|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | M. I. Marco Antonio Martínez Quintana |
| *Asignatura:* | Estructura de Datos y Algoritmos |
| *Grupo:* | 17 |
| *No de Práctica(s):* | 04 |
| *Integrante(s):* | Díaz Segura, Mauricio Iván |
| *No. de Equipo de cómputo empleado:* |  |
| *No. de Lista o Brigada:* |  |
| *Semestre:* | 2020-2 |
| *Fecha de entrega:* | 1 - III - 2020 |
| *Observaciones:* |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

**Objetivos**

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

**Introducción**

Un puntero es una variable que apunta a una dirección de memoria. Sin embargo, en ocasiones, es necesario ocupar más memoria, para ello, se accede a un espacio llamado montículo, de donde se pueden obtener nuevas localizaciones en la memoria. Todo esto implica reservar memoria para poder almacenar más información. Para esto, se tienen tres instrucciones que ayudan bastante a trabajar con la memoria dinámica, los cuales son malloc, calloc y realloc (Knuth, 1980).

No obstante, no hay que olvidar que el programador debe optimizar el programa que está realizando, esto implica que, una vez que se haya dejado de usar dichos espacios de memoria, se deben liberar para que la computadora trabaje con mayor eficacia y, para esto, se utiliza la instrucción free (). Con todo esto, es posible aclarar que, en la ejecución de un programa, puede haber estructuras con tamaños no conocidos o que cambien en el transcurso del mismo, por ello, reservar espacios de memoria es necesario para una mayor fluidez de trabajo (Knuth, 1980).

**Desarrollo**

En la imagen 1, se puede observar que, para comenzar, se declaran tres variables: dos de tipo entero y otro de tipo apuntador, este último con nombre \*arreglo.

El siguiente paso es saber cuánto espacio en la memoria se va a reservar, este dato se almacena en la variable num de tipo entero. Cuando se tiene toda la información anterior, se expande la memoria reservada por medio de la función malloc (). Como primer parámetro, se encuentra el tipo del apuntador; continúa con llamar a la función, ella tendrá dos parámetros que son el número de espacios dentro de la memoria que va a reservar y la función que realizará esto último.

Una vez cumplido con lo anterior, se imprimen las direcciones de memoria que se han reservado. Para terminar, es importante liberar la memoria por medio de la función free(apuntador) para un rendimiento óptimo del programa.

Otra función relevante en el manejo de memoria dinámica es calloc(), el cual es bastante parecido a malloc(), con la diferencia de que, la memoria reservada estará inicializada en cero. De esta forma, se puede emplear dicho espacio a conveniencia sin tener información que resulte irrelevante a la ejecución del programa. Todo esto se puede ver en la imagen dos, cuyo código es parecido al de la imagen 1, con el cambio de función para trabajar con la memoria dinámica.

