Licenciatura en Ingeniería de Software
Taller de Programación Visual III

UNIDAD I C# Avanzado	Tarea # 7
Nombre: LUIS ENRIQUE SANTOS LANDEROS	Fecha: 27/01/2019

Introducción

Un tema muy importante como lo es el recolector de basura, es que la capacidad en cómo se libera la memoria, de archivos basura, hace que cualquier programa dentro de cualquier sistema sea altamente eficaz, dando la confianza de que no sucederán fallos dentro de este, el cual al generar un objeto este asigna direcciones en la memoria que no volveremos a usar, como es el caso de los punteros, que al direccionar nuestra memoria básicamente quedan como trozos de pastel que jamás serán comidos por el sistema, haciendo más pesado el proceso de nuestra aplicación, dado un caso como lo es .net CLR que usa memoria temporal, el recolector de basura se encarga de que, como ya lo habíamos mencionado del trozo de pastel que no se comerá, este la recoge sin más ni más, limpiando nuestro sistema cada vez que sea necesario.

Su eficiencia se basa generaciones las cuales va tomando en cuenta las que se usan frecuentemente y por otro lado las que no, estas se van situando con archivos que se usan frecuentemente, en el momento que ya no se usan, va cambiando de generación hasta el momento en que ya no sea totalmente alcanzable para el recolector de basura, y es en este momento cuando elimina por completo.

Desarrollo

El recolector de basura, ¿Qué es?... Básicamente el recolector de basura es quien emplea detrás de nuestra aplicación en cualquier sistema, dado este caso es en el lenguaje de C#, este eficiente actualmente dado que en sus versiones anteriores su forma de trabajar era más lenta que la que actualmente se usa, básicamente lo que hace es ir direccionándose o por decirlo enlazándose a los archivos más significativos y los menos significativos en el sistema de nuestra memoria, esto se hace mediante generaciones los cuales empieza desde la primera generación la cual, se encuentran los archivos más usados por nuestra aplicación, mientras que aquellos que son menos usados en el momento van sumando, una nueva generación la cual no hace uso de esa memoria la cual fue creada, estas son temporales sin embargo, el sistema causa conflictos no tan notorios pero si dentro del sistema, en este caso pueden ser los objetos creados por la aplicación y que se usan temporalmente, en las versiones anteriores solo se enfocaban en la dirección de la memoria el cual

no se definía claramente, cuales se usaban temporalmente y cuales dejaban poco a poco de usarse, la versión actual de la (garbage collector) es más eficiente, siendo que se ejecuta en segundo plano, y no interfiere en la aplicación, es por eso que está mejor estructurada de esta forma , la limpieza es mucho mejor, ya que no interfiere sobre los procesos, quizás un ejemplo, que una aplicación deje mucha memoria sin usar, prácticamente tardaría unos dos minutos, lo cual no sería mucho si estamos hablando de cualquier CPU en general, que cualquier persona podría tener, dentro de esto la memoria que hace el papel principal de la garbage collector siempre como programador es importante, saber que vamos a usar continuamente y que no, al declarar las variables hay que ser cuidadosos de la memoria que hay que usar, esto es hacer un buen uso de optimización del recolector de basura.

Los objetos son siempre una referencia que debemos tener en cuanta al momento de usar fielmente la optimización de nuestra aplicación, un ejemplo son los archivos JSON (acrónimo de JavaScript Object Notation, «notación de objeto de JavaScript») el cual este formato de texto ligero para el intercambio de datos, podemos hacer las referencias de los cambios como si fueran unas cervezas, que con el tiempo se van enfriando, y así tenerlas frescas cada momento en que las vamos usando, pero si no logramos hacer que esas cervezas se mantengan frescas, el garbage collector va a trabajar en contra de nosotros, una forma de resolver eso si volvemos a la memoria a través de nuestros códigos, usamos la reutilización la cual va a hacer que las se mantengan con valores actuales, el consumo de memoria es bastante estable y la razón de esto es, por su puesto de que nosotros estamos creando un nuevo diccionario allí, una y otra vez estamos actualizando nuestras variables, con el fin de reutilizar nuestros archivos que se usan poco.

La cadenas duplicadas, es un tema de consola, por ejemplo al obtener los diferentes nombres de los días de la semana, en la memoria al menos tres veces sucederá el duplicado, y estarán en la memoria de cualquier red, esto hace que se ahogue el sistema en este tipo de cadenas, lo cual es conveniente reutilizar la cadena existente y simplemente hacer uso de una referencia por lo que si corremos la aplicación es probable de que la cadena aun siga dentro de la memoria o de lo contrario sería que permanezca hasta que la aplicación se cierre, esto sería una regla general, pero o se encarga la garbage collector, y se puede asegurar un buen uso del sistema en la memoria si duplicidad.

Esto es muy importante por los punteros, en los cuales podemos direccionar las que se usan más y cuáles no, siempre estarán ahí, apuntando a los diferentes tipos de valores asignados en la memoria, que posiblemente nosotros descartamos, pero la eficiencia del recolector de basura hace que es punto en concreto no se pierda de vista del sistema de mando en el código si no que la función principal es su direccionamiento a las nuevas y antiguas direcciones dentro de la memoria al inicio y al final de nuestra aplicación.

Conclusión

La garbage collector es una operación secundaria el cual siempre se encarga de eliminar o reutilizar partes de la memoria que no están siendo usadas, es por eso que se asignan generaciones dentro de la memoria para que se puedan optimizar cualquier aplicación que estemos haciendo en nuestros lenguajes de programación, en este caso es el lenguaje de C# con el .NET que si bien los objetos que estemos usando siempre estarán ahí sin que nos demos cuenta de su uso generando cache en la memoria por mantener la buena corrección y ejecución de nuestro programa, pero esto se va optimizando hasta que vuelva a correr o cerrar nuestra aplicación y esto es básicamente la garbage collector.