Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Estructuras de Datos
Proyecto del curso, 2023-10

1 Descripción del problema

El vehículo "Curiosity", desarrollado por la NASA, es un robot explorador que lleva a cabo diversos experimentos científicos. Su misión principal es buscar condiciones pasadas o presentes favorables para la vida y condiciones capaces de conservar registros de vida en áreas específicas del planeta Marte[1]. Para cumplir con su misión, el explorador debe desplazarse por el entorno marciano (caracterizado por arena, rocas y montículos) para recolectar fotografías y muestras del entorno. El desplazamiento del vehículo explorador se hace mediante secuencias de comandos enviadas desde la Tierra la noche anterior al día en que se desea sean ejecutadas[2]. Dependiendo de la irregularidad del terreno, el explorador tiene dos maneras diferentes de desplazarse hacia un punto geográfico:

1. Desplazarse según los comandos enviados la noche anterior.
2. Desplazarse de manera autónoma por el terreno hasta una ubicación dada. Para diseñar su camino, el robot explorador se basa en mapas de profundidad generados gracias a dos cámaras ubicadas al frente del vehículo.

El desplazamiento del "Curiosity" es una tarea crítica en el suelo marciano, ya que una mala planeación y ejecución de la trayectoria puede terminar en un estancamiento del vehículo, poniendo fin así a sus operaciones de exploración. Para validar la correcta ejecución de los comandos de desplazamiento enviados desde la Tierra, el vehículo explorador tiene definido un mecanismo de retroalimentación y validación, utilizando fotografías tomadas por medio de las cámaras instaladas en el frente del vehículo y el análisis de los componentes del suelo marciano.

2 Descripción del proyecto

El objetivo del presente proyecto es construir un sistema que permita simular las actividades e interacción entre el vehículo "Curiosity" y su centro de control de misión ubicado en las instalaciones de la NASA.

2.1 Comunicación con el "Curiosity"

La información que se envía al robot explorador para realizar su misión incluye dos tipos de datos: comandos de desplazamiento, que le permiten moverse sobre la superficie y realizar algunos análisis del suelo marciano; y datos de puntos de interés, que incluyen las ubicaciones geográficas de componentes o elementos hallados en el terreno.

2.1.1 Comandos de desplazamiento

Para indicar al vehículo las actividades que puede realizar sobre el suelo marciano, los comandos de desplazamiento y análisis se agrupan en un único archivo de ejecución de comandos, que será el que se enviará al robot la noche anterior. Los comandos son de dos tipos: movimiento y análisis.

- Comandos de movimiento: permiten al robot desplazarse sobre la superficie de Marte. Estos comandos tienen la siguiente estructura:
  `tipo_movimiento magnitud unidad_medida`
  Donde `tipo_movimiento` puede ser `avanzar` o `girar`, `magnitud` es el valor o cantidad del movimiento y `unidad_medida` es la unidad con la que se hace la medición del movimiento, de acuerdo a la siguiente tabla:

Tipo de movimiento	Unidades de medida aceptadas
`avanzar`	`cm`, `dm`, `m`, `km`
`girar`	`grd`, `rad`

[1] `https://mars.nasa.gov/msl/mission/rover/`
[2] `https://www.youtube.com/watch?v=_hN4XdS7NMY`

- Comandos de análisis: permiten al robot investigar la superficie de Marte para analizar los elementos que va encontrando en su desplazamiento. Estos comandos tienen la siguiente estructura:
  `tipo_analisis objeto comentario`
  Donde `tipo_analisis` puede ser `fotografiar`, `composicion` o `perforar`; `objeto` es el nombre del elemento sobre el cual se hace el análisis y `comentario` es un valor opcional que permite agregar información sobre el análisis a realizar o el elemento que se analizará, este comentario debe ingresarse entre comillas simples.

Un ejemplo de archivo de ejecución de comandos del "Curiosity" se presenta a continuación:

```
avanzar 40 cm
girar 35 grd
avanzar 35 cm
perforar roca
avanzar 26 cm
girar 25 grd
fotografiar colinas 'sistema de montañas'
avanzar 0.75 m
composicion suelo 'polvo'
girar 30 grd
avanzar 1.2 m
```

En este ejemplo se puede evidenciar que los comandos de movimiento y análisis están mezclados, lo que permite que el robot pueda desplazarse adecuadamente y hacer los análisis necesarios sobre el suelo marciano.

