**MALLOC**

La función malloc sirve para solicitar un bloque de memoria del tamaño suministrado como parámetro. Devuelve un puntero a la zona de memoria concedida:

**void\* malloc (unsigned numero de bytes);**

El tamaño se especifica en bytes. Se garantiza que la zona de memoria concedida no está ocupada por ninguna otra variable ni otra zona devuelta por malloc.

Si malloc es incapaz de conceder el bloque (p.ej. no hay memoria suficiente), devuelve un puntero nulo.

Punteros void\*

La función malloc devuelve un puntero inespecífico, que no apunta a un tipo de datos determinado. En C, estos punteros sin tipo se declaran como void\*

Muchas funciones que devuelven direcciones de memoria utilizan los punteros void\*. Un puntero void\* puede convertirse a cualquier otra clase de puntero:

**Char\* ptr = (char\*) malloc (1000);**

Operador **sizeof**

El problema de malloc es conocer cuántos bytes se quieren reservar. Si se quiere reservar una zona para diez enteros, habrá que multiplicar diez por el tamaño de un entero.

El tamaño en bytes de un elemento de tipo T se obtiene con la expresión

**Sizeof (T)**

El tamaño de un char siempre es 1 (uno).

**Función free**

Cuando una zona de memoria reservada con malloc ya no se necesita, puede ser liberada mediante la función free.

void free (void\* ptr);

ptr es un puntero de cualquier tipo que apunta a un área de memoria reservada previamente con malloc.

Si ptr apunta a una zona de memoria indebida, los efectos pueden ser desastrosos, igual que si se libera dos veces la misma zona.

### Ejemplo de uso de malloc, free y sizeof

#include <stdlib.h>

int\* ptr; /\* puntero a enteros \*/

int\* ptr2; /\* otro puntero \*/

...

/\* Reserva hueco para 300 enteros \*/

ptr = (int\*)malloc ( 300\*sizeof(int) );

...

ptr[33] = 15; /\* trabaja con el área de memoria \*/

rellena de ceros (10,ptr); /\* otro ejemplo \*/

ptr2 = ptr + 15; /\* asignación a otro puntero \*/

/\* Finalmente, libera la zona de memoria \*/

free(ptr);

Obsérvese que hay que convertir el puntero **void\*** devuelto por **malloc** a un **int\***, para que no haya incompatibilidad de tipos.

REALLOC

Uso de realloc ()

El tamaño de la memoria asignada dinámicamente se puede cambiar utilizando realloc ().

void \*realloc (void \*ptr, size\_t size);

Realloc desasigna el objeto antiguo al que apunta ptr y devuelve un puntero a un nuevo objeto que tiene el tamaño especificado por tamaño. El contenido del nuevo objeto es idéntico al del objeto anterior antes de la des asignación, hasta el menor de los tamaños nuevo y antiguo. Todos los bytes en el nuevo objeto más allá del tamaño del objeto antiguo tienen valores indeterminados.

El punto a tener en cuenta es que realloc () solo debe usarse para memoria asignada dinámicamente. Si la memoria no está asignada dinámicamente, el comportamiento es indefinido.

Por ejemplo, el programa 1 demuestra el uso incorrecto de realloc () y el programa 2 demuestra el uso correcto de realloc ().

#include <stdio.h>

#include <stdlib.h>

int main()

{

   int \*ptr = (int \*)malloc(sizeof(int)\*2);

   int i;

   int \*ptr\_new;

   \*ptr = 10;

   \*(ptr + 1) = 20;

   ptr\_new = (int \*) realloc (ptr, sizeof (int)\*3);

   \*(ptr\_new + 2) = 30;

   for(i = 0; i < 3; i++)

       printf("%d ", \*(ptr\_new + i));

   getchar();

   return 0;

}