Programación y Diseño Algorítmico
Práctica 1

1.1. Objetivos

1.1.1. Objetivo General

Esta práctica tiene como objetivo principal introducir al estudiante en los aspectos generales del ambiente del lenguaje de programación seleccionado, sus capacidades y sus características generales.

1.1.2. Objetivos Específicos

- Introducir al estudiante en el manejo del ambiente de trabajo del lenguaje de programación.
- Presentar los elementos básicos del lenguaje.
- Introducir algunos comandos de utilidad general.

1.2. Pre-requisitos

Es recomendable que el estudiante tenga un mínimo de destreza en el manejo del sistema operativo.

1.2.1. Actividades de Pre-Laboratorio

- Repasar y traer las fórmulas para calcular el área y el perímetro de un: círculo, cuadrado y triángulo.
- Investigar la fórmula para el cálculo de la suma de los n primeros términos de una progresión geométrica con razón r y primer término a_0.

1.3. Actividades de laboratorio

1.3.1. Primera parte

- El ambiente de trabajo del lenguaje de programación. Identificación de las ventanas. Espacio de trabajo, historia, ayuda, etc.

1.3.2. Segunda parte

- Escribir las instrucciones que deben introducirse en el editor para calcular el área y el perímetro de un círculo con radio r. Pruebe el programa para el caso en el que $r = 5$.
- Escribir las instrucciones que deben introducirse en el editor para calcular la suma de los primeros n términos de una progresión geométrica con razón r y primer término a_0. Pruebe para valores $r = 2,\ a_0 = 1/2,\ n = 10$.

1.4. Ejercicios propuestos

- Escribir las instrucciones que permitan calcular el área y el volumen de un cilindro de radio r y altura h. Pruebe para valores $r = 4,\ h = 15$.
- Escribir las instrucciones que permitan calcular el área y el perímetro de un rectángulo de lados $a = 14$ y $b = 8$.
- Escribir las instrucciones que permitan calcular el área de un triángulo de lados a, b, c. Use la fórmula:

$$Area = \sqrt{p(p-a)(p-b)(p-c)}$$

  donde $p = (a + b + c)/2$ (semiperímetro). Pruebe con los valores $a = 4$, $b = 5$, $c = 3$
- Escribir las instrucciones que permitan calcular el punto de intersección de las dos rectas en R^2. Pruebe con los valores

$$\begin{aligned} y &= -3x + 5 \\ y &= 2x + 7 \end{aligned}$$

- Escribir las instrucciones que permitan calcular las raíces de un polinomio de segundo grado. Pruebe con los valores:

$$3x^2 + -6x + 5$$
$$x^2 + 2x + 4$$

1.5. Evaluación

El estudiante deberá:

- Mostrar las actividades previas.
- Entregar las soluciones para los ejercicios propuestos, incluyendo los diseños algorítmicos, el código escrito en el lenguaje seleccionado y los resultados para cada caso.
- Presentar la solución de un ejercicio.

Programación y Diseño Algorítmico Práctica 2

2.1. Objetivos

2.1.1. Objetivo General

En esta práctica el estudiante utilizará expresiones lógicas en el programa de programación y aplicará las estructuras de selección para incorporar en el programa capacidades de decisión.

2.1.2. Objetivos Específicos

- Editar y almacenar programas.
- Generalizar las entradas y salidas de los programas usando las funciones específicas del lenguaje.
- Utilizar las estructuras de selección simple, doble y compuesta para resolver problemas específicos.

