	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Edgar Tista Garcia

Asignatura: Estructuras de Datos y Algoritmos I

Grupo: 2

No. de práctica(s): 4

Integrante(s): Álvarez López Carlos Manuel

No. de lista o brigada: 2

Semestre: 2023-2

Fecha de entrega: Sábado 18 de Marzo

Observaciones:

CALIFICACIÓN: _____

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

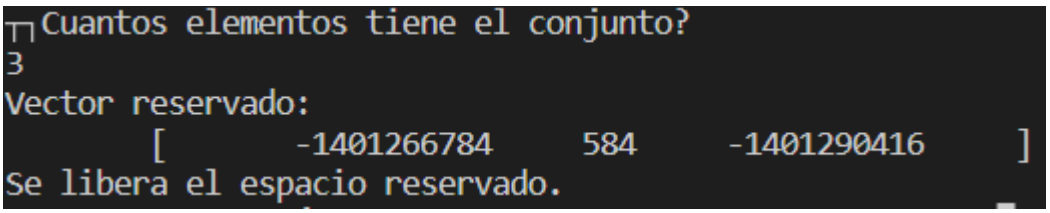
OBJETIVO DE CLASE: Comprender el uso básico de apuntadores aplicado al almacenamiento en tiempo de ejecución, así como las funciones esenciales para el trabajo de los mismos

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.

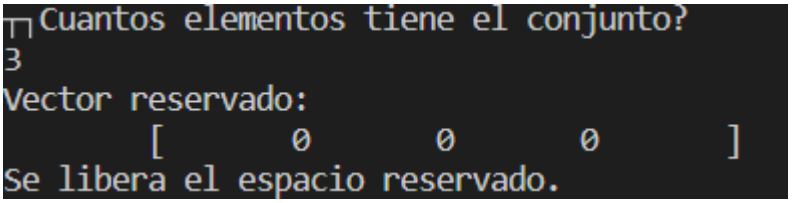
En pantalla solo se imprimen todos los elementos del arreglo, los cuales al solo haberse reservado, no contienen nada en específico sino que “basura”. Como se puede ver en el siguiente ejemplo de ejecución con un arreglo de 3 elementos. También se incluye un mensaje de que el espacio reservado fue liberado.



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.

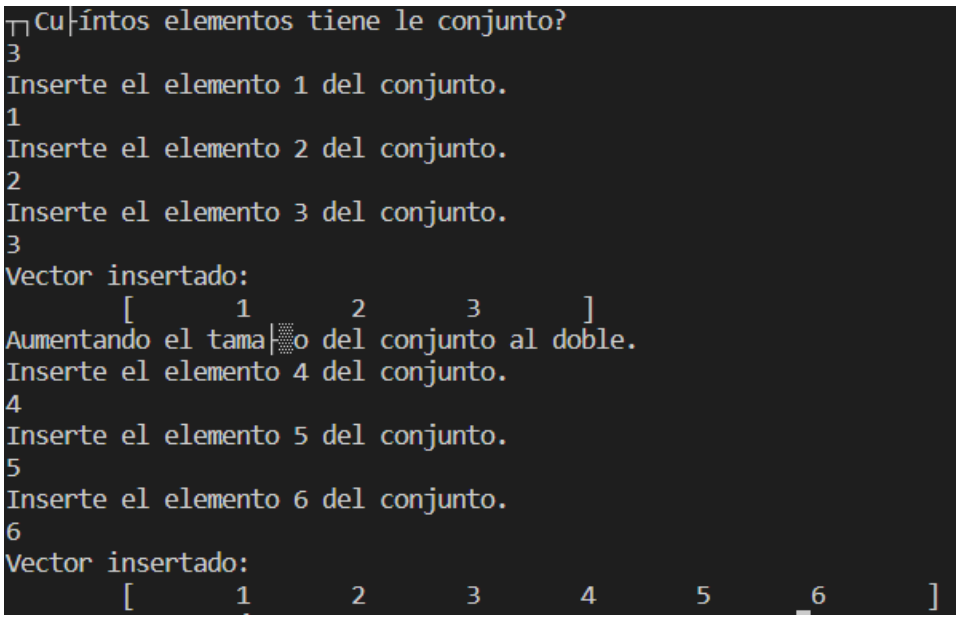


Este código comparte casi todos los elementos con el ejemplo anterior, pero con la diferencia del uso del calloc() para que los elementos del arreglo se limpien e igualen a “0”; mostrando el uso de calloc() y su diferencia con malloc().

