Algoritmos y Estructuras de Datos II

Documentación del Primer Proyecto Programado

Profesor: Antonio Gonzalez Torres

Integrantes:

Mauro Navarro Obando

Gabriel Vargas López

Fecha: 28 de abril 2021

Índice

Descripción del problema

El objetivo del proyecto es crear un IDE para el pseudo lenguaje C!. Este contiene las siguientes secciones:

- Editor de texto: Región del IDE donde se ingresa el código. Se ejecuta al presionar el botón Run. El usuario puede detener la ejecución o avanzar línea por línea.
- Stdout: Sección bajo el editor que cumple la función de standart output. Imprime el resultado de print.
- Application Log: Muestra cualquier error interno o llamada al servidor de memoria.
- RAM Live View: Aquí se logra observar el estado de la RAM en cada línea respectiva del programa cuando se ejecuta. Muestra la dirección de memora, valor, etiquera y conteo de referencias de cada línea.

El subconjunto del lenguaje maneja los tipos de datos int, long, char, float, double struct y reference. El tipo struct creacopias de las referencias y reference crea una copia de las direcciones. Ninguno de estos copia el dato.

Todos los tipos excepto el reference soportan la operación getAddr que retorna un tipo reference. Los reference soportan el operador getValue que retorna el valor apuntado por la referencia. También soporta la definición de un scope mediante {}, que permite definir la visibilidad de las variables.

El manejo de memorita se implementa a traves de un servidor en donde se realiza un malloc de la cantidad de memoria que se desea utilizar. Este servidor escucha peticiones del IDE de C! en formato JSON. La memoria en el servidor es manejada automáticamente, además de solicitar el conteo de referecias para eliminar espacios no referenciados con el garbage collector.

