ANÁLISIS Y DISEÑO DE ALGORITMOS I
Práctico Laboratorio. El lenguaje C ++

---

1. Escribir un programa que imprima el mensaje “Hola mundo” por pantalla.
2. Escribir un programa que lea dos números ingresados por teclado (separados por un espacio ‘ ‘), los multiplique e imprima el resultado por pantalla.
3. Escribir un programa que lea dos cadenas de caracteres ingresadas por teclado (separadas por un espacio ‘ ‘), las concatene e imprima el resultado por pantalla.
4. Explicar las diferencias entre las implementaciones de los ejercicios 2 y 3. Si no se utilizó el objeto `cin` en los ejercicios, reimplementar ambas soluciones para incorporarlo.
5. Escribir un programa que lea un mensaje desde el teclado, convierta mayúsculas a minúsculas (o viceversa) e imprima el mensaje resultante por pantalla. La conversión se realizará de acuerdo al parámetro que reciba la función `main` desde la línea de comandos (-M: mensaje a mayúsculas, -m: mensaje a minúsculas).
6. Dada la ecuación de segundo grado: ax2 + bx + c, al calcular el discriminante discr = b2 - 4ac, se pueden presentar tres casos distintos:
   Si discr > 0.0, las dos raíces son reales y distintas, y valen:
   x1= (-b + (discr)1/2) / (2a) y x2 = (-b - (discr)1/2) / (2a).
   Si discr = 0.0, las dos raíces son reales e iguales, y valen:
   x1 = x2 = -b / (2a)
   Si discr < 0.0, las dos raíces son complejas conjugadas. Las partes real e imaginaria valen:
   xr = -b / (2a) y xi = (-discr)1/2 / (2a)
   Codificar un programa que permita obtener las raíces de una ecuación de segundo grado.
7. Dada una cadena de caracteres, escribir un programa que cuente y muestre por pantalla la ocurrencia de caracteres divididos en tres grupos distintos: los dígitos, las letras en mayúscula y las letras en minúscula.
8. Resolver el ejercicio 7 utilizando una función que reciba como parámetros la cadena de caracteres y un carácter a buscar, y retorne la cantidad de ocurrencias del mismo. Comparar la complejidad temporal de esta solución con la planteada originalmente en el ejercicio 7.
9. Escribir una función que reciba como parámetro un arreglo de N números naturales, busque el elemento “mayoría” y retorne si existe el elemento mayoría y, en caso positivo, la cantidad de veces que aparece en el arreglo. El elemento mayoría es aquel que aparece más de N/2 veces en el arreglo.
10. Dado el arreglo “valores” de números enteros, escribir una función que genere el arreglo “posiciones” que contenga los índices de “valores”, de manera tal que permitan acceder a sus elementos en forma ordenada creciente.
    - Calcular la complejidad temporal de la solución propuesta.
    - Analizar si es posible realizar alguna mejora.
11. Escribir un programa que lea desde un archivo un conjunto de líneas de texto e imprima en otro archivo la de mayor longitud.

12. Leer desde un archivo las temperaturas máximas de todos los días de un año y escribir funciones que permitan:
    - Imprimir la temperatura máxima de un día y mes dado,
    - calcular e imprimir el promedio de temperaturas máximas para un mes dado,
    - calcular e imprimir el promedio de temperaturas máximas entre dos días dados.

    Por simplicidad, suponer que el año está compuesto de 12 meses de 30 días.

13. En el Registro Civil se administran los datos de las personas que han solicitado cambio de domicilio. Declarar una estructura para almacenar los datos de, como máximo, 100 personas registradas, donde cada elemento de la estructura contenga los campos nombre, apellido, número de DNI y domicilio.
    Codificar un programa que permita resolver las siguientes operaciones:
    - Cargar datos de una persona por teclado.
    - Cargar datos de personas desde un archivo.
    - Guardar los datos de las personas actuales a un archivo.
    - Modificar el domicilio de un DNI dado.
    - Mostrar por pantalla los datos de todas las personas registradas.

    Analizar el comportamiento del programa si no se conociera con anticipación el número de personas a registrar.

