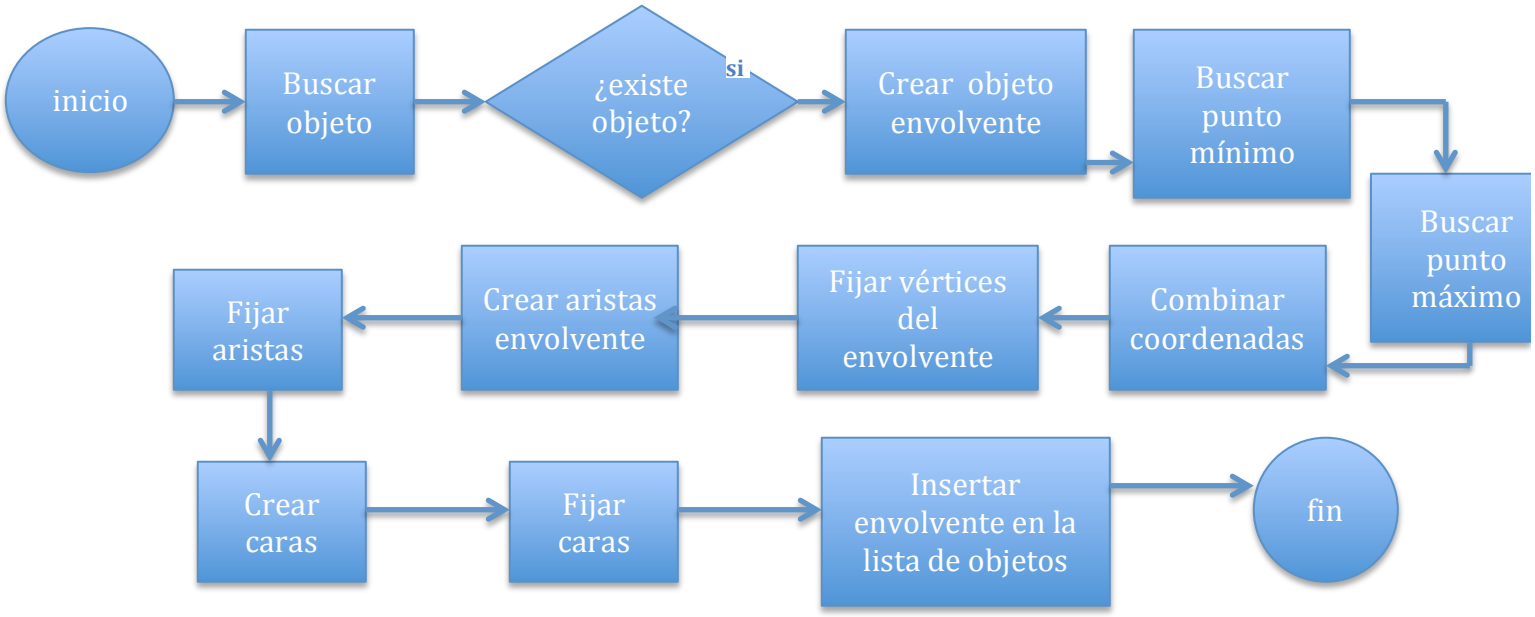
1. Buscar Objeto.
   En caso de encontrarlo:
2. Abrir flujo de escritura.
3. Imprimir en el archivo el nombre del objeto.
4. Imprimir en el archivo la cantidad de vértices del objeto.
5. Imprimir en el archivo las coordenadas de cada vértice.
6. Imprimir en el archivo el tamaño de la cara con índices de los vértices.
7. Imprimir en el archivo -1.