Diagrama de clases

Diagrama de clases inicial

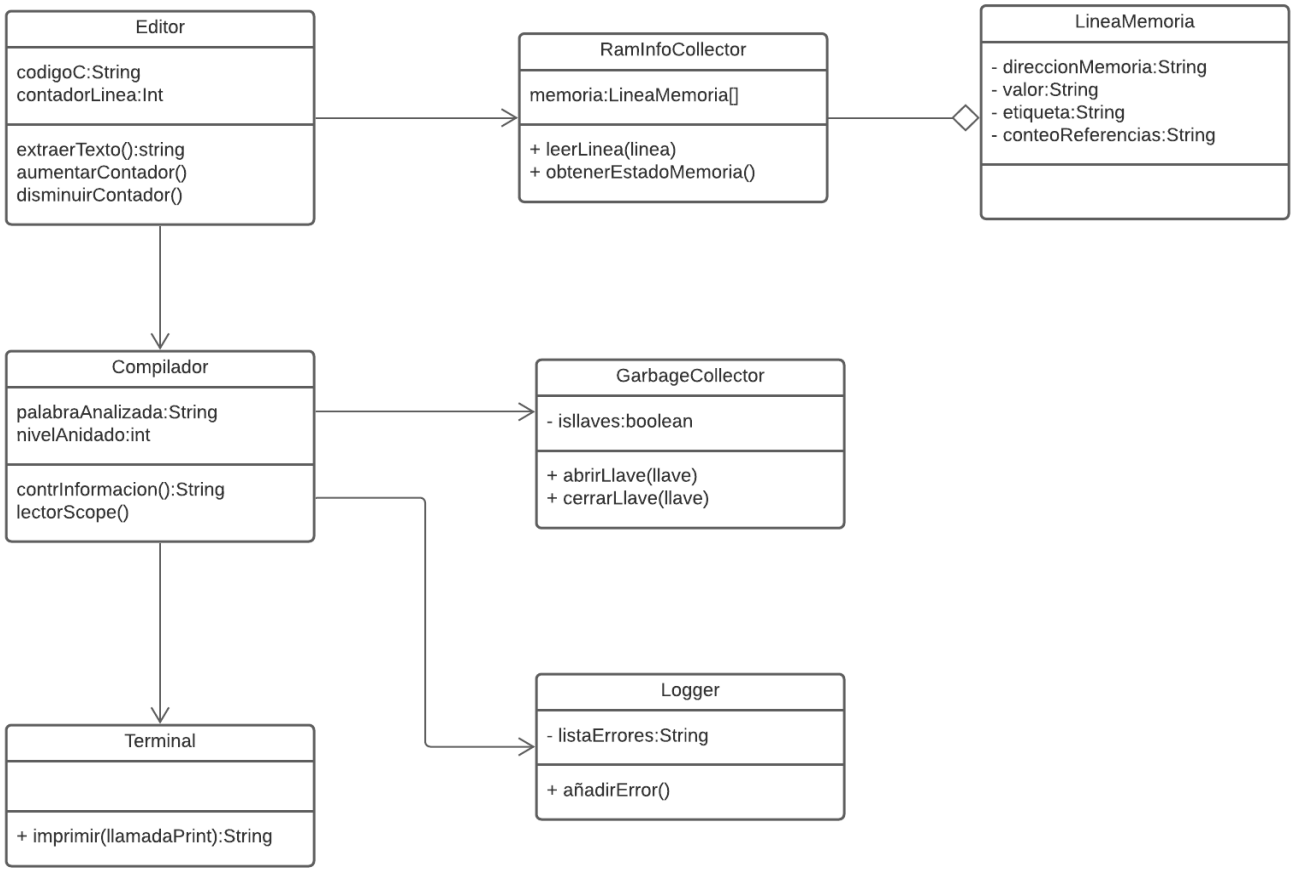