14. Implementar las funciones necesarias para resolver las siguientes definiciones recursivas:

    a) Múltiplicación de dos números (x e y):

$$x \times y = 0 \quad \text{si } y = 0$$
$$x \times y = x + x \times (y-1) \quad \text{si } y > 0$$

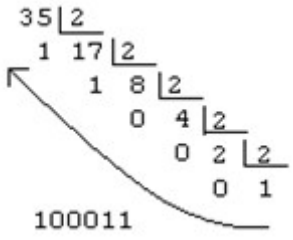
    b) Factorial de un número natural incluido el 0 (n!)

$$n! = 1 \quad \text{si } n = 0$$
$$n! = n \times (n-1)! \quad \text{si } n > 0$$

    c) Máximo común divisor de dos números enteros positivos:

$$mcd(a,b) = mcd(b,a) = \begin{cases} a & \text{si } a = b \\ mcd(a, b-a) & \text{si } a < b \\ mcd(a-b, b) & \text{si } a > b \end{cases}$$

15. Convertir un número decimal a su representación binaria, sabiendo que: la representación binaria para un número decimal n es la división entera de n entre 2 en binario, seguido del resto de dividir n entre 2. Por ejemplo:



16. Definir un arreglo dinámico de enteros y otro de punto flotante de doble precisión. Mostrar por pantalla las direcciones de memoria de cada posición y analizar las diferencias.

17. Codificar un programa que a partir de dos cadenas de caracteres, cree una tercera cadena de caracteres utilizando memoria dinámica para almacenar la concatenación de las dos cadenas originales.

18. Resolver el ejercicio 10 definiendo el arreglo “posiciones” como un arreglo de punteros a enteros, donde cada celda de “posiciones” ahora apunta al entero en sí en lugar de almacenar el índice del arreglo “valores”. Comparar esta solución con la planteada originalmente.

19. Codificar una función que dada una matriz cuadrada de enteros retorne si es simétrica o no. Una matriz es simétrica si Matriz[i,j] = Matriz[j,i] para todo par i,j dentro de los límites de la matriz. Ingresar la dimensión y los datos de la matriz por teclado.

20. Escribir una función que dada una matriz de caracteres de dimensiones n x m, devuelva una cadena de caracteres compuesta por el recorrido en espiral de la misma. El recorrido en espiral de la siguiente matriz es “ABCDHLPTSRQMIEFGKONJ”

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P
Q	R	S	T

21. Codificar un programa que implemente una estructura de arreglo de enteros de tamaño dinámico. Definir la manera y el momento en que el arreglo incrementa su tamaño para almacenar nuevos elementos. El arreglo dinámico debe permitir:
    - Agregar un elemento al principio.Agregar un elemento al final.
    - Agregar un elemento en una posición arbitraria.
    - Verificar si un elemento pertenece al arreglo.
    - Consultar la cantidad de elementos que contiene el arreglo.
    - Consultar si el arreglo está vacío.
    - Eliminar un elemento determinado.
    - Retornar el elemento ubicado en una posición arbitraria.

22. Codificar un programa que implemente una estructura de lista dinámica que permita:
    - Agregar un elemento al principio de la lista.
    - Agregar un elemento al final de la lista.
    - Agregar un elemento en una posición arbitraria de la lista.
    - Verificar si un elemento pertenece a la lista.
    - Consultar la cantidad de elementos que contiene la lista.
    - Consultar si la lista está vacía.
    - Eliminar un elemento de la lista.
    - Realizar una iteración sobre los elementos de la lista.

23. Analizar las ventajas y desventajas de las implementaciones realizadas en los ejercicios 21 y 22, de acuerdo a los distintos contextos donde se pueden utilizar las estructuras.

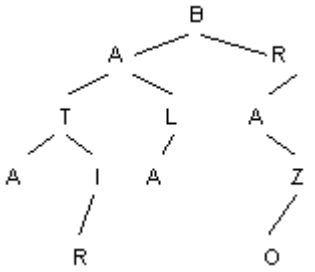
24. Considerando la siguiente estructura propuesta para un árbol binario de enteros, implementar las funcionalidades solicitadas a continuación:

```
struct nodoArbol { int dato;
                   nodoArbol *izq, *der; };
```

    - Visitar cada nodo del árbol en preorden, inorden y postorden.
    - Devolver la suma de todos los valores almacenados en el árbol.
    - Devolver el mayor y el menor número almacenado en el árbol.
    - Verificar si existe un número dado en el árbol.
    - Determinar si dos árboles son isomorfos.

