	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	M.C. JOSE MAURICIO MATAMOROS DE MARIA Y CAMPOS
Asignatura:	ESTRUCTURA DE DATOS Y ALGORITMOS I
Grupo:	12
No de Práctica(s):	03
Integrante(s):	ARELLANES CONDE ESTEBAN
No. de Equipo de cómputo empleado:	08
No. de Lista o Brigada:	01
Semestre:	2022-2
Fecha de entrega:	07 mar, 23:59
Observaciones:	

CALIFICACIÓN: _____

Práctica 3.
Manejo Dinámico de memoria

Estructuras de Datos y Algoritmos I

Autor: Arellanes Conde Esteban

Índice

1. Objetivo

El alumno conocerá las funciones en lenguage C que permiten el manejo dinámico de memoria (reserva y liberación) para el almacenamiento de información no previsible en tiempo de ejecución.

2. Introducción

El sistema operativo tendrá que generar una estructura especial en memoria que le permita controlar la ejecución del programa a fin de activarlo y desactivarlo a conveniencia, pues sólo así podrá ejecutar varios programas de forma concurrente y dar la impresión de que estos se ejecutan al mismo tiempo, es importante recalcar que ningún programa puede cargarse directamente en la memoria principal, ni siquiera aún después de haber sido traducido y para poder funcionar correctamente, un programa tiene que mantenerse como una unidad dentro de la memoria.

3. Actividad 1: Primitivas de manejo de memoria

El programa "`max.c`" consistía en reservar dos bloques de memoria usando las funciones de malloc (memory allocation) y calloc (contiguous allocation) respectivamente, para después redimensionar los bloques a un tamaño doble (2N), imprimiendo el contenido de la memoria del bloque en cada vez compilada y ejecutada del programa "`mem.c`" con la línea de comando "`./mem 10`" (N = 10 pasando como argumento) y utilizando la información para llenar la Tabla 1 a continuación:

3.1. Actividad 1: Tabla de resultados 1

Tabla 1: Direcciones de memoria y contenido de bloques reservados dinamicamente

	Dirección	Contenido
a (malloc)	0x5654001e32a0 =	{ 0 0 0 0 0 0 0 0 0 0 }
b (calloc)	0x5654001e36e0 =	{ 0 0 0 0 0 0 0 0 0 0 }
a (realloc)	0x5654001e32a0 =	{ 0 0 0 0 0 0 0 0 0 0 1041 0 909473840 808465461 842229041 2067607649 807415840 807415840 807415840 807415840 }
b (realloc)	0x5654001e36e0 =	0 0 0 0 0 0 0 0 0 0 133377 0 0 0 0 0 0 0 0 0

4. Actividad 2: Máximos y mínimos

El programa "`max.c`" imprime el máximo valor encontrado en arreglo unidimensional de tamaño fijo de números generados aleatoriamente con base en la semilla proporcionada.

[3 puntos] Modificando dicho programa para que imprima el valor mínimo encontrado en el arreglo. ¿Cuál es el mínimo reportado?: El valor mínimo reportado es 0(0).

Modificando el programa anterior para que aceptara como primer parámetro el tamaño del vector de números aleatorios a generar, y como segundo parámetro la semilla del generador de números aleatorios. Llenamos la Tabla 2.

4.1. Actividad 2: Tabla de resultados 2

Tabla 2: Máximos y Mínimos [2 puntos]

Parámetro 1	Parámetro 2	Máximo
10	0	60645 (5)
10	69	60645 (5)
10	80	60645 (5)
100	0	68576 (25)
100	69	68576 (25)
100	80	68576 (25)
1000	0	68965 (738)
1000	69	68965 (738)
1000	80	68965 (738)

5. Actividad 3: Producto punto

5.1. Actividad 3: Tabla de resultados 3

El próposito del programa "`max.c`" es el de imprimir el el producto punto de dos arreglos unidimensionales de tamaño fijo compuestos por números aleatorios. Modificando dicho programa para que imprima el producto punto $p = \overline{u} \cdot \overline{v}$ de dos vectores de números pseudoaleatorios con diferente semilla. En esta ocasión recibe tres parámetros:

i) El número de componentes en los vectores a generar (tamaño del arreglo).

ii) La semilla usada para generar las componentes del vector $\overline{u}$.

iii) La semilla usada para generar las componentes del vector $\overline{v}$.

Compilando y llenando la Tabla 3.

Tabla 3: Producto Punto [3 puntos]

Parámetro 1 ($\lvert\overline{u}\rvert = \lvert\overline{v}\rvert$)	Parámetro 2	Parámetro 3	$p = \overline{u} \cdot \overline{v}$
10	0	1	u · v = 2.73914337
10	69	80	u · v = 3.10264277
10	42	42	u · v = 1.61234188
100	0	1	u · v = 2.62317133
100	69	80	u · v = 2.55665684
100	42	42	u · v = 1.93315077
1000	0	1	u · v = 3.24957395
1000	69	80	u · v = 3.82529020
1000	42	42	u · v = 2.59516525

6. Cuestionario

Responda las siguientes preguntas considerando la información de la Tabla 1.

1. [2 puntos] ¿Cuál es la diferencia entre las funciones malloc y calloc?: Malloc es del tamaño del tipo de variable almacenada por el número de estos que queramos en un sólo un bloque de memoria eje. 5*sizeof(int) => 20 bytes; mientras que calloc lo almacena, pero en distintos bloques de memoria contiguos e inicializándolos a todos en cero. eje. 5, sizeof(int) => 20 bytes.

2. [1 punto] ¿Para qué sirve la función realloc? De manera breve, sirve para aumentar la alocación de memoria previamente almacenada y sirve para casos en los que el espacio es insuficiente.

3. [1 punto] ¿Para qué sirve la función free y cuándo debe utilizarse? Sirve para "desalojar" la memoria dinámica y debe utilizarse para evitar que en futuros programas el espacio en memoria ya esté tomado y produzca un error de segmentación.

7. Conclusión y Referencias

7.1. Conclusión:

El uso de malloc, calloc, realloc y free son formas de reserva de memoria dinámica no conocida a primera instancia y liberarla posteriormente, esto lo pudimos observar a lo larog de la práctica con los programas realizados durante el laboratorio.

7.2. Referencias:

- Thomas H. Cormen. (2009). Introduction to Algorithms 3e. United States of America: Massachusets Institute of Technology.
- Stephen R. Davis. (2015). Beginning Programming with C++ For Dummies. New Jersey, United States of America: John Wiley Sons.