- 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().

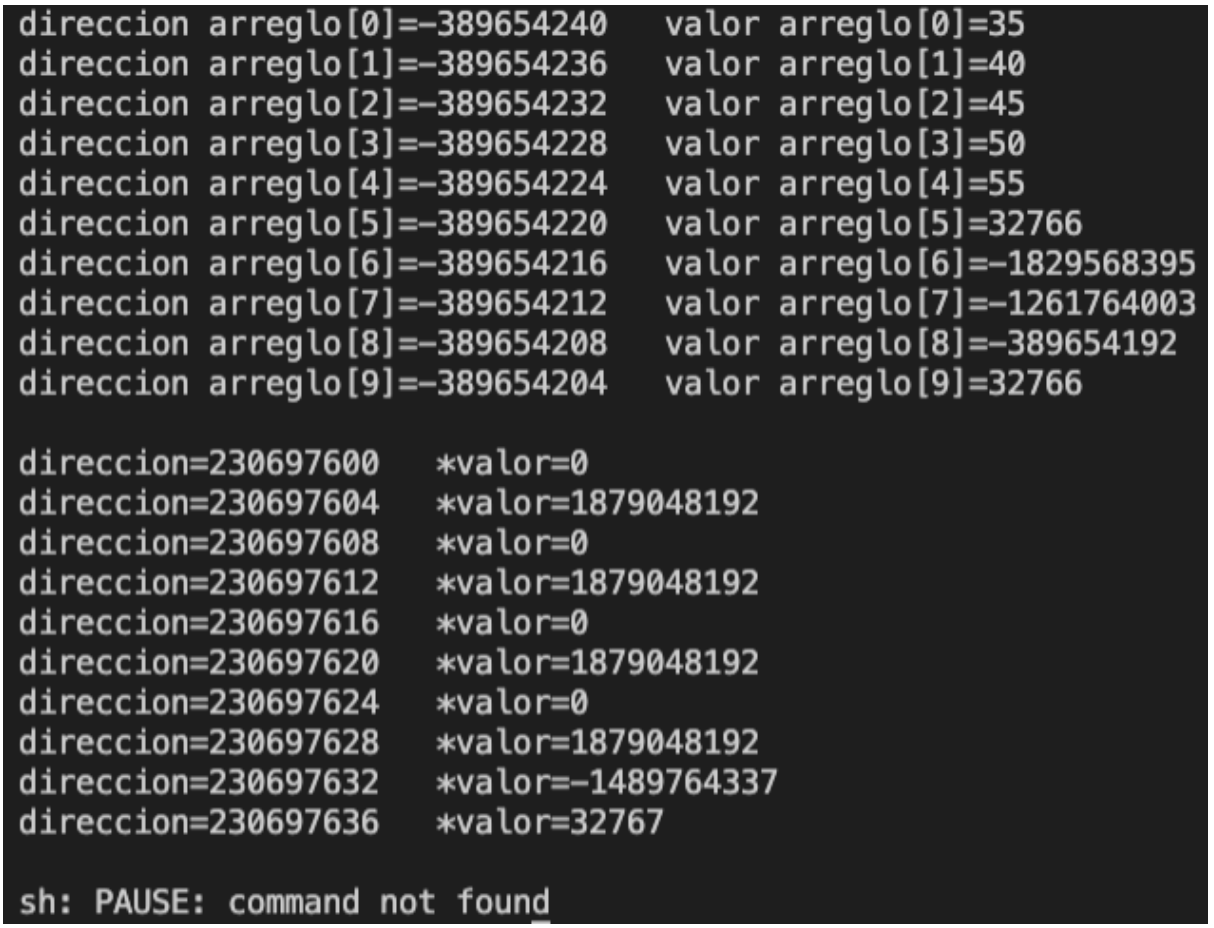
Este ejemplo es bueno para mostrar el uso de realloc() para redimensionar un arreglo de memoria dinámica, y la función que hacen los apuntadores que se ocupan al momento de redimensionar, así como de la liberación de la memoria apartada.


