Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores
(Computer Engineering Academic Area)

Programa de Licenciatura en Ingeniería en Computadores
(Licentiate Degree Program in Computer Engineering)

Curso: CE-1103 Algoritmos y Estructuras de Datos I

(Curse: CE-1103 Algorithms and Data Structures I)

Proyecto #1
(Project #1)

Realizado por:
Made by:
Deiber Granados Vega, 2017159397
Kisung Lim Ogawa, 2017098352

Profesor:
(Professor)
Jose I. Ramirez Herrera

Fecha: Cartago, Abril 18, 2018

I Semestre, 2018

Índice

Introducción

El presente proyecto tratará sobre el diseño e implementación de un ambiente de programación (IDE) para el pseudo lenguaje C!, el cual tiene una sintaxis derivada del lenguaje C. Todo esto junto con un manejador de memoria para el mismo. Además, se emplearán muchos temas vistos en clase como el manejo de la memoria dinámica y la creación de punteros en c++. Se nos muestra la primera prueba programada como proyecto en el curso de Algoritmos y Estructuras de Datos II donde el reto consiste en cómo podemos visualizar la memoria usada en un script o código simple usando el pseudo lenguaje C!. Al correr un programa cualquiera, la mayoría de las variable y los statements usados en la funciones se guardan en el stack o pila pero al querer manejar memoria la mayoría de estos enunciados son guardados en el Heap. Si queremos que nuestro objeto viva más que su propio scope, el uso del Heap es indispensable, por lo que en este proyecto se usarán los punteros para almacenar lo que codigo dicte y visualizarlo como un stack virtual. A continuación se le dará al lector lo que se quiere lograr con este proyecto en un ámbito académico y de formación, detallando las necesidades de un supuesto cliente con el trabajo y los patrones usados en el mismo.

Objetivo General

➔ Diseñar e implementar un ambiente de programación para el pseudo lenguaje C! junto con un manejador de memoria para el mismo.

Objetivos Específicos

➔ Aplicar conceptos de manejo de memoria.

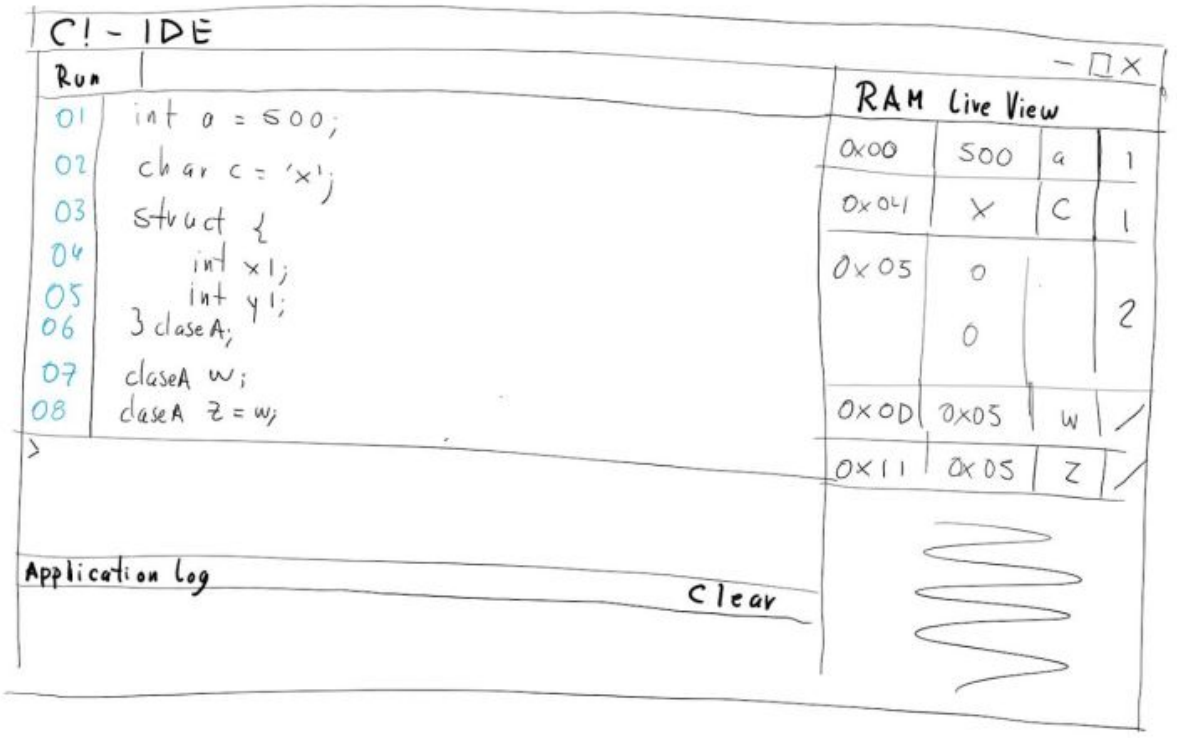
➔ Investigar y desarrollar una aplicación en el lenguaje de programación C++