2.1.2 Información de puntos de interés

Para identificar la complejidad del terreno sobre el cual debe moverse el vehículo, se almacena de forma adicional la información de elementos o componentes previamente hallados en el terreno. Esta información tiene la siguente estructura por cada elemento o componente:
`tipo_elemento tamaño unidad_medida coordenada_x coordenada_y`
Donde `tipo_elemento` puede ser `roca`, `crater`, `monticulo` o `duna`; `tamaño` es el valor de la dimensión del elemento, `unidad_medida` es la unidad con la que se realizó la medición del tamaño del elemento (las unidades de medida aceptadas son las mismas que las del comando `avanzar`), `coordenada_x` es la posición sobre el eje x en el plano cartesiano del elemento, y `coordenada_y` es la posición sobre el eje y en el plano cartesiano del elemento. Las coordenadas de posicionamiento de los elementos se encuentran en un sistema de referencia basado en metros como unidad básica de medida.

Un ejemplo de archivo de información de elementos de interés para el "Curiosity" se presenta a continuación:

```
roca 13 cm 3.45 15.4
duna 0.5 km 24.345 67.456
crater 0.55 m 15.8 34.923
```

2.2 Componentes del sistema

A continuación se describen los componentes individuales que conforman el presente proyecto:

2.2.1 Componente 1: Manejo de información

Objetivo: Realizar la administración de los comandos de desplazamiento del vehículo y de la información de puntos de interés encontrados en el suelo marciano. Este componente se implementará utilizando las estructuras lineales que se consideren adecuadas, y contiene los siguientes comandos:

- **comando:** `cargar_comandos` *`nombre_archivo`*
  salida en pantalla:
  (Archivo vacío) *`nombre_archivo`* `no contiene comandos.`
  (Archivo erróneo) *`nombre_archivo`* `no se encuentra o no puede leerse.`
  (Resultado exitoso) *`n`* `comandos cargados correctamente desde` *`nombre_archivo`*`.`
  descripción: Carga en memoria los comandos de desplazamiento contenidos en el archivo identificado por *`nombre_archivo`*, es decir, utiliza adecuadamente las estructuras lineales para cargar la información de los comandos en memoria. Si dentro de la misma sesión de trabajo ya se han cargado otros archivos de comandos (usando el comando `cargar_comandos`), la información debe sobreescribirse en memoria, es decir, no se deben combinar informaciones de comandos de diferentes archivos.

- **comando:** `cargar_elementos` *nombre_archivo*
  salida en pantalla:
  (Archivo vacío) *nombre_archivo* `no contiene elementos.`
  (Archivo erróneo) *nombre_archivo* `no se encuentra o no puede leerse.`
  (Resultado exitoso) *n* `elementos cargados correctamente desde` *nombre_archivo* `.`
  descripción: Carga en memoria los datos de puntos de interés o elementos contenidos en el archivo identificado por *nombre_archivo*, es decir, utiliza adecuadamente las estructuras lineales para cargar la información de los elementos en memoria. Si dentro de la misma sesión de trabajo ya se han cargado otros archivos de puntos de interés (usando el comando `cargar_elementos`), la información debe sobreescribirse en memoria, es decir, no se deben combinar informaciones de elementos de diferentes archivos.

- **comando:** `agregar_movimiento` *tipo_mov magnitud unidad_med*
  salida en pantalla:
  (Formato erróneo) `La información del movimiento no corresponde a los datos esperados (tipo, magnitud, unidad).`
  (Resultado exitoso) `El comando de movimiento ha sido agregado exitosamente.`
  descripción: Agrega el comando de movimiento descrito a la lista de comandos del robot "Curiosity". El movimiento puede ser de dos tipos: `avanzar` o `girar`. La magnitud corresponde al valor del movimiento; si es avanzar, el número de unidades que se espera avanzar, si es girar la cantidad de grados que debe girar. La unidad de medida corresponde a la convención con la que se mide la magnitud del movimiento, de acuerdo a la tabla presentada anteriormente. Si no se envía la información completa y adecuada del comando de movimiento, éste no puede agregarse a la lista de los comandos que se enviarán al robot desde la tierra.