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**

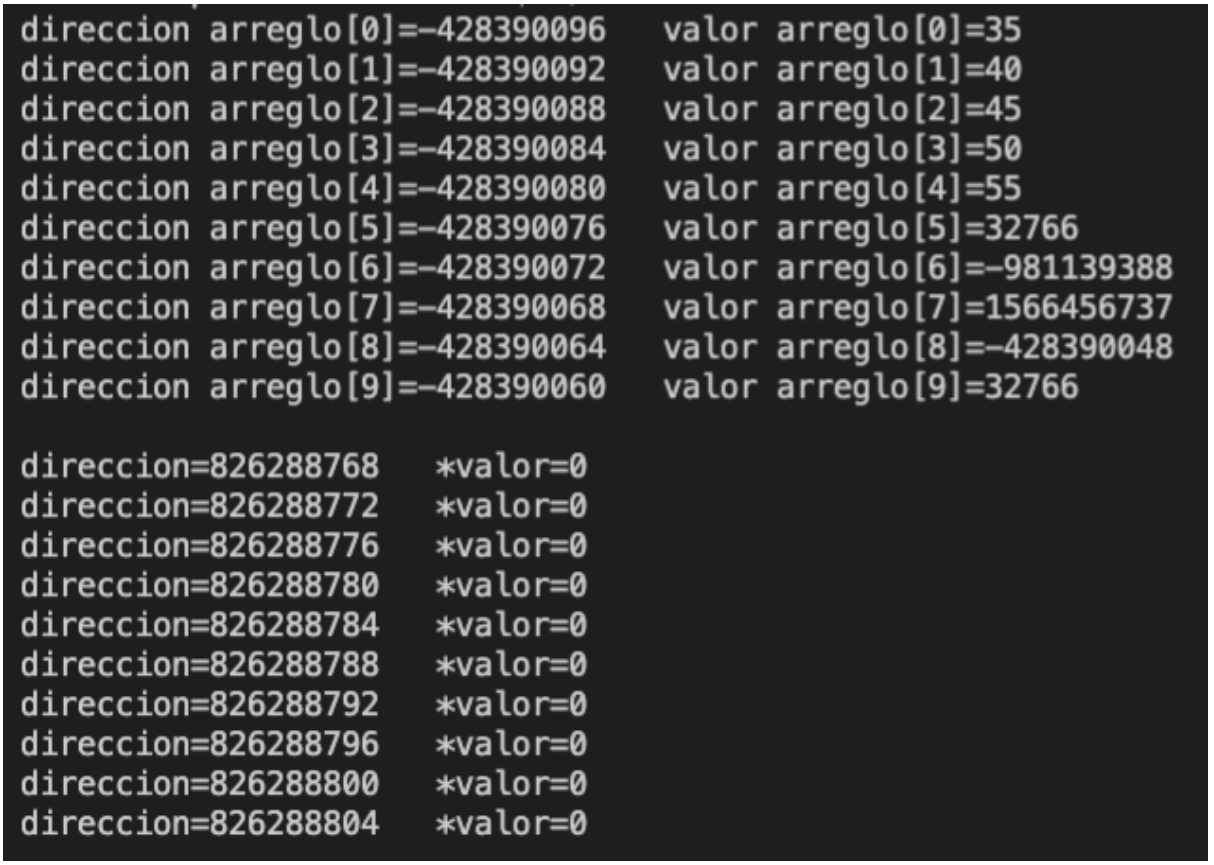
a) Captura la pantalla con la ejecución del programa y explica de manera detallada, los resultados que se observan.



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”.

