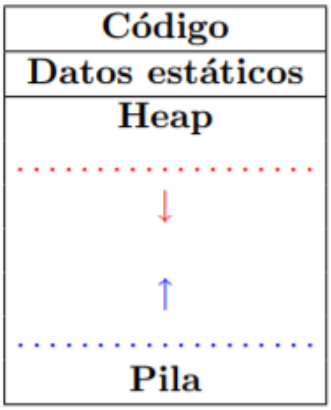
Organización de la memoria

Podemos distinguir tres tipos de memoria. En primer lugar, está la ocupada por el código del programa. Por otro lado, está la destinada a datos estáticos, que se gestiona en tiempo de compilación. Finalmente, tenemos la memoria que debe gestionarse en tiempo de ejecución. Hay poco que decir sobre la memoria ocupada por el programa. Su tamaño es conocido en tiempo de compilación y, en la mayoría de los lenguajes modernos, se puede decir que es "invisible": el código no puede referirse a sí mismo ni modificarse. La memoria destinada a datos estáticos se utiliza para las variables globales y algunas otras como las variables estáticas de las subrutinas. La gestión de esta memoria es sencilla y se puede realizar en tiempo de compilación. Al hablar de la memoria gestionada en tiempo de ejecución, podemos distinguir entre la memoria que se utiliza para albergar los objetos que se crean y destruyen con la ejecución de las subrutinas (parámetros, variables locales y algunos datos generados por el compilador) y la que se suele conocer como memoria dinámica, que se reserva explícitamente por el programador o que se necesita para almacenar objetos con tiempos de vida o tamaños desconocidos en tiempo de compilación. La memoria asociada a las subrutinas será gestionada mediante una pila. La memoria dinámica se localizara en un heap.

Normalmente, la memoria se organiza de una manera similar a:



Manejo de memoria estática

La gestión de la memoria estática es la más sencilla. De hecho, algunos lenguajes de programación antiguos, como las primeras versiones de FORTRAN, únicamente tienen reserva estática de memoria. Las principales ventajas son la sencillez de implementación y que los requerimientos de memoria del programa son conocidos una vez compilado el mismo.

Morgado Jacome Eduardo #17211545

Sin embargo, existen bastantes inconvenientes:

- El tamaño de los objetos debe ser conocido en tiempo de compilación: no se puede trabajar con objetos de longitud variable.
- Es difícil para el programador definir el tamaño de las estructuras que va a usar: si son demasiado grandes, desperdiciara memoria; si son pequeñas, no podrá utilizar el programa en todos los casos.
- Solo puede haber una instancia de cada objeto: no se pueden implementar procedimientos recursivos.

Por estas razones, casi todos los lenguajes de programación tienen además la posibilidad de gestionar parte de la memoria de manera dinámica. La gestión de esta memoria se puede hacer íntegramente en tiempo de compilación. Para acceder a un objeto, el programa únicamente necesita saber en que dirección se encuentra. Estas direcciones se pueden asignar secuencialmente. Basta con que el compilador tenga anotada la primera dirección libre del bloque de memoria estática y la vaya actualizando sumando la talla de los objetos que va reservando.