8. Cerrar flujo.



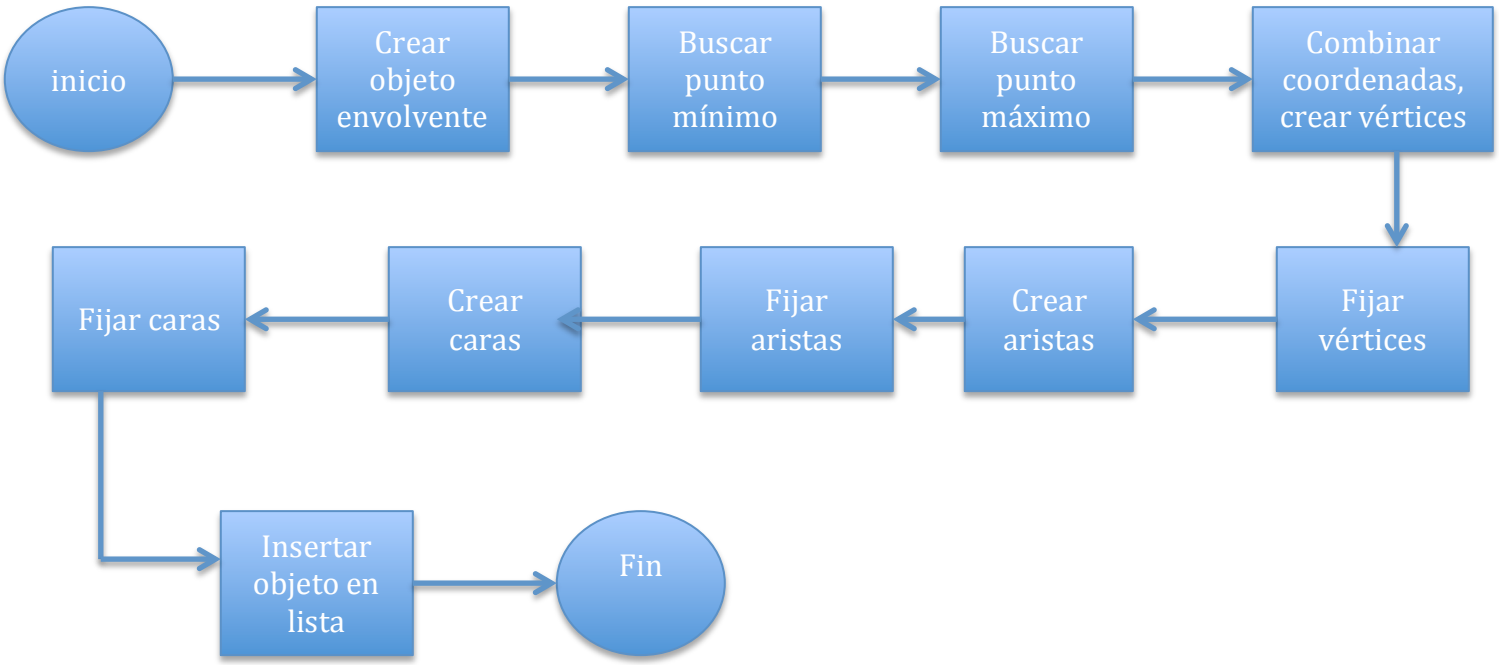
Comando envolvente nombre_objeto:

1. Buscar objeto.
2. Crear objeto envolvente.
3. Buscar punto mínimo del objeto, (se genera tomando las coordenadas mas pequeñas en cada eje).
4. Buscar punto máximo del objeto, (se genera tomando las coordenadas mas grandes en cada eje) .
5. Combinar coordenadas en x y z, entre las del punto máximo y el punto mínimo, para crear los vértices de la caja.
6. Fijar los vértices del objeto.
7. Crear aristas de la caja envolvente.
8. Fijar las aristas del objeto.
9. Crear caras del objeto.
10. Fijar caras.
11. Insertar objeto en la lista de objetos del sistema .



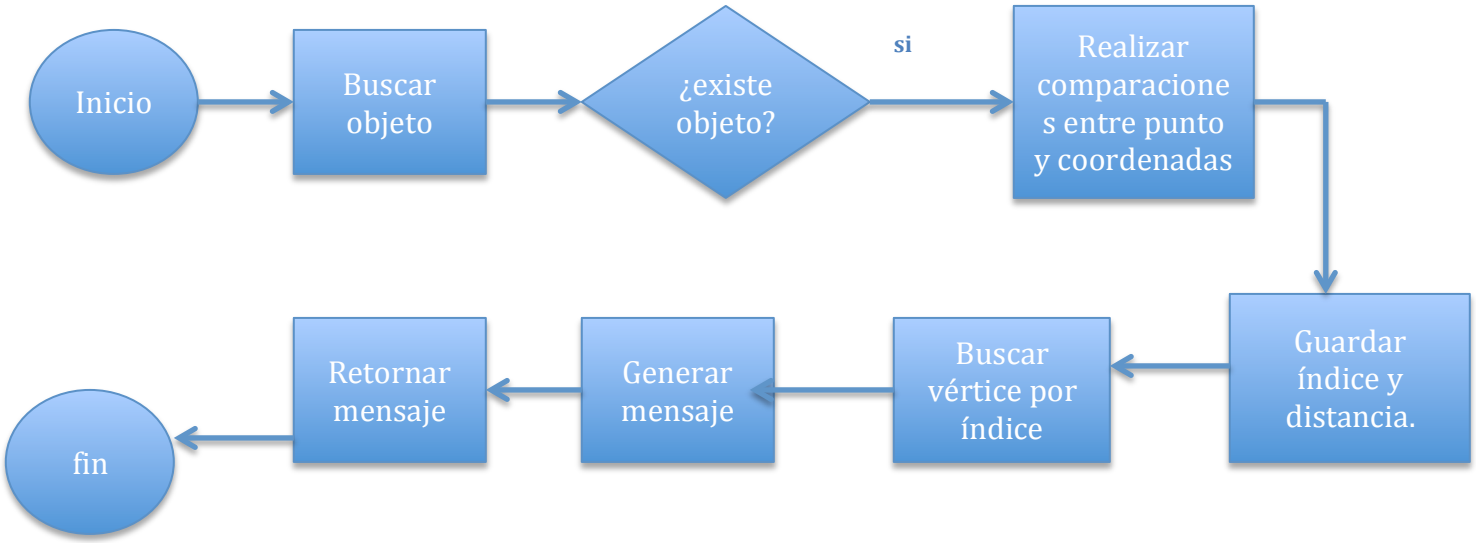
Comando envolvente:

1. Crear objeto envolvente total.
2. Buscar punto mínimo entre todos los objetos del sistema.
3. Buscar punto máximo entre todos los objetos del sistema.
4. Combinar coordenadas entre los puntos mínimo y máximo, para generar los vértices del objeto.
5. Fijar los vértices del objeto.
6. Crear aristas de la caja envolvente.
7. Fijar las aristas del objeto.
8. Crear caras del objeto.
9. Fijar caras.
10. Insertar objeto en la lista de objetos del sistema .