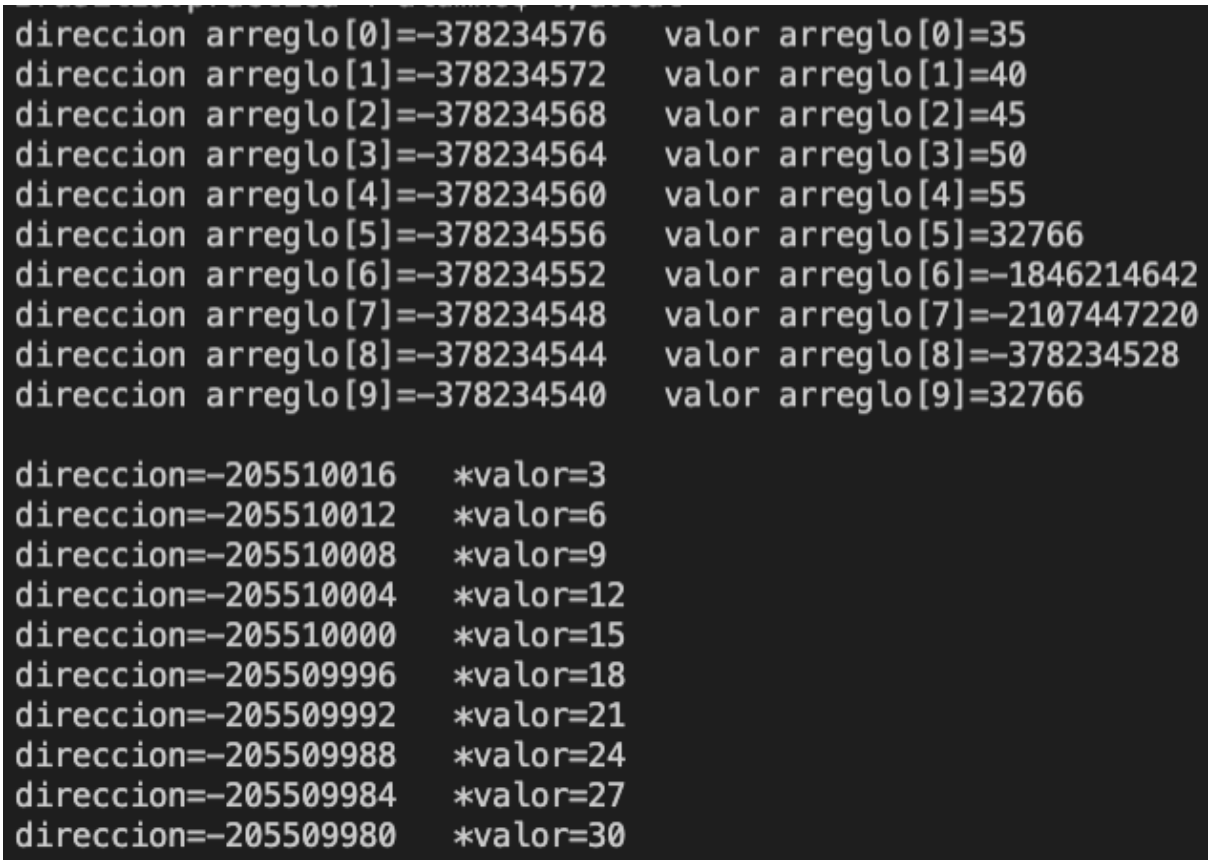
Del mismo modo se muestran las direcciones, siendo la primera la de uno de los apuntadores declarados, y sus respectivos valores que son “basura”.

b) Modifica la asignación inicial de “*ptr” de tal modo que ahora se utilice la función calloc, compila, ejecuta y explica la diferencia con la ejecución previa.



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) Agrega instrucciones para asignar valores en las localidades de memoria reservadas con memoria dinámica. (deberás asignar valores múltiplos de 3 (3,6,9…) Se deberá mostrar en pantalla los valores asignados y la dirección de éstos



d) Agrega instrucciones para utilizar la función realloc para redimensionar “ptr” con un nuevo tamaño de 25 espacios

- Utiliza realloc para asignar el nuevo tamaño en el mismo apuntador ptr
- Utiliza realloc para asignar el nuevo tamaño en el apuntador ptr3

