	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Reporte de práctica 3: Solución de problemas y Algoritmos.

Profesor(a): Oscar René Valdez Casillas

Asignatura: Fundamentos de Programación

Grupo: 21

No de Práctica(s): 3

Integrante(s): ***Santiago Durán Rendón***

Santiago Noriega Chiu

Jesús Ramírez Reyes

No. de lista o brigada: 01

Semestre: 2025-1

Fecha de entrega: 04/09/2024

Observaciones:

CALIFICACIÓN: _____

Índice

Resumen

Esta práctica se enfoca en la resolución de problemas mediante la elaboración de algoritmos eficientes. Se desarrollaron tres ejercicios que incluyen la identificación de conjuntos de entrada y salida, el diseño de algoritmos para la generación de la secuencia de Fibonacci, la gestión de ventas de boletos en un estadio, y el conteo de números primos menores a un número dado. Estos ejercicios permiten aplicar conceptos fundamentales del ciclo de vida del software, específicamente en las etapas de análisis y diseño, proporcionando una base sólida para el desarrollo de software efectivo y optimizado.

Introducción

En la programación, los algoritmos son fundamentales para resolver problemas de manera estructurada y eficiente. El diseño de un buen algoritmo implica un análisis cuidadoso de las entradas y salidas, así como la definición clara de los pasos necesarios para alcanzar el objetivo deseado. En esta práctica, se abordan problemas que son comunes en el ámbito de la ingeniería y la informática, como la generación de secuencias numéricas, la gestión de datos de ventas y el cálculo de números primos. Estos problemas se resuelven utilizando un enfoque sistemático que sigue las etapas del ciclo de vida del software, garantizando que los algoritmos sean correctos y eficientes.

Objetivo

Elaborar algoritmos correctos y eficientes en la solución de problemas siguiendo las etapas de Análisis y Diseño pertenecientes al Ciclo de vida del software.

Desarrollo de Contenidos:

Ejercicio 1.

Encuentre el algoritmo para calcular el término 10 de la secuencia de Fibonacci. Recuerde que los dos primeros números de la serie son 0 y 1. El resto se calcula con la suma de los dos números inmediatos que le preceden.

- **Conjunto de entrada:** Número de términos a generar (n).
- **Conjunto de salida:** Término 10 de la secuencia de Fibonacci.

Algoritmo:

1. Inicializar las variables enteras: a = 0, b = 1, c = 0
2. Do While 0 < n <= 10

- c = a + b
- a = b
- b = c
- n = n + 1
- imprimir el valor de “b”

3. Cuando n no sea igual a la condición dada imprimir mensaje de error.

Ejercicio 2.

En un estadio se tienen 5 tipos diferentes de localidades, las cuales se identifican por una clave numérica que es un valor comprendido entre 1 y 5. Los precios de cada localidad y los datos referentes a las ventas de boletos para el próximo juego son:

Clave 1 → Boleto 1	P1 → Precio boleto 1
Clave 2 → Boleto 2	P2 → Precio boleto2
Clave 3 → Boleto 3	P3 → Precio boleto 3
Clave 4 → Boleto 4	P4 → Precio boleto 4
Clave 5 → Boleto 5	P5 → Precio boleto 5

Algoritmo:

1. Leer los precios de los boletos P1, P2, P3, P4, P5.
2. Inicializar contadores para la cantidad de boletos vendidos de cada tipo.
3. Para cada venta:
    - Leer la clave del boleto y la cantidad vendida.
    - Calcular el importe de la venta: importe = precio * cantidad.
    - Incrementar el contador correspondiente para la cantidad de boletos vendidos.
    - Imprimir la clave, cantidad e importe de la venta.
4. Calcular e imprimir la recaudación total del estadio sumando los importes de todas las ventas.
5. Imprimir la cantidad total de boletos vendidos por cada tipo.

Algoritmo:

1. Inicialización de las variables:

P1, P2, P3, P4, P5 // Precios de los boletos

Cantidad1 = 0 // Cantidad de boletos vendidos para la Clave 1

Cantidad2 = 0 // Cantidad de boletos vendidos para la Clave 2

Cantidad3 = 0 // Cantidad de boletos vendidos para la Clave 3

Cantidad4 = 0 // Cantidad de boletos vendidos para la Clave 4

Cantidad5 = 0 // Cantidad de boletos vendidos para la Clave 5

RecaudacionTotal = 0 // Recaudación total del estadio

2. Leer P1, P2, P3, P4, P5
3. Inicializar la venta: Sí Venta=Sí:

   Leer la clave del tipo de boleto vendido: Leer Clave

   Leer la cantidad de boletos vendidos: Leer CantidadVendida

   Calcular el importe de la venta:

   Si Clave = 1 entonces Importe = P1 * CantidadVendida Cantidad1 = Cantidad1 + CantidadVendida

   Si Clave = 2 entonces Importe = P2 * CantidadVendida Cantidad2 = Cantidad2 + CantidadVendida

   Si Clave = 3 entonces Importe = P3 * CantidadVendida Cantidad3 = Cantidad3 + CantidadVendida

   Si Clave = 4 entonces Importe = P4 * CantidadVendida Cantidad4 = Cantidad4 + CantidadVendida

   Si Clave = 5 entonces Importe = P5 * CantidadVendida Cantidad5 = Cantidad5 + CantidadVendida