➔ Investigar acerca de programación orientada a objetos en C++.

➔ Aplicar patrones de diseño en la solución de un problema.

Descripción del problema

El proyecto consiste en el diseño e implementación de un IDE para el pseudo lenguaje C!. El IDE tendrá un
layout similar al siguiente:



El cual contiene **un editor de C!** (Es un editor de texto que permite ingresar el código de C!. C! tiene una sintaxis derivada del lenguaje C), **Stdout** (Es la sección directamente bajo el editor. Cumple la función de standard output), **Application Log** (En esta sección se debe mostrar el log del IDE, es decir cualquier error interno, cualquier llamada al servidor de memoria, todo lo que ocurre internamente, se mostrará en esta sección) y **RAM Live View** (En esta sección se ve en tiempo real, el estado del RAM conforme cada línea de C! se ejecuta).

El IDE interpreta y ejecuta el código C! cuando se presiona el botón Run. La ejecución se hace de manera detenida al estilo de depuración. El usuario podrá detener la ejecución o avanzar línea por línea.

En caso de que el IDE detecte un error de sintaxis o semántico, la ejecución del programa se detiene y se muestra el error en stdout.

Detalles necesarios para ejecución

Se requiere que el usuario tenga los siguientes programas:

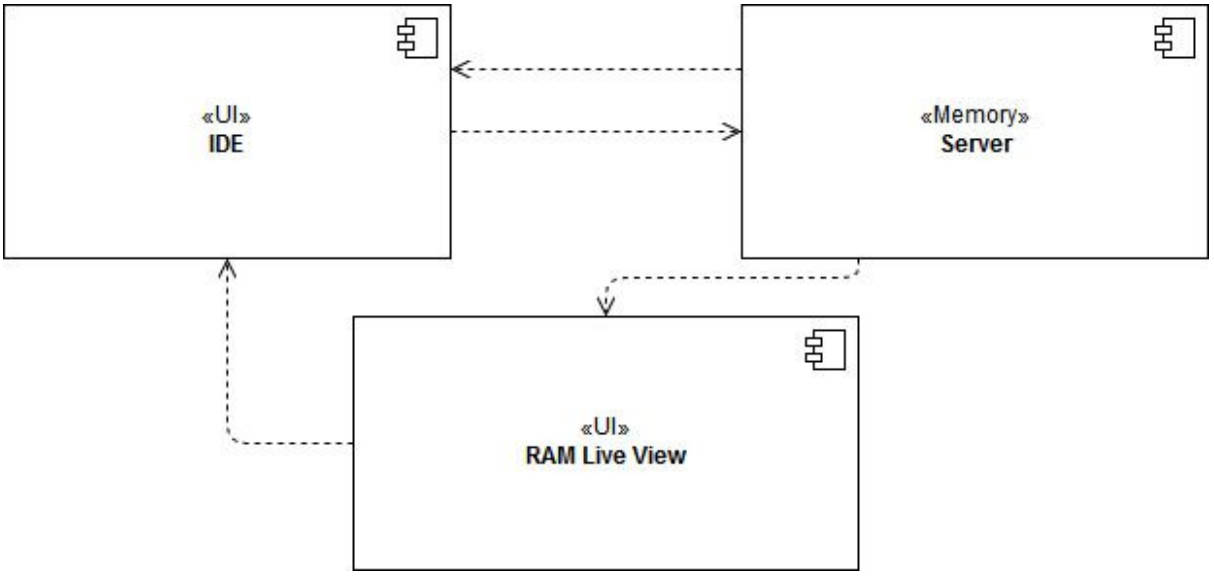
- Clion ver.2018.1.1 (código puro)
- GNU G++