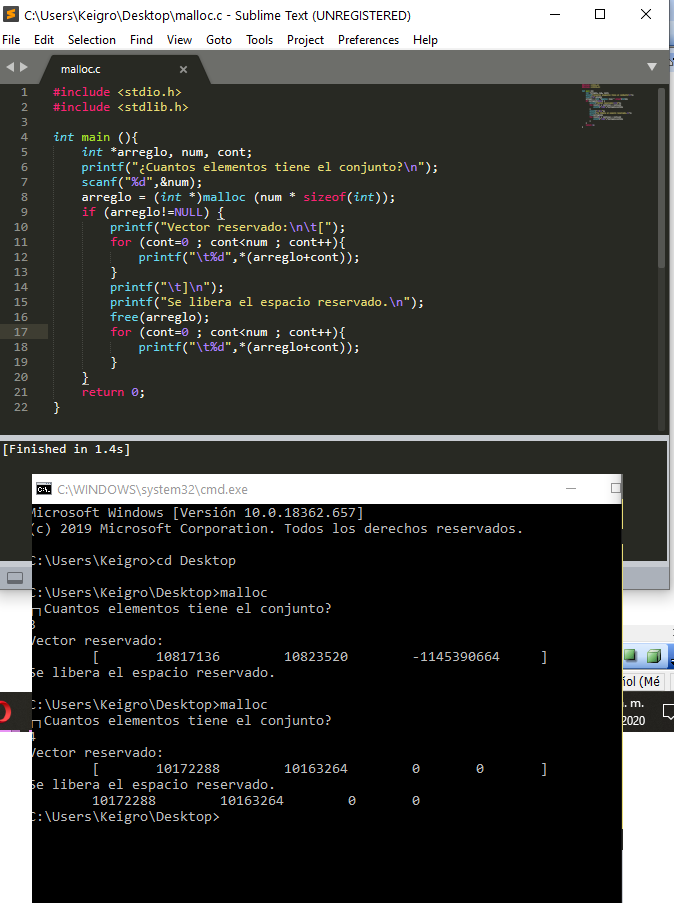


Imagen 1

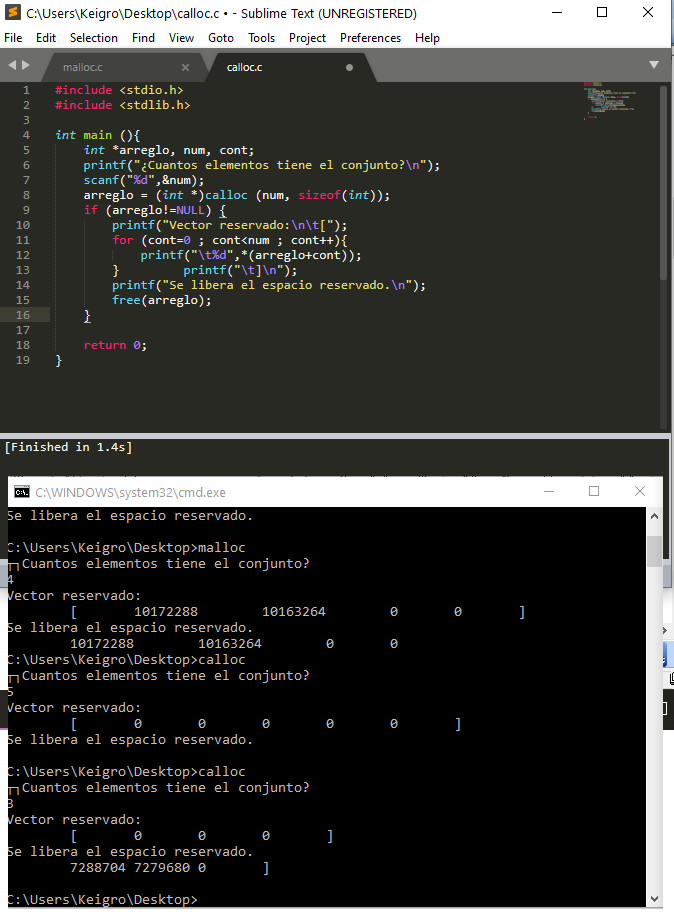


Imagen 2

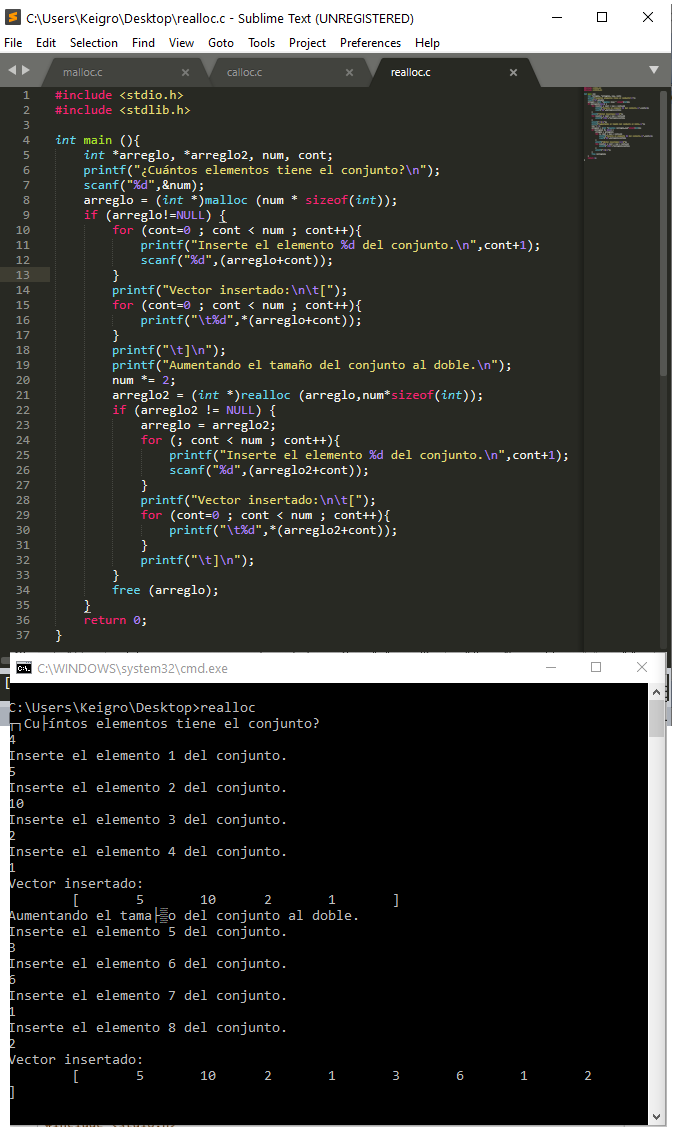


Imagen 3

Una vez que se han usado algunas de las funciones anteriores, es posible redimensionar el espacio que se ha reservado, esto con la finalidad de optimizar el manejo de datos dentro del programa. Para esto, se puede emplear la función realloc(). Hay que tomar en cuenta de que esta función puede ser usada únicamente cuando ya se ha reservado espacio de memoria con anterioridad, de otra forma, está función retornará un valor nulo porque no tiene con qué trabajar.

Por último, y como en los demás códigos, se libera la memoria.

**Conclusiones**

El manejo de memoria dinámica contribuye a que el programa en cuestión sea eficaz y eficiente a través de métodos que recuperan información de manera óptima. Los apuntadores son importantes para acceder a localidades de memoria de la computadora y, por medio de las funciones aprendidas en esta práctica, es posible ampliar el espacio de memoria a medida de que se ejecute el programa.

**Referencias**

Knuth, D. (1980). Algoritmos fundamentales, Vol. 1. Fecha de consulta: 27 de febrero de 2020. Recuperado de http://antares.sip.ucm.es/cpareja/libroCPP/docs/libro-completo.pdf