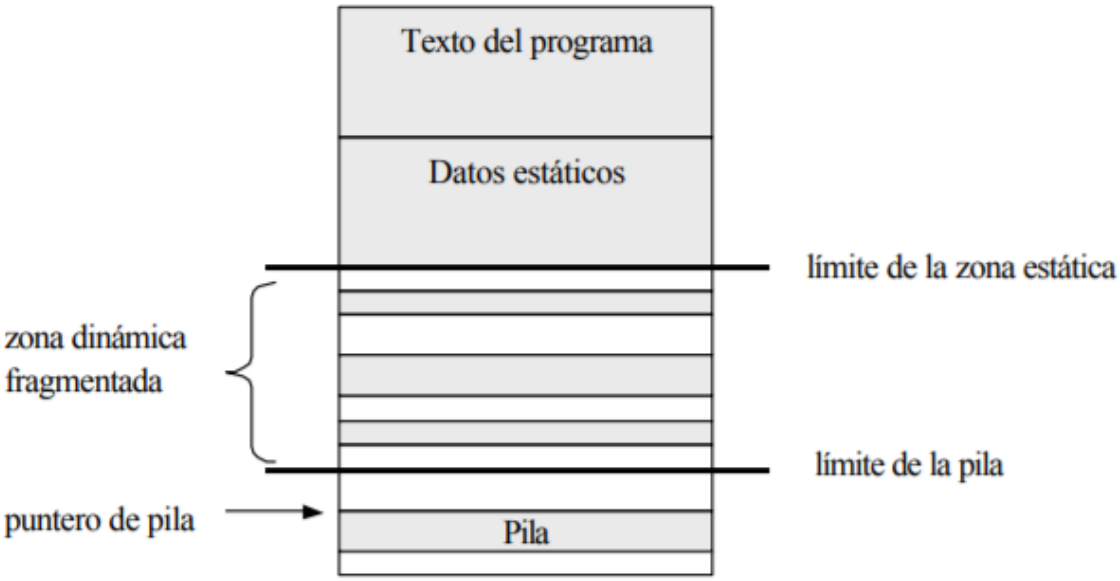
Manejo de memoria dinámica

Datos dinámicos: su tamaño y forma es variable (o puede serlo) a lo largo de un programa, por lo que se crean y destruyen en tiempo de ejecución. Esto permite dimensionar la estructura de datos de una forma precisa: se va asignando memoria en tiempo de ejecución según se va necesitando.

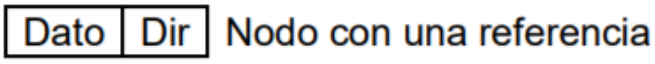
Cuando el sistema operativo carga un programa para ejecutarlo y lo convierte en proceso, le asigna cuatro partes lógicas en memoria principal: texto, datos (estáticos), pila y una zona libre. Esta zona libre (o heap) es la que va a contener los datos dinámicos, la cual, a su vez, en cada instante de la ejecución tendrá partes asignadas a los mismos y partes libres que fragmentarán esta zona, siendo posible que se agote si no se liberan las partes utilizadas ya inservibles. (La pila también varía su tamaño dinámicamente, pero la gestiona el sistema operativo, no el programador):

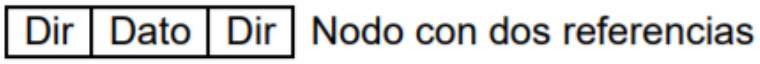
Para trabajar con datos dinámicos necesitamos dos cosas:

1. Subprogramas predefinidos en el lenguaje que nos permitan gestionar la memoria de forma dinámica (asignación y liberación).
2. Algún tipo de dato con el que podamos acceder a esos datos dinámicos (ya que con los tipos vistos hasta ahora solo podemos acceder a datos con un tamaño y forma ya determinados).



Así como existen estructuras de datos estáticas (arreglos), también existen estructuras de datos dinámicas (listas y árboles), estas últimas son generadas a partir de un tipo de dato conocido como referencia (dirección de memoria). Para utilizar las referencias se requiere de un elemento llamado nodo, el cual se estructura de la siguiente manera.





Morgado Jacome Eduardo #17211545

Bibliografía

Pablo López. (2006). Tema 2 – Memoria Dinámica., de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, Sitio web: http://www.lcc.uma.es/~lopez/lp2/apuntes/03-punteros/memoria_dinamica.pdf

Sin especificar. (2004). Unidad I: Introducción a las estructuras de datos., del Instituto Tecnológico de Piedras Negras, Sitio web: http://itpn.mx/recursosisc/3semestre/estructuradedatos/Unidad%20I.pdf

Sin especificar. (2008). Procesadores de lenguaje - Organización y gestión de la memoria., del repositorio de la Universidad Jaume I en Valencia., Sitio web: http://repositori.uji.es/xmlui/bitstream/handle/10234/5915/memoria.apun.pdf;jsessionid=43A22F67B56FB26CE935C49977F094B4?sequence=1

Morgado Jacome Eduardo #17211545