Lista de features e historias de usuario

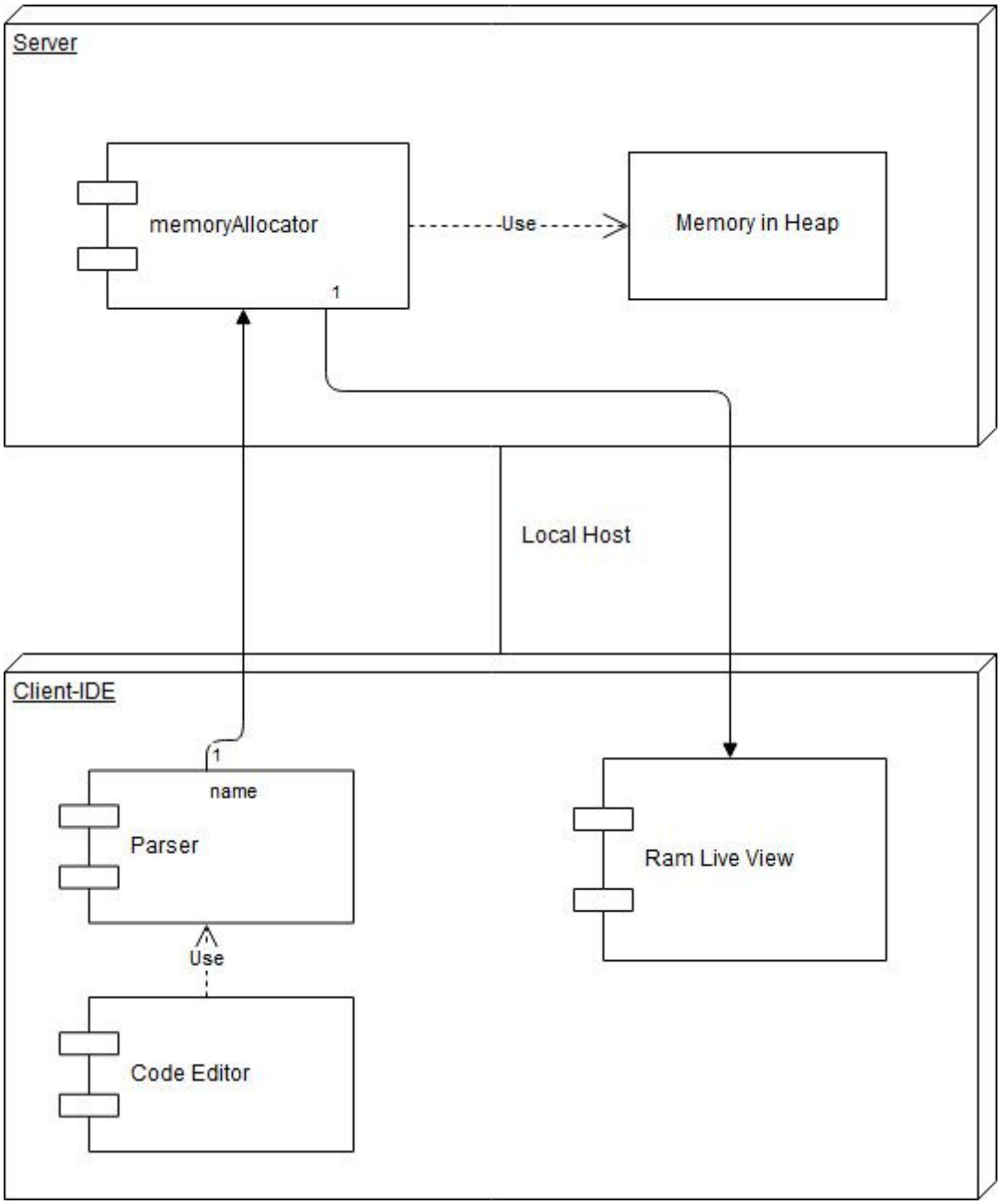
1. Poder ejecutar mi codigo linea por linea al darle run.
2. Visualizar las variables “vivas” en la memoria, su posición de memoria, su etiqueta y su número de referencias.
3. Tener un log del servidor donde se controla mi memoria.
4. El manejo de la memoria sea automático y tenga que intervenir en su uso.
5. Poder ver mis prints en un layout.
6. Escribir en un pseudo lenguaje de c! que contenga los tipos básicos, el struct y un referenciador que funcione como puntero.

