Proyecto 1-Entrega 2

ESTRUCTURAS DE DATOS

Estudiantes:

Laura Valentina Ovalle Benítez

Valentina García Alfonso

Juan Miguel Zuluaga Suárez

Carlos Gabriel López

Nicolás Padilla Medina

PRESENTADO A:

John Jairo Corredor Franco

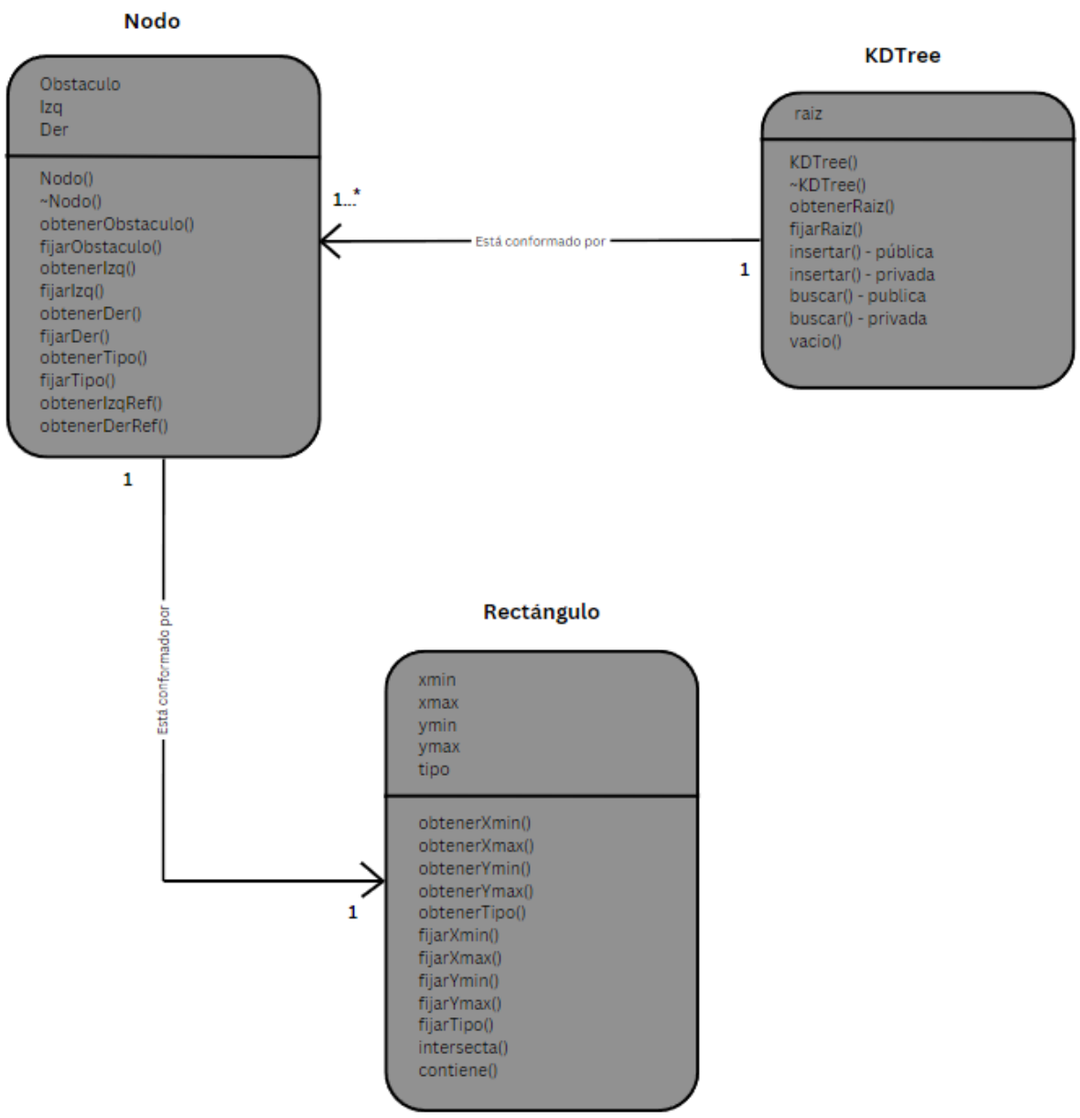
PONTIFICIA UNIVERSIDAD JAVERIANA

BOGOTÁ D.C

2023

Declaración de TAD's pertenecientes al segundo Componente

A continuación se presentan los TAD's que fueron añadidos al proyecto con el fin de cumplir con las funcionalidades propuestas en el "Componente 2":



