	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	02
		Página	1/8
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Laboratorios de computación salas A y B

**Profesor:** Marco Antonio Martínez

**Asignatura:** Estructura de datos y algoritmos 1

**Grupo:** 17

**No de Práctica(s):** 3. Almacenamiento en tiempo de ejecución

**Integrante(s):**

Pimentel Escobar Alondra Valeria


**No. de Equipo de cómputo empleado:**

**Semestre:** 2020-2

**Fecha de entrega:** 01/03/2020

**Observaciones:**

**CALIFICACIÓN:** \_\_\_\_\_

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	02
		Página	2/8
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Objetivo:

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

## 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.

## Desarrollo

### malloc


(Muestra datos basura en la memoria o residuos de programas anteriores)

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  int main (){
4      int *arreglo, num, cont;
5      printf("¿Cuántos elementos tiene el conjunto?\n");
6      scanf("%d",&num);
7      arreglo = (int *)malloc (num * sizeof(int));
8      if (arreglo!=NULL) {
9          printf("Vector reservado:\n\t"); for (cont=0 ; cont<num ; cont++){
10             }
11             printf("\t%d",*(arreglo+cont));
12             printf("\t]\n");
13             printf("Se libera el espacio reservado.\n");
14             free(arreglo);
15         }
16         return 0;
17     }

```

- Línea 4: Declaración de variables.

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	02
		Página	3/8
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

- Línea 5: pide la medida del vector
- Línea 6: se lee la variable
- Línea 7: malloc reserva el espacio que utilizara el vector
- Línea 8: aquí se pone una condición la cual dice que si el vector el cero lo cual quiere decir que es un vector nulo hará que no entre al ciclo ya que no se reservara memoria.
- Línea 14: libera la memoria reservada


Ejemplos:

```
¿Cuantos elementos tiene el conjunto?
0
Vector reservado:
      [      ]
Se libera el espacio reservado.
```

- En este ejemplo no se imprime nada en el vector ya que no está reservando memoria.

```
¿Cuantos elementos tiene el conjunto?
3
Vector reservado:
      [ 7536832 7545144 0 ]
Se libera el espacio reservado.
```

- Manda a imprimir los datos que se encuentran en esa memoria reservada y los datos que manda a imprimir es basura.

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	02
		Página	4/8
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

```

¿Cuántos elementos tiene el conjunto?
1
Vector reservado:
      [      11153760      ]
Se libera el espacio reservado.

```


- En este ejemplo pasa lo mismo que en el anterior.  
**Calloc**

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  int main (){
4      int *arreglo, num, cont;
5      printf("¿Cuántos elementos tiene el conjunto?\n");
6      scanf("%d",&num);
7      arreglo = (int *)calloc (num, sizeof(int));
8      if (arreglo!=NULL) {
9          printf("Vector reservado:\n\t["); for (cont=0 ; cont<num ; cont++){
10             }
11             printf("\t%d",*(arreglo+cont));
12             printf("\t]\n");
13             printf("Se libera el espacio reservado.\n");
14             free(arreglo);
15         }
16         return 0;
17     }

```

- Línea 4: Declaración de variables.
- Línea 5: pide la medida del vector
- Línea 6: se lee la variable
- Línea 7: calloc reserva el espacio que utilizara el vector y lo limpia
- Línea 8: aquí se pone una condición la cual dice que si el vector el cero lo cual quiere decir que es un vector nulo hará que no entre al ciclo ya que no se reservara memoria.
- Línea 14: libera la memoria reservada

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	02
		Página	5/8
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

### Ejemplos:

- En las siguientes imágenes calloc hizo su función de limpiar la memoria reservada por lo cual mandará a imprimir 0 en los vectores.

```

└─ Cuantos elementos tiene el conjunto?
3
Vector reservado:
      [      0      0      0      ]
Se libera el espacio reservado.

```

```


└─ Cuantos elementos tiene el conjunto?
5
Vector reservado:
      [      0      0      0      0      0      ]
Se libera el espacio reservado.

```

```

└─ Cuantos elementos tiene el conjunto?
1
Vector reservado:
      [      0      ]
Se libera el espacio reservado.