Diagramas

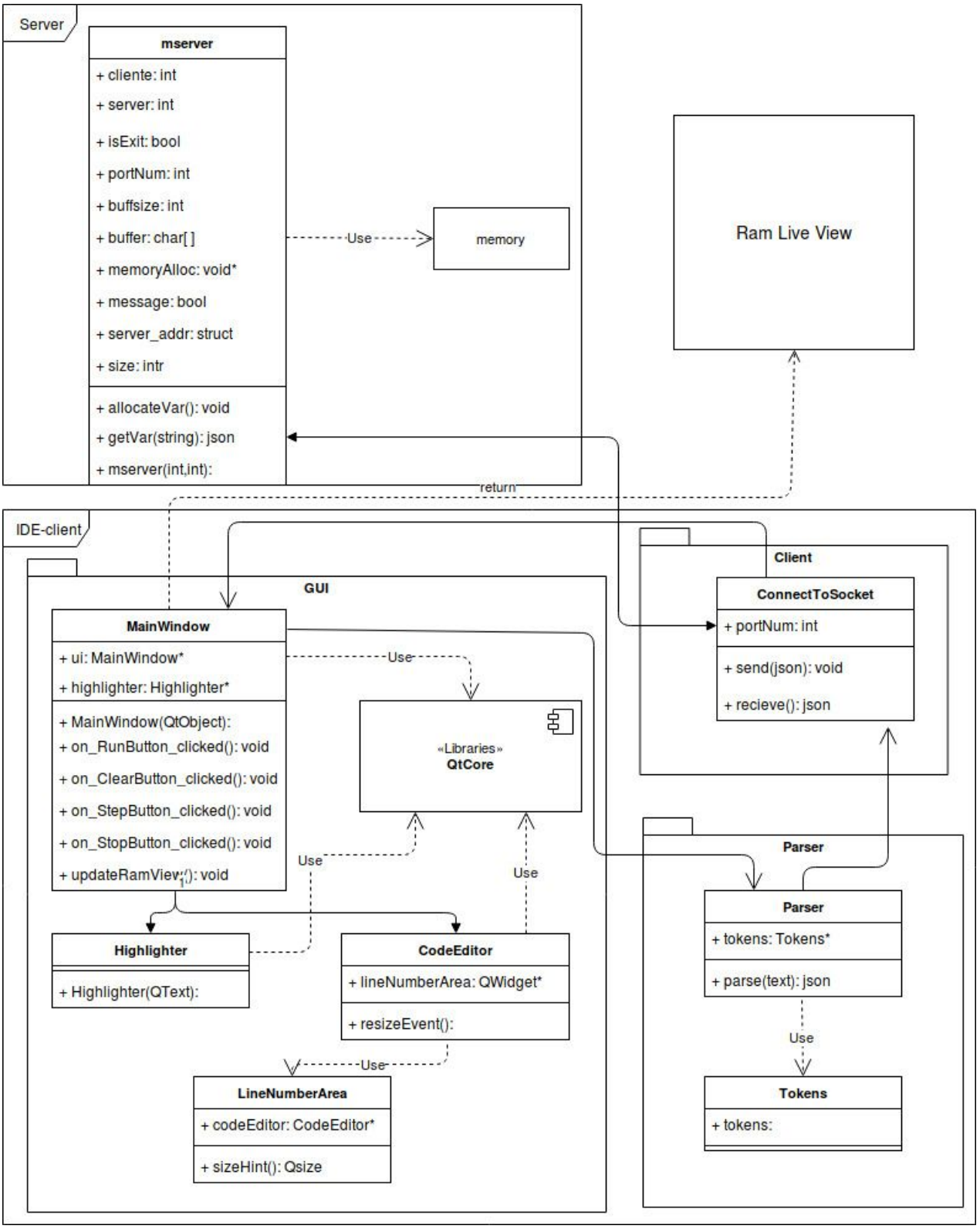
◆ Diagrama de componentes



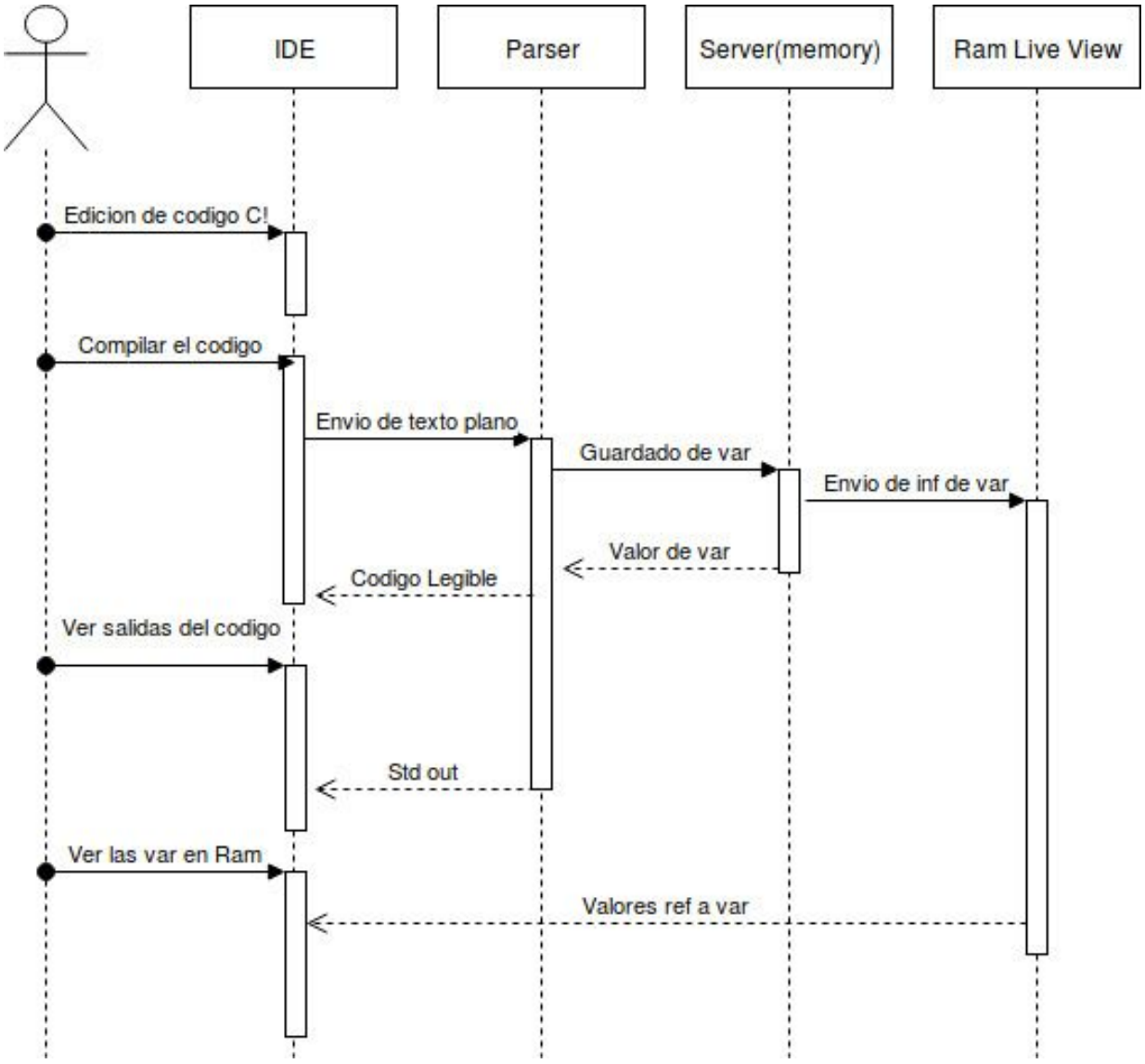
◆ Diagrama de despliegue



◆ Diagrama de clases



◆ **Diagrama de secuencia**



URL del repositorio de versiones

https://github.com/deibergv/Proyecto1DatosII

Descripción de las bibliotecas utilizadas

- **Qt**

Librería gráfica usada para la la creación de los elementos visuales del programa como la UI del IDE. Contiene varias librerías con las cuales se puede desarrollar objetos visuales como tablas, cuadros de textos y labels.

- **nlohmann/json**

Librería de un solo header que contiene las funciones de objetos jsons y puede parsear jsons desde strings. Usada para la comunicación entre el server y la IDE para la transmisión de información.

- **Loguru**

Librería de registro C ++ de solo cabecera, la cual detecta señales como errores de segmentación y divisiones por cero, además, de tener una manera amigable de reportalos por tipo.

Descripción detallada de los algoritmos desarrollados

- *Analizador sintáctico del código:*

Mediante

- *Guardado de variables en el servidor:*

