OBJETIVO: utilizarás funciones en lenguaje c que permiten reservar y almacenar información de manera dinámica (en tiempo de ejecución).

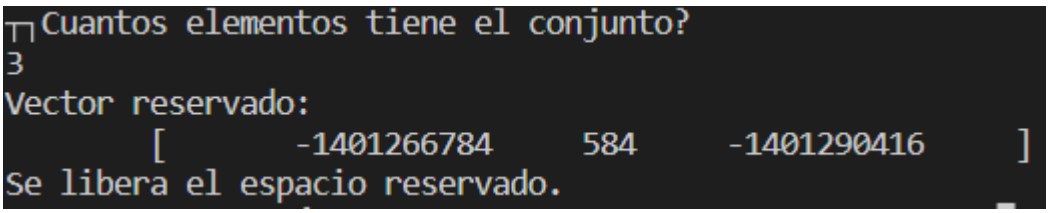
OBJETIVO DE CLASE: Comprender el uso básico de apuntadores aplicado al almacenamiento en tiempo de ejecución, así como las funciones esenciales para el trabajo de los mismos

Ejemplos de la guía

- Código (malloc)

El programa se declara una variable que será un contador, un apuntador para declarar un arreglo con memoria dinámica, y una variable para guardar el número de elementos del arreglo. Primero se pregunta por un número entero para que sea el número de elementos de un arreglo reservado con memoria dinámica. Este código cuenta con un condicional if en caso de que no se consiga reservar la memoria para el arreglo, y la función malloc regrese un apuntador NULL.

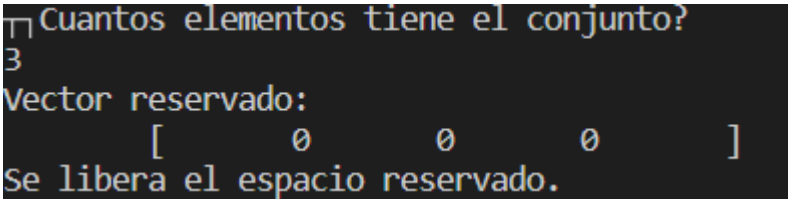
En pantalla solo se imprimen todos los elementos del arreglo, los cuales al solo haberse reservado, no contienen nada en específico sino que “basura”. Como se puede ver en el siguiente ejemplo de ejecución con un arreglo de 3 elementos. También se incluye un mensaje de que el espacio reservado fue liberado.



Finalmente se libera el espacio ocupando la función free(). Este ejemplo muestra la forma de apartar memoria con la función malloc(), y las característica de que no limpia los espacios de memoria que aparta.

- Código (calloc)

Este arreglo funciona casi de igual manera que el ejemplo anterior. Declara las mismas variables para el contador, número de elementos y apuntador para el arreglo; aunque en lugar de apartar la memoria con la función malloc(), ocupa calloc(), resultando en un arreglo como en el ejemplo anterior, pero con sus elementos igualados a “0”, incluyendo también en la impresión, un mensaje sobre la liberación del espacio reservado.