2.2. Actividades de Pre-Laboratorio

- Realice el diseño algorítmico de un programa que dada las longitudes de tres segmentos decida si ellas pueden formar un triángulo. Luego diga si es un triángulo equilátero, isósceles o escaleno.
  observacion: Para que tres segmentos formen un triángulo debe cumplirse que el mayor de ellos debe ser menor que la suma de los otro dos.
- Realice el diseño algorítmico de un programa que dados cuatro puntos en el plano, determine si son los vertices de un rectángulo cuyos lados son paralelos a los ejes.
- Realice el diseño algorítmico de un programa que reciba tres números a, b, c, luego los ordene tal que a tenga el menor valor y b el mayor
- Suponga que se introducen dos números. Que instrucciones pueden usar para que variable sign valga 1 si el producto es positivo y tome el valor -1 si el producto es negativo.
- Determine que se esta ejecutando con las siguientes instrucciones:

```
si (a>=200)
    imp <- 0.2*200
sino
    si (a>=150)
        imp <- 0.1*200
    sino
        imp <- 0
```

2.3. Actividades de Laboratorio

- Implementar el problema 1 del pre-laboratorio y probarlo con las siguientes ternas de datos:
  4,5 y 6
  1,2 y 8.
- Implementar el problema 2 del pre-laboratorio. Seleccione dos cuartetos de puntos en el plano y verifique la solución que da su programa.
- Implementar el problema 3 del pre-laboratorio con las siguientes ternas de datos:
  -3,-1 y 6
  4,2 y 0.

2.4. Ejercicios Propuestos

1. Escriba un programa que, reciba las coordenadas de un punto en el plano y el radio de una circunferencia centrada en el origen, para determinar si el punto está dentro de la circunferencia, entre la circunferencia y el cuadrado que la circunscribe o fuera de ese cuadrado.
   Pruebe su programa con:
   (-2.5,1), r:=2.6;
   (1.5,2.4), r:=1;
   (0.9,0.9), r:=0.9;
2. Escriba un programa que dados dos números enteros determine si uno de los dos es divisor del otro.
3. Escriba un programa que escriba los nombres de los días de las semana, en función de la entrada numérica para el día. Ademas, suponga que la semana comienza el domingo. Prueba con los valores de día igual a 3 y 5.

2.5. Evaluación

El estudiante deberá:

- Entregar las actividades previas.
- Entregar las soluciones para los ejercicios propuestos, incluyendo los algoritmos, el código y los resultados para cada caso.
- Presentar la solución de los ejercicios propuestos.

Programación y Diseño Algorítmico Práctica 3

3.1. Objetivos

3.1.1. Objetivo General

Comprender y aplicar los conceptos de las estructuras de repetición de condición implícita.

3.1.2. Objetivos Específicos

- Utilizar la estructura de repetición Repita Para.

3.2. Pre-requisitos

3.2.1. Conceptos básicos para integración numérica

El cálculo de la integral definida $\int_a^b f(x)dx$ es un problema que se plantea frecuentemente en matemática aplicada, física, ingeniería, economía y en problemas de modelado en otras disciplinas. Si $f(x)$ tiene primitiva podemos hacer este cálculo algebraicamente, pero si no tiene primitiva debemos emplear métodos numéricos. A continuación se presentan dos métodos cuya idea básica es remplazar $f(x)$ por un polinomio de interpolación.

Método de Trapecios: Utiliza un polinomio interpolatorio de primer grado.

Si dividimos el intervalo $[a, b]$ en n porciones iguales, y definimos $P(x)$ como el polinomio de primer grado que interpola a $f(x)$ en el intervalo $[x_i, x_{i+1}]$ el área bajo la curva $f(x)$ puede ser aproximada por la suma de las áreas de los n trapecios formados por los n polinomios construidos.

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} P(x)dx \qquad x_0 = a, x_n = b$$

$$P(x) = \frac{(x - x_{i+1})}{(x_i - x_{i+1})} f(x_i) + \frac{(x - x_i)}{(x_i - x_{i+1})} f(x_{i+1})$$

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{n-1} \frac{(b-a)}{2n} [f(x_i) + f(x_{i+1})]$$

Método de Simpson: Utiliza un polinomio interpolatorio de grado 2.

