Guia 4 - ESD

Sizeof

El operador **sizeof** informa del tamaño de almacenamiento utilizado por cualquier objeto, sea un tipo básico o derivado.

La función sizeof es utilizada para obtener el tamaño en bytes de un tipo de valor como puede ser (int, float, char, double, short int...).

Es muy sencilla de utilizar simplemente pide un parámetro que corresponde al tipo de variable por ejemplo: sizefo(int); pero se puede utilizar pasándole de parámetro una variable declarada para allar su valor en bytes por ejemplo: int variable; sizeof(variable); en efecto devolvería el mismo valor que el ejemplo anterior.

Malloc

Los variables y vectores en C ocupan un tamaño prefijado, no pueden variarlo durante la ejecución del programa.

Por medio de punteros se puede reservar o liberar memoria dinámicamente, es decir, según se necesite. Para ello existen varias funciones estándares, de la biblioteca **<stdlib.h>**.

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);
```

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);
```