Estructura de Datos I: Práctica 1

Objetivo de la práctica

El objetivo de esta práctica es entender las diferencias entre los espacios de memoria *heap* y *stack*, los motivos de eficiencia de la reserva de memoria dinámica, la creación de un suprograma recursivo básico y el paso de parámetros a subprogramas.

Enunciado

Se pretende comprobar las limitaciones de los diferentes espacios de memoria principal (montículo/*heap* y pila/*stack*) mediante el uso de subprogramas recursivos que reserven en cada llamada un array de grandes dimensiones hasta que se desborde dicho espacio de memoria. Queremos conocer, de manera aproximada, cuánto espacio disponemos en cada una de las zonas de memoria.

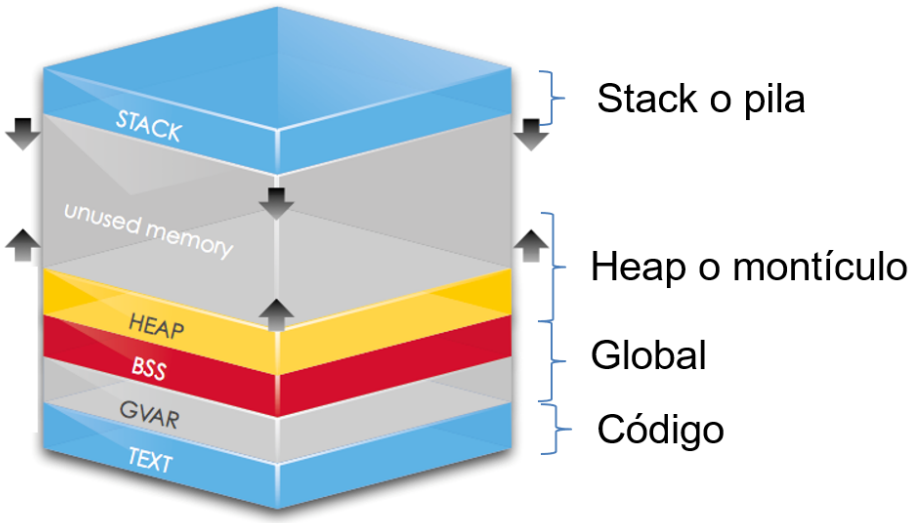

Imagen extraída de: www.sw-at.com

Para ello se harán 2 versiones de un subprograma, uno pasando por valor un array como parámetro (`subprogramaArray`), de tal manera que en cada llamada el sistema operativo reserve espacio para la copia del array en el stack de memoria; y otro pasando por valor un puntero con el que se reserve espacio en el heap para el array de grandes dimensiones mediante la llamada al operador `malloc` o `calloc` dentro del subprograma (`subprogramaPuntero`).

Para controlar el número de llamadas recursivas que se consiguen realizar antes de que se desborde la memoria, se pasará una segunda variable de tipo entero en cada subprograma inicializada al número de llamadas recursivas que se pretenden realizar (por ejemplo, empieza con 10), que se irá decrementando en la rama recursiva, y que alcanzará el caso base cuando dicha variable obtenga el valor de 0, devolviendo el control al programa principal. Se recuerda que un subprograma recursivo debe contener, al menos, una ramificación que distinga el cauce de ejecución de los casos recursivos respecto del caso base (o casos bases si hubiera varios), así como, al menos, una llamada recursiva con un parámetro modificado de manera que lo acerque al valor que debe obtener para llegar a un caso base y terminar la ejecución recursiva.

De esta manera, la llamada al subprograma que envía por valor un array de grandes dimensiones será algo parecido a:

```
subprogramaArray(contenedor, num);
```

siendo contenedor un array de, por ejemplo, 20.000 elementos enteros (ocupará del orden de $20.000 \cdot 4$ Bytes $\approx$ 80 KB) y `num` será un entero que indicará el número de llamadas recursivas (no iterativas, es decir, no debe haber bucles) que deben hacerse para llegar al caso base. El caso recursivo del subprograma simplemente contendrá una llamada recursiva decrementando el valor de `num` en una unidad.

Mientras que la llamada al subprograma que envía por valor un puntero capaz de reservar arrays de grandes dimensiones será algo parecido a:

```
subprogramaPuntero(puntContenedor, num);
```

siendo `puntContenedor` una variable de tipo puntero a array de los mismos 20.000 elementos enteros y, nuevamente, `num` será un entero que indicará el número de llamadas recursivas que deben hacerse para llegar al caso base. En cada llamada recursiva se reservará con el puntero un array del mismo tamaño que el anterior y, a continuación, se realizará la llamada recursiva decrementando el valor de `num` en una unidad. Tras la llamada recursiva, pero antes de salir del subprograma, deberíamos liberar la memoria reservada previamente con el puntero.

El programa terminará la ejecución cuando el valor con el que se inicialice la variable `num` sea suficientemente grande y genere tantas reservas de memoria que agoten el espacio asignado en la zona, pila de memoria para la versión de `subprogramaArray` y montículo para `subprogramaPuntero`.