Algoritmos y Estructuras de Datos I

Trabajo Práctico N°10.2	Unidad 10
Modalidad: Semi -Presencial	**Estratégica Didáctica:** Trabajo individual
Metodología de Desarrollo: acordar	**Metodología de Corrección:** acordar docente
Carácter de Trabajo: Obligatorio – Con Nota	**Fecha Entrega:** A confirmar por el Docente.

PUNTEROS
MARCO TEÓRICO

1. ¿Describir la distribución de Memoria y que tipo de dato alberga cada una?
La distribución de memoria, está organizada en distintas secciones que almacenan diferentes tipos de datos. Los datos estáticos incluyen variables globales y constantes, y se almacenan en el área de datos globales. Las variables locales y los datos del programa en ejecución se almacenan en la pila, que es administrada automáticamente por el sistema cuando se invocan y terminan las funciones. La memoria dinámica se reserva en el montón, un área gestionada explícitamente a través de operaciones con punteros y el uso del operador new. Esta área almacena datos dinámicos creados en tiempo de ejecución, los cuales solo pueden ser accedidos a través de punteros, como se ve en el ejemplo de asignación de punteros con el operador new. La memoria del código del programa y el sistema operativo también tienen áreas específicas en la memoria principal, aunque los datos locales y dinámicos son los principales elementos en constante asignación y liberación durante la ejecución del programa.

2. Que es el operador NEW, para que sirve?
El operador new se usa en C++ para la asignación de memoria dinámica en el montón. Este operador solicita un espacio de memoria suficiente para almacenar un dato de un tipo específico y devuelve un puntero a la primera dirección de esa memoria asignada. De esta forma, la memoria es accesible solo a través de punteros, ya que no está asociada a un identificador de variable estática. Por ejemplo, el código p = new int asigna memoria para un entero y devuelve la dirección de esa memoria a p. Es posible también inicializar la memoria asignada usando el operador new, como en p = new int(12), que asigna memoria para un entero y lo inicializa en 12. En sistemas sin recolector de basura, como en C++, toda memoria asignada con new debe ser liberada posteriormente con delete para evitar fugas de memoria y asegurar que el sistema recupere el espacio reservado. Esto es esencial, ya que cada dato dinámico debe tener algún acceso para evitar la pérdida de memoria no referenciada en el montón.

3. ¿Qué utilidad tiene el operador Delete, que tipo de memoria libera?
El operador delete permite devolver al montón la memoria usada por una variable dinámica que fue apuntada por un puntero. Al liberarse la memoria, el puntero

deja de contener una dirección válida. Es importante ejecutar tantos delete como new para evitar pérdidas de memoria y asegurarse de que la memoria del montón sea la misma antes y después de la ejecución del programa.

4. En qué lugar de la memoria se definen los Punteros int *Puntero?

Los punteros int *Puntero se definen en el montón, que es una región de la memoria dedicada a los datos dinámicos. Para que el puntero apunte a un espacio válido en memoria, se debe asignar una dirección utilizando el operador new, que reserva un bloque de memoria y devuelve su dirección para que el puntero pueda acceder a ese dato dinámico. A diferencia de una variable estática, la variable dinámica creada en el montón solo es accesible a través del puntero.

5. ¿Qué entiende por colisión Pila (Stack) Montón (Heap)?

El sistema operativo organiza la memoria en varias regiones, que incluyen la pila (stack), el montón (heap), datos globales y el código del programa. Los datos locales se almacenan en la pila, mientras que los datos dinámicos se almacenan en el montón. Ambas regiones crecen en direcciones opuestas; la pila crece al llamar subprogramas, y el montón se expande cuando se crean datos dinámicos. Existe la posibilidad de colisión pila-montón, lo que significa que los límites de estas regiones podrían encontrarse, agotando la memoria disponible y generando errores como el desbordamiento de la pila.

6. ¿Qué es el desbordamiento de Pila?

El desbordamiento de pila ocurre cuando se supera el tamaño máximo establecido para la pila, generalmente debido a un uso excesivo de memoria por llamados recursivos o variables locales. Al alcanzarse el límite, se agota el espacio en la pila, generando un error que puede causar que el programa falle.

7. ¿Describir los Errores más comunes en asignacion de Memoria Dinamica?