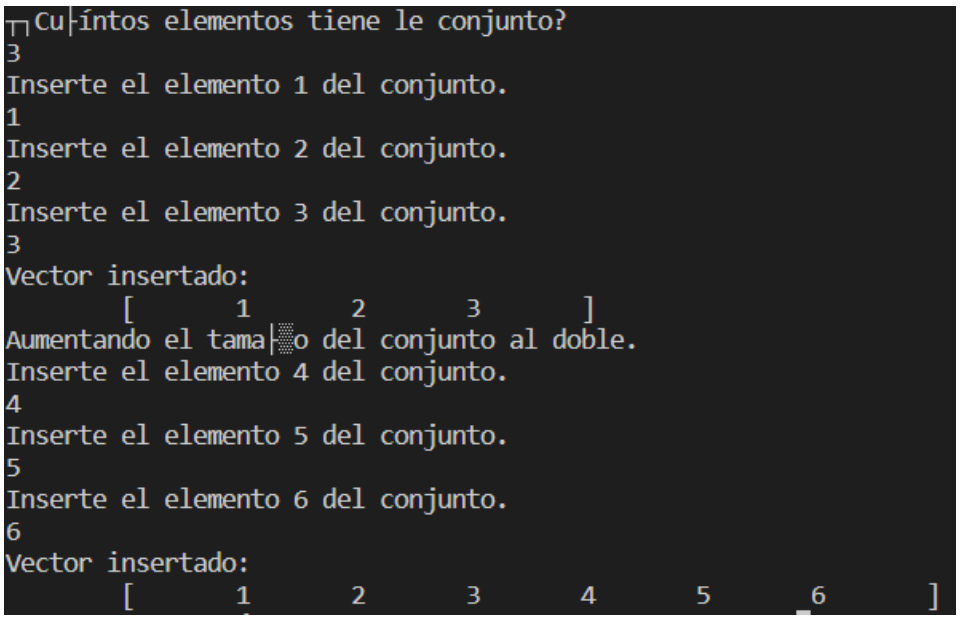


Este código comparte casi todos los elementos con el ejemplo anterior, pero con la diferencia del uso del calloc() para que los elementos del arreglo se limpien e igualen a “0”; mostrando el uso de calloc() y su diferencia con malloc().

- Código (realloc)

En este código también se declaran variables de tipo entero para guardar el número de elementos del arreglo, para un contador, un apuntador para el arreglo, además de un apuntador adicional. El código solicita el tamaño del arreglo con scanf(), y reserva el espacio para un arreglo de ese tamaño con malloc(); después pide elementos para llenar el arreglo. El siguiente