



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

Marco Antonio Martinez Quintana

*Profesor:*

Estructura de datos y Algoritmos 1

*Asignatura:*

*Grupo:*

16

*No de Práctica(s):*

Práctica 4

*Integrante(s):*

Tierrablanca Oviedo Evelyn

*No. de Equipo de  
cómputo empleado:*

*No. de Lista o Brigada:*

Segundo Semestre

*Semestre:*

*Fecha de entrega:*

*Observaciones:*

**CALIFICACIÓN:** \_\_\_\_\_

**Objetivo:**

Utilizarás funciones en lenguaje C que permiten reservar y almacenar información de manera dinámica (en tiempo de ejecución).

**Actividades:** f

Utilizar funciones para reservar memoria dinámica en lenguaje C.

Almacenar información en los elementos reservados con memoria dinámica.

**Introducción**

La memoria dinámica se refiere al espacio de almacenamiento que se reserva en tiempo de ejecución, debido a que su tamaño puede variar durante la ejecución del programa. El uso de memoria dinámica es necesario cuando a priori no se conoce el número de datos y/o elementos que se van a manejar.

**Memoria dinámica**

Dentro de la memoria RAM, la memoria reservada de forma dinámica está alojada en el heap o almacenamiento libre y la memoria estática (como los arreglos o las variables primitivas) en el stack o pila. La pila generalmente es una zona muy limitada. El heap, en cambio, en principio podría estar limitado por la cantidad de memoria disponible durante la ejecución del programa y el máximo de memoria que el sistema operativo permita direccionar a un proceso.

**Malloc**

La función malloc permite reservar un bloque de memoria de manera dinámica y devuelve un apuntador tipo void. Su sintaxis es la siguiente:

*void \*malloc(size\_t size);*

La función recibe como parámetro el número de bytes que se desean reservar. En caso de que no sea posible reservar el espacio en memoria, se devuelve el valor nulo (NULL).

**Free**

El almacenamiento en tiempo de ejecución se debe realizar de manera explícita, es decir, desde la aplicación se debe enviar la orden para reservar memoria. Así mismo, cuando la memoria ya no se utilice o cuando se termine el programa se debe liberar la memoria reservada. La función free permite liberar memoria que se reservó de manera dinámica.

Su sintaxis es la siguiente:

*void free(void \*ptr);*

El parámetro ptr es el apuntador al inicio de la memoria que se desea liberar. Si el apuntador es nulo la función no hace nada.

**Calloc**

La función calloc funciona de manera similar a la función malloc pero, además de reservar memoria en tiempo real, inicializa la memoria reservada con 0. Su sintaxis es la siguiente:

*void \*calloc (size\_t nelem, size\_t size);*

El parámetro nelem indica el número de elementos que se van a reservar y size indica el tamaño de cada elemento.

**Realloc**

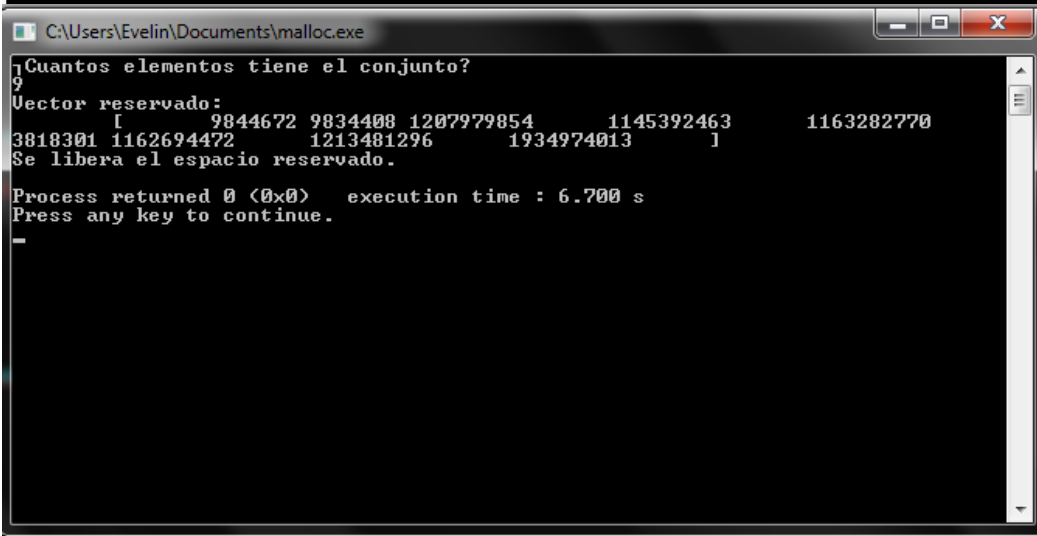
La función realloc permite redimensionar el espacio asignado previamente de forma dinámica, es decir, permite aumentar el tamaño de la memoria reservada de manera dinámica. Su sintaxis es la siguiente:

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

Donde ptr es el apuntador que se va a redimensionar y size el nuevo tamaño, en bytes, que se desea aumentar al conjunto. Si el apuntador que se desea redimensionar tiene el valor nulo, la función actúa como la función malloc. Si la reasignación no se pudo realizar, la función devuelve un apuntador a nulo, dejando intacto el apuntador que se pasa como parámetro (el espacio reservado previamente).