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

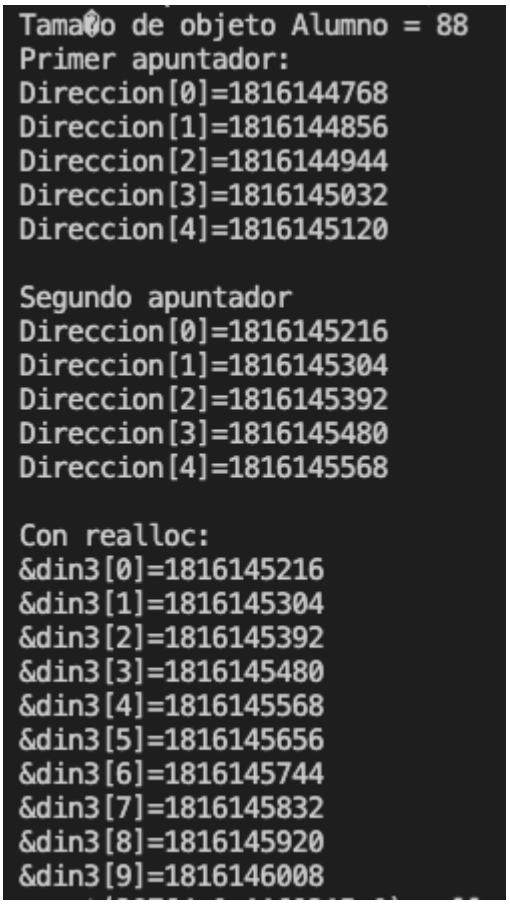
```
direccion=-1438635136   *valor=3
direccion=-1438635132   *valor=6
direccion=-1438635128   *valor=9
direccion=-1438635124   *valor=12
direccion=-1438635120   *valor=15
direccion=-1438635116   *valor=18
direccion=-1438635112   *valor=21
direccion=-1438635108   *valor=24
direccion=-1438635104   *valor=27
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.

Con este ejercicio se mostraron las disposiciones en memoria de los elementos de un arreglo convencional, como de uno apartado mediante memoria dinámica. Para las modificaciones solicitadas, se hizo uso de la aritmética de direcciones para recorrer los arreglos, y para poder imprimir las direcciones y los valores de hizo uso de los operadores de dirección(&) y de indirección(*) respectivamente. Otro detalle es que se observó el uso de realloc(), confirmando también que si se ocupan apuntadores distintos, ambos terminan haciendo referencia a las mismas localidades de memoria.

- **Ejercicio 2**

Codifica, compila y ejecuta el código “ejercicio2” posteriormente responde las preguntas.



a) Explica los resultados que se muestran en pantalla.

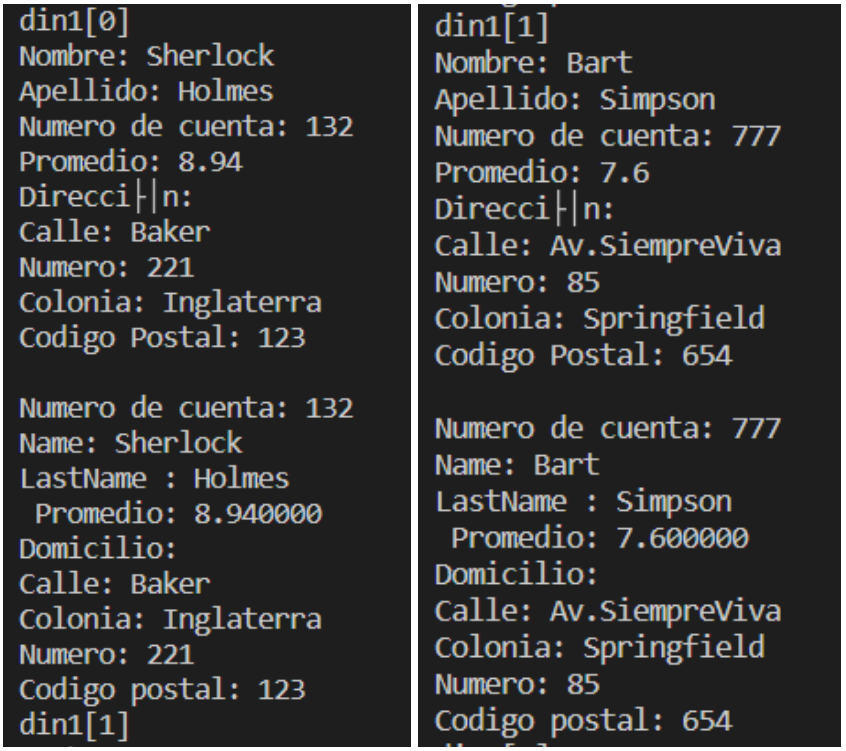
Los resultados que se muestran en pantalla son el tamaño del objeto alumno, resultante de sumar el tamaño de todos las propiedades del objeto.

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) ¿Cuál es el tamaño de la estructura Alumno? ¿Cómo se obtiene ese tamaño?

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. Además de esto se realiza un proceso llamado rellenado de estructura(structure padding), el cual se aplica en los casos en los que la suma de los bytes de las propiedades de una estructura no es múltiplo de la cantidad máxima que puede leer el procesador a la vez; por lo que para que el tamaño de la estructura sea un múltiplo de este, se añaden bytes entre propiedades o al final de estas, hasta que el tamaño de la estructura sea un múltiplo de la cantidad máxima de bytes que el procesador puede leer a la vez.

c) Agrega las instrucciones al código para pedir al usuario los datos de los alumnos que se reservan en el programa



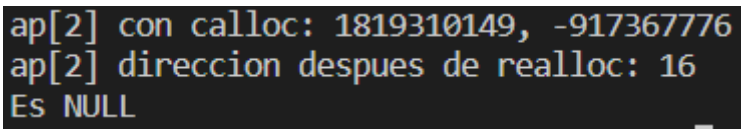
Se implementaron algunas funciones para registrar los datos de los alumnos. También para corroborar que el registro se realizó correctamente se utilizó la función imprimir alumno, a la que se le implementó una función para imprimir el domicilio. Dichas funciones se introdujeron en un ciclo de repetición para que recorriera todos los alumnos de las localidades de memoria apartadas dinámicamente.