paso es duplicar el tamaño del arreglo, multiplicando por 2 la variable con el tamaño del arreglo y ocupandola para redimensionar el arreglo ocupando la función realloc y el segundo apuntador declarado. Finalmente se pide al usuario llenar el resto de los elementos del arreglo. El programa termina imprimiendo los elementos del arreglo y liberando el espacio con la función free().

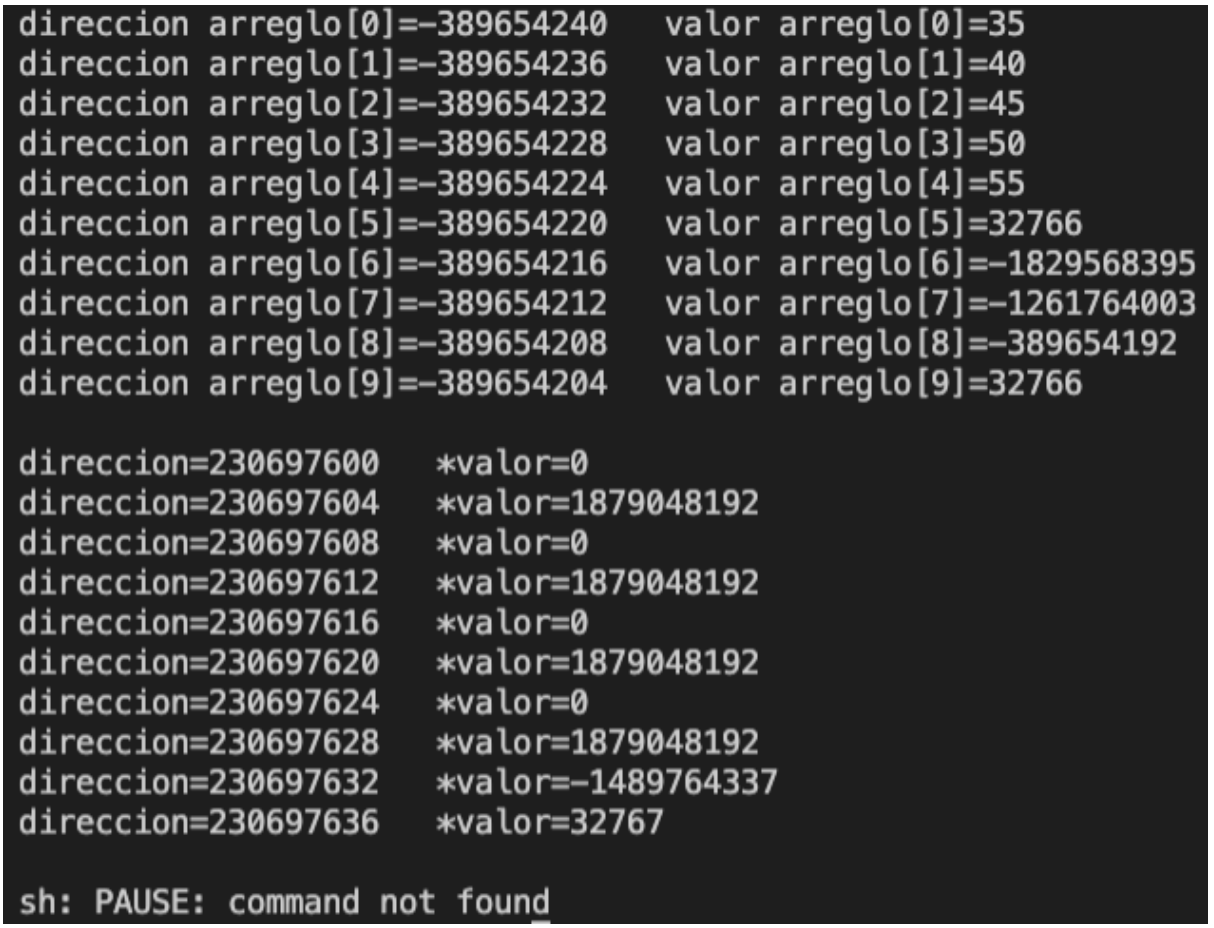
Este ejemplo es bueno para mostrar el uso de realloc() para redimensionar un arreglo de memoria dinámica, y la función que hacen los apuntadores que se ocupan al momento de redimensionar, así como de la liberación de la memoria apartada.


