UNIVERSIDAD NACIONAL DE INGENIERÍA

Facultad de Ciencias

Escuela Profesional de Ciencia de la Computación

Curso: Fundamentos de Programación CC112 Semestre 2025-I

Laboratorio 1

Funciones, recursividad

1. Escribir una Función para calcular el máximo de dos números enteros. Nombre a la función maximo, ésta debe recibir dos números enteros como parámetros y devolver el mayor de los dos.

   Objetivo: Recordar la sintaxis básica de funciones y el uso de parámetros y retorno de valores.

2. Implementa una función llamada cuadrado que tome un número entero como parámetro y devuelva su cuadrado.

   Objetivo: Practicar la definición de funciones y su uso en cálculos simples.

3. Escribe una función llamada suma_naturales que tome un número entero n como parámetro y devuelva la suma de los primeros n números naturales.

   Objetivo: Trabajar con bucles dentro de funciones y entender la acumulación.

4. Implementa una función llamada esPrimo que determine si un número dado es primo. La función debe devolver un valor booleano.

   Objetivo: Usar condicionales dentro de funciones y trabajar con lógica booleana.

5. Escribe una función llamada invertirCadena que reciba una cadena como parámetro y devuelva la cadena invertida.

   Objetivo: Manipulación de cadenas dentro de funciones y el uso de bucles.

6. Implementa las funciones factorial y factorialRecursivo que tome un número entero n y devuelva el factorial de n.
   Compara el consumo de memoria y tiempo de ejecución.

   Objetivo: Comparar la implementación iterativa vs recursiva.

7. Implementa las funciones mcd y mcdRecursivo que tome dos números enteros y devuelva su máximo común divisor utilizando el algoritmo de Euclides.

   Objetivo: Implementar algoritmos clásicos dentro de funciones. Comparar la implementación iterativa vs recursiva

8. Escribe dos funciones suma_naturales_iterativo y suma_naturales_recursivo para calcular la suma de los primeros n números naturales, una de forma iterativa y la otra recursiva. Luego, compara el rendimiento y las ventajas/desventajas de cada enfoque.

   Objetivo: Comparar la eficiencia de las implementaciones iterativas y recursivas, además de discutir cuándo es preferible usar cada una.

9. Escribe dos funciones para calcular el n-ésimo número en la secuencia de Fibonacci: una usando un bucle (fibonacciIterativo) y otra usando recursión (fibonacciRecursivo). Discute el rendimiento y posibles optimizaciones.

   Objetivo: Mostrar cómo la recursión puede ser ineficiente sin optimización (como la memoización) en comparación con la versión iterativa.

10. Implementa la búsqueda binaria tanto de manera iterativa (busquedaBinariaIterativa) como recursiva (busquedaBinariaRecursiva). Compara los dos enfoques en términos de claridad, facilidad de implementación y eficiencia.

    Objetivo: Explorar cómo la recursión puede simplificar algunos problemas de búsqueda en estructuras de datos como arrays.

11. Implementa dos funciones para calcular x^n, una de manera iterativa (potenciaIterativa) y otra de manera recursiva (potenciaRecursiva). Discute cómo se maneja la eficiencia en ambas implementaciones, especialmente en grandes exponentes.

    Objetivo: Entender cómo optimizar recursiones profundas y cómo la recursión puede a veces ser más clara pero menos eficiente.

Ejercicios de preparación

12. Implementar una función recursiva que permita convertir un número decimal a binario.

13. Una bacteria se divide en dos cada hora. Implementa una función recursiva que calcule la cantidad de bacterias después de N horas.

14. Implemente una función recursiva para calcular el n-ésimo término de la sucesión de Padovan

15. Escribe un programa que resuelva el problema de las Torres de Hanói utilizando recursividad. El programa debe pedir al usuario el número de discos. Imprimir cada movimiento necesario para trasladar los discos desde la varilla de origen hasta la varilla destino, usando una varilla auxiliar. Respetar la regla de que un disco más grande no puede estar sobre uno más pequeño.

16. Escribe un programa que encuentre un camino desde la entrada hasta la salida de un laberinto representado por una matriz de 0s (camino disponible) y 1s (pared). Debe inicializar la matriz del laberinto de forma aleatoria. El usuario debe ingresar la posición de inicio y la posición de salida.

    El programa debe usar recursividad para encontrar un camino y mostrarlo en la matriz. Si no hay camino, debe indicarlo.

17. Implementar una función recursiva para encontrar todas las permutaciones de una cadena de caracteres. Por ejemplo, si la cadena de entrada es "ABC", las permutaciones serían:
    ABC
    ACB
    BAC
    BCA
    CAB
    CBA