d) Explica de qué forma se podría liberar la memoria reservada por la función calloc o malloc sin utilizar la función free, realiza el intento en código e indica tus resultados

Una forma podría ser haciendo uso de la función realloc(), ocupando como segundo parámetro 0. Así se haría que las localidades de memoria apartadas anteriormente con malloc(), o calloc(), fueran 0.

Al hacer esto pude observar que el apuntador se volvía igual a “NULL”, por lo que consideraría que funcionó. Al hacer otras pruebas me di cuenta que a diferencia de cuando se libera la memoria con free(), si se intenta hacer referencia al valor del apuntador “liberado” con realloc, este cierra directamente el programa.

A continuación la ejecución del programa que realice, este crear un arreglo usando memoria dinámica, imprime el valor y la dirección de uno de sus elementos, “libera” la memoria usando realloc, e imprime la dirección resultante, además de indicar si el apuntador es nulo.



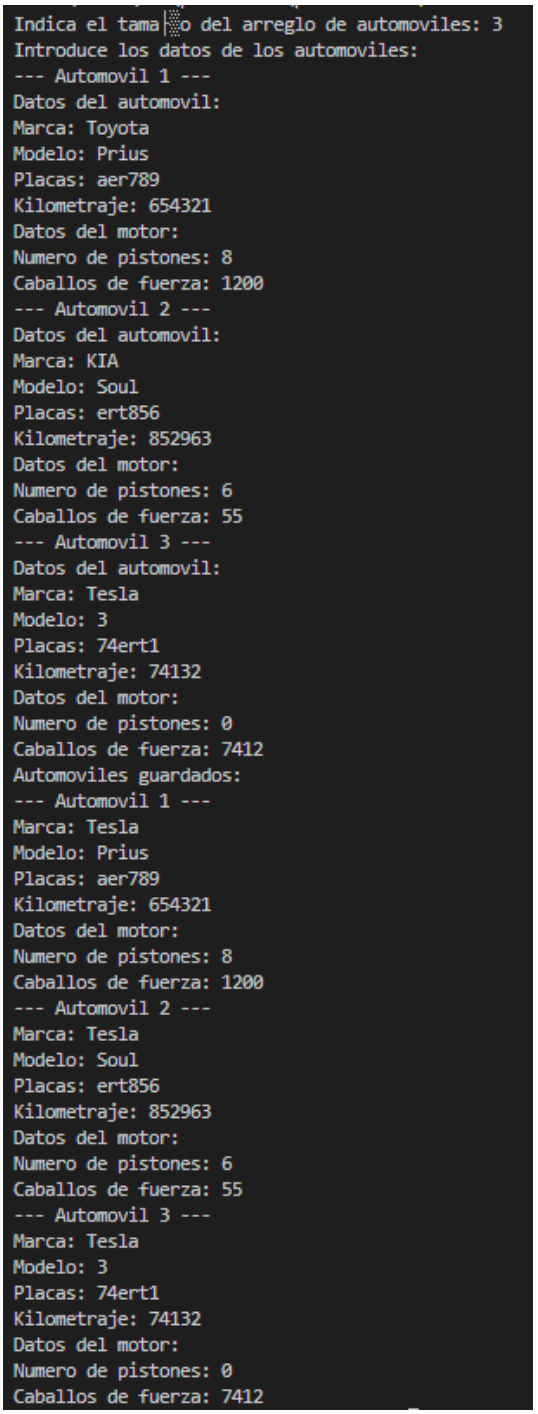
En el código de este ejercicio se solicita agregar instrucciones para introducir los datos de los alumnos de los arreglos declarados con memoria dinámica. El proceso se hicieron uso de funciones, las cuales creaban el alumno solicitando los datos al usuario para cada arreglo, e imprimiendolos inmediatamente para comprobar que todo se estuviera guardando correctamente.

Para este ejercicio ya se tenían algunas funciones definidas para crear e imprimir alumnos, por lo que se intentó trabajar en torno a ellas para aprovecharlas. Además se incluyeron funciones para realizar esas mismas acciones pero a la dirección, aunque de manera distinta en el caso de la de crearDireccion, ya que preferí que no tuviera parámetros, y en cambio dentro de la función se solicitaran.

Estas funciones adicionales se incluyeron ya que preferí separar el problema en secciones más reducidas para tanto la limpieza del código, como la facilidad de solucionar y localizar errores cuando sucedieran.

El manejo de estos elementos me permitió pensar en la memoria dinámica como un arreglo, facilitando la administración de los alumnos al pensar en ellos como elementos de un arreglo.

- **Ejercicio 3**



Este programa tiene una estructura llamada automóvil, con distintas propiedades de tipos de dato primitivo y una estructura llamada motor. Lo que se realiza es solicitar al usuario la cantidad de automóviles a ingresar, para crear un arreglo de ese tamaño. Para hacer uso de la memoria dinámica se creó un arreglo de un automóvil con malloc, y una vez se introduce el número de automóviles, este arreglo se redimensiona, con realloc usando como primer parámetro el apuntador al arreglo, y como segundo parámetro el número de elementos del arreglo.