Diagrama de clases final

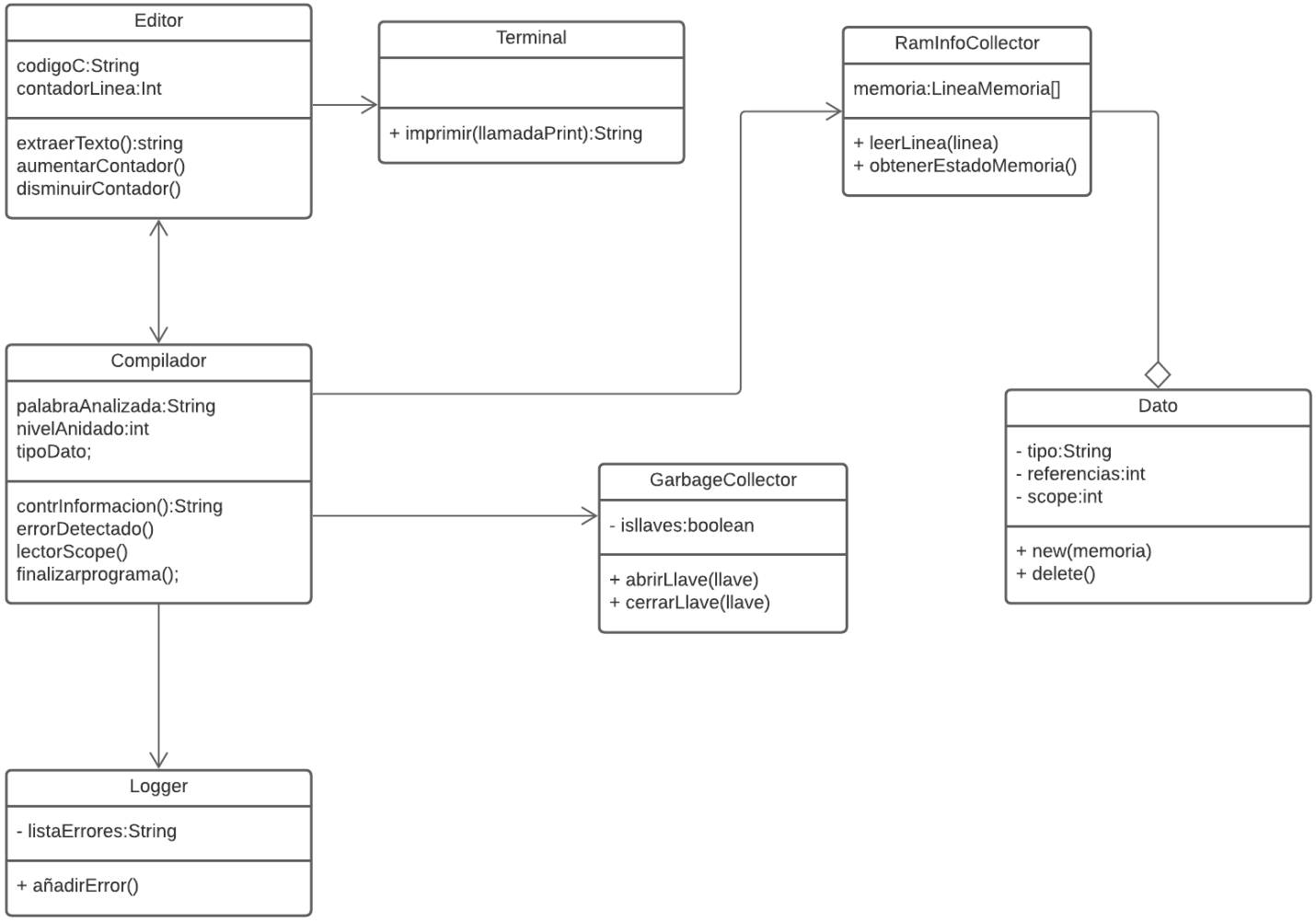
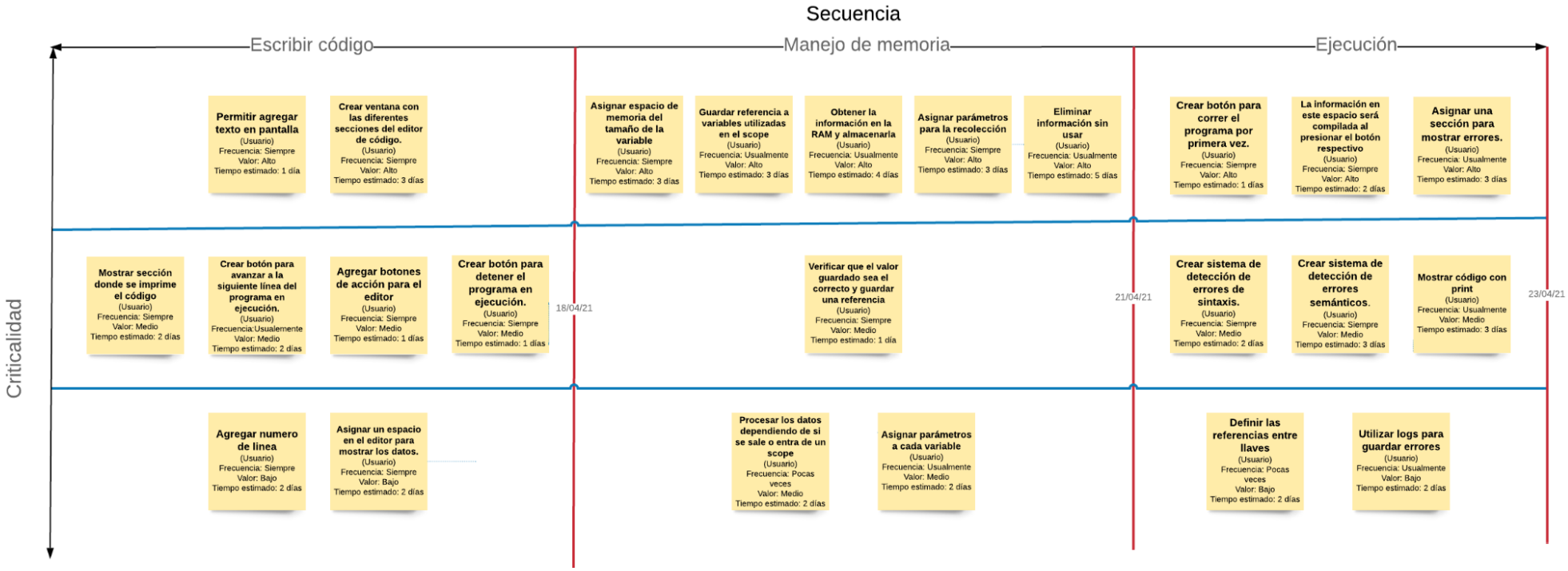