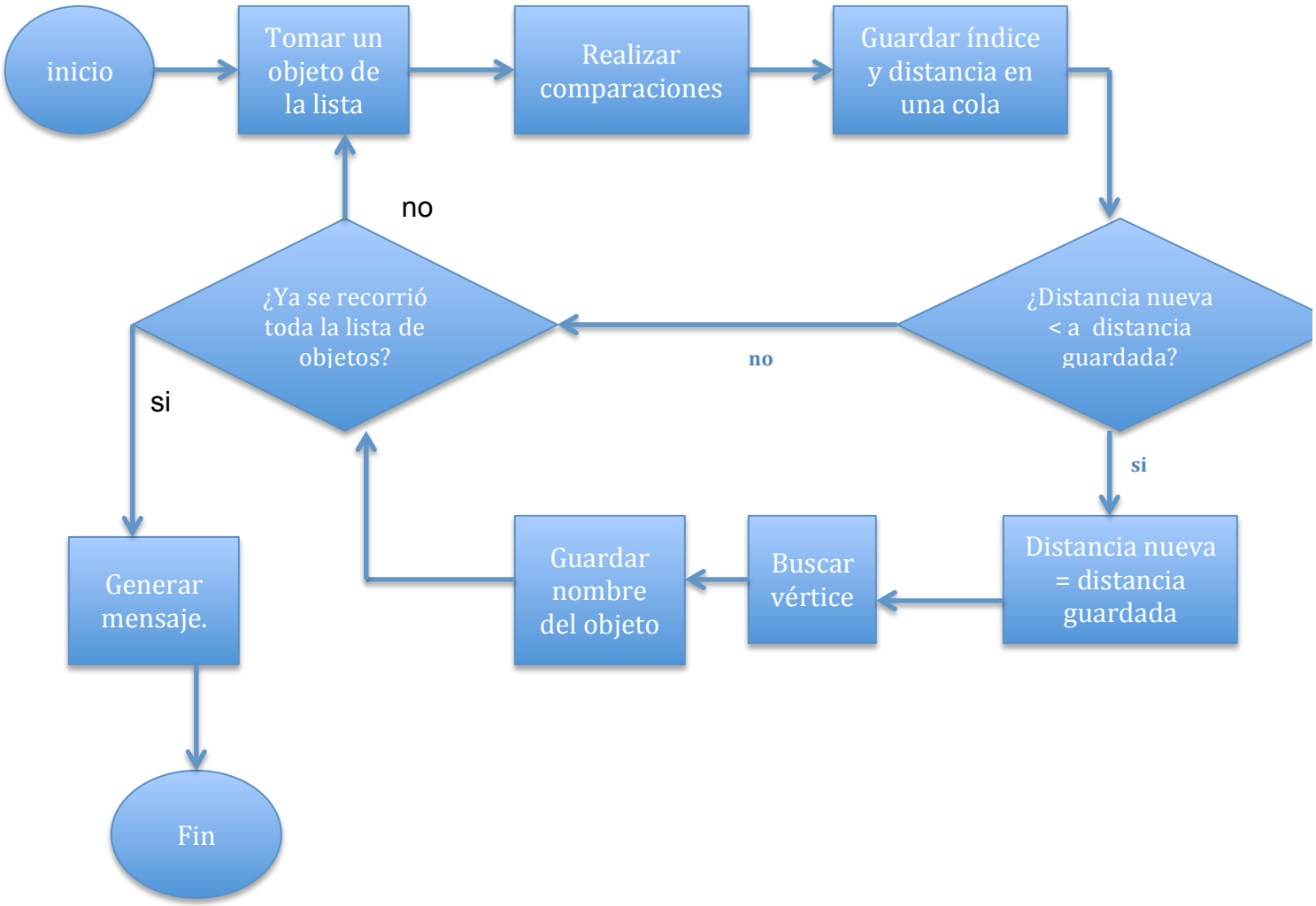
Comando v_cercano px py pz nombre_objeto:

1. Buscar el objeto.
   En caso de que el objeto se encuentre cargado en el sistema:
2. Realiza comparaciones de distancia entre sus vértices y el punto y va guardando el índice vértice que se va encontrando a menor distancia del punto, al igual que su distancia al punto.
3. Guarda el índice del vértice que tiene la distancia menor distancia, junto con la distancia al punto en una cola.
4. Busca el vértice por el índice.
5. Genera un mensaje mencionando el índice del vértice, sus coordenadas, el nombre del objeto al que pertenece y la distancia a la que esta del punto.
6. Retorna el mensaje.



