Proyecto I – C!

Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Algoritmos y Estructuras de Datos II (CE 2103)
Segundo Semestre 2017
Valor 25%

Objetivo General

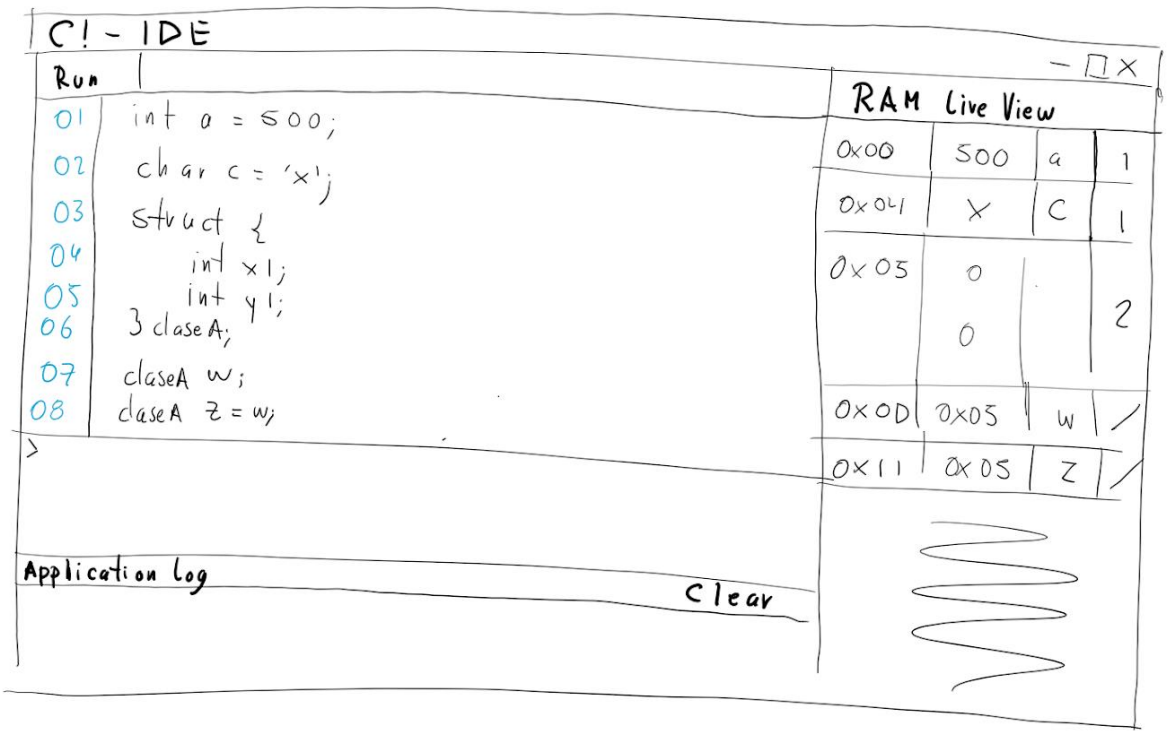
- ➔ Diseñar e implementar un ambiente de programación para el pseudo lenguaje C! junto con un manejador de memoria para el mismo.

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:



A continuación se explica cada una de estas secciones.

- **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. Este 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.

Sobre el lenguaje C!

Es un subconjunto del lenguaje C. C! tiene los siguientes tipos de datos:

int	4 bytes	Se maneja como copias
long	8 bytes	Se maneja como copias
char	1 byte	Se maneja como copias
float	4 bytes	Se maneja como copias
double	8 bytes	Se maneja como copias
struct	Variable. Se compone de tipo de datos primitivos. No soporta anidación de estructuras. Sigue el mismo sintaxis que C para declararlas.	Se maneja como referencias, es decir, el operador = no hace copias de los datos, únicamente de las referencias.
reference<*tipo*>	4 bytes	Se maneja como copias. Es decir, el operador = aplicado sobre dos referencias del mismo tipo, copia las direcciones, pero no los datos.

Todos los tipos con excepción de los reference, soportan la operación getAddr que retorna un reference<tipo>. Los tipos reference, soportan el operador getValue que retorna el valor apuntado por la referencia.

En C! todas las variables se alojan en el RAM remoto.

C! no soporta declaración de procedimientos/funciones. Sin embargo soporta scope mediante { }. Este scope permite definir la visibilidad de las variables. Es decir, una variable definida fuera de { } se puede ver internamente en el código dentro de { }. Sin embargo, una variable definida dentro de { } no puede ser vista fuera de dicho bloque.

OK	NOT OK

`int a = 5;`
`{`
`int b = a + 1;`
`}`
	`{`
	`int b = 1;`
	`}`
	`int a = b + 15;`

Conforme el código avance y se van cerrando los }, el conteo de referencias de dichas variables se reduce para que sea considerado posteriormente por el garbage collector.