Diagrama de criticalidad y secuencia



Estructuras de datos

La interfaz del cliente fue creada con QT. Se divide en 4 partes:

- En la parte de arriba se muestra una pequeña región donde se ingresan 2 cosas: el número de puerto para conectar al server y la cantidad de bytes de memoria a utilizar para guardar variables en el IDE.
- Abajo se encuentra un editor de texto en donde se ingresa el código a compilar. Es la región que más ocupa espacio en el IDE
- A la derecha del editor se encuentra el RAM Live View, en donde se muestra el estado de la memoria RAM de cada línea del editor y lo muestra en pantalla.
- Bajo estos dos cuadros se encuentra el botón Run, que compila el código que está en el editor de texto.
- En el último cuadro se muestra una región con dos pestañas: El standard output, que muestra las llamadas en print que se hayan ingresado al editor después de compilarlo, y el application log, donde muestra todos los errores de sintaxis que hayan en el editor de texto.

La interfaz fue realizada con ayuda de QT creator, ya que tiene una interfaz en donde se pueden arrastrar los diferentes elementos de la interfaz en una ventana para crear el IDE de manera más eficaz.

Algoritmos desarrollados

Garbage Collector:

En este caso, el algoritmo dedicado al garbage collector, ha sido basado en el proceso de manejo de memoria y las clases implementadas dentro del proyecto para cumplir con este objetivo. En este caso la clase que compone las variables, tiene un espacio específico para denotar cuántas veces dicha variable ha sido referenciada en memoria, para que cuando el garbage collector realice un parseo de las variables que se van agregando en memoria debido a un scope o simplemente a la secuencia del programa, el Garbage collector pueda liberar los espacios en los que se encuentran las variables que tienen un nùmero de referencias igual a cero.

El garbage collector se encargará entonces de pasar uno o varios espacios que se encontraban en estado de ocupados a estar libres para poder almacenar en ellos nuevos datos y que el programa no llegue a quedarse sin espacio para su funcionamiento en un momento determinado.

RAM:

El algoritmo creado para la administración y visualización de la memoria ram es de los más complejos dentro del programa, ya que este debe de manejar los nuevos datos que se van añadiendo luego de cada lìnea de código. Debe analizar qué tipo de informaciòn recibe, para luego determinar cuánto espacio requerirá en memoria y finalmente encontrar un espacio libre para colocar esta nueva información, todo esto debe de quedar almacenado y bien especificado para que posteriormente el servidor pueda enviar todos estos datos al cliente y

que a este se le muestren en pantalla, además el manejo correcto de estos datos también ayudará al funcionamiento correcto del garbage collector.

Sockets:

La interfaz gráfica se maneja con QT. En la ventana del IDE hay una sección donde pide el número de puerto y la cantidad de memoria a utilizar en el IDE. El número de puerto es ingresado en el socket del cliente, para que el servidor lo busque en un tiempo determinado. El cliente contiene la interfaz del programa, mientras que el servidor maneja todos los procesos de compilación y manejo de la RAM, donde el cliente muestra luego los resultados.