Errores comunes:

- Olvido de liberar un dato dinámico: si no se ejecuta delete para cada new, se deja memoria desperdiciada en el montón, lo cual no siempre es detectado por compiladores como G++, aunque Visual C++ puede indicar esta falta.
- Intento de eliminar un dato inexistente: intentar hacer delete de un puntero que ya ha sido eliminado puede causar errores graves.
- Pérdida de un dato dinámico: al reasignar un puntero a otro sin liberar el dato previamente apuntado, se pierde el acceso al dato anterior, generando una fuga de memoria.
- Intento de acceso a un dato tras su eliminación: después de realizar delete sobre un dato dinámico, intentar acceder a ese dato puede llevar a un acceso a memoria inexistente, causando errores y comportamientos inesperados en el programa.

8. ¿Que entiende por array a datos dinámicos?

Un array a datos dinámicos se refiere a un array de punteros que apuntan a datos que se almacenan dinámicamente en la memoria. Esto permite que la estructura del array se adapte en tamaño a medida que se insertan o eliminan elementos, aprovechando la memoria del montón.

9. ¿Que son los Array dinámicos?

Los arrays dinámicos son punteros que apuntan a un bloque de memoria en el montón, permitiendo la creación de un array de tamaño variable en tiempo de ejecución. Esto contrasta con los arrays estáticos, que tienen un tamaño fijo establecido en tiempo de compilación.

10. Describir las diferencias entre Arrays dinámicos vs. arrays de dinámicos

Las diferencias entre arrays dinámicos y arrays de datos dinámicos son las siguientes:

Los arrays dinámicos son punteros a un bloque de memoria que se asigna dinámicamente y permite un tamaño variable, mientras que los arrays de datos dinámicos son arrays de punteros a datos que también se almacenan en memoria dinámica.

Los arrays dinámicos gestionan directamente la memoria de un único tipo de dato, mientras que los arrays de datos dinámicos permiten almacenar referencias a múltiples elementos que pueden ser de tipos diferentes, a través de punteros.

En un array dinámico, los datos se almacenan de forma contigua, mientras que en un array de datos dinámicos, cada puntero puede apuntar a diferentes ubicaciones de memoria en el montón, permitiendo una mayor flexibilidad en la gestión de los datos.

MARCO PRÁCTICO

Tomando como Marco el Programa de la Unidad 7 (copiado al final) modificando Modularizándolo mediante el uso de Bibliotecas.

Tener en Cuenta:

- Modularizar el Programa.
- Proteger contra Inclusiones Múltiples.
- Aplicar Espacios de Nombres.
- Aplica apropiadamente los conceptos de abstracción, encapsulación y ocultamiento de información.
- Realiza una apropiada distribución de responsabilidades entre las entidades del espacio de la solución.
- Desarrolla para reusar.
- Reusa apropiadamente las entidades desarrolladas en el espacio curricular.
- Demuestra un uso apropiado de la sintaxis y semántica del lenguaje de programación C++.

COPIA UNIDAD 7

Desarrollar un Programa que:

1. Rellenar un array de 10 números, posteriormente utilizando punteros indicar cuales son números pares y su posición en memoria.
2. : Rellenar un arreglo con n números, posteriormente utilizando punteros determinar el menor elemento del vector.

3. Pedir al usuario N números, almacenarlos en un arreglo dinámico posteriormente ordenar los números en orden ascendente y mostrarlos en pantalla.NOTA: Utilizar cualquier método de ordenamiento.

4. Pedir una cadena de caracteres (string) al usuario, e indicar cuantas veces aparece la vocal a,e,i,o,u; en la cadena de caracteres. NOTA: Usar punteros

5. Realice un programa que calcule la suma de dos matrices dinámicas.

6. Realice un programa que lea una matriz dinámica de NxM y cree su matriz traspuesta. La matriz traspuesta es aquella en la que la columna i era la fila i de la matriz original.

7. Hacer una estructura llamada alumno, en la cual se tendrán los siguientes Campos: Nombre, edad, promedio, pedir datos al usuario para 3 alumnos, comprobar cuál de los 3 tiene el mejor promedio y posteriormente imprimir los datos del alumno. NOTA: Usar punteros a estructura.

Lic. Oemig José Luis.