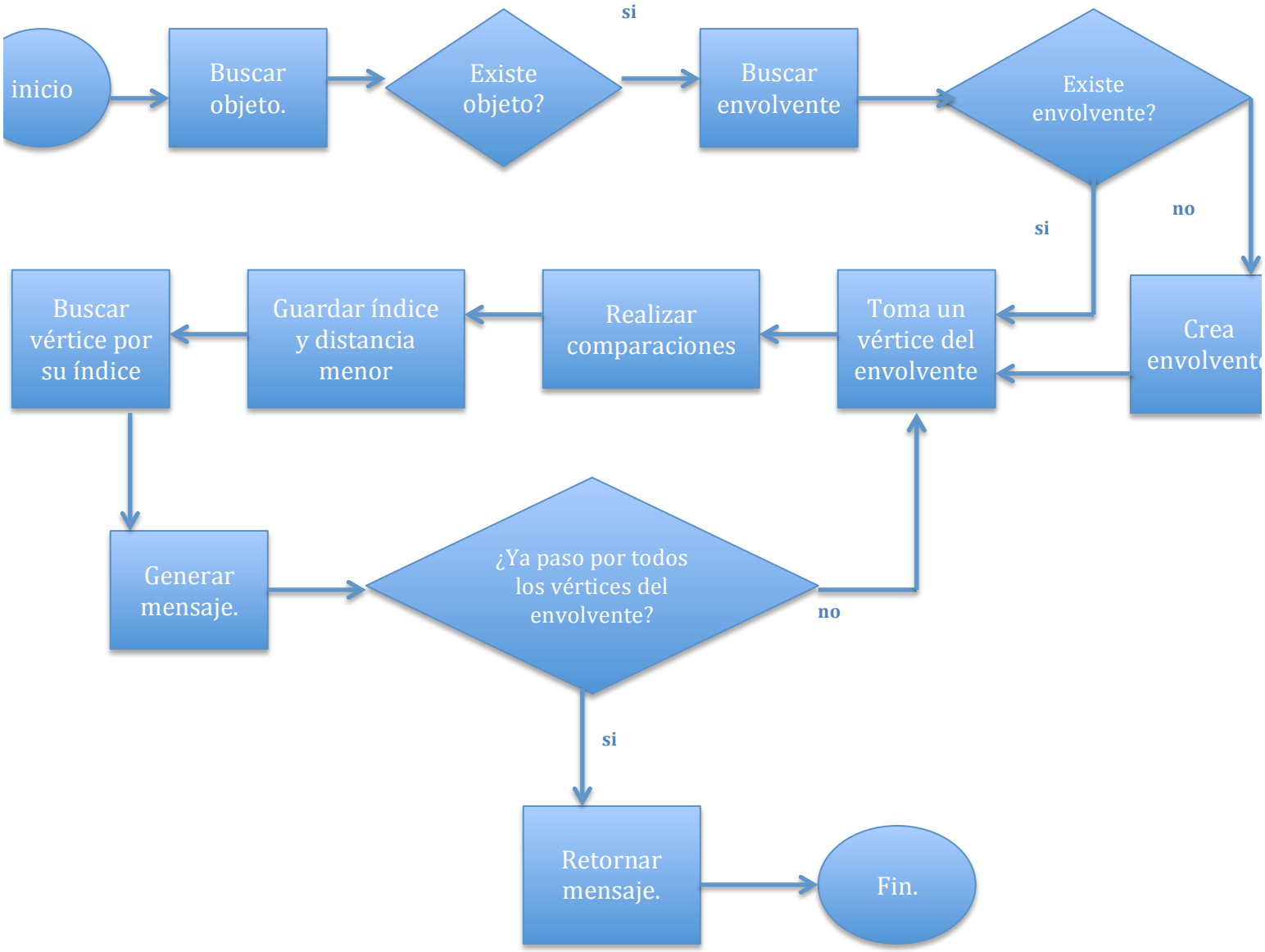
Comando v_cercano px py pz:

1. Tomar un objeto de la lista de objetos del sistema.
2. Realiza comparaciones de distancia entre sus vértices y el punto y va guardando el índice vértice que se va encontrando a menor distancia del punto, al igual que su distancia al punto.
3. Guarda el índice del vértice que tiene la distancia menor distancia, junto con la distancia al punto en una cola.
4. Realiza una comparación entre la distancia obtenida y la distancia guardada en una variable auxiliar.
   En caso de que la distancia obtenida sea menor:/en caso contrario saltar al paso 9.
5. Se guarda la distancia obtenida en una variable auxiliar.
6. Se busca el vértice al cual le pertenece ese índice.
7. Se guarda el nombre del objeto en el que se esta.
8. Se revisa si no se ha recorrido toda la lista de objetos.
9. En caso de que no se haya recorrido toda la lista, se repiten los pasos anteriores.
10. En caso de que ya se haya recorrido toda la lista de objetos, se genera un mensaje con el nombre del objeto que tiene el vértice con menor distancia al punto, las coordenadas del vértice, y la distancia entre el vértice y el punto.
11. Retorna el mensaje.



Comando v_cercanos_caja nombre_onjeto:

1. Busca el objeto.
   En caso de que el objeto exista:
2. Buscar el envolvente de este objeto.
   En caso de que no exista:
3. Crea el envolvente.
   Ya teniendo en el envolvente:
4. Toma un vértice del envolvente.
5. Realiza comparaciones de distancia entre los vértices del objeto y el vértice del envolvente y va guardando el índice vértice que se va encontrando a menor distancia del vértice, al igual que su distancia al vértice.
6. Guarda el índice del vértice que tiene la distancia menor distancia, junto con la distancia al vértice en una cola.
7. Buscar el vértice por su índice.
8. Genera un mensaje con las coordenadas del vértice del envolvente, las coordenadas de el vértice del objeto y la distancia entre ambos vértices.
9. Continuar los pasos del 4 al 8 hasta que ya se pase por todos los vértices del envolvente.
10. Retornar el mensaje.

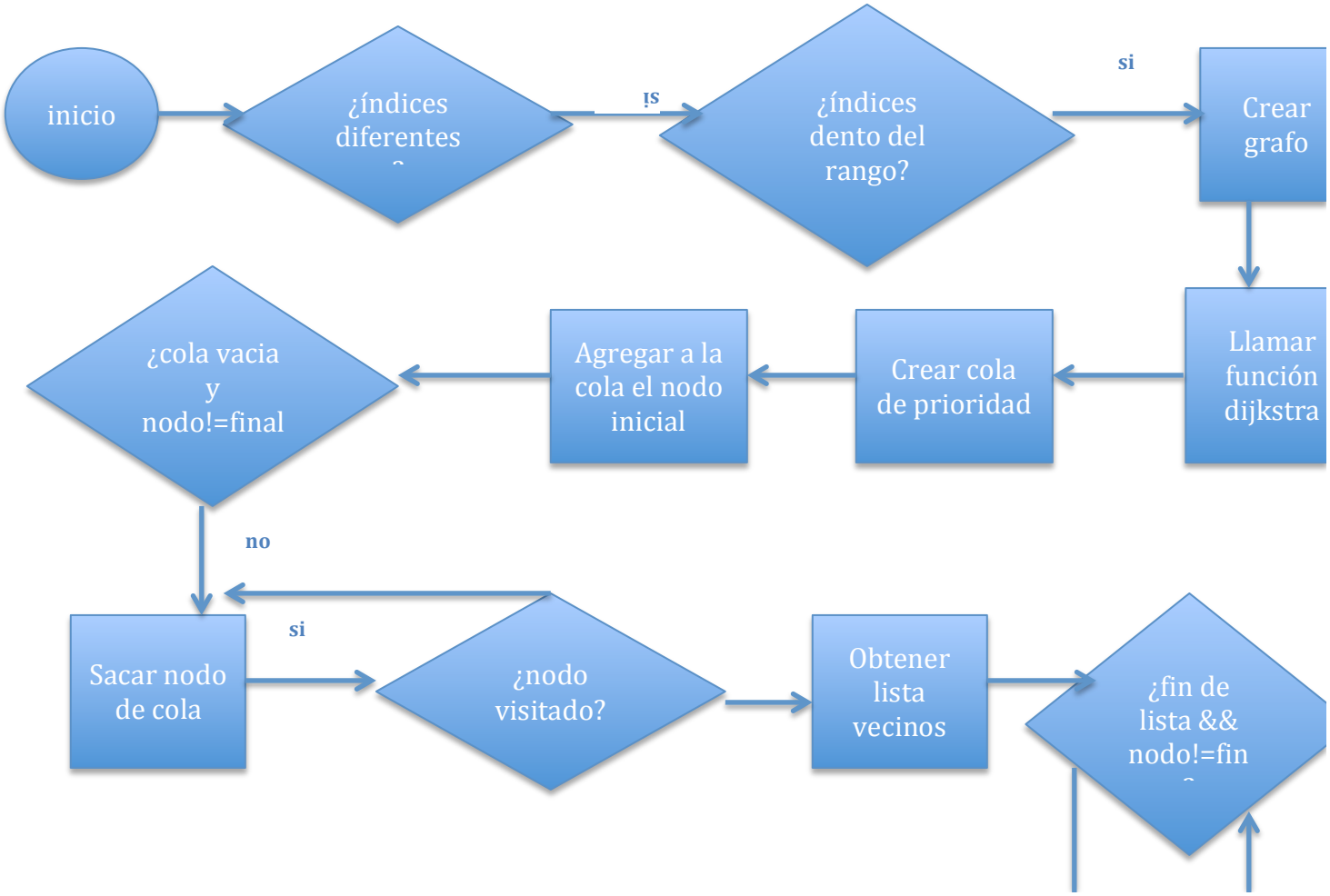


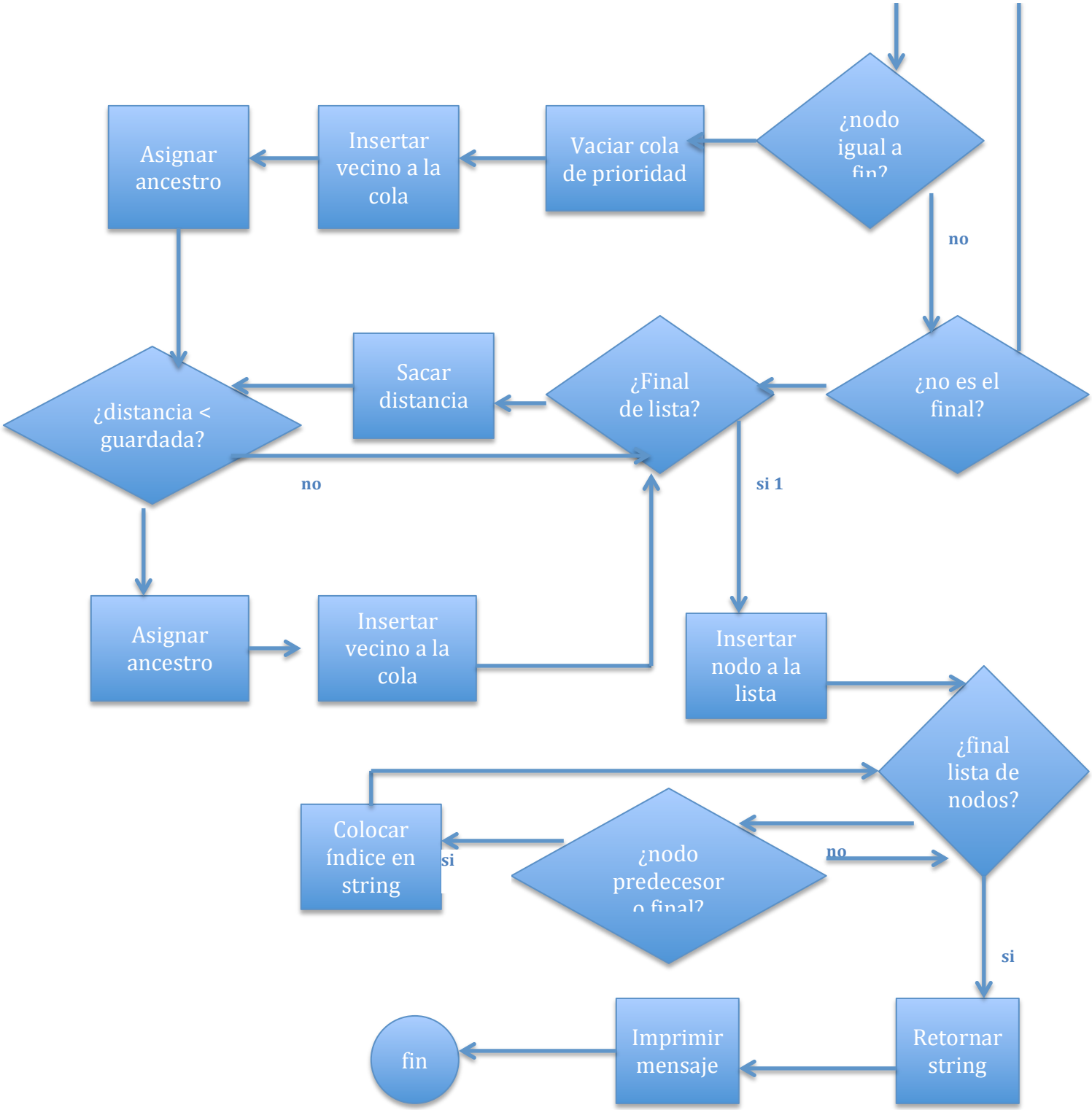
Comando ruta_corta:

1. Revisar que el objeto este cargado en memoria.
2. Revisar que los índices sean diferentes.
3. Revisar que los índices estén dentro del rango
4. Crear el grafo de manera temporal
5. Llamar función dijkstra
6. Crear cola de prioridad.
7. Agregar a la cola de prioridad el nodo con el índice de origen.
8. Revisar si la cola esta vacia y el nodo no sea el final
9. Sacar un nodo de la cola.
10. Revisar si el nodo ya fue visitado
11. Obtener la lista de vecinos del nodo

12. Iterar en la lista de vecinos, mientras no se llegue al final de la lista y el nodo vecino en el que se esta no sea el final
13. ¿El nodo vecino es el del índice fin?
    En caso de que sea
14. Vaciar cola de prioridad
15. Insertar el vecino a la cola de prioridad.
16. Asignar como ancestro a nodo
    En caso de que el vecino no sea el final
17. Iterar en la lista de vecinos hasta que el iterador no este al final de la lista
18. Sacar distancia hacia los vecinos.
19. Revisar que la distancia sacada sea menor a la ya guardada
20. Asignar ancestro a los vecinos
21. Insertar vecino en la cola de prioridad

22. Insertar nodo en la lista.
23. Iterar en la lista
24. Revisar que el nodo que esta en la lista sea un predecesor o el nodo final
    En caso de que sea un predecesor o el nodo final
25. Añadir el índice al string a retornar
26. Retornar el string.
27. Imprimir mensaje con longitud entre índices y el string con nodos.



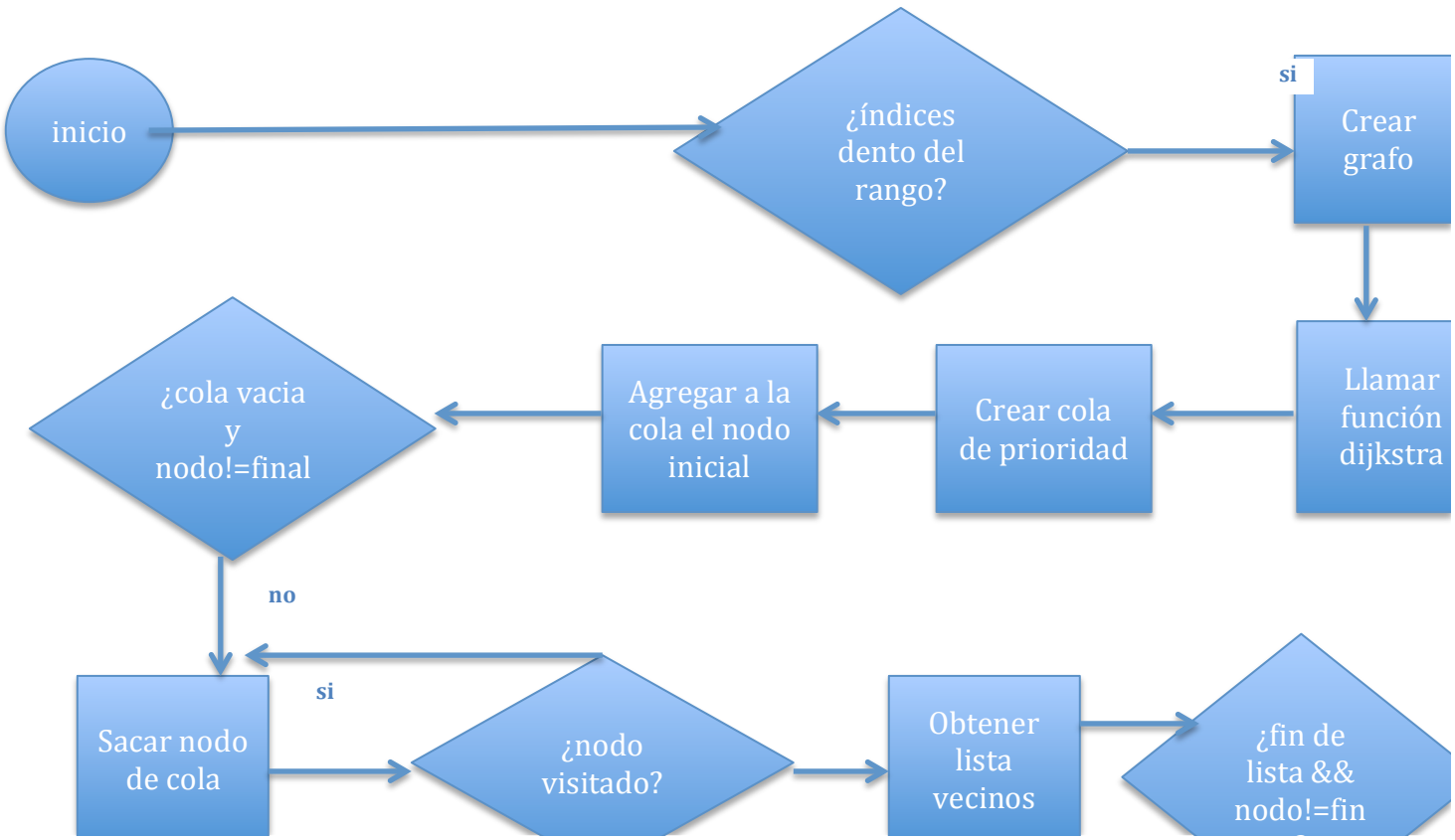


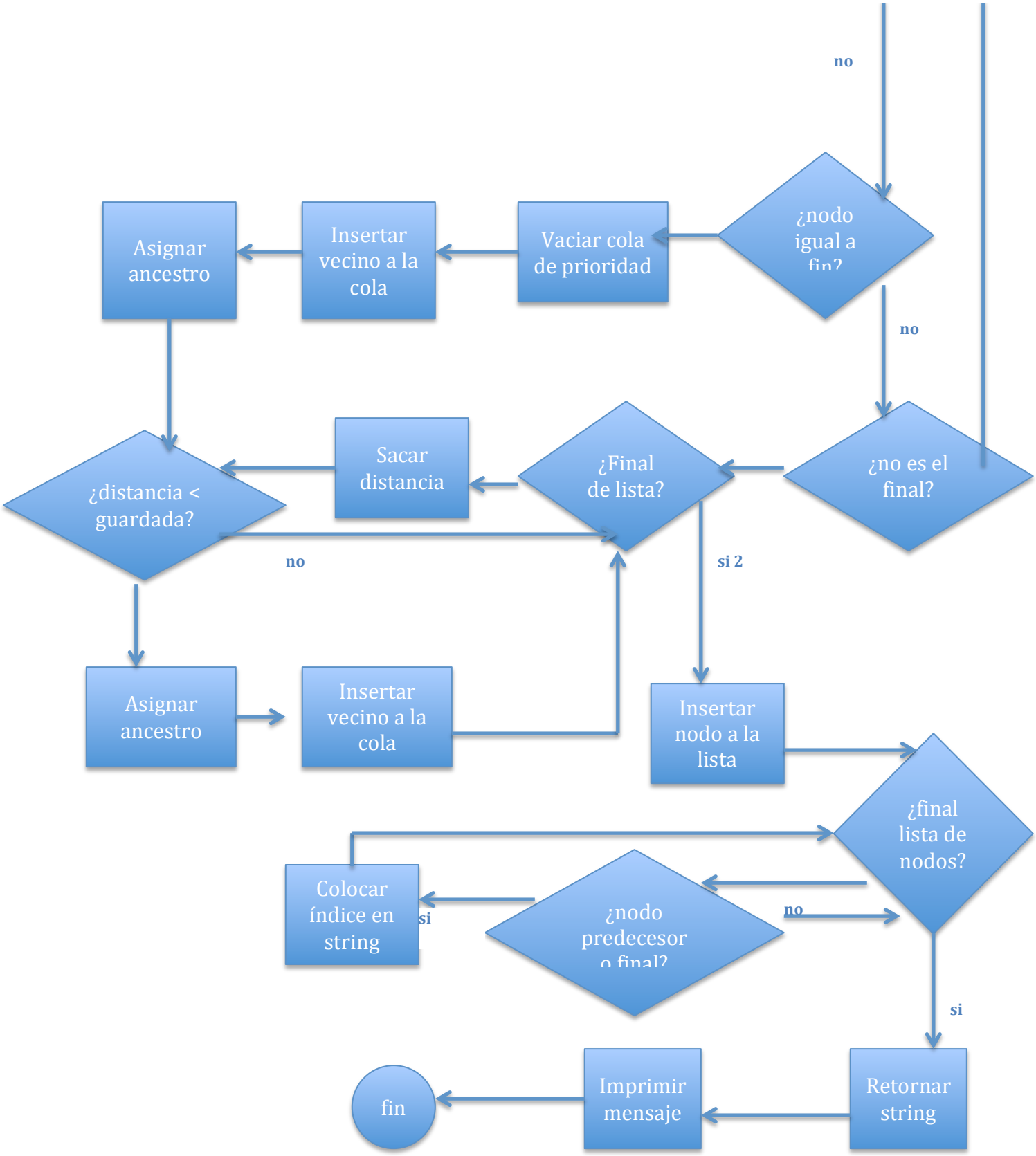
Comando ruta_corta_centra:

1. Revisar que el objeto este cargado en memoria.
2. Revisar que el índice este dentro del rango de índices del grafo.
3. Llamar función dijkstra
4. Crear cola de prioridad.
5. Agregar a la cola de prioridad el nodo con el índice de origen.
6. Revisar si la cola esta vacia y el nodo no sea el final

7. Sacar un nodo de la cola.
8. Revisar si el nodo ya fue visitado
9. Obtener la lista de vecinos del nodo
10. Iterar en la lista de vecinos, mientras no se llegue al final de la lista y el nodo vecino en el que se esta no sea el final
11. ¿El nodo vecino es el del índice fin?
    En caso de que sea
12. Vaciar cola de prioridad
13. Insertar el vecino a la cola de prioridad.
14. Asignar como ancestro a nodo
    En caso de que el vecino no sea el final
15. Iterar en la lista de vecinos hasta que el iterador no este al final de la lista
16. Sacar distancia hacia los vecinos.
17. Revisar que la distancia sacada sea menor a la ya guardada
18. Asignar ancestro a los vecinos
19. Insertar vecino en la cola de prioridad

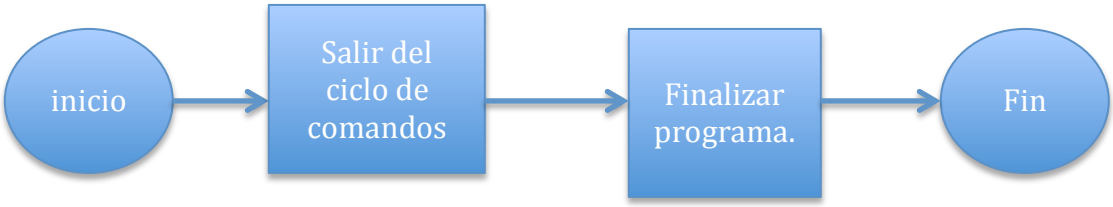
20. Insertar nodo en la lista.
21. Iterar en la lista
22. Revisar que el nodo que esta en la lista sea un predecesor o el nodo final
    En caso de que sea un predecesor o el nodo final
23. Añadir el índice al string a retornar
24. Retornar el string.
25. Imprimir mensaje con longitud entre índices y el string con nodos.





Comando salir:

1. Condición de salida del ciclo de comandos.
2. Finalizar programa.



Acta tercera entrega:
Se introduce el TAD grafo para realizar las operaciones de ruta_corta y ruta_corta_centro. El cual es usado para generar grafos temporales.
Comando v_cercano obj:

- En algunos casos, no se encuentra necesariamente el vértice más cercano, o puede ser que existan varios a la misma distancia.
- Explicación: La búsqueda del vértice más cercano selecciona el primer vértice seleccionado cuando existen varios a la misma distancia. Además, el resultado varia a diferencia de los demás debido que para corroborar que el punto si sea el más cercano se realiza en todos los sectores del octree.

Comando v_cercano:

- Mismo comentario que para el comando anterior.
- Explicación: Este se debe a que hace uso del comando anterior por lo cual genera un resultado similar.

Comando v_cercanos_caja:

- De nuevo, no necesariamente encuentra los vértices más cercanos a las esquinas de la caja.
- Explicación: Se debe a que hace uso del primer comando, por lo cual genera un resultado similar;