Manejo de memoria estática

La forma más fácil de almacenar el contenido de una variable en memoria en tiempo de ejecución es en memoria estática o permanente a lo largo de toda laejecución del programa.No todos los objetos (variables) pueden ser almacenados estáticamente.Para que un objeto pueda ser almacenado en memoria estática su tamaño(número de bytes necesarios para su almacenamiento) ha de ser conocido en tiempo de compilación, como consecuencia de esta condición no podrán almacenarse en memoria estática:* Los objetos correspondientes a procedimientos o funciones recursivas, ya que en tiempo de compilación no se sabe el número de variables que serán necesarias.* Las estructuras dinámicas de datos tales como listas, árboles, etc. ya que el número de elementos que las forman no es conocido hasta que el programa se ejecuta.Las técnicas de asignación de memoria estática son sencillas.A partir de una posición señalada por un puntero de referencia se aloja el objeto X, y se avanza el puntero tantos bytes como sean necesarios para almacenar el objeto X.La asignación de memoria puede hacerse en tiempo de compilación y los objetos están vigentes desde que comienza la ejecución del programa hasta que termina.En los lenguajes que permiten la existencia de subprogramas, y siempre que todos los objetos de estos subprogramas puedan almacenarse estáticamente se aloja en la memoria estática un registro de activación correspondiente a cada uno de los subprogramas.Estos registros de activación contendrán las variables locales, parámetros formales y valor devuelto por la función.Dentro de cada registro de activación las variables locales se organizan secuencialmente. Existe un solo registro de activación para cada procedimiento y por tanto no están permitidas las llamadas recursivas. El proceso que se sigue cuando un procedimiento p llama a otro q es el siguiente:1. p evalúa los parámetros de llamada, en caso de que se trate de expresiones complejas, usando para ello una zona de memoria temporal para el almacenamiento intermedio. Por ejemplos, si la llamada a q es q((3*5)+(2*2),7) las operaciones previas a la llamada propiamente dicha en código

máquina han de realizarse sobre alguna zona de memoria temporal. (En algún momento debe haber una zona de memoria que contenga el valor intermedio 15, y el valor intermedio 4 para sumarlos a continuación). En caso de utilización de memoria estática ésta zona de temporales puede ser común a todo el programa, ya que su tamaño puede deducirse en tiempo de compilación.2. q inicializa sus variables y comienza su ejecución.

Manejo de memoria dinámica

¿Qué es la memoria dinámica?Supongamos que nuestro programa debe manipular estructuras de datos de longitud desconocida. Un ejemplo simple podría ser el de un programa que lee las líneas de un archivo y las ordena. Por tanto, deberemos leer un número indeterminado de líneas, y tras leer la última, ordenarlas. Una manera de manejar ese ``número indeterminado'', sería declarar una constanteMAX_LINEAS, darle un valor vergonzosamente grande, y declarar un array de tamaño MAX_LINEAS. Esto, obviamente, es muy ineficiente (y feo). Nuestro programa no sólo quedaría limitado por ese valor máximo, sino que además gastaría esa enorme cantidad de memoria para procesar hasta el más pequeño de los ficheros.La solución consiste en utilizar memoria dinámica. La memoria dinámica es un espacio de almacenamiento que se solicita en tiempo de ejecución. De esa manera, a medida que el proceso va necesitando espacio para más líneas, va solicitando más memoria al sistema operativo para guardarlas. El medio para manejarla memoria que otorga el sistema operativo, es el puntero, puesto que no podemos saber en tiempo de compilación dónde nos dará huecos el sistema operativo (en la memoria de nuestro PC). Memoria Dinámica. Sobre el tratamiento de memoria, GLib dispone de una serie de instrucciones que sustituyen a las ya conocidas por todos malloc, free, etc. y, siguiendo con el modo de llamar a las funciones en GLib, las funciones que sustituyen a las ya mencionadas son g_malloc y g_free. Reserva de memoria .La función g_malloc posibilita la reserva de una zona de memoria, con un

número de bytes que le pasemos como parámetro. Además, también existe una función similar llamada g_malloc0 que, no sólo reserva una zona de memoria, sino que, además, llena esa zona de memoria con ceros, lo cual nos puede beneficiar si se necesita un zona de memoria totalmente limpia.gpointer g_malloc (gulong numero_de_bytes);gpointer g_malloc0 (gulong numero_de_bytes);Existe otro conjunto de funciones que nos permiten reservar memoria de una forma parecida a cómo se hace en los lenguajes orientados a objetos.

Bibliografias

Cinthia Carrasco, Estructura de datos.2012