Rectángulo

1. **Datos mínimos**
    a. xmin, entero, representa el valor mínimo que toma el rectángulo en el eje X del plano cartesiano
    b. xmax, entero, representa el valor máximo que toma el rectángulo en el eje X del plano cartesiano
    c. ymin, entero, representa el valor mínimo que toma el rectángulo en el eje Y del plano cartesiano
    d. ymax, entero, representa el valor máximo que toma el rectángulo en el eje Y del plano cartesiano
    e. tipo, cadena de caracteres, contiene el nombre del tipo de elemento a partir del cuál se va a crear el rectángulo, puede ser roca, cráter, montículo, duna o cuadrante
2. **Operaciones**
    a. fijarXmin(), fija un nueva coordenada mínima en el eje X para el rectángulo
    b. fijarXmax(), fija un nueva coordenada máxima en el eje X para el rectángulo
    c. fijarYmin(), fija un nueva coordenada mínima en el eje Y para el rectángulo
    d. fijarYmax(), fija un nueva coordenada máxima en el eje Y para el rectángulo
    e. fijarTipo(), fija un nuevo tipo a partir del cuál se crea el rectángulo
    f. obtenerXmin(), retorna la coordenada mínima actual del rectángulo en el eje X
    g. obtenerXmax(), retorna la coordenada máxima actual del rectángulo en el eje X

h. obtenerYmin(), retorna la coordenada mínima actual del rectángulo en el eje Y
i. obtenerYmax(), retorna la coordenada máxima actual del rectángulo en el eje Y
j. obtenerTipo(), retorna el tipo de elemento a partir del cuál se creó el rectángulo
k. intersecta(), evalúa si el rectángulo se intersecta con un rectángulo que recibe como parámetro. En caso afirmativo retorna True, de lo contrario retorna False.
l. contiene(), evalúa si el rectángulo se encuentra contenido dentro de un rectángulo que recibe como parámetro. En caso afirmativo retorna True, de lo contrario retorna False.

Nodo

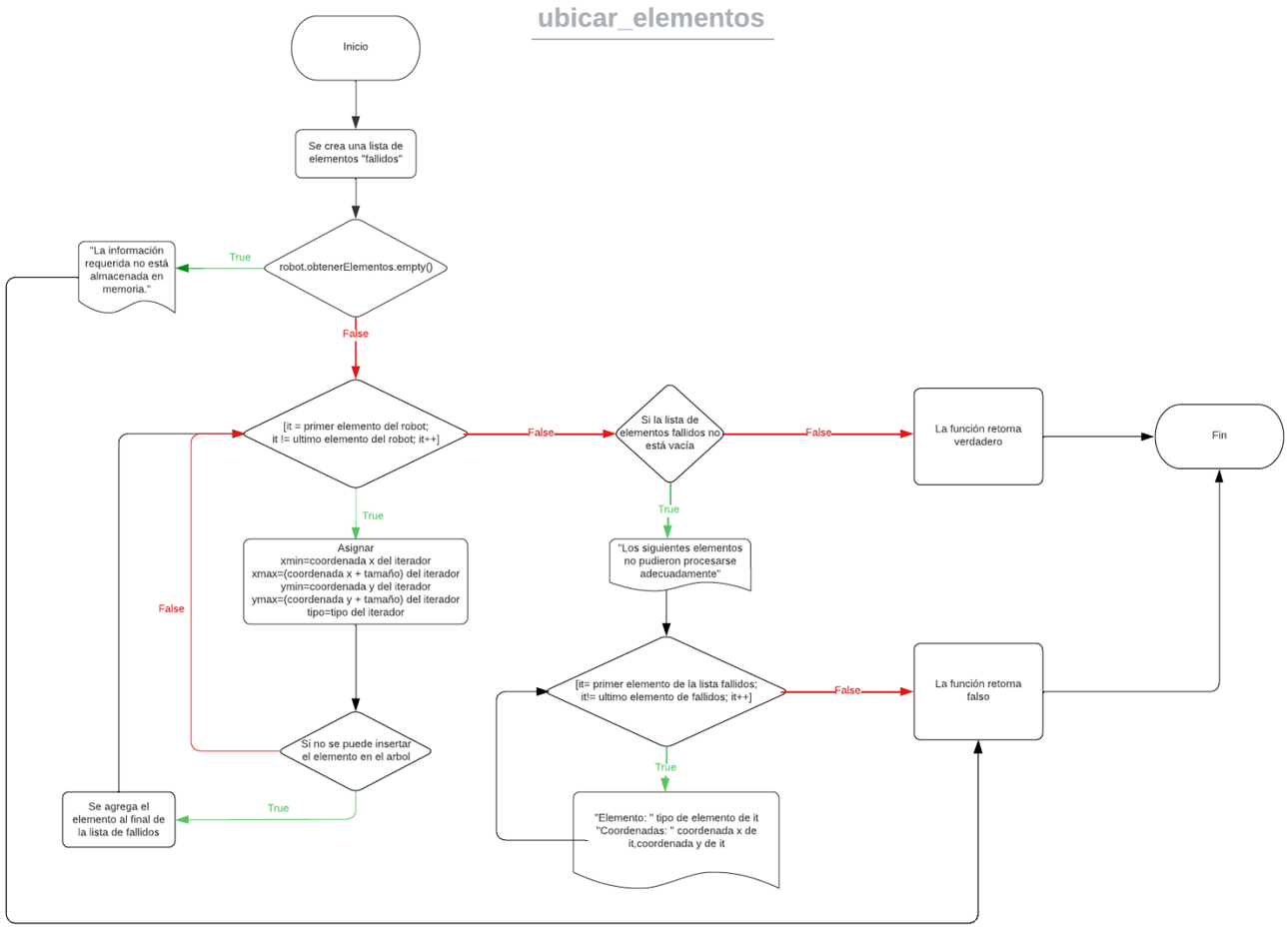
1. **Datos mínimos**
   a. Obstáculo, apuntador a un rectángulo, representa el elemento u obstáculo que se encuentra ubicado en el nodo
   b. Izq, apuntador a un nodo, representa el hijo izquierdo del nodo
   c. Der, apuntador a un nodo, representa el hijo derecho del nodo
2. **Operaciones**
   a. Nodo(), constructor, inicializa los valores del nodo asignando un apuntador a rectángulo que recibe como parámetro para el atributo “obstáculo”, y NULL para los atributos “izq” y “der”
   b. ~Nodo(), destructor, elimina los apuntadores “obstáculo”, “izq” y “der”, que corresponden a los atributos del nodo
   c. fijarObstaculo(), fija un nuevo valor para el elemento u obstáculo que se encuentra ubicado en el nodo
   d. fijarIzq(), fija un nuevo valor para el hijo izquierdo del nodo
   e. fijarDer(), fija un nuevo valor para el hijo derecho del nodo
   f. obtenerObstaculo(), retorna el apuntador que corresponde al elemento u obstáculo que se encuentra ubicado en el nodo actualmente
   g. obtenerIzq(), retorna el apuntador que corresponde al nodo hijo izquierdo del nodo actual
   h. ObtenerDer(), retorna el apuntador que corresponde al nodo hijo derecho del nodo actual
   i. ObtenerIzqRef(), retorna la referencia en memoria que corresponde al apuntador al nodo hijo izquierdo del nodo actual
   j. ObtenerDerRef(), retorna la referencia en memoria que corresponde al apuntador al nodo hijo derecho del nodo actual