Cuántos elementos tiene le conjunto?
3
Inserte el elemento 1 del conjunto.
1
Inserte el elemento 2 del conjunto.
2
Inserte el elemento 3 del conjunto.
3
Vector insertado:
[1 2 3]
Aumentando el tamaño del conjunto al doble.
Inserte el elemento 4 del conjunto.
4
Inserte el elemento 5 del conjunto.
5
Inserte el elemento 6 del conjunto.
6
Vector insertado:
[1 2 3 4 5 6]

Ejercicios

- Ejercicio 1

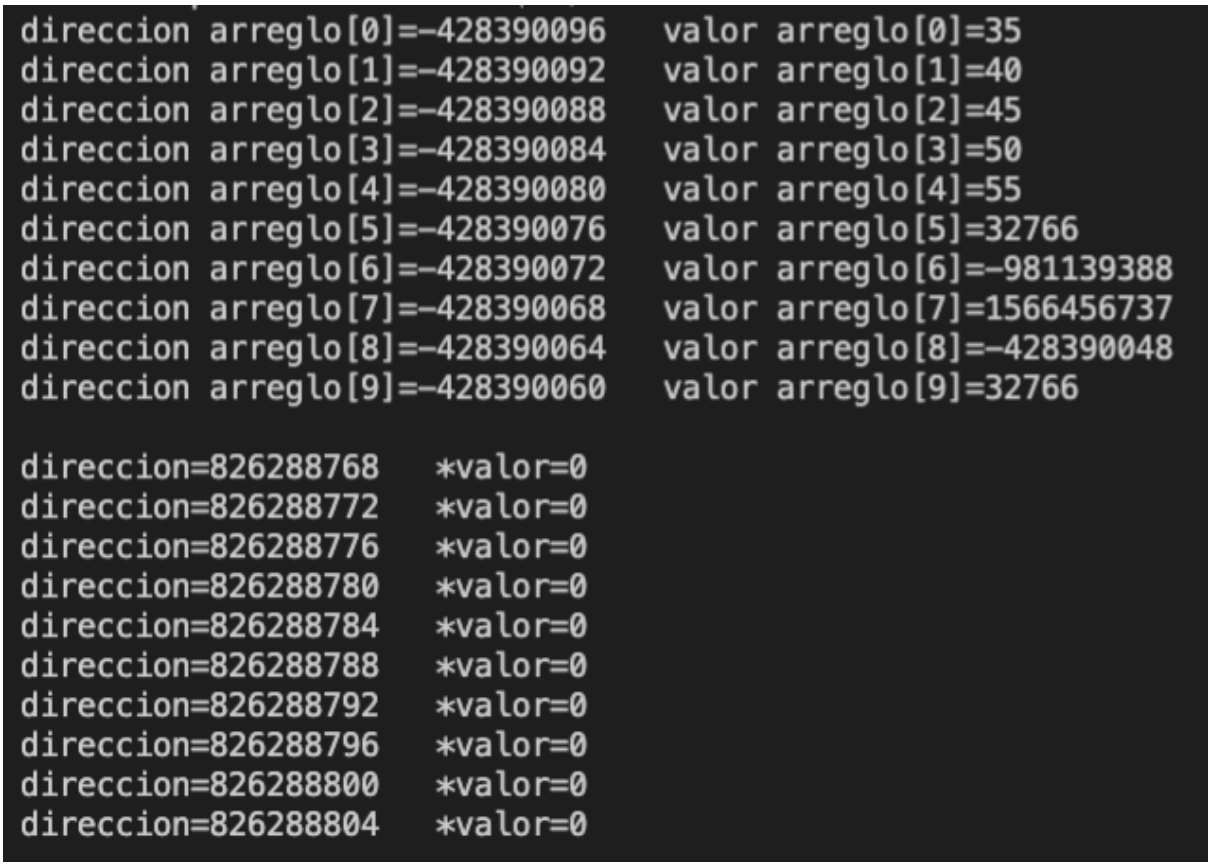
a) Captura



En el primer bloque se observa las direcciones del arreglo, que están contiguas; seguidas del valor que hay en esas localidades. También se ve que una vez se llega a localidades fuera del arreglo declarado y definido, se imprimen valores no relacionados al arreglo, “basura”.

Del mismo modo se muestran las direcciones, siendo la primera la de uno de los apuntadores declarados, y sus respectivos valores que son “basura”.

b)Ejecución



La principal diferencia de esta ejecución con la anterior es que los valores del arreglo declarado con memoria dinámica fueron igualados a 0 por la función calloc().

c)

```
direccion arreglo[0]=-378234576   valor arreglo[0]=35
direccion arreglo[1]=-378234572   valor arreglo[1]=40
direccion arreglo[2]=-378234568   valor arreglo[2]=45
direccion arreglo[3]=-378234564   valor arreglo[3]=50
direccion arreglo[4]=-378234560   valor arreglo[4]=55
direccion arreglo[5]=-378234556   valor arreglo[5]=32766
direccion arreglo[6]=-378234552   valor arreglo[6]=-1846214642
direccion arreglo[7]=-378234548   valor arreglo[7]=-2107447220
direccion arreglo[8]=-378234544   valor arreglo[8]=-378234528
direccion arreglo[9]=-378234540   valor arreglo[9]=32766

direccion=-205510016   *valor=3
direccion=-205510012   *valor=6
direccion=-205510008   *valor=9
direccion=-205510004   *valor=12
direccion=-205510000   *valor=15
direccion=-205509996   *valor=18
direccion=-205509992   *valor=21
direccion=-205509988   *valor=24
direccion=-205509984   *valor=27
direccion=-205509980   *valor=30
```

d)