```

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	02
		Página	6/8
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Realloc

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  int main () {
4      int *arreglo, *arreglo2, num, cont;
5      printf(";Cuántos elementos tiene el conjunto?\n");
6      scanf("%d",&num);
7      arreglo = (int *)malloc (num * sizeof(int));
8      if (arreglo!=NULL) {
9          for (cont=0 ; cont < num ; cont++){
10             printf("Inserte el elemento %d del conjunto.\n",cont+1);
11             scanf("%d", (arreglo+cont));
12         }
13         printf("Vector insertado:\n\t[");
14         for (cont=0 ; cont < num ; cont++){
15             printf("\t%d",*(arreglo+cont));
16         }
17         printf("\t]\n");

```

- Línea 4: declaración de variables en este caso usaremos dos apuntadores.
- Línea 5: pide la dimensión del vector.
- Línea 6: se lee la dimensión del vector.
- Línea 7: malloc aparta la memoria y la limpia.
- Línea 8: aquí se encuentra una condición la cual dice que si la dimensión del vector diferente 0 ósea un vector nulo entra y si es 0 no entra.
- Línea 9: en este ciclo nos estará pidiendo los datos que almacenara el vector
- Línea 10: aquí los pide los datos
- Línea 11: aquí los lee y los almacena
- Línea 14: en este ciclo te manda a imprimir los valores del vector



## Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I

Código:

MADO-19

Versión:

02

Página

7/8

Sección ISO

8.3

Fecha de  
emisión

20 de enero de 2017


Facultad de Ingeniería

Área/Departamento:  
Laboratorio de computación salas A y B

La impresión de este documento es una copia no controlada

```
17     printf("\tj\n");
18     printf("Aumentando el tamaño del conjunto al doble.\n");
19     num *= 2;
20     arreglo2 = (int *)realloc (arreglo,num*sizeof(int));
21     if (arreglo2 != NULL) {
22         arreglo = arreglo2;
23         for (; cont < num ; cont++){
24             printf("Inserte el elemento %d del conjunto.\n",cont+1);
25             scanf("%d", (arreglo2+cont));
26         }
27         printf("Vector insertado:\n\t[");
28         for (cont=0 ; cont < num ; cont++){
29             printf("\t%d",*(arreglo2+cont));
30         }
31         printf("\tj\n");
32     }
33     printf("Se libera el espacio reservado.\n");
34     free (arreglo);
35 }
36 return 0;
37 }
```

- Línea 20: con realloc estamos redimensionando el espacio del vector lo cual nos da a entender que este nos ayuda a expandir la memoria reservada.
- Línea 21: aquí se encuentra una condición la cual dice que si la dimensión del vector es 0 no entra.
- Línea 23: en este ciclo pide los datos que hacen falta ya que la dimensión del vector aumento al doble.
- Línea 25: aquí lo lee y los almacena.
- Línea 28: en este ciclo te manda a imprimir los valores del vector ya redimensionado.
- Línea 33: se libera el espacio reservado para el vector.

	<b>Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I</b>	Código:	MADO-19
		Versión:	02
		Página	8/8
		Sección ISO	8.3
		Fecha de emisión	20 de enero de 2017
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Ejemplo:

```

¿Cuántos elementos tiene el conjunto?
2
Inserte el elemento 1 del conjunto.
1
Inserte el elemento 2 del conjunto.
2
Vector insertado:
[ 1 2 ]
Aumentando el tamaño del conjunto al doble.
Inserte el elemento 3 del conjunto.
3
Inserte el elemento 4 del conjunto.
4
Vector insertado:
[ 1 2 3 4 ]
Se libera el espacio reservado.

```

- La primera dimensión del vector es el 2.
- Le damos valores.
- Cuando realloc hace su función la dimensión aumenta al doble.
- Nos pide dos valores más ya que aumento a 4 la dimensión del vector.
- Finalmente nos imprime el vector y libera el espacio que se reservó.

### Conclusión:

Podemos observar la importancia de reservar memoria y como realizarlo de un modo óptimo ya que al final de utilizarla la liberamos, esta parte es importante porque de no liberarla podríamos causar que nuestro programa tenga dificultades.

### Bibliografía:

El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.

Ariel Rodríguez (2010). How knowing C and C++ can help you write better iPhone apps, part 1. [Figura 1]. Consulta: Enero de 2016. Disponible en:  
<http://akosma.com/2010/10/11/how-knowing-c-and-c-can-help-you-write-betteriphone-apps-part-1/>