- **comando:** `agregar_analisis` *tipo_analisis objeto comentario*
  salida en pantalla:
  (Formato erróneo) `La información del análisis no corresponde a los datos esperados (tipo, objeto, comentario).`
  (Resultado exitoso) `El comando de análisis ha sido agregado exitosamente.`
  descripción: Agrega el comando de análisis descrito a la lista de comandos del robot "Curiosity". El análisis puede ser de tres tipos: `fotografiar`, `composicion` o `perforar`. El objeto es el nombre del elemento que se quiere analizar (roca, arena, monticulo, ...). El comentario es opcional y permite agregar más información sobre el elemento o el análisis, este comentario estará encerrado entre comillas simples (ejemplo: `'roca_cuadrante_32'`). Si no se envía la información completa y adecuada del comando de análisis, éste no puede agregarse a la lista de los comandos que se enviarán al robot desde la tierra.

- **comando:** `agregar_elemento` *tipo_comp tamaño unidad_med coordX coordY*
  salida en pantalla:
  (Formato erróneo) `La información del elemento no corresponde a los datos esperados (tipo, tamaño, unidad, x, y).`
  (Resultado exitoso) `El elemento ha sido agregado exitosamente.`
  descripción: Agrega el componente o elemento descrito a la lista de puntos de interés del robot "Curiosity". El tipo de componente puede ser uno entre `roca`, `crater`, `monticulo` o `duna`. El tamaño es un valor real que da cuenta de qué tan grande es el elemento; y la unidad de medida complementa este valor con la convención que se usó para su medición, de acuerdo a la tabla presentada anteriormente. Finalmente, las coordenadas `x` y `y` corresponden a números reales que permiten conocer la ubicación del elemento en el sistema de coordenadas del suelo marciano utilizado por el vehículo. Si no se envía la información completa y adecuada del elemento, éste no puede agregarse a la lista de puntos de interés que se enviarán al robot desde la tierra.

- **comando:** `guardar` *tipo_archivo nombre_archivo*
  salida en pantalla:
  (No hay información) `La información requerida no está almacenada en memoria.`
  (Escritura exitosa) `La información ha sido guardada en` *nombre_archivo* `.`
  (Problemas en archivo) `Error guardando en` *nombre_archivo* `.`
  descripción: Guarda en el archivo *nombre_archivo* la información solicitada de acuerdo al tipo de archivo: `comandos` almacena en el archivo la información de comandos de movimiento y de análisis que debe ejecutar el robot, `elementos` almacena en el archivo la información de los componentes o puntos de interés conocidos en el suelo marciano.

- **comando:** `simular_comandos` *coordX coordY*
  salida en pantalla:
  (No hay información) `La información requerida no está almacenada en memoria.`
  (Resultado exitoso) `La simulación de los comandos, a partir de la posición (`*coordX*`,`*coordY*`), deja al robot en la nueva posición (`*nuevoX*`,`*nuevoY*`).`
  descripción: Permite simular el resultado de los comandos de movimiento que se enviarán al robot "Curiosity" desde la Tierra, facilitando asi la validación de la nueva posición en la que podría ubicarse. Para ejecutarse adecuadamente, requiere conocer la posición actual (coordenadas x y y) del vehículo. A partir de la posición actual, se asume que el "Curiosity" está orientado mirando hacia la parte derecha del eje x en un sistema cartesiano (hacia la derecha). Los ángulos de giro positivos generan movimiento en el sentido contrario de las manecillas del reloj, mientras que los ángulos de giro negativos generan movimiento en el sentido de las manecillas del reloj. Hay que tener en cuenta que sólo los comandos de movimiento son necesarios, no los de análisis.

- **comando:** `salir`
  salida en pantalla:
  (No tiene salida por pantalla)
  descripción: Termina la ejecución de la aplicación.

2.2.2 Componente 2: Planeación de desplazamientos

Objetivo: Utilizar una estructura de datos jerárquica que permita almacenar datos geométricos para analizar los puntos de interés sobre el cielo marciano y así facilitar en el futuro la planificación automática de desplazamientos. Los comandos que se deben implementar como parte de este componente son:

- **comando:** `ubicar_elementos`
  salida en pantalla:
  (No hay información) `La información requerida no está almacenada en memoria.`
  (Problemas con elementos) `Los siguientes elementos no pudieron procesarse adecuadamente: ...`
  (Resultado exitoso) `Los elementos han sido procesados exitosamente.`
  descripción: El comando debe utilizar la información de puntos de interés almacenada en memoria para ubicarlos en una estructura de datos jerárquica adecuada, que permita luego realizar consultas geográficas sobre estos elementos. Si alguno de los elementos no puede agregarse adecuadamente, debe generarse un mensaje de error, pero deben terminarse de procesar todos los elementos almacenados en memoria.