25. Modificar la estructura anterior para implementar un árbol ternario y resolver:
    - Imprima los valores almacenados en los nodos que se encuentran en los niveles impares del árbol.
    - Dada una lista vacía y un entero k, devuelva en la lista las hojas del árbol cuyos niveles sean mayores o iguales a k

26. Dado un árbol binario de caracteres y un entero k, imprima por pantalla todas las palabras formadas por los caracteres almacenados entre la raíz y las hojas cuya longitud sea mayor que k. Para el árbol del ejemplo y k = 4, debe imprimir: BATIR y BRAZO.



27. Utilizando el árbol binario del ejercicio anterior: dado un camino expresado en forma de lista, determine si existe dicho camino en el árbol, teniendo en cuenta que el camino debe comenzar necesariamente en la raíz. Para el árbol del ejemplo, el camino <B,R,A> existe.

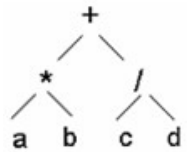
28. Dada la siguiente estructura, resuelva:

```
struct nodo {
   struct nodo* siguiente;
   struct nodo* otroNivel;
   int dato;
}
```

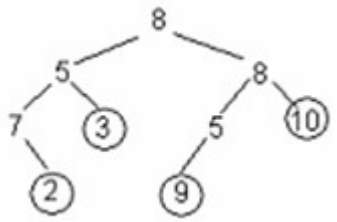
    - Determine la cantidad de enteros almacenados y su suma.
    - Determine si un número dado está almacenado en la estructura.
    - Imprima todos los datos almacenados en los niveles impares.
    - Almacene en un arreglo los datos que se encuentran en el nivel k.
    - Devuelva el nivel máximo.

29. Dado un árbol binario completo con el último nivel completo, cuyas hojas contienen cadenas de caracteres, rotule cada nodo interior del árbol de la siguiente manera: si el nivel es impar, con la concatenación de los rótulos de los hijos; si el nivel es par, con la cadena de mayor longitud de sus hijos.

30. Dados un árbol binario que representa una expresión aritmética (los nodos interiores representan operadores binarios y las hojas representan variables) y una asignación de valores a las variables, retorne el resultado de evaluar la expresión. Para el árbol del ejemplo, si a = 2, b = 2, c = 10 y d = 2, el resultado de la evaluación es 9.



31. Dado un árbol binario de enteros, como el que se presenta a continuación, resuelva:

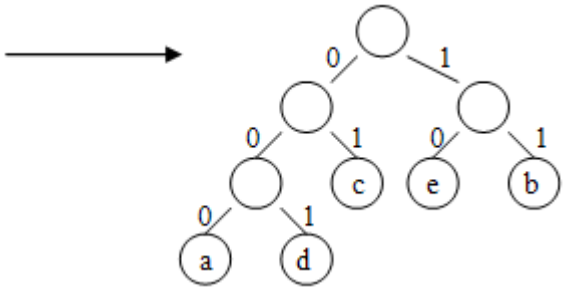


- Encuentrar la máxima diferencia (en valor absoluto) entre dos hojas adyacentes. Para el árbol del ejemplo, la máxima diferencia es 6 (| 3 – 9|).
- Almacenar en un arreglo la rama más liviana, es decir aquella rama que contiene valores cuya suma es menor a los de cualquier otra rama en el árbol. Para el árbol del ejemplo, almacenar <8,5,3>.

32. Dado un árbol binario generar los rótulos de las hojas a partir del recorrido del árbol teniendo en cuenta:
    a) el rótulo es una secuencia de 0s. y 1s.
    b) la bifurcación a un hijo izquierdo agrega un 0 en la secuencia.
    c) la bifurcación a un hijo derecho agrega un 1 en la secuencia.

33. Suponiendo la codificación de mensajes mediante secuencias de 0s y 1s tal que el código para un carácter no sea prefijo del código de otro carácter. Almacenar los códigos en un árbol binario tal que, las hojas almacenan los símbolos, y el camino de la raíz a las hojas (con la interpretación 0 a la izquierda y 1 a la derecha) da el código de cada hoja. Ejemplo:

símbolo	a	b	c	d	e
código	000	11	01	001	10



Escribir un algoritmo recursivo que permita decodificar mensajes. Por ejemplo, dado el código mostrado en la figura y el mensaje: "0001100110", la decodificación sería "abde".

34. Suponiendo tener mensajes codificados en código Morse (sistema convencional de lenguaje telegráfico en base a puntos y rayas). Almacene los códigos en un árbol binario y escriba un algoritmo recursivo que decodifique un mensaje dado.

35. Implementar las soluciones planteadas en los ejercicios 21 y 22 para incorporar el concepto de clases.