Los datos de los automóviles y sus motores se solicitan mediante funciones, una para cada estructura, siendo que la función crear automóvil llama la función crearMotor; del mismo modo para imprimir el automóvil, la función que imprime el automóvil llama a la función de crear motor. Aunque no se solicitó la impresión en pantalla de los automóviles, decidí incluirla para corroborar que los datos se guardarán correctamente. Las funciones resultan bastante útiles para planificar mejor el problema.

El eje del programa es el arreglo de memoria dinámica, ya que nos permite extender el arreglo ya declarado al número de elementos necesarios para albergar todos los automóviles.

- **Conclusiones**

Estando al término de esta práctica, ya que hemos ocupado las funciones referentes a la memoria dinámica, y comprendí el papel de los apuntadores en estas funciones que nos permiten manejar la memoria durante la misma ejecución del programa; puedo afirmar que se cumplen los objetivos de la práctica.

Habiendo completado los ejercicios de la práctica se puede hacer mención varios conocimientos y aprendizajes que se pusieron en práctica; como el reservar memoria con malloc, o hacer lo mismo inicializando en 0 con calloc, así como el redimensionar los arreglos de memoria dinámica reservados durante la ejecución en función de lo que se requiera con realloc, y el uso de este último como alternativa a free.

Los arreglos convencionales, a pesar de su gran utilidad tenían la desventaja de no ser flexibles en relación a su tamaño, en ese sentido la memoria dinámica resulta un complemento perfecto para estos elementos, quitándonos esa limitante del tamaño, aunque añadiendo el ligero inconveniente de tener que estar atento para liberarla.

A pesar de que la práctica goza de una variedad de ejercicios que nos permiten explorar diferentes aspectos de la memoria dinámica, considero que se podría haber explorado más el redimensionamiento de un arreglo, con un ejercicio que por ejemplo pidiera que se inicializa un arreglo 5 elementos, y el usuario introduzca valores, pero que el arreglo se tenga que redimensionar aumentando su tamaño de 5 en 5 elementos hasta que el usuario lo desee.

References

GeeksforGeeks. (2022, October 21). *How to deallocate memory without using free() in C?* GeeksforGeeks. Retrieved March 18, 2023, from https://www.geeksforgeeks.org/how-to-deallocate-memory-without-using-free-in-c/