4. Imprimir la Clave, la Cantidad Vendida y el Importe de la venta:

   Imprimir “Clave”: Clave

   Imprimir “Cantidad vendida”: CantidadVendida

   Imprimir ""”Importe de la venta”: Importe

   Actualizar la recaudación total: RecaudacionTotal = RecaudacionTotal + Importe

   Preguntar si hay otra Venta: (Sí o No)

Sí Venta=No

5. Imprimir la cantidad de boletos vendidos de cada tipo:

   Imprimir "Cantidad de boletos vendidos de la Clave 1:", Cantidad1

   Imprimir "Cantidad de boletos vendidos de la Clave 2:", Cantidad2

   Imprimir "Cantidad de boletos vendidos de la Clave 3:", Cantidad3

   Imprimir "Cantidad de boletos vendidos de la Clave 4:", Cantidad4

   Imprimir "Cantidad de boletos vendidos de la Clave 5:", Cantidad5

6. Imprimir la recaudación total del estadio: Imprimir “Recaudación total del estadio”: RecaudacionTotal
7. Fin del algoritmo

Tabla 1: Ejemplo de entrada y salida para el algoritmo de boletos

Clave	Precio de Boleto	Cantidad Vendida	Importe de la venta
1	$50	100	$5000
2	$30	150	$4500
3	$20	50	$1000
4	$10	200	$2000
5	$5	300	$1500
		Recaudación total del estadio:	$14000

Ejercicio 3.

Un número positivo N es un número primo si los únicos enteros positivos que lo dividen son exactamente 1 y N. Indique la forma en que dado un número M, se obtenga y cuente todo los números primos menores a M.

1. Se requiere encontrar todos los números primos menores que M y cuantos hay.
2. Definir una lista de números *primos* para almacenar los números primos encontrados .
3. Definir una variable *contador* e iniciarla en 0 para llevar el conteo de números primos.
4. Para cada número *n* desde 2 hasta M -1 (o ambos) verificar si *n* es primo.

5. Para cada n hacer lo siguiente:
    - Inicializar una variable en *esPrimo* en *Verdadero*.
    - Para cada número de *i* desde 2 hasta la raíz cuadrada de *n* (redondeada hacia abajo), hacer lo siguiente:
        - ❖ Si *n* es divisible por *i* (es decir, *n % i == 0*), entonces:
            - ➢ Establecer *esPrimo* en *Falso* (porque *n* tiene un divisor diferente de 1 y *n*).
            - ➢ Salir del bucle de verificación.
6. Si después de la verificación *esPrimo* sigue siendo *Verdadero*, agregar *n* a la lista primos y aumentar el *contador* en 1.
7. Al finalizar el bucle de iteración, la lista *primos* contendrá todos los números primos menores que M.
8. La variable *contador* indicará la cantidad total de números primos menores a M.
9. Mostrar o devolver a lista *primos*.
10. Mostrar o devolver el valor de *contador*.

Ejemplo de Implementación de pseudocódigo.

```
función contarPrimosMenoresQue(M):

  primos = lista vacía

  contador = 0


  para n desde 2 hasta M-1:

    esPrimo = Verdadero


    para i desde 2 hasta raíz cuadrada(n):

      si n % i == 0:

        esPrimo = Falso

        romper el bucle


    si esPrimo:
```

agregar n a *primos*

contador = contador + 1

devolver (primos, contador)

Este algoritmo garantiza que todos los números primos menores a M sean encontrados y contados de manera eficiente

Conclusiones

1. ***Santiago Durán Rendón***: Se aplicaron técnicas de análisis y diseño de algoritmos para resolver problemas algunos problemas. La identificación de entradas y salidas, junto con el desarrollo de algoritmos claros y eficientes, permitió desarrollar desafíos como la generación de secuencias, la gestión de datos, y el cálculo de números primos.
2. ***Santiago Noriega Chiu***: Esta práctica nos ayudó para poder fortalecer nuestros conocimientos sobre la programación y aplicar la teoría así como el implemento de algunos cálculos simples. Estos ejercicios, de igual forma, son útiles para poder seguir aprendiendo y estar preparados para nuestro futuro, tanto académico como laboral
3. ***José de Jesús Ramírez Reyes***: El análisis y diseño son pasos esenciales para desarrollar algoritmos efectivos. El análisis nos ayuda a entender el problema y los requisitos, mientras que el diseño nos guía en la creación de soluciones estructuradas y eficientes. A través de estas etapas, podemos asegurar que nuestros algoritmos no solo resuelvan el problema de manera correcta, sino también de forma optimizada y adecuada a los requisitos del usuario.

Bibliografía

- Knuth, D. E. (1997). *The Art of Computer Programming, Volume 1: Fundamental Algorithms* (3rd ed.). Addison-Wesley.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.
- Sedgewick, R. (2011). *Algorithms* (4th ed.). Addison-Wesley.
- University of Cambridge. (2023). *Fibonacci Sequence*. Recuperado de https://www.maths.cam.ac.uk
- The GNU Project. (2023). *GNU C Library Reference Manual*. Recuperado de https://www.gnu.org/software/libc/manual/
- Universidad Nacional Autónoma de México (UNAM), (2017) *Números primos y compuestos.* *https://www.rua.unam.mx/portal/recursosficha/17661/n%C3%BAmeros-primos-y-compuestos*