Memoria dinámica

Parte 1: malloc, realloc y free

Justificación

Los variables y vectores en C ocupan un tamaño prefijado y este no puede ser modificado durante la ejecución del programa, de allí que surge la necesidad de utilizar algún mecanismo que permita reservar o liberar memoria dinámicamente.

Segmentos de memoria

Cada vez que se ejecuta cualquier programa, el mismo deberá pasar a memoria.

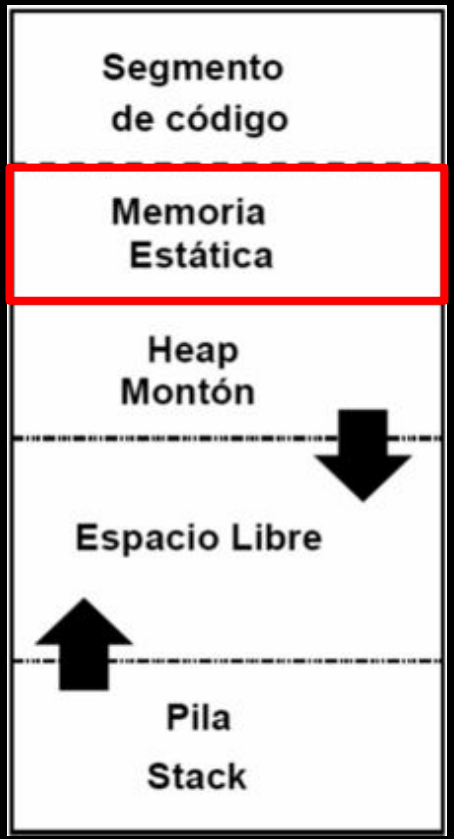
Los programas en memoria cuentan con distintos segmentos y por lo tanto los datos según su tipo serán almacenados en alguno de estos.

Segmento de código

En este segmento se guardan las instrucciones, en lenguaje máquina, de nuestro programa.

Segmento de Memoria estática

En este Segmento se guardan las variables globales del programa.



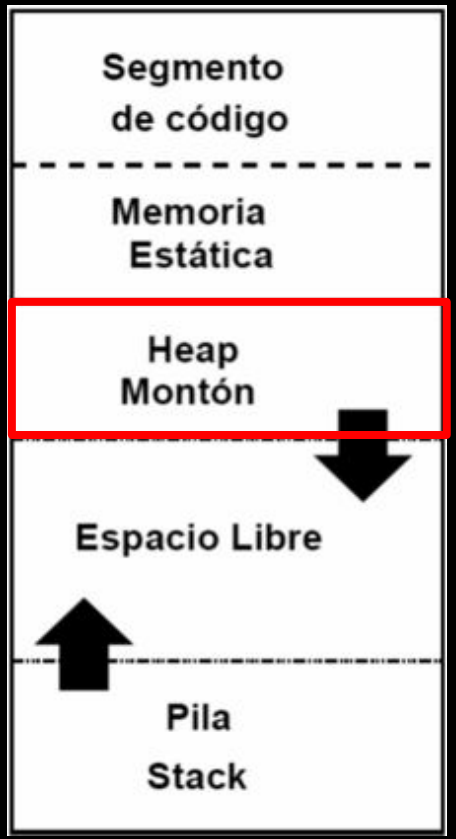
Segmento de Pila

Cada vez que se llama a una función entra en este segmento con toda su información y allí se guardan:

- Los llamados a las funciones
- Los parámetros de las funciones
- Las variables locales
- Otra información necesaria para el funcionamiento del programa.

Segmento de Heap

En este segmento se guardan las variables que han sido creadas dinámicamente en tiempo de ejecución.



Reserva dinámica de memoria

Para ello existen varias funciones estándares, de la biblioteca <stdlib.h>, nos centraremos en malloc.

```
void* malloc(unsigned int numBytes);
```

Devuelve un puntero al tipo de datos void (sin tipo). Dicho puntero puede ser asignado a una variable puntero de cualquier tipo mediante una conversión forzada de tipo de datos (casting).

Uso de malloc()

La función malloc(), devuelve un puntero nulo (NULL) si la reserva de memoria no puede realizarse, generalmente por falta de memoria disponible.

```
int *a;
a = (int *) malloc (sizeof(int));
if(a != NULL)
    *a = 22;
```

Uso de malloc()

```
int *a;
int i;
a = (int *) malloc (sizeof(int) * 20);
if(a != NULL)
{
    for(i=0; i<20;i++)
        *(a+i) = 0;
}
```

Reserva dinámica de memoria

El redimensionamiento dinámico de memoria intenta cambiar el tamaño de un bloque de memoria previamente asignado.

```
void* realloc(void* ptr, unsigned int numBytes);
```

Si el tamaño del bloque original no puede ser redimensionado, entonces 'realloc()' intentará asignar un nuevo bloque de memoria y copiará el contenido anterior.

Uso de realloc()

```
int *a;
int *pAux;
a = (int *) malloc (sizeof(int) * 20);

pAux = (int *) realloc (a , sizeof(int) * 200);

if(pAux != NULL)
    a = pAux;
```

Liberación dinámica de memoria

La memoria dinámica reservada es eliminada siempre al terminar la ejecución del programa por el propio sistema operativo. Sin embargo, durante la ejecución del programa puede ser interesante, e incluso necesario, proceder a liberar parte de la memoria reservada con anterioridad.

```
void free (void* ptr);
```

Uso de free()

```
int *a;
int *pAux;
a = (int *) malloc (sizeof(int) * 20);

free(a);
```