ptr

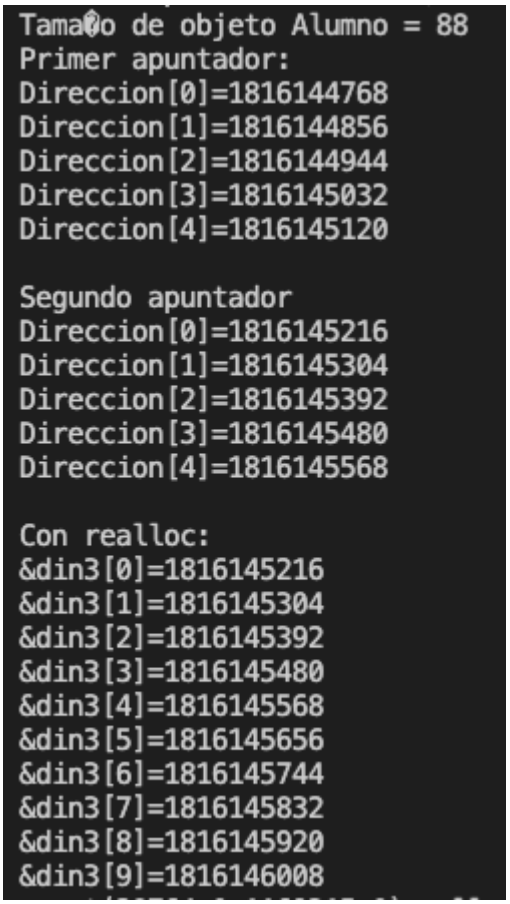
```
direccion=-1438635136   *valor=3
direccion=-1438635132   *valor=6
direccion=-1438635128   *valor=9
direccion=-1438635124   *valor=12
direccion=-1438635120   *valor=15
direccion=-1438635116   *valor=18
direccion=-1438635112   *valor=21
direccion=-1438635108   *valor=24
direccion=-1438635104   *valor=27
direccion=-1438635100   *valor=30
direccion=-1438635096   *valor=0
direccion=-1438635092   *valor=0
direccion=-1438635088   *valor=0
direccion=-1438635084   *valor=0
direccion=-1438635080   *valor=0
direccion=-1438635076   *valor=0
direccion=-1438635072   *valor=0
direccion=-1438635068   *valor=0
direccion=-1438635064   *valor=0
direccion=-1438635060   *valor=0
direccion=-1438635056   *valor=0
direccion=-1438635052   *valor=0
direccion=-1438635048   *valor=0
direccion=-1438635044   *valor=0
direccion=-1438635040   *valor=0
```

ptr3

```
direccion=-1438635136   *valor=3
direccion=-1438635132   *valor=6
direccion=-1438635128   *valor=9
direccion=-1438635124   *valor=12
direccion=-1438635120   *valor=15
direccion=-1438635116   *valor=18
direccion=-1438635112   *valor=21
direccion=-1438635108   *valor=24
direccion=-1438635104   *valor=27
direccion=-1438635100   *valor=30
direccion=-1438635096   *valor=0
direccion=-1438635092   *valor=0
direccion=-1438635088   *valor=0
direccion=-1438635084   *valor=0
direccion=-1438635080   *valor=0
direccion=-1438635076   *valor=0
direccion=-1438635072   *valor=0
direccion=-1438635068   *valor=0
direccion=-1438635064   *valor=0
direccion=-1438635060   *valor=0
direccion=-1438635056   *valor=0
direccion=-1438635052   *valor=0
direccion=-1438635048   *valor=0
direccion=-1438635044   *valor=0
direccion=-1438635040   *valor=0
```

Como se puede observar los datos se conservaron para cada apuntador, además de que no hubo diferencias al ocupar realloc con cualquiera de los dos apuntadores, resultando en que ambos hacen referencia a las mismas localidades de memoria.

- Ejercicio 2



a)
Los resultados que se muestran en pantalla son el tamaño del objeto alumno, resultante de sumar el tamaño de todos las propiedades del objeto.

También se imprimen las direcciones de los arreglos de objetos “alumno” declarados con memoria dinámica, los cuales van acorde al tamaño del mismo, siendo que cada elemento abarca 88 bytes. Esto también incluye al arreglo redimensionado con realloc, que está asociada a las mismas localidades que el segundo apuntador.

b)
La estructura alumno tiene un tamaño de 88 bytes, los cuales se obtienen al sumar el tamaño de cada una de sus propiedades, incluyendo las dirección.