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} P^2(x)dx \qquad x_0 = a, x_n = b$$

$$P^2(x) = \frac{(x-x_m)(x-x_{i+1})}{(x_i-x_m)(x_i-x_{i+1})}f(x_i) + \frac{(x-x_i)(x-x_{i+1})}{(x_m-x_i)(x_m-x_{i+1})}f(x_m) + \frac{(x-x_i)(x-x_m)}{(x_{i+1}-x_i)(x_{i+1}-x_m)}f(x_{i+1})$$

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{n-1} \frac{(b-a)}{6n}\left[f(x_i) + 4f(x_m) + f(x_{i+1})\right]$$

3.3. Actividades Previas

1. Realice el diseño algorítmico de un programa que implemente el método del trapecio y el método de Simpson para la función $f(x) = \frac{1}{1-x}$. Calcule, algebraicamente, la integral para comparar los valores de las aproximaciones.

2. Realice el diseño algorítmico de un programa que calcule los primeros n cubos mediante la conjetura de Nicómaco:

   - Sume el primer impar: 1·
   - Sume los dos siguientes impares: $3 + 5 = 8$.
   - Sume los tres siguientes impares: $7 + 9 + 11 = 27$.

3. Realice el diseño algorítmico de un programa que calcule el cuadrado de un número entero sólo usando la operación suma. ¿Cómo calcularía el cubo de un número usando sólo sumas?

4. Realice el diseño algorítmico de un programa que calcule π usando la fórmula de Vietè (1593).

$$\frac{2}{\pi} = \frac{\sqrt{2}}{2}\frac{\sqrt{2+\sqrt{2}}}{2}\frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2}...$$

5. Realice el diseño algorítmico de un programa que calcule π usando el método recursivo de Borwein (1987)

$$x_0 = \sqrt{2}, \qquad x_n = \frac{1}{2}\left(\sqrt{x_{n-1}} + \frac{1}{\sqrt{x_{n-1}}}\right)$$
$$y_1 = \sqrt[4]{2}, \qquad y_n = \frac{y_{n-1}\sqrt{x_{n-1}} + \frac{1}{\sqrt{x_{n-1}}}}{y_{n-1}+1}, \qquad \pi \simeq \pi_n, n \to \infty$$
$$\pi_0 = 2 + \sqrt{2}, \qquad \pi_n = \pi_{n-1}\left(\frac{x_n+1}{y_n+1}\right)$$

3.4. Actividades de laboratorio

1. Implemente el problema 1 de las actividades previas y complete la siguiente tabla:

n	a	b	Exacto	Trapezoide	Sipmson	Error Trapezoide	Error Sipmson
6	1.1	2					
10	1.1	2					
6	2	3					
10	2	3					

2. Codifique el problema 2 de las actividades previas y pruebe con dos valores diferentes para n.

3. Codifique el problema 3 de las actividades previas y pruebe con cualquier número entero.

4. Implemente el problema 4 y 5 de las actividades previas y complete la siguiente tabla (use una precisión de 8 dígitos).

n	Vietè	Borwein	Otro método
4			
8			
16			

3.5. Ejercicios propuestos

1. Realice el diseño algorítmico y la codificación para un programa que muestre los primeros n términos de la serie de Fibonacci.

2. Realice el diseño algorítmico y la codificación para un programa que calcule la probabilidad de sacar un full de ases en el poker.

3. Realice el diseño algorítmico y la codificación para un programa que calcule la integral de la función $f(x) = \frac{1}{1-x}$ a través de la generación de n segmentos aleatorios dentro del intervalo $[a, b]$.

$$\int_a^b f(x) \approx \sum_{i=1}^{n} f(x_i), \quad x_i : \text{un valor aleatorio en el intervalo } [a, b], \quad x_i < x_{i+1}$$

.

3.6. Evaluación

El estudiante deberá:

- Mostrar las actividades previas.
- Entregar las soluciones para los ejercicios propuestos, incluyendo los resultados para cada caso.
- Presentar la solución de un ejercicio para la fecha fijada.