- **comando:** `en_cuadrante` *coordX1 coordX2 coordY1 coordY2*
  salida en pantalla:
  (Formato erróneo) `La información del cuadrante no corresponde a los datos esperados (x_min, x_max, y_min, y_max).`
  (No hay información) `Los elementos no han sido ubicados todavía (con el comando ubicar_elementos).`
  (Resultado exitoso) `Los elementos ubicados en el cuadrante solicitado son: ...`
  descripción: Permite utilizar la estructura creada con el comando anterior para retornar una lista de los componentes o elementos que están dentro del cuadrante geográfico descrito por los límites de coordenadas en x y y. Es necesario haber ejecutado el comando `ubicar_elementos` para poder realizar la búsqueda por cuadrantes. Los límites de coordenadas deben garantizar que *coordX1* $<$ *coordX2* y *coordY1* $<$ *coordY2*.

2.2.3 Componente 3: Recorridos entre puntos de interés

Objetivo: A partir de la información de puntos de interés, utilizar representaciones en grafos para crear representaciones geográficas (mapas) que permitan posteriormente identificar regiones de interés sobre el suelo marciano para aterrizajes y exploración. Los comandos que se deben desarrollar como parte de este componente son:

- **comando:** `crear_mapa` *coeficiente_conectividad*
  salida en pantalla:
  (No hay información) `La información requerida no está almacenada en memoria.`
  (Resultado exitoso) `El mapa se ha generado exitosamente. Cada elemento tiene` n `vecinos.`
  descripción: El comando debe utilizar la información de puntos de interés almacenada en memoria para

ubicarlos en una estructura no lineal y conectarlos entre sí teniendo en cuenta el coeficiente de conectividad dado. El objetivo es que cada elemento esté conectado a los demás elementos más cercanos a él, midiendo la cercanía a través de la distancia euclidiana entre los elementos. Esta distancia euclidiana también se utiliza como el peso o etiqueta de la conexión entre los elementos. Con el coeficiente de conectividad se identifica la cantidad de vecinos que puede tener cada elemento tomando como base el total de elementos que se ubicarán en el mapa (ejemplo: si se van a ubicar 35 elementos, y el coeficiente de conectividad es 0.4, la cantidad de vecinos que cada elemento debe tener es 35 * 0.4 = 14).

- **comando:** `ruta_mas_larga`
  salida en pantalla:
  (No hay información) `El mapa no ha sido generado todavía (con el comando crear_mapa).`
  (Resultado exitoso) `Los puntos de interés más alejados entre sí son ... y ... La ruta que los conecta tiene una longitud total de ... y pasa por los siguentes elementos: ...`
  descripción: Con el mapa ya creado, el comando permite identificar los dos componentes más alejados entre sí de acuerdo a las conexiones generadas. Es importante aclarar que el comando retorna los elementos más alejados de acuerdo a las conexiones que se encuentran en el mapa, no los elementos que estén a mayor distancia euclidiana entre sí. Al encontrar esa ruta más larga dentro del mapa, el comando imprime en pantalla los elementos de origen y destino, la longitud total de la ruta, y la secuencia de elementos que hay que seguir para ir del elemento origen al elemento destino.

2.3 Interacción con el sistema

La interfaz del sistema a construir debe ser una consola interactiva donde los comandos correspondientes a los componentes serán tecleados por el usuario, de la misma forma que se trabaja en la terminal o consola del sistema operativo. El indicador de línea de comando debe ser el caracter `$`. Se debe incluir el comando `ayuda` para indicar una lista de los comandos disponibles en el momento. Así mismo, para cada comando se debe incluir una ayuda de uso que indique la forma correcta de hacer el llamado, es decir, el comando `ayuda comando` debe existir.

Cada comando debe presentar en pantalla mensajes de éxito o error que permitan al usuario saber, por un lado, cuando terminó el comando su procesamiento, y por el otro lado, el resultado de ese procesamiento. Los comandos de los diferentes componentes deben ensamblarse en un único sistema (es decir, funcionan todos dentro de un mismo programa, no como programas independientes por componentes).