Gracias al uso de la librería para la lectura y escritura json, se realiza la comunicación entre el servidor y el cliente por medio de esta anotación. Al recibir la lectura de json, el servidor envía al json a una función que interpreta los datos del json y determina el tipo de dato el cual se está refiriendo el json, el valor del dato y su etiqueta. La función determina cuanta memoria se necesita para guardar la variable en el void* sumando los bytes necesarios para el posicionamiento y ahí almacena la variable. Para los struct, se crea un espacio de memoria aún no determinado y se cuentan todas las variables inicializadas en el struct. Cuando se inicializa una variable pero a este no se le asigna valor, se guarda su espacio de memoria con su punto y su etiqueta, por lo que al serle asignado un valor, este es guardado en el espacio de memoria asignado.

Problemas encontrados

- *Implementación o creación del editor de código:*

Hubo grandes problemas en el planeamiento de lo necesario y el cómo se haría, luego de ardua investigación, se encontró en las páginas de la documentación de Qt, que ya existía un código base para aquello que buscaban la creación de un editor de código, además, se encontró otro código base para la implementación de coloreado en la sintaxis, los cuales fueron intentados implementar pero luego de mucho esfuerzo no se lograba que funcionara, se solució editando la llama a la clase del editor de texto plano implementado en la interfaz, haciendo que llamara a la clase creada con los implementos para editacion de código y sintaxis.

- *Analizador sintáctico del lenguaje C! y uso de librerías referentes a Json:*

Luego de exageradas búsquedas de información, se concluyó con que era mejor crear desde cero un analizador sintáctico, pero esto aún requería gran cantidad de tiempo y conocimientos que no se tenían, por lo que se usó como base otros analizadores. Todo buscando ser conectado con el servidor por medio de archivos Json que se intentaban crear al parsear el código, obteniendo la información que es verdaderamente relevante.

Bitácora

Antes de la siguiente fecha solo se dió investigación individual sobre información referente al proyecto.