Programación y Diseño Algorítmico Práctica 4

4.1. Objetivos

4.1.1. Objetivo General

Comprender y aplicar los conceptos de las estructuras de repetición de condición explícita.

4.1.2. Objetivos Específicos

- Utilizar la estructura de repetición Mientras para resolver problemas que involucren el uso de estructuras de repetición de condición explícita.

4.2. Pre-requisitos

4.2.1. Actividades Previas

1. Suponga que estamos interesados en obtener el número entero n más pequeño que satisface la siguiente desigualdad:
$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \geq M$$
donde M es una variable de entrada. Realice el diseño algorítmico del programa que lea el valor de M más pequeño que satisface la desigualdad.

2. **Metodo de bisección** Este método se basa en el teorema de valor intermedio. En particular, si f es una función continua digamos $\mathbb{R} \to \mathbb{R}$ y a, b son valores dados tales que $f(a)f(b) < 0$, entonces definimos $c = \frac{a+b}{2}$. Si $f(c) = 0$, entonces terminamos. De lo contrario reemplazaremos a ó b con c manteniendo la diferencia de signos.
Supongamos que $f : \mathbb{R} \to \mathbb{R}$ es una function continua y que a, b son números tales que $f(a)f(b) < 0$ y $e < 0$, también dado, es el error que estamos dispuestos a tolerar y será el criterio de parada del algoritmo general

leer(a,b,e)
$aact \leftarrow a$
$bact \leftarrow b$
$cant \leftarrow (aact + bact)$
$cact \leftarrow cant/2$
Mientras $(|cact - cant| \geq e)$
  si $(f(cact) * f(bact) < 0)$ entonces
    $aact \leftarrow cact$
  sino
    $bact \leftarrow cact$
  $cant \leftarrow cact$

$bact \leftarrow (aact + bact)/2$
escribir($cact$)

Verifique el algoritmo y realice el diseño algorítmico de un programa para encontrar las raíces de la función $f(x) = \log(x) + x$.

3. **Método de Newton:** Para aplicar este método debemos suponer que la función f es diferenciable. la recta tangente a f en el punto $(x_0, f(x_0))$. Este método parte de una aproximacion inicial, x_0, y obtiene una aproximación mejor, x_1, dada por la fórmula:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}, \ \ donde \ \ f'(x_0) \neq 0$$

Este proceso se repite con el punto $(x_1, f(x_1))$, etc; obteniendo la recursión:

$$x_{n+1} = x_n - \frac{f(x_n)}{f(x_n)}, \ \ donde \ \ f'(x_0) \neq 0, n \geq 0 \ \ y \ \ x_0 \ \ es \ \ dado$$

Esta recursión define el **método de newton** y el procedimiento finaliza cuando $|x_{n+1} - x_n| < e$. Sin embargo, la convergencia no esta garantizada y el programa se puede "colgar".

Escriba el diseño algorítmico para encontrar las raíces de la función $f(x) = \log(x) + x$.

4.3. Actividades de Laboratorio

4.3.1. Primera Parte

- Escriba un programa para el siguiente algoritmo y determine su funcionamiento:

```
x <- 1
Mientras ((1+x)>1)
    x <- x/2
x <- 2*x
Escribir(1,'El valor más pequeño es: ',x)
```

4.3.2. Segunda Parte

- Codifique y pruebe los problemas de las actividades previas.

4.4. Ejercicios Propuestos

1. Realice el diseño algorítmico y la codificación de un programa que determine si un número n es o no primo.

2. Se dice que un número es perfecto si es igual a la suma de todos los divisores menores que él.

   Por definición, 1 es perfecto.

$$6 = 1 + 2 + 3.$$

$$28 = 1 + 2 + 4 + 7 + 14.$$

Realice el diseño algorítmico y la codificación de un programa que dado un número diga si es perfecto o no.

3. Modifique el código que generó para el problema 1 de las actividades previas de la **practica 3** para que permita determinar el número de particiones (n) mínimo tal que la aproximación de la integral difiera en menos del 0,01 % del valor exacto de la integral. (Escoja una función a la que pueda calcular la integral exacta para probar su programa).

