TEC | Tecnológico de Costa Rica

Algoritmos y Estructuras de Datos II

Área de Ingeniería en Computadores

Semestre I

2021

Proyecto I: Diseño e implementación de un IDE para el pseudo lenguaje C!

Estudiantes	Carné
Anthony Montero Román	2019275097
Juan Peña Rostrán	2018080231

Profesor:

Antonio González Torres

Breve Descripción del Problema

Se trata de un programa llamado mserver, indicando como parámetros el puerto en el que está escuchando y el tamaño en bytes de la memoria total.

Acerca del servidor

El servidor realiza un solo malloc de memoria total. El servidor tiene un mapa interno de todo el bloque de memoria. Por ejemplo, al inicio, puede reservar 1 MB de memoria real. Con cada solicitud recibida de C!, El servidor maneja las compensaciones para determinar la posición de cada variable en C! dentro del bloque de memoria real. ¡El servidor escucha las solicitudes de C! enviado por el IDE en formato JSON. Cuando el IDE C! enviar una solicitud debe indicar al menos el tipo de dato, el nombre de la variable y el tamaño a reservar. El servidor maneja la memoria automáticamente. ¡Es decir, C! nunca tendrá que liberar memoria ni solicitar crearla. El servidor mantiene el recuento de referencias y cada cierto tiempo ejecuta el recolector de basura que elimina los espacios de memoria a los que no se hace referencia.

Acerca del IDE:

El editor de C!: Es un editor de texto que permite ingresar el código de C!. C! tiene una sintaxis derivada del lenguaje C. A no ser que se indique lo contrario, se asumen las mismas reglas que el lenguaje de programación C. Más adelante se explican algunas reglas diferentes en C!.

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 (El estudiante definirá los botones que hagan esto).

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.

Stdout: Es la sección directamente bajo el editor. Cumple la función de standard output. Si el código C! contiene llamadas a printf, print, cout o cualquier función similar (puede ser definida por el estudiante), el resultado se imprime en esta sección.

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. (Investigar el uso de algún framework de logging en C++ similar a Log4j).

RAM Live View: En esta sección se ve en tiempo real, el estado del RAM conforme cada línea de C! se ejecuta. Esta vista consulta constantemente al servidor de RAM para obtener el estado de la memoria y lo pinta en pantalla. Muestra al menos los siguientes datos:

- Dirección de la memoria
- Valor
- Etiqueta
- Conteo de referencias.

Bibliotecas necesarias:

Este proyecto fue realizado en Ubuntu (Linux) en el lenguaje C++ Primero se necesita una biblioteca de manejo de archivos JSON. Se puede instalar con los siguientes comandos:

```
sudo apt-get update
```

```
sudo apt-get install nlohmann-json3-dev
```

Además, se necesita la biblioteca lib boost. Se puede instalar con los siguientes comandos:

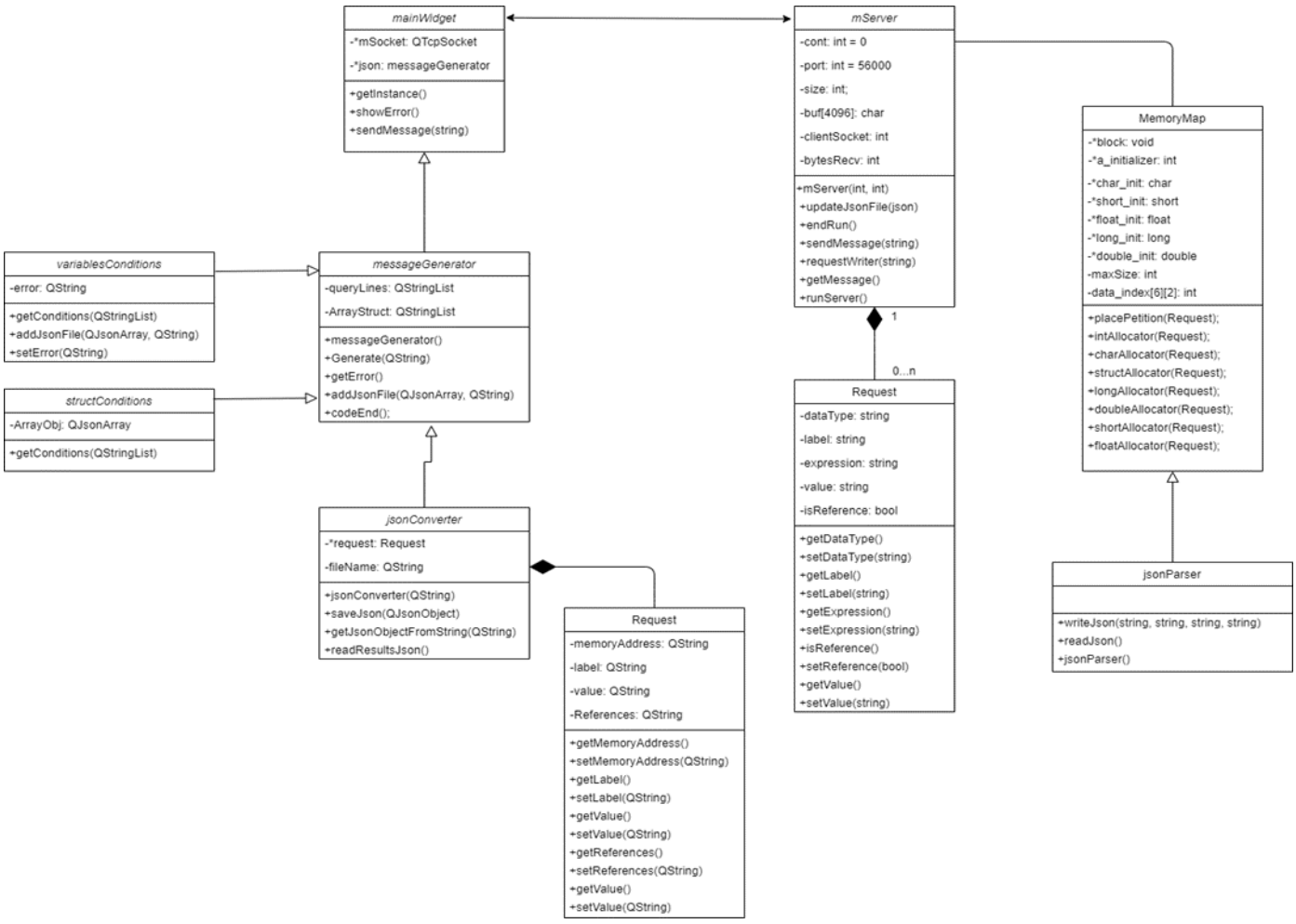
```
sudo apt update -y
```

```
sudo apt install -y libboost-all-dev
```

