Nombre: Zarco Alvarenga, Harold José – ZA18009

Sizeof

La función sizeof, devuelve el tamaño en bytes que ocupa una variable o algún tipo de dato. Para reservar memoria se debe saber exactamente el número de bytes que ocupa cualquier estructura de datos. Tal y como se ha comentado con anterioridad, una peculiaridad del lenguaje C es que estos tamaños pueden variar de una plataforma a otra.

Por Ejemplo: En este caso las 2 instrucciones devuelven lo mismo:

```
int edad=20;
int tamanio;

/* Tamanio de una variable */
tamanio=sizeof(edad);

/* Tamanio de un tipo de dato*/
tamanio=sizeof(int);
```

Hay que ver por ejemplo que un int siempre ocupa 2 bytes, en este caso no importa si edad es 20, 100, -4 o lo que sea siempre ocupará 2 bytes, igual ocurre con el resto de tipos de datos. También podemos ocupar el sizeof para otros tipos de datos como estructuras o cualquier otro:

```
struct persona
{
 char nombre[50];
 long edad;
};


struct persona Perso;
int tamanio;

/* Tamanio de una variable */
tamanio=sizeof(Perso);

/* Tamanio de un tipo de dato*/
tamanio=sizeof(persona);
```

En este caso los 2 devuelven 54.

Malloc

Esta función asigna memoria dinámica, es decir, se solicita un nuevo bloque de memoria libre de la memoria para el programa.

Se requiere como parámetro la longitud del bloque de memoria que desea ocuparse, dicha longitud se define en bytes, si por ejemplo ponemos: "malloc(1)", reservaríamos un byte.

Normalmente la dirección de memoria retornada por la función malloc() se asigna a un puntero de un dato como se puede ver en el ejemplo.

```
int * mitabla;
int contador;

MAIN_PROGRAM_CDIV
BEGIN_PROGRAM
   //...
   mitabla = (int *) malloc(sizeof(int) * 1000); // Solicita 1000 posiciones de memoria

   for (contador = 0; contador < 1000; contador++) // Accede a los datos
      mitabla[contador] = -1;

   free(mitabla);          // Libera la memoria ocupada por la tabla

END_PROGRAM
```

Este pequeño ejemplo muestra como se puede definir un puntero a una tabla (con el puntero no se reserva espacio para los datos de la tabla), se debe solicitar memoria con la función malloc().

Una vez solicitada la memoria para la tabla de datos se accede a los mismos (se inicializan a -1, en el ejemplo), y se libera la memoria.

Los bloques de memoria que no se liberen con la función free(), no serán liberados automáticamente por el sistema al finalizar el programa, tenga especial cuidado al llamar esta función siempre debe liberar la memoria.

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);
```

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