4.5. Evaluación

El estudiante deberá:

- Mostrar las actividades previas.
- Entregar las soluciones los ejercicios propuestos, incluyendo los algoritmos, las instrucciones y los resultados para cada caso.
- Presentar,en la fecha fijada para la entrega la solución de un ejercicio.

Programación y Diseño Algorítmico Práctica 5

5.1. Objetivos

5.1.1. Objetivo General

Manejar los arreglos de datos usando los esquemas básicos del lenguaje para estas estructuras.

5.1.2. Objetivos Específicos

- Definir y manipular arreglos de datos, incluyendo las cadenas de caracteres.
- Aprender el esquema de acceso a datos dentro de un arreglo.

5.2. Pre-requisitos

5.2.1. Actividades previas

- Algebra de matrices: Realice el diseño algorítmico de los programas que permitan: Sumar dos matrices, Multiplicar dos matrices y multiplicar un vector por una matriz.
- Realice el diseño algorítmico de un programa que determine si una matriz es o no simétrica.
- Realice el diseño algorítmico de un programa que dada una matriz permita calcular la suma de cada una de sus filas y columnas, dejando dichos resultados en dos vectores, uno de las sumas de las filas y otro de las columnas.
- Una matriz es diagonal dominante si cada elemento de la diagonal principal es mayor (en valor absoluto) que la suma de los valores absolutos de todos los demás elementos de la misma fila o columna. Realice el diseño algorítmico de un programa que determine si una matriz es o no diagonal dominante.
- Realice el diseño algorítmico de un programa que lea una matriz de unos (1) y ceros (0) y luego permita calcular el número de unos y ceros en cada fila y en cada columna. Además, debe mostrar el porcentaje global de unos y ceros en la matriz.
- Elabore un conjunto de matrices de prueba para los problemas anteriores.

5.2.2. Actividades de laboratorio

Primera parte

- Definir un arreglo unidimensional de $1 \times n$ (vector fila) colocando los valores iniciales dados por el usuario
- Definir una matriz de $n \times n$ colocando valores iniciales aleatorios

Segunda parte

- Codifique los algoritmos para los problemas planteados en las actividades previas. Verifique el funcionamiento de cada programa con los matrices de prueba que trajo a la práctica.

5.3. Ejercicios propuestos

1. Se desea visualizar un cuadrado mágico de dimensión impar n. El usuario deberá elegir el valor de n. Un cuadrado mágico se compone de números enteros comprendidos entre 1 y n^2. La suma de los números que figuran en cada fila, columna y diagonal son iguales.

   8 1 6
   3 5 7
   4 9 2

   Un método de generación consiste en situar el 1 en el centro de la primera fila, el número siguiente en la casilla por encima y a la derecha, y así sucesivamente. El cuadrado es cíclico, la fila encima de la primera es de techo la última y la columna a la derecha de la última es la primera. En caso de que el número generado caiga en una casilla ocupada, se elige la casilla que se encuentre debajo del número que acaba de ser incluido.

2. **El Juego de la Vida** -una simulación de la supervivencia- Se juega en un arreglo cuadradobidimensional que representa una superficie plana que puede contener "criatura". En principio, la superficie está salpicada aleatoriamente de criaturas; es decir, cada celda de la matriz puede estar aleatoriamente vacía o bien ocupada por una criatura. A continuación, el programa entra en un ciclo en el que cada vuelta representa un "tic"del reloj ambiental; en cada "tic", las criaturas se propagan o mueren según dos leyes inmutables:

   a) Habrá un "nacimiento" en una celda vacía si por lo menos tres celdas vecinas contienen criaturas.

   b) En una celda ocupada, la criatura seguirá viviendo, al menos otro "tic", si y sólo si, las celdas vecinas contienen por lo menos dos pero no más de tres criaturas.