Sobre el manejo de memoria

Se debe diseñar e implementar un servidor que administra la memoria. El flujo en términos generales es el siguiente:

- Se ejecuta un programa denominado *mserver* indicando como parámetros el puerto en el que escucha y el tamaño en bytes de la memoria total.
- El servidor realiza un único malloc de la memoria total. El servidor tiene un mapa interno del bloque entero de memoria. Por ejemplo, al iniciar puede reservar 10 MB de memoria real. Con cada petición recibida de C!, el servidor maneja offsets para determinar la posición de cada variable de C! dentro del bloque de memoria real.
- El servidor escucha peticiones de C! IDE enviados en formato JSON (el contenido de las peticiones es definido por el estudiante). Cuando C! IDE envía una petición debe indicar al menos el tipo de datos, el nombre de la variable (si aplica) y el tamaño que se deberá reservar.
- El servidor maneja la memoria automáticamente. Es decir, C! nunca tiene que liberar memoria o solicitar la creación de la misma.
- El servidor lleva el conteo de referencias y cada cierto tiempo ejecuta el garbage collector que elimina los espacios de memoria que no son referenciados.

Documentación requerida

1. Se deberá entregar un documento que contenga:
    a. Todas las partes estándar: Portada, índice, introducción, conclusión, bibliografía y anexos.
    b. El cuerpo del documento debe contener:
        ➔ Breve descripción del problema
        ➔ **Planificación y administración del proyecto**
            - Lista de features e historias de usuario identificados de la especificación
            - Distribución de historias de usuario por criticalidad y secuencia de uso
            - Minimal system span

- Plan de iteraciones que agrupen cada bloque de historias de usuario de forma que se vea un desarrollo incremental
- Asignación de user stories por miembro del equipo
- Descomposición de cada user story en tareas.
- Bitácora de trabajo que muestre tiempo invertido para cada actividad y la fecha en que se realizó

➔ **Diseño**

- Diagrama de componentes
- Diagrama de despliegue
- Diagrama de clases
- Diagrama de secuencia (Seleccionar al menos 4 Historias de Usuario)

➔ **Implementación**

- Descripción de las bibliotecas utilizadas
- Descripción detallada de los algoritmos desarrollados.
- Problemas encontrados: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo. Incluye descripción detallada, intentos de solución sin éxito, solución encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.

Aspectos operativos y evaluación:

1. **Fecha de entrega: De acuerdo al cronograma del curso**
2. El proyecto tiene un valor de 25% de la nota del curso
3. El trabajo es individual.
4. Es obligatorio utilizar un manejador de versiones del código. Puede ser git o svn. Se revisará que cada commit lleve comentarios relevantes relacionados con alguna tarea identificada en la sección de planificación
5. Los proyectos que no cumplan con los siguientes requisitos **no serán revisados**:
    a. Toda la solución debe estar integrada
6. El código tendrá un valor total de 65%, la documentación 30% y la defensa 5%.
7. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
8. Se evaluará que la documentación sea coherente, acorde a la dificultad/tamaño del proyecto y el trabajo realizado, se recomienda que realicen la documentación conforme se implementa el código.
9. La nota del código NO podrá exceder en 35 puntos la nota de la documentación, por lo cual se recomienda documentar conforme se programa.
10. Se debe enviar el código (preliminar) y la documentación a más tardar a las 23:59 del día de la fecha de entrega al email del profesor. **Se debe seguir el formato del Subject descrito en el Programa del Curso.**
11. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
12. Los estudiantes pueden seguir trabajando en el código hasta 15 minutos antes de la cita revisión oficial, momento en el cual deben enviar un email con el código fuente, si no lo hacen se asumirá que la versión oficial del código fue la enviada el día de la fecha de entrega junto con la documentación. **Se deben seguir las mismas indicaciones del punto 10.**

13. Aún cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
    a. Si no se entrega documentación, automáticamente se obtiene una nota de 0.
    b. Si no se utiliza un manejador de código se obtiene una nota de 0.
    c. Si el código y la documentación no se entregan en la fecha indicada se obtiene una nota de 0.
    d. Sí el código no compila se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
    e. El código debe ser desarrollado en C++ para GNU Linux, en caso contrario se obtendrá una nota de 0.
    f. Si se utilizan bibliotecas QT para algo diferente a UI se obtendrá una nota de 0.
    g. Si no se siguen las reglas del formato de email se obtendrá una nota de 0.
    h. La nota de la documentación debe ser acorde a la completitud del proyecto.
14. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto. El único requerimiento que se consultará durante la defensa del proyecto es el diagrama de clases, documentación interna y la documentación en el manejador de código.
15. Cada estudiante tendrá como máximo 15 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.
16. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
17. Cada estudiante es responsable de llevar los equipos requeridos para la revisión, si no cuentan con estos deberán avisar al menos 2 días antes de la revisión a el profesor para coordinar el préstamo de estos.
18. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.