3 Evaluación

Las entregas se harán en la correspondiente asignación de BrightSpace, hasta la media noche del día anterior al indicado para la sustentación de la entrega. Se debe entregar un archivo comprimido (único formato aceptado: .zip) que contenga dentro de un mismo directorio (sin estructura de carpetas interna) los documentos (único formato aceptado: .pdf) y el código fuente (.h, .hxx, .cxx, .cpp). Si la entrega contiene archivos en cualquier otro formato, será descartada y no será evaluada, es decir, la nota definitiva de la entrega será de 0 (cero) sobre 5 (cinco).

3.1 (10%) Entrega 0: semana 3

La entrega inicial corresponderá únicamente a la interfaz de usuario necesaria para interactuar con el sistema. De esta forma, se verificará el indicador de línea del comando, y que el sistema realice la validación de los comandos permitidos y sus parámetros (Revisar en particular el numeral 2.3).

3.2 (30%) Entrega 1: semana 6

Componente 1 completo y funcional. Esta entrega tendrá una sustentación durante la correspondiente sesión de clase, y se compone de:

- (30%) Documento de diseño. El documento de diseño debe seguir las pautas de ingeniería que usted ya conoce: descripción de entradas, salidas y condiciones para el procedimiento principal y las operaciones auxiliares (comandos). Para la descripción de los TADs utilizados, debe seguirse la plantilla definida en clase. Además, se exigirán esquemáticos (diagramas, gráficos, dibujos) que describan el funcionamiento general de las operaciones (comandos) principales.

- (5%) Plan de pruebas. Adjuntar al documento de diseño un plan de pruebas, que siga las pautas vistas en clase, para el comando `simular_comandos`.
- (55%) Código fuente compilable en el compilador gnu-g++ v4.0.0 (mínimo). Este porcentaje de la entrega será un promedio de la evaluación de cada comando.
- (10%) Sustentación con participación de todos los miembros del grupo.

3.3 (30%) Entrega 2: semana 12

Componentes 1 y 2 completos y funcionales. Esta entrega tendrá una sustentación durante la correspondiente sesión de clase, y se compone de:

- (15%) Completar la funcionalidad que aún no haya sido desarrollada de la primera entrega. Se debe generar un acta de evaluación de la entrega anterior (incluirla al principio del documento de diseño) que detalle los comentarios textuales (literales) hechos a la entrega y la forma en la que se corrigieron, arreglaron o completaron para la segunda entrega.
- (25%) Documento de diseño. El documento de diseño debe seguir las pautas de ingeniería que usted ya conoce: descripción de entradas, salidas y condiciones para el procedimiento principal y las operaciones auxiliares. Para la descripción de los TADs utilizados, debe seguirse la plantilla definida en clase. Además, se exigirán esquemáticos (diagramas, gráficos, dibujos) que describan el funcionamiento general de las operaciones (comandos) principales.
- (5%) Plan de pruebas. Adjuntar al documento de diseño un plan de pruebas, que siga las pautas vistas en clase, para el comando `en_cuadrante`.
- (45%) Código fuente compilable en el compilador gnu-g++ v4.0.0 (mínimo). Este porcentaje de la entrega será un promedio de la evaluación de cada comando.
- (10%) Sustentación con participación de todos los miembros del grupo.

3.4 (30%) Entrega 3: semana 18

Componentes 1, 2 y 3 completos y funcionales. Esta entrega tendrá una sustentación (del proyecto completo) entre las 8am y las 12m del último día de clase de la semana 18, y se compone de:

- (15%) Completar la funcionalidad que aún no haya sido desarrollada de la primera y segunda entregas. Se debe generar un acta de evaluación de la entrega anterior (incluirla al principio del documento de diseño) que detalle los comentrios textuales (literales) hechos a la entrega y la forma en la que se corrigieron, arreglaron o completaron para la tercera entrega.
- (25%) Documento de diseño. El documento de diseño debe seguir las pautas de ingeniería que usted ya conoce: descripción de entradas, salidas y condiciones para el procedimiento principal y las operaciones auxiliares. Para la descripción de los TADs utilizados, debe seguirse la plantilla definida en clase. Además, se exigirán esquemáticos (diagramas, gráficos, dibujos) que describan el funcionamiento general de las operaciones (comandos) principales.
- (5%) Plan de pruebas. Adjuntar al documento de diseño un plan de pruebas, que siga las pautas vistas en clase, para el comando `ruta_mas_larga`.
- (45%) Código fuente compilable en el compilador gnu-g++ v4.0.0 (mínimo). Este porcentaje de la entrega será un promedio de la evaluación de cada comando.
- (10%) Sustentación con participación de todos los miembros del grupo.