   Vecindad

$i-1, j-1$	$i-1, j$	$i-1, j+1$
$i, j-1$	i, j	$i, j+1$
$i+1, j-1$	$i+1, j$	$i+1, j+1$

   Así, el juego simula de manera "burda"los efectos de la congregación, la soledad y la sobrepoblación. Se desea jugar el Juego de la Vida para una retícula de dimensión $n \times n$.

   Para realizar el llenado inicial puede simular el "lanzamiento de una moneda", (para esto puede usar el generador de números aleatorios) Cabe señalar que cada celda tiene ocho vecinas: arriba, abajo, a la izquierda, a la derecha y en diagonal (ver tabla), con excepción de las celdas que están en el borde de la retícula. Cuide que su algoritmo no se salga de la retícula.

5.4. Evaluación

El estudiante deberá:

- Mostrar las actividades previas.
- Entregar las soluciones de los ejercicios propuestos, incluyendo los diseños algorítmicos, y el código.
- Presentar la solución de un ejercicio antes de la fecha fijada.

Programación y Diseño Algorítmico Práctica 6

6.1. Objetivos

6.1.1. Objetivo General

Definir, desarrollar y manipular subrutinas para implementar la solución a los problemas planteados.

6.1.2. Objetivos Específicos

- Aprender el esquema de definición y de llamado de subrutinas.
- Implementar el diseño algorítmico con subrutinas.

6.2. Pre-requisitos

6.2.1. Actividades previas

Escriba el diseño algorítmico para subrutinas que permitan:

1. Intercambiar los valores de dos variables.
2. Invierta simétricamente las posiciones de los valores en un arreglo unidimensional, usando la subrutina anterior. El elemento de la posición 1 se intercambiará con el de la posición n, el de la posición 2 con el de la $n-1$, y así sucesivamente.
3. Llenar una matriz cuadrada ($n \times n$) con valores binarios (del conjunto discreto [0,1]). A cada celda de la matriz se le asignará aleatoriamente el valor de 0 ó 1.
4. Determinar el número de vecinas de una celda en una matriz. La matriz es una matriz cuadrada ($n \times n$) donde el valor de cada celda pertenece al conjunto discreto [0,1]. Cada celda tiene ocho vecinas: arriba, abajo, a la izquierda, a la derecha y en diagonal; si la celda está en la última fila su vecina de abajo estará en la primera fila, si está en la última columna su vecina de la derecha estará en la primera columna, así sucesivamente. Se contarán como vecinas, únicamente las posiciones donde la celda almacene un uno (1).

6.3. Actividades de Laboratorio

- Escriba todas las subrutinas anteriores.
- Codifique un programa que lea los elementos de un arreglo unidimensional de longitud n y los intercambie usando la función cuyo del ejercicio 2 de las actividades propuestas.
- Codifique un programa que permita jugar al juego de la vida, usando las funciones cuyos algoritmos desarrolló para los ejercicios 3 y 4 de las actividades propuestas.

6.4. Ejercicios propuestos

1) Realice el diseño modular y la codificación para un programa que permita jugar a la **sopa de letras**. Usted tendrá una matriz cuadrada ($n \times n$) de caracteres (únicamente de letras minúsculas entre 'a' y 'z' ambas inclusive) y el programa leerá una palabra e indicará si se encuentra en la **sopa de letras** o no; en caso de ser encontrada la palabra deberá indicar la posición de inicio y la dirección.

   Ejemplo:

t	u	r	m	a	n	o
u	y	s	o	n	a	r
j	k	a	s	j	**s**	l
a	s	h	d	**u**	s	h
j	d	k	**s**	l	a	j
k	s	**e**	d	k	l	s
k	**j**	k	d	a	j	k

La palabra **jesus** está y comienza en la posición **7**,**2** con dirección **norte-este**.

6.5. Evaluación

El estudiante deberá entregar las soluciones para el ejercicio propuesto. Además, antes de finalizar la práctica deberá mostrar la solución para la segunda parte de las actividades de laboratorio.