KDTree

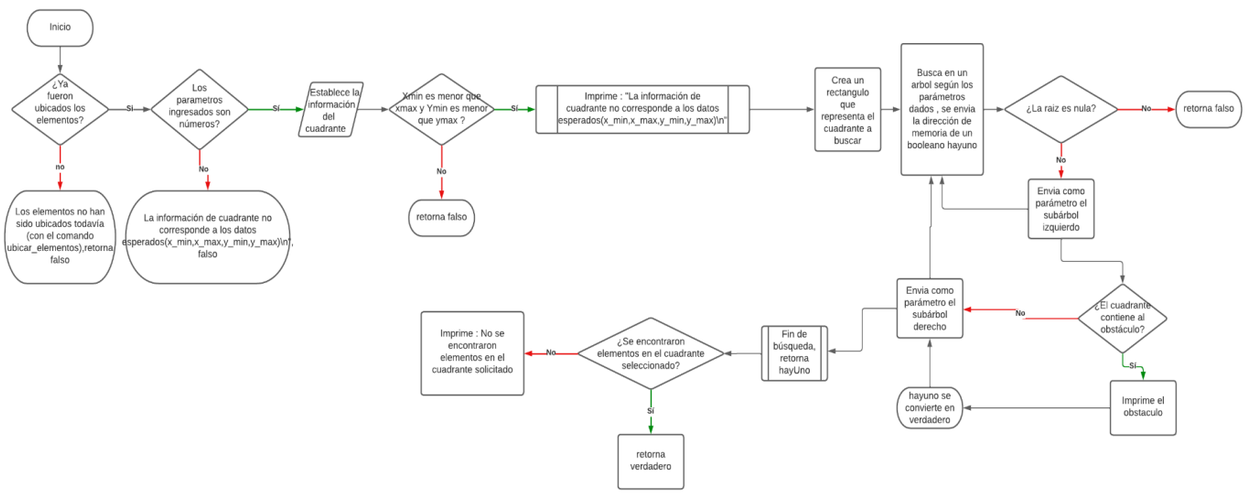
3. **Datos mínimos**
   a. Raíz, apuntador de tipo Nodo, representa el nodo a partir del cual se crear el árbol KDTree
4. **Operaciones**
   a. KDTree(), constructor, inicializa la raíz del árbol KDTree en NULL
   b. ~KDTree(), destructor, elimina el apuntador que corresponde a la raíz del árbol KDTree
   c. fijarRaiz(), fija un nuevo valor para la raíz del árbol KDTree
   d. obtenerRaiz(), retorna el apuntador a la raíz actual del árbol KDTree
   e. insertar(), función pública que utiliza la función privada “insertar” con el fin de agregar un nuevo nodo en el árbol
   f. insertar(), función privada que evalúa si es posible insertar un nuevo nodo en el árbol de acuerdo a sus coordenadas. En caso positivo inserta un nuevo nodo en el árbol KDTree y retorna True. En caso negativo retorna False.
   g. buscar(), función pública que utiliza la función privada “buscar” para recorrer el árbol KDTree y evaluar si un elemento dado de tipo rectángulo se encuentra intersectado con otro elemento dado de tipo rectángulo.
   h. buscar(), función privada que recorre el árbol KDTree y evalúa si un elemento dado de tipo rectángulo se encuentra intersectado con otro elemento dado de tipo rectángulo. En caso afirmativo este elemento se imprime.
   i. vacio(), evalúa si la raíz del árbol KDTree es nula, con el fin de determinar si el árbol está vacío

Diagramas de Flujo

1. Ubicar elementos:



2. En_cuadrante:



3. **Plan de pruebas:**

Con el fin de realizar el plan de pruebas de las nuevas funciones implementadas en el proyecto (“ubicar_elementos” y “en_cuadrante”), se cargarán 5 elementos del archivo de prueba ‘elementosp.txt’ para verificar el correcto funcionamiento del código. Cabe resaltar que los 5 elementos son insertados utilizando el siguiente formato:

Tipo de elemento|Tamaño|Distancia|Centro_x|Centro_y

A continuación se listan los elementos que se encuentran el archivo “elementosp.txt”

- crater|2|metros|2.0|2.0
- crater|2|metros|3.0|3.0
- monticulo|2|metros|6.0|8.0
- crater|5|metros|12.50|7.5
- duna|4|metros|18.00|14.00

A continuación se listan los cuadrantes que se utilizarán en conjunto con el comando “en_cuadrante”:.

