-Malloc void *malloc(size_t size);

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 cómo 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.

-Calloc void *calloc(size_t nmemb, size_t size);

La función **calloc ()** se usa para asignar dinámicamente espacio de memoria e inicializarlo a 0. calloc () asigna dinámicamente num espacios consecutivos de tamaño en la memoria e inicializa cada byte a 0. Entonces, su resultado es que se asignan números * tamaño de bytes de espacio de memoria, y el valor de cada byte es 0.

[Valor de retorno] Devuelve la dirección apuntando a la memoria si la asignación es exitosa, o NULL si falla. Si el valor de size es 0, el valor de retorno será diferente dependiendo de la implementación de la biblioteca estándar. Puede o no ser NULL, pero el puntero devuelto no debe ser referenciado nuevamente.

Nota: El tipo de valor de retorno de la función es void *, void no significa que no haya un valor de retorno o que devuelva un puntero nulo, pero el tipo de puntero de retorno es desconocido. Por lo tanto, cuando se usa calloc (), generalmente se requiere una conversión de tipo para convertir el puntero void al tipo que queremos.

ejemplo_Calloc.c

```c
#include <stdio.h>
#include <stdlib.h>

int main ()
{
    int i,n;
    int * pData;
        printf ("El número de dígitos que se introducirán:");
    scanf ("%d",&i);
    pData = (int*) calloc (i,sizeof(int));
    if (pData==NULL) exit (1);
    for (n=0;n<i;n++)
    {
                printf ("Ingrese un número #% d:", n + 1);
       scanf ("%d",&pData[n]);
    }
        printf ("El número que ingresaste es:");
    for (n=0;n<i;n++) printf ("%d ",pData[n]);

    free (pData);
    system("pause");
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
El número de dígitos que se introducirán:5
Ingrese un número # 1:23
Ingrese un número # 2:67
Ingrese un número # 3:89
Ingrese un número # 4:56
Ingrese un número # 5:45
sh: 1: pause: not found
El número que ingresaste es:23 67 89 56 45 root@DESKTOP-I1MG0EA:/mnt/c/Users/Erubiel Tun/Documents/StructuredPrograming2A/unit2#
```

-Free void free(void *ptr);

Esta función es la encargada de liberar los bloques de memoria que, habiendo sido reservados mediante calloc(), malloc() o realloc() hayan dejado de ser útiles, pudiendo ser devueltos a la lista de bloques disponibles que mantiene el sistema operativo. Los bloques así liberados servirán para reservar otros bloques, de tamaños iguales o distintos al del bloque devuelto

El único argumento de esta función, **ptr**, denota el puntero del bloque que se quiera liberar. Este puntero no puede ser NULL ni tampoco puede señalar a un bloque que ya haya sido liberado (mediante una llamada previa a free()). Todo intento de realizar una de estas operaciones tendrá graves consecuencias.

Emparejado con la función malloc () para liberar la memoria dinámica solicitada por la función malloc. (Otro: para la oración de free§, si p es un puntero NULL, no importa cuántas veces free funcione en p. Si p no es un puntero NULL, entonces dos operaciones consecutivas en p harán que el programa se ejecute incorrectamente. .) **La función free no retorna ningún valor.**

```c
#include <string.h>
#include <stdio.h>
#include <alloc.h> //or #include <malloc.h>
int main(void)
{
    char *str;
    /* allocate memory for string */
    str = (char *)malloc(10);
    /* copy "Hello" to string */
    strcpy(str, "Hello");
    /* display string */
    printf("String is %s\n", str);
    /* free memory */
    free(str);
    str=NULL;
    return 0;
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
root@DESKTOP-I1MG0EA:/mnt/c/Users/Erubiel Tun/Documents/StructuredPrograming2A/unit2# gcc ejemplo_free.c -o ejemplo_free
ejemplo_free.c:3:10: fatal error: alloc.h: No such file or directory
 #include <alloc.h> //or #include <malloc.h>
          ^~~~~~~~~~
compilation terminated.
```