## Malloc

```
#include <stdio.h>
#include <stdlib.h>
int main () {
    int *arreglo, num, cont; //variables de tipo entero
    printf("¿Cuántos elementos tiene el conjunto?\n");
    scanf("%d", &num);
    arreglo = (int *)malloc (num * sizeof(int)); //arreglo de tipo entero para reservar espacio en la memoria
    if (arreglo != NULL) { //condicional para el arreglo en caso de que el espacio no esté vacío
        printf("Vector reservado:\n\t");
        for (cont=0 ; cont<num ; cont++){ //un ciclo for para contar y reservar los espacios que el usuario indica
            printf("\t%d", *(arreglo+cont)); //imprime el espacio del arreglo
        }
        printf("\n");
        printf("Se libera el espacio reservado.\n");
        free(arreglo); //libera el espacio reservado
    }
    return 0;
}
```



```
C:\Users\Evelin\Documents\malloc.exe
¿Cuántos elementos tiene el conjunto?
9
Vector reservado:
1 9844672 9834408 1207979854 1145392463 1163282770
3818301 1162694472 1213481296 1934974013 1
Se libera el espacio reservado.

Process returned 0 (0x0) execution time : 6.700 s
Press any key to continue.
```

## Calloc

```
#include <stdio.h>
#include <stdlib.h>
int main () {
    int *arreglo, num, cont; //variables de tipo entero
    printf("¿Cuántos elementos tiene el conjunto?\n");
    scanf("%d", &num);
    arreglo = (int *)calloc (num, sizeof(int)); //arreglo para reservar el espacio en la memoria e inicializarlo en cero
    if (arreglo != NULL) { //condicional si el arreglo no está vacío
        printf("Vector reservado:\n\t");
        for (cont=0 ; cont<num ; cont++){ //ciclo for para contar y reservar los espacios que el usuario indica
            printf("\t%d", *(arreglo+cont)); //imprime el espacio del arreglo
        }
        printf("\n");
        printf("Se libera el espacio reservado.\n");
        free(arreglo); //libera el espacio reservado
    }
    return 0;
}
```

```
C:\Users\Evelin\Documents\calloc.exe
¿Cuántos elementos tiene el conjunto?
4
Vector reservado:
[ 0 0 0 0 1]
Se libera el espacio reservado.
Process returned 0 (0x0)   execution time : 2.794 s
Press any key to continue.
```

## Realloc

```
#include <stdio.h>
#include <stdlib.h>
int main () {
    int *arreglo, *arreglo2, num, cont;
    printf("¿Cuántos elementos tiene el conjunto?\n");
    scanf("%d", &num);
    arreglo = (int *)malloc (num * sizeof(int));
    if (arreglo != NULL) {
        for (cont=0 ; cont < num ; cont++){
            printf("Inserte el elemento %d del conjunto.\n", cont+1);
            scanf("%d", (arreglo+cont));
        }
        printf("Vector insertado:\n\t[");
        for (cont=0 ; cont < num ; cont++){
            printf("\t%d", *(arreglo+cont));
        }
        printf("\t]\n");
        printf("Aumentando el tamaño del conjunto al doble.\n");
        num *= 2;
        arreglo2 = (int *)realloc (arreglo, num*sizeof(int));
        if (arreglo2 != NULL) {
            arreglo = arreglo2;
            for (; cont < num ; cont++){
                printf("Inserte el elemento %d del conjunto.\n", cont+1);
                scanf("%d", (arreglo2+cont));
            }
            printf("Vector insertado:\n\t[");
            for (cont=0 ; cont < num ; cont++){
                printf("\t%d", *(arreglo2+cont));
            }
            printf("\t]\n");
        }
        free (arreglo);
    }
    return 0;
}
```

```
C:\Users\Evelin\Documents\realloc.exe
1 Cuantos elementos tiene el conjunto?
5
2 Inserte el elemento 1 del conjunto.
2
3 Inserte el elemento 2 del conjunto.
8
3 Inserte el elemento 3 del conjunto.
3
3 Inserte el elemento 4 del conjunto.
3
1 Inserte el elemento 5 del conjunto.
1
Vector insertado:
[      2      8      3      3      1      1
Aumentando el tamaño del conjunto al doble.
3 Inserte el elemento 6 del conjunto.
3
4 Inserte el elemento 7 del conjunto.
4
2 Inserte el elemento 8 del conjunto.
2
1 Inserte el elemento 9 del conjunto.
1
3 Inserte el elemento 10 del conjunto.
3
Vector insertado:
[      2      8      3      3      1      3      4      2
1      3      1
Process returned 0 (0x0)   execution time : 16.477 s
Press any key to continue.
```