1. en_cuadrante 0 100 0 100
2. en_cuadrante 6 18 2 14
3. en_cuadrante 0 5 0 4
4. en_cuadrante 50 60 50 60

Finalmente, se presenta el plan de pruebas para cada uno de los cuadrantes descritos anteriormente:

Plan de pruebas			
Descripción de caso	**Valores de entrada**	**Resultado esperado**	**Resultado obtenido**
1.Ingresar cuadrante	**xmim= 0** **xmax=100** **ymin=0** **ymax=100**	Elemento: Cráter Coordenadas: (2,2),(4,4) Elemento:crater Coordenadas: (12,7),(17,12) Elemento:monticulo Coordenadas: (6,8),(8,10) Elemento:duna Coordenadas: (18,14),(22,18)	Elemento: Cráter Coordenadas: (2,2),(4,4) Elemento:crater Coordenadas: (12,7),(17,12) Elemento:monticulo Coordenadas: (6,8),(8,10) Elemento:duna Coordenadas: (18,14),(22,18)

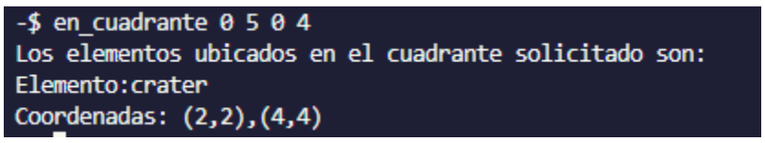
Como se puede apreciar en la siguiente imagen, el elemento cráter con coordenada (3,3) no se pudo guardar dado que se intersectaba con el elemento cráter con coordenada (2,2)

```
PS C:\Users\valeo\Downloads\EntregaCuriosity\ProyectoEntrega2> g++ -std=c++11 -o prog *.cpp
PS C:\Users\valeo\Downloads\EntregaCuriosity\ProyectoEntrega2> ./prog
-$ cargar_elementos elementosp.txt
Leyendo archivo 'elementosp.txt':
5 elementos cargados correctamente desde: elementosp.txt
-$ ubicar_elementos
Los siguientes elementos no pudieron procesarse adecuadamente:
Elemento: crater Coordenadas: (3,3)
-$ en_cuadrante 0 100 0 100
Los elementos ubicados en el cuadrante solicitado son:
Elemento:crater
Coordenadas: (2,2),(4,4)
Elemento:crater
Coordenadas: (12,7),(17,12)
Elemento:monticulo
Coordenadas: (6,8),(8,10)
Elemento:duna
Coordenadas: (18,14),(22,18)
```

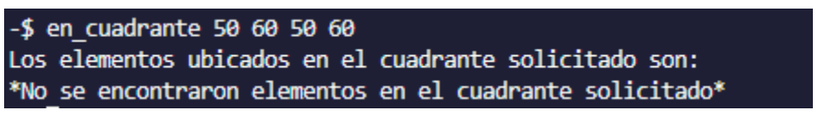
Plan de pruebas			
Descripción de caso	**Valores de entrada**	**Resultado esperado**	**Resultado obtenido**
1.Ingresar cuadrante	**xmim= 6** **xmax=18** **ymin=2** **ymax=14**	Elemento: Cráter Coordenadas: (12,7),(17,12) Elemento:monticulo Coordenadas: (6,8),(8,10) Elemento:duna Coordenadas: (18,14),(22,18)	Elemento:crater Coordenadas: (12,7),(17,12) Elemento:monticulo Coordenadas: (6,8),(8,10) Elemento:duna Coordenadas: (18,14),(22,18)

```
-$ en_cuadrante 6 18 2 14
Los elementos ubicados en el cuadrante solicitado son:
Elemento:crater
Coordenadas: (12,7),(17,12)
Elemento:monticulo
Coordenadas: (6,8),(8,10)
Elemento:duna
Coordenadas: (18,14),(22,18)
```

Plan de pruebas			
Descripción de caso	**Valores de entrada**	**Resultado esperado**	**Resultado obtenido**
1.Ingresar cuadrante	**xmim= 0** **xmax=5** **ymin=0** **ymax=4**	Elemento: Cráter Coordenadas: (2,2),(4,4)	Elemento: Cráter Coordenadas: (2,2),(4,4)



Plan de pruebas			
Descripción de caso	**Valores de entrada**	**Resultado esperado**	**Resultado obtenido**
1.Ingresar cuadrante	**xmim= 50** **xmax=60** **ymin=50** **ymax=60**	*No se encontraron elementos en el cuadrante solicitado	*No se encontraron elementos en el cuadrante solicitado



A continuación se adjunta una imagen de los elementos que fueron ingresados al programa, la cual sirvió de apoyo visual para así obtener una mejor visión de la representación de los obstáculos en el plano.