Descripción del repositorio Server:

En el repositorio del server de lenguaje C!, se almacena el código encargado de manejar el "*backend*" de la aplicación para el IDE del lenguaje C!, la cual es una aplicación que permite a los usuarios escribir código del lenguaje de programación C!.

Diagrama de clases



Planificación y administración del proyecto

Se utilizó Jira Software para manejar de forma remota la planificación y administración del proyecto entre los integrantes del grupo. Se pueden visualizar los siguientes datos aquí: https://anthonymoro08.atlassian.net/jira/software/c/projects/PIDII/boards/2/roadmap

Lista de características: Las características que presenta la aplicación se representaron por medio de *epics* de Jira. Los features considerados son:

-Editor de C!

-Standard output

-Application log

-RAM Live View

-Comunicación con Sockets para serialización en formato JSON

-Administración de memoria en el servidor

Historias de usuario distribuidas por criticalidad:

-Lista de historias de usuario:

1. Yo como usuario quiero un editor de texto que permita ingresar el código C!
2. Yo como usuario quiero un botón "run" con el que se ejecute el código
3. Yo como usuario quiero el tipo de dato _reference _para el lenguaje C! soporte el operador "getValue"
4. Yo como usuario quiero que los tipos de datos con excepción de los tipo reference soporten el operador "getAddr"
5. Yo como usuario quiero una sección bajo el editor que cumpla la función de standard output
6. Yo como usuario quiero que cuando haya un error este se muestre en el stdout.
7. Yo como usuario quiero que se muestre en el log del IDE todo lo que ocurre internamente en el programa.
8. Yo como usuario quiero una sección donde se vea en tiempo real el estado de la RAM conforme cada línea se ejecuta.
9. Yo como usuario quiero que C! se conecte a un servidor para cumplir con las funciones del programa.
10. Yo como usuario quiero que el servidor interprete peticiones de C! para ejecutar las operaciones requeridas.
11. Yo como usuario quiero un servidor que reserve una cantidad de memoria para usar el Lenguaje C!
12. Yo como usuario quiero que el servidor ejecute el garbage collector automáticamente para liberar memoria.
13. Yo como usuario quiero que el servidor tenga un mapa del bloque de memoria para el control del offsets.

A continuación, se muestra una tabla con las historias de usuario divididas según su prioridad en la implementación del proyecto.

Media	Alta
Historia de usuario 3	Historia de usuario 1
Historia de usuario 4	Historia de usuario 2
Historia de usuario 5	Historia de usuario 8
Historia de usuario 6	Historia de usuario 9
Historia de usuario 7	Historia de usuario 10
Historia de usuario 12	Historia de usuario 11
	Historia de usuario 13

Problemas sin resolver

En esta sección se detallan algunos errores del proyecto que no han sido resueltos.

Alineamiento de los datos tipo char: Hay un bug que provoca que los tipos de datos char no se ubiquen de forma alineada con los demás espacios de memoria que han sido ocupados.

Creación de objetos struct: Al mandar la dirección de memoria en la que se ubica la estructura se nota que no es una dirección que forma parte del bloque reservado. Esto puede darse porque la asignación de atributos dentro del struct no se puede hacer de forma dinámica.

El conteo de referencias: De momento no se ha habilitado el correcto funcionamiento del conteo de referencias en un scope del código hecho en el IDE.