20/03/18, 8pm – 21/03/18, 12am (Deiber)
Búsqueda exhaustiva de información referente a Qt, planeamiento sobre el proyecto e inicio de documentación base, orientación de ideas.

21/03/18, 4pm – 21/03/18, 9pm (Kisung)
Búsqueda de información sobre el la creación de sockets y servidores en c y c++.
Preparación y análisis del proyecto para el futuro desarrollo y los elementos necesarios en él.

25/03/18, 8pm – 25/04/18, 11pm (Deiber)
Inicio de creación de proyectos y pruebas de la información conseguida hasta el momento.

31/03/18, 2pm – 31/03/18, 8pm (Kisung)
Aplicación de conocimientos adquiridos tras investigación del servidor. Realización de pruebas de ejemplo y escogiendo el mejor código que se adapte al proyecto.

01/04/18, 1pm – 01/04/18, 8pm (Kisung)
Creación del servidor cliente para usar en el proyecto, realización de pruebas y determinado los componentes en que se montan en el servidor y en el cliente. Análisis de los posibles algoritmos a implementar en el memory allocator.

02/04/18, 6pm – 02/04/18, 11pm (Deiber)
Implementación de primera parte de la interfaz.

03/04/18, 1pm – 03/04/18, 3pm (Kisung)
Búsqueda de información acerca el parseo de código con el uso de librerías externas.

06/04/18, 7pm – 07/04/18, 2am (Deiber)
Implementación de editor de código en la interfaz, además, implementado un inicio en el coloreado de la sintaxis.

10/04/18, 1pm – 10/04/1, 3pm (Kisung)
Búsqueda de información acerca del manejo de Jsons y librerías para su posible uso.

11/04/18, 8pm – 11/04/18, 1am (Deiber)
Búsqueda exhaustiva sobre la forma de manejar el código y la implementación de un analizador sintáctico (parser)

14/04/18, 8pm – 15/04/18, 12am (Kisung)
Desarrollo del server para la comunicación con el cliente por medio de jsons.

15/04/18, 7pm – 16/04/18, 2am (Deiber)
Implementación del log, para la regulación de errores.

Actividad	**Tiempo**
Análisis de requerimientos	8 horas
Diseño de la aplicación	6 horas
Programación	38 horas
Documentación interna	3 hora
Documentación técnica	6 horas
Pruebas	4 horas
Total	**65 horas**

Conclusiones

Con este proyecto, se logró aprender sobre los procesos dado en la depuración de código para su compilación, además de investigar más a fondo que es un IDE o ambiente de programación y cómo funciona desde la parte más básica de programación, permitiendo llegar a completar el diseño e implementación de un ambiente de programación para C!.

El logró aprender sobre los conceptos relacionados con el manejo de memoria y uso de la misma. Además, luego de investigar y desarrollar la aplicación en el lenguaje C++, se aprendió que el ambiente que rodea CLion como editor de código, la implementación de librerías en el mismo y las grandes posibilidades que nos permite lo hacen una excelente opción para uso en grandes proyectos que mejoren el ambiente de todos los programadores y de los futuros amantes del software.

Bibliografía

Company, T. (2018). *Qt | Cross-platform software development for embedded & desktop*. [online] Qt.io. Available at: https://www.qt.io/ [Accessed March. 2018].

Doc.qt.io. (2018). *Code Editor Example | Qt Widgets 5.10*. [online] Available at: https://doc.qt.io/qt-5/qtwidgets-widgets-codeeditor-example.html [Accessed March. 2018].

Doc.qt.io. (2018). Syntax Highlighter Example | *Qt Widgets 5.10*. [online] Available at: https://doc.qt.io/qt-5/qtwidgets-richtext-syntaxhighlighter-example.html [Accessed March. 2018].

GitHub. (2018). *nlohmann/json*. [online] Available at: https://github.com/nlohmann/json [Accessed 10 Apr. 2018].

GitHub. (2018). *emilk/loguru*. [online] Available at: https://github.com/emilk/loguru [Accessed 14 Apr. 2018].