**Instituto Superior de Formación Técnica Nº 151 logo151-trans.gif  
Carrera: Analista de Sistemas  
1 Año. Algoritmos y Estructuras de Datos I**

|  |  |
| --- | --- |
| **Trabajo Práctico Nº 9.2** | **Unidad 9** |
| **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
2. ¿Qué es el operador NEW, para qué sirve?
3. ¿Qué utilidad tiene el operador Delete, qué tipo de memoria libera?
4. ¿En qué lugar de la memoria se definen los Punteros int \*Puntero?
5. ¿Qué entiende por colisión Pila (Stack) Montón (Heap)?
6. ¿Qué es el desbordamiento de Pila?
7. Describir los Errores más comunes en la asignación de Memoria Dinámica
8. ¿Qué entiende por array a datos dinámicos?
9. ¿Qué son los Array dinámicos?
10. Describir las diferencias entre Arrays dinámicos *vs.* arrays de dinámicos

**Marco Practico:**

**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++.

**Desarrollar unos Programas** que:

1. Rellenar un array de 10 números, posteriormente utilizando punteros indicar cuales son números pares y su posición en memoria. **Ej01\_02.cpp**
2. Rellenar un arreglo con n números, posteriormente utilizando punteros determinar el menor elemento del vector. **Ej02.cpp**
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. **Ej03.cpp**
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 **Ej04\_02.cpp**
5. Realice un programa que calcule la suma de dos matrices dinámicas. **MatricesDin.cpp** y **MatDinStruct.cpp** (encapsulada)
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. **MatricesDin.cpp** y **MatDinStruct.cpp** (encapsulada)
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. **Ejercicio\_7.rar**

Lic. Oemig José Luis.

1. Describir la distribución de Memoria y que tipo de dato alberga cada una

El sistema operativo distingue varias regiones en la memoria:

* La pila o stack, que es donde se almacenan los datos locales.
* El montón o Heap, que es donde se almacenan los datos dinámicos.
* La Memoria Principal, donde se guardan los datos globales, el código del programa y el Sistema Operativo.

1. ¿Qué es el operador NEW, para qué sirve?

Cuando se declara una variable de tipo puntero, el compilador reserva un lugar en el stack para el puntero en sí. Cuando se ejecuta el operador New se habilita un espacio de memoria en el heap cuya dirección de memoria es almacenada en el espacio del stack previamente reservado.

1. ¿Qué utilidad tiene el operador Delete, qué tipo de memoria libera?

El operador Delete se aplica a una variable de tipo puntero y lo que hace es liberar la memoria del heap que se comentó en el punto anterior.

1. ¿En qué lugar de la memoria se definen los Punteros int \*Puntero?

Como se comentó en el punto 2, la variable en sí de tipo puntero queda definida en el stack.

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



La pila y el heap crecen en direcciones opuestas (ver gráfico). Al llamar a subprogramas la pila crece. Al crear datos dinámicos el montón crece. Se llama colisión cuando los límites de ambas regiones se encuentran (se agota la memoria).

1. ¿Qué es el desbordamiento de Pila?

La pila suele tener un tamaño máximo establecido. Si se sobrepasa se agota la pila (desbordamiento)

1. Describir los Errores más comunes en la asignación de Memoria Dinámica

* **El olvido en la destrucción de un dato dinámico.** Recordar que debe haber tantos ***delete*** como ***new***: si falta algún ***delete***, se desperdiciará memoria (aunque el compilador no dé error).
* **Intento de destrucción de un dato inexistente:** intentar usar delete cuando la variable no tiene asignada una dirección de memoria en el heap, ya sea porque nunca se creó (no se usó el ***new***) o porque ya se había borrado previamente.
* **Pérdida de un dato dinámico:** si un puntero está apuntando a un dato del heap y de pronto apunta a otra dirección, sin que la dirección inicial sea apuntada por nadie más, dicha información queda perdida en el heap ocupando espacio.
* **Intento de acceso a un dato tras su eliminación.**

1. ¿Qué entiende por array a datos dinámicos?

Un array de punteros a datos dinámicos es justamente eso: Un array estático (que vive en el stack), donde cada uno de sus elementos apunta a un elemento en el heap.

1. ¿Qué son los Array dinámicos?

Un array dinámico es un array que se ubica en la memoria dinámica (heap).

1. Describir las diferencias entre Arrays dinámicos *vs.* arrays de dinámicos

El siguiente esquema describe a la perfección la diferencia entre los dos conceptos:

